"A Survey On Different Large Scale Reliable Distributed Storage Systems"

Parameshwar Reddy. V
M.Tech Student, VTU
Computer Networking, EWIT
Bangalore.

Roopanjali. Daddi
Assistant Professor,
Department Of ISE, EWIT
Bangalore.

Abstract

Large scale reliable storage systems are playing a major role in the today's world of managing data over disturbed systems. As the data volume increases in more and more applications fields of science, engineering, information sciences etc., the challenges posed by large scale storage systems gain an increasing importance. As data in PB(petabytes) doubling for every 2 years as the storage systems must be able to accommodate from PBs to EBs. In this paper we provide the survey on different large scale storage systems available in the market maintained by different organization. These distributed storage system includes Data-intensive oriented file systems, parallel file systems, cloud data storage services. Initially, we discussed new challenges and issues for the large-scale data storage along with desired principles. Second it was discussed different storage systems along with architecture of each distributed storage systems. Finally, provided compared these different storage systems with issues along with the future enhancement.

Key terms– challenges, data-intensive file systems, parallel file systems, cloud data storage.

1.Introduction

In the today's world large scale reliable storage systems playing a major role over the internet in protecting and retrieving the data. Distributed systems designed specifically to handle very large-scale data stream processing applications [1] are in their infancy. Distributed stream processing are becoming more common, that are more commercially available and more in development.

At a new approach to Big Data storage must also fulfil the following requirements:

- Ability to scale to billions of objects while maintaining performance for all users without disruption. Big Data storage means that storage systems must be able to accommodate PBs to EBs(Exabyte's) of capacity, and billions of objects or files, with satisfactory performance for millions of concurrent users.
- Support for ongoing non disruptive tech refreshes Storage systems must be able to handle equipment updates online, without downtime or manually intensive data migrations.
- Always available and online five nines (99.999%) availability all the time.
- Competitive TCO(total cost) Storage has become the largest line item for IT budgets, while an increasing climate of austerity is placing increasingly stringent demands to significantly reduce storage costs.

Executing data-intensive applications [2] at a very large scale raises the need to efficiently achieve several challenges:

Scalable architecture: The data storage and management infrastructure needs to be able to efficiently leverage a large number of storage resources which are amassed and continuously added in huge data centres or petascale supercomputers. Orders of tens of thousands of nodes are common.

Massive unstructured data: Most of the data in circulation is unstructured: pictures, sound, movies, documents, experimental measurements. All these mix to build the input of applications and grow to huge size, systems gain increasing importance as data in PB doubling for every 2 years. Such data is typically stored as huge objects and continuously updated by running applications.

Many data objects: A dataobject [3] is a logical cluster of all tables that represent an object view of related tables. Unstructured data can be made easily accessible through the data objects. Each object acts as a container for end users' files, and indexes are often held in RAM, making them very fast to access.

Transparency: Large scale distributed infrastructures heavily rely on explicit data localization and on explicit transfers. Managing huge amounts of data in an explicit way at a very large

IJERT IJERT

scale makes the usage of the data management system complex and tedious. One issue to be addressed in the first place is therefore the transparency with respect to data localization and data movements. The data storage system should automatically handle these aspects and thus substantially simplify user's access to data.

High throughput under heavy access concurrency: Several methods to process huge amounts of data have been established. Traditionally, message passing has been the choice of designing parallel and distributed applications. While this approach enables optimal exploitation of parallelism in the application, it requires the user to explicitly distribute work, perform data transfers and manage synchronization. Recent proposals such as MapReduce [4] and Dryad [5] try to address this by forcing the user to adhere to a special paradigm. While this is not always possible, it has the advantage that once the application is cast in the framework, the details of scheduling, distribution and data transfers can be handled automatically. Regardless of the approach, in the context of data-intensive applications [2] this translates to massively parallel data access that has to be handled efficiently by the underlying storage service. Since data-intensive applications spend considerable time to perform I/O, a high throughput in spite of heavy access concurrency is an important property that impacts on the total application execution time.

