SZBD Bigtable

Kamil Anikiej

Uniwersytet Warszawski

9 X 2008

Plan prezentacji

Kilka rzędów wielkości:

- 2.7 - 3.3 GB - rozmiar angielskiej Wikipedii (tekst)
- 5.625 PB - szacowany rozmiar przestrzeni dyskowej w serwerach Googla w 2004 r.
- 200 PB - wszystkie wydrukowane dane na świecie

`http://en.wikipedia.org/wiki/Orders_of_magnitude_(data)`

Bigtable - dlaczego powstał

Na początku roku 2004 Google rozpoczyna pracę nad SZBD Bigtable. Główne cele tego projektu to:

1. skalowalność do petabajtów danych na tysiącach komputerów
2. mnogość zastosowań (Bigtable znajduje aktualnie zastosowanie w większości serwisów Googla)
3. wysoka wydajność
4. wysoka stabilność

Konwencjonale podejście do wspomnianych wyzwań

Wierszowo Zorientowana Baza Danych (czyli typu MySQL itp.)

Zalety:

- powszechna dostępność
- sprawdzona w wielu zastosowaniach
- ...(wiele innych)

Konwencjonale podejście do wspomnianych wyzwań

Wierszowo Zorientowana Baza Danych (czyli typu MySQL itp.)

Zalety:

- powszechna dostępność
- sprawdzona w wielu zastosowaniach
- ...(wiele innych)

Wady:

- słaba skalowalność, gdy chce się wykorzystywać wszystkie funkcjonalności (np. JOINY)
- "sztywne" schematy tabel
- limit na liczbę kolumn

Konwencjonale podejście do wspomnianych wyzwań

Inne podejście, trochę mniej konwencjonalne, to *Kolumnowo Zorientowana Baza Danych*.

Kolumnowo Zorientowana Baza Danych

Rozpatrzmy następujący przykład:

EmpId	Lastname	Firstname	Salary
1	Smith	Joe	40000
2	Jones	Mary	50000
3	Johnson	Cathy	44000

W Wierszowo Zorientowanej Bazie Danych tabela ta mapowana jest na dysk następująco:
`1,Smith,Joe,40000;2,Jones,Mary,50000;3,Johnson,....;`

Jest to uproszczenie - w praktyce nie jest to takie proste. Przykład wzięty z

`http://en.wikipedia.org/wiki/Column-oriented_DBMS`

Kolumnowo Zorientowana Baza Danych

Rozpatrzmy następujący przykład:

EmpId	Lastname	Firstname	Salary
1	Smith	Joe	40000
2	Jones	Mary	50000
3	Johnson	Cathy	44000

W Wierszowo Zorientowanej Bazie Danych tabela ta mapowana jest na dysk następująco:
`1,Smith,Joe,40000;2,Jones,Mary,50000;3,Johnson,....;`
Natomiast w Kolumnowo Zorientowanej Bazie Danych tabela:
`1,2,3;Smith,Jones,Johnson;Joe,Mary,Cathy;40000,....;`

Jest to uproszczenie - w praktyce nie jest to takie proste. Przykład wzięty z

`http://en.wikipedia.org/wiki/Column-oriented_DBMS`

Kolumnowo Zorientowana Baza Danych

Kolumnowo Zorientowana Baza Danych ma podstawową zaletę wynikającą z takiej implementacji:

obliczenia, które muszą zostać wykonane dla *dużej* liczby wierszy, ale tylko dla pewnego *małego* podzbioru kolumn, są szybkie.

Intuicja - Bigtable jako arkusz w OOCalc

Rysunek: Bigtable jako arkusz w OOCalc. Czwarty wymiar (znacznik czasowy - *timestamp*) jest niewidoczny

29						
30						
31						
32			:striped	:fourlegs	:velvet	:male
33		Zebra	803	212	0	6
34		Ottoman	1	343	298	0
35		Fedora	3	0	145	34
36						
37						
38						
39						
40						
41						

TAG / REVIEW / PRICE

W skrócie

Przykład - sklep internetowy:

key	tag:striped	tag:fourlegs	tag:velvet	tag:male	review:joey	review:susan	price:US	price:EU
Zebra	803	212	0	6	"I like my zebra"		25.00	20.00
Ottoman	1	343	298	0		"this is an excellent ottoman"	200.00	175.00
Fedora	3	0	145	34		"a very fine hat"	50.00	40.00

W skrócie

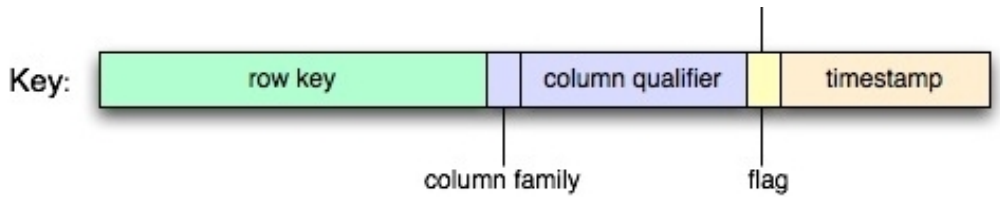
Przykład - sklep internetowy:

key	tag:striped	tag:fourlegs	tag:velvet	tag:male	review:joey	review:susan	price:US	price:EU
Zebra	803	212	0	6	"I like my zebra"		25.00	20.00
Ottoman	1	343	298	0		"this is an excellent ottoman"	200.00	175.00
Fedora	3	0	145	34		"a very fine hat"	50.00	40.00

