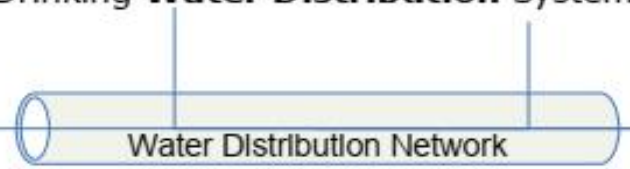
- Nearly 20-50 billion devices have been added to the internet.
- Nearly 43 trillion gigabytes of data is generated

(1) Drinking **Water Distribution** Systems



Seamlessly delivering Water as a Utility to Users

(2) **Electricity Distribution** Systems

kerosene lamp or **candles**

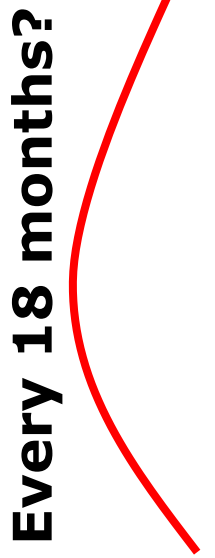
Seamless Electricity delivery as a Utility to Users

(3) **Computing Resource Distribution** Systems

Conventional Computing

Seamless computing delivery as a Utility 24/7 from any where

The Next Revolution in IT



- Classical Computing
  - Buy & Own
    - Hardware, System Software, Applications often to meet peak needs.
  - Install, Configure, Test, Verify
  - Manage
  - ..
  - Finally, use it
  - $$$$....$(High CapEx)

“Computer Utilities” Vision: Implications of the Internet

- 1969 – Leonard Kleinrock, ARPANET project
  - “As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of “Computer Utilities”, which, like present electric and telephone utilities, will service individual homes and offices across the country”.
  - During the last 50 years, several advances have taken place in both “computing” and “communications” areas that are turning the vision of “Computer Utilities” in to a reality.

- *I don’t care where my servers are, who manages them, where my documents are stored, or where my applications are hosted. I just want them always available and access them from any device connected through Internet. And I am willing to pay for this service for as a long as I need it.*

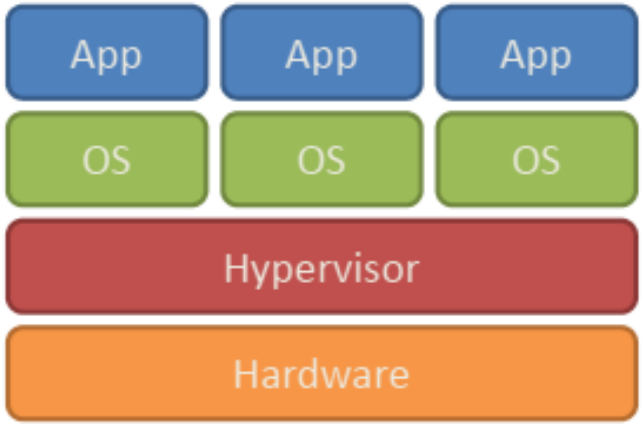
Clouds: Why Now (not then) ?

- Experience with very large datacenters
  – Unprecedented economies of scale
  – Transfer of risk
- Technology factors
  – Pervasive broadband Internet
  – Maturity in Virtualization Technology
- Business factors
  – Minimal capital expenditure
  – Pay-as-you-go billing model

I need to grow my infrastructure, but I do not know for how long…	I have a lot of infrastructure that I want to rent …
I cannot invest in infrastructure, I just started my business….	I have a surplus of infrastructure that I want to make use of
I want to focus on application logic and not maintenance and scalability issues	I have infrastructure and middleware and I can host applications
I want to access and edit my documents and photos from everywhere..	I have infrastructure and provide application services

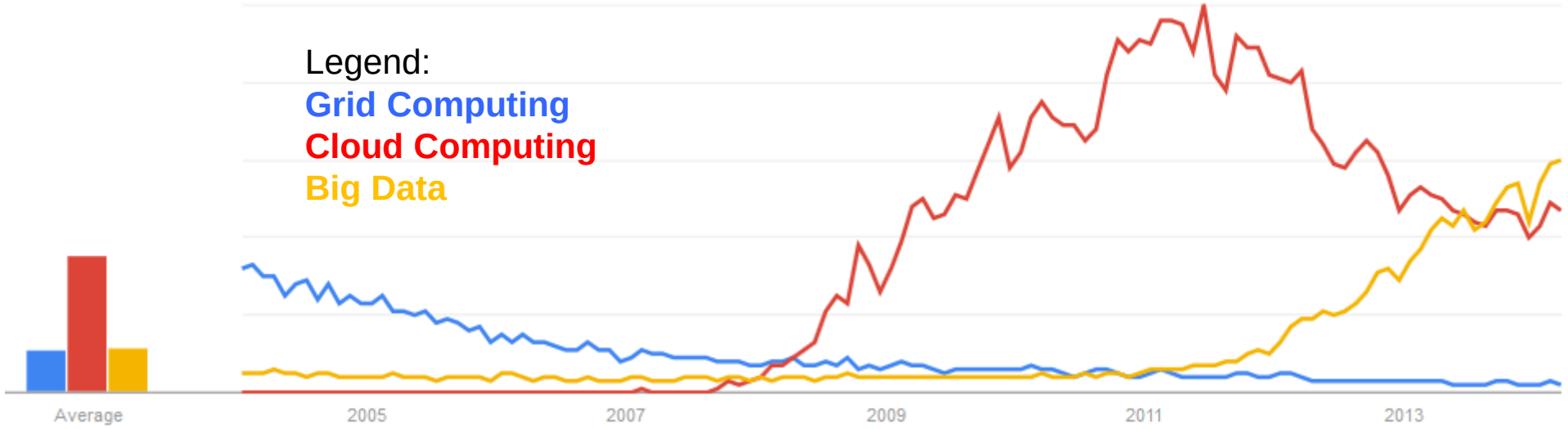
Virtualization

- Virtualization techniques are the basis of the cloud computing
- Virtualization technologies partition hardware and thus provide flexible and scalable computing platforms
- Virtual machine techniques
  – VMware and Xen
  – OpenNebula
  – Amazon EC2



Virtualized Stack

Interest over time
{grid, cloud, big data} computing



Defining Clouds

- Over 20 definitions
- Buyya's Scientific definition of Cloud Computing ☺
  - "Cloud is a **market-oriented** distributed computing system consisting of a collection of inter-connected and **virtualised** computers that are **dynamically provisioned** and presented as one or more unified computing resources based on **service-level agreements (SLAs)** established through **negotiation** between the service provider and consumers."
  - SLA = {negotiated and agreed QoS parameters + rewards + penalties for violation of agreement....}

Cloud Computing Characteristics

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability[1] at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Cloud Deployment Models

Public/Internet Clouds

*** 3rd party, multi-tenant Cloud infrastructure & services:**

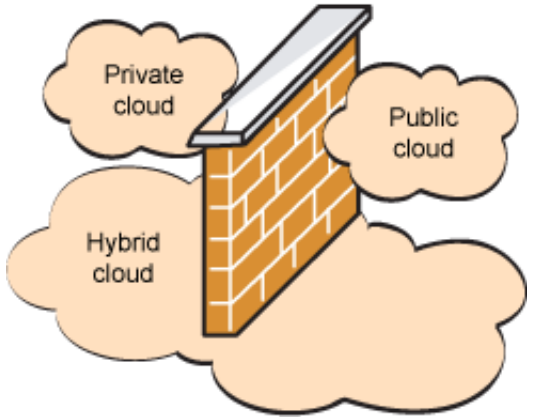
*** available on subscription basis to all.**

Private/Enterprise Clouds

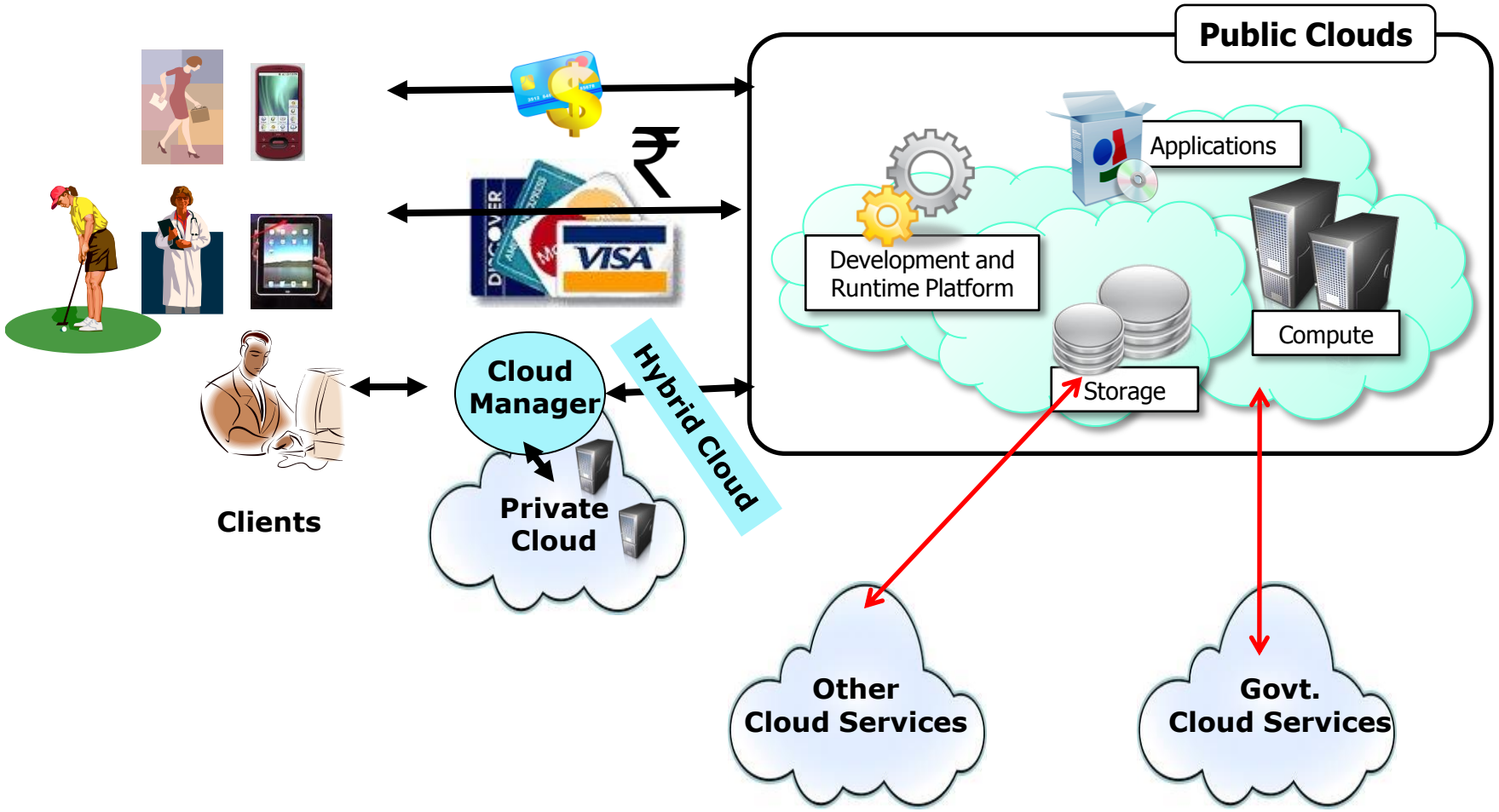
*** A public Cloud model within a company's own Data Center / infrastructure for internal and/or partners use.**

