УДК 004.451.52 + 004.652.4

СЕТЕВЫЕ ЖУРНАЛЬНЫЕ ФАЙЛОВЫЕ СИСТЕМЫ НА ОСНОВЕ СЕРВЕРОВ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

А. Б. Гаврилюк, В.А. Алексеев

Институт программных систем НАН Украины, проспект Академика Глушкова, 40, Киев, 03187 ,
тел.: (044) 266 6321, alife-soft@yandex.ru, avictor@d19.isofts.kiev.ua

Рассматриваются основные характеристики файловых систем, сетевых файловых систем. Формулируются требования к современной файловой системе. Исследуются существующие файловые системы на основе реляционной модели представления данных на предмет соответствия их характеристик требованиям к современной файловой системе. Формулируется концепция к разработке сетевой файловой системы, где виртуализация файловой системы представлена реляционной моделью.

Basic performances of file systems, network file systems are considered. Requirements to the modern file system are stated. Present file systems are probed on the basis of a relational model of a data display for correspondence of their characteristics to requirements to the modern file system. The concept to design of network file system where virtualization of the file system it presented by a relational model is stated.

Введение

Предназначение любой компьютерной системы состоит в создании, обработке, хранении и представлении данных. Файловые системы предоставляют возможность более абстрактной организации и управления данными, без понимания пользователем или программистом внутренней архитектуры системы, как эти данные физически представлены на диске.

Во время начального развития вычислительной техники[1–4] не предъявлялось высоких требований к абстракции данных, представляемых файловой системой. Количество файлов на дисковом носителе было довольно низким, а объем малым, что позволяло использовать простейшие механизмы поиска и представления данных. Как и сама таблица размещения файлов, так и иерархическая файловая структура была представлена на основе списка дисковых кластеров. Метаданные файловой системы были представлены в виде записей фиксированного размера, что заставляло “общаться” с файловой системой по жестко определенному разработчиком сценарию. Кроме того, это приводило к экспоненциальной потере производительности при больших объемах информации и параллельном обращении к файловой системе.

Во времена больших компьютеров и маленьких программ отсутствовала необходимость разграничения доступа к данным на уровне файловой системы. Сейчас, когда дисковые пространства файловых серверов даже небольших организаций приближаются или уже превышают размер в 1 Тб, требуются все более новые возможности расширения и настройки матрицы прав, включая групповые политики. Также требуется возможность параллельной работы с политиками безопасности без блокировки всей системы в связи с этим.

Требования к теоретическому размеру файловых потоков и размера тома файловой системы с увеличением объема носителей информации постоянно растут. Сейчас 32-х битная файловая система не способна удовлетворить требования к хранению на жестком диске даже файлов в формате DVD-Video, не говоря уже о научных или производственных задачах. Хотя Sun заявила, что через 20 лет по закону Мура требования, предъявляемые к объему хранимой информации превысят 65-й бит, по этой причине их последняя файловая система ZFS была спроектирована 128-битной. При наличии достаточно доступных 64-х битных процессоров, использование 64-х битных систем сейчас более оправданно, чем использование более емкой 128-битной, но проигрывающей в производительности.

При увеличении объемов информации часто предъявляются более высокие требования к возможности поиска и семантической структуризации файловой системы. Пока это требование выполняется с помощью систем локального поиска и индексирования данных, которые прежде использовались только для индексирования данных в среде Internet. Одним из недостатков данного метода является частая неактуальность индексированных данных. При перемещении файла или изменении его, системе необходимо производить повторную индексацию файловой системы и выявление изменений. Эта проблема решилась бы при применении реляционности данных, где данные и метаданные явно связаны.

В современных файловых системах используются механизмы ускорения поиска и представления файлов, основанные на различных модификациях индексных структур. В ReiserFS для индексирования используются Б-деревья [5], что дает логарифмическое время поиска файла в файловой структуре. Это позволяет избежать потери производительности при гигантских объемах тома файловой системы и количества файлов на ней. Таким образом, при наличии на диске 2^9 файлов худшее время доступа к атрибутам файла, при двух ключах на узел, будет составлять всего 9 итераций.

Большинство современных файловых систем поддерживают возможность транзакций при выполнении файловых операций, записывая метаданные и опционально файловые данные в журнал, а только после завершения транзакции перенося их в файловую систему. Это позволяет избежать разрушения файловой системы при сбоях питания или программного обеспечения, кроме того, обеспечивает быстрое восстановление системы после сбоя без сканирования всей файловой системы. Однако, полное или частичное журналирование операций с файловой системой негативно сказывается на скорости работы системы и требует дополнительного места для организации журнала. Несмотря на недостатки в меньшей скорости файловых операций и потере места при ведении журнала, применение данных файловых систем предпочтительнее по причине более высокой готовности системы.