Bigtable można rozumieć jako map, znany z STLa. Wartością każdej z krotek jest ciąg bajtów (wartości w mapie nie posiadają typów).

$$(\text{row:string, column:string, time:int64}) -> \text{string}$$

Rysunek: Co składa się na klucz krotki: (źródło: [1])



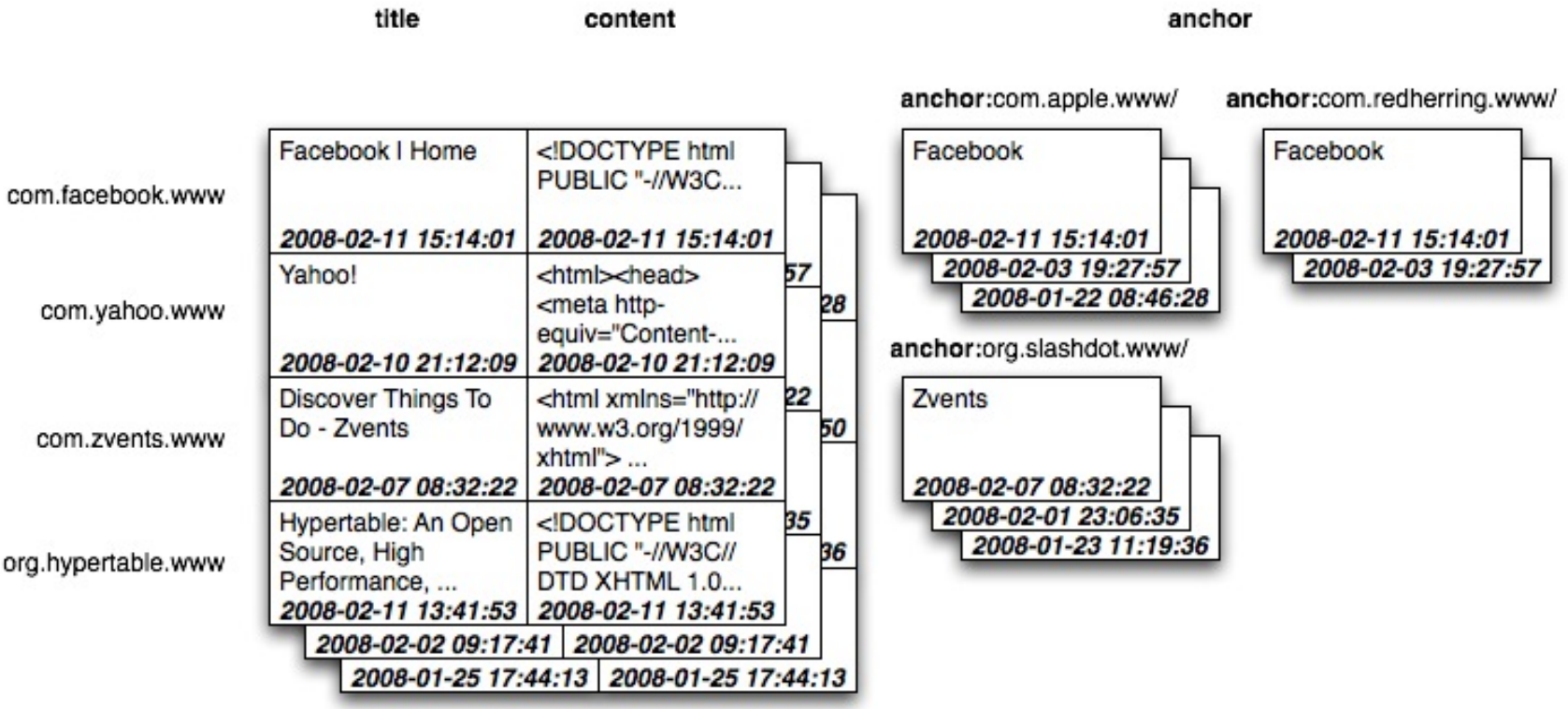
Klucz krotki

Na klucz krotki składa się:

- Identyfikator wiersza (*row key* z poprzedniego slajdu), po którym dane są sortowane.
- Rodzina kolumn (ang. *column family*) - przypomina dobrze nam znaną kolumnę z relacyjnych baz danych. Liczba rodzin kolumn powinna być niewielka (co najwyżej setki) i rzadko się zmieniać. Dane przechowywane w rodzinie kolumn są tego samego typu.
- Kwalifikator kolumny (ang. *qualifier*) - w jednej rodzinie kolumn może być nieskończenie wiele kwalifikatorów. Kwalifikator, wraz z nazwą rodziny, tworzy klucz kolumny, którego składnia jest następująca: *family:qualifier*.
- Znacznik czasowy (*timestamp*) - umożliwia wersjonowanie danej krotki(np. można trzymać dane nie starsze niż 3 dni, lub ich 8 ostatnich wersji).

