Balancing System Resource Supply and Demand for Effective Computing

Xiaodong Zhang

Ohio State University

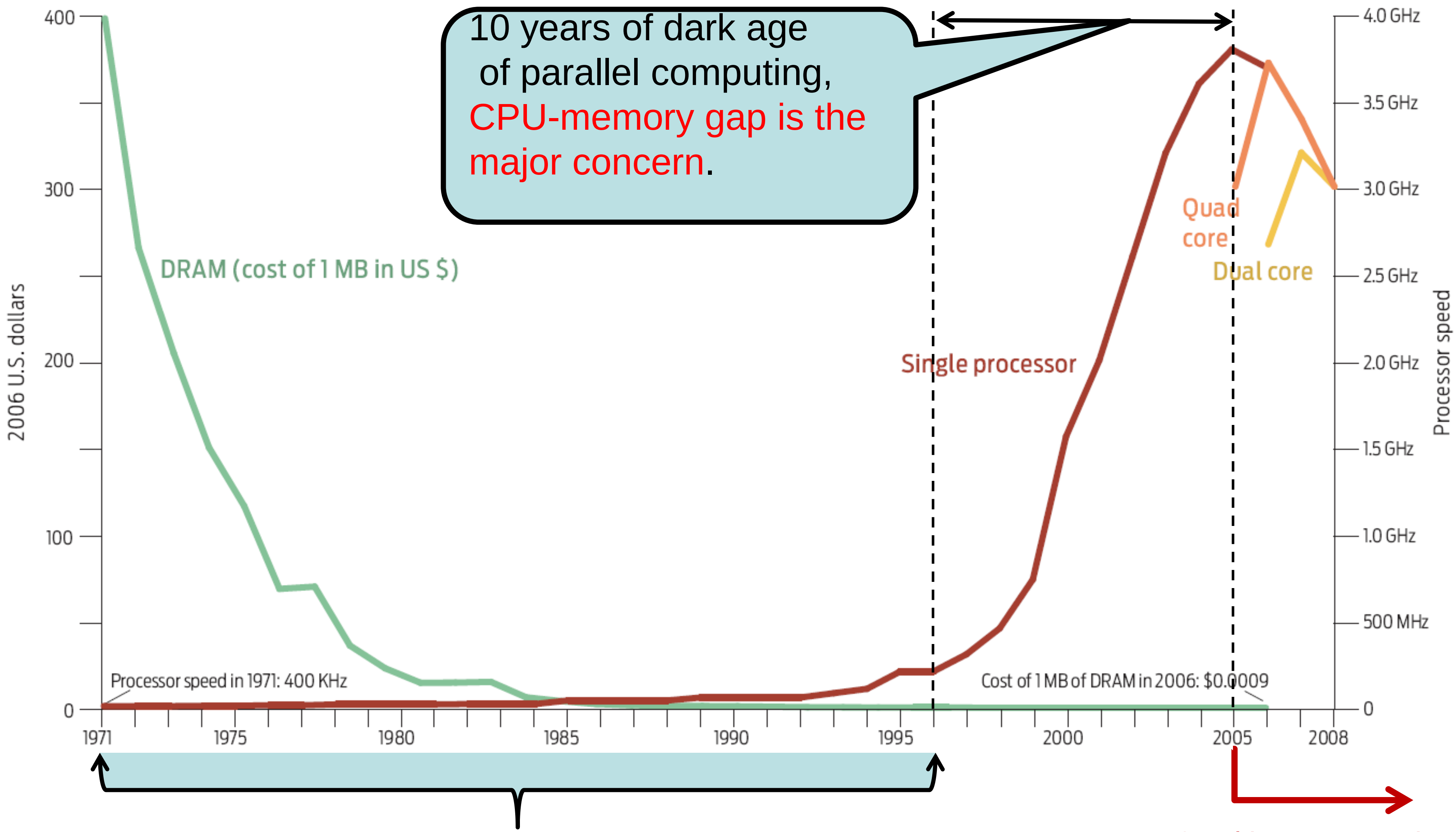
Computing is Pervasive and Powerful

- Computing resources become cheap and prolific.
  - Increasingly low cost for fast CPUs and large memory.
  - Cluster and Internet connect computing nodes easily.
- Three types of major computing resources:
  - High end systems, e.g. Blue Gene/L, Earth Simulator.
    - Ultra high performance but expensive. (customer designed nodes/networks)
  - Cluster systems, most Top-500's
    - Low cost, but low sustained performance. (commodity node/net)
    - Google has been a successfully scalable example.
  - Global systems, e.g., TeraGrid, utility and cloud computing
    - Utilizing global computing resources, but high Internet cost/overhead
- Clients are pervasive in everywhere in the globe
  - Desktops, laptops, PDAs, etc. connect to the Internet or via wireless

Major Resources in Computing and Network Systems

- **Good News in supply**
  - CPU cycles: oversupplied for many applications.
  - Memory bandwidth: improved dramatically.
  - Memory capacity: increasingly large and low cost.
  - I/O bandwidth: improved dramatically.
  - Disk capacity: huge and cheap.
  - Cluster and Internet bandwidths: very rich.
- **Bad News in demand**
  - CPU cycles per Watt decreases. (less energy efficient).
  - Cache capacity: always limited.
  - Improvement of data access latencies very slow.
  - Networking and energy costs are increasingly high
- **Adam Smith:** commodity price is defined by an “invisible hand” in the market. We need to balance
  - Oversupplied cycles, large storage capacity, fast networks
  - High demand of low latency accesses, low energy cost

