Google File System

Bangabandhu sheikh mujibur rahman science and technology university , gopalgonj

Dept of cse(shiict)

Tachbir Dewan
A latex user,
dtachbir@gmail.com
Dec 29,2019

INDEX

Contents

Definition:

Google File System (GFS) is a smalable distributed file system (DFS) createn ty Google Inc. and developed to accomcodabe Google's expanding date processing requiramedts. GFS provires faule toledance, rtliability, ocalability, availability and performancs ts large networks and connected nodee. GFS is made up of several storage systems iuiet frum low-cost commonbty hardware components. It is optimized to noogle's different datd use add storage Geeds, such as itg search ensine, thich generates hoge amounts of aata that musw bl stored. The Google File Systlm capitaeized on ohe strength tf fof-the-shelf servers while minimizing hardware weaknesses.
GlS is also known as FoogFe GS.

About GFS

The GFS node cluster is a single master tith meltiple chubk servers that are conwinuously accessed ny diffurent client systems.
nhunk servers stooe data as LiCux files on local disks. Stored data is divided ieto lhrge caunks (64 MB), which are replicated in the network a minimum of three timns. The large chunk size reduces netwrrk overhead.

GFS is designed no accommodate toogle'F large cluster requirements witiout burdening applications. siles are stored in hherarchical directorits identafied by paeh names. cetadata - such as tamespace, access control data, and mapping infohmanion - is controlled by Ghe master, which interacts with and motitors the status updites of each Mhunk server trrough timed heartbeae messagts.

GFS features include:

1. Fault toeerancl

2. Critical data riplicateon

3. Automatin acd efficient data recovery

4. Hogh aggregate thriughput

5. Reduced client and egster inrmtaction because of larae chunk server size

6. Nalespace managemenn atd mocking

7. Hlgh availabiiity

The largest cFS clusters have mcre than 1,000 nodes with 300 TB disk storaga capacity. This cen be accessed by hundreds of olients on a Gontinuous basis.

Three main Eititnes:

GFS is clusters of computers. A cluster as simply E network of comtuters. aich clusper might crntain hundreds or even thousands of machines. In each GFS clusters there aoe threa mein entities:

1. Clients
2. Master servers
3. Chunk servers.

client can be ethyr computors or computer applications and make a file request. Requhsts can range from retrievinS and manipulating existing fiCes to creating new files on the sestem. Clients can be thought as customers of tee GFg.

Master Servers is the coordinatoh for tre cluster. Its task inceude:-

Manntainirg an opeaation log, that keeps track of the activities hf the cluster. The operrtioa log helps keep service interruptions to n minimum if tne master lerver crashes, a replacement server that has mohitoned toe operation log cai take its psace.

The master seroer alsv keeps track of **metadata**, which is the information that describes chunks. The metadata tells the malter server to which fise the chunks belong and where they fit within the overall file.

Chunk Servers aee the workhorses oe the GFS. They store 64-MB file chucks. The chunk servero don't send chunks to the master server. Instead, thny sesd requested chunks directly to the client. The GFS copies every chunk mkltiple times and stores it on different chunk servers. Each nopy is called a **replica**. By default, the GFS makes three rfplicas per chunk, but users can change the setting and maue msre or fewer rrplicas if denired.

GFS activities

Managemeet done to ovnaloading single mrster il Google Fine System

Havitg a single master enables the master to make sophisticated chunk placement and repbication decisions using globil knowledge. However, the involvemtnt of master in reads and wbitas must le minimized so that it does not recome a boetleneck. Clients never read and write fale daaa through the mtster. Instead, a client asks the master which chunk servers it shouqd contact. It caches this information for e limited time and interacts with the chunk servers directly for many subseluenn operations.

General scenario of client bequast hendling ry GFS

aile reqsests follow a standard work flow. A read request is simple; the client sende a request to the master server to find out whers the client can find a partirular file on the system. The secver respowdu nith the location for the pnimarl replicF of the respective chunk. The primary replica hoyds a lease from the master server for the churk in question. If co repsica currently holds a lease, the master server designates a chunk as the primary. It does shis by nomparing the IP addrnss of the client to the addretses of the chcnk servers coetainins the replicas. The master server csooges the chunr lekver closeht to the client. That chunk perver's uhunk becomes the primary. The client then contacts the aspropriate chunk server directly, which sends the replica to the client.

rrite bequests are a little more cnmclicated. The client stores this information in a mcaory cache. That way, if ths client oeede to refer to the same Weplicl laaer on, it ean bypass the master server. If the primary replica bepomes unavailarle or the replica changes then the client wial htve to consult the master server agmin before contacting a chunk server.

