Informed Prefetching and Caching

R. Hugo Patterson,
Garth Gibson, Eka Ginting, Dan Stodolsky, Jim Zelenka

Goal: Reduce file read latency

Application: disclose future reads in hints

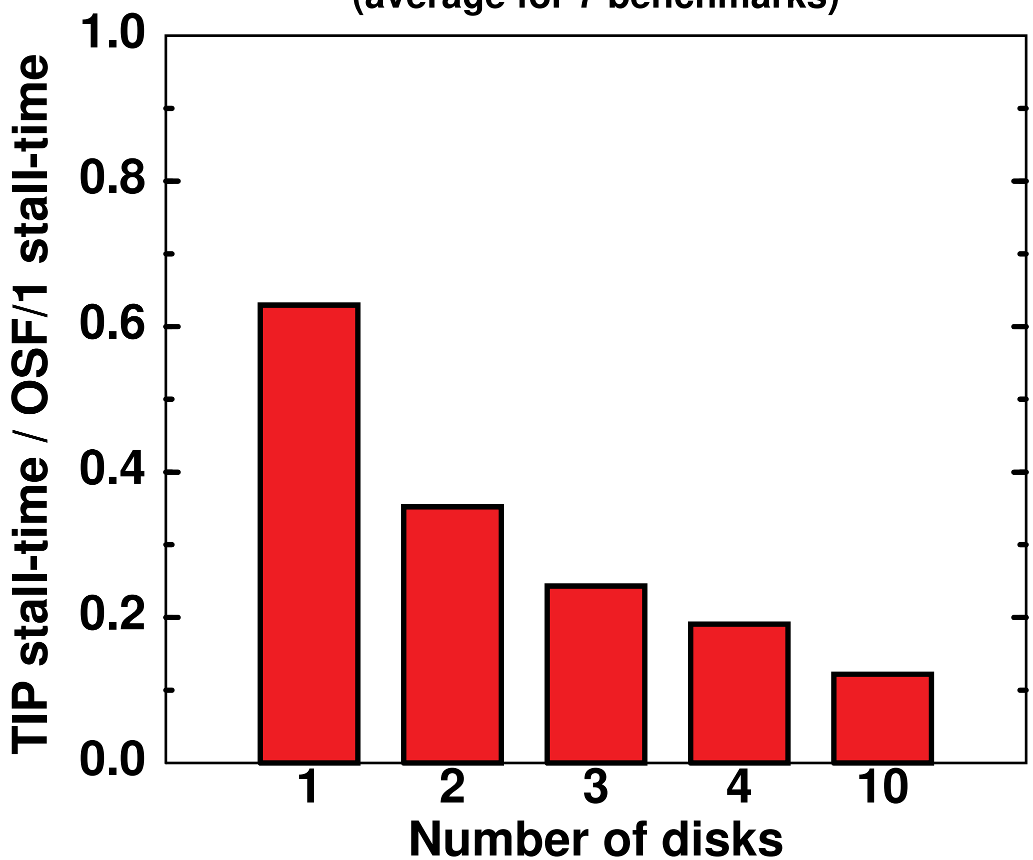
File system: prefetch and cache appropriately

- increase disk & disk array utilization
- increase cache hit ratio

What's the Benefit?

TIP stall-time as a fraction of OSF/1 stall-time

(average for 7 benchmarks)



Large reduction in I/O stall-time

Hints for Gnuld

Benchmark: load 562 kernel object files

Hints for final link phase:

```
/* load map already generated */

foreach section
    foreach input_file in section
        hint(input_file, text_offset, text_size);

foreach section
    foreach input_file in section
        seek to text_offset in input_file;
        read text_size bytes from input_file;
        patch addresses;
        seek & write to output_file;
```

Hints for Postgres

Benchmark: join of two relations, one indexed

Original code:

```
foreach tuple in outer relation
    look for match in index
    if there's a match
        seek & read match from inner relation
```

After loop splitting:

```
foreach tuple in outer relation
    look for match in index
    stash file_offset of match in an array

hint (file_offsets, block_sizes from stash array)

foreach tuple in outer relation
    if there's a match in stash array
        seek & read match from inner relation
```

Carnegie Mellon

Parallel Data Laboratory

What Does the File System Do with the Hints?

