УДК 004.27

В. М. Баканов, д-р техн. наук, проф., e-mail: vbakanov@hse.ru, Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ), г. Москва

Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов

Сформулированы задачи определения рациональных режимов работы вычислителей постфоннеймановской архитектуры с автоматическим распараллеливанием. Решение поставленных задач выполнено методами компьютерного моделирования, показана многовариантность решений и возможность выбора наиболее эффективных из них.

Ключевые слова: *распараллеливание процессов обработки данных, проблемы распараллеливания, автоматизация распараллеливания, программное и аппаратное распараллеливание, постфон-неймановские архитектуры, потоковые вычислители, интенсивность вычислений, компьютерное моделирование, стратегии управления интенсивностью вычислений*

В обозримом будущем повышение вычислительной мощности компьютеров реально лишь за счет распараллеливания процессов обработки данных. Специалисты прогнозируют значительные изменения в традиционных подходах к архитектурам параллельных систем (сейчас это в основном системы с распределенной памятью и передачей сообщений). В частности, специалист мирового уровня, разработчик вычислительных систем класса BEOWULF Томас Стерлинг считает [1], что следует обрабатывать данные сразу по мере их готовности, перемещая код к данным (в противоположность тому, как это принято сейчас в моделях передач сообщений), следует перемещать данные к тому месту, где они будут обработаны когда-то в будущем, опираться на микроархитектуру, построенную скорее на идеях обработки потока данных (DATA-FLOW), чем на процессорных ядрах, интенсивно использующих спекулятивное исполнение команд — так экономится энергия. Актуальными остаются вопросы рациональной загрузки вычислительных ядер современных многоядерных процессоров для персональных компьютеров — лишь небольшой процент полезных приложений "умеет" это делать.

Исследования в данной области представляют прямой интерес для программной инженерии, поскольку внедрение новых архитектур вычислителей не может не отразиться в будущем на базовых подходах к программированию (особенно к *параллельному программированию*) как инженерной деятельности.

В настоящее время параллельное программирование требует сложных процедур выявления скрытого параллелизма в алгоритме и распределения параллельных ветвей между независимо работающими вычислительными узлами. Хотя для многих известных алгоритмов такая задача решена, для каждого нового алгоритма ее приходится решать заново; при этом решение зачастую далеко от эффективного. В связи с этим представляют интерес архитектуры вычислителей, позволяющие выявлять параллелизм автоматически на аппаратном уровне.

Одна из таких архитектур — предложенная на рубеже 1960/1970-х гг. Джеком Б. Деннисом не фоннеймановская *потоковая* вычислительная архитектура. Эта архитектура основана на управлении ходом вычислительного процесса не программой (как в традиционном подходе CONTROL-FLOW с использованием регистра-счетчика команд, являющего серьезным препятствием к распараллеливанию), а собственно данными (конечно, корректно организованными). При этом операторы выполняются на свободных *исполнительных устройствах* (ИУ) в соответствии с принципом "*готовности к выполнению*" (ГКВ). Этот принцип, в свою очередь, является следствием "готовности" (результат присваивания значений) всех операндов данного оператора. В нашей стране исследованиями возможности применения DATA-FLOW при создании супервычислителей занимался создатель многопроцессорных вычислительных комплексов "ЭЛЬБРУС" академик В. С. Бурцев [2].

