Using Data Mining for Discovering Patterns in Autonomic Storage Systems

Zhenmin Li, Sudarshan M. Srinivasan, Zhifeng Chen, Yuanyuan Zhou,
Peter Tzvetkov, Xifeng Yan, and Jiawei Han

Department of Computer Science, University of Illinois at Urbana-Champaign

ABSTRACT

In order to be self-tuning, self-managing, self-healing and self-protecting, a storage system needs to be able to automatically characterize access patterns. This paper proposes an approach that uses data mining techniques to systematically mine access sequences in a storage system to characterize storage behaviors. More specifically, we use frequent sequence mining algorithms to find block access correlations which can be used to improve the effectiveness of subsystems such as storage caching and disk scheduling, and for disk power management. This paper reports our preliminary results of discovering block correlations from storage access sequences using a recently proposed data mining algorithm called CloSpan.

1. INTRODUCTION

Enterprise-scale storage systems are becoming increasingly complex, and the cost of maintaining these systems is a significant part of the *total cost of ownership (TCO)*. Managing and tuning such systems according to workload characteristics for providing high performance, reliability, availability and scalability demanded by enterprise customers is a non-trivial task. To maintain a storage system, skilled administrator involvement is required, and the tasks are time-consuming, effort-consuming and error-prone.

To address this problem, it is desirable to build storage systems that can be self-managing, self-tuning and self-protecting. To provide these autonomic capabilities, a system needs to automatically capture and characterize the existing storage access behavior based on which it can change its control policies or configurations to adapt to application workloads. However, without proper analysis tools, these behaviors such as temporal and spatial locality, block correlations and access frequencies are difficult to characterize, especially for workloads whose behaviors change dynamically from one time period to another.

Statistical methods have previously been used in storage access pattern analysis [2]. These techniques are very useful for capturing some simple system behavior. Several recent studies, such as [1] have examined the I/O patterns for commercial applications or desktop environments. Researchers have attempted to characterize different aspects of application I/O, such as burstiness, inter-arrival time and location in order to generate meaningful traces that reflect access patterns [3, 4].

However, these methods cannot detect complex access behaviors such as correlations of blocks that are not contiguous in disks. Block correlations occur commonly in many applications, but their presence is not evident at the level of the storage system because the system only exports block interfaces. It is difficult to extend the block I/O interface to allow upper levels to inform a storage system about block correlations that may be application-specific, such as the correlation between the access of an inode block of a file and the corresponding data blocks. Exploring these correlations is very useful for improving the effectiveness of storage caching, prefetching, data layout, disk scheduling, etc [6].

This paper investigates an approach that uses data mining techniques to systematically mine access sequences in a storage system to characterize storage access patterns which the system can use for configuring itself. More specifically, we use frequent sequence mining algorithms to find correlations among blocks. In this paper, we report our preliminary results of using a data mining algorithm called CloSpan [7] to find block correlations among several storage traces.

2. ARCHITECTURE

Figure 1 shows an example architecture of an autonomic storage system with self-tuning and self-managing capabilities.

The *system monitor* collects access information and passes it to the data mining engine. The *data mining engine* runs in the background on the same or different machine and generates access patterns. These patterns are stored in a pattern repository and read by different modules in the storage system to adapt their policies or algorithms using the patterns to improve performance.

The data mining engine is responsible for maintaining the repository; it inserts new patterns and purges obsolete ones. It also maintains the age of each pattern, allowing newer patterns to carry more weight than old ones. The data mining engine can use several algorithms, each performing different types of analysis.

The core component is the storage system, which is comprised of a number of subsystems. Some subsystems, such as the cache subsystem, data layout module and the disk scheduling module can be tuned by modifying parameters according to the workload to improve system performance. To reduce overheads for accessing the pattern repository, many frequently and recently used patterns can be stored in a memory buffer associated with each module. This buffer is called *pattern cache*.

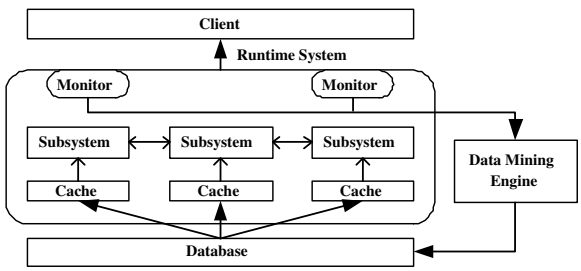


Figure 1: An example architecture

3. BLOCK CORRELATIONS

Block correlations commonly exist in many storage system workloads. Block a and b are correlated if after a block a is accessed, block b is very likely to be accessed in a near future. The spatial locality is a simple case of block correlations. In a complex block correlation, the correlated blocks may not be contiguous in locations. For example, in a B-tree layout that is commonly used by databases, a node is correlated to its parent node or an ancestor node. But this node may not be allocated in disk close to the parent node or the ancestor nodes. In a file system, a file block is correlated to its inode. But in most of file system layout, file blocks are allocated separately from inode blocks.

Exploiting these block correlations can improve the storage system performance. For example, if a strong correlation exists among blocks a, b, c, these blocks can be fetched together from disks whenever one of them is accessed. The disk read-ahead optimization is an example of exploring the simple sequential block correlations by prefetching subsequent disk blocks ahead of time. Several studies [2, 5] have shown that exploring even these simple correlations can significantly improve the storage system performance.

