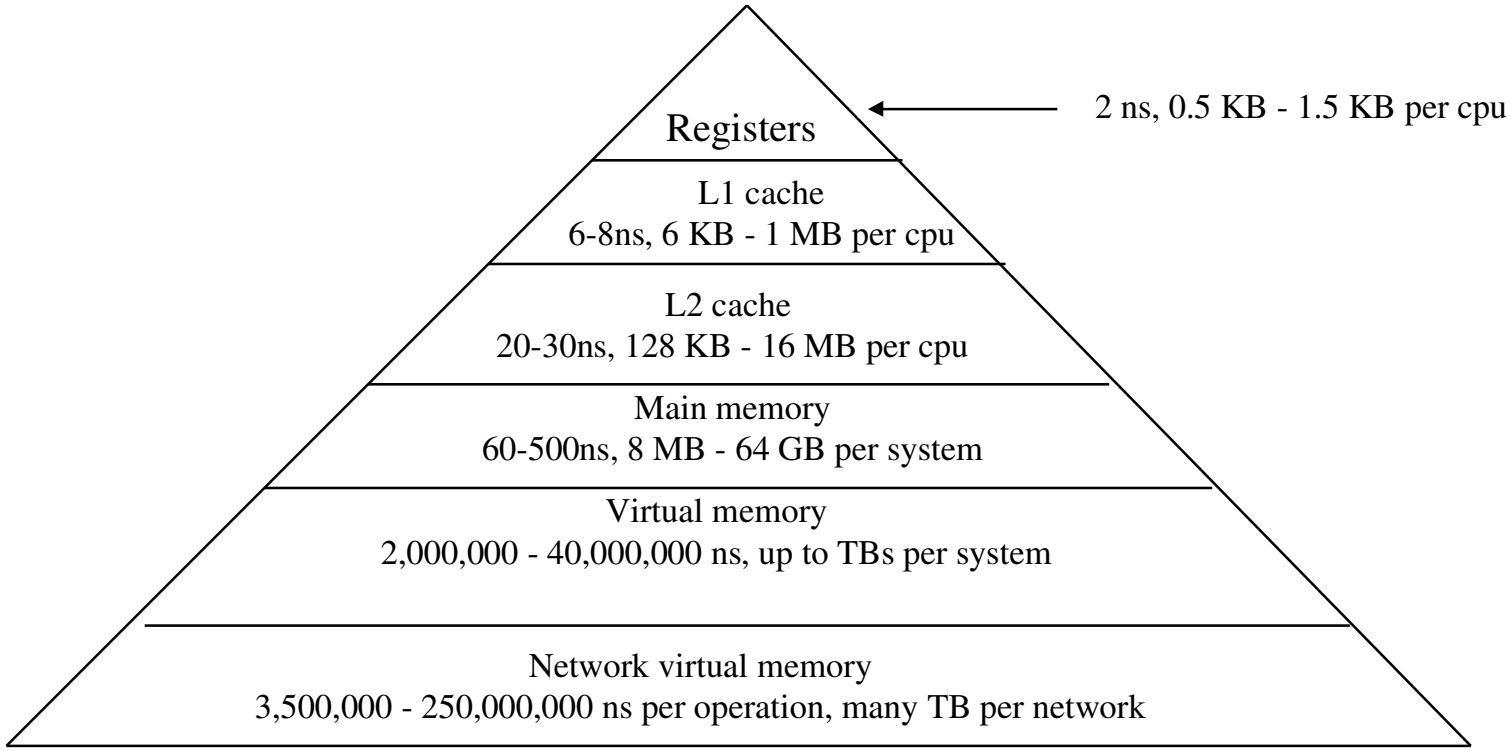
Applied Computer Architecture

Brian Wong
Chief Scientist, Enterprise Engineering
Sun Microsystems
November 6, 1997
blw@eng.sun.com

Two Day Agenda

- Applied architecture in the processor complex
- Busses and Interconnects
- Disks
- RAID levels (0, 1, 3, 1+0, 0+1, 5, 6)
- RAID optimization

The Virtual Memory Hierarchy



Effect of the Hierarchy

- End-user application performance is the sum of:
  - processor pipeline performance (~85-98% of the time)
  - E$ performance (most of the rest)
  - memory subsystem performance for cache misses
    - some apps miss E$ up to 10-15% of the time
    - long-latency machines have penalties of 100-200 cycles
    - even longer for NUMA machines
  - disk subsystem performance on page faults

Pipelining in the Real World

- How effective are superscalar pipelines?
- What's a good IPC?
- It depends
  - IPC ratings are > 2.0
    - assume execution from I$, D$
    - hand-rolled assembly code
  - sustained IPC > 1.6 is entirely possible
    - floating point codes
    - good compilers
    - well-behaved codes (e.g. matrix multiply)

Less Efficient Applications

- IPC of < 1 is not theoretical
  - typical DBMS codes average < 1.0 instr/cycle
  - even with good compilers
  - I$ miss rate can be very high
- IPC of << 1.0 is not theoretical!
  - Virtually any app that processes linked lists average << 1.0 cpi
  - E$ misses galore!

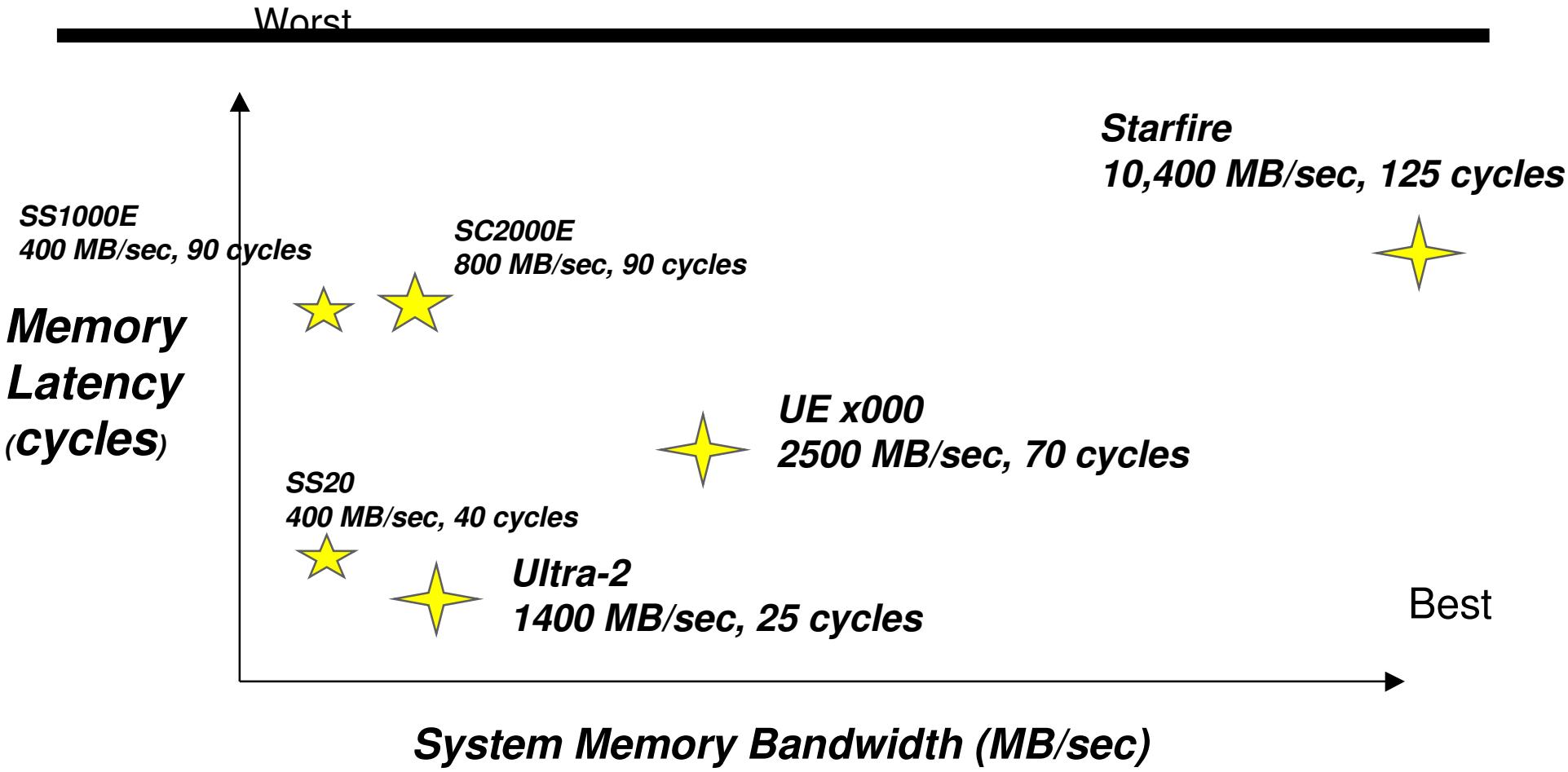
```
#define MAXSIZE 128
struct list {
   integer key;
   char *name;
   char *address;
   char *address2;
   char *city;
   char *state;
   char *country;
   char *postalCode;
   char *phone;
   char *fax;
   char blockData [MAXSIZE];
   struct list *nextp, *prevp;
};
```

nextp, prevp are in a different cache line than the key

Memory Latency

- The amount of time the combined CPU, memory and centerplane require to service a cache miss
- Higher bandwidth requires more (worse) latency
  - Ultra Enterprise 10000 is like a 747: it carries 400 people 8000 miles at 600mph, but you have to park at the airport, hire a car, wait for the flight, drive downtown from the airport.
  - Ultra-2 is like a Porsche 911: it carries only two people for only 500 miles at only 100mph, but you can get downtown directly without waiting.
  - Which would you take to go to San Jose, CA? San Jose Puerto Rico?

Memory Latency



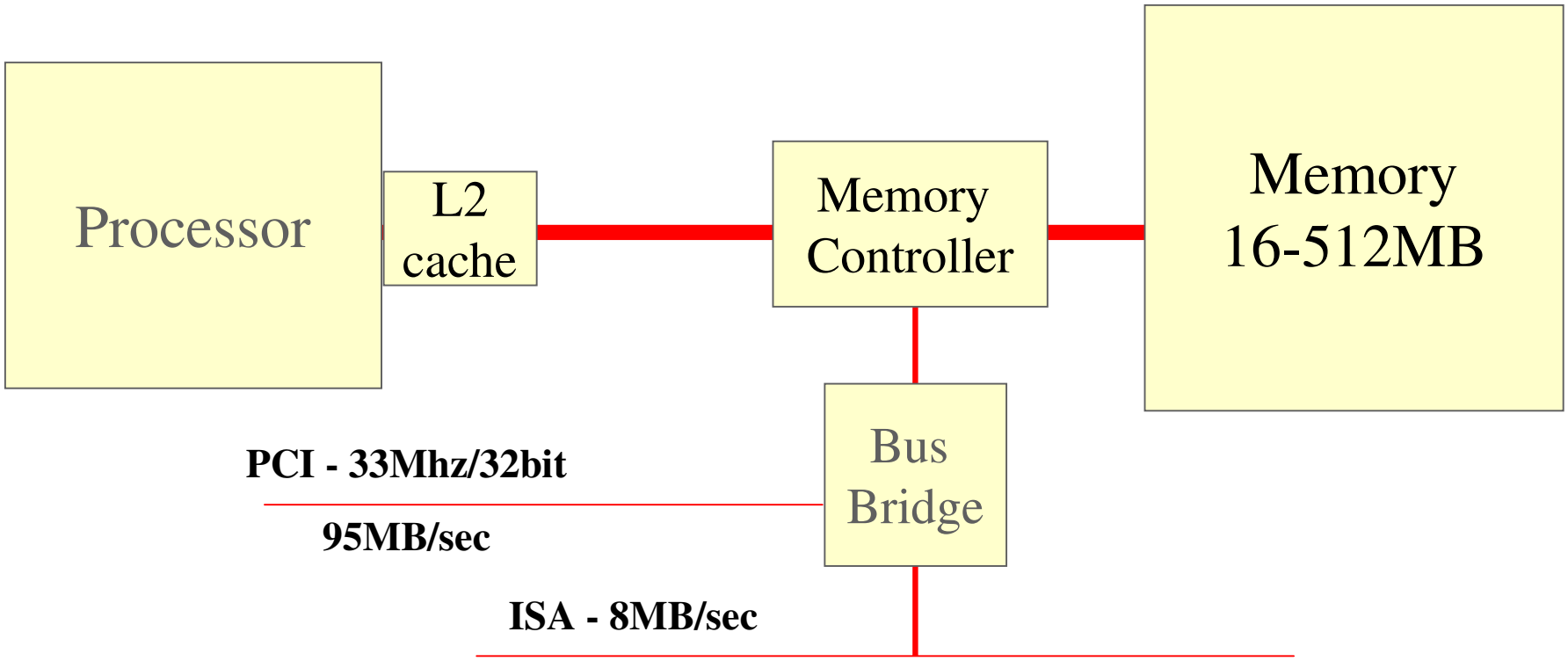
Busses and Interconnects

- a way to get from one place to another
  - in a computer they're like roads and blood vessels
  - a bunch of wires going the same place with a defined protocol
- there are *many* design differences
- four major classes of bus
  - backplane (processor-memory)
  - inter-node interconnect (between members of a NUMA cluster)
  - I/O (eg PCI, PC-MCIA/PC-Card, MicroChannel, Sbus)
  - peripheral (SCSI, FibreChannel, USB)

