OpenEdge® RDBMS
Настройка производительности – это просто!

EXCHANGE 2006 EDITION, 4 июня 2006 г.

Гас Бьорклунд, WIZARD,
PROGRESS SOFTWARE CORPORATION

PROGRESS
SOFTWARE

OpenEdge® RDBMS
Настройка производительности – это просто
Гас Бьорклунд

Оглавление

1. Введение

Данная монография является кратким руководством по методам оптимизации производительности OpenEdge RDBMS. Монография не представляет собой исчерпывающего исследования предмета. Ее цель – дать полезные советы и методы. Многие из этих рекомендаций очень просто реализовать на практике и они дают значительный эффект, не требуя при этом большого объема работ и размышлений[1]. Некоторые рекомендации намного проще реализовать при создании новой базы данных, чем для настройки имеющейся. Например, если существующая база имеет большой объем, то перестройка схемы хранения может потребовать неприемлемо много времени.

Чтобы показать простоту настройки производительности, мы использовали многопользовательские тесты производительности с высокой транзакционной активностью. При использовании "готовой" конфигурации OpenEdge 9.1D без настройки параметров, производительность составляла около 30 транзакций в секунду. Применив всего восемь приведенных здесь рекомендаций, мы получили увеличение производительности более чем на порядок – почти 600 транзакций в секунду![2]

Поскольку каждая база данных, компьютер, конфигурация операционной системы, приложение и его нагрузка имеет свои различия, не существует универсальной конфигурации базы данных, работающей оптимальным образом, или даже одинаково хорошо, для каждого случая. В то время как некоторые рекомендации, сделанные в данной монографии, будут полезны для большинства систем, для вашей системы они могут оказаться и непригодными. Поэтому, при их выполнении обязательно нужно провести измерения до внесения изменений и после, чтобы быть уверенным, что вы получили тот ожидаемый эффект. Не забывайте старое правило «семь раз отмерь, потом отрежь», измерения следует проводить также и после внесения изменений.

В следующих разделах в качестве имени базы данных используется "foo". Разумеется, вам везде нужно использовать имя вашей базы данных. Если вы обнаружите какие-либо ошибки или у вас есть предложения по улучшению данной монографии, пожалуйста, пишите по адресу gus@progress.com.

[1]Как сказал Тэд Уильямс (Ted Williams), "если Вы думаете не слишком хорошо, не думайте слишком много". Даже если Вы "думаете хорошо", размышления - это работа, а ненужной работы всегда следует избегать.

[2] Ваши показатели, разумеется, могут отличаться от наших значений.

2. Измерение производительности

Чтобы узнать, привели ли ваши усилия по оптимизации производительности к требуемым результатам и даже необходимы ли они вообще, следует иметь средства измерения эффекта/влияния изменений на производительность вашей системы.

Показатели, которые вы используете, должны быть конкретными (specific) и воспроизводимыми. Например, можно измерять промежуток времени, потребовавшийся при выполнении приложением определенной функции – добавление нового клиента, запуск задачи по закрытию месяца или генерации определенного отчета. Имеются разнообразные средства, позволяющие исследовать систему, а также таймеры и счетчики активности базы данных, дающие рекомендации, какие определенные аспекты, вероятно, требуют улучшения.

Вот краткий список широко доступных средств:

• Программа promon OpenEdge RDBMS, показывающая много информации о статусе, деятельности, скорости ввода-вывода, транзакциях базы данных и т.д., более чем на 50 различных экранах. promon входит в состав OpenEdge RDBMS. Она соединяется с локальными базами данных через область разделяемой памяти.

• Virtual System Tables (виртуальные системные таблицы) это ”магические” таблицы, имеющиеся в каждом экземпляре базы данных, которые можно попросить собирать информацию, аналогичную той, что показывает программа promon. Затратив немного усилий, можно написать маленькие программы на 4GL для мониторинга конкретных интересующих вас значений. Получить доступ к VST можно удаленно из приложений на базе 4GL или SQL.

• Файл журнала базы данных OpenEdge RDBMS (файл ”.lg”) записывает разнообразную информацию при запуске и останове экземпляра базы данных, при подключении и отключении пользователей и приложений, а также о генерируемых при работе системы сообщений об ошибках. Один из наборов информации может оказаться особенно полезным: при запуске экземпляра базы данных в файл журнала записывается большинство параметров конфигурации. Это может быть исключительно полезно при диагностировании проблем, связанных с производительностью и прочих.

• OpenEdge Management (первоначально Fathom Management), это консоль для мониторинга и управления системой, которая дает информацию об одном или нескольких экземплярах баз данных, ведет активный мониторинг их деятельности и генерирует предупреждения, когда параметры выходят за установленные пределы. В отличие от других средств, OpenEdge Management может также запоминать прошлые данные, поэтому можно исследовать тенденции в собираемых данных. OpenEdge Management предлагается Progress Software Corporation.

• ProTop это прекрасное бесплатное средство мониторинга, написанное на 4GL Томом Баскомом (Tom Bascom) из Greenfield Technologies. Для получения показываемых данных используются Virtual System Tables. ProTop ведет мониторинг базы данных, представляя данные подобно имеющейся в UNIX программе ”top” – отсюда и название. ProTop можно найти в интернете по адресу http://www.greenfieldtech.com.

• Pro Monitor, предлагаемый BravePoint, Inc., это еще одно великолепное средство мониторинга, написанное на 4GL. Оно также читает данные из Virtual System Tables. Его предназначение – дать простой вид реального времени вашей

системы Progress OpenEdge. Статистика производительности записывается ежедневно в формате, простом для чтения и понимания. Программу можно найти в интернете по адресу http://bravepoint.com/products.

• Адам Бэкмен (Adam Backman) из White Star Software написал набор полезных скриптов для администраторов баз данных UNIX. Их можно найти в Интернете по адресу http://peg.com/utilities.

• Операционные системы поставляются с различными средствами мониторинга: top, sar, iostat, vmstat, pstat, ps, perfmon и т.д., которые предоставляют информацию о параметрах функционирования операционной системы и о процессах, запущенных на ней. Хотя многие из этих средств доступны в большинстве операционных систем, набор средств у каждой системы различается. Имеются и средства, специфические для отдельных систем (например, Glance для HP), которые отсутствуют в других. Чтобы выяснить, какие средства вам доступны, обратитесь к документации.

• На Linux'е массу полезной информации можно найти в файловой системе /proc.

• Помимо других средств, Solaris предлагает команды pmap, proc и truss.

• Другой источник полезной информации – журналы операционной системы. Какая информация в них имеется, зависит от операционной системы, но их всегда следует использовать при диагностировании ошибок.

• И еще один источник информации, далеко не последний по значению – независимая группа Progress Email Group, организованная неутомимым Грегом Хиггинсом (Greg Higgins), расположенная в интернете по адресу http://www.peg.com. В этой группе PEG активно ведутся оживленные дискуссии по всем вопросам, связанным с Progress. Здесь можно многому научиться, задать вопросы и получить помощь в решении проблем.

Никакое из вышеперечисленных средств не скажет вам о некоторых важнейших вещах:

• Довольны ли ваши пользователи?

