Int. J. Communications, Network and System Sciences, 2015, 8, 181-186
Published Online May 2015 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2015.85019

A Comparison Study between Informed and Predictive Prefetching Mechanisms for I/O Storage Systems

Maen M. Al Assaf[1*], Ali Rodan[1], Mohammad Qatawneh[1], Mohamed Riduan Abid[2]

[1]King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[2]Al Akhawayn University, Ifrane, Morocco
Email: [*]m_alassaf@ju.edu.jo

Received 22 April 2015; accepted 17 May 2015; published 20 May 2015

Abstract

In this paper, we present a comparative study between informed and predictive prefetching mechanisms that were presented to leverage the performance gap between I/O storage systems and CPU. In particular, we will focus on transparent informed prefetching (TIP) and predictive prefetching using probability graph approach (PG). Our main objective is to show the main features, motivations, and implementation overview of each mechanism. We also conducted a performance evaluation discussion that shows a comparison between both mechanisms performance when using different cache size values.

Keywords

Informed Prefetching, Predictive Prefetching, Probability Graph, Parallel Storage Systems

1. Introduction

In I/O-intensive computing systems, I/O storage systems form a bottleneck in terms of performance due to the performance gap that is formed when they are compared with processors. Operating systems researchers have proposed a variety of software prefetching techniques that aim to preload the data from parallel disks prior to their actual on-demand requests [1]. Existing prefetching techniques can be categorized into two categories—informed and predictive. Predictive prefetching mechanisms predict the application' future I/O data accesses based on the historical I/O data accesses [2], whereas informed prefetching techniques take the advantage of the applications' ability to provide hints of their future I/O data accesses in order to preload data [3]-[5]. Transparent informed

[*]Corresponding author.

prefetching (TIP) [5] and predictive prefetching based on probability graph approach (PG) [2] are considered the leading researches in such scope. In this study, we propose a comparative study between TIP and PG approaches. We will discuss the features, motivations and contributions of both mechanisms. Our motivation of this research is: to our best knowledge, our study is the first that provides a study and performance evaluation that compares between informed and predictive data prefetching mechanisms. So, this comparison will guide researchers to the suitable prefetching mechanism when using a particular type of execution prototypes and applications. Hence, our main contribution in this research is conducted in a performance evaluation experiment that compares both of TIP and PG performance when using different cache size values.

2. Literature Review

Previous researchers have suggested that informed and predictive prefetching mechanisms improve I/O performance. To our best knowledge, however, our study is the first that provides a study and performance evaluation that compare between informed and predictive data prefetching mechanisms.

2.1. Informed Prefetching

An informed prefetching algorithm was firstly proposed by Patterson *et al*. [5]-[9] that takes the advantage of the application's ability to disclose hints of the future I/O data accesses. Hence, it invokes storage parallelisms to prefetch the data before they are actually accessed by the application. This eliminates I/O stalls and reduces the application's execution elapsed time [5] [6] [9].

In parallel storage systems, informed prefetching aims to leverage parallel I/O to improve prefetching performance [10] [11].

Other studies used prefetching to reduce storage system energy consumption. Eco-Storage built a module that prefetch a long sequence of future data accesses and turn off the disks system to reduce energy consumption [4].

Other studies investigated several ways of collecting information to offer accurate access hints for informed prefetching mechanisms. Accurate hints are important to make informed prefetching efficient [12] [13].

2.2. Predictive Prefetching

Predictive Prefetching (a.k.a., automatic prefetching) aims to build a history of the application on-demand data I/O requests in order to predict and to prefetch the future accesses [2] [14].

Griffioen and Appleton developed a predictive prefetching model that is based on probability graph approach. Probability graph in this approach is used to record the application's past access patterns in order to predict the future access probabilities [2]. Probability graph is a data structure that uses directed weighted graphs to estimate access probabilities [2].

