УДК 519.6

СИСТЕМА ПОДДЕРЖКИ МЕТАКОМПЬЮТЕРНЫХ РАСЧЕТОВ X-COM: АРХИТЕКТУРА И ТЕХНОЛОГИЯ РАБОТЫ

М. П. Филамофитский[1]

Рассматриваются подходы к организации распределенных вычислений через сеть Интернет. Приводится описание архитектуры и основных возможностей системы X-Com, созданной в Научно-исследовательском вычислительном центре Московского государственного университета.

Ключевые слова: распределенные вычисления, распределенная обработка данных, метакомпьютинг, метакомпьютер, Интернет, X-Com, параллельные вычисления.

1. Введение. Параллельные вычисления и распределенная обработка данных все больше входят в нашу жизнь — то, что еще 10 лет назад было привилегией избранных, теперь становится массовым. В наше время установка многопроцессорного сервера в информационном центре крупной компании уже ни у кого не вызывает излишних восторгов, а является вполне обычным делом.

Все первые параллельные вычислительные системы были уникальны: оригинальная архитектура, последние достижения микроэлектроники, специальное программное обеспечение, специализированные сервис и поддержка. Но уникальность означает высокую цену и недоступность. Сегодня ситуация резко изменилась. Вычислительный кластер можно собрать в любой лаборатории, отталкиваясь от реальных потребностей в вычислительной мощности и доступного бюджета. Для класса задач, где не предполагается тесного взаимодействия между параллельными процессами, кластерное решение даже на основе обычных персональных компьютеров и сети Fast Ethernet будет достаточно эффективным. А чем такое решение принципиально отличается от локальной сети современного предприятия? С точки зрения прикладного программиста — почти ничем. Если у него появится возможность использования подобной сети для решения своих задач, то для него такая конфигурация и будет параллельным компьютером.

Одним из ключевых факторов, определяющих развитие технологий параллельной и распределенной обработки данных, является распространение сетевых технологий и сети Интернет. Интернет можно рассматривать как самый большой параллельный компьютер, состоящий из множества компьютеров сети. Своего рода метакомпьютер, к которому всегда был и будет особый интерес, поскольку никакая отдельная вычислительная система не сравнится по своей мощности с потенциальными возможностями Глобальной сети. Главное — это научиться эффективно использовать ее потенциал.

Конструктивные идеи использования распределенных вычислительных ресурсов для решения сложных задач появились относительно недавно. Первые прототипы реальных систем метакомпьютинга стали доступными с середины 90-х годов. Некоторые претендовали на универсальность, часть систем была сразу ориентирована на решение конкретных задач, где-то ставка делалась на использование выделенных высокопроизводительных сетей и специальных сетевых протоколов, а где-то за основу брались обычные каналы и протокол HTTP.

В настоящее время в мире ведется несколько десятков проектов в области метакомпьютинга. Проект SETI@home для решения задачи поиска внеземных цивилизаций объединил миллионы компьютеров по всему миру для распределенной обработки поступающих с радиотелескопа данных. Проект GIMPS, направлен на поиск простых чисел Мерсена, т.е. чисел вида $2^P - 1$, где P простое число. В ноябре 2003 г. в рамках данного проекта было найдено максимальное на то время число Мерсена $2^{20\,996\,011} - 1$, содержащее в своей записи более 6 млн. цифр. Финансируемый Европейской Комиссией проект EuroGrid направлен на использование возможностей распределенной обработки данных в задачах математического моделирования, молекулярной биологии, предсказания погоды и ядерной физики. В России можно отметить проект MD@home, объединивший несколько десятков организаций для решения задач молекулярного моделирования. В 2003 г. был открыт проект по созданию информационно-вычислительной сети министерства атомной энергии Российской Федерации на основе технологий GRID.

Проанализировав работы в данной области, можно констатировать, что реальная работа по созданию и апробации систем метакомпьютинга сегодня активно идет по трем направлениям [1].

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, 119992, Москва; e-mail: voevodin@parallel.ru

Первое направление — это создание универсальных метакомпьютерных сред. Практически все основные производители программного обеспечения (Oracle, IBM, HP, Sun и др.) работают в данном направлении. Многие берут в качестве стандарта Globus, создавая программную инфраструктуру для своих платформ. На основе этого же пакета формируются глобальные полигоны, объединяющие высокоскоростными сетями значительные распределенные вычислительные ресурсы. Второе направление получается из первого, если универсальность среды заменить четкой ориентацией на конкретные задачи. Речь идет о создании специализированных метакомпьютерных сред для решения небольшого набора многократно используемых "тяжелых" вычислительных задач. Такая постановка намного более реалистична, поскольку специфика задачи известна заранее, что помогает спроектировать эффективную среду для ее решения. Третье направление состоит в разработке инструментария для организации распределенных вычислительных экспериментов.