Представление файла как совокупности отдельных записей и возможность гибкого формирования структуры метаданных, которое представлено некоторыми современными файловыми системами, такими, как NTFS и BFS позволяет получить более высокий уровень абстракции управления данными, что приближает такие файловые системы к иерархическим системам документооборота.

Как мы видим, развитие файловых систем и развитие технологий баз данных неразделимо. Современные файловые системы уже используют большое количество технологий, которые до этого применялись только для организации и проектирования технологий реляционных баз данных. Не совсем корректно сравнивать файловую систему с базой данных. Ее можно представить как универсальную иерархическую систему документооборота, так как файл – это, прежде всего, непрерывный поток данных, документ, чем обычный набор записей в базе данных.

Использование сетевых файловых систем позволяет убрать различие между операциями с локальной файловой системой и расположенной на компьютере, который находится в соседнем здании или же на другом материке, если это позволяет скорость канала передачи данных. Построение распределенных сетевых файловых хранилищ на основе существующих протоколов сетевых файловых систем не представляет никакой сложности. Вся сложность по кэшированию, разграничению доступа и выполнению файловых операций ложится на механизм сетевой файловой системы, которые на данный момент уже хорошо развиты и стабильны.

Кроме того, разработка своей собственной сетевой файловой системы негативно сказывается на ее переносимости и возможности работы в гетерогенной среде, так как большинство существующих программных комплексов используют существующие и проверенные сетевые файловые системы. Однако разработка файловых систем пока не удовлетворяет всем требованиям, которые выставляет к ним существующее аппаратное обеспечение. При этом, часто, в гетерогенной среде используется уже известная файловая система, в которой файловые представления локальной файловой системы динамически транслируются в универсальное сетевое представление.

Таким образом, проблемы хранилищ данных в гетерогенной среде представляют довольно большой интерес. В данной работе кратко рассмотрены существующие файловые системы, их соответствие современным требованиям и интеграция их с сетевыми файловыми системами. Там будут рассмотрены новые направления в построении интегрированных хранилищ данных, как файловые системы на основе реляционной модели данных. Далее рассматривается концептуальная структура организации виртуальной файловой системы на основе реляционной модели базы данных и интеграция ее с сетевыми протоколами представления файловых системам.

Сравнительный обзор параметров существующих файловых систем

Первичная функциональность всех файловых систем [6–26] заключается в предоставлении средств хранения поименованных потоков данных и в дальнейшем получения данной информации, используя их имя. Поток данных – неделимое целое, если мы не используем системы с наличием нескольких файловых потоков, и является файлом, который имеет различные атрибуты, каким-либо образом связанные с хранящейся информацией, называемые метаданными. Таким образом, при разработке нового типа файловой системы мы разрабатываем новую абстракцию данных, которая способна представлять в связной форме “сырые” данные, хранящиеся на диске. Данные абстракции обеспечиваются множеством методов и средств ее организации, которые рассмотрим далее в табл. 1.

У большинства Unix-like файловых систем матрица прав имеет довольно ограниченный механизм для администрирования прав доступа к файлам (рис. 1). В этих системах каждый файл или каталог имеют маску прав доступа хозяина, группы (ее пользователей), которой принадлежит файл и всех остальных. Матрица прав позволяет каждому из них выдать только три вида прав: чтение, запись, запуск файла на исполнение.

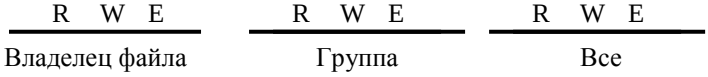

Рис. 1. Матрица прав доступа к файлу в виде маски. Read-чтение,Write-запись, Execute-запуск

Как было отмечено, наличие в системе механизма журналирования данных и метаданных дает высокую устойчивость системы к сбоям и гарантирует сохранность данных. Механизм журналирования данных перед записью на диск, конечно, замедляет скорость записи, но при больших объемах данных можно получить довольно большой выигрыш в скорости восстановления системы после сбоя.

Сравнительная характеристика файловых систем Таблица 1

ФС	Разграничение прав доступа	Журнал метаданных	Журнал данных	Альтернативные файловые потоки или расширенные метаданные	Максимальный размер файла /тома	Жесткие ссылки/ символические ссылки
FAT12	-	-	-	-	32Мб/32Мб	-/-
FAT16	-	-	-	-	2Гб /2Гб	-/-
FAT32	-	-	-	-	4Гб/2Тб	-/-
HPFS	-	-	-	+	4Гб/2Тб	-/-
Ext2	+	-	-	+	2Тб/32Тб	+/+
Ext3	+	+	+	+	2Тб/32Тб	+/+
NTFS	+	+	+	+	16Еб/16Еб	+/+
XFS	+	+	-	+	9Еб/9Еб	+/+
JFS	+	+	-	+	4Пб/32Пб	+/+
ReiserFS	+	+	+	+	8Тб/16Тб	+/+
BFS	-	+	+	+	256Гб/2Еб	+/+
NWFS	+	-	-	+	4Гб/1Тб	+/+
NSS	+	+	+	+	8Тб/8Тб	+/+
FFS	-	-	-	-	4Гб/256Тб	+/+
UFS1	+	-	-	-	256Тб/256Тб	+/+
UFS2	+	-	-	+	32Пб/10^{24}	+/+
VxFS	+	-	+	+	16Еб/16Еб	+/+
ZFS	+	-	+	+	$2^{128}/2^{128}$	+/+

