Using Provenance to Boost the Metadata Prefetching in Distributed Storage Systems

Guojin Wu[1], Yuhui Deng[1]*, Xiao Qin[2]
[1]Department of Computer Science, Jinan University, GuangZhou, China
[2]Department of Computer Science and Software Engineering,
Auburn University, Auburn, Alabama, USA
Email: gj.wu@foxmail.com; tyhdeng@jnu.edu.cn*; xqin@auburn.edu

***Abstract*—Caching and prefetching are effective approaches to boosting the performance of metadata access in distributed storage systems. Many research efforts have been devoted in developing new metadata prefetching methods by considering past file access patterns. However, the existing methods do not consider the correlations between processes and the corresponding files(e.g. file provenance). Therefore, the methods cannot obtain very rich and accurate correlations, thus decreasing the effectiveness of metadata prefetching. This paper presents a Provenance-based Metadata Prefetching(ProMP) scheme, which considers both provenance and the past file access patterns. Through mining the correlations between processes and corresponding files from provenance and past access history, ProMP can achieve accurate and rich correlation information. ProMP is conducive to employing aggressive metadata prefetching to boost the performance by leveraging the correlations. Our experimental results show that ProMP performs more effectively with less memory overhead than the existing solutions, while improving the hit rates by up to 49% and 7% in contrast to traditional LRU and a state-of-art metadata prefetching algorithm Nexus, respectively.**

I. Introduction

Due to the explosive growth of data size, the storage capacity of large-scale file systems is approaching Exabytes and the number of objects stored is reaching billions [1]. Most current distributed file systems separate metadata from file data, thus diverting actual file data flows away from metadata flows. This architecture achieves excellent scalability and considerably alleviates I/O bottlenecks in the storage systems [2] [3] [4].

All I/O requests in distributed file systems are categorized into metadata requests and user data requests(see Fig.1). Conventionally, metadata is saved in one or multiple Metadata Servers(MDS), which are responsible for answering user metadata requests in terms of file attributes, accessing permissions, and data indexes. While the file data are always stored in the data nodes. Normally, the size of metadata is much smaller than that of file data, while the number of metadata requests accounts for over 50% of all the I/O requests [5]. With the dramatic growth of the file data and the number of clients, this feature poses huge workload pressure to the MDS. More often than not, the efficiency of MDS is a performance bottleneck of the entire distributed file systems.

Caching and prefetching are effective methods to improve the I/O performance due to the spatial locality and temporal locality of data layout. Prefetching the metadata and then caching them on the clients can effectively reduce the number of metadata requests scheduled to the MDS, thus resulting in an overall performance improvement of the I/O system [6]. While the prefetching and caching methods for the traditional file data are inadequate to prefetch and cache the metadata due to the following reasons:

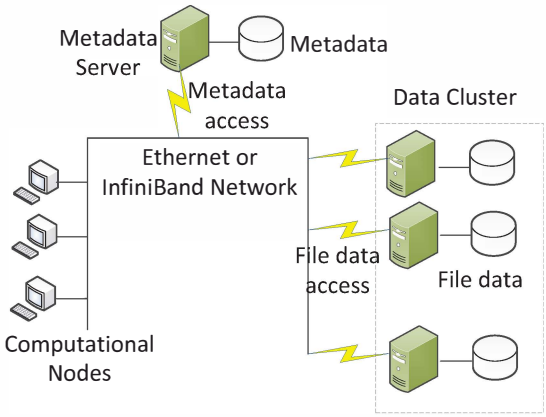


Fig. 1. System architecture of a typical distributed storage system

a) The size of traditional file data is much larger than that of metadata. Therefore, for a specific size of cache space, the number of cached file data is normally much less than that of metadata, and the cache policies designed for file data are pretty conservative. This is because when a cache missing occurs, the cached file data will incur a significant I/O penalty due to the large file data size. Assuming the block size of a file system is 4Kbytes, 65% of the metadata is less than 128 bytes and 98% of the metadata is less than 4224 bytes [7]. A cache miss of metadata will not occupy too much cache space, leading to a trivial I/O penalty. Thus, the prefetching strategy of the metadata prefers to prefetch a large number of metadata to fully utilize the cache space.

b) The semantic information contained in the metadata should be utilized to increase the precision of mining file access correlations [8]. Provenance is a type of metadata that is closely connected with the behavior of user processes. Such metadata describes where a data item comes from and why the item is found in

978-1-5090-5142-7/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

its current place [9]. Thus, the correlations between files and processes can be reflected by analyzing the provenance of data objects.

Many research efforts have been invested in developing new metadata prefetching methods by considering past file access patterns [7] [8] [10]. However, these methods do not consider the correlations between processes and the corresponding files(e.g. file provenance). This drawback makes the obtained correlation information not rich enough, thus decreasing the effectiveness of metadata prefetching. In this paper, we propose a Provenance-based Metadata Prefetching(ProMP) scheme to improve the caching hit rate by utilizing the characteristics of metadata. The ProMP analyzes the provenance of requested data, and then generates a Provenance Window(PW) with which file access correlations for the requested data within the same PW are computed. By utilizing the file access correlations, an effective and aggressive prefetching scheme for metadata is practically achieved.

