WORKSHOP

ON

PARALLEL PROCESSING

The Lewis Room
Capitol Holiday Inn
Washington, D.C.

August 28 - 29, 1989

Supported from the Defense Advanced Research Projects Agency
through the National Aeronautics and Space Administration

High-Performance File System Design for MIMD Parallel Processors

David Kotz
Duke University

with Prof. Carla Ellis

Motivations

• File systems for parallel processors have been largely ignored.

• To avoid serial bottlenecks, parallel I/O and file systems should be used.

• Caching and prefetching have been used successfully to boost file system performance in serial computers, and we believe this can be extended to parallel computers.

Issues

General:

- The combined performance of all threads in a parallel application is more important than that of any one thread.
  In order to speed up the application, we must speed up all threads.

I/O:

- To avoid waiting for I/O under multiprogramming, overlap the I/O with useful computation by switching to a "ready" process.

- Many parallel systems do not multiprogram the individual nodes.

- We thus use these idle moments of the application process to do *prefetching*.

Prefetching: Loading blocks into the cache before they are requested.

- Prefetching overhead must be minimized in order to attain maximum benefit.

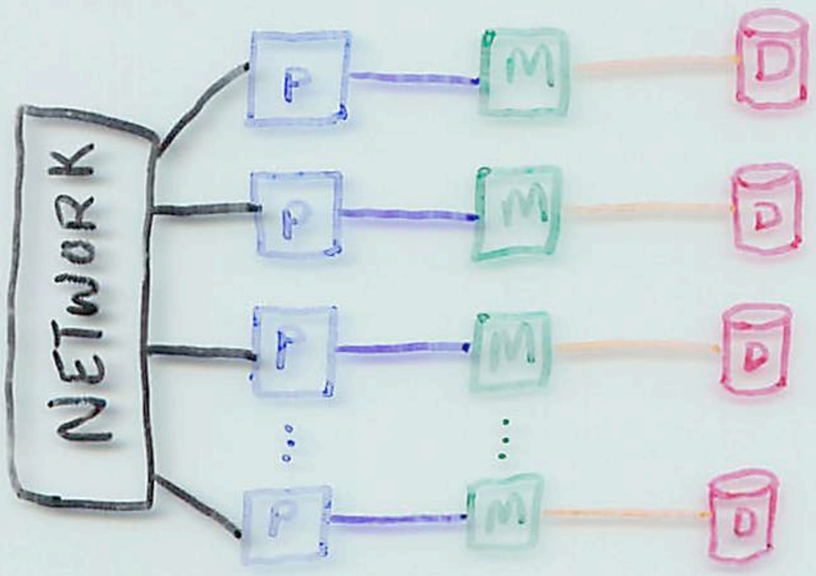
Completed Work

• Question: Is it possible to reduce overall application execution time by prefetching file blocks?

• Simplified version: given *a priori* an ordered list of all file references, will prefetching reduce the overall execution time?

If not, then it can't be done on-the-fly.

• Experimental approach:
    RAPID-Transit testbed.
    Architectural model:
        parallel independent disks.
    Synthetic workload.



Summary of Results

+ Average time required to read a block reduced in all cases.

+ Cache hit ratio increased from near zero to over 69% in all cases.

- Disk contention (as measured by queuing time) increases.

- Synchronization delays increase.

+ Total execution time reduced in most cases:

  Maximum 69% reduction

  Median 19% reduction

- In some cases, however, the application was 5-15% slower

Conclusion: potential for improving performance with prefetching is good.

Benefit Balancing

• Primary direct benefit of prefetching is reduced block read time.

• If the benefit is unevenly distributed between processors, some processors fall behind.

• The faster processors must wait for the slower processors at the next synchronization point.

this all leads to

• No improvement in overall execution time.

