Analytical Study on Performance, Challenges and Future Considerations of Google File System

Zahid Ullah, Sohail Jabbar, Muhammad Haris bin Tariq Alvi, and Awais Ahmad

***Abstract*—In this paper we present a brief analysis on the working and design of Google file System with its performance analysis. A brief Comparison of Google File system with other Distributed File Systems (DFS) namely HDFS, Global FS, GPFS and sector is also presented. Although Google shares many similarities with other DFSs but still it is unique in its design which is made specifically to serve Google's heavy workload. GFS's Architecture has a single master which is responsible for performing some major duties as explained in the Architecture session. Reliability of data is maintained through triple replication of data. So it also provides fast recovery and fault tolerance. Comparison and future consideration enables the reader to understand the current situation of GFS and its place in the future world. This paper not only explains GFS and its comparative study but also explains the complete background of the technology and its future.**

***Index Terms*—Distributed file system, HDFS, global FS.**

I. Introduction

File system is a data store used in computing for storing, retrieving as well as updating different set of files. For defining the files there will be either used the term abstract data structures, existent software or firmware to implement this abstraction. The actual contents of the files as well as the metadata both are accessed by the file system. The responsibility of the file system is to arrange and manage a reliable and efficient storage space that tuned with physical medium of storage. Among various available file systems for different purposes, distributed file system is one that is used on network and works in client server environment.

Distributed File System (DFS) is a file system that is used in the client server architecture where the files of any organization are organized in multiple distributed servers called "Server Message Blocks (SMBs)". These servers are used to share the files in DFS. The main features of DFS are redundancy of data and the location transparency in order to improve the availability of data. This is achieved by allowing the files to share on the multiple servers located on different places but they are grouped into one folder called the DFS root. DFD's are also known as the Network File Systems (NFS) since users can access their files to perform operations thereon like create, retrieve or alter as well as users can set the attributes of the files via operating system commands for those files and directories which are located on the remote systems.

There is a long list of distributed file systems available in the market. Some are open source, some are specially designed by some organization for fulfilling their specific needs and some are available for the use by the common users while some are provided to the end users for running their developed applications. Following are various types of Distributed File Systems (DFS) with respect to their offered features and dimensionality of their performance.

A. Distributed Fault-Tolerant File Systems

The replication of data that is distributed between multiple nodes (clients & servers) to get the fully availability of data as well as the offline operations. Some of the examples categorized under this type of file system are as follows:

- CODA from Carnegie Mellon University [1]
- Distributed File System (Dfs) from Microsoft
- InterMezzo from Cluster File Systems [2]
- Moose File System (MooseFS) from Gemius SA [3]
- Tahoe-LAFS is an open source secure, decentralized, fault-tolerant files System [4]

B. Distributed Parallel File Systems

This type of file system stripe data over several servers to gain high performance computing (HPC). Some of the examples categorized under this type of file system are as follows:

- Fraunhofer Parallel File System (FhGFS) from the Fraunhofer Society Competence Center [5]
- Parallel Virtual File System (PVFS, PVFS2, OrangeFS). This is available for Linux under GPL.
- STARFISH is a POSIX-compatible, N-way redundant file system created by Digital Bazaar Inc. [6].

C. Distributed Parallel Fault-Tolerant File Systems

Striping and replicating the data over multiple servers make the DFS parallel and fault-tolerant file system. Due to their high performance and data integrity, these types of file systems are used in both HPC and high-availability clusters. Among the long list of this type of file systems, some are given below:

- General Parallel File System (GPFS) by IBM [7]
- Google File System (GFS) by Google [8]
- Hadoop Distributed File System by Apache Software Foundation [9]
- Lustre by Cluster File Systems and currently supported by Intel [10]

Google is the pioneer in advance web searching and many advance web applications top of which are Google earth and

Manuscript received January 11, 2014; revised April 3, 2014.

Zahid Ullah is with the Department of Computer Science, Institute of Management Sciences, Peshawar (e-mail: zahid.ullah@imsciences.edu.pk).

Sohail Jabbar and Muhammad Haris bin Tariq Alvi are with the Department of Computer Science, Bahria University, Islamabad (e-mail: sjabbar.research@gmail.com, hbtalvi@hotmail.com).