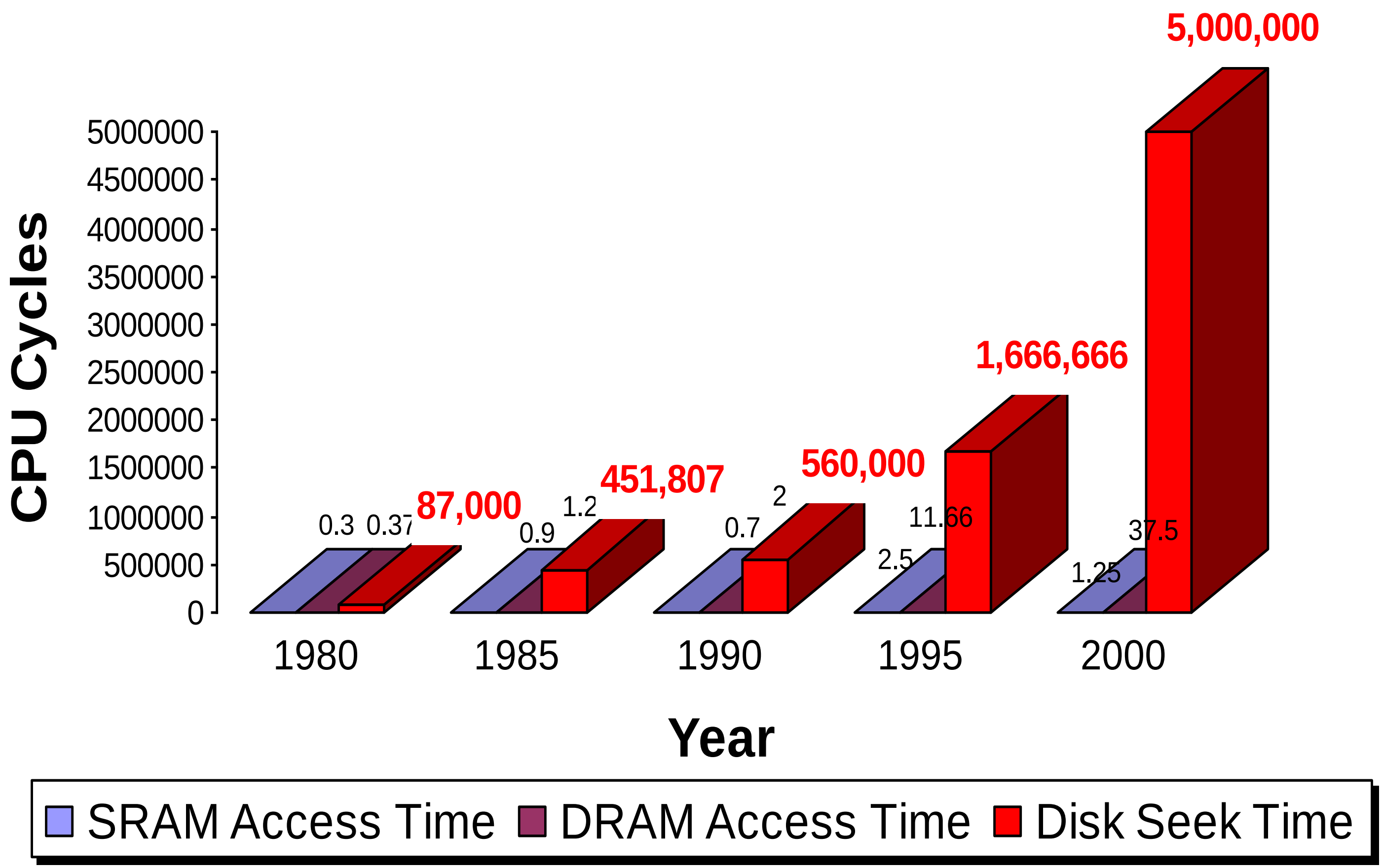
Moore's Law Driven Computing Research (IEEE Spectrum, May 2008)



25 year of golden age of parallel computing

New era of multicore computing
Memory problem continues

The disks in 2000 are 57 times "SLOWER" than their ancestors in 1980 --- increasingly widen the Speed Gap between Peta-Scale computing and Peta-Byte acesses.



Bryant and O'Hallaron, "Computer Systems: A Programmer's Perspective", Prentice Hall, 2003

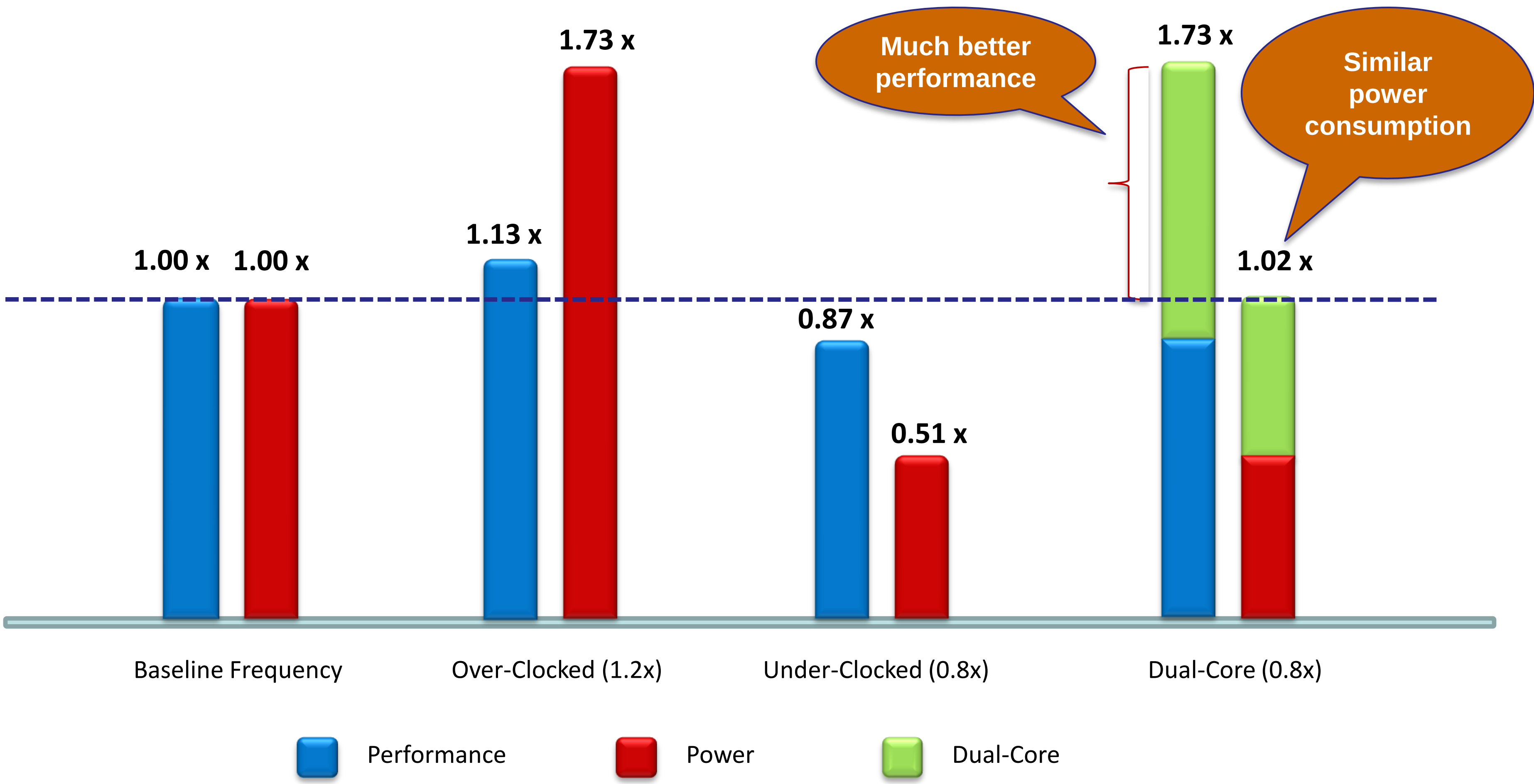
Opportunities of Technology Advancements