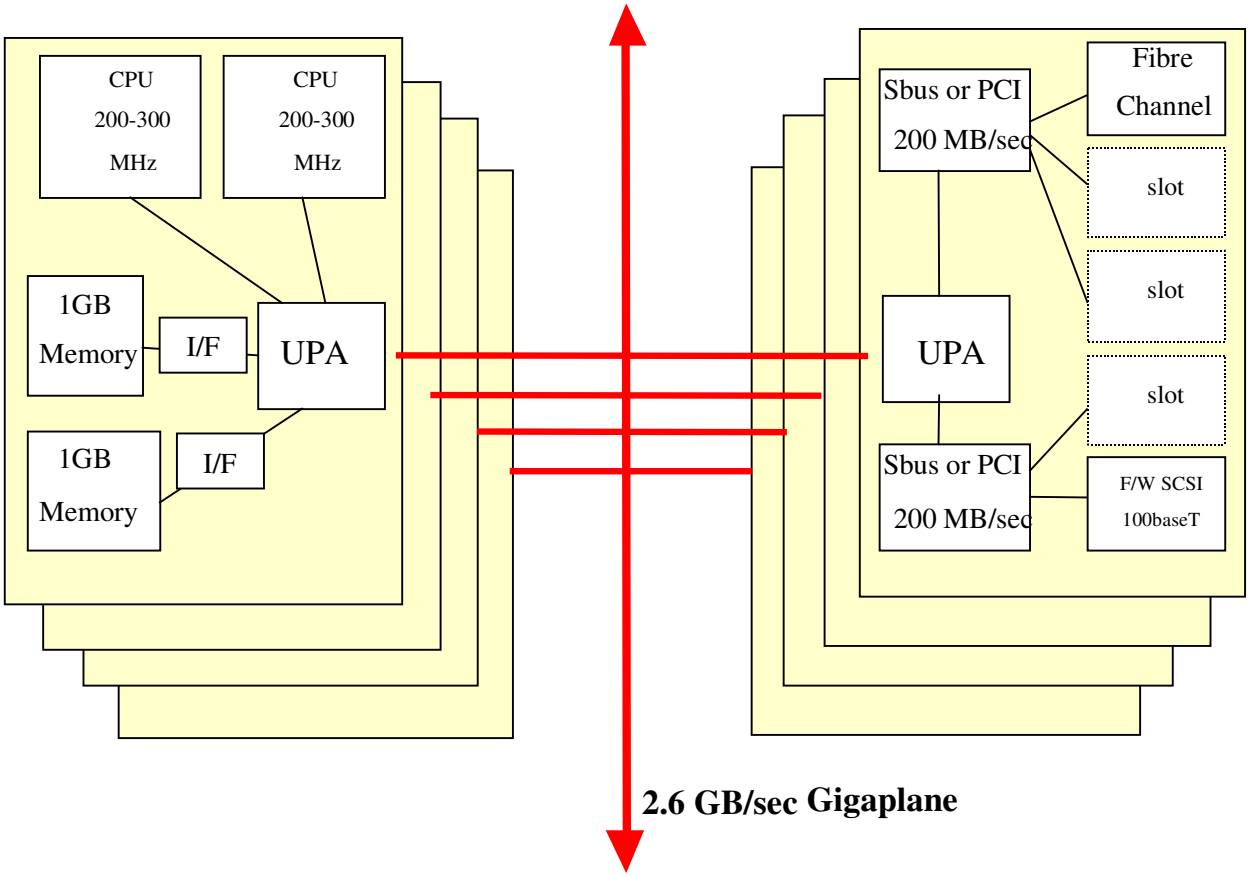
Design Tradeoffs in Busses

- overall speed
- parallelism (between bits, users, transactions)
- physical length
- configurability
- speed of various users (perhaps speed-matching)
  - cpu/memory - 100+ MB/sec per cpu
  - keyboard - 15 bytes/sec
  - disk - 0.2-200 MB/sec
  - graphics - 500+MB/sec
- cost (!)

"Thin" Architecture



A Larger and Wider System



Sun Ultra Enterprise x000

Bus/Interconnect Comparison

Bus	Date	Width	Clock	Theoretical	Achievable	Efficiency
MBus	1992	64	40	320	135	42%
XDBus	1992	64	40	320	270	84%
UPA	1995	128	83.3	1,333	1,220	92%
Gigaplane	1995	256	83.3	2,667	2,500	94%
Gigaplane XB	1997	1024	83.3	10,667	10,000	94%
VME	1979	32	10	40	25	63%
VME64	1988	64	10	80	55	69%
SBus	1989	32	25	100	65	65%
SBus-64	1994	64	25	200	105	53%
ISA	1981	8	8.3	8	5	60%
MicroChannel	1987	32	10	40	21	53%
EISA	1988	32	8.3	33	15	45%
PCI	1993	32	33.3	133	90	68%
PCI64	1997	64	33.3	267		
EPCI		64	66.7	533		
SCSI-1	1984	8	5	5	1.2	24%
SCSI-2	1989	8	5	5	4.4	88%
SCSI-2 fast/wide	1990	16	10	20	18.4	92%
SCSI-3 ultra/wide	1996	16	20	40	36.5	91%
FibreChannel	1993	1	266	25	22	88%
FC-100	1997	1	1064	100	90	90%

Bus Interconnect Comparison

Bus	Date	Width	Clock	Theoretical	Achievable	Efficiency	Notes
MBus	1992	64	40	320	135	42%	circuit-switched
XDBus	1992	64	40	320	270	84%	packet-switched
UPA	1995	128	83.3	1,333	1,220	92%	independent control, address, data
Gigaplane	1995	256	83.3	2,667	2,500	94%	both UPA and Gigaplane packet-switched
Gigaplane XB	1997	1024	83.3	10,667	10,000	94%	Crossbar switch using Gigaplanes
VME	1979	32	10	40	25	63%	
VME64	1988	64	10	80	55	69%	
SBus	1989	32	25	100	65	65%	but only 8 MB/sec in 1-byte PIO mode
SBus-64	1994	64	25	200	105	53%	SBus-64 multiplexes address lines
ISA	1981	8	8.3	8	5	60%	
MicroChannel	1987	32	10	40	21	53%	
EISA	1988	32	8.3	33	15	45%	
PCI	1993	32	33.3	133	90	68%	
PCI64	1997	64	33.3	267			only just shipping
EPCI		64	66.7	533			not yet commercialized
SCSI-1	1984	8	5	5	1.2	24%	async only; sync mode didn't work!
SCSI-2	1989	8	5	5	4.4	88%	synchronous transfers
SCSI-2 fast/wide	1990	16	10	20	18.4	92%	
SCSI-3 ultra/wide	1996	16	20	40	36.5	91%	
FibreChannel	1993	1	266	25	22	88%	full duplex - cited speeds for one direction
FC-100	1997	1	1064	100	90	90%	full duplex - cited speeds for one direction

Bus Protocol

- Three logical phases
  - arbitration - who gets to use the bus next
    - sometimes called control phase
  - address - what data will we be transferring?
  - Data - move the data itself
    - might be nil
- There are other bus transactions - error report, probe, etc
- Busses primarily differentiated by design in the three primary phases

Bus Design Choices

- Circuit-switched vs. packet-switched
  - circuit-switched: end-to-end connection stays up
  - packet-switched: disconnects during service, must be reconnected
- Shared vs. separate control, address, data lines
  - shared: fewer wires (much less $$$)
  - separate: can run three separate transactions simultaneously
- Data burst size
- Data transfer mode

Data Transfer Modes

- Programmed I/O (PIO)
  - processor directly issues load and store instructions to copy
  - uses processor cycles but requires no setup
  - pollutes the D$ and E$ because data moves directly through CPU
  - very simple
- Direct Virtual Memory Access (DVMA)
  - dedicated circuitry handles transfer mechanics (can be expensive)
  - operates in 4-, 8- and 64-byte blocks (usually 64-bytes)
  - DVMA requires setup but little intervention
  - setup can be non-trivial, requiring MMU setup/teardown

Typical Bandwidth Usage

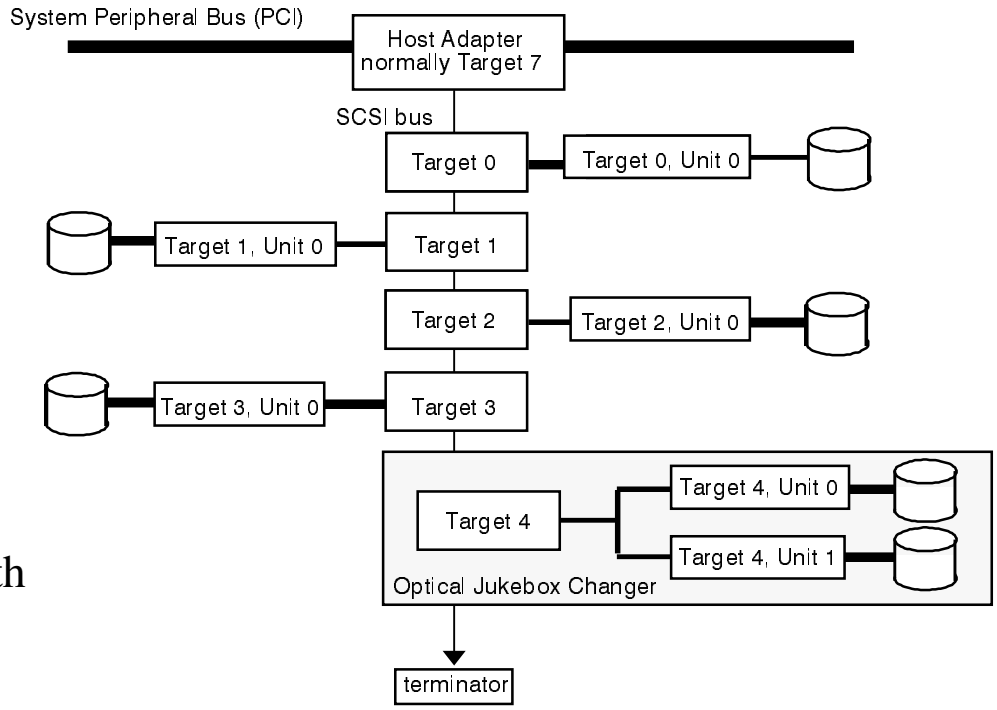
- Backplane busses
  - UltraSPARC, Pentium-II
    - 100-110 MB/sec per processor in “contended” apps
    - 25-30 MB/sec per processor in “shared nothing” apps
  - excludes I/O rate
- I/O busses
  - 20-40 MB/sec on PCI or Sbus (except graphics)
  - 3-5 MB/sec on ISA