• Имеются ли такие функции, о которых пользователи думают, что они выполняются слишком долго?

• Бывают ли ситуации, когда пользователи не получают приемлемого времени отклика?

Вышеперечисленные средства помогут вам понять, как следует решать определенные вопросы, но в качестве одного из средств мониторинга следует обязательно использовать мнение ваших пользователей. Другой полезный метод для измерения производительности – настроить ваше приложение для сбора данных о работе и затратах времени для тех операций, которые, как вы считаете, имеет смысл измерять.

3. Регулярно делайте резервные копии!

Если произойдет системный сбой (а это, в конце концов, случится), а у вас для восстановления нет недавней резервной копии базы, то производительность вашей базы данных станет равной нулю. И она будет оставаться равной нулю в течение длительного времени, пока Вы восстанавливаете свою базу по кусочкам или заново вводить данные вручную.

После того как вы выработаете разумные и правильные процедуры резервного копирования, можно приступать к процедурам повышения производительности.

4. Простые средства оптимизации

В данном разделе содержатся рекомендации, которые легко может выполнить каждый. Их можно быстро сделать за несколько минут, когда база остановлена. Однако следует понимать, что самые простые средства не обязательно будут оптимальными – но они существенно лучше настроек по умолчанию и должны приблизить вас к оптимальным значениям. Заметим, что под значениями параметров конфигурации по умолчанию, о которых говорится в данной монографии, понимаются значения для Progress версии 9.1. Более поздние версии могут иметь другие значения параметров по умолчанию.

4.1. Увеличьте размер пула буферов (Buffer Pool Size)

Самый эффективный способ увеличения производительности – выделить базе данных больше памяти для работы, начиная с размера пула буферов. Пул буферов базы данных – это область памяти, предназначенная для ускорения многих операций базы данных. Ускорение работы происходит благодаря снижению числа операций чтения с диска за счет сохранения в памяти копий недавно использованных блоков данных. Другой фактор ускорения работы – снижение числа операций записи, т.к. блоки данных в памяти могут многократно обновляться перед тем, как, в конечном счете, будут записаны на диск. Доступ к блокам, которые уже находятся в памяти, осуществляется намного быстрее, чем при считывании с диска. Пул буферов базы данных обычно занимает самую большую часть области совместно используемой памяти. Увеличение размера этой области обычно приводит к повышению производительности.

Число буферов в пуле базы данных устанавливается параметром -B. Размер буфера равен размеру блока данных базы данных.

Размер по умолчанию достаточно мал и очень далек от оптимального. Для однопользовательского режима, по умолчанию устанавливается 10 буферов, а для многопользовательского – в 8 раз больше, чем -n. Если –n по умолчанию равно 20, то число буферов по умолчанию будет равно 160. Хотя оптимальное значение зависит от конкретной базы данных и нагрузки приложений, для начала в качестве двух весьма приблизительных значений можно использовать:

• 10 % от размера базы данных
• от 2 до 4 МБ на пользователя

В программе promon можно найти значение параметра ”Buffer Pool Hit Ratio” показываемое на нескольких экранах. Это значение говорит о проценте операций доступа к базе данных, которые были выполнены с использованием данных, уже находящихся в памяти, когда не требовалось читать их с диска. Чем больше это значение, тем лучше. Если значение равно 98 %, это означает, что только 1 из каждых 50 операций доступа к базе данных требует чтения с диска. Если значение ниже 95 % и производительность неудовлетворительная, определенно следует увеличить размер пула буферов. Следует добиться значения не менее 98 %, если это возможно. Упоминавшаяся ранее программа ProTop, разработанная Томом Баскомом, имеет guesstimator, который может оказаться полезным для выбора размера пула буферов.

Вы можете увеличивать размер пула буферов до тех пор, пока у вас в системе имеется свободная память. Но имеются и ограничения: если вы сделаете его слишком большим, система может начать свопинг, который может привести к резкому падению производительности системы в целом. Многие системы имеют достаточно памяти, поэтому можно указать для пула буферов 100,000 или более

буферов. Заметим, что для базы данных с размером блока данных 8192 байт, 100,000 буферов потребуют около 900 мегабайт памяти.

Для 64-битных версий Progress OpenEdge, если система имеет достаточно физической памяти, можно установить общий размер области совместно используемой памяти вплоть до 116 гигабайт.

Для 32-битных версий Progress OpenEdge общий размер совместно используемой памяти не может превышать примерно 2 ГБ. Точное значение предела зависит от операционной системы, но для большинства систем, в идеальных условиях 32-битные программы могут использовать примерно 2 ГБ[3] памяти. Реальный лимит зачастую будет меньше, в зависимости от многих факторов[4].

Если ваше приложение соединяется со многими базами данных в режиме самообслуживания (self-serving mode), общее пространство, занимаемое областями совместно используемой памяти для всех баз данных, с которыми вы устанавливаете соединения одновременно, в сумме не должно быть больше указанного ранее предела. Кроме того, вы должны иметь достаточно памяти для хранения и других ресурсов, например, дескрипторов файлов для всех файлов, связанных с базами данных, и семафоров, используемых для взаимодействия процессов.

Чем больше число баз данных, с которыми ваше приложение устанавливает соединения одновременно, тем меньше для каждой из них должна быть область совместно используемой памяти, для того чтобы все они поместились в адресное пространство, доступное для приложений, использующих соединения в режиме самообслуживания.

4.2. Увеличьте число буферов для журнала Before-Image

Для кэширования в памяти информации о транзакциях используется свой набор буферов, которые впоследствии сбрасываются на диск в журнал before-image. Каждое изменение состояния транзакции и изменение состояния базы данных записывается в виде одной или более записей журнала транзакций, которые сначала помещаются в текущий буфер журнала before-image, а затем записываются в журнал before-image. Обычно это делает процессом записи BIW. Наличие в памяти достаточного количества буферов дает этому процессу необходимое время для выполнения записи. Когда текущий буфер журнала полон, а свободных пустых буферов больше нет, то транзакция, которая хочет записать данные в буфер журнала, должна ждать, пока не станет доступен пустой буфер. Увеличивая число буферов, мы снижаем вероятность ожидания при выполнении изменения базы данных.

По умолчанию число буферов журнала before-image равно 5 и устанавливается параметром -bibufs. Следует увеличить это значение примерно до 25. Заметим, что этот параметр не является ”регулятором скорости”. Необходимо просто иметь достаточное число буферов для сглаживания временных изменений в количестве данных журнала, которые следует записать. В программе promon можно увидеть число ожиданий пустых буферов для журнала before-image. Если значение превышает несколько единиц в секунду, следует увеличить число буферов. Однако если диск, на котором расположен журнал before-image, перегружен и не в состоянии своевременно выполнить дополнительные операции записи, то увеличение числа буферов не даст эффекта.

[3] на последних версиях Linux, максимум равен 2,7 ГБ

[4] рассмотрение которых выходит за рамки данной монографии

Если Вы не используете Before-Image Writer, увеличение числа буферов даст незначительный эффект или не даст его вообще.