The rest of the paper is organized as follows: The related work is introduced in Section II. The background knowledge is discussed in Section III. Section IV presents the detail design of the proposed ProMP scheme. The experimental results of evaluating the ProMP approach are summarized in Section V. Finally, Section VI concludes this paper.

II. Related Work

Provenance is a kind of metadata that represents the derivation of a data object, describing where the object comes from, and why it is found in its current place [9]. Provenance plays a significant role in scientific computing, archival systems, security systems, and etc [11].

There are many research efforts invested in collecting and applying provenance. Muniswamy-Reddy et al. [12] [13] presented a provenance-aware storage system, collecting and storing provenance. Concerning the huge amount of provenance data, many approaches were proposed to efficiently reduce the storage costs of provenance data [9] [14]. A provenance-based access control model was proposed to enhance the security of computer system [15]. By utilizing provenance, Liu et al. [16] efficiently improved the searching performance of for the cloud-based storage systems. However, provenance has never been used for analyzing file access correlations and prefetching metadata in distributed file systems.

Metadata prefetching and caching have been studied and implemented to render the performance benefit of metadata service in distributed file systems [6] [17]. However, the existing metadata prefetching algorithms mainly focus on using data mining technology to mine file access correlations according to past file access patterns.

AMP (Affinity-based Metadata Prefetching) [10] presented a n-gram language model and a data mining technology to leverage the past I/O request flow to prefetch groups of users' future metadata operations. In addition, Peng et al. [7] proposed a novel weighted-graph-based prefetching algorithm named Nexus. It leverages both the direct and indirect successor relationship to acquire performance benefits for group prefetching in cluster metadata servers. Compared to the conventional file prefetching algorithms, the above metadata prefetching algorithms achieve a high cache hit rate and exploit the benefits of aggressive prefetching. However, these approach only focus on analyzing the past file access patterns, without considering the abundant semantic attributes of metadata.

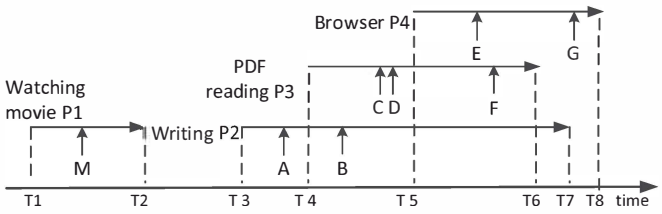


Fig. 2. An example of user operations on a computer

Little effort has been spent on the study of combining semantic attributes and past file access patterns. FARMER [8] was proposed to effectively mine and improve the precision of the file access correlation. However, FARMER just used file's static attributes to evaluate semantic distance between files, regardless of file's dynamic semantic attributes such as file's provenance.

In contrast to the existing work on metadata prefetching algorithms, this study incorporates the characteristics of provenance and metadata streams, we propose a ProMP algorithm to improve the performance of metadata accesses in distributed file systems.

III. Background

Based on the definition of provenance provided by OPM (Open Provenance Model) [18], provenance is composed of Process(P), Data(A) and Agent(Ag). Five relationships among these three components are as follow:

(1) *used*: process P accessed data A
(2) *wasGeneratedBy*: data A was generated by process P
(3) *wasControlledBy*: agent Ag was the master of process P
(4) *wasTriggeredBy*: process P1 was triggered by process P2
(5) *wasDerivedFrom*: data A2 was derived from data A1

Fig.2 shows a typical example demonstrating this relationship, where the characters T followed by a number on the X axis denotes a different time points (e.g. T1 indicates the first time point): process P1 is triggered and metadata M is accessed when a user is watching a movie. Then the user wants to write an article B, process P2 is triggered to read article A and copy A to a newly created article B. Metadata A and B are requested by process P2. Process P3 is generated by a PDF reader to reference related work, where metadata C, D, and F are requested by P3. Process P4 is triggered by opening a web browser to search the Internet. The process P4 requests metadata E and G. Therefore, a series of metadata M, A, B, C, D, E, F and G are requested by process P1∼P4.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

Base on the OPM model, the provenance generated by the user in Fig.2 can be described in the causal dependency graph (See Fig.3). Fig.3 shows that the files not only have correlations with each other (*wasDerivedFrom*), but also have relationships with the process (*used* or *wasGeneratedBy*). It demonstrates that there are correlations among the files accessed by the same process. For example, the metadata E and G are correlated to the same process P4. Besides, the processes with wasTriggeredBy relationships, such as (P2, P3, P4), can be aggregated as a single task. We define the life cycle of this task as a PW. We can determine that there are also correlations among the files accessed in the same PW, such as A, B, C, D, E, F, and G.