- Prefetch deeply
- Cache hinted data for reuse

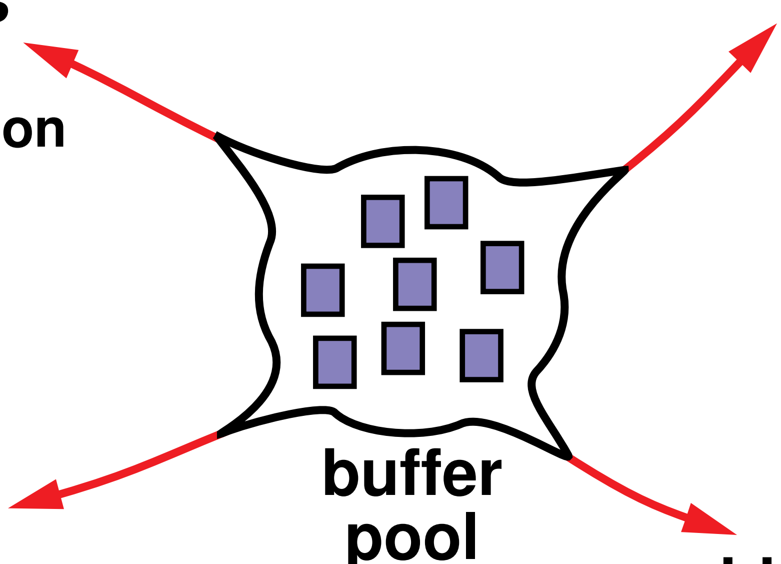
But, what of standard uses for cache buffers?

- Servicing demand misses
- LRU caching for unhinted accesses

The Tough Problem: Cache Management

demand miss
service I/O for stalled application

LRU cache
cache data for unhinted accesses



prefetch data
read hinted data to shorten stalls

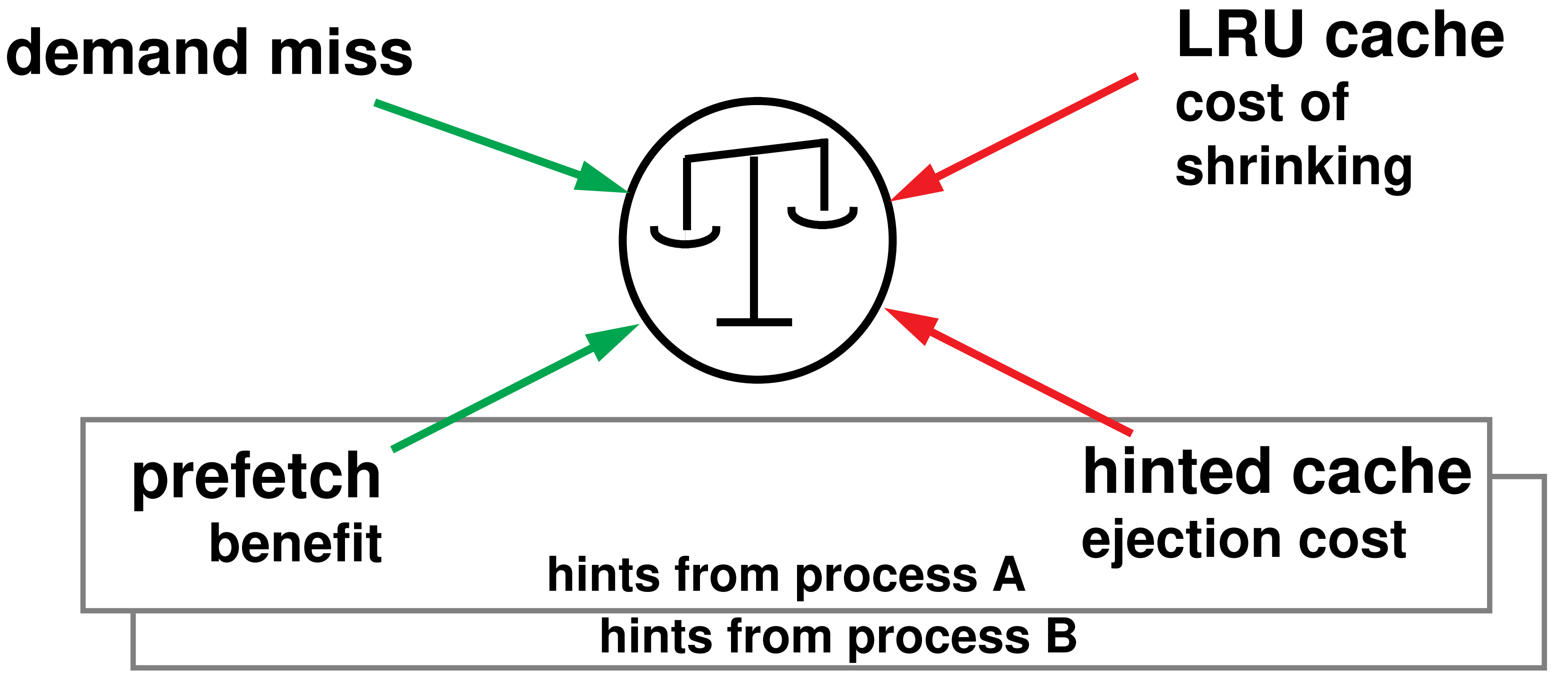
hinted cache
hold on to blocks for hinted reuse