Файловые системы с наличием нескольких файловых потоков присущих одному логическому файлу, позволяют организовать более удобную абстракцию для представления данных. На рис. 2 показано файл **somefile** с несколькими связанными с ним файловыми потоками. Основная запись данных идет в файловый поток **$main**, но в то же время наличие нескольких файловых потоков позволяет, например, вместе с файлом документа хранить его перевод в потоке **$translation** или набор картинок из документа в потоке **$images**. Для доступа к этим файловым потокам программа не должна знать формат файла документа и, таким образом не сможет его разрушить при неправильном разборе формата. Наличие файловых потоков характеризует возможность более гибкого управления семантическим контекстом системы.

Если наличие в файловой системе символических ссылок на папки представляет полезность для возможности более гибкого управления файловой структурой, путем представления иерархического дерева в виде ориентированного графа, то необходимость жестких ссылок весьма сомнительна.

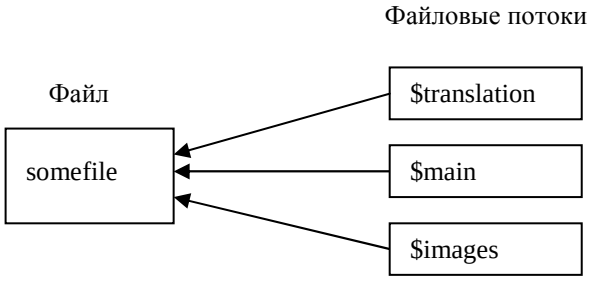

Рис. 2. Файловые потоки

После рассмотрения основных файловых систем были выделены ключевые требования для современных файловых систем, такие как:

- гибкая матрица прав для разграничения доступа к данным;
- журналирование как метаданных, так и данных при проведении файловых операций;
- наличие нескольких файловых потоков для удобства представления данных;
- возможность управления данными более 1Тб, оптимальным вариантом является использование 64-х битных указателей в файловой системе;
- возможность создания символических ссылок на объекты локальной файловой системы или объекты распределенных файловых систем для организации сетевых структур внутри иерархических систем.

Обзор общераспространенных сетевых файловых систем

Сетевые файловые системы [27–36] позволяют разделять доступ к локальной файловой системе с другими пользователями в локальной или глобальной сети. Ключевым компонентом любой распределенной системы является файловая система, которую в данном случае можно назвать распределенной. Как и в локальной системе, в распределенной системе функцией файловой системы является хранение программ и данных и предоставление по мере необходимости доступа к ним.

Однако в отличие от локальных файловых систем, которые возможно разрабатывать под определенные требования, сетевая файловая система должна быть реализована как на клиенте, так и на сервере, серверная и клиентская часть соответственно. Это накладывает определенные ограничения тем, что сетевая файловая система должна быть довольно распространенной. Несмотря на то, что существует более 20 различных реализаций сетевых файловых систем, самыми распространенными из них являются 3. Рассмотрим их далее.

File Transfer Protocol (FTP) общеизвестный протокол для обмена файлами между двумя компьютерами в локальной сети на базе сетевого протокола TCP/IP. Несмотря на примитивную схему аутентификации пользователей, через передачу имени пользователя и пароля в открытом виде, и отсутствие произвольного доступа к файловой системе этот протокол имеет преимущество в своей простоте, открытости и распространенности. Это позволяет практически любой системе, даже встроенной, обмениваться файлами по сети путем загрузки или выгрузки всего файла.

Network File System (NFS) – это классическая сетевая файловая система, позволяющая монтировать себя в локальную файловую систему и прозрачно предоставляющая доступ к файловой системе удаленного сервера. Работа NFS опирается на концепцию вызовов удаленных процедур (RPC). Согласно этой концепции, при доступе к удаленному ресурсу, например к файлу, вызов прозрачно перенаправляется на удаленный компьютер, обрабатывается там и уже готовый результат возвращается на клиентский компьютер. Каждый пакет RPC несет полную информацию о том, что необходимо выполнить на сервере, или о результатах выполнения процедуры. Это предполагает высокую надежность сервера, так как ему не нужно хранить информацию о текущем состоянии работы с клиентской машиной, но все сложности по выполнению удаленных файловых операций ложатся на клиент. Кроме того, это не позволяет открыть обмен данными с сервером в сессионном режиме, что приводит к проблемам при одновременной записи в удаленную файловую систему и отсутствию файловых блокировок. В NFS аутентификация производится исключительно на этапе монтирования файловой системы и только на основании доменного имени или IP адреса клиентской машины.

