Multi-Layer Prefetching for Hybrid Storage Systems: Algorithms, Models, and Evaluations

Mais Nijim[1], Ziliang Zong[2], Xiao Qin[3], Yousef Nijim[4]

1. Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville. TX, USA.
2. Department of Mathematics and Computer Science, South Dakota School of Mines & Technology, Rapid City, SD, USA.
3. Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA.
4. IEEE Senior Member, Cox Communications, Atlanta, GA, USA.

Abstract: **Parallel storage systems have been highly scalable and widely used in support of data-intensive applications. In future systems with the nature of massive data processing and storing, hybrid storage systems opt for a solution to fulfill a variety of demands such as large storage capacity, high I/O performance and low cost. Hybrid storage systems (HSS) contain both high-end storage components (e.g. solid-state disks and hard disk drives) to guarantee performance, and low-end storage components (e.g. tapes) to reduce cost. In HSS, transferring data back and forth among solid-state disks (SSDs), hard disk drives (HDDs), and tapes plays a critical role in achieving high I/O performance. Prefetching is a promising solution to reduce the latency of data transferring in HSS. However, prefetching in the context of HSS is technically challenging due to an interesting dilemma: aggressive prefetching is required to efficiently reduce I/O latency, whereas overaggressive prefetching may waste I/O bandwidth by transferring useless data from HDDs to SSDs or from tapes to HDDs. To address this problem, we propose a multi-layer prefetching algorithm that can judiciously prefetch data from tapes to HDDs and from HDDs to SSDs. To evaluate our algorithm, we develop an analytical model and the experimental results reveal that our prefetching algorithm improves the performance in hybrid storage systems.**

Keywords: hybrid storage systems; solid-state disks; tape storage systems; pre-fetching

I. INTRODUCTION

A recent study shows that new data is growing annually at the rate of 30% and five exabytes (5×260) of new information were generated in 2002 [6]. Major corporations, supercomputing centers and rich media organizations, including Lawrence Livermore National Laboratories, Oak Ridge National Laboratory, NASA Ames, Google, Boeing, and CNN rely on large-scale storage systems to meet demanding requirements of large data capacity with high performance and reliability [7]. Large-scale storage systems have to be developed to fulfill rapidly increasing demands on both large storage capacity and high I/O performance [8][9]. Traditionally, the storage capacity and I/O performance of a system are scaled up by simply employing more HDDs [10][11]. However, I/O performance and capacity improvements through the increased number of HDDs are an expensive solution due to huge expenses on new storage equipments and the increased maintenance fee. Hybrid storage systems (HSS) – containing SSDs, HDDs, and tapes subsystems− can provide an ideal data storage solution to significantly improve storage capacity and I/O performance at low cost.

Large-scale hybrid storage systems will become increasingly popular in the next few years for the following two reasons. First, HSS will keep its high performance by prefetching and caching highly accessed storage objects in high-speed storage components such as SSD or HDD. Second, hybrid storage systems are cost-effective, because inexpensive tapes help in increasing storage capacities at very low cost. Therefore, it is believed that hybrid storage systems, which have high performance, long archive life, and low cost, are ideal data storage platforms for a wide variety of data-intensive applications from human genome analysis [1] to remote-sensing applications [3]; from long running scientific simulations [4] to biological sequence analysis [2]. Hybrid storage systems are practically feasible because SSDs, as one of the newly developed storage components, can be easily connected to any other types of storage devices [5].

Highly efficient data transfer between tapes, HDDs and SSDs is important to hybrid storage systems. For a wide range of data-intensive applications, critical data are required to periodically or continuously backed up to tapes so that data restoration is possible in case of system crash or data loss. It is imperative to minimize data restore time in order to improve data I/O performance, which largely depends on high data transfer rate between disks and tapes. Thus, transferring data back and forth among SSDs, HDDs, and tapes plays a critical role in achieving high I/O performance.

Prefetching is a promising solution to reduce the latency of data transferring among SSDs, HDDs, and tapes. Prefetching is a process that aims at reducing the number of requests issued to HDDs or tapes by caching popular data in SSDs. Prefetching can be used to prevent I/O bandwidth underutilization by fully exploiting idle times of storage components to hide I/O latency. Prefetching in the context of hybrid storage systems is technically challenging due to an interesting dilemma: aggressive prefetching is needed to efficiently reduce I/O latency, whereas overaggressive prefetching may waste I/O bandwidth by transferring useless data from HDDs to SSDs or from tapes to HDDs. To overcome these technical obstacles, we investigate a multilayer prefetching algorithm (PreHySys) to enhance I/O performance of hybrid storage systems.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the architecture of the hybrid storage system. In section 4, we present the multi-layer prefetching module. Section 5 describes the analytical model of the multi-layer prefetching. The conclusion of the paper is discussed in section 6.