Support for highly parallel data workflows: Many data-intensive applications consist of multiple phases where data acquisition interleaves with data processing, resulting highly parallel data workflows [6]. Synchronizing access to the data under these circumstances is a difficult problem. Scalable ways of addressing this issue at the level of the storage service are thus highly desirable.

Large scale storage systems in the present market should achieve the following issues Centralized meta data [7], Versioning support, fine grain writes [8], Too many small files. In further we discus about different large scale distributed storage systems includes Data-intensive oriented file systems[9], parallel file systems[10] and cloud data storage services[11]. Data-intensive oriented file systems which has the General file system(GFS) which was maintained by the Google organization, and Hadoop distribute file system (HDFS) maintained by the yahoo. The parallel file system has General parallel file system(GPFS) and Lustre. Cloud data storage services includes simple storage service(S3) and Azure. Finally we will provide the comparison of these distributed storage systems along with new issues mentioned and future storage system which should come across these issues.

2. Related work

These large scale reliable distributed systems can be divided into three categories. They are

1. Data intensive oriented file systems
2. Parallel file systems
3. Cloud data storage services

In the below it was discussed with each file systems along with the examples.

2.1. Data intensive oriented file systems

Data-intensive distributed file systems [9] are emerging as a key component of large scale Internet services. They are designed from the ground up and are tuned for specific application workloads. In data-intensive oriented file systems the huge data can be divided into the small blocks to make efficient use of memory available. It has structured storage which can be built on top which allows fine grain concurrent reads [12] but not fine grain concurrent writes [12]. In this system locking is not mandatory and it has data location aware. Leading examples, such as the Google File System (GFS) [13], Hadoop distributed file system (HDFS).

Google file system (GFS): GFS which is maintained by the Google organization. GFS is optimized for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that needs to be retained. Files are divided into fixed-size chunks of 64megabytes.

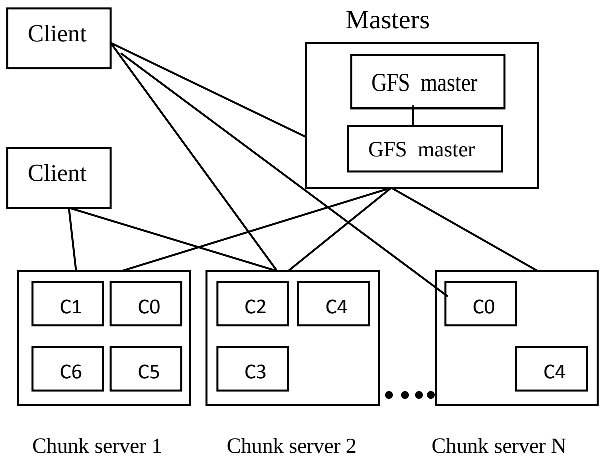


Figure 1: GFS architecture

A GFS cluster consists of multiple nodes. These nodes are divided into two types: one Master node and a large number of Chunkservers. Each file is divided into fixed-size chunks. Chunkservers store these chunks. Each chunk is assigned a unique 64-bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network, with the minimum being three, but even more for files that have high end-in demand or need more redundancy. GFS master manages metadata and data transfers between the client and chunkservers shown in the figure 1.

Each chunk can be saved minimum three times. Deciding from benchmarking results, when used with relatively small number of server, the file system achieves reading performance comparable to that of a single disk (80–100 MB/s), but has a reduced write performance (30 MB/s), and is relatively slow (5 MB/s)[13] in appending data to existing files.

