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

Аннотация
В работе 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. Также поставлена проблема формального описания и/или верификации визуализации.

1. Введение

Под визуализацией программного обеспечения *(Software Visualization)* понимается совокупность методик использования компьютерной графики и средств человеко-машинного взаимодействия, применяемых для спецификации и представления программных объектов и сущностей в процессе создания, отладки и анализа программ, а также для эффективной эксплуатации программного обеспечения. Наиболее перспективными представляются системы визуализации программного обеспечения 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 использование анимации. Изучение литературы, посвященной практике использования средств визуализации программного обеспечения параллельных вычислений, указывает на определенный застой в этой области. Попытки создать универсальные системы, предпринятые в 90-ых годах прошлого века, в целом не увенчались успехом, что объясняется целым рядом причин, в том числе и недостаточным вниманием со стороны проектировщиков к тем аспектам систем, которые можно отнести к проблемам “человеческого фактора”. В нашем 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 этой связи необходима система качественных и формализованных оценок визуализации.

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

2. 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 работы программ. Трасса программы отображает динамику конкретного выполнения программы. Представление и “проигрывание” трасс программ является важным элементов отладочных систем. Отображения графа вызовов также активно используются в системах отладки (настройки) производительности параллельных программ.

В системах отладки 70-ых и 80-ых 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, куда 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 системы, 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 отображения 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 исполнения за счет показа последовательность 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 работы 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). В последнем случае для отображения данных о структуре используются специальные круговые диаграммы, в которых все сущности проектируются 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 весьма популярной схеме анализа визуальных данных, предложенной известным ученым Б. Шнейдерманом. Так называемая мантра Шнейдермана – 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 работы 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, однако может породить сложные и плохо интерпретируемые виды отображения. Нас интересуют также возможные подходы к оцениванию метафор на этапе проектирования систем и выбора методов визуального представления сущностей параллельного программирования.

Существует примеры использования метафор для представления трасс и графов вызовов программ. В частности, при создании систем визуализации программного обеспечения параллельных вычислений используются метафоры притяжения/отталкивания (в качестве ее модификации можно рассматривать метафору молекулы и ме-

тафору физической частицы), метафоры комнаты, здания, города (ее 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 воспроизводилась работа параллельной программы (моделируется динамическая система), а результаты моделирования визуализировались (а также сонифицировались). Анимация показывала изменения в балансе между вычислениями и обменами как обычные движения некоторой динамической системы. (Таким образом, имеет место использование понятной для пользователя *метафоры притяжения/отталкивания*.) В частности определялась 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 отображаться на тела и их массу, пружины, связывающие тела, и силы притяжения между телами соответственно.

В ряде публикаций последнего времени рассматривается использование метафор города и ландшафта для представления трасс программ. Например, в [16] описывается представление трассы для случая параллелизма на основе нитей. Здания представляют статичные части программного комплекса. Трасса представляется в виде лучей-нитей, протянутых между зданиями. Визуализация такой ошибки, как deadlock естественно выглядит как окрашенное пересечение этих лучей. (Рис. 4-5.) В работе [17] используются аналогичные идеи визуального представления. Сущности программной системы (например, приложения) представляются в виде зданий, а улицы показывают потоки управления и данных, связывающие эти сущности. Возможно включение/выключение режима проигрывания трассы. Метафора города может также использоваться для представления структуры программных комплексов, например, [18]).

Отметим оригинальную метафору мозга (*“Brain” Metaphor*), использованную для анимационного представления исполнения 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 6.).

Визуализация графов вызовов должна обеспечить выявление таких сущностей, как причинные связи, порядок вызовов модулей, повторы, принадлежность данных к тому или иному типу, а также других отношений, необходимых разработчикам при отладке правильности и эффективности [20].

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

В рамках двумерного представления можно справиться с данными проблемами за счет разработки дополнительного инструментария, например за счет использования дополнительного диалога с системой при навигации по двумерному [20]. Нехватку места на экране дисплея можно преодолеть, добавив к разрабатываемой модели еще одно измерение. В работах [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, полученные на базе метафоры фабрики. (Рис 7.)

