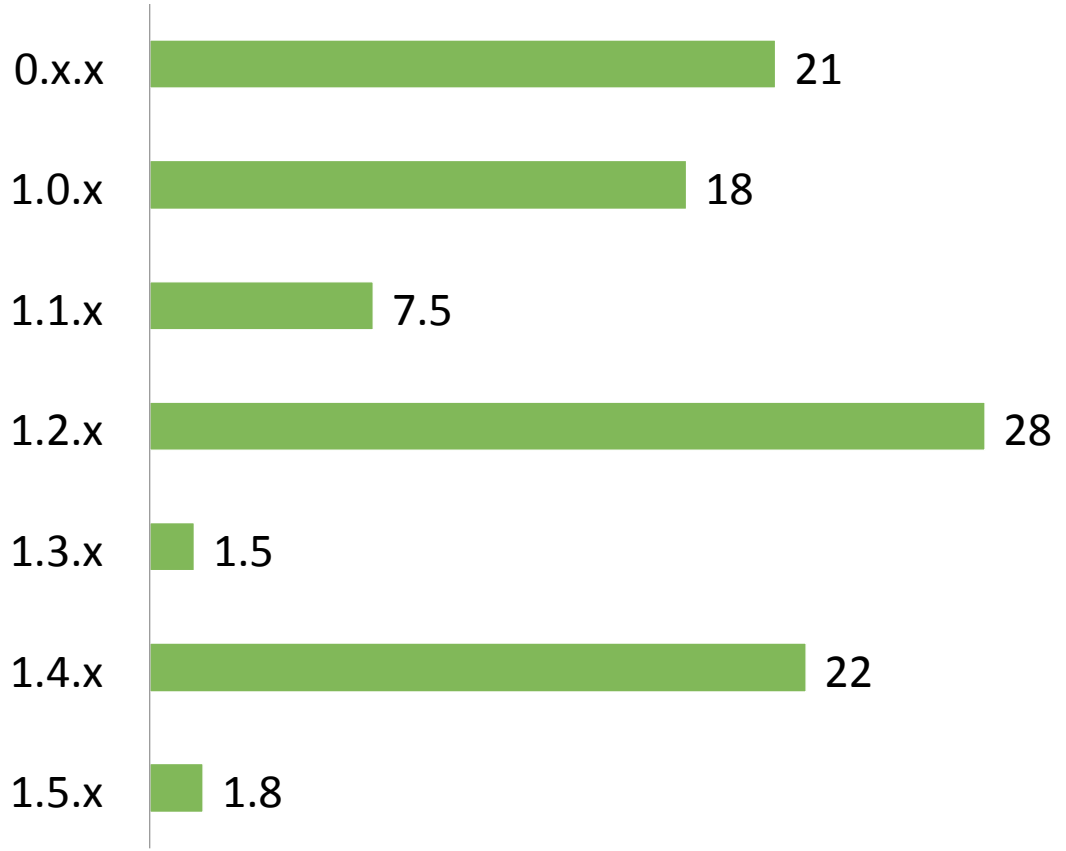
Что нового в nginx?

Максим Дунин

highload++ 2013
Конференция разработчиков высоконагруженных систем

Про ложь, наглую ложь и статистику

- nginx 0.x.x: более 20% (данные w3techs.com, openstat.ru)
- Debian Squeeze (oldstable): nginx 0.7.67



Немного о версионировании

- 0.x.x: давно и неправда
- 1.0.x, 1.2.x, 1.4.x, …
  стабильные версии, стабильное API, критические и security-исправления
- 1.1.x, 1.3.x, 1.5.x, …
  основные версии (mainline), новая функциональность тут, production ready

Актуально: 1.5.6, 1.4.3
Всё остальное - использовать не рекомендуется.

Новое в nginx 1.1.x

Улучшения cache loader, keepalive с бекендами, поддержка криптографии на эллиптических кривых (привет, NSA!), оптимизация потребления памяти SSL-соединениями, применение нескольких limit_conn и limit_req одновременно, MP4 streaming в коробке, proxy_cache_lock, регулярные выражения в proxy_redirect, disable_symlinks для shared-хостинга, а также поддержка PCRE JIT для тех, кто любит регулярные выражения.