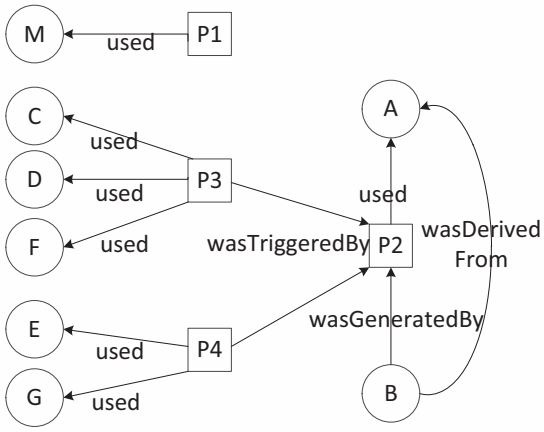


Fig. 3. Causal dependencies of user example

Two different PWs normally do not have or have very weak correlations, such as PW$[T1 \rightarrow T2]$ and PW$[T3 \rightarrow T8]$ (see Fig.2), which are triggered with different aims. Therefore, the I/O requests A, B, C, D, E, F and G within the same PW are strongly correlated, while requests within different PWs are weakly correlated. Thus, based on the relationships provided by provenance, we are able to mine the file access correlations among the files.

The aforementioned caching and prefetching approaches for metadata only leverage past file access patterns without considering the correlations between files and processes. Since uncorrelated I/O requests within different tasks are utilized for mining file correlations, the obtained file access correlations are not very accurate and rich. Consider the file access requests M, A, B, C, D, E, F and G in Fig.2, watching a movie and writing an article are two independent tasks. Therefore, employing the traditional approaches to compute the correlation degree between M and ABCDEFG can generate a lof of noise. To alleviate this situation, we assume that I/O requests within different tasks are uncorrelated. By extracting a PW for a specific task, uncorrelated I/O requests within different PWs can be filtered, thereby improving the prefetching accuracy.

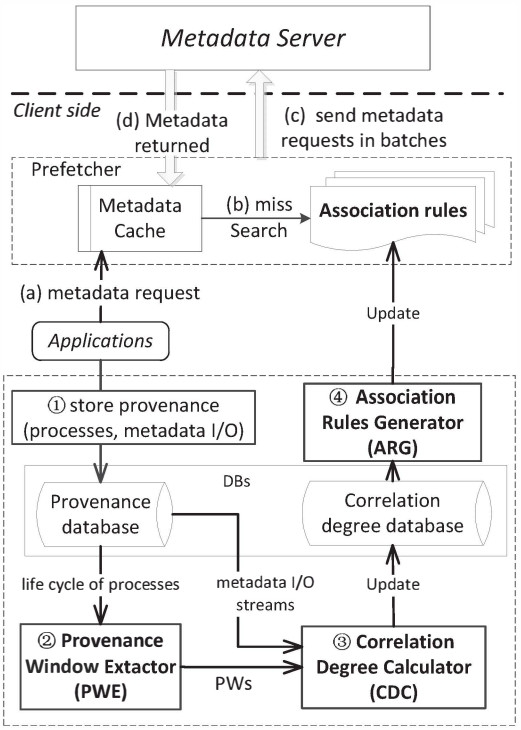


Fig. 4. Modules and flowchart of ProMP

IV. ProMP:A Provenance-based Metadata Prefetching Scheme

A. Overview

Fig. 4 demonstrates the flowchart and the main modules including a provenance-based correlation calculation and a Prefetcher module of ProMP. The module of provenance-based correlation calculation is illustrated in the dotted rectangle in the lower part of Fig. 4. This module has three important components including a Provenance Window Extractor(PWE), a Correlation Degree Calculator(CDC) and an Association Rules Generator(ARG). This module works with four important steps:

① The provenances (e.g. processes' life cycle and metadata I/O streams) collected are stored in a provenance database.
② PWE analyzes the collected provenance and extracts PWs periodically;
③ CDC employs the extracted PW and metadata access frequency to calculate the correlation degree.
④ ARG generates a list of association rules for metadata prefetching based on the computed correlation degree.

The Prefetcher module is composed of a metadata cache and generated association rules. The major steps of this module are labelled with sequence number as defined in the following descriptions. (a) When a metadata is requested by an application on a client side, it directly lookups the metadata cache; (b) If a cache miss occurs, the prefetcher selects the strongly correlated metadatas by searching the association rules; (c)

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

The client sends correlated metadata requests in batches to the metadata server; (d) Consequently, strongly correlated metadata objects obtained from the metadata server are then cached in the local cache of clients. This will reduce the overhead of network traffic incurred by the metadata requests.

B. Provenance Window Extractor

For simplicity of our presentation, we define a PW as follows:

Definition 1. The life cycle of a process P is defined as $L(P)$. It represents the time interval between the start time *st* and the ending time *et* of the process P:

$$L(P) = [st, et]. \tag{1}$$

Definition 2. If time $t \in L(P_a)$ and $t \in L(P_b)$, process P_a and P_b are correlated. We have $P_a \sim P_b$.