Hadoop distributed file system (HDFS): HDFS [15] is the part of yahoo is an open source which was implemented using the MapReduce [4] and it is java based. In this distributed storage system the file is divided into large blocks (64MB). These blocks are distributed across the clusters which are replicated several times to come across hardware failure. Data Placement is exposed so that computation can migrated to data. HDFS has master/slave architecture, where HDFS namenode manages all file system metadata in memory, it will list the files for each file name a set of blocks has been fixed and for each block , a set of Datanodes.

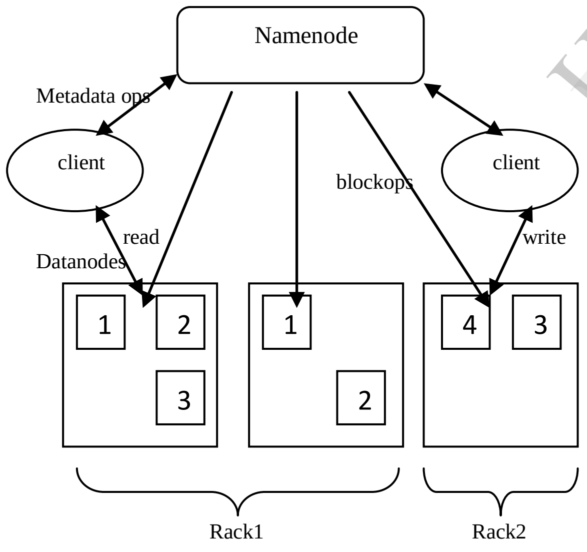


Figure 2: HDFS architecture

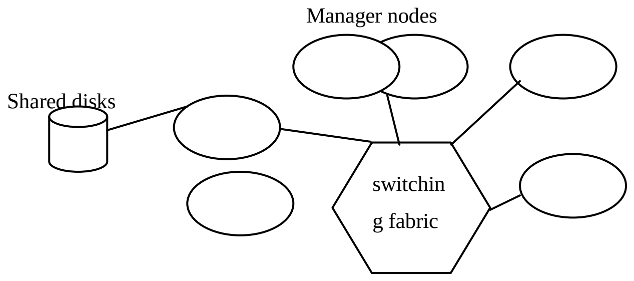
Name node controls the read/write access to files and it also manages the block replication. Datanodes is block server which stores the data in local file system, stores the metadata of a block, and serves the data and meta data to the clients. Block report will periodically sends a report of all existing blocks to the namenode. Racks are used to store the data blocks as shown in the figure 2 two are more Datanodes put together for a Rack. Here the replication can be performed upon the instruction by Namenode. Datanodes send the heartbeat signals to the namenode for every period of time and namenode uses these heart beats to detect Datanodes failures. Data correctness can be validated by using checksums (CRC32) [16].

2.2. Parallel file systems

Parallel file systems the data has been distributed across the disks and data striping can be done here. It has the advanced caching. In parallel file systems [10], a few nodes connected to the storage—known as I/O nodes—serve data to the rest of the cluster. The main advantages a parallel file system can provide include a global name space, scalability, and the capability to distribute large files across multiple nodes. In a cluster environment [17], large files are shared across multiple nodes, making a parallel file system well suited for I/O subsystems. Generally, a parallel file system includes a metadata server (MDS), which contains information about the data on the I/O nodes.

Metadata is the information about a file—for example, its name, location, and owner. Some parallel file systems use a dedicated server for the MDS, while other parallel file systems distribute the functionality of the MDS across the I/O nodes. Leading examples are General parallel file systems (GPFS) and Lustre.

General parallel file systems (GPFS): GPFS [10] was developed by the IBM which has high performance shared disk file systems used by many super computers in top500. It uses the technique of distributed storage system where the file divided into the blocks(less than1MB) and these blocks are distributed across the cluster. These blocks are RAID [18]- replicated or file system node replicated. It has very efficient indexing of directory entries for very large directories. In the figure 3 it was shown the GPFS architecture which has manager nodes, storage nodes attached to shared disks, file system nodes and switching fabric. File system nodes run user programs, read/write data to/from storage nodes and these nodes also cooperate with manager nodes to perform metadata operations.



