Distributed File System as a basis of Data-Intensive Computing

Assoc. Prof. Abzetdin Adamov
Chair, Computer Engineering Department,
Qafqaz University, Baku, Azerbaijan
aadamov@qu.edu.az

Abstract - The extremely fast grow of Internet Services, Web and Mobile Applications and advance of the related Pervasive, Ubiquity and Cloud Computing concepts have stumulated production of tremendous amounts of data available online. Event with the power of today's modern computers it still big challenge for business and government organizations to manage, search, analyze, and visualize this vast amount of data as information. Data-Intensive computing which is intended to address this problems become quite intense during the last few years yielding strong results.
Data intensive computing framework is a complex system which includes hardware, software, communications, and Distributed File System (DFS) architecture. This paper is giving comprehensive information on how distributed file system supports this approach of processing extra-large volumes of data. It is definitely expected that this work will contribute to future research on similar and related topics as spin off from this study.

Keywords: Data-Intensive Computing, Distributed File System, Fault-Tolerant System, GFS, HDFS

I. WHY BIG DATA BECAME SO BIG?

According to a recent IDC's Digital Universe Study the global volume of digital data increased by 62% between 2008 and 2009 to approximately 800,000 petabytes (PB). The report also noted that the rapidly expanding "Digital Universe" is expected to grow to 1.2 million PB, or 1.2 zettabytes (ZB) in 2010, 1.8 zettabyte in 2011, this year it is expected to grow up to 2.7 zettabytes (ZB) and reach 35 ZB by 2020. [1]
Data sets also grow in size because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), software logs, cameras, microphones, Radio-frequency identification readers, and wireless sensor networks.
Over 90% of this information will be unstructured, that is in the form which does not have predefined data model or information which is not within RDBMS. Examples of Unstructured Data may include books, journals, documents, audio, video, textual and binary files, the body of an e-mail message (email headers are structured information, since most of them have strong format and meaning), Web-page, etc. Generally, unstructured data is useless unless applying data mining or data extraction techniques in order to extract structured data. Two questions may arise naturally in connection with these forecasts: Why we produce this data if never consume it? What do we need in order to use this extra-large amount of data and benefit from it [2]?

II. DISTRIBUTED FILE SYSTEMS (DFS)

Because shared files are widely distributed across networks, there are growing problems of keeping users connected to the data they need. The distributed file system provides a mechanism of providing logical views of directories and files, regardless of where those files physically reside in the network. Fault tolerance of network storage resources is one of the most important characteristic of DFS [3].

A. Why is Distributed File System Useful

- Provide common view of centralized file system, but distributed over network
- Ability to open & update any file on any machine on network
- All of synchronization issues and capabilities of shared local files
- Data sharing of multiple users
- User mobility
- Location transparency
- Location independence
- Backups and centralized management

B. Motivation for DFS Implementation

- Need for a scalable DFS
- Large distributed data-intensive applications
- High data processing needs
- Performance, Reliability, Scalability and Availability
- Acute need for the systems with fault-tolerant capabilities

III. GOOGLE FILE SYSTEM

Google as one of the biggest Internet companies invented Google invented its own distributed file system named Google File System (GFS). Since the company is using this platform for all cloud services. According to GSF platform all application data is stored in huge files, so that data collecting by hundreds distributed machines may be stored in the same file.

Google's engineers designed GFS with clear intention to turn mass of cheap servers into distributed, reliable, fault-tolerant and scalable system which enables to store hudnreds of terabytes of data and provide remote access to this data by thousands of users and applications simultaneously at high speed. [4]

Like RAID 5 spreads data across multiple discs in order to avoid data losses, similarly GFS distributes same-size files (chunks) accross network. Chunks which are parts of the same file, thoretically may be replicated in cluster servers or even in geographically dispersed server nodes (see Fig. 1).

GFS is designed especially for enabling large data reads and transfers at extremely high speed [5].

A GFS cluster typically consists of a single master, multiple chunkservers and multiple clients as it is shown in figure 1.A file is divided into fixed-size chunks (64MB) and these chunks are distributed over the chunkservers. Each chunk is identified by a unique 64-bit chunk handle assigned by Master while creating the chunk. Each chunk is replicate on at least 3(can be changed) other chunkservers to increase the reliability of the system. [6]

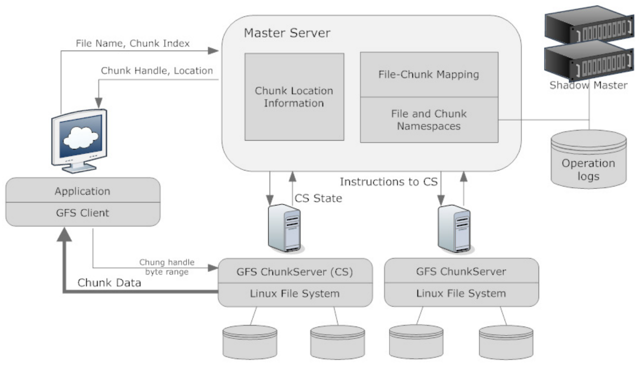

Fig. 1. Google File System Architecture.

