АНАЛИЗ ПАРАЛЛЕЛЬНЫХ ВИЗУАЛЬНЫХ ТЕХНОЛОГИЙ*

Д. В. Манаков
Институт математики и механики УрО РАН, Екатеринбург, Россия
e-mail: manakov@imm.uran.ru

A systematic classification of the visual supercomputing is presented. Possible methods and approaches aimed at getting maximal effectiveness during visualization of scientific parallel computing results are considered. These solutions are included into the system of interactive visualization of parallel computations which is currently being developed. Comparative analysis of the similar projects is given using this classification.

Введение

В настоящие время отмечается возрастание роли визуализации в параллельных вычислениях, особенно для коммерческих приложений, связанных, например, с обработкой видеоизображений. В то же время количество научных публикаций по этой теме сравнительно невелико, и ряд терминов, используемых в литературе, еще не устоялся. В связи с этим важно точно определить предметную область и дать классификацию используемых в ней подходов. Понятие "визуальные супервычисления" вводится в работе [1], где указывается, что они касаются инфраструктурных технологий для поддержки визуальных и интерактивных вычислений вообще и визуализации в частности в сложных сетевых вычислительных средах. Возможно, этот термин и его определение не совсем удачны, но отражают суть. Итак, проведенные исследования связаны с параллельной графикой и удаленной визуализацией и нацелены на оценку визуальных технологий, обеспечивающих параллельные вычисления.

Фактически возникает новая предметная область на стыке других дисциплин, которая имеет отношение к довольно большому кругу вопросов. Визуальные и интерактивные вычисления востребованы в таких областях, как, например, автоматизированное проектирование, компьютерная анимация и компьютерное зрение. Поэтому в работе введены следующие естественные ограничения. Прежде всего, рассмотрение предложенных подходов в рамках научной визуализации. Во-вторых, параллельные, а не распределенные вычисления. И, наконец, возможно самое главное, обязательное применение интерактивности, в том числе для визуального анализа результатов во время выполнения программы.

Так же, как и инфраструктурные технологии, визуальные супервычисления предполагают большой набор аппаратных технологий и программных систем для поддержки вычислений и управления задачами визуализации. Они обращаются к таким проблемам,

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 04-07-90120).

как планирование задач визуализации, аппаратное и программное конфигурирование, параллельные и распределенные вычисления, распределение данных, организация обменов между визуальной и счетной задачами. Отметим, что, по мнению авторов [1], проблематика визуальных супервычислений не касается конкретных алгоритмов и технологий обработки частных типов данных с целью получения результатов визуализации. Это не совсем правильно, так как, например, важная для визуальных супервычислений задача параллельной фильтрации тесно связана с алгоритмами, реализующими параллельные фильтры.

В настоящей работе мы рассмотрим принципы систематизации визуальных супервычислений, схему классификации, а также проблемы, связанные с реализацией систем, относящихся к тому или иному классу.

1. Принципы систематизации и структура визуальных супервычислений

Обзор, классификация и полное понимание некоторых аспектов визуальных супервычислений помогут получить основные принципы проектирования суперкомпьютерных систем, реализации многих способов и методов решения, визуального анализа результатов научных вычислений. Нет необходимости давать полный обзор систем, реализованных в этой области, важны только те, которые содержат интересные и перспективные идеи в рамках

Рис. 1. Схема классификации визуальных супервычислений.

классификации. Осмысление — основное отличие классификации от обзора, поэтому необходим анализ эффективности рассматриваемых подходов. Классификация должна быть структурирована. Следовательно, должны существовать принципы систематизирования, и их не должно быть много. Предлагаемая структура классификации визуальных супервычислений основана на двух принципах систематизирования:

— решение возникших проблем (возникновение проблемы требует ее решения, выбор оптимального решения ведет к созданию технологии);

— выделение характерных особенностей основных программных модулей.

Схематично классификация представлена на рис. 1, где используется сокращение PDF — параллельная фильтрация данных.

2. Классификация по решаемым проблемам

Если рассмотреть абстрактную систему визуализации в области параллельных или распределенных вычислений, естественно, первым вопросом будет, какова цель этой системы или для чего она предназначена, т. е. какие задачи можно решить с ее помощью. Следовательно, первый принцип систематизирования визуальных супервычислений основан на рассмотрении *проблем*, которые возникли или видоизменились с появлением параллельной техники в области научной визуализации.

Организация и управление интерактивностью. В настоящее время практически любая система визуализации включает интерактивность. Для параллельных и распределенных вычислений проблема заключается в том, что первоначально визуализируемые данные удалены от графической станции, т. е. применения стандартных методов взаимодействия недостаточно для получения полноценной системы визуализации.

Традиционно визуализация применяется лишь на последнем этапе проведения вычислительного эксперимента — как вспомогательный инструмент, позволяющий ускорить процесс понимания пользователем результатов вычислительного эксперимента. Такой подход к анализу называется off-line визуализация. По завершении работы программы на параллельном вычислителе происходит запись результатов вычислений в некоторый буфер (например, файл), откуда программа, которая должна отобразить результаты, забирает данные.

Off-line визуализация — один из шагов в цепочке пуска параллельной программы и последующего анализа результатов. Данные шаги выполняются последовательно, что накладывает некоторые ограничения на подобный подход к анализу и представлению результатов. Поэтому off-line визуализация вычислений часто бывает неудобной.

