

ПОСТРОЕНИЕ РАСПРЕДЕЛЕННЫХ ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА БАЗЕ ОБЪЕКТНО-АТТРИБУТНОЙ АРХИТЕКТУРЫ

Салибекян Сергей Михайлович, ст. преподаватель, Московский институт электроники и математики (технический университет), Россия, Москва, salibek@yandex.ru

Панфилов Пётр Борисович, к.т.н., доцент, Московский институт электроники и математики (технический университет), Россия, Москва, panfilov@miem.edu.ru

Введение

Объектно-атрибутная (ОА) архитектура, разработанная в Московском институте электроники и математики, предоставляет огромные возможности: «врожденный» параллелизм вычислений, изоморфизм (возможность изменения части программы без опасения, что нарушится логика работы всей программы) программ и данных, удобная стыковка частей программы и легкая переносимость ОА-программ с одной аппаратной платформы на другую, программирование распределенной гетерогенной вычислительной системы (ВС) как единого целого [1]. Как раз на последнем качестве и остановим наше внимание.

Проблема создания гетерогенных вычислительных систем (ВС) весьма актуальна во многих областях: системы автоматизации, робототехника, высокопроизводительные вычислительные системы и т.д. Однако, широко применяемое на сегодняшний день объектно-ориентированное программирование (ООП) не приспособлено для работы на распределенных системах, т.к. все поля и методы объекта привязаны к оперативной памяти (ОП) вычислительного узла: каждому объекту выделяется область в ОП, доступ к полям и методом объекта осуществляется по их адресу. В настоящее время имеются библиотеки для языков высокого уровня, которые позволяют создавать распределенные объектно-ориентированные системы, однако реализуются они не самым оптимальным образом: для вызова методов удаленных объектов используется весьма громоздкий механизм удаленного вызова процедур (RPC). Так, программисту лишь для того, чтобы описать интерфейс вызываемой удаленной процедуры, приходится использовать специализированный язык программирования IDL (Interactive Data Language). Примером таких систем могут быть Corba и D-COM.

В задачу исследования входила разработка приемов программирования распределенных гетерогенных вычислительных систем на базе ОА-архитектуры, чтобы обеспечить максимально эффективное программирование распределенных гетерогенных вычислительных систем. Основам ОА-архитектуры посвящены публикации [1-3]. Далее же мы будем рассматривать вопросы создания гетерогенных ВС на базе ОА-архитектуры.

1. Распределенность и гетерогенность ОА-системы

Итак, гетерогенная распределенная вычислительная система представляет собой множество вычислительных узлов различной аппаратной архитектуры и различной вычислительной мощности, объединенных линиями связи. Требуется создать на базе этих узлов единое вычислительное пространство и произвести программирование такой ВС, как единого целого, а не описывать алгоритм каждого вычислительного узла по отдельности, а затем налаживать обмен данными между этим практически автономными программами. Решение поставленной задачи разделяется на несколько подзадач.

Во-первых, необходимо обеспечить однородное вычислительное пространство всей системы. ОА-архитектура решает этот вопрос: интерфейс программы реализации логики работы виртуальных функциональных устройств (ВФУ) одинаков для всех типов ВФУ (напомним, что он состоит из трех полей: указателя на контекст ВФУ, индекса милликоманды и указателя на нагрузку милликоманды). Процедура реализации логики работы ВФУ как бы скрывает за собой все особенности архитектуры вычислительного узла,

на котором она запущена; ОА-образ же работает исключительно с ВФУ, и потому может одинаково функционировать на вычислительном узле любой архитектуры. В ОА-платформу (напомним, что ОА-платформа – это совокупность программ реализации логики работы ВФУ), запускаемую на конкретном вычислительном узле, включаются только те типы ВФУ, которые необходимы для решения конкретной задачи, что экономит оперативную память и другие ресурсы вычислительного узла. Поэтому в распределенную ОА-систему могут входить узлы любой вычислительной мощности: от простейших микроконтроллеров и до суперкомпьютеров. Например, для создания гетерогенных ВС можно использовать переносимую операционную систему Linux... однако в такую ВС могут входить только довольно мощные вычислительные узлы, которые кроме выполнения алгоритма смогут дополнительно обслуживать и саму операционную систему, которая требует для себя как дополнительный объем оперативной памяти, так и процессорного времени. На узлах ОА-системы устанавливается только ОА-платформа, отнимающая минимальные ресурсы.