4.3. Увеличьте число буферов для журнала After-Image

При включении after-image журналирования, также как и для before-image файла используется свой набор буферов для кэширования записи на диск в файл after-image. Каждая запись журнала транзакций, записываемая в журнал before-image[5] будет записываться также в журнал after-image. Обычно этим занимается процесс записи AIW. Как и для буферов журнала before-image, когда текущий буфер журнала after-image полон, а пустых буферов журнала after-image не имеется, транзакция, которая хочет записать данные в буфер журнала, должна ждать, пока не станет доступен пустой буфер after-image. Увеличивая число буферов, мы снижаем вероятность ожидания при выполнении изменения базы данных. Следует установить число буферов журнала after-image немного больше, чем число буферов журнала before-image, в большинстве случаев на 25-50 % выше. (Примечание комментатора – данная рекомендация не вполне корректна и лучше задавать значения параметров –bibufs и –aibufs равными друг другу.)

Если вы не запускаете процесс AIW, то увеличение числа буферов даст незначительный эффект или не даст его вообще.

Если Вы не используете after-image журналирования, увеличение числа буферов не даст никакого положительного эффекта, а приведет лишь к пустой трате памяти.

4.4. Использование параметра –spin

База данных может использовать механизм spinlock'ов как метод синхронизации деятельности процессов, осуществляющих одновременный доступ к данным, хранящимся в совместно используемой области разделяемой памяти, например, к пулу буферов базы. Если система, управляющая вашей базой данных, имеет более одного процессора, для параметра -spin следует установить ненулевое значение.

Поиск оптимального значения для параметра spin требует значительного времени, а выбор хорошего значения – нет: просто используйте значение 50,000. (Примечание комментатора – тесты показывают, что оптимальное значение этого параметра лежит в диапазоне 10,000 – 30,000.) Для современных систем, мы предлагаем в качестве приблизительного начального значения использовать 20,000, но проще запомнить значение 50,000 и это значение оказывается вполне достаточным. В большинстве случаев, определение точных лучших значений не столь важно и не стоит затрат вашего времени. В наших тестах со 150 пользователями, мы обнаружили очень незначительную разницу в производительности при изменении значения -spin от 10,000 до 90,000, при этом разница между 2,000 и 40,000 составляла около 50% (436 и 680 транзакций в секунду соответственно). Обычно, чем меньше число пользователей, осуществляющих доступ к базе данных, тем меньше сказывается разница в настройке параметра -spin.

При условии, что вы запустили свою базу с ненулевым значением параметра –spin, его можно изменить прямо во время работы базы, используя утилиту promon. Это позволит определить более удачное значение этого параметра.

[5] В действительности, имеется одно или два исключения, но не стоит о них беспокоиться

Сделав это, не забудьте обновить конфигурационный файл или скрипт запуска базы.

Мы видели большую систему со многими процессорами, для которой прекрасно работало значение 2,000,000, но это довольно нетипично.

Заметим, что лицензия Workgroup RDBMS не поддерживает использования параметра -spin.

4.5. Увеличьте размер кластера журнала Before-Image

В многопользовательском режиме база данных выполняет непрерывный процесс синхронизации содержимого разделяемой памяти с состоянием базы на диске, в рамках которого в фоновом режиме измененные блоки данных пишутся на диск. Новый цикл синхронизации начинается с открытием очередного кластера журнала before-image. Увеличение размера кластера журнала before-image приводит к увеличению интервала между контрольными точками, что дает процессам записи APWs необходимое время для их работы. Продолжительность последних 8 проверок показывается на экране Checkpoints программы promon. В идеальном состоянии интервал между проверками должен составлять около минуты или более, а число буферов, сброшенных на диск в контрольной точке, должно быть равным или близко к нулю. Более длительные интервалы возможны, но не нужны. Если они короче минуты, то следует увеличить размер кластера. Если число buffers flushed больше 10, то следует увеличить размер кластера, или увеличить число фоновых процессов APW, или повысить производительность дисков по записи.
Размер before-image кластера можно установить следующей командой:

```
proutil foo -C truncate bi -bi 4096
```

Размер кластера указывается в килобайтах. В качестве стартового значения можно использовать 4 МБ, но для баз данных с высоким уровнем транзакционной активности может потребоваться больший размер кластера. База запомнит значение размера кластера, которое вы ей задали.

При работе с лицензий Workgroup RDBMS размер кластера следует оставить небольшим. Значение 512 КБ или менее будет лучше, чем кластеры большего размера. Поскольку с этой лицензией Вы не можете запускать фоновые процессы, то все модифицированные блоки данных будут писаться на диск именно в контрольных точках. Чем больше размер кластера, тем больше измененных блоков накопится к этому моменту . Запись этих блоков на диск будет приводить к периодическому “подвисанию” базы.

4.6. Установите размер блока журнала Before-Image

Увеличение размера блока, используемого для работы с журналом before-image, увеличивает производительность. В UNIX-системах (Solaris, AIX, HP-UX, Tru64, UnixWare и т.д.) следует использовать значение 8КБ. На Linux-системах следует использовать значение 4КБ. Под Windows следует использовать значение 4КБ до тех пор, если вы не увеличили размер кластера NTFS, как указано в разделе, посвященном специфическим вопросам Windows.

Размер блока before-image можно установить в программе proutil следующей командой:

proutil foo -C truncate bi -biblocksize 8

Размер блока указывается в килобайтах.
(Примечание комментатора – размер before-image блока можно задать равным 8 или 16 КБ на всех платформах)

4.7. Установите размер блока журнала After-Image

Увеличение размера блока для журнала after-image увеличивает эффективность его работы. Желательно иметь размер after-image блока в точности равным размеру before-image блока. В UNIX-системах (Solaris, AIX, HP-UX, Tru64, UnixWare и т.д.), следует использовать 8КБ. На Linux-системах следует использовать 4КБ. Под Windows, следует использовать 4КБ до тех пор, пока вы не увеличите размер кластера NTFS как указано в разделе, посвященном специфическим вопросам Windows. Перед изменением размера блока after-image следует выключить журнал after-image (если он был включен) и обнулить журнал before-image:
rfutil foo -C aimage end
proutil foo -C truncate bi

Размер блока after-image можно установить следующей командой:

rfutil foo -C aimage truncate -aiblocksize 8

Размер блока указывается в килобайтах.

4.8. Используйте Before-Image Writer (BIW)

Before-image writer - это процесс, предназначенный для записи на диск только что заполненных буферов журнала before-image. Если этот процесс запущен, то серверу базы данных эти действия не нужно выполнять, что дает ему больше времени для полезной работы. При использовании самообслуживающихся клиентов, сервер и клиент будет находиться в равных условиях, но по-прежнему будете желательным, чтобы серверная часть была занята своей непосредственной работой.

Заметим, что лицензия Workgroup RDBMS не позволяет запускать before-image writer.

Запустить before-image writer можно следующей командой:

probiw foo

Можно запустить только один before-image writer.

