Эволюция системы метакомпьютинга X-Com*

Вл.В. Воеводин, Ю.А. Жолудев, С.И. Соболев, К.С. Стефанов

Статья посвящена текущему состоянию системы метакомпьтинга X-Com, разработанной в НИВЦ МГУ. С учетом опыта практического использования системы была существенно переработана ее архитектура и изменена технологическая основа. В новой версии системы X-Com сделан особый акцент на масштабируемости, поддержке распределенных сред с суперкомпьютерным уровнем производительности. Реализованы механизмы буферизующих серверов, средства синхронизации файлов и сетевой безопасности.

1. Введение

Система метакомпьютинга X-Com [1], разработанная в НИВЦ МГУ, представляет собой инструментарий низкого уровня, предназначенный для организации распределенных вычислительных сред, а также для адаптации и поддержки выполнения прикладных задач в таких средах. Основными требованиями к системе, исходя из которых велась ее разработка, была легкость установки и развертывания программного обеспечения, поддержка различных программно-аппаратных платформ, простота адаптации прикладных программ для работы в распределенных средах, отсутствие необходимости в административном доступе к подключаемым вы-

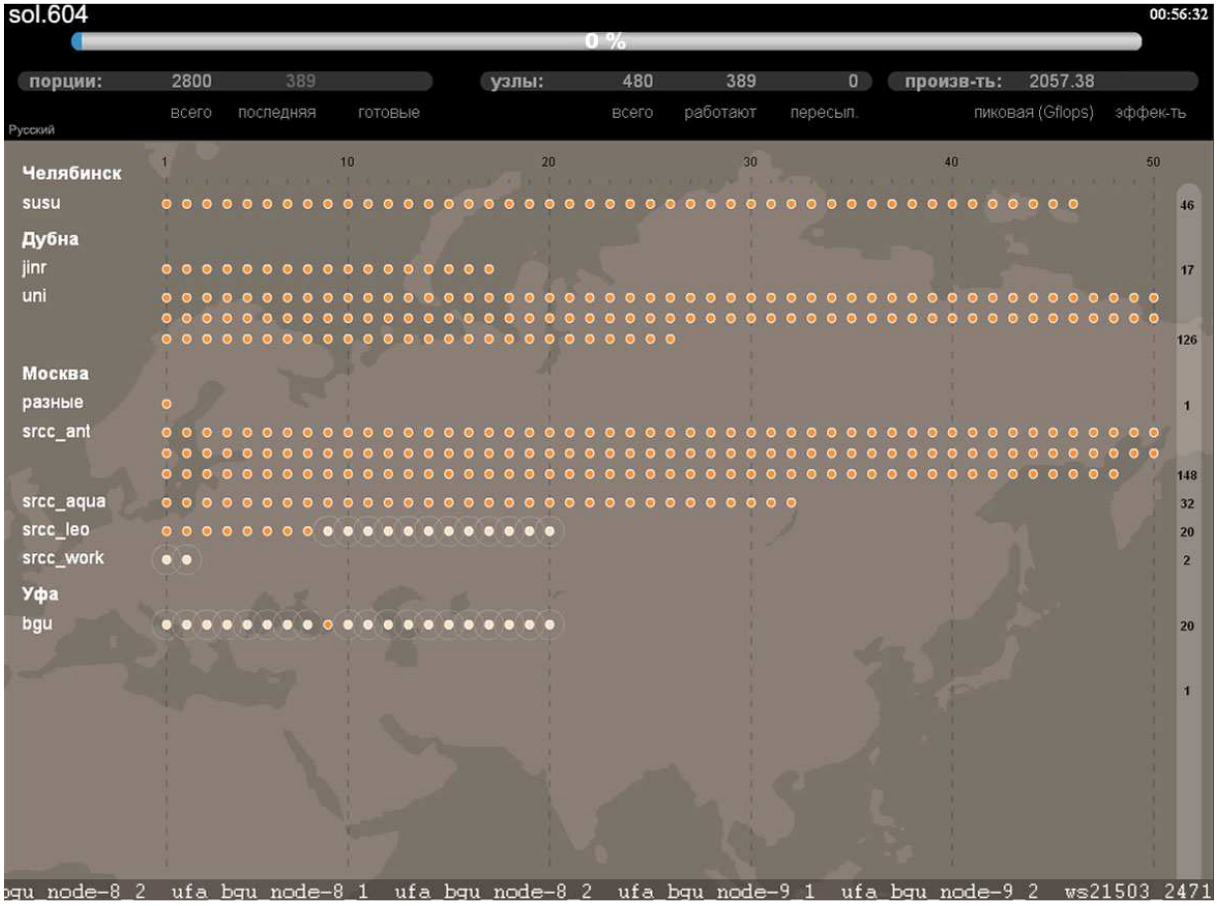

Рис. 1. Пример визуализации распределенного эксперимента – подключение к реальному расчету (виртуальный докинг молекул, 2006 г., [5]) узлов вычислительных кластеров из различных городов

* Работа выполнена при поддержке ФЦП "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2012 годы", научно-технической программы Союзного государства СКИФ-ГРИД.

