Effective Semantic Namespace for File System

Priya N. Parkhi
Department of Computer Enginnering
St. Vincent Pallotti College of Enginnering
Nagpur,India

Vivek B. Kute
Department of Computer Enginnering
St. Vincent Pallotti College of Enginnering
Nagpur,India

ABSTRACT

Almost all today's file system namespace management is based on hierarchical tree Structure. As data volume increase this tree based namespace impose great challenges to effectively and efficiently manage data and also leads to performance bottlenecks. The basic idea is to build file's namespace by considering their semantic correlation and avoid brute-force search in entire system. The semantic correlations in file is used to facilitate scalability, search-ability, data de-duplication, file-prefecting and minimize extra overhead.

Keywords

File system, namespace management, semantic correlation, search-ability

1. INTRODUCTION

Today's file systems are facing great challenges in handling large volume of data because of many applications such as social network webs, cloud computing, business transactions, scientific computing and mobile applications. This data volume has imposed great challenges to storage systems, particularly to the metadata management of file. For example, many systems are required to perform thousands of metadata operations per second and the performance is totally restricted by the hierarchical directory-tree based metadata management scheme used in almost all file systems[1].

The most important functions of namespace management are file identification and lookup. File system namespace is responsible for improving system's quality of service such as performance, scalability, and ease of use. Unfortunately, almost all current file systems are based on hierarchical directory trees. This namespace design invented more than 40 years ago since has not been changed[2] .

Selecting appropriate attributes is important due to two challenging constraints, i.e., the curse of dimensionality and dimensionality heterogeneity. First, when the dimensionality exceeds , traversing the existing data structures becomes slower than the linear-scan approach. This slowdown phenomenon is often called the "curse of dimensionality". We hence need to reduce the dimensionality in order to decrease operational complexity. Second, dimensionality heterogeneity, which means that two items that are close-by in one space might be far away in another space with a different dimensionality, is another great challenge. The data correlation is sensitive to the observation space selected. Two items that are correlated when observed in one attribute subset might be totally uncorrelated in another attribute subset

1.1 NEED OF EFFECTIVE NAMESPACE FILESYSTEM

As the complexity and data volume keep increasing, conventional namespace schemes based on hierarchical directory trees have exposed some challenges are as follows

1.1.1 System Scalability

The directory-based management is effective only when similar documents or files have been stored in the same directory [1]. Although the directory size distribution has not significantly changed [3], the file system capacity has increased dramatically. This not only causes great inconvenience for file systems users, but also slows down applications by generating random accesses to underlying disks. In addition, since file lookups are always performed recursively starting from root directories, disks or servers have a highly unbalanced share of the workloads, leading to a higher probability of becoming performance bottlenecks.

1.1.2 Depends on end-users to organize and lookup data

Locating a target file by manually navigating the directories through directory trees in a large system is just like searching a needle in a haystack. As the directory tree increases it is equally difficult for users to instruct the file systems where a file should be stored and to find them quickly. When one does not know the full pathname of a file need exhaustive search over all directories. Such exhaustive search on a large system with billions of files takes a prohibitive amount of time. It is even more difficult to locate correlated files since users often cannot explicitly define mandatory search criteria in most file systems.

1.1.3 Lack of metadata-semantics exploration

While it is difficult to manage large volume of data through a centralized hierarchical structure. In most of industry and academic, a small subset of file system's data serves a majority of data access requests. Being able to identify frequently accessed data by semantic exploration is hence beneficial for system optimizations such as file prefetching and data deduplication. Conventional file systems have by and large ignored the semantic context in which a file is created and accessed during its lifetime.

2. PROJECT OBJECTIVE

The proposed namespace management scheme for search attribute, which provides a flat but manageable and efficient namespace. In this namespace, the notion of semantic-aware namespace is proposed in which a file is represented by considering its semantic correlations to other files, instead of conventional static file names. Our goal is not to totally replace conventional directory-treemanagement that already invented more than 40 years ago. Instead, we aim to provide another metadata that will work parallely to directory trees. This propose namespace runs concurrently with the conventional file system that integrates it and takes over the responsibilities of file search and semantic file grouping from the file system when necessary. Moreover, this namespace while providing the same functionalities makes use of a new naming scheme that only requires constant-scale complexity to identify and aggregate semantically correlated files. This namespace extracts the semantic correlation information from a hierarchical tree. Fig. 1.3.1 illustrates the relationship and difference between propose and the existing hierarchical directory tree namespace. For instance, in order to serve a complex query, new namespace only needs to check the small and flat namespace one time, thus avoiding a time-consuming search of brute-forced traversal over the entire hierarchical tree[4].