Definition 3. A task T is defined as a set of correlated processes. It is represented as $T = \{P_1, P_2, ..., P_n\}, n \geq 1$. T has three properties as follows:

a) The life cycle of task T is:

$$L(T) = \left[\min_{1 \leq i \leq n}(st_i), \max_{1 \leq i \leq n}(et_i)\right]. \tag{2}$$

b) $\forall t \in L(T)$, if $\exists P \in T$, then $t \in L(P)$.

c) For any two different tasks T_1 and T_2:

$$L(T_1) \cap L(T_2) = \emptyset. \tag{3}$$

Definition 4. The life cycle of task T is defined as a **PW**.

According to the disjointness of tasks (*property c*), there is a time interval between different tasks. Therefore, a PW can be extracted by using equation(2), if the list of life cycle of processes are given.

A problem incurred by this approach is that daemon-like processes are always kept alive. Due to the long life cycle of daemon-like processes, it is difficult to extract useful PWs. To solve this problem, we apply a valid threshold *max_strength* to filter out daemon-like processes. When the life cycle of a specific process is larger than *max_strength* , the process will be filtered out. For simplicity, *max_strength* is set as three hours in PWE algorithm.

According to the definition of PW, the algorithm of PW extractor is presented in Algorithm 1. The algorithm traverses the entire process list and sorts the processes in terms of the start time *st* of every process. Corelated processes are combined as a task T. When the next process is not correlated with the current task, a PW is generated. The PWE algorithm outputs a list of PWs by executing the sorting and traversal operations. In the worst case, the time complexity of this algorithm is O(nlogn) + O(n) = O(nlogn). It indicates an acceptable complexity. Based on the extracted PWs, ProMP computes the correlation degree of any two metadata objects in the next step.

Algorithm 1: extractor for provenance window

```
Input : max_strength,
        list of processes with start time and end time (st, et) PLIST
Output : list of provenance windows TLIST
1   //To sort PLIST by start time st
2   Sort(PLIST);
3   T = P1;
4   forall process P ∈ PLIST do
5       if P.et – T.st > max_strength then
6           TLIST.push(T);
7           T = <T.et, P.et>;
8       else if P∽T then
9           //correlated, combined
10          T.et = P.et;
11      else  // not correlated, generate a new provenance window
12          TLIST.push(T);
13          T = P;
14      end
15  end
16  return TLIST
```

C. Correlation Degree Calculator

We assume that all metadata accesses within the same PW have correlations [19]. However, there are different correlation strength between predecessors and successors in the metadata access streams in a chronological order. For example, metadata B and C are always accessed behind metadata A. The metadata *(A, B)* and *(A, C)* are strongly correlated, whereas the correlation strength of *(A, C)* is weaker than that of *(A, B)* in chronological order. To quantify the correlation strength between current metadata access and its successors, an attenuation function over time is adopted to assign a correlation degree of any two metadata objects.

Consider a series of metadata requests $Q_1, Q_2 ... Q_n$ following a request Q. First of all, the correlation degree is assigned with an initial value S_0. The value attenuates according to the time interval(Δt) between request Q and the subsequent request Q_k. When the recursive process reaches Q_k, the correlation degree will be updated as S_k. The recursive computation of the correlation degree S_k between Q and its following requests ends until S_k is decreased down below 0. The attenuation function over time of the correlation degree is shown in equation(4):

$$S_k = S_{k-1} - \lceil \Delta t \rceil, 1 \leq k \leq n. \tag{4}$$

We use an example (see Fig.5), to illustrate the basic idea of computing correlation degree for any two different metadata objects. The characters above the axis denote a metadata access stream A, B, C, D, E, F and G. Each number below the axis indicates the time point when a request arrives.

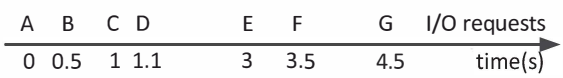


Fig. 5. An example of metadata requests in the provenance window

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

If S_0 is initialized as 10, according to equation(4), the correlation degrees between request A and its successors are computed as follows:

- $(A, B) = S_0 - \lceil 0.5 - 0 \rceil = 9 \rightarrow S_1$
- $(A, C) = S_1 - \lceil 1 - 0 \rceil = 8 \rightarrow S_2$
- $(A, D) = S_2 - \lceil 1.1 - 0 \rceil = 6 \rightarrow S_3$
- $(A, E) = S_3 - \lceil 3 - 0 \rceil = 3 \rightarrow S_4$
- $(A, F) = S_4 - \lceil 3.5 - 0 \rceil = -1 \rightarrow$ less than 0 and end

A → { B:9, C:8, D:6, E:3, F:-1 }
B → { C:9, D:8, E:5, F:2, G:-2 }
C → { D:9, E:7, F:4, G:0 }
D → { E:9, F:7, G:3 }
E → { F:9, G:7 }
F → { G:9 }
G → {∅}

