Трёхмерные методики визуализации программного обеспечения параллельных и распределённых вычислений

В.Л. Авербух, А.Ю. Байдалин, Д.Р. Исмагилов, А.Ю. Казанцев

Традиционно считаются, что использование трёхмерности и анимации повышают эффективность визуализации при представлении сущностей программного обеспечения. Однако необходим анализ как метафор визуализации, так и исходных объектов с тем, чтобы уяснить, когда и как следует применять данные методики. В работе этот анализ проводится на целом ряде примеров (включая работы авторов), описывающих использование метафор комнаты, молекулы, ландшафта в системах визуализации программного обеспечения параллельных и распределённых вычислений.

1. Введение

Под визуализацией программного обеспечения понимается совокупность методик использования графики и средств человеко-машинного взаимодействия, применяемых для лучшего уяснения понятий и эффективной эксплуатации программного обеспечения ЭВМ, а также для спецификации и представления программных объектов в процессе создания программ.

Визуализация программного обеспечения параллельных вычислений включает в себя исследования и разработки визуальных языков параллельного программирования, визуальных отладчиков правильности и систем настройки, отладки, измерения и анализа производительности параллельных программ. Визуализация помогает понизить интеллектуальную сложность параллельного программирования за счёт использования разнообразных методик. При этом традиционно считаются, что использование трёхмерности и анимации повышают эффективность визуализации при представлении сущностей программного обеспечения. Однако на практике необходим анализ как метафор визуализации, так и исходных объектов с тем, чтобы уяснить, когда и как следует применять данные методики.

Эта работа посвящена исследованию методик визуализации программного обеспечения параллельных вычислений. В ней рассматривается возможность применения таких трёхмерных метафор визуализации, как метафора комнаты, метафора молекулы, метафора ландшафта.

2. Визуализация в параллельных вычислениях

В прошлом десятилетии имел место настоящий бум визуализации программного обеспечения параллельных вычислений, когда в США, а также в ряде других стран был создан целый ряд интересных систем визуального программирования, отладки и особенно отладки и настройки правильности параллельных программ.

В наших обзорах [1-2] описаны результаты этой работы, а также состояние данной области на первые годы этого века.

В случае визуальных языков параллельного программирования имеют место две основные тенденции - изображать в визуальном виде [потенциальное] распараллеливание процессов и изображать взаимодействие параллельных процессов, например, построенное по правилам MPI.

Графические виды отображений визуальных отладчиков правильности параллельных программ обычно основаны на двумерных отображениях состояния ппараллельных программ, взаимодействия процессов, очередей сообщений и пр. Существуют примеры использования естественной, заимствованной из моделируемой области образности для представления хода работы программы.

Большинство проектов, вышедших за рамки академических, относятся именно к системам

отладки, настройки, измерения и анализа производительности параллельных программ. Визуализация в этих системах в основном основана на отображении метрик производительности параллельных программ. Метрикой называется зависящая от времени функция, характеризующая некоторые аспекты производительности параллельной программы. Примеры простых метрик производительности - нагрузка центрального процессора, использование памяти, число операций с плавающей запятой. Основой визуализации здесь являются виды отображения, заимствованные из статистической графики. Также бывает необходимо представление ряда других сущностей, например, графов вызовов.

Отметим определённое несоответствие между усилиями по обеспечению огромных возможностей системной составляющей отладки и ограниченными возможностями по выбору методик визуализации. Даже в тех случаях, когда для визуализации использовались среды виртуальной реальности, сами виды отображения были основаны на выводе графиков функций. В настоящее время, судя по анализу текущей литературы, интерес к проблематике визуализации для отладки производительности несколько снизился. Одной из причин этого явления можно считать перенос внимания на методики автоматической генерации кода, автоматического распараллеливания и автоматизированной отладки, где также, но менее активно, используется визуализация. Однако в практической работе серьёзных фирм применяются системы отладки и настройки производительности параллельных систем, активно использующие возможности визуализации. Поэтому именно отладка производительности параллельных программ является основной сферой приложения усилий проектировщиков новых методик визуализации. Перейдём к анализу тех решений в плане визуализации, которые основаны на трехмерных метафорах.