Безусловно, универсальные среды являются перспективным направлением, но такие среды появятся не скоро. В настоящее время не ясны даже принципы их использования. Globus Toolkit, стандарт дефакто, но он слишком тяжел в установке и сложен в использовании. А что делать, если 2000 компьютеров организации Вам могут отдать лишь на ночь или на два выходных дня? А если администраторы не хотят устанавливать ничего лишнего на свои компьютеры? Нужен простой инструментарий, который помог бы быстро создавать распределенные приложения и использовать доступные вычислительные ресурсы. По такому пути мы и пошли несколько лет назад, отрабатывая различные технологии организации и проведения распределенных вычислительных экспериментов.

Основная цель данной работы — дать краткое описание архитектуры и основных возможностей системы X-Com, созданной в Научно-исследовательском вычислительном центре Московского государственного университета (НИВЦ МГУ).

2. Основные компоненты системы X-Com. Система X-Com реализована в соответствии с принципами клиент-серверной архитектуры, в которой можно выделить два основных компонента: сервер X-Com и узлы (рис. 1). Сервер X-Com — это центральная часть системы, отвечающая за разделение исходной задачи на блоки, распределение заданий, координацию работ всех узлов, контроль целостности результата, сбор результата расчета в единое целое. Узел — любая вычислительная единица (рабочая станция, узел кластера, виртуальная машина), на которой происходит основной расчет прикладной программы. Блоки входных данных задачи передаются от сервера на узлы, где происходит расчет, результаты которого передаются обратно на сервер. Узлы отвечают за расчет блоков прикладной задачи, запрос заданий для расчета от сервера, передачу результатов расчета на сервер.

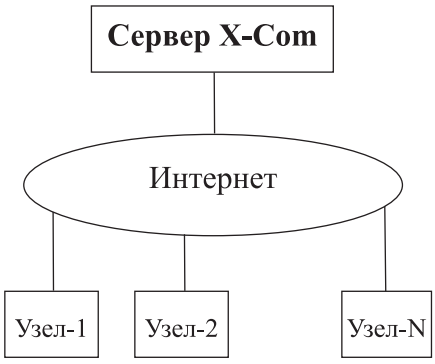

Рис. 1. Основные компоненты системы X-Com

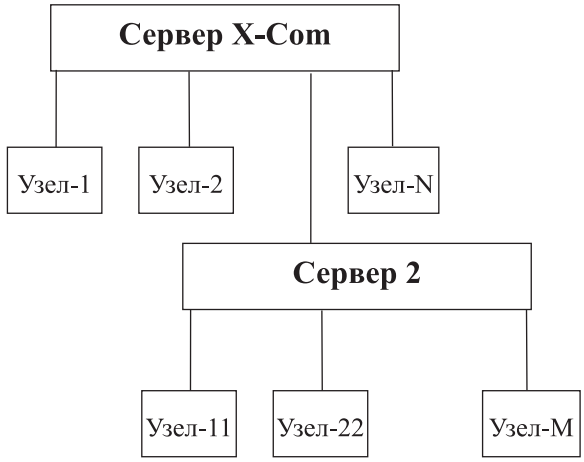

Рис. 2. Иерархия узлов и серверов X-Com

Все коммуникации между узлами и сервером в рамках X-Com происходят через сеть Интернет. При этом используется только стандартный протокол HTTP, что позволяет подключать к системе практически любые вычислительные мощности, доступные в Сети. Система не требует настройки для работы через прокси-сервера, firewall и другие системы защиты, а передаваемые системой данные аналогичны стандартному трафику Интернет.

3. Иерархия узлов и серверов. Для организации более эффективных коммуникаций и подключения большего количества узлов система может быть организована в виде произвольного дерева (рис. 2).

Листьями такого дерева всегда являются вычислительные узлы, а в корне дерева расположен центральный сервер X-Com. Внутренние узлы дерева соответствуют промежуточным серверам, играющим роль своего рода буфера.

