ISSN No. 0976-5697

Volume 5, No. 7, September-October 2014

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

# A Review on Distributed File System and Its Applications

ShikhaVashist, Ayush Gupta
Student, Dronacharya College of Engineering,
Gurgaon, Hr., India-123506

***Abstract:*** Need of storing a huge amount of data has grown over the past years. Whether data are of multimedia types (e.g. images, audio, or video) or are produced by scientific computation, they should be stored for future reuse. Users also need their data quickly. [2,5]Data files can be stored on a local file system or on a distributed file system. Local file system provides the data quickly but does not have enough capacity for storing a huge amount of the data. On the other side, a distributed file system provides many advantages such as [7]scalability, security, capacity, etc. In the report, we will provide the state of the art in DFS oriented on reliability and performance in thesetype of systems. First of all, traditional DFS like AFS, NFS and SMB will be explored. These DFS were chosen because of their regular usage. Next, new trends in these systems with a focus on reliability and increasing performance will be discussed. This includes the organization of data and metadata storage, usage of caching, and design of replication algorithms and algorithms for achieving more reliability.

## I. INTRODUCTION

File system is a subsystem of an operating system whose purpose is to organize, retrieve, store and allow sharing of data files. A Distributed File System is a distributed implementation of the classical time-sharing model of a file system, where multiple users who are geographically dispersed share files and storage resources. Accordingly, the file service activity in a distributed system has to be carried out across the whole network, and instead of a single centralized data repository there are multiple and independent storage devices. The [1]DFS can also be defined in terms of the abstract notation of a file. Permanent storage is a fundamental abstraction in computing. It consist of a named set of objects that come into existence by explicit creation, and are immune to temporary failures of the system. A file system is the refinement of such an abstraction. A DFS is a file system that supports the sharing of files in the form of persistent storage over a set of network connected nodes. The DFS has to satisfy three important requirements: [3]Transparency, Fault Tolerance and Scalability.

## II. DISTRIBUTED FILE SYSTEM AND METHOD

A distributed file system and method distributes file system objects across multiple self-contained volumes, where each of the volume is owned by a unique file system node. Logical links are to be used to reference a file system object between volumes. Each system node includes a move directory in which is maintained hard links to locally-stored file systemobjects that are referenced from another[4]system node using logical links.Various file system operations that involve multiple volumes are performed without having to place a write lock on more than one volume at a time.Various caching schemes allow the various file system nodes to cache file system object data and metadata.

