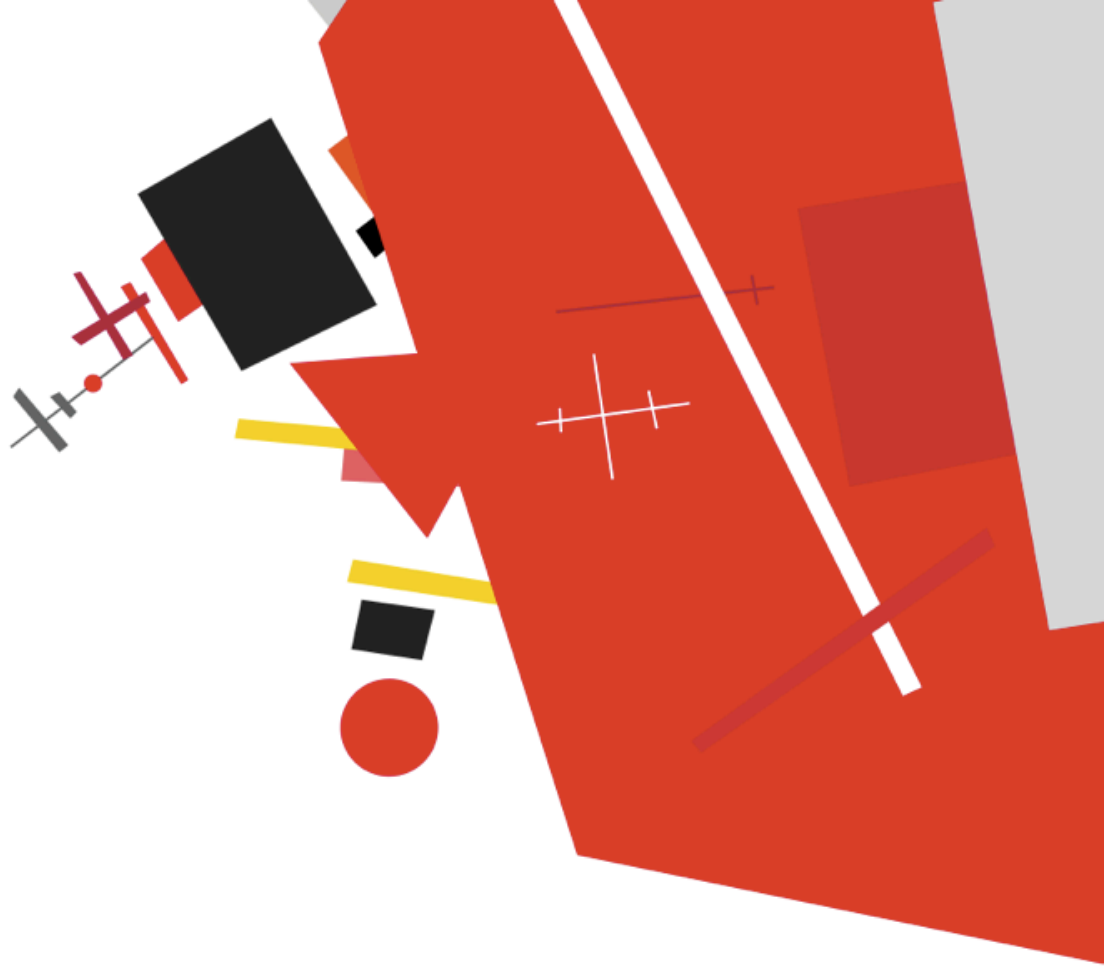
Что нового в nginx?

Максим Дунин



HighLoad++

Профессиональная конференция разработчиков высоконагруженных систем

Про ложь, наглую ложь и статистику

W3Techs:

```
0.x.x -  1.5%
1.x.x - 98.5%
```

https://w3techs.com/technologies/details/ws-nginx/all/all
“Nginx is used by 35.7% of all the websites whose web server we know.”