4.9. Используйте After-Image Writer (AIW)

Если вы используете after-image журналирование (а это следует делать, хотя и не для повышения производительности), следует запускать процесс after-image writer (AIW). Этот процесс предназначен для записи на диск только что заполненных буферов журнала after-image, для того чтобы этого не приходилось делать серверу. Если процесс запущен, то серверу базы данных эти действия не нужно выполнять, что дает ему больше времени для полезной работы. При использовании самообслуживающихся клиентов, сервер и клиент будет находиться в равных условиях, но по-прежнему будете желательным, чтобы серверная часть была занята своей непосредственной работой.

Заметим, что лицензия Workgroup RDBMS не позволяет запускать after-image writer.

Запустить after-image writer можно следующей командой:

proaiw foo

Можно запустить только один after-image writer.

4.10. Используйте асинхронные Page Writers (APWs)

Всегда следует запустить как минимум один асинхронный процесс page writer (APW).

Асинхронный page writer - это фоновый процесс, предназначенный для записи недавно измененных блоков базы данных на диск в область данных. В этом случае, серверу не приходится выполнять эту задачу, а блоки модифицированных данных не будут сбрасываться на диск все сразу в конце цикла проверки (checkpoint cycle). Ряд пунктов меню утилиты promon сообщает о подобных записях как об очистке буферов (buffers flushed). Для большинства проверок это число должно быть менее 10.

В большинстве случаев будет достаточно запуска одного или двух таких процессов. Излишнее их число обычно не приносит вреда, но слишком большое число, скажем 50, может несколько снизить производительность, поскольку увеличится число конфликтов при доступе к пулу буферов.

Заметим, что лицензия Workgroup RDBMS не позволяет запускать page writers.

Асинхронный page writer можно запустить следующей командой:

proapw foo

4.11. Избегайте однопользовательского режима Single-user Mode

Почти во всех рекомендациях, приведенных в данной главе, предполагается, что база данных работает в многопользовательском режиме. Обычно при использовании базы данных в однопользовательском режиме, ее производительность будет значительно хуже.

Когда база данных работает в однопользовательском режиме, имеется только одно соединение с базой данных, и нет области разделяемой памяти (вместо этого все структуры данных расположены в частной памяти процесса). Однопользовательский режим имеет следующие недостатки по сравнению с многопользовательским режимом:

• Отсутствуют асинхронные фоновые процессы для записи данных в области данных и в журналы транзакций. Это ограничивает производительность.
• Невозможно использовать программу promon или иные программы для мониторинга за работой базы данных.
• Невозможно использовать онлайновые утилиты, например, переключение активного after-image экстента, архивирование полных экстентов или создание копии базы на лету.

5. Оптимизация расположения данных на диске

Данный раздел посвящен оптимизации расположения ваших данных на дисковом пространстве. Для существующих баз данных перемещение или выгрузка и последующая загрузка больших объемов данных потребует длительного времени, что на практике может оказаться неприемлемым. Но при создании новой базы данных это не должно вызвать затруднений.

5.1. Используйте больше одного диска для базы данных

Один диск может одновременно осуществлять одну передачу данных. Два диска могут вести две передачи данных одновременно. Чем больше дисков вы будете использовать, тем выше может быть производительность. Скорость передачи данных не зависит от размера диска и несколько дисков меньшей емкости дадут большую производительность, чем один диск большой емкости.

5.2. Страйпинг и зеркалирование дисков (Stripe and Mirror Disks)

Для увеличения производительности, желательно иметь сбалансированную нагрузку ввода-вывода для имеющихся дисков, чтобы каждый из них вносил равный вклад в общий объем работы. Если один из дисков выполняет намного больше работы, чем другие, и он не в состоянии справиться с такой нагрузкой, то тем самым он замедлит работу всей системы.

Самый эффективный способ балансировки нагрузки между несколькими дисками – создать страйпинг, объединяющий все физические диски в один логический диск с равномерным распределением данных по нему. Использование одного только страйпинга имеет существенный недостаток: если из-за сбоя вы потеряете один из дисков, то вы потеряете данные на всех дисках. Для получения достаточной надежности, страйпинг следует объединить с зеркалированием. Вы должны использовать пары (или даже тройки) зеркальных дисков, а затем использовать страйпинг для этих пар. Это надежнее, чем наоборот, потому что подобная конфигурация более устойчива к сбоям дисков.

В тестах проведенных на Linux и Windows мы обнаружили, что чем больше размер страйпинга, тем большую производительность он обеспечивает. Максимальный размер stripe, который вы можете использовать, зависит от используемой операционной системы или дисковой подсистемы.
Мы не проводили тестирования страйпинга размером более 256 КБ, т.к. это был максимально возможный для нас размер.

5.3. Избегайте использования RAID 5

При использовании вместо страйпинга и зеркалирования дискового массива RAID-5, вы потеряете при нормальной работе около 45 % производительности[6] и еще больше – при выполнении операций обслуживания и восстановления. Потеря производительности объясняется дополнительными служебными операциями записи, заложенным в работу RAID-5 – каждый блок должен быть записан на два диска, один для данных, второй для восстановления данных. В массиве с четырьмя дисками процесс записи всегда занимает 50 % общей

[6] данные получены при тестировании 6 дисками

пропускной способности диска по записи. Потеря производительности тем меньше, чем больше число дисков в массиве. В массиве RAID-5 с 20 дисками, дополнительный объем операций записи может составлять всего 5 %.

Однако если вам потребуется заменить отказавший диск, придется считывать все данные со всех остальных дисков, пока происходит восстановление отказавшего диска. Это требует длительного времени и вызывает очень значительную потерю производительности до тех пор, пока не будет завершено восстановление работоспособности массива.

5.4. Выбор размера блока данных

Блоки большего размера обеспечивают бóльшую производительность ввода-вывода и делают хранение базы данных более эффективным. Во многих файловых системах UNIX заложен размер блока или размер страницы равным 4 или 8 КБ. Наилучшую производительность можно получить, если размер блока базы данных совпадает или кратен размеру страницы файловой системы.

Размер блока данных выбирается в момент создания новой базы, когда вы копируете пустую базу данных желаемого размера блока. После создания базы в нее нужно загрузить данные. Размер блока существующей базы данных изменить нельзя.

5.4.1. Размер блока области данных I-го типа

Если размер блока данных больше, чем размер страницы файловой системы, существует небольшая вероятность того, что запись блока данных в случае сбоя системы, например, при выключения электропитания, будет выполнена только частично. В большинстве случаев запись блока данных в две непрерывные страницы файловой системы будет выполнена за одну операцию записи на диск и обе страницы будут записаны вместе. Однако если две страницы файловой системы находятся на разных дорожках, то для их записи может потребоваться операция поиска. Если сбой питания произойдет в начале передачи первой страницы, диск может отключиться до того, как будет записана вторая страница. В этом случае на диск будет записана только половина блока данных, что приведет появлению оборванной страницы. Операция восстановления после сбоев не распознает подобную ситуацию и не сможет ее исправить.

На Linux-системах для соответствия размеру страницы файловой системы для базы данных следует использовать размер блока равный 4 КБ. Существующая архитектура виртуальной памяти на Linux'е не позволяет использовать страницы большего размера. В будущих версиях Linux это может измениться.

