# Distributed System Models and Enabling Technologies

**Ioan Raicu**
Computer Science Department
Illinois Institute of Technology

CS 553: Cloud Computing
February 7th, 2018

# Logistics

- Read chapter 1 from textbook
- PA1 is coming…

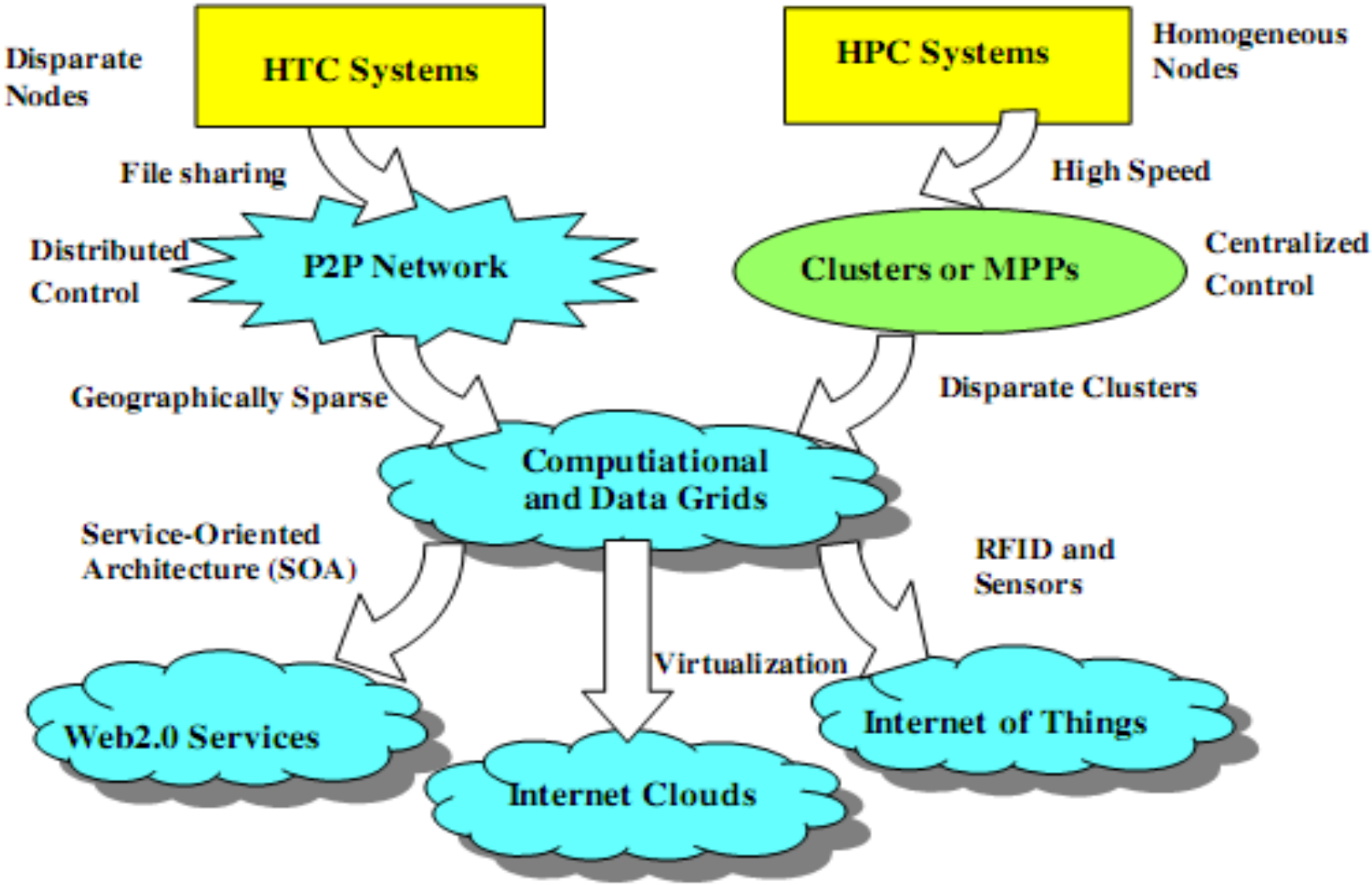
# Programming Assignment #1

- Write your own benchmarks:
  - C, C++, Java (still working on final languages allowed)
  - PThreads, Sockets
- Use existing benchmarks:
  - Linpack, Stream, IOZone, IPerf, Ping (still finalizing list)
- Measure various sub-systems
  - CPU, Memory, Disk, Network
- Vary parameters
  - Concurrency, block size, protocols, access patterns, etc
- **No GUI interfaces and automation everywhere**

# Programming Assignment #1

- CPU:
  - Datatype (Double, Integer), concurrency (1, 2, 4 threads)
- Memory:
  - Throughput: Operations (read+write sequential, read+write random), block size (1KB, 1MB), concurrency (1, 2, 4 threads)
  - Latency: Operations (read+write random), block size (1B), concurrency (1, 2, 4 threads)
- Disk:
  - Throughput: Operations (read sequential, read random, write sequential, write random), block size (1KB, 1MB), concurrency (1, 2, 4 threads)
  - Latency: Operations (read random, write random), block size (1B), concurrency (1, 2, 4 threads)
- Network:
  - Throughput: Network (loopback, Ethernet), protocol (TCP, UDP), buffer/packet size (1KB, 64KB), concurrency (1, 2, 4 threads)
  - Latency: Network (loopback, Ethernet), protocol (TCP, UDP), buffer/packet size (1B), concurrency (1, 2, 4 threads)

# The Age of Internet Computing



# The Age of Internet Computing

- Computing systems distinctions
  - Centralized Computing
  - Parallel Computing
  - Distributed Computing
  - Cloud Computing
- There exists significant overlap among all these

# The Age of Internet Computing
# Key Design Goals

- *Efficiency*
  - measures utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more related to job throughput, data access, storage, and power efficiency.
- *Dependability*
  - in terms of reliability and self-management from the chip to system and application levels. The purpose is to provide high-throughput service with QoS assurance even under failure conditions.
- *Adaptation*
  - in programming model which can support billions of job requests over massive datasets and virtualized cloud resources under various workload and service models.
- *Flexibility*
  - in application deployment. Distributed systems should be designed to run well in both HPC in science and engineering and business HTC applications.

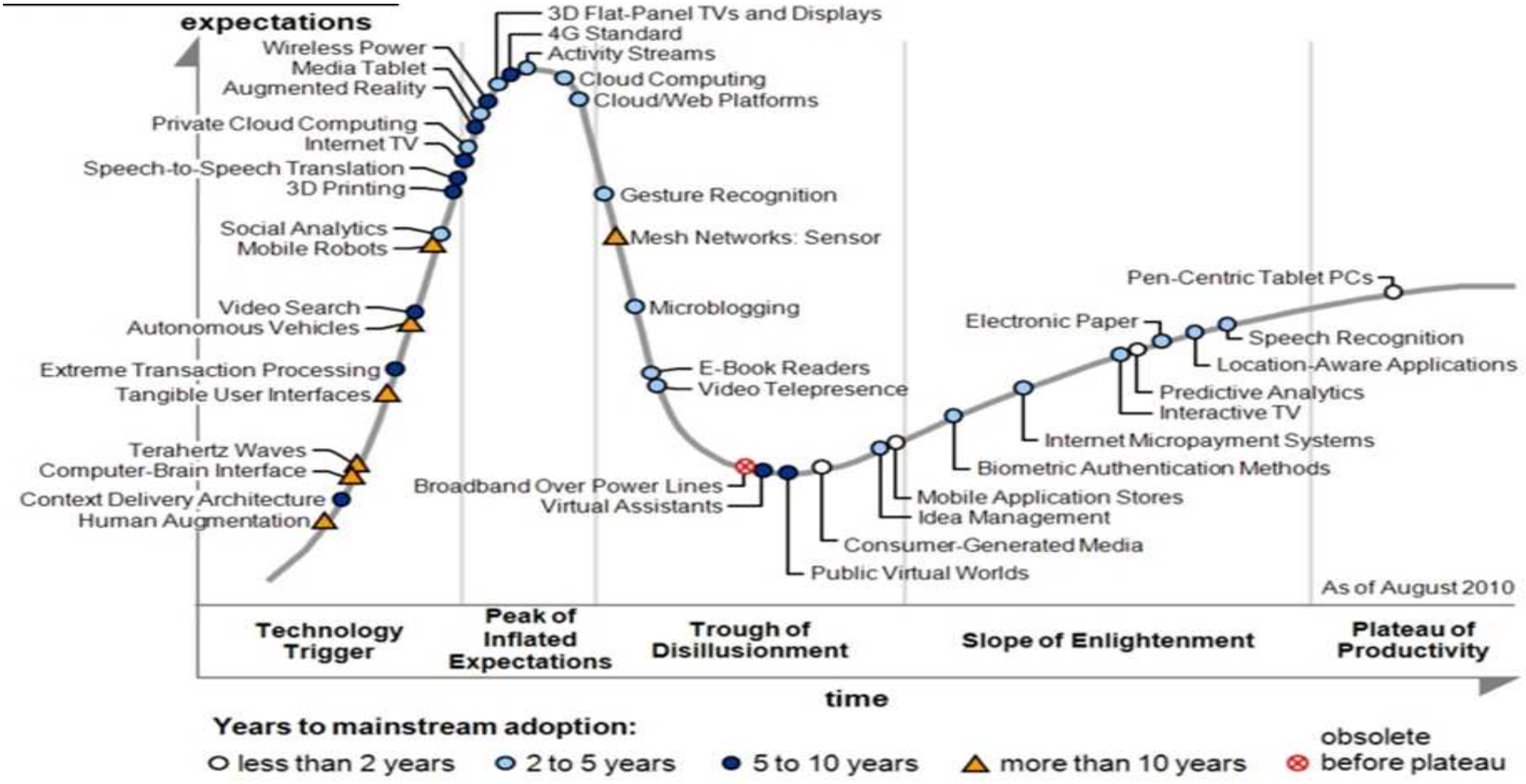
# Parallel Computing Trends and New Paradigms: Degree of Parallelism

- *Bit-level parallelism* (BLP)
  - converts bit-serial processing to word-level processing gradually
- *Instruction-level parallelism* (ILP)
  - execute multiple instructions simultaneously
- *Data-level parallelism* (DLP)
  - Enabled and popularized by SIMD (*single-instruction and multiple-data*) and vector machines using vector or array types of instructions
- *Task-level parallelism* (TLP)
  - Introduced with multicore processors and *chip multiprocessors* (CMP)
- *Job-level parallelism* (JLP)
  - Parallel processing ➔ distributed processing, we will see an increase of computing granularity

