Replikacja jako technika skalowalności w Internecie

Marcin Koziński

23 kwietnia 2009

Spis treści

Spis treści

Trzy wymiary skalowalności

W Internecie problem stanowi skalowalność różnych rodzajów:

- **numeryczna** – trzeba obsługiwać dużą liczbę użytkowników i/lub dostarczać im duże ilości danych,
- **geograficzna** – komunikujące się strony są od siebie bardzo oddalone,
- **administracyjna** – jest wiele różnych niezależnych podsieci, każda z innym administratorem, każda z inną polityką zapewniania bezpieczeństwa, itd.

Parametry nie są stałe

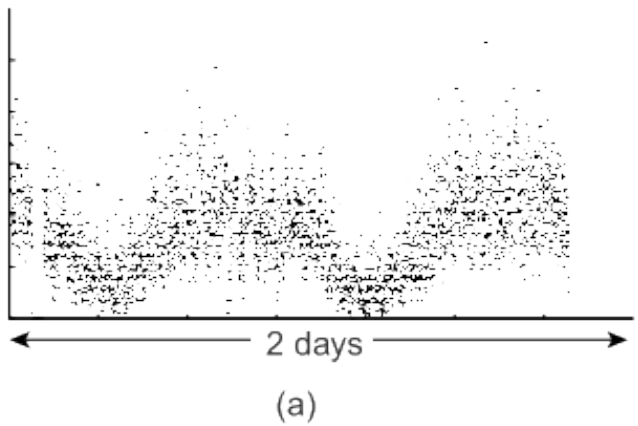
Czasy komunikacji między dwoma stronami podlegają fluktuacjom nawet w przypadku stałej topologii sieci. A topologia się zmienia.

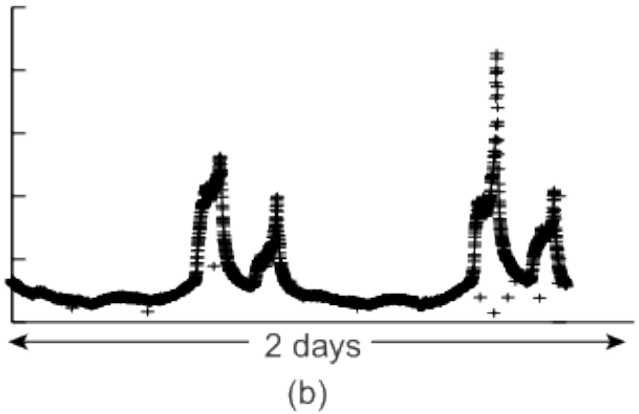
Serwisy są obciążone w różnym stopniu w różnych porach dnia, tygodnia i roku. Ponadto wzorce wejść na strony zmieniają się w wyniku nieprzewidywalnych czynników zewnętrznych.

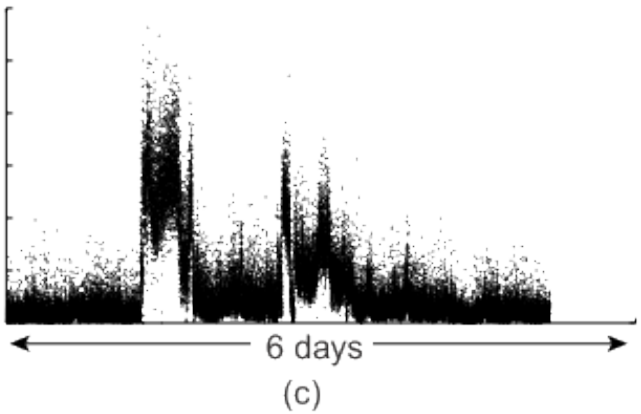
Flash crowd

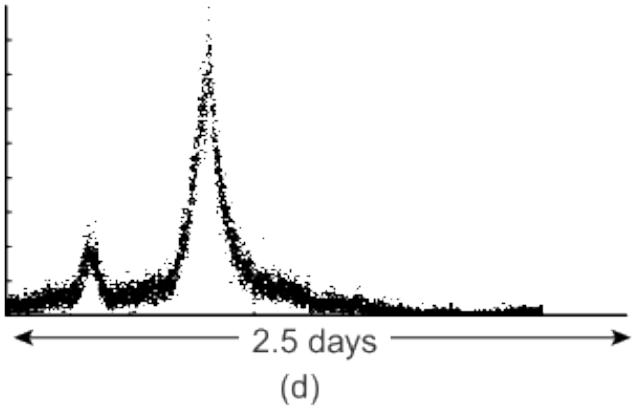
Znany też jako **slashdot effect**. Nagły wzrost liczby wejść na serwis o rząd lub nawet kilka rzędów wielkości. Najczęściej prowadzi do DoS.

Flash crowd









Tradycyjne rozwiązania 1/2

Rozpraszanie: Prostą metodą radzenia sobie z problemem skalowalności numerycznej jest dorzucenie większej liczby maszyn i podzielenie pracy pomiędzy nie. Jeśli cały sprzęt jest w jednym miejscu, to nie radzimy sobie ze skalowalnością geograficzną. Jeśli jest rozproszony po świecie, to możemy mieć wewnętrzny problem ze skalowalnością geograficzną.

Ponadto zwykłe klastry nie skalują się dobrze na rząd tysięcy maszyn. Także dają cały czas taką samą wydajność a obciążenie w szczycie może być o rząd wielkości wyższe od średniego.

Tradycyjne rozwiązania 2/2

Replikacja i cache'owanie: Umieszczanie kopii danych blisko miejsca, w którym są potrzebne. Dzięki temu postrzegany przez użytkownika czas oczekiwania jest krótszy. Jest dobrym sposobem skalowania geograficznego.

Prowadzi do problemów z zachowaniem spójności. Trzeba wybrać słabsze modele spójności, żeby osiągnąć zysk.

Proxy caches realizują ten pomysł, ale odsetek trafień w nich to 25-40%. Żeby dobrze działało, trzeba to robić mniej naiwnie.

