УДК 519.6

АНАЛИЗ И ОЦЕНКА СИСТЕМ ВИЗУАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

В. Л. Авербух, О. Г. Анненкова, М. О. Бахтерев, Д. В. Манаков
(ИММ УрО РАН, УрФУ, г. Екатеринбург)

Рассмотрены подходы к визуализации трасс и графов вызовов *параллельных* программ на основе использования ряда метафор визуализации программного обеспечения. Проведен анализ применимости метафор на основе критериев оценки визуализации. Приведены примеры использования средств визуального сопровождения разработки системного программного обеспечения нижнего уровня для современных процессоров с параллельной архитектурой.

Авторы ставят проблему формального описания и/или верификации визуализации.

Ключевые слова: визуализация программного обеспечения, трасса выполнения, граф вызовов, метафора визуализации.

Введение

Под визуализацией программного обеспечения (ПО) — *Software Visualization* — понимается совокупность методик использования компьютерной графики и средств взаимодействия человека и машины, применяемых для спецификации и представления программных объектов и сущностей в процессе создания, отладки и анализа программ, а также для эффективной эксплуатации ПО. Наиболее перспективными представляются системы визуализации ПО параллельных и распределенных вычислений, прежде всего средства отладки правильности и эффективности *параллельных* программ [1—3]. При этом возникает задача отображения фактической структуры алгоритмов и программ, состояния потоков управления и данных, отдельных элементов структур данных, последовательности обменов и других программных событий, для чего естественно применение анимации.

Изучение литературы, посвященной практике использования средств визуализации ПО параллельных вычислений, указывает на определенный застой в этой области. Попытки создать универсальные системы, предпринятые в 90-е годы прошлого века, в целом не увенчались успехом, что объясняется целым рядом причин, в том числе недостаточным вниманием со стороны проектировщиков к тем аспектам систем, которые можно отнести к проблемам человеческого фактора. В рассматриваемом случае человеческий фактор касается в основном задач проектирования адекватных видов отображения для сред визуализации, а также трудностей восприятия больших объемов сложно структурированных визуальных данных. В связи с этим необходима система качественных и формализованных оценок визуализации.

Данная работа посвящена вопросам анализа и оценки систем визуализации ПО, прежде всего методов представления трасс и графов вызовов параллельных программ.

Визуализация трасс и графов вызовов программ

Трассы программ и их графы вызовов применяются в том или ином виде во многих отладочных системах для описания динамики выполнения программ. Трасса программы отображает динамику конкретного выполнения программы. Визуализация и "проигрывание" трасс программ является важным элементом отладочных систем. Визуальное представление графа вызовов активно используется в системах отладки (настройки) производительности параллельных программ.

В системах отладки 70-х и 80-х годов XX века часто применялись методы, основанные на представлении программы в виде какой-либо схемы или диаграммы. Соответственно трасса отображалась в виде "прыжков" по схеме/диаграмме с изменением интенсивности закраски фона элемента схемы, на который переходило управление. Также имело место использование "прохода" (точнее "пробега") по тексту программы с выделением цветом текущей позиции. В случае высокопроизводительных вычислений такие методы визуализации малопригодны.

В целом ряде систем, включая разработанные для "промышленного" счета в последние годы, реализованы наборы комплексных видов отображения, включающих использование различных модификаций статистических диаграмм [4]. В системе Zinsigh [5] имеет место отображение потока событий в виде последовательности цветных линий (рис. 1). Диаграммы используются для визуализации статистики выполнения по событиям и схем, отображающих последовательность переключения контекста. В публикациях [6, 7] описывается использование набора двумерных видов отображения для визуализации трасс сложных программных комплексов. Этот набор включает диаграмматические и текстовые представления как структуры программы, так и ее выполнения за счет показа последовательности событий. Анализ и интерпретация проводятся пользователями в ходе (и за счет) взаимодействия с визуальными объектами. Для увеличения возможностей по восприятию данных используется анимация. Так, в системе SYNCTRACE [8] для представления трассы многонитевых программ применены двумерные виды отображения, основанные на модификации круговых диаграмм и содержащие возможность анимации выполнения изучаемых участков кода (рис. 2). В работах [9, 10] для представления трассы также используются комплексные виды отображения, сочетающие представление вызовов основных программных сущностей системы в хронологическом порядке и методику визуализации, опирающуюся на метафору *круговых узелков* (circular bundles). При этом используются специальные круговые диаграммы, в которых все сущности проецируются на окружность, а отношения между ними отображаются в виде сплетений (узелков) в центре (рис. 3).

Полученные виды отображения оцениваются с применением критерия качества информационной визуализации, основанного на весь-

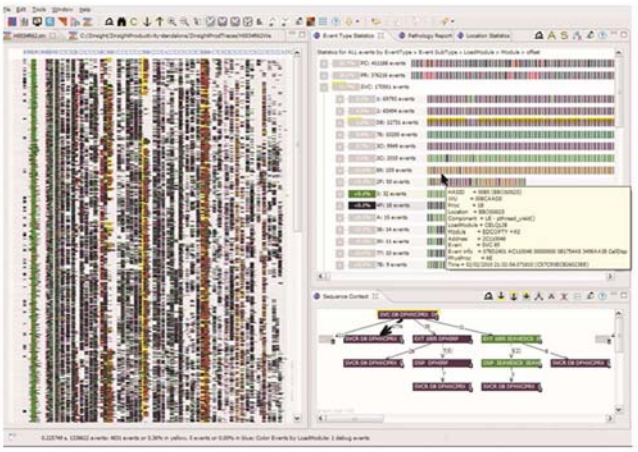

Рис. 1. Визуализация трассы в системе Zinsight [5]

Рис. 2. Основное окно системы SYNCTRACE [8]

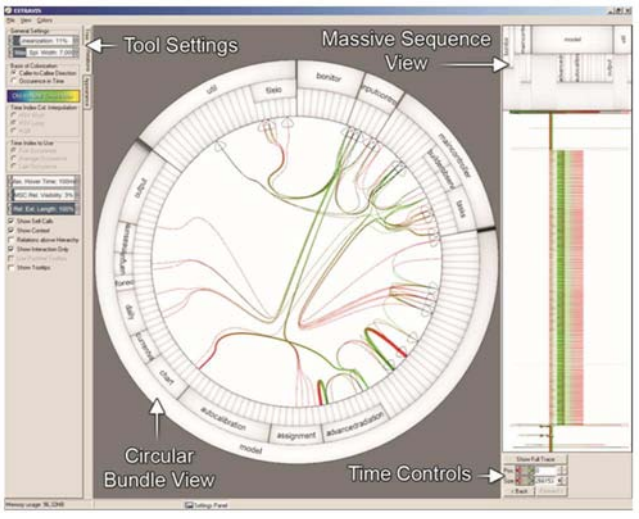

Рис. 3. Визуализации трассы в виде узелков [9, 10]

