COMP4611: Design and Analysis of Computer Architectures

Warehouse-Scale Computers to Exploit Request-Level and Data-Level Parallelism

Lin Gu

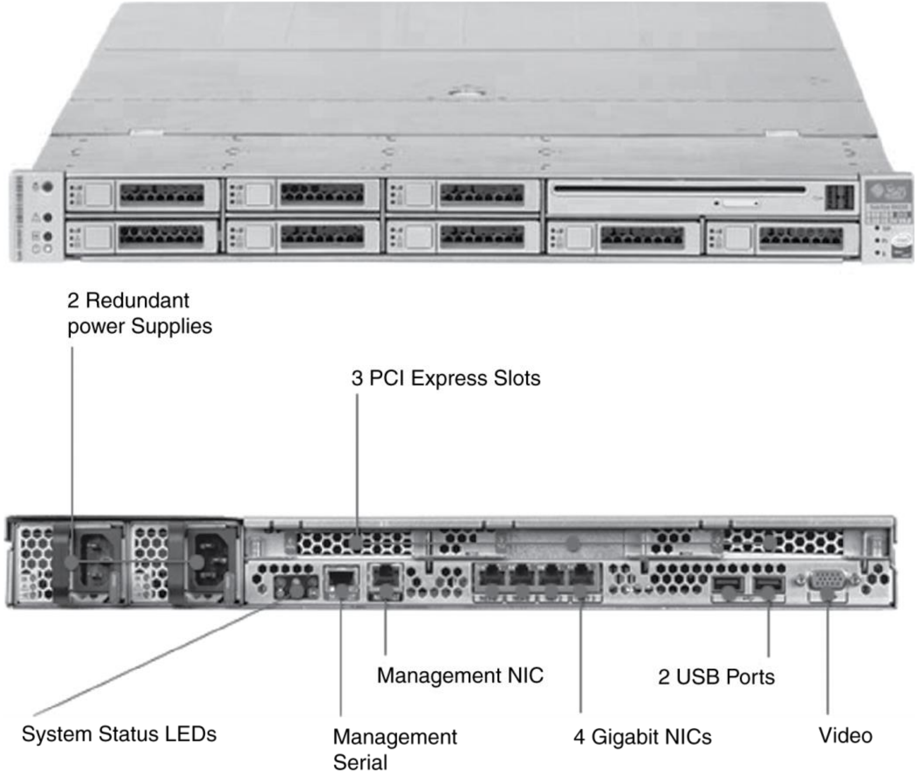
CSE, HKUST

Server Computers

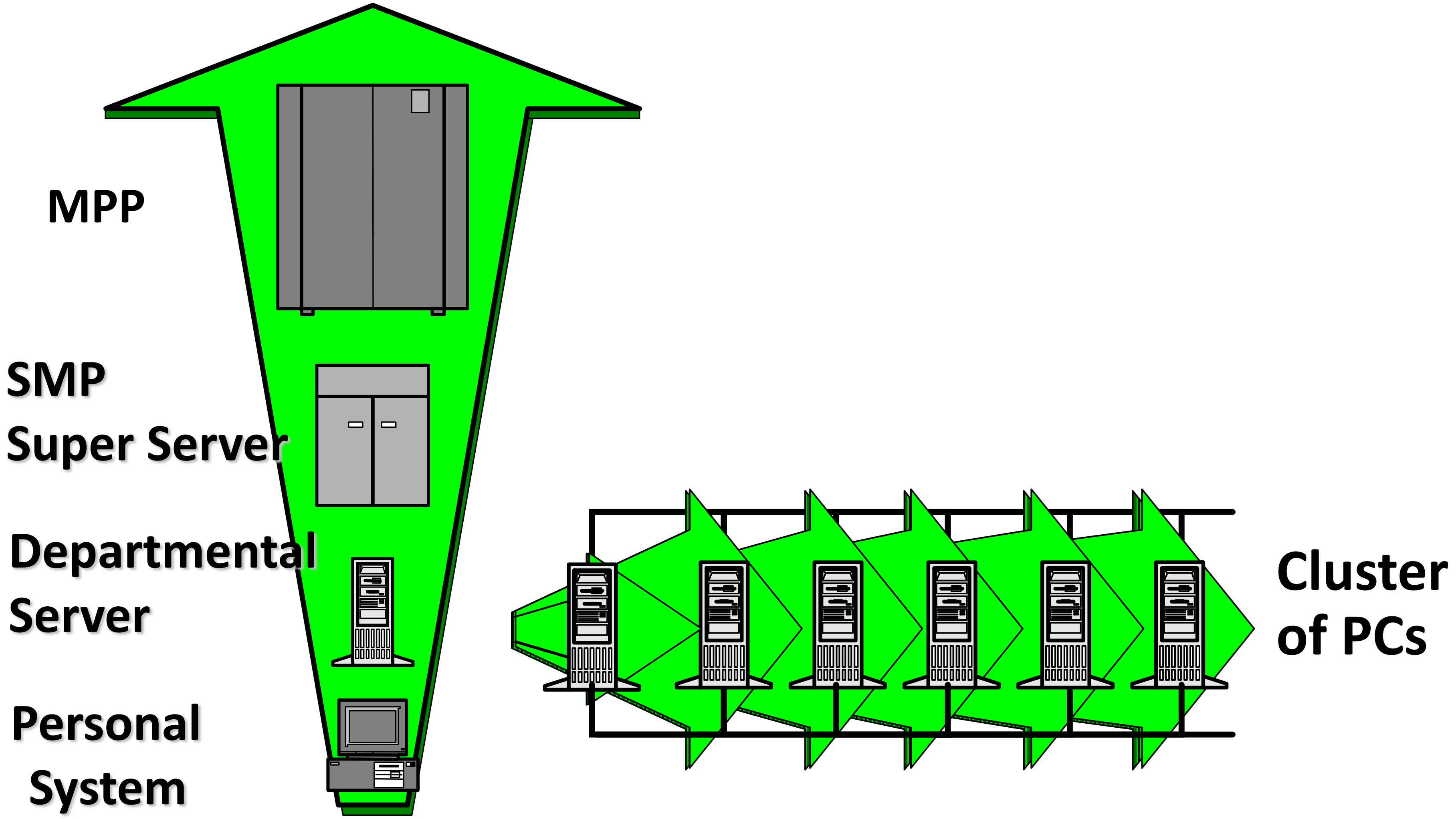
- Applications are increasingly run on servers
  - Web search, office apps, virtual worlds, …
- Requires large data center servers
  - Multiple processors, networks connections, massive storage
  - Space and power constraints
- Rack server equipment often in units of 1.75” (1U).
  - E.g., a 1U switch, a 2U server

