

Jet Info

ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ

№ 12 (187)/2008

Кластерные СУБД



ВЫЧИСЛИТЕЛЬНЫЕ
КОМПЛЕКСЫ

Кластерные СУБД

Денис Павлов,
инженер по направлению Вычислительные комплексы
Группы перспективных технологий

СОДЕРЖАНИЕ

Введение	3
Shared Nothing vs Shared Everything	3
Основные идеологии построения кластерных БД	3
Shared everything	4
Shared Nothing	5
IBM – DB2	5
Базовые понятия IBM DB2.....	5
Принципы построения параллельных СУБД.....	7
Организация доступа к параллельным средам	9
Отказоустойчивость.....	11
Резервное копирование	13
Операционные системы и аппаратные платформы для DB2	13
MySQL	14
Hypertable	16
Sybase ASE	19
Exasol и Paracel	20
Paracel	22
Exasol.....	22
Цена и производительность кластерных СУБД	23
Сравнение результатов TPC-H для Exasol и Paracel	24
Сравнение результатов TPC-H для Oracle, IBM и Exasol.....	24
Заключение	25
Использованные материалы	27
Перечень используемых сокращений	27

Введение

Направление развития информационных технологий все чаще затрагивает кластеризацию или разделение БД по нескольким серверам. Причем, если раньше кластеризация БД имела целью, прежде всего, создание высоконадежной и отказоустойчивой системы, то теперь, наряду с требованиями к надежности, все чаще преследуются цели распределения нагрузки по серверам, повышения производительности и масштабируемости систем, создания специализированных информационных «решеток» (grid) для параллельной обработки баз данных.

Компания Oracle, являющаяся крупнейшим разработчиком в области СУБД, уже довольно давно продвигает для этих целей свое решение — Oracle Real Application Cluster. Безусловно, компания добилась определенных успехов в разработке этой системы, и область применения кластеров RAC к моменту написания данной статьи существенно расширилась по сравнению с начальным периодом появления RAC на рынке ИТ. Однако в кругах специалистов существует мнение, что, не считая баснословно дорогой стоимости лицензий, RAC по своей архитектуре наделен рядом существенных технических недостатков, преодоление которых является весьма непростой задачей, и вряд ли будет осуществлено в ближайшем будущем.

Принимая во внимание тот факт, что специалистами компании «Инфосистемы Джет» для построения баз данных используются преимущественно решения Oracle, и недостаточную освещенность продуктов других производителей, было решено ознакомиться и провести первоначальное описание продуктов и технологий других разработчиков в области СУБД.

Таким образом, данная статья имеет целью описание и первоначальное сравнение продуктов и технологий СУБД, направленных на параллельную обработку и кластеризацию баз данных. Попутно в статье излагаются цели или примеры применений рассматриваемых технологий.

Другая, возможно, главная цель данной статьи — исключение некой однобокости при последующем проектировании и создании параллельных вычислительных систем для СУБД, знакомство с общей картиной доступных в этой области технологий.

Shared Nothing vs Shared Everything

Основные идеологии построения кластерных БД

В настоящий момент наиболее известными идеологиями построения параллельных кластерных СУБД являются:

- «С разделяемыми дисками» (shared disk — SD) (рис. 1). Случай, когда СУБД, располагающаяся на нескольких компьютерах, использует одни и те же устройства хранения.

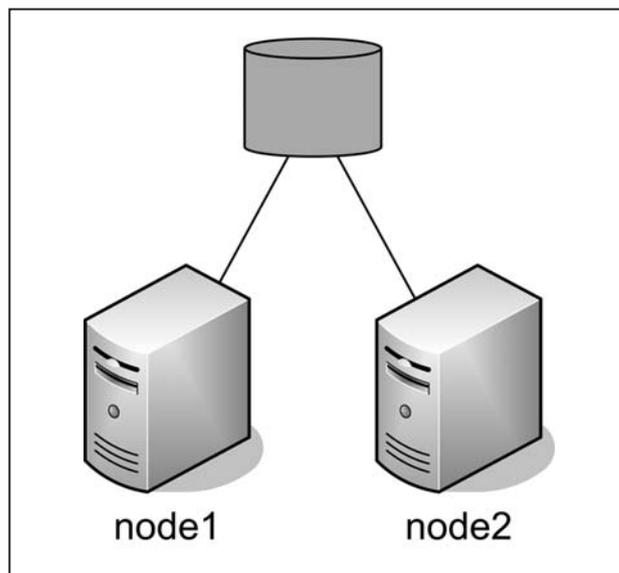


Рис. 1. Shared disk

- «С разделяемой памятью» (shared memory — SM). В этом случае много процессоров в СУБД, возможно, расположенных на разных машинах, используют общую (разделяемую) память (рис. 2, стр. 4).
- «Ничего общего» (shared nothing — SN). Ни устройства хранения, ни память таких систем не разделяются. В таких системах каждый узел обслуживает свой фрагмент БД, уникальность которого обеспечивается либо организацией БД, либо дополнительными средствами управления и мониторинга СУБД. Этот подход будет рассмотрен более подробно в главе Shared Nothing.

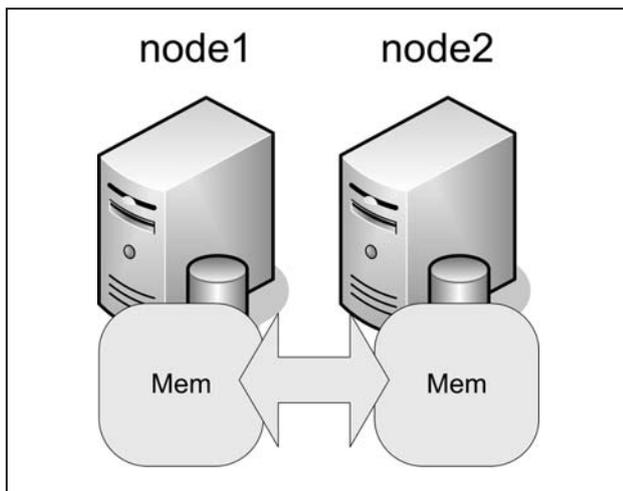


Рис. 2. Shared memory

Shared everything

Случай симбиоза SM+SD, когда СУБД использует и общие устройства ввода-вывода, и память, носит название shared everything (SE – «все общее» (рис. 3)). К таким системам, как раз, относятся Oracle RAC.

Такие системы, очевидно, должны обладать высокоскоростной связью между узлами для обеспечения работы с разделяемой памятью, а также какими-либо средствами совместного доступа к устройствам хранения – кластерная файловая система, менеджер устройств хранения и т.п. Достоинства здесь очевидны – это сочета-

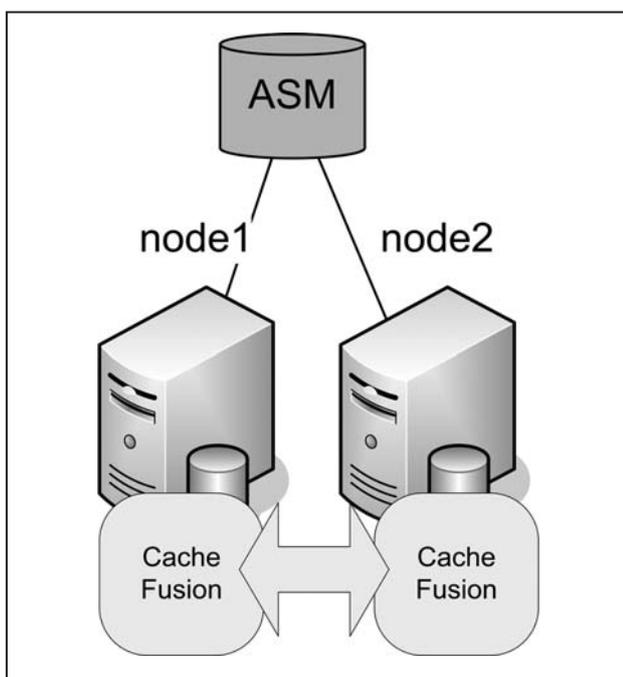


Рис. 3. Shared everything

ние параллельной обработки, масштабируемости и высокой отказоустойчивости системы.

Однако есть и недостатки, самый серьезный и очевидный из которых – конкуренция за разделяемые ресурсы (устройства ввода-вывода и память) между узлами кластера. В кластерах Oracle RAC наиболее болезненно эти недостатки проявляются при нагрузке, состоящей из большого числа коротких OLTP-запросов, которые требуют сравнительно мало процессорных ресурсов и содержат операции DML (insert, update, delete). Типичная ситуация, возникающая в этом случае, вызывает ожидания клиентских сессий, связанных с блокировками буферов глобального кэша. Механизм возникновения такой ситуации пояснен на рис. 4.

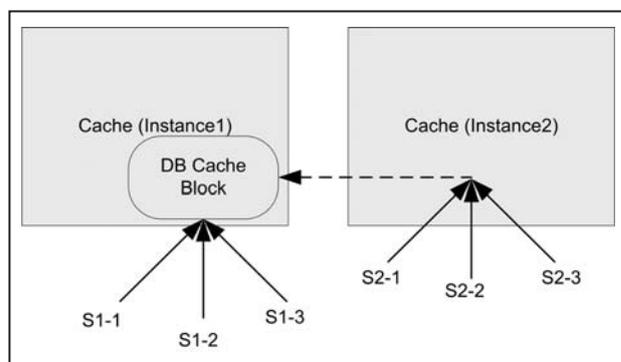


Рис. 4. Доступ к блокам глобального кэша

Сессии второго экземпляра запрашивают блок, уже находящийся в кэше первого экземпляра (в таком случае говорят, что блок присутствует в глобальном кэше). Хотя блок и может находиться в кэше всех экземпляров БД одновременно, текущая его версия может быть только одна. Для его обработки блок или статус текущего блока должен быть получен вторым экземпляром и затем обработан запрашивающей его сессией. Однако блок в данное время обрабатывается также и локальными сессиями первого экземпляра. Таким образом, локальная обработка блока мешает его обработке на удаленном экземпляре. В это время сессиям второго экземпляра приходится ждать. Следует также отметить, что локальная блокировка менее затратна по времени, чем передача блока для обработки на другой экземпляр. Но в определенный момент блок получает статус текущего на другом экземпляре, и сессии первого и второго экземпляров меняются ролями.

В результате, если не принимать специальных мер по разграничению доступа экземпляров к блокам БД, производительность узлов и масштабируемость кластера будет существенно ниже,

чем можно ожидать, увеличивая число узлов кластера. Эти меры известны и носят название функционального разбиения (functional partitioning) БД. К примеру, в пределах одной БД могут быть организованы сервисы, каждый из которых в штатном режиме работает на одном из узлов и отвечает только за «свои» таблицы или разделы таблиц. Можно также применять привязку к разделам и экземплярам на уровне приложения. В любом случае приложение должно быть оптимизировано с этой точки зрения.

Shared Nothing

Этот подход к созданию параллельной БД лишен указанного выше недостатка — конкуренции за разделяемые ресурсы по причине их отсутствия. В кластере, созданном в такой идеологии, узлы не разделяют ресурсы между собой. Каждый узел обрабатывает свой фрагмент базы. За счет этого существенно возрастает производительность и масштабируемость системы с такой организацией на нагрузках любого типа.

В минусе — большая сложность всей системы. Во-первых, система будет требовать дополнительных звеньев, целью которых будет мониторинг состояния и управление запросами и конфигурацией БД. Такая организация сложнее, чем организация систем с разделяемыми компонентами, и требует технически более сложных средств организации резервирования компонентов и обеспечения непрерывности функционирования. Во-вторых, для обеспечения отказоустойчивости и целостности распределенных данных необходимо тем или иным образом реплицировать узлы,

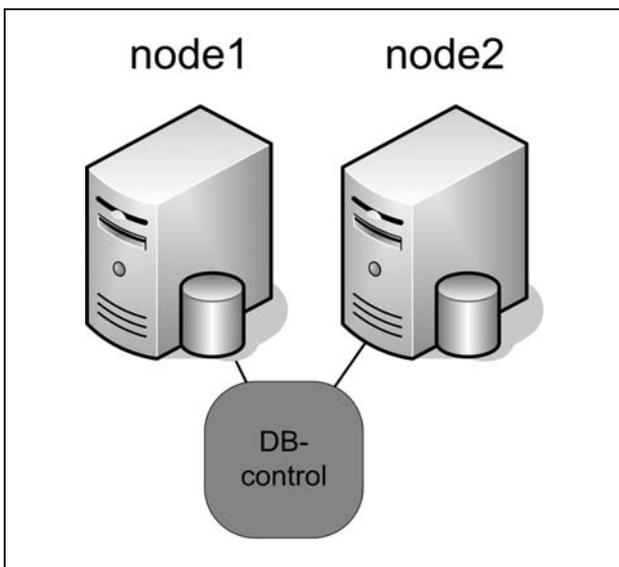


Рис. 5. Shared Nothing

обрабатывающие запросы. В разделах ниже рассматриваются продукты, использующие этот подход в тех или иных его вариациях.

IBM – DB2

Компания IBM еще в 90-е годы развивала концепцию и технологии параллельных баз данных на UNIX-системах. А с конца 90-х и начала 2000-х годов, в частности, развивает технологию массивно параллельных и кластеризованных баз на высокопроизводительных кластерах, работающих под управлением ОС AIX, Linux, HP-UX, Solaris, Windows. Для построения БД используется известный продукт фирмы — IBM DB2 Enterprise Server Edition (DB2 – ESE). Уже в 2003 году, по заявлениям компании, можно было строить кластеры DB2 до 1000 узлов!

Базовые понятия IBM DB2

В настоящее время, продукт IBM DB2 ESE позволяет создавать различные по конфигурации системы управления базами данных. Это могут быть как системы на одном сервере, так и кластеры из одинаковых однопроцессорных или многопроцессорных узлов, и их всевозможные сочетания для одной или нескольких баз данных.

Для дальнейшего описания DB2 необходимо привести и растолковать здесь термины, которыми пользуется IBM при создании и описании своих продуктов.

Рис. 6 поясняет основные понятия IBM DB2.

На самом верхнем уровне находится система (system). Система представляет собой некий аппаратный комплекс, созданный для работы DB2. Это может быть как одиночный сервер, так и кластер из множества машин, и более сложная и разнородная структура.

