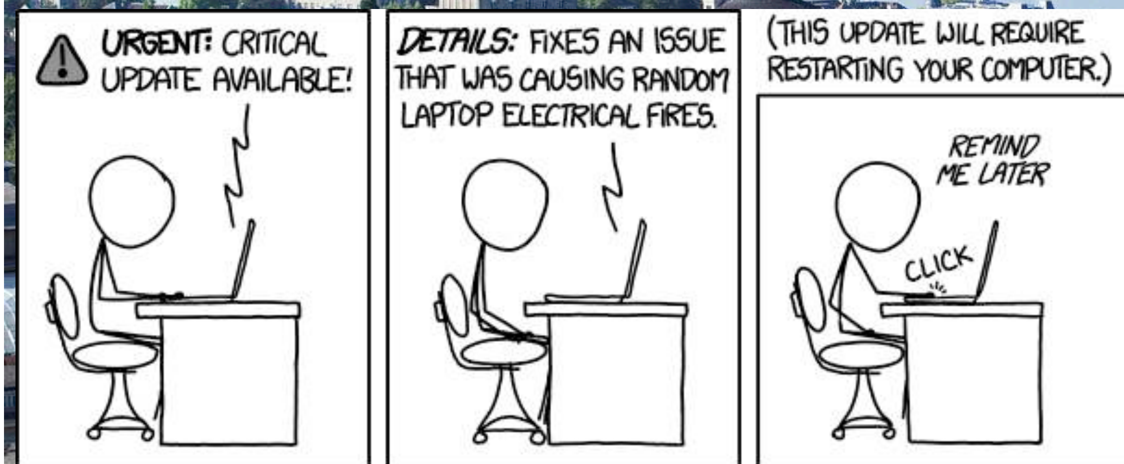
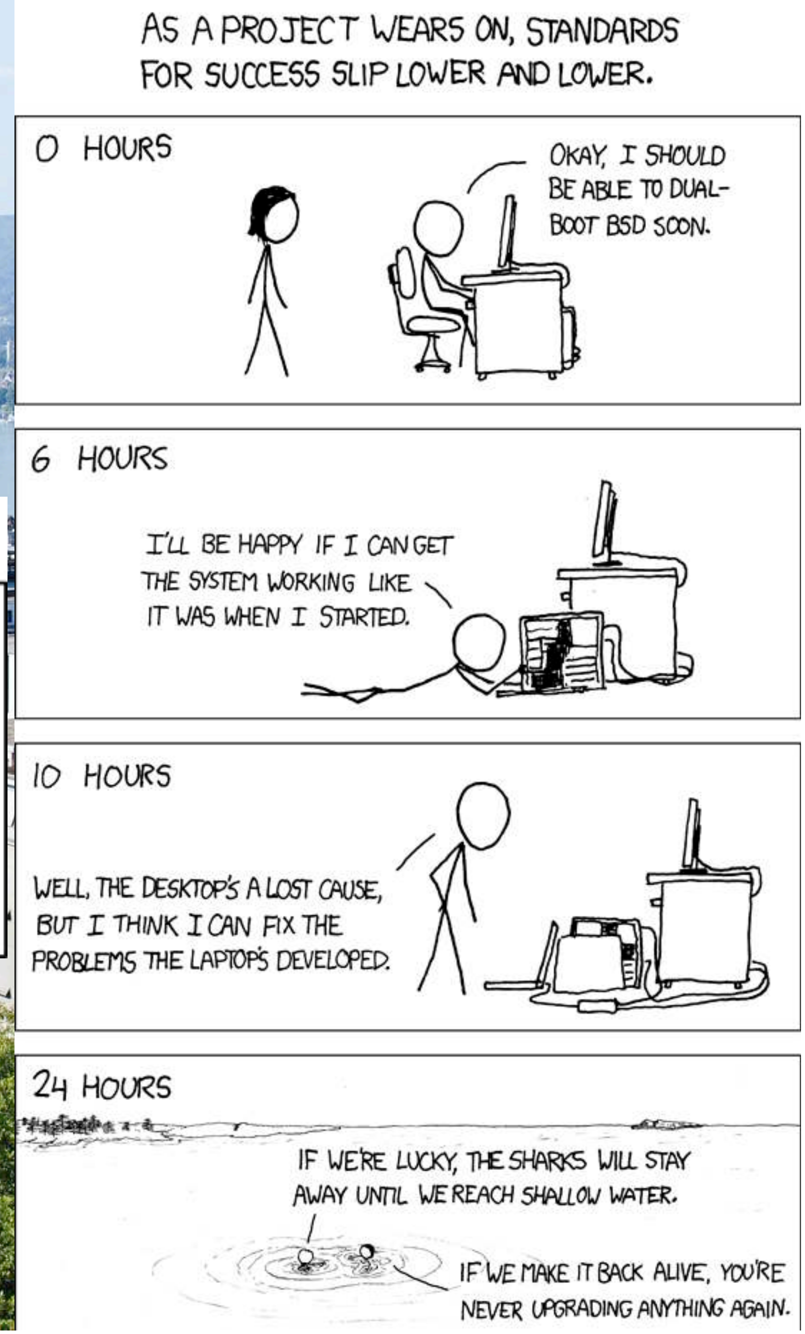
ADRIAN PERRIG & TORSTEN HOEFLER

Networks and Operating Systems Chapter 12: Reliable Storage, NUMA & The Future





Source: xkcd

Administrivia

- **Friday (tomorrow) is a holiday, exercises will be skipped**
  - Exercises this Thursday (today!) will also be skipped
  - Apologies for the late notice
- **The last OS exercises will be the week after Easter**
  - First week of Networking part
- **This is my last lecture this semester – Enjoy!!**

Basic exam tips

- **First of all, read the instructions**
- **Then, read the whole exam paper through**
- **Look at the number of points for each question**
  - This shows how long we think it will take to answer!
- **Find one you know you can answer, and answer it**
  - This will make you feel better early on.
- **Watch the clock!**
  - If you are taking too long on a question, consider dropping it and moving on to another one.
- **Always show your working**
- **You should be able to explain each summary slide**
  - Tip: form learning groups and present the slides to each other
  - Do **NOT** overly focus on the quiz questions!
  - Ask TAs if there are questions

Our Small Quiz

- **True or false (raise hand)**
  - Receiver side scaling randomizes on a per-packet basis
  - Virtual machines can be used to improve application performance
  - Virtual machines can be used to consolidate servers
  - A hypervisor implements functions similar to a normal OS
  - If a CPU is strictly virtualizable, then OS code execution causes nearly no overheads
  - x86 is not strictly virtualizable because some instructions fail when executed in ring 1
  - x86 can be virtualized by binary rewriting
  - A virtualized host operating system can set the hardware PTBR
  - Paravirtualization does not require changes to the guest OS
  - A page fault with shadow page tables is faster than nested page tables
  - A page fault with writeable page tables is faster than shadow page tables
  - Shadow page tables are safer than writable page tables
  - Shadow page tables require paravirtualization

Reliable Storage

OSPP Chapter 14

Reliability and Availabilty

A storage system is:

- ***Reliable*** **if it continues to store data and can read and write it.**
  ⇒ Reliability: probability it will be reliable for some period of time
- ***Available*** **if it responds to requests**
  ⇒ Availability: probability it is available at any given time

What goes wrong?

1. **Operating interruption: Crash, power failure**
   - Approach: use **transactions** to ensure data is consistent
   - Covered in the databases course
   - See book for additional material
2.

File system transactions

- **Not widely supported**
- **Only one atomic operation in POSIX:**
  - Rename
- **Careful design of file system data structures**
- **Recovery using fsck**
- **Superseded by transactions**
  - Internal to the file system
  - Exposed to applications

What goes wrong?

1. **Operating interruption: Crash, power failure**
   - Approach: use **transactions** to ensure data is consistent
   - Covered in the databases course
   - See book for additional material
2. **Loss of data: Media failure**
   - Approach: use **redundancy** to tolerate loss of media
   - E.g. RAID storage
   - Topic for today

Media failures 1: Sector and page failures

Disk keeps working, but a sector doesn't

- Sector writes don't work, reads are corrupted
- Page failure: the same for Flash memory

Approaches:

1. **Error correcting codes:**
   - Encode data with redundancy to recover from errors
   - Internally in the drive
2. **Remapping: identify bad sectors and avoid them**
   - Internally in the disk drive
   - Externally in the OS / file system

Caveats

- **Nonrecoverable error rates are significant**
  - And getting more so!
- **Nonrecoverable error rates are not constant**
  - Affected by age, workload, etc.
- **Failures are not independent**
  - Correlation in time and space
- **Error rates are not uniform**
  - Different models of disk have different behavior over time

A well-respected disk available now from pcp.ch

Seagate Barracuda 3TB,
7200rpm, 64MB, 3TB, SATA-3

Price this weekend: CHF 119.-
(last year CHF 105,-)
(in 2013 CHF 150,-)

Specifications (from manufacturer's website)

Seagate®

Persistent errors that are *not* masked by coding inside the drive

Specifications	3TB[1]	2TB[1]
Model Number	ST33000651AS	ST32000641AS
Interface Options	SATA 6Gb/s NCQ	SATA 6Gb/s NCQ
Performance		
Transfer Rate, Max Ext (MB/s)	600	600
Max Sustained Data Rate OD (MB/s)	149	138
Cache (MB)	64	64
Average Latency (ms)	4.16	4.16
Spindle Speed (RPM)	7200	7200
Configuration/Organization		
Heads/Disks	10/5	8/4
Bytes per Sector	512	512
Reliability/Data Integrity		
Load/Unload Cycles	300K	300K
Nonrecoverable Read Errors per Bits Read, Max	1 per 10E14	1 per 10E14
Annualized Failure Rate (AFR)	0.34%	0.34%
Mean Time Between Failures (hours)	750,000	750,000
Limited Warranty (years)	5	5
Power Management		
Startup Current +12 Peak (A, ±10%)	2.0	2.8

Unrecoverable read errors

- What's the chance we could read a *full* 3TB disk without errors?
- For each bit:

$$\Pr(success) = 1 - 10^{-14}$$

- Whole disk:

$$\Pr(success) = (1 - 10^{-14})^{8 \times 3 \times 10^{12}} \approx \mathbf{0.7868}$$

- Feeling lucky?

Lots of assumptions: Independent errors, etc.

