Request-Level Parallelism

Warehouse-Scale Computers, Large-Scale Computers

- Warehouse-Scale Computers
- Programming models
- Physical infrastructure
- Large-scale computers
- Loosely coupled clusters
- High-performance clusters

Warehouse-Scale Computers

- Provides Internet services
  - Search, social networking, online maps, video sharing, online shopping, e-mail, cloud computing, and so on
- Differences with HPC “clusters”:
  - Clusters have higher-performance processors and network.
  - Clusters emphasize thread-level parallelism; WSCs emphasize request-level parallelism.
- Differences with data centers:
  - Data centers consolidate different machines and software into one location.
  - Data centers emphasize virtual machines and hardware heterogeneity in order to serve varied customers.

Warehouse-Scale Computers (cont.)

- Data center
  - Collection of 10,000 to 100,000 servers
  - Networks connecting them together
- Single gigantic machine
- Very large applications (Internet service): search, e-mail, video sharing, social networking
- Very high availability

Unique to WSCs

- Ample parallelism
  - Request-level parallelism, for example, Web search
  - Data-level parallelism, for example, image classifier training
- Scale and its opportunities/problems
  - Scale of economy: low per-unit cost
  - Cloud computing: rent computing power with low costs
  - High number of failures
  - For example, four disks/server, annual failure rate: 4 percent → WSC of 50,000 servers: one disk fail/hour
- Operation cost count
  - Longer life time(>10 years)
  - Cost of equipment purchases << cost of ownership

WSC Architecture and Storage

- 1U server: 8 cores, 32 GB DRAM, 4×1 TB disk
  - DRAM: 16 GB, 100 ns, 20 GB/s
  - Disk: 2 TB, 10 ms, 200 MB/s
- Rack: 50 to 100 servers, Ethernet switch
  - 10$/1 Gbps/server
  - DRAM: 1 TB, 300 us, 100 MB/s
  - Disk: 160 TB, 11 ms, 100 MB/s
- Array (aka cluster): 16 to 32 racks
  - 10X bandwidth, 100X cost
  - DRAM: 30 TB, 500 us, 10 MB/s
  - Disk: 4.80 PB, 12 ms, 10 MB/s
- Lower latency to DRAM in another server than local disk
- Higher bandwidth to local disk than to DRAM in another server

Impact on WSC software

- Latency, bandwidth ➔ performance
  - Independent dataset within an array
  - Locality of access within sever or rack
- High failure rate ➔ reliability, availability
  - Preventing failures is expensive.
  - Cope with failures gracefully.
- Varying workloads ➔ availability
  - Scale up and down gracefully
- More challenging than software for single computers!

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

Design Factors for WSC (cont.)

- 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

ENGINEERING@SYRACUSE

Programming Models for Warehouse-Scale Computers

Programing Models/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

Programming Models/Workloads (cont.)

- 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));

Programming Models

- 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

What Is MapReduce?

- Simple data-parallel programming model and implementation for processing large dataset
  - Users specify the computation in terms of:
    - A map function, and
    - A reduce function
- Underlying runtime system
  - Automatically parallelizes the computation across large scale clusters of machines
  - Handles machine failure
  - Schedules intermachine communication to make efficient use of the networks

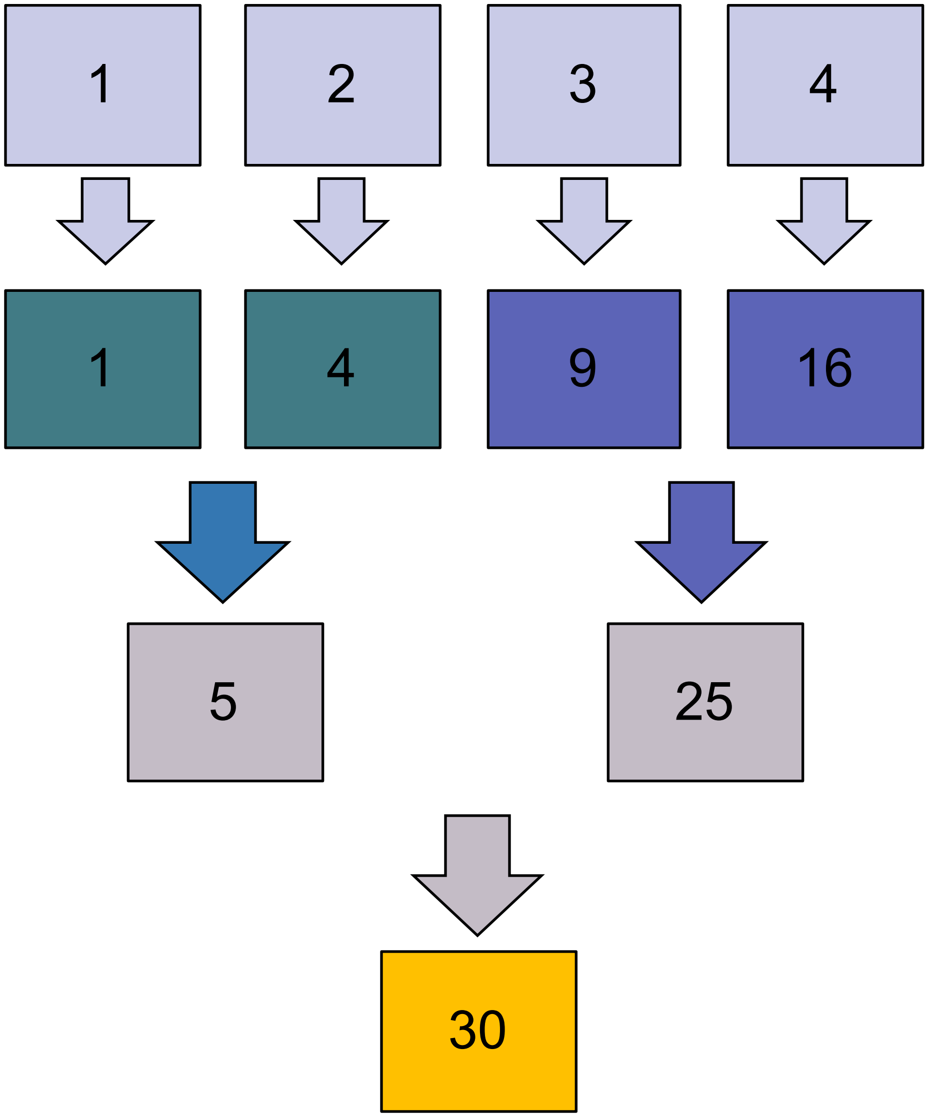
What Is MapReduce Used For?

