Google File System II

Marek Dzikiewicz

Plan prezentacji

1. GFS
2. Problemy z GFS
3. GFS2 i Caffeine

Google File System

- rozproszony system plików dla aplikacji przetwarzających duże ilości danych
- powstał na potrzeby systemu indeksowania firmy Google
- wydajny (obsługuje dużą liczbę klientów)
- skalowalny
- odporny na awarie
- obsługuje niedrogi sprzęt

Założenia

- tania, podatna na awarie platforma sprzętowa
  - wymagana ciągła diagnoza oraz automatyczne naprawianie błędów
- kilka milionów dużych (ok. 100 MB) plików
  - małe pliki też muszą być obsługiwane, ale bez optymalizacji
- wydajność wsadowa ważniejsza od czasu odpowiedzi

Założenia

- duże odczyty sekwencyjne oraz drobne odczyty losowe
- zapisy sekwencyjne na końcu pliku
  - zapisy w środku pliku też muszą być obsługiwane, ale bez optymalizacji
- współbieżny dostęp do plików
  - także do zapisu (dopisywania)!

Serwery GFS

- GFS składa się z dwóch rodzajów serwerów
- GFS Master (zarządca)
  - zarządca GFS, przechowuje metadane
  - jedna instancja
- GFS Chunkserver
  - przechowuje zawartość plików
  - wiele instancji

Organizacja systemu plików

- Pliki są zorganizowane hierarchicznie w katalogi
- Brak implementacji standardu POSIX
- Podstawowe operacje:
  create, *delete*, *open*, *close*, *read*, *write*
- Dodatkowe operacje:
  snapshot, record append

Przechowywanie plików

- Pliki są dzielone na kawałki (ang. *chunks*)
- Każdy kawałek ma stały rozmiar (64 MB)
- Każdy kawałek jest identyfikowany przez unikalną 64-bitową liczbę
- Kawałki są przechowywane w standardowym systemie plików (na maszynach chunkserver)
  - Brak przechowywania danych w pamięci pomocniczej

Odporność na awarie

- Źródła awarii
  - awarie sprzętowe (dyski, serwery, itp.)
  - problemy sieciowe (np. uszkodzone dane)
  - błędy w aplikacjach (np. systemy operacyjne)
  - i wiele innych...
- Strategie
  - replikacja danych (każdy kawałek jest co najmniej potrójnie replikowany)
  - wykrywanie błędów poprzez sumy kontrolne

GFS Master

- Przechowuje metadane systemu plików w pamięci operacyjnej
  - przestrzeń nazw
  - listy kontroli dostępu (ACL)
  - mapowanie plików na kawałki
  - lokalizacje kawałków i ich replik
- Przetwarza logi operacyjne
- Realizuje operacje na metadanych plików

GFS Master

- Zarządza serwerami kawałków (ang. *chunkserver*)
  - także wykrywa stan i dostępność chunkserwerów
- Przeprowadza operacje globalne
  - replikacja
  - odśmiecanie
  - weryfikacja systemu plików
  - migracja kawłków między chunkserwerami

GFS Chunkserver

- Przechowuje repliki kawałków jako pliki w standardowym systemie plików
  - leniwe rozszerzanie plików
- Obsługuje żądania odczytu i zapisu danych

