Министерство образования и науки Российской Федерации
Российский фонд фундаментальных исследований
Южный научный центр Российской академии наук
Южный федеральный университет
НИИ многопроцессорных вычислительных систем
имени академика А.В. Каляева Южного федерального университета
АО «НИЦЭВТ»
ФГУП «НИИ «Квант»
ООО «НИЦ супер-ЭВМ и нейрокомпьютеров»
Журнал «Вестник компьютерных и информационных технологий»
Журнал «Известия ЮФУ. Технические науки»

СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

СКТ-2016

Материалы 4-й Всероссийской
научно-технической конференции

19 – 24 сентября 2016 г.
Дивноморское, Геленджик

ТОМ 1

Ростов-на-Дону
Издательство Южного федерального университета
2016

УДК004.272.43
ББК32.973
С 73

С 73 **Суперкомпьютерные технологии (СКТ-2016)** (19–24 сентября 2016 г.) Материалы 4-й Всероссийской научно-технической конференции : в 2 т. – Ростов-на-Дону : Издательство Южного федерального университета, 2016. – 210 с.
ISBN 978-5-9275-2038-1
ISBN 978-5-9275-2039-8 (Т. 1)

В первом томе материалов 4-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии» (СКТ-2016) представлены доклады, посвященные вопросам создания суперкомпьютеров, их архитектуре и аппаратной базе, а также разработке математического и программного обеспечения суперкомпьютеров.

Издание осуществлено при финансовой поддержке Российского фонда фундаментальных исследований, проект № 16-07-20640 г.

М $\frac{2404000000}{6KO(03)-2016}$ без объявл. ББК 32.973

ISBN 978-5-9275-2039-8 (Т. 1) УДК 004.272.43
ISBN 978-5-9275-2038-1 ББК 32.973

ПРЕДИСЛОВИЕ

Суперкомпьютерные технологии (СКТ) сегодня отнесены к числу важнейших направлений развития Российской Федерации. Потенциал суперкомпьютерной отрасли позволяет решить фундаментальные и прикладные проблемы в науке, промышленности, образовании, обороны, моделирование и анализ которых требует проведения масштабных вычислений.

Активное и эффективное применение суперкомпьютерных технологий служит локомотивом инновационного развития страны, способствуя не только глубокой модернизации промышленности, ликвидации технологического отставания России, но и обеспечения лидерства в глобальной экономической конкуренции.

Ряд стран (США, Япония, Китай, страны Европы) приняли специальные программы на федеральном уровне, проводят масштабные организационные мероприятия по консолидации усилий различных организаций и выделяют большие деньги на развитие суперкомпьютерных технологий. В Российской Федерации также реализуются федеральные целевые программы по развитию СКТ.

Однако, несмотря на это, Россия отстает от развитых стран Америки, Европы и Азии в области создания и применения стандартных кластерных суперкомпьютеров. В последнем списке TOP-500 (ноябрь 2015), содержащем сведения о наиболее производительных суперкомпьютерах мирового сообщества, наша страна представлена только семью суперкомпьютерами, самый мощный из которых Ломоносов-2 (МГУ) занимает 35-е место.

В то же время по ряду перспективных направлений развития суперкомпьютеров (суперкомпьютеры с реконфигурируемой архитектурой, гетервычислительные системы и ряд других) Россия находится на передовых рубежах.

В России существуют крупные научные школы, решающие те или иные проблемы создания перспективных суперкомпьютерных технологий. В частности, такие школы имеются в ФГУП «НИИ «Квант», НИИ системных исследований РАН, НИИ многопроцессорных вычислительных систем Южного федерального университета, Институте прикладной математики РАН, Научно-исследовательском вычислительном центре МГУ, ОАО «ИНЭУМ», ОАО «НИЦ ЭВТ», Институте программных систем РАН, Институте точной механики и вычислительной техники и ряде других организаций.

Поэтому обсуждение таких опережающих подходов к развитию суперкомпьютерных технологий является очень актуальным и будет основным предметом рассмотрения на конференции СКТ-2016, которая собрала воедино как разработчиков аппаратных средств высокопроизводительных вычислений, их системного и программного обеспечения, представляющих различные отечественные научные школы, так и представителей организаций – потенциальных пользователей суперкомпьютеров.

Целью научно-технической конференции «Суперкомпьютерные технологии» (СКТ-2016) является работа по консолидации суперкомпьютерного сообщества, а также обобщение и развитие накопленного за последние годы отечественного опыта разработки, создания, программирования и применения перспективных суперкомпьютерных технологий и выработка рекомендаций по дальнейшему использованию этого опыта с точки зрения задач модернизации экономики России.

Научная программа конференции объединяет широкий круг вопросов по следующим основным направлениям развития суперкомпьютерных технологий:

- *Принципы построения и архитектура суперкомпьютеров*
- *Математическое и программное обеспечение суперкомпьютеров*
- *Распределенные вычисления и системы*
- *Облачные вычисления*
- *Применение суперкомпьютерных технологий в науке, технике и промышленности*

Таким образом, проведение конференции СКТ-2016 будет способствовать повышению уровня фундаментальных и прикладных исследований в области суперЭВМ, проводимых в России; содействовать созданию высокопроизводительных вычислительных систем новых поколений, привлечению творческой научной молодёжи к проведению фундаментальных и прикладных исследований в области суперкомпьютерных систем; повышению уровня подготовки специалистов и кадров высшей квалификации в высших учебных заведениях по данному направлению; модернизации экономики России и превращению ее из сырьевой в инновационную.

Оргкомитет СКТ-2016

СОДЕРЖАНИЕ

РАЗДЕЛ 1. ПРИНЦИПЫ ПОСТРОЕНИЯ И АРХИТЕКТУРА СУПЕРКОМПЬЮТЕРОВ

РАЗДЕЛ 2. МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СУПЕРКОМПЬЮТЕРОВ

РАЗДЕЛ 1

ПРИНЦИПЫ ПОСТРОЕНИЯ И АРХИТЕКТУРА СУПЕРКОМПЬЮТЕРОВ

С.М. Абрамов[1], А.С. Амелькин[1], А.М. Цирлин[1], А.А. Чичковский[2]

СУПЕРКОМПЬЮТЕР ДОЛЖЕН КИПЕТЬ

[1] Институт программных систем им. А.К. Айламазяна РАН,
г. Переславль-Залесский,
abramov@botik.ru,
[2] Группа компаний «Сторус», г. Москва,
chaa@storus.ru

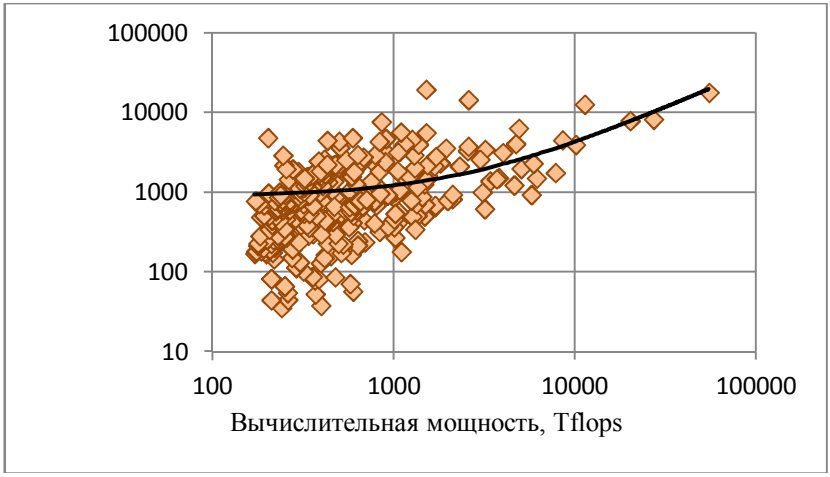
Введение

Сложные задачи построения полупроводниковых приборов, разработки фармацевтических препаратов, распознавания речи и образов, а также многие другие требуют все более мощных вычислительных комплексов [1]. Такие вычислительные устройства получили название суперкомпьютеров [2]. Развитие высокопроизводительной вычислительной техники идет по пути создания мощных компактных процессоров, развития высокоскоростных методов передачи данных, внедрения других технологических решений, повышающих вычислительную мощность. Это интенсивный путь развития, но есть и экстенсивный, когда увеличение вычислительной мощности происходит за счет увеличения числа процессоров в системе. И, если поиск новых технологий позволяет увеличивать мощность без существенного роста потребляемой системой энергии, то экстенсивный путь предполагает пропорциональное увеличение затрат энергии при увеличении количества процессоров. Анализ пятисот самых высокопроизводительных суперкомпьютеров мира [3] показывает, что между вычислительной мощностью и потребляемой электрической мощностью высокопроизводительных вычислительных комплексов наблюдается слабая корреляционная зависимость (коэффициент корреляции $r = 0{,}59$ – см. рисунок).

Показатели энергоэффективности вычислительных систем

Поскольку любая вычислительная машина не производит механической работы, вся потребляемая энергия расходуется на нагрев электронных компонентов, прежде всего, процессоров. При этом диапазон критических температур (граница, за которой процессор не будет рабо-

тать стабильно) для разных процессоров составляет от 60 до 100 °С. Поэтому важно поддерживать температуру процессоров, охлаждая их, а отобранную тепловую мощность рассеивать в окружающую среду.



Зависимость потребляемой электрической мощности от вычислительной мощности для пятисот самых высокопроизводительных суперкомпьютеров мира

Поэтому наряду с поиском более эффективных, с точки зрения тепловыделения, электронных компонентов, важно решить проблему эффективного охлаждения вычислительных комплексов. Что означает «эффективное охлаждение»? Для ответа на этот вопрос надо перечислить теплотехнические и конструктивные требования, которые предъявляются к вычислительным системам:

1) мощность, потребляемая системой охлаждения, должна быть минимальной, а температура окружающего воздуха, при которой обеспечивается поддержание температуры процессоров на уровне ниже критической и рассеивание заданной мощности в окружающую среду, должна быть максимальной;
2) вычислительная система должна быть компактной, а это означает отбор в единице объема как можно большей мощности, выделяемой процессорами суперкомпьютера;
3) поле температур в системе должно быть равномерным, несмотря на то, что охлаждаемые устройства выделяют теплоту совершенно не одинаково, могут быть расположены вдоль потока хладагента, интенсивность тепловыделения может изменяться во времени;

4) хладагент возвращается в систему охлаждения, отдав теплоту окружающему воздуху. Эта система не должна потреблять много энергии на создание потоков.

Здесь не перечислены требования, касающиеся безопасности, бесшумности системы, ее стоимости, гигиенические требования и пр., так как далее будут рассмотрены только теплофизические свойства систем охлаждения.

Каждое из требований определяет критерии энергоэффективности, поэтому для описания системы охлаждения мы получаем несколько не сводящихся друг к другу критериев качества. Требование минимальной мощности системы охлаждения формализуется тремя критериями: *PUE* (power usage effectiveness) [4], представляющим собой отношение общей мощности, которую потребляет вычислительная система, к мощности, потребляемой оборудованием, которое непосредственно задействовано в вычислительном процессе; энергоэффективность E, определяемая как отношение потребляемой мощности к скорости вычислительного процесса и измеряемая в кВт/Tflops; минимальная разность температур ΔT в ^{o}C между процессорами и окружающей средой. Компактность конструкции вычислительной системы может быть описана мощностью p_0, которую система охлаждения может отвести от 1 м^3 пространства, заполненного вычислительными устройствами. Равномерность поля температур описывается дисперсией D температур процессоров при полной загрузке вычислительной системы. Снижение энергии на организацию потоков связано с мощностью системы охлаждения. При этом надо учитывать, что система охлаждения может быть одноступенчатой (электроника→ воздух), двухступенчатой (электроника→ вода→ воздух) или трехступенчатой (электроника→ специальный хладагент→ вода→ воздух). Количество ступеней определяется требованиями к надежности вычислительного комплекса. При расчете многоступенчатой системы охлаждения общий перепад температур нужно оптимально распределить между ступенями.

Повышение эффективности системы охлаждения можно обеспечить за счет выбора хладагента и за счет организации системы охлаждения.

Теплофизические свойства хладагентов

Приведем теплофизические свойства хладагентов, во многом определяющие выбор системы.

Объемная теплоемкость воздуха равна 1,3 кДж/м^{3}К; для неэлектропроводной (диэлектрической) жидкости, использующейся в погружных системах охлаждения, она приблизительно в 1100 раз, а для воды в 3230 раз больше. Объемная теплоемкость определяет затраты энергии на перекачку потока хладагента, так как для того чтобы снять одну и ту же мощ-

ность, объем прокачиваемого воздуха должен быть в 1100 или 3230 раз больше, чем диэлектрической жидкости или воды соответственно.

Коэффициент теплоотдачи от металла к воздуху в Вт/м^{2}К зависит от скорости движения воздуха V. И, если эта скорость измеряется в м/с, он равен 5,6 +4V [5]. Тот же коэффициент от металла к жидкости равен 350+2100$\sqrt{V}$. При близком к реальности значении скорости V=0,1 м/с, коэффициент теплоотдачи при переходе от воздуха к жидкости возрастает в 110 раз. Если же жидкость еще и кипит при температуре контакта с охлаждаемым устройством, то коэффициент теплоотдачи равен приблизительно 4000, т.е. в 660 раз больше, чем для контакта с воздухом.

В реальных системах в зависимости от принятой конструкции термическое сопротивление между охлаждаемым устройством (процессором, например) и хладагентом определяется не только коэффициентом теплоотдачи, но и термическим сопротивлением термоинтерфейса, соединяющего процессор и радиатор, и теплопроводностью радиатора.

Перечисленные сведения, взятые из справочника Х. Кухлинга, носят приближенный характер, никак не учитываются свойства жидкости, ее вязкость, характер течения, шероховатость поверхности и др. Однако они дают представление о порядке величин и позволяют сравнить возможности систем охлаждения.

Сравнение систем охлаждения

1. Воздушные одноступенчатые системы, несмотря на температурный перепад между воздухом и нагревателем в 30 ^{o}C и больше, из-за малой теплоемкости воздуха требуют прокачивания значительных объемов, не гарантируют от локальных перегревов в силу неоднородности температурного поля хладагента. Как следствие они энергозатратны. Плотность расположения электронных устройств низкая, так как мал коэффициент теплоотдачи. Для машин большой мощности от воздушных систем охлаждения отказались.

2. Жидкостные двухступенчатые системы. В этих системах на первой ступени используется вода, на второй она охлаждается воздухом. Вода не должна контактировать с электроникой, поэтому ее пропускают внутри радиаторов по специальным герметичным каналам. Радиаторами могут служить массивные алюминиевые (латунные) пластины, к которым специальным теплопроводным клеем прикреплены охлаждаемые устройства. Охлаждающая вода протекает внутри пластин охлаждения по сложной системе каналов, обусловленной заданным распределением температур [6].

Практика выявила недостатки системы:

- дороговизна;
- малая надежность в связи с трудностью обеспечения с одной стороны герметичности подсоединения каналов охлаждения

к внешним трубопроводам, а с другой – возможности отключения платы от трубопровода для ее удаления из вычислительной системы, что может понадобиться, например, при замене платы;

- существенное термическое сопротивление за счет теплопередачи от охлаждаемого устройства к плате и неполное использование его поверхности;
- сильное увеличение веса системы из-за существенного веса массивных металлических охлаждающих пластин.

Отметим, что вода, используемая для охлаждения, должна быть тщательно очищена, так что никакие ее утечки недопустимы. Следовательно, на второй ступени охлаждения воды воздухом необходимо использовать энергоемкие устройства радиаторного типа.

3. Трехступенчатые системы с погружением охлаждаемых устройств в диэлектрическую жидкость (погружные). Такие системы выпускает в России компания ИММЕРС. Охлаждаемые устройства в суперкомпьютерах этой фирмы погружают в специальную диэлектрическую жидкость, которая не разъедает пластик и металл. При этом платы остаются стандартными, такими же, как в системах с воздушным охлаждением. Коэффициент теплоотдачи от охлаждаемых устройств (поверхности блоков памяти, радиаторов процессоров и блоков питания и др.) возрастает примерно в 100 раз, что позволяет повысить плотность их расположения. Насос для перекачивания диэлектрической жидкости тратит в сотни раз меньшую мощность, чем вентиляторы систем воздушного охлаждения.

На второй ступени можно либо охлаждать хладагент воздухом в системах радиаторного типа с присущими им значительными затратами мощности, либо охлаждать его водой, а воду, которая не требует высокой очистки, охлаждать на третьей ступени воздухом в градирне. Часть воды (примерно 0,5 %) при этом теряется, но затраты энергии в 5–6 раз меньше, чем при использовании радиаторов.

Погружные системы представляют собой существенный шаг вперед по сравнению с воздушными и водяными, но требуют тщательной проработки гидродинамики жидкости в ваннах с охлаждаемыми устройствами. Течение жидкости должно быть организовано так, чтобы не было каналов для ее прохождения без контакта с охлаждаемыми устройствами. Кроме того, поле скоростей жидкости при таком контакте должно быть максимально равномерным. Иначе неизбежны застойные зоны и местный перегрев.

Непростым вопросом является и конструкция радиаторов для таких систем: использование в системе погружного охлаждения стандартных радиаторов, разработанных для воздушного охлаждения, подобно

использованию геометрии корпуса и планера скоростного истребителя в качестве обводов подводной лодки.

Таким образом, несмотря на прорыв в эффективности процесса охлаждения, погружные системы пока еще не смогли достичь предельных значений показателей эффективности. Прежде всего это касается термостатирования вычислительной системы по всему занимаемому объему: распределение температур процессоров зависит от их расположения в системе.

4. Трехступенчатые системы с изменением фазового состояния хладагента (кипящий суперкомпьютер). Изменение фазового состояния хладагента (кипение в охлаждающей ванне, конденсация в системе отдачи теплоты в окружающую среду) решает многие проблемы охлаждения электронных устройств:

- в кипящей жидкости поле температур равномерно, и при постоянном давлении эта температура определяется только свойствами жидкости, что обеспечивает практически нулевую дисперсию температур процессоров при их одинаковой нагрузке;
- жидкость автоматически поступает в ту точку системы, где выделяется максимальный поток теплоты и откуда уносятся пузырьки пара;
- коэффициент теплоотдачи возрастает в 4 – 5 раз;
- отдача теплоты от насыщенного пара на второй ступени охлаждения происходит с максимально возможной интенсивностью, поэтому конденсатор может быть очень компактным;
- используемая на второй ступени в конденсаторе вода не требует очистки и может охлаждаться воздухом в градирне.

Система с кипящим суперкомпьютером реализована на вычислительной машине TSUBAME-KFC [7], разработанной в Токийском институте технологии и занявшей в 2014 г. первое место в рейтинге Green500. Конденсатор в ней представляет собой трубчатую решетку, внутри которой течет охлаждающая вода, а вся решетка помещена в паровое пространство охлаждающей ванны.

Основная трудность в создании кипящего суперкомпьютера состоит в выборе подходящей диэлектрической жидкости с температурой кипения порядка 50–55 ^{o}C и рациональной конструкции конденсатора.

5. Комбинированные системы охлаждения. Перечисленные принципы охлаждения электронных устройств могут быть использованы комплексно. При этом охлаждаемые устройства, характеризующиеся высокой плотностью теплового потока, могут быть погружены в емкости с кипящим хладагентом, а сами эти емкости, прежде всего их паровая часть, интенсивно охлаждаться водой. При этом высокий коэффи-

циент теплоотдачи может позволить обойтись без металлических радиаторов. Емкость с кипящей и конденсирующейся жидкостью образует «кипящий радиатор», который в отличие от обычных радиаторов не требует закрепления на нем охлаждаемого устройства и не создает дополнительного теплового сопротивления.

Первый в России вычислительный кластер, система охлаждения которого основана на фазовом переходе, разработан в Институте программных систем имени А.К. Айламазяна РАН. Проведены испытания работоспособности «кипящей» системы охлаждения, в ходе которых показана возможность термостатирования процессоров кластера в условиях их переменной загрузки. Это обеспечивается за счет постоянной температуры охлаждающей жидкости. Режим работы системы поддерживается на уровне пузырькового кипения, что соответствует максимальному коэффициенту теплопередачи и позволяет снять все радиаторы с вычислительных плат, а значит, значительно увеличить плотность компоновки вычислительных кластеров. Отсутствие шума, характерное для всех погружных систем охлаждения, термостатирование процессоров, высокая плотность компоновки и предельно высокая энергоэффективность (PUE < 1,01) – вот ключевые характеристики погружных систем охлаждения с фазовым переходом.

1. Сверхсложные вычислительные задачи, решаемые на суперкомпьютерах. Лаборатория параллельных информационных технологий Научно-исследовательского вычислительного центра Московского государственного университета имени М.В. Ломоносова. http://parallel.ru/research/challenges.html.
2. *Hoffman A. R.*; et al. (1990). Supercomputers: directions in technology and applications. National Academies. – P. 35 – 47.
3. Top500 Lists November 2014. http://www.top500.org/lists/2014/11/.
4. Коварный PUE. Хабрахабр. http://habrahabr.ru/company/ua-hosting/blog/244603/.
5. *Кухлинг Х.* Справочник по физике. – М.: Физматлит, 1975.
6. *Абрамов С.М., Заднепровский В.Ф., Шмелев А.Б., Московский А.А.* Супер-ЭВМ ряда 4 семейства СКИФ: штурм вершины суперкомпьютерных технологий // Тр. Междунар. науч. конф. «Параллельные вычислительные технологии (ПаВТ'2009)». – Нижний Новгород: Изд-во Нижегородского государственного университета имени Н.И. Лобачевского, 2009. – С. 5 – 16.
7. Heterogeneous Systems Dominate the Green500. HPC, November 20, 2013. http://www.hpcwire.com/off-the-wire/two-brains-better-one-heterogeneous-systems-dominate-green500/.

С.С. Анцыферов, А.С. Сигов, К.Е. Русанов

НАНО-СУПЕРКОМПЬЮТЕР

Московский технологический университет (МИРЭА), г. Москва,
c_standard@fel.mirea.ru

Введение

Суперкомпьютеры (СК) являются самыми мощными в мире по производительности и объему памяти вычислительными системами. Вместе с тем, это достаточно громоздкие и дорогостоящие системы. Так, например, китайский суперкомпьютер «Milky Way-2", занимающий одно из ведущих место в мире среди пятисот самых быстрых суперкомпьютеров (2,5 PF), весит более 150 т, включает более сотни шкафов, имеет несколько тысяч процессоров и видеочипов, стоит около 100 млн долл. Усилия разработчиков СК направлены на повышение быстродействия, емкости памяти, надежности, функциональности и технологичности производства при одновременном снижении материалоемкости, энергопотребления, себестоимости, сложности пользования и эксплуатации. Основу стратегии совершенствования составляет процесс постоянной миниатюризации элементной базы СК, создаваемой в настоящее время с помощью «кремниевых» технологий.

Ограничения «кремниевых» технологий

Существуют фундаментальные ограничения дальнейшего уменьшения размеров БИС: *термодинамические* – обусловлены конечной температурой компонентов, условиями теплообмена, нагреванием, как вследствие протекания тока (джоулевым тепловыделением и условиями теплоотвода), так и увеличения тактовой частоты. Все это приводит к росту энтропии и потере информации в системе; *электродинамические* – вызваны инерционностью емкостных и индуктивных компонентов в схемах, что препятствует быстрому изменению напряжений и токов при переходе от одного состояния к другому, в частности, при работе логических ключей в микропроцессоре или ячеек динамической памяти. Дополнительные ограничения на быстродействие накладывают конечная скорость распространения электромагнитных волн, движения носителей заряда, перемагничивание ферромагнетиков, переполяризация диэлектриков; *квантово-механические* – проявляются при уменьшении характерных размеров компонентов до атомарных масштабов. В этом случае начинает становиться заметной атомная и электронная дискретность в явлениях переноса и взаимодействия частиц. С уменьшением толщины пленки оксида кремния до 1 нм и менее (обычная ее толщина – несколько нанометров),

обладающей высокими диэлектрическими свойствами и выполняющей роль изолятора между отдельными элементами БИС, начинают сказываться квантовые эффекты туннелирования, приводящие к резкому увеличению тока утечки. Одним из узких мест на пути дальнейшей интеграции и миниатюризации БИС является литография со все большим разрешением. Использование наиболее удобного и освоенного оптического способа литографии предопределяет физический предел миниатюризации до нескольких десятков нанометров при экспонировании резиста коротковолновым ультрафиолетовым источником. Для дальнейшего повышения разрешения необходимо применять или более жесткое излучение (рентгеновское, электронное, ионное) или переходить к альтернативным технологиям. Существуют также проблемы при создании систем долговременной памяти, позволяющих хранить информацию в течение длительного срока без обновления и энергопотребления. Так, в рамках существующей концепции, предел миниатюризации записи на магнитном диске практически уже достигнут – вклад одного зерна размером ~ 10 нм в общую намагниченность области, занимаемой битом, измеряется долями процента. Однако поддерживать это соотношение по мере уменьшения размеров бита можно лишь до определенных пределов, поскольку устойчивость намагниченности к термическим флуктуациям падает с уменьшением размеров зерна. Тем не менее, не прекращаются поиски путей совершенствования «кремниевой» технологии.

Совершенствование «кремниевой технологии»

В настоящее время ведутся интенсивные разработки, направленные на преодоление указанных ограничений. Разрабатываются системы долговременной памяти, основанные на локальных изменениях фазового состояния носителя сфокусированным лазерным пучком, сохранении полученной фазы как угодно долго, обнаружении и считывании записанного бита неограниченное число раз без разрушения информации и при необходимости стирания ее, т.е. возвращения материала в исходное фазовое состояние. Наиболее удобный и освоенный к настоящему времени фазовый переход «кристаллическое состояние – аморфное состояние» уже используется в перезаписываемых компакт-дисках формата DVD. Увеличение быстродействия устройств достигается путем использования бинарных полупроводниковых соединений, характеризующихся высокой подвижностью электронов (In As, In Sb), значительно превышающей (в 30… 50 раз) их подвижность в кремнии. Кроме того, ведутся разработки устройств с использованием напряженного, упруго деформированного кремния. Компания Intel уже применяет его в своих серийных чипах для увеличения скорости переключения ячеек. Деформация решетки кремния всего на 1 % дает увеличение скорости пере-

ключения в полевом транзисторе до 20 % при увеличении себестоимости транзистора всего на 2 %. Показательными являются достижения в области накопителей информации на магнитных носителях. Использование эффекта гигантской магниторезистивности позволило существенно увеличить объем энергонезависимой магнитной памяти. Американская компания BeSang Inc ведет разработки первой в мире кремниевой нанопамяти по трехмерной архитектуре. Активно ведется не только разработка, но и коммерческий выпуск энергонезависимых запоминающих устройств (чипов) с ферромагнитными миниатюрными ячейками (~2,5 нм). Такие устройства являются важным компонентом при разработке реконфигурируемых логических элементов. Кроме того, создаются схемы переключения транзисторов (в современных микропроцессорах транзисторы как бы жестко «прошиты») для быстрого реконфигурирования различных систем.

В России проводится ряд перспективных исследований. Научно-исследовательский институт физических проблем им. Ф.В. Лукина ведет разработку однолитографических технологий. Данная разработка позволит стабилизировать технологические операции, обеспечить хорошую воспроизводимость предельно минимальных размеров и минимизацию привносимых дефектов. Исследование технологии создания нитевидных полупроводниковых нанокристаллов проводится в Санкт-Петербургском физико-технологическом научно-образовательном центре РАН. Работа направлена на создание новых полупроводниковых наноматериалов и наносистем с контролируемыми свойствами, которые могут использоваться в производстве полевых транзисторов. Центральный научно-исследовательский технологический институт "Техномаш" разрабатывает наноструктурированные материалы молекулярной фотоники и слоистых структур на их основе для оптической нейросетевой обработки информации. Применение этих материалов позволит создавать трехмерные функциональные структуры, а также информационные системы с параллельной обработкой информации, на несколько порядков превосходящие по быстродействию традиционные системы. Исследования магнитного состояния составных наноструктур методами нейтронного отражения и рассеяния проводит Объединенный институт ядерных исследований. Установлено, что составные наноструктуры могут быть использованы для создания логических элементов с четырьмя состояниями, что позволит значительно увеличить емкость запоминающих устройств. В Воронежском государственном университете проводятся работы по исследованию наноструктур для электронной техники нового поколения, а также изучению свойств наноразмерных гетерогенных систем, структур и нелинейных материалов. Исследуются нанокристаллы, функциональные материалы на основе слоистых и наногранулированных структур, материалы с

поверхностями, модифицированными малоатомными кластерами и наночастицами. Разработаны образцы релаксорных сегнетоэлектриков, технология формирования в тонкопленочных сегнетоэлектриках нанодоменов с размерами, обеспечивающими возможность создания перепрограммируемых энергонезависимых запоминающих устройств с большой емкостью. Институт радиотехники и электроники им. В.А. Котельникова РАН занимается фундаментальными исследованиями в области разработки одномерных структур на основе полупроводниковых гетероструктур GaAs / $AlGaA_5$ с двумерным электронным газом и пластин кремния на изоляторе. Исследуются свойства квазиодномерных наноматериалов, содержащих проводящие металлоцепочки атомных размеров. Результаты исследований служат технологической основой для сверхплотной записи и хранения информации. Технический университет НПК "Технологический центр" МИЭТ занимается созданием электронных приборов на базе гетероструктур, в частности, транзисторов с высокой подвижностью электронов, туннельно-резонансных диодов, гетероструктурных биполярных транзисторов. НИЦ "Курчатовский институт" осуществляет разработку голографической памяти сверхвысокой емкости. Вместе с тем, многие специалисты считают, что революционные преобразования информационной суперкомпьютерной техники произойдут за пределами «кремниевой» парадигмы.

Развитие «некремниевой» технологии

Радикальная стратегия развития «некремниевой» электроники отказывается от использования кремния в качестве основы интегральных схем и предполагает разработку новых нанотехнологий. Исследователям и технологам совершенно очевидными представляются огромные возможности промышленного применения углеродных нанотрубок, представляющих собой свернутые в цилиндр наносетки, состоящие из шестиугольных модулей на основе атомов углерода. Эти трубки обладают уникальными электрическими и магнитными характеристиками. Трубки обладают исключительно малым диаметром (несколько нанометров), их архитектура может иметь различную конфигурацию, что зависит от функциональных особенностей их применения. Например, туннельные углеродные нанотрубки могут использоваться в транзисторах, а трубки на основе структур с гетерогенными переходами или так называемые структуры «нейронного древа», – при создании когнитивных систем, моделировании нейронных сетей мозга и искусственного интеллекта. Нанотрубки со свойствами полупроводников могут использоваться в полевых транзисторах подобно кремнию. При этом роль регулируемого проводящего канала играет нанотрубка, а роль изолированного затвора – кремниевая подложка с тонкой пленкой оксида на поверхности. Изменение напряжения на затворе от +6 до –2 В меняет величи-

ну проводимости канала почти в 10^5 раз. За несколько лет развития (впервые возможность создания транзистора на нанотрубке была продемонстрирована в 1998 г.) удалось значительно улучшить характеристики нанотрубчатых полевых транзисторов и приблизить их к таковым у лучших «кремниевых» при значительно меньших размерах. На этих же принципах удалось построить экспериментальные логические устройства и ячейки памяти. Так, группой американских специалистов в качестве ячейки памяти было предложено использовать короткую, закрытую с двух сторон трубку, с размещенной в ней молекулой фуллерена C_{60}. Силы Ван-дер-Ваальса между молекулой и трубкой нарастают вблизи концов нанотрубки, что приводит к возникновению двухъямного потенциала. Одному крайнему положению молекулы можно приписать логический «0», а другому – логическую «1». Переключение между этими состояниями обеспечивается размещением атома металла в полость молекулы фуллерена. В результате ионизации металла, атом приобретает заряд. Приложение электрического поля вдоль оси трубки приводит к перемещению молекулы из одного крайнего положения в другое. Отключение поля не приводит к изменению достигнутого состояния, поскольку молекула находится в одной из двух потенциальных ям. В результате реализуется долговременная память без энергопотребления. Исследователи из Гарвардского университета предложили конструкцию запоминающей матрицы, в которой два ряда нанотрубок расположены под углом 90^o в параллельных плоскостях, разделенных зазором 1 – 2 нм. Такая система имеет двухъямный энергетический профиль, который создается силами упругости в самих трубках и ван-дер-ваальсовыми силами притяжения между ними. Приложение разности потенциалов к любой паре трубок, принадлежащим разным слоям, т.е. к выбранному узлу матрицы, вызывает изгиб верхней трубки и ее притяжение к нижней. При напряжении порядка единиц вольт происходит касание трубок, и сопротивление между ними падает на несколько порядков, что легко обнаруживается соответствующей электроникой. Силы Ван-дер-Ваальса удерживают трубки в контакте и после снятия напряжения, что обеспечивает энергонезависимость памяти. Стирание запомненного бита информации осуществляется приложением напряжения одинаковой полярности к трубкам данного узла, что приводит к расцеплению трубок и возвращению ячейки памяти в исходное состояние. Результаты исследований показывают, что данная технология позволяет создавать матрицы динамической памяти с размерами ячеек 5×5 нм, плотностью записи $\sim 10^{12}$ бит/см^2 и быстродействием ~100 ГГц. Показана возможность создания подобной запоминающей матрицы на нанотрубках, но с внутренним изолирующим слоем из нитрида кремния, содержащим глубокие ловушки электронов. Планируется создание устройств, содержащих до 400 млрд нанотрубок в одном $мм^2$, что существенным образом повлияет на качество соответствующей продукции электрон-

ной промышленности. Технологическими исследованиями в области разработки нанотрубок и их применением занимаются такие глобальные корпорации и научные организации, как IBM, Intel, NASA, NEC в США, Samsung и Shova Denko Com – panies в Японии, Институт М. Планка в Германии, НИЦ «Курчатовский институт» в России и др.

Итак, одно из основных преимуществ электроники на нанотрубках – плотность упаковки элементов, недостижимая в кремниевой технологии, при вполне приемлемых электрических характеристиках. В настоящее время осуществляется переход от лабораторных образцов к массовой технологии: созданы способы выращивания и управления ростом нанотрубок, их сортировки и встраивания в заданные конфигурации. Вместе с тем, сдерживающий фактор массового использования нанотрубок – отсутствие технологии выращивания их в больших количествах и простых способов соединения в необходимые электрические схемы. Большинство описанных в публикациях устройств на нанотрубках создано в единичных экземплярах в лабораторных условиях ценой больших затрат времени и труда. Кроме того, их характеристики не отличаются хорошей воспроизводимостью и надежностью. Таким образом, на пути к массовому применению безкремниевых технологий предстоит, по-видимому, преодолеть еще немало препятствий технического характера.

Необходимо также отметить, что ключом к успешному развитию нано-суперкомпьютерной технологии является стандартизация. Международная организация по стандартизации (ISO) и Международная электротехническая комиссия (IEC) совместно разработали, в качестве предшественника Международного стандарта, объемные Технические условия для терминологии в области нанотехнологии (ISO/TS 80 004). Сюда вошли особые термины и определения, касающиеся нанообъектов и наноструктурированных материалов, углеродных нанообъектов, нанометрологии, нанопроизводственных процессов.

Выводы

В настоящее время возможности «кремниевых» технологий до конца еще не исчерпаны, и при наличии больших производственных мощностей, отлаженного производства, специалистов, инфраструктуры, разогретых рынков сбыта это направление еще долго будет занимать на рынке доминирующие позиции.

Освоение нанометрового диапазона потребует создания принципиально новых физических основ и технологий производства элементной базы суперкомпьютеров, которые в общих чертах просматриваются уже сейчас.

1. http://www.nsf. gov/about/congress/106/hs- beyondsilicon.jsp
2. http://www. nsf. gov/news/overviews/physics/ physics- go2.jsp

И.В. Ашарина

УМЕНЬШЕНИЕ ВРЕМЕННОЙ ИЗБЫТОЧНОСТИ ПРИ ПАРАЛЛЕЛЬНОМ ОТКАЗОУСТОЙЧИВОМ РЕШЕНИИ СОВОКУПНОСТИ ВЗАИМОДЕЙСТВУЮЩИХ ЦЕЛЕВЫХ ЗАДАЧ

АО «НИИ «Субмикрон», НИУ МИЭТ, г. Зеленоград,
asharinairina@mail.ru

Ключевой проблемой отказоустойчивого выполнения целевой задачи в многомашинной вычислительной системе (МВС) при реализации в системе с динамической избыточностью [1] является взаимное информационное согласование (ВИС), обеспечивающее согласованность системной информации во всех цифровых вычислительных машинах (ЦВМ) системы в условиях возникновения допустимых неисправностей. Предложены подходы к снижению временных затрат на процесс ВИС в МВС заданной структуры, представляющих временную избыточность алгоритма ВИС.

Рассматриваются замкнутые [2] МВС, имеющие особенности [3]:

1) большое число автономных взаимосвязанных ЦВМ без общей памяти;
2) высокая степень распределенности управления среди ЦВМ при отсутствии централизованного управляющего органа;
3) обеспечение достоверности результата репликацией решаемых задач.

Графовая модель исходной МВС-ориентированный граф (орграф) системы, отображающий ее структуру, где вершины, пронумерованные в диапазоне 1, 2,..., соответствуют ЦВМ системы. Симплексный межмашинный канал связи изображается дугой, исходящей из соответствующей вершины, дуплексный канал связи (пара разнонаправленных псевдосимплексных каналов) – парой разнонаправленных дуг (или двунаправленной стрелкой). Рассматриваются МВС без дублированных каналов связи, поэтому их орграфы не имеют параллельных дуг. Полным орграфом называется орграф, имеющий пару разнонаправленных дуг между любыми двумя вершинами. Орграфы являются гомеоморфными, если существуют их изоморфные подразбиения, образующиеся в результате подразбиения дуг [4].

Существует ряд методов по решению проблемы ВИС в отдельном полносвязном комплексе согласования (подсистеме, удовлетворяющей определенным структурным требованиям) [5, 6], основанных на многораундных обменах сообщениями между ЦВМ комплекса с последующим вычислением вектора согласованных значений всех ЦВМ комплекса (ВСЗК) в каждой его ЦВМ [7].

Воспользуемся методами построения алгоритма ВИС для однокомплексных неполносвязных МВС [7, 8], в которых вся система явля-

ется комплексом. Эти методы основаны на том, что в системном орграфе T комплекса имеется орподграф ${}_{T}H$, гомеоморфный полному орграфу M, с количеством вершин более 3μ (μ – отказоустойчивость, т.е. допустимое число неисправных ЦВМ в МВС). Вершины орподграфа ${}_{T}H$, взаимно однозначно соответствующие вершинам орграфа M, называются основными и составляют множество O. Остальные вершины орграфа T называются неосновными и составляют множество N. Наличие такого орподграфа ${}_{T}H$ является первым из достаточных условий достижения ВИС в орграфе T. Второе достаточное условие: для каждой его неосновной вершины существуют исходящие из этой вершины и далее непересекающиеся пути не менее чем к $2\mu+1$ конечным основным вершинам. Третье условие: для каждой его неосновной вершины существуют входящие и ранее не пересекающиеся пути не менее чем от $2\mu+1$ начальных основных вершин [9].

Передача сообщения из p-й ЦВМ-передатчика в соседнюю q-ю ЦВМ-приемник осуществляется по симплексному каналу связи и занимает один временной квант – неделимый далее временной отрезок с синхронным началом и равной длительностью во всех ЦВМ системы, в течение которого по крайней мере из одной ЦВМ системы осуществляется передача сообщения(й) в одну (или несколько) соседнюю(их) ЦВМ и необходимая обработка полученного(ых) сообщения(й). Посылка сообщения из r-й ЦВМ-источника в s-ю ЦВМ-получатель по некоторому пути посылки выполняется посредством последовательности передач этого сообщения между соседними ЦВМ в этом пути [9] и осуществляется за несколько квантов. Каждому такому возможному пути посылки информации соответствует некоторый путь в орграфе МВС и наоборот.

В докладе рассматриваются системы с выделенными неполносвязными комплексами, для которых [8, 9] строится алгоритм ВИС для неполносвязной системы (ВИСНС), содержащий пять этапов.

1. Посылка копий согласуемого значения каждой неосновной ЦВМ комплекса в выбранное для нее подмножество основных ЦВМ.
2. Взаимообмен согласуемыми значениями между всеми основными ЦВМ комплекса и вычисление в каждой из них ВСЗК (этап ВИС основных вершин – ВИСОВ).
3. Посылка копий ВСЗК в каждую неосновную ЦВМ комплекса из определенного для нее подмножества неосновных ЦВМ и вычисления в ней ВСЗК.
4. Посылка ВСЗК комплекса-источника в ЦВМ комплекса-получателя.
5. Вычисление вектора согласованных значений системы (ВСЗС).

Выполнение алгоритма ВИСНС, включающее многократный взаимообмен сообщениями между всеми ЦВМ системы с промежуточным индивидуальным преобразованием полученной информации в каждой ЦВМ, требует определенной синхронности их действий. Естественной необходимостью алгоритма ВИСНС является высший уровень синхронизации по началу и концу его выполнения (синхронизация по началу и концу ВИСНС). Для полносвязных систем в [5, 6] и их алгоритмов ВИС предлагается также синхронизация более низкого уровня по началу и концу каждого из раундов взаимообмена (пораундная синхронизация), где раунд с синхронизированным началом во всех ЦВМ системы включает передачу сформированного сообщения из каждой ЦВМ системы во все остальные ее ЦВМ и последующую обработку полученных в этом раунде сообщений с формированием сообщения для следующего раунда, если таковой должен быть.

В алгоритмах ВИСНС [7, 8] для обеспечения синхронности индивидуальных действий различных ЦВМ системы введена синхронизация на уровне кванта. Кроме того, с целью упрощения методов построения алгоритмов ВИСНС, может вводиться синхронизация на уровне начала и окончания его внутренних этапов. Квантовая синхронизация является синхронизацией самого нижнего уровня, на основе которой строятся синхронизации более высоких уровней. Для алгоритмов ВИСНС возможны следующие сочетания уровней синхронизации (от высшего к низшему): 1) по началу и концу ВИСНС, по этапам его выполнения, по раундам взаимообмена на втором этапе, по квантам (переход к следующему раунду (этапу) возможен только после истечения количества квантов, предназначенного для полного завершения этого раунда (этапа) в случае отсутствия неисправностей в системе); 2) по началу и концу ВИСНС, по этапам его выполнения, по квантам; 3) по началу и концу ВИСНС, по квантам.

Алгоритмическая сложность метода построения распределенного алгоритма ВИС для рассматриваемых систем возрастает от первого к третьему сочетанию уровней синхронизации. В терминах передачи сформированных сообщений эти сочетания уровней синхронизации можно трактовать так: сформированное в текущем раунде (этапе) сообщение, предназначенное для передачи в следующем раунде (этапе), передается только в начале следующего раунда (этапа). Для третьего сочетания трактовка имеет следующий вид: сообщение, предназначенное для последующей передачи и подготовленное полностью в текущем кванте, передается по назначению в следующем кванте.

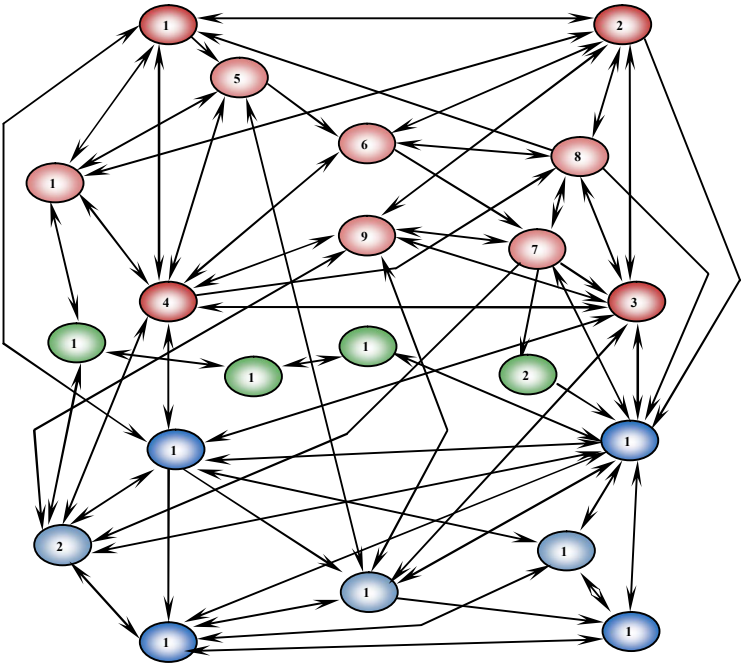
Вычислим время выполнения ВИСНС для двухкомплексной МВС, показанной на рисунке, считая отказоустойчивость первого комплекса $\mu_1=1$, второго комплекса – $\mu_2=1$ и среды межкомплексной посылки –

$\mu_{1\leftrightarrow 2}$=1. Считаем термины "вершина" и "ЦВМ" синонимами. Основными вершинами комплекса 1 являются вершины из множества $O_1 = \{1, 2, 3, 4\}$, комплекса 2 – $O_2^1 = \{11, 12, 13, 14\}$. Неосновными вершинами комплексов 1 и 2 соответственно являются вершины из множеств $N_1 = \{5,6,7,8,9,10\}$ и $N_2 = \{17,18,20\}$, каждая из которых передает свое согласуемое значение не менее чем в $2\mu_1$+1 ($2\mu_2$+1) основных вершин соответственно первого и второго комплекса на первом этапе согласования; остальные вершины МВС составляют среду межкомплексной посылки. В качестве примера рассмотрим передачу согласуемого значения неосновной 5-й ЦВМ в $2\mu_1$+1 основных ЦВМ комплекса 1. Для этого строятся непустые выражения в дизъюнктивной нормальной форме (ДНФ) исходящего пучка [9]:

$$SOP\_DNF(5, O_1^1, B_1^1 \setminus O_1^1) = 5\cdot 10\cdot 1\cdot 5\cdot 6\cdot 7\cdot 3\cdot 5\cdot 4 \vee \ldots .$$

Терм $5\cdot 10\cdot 1\cdot 5\cdot 6\cdot 7\cdot 3\cdot 5\cdot 4$ соответствует путям посылки сообщения из 5-й ЦВМ в ЦВМ соответственно 1-, 3-, 4-ю. Процесс ВИСОВ определяется построением непустых ДНФ входящих пучков:

$$SRP\_DNF(O_1^1 \setminus \{1\}, \widetilde{S}_1 \setminus O_1^1) = 2\cdot 1\cdot 3\cdot 8\cdot 1\cdot 4\cdot 1 \vee 2\cdot 1\cdot 3\cdot 8\cdot 1\cdot 4\cdot 5\cdot 10\cdot 1 \vee \ldots$$



Пример орграфовой модели МВС

Здесь каждый терм определяет пучок путей, сходящихся в первую вершину, с начальными вершинами из $O_1^1 \setminus \{1\}$, а все термы – всевозможные такие пучки. Так, первый терм состоит из трех частей: 2·1, 3·8·1, 4·1, каждая из которых представляет отдельный путь этого пучка соответственно из вершин 2, 3, 4. Аналогично строятся непустые ДНФ входящих пучков для остальных основных вершин комплекса.

Из этих выражений при построении U_i^j [9] образуется терм $2\cdot1\cdot3\cdot8\cdot1\cdot4\cdot1$ & $1\cdot2\cdot3\cdot2\cdot4\cdot6\cdot2$ & $1\cdot5\cdot6\cdot7\cdot3\cdot2\cdot3\cdot4\cdot3$ & $1\cdot4\cdot2\cdot4\cdot3\cdot9\cdot10\cdot11\cdot12\cdot4$, который полностью определяет пути посылки сообщений в процессе ВИСОВ.

Третий этап согласования – посылка результата согласования в каждую неосновную вершину не менее чем из 2μ + 1 основных вершин. В качестве примера рассмотрим неосновную 7-ю ЦВМ, для которой построим непустую ДНФ входящей смежности:

$SRP\_DNF(O_1^1 \setminus \{7\}, \widetilde{S}_1 \setminus O_1^1) = 1\cdot5\cdot6\cdot7\cdot2\cdot8\cdot7\cdot3\cdot9\cdot7 \vee \ldots \vee 1\cdot5\cdot6\cdot7\cdot2\cdot8\cdot7\cdot4\cdot9\cdot7 \vee \ldots$
$1\cdot10\cdot5\cdot6\cdot7\cdot2\cdot8\cdot7\cdot3\cdot9\cdot7 \vee \ldots$

Четвертый этап согласования – посылка ВСЗК комплекса-источника в ЦВМ нулевого ранга комплекса-получателя согласно условиям ранжирования вершин из [10] и вычисление в них ВСЗС. Комплексом-источником и комплексом-получателем могут быть любые комплексы МВС. Рассмотрим посылку ВСЗК 1-го комплекса во 2-й: $SRP\_DNF(O_2^1 \setminus \{11\}, \widetilde{S}_1 \setminus O_1^1) = 1\cdot11\cdot3\cdot11\cdot4\cdot11 \vee \ldots$ Аналогичные ДНФ стоятся для вершин 12, 17, 20.

Пятый этап согласования – вычисление ВСЗС в ЦВМ ненулевого ранга комплекса-получателя согласно условиям из [13]: $SRP\_DNF(O_2^1 \setminus \{13\}, \widetilde{S}_1 \setminus O_2^1) = 11\cdot18\cdot13\cdot12\cdot13\cdot17\cdot13 \vee 11\cdot17\cdot13\cdot12\cdot13\cdot14\cdot13 \vee \ldots$
Аналогичные ДНФ стоятся для вершин 14 и 18.

В соответствии с приведенным выше определением пораундной синхронизации (первый случай), время, необходимое для выполнения всех пяти этапов согласования, составляет сумму максимальных времен, необходимых для выполнения каждого этапа. Для орграфа, приведенного на рисунке, время ВИСНС составляет 20 квантов (см. таблицу).

Для определения времени согласования при поэтапной синхронизации (второй случай) выполним следующие действия.

Назовем орграфом R_i посылок согласуемого значения $З_i$ информации ЦВМ-источника M_i, передаваемого в своем начальном кванте из i-й вершины в процессе выполнения некоторого алгоритма ВИС, ациклический орграф, являющийся ориентированным деревом, в котором путь от корня дерева до некоторого его листа отображает процесс посылки копии значения $З_i$ или значения, сформированного с участием значения $З_i$,

из ЦВМ-источника этого согласуемого значения к ЦВМ-получателю этого значения. В таком пути возможно наличие повторяющихся вершин, отображающих многократность взаимообменов между ЦВМ в процессе ВИС. Объединение всех таких деревьев полностью определяет процесс ВИС для полносвязной МВС или ВИСНС для неполносвязной МВС. Ниже, без потери общности, будем рассматривать только алгоритм ВИСНС.

Таблица

Сравнительные значения временной избыточности в зависимости от способа синхронизации

Синхронизация	I этап	II этап	III этап	IV этап	V этап
пораундная – ***20 квантов***	3 кванта	8 квантов	2 кванта	4 кванта	3 кванта
поэтапная – ***18 квантов***	3 кванта	6 квантов	2 кванта	4 кванта	3 кванта
по началу и концу процесса согласования с промежуточным вычислением ВСЗК – ***14 квантов***	Вычисление ВСЗК в 1-й ЦВМ занимает 7 квантов во 2-й ЦВМ – 6 квантов в 3-й ЦВМ – 9 квантов в 4-й ЦВМ – 6 квантов в 5-й ЦВМ – 8 квантов в 6-й ЦВМ – 9 квантов в 7-й ЦВМ – 11 квантов в 8-й ЦВМ – 10 квантов в 9-й ЦВМ – 10 квантов в 10-й ЦВМ – 8 квантов			Посылка из 1-й ЦВМ в 11-ю занимает 1 квант из 2-й ЦВМ в 12 – 1 квант из 3-й ЦВМ в 11, 12, 17 – 1 квант из 4-й ЦВМ в 11, 20 – 1 квант из 5-й ЦВМ в 17 – 1 квант из 6-й ЦВМ посылки нет из 7-й ЦВМ в 12, 20 – 1 квант из 8-й ЦВМ в 12 – 1 квант из 9-й ЦВМ в 17, 20 – 1 квант из 10-й ЦВМ в 12 – 4 кванта	Вычисление ВСЗК в 11-й ЦВМ занимает 0 квантов в 12-й – 0 квантов в 17-й – 0 квантов в 20-й – 0 квантов в 13-й – 2 кванта в 14-й – 1 квант в 18-й – 1 квант

Если передача сообщения, поступившего в текущем кванте в ЦВМ-приемник, передается из этой ЦВМ в следующую по пути ЦВМ-приемник в следующем кванте, то вес соответствующей дуги будет равен единице. Если же некоторая ЦВМ-передатчик в составе этого пути задерживает передачу на t квантов, то вес соответствующей дуги будет равен t+1. Тогда вес пути будет равен длительности посылки в квантах по данному пути.

Объединим все орграфы посылок в орграф UR_i, тогда можно с достаточной очевидностью утверждать, что длительность процесса ВИСНС при любом сочетании уровней синхронизации будет равна наибольшему весу из весов всевозможных путей в UR_i. Этот наибольший вес будем называть величиной оценки временной избыточности рассматриваемого алгоритма ВИСНС.

Для приведенного примера в случае поэтапной синхронизации время ВИСНС составляет 18 квантов (см. таблицу).

Третье сочетание уровней синхронизации предполагает, что формирование ВСЗК происходит в каждой из ЦВМ комплекса-источника независимо от вычисления его в других ЦВМ после получения ею всех необходимых копий. Для предлагаемого примера сравнительные значения временной избыточности в зависимости от способа синхронизации приведены в таблице.

В результате работы получено следующее. В случае неполносвязных комплексов возможно сокращение временной избыточности путем перехода от пораундной синхронизации (первый случай) к поэтапной (второй случай). При переходе к третьему случаю (по началу и концу ВИСНС, по квантам) временная избыточность ВИСНС может быть существенно снижена, но при этом возрастает сложность алгоритма ВИСНС.

1. *Пархоменко П.П., Согомонян Е.С.* Основы технической диагностики: оптимизация алгоритмов диагностирования, аппаратурные средства / Под ред. П.П. Пархоменко. – М.: Энергия, 1981. – 320 с.
2. *Авиженис А.* Отказоустойчивость – свойство, обеспечивающее постоянную работоспособность цифровых систем // ТИИЭР. – 1978. – Т. 66. – № 10. – С. 5 – 25.
3. *Kuhl J.G., Reddy S.M.* Fault-tolerance considerations in large, multiple-processors systems // Computer. – 1986. – Vol. 19. – № 3. – P. 56 – 67.
4. *Нефедов В.Н., Осипова В.А.* Курс дискретной математики: уч. пос. – М.: Изд-во МАИ, 1992.
5. *Lamport L., Shostak R., Pease M.* The byzantine generals problem // ACM Trans. Progr. Lang. and Syst. – 1982. – Vol. 4. – № 3. – P. 382 – 401.
6. *Pease M., Shostak R., Lamport L.* Reaching agreement in the presence of faults // J. Ass. Comput. Mach. – 1980. – Vol. 27. – № 2. – P. 228 – 237.
7. *Ашарина И.В., Лобанов А.В., Мищенко И.Г.* Взаимное информационное согласование в неполносвязных многомашинных вычислительных системах // Автоматика и телемеханика. – 2003. – №5. – С. 190 – 198.
8. *Ашарина И.В., Лобанов А.В.* Взаимное информационное согласование в неполносвязных гетерогенных многомашинных вычислительных системах // Автоматика и телемеханика. – 2010. – № 5. – С. 133 – 146.
9. *Ашарина И.В., Лобанов А.В.* Выделение комплексов, обеспечивающих достаточные структурные условия системного взаимного информационного согласования в многокомплексных системах // Автоматика и телемеханика. – 2014. – №6. – С. 115 – 131.
10. *Ашарина И.В., Лобанов А.В*. Выделение структурной среды системного взаимного информационного согласования в многокомплексных системах. // Автоматика и телемеханика. – 2014. – №8. – С. 146 – 156.

А.Е. Васильев, Г.С. Васильянов, Т.Ю. Иванова,
Д.Ф. Кабесас Тапиа, А.Е. Переверзев, Я.Д. Садин

РАЗРАБОТКА ОТЕЧЕСТВЕННОГО МИКРОКОНТРОЛЛЕРА С АППАРАТНОЙ ПОДДЕРЖКОЙ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ НЕЧЕТКИХ ВЫЧИСЛЕНИЙ

Санкт-Петербургский политехнический университет Петра Великого,
г. Санкт-Петербург,
avasil@aivt.ftk.spbstu.ru

Введение

Встраиваемые интеллектуальные системы управления (ВИСУ) применяются в широком спектре задач адаптивного управления техническими объектами и процессами. В качестве аппаратных платформ для реализации ВИСУ разработчиками активно применяются функционально-ориентированные микроконтроллеры (ФОК), которые содержат в себе ряд схемотехнических и архитектурных решений, направленных на эффективную реализацию алгоритмов управления, в том числе адаптивного и интеллектуального управления.

Широко используются ВИСУ, построенные на основе концепции теории нечетких множеств, позволяя эффективнее реализовать сложные алгоритмы управления объектом. Следует отметить, что применение в микроконтроллерных ВИСУ программных эмуляторов нечетких вычислений сопряжено с проблемой времени выполнения эмуляции, при этом повышение тактовой частоты микроконтроллера зачастую недопустимо, так как приводит к увеличению риска сбоя в вычислительном процессе, а также к увеличению энергопотребления.

Существенно повысить производительность нечетких вычислений возможно путем использования разновидности ФОК – нечетких вычислителей (НВ), представляющих собой специализированные микроконтроллеры (МК), поддерживающие на аппаратном уровне реализацию нечетких алгоритмов управления.

Анализ существующих структурных и алгоритмических решений

Анализ способов применения существующих НВ в задачах интеллектуального управления выявляет характерные факторы, ограничивающие эффективность их использования:

- при использовании НВ в режиме нечеткого сопроцессора (НС) существенным ограничением целевой производительности являются частотные характеристики интерфейсной подсистемы;
- использование НВ, встроенного в основной кристалл МК, исключает параллельное функционирование двух вычислительных процессов –

исполнения основной программы и выполнения нечетких вычислений, так как на ряде стадий нечетких вычислений используются ресурсы процессорного ядра МК, что приводит к автоматической приостановке алгоритмических вычислений на время работы НВ.

Кроме того, в обоих режимах функционирования НВ существующие технические решения не обеспечивают возможности задания произвольных функций принадлежности, допуская лишь их кусочно-линейный вид, что негативно сказывается на достижимой точности вычислений. Для преодоления этого недостатка разработчики вынужденно усложняют нечеткое описание системы (увеличивая количество термов лингвистических переменных и используемых правил для их обработки), что неизбежно приводит к увеличению времени выполнения нечетких вычислений.

Таким образом, значимой научной проблемой является проблема повышения производительности функционирования ВИСУ с нечеткой обработкой информации путем разработки математического аппарата их описания и разработки схемотехнических решений, обеспечивающих эффективное (в характерных для ВИСУ смыслах) вычисление нечетких зависимостей на основе такого математического аппарата.

Вопросам разработки математического аппарата описания нечетких систем обработки информации для встраиваемых приложений посвящен ряд публикаций авторов данной статьи [1]. Ниже рассматриваются вопросы схемотехнической реализации ФОК с аппаратной поддержкой нечетких вычислений.

