

# Конфигурирование сервера Oracle для сверхбольших баз данных

Carry V. Millsap  
*Oracle Corporation*

21 августа, 1996

## **Аннотация**

Эта статья поможет читателю настраивать сверхбольшие базы данных Oracle (Very Large Database, в дальнейшем — VLDB) для достижения высокой производительности и высокой доступности при низких издержках на эксплуатацию. Она описывает решения выбора размера блока данных Oracle, применения RAID-технологий, использования «линейных» устройств (raw-devices), конфигурирования журнальных файлов, разбиения табличных пространств на разделы, выбора параметров хранения и настройки сегментов отката. Статья описывает технологии и связанные с ними ограничения, а также технически детальные методы для оптимизации конфигурации в рамках этих ограничений.

## Содержание

<b>1 Введение</b>	
1.1 Операционная сложность . . . . .	2
1.2 Высокая производительность . . . . .	2
1.3 Высокая доступность . . . . .	3
1.4 Преодоление сложностей VLDB . . . . .	3
<b>2 Размер блока СУБД Oracle</b>	<b>3</b>
2.1 Уменьшение нагрузки на ввод/вывод . . . . .	4
2.2 Улучшение использования пространства . . . . .	4
2.3 Предотвращение узких мест при параллелизме . . . . .	4
2.4 Компромиссы . . . . .	5
<b>3 Дисковая конфигурация</b>	<b>5</b>
3.1 Структурный анализ конфигурации . . . . .	6
3.2 RAID . . . . .	8
3.3 Размер сегмента чередования . . . . .	16
3.4 Размер массива . . . . .	20
3.5 Линейные устройства . . . . .	22
3.6 Конфигурация Oracle . . . . .	23
<b>4 Журнальные файлы</b>	<b>24</b>
4.1 Баланс производительности и доступности . . . . .	25
4.2 Контрольные точки сервера Oracle . . . . .	25
4.3 Восстановление инстанции сервера Oracle . . . . .	26
4.4 Размещение журнальных файлов . . . . .	27
4.5 Размер журнального файла . . . . .	28
4.6 Число журнальных файлов . . . . .	29
<b>5 Планирование табличных пространств</b>	<b>30</b>
5.1 Назначение сегментов табличным пространствам . . . . .	32
<b>6 Параметры хранения</b>	<b>32</b>
6.1 Maxextents . . . . .	33
6.2 Свойства параметров хранения . . . . .	33
6.3 Выбор параметров хранения . . . . .	34
<b>7 Сегменты отката</b>	<b>35</b>
7.1 Размер сегмента отката . . . . .	36
7.2 Количество сегментов отката . . . . .	37
<b>8 Заключение</b>	<b>38</b>

## 1 Введение

VLDB является аббревиатурой, принятой для обозначения СверхБольших Баз Данных (Very Large Database). Словосочетание «сверхбольшая» имеет разное наполнение для разных людей, но оно всегда связано с ощущением трудности, сложности, высокой стоимости и риска. VLDB являются тем, что люди связывают с огромными базами данных, поддержка которых граничит с искусством. Требуются большая изобретательность, умение планировать, упорный труд и значительные капиталовложения для того, чтобы избежать очень серьезных разочарований при попытке внедрения VLDB.

### 1.1 Операционная сложность

Операционная сложность является очевидной проблемой VLDB. Как оператор VLDB Вы должны постоянно оценивать множество сильно связанных параметров. Система будет «наказывать» попытки применить радикальные или случайные изменения. Огромные объекты и длительные процессы оставляют Вам мало пространства для маневра при устранении проблемы. Это требует, в первую очередь, осторожности при планировании устранения проблемы. Состояние сотен или даже тысяч людей зависит от Вашей способности быстро принимать правильные решения. Здесь требуется талант и упорный труд.

Длительность многих важнейших процедур, например резервное копирование и восстановление, пропорциональна размеру объектов, которыми они манипулируют. Поэтому в больших БД эти важнейшие процедуры могут требовать очень больших временных интервалов. С ростом БД стоимость ошибок увеличивается. Операции — подобные реконфигурации диска, перестроению объекта БД, подсчету числа строк в таблице, откату транзакции, — могут выглядеть вполне безобидно в небольших БД, но быть полностью неприемлемыми для VLDB. Если действие требует нескольких часов Вашего рабочего времени, Вы должны избегать его выполнения.

### 1.2 Высокая производительность

Увеличивают сложность и другие факторы. Рассмотрим доступ к данным. Вы можете до-

пустить использование неэффективного запроса, который требует 2 секунды работы процессора на высокопроизводительной современной системе с 20 пользователями, но Вы не сможете допустить такое же время обслуживания в системе с 1,200 пользователями, где идет непрерывная борьба за фиксаторы (latches), диски и процессорное время. Запросы к VLDB должны быть оптимизированы исключительно хорошо, иначе система развалится.

Более того, многие VLDB становятся «VL» в первую очередь вследствие большого числа одновременно работающих пользователей и пакетных заданий, создающих большой объем транзакций. Большой объем транзакций создает стрессовые ситуации в подсистеме ввода/вывода связанные с генерацией redo-информации и записью в файлы данных. Высокий уровень параллелизма процессов перегружает фиксаторы и механизмы блокировок, разработанные для перевода доступа к критическим ресурсам в последовательный режим (serialize).

### 1.3 Высокая доступность

Чем больше обдумываешь проблему — тем она сложнее выглядит. VLDB часто являются источником данных для важных приложений с высокими требованиями доступности. Разработчики промышленных СУБД слышат формулу «семь-дней-по-двадцать-четыре-часа», а мы цепене-ем от монументальной сложности этой задачи. Ввиду электрических и сетевых неисправностей, некачественных модулей памяти и дисковых контроллеров, ошибок приложений и операционных систем, модернизации ПО и оборудования и просто случайных ошибок оператора, — достичь нескольких секунд или минут останова в год крайне сложно. Как обнаружить и исправить логические разрушения сотен и даже тысяч гигабайт данных, которые образовались вследствие ошибочного ввода оператора?

### 1.4 Преодоление сложностей VLDB

Путь к преодолению сложностей, присущих VLDB, лежит в упрощении БД настолько, насколько это возможно. Вы оптимизируете как СУБД, так и само приложение для снижения

любых видов нагрузки. Вы выбираете структуры данных, которые минимизируют ввод/вывод при доступе к данным. Вы создаете приложения с максимально простыми транзакциями. Вы изолируете влияние вышедших из строя компонент на минимально возможные подмножества данных. Вы делаете единицы резервного копирования и восстановления минимально возможными.

Ниже перечислены некоторые аспекты, позволяющие построить хорошую VLDB:

- Оптимизация выполнения последовательности бизнес-операций
- Оптимизация логической модели данных
- Оптимизация схемы БД
- Построение надежной, высокопроизводительной конфигурации аппаратного обеспечения
- *Оптимизация физической конфигурации БД*
- Оптимизация SQL-операторов приложений
- Оптимизация операционной системы и инстанции сервера Oracle
- Создание корректных и надежных процедур сопровождения

В этой статье мы исследуем решения для физической конфигурации БД, необходимые для успешного построения VLDB.

## 2 Размер блока СУБД Oracle

Ирония жизни — в тот день, когда Ваш опыт в управлении сервером Oracle минимален, Вам необходимо ввести значение для параметра **db\_block\_size**, которое будет в дальнейшем неизменно на протяжении всей жизни БД. Важно то, что значение, которое Вы выбрали для размера блока данных Oracle, оказывает сильнейшее влияние на производительность системы. Последующие разделы обратят Ваше внимание на некоторые компромиссы, которые необходимо рассмотреть перед выбором размера блока данных Oracle для Вашей базы данных.

Оптимальный размер блока данных Oracle для VLDB лежит в пределах от 4KB до 64KB. Наиболее часто используемым является значение 8KB

и, на втором месте, — 16КВ. Сервер СУБД Oracle, установленный на системе с очень быстрой реализацией копирования больших объемов памяти может эффективно работать с размером блока в 32КВ и даже (как минимум, теоретически) с 64КВ. Я не знаком с обстоятельствами, которые могли бы требовать выбора размера блока для VLDB в 2КВ и лишь в очень специальных случаях размер в 4КВ будет подходящим для VLDB. В [6, Millsap (1996)] я давал развернутое техническое описание этого вопроса, здесь мы ограничимся только выводами <sup>1</sup>.

## 2.1 Уменьшение нагрузки на ввод/вывод

Наиболее важной характеристикой самых сложных реализаций VLDB является огромная нагрузка на ввод/вывод. Владельцы лучших мировых VLDB вкладывают большие деньги в средства, способные уменьшить нагрузку на ввод/вывод в их системах. Использование большого размера блока данных СУБД Oracle является простой техникой для уменьшения некоторых видов нагрузки на ввод/вывод.

- **Число операций ввода/вывода** — аппаратура имеет вполне определенную цену одной операции ввода/вывода. Манипуляция с небольшим числом больших блоков данных будет иметь преимущества над манипуляциями большим числом блоков меньшего размера.
- **Производительность индексов** — скорость работы с индексами обратно пропорциональна высоте двоичного дерева (B\*-tree) индекса. Большой размер блока Oracle позволит хранить больше ключей в одном блоке что, в свою очередь, даст лучшую производительность при поиске.
- **Снижение сцеплений** — сервер Oracle использует сцепление (*chaining*) для хранения строк таблиц, кластеров данных и хэш-таблиц размер которых (строк) превышает размер блока данных. Сцепление частей

<sup>1</sup>Чтобы не заставлять Вас искать указанный документ компании Oracle, что к стати является не простым делом, скажу, что одним из специальных случаев, когда выбор в 4КВ будет подходящим, заключается в наличии огромного числа небольших сегментов (меньше 100КВ).

строк между несколькими блоками данных приводит к увеличению числа операций ввода/вывода на одну обрабатываемую строку. Большой размер блока данных снижает вероятность того, что будет требоваться сцепления для хранения строки, и тем самым, уменьшает число операций ввода/вывода в системе.

## 2.2 Улучшение использования пространства

Сервер Oracle хранит небольшой объем служебной информации фиксированного размера в каждом блоке БД вне зависимости от того, какой объем пользовательских данных хранится в этом блоке. Таким образом, чем меньше блоков находится в БД, тем меньше пространства будет истрачено на накладные эти расходы. Для очень малых сегментов выигрыш от экономии на фиксированных накладных расходах может обернуться потерей пространства за счет неиспользуемых завершающих блоков сегмента, но эти потери, в свою очередь, незначительны по сравнению с потерями вызванными неточным заданием параметров хранения <sup>2</sup>. Использование большого размера блока Oracle для очень больших сегментов сохраняет примерно от двух до шести процентов общего объема базы данных. Этот выигрыш ведет к повышению производительности и снижению стоимости эксплуатации, что особенно заметно на снижении времени резервного копирования/восстановления.

## 2.3 Предотвращение узких мест при параллелизме

Использование большого размера блока данных Oracle увеличивает риск возникновения узких мест при совместном доступе за исключением случаев, когда архитектор базы данных понимает как использовать параметры хранения **initrans** и **maxtrans**. Значение **initrans** должно быть установлено в максимально ожидаемое число транзакций, одновременно манипулирующих блоком

<sup>2</sup> Как мы обсудим позднее, приближенное задание параметров хранения является хорошим компромиссным решением, снижающим стоимость эксплуатации, в сравнении с VLDB с очень точно заданными параметрами хранения.

данных<sup>3</sup>. Таким образом, если Вы увеличили размер блока данных Oracle в  $k$  раз, то Вам необходимо увеличить значение параметра **initrans** также в  $k$  раз. Для того чтобы позволить динамически расти «таблице списка входов транзакций», Вам необходимо установить значение параметра **maxtrans** в значение, превышающее значение **initrans**<sup>4</sup>.

Неудачный выбор параметров **initrans** и **maxtrans** для сегментов базы данных может быть причиной возникновения ожиданий пользовательских сессий. Сессии, которые изменяют блок, имеющий эту проблему, будут мешать другим сессиям получить входы в структуру данных блока, которая позволяет делать реконструкцию данных для достижения непротиворечивости чтения. Эта ситуация, имеющая название *ITL-конкуренция*, может быть обнаружена администраторами БД с помощью представления **v\$lock**. Наблюдения будут показывать сессии, ожидающие на блокировке **TX** (очередь транзакций) в режиме блокировки **4** (разделяемый доступ).

Вы в состоянии полностью избежать этих проблем, используя соответствующие техники, описанные в стандартной документации компании Oracle по настройке параметров **initrans** и **maxtrans** для сегментов БД.

## 2.4 Компромиссы

Хотя большой размер блока данных Oracle имеет преимущества для VLDB, имеются определенные ограничения на максимальный размер блока данных который Вы можете выбрать.

- *Физический размер ввода/вывода* — Размер блока данных СУБД Oracle не должен превышать максимально возможный размер физической операции чтения данных. Убедитесь также, что размер пакетного чтения для операции полного сканирования таблицы ограничен максимальным размером физического чтения операционной системы. К примеру, если размер блока данных Вашей

<sup>3</sup> Под транзакцией здесь понимается выполнение операторов **insert**, **update**, **delete**, или **select ... for update**

<sup>4</sup> Формально, «таблица списка входов транзакций» называется *списком интересующихся транзакций* (*interested transactions list*), или *ITL*.

СУБД равен 32KB и параметр **db\_file\_multiblock\_read\_count** равен 32 то маловероятно, что Вы сможете выбрать 1MB = 32KB × 32KB за одну физическую операцию чтения.

- *Redo-размер* — Процесс, пишущий в журнальные файлы Oracle (LGWR) записывает целые блоки при изменениях в файлах данных табличных пространств, переведенных в состояние резервного копирования. Поэтому решение уменьшить объем генерируемой redo-информации в течение «горячего» резервного копирования, может мотивировать Вас ограничить размер блока данных Oracle.
- *Размер копируемой области памяти* — В случае, если Вы выберете размер блока данных, превышающий максимальный размер памяти, который операционная система в состоянии обработать за одну операцию, Вы можете наблюдать увеличение загрузки процессора и снижение производительности, связанные с обработкой больших блоков в SGA.
- *Ограничения параллелизма* — Максимальные значения для параметров **initrans** и **maxtrans** равно 255. Если блок данных СУБД Oracle при обычных нагрузках получает более 255 запросов от транзакций на изменение, то это значит, что размер блока данных выбран слишком большим. Шансы возникновения такой ситуации близки к нулю: если действительно 255 транзакций одновременно модифицируют один блок, то результирующая конкуренция за фиксаторы будет настолько серьезной, что Вам непременно придется увеличить параметр **pctfree** для такого сегмента.

## 3 Дисковая конфигурация

Сердцем хорошей СУБД является надежная и производительная подсистема ввода/вывода. Поэтому проектирование подсистемы ввода/вывода — важная тема при создании любого приложения с использованием VLDB.

### 3.1 Структурный анализ конфигурации

На самом высоком уровне абстракции определение требований к системе является простым делом, поскольку все мы желаем трех вещей: *производительности, надежности и низкой стоимости*. Однако когда мы преобразуем эти общезначимые цели во что-то конкретное, мы сталкиваемся с решениями, основанными на компромиссах. Для понимания возникших конфликтов, Вы должны сделать две вещи: (1) понять Ваши цели и (2) разобраться в Ваших технологиях. Весь фокус заключается в поиске верной *микстуры* из скорости, надежности и стоимости, которая удовлетворяет специфические нужды Вашего бизнеса. Это означает что решение, верное для, Вас не всегда будет подходящим для Вашего соседа.