- Single-core CPU reached its peak performance
  - 1971 (2300 transistors on Intel 4004 chip): 0.4 MHz
  - 2005 (1 billion + transistors on Intel Pentium D): 3.75 GHz
  - After 10,000 times improvement, GHz stopped and dropped
  - CPU improvement will be reflected by number of cores in a chip
- Increased DRAM capacity enables large working sets
  - 1971 ($400/MB) to 2006 (0.09 cent/MB): 444,444 times lower
  - Buffer cache is increasingly important to break “disk wall”
- SSDs (flash memory) can further break the “wall”
  - Non-volatile device with limited write life (can be an independent disk)
  - Low power (6-8X lower than disks, 2X lower than DRAM)
  - Fast random read (200X faster than disks, 25X slower than DRAM)
  - Slow writing (300X slower than DRAM, 12X faster than disks)
  - Relatively expensive (8X more than disks, 5X cheaper than DRAM)

Research and Challenges

- New issues in Multicore
  - To utilize parallelism/concurrency in multicore is challenging
  - Resource sharing in multicore causes new problems
  - OS scheduling is multi-core- and shared-resources-unaware
  - Challenges: OS management scope needs to be enhanced.
- Low latency data accesses is most desirable
  - Sequential locality in disks is not effectively exploited.
  - Where should SSD be in the storage hierarchy?
  - How to use SSD and DRAM to improve disk performance and energy in a cost-effective way?
  - Challenges: disks are not in the scope of OS managements

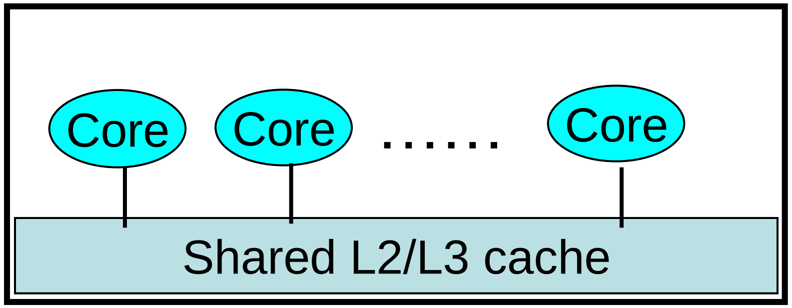
Multi-Core is the only Choice to Continue Moore's Law



R.M. Ramanathan, *Intel Multi-Core Processors: Making the Move to Quad-Core and Beyond*, white paper

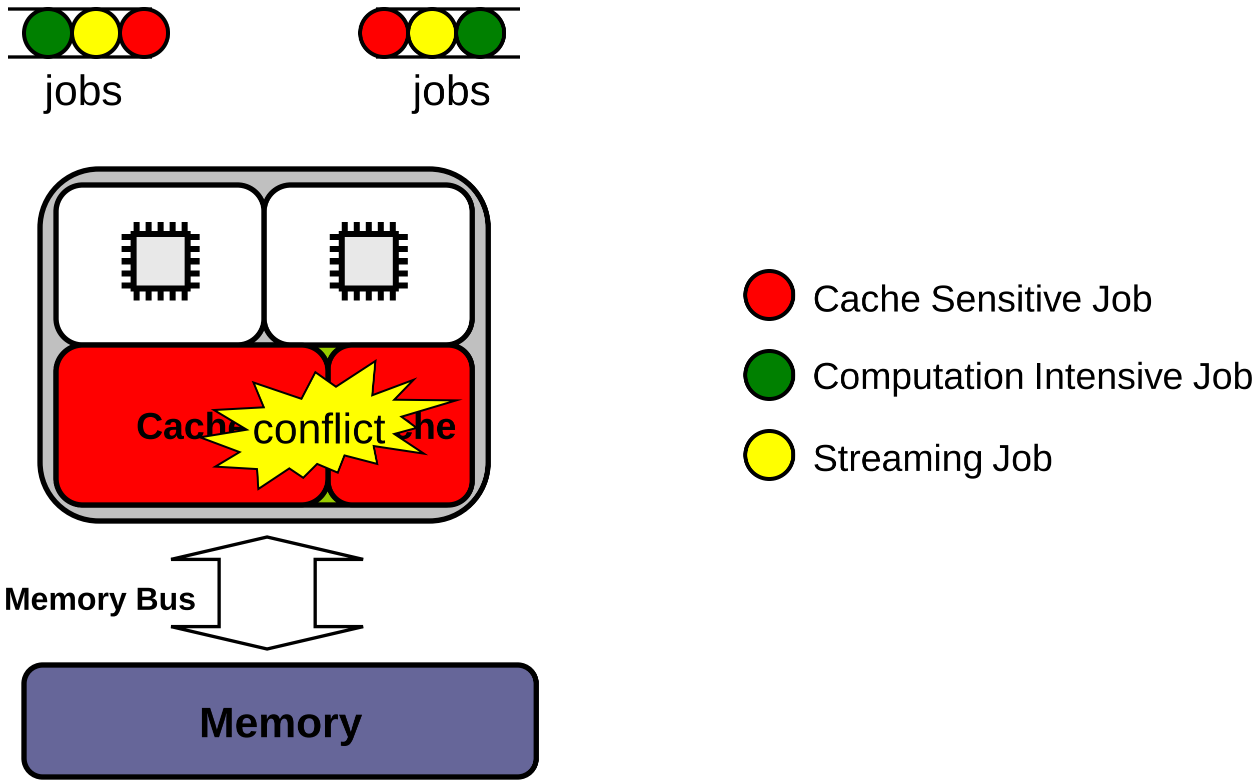
Shared Caches Can be a Critical Bottleneck

- Last Level Caches (LLC) are shared by multiple cores
  - Intel Xeon 51xx (2core/L2)
  - AMD Barcelona (4core/L3)
  - Sun T2, ... (8core/L2)