There exist several predictive prefetching models and algorithms including data mining, neural networks, and Markov predictors [15]-[21]. For example, Marko predictors are widely used in web prefetching to predict future data accesses by applying partial match to find recurring sequences of I/O events.

3. The Compared Prefetching Algorithms

In this section, we will illustrate the key features, motivation, and contribution of both; transparent informed prefetching (TIP) [5] and predictive prefetching based on probability graph approach (PG).

3.1. Transparent Informed Prefetching (TIP)

Patterson *et al*. [5]-[9] proposed a transparent informed prefetching (TIP) solution that takes the advantage of the application' ability to disclose hints of the future I/O data accesses. Since parallel storage systems are able to provide several data blocks in parallel, TIP invokes storage parallelisms to prefetch the data before it is actually accessed by the application. Transparent informed prefetching reduces the application elapsed time due to the reduction of I/O stalls [5] [6] [9].

The major contribution of Patterson *et al*. TIP approach is a cost-benefit model that performs informed prefetching and balances cache/buffer space that is shared between the LRU (least-recently-used) cache and the prefetching buffer [5]. It makes a compromise between the benefit of using more buffers for prefetching and the cost of ejecting a LRU block or a prefetched data block.

The key motivations of transparent informed prefetching (TIP) research are:

1. The existence of storage systems parallelism.
2. The ability of applications to disclose hints of their future I/O data accesses.
3. The un-utilized parallel storage system bandwidth by the application.

Accurate hints are important to make informed prefetching efficient [12] [13].

3.2. The Probability Graph Predictive Prefetching Approach (PG)

Griffioen and Appleton [2] proposed a predictive prefetching algorithm based on probability graph approach. The main concept of this approach is to keep tracking the application's on-demand I/O data requests in order to build a history of the past accesses to be used for predicting the future requests and to have them prefetched. Their solution contributed in reducing the execution time of I/O-intensive applications. Prefetching decisions accuracy is the most important performance metric in predictive prefetching approach.

The following key factors motivate predictive prefetching research:

1. Not all applications can offer hints of their future data accesses.
2. Predictive prefetching is good for I/O intensive applications. It can prefetch the data even if the application is not running.
3. It is also good for multiple executables; because it relies on building a history based on the applications' on-demand I/O data requests. Hence; it detects access patterns of multiple applications that are executed repeatedly rather than taking hints from the running applications.
4. It utilizes the un-used parallel storage system bandwidth.

In Griffioen and Appleton model [2], probability graph data structure uses directed weighted graph that consists of nodes and edges to estimate access probabilities. In the probability graph, there exists a node for each data block stored in the storage system. Probability graph makes connections among nodes using directed weighted edges. The edges weights are used to predict the probability that a particular set of data blocks will be accessed in the near future if a particular data block is currently accessed. Lookahead period is an important input parameter used to build the probability graph. It determines the relationship between each of the application's consequent accesses. Minimum chance value is another input parameter that determines what data blocks to prefetch in case a particular data block was accessed. Edges weights are considered in this mathematics. Both values of lookahead period and minimum chance determine the degree of prefetching aggressiveness and accuracy. In this approach, a least recently used (LRU) cache is used to cache both of; the data read by the application on-demand I/O data requests and the prefetched data.

4. Performance Evaluation

In this section, we will run a performance evaluation comparison between transparent informed prefetching (TIP) and probability graph predictive prefetching approach (PG). We build a trace driven simulator using C++ to do the evaluation. First, we will illustrate our system design and assumptions. Then, we will discuss our performance evaluation.

4.1. System Design

Our simulator implements a parallel storage system that consists of an array of Hard Disk Drives (HDDs). Our simulator; and as other researchers did, uses small cache sizes that span from 1 to 10 cache buffers where each buffer can temporarily store one data block. The motivation behind using small cache size is to show the impact of the prefetching algorithm on the performance. In addition, the cache uses least recently used (LRU) policy to buffer the on-demand requests and the prefetched data.