Для современной визуализации, как это ни странно, нет проблем в представлении, моделировании и отображении достаточно сложных объектов. Существует достаточно много различных видов отображения для различных исследуемых объектов, которые успешно применяются и удовлетворяют большинство потребностей в визуализации. Проблемы кроются в самом подходе к организации традиционной схемы моделирования объекта, а именно в последовательности шагов, применяемой обычно для проведения вычислительного алгоритма. Рассмотрим эти проблемы:

— большой объем данных, требующих визуализации. С ростом вычислительной мощности стали расти и объемы результатов вычислений. Для решения этой проблемы требуется использовать как минимум интерактивность, а также желательно более эффективные механизмы;

— необходимость получать информацию о текущем состоянии вычислений и влиять на работу параллельной программы. Данная задача появляется в результате того, что решение достаточно большой задачи влечет за собой продолжительный (многосуточный) счет, только после которого можно оценить результаты. Если же при запуске параллельной программы нельзя сразу определить правильные начальные данные, или по каким-то причинам они были заданы неверно, или была допущена ошибка в алгоритме работы программы, то потребуется повторный запуск программы, что влечет за собой большую потерю времени. Поэтому требуется возможность получать предварительные результаты вычислений, визуализировать их и корректировать работу на вычислителе, что даст возможность оценить правильность работы программы и скорректировать ее по ходу вычислений или в худшем случае прервать счет для исправления алгоритма программы. Стоит подчеркнуть диалектическое единство и противоположность между анализом промежуточных данных и результатов вычислений.

Перечисленные проблемы делают процесс отладки параллельной программы очень трудоемким. Пользователь часто тратит много времени фактически "вхолостую", загружая параллельный компьютер, так как убедиться в неправильной работе своей программы он может только после прохода полного цикла от пуска программы до получения ее результатов и последующего их анализа.

Общепринято называть визуализацию результатов вычислений в ходе выполнения программы on-line визуализацией. Важно отметить, что организации просто on-line визуализации недостаточно, необходима полноценная интерактивность, т. е. реализация обмена сообщениями в обе стороны как по вводу данных, так и по получению результатов с предоставлением удобного интерфейса.

Использование прилагательного "интерактивная" подчеркивает то, что к такой визуализации данных выполняемой программы добавляется возможность влиять на ее работу. Необходимость такого решения может появиться при решении многих задач, например интерполяционных. Часто для полного анализа вычислительного эксперимента и определения правильности поведения программы необходимо визуализировать не только конечные результаты программы, но и некоторые ее текущие данные.

Известно, что в ряде случаев возникает некоторое количество промежуточных данных, которые необходимы при вычислениях, однако несущественны при конечном анализе результатов вычислений. При включении в процесс счета интерактивности пользователь сам может определить, какие данные ему необходимы, а также способ представления этих данных, другими словами, определение того, как эти данные будут визуализированы: в виде текста, в том или ином графическом виде либо, если данные не нужны, исключить их из общего объема визуализируемых данных.

Понятно, что для реальных задач, когда время счета измеряется часами и сутками, а объемы данных, которыми оперирует программа, достигают терабайтных размеров, ни о какой интерактивности в наивном понимании этого слова (т. е. общение с пользователем в диалоговом режиме) не может быть и речи. Однако по вышеизложенным причинам необходимость привнесения элементов интерактивности также представляется очевидной. Возможным решением этой проблемы является реализация сценария работы программы, например, если долго нет отклика от пользователя, интерактивный режим заканчивается и подготовленные данные не визуализируются, а записываются в файл.

Итак, для решения всех этих проблем в процесс счета необходимо добавить возможность пользователю интерактивно изменять параметры счета и выбирать некоторый набор текущих данных, получать их, задавать способ их визуализации и отображать их

на рабочей станции. Такое решение приведет к упрощению процесса отладки программы, поскольку даст возможность получать сведения о состоянии программы не только по окончании вычислений, но и в процессе работы и тем самым позволит программисту осуществлять бóльший контроль над работающей программой.

Вторым несомненным плюсом такого подхода является возможность начать анализ результатов вычислительного эксперимента до окончания полного счета лишь по части просчитанных данных. Это позволяет непосредственно изменить параметры вычислений или останавливать программу, если пользователь увидит, что результат не соответствует ожидаемому.

Вообще on-line визуализация не имеет сильных антагонистических противоречий с off-line визуализацией, преимущества и недостатки каждого из подходов достаточно очевидны. В любом случае выбор должен оставаться за конечным пользователем, т. е. система визуализации должна содержать всю цепочку от off-line визуализации до полноценной интерактивности.

К рассматриваемой проблеме можно отнести и организацию совместной работы, которая позволяет многим пользователям взаимодействовать друг с другом и с объектами в распределенной виртуальной среде. Можно выделить три подхода [1]:

— разделение по дисплеям, когда выполняется единственное приложение, а интерфейс доступен для нескольких пользователей;

— разделение по данным, когда данные распределяются группе пользователей в зависимости от их желания;

— полное сотрудничество, в котором участники способны программировать способ совместной работы.

Для организации совместной работы возможно применение широковещания, что повлияет на механизм взаимодействия, интерактивность.

Визуализация данных большого объема. Данные подобного типа получаются, в частности, при решении задач гидрогазодинамики. В результате вычислений на параллельной технике возникают данные, объем которых измеряется терабайтами. Такие данные невозможно не только визуализировать, но и передать на рабочую станцию в полном объеме или за приемлемое время. Таким образом, возникает задача по сокращению передаваемой информации, для этого разрабатываются различные алгоритмы по сжатию полученных данных или их фильтрации непосредственно на параллельном вычислителе.