Следующий уровень — это экземпляр СУБД (database manager instance). Здесь у специалистов по Oracle может возникнуть разночтение, т.к. IBM вкладывает в понятие экземпляра именно среду управления множеством баз данных, в то время, как в понятии Oracle instance заключается экземпляр отдельно взятой БД. Экземпляров СУБД DB2 может быть несколько в пределах одной системы. И каждый из этих экземпляров мо-

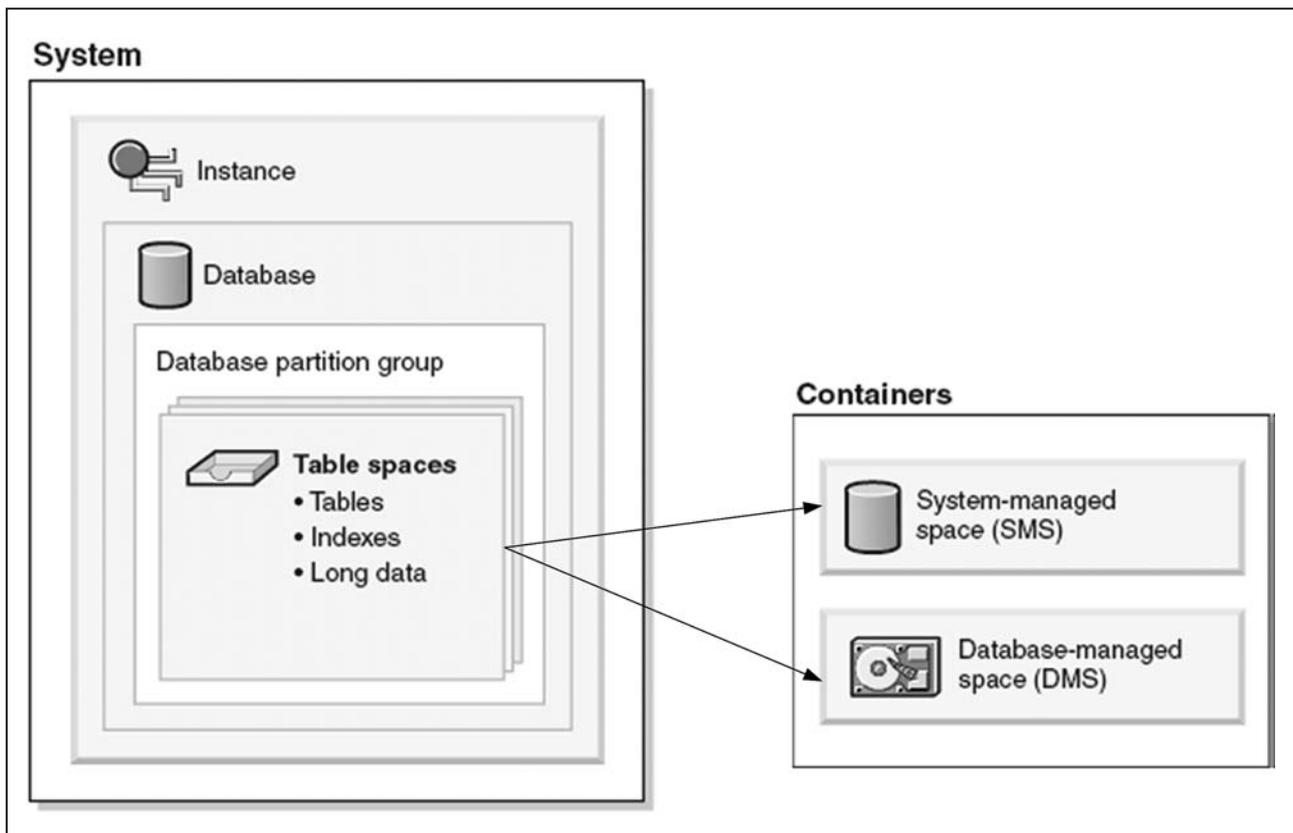


Рис. 6. Обобщенная схема построения СУБД IBM DB2 (1)

жет иметь собственные БД, недоступные другим экземплярам.

В одном экземпляре СУБД может быть создано несколько баз данных (database). Это понятие сходно с общепринятым.

Каждая БД может быть разбита на разделы (partitions). Разделы БД предусмотрены для организации параллельной обработки и исключения конкуренции за общие ресурсы. Каждый раздел БД имеет свое пространство данных, индексов и журналов транзакций. Таблицы могут размещаться как непосредственно в пределах одного раздела БД, так и могут распределяться по разделам. Запросы пользователя могут автоматически разбиваться на подзапросы, которые выполняются в разделах БД. Может также вестись параллельная обработка строк одной и той же таблицы, находящейся в нескольких разделах. Раздел БД может помещаться на узле кластерной системы (недаром, в более ранних версиях DB2 раздел БД назывался узлом БД – DB-node), но также может быть создан и в пределах одной многопроцессорной системы.

Для удобства размещения таблиц БД на одном или нескольких разделах, применяют группы разделов (partition group). В каждую группу мо-

жет быть включен один или несколько разделов. Группы разделов можно редактировать в процессе эксплуатации, добавляя или удаляя разделы БД из них.

Базовые объекты БД, такие как таблицы (tables), индексы (indexes) и пр. находятся в пределах табличных пространств (table space). Несмотря на раздельное написание слова table space, по смыслу этот объект в точности совпадает с табличными пространствами Oracle (tablespace). С помощью табличных пространств осуществляется привязка таблиц к группам разделов, а также распределение данных по устройствам хранения.

Привязка табличных пространств к устройствам хранения осуществляется на уровне контейнеров (container). Контейнеры табличных пространств бывают двух типов – управляемые базой данных (Database Managed Space – DMS) или системой (System Managed Space – SMS). Контейнеры табличных пространств класса SMS представляют собой каталоги файловой системы и, таким образом, управляются на уровне ОС и файловой системы. Контейнеры класса DMS – это «сырые» дисковые устройства или специально созданные файлы фиксированного размера. Доступ к таким контейнерам контролируется базой данных.

Принципы построения параллельных СУБД

Построение СУБД с параллельной обработкой изначально заложено в архитектуру DB2. Причем, параллельное использование ресурсов может происходить на нескольких уровнях. В документации IBM упоминается три типа параллелизма:

- Параллелизм на уровне ввода-вывода (I/O parallelism);
- Параллелизм при обработке запросов (Query parallelism);
- Параллелизм при использовании утилит обслуживания БД (Utility parallelism).

Параллелизм на уровне ввода-вывода представляет собой хорошо известные приемы параллельной работы (записи и чтения) с устройствами ввода-вывода, когда процесс может одновременно работать с несколькими устройствами ввода-вывода, что существенно ускоряет работу. В случае DB2 может, например, применяться назначение табличному пространству нескольких контейнеров, которые будут использоваться параллельно.

Параллелизм при обработке запросов объясняется возможностью разбивать запросы на подзапросы, а также обрабатывать несколько запросов параллельно в одном или нескольких разделах БД. При этом подразумевается параллельное использование ресурсов, доступных внутри раздела, и ресурсов разных разделов БД. Специфика построения БД и ее разделов для параллельного использования будет рассмотрена ниже.

Параллелизм при использовании утилит работы с БД совмещает параллельное использование вычислительных ресурсов разделов и за-

действие параллельного доступа к вводу-выводу. В данном случае утилиты работы с БД адаптируются таким образом, чтобы использовать параллельно ресурсы БД и системы с наибольшей эффективностью. Такой подход не является оригинальным и, наряду с IBM, используется другими производителями. Примерами могут служить утилиты параллельной загрузки (SQL loader) или резервного копирования (Rman) СУБД Oracle.

Применение параллелизма. Схемы БД с разделами

Самый простой пример системы с IBM DB2 – однопроцессорная система (uniprocessor system) с одним разделом БД. Очевидно, что в такой системе можно применить только параллельный доступ к устройствам хранения. Такая система в настоящее время является архаичной и не используется.

Некоторое время назад применялись массивно-параллельные (massively parallel processing – MPP) системы IBM DB2. Такие системы представляют собой кластеры из однопроцессорных компьютеров. Кластер MPP DB2 строится в соответствии с идеологией «shared nothing». На каждом узле организовывается раздел БД, с которым работает только данный узел. В кластере имеется узел-координатор, взаимодействуя с которым, пользователи работают с кластерной БД. Узел-координатор распределяет запросы по остальным узлам кластера, собирает результаты обработки и выдает пользователям конечные результаты (рис. 7).

Для соединения узлов кластера используется высокоскоростная сеть с низкой задержкой. В документах компании IBM приводятся примеры использования Ethernet и Infiniband в качестве такого соединения.

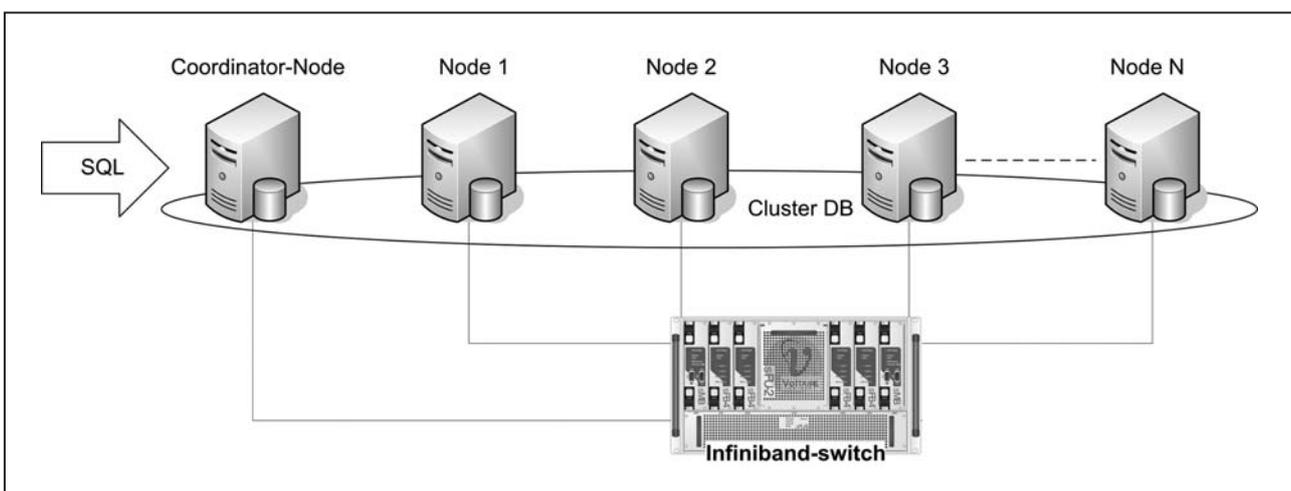


Рис. 7. Структура кластера MPP IBM DB2

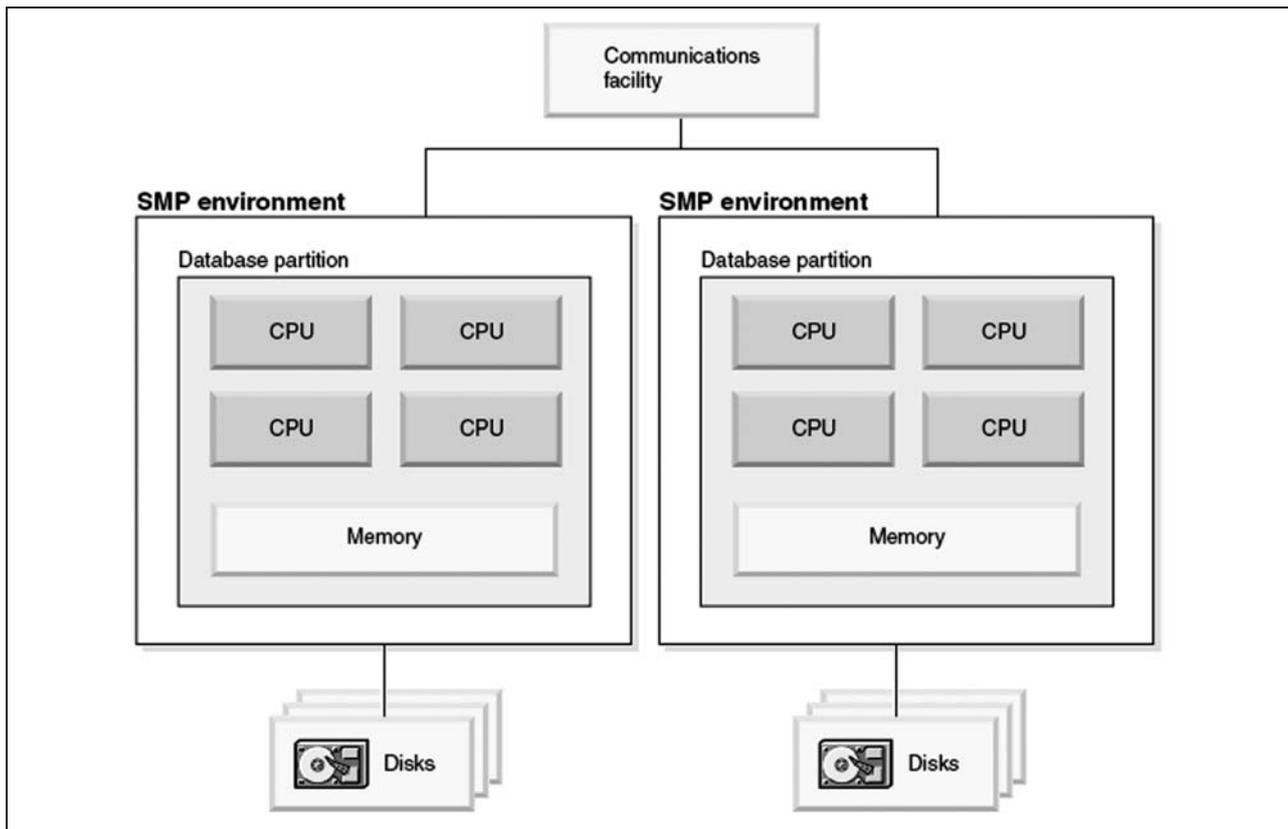


Рис. 8. Кластеризованная система DB2 (1)

В настоящее время для построения СУБД IBM DB2 как однораздельных, так и многораздельных, служат, в основном, многопроцессорные SMP (Symmetric Multiprocessing) машины. Многопроцессорные серверы – это системы с разделяемыми компонентами, такими, как память и устройства ввода-вывода. Таким образом, системы, построенные на их основе, в которых разделы БД используют более одного процессора, не являются в чистом виде системами «shared nothing». При построении таких систем нужно принимать во внимание возможность возникновения конкуренции за разделяемые ресурсы. Однако в данном случае такая конкуренция может возникать лишь локально, в пределах одного раздела, не затрагивая при этом напрямую всю базу.

Кроме того, есть возможность дальнейшего разделения БД в пределах одной SMP системы. Такая возможность носит название логического разделения (logical partitioning). Таким образом, теоретически, можно избежать конкуренции за использование общих ресурсов SMP-систем, т.к. при такой организации каждый раздел будет работать со своим участком памяти и своими устройствами ввода-вывода.