Hybrid/Inter Clouds

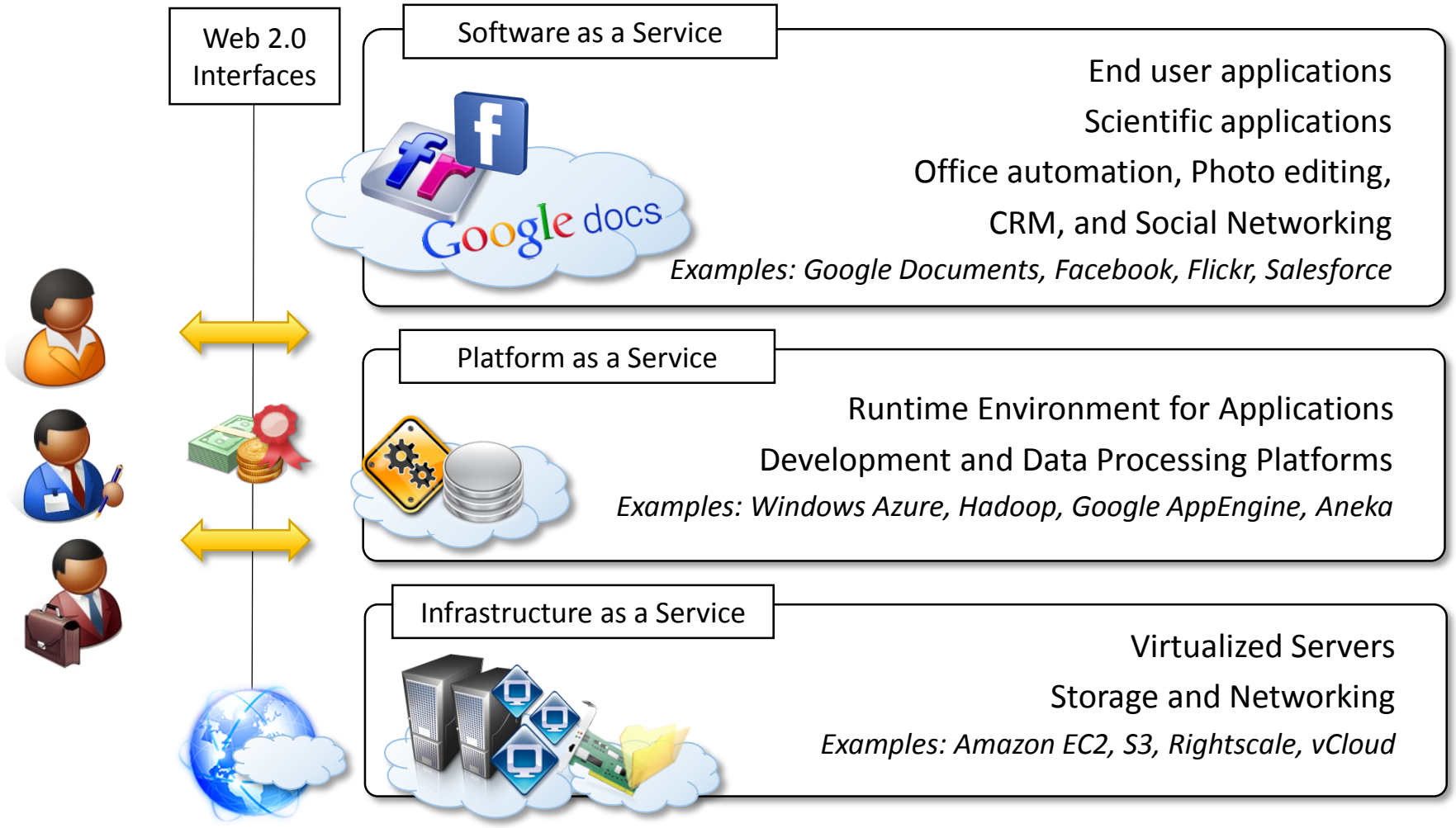
*** Mixed usage of private and public Clouds: Leasing public cloud services when private cloud capacity is insufficient**

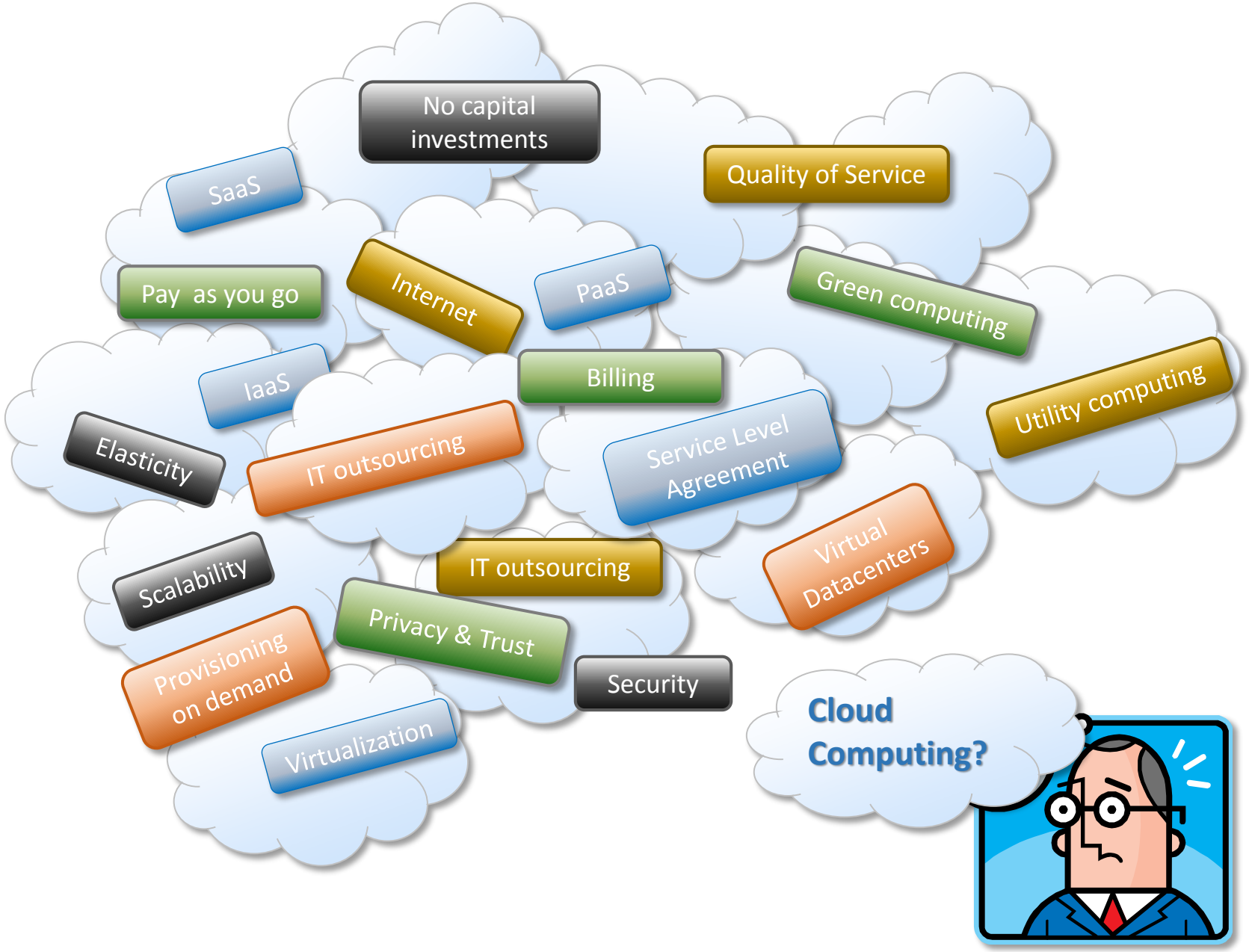


Subscription-Oriented Cloud Services:
X{compute, apps, data, ..}
as a Service (..aaS)



The cloud computing reference model




No capital investments
SaaS
Quality of Service
Pay as you go
Internet
PaaS
Green computing
IaaS
Billing
Utility computing
Elasticity
IT outsourcing
Service Level Agreement
Virtual Datacenters
Scalability
IT outsourcing
Privacy & Trust
Provisioning on demand
Security
Virtualization
Cloud Computing?

Parallel and Distributed Computing (I)

Overview

- Three major milestones have led to cloud computing evolution
  - Mainframes: Large computational facilities leveraging multiple processing units. Even though mainframes cannot be considered as distributed systems, they offered large computational power by using multiple processors, which were presented as a single entity to users.
  - Clusters: An alternative technological advancement to the use of mainframes and super computers.
  - Grids
  - Clouds

Eras of Computing

- Two fundamental and dominant models of computing are ***sequential*** and ***parallel***.
  - The sequential era began in the 1940s,
  - and Parallel(and distributed) computing era followed it within a decade.
- Four key elements of computing developed during these eras are
  - Architecture
  - Compilers
  - Applications
  - Problem solving environments
- The computing era started with development in ***hardware architectures***, which actually enabled the creation of ***system software*** – particularly in the area of **compilers and operating systems** – which support the management of such systems and the development of ***applications***.

Principles of Parallel and Distributed Computing

- The term parallel computing and distributed computing are **often used interchangeably**, even though they mean **slightly different things**.
- The term **parallel implies a tightly coupled system**, where as **distributed systems refers to a wider class of system, including those that are tightly coupled**.
- More precisely, the term **parallel computing** refers to a model in which **the computation is divided among several processors sharing the same memory**.
- The architecture of **parallel computing system** is often characterized by the **homogeneity of components**: **each processor is of the same type and it has the same capability as the others.**

Principles of Parallel and Distributed Computing Contd…

- The shared memory has a single address space, which is accessible to all the processors.
- Parallel programs are then broken down into several units of execution that can be allocated to different processors and can communicate with each other by means of shared memory.
- Originally parallel systems are considered as those architectures that featured multiple processors sharing the same physical memory and that were considered a single computer.
  - Over time, these restrictions have been relaxed, and parallel systems now include all architectures that are based on the concept of shared memory, whether this is physically present or created with the support of libraries, specific hardware, and a highly efficient networking infrastructure.
  - For example: a cluster of which of the nodes are connected through an InfiniBand network and configured with distributed shared memory system can be considered as a parallel system.

Principles of Parallel and Distributed Computing Contd…

- The term distributed computing encompasses any architecture or system that allows the computation to be broken down into units and executed concurrently on different computing elements, whether these are processors on different nodes, processors on the same computer, or cores within the same processor.
- Distributed computing includes a wider range of systems and applications than parallel computing and is often considered a more general term.
- Even though it is not a rule, the term distributed often implies that the locations of the computing elements are not the same and such elements might be heterogeneous in terms of hardware and software features.
- Classic examples of distributed computing systems are
  - Computing Grids
  - Internet Computing Systems

Elements of Parallel computing

- Silicon-based processor chips are reaching their physical limits. Processing speed is constrained by the speed of light, and the density of transistors packaged in a processor is constrained by thermodynamics limitations.
- A viable solution to overcome this limitation is to connect multiple processors working in coordination with each other to solve “Grand Challenge” problems.
- The first step in this direction led
  - To the development of parallel computing, which encompasses techniques, architectures, and systems for performing multiple activities in parallel.
  - As discussed earlier, the term parallel computing has blurred its edges with the term distributed computing and is often used in place of later term.