числительным ресурсам. Полнофункциональная версия системы X-Com первого поколения появилась в 2002 г. С учетом приведенных требований, аналогичных по функциональности программных продуктов на тот момент фактически не было. Так, инструментарий Globus Toolkit [2] уже в те времена отличался сложностью в настройке, установке и использовании. Система Condor [3] исходила из иных принципов, обеспечивая распределение потоков относительно небольших задач на доступные ресурсы. Система BOINC [4], лежащая в основе многих известных Интернет-проектов по распределенным вычислениям, еще находилась в стадии разработки.

Развитие и использование системы X-Com насчитывает уже более 7 лет. За это время система многократно использовалась для решения вычислительно сложных и практически важных задач из области молекулярного моделирования [5], электромагнитной динамики [6], биоинженерии и других. В составе системы X-Com появились модули генерации статистики по проведенным экспериментам и визуализации хода расчетов (Рис. 1), подсистема управления заданиями, наборы сценариев для установки и запуска клиентской части X-Com в различных режимах (стоит отметить, что при этом ядро системы практически не менялось). В ходе практической работы с системой, а также в результате общения с другими пользователями X-Com был получен огромный опыт, который позволил тщательно проанализировать систему в целом – ее программную архитектуру, используемые алгоритмы, методы взаимодействия между компонентами и т.д. Проведенный анализ позволил определить как сильные, так и слабые стороны системы.

Кратко перечислим "слабые" стороны системы X-Com на основе ядра первого поколения (заметим, что некоторые из них не являются недостатками в прямом смысле этого слова – ряд аспектов при разработке сознательно не учитывались, актуальность других на момент создания ядра была непонятна). Так, изначально система X-Com была ориентирована на запуск и работу только одного распределенного приложения в каждый момент времени. Это соответствует основой идее метакомпьютинга – подключение к расчетам всех доступных ресурсов. Однако такой режим работы не всегда является оптимальным с точки зрения эффективного использования ресурсов. Да и пользователям зачастую необходимо запустить не одну, а одновременно несколько задач. Возможность работы с несколькими распределенными задачами одновременно уже появилась в последней версии подсистемы управления заданиями, однако реализована она была несколько искусственно.

Далее, в системе X-Com не было реализовано никаких средств безопасности. В начале разработки на это шли сознательно, ставя во главу угла возможность как можно более простой и быстрой установки, настройки и запуска системы. Однако в настоящее время проблема безопасности становится все более актуальной. Подключая к расчетам мощные вычислительные установки и отдельные рабочие станции, необходимо гарантировать их владельцам, что их ресурсам (или же с помощью их ресурсов) не будет нанесено никакого вреда, а сам расчет будет проведен корректно.

Отсутствие контроля над клиентами. В версии X-Com первого поколения сервер, отправив задание на узел, ничего не знает о его состоянии до тех пор, пока узел не вернет результат. Работа по такой схеме была продиктована желанием минимизировать обмен данными между узлами и сервером. Но тем самым мы не имеем никакой возможности управлять ходом расчета на узлах. А это может потребоваться, например, в том случае, когда нужно срочно остановить работу на части узлов, т.е. освободить занимаемые ресурсы, или же при желании переключить часть узлов на решение другой задачи.

"Зоопарк" технологий программирования. Ядро системы X-Com, ее серверная часть, была написана на Java, клиентская – на Perl, для сервисных утилит использовался Perl и PHP. Очевидно, что различие используемых технологий программирования затрудняют интеграцию компонентов системы в единый программный комплекс, что приводит к потере свойства простоты установки и использования.

Имелся ряд проблем и с ядром системы. В частности, его программная структура не позволяла эффективно построить механизм промежуточных (буферизующих) серверов. Имеющаяся реализация промежуточных серверов выполняла свои функции лишь формально и не использовалась на практике.

Анализ выявленных слабых мест позволил сформулировать ряд требований к дальнейшему развитию системы X-Com, которые легли в основу архитектуры системы нового поколения.

2. Архитектура системы X-Com нового поколения

Безусловно, в новой архитектуре системы требовалось сохранить все ее сильные стороны, а именно оперативность развертывания распределенных вычислительных сред, поддержку различных программно-аппаратных платформ, различные режимы подключения вычислительных узлов, простоту адаптации прикладных задач. Для облегчения перехода пользователей со старой версии системы X-Com хотелось в целом сохранить логику работы и интерфейсы взаимодействия системы и прикладной задачи.

В новой архитектуре системы X-Com был сделан особый акцент на возможностях масштабируемости и поддержки распределенных сред с суперкомпьютерным уровнем производительности. Такой акцент не случаен – за последние несколько лет в России значительно выросло число суперкомпьютеров. Инфраструктура суперкомпьютеров, и в особенности кластерных вычислительных систем, представляет собой практически идеальный вариант для развертывания распределенных сред. Однородная структура узлов суперкомпьютера и возможность управления ими с выделенной (головной) машины позволяет оперативно подключать к распределенным расчетам сегменты терафлопного уровня производительности.