- At Google:
  - Index construction for Google Search
  - Article clustering for Google News
  - Statistical machine translation
  - For computing multilayered street maps
- At Yahoo!:
  - “Web map” powering Yahoo! Search
  - Spam detection for Yahoo! Mail
- At Facebook:
  - Data mining
  - Ad optimization
  - Spam detection

Map and Reduce Function

- Calculate: $\sum_{n=1}^{4} n^2$

```
l = [1, 2, 3, 4]
def square(x):
  return x * x
def sum(x, y):
  return x + y
reduce(sum,
  map(square, l))
```



MapReduce Programming Model

- ***Map***: `(in_key, in_value) → list(interm_key, interm_val)`

```
map(in_key, in_val):
  // DO WORK HERE
  emit(interm_key,interm_val)
```

  - Slices data into “shards” or “splits” and distribute to workers
  - Computes set of intermediate key-value pairs

- ***Reduce***: `(interm_key, list(interm_value)) → list(out_value)`

```
reduce(interm_key, list(interm_val)):
  // DO WORK HERE
  emit(out_key, out_val)
```

  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

MapReduce Word Count Example

- ***Map*** phase: (doc name, doc contents) → list(word, count)

```
// "I do I learn"" → [("I",1),("do",1),("I",1),("learn",1)]
map(key, value):
  for each word w in value:
    emit(w, 1)
```

- ***Reduce*** phase: (word, list(count)) → (word, count_sum)

```
// ("I", [1,1]) → ("I",2)
reduce(key, values):
  result = 0
  for each v in values:
    result += v
  emit(key, result)
```

ENGINEERING@SYRACUSE

Architecture Models for WSC

Computer Architecture of WSC

- WSC often use a hierarchy of networks for interconnection.
- Each 19" rack holds 48 1U servers connected to a rack switch.
- Rack switches are uplinked to switch higher in hierarchy.
  - Uplink has 48/n times lower bandwidth, where n = number 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 10X the bisection bandwidth of rack switch.
  - Cost of *n*-port switch grows as n^2.
  - Often utilize content addressable memory chips and FPGAs.

WSC Memory Hierarchy

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

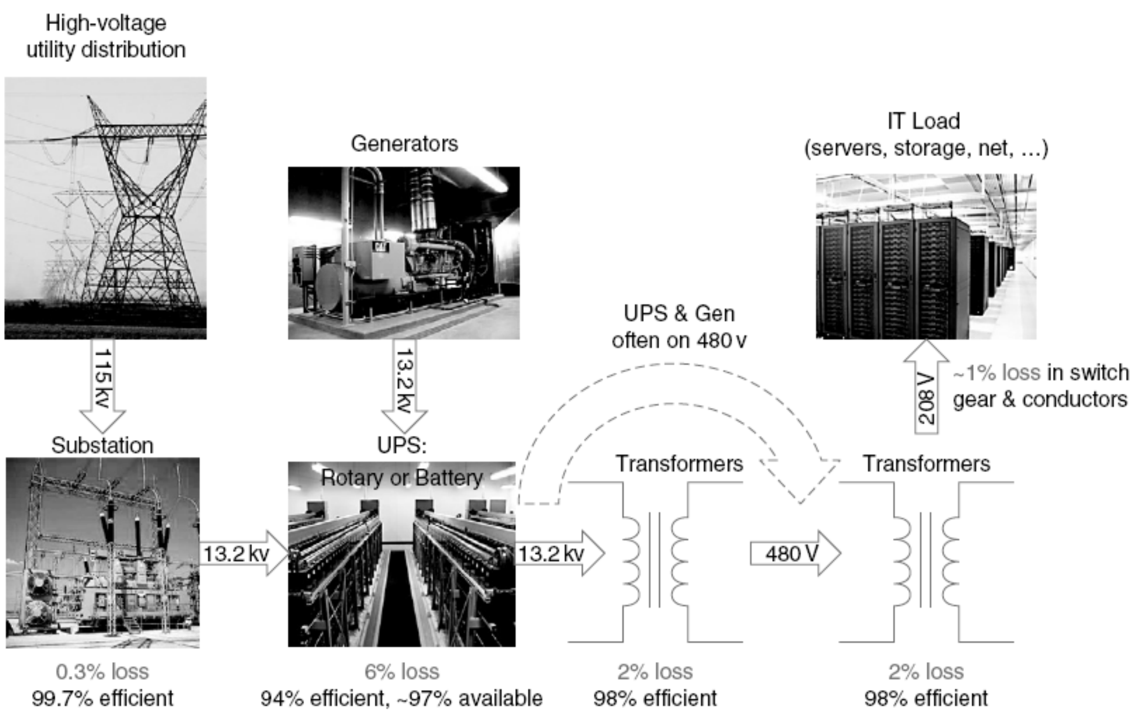
	Local	Rack	Array
DRAM latency (microseconds)	0.1	100	300
Disk latency (microseconds)	10,000	11,000	12,000
DRAM bandwidth (MB/sec)	20,000	100	10
Disk bandwidth (MB/sec)	200	100	10
DRAM capacity (GB)	16	1,040	31,200
Disk capacity (GB)	2000	160,000	4,800,000