В нашей работе [23] описаны трехмерные представления графа вызовов на базе метафоры здания, когда изображаются связанных между собой комнаты некого здания сложной архитектуры. (Рис 8.) Расположение комнат - трехуровневое. Все функции программы разбиваются на три части: функции пользователя, имеющие в качестве наследников пользовательские функции; функции пользователя, не имеющие таковых и системные функции. Каждой такой части соответствует свой уровень в результирующем изображении. Каждая комната - визуальное представление функции. Кроме того, за функцию “отвечает” и 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 могло дать более оптимальный результат. Также в [23] описано использование метафоры молекулы. Метафора молекулы аналогична метафоре притяжения/отталкивания. Вершины графа естественно отображать сферами, а связи между вершинами - стрелками. Все вершины отталкиваются друг от друга, а притягиваются только в том случае, если между ними есть связь. Введенный коэффициент упругости влияет на близость вершин друг от друга, а заряд на степень удаленности остальных вершин от данной. Их, конечно, стоит учитывать в качестве статических/структурных характеристик. Например, заряд выставлять равным числу связей с данной вершиной, коэффициент упругого взаимодействия - «мощность» связи. Одной только установкой заряда в зависимости от времени выполнения определенной функции при визуализации графа вызовов существенного изменения картины мы не получим. Для визуализации количественных показателей правильнее воспользоваться традиционными методами. Время работы функции следует показывать 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. 9).

3. Анализ применимости метафор визуализации

Для анализа применимости тех или иных метафор можно использовать схему работы метафоры. То есть необходимо получить ответы на вопросы о том, как данная метафора может помочь в представлении данных и во взаимодействии 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 (точнее, видов отображения, полученных на их базе) к представлению больших и очень больших объемов данных и деталей, необходимых для понимания процесса работы программ. Важно понять, какие объекты могут быть представлены с помощью данной метафоры. Также интересны возможности применения метафор в рамках систем, использующих современные среды компьютерной графики, в частности среды виртуальной реальности. Для всего этого необходимо описать способы проверки пригодности метафор для решения поставленных конкретных задач.

Рассмотрим сходные метафоры города и ландшафта. Среди их свойств можно выделить такие, как:

- *неограниченный контекст;*

При визуализации большого количества данных, неограниченность контекста позволяет бегло окинуть взглядом всю “картинку” и быстро выделить ключевые места.

- *естественность*;

Применительно к метафорам города и ландшафта кроме естественности пространственной ориентации, имеет место также естественность навигации.

- *организация внутренней структуры*;

Метафоры предполагают наличие внутренней структуры, причем в случае метафоры города эта структура диктуется самой метафорой, и задается она достаточно жестко - есть дома, кварталы, улицы, районы. При работе с метафорой ландшафта выбор структуры - свободный. В этом случае также можно говорить о вложенности ландшафтов.

- *наличие ключевых элементов*;

Метафоры допускают представление достаточно большого объема однородной в визуальном смысле информации. Для интерпретации этой информации возможно использование ключевых элементов, ("*ключей*" или "*якорей*"), которые являются отправными точками для интерпретации всех объектов. Например, в случае применения метафоры для выделения неких особенных (например - ошибочных) элементов среди множества других, эти элементы должны выделяться цветом, размером, формой и т.п.

- *устойчивость к масштабированию*.

Метафоры устойчивы к увеличению объема информации. Более того, метафоры ландшафта и города предполагают наличие некоторого достаточно большого объема предоставляемой информации для того, чтобы их применение было оправданным.

В случае реализации в системе визуализации метафоры города и ландшафта при проектировании систем визуализации программного обеспечения *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. Возможности реализации пролета над городом/ландшафтом создает возможности осуществления легкой навигации. Полет с изменением высоты позволяет осуществить масштабирование и зуминг. Таким образом, метафоры города и ландшафта позволяют создавать системы визуализации, удовлетворяющие критериям, основанным на схеме Шнейдермана. При этом интерпретация полученных на их основе графических выводов представляется простой.

Аналогично можно рассмотреть свойства метафоры молекулы и частицы, которые также могут использоваться для задач визуализации трасс исполнения 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 (также как и в случае метафор города и ландшафта) возможно построение систем, удовлетворяющих критериям, основанным на схеме Шнейдермана. Кроме того, возможности метафор

города/ландшафта и молекулы/частицы позволяют их использовать при проектировании систем визуализации на базе сред виртуальной и расширенной реальности.

В тоже время существует ряд ограничений по применению критериев качества визуализации. Тот факт, что та или иная система им удовлетворяет, не означает, что она полностью лишена недостатков и пригодна для внедрения в практику разработки программного обеспечения. Например, система, использующая при визуализации методику “круговых узелков”, вполне соответствует “критерию Шнейдермана”, но при ее применении получаются на наш взгляд 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, хотя бы, сможет опознать это. В случае “узелков” (как и других диаграмм), при создании которых применена, по сути, абстрактная и достаточно сложная методика визуализации, интерпретировать эти графические выводы, как представляется, непросто, так как 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 работающего кода, так и из-за чрезвычайно больших объемов трасс выполнения параллельных программ. Подобные возражения 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 показывать работу 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. В этих случаях также остаются проблемы восприятия больших и очень больших объемов информации. Да и само наблюдение за мерцающими и мигающими анимационными выводами не всегда приемлемо. (А у пользователей систем на базе сред виртуальной реальности могут возникать неприятные последствия в виде головокружения и пр.) Использование трехмерных отображений для представления графа вызовов может показаться избыточным. Дело в том, что для адекватного отображения структуры графов вызовов достаточно двумерных отображений. Но при представлении сложных структур вызовов параллельных программ достигается положительный эффект трёхмерного отображения подобных графов.