Resolve contention for the limited buffer resource

Our Approach

- Recall: goal is to reduce file read latency
- Want to allocate buffers to reduce latency
- Dynamically estimate impact on latency
- Allocate for greatest latency reduction

Cache Management by Cost-Benefit Analysis



1) **Estimate:**

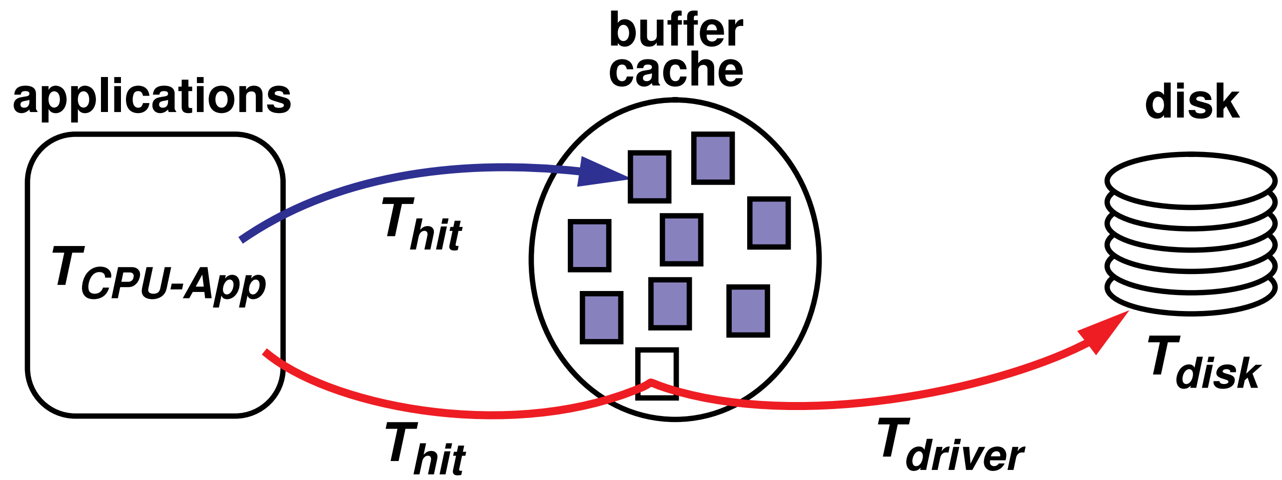
- *benefit* of using a buffer to initiate I/O
- *cost* of taking a buffer from a cache

2) **Reallocate when *benefit* > *cost***

Key Elements

1) Independent cost/benefit estimates
   - much simpler than a global optimization
2) Gold-standard currency
   - basis for comparison of estimates
3) Allocation algorithm
   - use estimates to allocate buffers