Server Message Block (SMB) относится к классу протоколов, ориентированных на установление соединения и выполнение файловых операций внутри установленной сессии. Работа протокола начинается с того, что клиент отправляет серверу специальное сообщение с запросом на установление соединения. В процессе установления соединения клиент и сервер обмениваются информацией о себе: они сообщают друг другу, какой диалект протокола SMB они будут использовать в этом соединении. Если сервер готов к установлению соединения, он отвечает сообщением-подтверждением. После установления соединения клиент может обращаться к серверу, передавая ему в сообщениях SMB команды манипулирования файлами и каталогами.

Протокол предоставляет произвольный доступ к файлам, оптимистическую и пессимистическую блокировку файлов, кэширование данных, предупреждение об файловых изменениях, автоматическую проверку версий протокола, множественные запросы в одном пакете и др.

Аутентификация производится двумя способами. Первый способ такой же, как у протокола FTP – посылка серверу имени пользователя и пароля в открытом виде. Второй способ базируется на протоколе запрос/ответ. Клиент посылает серверу имя пользователя, сервер в ответ посылает клиенту “запрос” на который клиент может ответить, если знает пароль. “Ответ” создается из “запроса” путем шифрации “запроса” 168-битный сессионным ключом, вычисленным на основе пароля пользователя. При получении правильного “ответа” сервер устанавливает соединение.

Как видно, протокол SMB способен удовлетворить все потребности современных систем в сетевой коммуникации. Самым основным недостатком этого протокола состоит сложность реализации, что не дает ему такого высокого распространения как двум вышеперечисленным. Одновременное использование всех трех вышеперечисленных протоколов дает универсальность файловому серверу в гетерогенной среде.

Файловые системы на основе реляционной модели данных

Основным отличием файловой системы на основе реляционной модели базы данных [37–45] является использование в качестве файлового хранилища сервера реляционной базы данных. Преимущество таких систем состоит в том, что структура файловой системы не зависит от физического размещения информации на диске. При этом, возможно проектирование абстракции файловой системы и в дальнейшем физическая поддержка механизма управления данными будет обеспечиваться реляционной моделью базы данных.

На данный момент самые известные решения подобных архитектур WinFS, IFS, DBFS. Каждое из них имеет большое количество отличий друг от друга, поэтому рассмотрим их все по отдельности.

WinFS – современное файловое хранилище (рис. 3), базирующееся на реляционной модели данных. Эта система позволяет улучшить управление данными по данным причинам:

- категоризация информации различными способами и связь элементов данных друг с другом;

- система предоставляет стандартный формат представления данных, как способы связи с людьми, мультимедийная информация, или каким-либо иным способом связанная с документом информация;
- способствует обмену “знаниями” между различными приложениями.

Одно из уникальных свойств WinFS – возможность использования языка запросов для доступа к файловым потокам и метаданным. Так как WinFS часть реляционной системы, то это позволяет широкую интеграцию данных, как по иерархической структуре, так и по семантической.

