электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

Концепции разработки и организации системы прозрачного доступа к файлам больших объемов и RAID-массивам в распределенной разнородной среде

77-48211/637968
09, сентябрь 2013

Виноградова М. В., Белоусова В. И., Мельник В. Н.

УДК: 004.63

Россия, МГТУ им. Н.Э. Баумана
vinogradova.m@bmstu.ru
belousova.vi@gmail.com
vasily.mln@yandex.ru

Введение

Для длительного хранения больших объемов неструктурированных данных используют дисковые накопители, работающие под управлением файловых систем, которые позволяют организовывать иерархии файлов и каталогов, хранить сведения об атрибутах и правах доступа. Для повышения надежности хранения используются RAID-массивы, обеспечивающие параллельное чтение и запись файлов и восстановление информации после сбоев. Современные файловые системы позволяют прозрачно работать с RAID-массивами и организовывать к ним сетевой доступ с возможностью аутентификации и разграничения прав. Однако, если файловое хранилище содержит несколько дисковых массивов, работающих под управление разнотипных файловых систем, то возникают проблемы скорости поиска и согласования имен файлов и каталогов.

По этой причине часто появляется необходимость в создании файлового архива, который обеспечивает для своих пользователей – людей или внешних программ - прозрачный доступ к хранилищу файлов, состоящему из нескольких сетевых дисков с различными файловыми системами.

На основе анализа информационных систем, использующих файловые архивы, были определены их общие черты. Это системы с общим хранилищем файлов, к которому обращаются пользователи различных, возможно, унаследованных подсистем. Загрузка и выгрузка файлов происходит со средней периодичностью. В качестве примеров можно привести автоматизированные системы управления (АСУ) вузов, предприятий и организаций, состоящие из независимых программных комплексов своих подразделений, которые обмениваются разнородными документами [1].

В результате исследования требований, предъявляемых к файловому архиву, были определены следующие критерии качества:

1. Скорость поиска файлов по мета-информации. Мета-информация о файлах должна быть представлена в виде, обеспечивающем наиболее быстрый доступ к данным.
2. Наличие утилиты или программного интерфейса (API) для работы с архивом. Наличие API или утилиты позволяет встраивать функции работы с архивом в сторонние приложения.
3. Скорость чтения и записи файлов. Архив должен обеспечивать высокую скорость чтения и записи.
4. Прозрачность обращения к файлам. Архив должен скрывать от пользователя физическое расположение файла. Путь, по которому пользователь обращается к файлу, должен минимально зависеть от физической иерархии каталогов в хранилище и физического расположения хранилища.

На основе выявленных критериев был проведен сравнительный анализ существующих технологий и средств, используемых при реализации файловых хранилищ, которые показал следующее:

1) Сетевая файловая система - позволяет работать с файлами, физически находящимися на разных машинах, прозрачно изменять физическое местоположение файлов. Недостатком сетевой файловой системы является отсутствие единого хранилища метаинформации, что существенно замедляет поиск по архиву [2].

2) Готовые решения по созданию файлового архива, например Apache Jackrabbit - предоставляют возможности по хранению и поиску файлов. Их недостатком является отсутствие удобного для разработки собственных приложений API и утилиты [3].

Поскольку средства операционных систем [4] и готовые решения недостаточно эффективны для создания архива файлов в распределенной системе, то возникает необходимость в выработке основных положений и концепций по созданию собственного файлового архива, который может быть встроен в существующую информационную систему организации или предприятия.

Целью данной статьи является исследование методов и технологий для создания системы файлового архива, с целью повышения эффективности его работы, а также разработка предложений по его реализации.

1 Решаемые задачи

Основными вопросами при проектировании и реализации файлового архива являются:

1. архитектура файлового архива;
2. технология и интерфейс доступа к файлам;
3. организация хранилища мета-информации;

4. поддержка транзакций при работе с архивом;
5. обеспечение целостности данных в архиве.

Далее рассмотрены предложения по решению каждого из поставленных вопросов.

2 Архитектура файлового архива

Прежде всего необходимо определить базовую концепцию файлового архива. Предлагается выделить два уровня: физический, обеспечивающий хранение файлов, и логический, реализующий внешнее представление архива. Это обеспечит прозрачность работы с файлами и возможность независимого изменения физической и логической структуры архива.