3. Метафора комнаты

Интерес к метафоре комнаты возник сразу после удачного внедрения метафоры рабочего стола. Кроме того, в качестве источника метафоры комнаты можно рассматривать представление различных помещений в многочисленных компьютерных играх. В рамках этой метафоры также имело место применение средств виртуальной реальности. Есть целый ряд интересных решений пользовательского интерфейса с использованием метафоры комнаты в двумерном варианте. (например, Однако в этих реализациях нет сильных отличий от метафоры «рабочего стола» в плане визуального представления, а следовательно нет и выигрыша в выразительности.

Перспективным представляется использование различных модификаций метафоры комнаты в системах визуализации программного обеспечения, особенно в случае параллельных вычислений.

К системам, неявно использующим метафору комнаты в трехмерном варианте, относится система Avatar [3], предназначенная для отладки производительности параллельных программ и функционировавшая на базе театра виртуальной реальности CAVE. Эта визуальная среда была предназначена для представления очень больших объемов данных о производительности параллельных систем, получаемых непосредственно в ходе их работы. Пользователь, оказывается внутри трехмерного помещения, где на стены проецируется видеоизображение. На внутренних гранях показаны оси, а на гранях куба -- полу и стенах помещения выводятся кривые, описывающие метрики производительности параллельной программы. Реализован режим прозрачности потолка и пола. Рассматривается объединение отдельных элементов (scattercube--матрица), которое напоминает стеклянный небоскреб, каждая из комнат которого содержит трехмерный вывод, описывающий различные аспекты поведения параллельной программы. Путешествие («виртуальный полет») по небоскребу дает возможность полного исследования данных о производительности прикладной параллельной программы. Обратим внимание на то, что в системе Avatar данные о производительности параллельных программ отображаются в виде традиционных двумерных графиков.

Нами предпринимался ряд попыток реализации метафоры комнаты в макетных системах визуального программирования и настройки производительности [4].

В одной из таких систем метафора комнаты использовалась в рамках системы визуального программирования. Пиктограммы, размещенные на стенах, представляли управляющие конструкции и данные программной функции. Связи между конструкциями показывалась в пространстве, а не на плоскости, как в традиционных двумерных диаграмматических и икониче-

ских системах визуального программирования. Вся программа в этом случае представлялась зданием (по примеру Avatar), а интересующая пользователя комната--функция вытягивалась из здания подобно блоку при блочном строительстве.В другой нашей реализации метафора комнаты применялась при визуализации графа вызовов. Графы вызовов (call graphs) могут использоваться для анализа и настройки производительности сложных программных комплексов, в том числе распределенных и параллельных программ. Отображая различные характеристики работы программы в характеристиках комнаты, получим некое комбинированное отображение, по которому можно судить о «слабых» в смысле эффективности работы местах программы. Можно обнаружить такие важные часто вызываемые функции, функции, в которых программа «проводит» значительное время, функции, ответственные за передачу данных между процессами (это важно для синхронизации программы и минимизации простоя узлов многопроцессорного вычислителя) и так далее. Существуют и другие применения графа вызовов, например, при анализе потенциального параллелизма программы. Граф вызовов был представлен как набор связанных между собой помещений. Каждая комната - визуальное представление функции. Кроме того, за функцию «отвечает» и пиктограмма на стене комнаты функции--родителя в графе вызовов. Ребра графа естественным образом переносятся на наше изображение --- они следуют от пиктограммы функции на стенке родителя к комнате, соответствующей данной функции. Наибольшую реалистичность изображения дает вид отображения «изнутри» комнаты в сочетании с возможностью «путешествия» между комнатами.

Приведем обзор свойств метафоры и методов представления с ее помощью информации. Метафора комнаты обладает такими свойствами:

1. Способность содержать какие-либо объекты внутри себя. Комната не только представляет собой отдельный объект, но и является контейнером для других объектов.
2. Ограничение контекста восприятия. Объекты внутри комнаты можно рассматривать в отрыве от «внешнего мира».
3. Замкнутость. Для работы с метафорой комнаты не требуется дополнительных элементов, кроме, возможно, объектов, расположенных внутри комнаты.
4. Включение в структуру. Из комнат можно «строить здания», рассматривая совокупность комнат. Поэтому комната может являться элементом построения некоторой сложной конструкции.
5. Естественность метафоры. Комната является естественной метафорой, с наличием соответствующего объекта в реальном мире. Это свойство делает интуитивно понятными все вышеописанные свойства. При использовании метафоры не возникает дополнительных аналогий и неестественных образов. Функциональность и характеристики реального объекта просто переносятся (хотя и в несколько расширенном понимании) в виртуальный мир.