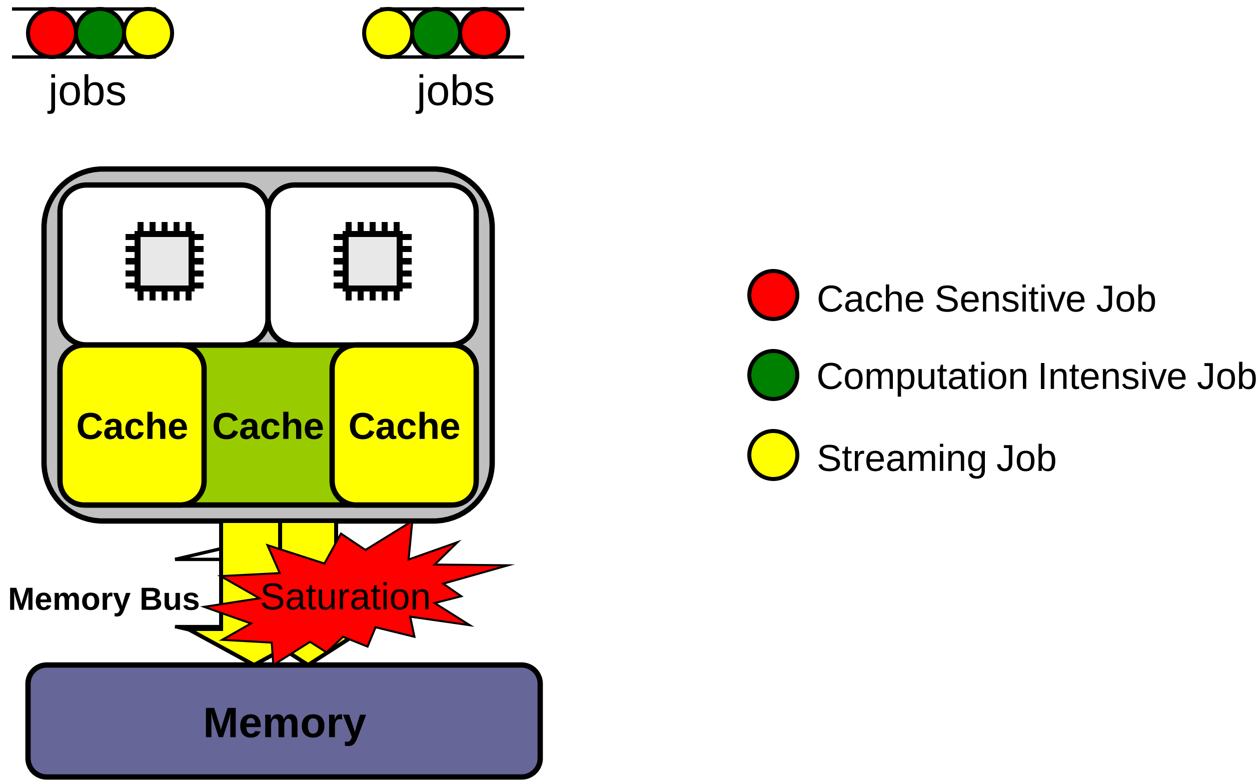
- Cache partitioning: allocate cache space to each process based their needs, fairness, and QoS.
- Hardware partitioning methods proposed in research
  - Performance: [HPCA'02], [HPCA'04], [Micro'06]
  - Fairness: [PACT'04], [ICS'07], [SIGMETRICS'07]
  - QoS: [ICS'04], [ISCA'07]
- None of them have been adopted in multicores
  - Runtime overhead in critical path
  - Design is too complicated

Shared Resource Conflicts in Multicores



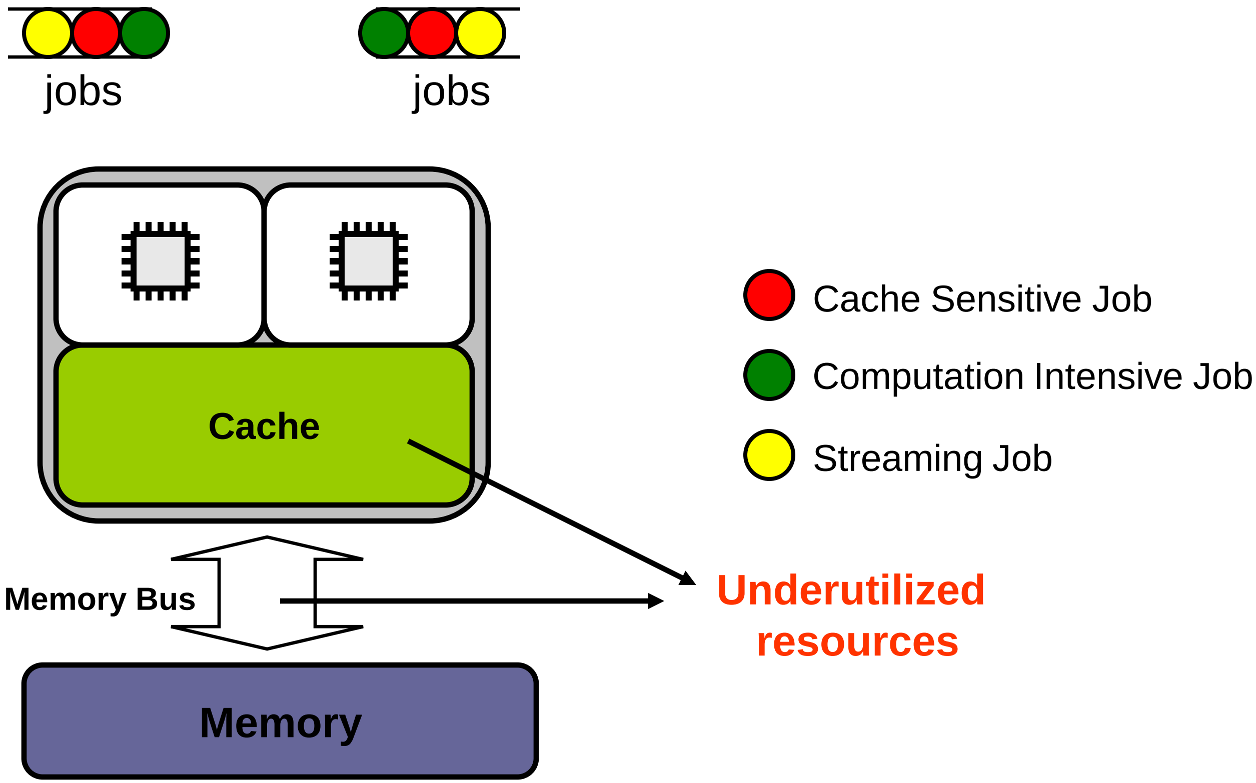
- **Scheduling two cache sensitive jobs - causing** cache ***conflicts***

Shared Resource Conflicts in Multicores



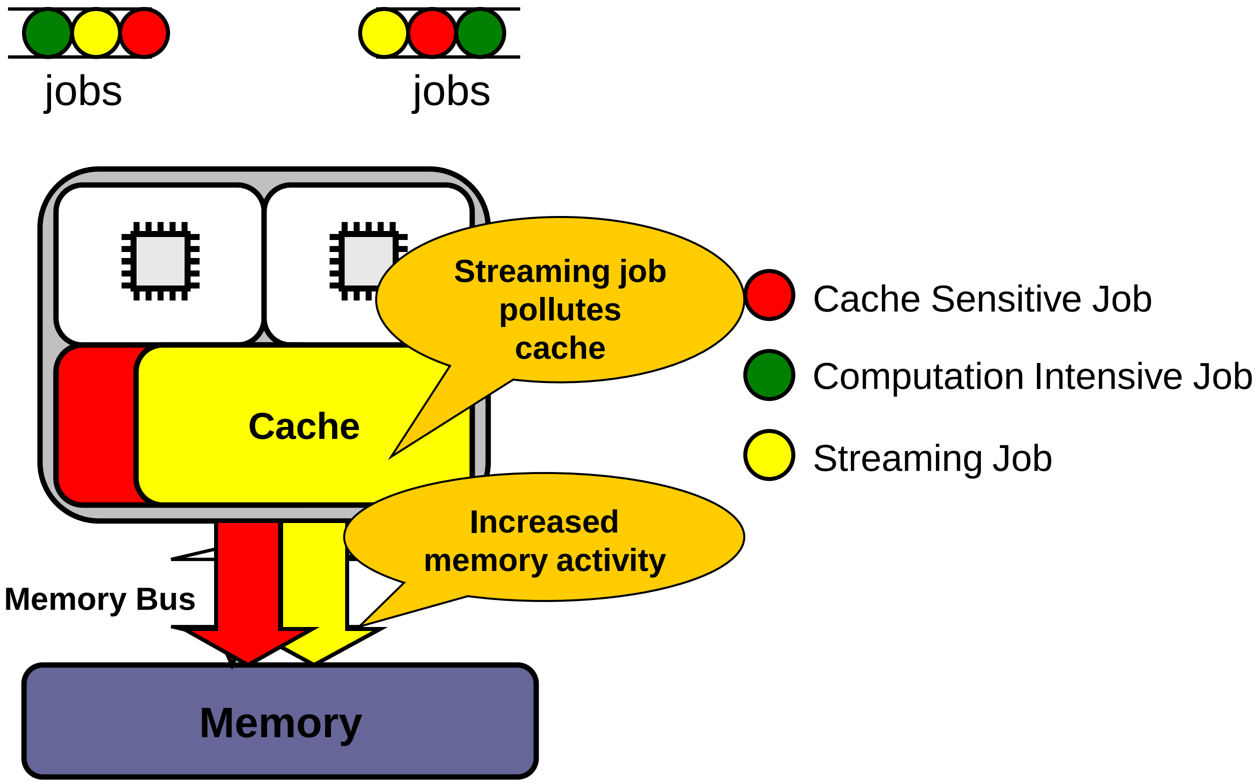
- **Scheduling two cache sensitive jobs - causing** cache ***conflicts***
- **Scheduling two streaming jobs - causing** memory bus ***congestions***