ENGINEERING@SYRACUSE

Physical Infrastructure

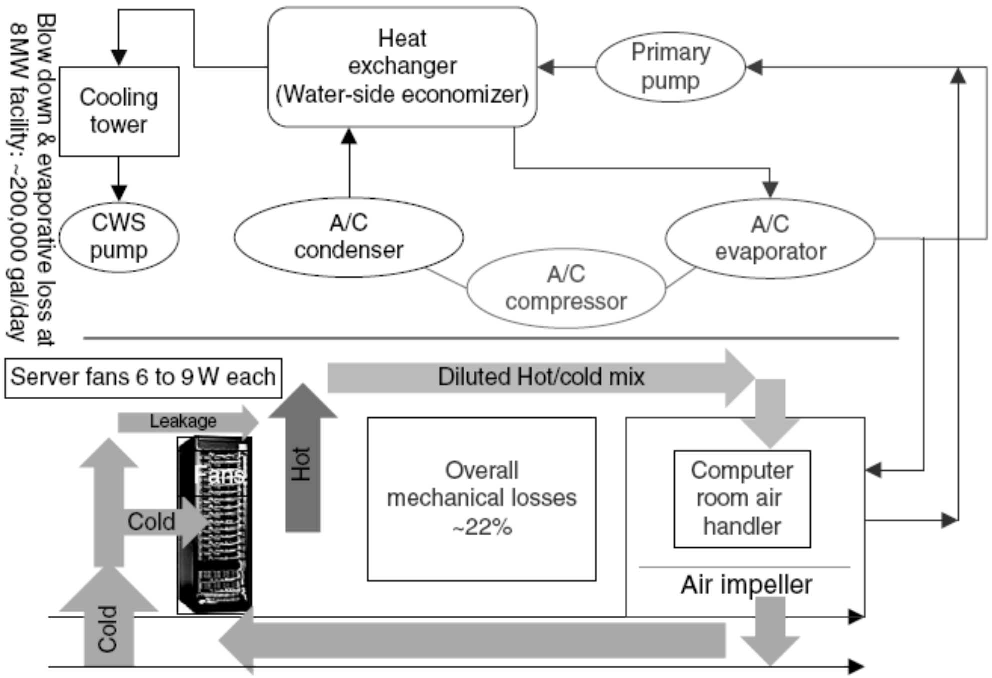
Infrastructure and Costs of WSC

- 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 (cont.)

- Cooling system also uses water (evaporation and spills).
  - For example, 70,000 to 200,000 gallons per day for an 8 MW facility
- Power cost breakdown:
  - Chillers: 30 to 50 percent of the power used by the IT equipment
  - Air conditioning: 10 to 20 percent 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 percent, but monitor power closely.

Measuring Efficiency of a WSC

- Power utilization effectiveness (PEU)
  - = 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).
    - For example, 99 perecnt 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 that cannot be achieved with a datacenter.
  - 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

ENGINEERING@SYRACUSE

Loosely Coupled Clusters

Loosely Coupled Clusters

- Network of independent computers
  - Each has private memory and OS.
  - Connected using I/O system.
    - For example, Ethernet/switch, Internet
- Suitable for applications with independent tasks
  - Web servers, databases, simulations
- High availability, scalable, affordable
- Problems
  - Administration cost (prefer virtual machines)
  - Low interconnect bandwidth
    - C.f., processor/memory bandwidth on an SMP

Sum Reduction

- Sum 100,000 on 100 processors.
- First distribute 100 numbers to each, then do partial sums.

```
sum = 0;
for (i = 0; i<1000; i = i + 1)
  sum = sum + AN[i];
```

- Reduction:
  - Half the processors send, other half receive and add.
  - A quarter send, quarter receive and add, and so on.

Sum Reduction (cont.)

- Given send() and receive() operations:

```
limit = 100; half = 100;  /* 100 processors */
repeat
  half = (half+1)/2;      /* send vs. receive dividing line */
  if (Pn >= half && Pn < limit)
    send(Pn - half, sum);
  if (Pn < (limit/2))
    sum = sum + receive();
  limit = half;           /* upper limit of senders */
until (half == 1);        /* exit with final sum */
```

- Send/receive also provide synchronization.
- Assumes send/receive take similar time to addition.

Grid Computing

- Separate computers interconnected by long-haul networks
  - For example, Internet connections.
  - Work units farmed out, results sent back.
- Can make use of idle time on PCs
  - For example, SETI@home, World Community Grid