Key Element: Independent Estimates



$$T_{miss} = T_{hit} + T_{driver} + T_{disk}$$

- **Benefit of deeper prefetching is shorter stalls**

  $T_{stall}(x+1) - T_{stall}(x)$

  $T_{stall}(x) = 0$, for x > prefetch horizon

- **Cost of shrinking LRU cache is more misses**

  $(H(n)-H(n-1))\ (T_{miss}-T_{hit})$

- **Cost of ejecting hinted block is re-reading it**

  $T_{driver} + T_{stall}(x)$

Carnegie Mellon

Parallel Data Laboratory

Key Element: Gold-Standard Currency

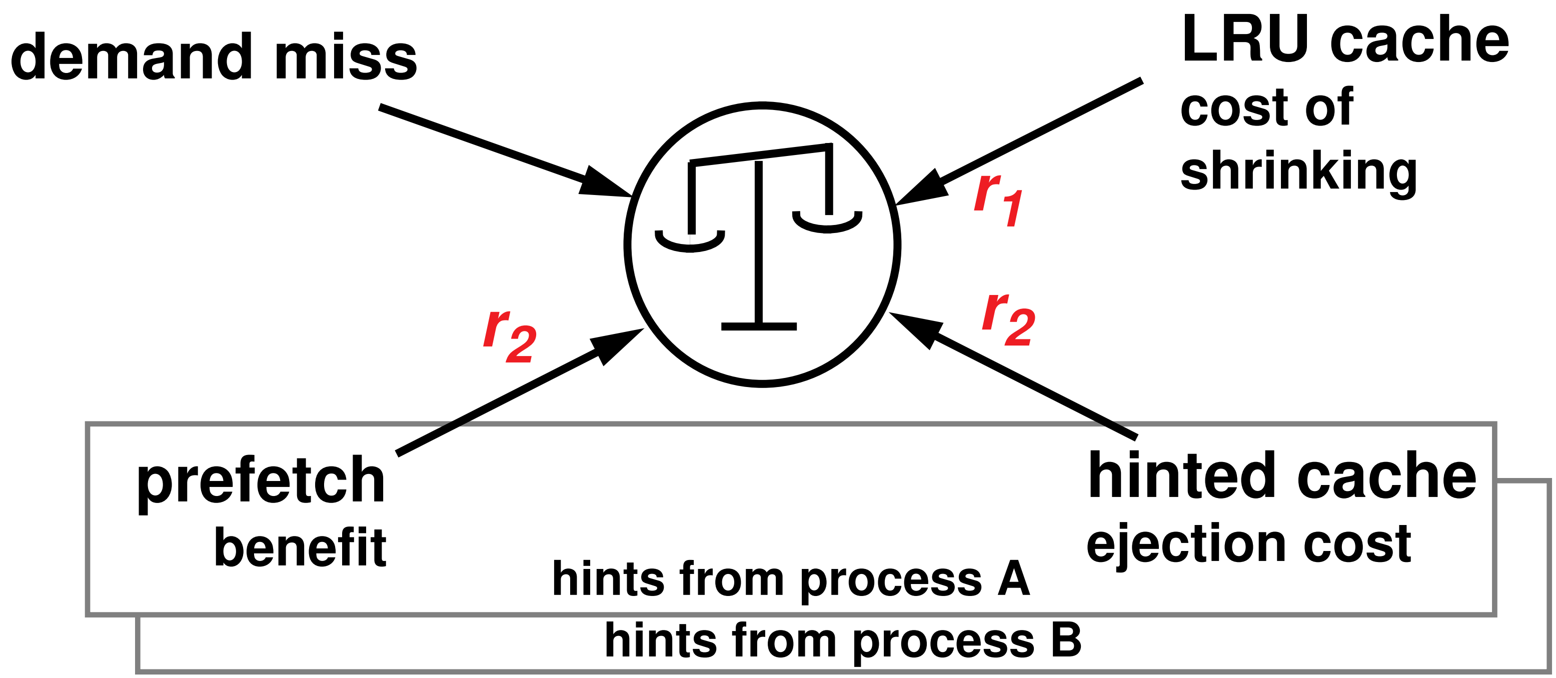
- **Basis for comparison of estimates is:**