Так как комната есть контейнер, то естественно в качестве первичного способа представления использовать помещение в нее объектов. С одной стороны информация может быть представлена типом (видом) объекта, без учета его размещения, с другой стороны можно рассматривать однотипные объекты, а основную информационную нагрузку будет нести их местоположение в комнате. Информацию может нести и интерьер комнаты. Под интерьером можно понимать такие характеристики, как цвет и геометрическая форма.

Динамическое изменение характеристик комнат во времени может являться источником информации. Возможна анимация в рамках всей совокупности комнат. Кроме того, анимация может затрагивать не только изменение пространственного положения, но и других характеристик комнат - цвета, размера, формы и т.д.

Можно взглянуть на параллельную программу не с позиции процессов, а с позиции ее данных. Кажется удобным с помощью метафоры комнаты описывать составные структуры данных программы (такие как массивы, структуры, объекты). Пользуясь свойством комнаты как контейнера, разместим внутри нее элементы составных типов данных. Кажется, что наибольший эффект от такого визуального представления данных можно ожидать в параллельных системах с общей памятью. Здесь также напрашивается рассмотрение динамики. Таким образом, будет возможность изучить как (и что также немаловажно - где) меняются данные. Это упрощает, например, поиск мест программы, где обрабатывается и/или пересылается большое количество данных.

4. “Физические” метафоры визуализации

В системах визуализации программного обеспечения, предназначенных для отладки производительности параллельных программ применяются метафоры, основанные на аналогиях с физическими системами. Одна из них реализована в среде настройки производительности Kanoko [5] и основана на сближении с динамической механической системой. Предлагается метод анимации, отображающий значения состояния параллельной программы, взятые из ее трассы, в набор состояний динамической модели системы. Затем воспроизводится работа параллельной программы (моделируется динамическая система), а результаты моделирования визуализируются (а также сонифицируются). Анимация показывает изменения в балансе между вычислениями и обменами как обычные движения некоторой динамической системы. В частности определяются используемая динамическая система и отображение элементов и состояний параллельной ЭВМ на тела и силы динамической системы. Моделирование параллельной ЭВМ ведется на основе топологии ее (физической) сети. Так, для параллельного компьютера вычислительные модули и производимые на них вычисления, коммуникационная сеть и количество обменов могут отображаться на тела и их массу, пружины, связывающие тела, и силы притяжения между телами соответственно.

В чём-то аналогичная идея (метафора молекулы) была предложена нами для представления графа вызовов [4]. Трёхмерная визуализация графа предопределила ряд решений. Вершины графа естественно отображать сферами, а связи между вершинами - стрелками. Будем стараться расположить вершины таким образом, чтобы структура графа просматривалась/воспринималась человеком наиболее достоверно и интерпретировалась без особого умственного напряжения. При этом ясно, что случайное распределение не годится, так как структура графа при этом не просматривается. Можно использовать аналогии из физики. Мы знаем что любая система в природе стремится к минимуму своей потенциальной энергии, как некой характеристики, определяющей глобальную устойчивость системы. Можно считать, что силы в природе стараются восстановить максимальную симметрию системы. Для получения оптимального расположения вершин необходимо свести к минимуму потенциальную энергию нашей системы. Для ее определения введем взаимодействие, для чего в вершинах разместим электростатический заряд, а связи между вершинами заменим упругим взаимодействием. Итак, все заряды/вершины отталкиваются друг от друга, а притягиваются только в том случае, если между ними есть связь. Отталкивание всех вершин от всех позволяет достичь глобальной симметрии системы. Наличие центральнй симметрии позволяет разбить взаимодействие попарно.