Awais Ahmad is with the Department of Computer Science, CIIT, Islamabad (e-mail: awais.ahmad@comsats.edu.pk).

DOI: 10.7763/IJCCE.2014.V3.336

Google Maps. These days Google has not just kept itself limited to web search and applications only and is also providing mailing accounts of Gigabytes to users. Recently a social networking site named as Google+ is started by Google which is considered to be a strong competitor of Facebook in the coming days. All these services are heavily data intensive. Providing these services efficiently is a big challenge to their systems. So Google planned to design it in a way that it is not an exception but it is natural that something will fail every day. These exceptions are handled by GFS in their distributed file system. Google File System is a large scaled distributed system of files for large Google applications. Google organizes and manipulates files with Google GFS system. The application developers can use research and development resources with this service.

The GFS is not for sale purpose rather it is specific for Google itself. Still there are some details of Google GFS is unknown to outsiders. For example, Google doesn't disclose the number of computers used for GFS. Because the Google officials said that there are thousands of computers used for GFS system. Google keep many things secret but on the other side there are many things that Google showed to the public about the structure as well as the operations of GFS. In the beginning GFS is used for storing Google's search indexes as well as the creeping of data. But now it is used to store the content that is generated by the user. A new version of the Google File System is codenamed Colossus.

II. GFS Architecture

This section gives a brief overview of the Google File System architecture. Google established the GFS onto the bunch of computers. A single unit of cluster is simply a network of multiple computers. There are three types of entities in a cluster which are Clients, Chunk servers and the Master servers.

The Client is an entity which is used to make request in the GFS. The range of the request is about retrieving as well as manipulating the files exist in the system in order to create new files. The client entity is either being a computer outside the system or might be the application program in the current system. So in GFS system, a client acts like a customer.

The next entity is master server which acts like a coordinator for the relative cluster in GFS system. There are multiple duties perform by master server like keeping the operation log that conserves the track of all the actions of the Master's cluster. The operation log helps to minimize the service disruption. Hence, in case of failure of master server, another server which already supervises the operation log can take the place of that crashed server. Another duty of master server is to manage the metadata. It is the information that explains the chunks or groups of clusters in the GFS. The metadata is used to check the compatibility of data files with the groups and indicate that which file belongs to which chunk. In the startup, the chunk servers are polled into the cluster. The chunk servers are responsible to inform the master about their inventories. At this point, the master keeps the location of all the chunks within that cluster. The chunks are directly sending the chunks to the clients instead of the master servers. Multiple copies of the chunk are stored on multiple chunk servers. These copies are known as the "replica". There are three replicas made by the GFS by default but the user can change these settings.

A standard is followed in file request either read or write request. In read request, the client requests to the master server about the file that exists in the GFS system. In the reply of that request the server replied with the target of the replica of individual chunk.

III. Research Results

Some of the research results are drawn from the comprehensive analysis of literature available on internet and various resources are given in the subsequent paragraphs [11], [12].

A. Fault Tolerance

Google File System is a scalable distributed file system that is designed for the large scaled distributed applications. GFS provides fault tolerance as it running on the non-expensive hardware but always delivers the data to the clients with highest performance.

B. Faster Recovery

As we know that in GFS, both the master and the chunk servers have been restarted as well as to restore their states within a few seconds. Recovery has also been done on the basis of the priority of the condition. The recovery process is very fast as the replicas of master is distributed across multiple server machines on different places, if one side goes down then it will recovers by any other machine or place. So it is a better way of time consuming during recovery of data.

C. Chunk Replication

Google File System has replicas of the chunk servers on different machines in the system across multiple racks. So this will helpful in the recovery of the chunks very easily and precisely if any chunk goes down. There are multiple levels for the multiple parts of the file namespace. GFS used the checksum verification to keep complete replication of each chunk as detection of the corrupted chunks is very easy to cater. Chunk servers are the source to determine or verify the checksums before returning, and checksum will be incrementally updated in order to detect the errors while reading.