ма популярной схеме анализа визуальных данных, предложенной известным ученым Б. Шнейдерманом. Так называемая *мантра Шнейдер-*

мана — последовательное повторение действий по обзору, изменению масштаба, фильтрации и детализации ("Overview first, zoom and filter, then details-on-demand"), позволяющих проанализировать большие объемы визуализированных данных [11]. Критерий качества предусматривает проверку системы визуализации на возможность выполнения мантры Шнейдермана. (При создании критерия в набор действий включены также возможности вывода зависимостей, получения истории поиска и извлечения подмножества из рассматриваемых данных [12].)

Рассмотрим примеры естественных и физических метафор визуализации, использованных в отладочных системах для описания динамики выполнения программ. Неформально под метафорой визуализации понимается главная идея при отображении прикладной области на визуальный мир. Метафоры используются для определения деятельности пользователя программной системы и его восприятия объектов и операций над ними. В принципе любая визуализация является метафоричной [13, 14], но в литературе часто противопоставляются традиционные методы отображения данных и новые (метафорические) идеи, помогающие уяснить сложные и абстрактные понятия и лучше увидеть отношения между объектами.

При удачном выборе естественная или физическая метафора может значительно повлиять на удобство использования визуализации, но может и породить сложные и плохо интерпретируемые виды отображения. Также представляют интерес возможные подходы к оценке метафор на этапе проектирования систем и выбора методов визуального представления сущностей параллельного программирования.

Существуют примеры использования метафор для представления трасс и графов вызовов программ. В частности, при создании систем визуализации ПО параллельных вычислений используются метафора притяжения/отталкивания (в качестве ее модификаций можно рассматривать метафоры молекулы и физической частицы), метафоры комнаты, здания, города (их модификации — метафоры ландшафта и фабрики).

В системе Kanoko [15] был реализован метод анимации, отображающий значения состояния программы, взятые из ее трассы, в набор состояний динамической модели системы. Затем воспроизводилось выполнение параллельной программы (моделировалась динамическая система), а результаты моделирования визуализировались (а также сонифицировались). Анимация представляла изменения в балансе между вычислениями и обменами как обычные движения некоторой динамической системы. Таким образом, имело место использование понятной для пользователя *метафоры притяжения/отталкивания*. В частности, определялись используемая динамическая система и отображение элементов и состояний параллельного вычислителя на тела и силы динамической системы. Моделирование вычислителя велось на основе топологии его физической сети. Так, для параллельного компьютера вычислительные модули и производимые на них вычисления, коммуникационная сеть и количество обменов могли соответственно отображаться на тела и их массу, пружины, связывающие тела, и силы притяжения между телами.

В ряде публикаций последнего времени рассматривается использование *метафор города и ландшафта* для представления трасс программ. Так, в [16] описывается представление трассы для случая параллелизма на основе нитей (рис. 4). Здания отображают статичные части программного комплекса. Трасса представляется в виде лучей-нитей, протянутых между зданиями. Визуализация ошибки deadlock, естественно, выглядит как окрашенное пересечение этих лучей. В работе [17] используются аналогичные идеи визуального представления. Сущности программной системы (например приложения) отображаются в виде зданий, а потокам управления и потокам данных, связывающим эти сущности, соответствуют улицы. Возможно включение/выключение режима проигрывания трассы. Метафора города может также применяться для представления структуры программных комплексов, как, например, в [18].

Отметим оригинальную *метафору мозга* (*"Brain" Metaphor*), использованную для анимационного представления выполнения программы в работе [19]. Идея визуализации работы мозга при предъявлении ему каких-либо стимулов перенесена на визуализацию активности программы или приложения (вызов процедур и функций, ввод/вывод и пр.) (рис. 5).

Визуализация графов вызовов должна обеспечить выявление причинных связей, порядка вызовов модулей, повторов, принадлежности данных к тому или иному типу, а также других отношений, необходимых разработчикам при от-

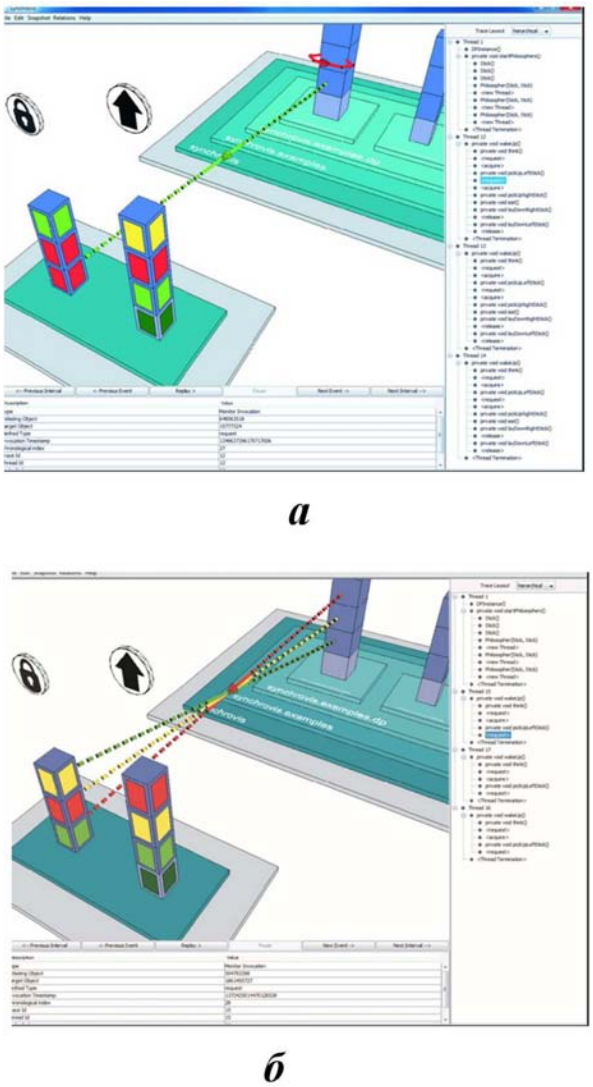

Рис. 4. Метафора города. Визуализация выполнения программы [16]: *а* — нормального; *б* — ошибочного (deadlock)

ладке правильности и эффективности параллельной программы [20].