После ввода взаимодействия введём потенциальную энергию системы как сумму потенциальных энергий ее составляющих. (То есть сумма потенциальных энергий взаимодействия всех пар вершин.) Полученную метафору можно назвать «метафорой молекулы», так как при данном подходе можно получить правильную визуализацию структур некоторых молекул (например, бензола). Сводить к минимуму потенциальную энергию будем путем сведения к нулю всех частных производных по координатам. Заметим, что вариация с величиной зарядов в вершинах, коэффициентом упругого взаимодействия, существенного изменения конечной структуры наблюдаться не будет. Однако, коэффициент упругости влияет на близость вершин друг от друга, а заряд на степень удаленности остальных вершин от данной, и их конечно стоит учитывать в качестве статических/структурных характеристик. Например заряд выставлять равным числу связей с данной вершиной, коэффициент упругого взаимодействия - «мощность» связи.

Соответственно, одной только установкой заряда в зависимости от времени выполнения определенной функции при визуализации графа вызовов существенного изменения картины мы не получим. Посмотрев первый раз на картину при выбранных таким образом параметрах, мы не сможем определить какая-же из функции заняла больше всего процессорного времени. Поэтому для качественной визуализации таких количественных показателей правильнее воспользоваться традиционными методами, например, время работы функции показывать как размер вершины. Возможно использование цвета для выделения/подсветки интересующих особенностей визуализируемого графа. Анимация (вращение молекулы) позволяет изучить структуру графа.

Алгоритм визуализации графа позволяет отобразить (и, главное, интерпретировать) графы с сотнями вершин.

5. Метафора ландшафта

Метафора ландшафта, метафора географического или городского пространства хорошо известна в информационной визуализации и визуализации программного обеспечения.Она может быть использована, например, для представления программных объектов при проектировании и разработке больших программных проектов. Пространственная метафора дает возможность получить представление структуры программного проекта в виде четкой и ясной карты некоторой местности. Компоненты программного обеспечения представляются как условные значки, имеющие точное пространственное местоположение в географическом пространстве. Такая карта может служить прекрасным средством для развития проекта, а также средством визуализации результатов работы программного инструментария. В работе [6] представлен хороший обзор использования метафоры ландшафта для нужд инженерии программного обеспечения. Наш дополнительный анализ позволяет выделить ее следующие свойства:

1. Неограниченный контекст.

При работе с метафорой ландшафта контекст пользователя не ограничивается искусственным образом. В результате, требуются дополнительные усилия со стороны пользователя для идентификации того или иного объекта среди множества других.

2. Естественность метафоры.

Естественность метафоры уменьшает усилия по интерпретации результирующего изображения. Применительно к метафоре ландшафта можно упомянуть кроме естественности самой метафоры (естественность пространственной ориентации), еще и естественность навигации при взаимодействии с изображением. Это делает метафору ландшафта хорошим кандидатом при построении систем виртуальной реальности.

3. «Вложенность» ландшафтов, организация внутренней структуры

Метафора ландшафта позволяет структурировать данные, «вкладывать» их в более крупные составляющие. Метафора ландшафта предполагает вывод достаточно большого объема информации, причем в большинстве случаев достаточно однородной в визуальном смысле. Для интерпретации этой информации пользователем необходимо наличие ключевых элементов, которые являются «отправными точками» для интепретации всех объектов. В случае применения метафоры для выделения неких «особенных» (например ошибочных) элементов, они и будут теми самыми «ключами», они должны быть визуально выделены.

Отметим также, увеличение оправданности использования метафоры ландшафта при увеличении объема информации. Все это показывает возможность ее использования также при проектировании средств визуализации для систем отладки и настройки производительности.

6. Заключение

В заключении отметим, что трёхмерность не является необходимой во всех случаях вывода данных о производительности. Наш опыт также показал, что трёхмерность представления в некоторых случаях оказалась просто излишней, там где можно обойтись плоскими графическими образами. Это связано с тем, что при разработке средств визуализации в системах отладки эффективности виды отображений определяются в первую очередь структурой визуализируемых данных, а в более общем смысле - моделью оценки производительности. Не требуется изобретать излишне сложные специализированные виды отображений для сравнительно простых данных. Ясно, что простые данные (например, скаляры) следует визуализировать простыми же общеупотребимыми способами. Вид отображения для структурированных данных не должен вводить пользователя в заблуждение, в смысле искажения структуры исходных данных. Структура визуализируемых данных также является объектом визуализации, как и сами данные в привычном нам смысле слова. Интерпретация визуализации и интерактивных манипуляций, построенных на базе данной метафоры предполагает, что структура построенного визуального образа не должна противоречить структуре исходного объекта. При интерпретации визуального представления объекта не должно возникать отношений, отсутствовавших в его прообразе.