Архитектурные ограничения приходят к нам из двух источников: (1) это экономические ограничения и (2) это технологические ограничения. Экономические ограничения Вам должны быть ясны очень хорошо, и я уверен, Вы не желаете иметь с ними дело. Однако, даже если вообразить, что Вы можете приобрести любое оборудование и программное обеспечение, какое только пожелаете, Вы все равно столкнетесь с ограничениями технологическими.

Предположим, к примеру, что Вы имеете OLTP-приложение и Вы оптимизировали дисковую конфигурацию Вашего сервера Oracle для достижения максимальной производительности при случайном вводе/выводе с ущербом производительности последовательного доступа к данным. Такое решение может выглядеть вполне разумным, поскольку доступ к файлам OLTP-базы данных в подавляющем большинстве основан на операциях чтения с использованием хорошо определенных индексов или хэш-структур, а также произвольных (scattered) операциях записи.

Однако первый же опыт реорганизации индекса на дисковом массиве, оптимизированном исключительно для доступа в условиях высокого параллелизма, потребует на 300–500 процентов больше времени, чем это могло бы занять в другой дисковой конфигурации. Ваше бескомпромиссное решение об оптимизации OLTP-производительности напрямую снизит доступность Вашей системы вследствие увеличения продолжительности недоступности индекса во

время его реорганизации.

Этот пример, как сотни подобных, указывают нам на важность и трудоемкость понимания Ваших требований и постижения Ваших технологий. Наиболее важным Вашим инструментом должен стать метод структурного анализа, который поможет Вам найти правильные ответы на Ваши вопросы. В этой статье мы будем структурировать анализ конструкции подсистемы ввода/вывода с помощью разбиения на *производительность, доступность и стоимость*:

- *Производительность*

- Скорость случайного чтения
- Скорость случайной записи
- Скорость последовательного чтения
- Скорость последовательной записи
- Влияние параллелизма

- *Доступность*

- Частота отказов
- Продолжительность простоя
- Снижение производительности при отказах

- *Стоимость*

- Стоимость приобретения
- Стоимость обслуживания

В статье мы оценим несколько инструментов и техник в контексте этих категорий. Их описание дано в последующих разделах.

#### 3.1.1 Производительность

- *Производительность случайного чтения* — примером могут служить запрос с использованием индексов или хэш-структур, чтение из сегментов отката. По-видимому, эта категория составляет основную массу вызовов операций чтения в высокоактивной OLTP-системе. Для хранилищ данных, эта категория представляет лишь незначительную часть от общей нагрузки.

- *Производительность случайной записи* — примером могут служить все операции записи данных, индексов и сегментов отката процессом DBWR. Эта категория составляет существенную часть общей массы операций записи в OLTP-системах в период основной работы. Для хранилищ данных данная категория составляет незначительную или нерегулярную долю нагрузки.
- *Производительность последовательного чтения* — примерами могут служить резервное копирование, полное сканирование таблиц, создание индексов, параллельное выполнение запросов, чтение из временных сегментов, операции восстановления и доката из журнальных файлов. Даже для хорошо оптимизированных OLTP-приложений, эта категория должна рассматриваться как составная часть нагрузки, особенно для систем с интенсивными пакетными заданиями. Для хранилищ данных последовательное чтение может быть практически единственным видом чтения.
- *Производительность последовательной записи* — примером служит работа LGWR-процесса, запись во временные сегменты, прямые загрузки данных (direct-load), создание индексов, создание табличных пространств и операция восстановления. Проектировщики, акцентируясь на транзакциях и обработке запросов иногда забывают эту категорию. Хотя во время первичной загрузки данных и в течении всей эксплуатации база данных любого типа генерирует большое число операций последовательной записи.
- *Влияние параллелизма* — в каждой из категорий, упомянутых выше, архитектор должен рассмотреть влияние различных уровней параллелизма.

### 3.1.2 Доступность

- *Частота отказов* — ожидаемое число случаев возникновения отказов в единицу времени. Частота отказов определяется с помощью MTTF (mean time to failure — среднее время возникновения отказа). Для примера, у

жесткого диска с MTTF равным 20,000 часов можно ожидать отказ каждые 23 года.

- *Продолжительность простоя* — стоимость простоя может быть оценена в деньгах, количество которых определяются как некоторая функция от продолжительности простоя. Стоимость простоя для некоторого события пропорциональна показателю среднего времени восстановления (MTTR, mean time to repair) для данного события — чем дольше продолжительность простоя, тем выше стоимость.
- *Снижение производительности в течение отказа* — различным дисковым конфигурациям присущи различные показатели производительности при возникновении отказа. Некоторые конфигурации не являются отказоустойчивыми вообще, в других при возникновении отказов снижается производительность, третьи вовсе нечувствительны к возникновению различных типов отказов. Важно сопоставить требования приложения с ожидаемым падением производительности при возникновении отказа.

### 3.1.3 Стоимость

- *Стоимость приобретения* — экономическая оценка закупки системы.
- *Стоимость обслуживания* — стоимость эксплуатации и поддержания системы. Стоимость обслуживания состоит из эксплуатационных издержек, которые зависят от надежности системы и из административных затрат, зависящих от сложности системы. Например, система сложная в обращении, требующая более подготовленных операторов и техников по ремонту, будет иметь более высокую стоимость обслуживания.

### 3.1.4 Метод анализа

Декомпозиция на производительность, надежность и стоимость дает нам важный инструмент для построения хорошей системы. Он позволяет нам оценить компромиссы как в пределах одной категории (например — скорость выполнения случайного чтения за счет последовательного

чтения), так и понять влияние одной категории на другую (например — сильную связь между скоростью выполнения последовательной записи и продолжительностью простоя).

Эти категории также позволяют нам формализовать наши рассуждения о системе. К примеру, понимание разных составляющих понятия надежности дает нам возможность изучать их независимо: (1) мы можем фокусировать усилия на снижении частоты отказов или (2) мы можем попытаться снизить продолжительность простоев вследствие отказов, которые мы предотвратить не в силах. И это напоминает нам еще раз, что продолжительность простоя является частично вопросом надежности, а частично — вопросом производительности.

Последующие разделы помогут Вам провести высокоуровневый анализ взаимосвязей производительности, доступности и стоимости при проектировании подсистемы ввода/вывода для СУБД Oracle. Этот раздел поможет Вам лучше понять Ваши задачи и технологии, поможет сделать Вам более информированные решения, чтобы Вы получили наилучший результат от Ваших инвестиций.

### 3.2 RAID

На втором месте по ненадежности, после человека, в компьютерных системах, является жесткий диск. Для увеличения надежности дисков большинство поставщиков решений сегодня предлагают дисковые массивы, называемые RAID — избыточные массивы недорогих дисков (redundant arrays of inexpensive disks). RAID-массивы представляют собой высокопроизводительные, устойчивые к отказам подсистемами ввода/вывода, которые используют менее дорогие технологии жестких дисков чем те, что используются в традиционных высоконадежных больших ЭВМ.

Понятие RAID было введено в 1987 году в статье опубликованной в Калифорнийском Университете [9, Patterson et al.(1988)]. Номера уровней RAID означают разную организацию простых технологий дисков для достижения производительности и надежности при относительно невысокой стоимости.

Наиболее важными уровнями RAID, которые должен понимать архитектор VLDB Oracle явля-

ются 0, 0+1, 3 и 5. Рисунок 1 дает концептуальное представление об этих RAID-конфигурациях. Обратите внимание, что поставщики RAID могут выбрать для реализации функций чередования и зеркалирования либо программное, либо аппаратное решение. Этот выбор влияет на количество и типы контроллеров необходимых для реализации RAID-массива.

Производительность, предлагаемая RAID-конфигурациями, впечатляет. За счет равномерного распределения физической нагрузки ввода/вывода по всем дискам, RAID-массивы с чередованием показывают бесподобное время ответа и пропускную способность. Массив с чередованием из пяти дисков может показать почти в пять раз большую производительность при последовательном вводе/выводе по сравнению с независимой конфигурацией дисков того же размера.

Равным образом впечатляет превосходство RAID-массивов в надежности. Дисковая система, состоящая из сотни дисков с MTTF равной 200.000 часов, сконфигурированная без избыточности имеет среднее время до потери данных (MTTDL) меньше чем 83 дня. Та же сотня дисков, организованная в RAID-массив с наличием избыточности имеют MTTDL порядка 26 лет [Chen et al.(1992), 161-163]. Среднее время восстановления RAID-конфигураций также превосходно. Вы можете изъять диск из активного RAID-массива 5 уровня и система, тем не менее, будет продолжать работать.

Однако, каждая RAID-конфигурация имеет свою, уникальную, стоимость. Вы можете обнаружить, что система, которая действительно удовлетворяет Вашим нуждам слишком дорога, а конфигурация, которую Вы в состоянии себе позволить, несет с собой много компромиссных решений. Архитекторы систем на базе СУБД Oracle должны двигаться в извилистом лабиринте компромиссов, принимая решение о способе применения RAID-массивов.

Несколько авторов сделали блестящую работу, описывая RAID-конфигурации и оценивая RAID-структуры с точки зрения надежности, производительности и стоимости ([1, Chen et al.(1994)]; [2, Gui (1993)]; [11, Sun (1995)]; и многие другие). Последующие разделы содержат итоги этих идей с конкретными рекомендациями о том когда и как использовать каждый тип RAID-массива

для приложений СУБД Oracle.

### 3.2.1 Определения

Учебная литература о RAID может иногда ввести в заблуждение, поскольку авторы из конъюнктурных соображений могут интерпретировать определения так как им необходимо. Для того, чтобы сделать эту статью максимально простой, определим предварительно некоторые термины.

- **Массив** — RAID-конфигурация 0, 0+1, 3 и 5 уровня, в которой диски группируются в наборы, называемые *группами с коррекцией ошибок* или *дисковыми массивами*. Мы будем называть группу дисков собранных в группу с коррекцией ошибок *массивом*.
- **Сегмент чередования** — чередование (*striping*), это программная или аппаратная возможность, при которой логически непрерывные данные записываются порциями, распределенными между дисками в массиве<sup>5</sup>. Эти порции называются *сегментами*.

Читая этот документ, важно помнить что, *массив* является коллекцией дисков, и что *чередование* заключается в распределении порций данных в пределах массива. Рисунок ниже показывает один дисковый массив, состоящий из пяти дисков. Каждый диск содержит пять сегментов чередования, общее число которых составляет 25.



### 3.2.2 Чередование без избыточности (RAID уровня 0)

RAID 0 является дисковой конфигурацией без избыточности с чередованием. Чередование, сконфигурированное соответствующим образом,

<sup>5</sup> Обратите внимание — *striping* пишется с одной буквой *p*.

дает исключительно хорошее время ответа при случайном доступе с высоким уровнем параллелизма, и превосходную пропускную способность при последовательном доступе с низким уровнем параллелизма. Выбор ширины сегмента чередования массива требует внимательного рассмотрения различных ограничений. Мы обсудим детали оптимизации размера сегмента чередования позже.

Дисковые конфигурации без избыточности полезны в тех случаях, где ограничения на стоимость приобретения перевешивают требования к надежности системы.

- **Производительность при случайном чтении** — отличная при любых уровнях параллелизма, если каждый запрос на чтение попадает в один сегмент чередования. Использование слишком малого размера сегмента чередования может привести к резкому снижению производительности в средах с высоким уровнем параллелизма.
- **Производительность при случайной записи** — та же, что и для случайного чтения.
- **Производительность при последовательном чтении** — отличная при малом размере сегмента чередования в средах с низким уровнем параллелизма. Также отличная в средах с высоким уровнем параллелизма при условии, что каждый запрос на чтение попадает в один сегмент чередования. Использование слишком малого размера сегмента чередования может привести к резкому снижению производительности в средах с высоким уровнем параллелизма.
- **Производительность при последовательной записи** — та же, что и для последовательного чтения.
- **Частота отказов** — неудовлетворительна. Отказ любого диска массива будет причиной отказа приложений, требующего процедуры восстановления носителя Oracle для каждого приложения, имеющего данные на этом массиве.
- **Длительность простоя** — неудовлетворительна. Продолжительность простоя RAID 0 состоит из времени, необходимого на обнаружение неисправности, замены диска и вы-

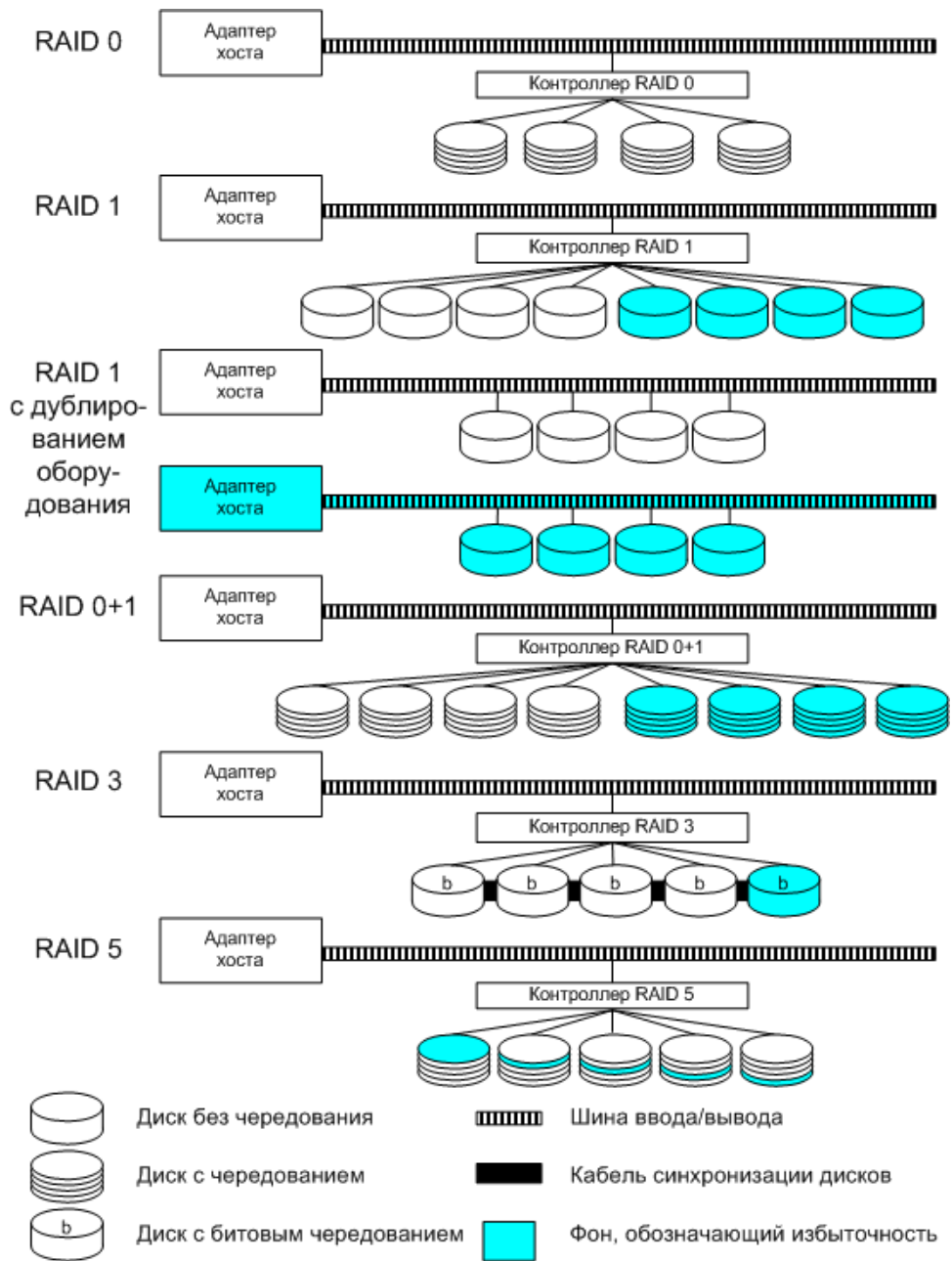


Рис. 1. Обзор RAID-архитектур

полнении процедуры восстановления носителя сервера Oracle.