Промежуточные серверы (например, Сервер-2 на рис. 2) целесообразно вводить, когда коммуникационный канал между центральным сервером и сервером второго уровня нестабилен или обладает малой пропускной способностью, а связь между промежуточным сервером и его узлами стабильная и поддерживает достаточно высокую скорость обмена данными. В этом случае промежуточный сервер буферизует как некоторую порцию заданий для нижележащих узлов, так и результаты их вычислений. Более того, без такого сервера разрыв канала на время, сопоставимое со временем расчета задачи на конечных узлах, может привести к потере результатов вычислений и необходимости повторной обработки отдельных заданий.

Заметим, что при такой организации полная архитектура системы не содержится и не управляется ни одним из компонентов системы. С точки зрения центрального сервера X-Com любой нижележащий промежуточный сервер выглядит как обычный узел, мощность которого в несколько раз превосходит мощность обычных узлов, а с точки зрения вычислительных узлов — как центральный сервер.

4. Архитектура серверов и вычислительных узлов X-Com. Система X-Com построена по модульному принципу и включает в себя несколько блоков, из которых состоят основные компоненты системы: центральный сервер, промежуточные серверы и вычислительные узлы (рис. 3).

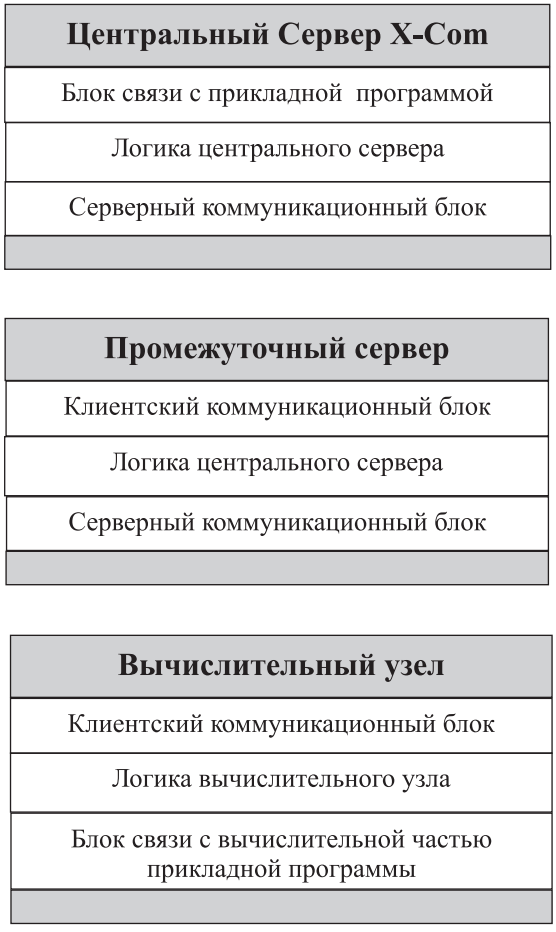

Рис. 3. Основные компоненты системы X-Com и составляющие их модули

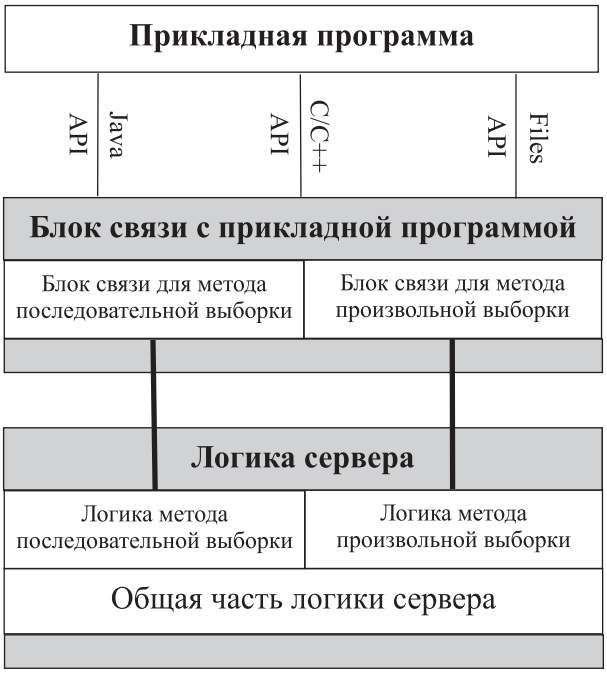

Рис. 4. Реализация методов последовательной и произвольной выборки

Блок связи с прикладной программой представляет собой набор интерфейсов для взаимодействия с серверной частью программы пользователя. В настоящей реализации система поддерживает три интерфейса связи с серверной частью прикладной программы:

— Java API: для программ на языке Java;

— C и C++ API: для программ на языках C и C++. Этим же интерфейсом можно пользоваться для линковки с любыми другими языками, поддерживающими объектные файлы;

— Files API: простой интерфейс, где для взаимодействия с прикладной программой используются файлы, расположенные в файловой системе сервера.