Другой подход к оценке визуализации заключается в анализе наличия инсайта (insight – озарение, понимание), то есть установление 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, создавать инсайт. Можно заключить, что цель метафорических визуализаций и заключается в получении инсайта. Однако наличие или отсутствие инсайта весьма субъективно. Поэтому и оценка по этому критерию остается субъективной.

4. Визуальное обеспечение процесса разработки

Как уже указывалось, попытки создать универсальные системы в целом не увенчались успехом. Опыт разработки систем компьютерной визуализации (прежде всего,

систем научной визуализации) показывает, что успех таких систем связан с наличием формальной модели визуализируемой области, представлением о том, как пользователь видит и понимает объекты данной области знаний, а также пониманием (хотя бы в общих чертах) того, как пользователь будет пользоваться проектируемой системой. Судя по имеющейся в нашем распоряжении литературе, разработчики систем визуального представления трасс исполнения и графов потока данных делают попытки создания систем без учёта особенностей их использования. В тоже время в общем случае визуализации программного обеспечения 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 представляется, что на данном этапе успеха можно достичь лишь в случае специализированных систем визуализации программного обеспечения с четко поставленными, хотя и ограниченными задачами. Не надо увлекаться поиском эффектных метафор и разработкой универсальных средств визуализации. Следует перейти к созданию средств визуального сопровождения процессов разработки, отладки и анализа программного обеспечения, основываясь на изучении конкретной деятельности программистов, работающих в рамках конкретной программно-аппаратной среды.

Опыт в области специализированных средств визуального сопровождения процесса разработки был получен в ходе реализации системного программного обеспечения для отечественных процессоров семейства Мультиклет. Архитектура процессоров семейства Мультиклет обеспечивает простой способ внеочередного исполнения команд, явный контроль программиста за упорядочиванием памяти, прозрачную для приложений реконфигурацию во время исполнения, отказоустойчивость, в том числе и благодаря оригинальному способу кодирования машинных команд. Процессоры семейства Мультиклет не задают явной модели упорядочивания 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 отличие от традиционных ассемблеров, в данном случае, задача построения такого графа проста. Фактически нужно перевести естественным образом код программы в описание графа, например, на языке Dot (система GraphViz – [28]). Первые результаты такого перевода оказались недостаточно выразительными, так как этот перевод осуществлялся без учёта особенностей алгоритмов размещения вершин, используемых в GraphViz для визуализации. Постепенно многие из этих особенностей удалось учесть и получить приемлемые для визуализации участки кода, по которым можно было проследить интересующий программистов поток данных в участке кода. Дополнительной особенностью результатов такой визуализации оказалась возможность узнавать некоторые алгоритмы по их графическому изображению. Важным для практики результатом работы является возможность быстро прослеживать степень возможного параллелизма анализируемого участка кода, к которой процессоры семейства Мультиклет чувствительны. Код с небольшой степенью параллелизма естественным образом визуализируется как вытянутая нить операций. Такое прослеживание необходимо при отладке производительности. (Рис. 10.])

Другой пример использования визуализации связан с разработкой компилятора C99 для процессоров семейства Мультиклет. Разработка трансляторов остаётся одной из самых сложных задач в программировании. Значительная часть этой сложности связана с трудоёмкостью отладки корректности формируемых в процессе трансляции нетривиальных структур данных, насыщенных перекрёстными ссылками. Уровень насыщенности таков, что использование традиционных пошаговых локальных отладчиков не позволяет сколько-нибудь эффективно получать информацию о процессе исполнения программы, потому что возможность изучить значения нескольких пере-

менных не даёт никакого представления о состоянии программы. Более приемлемой в такой ситуации можно считать отладку по трассе программы. Однако трассы процессов трансляции даже небольших программ на C99 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 тоже 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 используются 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. 11.)