На Windows-системах следует использовать размер блока данных в 4 КБ или установить размер кластера NTFS 8 КБ, после чего можно использовать размер блока базы данных в 8 КБ.

5.4.2. Размер блока области данных II-го типа

Для областей данных II-го типа для блоков вычисляются контрольные суммы, которые хранятся в заголовке блока, что позволяет обнаружить оборванные страницы. Если все ваши данные находятся в областях данных II-го типа, вы можете использовать размер блока, кратный размеру страницы файловой системы. (Примечание комментатора – в текущих версиях Progress'а проверка контрольных сумм

блоков осуществляется только утилитой dbrpr. При работе базы такие проверки не проводятся. Но даже в случае обнаружения разорванной страницы восстановление потерянных данных будет невозможно.)

Для областей данных II-го типа, установите размер блока данных 8 КБ. Будущие версии OpenEdge RDBMS будут поддерживать бóльшие размеры блока данных.

5.5. Используйте области данных II-го типа

Области данных II-го типа в сравнении с областями данных I-го типа обеспечивают более высокую производительность и лучшее использование дискового пространства с более низкой степенью фрагментации записей. Большинство утилит будут работать существенно быстрее с данными, хранящимися в областях этого типа. По этим причинам следует всегда использовать области данных II-го типа.

5.6. Используйте тома данных фиксированной длины

Тома фиксированной длины форматируются во время их создания. Тома переменной длины форматируются динамически, т.к. они расширяются во время работы, когда базе требуется больше места для хранения данных. Тома фиксированной длины обеспечивают несколько более высокую производительность потому, что не требуется расширения томов во время выделения места для индекса или данных в базе. Разница в производительности может существенно различаться в зависимости от конкретных операционной и файловой систем и от того, как сильно и как часто увеличивается ваша база данных. Разница максимальна при размере блока данных меньше чем 4096 байт. Разница в производительности по чтению между томами фиксированной и переменной длины отсутствует.

5.7. Отделите журнал Before-Image от томов данных

Разместите тома с журналами before-image на других дисках, отдельно от томов данных.

Вы ведь хотите, чтобы запись журнала before-image была как можно более эффективной и не конфликтовала с другой деятельностью.

Если на одной и тоже машине работают несколько баз, то следует разместить их журналы before-image вместе с томами данных. Вы ведь используете страйпинг для экстентов с данными, не так ли?

5.8. Подсистемы дисковых массивов

На рынке имеется большое разнообразие подсистем дискового хранения, начиная от различных типов контроллеров дисков, например, предлагаемых 3Ware, до крупных систем хранения, предлагаемых EMC, IBM и другими фирмами. Большинство из них может успешно использоваться с OpenEdge RDBMS.

6. Дополнительные вопросы

Методы оптимизации, описанные в данном разделе, может быть, несколько сложнее и менее удобны для выполнения, чем те, что было описаны выше. Некоторые из них могут быть при этом нецелесообразными, если ваша база данных очень велика или если ваша система не обладает достаточными ресурсами. Выполнение работ может потребовать более длительного останова базы, чем вы можете себе позволить.

6.1. Не забывайте про приложение

Поскольку данная монография посвящена настройке баз данных, мы не станем касаться написания приложений, но это аспект, о котором не следует забывать. Правильная конфигурация базы данных может оказать существенное влияние на производительность, но правильная архитектура приложения и программирование могут оказать еще большее влияние. Например, если ваше приложение использует запросы, которые требуют сканирования таблиц, повышение скорости сканирования таблиц никогда не будет столь же эффективным, как устранение сканирования вовсе.

6.2. Создание записей

Каждая таблица в базе данных имеет шаблон записи, в которой содержатся значения по умолчанию для столбцов таблицы. При создании новых записи, они инициируются за счет создания копии шаблона записи. Это происходит каждый раз, когда в 4GL выполняется оператор CREATE. Большинство приложений присваивает значения столбцам чуть позже. После создания строки следует немедленно присвоить значения всем столбцам, и это должно быть сделано в одном операторе ASSIGN. В этом случае при размещении новой записи будет учитываться ее размер, включающий значения всех столбцов. В противном случае, если при создании строки будут заполнены только некоторые значения столбцов, а остальные примут значения по умолчанию, пространство будет зарезервировано под текущий более короткий размер строки. А позднее, при заполнении остальных значений столбцов, строка будет обновлена. Если размер строки сильно изменяется между созданием строки и ее обновлением, то в отведенном для нее блоке данных может оказаться недостаточно свободного места для хранения более длинной строки. В этом случае, строка будет разделена на два или более фрагментов, и все они будут храниться в различных блоках данных, что увеличит время чтения этой записи.

6.3. Убедитесь, что вы используете правильные индексы

Никакая настройка баз данных или ее реконфигурирование не помогут при плохо написанном приложении, поэтому не забывайте о нем. Если вам необходимо немедленно увеличить производительность любыми способами, сделайте сначала очевидное конфигурирование и настройку для базы данных, а затем вернитесь к приложению. Другой связанный с приложением фактор, который может оказывать колоссальное влияние на производительность – использование правильных индексов, способных выполнить запросы вашего приложения без необходимости полного сканирования индекса.

Вы можете определить, использует ли ваше приложение 4GL полное сканирование индекса, откомпилировав 4GL код с параметром компиляции

XREF. Компилятор сообщит обо всех обращениях к базе данных и укажет к каким таблицам осуществлялся доступ и какой индекс (индексы) при этом использовались. Когда будет выполняться полное сканирование индекса, в листинге появится обозначение WHOLE-INDEX.

Для SQL-запросов, вы можете посмотреть планы выполнения, которые генерируются для конкретных запросов приложения. Заметим, что обработка SQL-запросов зачастую выигрывает от применения дополнительных индексов, помимо тех, что обычно используются в ваших хорошо написанных приложениях 4GL. Это объясняется тем, что многие SQL-запросы являются незапланированными (ad-hoc) и их сложность и объем данных, к которым они получают доступ, меняются от раза к разу.

Во многих случаях, создание дополнительных индексов может значительно улучшить производительность SQL-запросов. Если вы выполняете SQL-запрос с параметром NO-EXECUTE, вы сможет изучить план запросов, не выполняя их в действительности.

6.4. Внешняя кромка диска работает быстрее

Жесткие диски вращаются с постоянной угловой скоростью, поэтому дорожки у внешнего края существенно длиннее, чем те, что расположены у внутреннего края. Это означает, что на внешних дорожках содержится больше данных и скорость передачи данных выше примерно на 20%. Создав разделы диска соответствующим образом и храня тома данных (data extents) на самом внешнем (удаленном от центра диска) разделе, можно дополнительно выжать из системы еще несколько процентов производительности.

