

Usage Pattern Based Prefetching For Mechanical Mass Storage

Sohail Sarwar¹, Yasir Mahmood¹, H. Farooq Ahmed^{1,2}, Raihan-Ur-Rasool¹, Hironao Takahashi²

¹School of Electrical Engineering and Computer Science (Formerly NIIT)

National University of Sciences and Technology, Islamabad, Pakistan

²DTS Inc, Kyoudou Building 9F, 2-18-7 Higashiueno, Taitou-ku, Tokyo 110-0015 JAPAN

{sohail.sarwar, yasir drfarooq, dr.mehmood, raihan}@niit.edu.pk, hiro@dts-1.com

Abstract -Cache being the fastest medium in memory hierarchy has a vital role to play in concealing delays and access latencies during IO operations and hence in improving system response time. One of the most substantial approaches to fully exploit the significance of cache memory is data prefetching, where we envisage future requests of users and take data to memory in advance. Current prefetching techniques, performing limited prefetching, are based upon locality of reference principle (situation specific); Markov series (slow for practical implementation) or dual data caching (quite burdensome for programmer) with biased cache replacement policies. So we present a novel ‘usage pattern based’ approach for predictive prefetching; employing proven Neural Networks to broaden the scope of prefetching at user level. The efficacy of approach is revealed by its accuracy and minimal resource usage as affirmed by preliminary results

Key Words Caching, Locality of reference, solid state drives, neural networks

I. INTRODUCTION

The issue of disparity between ‘execution power of processors’ and ‘performance of peripherals’ (esp. that of secondary storage Devices (SSDs)) is not new to the research community. This performance gap is widened because of delays involved in the form of seek time (to locate the demanded data), rotational delay (to access the data location) and block transfer time (taking data from SSD to RAM) while manipulating the data. Disparate efforts [1,2,3] have been made to match IO performance of these peripherals to the speed of processors. These efforts have resulted in improved IO performance and provision of data in advance to the processors. Although the improvements made have not been able to make the speed of later and former comparable but with attained success, at least we can save much time and use the memory blocks efficiently; by lowering down the factors of seek time, rotational delay and block transfer time in searching and fetching the data blocks from secondary to primary memory. This improvement is courtesy to some available “cache prefetching” techniques with their respective pros and cons [1].

Cache prefetching is speculatively fetching data that will be accessed in the future to prevent wastage of CPU cycles,

searching from slow SSDs, match their speed gaps, avoid page miss and hide IO latencies [2].

The rest of paper is organized as follows. Section 2 discusses Related work. Section 3 presents the proposed solution and architecture. Section 4 provides the results and evaluation. Section 5 includes conclusion and future directions preceding the references.

II. RELATED WORK

There are several approaches that have been proposed, where majority of work related to caching is based upon "Locality of reference principle"[2], which in turn comprises of two parameters named ‘Temporal Locality’ and ‘Spatial Locality’.

First heuristic is based on the assumption that pages which have been referenced now are likely to be re-referenced in near future, so they must be retained in the memory (primary). The second one is based on the probability of demanding the pages which lie in a sequence next to the currently accessed pages, so pages residing in next sequence are fetched in advance to exploit the benefits of prefetching. Besides the foundation for prefetching laid by these principles we do own few concerns. These techniques are not flexible for considering the availability of resources, dynamically adjusting the proportion of memory allocation for data prefetching and amount of pages that can be prefetched. The first heuristic (temporal locality) remains unanswered for pages lying in sequence or we can say it does not appreciate the second principal when there is some sequential data available that can be prefetched. Also it requires the pages to be kept in main memory long enough so that they can be re-referenced for a cache hit (How long? Another issue). Whereas second one (spatial locality) is only suitable for the data that is in sequence and does not specify anything when comes the point to prefetch random pages (that means both models are situation specific).

Markov Prediction algorithm [4] has also been a strong candidate for its implementation in caching techniques. But the success of algorithm depends upon patterns in data stream and does not adapt itself to the situation for decision making.

Furthermore, these efforts for prefetching seem to be confined more on small workstations or PCs having small or medium level storage capacity rather than heavily loaded servers for efficient prefetching, carrying and accessing data from media containing data in terabytes or even in exa-bytes (such as clients requesting heavily loaded servers).

