ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 5, May 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

A Competent Modus Operandi for Hurling Video and Images with Designated Access Rights

V. Archana[1]
ME-CSE
Kingston Engineering College
Vellore, Tamil Nadu – India

M. Tamil Thendral[2]
Assistant Professor, CSE
Kingston Engineering College
Vellore, Tamil Nadu – India

Abstract: ***The Tremendous increase in size or quantity of files like images, videos and its complexity to accept is a great challenge for file system. SANE is implemented as a transparent middleware that can be deployed or embedded in most existing file systems without modifying the kernels or applications. It is used to compress the uploaded images and videos and store in a ultra file system by using Huffman coding algorithm. An access control mechanism is used in Semantic aware namespace (SANE), using this mechanism the user can set privileges to each friend as what he/she views. The image search can be done with the help of tags and files can be shared to any user with the help of tags. Triple DES Algorithm is used for encryption. The privilege can be set for the users of a explicit group, where the access rights can be revoked from selected members, if in case deemed. In the context of video, since the structure of video data is complex in nature, it requires a specific access control mechanism.***

Keywords: Compression method, Tags, Extract keyword, Rapid Search, Hierarchical file.

I. INTRODUCTION

In Modern storage system are storing billion of file in Petabytes of storage. Organizing and managing this data has become daunting task for both user and administrators for several reasons. User need to find file particular characteristics in the vast sea of data, administrator know and understand the nature of data stored to more effectively manage the storage. Both tasks require the ability to efficiently answer questions about the properties of the data being stored. Thus fast and scalable searches over file metadata benefits both user and administrator. In almost all today's file systems, the namespace management is based on hierarchal directory tree. The most important function of namespace management is file identification and lookup. As the data volume and complexity keep increasing rapidly, conventional namespace schemes based on hierarchical directory trees.

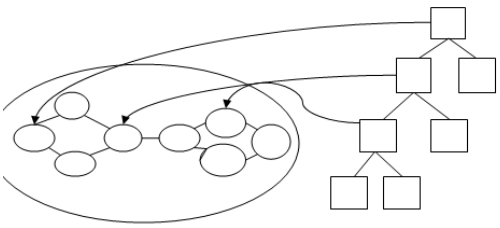


Fig.1. SANE and Hierarchical directory tree

SANE is also integration into modern file system such as PNFS, PVFS, GFS and HDFS. Our goal in this research is to complement existing file system and improve the system performance. Our major contributions are summarized below. First addressing the weaknesses 1, SANE is mainly designed semantic correlations residing in multi-dimensional attributes, rather

than the one-dimensional attributes such as pathname to represent. User can search the files by using pathname it's easy, but user don't know the pathname means it will take time to search the file.

II. DESIGN OVERVIEW

In designing an ultra file system for our needs, we have guided by assumption that offer both challenges and opportunities. We aim to exploit semantic correlation among files to define one flat, small, but accurate namespace for each file. We discuss in detail. To illustrate how sane works, we briefly describe its overall architecture. Sane explores semantic correlations residing in files to build the namespace representation that can accurately identify a file and track the evolution of file attribute. Sane includes three key function modules, Semantic correlation identification (SCI), namespace construction (NC) and keyword extraction in document (ked), as shown in fig .2. To quickly identify correlated files, the SCI will perform the locality sensitive hashing computation based on multiple file attributes such as tags, the NC will aggregates semantically correlated files into group by searching nearest neighbor file. The ked will use to uploading the document, extract keyword from the document and save it's as tags, by using the keyword the document can be easy access by the user by using stemming algorithm.

III. MODULES

3.1 File system creation

User creates a new account to upload the images. Each user has a separate container to upload the images. Tree like container is used for storing images from users. The user can be searched for a specific image or a group of images based on keywords. Existing containers like WinRAR just compress and store various image. Each file system (fs) has a root file which stores the images (I) in the file system. When an image is uploaded, tags are specified for each image. This is used to search the image from the root file in high speed search. The tags are stored in database (db). Each image have list of tags. The compressed images are stored in the root file. If there is a degree edges about the file system already means add the images to edge or else create a new edge and add the images on those edge.