Алгоритмические подходы к повышению производительности нечеткого вычислителя

Для реализации высокоскоростных вычислений в предлагаемом авторами алгоритме использован метод просмотровых таблиц, оперирующий термами, заданными совокупностью массивов-векторов, и базой правил, заданной матрицей.

Термы хранятся в виде массива значений степени принадлежности, упорядоченного по возрастанию четкой переменной. При выполнении фазификации соответствующий терму массив адресуется значением входной переменной, возвращая значение степени принадлежности данной переменной этому терму.

База правил представляется таблицей; количество строк в ней соответствует количеству правил. Каждая строка состоит из «входных» и «выходных» столбцов. «Входные» описывают номера термов входных переменных, участвующих в формировании условий данного правила, а «выходные» – номера выходов, вычисляемых при срабатывании условий данного правила.

Такая организация структур базы нечетких знаний обеспечивает одновременный значительный прирост точности и производительности

по сравнению с известными алгоритмическими решениями, так как позволяет оперировать с функциями принадлежности произвольного вида; при этом скорость фаззификации не зависит от вида функции принадлежности и является максимальной из достижимых (так как процедура вычисления значения функции принадлежности сводится к индексации заданной ячейки одномерного массива). Абсолютные значения прироста производительности определяются способом реализации алгоритма.

Структурно-функциональные подходы к повышению производительности нечеткого вычислителя

Сущность авторского решения [2] заключается в том, что на едином кристалле размещаются основной процессор с АЛУ общего назначения и набором типовых периферийных устройств (порты ввода-вывода, таймеры и т.п.), и блок нечетких вычислений с отдельным специализированным АЛУ, которые имеют доступ к общему адресному пространству памяти.

Преимущество предлагаемого структурного решения (рис. 1) заключается в том, что блок нечетких вычислений не использует АЛУ основного процессора, что позволяет выполнять все операции по нечеткой обработке информации автономно и параллельно с обработкой команд и данных основным процессором. Использование отдельного АЛУ блока нечетких вычислений позволяет также обрабатывать функции принадлежности произвольного вида, задаваемые табличным способом, что расширяет многообразие реализуемых нечетких систем управления по сравнению с известными решениями.

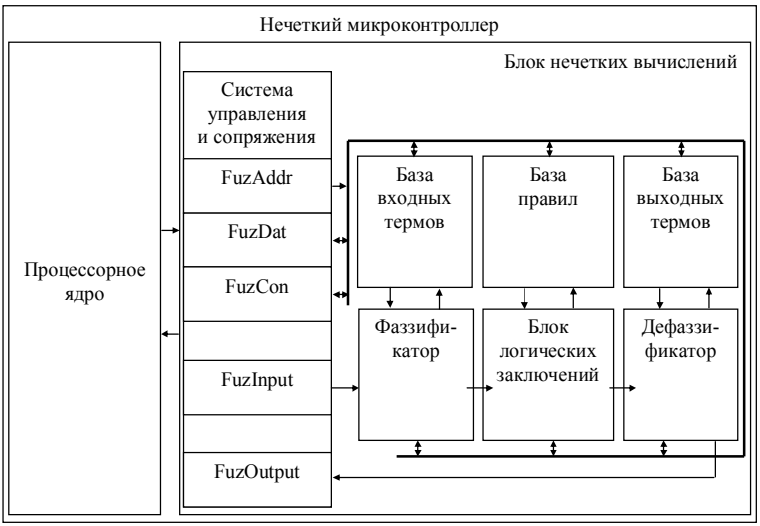


Рис. 1. НК с интегрированным нечетким сопроцессором

Доступ основного процессора и блока нечетких вычислений к общей памяти позволяет обеспечить высокую скорость обмена информацией между ними. Это также упрощает задание функций принадлежности произвольного вида табличным способом. Кроме того, доступ к общей памяти позволяет осуществлять оперативное переключение на другую активную базу нечетких знаний в темпе реального времени.

Аппаратная реализация и экспериментальные исследования нечеткого вычислителя

В качестве программно-аппаратной основы разработки применен созданный авторами в ходе предшествующих НИР в МНОЦ «Встраиваемые системы автоматики и вычислительной техники» комплекс средств автоматизированного проектирования функционально-ориентированных микроконтроллеров [3] (рис. 2).

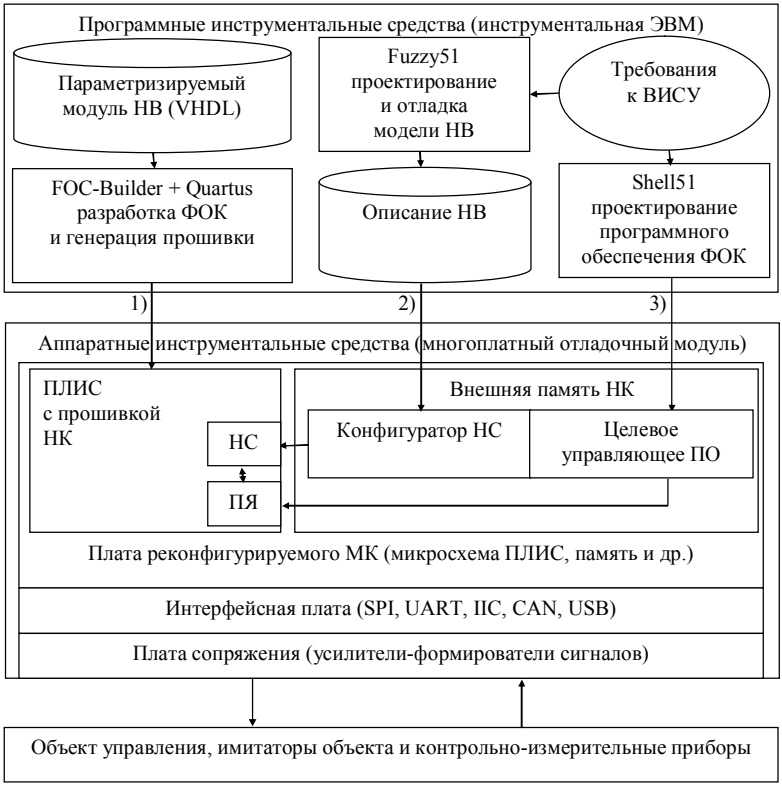


Рис. 2. Структура инструментального комплекса

Рассмотренные выше структуры блока нечетких вычислений были реализованы на языке VHDL с применением системы Quartus, интегрированы в библиотеки FOC-Builder как новое периферийное устройство и добавлены к процессорному ядру семейства MCS-51; таким образом, получен экземпляр x51-совместимого микроконтроллера со встроенным аппаратным вычислителем нечетких логических функций, обладающим следующими параметрами: число входных переменных – от 1 до 5, число выходных переменных – от 1 до 5, число правил – от 1 до 255, число термов на каждую лингвистическую переменную – от 1 до 10.

Сравнение характеристик разработанного НВ и серийно выпускаемых НВ показало 20-кратное превосходство авторской разработки по точности при одновременном 10-кратном преимуществе авторской разработки в быстродействии.

Заключение

Предлагаемые авторами подходы обеспечивают существенное комплексное улучшение значений показателей качества функционирования встраиваемых микроконтроллерных систем с нечеткой обработкой информации. Значение результатов работы для практики заключается в повышении быстродействия функционирования и снижении ресурсоемкости встраиваемых реализаций нечетких преобразователей информации, синтезированных с применением предложенных средств и методов, а также в разработке инструментальных комплексов автоматизированной генерации внутрикристальных нечетких преобразователей информации, использующих предложенные методы, что обеспечивает снижение трудоемкости их анализа и синтеза.

1. *Vasil'ev, A.E., Giganova, V.I.* Networks of elementary fuzzy solvers for synthesizing and analyzing fuzzy data processing and control systems // Journal of Computer and Systems Sciences International. Maik Nauka-Interperiodica Publishing. – 2014. – Vol. 53. – Issue № 6. – P. 808 – 818.
2. Микроконтроллер с аппаратным нечетким вычислителем переменной структуры // Патент РФ на изобретение № 2477525. Патентообладатель Васильев А.Е. – Опубл. 10.03.2013. – Бюлл. № 7.
3. *Vasil'ev A.E., Ivanova T.Yu, Cabezas T.D.F., Sadin Ya. D.* Filed-programmable gate array based function-oriented microcontrollers design automation // Automatic control and computer sciences. Allerton press. – 2015. – Vol. 49. – Issue № 6. – P. 404 – 411.

В.С. Горбунов, А.А. Соколов, П.В. Павлухин, Ю.А. Климов

КОММУНИКАЦИОННАЯ СЕТЬ «МВС-ЭКСПРЕСС» КОЛЬЦЕВОГО ТИПА ДЛЯ РЕШЕНИЯ РАЗРЕЖЕННЫХ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

ФГУП «НИИ «Квант», ИПМ им. М.В. Келдыша РАН, г. Москва,
gorbunov@rdi-kvant.ru, sokolov@rdi-kvant.ru, pavlukhin@rdi-kvant.ru,
yuklimov@keldysh.ru

Нередко суперкомпьютеры являются проблемно-ориентированными и строятся для решения определённого класса наиболее значимых прикладных задач. Среди актуальных современных задач существенную долю составляют те, скорость решения которых ограничивается возможностями коммуникационной подсистемы – её топологией, пропускными способностями интерфейсов, коммуникационными задержками и т.д. При этом если в целевой прикладной задаче шаблон коммуникаций требует эффективного взаимодействия произвольных пар узлов, то при построении суперкомпьютера целесообразно применение топологий типа звезда или дерево. Если же пиковые показатели производительности сети требуются лишь при взаимодействии определённых пар узлов, которые ещё к тому же можно и закольцевать между собой, то тогда при построении суперкомпьютера целесообразно применение топологии кольцевого типа.

К такому классу задач относятся те задачи, в которых основной трудоёмкой операцией является операция умножения большой разреженной матрицы на плотный вектор – например, задача решения большой разреженной системы линейных алгебраических уравнений методом сопряжённых градиентов. Метод сопряжённых градиентов решения СЛАУ требует эффективной реализации двух коммуникационных операций – Allgather и Reduce_scatter (в некоторых реализациях достаточно эффективно реализовать лишь одну операцию Allgather). В вычислительных комплексах с топологией «жирное дерево» для эффективной реализации этих коммуникационных операций применяются специальные алгоритмы, которые обладают некоторым преимуществом по сравнению с более простыми кольцевыми реализациями. Однако это преимущество проявляется только на больших конфигурациях (100 – 1000 узлов), а сам метод сопряжённых градиентов имеет внутренние ограничения масштабирования и практически не может эффективно выполнятся на таких конфигурациях. Поэтому для этой задачи разумно строить небольшие суперкомпьютеры, в которых применяются кольцевые коммуникационные сети, так как в сетях кольцевого типа нет избыточности оборудования в виде общего коммутатора, что в целом улучшает технико-экономические характеристики такого решения.

На протяжении нескольких лет в ФГУП «НИИ «Квант» совместно с ИПМ им. М.В. Келдыша РАН разрабатывается коммуникационная сеть «МВС-Экспресс» для решения задач, предъявляющих повышенные требования к коммуникационной подсистеме. Третье поколение сети «МВС-Экспресс» позволяет строить суперкомпьютеры с топологиями как «звезда», так и «кольцо». Для исследования возможностей третьего поколения сети «МВС-Экспресс» был создан экспериментальный образец кластера КВС-xPCIe-01, состоящий их 4-х серверов, в каждый из которых установлены 2-портовые адаптеры КВА-P8733-PCI-E-01, объединённые в кольцо (без использования коммутатора КВК-P8796-PCI-E-01).

Результаты измерений показали, что пропускная способность при взаимодействии соседних узлов кольца близка к пиковой пропускной способности интерфейса, что необходимо для эффективного решения заданных прикладных задач. Оптимизированные реализации коллективных коммуникационных операций Allgather и Reduce_Scatter продемонстрировали возможность применения сети «МВС-Экспресс» кольцевого типа при построении суперкомпьютеров, предназначенных для решения задач целевого класса.

Н.И. Дикарев, Б.М. Шабанов, А.С. Шмелев

ВЫБОР ОПТИМАЛЬНОЙ ПРОИЗВОДИТЕЛЬНОСТИ ЯДРА ВЕКТОРНОГО ПОТОКОВОГО ПРОЦЕССОРА*

Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, г. Москва,
nic@jscc.ru, shabanov@jscc.ru, guest8993@rambler.ru

Замедлившийся в последние годы рост степени интеграции СБИС говорит о том, что совершенствование КМОП-технологии, по-видимому, подходит к концу, и следует искать другие способы повышения производительности суперЭВМ помимо простого увеличения числа процессорных ядер на кристалле СБИС и числа таких кристаллов в суперЭВМ. Это тем более справедливо, если учесть, что при числе ядер в суперЭВМ, превышающем сотни тысяч, их и сейчас трудно задействовать для распараллеливания одной программы, поскольку простое увеличение размеров сетки в задачах моделирования уже не приводит к заметному повышению его точности. Однако точность моделирования можно повысить, если вместо упрощенных математических моделей реальных систем использовать более сложные модели, учитывающие важные детали этих систем. При этом модели становятся нелинейными

* Работа выполнена при поддержке гранта РФФИ 13-07-01124

и со связанными структурами, что приводит к необходимости работы с мелкоструктурным и нерегулярным параллелизмом, на котором современные процессоры и ускорители имеют низкую эффективность работы. Разрабатываемый в МСЦ РАН векторный процессор с архитектурой управления потоком данных (ВПП) имеет не только значительно более высокую производительность одного процессорного ядра, но и способность её сохранения при работе с мелкоструктурным параллелизмом [1], что обеспечивает ему существенные преимущества при использовании в суперЭВМ. Цель данной работы – оценить оптимальное значение пиковой производительности одного ядра ВВП, при которой достигается максимальное отношение реальной производительности ВПП на единицу площади кристалла, и сравнить этот параметр у ВПП с GPU.

Разрабатываемый ВПП выполняет команды по готовности операндов с возможностью выдачи до 8 команд в такт и реализует поиск параллелизма уровня отдельных команд в диапазоне до 32 тысяч команд. Это дает возможность выявлять гораздо больше параллелизма в программе, обеспечивая загрузку большого числа исполнительных устройств (ИУ) даже при работе с мелкоструктурным параллелизмом.

Использование в ВПП векторной обработки позволяет в VL раз сократить число выполняемых команд на векторизуемой части программы, где VL – длина вектора. Для записи результатов векторных команд и хранения массивов в ВПП используется линейно адресуемая память, содержащая два уровня – память векторов (ПВ) большой емкости, реализованную на микросхемах динамической памяти, и быструю локальную память векторов (ЛПВ) значительно меньшей емкости, размещенную на процессорном кристалле. Распределение ресурса ПВ и ЛПВ в ВПП реализовано на аппаратном уровне, и в качестве единицы фрагментации используется вектор с фиксированным числом слов VL_{max}=256. В этом случае входящее в состав ВПП устройство распределения памяти ведет список свободных векторов для ПВ и ЛПВ и выделяет для записи результата векторной команды свободный вектор из затребованного списка. Тогда адрес начального элемента вектора вместе с фактической длиной VL, которая меньше или равна VL_{max}, и битом уровня памяти составляют аппаратный указатель (имя) вектора. Этот указатель однозначно определяет вектор для его использования в качестве операнда и передается в поле данных токена на входы последующих команд согласно графу программы.

Из-за сравнительно низкой пропускной способности ПВ основным запоминающим устройством в разрабатываемом ВПП, обеспечивающем высокую пропускную способность для работы большого числа ИУ, является ЛПВ, которая расслоена на несколько сотен банков и имеет двухуровневую структуру. Верхний уровень составляют W идентичных

колец (lines), групповая синхронная работа которых позволяет читать вектора-операнды и записывать вектора-результаты конвейерных ИУ в векторном АЛУ страницами по W элементов в такт. Лишь блок выполнения специальных операций (БВСО) может выполнять одиночные обращения к ПВ и ЛПВ между кольцами для доступа к произвольным элементам вектора на скалярной обработке и при выполнении специальных векторных команд, таких как команды сдвига, собрать и разбросать элементы вектора. На нижнем уровне (в каждом кольце) ЛПВ состоит из $B=2^m$ двухпортовых банков, число которых выбирается так, чтобы могли одновременно работать несколько конвейерных ИУ в каждом кольце. Например, при числе колец W=32 для обеспечения ВПП производительностью 256 флоп в такт нужно, чтобы в каждом кольце было по 4 конвейерных сумматора и умножителя с плавающей запятой двойной точности. Тогда к ЛПВ из 16-двухпортовых банков может обращаться до 32 портов, что позволит через 24 порта читать операнды и записывать результаты для восьми ИУ векторного АЛУ и ещё три порта выделить для работы БВСО. Именно для такого ВПП в [1] приведена оценка производительности на программе перемножения матриц.

Последние изменения, внесенные в схему и VHDL-модель ВПП, заключались в замене раздельных сумматоров и умножителей с плавающей запятой в векторном блоке процессора на сдвоенные умножители и сумматоры (FMA). Эти конвейерные ИУ выполняют часто встречающуюся комбинацию из двух операций с плавающей запятой по вычислению функции A*B+C, и их основное преимущество в уменьшении числа портов доступа к ЛПВ. Так, использование четырех FMA в каждом из 32-х колец ВПП позволяет уменьшить число портов чтения и записи элементов вектора от этих ИУ к ЛПВ с 24 до 16. Это приводит к снижению конфликтов занятого банка в расслоенной ЛПВ, и производительность ВПП на программе перемножения матриц повышается с 222,2 флоп в такт до 243,6 флоп в такт, что составляет 95 % пиковой производительности процессора. На рис. 1 показано изменение реальной производительности ВПП на программах перемножения матриц и решение систем дифференциальных уравнений Stencil, полученное на VHDL-модели ВПП, при изменении числа FMA в каждом из его колец.

Зависимость 32W 32B на рис. 1 получена для ВВП с 32-мя кольцами (W=32) при изменении числа FMA в кольце от 2 до 8 и, соответственно, пиковой производительности ВПП от 128 до 512 флоп в такт. Для работы 8 и 6 FMA в кольце нужно расслоение ЛПВ на 32 банка, а при 4 и 2 FMA – число банков в 2 и 4 раза меньше. Зависимость 32W 16B построена для вдвое меньшего числа банков в ЛПВ. Это возможно, поскольку одним из трех операндов у команд FMA в моделируемых программах является скаляр, а не вектор. Тогда при 8 FMA в кольце для

доступа к ЛПВ нужно не 35 портов (с учетом 3-х портов от БВСО), а 31 порт, что обеспечивается шестнадцатью двухпортовыми банками. Конечно, производительность ВПП в этом случае будет несколько ниже из-за увеличения числа конфликтов занятого банка в ЛПВ, но такой вариант обеспечивает большую реальную производительность ВПП с единицы площади кристалла. Действительно, аппаратные затраты в коммутаторе, с помощью которого входы и выходы FMA подключаются к банкам ЛПВ, растут квадратично от числа портов доступа к ЛПВ в каждом кольце, и их можно существенно уменьшить, сократив вдвое число банков в кольце. Поэтому две другие зависимости реальной производительности ВПП от числа FMA в кольце на рис. 1 (64W 8B), полученные для вдвое большего числа колец (W=64) на программах перемножения матриц и Stencil, также соответствуют минимальному числу банков в кольце.

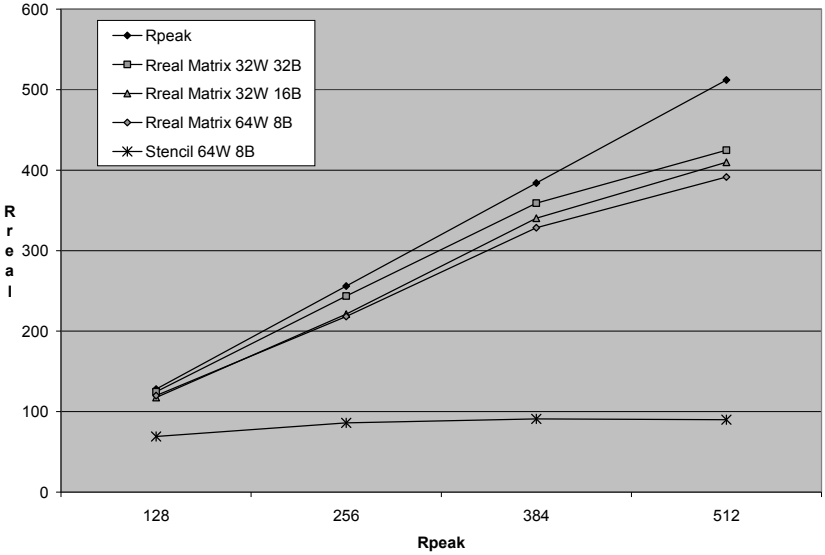


Рис. 1. Зависимость реальной производительности ВПП от пиковой

Поскольку увеличение числа колец уменьшает время выполнения команд сдвига в БВСО, то переход с 32-х колец на 64 позволил повысить производительность ВВП на программе Stencil на 25 %, до 96 флоп в такт (см. рис.1). Если при W=32 производительность ВПП ограничивалась командами сдвига, то теперь пропускной способностью к ПВ, которая при моделировании считалась равной 256 ГБ/с. Поэтому более предпочтительной является конфигурация ВПП с W=64, но для более обоснованного выбора конфигурации был проведен расчет пло-

щади кристалла СБИС с технологией изготовления 22 нм, как в [2], и для программы перемножения матриц определено отношение реальной производительности ВПП на единицу площади кристалла. Результаты, показанные на рис. 2, подтверждают, что минимальное число банков в кольце ВПП (32W 16B и 64W 8B) обеспечивает более высокую удельную производительность, достигая 3,24 и 3,12 флоп в такт с мм^2 площади соответственно, в то время как зависимость ВПП (32W 32B) имеет экстремум со значением 2,19 флоп в такт с мм^2. То, что у первых двух зависимостей с ростом числа FMA производительность на единицу площади кристалла монотонно увеличивается, позволяет надеяться, что удельную производительность ВПП можно будет ещё повысить. (Для этого необходимо совершенствование VHDL-модели ВПП.) Однако можно сравнить достигнутое значение удельной производительности с тем же параметром у последнего GPU фирмы NVIDIA Pascal P100. Это GPU при пиковой производительности 5,3 Тфлоп/с на частоте 1,48 ГГц показывает на программе перемножения матриц производительность 4,0 Тфлоп/с, или 2703 флоп в такт. Площадь кристалла GPU составляет 610 мм^2 при технологии изготовления 16 нм, что с учетом приведения к технологии 22 нм даёт ту же удельную производительность 3,22 флоп в такт с мм^2, что у ВПП.

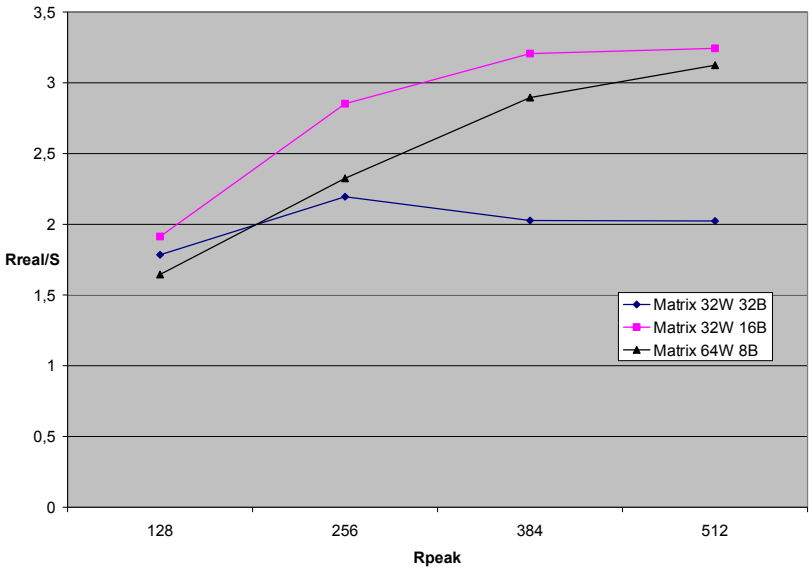


Рис. 2. Реальная производительность на единицу площади кристалла от пиковой производительности ядра ВПП

Таким образом, разрабатываемый ВПП способен обеспечить ту же производительность, что имеют лучшие GPU, но ВПП – универсальный процессор, а не ускоритель, и в отличие от GPU способен сохранять высокую производительность на программах с мелкоструктурным и нерегулярным параллелизмом.

1. *Дикарев Н.И., Шабанов Б.М., Шмелёв А.С.* Векторный потоковый процессор: оценка производительности // Известия ЮФУ. Технические науки. – 2014. – №12 (161). – С. 36 – 46.
2. *Дикарев Н.И., Шабанов Б.М., Шмелев А.С.* Проблемы масштабирования производительности в векторном процессоре с архитектурой управления потоком данных // Суперкомпьютерные технологии: материалы 2-й Всеросс. науч.-техн. конф. – Ростов-на-Дону: Изд-во ЮФУ, 2012. – С.34 – 38.

Ю.И. Доронченко, А.Г. Коваленко

БИБЛИОТЕКА VHDL-ЭЛЕМЕНТОВ ДЛЯ FPGA XILINX ULTRASCALE

ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог, doronchenko@superevm.ru. k.a.g@bk.ru

Создание прикладных программ для реконфигурируемых вычислительных систем (РВС) при современном уровне интеграции ПЛИС требует от разработчика существенных временных затрат – каждая реализуемая операция должна достигать необходимого уровня эффективности по частоте реализации и используемому аппаратному ресурсу.

В настоящее время в САПР для ПЛИС наметилась устойчивая тенденция к переходу на более высокий уровень проектирования. В САПР Vivado [1] фирмы Xilinx уже не поддерживается графический схемотехнический редактор, программирование ПЛИС осуществляется только на языке VHDL, активно развиваются возможности написания программ для ПЛИС на языке С. Предполагается, что в дальнейшем разработчику не нужно будет знать архитектурные особенности ПЛИС, а эффективность отображения программного кода на ресурсы кристалла должен взять на себя транслятор.

Тем не менее эффективность существующих средств трансляции, в том числе программ, написанных на языках С-группы, остается довольно низкой. Использование VHDL или IP-ядер по-прежнему требует специфичных знаний для достижения высокой тактовой частоты. Зачастую опытные разработчики имеют в своем арсенале собственные эффективно реализованные функции, которые могут быть использованы многократно в различных проектах.

В НИЦ супер-ЭВМ и нейрокомпьютеров разработана подобная библиотека схемотехнических элементов. Элементы реализованы на языке VHDL, что в целом позволяет использовать их для ПЛИС различ-

ных производителей и семейств, однако при разработке целевым являлось семейство UltraScale фирмы Xilinx.

Наибольшие возможности разработанная библиотека предоставляет программистам на языке высокого уровня COLAMO [2] при использовании среды разработки прикладных программ, созданной коллективами НИИ многопроцессорных вычислительных систем Южного федерального университета и НИЦ супер-ЭВМ и нейрокомпьютеров. Применение того или иного элемента библиотеки происходит в данном случае автоматически в зависимости от программного кода.

Библиотека схемотехнических элементов разбита на группы по функциональному назначению, обрабатываемой разрядности и наличию триггеров на выходных шинах элементов. Библиотека включает в себя следующие 20 групп:

- «COMMON» – элементы общего назначения; в группу входят служебные элементы, счетчики, программируемые буферы задержек данных и пр.;
- «COMMUTATION» – элементы, предназначенные для коммутации данных;
- «FP32» – элементы, выполняющие математические операции и операции сравнения над числами в формате с плавающей точкой одинарной точности;
- «FXS_RG» – элементы переменной разрядности, выполняющие математические операции и операции сравнения над знаковыми целыми числами, с триггерами на выходах;
- «FXS_OTHERS» – элементы фиксированной разрядности, выполняющие математические операции над знаковыми целыми числами;
- «FXU» – элементы переменной разрядности, выполняющие математические операции и операции сравнения над беззнаковыми целыми числами;
- «FXU_OTHERS» – элементы фиксированной разрядности, выполняющие математические операции над беззнаковыми целыми числами;
- «LOGIC» – элементы переменной разрядности, выполняющие типовые логические операции для двух – шести операндов;
- «LOGIC_OTHERS» – элементы фиксированной разрядности, выполняющие логическую операцию для 32-х операндов;
- «STORAGE» – элементы переменной разрядности, предназначенные для хранения пользовательских данных.

Группы «COMMON_64_2T», «FP64_2T», «FXS64_2T», «FXU64_2T», «STORAGE_64_2T» выполняют операции с данными разрядностью 64 бита, которые передаются по 32-битовым шинам за два такта.

Группы «COMMON_64», «FP64» выполняют операции с данными разрядностью 64 бита, которые передаются по 64-битовым шинам за один такт.

В группах «COMMUTATION_RG», «FXU_RG», «LOGIC_RG» схемотехнические элементы содержат триггеры на всех выходах. В среде разработки на языке COLAMO это позволяет полностью исключить работу по расположению в нужных местах схемы прореживающих триггеров для трассировки сигналов на необходимую тактовую частоту. Синхронизирующие триггеры устанавливаются автоматически. Однако при таком подходе количество триггеров может существенно вырасти.

Все элементы, выполняющие операции в формате с плавающей запятой, в том числе тригонометрические и логарифмические операции, соответствуют спецификациям стандарта IEEE-754, при этом разработан ряд элементов с сокращенным аппаратным ресурсом, запрещающих обработку денормализованных чисел с плавающей запятой.

На рис.1 изображено представление библиотеки в системе проектирования Vivado фирмы Xilinx. Всего библиотека содержит vhld-описание более 300 схемотехнических элементов. Тактовая частота работы элементов варьируется от 450 до 550 МГц на ПЛИС семейства UltraScale с параметром “speed grade”, равным – 1.

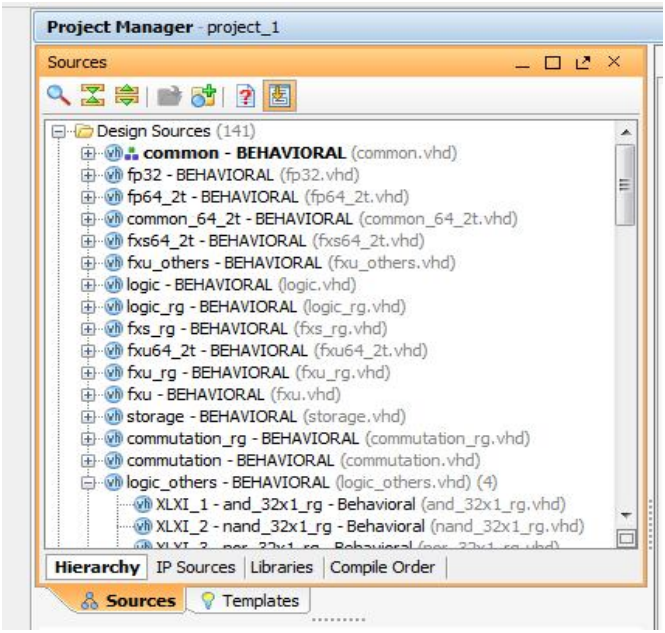


Рис. 1. Представление библиотеки схемотехнических элементов в САПР Vivado фирмы Xilinx

Текущая структура представления схемотехнических элементов адаптирована для их удобного использования в среде разработки на языке COLAMO. При написании программы разработчик при помощи директивы include указывает операционную библиотеку элементов и группу элементов, которую нужно использовать в проекте, в формате:

Include <название_библиотеки>.<название_группы>;

Для удобства пользователь может выбрать нужные группы элементов из выпадающего списка, как это показано на рис. 2.

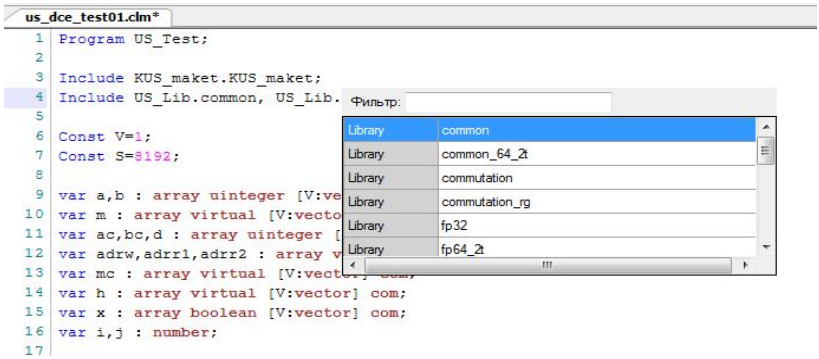


Рис. 2. Выбор групп элементов из библиотеки в среде разработки

VHLD-описание библиотечных элементов содержит в себе специальную структуру комментариев, предназначенных для синтезатора Fire!Constructor, входящего в программный комплекс разработки прикладных программ [3].

Разработанная библиотека схемотехнических элементов может быть полезна как опытным разработчикам для сокращения времени подготовки проектов, так и начинающим схемотехникам и программистам для достижения достаточно высоких характеристик производительности своих решений без многократных оптимизаций кода и углубленного изучения архитектуры ПЛИС. Кроме того, библиотека может корректироваться, улучшаться и дополняться пользователем исходя из предметной области решаемых задач.

1. http://www.xilinx.com/products/design-tools/vivado.html
2. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры / Изд. 2-е, перераб. и доп. / под общ. ред. И. А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
3. *Левин И.И., Дордопуло А.И., Гуленок А.А.* Синтез параллельных прикладных программ для многокристальных реконфигурируемых вычислителей. Синтезатор Fire!Constructor: учебное пособие. – Таганрог: Изд-во ТТИ ЮФУ, 2013. – 96 с.

Г.С. Елизаров, В.С. Горбунов, А.Г. Титов

ПЕРСПЕКТИВНЫЕ РЕКОНФИГУРИРУЕМЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МОДУЛИ НА БАЗЕ ИНТЕРФЕЙСА PCI EXPRESS И ПЛИС XILINX ULTRASCALE+

ФГУП «НИИ «Квант», г. Москва,
info@rdi-kvant.ru

Введение

Переход на перспективные ПЛИС обеспечивает рост технико-экономических показателей создаваемых реконфигурируемых суперкомпьютеров [1]. Фирма Xilinx начнет выпуск нового семейства ПЛИС UltraScale+ в 2017 г. по 16 нм технологическому процессу. По информации производителя переход на новый технологический процесс позволит увеличить производительность ПЛИС на Ватт потребляемой мощности до 2 – 2,5 раз по сравнению с ПЛИС Xilinx-7 (технологический процесс 28 нм) [2]. Поэтому актуальной является задача создания компонентов реконфигурируемых суперкомпьютеров на новой элементной базе. Основными компонентами реконфигурируемых суперкомпьютеров, разрабатываемых ФГУП “НИИ “Квант” являются реконфигурируемые вычислительные модули, реконфигурируемые вычислительные блоки и интерфейсные платы. Наиболее технически сложными изделиями среди перечисленных компонент являются реконфигурируемые вычислительные модули. В докладе рассматриваются подходы к проектированию и основные характеристики реконфигурируемых вычислительных модулей на новой элементной базе.

Характеристики нового семейства ПЛИС Xilinx Ultrascale+

Новое семейство ПЛИС Xilinx Ultrascale обладает следующими отличительными особенностями:

- при производстве применяется новый техпроцесс – 16 нм;
- в данной серии станет доступен новый ресурс памяти на кристалле, получивший название UltraRAM;
- наличие аппаратной поддержки интерфейса PCIe 4-го поколения (gen4) с пиковой пропускной способностью 16 Гбит/с на линию.

Основные характеристики ПЛИС Xilinx Ultrascale+ в сравнении с ПЛИС Ultrascale и Xilinx-7 приведены в таблице.

Реконфигурируемые модули и их основные характеристики

Реконфигурируемый вычислительный модуль (РВМ) на базе Xilinx Ultrascale+ КВ8-KUP-15-01 разработан в ФГУП «НИИ «Квант». Основные технические характеристики модуля:

- размер модуля: 375×150 мм;
- сервисная ПЛИС: Xilinx Virtex-6 (LX75/130/195/240T) FF(G)784;
- восемь рабочих ПЛИС: Xilinx Kintex Ultrascale+ (XCKU15P) FFVA1156;
- сеть PCI Express: одна СБИС PEX8648 коммутатора PCI Express Gen2 x4 (используются 11 портов), все ПЛИС подключены к PEX8648;
- микросхема FLASH-памяти: одна конфигурационная Flash 512 Мбит;
- память SDRAM: 1Gb DDR3 SDRAM DDR-800;
- сеть Ethernet: Fast Ethernet Base-T.

Таблица

Характеристики ПЛИС Xilinx Ultrascale+ в сравнении с ПЛИС Ultrascale и Xilinx-7

Параметр	Kintex 7	Kintex US	Kintex US+	Virtex US+
Тип ПЛИС	XC7K410	XCKU060	XCKU15P	XCVU7P
Корпус ПЛИС	FFG	FFVA	FFVA	FLVB
Техпроцесс	28 нм	20 нм	16 нм	16 нм
Частота, МГц	625	630	667	667
Емкость (тыс. ячеек)	406	580	915	1379
Объем памяти BRAM/ URAM, Мбит	28,6 /0	38,0 /0	34,6 /36	50,6 /180
Интерфейс PCIe	x8 gen2 32 Гбит/с	x8 gen3 64 Гбит/с	x8 gen4 128Гбит/с	x8 gen4 128Гбит/с

Системная ПЛИС и 8 рабочих ПЛИС подключены к коммутатору PCI Express, реализованному на микросхеме PEX8648 (12 портов x4) фирмы PLX Technology. Объединение ПЛИС реализовано по схеме «звезда». Дополнительно все ПЛИС объединены в кольцо (кольцо Rocket IOx4), что позволяет организовать высокоскоростной канал обмена данными. Кроме того, все ПЛИС подключены к общей шине для организации управления. Загрузку конфигурационных данных в рабочие ПЛИС осуществляет сервисная ПЛИС через 8 независимых каналов в режиме Slave Serial. Для организации мониторинга основных компонентов на плате реализован внешний сетевой интерфейс Ethernet, а для связи с внешними PCI Express-устройствами – два канала PCIex4.

В настоящее время в ФГУП «НИИ «Квант» завершена разработка еще одного типа вычислительного модуля КВ6-VUP-07-01. Модуль предназначен для установки рабочих ПЛИС серии Virtex Ultrascale+ типа XCVU7P, которые будут производиться фирмой Xilinx с середины 2017 г. Основными отличительными особенностями этого модуля от РВМ семейства КВ8 являются:

- сокращение числа рабочих ПЛИС (РП) до 6 шт., а также увеличение габаритного размера модуля на 74 мм (теперь 450 мм в длину), что связано с увеличением размера корпуса микросхем РП (47,5 х 47,5 мм вместо максимального для семейства KB8 35 х 35 мм);
- расширение функциональных возможностей за счет перехода на интерфейс PCI Express Gen3, добавление в систему мониторинга функции контроля потребляемой мощности каждой ПЛИС, доработка системы автономной диагностики и тестирования с целью упрощения процедуры наладки, тестирования и последующей эксплуатации РВМ;
- применение микросхемы памяти типа DDR4 вместо DDR3 обеспечило переход на современный стандарт памяти;
- применение низкопрофильных источников питания типа LTM4630 с максимальным током 36А (3 шт. на РП) вместо PTH08T250W (+1V_Core, 50А) обеспечило ток питания ядра каждой РП до 108 А (103 Вт) и возможность их охлаждения общим с РП радиатором.

Несмотря на сокращение количества рабочих ПЛИС до 6 общая суммарная логическая емкость модуля, выражаемая в логических ячейках, находится на уровне 8274, что несколько больше, чем в модуле KB8-KUP-15-01.

Заключение

В докладе рассмотрены два проекта реконфигурируемых вычислительных модулей. Приведены ожидаемые технические характеристики. В настоящее время оба проекта завершены. Идет подготовка к выпуску опытных образцов изделий. Для проведения отладки функционирования изделий при изготовлении опытных образцов будут использоваться совместимые по корпусам серийно выпускаемые ПЛИС серии Kintex Ultrascale. По мере выпуска ПЛИС серии Ultrascale+ в 2017 г. будет выпущена установочная партия изделий для определения технико-экономических характеристик и принятия решения о возможности тиражирования. Результаты исследований новых изделий планируется опубликовать по мере их появления в 2017 – 2018 гг.

1. *Елизаров Г.С., Горбунов В.С., Малахов И.Н., Титов А.Г.* Компоненты высокопроизводительных реконфигурируемых суперкомпьютеров на основе ПЛИС Xilinx Ultrascale // Суперкомпьютерные технологии: материалы 3-й Всеросс.науч.-техн. конф. – Ростов-на-Дону: Изд-во ЮФУ. – Т2. – 2014. – С. 21 – 25.
2. Multiplying the Value of 16nm – Staying a Generation Ahead. Xilinx, Inc© 2015.http://www.xilinx.com/support/documentation/product-briefs/multiplying-the-value-of-16nm.pdf

М.Ф. Каравай, В.С. Подлазов

СИСТЕМНАЯ СЕТЬ С ПОВЫШЕННЫМ ЧИСЛОМ УЗЛОВ И ПУТЕЙ. ОБОБЩЕННЫЕ РАСШИРЕННЫЕ МУЛЬТИКОЛЬЦА И «СПЛЮЩЕННЫЕ» БАБОЧКИ

Институт проблем управления им. В.А. Трапезникова РАН, г. Москва, mkaravay@ipu.ru, podlazov@ipu.ru

Введение

Системная сеть «сплющенная» бабочка (*Flattened Butterfly*) с n шагами (*FBn*) [1] считается перспективной системной сетью [2] для создания сетей связи в суперкомпьютерах на базе многопортовых коммутаторов типа 64-портового *YARC* [3]. Эта сеть получается «сплющиванием» n-каскадной k-ичной бабочки в плоскую сеть, при котором все коммутаторы с одинаковыми номерами в разных каскадах объединяются в один расширенный коммутатор, а симплексные каналы между каскадами бабочки объединяются в дуплексные каналы между разными расширенными коммутаторами.

Общее число абонентов (процессоров), объединяемых *FBn,* составляет $N_n = k^n$, а ее диаметр (число расширенных коммутаторов на пути между двумя абонентами) – $D_n = n$. Сеть *FBn* состоит из $M_n = N_n/k = k^{n-1}$ расширенных коммутаторов, каждый из которых содержит n коммутаторов $k \times k$ и имеет $m_n = n(k–1)+1$ дуплексных портов. Из них k портов используются для подсоединения k абонентов и $r_n = (n–1)(k–1)$ сетевых портов – для связи дуплексными каналами с другими составными коммутаторами сети. Число сетевых дуплексных каналов в *FBn* составляет величину $R_n = r_n M$.

Сеть *FBn* наследует коммутационные свойства сети n-каскадная k-ичная бабочка и поэтому не является ни неблокируемой, ни даже перестраиваемой [1]. Для выравнивания загрузки в ней используется дополнительные алгоритмы маршрутизации, которые повышают вдвое эффективный диаметр сети (реальные задержки передачи).

В сети *FBn* допускается прохождение каскадных коммутаторов в расширенных узлах в любой последовательности. Такая сеть изоморфна k-ичному $(n–1)$-мерному обобщенному гиперкубу, в котором к каждому узлу подсоединено k абонентов, а не 1, как обычно считается.

В докладе рассматривается задача увеличения числа абонентов сети по сравнению с *FBn* при сохранении неизменным числа портов узлов, числа сетевых каналов и диаметра сети. Увеличение числа узлов осуществляется за счет замены обобщенного гиперкуба на обобщенное расширенное мультикольцо. 2-мерные расширенные мультикольца предложены в [4]. В данной работе они применяются для построения обобщенных расширенных мультиколец 4-х измерений.

Мультикольца и их свойства

Мультикольцом мы называем набор r коммутируемых симплексных колец с разными шагами. По определению [5], мультикольцо объединяет N узлов степени r и состоит из набора r колец с разными шагами ${}^1S=1$, ${}^2S$,..., ${}^rS$, где ${}^iS \neq {}^jS$. Это означает, что из каждого узла выходят r дуг с «длинами» ${}^1S=1$, ,..., ${}^rS$, где длина дуги определяется разницей по $\bmod N$ номеров инцидентных ей узлов. Будем характеризовать такое мультикольцо набором C_r длин шагов колец $C_r = ({}^1S,...,{}^rS)_r$ и парой $\{N, C_r\}$.

Полным k-ичным t-мерным мультикольцом MRt мы называем мультикольцо, в котором $r_t = (k–1)t$, $N_t = k^t$ и C_r состоит из колец с шагами ${}^{m+l-1}S = mk^{l-1}$ $(1 \le m \le k-1, 1 \le l \le t)$, т.е. из t наборов колец, в котором l-й набор содержит кольца с шагами $\{k^{l-1}, 2k^{l-1}, ..., (k–1)k^{l-1}\}$, представляемых значениям разрядов в k-ичной системе счисления.

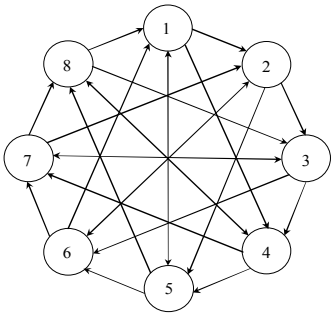
В полном мультикольце любой маршрут состоит не более чем из t дуг разных колец – не более одной из каждого кольца. Поэтому его диаметр определяется как $D_t = t+1$.

В сети с топологией полного мультикольца к каждому узлу подсоединен один абонент (процессор), т.е. каждый узел содержит $r+1$ дуплексных портов. Если подсоединить к каждому узлу k абонентов, то мы получим обобщенное мультикольцо $GMRt$, являющееся точным аналогом сети FBn при $t=n–1$. Сеть $GMRt$ содержит $N_t = k^{t+1}$ абонентов и $M_t = k^t$ узлов с $m_t = (k–1)t+k$ дуплексными портами, из которых $r_t = (k–1)t$ являются сетевыми, и имеют диаметр $D_t = t+1$.

Узел 2-мерного k-ичного мультикольца содержит проходной коммутатор $(k–1)\times(k–1)$, входной мультиплексоры $1\times2(k–1)$, 1×2 и выходной демультиплексоры $2(k–1)\times1$, 2×1. Путь от абонента-источника к абоненту-приемнику проходит через разные схемы в 3-х узлах – мультиплексор в источнике, коммутатор в промежуточном узле и демультплексор в приемнике. Поэтому диаметр $D_2 = 3$.

Так называемое расширенное 2-мерное мультикольцо $ER2$ [4] состоит из узлов с $p_2=2(k–1)$ сетевыми дуплексными портами, как и 2-мерное k-ичное мультикольцо. Однако в нем шаги колец подобраны таким образом, чтобы обеспечить максимальное число узлов P_2 и иметь не менее σ разных путей между любыми двумя узлами. Расширенное 2-мерное мультикольцо имеет диаметр $D_2 = 3$. На рисунке приводится пример расширенного 2-мерного мультикольца с $P_2 = 9$ узлами и $\sigma=1$ путем, состоящего из трех колец.

Будем задавать расширенное 2-мерное мультикольцо тройкой $\{P_2, \sigma, C_p\}$, где C_p – это шаги набора p_2 симплексных колец, составляющих мультикольцо. В частности, мультикольцо на рисунке задается тройкой $\{9, 1, (1, 3, 4)_3\}$.



Расширенное мультикольцо ER2 на с шагами колец 1, 3, 4

Можно привести еще пример мультикольца $\{35, 1, (1, 6, 7, 10, 28, 29, 34)_8\}$. Оно задается также набором 4 дуплексных колец с шагами $(\pm1,\pm6,\pm7,\pm10)$. Для четных p_2 существуют также мультикольца с несколькими разными путями между узлами. Это, например, мультикольца $\{42, 2, (\pm3,\pm6,\pm7,\pm12,\pm14)_{10}\}$ и $\{37, 3, (\pm1,\pm3,\pm4,\pm9,\pm11,\pm17)_{12}\}$.

В таблице 1 приводятся сравнительные характеристики двумерных расширенного и обычного мультиколец, где $c_1=P_2/N_2$. Из значений таблицы 1 можно сделать вывод, что при $\sigma = 1$ для больших k значение P_2 можно оценивать снизу эмпирической формулой $P_2 \approx c_1k^2$, где c_1=1,3.

Таблица 1

Характеристики расширенных мультиколец $\{P_2, 1, C_p\}$ и k-ичных мультиколец $\{N_2, C_r\}$, $N_2= k^2$ и $r = 2(k–1)= p_2$ при $P_2 \geq 1{,}3N_2$

p_2	5	6	7	8	9	10	12	14	16	18
k	3	4	4	5	5	6	7	8	9	10
P_2	17	21	30	35	43	51	67	85	108	130
N_2	12	16	20	25	39	36	49	64	81	100
c_1	1,42	1,31	1,50	1,40	1,43	1,42	1,37	1,33	1,33	1,3

В таблице 2 приводятся характеристики расширенных мультиколец с $\sigma > 1$. Исходя из этих характеристик можно сделать вывод, что при $\sigma = 2$ для больших k значение P_2 оценивается формулой $P_2 \approx c_2k^2$, где c_2 = 1,1, а при $\sigma = 3$ $–P_2 \approx c_3(k–1)^2$, c_3 = 0,95.

Таблица 2

Характеристики расширенных мультиколец $\{P_2, \sigma, C_p\}$

Число путей	$r=p_2$	4	6	8	10	12	14	16	18
$\sigma=2$	P_2	7	16	27	42	53	69	89	114
	N_2	9	16	25	36	49	64	81	100
$\sigma=3$	P_2	6	13	19	29	37	49	61	78
	N_2	4	9	16	25	36	49	64	81

Расширенное мультикольцо с $\sigma > 1$ имеет $(\sigma–1)$-отказоустойчивость по кольцам и допускает σ-кратную рандомизацию маршрутов с целью выравнивания загрузки колец. Здесь под $(\sigma–1)$-отказоустойчивостью понимается сохранение работоспособности и функциональности сети (диаметра и задержек) при отказе $\sigma–1$ кольца. Сохранение работоспособности возможно и при адаптивной маршрутизации, но за счет увеличения диаметра сети.

Подсоединение каждому узлу не одного, а $k > 1$ абонентов порождает обобщенные 2-мерные сети – k-ичное мультикольцо *GMR2* или расширенное мультикольцо *GER2* соответственно. Для этого в каждом узде *GMR2* мультиплексор $1\times2(k–1)$ и демультиплексор $2(k–1)\times1$ в каждом узле заменяются на коммутаторы $k\times2(k–1)$ и $2(k–1)\times k$. *GMR2* содержит $N_2^* = k^3$ узлов и имеет диаметр $D^* = 3$, а каждый узел имеет $r_2^* = 2(k–1)$ сетевых дуплексных портов.

В узле обобщенного расширенного мультикольца *GER*2 вместо коммутатора $(k–1)\times(k–1)$ необходимо использовать коммутатор $p_2\times p_2$, где $p_2 = 2(k–1)$. Такое обобщенное 2-мерное расширенное мультикольцо имеет с $P_2^* = P_2k = c_\sigma k^3$ узлов и диаметр $D_2^* = 3$. Каждый узел имеет $p_2^* = 2(k–1)$ сетевых портов и в целом $m_2^* = 2(k–1)+k$ портов.

Теперь сравним сети *FB*3 и *GER*2 при одинаковых размерах узлов и диаметрах. В таблице 3 они сравниваются по числу абонентов. В строках с $p_2^* > 18$ приводятся расчетные данные по эмпирической формуле $P_2^* \approx c_\sigma k^3$, так как соответствующие расширенные кольца еще не построены [4]. Видно, что при $\sigma = 1$ число абонентов в *GER*2 более чем на 30 % больше, чем в сети *FB*3.

Таблица 3

Число абонентов FB3 и GER2 при $m_3=m_2^*$ и $D_3=D_2^*=3$

*FB*3			*GER*2 ($\sigma = 1$)		*GER*2 ($\sigma = 2$)	*GER*2 ($\sigma = 3$)
k	m_3	N_3	p_2^*	P_2^*	P_2^*	P_2^*
6	16	216	10	306	252	174
8	22	512	14	688	552	392
10	28	1000	18	1300	1100	780
12	34	1728	22	≈2240	≈1900	≈1380
14	40	2744	26	≈3560	≈3010	≈2240

Специально отметим, что *GER*2 при $\sigma > 1$ обладает $(\sigma-1)$-отказоустойчивостью по кольцам и допускает σ-кратную рандомизацию маршрутов.

Обобщенные расширенные мультикольца размерности 4

Теперь возьмем мультикольцо *MRt* произвольной четной размерности $t=2l$ (l=1, 2, ...) и будем использовать кольца измерений l–1 и l также как в сети *ER*2, т.е. как в расширенном мультикольце. Кольца 1-го и 2-го измерений имеют длины шагов сети *ER*2, т.е. ${}^{1}S = 1$, ..., ${}^{p_2}S$. Кольца l-го и (l+1)-го измерений имеют длины шагов сети *ER*2, увеличенные в $P_2{}^l$ раз, т.е. ${}^{1+p_2}S = P_2^l$,..., ${}^{2p_2}S = {}^{p_2}SP_2^l$. Таким образом мы построим сеть *ERt,* которая имеет узлы с тем же числом портов и имеет тот же диаметр $D_t = t+1$, что и сеть *MRt,* но объединяет $P_t=P_2{}^l$ узлов.

Если в сети *ERt* к каждому узлу подсоединить не одного абонента, а k абонентов, то получится сеть обобщенное расширенное мультикольцо *GERt*. Каждый ее узел содержит $m_t^* = 2(k-1)l+k=(t+1)(k-1)+1$ дуплексных портов. Это означает, что сети *GERt* и *FBn* при одинаковых k и $n = t+1$ имеют одинаковое число портов в узлах и в расширенных узлах соответственно. Сеть *FBn* имеет диаметр $D_n = n$, а сеть *GERt* – $D_t = t+1$, т.е. их диаметры одинаковы. Однако сеть *GERt* может объединять бо́льшее число абонентов $P_t^* = kP_t^* \approx c_\sigma{}^l k^{t+1}$ при $\sigma = 1, 2$ или иметь большие пропускную способность и быстродействие при $\sigma = 2, 3$.

Узел сети *GER*4 состоит их 4 коммутаторов $p_2 \times p_2$, входного коммутатора $k \times 2p_2$ и выходного коммутатора $2p_2 \times k$. Этот узел допускает возможность прохождение колец на маршруте в любой последовательности. При $\sigma > 1$ это означает наличие в *GER*4 σ^2 разных маршрутов между любой парой узлов. Таким образом, *GER*4 обладает (σ^2-1)-отказоустойчивостью по кольцам и имеет возможность выравнивания загрузки колец за счет σ^2-кратной рандомизации маршрутов. Последняя возможность позволяет отказаться от дополнительной маршрутизации с целью выравнивании трафика и поэтому сократить почти вдвое задержки передачи по сравнению с *FB*5.

Сравним сети *FB*5 и *GER*4 по числу абонентов при одинаковом числе портов в узлах. Результаты сравнения представлены в таблице 4. Видно, что при $\sigma = 1$ число абонентов в *GER*4 почти на 70 % больше, чем в сети *FB*5, а при $\sigma = 2$ больше на 20 %. Заметим, что по числу абонентов сеть *GER*4 покрывает потребности современных суперкомпьютеров.

Таблица 4

Число абонентов FB5 и GER4 при D=5 и равных узлах

FB5			*GER4* ($\sigma = 1$)		*GER4* ($\sigma = 2$)	*GER4* ($\sigma = 3$)
k	m_5	N_3	p_2^*	P_4^*	P_4^*	P_4^*
6	26	7776	10	15606	10584	5046
8	36	32768	14	57800	38088	19208
10	46	100000	18	169000	129960	52290
12	56	248832	22	≈420000	≈300000	≈176000
14	66	537824	26	≈910000	≈650000	≈400000

Заключение

В докладе предложена новая плоская системная сеть с узлами в виде составных многопортовых коммутаторов, аналогичная сети «сплющенная» бабочка. В ней осуществляется сдваивание измерений многомерного мультикольца и замещение наборов колец полученных пар оригинальным двумерным расширенным мультикольцом, которое обеспечивает большее число абонентов и/или большее число разных маршрутов при равном числе портов узлов и равных диаметрах системной сети.

В докладе рассмотрены сравнительные характеристики сетей, обобщенное расширенное мультикольцо и сплющенная бабочка с малыми диаметрами, которые требуются для построения системных сетей современных суперкомпьютеров. Показано, что для диаметра 5 обобщенное расширенное мультикольцо обеспечивает на 70 % большее число узлов или до двух раз меньшие задержки передачи по сравнению со сплющенной бабочкой.

1. *Kim J., Dally W. J. and Abts D.* Flattened Butterfly: A Cost-Efficiently Topology for High-Radix Networks // Proc. 34th Intern. Symp. Comp. Archit. (ISCA'2007). – 2007. – P. 126 – 137.
2. *Корж А.А.* Инновационная платформа A-Class для мультипетафлопсных систем // Тр. суперкомп. конф. «Научный сервис в Интернет: многообразие суперкомпьютерных миров». – 2014.
3. *Scott S., Abts D., Kim J. and Dally W.* The Black Widow *High-radix Clos Network* // Proc. *33rd* Intern. Symp. Comp. Arch. (ISCA'2006). URL: http://cva.stanford.edu/people/ jjk12/isca06.pdf.
4. *Подлазов В.С.* Расширенное мультикольцо с диаметром 2 // Проблемы управления. – 2015. – № 4. – С. 35 – 40.
5. *Алленов А.В., Подлазов В.С.* Пропускная способность набора кольцевых каналов II. Кольцевые коммутаторы // Автоматика и телемеханика. – 1996. – № 4. – С. 162 – 172.

А.В. Колодзей

ГИБРИДНЫЕ ТОПОЛОГИИ КОММУНИКАЦИОННОЙ СРЕДЫ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ЗАО «ИнформИнвестГрупп», г. Москва,
ak@iigroup.ru

Введение

Вычислительная мощность проблемно-ориентированных вычислительных систем (ПОВС) в первую очередь определяется числом вычислительных узлов (ВУ), процессоров, ядер и использованием различных ускорителей вычислений (УВ), таких как графические ускорители Graphics Processing Unit (GPU) и программируемые логические интегральные схемы (ПЛИС, FPGA – Field-Programmable Gate Array). Однако эффективность использования ПОВС и их реальная производительность на многих задачах зависит от реализации коммуникационных сетей. Для задач с малым количеством обменов по сети, которые легко разбиваются на независимые части, не требуется специализированных решений, достаточно использование обычных коммерческих решений типа Gigabit Ethernet.

Производительность суперкомпьютеров на задачах с интенсивным обменом данными между узлами, в основном, определяется производительностью коммуникационной сети. Здесь наиболее широко используются топологии «толстое дерево» (Fat tree) или «многомерный тор», технологии Ethernet, InfiniBand [1]. На ПОВС уровня от вычислительной стойки 19” 42U до нескольких десятков таких стоек происходит вынужденный переход от Gigabit Ethernet к 10G Ethernet, который вносит существенный вклад в общую стоимость системы, не говоря о более производительных решениях. В условиях войны санкций и требований по импортозамещению актуальной является задача разработки отечественных вариантов коммуникационных сетей. К таким сетям можно отнести, например, разрабатываемую в НИЦЭВТ высокоскоростную коммуникационную сеть «Ангара», имеющую топологию «четырехмерный тор» и предназначенной для использования в качестве отдельной сети с адаптерами в виде плат-расширений PCI Express для кластерных систем [2].

