VOL. 10, NO. 10, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

IMPLEMENTATION OF ACTIVE STORAGE IN EFFICIENT VIRTUAL FILE SYSTEM

C. Saranya, V. Loganathan and S. Godfrey Winster
Saveetha Engineering College, Chennai, Tamilnadu, India
E-Mail: csaran08@gmail.com

ABSTRACT

In present decades nearly all the computation tasks use colossal volume of data. Moreover these computation tasks are additionally of elevated performance. These tasks transfer's data from the storage node to the computing node. The data transfer rate amid storage and compute nodes is elevated and to minimize it the active storage was introduced. Active storage increases the storage node's processing power and it additionally reduces the web traffic. Active storage was introduced in parallel file system's like lustre parallel file system, red hat GFS, etc. The proposal is to craft n number of adjacent file system in windows working system. In this the data can be stored and it does not inhabit space in the host system drive or memory. The data stored in the file system ought to inhabit less space for that the data is being compressed and then stored in the file system. The data is additionally safeguarded by encrypting the data before storing it into the file system. This file system is portable to any other system. Our target is to make the file system to an active storage. The file system is projected in such a method that it can present effectual fetching of data from the file system.

Keywords: active storage, lustre parallel file system.

1. INTRODUCTION

In computing, a file system is to manipulation the data stored in files and reclaiming or accessing those data or files. File systems can be utilized on countless disparate kinds of storage devices. It manages admission to both the content of files and the metadata concerning those files. It is accountable for arranging storage space; reliability and efficiency. File system allocate space in a well described manner, normally several physical constituents on the device. The file system is accountable for coordinating files and directories, and keeping trail of that spans of the mass media fit in to that file. It stores all the metadata data associated alongside the file encompassing the file term, the length of the contents of a file, and the locale of the file in the folder, distinct from the contents of the file. There are countless disparate kinds of file systems. Every single one has disparate construction and logic, properties of speed, flexibility, protection, size and more. A little file systems have been projected to be utilized for specific applications. The main setback in these file systems is to how effectually the needed data is being fetched from those file system for giving the computing tasks.

The main key feature in file system is indexing of files in the file system. An index encompasses a collection of data entries and this supports effectual retrieval of all data in the file system. Indexing is a data construction method for accessing records in a file. There are assorted Indexing methods has been evolved for the file system. Amid them the present trend is the semantic established indexing. The automatic indexing of files and gathering established on relativity is called "semantic". In the Semantic indexing, the user programmable nature of the system uses data concerning the semantics of uploaded file arrangement objects to fetch the properties for indexing. The setback in semantic established indexing method is that it does not furnish each level of protection to the data that is being stored in the file system.

Nowadays all the computing tasks are elevated computing tasks. These computing tasks use huge volume of data transfer amid the storage and compute nodes. Inorder to cut the data transfer amid nodes, active storage was introduced. The data transfer was decreased by active storage by rising the processing ability of storage nodes, and those computing tasks are done by the storage nodes itself. The active storage was implemented in parallel file systems such as Lustre file system, Redhat GFS, Parallel Virtual File System. The storage node ought to be capable plenty to perform those high performance computing tasks. The subject recognized here is that active storage has been requested merely in the linux clusters no one has dealt it with the windows file system.

2. RELATED WORK

Y. Hua, H. Jiang, Y. Zhu, *et al* [5] provides Semantic Namespace scheme that is being evolved to vanquish the difficulties in hierarchical directory established construction that is tough to use and less scalable for present colossal scale systems. Semantic Namespace provides a flat but tiny, manageable and effectual namespace for every single file. Here, a file is embodied by its semantic correlations to supplementary files, instead of standard file names. The metadata of files that are powerfully correlated are automatically aggregated and next stored jointly in semantic namespace. After a user performs a file lookup, semantic namespace will additionally present the user files that are powerfully correlated to this hunted file that contain the semantic-aware per-file namespace of this file. This permits the user to access the correlated files facilely lacking possessing to present supplementary searches. This enhances the performance of the system. Semantic namespacing leverages Locality Sensitive Hashing [7] to automatically coordinate semantically correlated files lacking the involvement of end-users or applications. But there is no protection for the data stored in the file system.