Примеры организации современных параллельных систем DB2 представлены на рис. 8, рис. 9. На рис. 8 показан кластер из систем SMP, где каж-

дый раздел БД занимает многопроцессорный узел кластера целиком. На рис. 9 показано логическое разбиение БД в пределах одной SMP системы. В качестве узла-координатора, обслуживающего

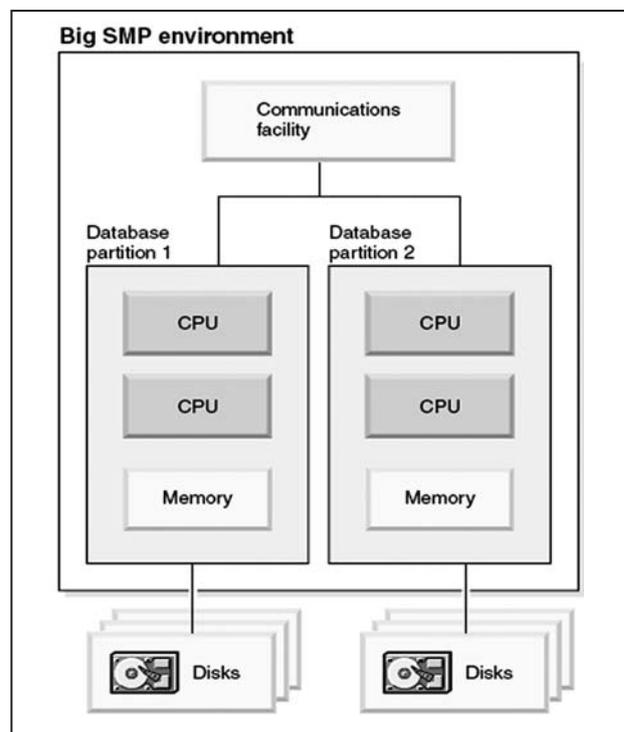


Рис. 9. Система DB2 с логическими разделами (1)

запросы в такой параллельной системе может выступать любой раздел БД, к которому осуществляется соединение клиентских сессий.

Организация доступа к параллельным средам

Доступ к таблицам. Hash partitioning

Хранение и доступ к таблицам многораздельной БД производится с помощью использования значений хэш-функций, связывающих части таблиц с разделами БД. Эта технология носит название hash-partitioning. Для пояснения можно рассмотреть таблицу БД, один столбец которой используется для вычисления хэш-функции и установления соответствия значения этой функции и узла, который будет работать с теми или иными строками этой таблицы. Такой столбец называется ключом распределения (distribution key). Схематически такая таблица изображена на рис. 10.

В принципе, любой столбец таблицы может использоваться в качестве ключа распределения, но очевидно, лучше использовать столбцы с большой дисперсией значений, а еще лучше — уни-

кальные значения для каждой строки. В таком случае можно распределить таблицы наиболее равномерно по произвольному числу узлов. И напротив, очень плохо подходят для этого столбцы с малой уникальностью значений. Так, например, таблица, в которой в столбце для ключа распределения может быть всего два значения, может распределиться лишь по двум узлам.

Типичным случаем использования ключа распределения является использование для этих целей первичного ключа таблиц. Механизм создания таблиц с hash partitioning заложен в средства языка SQL. Вот пример создания таблицы с единственным столбцом, который и выступает в качестве ключа распределения:

```
CREATE TABLE list (id INTEGER) PARTITIONING KEY(id) USING HASHING;
```

Данные из разделенных таблиц привязываются с помощью ключей распределения к конкретным разделам БД для их обработки.

Распределение табличных пространств. Группы разделов

Привязка таблиц к разделам осуществляется на уровне табличных пространств, в которые они

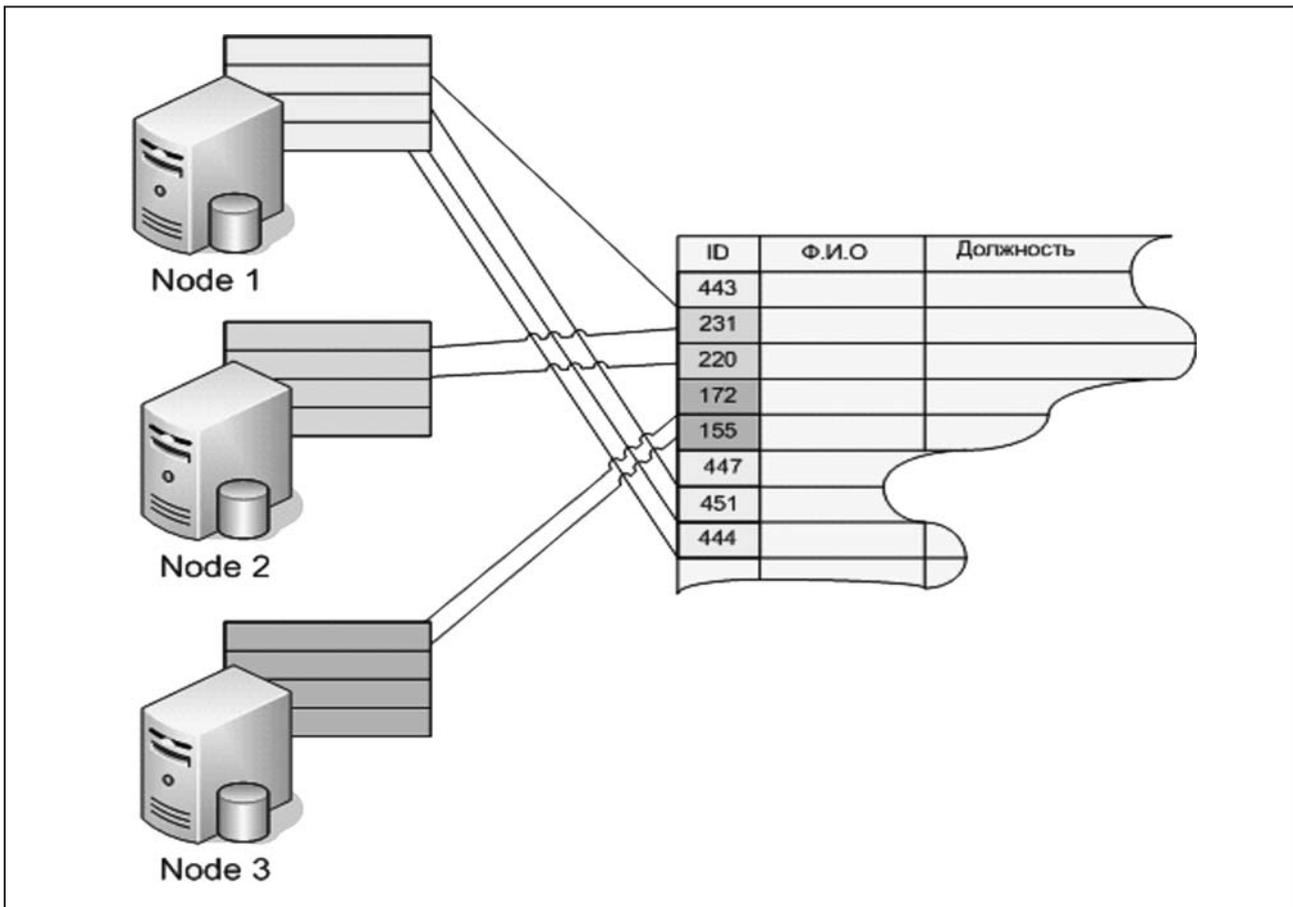


Рис. 10. Hash partitioning – распределенная обработка таблицы

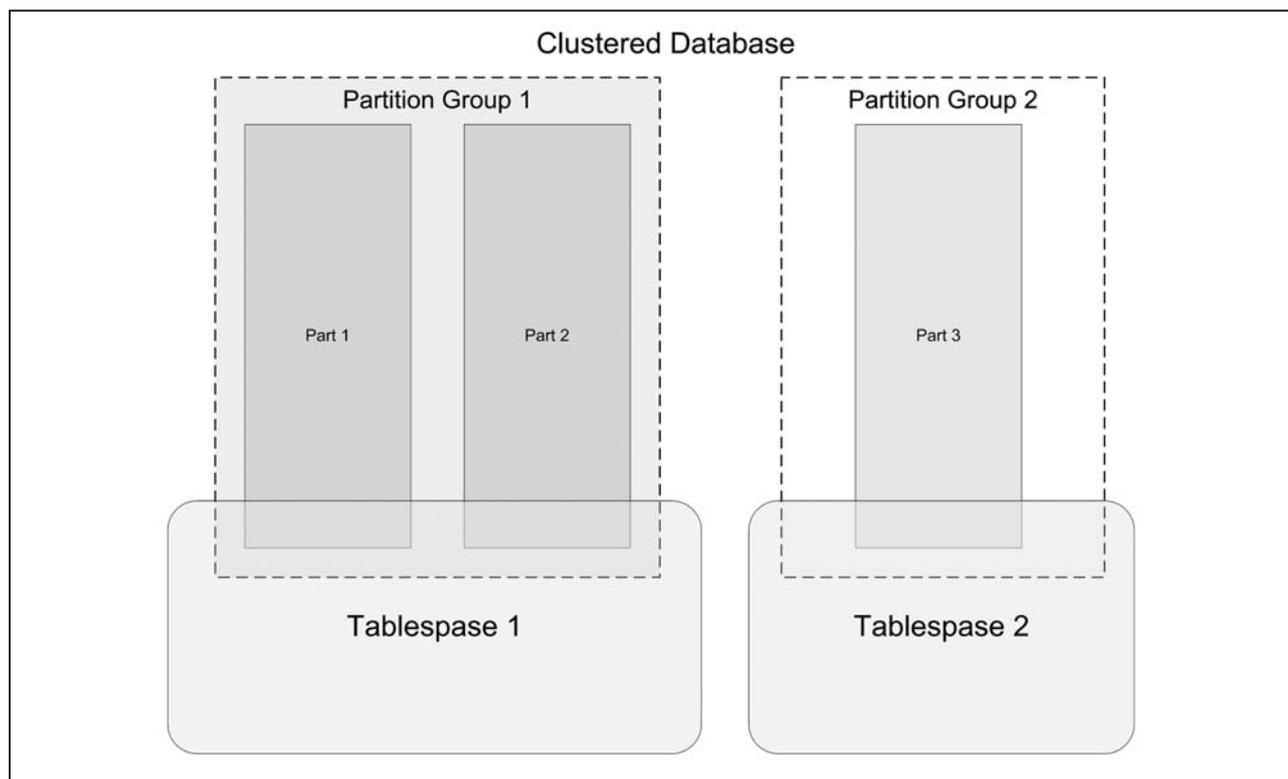


Рис. 11. Пример логической структуры многораздельной БД

входят. Для фактического разбиения табличных пространств по разделам БД и, в частности, узлам кластеризованной системы, применяют группы разделов (partition groups). Кроме этого, группы разделов могут служить для смыслового отделения данных БД друг от друга. Так, например, можно даже в пределах одной системы отделить системную часть БД от пользовательской, или в гибридных средах отделить OLTP-часть базы от части DSS.

При этом механизм масштабирования системы и распределения ресурсов для обработки конкретных данных весьма гибок. Таблицы создаются в табличных пространствах, которые, в свою очередь, принадлежат группам разделов БД. Если в кластеризованную систему добавился новый узел, его включают в нужные группы (alter database partition group ... add dbpartitionnum...), после чего нужно перераспределить данные в таблицах (redistribute database partition group ...) — содержимое всех таблиц во всех табличных пространствах, принадлежащих измененной группе разделов, будет распределено с учетом нового узла. Аналогичным образом происходит удаление узла или раздела из кластера или его назначение для обработки других данных БД.

Очевидно, что с помощью такого механизма можно избирательно распределять вычислительные ресурсы по отдельным частям БД, например, таблицам. Наиболее загруженные таблицы

можно разнести на большее число узлов кластера или выделить им больше процессорных ресурсов в случае единичной SMP-системы. Гипотетический пример распределения показан на рис. 11. Здесь компьютеры кластера (разделы) — part1-part3 объединены в группы разделов PartitionGroup1-2. Для более нагруженной части БД, расположенной в табличном пространстве Tablespace 1 используется группа разделов PartitionGroup1. Для менее нагруженной части используется группа разделов PartitionGroup2.

Операция перераспределения данных занимает определенное время и требует передачи существенного объема данных между узлами кластерной системы, что, как уже говорилось выше, определяет использование высокоскоростных соединений между узлами кластера. Такие операции должны планироваться заранее и выполняться во время низкой активности БД.

Карты распределения

Если в однораздельной БД задача нахождения нужного раздела для манипуляций с данными не стоит, то в многораздельной базе для хранения распределенных с помощью hash-partitioning таблиц необходимо определить раздел БД, к которому относятся обрабатываемые данные. Для этих целей дополнительно к механизму ключей расп-

ределения hash-partitioning вводят карты распределения (distribution map).

Карта распределения представляет собой одномерный массив фиксированной длины — на 4096 ячеек для многораздельных баз (для DB2 v.9.x). Для однораздельной БД карта тоже существует, но фактически представляет собой одну ячейку. Карты создаются для каждой группы разделов. Пример карты для группы из четырех разделов приведен на рис. 12.

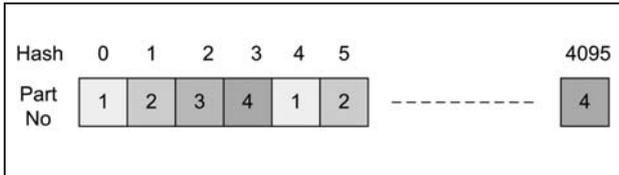


Рис. 12. Карта распределения

Адрес той или иной ячейки массива будет представлять собой значение хэша ключа распределения. Таким образом, для доступа к определенной части БД будет выбран тот или иной раздел.

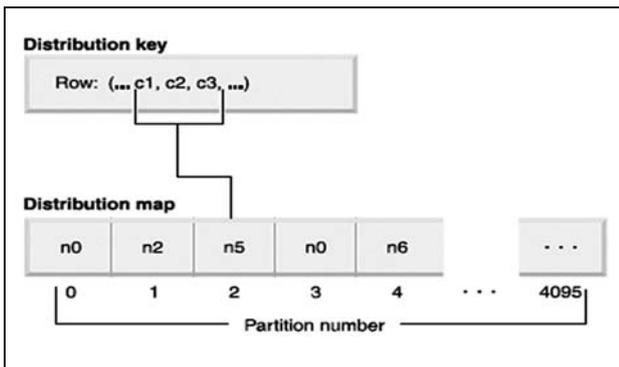


Рис. 13. Пример выбора раздела по карте распределения (1)

Отказоустойчивость

При взгляде на многораздельную СУБД IBM DB2 становится очевидной необходимость обеспечения специальных мер по отказоустойчивости этой сложной параллельной вычислительной системы. К сожалению, отказоустойчивость не является «врожденным» свойством таких систем и предполагает дополнительные весьма существенные расходы на аппаратные и программные средства. С другой стороны, в серьезных производственных средах цена потери одного из разделов БД может быть очень велика, и поэтому затраты на обеспечение отказоустойчивости, высокой надежности и доступности СУБД в таких случаях, как правило, необходимы и оправданы. Таким образом, в данном разделе приводится обзор воз-

можных решений по обеспечению отказоустойчивости и высокой доступности СУБД IBM DB2. Можно выделить два основных решения — кластеризация узлов с разделами БД и использование репликации системы с основной БД на систему с БД, находящуюся в «горячем» резерве.

Кластеризация узлов с разделами БД

Данный метод повышения отказоустойчивости и высокой доступности достаточно очевиден и подразумевает использование HA-кластеров для всех узлов СУБД. Таким образом, число компьютеров, используемых для СУБД в общем случае, возрастет в два раза.