- *Снижение производительности в течение отказа* — неудовлетворительно. Любой диск, вышедший из строя, влечет за собой останов приложений, данные которых располагались на дисковом массиве, до полного завершения процедуры восстановления носителя Oracle.
- *Стоимость приобретения* — отличная. RAID 0 является наименее дорогой RAID-конфигурацией.
- *Стоимость обслуживания* — от неудовлетворительной до плохой. Цена обслуживания, обусловленная частотой выполнения процедуры восстановления носителя, может превысить ценовые преимущества на приобретение перед избыточными конфигурациями. Необходимость увеличения емкости потребует либо закупки дополнительных массивов, либо переконфигурирования существующих массивов. Требуется обучение при конфигурировании дисковых массивов с чередованием для достижения оптимальной производительности.

### 3.2.3 Зеркалирование (RAID уровня 1)

Зеркалирование является фундаментальным инструментом архитектора VLDB для снижения частоты отказов дисков. Диски с поддержкой зеркальных копий — это системы, в которых идентичные копии Ваших данных записываются на два или более дисков при каждой операции записи, что позволяет Вашим приложениям работать до тех пор, пока цела хотя бы одна из копий. Несмотря на то, что наблюдается определенная потеря производительности при выполнении операции записи в массивах с зеркалированием по сравнению с конфигурациями без избыточности, зеркалирование обеспечивает наилучшую производительность операции записи среди дисковых конфигураций, устойчивых к сбоям. Зеркалирование особенно полезно для хранения файлов данных Oracle с высокой интенсивностью записи. Многие архитекторы используют зеркалирование для защиты оперативных и архивных журнальных файлов даже тогда, когда стоимостные

ограничения не позволяют использования зеркалирования всей дисковой подсистемы.

Дублирование адаптеров, шин и источников питания делает зеркалирование еще более нечувствительным к отказам оборудования. В продублированной конфигурации с зеркалированием,  $n$ -уровневая избыточность использует  $n$  одинаковых адаптеров. На сегодняшний день несколько клиентов компании Oracle используют трехуровневое дублирование для достижения гибкости операции восстановления. Тройное зеркалирование обеспечивает устойчивость к отказам подобно двойному зеркалированию и позволяет администратору БД восстанавливать СУБД на любое время в прошлом без использования стримеров. Позже мы обсудим одну технику тройного дублирования.

- *Производительность при случайном чтении* — хорошая. Если реализация контроллера RAID 1 оптимизирует операцию чтения, то несколько выше, чем у независимого диска, в противном случае — идентична независимому диску. Контроллер RAID 1 с оптимизатором будет обслуживать запрос на чтение, используя ту копию, для которой стоимость выполнения операции чтения минимальна, оставляя другой диск(и) в зеркальной группе для параллельного обслуживания других запросов.
- *Производительность при случайной записи* — хорошая. Если реализация контроллера RAID 1 использует оптимизацию чтения, то несколько хуже, чем у независимого диска, в противном случае — идентична независимому диску. Хотя запись нескольких копий может выполняться параллельно, скорость всей операции записи ограничена скоростью самого медленного устройства. Цена записи  $n$ -уровневой реализации RAID 1 равна  $\max(w_1, w_2, \dots, w_n)$ , где  $w_i$  — цена записи на  $i$ -тое устройство. Оптимизация чтения данных рассинхронизирует состояния зеркалированных дисков, вследствие чего  $w_i \neq w_j$  для  $i \neq j$  даже для идентичных дисков.
- *Производительность при последовательном чтении* — удовлетворительная. См. производительность при случайном чтении. Про-

пуская способность ограничена производительностью одного диска.

- *Производительность при последовательной записи* — удовлетворительная. См. производительность при случайной записи. Пропускная способность ограничена производительностью одного диска.
- *Частота отказов* — отличная.  $N$ -уровневое дублирование дисковой системы может обеспечить работоспособность системы при выходе из строя до  $n - 1$  дисков в зеркальном наборе. Однако RAID 1 не устраняет проблемы выхода из строя адаптера, шины ввода/вывода, RAID-контроллера, источника питания, аппаратных или программных ошибок. Для устранения этих видов ошибок необходимо дублирование оборудования. При  $n$ -кратном дублировании всех компонент система останется работоспособной при выходе из строя до  $n - 1$  адаптеров, шин ввода/вывода и пр. в пределах одного комплекта дублирования. Дублирование является наиболее отказоустойчивой дисковой конфигурацией.
- *Длительность простоя* — отличная. Продолжительность отказа вызванного  $n - 1$  или меньшим числом дисков или других продублированных компонент равно периоду времени, необходимому для замены этих компонент. В течение этого времени все приложения находятся в рабочем состоянии. Продолжительность простоя при потере данных, обусловленной выходом из строя всех  $n$  дисков сопоставима с показателем для RAID 0 конфигурации. В этом случае все зависимые приложения будут недоступны.
- *Снижение производительности в течение отказа* — отличное. При выходе из строя диска в RAID 1 не будет наблюдаться снижения производительности всего массива. Однако, на время замены диска, операция включения нового диска в массив будет генерировать большой объем ввода/вывода в восстанавливаемой зеркальной группе, что может привести к существенному снижению производительности приложений, читающих или записывающих данные на эту зеркальную груп-

пу. Замена других продублированных аппаратных компонент не влечет за собой снижения производительности.

- *Стоимость приобретения* — плохая. Стоимость емкости при  $n$ -уровневом зеркалировании в  $n$  раз выше, чем стоимость аналогичной емкости для RAID 0. Реализация RAID 1 требует специального контроллера RAID 1 в дополнение к тому же числу адаптеров, что и для RAID 0. Полное зеркалирование ограничено возможностями подсистемы ввода/вывода, поскольку  $n$ -уровневое зеркалирование требует в  $n$  раз больше посадочных мест, чем конфигурации без избыточности.
- *Стоимость обслуживания* — удовлетворительная. Стоимость включает обучение персонала администрированию систем с использованием зеркалирования, разработку программного обеспечения для интеграции процедур зеркалирования в плановые операции обслуживания и т.д.

### 3.2.4 Чередование и зеркалирование (RAID уровня 0+1)

Чередование и зеркалирование могут быть объединены в то, что многие люди сегодня называют «RAID 0+1»<sup>6</sup>. RAID 0+1 объединяет все преимущества и недостатки присущие RAID 0 и RAID 1: отличную производительность, отличную устойчивость при отказах дисков и высокую стоимость приобретения. Дублирование дисковых массивов с чередованием является основой для построения высокозагруженных OLTP-приложений, решающих критически важные задачи.

- *Производительность при случайном чтении* — отличная при всех уровнях параллелизма при условии, что каждый запрос на чтение использует один сегмент чередования. Использование слишком малого размера сегмента чередования может привести к резкому ухудшению производительности при высоком уровне параллелизма. Несколько выше, чем в RAID 0, если используется оптимизация чтения RAID 1.

<sup>6</sup>Обратите внимание, что изобретатели термина RAID включили в понятие RAID 1 возможности «RAID 0+1» [1, Chen et al.(1994), 153]. Поставщики оборудования обеспечили термину «RAID 0+1» насыщенную и счастливую жизнь.

- *Производительность при случайной записи* — отличная даже при очень высоком уровне параллелизма. Несколько хуже, чем в RAID 0, но много лучше, чем в RAID 5. См. производительность случайной записи в RAID 1.
- *Производительность при последовательном чтении* — отличная при всех уровнях параллелизма при условии, что каждый запрос на чтение использует один сегмент чередования. Использование слишком малого размера сегментов чередования может привести к резкому ухудшению производительности при высоком уровне конкуренции. Несколько выше, чем в RAID 0, если используется оптимизация чтения RAID 1.
- *Производительность при последовательной записи* — отличная. Несколько хуже, чем для RAID 0, но лучше, чем для RAID 5. См. производительность последовательной записи в RAID 1.
- *Частота отказов* — отличная. Та же, что и для RAID 1.
- *Длительность простоя* — отличная. Та же, что и для RAID 1.
- *Снижение производительности в течение отката* — отличное. То же, что и для RAID 1.
- *Стоимость приобретения* — плохая. Та же, что и у RAID 1 плюс возможны дополнительные затраты на программное или аппаратное обеспечение для поддержки чередования.
- *Стоимость обслуживания* — удовлетворительная. Оптимизация конфигурации с чередования требует обучения, также как и интеграция процедур зеркалирования в процедуры обслуживания. Увеличение емкости дисковой подсистемы потребует либо приобретения дополнительных массивов, либо реконфигурации существующих.

### 3.2.5 Побитовое чередование (RAID уровня 3)

RAID 3 является ответом высокой стоимости удвоенной избыточности RAID 1. В конфигура-

ции RAID 3 диски организованы в массив, в котором один диск выделен для хранения контрольных данных, расположенных на других дисках. В RAID 3 размер сегмента чередования равен 1 биту. Конфигурация имеет свойство, которое позволяет реконструировать данные любого диска с помощью данных, расположенных на других дисках, используя операцию исключающего ИЛИ (XOR).

Основное достоинство RAID 3 заключается в много более низкой стоимости, чем у RAID 1. Однако низкая производительность выполнения случайных операций ввода/вывода и низкая производительность при частичном отказе массива делают RAID 3 практически непригодным для большинства серверных приложений Oracle. RAID 3 является лучшим решением для приложений Oracle у которых экономические ограничения перевешивают требования надежности, с плохо оптимизируемыми операциями полного сканирования таблиц, без операций модификаций БД.

- *Производительность при случайном чтении* — плохая. Синхронизация дисков препятствует параллельному выполнению малых случайных запросов чтения. Плохая производительность при высоком уровне параллелизма.
- *Производительность при случайной записи* — плохая. Синхронизация дисков также препятствует параллельному выполнению малых случайных запросов записи. Плохая производительность при высоком уровне параллелизма.
- *Производительность при последовательном чтении* — очень хорошая для приложений с низким параллелизмом; ухудшается при повышении параллелизма.
- *Производительность при последовательной записи* — очень хорошая для приложений с низким параллелизмом; ухудшается при повышении параллелизма.
- *Частота отказов* — хорошая. RAID 3 может выдержать потерю любого диска в массиве без перевода приложения в состояние простоя. Потеря двух дисков в массиве станет причиной простоя и потребует выполнения

восстановления носителя. Обратите внимание, что устойчивость реализации RAID 3 деградирует с ростом числа дисков в массиве и что, в конечном счете, эта деградация может привести к большим потерям, чем полученная, на стоимости приобретения, экономия. Например, удвоение числа дисков в RAID 3 с 5 до 10 сохранит примерно 14% от стоимости приобретения (см. раздел стоимости приобретения RAID 3), в то же время это удвоение увеличит частоту отказов на 100%.

- *Длительность простоя* — хорошая. Продолжительность частичного простоя, связанного с выходом из строя одного диска в массиве равна времени на обнаружение неисправности и времени на замену диска в массиве. Длительность полного простоя, который влечет выход из строя двух или более дисков в массиве, адаптера, шины или других незащищенных компонент, увеличивается на время необходимое для выполнения процедуры восстановления носителя сервера Oracle.
- *Снижение производительности в течение отказа* — удовлетворительное. Потеря выделенного диска с контрольными данными не вызовет дополнительных издержек производительности до момента замены диска. Потеря диска с данными приведет к существенному снижению производительности приложения до момента замены диска, поскольку каждая операция ввода/вывода с вышедшим из строя диском потребует ввода/вывода на все другие диски в массиве. В течение замены любого диска в RAID 3, требуются операции ввода/вывода для реконструкции данных в заменяемом диске, которые будут конкурировать с операциями ввода/вывода приложения, что станет причиной снижения производительности.
- *Стоимость приобретения* — удовлетворительная. Стоимость дисковой емкости в  $g/(g - 1)$  раз выше, чем стоимость той же емкости для RAID 0, где  $g$  — число дисков в массиве, плюс стоимость специальных дисков с синхронизацией и специального контроллера для RAID 3. Таким образом, стоимость приобретения RAID 3 всегда будет выше, чем стоимость RAID 0, но обычно

меньше, чем стоимость RAID 1 при  $g > 2$ . Обратите внимание, что стоимость полезного пространства снижается с увеличением числа дисков в массиве. К примеру, использование массива с пятью дисками приведет к стоимости, равной 125% от стоимости аналогичной емкости RAID 0, но использование десяти дисков приведет лишь к 111% стоимости от стоимости RAID 0. RAID 3 требует специального контроллера RAID 3 в дополнение к адаптерам, необходимым для RAID 0.

- *Стоимость обслуживания* — удовлетворительная. Требуется обучение для создания процедур сопровождения для обработки различных событий возникновения простоя. Увеличение емкости требует либо закупки нового массива, либо переконфигурации существующих массивов.

### 3.2.6 Чередование блоков с распределенным контролем (RAID уровня 5)

RAID 5 подобен RAID 3 за исключением того, что размер сегмента чередования в RAID 5 можно настраивать, а также того, что контрольные блоки распределены по всем дискам в массиве. Сегмент чередования RAID 5 содержит либо данные, либо контрольную информацию. Любой запрос на запись в массиве RAID 5 требует выполнения шести ресурсоемких операций [11, Sun (1995)]:

1. Чтение блока, в который должна производиться запись.
2. Чтение соответствующего контрольного блока.
3. Вычет старой составляющей блока данных из контрольного блока.
4. Добавление новой составляющей блока данных в контрольный блок.
5. Запись контрольного блока.
6. Запись блока данных.

Энергонезависимый кэш в значительной мере снижает влияние издержек для операции записи, но его эффективность зависит от многих условий. Пакетные задания, генерирующие большой объем операций записи, могут быстро заполнить кэш, снижая его возможность устранить проблемы, присущие RAID 5.

RAID 5 является очень полезным для снижения стоимости подсистем ввода-вывода при хранении данных, требующих высокую устойчивость к отказам с высоким уровнем операций чтения, но не для часто изменяемых данных. Поскольку RAID 5 имеет крайне низкую производительность при выполнении операции записи, многие опытные проектировщики VLDB занимают негативную позицию по отношению к RAID 5. Для VLDB с интенсивным использованием операции чтения, в которых низкая производительность при восстановлении БД не так важна как стоимость приобретения, массив на основе RAID 5 предлагает приемлемую производительность при много меньшей стоимости чем RAID 1.