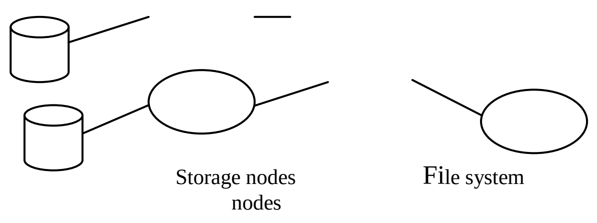
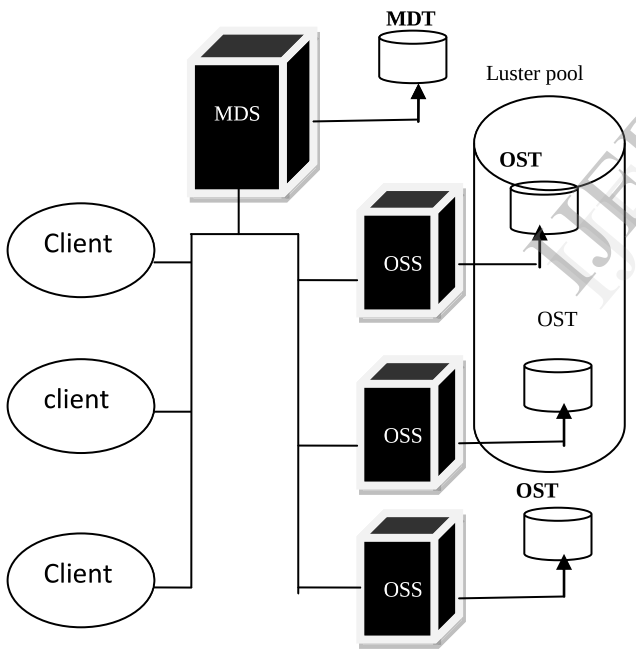


Figure 3: GPFS architecture

Storage nodes implements block I/O interface and they interact with manager nodes for recovery. Data and metadata stripped across multiple disks-multiple storage nodes. Manager nodes which is responsible for file system configuration like recovery and adding disks. It acts as a disk space allocation manager, quota manager, file metadata manager (maintain file data integrity), and global lock manager.

Lustre: Lustre [19] is the massively parallel distributed file system owned by the Oracle, which is used by most super computers (the world's fastest computer-Tianhe-1A [20], jaguar [21]).



MDS- Meta data server **MDT- Meta data targets**
OSS- Object storage server
OST- Object storage targets

Figure 4: Lustre Architecture

Lustre file systems [22] are scalable and can support tens of thousands of client systems, tens of petabytes (PB) of storage, and more than a terabyte per second (TB/s) of aggregate I/O throughput. It is open source which is an object based. Meta data target is a disk which is in metadata server for storing of data. Object storage server store data on object storage target which is an distributed locking. Clients which are attached to the DSS should posses the POSIX[23] semantics. Luster pool which combine two or more OST. The figure 4 shows the Lustre architecture[19].

2.3 Cloud data storage services

Cloud storage system can be considered to be a network of distributed data centers which typically uses cloud computing technologies like Virtualization, and offers some kind of interface for storing data. To increase the availability of the data, it may be redundantly stored at different locations. In general, all of this is not visible to the user. Many cloud storage providers are active on the market, offer various kinds offering services to their customers. Basic cloud storage services [11] are generally not designed to be accessed directly by users but rather incorporated into custom software using "application programming interfaces" (API) [24]. Examples of such basic cloud storage services are Amazon S3 and Microsoft Azure.

Simple storage service (S3): Amazon S3 [25] is an online storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces [25] (Examples REST, SOAP, and BitTorrent). S3 storage for internet which has pay per use policy(for storage, request, data transfers)Amazon claims that S3 uses the same scalable storage infrastructure that Amazon.com uses to run its own global e-commerce network. S3 stores arbitrary objects (computer files) up to 5 terabytes in size, each accompanied by up to 2 kilobytes of metadata.