Становление идеологии DATA-FLOW прошло через несколько ступеней — от "статической" (которой присущи "детские болезни" в виде требования единократного присваивания и других подобных) до "динамической" архитектуры. Последняя позволяет выполнять любые действия, доступные для иных архитектур, и при этом не утрачивает положительных качеств автоматизма аппаратного распараллеливания. В настоящее время на пути реализации полномасштабного DATA-FLOW-вычислителя стоят чисто технические барьеры. Необходимостью является применение ассоциативной оперативной памяти (АП, привычная RAM-память в данном случае не может быть использована в силу отсутствия возможности быстрого выявления ГКВ-операторов). Такая память необходимого размера и приемлемой стоимости сейчас не производится. Известные специалистам и связанные с особенностью функционирования SMP-архитектуры проблемные вопросы являются также технологическими и в последние годы активно решаются

в связи с разработкой и производством процессоров с числом ядер в десятки и сотни. Принцип DATA-FLOW уже два десятка лет эффективно используется при работе с кэшем (технология диспетчирования и выполнения — Dispatche/Execute, DE в процессорах P6, *Pentium Pro* [3]) в процессорах общего назначения и применяется в современных специализированных процессорах.

Для классического вычислителя DATA-FLOW размер гранулы параллелизма (последовательности команд, обладающих свойством независимости по данным) равен машинной инструкции. Известны исследовательские проекты с увеличенным размером гранулы (до нескольких сотен операторов и выше), однако при этом для поиска гранул в программе требуются услуги сложного распараллеливающего компилятора.

На основе сказанного автор считает, что поисковые исследования в области функционирования вычислительных архитектур DATA-FLOW перспективны.

Работа вычислителя DATA-FLOW характеризуется дополнительными параметрами (по сравнению с параметрами функционирования вычислителя классической фон-неймановской архитектуры). В случае DATA-FLOW важно знать, каким образом будет развиваться процесс вычислений по времени (*динамика процесса*), так как от этого зависит общее время выполнения программы. Не менее важно определить методы, позволяющие управлять этим процессом. Для определения этих параметров был разработан компьютерный симулятор DATA-FLOW-вычислителя, выполняющий определенную систему команд и позволяющий учитывать длительность исполнения каждой из них [4]. Эксперименты проводились на алгоритмах, наиболее часто встречающихся в задачах моделирования процессов и явлений в механике и физике и анализе данных.

Потоковый вычислитель аппаратно реализует анализ *информационного графа алгоритма* (ИГА, граф зависимостей вида "операции ↔ операнды") в *ярусно-параллельной форме.*

Зависимость числа ГКВ-операторов от времени выполнения программы имеет ярко выраженную неравномерность. Число выполняемых в текущий момент времени операций назовем *интенсивностью вычислений* (ИВ).

Зависимость числа выполненных операций (накопленная кривая ИВ) для конкретного алгоритма от времени выполнения программы и размера обрабатываемых данных представлена на рис. 1 (см. третью сторону обложки). В общем случае зависимость *текущего* числа N исполненных операций алгоритма зависит как от размера n обрабатываемых данных, так и от времени t выполнения программы.

Заметим, что обычно рассматривается только *функция вычислительной трудоемкости* $N(n)$, представляющая собой срез графика на рис. 1 в момент окончания вычислений $t = t_0$; таким образом, *форма кривой ИВ признается несущественной.* При описании работы потокового вычислителя естественно расширение понятия функции вычислительной трудоемкости в *части ее изменения по времени выполнения программы.*

В вычислителях DATA-FLOW-архитектуры возможно управление динамикой вычислений, полезное с точки зрения рационального использования как имеющихся исполнительных устройств (ИУ), так и нагрузки (трафика) внутрикристальных шин передачи данных. Пусть $N_{\text{ГКВ}}^{\max}$ — максимальное число готовых к выполнению команд в ходе выполнения программы; P — число ИУ. Работа DATA-FLOW-вычислителя характеризуется следующими режимами (однозадачный вариант использования потокового вычислителя):

1) $P = 1$, в этом случае реализуется последовательный режим вычислений;

2) $N_{\text{ГКВ}}^{\max} > P > 1$, при этом режиме осуществляется смешанное последовательно-параллельное выполнение программы;

3) $N_{\text{ГКВ}}^{\max} \leqslant P$, этот режим характеризуется максимальным (для данной программы) распараллеливанием.

