1С·БИТРИКС

Компания «1С-Битрикс» Системы управления веб-проектами

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

1С-Битрикс: Управление сайтом

Рекомендации по конфигурированию веб-систем для оптимальной работы с продуктом

Версия 5.0.2
Дата: 06.06.2006

1С·БИТРИКС

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

Содержание

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

Вступление

Документ предназначен для технических специалистов и системных администраторов.

В данном документе приведены рекомендации по настройке серверного ПО, которые выполняются сотрудниками компании "1С-Битрикс", при конфигурировании проектов с посещаемостью более 10 тысяч уникальных пользователей в день, либо при недостаточности системных ресурсов для обработки меньшей нагрузки.

При составлении документа подразумевалось, что разработчик владеет базовыми знаниями по управлению серверным программным обеспечением, обладает навыками управления базами данных, владеет языком программирования **PHP**.

Все рекомендации относятся в основном к **UNIX** системам или **Windows** системам, использующим веб-сервер **Apache**.

Рекомендованные решения не являются единственно возможными. Предполагается, что данные инструкции послужат руководством для создания и доработки системы в соответствии с имеющимися ресурсами, оборудованием, конфигурациями из нескольких серверов.

Документ составлен при консультационной поддержке компании CifNet USA (www.cifnet.com) и лично Евгения Круглова. Компания CifNet администрирует наибольшее число выделенных серверов с программным продуктом *"1С-Битрикс: Управление сайтом"*.

⚠ ***Важно!*** *Рекомендации подразумевают, что выполнены все обязательные требования и по возможности рекомендуемые установки на странице Настройки -> Инструменты -> Проверка сайта в административном разделе продукта.*

Типовые настройки серверного программного обеспечения рассчитаны на типовое оборудование и приложения. Внесение некоторых конфигурационных изменений в серверное ПО позволяет:

- в несколько раз увеличить производительность системы в целом;
- сократить время генерации страниц;
- увеличить устойчивость системы к пиковым нагрузкам.

Большие веб-проекты

Комплексное сочетание целого ряда факторов может привести к тому, что проект получает название "большой":

- большая посещаемость проекта в среднесуточном выражении;

- высокие пиковые нагрузки;
- невозможность кэшировать страницы в силу сложной бизнес-логики;
- большие интерактивные проекты: форумы, блоги, журналы;
- индивидуальные страницы для отдельных пользователей;
- большие объемы данных;
- недостаточность аппаратных ресурсов по отношению к предыдущим факторам.

Большие веб-проекты требует определенного конфигурирования и отношения.

Почему умирают сайты?

При больших нагрузках проект может прекратить нормальное функционирование. Возможные причины:

- **нехватка оперативной памяти** для нормальной работы процессов веб-сервера и базы данных; начинается свопирование процессов, производительность резко падает;
- **нехватка процессорных ресурсов** для одновременного выполнения процессов и обеспечения адекватного для пользователя времени реакции;
- **недостаточная производительность базы данных** при одновременных конкурентных запросах, невозможность полностью использовать ресурсы сервера;
- **общая несбалансированность веб-системы** при пиковых нагрузках и быстрое снижение производительности даже при незначительных стрессах;
- **нехватка дискового пространства**.

Оперативная память и большие процессы

В обычной конфигурации веб-сервер обрабатывает все запросы к PHP-страницам, к графическим файлам, бинарным файлам, таблицам стилей и другим составным частям сайта.

На одну PHP-страницу может приходиться от нескольких десятков до нескольких сотен графических элементов. Загрузка бинарных файлов, XML-файлов, таблиц стилей так же выполняется веб-сервером в обычных конфигурациях.

Учитывая, что процессы веб-сервера занимают достаточно много памяти, одной из важнейших задач является минимизация числа запросов, обрабатываемых веб-сервером.

Оперативная память и медленные каналы связи

Ключевая проблема при обработке запросов к сайту состоит в том, что пользователи сайта работают на относительно **медленных каналах связи** по меркам веб-сервера.