What is Parallel Processing?

- Processing of multiple tasks simultaneously on multiple processors is called ***parallel processing***.
- The parallel program consists of multiple active processes (tasks) simultaneously solving a given problem.
- A given task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different central processing unit (CPU).
- Programming on multi processor system using the divide-and-conquer technique is called ***parallel programming***.
- Many applications today require more computing power than a traditional sequential computer can offer.
- Parallel Processing provides a cost effective solution to this problem by increasing the number of CPUs in a computer and by adding an efficient communication system between them.
- The workload can then be shared between different processors. This setup results in higher computing power and performance than a single processor a system offers.

Parallel Processing influencing factors

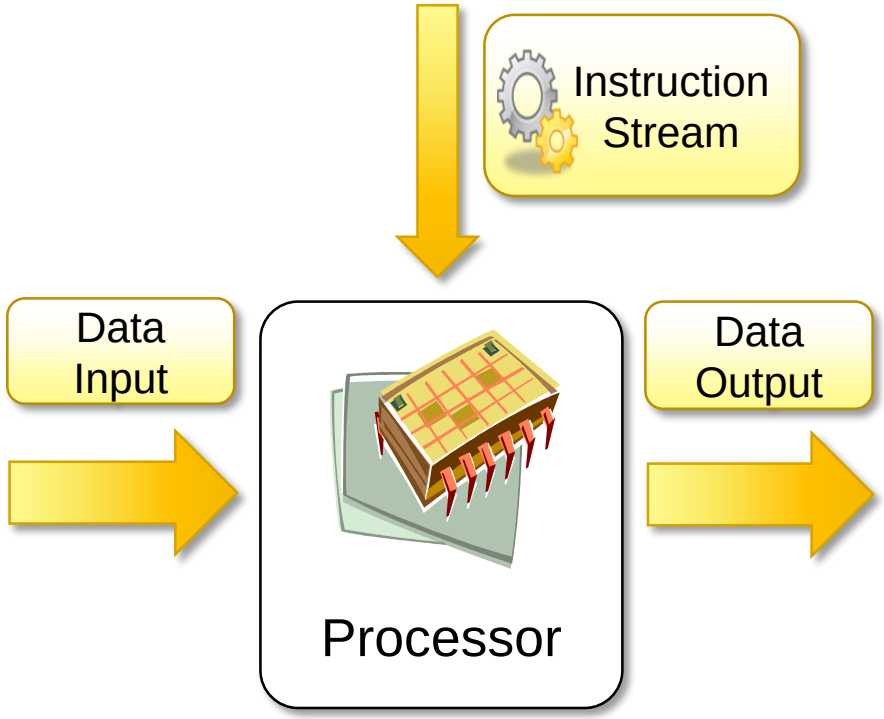
- The development of parallel processing is being influenced by many factors. The prominent among them include the following:
  - Computational requirements are ever increasing in the areas of both scientific and business computing. The technical computing problems, which require high-speed computational power, are related to
    - life sciences, aerospace, geographical information systems, mechanical design and analysis etc.
  - Sequential architectures are reaching mechanical physical limitations as they are constrained by the speed of light and thermodynamics laws.
    - The speed which sequential CPUs can operated is reaching saturation point (no more vertical growth), and hence an alternative way to get high computation speed is to connect multiple CPUs (opportunity for horizontal growth).
  - Hardware improvements in pipelining , super scalar, and the like are non scalable and require sophisticated compiler technology.
    - Developing such compiler technology is a difficult task.
  - Vector processing works well for certain kinds of problems. It is suitable mostly for scientific problems (involving lots of matrix operations) and graphical processing.
    - It is not useful for other areas, such as databases.
  - The technology of parallel processing is mature and can be exploited commercially
    - here is already significant R&D work on development tools and environments.
  - Significant development in networking technology is paving the way for
    - heterogeneous computing.

Hardware architectures for parallel Processing

- The core elements of parallel processing are CPUs. Based on the number of instructions and data streams, that can be processed simultaneously, computing systems are classified into the following four categories:
  - Single-instruction, Single-data (SISD) systems
  - Single-instruction, Multiple-data (SIMD) systems
  - Multiple-instruction, Single-data (MISD) systems
  - Multiple-instruction, Multiple-data (MIMD) systems

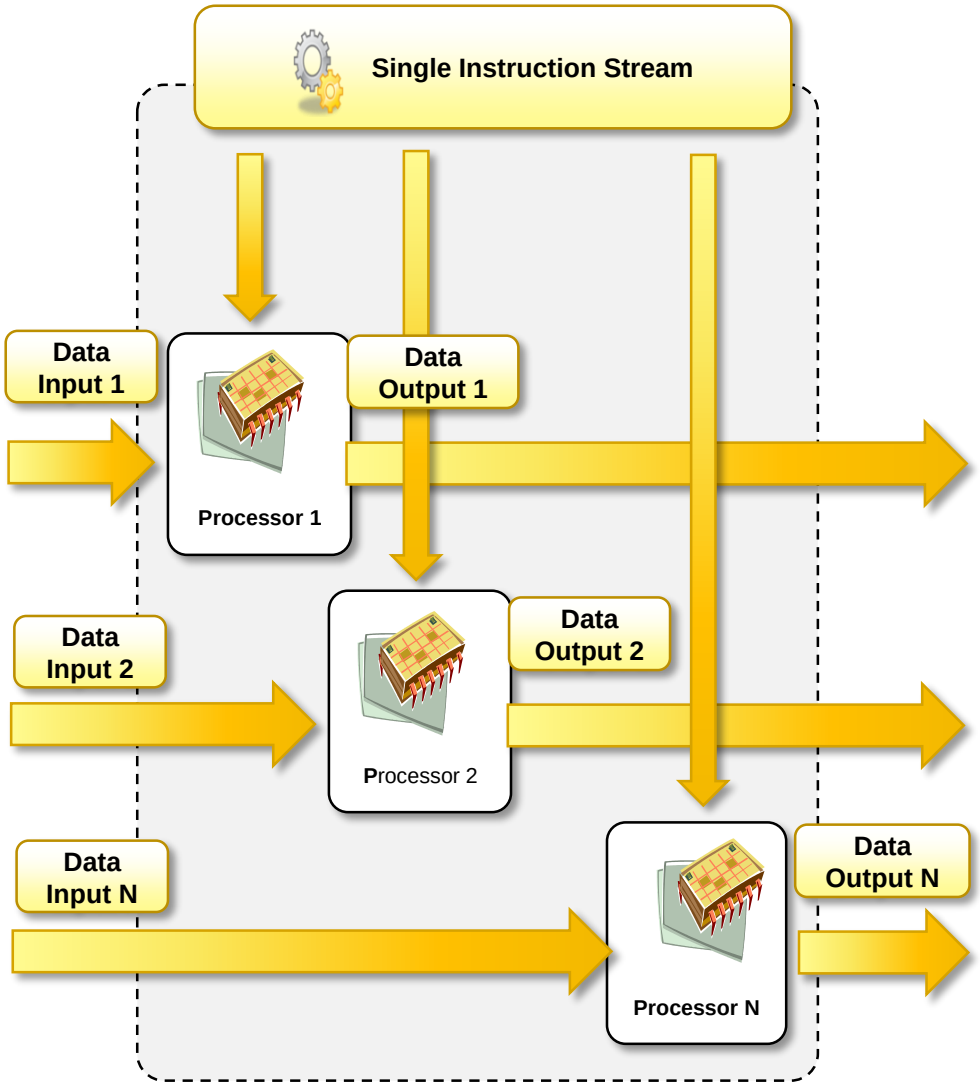
Single – Instruction , Single Data (SISD) systems

- SISD computing system is a uni-processor machine capable of executing a single instruction, which operates on a single data stream.
- Machine instructions are processed sequentially, hence computers adopting this model are popularly called sequential computers.
- Most conventional computers are built using SISD model.
- All the instructions and data to be processed have to be stored in primary memory.
- The speed of processing element in the SISD model is limited by the rate at which the computer can transfer information internally.
- Dominant representative SISD systems are IBM PC, Macintosh, and workstations.



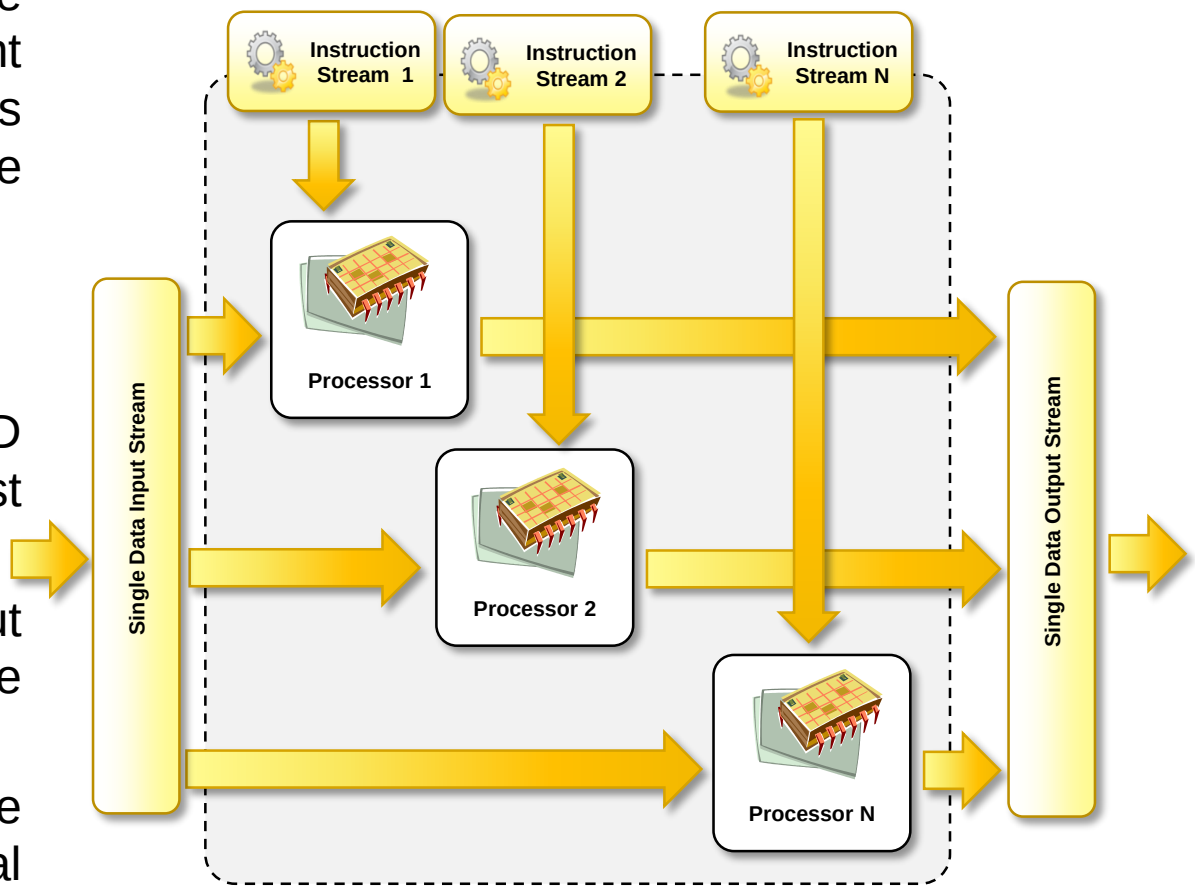
Single – Instruction , Multiple Data (SIMD) systems

- SIMD computing system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.
- Machines based on this model are well suited for scientific computing since they involve lots of vector and matrix operations.
- For instance statement Ci = Ai * Bi, can be passed to all the processing elements (PEs), organized data elements of vectors A and B can be divided into multiple sets (N- sets for N PE systems), and each PE can process one data set.
- Dominant representative SIMD systems are Cray's Vector processing machine and Thinking Machines Cm*, and GPGPU accelerators.