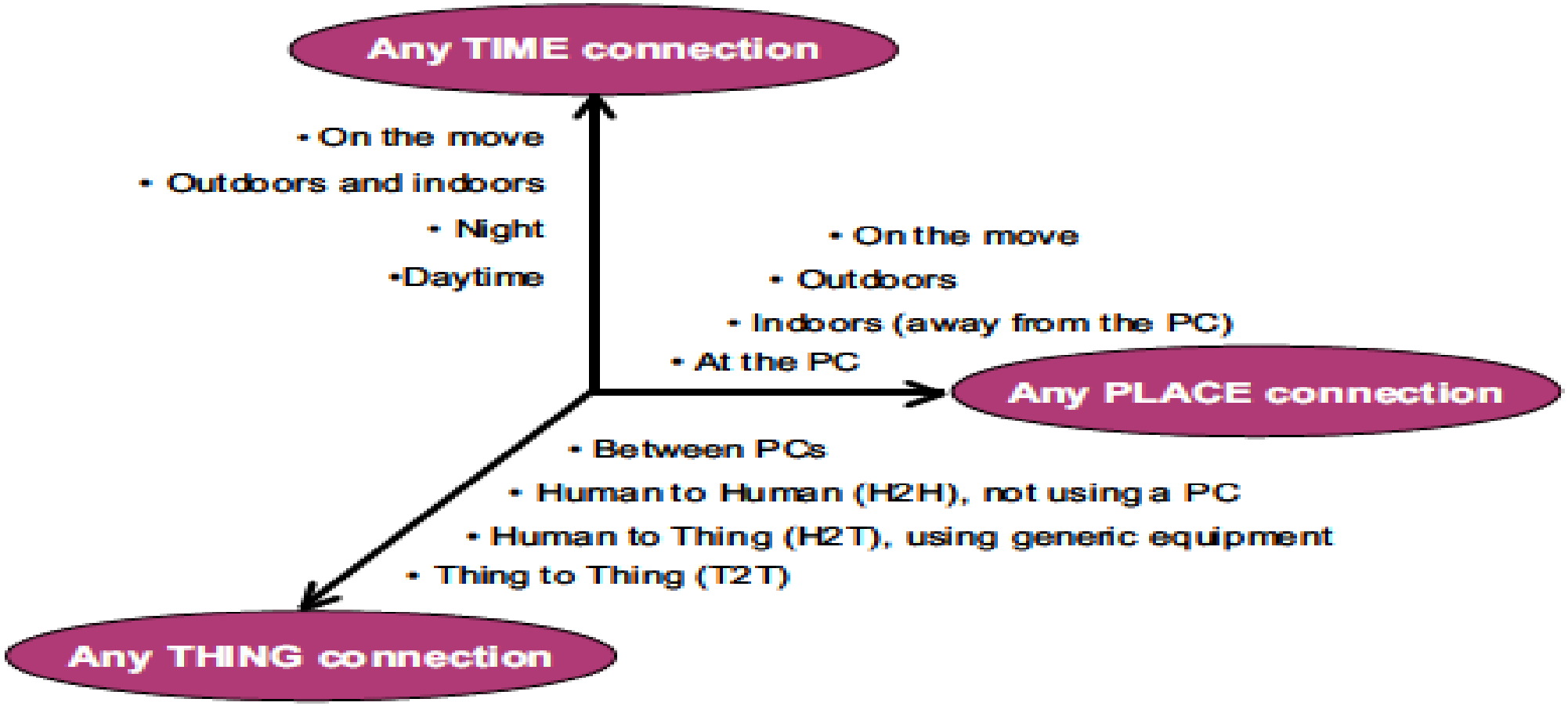
# Parallel Computing Trends and New Paradigms: Applications

| Domain | Specific Applications |
|---|---|
| Science and Engineering | Scientific simulations, genomic analysis, etc. |
| | Earthquake prediction, global warming, weather forecasting, etc. |
| Business, Education, service industry, and Health Care | Telecommunication, content delivery, e-commerce, etc. |
| | Banking, stock exchanges, transaction processing, etc. |
| | Air traffic control , electric power Grids, distance education, etc. |
| | Health care, hospital automation, telemedicine, etc. |
| Internet and Web Services and Government Applications | Internet search, datacenters, decision-make systems, etc. |
| | Traffic monitory , worm containment, cyber security, etc. |
| | Digital government, on-line tax return, social networking, etc. |
| Mission-Critical Applications | Military commend, control, intelligent systems, crisis management, etc. |

# Parallel Computing Trends and New Paradigms: Hype Cycle



# Internet of Things and Cyber-Physical Systems

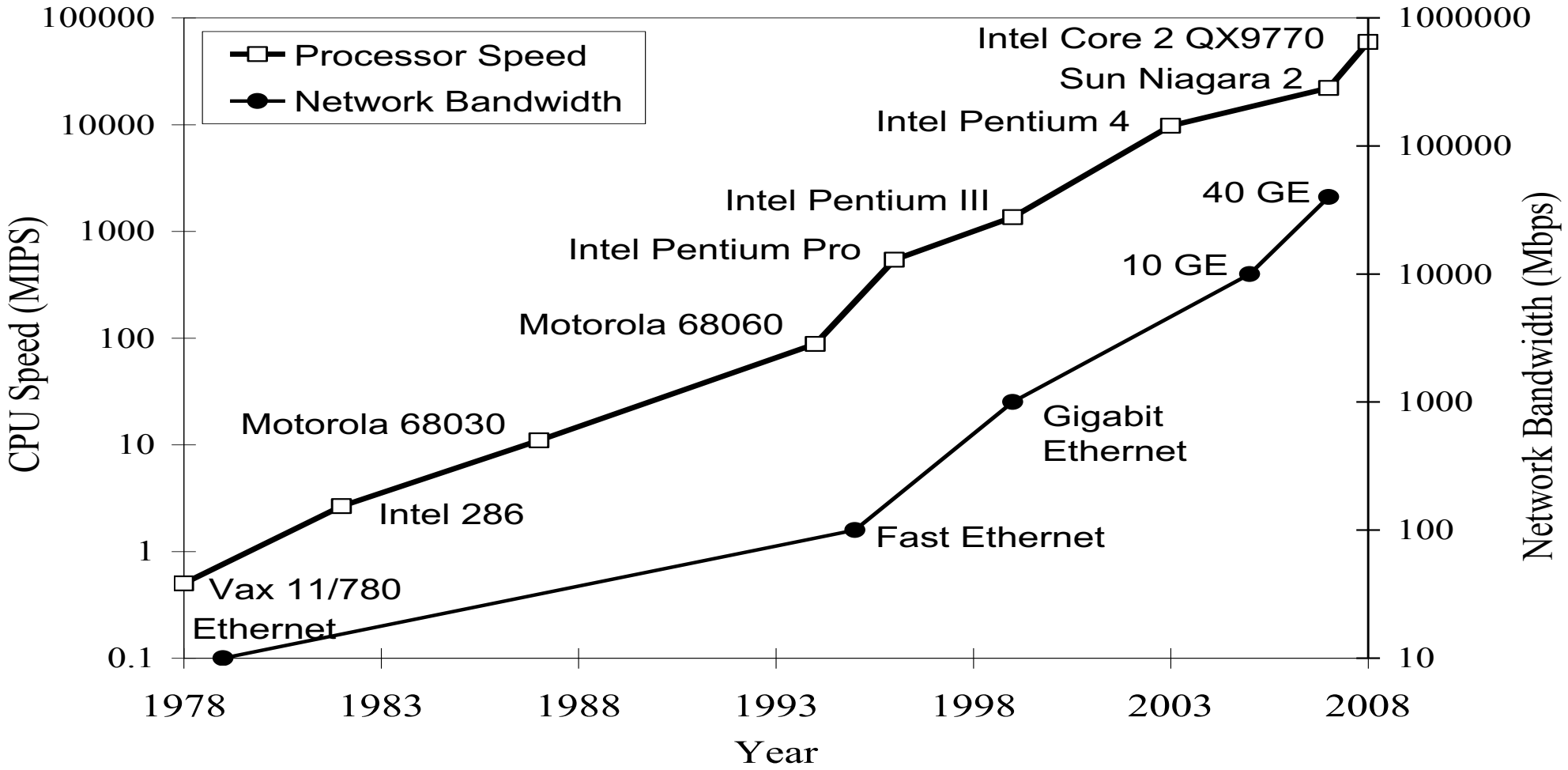


# Internet of Things and Cyber-Physical Systems

- Enabling Technologies
  - GPS: *global positioning system*
  - RFID: *radio-frequency identification*
  - IPv6: Internet Protocol version 6
    - $2^{128}$ IP addresses
- Other interesting numbers
  - $2^{32}$ IPv4 addresses
  - ~$2^{33}$ people on earth
  - ~$2^{47}$ number of cells in the human body
  - ~$2^{59}$ seconds – age of the universe
  - ~$2^{82}$ kg of mass on earth
  - ~$2^{167}$ molecules on earth