Вообще говоря, решение этой задачи требует серьезной инфраструктурной поддержки [1], к которой относятся:

— обеспечение достаточного объема памяти;

— управление механизмом распределения данных;

— выбор самого эффективного алгоритма;

— предоставление удобного и эффективного графического интерфейса.

Все это должно обеспечить получение требуемого решения за приемлемое время. Однако основной задачей при визуализации данных большого объема является сокращение передаваемых данных для последующей визуализации. Подходы к решению этой проблемы можно разбить на два основных класса — это применение интерактивности, а также сжатие и фильтрация данных. В иностранной литературе по решению этой задачи существует большое количество публикаций, но в основном рассматриваются вопросы, связанные с генерацией растрового изображения [2]. Этот подход в ряде случаев не только неэффективен, но и неприемлем, целесообразней передавать специальным образом отобранные

математические данные, а затем их визуализировать, т. е. применять параллельную фильтрацию данных.

Визуализация параллельного программного обеспечения. Прежде всего это визуальные среды разработки параллельных программ, визуальные параллельные языки, а также анализ эффективности и отладка правильности параллельных программ.

Наиболее перспективным кажется создание визуальных сред разработки параллельных программ. Другое, более функциональное название — интегрированные системы редактирования, компиляции и запуска параллельных интерактивных задач, включающие функции терминала. Стоит отметить "эмулятор общей памяти для интерактивных графических задач" (разработка ИММ РАН, Москва) [3, 4], визуальную среду разработки DVM программ [5], среду программирования Аванго [6], адаптивный конструктор для интерактивных задач [7]. Есть мнение, что параллельные вычисления не имеют достаточно широкого (массового) распространения из-за отсутствия "привычной" визуальной среды разработки. Конечно, степень развития вышеприведенных систем различна, но все же можно сформулировать основные направления — это использование трехзвенной модели клиент-сервер и применение проблемно-ориентированного протокола. Так, например, в среде Аванго передается вектор состояния объекта.

Что касается отладки, необходимо использовать трехмерные метафоры, которые значительно увеличат масштабируемость. Заслуживает внимания метафора молекулы [8], применяемая для визуализации графа вызова. Необходимо представлять в визуальном виде структуру программы, распределение данных и объем работы.

Визуализация параллельного программного обеспечения заслуживает более детального и отдельного изучения [9, 10].

Параллельные графические алгоритмы и обработка растрового изображения. Несомненно, для научной визуализации значительный интерес представляют внеядерные ("out-of-core") подходы или алгоритмы с внешней памятью, минимизирующие накладные расходы дискового ввода-вывода. Двумя наиболее часто используемыми подходами являются организация данных со многими разрешениями или применение мультиразрешения (используется для уровня детализации) и организация данных, зависящая от точки зрения. Для обоих случаев характерно применение k-дерева при реструктуризации данных. Также стандартным подходом является составление сложного объекта из простых и введение в сцену нескольких объектов.

Рассмотрим более детально внеядерный алгоритм [11]. Он предназначен для интерактивного построения линий тока на большой неструктурированной тетраэдральной сетке. Применяется восьмеричное дерево для разбиения на части и реструктуризации необработанных данных на подмножества, сохраняемые в файлах, что дает возможность быстрого извлечения данных. Результирующий алгоритм разбивается на две стадии: предобработка и построение линий тока. Целью первого шага является реструктуризация данных и сохранение всей информации на диске в более компактном представлении, использующие восьмеричное дерево. Он выполняется только один раз. На втором шаге сначала выбирается начальная точка, определяется, к какому октанту она принадлежит, строится линия тока в октанте, определяется соседний и продолжается построение. Плюсом является то, что читается не весь первоначальный большой файл, а только файл, соответствующий нужному октанту. Он вполне умещается в оперативную память (чем больше уровень разбиения, тем меньше файл), а следовательно, не возникает трешинг.

Авторы рассматривают этот подход как альтернативный параллельному, но основное время занимает предобработка, которую вполне можно было сделать параллельно. Еще

раз следует подчеркнуть, что между on-line и off-line визуализацией не существует антагонистических противоречий и взаимное применение этих подходов необходимо настраивать под решаемую задачу.

Было бы интересно рассмотреть применение восьмеричного дерева для детализации воксельного представления. Достаточно много публикаций, в частности в рамках супервычислений, посвящено воксельной графике. Это новое направление, получившее бурное развитие с появлением мощных видеокарт.

Следует отметить, что применяемые алгоритмы в результате ориентированы на получение конкретного вида изображения, следовательно, целесообразно использование проблемно-ориентированного или специализированного подхода.

Наиболее подробное описание применения технологии параллельной фильтрации данных можно найти в статье [12]. Преимущества применения предобработки на счетных узлах вычислителя сейчас для многих очевидны [13]: обрабатываемые для визуализации данные уже находятся на узлах вычислителя, что позволяет избежать дополнительных обменов; при визуализации можно учесть характерные особенности задачи; возможна on-line визуализация большого объема данных; высокая скорость визуализации. Кроме трудностей при реализации этого подхода основным недостатком является приостановка счетной задачи во время предобработки.

Также существуют работы по распараллеливанию ray tracing-a, ray casting-a, сжатию и обработке видеоинформации, нацеленные на получение фотореалистического изображения. Поэтому их рассмотрение более интересно в другой сфере, например для анимации персонажей.