Change in I/O service time per buffer-access

- **Relates resource usage to system goals**

- **Key to extensibility**
    - **Page demands from VM**
    - **Preferential caching for remote files**

Key Element: Allocation Algorithm



- Scale estimates by access rates, r_i
- Rank benefits and costs separately
- Take buffers from lowest ranked estimator
- Prefetch for greatest benefit iff benefit > cost

Evaluation

Baseline System

- Digital's OSF/1 with aggressive readahead
- 12 MB file buffer cache (1536 8 KB buffers)
- Software striping driver (64 KB stripe unit)
- Digital 3000/500 workstation

Experimental Systems

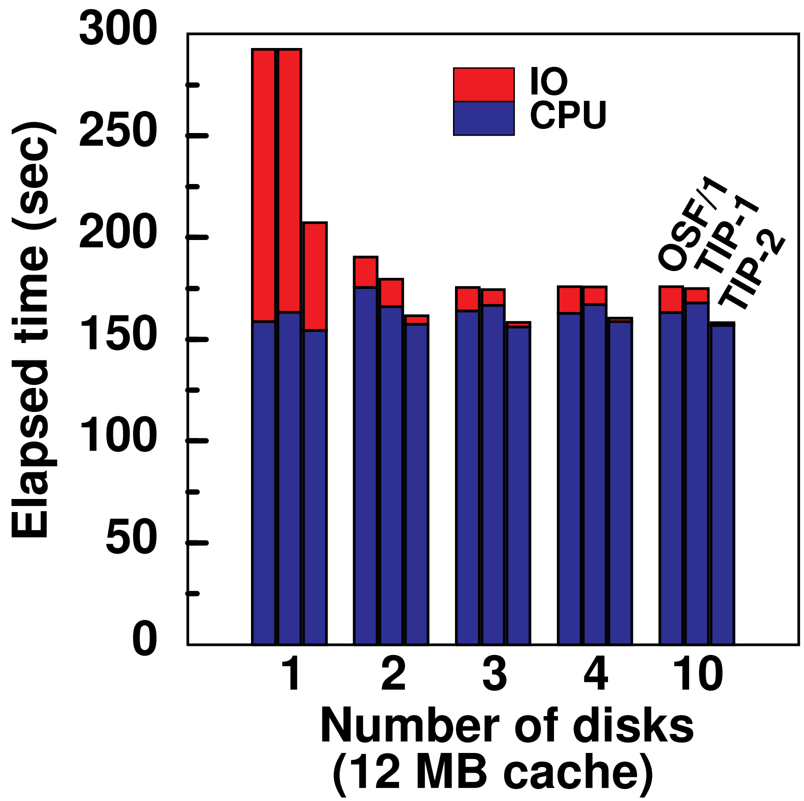
- TIP-1 adds informed prefetching
- TIP-2 has informed prefetching and caching

Application Test Suite

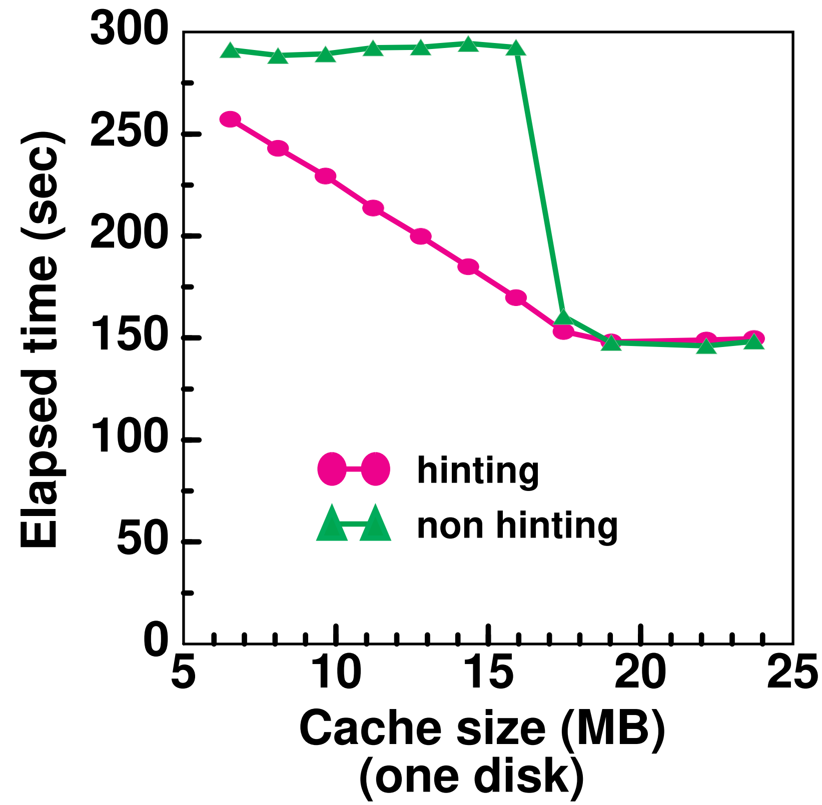
1)	*Davidson*	computational physics
2)	*XDataSlice*	3-D scientific visualization
3)	*Sphinx*	speech recognition
4)	*Postgres*	relational database
5)	*Gnuld*	object code linker
6)	*Agrep*	text search