В докладе обсуждаются «бюджетные» варианты построения коммуникационной сети для ПОВС в виде кластеров ВУ с установленными в них УВ на основе FPGA. Рассматривается вариант, когда УВ одновременно выполняют и функции маршрутизации.

Возможные варианты топологии коммуникационной среды

Обычная коммуникационная сеть состоит из узлов, в каждом из которых есть сетевой адаптер, соединенный с одним или несколькими

маршрутизаторами, которые в свою очередь соединяются между собой высокоскоростными каналами связи (линками). Структура сети, определяющая, как именно связаны между собой узлы системы, задается топологией сети (обычно решетка, тор или толстое дерево) и набором структурных параметров: количество измерений, количество уровней дерева, размеры сторон тора, число коммутаторов на уровнях дерева, число узлов сети, портов у маршрутизаторов [3].

Если в каждом ВУ сетевые адаптеры суммарно имеют $2n$ выходов, то это позволяет строить коммуникационную сеть с топологией «n – мерный тор», просто соединяя адаптеры между собой, например, оптическими кабелями. В целях увеличения плотности портов возможно использование промышленного стандарта QSFP (QSFP+), позволяющего использовать скорости передачи данных от 40 Гбит/с и более (4 канала по 10 Гбит/с). Однако использование таких соединений для каждой отдельной связи в решетке/торе представляется избыточным. В этом случае можно при помощи QSFP кабелей соединять сетевые адаптеры с пассивным коммуникационным блоком (КБ), а необходимую маршрутизацию осуществлять внутри каждого сетевого адаптера и внутри ВУ, если в нем несколько сетевых адаптеров.

Предлагаемая номенклатура сетевого оборудования ПОВС выглядит следующим образом:

- 4-канальный адаптер QFSP, 40 Гбит/с, интерфейс PCIe;
- пассивный кросс-бокс;
- кабель, перемычка, заглушка QFSP.

В одном ВУ может располагаться один адаптер или более. Каждый адаптер соединяется 4-канальным кабелем со своим 4-канальным входным портом кросс-бокса. Входные порты по заранее выбранной подстановке S уже одноканальными оптическими проводами соединяются либо между собой, либо с 4-канальным коммуникационными портами. Коммуникационных портов не больше (но может быть меньше), чем входных портов. Коммуникационные порты соединяются между собой перемычками или кабелями с входными портами других кросс-боксов. Заглушки по некоторой подстановке s перемыкают каналы для одного порта. Маршрутизация происходит внутри адаптеров и между разными адаптерами внутри одного ВУ.

Понятно, что для любого фиксированного числа ВУ, не превышающего максимально возможного числа портов кросс-бокса, мы всегда можем подобрать такую подстановку S, которая обеспечит наперед заданную топологию коммуникационной среды и без использования коммуникационных портов. Коммуникационные порты позволяют при необходимости масштабировать ПОВС как вверх, так и вниз и/или ре-

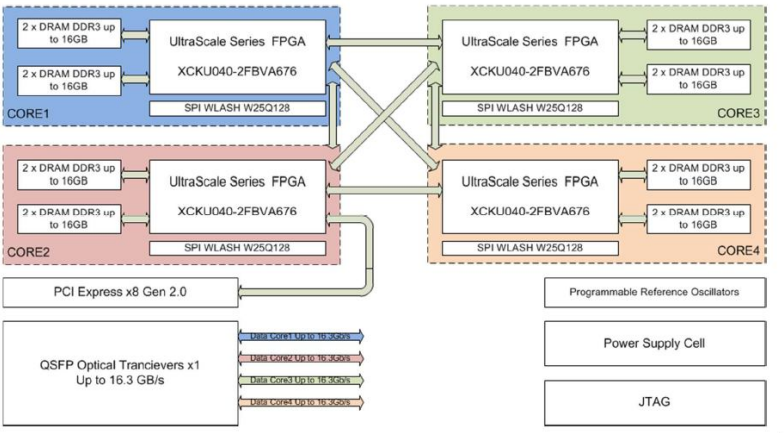
конфигурировать топологию коммуникационной среды ПОВС, устраняя выявленные узкие места.

Выбор топологии коммуникационной среды определяется классами решаемых задач. Обычно рассматриваются задачи, когда один узел рассылает данные всем остальным, либо, наоборот, все узлы шлют данные одному, либо когда все узлы посылают данные случайным адресатам [3]. Кроме того, предлагается рассматривать такой класс задач. Узлы разбиваются произвольным образом на рабочие группы, и данные случайным образом пересылаются от каждого узла к каждому внутри рабочей группы.

На этапе проектирования ПОВС в соответствии с требования Заказчика определяется типовая смесь решаемых задач и возможные пределы масштабирования ПОВС на всех этапах жизненного цикла. После этого путем имитационного моделирования выбирается оптимальная топология ПОВС. Как вариант, предлагается обеспечивать топологию «n-мерный тор» внутри вычислительных стоек, а соединения между вычислительными стойками при необходимости реконфигурировать по ходу эксплуатации и дальнейшего развития ПОВС. Предлагаемая номенклатура сетевого оборудования позволяет решать указанные задачи.

Пример реализации

Рассмотрим ПОВС, построенную в виде кластера ВУ, использующих УВ на основе ПЛИС. В ЗАО «ИнформИнвестГрупп» разработан УВ C8X4OPTO формата PCI Express, несущий 4 ПЛИС Kintex UltraScale. Блок-схема УВ приведена на рисунке.



Блок-схема ускорителя вычислений C8X4OPTO

Плата двойной толщины, ¾ – длины, активное охлаждение (радиатор + вентилятор), потребляемая мощность – не более 150 Вт. Форм-фактор платы и требования к вычислительным серверам, в том числе по разъемам дополнительного питания – такие же, как и для ускорителей вычислений на базе GPU. Таким образом, в ВУ может быть размещено такое же количество ускорителей вычислений С8Х4ОРТО, что и ускорителей на базе GPU. Возможны гибридные решения, когда ускорители на базе GPU и ПЛИС смешиваются в ПОВС тем или иным образом.

В зависимости от выбранной модели ВУ в нём может быть размещено от 1 до 8 ускорителей вычислений. Кроме того, возможен такой вариант, когда внутри ВУ размещается одна или несколько интерфейсных плат, а УВ размещаются во внешних вычислительных блоках (ВБ).

В качестве интерфейсной платы в ЗАО «ИнформИнвестГрупп» разработана плата С4Х4ОРТО, PCI Express, половинная длина, полная высота, одинарная толщина, оптический интерфейс до 4х10 Гбит/с, содержащая одну ПЛИС Kintex-7 410, потребляемая мощность не более 40 Вт, активное охлаждение. Указанная интерфейсная плата также одновременно может выполнять функции ускорителя вычисления. При таком подходе снижаются требования к ВУ. ВБ при этом должен обеспечить электропитание и охлаждение УВ. Сами УВ не обязательно нуждаются в использовании интерфейса PCIe, а в состоянии работать только по оптическому интерфейсу.

УВ С8Х4ОРТО, обладающий полносвязным графом соединения 4х ПЛИС между собой высокоскоростными последовательными линками и заметным объемом оперативной памяти, в состоянии помимо своей счетной роли, выступать в роли маршрутизаторов. Формирование таблиц маршрутизации и пр. может осуществляться уже в самих ВУ. Предполагается, что для обеспечения мониторинга, управления, первоначального конфигурирования и пр. УВ также соединены обычным Gigabit Ethernet.

Заключение

Предложенный подход к построению коммуникационной сети ПОВС можно назвать «кластерным». Когда каждый ВУ «вносит свою лепту» в систему маршрутизации, а все остальное сетевое оборудование – относительно дешевое пассивное. В качестве многоканальных сетевых адаптеров предлагается использовать сами ускорители вычислений, а вместо активных маршрутизаторов – пассивные кросс-блоки.

Такой подход позволяет существенно снизить стоимость сетевого оборудования по сравнению с использованием активного сетевого оборудования типа 10G Ethernet и выше.

За счет использования трансиверов стандарта QSFP удается повысить плотность портов и уменьшить размеры УВ по сравнению с использованием трансиверов SFP.

По сравнению с классической топологией «n-мерный тор» использование кросс-блоков позволяет гибко выбирать топологию коммуникационной среды ПОВС на этапах проектирования и изготовления, обеспечивает масштабирование как вверх, так и вниз, а также относительно простую возможность реконфигурирования топологии вычислительной сети в ходе эксплуатации ПОВС в зависимости от решаемых классов задач.

1. Список Топ500, июнь 2015 – www.top500.org/list/2015/06/
2. *Слуцкин А.И., Симонов А.С., Жабин И.А., Макагон Д.В., Сыромятников Е.Л.* Разработка межузловой коммуникационной сети ЕС8430 «Ангара» для перспективных российских суперкомпьютером – НИЦЭВТ. www.nicent.ru/publications/item/77-разработка-межузловой-коммуникационной-сети-ec8430-ангара-для-перспективных-российских-суперкомпьютеров.
3. *Макагон Д.В., Сыромятников Е.Л.* Сети для суперкомпьютеров // Открытые системы. – №7. – 2011. http://www.osp.ru/os/2011/07/13010500/

В.К. Левин, В.С. Горбунов, Г.С. Елизаров

СОВРЕМЕННЫЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

ФГУП «НИИ «Квант», г. Москва,
info@rdi-kvant.ru

Специфика высокопроизводительных вычислительных систем (суперкомпьютеров) как некоторого класса средств вычислительной техники определяется их предназначением – решение задач, которые не могут быть решены иными способами по причине высокой вычислительной сложности и большого объёма обрабатываемых данных. Выполнение больших расчетов и компьютерное моделирование сложных конструкций и процессов востребовано при проведении современных исследований и разработок в различных областях.

Ключевым фактором достижения высочайшей производительности суперкомпьютерных установок являлся уровень технологического производства микроэлектронных изделий. Значимость уровня микроэлектронного производства в последнее время усиливается по причине возросшей мощности энергопотребления суперкомпьютерных установок. Так, если сегодня энергопотребление суперкомпьютера в первых позициях списка Top-500 составляет порядка 15 МВт, то сейчас уже признается, что к 2020 г. суперкомпьютер с прогнози-

руемой на тот период производительностью 1 Exaflops может потреблять более 30 МВт.

В этих условиях практически единственным способом создания отечественного суперкомпьютера является его проблемная ориентация (специализация). Специализация суперкомпьютеров отражается в тенденции применения гибридных вычислений – сочетании процессоров общего назначения и процессоров-ускорителей для разного рода операций и процедур. В суперкомпьютерных центрах используются гетерогенные установки или комплексы систем различной специализации – сильно связанные вычислительные структуры, гибридные вычислители, упрощенное объединение множества процессоров, большая память.

Для специализации суперкомпьютеров можно использовать специальные сопроцессоры и реконфигурируемые вычислительные структуры. Неплохие показатели энергоэффективности демонстрируют системы на базе СБИС с перенастраиваемой логической структурой – ПЛИС (FPGA – field-programmable gate arrays). Элементно-конструктивная база таких суперкомпьютеров в России давно освоена. Одним из способов специализации суперкомпьютеров также является создание заказных проблемно-ориентированных и специализированных сопроцессоров, в которых эффективность использования аппаратуры может быть на уровне 70 – 80 %. Это, по всей вероятности, поможет решить проблему импортозамещения при создании отечественных суперкомпьютерных установок, что позволяет на реальных задачах значительно сократить отставание в технологической и микроэлементной базе.

На европейской суперкомпьютерной конференции был объявлен новый лидер списка TOP-500 – суперкомпьютер SunWay TaihuLight, созданный на базе процессора собственной разработки. Пиковая производительность суперкомпьютера составила более 125 PetaFlops. Производительность процессора, который разработали китайские инженеры, составила более 3-х teraflops, что находится на уровне лучших графических ускорителей компаний AMD и Nvidia. Достижение такого уровня производительности – результат специализации этого процессора на определенный круг решаемых задач.

В российских организациях накоплен значительный опыт и сформирован потенциал в разработках, поставках и применениях суперкомпьютеров. Это позволяет ставить задачи построения в России суперкомпьютеров с усиленной специализацией на наиболее важный круг решаемых задач с использованием отечественной элементно-конструктивной и микроэлектронной базы.

И.И. Левин[1], А.И. Дордопуло[2], А.М. Федоров[1]

ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ РЕКОНФИГУРИРУЕМАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА С ЖИДКОСТНЫМ ОХЛАЖДЕНИЕМ*

[1] ООО «НИЦ супер-ЭВМ и нейрокомпьютеров»,
[2] НИИ многопроцессорных вычислительных систем ЮФУ,
г. Таганрог,
scorpio@mvs.sfedu.ru

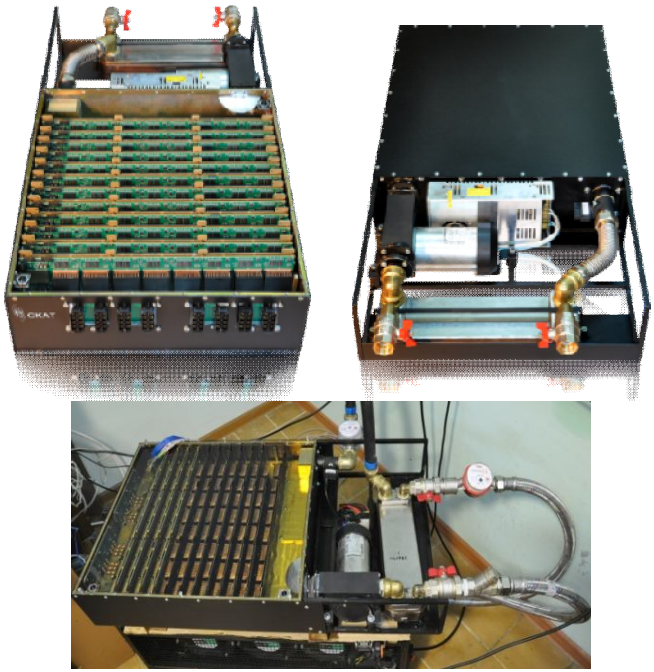
Одной из ключевых проблем современных вычислительных систем является охлаждение проектируемых высокопроизводительных вычислительных комплексов. Использование воздушных систем охлаждения для разрабатываемых суперкомпьютеров, в том числе вычислительных систем гибридного типа, практически достигло своего предела, что обусловлено снижением эффективности охлаждения с ростом потребляемой (и отдаваемой) мощности, вызванным увеличением степени интеграции кристаллов микропроцессоров и других микросхем. Поэтому большинство разработчиков вычислительной техники видят выход в применении систем жидкостного охлаждения как закрытого типа (без прямого контакта между жидкостью и электронными компонентами на печатных платах), так и открытого типа (погружные, в которых жидкость непосредственно омывает электронные компоненты).

Для реконфигурируемых вычислительных систем (РВС), содержащих не менее 6-8 кристаллов программируемых логических интегральных схем (ПЛИС) на одной печатной плате, наиболее перспективным является применение систем жидкостного охлаждения открытого типа с непосредственным погружением плат вычислительных модулей с электронными компонентами в жидкостный хладагент. Существующие технологии (например, технология IMMERS) не предназначены для охлаждения вычислительных модулей, содержащих большое число тепловыделяющих элементов. Это существенно затрудняет применение готовых решений для охлаждения вычислительных модулей РВС. Поэтому в НИЦ СЭ и НК разработан вычислительный модуль РВС с жидкостным охлаждением открытого типа оригинальной конструкции для печатных плат с высокой плотностью компоновки. Каждая печатная плата РВС содержит восемь ПЛИС семейства Virtex UltraScale логической емкостью не менее 100 млн эквивалентных вентилей и тепловыделением до

* Работа выполнена при частичной финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии № 14.578.21.0006 от 05.06.2014, уникальный идентификатор RFMEFI57814X0006

100 Вт каждая. Вычислительный модуль РВС с жидкостным охлаждением конструктивно состоит из рабочей секции, в которой размещено 12-16 плат вычислительных модулей с потребляемой мощностью до 850 Вт каждая, и теплообменной секции, в которой находятся насосная группа и теплообменник охлаждения хладагента. Все платы и электронные компоненты рабочей секции полностью погружены в электрически нейтральный жидкостный хладагент на основе диэлектрического минерального масла, разработанный по специальной технологии.

Для отработки технических и технологических решений разрабатываемой высокопроизводительной реконфигурируемой вычислительной системы с жидкостным охлаждением был создан ряд макетов, опытных и технологических образцов. Фотография технологического образца вычислительного модуля РВС «Скат» с жидкостным охлаждением представлена на рисунке.



Технологический образец вычислительного модуля РВС «Скат» с жидкостным охлаждением

На технологическом образце вычислительного модуля с жидкостным охлаждением были исследованы тепловые режимы работы элек-

тронных компонентов в различных режимах функционирования, по результатам которых температура кристаллов ПЛИС не превышает 56º С при тепловой нагрузке 91 Вт.

Вычислительный модуль РВС с жидкостным охлаждением предназначен для установки в стандартный 19” вычислительный шкаф, который подключается к промышленным воздушно-водяным системам охлаждения на основе чиллеров. В один вычислительный шкаф высотой 47U может устанавливаться до 12 вычислительных модулей РВС с жидкостным охлаждением высотой 3U, что позволяет создать РВС сверхпетафлопсной производительности в пределах одного вычислительного шкафа.

1. *Kalyaev I.A., Levin I.I., Semernikov E.A., Shmoilov V.I.* Rekonfiguriruyemiye multikonveyerniye vichislitelniye struktury. [Reconfigurable multipipeline computing structures] 2nd edition, revised and supplemented / Edited by I.A. Kalyaev. – Rostov-on-Don: SSC RAS Publishing, 2009. – 344 pp.
2. *Kalyaev I.A., Levin I.I., Dordopulo A.I., Slasten L.M.* Reconfigurable Computer Systems Based on Virtex-6 and Virtex-7 FPGAs // IFAC Proceedings Volumes, Programmable Devices and Embedded Systems. – Vol. №12. – Part №1. – 2013. – P. 210 – 214.
3. *Igor A. Kalyaev, Ilya I. Levin, Alexey I. Dordopulo, Liuba M. Slasten.* FPGA-based Reconfigurable Computer Systems // Science and Information Conference (SAI) (Oct 7-Oct 9 2013). – London: UK. – 2013. – P. 148 – 155.

Н.А. Лукин

АНАЛОГО-ЦИФРОВЫЕ ФУНКЦИОНАЛЬНО-ОРИЕНТИРОВАННЫЕ ПРОЦЕССОРЫ ДЛЯ ВСТРОЕННЫХ СУПЕРКОМПЬЮТЕРОВ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ*

Институт радиоэлектроники и информационных технологий
Уральского федерального университета,
НПО автоматики имени академика. Н.А. Семихатова,
Институт машиноведения УрО РАН,
г. Екатеринбург,
n.a.lookin@urfu.ru

Встроенные суперкомпьютеры в системах реального времени

Развитие вычислительных систем происходит в настоящее время по двум основным направлениям – ВС общего назначения и ВС, встроенные в системы реального времени. Первое направление – это суперкомпьютеры, ориентированные на обеспечение максимальной производительности в ходе решения задач пользователей. Универсализм архи-

* Работа выполнена частично за счет Научного гранта УрО РАН, проект № 17-7-1-20

тектуры вычислительных кластеров приводит к высоким значениям накладных расходов на обработку данных. Кроме того, гигагерцовые частоты переключения элементов в СБИС, количество которых исчисляется уже порядком миллионов, делают проблему отвода тепла от аппаратуры суперкомпьютеров чрезвычайно трудоемкой [1]. Поэтому дальнейшее повышение производительности суперкомпьютеров в настоящее время признается специалистами весьма проблематичным. Более того, достижение очередного рубежа производительности 10^{18} оп/сек (экзафлоп) требует принятия как в архитектуре, так и в схемотехнике и конструкции принципиально новых решений [2]. Все больше экспертов сходится во мнении, что достижение предельных величин производительности суперкомпьютеров в рамках существующих архитектур процессоров и систем потребует создания, по сути дела, новой микроэлектроники, причем далеко не очевидно, что на основе существующих полупроводниковых технологий. А это уже может означать необходимость создания новой индустрии, что тем более является проблематичным.

В то же время стремительно развиваются системы реального времени различного (СРВ) назначения. Главной тенденцией такого развития является тотальное их оснащение микропроцессорами различных архитектур. Интересно, что количество как самих СРВ, так и типов (классов) их архитектур продолжает увеличиваться. В итоге подавляющая часть выпускающихся микропроцессоров и других видов микроэлектронных компонентов предназначена для вычислительных систем СРВ (ВС РВ). Для целого ряда СРВ как гражданского, так и военного назначения уже в настоящее время требуется обеспечение производительности, сравнимой с суперкомпьютерной [3]. Необходимо также учесть, что требования сверхвысокой производительности сопровождается требованиями на минимальные массу, габариты и потребляемую мощность. Это не только делает все более актуальными исследования принципов эффективных вычислений в условиях ограничений, но и требует существенно более быстрого перехода к нестандартным архитектурам, чем в случае суперкомпьютеров общего назначения. При этом главной особенностью встроенных суперкомпьютеров (ВСКОМП) является ориентация на обеспечение максимальной эффективности решения задач конкретной предметной области.

Базовая структура ВСКОМП

Несмотря на проблемную (предметную) ориентацию ВСКОМП актуальной является проблема унификации принимаемых решений как на уровне архитектуры в целом, так и в части отдельных компонент. Это обстоятельство диктуется как экономическими соображениями (невозможно создавать элементную базу и конструкцию только для одного ВСКОМП), так и требованиями на основе разработанных решений проектировать мо-

дификации суперкомпьютеров для различных применений. Поэтому актуальными являются исследования базовых структур ВСКОМП, реализующих алгоритмы СРВ. Рассмотрим в качестве примера базовую структуру ВСКОМП, предназначенного для реализации алгоритмов навигационных систем для подвижных объектов (ракета, самолет и т.п.).

На рис. 1 изображена базовая структура ВСКОМП для интегрированной навигационной системы.

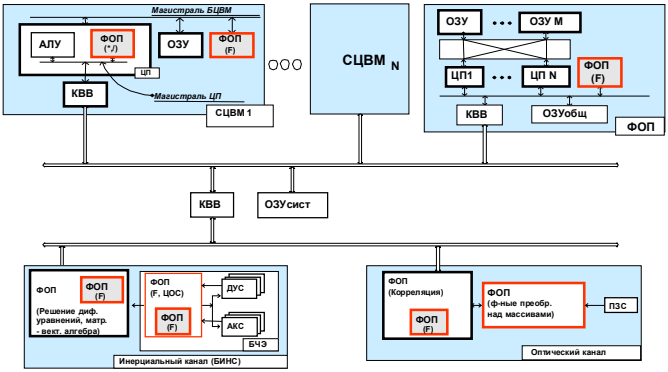


Рис. 1. Базовая структура встроенного суперкомпьютера интегрированной навигационной системы

Состав ВСКОМП:

- Многомашинная ВС, состоящая из множества идентичных специализированных БЦВМ. Предназначена для решения навигационных задач общего плана – навигация, стабилизация, управление. Для достаточно простых систем (например, для навигации автомобиля) достаточно одной БЦВМ, в то время как для многоконтурных систем управления высокоточными ракетными комплексами может потребоваться до (6 – 8)-ти БЦВМ.
- Многопроцессорный кластер, состоящий их множеств процессоров и массивов модулей ОЗУ. Все процессоры и модули ОЗУ соединены программируемыми коммутаторами.

Такой кластер предназначен для реализации алгоритмов высокой вычислительной сложности в режиме жесткого реального времени (например, классификации динамических признаков в течение полета).

- Функционально-ориентированная ВС РВ, предназначенная для реализации алгоритмов решения системы кинематических уравнений в реальном времени в составе бесплатфор-

менной инерциальной навигационной системы (Инерциальный канал). Система состоит из двух ФОП, реализующих алгоритмы интегрирования дифференциальных уравнений, и алгоритмы цифровой обработки сигналов и вычисления набора математических функций.

- Функционально-ориентированная ВС РВ, предназначенная для реализации алгоритмов обработки изображений, поступающих из оптического датчика (например, ПЗС-матриц) Алгоритмы характеризуются значительной вычислительной сложностью (особенно, внутрикадровая обработка изображений), поэтому соответствующий ФОП может иметь двумерную (в ряде случаев – систолическую) архитектуру.

Суммарная производительность подобного ВСКОМП может достигать порядка терафлоп, в то время как его масса не должна превышать 3-5 кг при допустимой мощности потребления не больше 30 – 50 Вт. Анализ вычислительной сложности алгоритмов, реализуемых с помощью ВСКОМП показывает, что максимум приходится на алгоритмы первичной обработки сигналов в периферийных контурах СУ РВ. Количество потоков данных может достигать до 10 тыс. (например, для антенн с фазированными решетками), а объемы хранимой информации порядка гигабайт (для систем корреляционно-экстремальной навигации). Поэтому значительная часть нестандартных архитектурных и схемотехнических решений приходится на эти подсистемы ВСКОМП.

Первичная обработка данных в ВСКОМП

Практически весь объем первичной обработки данных может быть представлен как совокупность следующих этапов, показанных на рис. 2.

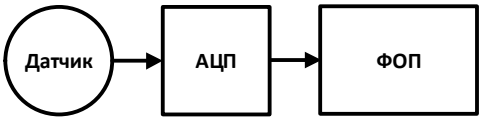


Рис. 2. Этапы первичной обработки сигналов в ВСКОМП

Основные особенности первичной обработки данных в ВСКОМП:

- Дифференциация требований на достоверность измерения сигналов с выходов датчиков для различных значений (большая разрешающая способность для начальных значений сигналов, чем для конца диапазона).
- Значительный диапазон обрабатываемых сигналов (до 100 dB).
- Минимальное соотношение "сигнал-шум" на выходе всех этапов первичной обработки.

Примерами могут служить обработка данных в высокоточных гироскопических системах, технической томографии.

Основой выполнения упомянутых требований является обеспечение высокого качества измерения сигналов в реальном времени и оперативность последующей обработки в ФОП. Предметами рассмотрения в настоящей статье являются методы и средства построения измерительно-преобразовательных подсистем ВСКОМП.

Измерительные аналого-цифровые ФОП

В настоящее время все большую актуальность для ВСКОМП приобретают встроенные системы прецизионного измерения малых электрических параметров в реальном времени. Можно выделить следующие основные области применения таких систем:

- Встроенная электроника MEMS.
- Прецизионные сенсоры для регистрации физических и химических процессов с очень малыми значениями энергии.
- Системы мониторинга состояния ответственных частей конструкций летательных аппаратов в процессе их эксплуатации.

Одновременное удовлетворение всем перечисленным требованиям с использованием существующих подходов, основанных на усилении входного сигнала и последующем АЦ-преобразовании, практически невозможно. Поэтому представляется целесообразным использование функциональных возможностей микроэлектроники, которые могут реализовать на аналоговом уровне первичную обработку входного сигнала с тем, чтобы как минимум существенно упростить АЦ-преобразование.

Возможным решением, обеспечивающим сочетание высокой точности измерений и большой диапазон изменения входных сигналов, является создание аналого-цифровых ФОП.

Основой предлагаемых ФОП является нелинейное АЦ-преобразование сигналов, реализуемое с помощью интегрирующего операционного усилителя с большим входным сопротивлением и специализированного процессора, формирующего вид функциональной шкалы (ФОП закона преобразования). На рис. 3 приведена функциональная схема измерительного тракта подсистемы первичной обработки данных в ВСКОМП.

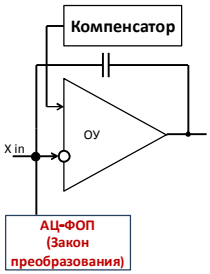


Рис. 3. Нелинейный АЦП со встроенным ФОП

Измерительная шкала, формируемая АЦ-ФОП, может быть представлена в виде

$$N = kX^n, \qquad (1)$$

где N – цифровой код на выходе АЦП; k – масштабный коэффициент, n < 1.

Для варианта описываемой шкалы суммарная относительная погрешность измерения сигналов может быть представлена в виде

$$\delta N = \delta k + n\delta X + n \ln X \delta n \,, \qquad (2)$$

где δN, δk, δx, δn – относительные значения суммарной погрешности измерения, погрешности входной величины, масштабного коэффициента и показателя степени соответственно.

Простейший анализ (2) показывает, что использование в качестве шкалы измерения нелинейной функции при любом n <1 будет обеспечивать меньшее значение относительной суммарной погрешности, чем для линейной шкалы измерения. Кроме того, исследования показывают, что при малых значениях X разрешающая способность нелинейного измерителя будет всегда выше, чем для линейного. Поэтому нелинейное АЦ-преобразование вида (1) предлагается в качестве основы для построения перспективных измерительных аналого-цифровых процессоров.

При измерении электрических параметров в широком диапазоне изменения их значений всегда суммарная относительная погрешность измерения значений в начале диапазона превышает погрешность измерения величин в конце диапазона. Более того, для эффективного подавления аддитивных шумов с помощью обработки уже измеренных параметров специально повышают чувствительность измерителей в области малых значений входных параметров, в то же время для больших значений вполне возможно загрубить измерения. С этой целью для линейной шкалы измерений применяют кусочно-линейную аппроксимацию с помощью разбиения диапазона измерения на поддиапазоны. Но это приводит к появлению выбросов при проведении измерений в реальном времени, когда значения входного параметра изменяются в окрестности границ разбиения. Уже отмечалось [4], что нелинейная шкала вида (1) устраняет необходимость в переключениях. С другой стороны, направленно изменяя параметр n, возможно уже переходить от одной нелинейности к другой, добиваясь оптимума качества измерения также без всяких переключений шкалы. Такое программируемое изменение вида нелинейности превращает обычный измерительный АЦП в аналого-цифровой функционально-ориентированный процессор (АЦ-ФОП), в котором встроенный ФОП реализует адаптивное изменение вида закона измерения. В простейшем случае это достигается с помощью изменения n.

Пример. В случае измерения тока параметр n зависит от вида зависимости I(t), где I – ток разряда измерительного конденсатора. В общем случае изменение тока разряда во времени имеет вид $I(t) = \alpha t^{\beta}$, где α, β – коэффициенты, зависящие от электрофизических параметров схемы либо формируемые по соответствующему закону. Формирование происходит по заранее реализованной программе средствами цифрового ФОП, кото-

рый для обеспечения требований минимума времени измерения может быть реализован на основе однородной вычислительной среды (ОВС).

На рис. 4 приведена функциональная схема N-канального измерительного АЦ-ФОП, предназначенного для прецизионного адаптивного измерения постоянного тока в диапазоне 1 nA, . . ., 1 μA. С помощью встроенного ФОП закона преобразования возможно достижение суммарной относительной погрешности измерения тока в 1 nA не более 0,1 %.

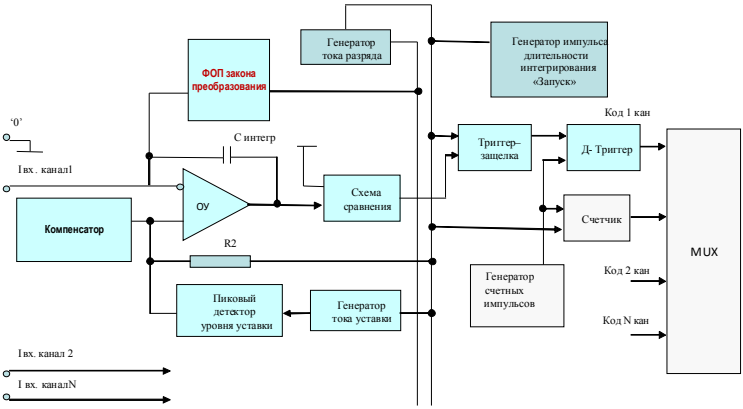


Рис. 4. Функциональная схема N-канального измерительного АЦ-ФОП

В настоящее время завершена разработка нелинейного измерителя токов на основе использования постоянной нелинейности. В частности, используется n = 0,5. Разработка доведена до работающего экспериментального образца 4-канального измерителя. Основные характеристики данного измерителя приведены в таблице.

Таблица

Основные характеристики измерителя токов на основе нелинейного АЦ-преобразования

Параметр	Значение
Входной ток	1 nA, …. , 1μA
Суммарная относительная погрешность измерения	1 nA – не более 1%
	1μA – не более 0.03%
Разрядность выходного кода	Не более 14 бит
Количество независимых каналов измерения	4
Время измерения	не более 10 ms на 4 канала

Заключение

Встроенные суперкомпьютеры архитектурно ориентированы на эффективное решение задач для систем реального времени. Одной из важнейших областей применения ВСКОМП является измерение различных параметров в составе систем реального времени. Требование высокой точности, реализации широкого диапазона измерений приводит к целесообразности создания АЦ-ФОП, архитектура и специфика аппаратурных решений которых обеспечивают оптимальные характеристики по сравнению со стандартными методами и средствами измерения.

1. *Wolf M.* High-Performance Embedded Computing: Architectures, Applications, and Methodologies. Morgan Kaufman. – 2014. – 506 p.
2. *Frank O'Mahony, Nicola da Dalt, Ken Chang and other. F6*: Energy-efficient I/O design for next-generation systems // 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). – P. 520 – 521.
3. High-performance Element Processing Architecture for Open Standards Radar Systems// White paper. December 19, 2012. Curtiss–Wright: https: //www.curtisswrightds.com/ infocenter/ white-papers/ high-performance-element-processing-architecture-for-open-standards-radar-systems.html.
4. *Лукин Н.А., Рубин Л.С.* Применение нелинейного функционального аналого-цифрового преобразования для прецизионных измерений малых электрических величин в реальном времени // Гироскопия и навигация. – 2014. – №4. – С. 131 – 141.

А.К. Мельников

АВТОМАТИЗАЦИЯ ПРОЦЕДУРЫ АНАЛИЗА СООБЩЕНИЙ НА ГИБРИДНЫХ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСАХ

НТЦ ЗАО «ИнформИнвестГрупп», г. Москва,
ak@iigroup.ru

В настоящее время актуальной является задача выделения в глобальном информационном пространстве сообщений, содержащих смысловую нагрузку. Для решения данной задачи на первом этапе необходимо из потока сообщений отсеять сообщения, заведомо не содержащие смысловой нагрузки, основным свойством которых является равновероятность встречаемости всех знаков, из которых они состоят. В работе автора [1] разработан обобщенный статистический метод анализа (ОСМА) сообщений, позволяющий с помощью применения статистических критериев согласия (СКС) с равновероятным распределением отсеивать из потока сообщений сообщения, не содержащие смысловую нагрузку. Основой эффективного применения ОСМА является использование в

применяемом критерии согласия распределения вероятности, соответствующего параметрам оцениваемого сообщения. Под эффективностью применения критерия понимается минимальность ошибки второго рода, т.е. минимальное количество сообщений с равновероятным распределением знаков, принятых за сообщения, с неравновероятным распределением знаков.

Целью данной работы является описание основной процедуры ОСМА в понятиях множеств для автоматизации процессов обработки сообщений на современных гибридных высокопроизводительных вычислительных комплексах с реконфигурируемой архитектурой [2].

Постановка задачи

Рассмотрим сообщение, состоящее из n знаков, которые принимают значения из множества (алфавита) $A_N=\{a_1, \dots, a_N\}$ мощности N, $|A|=N$.

Не ограничивая общности приводимых рассуждений в качестве статистики $S_{n,N}$ критерия, будем рассматривать введенную более 100 лет назад Карлом Пирсоном [3, 4] статистику хи-квадрат – χ_n:

$$\chi_n = \sum_{i=1}^{N} \frac{(h_i - np_i)^2}{np_i},$$

где h_i – частота встречаемости знака (исхода) a_i в первых n испытаниях, n – длина текста (объем выборки), N – число исходов полиномиальной схемы (мощность алфавита A_N) и p_i – вероятность a_i -го исхода.

Для применения статистики $S_{n,N}$ в статистическом критерии согласия по разделению двух простых гипотез о равновероятном и неравновероятном распределении знаков в анализируемом сообщении необходимо знать распределение вероятности значений статистики (распределение вероятности) $S_{n,N}$

$$P\{ S_{n,N} \ge c \}$$

для равновероятной полиномиальной схемы, когда

$$\{ p_i = 1/N \mid i = 1, \dots, N \}.$$

Применение в качестве распределения вероятности статистики точного распределения минимизирует ошибку второго рода. Анализ производительности современных вычислительных средств и вычислительной сложности расчета точных распределений показывает, что для многих значений n и N точные распределения статистик еще долгое время не смогут быть рассчитаны [1].

В работе автора [1] показано, что при различных сочетаниях значений n и N для минимизации ошибки второго рода в качестве $P\{ S_{n,N} \ge c \}$ должны выступать точные $P_T\{ S_{n,N} \ge c \}$, Δ–точные $P_\Delta\{ S_{n,N} \ge c \}$ (отличающихся от точных на заранее заданную величину $\Delta \mid P_\Delta\{ S_n \ge c \} - P_T\{ S_n \ge c \} \mid \le \Delta$) и предельные $P\{ \chi^2 \ge c \}$ с $(N–1)$

степенью свободы распределения. Границы применения различных видов распределения вероятности значений статистики $S_{n,N}$ определяются вычислительной сложностью их расчета – $K(\mathrm{P}\ \{\ S_{n,N} \geq c\ \})$, производительностью доступных во время t вычислительных средств – $p(t)$ и допустимым временем на поведение расчетов T. Значения n и N определяются из соотношения

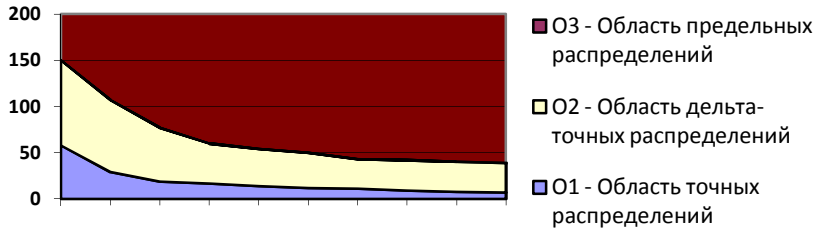
$$K(P\ \{\ S_{n,N} \geq c\ \}) = p(t) \times T,$$

где величину $(p(t) \times T)$ можно определить как доступный во время t вычислительный ресурс (computer power) – $CP(t)$.

В предположении доступного во время t в течение 1 месяца ($T = 2\ 592\ 000$ сек) вычислителя производительностью 10^{15} операций в секунду ($p(t) = 10^{15}$ оп./сек) в [1] были рассчитаны границы областей применимости точных, Δ–точных и предельных видов распределений вероятности значений статистики хи-квадрат – χ_n. Применение рассчитанных распределений вероятности в соответствующих областях при анализе сообщения с параметрами n и N позволяет:

- во-первых, провести анализ сообщения, так как это позволяет имеющийся вычислительный ресурс $CP(t)$,
- во-вторых, гарантированно получить наименьшую ошибку второго рода при применении СКС.

Области применения различных видов распределений статистики хи-квадрат – χ_n показаны на рисунке.



Области применения различных видов распределений статистики хи-квадрат – χ_n

Областью применения определенного вида распределения статистики $\mathrm{P}\ \{\ S_{n,N} \geq c\ \}$ является множество пар натуральных чисел (n,N), для которых данный вид распределения определен, может быть рассчитан в момент времени t с использованием вычислительного ресурса $CP(t)$ и одновременно является оптимальным в смысле минимизации ошибки второго рода СКС. Назовем сформулированное условие условием эф-

фективного существования определенного вида распределения. Отметим, что каждая точка множества пар (n,N) содержит N^n текстов длины n из алфавита A_N мощности N.

Области применения распределений

Область O_1 с верхней границей $G_1(t)$ содержит множество пар значений (n,N), для которых в момент времени t с использованием вычислительного ресурса $CP(t)$ могут быть рассчитаны точные распределения, обозначим это множество через $M_1(n,N)$. Область O_2 с нижней границей $G_1(t)$ и верхней границей $G_2(t)$ содержит множество пар значений (n,N), для которых в указанных условиях не могут быть рассчитаны точные распределения, но могут быть рассчитаны Δ–точные распределения, обозначим его через $M_2(n,N)$. Область O_3 с нижней границей $G_2(t)$ содержит множество пар значений (n,N), для которых не могут быть рассчитаны точные и Δ–точные распределения и поэтому применяются предельные распределения, обозначим его через $M_3(n,N)$. Верхняя граница области O_3 не определена и стремится к бесконечности.

Множество $M_1(n,N)$ состоит из пар натуральных чисел (n,N), удовлетворяющих соотношениям:

$$2 \le N \le 256$$
$$1 \le n \le log_N CP(t).$$

Мощность множества $M_1(n,N)$ можно оценить как

$$\sum_{i=1}^{256} \log_i CP(t).$$

Из условия эффективного существования распределений следует, что точное распределение $P_T \{ S_{n,N} \ge c \}$ существует на множестве $M_1(n,N)$ и не существует на множествах $M_2(n,N)$ и $M_3(n,N)$. Аналогичные рассуждения показывают, что Δ–точное распределение $P_\Delta\{ S_n \ge c \}$ существуют только на множестве $M_2(n,N)$ и не существуют на множествах $M_1(n,N)$ и $M_3(n,N)$, так как на $M_1(n,N)$ точное распределение дает лучший результат чем Δ–точное, а для $M_3(n,N)$ Δ–точное распределение рассчитано быть не может из-за нехватки вычислительного ресурса $CP(t)$. Также показывается, что предельное распределение существует только на множестве $M_3(n,N)$, так как применение его на множествах $M_1(n,N)$ и $M_2(n,N)$ увеличивает ошибку второго рода СКС.

Отдельно необходимо отметить, что границы $G_1(t)$ и $G_2(t)$, разделяющие области (множества), зависят от имеющегося во время t вычислительного ресурса $CP(t)$ и с его ростом увеличиваются, сдвигаются вверх.

Автоматизация процедуры ОСМА сообщений

Для применения ОСМА требуется вычислительная техника с большим вычислительным ресурсом. Такой вычислительной техникой могут быть гибридные высокопроизводительные вычислительные комплексы на базе реконфигурируемых вычислительных систем (РВС) на основе ПЛИС [5].

Как показано в данной работе, множества применения различного вида распределений $M_i(n,N)$ являются множествами множеств, которые с достаточной эффективностью могут быть описаны с использованием языка SET@L, реализующего парадигму ресурсонезависимого программирования [5, 6].

Заключение и выводы

Обобщенный статистический метод анализа сообщений может эффективно применяться для анализа потока сообщений при его реализации на гибридных высокопроизводительных вычислительных комплексах с помощью языка SET@L.

1. *Мельников А.К., Ронжин А.Ф.* Обобщенный статистический метод анализа текстов, основанный на расчете распределений вероятности значений статистик // Информатика и её применения. – М.: Изд-во «ТОРУС ПРЕСС», 2016, в печати, ISSN 1992-2264.
2. *Левин И.И., Дордопуло А.И., Каляев И.А., Дороненко Ю.И., Раскладкин М.К.* Современные высокопроизводительные вычислительные системы с реконфигурируемой архитектурой для решения задач обработки информации и управления // Материалы 8-й Всерос. мультиконф. по проблемам управления (МКПУ-2015): в 3 т. – Ростов-на-Дону. Изд-во ЮФУ, 2015. – Т.3. – С. 113-117. ISBN 978-5-9275-1634-6.
3. *Pearson K.* On the criterion that a given system of deviations from the probale in the case of a correlated system of variables in such that it can be reasonably supposed to have arisen from random sampling. "Phil. Hag.". – 1900, – V. 50. – Pp. 157-170.
4. *Smith P.F., Rae D.S., Manderscheid R.W., Silbergeld S.* Exact and approximate distributions of the chi-squared statistic for equiprobability. "Commun. Statist.", 1979. – B 8(2). – № 1. – Pp. 131-149.
5. *Левин И.И., Мельников А.К.* Методы управления гибридными высокопроизводительными вычислительными комплексами. // Суперкомпьютерные технологии: Материалы 3-й Всерос. науч.-техн. конф.: в 2 т. – Ростов-на-Дону: Изд-во ЮФУ, 2014. – Т.1. – С. 55-60. ISBN 978-9275-1283-6.
6. *Левин И.И., Мельников А.К.* Управление гибридными вычислительными системами на языке SET@L // Materiály XI mezinárodní vědecko-praktická conference. Aktuální vymoženosti vědy – 2015» Díl 7. Moderní informační technologie. Praha. Publishing House «Education and Science», 2015. – 96 c. – C.23-28. – ISSN 978-966-8736-05-6.

И.А. Морозов

ИНТЕГРАЦИЯ IP-ЯДЕР ДЛЯ ПЛИС В РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХСИСТЕМАХ

ФГУП «НИИ «Квант», г. Москва,
cooler@sha.su

Введение

Интеграция IP-ядер в реконфигурируемые вычислительные системы (РВС) на базе программируемых логических интегральных схем (ПЛИС), в зависимости от поставленных задач, может происходить различными путями. Ключевыми параметрами для определения таких путей являются общая структура РВС, характеристики применяемых ПЛИС, организация связей между ними и требования IP-ядер к ресурсам.

В данном докладе рассматриваются РВС, которые строятся по принципу подключения множества ПЛИС к управляющему серверу [1, 2] и размещения в каждой ПЛИС нескольких Вычислителей [3]. При построении таких РВС встает вопрос о том, как должна быть организована связь между Вычислителями и управляющим сервером.

Общие подходы

Типичным подходом является использование стандартизованных интерфейсов, таких как шины групп AMBA AXI [4, 5] и Altera Avalon [6], шина Wishbone [7] и пр. Данные шины подразумевают возможность интеграции множеств ведущих и ведомых устройств с использованием различных топологий подключения. Применяемые шины можно разделить на два класса – адресуемые (Memory-Mapped) и потоковые (Stream). У каждого из этих классов есть свои преимущества и недостатки при использовании в различных ситуациях.

В большинстве таких шин передача данных осуществляется с помощью «рукопожатия», поддерживается возможность передачи различного количества бит (как правило, кратного 8) за один раз, возможна интеграция различного количества регистровых уровней между подключаемыми устройствами.

Интерфейс KV-AXIS

Во ФГУП «НИИ «Квант» применяется собственный комплект разработчика приложений (КРП), состоящий из программной и аппаратной составляющих [3]. Данный комплект предназначен для работы с РВС на базе ПЛИС компании Xilinx, подключенных по шине PCI Express к серверу управления с установленной операционной системой Linux.

Ключевым компонентом КРП является интерфейс KV-AXIS, состоящий из двух разнонаправленных каналов AXI4-Stream с шириной 32 или 64 бита, работающих на частоте до 250 МГц. Подключение Вычислителей производится с помощью специализированных Адаптеров, предоставляющих удобный Вычислителям интерфейс, схема показана на рис. 1. Адаптеры разрабатываются отдельно для каждого типа Вычислителей.

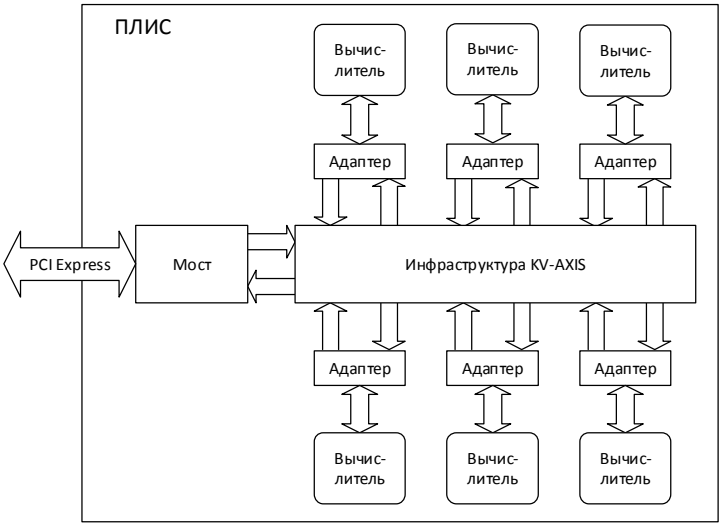


Рис. 1. Схема подключения Вычислителей

Элементы инфраструктуры KV-AXIS позволяют использовать различные топологии, наиболее востребованными являются стандартные топологии типа «Звезда» и «Кольцо», а также различные их комбинации.

Использование топологии типа «Кольцо» упрощает трассировку за счет снижения расхода ресурсов (в том числе линий связи) на сервисные элементы, а применение топологии типа «Звезда» позволяет снизить латентность и упростить трассировку за счет снижения пропускной способности (за счет количества передаваемых за один раз бит или частоты тактового сигнала) к каждому из «лучей» звезды.

Алгоритмы работы и адаптеры

Работа с Вычислителем может строиться по различным алгоритмам, наиболее типичные из которых показаны на рис. 2.

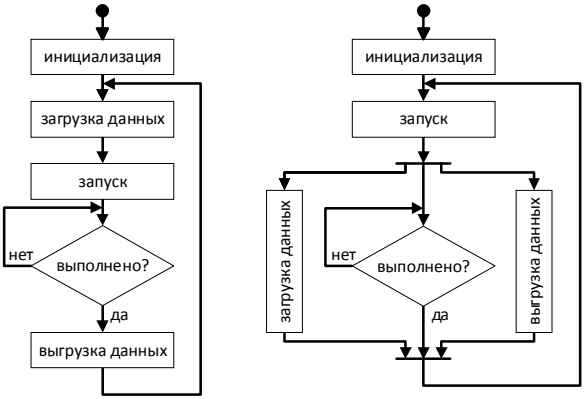


Рис. 2. Алгоритмы работы с Вычислителями

Исходя из анализа данных алгоритмов, можно выделить основные блоки Адаптера, показанные на рис. 3:

- блок управления – производит запуск вычисления, конфигурирование и сброс состояния вычислителя;
- блок статуса – осуществляет контроль состояния вычислителя;
- входной и выходной буфера для потоков данных.

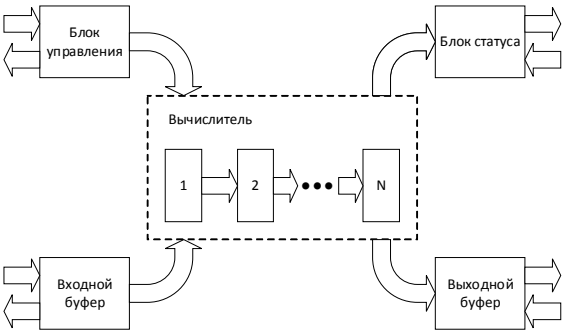


Рис. 3. Подключение модулей Адаптера к Вычислителю

Пакетный режим

Помимо подключения Вычислителей в ПЛИС встает вопрос доступа к ним с управляющего сервера, так как простое предоставление адресуемого доступа к каждому устройству при большом их количестве приводит к неоправданной трате ресурсов процессора на арбитраж.

Одним из эффективных решений является реализация пакетного режима работы. В нем все Вычислители являются независимыми и равнозначными. Работа с Вычислителями разбивается на две фазы – первичной настройки (загрузка инициализационных данных, производимая в режиме непосредственной адресации Вычислителя) и обработки независимых заданий. При этом количество заданий в пакете может превышать количество Вычислителей в ПЛИС, они подгружаются с помощью механизма Scatter-Gather DMA из ОЗУ по мере освобождения Вычислителей. Каждому заданию на входе соответствует набор ответов на выходе.

Данный подход позволяет упростить программный код Вычислителя. Также возможно перераспределение заданий из пакета между несколькими ПЛИС с одинаковой конфигурацией.

Заключение

Для построения эффективной инфраструктуры применяются интерфейсы на базе стандартизованных шин, с выделением стандартных блоков управления, статуса и буферов в Адаптеры с удобными для конкретных Вычислителей интерфейсами.

Могут применяться различные топологии на базе стандартных, каждая из которых имеет свои преимущества и недостатки.

Для снижения нагрузки на управляющий сервер и упрощения программного кода возможно применение пакетного режима с аппаратным арбитражем.

1. *Елизаров Г.С., Горбунов В.С., Малахов И.Н., Титов А.Г.* Компоненты высокопроизводительных реконфигурируемых суперкомпьютеров на основе ПЛИС Xilinx Ultrascale // Суперкомпьютерные технологии: Материалы 3-й Всеросс. науч.-техн. конф.: в 2 т. – Ростов-на-Дону: Изд-во ЮФУ, 2014.
2. *Елизаров Г.С., Горбунов В.С., Титов А.Г.* Элементно-конструктивная база реконфигурируемых компьютеров // НСКФ-2015.
3. *Морозов И.А.* Реализация комплекта разработчика приложений РВС на ПЛИС с поддержкой пакетного режима // НСКФ-2015.
4. AMBA AXI and ACE Protocol Specification / ARM – 2013 – Режим доступа: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022e/index.html
5. AMBA4 AXI4-Stream Protocol Specification / ARM – 2010 – Режим доступа: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0033/index.html
6. Avalon Interface Specifications [Электронный ресурс] / Altera Corporation – 2015 – Режим доступа: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf.
7. Wishbone B4 [Электронный ресурс] / OpenCores – 2010 – Режим доступа: http://cdn.opencores.org/downloads/wbspec_b4.pdf.

И.П. Осинин

ВЫСОКОНАДЕЖНЫЙ МОДУЛЯРНО-ЛОГАРИФМИЧЕСКИЙ ПРОЦЕССОР С РЕКОНФИГУРИРУЕМОЙ АРХИТЕКТУРОЙ

Российский федеральный ядерный центр – Всероссийский научно-исследовательский институт экспериментальной физики, г. Саров, iposinin@vniief.ru

Введение

Крупные задачи, критичные к скорости вычислений, возникают в самых разнообразных областях: в различных сферах промышленности, моделирования климата, в физике, космологии и многих других. На сегодняшний день большинство этих задач решается с использованием кластерных решений.

При этом известны такие традиционные пути повышения быстродействия, как совершенствование техпроцесса, например, создание трехмерных транзисторов, совершенствование алгоритмов выполнения программ, например, распараллеливание вычислений. Однако резерв повышения производительности, до сих пор не используемый при разработке универсальных процессоров, заложен в ускорении выполнения самих вычислительных операций, например, при применении модулярной арифметики на базе системы остаточных классов (СОК). Естественный параллелизм устройств, функционирующих на основе СОК, позволяет распараллелить процесс вычислений как на программном, так и на аппаратном уровне, а модульность и однородность обеспечивает эффективное проектирование структур в сверхбольшом интегральном исполнении (СБИС). Научные работы Червякова Н.И., Акушского И.Я., Akio Sasaki, Garner H., Omondi A. посвящены этому направлению [1].

При этом у СОК имеются и недостатки, среди которых медленное выполнение операций округление и деление, что существенно затрудняет применение плавающей точки. Решение, описанное в данной работе, состоит в применении логарифмической системы счисления (ЛСС), в которой отсутствует операция выравнивания порядков, а деление исходных чисел заменяется вычитанием их логарифмов. Основополагающий вклад внесли ученые Coleman J., Arnold M. [2].

С другой стороны, конвейерным параллелизмом обладают однородные вычислительные среды (ОВС), т.е. среды, аппаратура которых может реконфигурироваться, меняя свои функции, в зависимости от решаемых вычислительной системой задач. Это позволяет эффективно адаптировать архитектуру системы под структуру решаемой задачи, обеспечивая тем самым высокий уровень скорости вычислений. Данной тематике посвящен ряд работ отечественных и зарубежных ученых

Варшавского В.И., Каляева И.А., Князькова В.С., Flynn M., Moore G., MacSorley L. [3]. В общем случае ОВС представляет собой массив вычислительных ячеек структуры, которые объединены регулярными связями. Её применение в вычислительном ядре СБИС-процессора позволило практически пропорционально повышать производительность с увеличением числа ячеек в силу естественного параллелизма их работы.

Объединение в единой процессорной архитектуре рассмотренных форм естественного параллелизма приведено далее.

Организация архитектуры процессора

В разработанном и запатентованном арифметическом устройстве [4] именно эта особенность стала первой плоскостью распараллеливания арифметических операций. В общем случае число каналов равно по количеству остатков СОК (рис. 1).

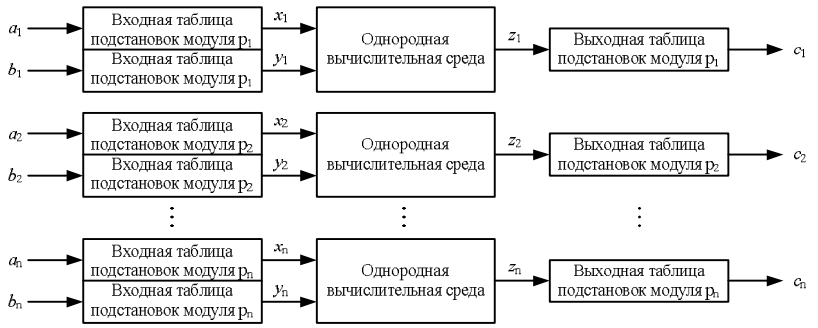


Рис. 1. Организация арифметического устройства

Важным преимуществом СОК является также то, что все основные арифметические операции могут выполняться также быстро как обычное суммирование. Для операций умножение и деление нацело это становится возможным при использовании входных и выходных таблиц подстановок (Look up table – LUT), осуществляющих однотактные преобразования операндов, вычисляя соответственно дискретные логарифмы и антилогарифмы в полях Галуа GF(p).

Так как операция сложения остатков является основополагающей в разработанном арифметическом устройстве, её выполнение распараллелено во второй плоскости с помощью ОВС – квадратной матрицы размерностью m+1, где m – разрядность модуля i-го вычислительного канала. В такой систолической структуре (рис. 2,а) базовые элементы (рис. 2, б) соединены регулярными связями, образуя вычислительный конвейер.

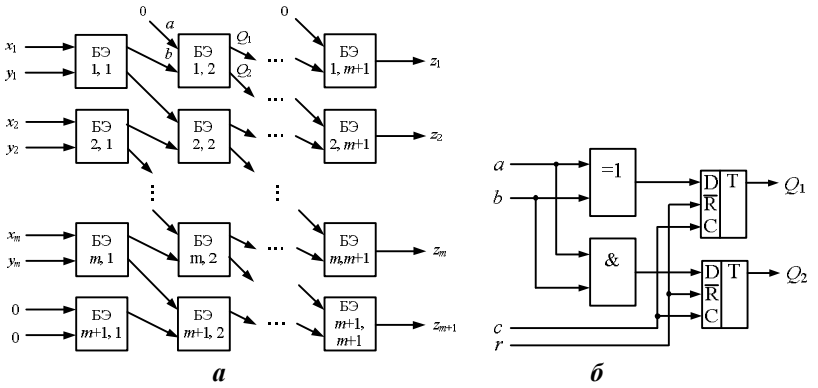


Рис. 2. Организация: а – однородной вычислительной среды; б – базового элемента

Таким образом, первый уровень распараллеливания обеспечен независимым счетом по каждому модулю, второй уровень – конвейерным выполнением операций в ОВС. Ускорение в общем случае достигает k раз для операций сложения и вычитания и k^2 раз для операций умножения и деления нацело по сравнению с последовательными алгоритмами ПСС, где k – разрядность позиционных операндов.

Рассмотренное арифметическое устройство является частью вычислительного ядра (рис. 3), где для снижения задержек доступа к памяти используется иерархическая структура памяти с многократным расслоением доступа по каждому модулю. Помимо трехпортовой кэш-памяти (два порта – чтение, один – запись), имеются буферные очереди операндов и результата. Благодаря линейной структуре FIFO они обладают высоким быстродействием, обеспечивая своевременную подкачку данных в арифметическое устройство.