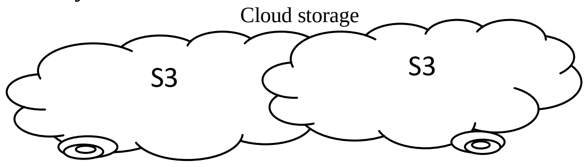


Figure 5: Cloud storage

Objects are organized into buckets (each owned by an Amazon Web Services or AWS account), and identified within each bucket by a unique, user-assigned key. Amazon Machine Images (AMIs) which are modified in the Elastic Compute Cloud (EC2) can be exported to S3 as bundles Buckets and objects can be created, listed and retrieved using either a REST style HTTP interface or a SOAP interface. Additionally, objects can be downloaded using the HTTP GET interface Requests are authorized using an access control list associated with each bucket and object.

IJERT

Azure: Azure [27] is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. It provides both platform as a service (PaaS) [28] and infrastructure as a service(IaaS) [28] services and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems. Here data manipulation can be done based on HTTP.

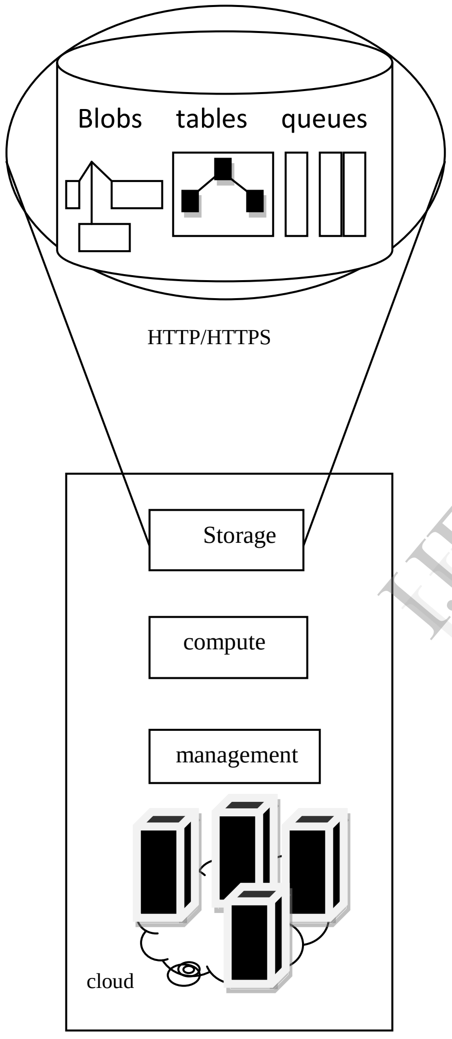


Figure 6: Windows Azure

The data is replicated 3 times to make it fault tolerant, availability, and for scalable. Here Azure has three components storage, compute and management. We will consider here only storage part as the storage has Blobs, tables and queues. Blobs have up to 1TB of unstructured data which are grouped in a container. Tables are the group of entities /records that contain properties which has fine grained access to structured data. Queues are the asynchronous communication between the cloud instances. Azure storage can be independently accessible that is it can be used from any platform, on-premise or cloud based. It is independently scalable as it does not depend on windows Azure compute.

3.Comparison

The existing large scale reliable distributed storage systems in the today's world are unable to come across some issues which are provided in the introduction. Lets us compare these storage systems with our issues.

Table1: comparison between the different storage devices.

Issue	Data-intensive FS	Parallel FS	Cloud storage
Centralized metadata	No	Yes	Yes
No version support	No	No	Yes
No fine grain writes	No	Yes	No
Too many small files	Yes	No	No

Yes – addressed issues

Comparison between these existing different large storage systems shows that all the issues can't be addressed by any storage system.