Fig. 6. The correlation degree of a metadata access stream

	A	B	C	D	E	F	G
A	-	**9**	**8**	**6**	**3**	-	-
B	-	-	**9**	**8**	**5**	**2**	-
C	-		-	**9**	**7**	**4**	-
D	-	-	-	-	**9**	**7**	**3**
E	-	-	-	-	-	**9**	**7**
F	-	-	-	-	-	-	**9**
G	-	-	-	-	-	-	-

Fig. 7. Correlation degree of the PW

By analogy, the correlation degree of the metadata access stream A, B, C, D, E, F and G can be obtained. Fig.6 shows the correlation degree of the stream. Invalid correlation degrees which are smaller than 0 are underlined in the figure. The correlation strength grows with the increase of the numbers which represent the correlation degree of two different metadata objects. Therefore, a big number implies a more accurate correlation degree between different two metadata objects. Fig.7 summarizes the correlation degrees of the PW which holds the metadata access stream A, B, C, D, E, F and G.

D. Association Rules Generator

By using the provenance based CDC, we generate association rules by processing large-scale provenance and past metadata accesses. A computational paradigm for ProMP is driven by demands from processing a large number of data. Existing computational paradigms for big data can be categorized into a stream processing style and a batch processing style. Stream processing is a real-time processing on data stream straightly. It is more complicated with high memory overhead. As a kind of store-then-process, batch processing meets the requirement of non-real-time situations because it processes data off line and costs less memory resources. Considering the memory overhead and the properties of provenance, batch processing is chosen for generating association rules.

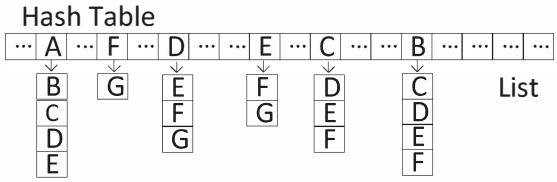


Fig. 8. Memory layout of association rules.

In our approach, all PWs and correlation degrees are stored in a key-value database, Tokyo Cabinet. An unique number *metanum* is assigned for every metadata object. Taking ($metanum_1$, $metanum_2$) as the primary key of correlation degree database, the value is the correlation degree between $metanum_1$ and $metanum_2$. For example, in Fig.7, the correlation degree of *(A, B)* is 9.

According to the correlation degree database, a strongly correlated *metanum* list (top-k highest correlation degree) of every metadata object is selected as association rules. Hash table is utilized to store the association rules. Take Fig.8 for example, *A* is represented as the *metanum* of metadata object A, the association rule of metadata object A is shown as the *metanum* list (BCDE). Association rules are fetched by the index of hash table when a cache miss of metadata request happens. And then a client sends metadata requests in batches to the metadata server according to the association rules. Consequently, strongly correlated metadata objects obtained from the metadata server are cached in the client's local cache. This caching method reduces the network traffic overhead of metadata requests.

V. Evaluation Methodology and Results

A. Workload Characteristics

We design a trace-driven simulator to evaluate the ProMP scheme by using three real-world traces. The modules and flowchart of the simulator are shown in Fig. 4. The characteristics of the three traces are presented in Table I. The Lasr trace was taken at the system-call level in 2000 and 2001 as a part of a security research project [20]. The Web trace was collected from a web server for an online library project [21]. The Res trace was collected on Hewlett-Packard series 700 workstations running HP-UX 9.05 that stored the research projects of graduate students [21].

As all the three traces were collected at the system-call level, a request from a trace can be represented as a tuple (timestamp, system call name, requested object). However, file data activities and metadata events are collected in the three traces. In our simulations, the system calls about file data requests including read and write are filtered out and only system calls about metadata requests are fed to our simulator. For example, the reserved system calls and the number of the corresponding requests of the Web trace are summarized in Table II .

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

TABLE I
CHARACTERISTICS OF THREE DATA SET

dataset	Lasr	Web	Res
usage	user data	Web Server	research project
# of metadata	482403	299028	446871
# of ops	41819957	5643177	23930834
period(days)	116	8	15

TABLE II
LIST OF OPERATIONS IN WEB TRACE

system call	# of ops	description
open	3384521	open a file
chmod	19243	modify file's mode
chown	303	change file's owner
utime	968	change access time
access	16061	check user's access permissions
stat	1442501	get file's attr
lstat	751096	get file's attr
fstat	3677	get file's attr
fchmod	60	modify file's mode

All the three traces contain not only I/O system calls but also system calls about the behavior of processes like *exit*. In our simulations, the timestamp of a process's first I/O operation is used as the start time(*st*) of the process, whereas the timestamp when the system call *exit* is finished is utilized as the ending time(*et*). Therefore, PWs can be extracted in these traces by using Algorithm 1 with aforementioned life cycle of processes(st, et).

At very beginning, initial correlation degrees are computed by training data requests sizing two days. The new provenance data (metadata requests and life cycle of processes) are stored in the key-value database. Periodically, PWs are extracted by analyzing the new provenance. Based on the extracted PWs, the correlation degrees are computed and updated to the correlation degree database.