Rack-Mounted Servers

Sun Fire x4150 1U server



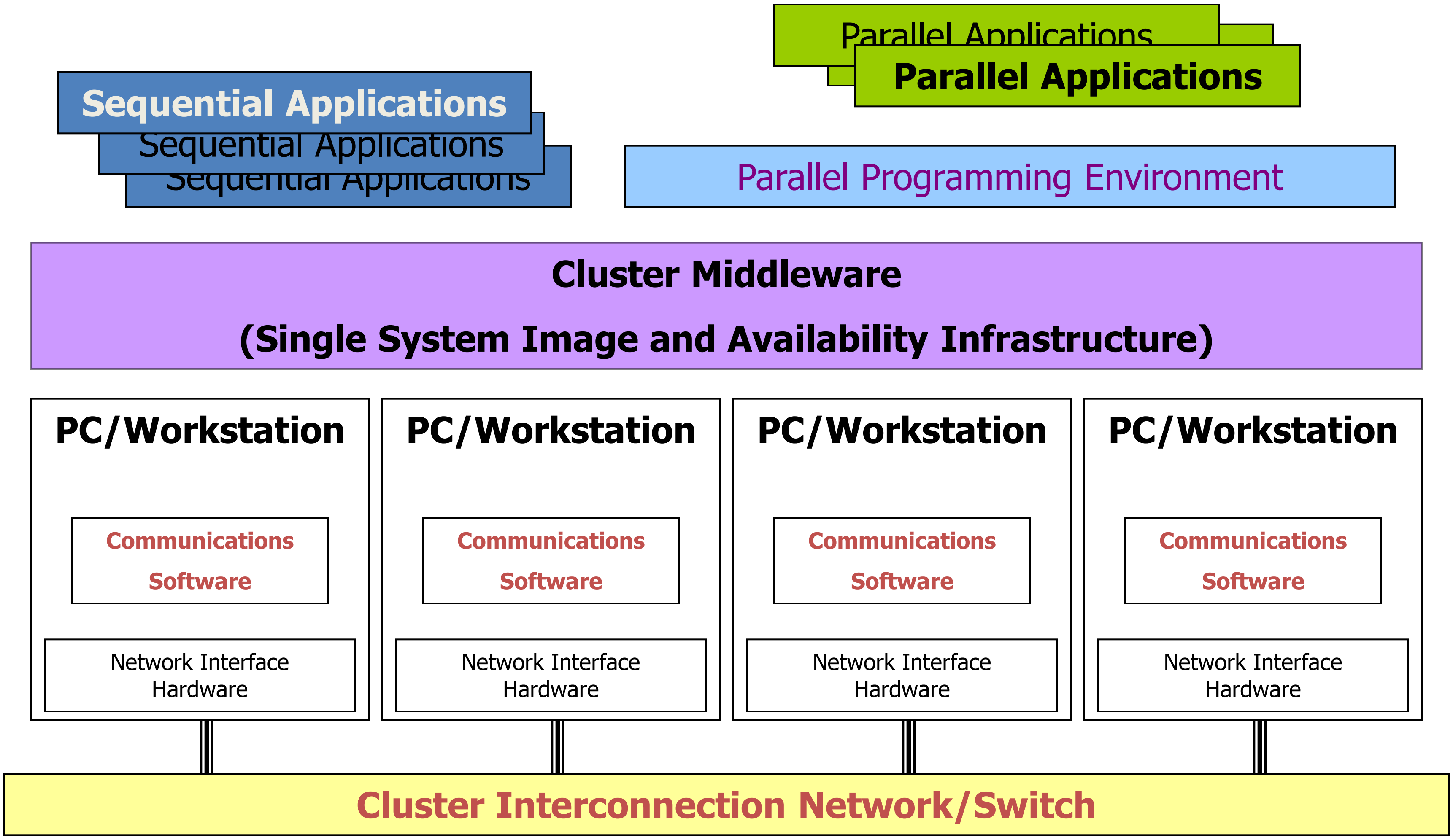
Scale Up vs. Scale Out



Motivations of using Clusters over Specialized Parallel Computers

- Individual PCs are becoming increasingly powerful
- Communication bandwidth between PCs is increasing and latency is decreasing (Gigabit Ethernet, Myrinet)
- PC clusters are easier to integrate into existing networks
- Typical low user utilization of PCs (<10%)
- Development tools for workstations and PCs are mature
- PC clusters are a cheap and readily available
- Clusters can be easily grown

Cluster Architecture



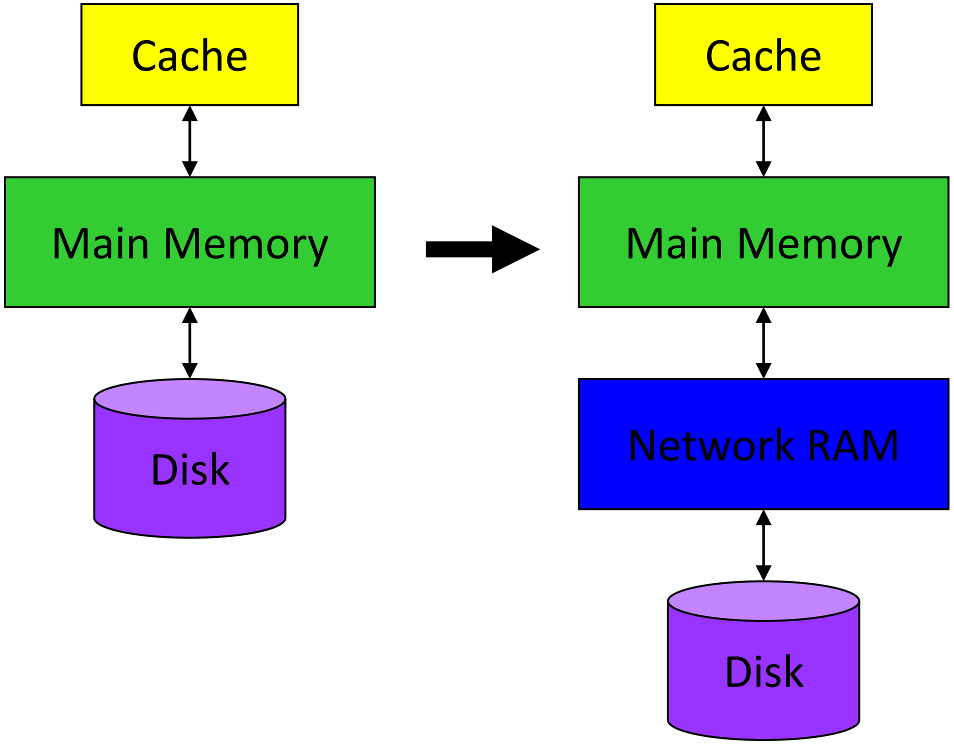
How Can we Benefit From Clusters?

