Network-Attached Secure Disks (NASD)

Garth Gibson, Carnegie Mellon Univ.

http://ww.cs.cmu.edu/afs/cs/project/nasd/WWW/NASD.html
http://www.cs.cmu.edu/Web/Groups/PDL/
garth+@cs.cmu.edu

Eliminate file server performance bottlenecks

Eliminate (parallel) client file system bottlenecks

Shore up network-attached storage security

Enable intelligent storage direct access to client

Executive Summary

Scalable storage based on aggregation

- switched networks and mesh interconnects
- storage striping

Traditional file server is a bottleneck

- stop & copy forwarding from peripheral net to LAN

Network-attached storage removes bottleneck

- transfers involve only client, network, storage

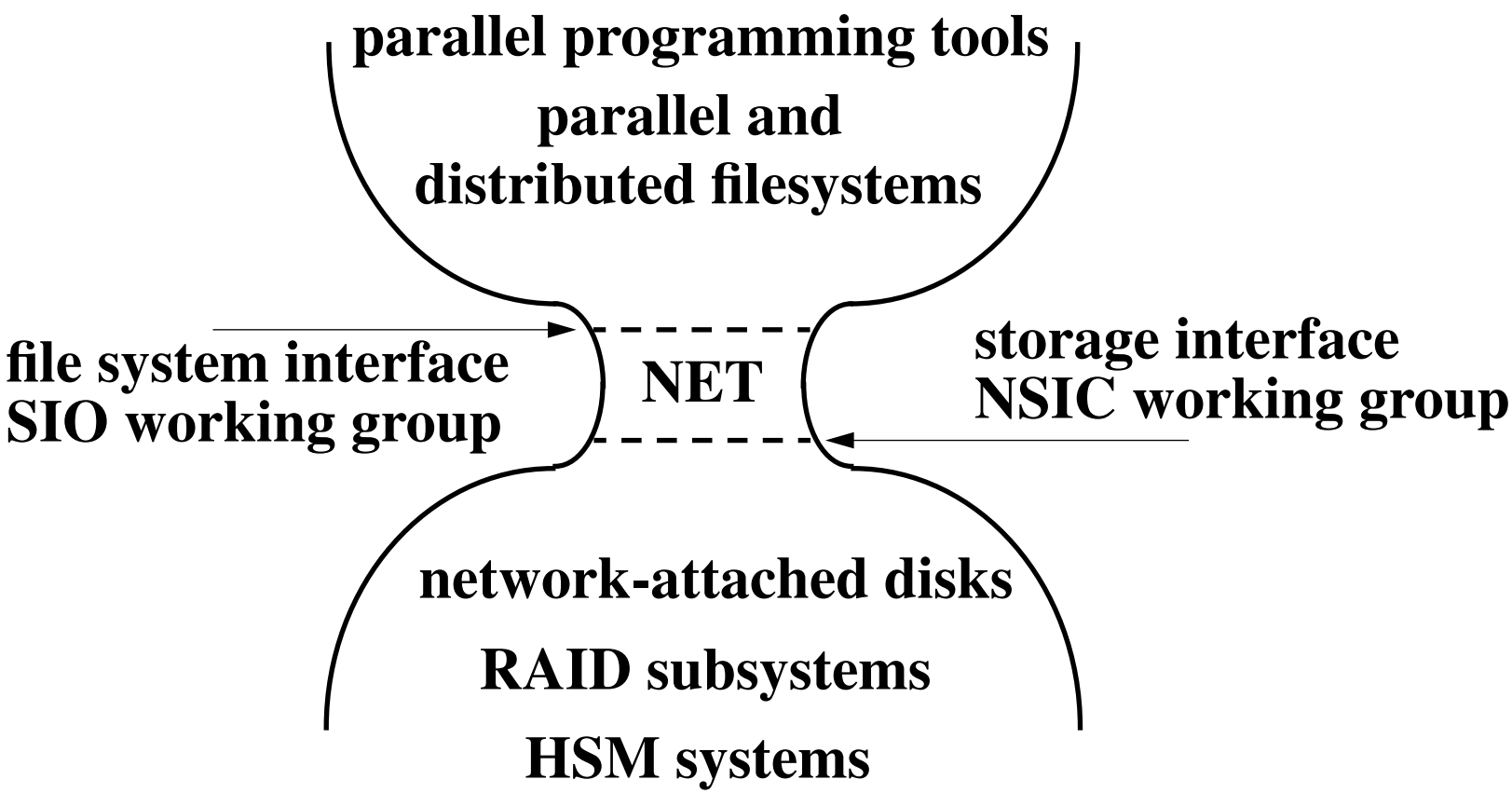
Client file system API has bottlenecks too

- SIO parallel FS low-level API eliminates copying

NASD exposed to insecure network

- NASD participates in LAN authentication protocol

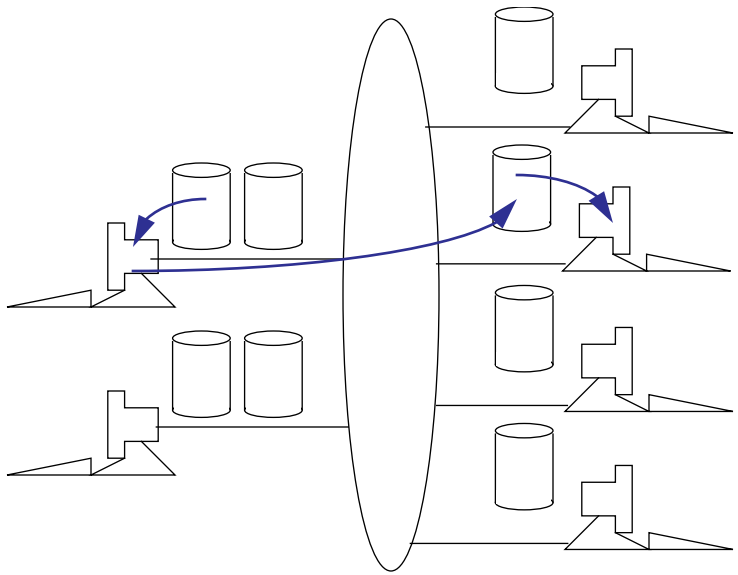
Strategy: only efficient networking between app and storage



Trends: Data bytes travel over LANs

Workstation/compute node a poor & costly server

- designed around caching for processor-local work
- induces extra copying
- network bandwidth often close to single disk bandwidth

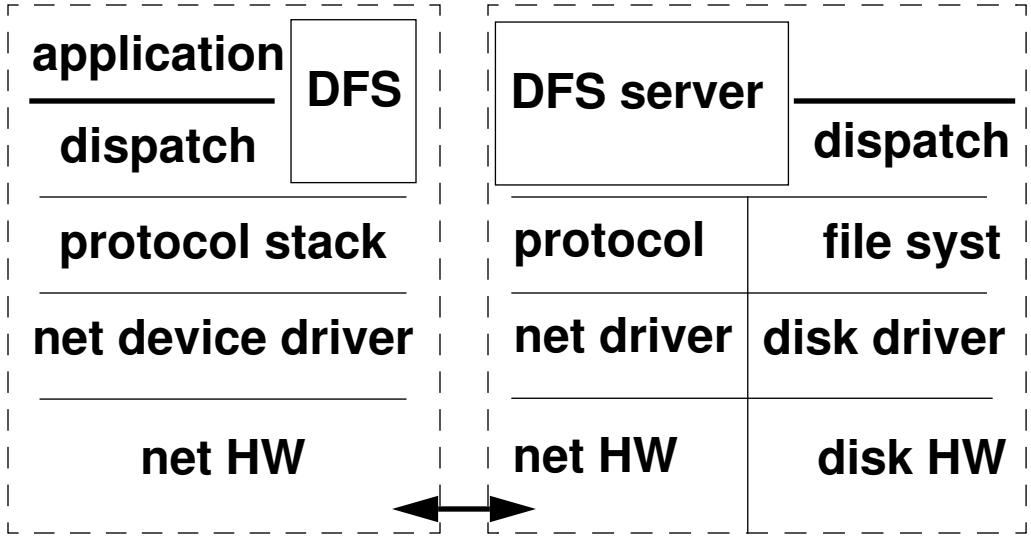


Today's Distributed File System Abstraction

Client-server partition of local file system

Server emphasizes:

- **scale # clients/server**
- **global file naming**
- **"efficient" bulk transfer**
- **partial failure modes, disconnected operation**
- **expensive state mgmt: atomicity, consistency, accounting, administration**



Trends: Storage "hardware" evolving rapidly

Drive data rate rising rapidly (~20% per year)

SCSI physical interconnect evolves to packet LAN

- Firewire, SSA, Fibrechannel

Fast-moving, profit-tight marketplace

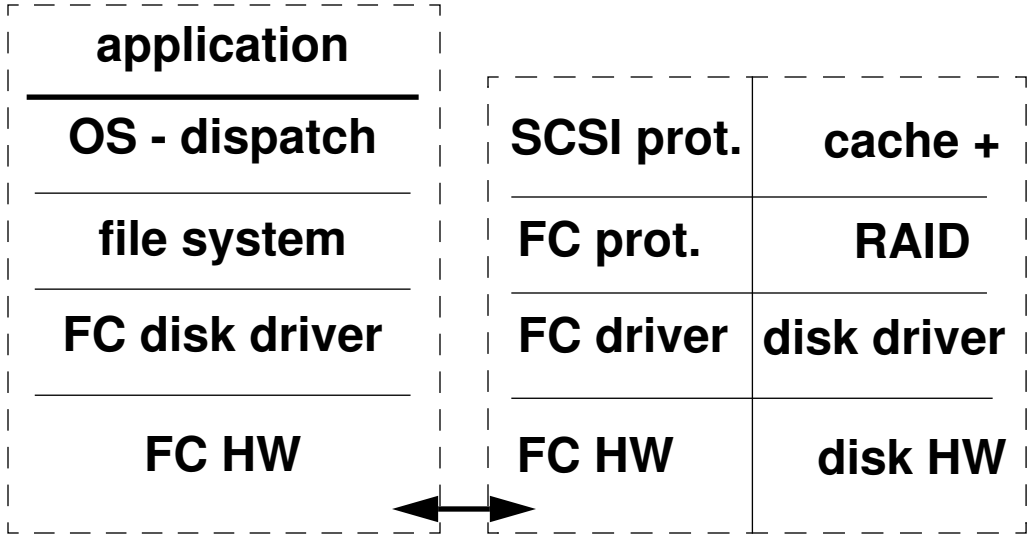
- Compete with value-added intelligence for performance
- Exploit VLSI for more local cycles and bytes
- Geometry sensitive disk scheduling, readahead/writebehind
- RAID support to offload parity update computation
- Cost of managing storage per year is more than storage cost
- Dynamic mapping for transparent optimizations

Local Storage System is Partitioned too

Local RAID disk more than disk arm & R/W head

Storage emphasizes:

- **fast interconnect: e.g. Fibrechannel, SSA**
- **power: e.g. subsystem embeds capabilities of workstation computer**
- **packetized SCSI is transparent to file system**
- **"cache +" includes dynamic remapping, compression, etc.**