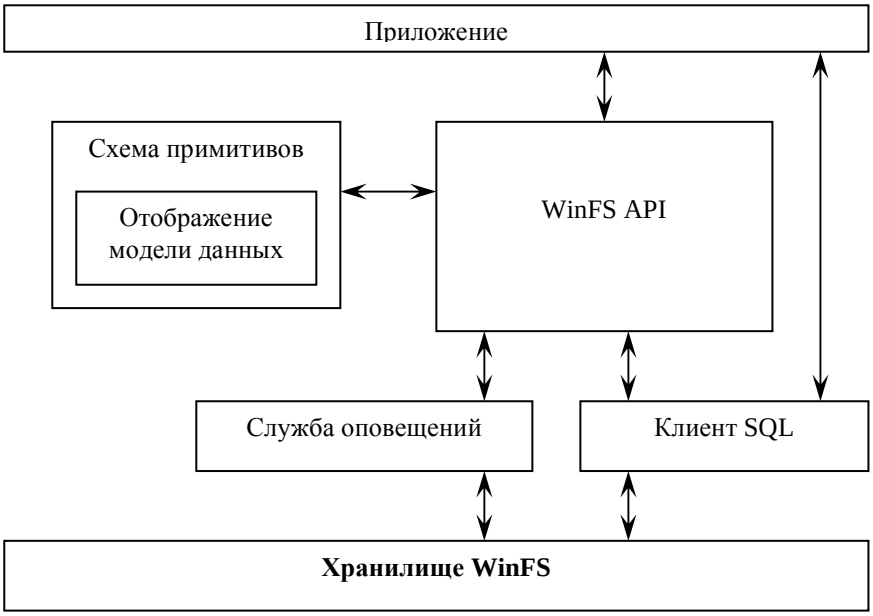

Рис. 3. Архитектура WinFS

Как видно, WinFS позволяет решить такие проблемы как:
- реструктуризацию абстракций модели представления данных;
- простой интерфейс разделения доступа к данным для различных приложений;
- множественное представление данных;
- возможность извещения при изменении элемента данных;
- возможность создания собственного представления данных через язык запросов.

DBFS (рис. 4) – новый тип представления данных, который расширяет представление об обычной файловой системе. Хотя это не полноценная файловая система, а скорее система индексации и поиска документов в локальной файловой системе. Эта файловая система из всех свойств реляционной файловой системы позволяет только семантическую реляционную организацию, связанных с файлом, метаданных.

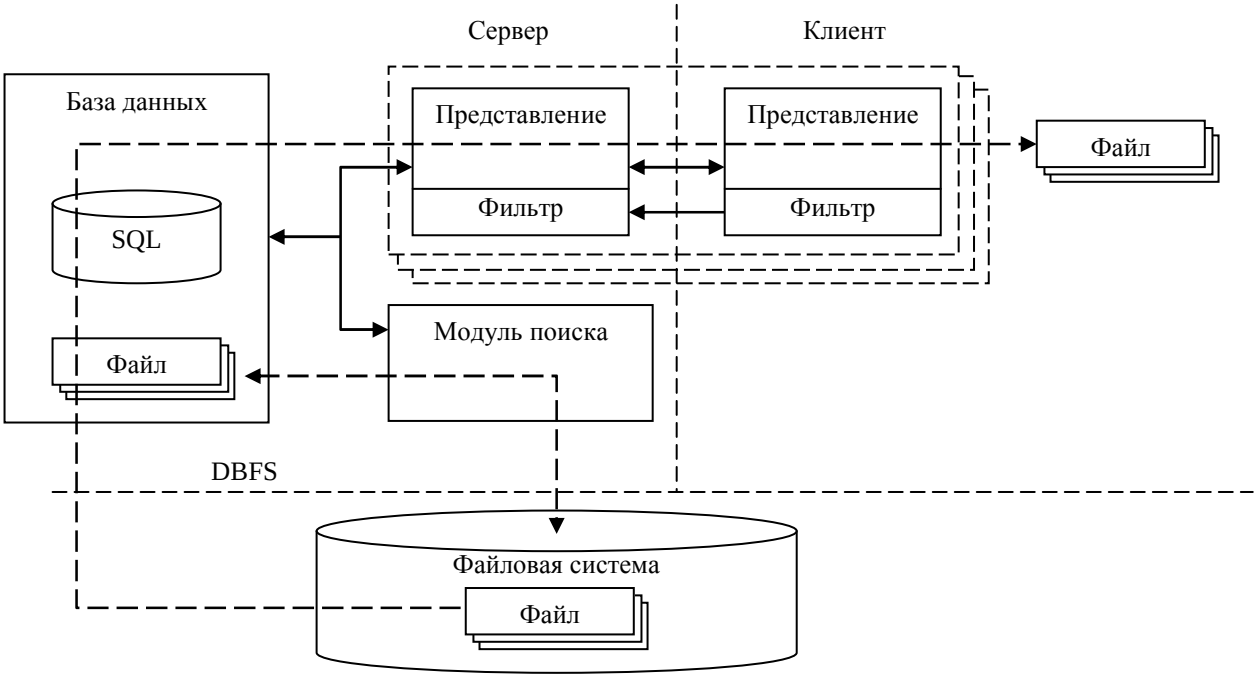

Рис. 4. Представление реализации DBFS

Файловая система при работе сканируется автоматическим модулем поиска, который заносит метаданные в реляционную базу данных. Далее, клиент при потребности формирует запрос на необходимое ему представление файлов, как, например, все почтовые файлы, автор которых “Смит”. На основе индексированной информации в DBFS, формирует SQL запрос, результатом которого являются реальные пути к файлам в локальной файловой системе. Так обеспечивается семантическая связь между данными.

Oracle IFS – это файловая система (рис. 5), базирующаяся на механизме реляционной базы данных. Эта система является классическим примером объединения файловой системы и базы данных. Для пользователей файловой системы Oracle IFS представляется как стандартный файловый сервер, организующий файлы в иерархию папок. Пользователи не видят структуру данных в базе, а работают как с обычным файловым, почтовым или web сервером. При этом пользователи системы не могут напрямую работать с базой данных.