Spis treści

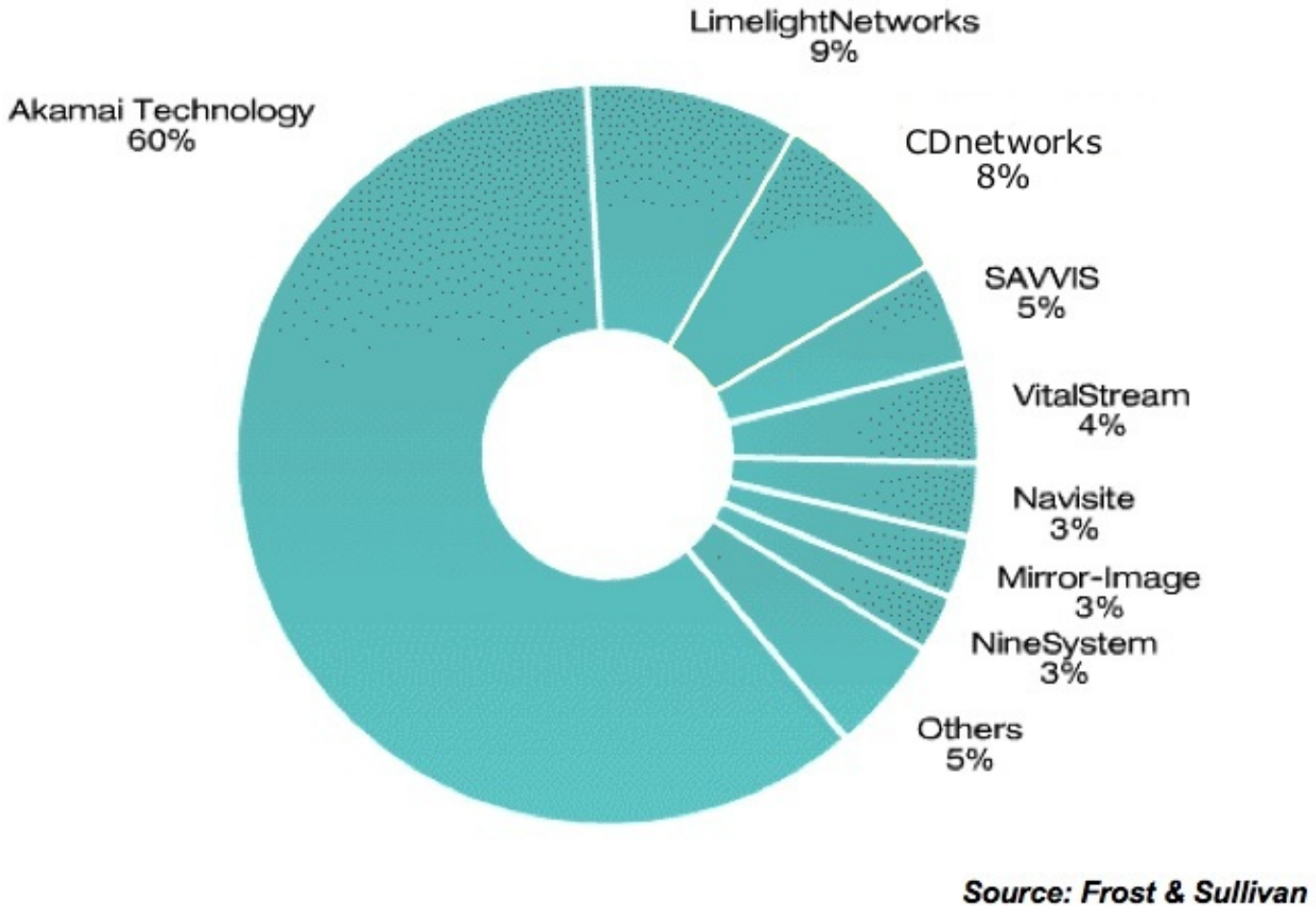
Content Delivery Network

Definicja

A *content delivery network* (CDN) is a system of computers networked together across the Internet that cooperate transparently to deliver content to end users, most often for the purpose of improving performance, scalability, and cost efficiency.

Z punktu widzenia klienta jest to usługa, która odpowiada na typowe wymagania wobec poważnego serwisu internetowego, który ma być dostępny wszędzie, zawsze i ma działać szybko niezależnie od miejsca i czasu.

Największy dostawca usługi CDN



12 000 serwerów w ponad 1000 sieciach.

Akamai

The company was founded in 1998 by then-MIT graduate student Daniel Lewin, along with MIT Applied Mathematics professor Tom Leighton and MIT Sloan School of Management students Jonathan Seelig and Preetish Nijhawan. Operating profit in 2007: 144,9 million dollars.

