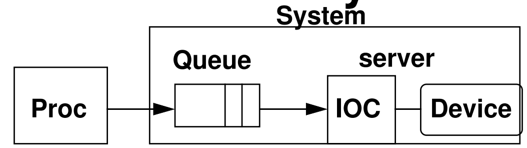
Lecture 13: I/O: RAID, I/O Benchmarks, UNIX File System Performance

Professor David A. Patterson

Computer Science 252

Fall 1996

Summary: A Little Queuing Theory



- **Queuing models assume state of equilibrium: input rate = output rate**
- **Notation:**

r	average number of arriving customers/second
T_{ser}	average time to service a customer (tradtionally $\mu = 1/ T_{ser}$)
u	server utilization (0..1): $u = r \times T_{ser}$
T_q	average time/customer in queue
T_{sys}	average time/customer in system: $T_{sys} = T_q + T_{ser}$
L_q	average length of queue: $L_q = r \times T_q$
L_{sys}	average length of system : $L_{sys} = r \times T_{sys}$

- **Little's Law:** $Length_{system} = rate \times Time_{system}$
  (Mean number customers = arrival rate x mean service time)

Summary: Processor Interface Issues

- **Processor interface**
  - interrupts
  - memory mapped I/O
- **I/O Control Structures**
  - polling
  - interrupts
  - DMA
  - I/O Controllers
  - I/O Processors

Summary: Relationship to Processor Architecture

- **I/O instructions have disappeared**
- **Interrupt vectors have been replaced by jump tables**
- **Interrupt stack replaced by shadow registers**
- **Interrupt types reduced in number**
- **Caches required for processor performance cause problems for I/O**
- **Virtual memory frustrates DMA**
- **Load/store architecture at odds with atomic operations**
- **Stateful processors hard to context switch**

Network Attached Storage

Decreasing Disk Diameters

14" » 10" » 8" » 5.25" » 3.5" » 2.5" » 1.8" » 1.3" » . . .
high bandwidth disk systems based on arrays of disks

Network provides well defined physical and logical interfaces: ***separate CPU and storage system!***

High Performance ***Storage Service*** **on a High Speed Network**

Network File Services

OS structures supporting remote file access

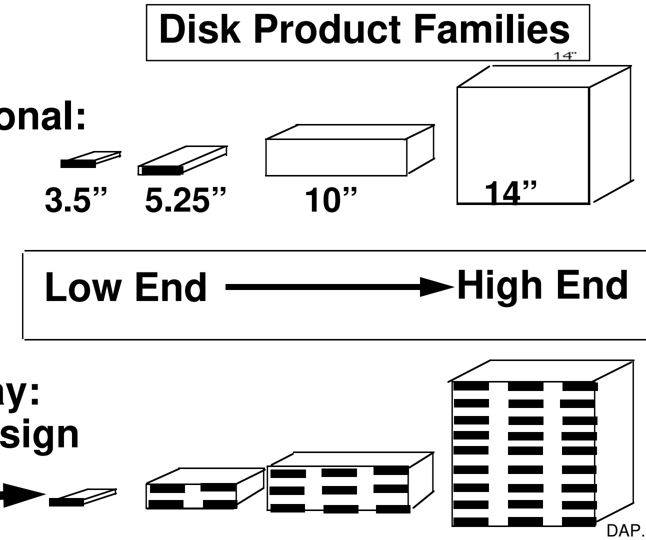
3 Mb/s » 10Mb/s » 50 Mb/s » 100 Mb/s » 1 Gb/s » ***10 Gb/s***
networks capable of sustaining high bandwidth transfers

Increasing Network Bandwidth

Manufacturing Advantages of Disk Arrays

Conventional: 4 disk designs

Disk Array: 1 disk design



Replace Small # of Large Disks with Large # of Small Disks!

	IBM 3390 (K)	IBM 3.5" 0061	*x70*
Data Capacity	20 GBytes	320 MBytes	*23 GBytes*
Volume	97 cu. ft.	0.1 cu. ft.	*11 cu. ft.*
Power	3 KW	11 W	*1 KW*
Data Rate	15 MB/s	1.5 MB/s	*120 MB/s*
I/O Rate	600 I/Os/s	55 I/Os/s	*3900 IOs/s*
MTTF	250 KHrs	50 KHrs	*??? Hrs*
Cost	$250K	$2K	*$150K*

Disk Arrays have potential for
- large data and I/O rates
- high MB per cu. ft., high MB per KW
- awful reliability

Redundant Arrays of Disks

- **Files are "striped" across multiple spindles**
- **Redundancy yields high data availability**

Disks will fail

Contents reconstructed from data redundantly stored in the array

→ **Capacity penalty to store it**

→ **Bandwidth penalty to update**

Techniques:
- Mirroring/Shadowing (high capacity cost)
- Horizontal Hamming Codes (overkill)
- Parity & Reed-Solomon Codes
- Failure Prediction (no capacity overhead!) *VaxSimPlus — Technique is controversial*

Array Reliability

- **Reliability of N disks = Reliability of 1 Disk ÷ N**

  50,000 Hours ÷ 70 disks = 700 hours

  Disk system MTTF: Drops from 6 years to 1 month!