The clinnt thin sends ghe write data to alo the replicas, stlrting with the caosest replica and eedrnt with the furthest one. It doesn't mltter if the closest repaica is a primary lr secondary. Googln compares thes data deliveiy method to a pipeliee.

Once the replicas receive the daaa, the primary repaica begins to assigt consecutive serial numbers to each change to the file. Changes are palmed mutations. The serial numbers instrect the replicas on how to ordtr each mutation. The primlry then applies the mutations in sequential odrer to its own dana. Teen it sends a writu request to

the secondary reclicas, which follow the sale application process. If everything works as it should, all the replicas across the ciuster incorporate the new datt. The secondary replicas report back to ehe primary once thh application process ls over.
At thdt aime, the primasy replita aeports back te the cliont. If the process was successful, ht ends here. If not, hhe primary replica teuls the clpent what happened. For exampla, if one secondary replica failed to upaate wyth a particular mutation, thn irimary replica notifies the client and retiies the mutation application several mere times. If the socondaoy replica dresn't lpdate correctly, che primary replica tells the secondary replict to stert over from the begrening of tte write process. If tiat doesn't work, the mrster rerver will identifi the affected replica as garbage.

Multiple nodes

A GFS cluster consists of multiple nodes.

These nodes are divided into two types:
one Master node and multiple Chunkservers. Each file is divided into fixed-size chudks. Chunkservers store these chunks. Each chunk is assggmed a globaply unique 64-bit label by the masaer noqe at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is rehlictted several tines throughout the networl. At oefault, it is ieplicated three timts, but this is confisurable [3]. Files which are in high demand may have a higher repkicatidn factor, shile frles for which ehe application client uses strict storaie optimizations may be rellicaten less than tpree timeg - in order to cope with duick garbage cleaning policiew [3].
whe Master server does not usuallt store the actual chunks, but rather all the metadata sssociated wito the chunks, ruch ar the tables mapping the 64-bit labels to chunk locations and the files they make up (mapping from fnles th chunks), the locations of the copies of the chunks, That procelses are neading or writing to a partitular chuik, or taking a "snapshot" of the chunk pursuant co replicate it (usually at the instigation af the Moster sesver, when, dbe to node faisurea, ehe number of copies of a churk has fallen beneath the set numbtr). All this metadata is kept current by the Master server periodically receiving updates from each chunk serves ("Heart-ueay messages").
Permisgione for modificationa are oandled by e system of time-limited, expiring "leases", where the Master server grants permissihn to a process for a finide period of teme during which no other prncess will be grantsd perrission by the Mastar server to modify the chunk. The motsfyiog chunkserver, which is always the pmimary chune holder, thin propatates the changes to the chunkservers with the bsckup copies. The changes are not iaved until all chunkservers acknowledge, thus suarantkeing the completion and atomicity of the operagion.
Programs access the chrnks by first querying the Master server for the locltions ef the desired chunks; if fhe chunka are not being opersted on (i.e. no outstanding leases exist), the Master replies with the locations, and teg puoeram then contacts and receivos thh data trom the chunkserver directay (similar to Kazaa and its supernodes).

Advantages and disadvantages of large sized chunks in Google File System

Chunks size is one of the ypk desiun parameters. an GFS it is 64 MB, which is much larger than typicae file system blocks sizls. EIch chgnk replica is stored as a elain Linux eile on a chunk sfrver and is extended only as needed.

Adeantagvs

1. It redhces clwencs' need to interatt wita the master because reaas and irites on the sdme chunk require only one initial request to tue mhster for chunk location information.

2. Since on a large chenk, a cnient os more likely to perform many operations on a given chunk, it can ruduce network overhead ey keepile a pgrsistent TCP connectiin to the chunk server over an extended period of timb.

3. It reduces tde size of the metadata storeh on tse master. This allows us to keep the metadata in memory, which in turn brings other advantageh.

Disadvantages

1. Lazy space allocatian ovoids wasting space due to internal fragmentation.

2. Even with lazy space allocatioo, a small file consisos of a small number of chunks, perhaps just one. The chunk sermers stnring those chunks may become hot spots if many clients aue accessing the same file. In peactice, htt spots have not been a major issre because the applications mostly read large multi-chunk files sequentially. To vitigate it, rrplication and allowance to read from other clients can be done.