4.2. Assumptions

Since we are using small cache sizes, there will be no need to issue too many concurrent prefetching I/O requests to the parallel storage system. So, regardless the size of the disk array, we assume enough I/O bandwidth that enables prefetching process to read few data blocks concurrently without causing any I/O congestion.

As we did in [3], [4], and [22], we use LASR real world trace [23] that represent an application that issues 11,686 I/O data read requests of about 800 distinct data blocks. Trace used in our experiments represents an application that performs a few overhead processing operations that consume a very tiny CPU processing time between each two subsequent I/O data requests. Hence, we will ignore the processing time. Our motivation behind that is to make the application purely I/O intensive. In case there exists some processing overhead between each two subsequent I/O data requests, this will easy the task of prefetching.

In [3], we validated the disk reading latency when using a range of several data block sizes that span from 1 to 10 MB using Intel 500 GB SATA 16 MB cache HDD. In this study used the smallest validated value (*i.e.* 1 MB) due to its suitable value (*i.e.* applications usually use small size data blocks). So, disk I/O reading latency for a single data block equals to **0.005 seconds**. Also, we used fixed size data blocks of size 1 MB. Hence; each cache buffer is also of size 1 MB.

In (PG) performance evaluation, we used a moderate degree of prefetching aggressiveness and accuracy by setting the lookahead period to 1 and the minimum chance to 0.5 as this setting represents the average case of (PG) performance.

4.3. Performance Evaluation

In this simulation, we test TIP and PG approaches performance in terms of trace (i.e. application) execution elapsed time when using different cache size values from 1 to 10. **Figure 1** shows the performance results. A decreased execution elapsed time indicates a performance improvement.

Figure 1 shows that both of TIP and PG provide a reduced execution elapsed time as the cache size increases due the increased hit ratio. TIP cannot provide a significant performance improvement when the cache size is very little (*i.e.* equals to 1 and 2). In [3], we called this case: "TIP's critical case"; where TIP is not able to allocate enough number of cache buffers for prefetching. In this case, TIP is only doing on-demand requests. Whereas PG in this case; provides some better performance; that is because it performs some aggressive prefetching. As the cache size increases, TIP shows significant performance improvement leaps. This is due to the accuracy of informed prefetching compared to the predictive one. When the cache size reaches to 9, TIP starts to show a little performance improvement leaps. This is because TIP at this point starts to reach the prefetching horizon where there exists enough cache buffers to store enough amount of data; so prefetching importance will become little.

PG shows a gradual performance improvement as the cache size increases. It is worth mentioning that both TIP and PG shows their best performance improvement jumps when using small caches. When cache size becomes