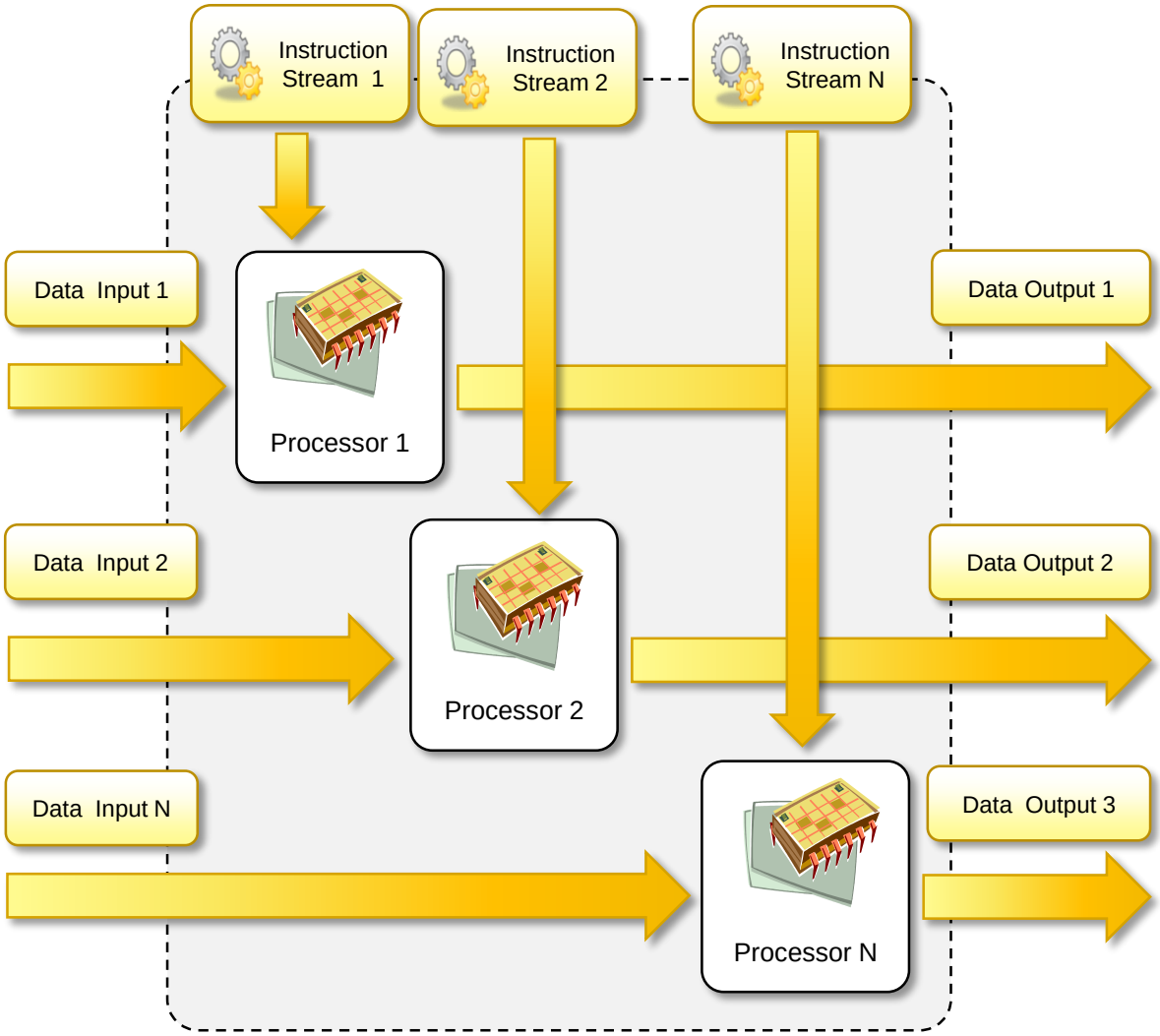
Multiple – Instruction , Single Data (MISD) systems

- MISD computing system is a multi processor machine capable of executing different instructions on different Pes all of them operating on the same data set.
- For example
  - y = sin(x) + cos(x) + tan(x)
- Machines built using MISD model are not useful in most of the applications.
- Few machines are built but none of them available commercially.
- This type of systems are more of an intellectual exercise than a practical configuration.



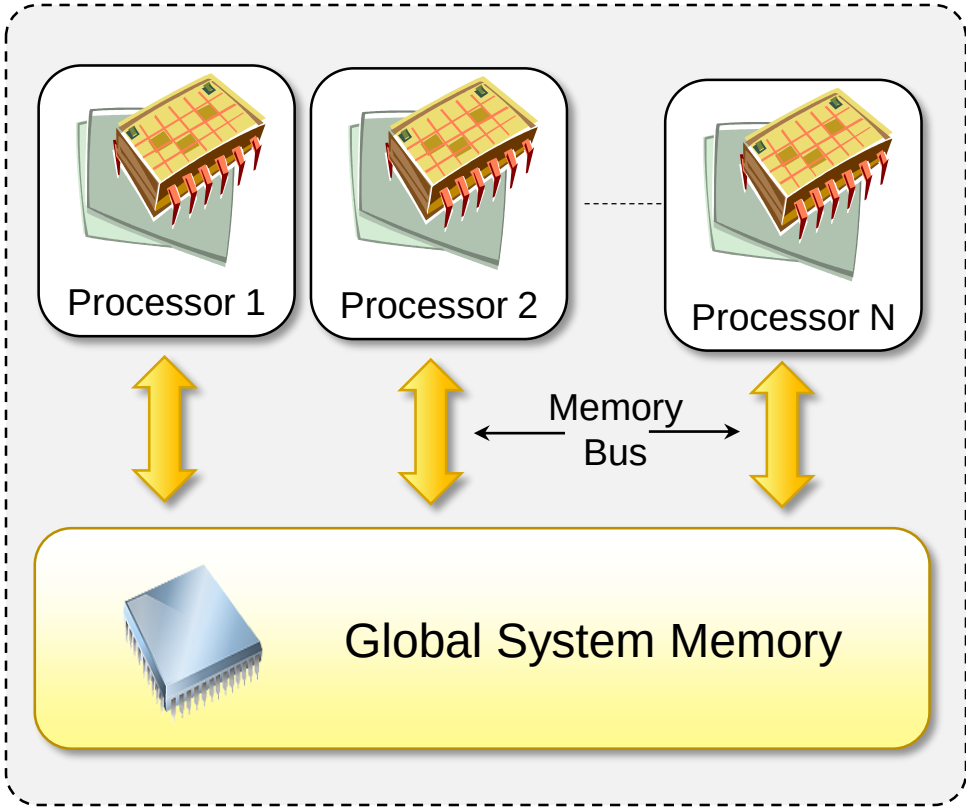
Multiple – Instruction , Multiple Data (MIMD) systems

- MIMD computing system is a multi processor machine capable of executing multiple instructions on multiple data sets.
- Each PE in the MIMD model has separate instruction and data streams, hence machines built using this model are well suited to any kind of application.
- Unlike SIMD, MISD machine, PEs in MIMD machines work asynchronously,
- MIMD machines are broadly categorized into shared-memory MIMD and distributed memory MIMD based on the way PEs are coupled to the main memory.



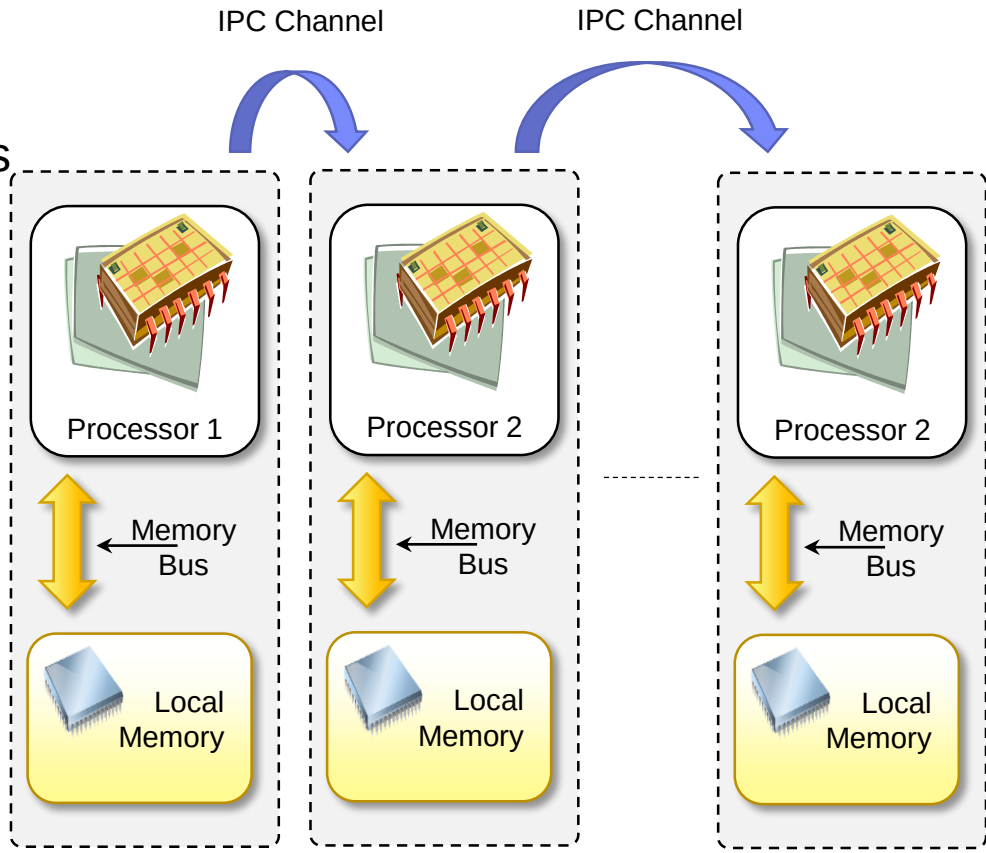
Shared Memory MIMD machines

- All the PEs are connected to a single global memory and they all have access to it.
- Systems based on this model are also called tightly coupled multi processor systems.
- The communication between PEs in this model takes place through the shared memory.
- Modification of the data stored in the global memory by one PE is visible to all other PEs.
- Dominant representative shared memory MIMD systems are silicon graphics machines and Sun/IBM SMP (Symmetric Multi-Processing).