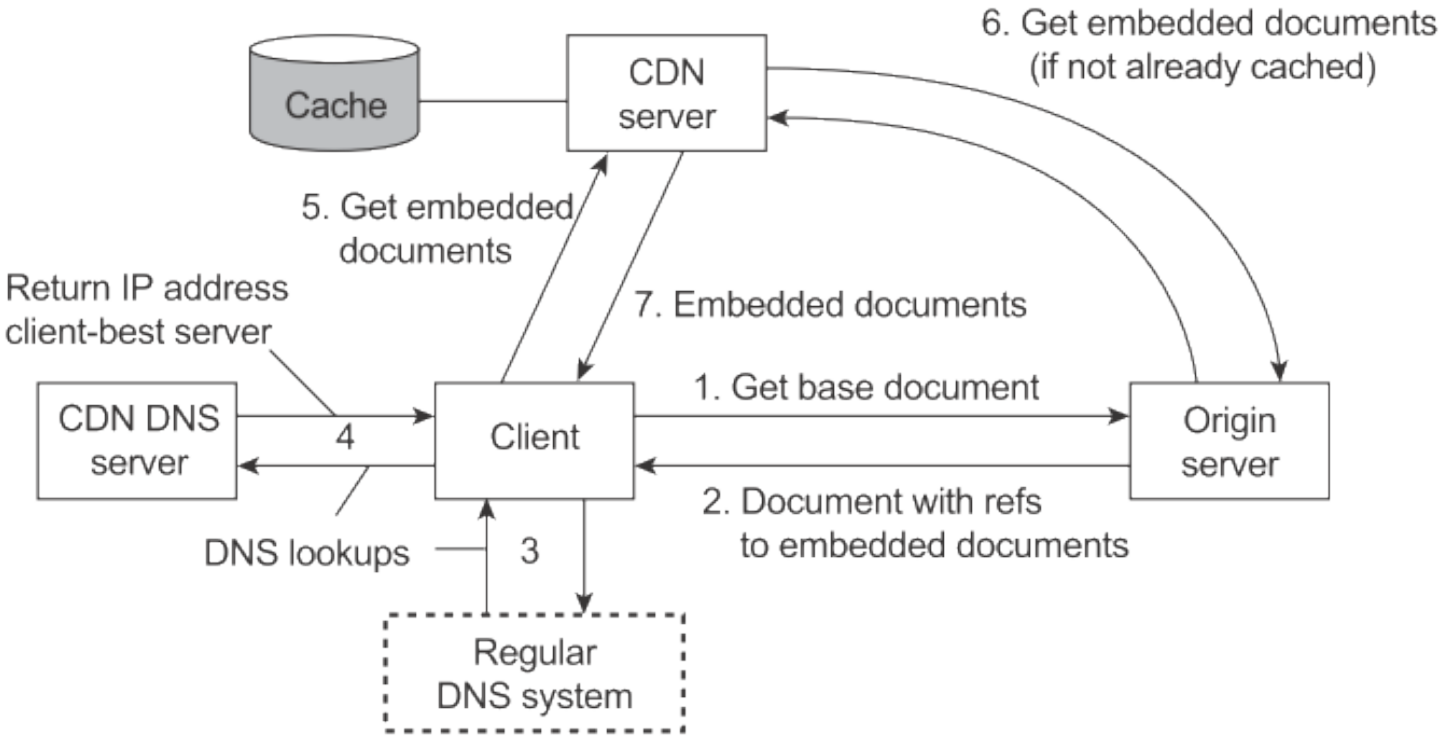
Klienci Akamai

Adobe,
Apple Computer,
AUDI AG,
BMW Japan,
Corel Corp.,
Department of Defense,
Federal Emergency Management Agency,
FedEx Corp.,
Food and Drug Administration,
FUJI TV,
Fujitsu,
General Motors,
Harley-Davidson, Inc.,
Hitachi,
IBM Corp.,
Logitech,
L'Oréal,
MTV Networks,
MySpace,
Nintendo,
Red Hat Inc.,
Reebok International Ltd.,
Reuters,
Sony Computer Entertainment Inc.,
Sony Ericsson Mobile Communications,
Toyota Motor Sales,
Yahoo!

Schemat działania 1/3

- Sieć składa się z głównie z dokumentów zagnieżdżanych. Sam `index.html` jest nieduży, znacznie większe są obrazki, obiekty flashowe, itd.
- Rozpraszane są więc te dodatkowe dokumenty. Użytkownik pobiera ze źródłowego serwera tylko `index.html` z referencjami postaci `http:\\a7.g.akamai.net\obrazek.jpg`
- Trzeba więc przejść wysłać zapytania DNS, które kierują do domeny Akamai, gdzie kolejne zapytania obsługuje serwer nazw Akamai. Ten serwer odsyła użytkownika do odpowiedniego serwera replik.

Schemat działania 2/3



Schemat działania 3/3

Cache'owanie jest użyte jako strategia decydowania o tym, co ma pamiętać konkretny serwer replik. Jeśli użytkownik zostanie przekierowany przez serwer nazw do maszyny, która nie posiada żądanego dokumentu zostanie on ściągnięty z serwera źródłowego i zapamiętany w cache'u.

Spójność jest zachowana dzięki wersjonowaniu plików lub dodawaniu hasha zawartości do nazwy. Dzięki temu pojawienie się nowej wersji spowoduje *cache miss* na serwerze replik i pobranie pliku z serwera źródłowego. Dodatkowo stara wersja może zostać usunięta.

Wybór serwera replik

Wybór odpowiedniego serwera replik dla każdego użytkownika jest oczywiście kluczowy. Od tego zależy postrzegana wydajność systemu oraz balansowanie obciążeniem.

Akamai przekierowuje do „*najbliższego dostępnego* serwera, który *prawdopodobnie* posiada żądaną zawartość".

- **najbliższy** to funkcja topologii sieci oraz dynamicznych parametrów połączenia, jak *round trip time* i procent gubionych pakietów;
- **dostępny** to funkcja obciążenia serwera i jego łącza;
- **prawdopodobnie** to funkcja wynikająca ze strategii rozmieszczania dokumentów konkretnego klienta w centrach danych.

Spis treści

Eksperyment

Pytanie

Czy opłaca się rozpraszać każdy dokument według jego indywidualnej najlepszej strategii, zamiast stosować jeden ogólny schemat dla wszystkich dokumentów?

- Zebrano ślady (*traces*) zapytań i uaktualnień dla wszystkich stron internetowych z dwóch różnych serwerów (w Amsterdamie i w Erlangen).
- Dla każdego zapytania sprawdzono:
  - z jakiego AS-u pochodził,
  - jakie było średnie opóźnienie do tego klienta,
  - jaka była średnia przepustowość do AS-u tego klienta.
- Odegrano pliki ze śladami dla wielu różnych konfiguracji systemu i wielu różnych scenariuszy rozproszenia.

Strategie cache'owania

- **No replication** (NR) – całkowity brak replikacji i cache'owania. Wszytkie klienty kierują swoje zapytania do serwera źródłowego.
- **Verification** (CV) – serwery brzegowe trzymają cache dokumentów. Przy każdym zapytaniu serwer źródłowy jest kontaktowany w celu sprawdzenia aktualności.
- **Limited validity** (CLV) – serwery brzegowe trzymają cache dokumentów. Zapamiętany dokument posiada czas wygaśnięcia, po którym staje się nieaktualny i jest wyrzucany z pamięci.
- **Delayed verifiaction** (CDV) – serwery brzegowe trzymają cache dokumentów. Zapamiętany dokument posiada czas wygaśnięcia, po którym następuje kontakt z serwerem źródłowym w celu sprawdzenia aktualności.

Strategie replikacji

- **Server invalidation** (SI) – serwery brzegowe trzymają cache dokumentów. Serwer źródłowy unieważnia kopie, kiedy dokument jest uaktualniany.
- **Server updates** (SUx) – serwer źródłowy przechowuje kopie na x najlepszych serwerach brzegowych; $x = 10$, 25 albo 50.
- Hybryda SU50 + CLV – serwer źródłowy przechowuje kopie na 50 najlepszych serwerach brzegowych. Pozostałe serwery zachowują się zgodnie ze strategią CLV.
- Hybryda SU50 + CDV – serwer źródłowy przechowuje kopie na 50 najlepszych serwerach brzegowych. Pozostałe serwery zachowują się zgodnie ze strategią CDV.