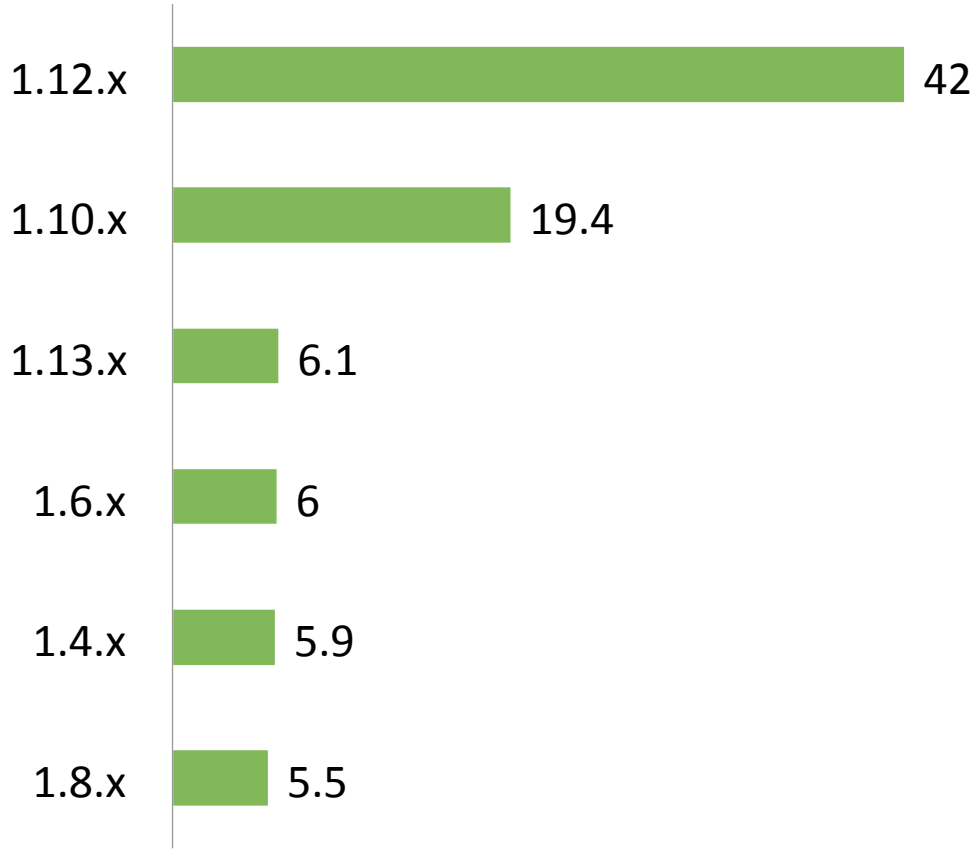
Подробнее про 1.x.x

Новое:

```
1.12.x  - 42.0%
1.13.x  -  6.1%
```

Старое:

```
1.10.x  - 19.4%
1.8.x   -  5.5%
1.6.x   -  6.0%
1.4.x   -  5.9%
```



1.11.x

Переменные в модуле stream, возможность "подсматривать" в SSL handshake, и множество всего ещё. Больше нет quick recovery, но есть max_conns, а cache manager научился быть незаметным, proxy_bind теперь умеет transparent, и умеет использовать IP_BIND_ADDRESS_NO_PORT. Мы выключили accept_mutex, научились использовать EPOLLEXCLUSIVE вместо него, и начали полагаться на EPOLLRDHUP, map научился сложным значениям, переменная $request_id для злопамятных. Теперь можно несколько SSL-сертификатов (наконец-то RSA и ECDSA одновременно!), умеем списки кривых для ECDH, выключили DH, упростили сборку динамических модулей. Умеем писать логи в JSON (если пришлёте тягач), а для старых рабочих процессов - добавили возможность задать таймаут на завершение. Кэш теперь понимает "Cache-Control: stale-while-revalidate=300" в ответах и умеет обновляться в фоне, а перенаправления теперь не обязательно абсолютные.