Для повышения уровня масштабируемости X-Com архитектура северной части системы является распределенной, с возможностью разнесения процессов на физически отдельные машины. Кроме того, был реализован полнофункциональный механизм буферизующих серверов, позволяющих как снизить нагрузку на центральный сервер системы, так и оптимизировать сетевые обмены.

В архитектуру заложены возможности по управлению узлами со стороны северной части. Каждый узел периодически может отсылать информацию о своем состоянии, получая в ответ команду на продолжение текущего расчета либо на совершение какого-либо действия, например, на смену задачи. Благодаря такой возможности динамического перераспределения узлов

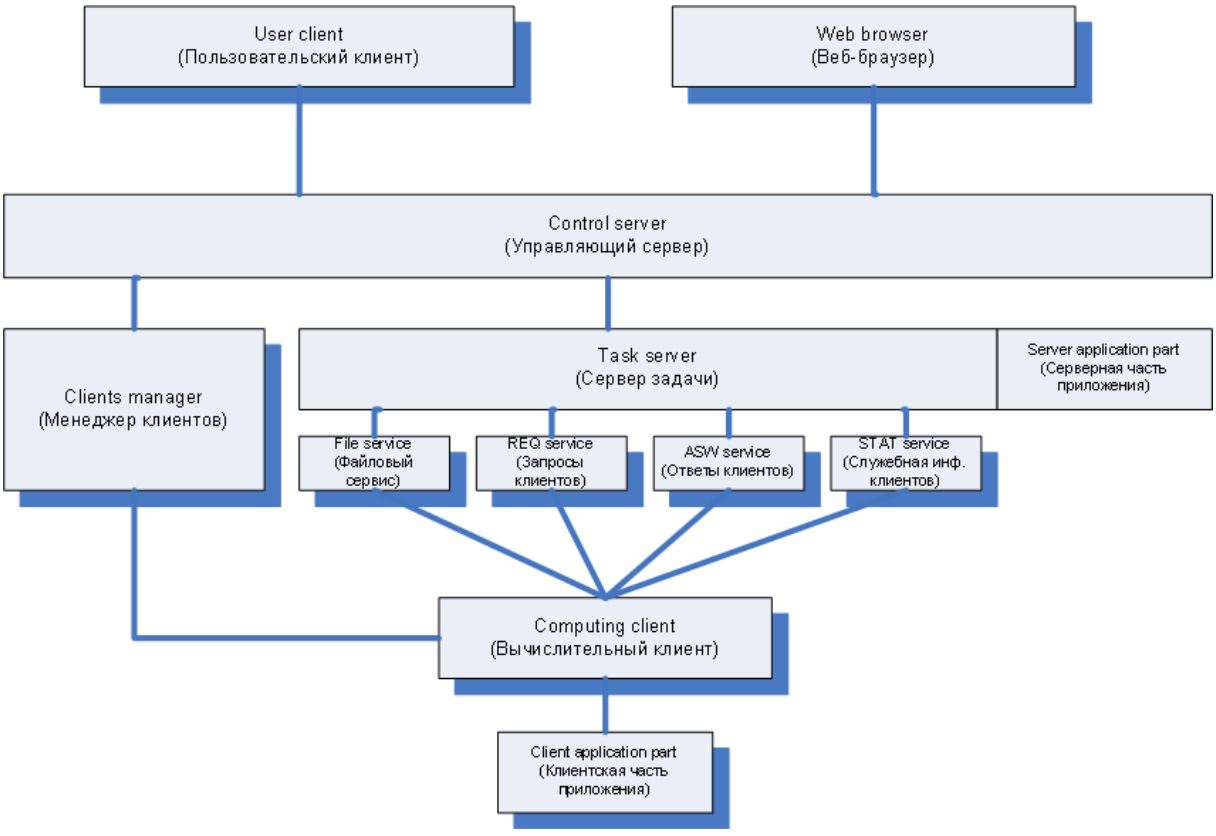

Рис. 2. Архитектура системы X-Com нового поколения

становится возможным реализовать один из вариантов многозадачности в распределенной среде. При появлении очередной задачи среда выделяет часть узлов для ее решения, обеспечивая таким образом одновременное выполнение нескольких задач. Другой режим многозадачности заключается в учете требований прикладной задачи, накладываемых на вычислительные узлы, на которых она может выполняться.

Новая архитектура X-Com предусматривает уровень безопасности с опциональным шифрованием данных, передающихся по открытым сетям, а также сертификацию доверенных клиентов и серверов. Дополнительные возможности включают в себя механизмы передачи файлов между серверной частью X-Com и вычислительными узлами, расширение функциональности клиентской части и другие.

