Nginx

vysoký výkon pod zátěží

Petr Krčmář

7. listopadu 2020

O mně

- linuxák od roku 1998
- správce serverů
- lektor a konzultant
- šéfredaktor Root.cz
- člen vpsFree.cz
- organizátor LinuxDays
- můj web je petrkrcmar.cz

ROOT.CZ_ vpsFree.cz

Prezentace už teď na webu

https://www.petrkrcmar.cz

ROOT.CZ_ vpsFree.cz

Logo Nginx

Co je to Nginx

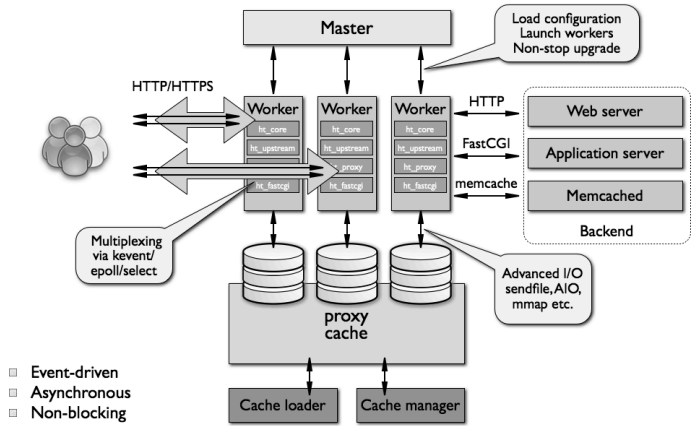
Co je to Nginx?

- asynchronní event-driven server
- (hlavně) web server (ale nejen)
  - kešující server
  - load balancer
- reverzní proxy pro HTTP/S (nejen)
  - SMTP, POP3, IMAP, HTTP cache
- často nasazován jako reverzní proxy
- téměř na polovině TOP1000 webů (W3Techs)

Event-driven?

- tradiční přístup: blokující fork pro každý požadavek
- velmi náročné na prostředky (paměť, přepínání kontextu)
- C10K (concurrently handling ten thousand connections)
- Nginx se neforkuje a je neblokující
- všechno vyřizují předem spuštěné workery
- malý (a předvídatelný) dopad na paměť
- server se neutaví ani při velkém provozu

Architektura Nginx



Obrázek: Architektura Nginx

Zásadní vlastnosti

- nízká náročnost na prostředky
- 10K požadavků potřebuje 2,5 MB paměti
- umí zpracovat velké množství požadavků
- je odolný proti DDoS útokům (Slowloris)
- velmi volná licence (dvouklauzulová BSD)
- multiplatformní: Linux, *BSD, Win, macOS, Solaris a další

Ladění výkonu

Hledání úzkého hrdla

1. poznejte dobře charakteristiku provozu
   - mnoho krátkých požadavků, málo dlouhých stahování...
2. identifikujte slabé místo – úzké hrdlo
   - velmi často není ve webserveru samotném
3. upravujte související konfigurační volby
   - vždy jen jednu, průběžně testujte výkon
4. průběžně studujte logy
   - nejen web serveru, ale i jádra a dalších komponent

- stav lze sledovat pasivně (logy, metriky)
- můžeme pouštět aktivní testy (například nástroj `wrk`)

ROOT.CZ_ vpsFree.cz

Co nejvíc vyřiďte na web serveru

- cílem je maximum požadavků vyřídit v Nginx
  - vyřešit brzy = vyřešit rychle bez ztráty výkonu
- odfiltrujte nežádoucí provoz
  - útočníky, agresivní klienty, boty…
- statické soubory servírujte rychle
  - nenechávejte to na webové aplikaci napsané v PHP
- kešujte maximum provozu, nezatěžujte backend
  - rychlé vyřízení požadavku ve web serveru

Jádro web serveru

Počet workerů

- výchozí počet je jedna = vytížíme jen jedno jádro
- rozumné je mít jeden proces na jádro (volba `auto`)
  - pokud ale workery dělají hodně I/O, chcete jich víc
- můžeme také ovlivnit maximální počet spojení na worker
  - výchozí je 512 (max. klientů = workery * spojení)
  - nejde jen o spojení na klienty!
  - počítají se do toho všechna spojení, včetně těch na backendy
  - pozor na max. počet otevřených souborů (`worker_rlimit_nofile`)

Nastavení workerů

```
worker_processes auto;
worker_rlimit_nofile 100000;
events {
   worker_connections 4000;
…
```