Small Computer System Interface

- SCSI is by far the most common peripheral bus standard
- Design goals:
  - distribute device-specific knowledge into peripheral
  - multiple device support on single bus
    - disks, disk arrays, tapes, tape libraries, CD, DVD, scanners, printers, computers can all be connected
  - generic host bus adapter (HBA) support, classed device support

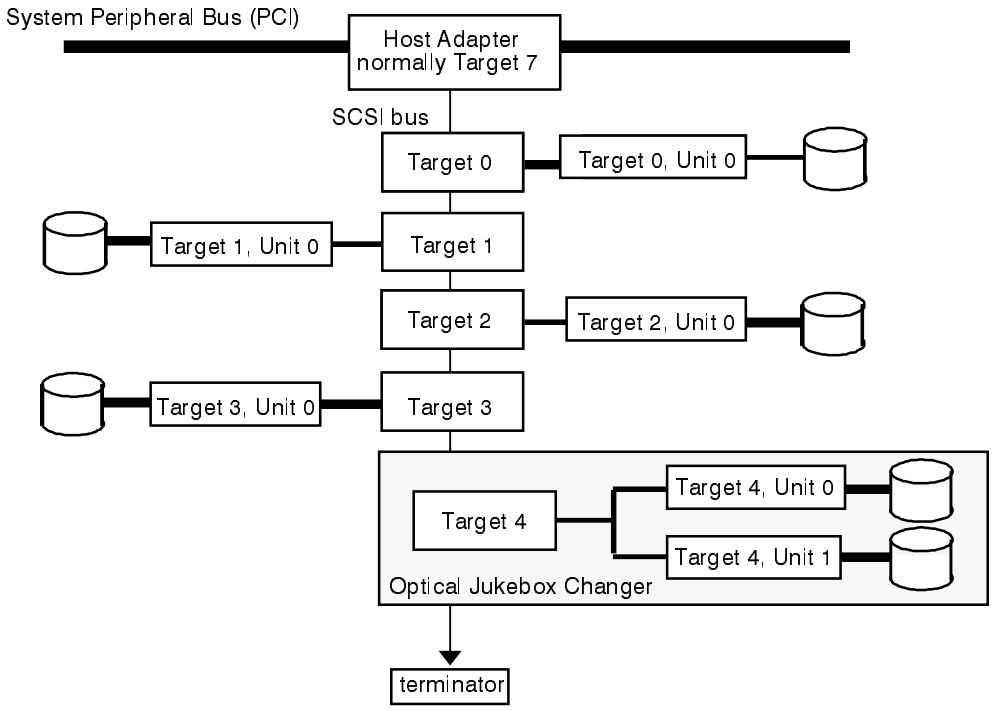
SCSI Organization

- Remote target always has an embedded controller
  - small computer with realtime operating system
  - Seagate ST52100 (2 GB disk) has a 16 MHz 80186, 640K RAM, 384K ROM
- Control and addressing are multiplexed on data lines
  - address limit depends on bus width
  - one available address per bus line
  - bandwidth efficiency ~90%

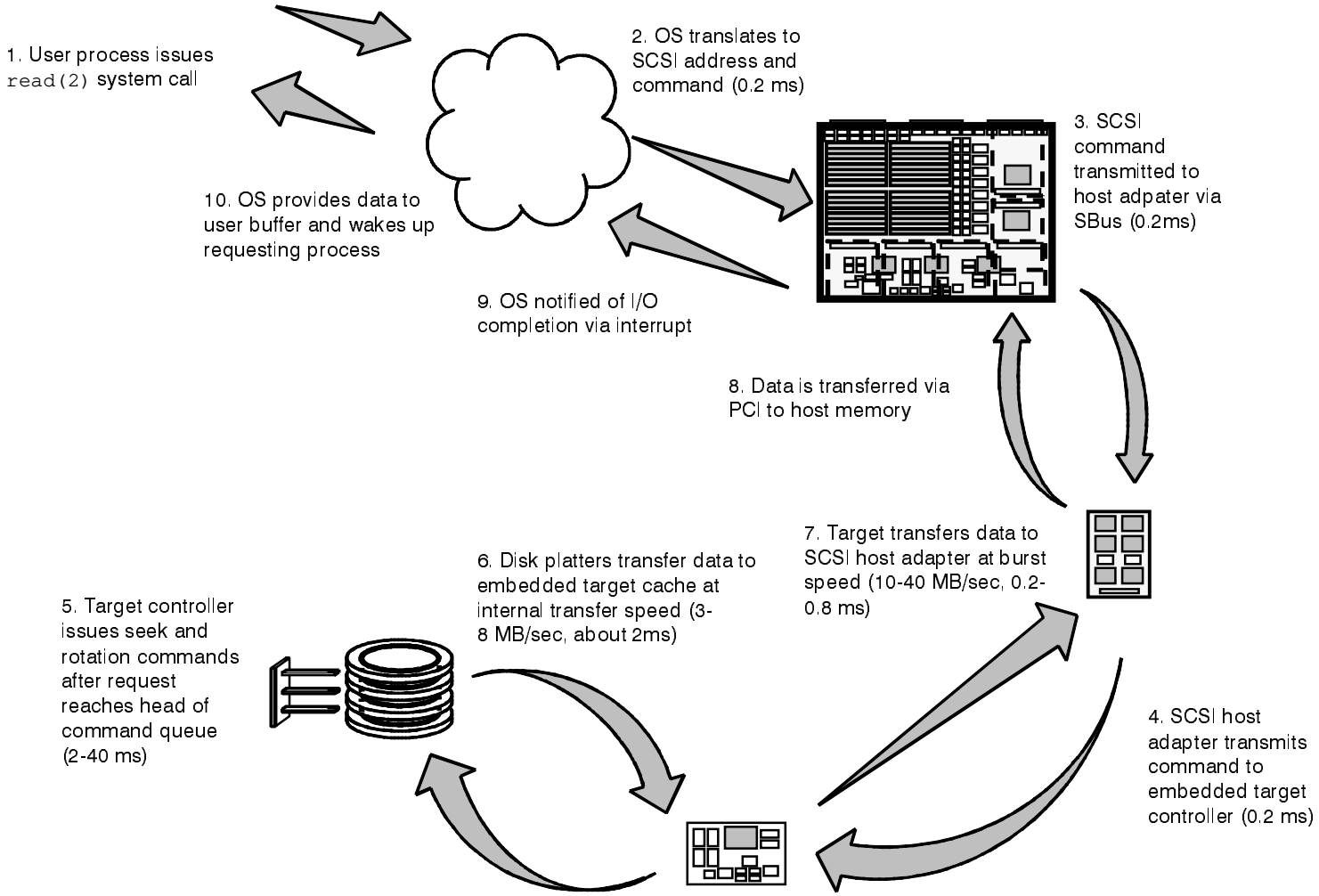


Flexibility and Bus Utilization

- Consider a scenario:
  - HBA is capable of fast-20/wide
  - Disk at target 0 is fast-20
  - Disk at target 1 is fast/wide
  - Tape at target 5 is fast
  - CDROM at target 6 is 1.2MB/sec async
- Does the bus run at 20 MB/sec, 10 MB/sec, 5 MB/sec or 1.2MB/sec?
  - Yes!



Half-Life of a SCSI I/O



SCSI-2

- The core specification in practice. It defines:
  - electrical signaling (single-ended or differential)
  - sync and async transfer modes
  - synchronous transfer @ 5 MHz, 8 bits = 5 MB/sec
  - bus protocol
  - options, all negotiated per target:
    - fast (10 MHz clock)
    - wide (16-bit or 32-bit)
    - command (tag) queuing
  - options can be combined (eg fast/wide = 20 MB/sec)

SCSI-3 ("UltraSCSI")

- Proper superset of SCSI-2
  - peripherals come up in simple async, then negotiate for options
  - new options:
    - fast-20 (20 MHz clock)
    - new electrical standard (low voltage differential)
  - alternate media (eg FibreChannel, many others)
- What's the value of SCSI-3 (as opposed to SCSI-2) in a PC or Workstation?

Disks

- Typical specifications
  - transfer speed
  - seek time
  - rotational speed
  - cache size

For example:

Seagate Barracuda-18 disk

- Formatted capacity of 18 gigabytes
- 100 Mbytes/sec external transfer rate
- 120-190 Mbits/sec platter transfer rate
- 7,200-RPM, 4.17 msec average latency
- 2048 KB multisegment cache
- 7.1 msec seek

Importance of Disk Optimization

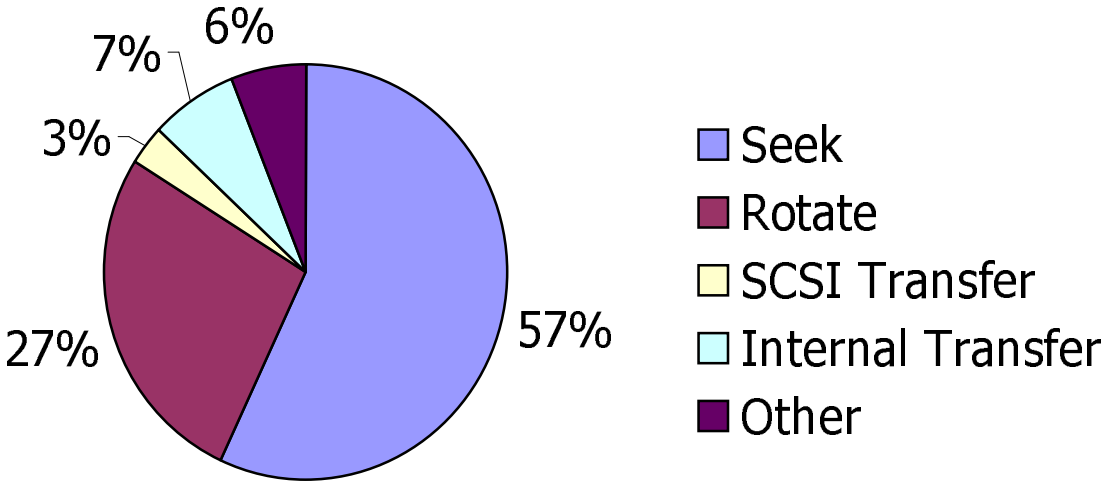
- Disk I/O service time takes 10-15 ms…
- but 15 ms is *15,000,000* nanoseconds!
- What can you do in 15,000,000 ns?
  - Modern cpus can execute 4 instructions in a clock cycle
  - Current 300 MHz processors cycle in 3+ns
  - So one disk I/O is equivalent to 455,000 instructions!
- *A disk I/O is a two-week paid vacation for a CPU*

Disk Geometry

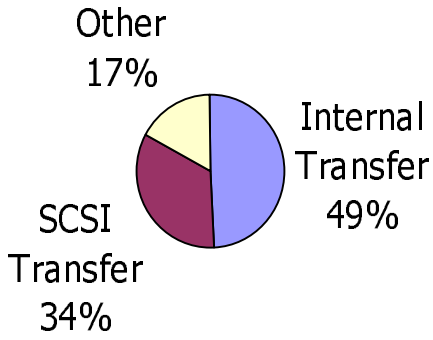
- Data location= <surface, track, sector>
  - selection of surface is instantaneous
  - selecting track requires seeking a disk arm (0.6-17 ms)
    - long seeks >> track-to-track
    - effect minimized on small platters
    - especially on physically large disks, long seeks are much longer than short seeks
    - clustering data is extremely productive
  - selecting sector requires rotational delay (4.1 ms @ 7200 rpm)

Breakdown of Disk I/O Time

Random I/O



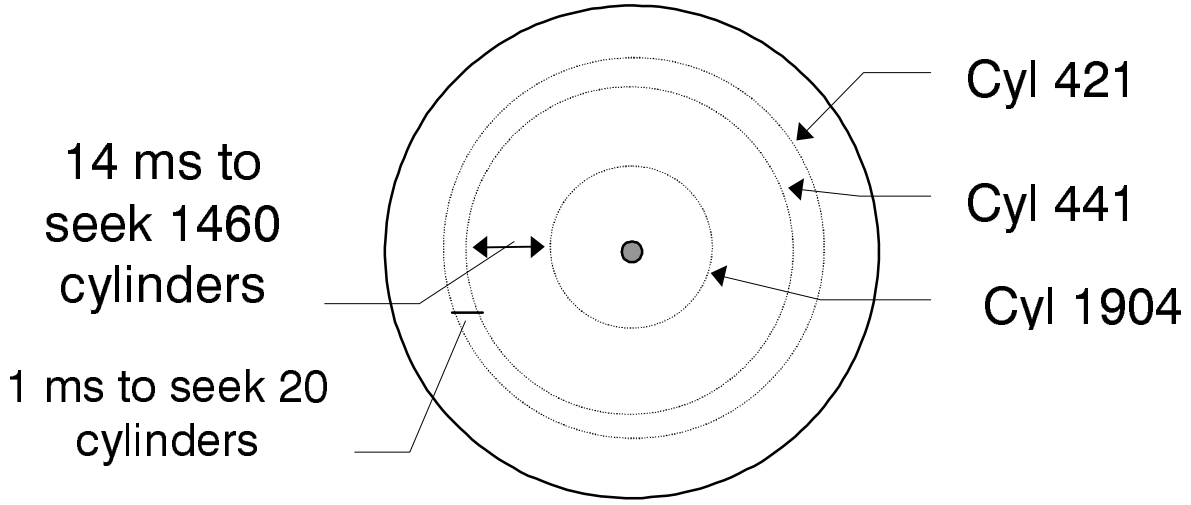
Sequential I/O



Note: relative sizes *are* proportional!

