М.А. СМЯТКИН

Обзор базовых компонентов распределенных систем

УДК 004.75

Муромский институт (филиал) ФГБОУ ВПО «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых», г. Муром

В статье рассматриваются основные компоненты распределенных систем как разновидности информационных систем: подсистемы хранения, передачи и обработки информации. Проводится обзор общих проблем при реализации компонентов, и приводятся традиционные подходы к их решению.

The paper examines basic components of distributed systems as the kind of information systems: data storage, exchange and processing subsystems. The overview of the most common problems of subsystems implementation and traditional solutions are provided.

Ежедневный рост обрабатываемых информационными системами данных привел к возникновению потребности в создании более эффективных вычислительных систем. Изначально, увеличение производительности достигалось за счет улучшения аппаратной части ЭВМ. Однако, в связи с некоторыми ограничениями (такими как максимальная допустимая тактовая частота ЦПУ) и высокой стоимостью новых устройств развился новый подход к созданию высоконагруженных систем – проектирование распределенных систем.

Распределенная система – это набор независимых компьютеров, предоставляющийся их пользователям единой объединенной системой [2]. Как любая информационная система, распределённая система состоит из следующих трёх компонентов:

1. Подсистема хранения информации.
2. Подсистема передачи информации.
3. Подсистема обработки информации.

Каждый отдельный компонент, помимо индивидуальных требований, должен удовлетворять требованиям всей системы в целом. За время формирования теории распределенных систем были созданы следующие фундаментальные требования [2]:

1. Масштабируемость. Система может быть расширяемой по трем различным показателям: нагрузке, географическому охвату и возможности независимого администрирования.

2. Открытость. Каждый функционирующий автономный узел системы должен быть в любой момент времени доступен другим узлам.

3. Прозрачность. Распределенная система для конечного пользователя должна иметь интерфейс, схожий с локальной системой.

Данные требования в совокупности с определением распределенной системы порождают ряд характерных проблем, которые необходимо решить при проектировании компонентов системы. Ниже рассмотрены соответствующие проблемы и их краткое описание, а затем изложены традиционные подходы к их решению.

Базовые проблемы проектирования компонентов распределенной системы [2, 3, 4].

Подсистема хранения информации в наиболее изолированном виде встречается в распределенных файловых системах. Основные проблемы, которые должна решать рассматриваемая подсистема в идеальном случае следующие:

1. Предоставление и контролирование распределенного доступа к данным. Доступ к данным для пользователя должен предоставляться через интерфейс подобный локальному аналогу, обеспечивая эффективность и расширяемость распределенной системы. Наиболее важной проблемой на данном этапе является разделение доступа между несколькими клиентами.

2. Репликация (тиражирование) данных. С целью повышения безопасности и балансировки нагрузки между узлами, хранимые в системе данные копируются на несколько узлов, тем самым выигрывая за счет повышения избыточной информации.

3. Поддержка непротиворечивости данных. Тиражируемая информация и информация, к которой предоставляется разделяемый доступ нескольких клиентов, подвержена появлению конфликтующих копий данных. Таким образом, одна из задач подсистемы – это контроль их согласованности.

Подсистема передачи информации, так или иначе, существует в любой распределенной системе. Цель данного компонента – обеспечить коммуникацию между узлами системы. Проблемы, которые требуется для этого решать, следующие:

1. Обеспечение отказоустойчивости. В распределённой системе отказ узла, как правило, обрабатывается как штатная ситуация и клиентская сторона должна быть максимально изолирована от подобных неполадок. То есть, отказ узла не должен влиять на общую работу распределенной системы.

2. Реализация прозрачной коммуникации между различными узлами системы. Существуют системы с общей и распределенной памятью. В локальных системах наиболее частым решением является использование общей памяти, но для распределенных систем в связи с особенностями архитектуры это является достаточно трудоемким подходом. Поэтому большую важность приобретает реализации коммуникаций между узлами.

3. Эффективность передачи данных. В то время как в подсистеме хранения балансировка нагрузки осуществляется на уровне загрузки хранящих информацию устройств, для передачи информации более актуальной является проблема поддержки эффективной загрузки передающего канала.

4. Надежность передачи информации. При передаче данных по сети нельзя делать предположений о доставке информации на узел в том виде, в котором она была отправлена. Существует строгая необходимость передачи информации в избыточной форме с целью проверки её актуальности на стороне адресата. Кроме того, в ряде систем необходимо также обеспечение защищённости информации.

Ролью подсистемы обработки информации в распределенной системе также является имитация локального аналога с применением достоинств распределенных систем: эффективности, масштабируемости, надежности. Основные проблемы, которые необходимо решить при проектировании этой подсистемы следующие:

1. Имитация локальной системы. При соблюдении простоты масштабируемости и высокой эффективности, распределенная система для конечного пользователя должна выглядеть максимально похожей на локальный аналог.

2. Оптимизация использования удаленных вычислительных ресурсов. Так, запросы на обработку данных в вычислительной системе должны оптимизироваться для максимально эффективной параллельной обработки.