Keš file descriptorů

- keš file descriptorů, časů modifikací a velikostí
  - pamatuje si výsledky vyhledávání (i chyby) a existenci adresářů
- dokáže navýšit výkon při práci s mnoha soubory
- nastavujeme: kolik si pamatujeme, po jaké době vyhazujeme
  - dále po jaké době validovat a kolik je minimum pro inactive
  - můžeme zapnout kešování chyb hledání souborů

Nastavení keše

```
open_file_cache          max=1000 inactive=20s;
open_file_cache_valid    30s;
open_file_cache_min_uses 2;
open_file_cache_errors   on;
```

Logování

- zápisy do logů zatěžují I/O
- pokud nám chybí výkon I/O, musíme omezit
  - nemusíme vypínat úplně, lze logovat jen kritické věci
  - můžeme vypnout kešování v určitých situacích
  - podle location nebo na základě generování chyby
  - pokud někoho vyhazujeme na 403, nemusíme logovat
- můžeme zapnout write-buffer – každá jedna položka nevyvolá zápis

Nastavení logování

```
access_log off;
access_log /var/log/access.log gzip buffer=32k;
error_log /var/log/nginx/error.log crit;
```

Komprese

- pokud nás trápí šířka pásma a máme dost výkonu
- soubory můžeme staticky předkomprimovat (`.gz`)
  - zapneme hledání pomocí `gzip_static`
- můžeme ovlivňovat dynamickou kompresi
  - stupeň komprese, minimální velikost souboru, typ obsahu…
  - typ `text/html` je vždy komprimován
  - můžeme použít i znak * pro všechny typy

Nastavení komprese

```
gzip on;
gzip_min_length 16384;
gzip_comp_level 5;
gzip_types text/css text/javascript
           text/plain image/svg+xml;
```

Použití TCP_NODELAY

- Nagleho algoritmus řeší problém zahlcení sítě malými pakety
- snaží se posílat data po větších celcích
- unixová implementace čeká 200 ms, zda nedostane další data
- to je výhodné pro sporadicky posílané kousky dat
- dnes se posílají velké soubory a často streamy
- chceme ušetřit 200 ms a mít rychlejší odezvu
  - vypneme tedy Nagleho algoritmus na TCP socketu

Zapnutí tcp_nodelay

```
tcp_nodelay on;
```

ROOT.CZ_ vpsFree.cz

Použití volání sendfile

- jaderné volání `sendfile()` kopíruje data mezi deskriptory
- umožňuje zrušit zbytečné přepínání kontextu z jádra do procesu
- rychlejší než klasické použití `read()` a `write()`
- dává smysl jen u statických souborů, ne u proxy
- lze nastavit limit přenosu na jedno volání
  - aby jedno rychlé spojení nevytížilo worker

Zapnutí sendfile

```
sendfile on;
sendfile_max_chunk 1m;
```

ROOT.CZ_ vpsFree.cz

Použití tcp_nopush

- umožňuje lépe využívat zaplnění paketů
- v prvním paketu odpovědi pošle hlavičky a rovnou začátek souboru
- zapne na socketu `TCP_CORK` (Linux) nebo `TCP_NOPUSH` (FreeBSD)
- musí být kombinováno se `sendfile`

Zapnutí tcp_nopush

```
sendfile on;
tcp_nopush on;
```

ROOT.CZ_ vpsFree.cz

Rychlejší odpojení uživatelů

Zbavit se nepříjemné zátěže

- chceme už v úvodní fázi odfiltrovat problémy
- co nejdřív odmítnout problém a neposílat aplikaci
- navíc je nemusíme ani logovat

Příklad filtrace

```
if ($query_string ~* ".*(where|drop|insert|select)") {
   return 403;
   }
```

ROOT.CZ_ vpsFree.cz

Pozor na výchozí server

- spoustu provozu vygenerují boti skenující IP
- narazí na IP adresu a ta je přesměruje na web
  - vyzradíme tím IP adresu webu za CDN
  - bot začne agresivně skenovat, co našel
- lepší je požadavek na IP (a cizí domény) odmítat
- nezapomeňte na HTTPS – vytvořte self-signed certifikát

Výchozí server

```
server {
  server_name _;
  listen 80 default_server;
  listen [::]:80 default_server;
  return 403;
}
```

Neodpovídající klienti

- velký nápor klientů – chceme rychle odpojovat neodpovídající
- odlehčíme TCP stacku, uvolníme rychleji paměť jádru
- nevyčerpáme tak rychle zdroje, budou se uvolňovat
- je třeba s hodnotami experimentovat a nepřepálit je

Odpojení uživatelů

```
reset_timedout_connection on; # pošli TCP RST
client_body_timeout 10;       # klient neposílá (60)
send_timeout 10;              # klient nepřijímá (60)
keepalive_timeout 20;         # zavření spojení (75)
```