Традиционно для представления графов вызовов используются двумерные виды отображения, построенные с помощью диаграмм, связанных стрелками. Однако при двумерном представлении графа вызовов значительной по объему и сложной по структуре программы с большой глубиной вложенности вызовов функций и большим количеством пользовательских функций возникают сложности в отображении на экране монитора и анализе пользователем конечного изображения.

В рамках двумерного представления можно справиться с данными проблемами за счет дополнительного инструментария, например дополнительного диалога с системой при навигации по графу [20]. Нехватку места на экране дисплея можно преодолеть, добавив к разрабатываемой модели еще одно измерение. В [21, 22] используется "дву-с-половиной-мерная" графика для представления графа вызовов. В этом случае узлы графа отображаются в виде примитивных изображений зданий, а связи между ними проводятся как бы по воздуху. Полученные графические выводы в какой-то мере напоминают отображения на базе *метафоры фабрики* (рис. 6).

В авторской работе [23] описаны трехмерные представления графа вызовов на базе *метафоры здания*, когда изображаются связанные между

Рис. 5. Использование метафоры мозга для представления выполнения программы (скриншот анимации) [19]

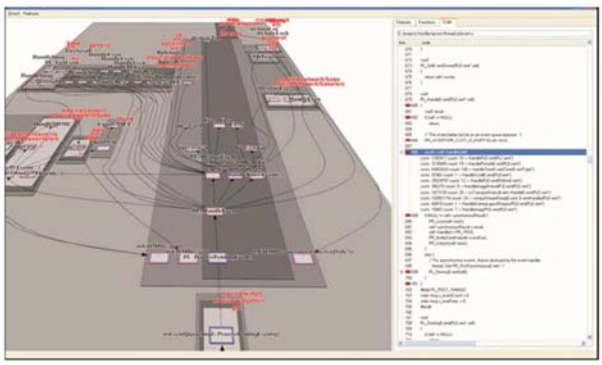

Рис. 6. Граф вызовов и текстовое описание кода [22]

собой комнаты некого здания сложной архитектуры (рис. 7, см. также цветную вкладку). Расположение комнат трехуровневое. Все функции программы разбиваются на три части: 1) функции пользователя, имеющие в качестве наследников пользовательские функции; 2) функции пользователя, не имеющие таковых; 3) системные функции. Каждой такой части соответствует свой уровень в результирующем изображении. Каждая комната — визуальное представление функции. Кроме того, о функции свидетельствует пиктограмма на стене комнаты функции-родителя в графе вызовов. Наибольшую реалистичность изображения дает вид отображения изнутри комнаты в сочетании с возможностью путешествия внутри здания между комнатами. Но в этом случае пользователь лишен структурного, объемного видения графа. Сочетание этих двух видов отображения графа может дать более приемлемый результат.

В [23] также описано использование *метафоры молекулы*. Эта метафора аналогична метафоре притяжения/отталкивания. Вершины графа естественно отображать сферами, а связи между вершинами — стрелками. Все вершины отталкиваются друг от друга; притягиваются они только в том случае, если между ними есть связь. Введенный коэффициент упругости влияет на близость вершин друг к другу, а заряд — на степень удаленности остальных вершин от данной. Эти параметры необходимо учитывать в качестве статических/структурных характеристик, например заряд задавать равным числу связей с данной вершиной, а коэффициент упругого взаимодействия — мощности связи. Но одной только установкой заряда в зависимости от времени выполнения определенной функции при визуализации графа вызовов существенного изменения картины не получить. Для визуализации количественных показателей правильнее воспользоваться традиционными методами. Время выполнения функции следует представлять как размер вершины. Возможно использование цвета для выделения/подсветки интересующих особенностей визуализируемого графа. Анимация (вращение молекулы) позволяет изучить структуру графа. Алгоритм визуализации графа дает возможность отобразить (и, главное, интерпретировать) графы с сотнями вершин (рис. 8).

В рассмотренных выше системах, как правило, используются комплексные виды отображения, в которых вместе с графическими объектами, построенными на базе той или иной метафоры, отображается текст программы и/или ее структура. В некоторых системах виды отображения строятся с учетом конкретного типа распараллеливания. Например, в системе SYNCTRACE [8] метафора визуализации специально построена для отображения выполнения многонитевой параллельной программы. В других случаях имеет место адаптация и дополнение существующих метафор к задачам анали-

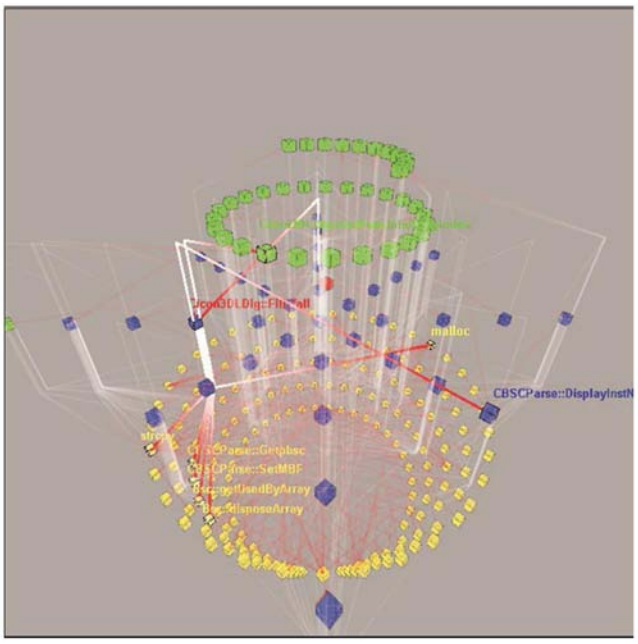

Рис. 7. Граф вызовов системы рисования графа вызовов на базе метафоры здания [23]