Во-вторых, необходимо обеспечить максимально простой интерфейс обмена информацией между вычислительными узлами. И эта задача также решается как бы сама собой: операнды между ВФУ передаются исключительно по одному (а не все разом, как в технологии RPC), что до предела упрощает протокол обмена информацией по сети, объединяющий вычислительные узлы. Более того, ОА-архитектура до минимума сводит заботу программиста о протоколе обмена данными: перед началом вычислительного процесса необходимо лишь только настроить шлюзы (ВФУ, специализирующиеся на передаче милликоманд по линиям связи) и маршрутизаторы (ВФУ, специализирующиеся на выборе канала связи, на который необходимо отправить милликоманду), а затем пересылка милликоманды, которая адресована для ВФУ на другом вычислительном узле, осуществляется автоматически: Шина (ВФУ, осуществляющее передачу милликоманд между другими ВФУ) определяет, что милликоманда адресована для ВФУ на другом узле, передает ее на маршрутизатор, который определяет необходимый канал обмена данными и передает милликоманду на ВФУ Шлюз, который и производит передачу милликоманды по линии связи. В состав вычислительного узла также входит Шлюз, который осуществляет прием переданной по линии связи милликоманды и его передачу на маршрутизатор, который, в свою очередь, передает ее на Шину, а с нее милликоманда попадает на ВФУ (рис.1).

