УДК 681.322

В.В. БОНДАРЕНКО, Ю.Э. ТКАЧ, А.Н. СКОСЫРЬ

Черниговский государственный технологический университет, Украина

ОБЕСПЕЧЕНИЕ БЕЗОТКАЗНОГО РЕШЕНИЯ ЗАДАЧ В РАСПРЕДЕЛЕННОЙ СИСТЕМЕ НЕОТЧУЖДАЕМЫХ РЕСУРСОВ

В статье рассматриваются вопросы обеспечения безотказного решения задач в распределенной системе, особенностью которой является использование неотчуждаемых от владельца вычислительных ресурсов. Приводится описание системы, использующей вычислительные узлы, которые могут свободно подключаться, отключаться и управлять выделением своих ресурсов для общего использования, тем самым в значительной степени сохраняя автономию. Проводится анализ существующих решений и проблем, возникающих при использовании неотчуждаемых ресурсов. Особое внимание уделяется механизмам регистрации вычислительных узлов, пересылки и запуска задач на вычисление, обработки результатов вычислений и возможное перераспределение нагрузки. В заключении статьи приводятся выводы об эффективности применения неотчуждаемых ресурсов для распределенного решения задач и обеспечения безотказности такого решения.

распределенная система, параллельные вычисления, неотчуждаемые ресурсы, управление ресурсами

Введение

Распределенные вычисления в последние годы развиваются в направлении использования резервных вычислительных ресурсов простаивающих персональных компьютеров [1]. Являясь недорогой и эффективной альтернативой дорогим кластерам и специализированным машинам, распределенные вычисления позволяют проводить различные научные исследования с минимальными затратами.

Современные научные исследования часто включают длительные вычисления, которые могут занимать дни, недели, или даже годы при выполнении их на отдельном компьютере. Экспертиза любой современной организации показывает, что существует сотни настольных компьютеров, простаивающих большую часть рабочего дня [2]. Наличие универсальной распределенной системы, использующей свободные ресурсы настольных компьютеров, позволит получить доступ к огромным вычислительным мощностям для выполнения необходимых расчетов.

Сложность эксплуатации кластеров и специализированных компьютеров требует наличия простой распределенной системы, которая избавит пользователей от длительного изучения сложной технической документации и упростит обслуживание системы. Изучение специализированных параллельных языков программирования для специальных параллельных машин может занимать длительное время. Возможность запрограммировать параллельные вычисления на одном из известных непараллельных языках с применением визуального проектирования позволит существенно сократить время разработки распределенных приложений.

Использование неотчуждаемых ресурсов для организации вычислений вносит новые проблемы в разработку распределенной системы: подключение и отключение ресурсов осуществляется в любой момент времени, сложная структура сетей и неоднородность оборудования, возможность завершения работы вычислительного узла в ходе выполнения расчетов (аварийного или инициированного пользователем ресурса). Эти проблемы требуют создания механизмов, которые бы осуществляли управление за процессом решения задач в среде, построенной с использованием неотчуждаемых ресурсов.

Анализ существующих систем

Проанализировав работы в области созданных распределенных систем, можно констатировать, что реальная работа сегодня активно идет по трем направлениям [3]. Первое направление — это создание универсальных метакомпьютерных сред (GRID системы) [4]. Практически все основные производители программного обеспечения работают в данном направлении (системы GLOBUS, UNICORE, Condor). Многие берут в качестве стандарта Globus, создавая программную инфраструктуру для своих платформ. 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. Третье направление состоит в разработке инструментария для организации распределенных вычислительных экспериментов.

Condor – один из наиболее длительно используемых проектов распределенных вычислений [5]. Condor преобразовывает набор рабочих станций и выделенных кластеров в высокопроизводительное мощное вычислительное средство. Когда задача готова для выполнения и добавлена в очередь задач, Condor пытается найти ресурсы для ее запуска и в случае успеха осуществляет ее запуск. Система предоставляет возможность обработки аварийного выхода из строя вычислительного узла в процессе работы за счет использования механизма контрольных точек, который сохраняет состояние задачи.

Несмотря на значительные успехи Condor, он имеет ряд ограничений. Программное обеспечение не является полностью кроссплатформенным. Разные версии программного обеспечения требуются для различных платформ, и нет такой версии Condor, которая доступна для некоторого количества архитектур и платформ. Защита машин, на которых производится вычисление, не является ответственностью программного обеспечения Condor и поэтому не гарантии, что вычисления не поставят под угрозу работоспособность машины.

GLOBUS является GRID-системой, которая предоставляет набор средств нижнего уровня, обеспечивающие основные механизмы, такие как связь, аутентификацию, доступ к данным [6]. Эти механизмы используются при построении различных метакомпьютерных сервисов, таких как средства параллельного программирования и диспетчеров.

Запросы пользовательских приложений на выполнение вычислений передаются брокеру ресурсов, который отвечает за высокоуровневую координацию использования ресурсов. На основе переданного пользовательским приложением запроса брокер ресурсов принимает решение о том, на каких вычислительных узлах будет выполняться задача, какой процент вычислительной мощности узла она может использовать.