D. Master Mechanism

Master server in the GFS can do many things or operations in order to restore data without data lose as well as without the interruption of operations:

- Master saves all the changes which are made on the metadata
- It keeps the periodic checkpoints of the log file
- It keeps the replicas of all the logs and the checkpoints on multiple machines
- The state of the master is also replicated on different machines
- There is availability of the shadow master if the actual master goes down. The shadow master has only to access the file to read when the primary master goes

down. It enhances the availability of data read in GFS.

E. Performance

At a single time, there is only one active master server within the cluster. This might be a bottle neck that there is only one machine which is used to coordinate the thousands of computers within a cluster that may case of data traffic occurred. To save from this condition there are very small messages those are sent or received by the master. Remember that handling of the data files is fully performed by the master server.

The performance of GFS is almost increases as there uses the chunk servers which are directly link to the data files and is capable of direct reading of the files data. In GFS, the master server does not linked directly to the data files, but the primary server is attached to the whole chunk servers which are directly linked to the files. This mechanism helps to increase the performance of reading operation comparable to writing operation. From the results of benchmarks decision, if the number of servers used is relatively small, then the system of file get the performance as compared to that of a single disk, but it reduces the write performance. We summarized our discussion on the performance of GFDS that most of the workloads are reading 90%. The performance of GFS on large successive reads is cautionary. Because I suspect that if a child adds a video to its product set using GFS, which is cost per-byte is better than YouTube or even to any other service of video sharing.

IV. Comparison of GFS with Other DFSs

In this section, we will be discussing the DFS with Hadoop Distributed File System, General Parallel File System, Global File System and Sector Distributed File System. Table I also summarized the results of their comparative analysis.

TABLE I: Comparative Analysis of Google File System with other Well-Known Distributed File Systems

Design Decision	GFS	HDFS	Sector	GPFS	GlobalFS
Datasets Divided into Files or Blocks	Blocks	Blocks	Files	Blocks	Both
Protocol for Passing Message	TCP	TCP	Group Messaging Protocol	Not Mentioned	TCP
Protocol for Data Transferring	TCP	TCP	UDP	Not Mentioned	MPI for NAS
Replication Strategy	Replicas created at the time of Writing	Replicas created at the time of Writing	Replicas created periodically by the system	RAID-Replicated	No
Security Model	Not Mentioned	None	User Level and File level Access control	Not Mentioned	None

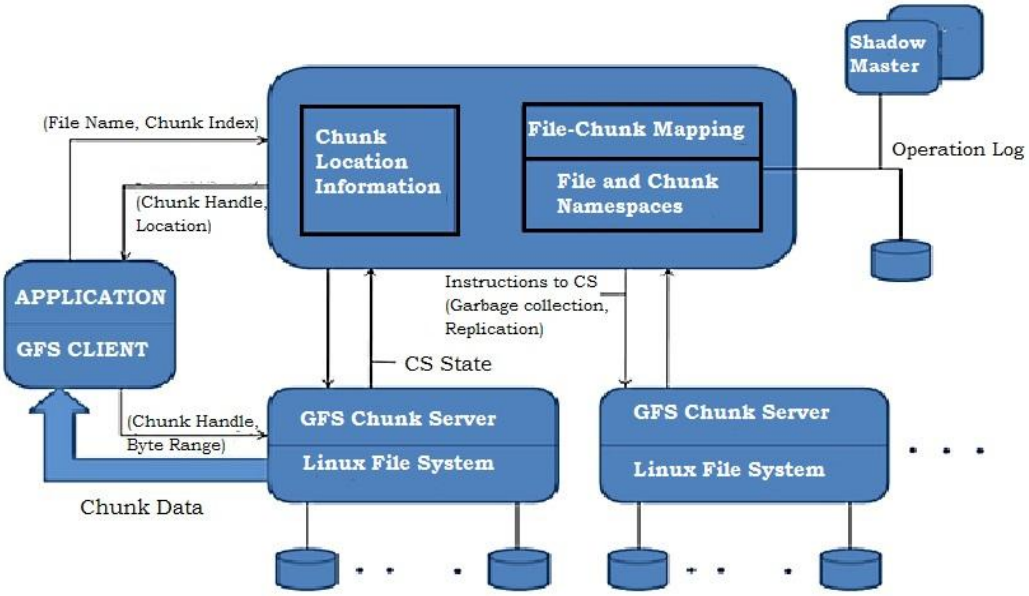

Fig. 1. GFS architecture.