- Given a certain user application

- **Phase 1**
  - If the application can run fast enough on a single PC, there is no need to do anything else
  - When it cannot, go to **Phase 2**
- **Phase 2**
  - Try to put the whole application on the DRAM to avoid going to the disk.
  - If that is not possible, use the DRAM of the other idle workstations
  - Network DRAM is 5 to 10 times faster than local disk

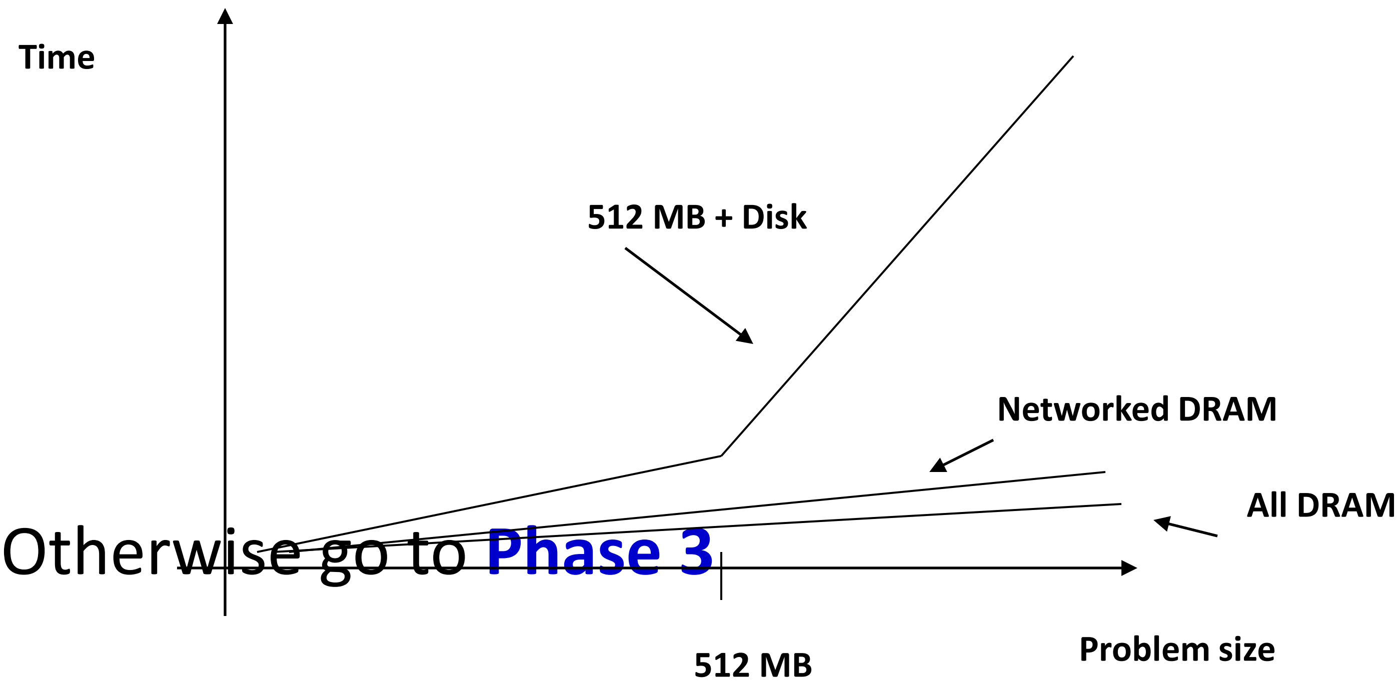
Remote Memory Paging

- Background
  - Application’s working sets have increased dramatically
  - Applications require more memory than a single workstation can provide.
- Solution
  - Inserts the Network DRAM in the memory hierarchy between local memory and the disk
  - Swaps the page to remote memory



How Can we Benefit From Clusters?

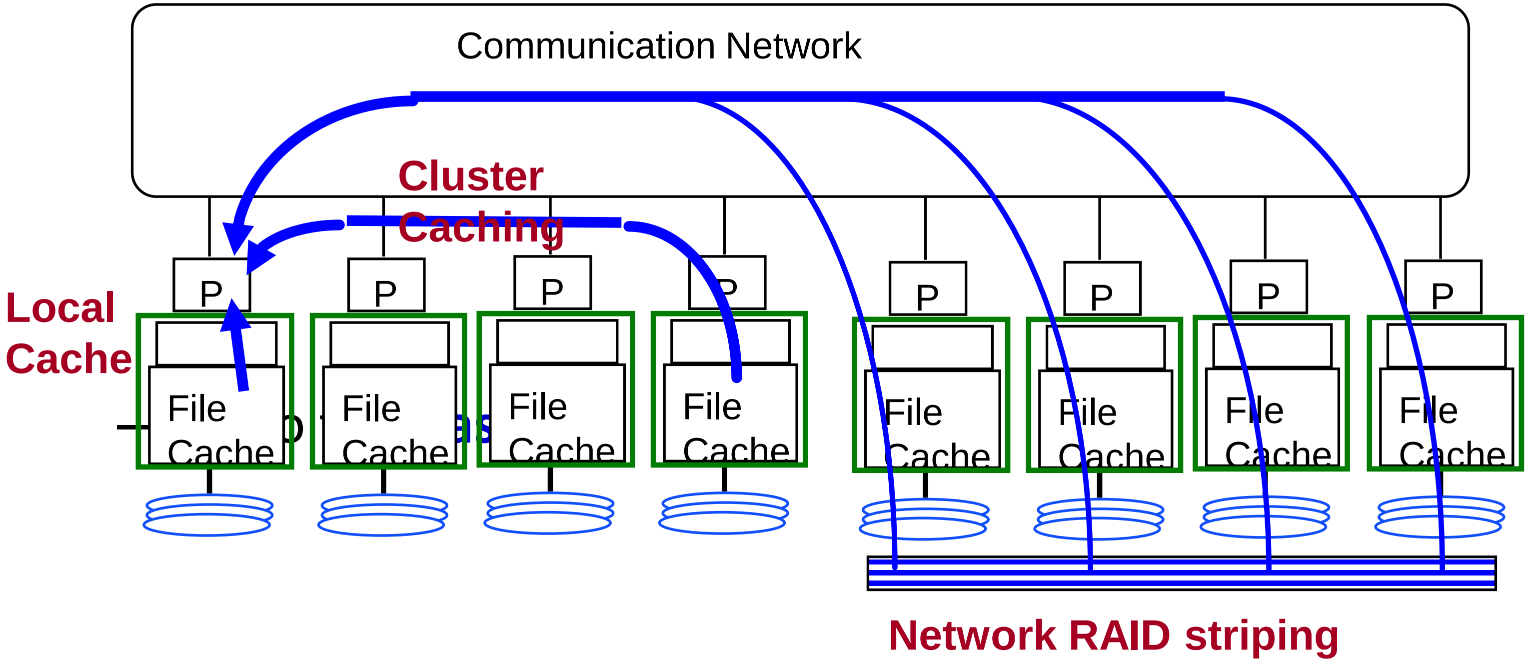
- In this case, the DRAM of the networked PCs behave like a ***huge cache*** system for the disk



