**GFS (Google File System)** — to jest duży rozproszony system plików, która może przechowywać i przetwarzać ogromne ilości informacji.

**Historia**

Według niektórych raportów praca nad GFS rozpoczęła się w 2000 roku; ogólne zasady budowy zostały dość szczegółowo opisane w dokumencie przedstawionym na ACM SIGOPS Operating Systems Review w 2003 roku. Aktualizacja GFS drugiej wersji (2009 rok) ma nazwę kodową Colossus.

**Właściwości systemu**

GFS zbudowana jest na podstawie następujących kryteriów:

* System jest zbudowany z dużej ilości zwykłego tanio sprzętu, który często daje się zawiesza. Muszą istnieć monitorowanie awarii i możliwość, aby w przypadku awarii jakiegokolwiek sprzętu przywrócić działanie systemu.
* System musi przechowywać wiele dużych plików. Zazwyczaj kilka milionów plików, każdy od 100 Mb i więcej. Również często mają do czynienia z wiele gigabajtnych plików, które także powinny skutecznie przechowywane. Małe pliki też muszą być, ale dla nich nie optymalizuje pracę systemu.
* Zazwyczaj występują dwa rodzaje czytania: czytanie dużej spójnego fragmentu danych i odczyt małej ilości dowolnych danych. Podczas czytania dużego strumienia danych nagminne jest żądanie fragmentu o wielkości do 1mb i więcej. Takie kolejne operacje od jednego klienta często czytają z rzędu idące kawałki tego samego pliku. Czytanie małego rozmiaru danych, zazwyczaj ma pojemność do kilku kilobajtów. Aplikacji, krytycznych czasu wykonania, musi zgromadzić pewną liczbę takich zapytań i sortować je według przesunięcie od początku pliku. To pozwoli uniknąć podczas czytania spacerze widoku do tyłu-do przodu.
* Często spotykane operacje zapisu dużej spójnego kawałka danych, które należy dopisać do pliku. Zwykle ilości danych do zapisu tego samego rzędu, co i do czytania. Wpisy małych ilości, ale w dowolne miejsce pliku, zazwyczaj wykonywane są nie skuteczne.
* System musi realizować ściśle określony semantykę równoległej pracy wielu klientów, w przypadku, gdy są one jednocześnie próbują dopisać dane w jeden i ten sam plik. Przy tym może się zdarzyć, że trafią jednocześnie setki wniosków o wpis w jeden plik. Aby poradzić sobie z tym, używany proces operacji dodawania danych do pliku, z pewnym synchronizacją. Czyli jeśli zostanie operacja na czytanie, to ona będzie działać, albo do kolejnej operacji zapisu, albo po.
* Wysoka przepustowość jest bardziej korzystne, niż małe opóźnienie. Tak, większość aplikacji w Google preferują pracy z dużymi ilościami danych, przy dużej prędkości, a wykonanie danej operacji

Pliki w GFS są zorganizowane hierarchicznie, przy pomocy katalogów, jak i w każdym innym systemie plików, i są identyfikowane swoją drogą. Z plikami w GFS można wykonywać zwykłe czynności: tworzenie, usuwanie, otwieranie, zamykanie, odczyt i zapis. Ponadto, GFS obsługuje kopii zapasowej, lub zdjęć (snapshot). Można tworzyć takie kopie zapasowe plików lub drzewa katalogów, a także z niewielkimi kosztami: odczytu i zapisu, ogólnie, może być rozciągnięty.

**Architektura GFS**

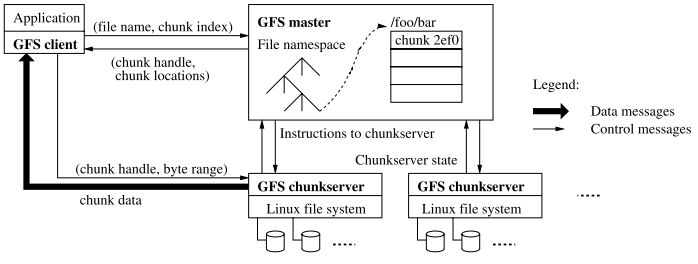
GFS jest przeznaczona do przechowywania dużych ilości danych (petabajtów). Pliki w FS są podzielone na kawałki o stałym rozmiarze (w terminologii GFS – "chunk") w 64 Mb (w ogólnym przypadku).

Pliki danych są przechowywane na chunk-węzłach (w terminologii GFS – "Chunkserver"). Chunkserver na poziomie sprzętowym jest commodity-sprzęt. Dlatego (i nie tylko dlatego) w celu zapewnienia bezpiecznego przechowywania danych dane GFS automatycznie replikowane co najmniej trzy razy.