При выборе вычислительного узла брокер ресурсов должен определить, какие узлы доступны в текущий момент, их загрузку, производительность и другие параметры, указанные в запросе, выбрать наиболее оптимальный вариант, сгенерировать новый запрос и передать его высокоуровневому менеджеру ресурсов**.**

Платформа Unicore имеет несколько особенностей, обусловленных тем, что с самого начала она предназначалась для обеспечения доступа к суперкомпьютерным центрам стран Центральной Европы. Это одна из немногих оригинальных платформ, которая не использовала Globus Toolkit, и лишь в последнее время происходит трансформация ее протоколов в сторону общепризнанных для грид [7].

Перед выполнением заданий серверы UNICORE производят диспетчеризацию задач с учетом зависимостей между ними. Для каждой задачи входные и выходные файлы автоматически импортируются/экспортируются из/в файловое пространство пользователя или передаются от более ранних задач в том же задании.

Для каждого задания пользователь определяет нужную исполнительную систему и требования к ресурсам для каждой задачи. Пользователи могут следить за своими заданиями и управлять ими посредством монитора заданий, который графически отображает состояния.

Наряду с преимуществами вышеизложенных технологий у них существует ряд недостатков, таких как отсутствие стандартов GRID. Особенностью GRID-систем является полное отчуждение ресурсов пользователя. Также эти системы слишком тяжелы в установке и сложны в использовании. Как более простая в использовании система может выступать система X-COM [8].

Система строится по клиент-серверной архитектуре. Особенностью системы является возможность выполнять только одну задачу на вычислительном узле. Для изменения алгоритма вычислений следует осуществлять реконфигурацию вычислительного узла.

Инициатором взаимодействия является вычислительный узел, который осуществляет обращение к серверному узлу за получением новых данных. В процессе обработки данных узел уведомляет сервер о ходе вычислений. При не поступлении от прикладной программы сигнала о решении, задание, которое была оправлено на вычислительный узел, вновь становится доступным для других прикладных программ.

Общая структура распределенной системы

Распределенная система состоит из нескольких модулей, которые, взаимодействуя между собой, позволяют пользователю спроектировать вычисления, выполнить решение необходимой задачи и обработать результат.

Модуль проектирования представляет собой отдельную программу в виде графического приложения, которая позволяет пользователю выполнять проектирование вычислений и управлять распределенной системой.

Модуль управления данными используется для эффективного управления данными в распределенной системе. Модуль оперирует параметрами задач, производит их загрузку на вычислительные узлы, осуществляет обмен данными между узлами и внешней средой.

Модуль мониторинга представляет комплекс средств сбора, хранения и анализа информации о состоянии вычислительных узлов и о ходе решения задач в системе.

Модуль управления заданиями используется для управления задачами в системе. Модуль пересылает задачи на рабочие узлы, запускает их на вычисление, обрабатывает результаты и перераспределяет задачи при выходе из строя рабочего узла, который используется для решения задачи.

Модуль состоит из двух относительно независимых частей: компоненты выполнения задач и компоненты вычислительный узел. Компонента выполнения задач, находящаяся на центральном узле, производит подготовку задач для решения в распределенной системе и обработку результатов вычислений. Компонента вычислительный узел осуществляет управление за решением отдельной задачи на узле, пересылку результатов вычислений и обеспечения механизмов предотвращения невыполнения задачи.

Модуль диспетчеризации содержит два специализированных диспетчера распределения задач, которые управляют выделением ресурсов рабочих узлов и распределением задачи между ними.

Модуль взаимодействия производит управление всеми модулями системы, которое включает обеспечение связи между модулями, проверку прав доступа, проверку подлинности информации, получаемую от модулей.

Архитектура системы. Система реализована по принципам иерархической клиент-серверной архи-

тектуры, в которой можно выделить три основных компонента: главный сервер, подчиненный сервер и рабочий узел.

Главный сервер, который может физически включать несколько компьютеров, содержит все модули системы и производит централизованное управление системой. Наличие нескольких отдельных компьютеров позволяет запустить на них разные модули, тем самым повысить отказоустойчивость системы.

Рабочий узел – это любой вычислительный ресурс (рабочая станция, виртуальная машина и т.п.), на котором могут храниться распределенные данные решаемых задач, а также происходить основные расчеты прикладных задач. На рабочем узле запускается минимального размера приложение, которое представляет собой отдельные части некоторых модулей системы (модуль управления данных, задачами, модуль мониторинга и взаимодействия).

Подчиненный сервер вводится в систему для организации более эффективных коммуникаций и подключения большего количества узлов. Для главного сервера подчиненный сервер выглядит как рабочий узел большой мощности. Для низлежащих рабочих узлов – как главный сервер.

Основные механизмы для обеспечения безотказного выполнения задач встроены в модуль управления заданиями.

