УДК 004.75, 004.772, 004.62

ОРГАНИЗАЦИЯ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ УЗЛОВ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

Н.М. Лукьянов, А.М. Дергачев

Описывается порядок взаимодействия узлов распределенной системы хранения данных, входящей в состав Интернет сервисов, а также методы обработки информационных потоков, применяемые для организации взаимодействия локальных сетевых сервисов каждого узла. Рассмотрена последовательность движения пользовательских данных внутри системы, описан метод выбора оптимального узла хранения с использованием ранжирования весовыми коэффициентами, а также описана процедура восстановления данных узла после программного или аппаратного сбоя в работе системы.

Ключевые слова: хранилище, распределенные, сетевые, данные, алгоритмы, Интернет, сервис, модуль.

Введение

Данная работа является продолжением публикаций по результатам исследований в области обработки и хранения данных в распределенных системах, являющихся составной частью современных Интернет-сервисов. В предыдущих публикациях рассматривались результаты всесторонних исследований распределенных хранилищ данных, приводился анализ качественных и количественных показателей их функционирования. В ходе исследований, лежащих в основе работы, основное внимание уделялось вопросам производительности системы. Были построены имитационные модели, просчитаны необходимые объемы и структура хранилища, особое внимание уделено надежности данных. Просчитана вероятность выхода из строя носителей информации в процессе их эксплуатации, а также рассмотрены способы резервирования дискового пространства системы [1]. В результате исследований был построен и испытан прототип проектируемой распределенной системы хранения данных, которая была использована в открытой социальной сети студентов факультета Высшей школы менеджмента СПбГУ. Наблюдение за работой системы в реальных условиях позволило более тонко настроить алгоритмы работы с данными, а также предложить более продвинутые способы обработки информации.

Постановка задачи

Задачей распределенной системы хранения данных как составного элемента централизованного Интернет-сервиса является хранение программ и данных и предоставление доступа к ним по мере необходимости со стороны неограниченного количества сетевых пользовательских сервисов. Следовательно, подобные распределенные системы должны быть хорошо масштабируемыми и иметь открытую гибкую архитектуру, что, в свою очередь, требует единого подхода к организации работы с данными во всех узлах распределенной системы, начиная от приема и сохранения данных и заканчивая предоставлением необходимых данных потребителю. Применение вертикального масштабирования аппаратных средств обработки и хранения данных, а также применение систем хранения данных, используемых в корпоративных информационных системах, не дает приемлемого соотношения цены и производительности при использовании этих подходов для организации информационно-емких Интернет-сервисов [2]. В свою очередь, решения на базе многомашинных комплексов с применением несложных аппаратных компонентов могут обеспечить необходимый уровень открытости и гибкости, а также являются легко масштабируемыми как в рамках распределенной системы хранения данных, так и в рамках построения распределенной системы управления сетевыми сервисами в целом.

Архитектура системы

Любую распределенную систему хранения и обработки данных можно рассматривать как совокупность распределенных программно-аппаратных компонентов. В данном случае такими аппаратными компонентами являются отдельные серверы, объединенные локальной вычислительной сетью. Программная составляющая реализации системы является наиболее важной и несет в себе всю функциональную нагрузку системы. Организация взаимодействия отдельных узлов распределенной системы хранения данных опирается на сервис-ориентированную архитектуру. Каждый сетевой сервис состоит из ряда модулей (рис. 1), определяющих его функциональность, а также функциональное назначение узла распределенной системы хранения данных, на котором размещен данный сетевой сервис.