Physical Disk Optimization

- Random I/O dominated by seek, rotate
- Rotation speed is constant
- Seek time proportional to seek distance
  - track-to-track seek = ~ 0.6ms
  - max distance seek = ~ 17ms



Variable Density Recording

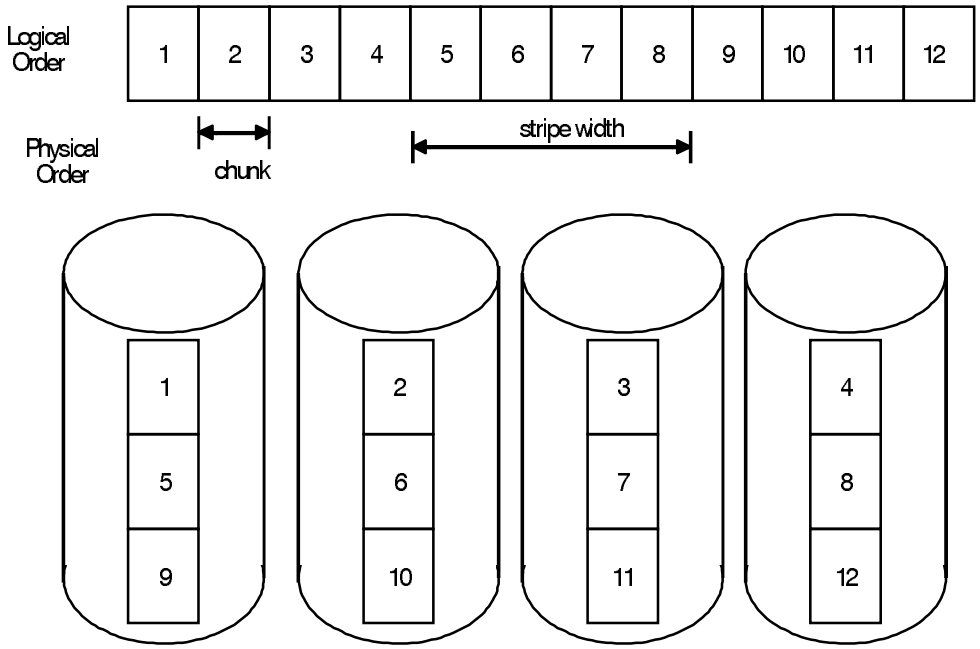
- Disks aren't constant speed or density
- Data recorded at equal density for all cyls
  - Outside cylinders holds more *and* transfers faster
  - ST32550W (2GB Barracuda-2LP) transfers 6.1-9 MB/sec
  - ST19171W (9GB Barracuda-9) transfers 10-15.5 MB/sec
- Seek time also reduced on outside cyls
  - given data fits in fewer cylinders
  - requires fewer and shorter seeks

Redundant Arrays of In Disks*

- In* = inexpensive, independent
- RAID = aggregations of disks
  - designed to overcome performance issues
  - also (maybe more importantly) to overcome reliability issues
- RAID is an architecture, not an enclosure

RAID-0: Striping

- Interleaves data across *members*
- The unit of data on each member is called the *chunk*, *segment size* or *interlace* (all equivalent terms)
- Developed to improve sequential transfer
- $MTBF_{raid\text{-}0} = MTBF_{disk} / N_{disks}$
  - 83k hours for 6-wide



RAID-0: analysis

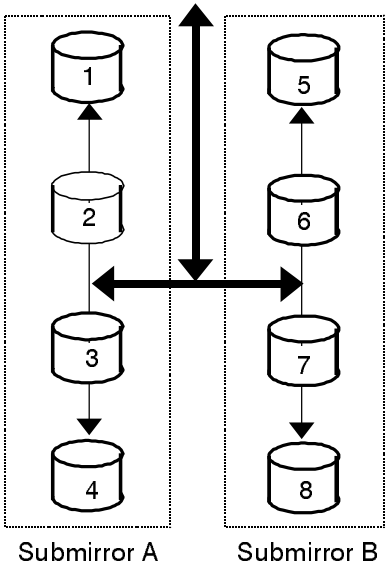
- Performance
  - Delivers great sequential throughput
  - Spreads data across members, improves random access
- Reliability
  - MTBF of RAID-0 much worse than single disks
  - failure of *any* member destroys the entire stripe
- Flexibility
  - Logical volume is much larger than single disk

RAID-1: Mirroring

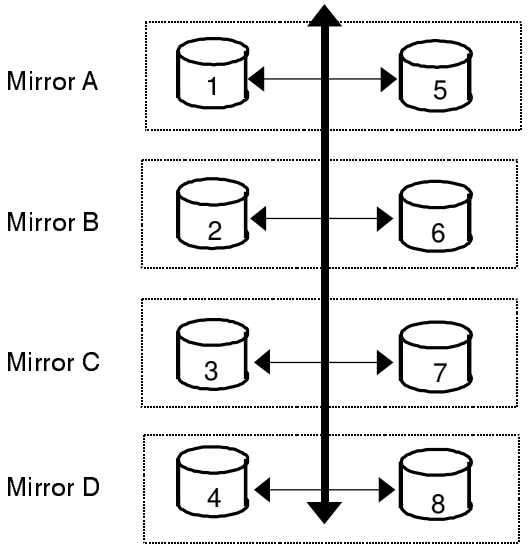
- Simple principle:
  - write each block on two (or more) member drives
- Performance:
  - About 15% degradation vs writes to single disk
  - Single thread reads = single disk
  - multithread random reads 2x single disk
- Reliability:
  - Clearly RAID-1 is far more reliable than a regular disk
  - Expensive. Requires 2N (or more) drives!
- Flexibility:
  - strictly speaking, applies only to single members

RAID-1+0, 0+1 Mirrored Stripes

- Indicates order of operation
  - 1+0 mirror first, stripe mirrors (*far* more failure resistant)
  - 0+1 stripe first, mirror the stripes



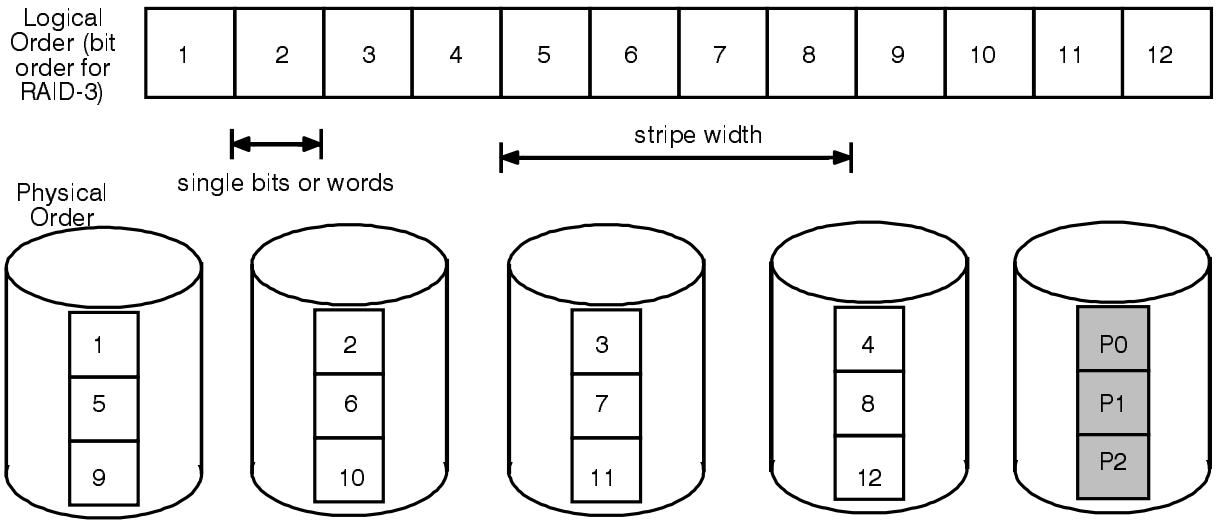
RAID-0+1



RAID-1+0

RAID-3: Striping with Parity

- Designed to improve RAID-0 reliability without using full duplication
- RAID-3 computes a XOR parity block for each stripe width and stores it on an extra disk drive
- XOR is reversible; if drive is missing, XOR the others to get missing data



$$P = c_0 \oplus c_1 \ldots \oplus c_{n-2} \oplus c_{n-1}$$

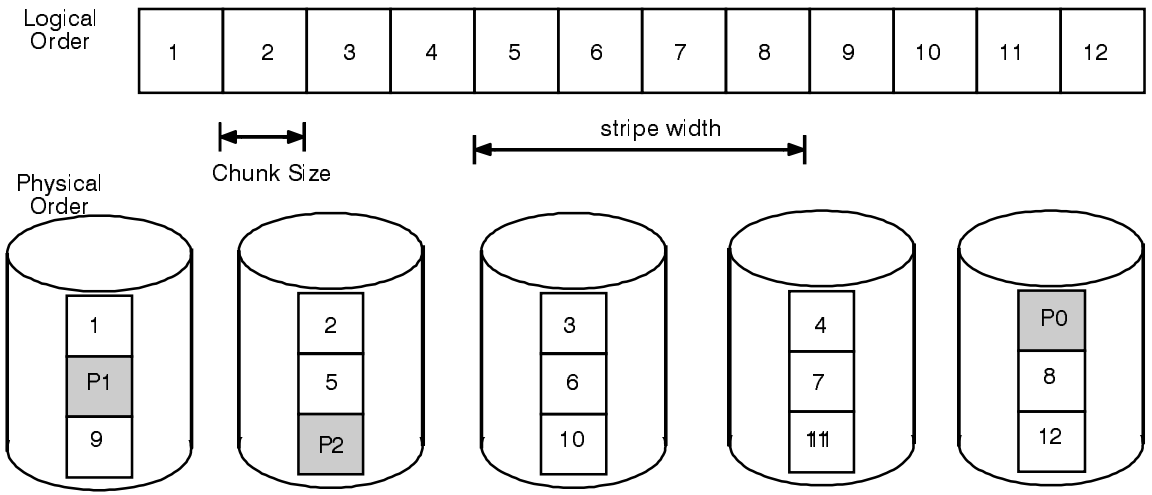
$$c_{missing} = P \oplus c_0 \oplus c_1 \oplus \cdots \oplus c_{missing\ -1} \oplus c_{missing\ +1} \oplus \cdots \oplus c_{n-1}$$

RAID-3: analysis

- Performance:
  - sequential: as good as N-1 RAID-0
  - random: as good as *one* disk (usually all spindles in sync)
- Reliability:
  - protects against one failure
  - 2nd failure loses entire volume
  - costs just 1 disk to provide protection
- Degraded mode:
  - writes OK (faster than normal mode)
  - reads require XOR recomputation

RAID-5: Rotated Parity

- Solves RAID-3 bottlenecks by rotating parity blocks across all members
- Most common form of RAID in production today, especially in smaller installations



RAID-5: analysis

- Performance:
  - sequential reads similar to RAID-0
  - writes typically involve read/modify/write cycles
    - must also maintain two-phase-commit log
    - r/m/w cycle causes *six* physical I/Os for every logical write
  - random I/O is *far* better than RAID-3 due to rotated parity
- Reliability:
  - protects against one member failure
  - costs just one member disk

RAID-5: more analysis

- Degraded mode:
  - Read to missing member requires N-1 reads, N-2 XORs! [Wow!]
  - Writes to missing member are faster than normal
- Reliability:
  - survives single failure
  - hot sparing improves RAID-1/3/5 reliability *substantially*
  - sometimes (often) multiple members on single controller

RAID-3 vs RAID-5

- RAID-5 is RAID-3 with rotating parity
  - RAID-3 handles sequential well but random poorly
  - RAID-5 handles sequential as well as RAID-3
    - same parity blocks and computation
    - handles random pretty well
- Most RAID-3 implementations are actually RAID-5!