На физическом уровне архив состоит из нескольких сетевых дисков или RAID-массивов [5] с различными файловыми системами, работающих под управлением различных операционных систем. Количество сетевых дисков, их характеристики, а также операционные и файловые системы выбираются исходя из потребностей и возможностей конкретной организации, или используются существующие в ней [6].

На логическом уровне файловый архив представляет собой совокупность корневых каталогов, доступ к содержимому которых осуществляется посредством программного компонента. В процессе работы пользователи помещают файлы из своих локальных хранилищ в файловый архив и получают файлы из архива в локальные хранилища.

Для взаимодействия с архивом необходимо наличие приложения для работы оператора и библиотеки функций для обращения к архиву из внешнего приложения. На рисунке 1 приведена концептуальная схема файлового архива.

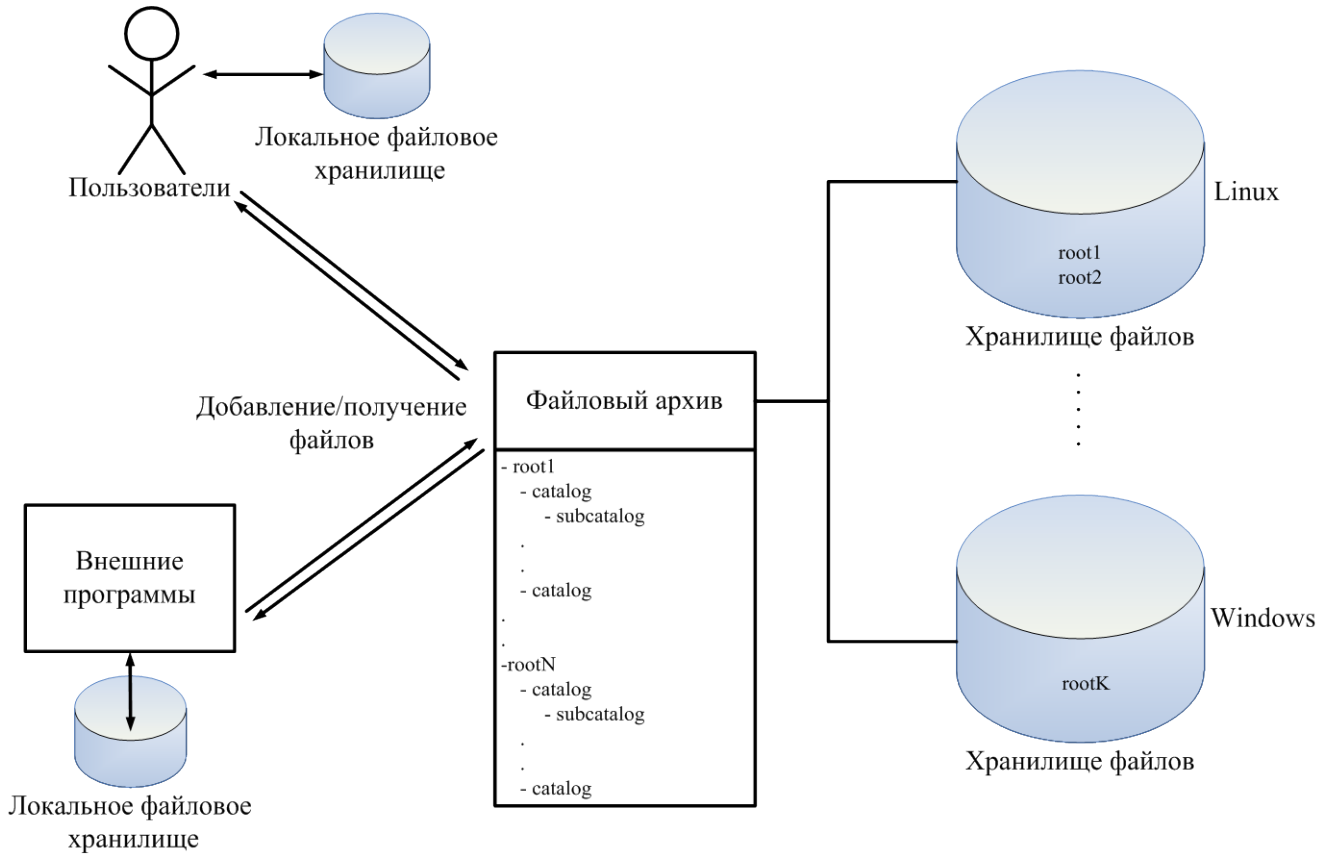

Рисунок 1 - Схема файлового архива

Предполагается, что корневые каталоги могут находиться в одном или разных разделах или дисках. К каждому корневому каталогу должен быть прописан путь и присвоен идентификатор (тэг). С использованием этого тэга должны формироваться все ссылки на файлы и каталоги. Для возможности физического перемещения каталогов архива вне программного комплекса при изменении физической структуры каталога с сохранением его логической структуры предлагается установить соответствие между логическими корневыми каталогами и физическими каталогами на дисках. При этом структура каталога будет соответствовать структуре каталога на диске.