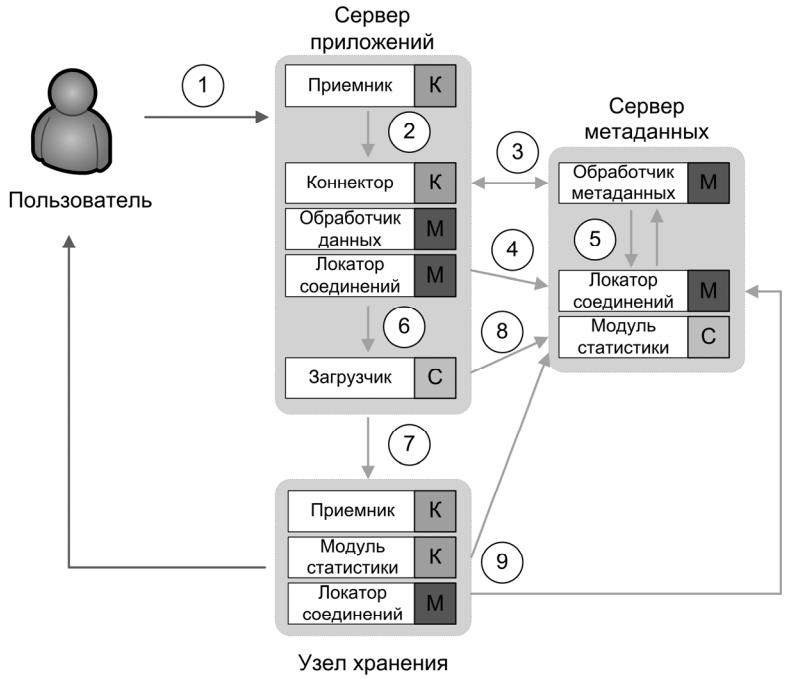

Рис. 1. Программные модули обработки данных

Условно всю распределенную систему хранения данных можно разделить на пять составляющих: пользователь, сервер приложений, сервер метаданных, узлы хранения и сервер производительности (на рисунке не показаны). Подобный подход является гибридным и использует составляющие всех трех разновидностей файловых систем: параллельных, симметричных и с API доступом [1]. Взаимодействие пользователя с системой происходит через сервер приложений, производящий прием и обработку данных для хранения, а также непосредственно с узлами хранения, предоставляющими пользователю ранее сохраненные данные. Алгоритмы и способы работы с информацией, применяемые в представляемой распределенной системе хранения данных, схематично изображены в виде модулей, которые подразделяются по своему типу:

– клиентские модули (К) являются сборщиками информации для ее сохранения в базе данных;
– серверные модули (С), основываясь на данных от клиентских модулей, выполняют заранее определенные действия;
– обособленные модули (М) похожи на серверные модули, но для работы не требуют клиентской части.

Отдельные алгоритмы, реализованные в виде модулей, позволяют обрабатывать разнородную информацию в зависимости от ее типа, среды передачи и условий внутри системы хранения. Происходит интеллектуальная, а не конвейерная обработка данных. Благодаря модульности, формализовав правила ввода/вывода данных, появляется возможность модернизировать последовательность обработки информации путем добавления новых модулей. Наиболее важными модулями являются модули статистики и загрузки/приема данных.

Модуль статистики

В процессе эксплуатации все узлы хранения периодически сообщают системе о своих нагрузочных параметрах. Их модуль статистики каждые три минуты передает в систему следующие характеристики: количество обработанных запросов, среднее время обработки запроса, нагрузка на центральный процессор с момента последнего опроса, загрузка канала передачи данных и объем свободного места на дисках. На стороне узла совершаются лишь простые действия по обработки статистических данных. Затем данные упаковываются и передаются серверу метаданных.

Серверная часть модуля статистики, выполняющая большую часть операций по обработке статистических данных, располагается на сервере метаданных. Основными параметрами, влияющими на быстродействие серверов, являются:

1. коэффициент нагрузки $k_l = \frac{q_{\text{запр}} \times t_{\text{ср}}}{t_{\text{пер}}}$, где $q_{\text{запр}}$ – количество запросов; $t_{\text{ср}}$ – среднее время обработки; $t_{\text{пер}}$ – период опроса;
2. нагрузка процессора;
3. нагрузка канала;
4. свободное место на диске.

Первые три параметра обрабатываются путем присвоения каждому из них весовых коэффициентов, с учетом которых далее рассчитывается общий индекс [3]. Весовые коэффициенты, применяемые в настоящей методике, приведены в таблице.

Параметр	Весовой коэффициент
Коэффициент нагрузки	0,75
Нагрузка процессора	0,1
Нагрузка канала	0,15

Таблица. Весовые коэффициенты параметров быстродействия

Таким образом, путем выставления рейтинга составляется упорядоченный список узлов, готовых к приему и предоставлению данных пользователю. Именно согласно этой таблице модуль загрузки на сервере приложений размещает данные на доступных узлах хранения. В случае отсутствия отчета от узла хранения за положенный промежуток времени считается, что сервер утратил связь с системой – вышел из строя или сильно загружен. Тогда этот узел изымается из списка до тех пор, пока не будет восстановлена его нормальная работа. В случае уменьшения лимита свободного пространства на дисках узла хранения до 10% узел автоматически переводится в режим «только чтение» и участвует в выборе подходящего для пользователя узла хранения только для чтения данных. Лимит в 10% введен для возможного использования сильно загруженных узлов в случаях крайней необходимости, например, при отсутствии свободного места на других узлах системы, а также для записи служебных данных при перемещении информации с одного узла хранения на другой.