4.Conclusion

This paper provides the requirements that are required for the large scale storage systems along with the challenges. We have arise some issues that are required for the today's world. We divide the available large scale storage systems into three categories according to their storage and accessing of data with examples along with architecture. At last we compare these storage systems with our issues that has been arises initially. At last through comparison between these existing different large storage systems and we had concluded that issues which we had given can't be addressed by any storage system.

IJERT

5. Future enhancements

In Future the new issues and requirements may arise as day by day the technology is growing, upon those satisfying those issues a new large scale reliable storage systems may be developed.

REFERENCES

[1] "A large scale data streaming system" Vincenzo Gulisano, Ricardo Jimenez-Peris, Marta Patino-Martinez, Patrick Valduriez *2010 International Conference on Distributed Computing Systems.*

[2] "Real-life Data Intensive Applications" Jacek Becla *2010 Salishan Conference on High Speed Computing*.

[3]*http://en.wikipedia.org/wiki/Object_storage_device.*

[4] "MapReduce: Simplified Data Processing on Large Clusters" Jeffrey Dean and Sanjay Ghemawat.

[5] "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", *Michael Isard, Mihai Budiu, Yuan Yu.*

[6] "Conception and design of parallel and distributed applications" Valentin CRISTEA, *University Politehnica of Bucharest.*

[7] "Metadata – centralized and distributed in dw2.0" by *whinmon.*

[8]"Fine-grain concurrency", Tony Hoare, *Communicating Process Architectures 2007*.

[9]"Data-intensive File Systems for Internet Services:" Wittawat Tantisiriroj, Swapnil Patil, Garth Gibson, *Parallel Data Laboratory 2008*.

[10]**"**Enhancing High-Performance Computing Clusters with Parallel File Systems" by amina saify; garima kochhar; jenwei hsieh, 2005 dell power solutions.

[11] "Cloud storage for cloud computing" open grid forum, SNIA.

[12] "Views:Synthesizing Fine-Grained Concurrency Control" *brian demsky, patrick lam, ACM journal.*

[13] "Google:Designs, Lessons and Advice from Building Large Distributed Systems", Jeff Dean.

[14] "On the security of cloud storage services" fraunhofer institute for secure information technology , 2010.

[15] "The Hadoop Distributed File System: Architecture and Design", by Dhruba Borthakur.

[16] "CYCLIC REDUNDANCY CHECK" from Wikipedia free encyclopedia.

[17]http://www.scfbio-iitd.res.in/doc/clustering.pdf

[18] "RAID Theory: An Overview the Cuddletech Veritas Volume Manager Series" Ben Rockwood, Cuddletech.

[19] "The Lustre Storage Architecture" Peter J. Braam, Cluster File Systems 2004.

[20] "The TianHe-1A Supercomputer: Its Hardware and Software" Xue-Jun Yang, Xiang-Ke Liao, Jun-Qiang Song, *journal of computer science and technology 26(3): 344–351 May 2011. DOI 10.1007/s11390-011-1137-4.*

[21] "Jaguar: The World's Most Powerful Computer" Author S. Bland, Ricky A. Kendall, Douglas B. Kothe, James H. Rogers, Galen M. Shipman, *Oak Ridge National Laboratory*.

[22] "LUSTR FILE SYSTEM High-Performance Storage Architecture and Scalable Cluster File System" White Paper December 2007.

[23] "pNFS, POSIX, and MPI-IO: A Tale of Three Semantics", Dean Hildebrand, Arifa Nisar, Roger Haskin.

[24] "Application Program Interface (API) A path to seamless software integration " By Sumanth Bail.

[25] http://aws.amazon.com/s3/.

[26] http://en.wikipedia.org/wiki/Web_service

[27]**"**Windows Azure platform", Dominick Baier, Christian Weyer.

[28]"http://www.apellestech.com/content/pdf/servic"

IJERT