Davidson Performance

Execution time vs. array size



Execution time vs. cache size

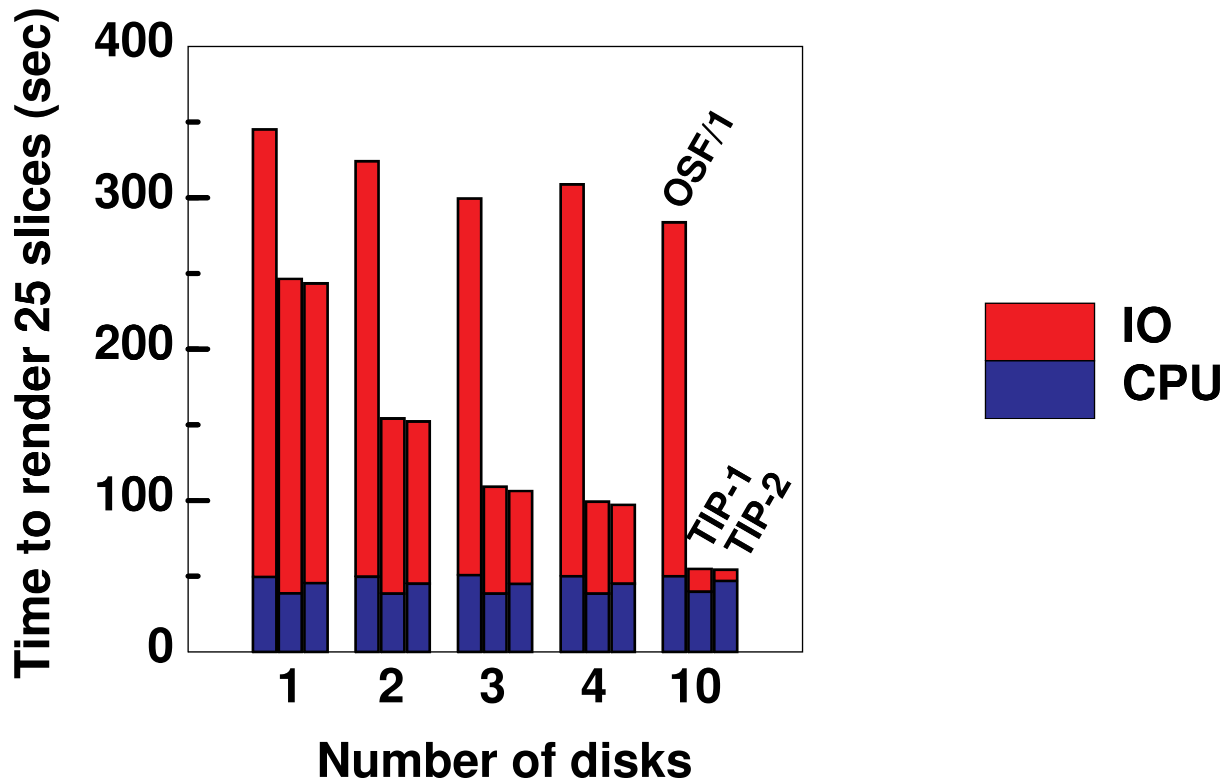


- OSF/1 sequential readahead works well
- Informed caching most important on one disk

