УДК 004.04

АРХИТЕКТУРА РАСПРЕДЕЛЕННОГО ХРАНЕНИЯ И ОБРАБОТКИ МАЛЕНЬКИХ ФАЙЛОВ

© Е.С. Чиркин, Н.Л. Королева, В.П. Дудаков

Ключевые слова: распределенное хранение данных; параллельная обработка; большое количество маленьких файлов.

Описано решение проблемы надежного хранения и быстрой обработки огромного количества маленьких текстовых файлов в информационно-поисковой системе «ИТ-специалист» с помощью разработанной распределенной системы хранения и обработки данных.

ВВЕДЕНИЕ

В процессе работы над информационно-поисковой системой «ИТ-специалист» возникла задача хранения и обработки большого количества маленьких файлов. Причем при хранении должна существовать возможность параллельной обработки информации на нескольких компьютерах, образующих «кластер» (о причинах взятия слова «кластер» в кавычки – ниже). Для упрощения работы и увеличения прозрачности системы, а также согласно специфическим требованиям (см. ниже), было разработано решение – распределенный комплексный сервис, похожий на распределенные файловые системы.

ХРАНИМЫЕ ИЛИ ОБРАБАТЫВАЕМЫЕ ДАННЫЕ

1. В ходе работы потребовалось хранить и использовать большое количество маленьких файлов (вебстраницы и их текстовые представления), каждый из которых имеет размер в 50–100 кб, но их суммарное количество огромно – порядка миллиарда в тестовой базе.
2. В проекте необходимо хранить некоторое количество «средних» файлов – размером 2–10 Мб.
3. В системе в некотором количестве представлены большие файлы – размером 10–200 Гб, некоторые из них непрерывно используются.

ПРОБЛЕМА

1. Впервые проявившись в процедурах резервного копирования, проблема заключается, в основном:

а) в значительных, на уровне файловой системы, накладных расходах на хранении большого количества файлов;

б) в значительном времени чтения оглавления каталога (причем, на данную операцию нет возможности повлиять – в каждом конкретном случае она является приблизительно константой).

2. Значительный объем обрабатываемой информации потребовал (хотя бы эпизодически) необходимости обработки информации на отдельных компьютерах гетерогенного «кластера», каждый из которых, условно, мог выйти из строя (временно или насовсем) в любой момент времени.

Слово «кластер» в данном случае употребляется в кавычках, т. к. традиционно (архитектуры, похожие на MPI [1]) в кластере каждый компьютер является значимым для процедуры расчета, и сбой или выход из строя даже одного задействованного компьютера приводит к аварийному завершению всего расчета.

ТРЕБОВАНИЯ К СИСТЕМЕ ХРАНЕНИЯ ДАННЫХ

1. Сохранность данных важнее быстродействия и времени отклика.
2. Гетерогенный «кластер» с высокой вариативностью составляющих образуется всеми доступными компьютерами, входящими в него. Компьютеры, входящие в него, потенциально должны сохранять возможность их использования пользователями.
3. В системе хранится большое количество маленьких файлов, каждый из которых может потребоваться каждому из компьютеров «кластера».
4. Некоторое количество программ обработки написано на разных языках программирования.
5. «Кластер» построен из обыкновенного оборудования, эксплуатируемого пользователями. Следовательно, сбои, потери отдельных файлов, недоступность узлов кластера – норма. На функционировании системы это не должно отражаться.
6. Как правило, наиболее часты следующие операции: чтение маленького файла, создание маленького файла, чтение линейного фрагмента большого файла. Создание, запись, перезапись, дозапись, удаление – операции сравнительно редкие, «создание» подразумевает, что на момент необходимости записи в файл известен его будущий объем.
7. Ввиду специфики разрабатываемой системы (узкоспециализированный поисковый движок), по окончании первичной обработки хранение почти всех маленьких файлов носит архивный характер (доступ к ним осуществляется исключительно редко и по необходимости).
8. Возможна ситуация конкурентного доступа на чтение и запись множества клиентов к одному файлу.

Требуется исполнять данные запросы строго последовательно.

9. Доступ к файлам осуществляется либо только на чтение, либо только на запись (плюс блокировки совместного доступа). Доступа типа чтение/запись не существует.

10. Хранение файлов должно быть иерархическим (при помощи каталогов) в глобальном пространстве имен.

11. Иногда требуется сохранить «снимок» ряда файлов (т. е. метаинформацию и содержимое файлов) на момент создания снимка, сами же файлы или каталоги могут быть впоследствии изменены или удалены. Удаление «снимка» должно повлечь за собой удаление всей информации, связанной только с этим снимком.