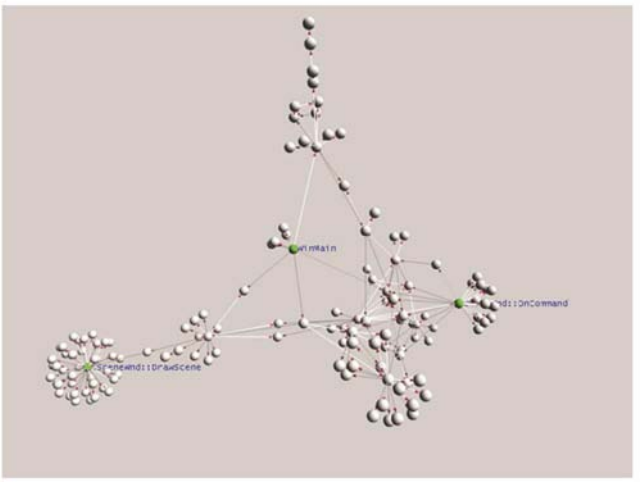

Рис. 8. Граф вызовов системы рисования графа вызовов на базе метафоры молекулы [23]

за программ без существенной привязки методик визуализации к типу параллелизма. Так, в [16] для представления выполнения многонитевых параллельных программ используется слегка адаптированная метафора города.

В большинстве рассмотренных работ нет подробных данных о реализации средств визуализации. Однако их анализ позволяет сделать некоторые заключения.

При отображении данных о многонитевых задачах на общей памяти рассматриваются развернутые по времени стеки вызовов (т. е. графы вызовов с учетом того, в какой последовательности они происходят). Применяются некоторые эвристики, по которым можно судить о связи нитей. Инструмент профилирования стандартно должен собирать информацию о том, из какой функции/процедуры произошла передача управления, в какую функцию управление пришло и когда произошли эти события. Именно такой набор событий на базе работы профилировщика отображается системами визуализации.

Для случая модели передачи сообщений (MPI-решений) очевидно использование стандартной идеи отображения событий типа Send/Receive. Было бы полезным совмещение в одном виде отображения трассы MPI с информацией от профилировщика. Однако в этом случае визуальное изучение связей между процессами при большом числе процессов затруднительно. Здесь проявляется одна из серьезных проблем визуализации ПО — проблема масштабирования видов отображения.

Сами по себе метафоры визуализации, как правило, не имеют привязки к конкретным языкам или средам программирования. На основе метафоры проектируется вид отображения, который включает в себя описание набора возможных объектов визуализации, их взаиморасположение на экране, а также возможное взаимодействие с ними. Именно при разработке видов отображения необходим учет конкретной задачи отладки и анализа данных.

Анализ применимости метафор визуализации

Проектировщику необходимо получить ответы на вопросы о том, как данная метафора может помочь в представлении данных и взаимодействий с ними, каковы свойства визуальных объектов, созданных на базе данной метафоры, к каким результатам приводит пользователя взаимодействие с метафорическими объектами [24]. Следует проанализировать возможности метафор по представлению больших и очень больших объемов данных и деталей, необходимых для понимания процесса выполнения программ. Важно понять, какие объекты могут быть представлены с помощью данной метафоры. Интересны возможности применения метафор в рамках систем, использующих современные среды компьютерной графики, в частности виртуальной реальности. Для всего этого необходимо описать способы проверки пригодности метафор при решении конкретных задач.

Рассмотрим сходные метафоры города и ландшафта. Среди их свойств можно выделить следующие:

- неограниченный контекст. При визуализации большого количества данных позволяет бегло окинуть взглядом всю "картинку" и выделить ключевые места;
- естественность. Имеет место как в пространственной ориентации, так и в навигации;
- наличие организованной внутренней структуры. В случае метафоры города эта структура задается достаточно жестко — есть дома, кварталы, улицы, районы. Метафора ландшафта не накладывает ограничений на выбор структуры. В этом случае можно говорить о вложенности ландшафтов;
- наличие ключевых элементов. Допускается представление достаточно большого объема однородной в визуальном смысле информации. При этом возможно использование ключевых элементов, которые являются "отправными точками" для интерпретации всех объектов. Например, некие особенные (например ошибочные) элементы должны выделяться среди множества других цветом, размером, формой и т. п.;
- устойчивость к масштабированию. Данные метафоры не только устойчивы к увеличению объема предоставляемой информации, но и предполагают изначально наличие ее большого объема (чтобы применение метафор было оправданным).

В случае реализации метафор города и ландшафта в системах визуализации ПО транспортные артерии могут использоваться для представления потоков управления, потоков данных и

иных связей между программными конструкциями или частями программного комплекса.

Таким образом, метафоры города и ландшафта могут служить базой при представлении значительных объемов структурированной информации с выделением случаев особого интереса, что необходимо при отладке правильности и эффективности параллельных программ. Возможность реализации пролета над городом/ландшафтом позволяет осуществлять навигацию, полета с изменением высоты — масштабирование и зуминг. Таким образом, с помощью метафор города и ландшафта можно создавать системы визуализации, удовлетворяющие критериям, основанным на схеме Шнейдермана. При этом интерпретация полученных на их основе графических выводов представляется простой.

Аналогично рассматриваются свойства метафоры молекулы (частицы), которая также может использоваться для задач визуализации трасс выполнения и графов вызовов параллельных программ.

Данная метафора предполагает представление большого объема *структурированной* информации. Интерпретация физической метафоры молекулы (частицы) и ее модификаций в целом проста и *естественна*, хотя и требует от пользователя некоторых (элементарных) знаний по физике. Связи между объектами визуализируемой модели также *естественно* представляются как связи между атомами/частицами. Виды отображения, построенные на базе метафоры молекулы (частицы), *устойчивы к масштабированию*. Метафора предполагает возможность выделения *ключевых элементов*, например за счет цвета или размера элементов молекулы и толщины связей между ними. Перемещение и навигация в полученных графических выводах могут выполняться за счет облета молекулы (совокупности частиц). Имеется опыт реализации "входа" в отдельный атом (частицу) и изучения визуальной информации, помещенной внутри отдельной частицы [25].

Таким образом, на базе метафор молекулы и частицы (как и в случае метафор города и ландшафта) возможно построение систем, которые удовлетворяют критериям, основанным на схеме Шнейдермана. Кроме того, возможности метафор города/ландшафта и молекулы/частицы позволяют использовать их при проектировании систем визуализации на базе сред виртуальной и расширенной реальности.