6.5. Страйпинг вручную

Если контроллеры ваших дисков или операционная система не поддерживают страйпинг, можно использовать метод так называемого ручного страйпинга. Идея состоит в создании многих томов данных одинакового размера, каждый из которых будет содержать небольшой кусок всей базы данных. Затем эти тома циклически распределяются по всем дискам. Например, если у вас есть 4 жестких диска, можно создать 32 тома и поместить по 8 на каждый диск. Поместите том 1 на первый жесткий диск, том 2 на второй, том 3 на третий, том 4 на четвертый, том 5 на первый и т.д. Если ваша база данных содержит 10 ГБ данных и у вас есть 4 жестких диска, при использовании размера тома 250 МБ, вам понадобится 40 томов, по 10 на каждом диске.

Недостатками подобного ручного страйпинга являются:

• Результаты не так хороши как при использовании страйпинга средствами операционной системы, контроллера диска или подсистемы хранения.

• Если база данных состоит из многих маленьких файлов, а не из нескольких больших, вам придется использовать намного больше дескрипторов файлов и других системных ресурсов.

• По мере того как база данных увеличивается и вы добавляете новые тома, обеспечить сбалансированную нагрузку становится все труднее,

• Встроенные средства страйпинга большинства операционных систем выполнят эту работу лучше.

• Данный метод не обеспечивает возможности зеркалирования.

• С течением времени баланс будет изменяться, поскольку нагрузка и данные меняются.

6.6. Используйте 64-разрядные версии

Все предлагаемые сейчас компьютеры Sun, IBM и HP, работающие под UNIX, имеют 64-разрядные процессоры. В течение нескольких лет 32-разрядные процессоры уже не выпускаются. Эти системы могут использовать и 32-разрядные, и 64-разрядные версии OpenEdge.
Если для вашей системы имеется 64-разрядная версия OpenEdge, вам следует рассмотреть возможность ее использования вместо 32-разрядной версии. Основное преимущество замены – возможность использования для баз данных намного бóльших пулов буферов. В 32-разрядных версиях, область совместно используемой памяти не может быть больше примерно 2 ГБ, а для 64-разрядных версий она может составлять до 116 ГБ.

6.7. Заблокируйте область совместно используемой памяти в оперативной памяти

В большинстве операционных систем область совместно используемой памяти создается таким образом, что она делится на страницы механизмом страничной организации операционной системы. Если область совместно используемой памяти действительно будет делиться на страницы (из-за недостатка физической памяти), производительность базы данных будет постепенно снижаться из-за того, что доступ к буферу данных и чтение данных из базы будет вызывать дополнительные операции ввода-вывода из-за разделения области совместно используемой памяти.

Вы можете избежать подобной ситуации, заблокировав область совместно используемой памяти в оперативной памяти при помощи параметра конфигурации -pinshm.
Использование параметра -pinshm может улучшить производительность, но может и привести к ухудшению ситуации. Если у вас недостаточно физической памяти, множество экземпляров баз данных, запущенных на одном сервере, и один или более из этих экземпляров редко используется, то блокирование всех совместно используемых областей в оперативной памяти, может вызвать рост пейджинга для оставшейся виртуальной памяти, используемой приложениями и другими программами.
В системах Solaris нет необходимости использования параметра –pinshm, т.к. база данных всегда использует параметр Solaris ISM (”Initmate Shared Memory”), который также вызывает блокировку совместно используемых областей в оперативной памяти.
Параметр -pinshm недоступен в системах Windows и AIX.

6.8. О параметре -directio

Обычно OpenEdge RDBMS осуществляет операции чтения и записи в UNIX с использованием буферизованного ввода-вывода с применением системных вызовов pread()и pwrite(), в сочетании с вызовами fdatasync()(или вызовами sync() в версиях, предшествующих 10.0) в конце каждого цикла проверки для обеспечения того, что данные, записанные в буфера файловой системы будут

сброшены на диск. В некоторых файловых и операционных системах, когда база данных использует буферизацию ввода-вывода, общая нагрузка на операции ввода-вывода может быть значительной. Так происходит потому, что когда процессы записи страниц записывают блоки базы на диск, данные помещаются в буферы файловой системы и записываются на диск позднее, возможно не раньше следующего вызова fdatasync (). Поэтому вся тщательная работа процессов записи страниц по планированию их активности для сглаживания процессов записи оказывается неэффективной.

При использовании параметра конфигурации -directio, база данных выполняет так называемый синхронный ввод-вывод и использует параметр O DSYNC или O SYNC при открытии файлов тома данных. В этом режиме все операции чтения и записи блоков данных используют буферы операционной системы обычным образом[7], а системный вызов pwrite()не возвращает значения до тех пор, пока данные не будут посланы на диск. В результате процессы записи страниц (page writers) будут осуществлять запись несколько медленнее, но в целом скорость ввода-вывода будет более равномерной и число пиков нагрузки на диск будет меньше.

Если база данных использует синхронный ввод-вывод, можно избежать использования fdatasync(), требующих много времени вызовов[8], что может блокировать поток вызовов до тех пор, пока все буферы, связанные с файлом не будут выгружены на диск.

При использовании параметра конфигурации -directio можно получить следующие преимущества:

• База данных не должна использовать занимающие много времени вызовы fdatasync()или sync(), и в случае sync(), устраняются ненужная дисковая активность, вызванная сбросом на диск данных, не связанных базой данных.

• Процесс записи на диск в целом проходит намного более равномерно, поскольку запись происходит тогда, когда этого требуют процессы записи страниц, а они стремятся организовать свою деятельность таким образом, чтобы обеспечить как можно более равномерную скорость записи.

При использовании –directio можно также увеличить число процессов записи страниц по сравнению с обычно необходимыми 1-2 процессами. Заметим, что применение параметра -directio в некоторых операционных системах дает незначительный эффект и может оказаться бесполезным в вашей ситуации. Было обнаружено, что наибольший эффект этот параметр дает в AIX-системах.

Параметр -directio может оказаться полезным и в Linux-системах, в ситуациях, когда возникает периодическое «замирание» системы под большой нагрузкой. Эти ситуации можно выявить с помощью программы vmstat,выполняющей мониторинг операций ввода-вывода на диск. Если имеются периоды высокой скорости операций ввода-вывода, после которых в течение нескольких секунд на диске операции ввода-вывода отсутствуют вовсе при большой нагрузке на систему, эту ситуацию можно исправить применением параметра -directio.

[7] Вы могли слышать утверждение, что параметр -directio позволяет обойти буферы операционной системы, но это неверно

[8] или вызовы sync()в версиях, предшествующих 10-й

6.9. Множественные наборы семафоров (Multiple Semaphore Sets)

База данных OpenEdge использует имеющиеся в операционной системе семафоры для различных целей, например для управления некоторыми внутренними блокировками и для блокировки процессов, которые должны ждать снятии блокировки строки, таблицы или схемы. В некоторых UNIX-системах производительность семафоров можно повысить, используя параметр -semsets. По умолчанию, OpenEdge RDBMS использует один большой набор семафоров для каждого экземпляра базы данных. Размер этого набора семафоров определяется максимальным числом пользователей, которые могут подсоединиться к базе данных. Параметр -semsets заставляет базу данных использовать указанное число более мелких наборов семафоров. Группам пользователей выделяются отдельные наборы семафоров. Это может повысить производительность, если в ядре системы имеется внутреннее разрешение конфликта, когда несколько пользователей одновременно выполняют операции на одном наборе семафоров. Повышение производительности будет наиболее заметным в системах с числом пользователей более 150 и может составить до 5%[9].