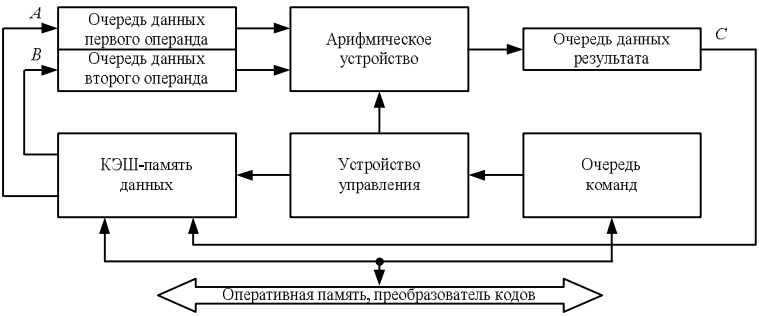


Рис. 3. Организация вычислительного ядра

В разработанном модулярно-логарифмическом MIMD-процессоре таких вычислительных ядер может быть несколько. Ключевой особенностью процессора является динамическая реконфигурация числа ядер за счет уникальности набора оснований каждого вычислительного ядра. Ядра могут работать либо параллельно и независимо, либо объединяться парами для удвоения точности, либо объединяться в четверки для учетверения точности вычислений. Это бывает необходимо для устранения ошибки переполнения разрядной сетки в процессе счета с помощью увеличения диапазона представления чисел.

Надежность вычислений

Надежность является одним из основных свойств любого вычислительного устройства, во многом определяющим его пригодность для использования по назначению. Существуют такие традиционные пути повышения надежности:

- резервирование на уровне компонент системы;
- дублирования вычислений;
- дополнительная обработка исключительных ситуаций и т.п.

Их недостаток состоит в том, что часть оборудования простаивает, т.е. дублирует работу, вплоть до отказа одного из компонентов системы. Данного недостатка можно избежать, внедрив средства повышения надежности на уровне архитектуры.

Сочетание корректирующих свойств системы остаточных классов и динамической реконфигурации ОВС позволили реализовать следующие средства повышения надежности.

Во-первых, это выявление и исправление ошибок за счет расширения вычислительного диапазона путем назначения одного или нескольких вычислительных каналов в качестве контрольных. Доказано [1], что при возникновении ошибки, результат операции окажется в запрещенном контрольном интервале. Причем в процессоре применен механизм, локализующий вычислительный канал, где произошел сбой. В случае подобного сбоя происходит исправление ошибки путем вычисления верного остатка результата в локализованном канале и инкремент счетчика ошибок вычислительного канала.

Во-вторых, при превышении заданного порога в счетчике ошибок, возникает ситуация отказа и канал блокируется. Однако в отличие от повсеместно распространенных универсальных вычислительных устройств, модулярно-логарифмический процессор (МЛП) продолжает работу, сокращается лишь диапазон представления чисел. Если при этом нельзя снижать точность вычислений, то можно использовать вычислительные каналы другого ядра, в том числе и контрольные. Такая реконфигурация позволяет создать процессор, устойчивый к постепенной де-

градации. В пределе, вычисления будут продолжаться, пока хотя бы один вычислительный канал будет работоспособен.

Кроме того, в процессе счета могут возникать неявные ошибки, связанные с округлением чисел в процессе вычислений по стандарту IEEE-754. К тому же работа с плавающей точкой связана с делением чисел, что негативно сказывается на скорости выполнения операций в СОК, так как данная операция является немодульной. Эти причины привели к необходимости применения аналога плавающей точки – логарифмической системы счисления (ЛСС), где в процессе вычислений отсутствуют фазы нормализации и денормализации, что избавляет от ошибок округления и необходимости деления на степень двойки.

Заключение

Подробнее модулярно-логарифмический формат, сравнительный расчет надежности и оценка производительности процессора приведены в [5].

Помимо универсальных процессоров, у МЛП существуют специализированные аналоги. Например, модулярный нейропроцессор, разработанный в Ставропольском военном институте связи [1]. Ключевое отличие МЛП от него состоит в применении ОВС вместо нейросетей, а также наличии блока обработки вещественных чисел на базе логарифмической системы счисления. Другой аналог – «European Logarithmic Microprocessor», разработанный в университете Ньюкасл [3] и функционирующий на базе ЛСС. Ключевое отличие МЛП состоит в применении СОК и ОВС для распараллеливания вычислений на уровне разрядов чисел.

Таким образом, в представленном модулярно-логарифмическом процессоре заложен потенциал параллельного счета и обеспечения отказоустойчивости на уровне архитектуры, что делает актуальным его применение для расчета задач, критичных не только к скорости, но и к надежности вычислений.

1. *Червяков Н.И.* Модулярные параллельные вычислительные структуры нейропроцессорных систем / Н.И. Червяков. – М.: Физматлит, 2003. – 288 с.
2. *Coleman, J.N.* Arithmetic on the European Logarithmic Microprocessor / J.N. Coleman // IEEE Transactions on Computers. – 2000. – Vol. 49. – No. 7. – P. 702 – 715.
3. *Каляев И.А.* Реконфигурируемые мультиконвейерные вычислительные структуры / И.А. Каляев. – Ростов-на-Дону: ЮНЦ РАН, 2008. – 320 с.
4. Ячейка однородной вычислительной среды, однородная вычислительная среда и устройство для конвейерных арифметических вычислений по заданному модулю: патент 2477513 РФ: МПК G06F7/72 / И.П. Осинин, В.С. Князьков; опубл. 10.03.2013г. Бюл. №7.
5. *Осинин И.П.* Высоконадежный модулярно-логарифмический процессор с реконфигурируемой архитектурой // Параллельные вычислительные технологии (ПаВТ'2016): труды Международной научной конференции. – Челябинск: ЮУрГУ, 2016. – С. 642 – 654.

Е.В. Пальчевский, А.Р. Халиков

РЕАЛИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА НА БАЗЕ ПРОЦЕССОРОВ INTEL XEON X5660

Уфимский государственный авиационный технический университет,
г. Уфа,
teelxp@inbox.ru, khalikov.albert.r@gmail.com

В данной статье представлена реализация кластерных технологий для повышения производительности и выдержки нагрузочной способности множественных и массивных вычислительных процессов. Проанализированы практическая и теоретическая мощности создаваемого кластера. Показана полная и детализированная настройка типа «один ко всем», применяемая к управлению всей узловой системой кластера в консольной оболочке на основе операционной системы «Ubuntu».

Введение

В современном мире кластеры играют важную роль, так как вычислительные операции достигают огромных объемов, для выполнения которых требуются большие объемы вычислительных ресурсов [1]. Кластер – это группа физических серверов, имеющая взаимосвязь между собой, за счет высокоскоростных каналов связи и представляют единый вычислительный ресурс [2].

Различают несколько видов кластеров [3]:

1) кластеры высокой доступности: большая мощность физических ресурсов, обеспечивающая повышенную отказоустойчивость. Деление данного вида происходит по двум принципам:
   - у кластера имеются активные и пассивные узлы связи;
   - созданный кластер с модульной избыточностью: недопустим простой вычислительных ресурсов;
2) кластеры с распределенными системами вычислений: происходит параллелизация вычислительных процессов. Основные недостатки:
   - низкая доступность всех кластерных узлов;
   - обязательным условием полноценной работы является полная изоляция процесса от параллельно-выполняемых;
3) балансировочные: распределенная нагрузка на ЭВМ. Основное достоинство: экономия физических ресурсов;
4) вычислительные кластеры: используются в научных исследованиях.

Реализация кластеров часто используется на предприятиях. Огромная доля кластерных возможностей распределяется между шестью странами.

Принцип работы кластера представлен на рис.1.

Целью работы является реализация вычислительного кластера, нагрузочная способность которого приравнивается к 18000 *Mflops*.

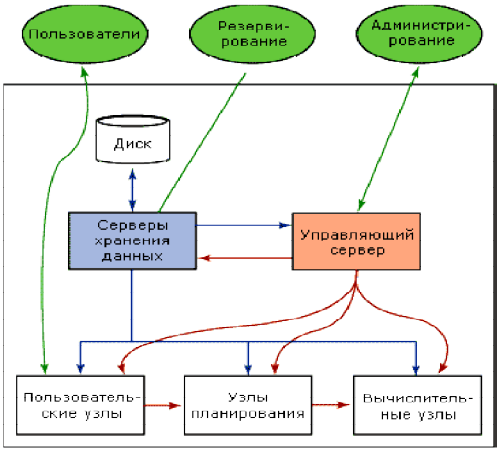


Рис.1. Схема кластерной системы [4]

Для достижения цели используется технология вычислительного кластера со следующими характеристиками:

- количество физических серверов – 30;
- процессор *Intel Xeon* 5660 (*CPU* – 60, физических ядер – 360, количество потоков – 720);
- оперативная память – 960*GB*;
- винчестеры – *RAID* 10 (*Intel S*3710 *SSDSC*2*BA*012*T*401 800*GB* каждый);
- внешний сетевой канал – 20*GB*/*s*.

Реализация кластера с управляющим консольным сервером на операционной системе «*Ubuntu* 15.10»

Реализуемый кластер увеличивает суммарную мощность, что дает преимущество в производительности.

Конфигурирование группы серверов в единое целое производится за счет следующих составляющих:

- этап подготовки;
- сетевая настройка;
- настройка безопасности.

Подготовительный этап представляет собой реализацию управляющего сервера в консольном режиме:

- реализация метода проксирования всех физических серверов, исключая головной: в консоли необходимо выполнить команду «*sudo apt-get install apt-cacher-ng*». Данная команда позволяет установить пакет кеширующего прокси на основной сервер (управляющий);

- с целью настройки прокси по всему кластеру – нужно осуществить ввод следующей строки: «*sh -c "echo \"Acquire::http { Proxy \\\"http://*95.47.161.115:3142*\\\"; }; \"> /etc/apt/apt.conf.d/*01*proxy*"», где *IP*-адрес – это консоль кластера во внутренней сети, а 3142 – порт кеширующего прокси;
- инсталлирование «*SSH*» для удаленного управления кластером: «*sudo apt-get install openssh-server nfs-kernel-server*»;
- загрузка необходимых библиотек, с целью совместимости запускаемого программного обеспечения: «*sudo apt-get install mc build-essential fort77 gfortran libstdc++5 libltdl7-dev*»;
- создание стандарта интерфейса передачи информации: «*sudo apt-get install openmpi-bin openmpi-doc*».

Настройка сетевых компонентов подразумевает собой организацию общего каталога для всех узлов создаваемого кластера, с целью хранения каких-либо данных. Сетевой ресурс создается вводом следующей команды, в консоли: «*sudo sh -c "echo \"$HOME/mpi* 95.47.161.115/24*(rw,insecure,nohide,all_squash,anonuid=*1000*,anongid=*10 00*,no_subtree_check)\" >> /etc/exports*"», где 95.47.161.115/24 – подсеть адресов, расположенных внутри системы. После применения вышенаписанной команды – необходимо перезагрузить сервис «*NFS*»: «*/etc/init.d/nfs-kernel-server restart*».

Для настройки безопасности необходимо сделать вход с единичного идентифицируемого ключа, реализующего возможность удаленного управления из под *SSH*-доступа: «*cat id_rsa.pub >> authorized_keys*2» (копирование созданного публичного ключа).

В конечном итоге был реализован кластер из тридцати физических серверов и одним основным, который создан для управления созданной системой.

Тестирование вычислительной мощности и сравнивание полученных результатов

Результаты, полученные в ходе тестирования вычислительной мощности созданного кластера, представлены в таблице.

Созданный кластер имеет повышенную производительность, относительно единичного сервера, и, следовательно, выдерживает большую нагрузочную способность, с горячей заменой каких-либо комплектующих/физических серверов.

В таблице произведен замер полной кластерной производительности по параметру «*Linpack*», который приравнивается к 12913 *MFlops*. Пиковая (теоретическая) мощность вышеприведенной системы: 15345 *MFlops*. Средняя выдаваемая мощность каждой ЭВМ: 430,4 *MFlops*. Максимальное количество серверов в создаваемом кластере: 32. Параметры каждого физического сервера:

- CPU 2 x Intel Xeon 5660 (в сумме 12 ядер / 24 потока по 2.8GHz, 12Mb Cache, 6.40 GT/s);
- RAM 32Gb DDR3-10600 ECC REG;
- SSD 4 x Intel S3710 SSDSC2BA012T401.

Таблица

Тестирование мощности вычислительных процессов, на основе процессоров «2 x Intel Xeon 5660 и винчестерах типа «SSD»

№ сервера	Выдаваемая мощность, *MFlops*	№ сервера	Выдаваемая мощность, *MFlops*
1	450	16	6956
2	880	17	7406
3	1300	18	7856
4	1710	19	8308
5	2100	20	8708
6	2520	21	9123
7	2980	22	9508,12
8	3380	23	9921,01
9	3870	24	10231,27
10	4295	25	10721,27
11	4760	26	11120
12	5190	27	11533
13	5585	28	12013
14	6071	29	12423
15	6462,5	30	12913

Заключение

В данной статье рассмотрены перспективные аспекты использования высокопроизводительной системы и статистика операций в секунду (см. таблицу). Описана конфигурация каждого физического сервера. В качестве полученного результата реализован вычислительный кластер, состоящий из 30 физических серверов и одного управляющего. Проанализированы все мощности и поочередно выведены в отдельную таблицу. Подсчитана средняя, «*Linpack*» и пиковые нагрузки, измеряемые в «*MFlops*».

1. *Колисниченко Д.* Linux. От новичка к профессионалу. – 2-е изд. / Д. Колисниченко. – 2010. – 764 с.
2. *Hein R.* Linux Administration Handbook/ R. Hein. – Изд-во «Вильямс», 2007. – 1071 с.
3. *Орлофф Д.* Ubuntu – бесплатная альтернатива Windows / Д. Орлофф. – Изд-во «Эксмо» 2009. – 352 с.
4. *Бурцев В.С.* Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ / В.С. Бурцев. – Изд-во «Москва», 2006. – 150 с.

А.С. Симонов, И.А. Жабин, Е.Р. Куштанов,
А.Е. Леонова, Д.В. Макагон, А.С. Семенов,
Е.Л. Сыромятников, А.Н. Щербак

ОПЫТ ПРИМЕНЕНИЯ СЕТИ «АНГАРА» ПРИ РЕШЕНИИ ПРИКЛАДНЫХ ЗАДАЧ

АО «НИЦЭВТ», г. Москва,
simonov@nicevt.ru

Сеть «Ангара»

Высокоскоростная сеть «Ангара» [1-3] – полностью российская разработка, сопоставимая по своей функциональности, производительности и надежности с современными разработками мировых лидеров в данной области. СБИС маршрутизатора коммуникационной сети ЕС8430, выпущенная на фабрике TSMC с использованием технологии 65 нм, стала итогом семилетней работы подразделения АО «НИЦЭВТ» – разработчика сети «Ангара».

Сеть «Ангара» поддерживает топологии сети «многомерный тор» (возможны варианты от 1D- до 4D-тор), режим прямого доступа к памяти удаленных узлов RDMA, технологию прямого доступа к памяти графических ускорителей GPUDirect, все стандартные средства программирования (библиотека MPI, технология OpenMP, библиотека SHMEM, стек протоколов TCP/IP), а также перспективные языки программирования Charm++, UPC. Реализована поддержка односторонних коммуникаций и коллективных операций. Коммуникационная сеть «Ангара» обеспечивает совместимость с процессорами x86, GPU, FPGA, а также с отечественными процессорами серии «Эльбрус». Сеть «Ангара» отличается высокой пропускной способностью линков и низкими задержками передачи, которые соответствуют мировому уровню.

Области применения сети «Ангара»

Области применения сетевого оборудования «Ангара» охватывают следующие направления:

1. *Высокопроизводительные вычислительные системы всех классов производительности (от десятков до тысяч узлов).* Данные системы необходимы для выполнения сложных научных, инженерных и промышленных расчетов.
2. *Встраиваемые вычислительные системы.* В бортовом применении и специализированных системах при помощи сети «Ангара» возможно объединение вычислительных элементов с высокой скоростью и низкими задержками, а также под-

держанной аппаратно возможностью обеспечить необходимый уровень отказоустойчивости.

3. *Центры обработки данных.* Сетевое оборудование «Ангара» необходимо как инфраструктурная составляющая, которая позволяет объединять с высокой скоростью различные вычислительные ресурсы.
4. *Программно-аппаратные комплексы.* Сетевое оборудование «Ангара» необходимо для достижения высокой производительности в создаваемых различных аналитических системах (семантический анализ, обработка графов, машинное обучение), системах обработки Больших Данных (Hadoop, Spark и др.), а также в инженерных вычислительных комплексах.

Результаты применения сети «Ангара»

В АО «НИЦЭВТ» собран кластер «Ангара-К1», состоящий из 36 узлов, объединенных сетью «Ангара». Кластер состоит из двух типов узлов с процессорами Intel Xeon E5-2630 и Intel Xeon E5-2660. Общее количество ядер – 384. Память каждого узла – 64 ГБ. Узлы объединены сетью «Ангара» с топологией 3D-тор 3x3x4. В испытаниях использовалась библиотека MPI (MPICH 3.0.4).

Сравнительное оценочное тестирование проводилось также на 36 узлах суперкомпьютера МВС-10П, установленном в МСЦ РАН и включающем более производительные процессоры Intel Xeon E5-2690, объединенные сетью Mellanox Infiniband 4xFDR. Сравнение результатов тестирования на синтетических коммуникационных тестах, тестах NAS Parallel Benchmarks и модели прогноза погоды ПЛАВ показало [4], что сеть «Ангара» не уступает по производительности сети Mellanox Infiniband 4xFDR, а в ряде случаев превосходит ее.

Высокопроизводительное сетевое оборудование протестировано сотрудниками РФЯЦ ВНИИЭФ, ИПМ им. М.В. Келдыша РАН, АО «Т-Платформы», ИВМ РАН, Гидрометцентра России, ОИВТ РАН, ИДСТУ СО РАН, АО «МЦСТ» совместно с процессорами архитектуры Эльбрус. Все проведенные исследования подтвердили работоспособность, производительность и надежность сетевого оборудования «Ангара».

Кластер «Ангара-К1» функционирует в открытом доступе для научных групп и потенциальных заказчиков с 01.11.2015. Число внешних пользователей приблизилось к 20, кластер функционирует в круглосуточном режиме.

На «Ангара-К1» проводятся расчеты по решению задачи моделирования течений газов в микроканалах технических систем. Данная задача решается при помощи мультимасштабного подхода, сочетающего решение уравнений квазигазодинамики и коррекцию газодинамических

параметров методом молекулярной динамики. Параллельная реализация алгоритма [5] разработана в ИПМ им. М.В. Келдыша РАН. С ноября 2015 г. по май 2016 г. на решение задачи моделирования течения газов потрачено более 2900 процессорочасов.

Второе поколение сети «Ангара»

АО «НИЦЭВТ» продолжает развитие отечественной коммуникационной сети «Ангара» и ведет разработку сети «Ангара» второго поколения, выпуск которой предполагается начать в 2017 г. Качественными улучшениями сети «Ангара» второго поколения будут являться более гибкая топология «модифицированный 4D-тор», повышенная пропускная способность линка и более низкая задержка передачи сообщений, возможность подключения к каждой СБИС нескольких процессоров или ускорителей, а также расширение функциональных возможностей.

Создание второго поколения коммуникационной сети «Ангара» создает условия, при которых пользователям при смене поколений оборудования не придется переучиваться и привыкать к новой технологии, а оптимизированные под сеть «Ангара» программы будут также эффективно работать при использовании сети «Ангара-2».

1. *Симонов А.С., Жабин И.А., Макагон Д.В.* Разработка межузловой коммуникационной сети с топологией «многомерный тор» и поддержкой глобально адресуемой памяти для перспективных отечественных суперкомпьютеров // Научно-техническая конференция «Перспективные направления развития вычислительной техники». – М.: Изд-во ОАО «НИЦЭВТ», 2011.
2. *Симонов А.С., Макагон Д.В., Жабин И.А., Щербак А.Н., Сыромятников Е.Л., Поляков Д.А.* Первое поколение высокоскоростной коммуникационной сети «Ангара» // Наукоемкие технологии. – 2014. – Т.15. – №1. – С. 21 – 28.
3. *Жабин И.А., Макагон Д.В., Симонов А.С.* Кристалл для Ангары // Суперкомпьютеры. – 2013. – С. 46 – 49.
4. *Агарков А.А., Исмагилов Т.Ф., Макагон Д.В., Семенов А.С., Симонов А.С.* Предварительные результаты оценочного тестирования отечественной высокоскоростной коммуникационной сети «Ангара» // Параллельные вычислительные технологии (ПаВТ'2016): труды международной научной конференции (28 марта – 1 апреля 2016 г., г. Архангельск). – Челябинск: Издательский центр ЮУрГУ, 2016. – С. 42 – 53.
5. *Подрыга В.О., Поляков С.В., Пузырьков Д.В.* Суперкомпьютерное молекулярное моделирование термодинамического равновесия в микросистемах газ-металл // Вычислительные методы и программирование: новые вычислительные технологии. – 2015. – Т.16. – №1. – С. 123 – 138.

А.А. Соколов, П.В. Павлухин, А.В. Будник

ПОДХОД К ОРГАНИЗАЦИИ УПРАВЛЕНИЯ СУПЕРКОМПЬЮТЕРОМ НА БАЗЕ СИСТЕМЫ ВИРТУАЛЬНЫХ МАШИН

ФГУП «НИИ «Квант», г. Москва,
sokolov@rdi-kvant.ru, pavlukhin@rdi-kvant.ru, budnik@rdi-kvant.ru

Традиционно при организации управления суперкомпьютером все сервисы управления реализуются на выделенном именно для этих сервисов оборудовании. При этом организация отказоустойчивости и доступности решается для каждого сервиса по отдельности. Совмещение нескольких сервисов на одном и том же оборудовании иногда позволяет оптимизировать инфраструктуру, но зачастую оно трудновыполнимо.

Применение технологий виртуализации для организации управления суперкомпьютером позволит абстрагироваться от аппаратной конфигурации, быстро заменять неработоспособные физические узлы, существенно снижая время неработоспособности ресурса или недоступности сервиса, и в итоге оптимизировать инфраструктуру.

Для организации управления суперкомпьютером на базе виртуальных машин (ВМ) необходимо подготовить пул сервисных ВМ, в которых реализуются функции управления. Для организации отказоустойчивого функционирования этих сервисов также необходима система управления ВМ, которая следит за состоянием оборудования, ВМ и их сервисов, и в случае обнаружения сбоев автоматически восстанавливает доступность сервисов управления.

Анализ наиболее распространённых открытых систем виртуализации показал, что они не могут быть напрямую (без модификации) применены при организации управления суперкомпьютером. Причинами являются либо отсутствие поддержки нестандартного оборудования и нестандартных конфигураций программного обеспечения суперкомпьютеров, либо недостаточность функционала по настройке индивидуальных проверок доступности сервисов, выполняющихся в ВМ, либо невозможность реализации многоступенчатой схемы гарантированного выключения сервера или ВМ. В тоже время для организации управления суперкомпьютером функционал распространённых систем виртуализации зачастую избыточен, а требуемый минимум может быть достаточно быстро реализован на базе пакетов открытого программного обеспечения.

В докладе представлен подход к организации управления суперкомпьютером с использованием разработанной в ФГУП «НИИ «Квант» отказоустойчивой системы управления виртуальными машинами (СУВМ). Требования к оборудованию со стороны СУВМ минимальны,

и сама система разработана с использованием свободного открытого ПО под ОС Linux. В качестве гипервизора в СУВМ в текущей версии используется Qemu-KVM [1] с библиотекой управления libvirt [2]. СУВМ устанавливается на сервера управления, объединенные коммуникационной сетью и имеющие общую кластерную файловую систему. Основной компонент системы – менеджер ресурсов SLURM [3]. Каждой ВМ соответствует задание в планировщике SLURM, который запускает задание с очередной ВМ на наименее загруженном (по числу процессорных ядер) сервере, обеспечивая тем самым балансировку нагрузки. Отказоустойчивость СУВМ реализована с помощью Red Hat Cluster (RHC) [4], работоспособность которого сохраняется вплоть до момента, пока исправно работает хотя бы один сервер управления.

СУВМ предоставляет следующие функциональные возможности:

- организация отказоустойчивого функционирования пула сервисных ВМ с возможностью динамического добавления и удаления отдельных ВМ из пула;
- ручной и автоматический запуск, останов и перезапуск отдельной или всего пула ВМ с учётом требований к ресурсам и соблюдением зависимостей между ними;
- контроль доступности сервисов ВМ, состояния самих ВМ и физических серверов, предназначенных для работы ВМ;
- выведения неработоспособных и добавление новых серверов для запуска ВМ;
- обнаружение программных сбоев и отказов оборудования, и восстановление пула ВМ и их сервисов на работоспособных серверах;
- отказоустойчивость СУВМ (работает, пока есть хоть один работоспособный сервер);
- многоступенчатая схема гарантированного выключения сбойных узлов для избегания коллизий при одновременном исполнении нескольких копий одной ВМ;
- обнаружение и ограничение числа непрерывных перезапусков неработоспособных ВМ;
- запись логов обо всех изменениях и ошибках ВМ и физических серверов.

Применение СУВМ при организации управления суперкомпьютерами позволит повысить отказоустойчивость и в итоге оптимизировать инфраструктуру системы управления.

1. http://wiki.qemu.org/KVM
2. http://libvirt.org
3. http://www.schedmd.com
4. http://www.redhat.com

Д.А. Сорокин, А.Ю. Матросов,
Е.Е. Семерникова, К.Н. Алексеев

СТРУКТУРНО-ПРОЦЕДУРНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА SRMP НА ПЛИС*

ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог,
alexseev91@mail.ru

В настоящее время в геолого-геофизических исследованиях по обнаружению в акваториях месторождений углеводородного сырья одной из актуальных проблем при обработке сейсмограмм является необходимость ослабления кратных волн-помех, наиболее интенсивные из которых связаны с переотражением вглубь среды от свободной поверхности, такой как граница воздух-вода. Для исключения помех из анализа обычно применяют двухшаговые методы: на первом шаге производится прогнозирование (моделирование) поля помех, а на втором – его адаптивное вычитание из исходных данных.

Зачастую для прогнозирования кратных волн, если мощность водного слоя достигает 100 м и более, применяется метод SRMP [1]. Он предполагает расчёт модели кратных волн без привлечения информации о глубинно-скоростном строении среды, который осуществляется посредством перебора всех возможных точек отражения в пределах некоторого интервала и суммирования результатов взаимных свёрток.

Алгоритм SRMP относится к классу вычислительно трудоемких сильносвязанных задач, в которых число информационных обменов сравнимо или превышает число выполняемых операций. В этой связи использование традиционных многопроцессорных вычислительных систем неэффективно, поскольку с увеличением числа процессоров, участвующих в решении задачи, значительно увеличится время информационного обмена между ними. В таких случаях перспективным является применение реконфигурируемых вычислительных систем (РВС), основным вычислительным элементом которых являются программируемые логические интегральные схемы (ПЛИС). Эти системы адаптируются под вычислительный граф решаемой задачи и обеспечивают почти линейный рост производительности при увеличении доступного аппаратного ресурса [2].

Алгоритм SRMP состоит из последовательного выполнения прямого БПФ над исходными данными, комплексного умножения с под-

* Работа выполнена при финансовой поддержке из бюджета Союзного государства в рамках реализации государственного контракта Министерства образования и науки Российской Федерации от 17 июня 2015г. № 14.964.11.0001.

суммированием результата и обратного БПФ. Наиболее эффективная для РВС структурная организация вычислений предполагает непосредственное отображение базового подграфа задачи на вычислительное поле системы. Однако для такой реализации алгоритма SRMP требуется вычислительный ресурс, которым не обладают даже современные РВС, поэтому необходимо перейти к структурно-процедурной организации вычислений с использованием конвейеризации.

В результате перехода к структурно-процедурной реализации задачи SRMP была выполнена редукция базового подграфа, на основе чего был разработан конвейер вычислительной структуры алгоритма. Укрупненная структурная схема конвейера представлена на рис. 1.

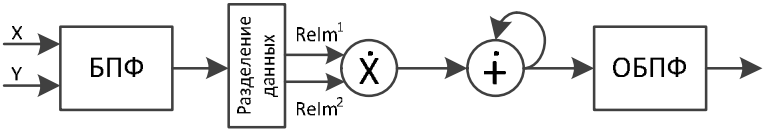


Рис. 1. Укрупнённая структурная схема конвейера алгоритма SRMP

Массивы входных 64-разрядных данных X и Y непрерывным потоком поступают на вход блока БПФ, который состоит из двенадцати базовых операций БПФ. Пройдя все двенадцать ступеней, данные разделяются на два потока и, пройдя через блок комплексного умножения, поступают на вход блока комплексного сложения с накоплением. С выхода блока комплексного сложения с накоплением данные выдаются со скважностью, определяемой размером входных данных, поэтому выполнение последнего этапа задачи SRMP, обратного БПФ, может быть реализовано процедурно с использованием одной ступени БПФ вместо двенадцати, что приводит к сокращению используемого вычислительного ресурса.

Согласно алгоритму задачи SRMP входной массив данных представляет собой совокупность $n \times m$ столбцов, каждый из которых состоит из s элементов. Столбцы объединяются в слои таким образом, что m-й слой – это совокупность столбцов от 1 до n при фиксированном значении m, а n-й слой – совокупность столбцов от 1 до m при фиксированном значении n. Структура массива данных задачи в общем виде представлена на рис. 2.

Формирование одного столбца результирующих данных происходит в результате поэлементной обработки m-го и n-го слоев конвейером SRMP. Следовательно, чтобы получить выходные значения всех m столбцов при фиксированном значении n с помощью конвейера SRMP, необходимо обработать все m слоёв с фиксированным слоем n.

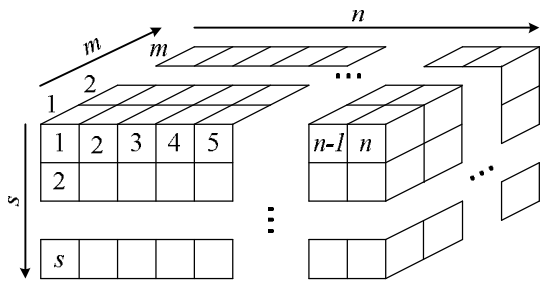


Рис. 2. Структура массива данных задачи SRMP

Получение результата зависит только от обработки входного массива данных. Элементы выходного массива не связаны друг с другом, поэтому не важно, в какой последовательности обрабатываются слои входных данных. Таким образом, при обеспечении параллельного доступа к входным данным некоторого количества слоёв k по направлению m можно параллельно формировать k столбцов результирующих данных.

На основе проведенного анализа структуры массивов входных и выходных данных была разработана структура организации вычислений и потоков данных при реализации алгоритма SRMP, представленная на рис. 3.

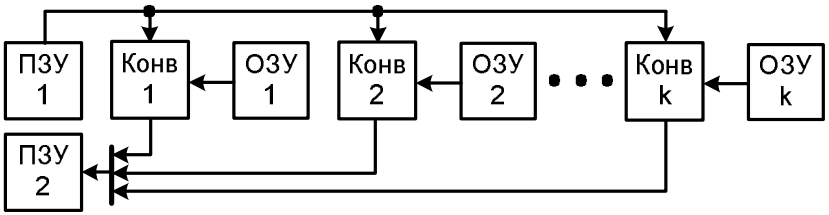


Рис. 3. Структура организации вычислений и потоков данных при структурно-процедурной реализации алгоритма SRMP

В ПЗУ 1 содержится весь массив входных данных, отсортированный по слоям направления n. Все памяти ОЗУ (ОЗУ 1 – ОЗУ k) загружаются различными слоями данных входного массива, отсортированных по направлению m, причем в каждой памяти ОЗУ может содержаться p слоев. Вычисления организованы таким образом, что

из ПЗУ 1 параллельно во все конвейеры (Конв 1 – Конв k) поступает первый слой направления *n,* а из источников ОЗУ – первые слои направления *m*. После обработки слоя направления *n* из ПЗУ 1 начинают поступать данные второго слоя, которые обрабатываются с данными первых слоев из источников ОЗУ. Результатом обработки всех слоев направления *n* с первыми слоями ОЗУ направления *m* будут являться *k* слоев направления *m*, записанных в ПЗУ 2. После этого необходимо повторить процедуру обработки с оставшимися в источниках ОЗУ слоями направления *m*. Когда будет обработано *p* слоев направления *m*, во все ОЗУ необходимо загрузить новые слои, после чего продолжить обработку по вышеописанному алгоритму. Такой способ реализации позволяет обрабатывать входные данные в темпе их поступления.

Таким образом, время, затрачиваемое на решение задачи, будет складываться из суммарного времени загрузки памятей ОЗУ и суммарного времени обработки слоев. Однако суммарное время загрузки памятей ОЗУ много меньше, чем суммарное время обработки слоев, так как на загрузку одного слоя приходится *n* его чтений.

Данную структуру можно реализовать на РВС, в которой вычислительные конвейеры будут реализованы на ПЛИС. При этом элементная база для реализации ПЗУ и ОЗУ должна отвечать требованиям к пропускной способности. При тактовой частоте схемотехнической реализации вычислительных конвейеров, равной 500 МГц, и разрядности шины входных данных 8 байт пропускная способность каждого входного канала данных конвейера составляет 4 Гбайт/с.

Так как в одной ПЛИС может размещаться множество конвейеров, можно объединить источники распределённой оперативной памяти ОЗУ 1 – ОЗУ k в один источник для каждой ПЛИС. Следовательно, при наличии *k* конвейеров в одной ПЛИС память ОЗУ должна обладать пропускной способностью не менее $k \times 4$ Гбайт/с.

К ПЗУ и ОЗУ предъявляются дополнительные требования к объему памяти. Оба ПЗУ должны иметь возможность содержать весь массив обрабатываемых данных, а ОЗУ должно быть как можно больше, потому что чем больше слоев можно разместить в ОЗУ, тем меньше будут суммарные временные затраты на его загрузку.

Для реализации ПЗУ данным требованиям отвечают твердотельные накопители данных SSD, а для реализации ОЗУ – микросхемы памяти HMC (Hybrid Memory Cube).

Предлагаемая организация вычислений хорошо масштабируема – при увеличении доступного вычислительного ресурса можно повышать число реализуемых в ПЛИС конвейеров.

Пренебрегая суммарным временем загрузки памятей ОЗУ и латентностью конвейеров, теоретическое время решения задачи с помощью представленной структуры можно рассчитать по формуле

$$T_{теор} = \frac{n \times m^2 \times s}{k \times N \times \nu},$$

где k – количество конвейеров в одной ПЛИС; N – количество ПЛИС; ν – частота работы; m, n, s – измерения массива данных.

Выполнена экспериментальная оценка времени работы программной реализации алгоритма SRMP. Программа на языке С осуществляла обработку массива данных m×n×s объемом 1024×1024×512 на персональном компьютере с процессором Intel Xeon X3460 2,8ГГц. Время работы программы составило 7585,32 с.

Анализ показал, что в ПЛИС фирмы Xilinx Kintex UltraScale+ KU11P-1 можно разместить k =11 вычислительных конвейеров, работающих на частоте ν=500 МГц. При условии, что на плате будет установлено N=8 ПЛИС, примерное время обработки массива данных m×n×s объемом 1024×1024×512 составит 12,5 с, что обеспечивает ускорение более чем в 600 раз. Исследования показали, что при увеличении объема обрабатываемых данных время решения задачи на РВС по сравнению с временем решения задачи на традиционных процессорах будет расти медленнее, что обусловлено архитектурными различиями вычислительных систем.

Таким образом, предложена структурно-процедурная реализация алгоритма SRMP для РВС, обеспечивающая масштабирование решения при росте доступного аппаратного ресурса и, как следствие, практически линейный рост производительности.

1. *Денисов М.С.* О подавлении кратных волн при обработке результатов морской площадной сейсморазведки. Часть 1: Прогнозирование. – М.: Изд-во ГЕОПРАЙМ, 2008.
2. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – 2-е изд., перераб. и доп. / под общ. ред. И.А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.

С.А. Степаненко

ДЕКОМПОЗИЦИЯ СТРУКТУРЫ ГИБРИДНЫХ ПРОЦЕССОРНЫХ ЭЛЕМЕНТОВ И БАЛАНСИРОВКА ВЫЧИСЛЕНИЙ

Российский федеральный ядерный центр - Всероссийский НИИ экспериментальной физики, Институт теоретической и математической физики, г. Саров,
SAStepanenko@vniief.ru

В [1] показана необходимость применения гибридных процессорных элементов для создания эксафлопсных суперЭВМ. В составе гибридного процессорного элемента задействованы многоядерные универсальные процессоры, реализующие MIMD-дисциплину вычислений, и арифметические ускорители (сопроцессоры), реализующие SIMD-дисциплину вычислений.

В [2, 3] получены аналитические оценки длительностей вычислений, достигаемых гибридными процессорными элементами, и коэффициентов ускорения вычислений по сравнению с ядром универсального процессора.

В основе применяемого в [2, 3] метода – два параметра, характеризующих выполняемый процесс: φ – доля MIMD-фрагмента в исполняемом процессе, и ρ – значение ускорения вычислений на SIMD-фрагменте, достигаемое применением ускорителя; предполагается, что MIMD- и SIMD-фрагменты выполняются последовательно.

Полученные в [2, 3] оценки позволяют оценить длительности выполнения заданного процесса в результате задействования того или иного количества ядер и ускорителей. Кроме того, в указанных работах получены предельные значения ускорения, достигаемые при сколь угодно больших количествах ядер и ускорителей.

В этой работе исследуются применения полученных зависимостей и оценок в условиях практических ограничений – количество ядер универсального процессора и количество ускорителей в составе процессорного элемента фиксированы и конструктивно неизменны.

Для достижения максимального ускорения вычислений предлагается метод декомпозиции структуры процессорного элемента, предусматривающий выделение однородной подструктуры и гибридной подструктуры. Однородная подструктура состоит либо из ядер универсального процессора, либо из ускорителей. Гибридная подструктура содержит и универсальные процессоры и ускорители.

Составы однородной и гибридной подструктур определяются:

- первичными параметрами φ и ρ выполняемого процесса;
- количеством ядер и количеством ускорителей в процессорном элементе;

– производительностью ядра и производительностью ускорителя; значения производительности выражаются через длительности исполняемого процесса.

Предлагаемый метод декомпозиции структуры процессорного элемента и соответствующего этой декомпозиции размещения вычислительного процесса позволяет достигнуть наименьшей в заданных условиях длительности выполнения этого процесса, т.е. достигнуть наибольшей производительности.

В отличие от известных методов динамической балансировки – см., например, [4], – применяющих варьирование потока заданий (процессов), предлагаемый в этой работе метод предназначен для выполнения заданного процесса заданным элементом за минимально возможный временной интервал.

1. *Степаненко С.А., Южаков В.В.* Эксафлопные суперЭВМ. Контуры архитектуры // Программные системы: теория и приложения: электрон. научн. журн. – 2013. – Т.4. – № 3(17). Режим доступа: http://psta.psiras.ru/read/
2. *Степаненко С.А.* Оценки ускорения вычислений гибридными системами // Параллельные вычисления и задачи управления PACO 2010: пленарные доклады Пятой международной конференции. 26-28 октября 2010 г. – М.: ИПУ им. В.А. Трапезникова РАН. – С.61 – 71, ISBN 978-5-91450-062-4.
3. *Степаненко С.А.* Оценки ускорения вычислений гибридными реконфигурируемыми системами // Известия ЮФУ. Технические науки. – 2014.– № 12. – С.74 – 83. ISSN 1999-9429.
4. *Mohan R., Gopalan N.P.* Dynamic Load Balancing using Graphics Processors. // L.J. Intelligent Systems and Applications. – 2014. – № 5. – P. 70 – 75.
5. *Воронин Б.Л., Ерофеев А.М., Копкин С.В., Крючков И.А., Рыбкин А.С., Степаненко С.А., Южаков В.В.* Применение арифметических ускорителей для расчета задач молекулярной динамики по программному комплексу // МД: «Вопросы атомной науки и техники». – Сер. «Математическое моделирование физических процессов». – 2009. – Вып. 2.

С.А. Степаненко

МЕТОД ПРОГНОЗНОГО РАСЧЕТА ПРОИЗВОДИТЕЛЬНОСТИ И ЭФФЕКТИВНОСТИ МУЛЬТИПРОЦЕССОРНЫХ СРЕД

Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Институт теоретической и математической физики, г. Саров,
SAStepanenko@vniief.ru

Компьютерное моделирование является одним из основных современных средств исследований в различных областях естествознания.

Сложность этих исследований требует применения высокопроизводительной вычислительной техники, в частности, параллельных вычислительных систем.

Эти системы позволяют получать решение задачи посредством распараллеливания вычислительного процесса, т.е. одновременного выполнения нескольких процессов.

Вычислительная система содержит вычислительную среду и периферийные компоненты.

Вычислительной средой, следуя [1], будем называть совокупность одинаковых вычислительных устройств, соединенных каналами связи.

Среды, рассматриваемые в этой работе, предназначены для выполнения научных расчетов. Вычислительные устройства в этих средах, помимо средств связи, содержат арифметические процессоры и называются далее процессорными элементами. Каждый процессорный элемент имеет локальную оперативную память, недоступную для других элементов. Такие среды будем называть распределенными мультипроцессорными средами или просто средами.

Определяющим параметром вычислительной системы является производительность ее мультипроцессорной среды.

В идеале производительность среды в процессе вычислений является суммой производительностей ее процессорных элементов. Реальная производительность может быть сильно меньше идеальной; это следствие затрат времени на выполнение обменов информацией между процессорными элементами, отказов отдельных процессорных элементов и т.п. Отношение реальной производительности, достигаемой на заданной программе, к идеальной называется эффективностью среды (на этой программе).

Эффективность зависит от особенностей решаемой задачи и параметров мультипроцессорной среды.

К особенностям задачи, точнее – алгоритма ее решения, относятся длительности нераспараллеливаемых фрагментов, количество и тип операций обмена информацией, синхронность вычислительных процессов и т.п. Создание прикладных алгоритмов и программ, позволяющих максимально использовать возможности вычислительных систем, является специфичной, проблемно-ориентированной областью математики и программирования [2,3] и здесь не рассматривается.

Параметрами мультипроцессорных сред, определяющими их эффективность, являются количество процессорных элементов, соотношение между арифметической производительностью процессоров и пропускной способностью средств связи, топология связей среды, надежность и т.д.

Значения этих величин зависят от используемых аппаратных и системных программных компонент. Они могут изменяться в значи-

тельных пределах и соответственно влиять на эффективность среды, возможности ее применения.

Мультипроцессорные среды могут содержать тысячи процессорных элементов.

Задача прогнозирования производительности и эффективности создаваемых (проектируемых) мультипроцессорных сред является весьма актуальной. Последнее обусловлено большой стоимостью этих систем, трудоемкостью их создания.

Необходимо иметь средства, позволяющие для определенного класса задач уже на стадии проектирования оценить производительность и эффективность разрабатываемой вычислительной среды. Тем самым определяется возможность ее применения по назначению.

Опубликованные методы прогноза производительности мультипроцессорных сред как в ранних [4,5], так и в последних [6,7] работах, основаны на анализе свойств программ, выбранных в качестве эталонов, и значениях различных параметров вычислительной системы. Учитывается количество арифметических операций, интенсивность обращений к памяти, производительность процессора, емкость КЭШ и оперативной памяти, параметры коммуникационной среды и другие подробные сведения об аппаратных и программных системных компонентах.

Эти сведения требуют подробного тестирования и детального знания программных и аппаратных средств. Они не всегда доступны. Указанные методы сложны, их реализация трудоемка и требует профилирования программ.

Исследуемый в этой работе метод прогнозного расчета производительности и эффективности проектируемой среды на заданной программе требует при выполнении определенных условий лишь значений производительности и эффективности, достигнутых на этой программе определенными компонентами проектируемой среды и в специально оговариваемом случае ранее созданной средой. Метод обобщается на гибридные мультипроцессорные среды, элементы которых содержат универсальные процессоры – MIMD-компоненту и арифметические ускорители – SIMD-компоненту.

Метод основан на использовании:

- принципа идентичности топологий мультипроцессорных сред;
- условий соблюдения баланса между пропускной способностью средств связи и арифметической производительностью среды.

Идентичность топологии мультипроцессорной среды часто встречается в современной практике. Это обусловлено модернизацией эксплуатируемых систем, применением программ маршрутизации, использующих топологические особенности среды, а также созданием гибридных мультипроцессорных сред посредством добавления в универсаль-

ные процессорные элементы (реализующие MIMD-вычисления) арифметических ускорителей (реализующих SIMD-вычисления).

Соблюдение идентичности топологий мультипроцессорных сред на различных уровнях параллелизма позволяет упростить процесс расчета значений производительности и эффективности.

Баланс между пропускной способностью средств связи и арифметической производительностью среды необходим для сохранения постоянства доли длительности обменов при выполнении вычислительного процесса к общей длительности процесса, определяющей значение эффективности.

Достоинством излагаемого метода является сравнительная простота и небольшое количество требуемых исходных данных. Недостатком – весьма жесткие, не всегда приемлемые на практике условия идентичности топологии и соблюдения баланса, требуемые для его применения. Метод не заменяет известные средства анализа производительности и эффективности. Он дополняет их в определенных выше условиях.

1. *Евреинов Э.В., Косарев Ю.Г.* Однородные универсальные вычислительные системы высокой производительности. – Новосибирск: Наука, 1966. – 308 с.
2. *Воеводин В.В.* Математические модели и методы в параллельных процессах. – М.: Наука, 1986.
3. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991. – 364 с.
4. *Софронов И.Д.* Оценка параметров вычислительной машины, предназначенной для решения задач механики сплошной среды // Численные методы механики сплошной среды. – 1975. – Т.6. – №3. – С.86 – 112.
5. *Бурцев В.С.* Новые подходы к оценке качества вычислительных средств. Параллелизм вычислительных процессов и развитие архитектуры супер-ЭВМ. – М., 1997. – С. 28 – 40.
6. *Barker K.J., Davis K., Hoisie A., Kerbyson D.J., Lang M., Pakin S., Sancho J.C.* Performance Analysis and Modeling: From Giga-to Peta-scale; http://www.c3.lanl.gov/pal/publications/bibtex/KD_Davis04:BlueGeneL.bib
7. *Barrett B., Barrett R., Brandt J. and others.* Report of Experiments and Evidence for ASC L2 Milestone 4467 – Demonstration of a Legacy Application's Path to Exascale; Sandia Report, SAND2012-1750, Printed March 2012.
8. *Цилькер Б.Я., Орлов С.А.* Организация ЭВМ и систем. – СПб: Питер, 2004.
9. Bob Alverson, Edwin Froese, Larry Kaplan and Duncan Roweth. Cray Inc. Cray XC Series Network. http://www.cray.com
10. *Степаненко С.А.* Оценки ускорения вычислений гибридными системами. // Пленарные доклады Пятой международной конференции «Параллельные вычисления и задачи управления»> PACO 2010, Москва (26-28 октября 2010г). – М.: ИПУ им. В.А. Трапезникова РАН, 2010. – С. 61 – 71, ISNB 978-5-91450-062-4.
11. Infiniband Roadmap. [Электронный ресурс] Режим доступа: http://www.infinibandta.org/content/pages.php?pg=technology_overview.

И.А. Терехова, Е.В. Талдыкин, В.А. Усатый

РЕКОНФИГУРИРУЕМАЯ ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПОИСКА И АНАЛИЗА ДАННЫХ В РЕАЛЬНОМ МАСШТАБЕ ВРЕМЕНИ

НИИЦ (г. Курск) ФГУП «18 ЦНИИ» МО РФ, г. Курск

Одной из наиболее актуальных и практически значимых задач для систем обнаружения вторжений в компьютерные сети и аналитической обработки информации в настоящее время является задача автоматизации кластеризации данных, транспортируемых в спутниковых каналах связи. Особое место при этом отводится процедурам выделения данных априорно-неизвестной структуры (ДАНС).

С целью осуществления автоматизированной обработки данных при мониторинге трафика спутниковых каналов связи разработан метод выделения ДАНС, структурно-функциональная реализация которого представлена на рис. 1.

Входной файловый поток Pf_j состоит из объектов F_i, заданных значениями признаков x_k, $k=1,..,N$, наборы которых одинаковы для всех объектов. Описание объекта F_i определяется совокупностью признаков $I(F_i)=(x1(F_i), \ldots, xN(F_i))$, *где* $i, k, j \in N$. В блок формирования исходных данных поступает *i-й* объект потока Pf_j. На выходе F_i и его сгенерированное описание $I(F_i)=(x_1(F_i), \ldots, x_N(F_i))$.

На вход блока, осуществляющего процедуру классификации, поступает сформированное описание объекта F_i. Формируется информационный вектор, на основе которого принимается решения об отнесении объекта F_i к тому или иному классу $\mathrm{Kf_r}$.

$$\propto (\mathrm{F_i}) = (\propto_1 (\mathrm{F_i}), \ldots, \propto_m (\mathrm{F_i})), \qquad (1)$$

где $\propto_r \in \{0\ (\mathrm{F_i} \notin \mathrm{Kf_r}), 1(\mathrm{F_i} \in \mathrm{Kf_r}),\ \Delta\ (\text{неизвестно})\}$.

Разбиение рассматриваемого множества объектов Pf_j на классы $\mathrm{Kf_r}$ задаётся указанием некоторых признаков, присущих всем его членам. Выходными данными блока является множество $\langle \mathrm{F_i}, \mathrm{I(F_i)}, \mathrm{class} \rangle$, где $class \in \mathrm{K}f_r$ и однозначно определен для анализируемого F_i.

Данные известной структуры (текст, мультимедиа и т.д.) записываются в базу и итоговое хранилище сортированных файлов (ИХСФ). Над файлами, принадлежащими классу $Kf_{ДАНС}$, проводится процедура детального анализа и принятия решения по варианту A (рис. 1).

После обработки потока в модулях 1-4 (рис. 1) выделяются дополнительные признаки каждого F_i и формируется матрица отличий RA_b, где $b \in [1, \ldots, 4]$. Элементами заполнения являются значения расстояний r_{ji} – мера близости между объектами F_i и F_j (чем ближе, тем больше значение), r_{ji}=1, тогда и только тогда, когда F_i=F_j.

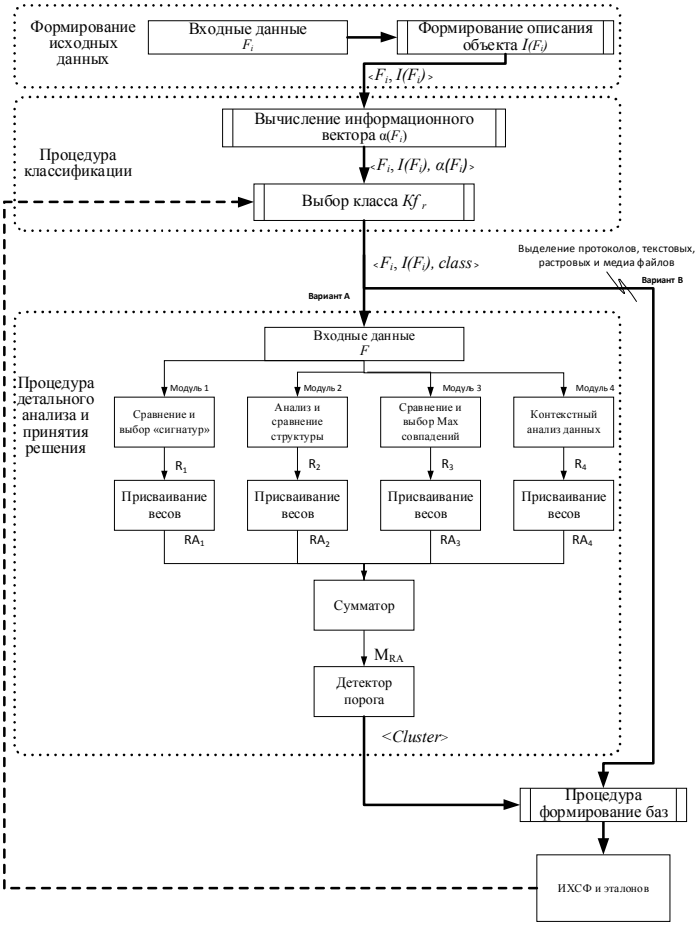


Рис. 1. Структурно-функциональная реализация метода

$$RA_b = \begin{pmatrix} 1 & r_{21} & r_{j1} \\ r_{12} & 1 & r_{j2} \\ r_{1i} & r_{2i} & 1 \end{pmatrix} \quad (2)$$

Вектор расстояний $r'_i=\{r_{1i},\ldots,r_{ji}\}$, $0 \le r_{ji} \le 1$ – расстояния от одного объекта F_i до всех объектов выборки Pf_j.

Таким образом, на выходе каждого модуля получается таблица, строки и столбцы которой помечены названиями объектов F_i. Далее

формируется матрица отличий $M_{RA}=\sum_{l=1}^{4} RA_l$ для принятия решения о принадлежности $F_i \in Cluster_g$ и массив кластеров $Cluster=\{Cluster_1, ..., Cluster_g\}$, $Cluster_g$ – кластер, содержащий похожие объекты из множества Pf_j, g – количество образовавшихся кластеров.

В настоящее время ДАНС составляют более 40% объема любых потоков, транспортируемых в спутниковых каналах связи. При символьной скорости ИСЗ Intelsat 904 1,8 Гбит/с это соответствует 7,56 Тбайт данных в сутки. Обработка этого объёма данных предложенным методом на современной ПЭВМ составит около 5 суток.

Для обеспечения обработки ДАНС в реальном масштабе времени, учитывая скоростные характеристики входного потока, необходимо аппаратное средство, адаптированное под решаемую задачу.

В связи с возможным изменением структур ДАНС, наиболее целесообразным является использование реконфигурируемых вычислительных систем (РВС) на базе программируемых логических интегральных схем (ПЛИС). Использование ПЛИС-технологии позволяет создавать большие вычислительные поля, обладающие возможностями «тонкой» настройки архитектуры под решаемую задачу [1]. За счет адаптации архитектуры вычислителя под структуры ДАНС такие системы позволяют достигать высокой реальной производительности.

Проведен анализ эффективности использования различных ПЛИС в реконфигурируемом вычислительном модуле (РВМ) для параллельно-конвейерной реализации блока анализа и кластеризации ДАНС (рис. 4), позволяющей максимально задействовать внутренние ресурсы ПЛИС. Для оценки эффективности РВМ предлагается использование показателя $D\left(\frac{\text{производительность РВМ}}{\text{стоимость ПЛИС}}\right)$.

Согласно экспериментальным данным на кристалле ПЛИС 5CEBA9 размещается 5 блоков анализа и кластеризации. При этом на один блок задействовано около 20 500 адаптивных логических модулей (ALM).

Для ПЛИС фирмы Xilinx количество адаптивных модулей получено путем пересчета количества секций (Slices) в показатели эквивалентные ALM.

Из данных, представленных в таблице, следует, что варианты разработки РВМ с использованием современных относительно дешевых ПЛИС средней производительности фирмы Xilinx (Kintex-7 XC7A105) и фирмы Altera (Cyclon-5 5CEA9F23) существенно выигрывают перед высокопроизводительными дорогими ПЛИС. Максимальный показатель D достигается в ПЛИС Cyclone-5. Соответственно, рекомендуется использовать ПЛИС 5CEA9F23.

На рис. 2 представлена структурная схема РВМ обработки ДАНС в реальном масштабе времени со скоростью входного потока до 12,8 Гбит/с и производительностью около $\mathbf{1,7 * 10^{13}}$ операций в секунду.

Таблица

Данные по вариантам РВМ на различных типах ПЛИС

Семейство ПЛИС, Тип ПЛИС	Кол-во (ALM) в ПЛИС	Блоков в ПЛИС	Размер ПЛИС, мм	Кол-во ПЛИС на плате РВМ/ рабочая частота MHz	Общее количество блоков на плате	Производительность платы РВМ, оп/сек Стоимость ПЛИС, тыс. руб.	Стоимость одного блока, руб.
Stratix-V 5SCD3	317 000	15	29 x 29	2/200 MHz	30	$3{,}1*10^{13}$ 1196,0	(598 000*2)/ 30 = **39866**
Cyclone-5 5CEBA9F23	113 560	5	23x23	4/150 MHz	20	$1{,}7*10^{13}$ 93,5	(23376*4)/20 = **4675,2**
Virtex-6 XC6VCX240	112 320	4	35x35	2/200 MHz	8	$1{,}12*10^{13}$ 179,17	(89585,54*2)/8 = **22396,38**
Artix-7 XC7A105	58 320	2	15x15	9/150 MHz	18	$1{,}9*10^{13}$ 150,17	(16686,48*9)/ 18 = **8343,24**
Kintex-7 XC7K160T	91 260	4	27x27	2/200 MHz	8	$0{,}91*10^{13}$ 49,96	(24980,40*2)/8 = **6245,1**
Virtex-7 XC7V585T	160 920	7	42,5x 42,5	1/200 MHz	7	$0{,}8*10^{13}$ 303,3	(303334*1)/7 = **43333,42**

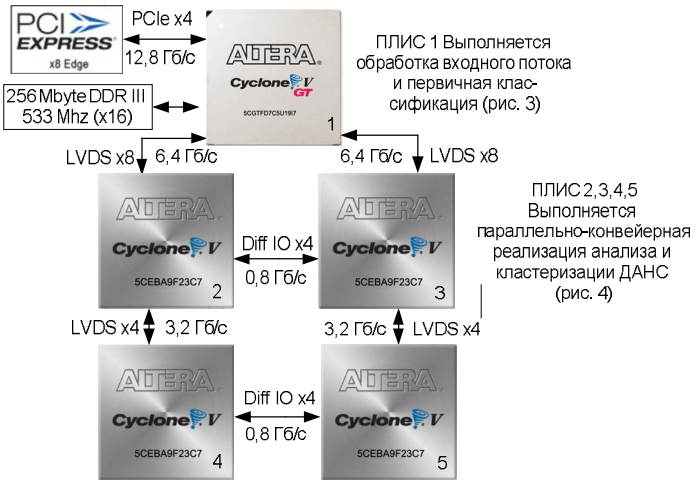


Рис. 2. Структурная схема реконфигурируемого вычислительного модуля

Обработка входного потока и первичная классификация ДАНС осуществляется в ПЛИС 5CGTFD7. Затем данные распараллеливаются на массив блоков анализа и кластеризации ДАНС, размещенных в четырех ПЛИС 5CEA9F23 (рис. 3).