In order to evaluate the impact of the cache size on the algorithm performance, we assume that the metadata are equal in size; then we compare the performance of ProMP under different number of metadatas, which represent the size of the cache. In our simulations, we test five different cache sizes.

B. Impact of Initial Correlation Degree S_0

As equation(4) presents, the initial correlation degree S_0 is a critical factor to determine how many successors will be computed. A bigger S_0 indicates more successors of a request have to be computed. This accounts for a more accurate correlation degree. To obtain an optimal S_0, we compare the cache hit rates by performing a number of experiments with different S_0 (5,10,15,20,25,30). The impact of different S_0 on the cache hit rates with different traces are presented in Fig.9.

Due to the strong locality contained in Res trace, Fig.9(c) shows that S_0 has a negligible impact on the hit rate. However, Fig.9(a) and Fig.9(b) suggest that the larger the S_0 is, the higher the hit rate achieves. The hit rates keep increasing when $S_0 \leq 15$. However, the hit rates remain almost unchanged when $S_0 > 15$. According to the computation costs and hit rates we obtained from the experiments, ProMP chooses 15 as the initial S_0.

C. Impact of Prefetching Size

To fully reap huge fruits of group prefetching, we need to determine the number of metadata to prefetch. We compare the cache hit rates by performing a number of experiments on different prefetching size. We choose 2, 4, 6, 8, 10, 12 and 14 metadatas as different prefetching size. The impact with Lasr, Web and Res traces of different prefetching size are presented in Fig.10. The hit rates are increasing when the prefetching size is smaller than 8. However, the hit rates remain almost unchanged even decreasing when the prefetching size is greater than 8. Taking computation cost and hit rates into account, ProMP chooses 8 as the prefetching size.

D. Evaluating Hit Rate

The overall cache hit rate of metadata in distributed file systems consists of three elements: local hit rate at client side, hit rate at metadata server and cooperative cache hit rate [7]. Apparently, local hit rate directly reflects the performance of the metadata prefetching algorithm because the client side executes the prefetching algorithm. In our experiment, client local hit rate is chosen as the main criteria for evaluating the performance of a prefetching algorithm.

Trace-driven simulations based on aforementioned Lasr trace, Web trace and Res trace were conducted to compare different prefetching algorithms. We compare ProMP with the traditional Least Recently Used(LRU) which is the most commonly-used caching algorithm, and a state-of-the-art metadata prefetching algorithm Nexus in terms of cache hit rate.

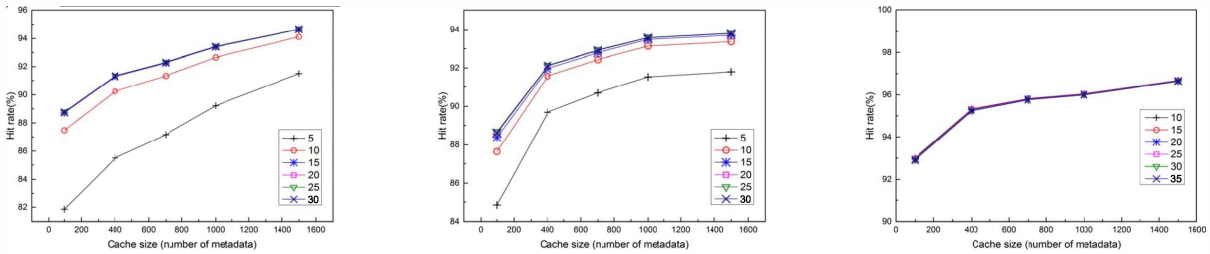
Figs.11(a) ,11(b) and 11(c) show the local hit rates when using Lasr trace, Web trace and Res trace, respectively. In both Figs.11(a) and 11(b), ProMP achieves better performance on the local cache hit rates than that of other two algorithms with Lasr trace and Web trace. On average, ProMP can achieve up to 49 percent higher local hit rate than that of LRU. In addition, the local hit rate of ProMP outperforms Nexus up to 7%. From Fig.11(c), we observe that the local cache hit rate of LRU is higher than 90% and Nexus's hit rate is reaching 95%, which is the reason why there is no room for ProMP to further improve the local hit rate.

Fig.11(a) shows that ProMP achieves an excellent local cache hit rate with a small cache size when using Lasr trace. For example, with the cache size of 100 metadatas, ProMP achieves the same cache hit rate as that of Nexus with a cache size of 1500 metadatas. This is because PW is utilized by ProMP to isolate different tasks, which reduces introducing noise into the correlation degree computing. As a result, Promp achieves a high prediction accuracy to prefetch metadata which in turn improves the cache utilization.