12. Сбой в работе сети, мастер-сервера, сервера хранения данных или клиента, не затронувший хранимые данные и метаданные, не должен влиять на работоспособность системы в целом, после устранения неполадок работа должна продолжаться с прерванного места.

Требования, предъявляемые к решению, напоминают проблемы, решаемые некоторыми существующими системами вроде GoogleFS [2] (решение закрытое), HDFS [3] (решение открыто, но не удовлетворяет требованиям), DFS [4] (решение привязано к архитектуре Windows Server и не удовлетворяет требованиям).

АРХИТЕКТУРА РЕШЕНИЯ

Собственно, очевидным, и, возможно, самым эффективным решением было бы помещение всех файлов в базу данных с поддержкой транзакций [5]. К сожалению, во-первых, это катастрофически снижает надежность всей системы и значительно повышает требования по быстродействию и устойчивости оборудования, а во-вторых, это никак не страхует от сбоев на носителе и файловой системе – требуется определенная процедура резервного копирования, осложняемая размером базы данных.

1. В процессе решения поставленных проблем в первую очередь была предпринята попытка избавиться от мелких файлов. Очевидно решение – помещение всех файлов в файл-контейнер и обращение к его содержимому по внешнему индексу. В качестве контейнера был выбран формат ZIP. При первичной обработке новых мелких файлов они обслуживаются или напрямую, или помещаются в отдельный ZIP-контейнер без сжатия. Отсутствие сжатия позволяет напрямую обращаться к любым фрагментам ZIP-контейнера. По окончании первичной обработки файлы из ZIP-контейнера фоновым потоком пережимаются с большей степенью сжатия (по умолчанию – максимальной) и становятся «архивными» данными. Все разработанные вспомогательные процедуры абсолютно прозрачны во внутреннем API и позволяют использовать ZIP-файлы любого формата и любой степени сжатия.

Второй этап – увеличения размера кластера на файловой системе. Было указано максимальное значение в 64 кб (против стандартного 4 кб). В результате размер таблицы размещения файлов пропорционально уменьшился в 16 раз, быстродействие возросло, потребление оперативной памяти системой также сократилось приблизительно на порядок.

В системе хранения выделены следующие элементы (рис. 1).

Рис. 1. Архитектура разработанного решения

Мастер-сервер – хранит метаинформацию обо всех файлах. Фактически, представляет собой серверное приложение – клиент для СУБД MySQL.

На мастер-сервере для каждого файла, дополнительно к обыкновенной метаинформации, хранится информация, в каком количестве экземпляров должен храниться каждый файл.

Сервера хранения – хранят собственно файлы. Для каждого файла существует главный сервер хранения (назначается мастер-сервером), с которого и осуществляется репликация на остальные сервера хранения по количеству необходимых копий каждого файла (количеству реплик). Обычно используются три реплики.

Мастер-сервер содержит физическую (задается вручную администратором) карту серверов хранения, которая включает физическое расположение, организацию каналов связи и пропускную способность сети (ведется работа над процедурами автоматической оценки последней). Это позволяет минимизировать риски повреждения и доступности хранимых файлов.

Значительное количество файлов, как было сказано ранее, хранится в ZIP-архивах. Клиентам доступ к их содержимому прозрачно организуется серверами хранения.

Клиент взаимодействует с мастер-сервером только для выполнения операций, связанных метаданными. Все остальные операции выполняются с серверами хранения напрямую.

МАСТЕР-СЕРВЕР

В системе используется только один мастер-сервер, что значительно упрощает архитектуру. Нагрузка на мастер-сервер сравнительно невелика, что позволяет:

– в реальном времени контролировать доступность и загруженность подчиненных серверов хранения. Собирать соответствующую статистику;

– в фоновом режиме производить операции обслуживания хранилища – проверку целостности отдельных файлов и файлов-контейнеров, удалять файлы, отмеченные для удаления, удалять лишние экземпляры файлов либо восстанавливать недостающие;

– организовывать, при необходимости, «снимки» файловой системы любой сложности.

База данных мастер-сервера реплицируется (подразумевается, средствами самой СУБД MySQL) на необходимое количество реплик. Приложение-сервер, взаимодействующее с клиентами, способно выбирать и назначать мастером соответствующий доступный экземпляр СУБД.