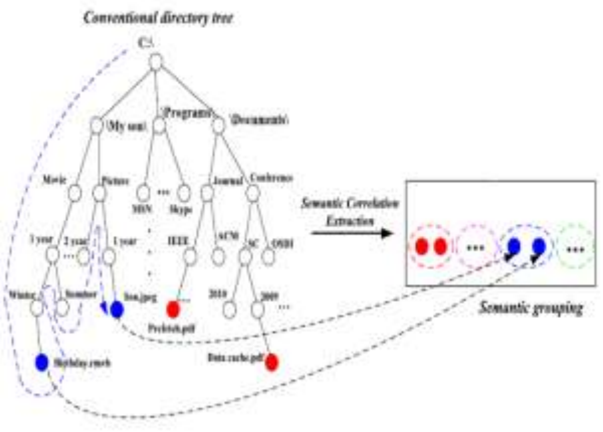

Fig 2 Semantic Correlation Extraction

Our goal in this research is to complement existing file systems and improve system search and delay. Major contributions in this project are summarized below

2.1 Scalability And Depends On End User

The propose namespace is designed to extract semantic correlations residing in multi-dimensional attributes instead of considering one-dimensional attributes such as pathnames, to represent a file. The metadata of files that are strongly correlated are aggregated together and then stored together in namespace. When a user performs a file lookup, propose namespace also present the user files that are strongly correlated to this searched file, which constitute the semantic-aware per-file namespace of this file. This allows the user to access the correlated files easily without performing additional searches otherwise directory tree navigations by considering all these things we made improvement in scalability.

2.2 SEMANTIC CORRELATION CONSIDERATION

Semantic namespace achieve by locality sensitive hashing (LSH) [5] which automatically organize semantically correlated files without the involvement of end-users or applications. This algorithm has very little performance overhead since LSH has a low complexity of probing constant-scale buckets. Semantic namespace represents each file based on its semantic correlations to other files. As the file system evolves, this namespace can efficiently identify their changes to update the namespace by exploiting the file semantics. The semantics in files are obtained from multiple dimensions, rather than a single one, thus also allowing us to optimize the overall system.

2.3 FILE-PREFETCHING

The propose namespace is implemented as a middleware that can be deployed/embedded in most existing file systems without modifying the kernels or applications. This research provides users with two auxiliary namespace views, i.e., default (conventional hierarchy) and customized (semantic correlated file representation). Both views hide the complex details of the physical representation of individual files, and export only a context-specific logical outlook of the data. Experimental results demonstrate that Semantic namespace efficiently supports query services for users, while facilitating system performance improvements, such as file prefetching and data de-duplication.

2.4 Minimize Extra Overheads And Delay Improvement

The propose namespace consider semantic correlations residing in multi-dimensional attributes to represent a namespace. The metadata of files which are strongly correlated are aggregated and then stored together in namespace. When a user performs a file lookup, this namespace allows the user to access the correlated files easily without having to perform additional searches over all directory trees. Thus this helps in minimize extra overhead and delay improvement.

3. DESIGN AND IMPLEMENTATION

3.1 Architectural Overview

The model gives idea about how efficient namespace created for search keywords. Here do prefetching of attributes of files from hierarchical tree structure i.e. filename, file contents, file access time. Enter the search keyword for which namespace created. Then identify semantic correlation i.e. similarity among files base on search keyword and prefetch attributes of files.Similarity of files calculates by considering normalization of files i.e. value of matching of file contents in between 0-1. After identification of similarity among files perform locality sensitive hashing on files. The basic idea here is to mapped the similar files into same bucket with high dimensionality. Construct the namespace for search keyword by fetching files which are strongly correlated or file who have maximum probability. This LSH also help for prefetching because it maintain namespace for that particular keyword search so next time when do searching for same keyword instead of going thorough all procedure just made look up in bucket.