Литература

1. Авербух В.Л., Байдалин А.Ю. Разработка средств визуализации программного обеспечения параллельных вычислений. Визуальное программирование и визуальная отладка параллельных программ // Вопросы атомной науки и техники. Сер Математическое моделирование физических процессов. 2003. Вып. 4. С. 68--80.

2. Авербух В.Л., Байдалин А.Ю. Разработка средств визуализации программного обеспечения параллельных вычислений. Оптимизация программ // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. вып. 1. С. 70-80.

3.Reed D., Scullin W., Tavera L., Shields K., Elford Ch. Virtual Reality and Parallel Systems Performance Analysis // IEEE Computer, V.28, N 11, (November 1995) pp. 57-67.

4. Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р., Казанцев А.Ю., Поддубная С.В. Трехмерное представление графа вызовов функций // Супервычисления и математическое моделирование: Тезисы международного семинара. Саров, 2003. ВНИИЭФ-РФЯЦ Стр. 12-13

5. Osawa N. An Enhanced 3-D Animation Tool for Performance Tuning of Parallel Programs Based on Dynamic Models // SPDP 98 Welches Or. USA. Pp. 72—80.

6. Balzer M., Noack A., Deussen O., Lewerentz C. Software Landscapes:Visualizing the Structure of Large Software Systems // Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004), pp. 261-266.

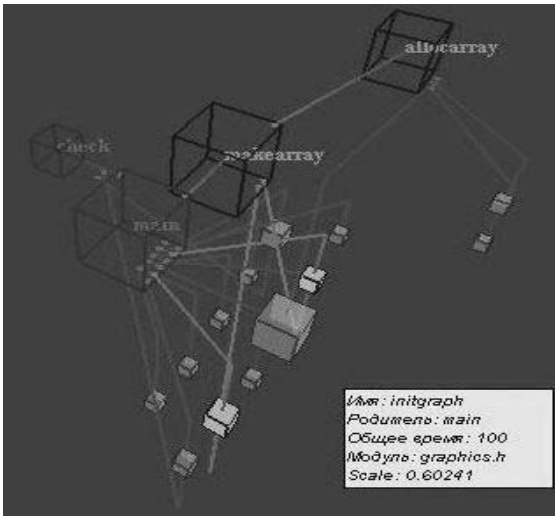

Рис. 1: Вывод графа вызовов при помощи метафоры комнаты

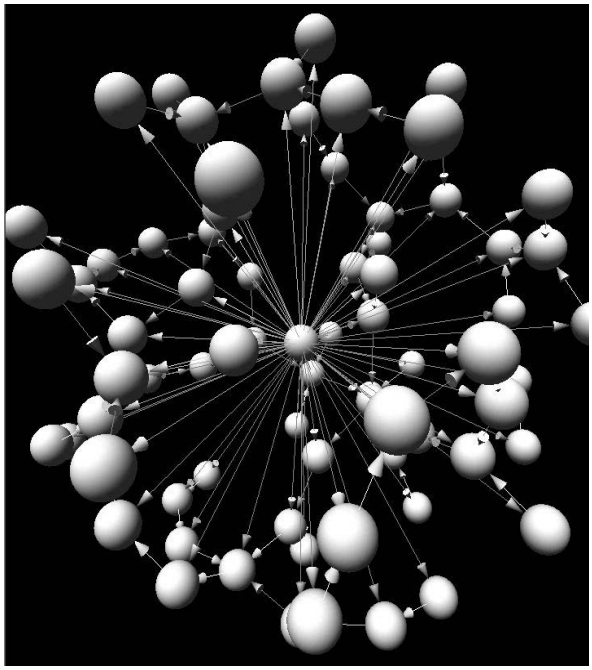

Рис. 2: Вывод графа вызовов при помощи метафоры молекулы