3. Классификация по программным модулям

Другой принцип систематизирования визуальных супервычислений базируется на рассмотрении характерных особенностей трех основных программных модулей: параллельного, транспортного и визуального. Он обеспечивает взгляд на систему визуализации со стороны разработчика, а не со стороны пользователя, на решение проблем, а не на их постановку. Несмотря на то что эти модули тесно взаимосвязаны, их можно считать достаточно автономными. Они отвечают за эффективное распараллеливание, эффективную передачу данных, эффективную визуализацию. Но если мы рассматриваем распараллеливание графического алгоритма, то можно говорить только об эффективности визуализации, если — отладку, то — об эффективности визуальной отладки.

Естественно начать рассмотрение с *параллельной части*. Известно, что классификация параллельного рендеринга является общепринятой [14]. Параллелизм различных типов может применяться на различных уровнях, например, функциональный или конвейерный может ускорить важные вычисления, а параллелизм по данным может использоваться для вычисления множественных результатов сразу. Параллелизм по данным, в свою очередь, может быть разделен на объектный и пиксельный или имидж-параллелизм. Стандартный графический конвейер отображения так же, как и приспособленный для параллельного случая, включает две принципиально важные части: геометрическую обработку данных и растеризацию. Если обе части выполняются параллельно, то такие системы принято называть полностью параллельными. Геометрическая обработка обычно распараллеливается назначением каждому процессору подмножества примитивов (объектов) в сцене, а растеризация — порции обрабатываемых пикселей. Суть задачи графического отображения есть вычисление воздействия каждого примитива на каждый пиксель. В результате

произвольной природы модели пиксели могут попасть на экран или не попасть. Следовательно, рендеринг может рассматриваться как проблема сортировки примитивов.

Таким образом, авторы предлагают классификацию по тому, в каком месте параллельного графического конвейера происходят сортировки. Во время геометрической обработки — "sort-first" сортировки выполняются вначале, в естественной точке разрыва конвейера — "sort-middle" в середине, во время растеризации — "sort-last" в конце. Выбор каждого из этих классов приводит к разным параллельным алгоритмам рендеринга с различными свойствами. Сортировки, выполняемые вначале, требуют распределения (перераспределения) "raw"— необработанных примитивов (предпочтительнее называть их с учетом предобработки математическими данными) до того, как их экранные параметры известны, выполняемые в середине — требуют экранных примитивов, выполняемые в конце — пикселей или фрагментов пикселей. В свою очередь "sort-last" алгоритмы авторы разделяют на разреженное и полнокадровое слияние, т. е. в первом случае каждый процессор обрабатывает свою часть изображения, и затем происходит объединение результирующего имиджа, в последнем каждый кадр целиком обрабатывается на своем процессоре. Также авторы дают оценку вычислительных и коммуникационных затрат, предлагают учитывать межкадровую взаимосвязь и делают анализ эффективности для баланса загрузки.

Основные комментарии к предложенной классификации в основном касаются вопросов эффективности. Полностью параллельный рендеринг может быть эффективен только для обработки растрового изображения. Чтобы воспользоваться аппаратными возможностями видеокарт, растеризацию необходимо исключить из параллельного конвейера и выполнять ее на рабочей станции. Также необходимо использовать предобработку или параллельную фильтрацию как математических данных, так и графических примитивов, что позволит не только сократить объем передаваемой информации, но и распараллелить графические алгоритмы.

Чтобы правильно классифицировать *транспортную подсистему*, необходимо ответить на два основных вопроса: как и какие данные передаются?

Первый вопрос влечет за собой два других: какой протокол нижнего уровня используется (например, TCP/IP) и какая применяется клиент-серверная модель?

Наиболее просто передачу информации между параллельно работающими процессами и графической программой можно реализовать через файлы с использованием файлов флагов. Несмотря на значительную латентность при передаче информации, такой подход имеет свои преимущества, в первую очередь для отладки.

Предпочтительнее использовать двухзвенную модель клиент-сервер (клиент находится на управляющей машине параллельного вычислителя) с применением X-терминала. Наиболее узким местом данного взаимодействия является организация обмена сообщениями между управляющей машиной и вычислителем (при отсутствии или неприменимости соответствующих системных средств). Для организации обменов можно использовать именованные каналы [15] или вставлять обмены в загрузочный модуль либо в операционную систему параллельной машины. Применение X-терминала также лишает возможности использования аппаратных средств видеокарт.

Исследования показали преимущества трехзвенной клиент-серверной модели, предполагающей размещение программ комплекса на рабочей станции, управляющей ЭВМ параллельного вычислителя и его процессорах. Возможны вырожденные случаи, когда хост-машина является одним из процессоров вычислителя или на ней происходит визуализация. Введение дополнительного узла (прокси-сервера) кажется нецелесообразным, так как приводит к дополнительным временным затратам и усложняет организацию обменов. Ко-

гда все звенья работают под Windows, достаточно двухзвенной модели (например, для распределенных систем), рассматриваемая задача упрощается, так как существует ряд стандартов, в частности CORBA- и COM-технологии.

Окончательное формирование изображения должно осуществляться на мощной рабочей станции. Как правило, математические данные занимают меньший объем, чем данные результирующего трехмерного изображения. Однако часть счета может выполняться и на рабочей станции. В то же время очевидно, что такие операции, как вращение, перемещение и сжатие графического 3D-объекта, надо выполнять на рабочей станции. Реализацию обменов между графической станцией и управляющей машиной довольно часто проводят с использованием Java. По-видимому, список протоколов нижнего уровня и механизмов для организации обменов не ограничен.