Например, **время генерации** страницы может составлять **0.01 секунды**, а **время передачи** страницы клиенту даже с компрессией может занимать **от 5 до 50 секунд** и более.

В течение всего времени передачи страницы клиенту веб-сервер будет держать в памяти бездействующий процесс **Apache**, который будет только дожидаться завершения передачи данных, но не сможет высвободить память и обработать другой запрос до полного получения страницы клиентом.

Сервер неэффективно расходует оперативную память на медленных каналах.

Двухуровневая конфигурация веб-сервера

Наилучшим решением для устранения перечисленных проблем является создание двухуровневой системы: **Front-End + Back-End** для обработки запросов.

⚠ ***Определение! Front-End*** *– публичная часть проекта, обеспечивающая прием запросов от пользователей, трансляцию запросов к* ***Back-End*** *и выдачу непосредственного содержимого пользователю.*

⚠ ***Определение! Back-End*** *– исполнительная часть системы, которая обеспечивает выполнение PHP-скриптов, формирование контентных страниц и работу бизнес-логики приложений.*

Front-end

Front-End исполняет все запросы, которые возможно обработать самостоятельно без обращения к **Back-End**. Графические файлы и таблицы стилей с веб-сервера запрашиваются только в соответствии с политикой кэширования. После этого файлы хранятся в кэше **Front-End** и отдаются пользователям без обращения к **Back-End**.

Наиболее эффективна конфигурация, когда **Front-End** напрямую с диска считывает статические файлы и не нагружает **Back-End** запросами.

Желательно, чтобы только для получения PHP-страниц **Front-End** обращался к **Back-End** серверу для получения страницы и при необходимости ожидал освобождения **Back-End** процессов.

В качестве **Front-End** сервера можно использовать:

- **NGINX** - http://sysoev.ru/nginx/

- **SQUID** - http://www.squid-cache.org/
- **OOPS** - http://www.oops-cache.org/

или другой аналогичный продукт.

Наилучшие практические результаты получены при использовании **NGINX**.

Если вы используете кэширующий прокси-сервер в качестве **Front-End**, обязательно настраивайте время кэширования документов. Графические файлы и таблицы стилей, XML файлы и другие статические объекты с веб-сервера должны запрашиваются только в соответствии с политикой кэширования. После этого файлы хранятся в прокси-сервере и отдаются пользователям без обращения к **Back-End** и **Apache**. Рекомендуется устанавливать время кэширования графических файлов 3-5 дней. Пример настройки кэширования через файл **.htaccees** в корне веб-сервера:

```
ExpiresActive on

ExpiresByType image/jpeg "access plus 3 day"

ExpiresByType image/gif "access plus 3 day"
```

⚠ ***Важно!*** *Для работы этого примера необходимо, чтобы веб-сервер позволял переопределение переменных через файл* ***.htaccess*** *и модуль* ***mod_expires*** *был установлен. В некоторых случаях на* ***Front-end*** *политика кэширования настраивается независимо от настроек* ***Back-end****.*

Таким образом, **Front-end** будет кэшировать все графические изображения. Запросы к контентным страницам не будут кэшироваться и будут перенаправляться к **Back-end**.

Back-End

Back-end представляет собой обычный веб-сервер **Apache**, который исполняет PHP-приложения.

Back-end готов исполнять запросы на графические и статические документы, если вы используете кэширующий прокси-сервер для **Front-End**.

⚠ ***Важно****, чтобы число таких запросов было минимальным и 99% запросов приходилось на выполнение именно PHP-страниц.*

Конфигурируя **Back-End**, можно добиться значительного выигрыша в производительности и стабилизировать систему по расходу памяти.

В большинстве случаев **Back-End** представляет собой обычный веб-сервер **Apache**, работающий на нестандартном порту, к примеру, на порту 88, и отвечающий только на запросы с **localhost** или IP адреса прокси-сервера.