На Рис. 2 приведена архитектура системы X-Com нового поколения. Прямоугольники на рисунке соответствуют процессам системы. Общение между процессами осуществляется по сетям TCP/IP. В архитектуре выделяются три уровня – пользовательский уровень, серверный уровень и уровень вычислительных узлов.

На пользовательском уровне находятся процессы пользовательских клиентов и веб-браузеров. Пользовательский клиент – компонент системы, обеспечивающий интерфейс между пользователями и системой метакомпьютинга. С его помощью пользователь может поставить задачу в очередь на выполнение и отслеживать ее состояние. При постановке задания в очередь пользователь формирует файл описания задания, в котором указываются входные данные прикладной задачи и требования к ее запуску, в том числе требования к ресурсам, на которых задача будет выполняться.

Веб-браузер – обычный интернет-браузер, с помощью которого пользователь может следить за ходом расчетов и состоянием распределенной среды в наглядной форме. При наличии веб-интерфейса к прикладной задаче с помощью браузера также может осуществляться постановка задачи в очередь.

На серверном уровне находятся процессы сервера задачи, управляющего сервера и менеджера клиентов. Сервер задачи – компонент, обеспечивающий выполнение одной конкретной прикладной задачи. Он взаимодействует с серверной частью прикладной задачи, которая генерирует порции данных и осуществляет финальную обработку результатов, приходящих от клиентов с вычислительных узлов. Сервер задачи реализует логику выдачи порций клиентам. С клиентами на вычислительных узлах сервер задачи взаимодействует не напрямую, а через процессы-сервисы, обозначенные на рисунке как File service, REQ, ASW и STAT. Файловый сервис – это фактически файловый сервер, у которого клиент запрашивает файлы прикладной задачи и вспомогательные файлы, если они необходимы. Через сервис REQ (request) клиент запрашивает очередную порцию данных у сервера задачи. Результаты выполнения прикладной задачи над данными очередной порции клиент пересылает сервису ASW (answer). Сервис STAT (statistics) обеспечивает обмен контрольной информацией между клиентами и серверной частью. Выделение сервисов REQ, ASW и других отельными процессами обусловлено требованием распределения нагрузки между компонентами системы. Это будет особенно важно при интенсивных обменах данными между клиентами и серверной частью при включенном шифрования (см. далее). В этом случае сервисы берут на себя всю предварительную обработку входящих и исходящих порций. В качестве файлового сервера может применяться любой http-сервер.

Управляющий сервер – компонент, обеспечивающий управление прохождением заданий в распределенной среде, в частности, реализующий логику очередей заданий или же логику одновременного выполнения задач. На каждое запускаемое задание управляющий сервер порождает свой сервер задачи. Управляющий сервер транслирует команды – например, на срочное прекращение расчета – серверу задачи, которые затем передаются клиентам, работающим с данным сервером.

Менеджер клиентов – компонент, к которому все клиенты обращаются при первом запуске. Менеджер клиентов перенаправляет вычислительные узлы к тем серверам задач, требованиям которых они соответствуют. По завершению работы с заданием клиенты вновь обращаются к менеджеру.

На уровне вычислительных узлов функционирует клиент X-Com, который получает от сервера задач очередную порцию данных, запускает клиентскую (вычислительную) часть прикладной задачи и передает результаты вычислений обратно на сервер. Очередную задачу кли-

ент получает через менеджер. В ответах на запросы клиентов, формируемые сервером, может быть предусмотрена смена адресов сервисов, с которыми клиент работал прежде. Таким образом, можно, например, динамически перенаправить клиента для работы с другой задачей.

В качестве единой технологической основы для реализации всех компонентов системы была выбрана технология программирования Perl. Perl доступен для практически всех программно-аппаратных платформ, он пригоден для написания переносимого кода. Perl фактически по умолчанию входит в дистрибутивы Linux, имеются хорошие реализации для платформы MS Windows. Perl также добавляет удобный интерфейс для программирования серверной части прикладных задач.

3. Особенности реализации и функционирования модулей системы X-Com

3.1. Промежуточные серверы

В базовом варианте организации распределенной среды на основе технологии X-Com в ней выделяется единственный сервер X-Com, с которым общаются все подключенные клиенты. Основное достоинство данного способа – единообразие настройки всех клиентов. Он может применяться в достаточно однородной среде с относительно небольшим числом входящих в нее узлов, каждый из которых имеет непосредственный доступ к серверу. Однако зачастую возникает ситуация, при которой часть узлов находится внутри закрытой локальной сети и не имеет прямого доступа к центральному серверу. Такая ситуация типична для кластерных вычислительных комплексов, узлы которых имеют "приватные" IP-адреса, и работа с ними может осуществляться только через головную машину кластера. С другой стороны, большое количество подключенных к центральному серверу узлов могут создать значительную нагрузку на сервер, что может привести к снижению эффективности использования вычислительной среды.