Иллюстрация этих режимов дана на рис. 2, где представлен результат моделирования управления процессом выполнения программы решения системы линейных алгебраических уравнений (СЛАУ) пятого порядка классическим методом Гаусса. Как видно, максимальное ускорение вычислений составляет чуть более 5 раз, а возможность управления динамикой вычислений (режимы интенсификации и депрессии) возможна только для второго из разобранных режимов (только в этом случае вариативность в порядке выполнения ГКВ-команд может оказывать влияние на динамику выполнения программы).

Зависимости *2* и *3* на рис. 2 соответственно иллюстрируют применение одной из стратегий управления ИВ в целях интенсификации процесса вычислений и его депрессии (по сравнению с равноприоритетной выборкой ГКВ-инструкций из буфера — кривая *1*). Интенсификация и депрессия в процессе вычислений, естественно, могут *целенаправленно* перемежаться. При значительном числе операций в алгоритме эффективность управления возрастает.

При определении приоритета выборки ГКВ-операторов из буфера команд в целях управления динамикой вычислений рационально использовать критерий "полезности" (фактически "*перспективности*") каждого оператора с точки зрения выполнения конечной цели — скорейшего завершения программы. В простейшем случае полезность оператора определяется числом иных операторов, для которых результат выполнения данного служит входным операндом (в рамках данной парадигмы возможны варианты, рис. 3). Такой подход требует просмотра ИГА минимум на одну дугу вперед.

Представляет интерес вопрос — для какого типа алгоритмов подобный метод управления является эффективным, а для какого таким не является? В целях выяснения ответа на этот вопрос была осуществлена серия вычислительных экспериментов по управлению динамикой вычислений для нескольких типов программ. Для исключения влияния различного времени выполнения отдельных арифметических инструкций это время принималось одинаковым.

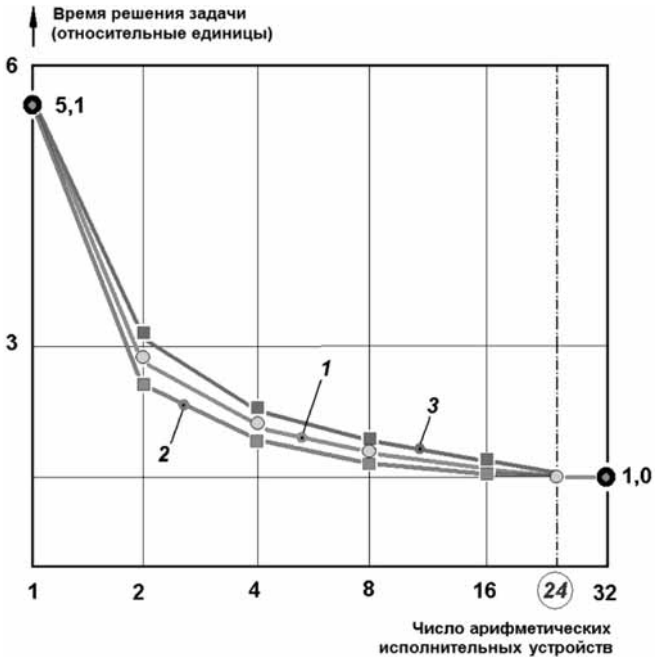

Рис. 2. Управление интенсивностью вычислений:

1 — равноприоритетное выполнение ГКВ-инструкций; *2* — режим интенсификации; *3* — депрессивный режим

В табл. 1 приведены результаты моделирования решения СЛАУ десятого порядка классическим методом Гаусса (число арифметических действий 805, максимальное ускорение при распараллеливании 12,8 при $N_{\text{ГКВ}}^{\max} = 90$) для трех (из нескольких рассматривавшихся) стратегий управления динамикой вычислительного процесса.

