isrjournals

International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 4, Issue: 3,Special Issue: 1 ,Mar,2016 ,ISSN_NO: 2321-3337

# 7 Zip Compression

P. Janakiramal, Dr.D.C.Jullie Josephine

M.E Computer Science, James College of Engineering and Technology, India.
Professor, Dept.of Computer Science, Kings Engineering College, India

***ABSTRACT—****The namespace management is based on hierarchical directory trees. This tree-based namespace scheme is prone to severe performance bottlenecks and often fails to provide real-time response to complex data lookups. The paper proposes a Semantic-Aware Namespace scheme, called SANE, which provides dynamic and adaptive namespace management for ultra-large storage systems with billions of files. Associative access on the files is provided by an initial extension to existing tree structured file system protocols, and by the use of these protocols that are designed specifically for content based file system  access. Access on the file details such as versions or any other concepts were interpreted as queries applied on our container engine, and thus provide flexible associative access to files. indexing of key properties of file system objects and indexing/ caching on the file system is one of the fantastic feature of our system. The automatic indexing of files and grouped based on relativity is called "semantic" because user programmable nature of the system uses information about the semantics of updated file system objects to extract the properties for indexing. The semantic correlations and file groups identified in SANE can also be used to facilitate file perfecting and data de-duplication, among other system-level optimizations.*

***Keywords*****— map, reduce, data processing, transpose, minify**.

## INTRODUCTION

Fast and flexible metadata retrieving is a critical requirement in the next-generation data storage systems serving high-end computing. As the storage capacity is approaching Exabyte and the number of files stored is reaching billions, directory-tree based metadata management widely deployed in conventional file systems can no longer meet the requirements of scalability and functionality.

Although existing distributed database systems can work well in some real-world data-intensive applications, they are inefficient in very large-scale file systems due to four main reasons. First, as the storage system is scaling up rapidly, a very large-scale file system, the main concern of this paper, generally consists of thousands of server nodes, contains trillions of files, and reaches Exabyte-data-volume (EB). Unfortunately, existing distributed databases fail to achieve efficient management of petabytes of data and thousands of concurrent requests

## SCOPE

The scope of this project is to indexing the data in the container and Version management of the data in the container. Easy access of data in container

## SYSTEM DESIGN

System Design involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of Meta modeling that are helpful only after the construction of the class diagram. In the FUSION method some object-oriented approach likes Object Modeling Technique (OMT), Classes, and Responsibilities. Collaborators (CRC), etc, are used. Objector used the term ”agents” to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after the analyst creates the use case diagram. Once the design is over, it is essential to decide which software is suitable for the application

## IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system.

### MODULES DESCRIPITION:

1. Authentication Module
2. Container Creator Module
3. Data Uploader Module (Encryption Compression and Semantic storage)
4. Container Visualize Module
5. Index Tuner Module
6. Operations Detail View Module

## SYSTEM TESTING OBJECTIVES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## TYPES OF TESTS

- Unit testing
- Integration testing
- Functional test

## TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

## FEATURES TO BE TESTED

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## CONCLUSION

We presents a new paradigm for organizing file metadata for next-generation file systems, called SmartStore, by exploiting file semantic information to provide efficient and scalable complex queries while enhancing system scalability and functionality. The novelty of SmartStore lies in it matches actual data distribution and physical layout with their logical semantic correlation so that a complex query can be successfully served within one or a small number of storage units. Specifically, a semantic grouping method is proposed to effectively identify files that are correlated in their physical attributes or behavioral attributes. SmartStore can very efficiently support complex queries, which will likely become increasingly important in the next-generation file systems. Our prototype implementation proves that SmartStore is highly scalable, and can be deployed in a large-scale distributed storage system with a large number of storage units.

## REFERENCES

[1] R. N. Rodrigues, L. L. Ling, and V. Govindaraju, “Robustness of multimodal biometric fusion methods against spoof attacks,” J.Vis. Lang. Comput., vol. 20, no. 3, pp. 169–179, 2009.

[2] P. Johnson, B. Tan, and S. Schuckers, “Multimodal fusion vulnerability

to non-zero effort (spoof) imposters,” in IEEE Int’l Workshop on Inf. Forensics and Security, 2010, pp. 1–5.

[3] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, “Polymorphic blending attacks,” in Proc. 15th Conf. on USENIX Security Symp. CA, USA: USENIX Association, 2006.

[4] G. L. Wittel and S. F. Wu, “On attacking statistical spam filters,” in 1st Conf. on Email and Anti-Spam, CA, USA, 2004.

[5] D. Lowd and C. Meek, “Good word attacks on statistical spam filters,” in 2nd Conf. on Email and Anti-Spam, CA, USA, 2005.