ROOT.CZ_ vpsFree.cz

Šifrování pomocí HTTPS

HTTPS je standard

- většina webů dnes nasazuje HTTPS (což je dobře!)
- umožňuje to mimo jiné nasadit HTTP/2
  - rychlejší přenos, paralelizace, push, komprese hlaviček
  - odlehčení web serveru i TCP stacku
- odezvu a zátěž serveru lze vylepšit nastavením

ROOT.CZ_ vpsFree.cz

Nasaďte HTTP/2

- HTTP/2 funguje jen s HTTPS
  - nový protokol po 80 v mnoha sítích neprojde
  - implementují to tak prohlížeče
- když už máte HTTPS, chcete HTTP/2

Zapnutí HTTP/2

```
server {
  listen 443 ssl http2;
…
```

Keš pro TLS sezení

- Nginx umí kešovat sezení pro TLS
  - ukládá parametry spojení TLS
  - zrychluje opakování handshake v TLS
- ve výchozím stavu je vypnuto
- lze zapnout sdílenou keš mezi workery
- uvádí se název keše a vyhrazená paměť
  - 1 MB paměti pojme zhruba 4000 položek

TLS keš

```
ssl_session_cache shared:TLS:100m;
ssl_session_timeout 30m;                # výchozí je 5m
```

ROOT.CZ_ vpsFree.cz

OCSP stapling

- klient zdržuje handshake při ověřování OCSP
  - umožňuje ověřit platnost certifikátu u autority
- server může požadavek získat sám a předat klientovi
  - odpověď je podepsaná = je validní i po předání
- urychlí se tím navázání spojení a tím obsloužení klienta

OCSP stapling

```
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate /path/to/full_chain.pem;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
```

Kešování webu

Keš je, když...

- mezipaměť pro ukládání webového obsahu
  - HTML, obrázků, CSS, JS...
- data procházejí a mohou být ukládána
- při příštím požadavku může být obsah vydán z keše
- automatické vyprazdňování keše podle různých strategií

ROOT.CZ_ vpsFree.cz

Proč chcete kešovat

- opakované použití připraveného obsahu
- snížení latence při načítání obsahu
  - stránka je sestavená a může být blíž uživateli
- snížení zátěže na web serveru
  - nemusíme sestavovat stejnou stránku opakovaně
  - i keš s krátkou dobou platnosti může pomoci
- možnost rozložení zátěže na více backendů
- možnost terminace spojení (HTTPS) na frontendu
- překlenutí krátkodobých výpadků

ROOT.CZ_ vpsFree.cz

Terminologie

Origin server	webserver schopný generovat obsah
Reverse proxy	předřazený webserver bez vlastního obsahu
Hit ratio	poměr stránek odbavených z keše (více = lépe)
Freshness	informace o aktuálnosti kešovaného obsahu
Validation	ověření aktuálnosti obsahu proti zdroji
Invalidation	předčasné zneplatnění obsahu v keši

Strategie pro kešování

- jak často se obsah aktualizuje?
- je obsah individualizován?

1. statický a málo se měnící obsah (nejdelší doba)
   - loga, pozadí, ikony
   - soubory s obecným CSS a JS
   - multimediální soubory
   - soubory ke stažení
2. obměňovaný uživatelský obsah (opatrně, můžeme invalidovat)
   - HTML soubory (třeba s články)
   - měnící se obrázky
   - často upravované CSS a JS
3. obsah nevhodný pro kešování
   - individuální obsah pro přihlášeného uživatele
   - citlivé informace (bankovnictví, webmail…)

ROOT.CZ_ vpsFree.cz

Hlavičky

- kešovací HTTP hlavičky může posílat klient i server
  - proxy vystupuje v obou rolích
- obě strany se informují o svých konfiguracích
- kešující strana může mít svou politiku
  - může invalidovat dříve, ale nikdy **ne později**
- hlavní slovo má obvykle zdroj obsahu
  - jeho konfigurace je *vysílána* právě pomocí hlaviček
- hlavičky **nejsou** citlivé na velikost písmen

Jak si zobrazit HTTP hlavičky

Příkazem Curl

```
$ curl -I https://www.petrkrcmar.cz/
```

Příkazem wget

```
$ wget -S --spider https://www.petrkrcmar.cz/
```

Funkce jednotlivých HTTP hlaviček

Expires	určuje čas v budoucnosti, do kterého je vydávaný obsah platný; poté už se musí klient znovu zeptat
Cache-Control	modernější a flexibilnější varianta téhož; je možné nastavit obě najednou
Etag	unikátní označení obsahu; keš se může kdykoliv zdroje zeptat na jeho platnost
Last-Modified	určuje čas poslední změny; hodí se při některých časově závislých strategiích kešování
Content-Length	velikost přenášeného obsahu; některé keše ukládají jen obsah s předem známou velikosti
Vary	určuje další hlavičky, které zahrnuje keš do svého rozhodovacího klíče