II. Related Work

Integrating Hard Disks with Tapes: Tape-based storage systems are essential archival storage components required by high-performance computing communities. In the past two decades, magnetic tapes have been considered the most cost effective and reliable way to archive and backup data for a long period of time. Although tape storage systems offer high capacity at low cost, tapes have a performance drawback – high I/O latencies and lower bandwidth due to sequential data accesses. Conventionally, the performance of tape-based storage systems can be improved by integrating hard disks with tapes [27]. Hard disks (e.g, RAID – redundant array of independent disks) can be employed in tape storage systems to store frequently used backup data. To reduce data transfer time between disks and tapes, data striping ideas are applied to increase I/O throughput and reduce response time [12].

Solid-State Disks: Solid State Disks are made of semiconductor memory devices. Currently, there are two types of SSDs: flash memory based SSDs and DRAM based SSDs. Both flash memory based and DRAM based SSDs can be integrated with HDDs and tapes to increase I/O performance of large-scale hybrid storage systems. Recently, flash memory based SSDs are becoming very popular for data-intensive computing because of the advantages inherited from flash memory such as high density and low power properties. There has been some work on applying caches to boosting performance of parallel disk systems. Kotz and Ellis investigated cache management techniques used in parallel file systems to close the gap between processor and disk speeds [15]. Karedla *et al.* examined the use of caching to reduce system response times and to improve data throughput [16]. It is believed that if the caches are separately treated, it is easy to control cache miss sequences. Each partition for a disk will be managed separately using the conventional LRU (Least Recently Used) replacement algorithm. We developed a cache partitioning mechanism for cluster storage system for achieving high security for data-intensive applications running on clusters [17]. More to the point, a number of cache partitioning methods have been proposed with different optimization objectives include performance [18], fairness [19], and quality of service [20]. On the other hand, SSDs can provide more advantages than the ones provided by the built-in cache because the built-in storage cache can only be used with that specific storage system while SSDs can be connected to any storage devices [21].

Hardware-Based Prefetching. Potential factors degrading I/O performance of storage systems include heavy load, low bandwidth, bandwidth underutilization, long seek time, and disk rotational delay. A promising approach to boosting I/O performance is to increase I/O bandwidth by prefetching data sets at various places, including storage servers, clients, and proxies. Prefetching techniques found in the literature take either software-based or hardware-based approaches [22] Software-based prefetching schemes depend on software to detect regular data access patterns, whereas hardware-based approaches rely on hardware to reduce data access penalty [23][24]. Hardware-based prefetching approaches can be classified into two groups: Spatial schemes where data access to the current block is the basis for making prefetching decisions, and temporal schemes where blocks to be prefetched is based on values of speculated data rather than data locality.

Software-Based Prefetching. Existing software-based prefetching solutions can be generally categorized into two camps: informed prefetching [25] and predictive prefetching [30]. Informed prefetching algorithms rely on user-disclose information about future requests to bring data into buffers. Predictive prefetching techniques utilize data access history to predict future data requests. There exist two types of predictive prefetching approaches: *dependency-graph-based predictive prefetching* (DGP) and *partialmatching-based predictive prefetching* (PMP). *DGP* algorithms use dependency graphs to model access patterns, whereas *PMP* algorithms maintain Markov predictors to calculate probabilities of future requests. These existing prefetching solutions have several defects: First, informed prefetching is inapplicable in scenarios that data intensive applications are unable to provide a priori information about which data blocks are likely to be accessed. Second, DGP algorithms predict forthcoming requests using previously accessed data with only considering first order dependencies. Third, PMP algorithms do not address the issue of how to choose a constant value for maximum order. Last, none of these existing prefetching algorithms considers bringing popular data sets from hard disks into solid-state disks.

