УДК 519.6

СРЕДА ПОДДЕРЖКИ ИНТЕРАКТИВНОЙ ВИЗУАЛИЗАЦИИ ДЛЯ СУПЕРКОМПЬЮТЕРНЫХ ВЫЧИСЛЕНИЙ

П. А. Васёв
(ИММ УрО РАН, г. Екатеринбург)

Возможность интерактивного взаимодействия исследователя с выполняющейся *параллельной программой* полезна для сокращения времени проведения вычислительных экспериментов и отладки. Представлена простая и удобная программная среда, реализующая различные сценарии интерактивного взаимодействия. Система предназначена для наблюдения и управления задачами малых и средних размеров, позволяет проводить удаленную визуализацию с применением современных технологий построения пользовательского интерфейса и может применяться как в локальных, так и глобальных сетях.

Ключевые слова: визуализация, параллельные вычисления, интерфейс.

Введение

Классический подход к современным высокопроизводительным вычислениям подразумевает пакетное выполнение *параллельных программ*. При этом в определенных случаях практически ценными оказываются наблюдение за состоянием считающейся программы и возможность управления ею по ходу выполнения. Это полезно как во время отладки алгоритма и программы, так и во время проведения вычислительных экспериментов. Основной выгодой здесь является сокращение сроков разработки программ и возможность провести большее количество экспериментов.

Возможность визуализации состояния программы во время ее выполнения реализуется с помощью так называемой *онлайн-визуализации*. Приставка *онлайн* заимствована из английского языка и означает дословно *на линии*, т. е. в режиме реального времени, когда оба абонента — расчетная программа и пользователь — активны и могут взаимодействовать друг с другом. Также ее иногда называют визуализацией *по ходу вычислений* — в противовес традиционной визуализации после вычислений, которая проводится после полного завершения расчетов.

Онлайн-визуализация не ограничивается какой-то единой технологией или подходом. В простейшем варианте это может быть вывод по ходу счета значений переменных программы в файл. Более сложные случаи требуют наличия специальной системы для поддержки онлайн-визуализации.

Кажется интересным наблюдение, что изначально компьютерная техника поддерживала режим визуализации по ходу вычислений. Тогда расчеты проводились непосредственно у пультов вычислительных машин либо через терминалы, и пользователь обладал всеми необходимыми возможностями по оперативному управлению программой.

С ростом вычислительных мощностей, а особенно с появлением *параллельной техники*, вырос и объем результатов счета, что сделало невозможным непосредственный контроль программы пользователем. Другая проблема связана с тем фактом, что в рамках библиотек поддержки параллельных вычислений — PVM [1], MPI [2] — оставлены без внимания как вопросы оперативного управления программами, так и визуализация состояния выполняющихся программ. Соответственно, чтобы "вернуть" прикладным программистам контроль над задачами, требуется разработка специальных технологий в дополнение к имеющимся системам параллельного программирования.

В настоящей работе проанализированы различные способы организации таких технологий.

Рассматриваются варианты распределения и передачи данных, а также вопросы внедрения систем онлайн-визуализации в инфраструктуру проведения расчетов.

Предлагается простая и удобная система онлайн-визуализации общего назначения. Система предназначена для наблюдения и управления задачами малых и средних размеров, но при этом поддается масштабированию.

Отдельный интерес представляет разрабатываемый универсальный *веб-интерфейс*, который поддерживает взаимодействие с определенным классом задач. Под веб-интерфейсом подразумевается ориентация системы на работу в современных интернет-браузерах, что делает визуализацию доступной практически с любого компьютера, из любого местоположения. Данный веб-интерфейс позволяет подключаться к программе "на лету", визуально анализировать ее состояние, изменять значения переменных и посылать специальные команды этой программе.

Другой особенностью системы является поддержка удаленной визуализации. В этом сценарии изображение создается удаленно — в вычислительной параллельной программе или отдельной специальной программе — и передается по сети конечному пользователю. Пользователь может взаимодействовать с этим изображением посредством мыши и элементов управления, что влечет перерисовку изображения на удаленной системе и его повторную передачу. Такой сценарий интересен, когда выгоднее передавать по сети именно изображение, а не математические объекты, из которых оно создается.

Варианты размещения данных и системы рендеринга

С технической точки зрения визуализация включает в себя этапы загрузки данных, преобразования этих данных в визуальные сущности, их *рендеринг* (т. е. процесс перевода трехмерной модели в двумерное изображение) и отображение на дисплее или другом устройстве [3]. Это относится как к онлайн-визуализации, так и к визуализации после счета. В связи с наличием этих этапов для архитектуры системы визуализации существенными являются следующие вопросы.