Poza Chunk-serwer w GFS jest jedynym Master-serwer. Master stanowi węzeł obliczeniowy, odpowiedzialny za koordynację i monitorowanie chunk-serwer. Przez analogię z NameNode w HDFS, Master GFS jest odpowiedzialny za:

* pracy z metadanymi – przestrzeni nazw, file-to-chunk mapping, informacje na temat dostępu do danych;
* operacje na chunk, w tym montaż śmieci ("osieroconych" chunk);
* monitorowanie Chunk-serwera za pośrednictwem hearbeat-wiadomości.

Podobnie jak w przypadku NameNode w HDFS, Master-węzeł GFS jest potencjalne "wąskie gardło" systemu, ma nadmierna ilość obowiązków i jest pojedynczego punktu awarii.



**Struktura GFS**

W systemie istnieją master serwer i chunk-serwera, właściwie, przechowujące dane. Zazwyczaj GFS klaster składa się z jednej głównej maszyny mistrza (master) i wielu maszyn, zawierających fragmenty plików chunk-serwery (chunkservers). Klienci mają dostęp do tych wszystkich maszyn. Pliki w GFS podzielone na kawałki (chunk, można powiedzieć fragment). Chunk ma stały rozmiar, który może być zmieniany. Każdy chunk ma niepowtarzalny i globalny 64 — bitowy klucz, który jest mistrzem w tworzeniu chunk. Chunk-serwery przechowują chunk, jak zwykłe Linux pliki na lokalnym dysku twardym. Dla bezpieczeństwa każdy chunk może być replikowane na innych chunk-serwery. Zwykle stosowane są trzy repliki. Master jest odpowiedzialny za pracę z metadanymi całego systemu plików. Metadane obejmują przestrzeni nazw, informacje o kontroli dostępu do danych, wyświetlanie plików w chunk, i aktualna pozycja chunk. Również mistrz kontroluje całą globalną działalność systemy takie, jak zarządzanie wolnymi chunk, montaż śmieci (zbieranie bardziej niepotrzebnych chunk) i przenoszenie chunk między chunk-serwerami. Mistrz stale wymienia wiadomości (HeartBeat messages) z chunk-serwerami, aby dać instrukcje, i określić ich stan (dowiedz się, czy nadal jeszcze). Klient komunikuje się z mistrzem tylko do wykonywania operacji związanych z metadanymi. Wszystkie operacje z samymi danymi są produkowane bezpośrednio z chunk-serwerami. GFS — system nie obsługuje POSIX API, więc, że deweloperzy nie musiał kontaktować się z dopuszczalnych węzłów poziomem Linux. Twórcy nie korzystają z buforowaniem danych, co prawda, klienci mają buforowane metadane. Na chunk serwerach system operacyjny Linux i tak powoduje buforowania najczęściej używane bloki w pamięci. W ogóle, rezygnacja z pamięci podręcznej pozwala nie myśleć o problemie validity pamięci podręcznej (cache coherence).

**Master**

Korzystanie z jednego mistrza znacznie upraszcza architekturę systemu. Pozwala produkować skomplikowane przenieść chunk, zorganizować replikacji, wykorzystując globalne dane. Wydawałoby się, że posiadanie tylko jednego mistrza musi być wąskim gardłem systemu, ale to nie tak. Klienci nie czytają i nie piszą dane przez kreatora. Zamiast tego, oni pytają u mistrza, z jakim chunk-serwer muszą kontaktować się, a następnie kontaktują się z chunk-serwerami bezpośrednio.

Zastanów się, jak następuje odczyt danych przez klienta. Najpierw, wiedząc rozmiar chunk, nazwę pliku i przesunięcie względem początku pliku, klient określa numer chunk wewnątrz pliku. Następnie wysyła żądanie kapitana, który zawiera nazwę pliku i numer chunk w tym pliku. Kreator wyświetla chunk-serwery, po jednym w każdej replice, które przechowują, którego nam chunk. Również mistrz wydaje klientowi identyfikator chunk.

Następnie klient decyduje, który z replik mu się podoba bardziej (zazwyczaj tej, która jest najbliżej) i wysyła żądanie, składający się z chunk i odsunięcia od początku chunk. Dalsze odczytu danych, nie wymaga interwencji kreatora. W praktyce, jak zwykle, klient na żądanie odczytu zawiera kilka chunk, i kreator daje współrzędne każdego z chunk w jednej odpowiedzi.

Rozmiar chunk jest ważną cechą systemu. Zazwyczaj jest ustawiona na 64 mb, co jest o wiele większy niż rozmiar bloku w zwykłym systemie plików. Oczywiste jest, że jeśli chcesz przechowywać wiele plików, których rozmiar jest mniejszy niż rozmiar chunk, to będziemy zużywały dużo pamięci. Ale wybór takiego dużego rozmiaru chunk wynika z zadaniami, które stanowią firmy Google decydować na swoich klastrach. Jak zwykle, coś wierzyć trzeba do wszystkich dokumentów w internecie, a więc pliki w tych zadaniach bardzo dużych rozmiarów.