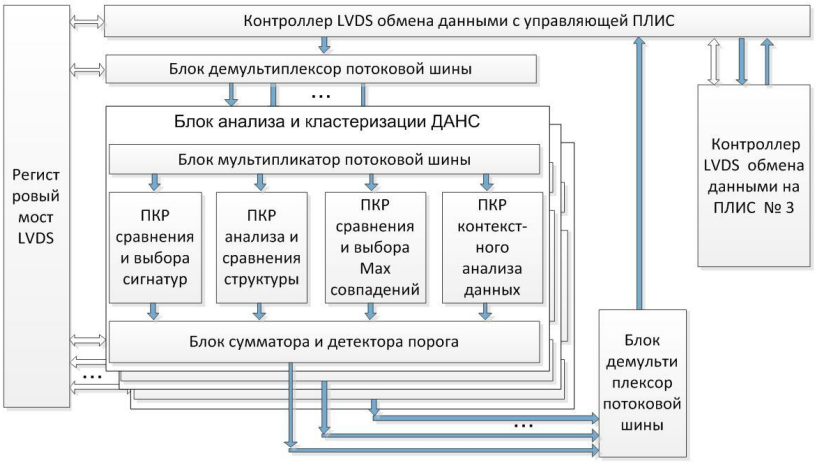


Рис. 3. Структурно-функциональная реализация «системы на кристалле», выполняющая блок ПКР кластеризации ДАНС

Параллельно-конвейерная реализация метода выделения ДАНС на предлагаемом реконфигурируемом вычислительном модуле позволит обрабатывать в реальном масштабе времени до четырех потоков Intelsat 904.

1. *Каляев И.И., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – 2-е изд., перераб. и доп. / под общ. ред. И.А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
2. *Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд.* Алгоритмы: построение и анализ. – 2-е изд. / пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1296 с.
3. *Скиена С.* Алгоритмы. Руководство по разработке. – 2-е изд.: пер. с англ. – СПб.: БХВ-Петербург, 2011. – 720 с.
4. *Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д.* Структуры данных и алгоритмы / Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 384 с.
5. *Фу К.* Структурные методы в распознавании образов. – М.: Мир, 1977.

В.В. Цветков

РЕАЛИЗАЦИЯ ПОДПРОГРАММЫ УМНОЖЕНИЯ МАТРИЦ НА ВЕКТОРНОМ СОПРОЦЕССОРЕ

Научно-исследовательский институт системных исследований РАН,
г. Москва,
tsvetkov@cs.niisi.ras.ru

В НИИСИ РАН разрабатывается линейка микропроцессоров с архитектурой КОМДИВ для выполнения встраиваемых приложений, требующих высокой производительности [1]. С целью повышения производительности разрабатываемых систем в ядра микропроцессоров встраиваются специализированные сопроцессоры, ориентированные на решение заданного типа задач, а в рамках создания программной поддержки разрабатываются специализированные библиотеки.

В работе [2] представлен выбранный нами подход к созданию специализированной библиотеки подпрограмм линейной алгебры для векторного сопроцессора. Векторный сопроцессор позволяет выполнять операции одинарной и двойной точности над векторами шириной 128 разрядов и выполнять загрузку двух 128-разрядных регистров.

В данной работе приводится описание и результаты тестирования базовой библиотечной функции перемножения матриц GEMM, входящей в состав этой библиотеки.

GEMM выполняет операцию перемножения матрицы A на матрицу B: C = A*B*a + C*b, где C – матрица результата, a и b –скаляры.

При реализации функции GEMM в нашей библиотеке используется блочный алгоритм перемножения матриц, предложенный K. Goto. Детали реализации алгоритма изложены в работе [3].

На верхнем уровне обработки матрицы разбиваются на панели, расположенные вертикально в матрице A и горизонтально в матрице B. На этом уровне алгоритм перемножения матриц реализуется через алгоритм перемножения панели на панель (GEPP). Алгоритм перемножения панели на панель в свою очередь реализуется через алгоритм GEBP, выполняющий перемножение блока на панель. Панель разбивается на столбцы шириной n_r. Умножение блока на панель сводится к многократному вычислению умножения блока на панельный столбец.

Эффективность вычислений, т.е. отношение времени, потраченного на вычисления, ко времени на перемещение данных при использовании алгоритма GEBP возрастает с увеличением размеров блока. Ограничения на размер блока могут накладываться как со стороны размеров кэш-памяти, так и со стороны ассоциативной таблицы преобразования виртуальных адресов (TLB).

Уменьшить количество обращений к TLB при загрузке из памяти элементов блока или панели можно, если данные блока и панели скопировать в буфера, в которых данные будут уже представлены в памяти непрерывно. Задача копирования преследует цель не только разместить в памяти данные блока и панели последовательно, но и разместить их таким образом, чтобы обеспечить оптимальный режим загрузки данных в регистры и выполнения вычислений. Отметим, что накладные расходы по времени на копирование при больших размерах матриц оказываются незначительными, поскольку время копирования пропорционально квадрату, а время вычислений – кубу размерности задачи.

Остановимся более подробно на реализации алгоритма GEBP на регистровом уровне. При вычислении произведения блока на панель выполняется разбиение блока на подматрицы размером $m_r * k_c$, а панели на столбцы размером $k_c * n_r$. Алгоритм GEBP реализуется через алгоритм умножения подматрицы блока на столбец панели (рис.1).

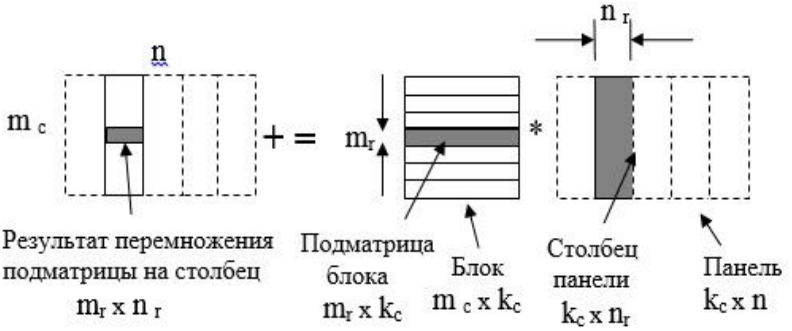


Рис. 1. Реализация алгоритма перемножения блока на панель GEBP через алгоритм перемножения подматрицы блока на столбец панели

Перемножение подматрицы блока на столбец панели выполняется поэтапно. Подматрица и панельный столбец нарезаются на фрагменты. Сначала перемножаются фрагменты из первых k_r столбцов подматрицы на фрагмент из k_r строк панельного столбца, затем к результату прибавляется результат перемножения следующей пары фрагментов и так далее до завершения процедуры перемножения всех фрагментов в подматрице и панельном столбце. На каждом этапе перемножаемые фрагменты загружаются в векторные регистры сопроцессора. Процедура поясняется на рис. 2, где символами Va_i, Vb_i и Vt_i обозначены векторные регистры сопроцессора, i обозначает номер регистра.

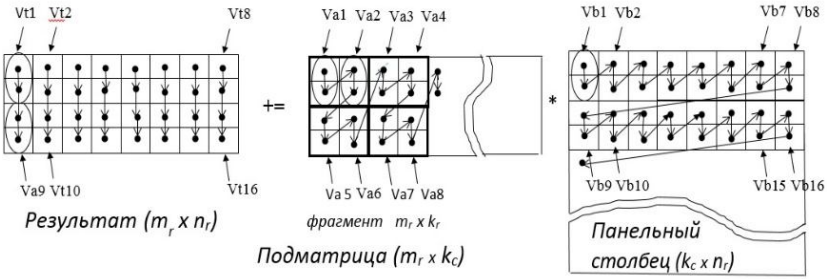


Рис. 2. Реализация на регистровом уровне алгоритма перемножения подматрицы на панельный столбец (пример для m_r=4, n_r=8, k_r=4)

В каждый векторный регистр загружаются по два расположенных друг под другом элемента подматрицы или панельного столбца. При копировании блока или панели в соответствующие буфера элементы в память укладывались в последовательности, указанной стрелками на рисунке. В результате вектора оказываются уложенными в память последовательно в соответствии с их порядковыми номерами. Это позволяет при загрузке регистров использовать команды одновременной загрузки двух или четырех данных двойной точности.

Два соседних регистра в подматрице образуют элементарную матрицу размером 2x2 элемента. В этом случае фрагмент подматрицы в дальнейших рассуждениях удобно рассматривать как матрицу, в которой элементы столбцов и строк представлены элементарными матрицами, а фрагмент панели – как матрицу, элементы которой представлены 2-компонентными векторами.

При умножении загруженных в регистры фрагментов подматрицы и панели применяются специализированные команды умножения матрицы размером 2x2 элемента на 2-компонентный вектор с накоплением результата.

Перемножение фрагментов на регистровом уровне выполняется следующим образом. Первая элементарная матрица в первом столбце выбранного фрагмента подматрицы поочередно умножается на все вектора в первой строке фрагмента панельного столбца, формируя промежуточный результат в регистрах первой строки результата. Такая же процедура, но с сохранением промежуточного результата в регистрах второй строки результата, выполняется для второй матрицы из первого столбца фрагмента подматрицы. Аналогичным образом перемножаются матрицы из второго столбца фрагмента подматрицы на вектора из второй строки фрагмента панели, суммируя результаты с результатами умножения матриц из предыдущего столбца.

Основными параметрами, определяющими эффективность работы алгоритма, являются размер блока (m_c x k_c), ширина панельного столбца (n_r) и высота подматрицы блока (m_r). Последние два параметра определяют размер блока регистров, в которых накапливается результат перемножения подматрицы на панельный столбец.

Ниже представлены результаты выполнения тестов Linpack при использовании специализированной библиотеки, в которой функция GEMM адаптирована под векторный сопроцессор, а также представлены результаты автономного тестирования этой функции при вызове ее из специализированной библиотеки. Результаты приведены для данных, представленных вещественными числами двойной точности. В использованной версии библиотеки при реализации функции GEMM был применен описанный выше алгоритм с параметрами m_c=144, k_c=112, m_r=2, n_r=8. Загрузка данных в векторные регистры выполнялась с использованием команд VLDM, выполняющих загрузку одного вектора. Векторный сопроцессор моделировался на ПЛИС ***ALTERA*** на тактовой частоте 62,5 Мгц.

Результаты измерения коэффициента ускорения (K) при перемножении матриц вещественных чисел приведены на рис. 3. Коэффициент ускорения определялся как отношение количества тактов, затраченных на выполнение функции на скалярном (fpu) и на векторном сопроцессорах.

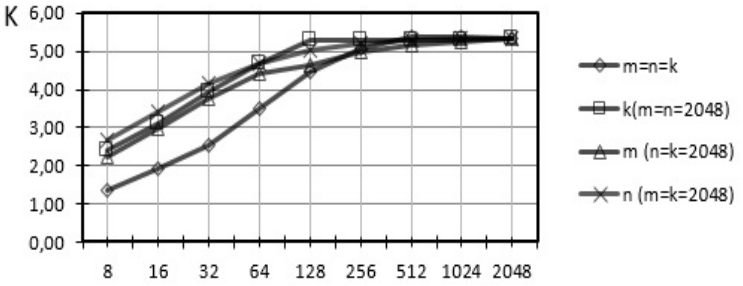


Рис. 3. Коэффициент ускорения выполнения функции GEMM на векторном сопроцессоре

Результаты выполнения тестов Linpack при использовании специализированной библиотеки представлены на рис.4. Использовалась версия hpl-1 пакета высокопроизводительных тестов Linpack для систем с распределенной памятью [4]. На рисунке приведены результаты измерения производительности для однопроцессорной системы.

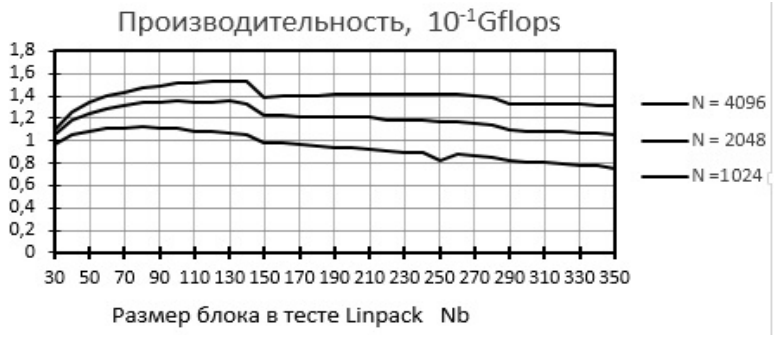


Рис. 4. Производительность векторного сопроцессора на тесте Linpack при различных размерностях задачи

При больших размерностях задачи основное время при выполнении тестов Linpack затрачивается на выполнение функции GEMM. Сокращение времени выполнения этой функции при выполнении ее на векторном сопроцессоре дает существенное увеличение производительности. Так, при размерностях задачи, например N = 4096, было получено увеличение производительности более чем в 4 раза при выполнении тестов Linpack на векторном сопроцессоре по сравнению с выполнением их на fpu.

1. *Бобков С.Г., Аряшев С. И., Зубковский П.С.* Арифметические сопроцессоры микропроцессоров с архитектурой КОМДИВ / Программа и тезисы докладов 6-го Московского суперкомпьютерного форума. – М., 2015. – С. 14.
2. *Аряшев С.И., Зубковский П.С., Кулешов А.С., Цветков В.В.* Адаптация библиотеки подпрограмм линейной алгебры GOTOBLAS к архитектуре векторного сопроцессора // Суперкомпьютерные технологии: Материалы 3-й Всеросс. науч.-техн. конф. – Ростов-на-Дону: Изд-во ЮФУ, 2014. – Т.1. – С.90.
3. *Kazushige Goto, Robert A. Van De Geijn.* Anatomy of High-Performance Matrix Multiplication // ACM Transaction on Mathematical Software, 2008. – Vol. 34.
4. www.netlib.org/benchmark/hpl/

РАЗДЕЛ 2

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СУПЕРКОМПЬЮТЕРОВ

А.А. Белозеров, С.Ю. Мельников, В.А. Пересыпкин

ТЕХНОЛОГИИ ОЧИСТКИ ТЕКСТОВЫХ КОРПУСОВ ДЛЯ РАЗРАБОТКИ МОДЕЛЕЙ ЯЗЫКА

ФГУП «НТЦ «Орион», ООО «Линфо», г. Москва,
melnikov@linfotech.ru

Современные задачи автоматической обработки текста и речи требуют наличия так называемых языковых моделей, отражающих основные закономерности естественных языков. Модель языка строится с использованием большого набора текстов достаточно объема на данном языке (такие наборы называются обучающими корпусами). Для наиболее распространенных мировых языков доступны представительные корпуса, распространяемые свободно или на коммерческой основе. Для редких и так называемых «малоресурсных» языков таких корпусов может не быть.

При автоматическом наборе корпусов по интернет-источникам, в частности, с помощью программной системы, описанной в [1], в корпус может попадать нецелевая текстовая информация. Эта информация, например, может содержать многократное дублирование каких-то сообщений, нетекстовые фрагменты или фрагменты текстов на других языках. Для формирования точных языковых моделей (необходимых, в частности, для систем коррекции искаженных текстов [2]) эту избыточную информацию необходимо убирать, проводя анализ и редактирование («очистку») корпуса.

В работах [3] и [4] определены несколько числовых характеристик текстовых корпусов: число токенов (слов), так называемое отношение TTR (Type-Token-Ratio), средняя длина слова, средняя длина предложения (вычисленные в словах). Это одни из важнейших числовых характеристик корпуса, и по их поведению можно судить о качестве собранного корпуса и о достоинствах/недостатках процедур очистки, использованных при создании корпуса.

Для больших корпусов (объемом в миллиарды слов) процедуры очистки проводить вручную невозможно, поэтому возникает необходимость создания автоматических или, по крайней мере, удобных интерактивных процедур.

Основные причины «зашумления» корпусов

При анализе набранных корпусов были выявлены следующие причины появления нецелевой информации:

- дублирование (возможно, нечеткое) одних и тех же текстов,
- наличие в текстах фрагментов на нецелевых языках и нетекстовой информации (содержание и оформление таблиц, остатки html разметки и подобная информация).

Наиболее заметной в собираемых материалах оказалась проблема дублирования текстов. Явление дублирования связано с перепечаткой новостными мировыми агентствами материалов друг друга с минимальным редактированием или без него. В дублирующих текстах могут быть ссылки на источник, а могут и не быть. Степень дублирования определяется политикой новостных агентств, газет, журналов и организаций – источников новостей, их связями друг с другом, важностью и особенностями самого перепечатываемого материала.

Были разработаны и программно реализованы три процедуры анализа и редактирования корпусов:

1. Удаление нечетких дубликатов.

Обзор и анализ алгоритмов поиска нечетких дубликатов приведен в [5], в описываемой системе реализован метод поиска нечетких дубликатов по словарному составу.

2. Отбор текстов по списку редко встречающихся слов.

Список формируется из редко встречаемых слов (с частотой встречаемости во всем корпусе меньше заданной) и непереведенных системой автоматического перевода слов из оставшихся. Отсекаются тексты, в которых процентное содержание слов из сформированного списка превышает заданный порог.

3. Отбор текстов по частоте встречаемости символов алфавита.

Процедура направлена на поиск текстов, содержащих большое количество символов, отсутствующих в алфавите языка. Таким текстом может оказаться, например, некорректно скачанный текст с присутствующей в нем html-разметкой в виде тегов. Задается базовый алфавит символов и вычисляется процентное отношение этих символов в тексте к общему числу символов. Отбор текстов происходит по фиксированному порогу.

О программной реализации процедур анализа

Процедуры реализованы программно и позволяют обрабатывать текстовые корпуса объемом в десятки гигабайт в приемлемое время. Приведем таблицу со значениями времени выполнения указанных программных процедур на компьютере Intel(R) Core(TM) i7-3970x CPU 3.5 GHZ, 12 ядер, ОЗУ 64 Гб. Обрабатывались три корпуса текстов на английском языке объемом 100 Мб, 500 Мб и 1 Гб.

Таблица

Значения времени выполнения программных процедур на компьютере Intel(R) Core(TM) i7-3970x

Объем корпуса (в Мб)	Отбор дубликатов (время в секундах)	Отбор статей по редким словам (время в секундах)
100	58	20
500	649	101
1024	2150	191

Трудоемкость процедуры отбора дубликатов зависит от объема корпуса нелинейно, так как используется сравнение каждой статьи с каждой. Время обработки удается уменьшить за счет сравнения не самих текстов, а их словарей или фрагментов статей. Кроме того, время сокращается за счет удаления из корпуса статей дубликатов обнаруженных на предыдущих шагах.

Трудоемкость процедуры отбора статей по списку подозрительных слов линейно зависит от объема корпуса. В расчет не берется время построения списка.

Результаты и выводы

При обработке процедурой удаления нечетких дубликатов, в зависимости от близости собираемых источников, от 30% (для малоресурсных языков, например, татарского) до 60% (для английского) текстов были определены как дубликаты и удалены из корпуса.

Процедура отбора текстов по списку редко встречающихся слов оказалась эффективной при удалении текстов на языках, отличных от языка корпуса, а также текстов, не подходящих по содержанию из-за большого присутствия имен собственных, аббревиатур, слов с ошибками написания.

Предложенные процедуры очистки корпусов позволяют значительно уменьшить размер строящихся языковых моделей и сократить время их построения, сохраняя при этом точность моделей.

1. *Белозеров А.А., Мельников С.Ю., Пересыпкин В.А.* Технологические аспекты построения системы сбора корпусов текстов // Суперкомпьютерные технологии: Материалы 4-й Всероссийсой научно-технической конференции. – Ростов-на-Дону: Изд-во ЮФУ, 2016. – Т.1. – С. 115 – 119.
2. *Мельников С.Ю., Пересыпкин В.А.* О применении вероятностных моделей языка для обнаружения ошибок в искаженных текстах // Вестник компьютерных и информационных технологий. – 2016. – №5. – С.29 – 33.

3. *Eckart T., Quasthoff U., Goldhahn D.* The Influence of Corpus Quality on Statistical Measurements on Language Resources // in: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). – Istanbul, Turkey. – 2012. – Pp. 2318 – 2321.
4. *Sarkar A., De Roeck A., Garthwaite P.* Easy measures for evaluating non-English corpora for language engineering. Some lessons from Arabic and Bengali // Dep. of Comp., Faculty of Math. and Comp., The Open University, Walton Hall, UK. Tech. Rep. – №2004/05. – Pp. 1 – 5.
5. *Зеленков Ю.Г., Сегалович И.В.* Сравнительный анализ методов определения нечетких дубликатов для Web-документов // Тр. 9-й Всеросс. научн. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». (RCDL'2007). – Переславль-Залесский. – 2007.

А.А. Белозеров, С.Ю. Мельников, В.А. Пересыпкин

ТЕХНОЛОГИЧЕСКИЕ АСПЕКТЫ ПОСТРОЕНИЯ СИСТЕМЫ СБОРА КОРПУСОВ ТЕКСТОВ

ФГУП «НТЦ «Орион», ООО «Линфо», г. Москва,
melnikov@linfotech.ru

В современных системах распознавания речи [1], рукописного текста [2], системах коррекции [3] используются так называемые модели языка, которые позволяют оценивать правдоподобие вариантов распознанного текста, вычислять степень их согласия с основными статистическими характеристиками языка. Модель языка строится с помощью того или иного способа статистической обработки набора текстов большого объема на данном языке (такие наборы называются корпусами). Если для наиболее распространенных мировых языков доступны представительные корпуса, распространяемые свободно или на коммерческой основе, то для редких и так называемых «малоресурсных» языков таких корпусов может и не быть. Для ряда приложений необходимо иметь языковые корпуса, находящиеся в актуальном состоянии, прежде всего по лексическому составу ([4]). В работе предлагается решение задачи создания корпусов на основе программной системы сбора текстов по Интернет-источникам.

Задача сбора корпусов по Интернет-источникам решалась разными исследователями, для этого использовались различные способы. Типичный случай описан в работе [5], где решалась задача сбора текстовых корпусов славянских языков для построения системы распознавания речи. Программное средство позволяло от заданного «начального» URL идти по ссылкам на глубину 20, собирая html-страницы и проводя после этого их «парсинг» (разбор) для выделения текста. К недостаткам тако-

го подхода относится непредсказуемость и плохая контролируемость состава собираемого корпуса (по ссылке можно уйти на рекламные сайты, сайты на других языках и др.).

В данной работе предложен иной подход к сбору текстовых корпусов, требующий участия экспертов-лингвистов, но обеспечивающий высокое качество собираемого текстового материала, и описана программная система, реализующая этот подход.

Поиск и выбор информационных ресурсов

На первом этапе лингвисты с помощью универсальных поисковых систем (Google, Bing, Ask, Yahoo), а также национальных поисковых машин составляют максимально полный перечень информационных ресурсов на целевом языке. Значительно повышает эффективность поиска использование запросов и механизмов расширенного поиска, позволяющих, например, искать только сайты на определённом языке или в определённой географической зоне. Использование данных механизмов заметно увеличивает релевантность поисковой выдачи и сокращает трудоёмкость работы эксперта.

Создаваемые корпуса текстов, с одной стороны, должны быть достаточно универсальны (для построения адекватных моделей языков), а с другой – отражать лишь определённый пласт языка, а именно, современную письменную лексику необходимой направленности. Наиболее соответствующими указанным требованиям информационными ресурсами в сети Интернет являются сайты сетевых и печатных СМИ, телеканалов, радиостанций, информационных агентств, общественно политические сайты и новостные ленты. Отдельным источником информации могут служить различные Wiki-ресурсы – сетевые энциклопедии, однако они характеризуются невысокой актуальностью, так как многие упоминаемые издания прекратили существование.

Например, в результате применения предложенного механизма поиска для арабского языка было найдено более 5 600 сайтов различных СМИ, организаций, партий, политических движений и др. Из этих сайтов более 3 700 пригодны для сбора материала в корпус арабского языка.

На последующих этапах подобранные перечни ресурсов обрабатываются разработанным программным обеспечением.

Получение ссылок на страницы

По результатам анализа материалов новостных сайтов выбраны следующие три способа получения ссылок на статьи, опубликованные на этих сайтах:

1. Получение и разбор RSS-лент с последующим выкачиванием статей по ссылкам из этих лент.
2. Получение ссылок на статьи из файлов Sitemap.xml (карт сайтов).
3. Получение ссылок на новые материалы из Twitter.

Таблица

Сравнительные характеристики источников информации

Характеристика	RSS	Sitemaps	Twitter
Сложность получения данных	Легко	Легко	Легко
Скорость обновления (в среднем)	Высокая	Средняя в зависимости от конкретного сайта	Высокая
Необходимость использования стороннего API	Нет	Нет	Да, работа с ограничениями
Влияние на работоспособность системы	Нет	Нет	Да
Подходит для сбора архивных материалов	Нет	Да	Зависит от ограничений
Разбиение статей по тематике	В некоторых случаях	Нет	В некоторых случаях

Программная система сбора корпусов текстов

Программно реализованная система сбора корпусов текстов состоит из следующих модулей: модуля сбора ссылок на новостные материалы и скачивания материалов, модуля хранения информации, интерфейса управления, модуля извлечения корпусов текстов. Система разработана на языке Python.

Модуль сбора ссылок на новостные материалы и скачивания статей. Модуль сбора ссылок на новостные материалы включает в себя робота для скачивания содержимого статей с новостных порталов. Модуль содержит в себе функционал для выкачивания и разбора RSS-лент и карт сайтов, который выделяет из них ссылки на новостные материалы, делает запрос к подсистеме хранения о том, не выкачивались ли статьи по этим ссылкам ранее и далее получает содержимое вебстраниц, размещенных по этим ссылкам. Робот сбора ссылок и выкачивания контента работает в многопоточном режиме. В ходе работы модуль постоянно взаимодействует с подсистемой хранения. Особенностью модуля является алгоритм определения кодировки, в нашем случае кодировка должна определяться с максимальной точностью, так как попадание ошибочно кодированных документов в подсистему хранения приведет к снижению качества извлекаемых корпусов текстов.

Модуль хранения информации. Модуль хранения информации служит для хранения данных, используемых в системе, и обмена информацией между различными модулями системы сбора корпусов текстов.

Модуль хранения информации состоит из нереляционной СУБД и набора скриптов для работы с СУБД, обработки информации и взаимодействия между различными модулями системы. В модуль хранения входит функционал, обеспечивающий работу очереди заданий, выдаваемых роботу сбора ссылок и выкачивания контента, а также функционал, отвечающий за генерацию этих заданий и постановку их в очередь на выполнение.

Информация о картинках, скриптах, стилях, объектах и пр. не хранится, что позволяет значительно экономить место для хранения. Сохраняется информация о ссылках, полученных из RSS-лент и карт сайтов, скачанные документы, логи работы системы, промежуточные итоги извлечения корпусов текстов, список RSS-лент и карт сайтов.

Интерфейс управления. Служит для управления системой сбора корпусов текстов, внесения в систему заданий на обработку RSS-лент, карт сайтов и т.д., просмотра логов работы системы, текущих рабочих характеристик, статистики, а также для отладки алгоритмов извлечения корпусов текстов.

В системе интерфейса управления предусмотрено два типа пользователей – администратор системы и обычный пользователь.

Администратор системы – пользователь, обладающий неограниченными правами, в том числе правами на добавление новых пользователей, изменения очередей обработки данных, просмотра логов работы системы, удаления данных и т.д.

Интерфейс обычных пользователей системы позволяет вносить списки RSS-лент и карт сайтов, которые в последствии будут проверены администратором и добавлены в систему на обработку.

Модуль извлечения корпусов текстов. Модуль извлечения корпусов текстов предназначен для обработки веб-страниц, собранных при помощи модуля сбора данных, взаимодействует только с модулем хранения данных, из которого получает данные для обработки, сохраняет промежуточные результаты обработки, а также отправляет статистику работы, которая в дальнейшем отображается в пользовательском интерфейсе.

В модуле извлечения корпусов текстов реализованы алгоритмы поиска элементов веб-страницы, которые содержат текст статьи, алгоритмы выделения итогового корпуса текста из таких элементов, алгоритм анализа и извлечения шаблонов документов. Эти алгоритмы по-

строены на основе использования оригинальной метрики количества текста.

Аппаратные средства комплекса сбора корпусов текстов

Исходя из специфики решаемых задач, аппаратные средства комплекса сбора корпусов текстов реализованы в виде совокупности серверов и отдельных рабочих мест операторов. Сервера оборудованы средствами удалённого управления и размещаются на технической площадке, удалённой от рабочих мест операторов. Средства удалённого управления могут обеспечить масштабирование системы в случае необходимости, а также мониторинг технического состояния серверов и возможность их удалённого перезапуска.

Результаты и выводы

За время использования системы в обработку добавлено около 20 тыс. различных RSS-лент, содержимое которых выкачивалось и обрабатывалось приблизительно каждые 2 часа, для целевых языков собраны корпуса объемом от 0,5 до 20 Гб.

Используемый подход к сбору данных и его практическая реализация в виде аппаратно-программного комплекса сбора корпусов текстов демонстрируют свою состоятельность и способность с высокой скоростью собирать корпуса текстов хорошего качества. Накопление объёмов корпусов текстов идет непрерывно за счет обновляющихся RSS-лент, что позволяет наращивать объёмы уже созданных корпусов, и осуществлять фильтрацию составляющих их текстов по датам, регионам, тематике и другим параметрам.

1. *Мещеряков Р.В.* Структура систем синтеза и распознавания речи // Известия Томского политехн. ун-та. – 2009. – Т. 315. – № 5. – С. 127 – 132.
2. *Plotz T., Fink G.* Markov models for offline handwriting recognition: a survey // International Journal of Document Analysis and Recognition. – 2009. – Vol. 12. – Pp. 269 – 298,
3. *Мельников С.Ю., Пересыпкин В.А.* О применении вероятностных моделей языка для обнаружения ошибок в искаженных текстах // Вестник компьютерных и информационных технологий. – 2016. – №5. – С.29 – 33.
4. *Смирнов А.В., Круглов В.М., Крижановский А.А., Луговая Н.Б., Карпов А.А., Кипяткова И.С.* Количественный анализ лексики русского WordNet и викисловарей. // Труды СПИИРАН– 2012. – № 23. – С. 231 – 253.
5. *Vu N.T., Schlippe T., Kraus F., Schultz T.* Rapid Bootstrapping of five Eastern European Languages using the Rapid Language Adaptation Toolkit // Interspeech 2010, 26-30 September 2010, Makuhari, Chiba, Japan, PP. 865 – 868.

С.А. Бутенков

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ МОДЕЛЕЙ ЗАДАЧ ДЛЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ИНФОРМАЦИОННОЙ ГРАНУЛЯЦИИ

ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог,
saabmount@gmail.com

Введение

При компьютерном решении задач изучения различных классов методами *вычислительного эксперимента* (ВЭ) обычно необходимо пройти три этапа. На *первом этапе* формализуется описание процесса и строится адекватная математическая модель. На *втором* разрабатывается корректный численный метод для новой модели [1]. Наконец, на *третьем этапе* выполняется имплементация численного метода в аппаратное и программное обеспечение для выбранного типа *вычислительной системы* (ВС) [2]. Классический подход к этой трехступенчатой задаче предусматривает независимое решение проблем, возникающих на каждом этапе [1]. Однако, в силу постоянного и быстрого роста сложности задач, изучаемых методами ВЭ и соответственно требований к ресурсам используемых ВС, становится необходимым учитывать особенности аппаратной и программной реализации высокопроизводительных и особенно реконфигурируемых ВС на начальных этапах проектирования ВЭ [2].

Методологией, которая позволяет связать воедино методологию реализации всех трех этапов проектирования ВЭ (включая аппаратное проектирование), является *информационная грануляция* (ИГ) [3, 4]. Этот *зонтичный термин* включает в себя набор методов приведения исходных данных задачи ВЭ к стандартизованной *гранулированной форме*, для которой решаются указанные выше задачи [4]. Гранулированные данные представляют собой некоторую форму сжатых данных, работать с которыми можно без распаковки [3]. На базе ИГ становится возможным полный цикл проектирования высокопроизводительных ВС, начиная с моделей процессов [5], затем – численных методов для них, позволяющих на основе типовых решений строить экономичные и эффективные реконфигурируемые ВС [2, 6].

Постановка задачи

Для организации моделей ВС с использованием типовых макропроцессорных блоков [2] введем базовые математические определения на основе идей работы [5], позволяющие изоморфно отображать раз-

личные области определения данных задач моделирования в *абстрактное пространство гранулированных данных*, а также отношения между элементами этого пространства [5]. В отличие от методов сеток и конечных элементов [1], гранулированные модели позволяют использовать макросредства для вычислений на гранулах, не являющихся бесконечно малыми элементами [2].

Модель пространства гранулированных данных

Пусть K – произвольное числовое поле, а $n \in \square$ – число. Тогда n-мерным *числовым вектором* над полем K мы будем называть любой кортеж, составленный из n чисел поля K, а n-мерным *числовым пространством* над полем K назовем совокупность всех n-мерных числовых векторов над этим полем [10]. Формализуем его свойства в следующих определениях.

Определение 1. Любое множество $\mathbf{L}$ произвольных элементов называется векторным пространством над данным числовым полем K, если:

1. Имеется некоторая операция, ставящая в соответствие каждой паре элементов $\mathbf{a}, \mathbf{b} \in \mathbf{L}$ некоторый элемент $\mathbf{c} = \mathbf{a} + \mathbf{b}$, $\mathbf{c} \in \mathbf{L}$, называемый суммой.

2. Имеется вторая операция, ставящая в соответствие каждому элементу $\mathbf{a} \in \mathbf{L}$ и каждому числу $k \in K$ $\mathbf{c} = k\mathbf{a}$, $\mathbf{c} \in \mathbf{L}$.

3. Обе операции удовлетворяют следующим аксиомам:

I. Для любых $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbf{L}$ имеют место соотношения:

а) $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$ (коммутативность),

б) $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$ (ассоциативность).

II. В $\mathbf{L}$ существует $\mathbf{0}$ (нулевой элемент) такой, что $\mathbf{a} + \mathbf{0} = \mathbf{a}$ для всех $\mathbf{a} \in \mathbf{L}$.

III. Для всех $\mathbf{a} \in \mathbf{L}$ существует такой элемент $-\mathbf{a}$, называемый противоположным для $\mathbf{a}$, что $\mathbf{a} - \mathbf{a} = \mathbf{0}$.

IV. Для любых $\mathbf{a}, \mathbf{b} \in L$ и любых чисел k_1 и k поля K имеют место соотношения:

а) $k_1(k_2\mathbf{a}) = (k_1 k_2)\mathbf{a}$, б) $(k + k_2)\mathbf{a} = k_1\mathbf{a} + k_2\mathbf{a}$, в) $k_1(\mathbf{a} + \mathbf{b}) = k_1\mathbf{a} + k_1\mathbf{b}$.

V. Для любого $\mathbf{a} \in \mathbf{L}$ имеет место $\mathbf{1a} = \mathbf{a}$.

Рассмотрим теперь m-мерное векторное пространство, на m координатных осях которого определены собственные единичные векторы (орты) $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_m$ и разложение вектора в виде $V = \sum_{i=1}^{m} v_i \mathbf{e}_i$, где v_i – координаты абстрактного вектора. Если в рассматриваемом векторном

пространстве не представляется возможным сравнение длин $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_m$, его называют *аффинным векторным пространством*.

Во многих физических и особенно социологических, химических, биологических и т.п. процессах, изучаемых с помощью ВЭ, имеет место *несоизмеримость* числовых значений анализируемых параметров. Меры на них вводятся искусственно и не всегда корректно [1]. Строгим является подход, основанный на использовании моделей данных в аффинном пространстве [6]. Для аффинного пространства введем понятие *определителя*.

Определение 2. *Определителем* порядка n называется функция от n n-мерных векторов вида $F\left({}^1\mathbf{a}, {}^2\mathbf{a}, ..., {}^n\mathbf{a}\right) = \left|{}^1\mathbf{a}, {}^2\mathbf{a}, ..., {}^n\mathbf{a}\right|$, удовлетворяющая следующим аксиомам:

a) $F\left({}^1\mathbf{a}, {}^2\mathbf{a}, ..., {}^n\mathbf{a}\right)$ линейна по отношению к каждому аргументу.

b) Если среди векторов ${}^1\mathbf{a}, {}^2\mathbf{a}, ..., {}^n\mathbf{a}$ есть хотя бы пара линейно зависимых, то определитель равен нулю.

c) $\left|{}^1\mathbf{e}, {}^2\mathbf{e}, ..., {}^n\mathbf{e}\right| = 1$.

Введем теперь модель гранулированного представления данных в аффинных векторных пространствах.

Определение 3. *Разбиение* конечного универсума $\mathbf{G}$ – это конечное множество подмножеств ${}^iG \in \mathbf{G}$, $i = 1, ..., n$ (атомарных гранул), удовлетворяющих следующим аксиомам:

1. ${}^iG \neq \varnothing, \quad i = 1, ..., n$;

2. ${}^iG \cap {}^jG = \varnothing$ при $i \neq j; \quad i = 1, ..., n; \; j = 1, ..., n;$

3. $\bigcup_i {}^iG = \mathbf{G}, \quad i = 1, ..., n$.

Определение 4. Покрытие τ конечного универсума U – это конечное множество подмножеств ${}^iG$, удовлетворяющих аксиомам 1 и 3.

Определение 5. Разбиение π (или покрытие τ) называется конъюнктивным разбиением (покрытием), если каждый класс эквивалентности из π (τ) – составная гранула.

Определение 6. Разбиение π_1 есть уточнение разбиения π_2 (или π_2 есть обобщение разбиения π_1), обозначаемое как $\pi_1 \prec \pi_2$, если каждая гранула из π_1 содержится в некоторой грануле из π_2. Покрытие τ_1 есть уточнение покрытия τ_2 (или τ_2 есть обобщение покрытия τ_1), обозначаемое как $\tau_1 \preceq \tau_2$, если каждая гранула из τ_1 содержится в некоторой грануле из τ_2.

В работах [3, 4,5] введено общее определение информационной гранулы, которое можно распространить на аффинное пространство.

Определение 7. Пусть заданы произвольные гранулы ${}^1G,...,{}^nG$ размерности 1 для переменных $U_1,...,U_n$ соответственно, тогда их декартово произведение $G_n = {}^1G \times \cdots \times {}^nG$ – это декартова гранула размерности n.

Определение 8. Декартова гранула G^+, определяемая как $G^+ = G_x \times G_y$, *инкапсулирует* произвольную гранулу G в том смысле, что является *точной верхней гранью* декартовых гранул, которые содержат G.

Меры подмножеств пространства гранулированных данных

Значение определителя элементов аффинного пространства является аналогом меры данного элемента, например, при $n=3$ имеем $\eta^3\left({}^1\mathbf{a},{}^2\mathbf{a},{}^3\mathbf{a}\right) = \left|{}^1\mathbf{a},{}^2\mathbf{a},{}^3\mathbf{a}\right|$. Введем теперь набор мер, связанных со специфическими отношениями в пространстве $\mathbf{G}$ гранулированного представления данных размерности n согласно [5].

Определение 9. *Мера общности* гранулы $G \in \mathbf{G}$ задается как $Glob(\mathrm{G}) = \eta^n(\mathrm{G})/\eta^n(\mathbf{G})$, она определяет относительный размер гранулы G.

Определение 10. Мера соответствия пары гранул ${}^iG,{}^jG \in \mathbf{G}$ задается в виде $AS({}^i\mathrm{G},{}^j\mathrm{G}) = \eta^n({}^i\mathrm{G},\cap{}^jG)/\eta^n(\mathbf{G})$.

Определение 11. Мера покрытия пары ${}^iG,{}^jG \in \mathbf{G}$ задается в виде $\mathrm{CV}({}^i\mathrm{G},{}^j\mathrm{G}) = \eta^n({}^i\mathrm{G},\cap{}^jG)/\eta^n({}^j\mathrm{G})$.

Определение 12. *Мера сходства* пары гранул ${}^iG_n, {}^jG_n \in \mathbf{G}$, инкапсулируемых гранулой $G_n^+\left({}^iG_n, {}^jG_n\right) \in \mathbf{G}$, представляется в виде $\mathrm{SIM}({}^i\mathrm{G}_n,{}^j\mathrm{G}_n) = \eta^n\left(G_n^+\right)\Big/\left(\eta^n\left({}^iG_n\right) +^3 \left({}^iG_n\right)\right)$.

В [5] показано, что функция гранул $\mathrm{SIM}({}^i\mathrm{G}_n,{}^j\mathrm{G}_n) \in$ □ является *мерой в аффинном пространстве* данных размерности n. Применение новых мер в задачах грануляции не требует нормы векторного пространства [1, 3].

Заключение

Введенные в работе базовые математические результаты развивают основные положения теории грануляции применительно к по-

строению параллельных систем вычислений для данных неметрической природы. Модель аффинного пространства гранулированных данных допускает преобразование координат, при этом нет необходимости вводить новые нормы. Такие свойства позволяют распространить методологию грануляции элементов на произвольные системы криволинейных координат, с использованием типовых решающих блоков (макросов) [2], что расширяет возможности моделирования различных типов данных (например, цветовых данных). В наших работах показано, что использование предложенных моделей данных позволяет строить эффективные (жадные) алгоритмы линейной сложности для обработки данных [7], а также реконфигурируемых ВС, решающие эти задачи путем организации макропроцессорных блоков на гранулах [2].

1. *Самарский А.А., Михайлов А.П.* Математическое моделирование: Идеи. Методы. Примеры. – 2-е изд., испр. – М.: Физматлит, 2001. – 320 с.
2. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
3. *Pedrysz W.* Granular Computing – the emerging paradigm // Journal of Uncertain Systems, – 2007. – Vol.1. – No.1. – Pp. 38 – 61.
4. *Zadeh L.A.* Toward a Theory of Fuzzy Information Granulation and its centrality in human reasoning and fuzzy logic // Fuzzy Sets Syst. – 1997. – Vol. 90. – Pp. 111 – 127.
5. *Бутенков С.А., Жуков А.Л.* Информационная грануляция на основе изоморфизма алгебраических систем // Сб. трудов Международной алгебраической конференции, посвященной 80-летию со дня рождения А.И. Кострикина. – Нальчик. – 2009. – С. 206 – 209.
6. *Бутенков С.А.* Методы информационной грануляции в параллельных вычислениях // Суперкомпьютерные технологии: Материалы 3-й Всеросс. науч.-техн. конф. – Ростов-на-Дону: Изд-во ЮФУ. – Т.1. – С. 99 – 104.
7. *Бутенков С.А., Нагоров А.Л.* Проектирование численных методов решения уравнений на гранулированных моделях сред // Сб. трудов IX Международной научно-практической конференции “Effektivni Nastroje Modernich Ved – 2013”, Прага, 27 апреля – 05 мая 2013 г., Praha: Publishing House “Education and Science”, 2013. – Т.40. – С. 5 – 9.

В.С. Горбунов, А.Н. Иванов, И.А. Морозов

РЕАЛИЗАЦИЯ И РАЗВИТИЕ КОМПЛЕКТА РАЗРАБОТЧИКА ПРИЛОЖЕНИЙ СУПЕРКОМПЬЮТЕРОВ НА ПЛИС

ФГУП «НИИ «Квант», г. Москва,
gorbunov@rdi-kvant.ru, alexiv@rdi-kvant.ru,
lab1113@rdi-kvant.ru

Одним из наиболее важных компонентов реконфигурируемых вычислительных систем (РВС) на базе программируемых логических интегральных схем (ПЛИС) является комплект разработчика приложений, так как он определяет возможность и эффективность объединения труда программистов и схемотехников при использовании РВС.

В ФГУП «НИИ «Квант» разработан комплект разработчика приложений, состоящий из программной и аппаратной составляющих. Данный комплект предназначен для работы с РВС на базе ПЛИС компании Xilinx, подключенных по шине PCI Express к серверу обработки под управлением ОС Red Hat Enterprise Linux.

Программная составляющая (SDK) включает в себя набор программных средств, обеспечивающих вывод информации о текущем состоянии РВС или ее частей, сброс текущей конфигурации, восстановление конфигурации из флеш-памяти или из файла, прошивку флеш-памяти, а также программные средства, обеспечивающие параллельную многопользовательскую работу с РВС и обмен данными с ПЛИС в пословном, блочном и смешанном режимах. В комплекте SDK также поставляются тесты и набор утилит командной строки, реализующих основные функции управления конфигурацией РВС.

Аппаратная составляющая включает в себя базовый проект для ПЛИС, позволяющий подключать множество идентичных вычислительных устройств, описанных на языках VHDL или Verilog, используя шину AXI. Для организации подключения к шине AXI могут быть использованы интеллектуальные ядра адаптеров интерфейсов, которые содержат буфера с возможностью случайного или последовательного доступа, работающие на частоте тактового сигнала вычислительного ядра, которая может не совпадать с частотой интерфейсного тактового сигнала. Для загрузки и выгрузки данных поддержан режим DMA. Для упрощения работы с множеством одинаковых вычислительных ядер предусмотрен пакетный режим работы, позволяющий передавать набор заданий, распределяемых по вычислителям в автоматическом режиме, получая на выходе набор ответов.

А.А. Гуленок

МЕТОД ИЕРАРХИЧЕСКОЙ ДЕКОМПОЗИЦИИ ГРАФОВ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ W-ОБРАЗНОЙ МОДЕЛИ*

НИИ многопроцессорных вычислительных систем ЮФУ, г. Таганрог, andrei_gulenok@mail.ru

В последнее время интенсивно развиваются реконфигурируемые вычислительные системы (РВС), основным аппаратным ресурсом которых является поле связанных программируемых логических интегральных схем (ПЛИС). Рост производительности кристаллов ПЛИС, опережающий рост традиционных микропроцессоров, и архитектурные особенности кристаллов ПЛИС позволяют на целых классах задач существенно превосходить по реальной производительности кластерные системы, при более низкой частоте.

Несмотря на постоянный рост вычислительной ёмкости кристаллов ПЛИС, существуют задачи, для эффективной реализации которых требуется сразу множество взаимосвязанных ПЛИС. При этом возникает проблема распределения фрагментов виртуального вычислительного устройства, решающего задачу, по кристаллам ПЛИС. Следует отметить, что с каждым новым поколением ПЛИС количество выводов кристалла ПЛИС не изменяется, а количество выводов, доступных пользователю для информационных обменов, ещё и сокращается.

Существующие средства программирования ПЛИС направлены, главным образом, на реализацию решений в рамках одной ПЛИС или в ряде независимых ПЛИС.

В НИИ МВС ЮФУ в последние десять лет разрабатываются и развиваются инструментальные средства программирования, которые обеспечивают быструю разработку эффективных параллельных программ для РВС на языке высокого уровня [1]. При решении прикладной задачи данными средствами выполняется автоматическое отображение виртуального вычислительного устройства, соответствующего параллельной программе, на аппаратный ресурс РВС. Одной из основных задач, решаемых при отображении виртуального вычислительного устройства на аппаратный ресурс РВС, является задача разбиения информационного графа, описывающего данное устройство, на непересекаю-

* Работа выполнена при финансовой поддержке стипендии Президента Российской Федерации молодым ученым и аспирантам, осуществляющим перспективные научные исследования и разработки по приоритетным направлениям модернизации российской экономики, на 2016-2018 гг. (СП-173.2016.5).

щиеся подграфы. При этом накладывается два ограничения на формируемые подграфы:

- предполагаемый занимаемый аппаратный ресурс фрагмента вычислительного устройства, описанного подграфом, не должен превышать аппаратного ресурса одной ПЛИС;
- количество предполагаемых трассируемых внешних связей фрагмента вычислительного устройства, описанного подграфом, не должно превышать количество пользовательских выводов ПЛИС, в которую данный фрагмент будет размещён.

Современные кристаллы ПЛИС, используемые для построения РВС, имеют большую вычислительную ёмкость, что позволяет в них размещать сотни схемотехнических блоков, выполняющих арифметические операции с плавающей запятой, тысячи схемотехнических блоков, выполняющих арифметические операции с фиксированной запятой, или десятки тысяч схемотехнических блоков, выполняющих логические операции. Поэтому виртуальные вычислительные устройства, разрабатываемые для решения прикладных задач на многокристальных РВС, могут содержать несколько сотен тысяч различных вычислительных блоков. Следовательно, и информационный граф, описывающий вычислительную структуру параллельной программы, может содержать такое количество вершин.

Подзадача разбиения информационного графа является самой трудоёмкой при решении задачи отображения параллельной программы на аппаратный ресурс РВС и от её решения зависит успешность решений подзадач размещения и трассировки. Для разбиения графов больших размерностей на практике применяют различные эвристические методы, основанные на методе иерархичной декомпозиции графов [2]. В силу того, что ни одна эвристика не может гарантировать хорошие варианты разбиения для всех информационных графов задач, целесообразно использовать параллельно сразу несколько методов разбиения и, возможно, с различными критериями.

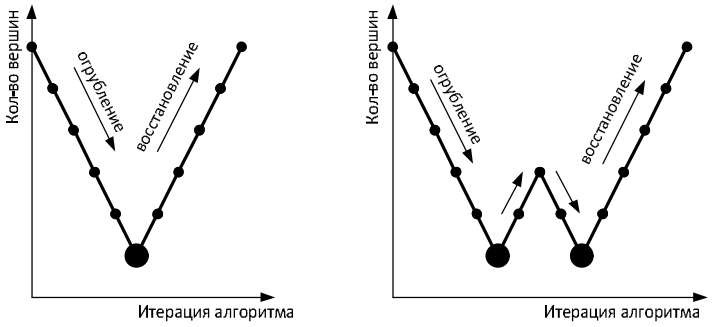
Из-за существующих ограничений для решения задачи разбиения информационных графов ряд эффективных методов разбиения (итерационные алгоритмы, биоинспирированные алгоритмы) не может быть использован. Другие же методы требуют существенной доработки (многоуровневая схема разбиения, рекурсивная бисекция, последовательные алгоритмы).

Для решения задачи разбиения информационных графов параллельных программ были модифицированы и разработаны методы и алгоритмы, основанные на методах многоуровневой схемы разбиения и рекурсивной бисекции [3]. Тем не менее из-за роста числа вершин в информационных графах и расширения областей применения РВС появи-

лась необходимость в разработке новых методов разбиения информационных графов параллельных программ для РВС. В данной работе рассматривается новый метод декомпозиции графа, основанный на W-образной модели иерархического разбиения графа.

Классическая (или V-образная) многоуровневая схема разбиения графа [2] (см. рис. 1,а) состоит из трёх этапов:

- итерационного «огрубление графа» (снижение числа вершин в графе) за счёт стягивания вершин, которые выгодно размещать вместе;
- разбиения огрублённого графа на подграфы;
- восстановления огрублённого графа и оптимизации полученного результата разбиения.



а *б*

Рис. 1. Многоуровневая схема разбиения графа V-образная (а) и W-образная (б)

Рост вычислительных ресурсов современных кристаллов ПЛИС усложняет процесс разбиения графа. На основании данных обстоятельств за основу нового разрабатываемого алгоритма разбиения графа взята более усложнённая модель иерархичной декомпозиции графов – W-образная модель (см. рис.1,б), которая позволяет в ряде случаев обходить локальные оптимумы в случае получения результатов разбиения, не удовлетворяющих ограничениям [4].

Метод иерархической декомпозиции графов согласно W-образной модели заключается в следующем: если в процессе огрубления получен некоторый результат, который не в полной мере соответствует заданным критериям, то в данном результате «плохие» подграфы разбиваются, и процесс огрубления может быть выполнен заново, в том числе и с другим критерием выбора пар стягиваемых вершин.

Под «плохими» подграфами понимаются такие, для которых количество внешних рёбер больше ограничения, задаваемого количеством

пользовательских выводов ПЛИС, или соотношение количества внешних рёбер к количеству внутренних рёбер больше некоторого параметра.

В разрабатываемом новом методе иерархической декомпозиции графа предлагается поочерёдно использовать три различных критерия выбора пар стягиваемых вершин:

- выбираются смежные вершины, которые в результате стягивания образуют вершину с максимальным отношением веса стянутой вершины (суммарный аппаратный ресурс ПЛИС, занимаемый схемотехнически блоками, соответствующими стянутым вершинам) к внешним связям стянутой вершины (критерий, используемый в ранее разработанном алгоритме стягивания вершин в информационных графах параллельных программ для РВС [3]);

- выбираются смежные вершины, которые в результате стягивания дают наибольшее снижение числа внешних связей в графе (критерий, используемый в алгоритме стягивания в классической многоуровневой схеме [2]);

- выбираются смежные вершины, которые в результате стягивания образуют вершину с максимальным отношением произведения суммарного числа вершин подграфа, образованного стянутой вершиной, на количество внешних связей стянутой вершины к числу удалённых рёбер исходного графа, входящие в стянутую вершину (новый критерий, позволяющий находить пары стягиваемых вершин с хорошим показателем числа удаляемых рёбер в стянутом графе по отношению к оставшемуся числу рёбер).

Предлагаемый новый метод иерархической декомпозиции графа на основе W-образной модели позволит в ряде случаев обходить локальные тупики, на которых останавливалась работа алгоритма, на основе ранее разработанного метода, а также разнообразит получаемые результаты декомпозиции графов, что, в свою очередь, может увеличить качество решения задачи отображения информационных графов на аппаратный ресурс РВС в целом.

1. Multi-level Programming of FPGA-based Computer Systems with Reconfigurable Macroobject Architecture, IFAC Proceedings Volumes, Programmable Devices and Embedded Systems. – Vol. №12. – Part №1. – 2013. – P. 204 – 209.
2. *B. Hendricson and R. Leland.* Multidemensional spectral load balancing. Sandia National Laboratories, Albuquerque, NM, 1993.
3. *Гуленок А.А.* Методы и алгоритмы отображения графов задач на реконфигурируемые вычислительные системы // Вестник компьютерных и информационных технологий. – М.: Машиностроение, 2011. – №6. – С. 3 – 11.
4. *Aydin B.* Recent Advances in Graph Partitioning. Technial Report November 2013, arXiv:1311.3144.

А.И. Долгов

О РАСШИРЕНИИ ВОЗМОЖНОСТЕЙ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ПРОГРАММНОЙ РЕАЛИЗАЦИИ МОДИФИКАЦИЙ ФОРМУЛЫ БАЙЕСА

АО «КБ «Связь», г. Ростов-на-Дону,
dolgov-ai@yandex.ru

Введение

Судя по современной научно-технической литературе, представляет теоретический и практический интерес разработка модификаций формулы Байеса при различном количестве рассматриваемых свидетельств (см., например, [1. С. 284-285] и [2. С. 235-236]).

В настоящее время известные модификации формулы Байеса ограничиваются с учётом лишь мультипликативно[1] накопленных свидетельств в виде произведения учитываемых событий, что не обеспечивает возможность решения ряда актуальных прикладных задач.

Одним из важнейших показателей математического и программного обеспечения является логическая корректность процесса обработки данных. Безотказность программного обеспечения определяется его корректностью (правильностью) и, следовательно, целиком зависит от наличия в нем ошибок, внесенных на этапах создания программы.

Следует обратить внимание на то, что в известных публикациях (кроме опубликованных автором) формулы, предназначенные для учёта неизменяемого количества мультипликативно накопленных свидетельств, используются и в случае изменяемого количества накапливаемых свидетельств, что ввиду некорректного нормирования (нормирующий делитель оказывается не равным сумме всех нормируемых вероятностей) приводит к неправильным результатам. Например, доказано, что при сравнительной оценке нормированных значений вероятностей комбинированных событий с изменяемым количеством мультипликативно накапливаемых условно независимых свидетельств (см. [2. С. 235-236]) в результате некорректного нормирования не соблюдается мультипликативный эффект – нормированное произведение вероятностей не уменьшается, а в ряде случаев увеличивается, в отличие от результатов, получаемых при корректном нормировании [4. С. 125-126].

Учёт мультипликативно накопленных свидетельств не обеспечивает решение существенного множества задач, например, связанных с

[1] Здесь применяется довольно широко используемое понятие мультипликативности, которая отличается от аддитивности тем, что рассматриваемые эффекты не суммируются, а перемножаются).

анализом нарастания опасностей и угроз, при решении которых возникает необходимость оценивания аддитивно накопленных (в виде суммы) свидетельств.

Данная публикация посвящена решению задачи построения модификации формулы Байеса, не нашедшей отражения в имеющейся литературе. Предлагается новая ранее не применявшаяся корректная модификация формулы, предназначенная для определения апостериорных условных вероятностей гипотез при аддитивно накопленных свидетельствах в виде суммы. Применение формулы расширяет состав инструментальных средств[2] математического обеспечения программной реализации прикладных задач на суперкомпьютерах и других компьютерных средствах, основанных на более широком развитии идей, разработанных Байесом.

Исходные положения

При дальнейшем изложении будут употребляться понятия:

- ***элементарное событие*** – событие, неразделимое на элементы;
- ***комбинированное событие*** – событие, представляющее то или иное сочетание элементарных событий.

Далее в качестве элементарных событий будут рассматриваться подтверждаемые гипотезы и свидетельства, а в качестве комбинированных событий только произведения элементарных событий.

Согласно теореме умножения вероятностей ([3. С. 46]), вероятность $P(H_k E)$ произведения элементарных событий H_k и E вычисляется в виде произведения вероятностей $P(H_k E) = P(E)\,P(H_k \mid E)$, в связи с чем формула Байеса часто записывается в виде $P(H_k \mid E) = \dfrac{P(H_k E)}{P(E)}$, описывающем определение апостериорных условных вероятностей $P(H_k \mid E)$ гипотез H_k ($k = 1, ..., n$) на основе нормирования априорных вероятностей $P(H_k E)$ учитываемых комбинированных несовместных событий $H_k E$. Все рассматриваемые сравниваемые по степени возможности комбинированные события $H_k E$ ($k = 1, ..., n$) образуют полную группу $\bigcup_{k=1}^{n} H_k E$ несовместных событий, в связи с чем, их веро-

[2] В состав инструментальных средств математического обеспечения входят математические методы, модели и алгоритмы обработки информации, используемые при решении функциональных и проектных задач в информационной системе

ятности $P(H_k E)$ нормируются с учётом формулы полной вероятности $P(E) = \sum_{k=1}^{n} P(H_k) P(E \mid H_k)$. Поэтому формула Байеса довольно часто записывается в наиболее употребляемом виде:

$$P(H_k \mid E) = \frac{P(H_k) P(E \mid H_k)}{\sum_{k=1}^{n} P(H_k) P(E \mid H_k)}, \quad (k = 1, ..., n). \qquad (1)$$

Необходимо отметить важное обстоятельство, заключающееся в том, что при построении формулы Байеса и её модификаций полная группа сравниваемых по вероятности комбинированных событий выбирается субъективно. В частности, при построении формулы Байеса учитывается вполне определённое количество гипотез, обусловленное конкретно решаемой прикладной задачей, а при построении модификаций формулы Байеса может учитываться то или иное количество как мультипликативно, так и аддитивно учитываемых свидетельств.

В данной публикации излагаются результаты построения модификации формулы Байеса, учитывающей неизменяемое, вполне определённое количество аддитивно накопленных свидетельств, общепринято считаемых условно независимыми. Такое ограничение обусловлено тем, что построение формулы, учитывающей изменяемое количество накапливаемых свидетельств, требует отдельного рассмотрения.

Разработка модификации формулы Байеса

Ввиду актуальности решения задач оценки аддитивно накопленных свидетельств, построим по аналогии с формулой Байеса её модификацию, предназначенную для определения апостериорных условных вероятностей $P(H_k \mid \sum_{i=1}^{m} E_i)$ гипотез H_k $(k = 1, ..., n)$ при сумме неизменяемого количества m подтвержденных свидетельств.

В таком случае условные вероятности $P(H_k \mid \sum_{i=1}^{m} E_i)$ должны определяться на основе нормирования априорных вероятностей сравниваемых по степени возможности нормируемых несовместных комбинированных событий выбираемой полной группы $\bigcup_{k=1}^{n} H_k \sum_{i=1}^{m} E_i$. При этом в числителе разрабатываемой модификации, как и в формуле Байеса, должна быть указана вероятность одного из несовместных комбинированных событий $P(H_k \sum_{i=1}^{m} E_i)$ упомянутой полной группы, а в знаменателе – сумма всех нормируемых вероятностей.