В качестве кластерного ПО для узлов могут применяться и поддерживаются IBM следующие продукты:

- IBM HACMP для ОС AIX;
- Microsoft Cluster Server для Windows;
- Sun Cluster или Veritas Cluster Server для ОС Solaris.

Репликация БД

Репликация БД DB2 обеспечивается опцией «High Availability and Disaster Recovery» (HADR). Эта опция входит в состав наиболее полного пакета СУБД DB2 ESE и не входит в другие дистрибутивы продукта.

HADR позволяет создать реплику основной базы, которая будет находиться в состоянии «горячего резерва» по отношению к основной базе. В штатном режиме все взаимодействие осуществляется с основной БД. При этом к резервной базе применяются все изменения, которые были отражены в журналах транзакций основной базы.

Запросы поступают к основной БД, записываются там в журналы транзакций и посылаются на резервную БД, где тоже записываются в файлы журналов. Основная и резервная базы взаимодействуют на уровне TCP/IP. Это дает возможность разносить системы с основной и резервной БД территориально. Хотя для резервной БД принципиально возможно применение более простой и менее быстродействующей системы, IBM рекомендует использовать и для основного, и для резервного вариантов БД аналогичные системы. Существует три режима репликации между основной и резервной БД.

Синхронный режим (SYNC Mode). Этот режим характеризуется самой высокой надежностью и доступностью и самым высоким временем выполнения транзакций. Пользовательские транзакции обязательно должны записаться в файлы журналов на основной и на резервной системах. Только после этого запись транзакции

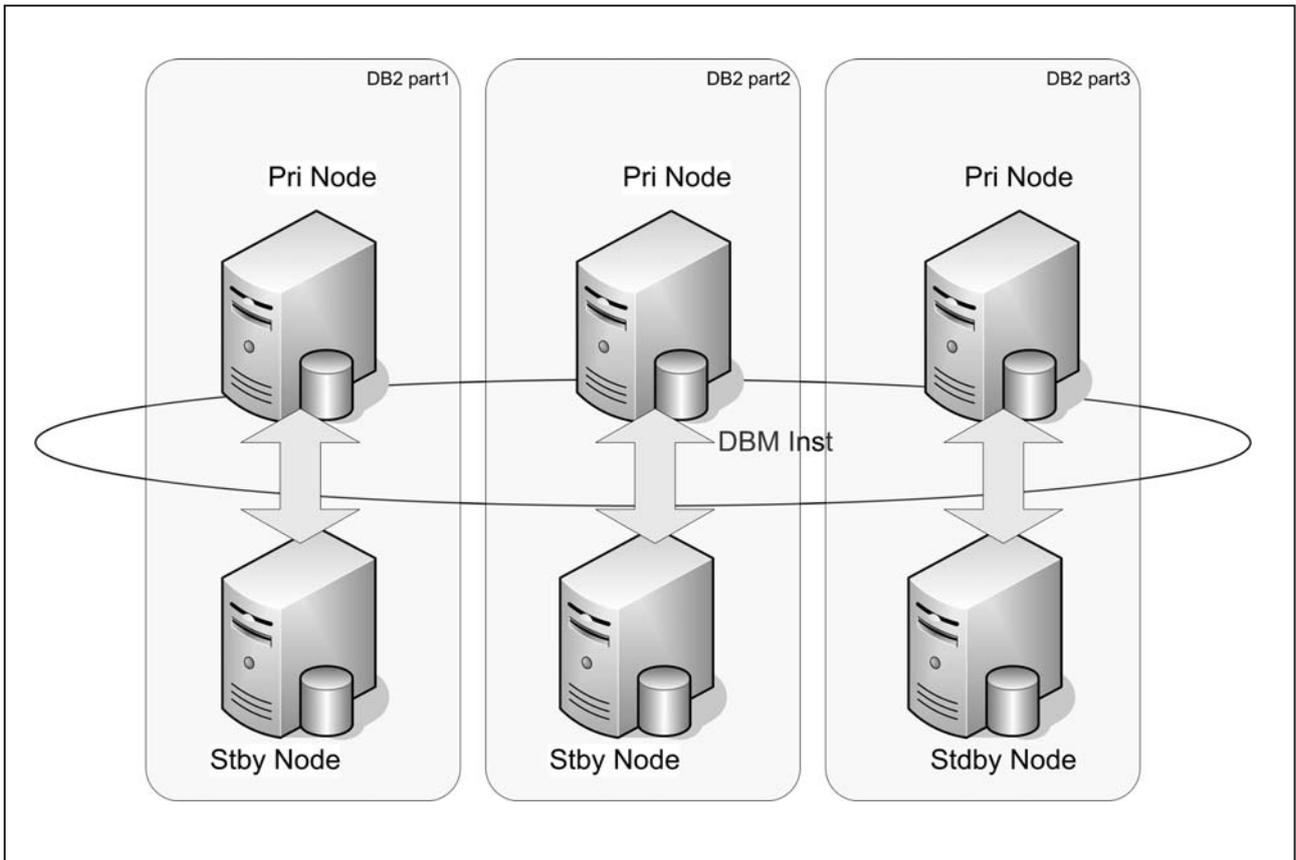


Рис. 14. Кластеризация узлов с разделами DB2

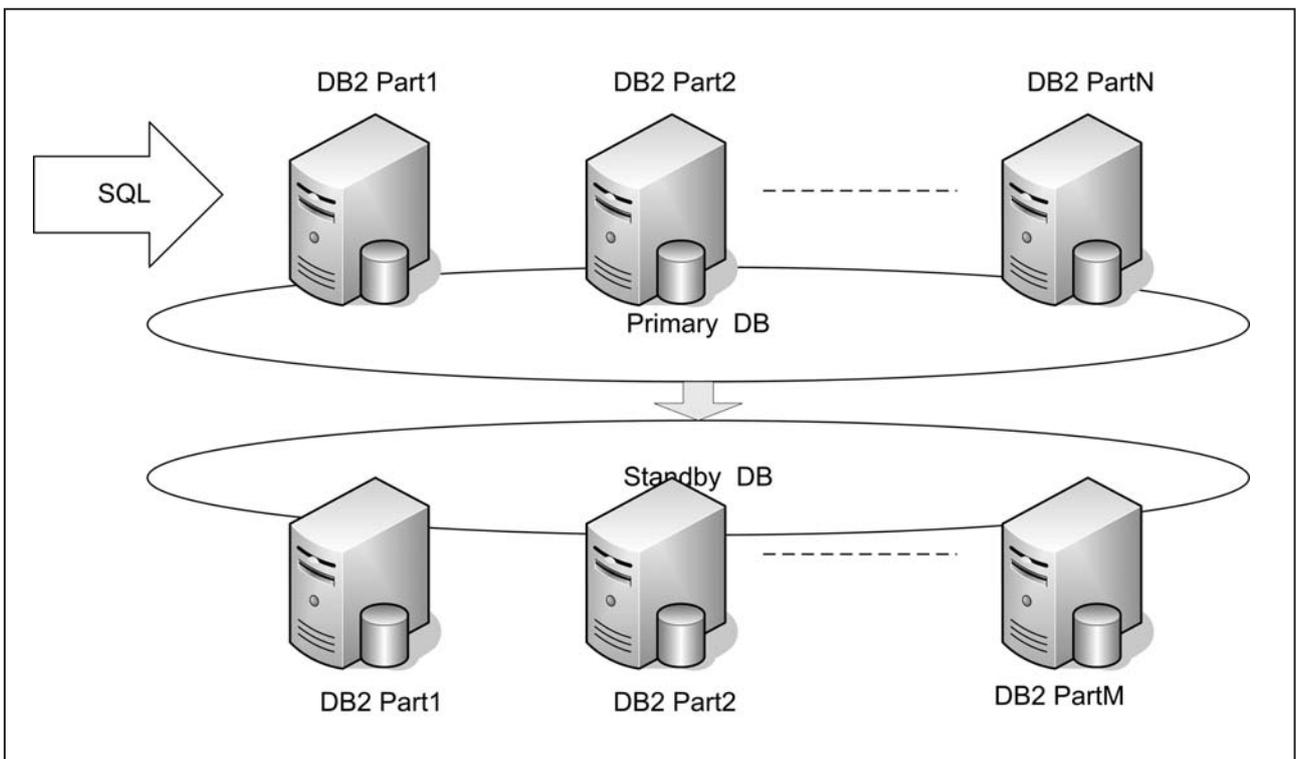


Рис. 15. Репликация баз DB2

считается успешной. Очевидно, что в случае аварии на основной БД, резервная база в любой ситуации будет полностью отражать состояние основной БД до аварии. Этот способ полностью исключает потерю данных.

Полусинхронный режим (NEARSYNC Mode). Чтобы повысить быстродействие и снизить время выполнения транзакций в большом числе случаев, можно применить этот режим. Здесь запись транзакции считается успешной, если она записана в файлы журналов в основной базе и записана хотя бы в буферы журналов на резервной базе.

Асинхронный режим (ASYNC Mode). Как следует из названия, в этом режиме запись транзакции считается успешной, если она записана в файлы журналов на основной базе и оказалась в TCP/IP стеке для отправки на резервную БД. Естественно, при такой организации репликации, в случае аварии на основной базе, существует возможность потери транзакций, которые еще не успели записаться в журналы резервной БД. Однако в ряде случаев такой режим репликации целесообразен, особенно при периодических значительных нагрузках на БД и территориально удаленной системы с резервной базы.

В аварийных ситуациях на основной БД резервная база берет на себя обязанности основной базы (failover или takeover). Все соединения автоматически перенаправляются на эту БД. Для этого используется функциональность Auto Client Reroute (ATR). Клиентам известен альтернативный адрес, на который нужно перенаправлять трафик в случае ошибок с текущим соединением. Когда клиент получает ошибку, он автоматически предпринимает попытку соединения с резервной БД. Если операция failover произошла удачно, и резервная БД взяла на себя обязанности основной базы, клиент установит соединение и продолжит работу.

Резервное копирование

Резервное копирование распределенной БД DB2 имеет много сходного с резервным копированием БД Oracle.

Встроенные возможности резервного копирования позволяют создавать образы (image) БД, раздела БД или табличных пространств и сохранять их на следующих носителях и СРК:

- в файлах;
- на дисковых и ленточных устройствах;

- Tivoli Storage Manager (TSM);
- другие СРК (например, Symantec NetBackup).

В базе существует режим архивирования журналов транзакций (archive log mode), при котором неактивные журналы архивируются и могут быть использованы для восстановления БД методом наката (rollforward recovery). Если журналы транзакций архивируются, то резервное копирование базы и табличных пространств может осуществляться в режиме online¹. Существует возможность задания двух альтернативных мест для заархивированных журналов — первичное и вторичное. В качестве таковых могут выступать перечисленные выше носители и СРК. Информация о времени проведения операций резервного копирования и восстановления, соответствии файлов журналов конкретным табличным пространствам и опциях и параметрах операций резервного копирования автоматически пишется в файлы истории, хранящиеся в дереве БД (рис. 16).

Полное восстановление БД в любом случае возможно только в режиме offline, однако, если включено архивирование журналов транзакций, то можно восстанавливать отдельные таблицы, табличные пространства и разделы в режиме online.

Операционные системы и аппаратные платформы для DB2

Текущая версия продукта — IBM DB2 ESE v 9.5, поддерживается на следующих платформах:

- AIX v. 5.3, 6.1 (64 bit);
- HP-UX v. 11.23.0505, 11.31;
- Linux:
 - Red Hat Enterprise Linux (RHEL) 4 Update 4;
 - Red Hat Enterprise Linux (RHEL) 5;
 - SUSE Linux Enterprise Server (SLES) 9 Service Pack 3;
 - SUSE Linux Enterprise Server (SLES) 10 Service Pack 1;
- Solaris 9,10 SPARC (64-bit);
- Solaris 10 x86-64;
- Windows Server 2003:
 - Standard Edition (64/32bit);
 - Enterprise Edition (64/32bit);
 - Datacenter Edition (64/32bit);
- Windows Server 2008:
 - Standard Edition (64/32bit);
 - Enterprise Edition (64/32bit);
 - Datacenter Edition (64/32bit);

¹ В отличие от Oracle функциональности online резервного копирования без архивирования журналов транзакций нет.

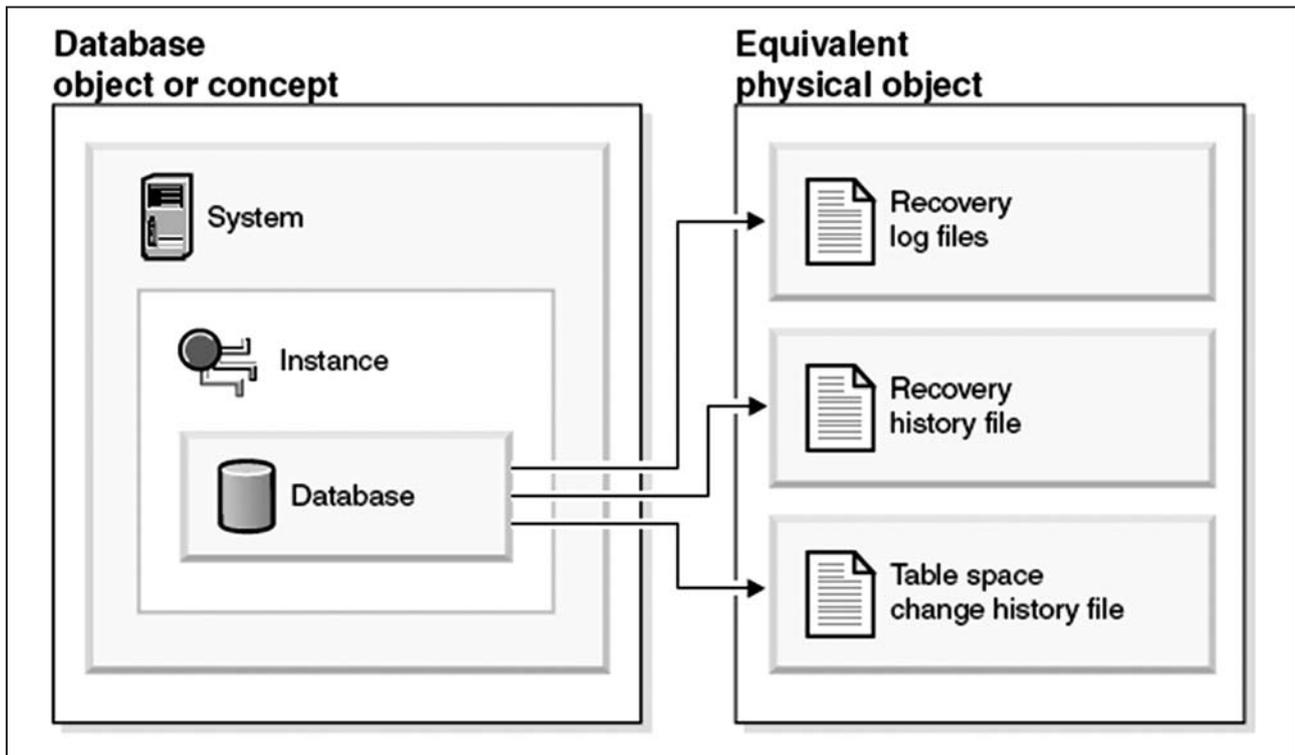


Рис. 16. Файлы истории операций резервного копирования – восстановления (1)

MySQL

В известном продукте MySQL возможность кластеризации появилась тоже достаточно давно. Производители заявляют, что кластер также имеет архитектуру «shared nothing», к которой действительно можно с небольшими оговорками отнести продукт.