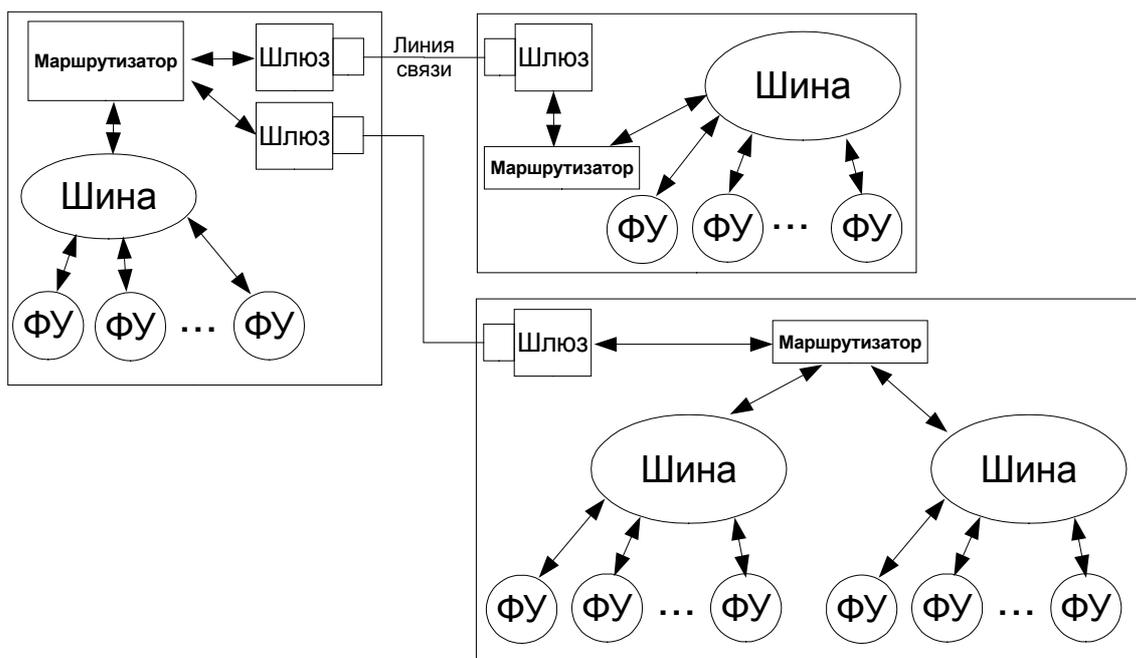


Рис. 1 – Шлюзование и маршрутизация в распределенной ОА-системе

В-третьих, весьма желательно, чтобы части программы могли перераспределяться с одного вычислительного узла на другой. ОА-архитектура обеспечивает решение и этого

вопроса: в некоторых пределах (насколько позволяет топология связей узлов) возможно перераспределение ОА-образа (рис. 2), т.е. ВФУ могут быть запущены на различных вычислительных узлах, при этом логика работы всей программы будет неизменена.

В-четвертых, желательно писать распределенную программу не как совокупность разрозненных частей, которые лишь обмениваются информацией между собой, но как единое целое. И здесь ОА-архитектура работает на нас: ОА-программа представляет собой совокупность капсул с данными и миллипрограммами, которые воспринимаются программистом как единое целое, и не привязываются к конкретным вычислительным узлам.

В-пятых, необходимо обеспечить абстракцию программы и данных на уровне всей распределенной ВС, а не только в пределах одного вычислительного узла. ОА-архитектура решает и эту задачу: ОА-конус абстракций и прочие ОА-информационные конструкции [1] могут охватывать сразу несколько вычислительных узлов (таким образом, объектная философия распространяется и на распределенные ВС).

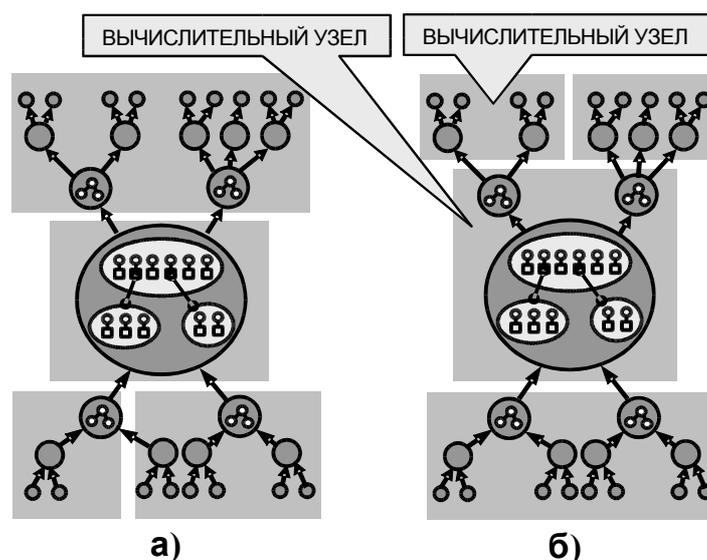


Рис. 2 – Перераспределение ОА-образа по вычислительным узлам

2. Автономное моделирование

Следует отметить ещё один «побочный эффект» довольно легкой переносимости ОА-программ на различные аппаратные платформы: возможность автономного моделирования аппаратно-программных комплексов (АПК). Итак, мы можем запускать ВФУ на выполнение как непосредственно на оборудовании распределенного гетерогенного АПК, так и на персональном компьютере или рабочей станции. Для эмуляции сигналов, которые приходят с датчиков, программист задает милликоманды или набор милликоманд, выводит их на Шину ОА-среды и наблюдает за реакцией среды на поступающую информацию. Реакцию же системы можно отследить по милликомандам, которыми ВФУ по Шине обмениваются между собой (на реальном объекте поставщиками информационных пар в ОА-среду будут датчики: они считывают данные, добавляют к ним соответствующие атрибуты и выдают на шину). При необходимости можно проводить и временное моделирование, для чего в алгоритм реализации логики работы ВФУ добавляется возможность отслеживания временных меток данных и эмуляция задержки вычислений. Для моделирования поведения более сложных объектов автоматизации составляются последовательности милликоманд, задается время поступления ИП в эмулятор системы, пишутся специальные программные модели, эмулирующие поведение управляемого объекта (модели объектов управления меняют состояние в зависимости от ИП, которые описывают выходные сигналы АПК, и сообщают модели АПК о своем состоянии также с помощью посылки информационных пар). Причем, тот же самый ОА-образ, используемый в качестве имитационной модели, впоследствии загружается в АПК, где на микроконтроллерах запущена ОА-платформа, и