Google WSC

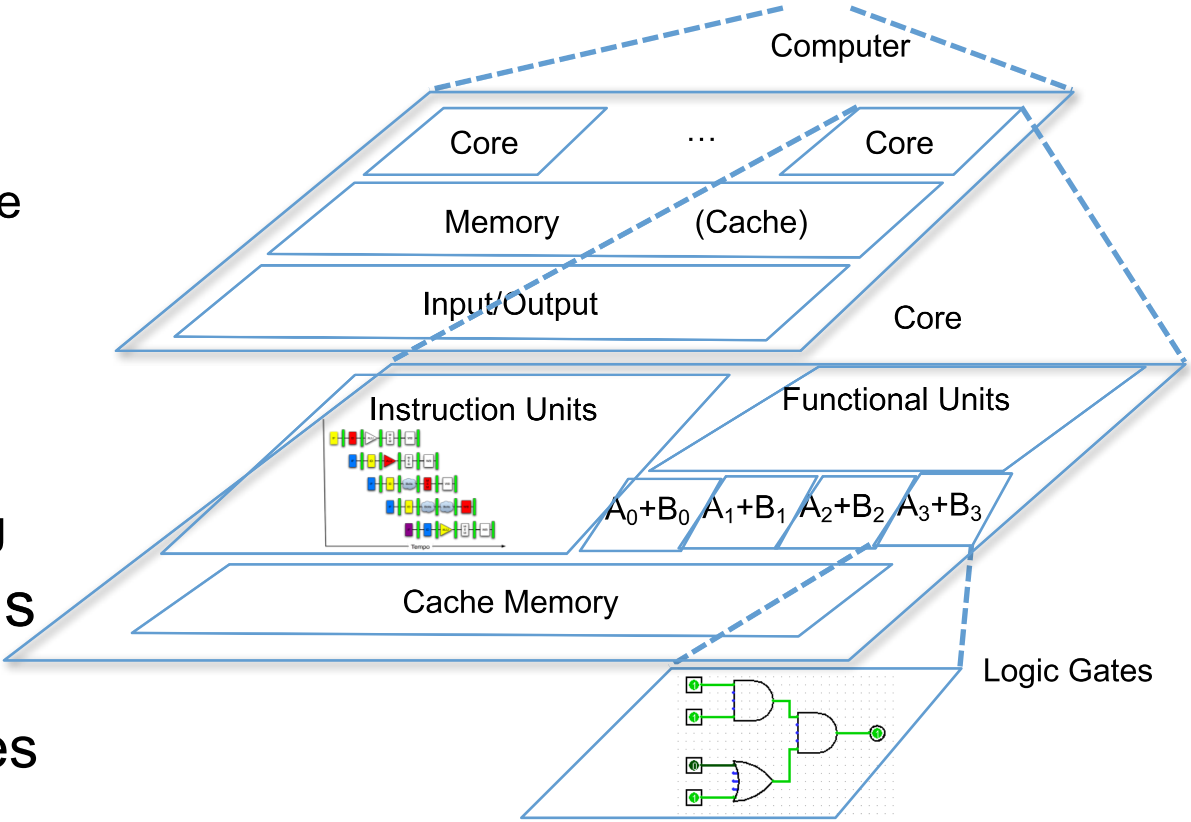
- https://cloud.google.com/container-engine/
- Search cost per day (per person) same as running a 60-watt bulb for three hours

ENGINEERING@SYRACUSE

Scalable Parallelism

Harnessing Parallelism

- Parallel requests
  - Assigned to computer
  - For example, search
- Parallel threads
  - Assigned to core
  - For example, lookup, ads
- Parallel instructions
  - >one instruction @ one time
  - For example, five pipelined instructions
- Parallel data
  - >one data item @ one time
  - For example, deep learning
- Hardware descriptions
  - All gates @ one time
- Programming languages



Data-Level Parallelism(DLP)

- SIMD
  - Supports data-level parallelism in a single machine
  - Additional instructions and hardware

  For example, matrix multiplication in memory

- DLP on WSC
  - Supports data-level parallelism across multiple machines
  - MapReduce and scalable file systems

  For example, training CNNs with images across multiple disks

Summary

- Warehouse-scale computers
  - New class of computers
  - Scalability, energy efficiency, high failure rate
- Request-level parallelism

  For example, Web search
- Data-level parallelism on a large dataset
  - MapReduce
  - Hadoop, Spark

Request-Level Parallelism (RLP)

- Hundreds of thousands of requests per sec
  - Not your laptop or cellphone, but popular Internet services like web search, social networking, and so on.
  - Such requests are largely independent.
    - Often involve read-mostly databases
    - Rarely involve read-write sharing or synchronization across requests
- Computation easily partitioned across different requests and even within a request