RAID-6: Dual Parity

- RAID-6 attempts to provide better protection than RAID-5 for few additional members
- Two parity blocks are computed and spread across N+2 members.
- Only one major commercial implementation
- Has same basic characteristics as RAID-5

Accelerating RAID-5

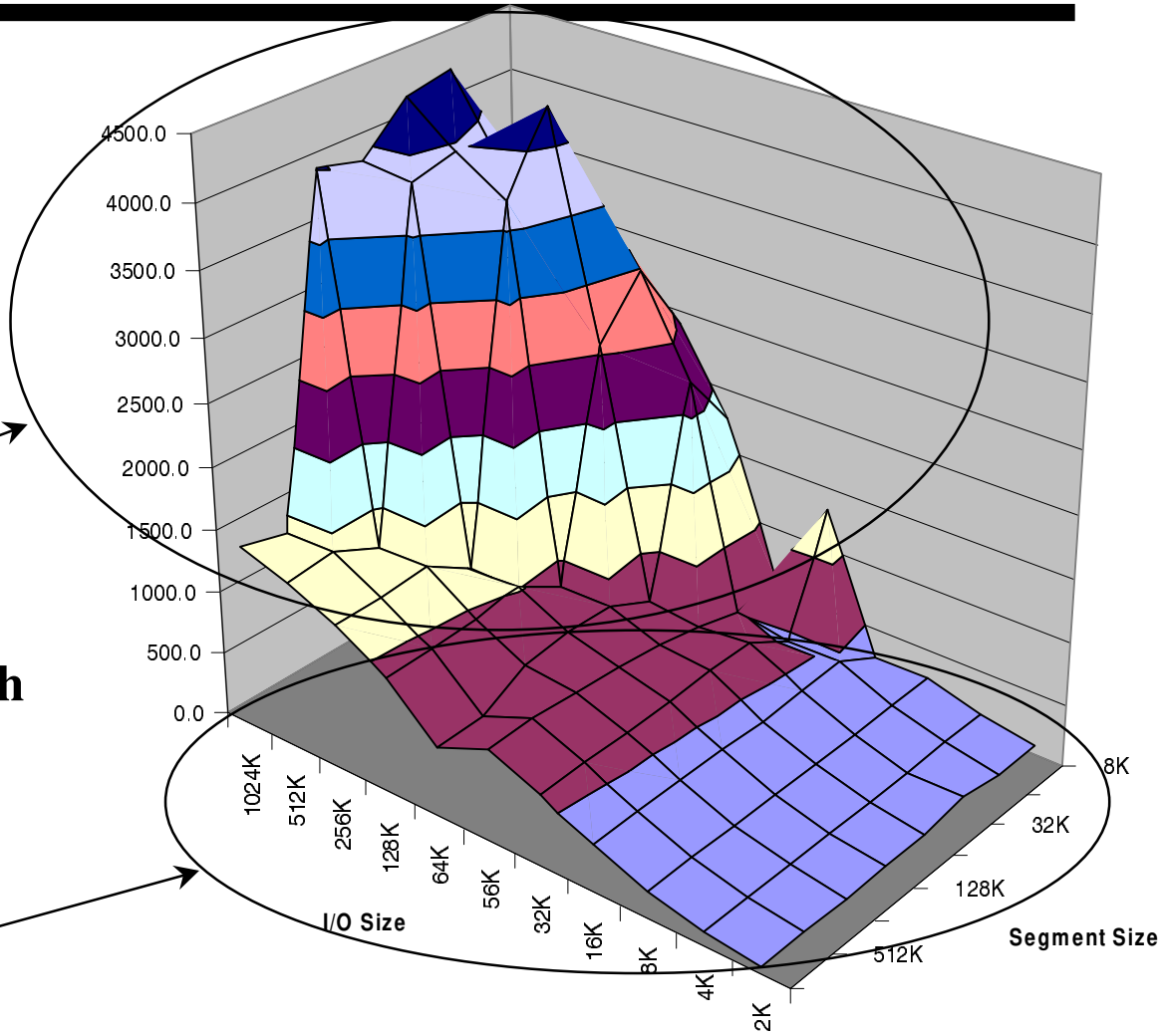
- A single RAID-5 logical write requires many I/Os to member disks
- Normally a write requires six member I/Os in 4 steps:
  - read old parity and data
  - write intent log
  - write new parity and data
  - mark intent complete in log
- Even worse, each step must wait until all previous are physically complete
  - I/O latency can be really, really bad!
  - Overhead can be really, really high!

Partial vs Full Stripe Writes

- Full-stripe writes are much faster than partial stripe writes (don't have to read old data and parity)

This part of the surface has I/O size >= stripe data width

This part of the surface has I/O size < stripe size, thus requiring read/modify/write cycles



Parity Cache

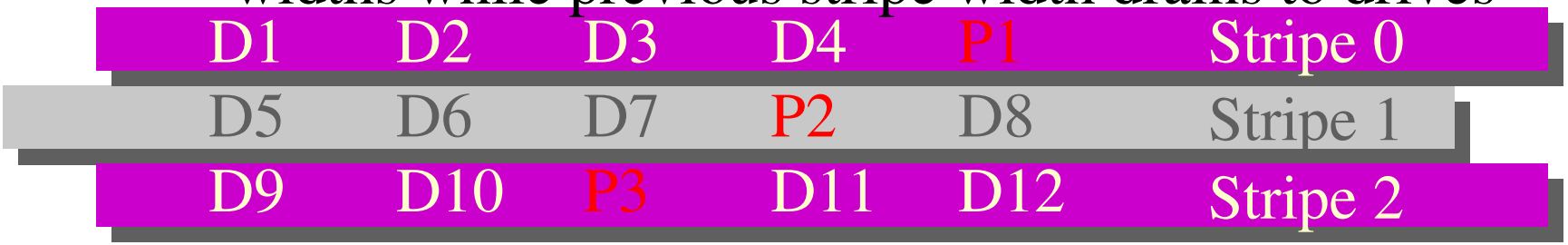
- For sequential writes, written data can be buffered in NVRAM to permit:
  - read parity and old data once rather than for every op
  - consider 2K I/Os to a 4+1 RAID-5 with 64K segment size
  - the first 32 I/Os all affect only D1 and P1 (below)
  - Saves re-reading parity and old data repetitively
- Primarily affects sequential I/O that is small compared to stripe data width
- Requires NVRAM buffer equal to 2x segment size

Stripe Consolidation

- For I/Os nearly as large as stripe data width:
  - caching the entire stripe width (eg D1, D2, D3, D4 below) in NVRAM avoids reading parity and old data at all
  - combines with parity cache above
- Affects sequential writes unless segment size is *very* large
- Requires NVRAM buffer space equal to twice stripe data width (~1.2MB for 8+1 with 128K segments)

Member Consolidation

- When I/O sizes are larger than stripe data width, this optimization improves disk utilization
  - Consider 4+1 LUN, 16K segment (thus stripe data width = 64K)
  - 128K write requires two writes to each member
  - Instead write once to each drive (eg D1+D5, D4+P2)
  - Might improve bandwidth, but same work gets done in one I/O instead of two
- Affects very large bulk I/O (eg TPC-D, HPC)
- NVRAM must be large enough to hold pending stripe widths while previous stripe width drains to drives



Write Latency Hiding

- Parity cache, stripe consolidation and member consolidation don't help random writes much
- NVRAM buffer helps by committing write into NVRAM
- Host and app perceive fast "write"
  - member I/Os (r/m/w) happen while app computes next I/O
- Note: NVRAM buffer need not be large as this class by definition is random I/O and low throughput
  - buffering one drive worth = 120 IOPs x 8K = 960KB
  - 128 MB can handle 130+ drives without overflow

Log Acceleration

- Unaccelerated RAID-5 requires an intent log
  - without the log, the LUN becomes corrupt if a logical write operation is interrupted before completion
  - Log must be written after parity computation and before writing new data and new parity
- NVRAM buffer can be used to store log
  - much faster than on-disk log
  - doesn't consume disk utilization
- Requires 2x segment size for each pending write
  - 128MB permits 1024 pending writes if 128KB segments; typical pending = ~10-12 if large I/O, ~140-200 if small

Dirty Region Log Utilization

- RAID-1 and RAID-1+0 mirrors use Dirty Region Logs to improve recovery of planned mirror detach
- Each segment size block of each member of the logical disk has a bit in a LUN-wide mask that is written if the corresponding block is modified during detach
- mirrored NVRAM DRL eliminates most write-backs to member disks for DRL update (updated every 10 seconds instead of every write).
- Uses 1MB for a 17+17 LUN, 9GB disk, 16K segment