Оптимальное число семафоров зависит от конкретной нагрузки и используемой операционной системы. Мы предлагаем попытаться использовать по одному семафору на каждые 20–50 пользователей. В Linux-системах, использование параметра конфигурации -semsets не дает эффекта. Операционные системы Windows не используют наборы семафоров.

6.10. Размещайте тома журналов After-Image на двух дисках

Цель журналирования after-image – дать возможность восстановления данных, если произойдет сбой диска (дисков), на которых размещена ваша база данных. Поэтому НЕЛЬЗЯ хранить никакие тома after-image на тех же физических дисках, что и тома данных.

Чередуйте тома журналов after-image между двумя дисками, поместив первый на диск 1, второй – на диск 2, третий на диск 1 и т.д. Чередование томов между двумя дисками позволяет читать и архивировать заполненные тома, не замедляя запись новых данных в текущий том. Если у вас нет недостаточного количества дисков, чтобы отвести два из них для журналирования after-image, разместите все тома after-image на одном диске.

6.11. Файловые системы

Большинство операционных систем может работать с различными файловыми системами. Во многих случаях выбор файловой системы практически не влияет на производительность базы данных, поскольку системы баз данных обычно не создают и не стирают файлов в большом количестве. Многие файловые системы достаточно хорошо подходят для хранения баз данных, но есть и некоторые исключения. Ниже в таблице для различных операционных систем приведены некоторые файловые системы, которые хорошо работают с базами данных.

[9] Ваши показатели могут отличаться.

Операционная система	Файловая система
Linux	ext3 или JFS
AIX	JFS или JFS2
Solaris	UFS или VxFS
Tru64	AFS
Windows	NTFS

Ниже в таблице указаны некоторые файловые системы, которые никогда не следует использовать.

Операционная система	Файловая система
Windows	FAT16 и FAT32
HP-UX	HFS
Linux	ReiserFS

6.12. Параметр монтирования noatime

Файловые системы, используемые в Linux и UNIX, имеют множество параметров, которые можно указать во время монтирования файловой системы. Один из таких параметров noatime.
Он позволяет не обновлять время последнего доступа для файлов. Указав во время монтирования системы этот параметр, можно несколько снизить дополнительные расходы при работе файловой системы. Ваши показатели могут различаться, в зависимости от операционной системы.

6.13. Увеличьте значение SHMMAX

Во многих операционных системах, максимальный размер сегмента совместно используемой памяти устанавливается меньше, чем максимальный размер, который может использовать OpenEdge RDBMS. В этом случае база данных будет создавать много более мелких сегментов. Следует установить максимальный размер равным 128 МБ (134,217,728 байт).
В тестах под Linux, это тоже привело к увеличению производительности на скромные 2%.

6.14. Планировщики операций ввода-вывода в Linux

В Linux-системах на ядре 2.6 вы можете выбирать из нескольких имеющихся планировщиков ввода-вывода. Тесты показывают, что некоторых из них следует избегать. Мы рекомендуем использовать планировщик ”deadline”. Избегайте использования планировщиков CFQ и опережающих планировщиков (anticipatory schedulers). Планировщик CFQ показывает особенно слабые результаты в тестах. Планировщик deadline дает производительность примерно на 10% выше, чем опережающий планировщик. Выбор планировщика производится добавлением строки в конфигурационный файл загрузки (boot loader configuration file). Например, для использования планировщика deadline, добавьте строку "elevator=deadline".

6.15. Используйте специально выделенный сервер для базы данных

Чем больше задач вы запускаете на машине, на которой расположена база данных, тем больше ресурсов вы отнимаете у базы, снижая ее производительность. Это означает, что не следуе использовать программы печати, программы, работающие с файлами, почтовые программы, экранные заставки, Microsoft Office и т.д.

7. Вопросы, касающиеся настройки сети (Networking-specific Topics)

• Сохраните локальную копию кэша схемы. Используйте команду: SAVE CACHE COMPLETE foo TO foo.cache для сохранения локальной копии информации о схеме базы данных. Соединение с БД будет происходить быстрее, потому что в момент соединения из БД будет считываться меньшее количество информации. Не забывайте обновлять кэш схемы при изменении таблицы, определении индексов или при добавлении новых.

• Используйте гигабитный Ethernet. При использовании для сетевых клиентов Gigabit Ethernet вместо 10-мегабитного, можно получить почти такую же производительность, что и для клиентов в режиме самообслуживания, запущенных на той же машине, что и БД.

• Сконфигурируйте сетевые интерфейсы для дуплексных соединений (FDX). Большинство современных сетевых устройств, таких как сетевые карты и свичи, поддерживают дуплексную передачу в Ethernet. В этом случае можно ощутимо повысить скорость сети. Заметим, что все устройства в одном сегменте сети должны быть сконфигурированы для дуплексной передачи.

• Увеличьте максимальный размер сообщения (параметр конфигурации -Mm).
При использовании буфера сообщений большего размера доступ к БД через сетевое соединение будет улучшен, т.к. будет посылаться меньшее число сообщений. По умолчанию размер буфера сообщений равен 1024 Б, что достаточно мало. Лучше использовать значение 16 КБ. Заметим, что и клиент и сервер, должны использовать одно и то же значение. Заметим также, что только некоторые сообщения требуют максимального буфера для посылки. Если сообщения меньше максимального размера, то будет передано только нужное количество байт.

• Уменьшите число клиентов на сервер. При снижении максимального числа клиентов на сервер каждый серверный процесс будет действовать в интересах меньшего числа клиентов и сможет обеспечить лучшее время отклика, поскольку когда придет новый запрос от клиента, вероятность того, что сервер уже занят, будет меньше. Если вы уменьшаете число клиентов на сервер, вам придется увеличить максимальное число серверов.

• Используйте множественные брокеры. Применение множественных брокеров дает возможность обеспечить различные уровни сервиса для разных пользователей. Например, вы можете сконфигурировать один брокер для пакетных задач, а другой – для интерактивных пользователей.

8. Вопросы, касающиеся настройки SQL

При использовании OpenEdge SQL Server с интерфейсами ODBC и JDBC, есть множество приемов, позволяющих увеличить производительность.

• Добавьте индексы – Интерактивные SQL-запросы, независимо от того, генерируются они программами вроде Crystal Reports или EasyAsk, или прямо вводятся пользователем, обычно более непредсказуемы, чем те, что встроены в приложения. Создание дополнительных индексов помогает при выполнении запросов, сгенерированных подобными средствами.

• Используйте отдельный брокер для SQL – Используя отдельный брокер для SQL-соединений, можно отделить их от 4GL-соединений, и управлять ими независимо. Это дает возможность использовать различные значения параметров конфигурации, например максимальное число серверов и число клиентов на сервер.

• Используйте области данных II типа – Разместив свои таблицы и индексы в областях данных II типа, вы дадите возможность SQL-серверу использовать преимущества более высокой производительности и других возможностей, предоставляемых областями данных II типа.

