**Курс для хостеров. Конфигурирование веб-систем**

**Front-end позволяет**

* защитить систему от медленных каналов пользователей и ускорить время выполнения запросов к базе данных
* снять зависимость производительности системы от медленных каналов пользователей
* уменьшить число запросов к Back-end за счет самостоятельной обработки статических файлов
* увеличить производительность PHP
* уменьшить число соединений к базе данных
* существенно уменьшить потребление оперативной памяти при обработке статических файлов

**Back-end – это**

* база данных MySQL/Oracle/MSSQL
* обычный веб-сервер Apache с установленным обработчиком PHP
* прокси-сервер или облегченный веб-сервер без PHP
* обычный веб-сервер Apache, только с неустановленным обработчиком PHP

**Для стабилизации системы по расходу памяти и для ограничения числа одновременно запущенных процессов Back-end нужно установить**

* MaxClients
* StartServers
* MinSpareServers

**Если на сервере одновременно запускается много процессов веб-сервера, то возможно**

* значительное снижение производительности системы из-за нехватки процессорных ресурсов для одновременного выполнения всех процессов
* новые процессы будут размещаться в виртуальной памяти, общая производительность системы снизится очень незначительно
* закончится место на диске
* перезапуск веб-сервера
* из-за нехватки оперативной памяти процессы будут вытесняться в виртуальную память и общая производительность системы резко упадет
* превышение установленного максимума для одновременных соединений к базе данных

**В результате построения двухуровневой архитектуры и выполнения ряда рекомендаций мы должны получить следующие результаты**

* процессорные ресурсы существенно высвобождены за счет прекомпиляции PHP-кода
* 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
* возможно безопасное использование постоянного соединения с базой данных без опасения превысить число возможных соединений; в памяти все время находится установленное число Back-end процессов, готовых к обработке запросов и с установленным соединением с базой данных;
* система стабилизирована по расходу памяти при запуске; Front-End и Back-End занимают заранее отведенный объем памяти, а при увеличении нагрузки будут использовать виртуальную память
* использование процессорных ресурсов ограничено числом одновременно работающих процессов Back-end в соответствии с MaxClients; не начнется регрессия производительности;
* пользователи комфортно работают со сжатыми страницами

**Время ожидания между Front-end и Back-end должно быть достаточно большим, чтобы**

* передать всю страницу пользователю на медленных каналах
* дождаться освобождения процессов Back-end, если все они заняты обработкой текущих запросов
* дождаться завершения работы длительных запросов к Back-end

**Если веб-сервер сам передает данные пользователю после их создания, то**

* число обработанных веб-сервером запросов напрямую зависит от скорости Интернет-канала посетителей сайта
* число обработанных запросов очень несущественно зависит от скорости Интернет-канала пользователя
* число обработанных запросов зависит только от производительности процессоров

**Как обойти Segmentation fault при использовании отложенной загрузки классов?**

* Определить в dbconn.php: define("NO\_BITRIX\_AUTOLOAD",true)
* Использовать другую версию PHP
* Использовать другой акселератор

**Front-end – это**

* компактный веб-сервер или кэширующий прокси-сервер
* обычный веб-сервер Apache с подключенным обработчиком PHP

**Чтобы обеспечить передачу реального IP адреса с Front-end в Back-end, необходимо**

* установить модуль mod\_realip для веб-сервера Apache
* отключить KeepAlive в Back-end
* произвести в Back-end замену переменной REMOTE\_ADDR на значение переменной HTTP\_X\_FORWARDED\_FOR, которая передается из Front-end

**Какой тип таблиц MySQL рекомендуется использовать для улучшения производительности при больших нагрузках?**

* MyISAM
* InnoDB

**Сжатие страниц модулем компрессии позволяет**

* ускорить работу PHP прекомпилятора и сделать работу пользователей более комфортной
* позволяет ускорить загрузку сайта со стороны клиента (пользователя сайта)
* снизить нагрузку на процессоры
* ускорить установку обновления программного продукта по технологии SiteUpdate