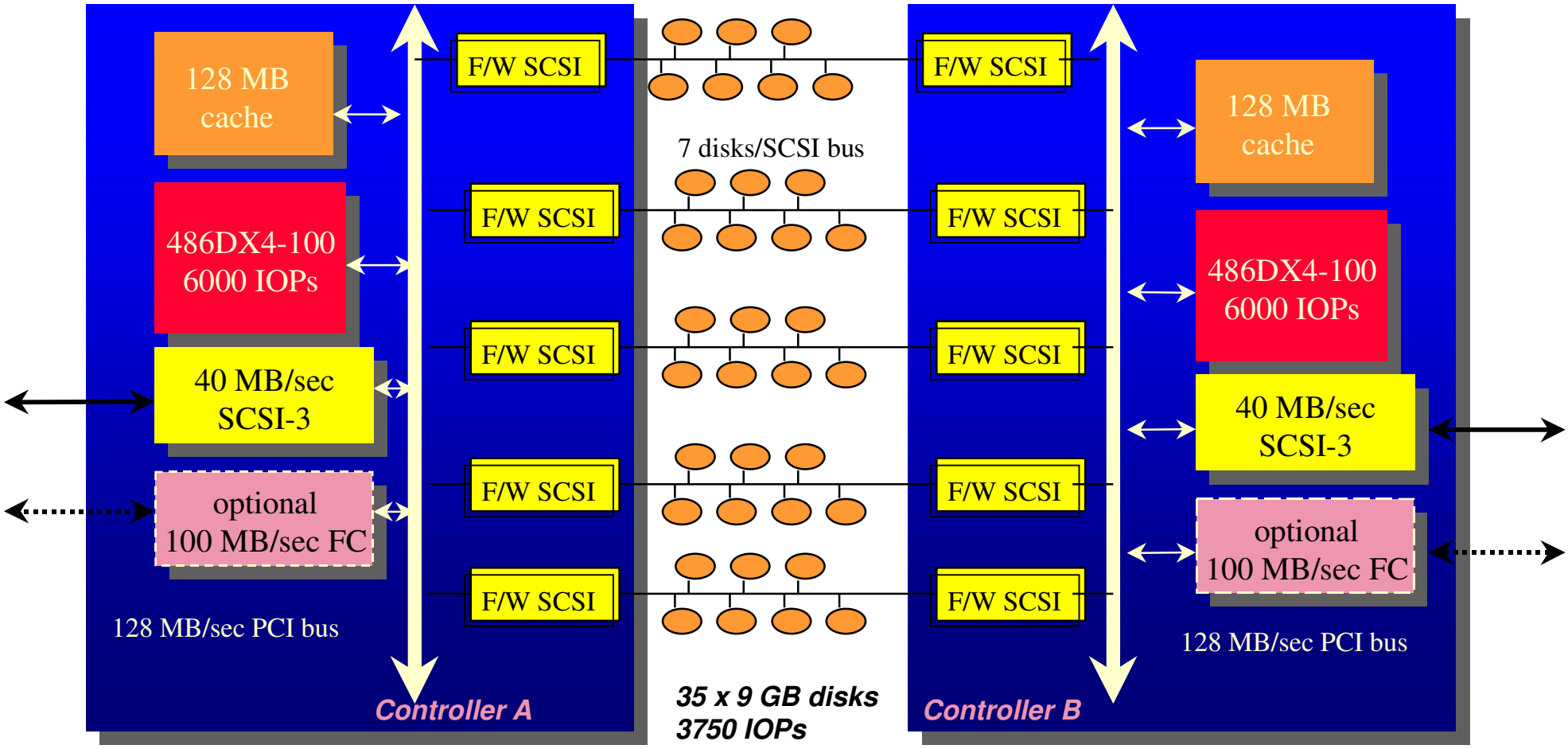
Application I/O Profiles

- Commercial data processing
  - typically indexed, random access
  - I/O size is small (2KB) because records are small
- Decision support - transaction set analysis
  - majority of I/O is sequential by user
  - I/O size is very large (64KB or more) for efficiency
    - might have to read an entire table (could be 1TB+)
  - Read mostly
- File Servers
  - file system I/O size is moderate (8K)
  - access is sequential by user but interleaved (so random) at drives

Common Implementations

- 4-128 drives are mounted in a box or rack
- 1-16 SCSI busses connect member disks
- member disks managed by a local controller
  - typically 100 MHz 486, 50 MHz i960, 120 MHz PPC 603e
  - controller also manages NVRAM cache
- controllers are replicated
- 2-16 SCSI or FibreChannel host connections

Typical 1997 Disk Array



36+ MB/sec sustained

36+ MB/sec sustained

Common Misconceptions

- "RAID boxes are inherently faster than regular disks"
  - nope: it's the RAID organization
  - in terms of latency, RAID boxes may even be slower than disk
- "RAID organizations check data on read"
  - nope: RAID-1/1+0 don't check on read
  - RAID-3/5 don't check on read unless a member reports error
  - member disks store ~20% ECC data, so reported errors are BIG

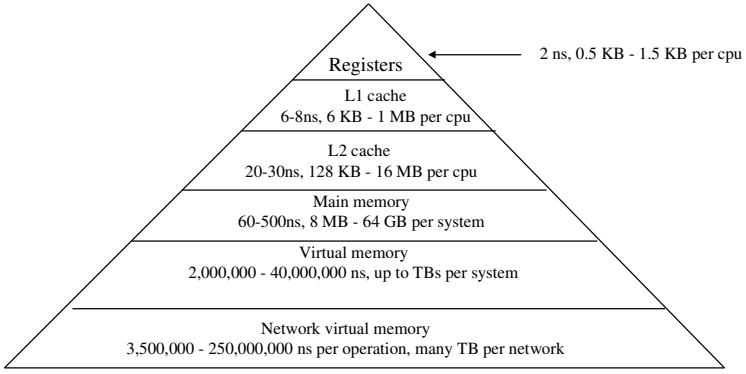
Applied Computer Architecture

Brian Wong
Chief Scientist, Enterprise Engineering
Sun Microsystems
November 6, 1997
blw@eng.sun.com

Two Day Agenda

- Applied architecture in the processor complex
- Busses and Interconnects
- Disks
- RAID levels (0, 1, 3, 1+0, 0+1, 5, 6)
- RAID optimization

The Virtual Memory Hierarchy



Effect of the Hierarchy

- End-user application performance is the sum of:
  - processor pipeline performance (~85-98% of the time)
  - E$ performance (most of the rest)
  - memory subsystem performance for cache misses
    - some apps miss E$ up to 10-15% of the time
    - long-latency machines have penalties of 100-200 cycles
    - even longer for NUMA machines
  - disk subsystem performance on page faults

Pipelining in the Real World

- How effective are superscalar pipelines?
- What's a good IPC?
- It depends
  - IPC ratings are > 2.0
    - assume execution from I$, D$
    - hand-rolled assembly code
  - sustained IPC > 1.6 is entirely possible
    - floating point codes
    - good compilers
    - well-behaved codes (e.g. matrix multiply)

Less Efficient Applications

- IPC of < 1 is not theoretical
  - typical DBMS codes average < 1.0 instr/cycle
  - even with good compilers
  - I$ miss rate can be very high
- IPC of << 1.0 is not theoretical!
  - Virtually any app that processes linked lists average << 1.0 cpi
  - E$ misses galore!

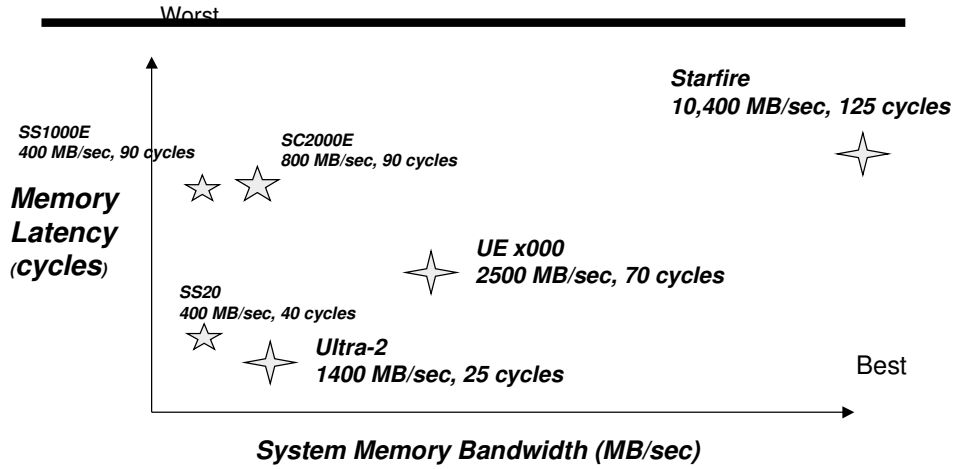
```
#define MAXSIZE 128
struct list {
  integer key;
  char *name;
  char *address;
  char *address2;
  char *city;
  char *state;
  char *country;
  char *postalCode;
  char *phone;
  char *fax;
  char blockData [MAXSIZE];
  struct list *nextp, *prevp;
};
```

nextp, prevp are in a different cache line than the key

Memory Latency

- The amount of time the combined CPU, memory and centerplane require to service a cache miss
- Higher bandwidth requires more (worse) latency
  - Ultra Enterprise 10000 is like a 747: it carries 400 people 8000 miles at 600mph, but you have to park at the airport, hire a car, wait for the flight, drive downtown from the airport.
  - Ultra-2 is like a Porsche 911: it carries only two people for only 500 miles at only 100mph, but you can get downtown directly without waiting.
  - Which would you take to go to San Jose, CA? San Jose Puerto Rico?

Memory Latency



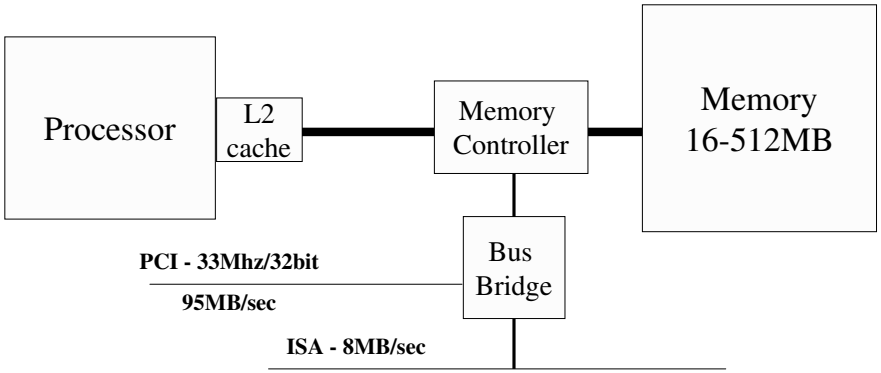
Busses and Interconnects

- a way to get from one place to another
  - in a computer they're like roads and blood vessels
  - a bunch of wires going the same place with a defined protocol
- there are *many* design differences
- four major classes of bus
  - backplane (processor-memory)
  - inter-node interconnect (between members of a NUMA cluster)
  - I/O (eg PCI, PC-MCIA/PC-Card, MicroChannel, Sbus)
  - peripheral (SCSI, FibreChannel, USB)

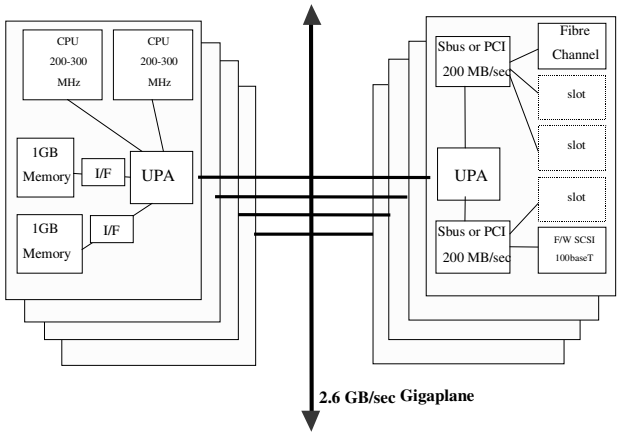
Design Tradeoffs in Busses

- overall speed
- parallelism (between bits, users, transactions)
- physical length
- configurability
- speed of various users (perhaps speed-matching)
  - cpu/memory - 100+ MB/sec per cpu
  - keyboard - 15 bytes/sec
  - disk - 0.2-200 MB/sec
  - graphics - 500+MB/sec
- cost (!)

