Некоторое время назад вышла новая, практически революционная версия php 7. Революционная, потому что обещает существенный прирост производительности, в отличие от предыдущих обновлений. По предварительным данным из описаний и обещаний, якобы в некоторых случаях может быть прирост скорости обработки php в разы. А если не повезет, то на 30-70%. Решил я это проверить на свою голову.

Данная статья устарела. Более подробную и актуальную информацию по обновлению php 7 читайте в новом материале на тему настройки web сервера nginx и php-fpm.

Содержание:

Введение

Я никогда не придавал большого значения версии php, так как не работал с высоконагруженными проектами. Для моих задач мне хватало любой версии, которую поддерживали необходимые мне скрипты. В основном это популярные движки сайтов, админка заббикса и другие прикладные инструменты системного администратора.

> Сразу предупреждаю, что не нужно слепо повторять то, что делал я. Просто примите к сведению информацию, которой я делюсь. Я не являюсь специалистом по тонкой настройке web серверов.

Я решил поэкспериментировать и проверить, насколько быстрее будет работать мой блог, если я перейду на **php 7**. Этот сайт работает на wordpress, до обновления он работал на php54 с включенной системой кэширования apc. Достаточно старая версия, но именно она ставится из стандартных репозиториев centos, которые я использую. Уже не помню точно, откуда он ставится, то ли из базового, то ли из epel. Как оказалось, не зря ставится эта версия. Серия моих экспериментов и проверок подтвердила, что именно на этой версии достигается максимальная производительность **в моем конкретном случае**.

Но обо всем по порядку. Для того, чтобы отследить изменения и понимать, стало лучше или нет, я решил провести некоторые замеры скорости работы сайта. Начал гуглить эту тему. Вариантов не особо много. Нашел 2 наиболее популярные утилиты, которыми пользуются для тестирования производительности web сервера: **ab** и **siege**. Первая входит в стандартные утилиты httpd или apache2, вторая как есть ставится через yum.

Я попробовал обе утилиты и остановился на siege. Она позволяет проводить измерения наиболее приближенные к реальному поведению пользователей на сайте. Не буду в этой статье подробно останавливаться на описании работы утилиты, в интернете информация есть, легко ищется. Если вам нужно, то сами все найдете и протестируете.

Я настроил siege и прогнал серию тестов, создающую разную нагрузку и зафиксировал результаты. После этого стал готовиться к обновлениям. Сразу скажу, что итоговый результат меня не удовлетворил и я вернулся на начальную свою конфигурацию. Обо всем по порядку расскажу.

Обновление php 5.4 до php 7

Сразу расскажу о проблемах, с которыми вы столкнетесь после обновления php70.

1. У вас не будет работать phpmyadmin без танцев с бубном. После обновления панель управления сразу перестала работать. Я немного погуглил тему, заставить работать можно, но нужно поковыряться. Мне не захотелось этого делать, подробно не разбирался в теме.

2. У меня перестал работать некоторый функционал плагинов в админке wordpress. При этом сам вордпресс работает нормально, его похоже оптимизировали для работы с php70. Получилось, что сайт в целом работает, но некоторые плагины не работают, либо ими нельзя управлять. WP Super Cache вообще не заработал, пока его не отключил, не мог загрузить главную страницу сайта, вместо нее белое полотно. Панель управления моей темой не открывала некоторые страницы с настройками.

Это то, что я заметил сам. Возможно не работает что-то еще. Все это я узнал постфактум, так что обновиться до php70 и прогнать тесты производительности успел.

Теперь информация об обновлении. Существуют как минимум 2 репозитория, которые можно подключить к CentOS 7 и установить обновление php70. Это либо **ius** с пакетом php70u, либо **webtactic** с php70w. Чем они отличаются я не знаю, не стал вникать. Я решил воспользоваться репозиторием ius. Подключаем его:

```
# curl 'https://setup.ius.io/' -o setup-ius.sh
# bash setup-ius.sh
```

Скрипт подключит нужное репо в соответствии с вашей системой. Теперь можно удалять старую версию php и устанавливать php70.

Дальнейшие действия будут зависеть от того, что вы используете на вашем веб сервере. У меня установлен nginx + php-fpm примерно по приведенной статье. Мне необходимо удалить пакеты:

```
# yum remove php-fpm php-cli php-common
```

Удаление этих пакетов тянет за собой удаление всех зависимостей. Запишите их куда-нибудь, чтобы потом установить новые версии этих пакетов. В качестве пакета к удалению будет в том числе и phpmyadmin. Впоследствии его можно будет установить только вручную из исходников. Если вы используете apache, то необходимо удалить **mod_php**, а затем заново установить **mod_php70u**.

Устанавливаем php70 вместе с необходимыми пакетами. У меня получился такой список. В нем оказалось не все, что было удалено, пары пакетов я не нашел в новом репо.

```
yum install php70u-common php70u-fpm-nginx php70u-pecl-apcu php70u-bcmath php70u-devel php70u-fpm php70u-gd php70u-ldap
php70u-mbstring php70u-mcrypt php70u-odbc php70u-pdo php70u-pear php70u-pecl-memcache php70u-process php70u-snmp php70u-
soap php70u-tidy php70u-xml php70u-xmlrpc php70u-cli php70u-mysqlnd
```

Я точно не помню, но скорее всего этот список соответствует требованиям wordpress и phpmyadmin. Больше у меня на сервере ничего не было, поэтому лишних пакетов быть не должно. После установки нужно чуть-чуть отредактировать конфигурацию php-fpm.