В то же время существует ряд ограничений по применению критериев качества визуализации. Если система им удовлетворяет, то это не означает, что она полностью лишена недостатков и пригодна для внедрения в практику разработки ПО. Например, система, использующая при визуализации методику круговых узелков, вполне соответствует критерию Шнейдермана, но при ее применении, по мнению авторов, получаются запутанные графические выводы. Отметим, что критерий Шнейдермана основывается на проверке только лишь необходимых, но не достаточных условий качества информационной визуализации. Использование схемы, построенной на базе мантры Шнейдермана, предполагает наличие структурированных данных большого, но все же обозримого объема. Кроме того, предполагается, что пользователь либо знает, что ищет, либо сможет это опознать. В случае узелков (как и других диаграмм), при создании которых применена по сути абстрактная и достаточно сложная методика визуализации, интерпретировать графические выводы, как представляется, непросто: пользователь должен все время соотносить картинку с неочевидными методами представления интересующих его данных.

Для высокопроизводительных вычислений многие методы визуализации трассы могут оказаться неэффективными как из-за сложности анализа выполняемого кода, так и из-за чрезвычайно больших объемов трасс выполнения параллельных программ. Подобные доводы можно выдвинуть во многих случаях использования новых абстрактных методов (метафор) визуализации для представления сложных данных. Виды отображения, использующие модификации статистических диаграмм, мало масштабируемы и не позволяют визуализировать выполнение сотен и тысяч процессов современных вычислений. Также можно сделать замечания по использованию естественных метафор. Интерпретация графических выводов, полученных в рамках естественных метафор, например интересной анимационной метафоры мозга, зачастую не представляется очевидной. Естественность образности метафор города и ландшафта в каких-то случаях может отвлекать пользователей систем визуализации. Также остаются проблемы восприятия больших и очень больших объемов информации. Не всегда приемлемо и само наблюдение за мерцающими и мигающими анимационными выводами. У пользователей систем на базе сред виртуаль-

ной реальности могут возникать неприятные последствия в виде головокружения и пр.

Другой подход к оценке визуализации заключается в анализе наличия инсайта (insight — озарение, понимание), т. е. установления в результате использования системы визуализации релевантных отношений между данными и существующей предметной областью (целевой областью) [26]. Визуализация должна создавать или способствовать созданию целостной ментальной модели и, как следствие, инсайта. Можно сказать, что цель метафорических визуализаций заключается в его получении. Однако наличие или отсутствие инсайта весьма субъективно. Поэтому и оценка по этому критерию остается субъективной.

Визуальное обеспечение процесса разработки ПО

Как уже отмечалось, попытки создать универсальные системы визуализации в целом не увенчались успехом. Опыт разработки, прежде всего систем научной визуализации, показывает, что их успех связан с наличием формальной модели визуализируемой области, с представлением о том, как пользователь видит и понимает объекты данной области знаний, а также пониманием (хотя бы в общих чертах), как он будет пользоваться проектируемой системой. Судя по имеющейся в распоряжении авторов литературе, разработчики систем визуального представления трасс исполнения и графов потока данных не учитывают особенностей их использования. В то же время в общем случае визуализации ПО параллельных систем нет ни универсальной формальной модели, ни всеобщей ментальной модели данной области. Также не существует общепризнанных методик работы программистов при отладке правильности или производительности параллельных систем. Поэтому на данном этапе представляются эффективными лишь специализированные системы визуализации ПО с четко поставленными, хотя и ограниченными задачами. Вместо поиска эффектных метафор и разработки универсальных средств визуализации следует перейти к созданию средств визуального сопровождения процессов разработки, отладки и анализа ПО, основываясь на изучении деятельности программистов, работающих в рамках конкретной программно-аппаратной среды.

Опыт в области специализированных средств визуального сопровождения процесса разработки был получен в ходе реализации системного ПО для отечественных процессоров семейства *Мультиклет*. Архитектура процессоров семейства *Мультиклет* обеспечивает простой способ внеочередного выполнения команд, явный контроль программиста за упорядочением памяти, прозрачную для приложений реконфигурацию во время выполнения, отказоустойчивость, в том числе благодаря оригинальному способу кодирования машинных команд. Процессоры семейства *Мультиклет* не задают явной модели упорядочения памяти. Элементарные вычислительные процессы запускаются параллельно [27].

Для упрощения анализа исходных текстов программ на ассемблере для процессоров семейства *Мультиклет* было предложено визуализировать графы различных участков программы. В данном случае нужно было перевести естественным образом код программы в описание графа, например на языке Dot (система GraphViz [28]). Первые результаты оказались недостаточно выразительными, так как перевод осуществлялся без учета особенностей алгоритмов размещения вершин, используемых в GraphViz для визуализации. Постепенно многие из этих особенностей удалось учесть и получить приемлемые для визуализации участки кода, на которых можно проследить интересующий программистов поток данных. Дополнительным результатом оказалась возможность узнавать некоторые алгоритмы по их графическому изображению. Важным для практики является то, что такая визуализация позволяет быстро прослеживать степень параллелизма анализируемого участка кода, к которой процессоры семейства *Мультиклет* "чувствительны". Код с небольшой степенью параллелизма естественным образом отображается как вытянутая нить операций. Такое прослеживание необходимо при отладке производительности [29] (рис. 9).

Другой пример использования визуализации связан с разработкой компилятора C99 для процессоров семейства *Мультиклет*. Разработка трансляторов остается одной из самых сложных задач в программировании. В значительной степени это связано с трудоемкостью отладки корректности формируемых в процессе трансляции нетривиальных структур данных, насыщенных перекрестными ссылками. Число ссылок таково, что использование традиционных пошаговых локальных отладчиков не позволяет