Отметим, что в рассмотренных случаях визуализация опирается на формальные описания изучаемых явлений, которые хорошо известны проектировщикам (они же и 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 проводили разработчики системного обеспечения процессоров семейства Мультиклет сами для себя, то так или иначе были учтены особенности процесса программирования и отладки. В данном конкретном случае реализации специализированных средств визуального обеспечения процесса разработки системного программного обеспечения 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 предлагаемым нами критериям оценки визуализации.

5. Верификация и валидация визуализации

Визуализация программного обеспечения является подобластью общей дисциплины – компьютерной визуализации. Под компьютерной визуализацией понимается методика перевода абстрактных представлений об объектах в геометрические образы, что дает возможность исследователю наблюдать результаты компьютерного моделирования явлений и процессов [31]. Визуализация, представляя результаты вычислений, обеспечивает интерпретацию и анализ полученных данных. В связи с развитием компьютерной визуализации, как самостоятельной дисциплины, ставится вопрос о ее теоретической базе. Теоретические исследования в области визуализации нужны, во-первых, для анализа и оценки существующих систем, во-вторых, для обучения новых специалистов и, 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, обеспечение интерак-

тивности (задача управления), 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 [32] 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, а в частном - визуализацией с неопределенностью. Он может рассматриваться, как пример некорректной по начальным данным задачи. Можно выделить качественные и количественные характеристики верификации. Качественная характеристика – это степень формализации (наивная, метафорическая, формальная) и количественная, связанная с распространением неопределенности, которые уместно именовать, как полнота и точность верификации.

Одновременно с верификацией визуализации необходимо рассматривать и валидацию визуализации [32]. Валидация – это мера адекватности. В математическом моделировании адекватность определяется как соответствие формальной модели и реализующего её программного обеспечения тому, что наблюдается на практике. В принципе, формально правильная модель может быть неадекватной. Частая путаница в определение верификации и валидации объясняется достаточно просто: наивная верификация эквивалентна валидации.

6. Заключение

Основное внимание в данной работе уделено проблеме применимости тех или иных метафор при визуализации больших и очень больших объемов сложноструктурированных данных определенного типа. В рассмотренных нами системах, как правило, используются комплексные виды отображения, в которых вместе 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, как, например, в системе SYNCTRACE [8], где метафора визуализации специально построена для отображения работы “многонитевой” параллельной программы. В других случаях имеет место адаптация и дополнение существующих метафор к задачам анализа программ без существенной привязки методик визуализации к типу параллелизма. Например, в [16] для представления работы “многонитевых” параллельных программ используется слегка адаптированная метафора города.

Отметим, что проектирование и реализация специализированных средств обеспечения процесса разработки и отладки программ нижнего уровня для процессоров семейства Мультиклет естественно проводились с полным учетом особенностей функционирования аппаратуры. Необходим также учет ментальных моделей разработчика [33] и характера его деятельности в процессе реализации ПО. Использование новых метафор и методик визуализации само по себе не всегда может дать нужный эффект. В конкретных системах оправдано использование достаточно простых, но четко интерпретируемых методов представления. (См. также [33].) Конечно, и трехмерность, и анимация очень полезны. Однако их применение не должно требовать перехода в некоторый новый и непонятный для программиста мир визуализации.

Анализируя методы представления информации в средах визуализации программного обеспечения параллельных вычислений, мы в значительной мере абстрагировались от реальных систем и тех задач, которые они призваны решать. При разработке конкретных средств отладки естественно возникают вопросы о связи методов представления данных и представления структуры программы и самого кода. Необходим

дополнительный анализ пригодности метафор визуализации для описания определенных типов распараллеливания.