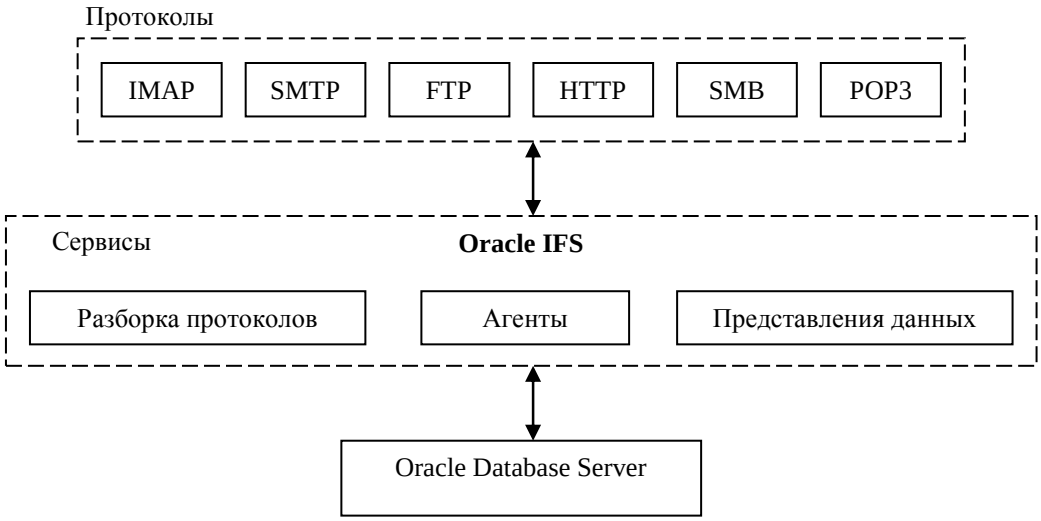

Рис. 5. Архитектура Oracle IFS

Oracle IFS представляет определенные удобства, как для администраторов, так и для конечных пользователей. Для администраторов удобство системы обеспечивается простым единым механизмом управления электронной почтой, файлами и web сервером, возможностью резервного копирования всей системы и простой поддержкой единого хранилища данных.

Для конечного пользователя удобство представляют:

- универсальный доступ к информации. Пользователи могут получить доступ к своим данным через файловый сервис, почтовый клиент или web клиент;
- улучшенный поиск. Возможности поиска Oracle IFS намного превышают возможности других аналогичных систем, кроме того, поиск может вестись не только по содержимому файлов, а и по метаданным;
- упрощенный обмен файлами с другими пользователями;
- высокая надежность при хранении данных, при резервном копировании данных происходит сохранение всей базы и всех пользовательских файлов.

Файловые системы на основе баз данных представляют собой простой и удобный механизм для хранения, поиска и управления различными типами данных. Надежность этих файловых систем довольно высока. Также, хранение метаданных упрощает поиск и управление пользовательскими данными.

Сетевая файловая система на основе реляционной модели базы данных

После рассмотрения всего вышеперечисленного была предложена концепция файлового хранилища для распределенных систем на основе реляционной модели данных (рис. 6). Использование реляционных механизмов для формирования виртуальной файловой системы позволит перенести характеристики существующих реляционных баз данных на рассматриваемую файловую систему. Это позволит как можно более близко совместить классы файловых систем и реляционных баз данных.

Концептуальная модель вышеуказанного файлового хранилища позволяет получить такие характеристики виртуальной файловой системы, которые вытекают из ее реляционной организации, как:

- возможность гибкого разграничения прав пользователей и их объединений с возможностью реконфигурации политик безопасности;
- практически полное отсутствие ограничений на размер базы и тома файловой системы, который в ней будет храниться, которая в большинстве серверов баз данных уже составляет 2^{64};
- структуризацию файла на поименованные файловые потоки, которые представляют для приложений сохранять служебную информацию о файле;
- количество итераций при выборке или поиске данных составляет $\log_{(\text{к-во ключей в узле})}(\text{количество файлов})$, обеспечиваемое механизмами индексации, которые являются стандартными для баз данных;

- наличие изменения структуры метаданных файловой системы и ее структуризации на основании значений этих метаданных, что позволит не смотря на размер файловой структуры всегда быстро находить нужную информацию;

- механизм транзакций, который является стандартным для сервера баз данных и повышает отказоустойчивость системы.