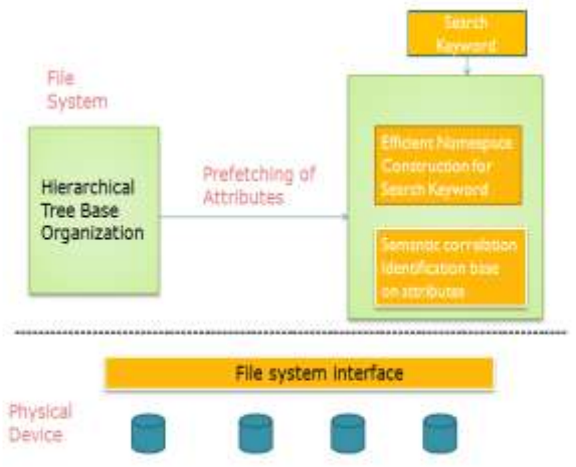
Search Keyword
File System
Hierarchical Tree Base Organization
Prefetching of Attributes
Efficient Namespace Construction for Search Keyword
Semantic correlation Identification base on attributes
File system interface
Physical Device

Fig 3 Architecture Propose Model

3.2 Prefetching Of File Attributes

Selecting appropriate attributes is important due to two challenging constraints

3.2.1 Curse Of Dimensionality

When the dimensionality exceeds at certain point then performance is decreases, this slowdown phenomenon is often called the "curse of dimensionality". Hence need to reduce the dimensionality in order to decrease operational complexity.

3.2.2 Dimensionality Heterogeneity

Dimensionality heterogeneity is another great challenge. Two items that are correlated when observed in one attribute subset might be totally uncorrelated in another attribute subset[6]. Select the particular directories where have to made search. Here we fetch four file attribute from hierarchical data structure.

1] File Name
2] File extension
3] File Contents
4] File Access time

3.3 Semantic Correlation Identification

Semantic correlation among the files are calculated on the basis of prefetch attributes and search keyword. Normalized each file with their attributes

SIM(Similarity)=$M_N{}' + M_C{}' + M_T{}'$ +other file attributes

$M_N{}'$ –Match with Name, Extension
$M_C{}'$ –Match with Contents
$M_T{}'$ –Match with Access Time

3.4 Apply Locality Sensitive Hashing

For the purposes of this section, consider functions that take two items an make decision about whether these items should be hash to same bucket or not. In many cases, the function f will "hash" items, and the decision will be based on whether or not the result is equal. Because it is convenient to use the notation f(x) = f(y) to mean that f(x, y) is "yes; put x and y into same bucket.

Let d1 and d2 be two distances where $0 \leq d1 < d2 \leq 1$ According to some distance measured.

A family F of functions is said to be (d1, d2, 1−d1, 1−d2)-sensitive if for every f in F:
1. If $d(x, y) \leq d1$, then the probability that f(x) = f(y) is at least (1-d1).

2. If $d(x, y) \geq d2$, then the probability that H(x) = H(y) is at most (1-d2)[4].

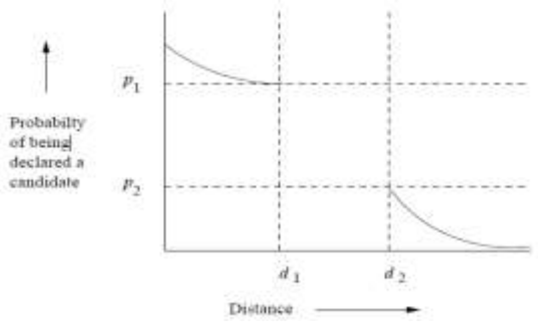

Fig 3.4 Behavior of a (d1, d2, p1, p2)-sensitive function

Example: We could let d1 = 0.3 and d2 = 0.6. Then we can assert that the family of functions is a (0.3, 0.6, 0.7, 0.4)-sensitive family. That is, if the distance/dissimilarity between x and y is at most 0.3 (i.e., SIM(x, y) ≥ 0.7) then there is at least a 0.7 chance that a minhash function will send x and y to the same bucket, and if the Dissimilarity between x and y is at least 0.6 (i.e., SIM(x, y) ≤ 0.4), then there is at most a 0.4 chance that x and y will be sent to the same bucket. Note that we could make the same assertion with another choice of d1 and d2; only d1 < d2 is required.

The semantic namespace of file f consists of t files (f1, f2….ft) that are the most strongly correlated with f based on p predefined semantics attributes, i.e., (a1, a2,a3,…ap). The correlation degrees, as a quantitative representation of semantic correlation, are (d1, d2….. dt) respectively.