Distributed Memory MIMD machines

- All PEs have a local memory. Systems based on this model are also called loosely coupled multi processor systems.
- The communication between PEs in this model takes place through the interconnection network, the inter process communication channel, or IPC.
- The network connecting PEs can be configured to tree, mesh, cube, and so on.
- Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them.



Shared Vs Distributed MIMD model

- The shared memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model.
- Failures, in a shared memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated.
- Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention.
- This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory.
- As a result, distributed memory MIMD architectures are most popular today.

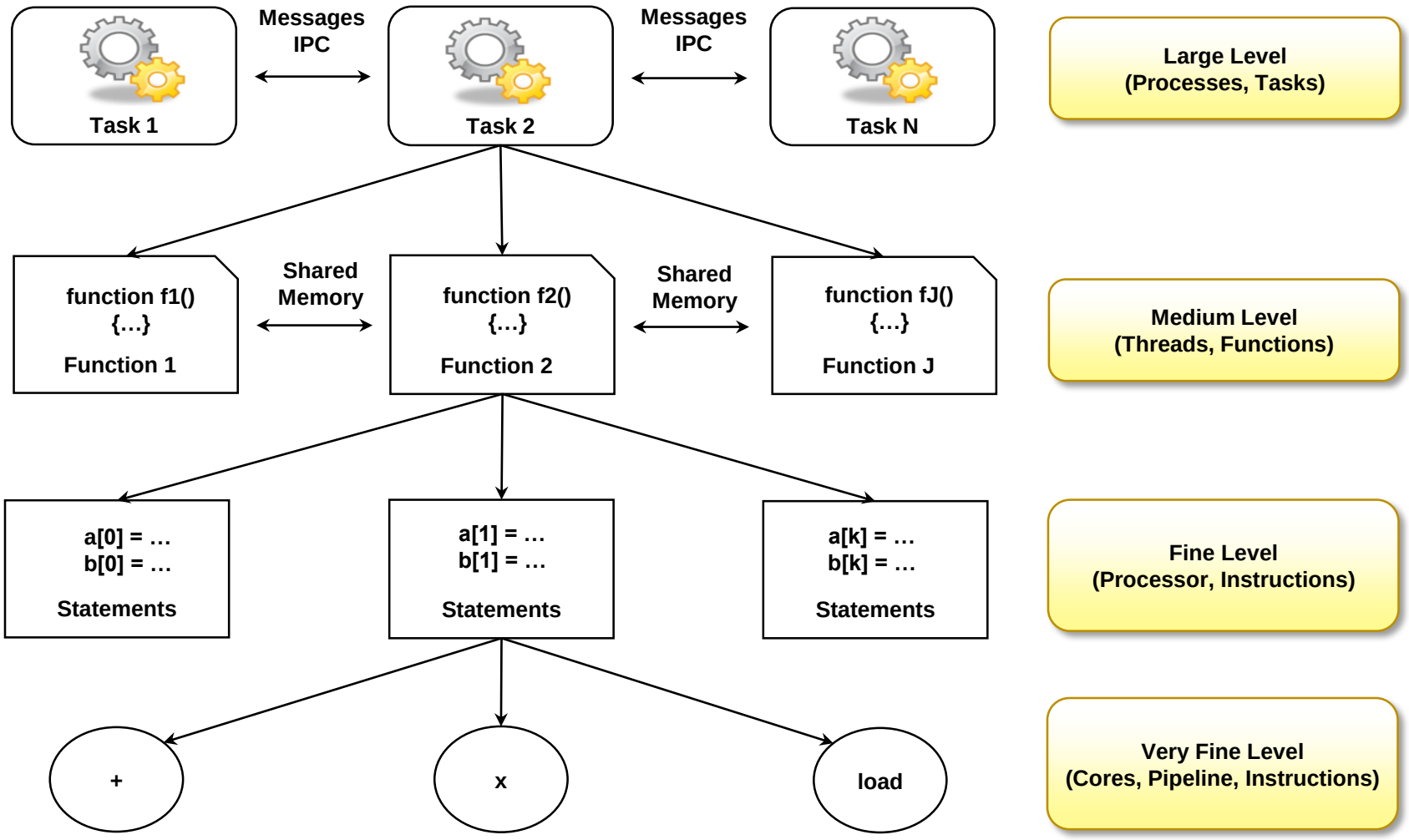
Approaches to Parallel Programming

- A sequential program is one that runs on a single processor and has a single line of control.
- To make many processors collectively work on a single program, the program must be divided into smaller independent chunks so that each processor can work on separate chunks of the problem.
- The program decomposed in this way is a parallel program.
- A wide variety of parallel programming approaches are available.

Approaches to Parallel Programming Contd…

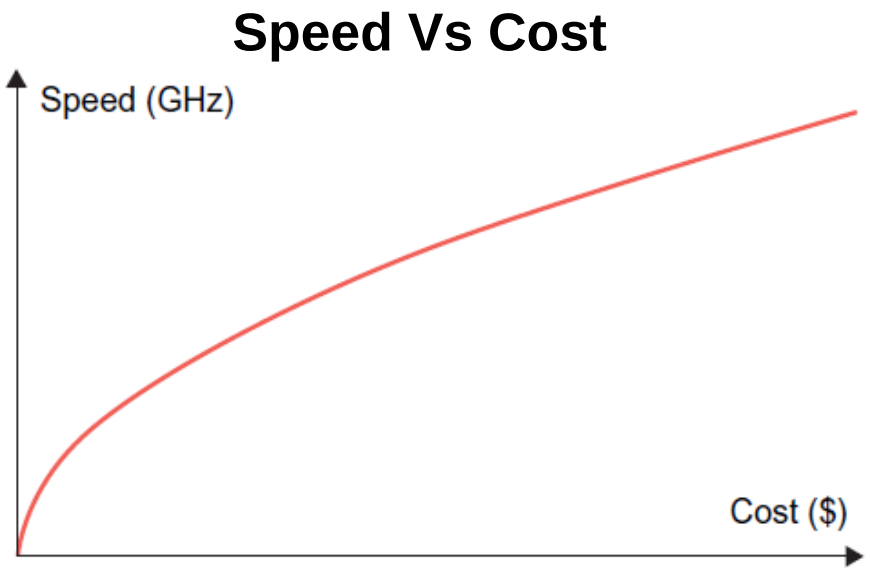
- The most prominent among them are the following.
  - Data Parallelism
  - Process Parallelism
  - Farmer-and-worker model
- The above said three models are suitable for task-level parallelism. In the case of data level parallelism, the divide-and-conquer technique is used to split data into multiple sets, and each data set is processed on different PEs using the same instruction.
- This approach is highly suitable to processing on machines based on the SIMD model.
- In the case of Process Parallelism, a given operation has multiple (but distinct) activities that can be processed on multiple processors.
- In the case of Farmer-and-Worker model, a job distribution approach is used, one processor is configured as master and all other remaining PEs are designated as slaves, the master assigns the jobs to slave PEs and, on completion, they inform the master, which in turn collects results.
- These approaches can be utilized in different levels of parallelism.

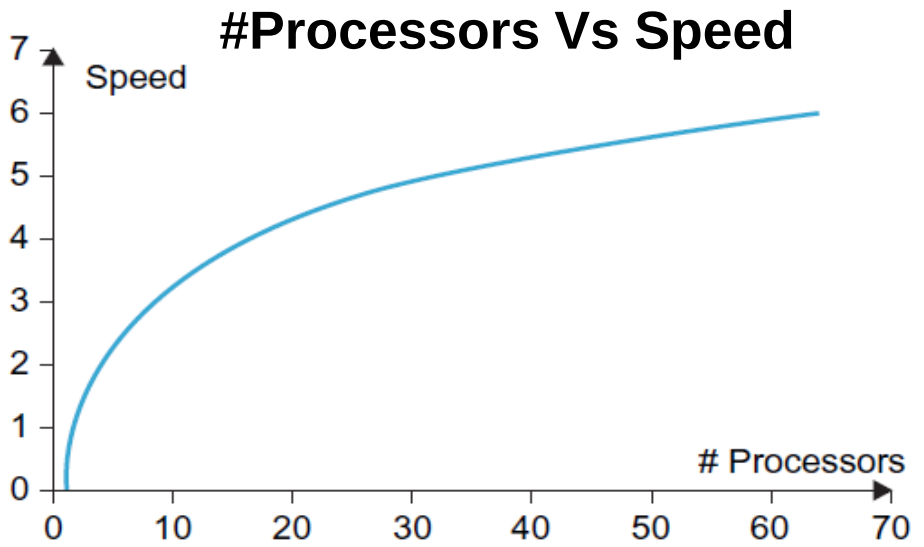
Levels of Parallelism



Laws of Caution

- Studying how much an application or a software system can gain from parallelism.
- In particular what need to keep in mind is that parallelism is used to perform multiple activities together so that the system can increase its throughput or its speed.
- But the relations that control the increment of speed are not linear.
- For example: for a given n processors, the user expects speed to be increase by in times. This is an ideal situation, but it rarely happens because of the communication overhead.
- Here two important guidelines to take into account.
  - Speed of computation is proportional to the square root of the system cost; they never increase linearly. Therefore, the faster a system becomes, the more expensive it is to increase its speed
  - Speed by a parallel computer increases as the logarithm of the number of processors (i.e. y=k*log(N)).





Parallel and Distributed Computing (II)

DISTRIBUTED SYSTEM

- A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

- E.g. Web Servers

The Strengths and Weaknesses of Distributed Computing

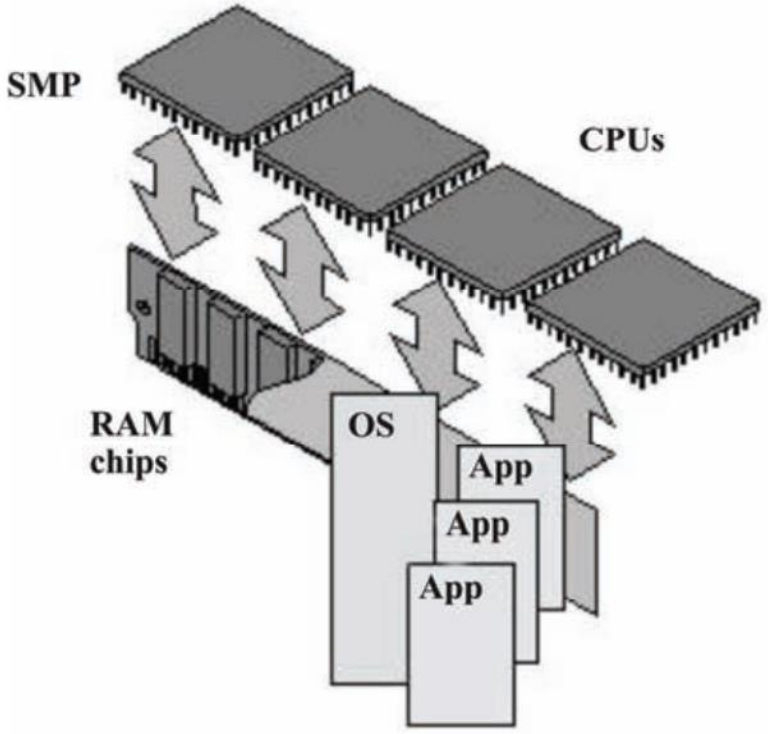
- Strengths
  - Affordability
  - Availability
  - Reliability
  - Resource Sharing
  - Scalability
- Weakness
  - Multiple points of failures
  - Security
  - Programming difficulty