Algorithm

```
Input: user u, list of images l []
Output: An ultra Filesystem (fs)
Begin
Create a root file to store metadata of fs
for each image img in I
Compress_Encrypt (img)
For each tags t in img Store t in db
If there is a degree edge (e) about in the file system fs then add img to e
else
create edge and add img to e
next
End
```

3.2 Compression and encryption

The images stored in the Filesystem should be compressed to save disk Space. For security reasons, the images are encrypted. The compression method used in the project is Huffman encoding/decoding. The encryption used in the project is Triple DES.

Algorithm:

Input: Text T

Output: Compressed Text CT

Begin

Calculate the letter frequency in T and store in the array freq ()

Sort freq () in descending order

Create a priority queue Q, to store the array freq ()

While not empty Q

Get the two higher nodes in the Q

Merge the nodes to form tree Tr

repeat

Label all the left edges as 0 and right edges as 1in tree Tr

replace the letters in the T to value of letter in Tr

CT= replaced Text

return CT

End

The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols, n. A node can be either a leaf node or an internal node.

ENCRYPTION

Triple DES uses a "key bundle" that comprises three DES keys, K1, K2 and K3, each of 56 bits (excluding parity bits). The encryption algorithm is: Cipher text = EK3(DK2(EK1(plaintext))) i.e., DES encrypts with K1, DES decrypt with K2, then DES encrypt with K3.The Decryption is the reverse: Plaintext = DK1(EK2(DK3(Ciphertext))) i.e., decrypt with K3, encrypt with K2, then decrypt with K1. Each triple encryption encrypts one block of 64 bits of data. In each case the middle operation is the reverse of the first and last. This improves the strength of the algorithm when using keying option 2, and provides backward compatibility with DES with keying option

The standards define three keying options:

- Keying option 1: All three keys are independent.
- Keying option 2: K1 and K2 are independent, and K3 = K1.
- Keying option 3: All three keys are identical, i.e. K1 = K2 = K3.

Keying option 1 is the strongest, with 3 × 56 = 168 independent key bits.

Keying option 2 provides less security, with 2 × 56 = 112 key bits. This option is stronger than simply DES encrypting twice, e.g. with K1 and K2, because it protects against meet-in-the-middle attacks. Keying option 3 is equivalent to DES, with only 56 key bits.

3.3 Tag management

The images are compressed and stored in the ultra file system in the containers. The searching can be done without decompressing. The keywords used to search called tags. Each image has list of tags. The tags may be a place name, name of persons in the image, tour name etc… once the tag is specified we can easily fetch the image from the container. The index is created to store the tags in the database which is easily fetch the tags from the database

3.4 DICTIONARY

The searches of images using tags cannot be efficient until a semantic link is made. At the time of uploading the images providing list of tags to each image. The tags can be stored in index. Searching the images with the help of tags which is semantically refer the dictionary. For example: The list of tags is specified to Tiger such as Tiger, National animal and forest cat. Suppose user can search and fetch the image from the list if tags. The tags have same semantic meaning. During a search, both the tags and its semantic meaning are used to search the images.

3.5 ACCESS CONTROL MATRIX

An Access Control Matrix or Access Matrix is an abstract, formal security model of protection state in computer systems that characterize the rights of each subject with respect to every object in the system. It was first introduced by Butler W. Lampson in 1971. An access matrix can be envisioned as a rectangular array of cells, with one row per subject and one column per object. The entry in a cell that is, the entry for a particular subject-object pair indicates the access mode that the subject is permitted to exercise on the object. Each column is equivalent to an access control list for the object; and each row is equivalent to an access profile for the subject. When a user uploads an image, before sharing to others he can set the privileges to other users. The privileges can be changed at any time, even after sharing the container. This increases the security. The sender uploads the image and downloads the container to transmit the images to receiver. The receiver will upload the images and upload the container to view the images on the sender side. Sender will set the Access Control mechanism to set the privilege what he/she want to view.