1. Расположение данных — локальное или удаленное. Под локальным расположением понимается наличие файлов на компьютере пользователя. Удаленное размещение подразумевает такую ситуацию, когда данные невозможно или неэффективно целиком передать на ЭВМ пользователя для визуализации.
2. Расположение системы рендеринга — локальное или удаленное. Локальный рендеринг задействует мощности компьютера пользователя. Удаленное расположение системы рендеринга — на внешних компьютерах (или даже на самом вычислителе) — может быть оправдано в различных случаях, например, когда процесс рендеринга требует вычислительных мощностей, превышающих возможности ЭВМ пользователя.

Обратимся к рассмотрению различных комбинаций этих решений.

1. *Данные и рендеринг — локальные.* Этот случай самый простой и удобный для работы, если считать, что время чтения и визуализации данных приемлемо. Плюсом этого подхода является относительная простота реализации системы визуализации.
2. *Данные удаленные, рендеринг локальный.* Возникает необходимость создания технических средств для удаленного чтения и передачи всех данных или их части. Например, это может быть программа, запущенная на компьютере, содержащем данные, которая по команде считывает и передает часть данных на компьютер пользователя. Команды на чтение и передачу соответственно инициируются системой рендеринга в зависимости от текущего вида и параметров отображения [4]. При этом локальный рендеринг обеспечивает оперативность взаимодействия с системой — например, поворот трехмерной сцены с помощью мыши может выглядеть непосредственным и плавным.

   Важной задачей, которую приходится решать в случае удаленных данных и локального рендеринга, является выбор правил передачи данных. Например, в широком спектре задач при визуализации расчетных сеток успешным является подход, когда данные огрубляются без потери когнитивности на удаленной ЭВМ, что позволяет передать их по сети и отобразить на локальной ЭВМ [5].
3. *Данные и рендеринг — удаленные.* В этом случае изображения создаются на специ-

альном компьютере и передаются по сети на компьютер исследователя. Исследователь может управлять визуализацией, влияя на процесс создания изображений, которые немедленно появляются на его компьютере.

Актуальной является ширина канала между ресурсом, содержащим данные, и системой рендеринга. Если этот канал обладает достаточной пропускной способностью, то система передачи данных может оказаться достаточно простой. Еще одним плюсом такого подхода является то, что система рендеринга может быть распределенной. При этом требование к ширине канала между этой системой и компьютером пользователя в общем случае определяется фиксированной величиной, зависящей от интенсивности графического обмена для данной задачи.

Нельзя утверждать, что какой-либо из перечисленных подходов предпочтительнее других. Выбор той или иной конфигурации определяется в первую очередь решаемой задачей и имеющимся системным окружением.

Способы поддержки онлайн-визуализации

Рассмотрим различные способы, которые позволяют наблюдать и управлять вычислительной задачей [6]. В настоящий момент наиболее распространено пакетное выполнение программ, которое, с одной стороны, накладывает свои ограничения на визуализацию, а с другой стороны, требует особого подхода. Способы визуализации призваны в первую очередь "вытащить" данные из программы и показать их.

Интеграция с параллельными программами. Идея заключается в том, чтобы внедрить в исходный код параллельной программы несколько специальных вызовов функций, которые и будут связывать программу с системой онлайн-визуализации. Особенность рассматриваемого подхода — минимизация необходимых изменений, которые потребуется внести в исходный код существующего проекта. Именно поэтому данный подход достаточно распространен. В частности, он реализован в системе pV3 [7], которая является *параллельной версией* стандартного модуля визуализации для пакета ANSYS CFD [8]. В качестве другого примера можно привести систему CUMULVS [9], которая не только позволяет визуализировать состояние программы, но и содержит поддержку контрольных точек.

В качестве примера вызовов функций, которые встраиваются в параллельную программу, можно привести передачу системе визуализации информации о структуре и расположении данных в памяти. Таким образом, при счете система сможет получить внешний запрос (инициированный пользователем), прочитать данные и отреагировать на него соответствующим образом. Этот пример является основой для предлагаемой архитектуры, изложенной ниже.

Вычислительные среды. Этот подход, в отличие от интеграции, подразумевает иной способ написания параллельных программ. Если в подходе с интеграцией предлагается дописать существующий проект, то в случае вычислительных сред программу надо переписать. Программисту предлагается *каркас*, являющийся логичной и законченной моделью параллельных вычислений. В таком каркасе уже реализованы общие операции — считывание исходных данных, предварительная обработка сеток, передача данных между узлами, запись сеток в различных форматах, визуализация.

Считается, что после освоения новой парадигмы и получения опыта в создании счетных модулей, взаимодействующих со средой по специальным интерфейсам, сложность написания вычислительных программ уменьшится.

Сервисы поддержки вычислений. Это вариант вычислительной среды, когда вместо полной модели вычислений программисту предлагается набор сетевых функций, упрощающих взаимодействие между модулями программы и работу с данными — чтение, запись, передачу. Например, программа считывает участок расчетной сетки не напрямую из файла, а делает запрос к сервису поддержки вычислений, который реализует считывание данных в определенном формате и организует их передачу. В этот сервис включается также система визуализации либо интерфейс для связывания с какой-нибудь внешней системой.