Таким образом, разрабатываемая формула может быть записана в следующем виде:

$$P(H_k \mid \sum_{i=1}^{m} E_i) = \frac{P(H_k \sum_{i=1}^{m} E_i)}{\sum_{k=1}^{n} P(H_k \sum_{i=1}^{m} E_i)}, \quad (k = 1,...,n;\ i = 1,...,m),$$

а с учётом того, что согласно теореме умножения вероятностей,

$$P(H_k \sum_{i=1}^{m} E_i) = P(H_k) P(\sum_{i=1}^{m} E_i \mid H_k),$$

искомая формула принимает вид:

$$P(H_k \mid \sum_{i=1}^{\partial} Å_i) = \frac{P(H_k) P(\sum_{i=1}^{\partial} Å_i \mid H_k)}{\sum_{k=1}^{n} P(H_k) P(\sum_{i=1}^{\partial} Å_i \mid H_k)}. \qquad (2)$$

Окончательный вид формула приобретёт в случае перехода от априорных условных вероятностей суммы к сумме вероятностей.

Методом полной индукции можно доказать общую формулу для условной вероятности суммы любого числа m совместных свидетельств при гипотезе H_k, аналогичную общеизвестной формуле для вероятности суммы любого числа совместных событий [3. С. 44]:

$$P(\sum_{i=1}^{\partial} Å_i \mid H_k) = \sum_{i} P(Å_i \mid H_k) - \sum_{i,j} P(E_i E_j \mid H_k) + \sum_{i,j,l} P(E_i E_j E_1 \mid H_k) - ...$$

$$... + (-1)^{m-1} P(E_1 E_2 \cdot ... \cdot E_m \mid H_k). \qquad (3)$$

Подставив в числитель формулы (2) нормируемое выражение (3) и указав в знаменателе сумму всевозможных нормируемых выражений, получаем соотношение для определения апостериорных условных вероятностей каждой из несовместных гипотез в случае интегральной оценки m аддитивно накапливаемых совместных свидетельств:

$$P(H_k \mid \sum_{i=1}^{\partial} E_i) = \frac{P(H_k) P(\sum_{i=1}^{\partial} E_i \mid H_k)}{\sum_{k=1}^{n} P(H_k) P(\sum_{i=1}^{\partial} E_i \mid H_k)} =$$

$$= \frac{P(H_k)[(\sum_{i=1}^{m} P(E_i \mid H_k)) + SE(k,m)]}{\sum_{k=1}^{n} P(H_k)[(\sum_{i=1}^{m} P(E_i \mid H_k)) + SE(k,m)]},$$

где, согласно (3), $\sum_{i=1}^{\partial} P(E_i \mid H_k)$ в числителе и в знаменателе суммируется с выражением

$$\mathrm{SE}(k,m) = -\sum_{i,j} \mathrm{P}(\mathrm{E}_i\ \mathrm{E}_j \,\big|\, \mathrm{H}_k) +$$

$$+\sum_{i,j,\mathrm{l}} \mathrm{P}(\mathrm{E}_i\ \mathrm{E}_j\ \mathrm{E}_1 \,\big|\, \mathrm{H}_k) \ldots + (-1)^{m-1} \mathrm{P}(\mathrm{E}_1\ \mathrm{E}_2 \times \ldots \times \mathrm{E}_m \,\big|\, \mathrm{H}_k),$$

а с учётом общепринятого допущения о независимости свидетельств $\mathrm{SE}(k,m)$ выражается через априорные условные вероятности:

$$\mathrm{SE}(k,m) = -\sum_{i,j} \mathrm{P}(\mathrm{E}_i \,\big|\, \mathrm{H}_k)\, \mathrm{P}(\mathrm{E}_j \,\big|\, \mathrm{H}_k) +$$

$$+\sum_{i,j,\mathrm{l}} \mathrm{P}(\mathrm{E}_i \,\big|\, \mathrm{H}_k)\, \mathrm{P}(\mathrm{E}_j \,\big|\, \mathrm{H}_k)\, \mathrm{P}(\mathrm{E}_1 \,\big|\, \mathrm{H}_k) + \ldots$$

$$\ldots + (-1)^{m-1} \mathrm{P}(\mathrm{E}_1 \,\big|\, \mathrm{H}_k)\, \mathrm{P}(\mathrm{E}_2 \,\big|\, \mathrm{H}_k) \ldots \mathrm{P}(\mathrm{E}_m \,\big|\, \mathrm{H}_k).$$

Заключение

Применение нетрадиционной разновидности модификации формулы Байеса позволит расширить возможности математического обеспечения программной реализации методов определения апостериорных условных вероятностей гипотез при неизменяемом количестве аддитивно накопленных свидетельств, а также обеспечить применение ранее не использовавшейся в суперкомпьютерах и других компьютерных средствах корректной формулы для решения новых прикладных задач.

1. *Муромцев Д.Ю., Муромцев Ю.Л., Тютюнник В.М., Белоусов О.А.* Экономическая эффективность и конкурентоспособность: учебное пособие / Д.Ю. Муромцев. – Тамбов: Изд-во Тамб. гос. техн. университета, 2007.– 96с.
2. *Нейлор К.* Как построить свою экспертную систему / пер. с англ. К. Нейлор. − М.: Энергоиздат, 1991. − 286 с.
3. *Вентцель Е.С.* Теория вероятностей: учебник для вузов. – 10-е изд., стереотип. / Е.С. Вентцель. – М.: Высшая школа, 2006. – 575 с.
4. *Долгов А.И.* Корректные модификации формулы Байеса для параллельного программирования // Материалы 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии». – Ростов-на-Дону: Изд-во ЮФУ. – 2014. – Т.1. – С. 122 – 126.

А.И. Дордопуло[1], И.И. Левин[2], И.А. Каляев[1],
В.А. Гудков[1], А.А. Гуленок[1]

РЕСУРСОНЕЗАВИСИМОЕ ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ГИБРИДНОГО ТИПА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ COLAMO*

[1] НИИ многопроцессорных вычислительных систем ЮФУ, г. Таганрог,
[2] ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог,
scorpio@mvs.sfedu.ru

Большинство реальных вычислительных задач, решаемых на суперкомпьютерах, содержат связанные между собой последовательные и параллельные фрагменты вычислений [1], для эффективной реализации которых необходимы как последовательные устройства (микропроцессоры), так и параллельные устройства (например, реализуемые в программируемых логических интегральных схемах, ПЛИС) в составе единой вычислительной системы гибридного типа (ВСГТ). Существующие технологии, применяемые для программирования ВСГТ (CUDA, OpenACC, OpenCL и др.) разделяют прикладную задачу на последовательные и параллельные фрагменты, каждый из которых реализуется на отдельном узле ВСГТ соответствующей архитектуры. Такая организация вычислений плохо масштабируется при изменении конфигурации ВСГТ, что существенно усложняет перенос готовых программ для различных ВСГТ. Поэтому для эффективного программирования ВСГТ необходима технология ресурсонезависимого программирования, которая обеспечит автоматическую адаптацию текста параллельной программы под изменившуюся конфигурацию ВСГТ как при увеличении, так и при сокращении доступного аппаратного ресурса.

Для описания параллельных программ в технологии ресурсонезависимого программирования ВСГТ можно использовать язык программирования высокого уровня с неявным описанием параллелизма COLAMO [2], позволяющий описывать фрагменты с различными типами организации вычислений. Автоматическое преобразование текста программы под заданную конфигурацию ВСГТ без корректировки исходного текста пользователем выполняет специальная программа – препроцессор языка COLAMO.

Для автоматического преобразования текста программы для ВСГТ необходимым условием является представление параллельной про-

* Работа выполнена при частичной финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии № 14.578.21.0006 от 05.06.2014, уникальный идентификатор RFMEFI57814X0006

граммы в канонической (единой параллельно-конвейерной) форме для быстрого перехода от структурной реализации вычислений к процедурной реализации и наоборот. Модуль анализа исходной параллельной программы препроцессора языка COLAMO преобразует параллельную программу в каноническую форму, после чего подсчитывает необходимый для ее реализации аппаратный ресурс и сопоставляет его с текущей конфигурацией ВСГТ для определения максимальной степени и типов необходимых преобразований. Если текущего аппаратного ресурса ВСГТ достаточно для выполнения программы, то синтезируются загрузочные модули для задействованных узлов ВСГТ, в противном случае выполняются специальные редукционные преобразования, сбалансированно сокращающие производительность программы и задействованный аппаратный ресурс ВСГТ. Редукция производительности выполняется в следующем порядке: редукция по одновременно выполняемым подграфам программы, редукция по разрядности, редукция по командам (устройствам), редукция по скважности (частоте). Сокращение занимаемого аппаратного ресурса для каждого вида редукции с учетом ее теоретически допустимой степени выполняет модуль анализа препроцессора, который определяет наиболее рациональный вариант использования редукции для обеспечения максимально возможной производительности программы для текущей конфигурации ВСГТ. Полученный в автоматизированном режиме после препроцессора текст редуцированной параллельной программы передается транслятору языка программирования COLAMO, который создает развернутый информационный граф прикладной задачи. Информационный граф прикладной задачи, содержащий фрагменты, реализуемые структурно на реконфигурируемых вычислительных узлах и процедурно на микропроцессорных вычислительных узлах, передается программе-синтезатору для автоматического распределения фрагментов задачи по доступным в текущей конфигурации ВСГТ реконфигурируемым и микропроцессорным вычислительным узлам. Согласование потоков данных между различными по типам организации вычислений узлам ВСГТ осуществляется на основе файла описания конфигурации ВСГТ, содержащего данные о типах синхронизируемых вычислительных узлов, разрядности шины данных, частоте их работы, скважности подачи данных и др. После установки необходимых интерфейсов и элементов синхронизации программа-синтезатор создает загрузочные конфигурационные файлы *.bit для реконфигурируемых вычислительных узлов и загрузочные файлы *.exe для микропроцессорных узлов ВСГТ и единую для всех вычислительных модулей ВСГТ управляющую вычислительным процессом программу.

1. *Kalyaev I.A., Levin I.I., Semernikov E.A., Shmoilov V.I.* Rekonfiguriruyemiye multikonveyerniye vichislitelniye struktury. [Reconfigurable multipipeline computing structures]. – 2nd edition, revised and supplemented / edited by I.A. Kalyaev. – Rostov-on-Don: SSC RAS Publishing, 2009. – 344 p.
2. *Igor A. Kalyaev, Ilya I. Levin, Alexey I. Dordopulo, Liuba M. Slasten.* FPGA-based Reconfigurable Computer Systems. Science and Information Conference (SAI), Oct 7-Oct 9, 2013. London, UK. – P. 148 – 155.

Е.И. Духнич[1], В.В. Подбельский[2]

ПРОЦЕССОРНЫЙ МАССИВ ДЛЯ КВАТЕРНИОННОГО ПРЕОБРАЗОВАНИЯ ЯКОБИ

[1]Государственный морской университет имени адмирала Ф.Ф.Ушакова, г. Новороссийск,
[2]Национальный исследовательский университет «Высшая школа экономики», г. Москва,
evgenydukhnich@gmail.com, vpodbelskiy@hse.ru

Введение

Метод Якоби применяется для вычисления собственных значений и сингулярного разложения матриц при решении многих задач линейной алгебры в физике и обработке сигналов и изображений. Он приводит симметричную действительную или комплексную эрмитову матрицу к диагональной форме с помощью некоторой последовательности плоских вращений в разных плоскостях исходного n-мерного векторного пространства, причем для эрмитовых матриц собственные значения совпадают с сингулярными. Сингулярное разложение используется при решении самых разных задач — от приближения методом наименьших квадратов и решения систем уравнений до сжатия и распознавания изображений. Высокая практическая важность быстрого решения подобного типа задач приводит к необходимости реализации метода Якоби с помощью проблемно-ориентированных массивов процессорных элементов (ПЭ). Известен ряд подобных разработок с использованием специализированных процессоров типа CORDIC, которые с помощью операций 2D-вращения параллельно обрабатывают пары элементов действительной матрицы за время одной операции умножения [1]. Для ускорения декомпозиции комплексных матриц был предложен 4D-CORDIC алгоритм [2]. Развитием этого подхода явилась разработка 8D-CORDIC алгоритма для декомпозиции кватернионных матриц [3]. Этот октонионный CORDIC-алгоритм (OCA) параллельно вычисляет 8 компонент действительного вектора (2 компоненты кватернионного вектора) при выполнении вращения Гивенса за время, сравнимое с длительностью одной операции умножения.

Декомпозиция кватернионных матриц напрямую, вместо соответствующих им действительных или комплексных матриц, в том числе и методом Якоби в последнее время все чаще находит применение в связи с достигаемым при этом сокращении вычислительной сложности и повышении точности вычислений [5]. Идея построения процессорного массива (Brent-Luk-Van Loan (BLV) систолический массив) для параллельной реализации метода Якоби была предложена еще в 1983 г. [6]. Однако вопросы организации работы такого массива с элементами, реализующими ОСА, для кватернионных матриц требуют существенных дополнительных исследований. При рассмотрении мы ограничимся случаем эрмитовых матриц, как имеющих важное самостоятельное применение.

Метод Якоби для кватернионных матриц

Задан блок 2×2 эрмитовой матрицы A $m \times m$:

$$\boldsymbol{A}_{rc} = \begin{bmatrix} \boldsymbol{a}_{rr} & \boldsymbol{a}_{rc} \\ \boldsymbol{a}_{cr} & \boldsymbol{a}_{cc} \end{bmatrix}, \quad (1)$$

где кватернионы $\boldsymbol{a}_{cr} = \overline{\boldsymbol{a}}_{rc}$ и вещественные диагональные элементы. Черта над переменной означает комплексное или кватернионное сопряжение.

Используя 2 × 2 унитарную кватернионную матрицу $\boldsymbol{P}$, преобразование Якоби для матрицы $\boldsymbol{A}_{rc}$ можно записать в виде

$$\boldsymbol{P}^{*}\boldsymbol{A}_{rc}\boldsymbol{P} = \boldsymbol{R}_{rc}, \quad (2)$$

где знак * обозначает комплексное сопряжение и $\boldsymbol{R}_{rc}$ – 2×2 кватернионная диагональная матрица с вещественными элементами:

$$\boldsymbol{R}_{rc} = \begin{bmatrix} r_{rr} & 0 \\ 0 & r_{cc} \end{bmatrix}. \quad (3)$$

В общем случае 2×2 унитарная кватернионная матрица $\boldsymbol{P}$ имеет элементы, модули которых равны синусам и косинусам некоторого угла θ:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{c} & \boldsymbol{s} \\ -\overline{\boldsymbol{s}} & \overline{\boldsymbol{c}} \end{bmatrix}, \quad (4)$$

где $|\boldsymbol{c}| = \boldsymbol{cos}\ \boldsymbol{\theta}, |\boldsymbol{s}| = \boldsymbol{\sin\theta}$ и $\boldsymbol{c}\overline{\boldsymbol{c}} + \boldsymbol{s}\overline{\boldsymbol{s}}$=1.

Двустороннее вращение (2) может быть выполнено как последовательность из двух 8-D поворотов на угол θ, который необходимо подсчитать заранее. Этот угол может быть представлен в неявном виде [2] как сумма постоянных углов элементарных поворотов со знаками $\{\alpha_i \dots \rho_i\}_{\theta}, (i = \overline{0,n})$. Угол $\theta = arctg\, y/x$ может быть определен по алгоритму ОСА [4] в векторном режиме для кватернионного вектора X=[x, y]^{T} как последовательность наборов упомянутых знаков. Если принять x=a_{cc}-a_{rr} и y=a_{cr}+$\overline{\boldsymbol{a}_{rc}}$, то последовательность наборов $\{\alpha_i \dots \rho_i\}_{2\theta}, (i = \overline{0,n})$ будет соответствовать углу 2θ.

Процессорный массив

Массив реализует итерационный алгоритм JA:

```
1     Algorithm JA (A(0)=A)
2     while  S1(s) / S2(s) > ε {
3          for (all r, c) -- для всех пар индексов r, c
,4                  { A(s+1) = P(s)* A(s) P (s)
5          } –end for
6     } –end while,
```

где S_1 – сумма модулей внедиагональных элементов матрицы $\boldsymbol{A}^{(s)}$ на приближении s; S_2 – сумма диагональных элементов матрицы $\boldsymbol{A}^{(s)}$ на приближении s. Предполагается, что пары индексов (r, c) выбираются циклически по строкам.

Параллельная реализация сингулярного разложения $m \times m$ матрицы основывается на квадратном процессорном массиве BLV с $m/2$ процессорных элементов по каждой стороне, а каждый ПЭ обрабатывает 2×2 подматрицу [6]. Принимая во внимание равенство $a_{cr} = \bar{a}_{rc}$, для эрмитовых матриц можно использовать треугольный массив с одним диагональным ПЭ и $(m-2)$ внедиагональных элементов на горизонтальной стороне и одним диагональным РЕ и $(m/2-1)$ внедиагональных элементов на вертикальной стороне. Конфигурация массива при сведении к нулю элемента a_{12} показана на рис. 1, как пример. Пунктиром показаны линии связи ПЭ для передачи знаков $\{\alpha_i \ldots \rho_i\}_\theta$ углов элементарных поворотов.

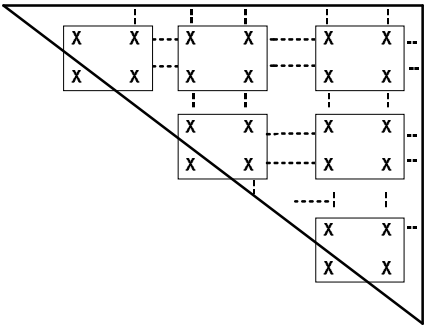


Рис. 1. Треугольный процессорный массив для реализации метода Якоби

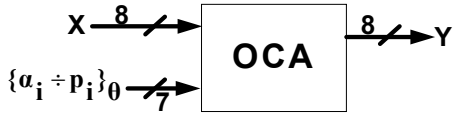


Рис. 2. OCA-процессор в режиме вращения

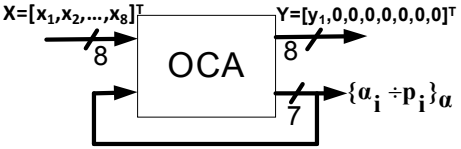


Рис. 3. OCA-процессор в векторном режиме

На рис. 2 и рис. 3 показан октонионный CORDIC-процессор (OCA) [3] в режимах вращения и векторном режиме. Каждый диагональный ПЭ выполняет преобразование Якоби для 2×2 матрицы с кватернионными компонентами. Он состоит из двух модулей OCA (OCA_1 и OCA_3) в векторном режиме (см. рис 3) и одного модуля для вращения вектора $X= [1, 0, 0, 0, 0, 0, 0, 0]^T$, чтобы получить две кватернионные компоненты в результирующем векторе $Y= [c, s]^T$.

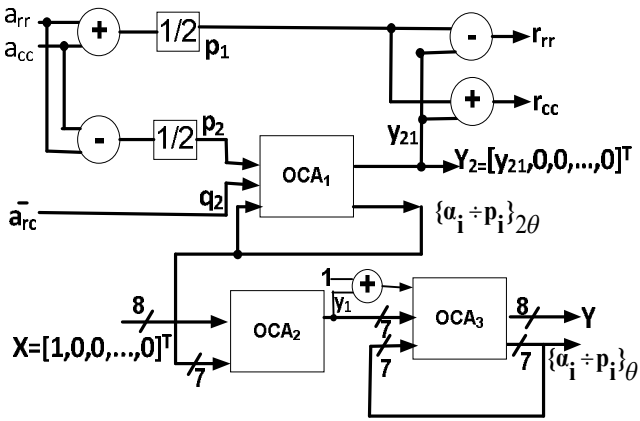


Рис. 4. Диагональный элемент процессорного массива

Этот результат используется, чтобы вырабатывать наборы знаков $\{\alpha_i \ldots \rho_i\}_\theta, (i = \overline{0,n})$ для передачи их внедиагональным элементам данного ряда на каждой итерации. Внедиагональные элементы выполняют

вращение 2-D векторов, как это показано на рис. 2. Для обнуления каждого внедиагонального элемента должна быть произведена перекоммутация соединений процессорных элементов в массиве таким образом, чтобы поместить в диагональный процессор элементы матрицы A_{rc} для обнуления. После окончания разложения диагональные элементы результирующей матрицы будут только вещественными [5].

Результаты моделирования работы процессорного массива для 1000 произвольных эрмитовых кватернионных матриц 15×15 по сравнению с [3] приведены в таблице [7]. Правый столбец показывает ухудшение результатов, если предварительно заменять кватернионную матрицу комплексной.

Таблица

Результаты моделирования работы процессорного массива для 1000 произвольных эрмитовых кватернионных матриц 15×15

Параметр	**Массив ОСА- ПЭ**	**Результаты программной реализации [5]**	**Результаты реализации через комплексную матрицу [3]**
Кол-во проходов	6.1	6.4	6.6
Кол-во вращений	658	672	2869
Относит. погрешность	1e-16	1.88e-15	2.37e-11

1. *P. Meher, J. Valls, T.-B., Juang, K. Sridharan, K. Maharatna.* 50 Years of CORDIC: Algorithms, Architectures and Applications // IEEE Trans. on Circuits and Systems I: Regular Papers, – Vol. 56. – No. 9. – P. 893 – 1907, Sept. 2009.
2. *S.-F.Hsiao, J.-M. Delosme.* Parallel Processing of Complex Data Using Quaternion and Pseudo-Quaternion CORDIC Algorithms // In Proceedings of the ASAP'94 Conf., University of California, – 1994. – P. 125 – 130.
3. *E. Doukhnitch, E. Ozen.* Hardware-oriented Algorithm for Quaternion Valued Matrix Decomposition // IEEE Transactions on Circuits and Systems--II: Express Briefs, 2011. – Vol. 58. – No. 4. – P. 225 – 229.
4. *Духнич Е.И.* Октонионные дискретные линейные преобразования // Суперкомпьютерные технологии: Материалы 3-й Всеросс. науч.-техн. конф. – Ростов-на-Дону: Изд-во ЮФУ, 2014. – Т.1. – С.127 – 132.
5. *N. Le Bihan and S. J. Sangwine.* Jacobi method for quaternion matrix singular value decomposition // Applied Mathematics and Computation, 187. – 2007. – P. 1265 – 1271.
6. *Brent, R. P., F.T. Luk, and C. Van Loan.* Computation of the Singular Value Decomposition Using Mesh-connected Processors // Technical report, Cornell University, Ithaca, NY, USA, 1983.
7. *E. Doukhnitch, V. Podbelskiy.* Higher Parallel Hardware-oriented Algorithm for Jacobi SVD of Quaternion Valued Matrix // Parallel and Cloud Computing Research. – 2013. – Vol. 1. – No.3. – P.41 – 49.

О.В. Ершова, Е.В. Кириченко, Е.А. Семерников, А.В. Чкан

ОШИБКИ ВЫЧИСЛЕНИЯ СПЕКТРА ПРИ УСЕЧЕНИИ РЕЗУЛЬТАТОВ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ В АЛГОРИТМАХ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ С МАСШТАБИРОВАНИЕМ ДАННЫХ

ООО "НИЦ супер-ЭВМ и нейрокомпьютеров", г. Таганрог,
ershova150681@mail.ru, e.v.kirichenko@yandex.ru,
semernikov@mvs.tsure.ru, chkan_andrey@mail.ru

Как известно, дискретное преобразование Фурье, широко применяемое в цифровой обработке сигналов, на практике реализуется с помощью алгоритма быстрого преобразования Фурье (БПФ). При обработке чисел в формате с фиксированной точкой эффекты ограничения разрядной сетки и масштабирования оказывают существенное влияние на ошибки результата вычислений [1, 2]. Наиболее полно вопрос об ошибках вычисления БПФ при наличии масштабирования поднимался в [1], где приведены нижняя и верхняя границы уровня ошибок. Однако приводимая в [1] верхняя граница среднего уровня ошибок имеет другой наклон, чем в эксперименте, а нижняя граница, как сказано там же, малопригодна для практики, так как является заниженной.

В данной статье рассмотрено влияние ошибок, обусловленных ограничением разрядности данных при выполнении умножения на поворачивающие множители и наличием масштабирования в алгоритмах БПФ с прореживанием по частоте и по времени.

В БПФ с масштабированием возникают ошибки двух видов: ошибки усечения произведения данных на поворачивающие множители и ошибки масштабирования. Эти ошибки обладают разными статистическими характеристиками.

Согласно [1], ошибки усечения при выполнении операции нетривиального умножения целочисленного числа на вещественный множитель имеют равномерный закон распределения, а математическое ожидание $\bar{e}_m$ и дисперсия σ_m^2 ошибок равны:

$$\bar{e}_m = -0{,}5 \cdot Q \cdot (1 + j), \tag{1}$$

$$\sigma_m^2 = 2 \cdot Q^2 / 12. \tag{2}$$

Здесь и далее $Q = 2^{-(b-1)}$, где b – количество разрядов формата представления данных; j – мнимая единица.

Ошибки усечения при выполнении операции масштабирования (делении на 2 с усечением результата) имеют дискретный закон распре-

деления $\{0,-0{,}5\cdot(1+j)\}$ с математическим ожиданием (МО) $\overline{e}_s$ и дисперсий σ_s^2, равными:

$$\overline{e}_s = -0{,}25\cdot Q\cdot(1+j), \tag{3}$$

$$\sigma_s^2 = 2\cdot Q^2/16. \tag{4}$$

В формулах (1) ÷ (4) учтен тот факт, что данные в БПФ являются комплексными величинами.

Как известно, при наличии масштабирования алгоритмы базовых операций (БО) на *i*-м этапе вычисления БПФ с прореживанием по частоте и по времени имеют вид:

$$\dot{A}_{i+1} = \left(\dot{A}_i + \dot{B}_i\right)/2; \tag{5}$$

$$\dot{B}_{i+1} = \left[(\dot{A}_i - \dot{B}_i)\cdot(\dot{W}_N)^l\right]/2; \tag{6}$$

и

$$\dot{A}_{i+1} = \left[\dot{A}_i + \dot{B}_i\cdot(\dot{W}_N)^l\right]/2; \tag{7}$$

$$\dot{B}_{i+1} = \left[\dot{A}_i - \dot{B}_i\cdot(\dot{W}_N)^l\right]/2 \tag{8}$$

соответственно.

Здесь: $(\dot{W}_N)^l$ – комплексный поворачивающий множитель БО; $\dot{A}_i$, $\dot{B}_i$ – входные данные БО; $\dot{A}_{i+1}$, $\dot{B}_{i+1}$ – выходные данные БО.

На основании анализа информационных графов БПФ с прореживанием по частоте и по времени можно сделать ряд выводов для анализа ошибок вычислений.

Из анализа формул (5), (6) следует, что в БПФ с прореживанием по частоте на *i*-м этапе вычисления возникают ошибки на выходе базовой операции:

- с параметрами (3), (4) – в ветви *A* базовой операции, а также в ветви *B*, если умножение на $(\dot{W}_N)^l$ является тривиальным,
- с параметрами (1), (2) – в ветви *B*, если умножение на $(\dot{W}_N)^l$ не является тривиальным.

Из анализа формул (7), (8) следует, что в БПФ с прореживанием по времени на *i*-м этапе вычисления возникают ошибки на выходе базовой операции:

- с параметрами (3), (4) – в ветвях *A* и *B*, если умножение на $(\dot{W}_N)^l$ является тривиальным;
- с параметрами (1), (2) – во входной ветви *B*, если умножение на $(\dot{W}_N)^l$ не является тривиальным.

Помимо этого, через БО распространяются ошибки арифметических вычислений, возникшие на предыдущих этапах.

Мощность ошибок усечения произведения и масштабирования $\overline{x^2(k)}$ для k-го спектрального отсчета складывается из суммы квадрата модуля математического ожидания $|\bar{x}(k)|^2$ и дисперсии $\sigma^2(k)$ ошибок усечения.

Для анализа ошибок перечислим основные свойства графов алгоритмов вычисления БПФ с прореживанием по частоте и по времени [3,4]. При этом воспользуемся двоичным представлением номера k-го спектрального отсчета:

$$k = 2^{M-1}q_0 + 2^{M-2}q_1 + ... + 2q_{M-2} + q_{M-1}, \quad (9)$$

а также двоично-инверсным представлением номера:

$$k_{BR} = 2^{M-1}q_{M-1} + 2^{M-2}q_{M-2} + ... + 2q_1 + q_0, \quad (10)$$

где $\{q_i\}$, $i = 0,1,..,M-1$ – значения битов в двоичном представлении номера k спектрального отсчета, принимающие значения 0 или 1.

Свойство 1. На i-м этапе данные k-го спектрального отсчета выходят по выходу A БО, если $q_i = 0$, и по выходу B, если $q_i = 1$.

Свойство 2. Множество из $N/2$ БО на i-м этапе БПФ с прореживанием по частоте можно разбить на 2^i непересекающихся групп по 2^{M-1-i} БО в каждой. Для произвольно выбранного k-го спектрального отсчета данные на i-м этапе проходят через БО одной из групп.

В БПФ с прореживанием по частоте степени поворачивающих множителей в группе различны и изменяется от 0 до $\left(N/2 - 2^i\right)$ с шагом 2^i. Набор степеней поворачивающих множителей одинаков для всех групп.

В БПФ с прореживанием по времени степени l поворачивающих множителей в каждой группе одинаковы, а шаг изменения степеней поворачивающих множителей в группах равен 2^{M-1-i}. Поворачивающий множитель k-го спектрального отсчета на i-м этапе равен $(\dot{W}_N)^l = \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{k}{2^{i+1}} \cdot N\right)$, при этом степень l поворачивающего множителя W_N с учетом периодичности комплексной экспоненты определяется как $l = \mathrm{mod}\left(k \cdot 2^{M-1-i}, N/2\right) = \mathrm{mod}\left(k, 2^i\right) \cdot 2^{M-1-i}$.

Свойство 3. В БПФ с прореживанием по частоте в каждой группе БО присутствует по 2 БО с тривиальными умножениями на ±1 и ±*j*.

В БПФ с прореживанием по времени множество групп БО на каждом этапе (начиная с первого) включает в себя две группы, степени поворачивающих множителей которых равны 0 и $N/4$, что соответствует тривиальным умножениям на ±1 и ±*j*. Данные k-го спектрального отсче-

та на i-м этапе проходят через группу БО с тривиальными умножениями в том случае, если выполняется условие $\mathrm{mod}\left(k, 2^{i-1}\right)=0$.

Свойство 4. В БПФ с прореживанием по частоте степени поворачивающих множителей равны 0 и $N/4$ на предпоследнем и последнем (M-2 и M-1) этапах, а в БПФ с прореживанием по времени – на нулевом и первом этапах. Это соответствует тривиальным умножениям на ±1 и ±j и нулевой ошибке округления произведения.

Свойства 1-4 необходимы для расчета ошибок усечения результатов умножения и масштабирования.

Используя указанные свойства в БПФ с прореживанием по частоте для k-го спектрального отсчета, были получены формулы для общей дисперсии ошибки усечения $\sigma_F^2(k)$, математического ожидания ошибки усечения $\bar{x}(k)$ и среднего квадрата модуля ошибки $\overline{x^2(k)}$:

$$\sigma_F^2(k)=\sum_{i=0}^{M-2}\frac{1}{2^{M-i-1}}\cdot\left[\left(\frac{\sigma_s^2-\sigma_m^2}{2^{M-i-2}}+\sigma_m^2\right)\cdot(q_i==1)+\sigma_s^2\cdot(q_i==0)\right] \quad (11)$$

$$\bar{x}(k)=\sum_{i=0}^{M-2}\left\{(q_i==0)\cdot\left(\left(\sum_{n=i+1}^{M-1}q_n\right)==0\right)\cdot\bar{e}_s+(q_i==1)\cdot(q_{i+1}==0)\times\right.$$

$$\left.\times\left[\bar{e}_m\cdot\left(\left(\sum_{n=i+1}^{M-1}q_n\right)==1\right)+\frac{2\cdot(\bar{e}_s-\bar{e}_m)}{2^{M-i-1}}\right]\right\}+\bar{e}_s \quad (12)$$

$$\overline{x_F^2(k)}=\left|\bar{x}(k)\right|^2+\sigma_F^2(k). \quad (13)$$

Аналогичным образом были получены формулы для дисперсии k-го спектрального отсчета в БПФ с прореживанием по времени:

$$\sigma_T^2(k)=\sigma_s^2\cdot\left(2-\frac{1}{2^{M-1}}\right)+\sum_{i=0}^{M-1}\left\{\frac{\left[\mathrm{mod}\left(k, 2^{i-1}\right)\neq 0\right]\cdot\sigma_m^2}{2^{M-i+1}}\right\}. \quad (14)$$

МО ошибок усечения в БПФ с прореживанием по времени на выходе i-го этапа вычисляется посредством рекуррентной формулы:

$$\overline{x_i(k)}=\overline{x_{i-1}(k)}\cdot\left[1+(-1)^{q_i}\cdot\exp\left(-j\cdot 2\cdot\pi\cdot k/2^{M-i}\right)\right]/2+$$

$$+(-1)^{q_i}\cdot\left[\mathrm{mod}\left(k, 2^{(i-1)}\right)\neq 0\right]\cdot\frac{\bar{e}_m}{2}+\bar{e}_s,\quad i=1,2,...,M-1 \quad (15)$$

$$\overline{x_0(k)}=\bar{e}_s. \quad (16)$$

Средний квадрат модуля ошибок усечения $\overline{x^2(k)}$ вычисляется по формуле

$$\overline{x_T^2(k)}=\left[\bar{x}_{M-1}(k)\right]^2+\sigma_T^2. \quad (17)$$

В формулах (11) - (15) операторы сравнения (●==0), (●==1), (●≠0) равны единице, если условие выполняется, и равны нулю, если условие не выполняется.

Была произведена проверка полученных формул для ошибок усечения произведения и масштабирования в БПФ с прореживанием по частоте и по времени для различных размеров БПФ *N*. Проверка показала хорошее совпадение экспериментальных и расчетных уровней ошибки усечения.

Выводы. 1. Предложен механизм возникновения ошибок усечения умножения на поворачивающие множители и масштабирования в БПФ с прореживанием по частоте и по времени. Получены аналитические зависимости и алгоритмы вычисления величины ошибок в зависимости от номера спектральной составляющей.

2. Показано, что в БПФ с прореживанием по частоте и по времени при наличии масштабирования наибольшие ошибки усечения сосредоточены в окрестности нуля, в области положительных частот.

3. Показано, что суммарные ошибки усечения произведения и масштабирования в БПФ с прореживанием по частоте меньше, чем в БПФ с прореживанием по времени. Причиной этому является то, что в БПФ с прореживанием по времени ошибки усечения возникают как во входных, так и в выходных ветвях базовых операций, а в БПФ с прореживанием по частоте ошибки усечения возникают только в выходных ветвях.

1. *Рабинер Л., Гоулд Б.* Теория и применение цифровой обработки сигналов. – М.: Мир, 1978.
2. LogiCORE IP Fast. Fourier Transform v9.0. Product Guide for Vivado. Design Suite. PG109 December 18, 2013. http://www.xilinx.com/ support/ documentation/ip_documentation/xfft/v9_0/pg109-xfft.pdf.
3. *Ершова О.В., Кириченко Е. В., Семерников Е.А., Чкан А.В.* Ошибки усечения результатов арифметических операций с фиксированной точкой в алгоритмах БПФ // Известия ЮФУ. Технический науки. – 2014. – № 12(161). – С. 138 – 148.
4. *T. Kaneko and B. Liu,* “Accumulation of roundoff errors in fast Fourier transforms”, J. Ass. Comput. Mach. – 1970. – Vol. 17. – P. 537-654.

А.Л. Жуков

ОБ ЭФФЕКТИВНЫХ АЛГОРИТМАХ ОБРАБОТКИ ДАННЫХ ДЛЯ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ НА ПОКРЫТИЯХ МАТРОИДОМ

НАО «Красная поляна», г. Сочи,
AnzorZhukov@gmail.com

Введение

Наибольшей эффективностью при использовании реконфигурируемых конвейерных и мультиконвейерных вычислительных систем обладают алгоритмы потоковой обработки данных [1]. Применительно к наиболее широким классам задач цифровой обработки такие алгоритмы основываются на использовании жадных алгоритмов [2]. Ввиду разнообразия математических моделей данных в различных областях вычислений необходима их предварительная обработка путем приведения к некоторой единой форме [3]. В работе предлагается единый алгебраический подход к анализу и построению алгоритмов обработки данных на основе алгебраического подхода к пространственной грануляции многомерных данных [4].

Постановка задачи

Математической основой построения массовых эффективных (линейной сложности) алгоритмов является преобразование исходной задачи к задаче на матроиде [2]. Существует теорема, показывающая, что на конечном множестве с неотрицательной весовой функцией можно построить жадный алгоритм поиска независимого множества с максимальным весом [2]. Для построения достаточно общих потоковых жадных алгоритмов мы предлагаем использовать преобразование исходных многомерных данных к покрытию матроидами нескольких специфических типов [3]. Для такой задачи становится возможным построение высокоэффективных алгоритмов обработки, позволяющих получить конвейерную реализацию [1].

Пространственная грануляция данных и матроид

Пусть исходные данные заданы в произвольном векторном пространстве V над телом F. Возьмем некоторую систему векторов $v_1, v_2, ..., v_m$ из пространства V (возможно с повторениями векторов). Эта система представляет заданные исходные данные. Тогда мы можем построить матроид M на заданной системе векторов исходных данных. Отметим, что для построения покрывающего матроида не требуется ли-

нейной независимости исходных векторов [2], что очень удобно для произвольных наборов данных. Теперь матроид M будет иметь m элементов и мы будем считать, что элементами построенного на векторах многомерных данных матроида M являются элементы $v_1, v_2, ..., v_m$, т.е. все они различны как элементы матроида M, но некоторые из них могут совпадать как элементы исходного векторного пространства данных V. В качестве баз мы должны выбрать все максимальные линейно независимые подсистемы из $v_1, v_2, ..., v_m$. Мы получили векторный матроид над телом F [2].

Развивая математический аппарат [3], возьмем матрицу $A = \left[a_{i,j} \right]$, $i = 1,...,n$, $j = 1,...,m$ над телом F. Строки матрицы A являются элементами пространства векторов-строк F^m. Поэтому матрице A отвечает векторный матроид над телом F, состоящий из n элементов – строк матрицы A. Его называют матроидом строк матрицы A. Отметим, что здесь строки с разными номерами являются разными элементами матроида, хотя некоторые из них могут совпадать как элементы пространства F^m. Аналогично определяется матроид столбцов матрицы A.

В ряде наших работ введены алгебраические основы грануляции данных с помощью алгебраических элементов Грассманна для произвольной размерности K, представляющих собой матрицы коэффициентов угловых точек пространственных объектов (пространственных гранул с геометрической точки зрения) [3,4]. Очевидно, что в связи с вышеизложенным, такие матрицы представляют собой основу для моделирующих матроидов, на которых можно строить алгоритмы обработки данных [3].

«Жадный алгоритм» и пространственный матроид

Рассмотрим теперь основные свойства базового алгоритма на произвольном матроиде [2]. Пусть M – непустое конечное множество и $w: M \to R_+$ — функция на нем. Число $w(m)$ будем называть весом элемента $m \in M$. Для любого непустого $Q \in M$ положим $w(Q) = \sum_{m \in Q} w(m)$ и будем называть $w(Q)$ весом подмножества Q.

Зафиксируем некоторое семейство $G \subseteq T(Q)$, в котором имеется хотя} бы одно непустое подмножество. Будем смотреть на G как на частично упорядоченное множество относительно теоретико-множественного включения. Элементы этого частично упорядоченного множества G будут являться гранулами [3]. Будем далее считать, что

G удовлетворяет аксиоме независимости [2], т.е. подмножество гранулы является гранулой [3]. Тогда G является матроидом.

В [2] рассматривается задача построения в частично упорядоченном множестве объектов G максимальной гранулы минимально возможного веса [4]. В [2] доказывается теорема о том, что для произвольного матроида и произвольной весовой функции существует жадный алгоритм, который находит искомое подмножество (гранулу). Следовательно, покрытие векторного пространства исходных данных матроидом гранул по методу [3] позволяет строить жадные алгоритмы обработки гранулированных данных с линейной сложностью.

Придавая различный смысл весовой функции, можно привести к формулировке на матроиде множество широко известных задач оптимизации, кластеризации и т.д. [4].

Заключение

В работе показано, что методы пространственной грануляции многомерных данных приводят к возможности их стандартного представления в виде матроида специального вида (пространственное гранулирование). Известные задачи обработки данных в результате могут сводиться к типовому «жадному алгоритму» минимизации весовой функции на матроиде. Поскольку такие «жадные алгоритмы» могут быть представлены как потоковые, открывается возможность разработки типовой структуры алгоритма грануляции (кластеризации) данных на конвейерных вычислительных системах [1].

1. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры.– Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009.– 344 с.
2. *Хаггарти Р.* Дискретная математика для программистов. – М.: Техносфера, 2003. – 320 с.
3. *Бутенков С.А., Жуков А.Л., Кривша Н.С.* Алгебраический подход в теории информационной и пространственной грануляции на базе элементов Грассманна // Сб. трудов Международного конгресса по интеллектуальным системам и информационным технологиям IS&IT-11, Геленджик-Дивноморское, 2-9 сентября 2011. – М.: ФИЗМАТЛИТ, 2011. – Т.1. – С. 396 – 403.
4. *Бутенков С.А., Жуков А.Л.* Информационная грануляция на основе изоморфизма алгебраических систем \\ Сб. трудов Международной алгебраической конференции, посвященной 80-летию со дня рождения А.И. Кострикина, Нальчик. – 2009. – С. 206 – 209.

Кхайнг Мин Зо

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА УМНОЖЕНИЯ МАТРИЦ ДЛЯ ИСПОЛЬЗОВАНИЯ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ

Национальный исследовательский университет «МИЭТ», г. Зеленоград, zawminkhaing51@gmail.com

В докладе описываются алгоритмы умножения матриц, с помощью которого можно быстро использовать крупномасштабным искусственным нейронным сетям за счет сокращения времени на изучение темы: «Умножение матриц». Мы предлагаем изучать алгоритмы матричного умножения на основе системы с общей памятью, а системы с распределенной памятью изучать с помощью технологий MPI и OpenMP. Результаты проведенных экспериментов показывают, что использование наших алгоритмов умножения матриц способствует сокращению времени освоения проблемы крупномасштабного матричного умножения и этот результат может оказаться полезным для разработки технических систем, использующих умножения матриц и применение нейронных сетей.

Введение

Нейросетевые алгоритмы находят широкое применение в вычислительной практике. В частности, особое внимание уделяется классификации и кластеризации изображений, распознаванию речи и изображений, прогнозу финансовых показателей, искусственному синтезу речи, аппроксимации функционалов, совершенствованию методов электроимпедансной и магнитоиндукционной томографии, анализу данных (data mining)[1]. Использование нейронных сетей с большим количеством нейронов, которые работают на больших параллельных компьютерах, может быть значительно сокращено, так как они все работают паралелльно. Однако человеческий мозг состоит приблизительно из 100 миллиардов нейронов, и каждый из них имеет около 1000 синапсов. Если мы предположим, что каждому синапсу необходим один байт, то это составит примерно ⟦10⟧^14 байт данных (100 терабайт). Мозг человека также способен выполнять примерно ⟦10⟧^14 до ⟦10⟧^16 команд в секунду [2].В настоящее время массивно параллельные компьютеры предлагают значительные возможности для моделирования человеческого мозга или для создания интеллектуальных систем. Но есть также проблема, которую необходимо реализовать, – параллельный алгоритм для использования в параллельном суперкомпьютере.

Предлагаемая работа является попыткой реализовать параллельные алгоритмы умножения матриц для использовании ИНС на парал-

лельных компьютерах. Результаты показывают, что использование параллельных алгоритмов на суперкомпьютере сокращает время операций за счет распараллеливания обработки входных данных и взвешенных данных, таких как весовые коэффициенты. Это может быть полезно в качестве первого шага к созданию программной модели функционирования человеческого мозга, а также для разработки полезных технических средств (например, для распознавания образов). Поскольку в процессе использования ИНС требуется обработка входных данных и взвешенных данных, и их часто представляют в матричной форме.

Линейная нейронная сеть

Линейная нейронная сеть, состоящая из распределительных нейронов и одного выходного нейрона, имеющего линейную функцию активации, называется адаптивным нейронным элементом. Выходное значение такой сети:

$$y = \sum_{i=1}^{n} w_{ij} x_i - T,$$

где T – пороговое значение указанного нейрона, а w_{ij} – весовой коэффициент, соответствующий j-ому распределительному нейрону. Заметим, что нейронную сеть с несколькими выходными элементами можно представить как суперпозицию соответствующего числа нейронных сетей с одним выходным элементом, ввиду независимости выходных элементов друг от друга. В этом разделе для простоты мы будем рассматривать сети с одним выходным элементом.

Для настройки весовых коэффициентов линейной нейронной сети с целью минимизации среднеквадратичной ошибки можно использовать матричное решение системы линейных уравнений.

Пусть размерность обучающей выборки – L; число выходных нейронов – m; число входных нейронов – n. Тогда матрицы выходных значений, входных значений и весовых коэффициентов, соответственно, имеют вид:

$$Y = \begin{bmatrix} y_1^1 \; y_1^2 \ldots . y_1^L \\ y_2^1 \; y_2^2 \ldots . y_2^L \\ \ldots \ldots \ldots \ldots \ldots \\ y_m^1 \; y_m^2 \ldots . y_m^L \end{bmatrix} - \text{матрица выходных значений,}$$

$$X = \begin{bmatrix} x_1^1 \; x_1^2 \ldots . x_1^L \\ x_2^1 \; x_2^2 \ldots . x_2^L \\ \ldots \ldots \ldots \ldots \ldots \\ x_m^1 \; x \ldots . x_m^L \end{bmatrix} - \text{матрица входных значений,}$$

$$W = \begin{bmatrix} w_{11} w_{12} \ldots . w_{1m} \\ w_{21} w_{22} \ldots . w_{2m} \\ \ldots \ldots \ldots \ldots \ldots . \\ w_{n1} w_{n2} \ldots . w_{nm} \end{bmatrix} - \text{матрица весовых коэффициентов.}$$

Тогда уравнение линейной нейронной сети можно представить в виде:

$$Y = WX$$

Параллельные алгоритмы умножения матриц

Умножение матриц возможно выполнять как в последовательном, так и параллельном режиме. Использование системы с распределенной памятью предлагает два возможных алгоритма α и β: способ α – разделение матриц, состоящий в разбиении данных на прямоугольные фрагменты (блоки), которые распределяются по 4 процессорам и перемножаются в соответствии с алгоритмом умножения прямоугольных матриц. Исходные матрицы находятся на центральном процессоре P0, который и производит их разбиение. Способ β состоит в разбиении второй матрицы на столбцы, реализующие так называемое ленточное разбиение, и параллельном умножении целой матрицы на вектор в соответствии с алгоритмом.

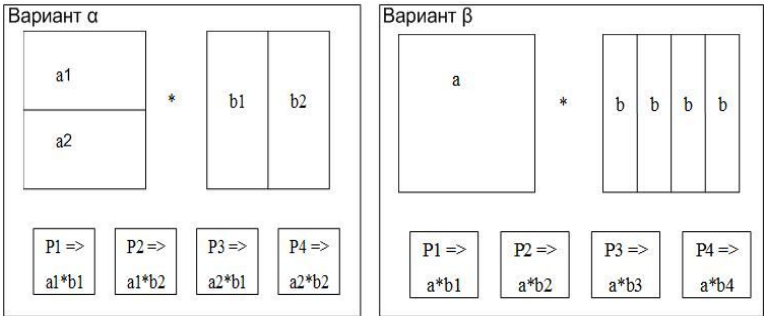


Рис.2. Вариант α и вариант β реализации параллельного алгоритма умножения матриц

Высокопроизводительные матричные вычисления можно реализовать с применением как распределенного, так и многопоточного программирования. В последнее время широкое распространение получили многоядерные процессоры с помощью OpenMP. Здесь рассматривается умножение матриц, для которой приводятся различные алгоритмы, основанные не только методами MPI, но и алгоритм параллельного многопоточного матричного умножения методами OpenMP.

Результаты эксприментов

Эксперименты выполнялись с N=200,000 на компьютере с двумя процессорами Intel®Xeon® E5335 (2.00)ГГц, (8)Мб кэш-памяти и (4) Гб оперативной памяти,8 вычислительных ядер.

В процессе реализации параллельных алгоритмов следует учитывать следующие критичные параметры разрабатываемой программы – временные характеристики, т.е. время, необходимое для реализации задачи, суммарный объем памяти,занимаемой всеми фрагментами матриц, размер собственно, программы, реализующей данный метод (таблица 1).

Таблица 1

Сравнение время работы таблиц вариант α и вариант β реализации параллельного алгоритма умножения матриц

Размер матрицы	Способ разбиения α			Способ разбиения β		
	Количество процессоров					
	4	16	64	4	16	64
512	0.988761	0.448983	0.881854	0.625327	0.290471	0.512724
1024	7.8482	3.51786	4.82385	6.79394	2.77872	37.8035
2048	63.1296	27.9336	31.1348	93.1467	37.8035	49.4763

В таблице 2 приведено время работы при реализации параллельной обработки данных на компьютерах с общей памятью при использовании технологии OpenMP.

Таблица 2

Время работы при реализации параллельной обработки данных на компьютерах с общей памятью при использовании технологии OpenMP

Количество ядер	Размер матрицы		
	512*512	1024*1024	2048*2048
1	1.58059	17.9033	266.213
2	0.802704	7.52096	152.528
3	0.649988	5.40284	76.5229
4	0.40159	3.79278	43.7907
5	0.320477	3.03067	32.7685
6	0.270627	2.70289	25.6377
7	0.232463	2.17568	23.3892
8	0.200407	1.89475	21.2664

Заключение

Оптимизация больших матричных вычислений, которые необходимы для оценки нейронных сетей, является сложной задачи. Как показано в данной работе, мы интегрируем алгоритмы умножения матриц для исследования эффективности использовании ИНС и для поддержки функционирования систем на базе нейронных сетей. Результаты проведенных экспериментов показывают, что использование наших алгоритмов умножения матриц способствует сокращению времени освоения проблемы крупномасштабного матричного умножения и этот результат может оказаться полезным для разработки технических систем, использующих умножения матриц и применение нейронных сетей.

1. *Изотов П.Ю., Суханов С.В., Головашкин Д.Л.* Технология реализации нейросетевого алгоритма в среде CUDA на примере распознавания рукописных ЦИФР. Компьютерная оптика. – 2010. – Т. 34. – №2.
2. *Lyle N. Long, Ankur Gupta.* Scalable Massively Parallel Artificial Neural Networks. Journal of Aerospace Computing, Information, and Communication, – 2008. – Vol. 5. – No. 1.

К.С. Исупов, В.С. Князьков, А.С. Куваев, М.В. Попов

ИСПОЛЬЗОВАНИЕ СИСТЕМ ОСТАТОЧНЫХ КЛАССОВ ДЛЯ ПОДДЕРЖКИ ПАРАЛЛЕЛЬНОЙ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ МНОГОКРАТНОЙ ТОЧНОСТИ НА СИСТЕМАХ С ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ[*]

Вятский государственный университет, г. Киров,
isupov.k@gmail.com

Введение

Целочисленные операции многократной точности являются ключевым компонентом для многих современных приложений, таких как криптосистемы с открытым ключом [1] и геометрические алгоритмы. Поскольку многоразрядные типы данных изначально не поддерживаются в аппаратном обеспечении общего назначения, используется программная эмуляция арифметики многократной точности – высокоточные библиотеки, такие как The GNU Multiple Precision Arithmetic Library (GMP) [2], Library of Efficient Data types and Algorithms (LEDA) [3] и пр. Однако эти средства в большинстве случаев нацелены на центральные процессорные архитектуры (CPU). Вместе с тем современные графические процессоры (GPU) являются мощным и экономически эффектив-

[*] Работа выполнена при финансовой поддержке РФФИ (проект № 16-37-60003 мол_а_дк)

ным средством для ресурсоемких вычислений общего назначения [4]. В основе многих современных суперкомпьютеров, в том числе наиболее производительных по версии TOP500, лежит гибридная CPU/GPU архитектура. Поэтому разработка программного обеспечения для поддержки операций многократной точности на гибридных вычислительных системах является важной задачей.

Система остаточных классов

Типы данных, реализованные в существующих высокоточных библиотеках, основаны на традиционных последовательных методах длинной арифметики и реализуют лишь последовательную обработку одного длинного числа. Учитывая большую вычислительную сложность алгоритмов, это негативно сказывается на производительности и зачастую не позволяет в полной мере использовать все ресурсы графических ускорителей, ориентированных на массовый параллелизм. В этой области привлекательными выглядят системы остаточных классов (СОК) [5, 6], позволяющие разделить трудоемкие операции по обработке многоразрядных чисел на несколько операций меньшей разрядности, выполняемых параллельно.

СОК определяется набором попарно взаимно простых модулей $\{m_1, m_2, \ldots, m_n\}$. Динамический диапазон задается произведением $M = m_1 \times m_2 \times \cdots \times m_n$. Любое целое X от 0 до $M-1$ представляется кортежем $\langle x_1, x_2, \ldots, x_n \rangle$, где $x_i = |X|_{m_i}$, или то же самое, что $x_i \equiv X \bmod m_i$. Так как остатки x_i не зависят друг от друга, такие арифметические операции, как сложение, вычитание и умножение могут осуществляться без переноса между остатками, что в большинстве случаев является ограничивающим фактором в двоичных системах счисления.

Интервальная оценка относительных величин в СОК

Недостатком СОК является сложность операций, требующих оценки величины числа. К таким операциям относятся сравнение, определение знака, обнаружение переполнения, общее деление и пр. Классический способ выполнения этих операций основан на китайской теореме об остатках [7] и состоит в вычислении двоичных представлений чисел с их последующим анализом. Такой способ является трудоемким, так как требует выполнения сложных операций умножения и сложения больших чисел и редукции по модулю *M*. Многие другие методы оценки величины в СОК базируются на преобразовании в систему со смешанными основаниями (Mixed-Radix Conversion) [5, 7], однако они часто оказываются неэффективными, так как требуют большого объема вычислений и/или использования больших подстановочных таблиц.

Альтернативный метод основан на вычислении и анализе интервальной функции от остатков, локализующей относительную величину числа в СОК – интервально-позиционной характеристики (ИПХ) $I(X/M)$, которая представляется двумя направленно округленными числами – нижней границей $\underline{X/M}$ и верхней границей $\overline{X/M}$ – и локализует в единичном полуинтервале значение *X*, масштабированное относительно произведения модулей *M* так, что $\underline{X/M} \le X/M \le \overline{X/M}$. Границы ИПХ – числа с плавающей точкой машинной точности, вычисляемые с использованием направленных округлений. Вычисление границ ИПХ на основе остатков числа в СОК производится за линейное и логарифмическое время в последовательном и параллельном случаях, соответственно. Кроме того, над ИПХ определены базовые арифметические операции. Сложение и умножение ИПХ выполняется в соответствии с правилами, справедливыми для обычных вещественных интервалов. Умножение и деление определяются с учетом относительности выражаемых величин, т.е. дополнительно учитывается вес $1/M$. Основные аспекты использования ИПХ для выполнения проблемных операций в СОК рассмотрены в [8].

Многоразрядный целый тип данных

Целое число произвольной длины представляется знаком *s*, мантиссой (модулем числа) в СОК $X = \langle x_1, x_2, \ldots, x_n \rangle$ и ИПХ мантиссы $I(X/M) = \left[\underline{X/M}, \overline{X/M} \right]$. Мантисса X выражает абсолютное значение числа. Знак и цифры (остатки) мантиссы представлены целыми машинными числами. ИПХ не участвует в образовании значения числа и используется при необходимости оценки величины мантиссы. Границы ИПХ представлены в формате с расширенной экспонентой (extended-range floating-point representation). Числовое представление с расширенной экспонентой строится посредством объединения машинного целого *c* со стандартным машинным числом с плавающей точкой *f*. Эта пара рассматривается как число $f \times B^c$, где B – основание системы счисления (константа). Основные алгоритмы обработки таких представлений легки в реализации и не приводят к значительным вычислительным издержкам [9]. Применение формата с расширенной экспонентой обеспечивает корректное использование ИПХ практически при любых значениях произведения модулей СОК *M*. Для компактной записи описанного числового представления используется нотация

$$x \to \{s, X, I(X/M)\}, \qquad (1)$$

которая выражает значение

$$x = (-1^s) \times \left| \sum_{i=1}^{n} x_i \mid M_i^{-1} \mid_{m_i} M_i \right|_M ,$$

где $M_i = M / m_i$, а $\mid M_i^{-1} \mid_{m_i}$ – мультипликативная инверсия M_i по отношению к m_i. Разрядность числа в битах определяется величиной $\lfloor \log_2 M \rfloor$. На рис. 1 представлены структуры данных, соответствующие описанному числовому формату (используется синтаксис языка C).

```
/*Extended-Range Floating-Point*/
typedef struct {
    er_sig_t sig;   //Significand
    er_exp_t exp;   //Exponent
} __extended_range_struct;

/*Modular Multiple-Precision Integer*/
typedef struct {
     mi_sig_t s;                      //Sign
     mi_res_t residue[MODULI_SIZE];   //Significand in RNS
     er_t     ipc_low;                //Lower IPC bound
     er_t     ipc_upp;                //Upper IPC bound
} __mp_integer_struct;
```

Рис. 1. Структуры данных для представления многоразрядных чисел

Общая схема вычислений

Параллелизм СОК позволяет эффективно задействовать ресурсы как CPU, так и GPU. При использовании GPU сразу n потоков обрабатывают цифры многоразрядной мантиссы, при этом каждый i-й GPU-поток выполняет операции над остатком x_i по i-му модулю m_i. Вычисления с ИПХ выполняются параллельно с вычислениями в СОК с использованием формул интервальной арифметики. Анализ ИПХ производится, когда нужно оценить мантиссу, к примеру, для контроля переполнения или определения знака разности. В этом случае вводятся точки синхронизации потоков. При необходимости ИПХ “обновляется”, т.е. пересчитывается на основании цифр мантиссы. Для обновления ИПХ используется специальный алгоритм [8], который также хорошо распараллеливается. Схема вычислений на GPU представлена на рис. 2.

В случае вычислений на CPU обработка мантисс и ИПХ векторизуется при помощи SIMD-расширений. Кроме того, при большом числе модулей и низком уровне параллелизма решаемой задачи может оказаться целесообразной многопоточная CPU-реализация вычислений, при которой цифры мантисс обрабатываются группами на различных ядрах универсального процессора.