3. Контролирование конкурентной обработки данных. Для систем параллельной обработки характерна проблема конкурентного доступа к данным. Данная проблема включает в себя управление блокировками, контроль версий и согласованности данных.

Традиционные подходы к решению обозначенных проблем.

Подсистема хранения информации [1]. Методы решения проблем распределенного хранения информации не должны нарушать требования прозрачности топологии системы. Традиционный подход для соблюдения прозрачности топологии – это использование двойной адресации сетевой файловой системы: физический (фактический) и логический адреса используются системой и клиентами соответственно, при этом одному логическому может соответствовать несколько физических.

Распределенный доступ к данным может предоставляться удалённо и локально (блоки данных передаются на клиент, где продолжается их обработка). В случае удаленного доступа обработка определяется набором предоставляемых сервисами функций, а контроль согласованности данных осуществляется стандартными средствами централизованных систем. В случае работы с локальными копиями данных существует ряд стандартных методов поддержки согласованности:

- семантика UNIX (изменения немедленно передаются на сервер);

- сессионная семантика (изменения видны после закрытия файла);

- механизм транзакций (все изменения атомарные);

- данные не подлежат обновлению (в некоторых системах возможен данный подход, что упрощает также процессы кэширования и репликации);

Важной характеристикой при обеспечении доступа к файлам является хранение состояния клиентов на сервере. Соответственно системы разделяются на stateless и statefull. Системы, хранящие со-

стояние о клиентах позволяют реализовывать блокировки на данных и снижать передаваемый по сети трафик, но увеличивая тем самым нагрузку на сервер.

Также существуют различные подходы для кэширования файлов: кэширование на стороне сервера и на стороне клиента. На стороне клиента кэширование может выполняться стандартными средствами операционной системы, системными вызовами ядра или отдельным кэшем, реализованным в клиентском приложении.

Для реализации репликации существуют три основных метода: точная репликация, ленивая и групповая. Точная репликация слабо пригодна для распределенных систем, т.к. ответственность за репликацию ложится на программиста. При групповой репликации клиентские операции адресуются сразу нескольким серверам, исключая возможность автоматической балансировки нагрузки на серверах и замедляя выполнение операции. Для снятия данных ограничений используется ленивая репликация – метод, когда запрос направляется на один сервер, а система позже находит наиболее подходящий момент для репликации.

Для поддержания согласованности данных при репликации разработаны три базовых алгоритма – репликация первой копии, голосование и голосование с приведением. Данные алгоритмы и более подробное описание приведенных выше методов можно найти в [1] и [4].

Подсистема передачи информации. Наиболее традиционным подходом для решения проблемы работы системы независимо от периодических отказов узлов является реализация всей системы в виде совокупности автономных узлов, решающих отдельные задачи. Такие узлы обладают минимальной информацией о топологии сети и других узлах, а система, придерживающаяся данного правила называется децентрализованной и предоставляет больше возможностей для реализации отказоустойчивости [7]. В противоположность децентрализованной системе существует понятие централизованной системы, в которой существуют узлы, непосредственно отвечающие за коммуникацию, балансировку нагрузки, проведение вычислений и т.д.

Централизованные системы, как правило, значительно проще в проектировании и реализации, однако центральные узлы – явля-

ются узким местом, как для производительности, так и для надежности.

Основная проблема при передаче информации в децентрализованной системе – это нахождение узлов для коммуникации. На практике существует ряд подходов, решающих эту задачу, один из которых – применение распределенной хэш-таблицы. Суть данного метода состоит в том, что узлу соответствует определенный индекс в хэш-таблице, который каждый другой узел способен вычислить в соответствии с поставленной задачей и, таким образом, способен в любой момент инициировать соединение. Данный подход используется в Torrent-протоколе и в ряде других реализаций.

Общих методов для построения устойчивой к отказам подсистемы передачи информации не существует, однако существует ряд рекомендаций, соблюдая которые можно найти частное решение для исследуемой предметной области. Подробный список рекомендаций можно найти в [3]. Наиболее важный подход к проектированию отказоустойчивой распределенной системы – не делать предположений о невозможности возникновения отказов, вероятность которых чрезвычайно мала.

Из-за отсутствия в распределенных системах естественной поддержки механизма общей памяти, эффективная реализация которой может оказаться чрезвычайно трудоемкой задачей, то для коммуникации применяются методы, характерные для систем с разделяемой памятью – методы передачи сообщений между процессами (узлами). Для параллельных систем был разработан стандарт MPI (интерфейс передачи сообщений), в полной мере применяемый и в распределенных системах.

Для соблюдения эффективной передачи данных существуют два различных подхода, применяемых совместно: снижение нагрузки на сеть уменьшением количества передаваемых данных и балансировка нагрузки между узлами. Первый вариант реализуется совокупностью различных идей, таких как кэширование данных, пересылка дельта версий и т.д., а второй – нахождением оптимального компромисса пропускной способности канала между узлами и загрузки этих узлов.