⚠ ***Совет администратору:*** *лучше использовать несколько внутренних IP адресов типа 127.0.0.2, 127.0.0.3 и т.д. с 80-м портом, иначе возможны нежелательные редиректы на неработающий порт у* ***Front-end****.*

Рассмотрим процесс обработки запроса пользователя к обычной странице сайта:

- запрос принимается **Front-End**, например, по адресу http://www.bitrixsoft.ru на 80 порту;
- транслируется веб-серверу **Apache** по адресу http://127.0.0.2:80/;
- запрос исполняется веб-сервером **Apache**;
- ответ принимается прокси-сервером;
- соединение между прокси-сервером и **Apache** сервером закрывается;
- память высвобождается;
- прокси-сервер передает готовую сформированную страницу посетителю по медленному каналу;
- получив страницу, браузер посетителя посылает последовательно серию запросов на графические элементы и таблицу стилей;
- все запросы принимаются прокси-сервером и обрабатываются без обращений к **Back-end**.

Как показала практика, такая конфигурация в несколько раз разгружает машину, уменьшает объемы потребляемой памяти, значительно ускоряет время обработки запросов и позволяет выделить больше памяти для работы базы данных. Такая конфигурация так же позволяет значительно разгрузить сервер при обработке большого числа закачиваемых файлов, например, музыки, дистрибутивов программных продуктов, презентаций и других физических объектов.

Стабилизируем Back-end по расходу оперативной памяти

⚠ ***Важно!*** *Для стабилизации системы по расходу памяти и минимизации числа запущенных процессов* ***Back-end****, можно рекомендовать установить параметр* ***MaxClients*** *в значении* ***5-50*** *в настройках* ***Apache****.*

Число необходимо подбирать исходя из системных ресурсов и нагрузки. Наличие лимита по числу возможных процессов позволит поставить жесткое ограничение по потреблению памяти и исключить выход машины из строя при стрессовых нагрузках.

Наличие лимита позволяет включить **Persistent** соединение к базе и уменьшить расходы на соединение и число работающих процессов базы данных.

Также рекомендуется подбирать параметры управления процессами в соответствии с установленным лимитом **MaxClients**. Например, если *MaxClients = 20*, тогда:

```
MinSpareServers 20

StartServers 20

MaxClients 20
```

Производительность PHP

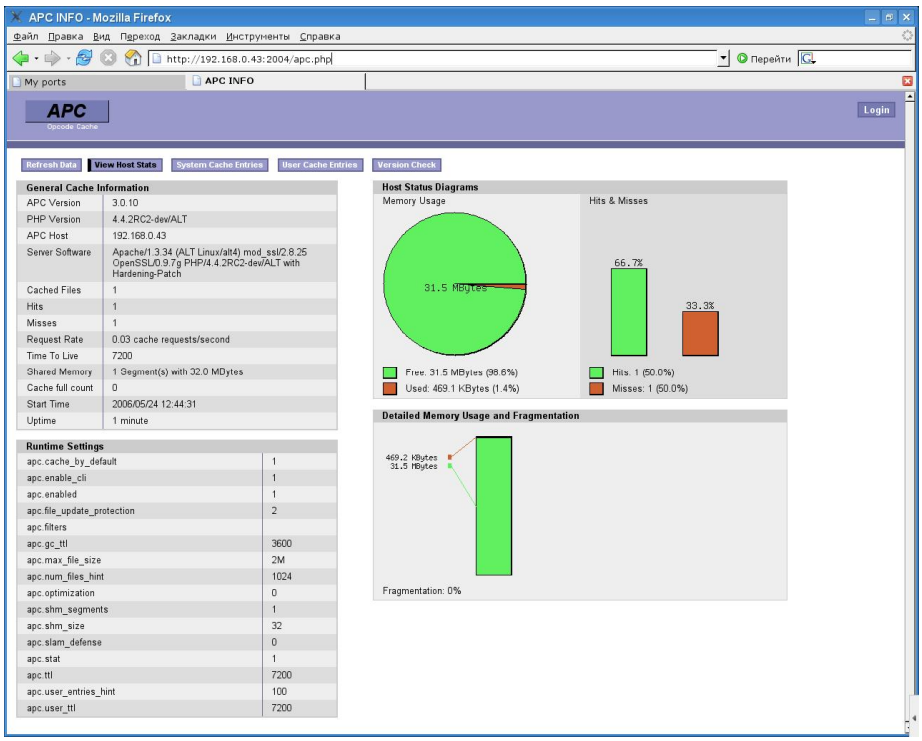
До **60%** рабочего времени веб-серверы тратят на повторную компиляцию PHP-кода перед исполнением.