III. The Hybrid Storage System Architecture

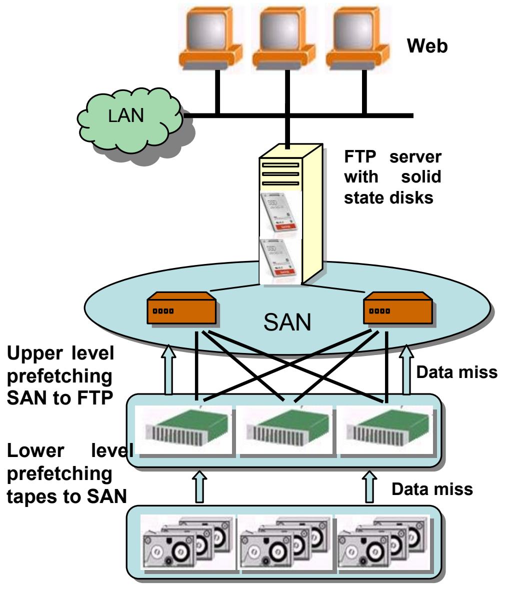

Web
LAN
FTP server with solid state disks
SAN
Upper level prefetching SAN to FTP
Data miss
Lower level prefetching tapes to SAN
Data miss

Fig.1 The Hybrid Storage System Architecture with Prefetching

Fig.1 depicts the architecture of a typical hybrid storage system. It consists of FTP servers with SSDs, storage area network (SAN) with HDDs, and a tape subsystem. The solid-state disks are designed to keep the most highly accessed files. Files with less priority or less access frequency are stored in HDDs of the SAN subsystem and all other files are stored in the tapes. The SSDs and HDDs are considered as high-end storage components while the tapes are classified as low-end storage components.

A prefetching scheme is designed to bring data into HDDs of SAN and optimally, into the SSDs before it is requested. Our multi-layer prefetching algorithm consists of two parts, the upper level prefetching and the lower level prefetching. The upper level prefetching transfer the data from the HDDS to the SSDs and the lower level prefetching transport the data from the tapes to the HDDs. A miss command will be issued when requested files cannot be found in SSDs and the missing files will be fetched to the SSDs from the HDDs. If the requested files are not located in the HDDs, the algorithm will issue next-level miss command and place the missing files from the tapes to the HDDs and finally to the SSDs.

IV. Prefetching Algorithms

A. Prefetching from Tapes to Hard Disks

In this section, we propose an approach to quickly move data from parallel tape storage system to hard disks to achieve high I/O performance. Parallel tapes libraries can intuitively increase the aggregate bandwidth between the disk storage and the tape storage and reduce the tape switch time by introducing parallel load/unload operation. This prefetching mechanism needs to schedule read requests in a way that the highest priority data blocks will be fetched first. To support the data transfer parallelism, a striping and data placement techniques for the hybrid storage system are used.

Striping is a well-known technique for improving the effective I/O bandwidth for storage systems. We consider data striping to access tape-resident data sets in parallel. Data sets will be divided into uniform chunks to be prefetched and stored to disks simultaneously. The data striping scheme is completely transparent to the PreHySys prefetching mechanism. To determine an optimal striping width, we consider both data size and I/O workload.

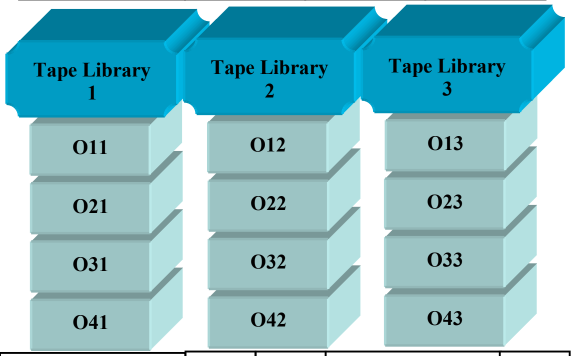
The striping technique can support data transfer parallelisms, thereby shortening I/O response times. However, striping causes a large number of small I/O calls, which in turn increase switch time of tape drives [28]. Considering the big penalty associated with tape switches and the long transfer time associated with the huge object size, we propose a data placement scheme to leverage object access probabilities and the relationship between objects. This scheme can improve tape switch parallelism and synchronize data seek with high probability, and thus increase the parallelism of the object transfer. The main goal of our data placement algorithm is to reduce the tape switches within the tape library, increase tape switch parallelism among tape libraries, and increase the data transfer parallelism among tape drives. To achieve these goals, we propose a data clustering mechanism that clusters objects with high probability to be requested together.