Кроме выполнения рутинных операций по чтению-записи сеток, другое основное назначение сервисов промежуточного уровня — замена обычного режима взаимодействия различных компонентов системы (генератор сеток, разбиение на домены, счет, визуализатор). В этом слу-

чае взаимодействие реализуется не через промежуточные файлы, а с помощью сетевых обменов. Такова, например, основная идея в системе Hercules [10].

Другой полезной функцией сервисов поддержки вычислений является возможность специализированного хранения и архивирования данных. Такой подход применяется в средах обеспечения вычислений некоторых ведущих научных центров России.

Новая система онлайн-визуализации

Идея предлагаемого решения близка к идеям, реализованным в уже упоминавшемся проекте CUMULVS [9] Окриджской национальной лаборатории. Остановимся более подробно на модификациях, внесенных в предложенный зарубежными коллегами подход, и некоторых аспектах программной реализации новой системы, осуществленной в ИММ УрО РАН.

Система состоит из трех компонентов:

1) вычислительных узлов (параллельной программы);
2) сервиса-посредника;
3) программ визуализации.

Вычислительные узлы — это процессы счетной программы с внедренными в нее специальными вызовами функций, которые и соединяют ее с системой визуализации. Посредник — "сердце" системы — принимает команды от программ визуализации и направляет их к вычислительным узлам. Программы визуализации — это набор программ, реализующих интерфейс пользователя, которые отображают текущее состояние параллельной программы и позволяют управлять ею. Для каждой параллельной программы подразумевается наличие своей собственной программы визуализации (при этом возможны и унифицированные решения, одно из них представлено в конце статьи).

Система основана на принципе *запрос—ответ*. Счетная программа после запуска выполняется в штатном режиме. Исследователь, производя те или иные действия, воздействует на программу визуализации, которая, в свою очередь, обращается к посреднику с командой или запросом на данные. Посредник направляет этот запрос к вычислительным узлам. Ответ возвращается через сервис-посредник программе визуализации, которая отображает результат выполнения запроса. На рис. 1 показана общая схема системы.

Все запросы основаны на именах. Каждая параллельная программа при запуске регистрируется у посредника со своим уникальным именем. А точнее, каждый ее процесс указывает это имя и свой порядковый номер. Затем каждый вычислительный процесс регистрирует у посредника свои ресурсы — данные или коммуникационные примитивы. Этот этап назван *публикацией данных*. Например, процесс параллельной программы `CALC_1` может опубликовать в системе массив чисел с плавающей точкой и дать ему имя `DNA_DATA`. Чтобы прочитать данные этого массива, программа визуализации должна будет указать имя задачи `CALC_1` и идентификатор данных `DNA_DATA`.

Запросы на данные подразумевают операции чтения и записи. При этом поддерживаются запросы и на часть данных — например, можно прочитать данные массива только с i-го по j-й элемент.

По умолчанию все операции чтения и записи производятся асинхронно. Это означает, что после публикации данных вычислительный процесс продолжает выполняться, производя расчеты. Когда же эти данные станут необходимы системе визуализации, они будут прочитаны из памяти вычислительного процесса асинхронно (из параллельного потока), вне зависимости от хода основного счета. Этого, как до сих пор показывает практика, вполне достаточно для большинства случаев. Конечно, в более сложных ситуациях потребуется использование примитивов синхронизации, которые также представлены в системе.

Запросы на данные могут содержать обращения к распределенным данным. Так, например, несколько вычислительных узлов могут опубликовать какой-либо массив с одним и тем же име-

Рис. 1. Архитектура предлагаемой системы

нем, но с указанием различных участков этого массива. Это распределение становится известным сервису-посреднику, и запрос на *обобщенные* данные распределяется им между узлами-участниками. Затем посредник посылает программе визуализации уже объединенные результаты.

Кроме запросов на данные, программы визуализации могут создавать запросы на управление. Например, это может быть запрос на состояние программы или узла (выполняется или ждет, стадия выполнения, список ресурсов и т. д.) или запрос на управление объектами синхронизации.

В рамках системы существует два программных интерфейса. Один предоставляется вычислительным узлам, другой — программам визуализации. Интерфейс для параллельных программ — это набор высокоуровневых вызовов, реализованных в специальной библиотеке. Для программ визуализации предоставляется интерфейс SOAP [11]. Сама же система внутри полностью ориентирована на применение этого сетевого протокола. При этом подчеркнем, что интерфейс для *параллельного программиста* — это не SOAP, а функции C или Fortran'a.

