NASD Simulation and Prefeching Evaluation

Haijun Cao, Lei Ni

May 12, 2000

1 Abstract

This paper is focused on evaluating various informed prefetching algorithm under NASD environment. In order to do that, a NASD simulator is built and validated first. Then traces form two different sources are used to simulate various caching and prefetching algorithms. The results shows great I/O performance improvement using the informed prefetch even with small cache size. Also, we compared the effectiveness of client side prefetching versus disk side prefetching. Our simulation shows caching and prefetching at the client side is much effective than at disk side. Finally, our simulation also points out the effectiveness of informed prefetching depends on the accuracy of the future "hints".

2 Introduction

In the last decade, advances in hardware technology have created a wide performance gap between microprocessors and disks. As Amdahl's law [Amd67] pointed out, modern computer systems' overall performance is greatly limited by the part that primarily involves disks. One major challenge for today's operating system is to bridge the gap and keep the performance of I/O dominant applications improving at almost the same rate as that of processor speeds.

Various approaches have been taken to reduce or eliminate the performance discrepancy between microprocessors and disks. File caching was first proposed by Maurice Wilkes [Wil65] and has been a traditional method to reduce disk traffic. More recent researches involves file caching on distributed file systems, as well as on client and sever systems. Prefetching has also been studied extensively in file systems [PGS93, GA94] and databases [CKV93]. The prefetching algorithms varies from native one-block lookahead [Gin77] to sophisticated Transparent-Informed Prefetching (TIP) [PGS93, PGG95] which depends on "hints" for the future accesses. Generally speaking, file caching and prefetching have been proven to be two effective ways to increase I/O performance.

At the other hand, distributed file system and parallel disk environment further help to reduce I/O access time. Recent advances in technology have made disks both cheap and smaller. As a result, parallel disk arrays have became an attractive means for achieving high performance at low cost. Network Attached Secure Disks (NASD) is being developed in CMU [GNC99]. A NASD is a disk with built-in memory and CPU. It's attached to network by its own network interface. NASD minimized server-based data movement by separating management and file system semantics from store-and-forward copying. Compared with centralized storage approach, NASD has lower latency and higher scalability. Several popular file system (NFS, AFS) have been ported to NASD and a prototype has been implemented by PDL.

In essence, we believe by combining all three elements together, ie. caching, prefetching and parallel disk devices, the I/O performance can be greatly enhanced. In this study, we focused on the effectiveness of client as well as disk side caching and prefetching with explicit "hints" about future accesses.

The basic problems of integrating caching and prefetching are two:

- Replacement policy.
- Prefetching algorithm.

Compared with other caching and prefetching algorithm, informed caching and prefetching have advantage because future "hints" enable better cache replacement and more accurate prefetch. Two distinct approaches have been taken to utilize the "hints" about future accesses. Pei Cao's aggressive algorithm [CFG95] uses LRU as the replacement policy and prefetch as far as free memory space is available. Patterson et. al. in PDL used cost-benefit analysis to quantify the caching and prefetching decision. Figure 1 illustrates the cost-benefit analysis. The cache manager's goal is to allocate resources to minimize I/O service time. The managed resources are disks and file cache buffers. The consumers of these resources are demand accesses that miss in the cache and prefetches of hinted blocks. Buffers are supplied by the LRU cache and the hinted cache. Cost-benefit analysis assigns each prefetch a benefit value and each ejection a cost value as indicated Figure 1. Prefetch is only done while the cache manager can find a cache buffer which has smaller eviction cost than the prefetch benefit. It is proven by Tomkins that a informed prefetch only need to bring a limited number of future accesses into the memory. In other word, there's no benefit to prefetch a block beyond a fixed-horizon.[Tom97]

In Pei Cao and Patterson's study, they showed that with enough parallelism, informed prefetch can be used to bring down the I/O stall time eventually to zero. However, in all the studies mentioned above, Pei Cao and Patterson manually modified kernel to intercept I/O access and then use them as traces for simulation. The "hints" used in their simulation is 100% accurate. Few researches have been done as far as how to generate future access "hints" in the real environment. Griffioen and Appleton used "probability graphs" to predict future accessed from past ones [GA94]. Tait and Duchamp gave a predictive mechanism called the "working graph" [TD90]. Recently study done by Todd Mowry showed that compilers can be used to generate I/O "hints" during the runtime [MDK96]. In all cases, "hints" about future access are far from being 100% accurate.