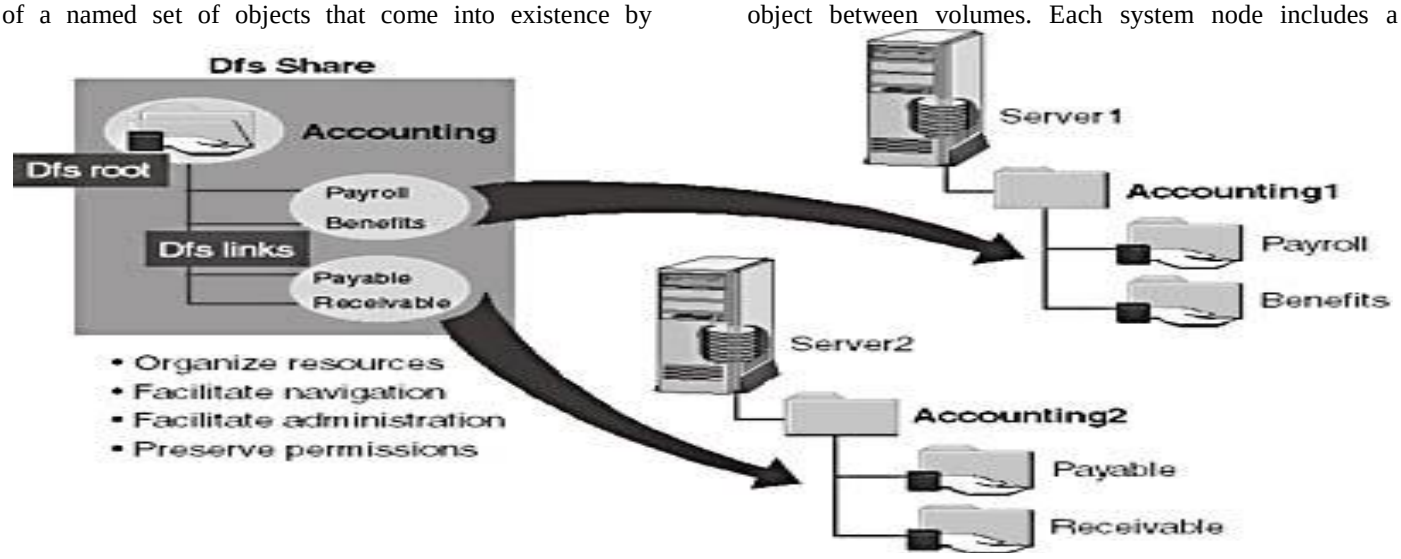


Figure 1.

## III. ON-DISK FILE FORMAT FOR A SERVERLESS DISTRIBUTED FILE SYSTEM

A file format for a serverless distributed file system is composed of two parts: a primary data stream and a metadata stream. The data stream contains a file that is divided into multiple blocks. The metadata stream has a header, a structure for indexing the encrypted blocks in the

primary data stream, and some user information. The indexing structure defines leaf nodes for every block. Each leaf node consists of an access value used for decryption of the associated block and a verification value used to verify the encrypted block independently of the other blocks. In one implementation, the access is formed by hashing the file block and encrypting the resultant hash value using a randomly generated key. The key is then encrypted using the user's key as the encryption key. The verification value is formed by considering the associated encrypted block using a one-way hash function. The file format supports verification of individual file blocks without knowledge of the randomly generated key or any user keys. To verify a block of the file, the file system traverses the tree to the appropriate leaf node associated with a target block to be verified. The file system hashes the target block and if the hash matches the access value contained in the leaf node, the block is authentic.

## IV. NOTIFICATION SYSTEM FOR DISTRIBUTED FILE SYSTEM

A method for notifying an application coupled to a distributed file system is described. A command for a file for a distributed file system is received. The distributed file system stores portions of files across a plurality of distinct physical storage locations. The command for the file is compared with a notification table of the distributed file system of the distributed file system. At least one application communicates with the distributed file system. The notification system notifies the corresponding application associated with the command with the notification system.

## V. DISTRIBUTED FILE SYSTEM INCLUDING MULTICAST RETRIEVAL

A distributed file system intelligently allocates portions of a file system from a server to one or more clients on a network. File system data used during power-on of a client may be stored on the client. A retrieval of the file system may subsequently be made to the extent the client has capacity to store the file system. A multicast operation, performed as a background operation on the client, may be used to retrieve the file system. For portions of the file system not allocated to the client, the client may still access the file system from the server. Network bandwidth may thus be reduced and client access to the file system may generally be more efficient.

## VI. DISTRIBUTED FILE MANAGEMENT CACHE MANAGEMENT

One or more cache management technique(s) are implemented in a distributed file system of a computer network to enforce equitable use of the cache among the file data in the cache. One of thetechnique is a timestamp handicapping routine that functions to keep small-to-medium files in the cache. Another technique implements a "cache quota", which is used for decreasing the percentage of the cache a single file may consume when there are other data in the cache. When caching of a single file approaches the cache quota, the file data is looks older than it really is so that upon a subsequent file [6]I/O operation, portions of such datas are recycled from the cache earlier than they would have been otherwise. When caching of a single file reaches the cache quota, the file must begin to reuse cache from itself. The cache quota technique has the effect of causing cached data towards the end of large files to get recycled from cache first. A third technique helps to detect the file I/O that is not conducive for caching, such as sequential I/O on a file that is larger than the entire cache. A cache policy prevents a large file from stealing cache space by establishing a small, but specific area of cache in which portions of such a large file may be stored and recycled without requiring least recently used (LRU) evaluation process.