начинает работать: в ОА-архитектуре ОА-образ является и моделью, и программой реализации алгоритма работы АПК одновременно [2].

3. Практическая реализация концепции – робот-шестиног

В настоящее время в лаборатории робототехники Московского института электроники и математики ведутся работы по созданию роботизированного комплекса «шестиногий робот». Систему управления данным роботом было решено выполнить на базе ОА-архитектуры.

Робот представляет собой шасси, на котором установлены шесть конечностей, каждая из которых имеет три степени свободы. На шасси располагаются три микроконтроллера марки Atmega (каждый контроллер управляет одной парой ног – это ограничение объясняется тем, что каждый микроконтроллер имеет только шесть управляющих выходов). Микроконтроллеры связываются между собой и с персональным компьютером, который также входит в состав аппаратно-программного комплекса (АПК) по стандартному интерфейсу RS-485 (связь с компьютером осуществляется через конвертер USB – RS-485). ОА-образ, описывающий походку робота, загружается на микроконтроллеры через ПЭВМ по интерфейсу RS-485. Робот может работать в двух режимах: автономном, когда на микроконтроллеры загружается весь образ, и в связанном, когда часть образа располагается на микроконтроллерах, а часть – на ПЭВМ (обмен информацией между компьютером осуществляется по проводной линии связи) (рис. 3). Введение связанного режима работы робота объясняется тем, что на используемых микроконтроллерах аппаратно не поддерживаются операции с плавающей точкой.

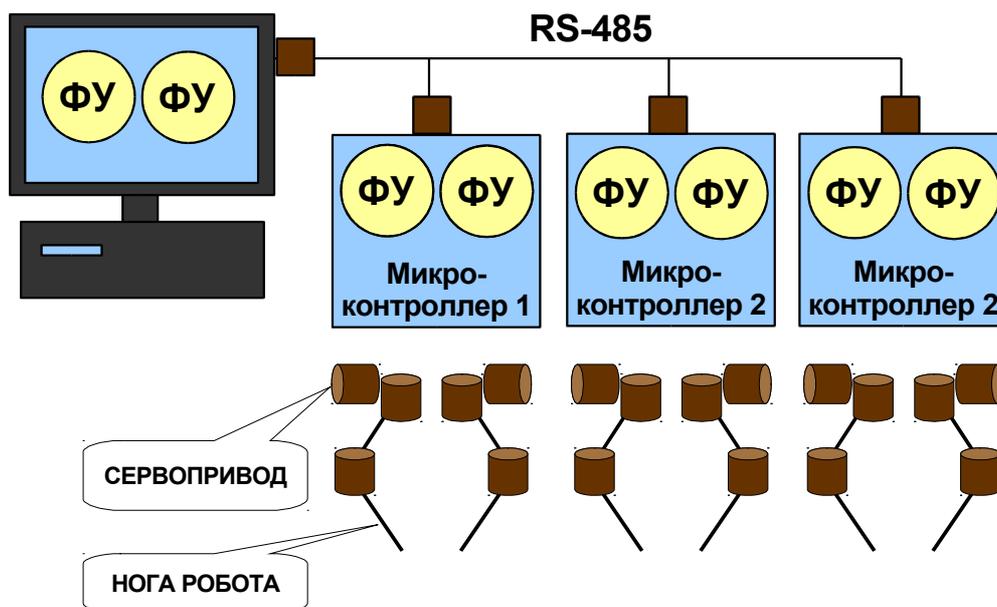


Рис. 3 – Распределение ОА-образа шестиногого робота по вычислительным узлам АПК

Работа по созданию АПК разделена на две части: автономное моделирование походки робота и запуск ОА-образа на оборудовании программно-аппаратного комплекса. На первом этапе производится создание и автономная отладка ОА-образа: для визуализации результатов работы образа используется графический язык OpenGL.