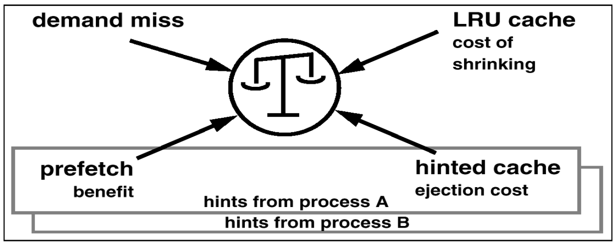

Fig1. Illustration of cost-benefit analysis approach

In this paper, we build a simulator for NASD environment and evaluated informed caching and prefetching algorithm using the simulator. We also measured how the accuracy of "hints" would affect caching and prefetching algorithm.

3 Simulator

We built the NASD simulator on top of disksim2.0 [Disksim]. We use the original disksim2.0 to simulate an array of bare disks without cache. On top of disk layer, we simulated network layer by adding network delay to all data transfered between disks and clients. An array of clients is then added to the simulator as the top layer which directly consumes traces.

As far as caching and prefetching concerned, we implemented caching and prefetching at both disk side and client side to mimic the real NASD environment. Four algorithms have been implemented. They are:

- LRU caching /no prefetching
- LRU caching /five-block lookahead prefetching
- LRU caching /aggressive prefetching with "hints"
- LRU caching with Cost-benefit analysis /fixed-horizon prefetching with "hints"

The detailed aggressive and fixed-horizon algorithms are illustrated in Pei Cao and Patterson's paper [CFK95][PGG95]. During the study, we tried to simplify the fixed-horizon algorithm by assigning a constant value to caching eviction cost. The result turn out to be a disaster. The final simulator we implemented only makes the assumption that the CPU process time for prefetch decision is 0. To validate the NASD simulator we built, we designed a trace file to validate all the caching and prefetching algorithms we implemented. The trace file has five distinctive records. Each of these five records repeats for 1000 time in the same order, ie. 1 2 3 4 5 1 2 3 4 5 ... The demand size for each record is 5kB. By controlling the cache size, we are able to measure the simulated cache hit number and compare it with prediction. Table 1. is the result we got.

Method	Client cache size	Disk cache size	Predicted client hit	Predicted disk hit	Client Hit	Disk Hit
0	5kB	5kB	0	0	0	0
0	20kB	20kB	0	0	0	0
0	40kB	40kB	4995	0	4995	0
1	5kB	5kB	0	0	0	0
1	20kB	20kB	0	0	0	0
1	40kB	40kB	4995	0	0	0
2	5kB	5kB	5000	5000	5000	5000
2	20kB	20kB	5000	5000	5000	5000
2	40kB	40kB	5000	0	5000	0
3	5kB	5kB	5000	5000	5000	5000
3	20kB	20kB	5000	5000	5000	5000
3	40kB	40kB	5000	0	5000	0

Table 1: NASD simulator vadilation results

The above result shows the hit number in the cache measured by our simulator is the same as we predicted. We didn't measure actually disk access time because it's calculated by parts of disksim2.0 codes which we didn't modify. We conclude that our simulator works for the trivial case and are ready to do further simulation with the simulator.

4 Traces

We used two sets of traces. The first set is from Pei Cao's study for aggressive prefetching study done in Princeton University. The traces were collected by tracing several applications running on a DEC 5000/200 workstations under Ultrix 4.3. This set of traces have the property that the running time of the applications are dominated by disk read access. The traces recorded I/O requests from one client to four disks in a time interval of about 150 seconds. The timing resolution is 40ns.

The other set of traces is Berkeley NOW trace. The Berkeley NOW trace records about 40 clients' file activities on 40 disks over a period of 24 hours. The Berkeley NOW trace are not read-dominated: about 10%-30% of the accesses are writes. However, we treat write accesses as if they are read access in our simulations. The timing resolution of the trace is 1 second. The rough timing of Berkeley NOW trace will certainly affect simulation result. However, because of the high parallel nature of Berkeley NOW trace. We made the assumption that client side queuing is not necessary, ie. there's no busts in the traces. Every I/O request on client side get processed right alway. The consequence of this assumption is that absolute time of trace is not important. Only the relative sequence matters.

