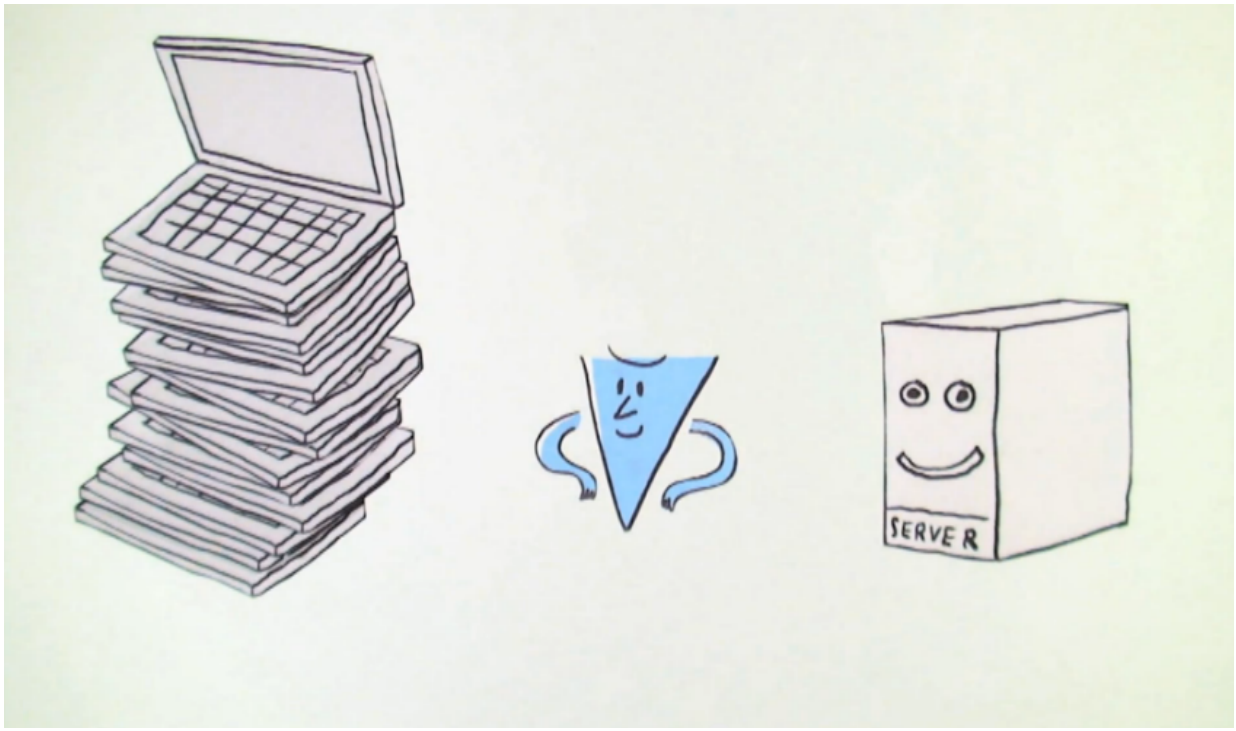
Varnish Cache

Varnish Cache to bardzo szybki i lekki HTTP accelerator (znany również pod nazwą reverse proxy) dla Linuxów, BSD oraz OS X udostępniony na licencji open source, napisany z myślą o serwowaniu ogromnych ilości statycznych oraz dynamicznych treści przez HTTP. Wykorzystując Varnish Cache w prosty sposób odciążamy serwer HTTP, jednocześnie zyskując około 80% wzrost wydajności w serwowanych aplikacjach (przy standardowej konfiguracji).

Można sobie wyobrazić, że Varnish działa jak bardzo duży słownik klucz-wartość, który przechowuje zapytania HTTP i zwraca treści. Każde zapytanie kierowane do naszego serwera przechodzi pierw przez Varnish, który stara się odpowiedzieć na podstawie zcache'owanych wcześniej danych. Jeśli nie pytaliśmy wcześniej o pewien dokument - Varnish za nas odpyta serwer i wyśle nam jego zcache'owaną wersję.