"Thin" Architecture



A Larger and Wider System



Sun Ultra Enterprise x000

Bus/Interconnect Comparison

Bus	Date	Width	Clock	Theoretical	Achievable	Efficiency
MBus	1992	64	40	320	135	42%
XDBus	1992	64	40	320	270	84%
UPA	1995	128	83.3	1,333	1,220	92%
Gigaplane	1995	256	83.3	2,667	2,500	94%
Gigaplane XB	1997	1024	83.3	10,667	10,000	94%
VME	1979	32	10	40	25	63%
VME64	1988	64	10	80	55	69%
SBus	1989	32	25	100	65	65%
SBus-64	1994	64	25	200	105	53%
ISA	1981	8	8.3	8	5	60%
MicroChannel	1987	32	10	40	21	53%
EISA	1988	32	8.3	33	15	45%
PCI	1993	32	33.3	133	90	68%
PCI64	1997	64	33.3	267		
EPCI		64	66.7	533		
SCSI-1	1984	8	5	5	1.2	24%
SCSI-2	1989	8	5	5	4.4	88%
SCSI-2 fast/wide	1990	16	10	20	18.4	92%
SCSI-3 ultra/wide	1996	16	20	40	36.5	91%
FibreChannel	1993	1	266	25	22	88%
FC-100	1997	1	1064	100	90	90%

Bus Interconnect Comparison

Bus	Date	Width	Clock	Theoretical	Achievable	Efficiency	Notes
MBus	1992	64	40	320	135	42%	circuit-switched
XDBus	1992	64	40	320	270	84%	packet-switched
UPA	1995	128	83.3	1,333	1,220	92%	independent control, address, data
Gigaplane	1995	256	83.3	2,667	2,500	94%	both UPA and Gigaplane packet-switched
Gigaplane XB	1997	1024	83.3	10,667	10,000	94%	Crossbar switch using Gigaplanes
VME	1979	32	10	40	25	63%	
VME64	1988	64	10	80	55	69%	
SBus	1989	32	25	100	65	65%	but only 8 MB/sec in 1-byte PIO mode
SBus-64	1994	64	25	200	105	53%	SBus-64 multiplexes address lines
ISA	1981	8	8.3	8	5	60%	
MicroChannel	1987	32	10	40	21	53%	
EISA	1988	32	8.3	33	15	45%	
PCI	1993	32	33.3	133	90	68%	
PCI64	1997	64	33.3	267			only just shipping
EPCI		64	66.7	533			not yet commercialized
SCSI-1	1984	8	5	5	1.2	24%	async only; sync mode didn't work!
SCSI-2	1989	8	5	5	4.4	88%	synchronous transfers
SCSI-2 fast/wide	1990	16	10	20	18.4	92%	
SCSI-3 ultra/wide	1996	16	20	40	36.5	91%	
FibreChannel	1993	1	266	25	22	88%	full duplex - cited speeds for one direction
FC-100	1997	1	1064	100	90	90%	full duplex - cited speeds for one direction

Bus Protocol

- Three logical phases
  - arbitration - who gets to use the bus next
    - sometimes called control phase
  - address - what data will we be transferring?
  - Data - move the data itself
    - might be nil
- There are other bus transactions - error report, probe, etc
- Busses primarily differentiated by design in the three primary phases

Bus Design Choices

- Circuit-switched vs. packet-switched
  - circuit-switched: end-to-end connection stays up
  - packet-switched: disconnects during service, must be reconnected
- Shared vs. separate control, address, data lines
  - shared: fewer wires (much less $$$)
  - separate: can run three separate transactions simultaneously
- Data burst size
- Data transfer mode

Data Transfer Modes

- Programmed I/O (PIO)
  - processor directly issues load and store instructions to copy
  - uses processor cycles but requires no setup
  - pollutes the D$ and E$ because data moves directly through CPU
  - very simple
- Direct Virtual Memory Access (DVMA)
  - dedicated circuitry handles transfer mechanics (can be expensive)
  - operates in 4-, 8- and 64-byte blocks (usually 64-bytes)
  - DVMA requires setup but little intervention
  - setup can be non-trivial, requiring MMU setup/teardown

Typical Bandwidth Usage

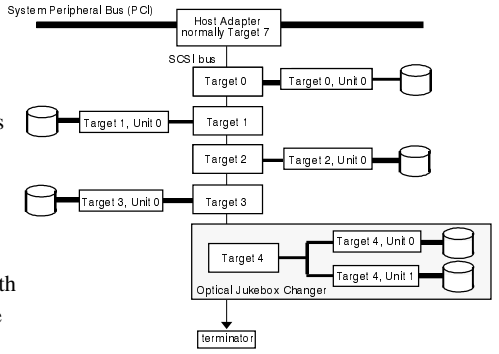
- Backplane busses
  - UltraSPARC, Pentium-II
    - 100-110 MB/sec per processor in “contended” apps
    - 25-30 MB/sec per processor in “shared nothing” apps
  - excludes I/O rate
- I/O busses
  - 20-40 MB/sec on PCI or Sbus (except graphics)
  - 3-5 MB/sec on ISA

Small Computer System Interface

- SCSI is by far the most common peripheral bus standard
- Design goals:
  - distribute device-specific knowledge into peripheral
  - multiple device support on single bus
    - disks, disk arrays, tapes, tape libraries, CD, DVD, scanners, printers, computers can all be connected
  - generic host bus adapter (HBA) support, classed device support

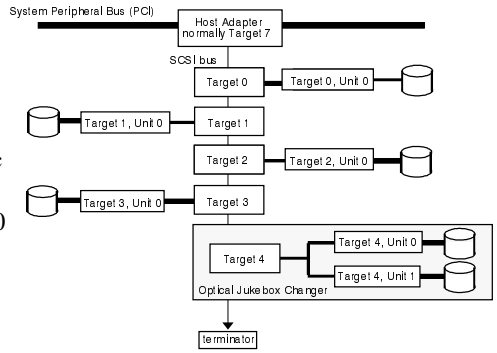
SCSI Organization

- Remote target always has an embedded controller
  - small computer with realtime operating system
  - Seagate ST52100 (2 GB disk) has a 16 MHz 80186, 640K RAM, 384K ROM
- Control and addressing are multiplexed on data lines
  - address limit depends on bus width
  - one available address per bus line
  - bandwidth efficiency ~90%

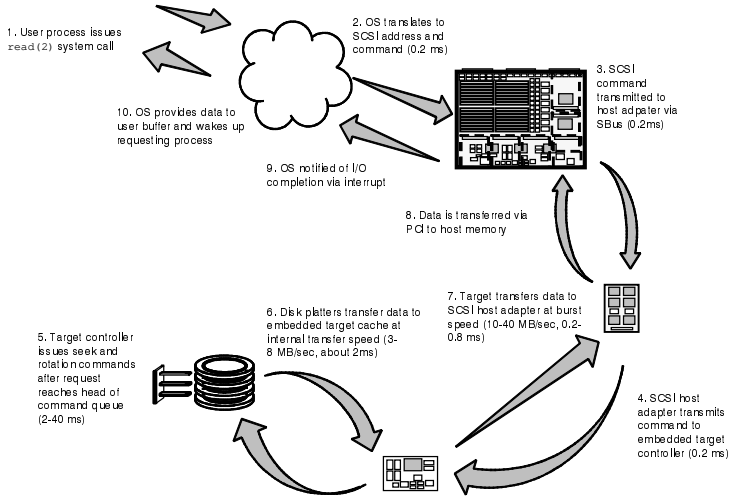


Flexibility and Bus Utilization

- Consider a scenario:
  - HBA is capable of fast-20/wide
  - Disk at target 0 is fast-20
  - Disk at target 1 is fast/wide
  - Tape at target 5 is fast
  - CDROM at target 6 is 1.2MB/sec async
- Does the bus run at 20 MB/sec, 10 MB/sec, 5 MB/sec or 1.2MB/sec?
  - Yes!



Half-Life of a SCSI I/O



SCSI-2

- The core specification in practice. It defines:
  - electrical signaling (single-ended or differential)
  - sync and async transfer modes
  - synchronous transfer @ 5 MHz, 8 bits = 5 MB/sec
  - bus protocol
  - options, all negotiated per target:
    - fast (10 MHz clock)
    - wide (16-bit or 32-bit)
    - command (tag) queuing
  - options can be combined (eg fast/wide = 20 MB/sec)

SCSI-3 ("UltraSCSI")

- Proper superset of SCSI-2
  - peripherals come up in simple async, then negotiate for options
  - new options:
    - fast-20 (20 MHz clock)
    - new electrical standard (low voltage differential)
  - alternate media (eg FibreChannel, many others)
- What's the value of SCSI-3 (as opposed to SCSI-2) in a PC or Workstation?

Disks

- Typical specifications
  - transfer speed
  - seek time
  - rotational speed
  - cache size

For example:

Seagate Barracuda-18 disk

- Formatted capacity of 18 gigabytes
- 100 Mbytes/sec external transfer rate
- 120-190 Mbits/sec platter transfer rate
- 7,200-RPM, 4.17 msec average latency
- 2048 KB multisegment cache
- 7.1 msec seek

Importance of Disk Optimization

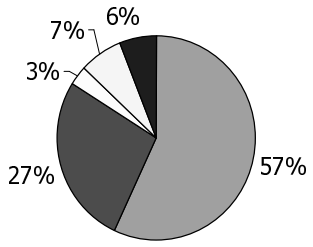
- Disk I/O service time takes 10-15 ms…
- but 15 ms is *15,000,000* nanoseconds!
- What can you do in 15,000,000 ns?
  - Modern cpus can execute 4 instructions in a clock cycle
  - Current 300 MHz processors cycle in 3+ns
  - So one disk I/O is equivalent to 455,000 instructions!
- *A disk I/O is a two-week paid vacation for a CPU*

Disk Geometry

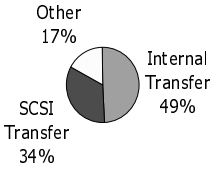
- Data location= <surface, track, sector>
  - selection of surface is instantaneous
  - selecting track requires seeking a disk arm (0.6-17 ms)
    - long seeks >> track-to-track
    - effect minimized on small platters
    - especially on physically large disks, long seeks are much longer than short seeks
    - clustering data is extremely productive
  - selecting sector requires rotational delay (4.1 ms @ 7200 rpm)

Breakdown of Disk I/O Time

Random I/O



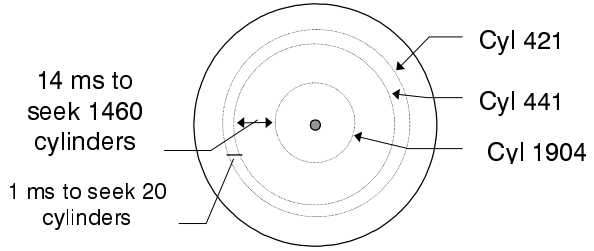
Sequential I/O



Note: relative sizes *are* proportional!

Physical Disk Optimization

- Random I/O dominated by seek, rotate
- Rotation speed is constant
- Seek time proportional to seek distance
  - track-to-track seek = ~ 0.6ms
  - max distance seek = ~ 17ms



Variable Density Recording

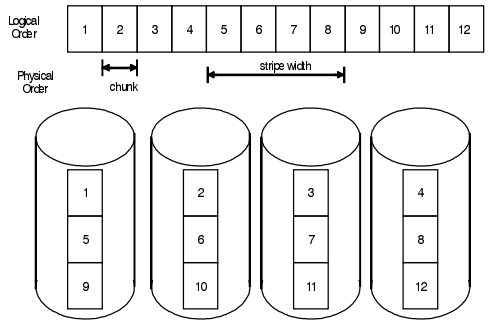
- Disks aren't constant speed or density
- Data recorded at equal density for all cyls
  - Outside cylinders holds more *and* transfers faster
  - ST32550W (2GB Barracuda-2LP) transfers 6.1-9 MB/sec
  - ST19171W (9GB Barracuda-9) transfers 10-15.5 MB/sec
- Seek time also reduced on outside cyls
  - given data fits in fewer cylinders
  - requires fewer and shorter seeks

Redundant Arrays of In Disks*

- In* = inexpensive, independent
- RAID = aggregations of disks
  - designed to overcome performance issues
  - also (maybe more importantly) to overcome reliability issues
- RAID is an architecture, not an enclosure

RAID-0: Striping

- Interleaves data across *members*
- The unit of data on each member is called the *chunk*, *segment size* or *interlace* (all equivalent terms)
- Developed to improve sequential transfer
- $MTBF_{raid\text{-}0} = MTBF_{disk} / N_{disks}$
  - 83k hours for 6-wide



RAID-0: analysis

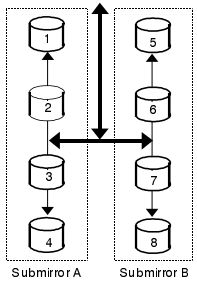
- Performance
  - Delivers great sequential throughput
  - Spreads data across members, improves random access
- Reliability
  - MTBF of RAID-0 much worse than single disks
  - failure of *any* member destroys the entire stripe
- Flexibility
  - Logical volume is much larger than single disk