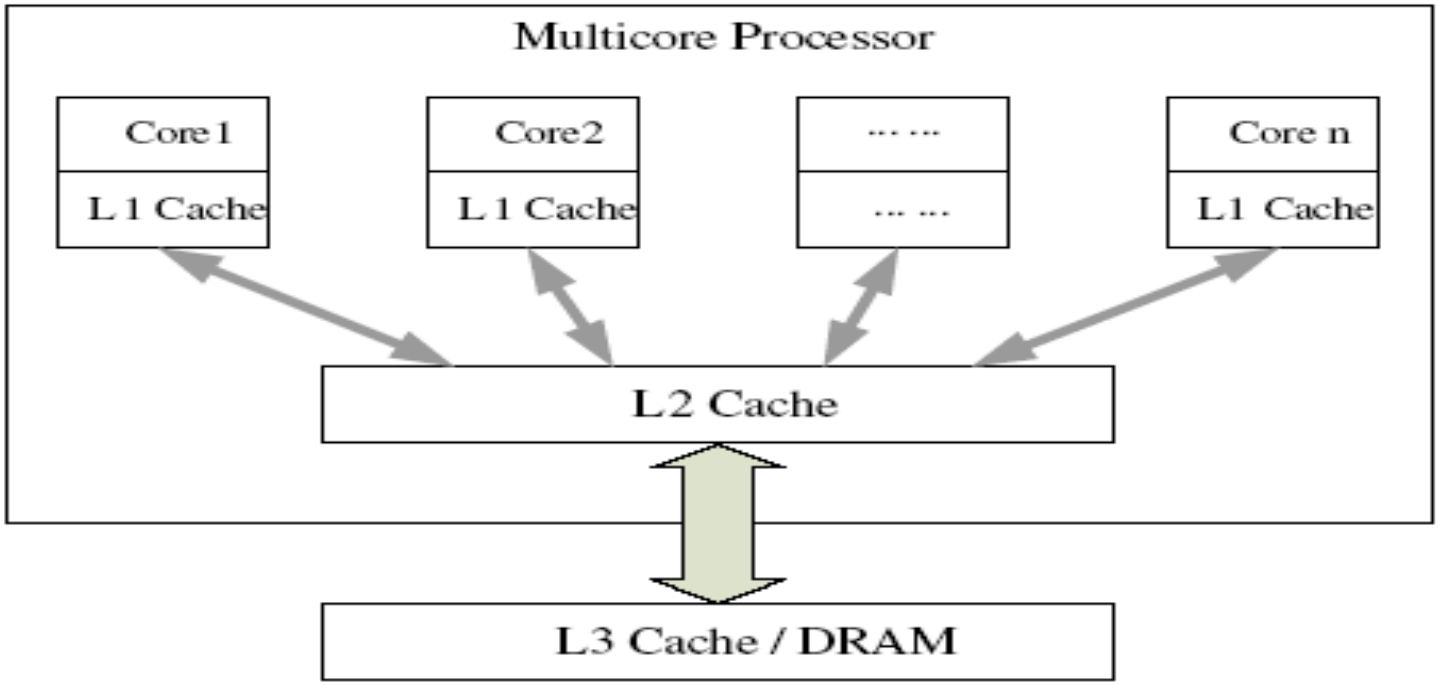
# Logistics

# Multicore CPUs and Multithreading Technologies



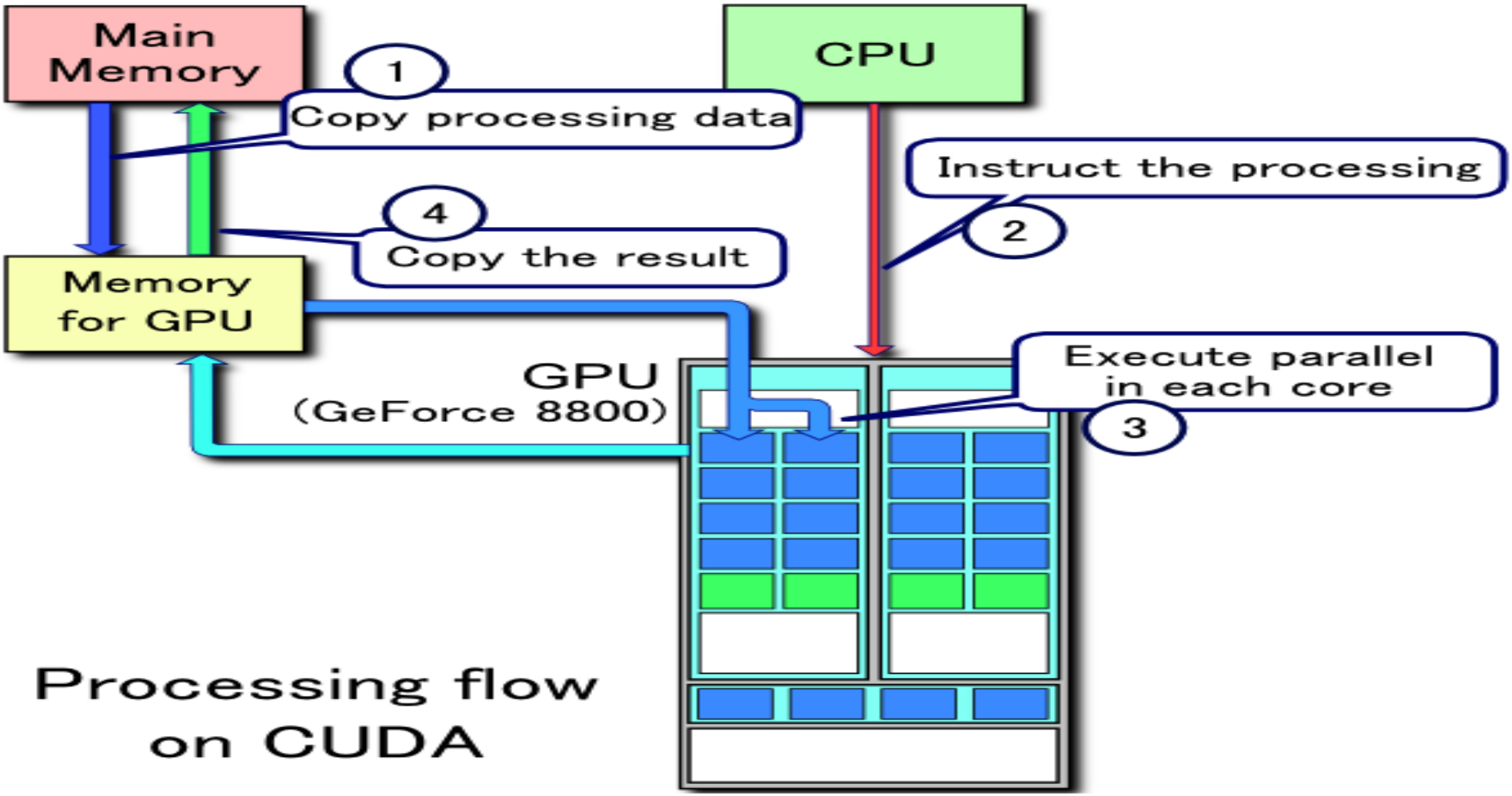

- Improvement of processor and network technologies over 30 years

# Multicore and Many-Core Architectures



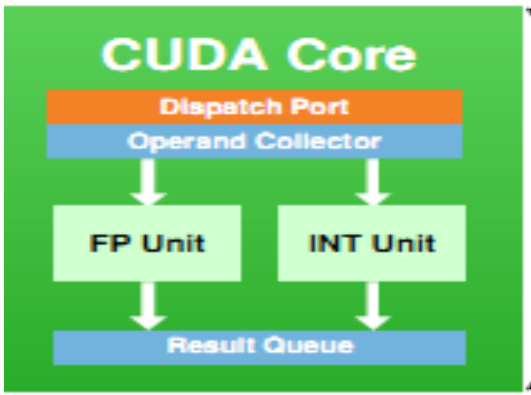

- The schematic of a modern multicore CPU chip using a hierarchy of caches, where the L1 cache is private to each core. The L2 cache is also on chip and shared by all cores. The L3 cache or DRAM is off the multicore chip.

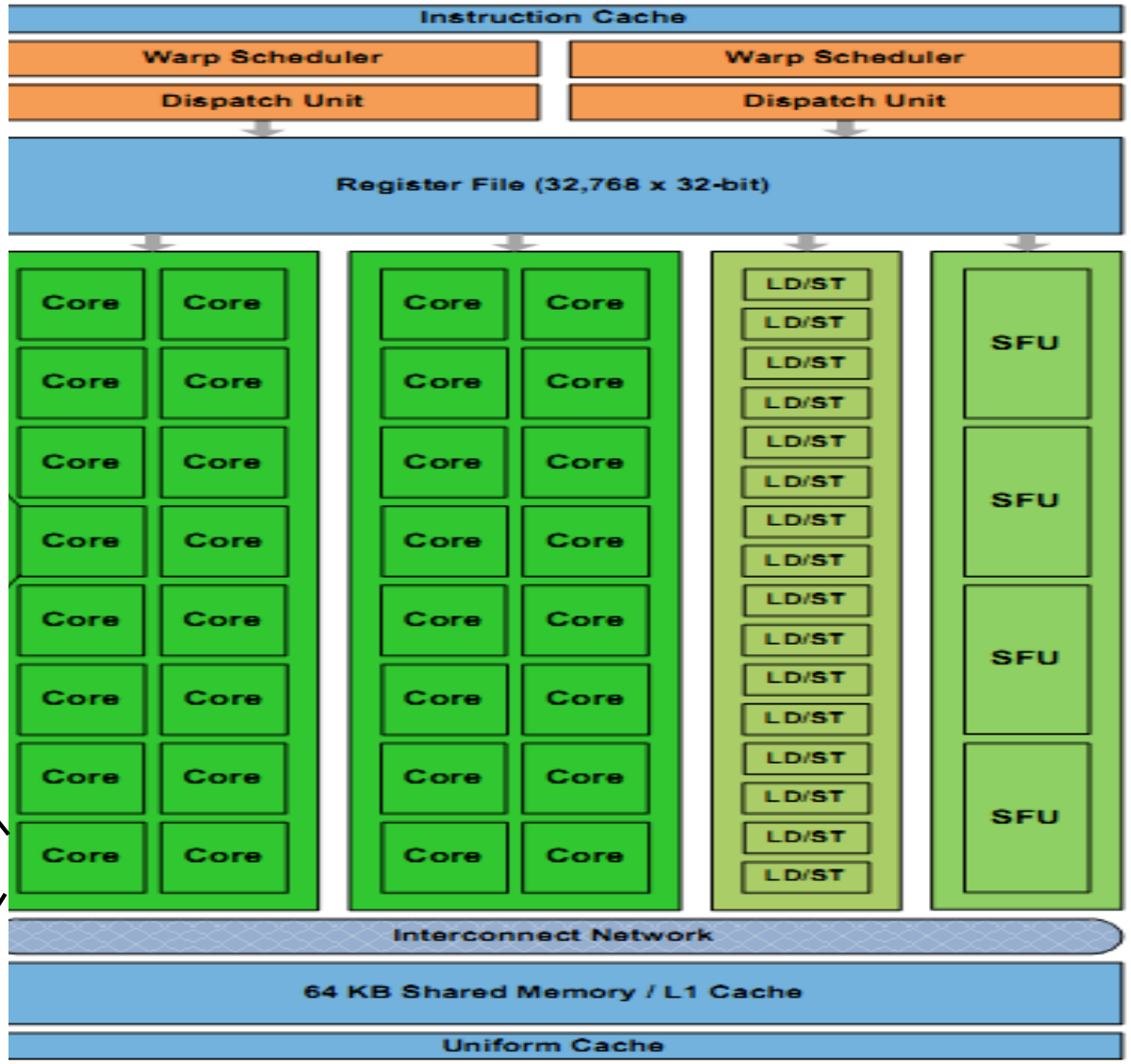
# GPU Computing To Exascale and Beyond



# GPU Computing To Exascale and Beyond

- Nvidia Fermi GPU built with 16 streaming multiprocessor s (SMs) of 32 Cuda cores each