Ключевой способ снизить нагрузку на процессор – использовать прекомпиляторы PHP-кода.

PHP прекомпиляторы:

- Alternative PHP Cache (APC) - http://pecl.php.net/package/apc
- eAccelerator - http://www.eaccelerator.net/
- Zend Performance Suite - http://www.zend.com/
- Turck MMCache - http://turck-mmcache.sourceforge.net/
- PHP Accelerator - http://www.php-accelerator.co.uk/
- AfterBurner Cache - http://www.bwcache.bware.it/

В последнее время активное развитие получил **APC** (Alternative PHP Cache). Есть информация, что данный продукт войдет в стандартную поставку версии **PHP6**. Мы рекомендуем использовать этот программный продукт или **eAccelerator** как качественную и проверенную альтернативу. Не забываете выделять достаточный объем оперативной памяти для размещения скомпилированных PHP файлов. Обычно бывает достаточно **16M**, но для уверенности можно увеличить объем выделяемой памяти до **32M**, в расчете на файлы административного раздела. Прекомпиляторы разделяют оперативную память между процессами веб-сервера и выделяемый объем будет использован только однажды.



Для **уменьшения потребляемой памяти** процессами веб-сервера, в котором запускается PHP, желательно исключить из компиляции или динамической загрузки все неиспользуемые модули.

Для ускорения работы с PHP **рекомендуется сохранять** файлы сессий в каталоге, который представляет собой виртуальный диск **в памяти** или использовать установку *session.save_handler=mm* в **php.ini**

Switch from file based sessions to shared memory sessions. Compile PHP with the --with-mm option and set session.save_handler=mm in php.ini. Informal benchmarks suggest that session management time is halved by this simple change. Added 1 Dec 2001. This hint should only be used for PHP 4.2.0 and above as there were bugs before this.

http://us4.php.net/ref.session

Если есть возможность, рекомендуется использовать системный “RAM disk”:

⚠ ***Note:*** *Optionally you can use shared memory allocation (mm), developed by Ralf S. Engelschall, for session storage. You have to download mm and install it. This option is not available for Windows platforms. Note that the session storage module for mm does not guarantee that concurrent accesses to the same session are properly locked. It might be more appropriate to use a shared memory based filesystem (such as tmpfs on Solaris/Linux, or /dev/md on BSD) to store sessions in files, because they are properly locked.*

Сжатие страниц

Компрессия при передаче данных между сервером и клиентом обеспечивает сжатие страниц **для ускорения вывода содержания** сайта пользователям.

Сжатие в несколько раз уменьшает объем передаваемых HTML-данных между сайтом и браузером клиента, что существенно увеличивает скорость работы как для посетителей, так и для администраторов сайта.

Предпочтительнее сжимать страницы на **Front-End**. При отсутствии такой возможности можно сжимать данные на **Back-End**.

Возможные способы сжатия:

- **mod_deflate** http://sysoev.ru/mod_deflate/
- модуль **Компрессии**, входящий в состав продукта *"1С-Битрикс: Управление сайтом"*
- стандартные модули **PHP**
- стандартные модули **Apache**

Дополнительные рекомендации для двухуровневой конфигурации:

► Для правильной работы модуля статистики необходимо обеспечить передачу реального IP адреса с **Front-end** в **Back-end**. В частности, **SQUID** обеспечивает декларирование переменной *HTTP_X_FORWARDED_FOR*. Для замены переменной в продукте необходимо в файле **/bitrix/php_interface/dbconn.php** вставить подобный пример кода:

```
if(strlen($_SERVER["HTTP_X_FORWARDED_FOR"])>0 &&
$_SERVER["REMOTE_ADDR"]=="127.0.0.1")
{

if($p = strrpos($_SERVER["HTTP_X_FORWARDED_FOR"], ","))
      {
            $_SERVER["REMOTE_ADDR"] = $REMOTE_ADDR =
trim(substr($_SERVER["HTTP_X_FORWARDED_FOR"], $p+1));
            $_SERVER["HTTP_X_FORWARDED_FOR"] =
substr($_SERVER["HTTP_X_FORWARDED_FOR"], 0, $p);
      }
else
      $_SERVER["REMOTE_ADDR"]=$REMOTE_ADDR=$_SERVER["HTTP_X_FORWARDED_F
OR"];
}
```

⚠ ***Совет администратору.*** *Установка такого обработчика не требуется при использовании* ***mod_realip*** *(* http://sysoev.ru/mod_realip/ *). С ним надо добавить "RealIP 127.0.0.1" или соответствующий адрес, на котором работает* ***Back-end****, в настройки нужного виртуального сервера* ***httpd.conf****, и* ***Apache*** *перепишет адрес сам.*

► В конфигурации **Apache** на **Back-end** желательно отключить **KeepAlive**. Так как **Front-end** находится или на этой машине или "рядом", более быстрое высвобождение ресурсов предпочтительнее;

Цель построения Front-end + Back-end системы

В результате построения двухуровневой архитектуры и выполнения ряда рекомендаций мы должны получить следующие результат:

- система **стабилизирована по расходу памяти**;
- **Front-End и Back-End занимают заранее отведенный объем памяти**, который не будет расти даже при увеличении нагрузки;
- в стрессовой ситуации **система будет стабильно и равномерно обрабатывать запросы**, **Back-End** не будет увеличивать число одновременно выполняемых процессов выше установленного лимита *MaxClients*, **Front-End** будет принимать все запросы от пользователей и ожидать освобождения процессов **Back-End**, не начнется регрессия производительности;
- **процессорные ресурсы существенно высвобождены** за счет прекомпиляции PHP-кода;
- пользователи комфортно работают со сжатыми страницами.

Оптимизация базы данных

Оптимизация работы с базой данных для **MySQL**, **Oracle**, **MSSQL** является одной из важнейших стратегий в оптимизации системы в целом.

Стратегия оптимизации состоит в том, чтобы **уменьшить число дисковых операций** при работе с базой данных и **увеличить возможности параллельной обработки** для наилучшего использования ресурсов машины.

В результате создания двухуровневой архитектуры удается **выделить большой объем памяти**, гарантированно не занятой веб-сервером.

Оставшаяся память может быть распределена под базу данных, буферы для чтения, кэш-память, области сортировки и другие служебные операции.

В большинстве систем около **60-80% оперативной памяти** удается выделить **для базы данных**.

Постоянное соединение с базой данных

Важным параметром, влияющим на потребление памяти базами данных (**MySQL\Oracle**), является максимальное число одновременных соединений *max_connections*.

При использовании двухуровневой архитектуры **Frone-end/Back-end** можно в несколько раз уменьшить число одновременных соединений и высвободить больше памяти для сортировки данных в памяти и буферизации.

Если установлено число *MaxClients* в настройках веб-сервера **Back-end**, то можно считать, что максимальное число соединений к базе данных будет соответствовать максимальному числу одновременных соединений.

⚠ ***Важно!*** *Таким образом, число соединений фиксировано!*

Рекомендуется подбирать параметр *max_connections* таким образом, чтобы иметь резерве в 10-20% свободных соединений от максимального значения.

База данных MySQL и ее настройка

Оптимизация работы с базой данных для **MySQL** версии продукта является одной из важнейших стратегий в оптимизации системы в целом, так как продукт очень активно работает с базой данных.

Стандартно **MySQL** работает с форматом данных **MyISAM**. Простой формат данных хранит каждую таблицу с данными или индекс в отдельном файле. В целом, на небольших по нагрузке сайтах данный формат является наиболее быстрым, хотя и не

обеспечивает полной целостности и надежного хранения данных за счет отсутствия транзакций.