- **Arrays without redundancy too unreliable to be useful!**

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

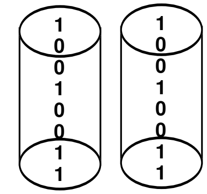
Redundant Arrays of Disks (RAID) Techniques

- ***Disk Mirroring, Shadowing***

  Each disk is fully duplicated onto its "shadow"

  Logical write = two physical writes

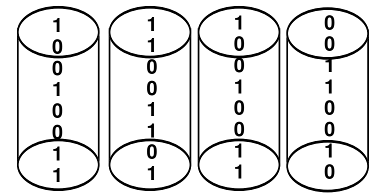
  100% capacity overhead



- ***Parity Data Bandwidth Array***

  Parity computed horizontally

  Logically a single high data bw disk



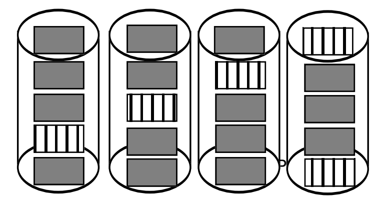
- ***High I/O Rate Parity Array***

  Interleaved parity blocks

  Independent reads and writes

  Logical write = 2 reads + 2 writes

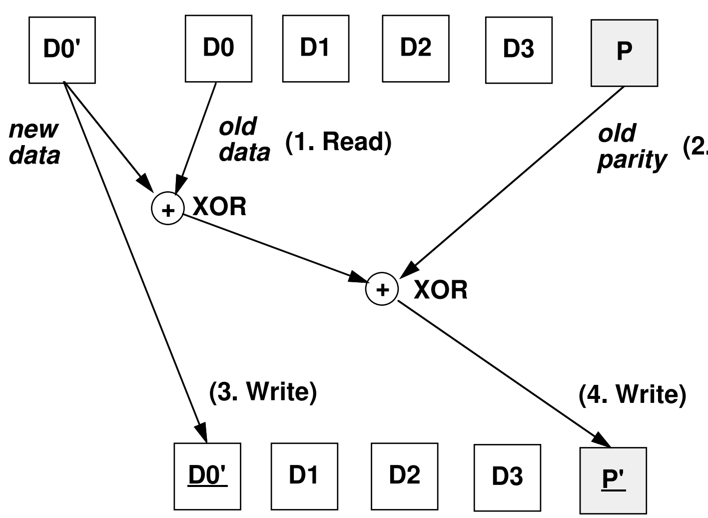
  Parity + Reed-Solomon codes



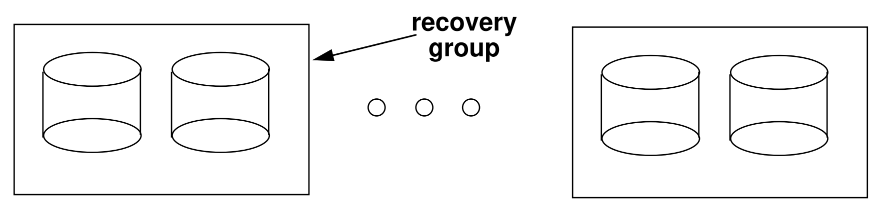
Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



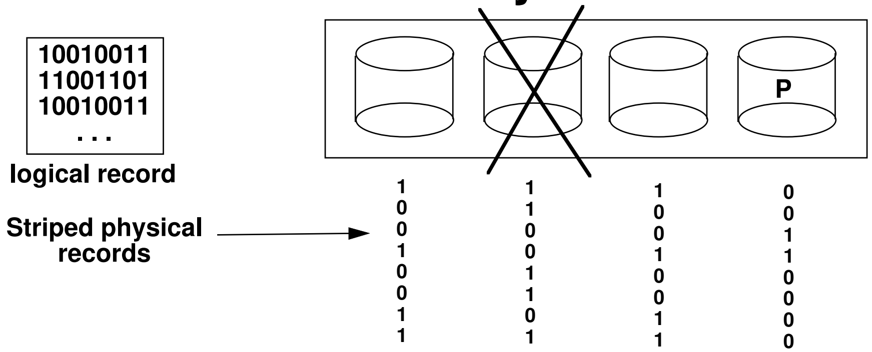
Redundant Arrays of Disks RAID 1: Disk Mirroring/Shadowing



- **Each disk is fully duplicated onto its "shadow"**
  Very high availability can be achieved
- **Bandwidth sacrifice on write:**
  Logical write = two physical writes
- **Reads may be optimized**
- **Most expensive solution: 100% capacity overhead**