Stream: балансировка соединений

```
stream {
    upstream backend {
        server 10.0.0.2:12345;
        server 10.0.0.3:12345;
    }
    server {
        listen 12345;
        proxy_pass backend;
    }
}
```

- 1.9.x
- Балансировка произвольных соединений
- SSL от клиентов, SSL к бэкендам
- Ограничения количества соединений и скорости
- PROXY protocol к бэкендам
- … и даже UDP

Stream: переменные

А также map, return, geo, geoip, split_clients, real ip, access log…

Теперь можно:

```
map $remote_addr $limit {
    127.0.0.1     "";
    default       $binary_remote_addr;
}

limit_conn_zone $limit zone=addr:10m;
limit_conn addr 1;
```

Stream: клиентские SSL-сертификаты

Теперь проверять клиентские сертификаты умеет и stream:

```
ssl_verify_client on;
ssl_client_certificate /path/to/root;
```

Не прошедшие проверку соединения будут закрыты.
Результат проверки в `$ssl_client_verify`.

Stream: $ssl_preread_server_name

Если нужно посмотреть в SSL-соединение,
но нельзя снимать SSL:

```
map $ssl_preread_server_name $backend {
    backend.example.com      192.0.2.1:12345;
    default                  192.0.2.2:12345;
}

server {
    listen       12345;
    proxy_pass   $backend;
    ssl_preread on;
}
```

Количество соединений к бэкендам

Пример:

```
upstream u {
    server 192.0.2.1:8080 max_conns=10;
    server 192.0.2.2:8080 max_conns=10;
    server 192.0.2.3:8080 max_conns=10;
}
```

Исходно для NGINX Plus.

Transparent proxy

Передать адрес клиента, традиционные решения:

- X-Forwarded-For / X-Real-IP
- PROXY protocol

Если очень надо:

```
proxy_bind $remote_addr transparent;
```

SO_BINDANY (NetBSD), IP_TRANSPARENT (Linux), IP_BINDANY (FreeBSD), нужен root

Linux и IP_BIND_ADDRESS_NO_PORT

Сколько соединений можно установить к одному бэкенду?
К двум?

А теперь пишем в конфиг:

```
proxy_bind 192.0.2.1;
```

и всё плохо. Почему?

- Bind() выбирает локальный порт
- IP_BIND_ADDRESS_NO_PORT (Linux 4.2, glibc 2.22)

Accept mutex, Linux и EPOLLEXCLUSIVE

Thundering herd problem.
Традиционный ответ – accept mutex.
Но:

- вообще-то у нас процессов мало
- часто мешает в тестах
- нельзя использовать на Windows
- не нужен с “listen … reuseport”
- на Linux появился EPOLLEXCLUSIVE (Linux 4.5, glibc 2.24)

Accept mutex, Linux и EPOLLEXCLUSIVE

Новый default:

```
accept_mutex off;
```

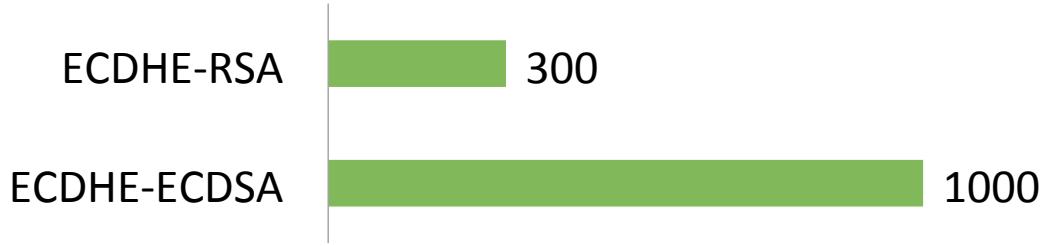
Ну и EPOLLEXCLUSIVE поддерживаем.