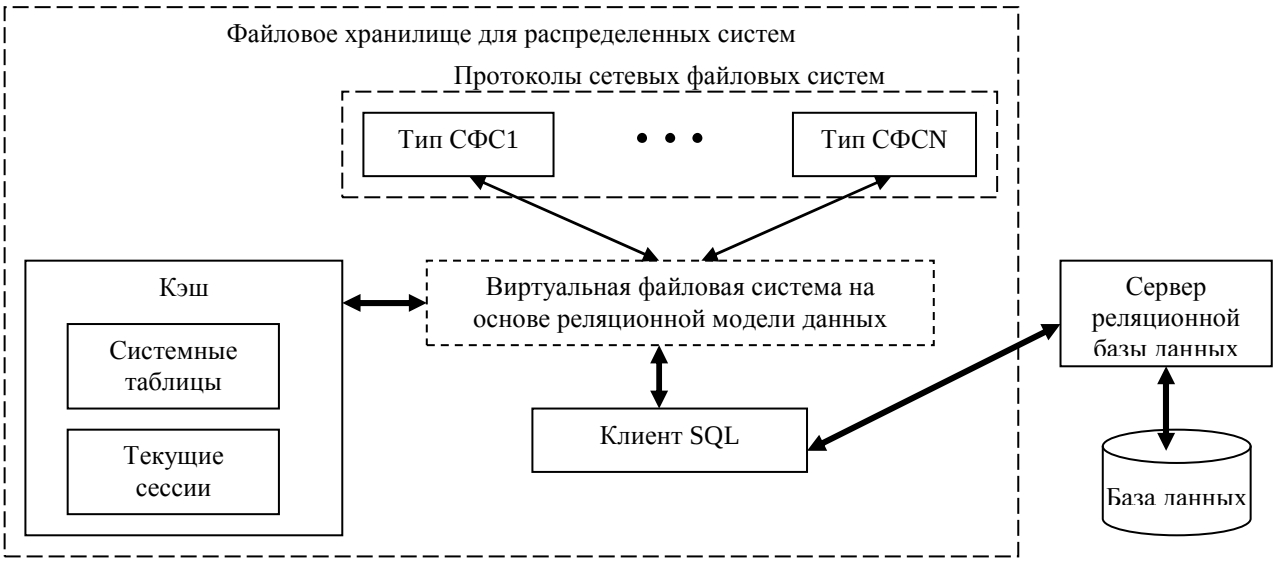

Рис. 6. Файловое хранилище для распределенных систем на основе реляционной модели данных

Как мы видим, направление построения файловых систем на основе баз данных мало изучено и не имеет четкой методологии. Преимущества этого подхода проявляются в возможности управления гигантскими объемами данных, что особенно актуально сейчас, когда стандартный объем носителей информации на персональных компьютерах приближается к терабайту. В связи с этим было принято решение к разработке методов и средств построения файловых систем на основе баз данных, результаты которых будут представлены в последующих работах.

Практические результаты

Принцип построения виртуальной файловой системы на основе базы данных был применен при разработке программного модуля ГАРТ-Факс, являющегося частью программного комплекса ГАРТ. Внутренняя иерархическая структура файловой системы была представлена реляционной структурой. Кроме того, наличие метаданных, связываемых с факсимильными сообщениями, позволяло осуществлять удобный поиск и управление файловой системой, чего нельзя было бы достигнуть при использовании стандартных файловых систем.

Выводы

Эволюция файловых систем призвана обеспечить возможности эффективного использования и управления данными на носителях информации, объем которых становится все больше. Использования классических методов для представления файловой информации сегодня становится недостаточно. Современные средства реляционных баз данных дают возможность создания файловых систем предоставляющих сервис, подобный системам документооборота, обеспечивая повышенную эффективность работы. Реализация концепции интеграции файловой системы в базу данных способны перенести свойства анализа и поиска информации, которые ранее были присущи только реляционным базам данных на файловые системы. Исследование данного подхода к реализации файловых систем представляет собой актуальную проблему и высокую практическую ценность.

1. *Олифер В.Г., Олифер Н.А.* Сетевые операционные системы –http://lib.aswl.ru/books/methodology/network-OSes/
2. *Семенов Ю.А.* Telecommunication technologies - телекоммуникационные технологии – http://book.itep.ru/
3. *Норенков И.П., Трудоношин В.А.* Телекоммуникационные технологии и сети –http://www.kgtu.runnet.ru/WD/TUTOR/net/net0.html
4. *Wikipedia* File system - http://en.wikipedia.org/wiki/File_system
5. *Kantor I.* Б, Б+ и Б++ деревья – http://algolist.manual.ru/ds/s_btr.php
6. *Хименко В.* Файлы, файлы, файлы – http://www.osp.ru/pcworld/2000/02/064.htm
7. *Олифер В.Г., Олифер Н.А.* Восстанавливаемость файловых систем – http://www.osp.ru/os/2001/09/023.htm
8. *Хуан С.Ф.* Журнальные файловые системы – http://www.osp.ru/os/2001/09/028.htm
9. *Зогин В.* Файловые системы – http://www.remont-pc.ru/fs.htm