Рассмотрим второй вопрос — какие данные передаются? Кроме передачи растрового изображения возможно применение аппаратно-независимого метафайла, этот подход можно разделить на передачу графических примитивов или внутренних данных графических библиотек (ABS.VTK) и передачу команд, проблемно-ориентированного (содержит математические данные) и аппаратно-зависимого (в кодах устройства, видеокарты) метафайлов [16]. В настоящее время последний подход не применяется, хотя и возможен.

Очевидно, что в условиях работы на параллельных вычислителях с ограниченными возможностями передачи информации по медленным сетям формирование растрового изображения на параллельных процессорах неэффективно. Довольно часто для передачи изображения в такой ситуации используются аппаратно-независимые графические протоколы или метафайлы. При этом на параллельных процессорах работает некоторая стандартная графическая система или графическая библиотека.

Правильно выбрав графическую библиотеку, мы обеспечиваем решение широкого класса задач. Плюсом является и то, что текст графической задачи, размещенной на одном из процессоров параллельного вычислителя, практически идентичен тексту последовательного варианта.

При разработке библиотеки, создающей метафайл, как правило, используются именованные функции. При этом передается номер функции, потом ее фактические аргументы, а возвращается код ответа. В многопроцессорном варианте используется идентификатор (номер) процесса для согласования момента рисования и определения соответствия данных. Кроме того, в объектно-ориентированных библиотеках используется идентификатор объекта. Кроме команд инициализации и деинициализации графики, которые можно скрыть, используются команды ожидания, принудительного выталкивания буфера и стандартные конструкции синхронизации (барьеры, семафоры, удаленный вызов процедур), которые тоже можно вставить в вызов некоторых графических функций. Также стандартно используются сохранение состояния и два режима: сохранный и немедленного исполнения [17]. Для графических функций код ответа часто не важен, поэтому используется буферизация, когда в одной посылке сообщения передается максимальное число функций. Недостатком этого подхода является большое количество графических функций, реализация которых однообразна, но трудоемка.

Наиболее часто встречается и эффективно используется передача отфильтрованных математических данных путем создания графического сервера приложений, например параллельная библиотека MATLAB [18]. По существу, это тот же метафайл с аналогичной структурой, только проблемно-ориентированный. Недостаток и одновременно плюс этого подхода — то, что он ориентирован на конкретные задачи, т. е. является не универсальным, а специализированным.

Для *подсистемы визуализации*, кроме того, важно, какая графическая библиотека используется на клиенте, а также является ли эта подсистема универсальной, унифицированной или специализированной. Это разделение стандартное для любой системы визуализации. Все же отметим характерные особенности параллельной реализации. Для универсальной системы необходим полностью декодирующий конвейер, а следовательно, не требуется редактирование клиентской части, текст параллельной части, касающийся вызова графических функций, практически идентичен последовательному. Для унифицированной достаточно вставки функционально-фиксированной последовательности обращения к вычислителю. Специализированное приложение разрабатывается для конечного пользователя с учетом особенностей, налагаемых параллельными или распределенными вычислениями, и также должно базироваться на предоставление соответствующих технологий.

Выбор той или иной методики, описанной в классификации, в первую очередь зависит от требований, возникающих при решении задачи. В ряде случаев вполне можно обойтись генерацией растрового изображения и off-line визуализацией, но наибольшей эффективности можно достичь, применяя проблемно-ориентированный протокол, параллельную фильтрацию данных и интерактивность.

4. Эффективность разработки систем визуализации

Для сложного инфраструктурного программирования на первый план выходит проблема эффективности не только решаемой задачи, но и разработки самой системы визуального параллельного программирования. Необходимо использовать передовые технологии программирования как для ускорения разработки создаваемой системы, так и для повышения качества сервисов, предоставляемых пользователю. Одним из возможных решений этих задач является применение технологии агентов, к которой мы прибегали для автоматического переноса объектно-ориентированной библиотеки VTK на параллельную технику [19]. Другая технология, часто применяемая для автоматизации параллельного программирования, — это предпроцессорная обработка (например, для DVM или OpenMP директив). Более подробно рассмотрим автономные вычисления.

Считается, что решением возрастающей административной сложности вычислительных инфраструктур являются автономные вычисления. Следовательно, ни одна инфраструктура визуальных супервычислений не может игнорировать эту проявляющуюся технологию. Автономные вычисления обращаются к вычислительным системам, которые обладают способностью самопознания и самоуправления. Такая система может характеризоваться одним или несколькими из атрибутов, таких как [1]:

— самоконфигурирование — система может объединять новые и существующие компоненты без вмешательства администратора;

— самооптимизация — система может непрерывно пытаться изменить конфигурацию, чтобы определить, является ли текущая оптимальной;

— самовосстановление (самооздоровление) — система может отслеживать ошибки и восстанавливаться после неправильных аппаратных или программных компонентов;

— самозащита — система может отслеживать попытки взлома и реагировать соответствующим образом.

Адаптировав модель развертывания, предложенную IBM [20] для постепенного развития самоуправляемых сред, можно выделить пять уровней для модели развертывания визуальных супервычислений (рис. 2).

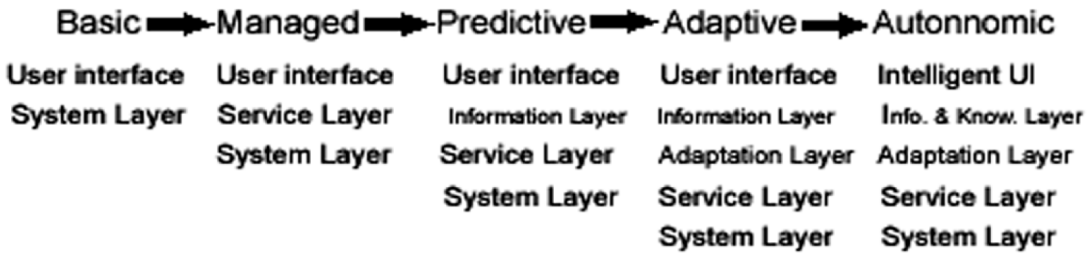

Рис. 2. Развитие новых визуальных приложений.