ENGINEERING@SYRACUSE

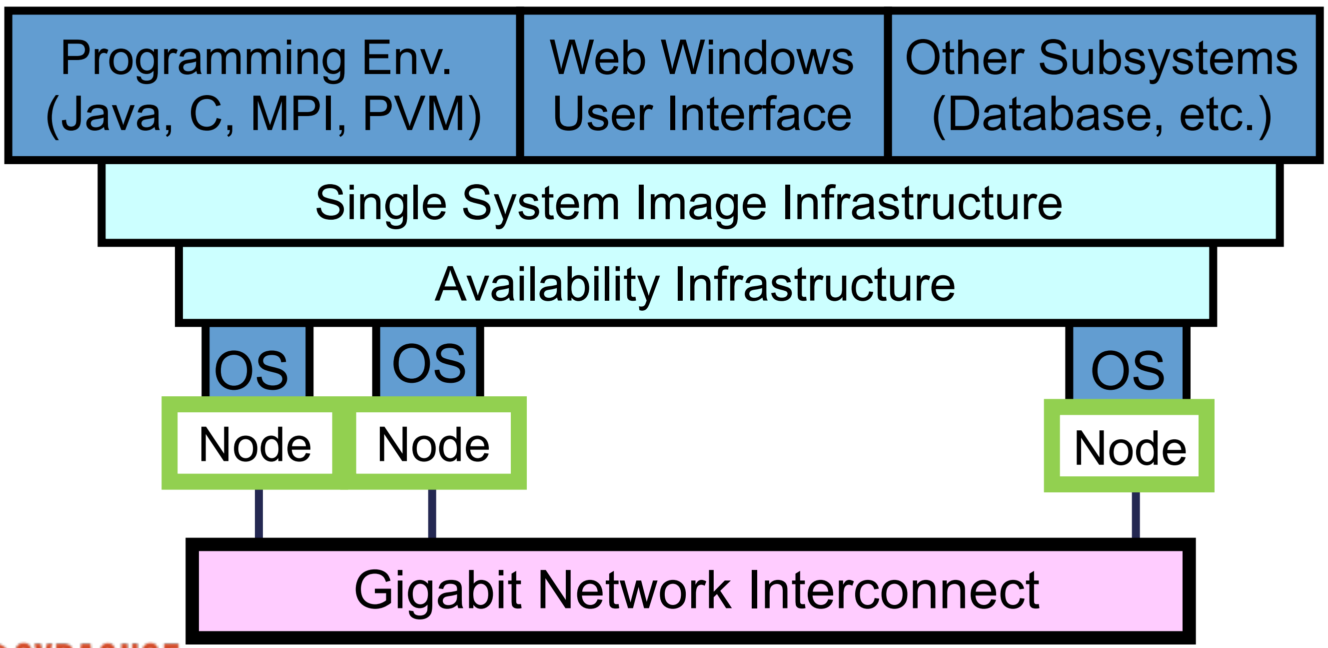
High-Performance Clusters

High-Performance Clusters

- Mainframe
- Supercomputer
- Minicomputer
- Workstation
- PC
- Clusters

Cluster Architecture

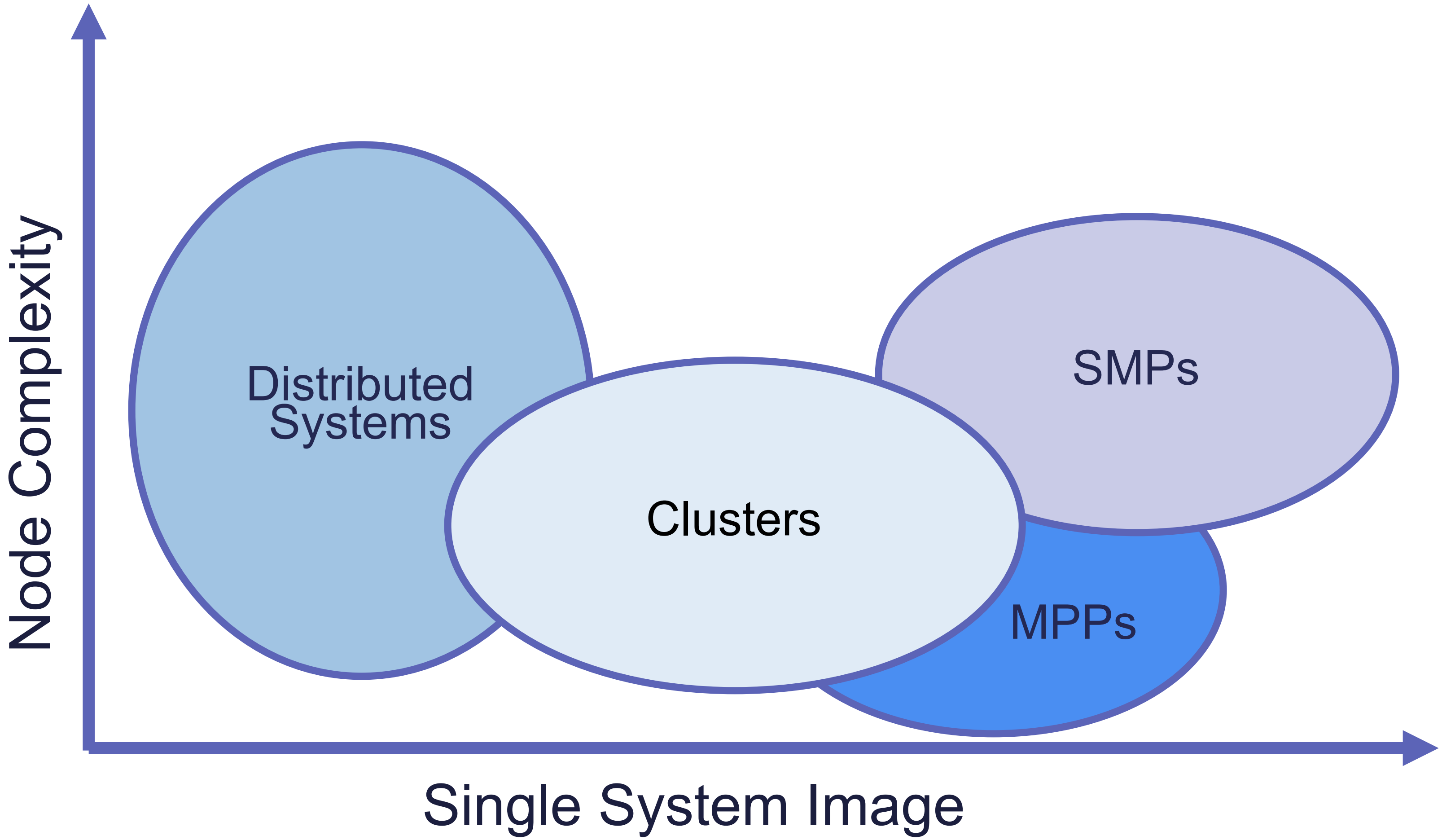
- Collection of independent computer systems working together as if a single system
- Coupled through a scalable, high bandwidth, low latency interconnect