Одной из особенностей компонента выполнения задач является наличие трех очередей, использующихся для хранения задач в соответствии с их текущим положением в системе:

– очередь ожидающих задач используется для хранения задач, которые требуют выполнения в системе;

– очередь выполняющихся задач, содержащая задачи, которые в данный момент выполняются в системе;

– очередь «подвешенных» задач, в которой находятся задачи, только что выполнившиеся и ожидающие действия по своей обработке. Особенностью задач является то, что одна задача может выполняться несколько раз, в соответствии с шагами итерации. Поэтому после выполнения задачи перемещаются в очередь подвешенных задач, где происходит анализ возможной следующей итерации или ее отсутствия. После анализа задача или вновь помещается в очередь ожидающих задач, или удаляется из очереди в случае отсутствия итерации.

Процесс обработки задачи представляет собой последовательный процесс перемещения в очередях, использование которых позволяет оперативно получать информацию о процессах решения в системе.

Использование очередей задач позволяет эффективно обрабатывать возможные сбои в работе вычислительных узлов системы. Задача при решении в системе одновременно хранится в очереди выполняющихся задач и на конечном вычислительном узле, где происходит ее выполнение. Такая избыточность позволяет в случае нарушения работы ВУ производить повторный запуск задачи на доступном вычислительном ресурсе.

Структура компонента вычислительный узел более проста, так как перед данным компонентом стоят более простые задачи. Основными его задачами являются:

– получение задачи для вычисления и ее запуск;

– передача результатов вычисления задачи для дальнейшего использования в системе.

Дополнительной возможностью компонента вычислительный узел являются накопление статистической информации, касающейся выполнения задач, которая может в дальнейшем изучаться при анализе процесса вычислений.

Важным вопросом, касающимся безопасности работы системы, является вопрос решения задачи в условиях ненадежности узла. Особенностями работы узлов является их возможный отказ в процессе работы. Это требует внесения некоторых дополнительных требований к компоненту вычислительный

узел: в процессе работы необходимо, чтобы вычислительный узел периодически извещал компонент управления о ходе выполнения задачи. Отсутствие сигнала от ВУ позволит компоненту управления судить об отказе узла и перераспределить нагрузку на доступный узел.

Последовательность взаимодействия компоненты выполнения задач с компонентом вычислительный узел:

1. Компонент выполнения задач производит выбор узла для решения задачи.
2. Пересылка задачи на вычислительный узел.
3. Запуск задачи на вычисление. Происходит после получения задачей всех необходимых входных параметров. Данные для обработки задачей доставляются на узел посредством использования модуля управления данными.
4. Периодическое извещение вычислительным узлом компонента выполнения задач о процессе вычисления задачи.
5. Пересылка результатов решения.

Описанный выше процесс взаимодействия представляет собой возможность решения, исключающий невыполнение задачи вследствие выхода из строя одного из вычислительных узлов.

Выводы

Предложена структура распределенной системы, которая ориентирована, в первую очередь, на крупнозернистый параллелизм и позволяет использовать различные вычислительные сети для эффективной организации решения задач большой размерности, позволяет гарантированно решить задачу в системе с использованием неотчуждаемых ресурсов. Распределенная система скрывает от пользователя детали работы с вычислительными узлами и механизмами их взаимодействия, управления данными и распределения заданий, позволяя повысить уровень абстракции при описании параллельных задач.

Высокая безопасность, свободная расширяемость и кросс-платформенность, отказоустойчивость и динамическая адаптация к изменениям распределенной среды позволяют эффективно использовать систему в гетерогенной вычислительной среде любой организации. Модульное построение системы позволяет гибко изменять и наращивать ее функциональность. Простота эксплуатации и программирования задач, мониторинг работы всех элементов системы, а также возможность вычислять несколько задач одновременно, делают систему удобным и эффективным инструментом решения задач для прикладных программистов.

Литература

1. Coulouris G., Dollimore J., Kindberg T. Distributed Systems // Concepts and Design, Addison Wesley, 2001. – P. 34-41.

2. Heap D.G. Taurus. A Taxonomy of Actual Utilization of Real UNIX and Windows Servers // Technical White Paper GM12-0191. – IBM eServer Library on ibm.com, 2003. – P.132-137.

3. Баер Ж.-Л., Барлоу Р., Вудворд М. и др. Системы параллельной обработки. – М.: Мир, 1985. – 416 с.

4. Joseph J., Fellenstein C. Grid Computing. – Prentice Hall, ISBN-0-1314-5660-1 - 2004.

5. Владимиров Д. Кластерная система Condor // Открытые системы. – 2000. – № 7-8. – С. 32-38.

6. Foster I., Kesselmany C. Globus: A Metacomputing Infrastructure Toolkit – 2005.

7. Dietmar Erwin – UNICORE Plus Final Report. Uniform Interface to Computing Resources. ISBN 3-00-011592-7 – 2004.

8. Филамофитский М.П.. Система поддержки метакомпьютерных расчетов X-COM: архитектура и технология работы. 2004. – 348 с.

Поступила в редакцию 15.03.2007

Рецензент – д-р техн. наук, проф. О. Г. Руденко, Харьковский национальный университет радиоэлектроники, Харьков.