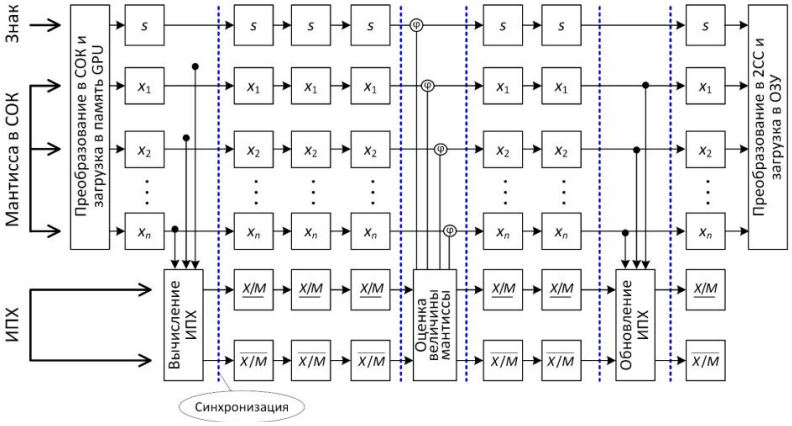


Рис. 2. Схема многоразрядных вычислений на GPU

Основные операции над числами вида (1) выполняются просто. К примеру, произведением x и y является число $z \to \{s_z, Z, I(Z/M)\}$, где $s_z = s_x \oplus s_y$, $Z = |X \times Y|_M$, $I(Z/M) = I(X/M) \times I(Y/M)$. При этом, если $\underline{Z/M} > \overline{\frac{M-1}{M}}$, то возникло переполнение; напротив, если $\overline{Z/M} \le \underline{\frac{M-1}{M}}$, то переполнения не возникло. Цифры мантисс умножаются параллельно:

$$Z = |X \times Y|_M = \left\langle \left|x_1 \times y_1\right|_{m_1}, \left|x_1 \times y_1\right|_{m_2}, \cdots, \left|x_1 \times y_1\right|_{m_n} \right\rangle.$$

Для нахождения разности $z = x - y$ в случае $s_x = s_y$ выполняется сравнение $I(X/M)$ и $I(Y/M)$: если $\underline{X/M} > \overline{Y/M}$, то $s_z = s_x$, $Z = |X - Y|_M$, $I(Z/M) = I(X/M) - I(Y/M)$. Если же $\overline{X/M} < \underline{Y/M}$, то $s_z = \neg s_x$, $Z = |Y - X|_M$, $I(Z/M) = I(Y/M) - I(X/M)$. Если ИПХ мантисс пересекаются, то знак результата требует уточнения.

Заключение

Рассмотрены некоторые аспекты использования систем остаточных классов для программной реализации целочисленной арифметики многократной точности с распараллеливанием обработки отдельных цифр многоразрядных чисел. Эти аспекты отражены в разрабатываемой высокоточной библиотеке для высокопроизводительных систем с гибридной CPU/GPU архитектурой. Прототипом библиотеки является разработанное ранее решение для универсальных процессоров [10]. В дальнейшем на основе описанного подхода планируется поддержка типов данных с плавающей точкой многократной точности для гибридных

CPU/GPU систем, а также перенос алгоритмов и структур данных на другие современные многоядерные архитектуры.

1. *Davies J.* Implementing SSL/TLS using cryptography and PKI. – USA: John Wiley & Sons, 2011.
2. *Granlund T.* The GNU Multiple Precision Arithmetic Library, version 6.1.0 [Electronic resource] URL: https://gmplib.org/gmp-man-6.1.0.pdf (date of access: 15.04.2015).
3. *Mehlhorn K., Näher St.* LEDA: A Platform for Combinatorial and Geometric Computing. – UK: Cambridge University Press, 1999.
4. *Owens J.D., Luebke D., Govindaraju N.K., Harris M., Kruger J., Lefohn A.E., Purcell T.J.* A Survey of General-Purpose Computation on Graphics Hardware // Computer Graphics Forum. – 2007. – Vol. 26, No. 1. – P. 80 – 113.
5. *Szabo N., Tanaka R.* Residue Arithmetic and its Application to Computer Technology. – USA: McGraw-Hill, 1967. – 236 p.
6. *Акушский И.Я., Юдицкий Д.И.* Машинная арифметика в остаточных классах. – М.: Сов. радио, 1968. – 440 с.
7. *Knuth D.E.* The Art of Computer Programming. Vol. 2. Seminumerical Algorithms. – USA: Addison-Wesley Professional, 1997. – 784 p.
8. *Исупов К.С., Князьков В.С.* Немодульные вычисления в системах остаточных классов с интервально-позиционными характеристиками. – Киров, 2015. – 92 с.: ил. – Библиогр. 54 назв. – Деп. в ВИНИТИ РАН 26.03.2015, № 61-В2015.
9. *Hauser J.R.* Handling Floating-Point Exceptions in Numeric Programs // ACM Transactions on Programming Languages and Systems. – 1996. – Vol. 18, No. 2. – P. 139 – 174.
10. *Isupov K., Knyazkov V.* A Modular-Positional Computation Technique for Multiple-Precision Floating-Point Arithmetic // Lecture Notes in Computer Science. – 2015. – Vol. 9251. – P. 47 – 61.

А.Г. Коваленко

МАСШТАБИРУЕМАЯ СТРУКТУРНО-ПРОЦЕДУРНАЯ РЕАЛИЗАЦИЯ ИТЕРАЦИОННЫХ МЕТОДОВ РЕШЕНИЯ СЕТОЧНЫХ УРАВНЕНИЙ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог, k.a.g@bk.ru

Многие задачи математической физики описываются дифференциальными уравнениями, при решении которых на вычислительных системах в заданной области выбирается равномерная сетка с начальными значениями, а частные производные представляются в конечно-разностном виде. Таким образом, при дискретизации математической модели получается система линейных алгебраических уравнений (СЛАУ), для разрешения которой применяются итерационные методы,

поскольку они позволяют получить решение с необходимой заранее заданной точностью быстрее прямых методов [1].

Перспективным вычислительным средством в решении задач математической физики являются реконфигурируемые вычислительные системы (РВС) на основе программируемых логических интегральных схем (ПЛИС). Такие системы адаптируются под вычислительный граф решаемой задачи и, как было показано во многих работах [2, 3], обеспечивают почти линейный рост производительности при увеличении доступного аппаратного ресурса.

Однако узким горлом РВС при масштабировании являются каналы распределенной памяти [3]. К сожалению, современные тенденции развития реконфигурируемых вычислительных систем обуславливают увеличение доступного аппаратного ресурса ПЛИС в разы, в то время как количество каналов памяти увеличивается на единицы или остается неизменным.

Наглядным примером задачи математической физики, использующим итерационный метод решения сеточных уравнений, является задача фильтрации жидкости в пористой среде [4]. Уравнение задачи позволяет вычислять значения давления p на основе заданной глубины D и подвижности жидкости λ. При решении задачи используется метод релаксации.

В работе [5] при реализации задачи фильтрации на РВС был выбран метод распараллеливания, заключающийся в одновременной обработке нескольких соседних точек сеточной области, что соответствовало аппаратному ресурсу выбранной РВС. При этом на нечетных итерациях происходили чтение данных первого канала и запись результатов во второй, а на четных – наоборот. Однако такой подход требует линейного роста каналов распределенной памяти, а потому плохо поддается масштабированию.

Если же для реализации выбрать современную РВС, например, вычислительный модуль «Тайгета», разработанный на основе ПЛИС фирмы Xilinx семейства Virtex-7 в Научно-исследовательском центре суперЭВМ и нейкомпьютеров, г. Таганрог, то вычислительная структура задачи фильтрации займет около 5 % аппаратного ресурса одной ПЛИС платы вычислительного модуля (ПВМ) «Тайгета» [6]. Кроме того, для организации информационных обменов требуется четыре канала распределенной памяти: три в качестве источников данных и один в качестве приемника. Тогда аппаратного ресурса ПВМ «Тайгета» достаточно для параллельной обработки 128 точек сетки. Однако в этом случае потребуется 512 каналов памяти, в то время как на ПВМ «Тайгета» доступно только 16 каналов.

Таким образом, необходим принципиально другой подход к организации вычислений при решении задач, использующих итерационные методы решения сеточных уравнений, обеспечивающий максимально возможную структурную реализацию итераций алгоритма.

Для обеспечения эффективной реализации необходимо выполнить распараллеливание по итерациям: результат обработки после первой

итерации не записывать во внешнюю память, а сразу подавать на следующую итерацию. Например, как уже говорилось, аппаратный ресурс ПВМ «Тайгета» позволяет реализовать 128 вычислительных структур, каждая из которых соответствует базовому подграфу задачи фильтрации жидкости в пористой среде (рис. 1). Соответственно за один проход можно выполнить преобразования 128 итераций вычислительного процесса решения задачи фильтрации. При этом требуемое количество каналов внешней памяти остается равным четырем.

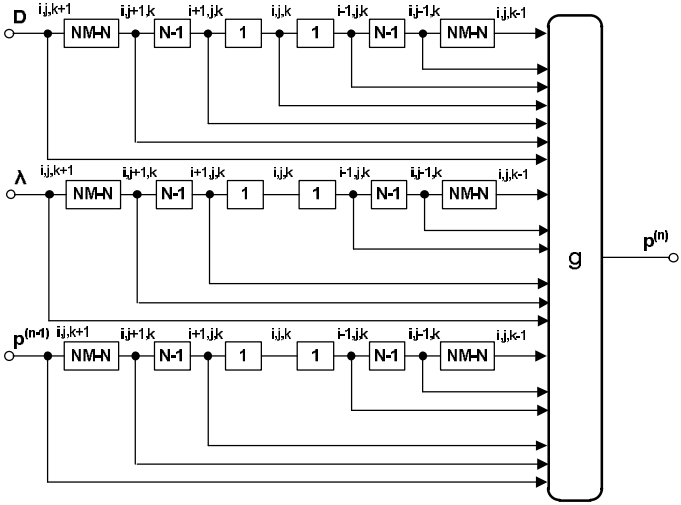


Рис. 1. Граф-схема базового подграфа задачи фильтрации жидкости в пористой среде, содержащая элементы задержки операндов для сокращения каналов распределенной памяти

На рис. 2 показано отображение вычислительной структуры задачи фильтрации жидкости в пористой среде на архитектуру платы вычислительного модуля «Тайгета» при использовании распараллеливания по итерациям.

На рис. 2 блок А выполняет преобразования, согласно рис. 1, а также осуществляет проверку условия завершения итерационного процесса вычислений. Если это условие выполнилось, то текущие значения в узлах сетки проходят через оставшиеся блоки А без изменений. Блок Б представляет собой коммутатор, который определяет, какой канал в паре (*p1-p2*) является источником данных, а какой – приемником.

Оценить эффективность реализации итерационных методов решения сеточных уравнений на РВС можно при помощи формулы:

$$P = Nop(g) \cdot N_A \cdot v \cdot k,$$

где *Nop(g)* – количество операций в базовом подграфе *g*;

N_A – количество реализованных вычислительных структур (базовых подграфов g);

v – тактовая частота работы вычислительной структуры задачи;

k – коэффициент, учитывающий накладные расходы на обмен данными с памятью (k≈0,85-0,9).

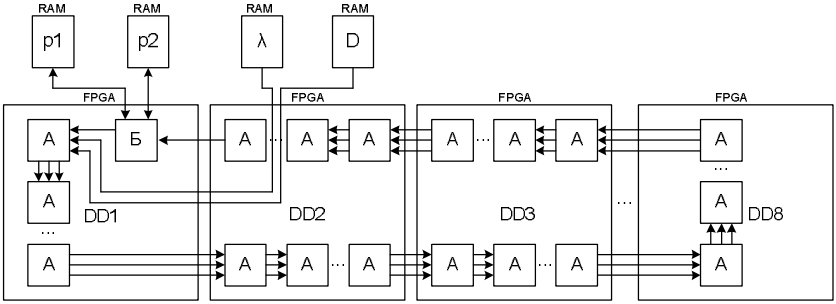


Рис. 2. Отображение вычислительной структуры задачи фильтрации на архитектуру платы вычислительного модуля «Тайгета»

Таким образом, использование распараллеливания по итерациям при решении сеточных уравнений итерационными методами на реконфигурируемых вычислительных системах позволяет эффективно масштабировать реализацию при увеличении доступного аппаратного ресурса РВС и снизить требования к количеству необходимых каналов памяти.

1. *Самарский А.А., Николаев Е.С.* Методы решения сеточных уравнений. – М.: Главная редакция физико-математической литературы изд-ва «Наука», 1978. – 592 с.
2. *Каляев А.В, Левин И.И.* Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
3. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – 2-е изд., перераб. и доп./ под общ. ред. И. А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
4. *Леонтьев Н.Е.* Основы теории фильтрации: Учебное пособие. – М.: Изд-во Центра прикладных исследований при механико-математическом факультете МГУ, 2009 – 88 с.
5. *Коваленко А.Г.* Структурно-процедурная реализация задачи фильтрации жидкости в пористой среде // Материалы Международной научно-технической конференции «Многопроцессорные вычислительные и управляющие системы – 2007».–Таганрог: Изд-во ТТИ ЮФУ, 2007. –Т.1. – С. 293 – 297.
6. http://superevm.ru/index.php?page=taigeta-1-2

А.С. Котляров

ПРОБЛЕМА ОБРАБОТКИ СЕТЕВЫХ ПРОТОКОЛОВ В ВЫСОКОСКОРОСТНЫХ КАНАЛАХ ДАННЫХ

НИИ многопроцессорных вычислительных систем ЮФУ, г. Таганрог, a-svetoch@ya.ru

В последнее десятилетие происходит непрерывное увеличение объемов сетевого трафика. Рост числа абонентов глобальной сети, появление новых сервисов и совершенствование старых приводят к постоянно возрастающей нагрузке на каналы связи, что требует расширения пропускной способности каналов связи и усложняет реализацию обработки сетевых протоколов. Так, уже при скорости канала порядка 10 Гбит/с при обработке сетевых протоколов современные процессоры серверного уровня загружены практически на сто процентов, что не позволяет использовать их для решения своих непосредственных задач [1].

Увеличение числа ядер и/или процессоров не приводит к качественному изменению ситуации из-за роста накладных расходов на обмен данными между процессорами. Решение данной проблемы ведущие специалисты видят в использовании специализированных устройств обработки – сверхбольших интегральных схем (СБИС) или программируемых логических интегральных схем (ПЛИС) [2]. Использование СБИС оправдано при массовом выпуске устройств передачи данных, алгоритм работы которых не предполагает частых изменений или расширений [3].

Альтернативным вариантом для решения задач ускорения обработки стека протоколов TCP/IP для высокоскоростных каналов данных является использование структурных и параллельно-конвейерных вычислений на основе реконфигурируемых вычислительных систем (РВС). Данные вычислительные системы, которые представляют собой вычислительные поля ПЛИС, обладают мощной пространственно-коммуникационной системой, что, в свою очередь, позволяет выполнить аппаратную реализацию всех необходимых функций структурно-процедурным способом. Такая организация вычислений при решении задач в предметной области позволяет построить линейный конвейер по передаче данных от входящего интерфейса к исходящему и наоборот [4].

Необходимо отметить, что реализация алгоритмов обработки сетевых пакетов IP протокола с пропускной способностью более 10 Гбит/с на РВС связана с большими трудозатратами [5]. Наиболее популярными технологиями программирования при реализации сетевых алгоритмов на РВС являются VHDL и OpenCL [6]. Как показал анализ существующих методов и средств реализации алгоритмов для ПЛИС, наиболее эффективным программным комплексом является комплекс на базе языка программирования

высокого уровня COLAMO с неявным описанием параллелизма. Отличительной особенностью COLAMO является возможность отображения структуры реализуемой задачи на теоретически неограниченное вычислительное поле, которое состоит из множества ПЛИС [7].

В настоящее время многие крупные разработчики коммуникационного оборудования идут по пути тесного взаимодействия с производителями ПЛИС [2]. Преимущества устройств на базе ПЛИС позволяют эффективно решать задачи обработки сетевых протоколов в высокоскоростных каналах данных. Однако ограничения, накладываемые существующими технологиями программирования (VHDL и OpenCL) при реализации сетевых алгоритмов на РВС, приводят к необходимости использования одной, но очень мощной ПЛИС, адаптированной именно для задач обработки сетевых протоколов в высокоскоростных каналах связи. Всё это приводит к удорожанию сетевого оборудования и, в конечном итоге, несмотря на возможности перепрограммирования ПЛИС, к невозможности использовать данное оборудование, если необходимо будет увеличить пропускную способность канала связи. Решение данной проблемы видится в использовании программного комплекса COLAMO, так как данный комплекс позволяет не ограничивать число ПЛИС на плате при проектировании коммуникационного сетевого оборудования. Использование множества ПЛИС гарантированно обеспечивает обработку сетевых протоколов в высокоскоростных каналах данных с возможностью расширения пропускной способности в дальнейшем без необходимости замены сетевого оборудования. При этом на этом же устройстве появляется возможность реализовать дополнительные функции: динамическую маршрутизацию, аппаратный антивирус или файервол, туннелирование, шифрование данных, дублирование в зависимости от необходимости.

1. 10 GbE for high-performance real-time embedded systems: What you need to know. AdvancedIO Systems Inc. - May 13, 2007.
2. http://mil-embedded.com/article-id/?2131=.
3. 40 and 100 Gigabit Ethernet: Ready for Real-Time? AdvancedIO Systems Inc. - March 2009. http://www.advancedio.com/wp-content/uploads/2010/10/AIO_RTC_MAR_09.pdf.
4. *Капустин В., Дементьев Е.* Информационно-вычислительные сети: учебное пособие. – Ульяновск: УлГТУ, 2011. – 141с.
5. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – 2-е изд., перер. и доп. / под общ. ред. И.А. Каляева. – Ростов на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
6. *Таненбаум Э.* Компьютерные сети. – 4-е изд. – СПб.: Питер, 2003. – 992 с.
7. Взгляд на 10G Ethernet со стороны FPGA разработчика. НТЦ Метротек 24 августа 2014. https://habrahabr.ru/company/metrotek/blog/234369/
8. *Каляев И.А., Левин И.И.* Высокопроизводительные модульно-наращиваемые многопроцессорные системы на основе реконфигурируемой элементной базы // Вычислительные методы и программирование. – М.: Изд-во МГУ, 2007. – Т.8. – №1. – С. 181-190.

Н.С. Кривша, В.В. Кривша

АЛГОРИТМЫ ГРАНУЛЯЦИИ МНОГОМЕРНЫХ ДАННЫХ В ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Инженерно-технологическая академия ЮФУ,
ООО «НИЦ супер-ЭВМ и нейрокомпьютеров», г. Таганрог,
natalie-home@yandex.ru

Введение

Существуют различные подходы к построению высокоэффективных (реконфигурируемых) вычислительных систем (ВС), основанные на преобразованиях заданных алгоритмов вычислений к эффективной форме [1]. Однако при этом проектирование алгоритмов, как правило, ведется без учета их вычислительной эффективности. И только на этапе проектирования ВС для этих алгоритмов используются методы повышения эффективности вычислений. Одним из них является использование стандартных макросов для задач, допускающих эффективное распараллеливание [1]. В качестве основы для проектирования таких макросов применяются методы вычислений на гранулированных данных [2] и специальные «*жадные* (линейной сложности) алгоритмы», ориентированные на использование в высокоэффективных реконфигурируемых вычислительных системах (РВС).

Постановка задачи

Основой построения «жадных алгоритмов» является преобразование исходной задачи к задаче на матроиде [4]. Известна теорема, показывающая, что на конечном множестве с неотрицательной весовой функцией можно построить «жадный алгоритм» поиска независимого множества с максимальным весом. К такой задаче можно свести многие классические численные задачи, в том числе, и задачу кластеризации [5]. Покажем, что пространство гранулированных данных является матроидом и рассмотрим алгоритмы грануляции для него, допускающие реализацию в виде макросов РВС.

Модель матроида гранулированных данных

Определение 1. *Матроидом* $M = \langle E, \varepsilon \rangle$ называется конечное множество E с нормой $|E| = n$ и семейство его подмножеств $\varepsilon \subset 2^E$, для которых выполняются следующие аксиомы:

а) $\varnothing \in \varepsilon$;

b) $A \in \varepsilon \,\&\, B \subset A \Rightarrow B \in \varepsilon$;

c) $A, B \in \varepsilon \,\&\, |B| = |A| + 1 \Rightarrow \exists e \in B \setminus A \, A \cup \{e\} \in \varepsilon$.

В наших работах были введены математические модели покрытия пространственными гранулами для аффинного пространства данных [6]. Покажем, что множество пространственных гранул **G**, покрывающее область значений данных, является матроидом [2]. Все элементы покрытия ${}^{i}G, {}^{j}G \in \mathbf{G}$ линейно независимы, значит, среди множества точек данных, принадлежащих грануле ${}^{i}G$ есть, по крайней мере, одна точка, не входящая в гранулу ${}^{j}G$ и не выражающаяся в виде линейно независимой комбинации точек, принадлежащих ${}^{j}G$ [7]. Тогда, добавляя такую точку к ${}^{j}G$, мы получим гранулу ${}^{k}G$, удовлетворяющую аксиоме *c)* определения 1. Элементы покрытия также удовлетворяют аксиомам *a)* и *b)* [4]. Следовательно, множество пространственных гранул **G** является матроидом.

Алгоритмы линейной сложности на матроидах

Рассмотрим теперь матроид M и весовую функцию на нем $w: E \to R_{+}$. Типовая задача оптимизации формулируется как поиск $X \in \varepsilon$, для которого выполняется:

$$w(X) = \max_{Y \in \varepsilon} w(Y), \text{ где } w(Z) := \sum_{e \in Z \subset E} w(e), \qquad (1)$$

т.е. сводится к выбору подмножества наибольшего веса [4]. Придавая различный смысл весовой функции, можно привести к формулировке в виде (1) множество известных задач оптимизации, кластеризации и т.д. [3]. В [5] доказывается следующая теорема о том, что для произвольного матроида и произвольной весовой функции существует «жадный алгоритм», который находит независимое множество X с наибольшим весом. Следовательно, покрытие по методу [6] позволяет строить «жадные алгоритмы» обработки гранулированных данных с линейной сложностью, эффективно реализуемые на конвейерных и мультиконвейерных системах системах. На их основе возможно построение макросов решения многих типовых задач обработки данных [1].

Алгоритмы гранулирования данных

Задача грануляции данных сводится к разбиению исходного множества на гранулы (кластеры) в соответствии с заданной математической моделью гранулы [7].

Дано множество объектов данных I, каждый из которых представлен набором атрибутов. Требуется построить множество гранул

(кластеров) C и отображение F множества I на множество C, т.е. $F: I \to C$. Отображение F задает модель данных, являющуюся решением задачи грануляции и алгоритма грануляции.

Выделяют два основных типа методов грануляции [4]:

a) агломеративные методы, использующие построение гранул *снизу вверх* путем последовательного объединения гранул;

b) дивизимные алгоритмы, строящие гранулы *сверху вниз* путем дробления исходного множества данных, в результате чего количество гранул увеличивается.

Основные принципы построения подобных моделей данных изложены в работах [7,8]. В работах введены единые алгебраические модели представления пространства данных в виде покрытий выпуклыми элементами произвольной размерности (гранулами). Однако, введенные в [8] новые модели для гранулирования многомерных данных требуют изучения с точки зрения использования в конвейерных и мультиконвейерных ВС [1]. Рассмотрим два основных алгоритма гранулирования, имеющих линейную сложность в соответствии с (1).

Алгоритм 1. Агломеративный алгоритм гранулирования.

Входные данные: Множество значений данных $Data$.

Выходные данные: множество гранулированных данных $Granules$.

Шаг 1. Покрыть исходное множество данных произвольными непересекающимися подмножествами $SubsetData_i$ исходного декартова произведения.

Шаг 2. Инициализировать множество гранулированных данных $Granules = SubsetData$.

Шаг 3. Просматривая все подмножества $SubsetData_i$ множества $Granules$ произвести процедуру оптимизации гранулы.

Шаг 4. Вывести множество гранулированных данных $Granules$.

Агломеративный алгоритм основан на объединении исходных подмножества гранулирования декартова произведения областей определения данных [7]. Основным требованием для подмножеств является выполнение $SubsetData_i \cap SubsetData_j = \varnothing$, $\forall i, j$. Количество гранул покрытия зависит от требуемой точности решения задачи [6].

Алгоритм 2. Дивизимный алгоритм гранулирования.

Входные данные: Множество данных $Data$, значение меры информативности $Info$.

Выходные данные: множество гранулированных данных $Granules$.

Шаг 1. Инициализировать множество гранулированных данных $Granules = Data$.

Шаг 2. Выполнить для каждой гранулы Granules_i процедуру оптимизации.

Шаг 3. Вычислить меру информативности *Measure*(Granules_i) всех гранул, включенных во множество *Granules* .

Шаг 4. Если *Measure*(Granules_i) < *Info* , заменить текущую гранулу двумя новыми, полученными в результате деления гранулы Granules_i и вернуться к шагу 2.

Шаг 5. Вывести множество гранулированных данных *Granules* .

Общие граф-схемы алгоритмов, позволяющие оценить возможность их применения в конвейерных и мультиконвейерных системах, представлены на рис 1.

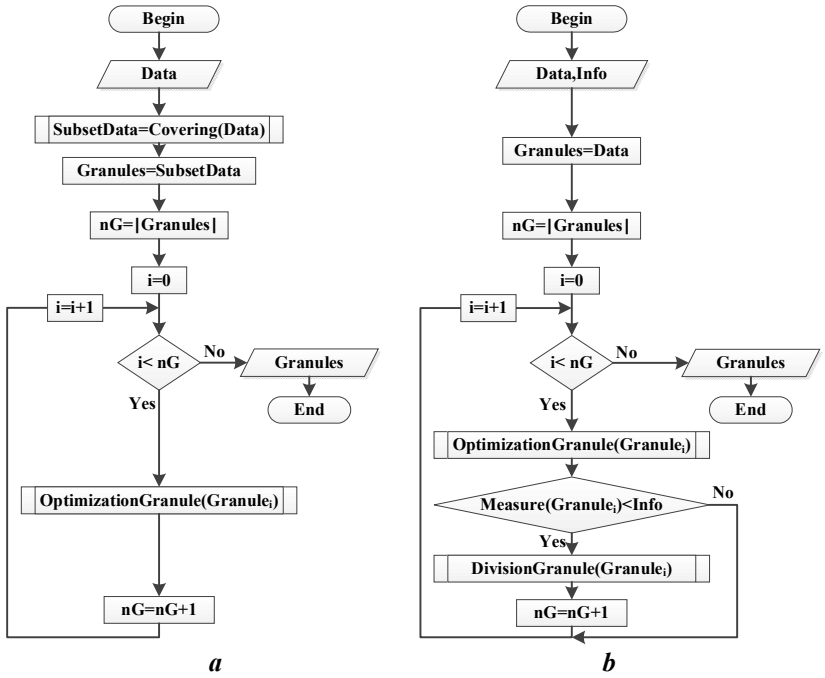


Рис. 1. Агломеративный a) и дивизимный b) алгоритмы грануляции данных с линейной сложностью для конвейерных ВС

Разработанный комплекс алгоритмов тестировался на различных классах черно-белых изображений. К недостаткам дивизимного алгоритма можно отнести значительное увеличивается время получения

гранулированных данных. Эту проблему можно решить, используя агломеративный алгоритм. Можно существенно снизить вычислительные затраты, но при этом количество гранул будет избыточным.

Заключение

Анализ результатов эксперимента показывает, что при применении дивизимного алгоритма гранулирования данных мы получаем более компактное гранулированное представление данных по сравнению с агломеративным алгоритмом. Наиболее это ощутимо при увеличении объема данных, представляющих интерес, когда необходимо увеличивать число разбиений или повышать значимость меры информативности [6].

Предложенные в работе алгоритмы могут быть реализованы в виде макрообъектов для реализации реконфигурируемых ВС на основе ПЛИС. Введенные алгоритмы описывают совокупность функциональных блоков, объединенных графической схемой коммутации для размещения в одной ПЛИС [1]. При этом возможно вложенное применение макрообъектов для реализации функциональных блоков, решающих задачи различных типов на основе единого грануляционного подхода [7].

1. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры.– Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
2. *Бутенков С.А.* Неметрический подход в задачах грануляции данных // Научные труды SWorld. – Иваново: Научный мир, 2015. – Вып. 4(41). – Т.2. – С. 91-99.
3. *Кривша Н.С., Кривша В.В., Бутенков С.А.* Эффективные алгоритмы грануляции данных на основе пространственной грануляции // Научные труды SWorld. – Иваново: Научный мир, 2015. – Вып. 4(41). – Т. 2. – С. 83-90.
4. *Хаггарти Р.* Дискретная математика для программистов. – М.: Техносфера, 2003. – 320 с.
5. *Мандель И. Д.* Кластерный анализ. – М.: Финансы и статистика, 1988. – 176 с.
6. *Бутенков С.А., Кривша В.В., Аль-Доуяни С.Х.С.* Построение системы нечетких отношений взаимного положения на декартовых гранулах // Сборник трудов Международной научно-технической конференции «Искусственные интеллектуальные системы» (IEEE AIS'06). – М.: ФИЗМАТЛИТ, 2006. – Т.2. – С. 99-105.
7. *Бутенков С.А.* Методы информационной грануляции в параллельных вычислениях // Материалы 3-й Всероссийской научно-технической конференции «СКТ-2014». – Ростов-на-Дону: Изд-во ЮФУ. – Т.1. – С. 99-104.

И.И. Кулагин, М.Г. Курносов

О СПЕКУЛЯТИВНОМ ВЫПОЛНЕНИИ КРИТИЧЕСКИХ СЕКЦИЙ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ

Сибирский государственный университет телекоммуникаций и информатики, г. Новосибирск,
ivan.i.kulagin@gmail.com, mkurnosov@gmail.com

Введение

При разработке параллельных программ для вычислительных систем с общей памятью неизбежно возникает необходимость защиты разделяемых ресурсов от возникновения состояния гонок за данными (data race). Для синхронизации доступа к разделяемым областям памяти активно используются примитивы синхронизации, основанные на механизме блокировок (семафоры, мьютексы и др.). Использование данных методов подразумевает создание в коде программы критических секций, выполнение которых возможно только одним потоком в каждый момент времени [1].

Практика показывает, что при одновременном выполнении одной критической секции потоки могут обращаться к непересекающимся областям памяти. В этом случае возникновение состояния гонки за данными не возникает, следовательно, блокировать выполнение потоков ненужно, так как использование блокировок увеличит накладные расходы на выполнение критической секции. На рис. 1 изображен пример данной ситуации. В критической секции содержится код добавления элемента в хеш-таблицу. При добавлении множеством потоков элементов с различными ключами *key* в хеш-таблицу *h*, то с большой долей вероятности хеш-коды *i* элементов будут отличаться и каждый поток будет выполнять добавление нового узла в отдельный список *h*[*i*]. Блокировка всей функции здесь является избыточной и неэффективной.

```
function hashtable_add(h, key, value)
    lock_acquire()
    i = hash(key)
    list_add_front(h[i], key, value)
    lock_release()
end function
```

Рис. 1. Добавление пары (key, value) в хеш-таблицу h

Таким образом, необходимы эффективные методы выполнения критических секций, позволяющие избежать избыточное блокирование выполнения потоков, которые не требуют глубокой переработки параллельной программы. Кроме того, данные методы должны обеспечить корректность программы – отсутствие состояний гонок за данными, взаимных блокировок (deadlock), ситуаций активных блокировок (livelock) и др.

В работе исследован метод спекулятивного выполнения критических секций с использованием возможностей программной транзакционной памяти (software transactional memory) для языков C/C++ в компиляторе GCC (библиотека libitm).

Программная транзакционная память

Программная транзакционная память (software transactional memory) – это подход к созданию потокобезопасных программ, основная идея которого заключается в защите области памяти от конкурентного доступа, а не участка кода, как в случае использования блокировок. В рамках программной транзакционной памяти программисту предоставляются языковые конструкции или API для формирования в программе транзакционных секций (transactional section) – участков кода, в которых осуществляется защита совместно используемых областей памяти. Выполнение потоками таких секций осуществляется без их блокирования. На среду выполнения (runtime) ложатся задачи по контролю за корректностью выполнения транзакций. Если во время выполнения транзакции другие потоки одновременно с ней не модифицировали защищенную область памяти, то транзакция считается корректной, и она фиксируется (commit). Если же два или более потока при выполнении транзакций обращаются к одной и той же области памяти и как минимум один из них выполняет операцию записи, то возникает конфликт (аналог состояния гонки данных). Для его разрешения выполнение одной или нескольких транзакций может быть либо приостановлено (до завершения конфликтующей транзакции), либо прервано, а все модифицированные ими (их потоками) области памяти приведены в исходное состояние (на момент старта транзакции) – отмена транзакции и восстановление (cancel and rollback).

На рис. 2 представлен пример создания транзакционной секции, в теле которой выполняется добавление элемента в хэш-таблицу множеством потоков. После выполнения тела транзакционной секции каждый поток приступит к выполнению кода, следующего за ней, в случае отсутствия конфликтов. В противном случае поток повторно будет выполнять транзакцию до тех пор, пока его транзакция не будет успешно зафиксирована.

```
/* Совместно используемая хеш-таблица */
hashtable_t *h;

/* Код потоков */
void *thread_start(void *arg) {
    struct data *d = (struct data *)arg;
    prepareData(d);

    /* Транзакционная секция */
    __transaction_atomic {
        /* Добавление элемента в хеш-таблицу */
        struct data *d = (struct data *)arg;
        hashtable_insert(h, d);
    }

    saveData(d);
    return NULL;
}
```

Рис. 2. Добавление пары (key, value) в хеш-таблицу h

Основными архитектурными решениями, определяющими поведение программной транзакционной памяти, являются политика обновления объектов в памяти, а также стратегия обнаружения конфликтов.

Политика обновления объектов в памяти определяет, когда изменения объектов в рамках транзакции будут записаны в память. Распространение получили две основные политики – ленивая и ранняя. Ленивая политика обновления объектов в памяти (lazy version management) откладывает все операции с объектами до момента фиксации транзакции. Все операции записываются в специальном журнале (redo log), который при фиксации используется для отложенного выполнения операций. Очевидно, что это замедляет операцию фиксации, но существенно упрощает процедуры ее отмены и восстановления. Примером реализаций ТП, использующих данную политику, являются RSTM-LLT [7] и RSTM-RingSW [8, 10].

Ранняя политика обновления (eager version management) предполагает, что все изменения объектов сразу записываются в память. В журнале отката (undo log) фиксируются все выполненные операции с памятью. Он используется для восстановления оригинального состояния модифицируемых участков памяти в случае возникновения конфликта. Эта политика характеризуется быстрым выполнением операции фиксации транзакции, но медленным выполнением процедуры ее отмены. Примерами реализаций, использующих раннюю политику обновления

данных, являются GCC (libitm), TinySTM [4], LSA-STM [5], Log-TM [9], RSTM [7] и др.

Момент времени, когда инициируется алгоритм обнаружения конфликта, определяется стратегией обнаружения конфликтов. При отложенной стратегии (lazy conflict detection) алгоритм обнаружения конфликтов запускается на этапе фиксации транзакции [10]. Недостатком этой стратегии является то, что временной интервал между возникновением конфликта и его обнаружением может быть достаточно большим. Эта стратегия используется в RSTM-LLT [7] и RSTM-RingSW [7, 8, 10].

Пессимистичная стратегия обнаружения конфликтов (eager conflict detection) запускает алгоритм их обнаружения при каждой операции обращения к памяти. Такой подход позволяет избежать недостатков отложенной стратегии, но может привести к значительным накладным расходам, а также, в некоторых случаях, может привести к увеличению числа откатов транзакций. Стратегия реализована в TinySTM [4], LSA-STM [5] и TL2 [11]. В компиляторе GCC (libitm) реализован комбинированный подход к обнаружению конфликтов – отложенная стратегия используется совместно с пессимистической.

Для обнаружения конфликтных операций требуется отслеживать изменения состояния используемых областей памяти. Информация о состоянии может соответствовать областям памяти различной степени гранулярности. Выбор гранулярности обнаружения конфликтов – один из ключевых моментов при реализации программной транзакционной памяти.

На сегодняшний день используются два уровня гранулярности: уровень программных объектов (object-based STM) и уровень слов памяти (word-based STM). Уровень программных объектов подразумевает отображение объектов модели памяти языка (объекты C++, Java, Scala) на метаданные runtime-библиотеки. При использовании уровня слов памяти осуществляется отображение блоков линейного адресного пространства процесса на метаданные. Метаданные хранятся в таблице, каждая строка которой соответствует объекту программы или области линейного адресного пространства процесса. В строке содержатся номер транзакции, выполняющей операцию чтения/записи памяти; номер версии отображаемых данных; их состояние и др. Модификация метаданных выполняется runtime-системой с помощью атомарных операций процессора.

Заключение

В работе предложено использовать программную транзакционную память для организации спекулятивного выполнения критических секций параллельных программ. Выполнен анализ основных архитектур-

ных аспектов реализации программной транзакционной памяти. Использование программной транзакционной памяти не требует существенных изменений параллельной программы, однако эффективное её применение ограничено. В частности, спекулятивное выполнение критических секций при помощи программной транзакционной памяти целесообразно, когда риск возникновения ситуации гонки за данными в защищаемом участке кода невелик.

1. *Herlihy M., Shavit N.* The Art of Multiprocessor Programming. Morgan Kaufmann PublishersInc. – San Francisco, CA, USA, 2008.
2. *Кузнецов С.Д.* Транзакционная память [HTML]. www.citforum.ru/programming/digest/transactional_me
3. *Shavit N., Touitou D.* Software Transactional Memory. In PODC'95: Proceedings of thefourteenth annual ACM symposium on Principles of distributed computing. – New York, NY, USA, Aug. 1995. ACM. – P. 204 – 213.
4. *Pascal Felber, Christof Fetzer, Patrick Marlier, and Torvald Riegel,* Time-based SoftwareTransactional Memory // IEEE Transactions on Parallel and Distributed Systems. – 2010. – Vol. 21. – Issue 12. – P. 1793 – 1807.
5. *Torvald Riegel, Pascal Felber, and Christof Fetzer.* A Lazy Snapshot Algorithm with EagerValidation // 20th International Symposium on Distributed Computing (DISC), 2006.
6. *Victor Luchango, Jens Maurer, Mark Moir.* Transactional memory for C++ [PDF]. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3718.pdf
7. Rochester Software Transactional Memory Runtime. Project web site [HTML]. www.cs.rochester.edu/research/synchronization/rstm/.
8. *Michael F. Spear, Luke Dalessandro, Virendra J. Marathe, and Michael L.* Scott. Acomprehensive strategy for contention management in software transactional memory // In PPoPP '09: Proc. 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. – 2009. – P. 141 – 150.
9. *Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill, and David A.* Wood.LogTM: Log-based transactional memory // In HPCA '06: Proc. 12th International Symposium on High-Performance Computer Architecture. – 2006. – P. 254 – 265.
10. *Michael F. Spear, Maged M. Michael, and Christoph von Praun.* RingSTM: scalabletransactions with a single atomic instruction // In SPAA '08: Proc. 20th Annual Symposium on Parallelism in Algorithms and Architectures. – 2008. –P. 275 – 284.
11. *Dave Dice, Ori Shalev, and Nir Shavit.* Transactional locking II // In DISC '06: Proc. 20th International Symposium on Distributed Computing. – Springer Verlag Lecture Notes in Computer Science. – 2006. – Vol. 4167. – P. 194 – 208.

И.И. Маркович, Н.О. Демьяненко

ЭФФЕКТИВНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ ОБРАТНОГО ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ ДЕЙСТВИТЕЛЬНЫХ СИГНАЛОВ

НКБ цифровой обработки сигналов ЮФУ, г. Таганрог,
marko@sfedu.ru; dsp@.sfedu.ru

В настоящее время с помощью суперкомпьютерных технологий, реализующих алгоритмы и методы цифровой пространственно-временной обработки шумоподобных сигналов, успешно решаются такие важнейшие задачи пассивной гидролокации, как обнаружение и классификация подводных и надводных объектов, определение их координат и параметров движения. При решении указанных задач широко используются стандартные процедуры быстрого преобразования Фурье (БПФ), позволяющие эффективно переводить комплексные сигналы из временной области в частотную и обратно.

Однако эффективность алгоритмов БПФ можно еще существенно повысить при вычислении прямого и обратного дискретного преобразования Фурье (ДПФ) действительных шумоподобных сигналов. Совмещенные алгоритмы, требующие меньшего количества операций для вычисления прямого ДПФ, описаны в работах [1–5].

Получим эффективный алгоритм вычисления обратного ДПФ (ОДПФ) вещественных последовательностей, используя алгоритмы прямого ДПФ.

Известен эффективный совмещенный алгоритм вычисления прямого ДПФ двух действительных сигналов длиной N с помощью одного БПФ размером N, позволяющий уменьшить требуемое количество базовых операций БПФ в 2 раза по сравнению со стандартным алгоритмом БПФ.

Для этого из двух вещественных последовательностей $x(nT)$ и $y(nT)$ формируют одну комплексную $z(nT)$ по следующему правилу:

$$z(nT) = x(nT) + jy(nT), \quad \text{где } n = 0, 1, 2, ..., N-1.$$

ДПФ искомых действительных последовательностей $X(k)$ и $Y(k)$ находят через ДПФ сформированной последовательности $Z(k)$ длиной N:

$$X(k) = \frac{Z(k) + Z^*(N-k)}{2}; \qquad Y(k) = \frac{Z(k) - Z^*(N-k)}{2j}.$$

Вычисления $X(k)$ и $Y(k)$ достаточно выполнить только для k = 0, 1, 2,...$N/2$, а остальные значения найти, используя свойства симметрии спектров действительных последовательностей.

Второй совмещенный алгоритм позволяет вычислить ДПФ одной вещественной последовательности $x(nT)$ длиной N с помощью применения процедуры БПФ размером $N/2$.

Для этого любую (действительную или комплексную), кратную двум, последовательность $x(nT)$ представляют в виде двух последовательностей $x_1(nT)$ и $x_2(nT)$ длиной $N/2$, сформированных из четных и нечетных отсчетов:

$$x_1(nT) = x(2nT), \quad \text{где} \quad n = 0, 1, 2, \ldots, N/2 - 1;$$

$$x_2(nT) = x[(2n+1)T], \quad \text{где } n = 0, 1, 2, \ldots, N/2 - 1.$$

Тогда ДПФ искомой последовательности $X(k)$ определяется через ДПФ $X_1(k)$ и $X_2(k)$ сформированных последовательностей:

$$X(k) = X_1(k) + X_2(k) e^{-j\frac{2\pi}{N}k}.$$

Последнее соотношение показывает связь ДПФ любой (действительной или комплексной) N-точечной последовательности с ДПФ двух $N/2$-точечных последовательностей, составленных из четных и нечетных отсчетов исходной последовательности.

Теперь полагая, что исходная последовательность $x(nT)$ является действительной, формируют новую комплексную последовательность длиной $N/2$ отсчетов из двух действительных последовательностей $x_1(nT)$ и $x_2(nT)$ по следующему правилу:

$$z_1(nT) = x_1(nT) + j\, x_2(nT), \quad \text{где} \quad n = 0, 1, 2, \ldots, N/2 - 1.$$

Тогда в соответствии с рассмотренным выше первым совмещенным алгоритмом вычисления ДПФ двух действительных последовательностей с помощью однократного использования процедуры БПФ для определения $Z_1(k)$ получают ДПФ исходных последовательностей длиной $N/2$:

$$X_1(k) = \frac{Z_1(k) + Z_1^*(\frac{N}{2} - k)}{2}; \qquad X_2(k) = \frac{Z_1(k) - Z_1^*\left(\frac{N}{2} - k\right)}{2j}.$$

Откуда следует алгоритм вычисления ДПФ N-точечной исходной действительной последовательности $x(nT)$ с помощью БПФ размером $N/2$ в удобной для практического использования форме:

$$X(k) = \frac{Z_1(k) + Z_1^*(\frac{N}{2} - k)}{2} + \frac{Z_1(k) - Z_1^*\left(\frac{N}{2} - k\right)}{2j} e^{-j\frac{2\pi}{N}k},$$

где $k = 0, 1, 2, \ldots, N/2$.

Значения $X(k)$ при других k находят, используя свойства симметрии спектров действительных последовательностей.

Эффективный алгоритм вычисления ОДПФ может быть получен непосредственно из последнего аналитического выражения для расчета прямого ДПФ.

Получим алгоритм вычисления ОДПФ действительных сигналов размером N с помощью стандартной процедуры ОБПФ размером $N/2$,

позволяющий существенно уменьшить требуемое количество вычислительных операций, по сравнению со стандартным алгоритмом.

Для вывода требуемых соотношений определим аналитические выражения для $X^*(k)$ и $X^*(N/2-k)$.

$$X^*(k) = \frac{Z_1^*(k) + Z_1(\frac{N}{2}-k)}{2} - \frac{Z_1^*(k) - Z_1\left(\frac{N}{2}-k\right)}{2j} e^{j\frac{2\pi}{N}k}.$$

$$X^*\left(\frac{N}{2}-k\right) = \frac{Z_1(k) + Z_1^*(\frac{N}{2}-k)}{2} - \frac{Z_1(k) - Z_1^*\left(\frac{N}{2}-k\right)}{2j} e^{-j\frac{2\pi}{N}k}.$$

Найдем разность и сумму уравнений для $X(k)$ и $X^*(N/2-k)$:

$$X(k) - X^*\left(\frac{N}{2}-k\right) = \frac{Z_1(k) - Z_1^*\left(\frac{N}{2}-k\right)}{j} e^{-j\frac{2\pi}{N}k}.$$

$$X(k) + X^*\left(\frac{N}{2}-k\right) = Z_1(k) + Z_1^*\left(\frac{N}{2}-k\right).$$

Выполним необходимые дополнительные преобразования:

$$Z_1(k) - Z_1{}^*\left(\frac{N}{2}-k\right) = j\left[X(k) - X^*\left(\frac{N}{2}-k\right)\right] e^{j\frac{2\pi}{N}k}.$$

$$Z_1(k) + Z_1{}^*\left(\frac{N}{2}-k\right) = X(k) + X^*\left(\frac{N}{2}-k\right).$$

Произведем сложение уравнений и получим аналитическое выражение для последовательности $Z_1(k)$ длиной $N/2$ отсчетов:

$$Z_1(k) = \frac{X(k) + X^*(\frac{N}{2}-k)}{2} - \frac{X(k) - X^*\left(\frac{N}{2}-k\right)}{2j} e^{j\frac{2\pi}{N}k},$$

где $k = 0, 1, 2, \ldots, N/2-1$.

Структурная схема алгоритма вычисления ОДПФ действительных сигналов представлена на рисунке.

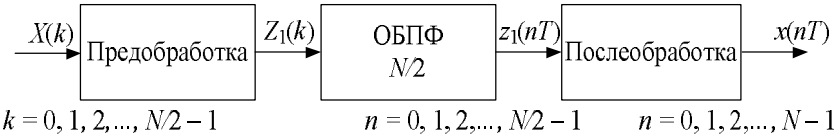
Алгоритм работает следующим образом.

На этапе предобработки формируется комплексная последовательность $Z_1(k)$.

Далее выполняется стандартное ОБПФ сформированной последовательности, результатом которого является комплексная последовательность $z_1(nT)$:

$$z_1(nT) = \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} Z_1(k) e^{j\frac{2\pi}{N/2}nk},$$

где $n = 0, 1, 2, \ldots, N/2-1$.



Структурная схема алгоритма вычисления обратного ДПФ действительного сигнала

Для восстановления исходного сигнала $x(nT)$, $n = 0,\ 1,\ \ldots,\ N-1$ служит этап послеобработки, который заключается в разделении реальных и мнимых частей полученных комплексных отсчетов:

$$x(2nT) = \mathrm{Re}\, z_1(nT), \quad \text{где} \quad n = 0, 1, \ldots,\ N/2 - 1;$$

$$x[(2n+1)T] = \mathrm{Im} z_1(nT), \quad \text{где } n = 0, 1, \ldots,\ N/2 - 1.$$

Эффективность данного алгоритма по количеству комплексных умножений определяется так же, как и второго совмещенного алгоритма вычисления прямого ДПФ, по формуле

$$Э_{к.у.} = \frac{\frac{N}{2}\log_2 N}{\frac{N}{4}\log_2 \frac{N}{2} + \frac{N}{2}} = \frac{2\log_2 N}{\log_2 \frac{N}{2} + 2}$$

и меньше эффективности первого совмещенного алгоритма вычисления ДПФ действительного сигнала, равной двум, так как выполняется дополнительно $N/2$ умножений на экспоненту.

1. *Бендат Дж., Пирсол А.* Прикладной анализ случайных данных. – М.: Мир, 1989. – 540 с.
2. *Ярославский Л.П.* Введение в цифровую обработку изображений. – М.: Сов. радио, 1979. – 312 с.
3. *Ричард Лайонс.* Цифровая обработка сигналов. – 2-е изд. / пер. с англ. – М.: ООО «Бином-пресс», 2006. – 656 с.
4. *Маркович И.И.* Цифровая обработка сигналов в системах и устройствах: монография / Южный федеральный университет. – Ростов-на-Дону: Изд-во ЮФУ, 2012. – 236 с.
5. *Маркович И.И.* Алгоритмы вычисления спектров сигналов в системах обработки изображений // Научно-теоретический журнал «Искусственный интеллект». – Донецк: Институт проблем искусственного интеллекта Министерства образования и науки Украины и НАН Украины, 2012. – № 3. – С. 172 – 177.

Е.О. Мовчан, В.В. Гетманский, А.Е. Андреев

ВЕКТОРИЗАЦИЯ АЛГОРИТМОВ ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИКИ СИСТЕМ ТЕЛ

Волгоградский государственный технический университет,
г. Волгоград,
verborum123@mail.ru

Состояние вопроса моделирования динамического напряженно-деформируемого состояния (НДС)

При проектировании машин актуальной является задача определения динамических напряжений в деталях, подверженных наибольшим динамическим нагрузкам. Одним из методов решения является метод дискретных элементов [1]. Метод сводится к моделированию динамики систем тел без жестких связей, образующих регулярную ортогональную сетку в расчетной области, соответствующей геометрии деталей. Исследования по ускорению этого метода [1-4] показали, что распараллеливание алгоритмов расчета под системы с общей и распределенной памятью дают максимальное ускорение в 10 раз из-за ограничений масштабируемости алгоритма. Дальнейшее направление по ускорению метода – это оптимизация кода. Для этого предлагается использовать векторизацию, так как векторные регистры в современных процессорах определяют их производительность, а их разрядность постепенно растет.

В предлагаемом методе для расчета деформаций в упругом теле используется аппроксимация тела дискретными твердотельными элементами, при этом расчетная область представляет собой регулярную ортогональную сетку в 3D-пространстве. Каждый дискретный элемент имеет форму куба, 6 степеней свободы и от 1 до 6 связей с соседними по граням элементами. Из-за одинаковой формы дискретных элементов значительно упрощается составление системы уравнений движения в физических координатах, так как матрица инерции является диагональной, а все элементы с точностью до количества связей описываются одинаковыми обыкновенными дифференциальными уравнениями вида

$$\mathrm{M}\ddot{\mathrm{y}} = \mathrm{q}\left(\dot{\mathrm{y}},\mathrm{y},t\right) - \mathrm{Ma}\left(t\right) + \mathrm{s}\left(\dot{\mathrm{y}},\mathrm{y}\right), \qquad (1)$$

где y – координаты дискретных элементов; M – матрица инерции; a – вектор, составленный из компонент вектора ускорений твердого тела, входящего в полную динамическую модель; $\mathrm{s}\left(\dot{\mathrm{y}},\mathrm{y}\right)$ – силы стабилизации.

Функция $\mathrm{q}\left(\dot{\mathrm{y}},\mathrm{y},t\right)$ описывает силы между дискретными элементами. Случайные возмущения, передаваемые от динамической модели, могут вызывать смещение дискретных элементов, поэтому для их при-

вязки к опорному телу введены силы стабилизации $s(\dot{y}, y)$. Уравнение (1) записано в системе координат опорного тела, поэтому правая часть (1) также содержит силы инерции, рассчитываемые на основе ускорений опорного тела из динамической модели.

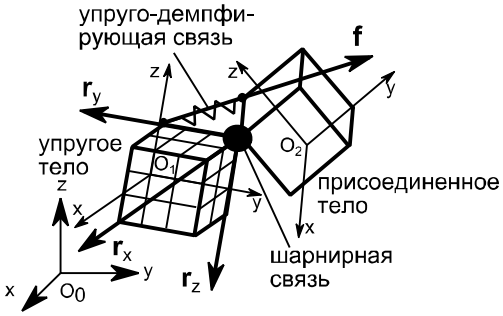


Рис. 1. Связи упругого тела с присоединенным телом

Деформируемое тело привязано к твердому телу, которое связано с другими телами в системы тел. Как показано на рис. 1, связи могут быть двух типов: упругодемпфирующая $f_1 = f$ и шарнирная (жесткая связь) $f_2 = (r_x, r_y, r_z)$. Для уравнений, описывающих граничные узлы, в правые части добавляются силы реакций, возникающие в связях. Уравнение для граничного элемента с использованием равномерного распределения внешних сил по граничным узлам имеет вид

$$M_i \ddot{y}_i = f(t)/|F| + q_i(\dot{y}, y, t) - M_i a_i(t) + s_i(\dot{y}, y), \; i \in F, \qquad (2)$$

где F – множество индексов узлов, входящих в границу, к которой приложена сила f.

Таким образом, данные, полученные из динамической модели, позволяют определить силы инерции и силы, возникающие в связях.

Уравнение (1) интегрируется явным методом Рунге – Кутты 4 порядка. При обновлении результатов расчета динамической модели происходит обновление правых частей для решателя НДС. Из уравнения 2 видно, что основные вычисления требуют выполнения матрично-векторных операций, векторизацию можно применить при вычислении правых частей, кроме того, ее можно использовать для ускорения численного интегрирования.

Описание способа организации памяти

Одной из причин замедления работы программ является использование невыровненной памяти. Данная ситуация возникает, когда блок данных выходит за границы так называемого машинного слова, либо является сдвинутым относительно них.

Некоторые процессоры не поддерживают операции над невыровненными данными. Другие поддерживают, но обращаются к невыровненным данным дольше, нежели к данным, находящимся внутри целого машинного слова в памяти.

Чтобы ускорить доступ к памяти, в программе использовано выровненное выделение памяти под массивы данных, для чего используются команды вида:

```
_initX = (double*)_aligned_malloc(varSize, alignment);
```

Кроме того, необходимо подготовить данные для загрузки в векторные регистры. Исходные матрицы и векторы имеют размерность 3x3 или 1x3, соответственно, и при загрузке напрямую необходимо было бы существенно модифицировать алгоритм вычислений. Для загрузки в регистр размерность должна быть 3x4 или 1x4, чтобы загружать матрицы и векторы по строкам.

Такая размерность достигается путем добавления в конце каждой строки матрицы и каждого вектора дополнительного фиктивного элемента. Чтобы значение этого элемента не влияло на результаты операций, он принят равным 0. После этого строка матрицы либо целиком помещается в векторный регистр (для AVX), либо занимает два полных регистра меньшего размера (для SSE).

Векторизация численного интегрирования

Алгоритм численного интегрирования по явному методу Рунге-Кутты 4-го порядка точности предполагает вычисление следующего приближения в 4 шага. На каждом шаге вычисляются коэффициенты, которые используются на следующих шагах, что требует обновления данных в памяти после каждого шага.

Рассмотрим векторизацию данного метода на примере 1-го шага вычислений. Невекторизованный фрагмент кода выглядит так:

```
for (int j = 0; j < _nVariables; j ++) {
        _hDDX1[j] = _varDDX[j] * _timeStep;
        _varX[j] += _varDX[j] * _timeStep2;
        _varDX[j] += _hDDX1[j] * 0.5;                    }
```

Этот же фрагмент с помощью SSE-интринсиков можно записать:

```
Xtmp = _mm_load_pd(_varX + j);              // Загрузка данных в регистры
DXtmp = _mm_load_pd(_varDX + j);
DDXtmp = _mm_load_pd(_varDDX + j);
hDDX1 = _mm_mul_pd(DDXtmp, timeStep); // Выполнение операций
_mm_store_pd(_hDDX1 + j, hDDX1);          // Выгрузка данных в память
tmp = _mm_add_pd(_mm_mul_pd(DXtmp, timeStep2), Xtmp);
_mm_store_pd(_varX + j, tmp);
tmp = _mm_add_pd(_mm_mul_pd(hDDX1, constant), DXtmp);
_mm_store_pd(_varDX + j, tmp);          }
```

В идеале ускорение вычисления данной функции должно быть равно размеру использованных векторных регистров – 4 раза для AVX и 2 для SSE. Из-за большого числа операций загрузки и выгрузки данных достигаемый эффект не так значителен – 2,5 и 1,5 раза соответственно.

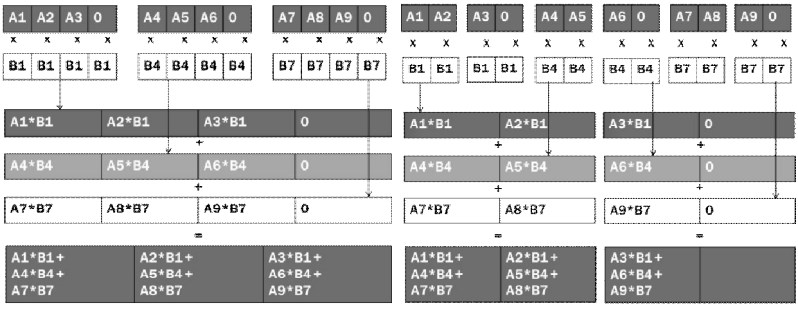
Векторизованный алгоритм вычисления правых частей

Наибольшие временные затраты при расчете НДС приходятся на вычисление правых частей. Все операции в данной функции являются матричными или векторными по выровненной схеме, описанной выше.

Рассмотрим векторизацию матричных операций на примере транспонированного перемножения матриц:

MathHelpers::Mat3x4 matA21 = matA02.Tmul(matA01);

Для векторизации с помощью интринсиков применяется схема, представленная на рис. 2. На схеме приведен пример вычисления 1-й строки результирующей матрицы для SSE- и AVX-инструкций.



а *б*

Рис. 2. Схема векторизованного перемножения матриц: а – с помощью инструкций AVX, б – с помощью инструкций SSE

Для данной функции максимальное ускорение ограничивается размерностью матриц – 3 раза, и это ускорение обуславливает общее ускорение программы, так как операция является самой трудоемкой.

Результаты по ускорению за счет оптимизации

Результаты ускорения работы векторизованного решателя на примере расчета модели балки представлены на рис. 3.

Векторизация кода решателя дала следующие результаты:

- среднее ускорение в 1,4 раза с помощью автовекторизации;
- среднее ускорение в 2,4 раза на SSE;
- среднее ускорение в 2,8 раза на AXV;
- увеличение ускорения за счет FMA-инструкций до 3 раз.

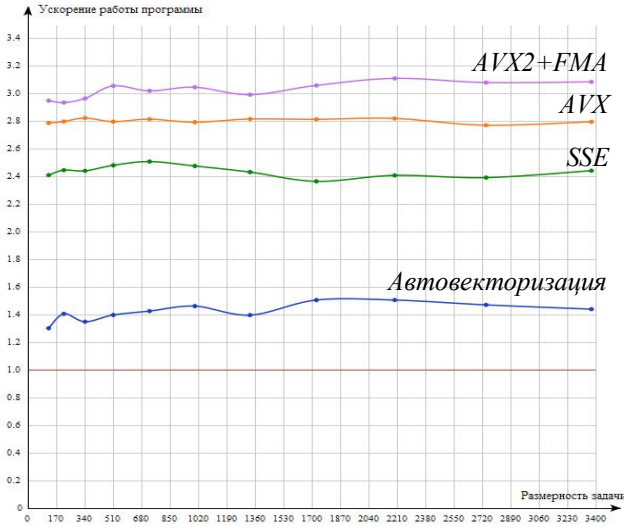


Рис. 3. Ускорение решателя НДС с помощью векторизации

Дальнейшая оптимизация алгоритмов для высокопроизводительных кластеров возможна за счет векторизации под новые архитектуры, например, поддерживающие AXV-512, и переноса кода пространственных преобразований для определения положения тел на графические вычислители и сопроцессоры.