A. Google File System

GFS is an application oriented file system optimized for Google's core data storage. A fixed chunk size of 64MB with a 64bit chunk handle is distributed to the chunk server. A single master node maintains metadata and mapping from file to chunk. To overcome the bottleneck of the master, the master's memory solely stores metadata. This increases the speed but the overall system size is limited by the master node's memory. A typical architecture of GFS is represented in Fig. 1.

GFS keeps the data in three replicas. If a server goes down, the master node redirects data requests to the other replica data. If master goes down another node can be selected to generate metadata by scanning over chunk server. With all these features GFS still have some drawbacks that are discussed in the later section below (Drawbacks/Loop Holes of GFS).

B. Hadoop Distributed File System

Yahoo! made an open source version of Google file system named as HDFS. Unlike GFS, HDFS don't provide the appending function. As both GFS and HDFS depends on a single master node which at times proves to be a failure point. So, different variants of HDFS were introduced namely RFS (Ring File System) and EDFS (Efficient Distributed File System).

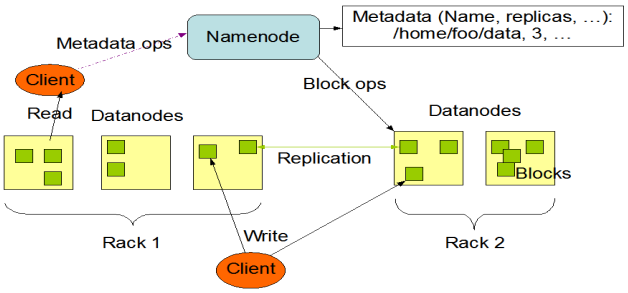

Fig. 2. HDFS architecture [13].

Just like GFS the dataset is divided into blocks with TCP used as a Protocol for both message passing and data transferring. As in GFS replicas are created at the time of creation. HDFS is almost similar to GFS apart from the appending function as mentioned earlier. Fig. 2 shows the architectural diagram of HDFS.

C. General Parallel File System

IBM created a shared disk parallel file system for super and cluster computer. Extreme scalability is achieved by DFS centralization and shared disk architecture enables GPFS. A total of 4096 disks with maximum 1TB size is supported with a scale of 4PB. A default block is of 256KB which is configurable from 16KB to 1MB. Sub blocks with a size of 1/32 of ordinary block are used for storing small files. Hashing is used for searching the large directory that may contain millions of files. Multi reading and writing is supported. An architectural diagram is also shown there in Fig. 3.

Just like the GFS, GPFS also creates a new meta node in case of failure of earlier meta node but unlike GFS this meta node never issues a new token till the log is recovered. GPFS supports POSIX fully but fault tolerance can't be compared to HDFS and GFS as data is not replicated by RAID.

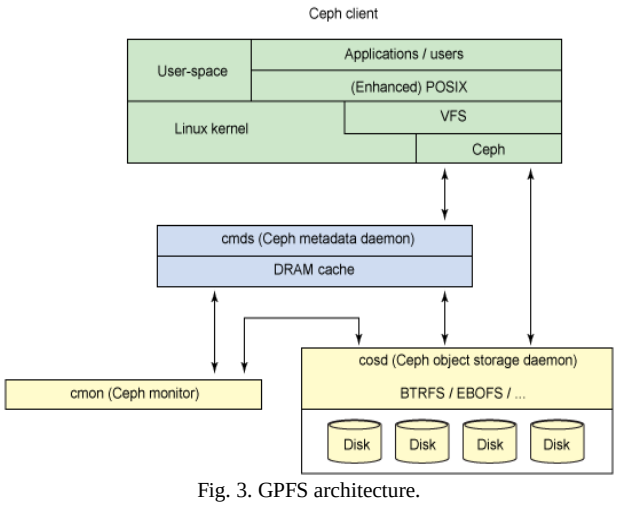

Fig. 3. GPFS architecture.