**Metadane**

Kreator przechowuje trzy ważne rodzaje metadanych: przestrzeni nazw plików i chunk, wyświetlanie pliku w chunk i położenie replik chunk. Wszystkie metadane są przechowywane w pamięci mistrza. Tak jak metadane są przechowywane w pamięci, operacji kreatora są wykonywane szybko. Stan spraw w systemie master dowie się po prostu i efektywnie. On skanowanie chunk-serwera w tle. Te okresowe skanowania są używane do złożenia śmieci, dodatkowe powtórzeniach, w przypadku stwierdzenia, niedostępne chunk-serwera i przeniesienie chunk, w celu równoważenia obciążenia i ilość wolnego miejsca na dyskach twardych chunk-serwera. Kreator śledzi położenie chunk. Przy starcie chunk-serwera master zapamiętuje go chunk. W trakcie pracy kreatora kontroluje wszystkie ruchy chunk i stanu chunk-serwera. W ten sposób, że posiada wszystkie informacje o położeniu każdego chunk.

Ważną częścią metadanych — to log z operacji. Kreator zapisuje sekwencję operacji krytycznych zmian metadanych. W tym markiery w logu operacji, zależy logicznego czas systemu. To logiczne czas określa wersji plików i chunk. Tak jak log z operacji ważna część, to powinien on być bezpiecznie przechowywane, a wszystkie zmiany w nim powinny stać się widoczne dla klientów, tylko wtedy, gdy zmienią się metadane. Dziennik transakcji jest replikowana na kilku zdalnych maszyn i system reaguje na klienta operację, tylko po zapisaniu tego logu na dysk kreatora i dysków zdalnych maszyn.

Kreator przywraca stan systemu, wykonując dziennika operacji. Log z operacji zachowuje stosunkowo niewielki rozmiar, przy zachowaniu tylko ostatnich transakcji. W trakcie pracy kreator tworzy punkty kontrolne, gdy rozmiar logu przekracza pewnej wartości, i przywrócić system można tylko do najbliższego punktu kontrolnego. Dalej po logowaniu można ponownie odtworzyć niektóre operacje, w ten sposób, system może poruszać się do punktu, który znajduje się między ostatnim punktem kontrolnym i aktualnym czasem.

**Podstawowe zasady**

Podstawowymi zasadami, określonymi przez inżynierów Google GFS są:

* Awarii sprzętu w systemach klastrowych należy traktować jako normę, a nie wyjątek.
* Dane już mają ogromny rozmiar i ich ilość będzie tylko rosnąć. System wejścia-wejścia musi być zaprojektowana z myślą o rozmiar danych, którymi będzie operować.
* Pliki są modyfikowane poprzez dodanie nowych danych na końcu pliku, a nie poprzez przepisanie istniejących danych. Nagrywanie w dowolne miejsca, jak nie sekwencyjny odczyt danych z pliku hiper nie są skuteczne. Efektywne wykorzystanie podejścia "write once, read many".
* Projekt rozproszonego FS, nie musi polegać na projekt aplikacji, korzystających. W ten sposób system plików musi wiedzieć minimalna ilość informacji o klientach i charakteru operacji na danych, które ci klienci realizują.

**Cechy GFS**

GFS rozdziela strumienie danych i wątki. Wątki przechodzą od Master do klientów i od Master do Chunk-serwerów. Podczas gdy strumienie danych odbywają się od Chunkserver do klientów bezpośrednio, z pominięciem Master. Pozwala to zwiększyć zdolność do skalowalność całego GFS klastra w ogóle, tak jak Master nie jest "wąskim gardłem" w komunikacji z klientami.

Oprócz "standardowych" dla FS polecenia: create, delete, open, close, read, write – GFS obsługuje również transakcje snapshot i record append. Snapshot skutecznie tworzy kopię pliku lub katalogu, a record append pozwala dodawać dane na końcu pliku jednocześnie do wielu klientów z gwarancją niepodzielność zmian.

Jak we wszystkich systemach zbudowanych w architekturze Master-Slave, Master serwer GFS jest pojedynczego punktu awarii. Do tego, aby maksymalnie zniwelować skutek niedostępności Master na ogólną dostępność systemu w GFS funkcjonuje system logowania operacji i checkpoint'ów (po osiągnięciu logu określonych ilości). Ponadto, istnieje:

* system monitorowania dostępności Master;
* "Shadow Master, które pozwala zminimalizować czas "podnoszenia" nowego Master;
* rekord nazwy kanonicznej Master, w którym zwracają się do Master serwerem klienci, pozwala przy pomocy DNS alias w jak najkrótszym czasie przełączyć klientów na nowy Master.