Media failures 2: Device failure

- **Entire disk (or SSD) just stops working**
  - Note: always detected by the OS
  - Explicit failure ⇒ less redundancy required
- **Expressed as:**
  - Mean Time to Failure (MTTF)
    (expected time before disk fails)
  - Annual Failure Rate = 1/MTTF
    (fraction of disks failing in a year)

Specifications (from manufacturer's website)

Seagate®

Specifications	3TB[1]	2TB[1]
Model Number	ST33000651AS	ST32000641AS
Interface Options	SATA 6Gb/s NCQ	SATA 6Gb/s NCQ
Performance		
Transfer Rate, Max Ext (MB/s)	600	600
Max Sustained Data Rate OD (MB/s)	149	138
Cache (MB)	64	64
Average Latency (ms)	4.16	4.16
Spindle Speed (RPM)	7200	7200
Configuration/Organization		
Heads/Disks	10/5	8/4
Bytes per Sector	512	512
Reliability/Data Integrity		
Load/Unload Cycles	300K	300K
Nonrecoverable Read Errors per Bits Read, Max	1 per 10E14	1 per 10E14
Annualized Failure Rate (AFR)	0.34%	0.34%
Mean Time Between Failures (hours)	750,000	750,000
Limited Warranty (years)	5	5
Power Management		
Startup Current +12 Peak (A, ±10%)	2.0	2.8

Caveats

- **Advertised failure rates can be misleading**
  - Depend on conditions, tests, definitions of failure…
- **Failures are not uncorrelated**
  - Disks of similar age, close together in a rack, etc.
- **MTTF is not useful life!**
  - Annual failure rate only applies during design life!
- **Failure rates are not constant**
  - Devices fail very quickly or last a long time

And Reality?

Appears in the Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07), February 2007

Failure Trends in a Large Disk Drive Population

(S.M.A.R.T – Self-Monitoring, Analysis, and Reporting Technology)

Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz André Barroso
Google Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
{edpin,wolf,luiz}@google.com

Abstract

It is estimated that over 90% of all new information produced in the world is being stored on magnetic media, most of it on hard disk drives. Despite their importance, there is relatively little published work on the failure patterns of disk drives, and the key factors that affect their lifetime. Most available data are either based on extrapolation from accelerated aging experiments or from relatively modest sized field studies. Moreover, larger population studies rarely have the infrastructure in place to collect health signals from components in operation, which is critical information for detailed failure analysis.

We present data collected from detailed observations of a large disk drive population in a production Internet services deployment. The population observed is many times larger than that of previous studies. In addition to presenting failure statistics, we analyze the correlation between failures and several parameters generally believed to impact longevity.

for guiding the design of
vising deployment and m

Despite the importanc
few published studies on
drives. Most of the avai
the disk manufacturers tl
typically based on extra
test data of small popu
databases. Accelerated li
viding insight into how s
affect disk drive lifetime
predictors of actual failu
in the field [7]. Statistic
cally based on much larger populations, but since there is little or no visibility into the deployment characteristics, the analysis lacks valuable insight into what actually happened to the drive during operation. In addition,

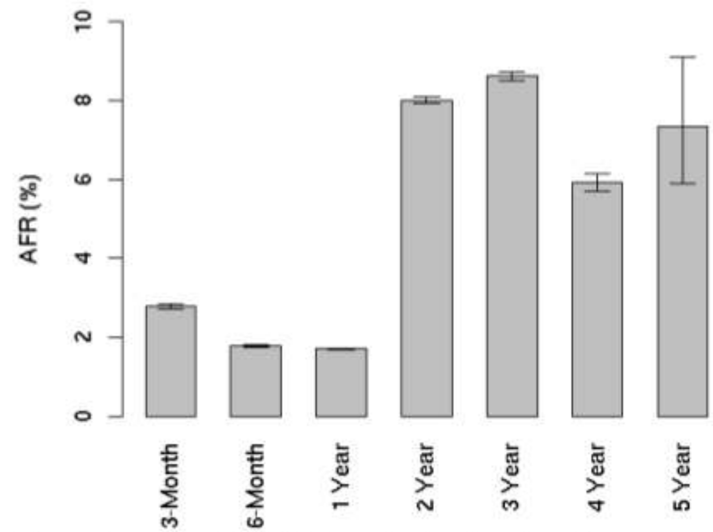
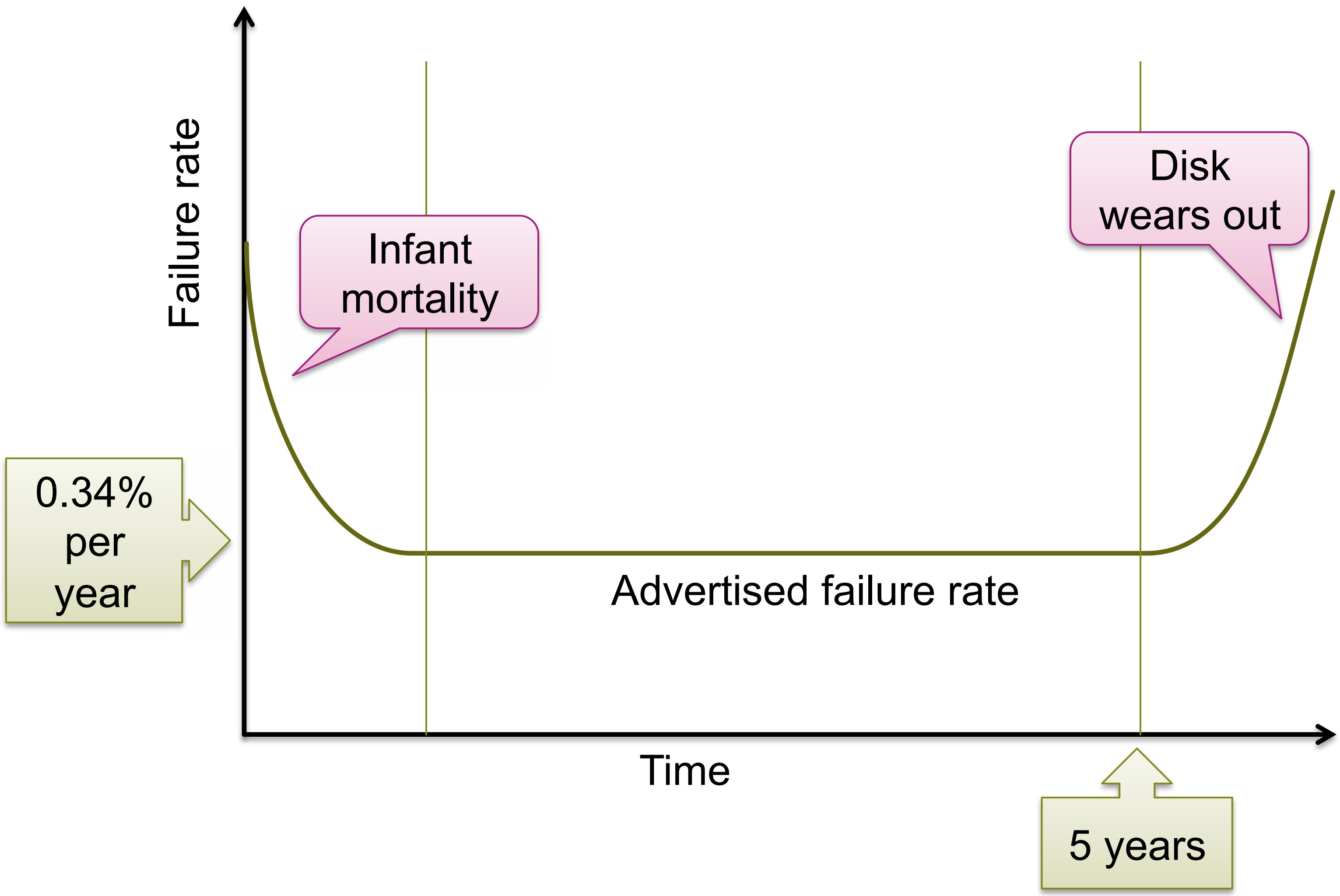
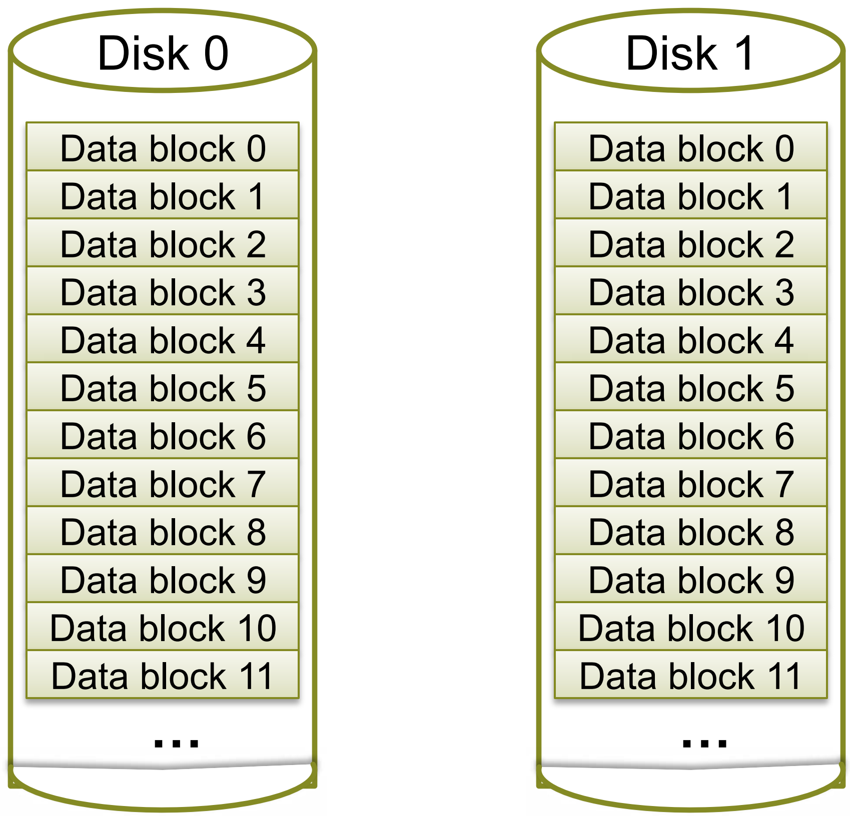


Figure 2: Annualized failure rates broken down by age groups

Bathtub curve



RAID 1: simple mirroring



Writes go to both disks

Reads from either disk (may be faster)

Sector or whole disk failure ⇒ data can still be recovered

Parity disks and striping

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
Block 0	Block 1	Block 2	Block 3	Parity(0-3)
Block 4	Block 5	Block 6	Block 7	Parity(4-7)
Block 8	Block 9	Block 10	Block 11	Parity(8-11)
Block 12	Block 13	Block 14	Block 15	Parity(12-15)
Block 16	Block 17	Block 18	Block 19	Parity(16-19)
Block 20	Block 21	Block 22	Block 23	Parity(20-23)
Block 24	Block 25	Block 26	Block 27	Parity(24-27)
Block 28	Block 29	Block 30	Block 31	Parity(28-31)
Block 32	Block 33	Block 34	Block 35	Parity(32-35)
Block 36	Block 37	Block 38	Block 39	Parity(36-39)
Block 40	Block 41	Block 42	Block 43	Parity(40-43)
Block 44	Block 45	Block 46	Block 47	Parity(44-47)
...	...	...	...	...