Organizacja danych - model

Rysunek: (źródło: [1])



Organizacja danych - model

Rysunek: (źródło: [1])

key	value
com.facebook.www title *2008-02-11 15:14:01*	Facebook I Home
com.facebook.www title *2008-02-03 19:27:57*	Facebook I Home
com.facebook.www title *2008-01-22 08:46:28*	Facebook I Home
com.facebook.www content *2008-02-11 15:14:01*	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www content *2008-02-03 19:27:57*	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www content *2008-01-22 08:46:28*	<!DOCTYPE html PUBLIC "-//W3C//DTD...
com.facebook.www anchor:com.apple.www/ *2008-02-11 15:14:01*	Facebook
com.facebook.www anchor:com.apple.www/ *2008-02-03 19:27:57*	Facebook
com.facebook.www anchor:com.apple.www/ *2008-01-22 08:46:28*	Facebook
com.facebook.www anchor:com.redherring.www/ *2008-02-11 15:14:01*	Facebook
com.facebook.www anchor:com.redherring.www/ *2008-02-03 19:27:57*	Facebook
com.yahoo.www title *2008-02-10 21:12:09*	Yahoo!
com.yahoo.www title *2008-02-04 03:46:22*	Yahoo!
com.yahoo.www title *2008-01-22 08:46:28*	Yahoo!
com.yahoo.www content *2008-02-10 21:12:09*	<html><head><meta http-equiv="Content-...
com.yahoo.www content *2008-02-04 03:46:22*	<html><head><meta http-equiv="Content-...
...	...

Organizacja danych - model - Tablets

- Dane są podzielone na zbiory o wielkości ok. 100 MB. Są to przedziały (*tablets*) zdefiniowane przez klucz wiersza pierwszej i ostatniej krotki. Każdy z takich przedziałów jest przydzielony jednej maszynie (*tablet server*).
- Początkowo każda tabela składa się z jednego przedziału, który rozpina wszystkie klucze. W momencie, gdy przedział staje się zbyt duży (jego rozmiar przekracza 200 MB), dzieli się on na dwa przedziały ze środkowym kluczem jako miejscem podziału. Nowy przedział zostanie przydzielony do nowego serwera przez serwer główny.

Organizacja danych - model - SSTable

- Dane przechowywane są w plikach w specjalnym formacie o nazwie *SSTable*
- SSTable to posortowane odwzorowanie z kluczy do ich wartości, gdzie zarówno klucze jak i wartości są po prostu ciągami bajtów.

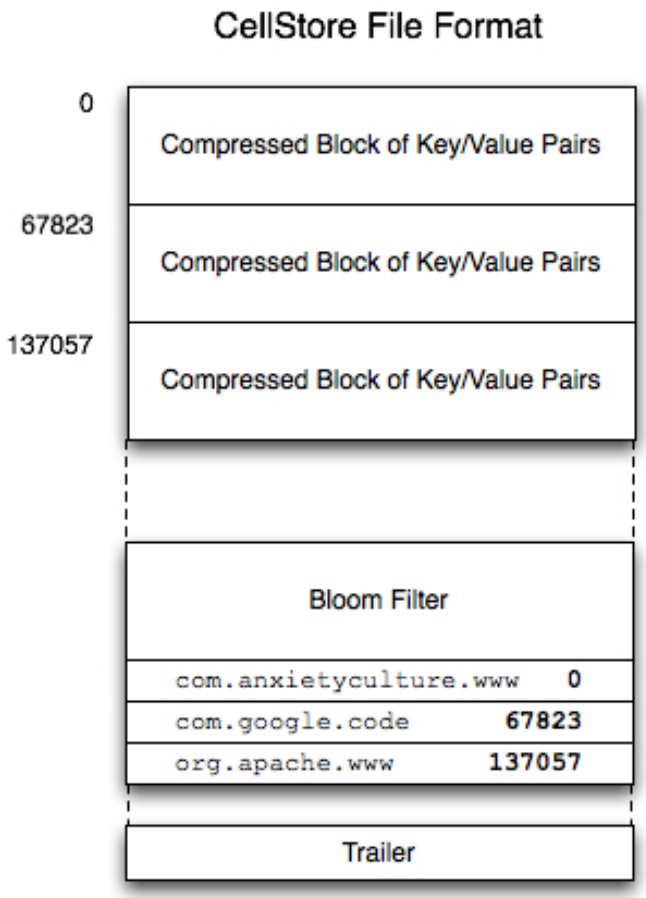
Organizacja danych - model - SSTable

- Pliki w formacie SSTable są podzielone na bloki po 64 KB każdy.
- Na końcu pliku znajduje się indeks bloków, który po otwarciu pliku jest w całości kopiowany do pamięci. Indeks ten jest zbiorem par następującej postaci:

  (ostatni klucz w bloku, położenie bloku)
- Odczyt wartości krotki z danego pliku odbywa się poprzez wyszukanie przedziału, w którym znajduje się dana wartość klucza - tj. zdefiniowanie właściwego bloku (wykorzystywany jest zwykły algorytm binarny) oraz na odczytaniu tego bloku z dysku.
- W pliku może być utworzony filtr Blooma. Jego zastosowanie jest omawiane na dalszym slajdzie

Organizacja danych - reprezentacja fizyczna

Rysunek: (źródło: [1])



Bigtable - z czego jest zbudowany

Bigtable wykorzystuje inne komponenty infrastruktury Googla.