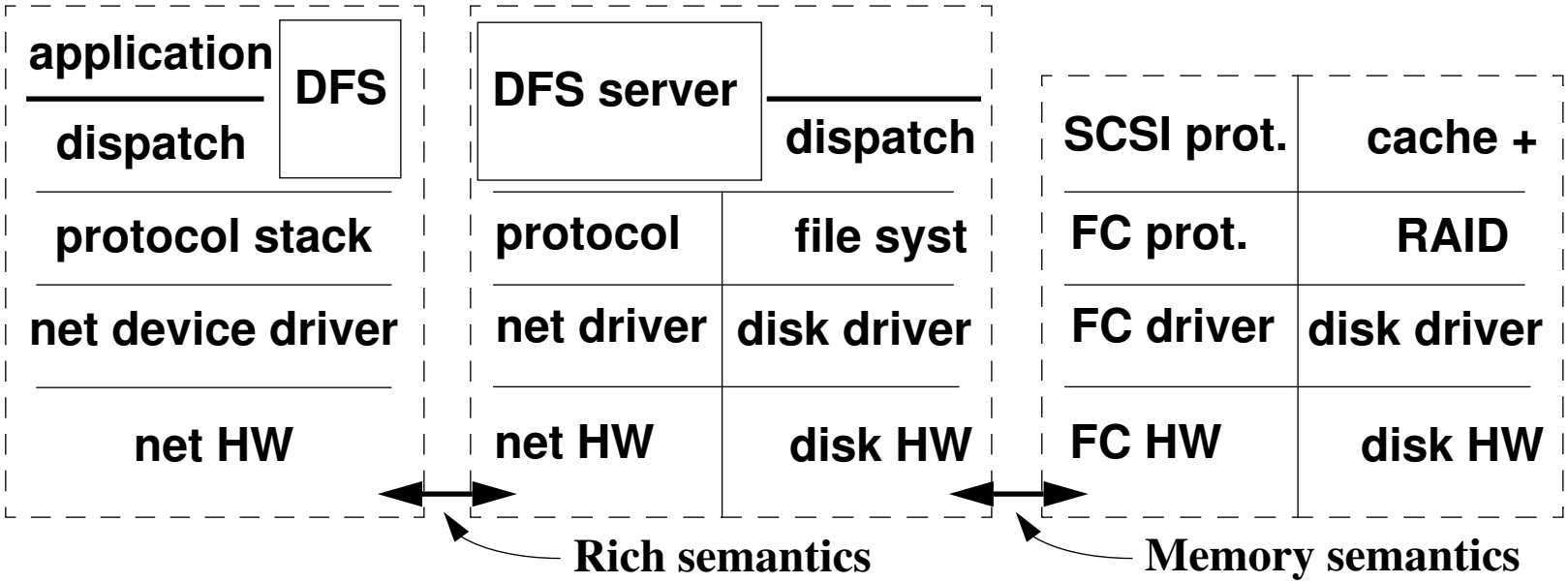


Storage System Structure Revisited

Storage subsystem limited by file server

- higher bandwidth
- better cost-performance
- interoperates more widely

Too many protocol layers, too much copying

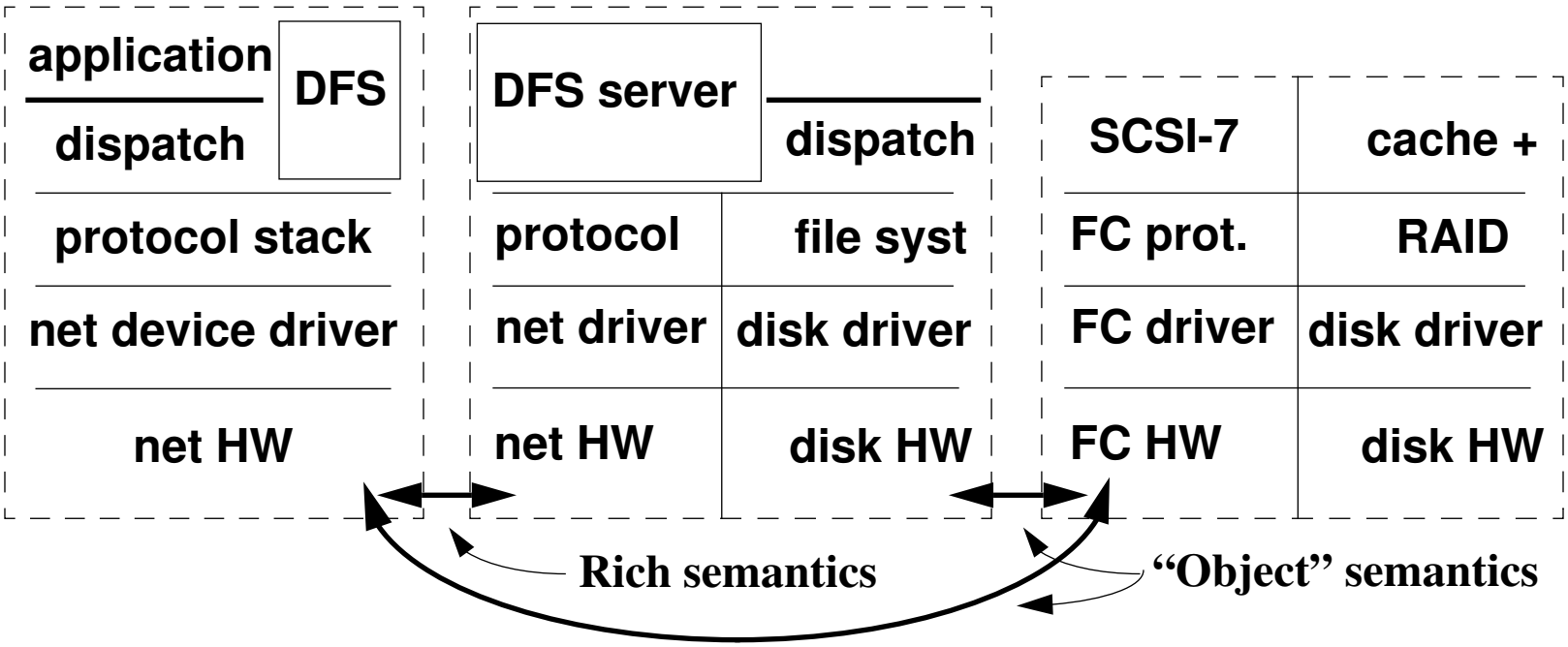


Network-Attached Storage System Structure

Fastpath common operations (read, write, open)

Raise storage subsystem interface to objects

Network "DMA" (copy, scatter/gather)



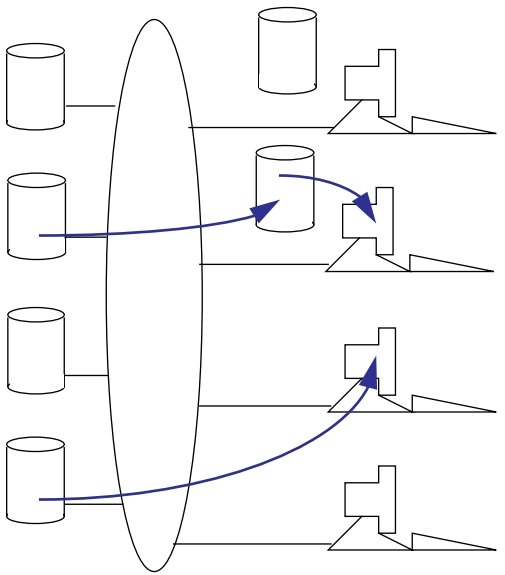
DFS, NASD Functional Partition

Distributed file system remaining functions

- names interpretation and administration
- concurrency and consistency semantics
- access control interpretation and accounting

NASD function

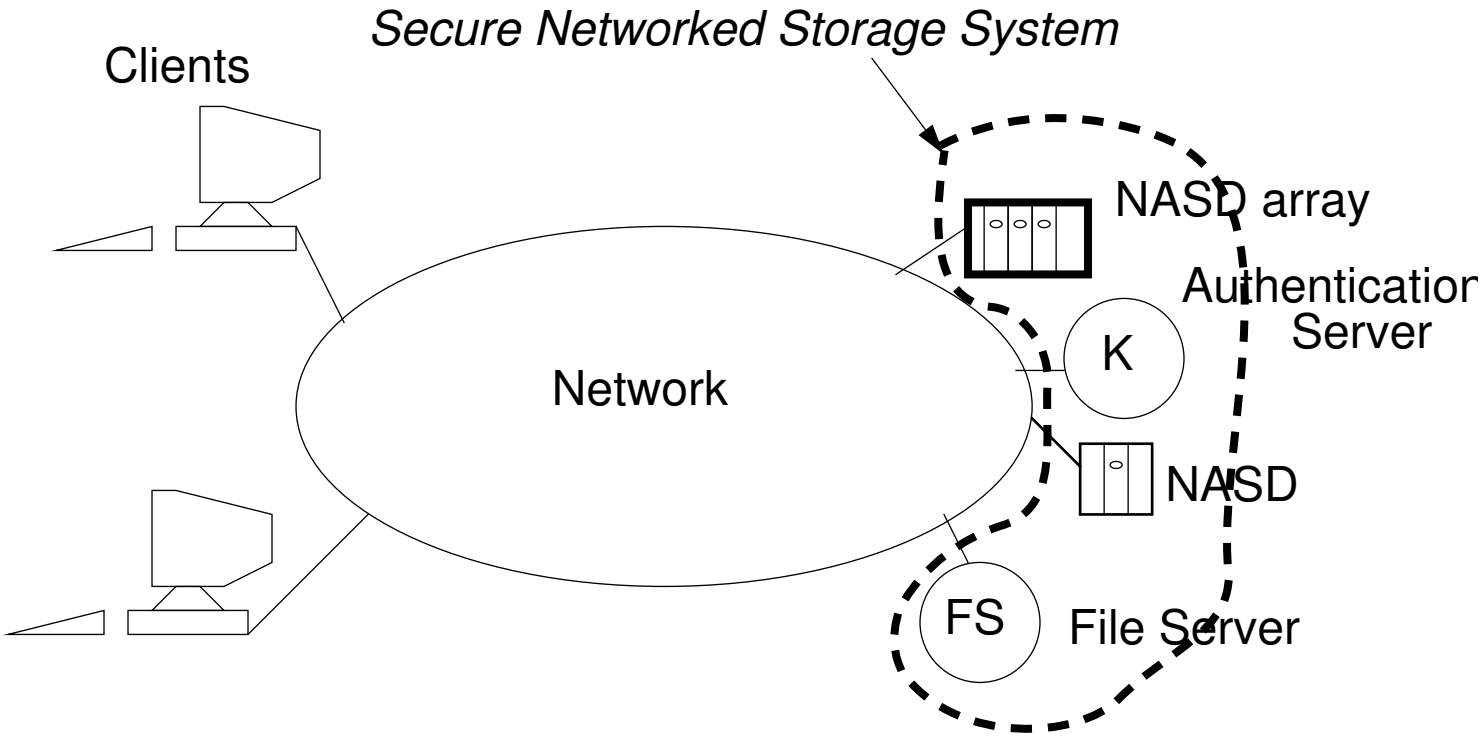
- object storage metadata
- common command processing
- efficient network transfer
- access control enforcement
- storage value-added



NASD Security Model

Arbitrary packets must not be seen as commands

- **private key cryptography; kerberos standard**
- **tamper-resistant encryption chip on drive**



NASD Security Protocol Goals

Security goals:

1) No unauthorized modification of data

2) No unauthorized disclosure of data

3) No unauthorized allocation of resources

4) Limit threat caused by a compromised machine

• Performance goals:

- Minimize number of messages
- Direct request/reply between client and drive

• Flexibility goals:

- Protocol unaware of filesystem implementation

How to Exploit NASD Interface

For existing distributed file systems

- reduce file server load improves cost-performance, scalability
- fixed client filesystem API limits application benefit

