WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKI WROCŁAWSKIEJ

PEER-TO-PEER WEB OBJECTS CACHEING PROXY

TOMASZ DRWIĘGA

Praca magisterska napisana
pod kierunkiem
dra Mirosława Korzeniowskiego

WROCŁAW 2013

Spis treści

0.1 Sformułowanie problemu 3
0.2 Istniejące rozwiązania 3

1 Historia keszowania **3**
1.1 Serwery cache'ujące 3
1.2 Rozproszony cache 3

2 Opis działania **4**

3 Analiza bezpieczeństwa i wydajności **4**
3.1 System, w którym klucz == url 4
3.2 System, w którym klucz == hash(url) 4

4 Analiza różnych metod cachowania **5**

5 Testy **6**
5.1 Spreparowane dane 6
5.2 Wdrożenie 6

6 Implementaja **6**
6.1 Wybór technologii 6
6.2 Biblioteki 6
6.3 Instalacja i uruchomienie 6

0.1 Sformułowanie problemu

0.2 Istniejące rozwiązania

1 Historia keszowania

Wraz z rosnącą liczbą użytkowników Internetu w latach dziewięćdziesiątych zaczęły pojawiać się problemy z dostępem do pewnych zasobów. Duża liczba odwołań do popularnych strony, w krótkim czasie powodowała znaczące obciążenie serwerów, które dany zasób posiadały. Czasami z powodu nadmiernego zalania[1] żądaniami serwer nie był w stanie obsłużyć wszystkich przez co strona stawała się niedostępna.

1.1 Serwery cache'ujące

Jako rozwiązanie problemu "gorących punktów" (ang. *hot spots*) pojawiły się serwery cache'ujące. Rysunek 1 przedstawia wprowadzenie transparentnego, lokalnego cache'a i jego relację z serwerem docelowym. Żądania zasobów wychodzące od użytkowników końcowych są w pierwszej kolejności obsługiwane przez serwer cache'ujące, który odpowiada zapamiętanym zasobem lub kontaktuje się z serwerem docelowym i zapamiętuje odpowiedź.

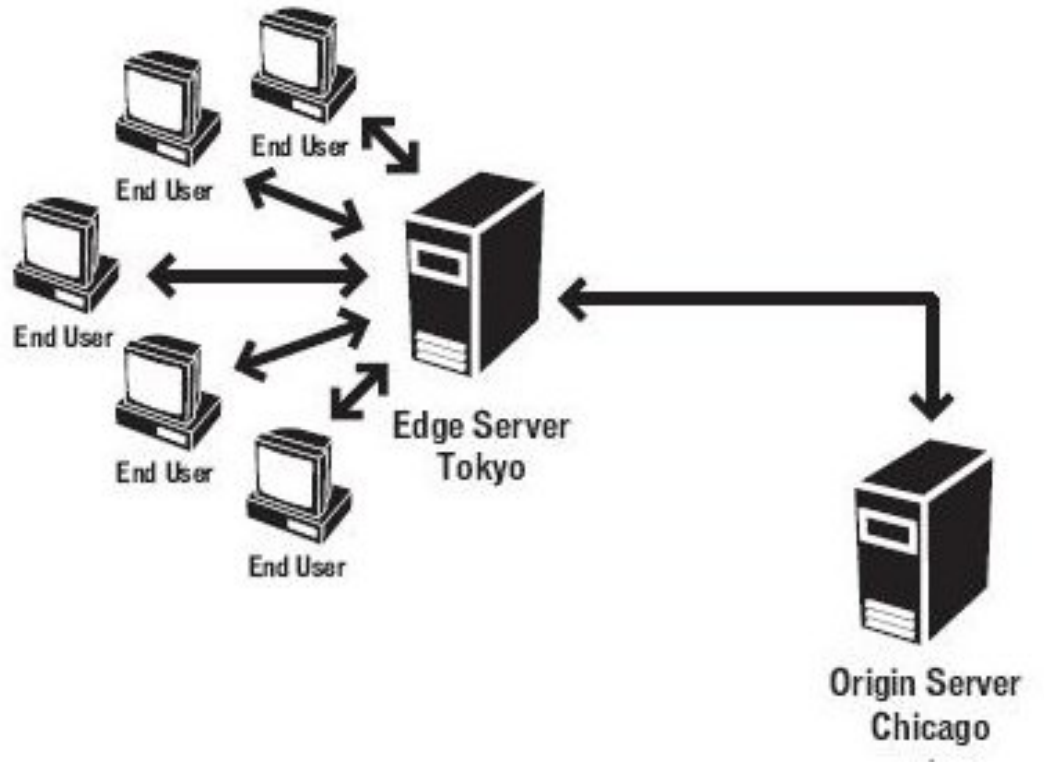
Wprowadzenie cache'owania niesie ze sobą szereg zalet nie tylko dla administratorówserwerów zawierających zasoby. Serwery znajdujące się na granicy sieci lokalnej z Internetem (jak na rysunku 1) oferują użytkownikom tej sieci lepszy transfer i mniejsze opóźnienia w dostępie do popularnych zasobów. Ponieważ łącze wychodzące z sieci ma ograniczoną przepustowość, cache pozwala również na oszczędniejsze wykorzystanie łącza, a tym samym poprawę jakości dostępu do Internetu.

1.2 Rozproszony cache

Rozwiązanie przedstawione w rozdziale 1.1 pozwala na odciążenie serwera docelowego. Zastanówmy się jednak co będzie się działo w przypadku zwiększania liczby użytkowników korzystających z serwera cache'ującego. Duża liczba żądań może spowodować dokładnie taką samą sytuację jak w przypadku serwera docelowego - cache zostanie zalany i nie będzie w stanie obsłużyć wszystkich zapytań.