- GFS - do przechowywania logów oraz plików z danymi.
- Chubby - mechanizm blokad oraz przechowywanie małych plików.

GFS - w skrócie

GFS to rozproszony system plików, który przez Bigtable jest wykorzystywany do przechowywania logów oraz plików z danymi.

- GFS to jeden serwer główny i kilkaset do kilku tysięcy serwerów kawałków (chunk-servers)
- Serwer główny (*master*) przechowuje dane o położeniu bloków (dlatego są one duże 64MB) i monitoruje dostępność chunk-serwerów.
- Każdy blok jest powielany w co najmniej 3 kopiach.

Chubby - w skrócie

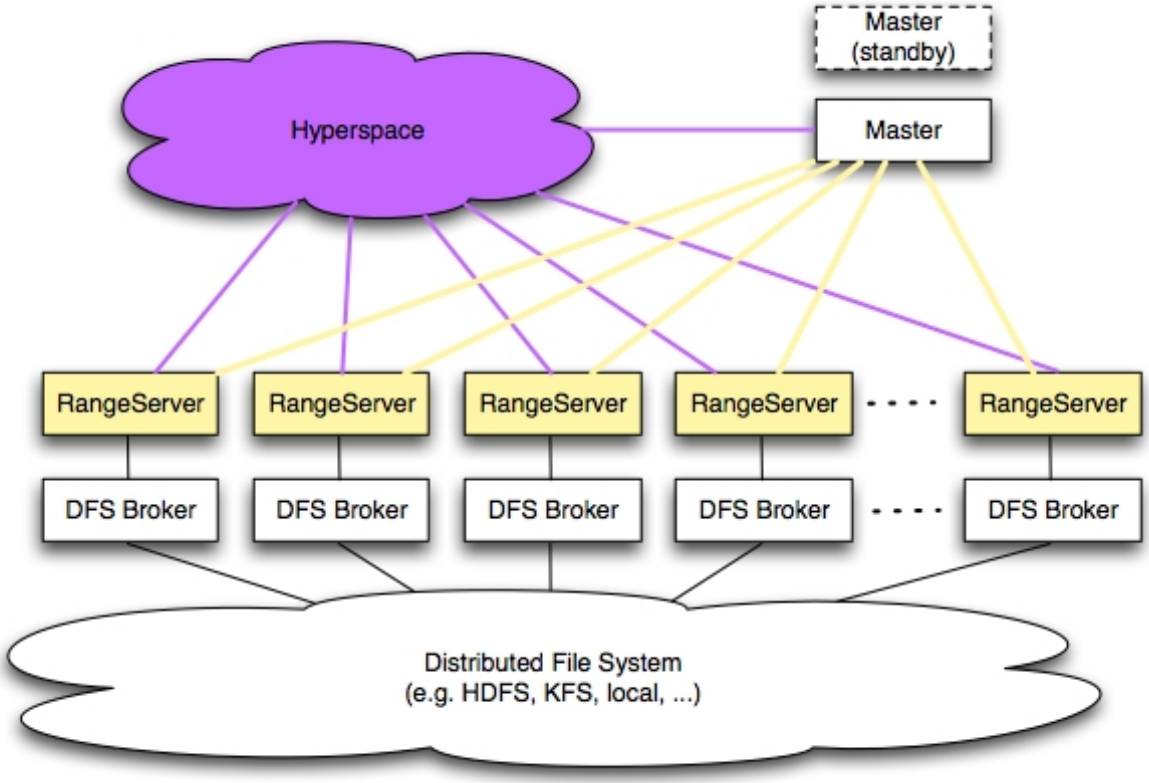
Chubby to odporny na błędy zarządca mechanizmu blokad dla rozproszonych systemów (jest wykorzystywany przez GFS i Bigtable). Służy także do przechowywania małych plików. Do jednego klastra jest przypisany jeden Chubby.

- Odporność na awarie jest uzyskiwana przez replikację - na serwis ten przypada pięc replik, z których jedna jest wybierana do aktywnej obsługi zapytań (pozostałe repliki na zapytania klientów odpowiadają, adresem IP aktywnej repliki). Serwis przestaje działać, gdy większość replik jest niedostępna.
- Dla zapewnienia spójności danych między replikami wykorzystywany jest algorytm Paxos
- Zapisy i odczyty do plików udostępnianych przez serwis Chubby są atomowe.

0.0047% czasu awarii, gdy pewne dane w Bigtable były niedostępne, było spowodowanych niedostępnością serwisu Chubby. Są to średnio 4ry sekundy dziennie lub niecałe 12 godzin w roku.