Уровень 1 (основной). На этом уровне инфраструктура визуальных супервычислений является интегрированной системной платформой, которая обеспечивает визуальные приложения необходимыми вычислениями и вычислительными ресурсами. Обычно пользователь полностью привлекается к определению подходящих инструментов, установлению положения вычислительных ресурсов и управлению распределением данных. Часто пользователю необходимо самостоятельно применять управление через сложные технические препятствия, такие как сеть, безопасность, параллельные вычисления, копирование данных и т. д.

Уровень 2 (управляемый). На этом уровне инфраструктура имеет прослойку управляемых сервисов между интерфейсом пользователя и системной платформой. Сервисная прослойка знает о наличии и сущности (онтологии) данных и ресурсов и может обеспечить обслуживанием различные визуальные приложения в соответствии с динамическими потребностями пользователей и приложений, а также снабдить их динамическим состоянием системной платформы. Чтобы управлять визуальными приложениями эффективно, необходимо подключить наиболее передовые сервисные функции для поддержки многообразия визуальных потребностей, таких как интерактивные, распределенные, мобильные и потребности для проблемно-ориентированных приложений.

Уровень 3 (предсказывающий). На этом уровне инфраструктура имеет информационную прослойку между интерфейсом пользователя и сервисной прослойкой, которая собирает, отслеживает и исправляет различные данные взаимодействия пользователя и данные системного выполнения. Она обеспечивает пользователей аналитическими данными, которые могут показать качество результатов визуализации и эффективность визуальных инструментов. Кроме того, информационная прослойка может позволить более быструю и лучшую спецификацию задачи, показывая возможные проблемы и рекомендуя подходящие инструменты и визуальные представления.

Уровень 4 (адаптивный). На этом уровне инфраструктура имеет адаптивную прослойку между информационной и сервисной прослойками. Основываясь на собранной информации, она имеет функциональность для самоконфигурирования, самооптимизации и самоуправления.

Уровень 5 (автономный). На этом уровне обычный интерфейс пользователя заменяется интеллектуальным. Например, “визуальный секретарь” допускает преобразование информации в знание и обеспечивает пользователя широкой помощью. Он может включать спецификацию задач визуализации, планирование взаимозависимых работ, организацию необработанных данных и результатов визуализации, управление безопасностью, проверку качества обслуживания и результатов, организацию разделения данных с другими пользователями.

Идея автономных вычислений привлекательна, но требует более детального анализа. Так, для самовосстановления возможно применение контрольных точек, резервного копирования, а интерактивность может снизить степень безопасности.

5. Сравнительный анализ систем визуализации

Результатом классификации и исследований, проведенных в области визуальных супервычислений, можно считать разработку проекта системы интерактивной визуализации параллельных вычислений [21], которая предполагает функционирование по следующей схеме:

— задача считается на параллельном вычислителе и оставляет (окончательный или промежуточный) результат в файлах или в памяти процессоров вычислителя;

— на параллельном вычислителе проводятся некоторая предобработка и параллельная фильтрация данных, связанная с будущей визуализацией;

— по заданию пользователя строятся модули описания параллельных фильтров и видов отображения, на вход которым подаются данные после предобработки (на первых этапах предлагается использовать предопределенные в системе виды отображения);

— сама визуализация может идти как на графической рабочей станции, так и на параллельной машине (в последнем случае на ПЭВМ может подаваться собственно растр).

Предполагается рассмотреть два варианта использования системы — в режимах off-line (после окончания счета) и on-line (по ходу счета параллельной программы). Также рассматриваются два варианта размещения средств предобработки и визуализации на процессоры параллельного вычислителя — на те же процессоры, где происходил счет, или на дополнительно выделенные процессоры вычислителя.

Таким образом, в проектируемой системе параллельной визуализации предполагаются три независимых класса модулей, отвечающих за параллельную предобработку и фильтрацию, сбор, хранение и передачу данных и, наконец, визуализацию.

Для создания системы интерактивной визуализации параллельных вычислений выбран проблемно-ориентированный подход — передача отфильтрованных математических данных на графическую станцию для визуализации. Эффективность этого подхода достигается, с одной стороны, за счет независимости от конкретной графической библиотеки, следовательно, возможна аппаратная поддержка, а с другой — за счет параллельной предобработки или фильтрации математических значений, обеспечивающих минимизацию передаваемых данных. В отличие от аппаратно-независимого подхода или переноса стандартных графических библиотек на параллельную технику, при котором возможно применение параллельного рендеринга, использование фильтров кажется более целесообразным. Впрочем, параллельная фильтрация возможна для объектно-ориентированных и модульных графических библиотек, но для своих внутренних данных. На базе проблемно-ориентированного подхода разрабатывается графический сервер приложений, в частности, связанных с задачами на трехмерных сетках.

Разработка интерактивных средств, специально предназначенных для каждой отдельной задачи, трудоемка, а объединение таких средств для выполнения нескольких различных задач в одну систему приводит к значительному увеличению используемых библиотек. Поэтому для ускорения реализации интерактивных графических средств, обеспечивающих параллельное моделирование, необходимо предоставить среду разработки или технологию программирования интерактивных задач. В связи с этим целесообразно позаим-

ствовать ряд идей из систем CUMULVS, Аванго, адаптивного конструктора интерфейсов.