Targeted for high I/O rate , high availability environments

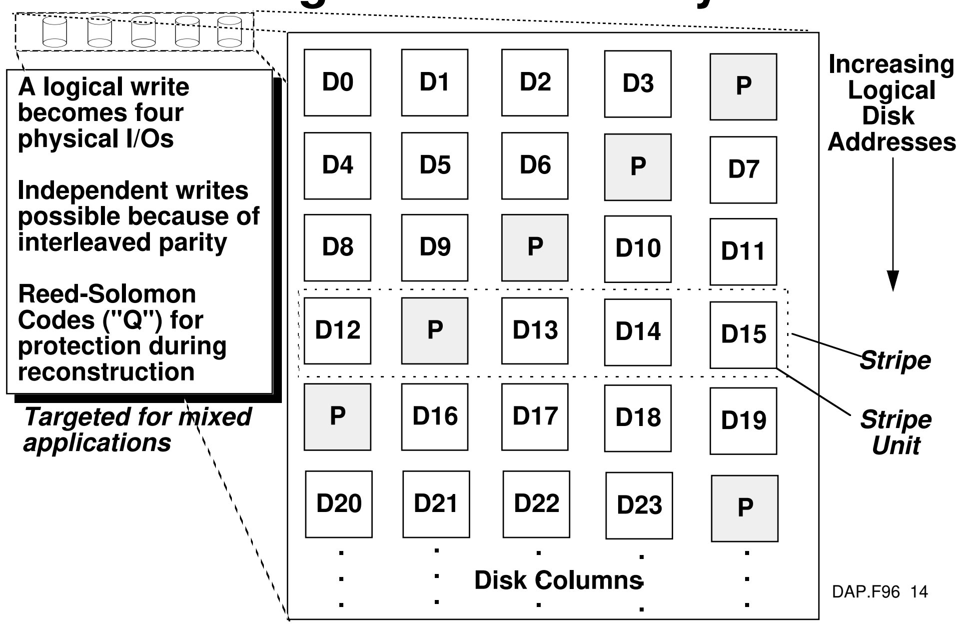
Redundant Arrays of Disks RAID 3: Parity Disk



- Parity computed across recovery group to protect against hard disk failures
  - 33% capacity cost for parity in this configuration
  - wider arrays reduce capacity costs, decrease expected availability, increase reconstruction time
- Arms logically synchronized, spindles rotationally synchronized
  - logically a single high capacity, high transfer rate disk

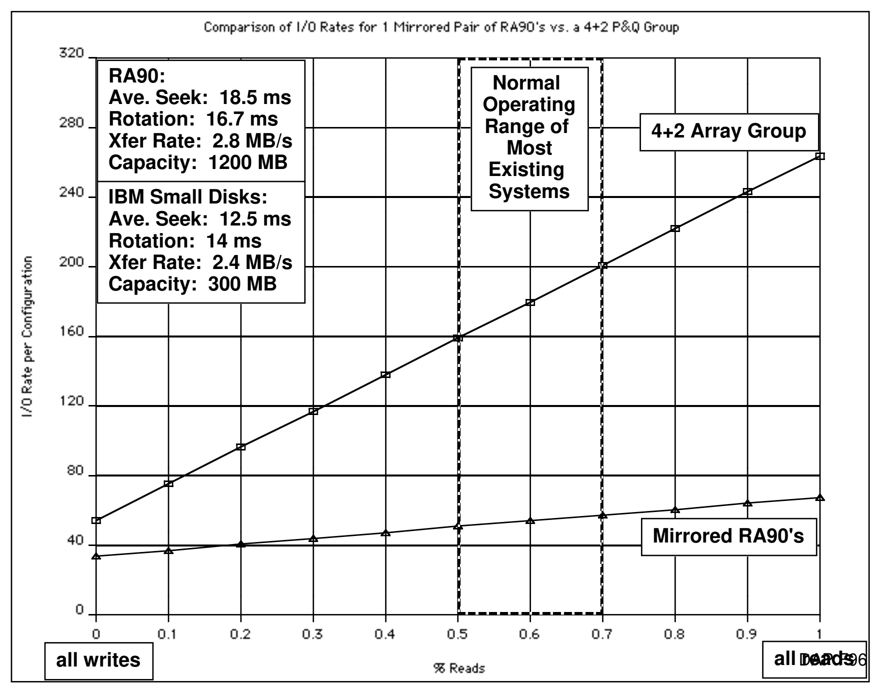
Targeted for high bandwidth applications: Scientific, Image Processing