Изначально сетевое взаимодействие между узлами MySQL было разработано в целях параллельной обработки и повышения надежности функционирования системы с таблицами БД, размещенными непосредственно в ОЗУ серверов. Такое направление покажется логичным, если вспомнить, что изначально основное применение MySQL – СУБД для организации web-порталов, онлайн-магазинов и других подобных приложений для Интранет и Интернет. Основные требования для подобных систем – быстрота и надежность обработки значительных потоков «легких» OLTP-транзакций. Организация таблиц БД и индексов в памяти серверов существенно убаьстряет работу системы и позволяет ей справляться с большим количеством конкурентных запросов пользователей. Однако целостность данных в такой системе будет подвергаться гораздо большей угрозе, чем в системе с таблицами в энергонезависимых ЗУ, а это неприемлемо для современных приложений. Поэтому для обеспечения должной

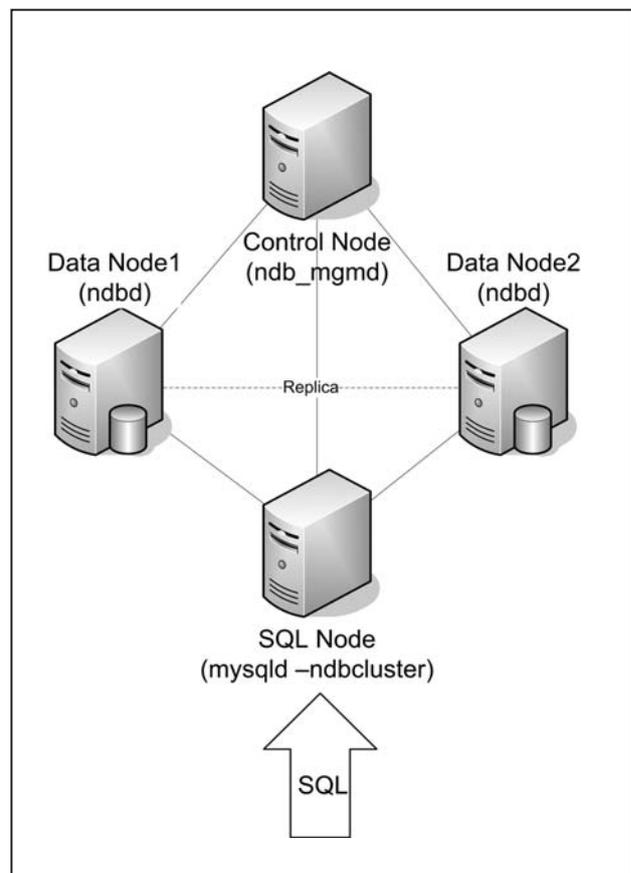


Рис. 17. Минимальная рекомендуемая конфигурация кластера MySQL

целостности данных таблиц в оперативной памяти и был разработан так называемый NDB-кластер (Network Database Cluster), применяемый в MySQL. Структура такого кластера приведена на рис. 17.

Для примера здесь приводится минимальная рекомендованная производителем ПО конфигурация кластера. В данном случае управляющий узел не производит ничего, кроме изначальной конфигурации кластера и его реконфигурации в ситуациях «split brain». Узел запросов (SQL node) работает с кластерной БД, принимая запросы пользовательских приложений. Основу кластера составляют узлы данных (data nodes), которые непосредственно содержат БД и производят обработку попавших на них клиентских запросов.

«Узел» в понятии MySQL — это в общем случае процесс на сервере. Таким образом, компьютеров в системе на рис. 17 может быть меньше четырех. Управляющий узел в системе один, узлов другого типа может быть много. Так два компьютера в системе могут служить для приема запросов и их обработки (выступать одновременно и узлами запросов, и узлами данных), или можно совместить на одной машине функции управляющего узла и узла запросов. В любом случае, не рекомендуется использовать менее трех машин в системе, т.к. это существенно снизит надежность кластера — появится возможность потери одновременно и узла управления, и узла данных.

Работа такого кластера напоминает операцию синхронной репликации: запросы, изменяю-

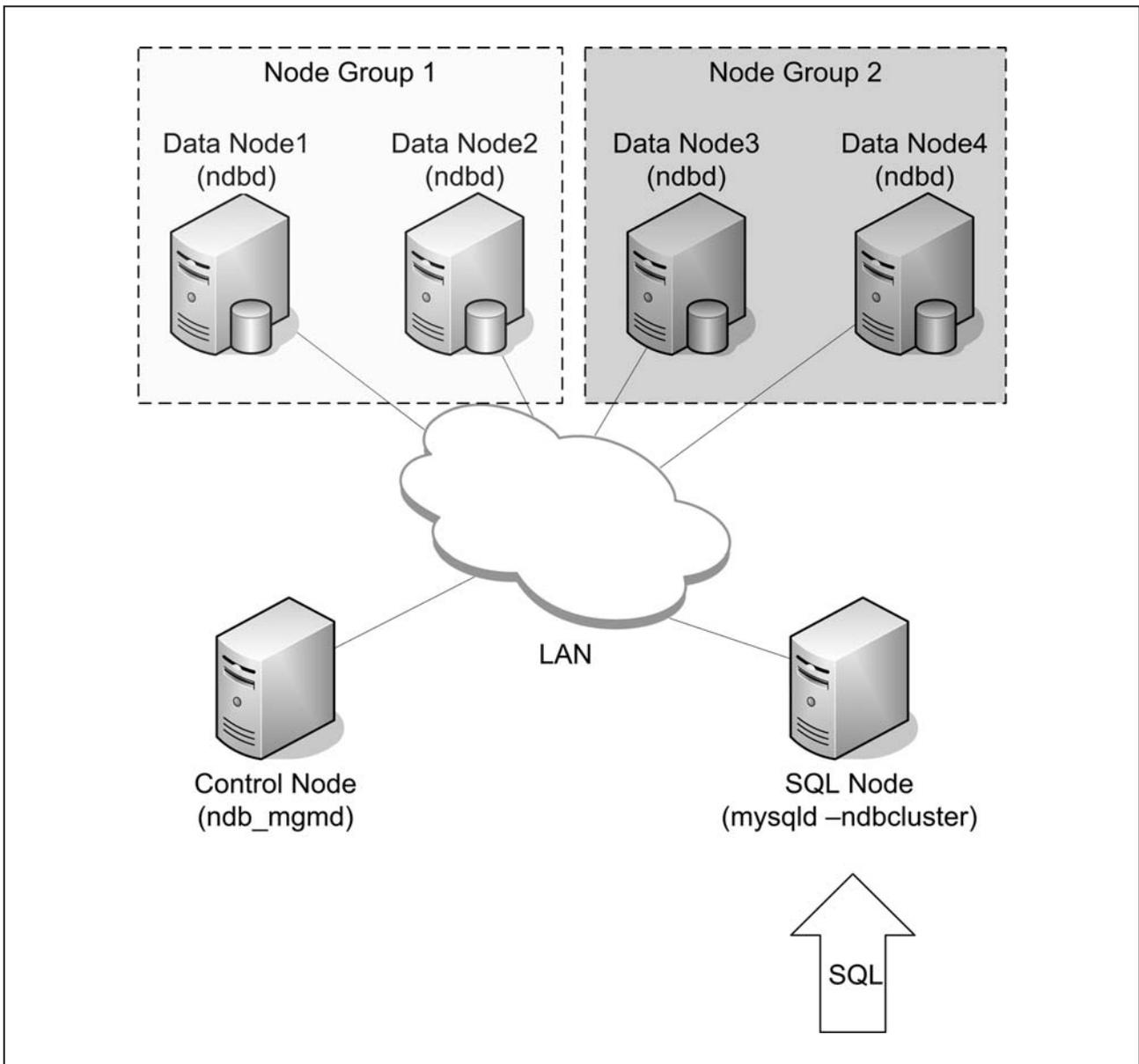


Рис. 18. Кластер MySQL с двумя группами узлов

щие БД, немедленно исполняются и на соседнем узле. Запросы, не изменяющие БД, могут выполняться параллельно на двух узлах.

При ситуациях потери одного из узлов в системе работоспособность не теряется, и целостность данных не нарушается. Таким образом, если принимать во внимание расположение таблиц кластерной БД в оперативной памяти, получается довольно быстродействующая и надежная конструкция.

Недостаток такой конструкции, и особенно при применении подобных систем для корпоративных БД, очевиден — сравнительно малый объем памяти для таких таблиц и необходимость специальной организации БД для долговременного хранения данных из этих таблиц на дисковых устройствах и их доступности по требованию.

Для увеличения размера кластерной БД, а также и быстродействия при конкурентных операциях, изменяющих данные в MySQL, может применяться разбиение БД на фрагменты и соответствующие группы узлов (node groups). Число резервных узлов — «реплик» задается в параметре конфигурации кластера и на настоящий момент не может быть больше четырех. На рис. 17 показан кластер с числом реплик, равным двум. Таким образом, если заданное число реплик, например, 2, а узлов данных в кластере — 4, то произойдет фрагментация БД на две группы узлов, в каждой из которых будет по две реплики (рис. 18). При этом каждая группа узлов будет обрабатывать свою часть данных БД, что, несомненно, положительно скажется как на повышении быстродействия при операциях с изменением данных, так и на суммарном возможном объеме таблиц в кластере. Таблицы в таком кластере при создании автоматически разбиваются на разделы (partitions), каждый из которых обрабатывается соответствующей группой узлов. Такой кластер также сохранит работоспособность при потере одного из узлов.

В поздних версиях продукта (5.1 и выше) в подсистеме NDB-engine появилась-таки возможность обслуживания наряду с данными в ОЗУ данных на дисковых устройствах. Это обстоятельство, безусловно, способствует продвижению БД MySQL на рынок корпоративных СУБД. Однако при работе с табличными пространствами и таблицами на дисках существует ряд неудобств, возможно временных:

- В таблицах на дисках данные переменной длины поддерживаются не полностью — данные вне зависимости от фактической длины будут расходовать пространство, требуемое для максимальной определенной для них длины.

- Не предусмотрено операции освобождения неиспользуемых таблицами блоков расширения (extents). Единственная операция, которая может сократить число таких блоков, — полное уничтожение таблицы (DROP) и создание ее заново.
- Табличные пространства не поддерживают автоматическое расширение (autoextend).

Hypertable

В этом разделе пойдет речь о, пожалуй, самой нетрадиционной кластерной БД из всех рассматриваемых. Эта база конструктивно не обладает конкурентной производительностью и надежностью хранения данных с БД, рассмотренными в данном документе выше. Несмотря на это, ее стоит рассмотреть, т.к. подобная конструкция оказывается интересна с точки зрения расширяемости и высокой доступности сервиса, где в определенных случаях она может составить хорошую конкуренцию более известным схемам построения БД. Прежде всего, такими случаями являются всевозможные поисковые системы с периодическим обновлением БД поиска. Для таких систем важно, что база всегда доступна для обновления информации по поиску. По причине периодического обновления информации в БД для этих систем приемлемо отсутствие отдельных элементарных частей БД в течение некоторого промежутка времени.

База Hypertable создана подобно распределенным БД Google (BigTable), используемым для хранения тэгов и индексов страниц Internet-сайтов. В качестве серверов БД выступает большое количество компьютеров общего назначения (PC), на каждом из которых хранится своя часть базы. Основное отличие от хранения других БД в том, что в данном случае на сервере БД необязательно хранить строки или столбцы таблиц целиком. Структура базы напоминает ячейки в многомерном адресном пространстве, и каждая ячейка может иметь модификации по времени. Такие ячейки и хранят компьютеры в распределенной БД Hypertable. Первое измерение — это идентификатор строки (row id). Он же является первичным ключом, в соответствие с которым происходит сортировка, хранение и запросы к данным hypertable. Второе измерение — это столбец.

Таб. 1. Пример таблицы БД документов для поиска

ID	Имя	Кр. содержание	Подразделение	Тэг1	Тэг2
1	Модернизация SAN в ОАО «Феникс»	Этот документ рассказывает...	ОПВК	SAN	ПЗ
2	Обзор серверов на базе новых процессоров	Введение: Новые технологии...	ГПТ		
3	Установка VxVM на серверах с нетрадиционной архитектурой	В этом документе описан процесс инсталляции и...	СЦ	VxVM	

Третье измерение – подкласс столбца. Благодаря подклассам можно устанавливать для ячеек произвольные дополнительные свойства. Подклассов столбца может быть неограниченно много. Чет-

вертое измерение – время изменения записи (timestamp).

Таким образом, Hypertable – четырехмерная распределенная БД, не являющаяся реляционной БД.

Таб. 2. Хранение данных в БД Hypertable

Row ID	Key			Value
	Col ID	Col Qualifier	Timestamp	
1	Имя		10.03.2009 12:30	Модернизация SAN в ОАО «Феникс»
1	Имя		10.03.2009 12:15	Модернизация SAN в ОАО «Феникс»
1	Кр. Содержание		10.03.2009 12:30	Этот документ рассказывает...
1	Кр. Содержание		10.03.2009 12:15	Главной целью этого документа является...
1	Подразделение		10.03.2009 12:30	ОПВК
1	Подразделение		10.03.2009 12:15	ОПВК
1	Тэг	1	10.03.2009 12:30	SAN
1	Тэг	1	10.03.2009 12:15	SAN
1	Тэг	2	10.03.2009 12:30	ПЗ
1	Тэг	2	10.03.2009 12:15	ПЗ
2	Имя		10.03.2009 12:30	Обзор серверов на базе новых процессоров
2	Имя		10.03.2009 12:15	Обзор процессоров, установленных...
2	Кр. Содержание		10.03.2009 12:30	Введение: Новые технологии...
2	Кр. Содержание		10.03.2009 12:15	Введение: Новые технологии...
2	Подразделение		10.03.2009 12:30	ГПТ
2	Подразделение		10.03.2009 12:15	ГПТ
3	Имя		10.03.2009 12:30	Установка VxVM на серверах с нетрадиционной архитектурой
3	Имя		10.03.2009 12:15	Установка VxVM на серверах с нетрадиционной архитектурой
3	Кр. Содержание		10.03.2009 12:30	В этом документе описан процесс инсталляции и...
3	Кр. Содержание		10.03.2009 12:15	В этом документе описан процесс инсталляции и...
3	Подразделение		10.03.2009 12:30	СЦ
3	Подразделение		10.03.2009 12:15	СЦ
3	Тэг	1	10.03.2009 12:30	VxVM
3	Тэг	1	10.03.2009 12:15	Veritas

Ниже рассматривается гипотетическая таблица для поиска документов и ее реализация в «классическом» виде и hypertable. В таб. 1 на стр.17 приведен пример привычного хранения данных БД в виде таблицы с данными, которую характеризуют строки и столбцы. Одно или несколько полей такой таблицы может являться ключевым.