1. *Гетманский В.В., Горобцов А.С., Измайлов Т.Д.* Распараллеливание расчёта напряжённо-деформированного состояния тела в многотельной модели методом декомпозиции расчётной области // Известия ВолгГТУ. – Сер. "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". – Вып. 16. № 8 (111). – Волгоград: Изд-во ВолгГТУ, 2013. – С. 5 – 10.
2. *Getmanskiy V.V., Gorobtsov A.S., Sergeev E.S., Ismailov T.D., Shapovalov O.V.* Concurrent simulation of multibody systems coupled with stress-strain and heat transfer solvers// Journal of Computational Science. – 2012. – No 3(6). – P. 492 – 497.
3. *Сергеев Е.С., Гетманский В.В., Горобцов А.С.* Перенос системы многотельной динамики на вычислительный кластер // Научно-технические ведомости Санкт-Петербургского гос. политехн. ун-та. – 2010. – Вып. 101. – С. 93-99.
4. *Горобцов А.С., Гетманский В.В., Андреев А.Е., Doan D.T.* Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings / ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015. – P. 379 - 91.

О.А. Назаркин, П.В. Сараев

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОГО ОБУЧЕНИЯ АНСАМБЛЕЙ АППРОКСИМАТОРОВ НА ОСНОВЕ НЕНОРМАЛИЗОВАННОГО ВАРИАНТА МОДЕЛЕЙ ANFIS [*]

Липецкий государственный технический университет, г. Липецк,
nazarkino@mail.ru, psaraev@yandex.ru

Построение ненормализованного варианта модели ANFIS с гауссовыми функциями

Гибридные нейронечеткие модели [1] являются удобным и эффективным инструментом извлечения знаний о сложных процессах. Они совмещают высокое качество аппроксимации сложных функций с возможностью удобной интерпретации структуры и параметров моделей. Нейроструктурные системы ANFIS [2] с количеством правил R, векторным аргументом $\mathbf{x}$ размерности M и скалярным выходом определяются кортежем настраиваемых параметров $p=\{a_{rm},q_{rm},k_{rm},b_r\}$, $r=1..R$, $m=1..M$. Модели этого класса обычно используются в нормализованной форме (1):

$$A_p(\mathrm{x})=\sum_{r=1}^{R}\frac{\prod_{m=1}^{M}\mu_{rm}(x_m)}{\sum_{j=1}^{R}\prod_{m=1}^{M}\mu_{jm}(x_m)}L_r(\mathrm{x})=\sum_{r=1}^{R}\frac{e^{-\sum_{m=1}^{M}\left(\frac{x_m-a_{rm}}{q_{rm}}\right)^2}}{\sum_{j=1}^{R}e^{-\sum_{m=1}^{M}\left(\frac{x_m-a_{jm}}{q_{jm}}\right)^2}}\left(b_r+\sum_{m=1}^{M}k_{rm}x_m\right). \quad (1)$$

За нормализацию отвечает компонент (2):

$$\rho(\mathrm{x})=\sum_{r=1}^{R}\prod_{m=1}^{M}\mu_{jm}(x_m)=\sum_{r=1}^{R}e^{-\sum_{m=1}^{M}\left(\frac{x_m-a_{jm}}{q_{jm}}\right)^2},\ \rho(\mathrm{x})>0. \quad (2)$$

Модели похожего по структуре класса RBFN (Radial Basis Function Networks) применяются как в нормализованной, так и в ненормализованной форме. Ненормализованная форма проще по структуре, из-за чего менее требовательна к вычислительным ресурсам при обучении.

Целью исследования является построение нейроструктурных аппроксиматоров на основе ненормализованного варианта (3) моделей класса ANFIS с гауссовыми функциями:

$$U_p(\mathrm{x})=\rho(\mathrm{x})A_p(\mathrm{x})=\sum_{r=1}^{R}\prod_{m=1}^{M}\mu_{rm}(x_m)L_r(\mathrm{x}). \quad (3)$$

[*] Исследование выполнено при финансовой поддержке РФФИ и Администрации Липецкой области в рамках научного проекта № *14-47-03611*-р_центр_а

Процедура обучения модели, т.е. поиск оптимальных значений параметров p_T^* на обучающем множестве $T=\{\mathbf{x}_i, f(\mathbf{x}_i)\}$ размерности N является стандартной задачей метода наименьших квадратов (4):

$$p_S^* = \arg\min_P F(p),\ F(p)=\sum_{i=1}^{N}\left[U_p(\mathrm{x}_i)-f(\mathrm{x}_i)\right]^2=\sum_{i=1}^{N} err_i^2\ , \qquad (4)$$

$i=1..N$, err_i^2 – ошибка модели на образце $\mathbf{x}_i \in T$.

Введем дополнительные обозначения $P_{ri}=\prod_{m=1}^{M}\mu_{rm}(x_{im})err_i$, $D_{ri}=P_{ri}L_r(\mathbf{x}_i)$, тогда частные производные (5) функционала качества

$$\frac{\partial F(p)}{\partial a_{rm}}=\frac{4}{q_{rm}^2}\sum_{i=1}^{N}(x_{im}-a_{rm})D_{ri}\ ;\ \frac{\partial F(p)}{\partial q_{rm}}=\frac{4}{q_{rm}^3}\sum_{i=1}^{N}(x_{im}-a_{rm})^2 D_{ri}\ ;$$

$$\frac{\partial F(p)}{\partial k_{rm}}=2\sum_{i=1}^{N}x_{im}P_{ri}\ ;\ \frac{\partial F(p)}{\partial b_r}=2\sum_{i=1}^{N}P_{ri}\ . \qquad (5)$$

Выражения (5) значительно проще аналогичных для нормализованной модели ANFIS [3]. Следовательно, процесс обучения ненормализованной модели $U_p(\mathbf{x})$ с использованием квазиньютоновских методов оптимизации, таких как L-BFGS, требует меньших вычислительных ресурсов.

Аддитивный метод построения аппроксиматоров с параллельным обучением

Еще одним основанием для применения ненормализованного варианта ANFIS является удобство агрегирования моделей на основе концепции улучшающего объединения (bagging, bootstrap aggregating). Так можно получить композиционную модель достаточно сложных аппроксиматоров на основе нескольких простых моделей ANFIS. При этом композиция моделей в ансамбле строится как сумма выходов нескольких ненормализованных моделей для $\mathbf{x}$ одной размерности (6):

$$\sum_{k=1}^{K}U_k(\mathrm{x})=\sum_{k=1}^{K}\sum_{j=1}^{R_k}\prod_{m=1}^{M}\mu_{jkm}(x_m)L_{jk}(\mathrm{x})=\sum_{r=1}^{R}\prod_{m=1}^{M}\mu_{rm}(x_m)L_r(\mathrm{x}), \qquad (6)$$

$$R=\sum_{k=1}^{K}R_k\ ,\ r=1...R\ .$$

Ансамбль (6) агрегирует информацию из базовых моделей $U_k(\mathbf{x})$, обучение которых можно выполнять параллельно (раздельное независимое обучение) с разными стартовыми значениями параметров, что ускоряет процедуру без какой-либо модификации применяемых алгоритмов.

Отличие параллельного обучения по этой схеме от обычного мультистарта заключается в том, что последний позволяет выбрать один вариант модели среди множества сопоставимых с ним по сложности. Если мультистарт запускает поиск простых моделей, то и лучший вариант тоже будет *простым*. Предлагаемая методика собирает из простых составляющих *сложный* вариант. В терминологии ANFIS сложность соответствует количеству правил в модели.

Принципы реализации распределенной высокопроизводительной системы построения моделей

Самой требовательной к вычислительным ресурсам в рассматриваемом классе задач является процедура обучения нейроструктурных моделей на основе квазиньютоновских алгоритмов многомерной оптимизации. Ключевым фактором, определяющим использование описанных выше методов построения аппроксиматоров и классификаторов на основе ненормализованных моделей ANFIS, является возможность удобной высокоуровневой декомпозиции алгоритмов обучения на параллельные задачи.

Предпосылки, изложенные выше, позволяют объединять результаты всех параллельных ветвей обучения в единую модель без потери информации. Такая схема хорошо подходит как для параллельных вычислений на одном узле, так и для распределенных вычислений на нескольких узлах. Следовательно, при наличии единого центра, осуществляющего сбор частных обученных моделей, можно обеспечивать распределенное построение одного варианта аппроксиматора (с повышением производительности и отсутствием ограничений по масштабированию).

Широкие возможности по организации высокопроизводительных вычислений предоставляет технология GRID. В [4] рассмотрены некоторые варианты проектирования высокопроизводительной системы нейроструктурного моделирования на основе распределенной схемы вычислений. В нашей работе применяется следующий вариант: в рамках единой сети (как правило, LAN, но может осуществляться и развертывание в Интернет) объединяются сервер СУБД, web-сервер обслуживания распределенных транзакций, а также несколько рабочих станций, оснащенных многоядерными процессорами. В качестве СУБД применяется MongoDB, предназначенная для эффективного доступа к объектно-ориентированным блокам информации (JSON-документам). Web-сервер обслуживания распределенных транзакций реализован на платформе Node.js, предназначенной для высокопроизводительной обработки web-запросов (в том числе для приложений реального времени) и имеющей эффективные средства взаимодействия с MongoDB. Клиентское программное обеспечение рабочих узлов реализовано на языке JavaScript и

может выполняться в контексте любого web-браузера, поддерживающего интерфейс параллельного программирования Web Workers.

Такая организация обладает следующими достоинствами: весь стек программного обеспечения реализован на едином языке (JavaScript, серверный и браузерный контекст), JSON-ориентированная система хранения данных не требует дополнительных объектно-реляционных преобразований, принципиально кроссплатформенный браузерный контекст параллельных вычислений позволяет задействовать процессорные ресурсы любых современных рабочих станций, с практически неограниченными возможностями масштабирования.

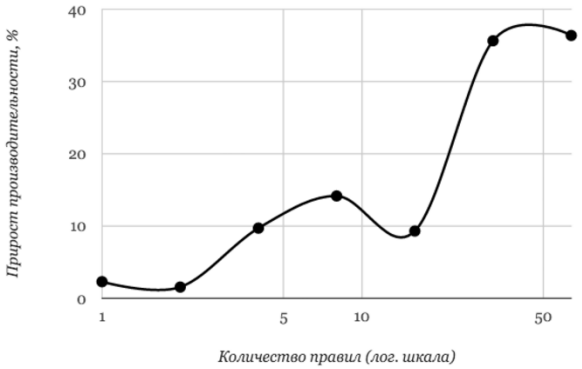
При необходимости в систему распределенных вычислений в качестве рабочих узлов могут быть включены даже мобильные устройства с многоядерными процессорами, парк которых является в настоящее время достаточно многочисленным; единственным ограничением при этом является скорость обмена данными по беспроводным каналам (Wi-Fi). Специфика решаемых задач – машинное обучение с высокоуровневой декомпозицией на параллельные операции – не предъявляет высоких требований к этой характеристике, так как длительность вычислительной фазы (работа алгоритма L-BFGS многомерной оптимизации) многократно превышает длительность фазы обмена данными (загрузка обучающих выборок или сохранение массивов параметров моделей).

Режим работы пользователей с системой предполагает, что с любой рабочей станции можно зайти на web-сайт системы, доступный в интрасети или в Интернет. При этом пользователем выбирается тип станции: рабочий узел или супервизор операций. Если выбран тип «рабочий узел», то web-сервер обслуживания распределенных транзакций возвращает браузеру страницу со специальным скриптом, циклически выполняющим фоновые вычисления и отправляющим web-серверу их результаты; пользователь имеет возможность продолжать свою привычную деятельность на данном компьютере. Если выбран тип «супервизор операций», пользователь получает интерактивную web-страницу, на которой в реальном времени отображаются все сведения о работе распределенной системы вычислений: список текущих задач по обучению моделей, найденные оптимальные параметры моделей для аппроксимации заданных функций и многие другие показатели.

Экспериментальные результаты

Практическое применение ненормализованного варианта ANFIS с раздельным обучением (6) демонстрирует прирост производительности обучения до 40 % по отношению к нормализованному варианту (см. рисунок), что можно считать существенным результатом, достигнутым исключительно за счет упрощения структуры математической модели

без какой-либо дополнительной оптимизации программного обеспечения. В проведенных экспериментах было зафиксировано также повышение качества аппроксимации ансамбля моделей: объединенная модель демонстрирует ошибку обобщения на 5 – 8 % ниже средней по частным моделям.



Прирост производительности по отношению к нормализованному варианту модели ANFIS

Уровень снижения ошибки зависит больше от свойств исследуемого процесса, в то время как зависимость от количества правил в частных моделях слабая. Следовательно, нет необходимости (длительно) обучать сложные частные модели, гораздо эффективнее объединять несколько параллельно обучающихся простых моделей в одну суммарную.

1. *Сараев П.В.* Развитие нейросетевого моделирования сложных систем на основе нейроструктурного подхода // Вести вузов Черноземья, 2012. – №2(28). – С. 30 – 35.
2. *Jang J.S.R.* ANFIS: Adaptive-Network-Based Fuzzy Inference System // IEEE Transactions on Systems, Man and Cybernetics, 1993. – Vol. 23. – №3. – P. 665 – 685.
3. *Назаркин О.А., Сараев П.В., Журавлева М.Г., Алексеев В.А.* OpenCL-реализация нейронечеткой модели ANFIS // Системы управления и информационные технологии, 2015. – №2(60). – С. 15 – 20.
4. *Saraev P.V., Domashnev P.A., Zhuravlyova M.G., Nazarkin O.A.* Design and implementation notes on heterogeneous distributed computations for research prototyping of neuro-structural models // Вести высших учебных заведений Черноземья. – 2015. – №2. – С.48 – 53.

Ю.И. Нечаев

СОВРЕМЕННАЯ КОМПЬЮТЕРНАЯ МАТЕМАТИКА И ПАРАДИГМЫ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ ПРИ РЕАЛИЗАЦИИ ДИНАМИЧЕСКОЙ ТЕОРИИ КАТАСТРОФ

Санкт-Петербургский государственный морской технический университет, г. Санкт-Петербург, nechaev@ifmo.mail.ru

Проблема неопределенности и парадигмы высокопроизводительных вычислений

Рассмотрим проблему реализации методов современной компьютерной математики при параллельной обработке информации в мультипроцессорной вычислительной среде [1 – 7] (рис.1). Задача оценки параметров пространства поведения и управления динамической теории катастроф содержат неопределенности при взаимодействии динамического объекта (ДО) с внешней средой на основе данных измерений.

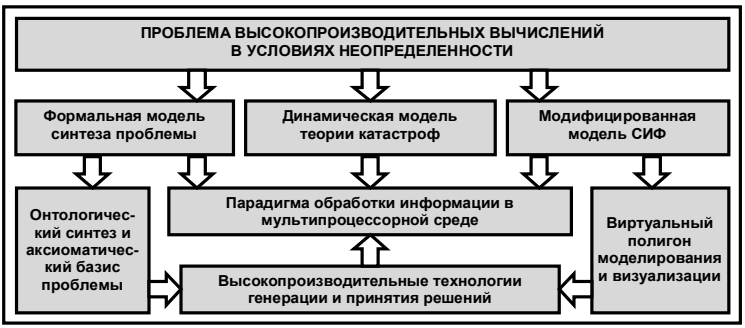


Рис.1. Концептуальная модель высокопроизводительных вычислений в условиях неопределенности

Совокупность неопределенностей разделяют на две группы: неопределенности в исходных данных и в процедурах моделирования. К первой группе относятся: структурная неопределенность при неполной информации связей «вход – выход» исследуемой модели; стохастическая неопределенность, определяемая уровнем шума в выборке данных, и информационная неопределенность, зависящая от качества выборки и числа измерений. Вторая группа включает: функциональную неопределенность, возникающую при выборе базисного набора функций; пара-

метрическую неопределенность, определяющую выбор алгоритмов решения задачи идентификации, аппроксимации и прогноза, и критериальную неопределенность, связанную с выбором критериев и методов решения задач структурной идентификации.

При формализации неопределенности среды функционирования динамической модели катастроф с учетом неопределенностей в исходных данных и процедурах моделирования приложений компьютерной математики можно выделить три среды моделирования и визуализации текущей ситуации, которые используются при оценке адекватности математических моделей: *слабая неопределенность,* определяемая сложностью формализации динамики взаимодействия с учетом действующих факторов и критериального базиса; *значительная неопределенность* в сложных условиях взаимодействия и реальной картины поведения ДО при различном уровне внешних возмущений; *полная неопределенность,* обусловленная отсутствием модели взаимодействия и представляющая собой исследование текущей ситуации на основе гипотез и упрощающих предположений [1] – [3].

Оценка адекватности функции интерпретации поведения ДО осуществляется путем сопоставления текущей M[S(Cur)] и эталонной M[S(Prec)] моделей. В условиях значительной неопределенности используется нечеткий логический базис: модель функции интерпретации считается адекватной выделенному прецеденту при условии выполнения критерия адекватности в виде

$$CR\{M[S(Cur)], M[S(\Pr ec)]\} \in \mu(tr), \qquad (1)$$

где μ(tr) – функция принадлежности, определяющая значения порога нечеткого равенства ситуаций $\mu(tr) \in [0{,}8;\ 1{,}0]$.

Таким образом, модель (1) обеспечивает выполнение основной задачи компьютерной математики – создание вычислительной среды виртуального моделирования процессов, протекающих параллельно и обеспечивающих организацию информационного обмена (синхронизацию), а также механизмы генерации и принятия решений в сложной динамической среде. Направления исследований в задачах динамики в условиях неопределенности связаны с формированием множества альтернатив и выбором решения на основе концепции конкуренции, а также ансамблевого моделирования, которое рассматривается как управление совокупностью альтернативных решений.

Интерпретация поведения в условиях непрерывного изменения динамики объекта и внешней среды

Структура моделей экстренных вычислений [6] в задачах компьютерной математики при интерпретации поведения ДО на основе интеллектуальных технологий (ИТ) представляет собой отображение $f : U \rightarrow$

U* интегрированной модели контроля текущей ситуации [3] в виде направленного множества:

$$U(t_0, t_k) \Rightarrow U^*(R_1, R_2, R_3)[f_t \in \Omega], \quad (2)$$

где $U(t_0, t_k)$ – управление на интервале реализации; $U^*(R_1, R_2, R_3)$ – функция интерпретации режимов функционирования ИС, формирующая управление в зависимости от сложности и неопределенности контролируемой ситуации; Ω – область допустимых значений параметров взаимодействия.

Функции интерпретации для выделенных режимов R_1, R_2, R_3 определяется кортежем:

$$U^*(R_1, R_2, R_3) = \langle U_1(ST, MF, ANN), U_2(ST, NF, E(ANN)), U_3(PR) \rangle, \quad (3)$$

где $U_1(\bullet)$, $U_2(\bullet)$, $U_3(\bullet)$ – структуры управления, реализующие конкурирующие вычислительные технологии на основе стандартной (ST), нечеткой (матрица MF), нейросетевой (ANN), нейронечеткой (NF), нейросетевого ансамбля E(ANN) и модели вывода по прецеденту PR.

На основе выражений (2), (3) осуществляется построение системы контроля поведения ДО, обеспечивающей реализацию механизма принятия решений. В зависимости от особенностей взаимодействия, определяемых выделенными режимами движения, используются различные структуры системы. Наиболее простая структура реализуется на базе нечеткой модели с коррекцией правил и обеспечивает режим R_1 обработки информации в условиях слабой неопределенности. Расширенная структура (режим R_2) предназначена для функционирования в условиях значительной неопределенности и дополняется блоком, содержащим управляющий контроллер. В особо сложных ситуациях при полной неопределенности (режим R_3) осуществляется логический вывод по прецеденту [2] с соответствующей реализацией динамической картины взаимодействия. Матрицы нечетких логических моделей адаптируются к изменяющимся внешним условиям, а нейросетевые ансамбли – распознают «шаблоны» (причинно-следственные отношения) каждого контролируемого процесса. Такой непрерывный процесс самообучения позволяет накапливать информацию о динамике взаимодействия и предсказывать отклонения параметров от режимных (допустимых) значений.

Теоретическая база современной компьютерной математики при моделировании динамических ситуаций формируется на основе эффективного сочетания накопленной системы знаний с новыми подходами и парадигмами ИТ. Процедуры экстренных вычислений, обеспечивающие функционирование динамической базы знаний, содержат большой объем экспериментальных исследований. Методы и модели, реализующие

генерацию нестандартных ситуаций и их интерпретацию при обработке измерительной информации, находят применение в задачах предсказания эволюции динамических характеристик и временных интервалов, определяющих критические условия, исходя из обеспечения безопасности ДО. Методы оценки риска позволяют организовать интерфейс оператора при контроле опасности нестандартных ситуаций, а также прикладных проблем управления и принятия решений (анализ альтернатив, распознавание экстремальных ситуаций и др.).

Методы современной компьютерной математики в интеллектуальных системах новых поколений

При разработке бортовых ИС новых поколений используются ИТ в рамках GRID-систем [3], технологий «облачных» вычислений (CLOUD-технологии [4]) и СОА [5] (рис. 2). Принципиально важным является переход от формальных методов интерпретации динамических ситуаций к практическим результатам представления состояний систем, полученных на основе современной компьютерной математики и динамической теории катастроф [3 – 5].

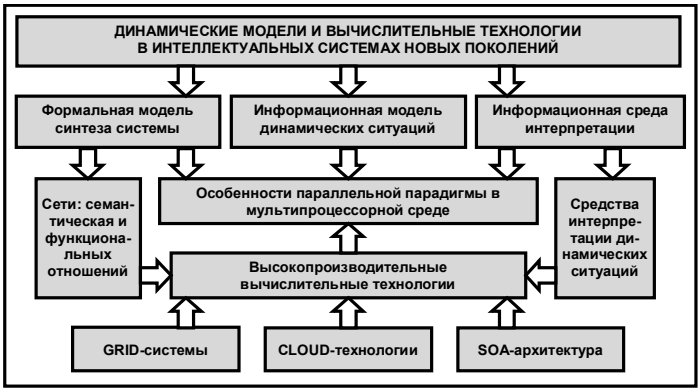


Рис. 2. Перспективы развития ИТ в бортовых ИС новых поколений

При этом возникают проблемы, требующие дальнейшего совершенствования теоретической базы исследования сложных систем методами современной компьютерной математики в рамках концепции экстренных вычислений.

Первая проблема состоит в поиске новых подходов к интерпретации качественной и количественной картины динамики систем в условиях неопределенности, создании гибких механизмов управления и устранения ошибок в моделях взаимодействия и, в первую очередь – со-

вершенствовании математического аппарата и критериального базиса эволюции системы при непрерывном изменении динамики объекта и внешней среды. Решение этой проблемы ведется путем выделения порождающих задач компьютерной математики, допускающих декомпозицию и агрегирование с помощью формальных процедур организации структур данных в условиях неопределенности возмущений.

Вторая проблема состоит в формировании знаний о поведении систем в сложных ситуациях на основе динамической модели катастроф. В основе моделей компьютерной математики лежит интерпретация транспортных катастроф в виде последовательности изменений в организации системы. При этом возмущения рассматриваются не только в соответствующем функционале, но и в каждой точке порождающей задачи на основе построения функции управления. Такая процедура сводится к определению множества решений и может быть реализована в рамках концепции развивающихся иерархических систем с учетом новых данных о динамике взаимодействия и физических закономерностей исследуемых процессов и явлений.

Третья проблема является общей при компьютерной интерпретации систем на основе формализованной модели знаний. Эта проблема особенно актуальна при реализации основного принципа эволюции системы в рамках динамической модели катастроф – поведение системы при движении к целевому аттрактору и при потере устойчивости. В этих условиях одним из важнейших концептуальных решений является построение формальных процедур анализа риска при возникновении катастрофы на множестве поведений и управлений для исследуемого функционала в заданных ограничениях. В рамках такой интерпретации открываются возможности решения прикладных задач поведения систем в сложных динамических средах.

Заключение

Практические приложения современной компьютерной математики охватывают широкий класс задач взаимодействия ДО с внешней средой: возникновения пульсирующей «потенциальной ямы», посадка летательных аппаратов в морских условиях, моделирование чрезвычайных ситуаций при ударе разрушающейся и экстремальной волны (волны-убийцы), а также динамики транспортных потоков [1 – 3].

1. *Нечаев Ю.И.* Теория катастроф: современный подход при принятии решений. – СПб: Арт-Экспресс, 2011.
2. *Нечаев Ю.И., Петров О.Н.* Непотопляемость судов: подход на основе современной теории катастроф. – СПб: Арт-Экспресс, 2014.
3. *Нечаев Ю.И.* Топология нелинейных нестационарных систем: теория и приложения. – СПб: Арт-Экспресс, 2015.

4. GTSI Cloud Computing Maturity Model [Электронный ресурс]: http://www.gtsi.com/sms/documents/White-Papers/Cloud Compuing.pdf
5. *Lublinsky B.* Defining SOA as an architectural style. 9 January 2007. [Электронный ресурс] / Lublinsky B. – Режим доступа: http://www.ibm.com/developerworks/architecture/library/ar–soastyle/
6. Urgent Computing Workshop 2007 [Электронный ресурс] / Argonne National Lab, University of Chicago, April 25-26, 2007. – Режим доступа: http://spruce.teragrid.org/workshop/urgent07.php.
7. *Zadeh L.* Fuzzy logic, neural networks and soft computing // Commutation on the ASM-1994. – Vol.37. – №3. – P. 77 – 84.

А.В. Пелипец

РАСПАРАЛЛЕЛИВАНИЕ ИТЕРАЦИОННЫХ МЕТОДОВ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ НА РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ*

НИИ многопроцессорных вычислительных систем ЮФУ, г. Таганрог, pelipets@mail.ru

Существует множество методов решения систем линейных алгебраических уравнений (СЛАУ), хорошо зарекомендовавших себя в различных областях применения. Среди них можно выделить итерационные методы, которые наилучшим образом подходят для решения задач больших размерностей, где прямые методы сталкиваются с проблемами недостатка оперативной памяти, падения производительности и точности вычислений из-за необходимости выполнения очень большого числа арифметических операций [1]. Кроме того, итерационные методы наиболее эффективны для задач математической физики с разреженными матрицами специального вида.

Одним из главных направлений развития вычислительной линейной алгебры, в частности, итерационных методов решения СЛАУ, стала всесторонняя адаптация алгоритмов и программ к прогрессирующим вычислительным архитектурам. Эффективное отображение информационной структуры алгоритма или программы на архитектуру компьютера долгое время считалось краеугольной концепцией высокопроизводительных вычислений.

Однако рост аппаратного параллелизма за счет многопроцессорности и многоядерности породили проблему затрат на осуществление трансфера данных между компонентами системы, что стало негативно

* Работа выполнена при частичной финансовой поддержке Южного научного центра Российской академии наук по теме № 0256-2015-0080 (**00-16-15**) государственного задания в рамках программы I.5П "Проблемы создания высокопроизводительных, распределенных и облачных систем и технологий. Интеллектуальные информационные технологии и системы"

сказываться на реальной производительности. Наиболее узким местом решения СЛАУ на суперкомпьютере стал обмен данными с подсистемой памяти, максимальная частота работы которой значительно отстает от скорости процессора.

В настоящее время решение обозначенной проблемы сводится к созданию вычислительных архитектур и адаптированных к ним алгоритмов, рассчитанных на то, что система хранения данных организована иерархическим образом. Идея иерархии памяти заключается в разбиении каналов передачи данных на различные уровни, чтобы транспортные расходы для уровней были существенно меньше, чем для всей иерархии в целом. Однако значительная фрагментация памяти ведет к дефициту каналов памяти для осуществления доступа к различным уровням иерархии.

Современная альтернатива многопроцессорным системам с неизменной «жесткой» архитектурой – реконфигурируемые вычислительные системы (РВС), построенные на основе программируемых логических интегральных схем (ПЛИС) [2], для которых проблема дефицита внешних каналов обмена также имеет место. Причина этого в том, что в каждом новом поколении программируемых микросхем усиливается тенденция удельного отставания числа пользовательских выводов относительно логической емкости кристалла [3-4]. В этой связи становится актуальным такое распараллеливание решения задач линейной алгебры, при котором достигается минимизация числа используемых внешних каналов обмена.

Рассмотрим квадратную систему уравнений размерностью n:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n, \end{aligned} \tag{1}$$

где a_{ij} – коэффициенты при неизвестных, b_i – вектор правой части системы.

Согласно итерационному методу Якоби для системы уравнений (1) общей формулой вычислений компонент вектора-приближения будет

$$x_i^{k+1} = \gamma_i \left(\delta_i - \sum_{j \neq i} a_{ij} x_j^k \right), \tag{2}$$

где $\gamma_i = \dfrac{1}{a_{ii}}$; $\delta_i = \dfrac{b_i}{a_{ii}}$ – константы, которые рассчитываются однократно, до начала основных вычислений; $i = 1,2,\ldots,n$; $k = 0,1,2,\ldots$ – номер итерации.

Итерации завершаются, когда выполняется условие:

$$\| x_j^{k+1} - x_j^k \| \le \varepsilon \ , \qquad (3)$$

где ε – это требуемая точность решения.

Если процесс нахождения одного неизвестного x_i на (k+1)-й итерации отобразить в виде информационного подграфа α_i, реализующего формулу (2), то полный вычислительный граф β процесса решения задачи может быть представлен в параллельном виде, как показано на рис.1.

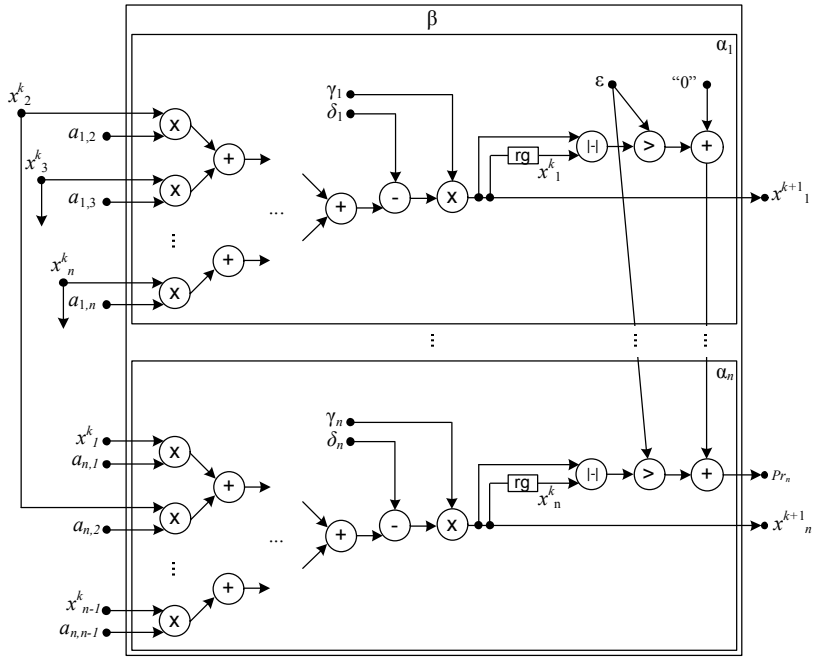


Рис. 1. Параллельная форма метода Якоби решения СЛАУ

В пределах одной итерации k информационный граф β параллельной формы данной задачи можно рассматривать как кортеж информационно-независимых подграфов $\langle \alpha^k_1, \alpha^k_2, \ldots, \alpha^k_n \rangle$. Информационная независимость подграфов α_i определяется отсутствием дуг, соединяющих операционные вершины двух или более подграфов в кортеже.

Каждый n изоморфных информационных подграфов α_i на (k+1)-й итерации реализует n умножений и n сложений-вычитаний для получения на выходе значения неизвестной x_i^{k+1}. Кроме того, операции модуля разности, сравнения и суммирования реализуют процедуру проверки условия сходи-

мости (3), результатом которой является предикат Pr_n. Управляющая программа останавливает вычислительный процесс при обнаружении предиката, значение которого говорит о выполненном условии сходимости.

Общее число операционных вершин полного графа β параллельной формы задачи будет составлять $2n^2 - n$. Если бы количество итераций, которые необходимо пройти до завершения процесса вычислений, было известно заранее (например, m), то алгоритм мог быть представлен в абсолютно параллельной форме с числом базовых подграфов, равным $n \cdot m$. Но поскольку необходимое количество итераций изначально не определено и зависит от выполнения условий сходимости, то число информационных подграфов в параллельной форме равно размерности матрицы n, а количество реализованных ими итераций может быть произвольным. Промежуточные результаты итерации k, являющиеся исходными данными для итерации k+1, должны записываться и считываться из внешней оперативной памяти.

Количество аппаратного ресурса для реализации параллельной формы задачи рассчитывается по формуле

$$A = n \cdot N(\alpha_i), \tag{4}$$

где $N(\alpha_i)$ – количество ресурса, необходимого для реализации базового информационного подграфа α_i.

Из (4) следует, что структурная реализация параллельной формы задачи требует наличия $(2n^2 - n)$ процессорных элементов и $4n^2$ внешних каналов обмена.

Если вычислительного ресурса недостаточно для реализации всех n базовых подграфов, то возможна редукция используемой аппаратуры за счет сокращения числа базовых подграфов в кортеже и уменьшения количества операционных вершин. Кадр Q, к которому был сведен кортеж, редуцированный до одного базового подграфа с предельным сокращением операционных вершин, приведен на рис. 2. Процесс решения задачи в данной кадровой форме будет представлять собой итерационную обработку потока операндов, где общее число итераций увеличится пропорционально степени редукции полного информационного графа.

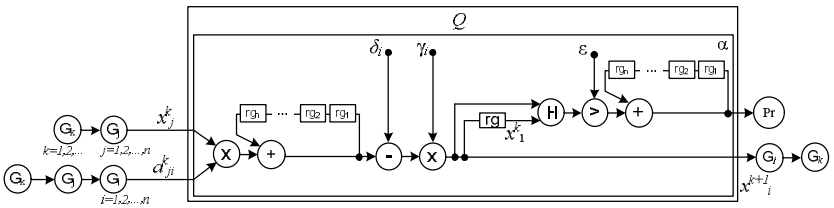


Рис. 2. Кадровая форма метода Якоби решения СЛАУ, с предельной редукцией числа базовых подграфов и операционных вершин

Важная особенность итерационного метода Якоби состоит в том, что информационная зависимость присутствует между подграфами соседних итераций, на которых ведется последовательный перерасчет одной и той же неизвестной. Это делает возможным организацию параллельно-конвейерной формы вычислительного процесса, при которой каждая ступень конвейера *P* одновременно реализует один базовый подграф (рис. 3). При этом результатом такой формы распараллеливания по итерациям является сохранение минимального количества внешних каналов обмена, которое было достигнуто в результате предельной редукции базового подграфа по числу функциональных устройств.

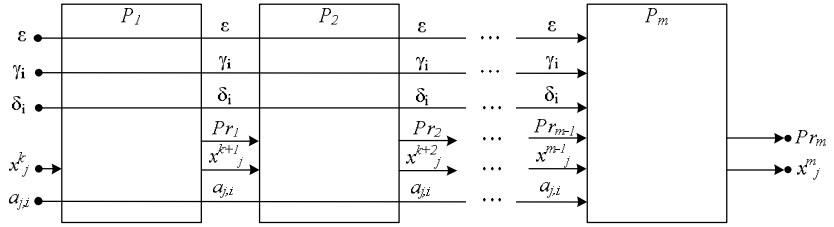


Рис. 3. Параллельно-конвейерная форма метода Якоби решения СЛАУ с применением распараллеливания по итерациям

Параллельно-конвейерная форма метода Якоби решения СЛАУ размерностью 10^4 с применением распараллеливания по итерациям была реализована автором на базовом модуле РВС «Тайгета», содержащем 16 ПЛИС XC7VX485T. Экспериментальная проверка показала, что размещение 92 ступеней вычислительного конвейера в одной ПЛИС позволило получить преимущество в скорости по сравнению с персональным компьютером (Intel Core i5-3570K, 3.4 ГГц, 8 Гб ОЗУ) примерно в 70 раз. Результаты реализации, полученные на примере метода Якоби, говорят о перспективах применения метода распараллеливания по итерациям для решения СЛАУ на реконфигурируемых вычислительных системах.

1. *Амосов А.А., Дубинский Ю.А., Копченова Н.В.* Вычислительные методы для инженеров. – М.: Изд-во МЭИ, 2003.
2. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – 2-е изд., перераб. и доп. / под общ. ред. И. А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
3. 7 Series FPGAs Overview. Xilinx, 2015. [Электронный ресурс]. URL: http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (дата обращения: 23.05.2016).
4. UltraScale Architecture and Product Overview. Xilinx, 2016. [Электронный ресурс]. URL: http://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf (дата обращения: 23.05.2016).

Е.Е. Семерникова, В.А. Гудков

УТИЛИТА ТРАНСЛЯТОРА ЯЗЫКА COLAMO ДЛЯ БИТОВОЙ ОБРАБОТКИ ДАННЫХ

НИИ многопроцессорных вычислительных систем ЮФУ,
г. Таганрог,
semernikova_e@mail.ru

Имеющиеся сегодня средства программирования РВС для битовой обработки данных не обладают нужной эффективностью, поскольку совокупность инструментов работы с отдельными битами, автоматизированная работа на уровне заданной точности и работа с произвольной разрядностью в таких системах отсутствуют.

Для расширения возможностей доступа к битам в языке COLAMO предложена новая конструкция, позволяющая не только обращаться к любому биту в отдельности, но и разграничивать способ доступа к нему на параллельный и последовательный:

Var <Имя_переменной> : [<Число_бит> : Bit<Тип_доступа>];

Тем самым одна и та же программа, но с разными типами данных, будет выполняться параллельно (для типа доступа Vector) или конвейерно (для типа доступа Stream).

В дополнение к имеющимся двум измерениям массивов в языке COLAMO – количеству каналов данных (Vector) и глубине каналов (Stream) – предложено ввести ширину каналов данных в битах, которая позволяет разграничивать доступ на параллельный и последовательный (Bit Vector и Bit Stream). При этом сохраняется ортогональность принципов языка COLAMO, допускаются любые сочетания измерений массивов, работа со смешанными типами.

С помощью введенных инструментов по обработке битовых данных становится возможной разработка секционных арифметических вычислительных устройств, таких как сумматоры, вычитатели, умножители и т.д. Такие вычислительные устройства могут быть различных типов: параллельно обрабатывающие входные данные, последовательно обрабатывающие, масштабирующие разрядность полученного результата или не масштабирующие. Созданные подобным образом операционные блоки ложатся в основу программной библиотеки вычислительных устройств с переменной разрядностью.

Арифметические операции над битовыми данными произвольной длины требуют от разработчика построения вычислительных блоков соответствующей разрядности. В противном случае использование стандартных вычислительных блоков фиксированной разрядности приведет к использованию избыточного аппаратного ресурса. Для упроще-

ния процесса написания программ по обработке битовых данных в языке COLAMO предложены арифметико-логические макрооперации с возможностью масштабирования разрядности результата, аргументами которых являются битовые массивы Bit Vector или Bit Stream.

В зависимости от способа обработки битов, указанных при описании массивов, на место соответствующей макрооперации транслятором будут подставляться параллельные или последовательные арифметические устройства требуемой разрядности из программной библиотеки. Макрооперации делятся на два типа: масштабирующие, возвращающие результат разрядности, заданной пользователем, и немасштабирующие, возвращающие значения с увеличенной разрядной сеткой (для сложения и вычитания на 1 разряд, для умножения – сумма разрядностей входных операндов).

Так как разрядность операндов и результата может изменяться в процессе вычислений и заранее не определена, то появляется необходимость создания типа виртуальной переменной Virtual, размерность которой изначально не задана и определяется разрядностью первого присвоенного ей значения.

Введенные виртуальные переменные рассматриваются как совокупность произвольного числа одновременно доступных битов, т.е. переменной типа Virtual ставится в соответствие виртуальный битовый массив [Virtual : bit vector], размерность которого будет определяться в процессе разбора программного кода. Разработаны правила определения разрядностей виртуальных битовых структур в контексте работы с немасштабирующими и масштабирующими вычислительными операциями, а также в контексте работы с условными операторами, размерностью массивов и счетчиком генераторов.

Введенные механизмы контекстного определения разрядности типа данных Virtual и вычислительных макроопераций позволяют сократить время реализации алгоритмов с произвольной разрядностью данных.

На основании разработанных методов обработки и трансляции битовых и виртуальных конструкций создана утилита транслятора языка COLAMO для битовой обработки данных. Утилита осуществляет анализ текста на языке COLAMO, поиск соответствующих конструкций, их разбор и преобразование к формату, подходящему для дальнейшей обработки непосредственно транслятором языка.

Разбор битовых конструкций типа Bit Vector и Bit Stream, а также обработка виртуальных конструкций типа Virtual происходит на основе методов и алгоритмов, разработанных автором лично. Они позволяют определять разрядности как самих переменных, так и операционных блоков, аргументами которых эти переменные являются. Рассчитанная таким образом ширина в битах позволяет выбирать из программной

библиотеки вычислительные устройства требуемой разрядности, что, в конечном счете, позволяет наиболее точно адаптировать структурную реализацию к задаче.

Структура утилиты, отличающаяся наличием процедур приведения к единой степени распараллеливания по битам и масштабирования по разрядности, а также библиотеки вычислительных устройств с переменной разрядностью, приведена на рис. 1.

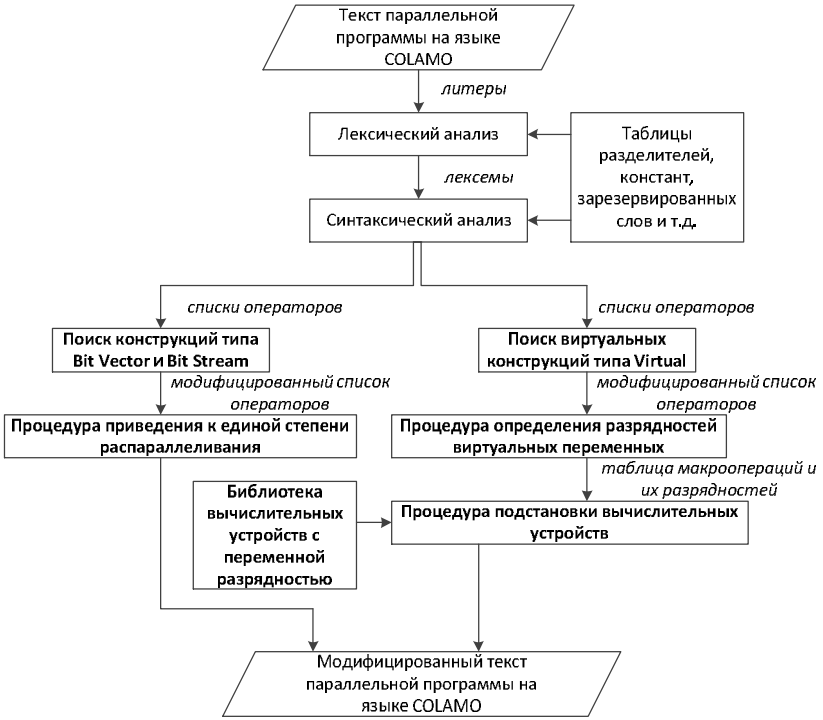


Рис. 1. Схема структуры утилиты транслятора языка COLAMO

Разработан ряд алгоритмов для согласования потоков данных и контекстного определения разрядностей виртуальных переменных и вычислительных устройств. В частности, разработан алгоритм обработки списка операторов, отличающийся поиском битовых конструкций и наличием следующих процедур: приведением к единой степени распараллеливания, определением разрядностей виртуальных переменных, подстановкой вычислительных устройств. В результате работы алгоритма происходит модификация исходного текста программы (рис. 2).

Утилита может входить в состав интегрированной среды разработки параллельных программ в качестве подключаемого программного модуля. В таком случае она будет являться составной частью комплекса математического обеспечения: транслятора, синтезатора, библиотеки VHDL описания элементов, разрабатываемого в НИИ МВС ЮФУ.

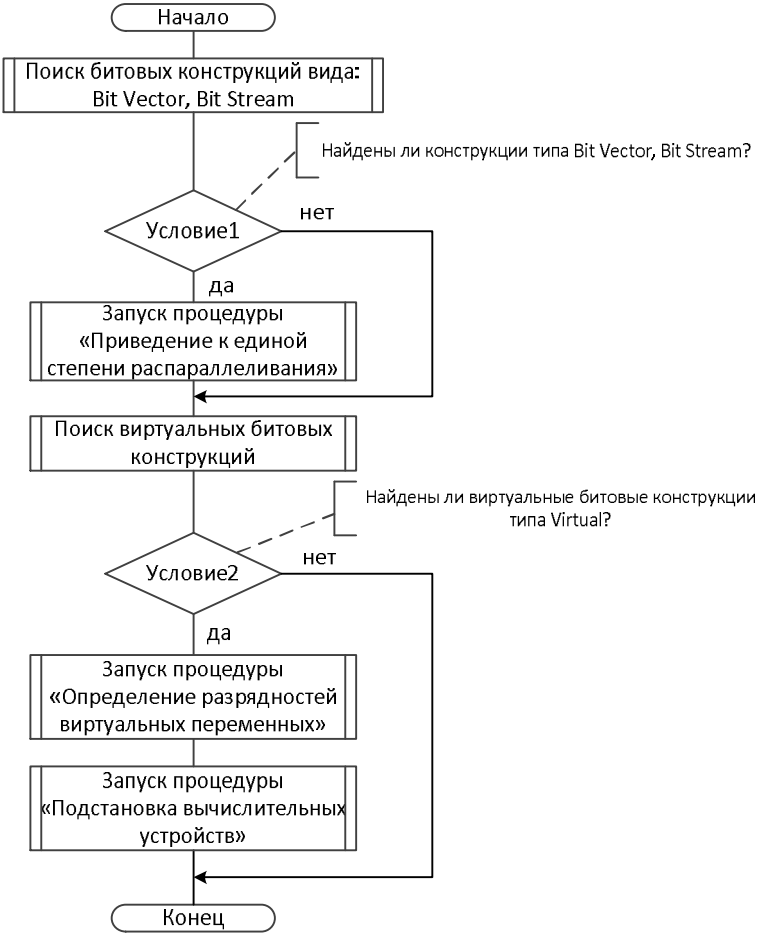


Рис. 2. Алгоритм обработки списка операторов

В окне подключения утилиты к интегрированной среде разработки параллельных программ есть параметр настройки утилиты, при вызове которого открывается диалоговое окно настроек, в котором осуществ-

ляются подключение необходимых программных библиотек вычислительных устройств переменной разрядности и дополнительные настройки работы утилиты.

Анализ эффективности работы утилиты транслятора языка программирования COLAMO для битовой обработки данных на различных задачах из области ЦОС и символьной обработки показал, что при создании масштабируемых по разрядности программ удалось достичь сокращения времени их создания на 7 – 10% по сравнению с временем создания аналогичных программ на языке VHDL. Еще большего выигрыша по времени (в несколько десятков раз) удалось достичь при переходе к новой разрядности (при изменении технических характеристик устройств, подающих входные данные), а также при переходе к новому варианту распараллеливания.

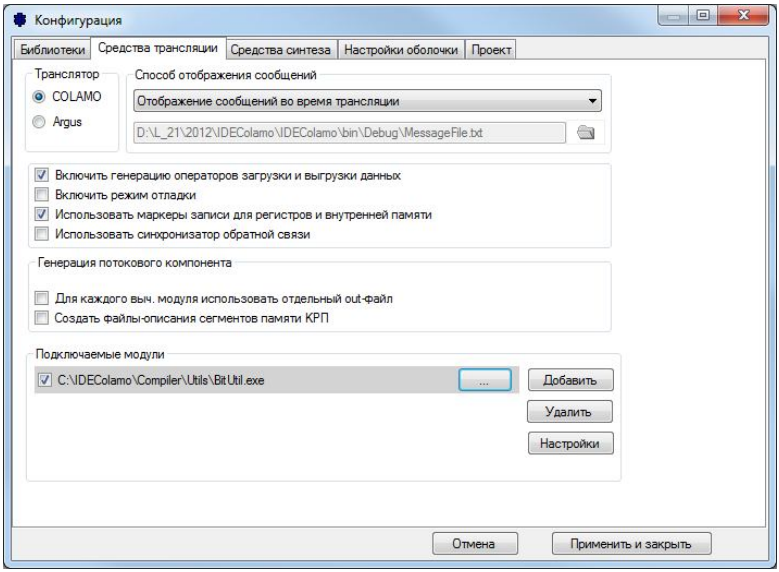


Рис. 3. Окно параметров интегрированной среды разработки прикладных программ с подключенной утилитой

Таким образом, использование утилиты транслятора языка COLAMO для битовой обработки данных позволяет сократить время разработки прикладных программ за счет сокращения строк программного кода, при этом сохраняется заданный уровень удельной производительности задачи. Более компактная запись исходного текста программы улучшает ее читабельность, а также облегчает дальнейшую отладку.

А.В. Уткин, В.М. Фомин

ПРИМЕНЕНИЕ ГИБРИДНЫХ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ В МЕТОДЕ МОЛЕКУЛЯРНОЙ ДИНАМИКИ

Институт теоретической и прикладной механики им. С.А. Христиановича СО РАН, г. Новосибирск, utkin@itam.nsc.ru

Метод молекулярной динамики дает хорошую возможность для детализированных исследований на субмолекулярном уровне особенностей фазовых переходов и механизма формирования наноструктур, а также позволяет получать их термомеханические характеристики. Одним из наиболее сложных моментов моделирования в рамках метода молекулярной динамики является значительное расчетное время задачи. В представленной работе рассматриваются различные подходы к созданию высокоэффективных параллельных алгоритмов, позволяющие как существенно ускорить сам процесс численного расчета, так и увеличить размер моделируемой системы.

Физическая система и постановка задачи

Появление гибридных вычислительных кластеров обусловило необходимость разработки новых алгоритмов, которые используют технологию CUDA для проведения части ресурсоемких расчетов на ГПУ, а для связи между различными ГПУ, физически принадлежащих разным узлам кластера, используется технология MPI. В представленной работе рассматривается подобный гибридный алгоритм, позволяющий объединить технологии CUDA и MPI в одной программе. Было проведено сравнение эффективности созданного кода как с программой, использующей только интерфейс MPI, так и с CUDA программой предназначенной для работы только с одним ГПУ. Тестирование алгоритмов и программ проводилось на физической системе, описывающей столкновение двух медных кластеров (рис. 1, 2). Взаимодействие атомов меди описывалось моделью погружённого атома (Embedded Atom Model – EAM), которая хорошо зарекомендовала себя для молекулярно-динамического моделирования различных металлов [1]. Число атомов системы составляло 243 644 и 709 214 атомов. Интегрирование уравнения движений поводилось с использованием схемы Верле второго порядка точности по временному шагу [2]. Шаг по времени при численном моделировании составлял 10^{-16} с .

Расчет сил межатомного взаимодействия является наиболее трудоемким, с точки зрения вычислительных ресурсов, частью молекулярно-динамического моделирования, поскольку при расчете сил, действующих на атом i, необходимо учитывать вклад со стороны всех соседних атомов.

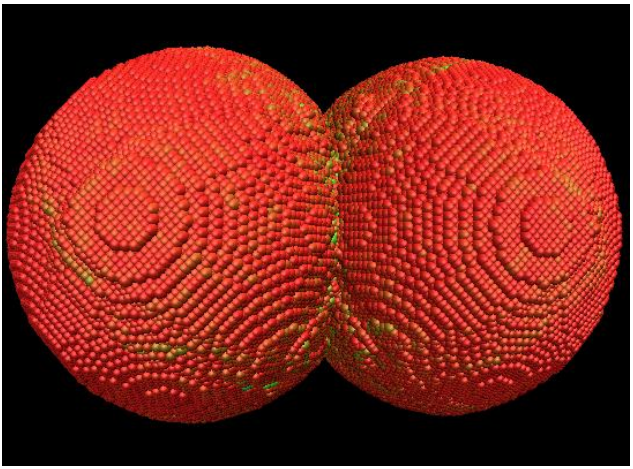


Рис. 1. Столкновение медных кластеров. Начальная скорость кластеров 350 м/с

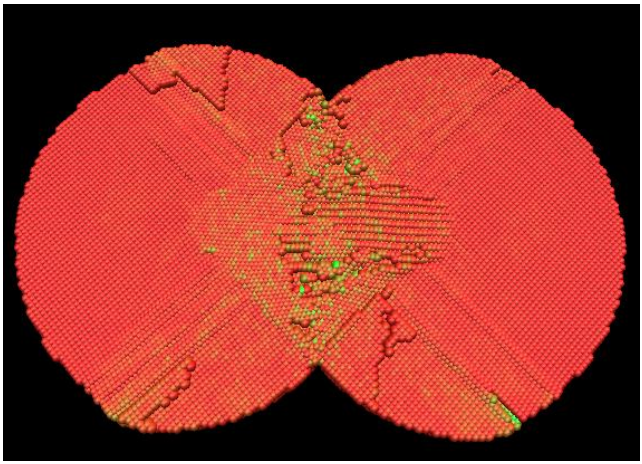


Рис. 2. Продольный разрез столкнувшихся кластеров в момент времени 10 пс. Желтым цветом обозначены атомы с наибольшей кинетической энергией

Существует ряд общепринятых методик оптимизации расчетов сил, которые позволяют значительно сократить время расчета [2]. К наиболее часто используемым способам оптимизации можно отнести список Верле (список соседей), метод связанных списков (Cell linked list

method-CLLM), а также различные гибридные комбинации метода связанных списков и списка Верле (гибридный список Верле). Самой очевидной гибридной комбинацией является метод, где список Верле для атома i создается не на основе перебора всех атомов системы, а на основе анализа расстояния до атомов в ячейке, которой принадлежит атом i и расстояния до атомов в соседних 26 ячейках. Тем самым исключается одна из основных проблем, связанных с необходимостью перебора всего набора атомов системы, при построении списка Верле.

Время расчета задачи (секунды)

Число атомов	**MPI (9ЦПУ)**	**MPI+CUDA (9ЦПУ+ 9ГПУ)**	**CUDA (1 ГПУ)**
243644	14792	4234	5269
709214	32081	8218	16970
Метод сортировки	CLLM	CLLM (26 сосоедних ячеек)	CLLM+Verlet list*

**– Обновление списков соседей проводилось каждые 20 шагов. Расчеты проводились на гибридном вычислительном кластере, установленном в ССКЦ СОРАН (НКС-30Т+ГПУ, 40 узлов, на каждом 2 Xeon X5670, 3 ГПУ Tesla M 2090).*

Создание вычислительного кода проводилось на основе уже апробированной ранее MPI-программы. Данная программа основана на разбиении системы атомов по пространству (одномерная параллелизация с дополнительным алгоритмом динамической балансировки). Поскольку расчет силовых взаимодействий является самой трудоемкой частью вычислений, именно эта часть переносилась с ЦПУ на ГПУ. Так как операция копирования данных с ЦПУ на ГПУ отнимает очень много времени, было принято решение создавать метод связных списков непосредственно на ГПУ, а не копировать уже готовые массивы данных из основной памяти компьютера. Соответственно расчет сил проводился на ГПУ при помощи метода связных списков (26 соседних ячеек).

Очевидным недостатком данной схемы является необходимость копирования большого количества информации между ЦПУ и ГПУ на каждом расчетном шаге, с целью организации обмена данными между ГПУ о перемещении атомов внутри расчетной области. Меняющееся на каждом шаге число атомов NЦПУ на каждом вычислительном ядре ЦПУ делает невозможным использование списков Верле.

Результаты

Было выполнено сравнительное тестирование созданных программ. Общее время расчета составляло 0,005 нс. – 50000 шагов 10^{-16} с. Из представленных в таблице временных зависимостей следует, что самым быстрым вариантом является гибридная программа MPI+CUDA. Для системы, состоящей из 243 644 атомов, программа основанная только на технологии CUDA и гибридных списках Верле использует немного больше временных ресурсов (запуск проводился на 1 ГПУ). Гибридная программа MPI+CUDA считает быстрее чем MPI-программа почти в три с половиной раза для системы, состоящей из 243 644 атомов, и в четыре раза быстрее для системы, состоящей из 709 214 атомов.

Следует отметить, что алгоритм MPI+CUDA несет в себе все недостатки, присущие алгоритму MPI, связанные, в первую очередь, с необходимостью обменов между узлами ЦПУ, а также с их неравномерной загрузкой. Как уже упоминалось выше, из-за постоянно меняющегося числа атомов на каждом узле невозможно использовать списки Верле, что также усугубляет ситуацию.

Однако не смотря на все эти недостатки, полученный код будет эффективен при моделировании физических систем, состоящих из большого числа атомов. Это связано с тем, что рост числа атомов приведет к неизбежному росту требований к памяти для хранения информации. Возможна ситуация, когда программа, основанная только на технологии CUDA и гибридных списках Верле, затребует больше ресурсов, чем может обеспечить ГПУ. В случае же использования гибрида MPI+CUDA подобная ситуация маловероятна, поскольку размер запрашиваемых ресурсов будет обратно пропорционален числу используемых ГПУ и ЦПУ.

Таким образом, был разработан и программно реализован гибридный параллельный алгоритм, позволяющий проводить расчеты с использованием ГПУ и ЦПУ (CUDA+MPI). Проведенное сравнение с аналогами, реализованными только на MPI или с использованием CUDA, демонстрирует хорошее быстродействие гибридного кода. Разработанные алгоритмы и программы были использованы для молекулярно-динамического моделирования на микроуровне явлений в твердых телах, при интенсивных внешних нагрузках.

1. *Voter A.F.* Embedded Atom Method Potentials for Seven FCC Metals: Ni, Pd, Pt, Cu, Ag, Au, and Al: Los Alamos Unclassified Technical Report / Los Alamos National Laboratory. – Los Alamos, 1993. – 9 p.– N. LA-UR 93-3901.
2. *M.P. Allen, D.J. Tildesley* Computer Simulation of Liquids. Oxford Science Publications, 2000. – 385 p.

АВТОРСКИЙ УКАЗАТЕЛЬ

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

СКТ-2016

Материалы 4-й Всероссийской научно-технической конференции

19 – 24 сентября 2016 г.
Дивноморское, Геленджик

ТОМ 1

Компьютерная верстка Иванова Н.Ю.
Редакторы: Кочергина Т.Ф., Проценко И.А., Селезнева Н.И.
Корректоры: Надточий З.И., Чиканенко Л.В.

Подписано в печать 29.07.2016.
Формат 60×84 1/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 12,21. Уч.-изд. л. 11,0.
Тираж 100 экз. Заказ № 5263.

Издательство Южного федерального университета.
344090, г. Ростов-на-Дону, пр. Стачки, 200/1, тел. (863) 247-80-51.

Отпечатано в отделе полиграфической, корпоративной и сувенирной продукции
Издательско-полиграфического комплекса КИБИ МЕДИА ЦЕНТРА ЮФУ
344090, г. Ростов-на-Дону, пр. Стачки, 200/1, тел. (863) 247-80-51.