ПОСТРОЕНИЕ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕД НА ОСНОВЕ ОДНОРАНГОВЫХ ОВЕРЛЕЙНЫХ СЕТЕЙ

Ю.А. Жолудев

Введение

Распределенные компьютерные технологии развиваются интенсивно во всем мире. Одним из центральных направлений в развитии распределенных вычислительных систем является построение вычислительных сред с использованием доступных неоднородных, территориально распределенных вычислительных ресурсов, характер участия которых в распределенных расчетах динамичен и зачастую основывается на добровольных началах их владельцев [1].

Для большинства таких систем характерна централизованность, при которой все ресурсы объединяет в среду один исполняемый компонент, размещенный на серверной стороне, задачами которого является не только координация расчетов, но и хранение их результатов. Это означает зависимость всей вычислительной среды и проводимых в ней расчетов от серверной части среды, представленной обычно одним или несколькими экземплярами сервера, т.е. явно присутствует компонент в распределенной системе, являющийся точкой полного отказа всей системы. Кроме того, в условиях большого количества участвующих в расчете вычислительных ресурсов, нагрузка, связанная с обеспечением работы вычислительной среды, концентрируется также на серверной части и каналах связи, что негативно сказывается на реальной производительности[2].

Еще одной особенностью таких сред является ограниченность класса задач, решаемых с их помощью: исходная задача должна быть поделена на сравнительно небольшие по вычислительной сложности и размеру исходных данных подзадачи, независимые друг от друга. Все обмены данных возможны только через серверную часть среды; синхронизация состояния всех участвующих в расчете ресурсов через сервер - операция дорогостоящая, так как создает ощутимые дополнительные накладные расходы.

Подход с использованием одноранговых (или peer-to-peer)[3,4] оверлейных сетей позволяет ослабить создаваемые описанными особенностями традиционных распределенных вычислительных сред ограничения. Одноранговость сети означает равноправие всех её участников. Каждый компьютер, подключенный к такой сети, является одновременно и сервером, и клиентом. Это позволяет избежать обменов данными через одну выделенную точку в системе, передавая сообщения между распределенными компонентами напрямую и тем самым распределяя нагрузку на каналы связи и на вычислительные ресурсы. Также передача сообщений между участниками сети напрямую позволяет решать задачи, в которых решение одной части задачи зависит от решения другой части задачи. Использование одноранговых сетей уже показало свою эффективность в области распределенного хранения и распространения данных (файлообменные сети BitTorrent, DC)[5,6] и передачи сообщений (XMPP)[7]. Описанный подход позволяет решить не только проблему оптимального распределения нагрузки на каналы связи и вычислительные ресурсы участников среды при проведении расчетов, но и позволяет построить многопользовательскую распределенную вычислительную среду, с помощью которой можно эффективно решать множество задач с использованием большого количества неоднородных распределенных вычислительных ресурсов, накладывая при этом минимальные расходы при развертывании среды и отправке задачи на счет.

Требования

В соответствии с описанным выше, помимо безопасности, масштабируемости и производительности выдвигаются дополнительные требования к одноранговой распределенной вычислительной среде.

Пусть необходимо эффективно на доступных компьютерах (рабочих станциях, web-серверах и т.д.) решить большую задачу, допускающую разбиение на множество небольших подзадач, и решение одних таких подзадач может зависеть от результатов других подзадач, при этом структуру таких зависимостей заранее определить не представляется возможным. Примерами таких задач могут послужить задача ВЫП (выполнимость булевой формулы), многопараметрическая оптимизация (в том числе с использованием генетических алгоритмов), поиск кратчайшего пути в графе и многие другие. Формируется требование поддержки передачи сообщений между различными компьютерами, участвующими в расчете. Также для многих задач необходима поддержка распространения среди всех участников расчета большого объема исходных данных.

Помимо поддержки передачи сообщений и распространения данных, необходимо, чтобы решение задачи не прерывалось при выключении одного или нескольких компьютеров из расчета, то есть распределенная среда должна быть децентрализованной, и должна иметь возможность восстановления своего состояния при потере части данных, хранящихся на выключенных в ходе расчета компьютерах. Для этого каждый узел среды должен иметь возможность следить за изменениями состава участвующих в расчете узлов.

Функционально, распределенная вычислительная среда должна предоставлять API, позволяющий

порождать и назначать подзадачи, отправку и прием результатов решения подзадач с прозрачной (не зависящей от физического расположения узлов) адресацией, а также хранить результаты полностью решенных задач, сохраняя возможность их получения с любого из участвующих в расчете узлов.