Recently, we observed that some ‘Solid State Drives’ [4] that use ‘flash memory chips’ to store data, had crouched in to replace SSDs or to overcome the associated problems of mechanical storage devices. These newly emerging devices are smaller in size, consume less power and have no moving parts (prevent mechanical delay and failure). Despite the advent of ‘flash memory’ based devices that have eliminated seek time and rotational delay unlike mechanical disks; still we foresee the need to improve the efficiency of mechanical storage devices. The reasons are quite apparent, “these devices are very expensive, with smaller storage capacity and it will take much more time in completely shifting the paradigm from mechanical to flash memory based mass storage devices due to mentioned constraints”.

So in order to cater the stated issues pertaining to improving the performance of mechanical devices for mass storage, we have introduced a prefetching technique which performs “Autonomous Trend monitoring of Information usage” for every user at runtime over the data that has been demanded and accessed. Previous systems prefetch information to bulk of users but we maintain and provide data for individual users. Our system dynamically decides about proportion of pages to be prefetched and performs safe prefetching (minimize the cache pollution and ejection of useful pages).

III. PROPOSED ARCHITECTURE

In our approach, we exploit the past information about use of a particular page while prefetching the data. Prefetching takes place by making prediction on the basis of user’s browsing history instead of implying adhoc methods paving the way to blind prefetching. We perform the above task in two phases; Learning mode (Fig 2) and Operational Mode (Fig 3). The information about usage patterns of data is collected during “Learning mode” of our system. We have instilled this phase of “learning” in the system because of its successful implementation in “Neural Networks (NNs)” [4] under the domain of Artificial Intelligence (equipping the system with capability of pattern extraction, trend detection and finally their recognition once learning phase is complete). This task of extracting patterns and detecting trends about individual users is carried out using ‘trend extractor’ (Fig 2). We sought out the neural networks to be suitable for our system due to their peculiarity of knowledge acquisition through learning and ability to store neural network’s knowledge within inter-neuron connection strengths known as ‘synaptic weights’ [4, 5]. In order to follow the analogy of NNs, we have used “Pattern Storage Repository (PSR)” (serving same as synaptic weights in

Neural Networks) to store the knowledge attained as a result of “learning phase”. In this phase system is trained and tamed with actual field data for sufficient amount of time (same as process of human learning). So that system, when subjected to real time ‘operational mode’, makes intelligent decisions suitable to scenarios by again recognizing extracted patterns from PSR. One question may arise about storage of these patterns after they have been extracted in operational mode. The answer might be explicated through analogy of ‘individual user profiles’ (such as user profiles in Windows or any web portal); Where, information is captured, stored, updated and re-loaded for individual users during the span of system’s information usage. When a user logs-on, his respective ‘prefetching information’ (from PSR) is furnished into the main memory automatically. Again, this process of taking the proportion of PSR for a user in memory entails consumption of time but according to our observation this instantaneous fraction of time spent is far more appreciated than suffering from several types of delays involved in searching, retrieving and fetching data and putting CPU in stall state (during IO processes from memory or peripherals) and user in waiting state unless IO operations are complete.

We expose our system to “Learning Mode” for a week at least in user mode. The system tracks user preferences for specific information requested. This enables us to meet user preferences in advance when he logs on to the system next time by recognizing trend information in PSR using ‘trend recognizer’ in ‘Operational Mode’ (Fig 3). This ability ensures the efficient “Resource usage” of overall system in normal and load situations as well. It means that we will be in position to calculate the CPU cycles consumed, memory required, pages cached, pages prefetched and time to retain those pages in memory for a specific user (as a result of data trends). After acquiring this information, we can not only predict but control the passage in and passage out for pages to and from memory as well. Hence we get in a position to even make a decision for deferring or even canceling the process of prefetching dynamically (due to increased memory pressure). This process of postponing the prefetching might be triggered due to several reasons such as congestion on server due to bulk of simultaneous requests for prefetching, insufficient time to train system for new users etc.