- Otherwise go to **Phase 3**

How Can we Benefit From Clusters?

- **Phase 3**
  - If the network DRAM is not large enough, try using all the disks in the network in parallel for reading and writing data and program code (e.g., RAID) to speedup the I/O

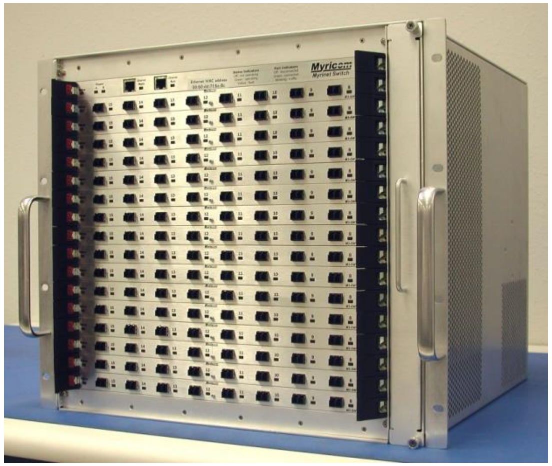


How Can we Benefit From Clusters?

- **Phase 4**
  - Execute the program on a multiple number of workstations (PCs) at the same time – **Parallel processing**
- **Tools**
  - There are many tools that do all these phases in a **transparent** way (except parallelizing the program) as well as load-balancing and scheduling.
    - **Beowulf (CalTech and NASA) - USA**
    - **Condor - Wisconsin State University, USA**
    - **MPI (MPI Forum, MPICH is one of the popular implementations)**
    - **NOW (Network of Workstations) - Berkeley, USA**
    - **PVM - Oak Ridge National Lab./UTK/Emory, USA**

What network should be used?

	Fast Ethernet	**Gigabit Ethernet**	**Myrinet**	**10GbE**
Latency	~120µs	~120 µs	~7 µs	10s of µs's
Bandwidth	~100Mbps peak	~1Gbps peak	~1.98Gbps real	10Gbps peak



Dependability

Definitions

- Examples on why precise definitions so important for reliability
- **Is a programming mistake a fault, error, or failure?**
  - **Are we talking about the time it was designed or the time the program is run?**
  - **If the running program doesn't exercise the mistake, is it still a fault/error/failure?**
- **If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?**
  - **Is it a fault/error/failure if the memory doesn't access the changed bit?**
  - **Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?**

IFIP Standard terminology

- Computer system ***dependability***: quality of delivered service such that reliance can be justifiably placed on the service
- ***Service*** is observed ***actual behavior*** as perceived by other system(s) interacting with this system's users
- Each module has ideal ***specified behavior***, where ***service specification*** is agreed description of expected behavior
- A system ***failure*** occurs when the actual behavior deviates from the specified behavior
- failure occurred because an ***error***, a defect in module
- The cause of an error is a ***fault***
- When a fault occurs it creates a ***latent error***, which becomes ***effective*** when it is activated
- When error actually affects the delivered service, a failure occurs (time from error to failure is ***error latency***)

Fault v. (Latent) Error v. Failure

- **An error is manifestation *in the system* of a fault,**
  a failure is manifestation *on the service* of an error
- **If an alpha particle hits a DRAM memory cell, is it a fault/error/failure if it doesn't change the value?**
  - **Is it a fault/error/failure if the memory doesn't access the changed bit?**
  - **Did a fault/error/failure still occur if the memory had error correction and delivered the corrected value to the CPU?**
- **An alpha particle hitting a DRAM can be a fault**
- **If it changes the memory, it creates an error**
- **Error remains latent until affected memory word is read**
- **If the effected word error affects the delivered service, a failure occurs**

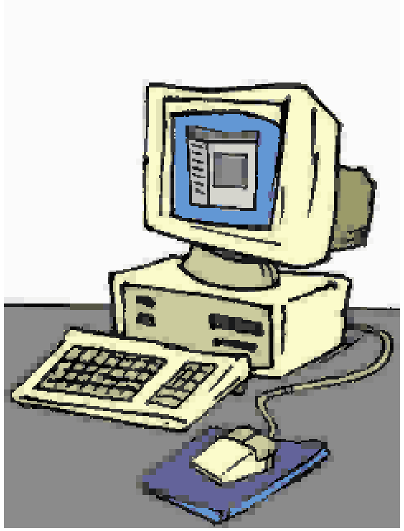
Fault Categories

1. Hardware faults: Devices that fail, such alpha particle hitting a memory cell
2. Design faults: Faults in software (usually) and hardware design (occasionally)
3. Operation faults: Mistakes by operations and maintenance personnel
4. Environmental faults: Fire, flood, earthquake, power failure, and sabotage

- Also by duration:

1. Transient faults exist for limited time and not recurring
2. Intermittent faults cause a system to oscillate between faulty and fault-free operation
3. Permanent faults do not correct themselves over time