Предложенная схема отрабатывается на задачах математической физики, в частности, связанных с сеточными методами вычислений [22].

В процессе визуализации данных используется стандартная схема специализированной системы визуализации сеточных данных с системой комплексных (множественных) видов отображения, управляемой камерой, и многоступенчатой фильтрацией данных.

Основным видом отображения является отфильтрованный участок сетки (рис. 3), расположенный в центре. Для реализации вспомогательного вида отображения "MiniMap" (слева вверху) используется воксельная графика, обеспечивающая целостное восприятие сетки. Взаимодействовать с основным видом отображения можно только через вспомогательные, кроме "MiniMap" к ним относятся поле диапазонов (range bar справа), обеспечивающее фильтрацию по значению, и визуальные элементы пользовательского интерфейса (всплывающие меню вверху), отображающие введенную информацию. Они также считаются самостоятельными видами отображения, входящими в систему. Однако влияние последних на адекватность восприятия информации в целом невелико, и их изображение во время работы в основном отсутствует.

Для сравнения приведем описание проекта USC-ISI Center for Grid Technologies [23]. Достижения в науке наложили высокие требования на инструменты исследования и анализа высокопроизводительной визуализации широкомасштабных данных. Физические модели могут создавать большое количество данных, возможно, с высоким разрешением (в трехмерном пространстве) и за продолжительное время. Одномашинные реализации визуализации не могут справиться с этими данными. Для того чтобы решить эту проблему, авторы предлагают гибкую и расширяемую структуру визуализации для критичной по времени, интерактивно управляемой передачи набора файлов с целью визуального просмотра больших пространственных и временных наборов данных в Grid-среде. Эта структура использует Grid-ресурсы для масштабируемых вычислений и хранения данных, чтобы увеличить производительность, функциональность и гибкость легковесных односистемных инструментов.

Система использует Globus Toolkit 2.4: компоненты безопасности (GSI), распределения и управления ресурсами (DUROC, GRAM) и обменов (Globus-IO), чтобы соединить преимущества настольных компьютеров с удаленным, распределенным хранением и вы-

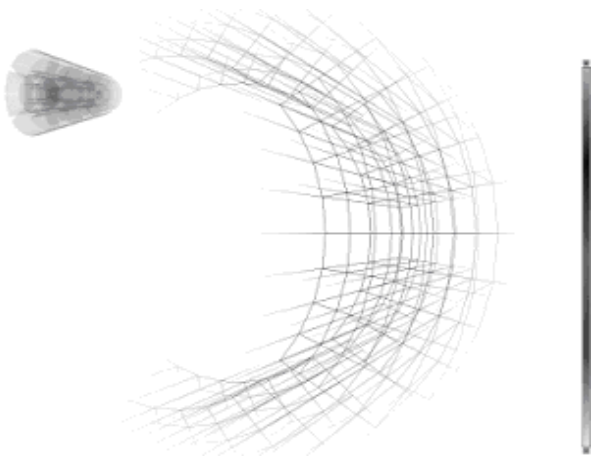

Рис. 3. Участок сетки с шестигранными примитивами с применением воксельной графики ("MiniMap" слева вверху).

числительными Grid-ресурсами для интерактивного исследования данных. Существуют две основные части в системе — это Grid Data Transport (GDT — передача данных) и Grid Visualization Utility (GVU — утилита визуализации). GDT обеспечивает библиотеки для параллельной фильтрации данных и параллельного обмена данными в Grid-ресурсах. GDT позволяет встраивать произвольную фильтрацию данных. Он также облегчает построение многозвенных конвейерных топологий вычислительных ресурсов и дисплеев. В дополнение к приложениям научной визуализации GDT может использоваться для поддержки других приложений, которые требуют параллельной обработки и передачи частично упорядоченных независимых файлов.

На основе GDT разрабатывается GVU, которая предназначена для того, чтобы помочь в управлении визуальным набором данных, включая форматирование файла, передачу данных и автоматический обмен байтами. GVU также поддерживает параметризованные фильтры сокращения данных, такие как поточечный отбор в выбраковке скалярных линий, объемное отсечение и уменьшающий отбор. GVU может использоваться для облегчения параллельного выполнения существующих фильтров трансформации, таких как, например, VTK-фильтр изоповерхностей — марширующие кубы и другие сделанные на заказ проблемно-ориентированные фильтры. Первоначальная реализация поддерживает удаленный синтез дисплейных списков, независимых от точки зрения. Эта особенность позволяет локальной отображающей машине управлять точкой зрения для сокращения латентности рендеринга с несколькими точками зрения и с несколькими видами отображения (стереорендеринг).

В этой системе, в отличие от нашей, параллельный и транспортный модули объединены в один, что несущественно. Происходит передача независимых файлов (данных), а следовательно, и фильтрация независимых данных, что упрощает решение, но накладывает ограничение на класс решаемых задач. Она предназначена для работы в Grid-среде. Возможно, существует пересечение по фильтрам, в любом случае пропагандируется проблемно-ориентированный подход. Из-за отсутствия детальной информации на ряд вопросов невозможно ответить. Хотелось бы иметь больше научных работ в области визуальных супервычислений. Данная работа — один из первых шагов в этом направлении.

Заключение

Конечно, выбор двух этих систем для сравнительного анализа неслучаен. Обе они решают достаточно широкий класс задач в области визуальных супервычислений, в том числе проблемы, связанные с интерактивностью и визуализацией данных большого объема, применяют наиболее эффективные технологии (параллельная фильтрация данных и проблемно-ориентированный подход).