3 Технология и средства доступа к файлам

При реализации программного комплекса файлового архива возникает вопрос о технологии доступа к файлам. Возможно обращение к файлам через стандартные средства операционной системы или через специализированные сервера, например, FTP-сервер. Рассмотрим эти варианты подробнее.

Использование сетевых дисков позволяет работать с удаленными хранилищами данных как с локальными дисками в синхронном режиме. Однако, данное решение накладывает

ограничения на программные платформы клиента и сервера, которые должны поддерживать технологию монтирования сетевых дисков.

Использование FTP-сервера позволяет работать с хранилищем через стандартный протокол, поддерживаемый всеми современными ОС. Также FTP позволяет скрыть от внешних систем конфигурацию дискового массива и обеспечивает прозрачную работы с RAID. Недостатком использования FTP является более сложная программная реализация взаимодействия клиента и сервера.

Согласно вышеизложенным доводам и общим требованиям к системе, предлагается использовать FTP-сервер при реализации файлового хранилища. Выбор конкретного сервера зависит от объема данных, количества пользователей и имеющихся ресурсов конкретной организации.

4 Хранение мета-информации

Структура базы данных и набор хранимой мета-информации зависит от потребностей и особенностей конкретной организации [7]. Рассмотрим общие вопросы создания базы данных файлового архива.

Для возможности хранить иерархическую структуру каталогов база должна иметь рекурсивный вид и содержать следующие атрибуты:

– данные о физических адресах (именах в локальной сети) массивов жестких дисков и соответствующие им имена корневых каталогов;

– иерархическая структура каталогов;

– имена файлов и их атрибуты, например, дата и время создания или перемещения в архив, размер файла, контрольная сумма;

– права доступа к файлам и каталогам.

Данная схема БД является плоской, с зависимостью от простого первичного ключа. Для оптимизации модели нужно воспользоваться соображениями удаления избыточности при хранении данных. Очевидно, что каждая запись в таблице соответствует либо каталогу, либо файлу, либо корневому каталогу. Поэтому воспользуемся идеей наследования объектов для «оптимизации» структуры [8].

При наследовании базовой структурой является узел файловой системы, описываемый тегом (или названием файла/каталога), ссылкой на родительский элемент и атрибутами файла/каталога. От него унаследованы (физически – связь 1:1) два объекта: корневой каталог и файл. Корневой каталог описывается абсолютным путем к каталогу и файловой системой. Файл описывается дополнительными атрибутами.

Такое разбиение позволяет единообразно работать с файлами и каталогами и избежать избыточности хранения данных. При создании нового объекта в системе в БД создается запись в таблице «Структура каталогов». Если создается файл, то создается дополнительная запись в

таблице «Файлы». Если создается корневой каталог, то создается дополнительная запись в таблице «Корневые каталоги». При такой организации данных происходит экономия объема дискового пространства примерно 30% размера одной записи.

Возможно альтернативная структура данных, при которой в таблицах «Файл», «Каталог» и «Корневой каталог» содержатся все их атрибуты. Корневой каталог связан с Каталогом связью 1:1, что аналогично идее предыдущего варианта. С точки зрения экономии дискового пространства данная схема аналогична предыдущей, но значительно упрощает работу с базой и позволяет создавать прямое объектное отображение таблиц.

Предлагаемая в данной работе структура данных БД файлового архива представлена на рисунке 2. Набор атрибутов в таблицах определен из перечня основных атрибутов объектов файловой системы и может быть расширен в конкретной реализации.