Загрузка/прием данных

Загрузку данных в систему можно условно разделить на три этапа: прием данных от пользователя, обработка данных и определение узлов хранения и загрузка данных на узлы. Прием данных от пользователя осуществляется по HTTP протоколу через сервер приложений. Затем с использованием данных о состоянии узлов хранения, предоставляемых сервером статистики, определяются три узла хранения, на которые будут загружены три копии данных, полученных от пользователя. Обработчик метаданных генерирует уникальный строковый идентификатор файла, по которому можно будет однозначно определять его независимо от того, на скольких серверах он находится. Модуль загрузки поочередно соединяется с узлами хранения. Для аутентификации на узлах хранения используются специальные ключи безопасности, которые позволяют однозначно определить сервер метаданных. После успешной аутентификации файл загружается, и загрузчик переходит к следующему узлу хранения. В случае отказа одной (или двух) из выбранных целей хранения выполнение задания все равно считается успешным. Информация о необходимости загрузки дополнительных копий файла передается серверу производительности в модуль загрузки, который в дальнейшем, аналогично модулю загрузки сервера метаданных, обеспечивает необходимый уровень повторяемости данных.

Локатор соединений

Локаторы соединений – это модули, реализующие интеллектуальный выбор узлов хранения, обеспечивающих наибольшее быстродействие обмена данными с конкретным пользователем. Локаторы соединений расположены на сервере приложений и узлах хранения. Статистические данные о скорости соединения пользователя с системой передаются во время загрузки данных на сервер приложений, а также при получении пользователя данных с узла хранения. Вначале для получения данных пользователь направляется на один из наименее загруженных узлов хранения. Данные о скорости загрузки направляются на сервер метаданных. В процессе работы пользователя с системой собирается статистика о скорости соединения его IP адреса с различными узлами хранения, составляется рейтинг скорости загрузки. В дальнейшем, при формировании ответа пользователю, происходит выборка не только наименее загруженного узла хранения, но и имеющего наилучшие скоростные характеристики обмена данными для данного пользователя [4].

Восстановление узла данных

Не менее важным моментом в работе всей системы является последовательность восстановления данных на узлах.

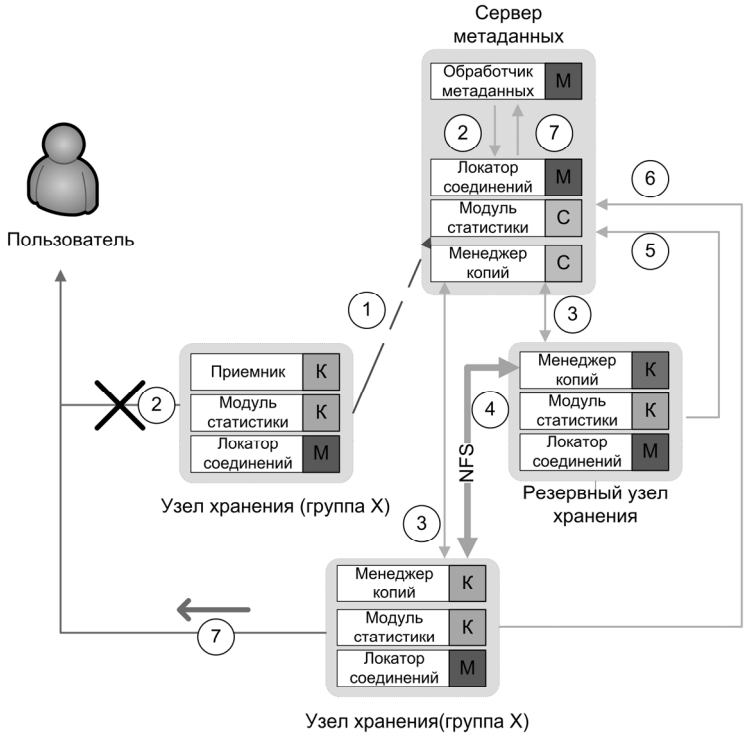

Рис. 2. Последовательность операций восстановления узла

В случае выхода из строя одного или двух узлов хранения данных, находящих в одной группе, необходимо, чтобы система автоматически или с минимальным участием администратора восстановила свою нормальную работу. На рис. 2 представлена последовательность операций в системе в случае выхода из строя одного из узлов группы X, состоящей из трех узлов хранения с одинаковым набором данных.