Algorithm
Input: A container C, Tag T
Output: Image View V
Begin
Boolean flag=getACM (Currentuser, T)
if(T==true) then
V=download Image from Container
else
V=Nothing
End IF
return V
End Algorithm

After the receiver views or download the images, the sender will remove the privilege. After the sender removes the access control, the receiver cannot be able to view or even download the images from the sender sides, and this proves the security of the system.

3.6 HURLING FILES TO A EXPLICIT GROUP

Hurling files like video and images to the explicit group from the number of groups. The image can be observed by all members who are all in the group. For example whatsapp, google, group.

3.7 DESIGNATED ACCESS RIGHTS TO ELECTED MEMBERS.

After hurling the file to explicit group, the files can be observed by all the members. By providing designated access rights the elected members can only observed the file like images and video other members in that group can't observe the file.

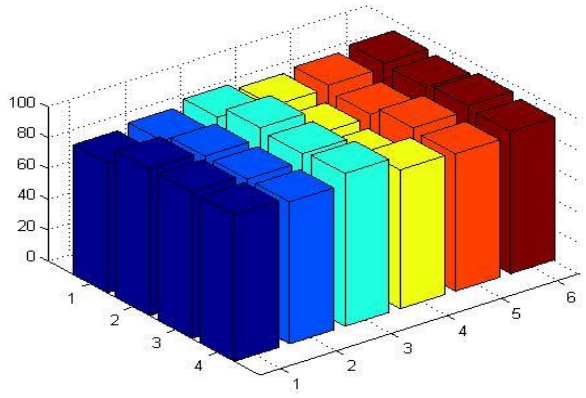
3.8 REVOKING ACCESS RIGHTS TO ELECTED MEMBERS.

Designated access rights are provided to elected members. The Elected members can only observed the files for certain period of time like 3days, 1 hour. After I revoke the access rights to elected members, the elected members also can't able to observe the files.