Shared Resource Conflicts in Multicores



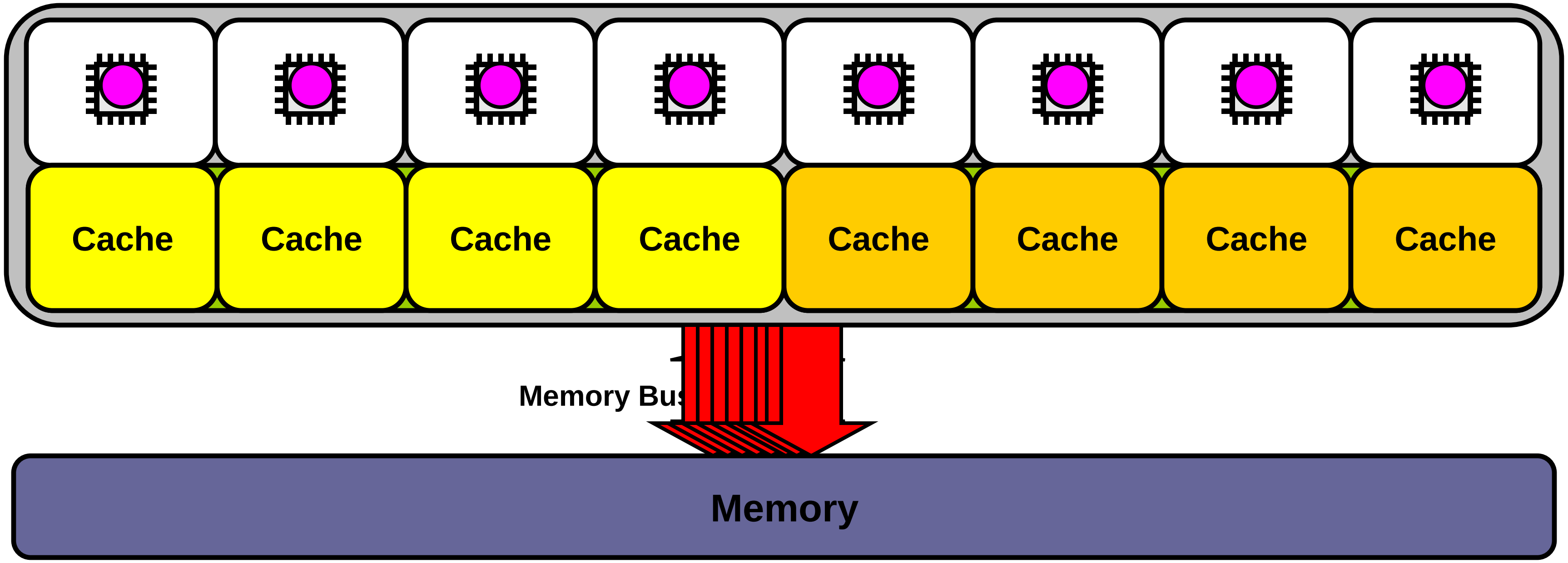
- **Scheduling two cache sensitive jobs - causing** cache ***conflicts***
- **Scheduling two streaming jobs - causing** memory bus ***congestions***
- **Scheduling two CPU intensive jobs – *underutilizing*** cache and bus

Shared Resource Conflicts in Multicores



- **Scheduling two cache sensitive jobs - causing** cache ***conflicts***
- **Scheduling two streaming jobs - causing** memory bus ***congestions***
- **Scheduling two CPU intensive jobs *– underutilizing*** cache and bus
- **Scheduling cache sensitive & streaming jobs *– conflicts & congestion***

Challenges of Many Cores, ***Shared*** Cache, ***Single*** Bus



- ***Many*** Cores – oversupplying computational power
- ***Shared*** Cache – lowering average cache capacity per process and per core
- ***Single*** Bus – increasing bandwidth sharing by many cores

Can OS be Able to Address All These Concerns?

- Inabilities of OS to handle workloads in multicores
  - Lacking application domain knowledge (static & dynamic)
  - Unaware of shared cache structures
  - Limited communication with hardware & programs
  - Insufficient information to effectively schedule threads
- To address all these concerns, OS must
  - frequently monitor and analyze application execution
  - directly interface with processor architecture
  - unaffordable tasks for both OS and multicore processors
- Hybrid approach is effective:
  - Applications monitor or give hints of access patterns
  - Scheduling can be at user level or get hints from application
  - OS indirectly manages the shared cache for space allocation
  - Design affordable hardware interface to support OS

Data-Intensive Scalable Computing (DISC)

Massively Accessing/Processing Data Sets in Fast Speed

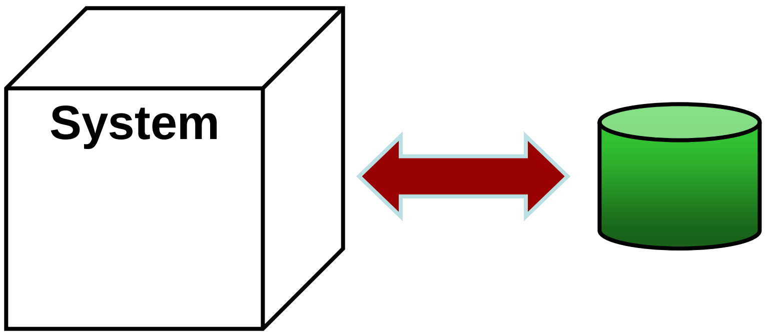
- drafted by R. Bryant at CMU, endorsed by Industries: Intel, Google, Microsoft, Sun, and scientists in many areas.
- Applications in science, industry, and business.