All the trace files are in binary form and are described using file system terminology. In order to feed the trace file to the disksim2.0 we used. We extracted useful data for every record in the original trace file. Those useful data included access time, disk device no., file description no., file offset, demand block number, client no. and application number. Then we also discarded the accesses to devices other than disk. Finally, we used a linear interpretation to translate the file description to disk description.

$$filedeviceno. = diskdeviceno.$$

$$(filedescriptionno. \times averagefilesize + offset)\%disksize = diskblkno$$

5 Results

Berkeley NOW trace files used in this study are huge. The longest run took 126 CPU hours to finish. Most of the simulation results have been achieved by flocking jobs to a condor pool which has 42 almost dedicated ultra sparc2 processors and 1 TB disk space.

The results presented below use two metrics: average I/O stall time and hit number in the cache. These two metrics are closely related. However, we didn't go to the detail to calculate the disk access time. The average I/O stall time is calculated by disksim2.0 using the disk setting below.

All the simulation uses same disk configuration which mimics a Seagate Cheetah disk (SEAGATE_ST34501N). The disk size is 5GB. It has 8 surfaces and 6581 cylinders. The seek time for the disk various from 0.63600ms for single cylinder seek and 16.10700ms for full strobe seek. Other detailed information about the disk can be found in the attached diskspecs file.

5.1 Caching and Prefetching Method

Caching and prefetching method determines the effectiveness of cache usage. We first used Pei Cao's trace to evaluate the four informed prefetched algorithms. In this experiment, client and disk side cache have same cache size and prefetching method. The cache buffer is set to be 500k. The small buffer size is due to short time duration of Pei Cao's trace. The accuracy of future "hints" is set to be 100%. Figure 2 show the result of our simulation.

According to the simulation, informed aggressive and fixed-horizon prefetch reduced I/O stall time by one magnitude in the NASD environment compared with no prefetch. In figure 2, black boxes represent average I/O stall time for different algorithms while grey Boxes represent hit number in the client cache. It is observed that even without any prefetching, most of the I/O demands hit in the client LRU cache. However, with the informed prefetch, almost all I/O demands can be satisfied by client cache.

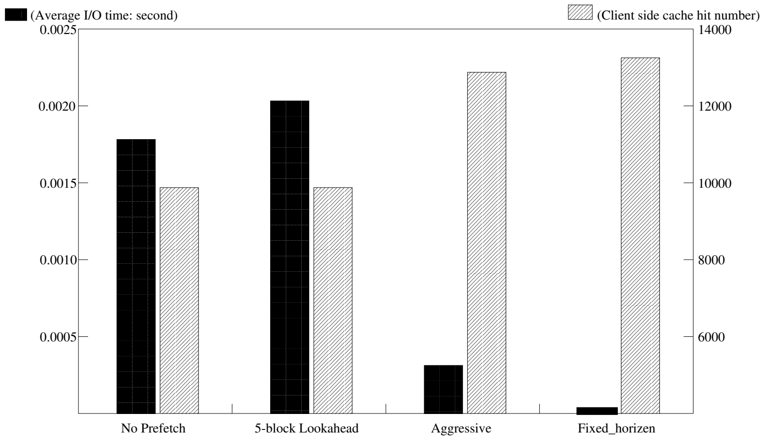

Figure 2. The Effect of prefetch algorithm on I/O performance(Pei Cao's trace)

We also did the same study using Berkeley NOW trace. The particular trace we used has more than 1.4 million records spread over 40 clients and 40 disks. The cache side used for this simulation is 50MB. Figure 3 is simulation result we got from that set of trace data.

The simulation result for NOW trace data again shows great I/O performance improvement using informed prefetch. The average I/O access time are reduced to nearly zero for both aggressive prefetch and fixed-horizon prefetch. We believe it's at lest partly because we disabled client side queuing for the NOW trace. The consequence is there's no burst in NOW trace. The simulator always have enough time to make the best prefetch decision and put is in cache before actual read happens. It is also observed that even with no prefetch, the average I/O access time of NOW trace is a magnitude smaller than that of Pei Cao's trace. Again, we think it's a result of lack of busts in the NOW trace.

By comparing the average I/O access time of 5-block lookahead an no-prefetch, we found that 5-block lookahead gave us worse performance in our case. There are several reasons, extra network delay is one of them. We choice 5 block arbitrary in our simulation. Prefetching less number of blocks may helps to reduce the extra network delay.