Таким образом, для того чтобы сделать параллельную программу управляемой и поддерживающей онлайн-визуализацию, программисту необходимо внедрить ряд вызовов функций и скомпоновать программу со специальной библиотекой. Покажем это на следующем примере для программы на языке Fortran:

```
* Некоторые данные программы
   dimension *x(117), *x0(117)
* Подключение системы онлайн-визуализации
   call iv_init('TASK_CALC_XG',myrank)
* Публикация данных
   call iv_publish('CALC_X',x,117)
   call iv_publish('CALC_X0',x0,117)
   ....
* Параметры вычислительного алгоритма
   alfa = 1.E-11
   mu = 0.01
* Публикация параметров
   call iv_parameter('CALC_ALPHA',alfa)
   call iv_parameter('CALC_MU',mu)
* Далее следуют вычисления.  Данные будут
* прочитаны и/или записаны асинхронно.
```

После оснащения параллельной программы указанными вызовами функций системы любой сетевой клиент может присоединиться к посреднику с помощью протокола SOAP и запросить содержимое ресурса `CALC_X`, который в данном случае является массивом `x(17)`, расположенным в оперативной памяти конкретного узла. В дополнение сетевой клиент может изменить содержимое этого массива или, например, переменной `alfa`, выполнив соответствующий запрос на запись.

Среди примитивов синхронизации в системе представлены *точки сбора* (joins) и так называемые *барьеры*. Точка сбора — это именованное место в узлах параллельной программы. Когда процесс достигает точки сбора, его выполнение приостанавливается до тех пор, пока в точке сбора не окажется определенное число таких процессов. Барьер — это также место в коде программы с указанием имени переменной-счетчика. Когда процесс достигает барьера, анализируется значение упомянутого счетчика. Если значение положительно, оно уменьшается на единицу, а процесс продолжает свою работу. Если значение равно нулю, процесс переходит в режим ожидания — до тех пор, пока действиями извне (например, программой визуализации) не увеличится значение счетчика. Барьеры и точки сбора показали свою полезность при интерактивном управлении вычислениями.

Относительно создания сетевых клиентов можно сказать следующее. Для каждой вычислительной программы необходима разработка собственной клиентской программы визуализации. Теоретически любая среда с поддержкой SOAP для этого подходит. В ИММ УрО РАН успешно реализован ряд клиентов в средах Borland C++ [12], Adobe Flex 2 [13] и HTML/Javascript [14].

Когда все подготовительные работы завершены, исследователь обычно выполняет действия в следующем порядке: запускает параллельную программу, а затем, по мере необходимости, открывает программу визуализации. Эта программа подключается к вычислительной задаче, отображает ее состояние и позволяет исследователю управлять задачей.

В примере работы предлагаемой системы, приведенном на рис. 2, показаны результаты выполнения программы трехмерного моделирования течения вязкой жидкости при определенных воздействиях [15]. В памяти узлов параллельной программы формируется трехмерный массив. Пользователь работает с клиентским приложением, выбирает параметр для построения

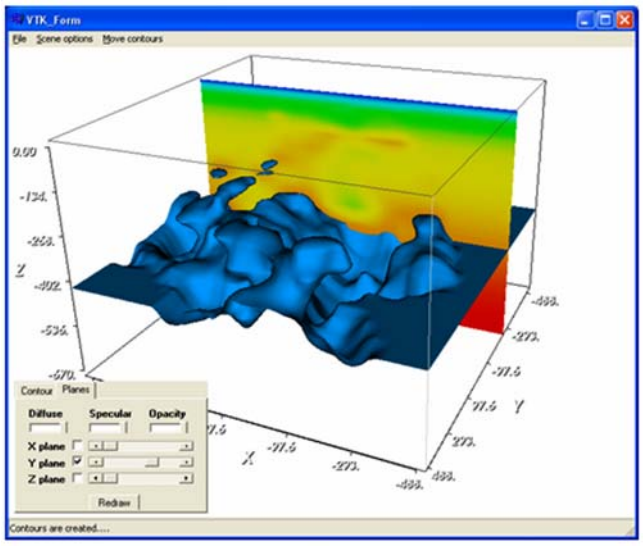

Рис. 2. Пример работы системы

изоповерхности, и соответствующий запрос на построение такой поверхности обрабатывается в процессах задачи. Справедливо будет подчеркнуть, что передача изоповерхностей без специального прореживания (которое возможно — [5]) оказалась очень длительной по времени.

Другим примером может послужить применение системы для проведения численных экспериментов при решении некоторых одномерных задач, которое более подробно описано ниже.

Система показала свою применимость и расширяемость. Среди плюсов можно отметить возможность работы в любой среде параллельного исполнения, с языками C/C++ и Fortran, поддержку операционных систем Windows и Linux.

Веб-интерфейс для системы онлайн-визуализации

Как отмечено выше, в рамках предлагаемой системы подразумевается создание индивидуальной программы визуализации для каждой конкретной вычислительной программы. По-видимому, существуют потребность и возможность для создания универсальных программ визуализации. Далее представлена программа, ориентированная на работу с одномерными расчетными задачами.