D. Global File System

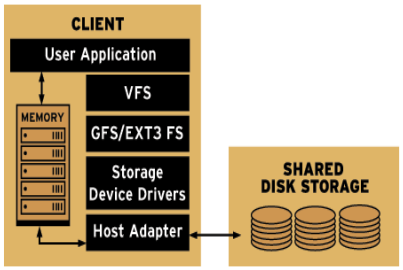

Fig. 4. GlobalFS architecture.

GlobalFS is shared disk file system for clusters of Linux computers. GlobalFS differs from the other DFS in a way that they allow direct access to the shared block storage. This can be used as a local file system. Its first native is with 64bit FS cluster on Linux. One of the main advantages is that applications don't have to be re coded for using GlobalFS [14]. It is a journaling FS with standard UNIX/POSIX file semantics. In case of a node failure consistency can be maintained by replaying the metadata operations. The complications of GlobalFS make it enable with all POSIX functionalities, whereas Google File System is aimed to make simpler state with lesser functions. An architectural diagram of GlobalFS is also shown there in Fig. 4.

E. Sector Distributed File System

Sector DFS is created on the basis of Sphere Compute Cloud that allows user a large downloadable dataset from almost anywhere. Unlike all other distributed FS those are mentioned above, it has security server with multiple salves. Divisions of files are in sector slices. The slave stores slices in its native file system. So, the sector is interoperate-able as and when required. A typical architectural diagram of Sector GFS is shown in Fig. 5.

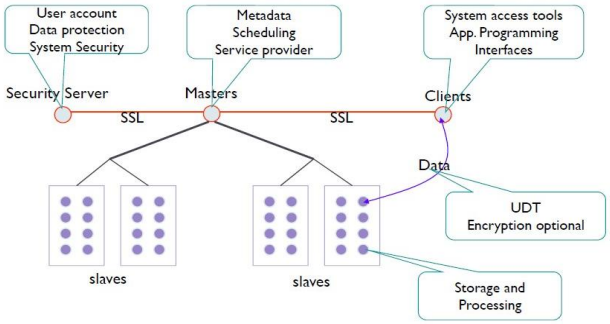

Fig. 5. Sector DFS architecture.

As mentioned earlier the most strong point of the sector over other Distributed file systems is its connectivity of security server with master. Each client request is passed through a security check to authenticate legality of the client. To ensure security salves has the capability to only listen to the master. If authentication is provided the salve opens a connection to get the data transfer started.

Just like GFS the data is triple replicated so any failure can be recovered easily. In case of a master failure, the reconstruction of metadata can be done by scanning salves with native FS. UDP is used instead of TCP that allows more speed. Because all these right now sector can be claimed to be more fast and reliable as compared to GFS.

V. Bottlenecks of GFS

In spite of the maximum beauties of Google File System in performance, implementation and its multi-facet functionalities, the fast running needs of time is constantly limiting it. In the subsequent paragraphs, we are presenting the bottlenecks in Google File System. The possible precautionary measures and expected solutions by Google team for the related issues are also given with it.

Since GFS is an application oriented file system. So, it might work well for one application but not for the other. GFS is single master node configuration. This configuration can result in a bottle neck though it only handles queries not the data.

Solution: Google Engineers are working on a distributed master system design the reason presented for single master system design is that in the original GFS it was done to make the design simple.

If a small file has only one chunk stored on one chunk server, it may become hot spot due to multiple accesses to the same file.

Solution: There are three possible solutions: 1) this can be reduced by introducing higher replication factor, 2) application start time is a major factor, staggering it can help reduce the factor, and 3) Making communication between client to client can also help.

Chunk size in GFS is 64MB that is much larger than the system block. A big issue for large block size is internal fragmentation (write a block which is less the block size to the disk will cause fragmentation inside block).

Solution: Lazy space allocation can prove to be a good solution for this which will first reserve the space for the block and to perform writes afterwards.