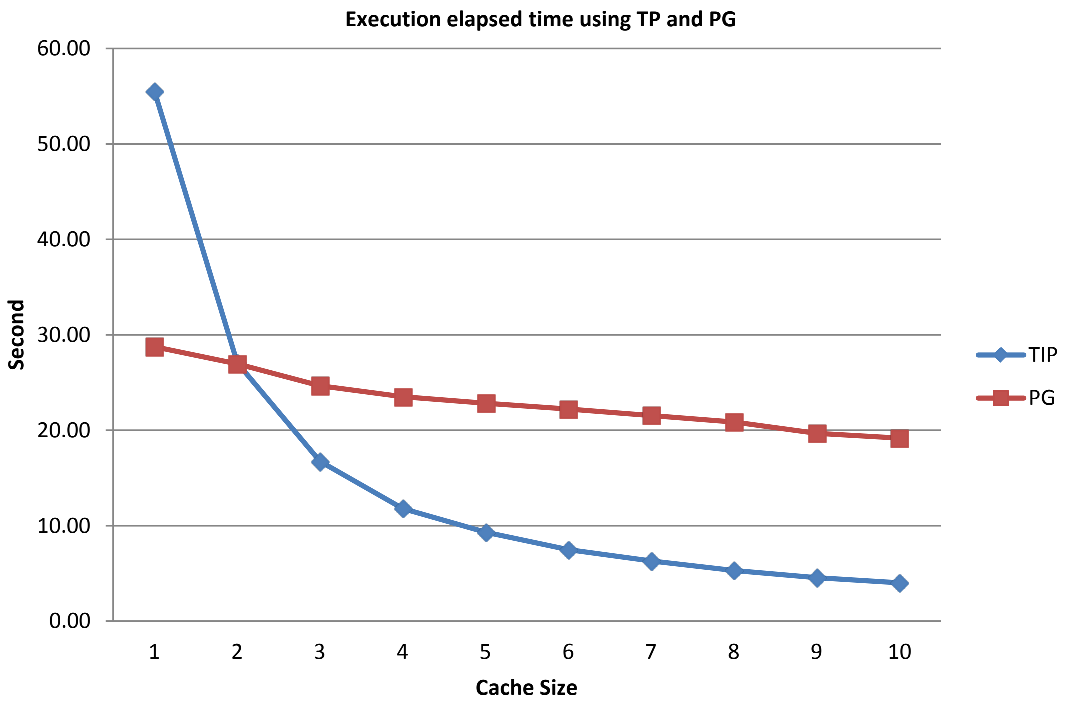

Figure 1. LASR trace execution elapsed time when implementing TIP and PG approaches using different cache size values from 1 to 10.

larger, performance improvement caused by prefetching will become limited. What makes TIP show a better performance improvement than PG in the most cases is the accuracy of prefetching decisions.

5. Conclusion

In this paper, we presented a comparison study between transparent informed prefetching (TIP) and probability graph predictive prefetching (PG) in terms of research motivation, implementation, and performance. We conducted a performance evaluation that compares both approaches when using different cache size values. Our performance evaluation shows that TIP in average provides better performance improvement due to its accurate prefetching decisions. In general, our results show that prefetching can provide significant leaps in performance improvement when using small size caches.

References

[1] Yang, C.-K., Mitra, T. and Chiueh, T. (2002) A Decoupled Architecture for Application-Specific File Prefetching. *USENIX Annual Technical Conference*, *FREENIX Track*.

[2] Griffioen, J. and Appleton, R. (1994) Reducing File System Latency Using a Predictive Approach. *USENIX Summer*, 197-207.

[3] Al Assaf, M.M. (2011) Informed Prefetching in Distributed Multi-Level Storage Systems. http://hdl.handle.net/10415/2935

[4] Al Assaf, M.M., Jiang, X.F., Abid, M.R. and Qin, X. (2013) Eco-Storage: A Hybrid Storage System with Energy-Efficient Informed Prefetching. *Journal of Signal Processing Systems*, **72**, 165-180. http://dx.doi.org/10.1007/s11265-013-0784-9

[5] Patterson, R.H., Gibson, G.A., Ginting, E., Stodolsky, D. and Zelenka, J. (1995) Informed Prefetching and Caching. *Proceedings of the* 15*th ACM Symposium on Operating System Principles*, Copper Mountain Resort, 3-6 December 1995, 79-95.

[6] Tomkins, A., Patterson, R.H. and Gibson, G. (1997) Informed Multi-Process Prefetching and Caching. *Proceedings of the* 1997 *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, **25**, 100-114. http://dx.doi.org/10.1145/258612.258680

[7] Patterson, R.H., Gibson, G.A. and Satyanarayanan, M. (1993) A Status Report on Research in Transparent Informed Prefetching. *ACM SIGOPS Operating Systems Review*, **27**, 21-34. http://dx.doi.org/10.1145/155848.155855

[8] Patterson, R.H., Gibson, G.A. and Satyanarayanan, M. (1992) Using Transparent Informed Prefetching (TIP) to Reduce File Read Latency. *Proceedings of Conference on Mass Storage Systems and Technologies*, Greenbelt, MD, September 1992, 329-342.