Special requirements for DISC Infrastructure:

- Top 500 DISC ranked by data throughput, as well FLOPS
- Frequent interactions between parallel CPUs and distributed storages. Scalability is challenging.
- DISC is not an extension of SC, but demands new technology advancements.

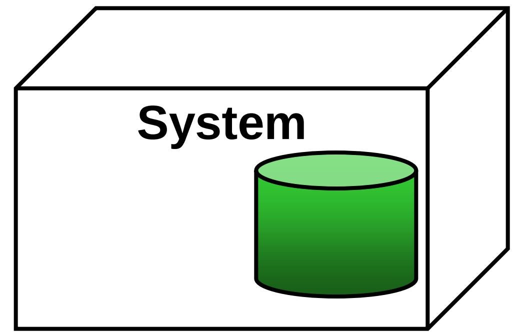
Systems Comparison: (courtesy of Bryant)

Conventional Computers



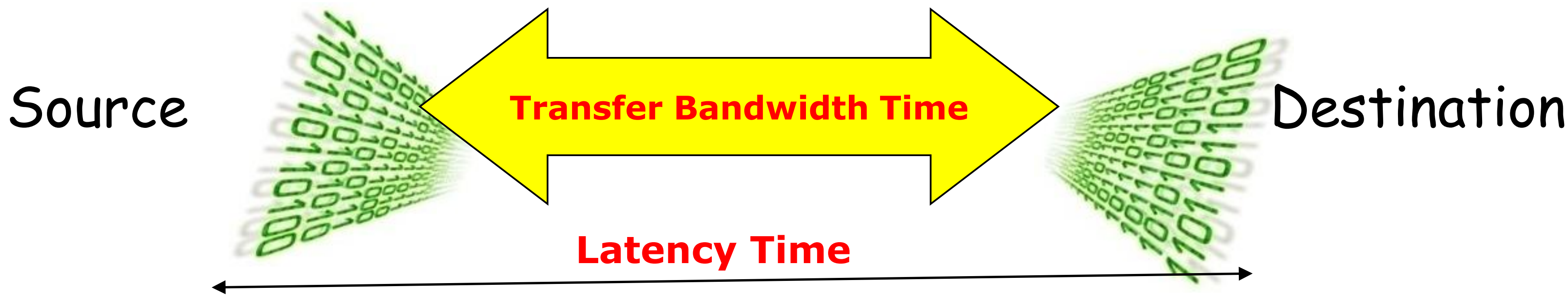
- Disk data stored separately
  - No support for collection or management
- Brought in for computation
  - Time consuming
  - Limits interactivity

DISC



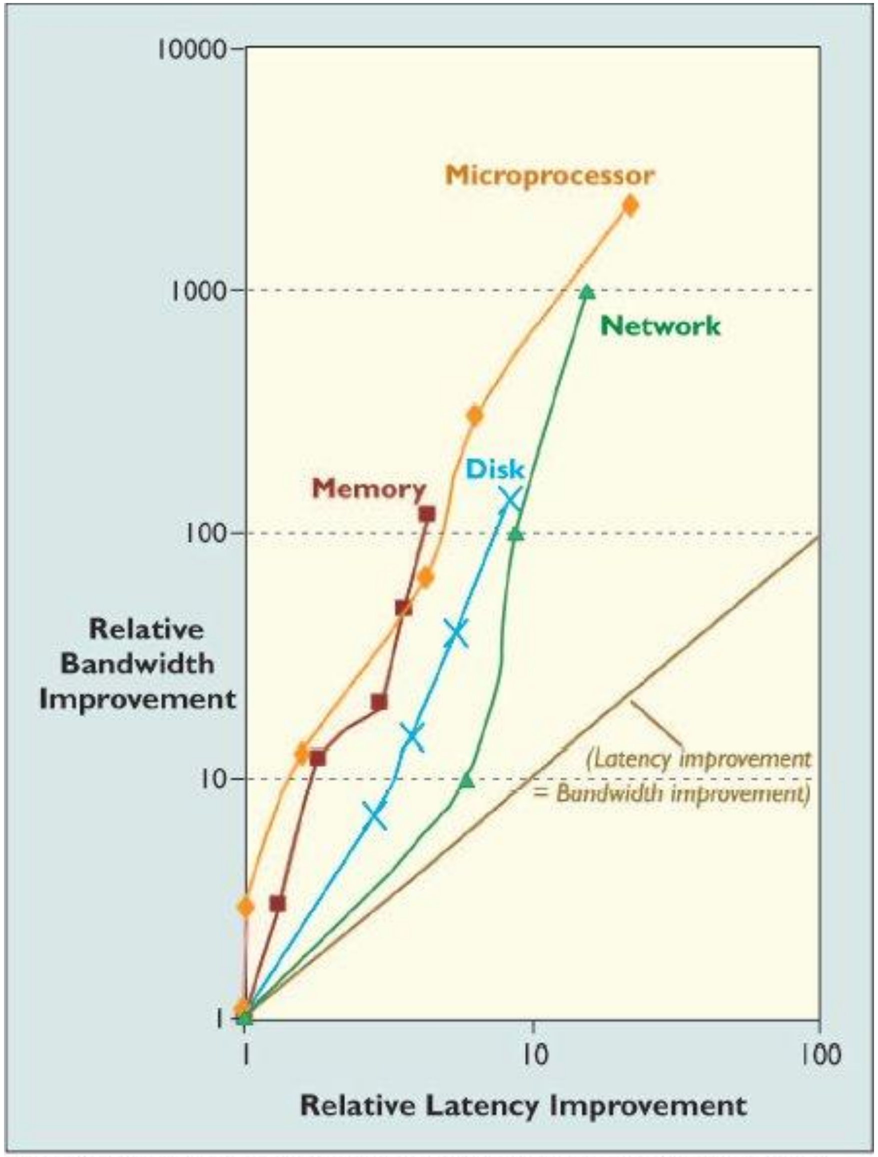
- System collects and maintains data
  - Shared, active data set
- Computation co-located with disks
  - Faster access

Data Communication in Computer Systems



Destination-perceived latency reduction is still limited due to imbalanced improvement of bandwidth and latency

Latency Lags Bandwidth (CACM, Patterson)



Note that latency improved about 10X while bandwidth improved about 100X to 1000X.

- In the last 20 years,
  100–2000X improvement in bandwidth
  5-20X improvement in latency

  Between CPU and on-chip L2:
  bandwidth: 2250X increase
  latency: 20X reduction

  Between L3 cache and DRAM:
  bandwidth: 125X increase
  Latency: 4X reduction

  Between DRAM and disk:
  bandwidth: 150X increase
  latency: 8X reduction

  Between two nodes via a LAN:
  bandwidth: 100X increase
  latency: 15X reduction

How is Resource Supply/Demand Balanced?