The proposal of active storage [1] is an expansion to established active disk [9] by adopting it to the parallel file system. In preceding days, the processing manipulation of disk controller CPU was utilized inorder to process the data stored on the disks. Nowadays the number of the data has been increased incredibly and inorder to grasp, transfer and process those data, the computing and storage nodes ought to demand huge processing power. For that Evan J. Felix, Jarek Nieplocha, *et al*. [1], counselled the active storage concept. Active storage is a knowledge aimed at cutting the bandwidth necessities of present supercomputing systems and to impact the processing ability of the storage nodes by a little present file systems. Active storage moves precise computing tasks from compute nodes to the storage nodes whereas the demanded data for the task resides. Active storage differs from active disk by two main things such as:

- Storage servers are full-fledged computers.
- They contain a feature affluent nature endowed normally by a linux working system.

The active storage has been implemented in a little parallel file systems such as Lustre file system, Parallel Virtual file system [4]. Atfirst, active storage was introduced in Lustre that is an open source file system. In this proposal, active storage was implemented merely in the kernel space of the lustre file system and consequently it is believed as a traditional approach.

Later the implementation was completed in the user space of the file system additionally by Juan Piernas-Canovas, Jarek Nieplocha [2]. This evaluation depicts that the user space implementation of active storage is extra effectual than that of the established kernel space implementation.

3. OBJECTIVE

The main target of the proposed system is to create number of file systems whereas we can store our files, documents and images. The files will be stored in containers crafted by the user. A file system can have number of containers alongside it. The containers have various blocks of files that are to be stored in the file system with assorted file formats. The uploaded files will splitted and stored in the containers. The active storage is being utilized in the file system for effectually retrieving the data from the containers of the file system. Inorder to furnish protection to the file system, the data that are being uploaded into the file system are being encrypted by employing the Rijndael Encryption algorithm. Additionally inorder to effectual grasp the storage space in the file arrangement, the data is being compressed and next stored in the container so that it will inhabit less space and consequently we can store huge volume of data than ever before. After a file is being uploaded in the file system, the index for the file is generated established on the data present in the file. This kind of indexing is recognized as semantic established indexing technique. The main supremacy of the system is that the crafted file system can be ported to any other system.

4. FILE SYSTEM ARCHITECTURE

In this section we first give the overview of our proposed system. Then we define the proposed file system architecture.

4.1 Proposed file system

In Semantic aware name spacing scheme, the files are indexed based on contents present in the file. Though in semantic namespacing is used for easy retrieval of data from file system, there is no security for the data stored in it. These file arrangements inhabit huge recollection in the system. All the elevated presentation computing are completed merely in linux cluster file arrangements such as lustre, PVFS, etc alongside alert storage. But there is no effectual file arrangement in windows working arrangement alongside the implementation of alert storage. File arrangement cannot be ported from one working arrangement to another. Hence to overcome the existing systems difficulties, the proposed work is to create a file system by implementing active storage in it for windows operating system. The created file system is being provided with data encryption and data compression and it will be portable to any other systems. By providing encryption to data, the files will be secured and through compression the disk space occupied by the file system will be reduced.

The main proposal of the project is to create a modernized file system in windows operating system. This modernized file system will be a portable file system with security for the data. The proposed system at first creates a number of file systems. Our system asks the user for the count of the file system to be created for them. Then the system creates the container for uploading the files in the container. The files with different files formats are placed in various containers. Therefore the files are arranged as blocks of similar format files. The data stored in the file arrangement are being automatically indexed established on semantic namespacing scheme. Indexing of data helps us to reclaim the data effectually in the subsequent period of our system. The concept of active storage is being used in the efficient retrieval of data from the file system.

Inorder to provide security, the data stored in the containers will be encrypted. Another main aim is to lower the memory space occupied by the files by compressing the files before placing it in the containers. To the best of our knowledge no system has the encryption and compression of files in semantic namespacing scheme.

Therefore the file is being compressed, encrypted and splitted simultaneously and then placed in the container. When a user places a request for the data, the file system should efficiently fetch the data and then process those data for performing the task.