Открываем на редактирование */etc/php-fpm.d/www.conf* и добавляем туда параметр:

```
# mcedit /etc/php-fpm.d/www.conf
```

```
listen.acl_users = nginx
```

Если в качестве подключения к php-fpm использовали не unix socket, то придется перейти на него. Для этого закомментируйте строку:

```
listen = 127.0.0.1:9000
```

и добавьте новую:

```
listen = /var/run/php-fpm/php5-fpm.sock
```

Сохраняем конфиг и перезапускаем php-fpm:

```
# systemctl restart php-fpm
```

Если вы использовали unix socket, то в конфиге nginx ничего менять не надо, если же TCP socket, то нужно заменить строку:

```
fastcgi_pass 127.0.0.1:9000
```

на

```
fastcgi_pass /var/run/php-fpm/php5-fpm.sock
```

После этого перезапустите nginx:

```
# systemctl restart nginx
```

Обновление php до версии 7.0 окончено. Можно проверять вывод phpinfo();

Подключение модулей кэширования и тестирование производительности web сервера

PHP обновили, сайт запустился, дальше можно погонять тесты. Цифры я приводить не буду, так как в них нет никакого смысла. Они зависят от огромного числа параметров, поэтому в абсолютном значении не представляют ценности. Важны именно изменения значений в рамках одной тестируемой среды. Я буду говорить о примерном результате.

Первым делом я запустил тесты голого php70, без кэширования. Результаты при средней нагрузке, когда сервер успевает обработать все запросы, но работает на пределе своих возможностей, примерно оказались равны php54+apc. Но когда нагрузка сильно возрастает, образуется очередь запросов, php70 начинает в 2-3 раза медленнее обслуживать запросы, время отклика вырастает в 2-3 раза.

Я так прикинул, думаю, вроде неплохой результат. Сейчас включу apc и замерю как с ним будет. Оказалось, что модуль **apc** давно не поддерживается и поставить его на версию выше php54 нельзя. Вместо него теперь **apcu**. Думаю ладно, не проблема. Подключаю apcu и тестирую с ним. Результат меня расстроил. На средней нагрузке результат практически не изменился, на высокой нагрузке стал чуть хуже, а на очень высокой вообще в 2 раза просел по сравнению с работой без модуля.

Я понял, что никакого чуда с обновлением php70 не произошло. Прироста производительности я не получил, а получил кучу проблем в виде неработающих плагинов и phpmyadmin. Я принял решение откатываться назад, но не на версию php54, как было, а решил попробовать php56, чтобы проверить, что у него со скоростью.

К сожалению, уже после удаления 7-й версии php, я узнал, что модуль apc и apcu имеют принципиальное отличие и сравнивать только их нельзя. В результате мои тесты оказались недостоверны и с практической точки зрения бесполезны. Дело в том, что **apc** является *opcode cache and data store*, а **apcu** только *data store*. Таким образом, чтобы корректно протестировать производительность, мне нужно было в php70 включить еще **opcache**, который является *opcode cache*. Такая связка показала бы сопоставимый результат.

Мне все же любопытно проверить реальную производительность php70 в рабочей обстановке. Но постоянно пользоваться им пока не представляется возможным из-за проблем совместимости.

Откат обновления php 7.0 до php 5.6

Я решил откатиться на версию php 5.6. Ничего сложного в этом нет. Я уже рассказывал ранее, как в centos обновить php54 до php56. Воспользуемся информацией из этого материала. Сначала удаляем php70:

```
yum remove php70u-bcmath php70u-devel php70u-fpm php70u-gd php70u-ldap php70u-mbstring php70u-mcrypt php70u-odbc php70u-pdo php70u-pear php70u-pecl-memcache php70u-process php70u-snmp php70u-soap php70u-tidy php70u-xml php70u-xmlrpc php70u-cli php70u-mysqlnd php70u-fpm-nginx php70u-pecl-apcu php70u-common
```

Подключаем remi репозиторий:

```
# wget http://rpms.remirepo.net/enterprise/remi-release-7.rpm
```

```
# rpm -Uvh remi-release-7*.rpm
```

И устанавливаем все те же пакеты, что мы до этого удалили из версии php54, потом поставили и удалили php70 :)

```
yum --enablerepo=remi,remi-php56 install php php-common php-bcmath php-devel php-fpm php-gd php-ldap php-mbstring php-mcrypt php-mysql php-odbc php-pdo php-pear php-pecl-memcache php-php-gettext php-process php-snmp php-soap php-tcpdf php-tcpdf-dejavu-sans-fonts php-tidy php-xml php-xmlrpc php-pecl-apcu phpMyAdmin
```

Перезапускаем php-fpm. Он может ругнуться на строку:

```
listen.acl_users = nginx
```

Если так, то удалите ее. Я не помню, в какой версии она появляется, в 5.6 или в 7.0, в 5.4 ее точно не должно быть.