Программа является дополнением к сервису-посреднику, которое позволяет пользователю с помощью веб-интерфейса наблюдать и управлять своей задачей. Повторимся, что под веб-интерфейсом понимается такая организация системы визуализации, когда она доступна для работы из интернет-браузеров, которые в настоящее время установлены практически на любом персональном компьютере. При запуске (т. е. при открытии веб-страницы с определенным адресом) программа отображает список опубликованных данных, а также графическое представление опубликованных одномерных массивов.

На рис. 3 показан визуальный интерфейс одной из версий программы, рассчитанный на решение некорректных задач методами регуляризации и итерационной аппроксимации [16]. В данном случае имеет место серия вычислительных экспериментов. На графиках показано состояние основной вычисленной функции, а также некоторые важные производные и история последних шагов счета.

Разрабатывается новая версия указанного интерфейса, которая позволяет отображать не только одномерные массивы, но и результат операций над ними, например, разность двух массивов или их совмещение. Также внедряется поддержка изменения любых параметров, опубликованных в программе. Этот подход успешно опробируется, в частности, на решении ряда криптографических задач [17].

Отдельный интерес представляет поддержка отображения не только одномерных данных, но двумерных и трехмерных. Двумерные данные можно передавать и в графическом, и в исходном виде (возможно, с применением фильтрации) и растеризовать локально на компьютере пользователя.

С трехмерными данными дело обстоит несколько сложнее. В настоящее время тех-

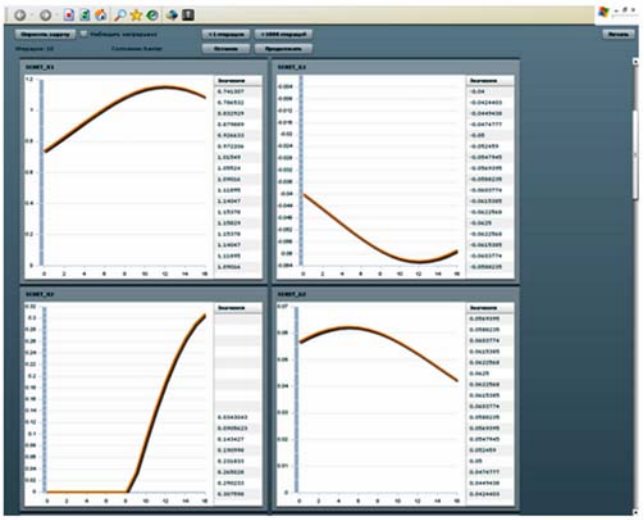

Рис. 3. Пример веб-интерфейса пользователя

нологии уровня Flash [13] уже поддерживают аппаратный рендеринг трехмерных сцен, однако проблема сообразной компрессии данных для передачи по сети [5] сохраняется.

Одним из многих вариантов решений указанной проблемы является применение подхода удаленной визуализации, который при определенных технических условиях может быть вписан в любой веб-интерфейс.

Поддержка удаленной визуализации

Удаленная визуализация подразумевает такую модификацию системы визуализации для научных исследований, когда изображения создаются на специальном компьютере и передаются по сети на компьютер исследователя. Исследователь может управлять визуализацией, влияя на процесс создания изображений, которые немедленно появляются на его компьютере.

В этом случае не предъявляется серьезных аппаратных требований к рабочему компьютеру исследователя. Кроме того, время передачи графических изображений в достаточно широком спектре задач меньше, чем при передаче даже отфильтрованных данных. При этом требования к ширине канала для передачи именно изображений на практике оказываются приемлемыми, что будет показано на примере ниже.

Отметим, что при данном подходе программное обеспечение, реализующее как обработку данных, так и рендеринг, концентрируется на выделенных мощностях. Таким образом, появляется возможность осуществлять указанные процессы в непосредственной близости к вычислительным ресурсам. Это существенно сокращает время, необходимое на чтение данных, и экономит пропускную способность сети. Кроме того, централизованное программное обеспечение проще модифицировать и администрировать.

В первую очередь хотелось бы упомянуть существующие технологические решения, которые позволяют осуществлять удаленную визуализацию. В ИММ УрО РАН исследованы возможности систем Windows Terminal Services 2008 [18], Citrix XenApp [19], решений VNC [20] и X Window [21], а также ряда менее известных систем. Анализ показывает, что ведущие разработчики этих решений только приступают к поддержке программ с аппаратной трехмерной графикой. Так, в пресс-релизах компаний Microsoft и Citrix 2008 года заявлено, что следующие версии их продуктов будут непосредственно поддерживать приложения, использующие DirectX [22] и OpenGL [23]. Безусловно, данные заявления вселяют определенную уверенность и желание применить эти широко используемые технологии, однако в настоящий момент единственно возможный путь — рассмотрение альтернативных решений.