Redundant Arrays of Disks RAID 5+: High I/O Rate Parity

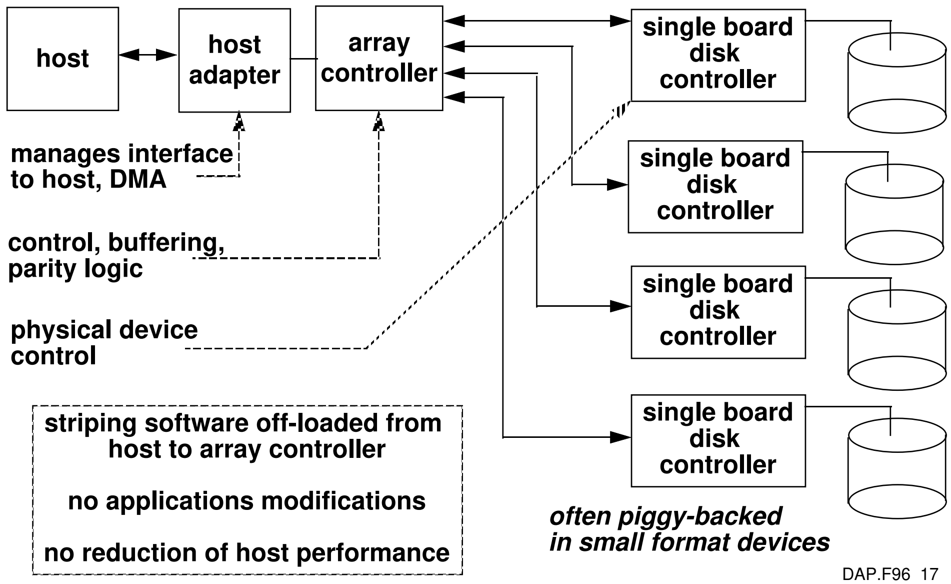


CS 252 Administrivia

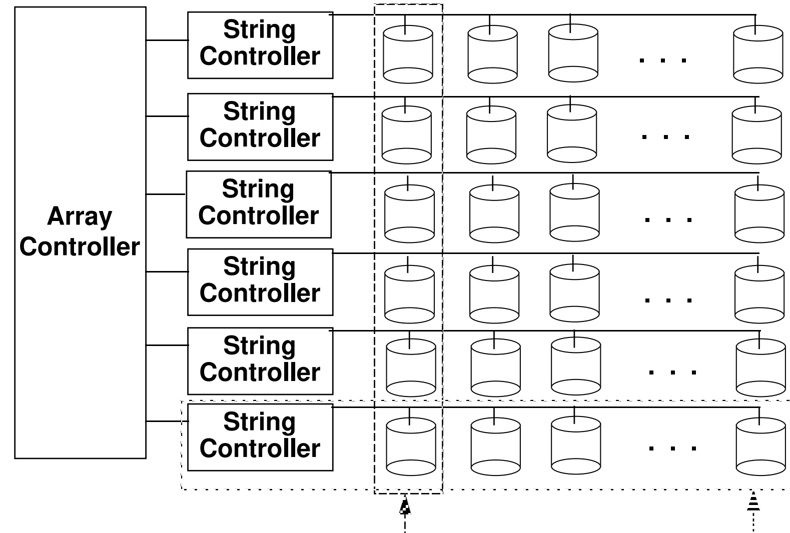
- Homework on Chapter 6 due Monday10/21 at 5PM in 252 box, done in pairs:
  - Exercises 6.5, 6.16, 6.17
- Project Survey #2 due Wednesday 10/23 in class



Subsystem Organization



System Availability: Orthogonal RAIDs

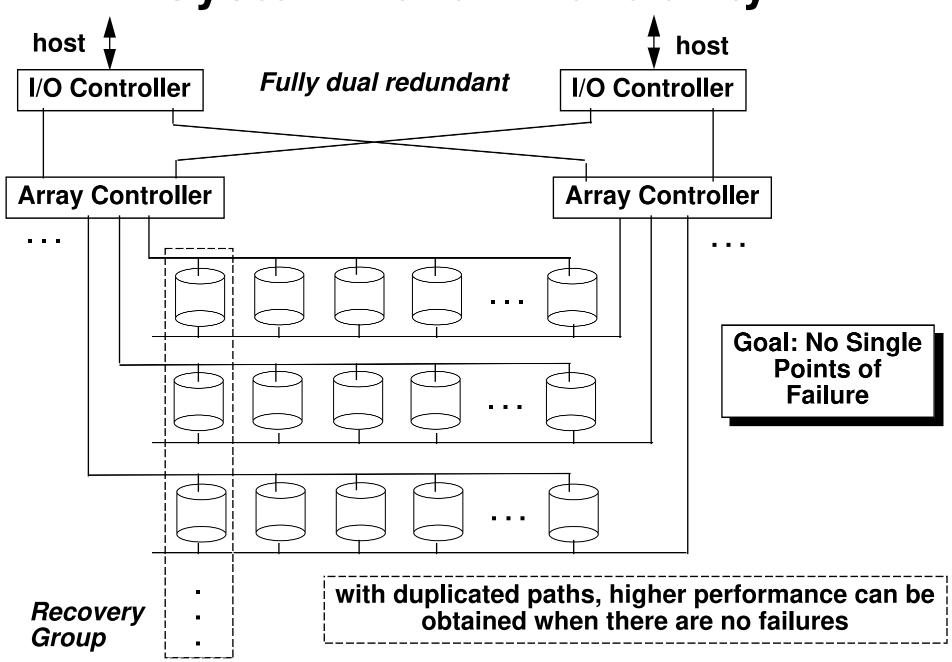


Data Recovery Group: unit of data redundancy

Redundant Support Components: fans, power supplies, controller, cables

End to End Data Integrity: internal parity protected data paths

System-Level Availability



Review: Storage System Issues

- Historical Context of Storage I/O
- Storage I/O Performance Measures
- Secondary and Tertiary Storage Devices
- A Little Queuing Theory
- Processor Interface Issues
- I/O & Memory Buses
- RAID
- ABCs of UNIX File Systems
- I/O Benchmarks
- Comparing UNIX File System Performance
- Tertiary Storage Possbilities