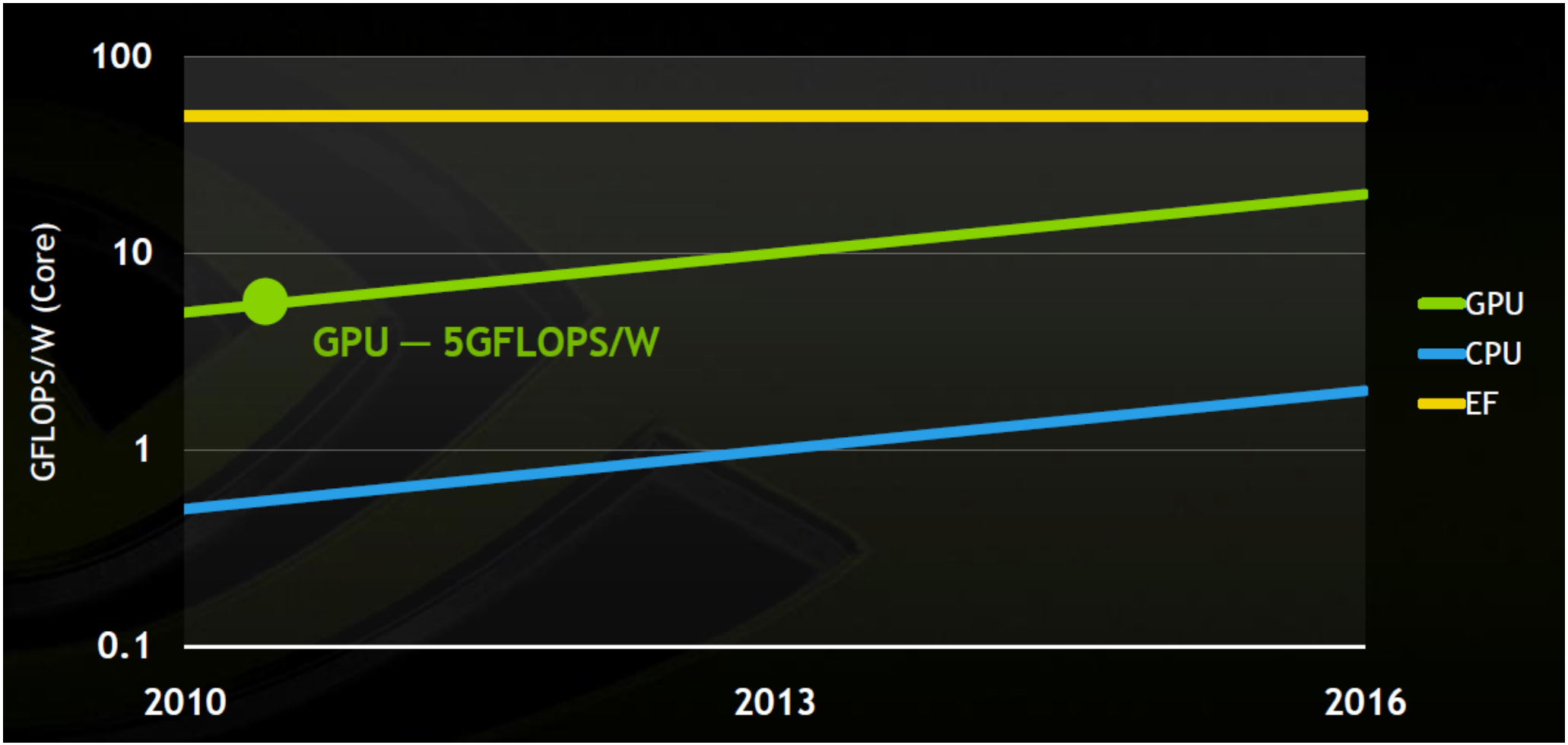



# GPU Computing To Exascale and Beyond

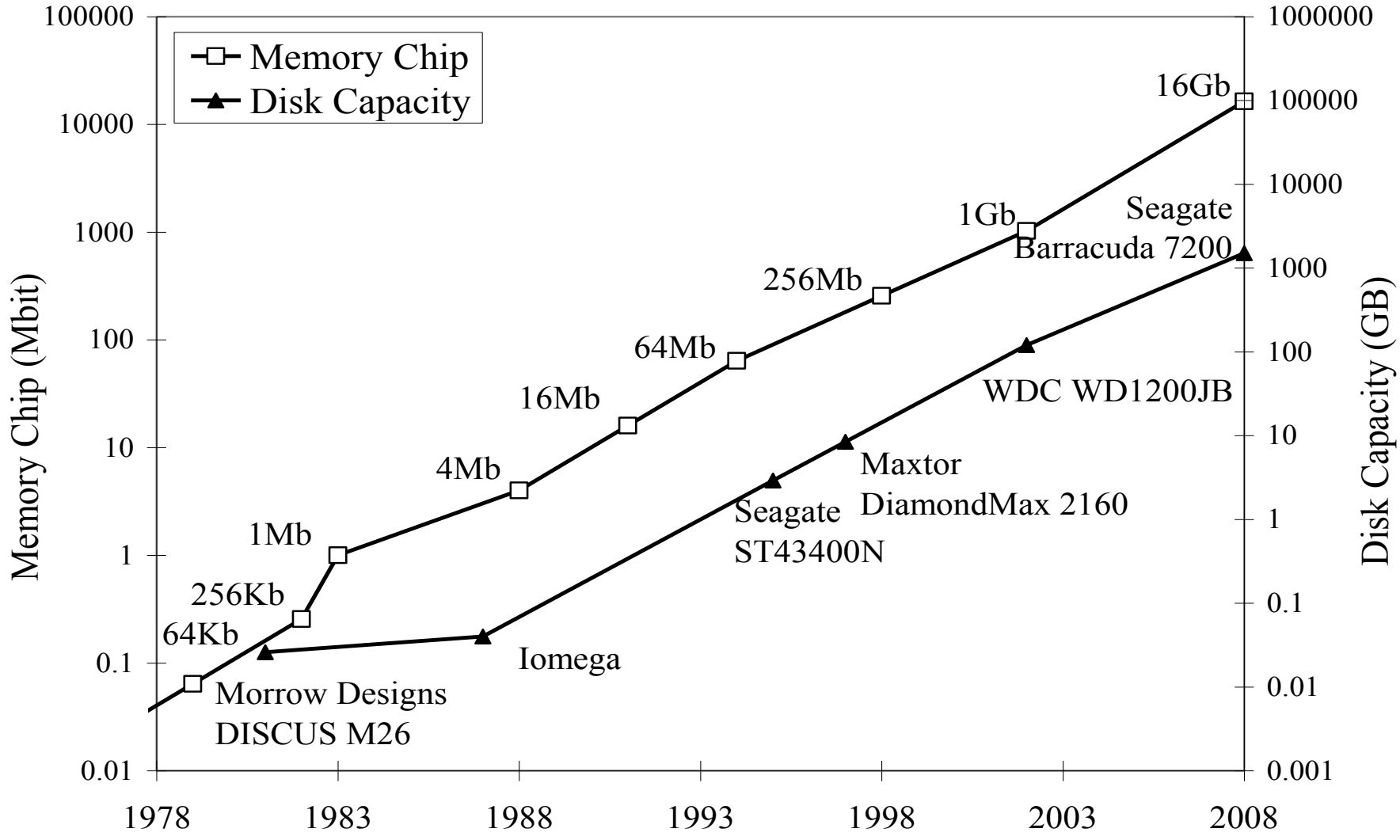
- Future exascale systems ($10^{18}$ flops)
  - Will likely have hybrid architectures, where GPUs will be used in combination with CPUs
- Challenges of Exascale Computing
  - energy and power
  - memory and storage
  - concurrency and locality
  - system resiliency

# GPU Computing To Exascale and Beyond

- Power efficiency of GPUs



# Memory, Storage, and System-Area Networking

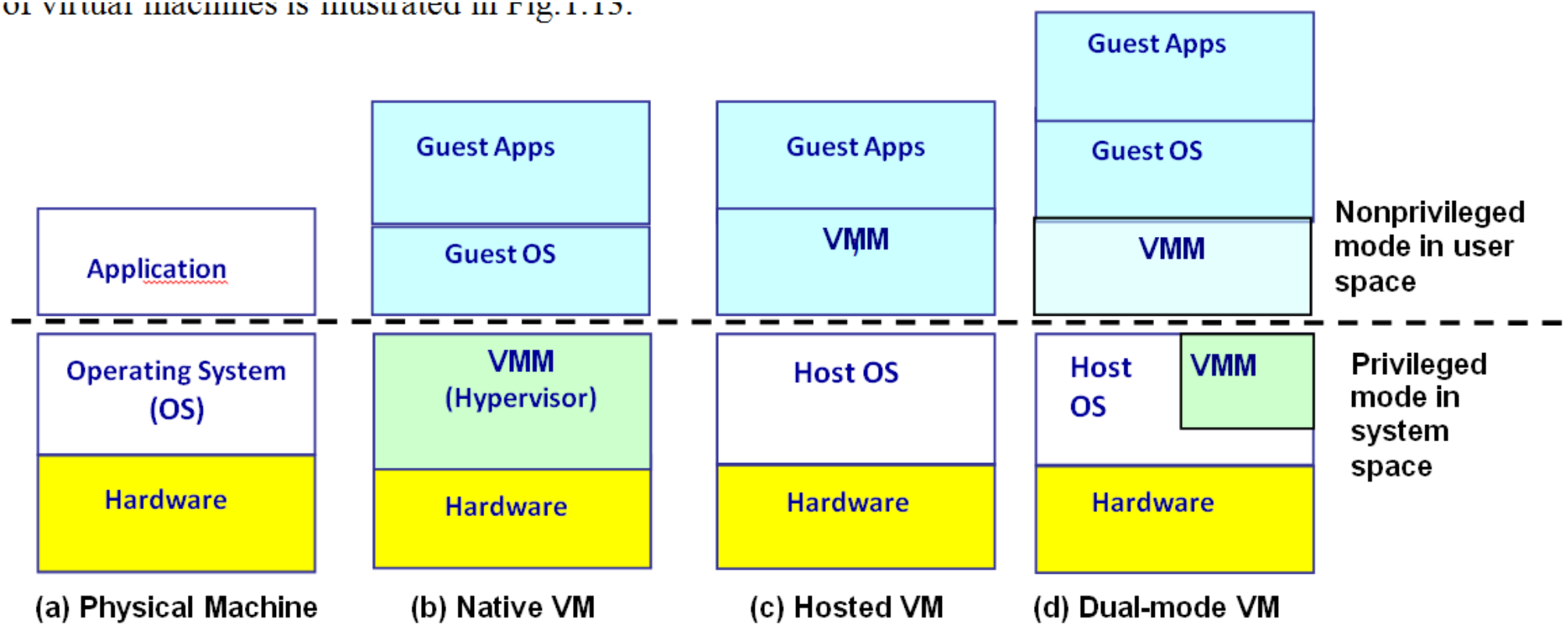



- Improvement of memory and disk technologies over 30 years

# Logistics

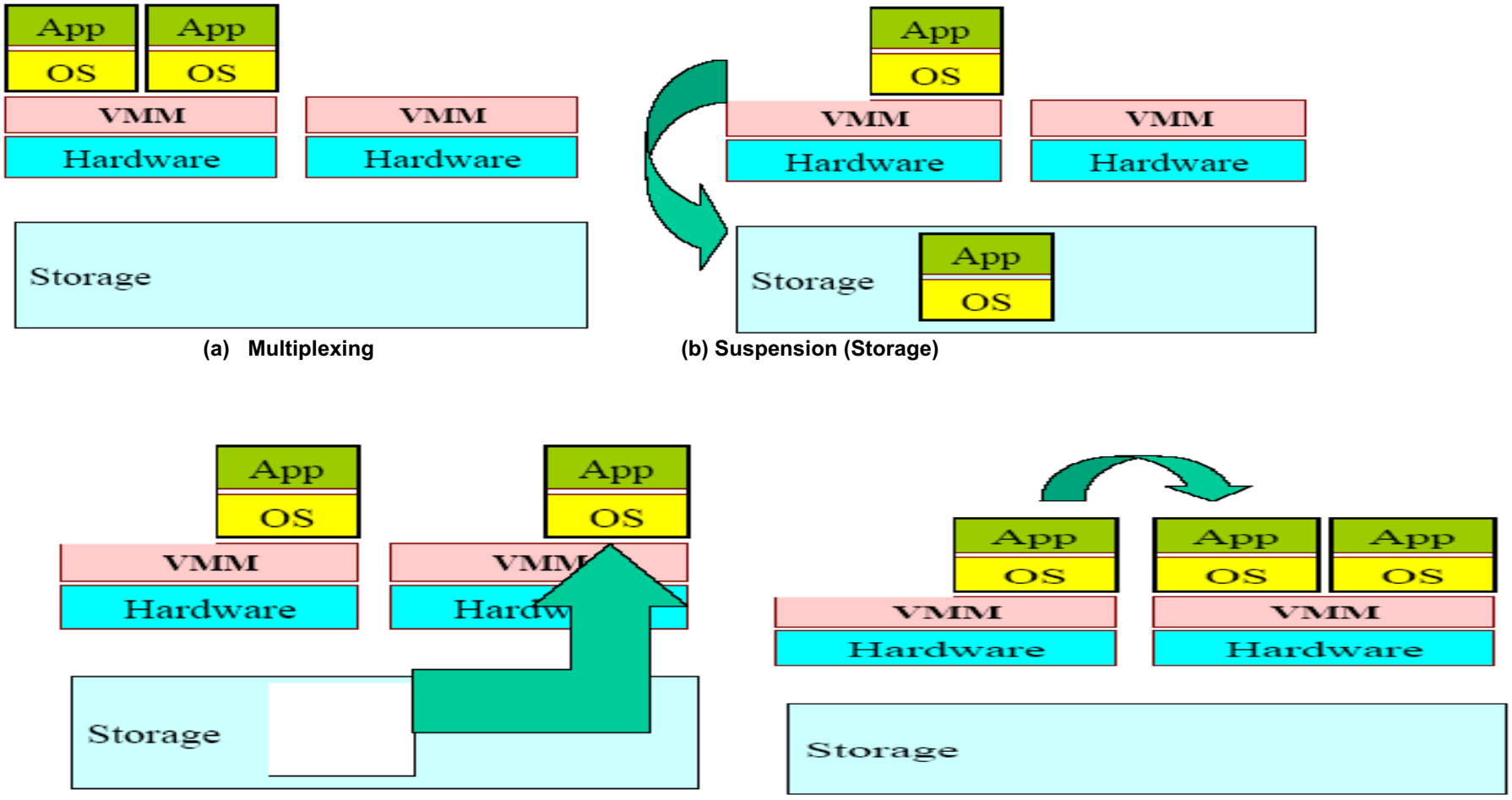
# Virtual Machines and Virtualization Middleware

of virtual machines is illustrated in Fig.1.13.