Дополнительным требованием к вычислительной среде является легкость: далеко не все владельцы вычислительных ресурсов готовы предоставить права администратора на своих компьютерах, а сами компьютеры не всегда свободны в той степени, чтобы можно было позволить длительную установку и настройку программного обеспечения на каждом из них.

Наиболее заметными в области одноранговых распределенных вычислительных систем являются проекты Ourgrid[8] и Orbweb[9], использующие в качестве основы для построения одноранговой сети набор открытых стандартов XMPP(Jabber), изначально разработанный для обмена текстовыми сообщениями. Оба проекта поддерживают должный уровень безопасности, производительности и поддержки различных программно-аппаратных платформ, но и их использование накладывает ряд ограничений, несовместимых с требованием легкости.

Ourgrid представляет собой основанную на peer-to-peer сети инфраструктуру для передачи на выполнение так называемой bag of tasks, представляющей собой набор независимых между последовательных программ, которые впоследствии параллельно запускаются на доступных в инфраструктуре компьютерах в виртуальной машине.

Orbweb, в отличие от Ourgrid представляют собой библиотеку для построения одноранговой сети на основе XMPP, которая предоставляет средства не только для обеспечения связи между участниками сети, но и средства групповой пересылки сообщений, оптимизированное сжатие XML-данных и многое другое, что позволило использовать Orbweb в качестве среды передачи сообщений для распределенного масштабного решения задачи выполнимости [10].

Для развертывания вычислительной среды на доступных ресурсах с использованием инструментария Ourgrid или Cohesion Orbweb необходимы права администратора: требуется устанавливать в систему дополнительное программное обеспечение (в обоих проектах используется XMPP-сервер Openfire в качестве основы одноранговой сети); дистрибутив Ourgrid имеет размер порядка 2 Гб. Кроме того, затраты на развертывание среды включают в себя процесс тонкой настройки устанавливаемых на компьютеры компонент.

Архитектура

Архитектура приложения узла строится из соображения равноправия всех участников оверлейной сети, на основе которой строится распределенная среда, поэтому, независимо от того, какие роли выполняют узел при счете задачи, приложение у всех работает одно и то же.

Из соображений легкости в установке, кроссплатформенности и безопасности в качестве языка и среды выполнения была выбрана технология программирования Java. Java является объектно-ориентированным языком программирования, разработанным в начале 1990 годов. Одной из главных черт Java является независимость от программно-аппаратной платформы, благодаря компиляции исходных кодов в промежуточный байткод, исполняемый в виртуальном окружении Java Runtime Environment, реализация которого доступна для практически всех современных широко используемых операционных систем. Встроенный в виртуальное окружение функционал платформенной виртуализации позволяет программно определять целый ряд настроек для обеспечения безопасности запуска программ.

На рисунке 1 представлен общий вид компонентов разрабатываемого приложения, где стрелками отмечено направление передачи данных. Архитектура приложения строится в виде 3 уровней абстракции. Верхний уровень — прикладной. На нем расположен компонент Application manager, задачей которого является контроль всего жизненного цикла компонент-подзадач, начиная от порождения и заканчивая либо передачей задачи на выполнение другому узлу в сети либо завершением её выполнения. Еще одной важной функцией Application Manager'a является хранение и передача состояния выполняемых под его контролем подзадач исходной задачи. Компоненты Application##:task## представляют собой подзадачи, порожденные в ходе решения исходной задачи. Каждая подзадача должна поддерживать сериализацию своего состояния в момент блокировки при ожидании сообщения. Их выполнение может мигрировать (в соответствии с логикой работы Application Manager'a) с узла на узел, в зависимости от загруженности узлов сети. Каждой подзадаче выделяется отдельный буфер, в котором хранятся все пришедшие и не доставленные сообщения, передаваемые через компоненту Application messaging.

Уровнем ниже располагается уровень абстракции между прикладными компонентами и внешними библиотеками однорангового взаимодействия по сети. Компонента Application storage предоставляет интерфейс для подзадач для хранения итоговых результатов расчетов и объемных исходных данных задачи в глобальном хранилище (которое, вообще говоря, необязательно должно быть распределенным) и предоставляет средства для прозрачного именования файлов, изолируя таким образом данные одной большой задачи от другой. Компонента Application messaging предоставляет прозрачную адресацию (в рамках одной задачи адресатами являются подзадачи, выполняющиеся на других узлах) при отправке и получении сообщений. Компонента Infrastructure messaging отвечает за передачу служебных сообщений между узлами одноранговой сети. Под служебными сообщениями понимается запрос или отправка подзадачи на счет кому-либо из простаивающих

узлов.

