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
- Show and Tell
- ABCs of UNIX File Systems
- I/O Benchmarks
- Comparing UNIX File System Performance
- RAID
- Teritary Storage Possbilities

Review: Bus Options

Option	*High performance*	*Low cost*
Bus width	Separate address & data lines	Multiplex address & data lines
Data width	Wider is faster (e.g., 32 bits)	Narrower is cheaper (e.g., 8 bits)
Transfer size	Multiple words has less bus overhead	Single-word transfer is simpler
Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
Split transaction?	Yes separate Request and Reply packets gets higher bandwidth (needs multiple masters)	No continuous connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

Review: Show and Tell

In class demo of 14 , 5.25 , 3.5 , and 2.5 disks

In class demo SCSI chain

In class demo fiber optic cable & Electro-optical converter

In class demo of Sun 4, first commercially successful RISC machine: VME bus, rack, memory cards

In class demo of SPARCstation-1, first э pizza box workstation: SCSI, SIMM modules, S-bus

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 generartion (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, thereby invalidating local copies?
  - ≅ Other file systems use cache coherency with write back to check state and selectively invalidate or update

I/O Benchmarks

For better or worse, benchmarks shape a field

Procesor 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 3 self-scaling benchmarks

- Transacition Processing: TPC-A, TPC-B, TPC-C
- NFS: SPEC SFS (LADDIS)
- Unix I/O: Wylie

I/O Benchmarks: Tranaction 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

The 2-phase transaction 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 Transactios Per Second (TPS):

- under what workload? how machine configured?

I/O Benchmarks: Tranaction 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 stanard benchmark

Published A measure of transaction processing power, Datamation, 1985 by Anonymous et al

- To incidate that this was effort of large group
- To avoid delays of legal department of each author s firm
- Still get mail at Tandem to author

I/O Benchmarks: TP by Anon et al

Proposed 3 standard tests to characterize commercial OLTP

- ≅ OLTP test, DebitCredit, simulates ATMs (TP1)
- ≅ batch sort
- ≅ batch scan

DebitCredit:

- ≅ 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% transactios take | 1 second

Configuration control: just report cost (initial purchase price + 5 year maintenance = cost of ownership)

By publishing, in public domain

I/O Benchmarks: TP 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 (Omri Serlin): founders were CDC, DEC, ICL, Pyramid, Status, Sybase, Tandem, and Wang

Led to TPC standard benchmarks in 1990

I/O Benchmarks: TPC benchmarks

TPC-A: version of DebitCredit

- ≅ random vs. uniform arrival
- ≅ smart terminals vs. dumb
- ≅ 10 terminals per TPS vs. 100
- ≅ 90% | 2 seconds vs. 95% | 1,
- ≅ 1 branch record per TPS vs. 10
- ≅ full disclosure, apporved by TPC
- ≅ complete TPS vs reponse 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 processeing (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: NSFstones

NSFStones: syntheic benchmark that generates series of NFS requests from single client to test server: reads, writes, & commands & file sizes from other studies

- **problem: 1 server 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 strandard benchmark: Legato, Auspex, Data General, DEC, Interpahse, 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

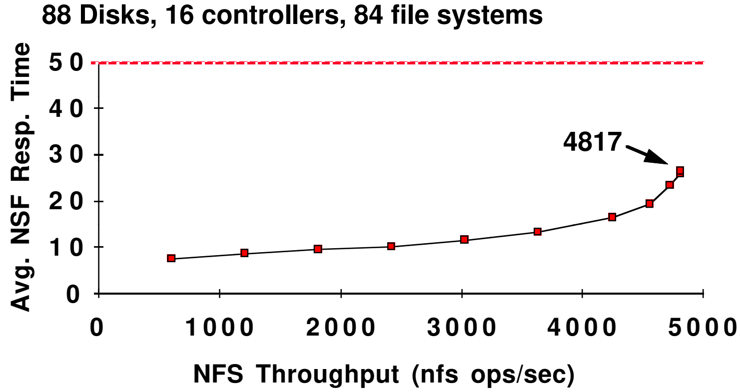
Example SPEC SFS Result: DEC Alpha

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
- By Chen and Patterson, 1993
- Self scaling to automatically explore system size
- Examines 5 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**: typcially 50%
  - ≅ **Number of processes**: concurrency of workload (number processes issuing I/O requests)
- Fix 4 parameters while vary 1 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

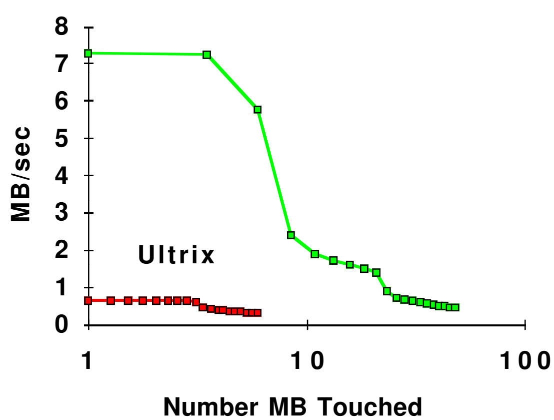
Problems:

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 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 restrcited response time is normal measure