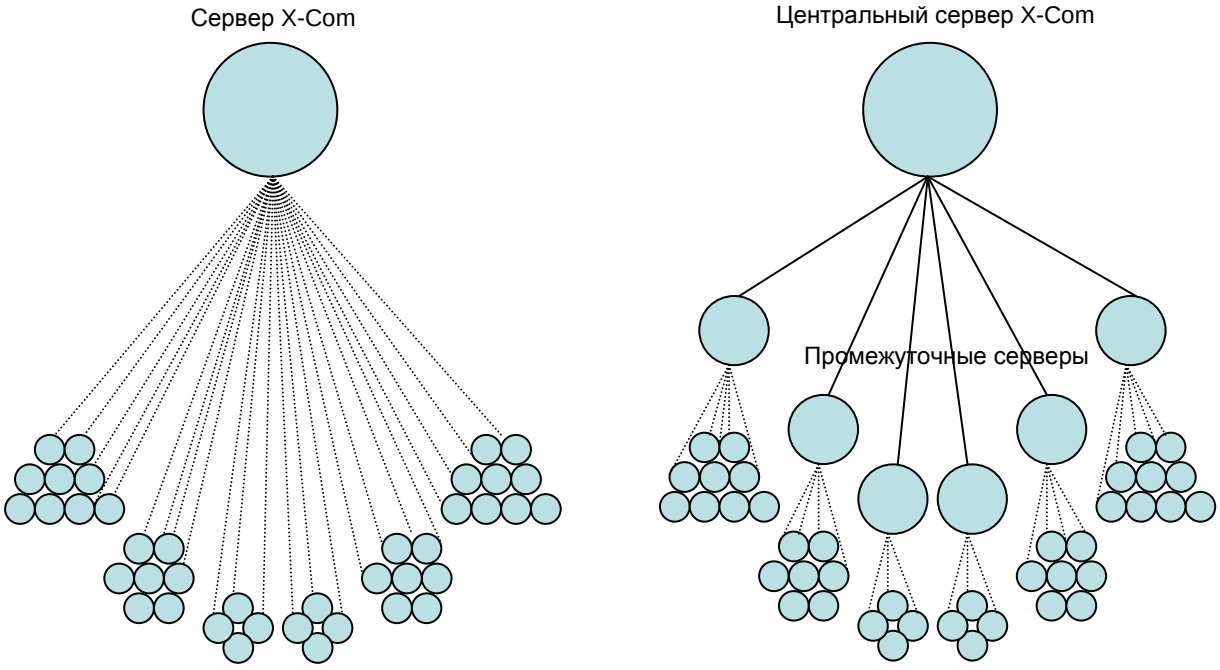

Рис. 3. Организация распределенной среды в базовом варианте (слева) и с использованием промежуточных серверов (справа)

Для решения этих проблем в иерархическую структуру сервер-клиенты вводится еще один класс компонентов – промежуточных серверов, располагающихся между основным сервером и вычислительными узлами. Использование их позволяет организовать топологию распределенной среды в виде произвольного иерархического дерева. Листьями такого дерева всегда будут

вычислительные узлы, в корне дерева остается центральный сервер X-Com, а во внутренних узлах дерева будут располагаться промежуточные серверы X-Com (Рис. 3).

Промежуточные серверы с точки зрения центрального сервера выглядят как обычные вычислительные узлы, а с точки зрения нижележащих вычислительных узлов как центральный сервер X-Com. Полная архитектура системы не доступна ни одному из компонентов системы. Для центрального сервера любой нижележащий промежуточный сервер X-Com представляется узлом, мощность которого в несколько раз превосходит мощность обычных узлов. Такой подход позволяет в любой момент времени подключать и отключать как обычные вычислительные узлы, так и целые кластеры узлов через промежуточный сервер. На результатах вычислений подключение и отключение узлов и промежуточных серверов не отразится, это повлияет только на производительность системы в целом. Одновременно это же свойство помогает обеспечить необходимую степень масштабируемости – поскольку часть клиентов находится за промежуточными серверами, центральному серверу X-Com не требуется поддерживать большое количество сетевых соединений, и таким образом нагрузка на него выравнивается.

Алгоритм работы промежуточного сервера X-Com следующий. Соединившись с центральным сервером, он получает от него P вычислительных порций для нижележащих узлов. Далее, в зависимости от настроек промежуточный сервер либо постоянно поддерживает окно в P порций, делая запрос к центральному серверу сразу же после получения порций конечным узлом, либо всякий раз запрашивает по P порций и дожидается их исчерпания. Результаты, приходящие от нижележащих узлов, могут отправляться центральному серверу сразу же, а могут группироваться по Q порций. При выставлении настроек P = Q = 1 промежуточный сервер не применяет буферизацию, фактически транслируя запросы между центральным сервером и узлами. Такой вариант имеет смысл использовать, например, в случае, если узлы находятся во внутренней закрытой сети, каналы связи стабильны, а условия решения прикладной задачи требуют минимизации задержки между отправкой входных порций данных и получением результатов их обработки.