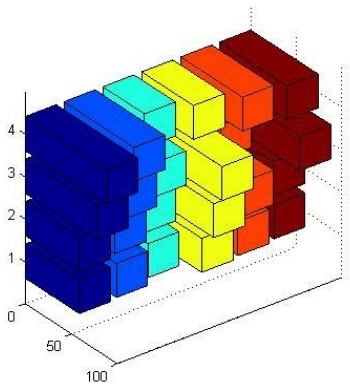
4 PERFORMANCE EVALUATIONS

4.1 Experimental Setup

Raised area we have implemented a trial product of SANE in windows 7. SANE use three metrics, namely, cost-effectiveness, searchability and scalability. All Experiments have taken into account the vibrant growth of file systems, such as box file creations and deletions. In addition, the user interfaces of SANE for namespace representation, renaming and query service are also implemented in our prototype. We design and implement most modules at the user space level so that our prototype can run on many existing systems. The prototype has implemented the three basic function components of SANE discussed in the previous Section, namely, LSH-based semantic connection detection, namespace construction of attributes and correlations. Intensified Traces. We use four spokesperson traces, of which three collected from the industry and one from the academia. These traces include HP file system trace. MSN trace EECS NFS server (EECS) trace at Harvard and Google clusters trace, which drive the SANE performance evaluation. Moreover, for the sizes of these traces, Google cluster contains 75 five-minute reporting intervals. There are a total of 3,535,029 observations, 9,218 unique jobs and 176,580 unique tasks. HP contains 94.7 million requests for a total of 4 million files from 32 users. MSN has 1.25 million files and records 4.47 million operations, in which there are 3.3 million read and 1.17 million write operations. EECS contains 4.44 million operations. The number and size of read operations are respectively 0.46 million and 5.1 GB. Those of write operations are respectively 0.667 million and 9.1 GB. Due to their relatively small sizes, we use a scaled-up method to intensify them. In order to emulate the I/O behaviors of large-scale file systems for which no realistic traces are publicly available, we scaled up the existing I/O traces of current storage systems both spatially and temporally. This method has been successfully used in Glance [12] and SmartStore [15]. Specifically, a trace is first decomposed into sub-traces. We then add a unique sub-trace ID to all files to intentionally increase the working set. The start times of all sub-traces are set to zero so that they are replayed concurrently. The chronological order among all requests within a sub-trace is faithfully preserved. The combined trace contains the same histogram of file system calls as the original one but represents a heavier workload (higher intensity). The number of sub-traces replayed concurrently is denoted as Trace Intensifying Factor (TIF) and in our experiments the default TIF value is 400. Moreover, the multi-dimensional attributes chosen for this evaluation are faithfully extracted from the original traces. In addition, in order to obtain reasonable R values for the LSH computation (Section 3.2) in our experiments, we use the sampling method, which has been verified through practical applications. We determine the R values to be 1,200, 800, 1,000 and 950, respectively for the intensified HP, MSN, EECS and Google traces. In fact, the SANE system can be dynamically configured according to the requirements of users or systems, such as query accuracy and latency, available space, bandwidth and computing resources choose specific comparison schemes in order to make the comparison relevant and fair. We compare SANE with the target schemes only in the relevant aspects such as namespace construction, query performance for users, and file prefetching and data deduplication for systems. To evaluate cost effectiveness, we choose to compare SANE with the Ext4 file system (Linux kernel 2.6.28) that serves as a classic representative of the conventional namespace schemes based on hierarchical directory trees. In order to support directory indexing, Ext4 examines a hashed B-tree that uses a hash table of the filenames. For Searchability, we select some state-of-the-art comparable systems, including Spyglass [1] and SmartStore [4].



Both systems support various types of queries in file systems with competitive performance. Note that since there is no open source code available for Spyglass, we implemented its main components, such as the crawler, multiple partitions and versions, and K-D tree, according to the descriptions presented in [7]. In the experiments, for a given file f, SANE first chooses its nearest neighbor file (t ¼ 1) as the member of its namespace. We then obtain the value of MSDðfÞ. If the value of MSDðfÞ increases after adding its next most closely correlated file to the namespace, the file is considered a member of file's namespace and t is increased by 1, while MSDðfÞ is updated accordingly. Otherwise, the namespace construction finishes. The bounds (minimum and maximum) and average values of t in four traces are respectively (min-max: 22-121; average: 31.6) in MSN, (min-max: 11-31; average: In general, filename-based point query is very popular in most file system workloads. There are no file system I/O traces for both point and complex queries (range and top-k) requests. In order to address this issue, we leverage a synthetic approach to generating not only point query, but also complex queries within the multi-dimensional attribute space. The basic idea is to statistically generate random queries in a multi-dimensional space. We study the file static attributes and behavioral attributes from the available I/O traces. For example, a point query in the form of (17:50, 85.2, 36.5) represents a search for the files that are closest to the description of a file that is last revised at time 17:50, with the amounts of "read" and "write" data being approximately85.2 and 36.5 MB. Moreover, a range query aiming to find all the files that were revised between time 8:20 to 10:50, with the amount of "read" data ranging from 22 to 40 MB, and the amount of "write" data ranging from 7 to 12 MB, can be represented by two points in a three-dimensional attribute space, i.e., (8:20, 22, 7) and (10:50, 40, 12). Similarly, a top-k query in the form of (9:30, 17.2, 75.8, 5) represents a search for the top-five files that are closest to the description of a file that is last revised at time 9:30, with the amounts of "read" and "write" data being approximately 17.2 and 75.8 MB, respectively.



In addition, the semantic correlations recognized in SANE create new opportunities for system designers to implement system optimizations from a totally new perspective. In this paper, we take data deduplication to illustrate potential benefits of

SANE from a system perspective. We have leveraged semantic correlations identified by SANE to optimize system function of deduplication.