Литература

1. Авербух В.Л., Байдалин А.Ю. Разработка средств визуализации программного обеспечения параллельных вычислений. Оптимизация параллельных программ // ВАНТ, сер. Математическое моделирование физических процессов, вып. 1. Стр. 70-79, 2004.
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. Стр. 179-186.
3. Посыпкин М.А., Соколов А.А. Обзор методов автоматизации мониторинга, анализа и визуализации поведения параллельных процессов, взаимодействующих с помощью передачи сообщений. Препринты ИСП РАН, Препринт 7, 2005 г.
4. Mohr B. Scalable parallel performance measurement and analysis tools - state-of-the-art and future challenges // Supercomputing frontiers and innovations. Volume 1. Number 2 (2014). Pp. 108-123.
5. De Pauw W., Heisig S. Zinsight: a visual and analytic environment for exploring large event traces // Proceedings of the 5th international symposium on Software visualization, ACM, 2010. Pp. 143-152.
6. Trumper J., Bohnet J., Dollner J. Understanding Complex Multithreaded Software Systems by Using Trace Visualization // Proceedings of the 5th International Symposium on Software Visualization, ACM. 2010. Pp.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., 2012, pp. 45–52.
8. Karran B., Trumper J., Dollner J. SYNCTRACE: Visual Thread-Interplay Analysis // Proceedings of the 1st Working Conference on Software Visualization, IEEE Computer Society, 2013. 10 pp.
9. Cornelissen B., Holten D., Zaidman A., Moonen L., van Wijk J.J., van Deursen A. Understanding execution traces using massive sequence and circular bundle views // Proc. of the 15th IEEE Int. Conf. on Program Comprehension. IEEE, 2007, pp. 49–58.
10. Cornelissen B., Zaidman A., Holten D., Moonen L., van Deursen A., van Wijk J.J. Execution Trace Analysis through Massive Sequence and Circular Bundle Views // Journal of Systems and Software. Vol. 81. 2008. Issue 12, December. Pp. 2252-2268.
11. Shneiderman B. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations // Proceedings of the IEEE Conference on Visual Languages, September 3-6, 1996. Pp. 336-343.
12. Maletic J.I., Marcus A., Collard M.L. A task oriented view of software visualization // International Workshop on Visualizing Software for Understanding and Analysis. 2002. Pp. 32–40.
13. Averbukh V.L., Bakhterev M.O., Baydalin A.Yu., Gorbashevskiy D. Yu., Ismagilov D.R., Kazantsev A.Yu., Nebogatikova P.V., Popova A.V., Vasev P.A. Searching and Analysis of Interface and Visualization Metaphors // Human-Computer Interaction, New Developments. / Edited by Kikuo Asai. Chapter 3, Vienna, In-teh. Pp. 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. *SPDP 98* Welches Or. USA. pp.72-80.

16. Waller J., Wulf Ch., Fittkau F., Dohring Ph., Hasselbring W. SynchroVis: 3D Visualization of Monitoring Traces in the City Metaphor for Analyzing Concurrency // First IEEE Working Conference on Software Visualization (VISSOFT), 2013. 4 pp.
17. Fittkau F., Waller J., Wulf Ch., Hasselbring W. Live trace visualization for comprehending large software landscapes: The ExplorViz approach // Proceedings of the 1st Working Conference on Software Visualization (VISSOFT), IEEE Computer Society, 2013. 4 pp.
18. Kobayashi K., Kamimura M., Yano K., Kato K., Matsuo A. SArF Map: Visualizing Software Architecture from Feature and Layer Viewpoints // 21st IEEE International Conference on Program Comprehension (ICPC), 2013. Pp. 43 – 52.
19. Palepu V.K., Jones J.A. Visualizing constituent behaviors within executions // Proceedings of the 1st Working Conference on Software Visualization (VISSOFT), IEEE Computer Society, 2013. 4 pp.
20. LaToza T., Myers B. Visualizing call graphs // 2011 IEEE Symp. on Visual Languages and Human-Centric Computing, 2011, pp. 117–124.
21. Bohnet J., Dollner J. Visual exploration of function call graphs for feature location in complex software systems // Proc. of Symp. Soft. Vis., 2006, pp. 95–104.
22. Bohnet J., Dollner J. Facilitating Exploration of Unfamiliar Source Code by Providing 21/2D Visualizations of Dynamic Call Graphs // Proceeding of 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. Pp. 63-66.
23. Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р., Казанцев А.Ю. Трёхмерные методики визуализации программного обеспечения параллельных и распределённых вычислений // Труды ПаВТ'2008. Челябинск, ЮУрГУ, 2008. Стр. 283-288.
24. Авербух В. Семиотический подход к формированию теории компьютерной визуализации // Научная визуализация, 2013. Квартал: 1. Том: 5. Номер: 1. Стр. 1-25.
25. Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р., Казанцев А.Ю., Тимошпольский С.П., Использование трехмерных метафор визуализации // 14-я Международная Конференция по Компьютерной Графике и Зрению ГрафиКон'2004 6-10 Сентября 2004 Москва, Россия. Труды Конференции. МГУ им. М.В. Ломоносова. Стр.295-298.
26. North Ch. Toward Measuring Visualization Insight // IEEE Computer Graphics and Applications May/June 2006, Volume: 26, Issue: 3, pp. 20-23.
27. Стрельцов Н.В. Архитектура и реализация мультиклеточных процессоров // Труды V Международной научной конференции «Параллельные вычисления и задачи управления» - Москва, 26-28 октября 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 // Труд Международной научной конференции ПАВТ’2014. ЮФУ. Стр. 7–18.
30. http://ubietylab.net/ubigraph/
31. Visualization in Scientific Computing, Special Issue, ACM SIGRAPH Computer Graphics, V. 21, N 6, November 1987.
32. Kirby R., Silva C. The need for verifiable visualization // IEEE Computer Graphics and Applications. Sep 2008. Vol.28. No5. Pp.78 –83.
33. Petre M. Mental imagery and software visualization in high-performance software development teams // Journal of Visual Languages and Computing, 21 (3), 2010. Pp. 171–183.