Our data clustering idea is based on an assumption that related data requests are highly to be requested together. Data blocks with a high probability to be requested together will be grouped in one cluster. The tape subsystem will be composed of *n* tape libraries. Each tape library has *d* identical tape drives, *s s*witch drives, and *t* tapes. The basic idea is to divide each tape library into two groups. The first group contains tapes that are mounted all the time and contains high emergency data. The number of tapes that are mounted all the time is *d-s.* The second and later clusters contain *s* tapes for each tape library, which is mounted during startup time and will swapped out if the requested stripe can not be found within the mounted tapes. Fig.2 is an illustrated example of object clustering mechanism in the tape storage. Assume there are three tape libraries where each tape library contains seven tapes, five tape drives, and two switch drives. The first batch for each library contains three tapes, and the second and later batches contain two tapes each.

a)

Index	1	2	3
Reference	O1	O2	O3
Priority	4	3	2

b)



c)

Step	1	2	3	4
Disk 1	O11	O13	O31	O33
Disk 2	O12	O21	O32	
Updated priority	- 3	- 2	- 1	
LRU Eviction	-	-	Either O11,O12 or O13	

Fig. 2 The data structure of the PRE-TD Algorithm. Fig. 2(a) is an access pattern known in advance. Fig. 2(b) shows the striping units of I/O requests distributed among tape libraries. Fig. 2(c) shows a prefetching schedule and updated priorities.

We develop a prefetching algorithm called PRE_TD for tape resident data. When the requested data sets are not located in the SAN subsystem, PRE-TD will fetch the data set from the tapes to the HDDs in SAN. PRE-TD does not prefetch data blocks if a higher priority data block must be evicted from the disks. We assume that the user's access pattern is known in advance. Each I/O request in the access patterns is assigned a priority level depending on data access history. It is worth noting that the priorities of striping units are equal to the priority of the corresponding I/O requests. Moreover, the priority of a data block in a disk is derived from the priority of the next reference to the data block. If there is no reference to the data block in the access pattern, the block's priority will be *i-Max*, where *i* is the index of the most recent reference in the data block, and *Max* is a large positive integer that is greater than the length of the access pattern. If data blocks residing in the disk subsystem are not appeared in access reference lists, then the blocks will be assigned the lowest priority. Data blocks will be migrated to tapes based on the least recent used policy. When PRE-TD is invoked, the striping units of an I/O request will be mapped to the disk subsystem using the round-robin mapping strategy. PRE-TD will construct a list of prefetched requests in accordance with their priorities. PRE-TD will have all the I/O requests in the list processed by disk drives after the requests are assigned to the disks. The PRE-TD algorithm is described in fig.3a.

B. *Prefetching from Hard Disks to Solid State Disks*

The performance of large-scale storage systems will be substantially improved by employing a number of solid-state disks that can help in reducing the number of hard disk accesses. We develop a parallel data transfer algorithm called PRE-DS that prefetches data set from the disk system to the solid state disks. The first component in PRE-DS is a SSD partitioning algorithm (or PaSSD for short), which dynamically partitions an array of SSDs among HDDs in such a way to maximize I/O performance. The basic idea of PaSSD is motivated by the observation that I/O workload is not uniformly distributed among parallel hard disks. The solid-state disks will be allocated dynamically depending on the popularity, size of content, and access patterns. Taking these factors into consideration, two approaches are implemented in PaSSD to assign weights for contents and associate a solid-state drive to each content. The first approach is called *Content-Popularity-Based Weight Assignment, in* which the weight of a new content is assigned based on its popularity of other relevant contents that already have measured popularities. If a large number of users have requesting content regularly, the content should be fetched and cached on a solid-state drive based on the measured weight. Weights will be adjusted dynamically and regularly to reflect any changes on the fly. The second approach is called *Collaborative-Popularity-Based Weight Assignment*. In this approach, we specify popularity weights by considering correlations among requested contents. For example, if user *u1* requested content *c1* and *c2*, and user *u2* requested content *c2*, then *c2* will receive a higher weight because there is a strong likelihood that both *u1* and *u2* will request *c2.* PaSSD will keep assessing contents and estimating the weights on a regular basis in a dynamic environment. After popularity weights are assigned using the above two approaches, PRE-DS will decide to forward requests to either SSDs or HDDs.