Architektura GFS

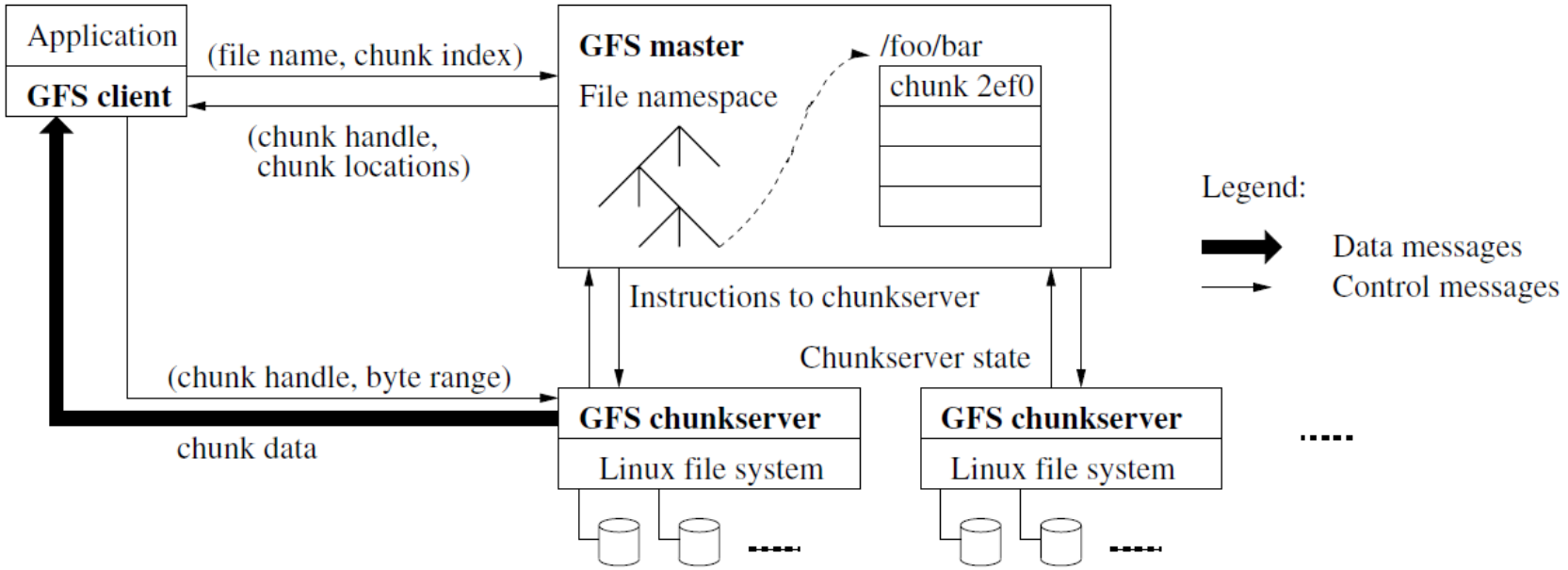


Figure 1: GFS Architecture

Źródło: http://labs.google.com/papers/gfs-sosp2003.pdf

Rodzaje przechowywanych danych

- małe obiekty zagregowane w duże pliki
  - przechowywanie osobno miliardów małych obiektów jest niewykonalne!
- strumienie danych do przetworzenia na różnych maszynach
  - np. kolejki producent-konsument
- dane archiwalne

Formaty przechowywanych danych

- RecordIO
  - sekwencja rekordów zmiennej długości
  - używany do przechowywania dzienników (ang. *logs*)
- SSTable (Sorted String Table)
  - posortowana sekwencja par (klucz, wartość)
  - zawiera indeks
  - niemodyfikowalna struktura danych
- Oba formaty obsługują kompresję oraz sumy kontrolne

Problemy związane z GFS

- podatność na awarie
- ograniczona skalowalność
- ograniczona liczba plików
- opóźnienia żądań
- spójność danych

Problemy związane z GFS

„The original consumer of all our earliest GFS versions was basically this tremendously large crawling and indexing system. The second wave came when our quality team and research groups started using GFS rather aggressively—and basically, they were all looking to use GFS to store large data sets. And then, before long, we had 50 users, all of whom required a little support from time to time so they'd all keep playing nicely with each other."

Howard Gobioff
(jeden z architektów GFS'a)

Pojedynczy master - wady

- podatny na awarie (ang. *single point-of-failure*)
- ogranicza wydajność
- ogranicza skalowalność

Pojedynczy master - zalety

- upraszcza implementację
  - mniejsze prawdopodobieństwo wystąpienia błędów implementacyjnych
- możliwa implementacja w krótkim czasie
- master posiada „globalną” informację o systemie
  - możliwa implementacja złożynach algorytmów rozmieszczania kawałków
  - łatwiejsza implementacja replikacji, odśmiecania, itp.

Podatność na awarie

- stanowi problem w przypadku internetowych aplikacji klienckich
  - np. Gmail, YouTube
- mniej istotny w aplikacjach przetwarzających dane wsadowo
  - np. MapReduce, indeksowanie danych, aplikacje typu web crawler

Podatność na awarie

- częściowe rozwiązanie problemu – wiele serwerów zapasowych oraz sprawny mechanizm przełączania mastera (ang. *master-failover*)
- czas potrzebny do przełączenia mastera
  - w pierwszej implementacji: ok. 60 minut
  - aktualnie: ok. 10 sekund

Ograniczona skalowlaność

- Problemy z wydajnością wystąpiły po zwiększeniu ilości przechowywanych danych powyżej 1000 TB
- Stale rosnąca liczba klientów powodowała coraz większe obciążenie mastera
- Aplikacje typu MapReduce mogą wykonywać liczne, współbieżne operacje masterze
  - np. jednoczesne otworzenie kilku tysięcy plików