## VII. MAP REDUCE READY DISTRIBUTED FILE MANAGEMENT SYSTEM

A map-reduce compatible distributed file system consists of successive component layers that provide the basis on which the next layer is built provides transactional read-write-update semantics with file chunk replication and huge file-create rate. A primitive storage layer (storage pools) knits together raw block of stores and provides a storage mechanism for the containers and transaction logs. Storage pools are manipulated by each file servers. A container location database allows the containers to be found among every file servers, as well as defining precedence among the replicas of the containers to organize transactional updates of container contents. The Volume facilitates control of data placement, and retention of a variety of control and policy information. Key-value stores related keys to data for such purposes as directories, and container location maps in compressed files.

## VIII. WEB SERVER USER AUTHENTICATION WITH COOKIES

A method of authenticating a Web client to the Web server connectable to a distributed file system of distributed computing environment. The distributed computing environments have a security service for returning of a credential to a user authenticated to access the file system. In response to receipt by the server of a user id and password from the Web client, a login protocol is then executed with the security service. In particular, when Web client desires to make a subsequest request to the distributed file system, the persistent client state object along with the identifier is used in lieu of the user's id and password, which makes the session much more safe. In this operation, the cookie identifier is used as pointer into the credential storage table, and the credential is then retrieved back and used to facilitate the multiple file accesses from the distributed file system. On the same time, the Web client may obtain access to the Web server (as opposed to distributed file system) documents via conventional user id and password in an HTTP request.

## IX. CONCLUSION

The purpose of a distributed file system (DFS) is to allow many users of physically distributed computers to share data and storage resources by using a common file system. A typical arrangement for a DFS is a collection of workstations and mainframes connected by a local area

network. A distributed file system is a client/server-based application that allows clients to access and process data stored on server as if it were on their own computer. Ideally, a distributed file system organizes file and directory services of anindividual server into a [8]global directory in such a way that remote data access is not location-specific but it is identical from any client. All files are available to all users of the global file system and organization is anhierarchical and directory-based.

## X. REFERENCES


[1]. Howard J. "An Overview of the Andrew File System".

[2]. Sandberg R., Goldberg D., Kleiman S., Walsh D., Lyon B., "Design and Implementation of the Sun Network Filesystem".

[3]. Giacomo Cabri, Antonio Corradi, Giacomo Cabri, Antonio Corradi, Franco Zambonelli"Experience of Adaptive Replication in Distributed File Systems"- Copyright IEEE. Published inthe Proceedings of EUROMICRO '96, September 1996 at Praha, Chzech Republic.

[4]. Gerald Popek, Richard Guy, Thomas Page, John Heidemann "Replication in Ficus Distributed File Systems"- Proceedings of the Workshop on Management of Replicated Data, November 1990.

[5]. ABOUZIED, A., BAJDA-PAWLIKOWSKI, K., ABADI, D. J., SILBERSCHATZ, A., AND RASIN, A. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. In Proceedings of the 35th Conference on Very Large Data Bases (VLDB) (Lyon, France, 2009).

[6]. ALVARO, P., CONDIE, T., CONWAY, N., ELMELEEGY, K, HELLERSTEIN, J. M., AND SEARS, R. BOOM analytics: Exploring data-centric, declarative programming for the cloud. In Proceedings of the 5th European Conference on Computer Systems (EuroSys) (Paris, France, 2010).

[7]. A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R.Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Dec. 2002.USENIX.

[8]. P. A. Alsberg and J. D. Day. A principle for resilient sharing of distributed resources. In Proceedings of the 2nd International Conference on Software Engineering, pages 562–570. IEEE Computer Society Press, 1976.