Блок логики центрального сервера определяет логику распределения заданий по узлам, а также алгоритмы сбора и анализа результатов вычислений. Этот блок сопровождает и использует необходимую статистическую информацию (информацию о статистике соединений с каждым узлом, вычислительных возможностях узлов и т.п.). Блок логики центрального сервера строится по принципу стека интерфейсов, который последовательно переводит логику пакетов от прикладной задачи к серверному коммуникационному блоку.

Серверный коммуникационный блок отвечает за пересылку пакета с заданием на узлы и прием от узлов результатов вычислений (под заданием здесь и далее мы будем понимать порцию входных данных, подлежащую обработке). Подключение к узлам происходит по протоколу HTTP. Сервер X-Com имеет два основных режима взаимодействия, которые реализуются в серверном коммуникационном блоке: синхронный и асинхронный. Синхронный режим означает, что на все время вычислений узел имеет доступ к серверу и статистика о решении задачи передается в режиме online. Асинхронный режим означает периодический контакт с вычислительным узлом, и вырожденным случаем данного режима является выдача задания клиенту и ожидание от него соединения с информацией об окончании расчета.

Клиентский коммуникационный блок находится либо на конечном вычислительном узле, либо на промежуточном вычислительном сервере. Основные функции клиентского коммуникационного блока — это прием пакета с заданием на клиентском компьютере и передача результатов расчетов.

Блок логики промежуточного сервера реализует функции буферизации, необходимые для работы промежуточного сервера. Напомним, что с точки зрения центрального севера промежуточный сервер выглядит как обычный вычислительный узел с большой мощностью, что позволяет ему запрашивать большие пакеты вычислительных заданий. Эти блоки заданий буферизуются на промежуточном сервере и распределяются между подключенными к нему узлами. Результаты вычислений также буферизуются на промежуточном сервере и передаются на центральный сервер пакетом.

Блок логики вычислительного узла находится на конечных вычислительных узлах системы. Его основная функция состоит в управлении вычислительной частью прикладной программы пользователя и логикой ее работы (контролю версий, реализации разных режимов работы и т.д.) В задачи данного блока входит сбор информации о типе узла, на котором происходит расчет. Эта информация используется при распределении заданий на сервере X-Com, а также для загрузки "правильной" версии прикладной программы на вычислительный узел.

В общих чертах алгоритм работы вычислительного узла выглядит следующим образом.

1. Сразу после своего запуска узел переходит в режим периодических запросов к серверу. Период запросов варьируется с помощью генератора случайных чисел, чтобы не создавать излишних пиков нагрузки на сервер.

2. После любого соединения с сервером проверяется наличие на узле правильной версии вычислительной части прикладной программы. Если ее нет либо версия не соответствует заданию сервера X-Com, то узел посылает ему соответствующий запрос.

3. Запрашивается очередная порция данных для расчета, которая передается загруженной ранее вычислительной части прикладной программы; после получения результата он передается серверу и цикл повторяется снова. Если по какой-либо причине сервер оказался недоступным, то клиент входит в режим периодических запросов.

Блок связи с вычислительной частью прикладной программы отвечает за взаимодействие с прикладной программой пользователя. В текущей реализации системы предусмотрено три интерфейса связи с вычислительной частью прикладной программы:

— C и C++ API: для программ на языках C и C++. Этим же интерфейсом можно пользоваться для линковки с любыми другими языками, поддерживающими объектные файлы;

— STDIN-STDOUT API: интерфейс, использующий стандартные потоки ввода-вывода;

— Files API: интерфейс, где для взаимодействия с вычислительной частью прикладной программы используются файлы, расположенные в файловой системе узла.

5. Два метода разбиения исходной задачи на блоки. В системе X-Com предусмотрено два основных метода разбиения входных данных решаемой задачи на блоки: методы *последовательной* и *произвольной* выборки. Выбор того или иного метода производится исходя из специфики прикладной программы. По существу, эти методы отличаются лишь способом получения от прикладной программы

очередного задания и возврата полученного результата.

В методе последовательной выборки пакеты заданий поступают от прикладной программы по порядку, один за другим, и в той же последовательности ей возвращаются результаты вычислений. При запросе очередного задания прикладная программа выдает идентификатор пакета и порцию данных для расчета, каждое задание выдается строго последовательно и один раз. При получении результатов расчета прикладной программе выдается идентификатор пакета (полученный вместе с заданием) и результат расчета. При этом гарантируется, что если пакет **A** был выдан раньше пакета **B**, то и результаты будут получены в том же порядке. Сквозной нумерации пакетов заданий нет.

