Potentials and Limitations of Fault-Based Markov Prefetching for Virtual Memory Pages

Gretta Bartels, Anna Karlin,
Henry Levy, Geoffrey Voelker
Department of Computer Science and Engineering
University of Washington
Seattle, WA
{gretta,karlin,levy,voelker}@cs.washington.edu

Darrell Anderson, Jeffrey Chase
Department of Computer Science
Duke University
Durham, NC
{anderson,chase}@cs.duke.edu

1. INTRODUCTION

Prefetching non-resident pages into memory before they are accessed can greatly reduce I/O stall time when compared with a fetch-on-demand strategy. While much recent research has focused on prefetching algorithms, some of this work assumes complete knowledge of a program's future reference stream[3,4]. Without such future knowledge, however, two limitations severely restrict the effectiveness of prefetching. First, with the exception of sequentially accessed files, prediction of future access patterns is difficult. Second, given the large latency of disk transfers, the cost of prediction errors is high, and therefore prefetching from disk may actually *degrade* performance by wasting critical system resources.

This work focuses on fault-based predictive prefetching in high-speed local-area networks. Our approach is fault-based because it accumulates access information only at page-fault events; the advantage of this scheme is that it is highly efficient and easily implementable within the operating system. We show that adaptive fault-based prefetching based on a one-level Markov net can achieve high prediction accuracy for some classes of applications.

Our study also focuses on high-speed LAN environments, because the combination of newer network technologies (e.g., gigabit-per-second networks such as Myrinet[1]) and cooperative network memory systems[2,5] offers a new opportunity for prefetching. When network memory is being used as backing store instead of disk, the cost of transferring (or prefetching) a page over a gigabit network is low – perhaps 50 times lower than the cost of a disk fetch[6]. However, this reduction in I/O latency is a two-edged sword: (1) it reduces the cost of a prediction error, thereby increasing the potential for speculative prefetching (i.e., prefetching without future knowledge); but (2) it reduces the total I/O stall time for the program, thereby also decreasing the potential benefit of successful prefetching. Our goal is to examine this tradeoff to determine the potential for speculative prefetching in this high-speed network environment.

2. MARKOV PREDICTION

Markov predictors are one implementation of prediction by partial matching (PPM). PPM estimates probabilities based on prior observation. A PPM of order n encodes sequences of length n and predicts the next element of a sequence given the n-1 immediately preceding elements. The predictor chooses the most frequently occurring transition from the state beginning with the n-1 observed elements and predicts the final element of the sequence.

Markov predictors encode sequences of elements as chains of nodes in a potentially sparse network, called a Markov net. Directed transitions, weighted by frequency, represent the likelihood that one element follows another. A Markov predictor traverses the Markov net beginning at a well defined start state and walks the net to the last known element in a sequence. The highest-weighted transition from this last element predicts the next element.

For a statistical prediction method such as a Markov predictor to succeed, patterns must exist in the data stream. Memory references frequently exhibit patterns such as repeatedly traversing a data structure or following a common path through conditional statements. For this reason, Markov nets make good memory reference predictors.

3. PREDICTION ACCURACY

We devised several alternative algorithms for fault-based prefetching, and evaluated the prediction accuracy of these algorithms when applied to various application types. The algorithms we tested included predictors designed to exploit spatial locality (by assuming sequential or strided access), temporal locality (by assuming repetitive behavior), or both. We included Markov predictors of order 1 and 2 in our tests. We found that fault-based Markov prediction schemes achieve good performance accuracy for some array-based scientific benchmarks and pointer-based benchmarks, but that they work poorly on applications with a large number of cold misses, which cannot be predicted by an adaptive algorithm.

Perhaps one of our most surprising results was that the Markov predictor of order 1 often outperformed the Markov predictor of order 2, because Markov-2 was too conservative. Markov-2 consistently had lower rates of misprediction. However, very often no prediction could be made because the sequence of two references needed to make a prediction had never been observed. Consequently, it had much lower rates of correct prediction.

4. PERFORMANCE

4.1 A Simple Model

We have developed a simple model for predicting a program's execution time speedup as a function of its miss rate, the average fault time, and the prefetching prediction accuracy. We assume that faults are distributed uniformly throughout the program, and that, at the time a fault occurs, a prediction is issued for the next fault. Because some of the algorithms decline to predict on a given fault (which may be better than mispredicting), we incorporate this possibility into our model. We use the following parameters:

- f – the fault time, measured in units of average inter-reference CPU time;
- r – the average number of references between misses (so that $1/r$ is the miss rate);
- p – the probability of correctly predicting the next miss; and
- n – the probability of not offering a prediction.