- *Производительность при случайном чтении* — отличная при всех уровнях параллелизма при условии, что каждый запрос на чтение использует один сегмент чередования. Использование слишком малых размеров сегментов чередования может привести к резкому ухудшению производительности при высоком уровне конкуренции.
- *Производительность при случайной записи* — плохая. Наихудшая при высоком уровне параллелизма. Цикл чтение-изменение-запись, присущий RAID 5 и необходимый для поддержания контрольной информации делают его на порядок хуже в сравнении с RAID 0. Использование дискового кэша может улучшить ситуацию, если он имеет достаточный размер для обработки необходимого уровня параллелизма.
- *Производительность при последовательном чтении* — отличная при малом размере сегмента чередования в средах с низким уровнем параллелизма. Также отличная в средах с высоким уровнем параллелизма при условии, что каждый запрос на чтение попадает в один сегмент чередования. Использование слишком малого размера сегмента чере-

дования может привести к резкому снижению производительности в средах с высоким уровнем параллелизма.

- *Производительность при последовательной записи* — удовлетворительная в средах с низким уровнем параллелизма. При высоком уровне параллелизма может быть на порядок хуже, чем у RAID 0. Большие объемы операций записи заполняют дисковый кэш RAID 5, сводя к нулю его возможности по смягчению низкой производительности массива. Как и для последовательного чтения, высокий уровень параллелизма при малом размере сегмента чередования снижает производительность.
- *Частота отказов* — хорошая. Выход из строя любого одного диска в массиве не влияет на доступность массива RAID 5 и приложений. Потеря двух дисков приведет к потере данных, которую можно устранить только с помощью восстановления носителя. Следует отметить, что надежность RAID 5 падает с ростом числа дисков в массиве и что потеря надежности может свести к нулю преимущества от невысокой стоимости приобретения. См. частоту отказов для RAID 3.
- *Длительность простоя* — хорошая. Продолжительность частичного простоя, связанного с выходом из строя одного диска в массиве равна времени на обнаружение неисправности и времени на замену диска в массиве. Длительность полного простоя, который влечет выход из строя более одного диска в массиве, адаптера, шины или других незащищенных компонент, увеличивается на время необходимое для выполнения процедуры восстановления носителя сервера Oracle.
- *Снижение производительности в течение отказа* — удовлетворительное. При чтении данных, располагающихся на неповрежденном диске, снижения производительности не будет. Запись данных на неповрежденный диск требует выполнения цикла чтение-изменение-запись. Чтение и запись данных, которые располагались на поврежденном диске, влечет за собой большие издержки и значительное снижение производительности, поскольку такие операции требуют

данных, располагающихся на всех дисках массива. Реконструкция массива при замене диска связана с резким ухудшением производительности в массиве.

- *Стоимость приобретения* — удовлетворительная. Стоимость дисковой емкости в  $g/(g - 1)$  раз выше, чем стоимость той же емкости для RAID 0, где  $g$  — число дисков в массиве, плюс стоимость контроллера RAID 5. Стоимость приобретения RAID 5 всегда выше, чем стоимость RAID 0, но в общем, меньше, чем стоимость RAID 3 и теоретически меньше чем RAID 1 для  $g > 2$ . В реальной жизни, ожидания производительности RAID 5 иногда превышают имеющиеся возможности по конфигурированию. Стоимость анализа и дополнительного оборудования, в итоге, может оказаться даже выше, чем у RAID 0+1.
- *Стоимость обслуживания* — удовлетворительная. Для достижения оптимальной производительности требуется обучение конфигурированию массивов с чередованием. Увеличение емкости требует либо закупки дополнительных массивов, либо реконфигурирования существующих.

### 3.2.7 Резюме

Дисковые технологии развиваются столь стремительно, что сложно сделать заключение по производительности реализаций различных технологий без некоторого рода обобщений, которые могут быть в конкретных случаях неверными. Если Вы живете по золотому правилу *знай свои цели и понимай доступные технологии*, то Вы будете периодически переосмысливать Ваши решения. Лучший путь для осознания Ваших целей и понимания имеющихся технологий — общение с опытными профессионалами и тщательная практическая проверка Ваших суждений. Увеличение риска для бизнеса требует увеличения важности задачи тщательного тестирования. Таблица 1 показывает оценки различных конфигураций RAID специально в применении к серверу Oracle.

## 3.3 Размер сегмента чередования

Чередование принесет пользу только в том случае, если Вы оптимизировали размер сегмента чередования. Поскольку различные размеры оптимизируют разные виды операций, наилучшая конфигурация VLDB будет требовать использования нескольких размеров сегментов в различных дисковых массивах. Наиболее популярным является мнение о преимуществах малых размеров сегментов чередования. Однако использование малых размеров сегментов для неподходящих видов операций может оказаться пагубным. В следующих разделах мы исследуем вопрос об использовании чередования с малым размером сегмента.

### 3.3.1 Чередование с малым размером сегмента

Конфигурации с малым размером сегмента чередования распределяют данные небольшими порциями по всем дискам таким образом, что любой запрос на ввод/вывод, в независимости от его размера, приводит к использованию всех дисков в массиве. Преимуществом такого подхода является высокая скорость обмена для любых запросов ввода/вывода. Недостатками подхода является то, что [Chen et al. (1993), 15]:

- Только один запрос на ввод/вывод может выполняться в любой момент времени. Таким образом, за возможность распараллеливания обработки одного запроса по всем дискам, приходится платить тем, что становится невозможным параллельная обработка большого числа запросов.
- Все диски должны тратить время для позиционирования для каждого запроса. Таким образом, если размер сегмента чередования меньше чем запрос на ввода/вывод, то два или более диска должны ожидать позиционирования при каждом запросе.

Мы можем избежать указанных недостатков массивов с малым размером сегмента чередования двумя путями — либо разрабатывая наши приложения и дисковую компоновку согласовано, либо, если это возможно, используя большие размеры сегментов чередования. Теперь, зная преимущества и недостатки чередования с малым

	Уровень RAID					
	Нет	0	1	0+1	3	5
Производительность контрольного файла	2	1	2	1	5	3
Производительность журнального файла	4	1	5	1	2	3
Производительность пространства <b>system</b>	2	1	2	1	5	3
Производительность сегментов сортировки	4	1	5	1	2	3
Производительность сегментов отката	2	1	2	1	5	5
Файлы с индексами, только чтение	2	1	2	1	5	1
Файлы с только последовательным чтением	4	1	5	1	2	3
Файлы с высокой активностью DBWR	1	1	2	1	5	5
Файлы с интенсивной прямой загрузкой	4	1	5	1	2	3
Защищенность данных	4	5	1	1	2	2
Стоимость приобретения и сопровождения	1	1	5	5	3	3

Таблица 1. Обзор относительной пригодности RAID-конфигураций от 1 (наилучшая) до 5 (наихудшая) для различных типов файлов Oracle. RAID 0+1 является наилучшим техническим решением, но вместе с тем, и самым дорогостоящим. Разумное применение RAID 5 массивов позволяет архитекторам дисковых подсистем снизить стоимость системы с минимальными потерями производительности и надежности. Данные из [11, Sun (1995), 28].

размером сегмента, мы можем сделать некоторые заключения об его использовании с сервером Oracle.

### 3.3.2 Операции с высоким уровнем параллелизма

Если Вы выполняете приложение с высоким уровнем параллелизма на дисковом массиве (т.е. много одновременно исполняемых процессов будут конкурировать за доступ к массиву), то Вы должны убедиться, что размер сегмента чередования имеет достаточно большой размер для того, чтобы любой запрос на ввод/вывод обслуживался ровно одним диском. В противном случае, число физических запросов на ввод/вывод может резко вырасти, что создаст огромную нагрузку на ядро операционной системы и всю подсистему ввода/вывода.

Нельзя гарантировать, что блоки данных Oracle будут выровнены по границам сегментов чередования поэтому, если размер сегмента чередования будет совпадать с размером блока обмена операций ввода/вывода, то это, вероятно, породит больше физических операций, чем число запросов на ввод/вывод. Выбор размера сегмента чередования вдвое больший, чем размер блока ввода/вывода даст 50%-ый шанс того, что операция потребует работы не более чем с одним дис-

ком. В общем, использование размера сегмента чередования в  $k$  раз большего, чем размер блока ввода/вывода даст вероятность того, что потребуется не более одного диска для обработки одного запроса, равную  $(k - 1)/k$ . Для выбранного  $k$  Вы должны гарантировать, что Ваш дисковый массив в состоянии выполнить в  $(k + 1)/k$  больше операций ввода/вывода, чем генерирует Ваше приложение.

Таким образом, если профиль доступа к массиву — это доступ, исключительно основанный на индексе или хэш-структуре при высоком уровне параллелизма, то выбор размера сегмента чередования вдвое или более раз, чем значение параметра **db\_block\_size** даст высокую производительность. Если профиль доступа к данным включает частые последовательные сканирования, оптимальный размер сегмента чередования должен быть, как минимум, вдвое больший, чем значение **db\_file\_multiblock\_read\_count** × **db\_block\_size**<sup>7</sup>.

Массив с малым размером сегмента чередования может оказаться плохим выбором для хранения табличного пространства, которое используется сервером Oracle для создания сегментов

<sup>7</sup> В Oracle 7.3 значение **db\_file\_multiblock\_read\_count** может настраиваться на уровне сессии, что дает более полный контроль над производительностью операций полного сканирования

сортировки в тех случаях, когда одновременно выполняется большое число сортировок. Малый размер сегмента чередования может показать удивительно низкую производительность и при использовании возможности параллельной обработки запроса (PQO), поскольку PQO использует несколько подзапросов, что создает высокий уровень параллелизма даже при одной активной пользовательской сессии.

### 3.3.3 Операции с низким уровнем параллелизма

Если лишь небольшое число процессов конкурируют за ввода/вывод на дисковом массиве, то Вы имеете больше свободы в выборе размера сегмента чередования, который может быть меньше, чем размеры логических запросов на ввода/вывод. При низком уровне параллелизма Вы должны сосредоточиться на увеличении пропускной способности дискового массива для одного процесса.

Прекрасным примером минимального уровня параллелизма с большим объемом последовательной записи на диск является процесс записи в журнальные файлы (LGWR). LGWR является однопоточным процессом, который выполняет большие объемы последовательной записи как при OLTP-нагрузках, так и в течение загрузки данных<sup>8</sup>.

Размещение журнальных файлов на выделенном дисковом массиве с малым размером сегмента чередования может дать исключительно высокую производительность, поскольку весь массив может быть нацелен на единственную задачу — распараллеливание операции записи для процесса, скорость работы которого является критичной для приложения.

Для хорошо разработанных приложений также существуют определенные преимущества в использовании дисковых массивов с малым размером сегмента чередования. Программы внесения больших изменений в базу данных могут быть сформированы как однопоточное пакетное зада-

ние, которое генерирует операторы модификации базы данных с большой интенсивностью. Такая техника проектирования позволяет эффективно свести к минимуму влияние низкой производительности массивов с малым размером сегмента чередования в средах с высоким уровнем параллелизма, делая их более привлекательными. Дисковый массив с малым размером сегмента чередования может быть использован максимально эффективно для выполнения, как отчетов, так и для загрузки данных, если процесс будет разработан или запланирован таким образом, что он не будет конкурировать за ввод/вывод.

Финальным примером однопоточного процесса порождающего большой объем операций ввода/вывода может служить операция с целым файлом данных, например процесс восстановления файла данных. Критическим вопросом при оценке доступности БД является вопрос о том, насколько быстро может быть восстановлена база данных после краха. Процедура восстановления обычно разрабатывается как единственно активная, поскольку крайне важна ее скорость. Таким образом, требования доступности и надежности, в общем случае, не ограничивают Вашу возможность использовать дисковые массивы с малым размером сегмента чередования для достижения максимальной производительности выполнения повседневных задач.

### 3.3.4 Трудности

Труднейшей проблемой при выборе правильного размера сегмента чередования дискового массива является то, что характеристики использования массива зачастую не могут быть всецело отнесены к той или иной категории. К примеру, файл к которому обычно осуществляется индексный доступ с высоким уровнем параллелизма, иногда может использоваться в операциях полного сканирования таблиц. Если Вы желаете действительно достичь компромиссного решения в Вашем анализе, наилучшим ответом почти всегда будет использование одной или нескольких нижеследующих техник:

- **Планирование заданий** — не запускайте больших загрузок данных в то время, когда имеется и без того высокая конкуренция за ввод/вывод со стороны процессов решающих

<sup>8</sup> Если быть более точным, LGWR записывает информацию только в случае выполнения изменений в СУБД с помощью стандартных операторов SQL. LGWR не записывает информацию при прямых загрузках данных (direct-path loads) или при выполнении транзакций без поддержки восстановления (unrecoverable)

важные задачи.

- *Размещение данных* — размещайте данные Вашей БД по файлам с подобными характеристиками ввода/вывода, размещайте файлы данных на одном массиве только в том случае, если они имеют одинаковый профиль доступа. К примеру, плохо выглядит идея разместить на одном массиве с малым размером сегмента чередования журнальные файлы, если Вы уже поместили на этот массив другие файлы БД, поскольку при этом возникнет конкуренция с процессом LGWR за ввод/вывод.
- *Разработка приложения* — отдавайте предпочтение однопоточным (или пакетным), высокоактивным загрузкам в БД вместо использования ресурсоемких транзакций, работающих в режиме высокого параллелизма. Плохое проектирование транзакций не только затрудняет построение хорошей дисковой конфигурации, но и приводит к активному использованию механизма фиксаторов и блокировок Oracle, что в свою очередь может породить каскад трудноразрешимых проблем.
- *Оптимизация SQL-операторов* — сведите к минимуму сортировки и полные сканирования таблиц. Устранение частого запуска операций сортировки или полного сканирования таблиц может дать замечательный результат для всего приложения.

К счастью, большинство конфликтов могут быть решены правильным сочетанием, во-первых, инвестиций в хорошее оборудование и программное обеспечение и, во-вторых, готовностью к оптимизации архитектуры приложения и использованию помощи со стороны поставщиков аппаратного и программного обеспечения.

### 3.3.5 Резюме

Не существует единого «наилучшего размера сегмента чередования» для всех видов приложений и даже более того, — для всех видов операций в пределах одного приложения. При хорошем проектировании подсистем ввода/вывода VLDB Вы должны быть готовы к использованию различных размеров сегмента чередования для

разных дисковых массивов. В общем, следует использовать наименьший из возможных размеров сегмента чередования для того, чтобы исключить возможность появления «горячих зон» (hot spot) на дисках, — с одной стороны, а с другой — Вы не должны уменьшать размер сегмента чередования настолько, чтобы появлялась опасность возникновения неэффективного использования дисков при высоких уровнях параллелизма. Вы должны использовать следующие руководящие принципы при выборе размера сегмента чередования:

- *Высокий уровень параллелизма* — если Вы ожидаете высокий уровень параллелизма ввода/вывода на дисковом массиве, то используйте размер сегмента чередования как минимум в два раза больший, чем наименьший запрос на ввод/вывод. Для файлов данных сервера Oracle с высоким уровнем параллелизма ввода/вывода это означает, что Вы никогда не должны использовать размер сегмента чередования, меньший чем  $2 \times \text{db\_block\_size}$ . Если файл данных часто используется для последовательного чтения, Вы также должны убедиться, что размер сегмента чередования равен как минимум  $2 \times \text{db\_block\_size} \times \text{db\_file\_multiblock\_read\_count}$ .
- *Низкий уровень параллелизма* — если уровень параллелизма мал для отдельного дискового массива, то возможно, Вы захотите использовать размер сегмента чередования меньший, чем размер запросов на ввод/вывод, с целью увеличения пропускной способности дискового массива при малом числе параллельных процессов. Чередование с очень малым размером сегмента может хорошо себя показать, например, для журнальных файлов.