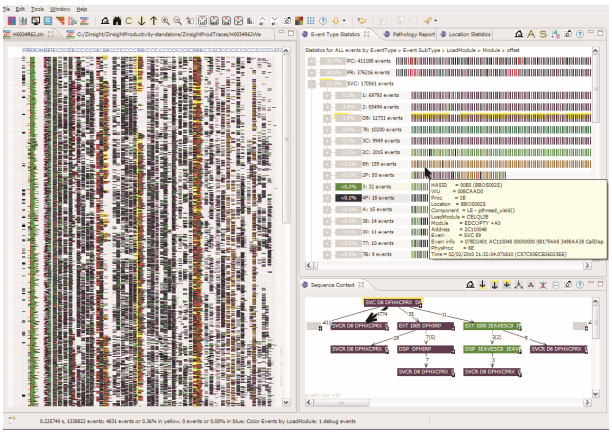

Рисунок 1: Визуализация трассы в системе Zinsight [5]

Рисунок 2: Основное окно системы SYNCTRACE [8]

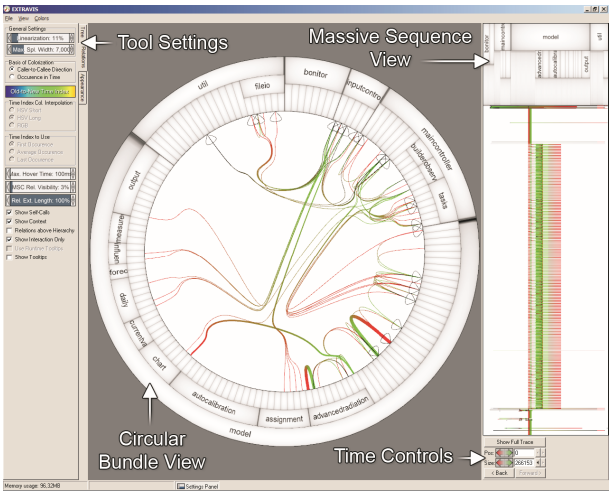

Рисунок 3: Визуализация трассы в виде узелков [9-10]

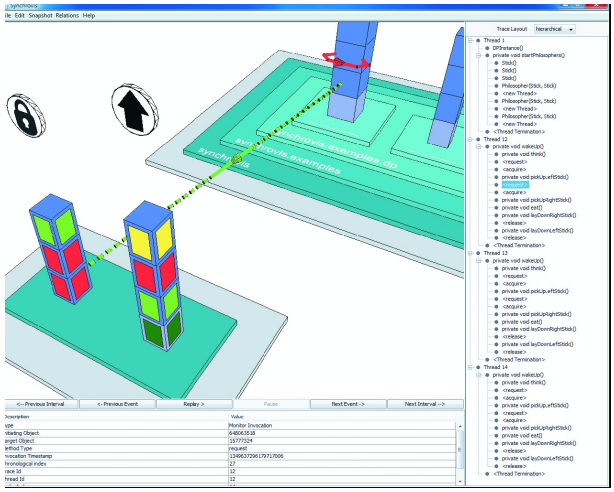

Рисунок 4: Визуализация нормальной работы программы [16]

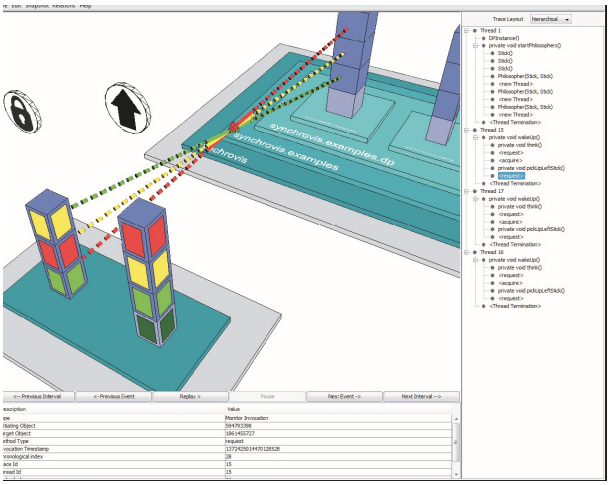

Рисунок 5: Визуализация ошибочной (deadlock) работы программы [16]