На нижнем уровне архитектуры расположены конкретные реализации оверлейных сетей. В частности для распределенного хранения файлов и для передачи сообщений используются разные сети: DHT (BitTorrent) для хранения и распространения больших файлов, XMPP или JXTA[11] — для передачи сообщений.

Между уровнями располагается компонента Peer Discovery+Message Routing, задачей которой является определение того, через какие узлы будут проходить сообщения (Message Routing) и распознавание и оповещение новых узлов о смене состояния вычислительной среды. Данный компонент находится между уровнями, потому что часть алгоритмов по маршрутизации сообщений и поиску уже реализованы в рамках протоколов XMPP и JXTA, а часть реализуется на основе средств упомянутых библиотек в соответствии с абстракциям на уровне Application messaging и Infrastructure messaging, потому как требования к объему, скорости и содержанию сообщений этих двух компонент разные.

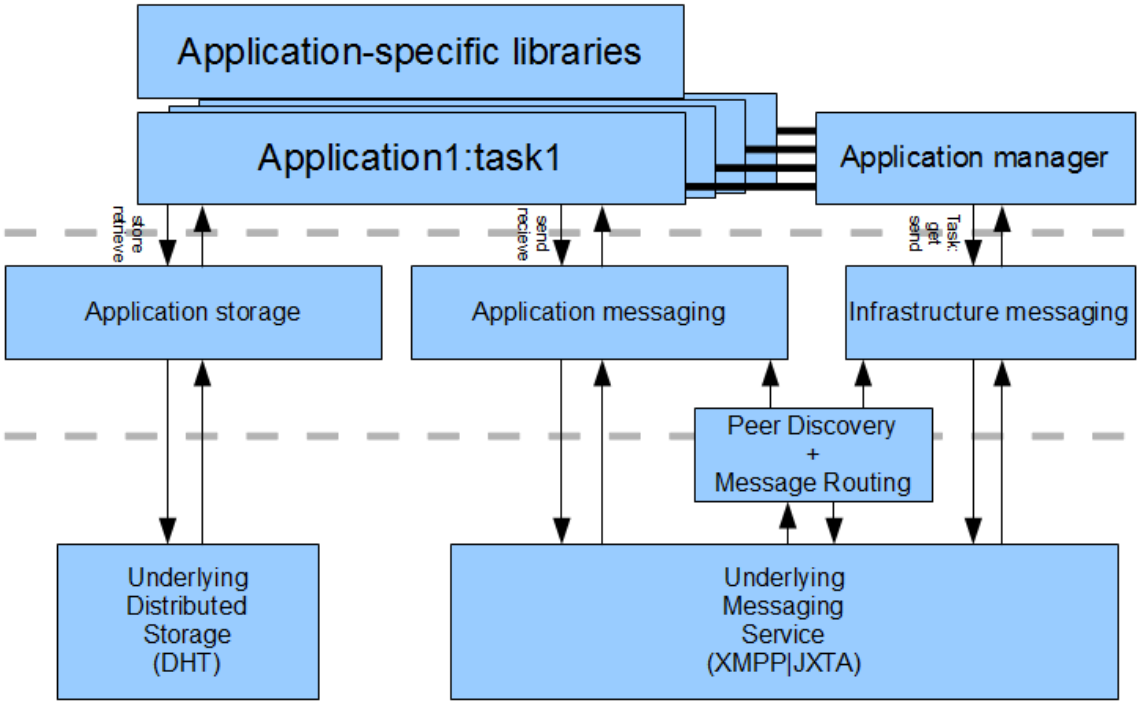

Рис. 1 Архитектура приложения узла распределенной среды

Кратко опишем жизненный цикл нового узла с новой задачей в распределенной среде.