Carnegie Mellon

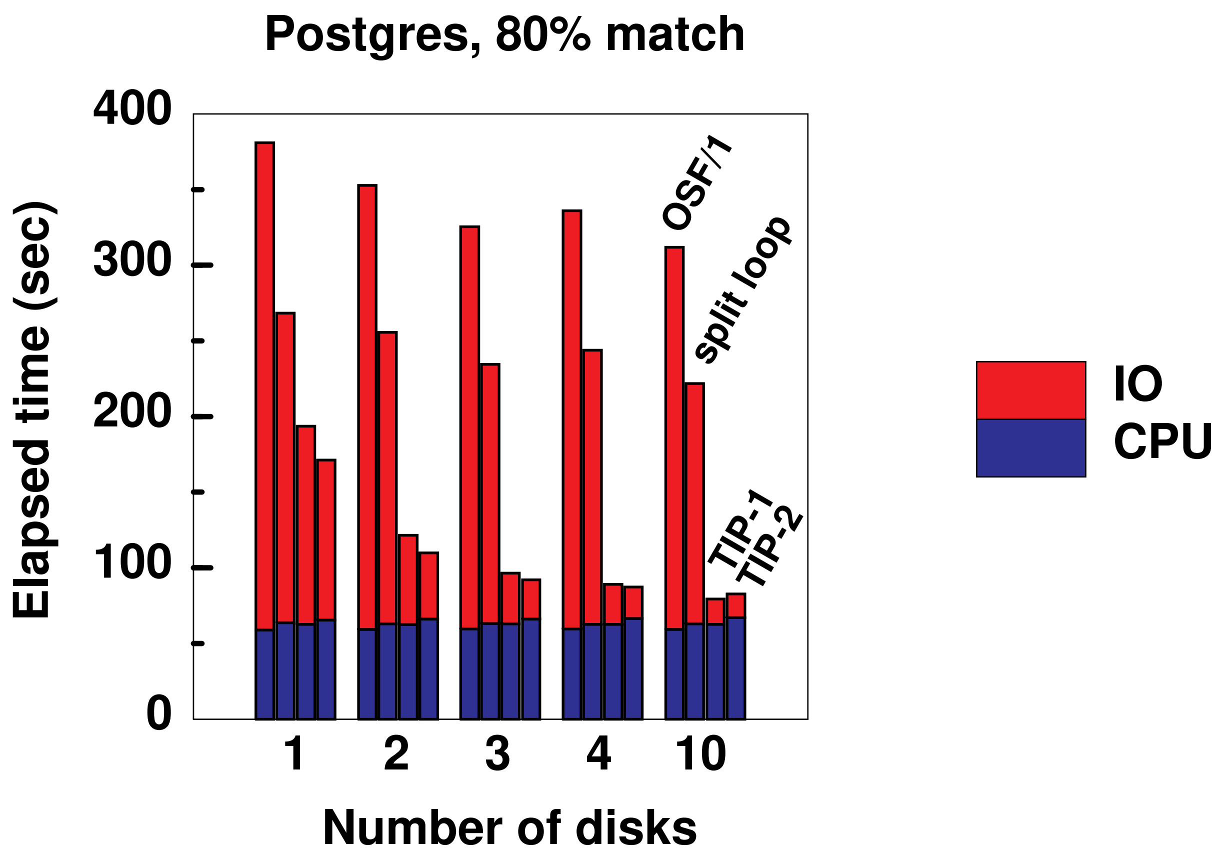
Parallel Data Laboratory

XDataSlice Performance



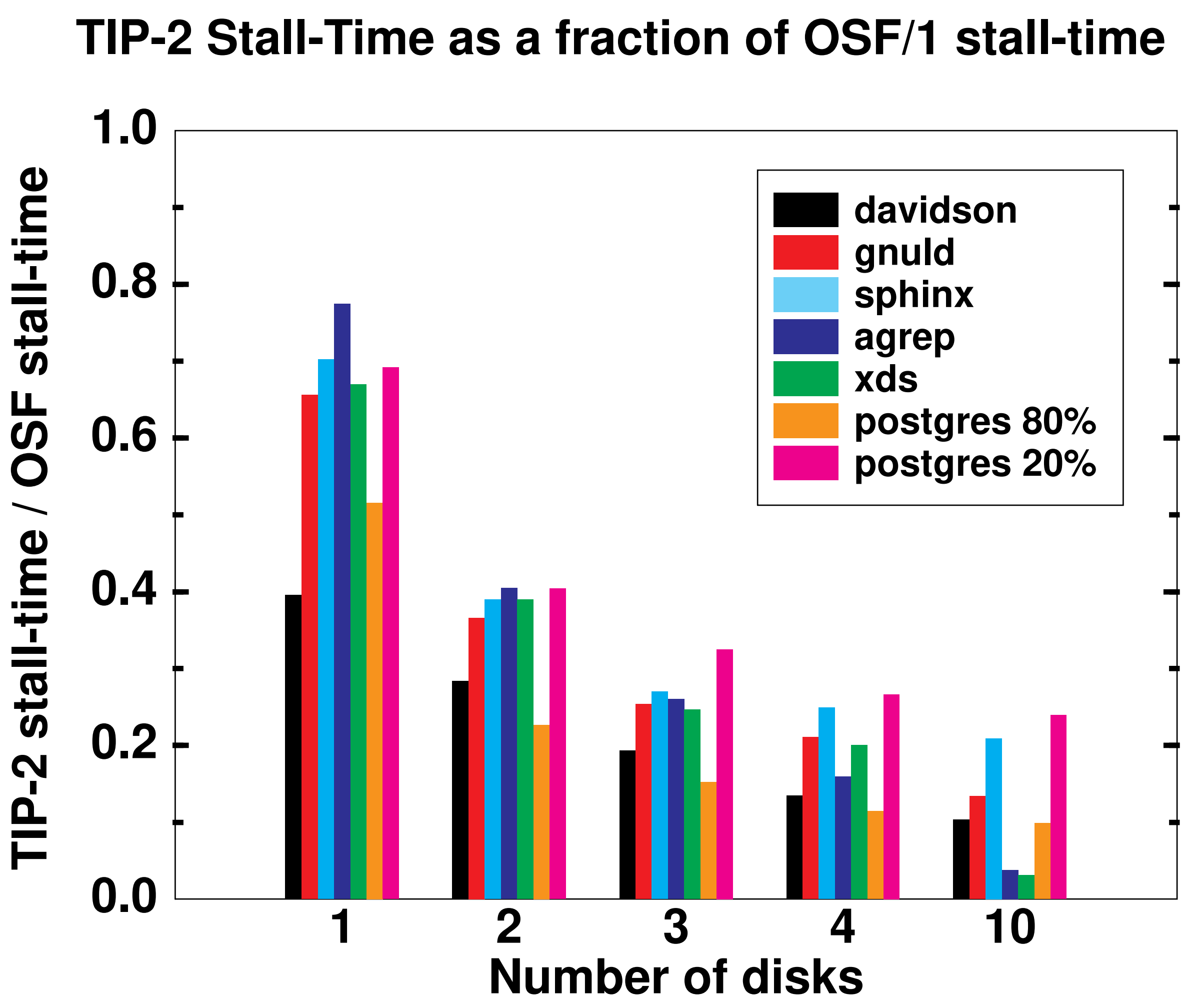
- No benefit from informed caching
- *XDataSlice* needs TIP to exploit disk array

Postgres Performance



- Informed caching most useful on one disk
- Informed prefetching wins most on array
- Three disks enough for most of the benefit

Performance Summary



Carnegie Mellon

Parallel Data Laboratory

Conclusions

- Resource management based on dynamic cost/benefit estimates works
- Informed prefetching → concurrent I/O
- I.P. delivers big wins even from small arrays
- Informed caching → fewer I/Os
- I.C. wins most when bandwidth is limited
- Combination increases throughput when multiprogramming