Fault Tolerance vs Disaster Tolerance

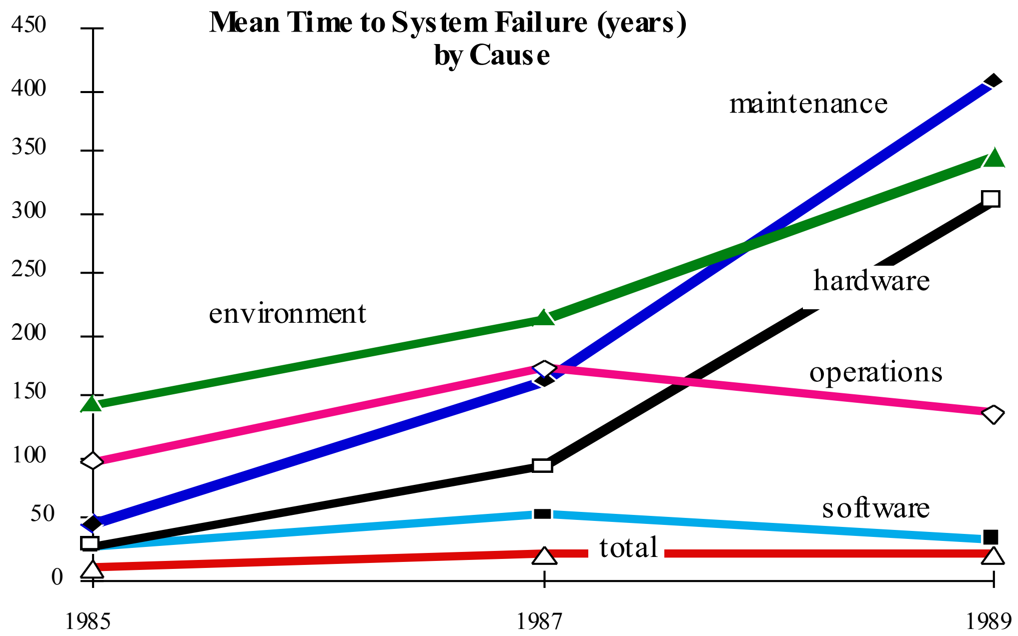


- **Fault-Tolerance (or more properly, Error-Tolerance):** mask local faults
  (prevent errors from becoming failures)
  - RAID disks
  - Uninterruptible Power Supply (UPS)
  - Cluster Failover
- **Disaster Tolerance:** masks site errors
  (prevent site errors from causing service failures)
  - Protects against fire, flood, sabotage,..
  - Redundant system and service at remote site.
  - Use design diversity

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

Case Studies - Tandem Trends

Reported MTTF by Component



	1985	1987	1990	
SOFTWARE	2	53	33	Years
HARDWARE	29	91	310	Years
MAINTENANCE	45	162	409	Years
OPERATIONS	99	171	136	Years
ENVIRONMENT	142	214	346	Years
SYSTEM	**8**	**20**	**21**	**Years**

Problem: Systematic Under-reporting

From Jim Gray's "Talk at UC Berkeley on Fault Tolerance " 11/9/00

HW Failures in Real Systems: Tertiary Disks

A cluster of 20 PCs in seven 7-foot high, 19-inch wide racks with 368 8.4 GB, 7200 RPM, 3.5-inch IBM disks. The PCs are P6-200MHz with 96 MB of DRAM each. They run FreeBSD 3.0 and the hosts are connected via switched 100 Mbit/second Ethernet. Data collected during 18 months of operation.

Component	Total in System	Total Failed	% Failed
SCSI Controller	44	1	2.3%
SCSI Cable	39	1	2.6%
SCSI Disk	368	7	1.9%
IDE Disk	24	6	25.0%
Disk Enclosure -Backplane	46	13	28.3%
Disk Enclosure - Power Supply	92	3	3.3%
Ethernet Controller	20	1	5.0%
Ethernet Switch	2	1	50.0%
Ethernet Cable	42	1	2.3%
CPU/Motherboard	20	0	0%

How Realistic is "5 Nines"?

- HP claims HP-9000 server HW and HP-UX OS can deliver 99.999% availability guarantee “in certain pre-defined, pre-tested customer environments”
  - Application faults?
  - Operator faults?
  - Environmental faults?
- Collocation sites (lots of computers in 1 building on Internet) have
  - 1 network outage per year (~1 day)
  - 1 power failure per year (~1 day)
- In 2008, Microsoft Network unavailable for a day due to problem in Domain Name Server: if only outage per year, 99.7% or 2 Nines

Data Processing for Today's Web-Scale Services

What happens if we have a thousand disks?

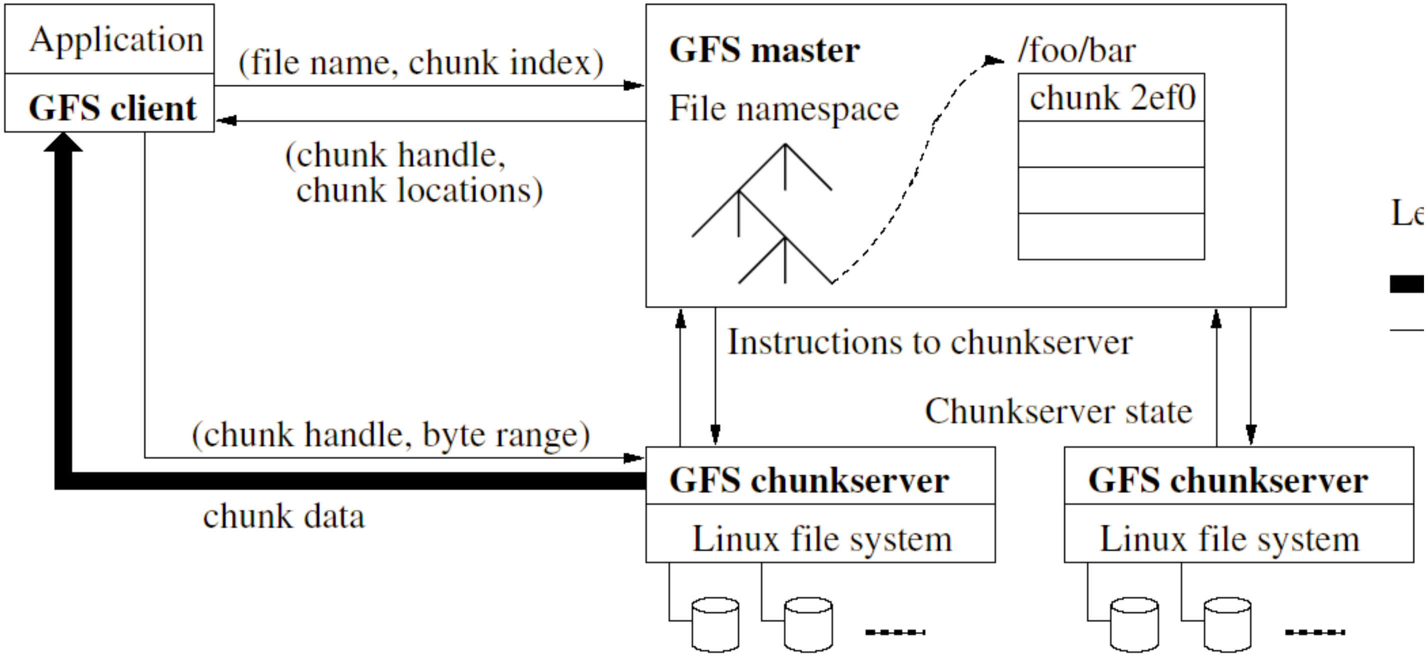
Application tasks	Data size	Computation	Network
Web crawl	800TB	Highly parallel	High bandwidth
Data analytics	200TB	Intensive	High bandwidth, low latency
Orkut (social network)	9TB	Parallelizable	Low latency
Youtube	Estimated multi-petabytes	Intensive, parallelizable	Very high bandwidth, low latency
e-business (e.g., Amazon)	Estimated multi-petabytes	Intensive	High bandwidth, very low latency