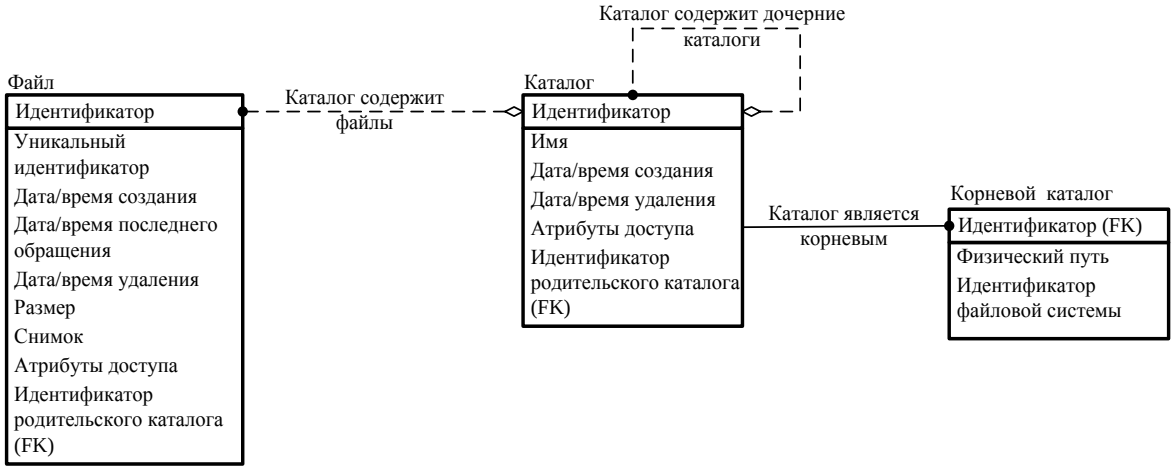

Рисунок 2 - Структура данных базы данных файлового архива

Данный вариант представляет собой объединение решения с выделенными таблицами для каждой логической сущности с полным набором атрибутов (Вариант 2) , с представленным ранее вариантом выделения в отдельные сущности только специфических атрибутов (Вариант 1). Обоснуем такой выбор тем, что при реализации Варианта 1, каждый запрос на получение файла или поиск файловой информации будет требовать естественного соединения двух таблиц. При выборе Варианта 2 мы устраняем этот недостаток, не создавая избыточности данных.

При этом корневой каталог все еще хранится, как совокупность обычного каталога и дополнительной информации. Это обусловлено тем, что при выделении корневого каталога в автономную сущность для его связи с дочерними каталогами в последних пришлось бы вводить дополнительную ссылку на корневой каталог, тем самым создавая избыточность данных и разрушая общую связь каталогов в рамках одного дерева.

Для идентификации записей БД предлагается использовать 128-битный GUID, который будет однозначно определять файл в архиве. Использование отдельного GUID, а не внутреннего идентификатора или последовательности позволит производить миграцию данных между различными базами без необходимости внешним пользователям обновлять идентификаторы файлов, с которыми они работают.

Для ускорения поиска и унификации именования файлов предлагается хранить метаданные файлового архива в базе данных, развернутой на отдельном сервере или группе серверов. В базе данных должны храниться атрибуты файлов, информация о физическом расположении различных частей файлового хранилища и структура дерева каталогов. Это решение позволяет ускорить поиск и навигацию по архиву за счет оптимизации выполнения запросов, выполняемой в СУБД.

Выбор конкретной СУБД для реализации базы данных зависит от конкретной организации, в том числе, от объема хранимых данных и имеющихся ресурсов [9]. Единственным требованием к ней является поддержка рекурсивных запросов. Этому требованию удовлетворяют как большие известные системы, например от Oracle или Microsoft, так и бесплатно распространяемые, например, PostgreSQL.

5 Интерфейс доступа к файловому архиву

Следующим вопросом при реализации файлового архива является разработка клиентского и серверных приложений. Рассмотрим возможные варианты реализации пользовательского интерфейса программного комплекса файлового архива:

1. **Web-приложение.** Данное решение является стандартным для приложений, обеспечивающих доступ к единой базе данных. Его достоинством является отсутствие необходимости разворачивать клиентское программное обеспечение на компьютере пользователей и использование интернет-браузера в качестве тонкого клиента. Также наличие развитых библиотек позволяет быстро разработать приложение для отображения содержимого базы данных. Недостатком является сложность работы с файлами больших объемов – для сохранения файла в архив требуется предать его на сервер приложения, а затем – на сервер FTP. То есть, осуществляется двойная передача данных.