RAID-1: Mirroring

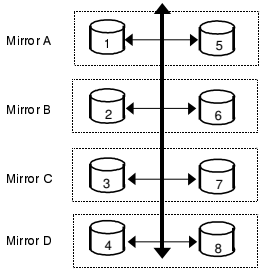
- Simple principle:
  - write each block on two (or more) member drives
- Performance:
  - About 15% degradation vs writes to single disk
  - Single thread reads = single disk
  - multithread random reads 2x single disk
- Reliability:
  - Clearly RAID-1 is far more reliable than a regular disk
  - Expensive. Requires 2N (or more) drives!
- Flexibility:
  - strictly speaking, applies only to single members

RAID-1+0, 0+1 Mirrored Stripes

- Indicates order of operation
  - 1+0 mirror first, stripe mirrors (*far* more failure resistant)
  - 0+1 stripe first, mirror the stripes



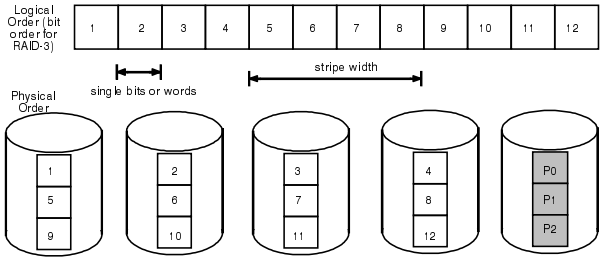
RAID-0+1



RAID-1+0

RAID-3: Striping with Parity

- Designed to improve RAID-0 reliability without using full duplication
- RAID-3 computes a XOR parity block for each stripe width and stores it on an extra disk drive
- XOR is reversible; if drive is missing, XOR the others to get missing data



$$P = c_0 \oplus c_1 \ldots \oplus c_{n-2} \oplus c_{n-1}$$

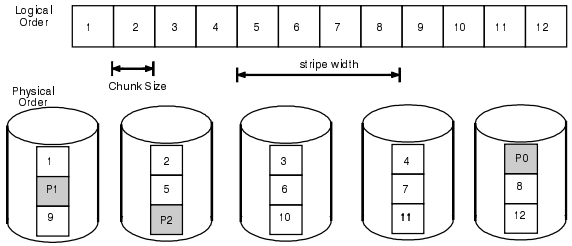
$$c_{missing} = P \oplus c_0 \oplus c_1 \oplus \cdots \oplus c_{missing\,-1} \oplus c_{missing\,+1} \oplus \cdots \oplus c_{n-1}$$

RAID-3: analysis

- Performance:
  - sequential: as good as N-1 RAID-0
  - random: as good as *one* disk (usually all spindles in sync)
- Reliability:
  - protects against one failure
  - 2nd failure loses entire volume
  - costs just 1 disk to provide protection
- Degraded mode:
  - writes OK (faster than normal mode)
  - reads require XOR recomputation

RAID-5: Rotated Parity

- Solves RAID-3 bottlenecks by rotating parity blocks across all members
- Most common form of RAID in production today, especially in smaller installations



RAID-5: analysis

- Performance:
  - sequential reads similar to RAID-0
  - writes typically involve read/modify/write cycles
    - must also maintain two-phase-commit log
    - r/m/w cycle causes *six* physical I/Os for every logical write
  - random I/O is *far* better than RAID-3 due to rotated parity
- Reliability:
  - protects against one member failure
  - costs just one member disk

RAID-5: more analysis

- Degraded mode:
  - Read to missing member requires N-1 reads, N-2 XORs! [Wow!]
  - Writes to missing member are faster than normal
- Reliability:
  - survives single failure
  - hot sparing improves RAID-1/3/5 reliability *substantially*
  - sometimes (often) multiple members on single controller

RAID-3 vs RAID-5

- RAID-5 is RAID-3 with rotating parity
  - RAID-3 handles sequential well but random poorly
  - RAID-5 handles sequential as well as RAID-3
    - same parity blocks and computation
    - handles random pretty well
- Most RAID-3 implementations are actually RAID-5!

RAID-6: Dual Parity

- RAID-6 attempts to provide better protection than RAID-5 for few additional members
- Two parity blocks are computed and spread across N+2 members.
- Only one major commercial implementation
- Has same basic characteristics as RAID-5

Accelerating RAID-5

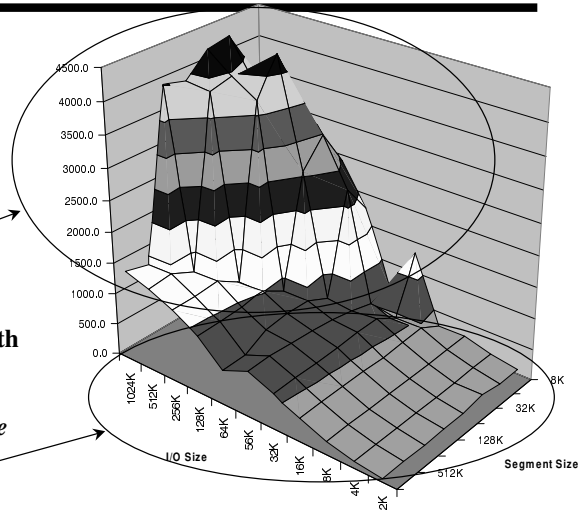
- A single RAID-5 logical write requires many I/Os to member disks
- Normally a write requires six member I/Os in 4 steps:
  - read old parity and data
  - write intent log
  - write new parity and data
  - mark intent complete in log
- Even worse, each step must wait until all previous are physically complete
  - I/O latency can be really, really bad!
  - Overhead can be really, really high!

Partial vs Full Stripe Writes

- Full-stripe writes are much faster than partial stripe writes (don't have to read old data and parity)

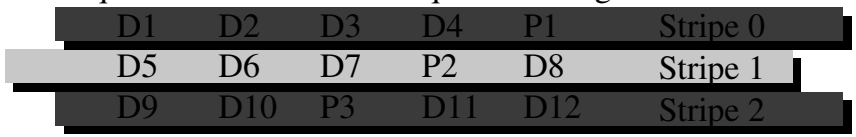
This part of the surface has I/O size >= stripe data width

This part of the surface has I/O size < stripe size, thus requiring read/modify/write cycles



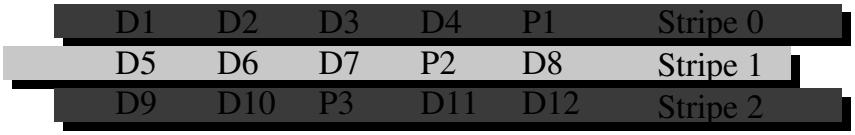
Parity Cache

- For sequential writes, written data can be buffered in NVRAM to permit:
  - read parity and old data once rather than for every op
  - consider 2K I/Os to a 4+1 RAID-5 with 64K segment size
  - the first 32 I/Os all affect only D1 and P1 (below)
  - Saves re-reading parity and old data repetitively
- Primarily affects sequential I/O that is small compared to stripe data width
- Requires NVRAM buffer equal to 2x segment size



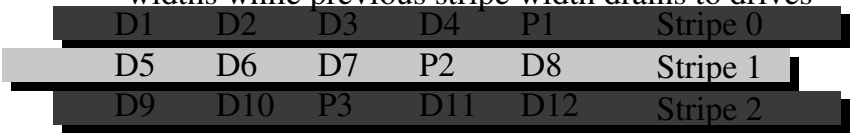
Stripe Consolidation

- For I/Os nearly as large as stripe data width:
  - caching the entire stripe width (eg D1, D2, D3, D4 below) in NVRAM avoids reading parity and old data at all
  - combines with parity cache above
- Affects sequential writes unless segment size is *very* large
- Requires NVRAM buffer space equal to twice stripe data width (~1.2MB for 8+1 with 128K segments)



Member Consolidation

- When I/O sizes are larger than stripe data width, this optimization improves disk utilization
  - Consider 4+1 LUN, 16K segment (thus stripe data width = 64K)
  - 128K write requires two writes to each member
  - Instead write once to each drive (eg D1+D5, D4+P2)
  - Might improve bandwidth, but same work gets done in one I/O instead of two
- Affects very large bulk I/O (eg TPC-D, HPC)
- NVRAM must be large enough to hold pending stripe widths while previous stripe width drains to drives



Write Latency Hiding

- Parity cache, stripe consolidation and member consolidation don't help random writes much
- NVRAM buffer helps by committing write into NVRAM
- Host and app perceive fast "write"
  - member I/Os (r/m/w) happen while app computes next I/O
- Note: NVRAM buffer need not be large as this class by definition is random I/O and low throughput
  - buffering one drive worth = 120 IOPs x 8K = 960KB
  - 128 MB can handle 130+ drives without overflow

Log Acceleration

- Unaccelerated RAID-5 requires an intent log
  - without the log, the LUN becomes corrupt if a logical write operation is interrupted before completion
  - Log must be written after parity computation and before writing new data and new parity
- NVRAM buffer can be used to store log
  - much faster than on-disk log
  - doesn't consume disk utilization
- Requires 2x segment size for each pending write
  - 128MB permits 1024 pending writes if 128KB segments; typical pending = ~10-12 if large I/O, ~140-200 if small

Dirty Region Log Utilization

- RAID-1 and RAID-1+0 mirrors use Dirty Region Logs to improve recovery of planned mirror detach
- Each segment size block of each member of the logical disk has a bit in a LUN-wide mask that is written if the corresponding block is modified during detach
- mirrored NVRAM DRL eliminates most write-backs to member disks for DRL update (updated every 10 seconds instead of every write).
- Uses 1MB for a 17+17 LUN, 9GB disk, 16K segment

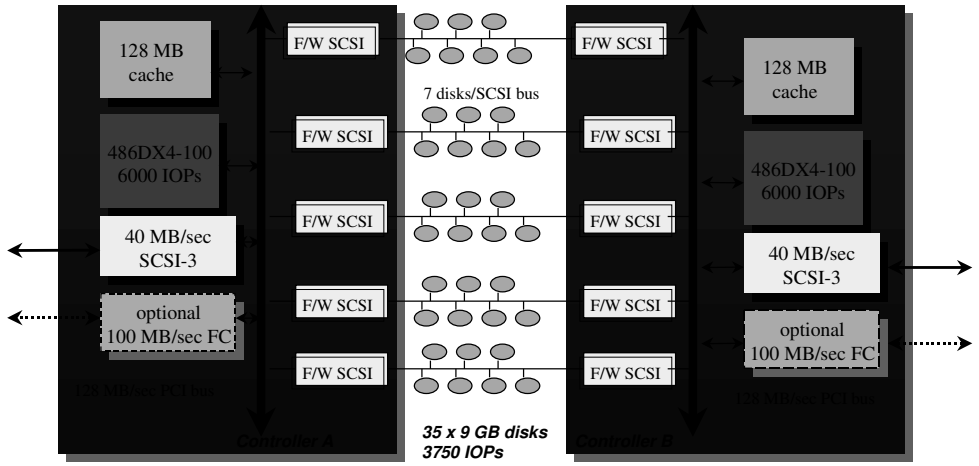
Application I/O Profiles

- Commercial data processing
  - typically indexed, random access
  - I/O size is small (2KB) because records are small
- Decision support - transaction set analysis
  - majority of I/O is sequential by user
  - I/O size is very large (64KB or more) for efficiency
    - might have to read an entire table (could be 1TB+)
  - Read mostly
- File Servers
  - file system I/O size is moderate (8K)
  - access is sequential by user but interleaved (so random) at drives

Common Implementations

- 4-128 drives are mounted in a box or rack
- 1-16 SCSI busses connect member disks
- member disks managed by a local controller
  - typically 100 MHz 486, 50 MHz i960, 120 MHz PPC 603e
  - controller also manages NVRAM cache
- controllers are replicated
- 2-16 SCSI or FibreChannel host connections

Typical 1997 Disk Array



36+ MB/sec sustained ***36+ MB/sec sustained***

Common Misconceptions

- "RAID boxes are inherently faster than regular disks"
  - nope: it's the RAID organization
  - in terms of latency, RAID boxes may even be slower than disk
- "RAID organizations check data on read"
  - nope: RAID-1/1+0 don't check on read
  - RAID-3/5 don't check on read unless a member reports error
  - member disks store ~20% ECC data, so reported errors are BIG