Для организации такого интерфейса приходится поддерживать окно заданий, куда попадают все обрабатываемые в данный момент пакеты данных. При обращении очередного клиента за порцией вычислений происходит выбор между уже выданными заданиями из окна (если истекло время ожидания ответа от узла, получившего это задание) и запросом прикладной программы о новом задании.

Метод последовательной выборки удобен с точки зрения прикладной программы, но требует больших накладных расходов по памяти на поддержку окна, особенно при большом размере исходных и результирующих пакетов. Этот метод следует использовать, если того требует специфика прикладной задачи (например, результаты предыдущих расчетов используются при выдаче очередного пакета).

В методе произвольной выборки прикладная программа должна обеспечивать сквозную нумерацию заданий и быть готова в любой момент выдать задание с номером **N** из диапазона прикладной задачи. Результаты вычислений также могут приходить в произвольном порядке, но гарантируется, что все результаты будут получены, причем только один раз. В этом методе не требуется хранить в окне заданий сам запрос и результат вычислений, хранятся только номера пакетов, которые в данный момент находятся в процессе обработки. Если возникает необходимость послать пакет другому вычислительному узлу (например, при истечении времени ожидания), то прикладная программа запрашивается еще раз и заново выдает задание. Полученный от вычислительного узла результат расчета сразу же передается прикладной программе.

Для реализации двух методов разбиения исходной задачи блок связи с прикладной программой, как и блок логики сервера, разбит на два независимых блока (рис. 4). Поскольку прикладная программа взаимодействует с сервером X-Com через один из упомянутых выше трех интерфейсов, то в каждом из них предусмотрены независимые наборы функций как для одного, так и для другого метода.

6. Прикладная программа в системе X-Com. Прикладная программа в системе X-Com разбивается на две части: серверную и вычислительную (рис. 5). *Серверная часть* прикладной программы управляет формированием заданий для расчета на узлах. Программа может быть реализована с помощью любой технологии программирования, с использованием API X-Com.

Вычислительная часть прикладной программы представляет собой основной расчетный модуль, который также может быть реализован с помощью любой технологии программирования (возможно и отличной от технологии серверной части прикладной программы) с соблюдением API X-Com.

Отметим, что интерфейсы вычислительной части прикладной программы никак не связаны с интерфейсами ее серверной части. Более того, они не используют методы последовательной и произвольной выборки, поскольку метод разбиения исходной задачи на блоки никак не влияет на расчет каждого блока.

7. Ход вычислений в системе X-Com. Для того чтобы лучше понять работу системы и ее отдельных компонент, рассмотрим ход вычислений на некоторой задаче. Для простоты предположим, что задача допускает решение методом прямой выборки и используется базовая архитектура с одним центральным сервером и несколькими узлами (без промежуточных серверов). Прикладная программа разбивается на два блока: серверный и клиентский. Серверный блок отвечает за выдачу заданий и по запросу выдает либо очередную порцию данных, либо сигнал о том, что все задания выданы. Клиентский блок проводит расчет задания и выдает результат.

Исходная задача разбивается на блоки. Поскольку мы предположили, что применяется метод прямой выборки, то количество блоков известно и все блоки можно пронумеровать от 1 до N. В дальнейшем мы будем оперировать этими номерами для идентификации вычислительного блока.

В начальный момент времени сервер находится в состоянии ожидания запросов от вычислительных узлов. Пусть инициировано некоторое количество вычислительных узлов. Каждый узел все время своего функционирования периодически посылает запросы “дай задание” на сервер. Если произошел сбой соединения либо сервер еще не готов, то узел ожидает некоторое время и повторяет попытку.

Предположим, что некоторый вычислительный узел соединился с сервером. Первым о соединении узнает серверный коммуникационный блок. Он не реализует никакой логики, просто принимает запрос, разбирает его заголовки и передает данные запроса в блок логики сервера.