A possible solution:

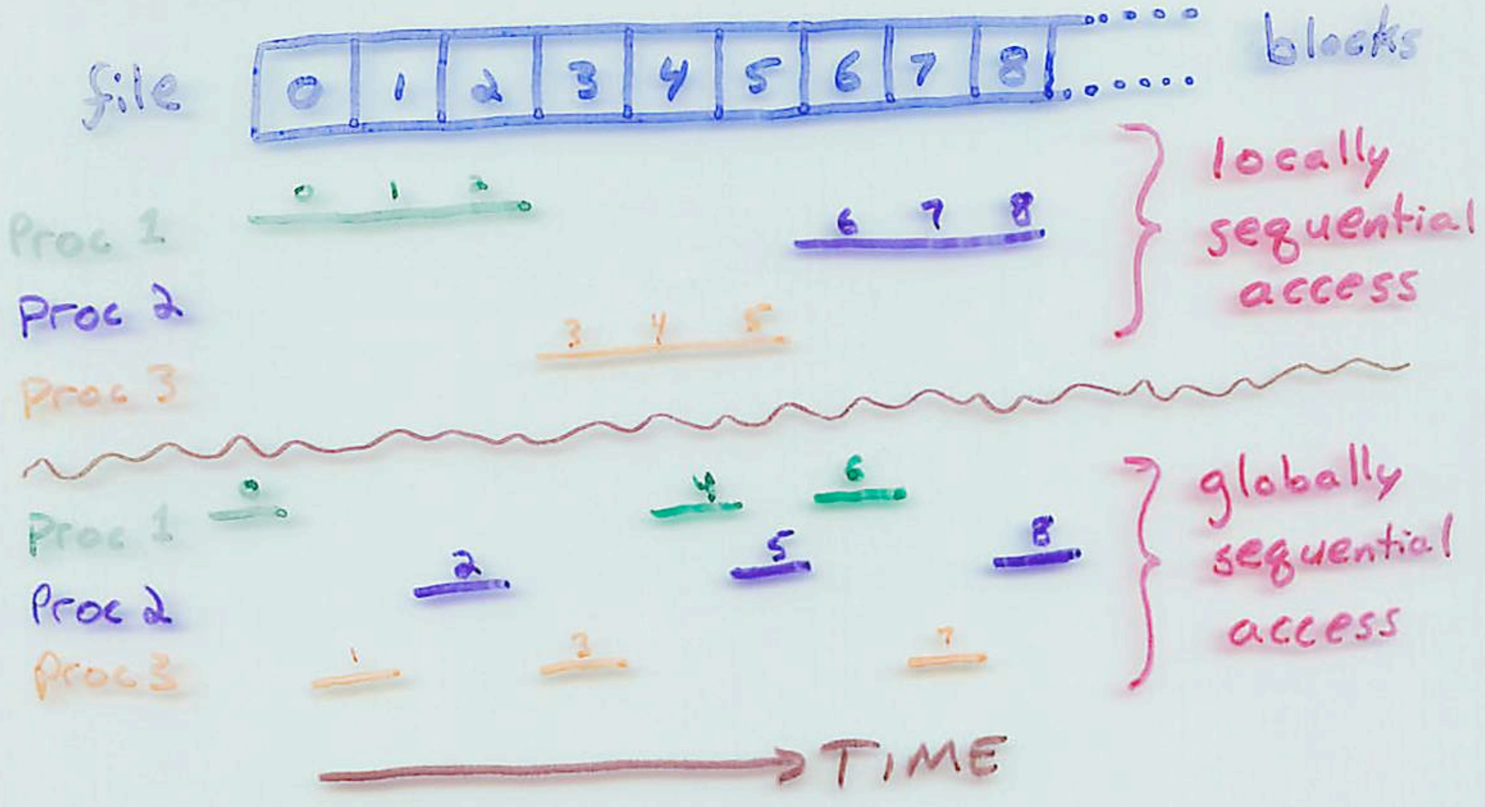
• Adjust the prefetching strategy so that prefetching benefits all processors equally.

• Preliminary results: prefetching now improves total execution time in all cases.

Access Pattern Prediction

Given a sequential access pattern, predicting the future is easy - in a serial processor.

In a parallel processor, what is a "sequential" access pattern?



How to detect globally sequential patterns, in parallel, on the fly, with low overhead?

Other Continuing Work

- Attain more accurate and complete workload models.
- Scaling of algorithms, architecture.
- Layout of files on disk: contiguous?
- Location of file system control: dedicated I/O nodes?
- NUMA issues in buffering.
- Analytical models.