E. Evaluating Memory Overhead

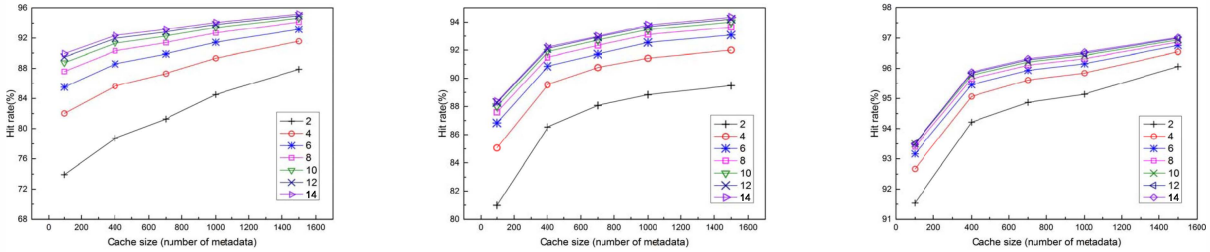
Because ProMP adopts batch-processing paradigm offline, provenance and the correlation degrees are stored in a key-value database, and the association rules generated are updated

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.



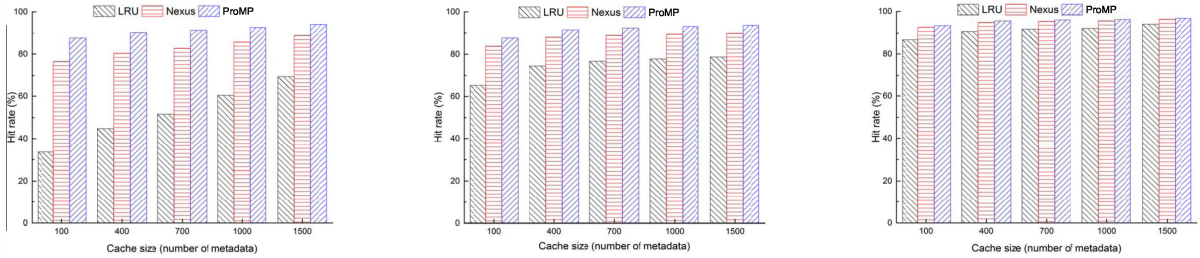
(a) Lasr trace (b) Web trace (c) Res trace

Fig. 9. The influence of S_0.



(a) Lasr trace (b) Web trace (c) Res trace

Fig. 10. The impact of prefetching size.



(a) Lasr trace (b) Web trace (c) Res trace

Fig. 11. Comparison of cache hit rate.

in memory for querying. Accordingly, the space overhead of ProMP is composed of the key-value database stored in disk and the association rules stored in memory. Compared to hard disks, main memory is small and expensive [22]. As a critical factor affecting server performance, memory overhead is considered as another criteria for the implementation of these metadata prefetching algorithms.

Relationship graph of Nexus [7] was generated by processing metadata access streams and stored in a weighted directed graph based on an adjacent matrix. However, the amount of data objects stored in current large-scale storage systems are reaching billions, which leads to an unacceptable memory overhead costed by adjacent matrix graph. In our experiments, the adjacent matrix is replaced by a spare matrix to store the relationship of Nexus algorithm, which results in the lowest memory overhead. Therefore, the memory overhead of Nexus obtained in our experiment is an optimized one. Fig.12 shows the memory overhead incurred when using the Lasr trace, Web trace and Res trace. As expected, ProMP achieves a much better storage efficiency than that of Nexus on all traces. In particular, the memory overhead of ProMP is about half of that of Nexus with Web trace.

F. Evaluating Querying Performance of Association Rules

Association rules are fetched frequently when a cache miss happens. Unlike conventional prefetching algorithms for file data, the metadata prefetching algorithms aggressively fetch a dozen of metadata so that the association rules are frequently queried. Therefore, the performance of querying association rules is considered as another criteria for the implementation of these metadata prefetching algorithms. As a stream processing paradigm, Nexus immediately updates the relationship graph. Nexus has to traverse all out-degrees to find out top-k strongly correlated objects. On the contrary, ProMP generates the

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.

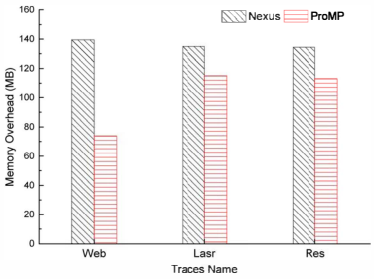


Fig. 12. Memory overhead of Nexus and ProMP.

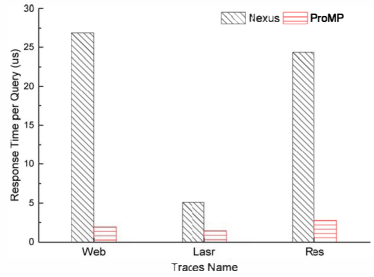


Fig. 13. Time overhead of querying association rules.

association rules offline and it directly fetches the association rules with fast hash lookup. As expected, ProMP has the best querying performance in terms of Fig.13.