В hypertable данные фактически располагаются на компьютерах в виде отсортированных пар «ключ-значение». Ключ в данном случае будет содержать все четыре измерения БД:

- Идентификатор строки;
- Идентификатор столбца;
- Подкласс идентификатора столбца;
- Время занесения / обновления (timestamp).

Сортировка производится по ключу и по времени создания записи (timestamp). Таким образом, раньше всех располагается наиболее «свежая» запись «ключ-значение». Фрагмент такой организации хранения для рассмотренного ранее примера гипотетической БД приводится в таб. 2 на стр. 17.

Очевидно, для хранения таких данных физически на дисках многих компьютеров потребу-

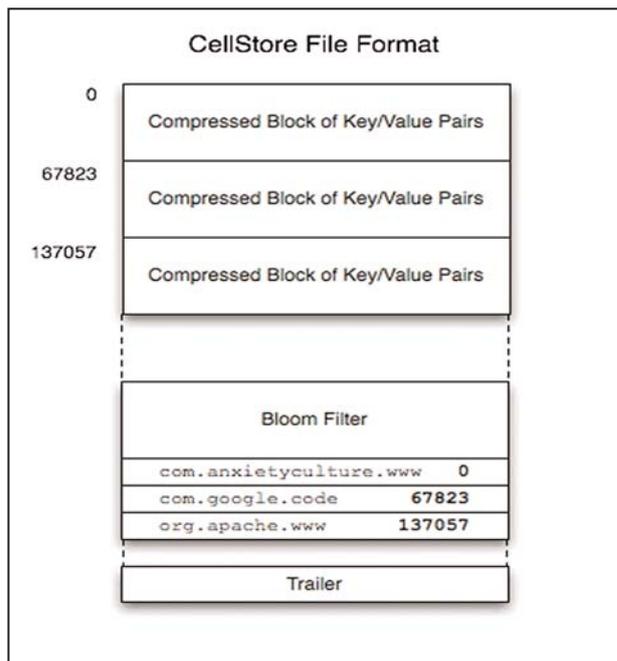


Рис. 19. Формат хранения «ячейки» (3)

ется распределенная файловая система² и управление конфигурацией и записью в БД.

Данные хранятся в файлах, называемых CellStore, в виде, представленном на рис. 19, и

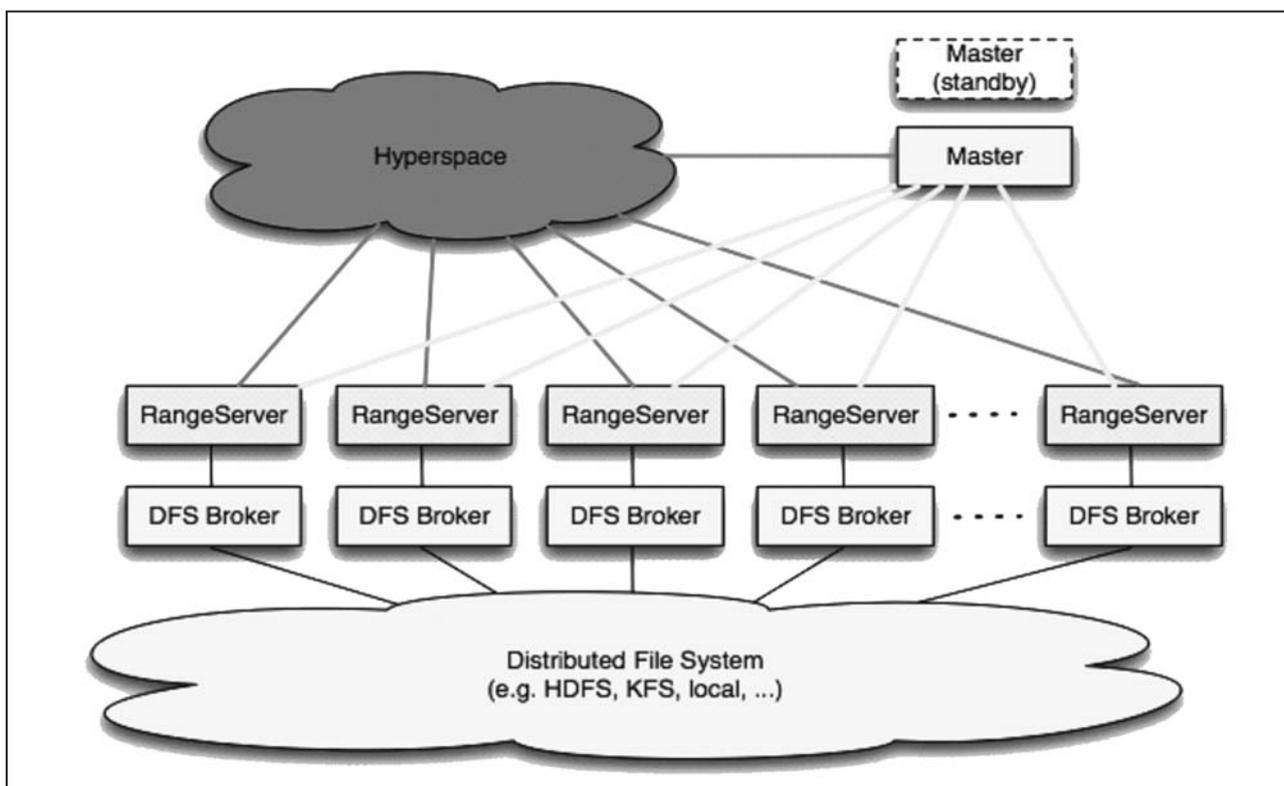


Рис. 20. Структура БД hypertable (3)

² hypertable ориентируется на Hadoop DFS, хотя в этом качестве, как заявляют производители, может выступать любая распределенная файловая система.

представляют собой набор сжатых блоков размером 65К. В конце каждого блока имеется индекс, представляющий собой пару последнего ключа в блоке и смещение. Эти индексы грузятся в память (CellCache) при обращении к CellStore.

Структура СУБД hypertable показана на рис. 20. Данные хранятся на серверах, называемых Range Server. Каждый range server отвечает за хранение своего участка с данными (range).

Таким образом, таблицы делятся на участки данных, располагающиеся на серверах данных. При создании и конфигурировании БД для всех таких серверов определяется порог заполнения данными, при котором половина информации будет передаваться на следующий «чистый» range server, высвобождая половину заполненного пространства.

Конфигурированием всей системы и переносом данных по мере заполнения серверов с данными занимается мастер-сервер (Master). Во время работы, не требующей реконфигурации системы, обязательное наличие мастера не требуется. Но так или иначе в будущих версиях hypertable планируется кластеризация мастера.

Компонент системы, называемый Hyperspace, управляет доступом к данным системы. Он обеспечивает нахождение нужных данных в пространстве серверов, координацию зап-

росов и управление блокировками. В настоящий момент компонент представлен единственным сервисом на одном компьютере. В ближайшем будущем планируется его разнесение на множество машин и кластеризация для большей отказоустойчивости.

Sybase ASE

В разговоре про кластеризованные БД нельзя не упомянуть Sybase ASE 15 CE (Cluster Edition). Компания Sybase стремится не отставать от конкурентов и в 2008 году выпустила свою кластеризованную параллельную БД (рис. 21). Основной целью создания такой системы служит достижение гибкой масштабируемой среды, обладающей высокой производительностью и отказоустойчивостью.

Продукт документирован пока весьма скудно. Из листовок можно заключить, что кластер Sybase строится подобно Oracle RAC. Очевидно, ему будут также присущи типовые недостатки таких систем — конкуренция за общие ресурсы.

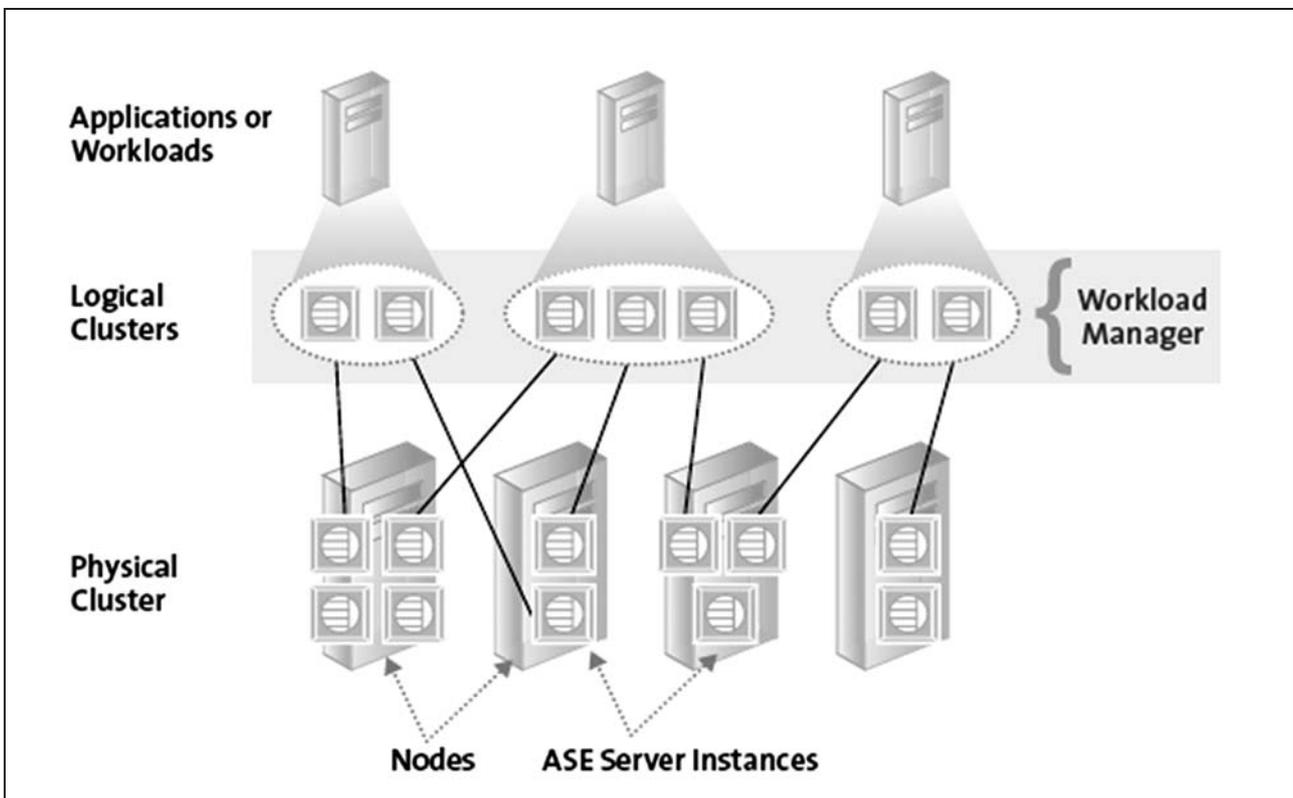


Рис. 21. Высокоуровневая структура кластера Sybase ASE (2)

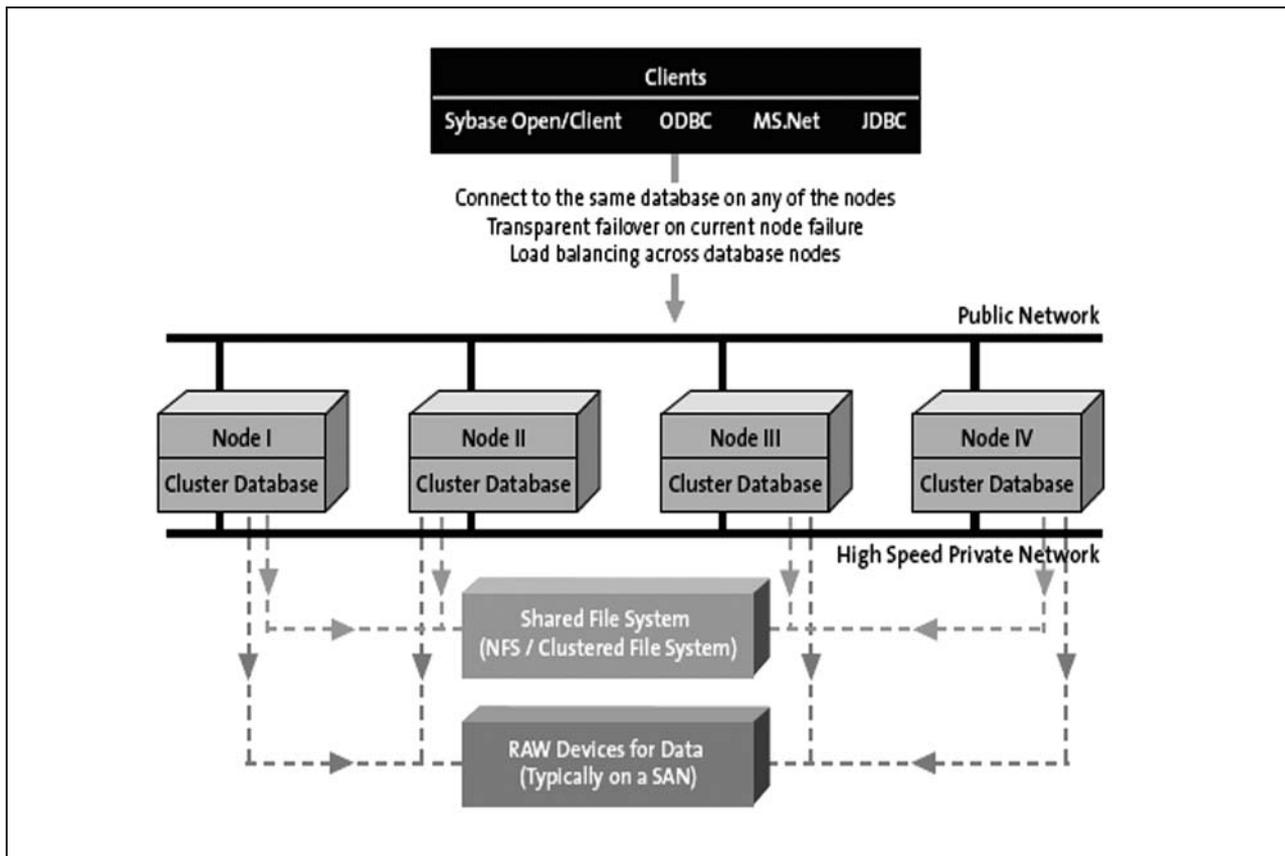


Рис. 22. Схема построения кластерной БД Sybase ASE (2)

Однако из присутствующего небольшого объема документов можно заметить, что основной упор делается компанией не на параллельную обработку, а на достижение отказоустойчивости и обеспечение среды для отдельных БД, находящихся под единым управлением.