[9] Patterson, R.H. and Gibson, G. (1994) Exposing I/O Concurrency with Informed Prefetching. *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, Austin, TX, 28-30 September 1994, 7-16. http://dx.doi.org/10.1109/PDIS.1994.331737

[10] Kimbrel, T., Cao, P., Felten, E., Karlin, A. and Li, K. (1996) Integrated Parallel Prefetching and Caching. *Proceedings of the* 1996 *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 262-263. http://dx.doi.org/10.1145/233013.233052

[11] Ganger, G.R., Worthington, B.L., Hou, R.Y. and Patt, Y.N. (1994) Disk Arrays: High-Performance, High-Reliability Storage Subsystems. *Computer*, **27**, 30-36. http://dx.doi.org/10.1109/2.268882

[12] Chang, F. and Gibson, G.A. (1999) Automatic I/O Hint Generation through Speculative Execution. *Proceedings of the* 3*rd Symposium on Operating Systems Design and Implementation*, New Orleans, February 1999, 1-14.

[13] Byna, S., Chen, Y., Sun, X.-H., Thakur, R. and Gropp, W. (2008) Parallel I/O Prefetching Using MPI File Caching and I/O Signatures. *International Conference for High Performance Computing*, *Networking*, *Storage and Analysis*, Austin, 15-21 November 2008, 1-12. http://dx.doi.org/10.1109/sc.2008.5213604

[14] Lewis, J., Alghamdi, M.I., Assaf, M.A., Ruan, X.-J., Ding, Z.-Y. and Qin, X. (2010) An Automatic Prefetching and Caching System. *Proceedings of the* 29*th International Performance Computing and Communications Conference*, Albuquerque, 9-11 December 2010, 180-187. http://dx.doi.org/10.1109/PCCC.2010.5682310

[15] Chen, Y., Byna, S. and Sun, X. (2007) Data Access History Cache and Associated Data Prefetching Mechanisms. *Proceedings of the AMC*/*IEEE Conference on Supercomputing*, Reno, 10-16 November 2007, 1-12. http://dx.doi.org/10.1145/1362622.1362651

[16] Nanopoulos, A., Katsaros, D. and Manolopoulos, Y. (2003) A Data Mining Algorithm for Generalized Web Prefetching. *IEEE Transactions on Knowledge and Data Engineering*, **15**, 1155-1169.

http://dx.doi.org/10.1109/TKDE.2003.1232270

[17] Vellanki, V. and Chervenak, A.L. (1999) A Cost-Benefit Scheme for High Performance Predictive Prefetching. *Proceedings of the* 1999 *ACM/IEEE Conference on Supercomputing*, Portland, 14-19 November 1999, Article No. 50.

[18] Wang, J.Y.Q., Ong, J.S., Coady, Y. and Feeley, M.J. (2000) Using Idle Workstations to Implement Predictive Prefetching. *Proceedings of the* 9*th IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, August 2000, 87-94.

[19] Domenech, J., Sahuquillo, J., Gil, J.A. and Pont, A. (2006) The Impact of the Web Prefetching Architecture on the Limits of Reducing User's Perceived Latency. *IEEE/WIC/ACM International Conference on Web Intelligence*, Hong Kong, 18-22 December 2006, 740-744.

[20] Jeon, J., Lee, G., Cho, H. and Ahn, B. (2003) A Prefetching Web Caching Method Using Adaptive Search Patterns. 2003 *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, **1**, 37-40.

[21] Oly, J. and Reed, D.A. (2002) Markov Model Prediction of I/O Requests for Scientific Applications. *Proceedings of the* 16*th International Conference on Supercomputing*, New York, 22-26 June 2002, 147-155.
http://dx.doi.org/10.1145/514191.514214

[22] Al Assaf, M.M., Qin, X., Jiang, X., Zhang, J. and Alghamdi, M. (2012) A Pipelining Approach to Informed Prefetching in Distributed Multi-Level Storage Systems. 11*th IEEE International Symposium on Network Computing and Applications*, Cambridge, 23-25 August 2012, 87-95.

[23] LASR Trace Machine 01. http://iotta.snia.org/