Один из ключевых параметров собираемой статистики – это надежность сервера хранения, которая состоит из четырех простых предположений:

а) лучше других тот сервер, который локально во времени чаще доступен, что следует понимать, что данный сервер существует, работоспособен, включен и не используется;

б) лучше других тот сервер, который глобально во времени меньше используется пользователями, – предположение основано на эмпирическом наблюдении, что по сравнению с другими факторами непреодолимого характера, в неработоспособности средней компьютерной системы виноваты обычно только ее пользователи;

в) лучше других тот сервер, который глобально во времени был чаще доступен, – предположение, почти напрямую связанное с предыдущим пунктом, – вероятность того, что редко используемая компьютерная система выйдет из строя при сохранении режима ее использования, низка;

г) лучше других тот сервер, который локально во времени меньше используется, – опять же, прямая отсылка к эксплуатации системы пользователями.

По этим критериям формируется условный рейтинг, на основании которого определяется, какие доступные серверы следует эксплуатировать в данный конкретный момент.

«Снимки» состояния файловой системы формируются на уровне транзакции в СУБД.

МЕТАИНФОРМАЦИЯ

Метаинформация, хранимая на мастер-сервере, включает в себя:

– ID файла, его контрольные суммы и хеш;
– имя файла, полный путь к нему;
– размер;
– атрибуты создания, последней модификации;
– признак удаления (файл не удаляется сразу, только в процессе обслуживания);
– статус блокировки (используется модель оптимистичной блокировки);
– атрибуты доступа (поддерживается ролевая модель разграничения доступа);
– количество хранимых копий;
– главный сервер хранения для данного файла.

СЕРВЕР ХРАНЕНИЯ ДАННЫХ

При старте – собирает информацию обо всех хранимых на нем файлах и передает ее мастер-серверу, отвечает на прямые запросы через API – на собственно отдачу и сохранение данных. Ведется журнал выполненных операций. При отсутствии загруженности в течение некоторого промежутка времени сервер хранения данных начинает процедуру проверки количества копий файлов, сначала – тех, для которых он отмечен как главный сервер хранения, потом – всех остальных. Если для какого-то файла количество его копий упало ниже минимального допустимого, инициируется процедура его дублирования. Если количество копий превышает необходимое количество, мастер-северу посылается специальное уведомление, по которому он начинает удалять неиспользуемые экземпляры файла с наименее надежных, согласно текущему рейтингу, компьютерных систем.

КЛИЕНТ

Для большинства клиентов назначено их положение на физической карте, хранимой на мастер-сервере (аналогично карте серверов хранения).

Например, процедура чтения специфичным клиентом файла выглядит следующим образом (имя файла известно).

Клиент подключается к мастер-серверу и получает у него имена серверов хранения данных и идентификаторы файлов, к которым следует обращаться за получением файла в порядке убывания приоритета: сначала – рекомендуемый – самый близко расположенный, потом – более удаленный и т. п., согласно физическому распо-

ложения и загруженности серверов и сети. При формировании данного ответа мастер-сервер учитывает результаты последней проверки доступности серверов хранения.

ЗАКЛЮЧЕНИЕ

Разработанное решение покрывает потребности системы в обработке данных как при регулярной эксплуатации, так и при экстренных нагрузках и приближается к теоретическому пределу использования оборудования с учетом сформулированных ранее ограничений. К уникальной особенности решения следует отнести ее способность эффективно обрабатывать и хранить большое количество маленьких файлов.

ЛИТЕРАТУРА

1. MPI: A Message Passing Interface Version 3.0 // Message Passing Interface Forum. 2012. September 21. URL: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (accessed: 30.09.2013).
2. *Ghemawat S., Gobioff H., Leung S.-T.* The Google File System. URL: http://research.google.com/archive/gfs.html (accessed: 30.09.2013).
3. Yahoo! Hadoop Tutorial. URL: http://developer.yahoo.com/hadoop/tutorial/index.html (accessed: 30.09.2013).
4. DFS Namespaces and DFS Replication Overview. 2013. June 24. URL: http://technet.microsoft.com/library/jj127250.aspx (accessed: 30.09.2013).
5. *Кузнецов С.Д.* Основы баз данных. М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007.

БЛАГОДАРНОСТИ: Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 12-07-00512).

Поступила в редакцию 8 октября 2013 г.

Chirkin E.S., Koroleva N.L., Dudakov V.P. ARCHITECTURE OF DISTRIBUTED STORAGE AND PROCESSING OF SMALL FILES

This article describes safe solution to the problem of storage and fast processing of large numbers of small text files in a retrieval system, “IT Expert” with the help of a distributed storage system and data processing.

Key words: distributed data storage; parallel processing; a large number of small files.