ABCs of UNIX File Systems

- **Key Issues**
  - File vs. Raw I/O
  - File Cache Size Policy
  - Write Policy
  - Local Disk vs. Server Disk
- **File vs. Raw:**
  - File system access is the norm: standard policies apply
  - Raw: alternate I/O system to avoid file system, used by data bases
- **File Cache Size Policy**
  - % of main memory dedicated to file cache is fixed at system generation (e.g., 10%)
  - % of main memory for file cache varies depending on amount of file I/O (e.g., up to 80%)

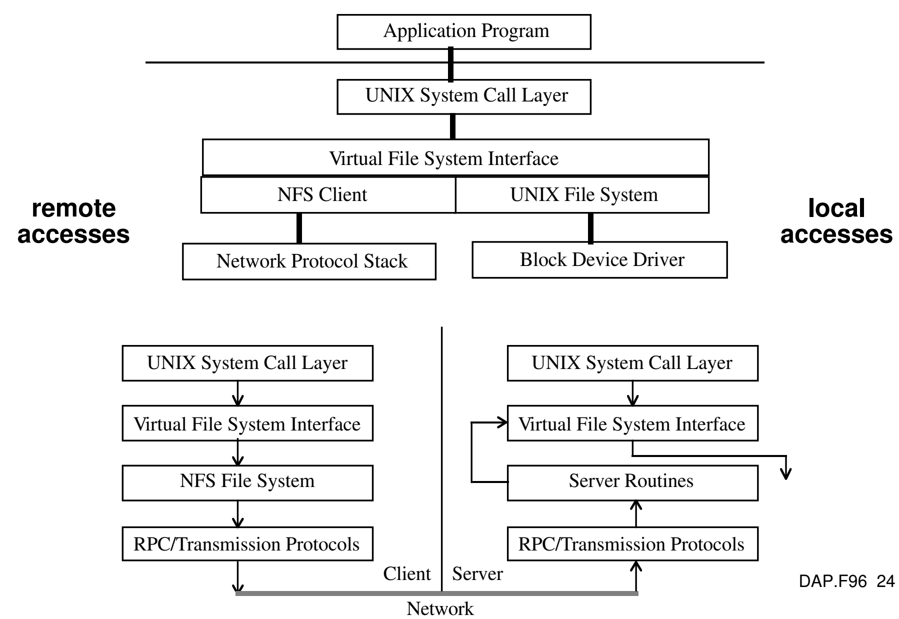
ABCs of UNIX File Systems

- **Write Policy**
  - File Storage should be permanent; either write immediately or flush file cache after fixed period (e.g., 30 seconds)
  - Write Through with Write Buffer
  - Write Back
  - Write Buffer often confused with Write Back
    - » Write Through with Write Buffer, all writes go to disk
    - » Write Through with Write Buffer, writes are asynchronous, so processor doesn't have to wait for disk write
    - » Write Back will combine multiple writes to same page; hence can be called Write Cancelling

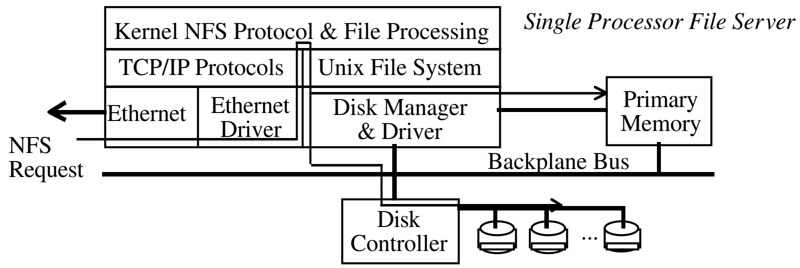
ABCs of UNIX File Systems

- **Local vs. Server**
  - Unix File systems have historically had different policies (and even file sytems) for local client vs. remote server
  - NFS local disk allows 30 second delay to flush writes
  - NFS server disk writes through to disk on file close
  - Cache coherency problem if allow clients to have file caches in addition to server file cache
    - » NFS just writes through on file close
Stateless protocol: periodically get new copies of file blocks
    - » Other file systems use cache coherency with write back to check state and selectively invalidate or update

Network File Systems



Typical File Server Architecture



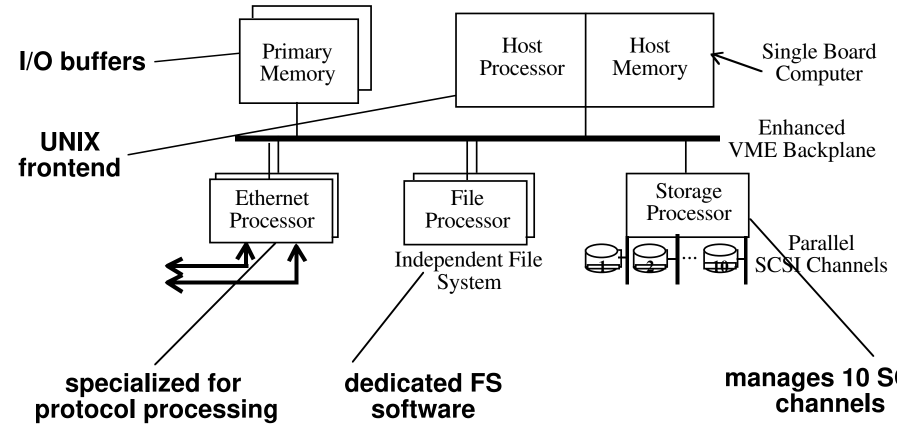
Limits to performance: data copying

- read data staged from device to primary memory
- copy again into network packet templates
- copy yet again to network interface