Новое в nginx 1.1.x

- Улучшения cache loader
- Keepalive с бекендами
- Поддержка криптографии на эллиптических кривых (ECDHE)
- Оптимизация потребления памяти SSL-соединениями
- Применение нескольких `limit_conn` и `limit_req` одновременно
- MP4 streaming в коробке
- `proxy_cache_lock`
- Регулярные выражения в `proxy_redirect`
- `disable_symlinks` для shared-хостинга
- Поддержка PCRE JIT

Улучшения cache loader

- Теперь nginx обходится данными от readdir(),
  не пытаясь читать файлы кеша
- Алгоритм хорошо работает с SSD
- Гибкая конфигурация загрузки:
  `loader_files`, `loader_sleep`, `loader_threshold`
- Кеш 6 млн. файлов / 400 Гб загружается за 50 минут без потери производительности сервера (ранее – 2 суток)

Keepalive с бекендами

- Поддержка постоянных соединений с бекендами
- Работает с:
  - proxy_pass
  - fastcgi_pass
  - memcached_pass

Keepalive с бекендами

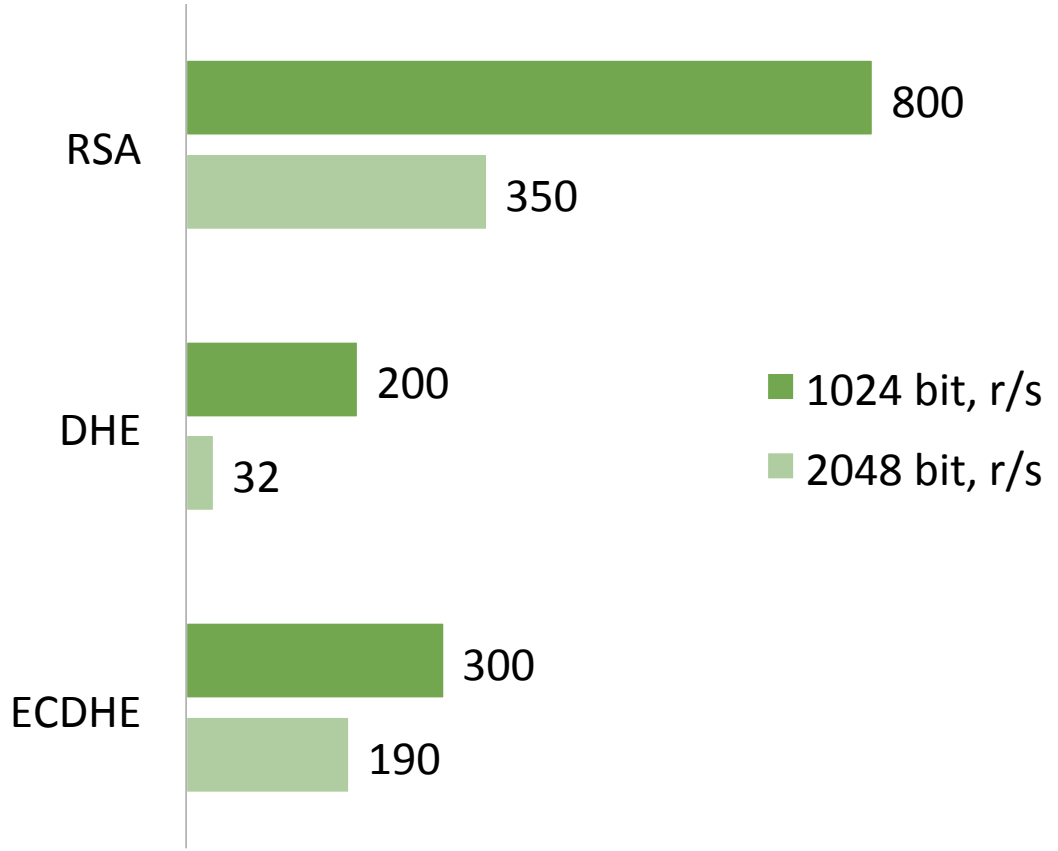
Пример:

```
upstream memd {
    server 127.0.0.1:11211;
    keepalive 42;
}
location / {
    memcached_pass memd;
}
```

Elliptic curve Diffie–Hellman (ECDH)

Forward secrecy
по разумной цене

- Нюанс: нужно собрать с правильным OpenSSL
- Lavabit court order
- Bruce Schneier recommendations



Память и SSL-соединения

Было:

- 64k на соединение
- или даже 600k на соединение (сжатие!)

Стало:

- 16k на соединение

(В nginx 1.3.2+ - сжатие запрещено во всех версиях OpenSSL, включая 0.9.x, где отсутствует соответствующая опция.)