3.2. Методы работы клиентской часть X-Com

Клиент X-Com отвечает за загрузку клиентской части прикладной задачи и необходимых для работы файлов на узлы, производящие вычисления, формирование рабочего окружения, получение порций данных, запуск задачи и передачу полученных результатов серверу X-Com.

После запуска клиент находится в начальном состоянии, он создает рабочий каталог, производит определение характеристик платформы, на которой он запущен, и посылает серверу запрос на описание задачи (GetTask). В ответ сервер выдает описание задачи (команда ProcessTask). В описании находится информация об URL файлов задачи (исполняемых файлов и файлов данных).

После получения описания задачи клиент приступает к загрузке файлов. Все файлы, указанные в описании задачи, загружаются при помощи метода GET протокола HTTP. Имеется возможность передавать файлы в виде архива tar, упакованного gzip.

После получения всех необходимых файлов клиент производит инициализацию клиентской части прикладной задачи и посылает запрос порции данных для счета (GetPortion). После получения порции данных клиент запускает обработку порции.

После обработки порции данных клиент отсылает результат на сервер (запрос PostResult), после чего получает новую порцию данных и так далее.

Если в результате какого-либо запроса получено не та команда сервера, которую ожидал клиент, либо получена специальная команда Initialize, клиент переходит в начальное состояние.

Протокол общения между клиентом и сервером X-Com реализован поверх протокола HTTP с использованием библиотеки libwww-perl (LWP). Использование HTTP позволяет организовать соединение через HTTP-прокси в том случае, когда сервер напрямую недоступен с клиента, а промежуточные серверы по каким-то причинам не применяются.

Файлы, указанные в описании задачи, загружаются при помощи метода GET протокола HTTP. Все остальные запросы клиента серверу реализованы при помощи метода POST. Первая часть запроса и ответа представляет собой описание на языке XML. Если в запросе клиента или в ответе сервера нужно передать дополнительную информацию (порцию данных или результат

расчета), то передается MIME-сообщение из нескольких частей, первая из которых – это описание на XML, а остальные – необходимые данные.

Для разбора XML используется модуль XML::Parser, который является надстройкой над библиотекой Expat. Для создания XML используется модуль XML::Writer.

Для изоляции кода клиента X-Com от кода клиентской части прикладной задачи, который также пишется на Perl, используется модуль Safe из комплекта поставки Perl. Этот модуль позволяет запускать код в изолированном окружении без получения доступа к пространству имен вызывающих модулей.

3.3. Определение характеристик вычислительных узлов

Корректное определение характеристик узлов позволяет учесть специфику аппаратного и программного обеспечения для того, чтобы, во-первых, повысить эффективность работы клиентской части, во-вторых, избежать связанных с несовместимостью проблем, в-третьих, позволить более точно оценить эффективность работы задач на X-Com в целом. Зная архитектуру узла, можно оптимизировать клиентскую часть задачи с учетом этой архитектуры. С точки зрения совместимости интерес представляют характеристики, которые были распознаны и могут использоваться работающей на узле операционной системой и приложениями, в ней выполняющимися.

Клиент X-Com собирает и предоставляет серверной части X-Com следующие данные:

- тип операционной системы (Linux/Windows)
- аппаратную архитектуру процессора (x86/x86_64/IA64/MIPS/Alpha/PowerPC)
- производительность процессора (flops)
- объем оперативной памяти (bytes)

Информация об аппаратной архитектуре - это, по сути, часть информации о самой операционной системе, точнее, о том, на каких архитектурах ей предназначено работать и какие приложения могут на ней выполняться. Поэтому было решено брать архитектуру прямо из версии операционной системы, работающей на узле. В Linux архитектуру можно узнать с помощью системного вызова uname, в Windows архитектуру можно получить с помощью интерфейсов WMI (Windows Management Instrumentation).

Объем оперативной памяти, контролируемой ОС, также весьма важен для решения задач с помощью X-Com. Так, если объем памяти узла относительно невелик, то этот факт необходимо учитывать, передавая узлу на обработку порцию данных или исполняемую часть задания.

Современные процессоры предоставляет доступ к технической информации о себе через ассемблерную инструкцию CPUID, которая поддерживается практически всеми моделями различных производителей, начиная с Intel 486DX/SX/DX2 SL, AMD 486DX4, Cyrix 6x86 (M1) и UMC U5S. В зависимости от параметра (содержания регистра EAX) можно получить информацию о производителе, архитектуре, номере модели, номере семейства, тактовой частоте, а также о поддерживаемых расширениях инструкций, дающих доступ к определенным архитектурным дополнениям процессора (например, Streaming SIMD Extensions - SSE, позволяющим за один такт производить сразу несколько операций).