Advancement in parallel processing

- Ability to perform CPU and I/O operation
- Multiprogramming
  - Multiple programs are allowed to use the processor for a short time
  - Round-robin scheduling
- Multiprocessing
  - Two or more processors share a common workload
  - Master/slave model
  - Vector processing
    - Well-formed data

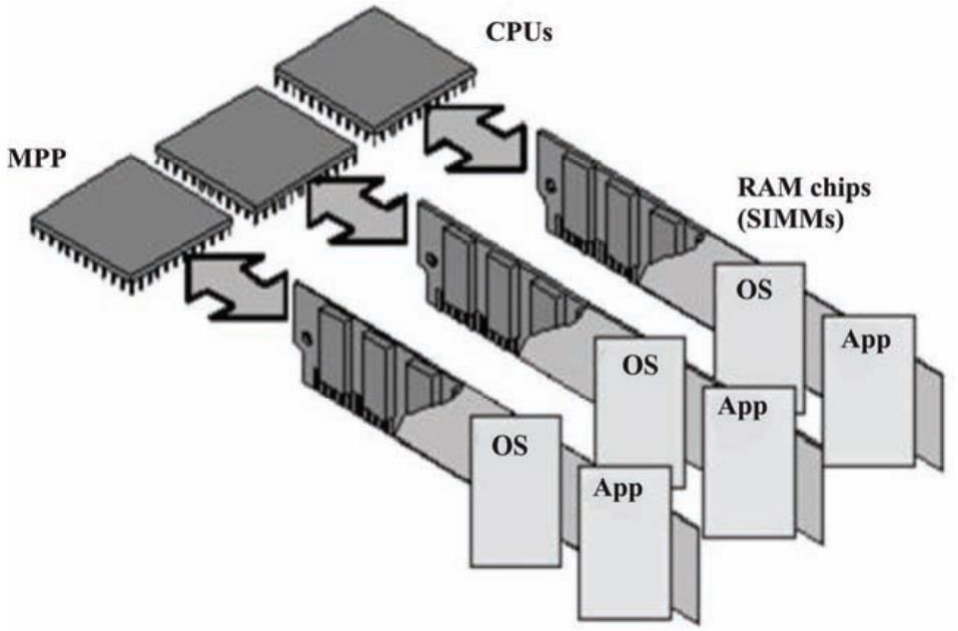
Advancement in parallel processing

- Symmetric Multiprocessing Systems
  - One of the CPUs boots the system and loads the SMP operating system, which brings the other CPUs online
  - Only one instance of the operating system and one instance of the application in memory
  - Multiple applications, Multiple threads



Advancement in parallel processing

- Massively Parallel Processing Systems
  - Each CPU contains its own memory and copy of the operating system and application.
  - Each subsystem communicates with the others via a high-speed interconnect.

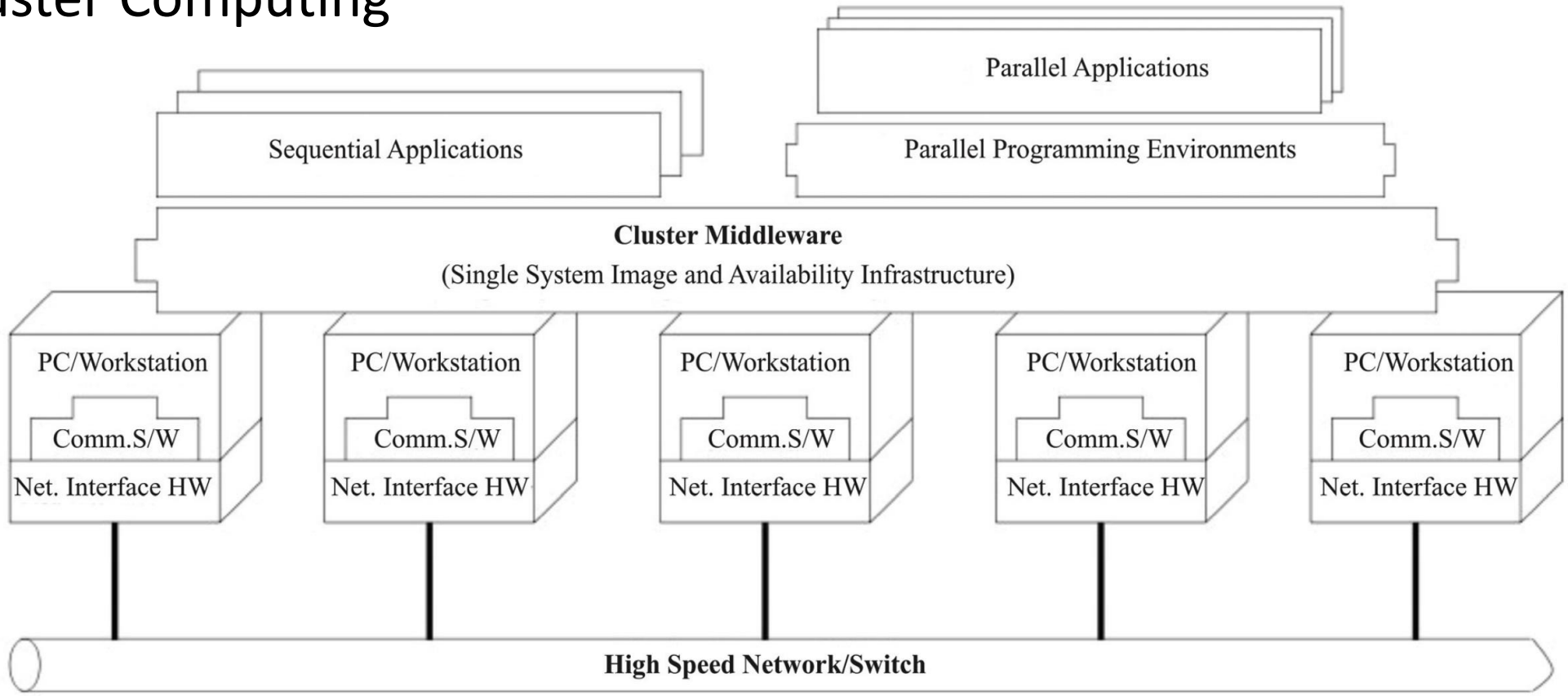


Advancement in parallel processing

- Cluster Computing
  - A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone/complete computers cooperatively working together as a single, integrated computing resource.
  - A node can be a single or multiprocessor system, such as PC, workstation, or SMP.
  - A cluster can be in a single cabinet or physically separated and connected via a LAN.
  - Typically a cluster will appear as a single system to users and applications.

Advancement in parallel processing

- Cluster Computing



Advancement in parallel processing

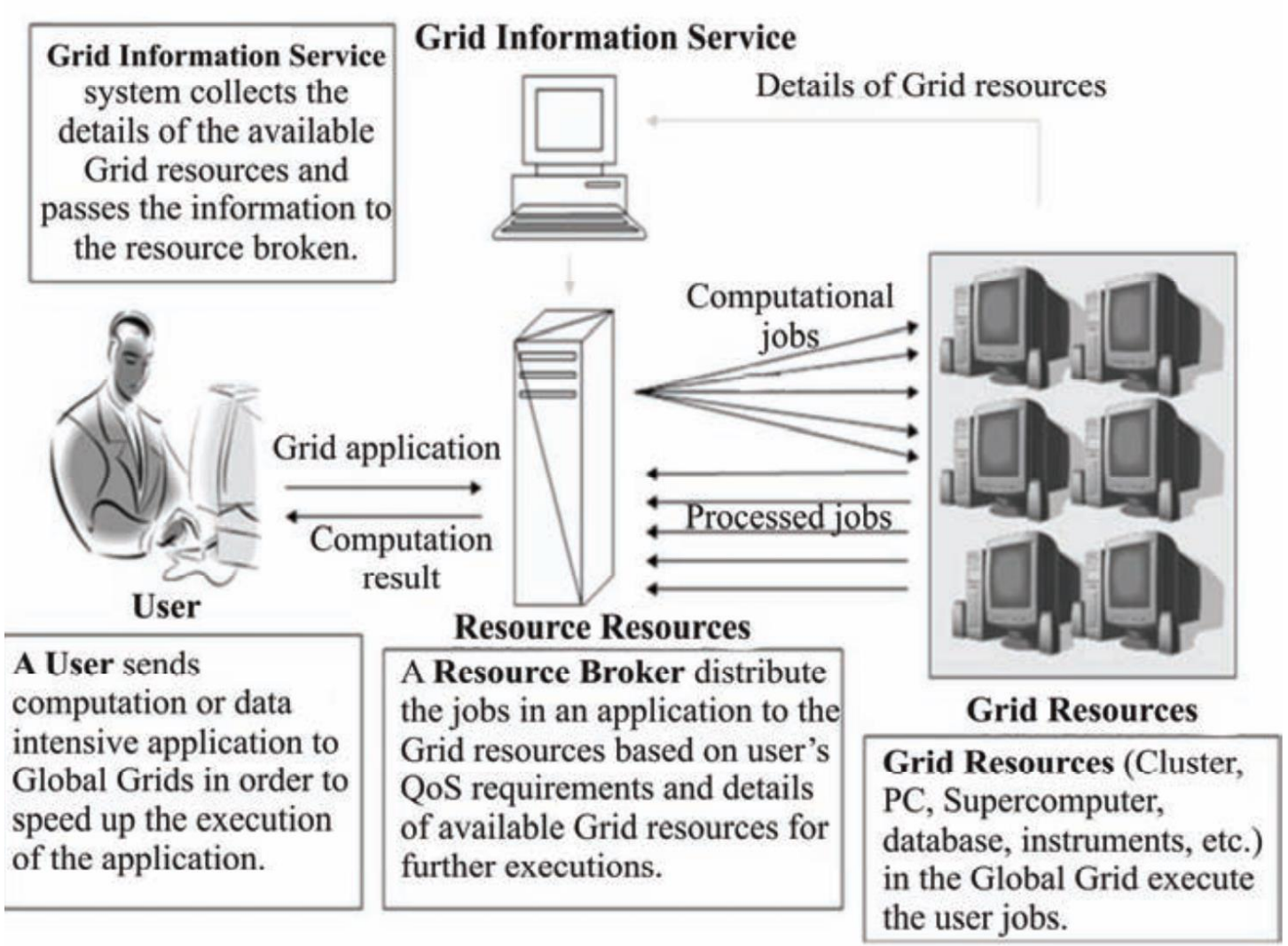
- Cluster Computing
  - Advantages: Easy to deploy and manage, Complete, Optimized, Expandable, etc.
  - Different kinds of Clusters
    - High Availability (HA) Clusters
      - Server Applications, Redundant Nodes
    - Load Balancing Clusters
    - High Performance (HP) Clusters

Advancement in parallel processing

- Grid Computing
  - A form of distributed computing whereby a "super and virtual computer" is composed of a cluster of networked, loosely coupled computers, acting in concert to perform very large tasks.
  - Cluster of clusters
  - Facilitates the executions of large-scale resource intensive applications on geographically distributed computing resources.