– Petabytes of data and demanding computation

– Network performance is essential!

Chang, F. et al. Bigtable: a distributed storage system for structured data. In Proceedings of the 7th Symposium on Operating Systems Design and Implementation (Seattle, Washington, November 06 - 08, 2006). 205-218. http://baijia.info/showthread.php?tid=4

Large-Scale Fault Tolerant File System



- **A distributed file system at work (GFS)**
  - **Single master and numerous slaves communicate with each other**
  - **File data unit, “chunk”, is up to 64MB. Chunks are replicated.**
- **Requires extremely high network bandwidth, very low network latency**

Appendix

2007 Top500 List

- Clusters are the fastest growing category of supercomputers in the TOP500 List.
  - 406 clusters (81%) in November 2007 list
  - 130 clusters (23%) in the June 2003 list
  - 80 clusters (16%) in the June 2002 list
  - 33 clusters (6.6%) in the June 2001 list
- 4% of the supercomputers in the November 2007 TOP500 list use Myrinet technology!
- 54% of the supercomputers in the November 2007 TOP500 list Gigabit Ethernet technology!

Introduction

- Warehouse-scale computer (WSC)
  - Provides Internet services
    - Search, social networking, online maps, video sharing, online shopping, email, cloud computing, etc.
  - Differences with HPC “clusters”:
    - Clusters have higher performance processors and network
    - Clusters emphasize thread-level parallelism, WSCs emphasize request-level parallelism

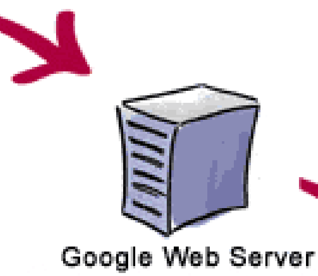
Introduction

Important design factors for WSC:

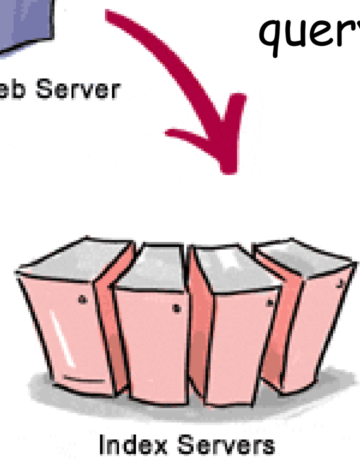
- Cost-performance
  - Small savings add up
- Energy efficiency
  - Affects power distribution and cooling
  - Work per joule
- Dependability via redundancy
- Network I/O
- Interactive and batch processing workloads
- Ample computational parallelism is not important
  - Most jobs are totally independent
  - "Request-level parallelism"
- Operational costs count
  - Power consumption is a primary, not secondary, constraint when designing system
- Scale and its opportunities and problems
  - Can afford to build customized systems since WSC require volume purchase

Google

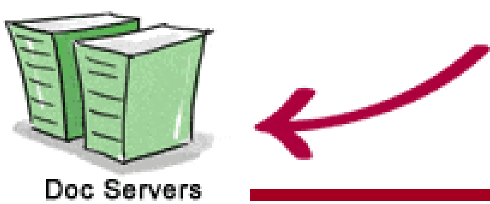
1. The user enters a query on a web form sent to the Google web server.



2. The web server sends the query to the Index Server cluster, which matches the query to documents.



3. The match is sent to the Doc Server cluster, which retrieves the documents to generate abstracts and cached copies.



4. The list, with abstracts, is displayed by the web server to the user, sorted (using a secret formula involving PageRank).

Google Requirements

- Google: search engine that scales at Internet growth rates
- Search engines: 24x7 availability
- Google : 600M queries/day, or AVERAGE of 7500 queries/s all day (per an earlier report)
- Google crawls WWW and puts up new index every 2 weeks (old data)
- Storage: 5.3 billion web pages, 950 million newsgroup messages, and 925 million images indexed, Millions of videos (very old data)
- Response time goal: < 0.5 s per search (old data)

Google

(Based on old data)

- Require high amounts of computation per request
- A single query on Google (on average)
  - reads hundreds of megabytes of data
  - consumes tens of billions of CPU cycles
- A peak request stream on Google
  - requires an infrastructure comparable in size to largest **supercomputer** installations
- Typical google Data center: 15000 PCs (Linux), 30000 disks: many petabytes!
- Google application affords easy parallelization
  - Different queries can run on different processors
  - A single query can use multiple processors
    - because the overall index is partitioned

Prgrm'g Models and Workloads

Batch processing framework: MapReduce

- **Map:** applies a programmer-supplied function to each logical input record
  - Runs on thousands of computers
  - Provides new set of key-value pairs as intermediate values
- **Reduce:** collapses values using another programmer-supplied function