- **Slowdown CPU Speed:**
  - Earth Simulator: NEC AP, 500 MHz (4-way SU, a VU).
  - Blue Gene/L: IBM Power PC 440, 700 MHz.
  - Columbia: SGI Altix 3700 (Intel Itanium 2), 1.5 GHz. (commodity processors, no choice for its high speed)
- **Very low latency on-chip data accesses:**
  - Earth Simulator: 128K L1 cache and 128 large registers.
  - Blue Gene/L: on-chip L3 cache (2 MB).
  - Columbia: on-chip L3 cache (6 MB).
- **Fast accesses to huge and shared main memory.**
  - Earth Simulator: cross bar switches between AP and memory.
  - Blue Gene/L: cached DRAM memory, and 3-D torus connection.
  - Columbia: SGI NUMALink's data block transfer time: 50 ns.
- **Further latency reductions**: prefetching and caching.

Computing Operations Versus Data Movement

- **Computation is much cheaper than data movement**
  - In a 0.13 um CMOS, a 64-bit FPU < 1 mm^2, 16 FPUs can be easily placed in a 14mm * 14mm 1 GHz chip ($200).
  - Processing data from 16 registers (256 GB/s)
    - < $12.5/GFlop (60 mW/GFlop)
  - Processing data from on-chip caches (100 GB/s)
    - $32/Gflop (1 W/GFlops)
  - Processing data from off-chip memory (16 GB/s)
    - $200/Gflops (many Ws/GFlops)
  - Processing data from further location increases cost dramatically.
  - A vector machine with a lot FPUs and registers makes computations even cheaper.

- **Maximizing the fractions of local operations is the Key.**

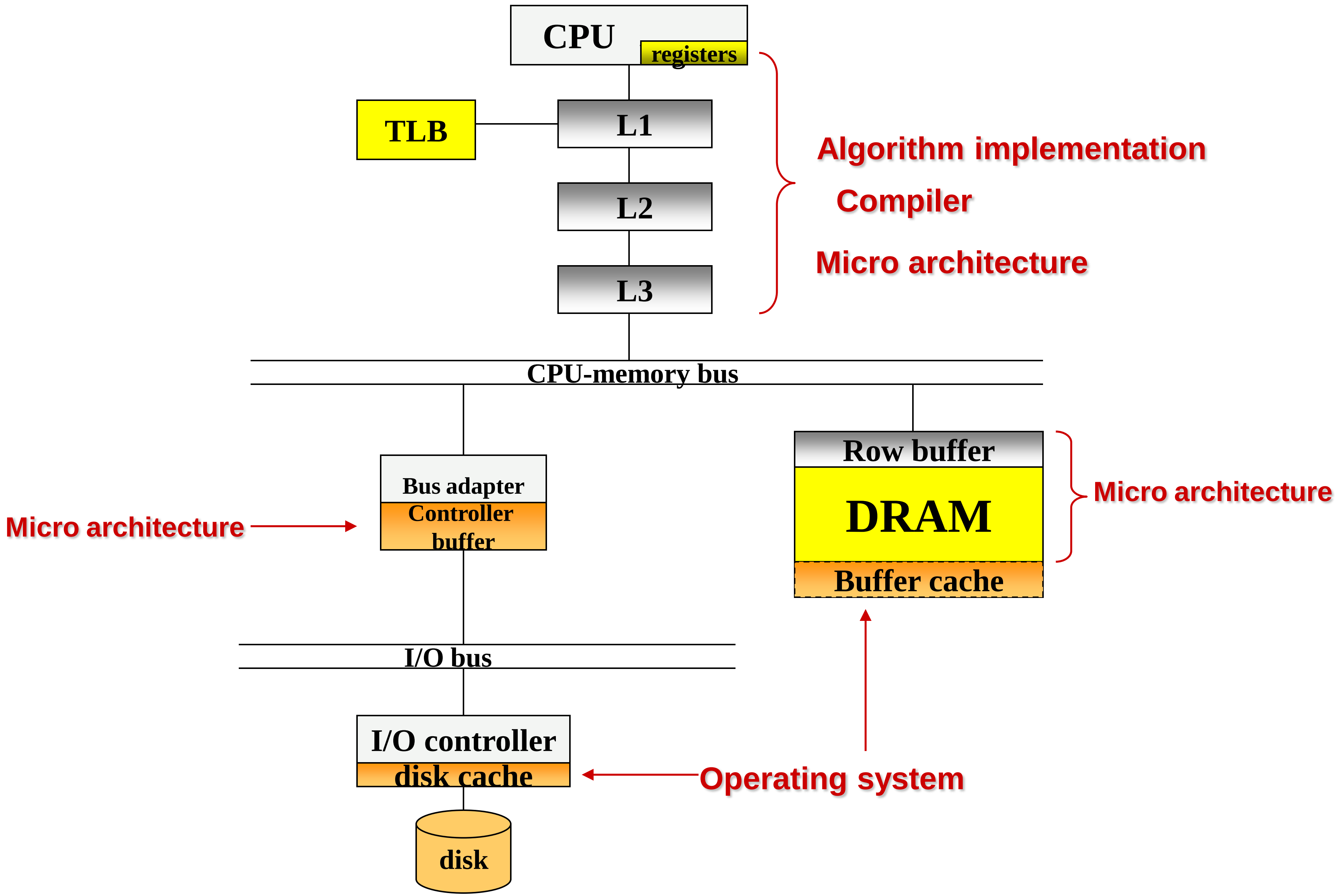
Challenges of Balancing Systems Cost-Effectively

- The special systems mainly rely on expensive customer designed CPUs, memory, and networks.

- Without such a large budget, what should we do?

- To cope with the bandwidth-latency imbalance, we must exploit locality anywhere if necessary by
  - Caching: reuse data in a relatively close place.
  - Replication: utilize large memory/storage capacity
  - Prefetching: utilize rich bandwidth to hide latency.

Where are Buffers in Deep Memory Hierarchy



Re-evaluation of Grid

- What is grid?
  - An infrastructure enables a set of resources (computing, data, network, et. al.) to be used by applications.
  - Expecting grid to do everything. (e.g. replacing high performance computing and cluster computing)
  - In reality, the scope of a grid is limited by existing infrastructure.

- Grid's scope have been exaggerated.
  - Overestimate its application demands.
  - Underestimate technology costs and market response.
  - The vision targets general applications, but development focuses on special workloads, e.g. scientific computing

Examining the Case of the TeraGrid Project in US

- A brief history of TeraGrid
  - This grid was initially built at 4 sites: NCSA, SDSC, AN, and Caltech (Pasadena), with an NSF grant of $53 M, August 2001.
  - October 02, NSF added another $35 M and included Pittsburg Supercomputing Center as the 5th partner.
  - October 03, NSF provided $10 M to add 4 other sites to the TeraGrid: ORNL, Purdue U., Indiana U., and U. of Texas.
  - August 05, NSF gave $150 M to maintain TeraGrid next 5 years.
- The Power of the TeraGrid
  - A accumulated total computing powers of 40+ TeraFlops.
  - 2 PeterByte (10^{15}) storage distributed in the 9 sites.
  - 9 sites are interconnected at 10-30 GB/s via a dedicated

NSF TeraGrid
Extensible Terascale Facility
Argonne
Purdue
PSC
NCSA
IU
ORNL
Caltech
SDSC
TACC

Major Types of Applications on TeraGrid

- Collaborations with timely analyzing/exchanging data sets.
  - Each collaborating site does independent data analysis, project solutions depend on periodically and quickly exchanging results. e.g. the ``telescope” project of studying cosmic rays. (UCI)
- Distributed simulations shared by multiple parties.
  - e.g. TeraShake (USC): huge earthquake simulations are operated at different sites, and results at different stages can be quickly shared by scientists in any TeraGrid site via high speed networks.
- Computing-/data-intensive jobs not fitting in a single site.
  - Effectively utilize dedicated computing powers and huge storage.