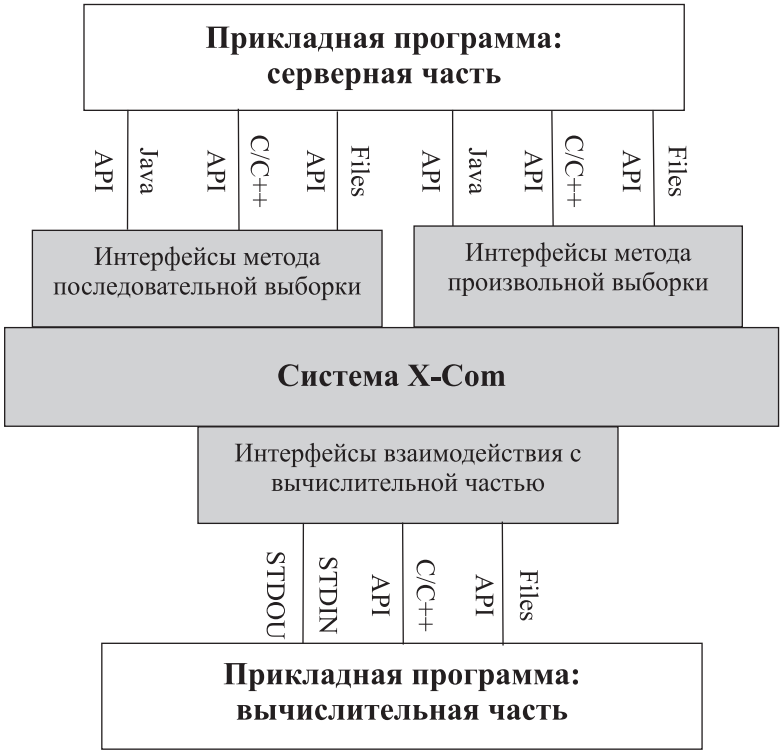

Рис. 5. Интерфейсы взаимодействия прикладной программы и системы X-Com

Возможны пять типов соединений.

1. “Дай версию расчетной программы для моей программно-аппаратной платформы”. В данном запросе узел указывает код своей программно-аппаратной платформы и сервер передает на узел соответствующую расчетную часть прикладной программы.

2. “Дай задание” — первичный запрос задания.

3. “Получи результат и дай следующее задание” — возврат выполненного задания и запрос следующего. Этот запрос делается в одной сессии для оптимизации сетевого взаимодействия.

4. “Получи результат” — возврат выполненного задания и сообщение о завершении работы. Этот тип соединения используется, когда на сервере нужно корректно отслеживать статус вычислительных узлов.

5. Сообщение о статусе расчета на данном узле. Этот тип запроса важен при длительных вычислениях, чтобы сервер “помнил” о существовании узла. Такие запросы передаются напрямую в блок сбора статистики о ходе вычислений и прямо не влияют на ход вычислений.

Любой подключившийся узел независимо от типа соединения проходит процедуру идентификации: либо находится соответствующая ему запись в таблице узлов, либо, если это первое обращение данного узла, заводится новая запись.

После подключения и идентификации проверяется наличие на узле правильной версии вычислительной части прикладной программы. Если ее нет либо версия не соответствует выполняющейся задаче, то нужная версия запрашивается у сервера. При этом загружается та версия прикладной программы, которая соответствует программно-аппаратной платформе узла. Если на сервере X-Com такой версии нет, то узел переходит в режим периодических запросов к серверу: нужная версия появится либо во время эксперимента, либо на сервере будет запущена другая программа.

После загрузки вычислительной части прикладной программы запускается проверка корректности среды выполнения. Это опциональная часть прикладной программы, которая проверяет наличие на узле необходимых для расчета ресурсов, проводит дополнительный контроль целостности загруженной версии и возможности расчета на данном узле. После проверки корректности узел переходит в режим запросов заданий.

Предположим, что на сервер X-Com от некоторого узла поступает запрос “дай задание”. Блок логики сервера передал этот запрос через API прикладной программе, которая вернула порцию данных,

подлежащую обработке. Серверная часть прикладной программы не знает, на каком узле будет производиться расчет, она просто выдает очередной блок данных по номеру, который генерирует блок логики сервера. Затем задание на расчет через блок логики, серверный коммуникационный блок и клиентский коммуникационный блок попадает на узел.

Получив задание, узел запускает прикладную программу для расчета. По ходу расчета фоновый процесс блока логики узла периодически посылает информацию, что узел находится в состоянии вычисления полученной порции. Окончив расчет, узел посылает результат расчета на сервер; для этого используется запрос "Получи результат и дай следующее задание". Пройдя серверный коммуникационный блок, процедуру идентификации и проверку корректности, содержащий результат вычислений запрос передается прикладной программе. После этого происходит запрос новой порции вычислений, что с точки зрения прикладной программы полностью аналогично первичному запросу задания.