Надежность передающей среды осуществляется на двух уровнях – контроль актуальности и контроль защищенности информа-

ции. Соблюдение подлинности информации, как правило, обеспечивается за счет избыточности применением различных алгоритмов кодирования, предоставляющих на выходе, как правило, данные и соответствующую им контрольную сумму. Для защиты информации в передающей среде многие современные реализации используют готовые решения, соответствующие требованиям стандартов, такие как Kerberos.

Подсистема обработки информации. Для обеспечения прозрачности вычислительная система должна предоставлять пользователю интерфейс, имитирующий локальную систему. Наиболее традиционным подходом к решению этой задачи является предоставление пользователю приложения виртуальной вычислительной системы. По аналогии с традиционными системами, вычисления в распределенных системах производятся вызовом соответствующих подпрограмм. Механизм, обеспечивающий удаленный вызов подпрограмм (процедур) получил название RPC (Remote Procedure Call). Из-за отсутствия общей памяти параметры подпрограмм всегда передаются через значение, а процесс представления объекта/структуры в виде потока данных получил название «сериализация». Теоретическое описание вызова удаленных процедур можно найти в [1, 2].

Вычисления в распределенной системе принято рассматривать иерархически, в виде графа. Вершины такого графа – узлы системы, а дуги – связи в передающей среде. Традиционно, запрос разделяется на независимые подзадачи (по аналогии с параллельной вычислительной системой) и выполняются независимо на отдельных узлах, позволяя им работать автономно с минимальным знанием окружающей среды. Основные задачи оптимизации – состоят в создании наиболее эффективного пути обхода полученного графа. Автономная работа узлов снижает зависимости в системе, тем самым упрощая задачу продолжения работы после отказа на определенном узле. Основные проблемы, которые необходимо учитывать при построении распределенных алгоритмов это: атомарность операций, исключение цикличности операций, поддержка согласованности вычислений, взаимное исключение и некоторые другие (подробный список можно найти в [5]). Решение этих проблем, так или иначе, приводит к частичному решению проблем оптимизации и

конкурентной обработки данных. Особую роль распределенные алгоритмы обработки информации играют при обработке больших объемов многомерных научных данных [9].

Распределенные системы, как и все информационные системы, проектируются и реализуются в зависимости от специфик предметной области. Верная и своевременная декомпозиция на подсистемы, и оценка их важности для системы в целом являются залогом к нахождению компромисса между эффективным решением поставленных задач и сложностью реализации. Не каждая система строится с упором на все три компонента: так torrent-протокол не включает компонент обработки данных и слабо учитывает проблемы подсистемы хранения, однако эффективно решает задачи передачи данных. Распределенные файловые системы, напротив, решают преимущественно задачи хранения, уделяя меньше внимания обработке и передаче информации. Наиболее естественным образом подсистемы распределенной обработки, передачи и хранения данных сочетаются в распределенных СУБД, таких как SciDB [8], которым в настоящее время уделяется всё большее внимание [6].

Большинство проблем при проектировании компонентов может решаться с применением нескольких подходов. В статье описаны наиболее общие из них, нашедшие применение в реальных системах. Каждый из предоставленных подходов имеет ряд достоинств и недостатков, оставляя возможность поиска новых решений или оптимальных комбинаций старых, соответствующих требованиям исследуемой области.

Литература

1. *Н.А.Олифер, В.Г.Олифер*. Сетевые операционные системы. – СПб: Питер, 2003. – с. 544.

2. *Э. Таненбаум, М. ван Стенн.* Распределенные системы. Принципы и парадигмы.- СПб.: Питер. 2003. – 877 с. – (Серия «Классика Computer Science»).

3. Google Code University’s introduction to distributed system design. – URL: http://code.google.com/intl/ru-RU/edu/parallel/dsd-tutorial.html (дата обращения: 14.09.2011).

4. *Paul Krzyzanowski*. Online course: Distributed Systems: Concepts and Design. – Rutgers University.

5. *Rachid Guerraoui, Luis Rodrigues*. Introduction to Distributed Algorithms. – New York: Springer-Verlag, 2004. – 238 с.

6. *С.Д. Кузнецов.* Год эпохи перемен в технологии баз данных // Труды Института системного программирования РАН. 2010. Т. 19. С. 9-33.

7. *А.В. Смирнов, Д.Е. Андрианов.* Особенности распределения данных в многомерной СУБД SciDB. // Алгоритмы, методы и системы обработки данных. 2009. №14. С.158-163.

8. *M. Stonebreaker, R. Simakov, P. Velikov* [и др]. A demonstration of SciDB: a Science-Oriented DBMS. // VLDB'09: Proceedings of the 2009 VLDB Endowment. 2009.

9. *M. Stonebreaker, P. Velikov, R. Simakov, A. Smirnov* [и др]. Overview of SciDB: large scale array storage, processing and analysis. // Proceedings of the ACM SIGMOD International Conference on Management of Data, 06.06.2010-11.06.2010, Indianapolis, IN. P. 963-968.

E-MAIL: SMYATKINMAXIM@GMAIL.COM