Summary

- Parallel I/O is an important issue.
- Caching and prefetching are important for high performance.
- Result: prefetching has the potential for improving I/O performance.
- There are many issues to study regarding automated prefetching in parallel.

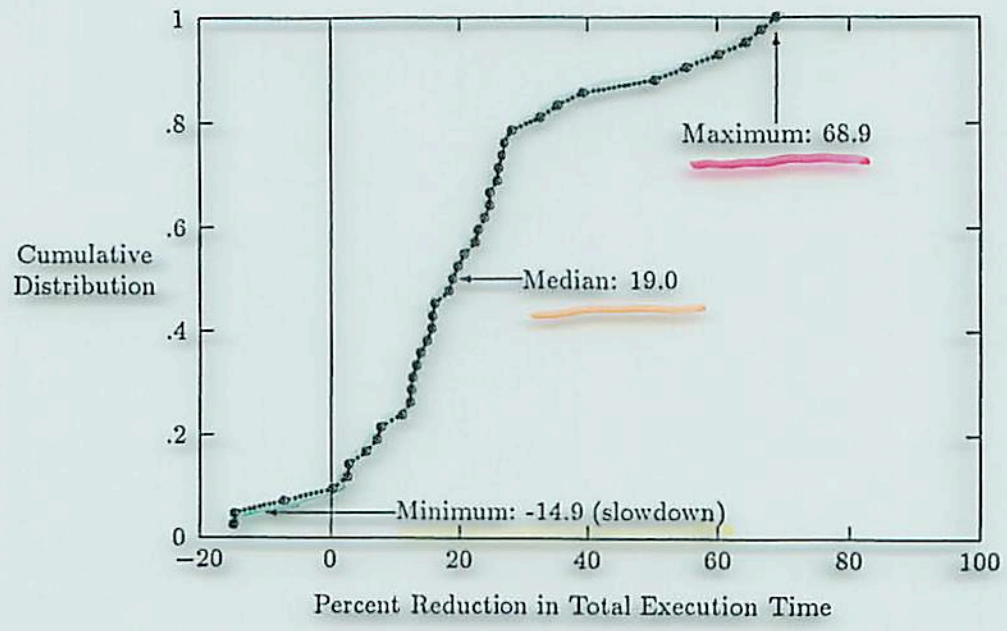
Project Publications

David Kotz and Carla Ellis, "Prefetching in File Systems for MIMD Multiprocessors", IEEE Trans. on Parallel and Distributed Systems, Volume 1, Number 1. (shorter version in ICPP '89).

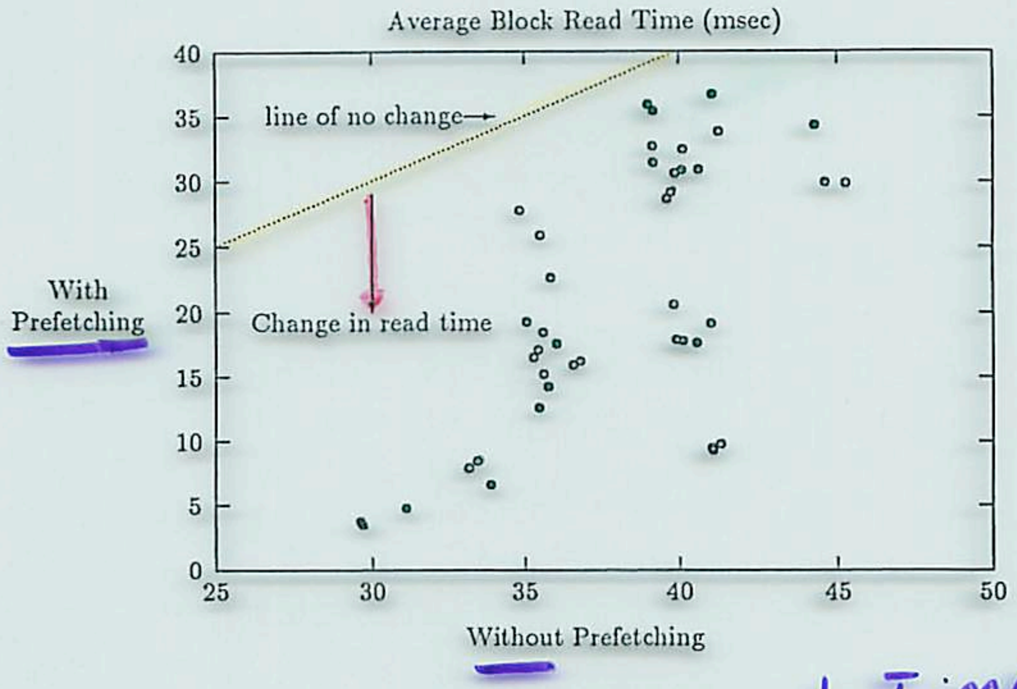
Mark Holliday, "Reference History, Page Size, and Migration Daemons in Local/Remote Architectures", Proc. Third Int. Conf. on Architectural Support for Programing Languages and Operating Systems, 1989.