Parity disks

- Note: errors are always detected
  ⇒ Parity allows errors to be corrected

- Write *d'* to block ⇒ must also update parity, e.g.
  - Read *d* from block, parity block, then:

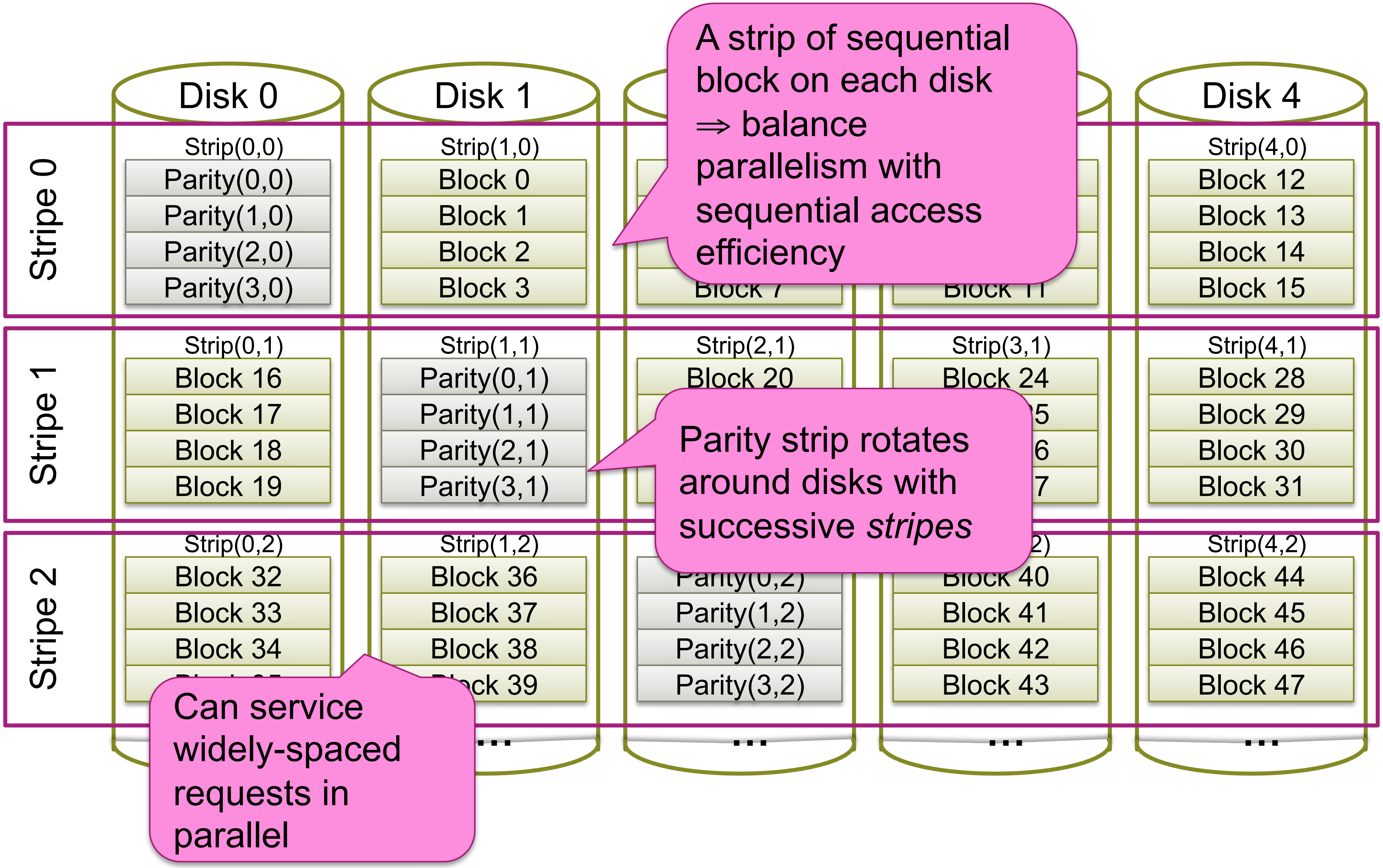
$$parity' = parity \oplus n' \oplus n$$

  - Write *d'* to block *n*, *parity'* to parity block

High overhead for small writes

- Problem: with 5 disks, parity disk is accessed 4 times as often on average!

RAID5: Rotating parity



Atomic update of data and parity

What if system crashes in the middle?

1. **Use non-volatile write buffer**
2. **Transactional update to blocks**
3. **Recovery scan**
   - And hope nothing goes wrong during the scan
4. **Do nothing (seriously)**

Recovery

- **Unrecoverable read error on a sector:**
  - Remap bad sector
  - Reconstruct contents from stripe and parity
- **Whole disk failure:**
  - Replace disk
  - Reconstruct data from the other disks
  - Hope nothing else goes wrong…

Mean time to repair (MTTR)

RAID-5 can lose data in three ways:

1. **Two full disk failures (second while the first is recovering)**
2. **Full disk failure and sector failure on another disk**
3. **Overlapping sector failures on two disks**

- **MTTR: Mean time to repair**
  - Expected time from disk failure to when new disk is fully rewritten, often hours
- **MTTDL: Mean time to data loss**
  - Expected time until 1, 2 or 3 happens

Analysis

See the book for *independent* failures

- **Key result: most likely scenario is #2.**

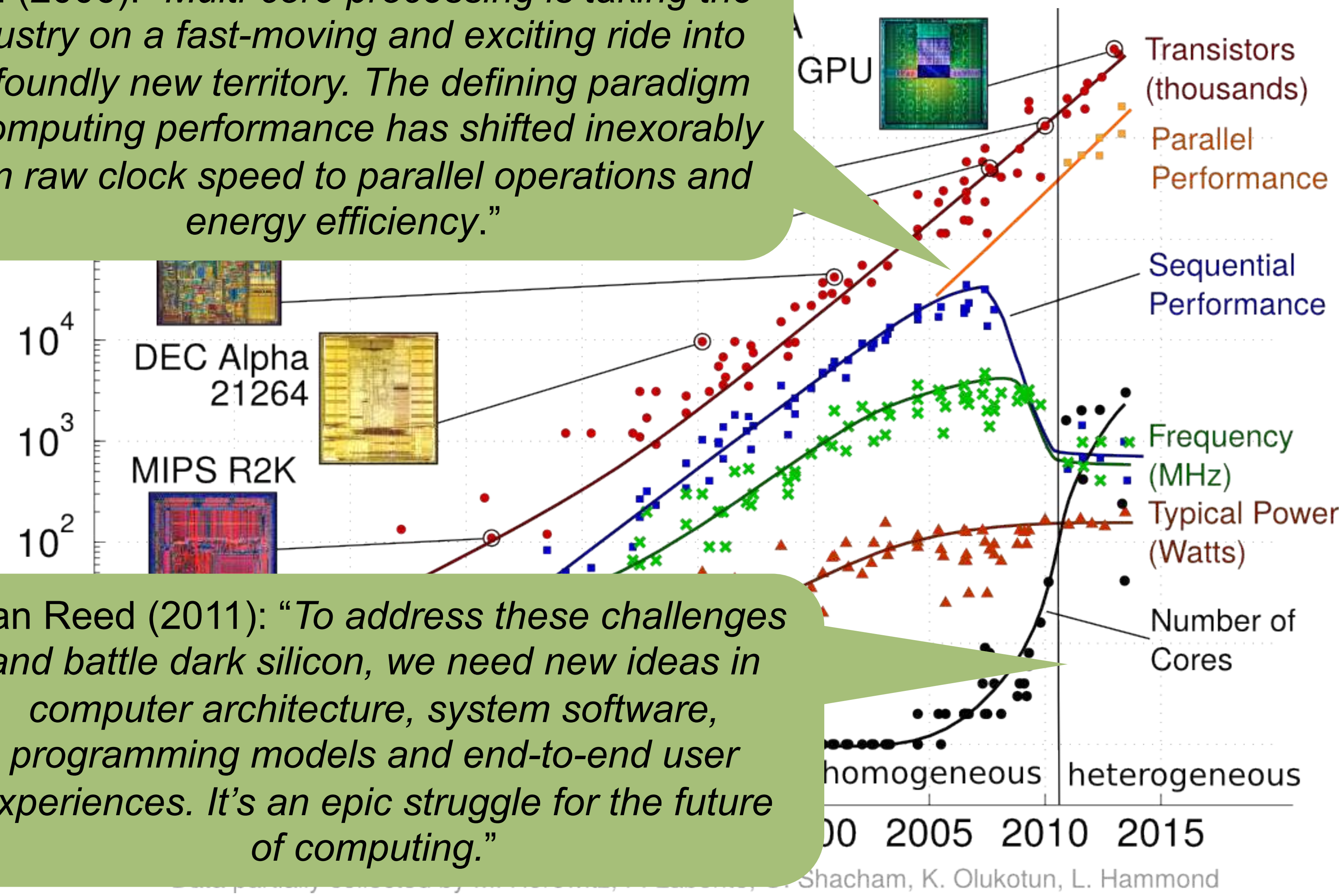
Solutions:

1. **More redundant disks, erasure coding**
2. **Scrubbing**
   - Regularly read the whole disk to catch UREs early
3. **Buy more expensive disks.**
   - I.e. disks with much lower error rates
4. **Hot spares**
   - Reduce time to plug/unplug disk

Hardware Trends

The future is exciting!