Prgrm’g Models and Workloads

Example:

- **map (String key, String value)**:
  - **// key: document name**
  - **// value: document contents**
  - **for each word w in value**
    - **EmitIntermediate(w,”1”); // Produce list of all words**

- **reduce (String key, Iterator values):**
  - **// key: a word**
  - **// value: a list of counts**
  - **int result = 0;**
  - **for each v in values:**
    - **result += ParseInt(v); // get integer from key-value pair**
  - **Emit(AsString(result));**

Prgrm’g Models and Workloads

- **MapReduce runtime environment schedules map and reduce task to WSC nodes**
- **Availability:**
  - **Use replicas of data across different servers**
  - **Use relaxed consistency:**
    - **No need for all replicas to always agree**
- **Workload demands**
  - **Often vary considerably**

Program Execution on Web-Scale Data

The MapReduce Approach

Massive parallel processing made simple

- Example: world count
- Map: parse a document and generate <word, 1> pairs
- Reduce: receive all pairs for a specific word, and count (sum)

Map

```
// D is a document
for each word w in D
  output <w, 1>
```

Reduce

```
Reduce for key w:
count = 0
for each input item
  count = count + 1
output <w, count>
```

Cloud Software is evolving and multiplying quickly

This means we have not found the right solution, yet.

MapReduce/Hadoop

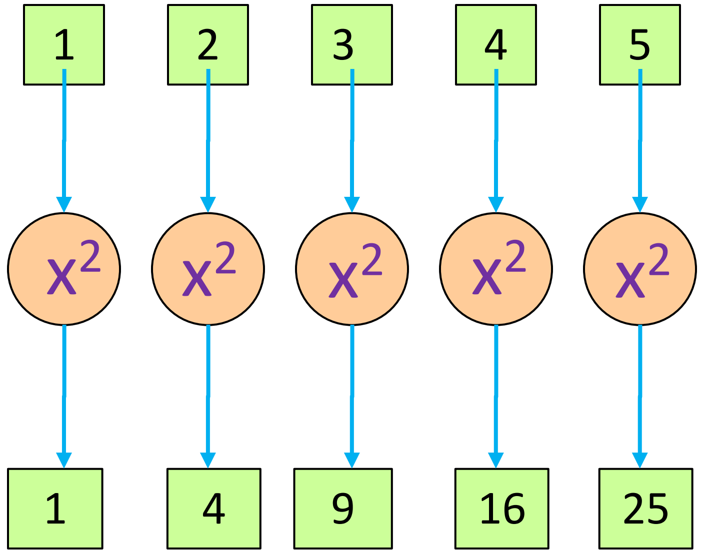
- Around 2004, Google invented MapReduce to parallelize computation of large data sets. It's been a key component in Google's technology foundation
- Around 2008, Yahoo! developed the open-source variant of MapReduce named Hadoop
- After 2008, MapReduce/Hadoop become a key technology component in cloud computing
- In 2010, the U.S. conferred the MapReduce patent to Google

MapReduce Hadoop … Hadoop or variants …

Data-Intensive Computation

The MapReduce Approach

- MapReduce—parallel computing for Web-scale data
- Map: a higher-order function applied to all elements in a list
  - Result is a new list

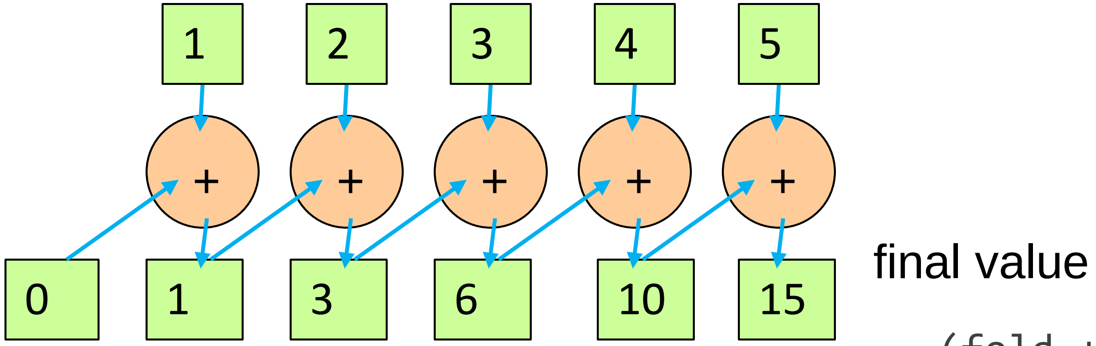



```
(map (lambda (x) (* x x))
     '(1 2 3 4 5))
 → '(1 4 9 16 25)
```

Data-Intensive Computation

The MapReduce Approach

- Reduce is also a higher-order function
- Like "fold": aggregate elements of a list
  - Accumulator set to initial value
  - Function applied to list element and the accumulator
  - Result stored in the accumulator
  - Repeated for every item in the list
  - Result is the final value in the accumulator




```
(fold + 0 '(1 2 3 4 5)) → 15
(fold * 1 '(1 2 3 4 5)) → 120
```

Problems of MapReduce/Hadoop

- Slow
  - Cannot support low-latency, interactive, or real-time programs
- Very slow, if the algorithm
  - processes data in a graph model
  - contains non-trivial dependences among operations
  - walks through multiple iterations
  - follows a complex data/control flow
- Not programmable, if the algorithm
  - invokes recursion of Map and Reduce steps
  - takes unpredictable execution time

Requires an external "glue" language

Computer Architecture of WSC

- **WSC often use a hierarchy of networks for interconnection**
- **Each rack holds dozens of servers connected to a rack switch**
- **Rack switches are uplinked to switch higher in hierarchy**
  - **Uplink has 48 / n times lower bandwidth, where n = # of uplink ports**
    - **"Oversubscription"**
  - **Goal is to maximize locality of communication relative to the rack**

Storage

Storage options:

- **Use disks inside the servers, or**
- **Network attached storage through Infiniband**

- **WSCs generally rely on local disks**
- **Google File System (GFS) uses local disks and maintains at least three replicas**

Array Switch

Switch that connects an array of racks

- **Array switch should have 10 X the bisection bandwidth of rack switch**
- **Cost of *n*-port switch grows as n^2**
- **Often utilize content addressible memory chips and FPGAs**