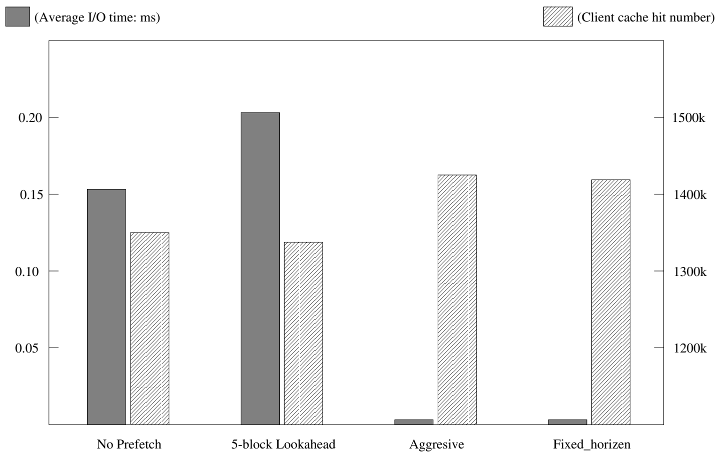

Figure 3. The Effect of prefetch algorithm on I/O performance (NOW trace)

5.2 The Effect of Cache Size

The next experiment we did is to evaluate the effect of cache size for prefetch algorithms. We used Berkeley NOW trace for this experiment. However, because of the large size of the trace. we truncated the trace file to get a small portion the original trace. The data set we work on has 200000 records. By doing that, the simulation result can be achieved within a day.

In this experiment, we again set the "hint" accuracy to be 100%. The client and disk side cache have same buffer size and prefetching method. We then increased cache size at both client and disk side for all four algorithms. The average I/O stall time was measured. It's not surprising to see form our simulation result that I/O generally performance better with larger cache, Figure 4. However, we observed that in the case of without prefetch and with n-block look ahead prefetch, the performance increase stopped at a certain value. We even increased the cache size to be 1GB in our experiment. The average I/O stall time for these two algorithm remained unchanged.

In the case of aggressive and fixed-horizon, the average I/O stall time quickly decreased to almost 0. Fixed-horizon performed better than aggressive. The I/O stall time for fixed-horizon reached 0 when th cache size is 10MB. Give the cache size of modern computer, it is expected that fixed-horizon can fill the gap between I/O and CPU performance at most cases.

However, the above simulation didn't consider I/O busts as we used NOW trace data.

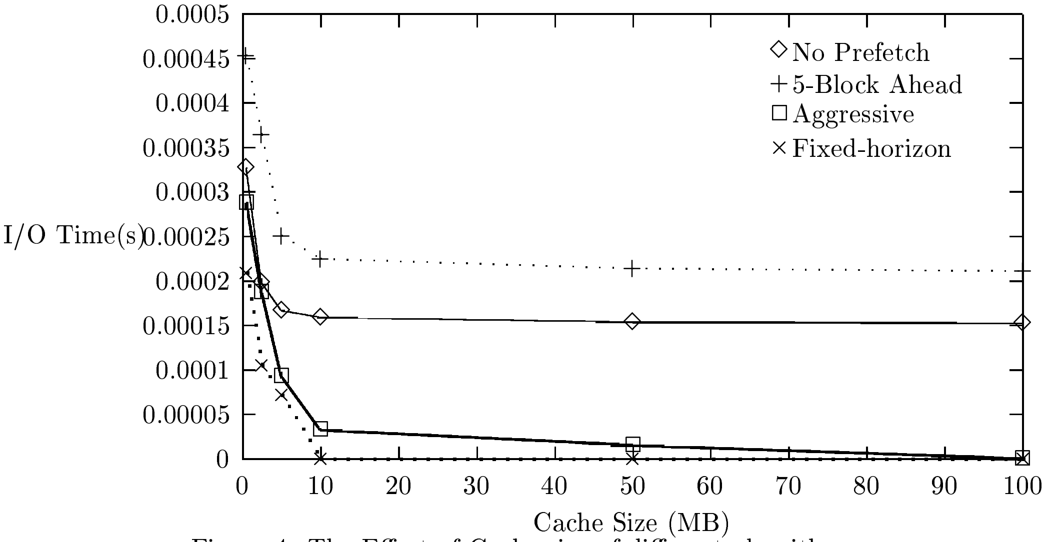

Figure 4. The Effect of Cache size of different algorithms

5.3 Client Side Versus Disk Side

We then studied the effectiveness of client prefetch versus disk side prefetch.

We didn't use n-block lookahead algorithm in this experiment because it gave us worse performance than no prefetch in our simulation. The client side prefetch algorithm was then set to be one of no-prefetch, aggressive and fixed-horizon. For each algorithm used at the client side, we used three different caching and prefetching algorithms at the disk side, ie., no prefetch, aggressive and fixed-horizon. The cache side is set to be 10MB at both sides. The trace data is Berkeley NOW trace.