Продукт Sybase ASE используется на корпоративном рынке достаточно давно, наряду с такими продуктами, как: Oracle, IBM DB2, MS-SQL. Он изначально обладает всеми необходимыми чертами корпоративной БД — механизмом журналирования транзакций, разбиением на разделы, поддержкой «сырых» устройств (raw devices), возможностями резервного копирования в режиме online и пр. Таким образом, у Oracle RAC появляется еще один конкурент, весьма сходный по функциональности. И хотя этот продукт еще весьма молод, принимая во внимание ценовую политику Oracle, можно предположить, что ASE CE займет свою нишу на рынке кластеризованных БД.

Для Sybase ASE 15 CE поддерживаются следующие платформы:

- Sun Solaris SPARC 9,10 (64 bit);
- SuSE Linux 9,10 (x86-64);
- RHEL 4,5 (x86-64);
- AIX v6.1 (AIX64);
- HPUX v11.23, 11.31(IA64).

Exasol и Paracel

В последнее время (2007-2008гг) в отчетах по тестированию аналитических СУБД на сайте TPC можно наблюдать двух новых лидеров. Это СУБД от компаний Exasol и Paracel. На сайте приводятся результаты испытаний довольно крупных систем на их основе. К сожалению, компании пока не предоставляют в общий доступ техническую документацию по системам, поэтому сведения, приведенные в этом разделе, имеют весьма поверхностный и общий характер. Они базируются на материалах брифингов и семинаров, посвященных СУБД DSS, листовках компаний и небольших интервью, найденных в сети.

Обе системы представляют собой БД, ориентированные на столбцы (column oriented DB). В отличие от традиционного подхода, когда базовым элементом всех таблиц является строка (row-oriented DB), в этих системах базовым элементом является столбец. По мнениям специалистов этих компаний, такой подход оптимизирует БД для задач Business Intelligence (BI) и DSS, т.к. упрощает операции выборки данных, их объединений и представлений, не требуя полной выборки и анализа строк. Он также позволяет более эффективно распределять участки таблиц между узлами

кластеризованной системы, выполненной в идеологии «shared nothing».

Обе системы строятся именно в этой идеологии и представляют собой массивно-параллельные (МРР) вычислительные кластеры с распределенной памятью (Distributed memory). Таблицы в системах распределяются по узлам при помощи технологии hash-partitioning, подобно системам с IBM DB2 (3.3.1) или с MySQL (4).

Также примечательно, что обе системы в качестве аппаратных платформ используют серверы архитектуры x86-64.

Компании Paracel и Exasol вышли на рынок ИТ сравнительно недавно, уже после 2000 года, но, тем не менее, активно развиваются и соперничают с именитыми и признанными производителями БД.

Продуктами этих компаний активно интересуются производители ПО аналитических и BI-систем, такие, как: Business Objects (SAP), Cognos (IBM), MicroStrategy, SAS и др.

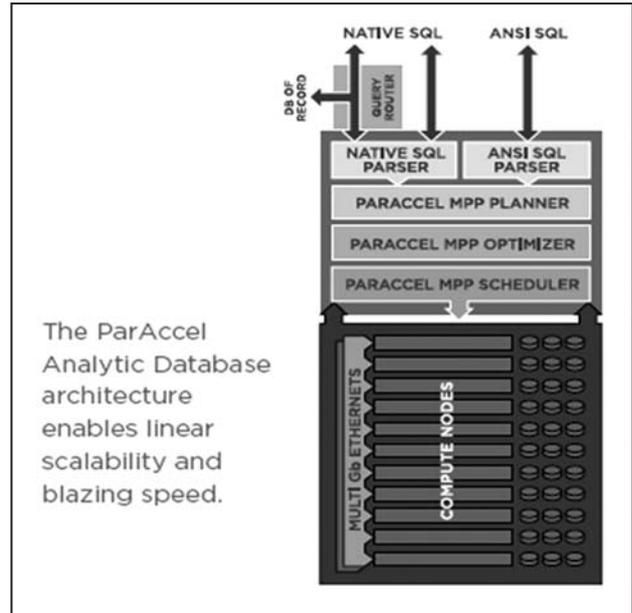


Рис. 23. Укрупненная структура СУБД Paracel Analytic Database (6)

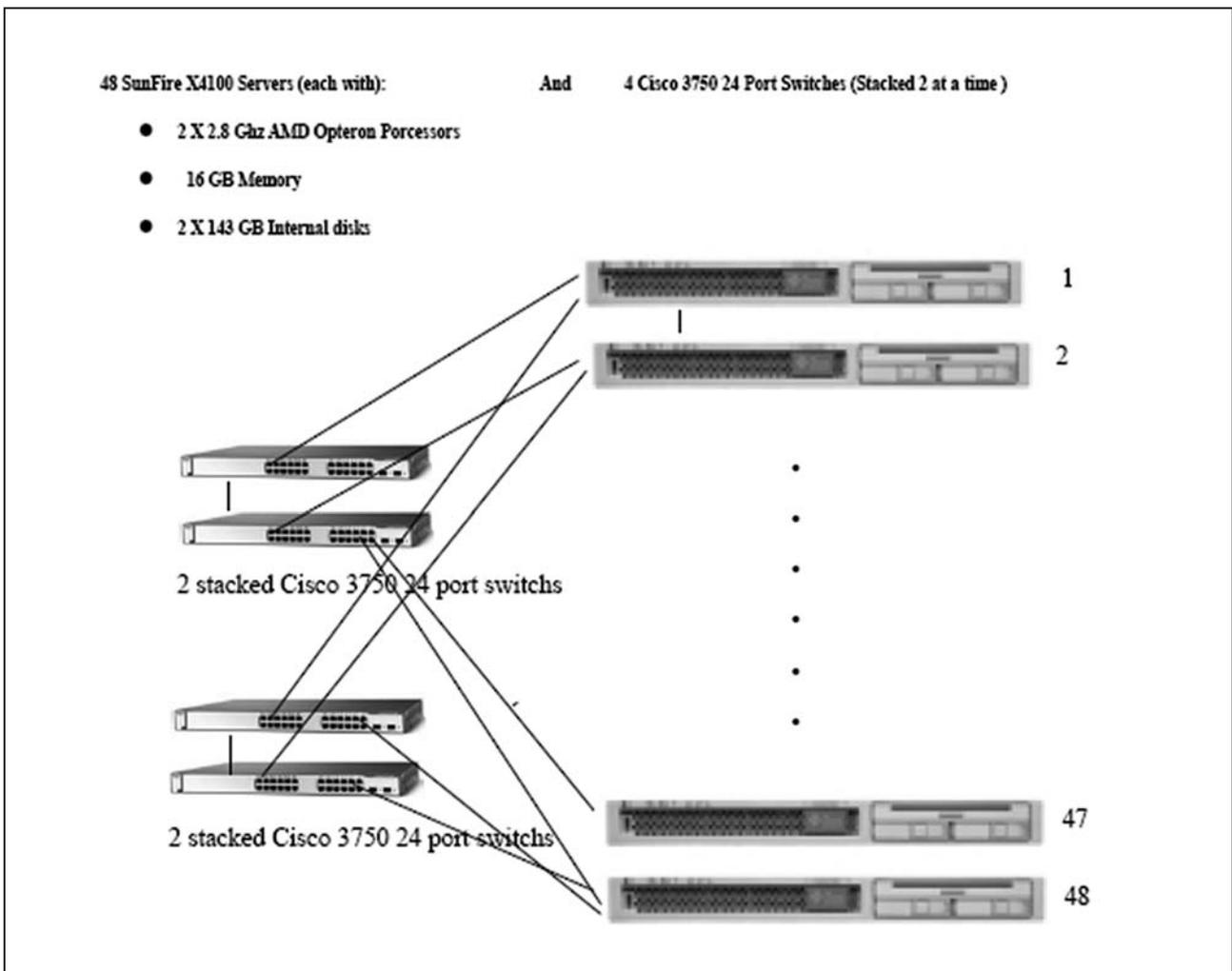


Рис. 24. Кластер Paracel Analytic Database для тестов TPC-H (5)

Paracel

Компания Paracel представляет на рынке свой продукт для построения СУБД DSS — Paracel Analytic Database (PADB). Продукт задумывался для использования в качестве самостоятельных решений BI и DSS, либо в составе с другими комплексами СУБД, такими, как Oracle или MS SQL. Среди характерных черт продукта можно выделить:

- Использование стандартного оборудования для кластера MPP и соединения между узлами кластера (используется обычная среда Gigabit Ethernet). Это существенно упрощает эксплуатацию и снижает затраты на оборудование и его поддержку.
- Использование стандартной ОС. В данный момент известны применения продукта с ОС Red Hat Linux Enterprise Server.
- Адаптивная компрессия данных — при записи на диск или передаче другим узлам данные автоматически сжимаются. Степень компрессии варьируется и может достигать 4. Это позволяет увеличить скорость операций ввода-вывода и сэкономить пропускную способность междуузловых соединений.
- Возможность хранения данных и выполнения операций полностью в оперативной памяти. Наиболее востребованные данные, а при достаточном объеме оперативной памяти, и все данные БД, необходимые для анализа, могут располагаться в ОЗУ серверов, не требуя операций ввода-вывода и очень сильно увеличивая производительность.
- Использование выделенного узла-контроллера для управления СУБД и обменом с клиентами.
- Технология AMIGO. Возможность использования продукта в качестве аналитической подсистемы в составе систем с другими

СУБД. Возможность параллельного использования ПО и постепенного перехода с других систем на PADB. Эта технология представляет собой некий маршрутизатор запросов, который на основании настроек и анализа запросов автоматически пересылает запросы на обработку либо в PADB, либо в другие СУБД, работающие параллельно с PADB.

Самая крупная система, участвовавшая в тестах TPC-H, имела размер БД 1ТВ и располагалась на 48 узлах кластера-суперкомпьютера, в качестве которых выступали серверы Sun X4100 с процессорами AMD-Opteron 2,8GHz и 16GB оперативной памяти. В качестве хранилища для БД использовались внутренние диски серверов, объединенных попарно в зеркальные тома на каждом сервере. Аппаратная конфигурация системы очень простая. Она приводится на рис. 24. Общая цена конструкции получилась порядка \$1,4 млн., большую часть из которых занимает лицензия на СУБД (\$1 млн.). Стоимость лицензии на СУБД вычислялась из расчета 1000 долларов на 1GB объема БД.

По заявлениям компании продукт обладает линейной масштабируемостью, что, в общем-то, подтверждают результаты тестов TPC-H.

В 2007 году в тестах TPC-H PADB заняла лидирующую позицию среди кластерных и одиночных систем с БД объемом 1ТВ по показателям «производительность» и «цена/производительность», а в 2008 году уступила пальму первенства по этим параметрам продукту компании Exasol .

Exasol

Немецкая компания Exasol, основанная в 2006 году, занимается разработкой высокопроизводительной масштабируемой БД Exasolution, занявшей в 2008 году лидирующую позицию по основным показателям тестов TPC-H. Эта БД имеет много общего с PADB, рассмотренной в предыдущем подразделе:

- Организация с ориентацией на столбцы (column oriented).
- Построение в виде кластера shared-nothing с распределенной памятью.
- Возможность манипулирования данными «полностью в ОЗУ».
- Использование аппаратной платформы x86-64 для узлов кластера.

Однако есть и ряд отличий, некоторые из которых весьма существенны:

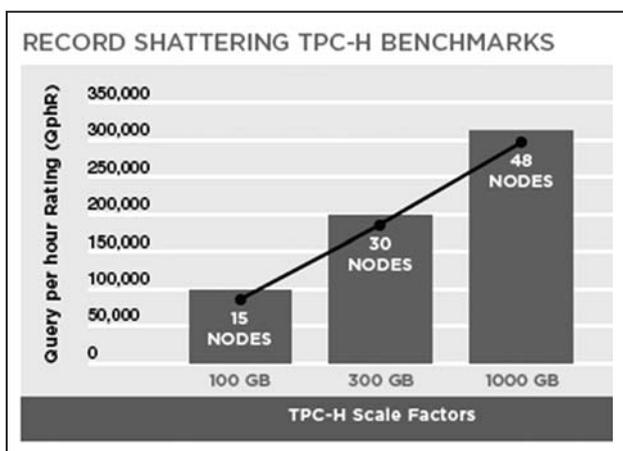


Рис. 25. Масштабируемость системы Paracel Analytic Database по данным тестов TPC-H (5)

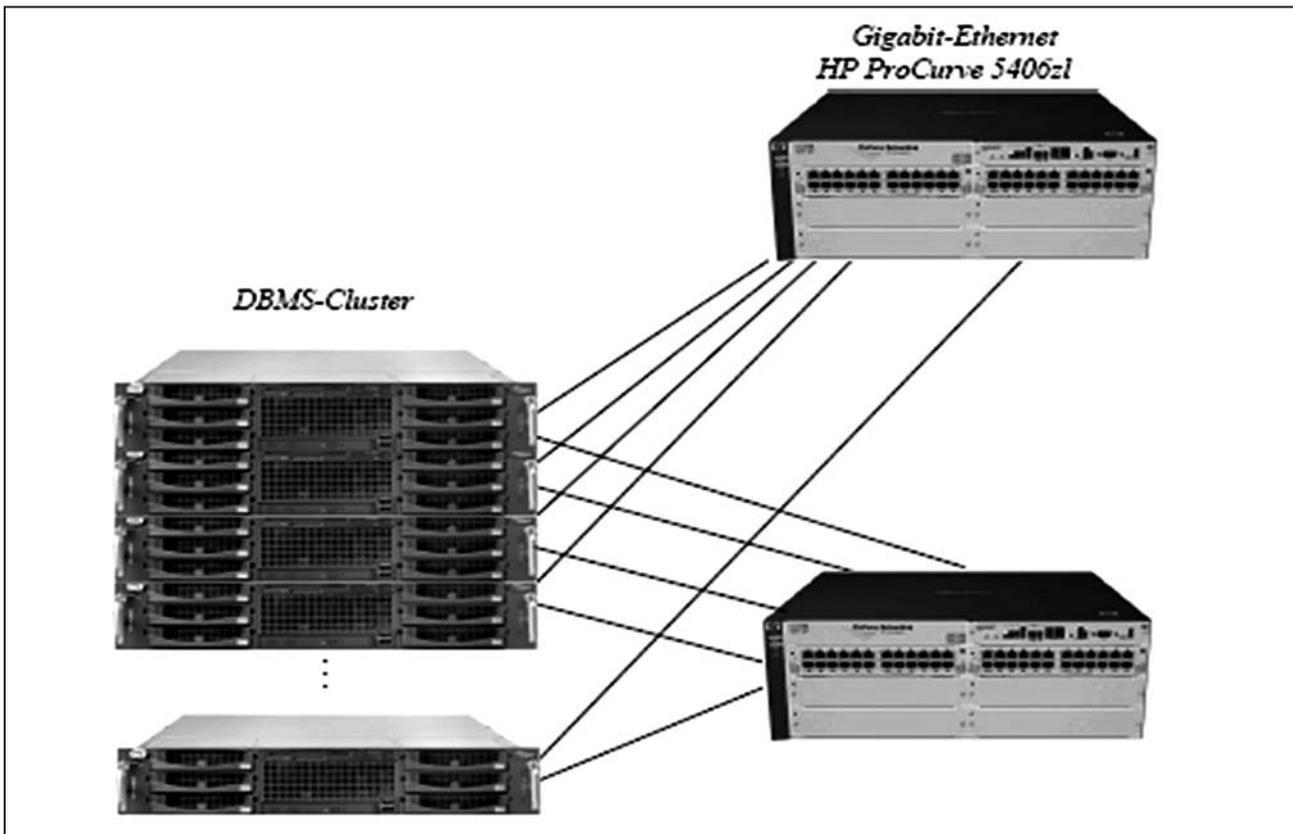


Рис. 26. Кластер Exasolution v. 2.1 для тестов TPC-H (5)

- Большая степень интеграции с ОС и использование специфичной ОС. Продукт использует свою операционную систему, называемую ExaCluster. Эта ОС использует существенно переработанное ядро Linux и другие компоненты, разрабатываемые компанией, позволяющие строить MPP-кластеры, адаптированные для работы СУБД Exasolution и обеспечивающие ей средства управления, возможности репликации, высокую отказоустойчивость и надежность.
- Отсутствие выделенного контрольного узла. Кластер Exasolution универсален с точки зрения доступа клиентов и координации запросов.