Wynik dla globalnej strategii

Turnaround time and bandwidth in relative measures; stale documents as fraction of total requested documents.

Strategy	Site 1			Site 2		
	Turnaround	Stale docs	Bandwidth	Turnaround	Stale docs	Bandwidth
NR	203	0	118	183	0	115
CV	227	0	113	190	0	100
CLV	182	0.0061	113	142	0.0060	100
CDV	182	0.0059	113	142	0.0057	100
SI	182	0	113	141	0	100
SU10	128	0	100	160	0	114
SU25	114	0	123	132	0	119
SU50	102	0	165	114	0	132
SU50+CLV	100	0.0011	165	100	0.0019	125
SU50+CDV	100	0.0011	165	100	0.0017	125

Wniosek: nie ma najlepszej strategii globalnej.

Strategie indywidaulne 1/2

- Załóżmy, że mamy k metryk wydajności $m_1, \ldots, m_k$.
- Niech D będzie zbiorem dokumentów a S zbiorem strategii.
- Niech w będzie wektorem wag ($\sum w_i = 1, w_i \geqslant 0$).
- Niech $res(m_i, d, s)$ będzie wartością metryki m_i dla dokumentu d przy strategii s.
- **Układ** (arrangement): Zbiór par (dokument, strategia): $A = \{(d, s_A(d)) | d \in D, s_A(d) \in S\}$. Każdy układ ma swój koszt:

$$cost_w(A) = \sum_{i=1}^{k} w_i \cdot \left(\sum_{d \in D} res(m_i, d, s_A(d)) \right)$$

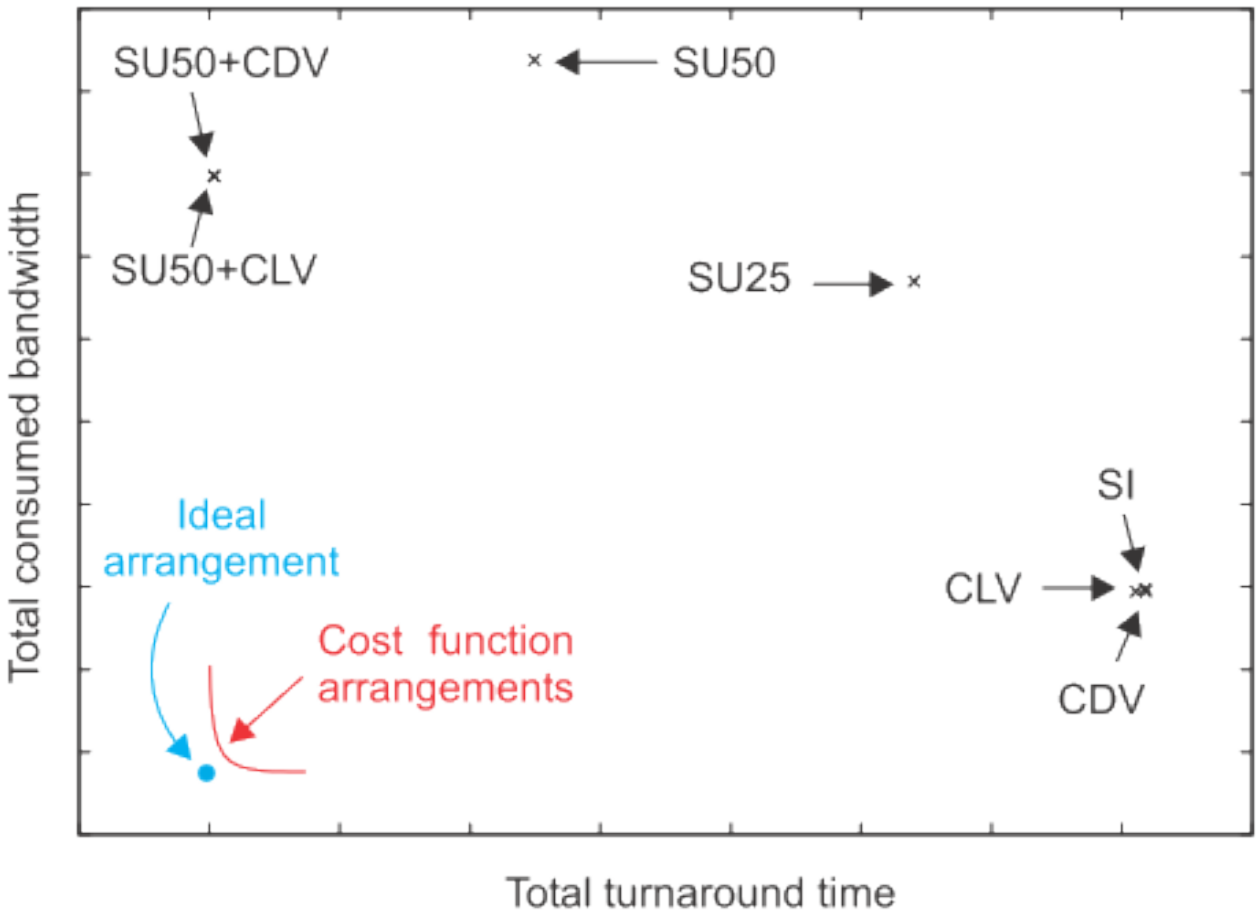
Strategie indywidaulne 2/2

- **Cel:** Szukamy takiego układu A,że $cost_w(A)$ jest minimalny.
- Według autorów eksperymentu można przyjąć następującą równość:

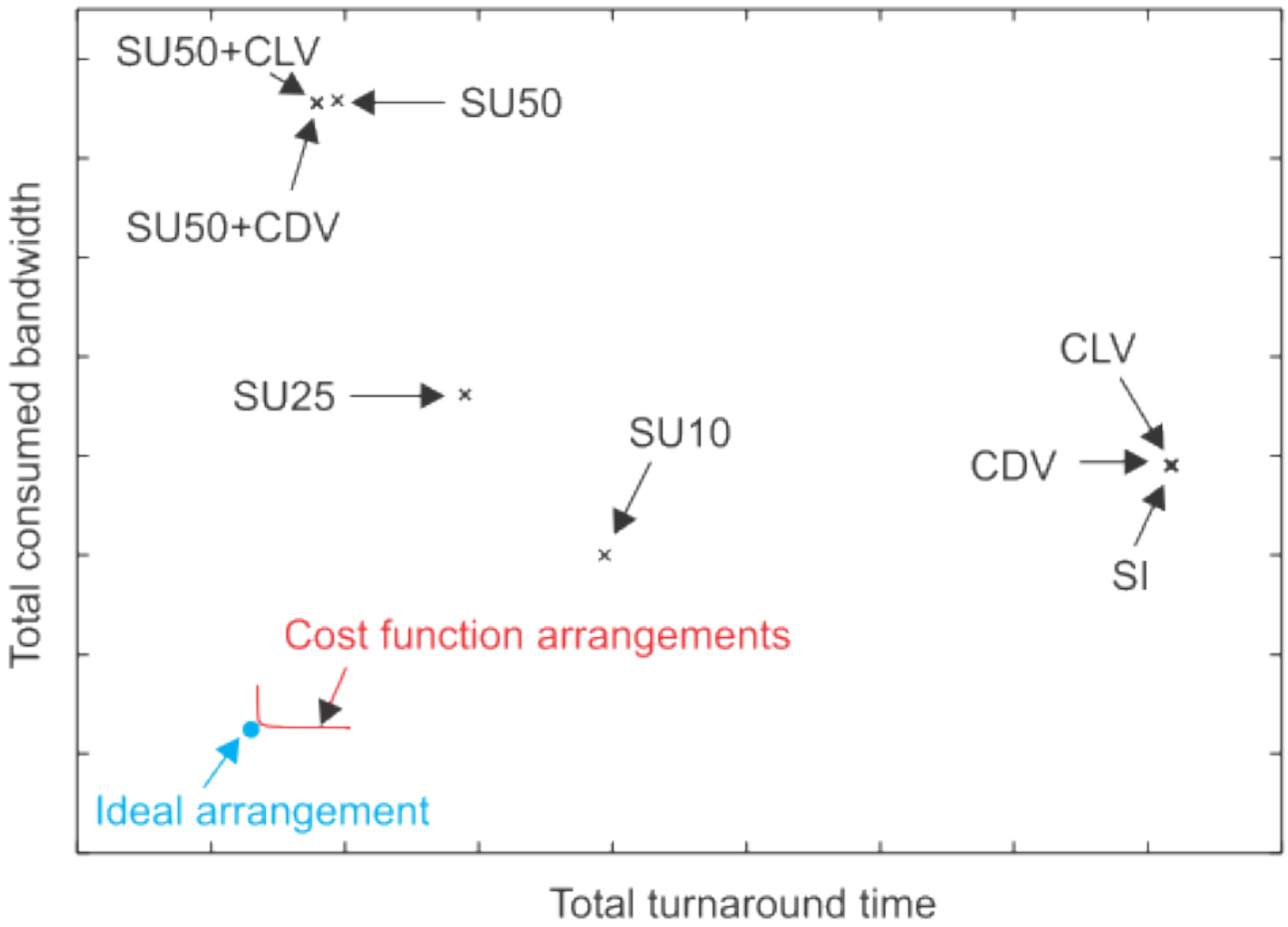
$$\sum_{i=1}^{k} w_i \cdot \sum_{d \in D} res(m_i, d, s_A(d)) = \sum_{d \in D} \sum_{i=1}^{k} w_i \cdot res(m_i, d, s_A(d))$$

  Tzn. że możemy szukać minimalnego kosztu dla każdego dokumentu osobno.
- To jest bardzo dobra wiadomość, bo oznacza tylko $|D| \cdot |S|$ strategii (w porównaniu z $|S|^{|D|}$ wszystkich układów). Takie obliczenia można przeprowadzać nawet w trakcie działania systemu.

Wyniki: Serwer 1



Wyniki: Serwer 2



Wykorzystywane strategie

Fraction of documents to which a strategy is assigned.

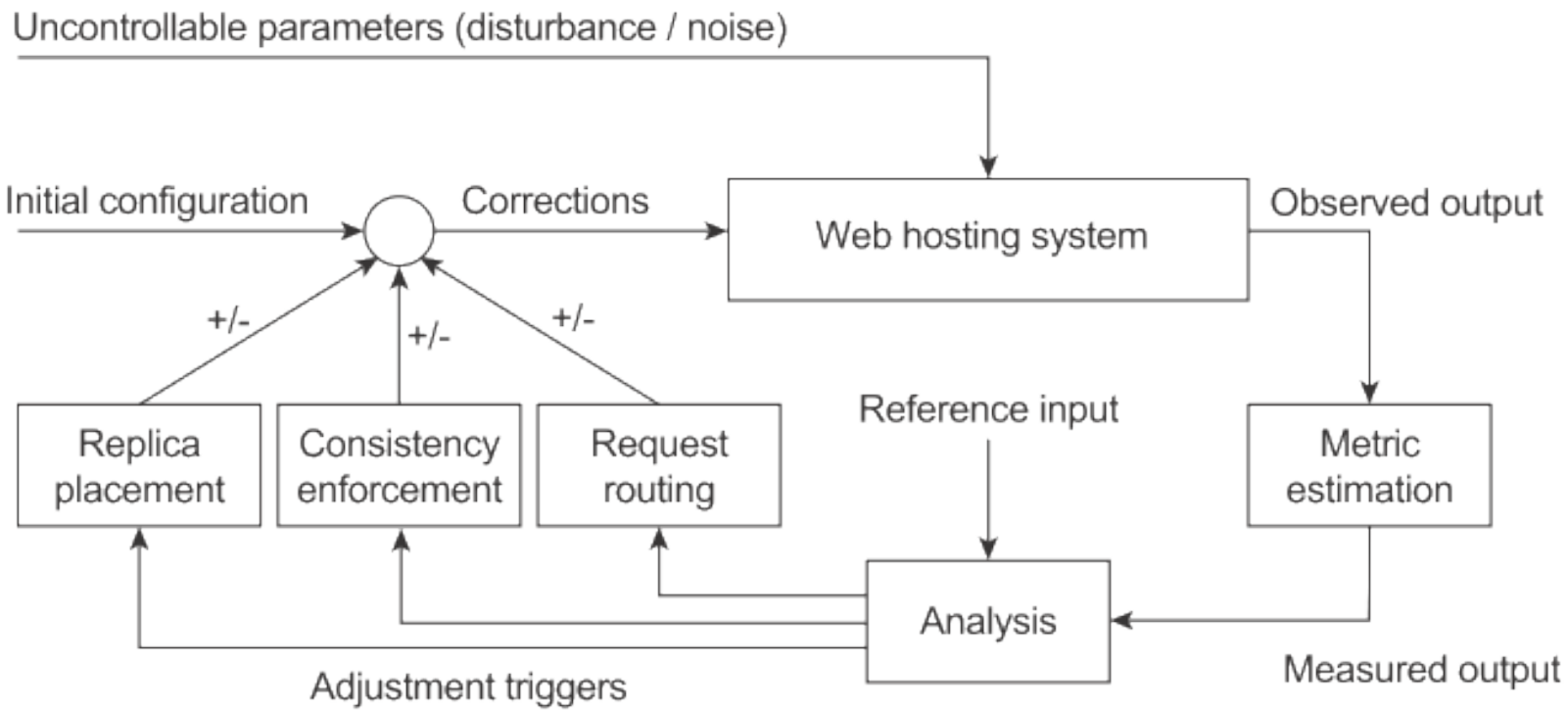
Strategy	Site 1	Site 2
NR	0.0973	0.0597
CV	0.0001	0.0000
CLV	0.0131	0.0029
CDV	0.0000	0.0000
SI	0.0089	0.0061
SU10	0.1321	0.6087
SU25	0.1615	0.1433
SU50	0.4620	0.1490
SU50+CLV	0.1232	0.0301
SU50+CDV	0.0017	0.0002