NORMAL

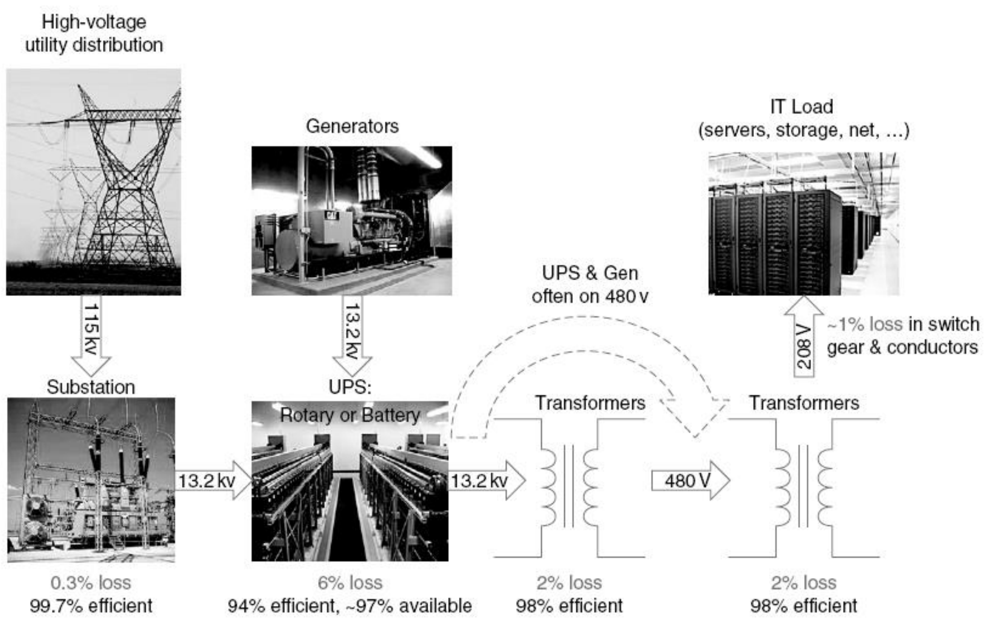
WSC Memory Hierarchy

Servers can access DRAM and disks on other servers using a NUMA-style interface

	Local	Rack	Array
DRAM latency (microseconds)	0.1	100	300
Disk latency (microseconds)	10,000	11,000	12,000
DRAM bandwidth (MB/sec)	20,000	100	10
Disk bandwidth (MB/sec)	200	100	10
DRAM capacity (GB)	16	1,040	31,200
Disk capacity (GB)	2000	160,000	4,800,000

Infrastructure and Costs of WSC

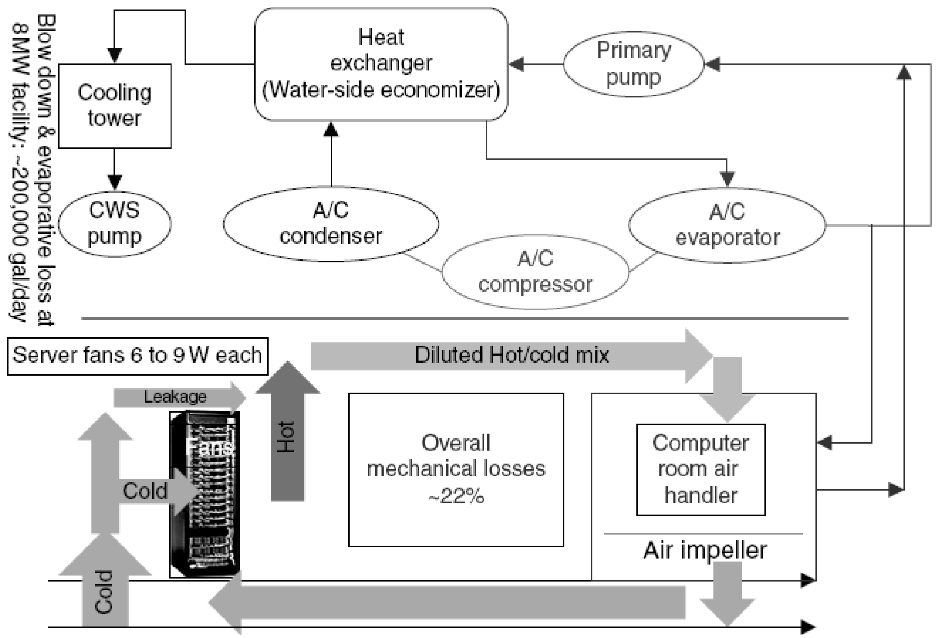
- **Location of WSC**
  - **Proximity to Internet backbones, electricity cost, property tax rates, low risk from earthquakes, floods, and hurricanes**
- **Power distribution**



Infrastructure and Costs of WSC

Cooling

- **Air conditioning used to cool server room**
- **64 F – 71 F**
  - **Keep temperature higher (closer to 71 F)**
- **Cooling towers can also be used**
  - **Minimum temperature is "wet bulb temperature"**



Infrastructure and Costs of WSC

- **Cooling system also uses water (evaporation and spills)**
  - **E.g. 70,000 to 200,000 gallons per day for an 8 MW facility**

- **Power cost breakdown:**
  - **Chillers: 30-50% of the power used by the IT equipment**
  - **Air conditioning: 10-20% of the IT power, mostly due to fans**

- **How many servers can a WSC support?**
  - **Each server:**
    - **"Nameplate power rating" gives maximum power consumption**
    - **To get actual, measure power under actual workloads**
  - **Oversubscribe cumulative server power by 40%, but monitor power closely**

Measuring Efficiency of a WSC

- **Power Utilization Effectiveness (PUE)**
  - **= Total facility power / IT equipment power**
  - **Median PUE on 2006 study was 1.69**

- **Performance**
  - **Latency is important metric because it is seen by users**
  - **Bing study: users will use search less as response time increases**
  - **Service Level Objectives (SLOs)/Service Level Agreements (SLAs)**
    - **E.g. 99% of requests be below 100 ms**

Cost of a WSC

- **Capital expenditures (CAPEX)**
  - **Cost to build a WSC**

- **Operational expenditures (OPEX)**
  - **Cost to operate a WSC**

Cloud Computing

- **WSCs offer economies of scale**
  - **5.7 times reduction in storage costs**
  - **7.1 times reduction in administrative costs**
  - **7.3 times reduction in networking costs**
  - **This has given rise to cloud services such as Amazon Web Services**
    - **“Utility Computing”**
    - **Based on using open source virtual machine and operating system software**