IV. Conclusion and Future Enhancement

This paper has focused on describing the nature of SANE-Semantic Aware Namespace. Also a broad list of modules has been discussed. We propose to extend this paper by implementing a Hurling files like video and images to the explicit group from the number of groups. The image can be observed by all members who are all in the group. For example whatsapp, go ogle group. by providing designated access rights elected members can only observe the file for a certain period of time and revoking the access rights to the elected members to observe the file.

Acknowledgement

I would like to take this opportunity to express my profound gratitude and deep regard to my guide, Assistant Professor M.Tamil Thendral CSE, Kingston Engineering College, for his exemplary guidance, valuable feedback and constant encouragement in completing this paper. His valuable suggestions were of immense help in getting this work done. Working under him, was an extremely knowledgeable experience. Also, I would like to extend my sincere gratitude to my parents for their constant support and encouragement in completing this paper.

References

1. A.W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E.L. Miller, “Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems,” Proc. Seventh USENIX Conf. File and Storage Technologies (FAST), 2009.
2. H. Huang, N. Zhang, W. Wang, G. Das, and A. Szalay, “Just-In- Time Analytics on Large File Systems,” Proc. Ninth USENIX Conf. File and Storage Technologies (FAST), 2011.
3. K. Veeraraghavan, J. Flinn, E.B. Nightingale, and B. Noble, “quFiles: The Right File at the Right Time,” Proc. USENIX Conf. File and Storage Technologies (FAST), 2010.
4. Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian, “SmartStore: A New Metadata Organization Paradigm with Semantic-Awareness for Next-Generation File Systems,” Proc. ACM/IEEE Supercomputing Conf. (SC), 2009.
5. D. Beaver, S. Kumar, H. Li, J. Sobel, and P. Vajgel, “Finding a Needle in Haystack: Facebooks Photo Storage,” Proc. Ninth USENIX Conf. Operating Systems Design and Implementation (OSDI), 2010.
6. PVFS2. Parallel Virtual File System, Version 2, http:// www.pvfs2.org, 2013.
7. Hadoop Project, http://hadoop.apache.org, 2013.
8. C. Maltzahn, E. Molina Estolano, A. Khurana, A. J. Nelson, S. A. Brandt, and S. Weil, “Ceph as a Scalable Alternative to the Hadoop Distributed File System,” login: The USENIX Magazine, vol. 35, pp. 38-49, Aug. 2010.
9. “Symantec. 2010 State of the Data Center Global Data.,” http:// www.symantec.com/content/en/us/about/media/pdfs/ Symantec_DataCenter10_Report Global.pdf, Jan. 2010., 2013. Gorton, P. Greenfield, A. Szalay, and R. Williams, “Data- Intensive Computing in the 21st Century,” Computer, vol. 41, no. 4, pp. 30-32, 2008.S. Patil and G. Gibson, “Scale and Concurrency of GIGA+: File
10. System Directories with Millions of Files,” Proc. Ninth USENIX
11. Conf. File and Storage Technologies (FAST), 2011.
12. J. Xing, J. Xiong, N. Sun, and J. Ma, “Adaptive and Scalable Metadata Management to Support a Trillion Files,” Proc. ACM/IEEE Supercomputing Conf. (SC), 2009.
13. S. Weil, K. Pollack, S. Brandt, and E. Miller, “Dynamic Metadata Management for Petabyte-scale File Systems,” Proc. ACM/IEEE Super computing, 2004.

AUTHOR(S) PROFILE

V. Archana received the B.Tech (IT) degree in 2014 from Priyadarshini Engineering College, India. She is a post graduate student in the Computer Science Department, Kingston Engineering College, India. Her research interests are Network Security. She has published two papers.

M. Tamil Thendral, Assistant professor, Department of computer science, Kingston Engineering College. He received his B.Tech (IT) degree in 2005 from SKP Engineering College, India and He then complete his M.E degree in 2011 from Madha Engineering College, India, respectively. His area of interest is Network Security. He has published four papers and has attended 2 conferences.