Несколько limit_conn одновременно

Ограничить количество соединений с одного IP-адреса и количество соединений к одному серверу:

```
limit_conn_zone $binary_remote_addr zone=conn_addr:10m;
limit_conn_zone $server_name        zone=conn_name:10m;

server {
    ...
    limit_conn addr 5;
    limit_conn name 10;
    ...
}
```

Несколько limit_req одновременно

Ограничить количество запросов в секунду с одного IP-адреса и количество запросов в секунду к одному серверу:

```
limit_req_zone $binary_remote_addr zone=req_addr:10m rate=1r/s;
limit_req_zone $server_name        zone=req_name:10m rate=5r/s;

server {
    ...
    limit_req zone=addr burst=5 nodelay;
    limit_req zone=name burst=10 nodelay;
    ...
}
```

MP4 http pseudo streaming в коробке

Пример:

```
location /mp4/ {
    mp4;
}
```

proxy_cache_lock

Пример:

```
location / {
    proxy_pass http://backend;
    proxy_cache one;
    proxy_cache_lock on;
    proxy_cache_use_stale updating;
}
```

Если более одного запроса хотят загрузить в кеш новый документ – только один уйдёт на бекенд, остальные запросы будут ждать.

Регулярные выражения в proxy_redirect

- Директиву proxy_redirect теперь можно использовать с регулярными выражениями:

```
proxy_redirect ~/user/([^/]+)/(.+)$
               http://$1.example.com/$2;
```

- Директивы proxy_cookie_path, proxy_cookie_domain для изменения кук при проксировании:

```
proxy_cookie_domain localhost example.org;
proxy_cookie_path   /two/     /;
```

Директива disable_symlinks

Пример:

```
disable_symlinks on;
```

Или, более гуманно, но дороже:

```
disable_symlinks if_not_owner;
```

Нюанс: работает это только на современных операционных системах, поддерживающих openat(). Без этого сделать соответствующую проверку без race condition'а нельзя.

PCRE JIT

Пример:

```
pcre_jit on;
```

Использовать с осторожностью - *замедляет* на некоторых нагрузках.

Пример:

```
rewrite /((ab?)*)*bb /foo;
```

Включение PCRE JIT ускоряет обработку запросов к “/aaaaaaaaaaaaaaaaaaaab” в 10 раз.

Новое в nginx 1.3.x

Улучшения в поддержке IPv6, балансировщик least_conn, поддержка ETag (а значит, докачка в IE9+), gunzip и возможность хранить ресурсы сжатыми, OCSP Stapling, SPDY, поддержка передачи тела запроса chunk'ами, проксирование WebSocket'ов и возможность писать логи уже сжатыми.

Новое в nginx 1.3.x

- Улучшения в поддержке IPv6
- Балансировщик least_conn
- Поддержка ETag (а значит, докачка в IE9+)
- Фильтр gunzip и возможность хранить ресурсы сжатыми
- OCSP Stapling
- SPDY
- Поддержка передачи тела запроса chunk'ами
- Возможность писать логи уже сжатыми
- Проксирование WebSocket'ов

Улучшения в поддержке IPv6

IPv6 поддерживается практически везде:

- Geo, geoip, бекенды, access-проверки
  и т.п.

Одно из немногих исключений - встроенный resolver. Мы работаем над этим.

Балансировщик least_conn

Пример:

```
upstream u {
    least_conn;
    server 192.0.2.1;
    server 192.0.2.2;
}
```

- Выбирается бекенд, к которому установлено меньше всего соединений
- Удобно для балансировки бекендов, где соединения == загрузка.
  Т.е. почти всегда.

Поддержка ETag

- Докачка в IE9+
- Обработка запросов из кеша
- ETag статических файлов не включает номер i-node, нет проблемы с раздачей файлов с нескольких серверов (как ранее в Apache)

Gunzip: храним ресурсы сжатыми

- Разжимаем сжатые ответы на лету – если клиент не понимает gzip
- Понимают gzip - более 95%

Gunzip: храним ресурсы сжатыми

Экономим немного трафика к бекендам:

```
location /proxy/ {
    gunzip on;
    proxy_pass ...
    proxy_set_header Accept-Encoding gzip;
}
```

Gunzip: храним ресурсы сжатыми

Экономим память в memcached:

```
location /memd/ {
    gunzip on;
    memcached_gzip_flag 2;
    memcached_pass ...
}
```

Gunzip: храним ресурсы сжатыми

Экономим диск и память под файловый кеш:

```
location /store/ {
    gunzip on;
    gzip_static always;
}
```

OCSP Stapling

- OCSP – Online Certificate Status Protocol
- Прикрепление свежего OCSP-ответа к сертификату, отправляемому при SSL handshake
- Экономим ресурсы клиентов

OCSP Stapling

Пример:

```
server {
    listen 443 ssl;

    ssl_certificate foo.crt;
    ssl_certificate_key foo.key;

    ssl_stapling on;

    resolver 192.0.2.1;
}
```

SPDY

Пример:

```
server {
    listen 443 ssl spdy;
    ...
}
```

Экспериментальный модуль для работы по экспериментальному протоколу. Можно использовать без SSL, но не с браузерами.