Простой эмпирический метод для выбора наилучшего размера сегмента чередования для данной конкретной операции заключается в следующем:

1. Определите типы операций ввода/вывода, которые будут иметь место на рассматриваемом дисковом массиве. Наиболее общая ошибка в выборе размера сегмента чередования заключается в неверном заключении об

операциях ввода/вывода, присущих данным, располагаемым на дисковом массиве.

2. Создайте несколько массивов с различными размерами сегмента чередования на Вашем оборудовании для использования Вашим приложением.
3. Выполните идентичные задачи на каждом массиве. Проведите тесты с интенсивным использованием операции записи, включая создания табличных пространств, прямые загрузки данных, создания индексов. Проведите тесты с интенсивным использованием операции чтения, включая полные сканирования таблиц и сканирования с использованием индексов.
4. Оцените производительность на Ваших тестах и выберите конфигурацию, показавшую наилучший результат. Ключевыми показателями следует считать затраченное время, число физических операций чтения и записи и, для RAID 5, — число чтений блоков с контрольными суммами.

Мотивируемый размер сегмента чередования для конфигураций серверов Oracle будет лежать в диапазоне 16–32KB для массивов с малым размером сегмента чередования; для массивов с большим размером сегмента чередования — размер будет в два-четыре раза больше, чем максимальный размер блока ввода/вывода. На сегодняшний день большинство платформ поддерживают обмен с максимальным размером 64KB, некоторые платформы позволяют обмениваться блоками в 128KB<sup>9</sup>. Таким образом хорошими, для массивов с большим размером сегмента чередования, можно считать размеры в диапазоне от 128KB до 512KB.

### 3.4 Размер массива

Применение чередования для хранения данных позволяет равномерно распределить нагрузку на ввод/вывод между всеми доступными дисками, что побуждает использовать большие дисковые

<sup>9</sup> Максимальный размер блока ввода/вывода на физическом уровне, поддерживаемый системами UNIX, составляет от 64KB до 512KB на большинстве платформ.

массивы. Однако увеличение числа дисков в массиве ведет к росту частоты отказов. Рассмотрим эти два вопроса вместе для определения метода выбора оптимального числа дисков в массиве.

#### 3.4.1 Пропускная способность

Чередование, при корректном использовании, является прекрасным инструментом для увеличения пропускной способности. Чередование предоставляет прозрачное распределение операций ввода/вывода между относительно недорогими дисками. Это, с одной стороны, дает возможность обслуживать одновременно большое число небольших операций ввода/вывода, а с другой, — увеличить в разы скорость обмена большими операциями ввода/вывода по сравнению со скоростью самого быстрого диска [Chen et al. (1993), 151].

Для расчета минимального размера RAID-массива, необходимого для достижения требуемой пропускной способности, можно воспользоваться следующей простой техникой.

1. Определите устойчивую максимальную пропускную способность каждого из Ваших жестких дисков. Этот показатель оценивается как устойчивое максимальное число операций ввода/вывода для каждого жесткого диска<sup>10</sup>. Обозначим этот показатель как  $s$ , размерность — число операций ввода/вывода в секунду.
2. Оцените общее число  $t$  операций ввода/вывода, которое будут генерировать одновременно работающие транзакции.
3. Рассчитайте общее число  $r$  физических операций ввода/вывода в секунду, которые требуются от Вашего RAID-массива, с учетом размера сегмента чередования, используя формулу:

$$r = \frac{k + 1}{k} t,$$

где  $k$  — размер сегмента чередования деленный на размер блока ввода/вывода (описание

<sup>10</sup> Указанный устойчивый уровень операций ввода/вывода рассчитывается поставщиком жесткого диска как наивысший уровень которого может достичь диск с коэффициентом утилизации, не превышающем 60%–70%. Приведенное ограничение на утилизацию дает возможность удерживать время ожидания в предсказуемых границах.

Параллелизм	Блок ввода/вывода	Наилучший размер сегмента чередования	Примеры СУБД Oracle
низкий	малый	$k \times \text{db\_block\_size}$ , $k = 2, 3, 4, \dots$	DBWR
низкий	большой	$k \times \text{db\_block\_size}$ , $k = 0.25, 0.5, 1, 2, 3, \dots$	LGWR, ARCH, однопоточные загрузки данных и другие пакетные задания, системы принятия решений (DSS) без параллельного выполнения запроса, однопоточные операции сопровождения
высокий	малый	$k \times \text{db\_block\_size}$ , $k = 2, 3, 4, \dots$	OLTP-системы
высокий	большой	$k \times \text{db\_block\_size} \times \text{db\_file\_multiblock\_read\_count}$ , $k = 2, 3, 4, \dots$	Массовые формирования отчетности, любые параллельные операции СУБД Oracle, любые одновременно выполняющиеся пакетные задания с высокой загрузкой ввода/вывода

Таблица 2. Оптимальный размер сегмента чередования как функция от уровня параллелизма и размера блока ввода/вывода. Размер сегмента чередования не должен совпадать с размером блока ввода/вывода в дисковых массивах с высоким уровнем параллелизма поскольку границы сегментов чередования не обязательно совпадут с границам блоков ввода/вывода. Таким образом, рекомендуемый размер сегмента чередования должен быть в  $k$  раз больше чем размер блока ввода/вывода что будет гарантировать, что каждый запрос ввода/вывода будет обслуживаться одним диском с вероятностью  $(k - 1)/k$ .

см. выше в обсуждении размера сегмента чередования). К примеру, если Ваше приложение генерирует 300 операций ввода/вывода в секунду на массиве, а размер сегмента чередования в два раза больше запросов на ввод/вывод, то Ваш массив должен в состоянии поддерживать обработку 450 физических операций ввода/вывода в секунду.

4. Рассчитайте минимальное число дисков  $g = r/c$ , которые необходимо включить в дисковый массив, для поддержания требуемой нагрузки:

$$g = r/c = \frac{k+1}{kc}t$$

5. Обратите внимание, что требуемая пропускная способность  $r$  (определенная на шаге 2) может зависеть от размера Вашего дискового массива  $g$  (рассчитанного на шаге 4). Например, половина дисков от необходимого числа, могут обслужить примерно половину запросов на ввод/вывод от транзакций. Поставщик оборудования может дать информацию о максимальном рекомендуемом числе дисков в дисковом массиве.

Приведенные формулы могут использоваться для расчета дисковых массивов RAID 0 и RAID 0+1. Если Вы будете использовать эту технику для расчета массива RAID 5, то Вам необходимо увеличить оценку необходимого числа запросов на ввода/вывода на одну транзакцию для отражения того факта, что этот массив имеет большие издержки при выполнении каждой операции записи. Влияние дискового кэша, уменьшающего степень этих издержек, несколько усложняет эти вычисления.

### 3.4.2 Доступность

Расчет правильного размера дискового массива в случае использования RAID 3 или RAID 5 является несколько более трудоемким, чем просто расчет необходимых устройств для поддержки требуемой пропускной способности. Дополнительные сложности связаны с падением производительности конфигураций RAID 3 и RAID 5 во время отказа одного из дисков.

Напомним, что массивы RAID 3 и RAID 5 защищают данные от потерь и имеют существенные преимущества в стоимости приобретения по сравнению с RAID 1. Однако разработчики, использующие массивы RAID 3 и RAID 5, жертвуют, ради этого выигрыша, существенным повышением вероятности снижения производительности массива в течение отказа. Мы также видим, что одним из способов снижения стоимости RAID 3 или RAID 5 конфигураций является увеличение размера дискового массива (числа дисков в массиве), но эта экономия приводит к увеличению стоимости обслуживания из-за увлечения частоты отказов. Следовательно, определение уровня Вашего нежелания платить потерей производительности, вызванной выходом из строя диска, является важнейшим шагом в расчете лучшего для Вас размера дискового массива RAID 3 или RAID 5.

Производительность массива RAID 1 или RAID 0+1 не падает в результате выхода из строя диска благодаря наличию зеркальной копии. Процедура расчета оптимального размера массива RAID 1 или RAID 0+1, таким образом, не ограничена приведенными выше соображениями.

### 3.4.3 Резюме

Не существует «наилучшего размера дискового массива», годного для любых приложений. Большие, по размеру, дисковые массивы имеют лучшую пропускную способность, но платой за это является повышение частоты отказов дисков в массиве. Хорошо спроектированные дисковые подсистемы могут иметь как несколько различных RAID-конфигураций, так и несколько различных размеров, в зависимости от хранимых данных в каждом массиве.

Для массивов RAID 1 и RAID 0+1, большие размеры массивов увеличивают производительность без снижения устойчивости к отказам, конфигурации RAID 3 и RAID 5 при увеличении размера показывают лучшую производительность, но при этом их устойчивость к отказам падает.

## 3.5 Линейные устройства

Линейные устройства являются важным инструментом, который использует архитектор

VLDB, для снижения загрузки ЦПУ приложениями с высокой интенсивностью записи. *Линейное устройство (raw device)* — это неформатированный раздел диска в UNIX, который сервер Oracle может открыть как файл данных или как оперативный журнал минуя службы буферизации ввода/вывода UNIX-системы. Возможность сервера Oracle обходить буферизацию UNIX уменьшает объем кода операционной системы, который будет выполнен при вызове операций записи. В связи с этим, линейные устройства рекомендуются для хорошо спроектированных VLDB с высокими требованиями транзакционной пропускной способности.

Линейные устройства, на сегодняшний день, являются необходимыми, если Вы планируете использовать параллельный сервер Oracle (Oracle Parallel Server) под UNIX. Большинство UNIX-реализаций не позволяют двум узлам кластера иметь одновременный доступ к смонтированной файловой системе.

Стоимость обслуживания линейных устройств выше чем для файловых систем UNIX (ufs) [4, Millsap (1995a), 15–17]. Но для VLDB с интенсивной записью эта цена крайне мала в сравнении со стоимостью ненужной загрузки ЦПУ и уже без того высокой ценой администрирования системы с сотнями или тысячами дисковых устройств.

- *Производительность при случайном чтении* — Крайне незначительно лучше по сравнению с ufs.
- *Производительность при случайной записи* — Значительно лучше, в сравнение с ufs, из-за уменьшения объема выполняемого кода. Линейные устройства также позволяют реализовать асинхронный ввод-вывод, если он поддерживается операционной платформой.
- *Производительность при последовательном чтении* — Незначительно хуже в сравнение с ufs. Использование линейных устройств может катастрофически снизить производительность плохо оптимизированных SQL-приложений по сравнению с ufs-реализацией, поскольку UNIX-кэширование работает лучше кэширования сервера Oracle при полном сканировании таблиц.
- *Производительность при последовательной записи* — Значительно лучше, в сравнение

с ufs, из-за уменьшения объема выполняемого кода и возможности асинхронного ввода/вывода.

- *Частота отказов* — Риск возникновения возрастает из-за необходимости в более опытном администраторе.
- *Длительность простоя* — Риск увеличения возрастает из-за необходимости в более опытном администраторе.
- *Снижение производительности в течение отказа* — Отличия от нормальной производительности обусловлены замечаниями приведенными выше.
- *Стоимость приобретения* — Та же, что для ufs.
- *Стоимость обслуживания* — Хуже, чем для ufs. Требуется большее обучение и оплата труда персонала для конфигурирования и обслуживания линейных устройств. Конфигурирование линейных устройств также требуют закупки или разработки программных средств для упрощения управления подсистемой ввода/вывода. Различия стоимости администрирования линейных устройств и ufs незаметны на фоне общей стоимости системы ввода/вывода для VLDB с тысячами дисков. Стоимость обслуживания оперативных журнальных файлов не изменяется, поскольку эти файлы архивируются также как на ufs процессом ARCH.

### 3.6 Конфигурация Oracle

Не существует такой вещи, как единая оптимальная конфигурация СУБД Oracle для всех VLDB-приложений. Ваша оптимальная смесь *скорости, надежности и экономичности* зависит от Ваших конкретных задач. Однако мы можем сделать некоторые заключения о том, как Вы могли бы выполнить оптимизацию Вашей конфигурации VLDB, если бы Вы не были связаны экономическими ограничениями:

- *Контрольные файлы*
  - *n*-кратное дублирование RAID 0+1

- размещение на независимых подсистемах ввода/вывода
- линейные устройства в случае OPS
- *Оперативные журнальные файлы*
  - все файлы одного размера
  - линейные устройства
  - $n$ -кратное дублирование RAID 0+1
  - малый размер сегмента чередования
  - размещение на выделенных дисках
- *Архивные журнальные файлы*
  - ufs
  - $n$ -кратное дублирование RAID 0+1
  - малый размер сегмента чередования
  - размещение на выделенных дисках отдельно от оперативных журнальных файлов
- *Файлы данных*
  - все файлы одного из трех стандартных размеров
  - линейные устройства, если файл активно изменяется, либо используется OPS
  - ufs, если используется полное сканирование таблиц
  - $n$ -кратное дублирование RAID 0+1
  - размер сегмента чередования равен, как минимум,  $2 \times$  размер ввода/вывода, если высокий уровень параллелизма
  - размер сегмента чередования меньше чем  $2 \times$  размер ввода/вывода, если низкий уровень параллелизма
  - RAID 5, если низкая активность изменений файла
- *Другие файлы*
  - ufs
  - $n$ -кратное дублирование RAID 1
  - размещение согласно стандарту OFA

Если Вы ограничены стоимостью решения, то Вы должны определить наименее дорогие возможности для конфигурации, дающие Вам максимальные преимущества на единицу затрат.

### 3.6.1 Примеры конфигураций

Вы можете смешивать и оценивать технологии большим числом способов. Три простых конфигурации БД сервера Oracle представлены в таблице 3.

Каждая из них требует разного уровня инвестиций для достижения заданных целей, определенных в конфигурации.

- *Конфигурация А* — Дублирование ( $n = 2, 3$ ) RAID 0+1 для всех файлов, линейные устройства везде, где это возможно. Может использоваться для решения критически важных задач, OPS, очень высокая скорость транзакций, очень высокая скорость выполнения запросов, сложность обслуживания, высокая стоимость.
- *Конфигурация В* — RAID 0+1 на линейных устройствах для файлов с интенсивным уровнем изменений, RAID 5 на ufs для нечасто изменяемых файлов. Сравнительно высокая доступность, непригодна для OPS, средняя скорость выполнения транзакций, очень высокая скорость выполнения запросов, средняя сложность обслуживания, средняя стоимость.
- *Конфигурация С* — RAID 0+1 на линейных устройствах для файлов с интенсивным уровнем изменений, RAID 0 на ufs для всех остальных файлов Oracle. Непригодна для решения критически важных задач, непригодна для OPS, средняя скорость выполнения транзакций, очень высокая скорость выполнения запросов, низкая сложность обслуживания, низкая стоимость.

## 4 Журнальные файлы

Сервер Oracle использует выделенный процесс, называемый (*писателем журнальных файлов redo log writer, LGWR*) для записи пакетов записей о транзакциях на диск. Работа LGWR позволяет серверу Oracle записывать подтвержденные измененные блоки на диск асинхронно по отношению к пользовательским запросам на подтверждение транзакций и поддерживать при этом целостность данных даже если сервер прервет свою работу в самый неподходящий момент. Выделенный

Тип файла	Конфигурация		
	A	B	C
Оперативные журнальные	raw 0+1	raw 0+1	raw 0+1
<b>system, temp, rbs</b>	raw 0+1	raw 0+1	ufs 0
Другие файлы данных	raw 0+1	ufs 5	ufs 0
Архивные журнальные	ufs 0+1	ufs 0+1	ufs 0+1
Контрольные файлы	ufs 0+1	ufs 0+1	ufs 0+1
Другие файлы	raw 0+1	ufs 5	ufs 0

Таблица 3. Примеры дисковых конфигураций для Oracle-приложений. Эта таблица описывает три простые конфигурации с использованием линейных устройств (raw) и файловых систем UNIX (ufs) и различных RAID-массивов для решения разных бизнес-задач. Цели и достоинства конфигураций A, B и C описаны в тексте.