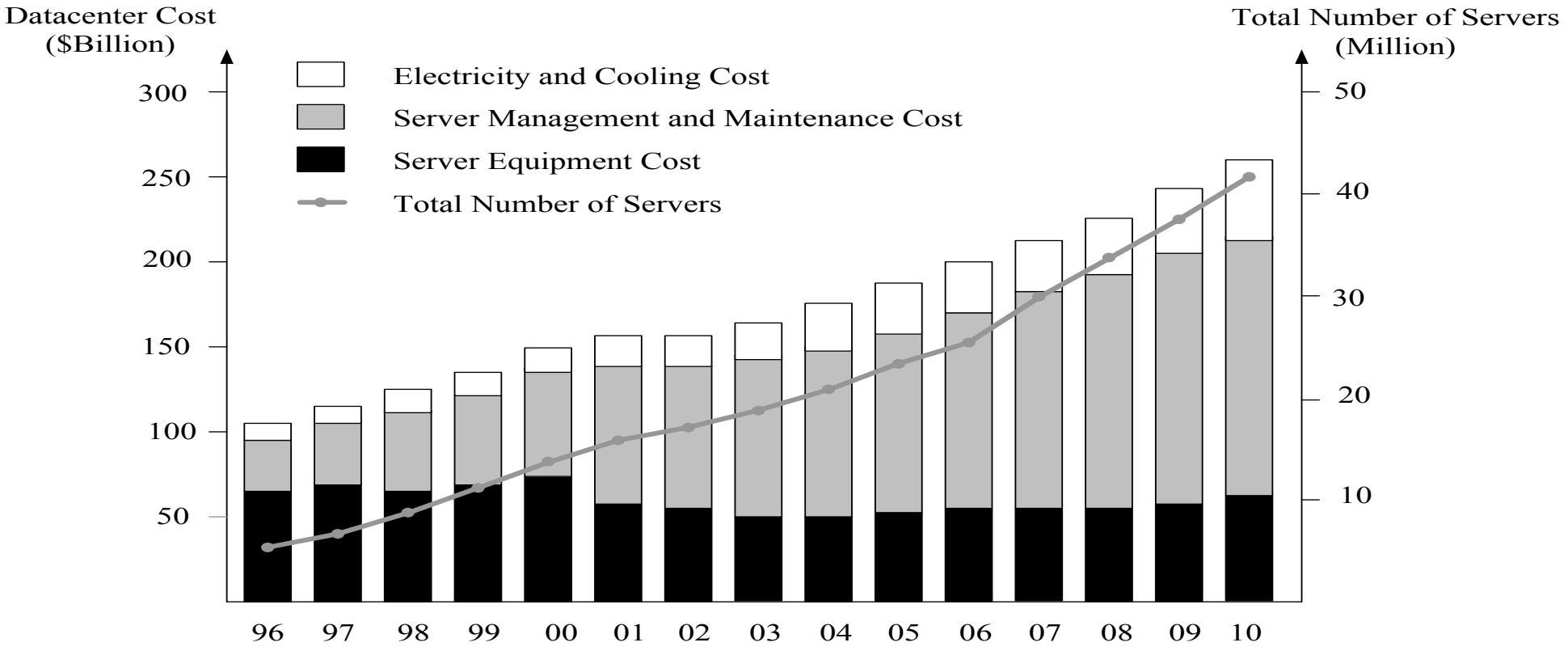

- Three virtual machine (VM) architectures in Parts (b-d), compared with the traditional physical machine shown in Part (a)

# Virtual Machines and Virtualization Middleware




- Virtual machine multiplexing, suspension, provision, and migration in a distributed computing environment

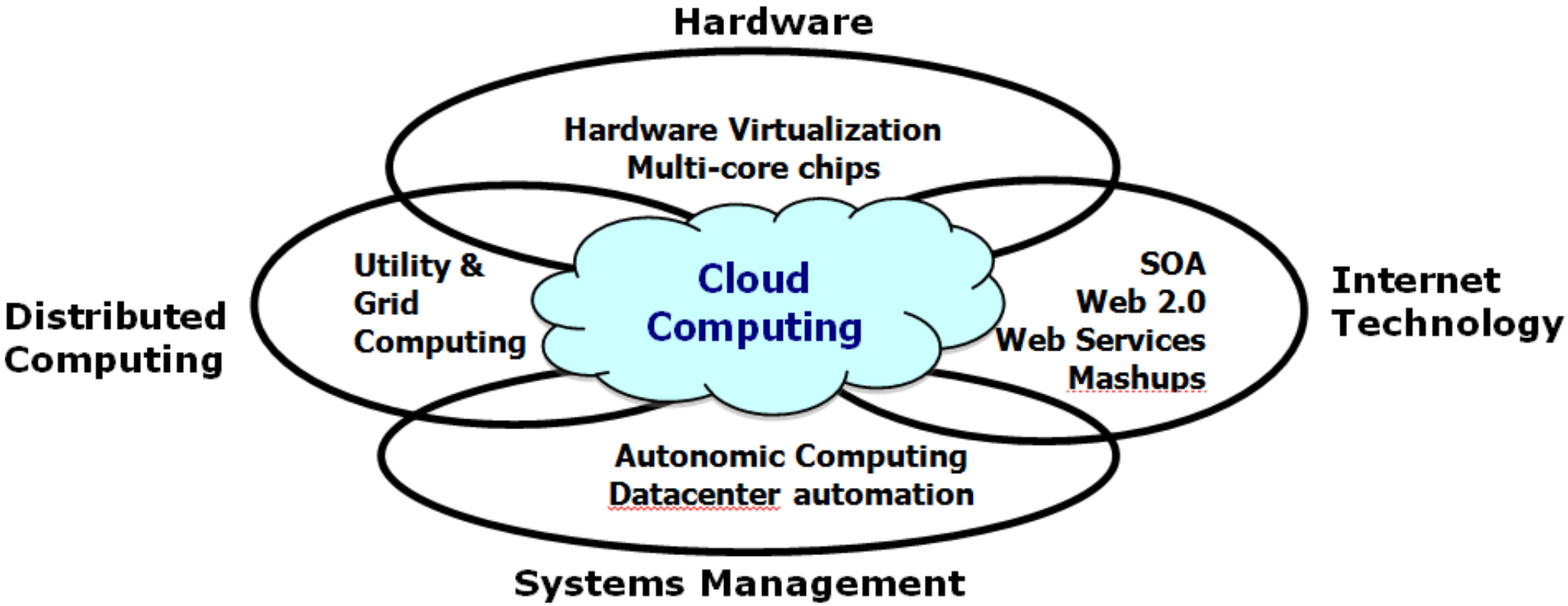
# Datacenter Virtualization for Cloud Computing




- Growth and cost breakdown of datacenters over the years

# Logistics

# Datacenter Virtualization for Cloud Computing




- Technological convergence enabling cloud computing over the Internet

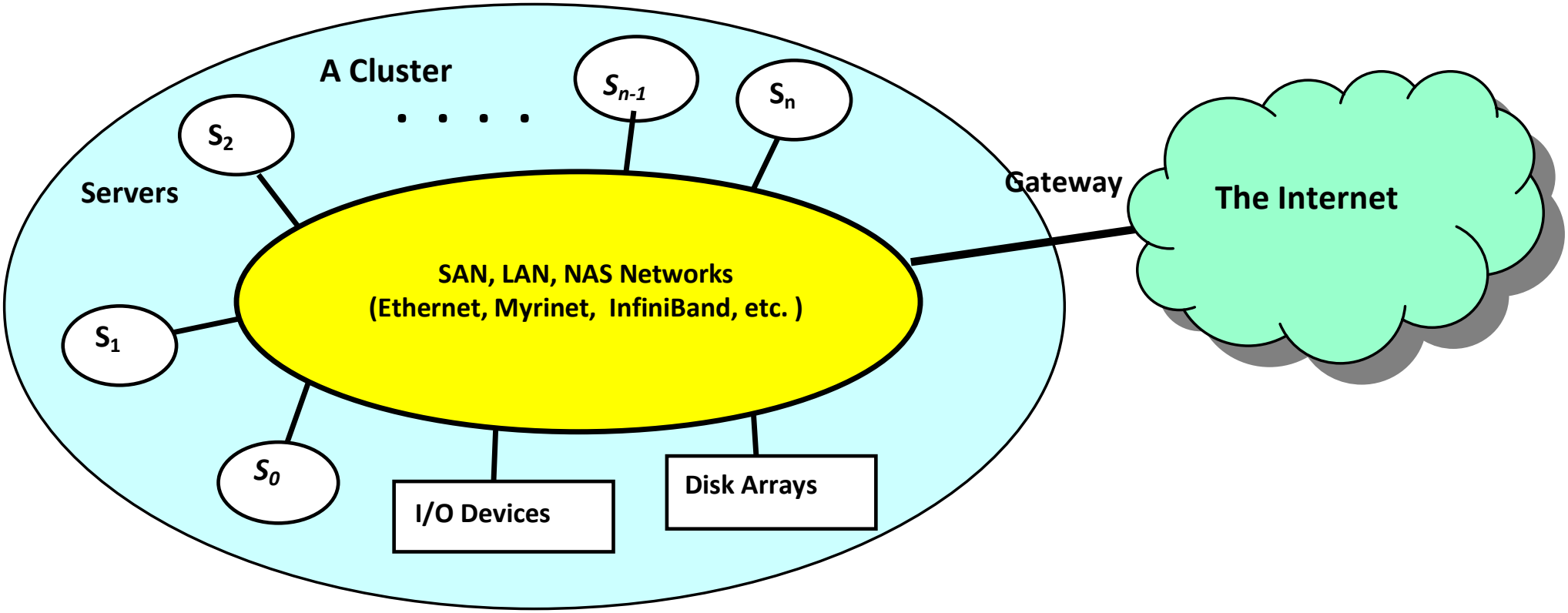
# System Models for Distributed and Cloud Computing

- *Clusters*
- *P2P networks*
- *Computing grids*
- *Internet clouds*

# Classification of Parallel and Distributed Computing Systems

| Functionality, Applications | Computer Clusters [10, 28, 35] | Peer-to-Peer Networks [42] | Data/Computational Grids [6, 44] | Cloud Platforms [1, 9, 11, 13, 30] |
|---|---|---|---|---|
| **Architecture, Network Connectivity and Size** | Network of compute nodes interconnected by SAN, LAN, or WAN, hierarchically | Flexible network of client machines logically connected by an overlay network | Heterogeneous clusters interconnected by high-speed network links over selected resource sites. | Virtualized cluster of servers over datacenters via service-level agreement |
| **Control and Resources Management** | Homogeneous nodes with distributed control, running Unix or Linux | Autonomous client nodes, free in and out, with self-organization | Centralized control, server oriented with authenticated security | Dynamic resource provisioning of servers, storage, and networks |
| **Applications and network-centric services** | High-performance computing, search engines, and web services, etc. | Most appealing to business file sharing, content delivery, and social networking | Distributed super-computing, global problem solving, and datacenter services | Upgraded web search, utility computing, and outsourced computing services |
| **Representative Operational Systems** | Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc. | Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA | TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc. | Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure |

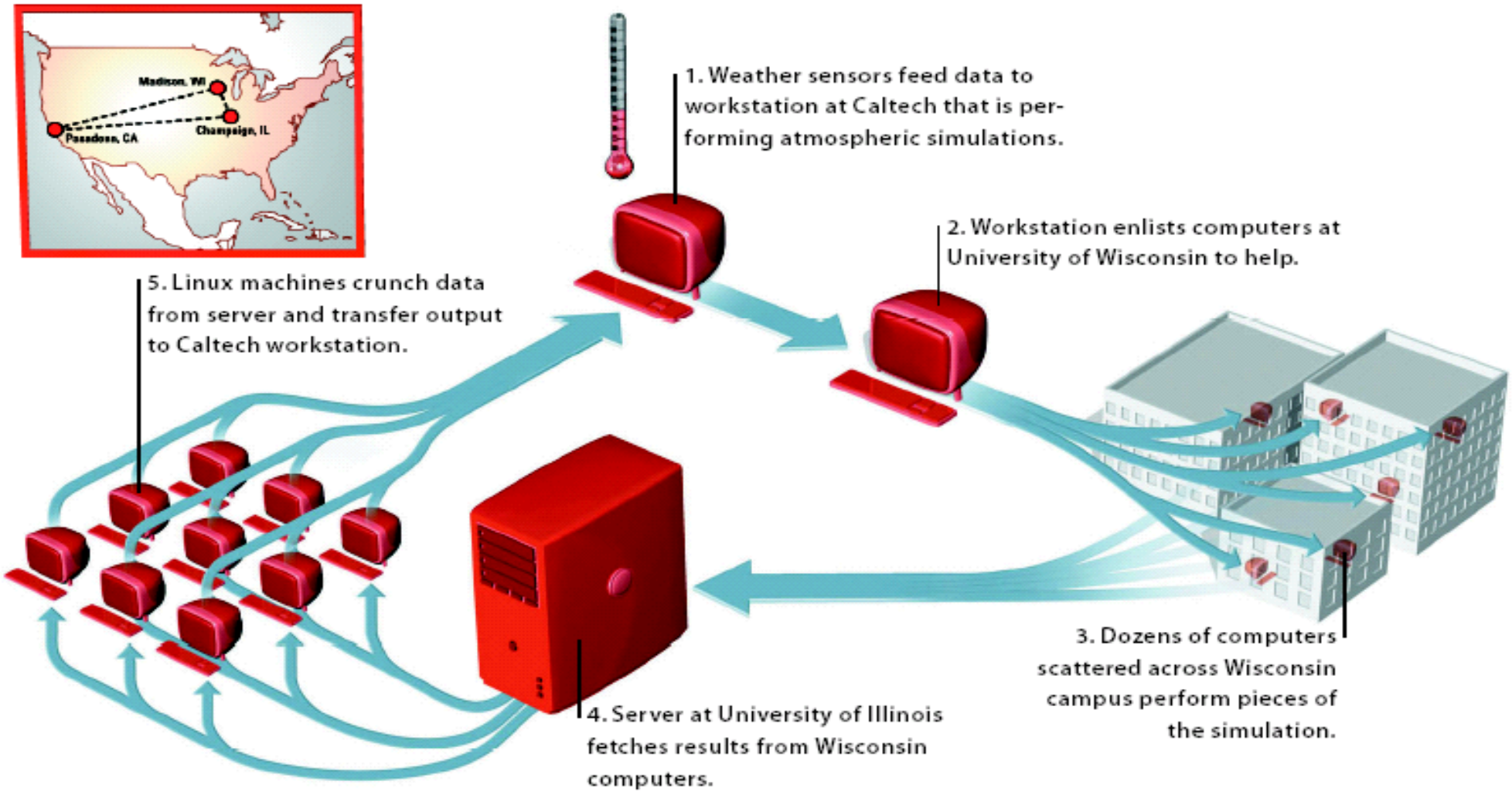
# Clusters of Cooperative Computers



# Clusters of Cooperative Computers
# Critical Design Issues

| Features | Functional Characterization | Feasible Implementations |
|---|---|---|
| **Availability Support** | Hardware and software support for sustained high availability in cluster | Failover, failback, checkpointing, roll back recovery, non-stop OS, etc |
| **Hardware Fault-Tolerance** | Automated failure management to eliminate all single points of failure | Component redundancy, hot swapping, RAID, and multiple power supplies, etc. |
| **Single-System Image (SSI)** | Achieving SSI at functional level with hardware and software support, middleware, or OS extensions. | Hardware mechanisms or middleware support to achieve distributed shared memory (DSM) at coherent cache level. |
| **Efficient Communications** | To reduce message-passing system overhead and hide latencies | Fast message passing , active messages, enhanced MPI library, etc. |
| **Cluster-wide Job Management** | Use a global job management system with better scheduling and monitory | Apply single-job management systems such as LSF, Codine, etc |
| **Dynamic Load Balancing** | Balance the workload of all processing nodes along with failure recovery | Workload monitory, process migration, job replication and gang scheduling, etc. |
| **Scalability and Programmability** | Adding more servers to a cluster or adding more clusters to a Grid as the workload or data set increases | Use scalable interconnect, performance monitory, distributed execution environment, and better software tools |

# Grid Computing Infrastructures

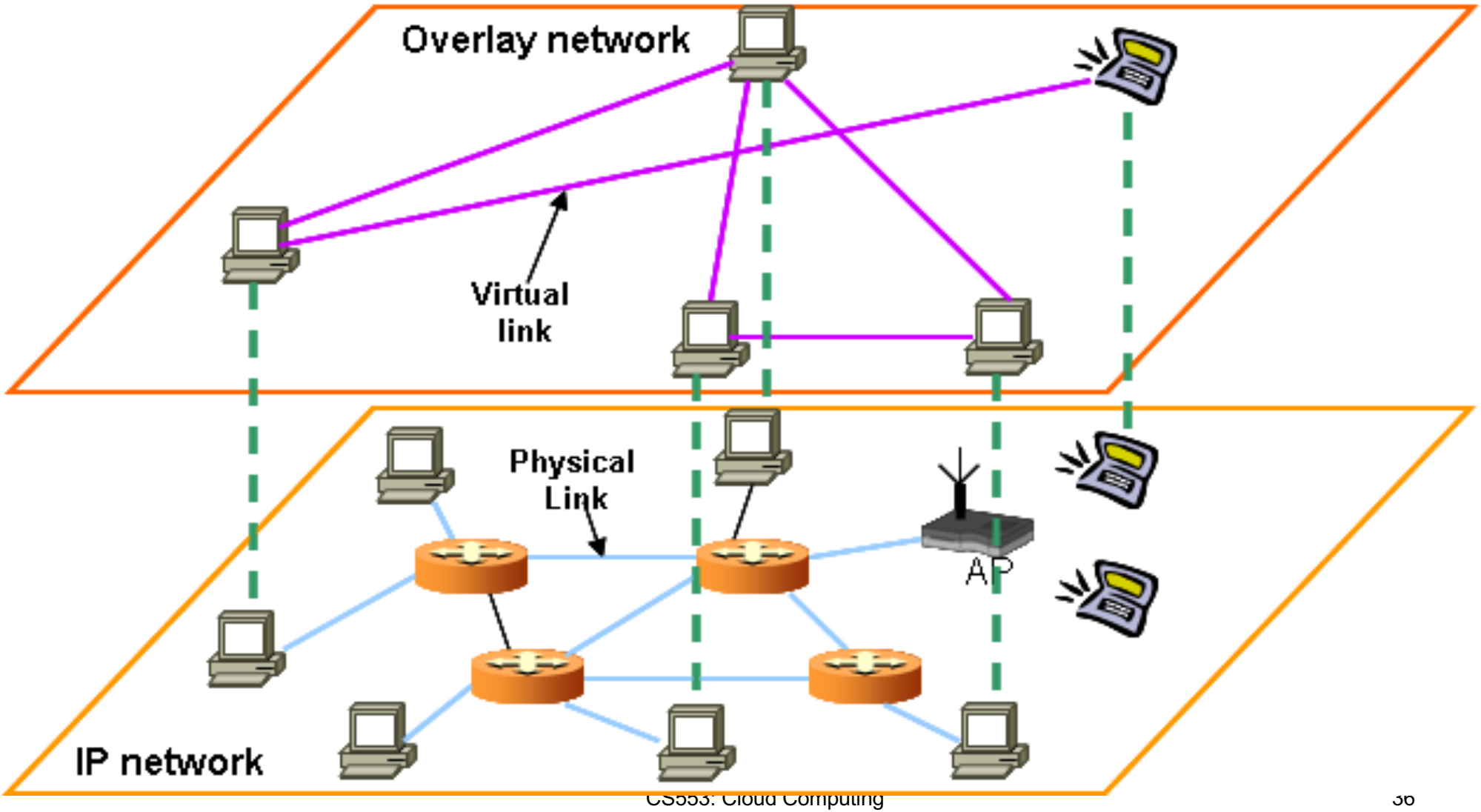


# Grid Computing Infrastructures

- Two Grid Computing Infrastructures and Representative Systems

| Design Issues | Computational and Data Grids | P2P Grids |
|---|---|---|
| Grid Applications reported | Distributed Supercomputing, National Grid Initiatives, etc | Open grid with P2P flexibility, all resources from client machines |
| Representative Systems | TeraGrid in US, ChinaGrid, UK e-Science, etc. | JXTA, FightAid@home, SETI@home |
| Development Lessons learned | Restricted user groups, middleware bugs, protocols to acquire resources | Unreliable user-contributed resources, limited to a few apps. |

# Peer-to-Peer Network Families



# Peer-to-Peer Network Families

- **Major Categories of Peer-to-Peer Network Families**

| System Features | Distributed File Sharing | Collaborative Platform | Distributed P2P Computing | Peer-to-Peer Platform |
|---|---|---|---|---|
| **Attractive Applications** | Content distribution of MP3 music, video, open software, etc. | Instant Messaging, Collaborative design and gaming | Scientific exploration and social networking | Open networks for public resources |
| **Operational Problems** | Loose security and serious on-line copyright violations | Lack of trust, disturbed by spam, privacy, and peer collusions | Security holes, selfish partners, and peer collusion | Lack of standards or protection protocols |
| **Example Systems** | Gnutella, Napster, eMule, BitTorrent, Aimster, KaZaA, etc. | ICQ, AIM, Groove, Magi, Multiplayer Games, Skype, etc. | SETI@home, Geonome@ home, etc. | JXTA, .NET, FightingAid@ home, etc. |

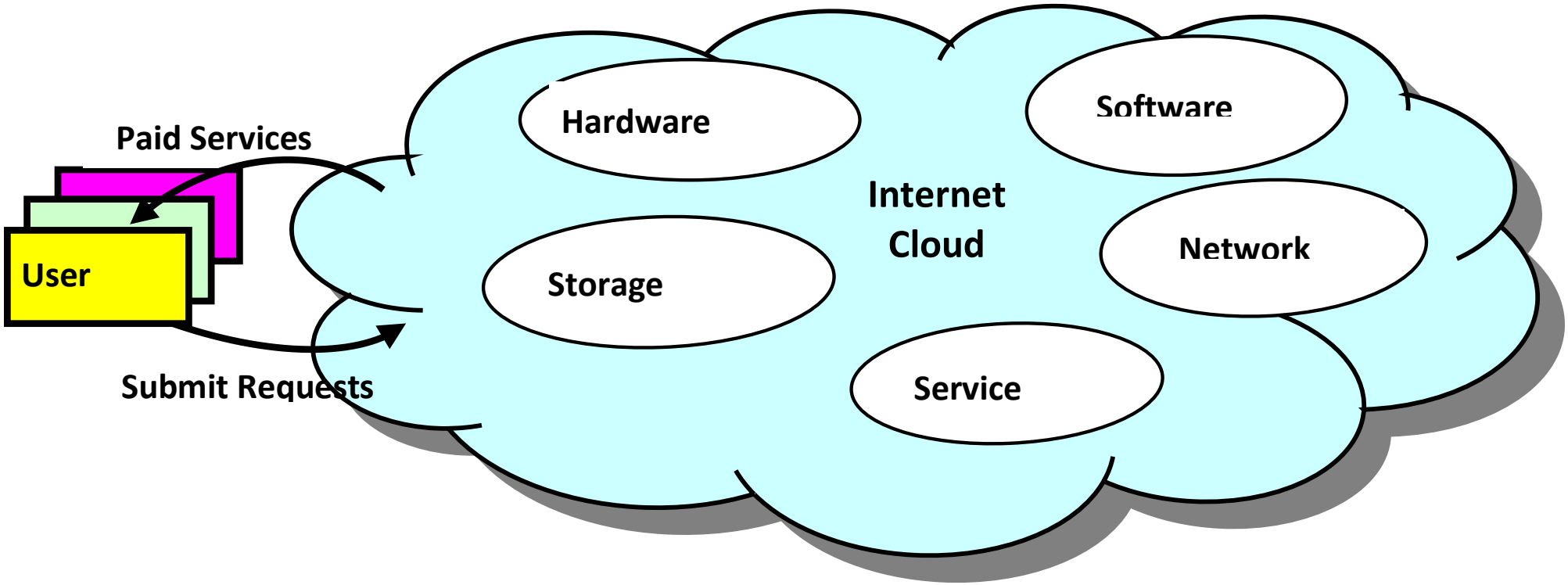
# Cloud Computing over The Internet Definition