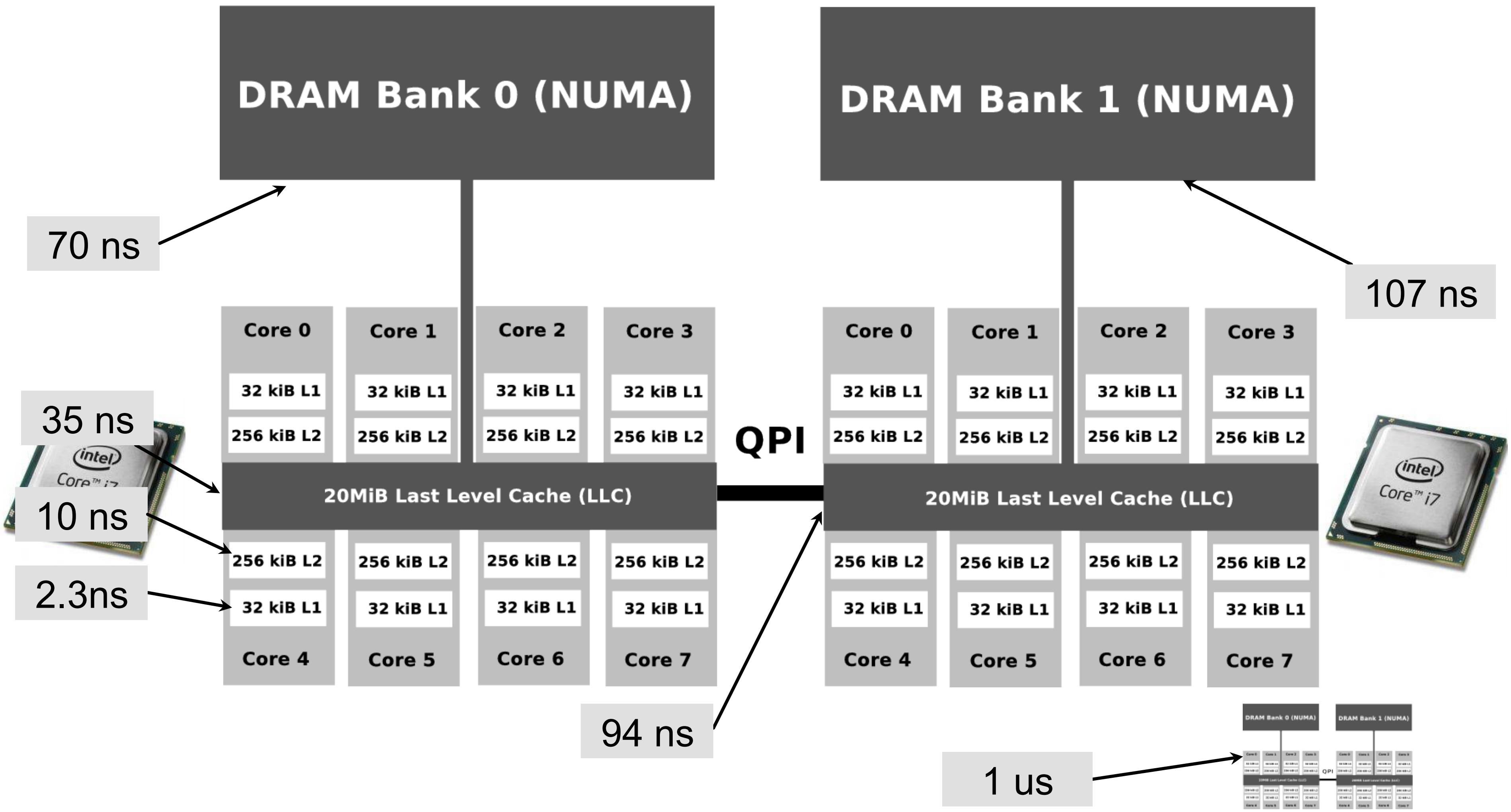
Intel (2006): "*Multi-core processing is taking the industry on a fast-moving and exciting ride into profoundly new territory. The defining paradigm in computing performance has shifted inexorably from raw clock speed to parallel operations and energy efficiency*."



Dan Reed (2011): "*To address these challenges and battle dark silicon, we need new ideas in computer architecture, system software, programming models and end-to-end user experiences. It's an epic struggle for the future of computing.*"

More and more cores …

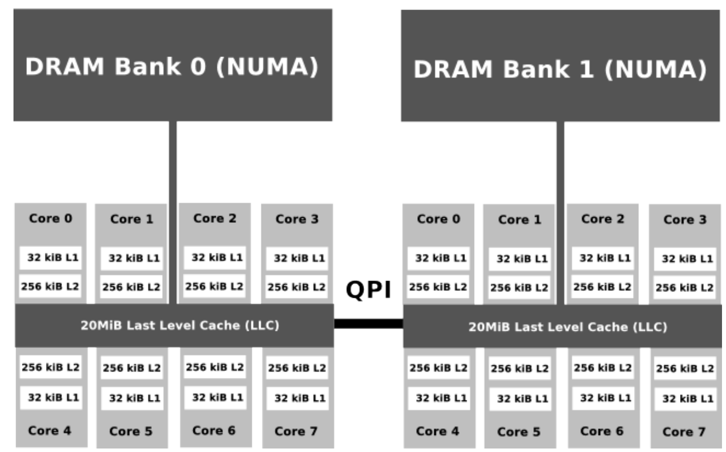
- **Like this dual-socket Sandy Bridge system:**



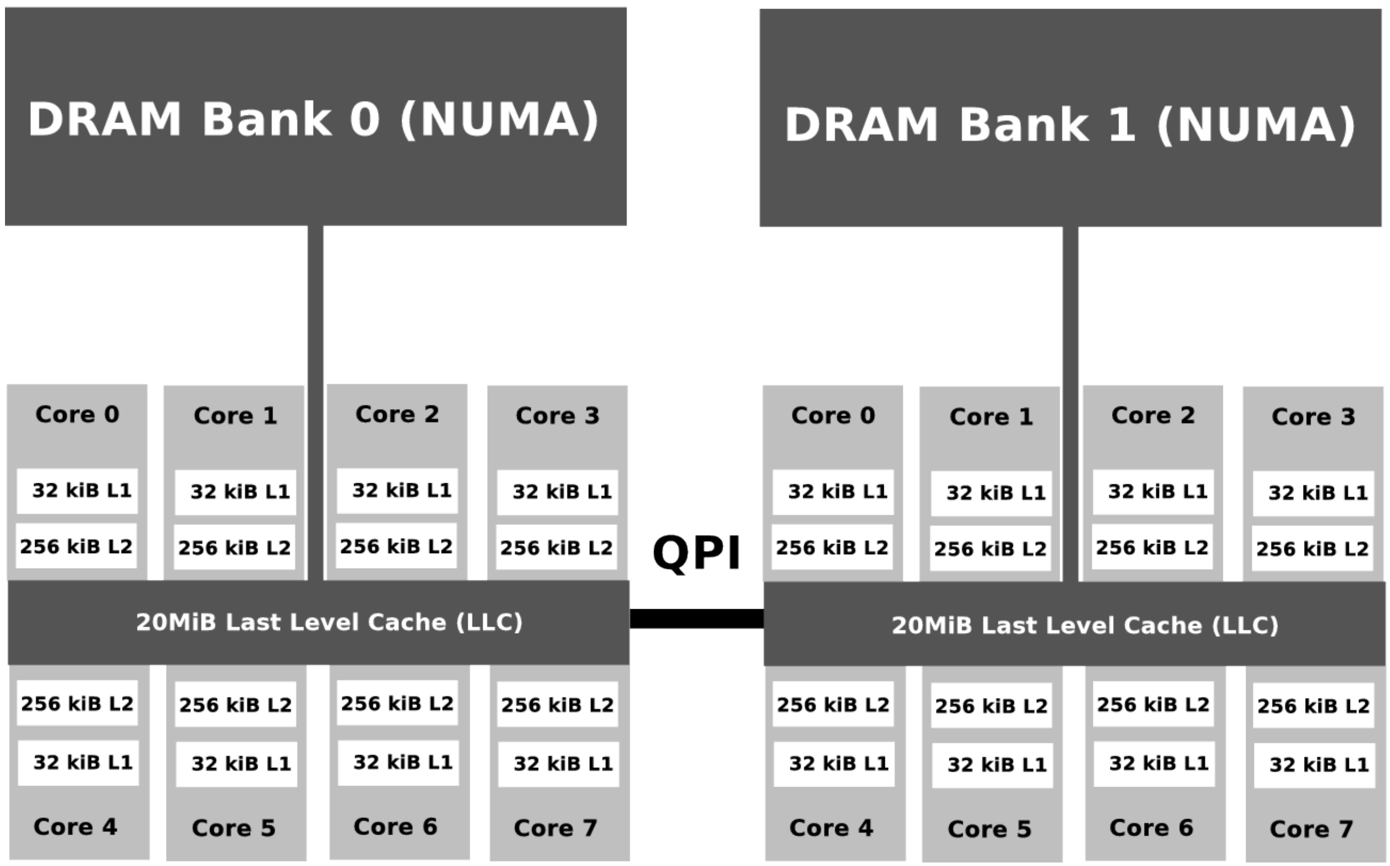
What does that mean, a nanosecond is short!!

- **How fast can you add two numbers?**
  - You’re smart, so let’s say 1s ☺

- **One core performs 8 floating point operations per cycle**
  - A cycle takes 0.45ns
- **Then ….**
  - A L1 cache access (2.3ns) takes 5s
  - A L2 cache access (10ns) takes 22s
  - A L3 cache access (35ns) takes 78s
  - A local DRAM access (70ns) takes 2.5 mins
  - A remote chip access (94ns) takes 3.5 mins
  - A remote DRAM access (107ns) takes 4 mins
  - A remote node memory access (1us) takes 37 mins



Non-Uniform Memory Access (NUMA)



NUMA in Operating Systems

- **Classify memory into NUMA nodes**
  - Affinity to processors and devices
  - Node-local accesses are fastest
- **Memory allocator and scheduler should cooperate!**
  - Schedule processes close to the NUMA node with their memory

- **State of the art:**
  - Ignore it (no semantic difference)
  - Striping in hardware (consecutive CLs come from different NUMA nodes)
    Homogeneous performance, no support in OS needed
  - Heuristics in NUMA-aware OS
  - Special NUMA control in OS
  - Application control

Heuristics in NUMA-aware OS

- **“First touch” allocation policy**
  - Allocate memory in the node where the process is running
  - Can create big problems for parallel applications (see DPHPC class)
- **NUMA-aware scheduling**
  - Prefer CPUs in NUMA nodes where a process has memory
- **Replicate “hot” OS data structures**
  - One copy per NUMA node
- **Some do page striping in software**
  - Allocate pages round robin
  - Unclear benefits

Special configurations

- **Administrator/command line configurations**
  - Special tools (e.g., Linux)
    taskset: set a process' CPU affinity
    numactl: set NUMA policies

- **Application configuration**
  - Syscalls to control NUMA (e.g., Linux)
    cpuset and friends, see "man 7 numa"