При запуске приложения указываются параметры: адрес любого из способных принимать соединения узлов в распределенной среде и, если административные ограничение это допускают — номер порта, по которому приложение может принимать входящие соединения. Далее, компонента Peer Discovery опрашивает начальный узел на предмет соседей и сведения о топологии распределенной среды (представляет собой ориентированный граф), включая пропускную способность каналов связи и характеристики узлов. После инициализации нижнего уровня архитектуры, порождаются объекты Infrastructure messaging и Application manager, для которых генерируется уникальный идентификатор сессии узла и происходит загрузка и запуск Java-сборки с исходной задачей (которая с точки зрения приложения есть всего лишь одна подзадача) и исходных файлов задачи. Для задачи и всех подзадач также генерируется Application manager'ом уникальный идентификатор, по которому впоследствии будут разрешаться адреса при передаче сообщений и пути к файлам в Application storage. После запуска задачи Application manager оповещает все соседние узлы среды о наличии новой задачи, и данное сообщение распространяется далее. Исходная задача при этом порождает свои подзадачи, запрашивая у Application Manager'a новые идентификаторы. Во все время работы всего приложения Application Manager последовательно передает на выполнение запущенные подзадачи и переключает управление на другие в моменты блокировок (сохраняя при этом состояние задачи) или при возможности миграции задачи на новые узлы. Для миграции задач существует несколько алгоритмов, в число наиболее эффективных среди них входят являются random work stealing и random work pushing[12]: когда простаивающие узлы забирают на себя подзадачи из очередей случайно выбранных соседей (work stealing),а узлы с избытком задач, напротив, свои задачи соседям передают (work pushing). Итоговые результаты решения задачи впоследствии сохраняются в Application storage, которые потом можно получить с помощью исходного

приложения по идентификатору задачи.

Предложенная архитектура дает широкие возможности для реализации автоматического распределенного дублирования всех процессов, происходящих на узлах, а для доведения задачи до конца совсем необязательно, чтобы узел, её поставивший на счет, всегда был подключен к сети.

Заключение

В рамках данной работы была рассмотрена возможность создания одноранговых распределенных вычислительных сред на основе одноранговых сетей. Большой опыт использования таких сетей, к которому в разных прикладных областях в разное время приходили от клиент-серверных архитектур, показал многие достоинства идеи peer-to-peer, включая высокие показатели масштабируемости, производительности и устойчивость к неисправностям. Для создания распределенных вычислительных сред на основе таких сетей уже есть практически всё, что нужно: имеются необходимые протоколы коммуникации, маршрутизации и поиска в одноранговых сетях, а также их реализации для множества языков программирования.

Исходя из таких соображений, была предложена архитектура и кратко рассмотрен способ взаимодействия узлов такой сети для решения больших задач, при этом были сформулированы требования, ранее не предъявлявшиеся к распределенным вычислительным средам, включая поддержку передачи сообщений между любыми узлами сети.

В данный момент ведется разработка инструментария на основе предложенного в данной работе подхода, который будет удовлетворять всем описанным в данной работе требованиям.

Работа выполняется при поддержке гранта Президента Российской Федерации для государственной поддержки молодых российских ученых-кандидатов наук МК-5104.2011.9.

ЛИТЕРАТУРА:

1. Система метакомпьютинга X-Com// URL:http://x-com.parallel.ru
2. А.С. Хританков. Анализ производительности распределенных вычислительных комплексов на примере системы X-Com. Труды Всероссийской суперкомпьютерной конференции "Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность (г. Новороссийск, 21-26 сентября 2009 г.), 2009, с. 46-52.
3. R. Steinmetz and K. Wehrle. Peer-to-Peer Networking and Computing.//Informatik-Spectrum, 27(1). Springer. 2004.
4. A. Kim and L. Hoffman. Napster and other Internet peer-to-peer applications.//George Washington University, 2002, citeseer. ist.psu.edu/kim01pricing.html.
5. S. Saroiu, K.P. Gummadi, R.J. Dunn, S. Gribble, H.M. Levy, An analysis of internet content delivery systems//in: Proc. of the Fifth Symposium on Operating System Design and Implementation, OSDI, 2002.
6. A. Bharambe, C. Herley, V. Padmanabhan, Analyzing and improving bittorrent performance//in Proc. of IEEE INFOCOM, 2006.
7. The XMPP Software Foundation.//URL: http://www.xmpp.org
8. OurGrid//URL: http://www.ourgrid.org/
9. Cohesion Orbweb.//URL: http://cohesion.de/cms/index.php?id=56
10. Sven Schulz Wolfgang Blochinger, Parallel SAT-Solving on Peer-to-Peer Desktop Grids//[PDF] http://cohesion.de/cms/fileadmin/publications/satciety.jogc2010.pdf
11. JXTA. URL: http://www.jxta.org
12. Robert D. Blumofe and Charles E. Leiserson., Scheduling multithreaded computations by work stealing.//Journal of the ACM, 46(5):720-748, 1999.
13. Son S., Livny, M.: Recovering Internet Symmetry in Distributed Computing.//Proceedings of GAN’03 Workshop on Grids and Advanced Networks, Tokyo, Japan, 12-15 May 2003.