Все три стратегии управления основаны на просмотре ИГА вперед (относительно рассматриваемой инструкции) на один шаг с дальнейшим вычислением приоритета данной инструкции при выборке из буфера ГКВ-инструкций на исполнение свободным ИУ. Например, стратегия *A* прямо использует данные

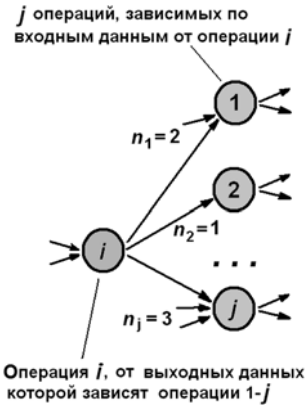

Рис. 3. К определению "полезности" (перспективности) оператора *i*; n_j — число операндов оператора *j*

о числе иных инструкций, зависящих по цепочке "результат → операнды" от выполнения данной, другие стратегии являются развитием описываемой при различных способах учета информационных зависимостей в графе. Значения таблицы показывают изменение времени выполнения программы (где "−/+" означают уменьшение/увеличение соответственно) относительно режима выборки из буфера ГКВ-инструкций в простейшем режиме (стек типа "первый вошел — первый вышел"). Прямое применение стратегии подразумевает формирование приоритета команд аналогично вышеописанному. При обратном применении приоритет вычисляется как обратная величина.

Из данных табл. 1 видно, что наиболее эффективной (из рассмотренных) является стратегия *C*, позволяющая управлять скоростью выполнения программы в пределах 4,5 % (двойной размах). Стратегии *A* и *B* показывают примерно одинаковый результат (двойной размах до 2,5 %). Заметим, что относительный "вес по времени выполнения" одной инструкции равен $1/805 \approx 0{,}12$ %. При этом стратегия *C* наиболее трудоемка и дополнительно нагружает АП при определении приоритетов.

Таблица 1

Эффективность управления динамикой исполнения программы для трех из ряда рассматриваемых стратегий (решение СЛАУ десятого порядка)

Число ИУ	Стратегия *A*		Стратегия *B*		Стратегия *C*	
	Эффект прямого применения, %	Эффект обратного применения, %	Эффект прямого применения, %	Эффект обратного применения, %	Эффект прямого применения, %	Эффект обратного применения, %
2	−0,24	+0,48	−0,24	+0,48	−1,69	+0,73
3	−0,35	+0,35	−0,35	+0,35	−2,13	+1,77
4	−0,46	+1,37	−0,46	+1,37	−2,28	+2,28
5	−0,55	+0,55	−0,55	+0,55	−1,10	+3,31
10	−0,84	0	−0,89	0	−0,89	0
20	−1,23	+1,23	−1,23	+1,23	−2,47	0
50	0	+1,52	0	+1,54	0	+1,54
70	0	0	0	0	0	+1,56

Таблица 2

Эффективность управления динамикой исполнения программы для трех из ряда рассматриваемых стратегий (вычисление коэффициента парной корреляции по 20 точкам)

Число ИУ	Стратегия *A*		Стратегия *B*		Стратегия *C*	
	Эффект прямого применения, %	Эффект обратного применения, %	Эффект прямого применения, %	Эффект обратного применения, %	Эффект прямого применения, %	Эффект обратного применения, %
2	0	0	0	0	+9,41%	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	+13,5%	0
10	0	0	0	0	0	0
20	0	0	0	0	−3,79	0
30	0	0	0	0	0	0
40	0	0	0	0	0	0

Анализ процедур решения СЛАУ различного порядка показывает увеличение управляемости с ростом объема программы. Анализ процедур решения СЛАУ, требующих повышенного времени выполнения операций — операций умножения и деления, и операций сложения и вычитания, показывает приблизительно близкие данные (однако менее продуктивен вследствие априорного незнания статистики по типам операций).

Интересно, что наибольший эффект управления достигается при небольшом числе ИУ (на порядки меньшем, чем $N_{\text{ГКВ}}^{\max}$), что и будет наиболее вероятным режимом при реальной эксплуатации потоковых вычислителей.