The simulation results in Figure 5 shows client caching and prefetching is more effective than disk caching and prefetching. In figure 5, clients using same prefetching algorithm are clustered together. It is observed that the difference inside each cluster (same client side prefetching algorithm, different disk side prefetching algorithm) is much less than the difference among clusters (different client side prefetching algorithm).

It is believed that client side caching and prefetching out-performed those of disk side. In the case of client side with no prefetching, every demand-miss has the minimum stall time of network delay. In the case of client side with informed prefetch, the disk side caching and prefetching algorithm is trivial because most of the I/O access can be satisfied with the client local cache.

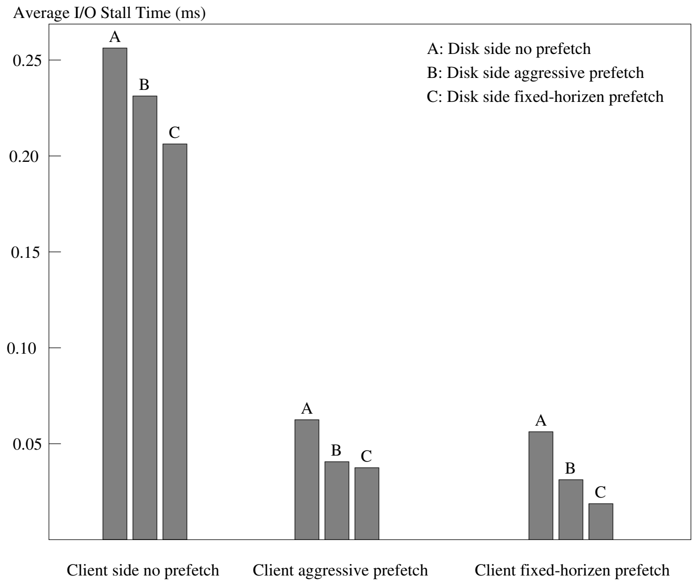

Figure 5. The effect of client prefetch vesus disk prefetch

5.4 The Effect of Hint Accuracy

The last study we did is the effect of hit accuracy for informed prefetching. As indicated in the introduction part, the "hints" about future I/O access can not be predicted accurately in the real time. With our simulator, we are able to vary the "hints" accuracy and measure the I/O stall time for informed prefetching algorithm.

The trace data we used is again Berkeley NOW trace. We set cache size to be 10MB at both client side and disk side. The fixed-horizon algorithm was chosen for this experiment because it had best performance thus far. The "hint" accuracy was varied from 50% to 100%. The average I/O stall time is measured. The result is shown in figure 6.

The simulation result shows great dependence of I/O stall time on "hint" accuracy. The I/O stall time rapidly increased in the range of 80% to 50% "hint" accuracy. Given the simulation we already did in the cache size study, we know that the average I/O time for no-prefetch at this circumstance is about 0.2ms. This number is about same as fixed-horizon prefetching with the "hint" accuracy of about 80%.

According to the simulation, informed prefetching with low "hint" accuracy "can do damage to I/O performance. The reason is fixed-horizon doesn't use complete LRU caching, it uses cost-benefit analysis to find the lowest cost buffer to evict. Wrong "hints" about future usage will make the buffer manager evict those blocks needed in the near future.