For new parallel file systems

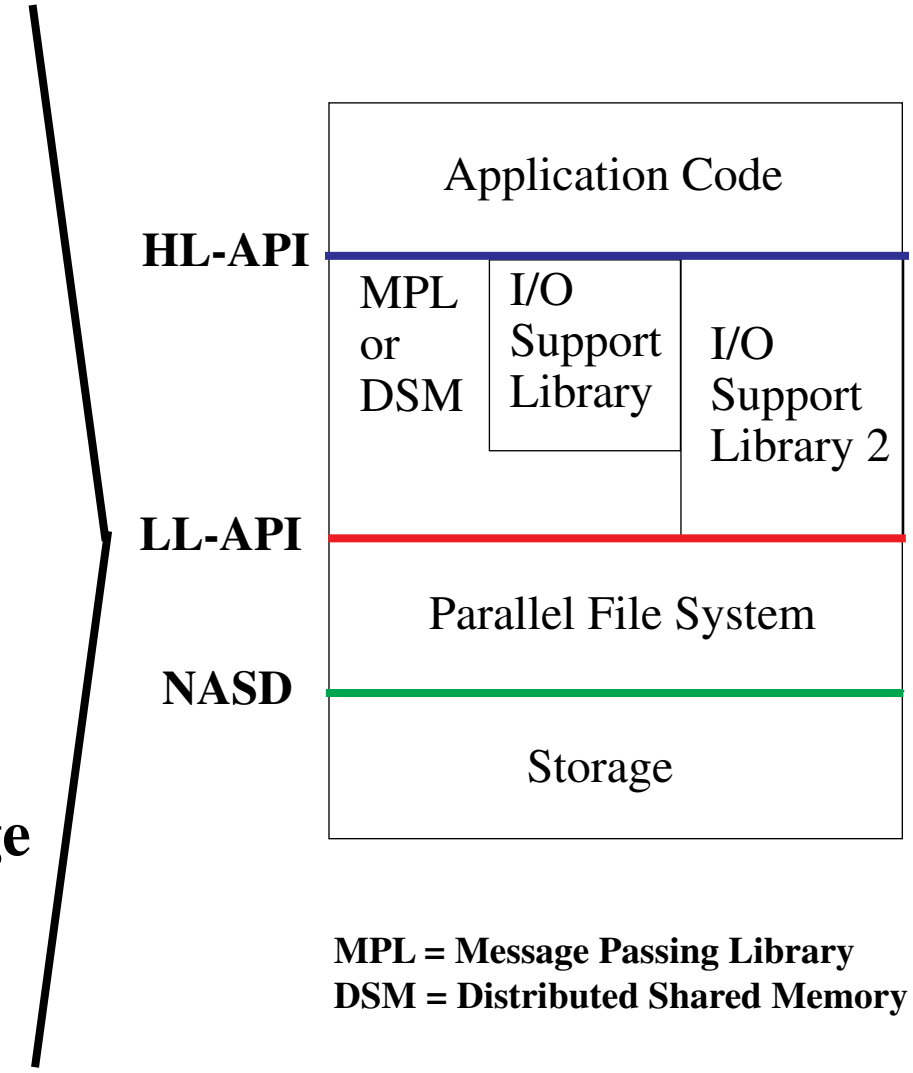
- parallel FS structures/APIs still developing
- Scalable I/O developing new high-performance standards

SIO focus on low-level API design for client

- portable, flexible, minimal number of functions
- minimal constraints on parallelism, efficiency
- applicable to MPP and LAN-based multicomputer

SIO Parallel File System Low Level API

- **Scatter/Gather**
- **Copy Avoidance**
- **Collective Transfer**
- **Copy-on-write**
- **Client cache control**
- **Hints to/from storage**
- **Asynch**



MPL = Message Passing Library
DSM = Distributed Shared Memory

Exploiting NASD: Network Support for Parallel Flows

Switch-style scalable storage transfers in multiple streams

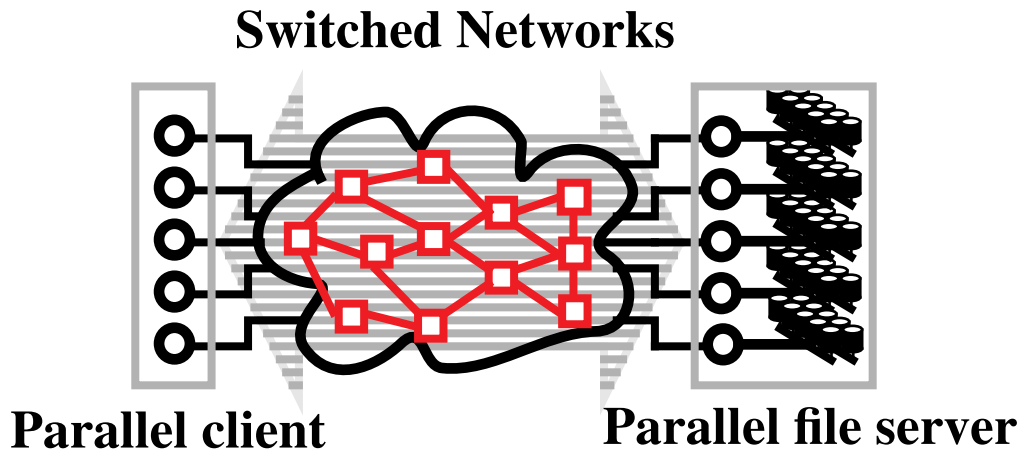
- but networks deal in individual connections

Network support for coordinated routing of multiple streams

- multi-path connections, source routing, load-sensitive

API for negotiating parallel flow service

- enable applications to adapt to bandwidth availability



NASD Summary

Network-attached Secure Disks (NASD)

- **“first class network citizens”**
- **direct data transfer to client**
- **eliminate workstation file server bottleneck**
- **raise storage interface to object storage system**
- **integrate with authentication-based security**
- **repartition file system between client and storage**
- **support multiple client filesystem personalities**
- **network support for storage striped transfers**
- **long-range push to cost-constrained, transfer-oriented drives**