Varnish wspiera jedynie ruch HTTP, w odróżnieniu od innych proxy, które często działają także na FTP, SMTP i innych protokołach. Varnish stara się wykorzystywać w pełni architekturę nowoczesnych systemów operacyjnych. Cache'owane dane trzymane są w pamięci wirtualnej - o tym co zostanie w pamięci, a co zapisane na dysk decyduje system operacyjny. Pomaga to unikać sytuacji, w których system operacyjny zaczyna cache'ować dane w momencie, w którym zapisywane są na dysk przez aplikację. Ponadto każde połączenie obsługiwane jest przez pojedynczego workera działającego w postaci wątku - w przypadku osiągnięcia limitu aktywnych workerów przychodzące połączenia trafiają do kolejki.

Ważną zaletą Varnish Cache jest to, że jego zastosowanie nie wymaga od nas zmian w kodzie aplikacji, a jedynie instalacji i uruchomienia po stronie serwera. Możemy skonfigurować serwer tak, aby pobierał dane z różnych serwerów lub stworzyć klaster i serwować treści z kilku serwerów cache. Zmian w konfiguracji dokonuje się przy pomocy edycji skryptów startowych, plików VCL oraz instalacji Varnish Modules (pluginów).

W sieci można znaleźć wiele gotowych konfiguracji do Varnish Cache zoptymalizowanych pod popularne serwisy/frameworki takie jak Drupal, WordPress itp.

Ciekawostki

- pierwsza wersja Varnish Cache pozwoliła jej pierwszemu użytkownikowi - norweskiej gazecie VG w 2006 roku zastąpić 12 maszyn ze Squidem (100% użycia CPU na każdej) jedynie 3 maszynami z Varnishem (90% użycia CPU na każdej)
- Mozilla po wykorzystaniu Varnish Cache zmniejszyła czas ładowania stron średnio o 2.2 sekundy, zwiększając jednocześnie "*download conversion*" o 15.4%
- zdaniem autorów Varnish będzie działać tak szybko, jak szybkie mamy połączenie oraz jak szybka są implementacje *pthreads* (POSIX Threads) oraz *malloc'*a na danej maszynie
- obecnie Varnish Cache wykorzystywany jest między innymi przez:
    - Facebook
    - Tumblr
    - Twitter
    - LinkedIn
    - vimeo
    - Verdens Gang
    - Heroku

Instalacja

Wymagane zależności

- autotools-dev
- automake1.9
- libtool
- autoconf
- libncurses-dev
- xsltproc
- groff-base
- libpcre3-dev
- pkg-config

Instalacja na Debianie

- curl http://repo.varnish-cache.org/debian/GPG-key.txt | apt-key add -
- echo "deb http://repo.varnish-cache.org/debian/ wheezy varnish-3.0" >> /etc/apt/sources.list
- apt-get update
- apt-get install varnish

Podstawowa konfiguracja

Varnish Cache korzysta z minijęzyka zwanego VCL, odrobinę podobnego do C. Pozwala on na zmiany w nagłówkach zapytań i odpowiedzi oraz kierowanie requestu gdzie trzeba oraz dodatkowe akcje takie jak na przykład purgowanie obiektów w cache. Pliki VCL mogą być przeładowywane w locie dzięki czemu nie traci się cache przy zmianie ustawień. Podczas ładowania konfiguracji pliki VCL tłumaczone są do C, kompilowane i ładowane bezpośrednio do głównego programu.

Należy pamiętać, że Varnish Cache przy standardowej konfiguracji nie będzie cache'ował dokumentów, które zwracają pliki cookie!

Edytujemy plik /etc/varnish/default.vcl:

```
backend default {
    .host = "127.0.0.1";
    .port = "80";
    .first_byte_timeout = 180s;
    .connect_timeout = 45s;

}
```

Powyższa konfiguracja tworzy nowy backendowy serwer Varnisha o nazwie default, który będzie pobierać treści z adresu 127.0.0.1:80, timeout przy łączeniu wynosi 45s, a użytkownik otrzyma błąd 503 dopiero po 180 kolejnych sekundach (łącznie 225s).