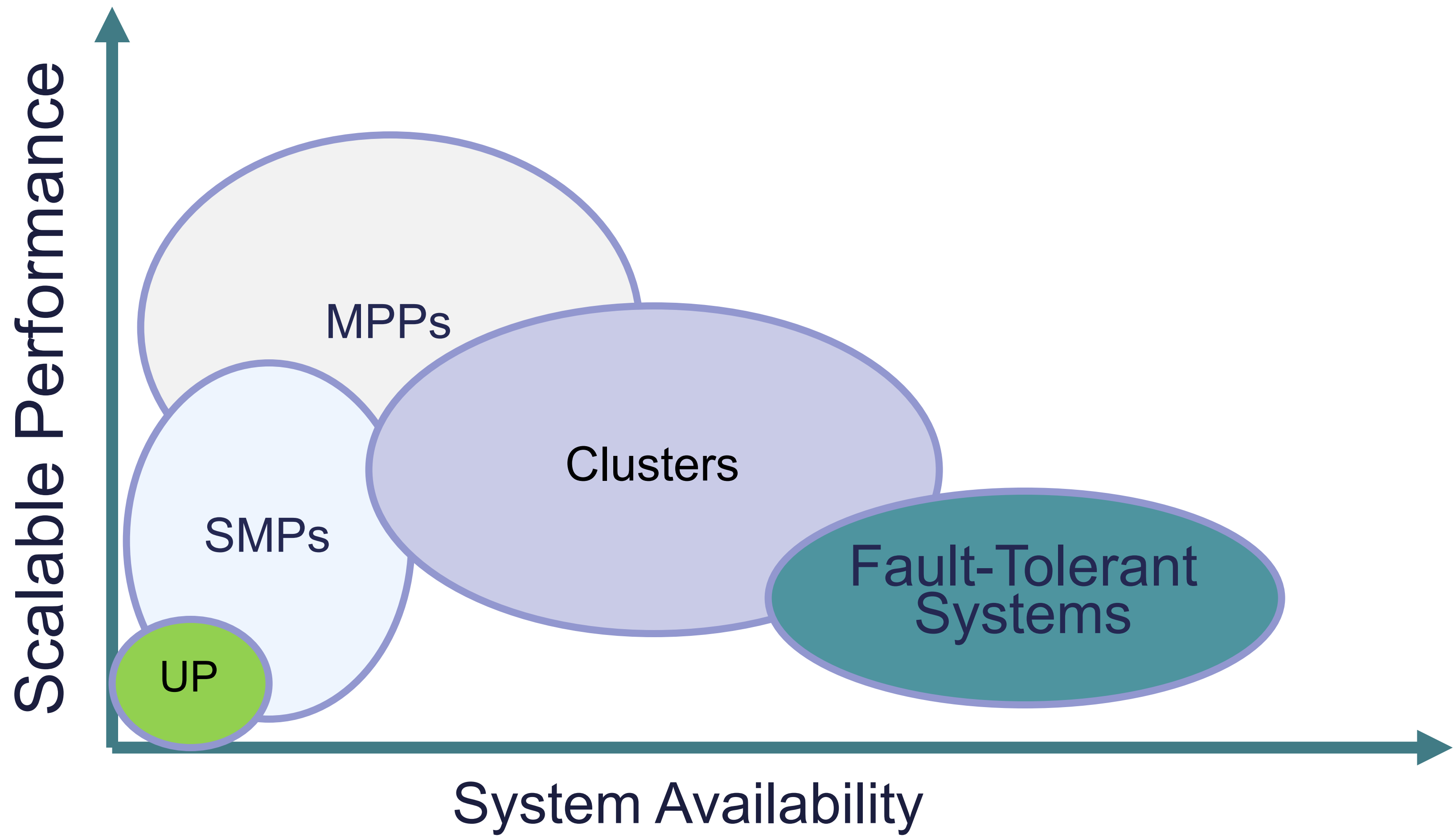
Advantages

- Commodity parts.
- Intelligent network interface.
- Scalability.
- Independent failure.
- Fast/scalable communication.
- Each node is a system.
- Questions:
  - Performance? Availability? Cost? Capacity?

Single System Image



Performance vs. Availability



MIMD Computers

Multiprocessors
(Single address space with shared memory)