Chunked request body

- Ранее не было – потому что почти не надо, хотя HTTP/1.1 требует
- Используется WebDAV-клиентами, Java-приложениями
- Позволяет делать так:

```
$ ./script.pl |
  curl --upload-file - http://example.com/
```

без временных файлов.

Возможность писать логи уже сжатыми

Пример:

```
access_log /path/to/access.log.gz combined gzip;
```

Дополнительные параметры:

```
buffer=64k
flush=1s
```

Занимает мало памяти и очень немного процессора.
Добавляет счастья.

Проксирование WebSocket'ов

Пример:

```
location /chat/ {
    proxy_pass …
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

Нюанс: WebSocket'ы используют механизм Upgrade в HTTP/1.1, и он не рассчитан на работу через прокси. В случае forward proxy спецификация WebSocket'ов предлагает использовать CONNECT, а для reverse proxy - приходится явно писать конфигурацию.

Новое в nginx 1.5.x (still counting)

Поддержка EPOLLRDHUP на Linux, использование O_PATH для disable_symlinks на Linux, несколько директив error_log одновременно, SMTP pipelining, очередные оптимизации SSL, proxy_ssl_protocols и proxy_ssl_ciphers, модуль auth request, небуферизированная работа с FastCGI-бекендами (fastcgi_buffering) для тех, кому нужен streaming ответов.

Новое в nginx 1.5.x (still counting)

- Поддержка EPOLLRDHUP на Linux
- Использование O_PATH для disable_symlinks на Linux
- Несколько директив error_log одновременно
- SMTP pipelining
- Очередные оптимизации SSL
- Директивы proxy_ssl_protocols и proxy_ssl_ciphers
- Модуль auth request
- Небуферизированная работа с FastCGI-бекендами (fastcgi_buffering) для тех, кому нужен streaming ответов

Поддержка EPOLLRDHUP на Linux

Выясняем, закрыл ли клиент соединение:

- Универсальный метод - recv(MSG_PEEK)
  - Нет информации, если присутствуют непрочитанные данные
- Платформо-специфичные методы:
  - kqueue – EV_EOF
  - epoll – EPOLLRDHUP
- Помогает при long polling

O_PATH для disable_symlinks на Linux

- Linux не умеет O_SEARCH
- При использовании disable_symlinks
  на промежуточных каталогах нужны права **r-x**
- Но в 2.6.39+ есть O_PATH, очень похожий
- Достаточно прав **--x**

SMTP pipelining

- И разные другие мелкие улучшения почтового прокси-сервера
- Имеет смысл обратить внимание, если вы используете nginx в качестве почтового прокси-сервера

Очередные оптимизации SSL

- Убран лишний round-trip при использовании длинных цепочек сертификатов
- Просто перестало хватать 4k буфера, используемого OpenSSL для буферизации записи во время handshake’а

Директивы proxy_ssl_protocols, proxy_ssl_ciphers

Позволяют:

- Улучшить совместимость с различными HTTPS-бекендами
- Ускорить работу
- Потребовать forward secrecy

Модуль auth request

- Аутентификация и авторизация через подзапрос, в том числе – внешними средствами
  - Если подзапрос вернул 200 – доступ разрешён, иначе – нет
- Что-то вроде fastcgi authorizers
- Удобнее, чем `X-Accel-Redirect`
- Лучше, чем блокирующиеся сторонние модули auth_pam и auth_ldap

Модуль auth request

Например, так:

```
location / {
    auth_request /auth;
}

location = /auth {
    proxy_pass ...
}
```

Небуферизированная работа с FastCGI-бекендами

Пример:

```
fastcgi_buffering off;
```

Или даже так, в заголовках ответа FastCGI-приложения:

```
Status: 200 OK
X-Accel-Buffering: no
...
```

Для тех, кому нужен streaming ответов или прогрессивная загрузка страниц.

Где взять свежий nginx?

- Исходные коды, как обычно:
  http://nginx.org/ru/download.html

- Для FreeBSD:
  используйте порты nginx и nginx-devel, там всё новое
  Спасибо Сергею Осокину!

- Для Linux разных версий:
  системные пакеты - обычно старые, свежие пакеты доступны на
  http://nginx.org/ru/linux_packages.html

Вопросы?

Максим Дунин

mdounin@mdounin.ru

NGINX Plus

Коммерческая версия.

Дополнительная функциональность:

- Advanced load balancing:
  - Dynamic upstream reconfiguration, active health checks, sticky balancing
- Advanced video streaming:
  - HTTP Live streaming
  - HTTP Dynamic streaming
- Advanced logging:
  - Remote logging via syslog protocol, session logging
- Status module