фоновый процесс для последовательного упорядочивания и пакетной записи векторов изменений на диск — это основа высочайшей производительности сервера Oracle.

Производительность процесса LGWR является критической для скорости работы OLTP-систем и процедур загрузки данных которые не используют возможность работы в режиме «без восстановления» (unrecoverable). В средах, где не выполняются интенсивные изменения в базе данных с помощью стандартных операторов SQL, LGWR не испытывает значительной нагрузки. Дegradaция производительности и/или увеличение времени простоя будут наблюдаться главным образом в высокоинтенсивных OLTP-системах; именно поэтому задача конфигурирования журнальных файлов является темой этого раздела.

#### 4.1 Баланс производительности и доступности

Борьба между производительностью и надежностью особенно ярко проявляется на примере журнальных файлов. Для оптимизации производительности Вам необходимо конфигурировать LGWR и DBWR так, чтобы операции записи выполнялись как можно реже. Однако для минимизации времени, необходимого для выполнения доката данных (roll-forward) при восстановлении инстанции, Вам необходимо конфигурировать LGWR и DBWR так, чтобы операции записи производились как можно чаще. Решение этой проблемы заключается в достаточном понимании технологий для того, чтобы выбрать значение параметра `log_checkpoint_interval`, которое было бы и достаточно большим и достаточно малым

для удовлетворения обоих требований.

Понимание технологий, относящихся к конфигурированию журнальных файлов, лежит в осознании двух ключевых событий: контрольной точки сервера Oracle и восстановление инстанции сервера Oracle.

#### 4.2 Контрольные точки сервера Oracle

Контрольная точка сервера Oracle — это событие, которое начинается в момент, когда LGWR оповещает DBWR о необходимости записи всех модифицированных блоков из кэша данных СУБД, включая как подтвержденные, так и не подтвержденные данные, в файлы данных [8, Concepts (1992), 23, 9-12]. Контрольные точки вызывают непродолжительные периоды высокой загрузки системы — эффект, который администратор базы данных стремится минимизировать с помощью настроек [7, Admin (1992), 24, 6-7]. Обычные контрольные точки, возникающие в процессе работы, появляются в следующих случаях:

- *Переключение журнального файла (log switch)* — когда LGWR заполняет текущий журнальный файл и пытается переключиться на следующий в круговой очереди.
- *Предопределенный интервал* — LGWR будет порождать контрольную точку либо когда запишет число блоков операционной системы равное значению параметра `log_checkpoint_interval`, с момента последней контрольной точки, либо когда пройдет число секунд равное значению параметра `log_`

**checkpoint\_timeout**, с момента последней контрольной точки.

Размер журнального файла и параметры **log\_checkpoint\_interval** и **log\_checkpoint\_timeout** оказывают наиболее важное влияние на изменение производительности при обычной работе механизма контрольных точек. Многие администраторы БД отключают тайм-аут, устанавливая его в ноль и регулируют работу механизма контрольных точек только с помощью параметра **log\_checkpoint\_interval**. Некоторые администраторы деактивируют оба параметра и контрольные точки возникают лишь вследствие переключения журнальных файлов.

### 4.3 Восстановление инстанции сервера Oracle

Длительность времени, необходимая для выполнения восстановления инстанции сервера Oracle является важным параметром для архитектора VLDB, поскольку она определяет период, в течение которого приложение будет недоступно после возникновения следующих событий:

- *Отказ узла без кластера* — отказ процессора, шины или памяти на незащищенном с помощью кластера узле.
- *Отказ узла параллельного сервера Oracle* — отказ одного или нескольких узлов в конфигурации с параллельным сервером Oracle будет причиной кратковременной недоступности всего кластера, которая оканчивается к моменту, когда «выжившие» узлы завершат восстановление нитей отказавших узлов.

Реализация отложенного, «по требованию», отката в Oracle 7.3 делает период недоступности кластера зависящим, в основном, от времени которое необходимо для реконструкции буферного кэша. Диаграмма, сравнивающая время восстановления после отказа в Oracle 7.3 и Oracle 7.2, приведена на рисунке 2.

Восстановление инстанции сервера Oracle — настраиваемый процесс, время выполнения которого, в основном, зависит от двух факторов [3, Maulik and Patkar (1995)]:

- Объем повторно-применяемой информации (redo-информации), которая была сгенерирована с момента последней контрольной точки (чем меньше, тем лучше), и
- Качество буферного кэша базы данных в течение выполнения восстановления (чем выше, тем лучше).

На первый фактор, в основном, влияет скорость генерации redo-информации и частота выполнения контрольных точек. Уменьшение объема генерируемой redo-информации является задачей разработчика приложения и связано с минимизацией «веса» транзакций, минимизацией числа транзакций, необходимых для достижения требований бизнеса и использованием там, где это допустимо, режима работы без восстановления. Уменьшение частоты выполнения контрольных точек достигается изменением параметра, определяющего интервал между контрольными точками или, возможно, увеличением размера журнального файла.

Улучшения, относящиеся ко второму фактору, Вы можете достичь с помощью увеличения размера буферного кэша базы данных и уменьшения, тем самым, вероятности промахов в буферном кэше в течение выполнения процедуры восстановления. Увеличение значения параметра **db\_block\_buffers** для работы процесса восстановления (ценой уменьшения значения параметра **shared\_pool\_size**, если необходимо) в общем, уменьшает число промахов в буферном кэше. В течение процедуры восстановления, при котором не требуется выполнения восстановления носителя, Вы не имеете других способов влиять на процент попаданий в буферный кэш, поскольку должна быть повторно применена вся информация, которая была сгенерирована с момента последней контрольной точки <sup>11</sup>.

Меняя параметр **log\_checkpoint\_interval** в диапазоне от нескольких килобайт до нескольких сотен мегабайт (в блоках операционной системы) можно достичь приемлемого компромисса с про-

<sup>11</sup> Получены важные результаты, относящиеся к восстановлению носителя сервера Oracle, когда повреждены несколько файлов данных. Рекомендуется сделать несколько циклов *восстановление-файла/восстановление-инстанции* для каждого файла данных. В этом случае может наблюдаться лучшее качество использования буферного кэша базы данных [3, Maulik and Patkar (1995)].

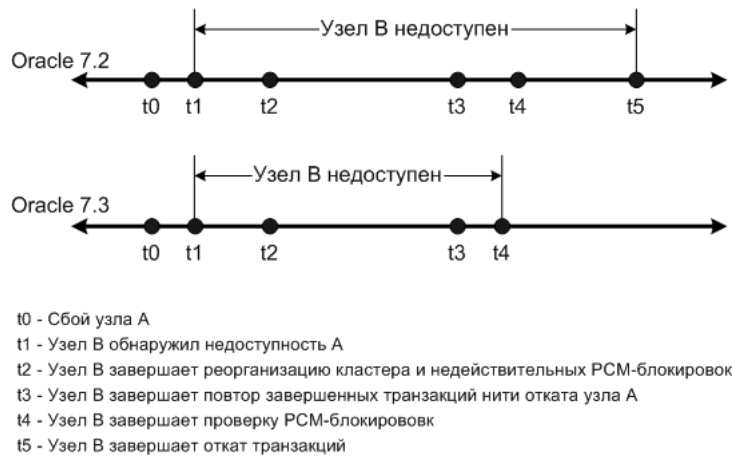


Рис. 2. Временная диаграмма восстановления после отказа для параллельного сервера Oracle 7.2 и 7.3. OPS 7.3 откладывает откат транзакций для минимизации периода недоступности кластера при отказе узла, что дает улучшение производительности восстановления по сравнению с OPS 7.2 или ниже. В OPS 7.3 время завершения отката инстанции не связано с доступностью приложений, поскольку все откаты запускаются «по требованию», на уровне транзакций.

изводительностью в OLTP-среде. Скорость доката при восстановлении варьируется почти также как кардинально, как и требования к скорости доката<sup>12</sup>. В средах с жестко заданными требованиями к производительности и надежности малое изменение `log_checkpoint_interval` может привести к значительному изменению времени восстановления инстанции. Вследствие этого поиск оптимального значения `log_checkpoint_interval` часто требует тестирования в среде эксплуатации.

#### 4.4 Размещение журнальных файлов

В хорошей конфигурации VLDB высокоактивные, с точки зрения ввода/вывода, журнальные файлы должны быть изолированы от других файлов с высокой активностью ввода/вывода настолько, насколько это возможно. Физическая изоляция на выделенных дисках и контроллерах позволит Вам снизить до минимума шанс возникновения «узкого места» при выполнении операций ввода/вывода в журнальные файлы. Техника изолирования хорошо комбинируется с рассмот-

ренным ранее желанием размещать журнальные файлы на дисковом массиве с малым размером сегмента чередования доступ к которому наиболее эффективен при работе однопоточного (с очень низким уровнем параллелизма) процесса.

В дополнение к отделению журнальных файлов от других файлов, фактически Вам необходимо физически отделить журнальные файлы друг от друга. В то время как LGWR-процесс пишет в журнальный файл с максимально возможной скоростью, процесс ARCH читает журнальный файл с максимально возможной скоростью (в любой момент времени он может читать один из журнальных файлов, кроме того, который открыт LGWR). Одновременно ARCH пишет с максимально возможной скоростью в один архивный журнальный файл. Использование чередования с малым размером сегмента может свести к минимуму конкуренцию между LGWR и ARCH.

Сервер Oracle позволяет, если Вы этого желаете, архивировать журнальные файлы непосредственно на ленточное устройство. Имеется несколько причин, из-за которых администраторы VLDB почти всегда выполняют архивирование на диск и лишь потом делают  $n$  копий файлов на ленточный накопитель.

- *Надежность лучше* — ленточный накопи-

<sup>12</sup> Администраторы СУБД Oracle дают большой разброс при оценке скорости доката. Я слышал как низкие оценки — 50KB/сек, так и высокие — до 1,750KB/сек.

тель существенно менее надежен, чем диск. Оставлять лишь одну копию критически важного файла на ленточном носителе является не очень хорошим решением.

- *Пропускная способность лучше* — вы должны помнить, что ARCH-процесс не является чисто потоковым процессом копирования, но выполняется лучше на устройствах с возможностью произвольного доступа (random I/O). Архивирование на диск выполняется намного быстрее чем на ленточный накопитель, что в свою очередь снижает вероятность возникновения ненужного «узкого места».
- *Управление ошибками лучше* — управление дисковыми устройствами, при переполнении, много надежнее, чем ленточными устройствами. Управление пространством для дисков может быть автоматизировано с помощью программного обеспечения, в то время как для ленточных устройств требуется вмешательство человека.
- *Время восстановления быстрее* — многие администраторы БД хранят максимально возможное число архивных журнальных файлов на дисках, которые могут быть необходимы для восстановления с момента последнего «горячего» копирования. Такая техника уменьшает продолжительность процесса восстановления на время необходимое для нахождения требуемой ленты, ее монтирования и чтения с нее архивных журнальных файлов на диск.

#### 4.5 Размер журнального файла

Оптимальный размер журнального файла в системе зависит от объема генерируемой redo-информации и требований к продолжительности восстановления инстанции. Для расчета правильного интервала между контрольными точками и размера журнального файла для Вашей системы используйте следующий метод:

1. Определите требования к времени восстановления после краха. Обозначим это время, заданное в секундах, как  $t$ .

2. Рассчитайте скорость, с которой система применяет redo-информацию при восстановлении инстанции. Обозначим эту скорость, заданную в байтах в секунду, как  $r$ .

3. Установите значение параметра **log\_checkpoint\_interval** в  $r \times t/b$ , где  $b$  — размер блока ввода/вывода в байтах в Вашей операционной системе.

4. Создайте журнальные файлы размером  $f = k \times r \times t$ , для некоторого целого  $k$  (т.е.  $f$  кратно  $r \times t$ ).

5. Если новые контрольные точки накапливаются без завершения предыдущих, то Вы должны увеличить значение **log\_checkpoint\_interval**. Вы можете определить частоту возникновения таких ситуаций сравнив значения статистик `background_checkpoints_started` и `background_checkpoints_completed` из **v\$sysstat**. Если значения отличаются более чем на 1, значит контрольные точки не завершались к тому моменту, когда начинались новые.

6. Установите **log\_buffer** в  $l = 3 \times n \times s$ , где  $n$  — число дисков в массиве, хранящем журнальные файлы и  $s$  — размер сегмента чередования массива в байтах. Наибольший размер операции записи, генерируемый LGWR, будет примерно равен  $l/3 = n \times s$ <sup>13</sup>.

Приведенный метод сводит задачу выбора оптимального размера журнальных файлов к задаче правильного выбора множителя  $k$ . Факторы, которые будут влиять на выбор большего значения, могут быть следующими:

- снижение частоты переключения журнальных файлов;

<sup>13</sup> В OLTP-операциях, частые подтверждения транзакций будут порождать записи LGWR и LGWR будет делать очень много малых записей. В пакетных операциях, нечастые подтверждения транзакций будут порождать большие объемы не записанной в журнальный файл информации, расположенной в журнальном буфере. Если информация будет накапливаться быстрее, чем LGWR будет получать команды для записи, то заполненность буфера более чем на 1/3 будет основанием для начала записи до  $l$  байт, где  $l$  — размер журнального буфера, заданного с помощью **log\_buffer**.

- упрощение процедуры размещения журнальных файлов;
- снижение сложности при полном восстановлении за счет снижения числа обрабатываемых файлов;
- снижение частоты возникновения событий ожидания контрольных точек и занятости процесса архивирования.

Факторы, которые будут влиять на выбор меньшего значения, могут быть следующими:

- снижение стоимости отказа, связанной с потерей всех копий активного журнального файла <sup>14</sup>;
- улучшение гибкости в предотвращении ситуации переполнения файловой системы, хранящей архивные журнальные файлы;
- снижение потерь данных в конфигурациях с резервной (standby) БД.

Общим правилом при выборе размера журнального файла можно принять пожелание о том, что переключение журнальных файлов не должно происходить чаще, чем два раза в час.

## 4.6 Число журнальных файлов

Число оперативных журнальных файлов дает Вам возможность определить, будут ли контрольные точки и архивирование журнальных файлов причиной возникновения «узких мест» при обработке транзакций. Выбор правильного числа журнальных файлов поможет Вам снизить:

- *Ожидания занятой контрольной точки (checkpoint busy wait)* — ожидание занятой контрольной точки возникает, когда LGWR пытается переключиться на журнальный файл до того, как связанная с этим журнальным файлом предыдущая контрольная точка была завершена.

<sup>14</sup> Вы должны попытаться защитить себя от столь катастрофического события, используя *n*-кратное дублирование аппаратуры для хранения журнальных файлов. Если Вы не в состоянии позволить себе дублирование с помощью аппаратного решения, Oracle7 дает возможность введения зеркалирования с помощью программных средств, используя несколько членов журнальных файлов в группе журнальных файлов.

- *Ожидание занятого процесса архивирования (archiver busy wait)* — ожидание занятого процесса архивирования возникает, когда LGWR пытается переключиться на журнальный файл, который еще не было скопирован в архивный журнальный файл процессом ARCH.