В табл. 2 приведены данные эксперимента для программы определения коэффициента парной корреляции по 20 парам точек (число арифметических действий 168, максимальное ускорение при распараллеливании 6,72 при $N_{\text{ГКВ}}^{\max} = 62$).

По данным табл. 2 видно, что в случае вычисления коэффициента парной корреляции (похожие результаты получены для задачи приближения функции методом наименьших квадратов) эффективность управления (управляемость) динамикой выполнения программы значительно хуже, чем для решения СЛАУ (табл. 1). Для стратегий *A* и *B* она нулевая, для оценки эффективности стратегии *C* требуются дополнительные эксперименты. Очевидно, что величина управляемости определяется свойствами ИГА.

Выявление критериев управляемости — непростая задача для будущих исследований. Например, плохая управляемость динамикой вычислений в задаче определения коэффициента парной корреляции и приближения функции методом наименьших квадратов может объясняться наличием множества последовательных цепочек при вычислениях сумм типа Σx_i, $\Sigma x_i y_i$ и подобных им при формальном "развертывании" циклов (для "разумного" компилятора естественно использовать процесс сдваивания при суммировании в целях получения алгоритма наименьшей высоты). Основой для определения управляемости (в указанном смысле) может быть уровень связности (ветвистости) ИГА данной программы.

Практический интерес представляет возможность априорного определения управляемости процесса вычислений в представленном выше понимании; это позволит более эффективно определить стратегию управления для данного алгоритма.

Логично разделить (условно) алгоритмы на "легко-" и "трудноуправляемые" (ЛУ и ТУ соответственно) в указанном смысле. К алгоритмам ЛУ-класса логично отнести алгоритмы, характеризующиеся значительным ветвлением (связностью) ИГА (число зависящих по связи "результат → операнды" от выполнения каждого оператора иных операторов велико), и к классу ТУ в противном случае. В первом случае широк выбор альтернативных путей "захвата" вершин ИГА (выполнения операций), во втором случае он более узок.

В табл. 3 дано распределение (гистограмма по числу карманов до 10) числа встречаемости связей между оператором и зависящими от него оператором по связи "результат → операнды". Например, в каждом столбце "1 → *N*" приведено общее число случаев связи по принципу "результат выполнения данной операции → число *N* зависящих от него иных операций". Программы slau_5 и slau_10 — решение СЛАУ пятого и десятого порядков соответственно, mnk_10 и mnk_20 — линейное приближение по методу наименьших квадратов для 10 и 20 точек, korr_10 и korr_20 — определение коэффициента парной корреляции для 10 и 20 точек. Параметр связности ИГА оценивался как отношение сумм числа альтернативных связей ("1 → *N*" при $N > 1$) к числу безальтернативных связей (связи "1 → 1"). Данные табл. 3 показывают, что алгоритмы ТУ-класса характеризуются параметром связности в 10—25 раз меньшим, чем алгоритмы ЛУ-класса. Разработки приемлемой во всех случаях стратегии управления динамикой вычислений продолжаются.

Таблица 3

Параметр связности ИГА (гистограмма) для различных программ потокового вычислителя

Программа	Карманы уровней связности операций по принципу "результат данной операции → число операндов зависящей от него операции"										Максимальное ускорение	Параметр связности
	1→1	1→2	1→3	1→4	1→5	1→6	1→7	1→8	1→9	1→10		
slau_5	89	5	7	9	4	0	0	0	0	0	4,60	0,281
slau_10	698	6	7	9	11	13	15	17	19	9	12,8	0,152
mnk_10	63	1	0	1	0	0	0	0	0	0	4,13	0,032
mnk_20	123	1	0	1	0	0	0	0	0	0	4,85	0,016
korr_10	85	0	2	0	0	0	0	0	0	0	5,87	0,024
korr_20	165	0	2	0	0	0	0	0	0	0	6,72	0,012

Итак, методом имитационного моделирования показана не только собственно возможность целенаправленного управления динамикой вычислений в процессорах потоковой архитектуры путем разработки обоснованных стратегий управления приоритетом выполнения машинных инструкций, но и определена количественная оценка возможностей этого управления для априори заданных типов алгоритмов. Результаты проведенных исследований целесообразно использовать при разработке DATA-FLOW-процессоров и встроенного программного обеспечения для них.