ОС Windows и Linux самостоятельно с помощью CPUID определяют характеристики процессоров. Данные, полученные Windows, сохраняются в реестре, а Linux информацию обо всех процессорах отображает в файле /proc/cpuinfo. Следует отметить, что определенные версии ОС Linux могут и "не заметить" некоторых особенностей процессора. Дело в том, что CPUID возвращает информацию о поддерживаемых инструкциях и связанных с ними архитектурных решениях в виде набора битовых флагов, каждый из которых отвечает за поддержку определенных инструкций или является зарезервированным на будущее. Разбор операционной системой массива флагов может быть неверным просто потому, что ОС является устаревшей и попрежнему считает определенный бит зарезервированным и игнорирует представленное в нем значение. Например, ОС Linux с ядром версии 2.6.17-13mdv (Mandriva 2007), вызывая CPUID с параметром в EAX=0x1 на процессоре AMD Phenom, может проигнорировать значение 6го бита в ECX, который отвечает за поддержку SSE4a, и этот факт не найдет отражения в /proc/cpuinfo. Таким образом, с точки зрения ОС, инструкции SSE4a (да и все другие расшире-

ния, появившиеся позднее) не существуют, а следовательно, процессор "не умеет" выполнять за такт 4 операции над вещественными числами с их помощью.

Но это всего лишь означает, что сама ОС не сможет воспользоваться расширениями инструкций процессора, когда как приложениям использовать эти расширения можно, и за счет их использования можно достигнуть ускорения, на которое и рассчитаны дополнительные инструкции процессора. В этой связи было рассмотрено несколько вариантов определения производительности процессора на основе присутствующих в нем расширениях.

Первый вариант - самостоятельно обращаться к CPUID и разбирать массив флагов. Это неудобно тем, что в реализации невозможно обойтись только сценариями Perl; при использовании исполняемой бинарной компоненты будет нанесен ущерб кросс-платформенности клиентской части X-Com.

Второй вариант заключается в определении производительности только по трем значениям: производитель, идентификатор модели и тактовая частота. Эти значения корректно определяются практически всеми современными версиями Linux и Windows. Такой подход более гибок и потому является предпочтительным (несмотря на то, что модельный ряд процессоров постоянно пополняется и в этой связи процедуру определения производительности следует регулярно пересматривать и обновлять).

3.4. Механизмы сетевой безопасности в системе X-Com

При общении в Internet между клиентской частью X-Com и серверной частью возможны различного рода удаленные атаки. Выделяются следующие типы зловредных действий.

Анализ и/или модификация трафика. Если злоумышленник имеет доступ к узлу, через который проходит трафик, то он имеет возможность снимать с него передаваемые данные и/или модифицировать их. При анализе трафика конфиденциальность нарушается, а модификация (модификация порций данных, клиентской части задания, описания узлов, результатов вычислений, управляющих сообщений) в результате может привести к срыву вычислительного процесса.

Подмена одной из сторон передачи данных. При подмене серверной части злоумышленник получает контроль над клиентами X-Com, что может привести не только к срыву вычислительного процесса, но и к использованию клиентов в зловредных целях, (например, для распределенных удаленных атак или для нанесения ущерба узлам, на которых работают клиенты). При подмене клиентской части злоумышленник может передавать неверные результаты вычислений или не передавать их вовсе. Особенно ощутимым такое вмешательство может оказаться в случае, если подменяемый объект – промежуточный сервер; в таком случае все клиенты, находящиеся под ведением истинного промежуточного сервера, выбывают из вычислительного процесса. Также при такой подмене нарушается конфиденциальность: злоумышленник получает часть порций данных, предназначенных для обработки.

Внедрение ложного клиента. Злоумышленник может принять участие в вычислениях, используя модифицированный клиент X-Com. При этом он получает доступ к порциям данных, предназначенных для обработки, и имеет возможность отсылать неверные результаты вычислений.

В целях защиты от описанных действий были реализованы механизмы идентификации клиентов и серверов, шифрования трафика и верификации результатов вычислений.

Шифрование трафика серьезно затрудняет его анализ и модификацию. Идентификация при каком-либо общении клиента с сервером позволяет установить, являются ли обе стороны доверенными. Это не дает злоумышленнику их подменить или внедрить своего клиента. Верификация результатов вычислений помогает установить подлинность полученных результатов.

В X-Com механизмы идентификации и шифрования реализованы с помощью OpenSSL – открытой реализации протоколов TLS/SSL [7], а также утилит для шифрования, создания центров сертификации (ЦС), проверки сертификатов и др.

В клиентской части X-Com используется модуль Crypt::SSLeay, обеспечивающий поддержку протокола HTTPS для LWP::UserAgent (libwww-perl). Для того, чтобы начать безопасную передачу данных по http, вместо префикса http в URL указывается префикс https. Дополнительно в переменных окружения HTTPS_KEY_FILE, HTTPS_CERT_FILE, HTTPS_CA_PATH

(Рис. 4) указывается путь к закрытому ключу клиента, сертификату и директории с сертификатами доверенных центров сертификации соответственно.

```
   ...
   $ENV{HTTPS_KEY_FILE} =
‘/home/user/gcli/certs/clientkey.pem’;
   $ENV{HTTPS_CERT_FILE} =
‘/home/user/gcli/certs/clientcert.pem’;
   $ENV{HTTPS_CA_PATH} = ‘/home/user/gcli/certs/TRUSTED’;
   ...
```

Рис. 4. Указание расположения файлов сертификатов и ключа

В случае, если файл clientkey.pem содержит закрытый ключ в зашифрованном виде, то Crypt::SSLeay будет запрашивать ввод от пароля при каждой попытке прочесть ключ.