Exploiting NASD Interface in Storage Subsystem: Informed Prefetching and Caching

with R. Hugo Patterson
Eka Ginting, Dan Stodolsky, Jim Zelenka

Bypassing client OS, use object interface for aggressive prefetching and caching

Reduce file latency, exploit storage parallelism

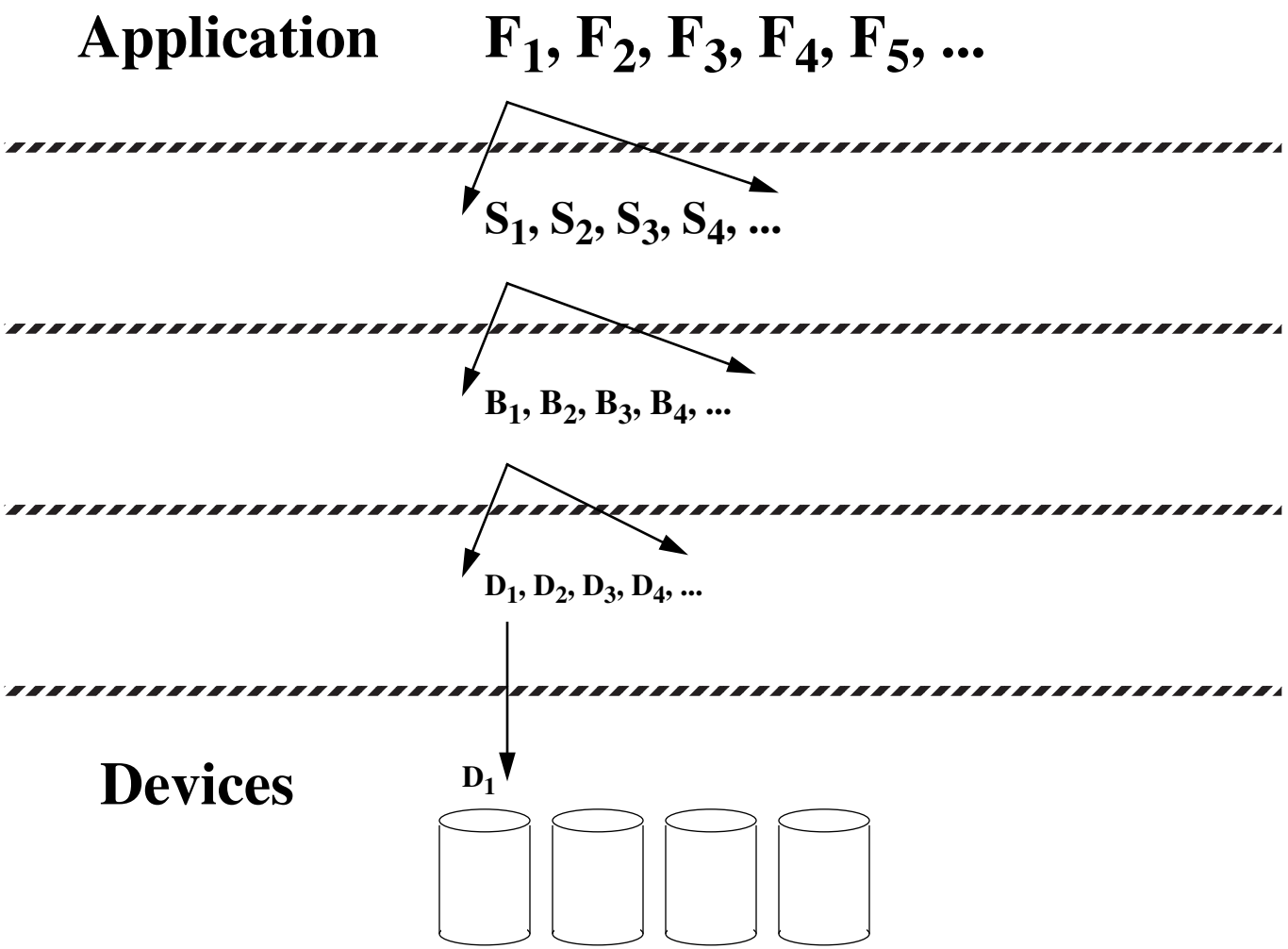
First, extract access pattern hints

Second, hints guide resource allocation

Experiment: 7 I/O-intensive applications

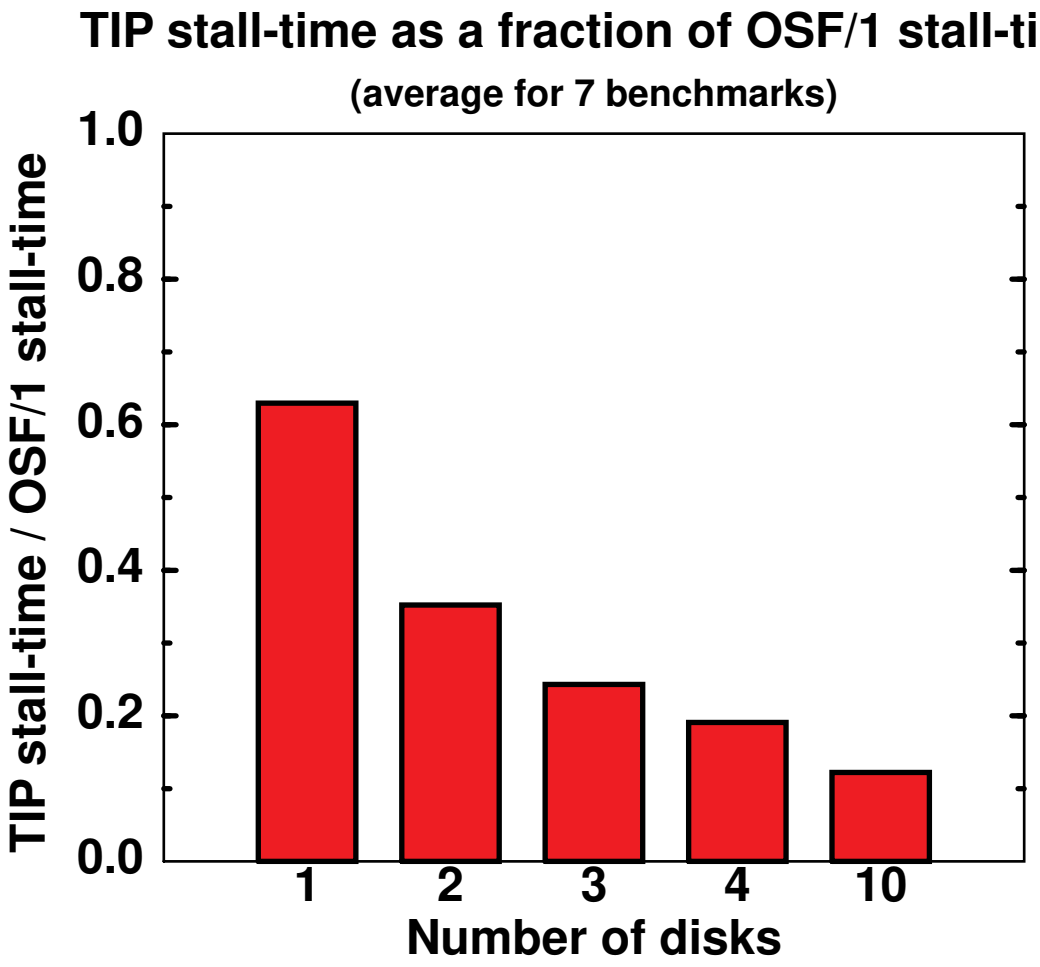
Result: reduce stall time by up to 83%

Combatting the Application Disclosure Bottleneck



- **Use hints to inform system of future accesses**

What's the Benefit?



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

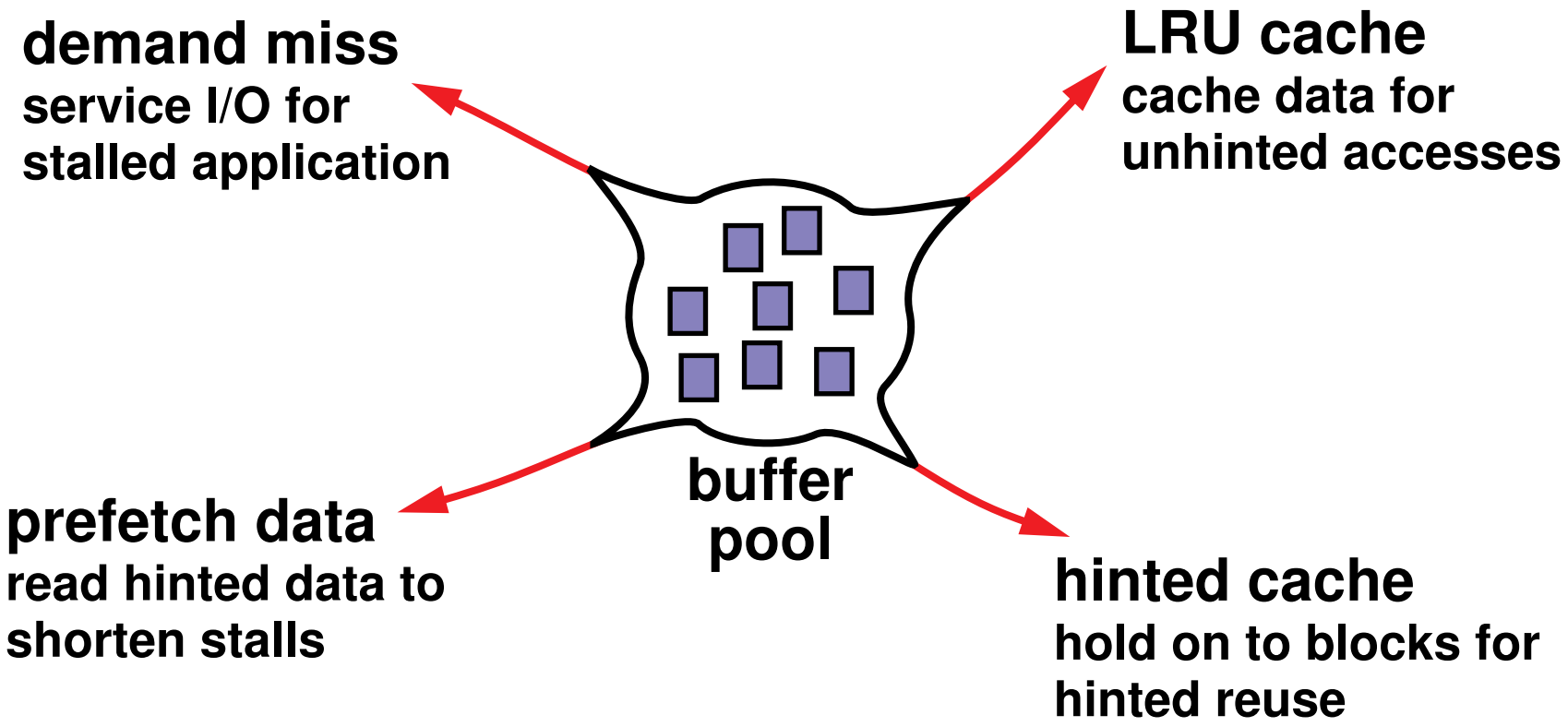
After loop splitting:

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)