Advancement in parallel processing

- Grid Computing

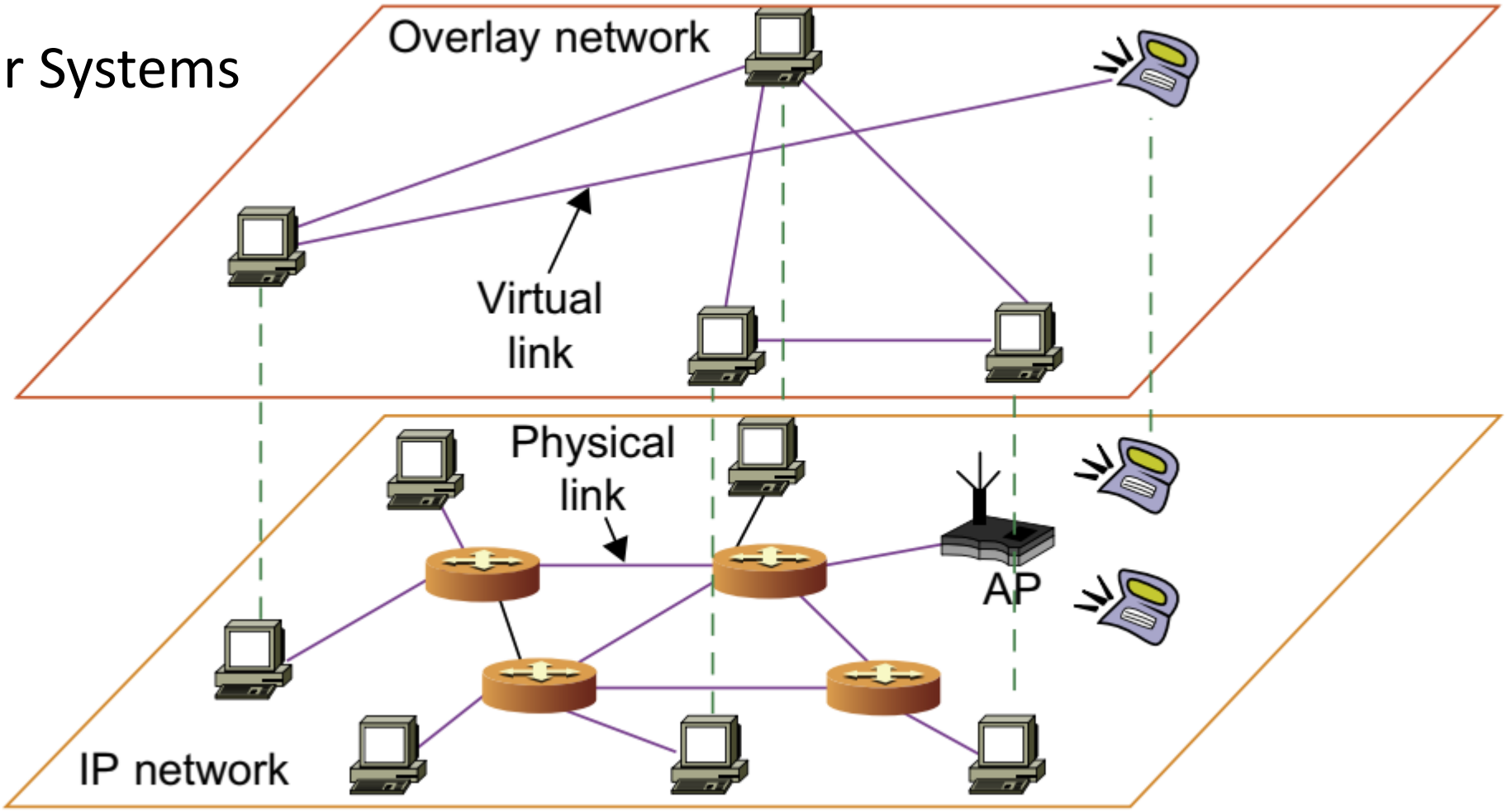


Advancement in parallel processing

- Peer to Peer Systems
  - Every node acts as both a client and a server, providing part of the system resources.
  - Peer machines are simply client computers connected to the Internet.
  - All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers.
  - No central coordination or central database is needed. In other words, no peer machine has a global view of the entire P2P system.
  - The system is self-organizing with distributed control.

Advancement in parallel processing

- Peer to Peer Systems



Advancement in parallel processing

- Peer to Peer Systems

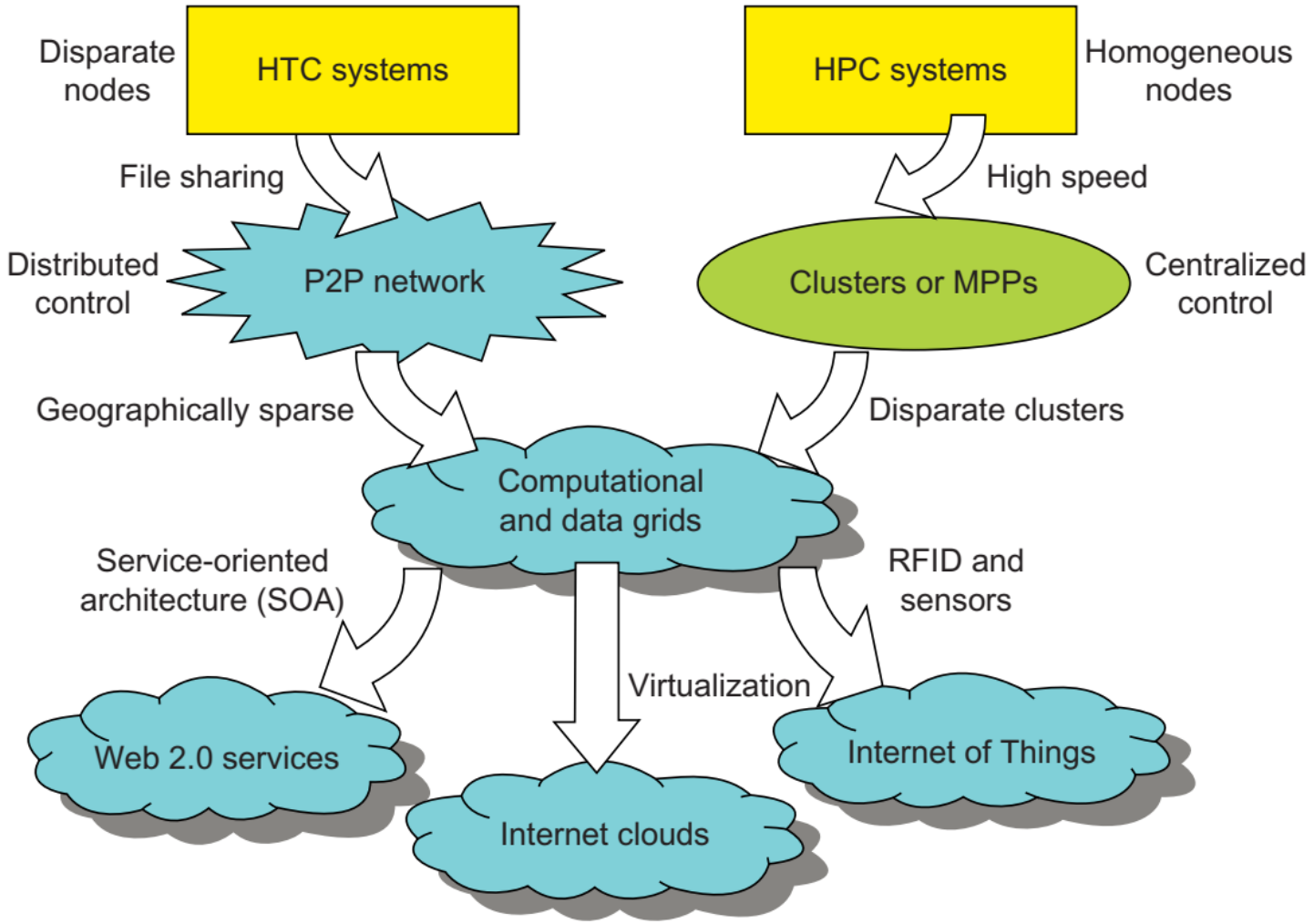
System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	P2P Platform
Attractive Applications	Content distribution of MP3 music, video, open software, etc.	Instant messaging, collaborative design and gaming	Scientific exploration and social networking	Open networks for public resources
Operational Problems	Loose security and serious online copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusion	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.

Advancement in parallel processing

- Virtualization
  - Cloud Computing

Functionality, Applications	Computer Clusters [10,28,38]	Peer-to-Peer Networks [34,46]	Data/ Computational Grids [6,18,51]	Cloud Platforms [1,9,11,12,30]
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 data centers via SLA
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 supercomputing, global problem solving, and data center 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

Evolution


Disparate nodes
HTC systems
HPC systems
Homogeneous nodes
File sharing
High speed
Distributed control
P2P network
Clusters or MPPs
Centralized control
Geographically sparse
Disparate clusters
Computational and data grids
Service-oriented architecture (SOA)
RFID and sensors
Virtualization
Web 2.0 services
Internet clouds
Internet of Things

Applications of High-Performance and High-Throughput Systems

Domain	Specific Applications
Science and engineering	Scientific simulations, genomic analysis, etc.
	Earthquake prediction, global warming, weather forecasting, etc.
Business, education, services 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, data centers, decision-making systems, etc.
	Traffic monitoring, worm containment, cyber security, etc.
	Digital government, online tax return processing, social networking, etc.
Mission-critical applications	Military command and control, intelligent systems, crisis management, etc.

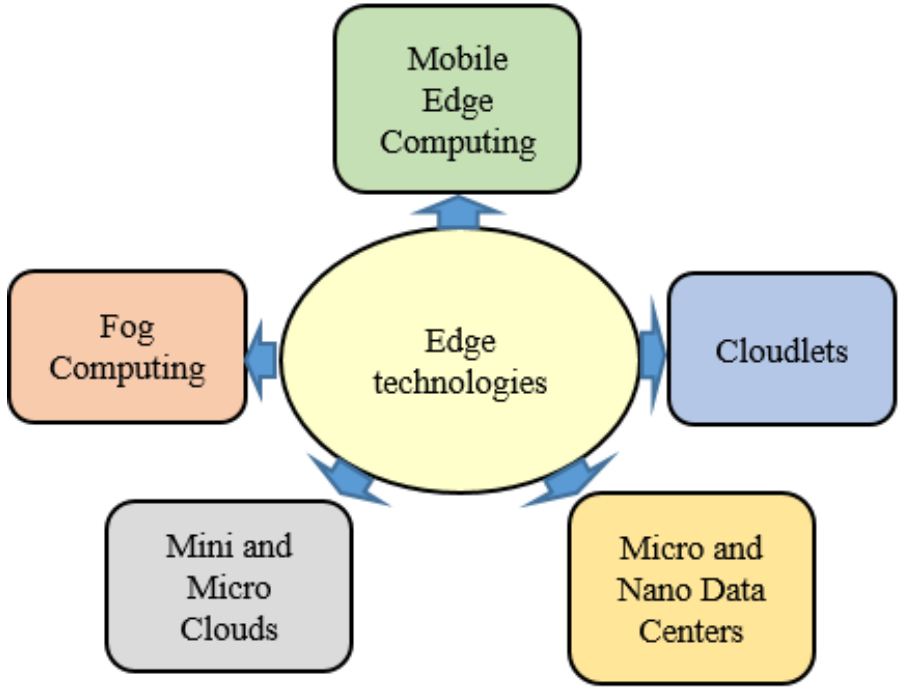
Distributed Edge Computing Paradigms

Is Cloud Computing Enough?