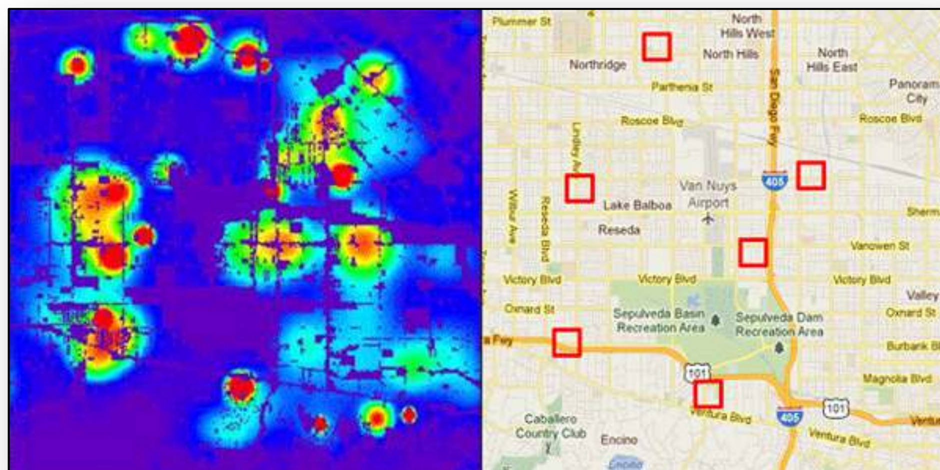
Non-local system times ☺

- **One core performs 8 floating point operations per cycle**
  - A cycle takes 0.45ns
- **Then ….**
  - A L1 cache access (2.3ns) takes 5s
  - A L2 cache access (10ns) takes 22s
  - A L3 cache access (35ns) takes 78s
  - A local DRAM access (70ns) takes 2.5 mins
  - A remote chip access (94ns) takes 3.5 mins
  - A remote DRAM access (107ns) takes 4 mins
  - A remote node memory access (1us) takes 37 mins
  - Solid state disk access (100us) takes 2.6 days
  - Magnetic disk access (5ms) takes 8.3 months
  - Internet Zurich to Chicago (150ms) takes 10.3 years
  - VMM OS reboot (4s) takes 277 years
  - Physical machine reboot (30s) 2 millennia

How to compute fast?

Why computing fast?

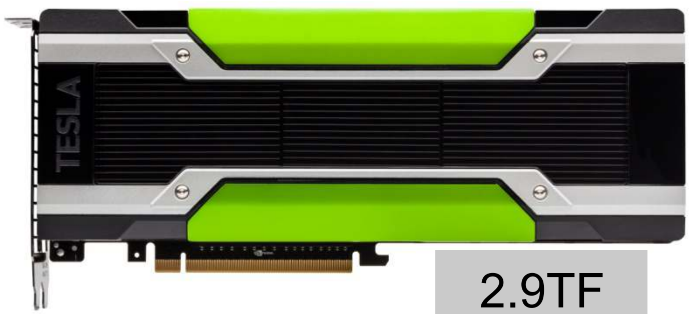
- **Computation is the third pillar of science**



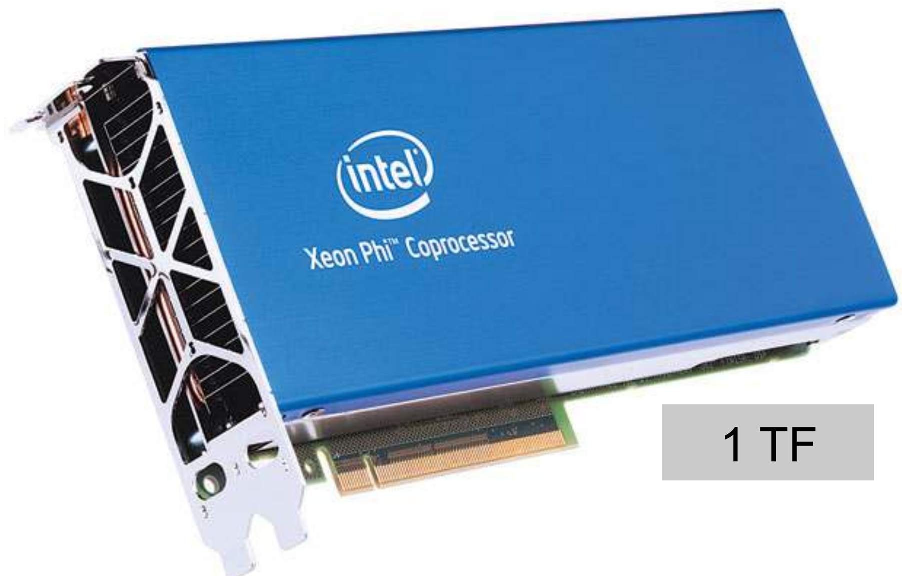
1 Teraflop in 1997

$67 Million

1 Teraflop 18 years later (2015)



2.9TF



1 TF

3 TF

"Amazon.com by Intel even has the co-processor selling for just $142 (plus $12 shipping) though they seem to be now out of stock until early December." (Nov. 11, 2014)

1 Teraflop 23 years later (2020)

1 Teraflop 33 years later (2030)

High-performance Computing (Supercomputing)

Datacenter Networking/RDMA

Heterogeneous Computing

Vectorization

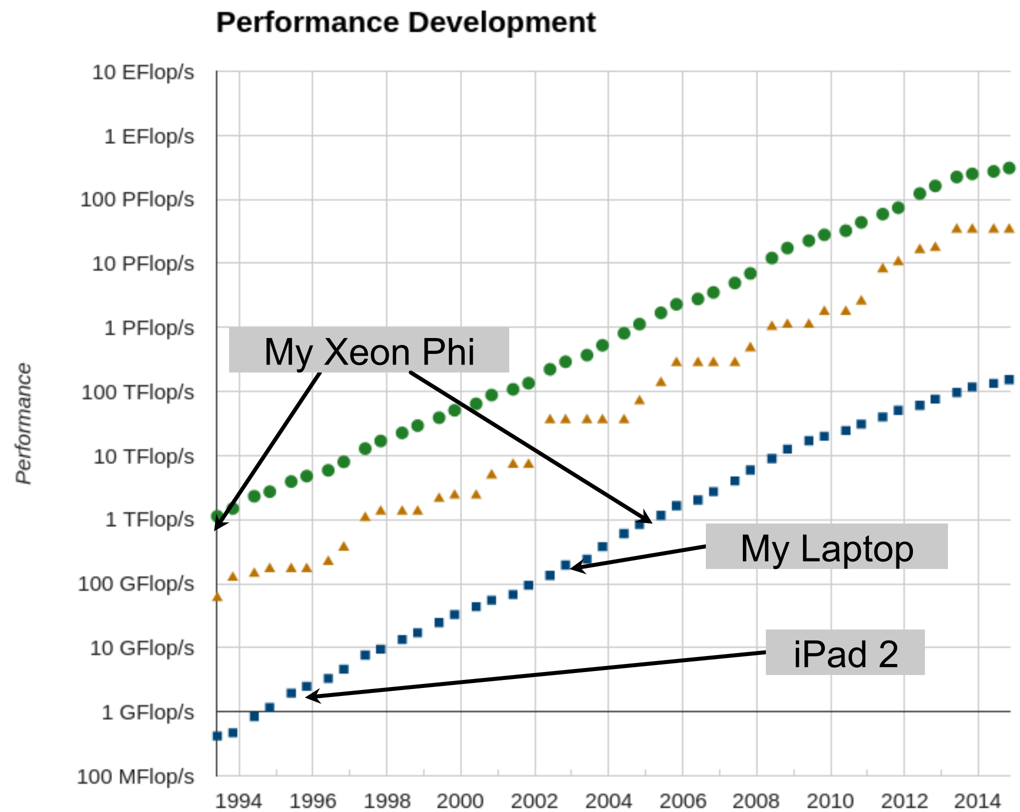
Multicore/SMP

IEEE Floating Point

….

Top 500

- **A benchmark, solve Ax=b**
  - As fast as possible! → as big as possible ☺
  - Reflects **some** applications, not all, not even many
  - Very good historic data!
- **Speed comparison for computing centers, states, countries, nations, continents ☹**
  - Politicized (sometimes good, sometimes bad)
  - Yet, fun to watch



The November 2014 List

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510

IDC, 2009: *"expects the HPC technical server market to grow at a healthy 7% to 8% yearly rate to reach revenues of $13.4 billion by 2015."*

"The non-HPC portion of the server market was actually down 20.5 per cent, to $34.6bn"