В директории TRUSTED находятся все сертификаты доверенных центров сертификации и выпущенные ими CRL. Имена этих файлов должны быть “<хеш от имени ЦС>.r0”. Также имя файла может оканчиваться на “.r1”, “.r2” и так далее, если файлов сертификатов и CRL с таким хешем несколько. При указании всех этих параметров клиент X-Com будет всегда требовать у сервера наличия сертификата, подлинность которого можно установить с помощью сертификатов, находящихся в директории TRUSTED.

В серверной части X-Com для поддержки TLS/SSL необходимы модули Net::SSLeay, IO::Socket::SSL и HTTPS::Daemon::SSL. При подключении клиента производится SSL handshake, который параметризуется указанием сертификата и ключа сервера, а также директории с сертификатами доверенных ЦС. В случае, если файл serverkey.pem содержит закрытый ключ в зашифрованном виде, будет запрашиваться пароль к ключу. Дополнительно есть возможность назначить функцию, которая будет вызываться вместо запроса на ввод пароля пользователем, если указать дополнительный параметр SSL_passwd_cb, значение которого должно быть указателем на соответствующую функцию, возвращаю строку пароля.

При подключении клиента обе стороны полностью проверяют сертификаты друг друга (прозрачная реализация такой проверки полностью включена в модули Crypt::SSLeay и Net::SSLeay), и, если она прошла успешно, происходит передача данных по шифрованному каналу.

Каждый работающий клиент или сервер должен использовать собственные ключи шифрования и собственный сертификат. Поскольку одно лицо может отвечать за работу сразу множества клиентов X-Com, было предложено использовать сертификаты, условно называемые Сертификаты Администратора клиентов, которые могут быть использованы в качестве сертификата локального ЦС, предназначенного для подписи сертификатов клиентов, находящихся в ведении Администратора. В таком случае, для того, чтобы сервер мог аутентифицировать клиента с сертификатом, подписанным администратором клиента, в директории TRUSTED на сервере должны лежать сертификаты администратора, сертификат доверенного ЦС, подписавшего сертификат администратора, и так далее до сертификата корневого ЦС.

Для того, чтобы клиент мог аутентифицировать сервер, в директории TRUSTED должны быть сертификаты, составляющую цепочку от ЦС, подписавшего сертификат сервера до корневого ЦС. Файлы CRL, изданные каждым ЦС из упомянутых цепочек, также должны присутствовать в TRUSTED, что пресечет успешную аутентификацию стороны, пользующейся не действительным сертификатом.

4. Заключение

Новая версия системы X-Com была апробирована в ходе серии масштабных вычислительных экспериментов, проведенных в ноябре 2008 г. Построенная распределенная среда для экспериментов объединила ресурсы крупнейших суперкомпьютерных центров МГУ (г. Москва), ЮУрГУ (г. Челябинск), УГАТУ (г. Уфа), СПбГТУ (г. Санкт-Петербург), ИВМиМГ СО РАН (г. Новосибирск) и ТГУ (г. Томск). В ходе экспериментов помимо решения реальных прикладных задач были отработаны и новые архитектурные решения, примененные в системе. Был выявлен и устранен ряд узких мест системы, при этом эффективность работы системы и среды в целом

была крайне высокой. Мы искренне благодарны коллегам из упомянутых университетов за помощь и исключительно доброжелательное отношение к нашим исследованиям.

Система X-Com распространяется свободно, дистрибутив и документация к системе доступны на официальном сайте проекта [1].

Литература

1. Система метакомпьютинга X-Com: [http://x-com.parallel.ru/], 12.02.2009
2. The Globus Alliance: [http://www.globus.org/toolkit/], 12.02.2009
3. Condor Project Homepage: [http://www.cs.wisc.edu/condor/], 12.02.2009
4. Berkeley Open Infrastructure for Network Computing: [http://boinc.berkeley.edu/], 12.02.2009
5. Соболев С.И. Использование распределенных компьютерных ресурсов для решения вычислительно сложных задач // Системы управления и информационные технологии. 2007, №1.3 (27). С. 391-395.
6. М.Ю. Медведик, Ю.Г. Смирнов, С.И. Соболев. Параллельный алгоритм расчета поверхностных токов в электромагнитной задаче дифракции на экране // Вычислительные методы и программирование. 2005. Том 6, №1. 86-95.
7. OpenSSL: The OpenSource toolkit for TLS/SSL: [http://openssl.org/], 12.02.2009