После отката на php5.6 я подключил модуль apcu и начал гонять тесты. Думаю и так понятно, что они все были хуже, чем php54+apc, так как принципы работы apc и apcu разные. Так что не буду останавливаться на этом. Жаль, что узнал об этом отличии я слишком поздно, когда уже вернулся обратно на php54 и стал спокойно разбираться в ситуации.

Я принял решение откатиться с версии php56 обратно на php54.

Отмена обновления php 5.6 и возврат на php 5.4

Тут все просто. Удаляем php56:

```
yum remove php php-common php-bcmath php-devel php-fpm php-gd php-ldap php-mbstring php-mcrypt php-mysql php-odbc php-pdo php-pear php-pecl-memcache php-php-gettext php-process php-snmp php-soap php-tcpdf php-tcpdf-dejavu-sans-fonts php-tidy
```

```
php-xml php-xmlrpc php-pecl-apcu
```

И ставим php54:

```
yum install php php-common php-bcmath php-devel php-fpm php-gd php-ldap php-mbstring php-mcrypt php-mysql php-odbc php-pdo php-pear php-pecl-memcache php-php-gettext php-process php-snmp php-soap php-tcpdf php-tcpdf-dejavu-sans-fonts php-tidy php-xml php-xmlrpc phpMyAdmin
```

Подключаем и настраиваем apc как описано в моей статье и проводим тесты. Убеждаемся, что производительность вернулась на прежний уровень.

Заключение

Устроил марафон на своем сервере с сайтом. Результат практически нулевой, за исключением полученных знаний. Так как раньше я вообще был не в теме всех нюансов настройки php и веб сервера, получил некоторое представление о том, как все это работает и в какую сторону нужно двигаться для увеличения производительности. Запланировал на тестовом сервере спокойно разобрать этот вопрос и протестировать производительность wordpress.

Подозреваю, что на слабеньких VDS с небольшой памятью и одним процом большого смысла городить дополнительные модули, которые тратят и так маленькие ресурсы сервера, не нужно. Существенного прироста производительности не будет. У меня все всегда упиралось в процессор при высоких нагрузках. Если нужно быстро ускорить wordpress в разы, то достаточно просто включить какой-нибудь кэширующий плагин, который генерирует статические страницы и выдает их пользователям. Прирост сразу же в десятки и сотни раз. Статический контент nginx отдает моментально и даже слабый VDS самого начального уровня способен будет обрабатывать одновременно сотни запросов.

С подобным кэшированием будут вопросы другого рода. Так как людям отдается статика, будут проблемы с комментированием, с интерактивными плагинами. Банальный счетчик просмотров работать не будет. Все эти вопросы можно решить, и некоторые решают, но тут уже нужно подключать специалистов, либо самому разбираться. Это сложные вопросы и решить их просто погуглив не получится. Готовых рецептов нет.

Вот часть таблицы с моими результатами по тесту с одним из наборов параметров. Просто для оценки разницы значений. Первый столбец то, что было в самом начале. Предпоследний - то, к чему в итоге вернулся после нескольких итераций. Последний столбец для наглядности, чтобы было понятно, что такое плагин кэширования для wordpress и как он влияет на время отклика.

	-c 5 -r 100	-c 5 -r 100	-c 5 -r 100	-c 5 -r 100	-c 5 -r 100	-c 5 -r 100	-c 5 -r 100
Availability	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
Elapsed time	254.47 secs	250.00 secs	267.16 secs	345.11 secs	384.56 secs	254.23 secs	162.83 secs
Response time	0.96 secs	0.95 secs	0.94 secs	1.95 secs	2.16 secs	0.92 secs	0.03 secs
Transaction rate	1.96 trans/sec	2.00 trans/sec	1.87 trans/sec	1.45 trans/sec	1.30 trans/sec	1.97 trans/sec	3.07 trans/sec
Throughput	0.04 MB/sec	0.04 MB/sec	0.03 MB/sec	0.03 MB/sec	0.02 MB/sec	0.03 MB/sec	0.05 MB/sec
Concurrency	1.88	1.90	1.75	2.83	2.81	1.81	0.09
Longest transaction	1.93	2.03	3.04	3.39	4.52	1.85	0.07
Shortest transaction	0.36	0.35	0.38	0.59	0.58	0.35	0.01
	php54+apc	php70 без apc	php70+apcu	php5.6+apcu	php5.4+apcu	php5.4+apc	php5.4+apc
							WP Supre cache

Буду рад комментариям, замечаниям и советам по теме текущей статьи. У меня нет сейчас достаточно времени, чтобы разобраться в нюансах, но есть желание научиться настраивать максимально производительный web сервер для базовых задач размещения популярных движков. Время отклика сайта сейчас влияет на ранжирование сайта в результатах поиска. По крайней мере гугл открыто об этом говорит и призывает всех делать быстрые сайты.

Помогла статья? Подписывайся на telegram канал автора

Анонсы всех статей, плюс много другой полезной и интересной информации, которая не попадает на сайт.