2. **Толстый клиент.** Использование клиентского приложения в качестве толстого клиента имеет недостатки в виде необходимости раздельной установке на каждую рабочую станцию, работающую с файловым архивом. Однако, непосредственный доступ приложения к локальной файловой системе значительно ускоряет работу с файлами большого объема, что является важным для реализуемой системы.

Таким образом, более эффективным в качестве метода реализации интерфейса доступа к архиву является толстый клиент.

Базовая часть клиента будет состоять из утилиты и API, которые выполняют одинаковые функции и обращаются к разделу БД. Утилита — это программа, которая запускается другим процессом или человеком, принимает параметры из командной строки, выполняет некоторые функции и завершается. API — это библиотека функций (независимых или методов классов), которые вызываются другим процессом, принимают на вход набор параметров, выполняют действия и возвращают результат.

Возможны два варианта их взаимодействия:

– утилита и API дублируют друг друга и независимы;

– утилита обращается к API для выполнения функций.

Недостатком варианта 1 является дублирование функций, что усложняет модернизацию, модификацию и сопровождение модуля. Поэтому предлагается вариант 2. В этом случает API реализует основной функционал модуля, а утилита — интерфейс через командную строку и вызов функций API. Для снижения зависимостей между API и утилитой предлагается заменить прямые вызовы на обращение к интерфейсу [10].

Кроме утилиты и API клиент содержит графический интерфейс пользователя, который обеспечивает взаимодействие человека и программного комплекса. Основной задачей графического интерфейса является прием запросов пользователя, вызов функций API и отображение результатов их выполнения в удобном для человека виде.

Таким образом, структура файлового архива имеет вид, представленный на рисунке 3.

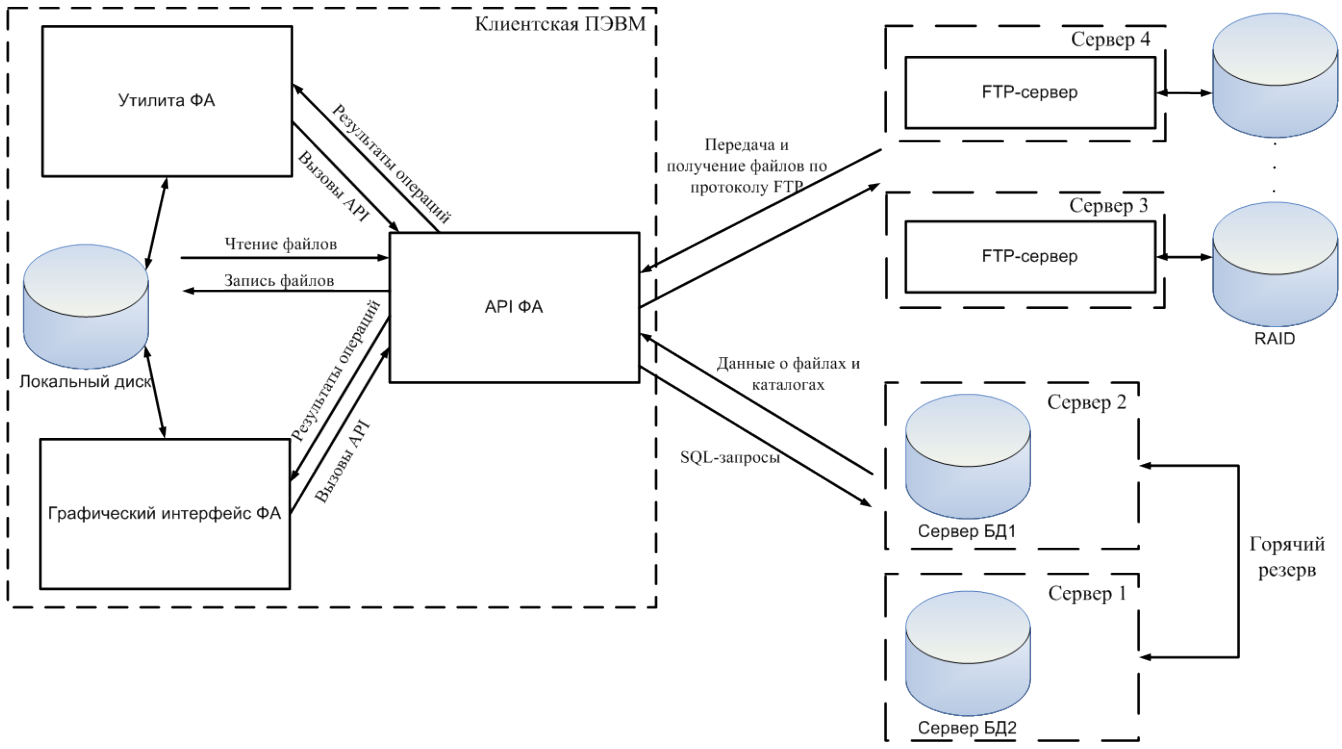

Рисунок 3 - Структура файлового архива

6 Базовый набор функций по работе с файловым архивом

На основе анализа функциональных требований к файловому архиву предлагается следующий базовый набор функций для реализации файлового архива:

– Создание корневого каталога - включение существующего сетевого диска в файловый архив;

– Создание дочернего каталога - создание подкаталога в существующем каталоге файлового архива;

– Запись файла в архив - запись существующего файла в один из каталогов файлового архива;

– Копирование файла - копирование файла внутри файлового архива;

– Копирование каталога (с подкаталогами и файлами) - копирование каталога и его содержимого внутри файлового архива;

– Перемещение файла - перемещение файла внутри файлового архива;

– Перемещение каталога (с файлами и подкаталогами) - перемещение каталога и его содержимого внутри файлового архива;

– Получение файла - копирование файла из файлового архива в указанное местоположение;

– Удаление файла - удаление файла из файлового архива;

– Удаление каталога (с файлами и подкаталогами) - удаление каталога;

– Получение списка корневых каталогов - получение списка имен (тегов) корневых каталогов;

– Получение содержимого каталога - получения списка имен подкаталогов и файлов данного каталога;

– Поиск файла по параметрам - поиск файла в файловом архиве по заданным параметрам;

– Поиск файла в каталоге - поиск файла в заданном каталоге по заданным параметрам;

– Поиск файловой информации по различным критериям;

– Получение списка одинаковых файлов - получение списка имен файлов, имеющих один или более дубликат;

– Поиск подобных файлов - список полных имен файлов, подобных указанному.

Алгоритмы большинства из перечисленных функций тривиальны. Внимания заслуживают вопросы синхронизации между содержимым базы данных и файловой системой и выявления дубликатов файлов.

7 Организация транзакций файлового архива

Целью создания транзакций является обеспечение отсутствия противоречий между информацией в БД и фактическим наличием файлов на жестком диске. Современные СУБД и файловые системы поддерживают собственные механизмы транзакций. В рамках разработки файлового архива требуется обеспечить взаимодействие между данными механизмами при операциях, связанных с изменением, удалением или созданием файлов.

Основной проблемой в данном случае является то, что уровни ФС и БД взаимодействуют через уровень абстракций. Для взаимодействия предложим следующий алгоритм:

– Уровень абстракций получает запрос на изменение файла.

– Уровень абстракции формирует объект AbstractTransaction с пустой коллекцией объектов, сброшенными признаками ошибки и пустыми именами транзакций.

– Уровень абстракции формирует коллекцию объектов, каждый из которых соответствует изменяемому файлу. Данная коллекция помещается в объект AbstractTransaction, который передается уровням взаимодействия с ФС и БД в качестве аргумента операции.

– Уровни ФС и БД формируют транзакции и присваивают им уникальные имена, которые заносятся в объект AbstractTransaction. Далее выполняются транзакции. В случае ошибки при выполнении транзакции в соответствующее поле признака заносится признак неудачи, в противном случае – признак успеха.

– Уровень абстракции ожидает, когда оба признака получат непустое значение. В случае, если хотя бы один из признаков содержит признак неудачи, уровням БД и ФС дается команда откатить транзакции. В противном случае дается команда накатить транзакции.

8 Алгоритм поиска дубликатов файлов

При выборе метода выявления одинаковых файлов следует рассмотреть известные алгоритмы. Критерии выбора (в порядке убывания важности):

– Скорость поиска дубликатов;

– Требуемый объем оперативной памяти;

– Требуемый объем жесткого диска БД для хранения дополнительной информации;

– Простота реализации.

Исходя из названных критериев, исключаем методы, основанные на побайтовом сравнении файлов или их частей по причине того, что сложность подобных методов определяется не только размеров вновь поступившего в систему файла, но и количеством файлов в системе.