W ramach projektu Harvest [1] A. Chankhunthod i inni [2] opracowali cache hierarchiczny.

[1] ang. *flooded*, *swamped*



Źródło: `http://www.neatfilm.com/2005/11/16/distributed-media-server-infrastructure/`

Rysunek 1: Przykład lokalnego serwera cache'ującego. Zamiast wielokrotnie odwoływać się do docelowego serwera, który zawiera daną stronę możemy zapamiętać ją na serwerze cache'ującym. Takie rozwiązanie niesie ze sobą zalety zarówno dla administratorów serwera w Chicago, jak i użytkowników sieci w Japoni: mniej żądań skutkuje mniejszym obciążeniem serwera, a "lokalność" serwera cache'ującego zmniejsza opóźnienia i zwiększa szybkość transferu zasobu.

2 Opis działania

3 Analiza bezpieczeństwa i wydajności

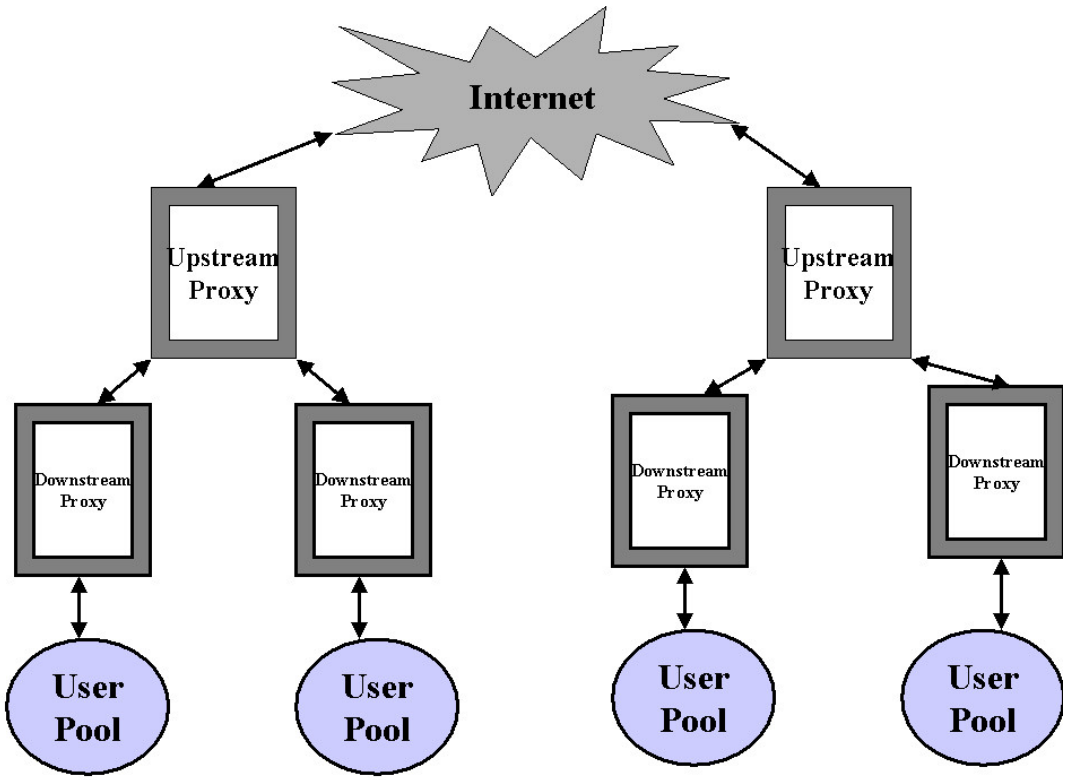
3.1 System, w którym klucz == url

Można śledzić co przeglądają użytkownicy.

Nie jest wymagane STORE zasobu - węzły mogą rozpocząć pobieranie podczas operacji SEARCH.

3.2 System, w którym klucz == hash(url)

Dodatkowo zawartość plików może być szyfrowana prawdziwym `url`. W ten sposób nie można podglądać zawartości plików.



Źródło: `http://www.cse.wustl.edu/~jain/cis788-99/ftp/web_caching/`

Rysunek 2: Połączenie kilu serwerów w system hierarchiczny. Każdy z serwerów znajdujących się w liściach obsługuje określoną liczbę użytkowników. Zapytania o zasoby, które nie znajdują się w cache'u wysyłane są do rodzica.

4 Analiza różnych metod cachowania

Cache wielopoziomowy:

1. w pamięci,
2. na dysku,
3. w sieci P2P (te dane również w pamięci, na dysku)

Różne strategie przechodzenia między poziomami.

5 Testy

5.1 Spreparowane dane

5.2 Wdrożenie

6 Implementaja

6.1 Wybór technologii

Tutaj próby zrobienia tego w JS jako plugin do przeglądarki.

6.2 Biblioteki

Twisted, Entangled.

6.3 Instalacja i uruchomienie

Literatura

[1] C Mic Bowman, Peter B Danzig, Darren R Hardy, Udi Manber, Michael F Schwartz, and D Wessels. The harvest information discovery and access system. *Internet Research Task Force-Resource Discovery, http://harvest. transarc. com*, 1994.

[2] Anawat Chankhunthod, Peter B Danzig, Chuck Neerdaels, Michael F Schwartz, and Kurt J Worrell. A hierarchical internet object cache. Technical report, DTIC Document, 1995.