Mark Holliday, "Page Table Management in Local/Remote Architectures", ACM SIGARCH Int. Conf. on Supercomputing, 1988.

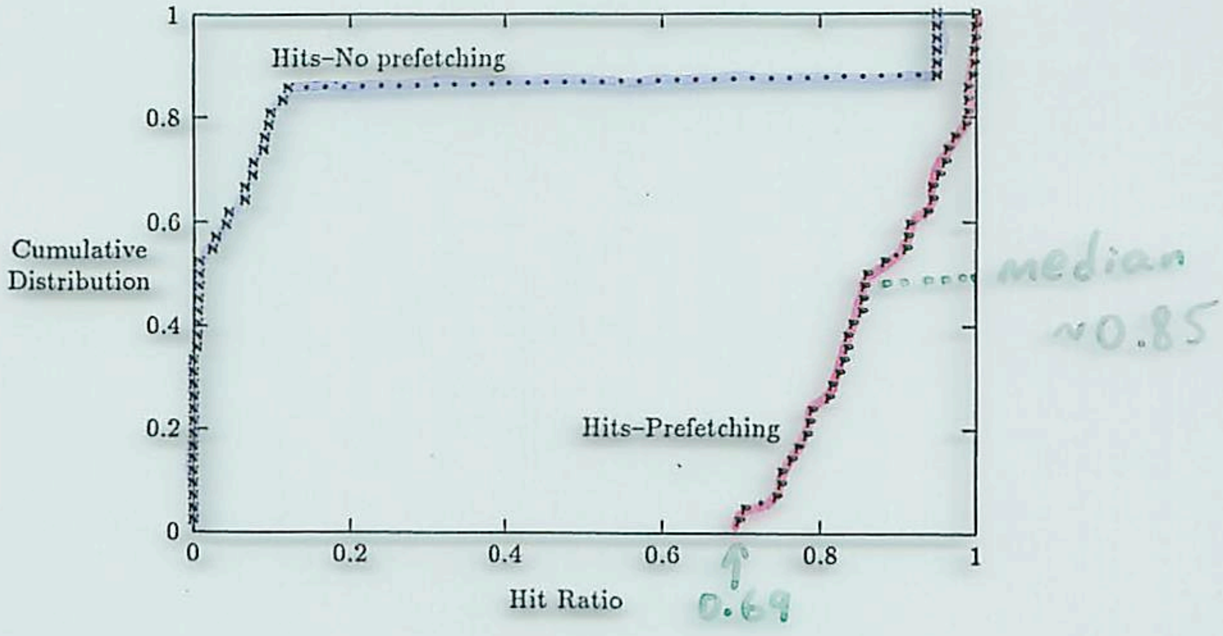
Rick LaRowe, "NUMA Multiprocessor Page Placement Policies", Duke University TR CS-1989-2, submitted for publication.



% Reduction in Total Time



Average Block Read Time in Milliseconds


Hits-No prefetching
Hits-Prefetching
Cumulative Distribution
Hit Ratio
median ~0.85
0.69
0
0.2
0.4
0.6
0.8
1

Hit Ratio