Список литературы

1. **Стерлинг Т.** Многоточие Стерлинга // Суперкомпьютеры. 2010. № 3. С. 17—20.
2. **Бурцев В. С.** Вычислительные процессы с массовым параллелизмом // Электроника: Наука, Технология, Бизнес. 2002. № 2. С. 32—35.
3. **Шпаковский Г. И.** Организация параллельных ЭВМ и суперскалярных процессоров. Минск: Изд. Белорусского гос. ун-та, 1996. 296 с.
4. **Ханжин Д. А., Баканов В. М.** Симулятор DATA-FLOW-вычислителя с массовым распараллеливанием операций на уровне инструкций процессора. Свидетельство о государственной регистрации программ для ЭВМ № 2013614155. Федеральная служба РФ по интеллектуальной собственности (Роспатент), дата регистрации 24.04.2013 г.

V. M. Bakanov, Professor, e-mail: vbakanov@hse.ru, National Research University Higher School of Economics, Moscow

Dynamics Control Computing in the Processor Data Flow Architecture for Different Types of Algorithms

The paper set out the goals of definition of rational modes of calculators post Von Neumann architecture with automatic parallelization of hardware (streaming, DATA FLOW architecture) and proposes solutions to some of them. Evaluators streaming architecture have undoubted advantages in the form of a fully automatic parallelization of data processing at the hardware level and can be considered as an alternative future of modern processors classical von Neumann architecture. In contrast to the traditional architecture of calculators in this case, you can control the intensity of computation (number of simultaneous operations), defined the conditions for such management. Targeted control of the intensity calculation is useful in terms of rational use of both existing performing devices and loads (traffic) inside chip data bus and allows more efficient use of calculators streaming in single and multi-tasking. Management strategies implemented by setting priorities sample ready to run operators from the buffer memory streaming command calculator, with the possible intensification of regimes as well as the intensity of depression calculations. The concept of function computing complexity of expanding its change in time of the program. These tasks performed by computer simulation shows the multiplicity of solutions and the ability to choose the most effective one. The effectiveness of several proposed strategies for managing the intensity calculation is shown by the example of several commonly used standard algorithms.

Keywords: *parallelization of data processing problems paralleling paralleling automation, software and hardware parallelism, post the von-Neumann architecture, streaming calculators, of computing-intensive, computer simulation, strategy, controlling the intensity calculations*

References

1. **Sterling T.** Mnogotochie Sterlinga. *Superkompyuteryi*, 2010, no. 3, pp. 17—20 (in Russian).
2. **Burtsev V. S.** Vyichislitelnyie protsessyi s massovyim parallelizmom. *Elektronika: Nauka, Tehnologiya, Biznes*, 2002, no. 2, pp. 32—35 (in Russian).
3. **Shpakovskiy G. I.** *Organizatsiya parallelnyih EVM i superskalyarnyih protsessorov.* Minsk, Izd-vo Belorusskogo gos. un-ta, 1996. 296 p. (in Russian).
4. **Hanzhin D. A., Bakanov V. M.** Simulyator DATA-FLOW — vyichislitelya s massovyim rasparallelivaniem operatsiy na urovne instruktsiy protsessora. Svidetelstvo o gosudarstvennoy registratsii programm dlya EVM # 2013614155. Federalnaya sluzhba RF po intellektualnoy sobstvennosti (Rospatent), date of registration 24.04.2013.