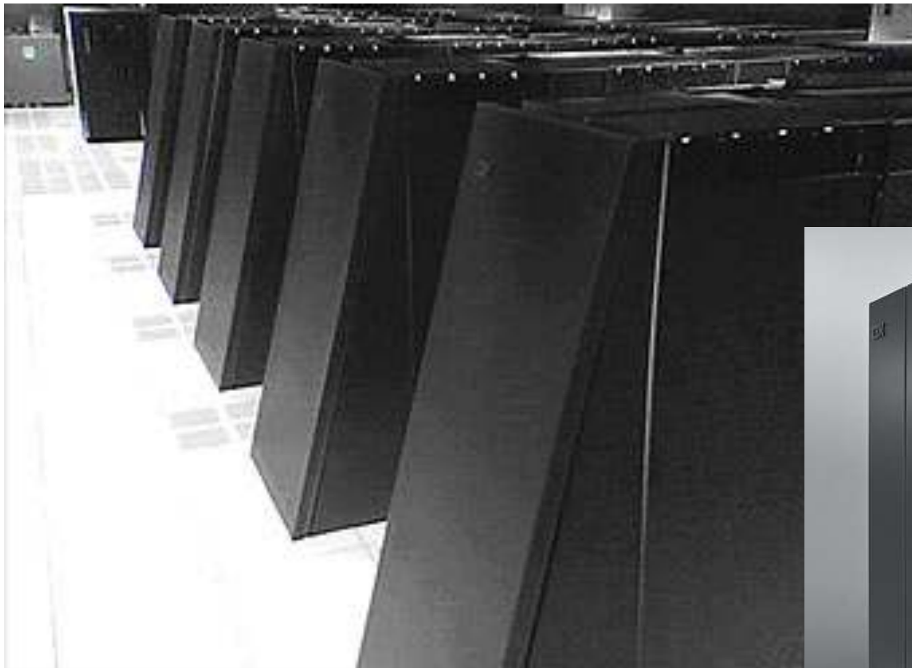
www.top500.org

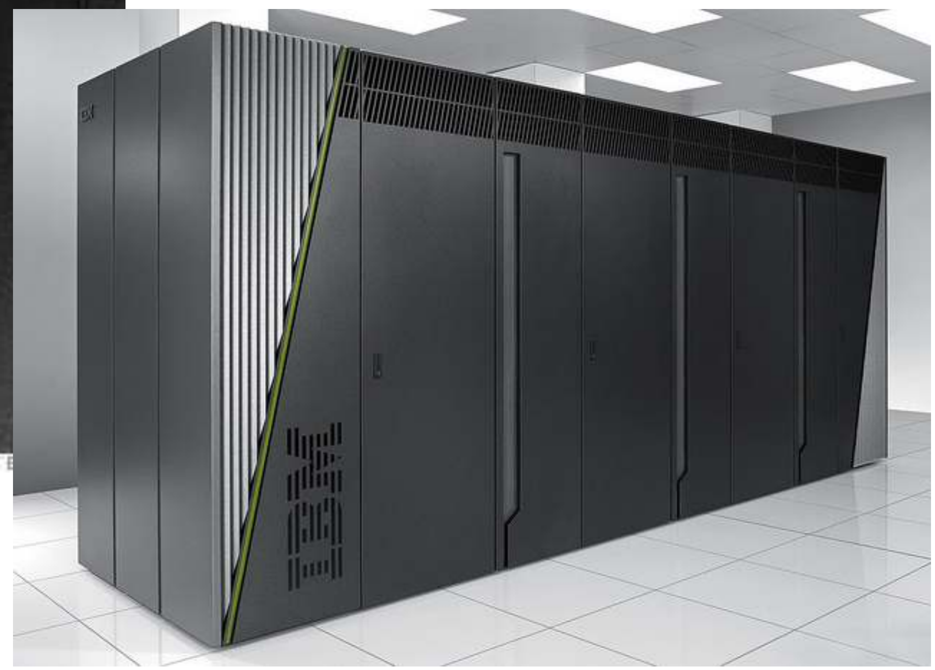
Case study: OS for High-Performance Computing

- **Remember the OS design goals?**
  - What if performance is #1?

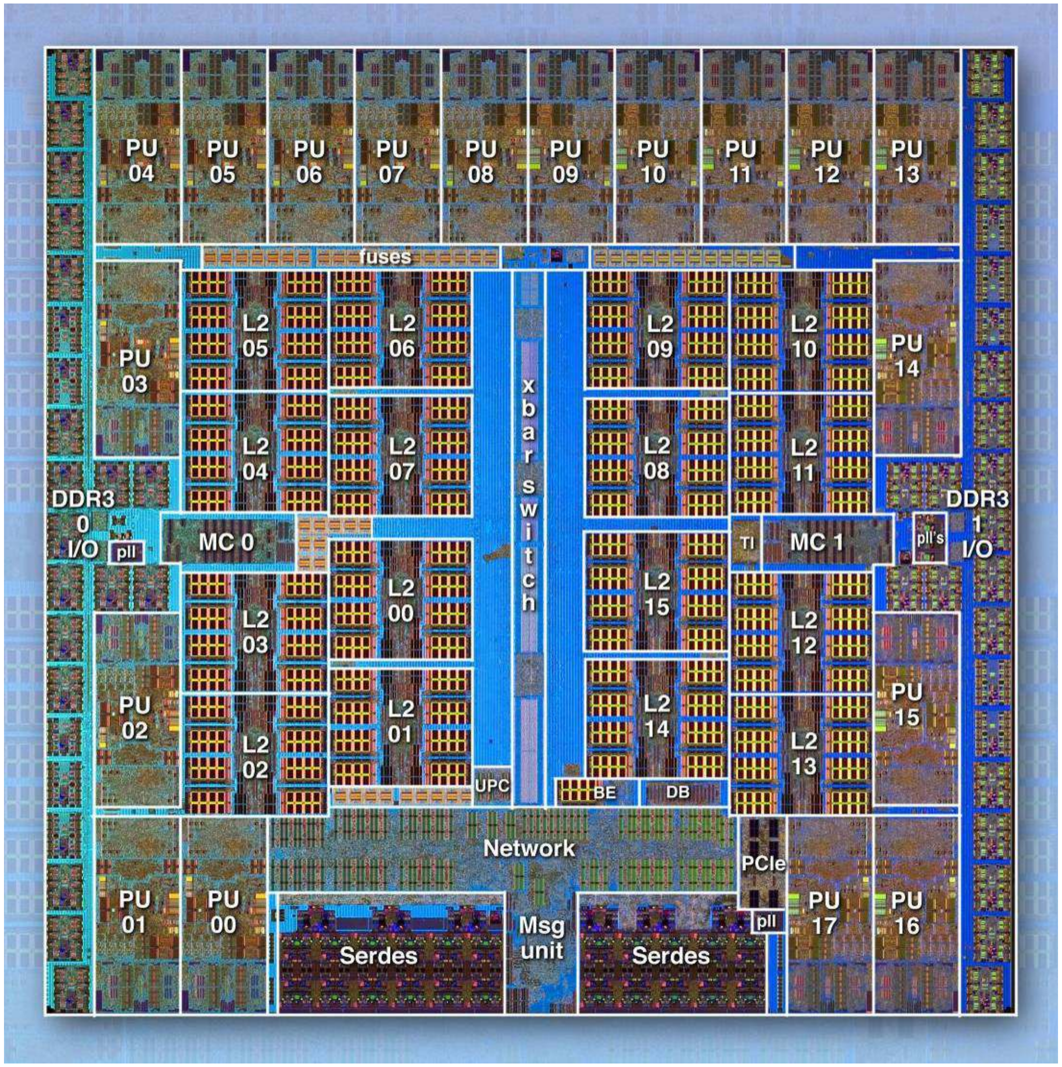
- **Different environment**
  - Clusters, special architectures, datacenters
  - Tens of thousands of nodes
  - Hundreds of thousands of cores
  - Millions of CHFs
  - Unlimited fun ☺

Case Study: IBM Blue Gene





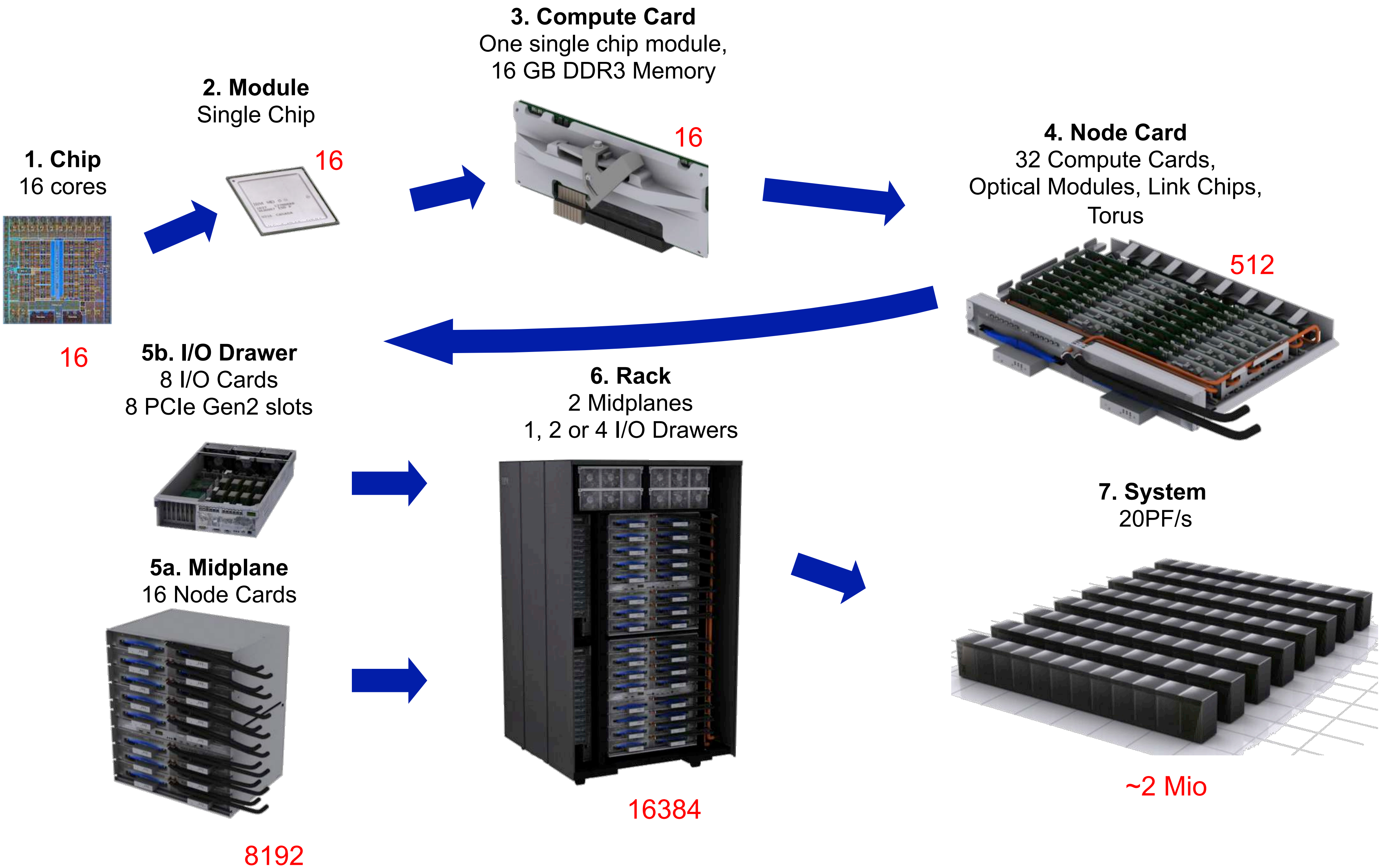
BlueGene/Q Compute chip



Ref: SC2010, IBM

- **360 mm² Cu-45 technology (SOI)**
  - ~ 1.47 B transistors
- **16 user + 1 service processors**
  - plus 1 redundant processor
  - all processors are symmetric
  - each 4-way multi-threaded
  - 64 bits PowerISA™
  - 1.6 GHz
  - L1 I/D cache = 16kB/16kB
  - L1 prefetch engines
  - each processor has Quad FPU (4-wide double precision, SIMD)
  - peak performance 204.8 GFLOPS@55W
- **Central shared L2 cache: 32 MB**
  - eDRAM
  - multiversioned cache will support transactional memory, speculative execution.
  - supports atomic ops
- **Dual memory controller**
  - 16 GB external DDR3 memory
  - 1.33 Gb/s
  - 2 * 16 byte-wide interface (+ECC)
- **Chip-to-chip networking**
  - Router logic integrated into BQC chip.