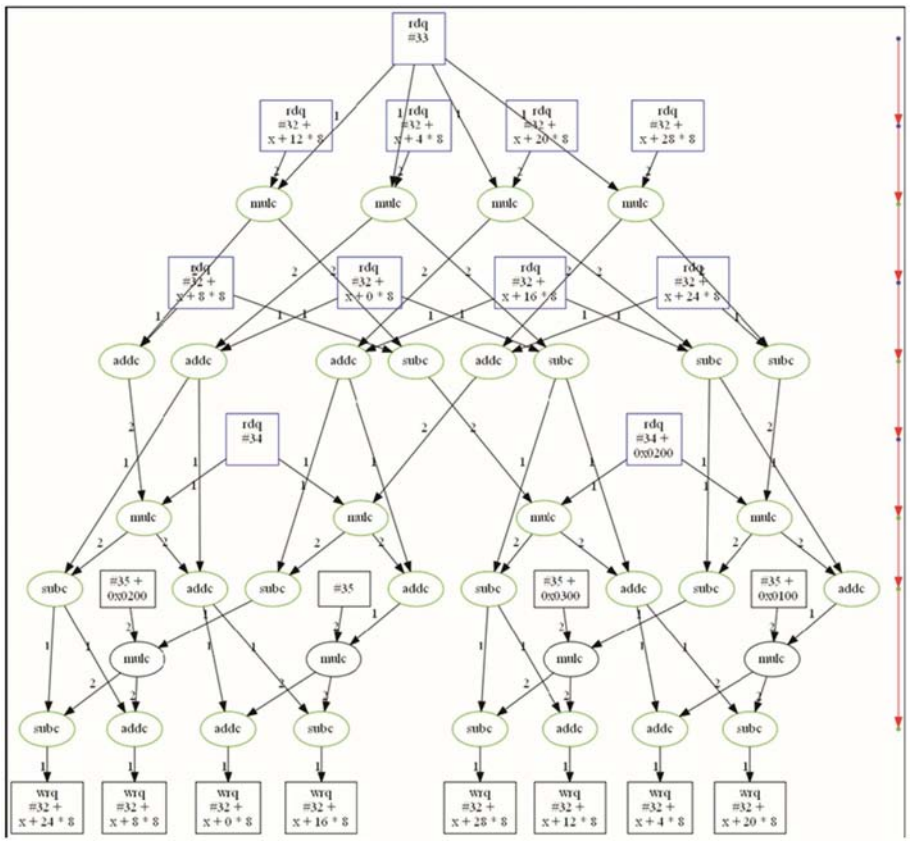

Рис. 9. Визуализация графа участка программы на ассемблере процессора семейства *Мультиклет* [29]

сколько-нибудь эффективно получать информацию о процессе выполнения программы, потому что возможность изучить значения нескольких переменных не дает никакого представления о состоянии программы. Более приемлемой в такой ситуации можно считать отладку по трассе программы. Однако трассы процессов трансляции на C99 даже небольших программ оказываются слишком длинными для непосредственного восприятия человеком. При представлении отладочного дампа в интерактивном виде рассматривались такие подходы, как трансляция в гипертекст и генерация графической анимации. Применение веб-браузеров для анализа структур данных и отладки программ обеспечивает простоту динамической генерации и дает возможность использования интерактивных и навигационных средств. В то же время гипертекст не способен отразить некоторых особенностей исследуемой динамической структуры. Чтобы отследить структурные изменения, отражающиеся на глобальной структуре графа, необходимы трехмерные анимационные отображения. Для построения анимации графов используется графическая библиотека динамической визуализации графов Ubigraph [30]. При визуализации применяется реализованная в данной системе метафора физической частицы. Вершина представляется точечной частицей пространства, координаты рассчитываются по физическим формулам (рис. 10).

Отметим, что в рассмотренных случаях визуализация опирается на формальные описания изучаемых явлений, которые хорошо известны

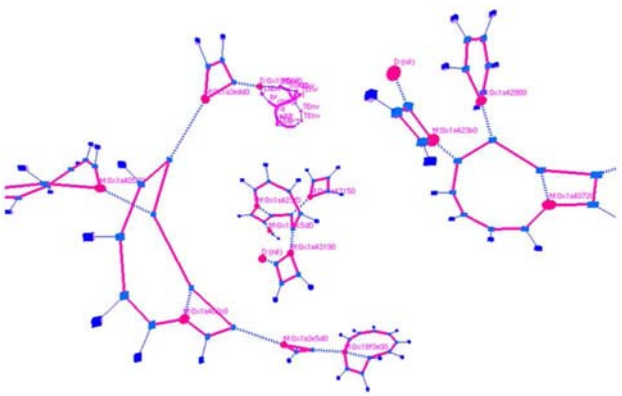

Рис. 10. Визуализация структуры графа трансляции

проектировщикам (они же пользователи системы). Можно говорить о существовании достаточно четкой ментальной модели динамики выполнения программы. Имеет место понимание того, правильно или неправильно работает программа. Программист получает возможность уяснить, в каком месте возникает ошибка и каков ее тип. Так как проектирование системы визуализации выполняли сами разработчики системного обеспечения процессоров семейства *Мультиклет*, то так или иначе были учтены особенности процесса программирования и отладки. В данном случае реализации специализированных средств визуального обеспечения процесса разработки системного ПО присутствуют элементы формальной и ментальной модели параллельного выполнения процессов. Средства визуализации учитывают особенности реальной деятельности программиста по анализу и отладке сложного кода. В каком-то смысле можно говорить о том, что разработанные средства визуализации обеспечивают инсайт. Таким образом, полученная система удовлетворяет критериям оценки визуализации, предлагаемым авторами данной статьи.

Верификация и валидация визуализации

Визуализация ПО является подобластью общей дисциплины — компьютерной визуализации. Визуализация, представляя результаты вычислений, обеспечивает интерпретацию и анализ полученных данных. В связи с развитием компьютерной визуализации ставится вопрос о ее теоретической базе. Теоретические исследования нужны, во-первых, для анализа и оценки существующих систем, во-вторых, для обучения новых специалистов, в-третьих, для обоснования решений при проектировании новых систем. Важной задачей теории визуализации является создание научных основ для качественного и надежного проектирования, разработки и оценки визуальных интерактивных систем. При оценке визуализации можно рассматривать задачи валидации (оценка общей применимости и адекватности применения) и верификации (оценка правильности).