Ожидание занятой контрольной точки часто возникало в дни, когда в большинстве установок сервера Oracle использовалось всего два журнальных файла, устанавливаемых по умолчанию при установке. Вы можете использовать следующую процедуру для определения наличия и устранения ожиданий занятой контрольной точки:

1. Проверьте событие *log file switch (checkpoint incomplete)* в `v$session_wait`. Если Вы установили параметр `log_checkpoints_to_alert` в значение `true`, то Вы можете определить возникновение этой ситуации с помощью текстового вхождения «cannot allocate» в файле `alert.log`.
2. Снизить частоту возникновения события ожидания контрольной точки Вы можете:

- увеличивая число оперативных журнальных файлов для снижения вероятности того, что LGWR сможет заполнить все файлы до того, как будет завершена контрольная точка; или
- добавляя DBWR процессы для увеличения скорости выполнения контрольной точки (только если Вы используете синхронную запись); или
- увеличивая значение параметра `db_block_checkpoint_batch` для увеличения скорости выполнения контрольной точки; или
- уменьшая значение параметра `db_block_buffers` для снижения объема работы в контрольной точке; или
- снижая число и размеры сегментов отката в базе данных (описанных в ниже следующем разделе)

Событие ожидания занятого процесса архивирования обычно возникает при генерации большого числа транзакций во время пакетной обработке, когда скорость записи redo-информации LGWR превышает возможности копирования процесса ARCH. Оно также служит причиной возникновения ожидания контрольной точки — ARCH становится «узким местом» для завершения всех транзакций в системе.

Следующие техники могут использоваться для смягчения проблемы:

- *Чередование с малым размером сегмента* — разместите Ваши архивные журнальные файлы на дисковом массиве с малым размером сегмента чередования для увеличения скорости обращения к файлам процессом ARCH.
- *Большее число оперативных журнальных файлов* — создайте достаточное число оперативных журнальных файлов с тем, чтобы LGWR не смог заполнить все журнальные файлы до того как процесс ARCH закончил копировать.
- *Увеличение числа ARCH-процессов* — используйте несколько ARCH-процессов с помощью команды сервера Oracle **alter system archive log all to . . . .** Консультанты службы поддержки Oracle подготовили несколько инструментов для автоматизации этого процесса.

## 5 Планирование табличных пространств

При преобразовании логической модели данных в физическую, Вам требуется сделать долгосрочные решения о том, как Вы поделите сегменты базы данных между табличными пространствами. Следующие ограничения определяют то, как Вы должны распределить сегменты по табличным пространствам:

- *Производительность операций ввода/вывода* — каждое табличное пространство должно включать сегменты, имеющие схожие характеристики ввода/вывода (уровень параллелизма, объем) — такое разбиение упростит

выбор размера сегмента чередования и емкости дискового массива. Объединение сегментов, используемых только для чтения, в табличное пространство только для чтения, снижает объем передаваемых данных при резервном копировании и восстановлении, а также снижает издержки на РСМ-блокировки. Разделение сегментов, в которые часто осуществляется запись, от сегментов с низкой интенсивностью записи придает больше гибкости при конструировании процедуры «горячего» резервного копирования.

- *Устойчивость к отказам* — малые табличные пространства позволяют минимизировать простой приложения во время работ, требующих перевода этих табличных пространств в автономный режим (offline). Табличное пространство **system** не может быть переведено в автономный режим, также как не может быть удалено и пересоздано, поэтому храните минимально необходимое число сегментов в этом табличном пространстве. Изоляция сегментов отката в минимально необходимом числе табличных пространств снижает частоту возникновения простоев, поскольку табличное пространство с активным (online) сегментом отката невозможно перевести в автономный режим. Хранение связанных, с помощью ссылочной целостности, сегментов в небольшой группе табличных пространств даст Вам возможность использовать процедуру восстановления на заданный момент времени («point-in-time»), введенную в Oracle8.
- *Управление пространством* — изоляция сегментов с коротким временным жизненным циклом минимизирует влияние фрагментации свободного табличного пространства, которая может блокировать выделение новых экстендов сервером Oracle. Администраторам VLDB принесет пользу возможность использования умалчиваемых параметров хранения, определенных на уровне табличного пространства, вместо явного указания этих параметров в описании сегментов.
- *Управление квотами* — управление квотами пространства в сервере Oracle выполняется на уровне пользователей и табличных про-

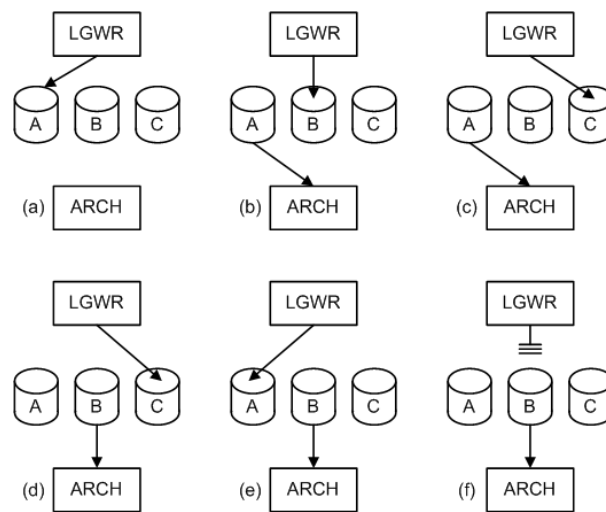


Рис. 3. Ожидания занятого процесса архивирования. Рисунок демонстрирует, каким образом быстрый LGWR-процесс может обогнать медленный ARCH-процесс, что приведет к возникновению «узкого места» и к снижению производительности. Эпизод (a) демонстрирует состояние инстанции сразу после старта — процесс LGWR пишет в файл A, процесс ARCH ничего не делает. В эпизоде (b) LGWR переключился на запись в файл B, позволяя ARCH открыть A и начать его архивирование. В (c) LGWR переключился на файл C, в то время как ARCH еще не обработал A. В эпизоде (d) ARCH переключился на обработку файла B, в (e) LGWR пишет в файл A. В эпизоде (f) LGWR пытается переключиться на файл B, но не может выполнить этого, поскольку ARCH еще не сделал его архивную копию. Начиная с этого момента, из-за ARCH, системой будут блокироваться попытки фиксации транзакции до тех пор, пока ARCH не переключится на файл C.

странств. Поэтому размещайте сегменты одной схемы в небольшом числе табличных пространств.

### 5.1 Назначение сегментов табличным пространствам

Метод, приведенный ниже, поможет Вам принять решение о том как разделить сегменты по табличным пространствам.

1. В табличном пространстве **system** храните только сегменты словаря данных (сегменты, принадлежащие схеме **sys**). Перенесите таблицу **aud\$** в табличное пространство отличное от **system**, это позволит Вам управлять размером таблицы (усекать ее) не влияя на фрагментацию свободного пространства в табличном пространстве **system**. Некоторые эксперты рекомендуют отредактировать файл **sql.bsq** (только строки расположенные ниже вхождения //) с тем, чтобы изменить параметры хранения таблиц словаря данных, связанных с хранимыми процедурами и триггерами. Для полной уверенности, делайте это совместно со службой поддержки Oracle.
2. Создайте два или более табличных пространства, предназначенных исключительно для временных сегментов. Создайте программу, которая позволит быстро переключать на доступное временное табличное пространство, отличное от **system**, в случае недоступности стандартного временного табличного пространства [10, To (1995), 6.12].
3. Создайте одно или более табличных пространств, предназначенных исключительно для сегментов отката. Не помещайте сегменты отката в табличные пространства отличные от тех, что были специально для этого спроектированы.
4. Изолируйте короткоживущие сегменты, используя минимально возможное количество табличных пространств. Не помещайте короткоживущие прикладные таблицы и индексы в табличные пространства, отличные от тех, что были специально разработаны для короткоживущих сегментов.
5. По возможности изолируйте сегменты только для чтения (**read only**) в табличных пространствах, которые содержат исключительно такие сегменты. Переведите эти табличные пространства в режим «только для чтения».
6. Проведите классификацию оставшихся сегментов по их размерам. В каждом табличном пространстве храните сегменты с похожими размерами.
7. Ограничьте максимальный размер табличного пространства в пределах 10GB. Перевод небольшого табличного пространства в автономный режим (**off-line**) потенциально влечет лишь ограниченную недоступность базы данных. Недоступность большого табличного пространства, из-за потери файла данных, с большей вероятностью приведет к недоступности всей базы данных. Использование малых табличных пространств позволит получить преимущества от распараллеливания процедуры восстановления на заданный момент времени в Oracle8.
8. Если Вы используете чередование с малым размером сегмента, то нет необходимости разделять индексы и таблицы для распределения нагрузки на диски. Однако, если Ваш режим эксплуатации требует регулярной перестройки индексов, то Вам имеет смысл рассмотреть возможность поместить индексы в отдельное табличное пространство для минимизации возможного влияния возникающей фрагментации свободного пространства из-за применения оператора **drop index**.

## 6 Параметры хранения

Параметры хранения были долгое время горячей темой для обсуждения, поскольку многие администраторы Oracle верили, что важно пытаться разместить каждый сегмент в базе данных в не более чем одном экстенде. Многие люди были обучены тому, что наличие большого числа экстендов негативно влияет на производительность всех DML-операторов Oracle. Время, потраченное на «сжатие» таблиц и индексов в один экстенд —

это время, которое можно было бы потратить на что-то более стоящее <sup>15</sup>.

## 6.1 Maxextents

Действительная причина того, что администраторы БД Oracle должны обращать внимание на параметры хранения, заключается в том, что в Oracle до версии 7.3 максимальное значение параметра **maxextents** ограничено и зависит от значения параметра **db\_block\_buffers**. Опасность того, что приложение может быть прервано из-за ошибки «*max # extents reached*» заставляла администраторов помещать сегменты в ограниченное количество сегментов. С возможностью **maxextents unlimited**, появившейся в версии Oracle 7.3, эта опасность исчезла.

Забавно, но для VLDB, параметры хранения являются важными по прямо противоположным соображениям. Для VLDB существуют несколько случаев, когда *желательно*, чтобы сегмент хранился в более чем одном экстенсте.

- Наличие множества экстенстов позволяет эффективнее использовать свободное пространство с помощью команды **truncate ... drop storage**.
- Вам *желательно*, чтобы сегменты отката имели несколько экстенстов для снижения загрузки системы из-за динамического выделения и освобождения экстенстов [5, Millsap (1995b)].
- Вы хотите иметь возможность закупать новые диски с ростом системы и чтобы СУБД Oracle динамически размещала новые экстенсты в соответствии с этим ростом.
- Вам *желательна* предварительная фрагментация во всех табличных пространствах, содержащих временные сегменты и сегменты отката для минимизации накладных расходов, связанных с выделением экстенстов.

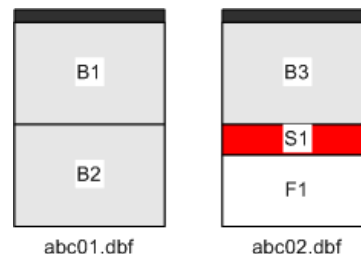
<sup>15</sup> Если Вы можете получить улучшение производительности DML-операторов от «сжатия» сегмента, то Вы сможете получить то же улучшение с помощью правильного управления сегмента с несколькими экстенстами [5, Millsap (1995b)].

## 6.2 Свойства параметров хранения

Хорошо выбранные параметры хранения VLDB сервера Oracle должны иметь следующие свойства:

- *Кратность размеру блока данных* — все размеры экстенстов должны быть кратные размеру блока данных базы данных. Oracle будет всегда округлять размер экстенста до размера блока данных. Например, неразумно пытаться установить значение **10KB** для параметра **initial** в базе данных с размером блока данных 8KB. Начальный экстенст с 8KB-блоком данных всегда будет занимать как минимум 16KB, в независимости от того что Вы запросите.
- *Кратность размеру файла данных* — все размеры экстенстов должны быть кратные используемым размерам файлов данных. Это позволит наилучшим способом использовать пространство в файлах данных без возникновения потерь.
- *Небольшое число размеров экстенстов* — число различных размеров экстенстов в каждом табличном пространстве должно быть небольшим, к примеру — только один размер. Если Вы используете несколько размеров экстенстов, все они должны быть кратные, либо делить друг друга нацело.

На рисунке ниже показаны два файла данных, в которых выделено по два экстенста. Малые области в начале каждого файла представляют заголовки файлов <sup>16</sup>.



<sup>16</sup> Размер заголовка равен одному блоку данных если Вы используете файловую систему и двум блокам, в случае использования линейных устройств.

На этом примере большие экстенды *B1*, *B2* и *B3* имеют такой размер, что ровно два экстенда могут разместиться в каждом файле данных. Файл данных, называемый **abc01.dbf** содержит ровно два таких больших экстенда. В **abc02.dbf** находится один большой экстенд *B3*, а также небольшой экстенд, обозначенный *S1*. Оставшееся свободное пространство, помеченное как *F1*, имеет большой размер, — почти половину размера файла данных, — но недостаточный для хранения еще одного большого экстенда, такого как *B1*, *B2* и *B3*.

Экстенд *S1* — причина появления неиспользуемого пространства, поскольку его размер отличается от размера больших экстендов, что порождает неиспользуемый фрагмент свободного пространства *F1*. Фактически, если размер больших экстендов не кратен размеру *S1*, то это приведет к тому что оставшееся свободное пространство в **abc02.dbf** не сможет быть полностью освоено.

Для понимания истинного влияния этого типа проблемы на VLDB, Вам необходимо помнить, что число файлов данных с Oracle7 ограничено несколькими сотнями, в Oracle8 — несколькими тысячами. Проблема, которая требует одного часа администратора базы данных в БД размером 5GB с 20 файлами данных, для аналогичного решения в VLDB размером в 1000GB и 600 файлами потребует более 30 часов. Комбинация возрастания размеров проблем и более жестких требований доступности в среде VLDB погубит администратора базы данных если он не сможет реализовать стандартные решения, подобные тем, что мы обсуждаем.

В табличных пространствах, содержащих сегменты малого размера, возможно Вам потребуется иметь несколько размеров сегментов (для того, чтобы сберечь пространство, уменьшить число файлов данных). Если Вы вынуждены использовать несколько размеров сегментов в табличном пространстве используйте такие размеры, чтобы все они были кратные, либо делили друг друга нацело, для возможности повторного использования пространства.

### 6.3 Выбор параметров хранения

Процедура, описанная ниже, поможет Вам выбрать значения параметров хранения, удовлетво-

Блоков Oracle	Байт	Описание
1	8,192	Размер блока Oracle
262,144	2,147,483,648	размер файла ОС
2		ufs=1, raw=2
262,142	2,147,467,264	размер полезного пространства

ряющих ограничениям описанным ранее, для Вашей VLDB.

1. Вычислите размер используемого пространства, доступного в Ваших файлах данных (использование стандартных размеров для файлов данных, как рекомендовалось ранее, упростит эту задачу). Самый простой способ сделать это — использовать запрос к представлению словаря данных **dba\_free\_space** сразу же после создания файла данных.

```
select
  blocks, bytes
from
  dba_free_space
where
  file_name = 'filename';
```

Вы можете заранее узнать размер полезного пространства в файлах данных с помощью таблицы, подобной той, что приведена ниже.

Если Вы используете подобный подход, то убедитесь, что Ваши предсказания верны с помощью запроса к **dba\_free\_space** приведенного выше.