The semantic-aware namespace of f is denoted by a t-tuple Namespace(f)={(f1,d1),(f2,d2)…….(ft,dt)}

4. EXPERIMENTAL RESULTS AND ANALYSIS

This project made conclusion is that propose design namespace is more effective in search and also it helps in delay improvement.

4.1 Effective Search

Construct the namespace for search keyword by fetching files which are strongly correlated to file who have maximum probability. It provides effective search result as compare to normal search result. Table show the search result by taking different keywords.

Table.4.1 Effective Search Result value

Normal Namespace	**Propose Namespace**
51	73
20	30
38	45
17	23
18	25
41	75
38	50
37	47
49	55
38	72

From Table No.4.1 draw the graph where x axis contain different search keyword and y axis contain number of effective search result. From the graph and table made conclusion that search is more efficient in propose namespace as compare to normal search

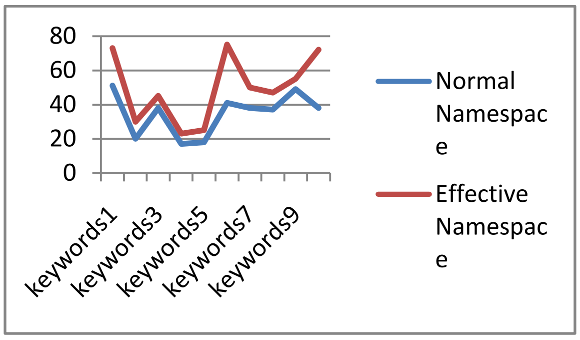

Fig.4.1 Search result of effective and Normal Namespace

4.2 Delay Improvement

When a user performs a file lookup, this namespace allows the user to access the correlated files easilywithout having to perform additional searches over all directory trees.

Table 4.2 Delay in millisecond for Normal and Effective search

Normal Namespace	Propose Namespace
1057	1000
1115	788
1194	935
1121	678
1164	745
1137	915
1239	720
928	714
1049	900
805	720

Thus this help in minimize extra overhead and delay improvement. Table show the delay result by taking different keywords in millisecond.

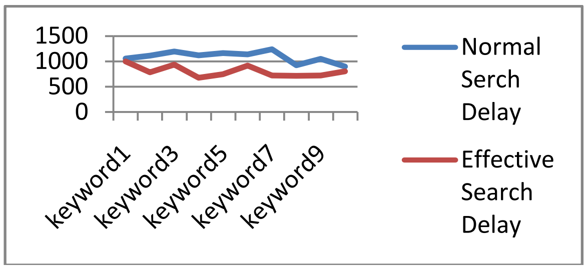

Fig 4.2 Delay result of effective and normal namespace

From Table 4.2 draw the graph where x axis contain different search keyword and y axis contain delay in millisecond. From the graph and table conclusion is that Propose system take less time as compare to normal search

5. CONCLUSION

There is a need of ease and efficiency of data access and this proposed system help to remove traditional file system's drawback.This work targets file attribute prefetching, semantic correlation identification among files, prefetching and namespace construction. Through experimental analysis we are able to shows effective result for search and reduction in delay as compare to normal search hence the propose system help in effective search and delay**.** In future, this project work can be extended for image documents and it can be release for public use. It will also be helpful in distributed environment.

6. REFERENCES

[1] DingQ. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search," Proc. VLDB, pp. 950-961, 2007.
[2] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," ACM SIGMOD Record, vol. 1, pp. 47-57, 1984.
[3] Yu Hua, Hong Jiang, Yifeng Zhu and Lei Xu," SANE: Semantic-Aware Namespace in Ultra-Large-Scale File Systems", VOL. 25, NO. 5, MAY 2014
[4] M. Seltzer and N. Murphy, "Hierarchical File Systems are Dead," Proc. 12th Conf. Hot Topics in Operating Systems (HotOS'09), 2009.
[5] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi- Probe LSH: Efficient Indexing for high-Dimensional SimilaritySearch," Proc. VLDB, pp. 950-961, 2007
[6] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of imensionality," Proc. 30th Ann. ACM Symp. Theory of Computing (STOC), 1998.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," Comm. The ACM, vol. 51, pp. 117-122, 2008.
[12] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," ACM SIGMOD Record, vol. 1, pp. 47-57, 1984.
[13] D.K. Gifford, P. Jouvelot, M.A. Sheldon, and J.W.O. Jr, "Semantic File Systems," Proc. Symp. Operating Systems Principle (SOSP), 1991.