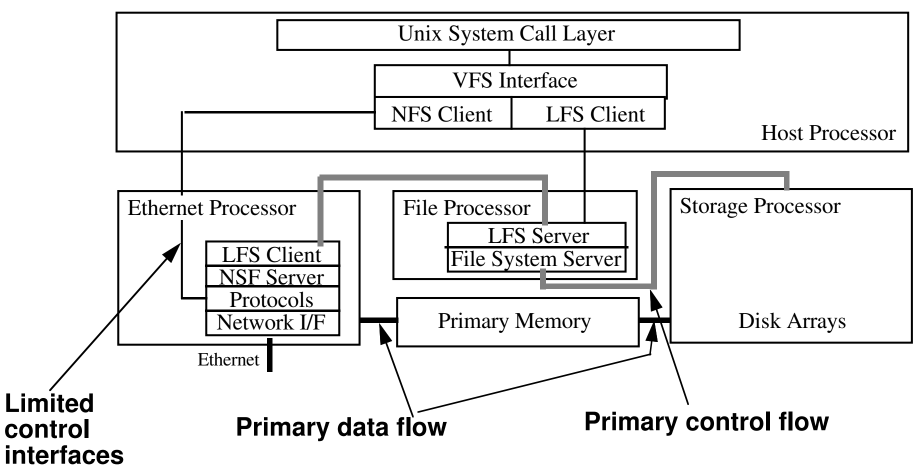
No specialization for fast processing between network and disk

AUSPEX NS5000 File Server

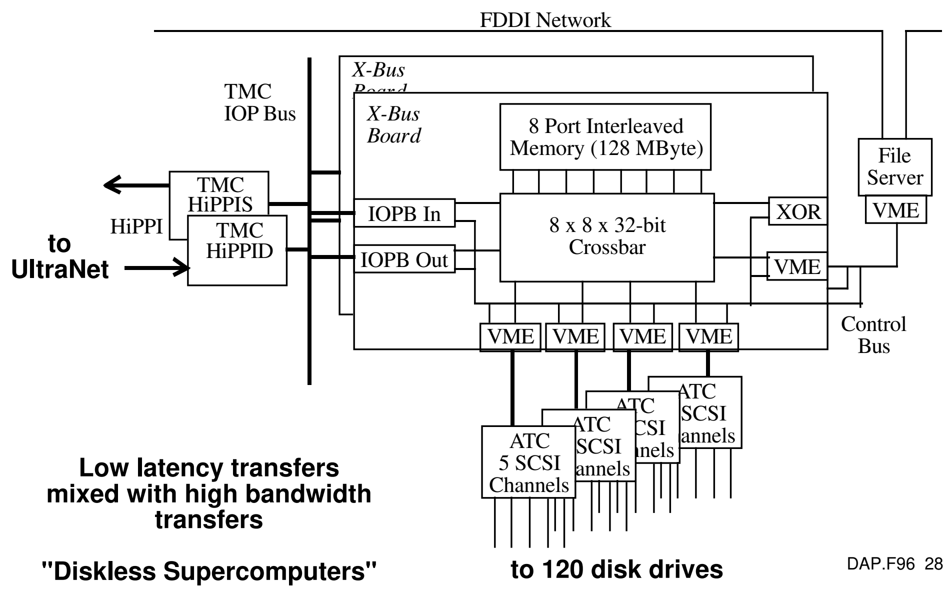
- **Special hardware/software architecture for high performance NFS I/O**
- **Functional multiprocessing**



AUSPEX Software Architecture



Berkeley RAID-II Disk Array File Server



Low latency transfers mixed with high bandwidth transfers

"Diskless Supercomputers"

5 minute Class Break

- **Lecture Format:**
  - ≈ 1 minute: review last time & motivate this lecture
  - ≈ 20 minute lecture
  - ≈ 3 minutes: discuss class manangement
  - ≈ 25 minutes: lecture
  - 5 minutes: break
  - ≈25 minutes: lecture
  - ≈1 minute: summary of today's important topics

I/O Benchmarks

- **For better or worse, benchmarks shape a field**
  - Processor benchmarks classically aimed at response time for fixed sized problem
  - I/O benchmarks typically measure throughput, possibly with upper limit on response times (or 90% of response times)
- **What if fix problem size, given 60%/year increase in DRAM capacity?**

Benchmark	*Size of Data*	*% Time I/O*	*Year*
I/OStones	1 MB	26%	1990
Andrew	4.5 MB	4%	1988