Blue Gene/Q packaging hierarchy

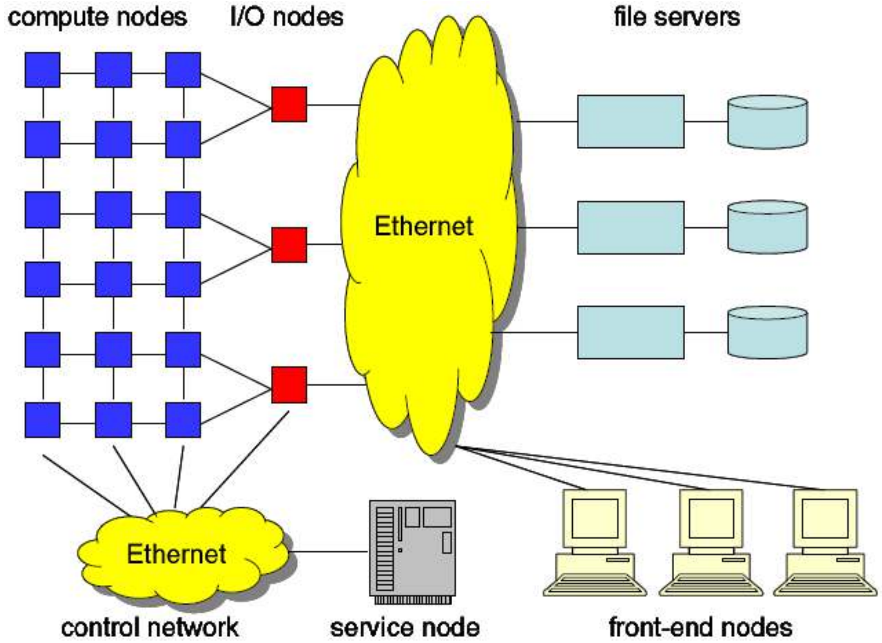


Ref: SC2010, IBM

Blue Gene/L System Organization

Heterogeneous nodes:

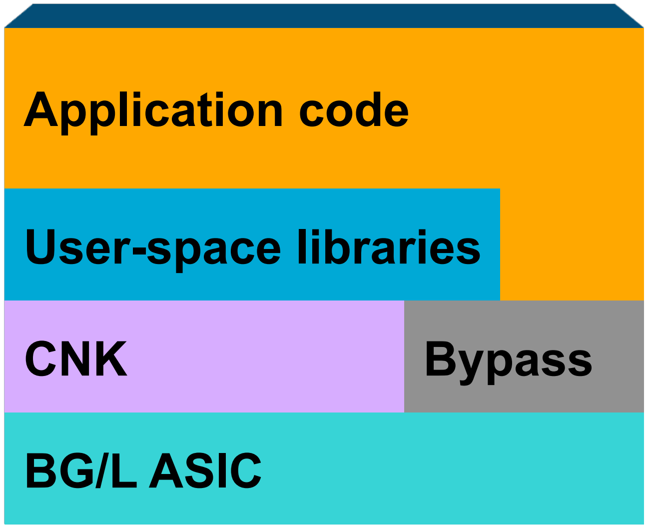
- **Compute (BG/L specific)**
  - Run specialized OS supporting computations efficiently
- **I/O (BG/L specific)**
  - Use OS flexibly supporting various forms of I/O
- **Service (generic)**
  - Uses conventional off-the-shelf OS
  - Provides support for the execution of compute and I/O node operating systems
- **Front-end (generic)**
  - Support program compilation, submission and debugging
- **File server (generic)**
  - Store data that the I/O nodes read and write



Source: Jose Moreira et al. "Designing Highly-Scalable Operating System: The Blue Gene/L Story", http://sc06.supercomputing.org/schedule/pdf/pap178.pdf

Software Stack in Compute Node

- CNK controls all access to hardware, and enables bypass for application use
- User-space libraries and applications can directly access torus and tree through bypass
- As a policy, user-space code should not directly touch hardware, but there is no enforcement of that policy



Source: http://www.research.ibm.com/bluegene/presentations/BGWS_05_SystemSoftware.ppt

Compute Node Kernel (CNK)

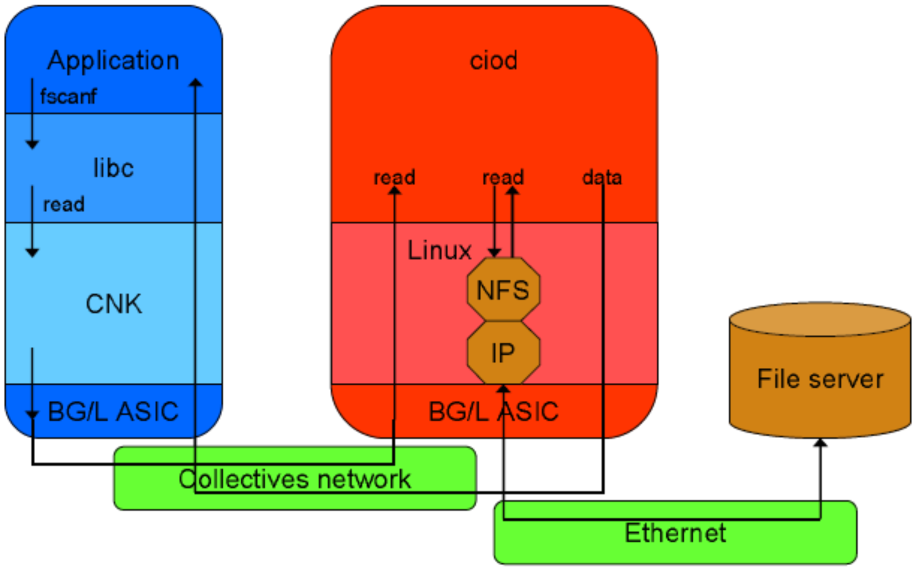
- **Lean Linux-like kernel (fits in 1MB of memory)**
  - stay out of way and let the application run
- **Performs job startup sequence on every node of a partition**
  - Creates address space for execution of compute process(es)
  - Loads code and initialized data for the executable
  - Transfers processor control to the loaded executable
- **Memory management**
  - Address spaces are flat and fixed (no paging), and fit statically into PowerPC 440 TLBs
- **No process scheduling: only one thread per processor**
- **Processor control stays within the application, unless:**
  - The application issues a system call
  - Timer interrupt is received (requested by the application code)
  - An abnormal event is detected, requiring kernel’s attention

CNK System Calls

- **Compute Node Kernel supports**
  - 68 Linux system calls (file I/O, directory operations, signals, process information, time, sockets)
  - 18 CNK-specific calls (cache manipulation, SRAM and DRAM management, machine and job information, special-purpose register access)
- **System call scenarios**
  - **Simple** calls requiring little OS functionality (e.g. accessing timing register) are handled locally
  - **I/O** calls using file system infrastructure or IP stack are shipped for execution in the I/O node associated with the issuing compute node
  - **Unsupported** calls requiring infrastructure not supported in BG/L (e.g. *fork()* or *mmap()*) return immediately with error condition

Function Shipping from CNK to CIOD

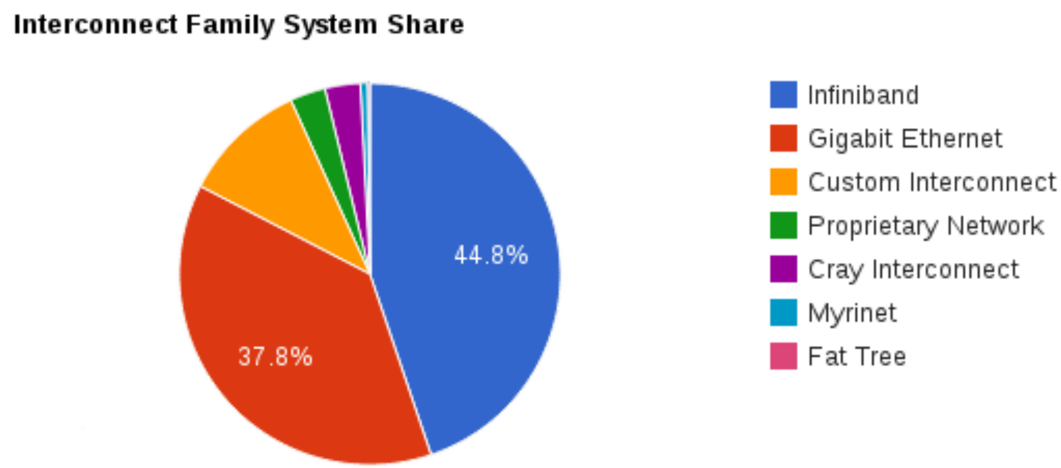
- **CIOD processes requests from**
  - Control system using socket to the service node
  - Debug server using a pipe to a local process
  - Compute nodes using the tree network
- **I/O system call sequence:**
  - CNK trap
  - Call parameters are packaged and sent to CIOD in the corresponding I/O node
  - CIOD unpacks the message and reissues it to Linux kernel on I/O node
  - After call completes, the results are sent back to the requesting CNK (and the application)

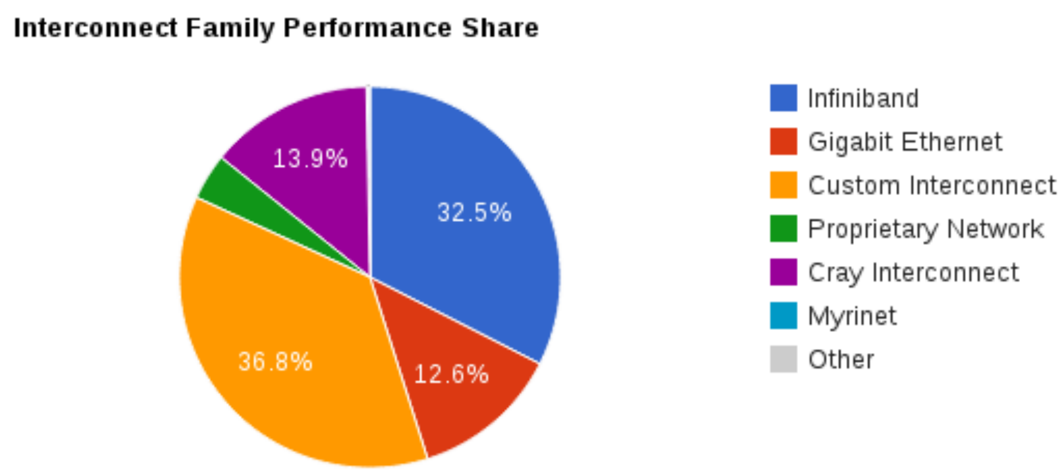


Source: IBM

How to communicate?