В тестировании TPC-H участвовали системы с БД объемом до 3ТВ. Все они показали хорошие результаты, оставив позади системы Oracle, IBM, MS-SQL и, с меньшим отрывом, Paraccel.

Система с БД 1ТВ, принявшая участие в тестировании, состояла из 48 узлов кластера под управлением ОС ExaCluster, каждый из которых являлся сервером Fujitsu PRIMERGY RX300 S4 с двумя четырехядерными процессорами Intel Xeon X5460 3,16 GHz и объемом ОЗУ 16GB. Схема системы показана на рис. 26. По организации

аппаратной части она не сложнее, чем система для PADB, рассмотренная в 7.1. В качестве устройств хранения БД также выступали внутренние диски серверов, попарно объединенные в зеркальные тома. Система имела полную стоимость чуть выше миллиона долларов, из которой опять же основной частью являлась стоимость лицензий на ПО Exasolution v. 2.1 (\$756 тыс.).

Сравнительные результаты тестирования СУБД Paraccel и Exasol для описанных выше конфигураций и базы объемом 1ТВ приведены ниже.

Цена и производительность кластерных СУБД

Для оценки сравнительной производительности СУБД от различных производителей, была предпринята попытка обратиться на сайт Transaction Processing Performance Council (TPC), являющийся популярным ресурсом для оценки производи-

тельности и соотношения цена/производительность. Целесообразным представлялось провести анализ результатов тестирования различных систем на нагрузках OLTP и DSS.

Из тестов OLTP, предлагаемых на сайте, подходящими являются TPC-C и TPC-E. Тест TPC-E из рассмотрения пришлось исключить, т.к. системы, подвергавшиеся этому тесту, представляли собой исключительно одиночные серверы с ОС Windows и СУБД MS SQL Server. Но и среди систем, подвергавшихся тестам TPC-C, сравнительный анализ провести не удалось, т.к. из всего множества систем кластеризованной СУБД оказалась лишь одна – hp Integrity rx5670 Cluster 64P. Это система из 16 четырехпроцессорных серверов с процессорами Intel Itanium, использовавшая СУБД Oracle RAC 10g. Тестирование этой системы проводилось в 2003г. Таким образом, более-менее актуальной информации по тестированию кластерных систем на нагрузке OLTP на сайте не представлено.

Для DSS-систем ситуация обстоит несколько лучше – на сайте предоставлены отчеты по тестам TPC-H для различных систем. В частности, кластеризованные СУБД представлены следующими производителями Oracle, IBM, EXASOL и Paracel. Далее приводятся сравнительные таблицы для 1 и 3ТВ БД.

Сравнение результатов TPC-H для Exasol и Paracel

Основные характеристики систем и результаты тестирования СУБД Exasolution и Paracel Analytic DB отражены в таб. 3. Т.к. на данном

объеме БД сколько-нибудь сравнимых по результатам и конфигурациям систем от других производителей не нашлось, в таблице сравниваются только эти две системы, работавшие с БД объемом 1ТВ.

Параметр оценки производительности системы в тесте TPC-H – QphH@Size. Величина QphH@Size (TPC-H Queries per hour на определенном объеме) является составной и характеризует:

- зависимость скорости обработки от размера БД;
- вычислительную мощность системы при обработке одного запроса;
- скорость обработки множественных конкурентных запросов.

Из таблицы видно, что система с БД Exasolution выигрывает у системы Paracel в производительности в три раза. Этот выигрыш обусловлен, в основном, такими очевидными факторами, как более быстрые процессоры и вдвое большее число процессорных ядер в системе Exasol, а также, возможно, в меньшей степени, архитектурными особенностями СУБД и оптимизацией ОС для работы с БД. Проигрыш в производительности, а также несколько более дорогое аппаратное обеспечение делают систему Paracel существенно, в 3,5 раза, проигрывающей и по показателю цена/производительность.

Сравнение результатов TPC-H для Oracle, IBM и Exasol

В этом разделе приводятся результаты для систем с БД разных производителей объемом 3ТВ.

Таб. 3. Сравнительные данные для СУБД Exasol и Paracel – БД объемом 1ТВ

	Exasol	Paracel
Спонсор тестов	Exasol	Sun
Число узлов	48	48
CPUs/Cores/Threads	96/384/384	96/192/192
CPU Model	Intel Xeon Quad-Core X5460 – 3.16 GHz	AMD Opteron – 2.8 GHz
RAM / node (GB)	16	16
ОС	EXASOL ExaCluster OS v. 2.1	RHEL 4.4
RDBMS	EXASOL ExaSolution v. 2.1	Paracel Analytic Database
QphH@1TB	1,018,321	315,842
Price/QphH	1,18	4,57
Полная цена	1,200,544 USD	1,442,050 USD
Цена аппаратной части (приб.)	320 тыс. USD	360 тыс. USD
Дата тестирования	2.06.2008	29.10.2007

Таб. 4. Сравнительные данные для СУБД Exasol, IBM, Oracle – БД 3ТВ

	Exasol	IBM	Oracle
Спонсор тестов	Exasol	Sun/IBM	HP
Число узлов	80	10	16
CPUs/Cores/Threads	160/640/640	20/40/40	64/128/128
CPU Model	Intel Xeon Quad-Core X5460 – 3.16 GHz	AMD Opteron – 2.8 GHz	AMD Opteron – 2.8 GHz
RAM / node (GB)	24	16	12
ОС	EXASOL ExaCluster OS v. 2.1	Sun Solaris 10 x86-64	RHEL 4
RDBMS	EXASOL ExaSolution v. 2.1	IBM DB2 ESE v.9.1	Oracle 10g RAC
QphH@3ТВ	1,608,920	38,672	110,576
Price/QphH	1,36	29,39	37,80
Полная цена	2,179,072 USD	1,136,536 USD	4,179,238 USD
Цена аппаратной части (приб.)	587 тыс. USD	370 тыс. USD	2519 тыс. USD
Дата тестирования	2.06.2008	10.12.2007	8.06.2006

Комментарии к таб. 4:

- Самой производительной системой оказывается Exasol Exasolution на 80 узлах Fujitsu PRIMERGY. По показателю QphH она более, чем в 10 раз превосходит своих оппонентов. Кроме прогрессивной архитектуры, рассмотренной в общих чертах в 7, она имеет и самое большое количество узлов, и самые быстрые процессоры с самым большим числом ядер. Она же имеет самый выгодный показатель цена/производительность. Следует заметить, что система использует внутренние SAS диски объема 146GB на серверах для хранения БД. Важно также учесть, что среди оппонентов этот тест является самым недавним, таким образом, система использует более дешевую, современную и производительную элементную базу по сравнению с конкурентами, тестируемыми ранее.
- Системы с Oracle RAC и DB2 собраны на серверах с одинаковыми процессорами. Однако аппаратная конфигурация кластера Oracle RAC более мощная по количеству процессоров и использует в три раза больше процессорных ядер, чем система с DB2, вследствие чего имеет в тестах в три раза больший показатель производительности.
- Система с DB2 оказывается самой дешевой из рассмотренных систем, так как имеет минимальное по сравнению с другими количество узлов и процессоров, и ей не нужны для работы дополнительные внешние дисковые устройства. Эта система, за счет использованных серверов Sun X4500 с большим коли-

чеством внутренних дисков SATA, позволяет создавать всего при 10 узлах БД размером до 200ТВ, пользуясь для хранения БД исключительно внутренними дисками.

- Конфигурация системы с Oracle RAC оказывается самой дорогой из всех рассматриваемых. Одной из главных причин этого является использование внешнего дискового массива. Использование внешних устройств хранения БД – дисковых массивов и SAN, является архитектурной особенностью систем с Oracle RAC, т.к. доступ к одним и тем же дисковым устройствам необходим со всех узлов системы.

Заключение

Объем статьи и сроки ее подготовки не позволили рассмотреть все кластеризованные СУБД, заметные в данный момент на рынке ИТ. Так, например, за кадром остались такие системы для DSS и BI, как: Green Plum (<http://www.greenplum.com>), аппаратно-программный комплекс KickFire (<http://www.kickfire.com>), использующий специальный SQL-chip для обработки запросов и работающий с СУБД MySQL, а также другие разработки.

Однако, на основе рассмотренных систем, можно сделать ряд выводов, касающихся даль-

нейшего развития кластеризованных СУБД и целесообразности их освоения:

- Основная область применения кластеризованных СУБД на настоящий момент — аналитические СУБД DSS и BI. И наиболее перспективными системами для таких задач выступают кластеры архитектуры «shared nothing» с организацией БД по принципу «column oriented DB». Дальнейшее развитие таких систем будет, очевидно, идти в сторону кластерных систем x86-64, использующих внутренние диски для хранения БД. Такие системы дешевы, довольно просты в реализации и обладают достаточной надежностью и высокой вычислительной мощностью.
- Применение кластеризованных СУБД для задач OLTP в данный момент не является широко используемой практикой. Судя по проводимым тестам, основным направлением развития систем класса OLTP по-прежнему являются SMP системы с мощной подсистемой ввода-вывода.
- Системы СУБД все больше специализируются. Так, например, очевидно, что для задач OLTP с интенсивными вставками и исправлениями строк, организация «column oriented DB» подходит хуже, чем «row oriented DB». В отношении построения универсальных СУБД в выгодном свете выглядит БД IBM DB2, т.к. является «row oriented» и позволяет строить сложные гибридные системы, сочетающие в себе кластеризацию по принципу «shared nothing» и разделы из одиночных SMP-систем. С другой стороны, возможно, станет более популярным и применение разнородных сред, таких, как, например, мощная SMP-система с Oracle для обработки OLTP-транзакций и кластеризованная система Paraccel для аналитической части БД.
- Во время написания данной статьи произошло такое важное событие, как покупка компании Sun Microsystems компанией Oracle. Вследствие этого может произойти как существенная модификация продуктов Oracle с заимствованием некоторых идей от MySQL, так и полное исчезновение MySQL как самостоятельного продукта. Также компания IBM может оказаться в менее выгодных усло-

виях продажи своих систем с Oracle, чем это было ранее, когда у Oracle не было своего аппаратного обеспечения. В таких условиях IBM DB2 может приобрести более ярко выраженный альтернативный характер по отношению к Oracle на платформе IBM.

- Все выше перечисленные выводы позволяют заключить, что кластеризованные СУБД, особенно в приложениях DSS, BI и других аналитических и отчетных системах, являются перспективным направлением и, безусловно, требуют освоения. Другим перспективным направлением является совместная работа гибридных систем с СУБД от различных производителей, а также миграция данных между СУБД различных производителей. В приложении к российскому сегменту рынка ИТ эти направления деятельности также могут быть интересны с точки зрения предложения решений, обладающих изначально лучшими, чем у конкурентов, показателями производительности, масштабируемости и цены. Однако каждый такой случай будет требовать индивидуального подхода. В этих случаях, чтобы суметь успешно разработать и внедрить решение по построению систем с СУБД, альтернативного традиционно используемым системам с Oracle, нужно обладать как минимум следующими базовыми навыками:
- Установка, конфигурация и тестирование хотя бы одного варианта системы с архитектурой «shared nothing».
- Установка, конфигурация и тестирование хотя бы одного варианта системы с организацией «column oriented DB».
- Практическое использование гибридных систем OLTP/DSS с СУБД от различных производителей.

Наиболее близким и доступным для изучения из рассмотренных продуктов на сегодняшний день является IBM DB2:

- Продукт может быть установлен на имеющемся стендовом оборудовании.
- Демо-версия продукта доступна для скачивания.

ИСПОЛЬЗОВАННЫЕ МАТЕРИАЛЫ

- 1) Руководства фирмы IBM к продукту DB2
(<http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg27009727>):
IBM DB2 Administration Guide – Planning
IBM DB2 Administration Guide – Implementation
IBM DB2 Performance Guide
IBM DB2 High Availability Guide
IBM DB2 Partitioning and Clustering Guide
- 2) Материалы по Sybase ASE 15 CE с сайта Sybase (<http://www.sybase.com>):
Sybase ASE 15 Clustered Database. Product Datasheet.
Implementing Sybase ASE 15 CE
- 3) Материалы с сайта Hypertable (<http://www.hypertable.org>):
- 4) Руководства MySQL
(<http://dev.mysql.com/doc/>):
MySQL Reference Manual
MySQL Cluster NDB Reference Guide
- 5) Материалы отчетов по тестам TPC
(<http://www.tpc.org/>)
- 6) Листовки Paracel
(<http://www.paracel.com>):
Paracel Analytic Database for Data Warehousing
- 7) Материалы с сайта компании EXASOL
(<http://www.exasol.com>)

Перечень используемых сокращений

Сокращение	Полное наименование
ЛВС	Локальная вычислительная сеть
ПО	Программное обеспечение
СХД	Сеть хранения данных, тоже что и SAN
ASM	Automatic Storage Management
BI	Business Intelligence
DSS	Decision Support Systeem
FC	Fibre Channel
LAN	Local Area Network
MPP	Massively Parallel Processing
NDB	Network Database
OLTP	Online Transaction Processing
SAN	Storage Area Network
GE	Gigabit Ethernet
IB	Infiniband
RAC	Real Application Cluster
SMP	Symmetric Multiprocessing

Jet Info

ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ

Издается с 1995 года

Главный редактор: Дмитриев В.Ю. (vlad@jet.msk.su)
Редактор: Слободчикова Т.А. (slobodchikova@jet.msk.su)
Россия, 127015, Москва, Б. Новодмитровская, 14/1
тел. (495) 411 76 01
факс (495) 411 76 02
[email: JetInfo@jet.msk.su](mailto:JetInfo@jet.msk.su) <http://www.jetinfo.ru>



Издатель: компания «Инфосистемы Джет»

Подписной индекс по каталогу Роспечати

32555

Полное или частичное воспроизведение материалов, содержащихся в настоящем издании, допускается только по согласованию с издателем