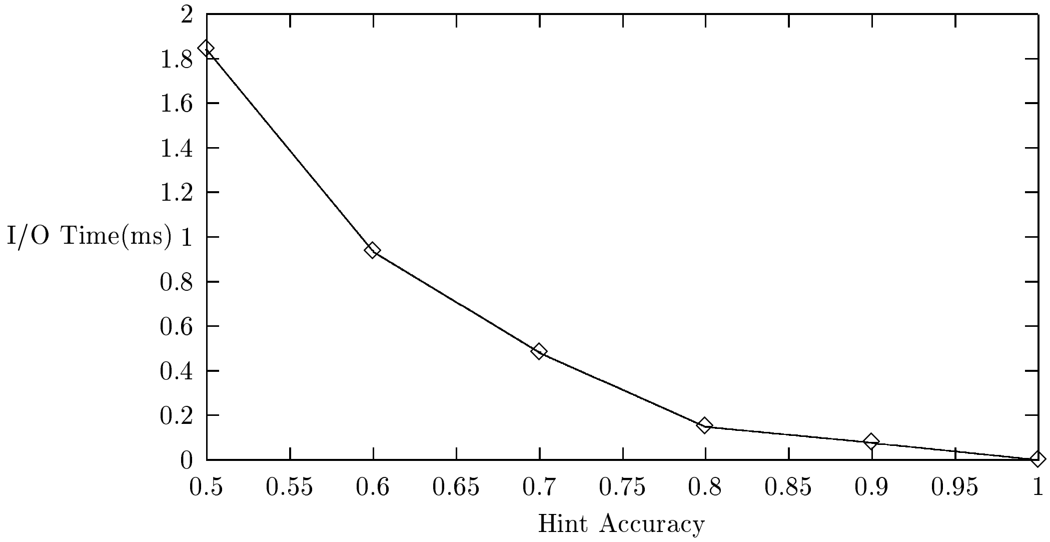

Figure 6. The Effect of Hint Accuracy

6 Conclusion

File caching and prefetching have been studied for a long time. However, informed prefetching appeared only in recently years. In this paper, we studied informed prefetching under NASD environment. Here are the conclusions we made for this study.

- With accurate "hint" about future, informed prefetch can greatly reduce the I/O stall time. With high parallel disk environment like NASD, informed prefetching can potential bring the I/O stall time down to 0.

- Memory in modern computer is large enough for file caching and prefetching purpose. According to our simulation study, there's no need to increase cache size beyond 50MB. However, we also believe our simulation result is limited by the trace file we used.

- Client side caching and prefetching is more efficient than disk side caching and prefetching. Our simulation study shows the effect of disk side caching and prefetching in the NASD environment is limited. Partially it's because it's out-performed by client side caching and prefetching. Also, network delay between disk and client contributes to the result.

- "Hint" accuracy has large impact on the effectiveness of informed prefetching. In order to make informed prefetching work efficiently, the "hint" accuracy has to be greater than 80% according to our simulation.

7 Bibliography

[Amd67] G. M. Amdahl. Validity of the single Processor Approach to Achieving Large Scale Computing Capabilities. In Proceedings of AFIPS Spring Joint Computer Conference, page 483-485, April 1967.

[CFK95] Pei Cao,Edward W. Felten,Anna Karlin and Kai Li.Integrated Parallel Prefetching and Caching.In Proceedings of SIGMETRICS/Peformance'95.

[CKV93] Kenneth M. Curewitz,P. Krishnan, and Jeffrey Scott Vitter.Practical Prefetching via Data Compression.In Proc.1993 ACN-SIGMOD Conference on Management of Data, page 257-266, May 1993.

[Disksim]http://www.ece.cmu.edu/~ganger/disksim/

[GA94] Jim Griffioen and Randy Appleton.Reducing File System Latency Using a Predictive Approach. In Conference Proceedings of the USENIX Summer 1994 Technical Conference, page 197-208, June 1994.

[Gin77] J.Gindele. Buffer block prefetching method.IBM Technical Disclosure Bulletin, 20(2):696-697, July 1977.

[GNC99] A. Gibson, David F. Nagle, William Courtrigh II, Nat Lanza, Paul Mazaitis, Marc Unangst, and Jim Zelenka. NASD Scalable Storage Systems. In the Proceedings of USENIX 1999,Linux Workshop,Monterey, CA. June 1999.

[MDK96] T. Mowry, A. Demke, and O. Krieger. Automatic Compiler-Insered I/O Prefetching for Out-of_core Applications. In Proceedings of Second USENIX Symposium on Operating System Design and Implementation (OSDI). page 28-31. October 1996.

[PGG95] R. Hugo Patterson, Garth A. Gibson. Eka Ginting, Daniel Stodolsky, and Jim Zelenka. Informed Prefetching and Caching. In Proceedings of 15th ACM Symposium on Operating System Principles,December 1995.

[PGS93] Hugo Patterson, Garth Gibson, and M. Satyanarayanan. Transparent Informed Prefetching. ACM Operating System Reviews, page 21-34, April 1993.

[TD90] Carl D. Tait and Dan Duchamp. Detection and Exploitation of File Working Sets. Technical Report CUCS-050-90, Computer Science Deprtment, Columbia University, 1990

[Tom97] Andrew Tomkins. practical and Theoretical Issues in Prefetching and Caching. h.D Dissertation CMU-CS-97-181. October 1997.

[Wil65] Maurice Wilkes. Slave Memories and Dynamic Storage Allocation. IEEE Transactions on Electronic Computers, page EC-14(2):270-271, April 1965