4.2 Proposed file system architecture

Figure-1 depicts the architecture of the proposed efficient virtual file system. In this proposed system of file system creation the user can store any kind of data. The data format that is being uploaded can be of doc, jpg, img, mov, etc. At the start of our system design, we generate a user defined file system. The user is allowed to create any number of file system according to the needs. After creating the file system, the user has to login into the system inorder to provide security to access the file system. Once the login is successful, the user can create containers. The containers are used to organize the data stored in it. Then the users are asked to create the objects. Objects are the index for grouping the files. The files that are named under the same object are grouped together and placed in the same container. In our file system, the file is stored in the form of blocks. Each file blocks are of fixed size and files are splitted according to it. For example, if the user uploads a file size of 10MB and it is defined that block size as 1MB, then 10MB file will be splitted into 10 file blocks each of 1MB in size. In some cases the size of the last block will be less than the defined block size.

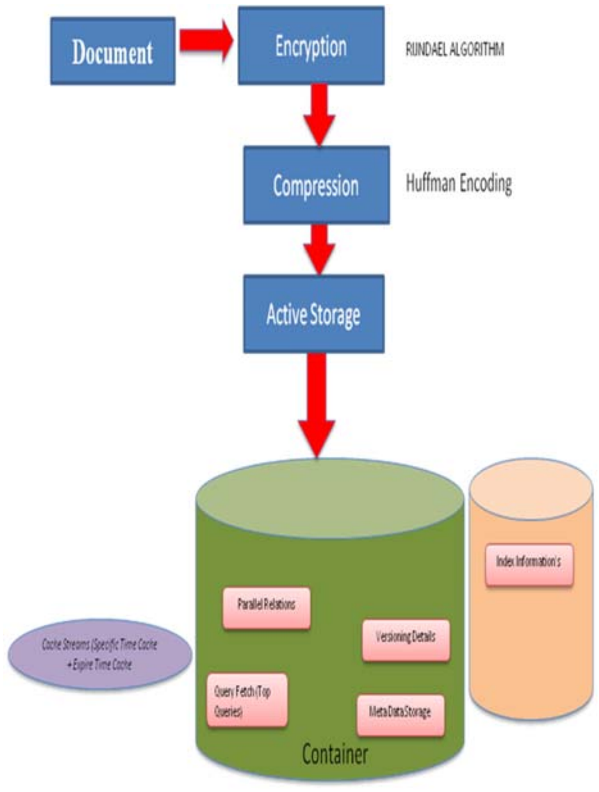

Figure-1. File System Architecture.

While uploading the files inside the container, the file will be compressed inorder to reduce the amount of memory space it occupies inside the container. Then the compressed file will be encrypted inorder to provide security for the file system. The algorithm utilized for compression is Huffman encoding algorithm and for encryption Rijndael Encryption algorithm is used. The files are indexed established on the contents stored in the files. The files listed under same objects are grouped together automatically which is similar to the semantic indexing technique. Based on those indexes, the files are searched upon an user request for the data in the file system. The storage system will perform the computing task after fetching the information from the container.

5. DESIGN AND IMPLEMENTATION

The implementation of virtual file system is explained in various sections described below. The aim of generating the file system is to efficiently index the files using semantic aware indexing technique and fetch those files from the file system based on the user query.

5.1 File system and container creation

The user is allowed to create the virtual file system. The user is at first allowed to enter the number of file system's to be created by the user. The system asks the user to create the file system by providing the name for the file system. After providing the name for the file system, the system creates the file system for the user. The specific universal PVFS commands are used for creating the file system.

After creating the file system the user can able to upload the files in the files system, but when the user uploads many files with various file formats then it will be difficult to categorize those files. Without containers the files will be scattered throughout the file system. The containers are used to wrap the files with different file formats. Therefore this module allows the user to create number of containers according to the user needs. The container will be created in that user specified name. Now the container is ready for receiving the files or data that needs to be stored in those containers. The containers store the files as blocks of data in it.

5.2 Data uploader

While uploading data into container, the files are to be splitted. The files are splitted by the system manually based on the number of containers created by the user. Each splitted file is placed in all the created containers. But before placing it in containers the splitted parts are being compressed and encrypted by the system. The encryption is being done by the Rijndael algorithm. The compression is being performed by Huffman Encoding algorithm.