VI. Conclusion

In this study, we developed the ProMP Scheme for distributed file systems. A trace-driven simulator was designed to evaluate the effectiveness of ProMP. PW is proposed to filter out the noise incurred by mining correlation when using the traditional approaches. The experimental results demonstrate that by exploiting PW and the past file access patterns, ProMP can mine the correlations between processes and the corresponding files, thus achieving the aggressive prefetching. Additionally, the effectiveness of ProMP grows with the increase of cache size. Furthermore, the proposed algorithm performs more effectively with less memory overhead than the state-of-the-art metadata prefetching algorithm Nexus. The experimental results show that ProMP performs more effectively with less memory overhead than the existing schemes, while improving the hit rates by up to 49% and 7% in contrast to traditional LRU and a state-of-art metadata prefetching algorithm Nexus, respectively.

Acknowledgment

This work is sponsored in part by the National Natural Science Foundation of China under Grant No. 61572232 and No.61272073, key project of Natural Science Foundation of Guangdong Province under Grant No.S2013020012865. The corresponding author of this paper is Yuhui Deng.

References

[1] C. L. Abad, H. Luu, N. Roberts, K. Lee, Y. Lu, and R. H. Campbell, "Metadata traces and workload models for evaluating big storage systems," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2012, pp. 125–132.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIG●PS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[3] P. J. Braam *et al.*, "The lustre storage architecture," 2004.

[4] D. Borthakur, "Hdfs architecture guide," *HAD●●P APACHE PR●JECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, 2008.

[5] 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," in *Proceedings of the conference on high performance computing networking, storage and analysis*. ACM, 2009, p. 10.

[6] B. Zhang, B. Ross, S. Tripathi, S. Batra, and T. Kosar, "Network-aware data caching and prefetching for cloud-hosted metadata retrieval," in *Proceedings of the Third International Workshop on Network-Aware Data Management*. ACM, 2013, p. 4.

[7] P. Gu, J. Wang, Y. Zhu, H. Jiang, and P. Shang, "A novel weighted-graph-based grouping algorithm for metadata prefetching," *Computers, IEEE Transactions on*, vol. 59, no. l, pp. 1–15, 2010.

[8] P. Xia, D. Feng, H. Jiang, L. Tian, and F. Wang, "Farmer: a novel approach to file access correlation mining and evaluation reference model for optimizing peta-scale file system performance," in *Proceedings of the 17th international symposium on High performance distributed computing*. ACM, 2008, pp. 185–196.

[9] A. P. Chapman, H. V. Jagadish, and P. Ramanan, "Efficient provenance storage," in *Proceedings of the 2008 ACM SIGM●D international conference on Management of data*. ACM, 2008, pp. 993–1006.

[10] L. Lin, X. Li, H. Jiang, Y. Zhu, and L. Tian, "Amp: an affinity-based metadata prefetching scheme in large-scale distributed storage systems," in *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*. IEEE, 2008, pp. 459–466.

[11] K.-K. Muniswamy-Reddy and M. Seltzer, "Provenance as first class cloud data," *ACM SIG●PS ●perating Systems Review*, vol. 43, no. 4, pp. 11–16, 2010.

[12] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems." in *USENIX Annual Technical Conference, General Track*, 2006, pp. 43–56.

[13] K.-K. Muniswamy-Reddy, P. Macko, and M. I. Seltzer, "Provenance for the cloud." in *FAST*, vol. 10, 2010, pp. 15–14.

[14] Y. Xie, K.-K. Muniswamy-Reddy, D. Feng, Y. Li, and D. D. Long, "Evaluation of a hybrid approach for efficient provenance storage," *ACM Transactions on Storage (T●S)*, vol. 9, no. 4, p. 14, 2013.

[15] J. Park, D. Nguyen, and R. Sandhu, "A provenance-based access control model," in *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*. IEEE, 2012, pp. 137–144.

[16] J. Liu, D. Feng, Y. Hua, B. Peng, and Z. Nie, "Using provenance to efficiently improve metadata searching performance in storage systems," *Future Generation Computer Systems*, vol. 50, pp. 99–110, 2015.

[17] J. Liu, J. Chen, W. Tong, L. Tian, and C. Chen, "Design and implementation of metadata cache management strategy for the distributed file system," *International Journal of Grid & Distributed Computing*, vol. 7, no. 3, 2014.

[18] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers *et al.*, "The open provenance model core specification (v1. 1)," *Future generation computer systems*, vol. 27, no. 6, pp. 743–756, 2011.

[19] S. Somogyi, T. F. Wenisch, A. Ailamaki, and B. Falsafi, "Spatio-temporal memory streaming," *Acm Sigarch Computer Architecture News*, vol. 37, no. 3, pp. 69–80, 2009.

[20] S. I. Repository. (2016) Lasr traces. [Online]. Available: http://iotta.snia.org/traces/2

[21] D. S. Roselli, J. R. Lorch, T. E. Anderson *et al.*, "A comparison of file system workloads." in *USENIX annual technical conference, general track*, 2000, pp. 41–54.

[22] Y. Deng, "What is the future of disk drives, death or rebirth?" *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 23, 2011.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:02:24 UTC from IEEE Xplore. Restrictions apply.