В некоторый момент времени все задания для расчета будут выданы. В такой ситуации очередной узел при возврате результата своего расчета получит ответ от сервера "Нет заданий", после чего этот узел отключится и перейдет в режим периодических запросов сервера. По мере окончания вычислений все узлы вернут результаты порученных им расчетов и сервер зафиксирует окончание вычислений.

8. Перераспределение заданий. Предположим, что серверу требуется выдать очередное задание для обработки. У него всегда есть выбор: запросить у пользовательской программы новую порцию данных либо повторно выдать задание узлу, который считается выбывшим из вычислений. В системе предусмотрены два основных сценария распределения заданий.

1. Все задания распределяются по очереди. В тот момент, когда все задания выданы, очередному узлу, обратившемуся с запросом "дай задание", сервер выдает то задание, которое было выдано раньше остальных, но результат которого еще не был получен. Этот сценарий можно применять только в методе произвольной выборки; в общем случае он позволяет закончить все вычисления быстрее.

2. Задание, переданное на узел, который признан выбывшим из вычислений, передается другому узлу до полного распределения заданий. Отметим, что в данном сценарии возможна ситуация, когда вычисления еще не закончены, но очередной узел, обратившись за заданием, не получает ничего.

В обоих сценариях возможна повторная обработка задания. Это может произойти при временном разрыве связи либо из-за повышенной загрузки вычислительных узлов другими задачами.

Во втором сценарии при необходимости выдачи нового задания сервер вначале просматривает список уже инициированных заданий: если время ожидания какого-либо из них истекло, то выдает его. Если все времена заданий, которые сейчас находятся в обработке, еще не истекли, то об очередном задании запрашивается прикладная программа.

Время ожидания ответа рассчитывается как расчетное время ожидания ответа, умноженное на коэффициент K_wait. Расчетное время ожидания ответа, в свою очередь, определяется исходя из статистики работы с каждым конкретным клиентом и размера текущего задания. Коэффициент K_wait — важный атрибут работы системы: при его увеличении мы ожидаем пакет в течение большего времени и соответственно меньше шанс, что мы пошлем один и тот же пакет двум разным вычислительным узлам (задержка может быть связана как с неточностью предсказания времени расчета, так и с загрузкой вычислительного узла). С другой стороны, при увеличении этого коэффициента мы увеличиваем размеры окна, что ведет к увеличению времени ожидания очередной порции данных и требует дополнительных ресурсов памяти центрального сервера на поддержку увеличения окна.

9. Фоновые процессы на сервере. Параллельно с основными процессами на сервере работают два фоновых процесса, которые не влияют на ход вычислений: проверка состояния узлов и отображение хода вычислений. Проверка состояний узлов — это процесс, периодически проверяющий все узлы из таблицы текущих вычислений. Второй фоновый процесс отвечает за отображение хода вычислений. Данный процесс периодически сбрасывает на жесткий диск доступный через Веб-сервер файл, в котором отображается полная таблица подключенных узлов и статистическая информация о ходе вычислений: количество активных узлов, количество узлов, находящихся в процессе взаимодействия с сервером, суммарная вычислительная мощность системы и ряд других параметров.

10. Проверка корректности результата. При получении сервером результатов расчета в ряде случаев возникает необходимость в проверке их правильности. Это необходимо, чтобы избежать умышленного искажения результата либо искажения результатов расчетов из-за некорректно работающего узла. Есть четыре основных способа проверки корректности.

Прямая проверка корректности серверной частью прикладной программы. Прикладная программа запрашивается с уровня проверки корректности парой: запрос + результат, на что должна выдать ответ: верный результат либо нет. При неверном результате информация об ошибке сохраняется в специальных

логах для дальнейшей разборки, почему это произошло. В штатном режиме функционирования системы ошибок быть не должно, поэтому номер узла, выдавшего ошибку, сохраняется в черном списке и соединения от него больше не принимаются.

Прямая проверка корректности другим узлом. В этом случае в начало общего списка заданий ставится задание на проверку и решается обратная задача, где исходными данными служит только что полученный результат вычислений. Результат этого расчета должен совпасть с исходными данными (проверка на идентичность производится в прикладной программе), в противном случае исходное задание будет обработано повторно, а номера узлов, занимавшихся проверкой и перепроверкой, заносятся в черный список.

Метод многократного перерасчета. Ключевым параметром метода служит коэффициент перепроверки — вещественное число, большее единицы. В случае целого числа этот коэффициент показывает, скольким узлам надо выдать одно и то же задание. Полученные результаты сверяются друг с другом; в случае расхождения используется метод голосования для определения правильного результата. Если равное количество узлов проголосовало за каждый вариант результата, проводятся дополнительные проверки для определения победителя. Все узлы, которые выдали неверный результат, заносятся в черный список и в дальнейшем не смогут получать задания на обработку. Если коэффициент не является целым числом, то по нему определяется вероятность, с которой очередной пакет будет проверен.