Furthermore, PRE-DS will be used to transfer data from HDDs to SSDs if requested popular data blocks are not available in the solid-state drives, and if transferring that data block does not cause a data block with higher priority to be evicted from the solid-state disks. To improve I/O performance, PRE-DS strive to fetch as many popular data blocks as possible from all hard disks and store these popular data in the high-speed solid-state disks. Fig.3b describes the PRE-DS algorithm.

a) The Prefetching Algorithm from Tapes to HDDs (PRE-TD)

Input

- The look-ahead reference is denoted by the sequence of references $R = (r_i, r_{i+1}, ..., r_j)$
- $P_{(r)}$: is the priority of request *r*
- Tape-library(r): is the tape library where tape(r) is located
- Tape(r): is the tape from which reference *r* needs to be fetched

PRE-TD:

- If the requested data is located in the disk system
    - No Prefetching is necessary
- If the requested data is not located in the disk system
    - Fetch the requested data from as many tape libraries as possible
- Use round robin mapping technique to map the striping unit to the corresponding disk
- Use the data placement algorithm to place the requests in the tapes
- If the disk is full
    - Use the LRU eviction policy to migrate request from disk back to tape

b) The Prefetching Algorithm for HDDs to SSDs (PRE-TD)

Input:

- The lookahead reference is denoted by the sequence of references $R = (r_i, r_{i+1}, ..., r_j)$
- $P_{(r)}$: is the priority of request *r*
- *Block(r):* is the block where request r is located
- *Disk(r):* is the disk where block(r) is located

PRE-DS:

- If the requested data is located in the solid-state disks
  - No Prefetching is necessary
- If the requested data is not located in the solid-state disks
  - Apply PaSSD mechanism to partition solid-state disks among disks
  - Fetch a request from as many disks as possible in the same I/O step
- If the solid-state disks is full
  - Use the LRU eviction policy to migrate requests from solid-state disks back to hard disk drives based on their popularity

Fig.3. The Multi-Layer Prefetching Algorithm (PreHySys)

VI. CONCLUSIONS

The use of large scale parallel disk systems continues to rise as the demands for data-intensive applications with large capacities grow. Traditional storage systems scale up storage capacity by employing more hard disk drives, which tends to be an expensive solution due to ever increasing cost for HDDs. With the new evolutionary technology on storage components like solid-state disks, hybrid storage systems, which contain both high-end storage components (SSDs and HDDs) and low-cost storage components (tapes), will be an ideal solution for next generation of data-intensive applications at petascale or extrascale level.

In hybrid storage systems, judiciously transferring data back and forth among SSDs, HDDS, and tapes is critical for I/O performance. We propose a multi-layer prefetching algorithm (PreHySys) that can reduce missing rate of high-end storage components thereby reducing the average response time for data requests in hybrid storage systems. To validate PreHySys, we also build an analytical model that can mathematically evaluate the performance improvement (i.e. the average access time improvement) when prefetching is carried out.

The contributions of this research are two folds. First, to the best of our knowledge, this is the first time multi-layer prefetching techniques are proposed in the context of hybrid storage systems. Second, we present the first mathematical model to evaluate the prefetching algorithms for hybrid storage systems. For future work, we will further improve our PreHySys algorithms and analytical model by applying them to real world hybrid storage systems with data-intensive applications.

ACKNOWLEDGMENT

The work reported in this paper was supported by the US National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CNS-0915762 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS).

REFERENCES

[1] M. Shenalon, R Heller, A Chai, P O Brown, and R W Davis, “Parallel human genome analysis: microarray-based expression monitoring of 1000 genes,” *Proc. Nat Acad Sci U S A.*, vol. 93, no. 20, pp. 10614–10619, Oct. 1996

[2] J.Hawkins and M. Boden, M., “The applicability of recurrent neural networks for biological sequence analysis,” *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 2, no. 3, pp. 243 – 253, July-Sept. 2005.

[3] D.B. Trizna, “Microwave and HF multi-frequency radars for dual-use coastal remote sensing applications,” *Proc. MTS/IEEE OCEANS*, pp. 532 - 537, Sept. 2005.

[4] H. Eom and J.K. Hollingsworth, “Speed vs. accuracy in simulation for I/O-intensive applications,” *Proc. Int’l Symp. Parallel and Distributed Processing Symposium*, pp. 315 –322, May 2000.

[5] W.W. Hutsell, A. Martz, “Caching is solid with disk option: solid-state, disk-based cache can improve the performance of sluggish networks-Storage”, *Communication News*, Dec.2003.

[6] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang and Z. Song, “PRO: A Popularity-based Multi-threaded Reconstruction Optimization for RAID-Structured Storage Systems,” *Proc. 5th USENIX conference on File and Storage Technologies (FAST’07)*,San Jose, CA, February, 2007.

[7] S.H. Baek et al., “Reliability and performance of hierarchical RAID with multiple controllers,”*Proc. twentieth annual ACM symposium on principles of distributed computing*, pp. 246 –254, 2001.