Основным недостатком **MyISAM** с точки зрения производительности является локировка на уровне таблицы при выполнении тех или иных операций. В результате, при большой нагрузке **MySQL** именно **MyISAM** таблицы становятся основным узким местом в системе, мешая увеличивать утилизацию машины и число обрабатываемых запросов. Это так же приводит к увеличению времени работы страницы за счет ожидания используемых таблиц на уровне **MySQL**.

Рекомендуется переводить все таблицы проекта в формат данных **InnoDB** (http://dev.mysql.com/doc/mysql/en/innodb.html). Формат **InnoDB**, начиная с версии **MySQL** 4.0, входит в стандартную поставку продукта и обеспечивает надежное хранение данных, транзакционность и локирование данных на уровне строки.

Поменять тип таблиц на **InnoDB** можно следующим образом:

► В административном меню на странице *Настройки -> Инструменты -> SQL запрос* выполнить команду:

```
SHOW TABLES
```

► В результате Вы получите список всех текущих таблиц продукта. Для каждой таблицы необходимо выполнить команду:

```
ALTER TABLE <ИМЯ ТАБЛИЦЫ>, type=InnoDB
```

В "Вопросах и ответах" на сайте http://www.bitrixsoft.ru приведен пример создания скрипта для перевода таблиц в **InnoDB**.

► После перевода таблиц Вашей базы в **InnoDB** надо добавить в файл **/bitrix/php_interface/dbconn.php** нижеследующий код:

```
define("MYSQL_TABLE_TYPE", "InnoDB");
```

Переход на **InnoDB** позволяет избежать возникновения узкого участка в производительности при работе с базой данных и в полном объеме использовать системные ресурсы.

⚠ ***Важно!*** *Конфигурируйте InnoDB! Обязательно рекомендуется настраивать все переменные* ***innodb_**

⚠ ***Внимание!*** *Для лучшей производительности базы данных при работе с* ***InnoDB*** *рекомендуется настроить* ***my.cnf*** *для* ***MySQL*** *в разделе параметров для* ***InnoDB*** *http://dev.mysql.com/doc/mysql/en/innodb-configuration.html*

Наибольшее внимание следует обратить на следующие параметры и примеры:

```
set-variable = innodb_buffer_pool_size=250M

set-variable = innodb_additional_mem_pool_size=50M

set-variable = innodb_file_io_threads=8

set-variable = innodb_lock_wait_timeout=50

set-variable = innodb_log_buffer_size=8M

set-variable = innodb_flush_log_at_trx_commit=0
```

Для сокращения времени ответа сервера можно использовать отложенные транзакции и в частности устанавливать переменную

```
set-variable = innodb_flush_log_at_trx_commit=0
```

Если **MyISAM** уже не используется активно, можно высвободить память в пользу **InnoDB** параметров.

Желательно, чтобы кэш вмещал в себя основной объем данных, используемых продуктом в работе. Обычно для работы базы данных выделяется около 60-80% свободной памяти в системе.

Рекомендуется так же производить многопотоковую (multithreading) сборку **MySQL** для эффективного использования ресурсов вычислительной машины..

⚠ ***Пример*** *рекомендуемых настроек для сервера с 2Г оперативной памяти, работающего с операционной системой* ***FreeBSD/Linux****:*

```
set-variable = table_cache=512
```

В составе продукта около 250 таблиц, поэтому рекомендуется увеличивать кэш для заголовков таблиц.

```
set-variable = key_buffer_size=16M

set-variable = sort_buffer=8M

set-variable = read_buffer_size=16M
```

Эти параметры используются только для ***MyISAM****. Если в базе нет таблиц* ***MyISAM****, то кэш лучше установить в минимальные значения.*

```
set-variable = query_cache_size=64M
```

```
set-variable = query_cache_type=1M
```

*Кэширование результатов запросов. Обычно бывает достаточно 32 Мб (смотреть на статус **Qcache_lowmem_prunes**). Максимальный размер результата (по умолчанию - 1 Мб) можно регулировать.*

```
set-variable = innodb_buffer_pool_size=780M
```