В случае отсутствия отчетов модуля статистики от узла хранения принимается решение о его недееспособности, обработчик метаданных изымает узел из списка работоспособных, а затем инициируется процесс дублирования потерянного узла. Для этого менеджер копий на сервере метаданных выбирает из списка пустой резервный узел и запускает автономный процесс копирования данных между одним из узлов группы X и выбранным пустым узлом по протоколу NFS v4, что позволяет избежать участия сервера метаданных в этом процессе, однако требует высокоскоростного и защищенного соединения. По завершению копирования новый узел группы через модуль статистики сообщает о своей готовности и работоспособности, после чего узел заносится в список доступных для использования узлов [5].

Заключение

Применение сервис-ориентированной архитектуры в программной части реализации распределенной системы хранения данных позволяет отказаться от дорогостоящих аппаратных систем хранения данных при организации информационно-емких Интернет-сервисов. Кроме того, описанная выше идеология восстановления работоспособности узлов хранения данных позволяет легко масштабировать ресурсы системы, не уступая при этом в вопросах высокой доступности и надежности дорогостоящим многомашинным вычислительным комплексам. Гибкость и простота модульной системы позволяет наращивать функциональность как отдельных компонентов, так и всей системы в целом прямо в процессе эксплуатации. Так, уже на этапе эксплуатации распределенной системы хранения данных начал внедряться новый модуль – «локатор соединений», реализующий интеллектуальный выбор доступных зеркал данных на различных узлах хранения.

В ходе работы с помощью языков программирования были реализованы ранее недоступные аппаратные реализации алгоритмов резервирования данных. Разработан алгоритм выбора оптимального узла хранения, основанный на методе ранжирования с весовыми коэффициентами. Рассмотрена последовательность движения пользовательских данных внутри системы. Благодаря использованию внутреннего высокоскоростного соединения узлов хранения стало возможным переносить данные с узла на узел за минимально возможное время – 1500 Мбайт за 7,2 часа, что в три раза превосходит среднюю скорость восстановления RAID массива уровня 5.

В дальнейшем, по завершению всей запланированной функциональности, распределенная система хранения данных будет интегрирована с одной из популярных систем управления содержимым. Также в ближайшее время будет опубликован код системы, что позволит осуществлять совместную с открытым сообществом работу над улучшением системы, а также сделает доступным распределенную систему хранения данных любому желающему.

Литература

1. Лукьянов Н.М. Анализ факторов, влияющих на качественные и количественные показатели функционирования систем распределенного хранения данных // Научно-технический вестник СПбГУ ИТМО. – 2008. – № 56. – 9 с.
2. Hoff T. Google Architecture // HighScalability.com. – 2008. [Электронный ресурс] – URL: http://highscalability.com/google-architecture (дата обращения: 02.09.2010).
3. Корников В.В. Байесовская модель обработки нечисловой, неточной и неполной информации о весовых коэффициентах. – Санкт-Петербургский государственный университет, 2000 [Электронный ресурс] – URL: http://www.inftech.webservis.ru/it/conference/scm/2000/session3/kornikov.htm (дата обращения: 02.09.2010).
4. Vogt M., Troup J. Dynamic Mirror Decisions // OOO Canonical – 2006 [Электронный ресурс] – URL: https://wiki.ubuntu.com/DynamicMirrorDecisions (дата обращения: 02.09.2010).
5. Yokota H. A proposal of DNS-based adaptive load balancing method for mirror server systems and its implementation // Конференция Advanced Information Networking and Applications – США: IEEE Computer Society, 2004. – 208 с.

Лукьянов Николай Михайлович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, nikolay.lukianov@gmail.com

Дергачев Андрей Михайлович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, ст. преподаватель, dam600@gmail.com