A discussion of the derivation of our model is omitted here, but we suggest the following measure R of the performance of the prefetching scheme: R is the ratio of the program's execution time *with* prefetching to its execution time *without* prefetching.

$$R = \frac{(1-n)\max(f,r) + f(1-p) + nr}{f+r}$$

4.2 Measurements

To test the performance of our fault-based prediction and prefetching scheme, we modified Voelker et. al.'s prefetching global memory system to use a one-level Markov net for speculatively prefetching virtual memory pages at fault time. The PGMS system[5] supports remote paging of file system and virtual memory blocks from the memories of other nodes on the network. Prefetched pages are marked invalid when they arrive, so that the system gains control if that prediction is correct; this allows the OS to update the Markov net and issue the next prefetch request on correct predictions.

In one experiment, we ran the *wave5* benchmark from the SPEC95 benchmark suite on the PGMS system with speculative Markov prefetching. The Markov predictor, the best predictor for this application, had a prediction accuracy of 80% for *wave5*. We ran the experiment on a small network of 600 MHz DEC Alpha 21164 processors, connected by a 1Gb/sec Myrinet[1] switched network running with Trapeze software and firmware[6] to accelerate page transfers. A page fault from a remote memory in PGMS on this configuration takes less than 200 μsec. In this environment, we measured only a 6% speedup for *wave5*.

Given the parameters for this configuration and *wave5*, our model predicts a *maximum* speedup of 10%. However, remember that obtaining this speedup would require, among other things, a perfectly uniform distribution of faults, in order to maximize overlap and hence benefit from prefetching. Our analysis of *wave5* indicates that more than 50% of its inter-fault times are less than f (200 μsec). This clustering causes the reduction in speedup from optimal.

5. CONCLUSIONS

We have examined fault-based Markov prefetching for virtual memory pages. Our results are twofold. First, we show that Markov prediction based only on the sequence of program-issued faults can achieve reasonably high levels of accuracy for some scientific applications. Using the fault sequence to predict future faults makes speculative prefetching of VM pages feasible to implement. We confirm this by implementing Markov prefetching in the PGMS network global memory system. Second, we show that despite high prediction accuracy, it is difficult in practice to obtain significant speedup from one-fault ahead speculative prefetching. This is because a system that can benefit from this kind of speculative prefetching must necessarily have a fault rate and a fault latency that are sufficiently low that the I/O overhead is not prohibitive in the first place.

We also note that a common criticism of many prefetching studies is the assumption that the prefetching system has full future knowledge. In fact, our study indicates that, while such future knowledge may be difficult or impossible to achieve, it may be required, in addition to parallel I/O, in order to reduce stall time.

6. ACKNOWLEDGMENTS

This work is supported in part by NSF grants CDA-95-12356, BSF-96-00247, CCR-96-24857, EIA-98-70724, EIA-98-70740, and by DARPA grant F30602-97-2-0226. Bartels is supported by a NSF Graduate Research Fellowship.

7. REFERENCES

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. -K. Su. Myrinet: a gigabit per second local area network. *IEEE Micro*, 15(1), 1995.

[2] M. Feeley, W. Morgan, F. Pighin, A. Karlin, H. Levy, and C. Thekkath. Implementing global memory management in a workstation cluster. In *Proc. of the 15th ACM Symp. on Operating Systems Principles*, 1995.

[3] T. Kimbrel, A. Tomkins, R. Patterson, B. Bershad, P. Cao, E. Felten, G. Gibson, A. Karlin, and K. Li. A trace-driven comparison of algorithms for parallel prefetching and caching. In *Proc. of the 2nd Symp. on Operating Systems Design and Implementation*. USENIX, 1996.

[4] T. Mowry, A. Demke, and O. Krieger. Automatic compiler-inserted I/O prefetching for out-of-core applications. In *Proc. of the 2nd Symp. on Operating Systems Design and Implementation*. USENIX, 1996.

[5] G. Voelker, E. Anderson, T. Kimbrel, M. Feeley, J. Chase, A. Karlin, and H. Levy. Implementing cooperative prefetching and caching in a globally-managed memory system. In *Proc. of the ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, 1998.

[6] K. Yocum, J. Chase, A. Gallatin, and A. R. Lebeck. Cut-through delivery in trapeze: An exercise in low-latency messaging. In *Proc. of the IEEE International Symp. on High Performance Distributed Computing*, 1997.