Organiczona liczba plików

- metadane wszystkich plików systemu są przechowywane w pamięci operacyjnej mastera
- master przechowuje metadane dla każdego pliku oraz jego kawałków
- pliki przechowywane w GFS nie powinne być mniejsze niż 64 MB
  - problem dla niektórych aplikacji

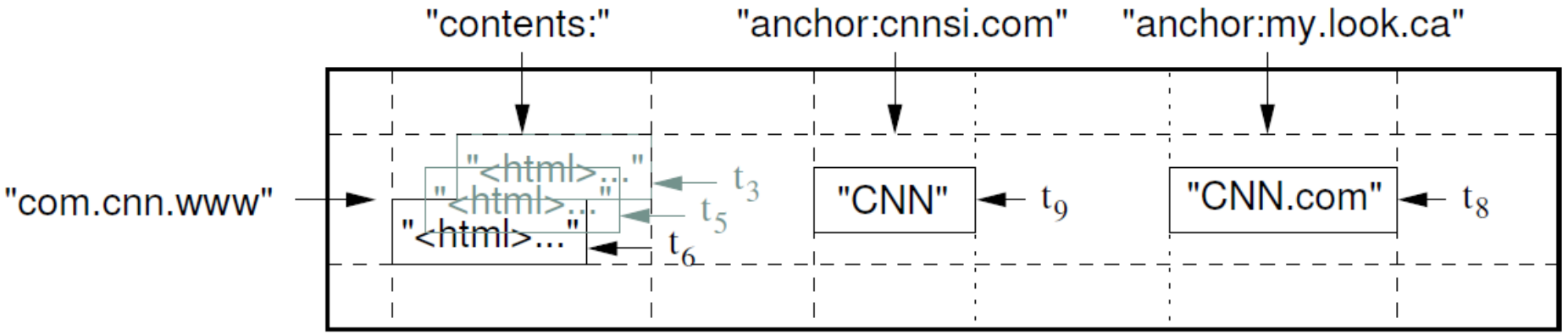
Organiczona liczba plików

- możliwe rozwiązania
  - małe pliki można przechowywać pakując je razem (można do tego użyć bazy danych, np. BigTable)
  - rozproszenie mastera na wiele maszyn fizycznych

BigTable

- strukturalna baza danych oparta o GFS
  - nie relacyjna
  - bardzo skalowalna
- przechowuje dane w postaci (klucz, wartość)
- często wykorzystywana jako warstwa pośrednia pomiędzy aplikacjami internetowymi a GFS

BigTable



- dane są przechowywane w wielo-wymiarowych (wiersz, kolumna, znacznik czasowy) tabelach
- tabele są dzielone na tzw. tablety, rozpraszane i przechowywane w GFS

Źródło: http://labs.google.com/papers/bigtable.html

Opóźnienia żądań

- GFS jest zoptymalizowany na przetwarzanie wsadowe dużej ilości danych
  - brak nacisku na szybkie zakończenie operacji wejścia/wyjścia
  - często wykorzystywane są kolejki
  - blokady kawałków w wyniku awarii
- problem w przypadku internetowych aplikacji klienckich
  - np. Gmail

Opóźnienia żądań

- Możliwe rozwiązania
  - użycie BigTable
  - ukrycie zawodności GFS w warstwie aplikacji (np. podwójny dziennik transakcyjny w BigTable)
  - rozproszony master

Podział systemu na komórki

- Częściowe rozwiązanie problemów powodowanych przez pojedynczego mastera
- Podział pojedynczego centrum danych na komórki GFS (ang. *cells*)
- Każda komórka jest osobną instancją GFS i posiada własnego mastera
- Dane są rozproszone na wiele komórek
  - implementacja rozpraszania danych w aplikacji

Problem spójności danych

- odczytane dane mogą być różne w zależności od repliki
  - w szczególności w wypadku awarii klienta

Problem spójności danych

- „luźna” semantyka RecordAppend
  - dane mogą być zapisane w pliku wiele razy
  - dane mogą być zapisane w różnej kolejności
  - aplikacja musi być tego świadoma i implementować odpowiednią obsługę
  - rozwiązanie: kolejkowanie żądań przez proces weryfikujący poprawność zapisanych danych

GFS2 codename Colossus

- nowa wersja Google File System
- rozwijana przez ok. 2 lata
- wykorzystana po raz pierwszy w Google Caffeine (2009/2010)