2. Для каждого табличного пространства установите умалчиваемые параметры хранения в следующие значения:
  - Выберите значение **initial** таким, чтобы оно было кратным блоку данных БД и чтобы размер полезного пространства файла данных был кратен этому значению.
  - Установите значение **next** равным **initial**.

$n$	$k = 4^n$	Сегмент, блоки Oracle	<b>initial, next</b>
*	131,071	2	16,384
8	32,768	7	57,344
7	16,384	15	122,880
6	4,096	63	516,096
5	1,024	255	2,088,960
4	256	1,023	8,380,416
3	64	4,095	33,546,240
2	16	16,383	134,209,536
1	4	65,535	536,862,720
0	1	262,142	2,147,467,264

$n$	$k = 4^n$	Сегмент, блоки Oracle	<b>initial, next</b>
*	131,071	2	16,384
4	10,000	26	212,992
3	1,000	262	2,146,304
2	100	2,621	21,471,232
1	10	26,214	214,745,088
0	1	262,142	2,147,467,264

- Установите **pctincrease** в **0**.

Таблицы, приведенные ниже, показывают два различных набора параметров хранения, которые подходят под эти ограничения. Каждый набор содержит экстенд из двух блоков, используемый для очень малых сегментов, оставшиеся размеры параметров хранения рассчитываются таким образом, чтобы каждый разбивал полезное пространство файла данных на  $k$  равных частей. Различие между двумя наборами заключается в выборе числового ряда для  $k$ . В первом случае ряд значений — это степень 4, во втором — степень 10.

3. Везде, где это возможно, используйте умалчиваемые, определенные на уровне табличного пространства, параметры хранения для сегментов. Оптимальным вариантом было бы наличие только одного набора значений **initial**, **next** и **pctincrease** для каждого табличного пространства. Для сегментов с индивидуальными параметрами хранения, отличными от умалчиваемых, используйте следующие правила:

- Установите значение **initial** таким, что-

бы оно было кратным размеру блока данных СУБД и чтобы размер полезного пространства в файле данных был кратен этому значению.

- Установите значение **next** равным **initial**.
- Установите значение **pctincrease** в **0** для больших сегментов и в **0** или **100** для малых сегментов.

4. Для каждого сегмента с параметрами хранения отличными от вышеприведенных, используйте индивидуальные значения.

Если Вы используете параметры хранения из таблицы с  $k = 10^n$  и размером файла данных показанным ранее, то Вы найдете только шесть различных наборов параметров хранения при следующем запросе:

```
select distinct (
    initial_extent||','||
    next_extent||','||
    pct_increase
) "initial,next,pctincrease"
from dba_segments;
```

```
initial,next,pctincrease
-----
16384,16384,0
212992,212992,0
2146304,2146304,0
21471232,21471232,0
214745088,214745088,0
2147467264,2147467264,0
```

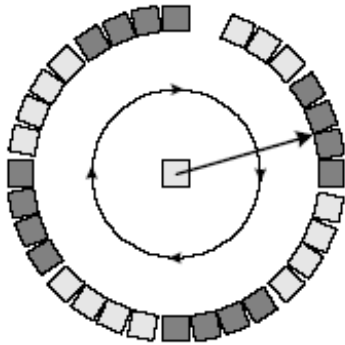
## 7 Сегменты отката

Сегменты отката — это структуры, данных которые сервер Oracle использует для предоставления следующих услуг:

- *Непротиворечивость чтения* — сегменты отката содержат информацию отката (undo) транзакций. Сервер Oracle использует информацию отката для поддержания непротиворечивости чтения запросов, использующих блоки, модифицированные (другими транзакциями) после начала транзакции.

- *Автоматический откат транзакции* — эта же информация отката используется для поддержания целостности данных после краха СУБД, а также при требовании пользователя отменить изменения незафиксированной транзакции.

Сегмент отката — это циклическая очередь, состоящая из блоков данных Oracle, в которые процессы сервера Oracle записывают информацию отката в течение выполнения транзакции. Рисунок ниже демонстрирует сегмент отката с 8 экстендами, каждый из которых содержит 4 блока данных Oracle. Блок, нарисованный в центре — это заголовок сегмента отката. Стрелка, указывающая на начало третьего блока во втором экстенде определяет место, в которое будет производиться очередная запись в сегмент отката. Она называется «указатель записи» в сегменте отката и движется по часовой стрелке повторно используя блоки сегмента отката после завершения круга.



## 7.1 Размер сегмента отката

Ключевыми ограничениями для правильного выбора размера сегментов отката являются:

- *Достаточно малый для кэширования* — блоки сегментов отката загружаются и выгружаются в SGA по тем же правилам (LRU), что и другие блоки базы данных. Большие размеры сегментов отката заполняют большую часть кэша блоков данных и, тем самым, вытесняют другие блоки данных (например, блоки данных ветвей индексов), которые улучшали бы производительность приложения.
- *Возросшее число неуспехов в кэше блоков данных БД, вызванное большим размером сегментов отката*, не только снизит качество кэша, но резко увеличивает загрузку ЦПУ и подсистемы ввода/вывода.
- *Достаточно большой размер для больших транзакций* — если Ваше приложение использует длительные транзакции, генерирующие сотни килобайт информации отката без фиксации, то Вы должны иметь, как минимум, один сегмент отката с таким размером, чтобы хранить всю информацию отката, создаваемую самой длительной Вашей транзакцией. Иначе, транзакция может завершиться с ошибкой.

Oracle7 имеет несколько возможностей, которые резко снижают сложность администрирования сегментов отката по сравнению с тем, что мы имели в Oracle6: (1) сегменты отката могут расти и сжиматься, и (2) имеется возможность перевода сегментов отката в автономный или оперативный режимы без необходимости останова инстанции. Даже с учетом этих возможностей администрирование сегментов отката остается сложной задачей в смешанных средах с OLTP и пакетной обработкой. Ключевыми средствами, снижающими негативное влияние сегментов отката на производительность, являются:

- *Проектирование приложения* — проектируйте Ваши приложения так, чтобы большие объемы DML (особенно вставки) выполнялись с `unrecoverable`-опцией везде, где это допустимо. Для транзакций, которые должны выполняться в обычном режиме, проектируйте приложение таким образом, чтобы оно вызывало частые промежуточные фиксации. Если Вы не можете выполнить этого, то как минимум, для хранения информации отката, выбирайте упомянутый ранее большой сегмент отката с помощью команды **set transaction use rollback segment**.
- *Планирование заданий* — если Вы вынуждены использовать пакетные программы, выполняющие большой объем DML-операций без промежуточных фиксаций (например, предустановленные программы, которые Вы

не имеете возможности настраивать), то старайтесь планировать такие задания таким образом, чтобы они не конкурировали с интенсивными параллельными процессами работающими с SGA. Для снижения накладных расходов по поддержанию непротиворечивости чтения, а также для предотвращения ошибок «*snapshot too old*», Вы также должны стараться разделить по времени запуск таких программ и программ читающих изменяемые данные.

Вы можете использовать следующий метод для определения оптимального размера сегментов отката для Вашей системы.

1. Убедитесь, что все оперативные сегменты отката имеют одинаковый размер. Вы можете держать несколько сегментов отката специального размера в автономном режиме для использования во временном окне, выделенном для пакетной обработки. Если Вы используете такую схему, то в начале окна пакетной обработки, Вы должны перевести небольшие OLTP-сегменты отката в автономный режим и перевести сегменты отката предназначенные для пакетной обработки в оперативный режим. После окончания окна пакетной обработки Вам необходимо выполнить обратные действия для подготовки БД к OLTP-нагрузке.
2. Выберите размер сегмента отката таким, чтобы он был достаточно большим для хранения информации отката некоторого небольшого числа  $k$  одновременно запущенных наибольших транзакций, которые Вы будете использовать в момент активности этого сегмента отката. Критерий для выбора значения  $k$  заключается в минимизации числа блоков в кэше блоков данных БД, выделенных для хранения сегмента отката. Если наибольшая транзакция генерирует менее чем один блок БД информации отката, то  $k = 4, 5, 6 \dots, 10$ . Если наибольшая транзакция порождает сотни килобайт информации отката, то установите  $k < 4$  и побеседуйте с людьми, которые разработали приложение.
3. Установите значение **minextents** в диапазоне от **8** до **20** для каждого сегмента отката, а

также значение **optimal** в значение, гарантирующее, что сегмент отката не будет сокращаться менее чем до 8 (до 20) экстенгов. Обратите внимание, что большое значение **minextents** предполагает небольшое значение для размера экстенга сегмента отката. Использование нескольких экстенгов в сегмента отката снижает частоту появления событий роста и сжатия [5, Millsap (1995b)].

## 7.2 Количество сегментов отката

Ограничения на количество сегментов отката подобны ограничениям на размер сегментов отката:

- *Достаточно малое, чтобы кэшироваться* — Наличие слишком большого числа сегментов отката имеет тот же болезненный эффект на качество кэша в SGA и обработку контрольной точки, что и слишком большие размеры сегментов отката.
- *Достаточно большое, чтобы избежать конкуренции* — Наличие слишком малого числа сегментов отката будет причиной конкуренции за *транзакционную таблицу (transaction table)* сервера Oracle, которая хранится в заголовке сегмента отката. Определить наличие конкуренции за сегменты отката Вы можете с помощью показателя *undo header waits* из динамической таблицы производительности **v\$waitstat**.

Таким образом, Вам необходимо создать достаточное количество сегментов отката для того, чтобы избежать возникновения событий ожидания на заголовках сегментов отката, но не более чем максимальное число одновременных активных транзакций. Если Вы не имеете фактических измерений для Вашей системы, то Вы можете оценить необходимое число сегментов отката с помощью следующего процесса:

1. Выберите коэффициент достоверности  $C$ , где  $0 < C < 1$ . К примеру, если Вы хотите получить 90% оценку, то  $C = 0.90$ .
2. Оцените максимальное число активных пользователей в момент пиковой загрузки системы. Обозначим это число пользователей как  $n$ .

3. Оцените вероятность того, произвольно взятая транзакция будет активной в случайно заданный момент. Обозначим эту вероятность как  $p$ .
4. Найдите наименьшее значение  $x$  для которого

$$P(X \leq x) > C,$$

где

$$P(X \leq x) = \sum_{k=1}^n P(X = k),$$

и  $P(X = x)$  есть функция биномиального распределения Бернулли, определенная как

$$P(X = x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}.$$

Если Вы имеете Excel или подобный инструмент, то можете найти значение  $x$  очень просто, как показано на таблице 4.

5. После некоторого времени эксплуатации Вы можете выполнить тонкую настройку сегментов отката на основе наблюдений за таблицей **v\$waitstat**. Если присутствует событие ожидания *undo header waits*, то попытайтесь решить проблему с помощью увеличения числа сегментов отката. Если событий ожидания заголовка сегмента отката нет, Вы можете удалить один или несколько сегментов отката, не оказывая влияния на производительность системы.

$n = 170, p = 0.05$		
$x$	$P(X = x)$	$P(X \leq x)$
0	0.000	0.000
1	0.001	0.002
2	0.006	0.008
3	0.019	0.027
4	0.042	0.069
5	0.074	0.143
6	0.106	0.249
7	0.131	0.381
8	0.141	0.521
9	0.133	0.655
10	0.113	0.768
11	0.086	0.854
12	0.060	0.915
13	0.039	0.953
14	0.023	0.976
15	0.012	0.988
16	0.006	0.995
17	0.003	0.998
18	0.001	0.999
19	0.001	1.000
Итого	1.000	

Таблица 4. Оценка требуемого числа сегментов отката для OLTP. В данном примере, если мы имеем 170 одновременно работающих пользователей, каждый из которых запускает 3-секундную транзакцию раз в минуту, наличие 12 сегментов отката будет обеспечивать транзакцию сегментом отката в 90% времени. Таблица создана в Excel с помощью функции **binomdist**.

## 8 Заключение

Для успешной реализации очень больших баз данных с помощью сервера Oracle, Вам необходимо знать требования и технологии. В этой статье я идентифицировал несколько проблем конфигурирования из-за которых система не сможет достичь ожиданий, возложенных на нее. В то же время я пытался структурировать проблему, что позволило явно определить компромиссы которые, в конечном счете, определяют, что Вам действительно нужно для достижения Ваших требований.

Не существует единой конфигурации для VLDB, поскольку у разных компаний, эксплуатирующих разные системы, всегда имеются различия в целях и приоритетах. Цель, которую я пытался достичь, заключается в идентификации наиболее важных компромиссных ограничений, присущих архитектуре сервера Oracle и попытке донести до читателя точную информацию, которая поможет сделать решения в поиске его совершенной системы,

## Благодарности

Я искренне благодарю друзей за потраченное ими время и полезные комментарии: Dominic Delmolino, Greg Doherty, Tim Gorman, Todd Guay, Deepak Gupta, Gary Hallmark, Andrew Holdsworth, Phil Joel, Anjo Kolk, Mark Pavkovic, Richard Powell, Lyn Pratt, Willis Ranney, Craig Shallahamer, Hank Tullis, Hugh Ujhazy, Peter Utzig, Mitch Wallace, и Graham Wood.

## Список литературы

- [1] CHEN, P.; LEE, E.; GIBSON, G.; KATZ, R.; PATTERSON, D. 1994. «RAID: high-performance, reliable secondary storage» in *ACM Computing Surveys*, Vol. 26 No. 2 (Jun 1994).
- [2] GUI, JEFFREY. 1993. «OLTP and System Reliability» in *OLTP Handbook*, edited by Gary McClain, Intertext/McGraw-Hill, New York NY.
- [3] MAULIK, B.; PATKAR, S. 1995. «Outage recovery timings» in *Technical Reports Compendium* Vol. I (Dec 1995). Oracle internal document.
- [4] MILLSAP, C. 1995a. «*The OFA Standard-Oracle7 for Open Systems*». Oracle internal document, available on-line at <http://www.europa.com/~orapub/>.
- [5] MILLSAP, C. 1995b. «*Oracle7 Server Space Management*». Oracle internal document, available on-line at <http://www.europa.com/~orapub/>.
- [6] MILLSAP, C. 1996. *Selecting the Optimal Oracle Database Block Size*. Oracle internal document. Not yet available on-line.
- [7] *Oracle7 Server Administrator's Guide*. 1996. Oracle standard product documentation, Redwood Shores CA.
- [8] *Oracle7 Server Concepts Manual*. 1996. Oracle standard product documentation, Redwood Shores CA.
- [9] PATTERSON, D.; GIBSON, G.; KATZ, R. 1988. «A case for redundant arrays of inexpensive disks (RAID)» in *International Conference on Management of Data (SIGMOD)*. ACM, New York: 109-116.
- [10] TO, L. 1995. «Outage prevention, detection, and repair» in *Technical Reports Compendium* Vol. I (Dec 1995). Oracle internal document.
- [11] *Understanding Disk Arrays*. 1995. Sun Microsystems white paper, Mountain View CA.

### **Замечания к переводу**

Перевод выполнен Михаилом Прусовым (<http://mprusov.narod.ru/>). По всем вопросам и предложениям пишите по адресу <mailto:mprusov@yandex.ru>.

Приведенная в библиографии ссылка на <http://www.europa.com/~orapub/>, по всей видимости, уже не действительны. Указанные документы можно найти либо на сайте компании *Hotsos* <http://www.hotsos.com/>, либо на сайте <http://www.orapub.com/>. В обоих случаях, Вам необходимо пройти бесплатную регистрацию.