Infrastruktura Bigtable na przykładzie Hypertable

Rysunek: Komponenty: Hyperspace (aka Chubby), RangeServer (aka Tablet Server) (źródło: [1])



Serwery przedziałów - *Tablet servers*

- każdy serwer przedziałów jest odpowiedzialny za obsługę żądań zapisu i odczytu od klientów (dane nie przechodzą przez serwer główny)
- jednemu serwerowi może być przydzielonych wiele tysięcy przedziałów.
- nowe serwery przedziałów mogą być łatwo dołączane lub odłączane od klastra. Można dzięki temu szybko reagować na zmiany obciążenia

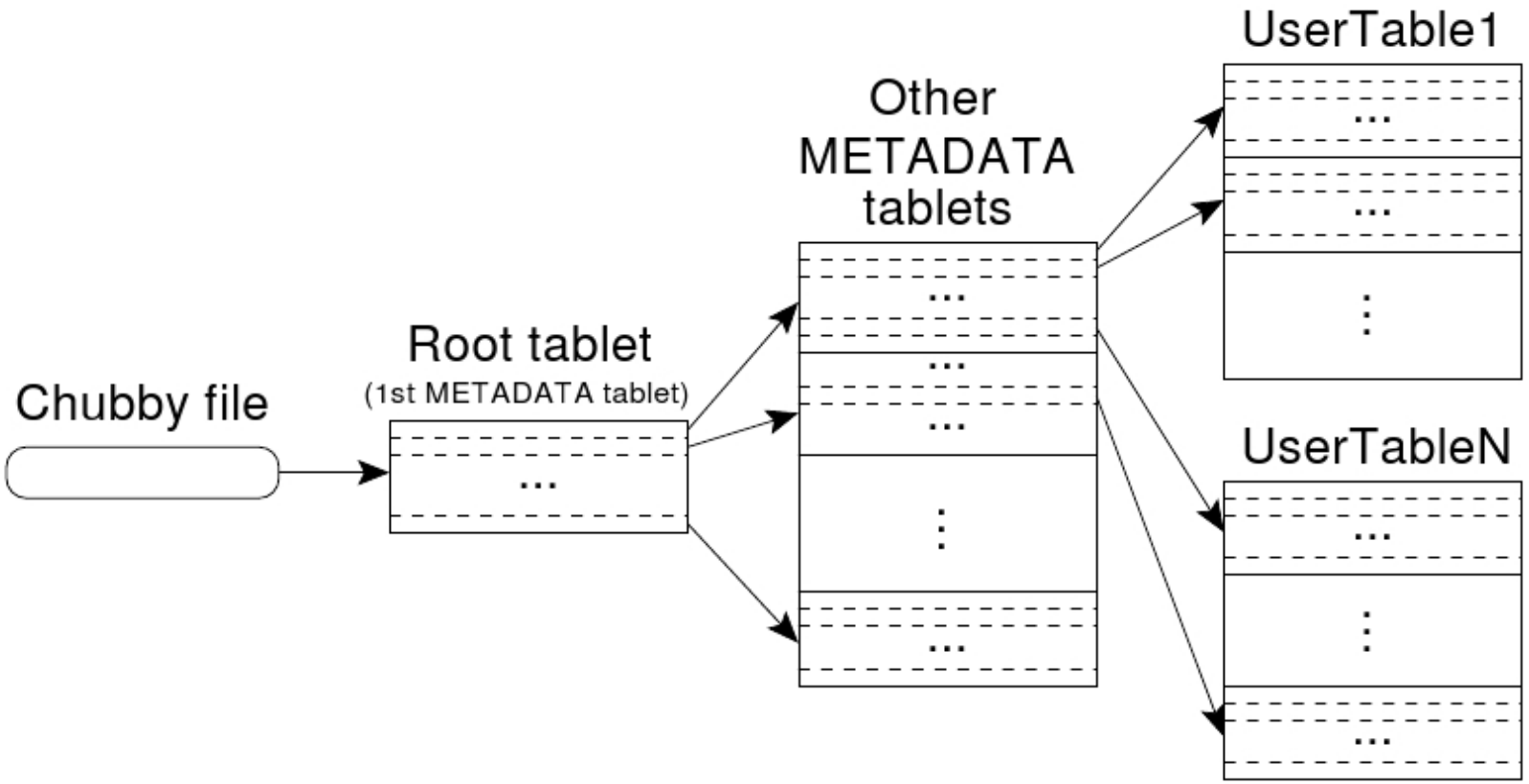
Serwer główny - *master*

Główne zadania serwera głównego to:

- zarządzanie przedziałami kluczy - przydzielane ich do odpowiednich serwerów
- wykrywanie dołączenia i odłączenia serwera przedziałów od klastra
- rozpraszanie obciążenia pomiędzy serwery
- usuwanie niepotrzebnych plików z GFS

Odnajdywanie przedziałów

Rysunek: Odnajdywanie przedziału - hierarchia (źródło: [2])