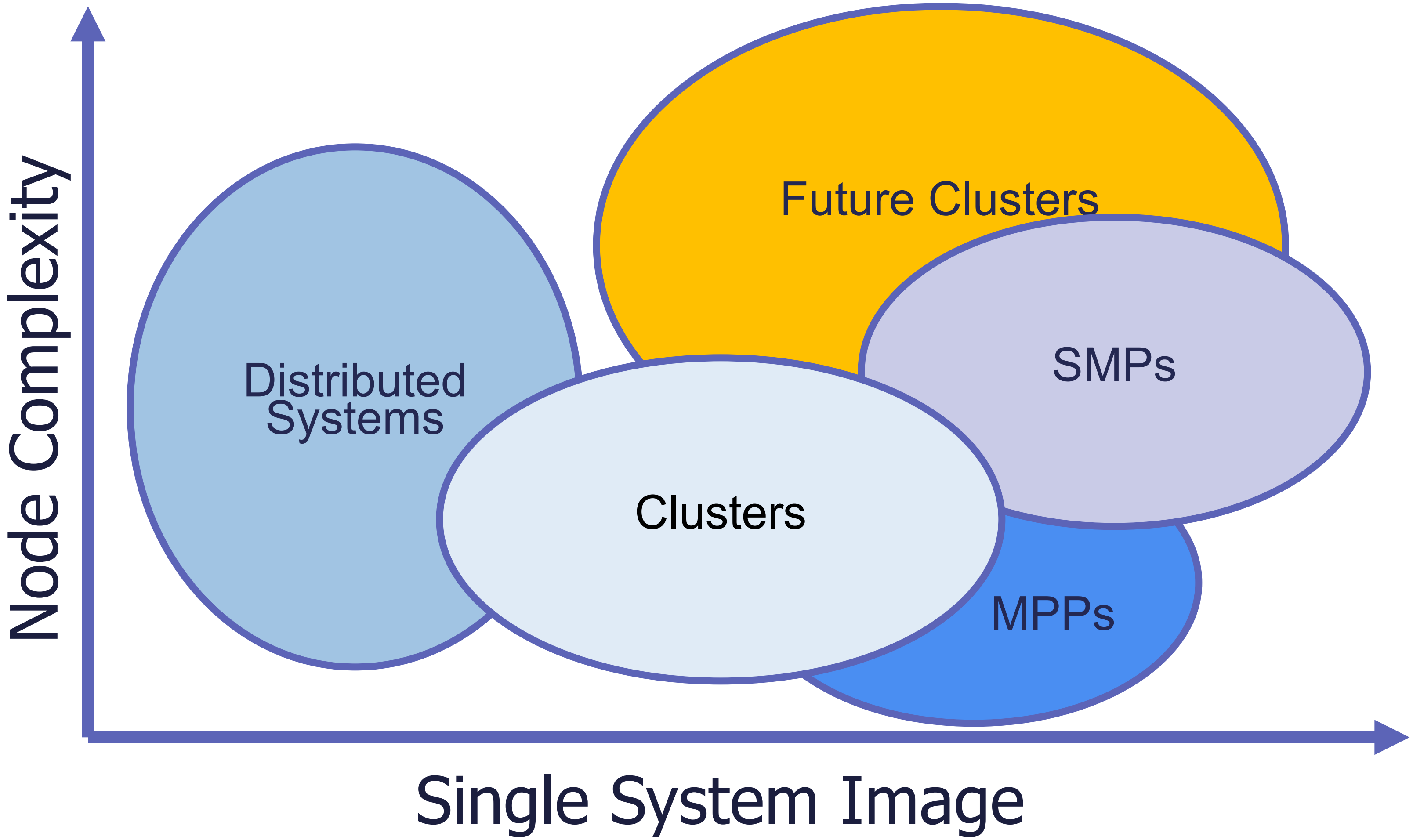
Multicomputers
(Multiple address spaces with nonshared memory)

- MIMD
  - Multiprocessors
    - UMA
      - PVP (Cray T90)
      - SMP (Intel SHV, Sun, SGI, IBM)
    - NUMA
      - COMA (KSR-1)
      - CC- NUMA (Stanford Dash, SGI Origin, Sequent NUMA, HP Exemplar)
      - NCC-NUMA (Cray T3E, etc.)
      - DSM or SC-NUMA (TreadMarks, Wind Tunnel, Shrimp)
  - Multicomputers
    - Cluster (IBM SP2, TruCluster, Solaris MC, Tandem Hymalaya, Wolfpack, NOW, PearlCluster)
    - MPP (Intel Option Red, IBM Blue Pacific, SGI/Cray Blue Mountain)

Comparison of Systems

	MPP	SMP CC-NUMA	Cluster	Distributed System
# of Nodes	O(100)–O(1000)	O(10)–O(100)	O(100) or less	O(10)–O(1000)
Node complexity	Fine/medium	Medium/coarse	Medium	Wide range
Internode communication	Message passing	Centralized or shared memory	Message passing	Shared files, message passing
Job scheduling	Single run queue at host	Single run queue mostly	Multiple queues coordinated	Independent multiple queues
SSI support	Partially	Always for SMP	Desired	No
Node and host OS	N microkernels, one monolithic OS	One monolithic	N (Homogeneous or microkernel)	N (Heterogenous)
Address space	Multiple	Single	Multiple	Multiple

Trends


Node Complexity
Future Clusters
Distributed Systems
SMPs
Clusters
MPPs
Single System Image