**Постоянное соединение с базой данных предпочтительнее, потому что**

* соединение к базе данных всегда открыто и тратится меньше времени и ресурсов на повторное открытие соединений
* SQL запросы исполняются быстрее
* база данных потребляет меньше оперативной памяти и больше памяти остается для кэширования данных

**Значение MaxClients надо подбирать так, чтобы**

* число процессов Back-end было всегда больше максимального одновременного числа запросов к сайту
* даже при стрессовых нагрузках запущенные процессы Back-end потребляли в среднем не более 90% процессорных ресурсов
* объем используемой памяти процессами Back-end никогда не превышал объем свободной оперативной памяти
* MaxClients должен быть меньше или равен максимальному числа соединений с базой данных

**Можно ли выносить временную папку MySQL на ramdisk в памяти**

* Да.
* Нет.
* Желательно это делать при наличии достаточного количества ОЗУ.

**Основной параметр настройки веб сервера для стабилизации расхода памяти**

* MaxClients в Аpache
* memory\_limit в php.ini

**Постоянное соединение с базой данных устанавливается**

* в файле /bitrix/php\_interface/dbconn.php константой DBPersistent
* в любом месте продукта конструкцией define("DBPersistent", true);
* в файле /bitrix/php\_interface/dbconn.php конструкцией define("DBPersistent", yes);

**Использование общего веб-сервера для обработки PHP программ и статических файлов**

* это лучшее решение для обеспечения оптимального расходования оперативной памяти
* неэффективное решение из-за большого и неэффективного использования оперативной памяти веб-сервером для обработки статических файлов
* неэффективное решение из-за наличия задержек при определении типа файлов: статические или PHP программы

**PHP-скрипт выполняется**

* только на сервере и представляет собой отдельную программу, которая компилируется и выполняется только один раз в момент первого запуска
* практически всегда на сервере, но в некоторых случаях может быть выполнен браузером
* только на сервере и является интерпретируемым языком

**Зачем нужен Zend optimizer+ ?**

* для ускорения работы php
* для ускорения работы Back-End
* для ускорения работы Front-End

**Для соединения с базой данных PHP использует**

* собственный обработчик соединений "Битрикс" для данного типа базы данных
* встроенные библиотеки PHP для данного типа базы данных

**Порядок работы Back-end:**

* получает запросы от Front-end и передает готовые (сгенерированные) страницы и статические файлы Front-end для передачи их пользователям.
* получает запросы от пользователей, обрабатывает и передает данные Front-end для передачи их пользователям
* получает запросы от Front-end, обрабатывает его и передает готовые (сгенерированные) страницы Front-end для передачи их пользователям.
* получает запросы от Front-end и передает готовые (сгенерированные) страницы и статические файлы пользователям

**Как обойти Segmentation fault на скрипты в cron или консоли?**

* Уменьшить время работы скриптов в настройках главного модуля
* В "падающем" скрипте определить: define('BX\_NO\_ACCELERATOR\_RESET', true)
* Уменьшить MaxClients в настройках Apache
* Обратиться в техподдержку битрикс
* Использовать другую конфигурацию PHP без подключения акселератора (или другую версию PHP)

**Соединение с базой данных по имени localhost позволяет**

* гарантированно соединиться с базой данных, установленной на локальной машине
* соединиться с базой данных без использования стека TCP/IP, что ускоряет работу

**Статические файлы на веб-сайте это**

* XML файлы, PowerPoint (PPT), Microsoft Word (DOC), PDF
* PHP страницы и страницы программного продукта "Битрикс"
* страницы HTML, которые не содержат программный код
* таблицы стилей CSS
* JavaScript файлы
* музыкальные MP3 файлы и PHP скрипты
* все графические файлы: gif, jpg, png, bmp

**Какие возможности MySQL стоит использовать для улучшения производительности?**

* при использовании InnoDB обязательно конфигурировать переменные innodb\_\*
* многопотоковую (multithreading) сборку MySQL
* отложенные транзакции для InnoDB (innodb\_flush\_log\_at\_trx\_commit)

**Для ускорения работы PHP интерпретатора рекомендуется**

* настраивать Front-end и Back-end конфигурацию
* использовать прекомпилятор PHP
* максимально ускорить работу базы данных