Kešování v Nginx

Nginx nejen jako web server

- Nginx je web server se spoustou možností
- umí fungovat i jako reverzní proxy
- má velmi mocné kešování
  - v roli web serveru (FastCGI keš)
  - v roli reverzní proxy (proxy keš)
- ve výchozím stavu se kešují jen požadavky `GET` a `HEAD`
- respektuje se hlavička `Cache-Control`
- nekešují se `Private`, `No-Cache`, `No-Store` a `Set-Cookie`
- dle klíče vytváří MD5 pro název souboru

ROOT.CZ_ vpsFree.cz

Konfigurace kešující proxy

proxy_cache_path nastavuje parametry keše: cestu, klíč…
proxy_cache vybírá kešovací zónu pro danou část serveru

```
proxy_cache_path /path/to/cache levels=1:2 keys_zone=my_cache:10m max_size=10g
                 inactive=60m use_temp_path=off;
server {
    location / {
        proxy_cache my_cache;
        proxy_pass http://my_upstream;
    }
}
```

levels víceúrovňová struktura; snižuje počet souborů v adresáři
keys_zone pro rychlé hledání ukládá klíče v RAM (1 MB = 8K záznamů)
max_size maximální velikost diskové keše; nepoužívané odstraňuje
inactive doba pro odstranění záznamu; nemaže se podle hlaviček
use_temp_path uloží obsah na stejné místo a pak přejmenuje
proxy_cache zapne použití keše; může být v http, server, location

ROOT.CZ vpsFree.cz

Definice klíče

- je možné definovat, jak se sestavuje klíč
  - z interních proměnných poskytovaných serverem
- ze zvoleného řetězce vznikne MD5 pro název souboru
- může se používat vícestupňová (až 3) adresářová struktura
  - pro snížení počtu souboru v adresáři
  - hierarchie má 1 až 3 stupně, počet zvolených znaků je 1 nebo 2

Příklad cesty k souboru

```
/data/nginx/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Příklad klíče

```
proxy_cache_key "$scheme$proxy_host$request_uri";
```

Čistění keše

- Nginx má mechanismus pro odstranění stránky z keše
- při zavolání správné cesty označí za neplatné
- WordPress má plugin nginx-helper, který sám zavolá

Příklad konfigurace

```
server {
…
location ~ /purge(/.*) {
   allow 127.0.0.1;
   proxy_cache_purge my_cache "$scheme$proxy_host$1";
}
```

ROOT.CZ_ vpsFree.cz

Informace v hlavičce

- Nginx umí přidat informace o úspěchu keše
- nachází se v proměnné `upstream_cache_status`
- existuje řada dalších souvisejících proměnných

Příklad konfigurace

```
add_header X-Cache $upstream_cache_status;
```

Význam jednotlivých výsledků

MISS	odpověď nebyla v keši, byla stažena ze zdroje
BYPASS	nebyla použita keš kvůli podmínkám v `proxy_cache_bypass`
EXPIRED	obsah v keši vypršel; odpověď je ze zdroje
STALE	zdroj neodpovídá, odpověď je z keše podle `proxy_cache_use_stale`
UPDATING	odpověď je z keše, nová verze se stahuje
REVALIDATED	keš ověřila validnost obsahu podle `proxy_cache_revalidate`
HIT	v keši byla validní odpověď, kterou jste právě dostali

FastCGI keš v Nginx

- Nginx umí kešovat i odpovědi z FastCGI
- konfigurace je velmi podobná

Příklad konfigurace

```
fastcgi_cache_path /path/to/cache levels=1:2
                 keys_zone=my_cache:10m max_size=10g
                 inactive=60m use_temp_path=off;

server {
    # ...
    location ~ \.php$ {
        fastcgi_cache my_cache;
        fastcgi_pass unix:/run/php/php7.3-fpm.sock;
    }
}
```

ROOT.CZ_ vpsFree.cz

Shrnutí

- hlídejte úzká hrdla svého serveru
- odfiltrujte zbytečné zatěžující dotazy
- odbavte maximum už na web serveru
- nasaďte silné kešování, kde to jen jde
- laďte parametry jádra webového serveru
- omezte zbytečné I/O operace (logy, keše)
- věnujte se šifrování a zapnutí HTTP/2 (/3)
- opakujte průběžně a pořád dokola

ROOT.CZ_ vpsFree.cz

Děkuji za pozornost

Otázky?

Petr Krčmář
petr.krcmar@iinfo.cz

ROOT.CZ_ vpsFree.cz