GFS2 codename Colossus

- rozproszony master
  - silnie związany z BigTable
- umożliwia przechowywanie dużej ilości plików
  - mniejsze kawałki plików (1 MB)
- automatyzacja
  - narzędzia do autonomicznej diagnozy systemu
  - przechowywanie danych historycznych
- rozpraszanie uwzględniające lokalizacje centrów danych

Google Caffeine

- nowy system do budowania indeksu stron internetowych (dla wyszukiwarki Google)
- ogłoszony w sierpniu 2009 r.
- umożliwia inkrementalną rozbudowę indeksu
- odejście od indeksowania opartego o MapReduce

Google Caffeine

- Problemy z budowaniem indeksu opartym o MapReduce
  - aktualizacja indeksu wymaga przetworzenia nowych oraz istniejących danych
  - długość operacji uaktualniania proporcjonalna do wielkości indeksu
  - długi czas między odnalezieniem strony, a uwzględnieniem jej w wynikach wyszukiwarki

Percolator

- Caffeine opiera się na zmodyfikowanej wersji bazy danych BigTable, o nazwie Percolator
- Nowe funkcjonalności
  - wielowierszowe transakcje ACID (izolacja typu snapshot)
  - obserwatorów (ang. *observers*), czyli programów uruchamiane po modyfikacji danych w tabeli

Działanie Caffeine

1. indeks jest modyfikowany od razu przy przeglądaniu Internetu
2. nowa lub zmodyfikowana pozycja indeksu powoduje wykonanie kodu obserwatorów
3. kod obserwatorów uaktualnia kolejne pozycje indeksu

Percolator

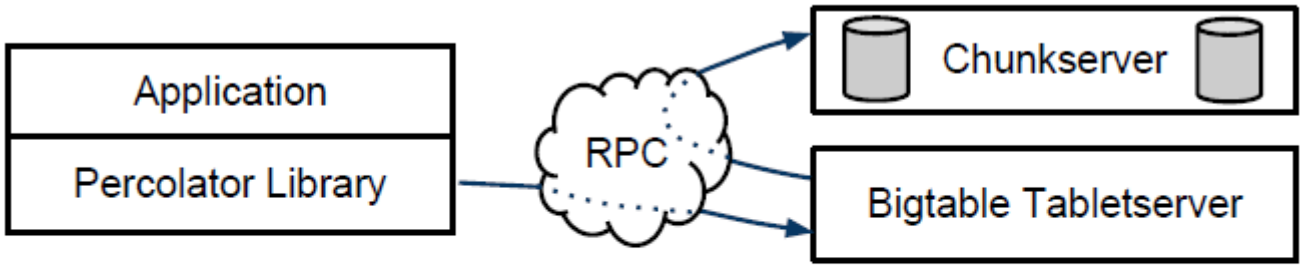


Figure 1: Percolator and its dependencies

Źródło: http://research.google.com/pubs/archive/36726.pdf

Zalety Caffeine

- 50% świeższe wyniki w wyszukiwarce
  - w poprzednim systemie przetwarzanie indeksu zajmowało 2-3 dni
- dodawanie danych do indeksu nie wymaga przetwarzania całego indeksu
  - lepsza skalowalność

Podsumowanie

- GFS
  - oparty na prostym projekcie
  - niewątpliwie odniósł sukces
  - sprawdził się w warunkach znacznie większego obciążenia niż przewidywali jego twórcy
- GFS2
  - rozwiązuje problemy GFS
  - naturalna kontynuacja GFS gwarantująca kontynuację istnienia systemu

Pytania

?

Bibliografia

- Google File System
  http://labs.google.com/papers/gfs-sosp2003.pdf
- ACM Queue Case Study - GFS: Evolution on Fast-forward
  http://queue.acm.org/detail.cfm?id=1594206
- Sean Quinlan, „Storage at Scale”
  http://blip.tv/file/1027764
- Google File System II: Dawn of the Multiplying Master Nodes
  http://www.theregister.co.uk/2009/08/12/google_file_system_part_deux
- Google Works New File System into Caffeine Plans
  http://www.dailytech.com/Google+Works+New+File+System+into+Caffeine+Plans/article16004.htm#cmt483256

Bibliografia

- Google Testing New Storage System for 'Caffeine, http://www.eweek.com/c/a/Data-Storage/Google-Testing-New-Storage-System-for-Caffeine-553512/
- Google search index splits with MapReduce http://www.theregister.co.uk/2010/09/09/google_caffeine_explained
- Bigtable: A Distributed Storage System for Structured Data http://labs.google.com/papers/bigtable.html
- Large-scale Incremental Processing Using Distributed Transactions and Notifications http://research.google.com/pubs/archive/36726.pdf