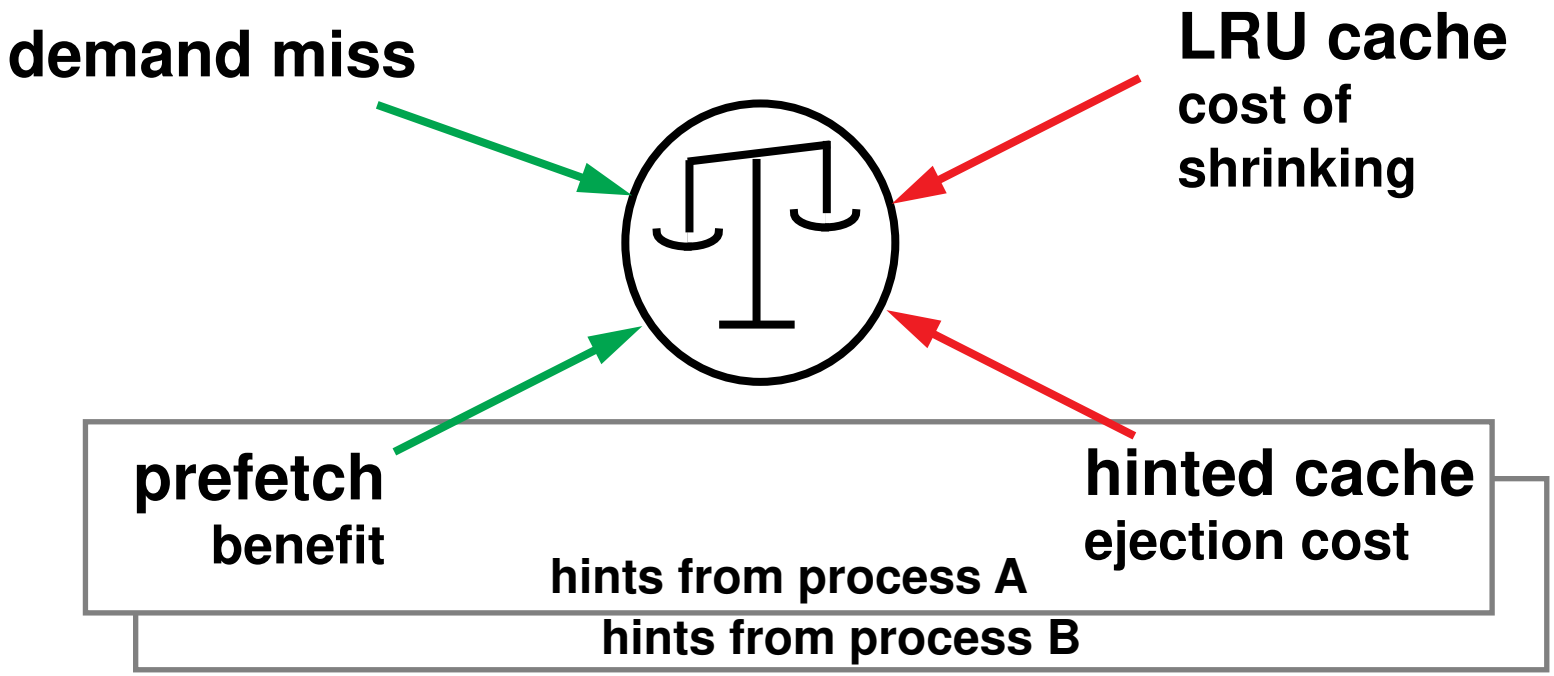
foreach tuple in outer relation
    if there's a match in stash array
        seek & read match from inner relation