A challenging question is how to explore more complex correlations such as tree-like accesses or striped accesses that are commonly exhibited in databases or file system metadata. We propose an approach that uses data mining to capture block correlations as an example to show how data mining can be used in the architecture described in Section 2.

Frequent sequence mining can be used to discover block correlations in a storage access sequence. In our experiment, we use a recently proposed frequent sequence mining algorithm called **CloSpan** (Closed Sequential Pattern mining) [7].

4. EXPERIMENTAL RESULTS

To show the effectiveness of using CloSpan for discovering block correlations, we conduct experiments on two types of I/O traces: the synthetic traces including loop-sequential trace, loop-striped trace, and B-tree trace, and the real system traces including HP Cello trace and MS-SQL TPC-C trace. The goal is to evaluate whether the CloSpan algorithm can capture interesting block correlations, and how efficient it is. Due to the space limitation, we only show the results for loop-striped and Cello traces.

(A) Loop-striped (10^6 requests)

min_sup	#patterns	time(sec)
50	10195	21.250
100	3302	5.063
200	25	0.234

(B) Cello (6.6×10^6 requests)

min_sup	#patterns	time(sec)
50	18840	39.343
200	2863	4.578
1000	399	0.844

Table 1: Overall results

Table 1 shows the number of patterns and running time of CloSpan with different minimum support constraints. The results demonstrate that data mining algorithms such as CloSpan are efficient and practical for analyzing the I/O traces. For example, it takes about 5 seconds to discover 2863 patterns with support greater than 200 from the Cello trace that contains a full-day's disk requests.

Figure 2 shows the block access correlations. We plot the patterns with length of 2. The x-axis represents the prefix of a pattern, and the y-axis represents the suffix. Any point (x, y) indicates a correlation between blocks x and y, that is, if block x is accessed, then block y is likely to be accessed very soon.

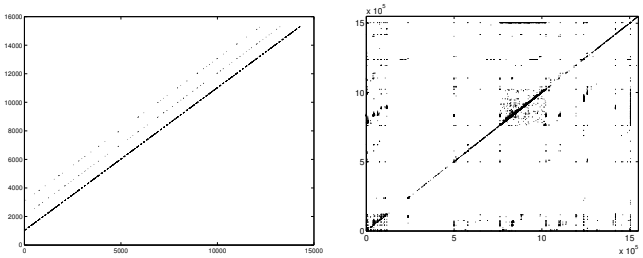
Various types of patterns can be figured out from these graphs:

Temporal locality can be represented by patterns $\{x, x\}$, which means that if x is accessed, it will be accessed again soon. Therefore, the points standing showing temporal locality are located on the diagonal line $y = x$ in the graphs as shown in figure 2(b).

Spatial locality can be represented by patterns $\{x, x+k\}$ where k is a small number, which means that if x is accessed, its neighbor blocks are likely to be accessed soon. Hence, the points standing showing spatial locality are located around the diagonal line $y = x$ in the graphs as shown in figure 2(b).

Striped access patterns have a fixed striped distance a, so they can be represented by $\{x, x + a\}$ where a is a constant number. Hence, the points showing striped access patterns constitute a line $y = x + a$ as shown in figure 2(a).

Some **other more complex patterns** are very common and useful, such as B-Tree layout access pattern, and the correlation between metadata (e.g, inode) and user data (e.g, file blocks). Figure 2(b) contains many points that are distributed over the whole block address space, which demonstrates that data mining algorithms can find such complex patterns.



(a) *Loop-striped trace* (b) *Cello trace*

Figure 2: Block access correlations

5. CONCLUSIONS

This paper proposes a novel approach that uses data mining techniques to systematically mine access sequence in a storage system to characterize storage behaviors. More specifically, we use frequent sequence mining algorithms to find correlations among blocks. We have reported our preliminary results of using a recently proposed data mining algorithm called CloSpan to find block correlations from storage access sequences.

6. REFERENCES

[1] G. Alvarez, K. Keeton, E. Riedel, and M. Uysal. Characterizing data-intensive workloads on modern disk arrays. In *Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW'01)*, January 2001.

[2] J. Choi, S. H. Noh, S. L. Min, and Y. Cho. Towards application/file-level characterization of block references: a case for fine-grained buffer management. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2000.

[3] M. Gomez and V. Santonja. Self-similarity in I/O workload: Analysis and modeling. In *Workshop on Workload Characterization*, 1998.

[4] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L. Miller. Self-similarity in file systems. In *Proc. ACM SIGMETRICS'98*, pages 141–150, Madison, USA, June 1998.

[5] J. Kim, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, and C. Kim. A low-overhead high-performance unified buffer management scheme that exploits sequential and looping references. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 119–134, San Diego, CA, Oct. 2000.

[6] M. Sivathanu, V. Prabhakaran, F. Popovici, T. E. Denehy, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Semantically-smart disk systems. In *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST'03)*, pages 73–88, San Fransisco, CA, March 2003.

[7] X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In *Proc. 2003 SIAM Int. Conf. Data Mining (SDM'03)*, San Fransisco, CA, May 2003.