Критерии требуемого объема жесткого диска и памяти позволяют остановиться на методах, основанных на хэш-функциях. Данные методы:

– имеют сложность поиска O(1) на стороне программы клиента, за счет извлечения единичной записи из БД, и O(log n) на стороне сервера СУБД, за счет построения индекса по значению хэш-функции;

– при вычислении хэш-функции файла наиболее распространенные алгоритмы последовательно обрабатывают блоки стандартной длины, т.о. получаем наименьший расход оперативной памяти на этапе вычисления хэш-функции;

– распространенные хэши имеют длину 64-128 бит, т.о. делая незначительными затраты на хранение «снимков» файлов в БД.

Рассмотрим следующие варианты вычисления хэш-функции:

Простое вычисление хэш-функции файла. Файл (вне зависимости от формата) представляется как бинарный поток, для которого при добавлении на диск считается значение хэш-функции, которое сохраняется в базу данных.

Достоинства:

– время поиска совпадающего файла не зависит от количества файлов, хранящихся в системе;

– наличие стандартных библиотек для вычисления хэш-функций.

Недостатки:

– для файлов большого размера вычисление хэш-функций занимает длительное время;

– гипотетическая вероятность коллизий.

Вычисление хэш-функции заголовка файла. Медиа-форматы файлов имеют заголовки, содержащие различную мета-информацию. Возможно хэширование только мета-инфомации.

Достоинства:

– уменьшает время вычисления хэш-функции для больших файлов.

Недостатки:

– на практике метаданные файлов часто удаляются (например EXIF для изображений). В файлах формата MP3 данные заголовка распределены по всей длине файла, что снижает ценность выбранного метода;

– требуется написание индивидуального кода для разных форматов файлов.

Можно сделать вывод, что экономичнее всего использовать стандартные хэш-функции, для вычисления снимка файла вне зависимости от его типа.

Предлагается использовать хэш-функцию MD-5, т.к. она имеет стандартную реализацию в виде утилиты Linux и низкую вероятность коллизий, что подтверждает использование её в качестве основы для создания «снимков» файлов в файло-обменных сетях.

Заключение

В результате проведенного исследования технологий создания файлового архива был разработан набор предложений по его реализации, который может быть использован при разработке конкретного программного комплекса.

Предложена концепция системы, реализуемой как уровень абстракции над файловыми системами, хранящий информацию о файлах в унифицированном виде, обеспечивающий быстрый

поиск и сравнение файлов, а также целостность данных. Данная система обеспечивает прозрачную работу с распределенным файловым архивом, позволяет совершать операции над файлами большого размера и предоставляет графический интерфейс, утилиту и API.

Список источников

1. Информационная управляющая система МГТУ им. Н.Э. Баумана «Электронный университет»: концепция и реализация / Агеева Т.И., Балдин А.В., Барышников В.А. и др. [под ред. Федорова И.Б., Черненького В.М.] - М.: Изд-во МГТУ им. Н.Э.Баумана. – 2009. – 376 с.
2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – СПб.: Питер. – 2003. – 544 с.
3. Официальный сайт проекта Apache Jackrabbit. Режим доступа: http://jackrabbit.apache.org (дата обращения 30.09.2013).
4. Черненький В. М., Шульмин А. С. Формализованная структурная модель операционной системы Windows 2000 // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2011. № 12 Режим доступа: http://technomag.bmstu.ru/doc/292997.html (дата обращения 20.10.2013).
5. Постников В. М., Спиридонов С. Б. Подход к расчету весовых коэффициентов ранговых оценок экспертов при выборе варианта развития информационной системы // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 08 Режим доступа: http://technomag.bmstu.ru/doc/580272.html (дата обращения 20.10.2013).
6. RAID массивы начального уровня. Режим доступа: http://www.oszone.net/3248/RAID (дата обращения 20.09.2013).
7. Виноградова М.В., Гжельская М.О. Технология проектирования баз данных на примере пропускной системы общежития // Проблемы построения и эксплуатации систем обработки информации и управления: Сб. Статей. Выпуск 8. М.:Издательство МГТУ им. Н.Э. Баумана, 2011. С. 55-63.
8. Дейт К. Дж. Введение в системы баз данных, М.: «Вильямс», 2006. 1328с.
9. Григорьев Ю. А., Плужникова О.Ю. Методика анализа характеристик производительности информационных систем с использованием комплекса инструментальных средств анализа моделей доступа к базам данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 09 Режим доступа: http://technomag.bmstu.ru/doc/630574.html (дата обращения 20.10.2013).
10. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2007. 366 с.