- *A large-scale distributed computing paradigm that is driven by economics of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.* *-- Ian Foster, 2008*

# Cloud Computing over The Internet Common Characteristics

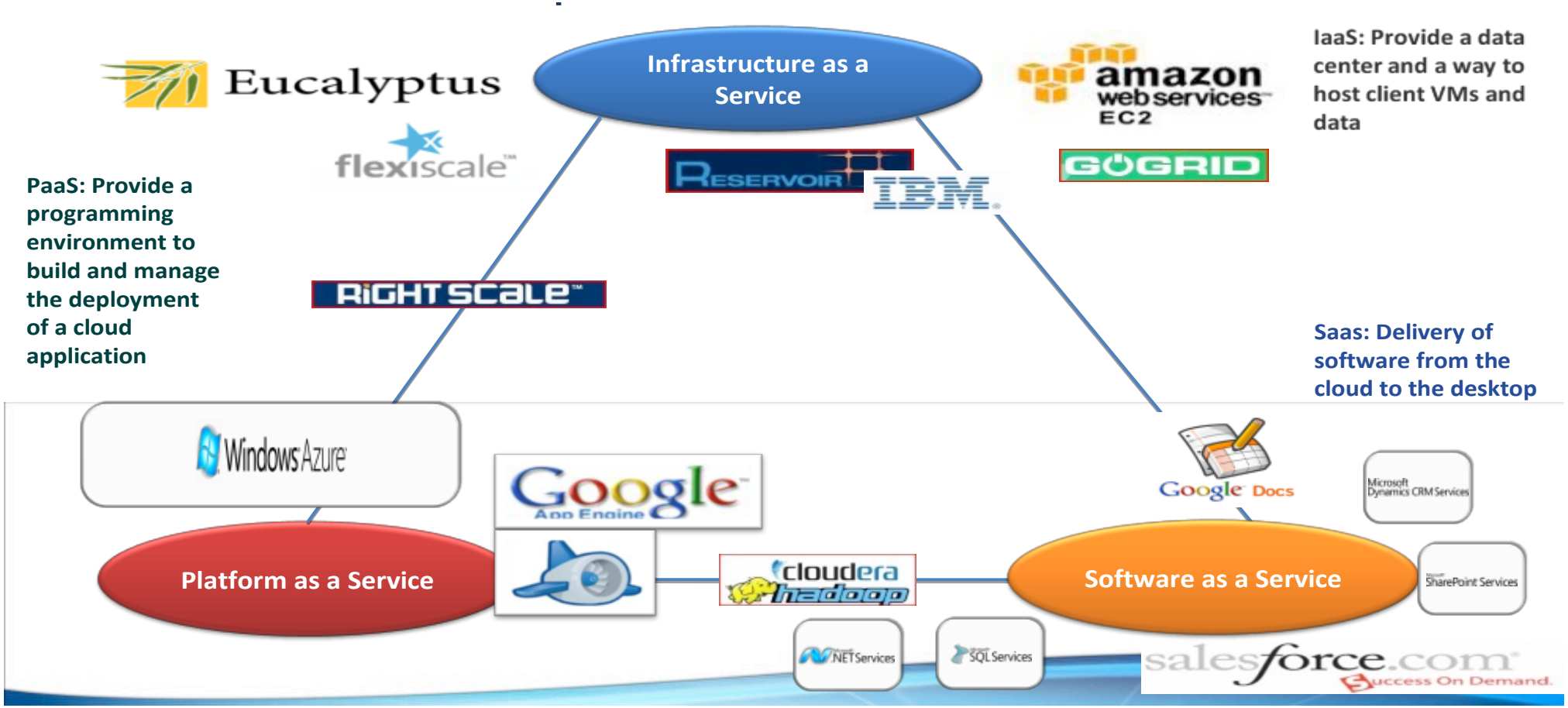
1. Cloud platform offers a scalable computing paradigm built around the datacenters.
2. Cloud resources are dynamically provisioned by datacenters upon user demand.
3. Cloud system provides compute, storage and flexible platforms for upgraded web services.
4. Cloud computing relies heavily on the virtualization of all sorts of resources.
5. Cloud computing defines a new paradigm for collective computing, data consumption and delivery of information services over the Internet.
6. Clouds stress the cost of ownership reduction in mega datacenters.

# Cloud Computing over The Internet
# Pay as you go Cloud Service



# Cloud Computing over The Internet Infrastructure

# Cloud Computing over The Internet
# Cloud Landscape

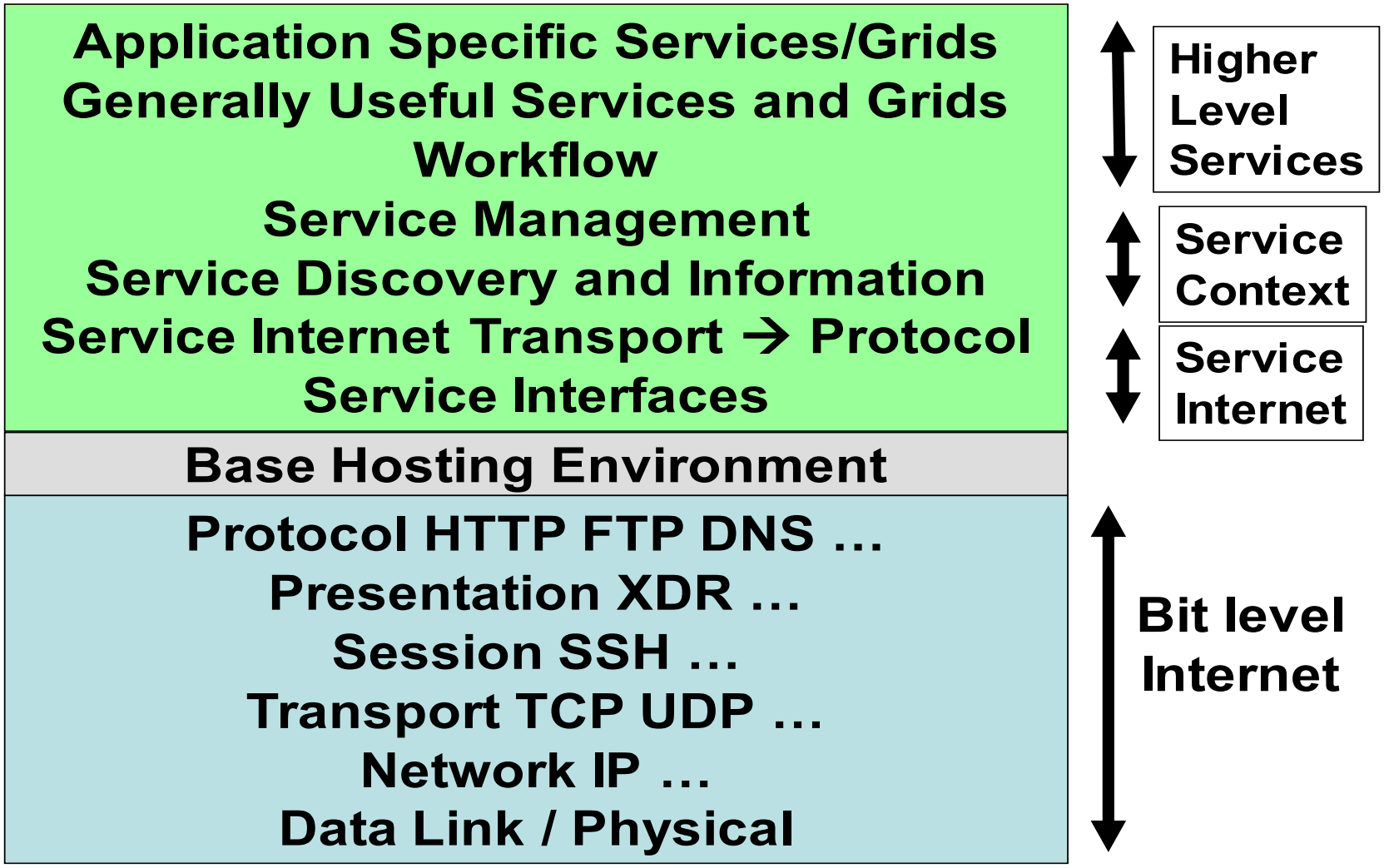


# Logistics

# Cloud Computing over The Internet
# Cloud Landscape

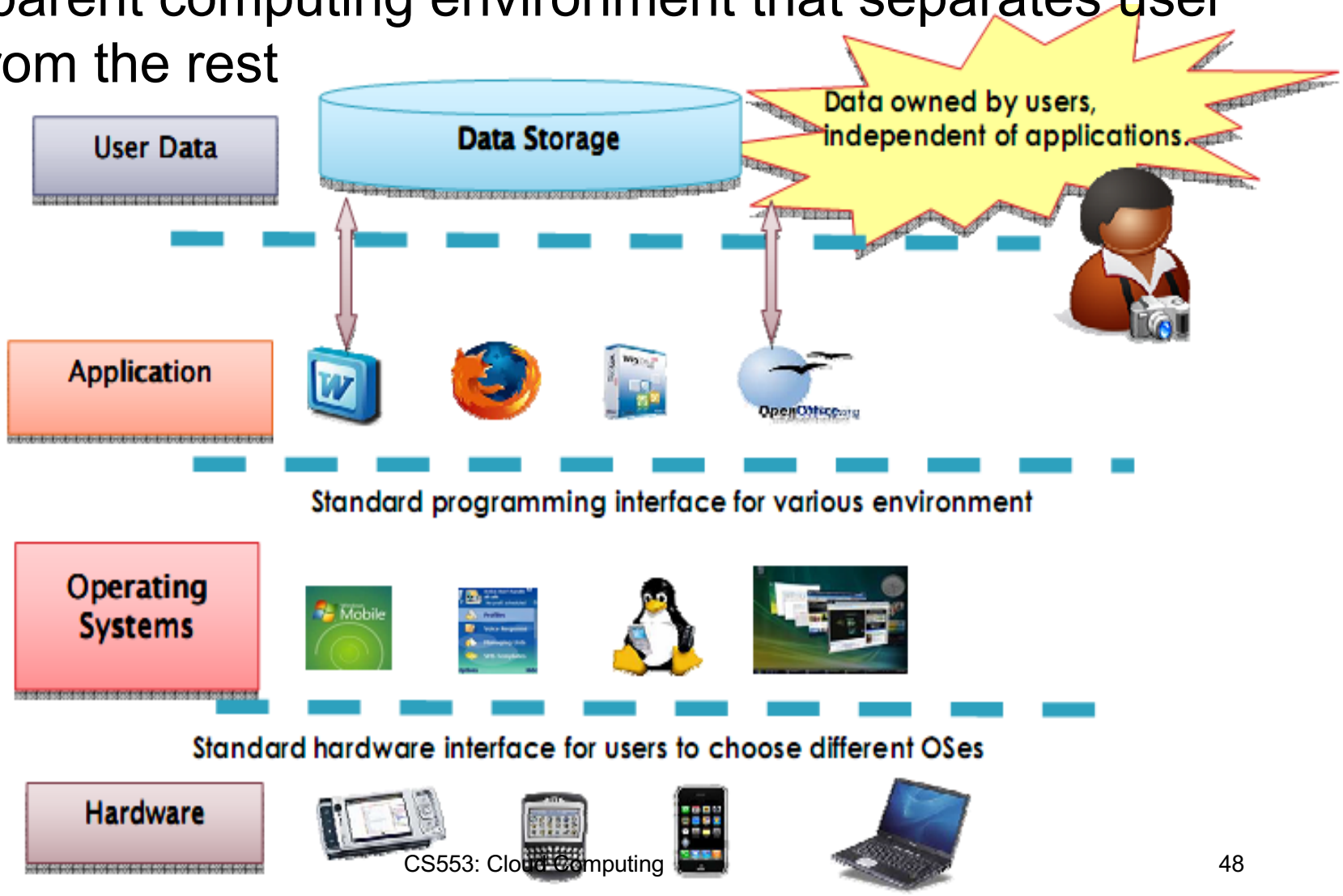
- ***Infrastructure as a Service (IaaS):***
  - This model put together infrastructures demanded by users, namely servers, storage, networks, and datacenter fabric. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.
- ***Platform as a Service (PaaS)****:*
  - This model provides the user to deploy user-built applications onto a virtualized cloud platform. PaaS include middleware, database, development tools, and some runtime supports like Web 2.0 and Java, etc. The platform include both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, python, Web2.0, .Net). The user is freed from managing the cloud infrastructure.
- ***Software as a Service (SaaS):***
  - This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, CRM (*consumer relationship management)*, ERP (*enterprise resources planning*), HR (*human resources*) and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