Some Other identified issues in GFS are as follows:

- GFS is not able to support links (neither hard nor soft)
- Bytes wise identically replicas are not guaranteed by GFS, still one copy is granted by it.
- Applications and clients face a risk of receiving staled chunk.
- If the application's write is large enough it can be a fragment added from another client.
- GFS don't have a standard API like POSIX

VI. Future Considerations and Challenges

Google has a strong team for research and development that constantly keep on looking beyond the current arisen horizon. The interest of packing the maximum facilities at the same platform is opening the new door of challenges especially with respect to scalability, effectiveness and efficiency as well. In the this section of future considerations and expected challenges, we are presenting the future plans on which Google is still working or any expected upcoming needs of time and its impact on the structure and architecture of Google File System.

A. Cross Language Informational Retrieval

Google Translator is aiming to translate all kind of languages in this world to all kind of languages. These translations will increase index size to a great deal; this application is really expensive in term of computation cost but once done it can turn out to show many benefits.

Challenges: 1) Maintaining the quality of translation is a big task, 2) Language models are really complex large scale systems are required to deal with it

B. ACLS in Information Retrieval System

Modern world has ever going communicating scenarios, a single user can have all type of data from confidential to public retrieval patterns can be different for all.

Challenges: 1) Building a system that has to deal with widely varying in size ACLs.

C. Automatic Construction of Efficient IR System

Although interfaces are common but the implementation method can vary greatly for efficiency. This can work very well but the efforts required to extend and maintain are really immense

Challenges: 1) Constructing a single system working on some particular parameters to automatic construct efficient retrieval system

D. Extraction of Information from Semi Structured Data

The total data in this world has a really small amount of data with semantic labels and the rest there is a large amount of semi structured data.

Challenges: 1) Making a algorithm for efficient extraction of structured information from unstructured/semi-structured data.

VII. Conclusion

Google File System is an application oriented distributed file system which is a result of Google's Engineering Genius. The single master node can be a cause of the bottle neck. Instead of the efforts Sector DFSs is still considered as the best DFSs but keeping in view the fact that GFS is solely made to server Google's workload so its commercial comparison is not possible. Large scale data processing quality essentials are demonstrated by GFS. Some of the design parameters are specifically for Google's own setting but tasks of data processing of similar magnitude can be performed with it. Google has made it a standard that failure is a rule not an exception, their assumption has made them stand out of crowd. Overall system is improvement is achieved by optimization of huge file that are usually the appended form of files then read operation is performed both extending and relaxing the standard file system. Fault tolerance is achieved by constant monitoring and replicating crucial data which is supported by fast and automatic recovery. Chunk server failure is tolerated by using chunk replication. High through output is achieved by minimizing the involvement of master in client server communication by separating system control from data transfer. This also minimizes the chance of master bottle neck and Google believes that their improvement will lift current limitation to write which the clients are facing right now. GFS is the backbone that has allowed Google to meet all its storage needs and has enabled them to attack problems on the scale of the entire web and continue its innovation.

REFERENCES

[1] P. J. Braam, "The coda distributed file system," *Linux Journal*, vol. 1998, pp. 46-50, June 1998.

[2] P. J. Braam, M. Callahan, and P. Schwan. The InterMezzo File System. [Online]. Available: http://www.cs.cmu.edu/~coda/docdir/intermezzo99.pdf

[3] S. Bai and K. Wu, "The performance study on several distributed file systems," in *Proc. 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (CyberC), Oct. 2011, pp. 226-229.

[4] Z. Wilcox-O'Hearn and Brian Warner, "Tahoe: the least-authority filesystem," in *Proc. the 4th ACM international workshop on Storage security and survivability*, 2008, pp. 21-26.

[5] Fraunhofer. (2010). Competence center for high performance computing. ITWM. [Online]. Available: http://www.itwm.fraunhofer.de/abteilungen/hpc.html

[6] G. Garzoglio, K. Chadwick, and T. Hesselroth, "Investigation of storage options for scientific computing on grid and cloud facilities," presented at the International Symposium on Grids and Clouds and the Open Grid Forum Academia Sinica, Taipei, Taiwan, March 19-25, 2011.

[7] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Bilgen Cetin, and S. Babu, "Starfish: A self-tuning system for big data analytics," presented at 5th Biennial Conference on Innovative Data Systems Research (CIDR '11), Asilomar, California, USA, January 9-12, 2011.