Odnajdywanie przedziałów

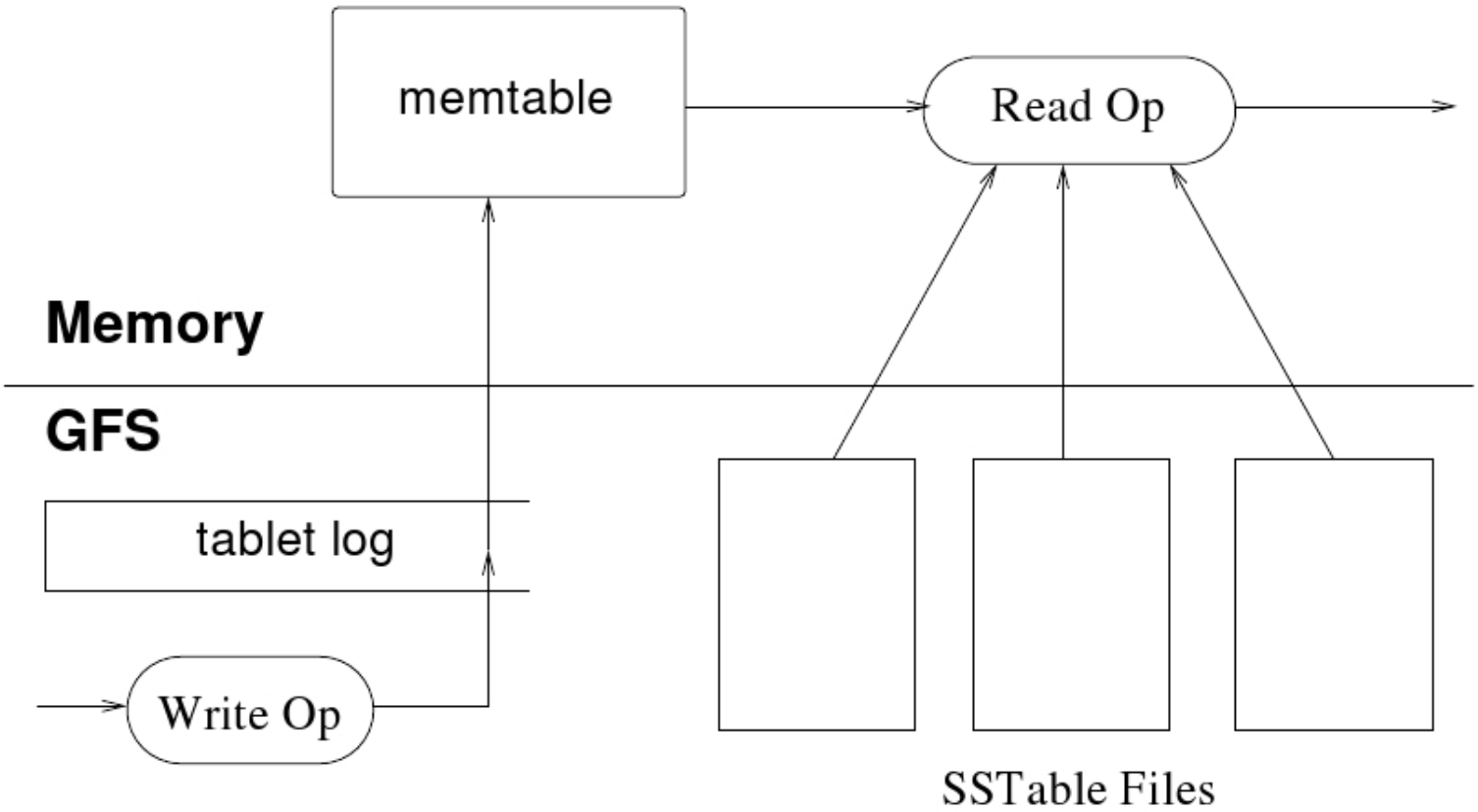
- (*root tablet*) zawiera lokalizacje wszystkich przedziałów wchodzących w skład specjalnej tabeli zawierającej metadane o przedziałach. Jest to tak naprawdę pierwszy przedział tej tabeli. Jest on traktowany specjalnie jedynie w tym sensie, że nigdy nie podlega podziałowi dla zapewnienia co najwyżej trój poziomowej hierarchii.
- tabela metadanych zawiera lokalizację przedziału (listę plików w formacie SSTable, oraz wskaźniki do plików dziennika), który jest definiowany przez nazwę tabeli, do której należy przedział oraz jego końcowy klucz.
- przedziały w tabeli metadanych mają rozmiar co najwyżej 128 MB co, przy średniej wielkości wiersza w tej tabeli wynoszącej 1KB, umożliwia zaadresowanie 2^{34} przedziałów (lub 2^{61} danych przy rozmiarze przedziału wynoszącym 128MB)

Odnajdywanie przedziałów

- klienci przechowują zapamiętane informacje o położeniu przedziału.
- gdy klient nie zna położenia przedziału, albo gdy zapamiętana informacja jest nieaktualna, wtedy rekurencyjnie przeszukuje wyższe poziomy hierarchii.
- w przypadku pustej pamięci podręcznej, klient potrzebuje trzech zapytań (z włączeniem odczytu pliku w serwisie Chubby).
- w przypadku nieaktualnej pamięci liczba ta może wzrosnąć do sześciu zapytań (może nawet więcej, gdy w tym czasie jakaś przedział w tabeli metadanych zmieni położenie)
- dla przyspieszenia tego procesu klient pobiera informacje o innych przedziałach (*prefetching*), a same przedziały tabeli metadanych są przechowywane w pamięci oszczędzając tym samym dostęp do dysku

Przedział w praktyce

Rysunek: Przedział (*tablet*) w praktyce (źródło: [2])



Przedział w praktyce - operacja zapisu

zapis - przykładowy kod (źródło: [2])

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");
// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Przedział w praktyce - operacja zapisu

Gdy serwer odbiera zlecenie do przeprowadzenia operacji zapisu

- sprawdza uprawnienia do przeprowadzenia tej operacji (serwis Chubby)
- operacja jest zapisywana w dzienniku
- po wykonaniu commita zmiana zapisywana jest w pamięci nietrwałej - *memtable*