Another debatable issue is to decide the threshold for retention of prefetched pages in memory and their ejection after that time slot. This threshold has been static in most of the techniques previously (e.g. techniques exploiting temporal locality or LRU, MRU etc) and favored only frequently used pages to be preserved in memory. The pages which had been prefetched after utilizing time and IO resources were recklessly ejected without taking into account that they might be the useful ones (adding to cache pollution) and may be referenced in next cycle. The demand for these ejected pages will again consume resources and if same happens frequently, system performance may be

compromised. Therefore, we felt the need to put forth an unbiased threshold that is computed dynamically at run time on the basis of usage trend of the data.

We specifically focused to avoid bumping of useful pages and getting rid of stale ones. ‘Trend Recognizer’ of our system calculates the threshold considering available proportion of memory and room for page prefetching in operational mode(Fig3) .Moreover it updates newly captured request trends and plays a basic role in preventing CPU cycles from being wasted during IO operations.

We are mainly concerned with two aspects while prefetching; one is Accuracy (Extent to which the predicted data meets our requirement) and other is Coverage (the depth or amount of information prefetched for usage).

These measures are well checked and conformed by our “usage trend analysis” based technique. On one side if the “Learning phase” improves accuracy of prediction for page prefetching; coverage is improved. by efficient utilization of resources on the other hand. But sometimes we have to make a tradeoff between either of them.

There might be few concerns about overall system performance due to factors such as use of Neural Networks necessitating extensive use of resources, CPU usage for carrying out these tasks of predicting and prefetching, proportion of memory required or implication of loading irrelevant pages causing memory pressure.

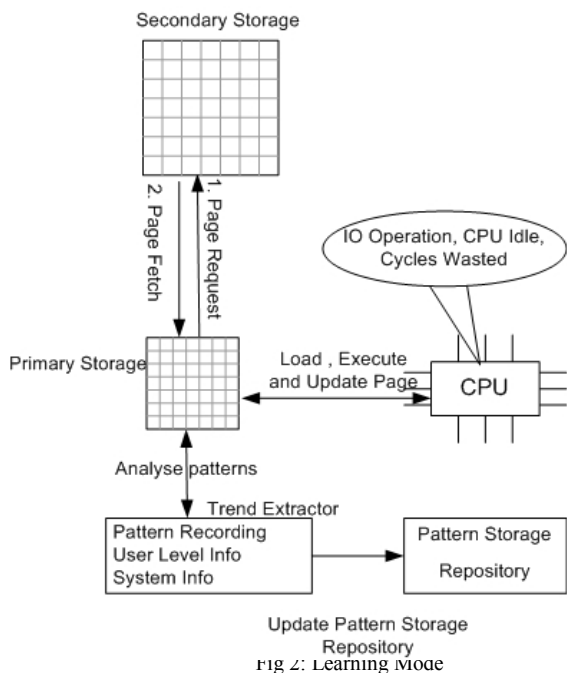


Fig 2: Learning Mode

These issues might lead to performance bottleneck in terms of CPU cycles and memory ultimately. We are looking forward to evaluate the complexity of initial version of our system in terms of CPU time consumed and memory bytes used using complexity measurement tool such as JProf (a tool to measure the code complexity of java code).

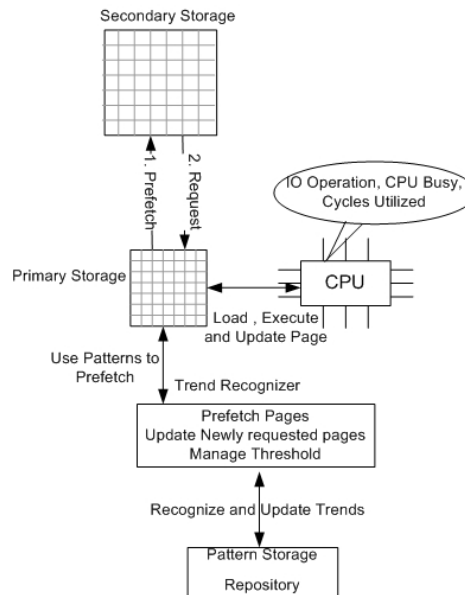


Fig 3: Operational Mode

Now let us see what are the suitable scenarios for deploying our systems? The developed system has a vast range of application in real time systems such as

- Database applications with IO intensive operations
- High performance database applications
- High IO transaction server and web servers

The above systems are the most suitable candidates for our application because of its assurance to improve quality of service at user level for taking into account preferences of individual users.