[8] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. the Conference on File and Storage Technologies (FAST'02)*, 2002, pp. 231–244.

[9] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," in *Proc. the Nineteenth ACM Symposium on Operating Systems Principles* SOSP '03, December 2003, pp. 29-43.

[10] K. Shvachko, K. Hairong, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies* (MSST), May 2010, pp. 1–10.

[11] P. Schwan, "Lustre: Building a file system for 1,000-node clusters," presented at the Linux Symposium, Ottawa, Ontario, Canada, July 23th–26th, 2003.

[12] A. Fikes. (2010). Storage architecture and challenges of Google file system. presented at Google Faculty Summit. [Online]. Available: http://static.googleusercontent.com/media/research.google.com/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf

[13] J. Passing. (April, 2008).The Google File System and its application in MapReduce. [Online]. Available: http://int3.de/res/GfsMapReduce/GfsAndMapReduce.pdf

[14] N. UzZaman, "Survey on Google file system," Survey Paper for CSC 456 (Operating Systems), University of Rochester, Fall 2007.

Zahid Ullah is an HEC scholar and currently pursing PhD and working as a lecturer at IMSciences Peshawar, Pakistan. He received his BCS (H) degree in computer science from KPK Agricultural University, Peshawar, Pakistan in 2005 and MS degree in computer and telecom from Gandhara University Peshawar, Pakistan in 2008. He has more than 8 years of experience in computer networks and worked on Cisco and Huawei routers. His research interest includes mobile ad-hoc networks and wireless sensor networks.

Sohail Jabbar is an HEC scholar and is pursuing his PhD in computer science from Bahria University, Islamabad. He did MS (telecom and networking) from the same University with the honor of Magna Cum Laude and secured his BS (computer science) degree from Allama Iqbal Open University (AIOU), Islamabad, Pakistan in 2009 and 2005 respectively. His research work is emerged with more than 24 international publications in various IEEE, ACM and IAENG (International Association of Engineers) conferences held in USA, UK, South Korea, China and Pakistan and journals. He has been a reviewer of international journal of distributed sensor networks (IJDSN), journal of engineering applications of artificial intelligence (EAAI), Ad Hoc & Sensor Wireless Networks Journal and World Applied Science Journal (WASJ). His current fields of interest are wireless sensor network, cloud computing and integration of cloud computing and sensor networks.

Haris Bin Tariq Alvi completed his matriculation with science and intermediate from Federal board of intermediate and secondary education, Pakistan with 1st division and attended Hamza Army Public School and College Rawalpindi, Pakistan. He is currently in the final year of his bachelor of electrical engineering from Bahria University, Islamabad, Pakistan, and working with Microsoft as an Intern.

Being an engineering student he has interest to cover all the possible aspects of it. He has worked from computer science to mechanical aspect of the things and likes to learn new things.

Awais Ahmad did his BS(CS) and MS from Department of Computer Science University of Peshawar, Pakistan in 2008 and Bahria University Islamabad Pakistan in telecommunication and networking respectively. He is a member of "Cloud Computing" research group in University of BridgePort USA. After completing his master, he joined Department of Computer Science, CIIT Islamabad as a Lecturer (on study leave). He also worked on Underwater Acoustic Sensor Network (mainly: Physical, Transport and Routing Layer). With parallel to this work, he also worked on IEEE802.11n MAC for terrestrial wireless sensor network and underwater acoustic sensor network. His current research work is on machine-to-machine communication (handover techniques, IEEE 802.11n MAC). He served as a reviewer for many Journals including Journal of Super Computing (Springer), Mathematical Problems in Engineering, Journal of Applied Mathematics, International Journal Distributed Sensor Network (Hindawi Publishing Corporation) as well as several IEEE & ACM International Conferences. He also received three prestigious awards: i) Beast research award 2011 from Honorable Rector of Bahria University, ii) Best paper nomination award in WCECS 2011 University of California, San Francisco USA, and iii) Best research paper award, 1st Symposium on Computer Science and Engineering, Moju Resort, Korea 2013.