The Tough Problem: Cache Management



Resolve contention for the limited buffer resource

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*

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

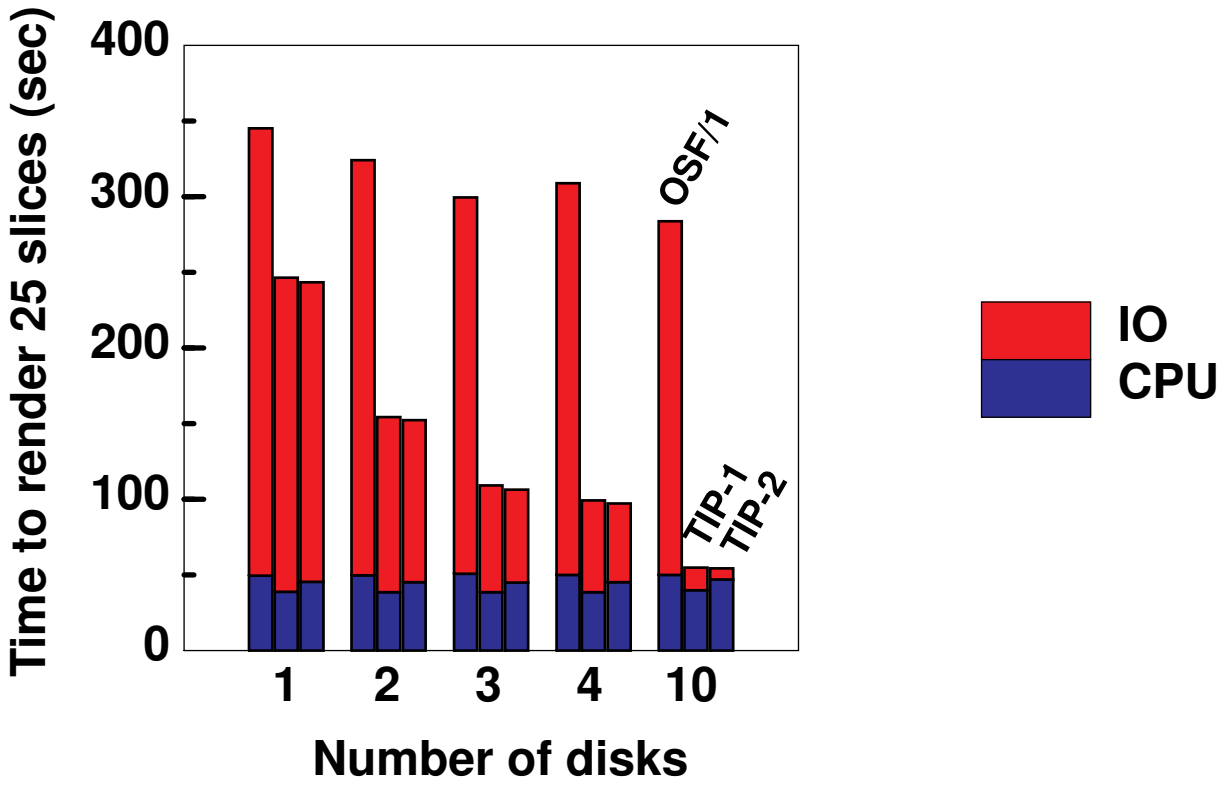
Experimental Systems

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

Application Test Suite

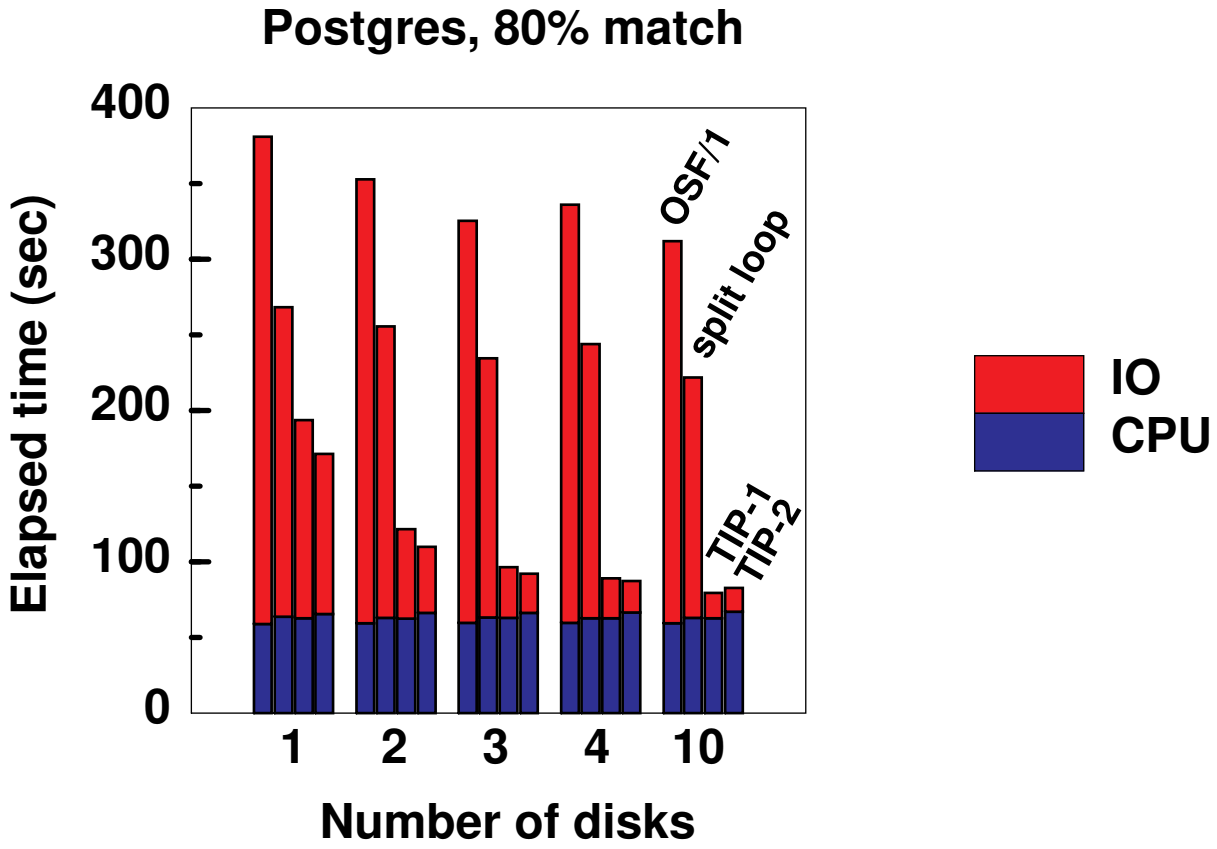
1) *Davidson* computational physics
   - sequentially reads 17 MB file 61 times
2) *XDataSlice* 3-D scientific visualization
   - reads arbitrary slices from 500 MB dataset
3) *Sphinx* speech recognition
   - 200 MB sequential read, many random reads
4) *Postgres* relational database
   - sequential read of 3.2 MB relation
   - random reads of index and data of 32 MB relation
5) *Gnuld* object code linker
   - multiple passes reading portions of 562 files
6) *Agrep* text search
   - sequentially reads 1349 small files

XDataSlice Performance



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

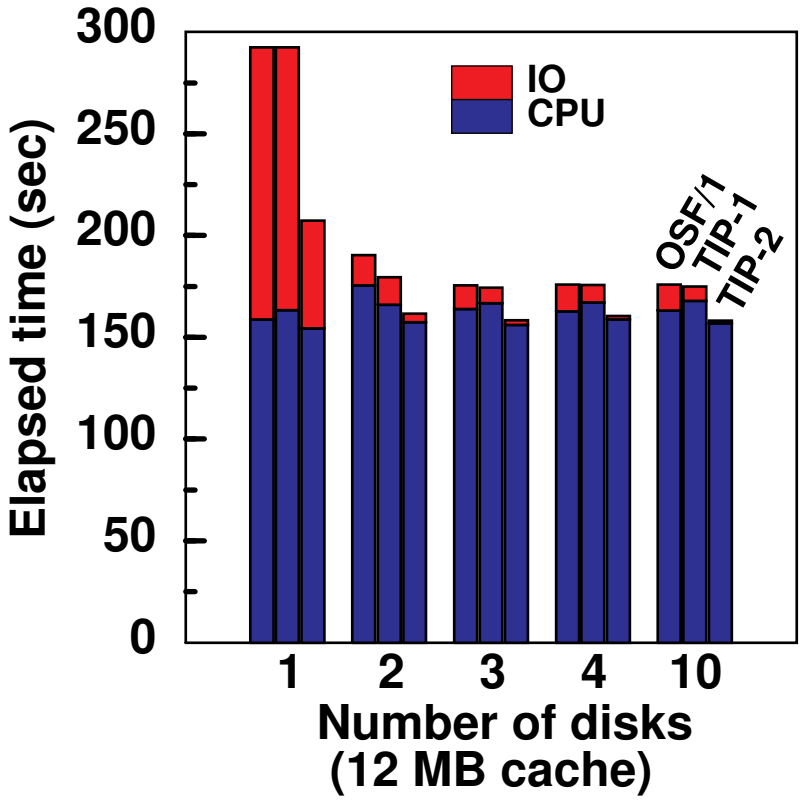
Postgres Performance



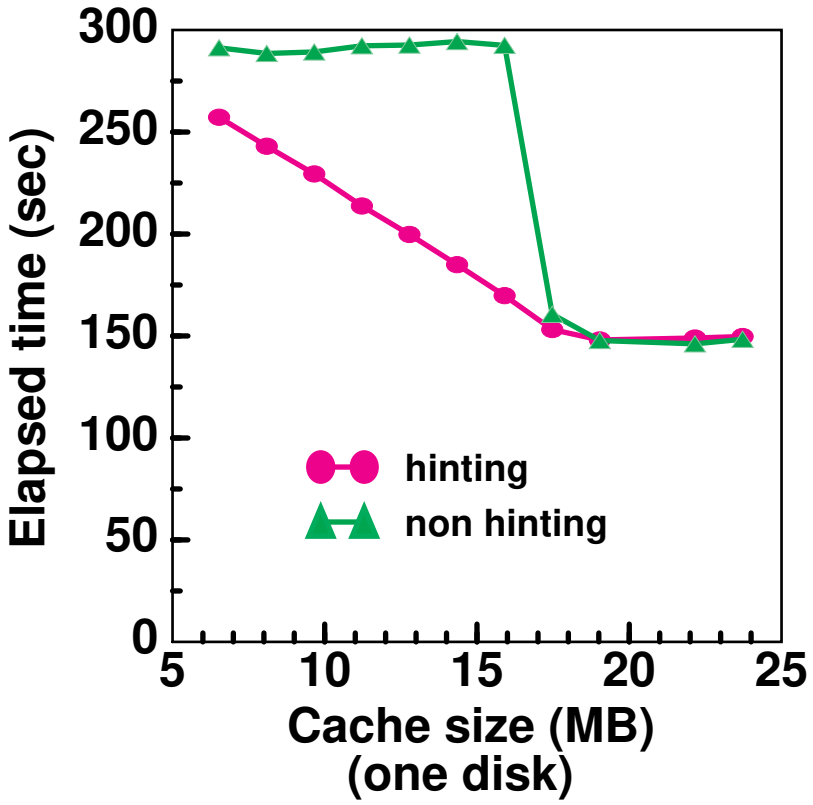
- Informed caching most useful on one disk
- Informed prefetching wins most on array
- Three disks enough for most of the benefit