Wniosek: warto przydzielać indywidualne strategie.

Globule

- Globule: an Open-Source Content Distribution Network
- a **collaborative** content delivery network (CCDN) developed by research group at Vrije Universiteit of Amsterdam. Globule is composed of Web servers that cooperate across a wide-area network to provide performance and availability guarantees to the sites they host.
- `http://www.globule.org/`

Pętla ze sprzężeniem zwrotnym



Wybór metryk

Jakich metryk potrzebujemy?

- czas odpowiedzi do klienta,
- procent gubionych pakietów,
- ilość zużytej przepustowości łącza,
- obciążenie serwerów,
- spójność danych.

Obliczanie metryk 1/2

Obliczanie metryk dla każdego klienta osobno się nie skaluje. Rozwiązaniem jest **grupowanie klientów** (client clustering) i wyliczenie metryki dla całej grupy. Założeniem jest, że wartość dla grupy jest dobrym estymatorem wartości dla każdego klienta należącego do niej.

Obliczanie metryk 2/2

Można grupować według:

- lokalnych serwerów nazw – słaba korelacja w czasach transmisji pomiędzy połączeniem serwer nazw $\longrightarrow$ inny serwer, a klient $\longrightarrow$ inny serwer,
- systemów autonomicznych – tak naprawdę domena administracyjna, klienty nie zawsze są blisko siebie geograficznie; wykorzystane w Globule,
- serwerów proxy – dobre, ale wiele klientów nie używa proxy,
- topologii sieci – *network-aware clustering*, polega na wykorzystaniu informacji, które routery wymieniają przez BGP do zgrupowania klientów, które są rzeczywiście blisko siebie; skuteczność na poziomie 99,99%.

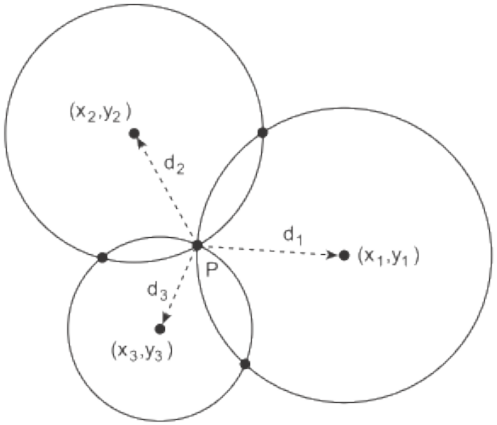
Estymacja metryk: czas oczekiwania klienta 1/3

Czas oczekiwania jest miarą odległości (w logicznym sensie). Idealnie byłoby go zmierzyć w chwili, kiedy potrzebujemy wykonać transmisję. **Parametry dynamiczne** są co prawda dokładniejsze, ale wymagają zbierania danych i przeliczania na bieżąco. Jeśli tylko jest to możliwe i prowadzi do dobrego przybliżenia, opłaca się używać **parametrów statycznych**.

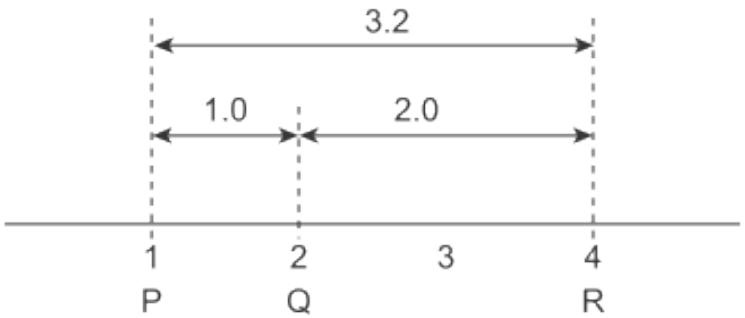
Rozwiązanie: Każdego klienta (lub grupę klientów) umieszczamy w N-wymiarowej przestrzeni, takiej żeby $d(P, Q)$ była rozsądnym przybliżeniem czasów oczekiwania pomiędzy P i Q.

Estymacja metryk: czas oczekiwania klienta 2/3

- Potrzeba $N + 1$ punktów orientacyjnych (punktów odniesienia), żeby obliczyć jednoznacznie pozycję w przestrzeni N-wymiarowej.