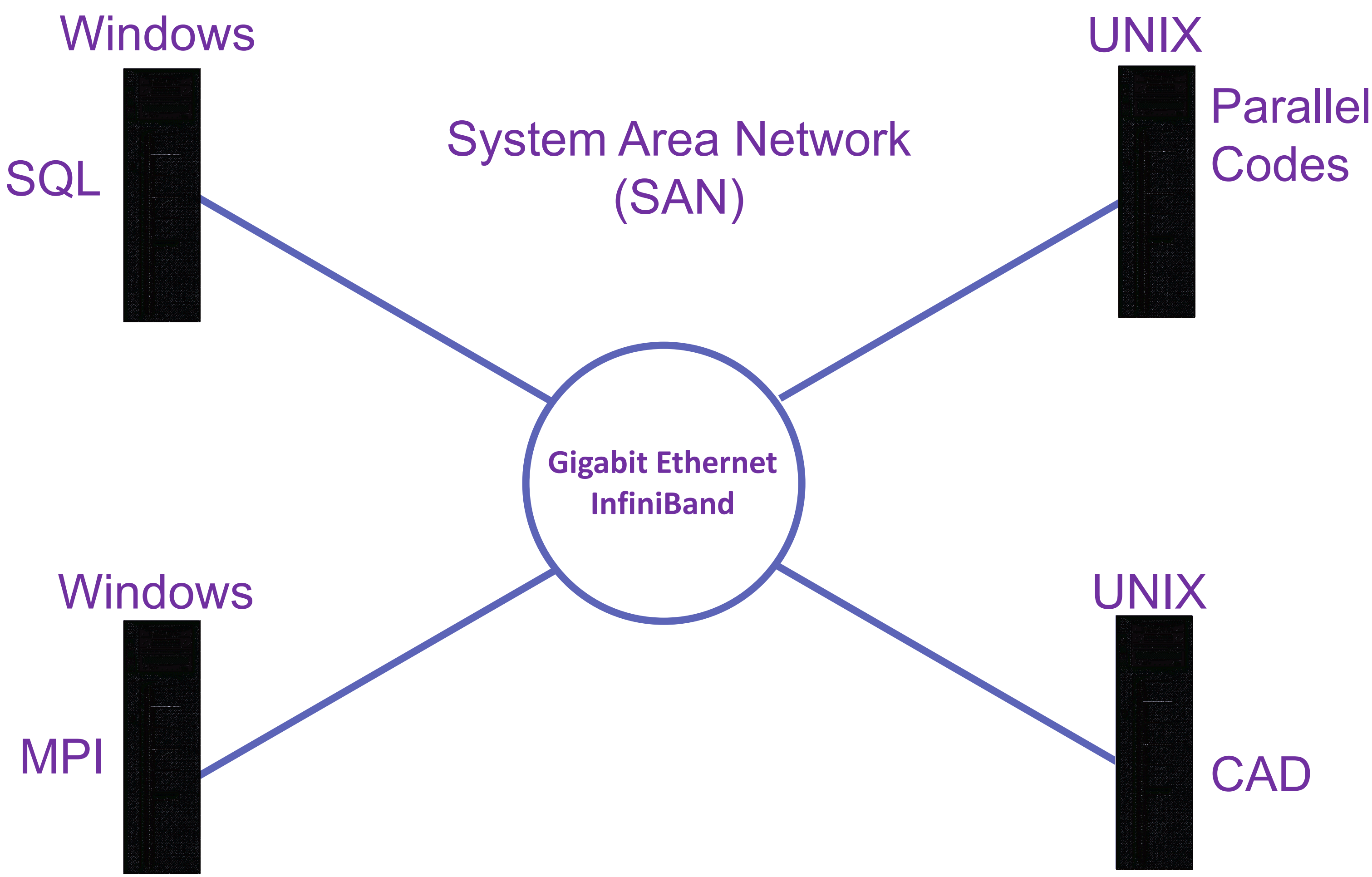
Demands for Clusters

- Increasing demand for clusters
  - Internet services
  - Huge demand for scalable, available, dedicated Internet servers
- Distributed multimedia processing in e-commerce
- Dedicated digital libraries for distance education
- Bioinformatics for health care, telemedicine
- Economic crisis management
- Collaborative designs

HPC Summary

- Performance → generality
- From technology “shift” to technology “trend”
- Cluster communication becoming cheap
  - Gigabit Ethernet
- System area networks becoming commodity
  - Myricom, Compaq ServerNet, SGI, HAL, Sun
- Improvements in interconnect BW
  - Gigabyte per second and beyond
- Bus connections improving
  - PCI, ePCI, Pentium II cluster slot
- Operating system out of the way
  - VIA

Interconnecting Clusters!



Cluster Solution

Applications	Scalable and available apps
Operating Systems	Cluster APIs
	No common interface
System Area Network	Fast NICs and switches
Platforms	Standard slots/connectors
CPUs and Chipsets	Standard I/O ports

ENGINEERING@SYRACUSE

Concluding Remarks

Quantitative Approach

- New era with growing diversity in:
  - Applications: office, scientific, IoT, smart things
  - Systems: CPU, SoC, small-scale, large-scale
  - And so on
- Skill sets
  - Principles: parallelism, locality, common case, bottlenecks
  - Evaluations: performance, cost, power, reliability
  - Solid interfaces
  - Technology tracking and anticipation

Nucleus of Processors

- Building blocks: computation, communication, storage
- Logic design: gates, flip-flops, latches, clock
- Datapath: register, memory, branch operations
- Implementation: fetch, decode, data, execute, store, next

Multiple Operations at One Time

- Hazards: structural, data, control
- Exception handling
- Implementations: ARM vs. Intel
- I/O: latency, throughput
- Storage: disk, flash drive, RAID
- Busses: parallel, serial
- Control: interruption, polling

Multiple Data at Various Places

- Memory hierarchy: cache, main memory, TLB, VM, storage
- Optimizations: four questions: where, how, which, what
- Memory technology: SRAM, DRAM, NVRAM
- Virtual memory: management, protection, address translation

Multiple Instructions at One Time

- Dynamic scheduling: out-of-order execution, speculation
- Multiple issue and static scheduling
  - At-compile time, fixed number of operations/instruction, VLIW
- Multiple issue and dynamic scheduling
  - At runtime, variable number of operations/instruction, superscalar
- Loops
  - Unrolling, software pipelining

Multiple Data at One Time

- ILP → TLP → DLP
- Demand and technology
  - Data-intensive applications
  - CPU+GP-GPU
- Vector machines and supercomputers
  - Lots of registers, memory banks
  - Good for vector operations
  - Simple programming
- SIMD extensions for multimedia
- GPU systems
  - Faster/efficient executions

Multiple Threads at One Time

- Multithreading
  - Fine grain, coarse grain, simultaneous multithreading
- Modeling performance
  - Roofline diagram
- Comparing multicore systems

Multiple Computers at One Time

- Parallel computing: science, engineering, commercial
- Parallel architecture: convergence, scalability
- Abstraction models: programming, communication
- Centralized memory systems: small scale, snoopy cache coherence
- Distributed memory systems: large scale, directory-based coherence
- Synchronization: atomic, spins, load locked store conditional, relaxed models

Multiple Communications

- Devices: components, computers, systems
- Domains: OCN, SAN, LAN, WAN
- Metrics: latency, effective BW
- Media: shared vs. switched, low cost vs. scalable
- Topologies: bus, k-ary n-cube, multistage
- Comparisons: bisection BW, links, switches
- Routing: deterministic/adaptive
- Flow: store and forward, wormhole

Multiple Requests at One Time

- Warehouse-scale computers
- High-performance clusters
- Cloud computing
- Grid computing
- Data centers

ENGINEERING@SYRACUSE