Dodawanie plików do cache'owania

Poniższa reguła dodaje pliki gif, jpg, swf i css do backendu o nazwie default (zdefiniowany powyżej), jednocześnie usuwając ustawione ciasteczka.

```
sub vcl_recv {
   if (req.url ~ "\.(gif|jpg|swf|css)$") {
      unset req.http.cookie;
      set req.backend = default;
   }
}
```

Uruchomienie

varnishd -s malloc,1G -T 127.0.0.1:2000 -a 0.0.0.0:8080

- -s malloc,1G Varnish będzie cache'ować dane w 1GB pamięci operacyjnej
- -T 127.0.0.1:2000 otwieramy prosty tekstowy interfejs na adresie 127.0.0.1:2000
- -a 0.0.0.0:8080 akceptujemy połączenia przychodzące na adresie 0.0.0.0:8080

Zamiast wpisywać polecenie z ręki możemy zmienić skrypt startowy znajdujący się w /etc/default/varnish (zmienna DAEMON_OPTS).

Przeładowanie zmian przy pomocy
service varnish restart

Logi

Varnish nie zapisuje logów na dysk, lecz trzyma je w pamięci (ich część). W każdej chwili możemy się podłączyć i zobaczyć logi przy użyciu polecenia **varnishlog**

Przykładowy log

```
11 SessionOpen  c 127.0.0.1 58912 0.0.0.0:8080
11 ReqStart     c 127.0.0.1 58912 595005213
11 RxRequest    c GET
11 RxURL        c /
11 RxProtocol   c HTTP/1.1
11 RxHeader     c Host: localhost:8080
11 RxHeader     c Connection: keep-alive
```

Prosty load balancer

```
backend node1 {
  .host  = "node1";
  .port  = "8080";
}
backend node2 {
  .host  = "node2";
  .port  = "8080";
}
backend node3 {
  .host  = "node3";
  .port  = "8080";
}
backend node4 {
  .host  = "node4";
  .port  = "8080";
}
director nodes round-robin {
  { .backend = node1 ; }
  { .backend = node2 ; }
  { .backend = node3 ; }
```

```
  { .backend = node4 ; }
}
sub vcl_recv {
  set req.backend = nodes;
}

sub vcl_hash {
  hash_data(req.url);
  return (hash);
}
```

Zadania na ćwiczenia

1. Zapoznaj się z dokumentacją Varnish Cache oraz przykładowymi plikami VCL (przydatne odnośniki poniżej).
2. Zainstaluj i skonfiguruj serwer Varnish Cache tak, żeby działał ze skonfigurowanym na wcześniejszych zajęciach serwerem Apache.
3. Zainstaluj pewną aplikację na serwerze (Drupal, Wordpress, jakieś forum).
4. Zmień pliki VCL serwera Varnish, tak aby cache'ował potrzebne pliki.
5. Sprawdź wydajność swojego serwera przy pomocy narzędzia ApacheBench.
6. Wyłącz serwer Varnish Cache, ponownie sprawdź wydajność i porównaj wyniki.

Przydatne linki

- https://www.varnish-cache.org/docs/3.0/
- https://www.varnish-cache.org/docs/3.0/tutorial/index.html
- https://www.varnish-cache.org/trac/wiki/VCLExamples
- https://www.varnish-software.com/products-services/varnish-cache
- https://www.varnish-cache.org/about
- https://www.varnish-cache.org/vmods
- https://github.com/varnish/Varnish-Cache
- https://www.digitalocean.com/community/articles/how-to-install-and-configure-varnish-with-apache-on-ubuntu-12-04--3
- http://www.euperia.com/website-performance-2/setting-up-varnish-with-apache-tutorial/299
- http://cainmanor.com/tech/how-to-use-varnish-cache-with-wordpress/
- http://dev.theladders.com/2013/05/varnish-in-five-acts/