Отдельно стоит вопрос использования технологий VNC. Применение этих подходов означает возникновение двух проблем: отображение всего рабочего стола (с размещенной на нем системой визуализации) и непосредственный пользовательский доступ к внутренним узлам системы визуализации. Отметим, что в определенных организациях и системах такие проблемы не являются помехой для применения данных технологий.

Рассмотрим возможные альтернативные варианты решения вопроса удаленной визуализации, включая задачи рендеринга трехмерных сцен с результатами расчетов. Среди флагманов настоящего времени можно отметить Paraview Enterprise Edition [24], Sun Visualization System [25], HP Scalable Visualization Array [26], Mental Images Reality Server [27] и StreamMyGame [28]. Подчеркнем, что большинство из этих решений ориентировано на пользовательский интерфейс, базирующийся на веб-технологиях. Среди представленных систем отдельно хотелось бы выделить следующие:

Paraview Enterprise Edition — содержит веб-интерфейс к системе визуализации Paraview для организации удаленной визуализации. Paraview является распространенным инструментом для визуализации научных данных, включая результаты различных расчетов. Среди плюсов программы — богатый набор видов отображения и различные варианты конфигурации распределенного рендеринга. Минус — необходимость конвертации данных в формат Paraview.

Reality Server — является серверной платформой для создания 3D-веб-приложений. Система содержит инфраструктуру для рендеринга трехмерных сцен, передачи изображений в веб-интерфейс и обеспечивает прием команд от пользователя веб-интерфейса с последующей обработкой их на сервере. Су-

ществует возможность как настройки веб-интерфейса, так и программирования серверной логики. Среди плюсов программы можно отметить поддержку аппаратного рендеринга. Среди минусов — закрытость разработки. Кроме того, отдельной проблемой является необходимость конвертации данных в формат сцен *.m*, применяющийся в Reality Server.

В ИММ УрО РАН было принято решение о создании новой системы удаленной визуализации. Это позволит в будущем глубже понимать проблематику данной технической области, а также производить достаточно гибкие изменения, которые будет требовать практика.

Удалось использовать уже имеющиеся наработки — систему онлайн-визуализации, расширив ее возможности. Программа, которая обеспечивает функцию рендеринга, — модуль визуализации — подключается к системе точно так же, как и узлы счетной задачи при онлайн-визуализации. Модуль визуализации сообщает системе адрес функции, которая осуществляет рендеринг. Эта функция вызывается системой асинхронно, в потоке, отличном от основного. На вход ей поступает требуемое разрешение изображения, а на выходе функция должна вернуть созданный растр.

Обратимся к общей схеме работы (рис. 4). Пользователь взаимодействует с системой через веб-интерфейс, в котором представлены элементы управления для модуля визуализации. В основе этого интерфейса — сгенерированное визуальное представление данных. Исследователь может оказывать, например с помощью мыши, воздействие на изображение, а также на элементы управления. При воздействии веб-интерфейс направляет соответствующие команды сервису-посреднику (на рис. 4 он представлен системным модулем) с помощью протокола HTTP Ajax [29]. Посредник, в свою очередь, транслирует эти команды конкретному процессу модуля визуализации. Это могут быть запросы на изменение тех или иных параметров либо на перерисовку изображения. В случае запроса на перерисовку в модуле визуализации вызывается уже упоминавшаяся функция рендеринга. После отработки этой функции изображение упаковывается (сжимается) в определенный формат и передается через сервис-посредник в браузер пользователя.

Правилам сжатия изображений требуется уделить особое внимание. На текущий момент изображения упаковываются в формат PNG [30] по следующей схеме. При первом запросе на кадр система высылает изображение, уменьшенное в 4 раза. В случае, если пользователь не проявляет

Рис. 4. Общая схема системы удаленной визуализации

какой-либо активности в течение нескольких секунд, посылается улучшенная копия изображения с коэффициентом уменьшения 2. Если в течение последующих секунд пользователь не проявляет управляющей активности, то передается и показывается исходное изображение. Этот подход позволяет уменьшить объем данных, передаваемых по сети. Действительно, если пользователь с помощью мыши вызывает интерактивное вращение трехмерной сцены, то с точки зрения восприятия важно показать результат быстро, пусть и огрубленный. Для одного конкретного приложения, которое будет приведено ниже, размер изображения PNG с разрешением 640×480 в среднем составляет 60 Кб, а уменьшенного в 4 раза — 5 Кб. Соответственно при поворотах сцены передается 10—50 изображений размером всего в 5 Кб. Это позволяет сократить время отклика системы.