10. *Обзор* файловых систем FAT, HPFS и NTFS – http://support.microsoft.com/kb/100108/
11. *Newman H.* Review of Current File Systems and Volume Managers – http://www.samag.com/documents/s=7789/sam0302h/0302h.htm
12. *Ext2fs* Home Page – http://e2fsprogs.sourceforge.net/ext2.html
13. *NTFS:* Platform SDK – http://msdn.microsoft.com/library/en-us/fileio/fs/ntfs.asp
14. *Inside* Win2K NTFS, Part 1 (Windows 2000 Magazine (2000)) – http://msdn.microsoft.com/library/en-us/dnw2kmag00/html/NTFSPart1.asp
15. *Linux-NTFS* Project – http://linux-ntfs.sourceforge.net/
16. *Русинович М.* Возможности NTFS – http://www.osp.ru/win2000/2001/07/030.htm
17. *Ракалин А.* Большие файлы в Unix – http://www.osp.ru/os/2000/07-08/034.htm
18. *Sweeney A., Doucette D.* Scalability in the XFS File System – http://oss.sgi.com/projects/xfs/papers/xfs_usenix/index.html
19. *Hellwig C.* XFS for Linux – http://oss.sgi.com/projects/xfs/papers/ukuug2003.pdf
20. *Journaled* File System Technology for Linux – http://jfs.sourceforge.net/
21. *Stephen C.T.* Journaling the Linux ext2fs Filesystem – ftp://ftp.linux.org.uk/pub/linux/sct/fs/jfs/journal-design.ps.gz
22. *Reiser4* – http://www.namesys.com/v4/v4.html
23. *Novell NetWare* – http://www.novell.com/products/netware/
24. *Giampaolo D.* File System Design with the Be File System – http://www.nobius.org/~dbg/practical-file-system-design.pdf
25. *Zeta* is the operating system – http://www.yellowtab.com/products/
26. *Сообщество* пользователей BeOS – http://www.qube.ru/
27. *Postel J., Reynolds J.* RFC 959 - File Transfer Protocol (FTP) – http://www.rfc-editor.org/rfc/rfc959.txt
28. *Пьянзин К.* Основные особенности работы файловой системы NFS на платформе UNIX – http://www.osp.ru/lan/2000/01/049.htm
29. *Callaghan B., Pawlowski B., Staubach P.* RFC 1813 - NFS Version 3 Protocol Specification – http://www.rfc-editor.org/rfc/rfc1813.txt
30. *RFC1001* - Protocol Standard for a NetBIOS service on a TCP/UDP transport: concepts and method – http://www.rfc-editor.org/rfc/rfc1001.txt
31. *RFC1002* - Protocol Standard for a NetBIOS service on a TCP/UDP transport: detailed specifications – http://www.rfc-editor.org/rfc/rfc1002.txt
32. *The Samba* Developer's Guide – http://us1.samba.org/samba/docs/Samba-Developers-Guide.pdf
33. *CIFS* Technical Reference 1.00 – http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf
34. *Netatalk* - Networking Apple Macintosh through Open Source – http://netatalk.sourceforge.net/
35. *Coda* File System – http://www.coda.cs.cmu.edu/
36. *Pike R., Presotto D., Dorward S., Flandrena B.* Plan 9 From Bell Labs – http://plan9.bell-labs.com/sys/doc/9.html
37. *Grimes R.* Revolutionary File Storage System Lets Users Search and Manage Files Based on Content – http://msdn.microsoft.com/data/default.aspx?pull=/msdnmag/issues/04/01/WinFS/default.aspx
38. *Rector B.* Data Access and Storage Developer Center: Building WinFS Solutions: Chapter 4: Storage – http://msdn.microsoft.com/data/winfs/default.aspx?pull=/library/en-us/dnintlong/html/longhornch04.asp
39. *Rizzo T., Grimaldi S.* An Introduction to "WinFS" OPath – http://msdn.microsoft.com/data/default.aspx?pull=/library/en-us/dnwinfs/html/winfs10182004.asp
40. *Building* WinFS Solutions – http://msdn.microsoft.com/data/winfs/
41. *Gorter O.* DBFS – http://ozy.student.utwente.nl/projects/dbfs/
42. *Gorter O.* Database File System An Alternative to Hierarchy Based File Systems – http://ozy.student.utwente.nl/projects/dbfs/dbfs-paper.pdf
43. *Apple* - Mac OS X – Spotlight – http://www.apple.com/macosx/features/spotlight/
44. *Oracle* Internet File System Setup and Administration Guide Release 1.1 – http://www.mid.main.vsu.ru/docs/oracle9/ifs.901/a81197/toc.htm
45. *Oracle* Internet File System Archive Documentation –http://www.oracle.com/technology/documentation/ifs_arch.html