5.2.1 Huffman encoding

The Huffman encoding algorithm is an optimal compression algorithm where merely the frequency of individual messages is utilized to compress the data. The system behind the algorithm is that if you have a characters that are extra recurrent than others, it makes sense to use less bits to encode those letters than to encode the less recurrent letters. Huffman's scheme uses a table of frequency of occurrence for every single signal (or character) in the input. This table could be derived from the input itself or from data that is representative of the input. It next demands to allocate a variable-length bit thread to every single character that unambiguously embodies that character. An encoding for every single character is discovered by pursuing the tree from the path to the character in the leaf: the encoding is the thread of symbols on every single division followed. The symbols can be 0's and 1's. The steps to be pursued in the algorithm:

- Scan text to be compressed and tally occurrence of all characters.
- Sort or prioritize acts established on number of occurrences in text.
- Build Huffman program tree established on prioritized list.
- Perform a traversal of tree to ascertain all program words.
- Scan text once more and craft new file employing the Huffman codes.

5.3 Index tuning and operational detail

To enhance the performance of the document storage, the index for each compressed file will be placed on hold to the documents. The documents will be hierarchically stored and indexed based on the logically metadata information about the documents. The indexing of data is done through latent semantic indexing technique which indexes based on contents in the file.

This part of implementation provides a detailed view about the operations happening on the system. The file system is made to process the input output procedures above the data that are being present in the file system. The file search and all their file I/O operations are being performed by the system itself. After the semantic search and the personalized ranking, the recommender shows the list of found document units to the user. For each of the retrieved document units the user can also see additional information that come from its annotation data (e.g., the list of annotation concepts, the number of reuses, the number and list of users, the list of documents in which it appears and the number of versions).

6. CONCLUSION AND FUTURE WORK

The file system crafted in each system that is being developed will inhabit the disk space in the system where it is being crafted or installed. The crafted file system ought to inhabit less space in the arrangement drive. None of the existing file system provides security for the data stored in the file system. Our proposed work provides security by splitting files and encrypting the files in the container. In the namespace scheme, semantic based namespace system is being used which is a recent trend. The created file system is portable which is not being done in any other file system. Our file system performs the file I/O operations. In future the work can be extended to make the file system to perform all computing tasks.

REFERENCES

[1] Evan J. Felix, Kevin Fox, Kevin Regimbal, and Jarek Nieplocha. 2006. Active Storage Processing in a Parallel File System. Proceedings of the 6th LCI International Conference on Linux Clusters: The HPC Revolution.

[2] Juan Piernas-Canovas and Jarek Nieplocha. 2010 Implementation and Evaluation of Active Storage in Modern Parallel File Systems. Elsevier Journal. Proceedings of Parallel computing. pp. 26-47.

[3] Juan Piernas, Jarek Nieplocha, and Evan J. Felix. 2007. Evaluation of Active Storage Strategies for the Lustre Parallel File System. Proceedings of the Supercomputing Conference (SC07).

[4] Philip H. Carns and Walter B Ligon III. 2000. PVFS: A Parallel File System for Linux Clusters. Proceedings of the Extreme Linux Track: 4th Annual Linux Showcase and Conference.

[5] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian. 2009. Rapport: Semantic-sensitive Namespace Management in Large-scale File Systems. Proceedings of ACM/IEEE Supercomputing Conference (SC).

[6] Seung Woo Son, Samuel Lang, Philip Carns, Robert Ross, and Rajeev Thakur. 2010. Enabling Active Storage on Parallel I/O Software Stacks. IEEE 26th Symposium, Mass Storage Systems and Technologies (MSST).

[7] M. Datar, N. Immorlica, P. Indyk and V. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. Proc. Annual Symposium on Computational Geometry, pp. 253-262.

[8] Yu Hua, Hong Jiang, Yifeng Zhu, Dan Feng, Lei Xu. 2013. SANE: Semantic-Aware Namespace in Ultra-large-scale File Systems. IEEE Transactions on Parallel and Distributed Systems. 1(1).

[9] A.Acharya, M. Uysal and J. H. Saltz. 1998. Active Disks: Programming Model, Algorithms and Evaluation. Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 81-91.

[10] R. Motwani, A. Naor, and R. Panigrahy. 2006. SLower bounds on locality sensitive hashing. ACM Symposium on Computational Geometry.