IV. Hadoop Distributed File System (HDFS)

HDFS is distributed file system inspired by and similar to GFS. It is scalable to thousands of nodes which run on commodity hardware, so that HDFS assumes failures (both hardware and software) are common, targeted towards small number of very large files, write once and read multiple times. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

Hadoop files are composed of set of fixed-size blocks (typically 64MB). Each block is stored as a seperate file on the file system (see Fig. 2).

HDFS's placement policy meets the objectives of load balancing, fast access, fault tolerance. Each block is copying three times into different datanodes, and this process is also called "Block Level Replication"

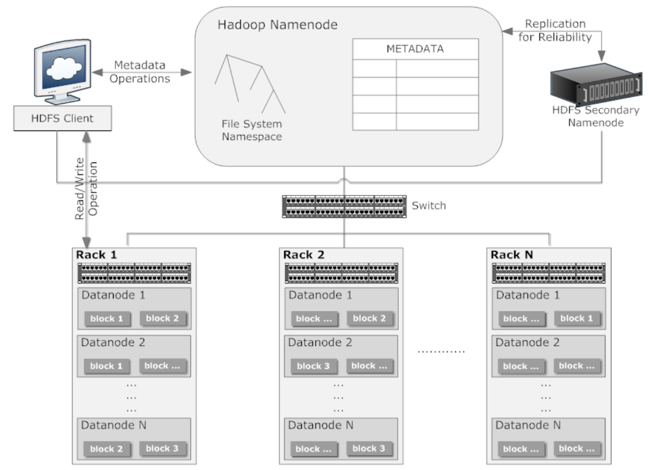

Fig. 2. Hadoop Distributed File System Architecture.

HDFS has two main types of Nodes: NameNode, and DataNodes. HDFS stores all filesystem metadata information on NameNode and actual data in DataNodes. NameNode is constantly checking the state of DataNodes. Everytime a client application wants to read or write data, it sends a message to the NameNode and the NameNode checks where the data should be read from or written to. After receiving appropriate location the client application reads/writes data directly to the DataNode.

A. Cluster Rebalancing

Another essential feature of the HDFS architecture is compatibility with data rebalancing schemes. A scheme might automatically move data from one DataNode to another if the free space on a DataNode falls below a certain threshold. In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster. [7]

B. What is not typical for HDFS?

- HDFS is not a POSIX file system
- HDFS is not designed for low latency access to a huge number of small files
- HDFS is not a platform for relational database and does not support transactions
- HDFS is not focused on security, encryption or multitenancy

Conclusion

Distributed File System is essential component of any Large-scale Data Processing or Data-Intensive

Computing System. Although, there are many implementations of DFS, but GFS and HDFS are the most successful platforms that became significant and vital component of digital infrastructure of biggest Internet players like Google, Yahoo, IBM, Amazon, etc. Even these DFSs provide relatively reliable, scalable and fault-tolerant platforms, there is still a need for challenge of new approaches and new ways of solving Data-Intensive Computing problems.

References

[1] Digital Universe Study, IDC analysts, June 2011, URL: http://www.emc.com/leadership/programs/digital-universe.htm

[2] Gokhale, M.; Cohen, J.; Yoo, A.; Miller, W.M.; Jacob, A.; Ulmer, C.; Pearce, R.; , "Hardware Technologies for High-Performance Data-Intensive Computing," Computer , vol.41, no.4, pp.60-68, April 2008, doi: 10.1109/MC.2008.125, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4488252&isnumber=4488232

[3] Step-by-Step Guide to Distributed File System, Microsoft, URL: http://technet.microsoft.com/en-us/library/bb727150.aspx

[4] Sakr, S.; Liu, A.; Batista, D.M.; Alomari, M.; , "A Survey of Large Scale Data Management Approaches in Cloud Environments," Communications Surveys & Tutorials, IEEE , vol.13, no.3, pp.311-336, Third Quarter 2011, doi: 10.1109/SURV.2011.032211.00087, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5742778&isnumber=6026692

[5] Sean Gallagher, The Great Disk Drive in the Sky: How Web giants store big—and we mean big—data, Jan 27 2012, URL: http://arstechnica.com/business/2012/01/the-big-disk-drive-in-the-sky-how-the-giants-of-the-web-store-big-data/

[6] Sanjay Ghemawat, Howard Gobioff, The Google File System, URL: http://research.google.com/archive/gfs.html, October, 2003

[7] Robert Chansler, Hairong Kuang, Sanjay Radia, The Hadoop Distributed File System, URL: http://www.aosabook.org/en/hdfs.html