Хотелось бы объяснить, почему используется формат PNG, а не JPEG [31]. По наблюдениям автора, использование формата PNG более предпочтительно из-за применения компрессии без потерь. При необходимости система выполняет дополнительную компрессию, сжимая изображение по осям. При этом сжатое, а потом растянутое изображение визуально выглядит более приемлемо при динамических изменениях (например, во время вращения сцены), чем в случае сжатия JPEG с потерей качества. Последний формат, хотя и предоставляет лучший коэффициент сжатия, но изображения в нем в динамике воспринимаются неаккуратными.

Отдельным вопросом является генерация веб-интерфейса. На текущий момент он создается вручную для каждой конкретной задачи и размещается на сервисе-посреднике, который играет роль веб-сервера. В ближайшей перспективе — разработка инструментария для упрощения создания таких веб-интерфейсов, например, за счет создания библиотеки элементов управления с автоматической синхронизацией с параметрами задачи.

На рис. 5 показан один из шагов расчета процесса определенного воздействия на сечение трубы нефтепровода [32]. Пользователь может нажать кнопку мыши на изображении и перемещать ее, при этом изображение будет немедленно обновляться, показывая вращение объекта. Рендеринг осуществляется на удаленном сервере визуализации аппаратно специальной программой с применением технологии DirectX. В локальной сети с пропускной способностью

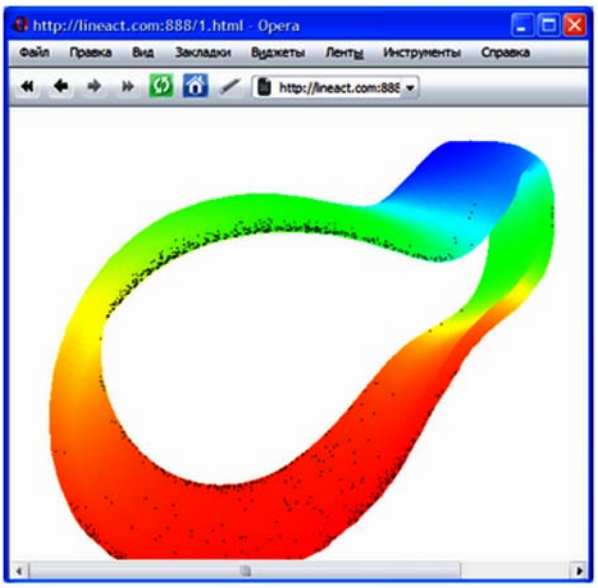

Рис. 5. Пример использования системы удаленной визуализации

10 Мбит/с удалось достичь скорости работы 15 кадров в секунду. В эксперименте с визуализацией через Интернет по каналу Екатеринбург—Москва—Челябинск система показала производительность около 3 кадров в секунду.

Заметим, что модуль визуализации не привязан к конечному типу оборудования — это может быть отдельное приложение на сервере визуализации или параллельное приложение, работающее на вычислителе. Более того, функции модуля визуализации может выполнять часть узлов вычислительной программы. Таким образом, система не навязывает архитектуру для модуля визуализации и может применяться как для оффлайн- так и для онлайн-визуализации.

Заключение

Предлагаемая система показала свою применимость и расширяемость; она функционирует с любой средой параллельного исполнения, с языками C/C++ и Fortran, поддерживает Windows, Linux и переносима на другие операционные системы. Веб-интерфейс не зависит от браузера и подразумевает наличие технологии Flash. Поддержка удаленной визуализации при этом не требует наличия Flash и ориентирована на браузеры с включенным интерпретатором Javascript.

Одно из направлений, которое представляется перспективным, — развитие веб-интерфейса к системе онлайн-визуализации. В идеале можно

получить такой сервис онлайн-визуализации, когда прикладному программисту достаточно внедрить в исходный код несколько функций публикации данных и немедленно получить возможность видеть и изменять эти данные по ходу выполнения программы.

Удаленная визуализация также может приносить пользу в тех случаях, когда выгодней и проще растеризовать данные удаленно, без их передачи на компьютер пользователя.

Созданная реализация системы показала достаточно высокое удобство использования и восприятия — до 15 кадров в секунду, что соизмеримо по скорости с визуализацией на компьютере пользователя.

К настоящему моменту разработан модуль для визуализации определенного класса расчетных сеток. При этом необходимо подчеркнуть, что в общем случае в системе подразумевается создание индивидуального модуля визуализации для каждой конкретной задачи. Перспективным направлением, возможно, является создание набора универсальных модулей визуализации для определенных классов задач. В их основу могла бы быть положена какая-либо существующая система визуализации для научных исследований.

В заключение автор выражает искреннюю благодарность и признательность В. Л. Авербуху за научное руководство; С. В. Шарфу, Д. В. Манакову и М. В. Якобовскому (ИММ РАН, г. Москва) за конструктивные обсуждения проблематики; М. О. Бахтереву и А. Ю. Казанцеву за содействие в продумывании и реализации изложенных идей, а также уважаемым коллегам из Сарова (РФЯЦ-ВНИИЭФ) и Челябинска (ЮУрГУ) за поддержку и плодотворный обмен идеями.