SSL и сертификаты

RSA – совместимо, но дорого

ECDSA – быстро, но не будет работать в XP (Android до 4.0, OpenSSL до 1.0.0)

```
rsa2048:     400 signs/s
ecdsap256:  5400 signs/s
```



Теперь можно так:

```
ssl_certificate     example.com.rsa.crt;
ssl_certificate_key example.com.rsa.key;
ssl_certificate     example.com.ecdsa.crt;
ssl_certificate_key example.com.ecdsa.key;
```

SSL и переменные

- `$ssl_ciphers`
  - “AES128-SHA:AES256-SHA:0x00ff”
- `$ssl_curves`
  - “0x001d:prime256v1:secp521r1:secp384r1”
- `$ssl_client_verify` теперь содержит текст ошибки
  - “FAILED:certificate has expired”
- `$ssl_client_s_dn`, `$ssl_client_i_dn` – теперь в формате RFC 4514
  - `$ssl_client_s_dn_legacy`, `$ssl_client_i_dn_legacy`
- `$ssl_client_v_start`, `$ssl_client_v_end`
  - “May 6 01:20:48 2017 GMT”
- `$ssl_client_v_remain`
  - “12”

SSL и другое

- `ssl_ecdh_curve` умеет список кривых (OpenSSL 1.0.2+)
  - “auto” по умолчанию
  - OpenSSL 1.1.0: X25519:prime256v1:secp521r1:secp384r1
- `ssl_dhparam` по умолчанию отсутствует
  - То есть DHE-шифры выключены
  - Медленно
  - Атаки на DHE, weakdh.org
- `ssl_session_ticket_key` умеет AES256
  - Нужно использовать файлы размером 80 байт вместо 48

Динамические модули

Появились в 1.9.x.
Собрать:

```
./configure --add-dynamic-module=/path/to/module
make modules
```

Загрузить в nginx.conf:

```
load_module modules/ngx_my_cool_module.so;
```

Динамические модули: совместимость

Проблема:

- Структуры зависят от опций сборки
- Собирать модули надо с теми же опциями

Теперь проще:

```
./configure --with-compat ...
./configure --with-compat --add-dynamic-module=/path/to/module
```

Бонус: совместимость с NGINX Plus.

Завершение старых рабочих процессов

Проблема:

- При обновлении конфигурации старые рабочие процессы завершаются, когда все соединения будут закрыты
- WebSocket’ы, long-polling, stream – долго

Теперь можно так:

```
worker_shutdown_timeout 15m;
```

Cache: устаревшие ответы

Было:

```
proxy_cache_use_stale updating;
```

- Только из конфигурации
- Может использоваться очень старый ответ

Cache: устаревшие ответы

Теперь можно управлять заголовком `Cache-Control`:

```
HTTP/1.1 200 OK
Cache-Control: stale-while-revalidate=300
```

- RFC 5861
- Можно управлять с бэкенда
- Задаётся время, после которого ответ непригоден
- Для ошибок – `stale-if-error`

Cache: обновление в фоне

Если используется “`proxy_cache_use_stale updating`” или `stale-while-revalidate`:

- Почти все получают ответ из кэша
- Один клиент ждёт бэкенда

```
proxy_cache_background_update on;
```

- Обновление в подзапросе
- Одновременно с возвратом ответа клиенту из кэша
- Если не успеет, то задержит следующие запросы в соединении

Cache: разное

- Поменяли “`proxy_cache_path ... use_temp_path=off`”
  - Появилось в 1.7.10 вместе с переменными в `proxy_cache`, переключаться между кэшами
  - Теперь временные файлы создаются в том же каталоге, что и целевой файл
  - Иначе lock contention на Linux: `rename()` между каталогами берёт lock на всю файловую систему
- Директива `proxy_cache_max_range_offset`
  - Обычно скачиваем в кэш файл целиком, и отдаём range из него
  - Запрет использования кэша для «далёких» range-запросов

Относительные перенаправления

- RFC 2616: только абсолютные URI в `Location`
  - `Location = "Location" ":" absoluteURI`
  - При необходимости nginx сам строит абсолютный URI
    - `server_name_in_redirect`, `port_in_redirect`
  - Но: относительные перенаправления обычно работают
- RFC 7231: можно относительные
  - `Location = URI-reference`
- Директива `absolute_redirect`

Логгирование в JSON

```
log_format json_combined
    '{ "time_local": "$time_local", '
    '"remote_addr": "$remote_addr”, '
    '"remote_user": "$remote_user", '
    '"request": "$request", '
    '"status": "$status", ’
    '"body_bytes_sent": "$body_bytes_sent", '
    '"request_time": "$request_time", '
    '"http_referrer": "$http_referer", '
    '"http_user_agent": "$http_user_agent" }';
```