[8] G.R. Ganger, B.L. Worthington, R.Y. Hou and Y.N. Patt, “Disk arrays: high-performance,high-reliability storage subsystems,” *IEEE Computer*, Volume 27 , Issue 3, pp. 30-36, March 1994.

[9] T.Gerstel, “Streams and standards: delivering mobile video,” *ACM Queue*, Vol. 3, Issue 4, pp. 48-53, May 2005.

[10] R. Hou, J. Menon, and Y. Patt, “Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array,” *Proc. the Hawaii Int’l Conf. on Systems Sciences*, pages 70-79, 1993.

[11] G.F. Hughes and J.F. Murray, “Reliability and security of RAID storage systems and D2D archives using SATA disk drives,” *ACM Transactions on Storage*, Vol. 1, Issue 1, pp. 95 – 107, February 2005.

[12] A.A. Drapeau and R.H. Katz, “Striped tape arrays,” *Proc. 12th IEEE Symposium on Mass Storage Systems*, 1993.

[13] C. Georgiadis, P. Triantafillou, C. Faloutsos, “Fundamental of Scheduling and Performance of Video Tape Libraries,” *Multimedia Tools and Applications*, pp. 137-158, 2002.

[14] J. Li and S. Prabhakar, “Data Placement for tertiary storage,” *Proc. 19th IEEE Symposium on Mass Storage Systems*, pp. 193-207, April 2002.

[15] D. Kotz, D and C.S. Ellis, “Caching and writeback policies in parallel file systems,” *Proc. the third IEEE Symposium on Parallel and Distributed Processing,* 1991.

[16] R. Karedla, J. Spencer Love, and B. Wherry, “Caching Strategies to Improve Disk System Performance,” *IEEE Computer,* vol. 27, no. 3, pp. 38-46, March 1994.

[17] M.Nijim, Z. Zong, X. Qin, “Security-Aware Cache Management for Cluster Storage Systems”, *In Proc. ICCCN*, 2008.

[18] J.Chang and G.S. Sohi,”cooperative cache partitioning for chip multiprocessors, *In Proc. ICS’07*, 2007.

[19] S.Kim, D. Chandra, and Y.Solihin,”Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture, *In Proc. PACT’04*, 2004.

[20] R.Lyer,”QoS: A framework for enabling qos in shared caches cmp platforms”, *In Proc.ICS’04*, pp.257-266, 2004.

[21] WOODY Hutsell, Aaron Martz,”Caching is solid with disk option: solid-state, disk-based cache can improve the performance of sluggish networks-Storage”, *Communication News*, Dec. 2003.

[22] R.R.M. Rabbah, H. Sandanagobalane, M. Ekpanyapong, and W.-F. Wong, “Compiler Prefetching via Speculation and Prediction”, *Proc. of the 11th International Conference on Architectural support for programming languages and operating system*, 2004.

[23] J.-L, Baer and T.-F. Chen, “An effective on-chip preloading scheme to reduce data access penalty,” Proc. Supercomputing, pp. 176-186, 1991.

[24] Y.Chen, S. Byna, X.-H. Sun, R. Thakur, W. Gropp,”Exploring Parallel I/O Concurrency with Speculative Prefetching”, *Proc. 37th International Conference on Parallel Processing,* 2008.

[25] K.MK.M. Curewitz, P. Krishnan, and J.S. Vitter, “Practical Prefetching via Data Compression,”*Proc. ACM Conf. Management of Data* (SIGMOD’93), pp.257-266, June 1993.

[26] J. Griffioen and R. Appleton, “Reducing File System Latency Using a Predictive Approach,” *Proc. USENIX Ann. Technical Conf.* (USENIX’95), pp. 197-207, Jan. 1995.

[27] U.U. Hann, W. Dilling, D. Kaletta, “Improved adaptive replacement algorithm for disk caches in HSM systems”, Proc. 16th *IEEE Symposium on Mass Storage Systems*, pp128-140, 1999.

[28] X..-B. Zhang, D.-S. He, D. Du, and Y.-P. Lu, “Object Placement in Parallel Tape Storage, *ICPP* 2006.

[29] Z. Jiang and L. Kleinrock, “An Adaptive Network Prefetch scheme”, *IEEE Journal on Selected Areas in Communications*, pp. 358-368, 1998.

[30] L. Kleinrock, “Queuing Systems” Vol. 2: *Computer Applications*, Wiley, 1975.