Przedział w praktyce - operacja odczytu

odczyt - przykładowy kod (źródło: [2])

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
        printf("%s %s %lld %s\n",
        scanner.RowName(),
        stream->ColumnName(),
        stream->MicroTimestamp(),
        stream->Value());
}
```

Przedział w praktyce - operacja odczytu

Gdy serwer odbiera zlecenie do przeprowadzenia operacji odczytu

- sprawdza uprawnienia do przeprowadzenia tej operacji (serwis Chubby)
- operacja odczytu przeprowadzana jest na złączonych plikach SSTable oraz bufora - *memtable*
- każdy z plików SSTable oraz sam bufor są posortowane, zatem złączony widok tych obiektów może być obliczony efektywnie.

Złączenia plików w formacie SSTable

- w momencie, gdy *memtable* stanie się zbyt duże, tworzony jest nowy bufor *memtable*, a stary zapisywany jest na dysku w formacie SSTable
- ma to 2 zalety - zmniejsza rozmiar pamięci używanej przez serwer przedziałów oraz zmniejsza rozmiar danych jakie muszą być odtworzone z dziennika w przypadku awarii.
- gdy utworzonych zostaje zbyt wiele plików SSTable - pliki te, wraz z buforem *memtable* z pamięci serwera są łączone w jeden większy plik SSTable.

Złączenia plików w formacie SSTable - sposób na operację *delete*

Wspomniane łączenie plików SSTable jest sposobem na uproszczenie plików, które, jeżeli były utworzone bezpośrednio z *memtable* mogły zawierać wpisy z żądaniem usunięcia danych. Wpisy te nie były wykonywane od razu, a jedynie przysłaniały poprzednio starsze dane. Łączenie plików SSTable jest szansą na pozbycie się takich wpisów, a przez to odzyskanie zasobów zajmowanych przez usunięte dane.

Złączenia plików w formacie SSTable - sposób na operację *delete*

Wspomniane łączenie plików SSTable jest sposobem na uproszczenie plików, które, jeżeli były utworzone bezpośrednio z *memtable* mogły zawierać wpisy z żądaniem usunięcia danych. Wpisy te nie były wykonywane od razu, a jedynie przysłaniały poprzednio starsze dane. Łączenie plików SSTable jest szansą na pozbycie się takich wpisów, a przez to odzyskanie zasobów zajmowanych przez usunięte dane.

Grupy dostępu

- Klienci mogą łączyć różne rodziny kolumn (*column families*) w grupę dostępu (*locality group*). Plik SSTable jest tworzony oddzielnie dla każdej z grup dostępu. Warto więc rozdzielać dane, które nie są odczytywane razem między różne grupy dostępu.
- np. znaczniki *meta* ze strony WWW mogą być w innej grupie niż treść strony.

Grupy dostępu

- Gdy każda z kolumn jest w osobnej grupie dostępu mamy do czynienia z Kolumnowo Zorientowaną Bazą Danych
- Gdy wszystkie kolumny są w tej samej grupie dostępu mamy do czynienia z Wierszowo Zorientowana Bazą Danych

Grupy dostępu

Grupy dostępu posiadają dodatkowy parametr określający, czy dana grupa jest przechowywana w całości w pamięci, czy może na dysku. Tabele zawierające metadane przechowywane są jak już wspomnieliśmy w pamięci.

Kompresja

- Klienci mogą zdefiniować czy i jak dana grupa dostępu (plik SSTable) jest kompresowana.
- Każdy blok jest kompresowany osobno (powoduje to gorszy współczynnik kompresji, ale odczyty mogą zostać dokonane bez dekompresji całego pliku)