Основной буфер, чем больше, тем лучше:

```
set-variable = innodb_additional_mem_pool_size=20M
```

Вспомогательный буфер на внутренние структуры большой делать не имеет смысла:

```
set-variable = innodb_log_file_size=100M

set-variable = innodb_log_buffer_size=16M
```

*Чем больше размер лог-файла, тем реже будет происходить запись в основной файл данных. Суммарный размер лог-файла может быть сопоставим с величиной **innodb_buffer_pool_size** (по умолчанию ведется два лога).*

```
set-variable = innodb_flush_log_at_trx_commit=0
```

Отложенная фиксация транзакций, раз в секунду

```
set-variable = tmp_table_size=32m
```

Размер временных таблиц рекомендуется увеличивать до 32M.

⚠ ***Совет администратору!** Переход на **InnoDB** может привести к значительному замедлению некоторых масштабных операций записи и обновления данных. Это связано с тем, что все операции по изменению данных начинают выполняться с использованием транзакций. Вы должны сами определить оптимальность перехода вашего проекта на формат данных **InnoDB**.*

⚠ ***Внимание!** Если по каким-то причинам вы решили продолжить работу с типом данных **MyISAM**, обязательно проведите конфигурирование **MySQL** для увеличения объемов кэшируемой информации, областей сортировки и минимизации числа дисковых операций. Использование для базы данных 60-80% оперативной памяти может ускорить работу стандартного проекта в несколько раз.*

База данных Oracle и ее настройка

Продукт "*1С-Битрикс: Управление сайтом*" версии **Oracle** протестирован и успешно работает с Oracle 9i и 10g.

Использование **Oracle** в качестве базы данных позволяет значительно увеличить надежность системы и производительность при большой нагрузке. Архитектура **Oracle** позволяет практически полностью использовать ресурсы серверов, добиваясь при этом прекрасной интерактивности приложений даже при работе с большими объемами данных, построении обширных отчетов с большим числом одновременных соединений. Работа с **Oracle** так же позволяет использовать самые разные варианты масштабирования интернет-проекта.

Общие рекомендации по настройке **Oracle** совпадают с рекомендациями для **MySQL** по конфигурированию системы для уменьшения дисковых операций чтения, сортировки и перестроения.

Обратите внимание на системные переменные управления памятью. Рекомендуется использовать до 60-80% оперативной памяти для кэширования данных за счет управления переменными:

- *db_cache_size*
- *shared_pool_size*
- *pga_aggregate_target*

Для уменьшения расходов на повторный разбор SQL запросов рекомендуется ставить следующие значения:

```
cursor_space_for_time=TRUE

cursor_sharing=SIMILAR

star_transformation_enabled=TRUE
```

Если **Oracle** установлен на той же машине, что и веб-сервер, рекомендуется использовать протокол **IPC** (PROTOCOL = IPC) и (KEY = EXTPROC) для соединения к базе для исключения работы через IP стек.

Если реализована двухуровневая схема обработки запросов с **Front-end** и **Back-end** и установлен параметр *MaxClients*, можно без опасений использовать постоянные соединения между **PHP** и **Oracle** (Persistent connection), выполнив в файле **/bitrix/php_interface/dbconn.php** следующую установку

```
define("DBPersistent",true);
```

Начиная с релиза Oracle 10g R2 рекомендуется использовать отложенные транзакции (Enhanced COMMIT).

Использование отложенных транзакций существенно ускоряет проект, позволяет снять прямую зависимость от производительности дисковой подсистемы и существенно уменьшить время генерации страниц.

Выводы

Практика эксплуатации PHP проектов под большой нагрузкой показывает, что при правильном подходе к распределению ресурсов и конфигурированию, производительность одного и того же оборудования может отличаться в несколько раз.

Для получения более подробной информации обращайтесь в техническую поддержку компании "1С-Битрикс": http://dev.1c-bitrix.ru/support.