• Обновляйте статистику – Для выбора из многих возможных планов выполнения запросов оптимизатор SQL-запросов полагается на статистику о данных в БД. Следует время от времени выполнять команду UPDATE STATISTICS, чтобы быть уверенным в актуальности данных статистики. Важнее всего обновлять статистику индексов.

• Конструкция WITH оператора SELECT – Конструкция WITH оператора SELECT дает возможность более тонкого управления поведением блокировки и позволяет настроить период ожидания снятия блокировки (lock-wait timeout).

• NO REORDER – Вы можете использовать параметр NO REORDER вместе с конструкцией (clause) FROM в операторах SELECT для принудительного соединения таблиц слева направо, как написано в тексте запроса. Иногда это может быть полезно для повышения скорости конкретных соединений, если оптимизатор делает неправильный выбор.

• Эффективно используйте API – web-сайт разработчиков DataDirect предлагает несколько технических статей, в которых обсуждается, как нужно использовать API для ODBC и JDBC, чтобы получить наилучшую производительность своего SQL-приложения.

• Используйте версию 10.1 OpenEdge, в которой для повышения производительности сделано много усовершенствований в SQL-сервере, а также в драйверах ODBC и JDBC, использующих новый, более быстрый SQL-протокол.

9. Вопросы, касающиеся настройки Windows

В данном разделе рассматривается, что можно сделать, чтобы Windows-серверы работали лучше.

• Используйте файловые системы NTFS. В Windows можно выбирать из нескольких файловых систем. Используйте NTFS. Никогда не используйте FAT-16 и FAT-32.

• Установите размер кластера NTFS 8КБ – по умолчанию размер кластера в файловой системе NTFS равен 4КБ. Его увеличение увеличит производительность для больших файлов, и, кроме этого, при размере кластера 8 КБ вы сможете использовать для томов данных и журналов транзакций размер блока 8 КБ. Для форматирования нового раздела NTFS используйте следующую команду:
format <drive>:/fs:ntfs /A:8k.

• Сконфигурируйте оптимизацию памяти для приложений, а не для служб, работающих в фоновом режиме. Для этого выполните следующие действия:
– Щелкните правой кнопкой мыши по ”Мое сетевое окружение”
– Выберите ”Свойства”.
– Когда появится диалоговое окно ”Сеть и удаленный доступ к сети”, щелкните правой кнопкой по ”Соединения в локальной сети (Local Area Connections)”
– Снова выберите ”Свойства”.
– В панели ”Компоненты”, щелкните ”Общий доступ к файлам и принтерам в сетях Microsoft (File and Printer Sharing for Microsoft Networks)”.
– Снова щелкните по ”Свойства”!
– Выберите ”Оптимизировать производительность для сетевых приложений (Maximize Throughput for Network Applications)”.
– Щелкните по ”OK”.

• Удалите программы, установленные без вашего ведома – Windows-системы исключительно подвержены установке программ без вашего уведомления. Достаточно просто посетить вредоносный web-сайт в Internet Explorer. Подобные программы могут значительно снизить производительность, надежность и безопасность системы.
Для удаления подобных программ следует использовать такие средства как Ad-Aware или Spybot Search and Destroy. Серверы баз данных, работающие в Windows, вообще не должны использоваться для доступа в Интернет.

• Короткие имена файлов – современные системы Windows не используют короткие имена файлов из 8 символов, какие использовались в DOS, но они по-прежнему генерируются для совместимости. Вы можете получить небольшой прирост производительности, отключив создание кратких имен. Просмотр каталогов и создание файлов будет происходить немного быстрее. Кроме того, для повышения производительности БД, не следует на сервере базы данных создавать и удалять множество файлов; после запуска БД на сервере БД нет необходимости часто открывать много файлов.

• Отключите все ненужные сервисы/службы – большинство Windows-систем имеет множество системных служб, запускаемых по умолчанию. Отключив те из них, которые вам не нужны, вы можете высвободить некоторый объем памяти и процессорного времени (CPU cycles), которые можно использовать лучшим образом. На web-сайте TechNet компании Microsoft (по адресу http://www.microsoft.com/technet) имеется статья ”System Services for the Windows Server 2003 Family and Windows XP Operating Systems”, описывающая

все эти службы. Следующая web-страница: http://majorgeeks.com/page.php?id=12 объясняет, какие из них следует отключить.

• Отключите все ненужные запускаемые при старте системы программы – многие системы Windows сконфигурированы таким образом, что при старте системы запускается множество программ. Эти программы потребляют память и процессорное время. Следует отключить все ненужные вам программы.

• Отключите украшения GUI. Различные варианты более изящного оформления окон и диалоговых окон требуют памяти и процессорного времени.

• Отключите общий доступ к файлам. Сервер БД не должен использоваться для других целей – печати или хранения файлов, потому что все эти операции требуют памяти, процессорного времени и снижают производительность жестких дисков по чтению и записи.

• Удалите Office – Microsoft Office имеет множество фоновых процессов, использующих память и процессорное время. Наиболее вредный – служба индексирования файлов.

• Не используйте экранные заставки – многие экранные заставки для Windows потребляют значительное количество процессорного времени. Для оптимальной производительности выберите в качестве заставки просто черный экран.

• Не запускайте антивирусные программы – они могут потреблять значительное количество процессорного времени и снижать производительность жестких дисков. Вы можете спросить: ”но как же мне тогда защитить систему Windows от заражения вирусами?”. Установите Linux☺.

10. Заключение

Во введении мы утверждали, что увеличили производительность с 30 до почти 600 транзакций в секунду, выполнив всего несколько простых действий по настройке. Мы сделали следующие изменения в настройках по умолчанию:

• Установили размер блока БД 8192
• Установили размер пула буферов (-B) to 64 000
• Сделали страйпинг томов данных, распределив их по 8 дискам
• Поместили журнал bi на отдельный диск
• Запустили 4 процесса записи страниц[10]
• Запустили процесс записи bi
• Установили размер кластера bi 16МБ
• Установили значение параметра -spin равным 50 000
• Увеличили число буферов bi (-bibufs) до 25

Рекомендации, приведенные в данной монографии, должны отправить Вас в дальнюю дорогу улучшения производительности, но не следует при этом забывать о трех вещах:

• Все системы отличаются друг от друга. Ваши показатели будут отличаться от приведенных.

• Записывайте, что вы сделали и почему. Позднее, когда вы забудете о своих действиях, эти записи помогут вам вспомнить свои действия и разобраться в них. Пишите комментарии в сценариях и .pf-файлах чтобы документировать их.

• Остановитесь, когда достигнете достаточного результата. Большинство людей всегда может найти способ еще что-нибудь улучшить и все равно всегда останется еще что-нибудь, на что пока не нашлось времени. Чем больше вы настраиваете БД, тем меньше обычно становится эффект от ваших следующих действий. Вы можете добиться значительного результата, выполнив всего несколько рекомендаций. Добавив памяти и распределив нагрузку между многими дисками вы получите основной эффект от настройки БД. Чего же вы ждете?

[10] Вероятно, это больше, чем потребуется вам. Одного или двух должно быть достаточно.