- stosowany jest dwufazowy algorytm kompresji, kompresujący ciągi najpierw w dużym oknie danych, następnie kompresowane są ciągi w małym 16KB oknie.
- Algorytm kompresji jest pomyślany, aby był po pierwsze szybki, a dopiero po drugie wydajny.
- kompresja dokonuje się z prędkością 100-200 MB/s, a dekompresja 400-1000 MB/s.
- współczynnik kompresji to 10 do 1 (zwykły Gzip osiąga współczynnik 4 do 1)

Filtry Blooma

- operacja odczytu angażuje wszystkie pliki SSTable, z których składa się przedział.
- jeżeli pliki te nie są w pamięci możemy zbyt często angażować dysk do odczytów, które nie znajdują szukanej informacji (indeks bloków, który wchodzi w skład pliku SSTable zawiera tylko ostatnie klucze każdego z bloków - pewność o tym czy klucz znajduje się na dysku uzyskujemy dopiero w momencie wczytania szukanego blok z dysku)
- aby zredukować liczbę błędnych odczytów stosuje się filtry Blooma

Dziennik - implementacja

Każdy serwer przedziałów utrzymuje jeden plik będący dziennikiem dla każdego z przedziałów. Osobny plik dla każdego z przedziałów oznaczałby:

- dużą liczbę otwartych plików
- wyszukiwanie tych plików na dysku wraz z każdą ich zmianą.

Dziennik - implementacja

W przypadku awarii serwera przedziałów:

- przedziały przydzielone temu serwerowi są dzielone między inne serwery
- nowy serwer w celu odtworzenia stanu przydzielonego mu przedziału musi odtworzyć wszystkie zmiany z dziennika serwera, który uległ awarii.
- jeżeli przedziały zostały przydzielone 100 nowym serwerom oznaczałoby to, że dziennik jest czytany przez 100 maszyn
- aby temu zapobiec wpisy w dzienniku sortowane są według kluczy:

  $<$tablica, klucz wiersza, numer wpisu w dzienniku$>$
- dzięki temu nowy serwer szybko uzyskuje potrzebne mu dane

Wydajność

Rysunek: Liczba odczytów/zapisów na sekundę wartości o rozmiarze 1000 bajtów, średnio na jeden serwer(źródło: [2])

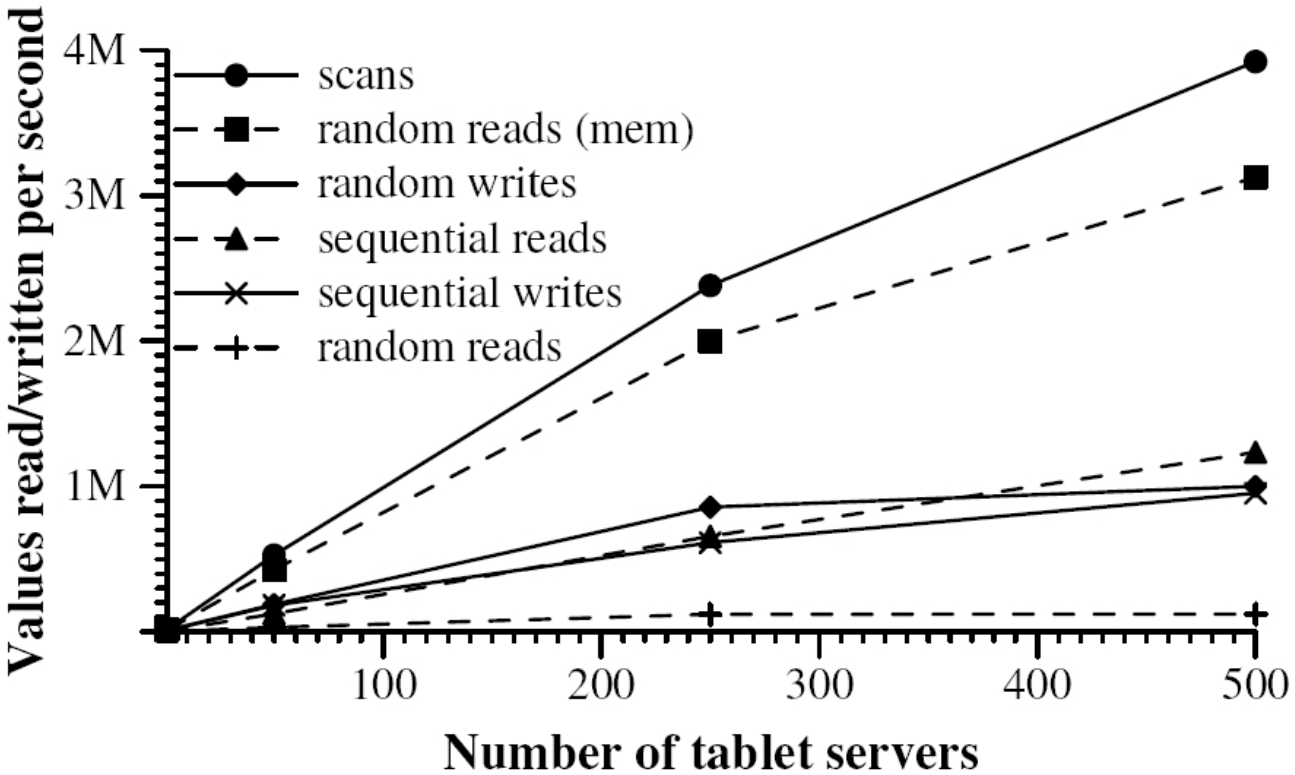
Experiment	# of Tablet Servers			
	1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

Wydajność - przypadek jednego serwera w sieci

- losowe odczyty są wolniejsze o jeden rząd wielkości od innych operacji - każdy odczyt 1000 bajtów powoduje przesłanie bloku wielkości 64KB. Owe 1200 odczytów na sekundą to około 75MB/s danych odczytanych z GFS.
- odczyty z pamięci są znacznie szybsze - nie trzeba za każdym razem pobierać dużego bloku z dysku.
- zapisy (zarówno losowe jak i sekwencyjne) są kilku krotnie szybsze od odczytów - serwer przedziałów grupuje żądania zapisu w dzienniku, dlatego same zapisy do GFS są wydajniejsze.
- sekwencyjne odczyty są szybsze od losowych dzięki wykorzystaniu pamięci podręcznej.
- ostatni test *scan* wypadł najlepiej ponieważ dla jednego RPC od klienta serwer może zwrócić wiele wartości

Wydajność

Rysunek: Liczba odczytów/zapisów na sekundę wartości o rozmiarze 1000 bajtów, sumarycznie (źródło: [2])



Q & A


Architectural overview.
`http://code.google.com/p/hypertable/wiki/ArchitecturalOverview`.


Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Mike Burrows.
Bigtable: A distributed storage system for structured data.
Technical report, Google, Inc., 2006.