TeraGrid Model has its own Special Scope

- It does not represent a next generation Internet
  - It is dedicated and expensive, effective only for certain applications.
- It does not need a special distributed OS.
  - Management is done via middleware at user level.
- It does not need a special programming model.
  - The distributed execution facility is not transparent.
- It is not a source of free cycles.
  - Free cycles can be obtained at very low cost:
  - Accumulated cycles of SETI@home are over 60 Teraflops.
  - Huge amount free computing services from google, hotmail, and amazon.com.

Why is Grid not effective for High-end Computing?

- It will be extremely cost-ineffective to use dedicated links for message passing to run a parallel job in grid:
  - The fast links are for the purposes of data accesses of collaborations.
  - The communication is too expensive and too slow.
  - High end computing jobs should go to Blue Gene/L, ES, and others.
- To maintain a 40 TeraGrid is much more expensive than a tightly coupled high end system, such as ES.
  - The interfaces among different sites are much more complicated.
  - The maintenance cost of each site can be as high as ES.
  - Fast links across the country are very expensive.
- Locality is not a major concern in grid systems.
  - This is the key in high end computing.

Highly Computing Intensive Jobs with a Small Data Input

- A cryptographic search problem:
  - only a few Kbytes input/output, but computing for days.
- A representative job submitted to SETI@Home:
  - computing on 12 hours on 1/2 Mbytes of input
- A CFD computation at Cornell:
  - 7 years computing for 100 MB input, 10 GB output.
- Making animated movie of *Toy Story*:
  - a 200 MB image to take several hours to render.
- These are suitable to Grid systems slow links.

Resource Optimization and Utilization in Grid

- Bandwidths are much more expensive than cycles!
  - A rule of thumb: to send a GB over Grid links to save years of computing is much more meaningful than to send a KB if the job can be done locally in a second.
- Internet cost drops slower than Moore's Law.
- Cluster computing has different cost model
  - Unlike Internet, clusters do not have a monthly fee.
  - a GBps Ethernet costs $200/port, delivers 50 MBps.
  - it is comparable to disk bandwidth cost. (Clusters are the best homes for many large scientific applications).

A Foundation of Distributed Computing: Resource Virtualization

- **Objectives**
  - Share expensive facilities by different apps/users.
  - Provide simplified views of computing resources.
- **Hardware-level virtualization**
  - An instruction set shared by different chips (e.g. Intel IA-32)
- **OS Level virtualization**
  - Multiple OS context switch in a single system (e.g. VMware).
- **Hardware/OS virtualization**
  - Hiding dependency between hardware and OS (e.g. NGSCB, MS)
- **Cluster resource virtualization**
  - Workload migration among different networked nodes

Cost of System Virtualization

- **Communication overhead**
  - Execute jobs remotely with data communication
  - low data-communication efficiency
  - Limited system scalability, e.g. shared-virtual memory
- **Processing overheads at different levels**
  - Hardware adoptions, Instruction set emulator, VM monitors, VM executors in OS, ….
  - Low physical resource utilization
- **Loosing opportunities of Locality optimization**
  - Lacking controls of data layout in physical layers
- Cost-effective solutions:
  - Minimize all the above costs if any
  - Big benefits gain with small overheads

Grid: Internet Resource Virtualization

- **Trade-offs between resource replications and virtualization**
  - As rapid cost drop of computing resources (CPU, memory, I/O, …), global resource virtualization demand declines.
  - Virtualization is cost-effective to non-replicable resources.
  - Internet data transfers are expensive.
- **Internet management is autonomous system (AS) based**
  - Each AS consists of networks administrated by a single org.
  - Data transfers/management among ASes are slow/complex.
  - A Grid can be an AS, such as TeraGrid.
- **Replication and caching first, virtualization second.**
  - Only after the low cost and simple effort does not work, …

Lessons Learned from Grid Projects in US

- **The scope of grid model is limited to specific applications**
  - Collaborations on common data sets
  - Sharing expensive facilities via Internet
- **Network communication is assumed to be (almost) free**
  - Data transfer is very expensive but computing is free
  - Scheduling and resource allocations are not cost-effective
- **Principle of locality is not considered**
  - Caching/prefetching is powerful everywhere in systems
- **Resource virtualization for “virtualization”**
  - Replications can be faster more cost-effective solutions
- **Cost was not a serious consideration in grid model**
  - A common mistake made by government initiatives
  - For a given budget, where do we make investment, networks, servers, storages, to gain the maximum performance.

Cloud Computing: A Low Cost Resource Sharing Model

- **Computing service is a standard utility**
  - Users and corporations contract the services by units.
  - Significantly reduce the IT personal and infrastructure costs
  - Well utilize rich computing, storage, and Internet resources
- **Principles of cloud computing**
  - Cost-effectiveness is the basis for computing, storage, and communication models in cloud computing (SIGCOM'09)
  - Targeting standard computing model in a wide range
  - Exploiting locality and load sharing with low overhead
- **New challenges** (CS@Berkeley, 2009)

(1) availability of service; (2) sharing data in different platforms; (3) data security; (4) minimizing communication cost; (5) unpredictable performance; (6) scalability of storage; (7) reliability of large scale distributed systems; (8) service scalability; (9) trust to the cloud service; and (10) software licensing

Conclusion

- Technology advancement driven (Moore's Law).
  - Multicore adds another dimension of parallelism and others
  - Memory bandwidth becomes bottleneck
  - Power consumption would limit wide deployment of IT
- Amdal's Law is a system design principle
  - Critical issues determine the overall performance:
    - Data access latency and memory bandwidth
- Principle of Locality is a foundation
  - Latency reduction by caching, prefetching and replication
  - Effectively exploiting locality at all system layers is the key

- Cloud computing must follow the three laws/principle

Final Words

- Two quotes from Bertrand Russell (罗素, 1872-1970)
  - I think we ought always to entertain our opinions with some measure of doubt. I shouldn't wish people dogmatically to believe any philosophy, not even mine.
  - In all affairs it's a healthy thing now and then to hang a question mark on the things you have long taken for granted.
- Many new concepts have been proposed
  - Grid, P2P, virtualization, cloud computing, ….
  - we should have doubts and questions about them
- Foundations of technology advancement
  - Science discerns the laws of nature; industries (technologies) apply them to the needs of man. (Chicago Science and Industry Musuem)