ОА-образ построен так, чтобы максимально облегчить его перенос с ПЭВМ на оборудование АПК – ОА-образ разделен на две части: расчетную (где непосредственно реализуется алгоритм походки робота) и управляющую (которая в случае управления реальным роботом производит управление сервоприводами конечностей робота или в случае моделирования производит подготовку и вывод изображения модели робота на экран). Для стыковки этих двух частей имеется специальное ВФУ под названием «Интерпретатор», в задачу которого входит интерпретация потока милликоманд. Расчетная часть ОА-образа работает с интерпретатором как с обычным ВФУ и выдает последовательность милликоманд

(интерпретатор принимает от расчетной части углы поворота конечностей робота). Интерпретатор же, если происходит моделирование системы, переводит поток милликоманд с указаниями углов поворота сервоприводов в милликоманды для формирования и вывода графического образа робота на экран; в том же случае, когда происходит непосредственно управление роботом, интерпретатор выдает на выход последовательность милликоманд для управления сервоприводами. Таким образом, для того, чтобы перейти со стадии моделирования на стадию запуска ОА-образа на реальном АПК, необходимо лишь создать управляющую часть ОА-образа, состоящую из интерпретаторов, преобразующих пришедшие от ПЭВМ углы поворота конечностей в милликоманды для ВФУ управления сервоприводами и сами ВФУ управления сервоприводами.

ОА-образ, запущенный на микроконтроллерах, работает на них как единое целое, т.е. нет необходимости программировать каждый микроконтроллер по отдельности, как это обычно делается для подобных систем [4]. ОА-платформа для микроконтроллера создается на языке С (ОА-платформа, запускаемая на ПЭВМ, реализована с помощью языка Delphi). Ввиду того, что в микроконтроллерах Atmega не предусмотрена аппаратная реализация операций с плавающей точкой, весьма эффективным оказалось включение в состав АПК персонального компьютера, на котором и производятся все необходимые расчеты, а затем через конвертер USB – RS-485 уже вычисленные углы поворота серводвигателей передаются на микроконтроллеры. Схема организации ОА-образа приведена на рис. 4.

Для полного полноценного функционирования ОА-образов на ОА-платформе, запускаемой на микроконтроллере, понадобились реализовать следующие типы ВФУ:

- Шина-маршрутизатор-шлюз. Данное ВФУ объединяет в себе сразу три функции: во-первых, Шина (ВФУ которое обеспечивает обмен данными между другими ВФУ); маршрутизатор (ВФУ, которое выбирает канал, по которому следует передавать милликоманду); Шлюз (устройство, которое реализует протокол обмена данными по линии связи). Все эти три функции были объединены в одно виртуальное устройство ввиду того, что на микроконтроллере применяется исключительно один канал связи и сама Шина легко может произвести как маршрутизацию, так и обеспечить протокол передачи данных по каналу связи RS-485.
- ВФУ управления сервоприводом. Выдает сигнал на сервопривод.
- ВФУ интерпретатор.
- Целочисленное АЛУ. Необходимо для того, чтобы производить элементарные арифметические вычисления.

Как видно, в платформу добавляются лишь только необходимые для решения конкретной задачи типы ВФУ, что экономит оперативную память и другие ресурсы вычислительного узла.