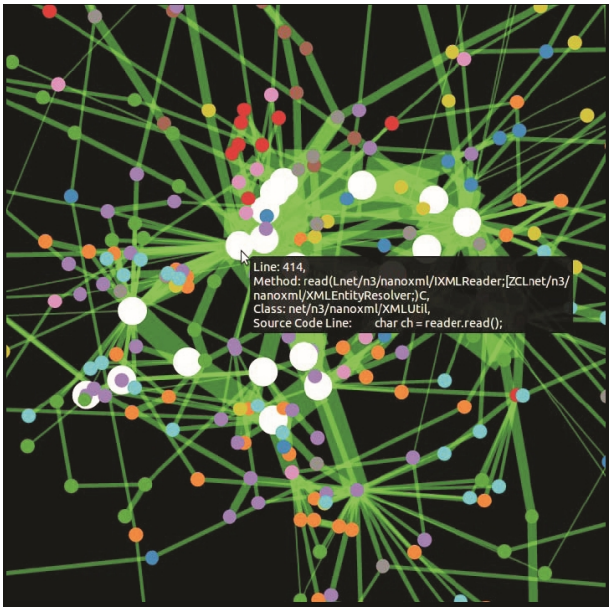

Рисунок 6: Использование метафоры мозга для представления работы программы (скриншот анимации) [19]

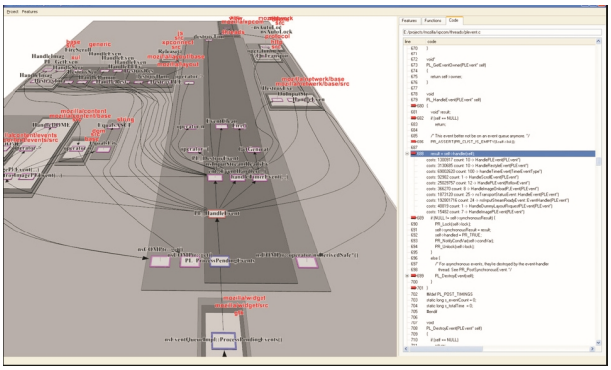

Рисунок 7: Граф вызовов и текстовое описание кода [22]

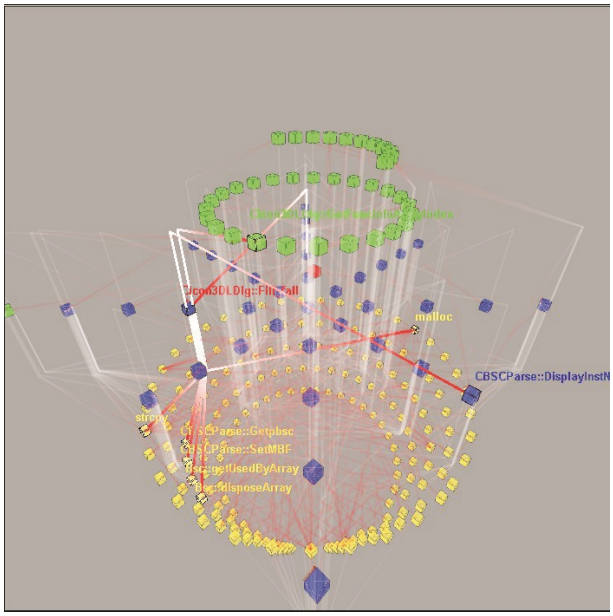

Рисунок 8: Граф вызовов системы рисования графа вызовов на базе метафоры здания [23]

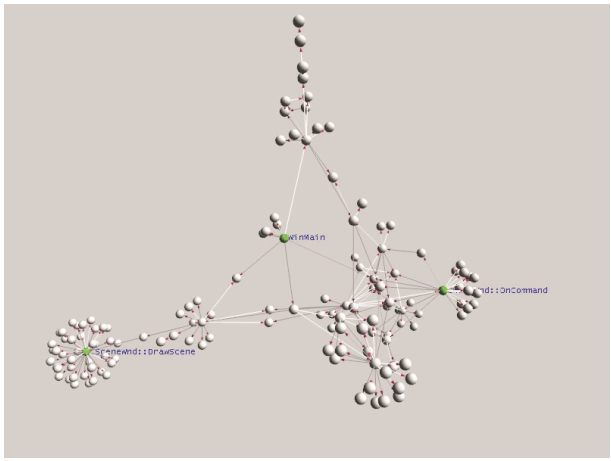

Рисунок 9: Граф вызовов системы рисования графа вызовов на безе метафоры молекулы [23]