Логгирование в JSON

```
log_format json_combined escape=json
    '{ "time_local": "$time_local", '
    '"remote_addr": "$remote_addr”, '
    '"remote_user": "$remote_user", '
    '"request": "$request", '
    '"status": "$status", ’
    '"body_bytes_sent": "$body_bytes_sent", '
    '"request_time": "$request_time", '
    '"http_referrer": "$http_referer", '
    '"http_user_agent": "$http_user_agent" }';
```

Порт клиента

- `set_real_ip_from` теперь ставит не только адрес, но и порт

```
proxy_pass http://backend;
proxy_set_header X-Real-IP
                 $remote_addr:$remote_port;
```

- переменные `$proxy_protocol_port`, `$realip_remote_port`

Разное

- На EPOLLRDHUP мы теперь полагаемся, если ядро новое
- HTTP/2 – исправления и улучшения
- Map научился сложным комбинациям переменных и строк

```
map $host $log {
    ~foo      /var/log/$host.log;
    default   /var/log/default.log;
}
```

  и обзавёлся параметром `volatile`
- Поддержка WebP в `image_filter`
- `$request_id`

Всё это доступно в nginx 1.12.2

1.13.x

Модуль mirror для анализа трафика, базовая поддержка TLS 1.3, $ssl_client_escaped_cert, разрешили SSL renegotiation с бэкендами, в логах теперь есть PID процесса, отправившего сигнал, появилась директива add_trailer, а CPU affinity можно задавать на DragonFly BSD.

Mirror

Задача:

- Отправить запрос в дополнительное место
- Для анализа / сбора статистики / тестирования

Решение:

```
location / {
    mirror /mirror;
    proxy_pass http://real-backend;
}

location /mirror {
    proxy_pass http://mirror-backend;
    proxy_set_header X-Original-URI $request_uri;
}
```

Mirror: детали

- Работает с помощью background-подзапросов, как и обновление кэша
  - Переписали background-подзапросы
- Можно не читать тело
  - `mirror_request_body off;`
- Заодно образовалась precontent-фаза
  - Там try_files и mirror
  - Можно встраивать свои модули
- Вычистили проблему с ранним освобождением тела в proxy_pass
  - Вообще это оптимизация – зачем нам тело, если оно уже у бэкенда?
  - Можно было наступить в SSI

SSL: основы TLS 1.3

- Full handshake − 1 RTT
  - Не всегда, но обычно (если key_share клиента подходит)
  - Ранее full handshake − 2 RTT, abbreviated handshake − 1 RTT
- Режим 0-RTT
  - Но: без replay protection

- Стандарта ещё нет
- Middleboxes issue, 1.5% .. 3.4% error rate
- Datacenter monitoring - TEH DRAMMA

SSL: поддержка TLS 1.3

Теперь nginx понимает:

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
```

- По умолчанию выключено
- Нужен OpenSSL 1.1.1 (not yet released), собранный с “enable-tls1_3”
- В браузерах Draft 18, в OpenSSL – Draft 21, нужна ветка "tls1.3-draft-18”
- Не будет работать session reuse с бэкендами, поправим
- Пока нет поддержки 0-RTT, в планах

SSL: разное

- теперь можно SSL renegotiation к бэкендам
  - если ваш бэкенд очень хочет renegotiation
- переменная `$ssl_client_escaped_cert`

Разное

- В логах пишется PID процесса, отправившего сигнал
- Теперь `worker_cpu_affinity` работает на DragonFly BSD
- Базовая поддержка HTTP trailers и директива `add_trailer`
- Имена в `set_real_ip_from`
- В stream и mail – параметры `rcvbuf` и `sndbuf` у директивы `listen`

Всё это доступно в nginx 1.13.6

Still counting

Где взять дене^Wсвежий 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

  (отдельные пакеты для модулей geoip, image filter, njs, perl, xslt)

Спасибо!

Вопросы?

Максим Дунин
mdounin@mdounin.ru