Целью работы является в первую очередь обобщение и систематизация накопленных знаний, чтобы в дальнейшем перейти к более детальному анализу в рамках теории визуальных супервычислений, в частности технологии параллельной фильтрации данных.

Список литературы

[1] Brodlie K., Brooke J., Chen M. et al. Visual Supercomputing — Technologies, Applications and Challenges, Eurographics. STAR Reports. 2004. P. 37–68.

[2] Heermann P.D. Production visualisation for the ASCI one teraFLOPS machine // Proc. of the 9th Annual IEEE Conf. on Visualization (VIS-98), Oct. 18–23 1998. N.Y.: ACM Press, 1998. P. 459–482.

[3] Суков С.А., Якобовский М.В. Обработка трехмерных неструктурированных сеток на многопроцессорных системах с распределенной памятью // Фундаментальные физико-математические проблемы и моделирование технико-технологических систем. М.: МГТУ "СТАНКИН", 2003. Вып. 6. С. 8.

[4] Абалакин И.В., Болдырев С.Н., Жохова А.В. Параллельный алгоритм расчета газодинамических течений на нерегулярных сетках // Фундаментальные физико-математические проблемы и моделирование технико-технологических систем. М.: МГТУ "СТАНКИН", 2000. Вып. 3. С. 41–45.

[5] Байдалин А.Ю. Опыт разработки полнофункционоальной системы визуализации параллельного программирования // Тр. 35-й Регион. школы-конф. "Проблемы теоретической и прикладной математики", Екатеринбург, 26.01–30.01. 2004. С. 303–308.

[6] Aleshin A., Afanasiev V., Baygozin D. et al. Virtual environment visualization system for the tasks of space exploration: Current status // Тр. 14-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2004, 6–10 сент. 2004. М.: МГУ им. М.В. Ломоносова. С. 12–15.

[7] Manakov D., Shagubakov M. Adaptive builder for interactive tasks in mass-parallel machines // Тр. 12-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2002, 16–21 сент. Н. Новгород, 2002. С. 405–408.

[8] Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р. и др. Использование трехмерных метафор визуализации // Тр. 14-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2004, 6–10 сент. 2004. М.: МГУ им. М.В. Ломоносова, 2004. С. 295–298.

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

[10] Авербух В.Л., Байдалин А.Ю. Разработка средств визуализации программного обеспечения параллельных вычислений. Оптимизация программ // Вопр. атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. Вып. 1. С. 70–80.

[11] Ueng S.-K., Sikorski C., Ma K.-L. Out-of-core streamline visualization on large unstructured meshes // IEEE Transactions on Visualization and Computer Graphics. 1997. N 3, 4. P. 370–380.

[12] Manakov D., Mukhachev A., Shinkevich A. Visualization of the distributed data of huge volume. Assembly, filtration, sorting // Тр. 13-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2003, 5–10 сент. 2003. М.: МГУ им. М.В. Ломоносова, 2003. С. 198–201.

[13] Karpov A.N. Data visualization on parallel computer systems // Тр. 15-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2005, 20–24 июля 2005. Новосибирск, 2005. С. 211–214.

[14] Molnar S., Cox M., Ellsworth D., Fuchs H. A sorting classification of parallel rendering // IEEE Computer Graphics and Applications. 1994. Vol. 14, N 4. P. 23–32.

[15] Авербух В.Л., Исмагилов Т.Р., Манаков Д.В. Подходы к реализации интерактивной графики на МВС-1000 // Матер. Всерос. конф. “Высокопроизводительные вычисления и их приложения”. Подмоск. филиал МГУ им. М.В. Ломоносова, Ин-т проблем химической физики РАН, 2000. С. 241–244.

[16] Вельтмандер П.В. Архитектуры графических систем: Учеб. пособие. ermak.cs.nstu.ru/kg_rivs/kg03.htm

[17] Homan I., Gordon S., Pat H. The design of a parallel graphics interface // Proc. of SIGGRAPH’98, July 1998. P. 141–150.

[18] Husbands P., Isbell Ch. The parallel problem server: a client-server model for large-scale scientific computation. www.mit.edu/parry/ppserver.ps

[19] Manakov D., Komarovsky I. Principles of automatic generation and parsing of device-independent metafiles by the example of object-oriented VTK library adaptation // Тр. 13-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2003, 5–10 сент. 2003. М.: МГУ им. М.В. Ломоносова, 2003. С. 196–197.

[20] IBM: Autonomic deployment model. www-306.ibm.com/autonomic/levels.shtml, 2004.

[21] Авербух В.Л., Васев П.А., Горбашевский Д.Ю. и др. Система интерактивной визуализации параллельных вычислений // Тр. 14-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2004, 6–10 сент. 2004. М.: МГУ им. М.В. Ломоносова. С. 291–294.

[22] Горбашевский Д.Ю., Казанцев А.Ю. Визуализация сеточных данных большого объема // Тр. 15-й Междунар. конф. по компьютерной графике и зрению — ГрафиКон-2005, 20–24 июля 2005. Новосибирск, 2005. С. 366–367.

[23] Thiebaux M., Tangmunarunkit H., Czajkowski K., Kesselman C. Scalable Grid-Based Visualization Framework. Technical Report ISI-TR-2004-592, USC/Information Sciences Institute, June 2004.

Поступила в редакцию 24 апреля 2006 г.,
в переработанном виде — 10 октября 2006 г.