  - Pomiary czasów oczekiwania się podlegają fluktuacjom. Ponadto mogą być ze sobę niespójne:



Estymacja metryk: czas oczekiwania klienta 3/3

Rozwiązanie: Niech L punktów orientacyjnych zmierzy czas oczekiwania $d(b_i, b_j)$ pomiędzy sobą. Niech koordynator ustali pozycje tych punktów minimalizując

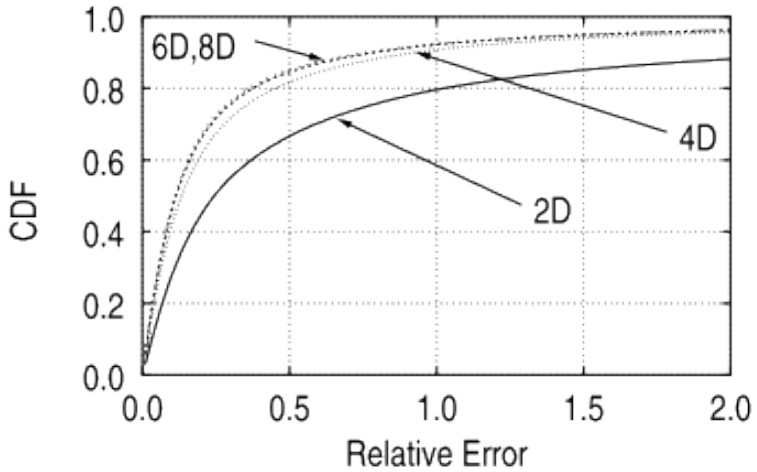
$$\sum_{i=1}^{L} \sum_{j=i+1}^{L} \left[\frac{d(b_i, b_j) - \hat{d}(b_i, b_j)}{d(b_i, b_j)} \right]^2,$$

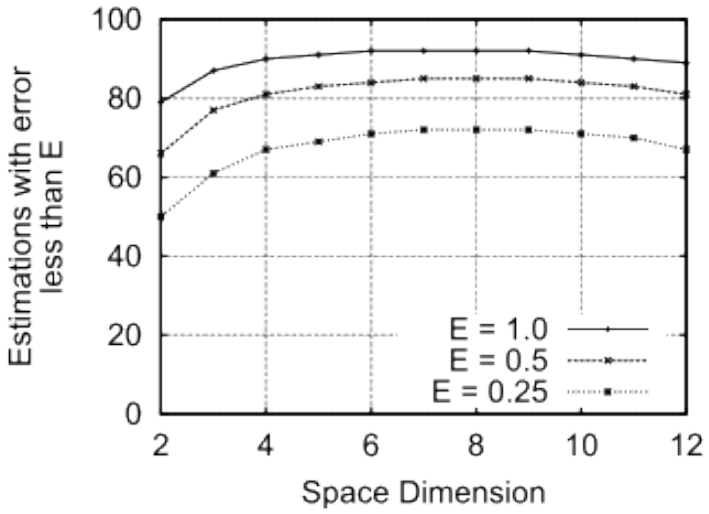
gdzie $\hat{d}(b_i, b_j)$ jest odległością do punktu b_i przy obecnie wyliczonych współrzędnych dla b_j.

Następnie każdy węzeł P minimalizuje

$$\varepsilon = \sum_{i=1}^{L} \left[\frac{d(b_i, P) - \hat{d}(b_i, P)}{d(b_i, P)} \right]^2$$

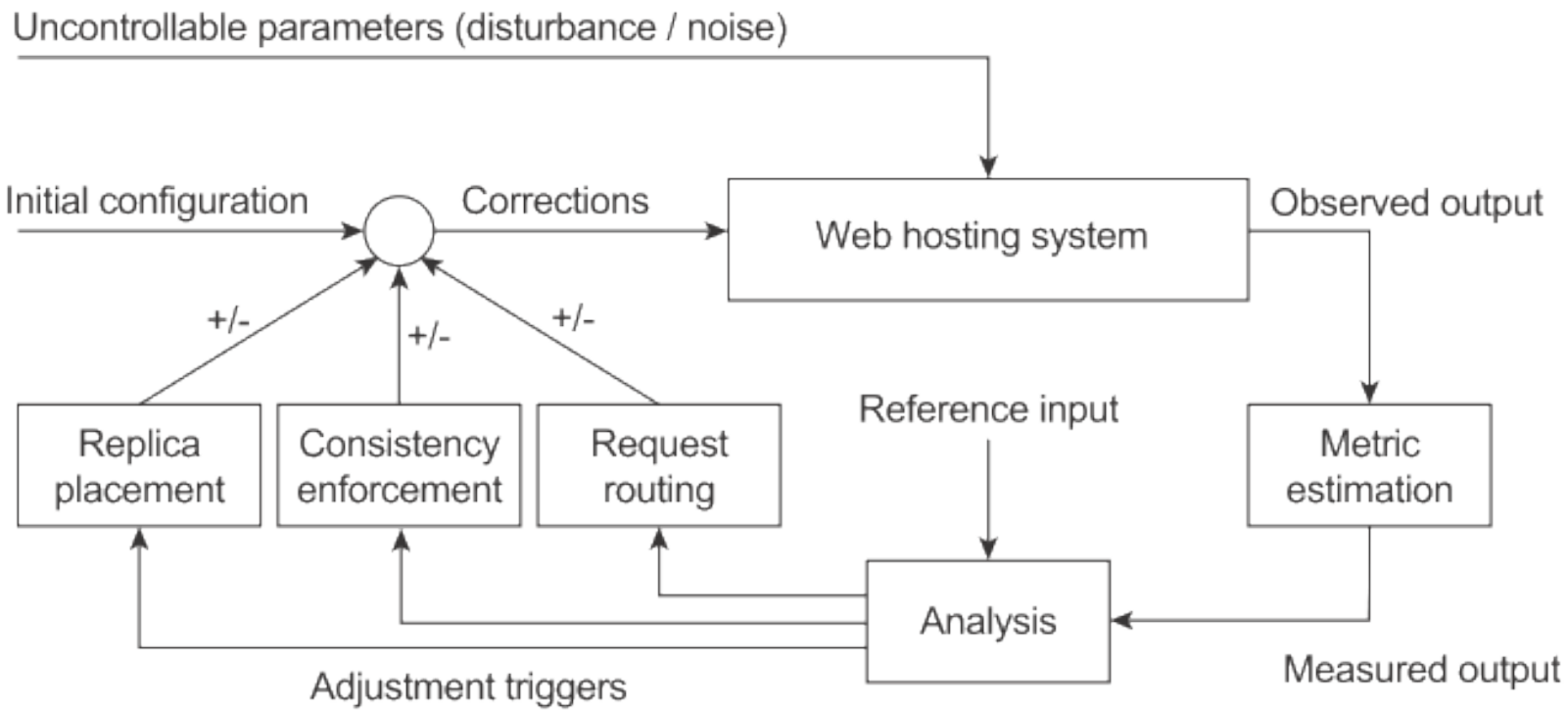
Ile wymiarów ma Internet?