Верификация визуализации подразумевает наличие формальной модели, которая только предметной областью отличается от подобных моделей других областей. Востребована верификация таких слабо формализуемых и связанных с когнитивными аспектами подзадач, как визуализация данных большого объема, интерпретация результатов визуализации. Верификация как сравнение с эталоном широко применяется в научной визуализации, например при разработке инженерных пакетов. Трудно представить, что, решая одни и те же уравнения одними и теми же методами, применяя одинаковые виды отображения, будут получены разные изображения. Однако это возможно в результате накопления вычислительной погрешности. В [31] вводится понятие верифицируемой визуализации, которая отслеживает, как распространяется погрешность (неопределенность) на всех этапах вычислительного конвейера, включая визуализацию. Подобный подход в общем случае принято называть моделью с неопределенностью, в частном случае — визуализацией с неопределенностью. Он может рассматриваться как пример некорректной по начальным данным задачи. Можно выделить качественные и количественные характеристики верификации. Качественную характеристику — степень формализации (наивная, метафорическая, формальная) — и количественную, связанную с распространением неопределенности, уместно трактовать как полноту и точность верификации.

Одновременно с верификацией визуализации необходимо рассматривать валидацию визуализации как меру адекватности. В математическом моделировании адекватность определяется как соответствие формальной модели и реализующего ее ПО тому, что наблюдается на практике. Формально правильная модель может быть неадекватной. Частая путаница в определении верификации и валидации объясняется тем, что наивная верификация эквивалентна валидации.

Заключение

Системы визуализации ПО параллельных вычислений могут выполнять задачи сопровождения процесса разработки ПО на различных этапах. Наиболее перспективным является использование визуализации в системах отладки правильности и производительности параллельных программ. В начале статьи отмечен определенный застой, проявившийся в развитии визуализации ПО параллельных вычислений. Чтобы вернуть интерес к этому направлению, следует решить ряд задач.

Прежде всего, это проблема выбора объектов, подлежащих визуализации в рамках про-

цесса отладки. В случае параллельных вычислений само определение объектов программы, связанных с ее ошибочным состоянием, является сложной задачей. Трасса выполнения программы — лишь одна из возможных сущностей, подлежащих анализу и, как следствие, визуализации. При различных парадигмах параллелизма резко меняется набор и суть анализируемых программных объектов. В случае отладки (настройки) производительности также нет ясности с выбором тех сущностей, изучение которых способно помочь в улучшении производительности.

С самого начала развития средств визуализации программирования встал вопрос о способах графического представления программных объектов. Проектировщики опирались на стандартные ("бумажные") подходы к визуализации ПО либо перекладывали эту задачу на пользователей-программистов, предоставляя им инструментарий рисования. Использование оригинальных метафор оказалось не всегда оправданным, так как нужны простые, но четко интерпретируемые методы представления. Привязка метафор визуализации к конкретным средам программирования требует отдельного рассмотрения. Существует также проблема масштабирования видов отображения, основанных на тех или иных метафорах. "Графовые" метафоры, как и диаграммы различного вида, имеют в данном контексте значительные ограничения. Необходим учет ментальных моделей разработчика [32] и характера его деятельности в процессе реализации ПО. Оправдан переход к созданию специализированных средств обеспечения процесса разработки и отладки программ, учитывающих особенности аппаратуры и языков разработки, а также запросы самих разработчиков.

В связи с этим интересно применение методов, хорошо зарекомендовавших себя в смежных областях визуализации. Метод параллельных координат, используемый в информационной визуализации для описания эффективности экономических моделей [33], может быть применен и при представлении данных об эффективности вычислений, так как непринципиально, анализируется ли мощность двигателя или количество попаданий в кэш. При достаточно мелком разбиении по длине программы или времени для вывода данных о производительности оправдано использование видов отображения, построенных на базе *информационной фрески* (Information Mural) [34]. В [35] представлен интересный пример использования циклограмм для "визуальной верификации" управляющих программ реального времени. В [36] предлагается формализация понятий, связанных с производительностью программ и рассматриваемых как функции многих переменных. Это позволяет использовать методики представления трехмерных объектов, характерных для научной визуализации.

Для создания эффективных и достаточно универсальных систем визуализации ПО параллельных вычислений требуются комплексные усилия исследователей и разработчиков.

Список литературы

1. *Авербух В. Л., Байдалин А. Ю.* Разработка средств визуализации программного обеспечения параллельных вычислений. Оптимизация параллельных программ // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. Вып. 1. С. 70—79.
2. *Авербух В. Л., Байдалин А. Ю., Исмагилов Д. Р., Казанцев А. Ю.* Состояние дел в визуализации программного обеспечения параллельных вычислений // ГРАФИКОН-2005. Труды конференции. Новосибирск: ИВМиМГ СО РАН, 2005. С. 179—186.
3. *Посыпкин М. А., Соколов А. А.* Обзор методов автоматизации мониторинга, анализа и визуализации поведения параллельных процессов, взаимодействующих с помощью передачи сообщений: Препринт № 7. М.: ИСП РАН, 2005.
4. *Mohr B.* Scalable parallel performance measurement and analysis tools — state-of-the-art and future challenges // Supercomputing Frontiers and Innovations. 2014. Vol. 1, No 2. P. 108—123.
5. *De Pauw W., Heisig S.* Zinsight: a visual and analytic environment for exploring large event traces // Proc. of 5th Int. Symposium on Software Visualization. October 25—26, 2010. Salt Lake City, UT, USA: ACM, 2010. P. 143—152.
6. *Trumper J., Bohnet J., Dollner J.* Understanding complex multithreaded software systems by using trace visualization // Ibid. P. 133—142.
7. *Trumper J., Telea A., Dollner J.* Viewfusion: Correlating structure and activity views for

execution traces // Proc. of 10th Theory and Practice of Comp. Graph. Conf. Euro. Asso. for Comp. Graph. University of Rutherford, UK, 2012. P. 45—52.