Davidson Performance

Execution time vs. array size

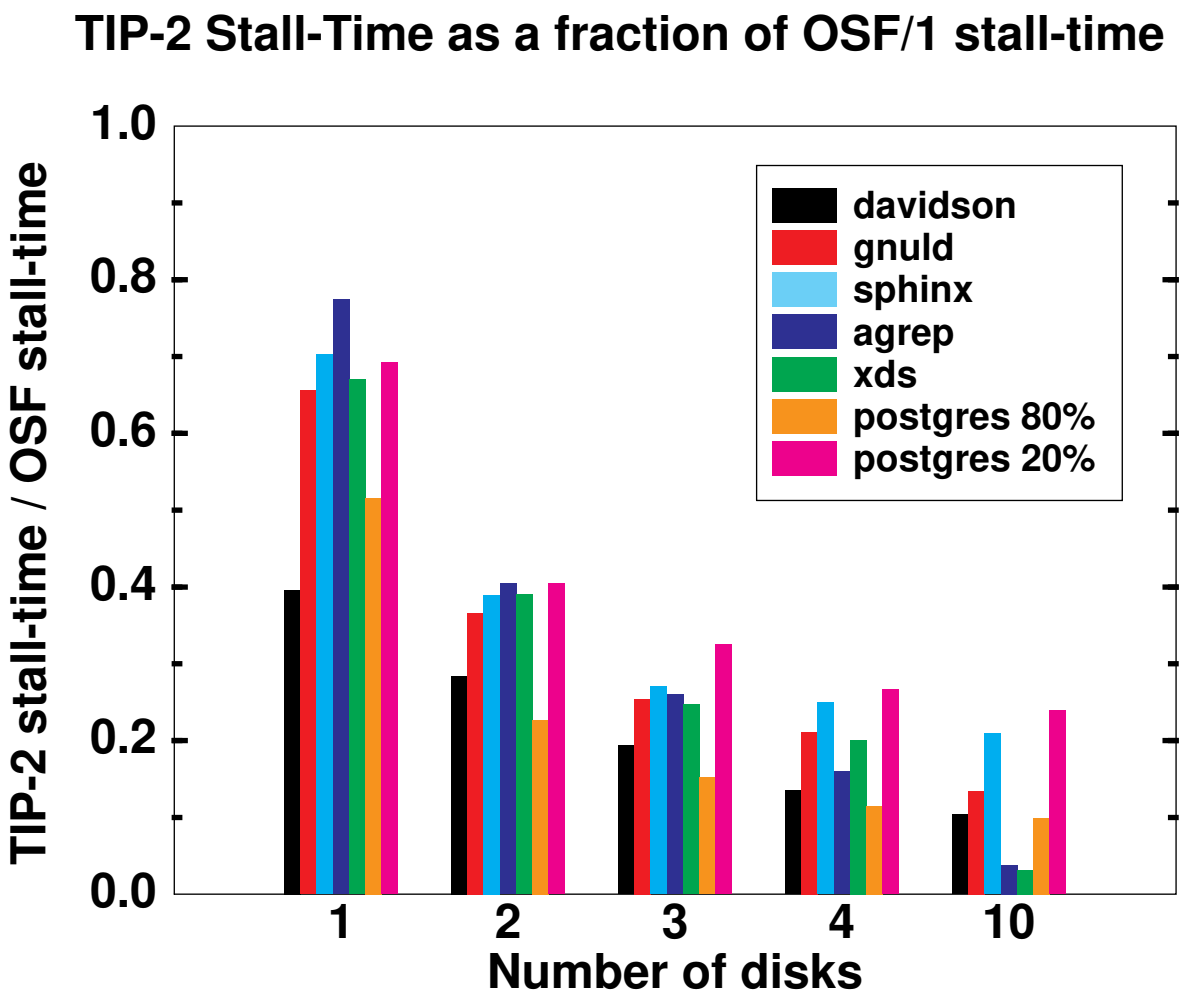


Execution time vs. cache size



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

Performance Summary



Informed Prefetching and Caching: 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

Current and Future Directions

Automatic extraction of disclosure

- context-dependent access pattern learning
- compiler extraction for out-of-core scientific codes

Integrate with parallel and distributed file system

- global management of server and client cache space

Integrate with VM management

- prepage predictably accessed memory objects

Non-homogeneous and network devices

- non-uniform prefetch depth to avoid hot spots

Proposal for NSIC NASD collaborative research

Garth Gibson, CMU
May 31, 1995

Not focussed on fund raising

Pre-competetive techn. = pre-standards interfaces

NSIC working group exchange forum

Funding our research

NSIC as a leader of government proposal

- ATP money evaporating
- industry resource commitment inconvenient

Participant efforts independent of NSIC funding

NSIC research = pre-competitive technology

- phyics, chemistry etc

What is physics/chemistry of computer systems?

- layering? management policies? applications?

Focus on Pre-standards effort

Standards is known method of collaboration

- agreement significant enabler of effective competetion

Proposed work not mature enough for standards

- implications on commodity markets
- interactions with complex, formerly independent systems

Focus on proof-of-concept demonstrations

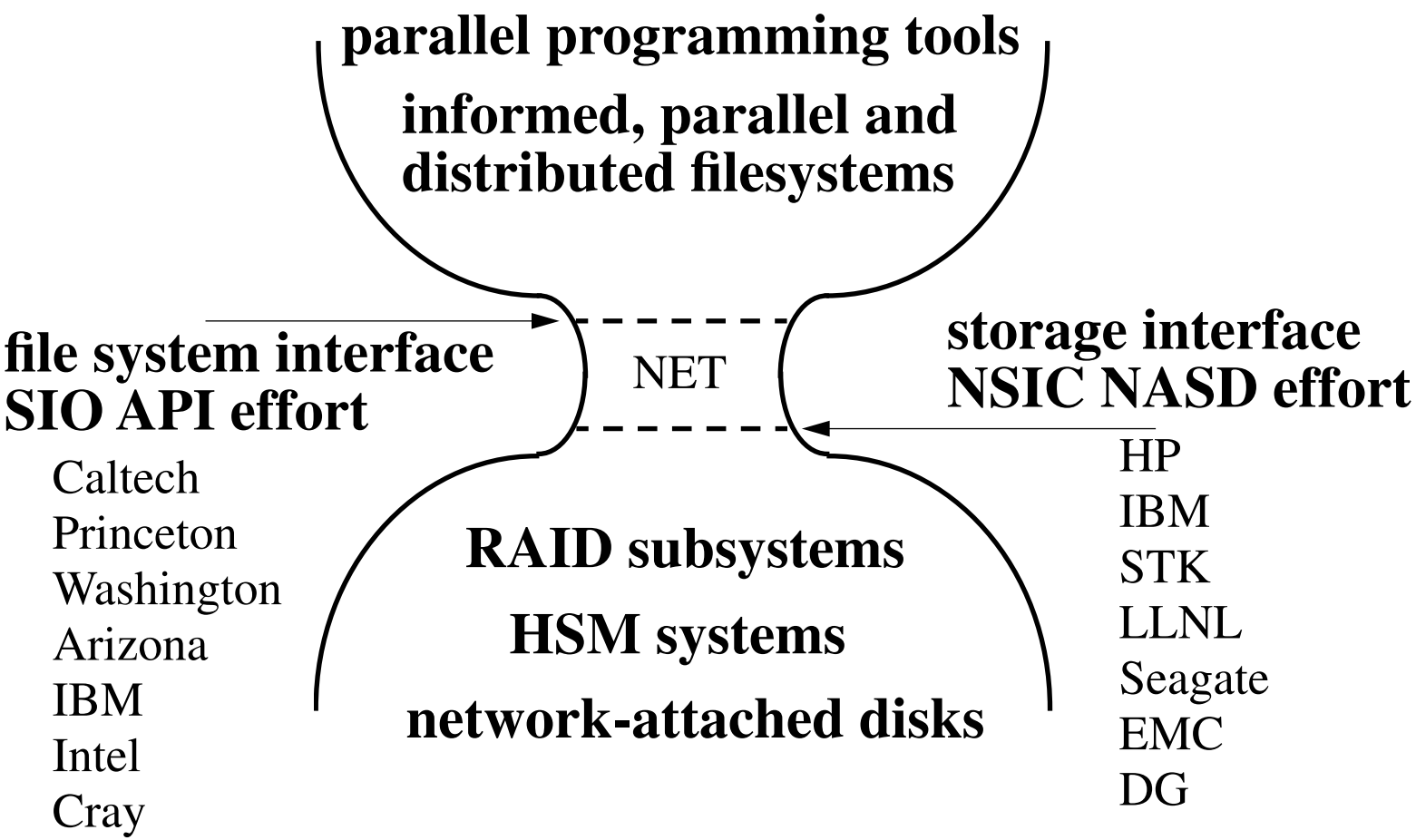
- analytic and simulation modeling
- experimental prototypes
- application suitability case studies

NSIC working group

Discussion forum

- regular meetings encourage regular progress
- demand evidence to influence architectural model
- modeling/experimental parameters/benchmarks
- draw in impacted organizations

Manipulation of public research program

- scrutinize and direct university efforts
- campus site for extended collaborative visits

CMU Parallel Data Lab's Overall Strategy