Список литературы

1. *Sunderam V. S.* PVM: A framework for parallel distributed computing // Concurrency: Practice and Experience. 1990. Vol 2, No 4. P. 315—339.
2. MPI: The Message Passing Interface. http://www.mcs.anl.gov/research/projects/mpi.
3. *Авербух В. Л., Манаков Д. В, Васёв П. А. и др.* Подходы к реализации средств on-line визуализации параллельных вычислений // Супервычисления и математическое моделирование: Тез. межд. семинара. Саров: РФЯЦ-ВНИИЭФ, 2003. С. 14—16.
4. *Потехин А. Л., Тарасов В. И., Фирсов С. А. и др.* ScientificView — система параллельной постобработки результатов, полученных при численном моделировании физических процессов // Тр. межд. конф. по компьютерной графике и визуализации "Графикон". М.: МГУ, 2008. С. 65—69.
5. *Якобовский М. В.* Решение сеточных задач на распределенных системах // Параллельные вычислительные технологии. Тр. науч. конф. Т. 2. Челябинск: ЮУрГУ, 2007. С. 201—211.
6. *Васёв П. А., Манаков Д. В., Шинкевич А. Н.* Основные направления развития визуальных супервычислений // Параллельные вычислительные технологии. Тр. науч. конф. С.-Пб. Челябинск: ЮУрГУ, 2008. С. 328—331.
7. *Haimes R.* A Tractable Approach to understanding the results from large-scale 3D transient simulations // AIAA. Reno, NV. Jan. 2001. Paper No 2001—0918.
8. ANSYS ICEM CFD. http://www.ansys.com/products/icemcfd.asp.
9. *Geist G. A., Kohl J. A., Papadopoulos P. M.* CUMULVS: Providing fault-tolerance, visualization and steering of parallel applications // Int. J. of High Performance Comp. Appl. 1997. Vol. 11, No 3. P. 224—236.
10. *Tu T., Yu H., Ramirez-Guzman L. et al.* From mesh generation to scientific visualization — an end-to-end approach to parallel supercomputing // SC2006. Tampa, FL. November 2006. Article No 91.
11. SOAP. Simple Object Access Protocol. http://ru.wikipedia.org/wiki/SOAP.
12. Borland C++. http://www.codegear.com/downloads/free/cppbuilder.
13. Adobe Flex, Flash. http://www.adobe.com/products/flex.
14. HTML, Javascript. http://www.w3schools.com/JS/default.asp.
15. *Короткий А. И., Цепелев И. А.* Трехмерное моделирование прямых и обратных задач Рэлея—Бенара и Рэлея—Тейлора // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2002. Вып. 3. С. 22—32.

16. *Васин В. В., Шарф С. В., Серёжникова Т. И., Васёв П. А.* О распараллеливании и визуализации при решении некорректных задач методами регуляризации и итерационной аппроксимации на вычислительном комплексе МВС-1 000 // Параллельные вычислительные технологии. Тр. межд. науч. конф. Челябинск: ЮУрГУ, 2008. С. 227—233.
17. *Albrekht I.* Some methods for DLP solving and time estimation // Proc. of Information Security Conference in Yekaterinburg. Russia, USU, 2005. P. 35—36.
18. Microsoft Windows Terminal Services 2008. http://www.microsoft.com/windowsserver2008/en/us/rds-product-home.aspx.
19. Citrix XenApp. http://en.wikipedia.org/wiki/Citrix XenApp.
20. VNC: Virtual Network Computing. http://ru.wikipedia.org/wiki/Virtual Network Computing.
21. X Window System. http://ru.wikipedia.org/wiki/X Window System.
22. Direct X. http://msdn.microsoft.com/directX.
23. Open GL. http://www.opengl.org.
24. Paraview Enterprise Edition. http://www.paraview.org.
25. Sun Visualization System. http://www.sun.com/servers/visualization.
26. HP Scalable Visualization Array. http://www.hp.com.
27. Mental Images Reality Server. http://www.mentalimages.com/products/realityserver.html.
28. Stream My Game. http://streammygame.com.
29. HTTP, AJAX. http://ru.wikipedia.org/wiki/AJAX.
30. PNG. http://www.w3.org/TR/PNG.
31. JPEG. http://www.jpeg.org.
32. *Радченко Г. И., Дорохов В. А., Насибулина Р. С. и др.* Технология создания виртуальных испытательных стендов в грид-средах // Вторая межд. науч. конф. "Суперкомпьютерные системы и их применение" (SSA'2008). Минск, 27—29 октября 2008 г. Минск: ОИПИ НАН Беларуси, 2008. С. 194—198.

Статья поступила в редакцию 12.12.08.