8. *Karran B., Trumper J., Dollner J.* SYNCTRACE: Visual thread-interplay analysis // Proc. of 1st Working Conf. on Software Visualization. Eindhoven, NL: IEEE Computer Society, 2013.
9. *Cornelissen B., Holten D., Zaidman A. et al.* Understanding execution traces using massive sequence and circular bundle views // Proc. of 15th IEEE Int. Conf. on Program Comprehension. Banff, Alberta, Canada: IEEE, 2007. P. 49—58.
10. *Cornelissen B., Zaidman A., Holten D. et al.* Execution trace analysis through massive sequence and circular bundle views // J. Systems and Software. 2008. Vol. 81, Issue 12. P. 2252—2268.
11. *Shneiderman B.* The eyes have it: A task by data type taxonomy for information visualizations // Proc. of IEEE Conf. on Visual Languages. September 3—6, 1996. Boulder, CO: IEEE, 1996. P. 336—343.
12. *Maletic J. I., Marcus A., Collard M. L.* A task oriented view of software visualization // Int. Workshop on Visualizing Software for Understanding and Analysis. Paris, France, 2002. P. 32—40.
13. *Averbukh V. L., Bakhterev M. O., Baydalin A. Yu. et al.* Searching and analysis of interface and visualization metaphors // Human-Computer Interaction, New Developments / Ed. by K. Asai. Chapter 3. Vienna: In-teh, 2008. P. 49—84.
14. *Knight C., Munro M.* Software Visualisation Conundrums. University of Durham, Computer Science. Technical Report 05/01. July 2001. http://vrg.dur.ac.uk/papers/getpaper.php3?id=27.
15. *Osawa N.* An enhanced 3-D animation tool for performance tuning of parallel programs based on dynamic models // Proc. of SIGMETRICS Symp. on Parallel and Distributed Tools "SPDP 98". Welches Or, USA, 1998. P. 72—80.
16. *Waller J., Wulf Ch., Fittkau F. et al.* SynchroVis: 3D visualization of monitoring traces in the city metaphor for analyzing concurrency // Proc. of 1st IEEE Working Conf. on Software Visualization. Eindhoven, NL: IEEE Computer Society, 2013.
17. *Fittkau F., Waller J., Wulf Ch., Hasselbring W.* Live trace visualization for comprehending large software landscapes: The ExplorViz approach // Ibid.
18. *Kobayashi K., Kamimura M., Yano K. et al.* SArF map: Visualizing software architecture from feature and layer viewpoints // Proc. of 21st IEEE Int. Conf. on Program Comprehension (ICPC). San Francisco, CA, USA: IEEE, 2013. P. 43—52.
19. *Palepu V. K., Jones J. A.* Visualizing constituent behaviors within executions // Proc. of 1st Working Conf. on Software Visualization. Eindhoven, NL: IEEE Computer Society, 2013.
20. *LaToza T., Myers B.* Visualizing call graphs // Proc. of IEEE Symp. on Visual Languages and Human-Centric Computing. Pittsburgh, PA, USA: IEEE, 2011. P. 117—124.
21. *Bohnet J., Dollner J.* Visual exploration of function call graphs for feature location in complex software systems // Proc. of ACM Symp. on Software Visualization. Brighton, UK: ACM, 2006. P. 95—104.
22. *Bohnet J., Dollner J.* Facilitating exploration of unfamiliar source code by providing 21/2D visualizations of dynamic call graphs // Proc. of 4th IEEE Int. Workshop on Visualizing Software for Understanding and Analysis. Banff, Alberta, Canada: IEEE, 2007. P. 63—66.
23. *Авербух В. Л., Байдалин А. Ю., Исмагилов Д. Р., Казанцев А. Ю.* Трехмерные методики визуализации программного обеспечения параллельных и распределенных вычислений // Труды ПаВТ'2008. Челябинск: ЮУрГУ, 2008. С. 283—288.
24. *Авербух В.* Семиотический подход к формированию теории компьютерной визуализации // Научная визуализация. 2013. Т. 5, № 1. С. 1—25.
25. *Авербух В. Л., Байдалин А. Ю., Исмагилов Д. Р. и др.* Использование трехмерных метафор визуализации // Тр. 14-й Межд. конф. по компьютерной графике и зрению "ГрафиКон'2004". Москва, 6—10 сентября 2004 г. М.: МГУ им. М. В. Ломоносова, 2004. С. 295—298.

26. *North Ch.* Toward measuring visualization insight // IEEE Computer Graphics and Applications. 2006. Vol. 26, No 3. P. 20—23.
27. *Стрельцов Н. В.* Архитектура и реализация мультиклеточных процессоров // Тр. V Межд. науч. конф. "Параллельные вычисления и задачи управления". Москва, 26—28 октября 2010 г. М.: Институт проблем управления им. В. А. Трапезникова РАН, 2010. С. 1087—1104.
28. Graphviz — Graph Visualization Software. http://www.graphviz.org/.
29. *Авербух В. Л., Анненкова О. Г., Бахтерев М. О., Манаков Д. В.* Задачи визуализации программного обеспечения параллельных и распределенных вычислений // Тр. Межд. науч. конф. ПаВТ'2014. Ростов-на-Дону, 2014. Челябинск: ЮФУ, 2014. С. 7—18.
30. Графическая библиотека динамической визуализации графов Ubigraph. http://ubietylab.net/ubigraph/.
31. *Kirby R., Silva C.* The need for verifiable visualization // IEEE Computer Graphics and Applications. 2008. Vol. 28, No 5. P. 78—83.
32. *Petre M.* Mental imagery and software visualization in high-performance software development teams // J. of Visual Languages and Computing. 2010. Vol. 21(3). P. 171—183.
33. *Dasgupta A., Chen M., Kosara R.* Conceptualizing visual uncertainty in parallel coordinates // Comput. Graph. Forum. 2012. Vol. 31(3). P. 1015—1024.
34. *D. F. Jerding, Stasko J. T.* The information mural: A technique for displaying and navigating large information spaces // IEEE Trans. Vis. Comput. Graph. 1998. Vol. 4(3). P. 257—271.
35. *Тюгашев А. А., Богатов А. Ю., Шулындин А. В.* Визуальный подход к верификации управляющих программ реального времени // Вестник Самарского гос. аэрокосмического ун-та. 2012. № 1(32) С. 219—225.
36. *Теплов А. М.* Об одном подходе к сравнению масштабируемости параллельных программ // Вычислительные методы и программирование. 2014. Т. 15. Вып. 4. С. 697—711.

Статья поступила в редакцию 18.11.14.

ANALYSIS AND ASSESSMENT OF VISUALIZATION SYSTEMS USED IN SOFTWARE FOR PARALLEL COMPUTATIONS / V. L. Averbukh, O. G. Annenkova, M. O. Bakhterev, D. V. Manakov (IMM of the RAS UrB, UrFU, Ekaterinburg).

Approaches to visualizing routs and graphs of parallel program calls using a number of software visualization metaphors are considered. The applicability of metaphors on the basis of visualization criteria has been analyzed. Examples of using tools for visualizing the process of developing system software of a lower level for modern parallel-architecture processors are given.

The authors state the problem of formally describing and/or verifying the visualization results.

Keywords: software visualization, execution route, call graph, visualization metaphor.