$N = 6$ wymiarów wystarcza, żeby błąd był na satysfakcjonującym poziomie.

Powrót pętli ze sprzężeniem zwrotnym



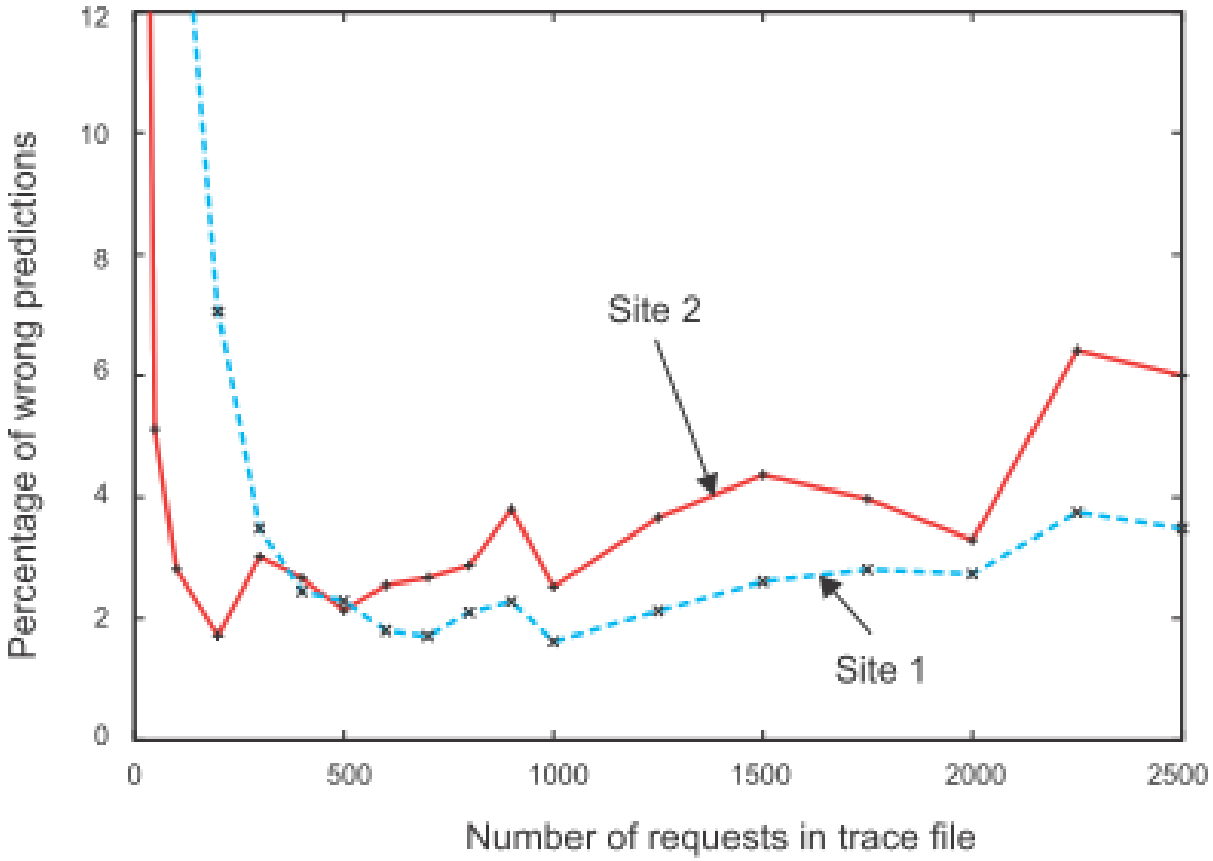
Wybór strategii

Pomysł: Nie mamy modelu analitycznego, więc wykorzystajmy symulacje różnych scenariuszy na podstawie śladów (trace-driven simulations). Np. Globule:

- Każda symulacja (czyli ewaluacja strategii) dla jednego dokumentu trwała 140 ms.
- Branie pod uwagę historii zmian strategii (dla zredukowania liczby ewaluowanych strategii) może przynieść 10-krotne usprawnienie.

Wniosek: Analiza typu „*what-if*” w trakcie działania systemu jest możliwa, jeśli ograniczymy liczbę strategii.
Obserwacja: Badania pokazują, że nawet „nudne” strony potrzebują ciągłego dostosowywania strategii.

Jak długi powinien być ślad?



Wygląda na to, że 500 zapytań jest bliskie wartości optymalnej.
Ale czy to się generalizuje na inne eksperymenty?

Spis treści

Wnioski

- Replikacja w Internecie wymaga dostosowywania strategii zarówno dla konkretnego dokumentu jak i do chwilowych warunków.
- Ciężko z góry przewidzieć, co będzie najlepsze. Dlatego najlepiej sprawdzają się systemy oparte na dynamicznej analizie własnego stanu.
- Trzeba umieć zaobserwować i zmierzyć stan systemu, porównać go z oczekiwaną wydajnością idealną i przewidzieć odpowiednie strategie dostosowywania.

O czym nie było?

- Zawartość dynamiczna – technologie typu Edge Side Includes (ESI).
- Replikacja aplikacji webowych – całkowita lub częściowa replikacja bazy danych, content-blind i content-aware cache.
- Kiedy dokładnie dokonywać zmiany w strategii – periodycznie, aperiodycznie, hybrydowo.

Bibliografia

1. G. Pierre, M. van Steen and A. Tanenbaum. "Dynamically Selecting Optimal Distribution Strategies for Web Documents." *IEEE Trans. Comp.*, 51(6):637-651, Czerwiec 2002.
2. S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. "Replication for Web Hosting Systems." *ACM Comput. Surv.*, 36(3):1–44, Wrzesień 2004.
3. G. Pierre, M. van Steen. "Globule: A Collaborative Content Delivery Network." *IEEE Communications Magazine* 44(8):127-133, Sierpień 2006.
4. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. "Globally Distributed Content Delivery." *IEEE Internet Computing* 6, 5, 50–58., Wrzesień 2002.