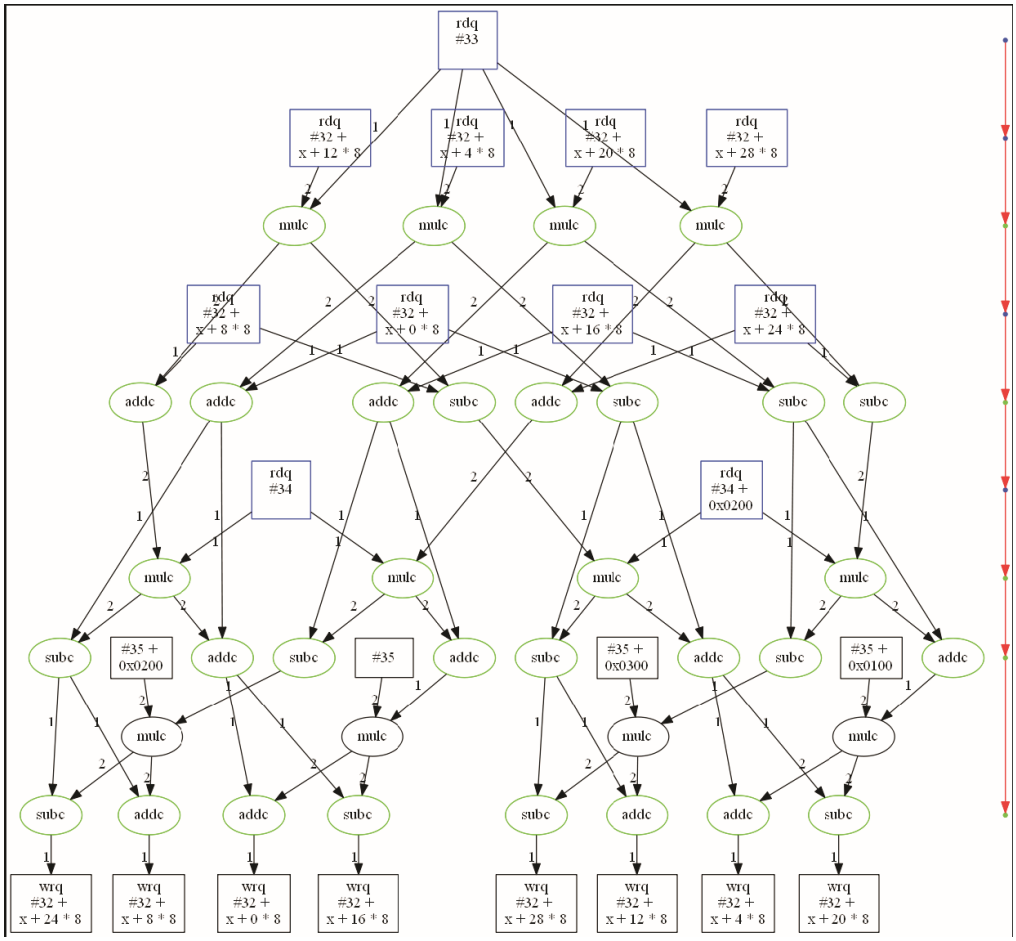

Рисунок 10: визуализация графа участка программы на ассемблере процессора семейства Мультиклет: улучшение структуры [29].

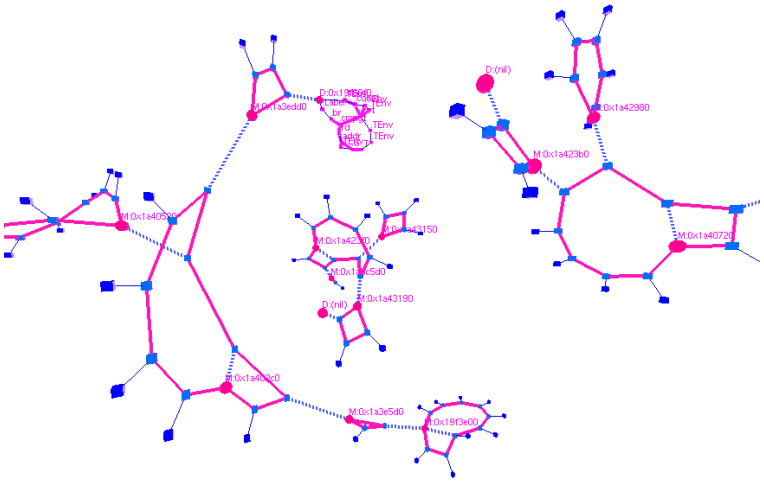

Рисунок 11: Визуализация структуры графа трансляции.