Вышеперечисленный перечень типов ВФУ достаточен для конфигурации АПК, в которой все вычисления с плавающей запятой производятся на ПЭВМ и затем уже вычисленные углы поворота передаются на микроконтроллеры. В той же конфигурации, когда вычисления будут производиться непосредственно на микроконтроллерах, возникает необходимость и в других типах ВФУ: дробное АЛУ (производит вычисления с плавающей точкой), тригонометрическое ВФУ (производит вычисление тригонометрических выражений), автомат (позволяет реализовывать определённые алгоритмы), Геометрическое ВФУ (производит преобразования пространственных координат: поворот, сдвиг по координатным осям, масштабирование и т.д.). При такой конфигурации можно изготовить автономного робота, т.е. робота, который реализует алгоритм ходьбы только с помощью своего бортового вычислителя. При этом ОА-образ при переходе на автономный вариант практически не подвергается переделке: те ВФУ, которые ранее запускались на ПЭВМ, теперь будут запускаться на микроконтроллерах.

Выводы

В результате проделанной работы доказана работоспособность концепции распределенных гетерогенных вычислительных комплексов на базе ОА-архитектуры. Можно сказать, что ОА-система может реализовывать объекты, которые «живут» сразу на нескольких вычислительных узлах (в классическом ООП объекты «обитают» исключительно в оперативной памяти одного вычислительного узла). Также были разработаны основные приемы создания подобных систем.

Дополнительно отработана концепция автономного моделирования АПК, которая впоследствии поможет отказаться или свести к минимуму дорогостоящее и трудоёмкое полунатурное моделирование.

Литература

1. Салибекян С.М., Панфилов П.Б. ОА-архитектура – новый подход к созданию объектных систем // Объектные системы – 2011: материалы III Международной научно-практической конференции (Ростов-на-Дону 10-12 мая 2011 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону, 2011. – С. 73-79 (http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf)
2. Салибекян С.М. Принципы милликомандной архитектуры как основа построения высокопроизводительных адаптивных вычислительных систем // *Автоматизация и современные технологии*. 2002. № 5.
3. Салибекян С.М., Панфилов П.Б. ОА-архитектура построения и моделирования распределенных систем автоматизации // *Автоматизация в промышленности*. 2010 №11
4. Денисенко В. В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. М.: Горячая линия Телеком, 2009.

УДК 004.04

РЕАЛИЗАЦИЯ ПЛАТФОРМЫ ДЛЯ СОЗДАНИЯ И УПРАВЛЕНИЯ ТОРГОВЫМИ БИРЖЕВЫМИ РОБОТАМИ

Олейник Павел Петрович, к.т.н., Системный архитектор программного обеспечения, ОАО «Астон», Россия, Ростов-на-Дону, xsl@list.ru

Введение

В настоящее время существует множество программных реализаций, автоматизирующих деятельность практически любой прикладной предметной области. Не исключением является и биржевая торговля. Сейчас имеется множество различных программных продуктов, позволяющих упростить процесс выставления/перемещения/отмены заявок в торговую систему. Многие из них позволяют разрабатывать собственные стратегии торговли на базе различных алгоритмов. Подобные программы принято называть «биржевые роботы», к достоинствам которых относятся:

1. Робот, в отличие от человека, не подвержен стрессу и поэтому будет следовать реализованному алгоритму независимо от текущего положения на рынке.
2. Трейдеру не требуется тотальный контроль за работой программного приложения. Т.е. нет необходимости постоянно находиться за монитором компьютера и отслеживать процесс торговли.

Несмотря на наличие описанных выше достоинств, имеется серьёзный недостаток: написанный алгоритм может содержать ошибки, приводящие к финансовым потерям. Однако в настоящий момент имеется множество методологий и инструментов, позволяющих минимизировать количество ошибок в приложении.

1. Критерии оптимальности разрабатываемой платформы

Разработка любого программного обеспечения начинается с формулирования задач, которые необходимо решить с помощью него. Поэтому необходимо сформировать ряд