IV. RESULTS AND EVALUATION

The implementation of the architecture that we proposed is still in progress. The partially developed system resulted by tailoring and customizing open source tools named ‘netSaint’ [6], ‘Shipton’ and ‘Nagios [7]’ for evaluating our system. We deployed the developed system on server with quad-processor (3.2 GHz), RAM (24 GB) and HDD (320 GB). The partially developed system and some prediction model [8] have shown encouraging results in terms of Accuracy and Coverage. We specifically calculated the prefetching accuracy with and without incorporation of our solution. Prefetching accuracy is calculated as the ratio of relevant pages (available for prefetching in PSR for single user) to the total pages. Mathematically Prefetching Accuracy can be represented as

$$P(A) = (P_R \cap P_F) / P_T \quad (1)$$

Where

P(A)= Prefetching Accuracy

P_R= Relevant pages

P_F= Pages fetched out of relevant ones

P_T= Total number of pages

After plotting the results from partially implemented system and results of accuracy from above formula we can observe much improved behavior in following graph. The level of accuracy is quite low and unpredictable in absence of our approach.

The graph has not been plotted only the basis of partially developed system but exploits the “80-20 rule” as well. This rule specifies the heuristic that 80% of people access only 20% of information which means rest of information (80%) is not relevant to them. But our system on the basis of patterns in PSR brings almost 100% relevant information. But we cannot deny the presence of overheads involved that is not more than 30%, so ultimate level of productivity is assumed to be around 70% in terms of accuracy. When our approach is implied on the data, a comprehensive improvement can be observed. We can see that initially, level of accuracy is comparatively low even when our approach is in operation; this is the due to training phase (Learning mode) of our system. When this training phase completes (Operational mode starts) we see an up growth depicting enhanced level of accuracy.

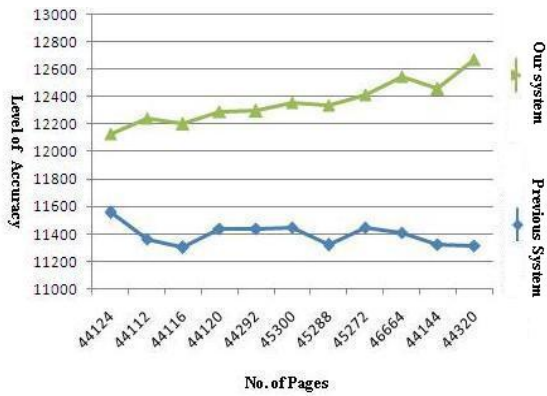


Fig. 4. Comparison of Existing and our proposed System

V. CONCLUSION

We propose architecture to overcome the latency of slow peripherals for saving CPU cycles and optimum resource usage via prefetching. This prefetching is based on usage history and intelligent pattern manipulation specific to users. The novelty of approach is signified to incorporation of Neural Networks for learning of system in extracting and recognizing knowledge patterns.

We are looking forward to make the system fully operational and deploy it in a real time working environment.

REFERENCES

- [1] Athanasios E. Papathanasiou and Michael L Scott, University of Rochester; Aggressive Prefetching: An idea whose time has come.
- [2] IO data Prefetching based on Sequential Stream Recognition, www.cs.unh.edu/~verkik/publication/cache.pdf.
- [3] S.V.Vardan, K.R. Arts Science Technology and Research Academy; Application of NN in predictive prefetching.
- [4] Flash Vs SSD; www.samsung.com/global/business/semiconductors/products/flash/products_flashSSD.html.
- [5] Christos Stergiou And Dimitrios Siganos. Neural Networks.
- [6] Stefano nolfi, dominico parisi, Institute of Technology National research council; Evolution of Artificial neural Networks.
- [7] www.netsaint.org.
- [8] www.nagios.org.