Отсутствие проверки корректности. Результат вычислений будет передаваться для сохранения в прикладную программу без проверки корректности.

11. Эксперименты с системой X-Com. Совместно с группой специалистов из Центра "Биоинженерия" РАН в распределенном режиме решалась задача определения скрытой периодичности в генетических последовательностях. Основную сложность в данном случае представляли высокая вычислительная сложность алгоритмов и огромный объем входных данных. По самым скромным подсчетам для обработки материала, имеющего реальное научное значение, на одном процессоре потребовалось бы обработать несколько гигабайт входной информации и многие месяцы, а то и годы, непрерывных вычислений.

Наряду с решением основной прикладной задачи преследовались две дополнительные цели: проверить потенциальные возможности X-Com и определить особенности ее работы при пиковых нагрузках. В данной работе остановимся на кратком описании лишь двух из наиболее интересных экспериментов, более полное описание можно найти, например, в [2, 3].

Целью первого эксперимента был географический размах. Было задействовано 14 удаленных вычислительных систем, принадлежащих 10 организациям, расположенным в 8 городах и 6 различных часовых поясах. В разные моменты данного промежутка времени к эксперименту подключалось 407 процессоров, причем максимальное число одновременно работающих процессоров составило 385, а в среднем по эксперименту эта величина оказалась равной 296. Общее время эксперимента составило около 64 часов. Один средний компьютер справился бы с подобной задачей лишь за два года непрерывной работы.

Задача второго эксперимента — это работа в условиях максимальной неоднородности. Организаторами эксперимента были НИВЦ МГУ и Межведомственный суперкомпьютерный центр. В расчете участвовали четыре типа вычислительных ресурсов: отдельные компьютеры, учебный класс, традиционные вычислительные кластеры и суперкомпьютер МВС-1000М. Часть компьютеров работала под управлением MS Windows, а часть — под Linux (как правило, Red Hat). На одной части компьютеров использовались процессоры Intel, а на другой — Alpha. Все ресурсы использовались в трех различных режимах. Эксперимент продолжался 14,5 часов. Среднее число одновременно работающих процессоров составило 125, максимум зафиксирован на значении 150, а всего к эксперименту на разных этапах подключалось 437 различных процессоров. Казалось бы слишком много, учитывая среднее число 125. Но объясняется это очень просто. Суперкомпьютер МВС-1000М объединяет 768 процессоров, а система управления заданиями при распределении программы на реальные процессоры не обязана учитывать, куда были переданы предыдущие варианты этой же программы. Суммарно за время эксперимента было передано 26,8 Гбайт данных. Время обработки одной порции данных менялось от 1 секунды до 10 минут. Несмотря на значительную коммуникационную нагрузку и специально созданную неоднородность, эффективность работы всей системы составила 92,9 %.

12. Заключение. В ходе проведения экспериментов система X-Com подтвердила те качества, которые мы закладывали при ее проектировании. Одно из уникальных качеств X-Com — это простота настройки и использования системы, а также быстрая адаптация прикладных программ. Для работы X-Com не требуется специализированных компьютеров и высокоскоростных сетей. Настройка вычислительных ресурсов не требует специальных навыков и особых прав доступа. Благодаря этим свойствам системы удалось убедительно продемонстрировать возможность решения больших реальных задач с ис-

пользованием уже существующих распределенных вычислительных ресурсов. В настоящее время система проходит опытную эксплуатацию, запланировано ее дальнейшее развитие по целому ряду направлений.

Данная работа выполнена при поддержке РФФИ, грант № 02–07–90442.

СПИСОК ЛИТЕРАТУРЫ

1. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб: БХВ-Петербург, 2002.
2. *Воеводин Вл.В., Филамофитский М.П.* Суперкомпьютер на выходные // Открытые системы. 2003. № 5. 43–48.
3. *Воеводин В.В., Смирнов Ю.Г., Филамофитский М.П., Цупак А.А.* Решение задачи дифракции на диэлектрическом теле методом объемных интегральных уравнений на многопроцессорных системах // Актуальные проблемы науки и образования. Труды Международного юбилейного симпозиума. Пенза: Информационно-издательский центр ПГУ, 2003. 5–8.

Поступила в редакцию
06.04.2004