  - Not much time in I/O
  - Not measuring disk (or even main memory)

I/O Benchmarks

- **Alternative: self-scaling benchmark; automatically and dynamically increase aspects of workload to match characteristics of system measured**
  - Measures wide range of current & future
- **Describe three self-scaling benchmarks**
  - Transacition Processing: TPC-A, TPC-B, TPC-C
  - NFS: SPEC SFS (LADDIS)
  - Unix I/O: Willy

I/O Benchmarks: Transaction Processing

- **Transaction Processing (TP) (or On-line TP=OLTP)**
  - Changes to a large body of shared information from many terminals, with the TP system guaranteeing proper behavior on a failure
  - If a bank's computer fails when a customer withdraws money, the TP system would guarantee that the account is debited if the customer received the money and that the account is unchanged if the money was not received
  - Airline reservation systems & banks use TP
- **Atomic transactions makes this work**
- **Each transaction => 2 to 10 disk I/Os & 5,000 and 20,000 CPU instructions per disk I/O**
  - Efficiency of TP SW & avoiding disks accesses by keeping information in main memory
- **Classic metric is Transactions Per Second (TPS)**
  - Under what workload? how machine configured?

I/O Benchmarks: Transaction Processing

- Early 1980s great interest in OLTP
  - Expecting demand for high TPS (e.g., ATM machines, credit cards)
  - Tandem's success implied medium range OLTP expands
  - Each vendor picked own conditions for TPS claims, report only CPU times with widely different I/O
  - Conflicting claims led to disbelief of all benchmarks=> chaos
- 1984 Jim Gray of Tandem distributed paper to Tandem employees and 19 in other industries to propose standard benchmark
- Published "A measure of transaction processing power," Datamation, 1985 by Anonymous et. al
  - To indicate that this was effort of large group
  - To avoid delays of legal department of each author's firm

I/O Benchmarks: TP by Anon et. al

- Proposed 3 standard tests to characterize commercial OLTP
  - TP1: OLTP test, DebitCredit, simulates ATMs (TP1)
  - Batch sort
  - Batch scan
- Debit/Credit:
  - One type of transaction: 100 bytes each
  - Recorded 3 places: account file, branch file, teller file + events recorded in history file (90 days)
    - » 15% requests for different branches
  - Under what conditions, how report results?

I/O Benchmarks: TP1 by Anon et. al

- DebitCredit Scalability: size of account, branch, teller, history function of throughput

TPS	Number of ATMs	Account-file size
10	1,000	0.1 GB
100	10,000	1.0 GB
1,000	100,000	10.0 GB
10,000	1,000,000	100.0 GB

  - Each input TPS =>100,000 account records, 10 branches, 100 ATMs
  - Accounts must grow since a person is not likely to use the bank more frequently just because the bank has a faster computer!
- Response time: 95% transactions take $\leq$ 1 second
- Configuration control: just report price (initial purchase price + 5 year maintenance = cost of ownership)
- By publishing, in public domain

I/O Benchmarks: TP1 by Anon et. al

- Problems
  - Often ignored the user network to terminals
  - Used transaction generator with no think time; made sense for database vendors, but not what customer would see
- Solution: Hire auditor to certify results
  - Auditors soon saw many variations of ways to trick system
- Proposed minimum compliance list (13 pages); still, DEC tried IBM test on different machine with poorer results than claimed by auditor
- Created Transaction Processing Performance Council in 1988: founders were CDC, DEC, ICL, Pyramid, Stratus, Sybase, Tandem, and Wang
- Led to TPC standard benchmarks in 1990

I/O Benchmarks: TPC Benchmarks

- TPC-A: Revised version of TP1/DebitCredit
  - Arrivals: Random (TPC) vs. uniform (TP1)
  - Terminals: Smart vs. dumb (affects instruction path length)
  - ATM scaling: 10 terminals per TPS vs. 100
  - Branch scaling: 1 branch record per TPS vs. 10
  - Response time constraint: 90% ≤2 seconds vs. 95% ≤1
  - Full disclosure, approved by TPC
  - Complete TPS vs. response time plots vs. single point
- TPC-B: Same as TPC-A but without terminals—batch processing of requests
  - Response time makes no sense: plots tps vs. residence time (time of transaction resides in system)
- Other efforts underway on complex query processing (C) and decision support (D)

TPC Results

TPC-A

Machine	*tpsA-local*	*K$/tps*	*OS/DB*	*Date*
HP 852S	43	24	HPUX 7/Infmx 4	12/90
VAX 4000	41	23	VMS 5.4/Dec 6	7/90
IBM RS6/550	32	20	Aix 3.1/infmx 4	1/91
Compaq SysPro	172	5	??	1/93
SPARCserve41	108	7	??	1/93
HP 9000 890/4	710	8	??	1/93

TPC-B

Machine	*tpsB*	*K$/tps*	*OS/DB*	*Date*
HP 852S	90	5	HPUX 7/Infmx 4	12/90
IBM RS6/550	58	5	Aix 3.1/infmx 4	1/91
Sun SS 490	57	8	Sun4.1/Sybase 4	10/90
Sun SS 2	52	4	Sun4.1/Sybase 4	10/90