# Service-Oriented Architecture (SOA) Layered Architecture



# Trends towards Distributed Operating Systems

- Transparent computing environment that separates user data from the rest



# Parallel and Distributed Programming Models

- Parallel and Distributed Programming Models and Toolsets

| Model | Objectives and Web Link | Attractive Features Implemented |
|---|---|---|
| **MPI** | Message-Passing Interface is a library of subprograms that can be called from C or Fortran to write parallel programs running on distributed computer systems [3, 28, 42] | Specify synchronous or asynchronous point-to-point and collective communication commands and I/O operations in user programs for message-passing execution |
| **MapReduce** | A web programming model for scalable data processing on large cluster over large datasets, or in web search operations [17] | A *Map* function generates a set of intermediate key/value pairs. A *Reduce* function to merge all intermediate values with the same key |
| **Hadoop** | A software library to write and run large user applications on vas datasets in business applications. http://hadoop.apache.org/core/ | Hadoop is scalable, economical, efficient and reliable in providing users with easy access of commercial clusters |

# System Scalability

- ***Size Scalability:***
  - This refers to achieve higher performance or performing more functionality by increasing the *machine size*. The word "size" refers to adding the number of processors; more cache, memory, storage or I/O channels. The most obvious way to simple counting the number of processors installed. Not all parallel computer or distributed architectures are equally size-scalable. For example, IBM S2 was scaled up to 512 processors in 1997. But in 2008, the IBM BlueGene/L system scaled up to 65,000 processors.
- ***Software Scalability:***
  - This refers to upgrades in OS or compilers, adding mathematical and engineering libraries, porting new application software, and install more user-friendly programming environment. Some software upgrade may not work with large system configurations. Testing and fine-tuning of new software on larger system is a non-trivial job.
- ***Application scalability:***
  - This refers to the match of *problem size* scalability with the *machine size* scalability. Problem size affects the size of the data set or the workload increase. Instead of increasing machine size, users can enlarge the problem size to enhance the system efficiency or cost-effectiveness.
- ***Technology Scalability:***
  - This refers to a system that can adapt to changes in building technologies, such as those component and networking technologies discussed in Section 3.1. Scaling a system design with new technology must consider three aspects: *time, space,* and *heterogeneity.* Time refers to generation scalability. Changing to new-generation processors, one must consider the impact to motherboard, power supply, packaging and cooling, etc. Based on the past experience, most system upgrade their commodity processors every 3 to 5 years. Space is more related to packaging and energy concerns. Technology scalability demands harmony and portability among suppliers.

# Amdahl's Law

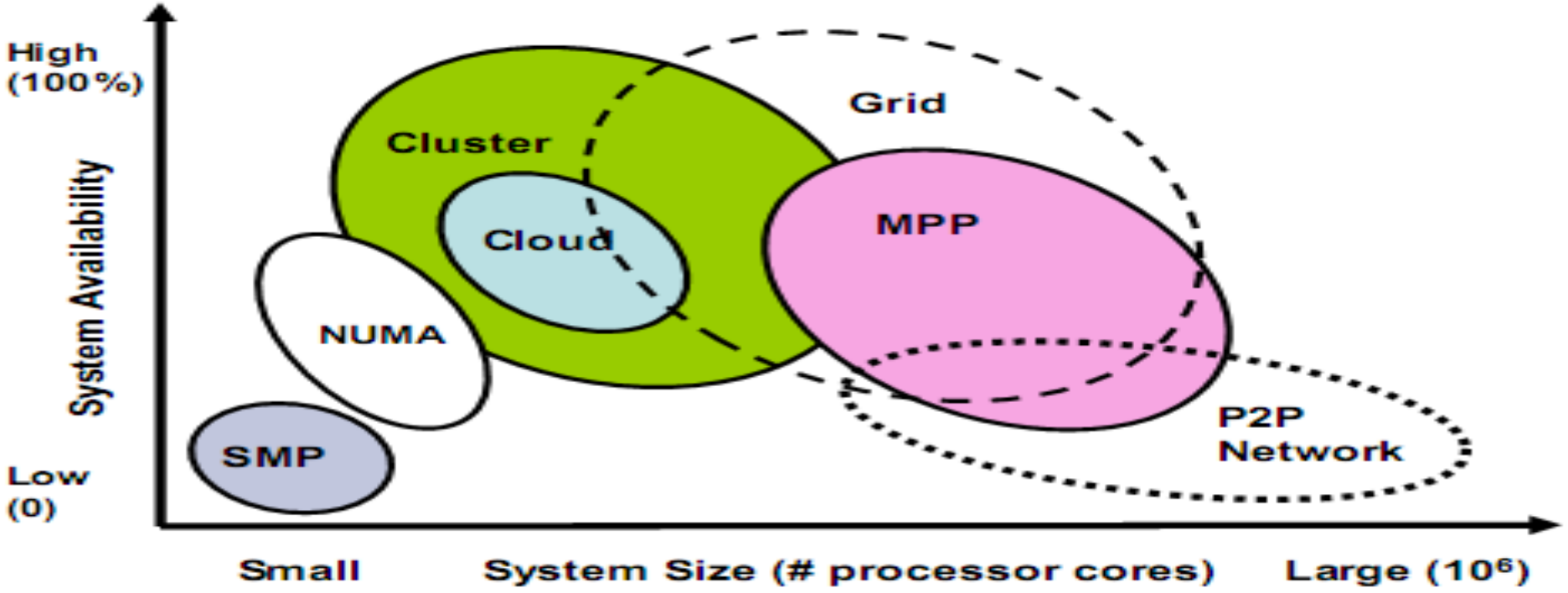
- **Speedup = $S = T / [ \alpha T + (1-\alpha) T/n ] = 1 / [ \alpha + (1-\alpha)/n ]$**
- The maximum speedup of n is achieved, only if the sequential bottleneck α is reduced to zero or the code is fully parallelizable with α = 0. As the cluster becomes sufficiently large, i.e. $n \rightarrow \infty$, S approaches $1/\alpha$, an upper bound on the speedup S.
- Amdahl's law teaches us that we should make the sequential bottleneck as small as possible.
- Increasing the cluster size alone may not give a good speedup in this case.

# Amdahl's Law

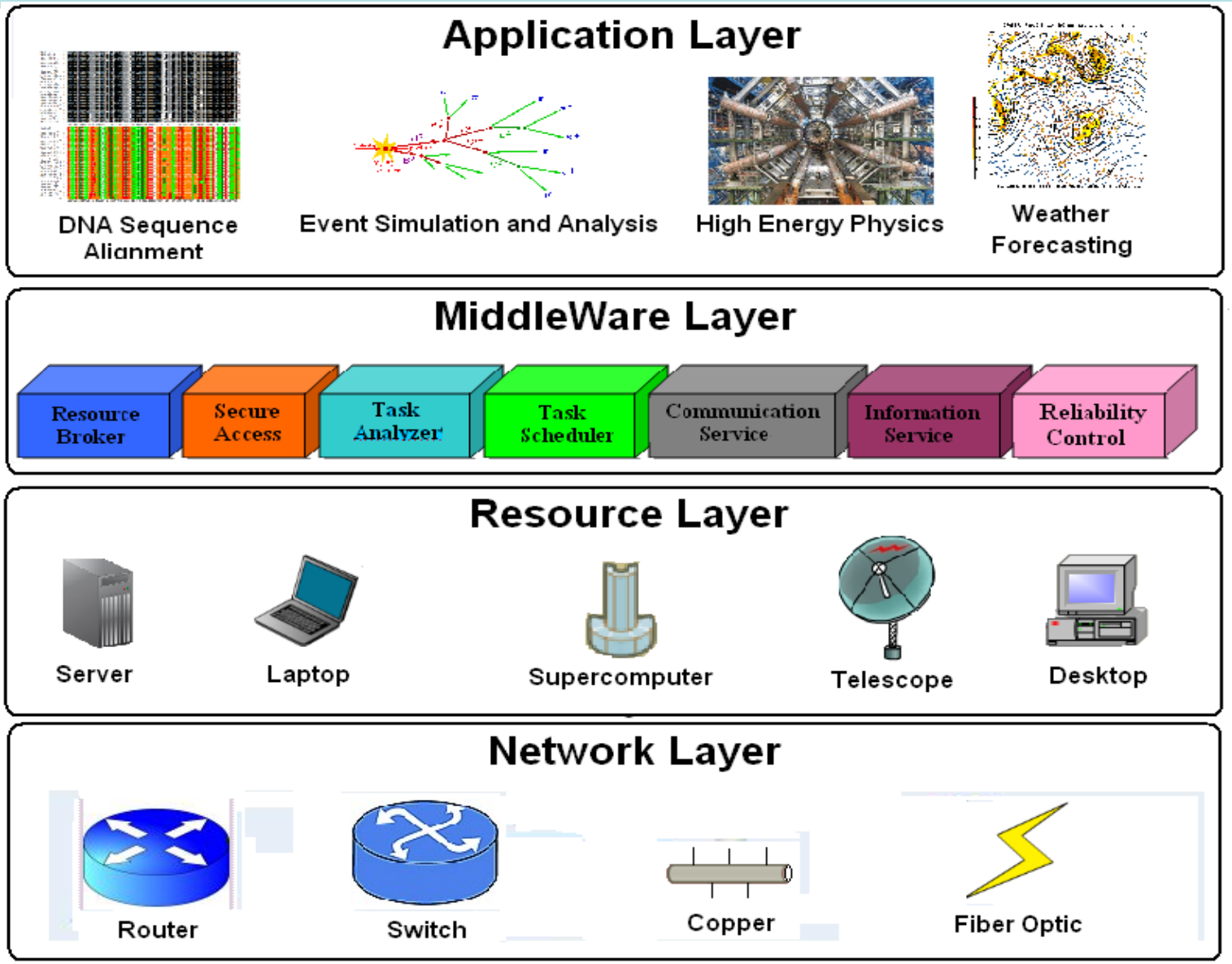
- $E = S / n = 1 / [\ \alpha\, n + 1 - \alpha\ ]$
- Fixed workload size poses efficiency problems
- **Gustafson's Law**
  - Allows variable size workloads to increase efficiency
  - For fixed workload, apply Amdahl's law
  - For scaled problems, apply Gustafson's Law

# Fault-Tolerance and System Availability

- *System Availability = MTTF / ( MTTF + MTTR )*



# Energy-Efficiency in Distributed Computing



# Questions