- **Communication is key in problem solving ☺**
  - Not just relationships!
  - Also scientific computations





Source: top500.org

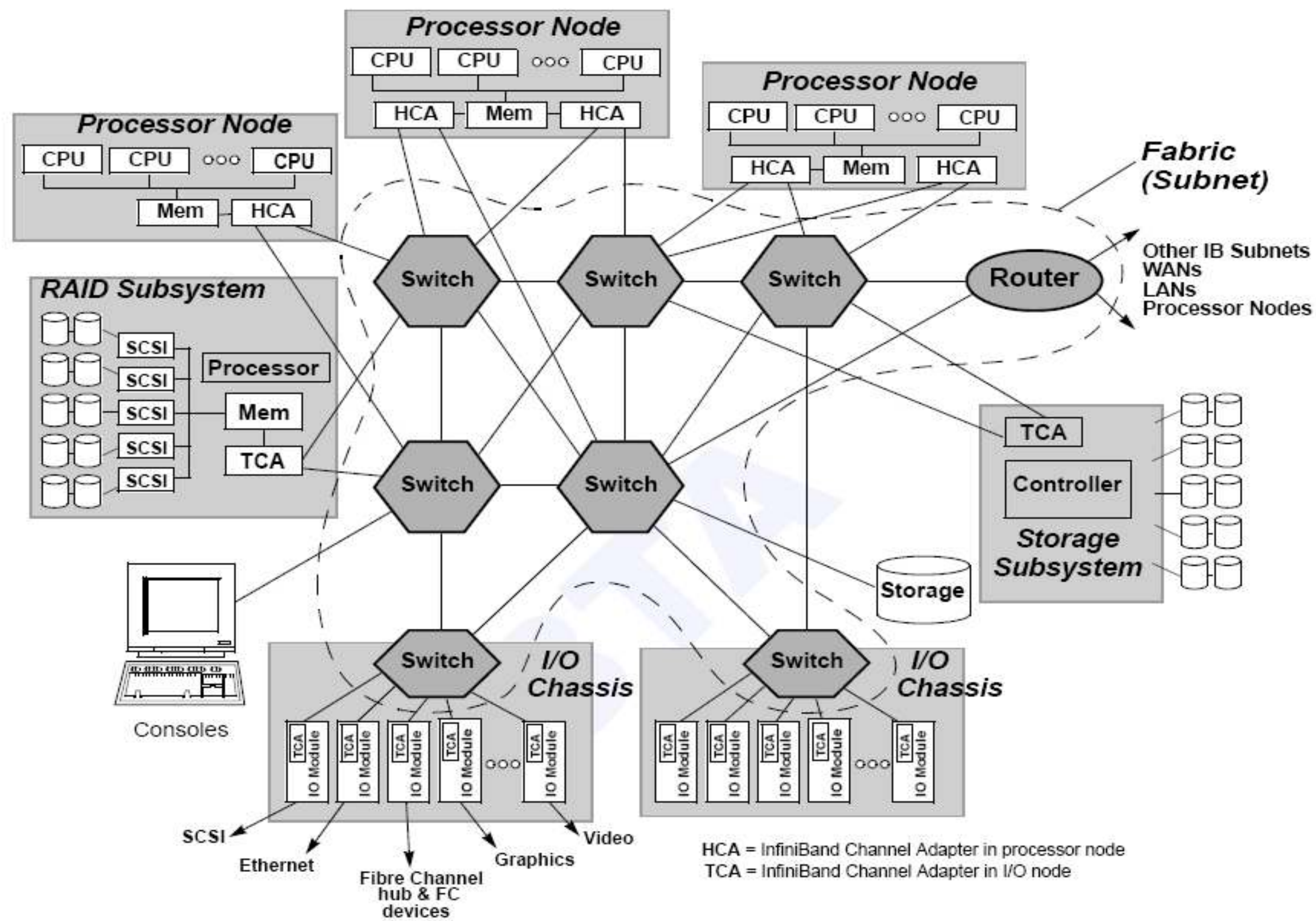
Remote Direct Memory Access

- **Remember that guy?**
  - EDR
  - 2x2x100 Gb/s → ~50 GB/s
  - Memory bandwidth: ~80 GB/s
  - 0.8 copies ☹
- **Solution:**
  - RDMA, similar to DMA
  - OS too expensive, bypass
  - Communication offloading

InfiniBand Overview

- **Components:**
  - Links/Channel adaptors
  - Switches/Routers
- **Routing is supported but rarely used, most IB networks are "LANs"**
- **Supports arbitrary topologies**
  - "Typical" topologies: fat tree, torus, islands
- **Link speed (all 4x):**
  - Single data rate (SDR): 10 Gb/s
  - Double data rate (DDR): 20 Gb/s
  - Quad data rate (QDR): 40 Gb/s
  - Fourteen data rate (FDR): 56 Gb/s
  - Enhanced data rate (EDR): 102 Gb/s

InfiniBand Network Structure



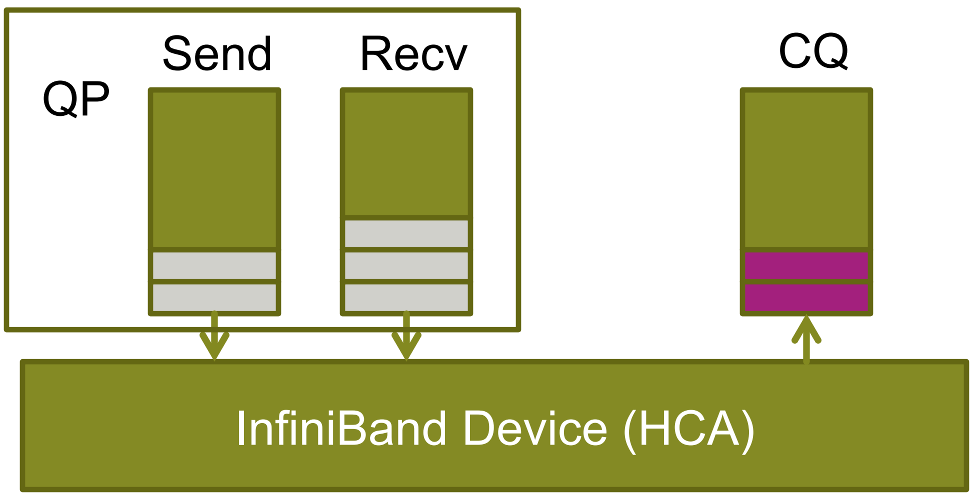
Source: IBA Spec

InfiniBand Subnet Routing

- ***No spanning tree protocol, allows parallel links&loops, initialization phases:***
  - *Topology discovery:* discovery MADs
  - *Path computation*: MinHop, …, DFSSSP
  - *Path distribution phase*: Configure routing tables
- **Problem: how to generate paths?**
  - MinHop == OSPF
  - Potentially bad bandwidth allocation!

Interaction with IB HCAs

- **Systems calls only for setup:**
  - Establish connection, register memory
- **Communication (send/recv, put, get, atomics) all in user-level!**
  - Through “verbs” interface



Open Fabrics Stack

- **OFED offers a unified programming interface**
  - Cf. Sockets
  - Originated in IB verbs
  - Direct interaction with device
  - Direct memory exposure

    Requires page pinning (avoid OS interference)
- **Device offers**
  - User-level driver interface
  - Memory-mapped registers

iWARP and RoCE

- **iWARP: RDMA over TCP/IP**
  - Ups:
    Routable with existing infrastructure
    Easily portable (filtering, etc.)
  - Downs:
    Higher latency (complex TOE)
    Higher complexity in NIC
    TCP/IP is not designed for datacenter networks
- **RoCE: RDMA over Converged Ethernet**
  - Data-center Ethernet!

Student Cluster Competition

- **5 undergrads, 1 advisor, 1 cluster, 2x13 amps**
  - 8 teams, 4 continents @SC
  - 48 hours, five applications, non-stop!
  - top-class conference (>11000 attendees)
- **Lots of fun**
  - Even more experience!
- **A Swiss team 2017?**
  - Search for "Student Cluster Challenge"
  - HPC-CH/CSCS may help

What to remember in 10 years!

The Lecture's Elevator Pitch

- **Roles:**
  - Referee, Illusionist, Glue
- **Example: processes, threads, and scheduling**
  - R: Scheduling algorithms (batch, interactive, realtime)
  - I: Resource abstractions (memory, CPU)
  - G: Syscalls, services, driver interface
- **Slicing along another dimension:**
  - Abstractions
  - Mechanisms

The Lecture's Elevator Pitch

- **IPC and other communications**
  - A: Sockets, channels, read/write
  - M: Network devices, packets, protocols
- **Memory Protection**
  - A: Access control
  - M: Paging, protection rings, MMU
- **Paging/Segmentation**
  - A: Infinite memory, performance
  - M: Caching, TLB, replacement algorithms, tables

The Lecture's Elevator Pitch

- **Naming**
  - A: (hierarchical) name spaces
  - M: DNS, name lookup, directories
- **File System**
  - A: Files, directories, links
  - M: Block allocation, inodes, tables
- **I/O**
  - A: Device services (music, pictures ☺)
  - M: Registers, PIO, interrupts, DMA

The Lecture's Elevator Pitch

- **Reliability:**
  - A: reliable hardware (storage)
  - M: Checksums, transactions, raid 0/5
- **And everything can be virtualized!**
  - CPU, MMU, memory, devices, network
  - A: virtualized x86 CPU
  - M: paravirtualization, rewriting, hardware extensions
  - A: virtualized memory protection/management
  - M: writable pages, shadow pages, hw support, IOMMU

The Lecture's ~~Elevator~~ Escalator Pitch

- **Ok, fine, it was an escalator pitch … in Moscow**
- **Please remember all for at least 10 years!**
  - Systems principles
  - … and how to make them fast ☺

Finito – Happy Easter!!

- **Thanks for being such fun to teach ☺**
  - Comments (also anonymous) are always appreciated!
- **If you are interested in parallel computing research, talk to me!**
  - Large-scale (datacenter) systems
  - Parallel computing (SMP and MPI)
  - GPUs (CUDA), FPGAs, Manycore …
  - … on twitter: @spcl_eth ☺
  - Hope to see you again! *Maybe in Design of Parallel and High-Performance Computing next semester ☺*
  - Or theses: *http://spcl.inf.ethz.ch/SeMa/*