SPEC SFS/LADDIS Predecessor: NFSstones

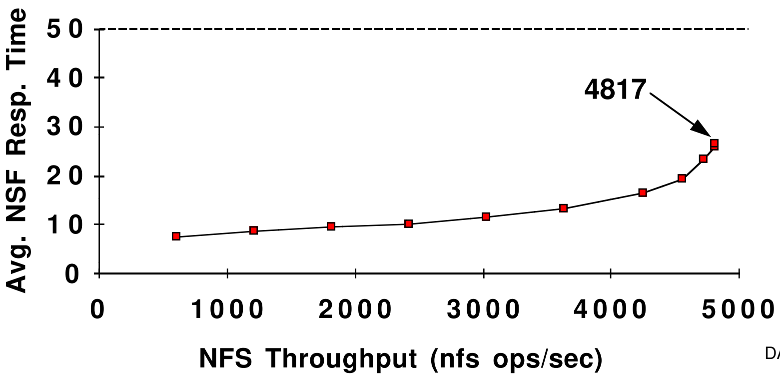
- NFSStones: synthetic benchmark that generates series of NFS requests from single client to test server: reads, writes, & commands & file sizes from other studies
  - Problem: 1 client could not always stress server
  - Files and block sizes not realistic
  - Clients had to run SunOS

SPEC SFS/LADDIS

- 1993 Attempt by NFS companies to agree on standard benchmark: Legato, Auspex, Data General, DEC, Interphase, Sun. Like NFSstones but
  - Run on multiple clients & networks (to prevent bottlenecks)
  - Same caching policy in all clients
  - Reads: 85% full block & 15% partial blocks
  - Writes: 50% full block & 50% partial blocks
  - Average response time: 50 ms
  - Scaling: for every 100 NFS ops/sec, increase capacity 1GB
  - Results: plot of server load (throughput) vs. response time & number of users
    - » Assumes: 1 user => 10 NFS ops/sec

Example SPEC SFS Result: DEC Alpha

- 200 MHz 21064: 8KI + 8KD + 2MB L2; 512 MB; 1 Gigaswitch
- DEC OSF1 v2.0
- 4 FDDI networks; 32 NFS Daemons, 24 GB file size
- 88 Disks, 16 controllers, 84 file systems



Willy

- UNIX File System Benchmark that gives insight into I/O system behavior (Chen and Patterson, 1993)
- Self scaling to automatically explore system size
- Examines five parameters
  - Unique bytes touched: ≈ data size; locality via LRU
    - » Gives file cache size
  - Percentage of reads: %writes = 1 – % reads; typically 50%
    - » 100% reads gives peak throughput
  - Average I/O Request Size: Bernoulli, C=1
  - Percentage sequential requests: typically 50%
  - Number of processes: concurrency of workload (number processes issuing I/O requests)
- Fix four parameters while vary one parameter
- Searches space to find high throughput

Example Willy: DS 5000

	Sprite	Ultrix
Avg. Access Size	32 KB	13 KB
Data touched (file cache)	2MB, 15 MB	2 MB
Data touched (disk)	36 MB	6 MB

- % reads = 50%, % sequential = 50%
- DS 5000 32 MB memory
- Ultrix: Fixed File Cache Size, Write through
- Sprite: Dynamic File Cache Size, Write back (Write cancelling)

Sprite's Log Structured File System

Large file caches effective in reducing disk reads

Disk traffic likely to be dominated by writes

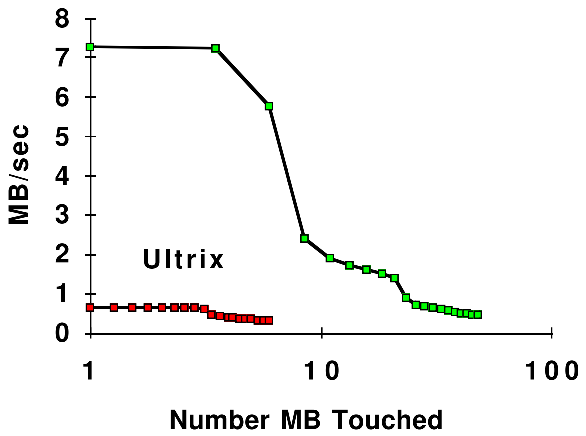
Write-Optimized File System

- Only representation on disk is log
- Stream out files, directories, maps without seeks

Advantages:
- Speed
- Stripes easily across several disks
- Fast recovery
- Temporal locality
- Versioning

Problems:
- Random access retrieval
- Log wrap
- Disk space utilization

Willy: DS 5000 Number Bytes Touched



- Log Structured File System: effective write cache of LFS much smaller (5-8 MB) than read cache (20 MB)
  – Reads cached while writes are not => 3 plateaus

Summary: I/O Benchmarks

- Scaling to track technological change
- TPC: price performance as nomalizing configuration feature
- Auditing to ensure no foul play
- Througput with restricted response time is normal measure