- By 2025, around 22 billion IoT devices will occupy the globe resulting in the number of connected devices per person as 6.58 [1].
- Around 2.8 billion intelligent and lightweight mobile devices are already connected to the internet through wireless communication networks [2].
- The amount of data traffic generated by these IoT devices has increased at an annual growth rate of 47% from 2016 to 2021 [3].
- Distance between cloud servers and IoT devices
  - Context-aware processing and low-latency applications (LLAs)

Edge Computing is Solution

- New distributed computing paradigms have emerged that offer computing and storage services in the proximity of service users.
  - Edge Computing (EC),
  - Fog Computing (FC),
  - Multi-access Edge Computing (MEC),
  - Cloudlets,
  - Mini and Micro clouds,
  - Edge Clouds,
  - Micro Data Centers (MDC),
  - Nano Data Centers (NDC) etc.



Some Motivational Use Cases

- Computation offloading
  - Compute-intensive compression tasks
  - Compute-intensive encryption
  - Compute-intensive tasks
  - Video storage and analysis
  - Data filtering
- Low latency
  - Three types of latencies are generally involved:
    - Communication: data transfer rate of the network
    - Propagation: total propagation distance
    - Computation: computation capacity

Some Motivational Use Cases

- Context-awareness
  - Contexts: “any information that can be used to characterize the situation of an entity. An entity is a person, place, piece of software, software service or object that is considered relevant to the interaction between a user and an application, including the user and application themselves” [4].
  - Context-awareness: “the ability of a system to provide relevant information or services to users using context information where relevance depends on the user’s task” [4].
  - For a large number of mobile users and IoT devices, processing a huge amount of this contextual information is challenging.
- Optimal Energy Consumption
  - Extend the battery life of end-user devices by offloading the computation tasks

Some Motivational Use Cases

- Reduced load on Cloud
  - Location and activity logging applications such as Nike+, Foursquare, Runtastic, Runkeeper, and Endomondo are very popular and used by millions of users.
  - Users' daily activities are aggregated by these applications, which execute on high-end smartphones coupled with many sensors, e.g. gyroscope, temperature sensor, motion sensor, accelerometer, GPS, etc.
  - This user's information, along with some other information such as user id, time, distance, speed, duration, longitude, latitude, weather, location, calorie burned, etc., is sent to the cloud for analytics.
  - Cumulative information, if counted for all the users, is huge.
  - A study on Endomondo reveals that the number of tuples generated by 30 million users could reach 25000 tuples/sec. This huge data congests the backbone network and overload the cloud [5].

Applications of Distributed Edge Computing

- Industry: *Actuation, Building Automation, Industrial Control, Structural Monitoring*
- Automotive: *Traffic Monitoring, Road Monitoring, Intelligent Transportation System, Vehicle-To-Vehicle Communications, Security*
- Energy: *Smart Metering, Smart Lighting, Data Collectors*
- Healthcare: *Patient Monitoring, Body Area Networks, Smart Health, Elderly Care, Telecare*
- Lifestyle: *Wearable Gadgets, Gaming, Augmented Reality, Fitness & Wellness*

Comparison of Distributed computing paradigms

	Fog	MEC	Cloudlet
Originally proposed by	CISCO	ETSI	[39]
Place of Operation	Networking (e.g. gateways, router, switches, access points) and specialized computing machines	Base stations, RAN in 4G and 5G, small scale data centers with virtualization capability	Dedicated small servers at one hop from user
Resource Access	Distributed	Hybrid	Hybrid
Hardware Connectivity	Cellular, Wi-Fi, WAN, WLAN, LAN	Cellular, WAN	Cellular, Wi-Fi, WAN, WLAN, LAN
Policy Manager	Service providers	Network Infrastructure providers	Network Infrastructure providers or cloud service providers
Resource Assignment	Shared or Virtualized	Shared	Dedicated
Computational Power	Medium	Limited	Ample
Cloud Communication	Seamless	Through Base Stations	Seamless

Comparison of Distributed computing paradigms

	Fog	MEC	Cloudlet
Mobility Support	Medium	Good	Limited
Architecture	Hierarchical/Decentralized	Hierarchical/Localized	Localized
Coverage	Ample	Ample	Limited
Service Access	Through Connected devices	At the edge of the internet	At the edge of the internet
Context-awareness	Yes	Yes	Could be
Hierarchy	3+-tiers	2-tiers	2-tiers
General Use Cases	IoT, Video surveillance, smart city/smart grid, smart delivery, smart healthcare, real-time subsurface imaging	Video caching and analytics, health monitoring, local video surveillance, traffic control, content delivery	face recognition at airports, speech recognition, augmented reality

References

1. Lueth, K.L.: State of the IoT 2018: Number of IoT devices now at 7B, https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/
2. Ren, J., Zhang, Y., Deng, R., Zhang, N., Zhang, D., Shen, X.S.: Joint channel access and sampling rate control in energy harvesting cognitive radio sensor networks. IEEE Trans. Emerg. Top. Comput. 7, 149–161 (2019). https://doi.org/10.1109/TETC.2016.2555806
3. Cisco, T.: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper. Growth Lakel. 2011, 2010–2015 (2011)
4. Abowd, G.D., Dey, A.K., Brown, P.J.,Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 304–307 (1999)
5. Cortés, R., Bonnaire, X., Marin, O., Sens, P.: Stream processing of healthcare sensor data: Studying user traces to identify challenges from a big data perspective. In: Procedia Computer Science. pp. 1004–1009 (2015)

Virtualization in Cloud Computing

Reference: Mastering Cloud Computing by Rajkumar Buyya et.al.

Virtualization

- Resource: A resource is a source or supply from which benefit is produced, typically of limited availability.
- Resources in computing: Physical or virtual entities of limited availability (e.g., memory, processing capacity, I/O devices etc.).
- Virtual: not physically existing as such but made by software to appear to do so
- Virtual Resource: An illusion supported by an OS through use of a real resource. An OS may use the same real resource to support several virtual resources. a virtual resource is an abstract view of a resource taken by a program.

- E.g. I/O Devices, Print Server, Virtual Memory, Virtual Machine

Introduction

- Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure-based services.

- **Virtualization refers to the creation of a virtual resource such as a server, desktop, operating system, file, storage or network.**

- The basis of this technology is the ability of a computer program—or a combination of software and hardware—to emulate an executing environment separate from the one that hosts such programs

- For example, we can run Windows OS on top of a virtual machine, which itself is running on Linux OS

- The IBM CP/CMS mainframes were the first systems to introduce the concept of hardware virtualization and hypervisors

- **Virtualization** is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether virtual hardware or an operating system—to run applications.

Confluence of several phenomena --

a. Increased performance and computing capacity.
b. Underutilized hardware and software resources
c. Lack of space.

   Server Consolidation

d. Greening initiatives

   Data centers are one of the major power consumers Rise of administrative costs

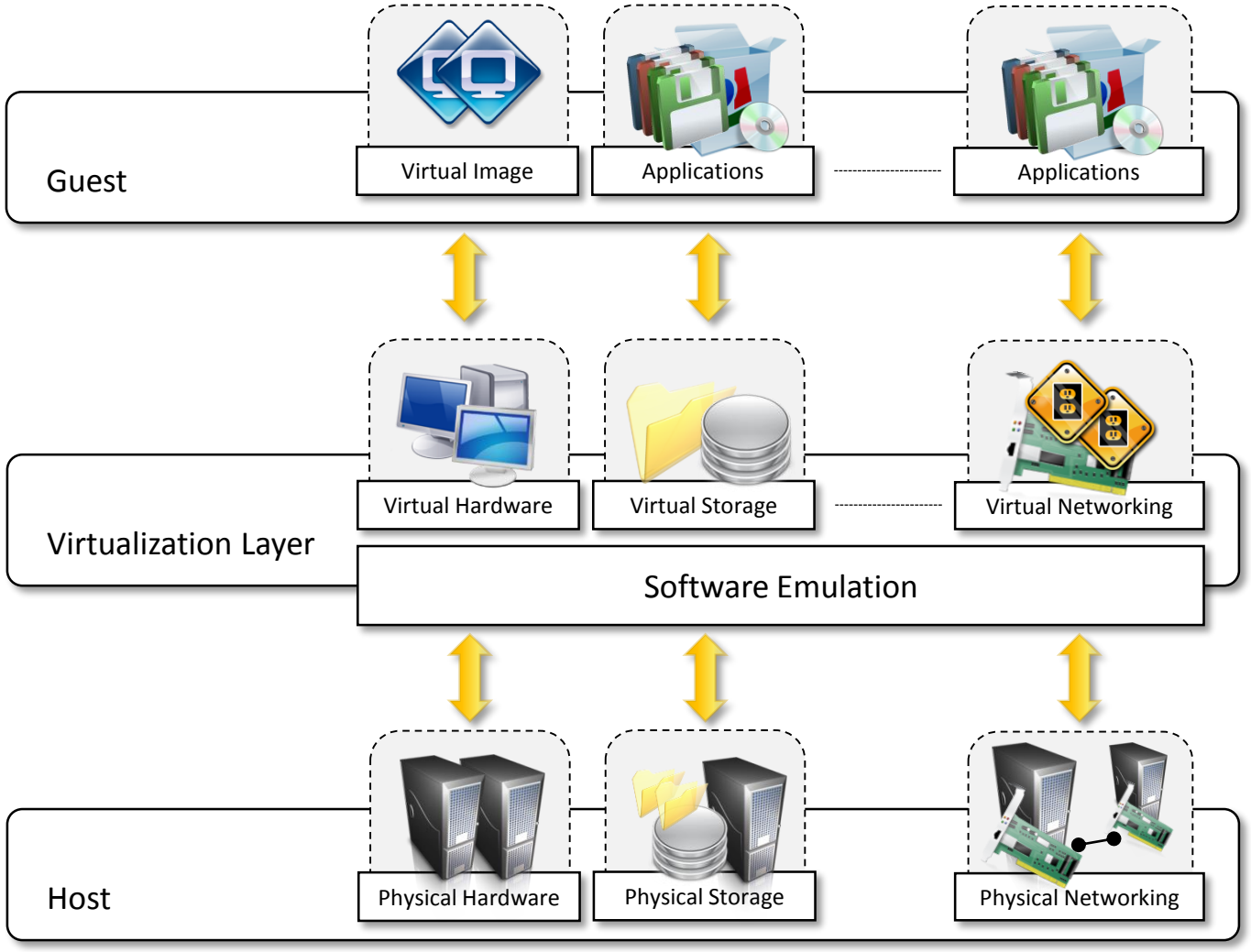
e. Rise of administrative costs.

   Power consumption and cooling costs have now become higher than the cost of IT equipment

- Java
- .NET Framework
- Python
- All the above three programming languages are based on the virtual machine model
- This trend of shifting toward virtualization from a programming language perspective demonstrated an important fact.
- The technology was ready to support virtualized solutions without a significant performance overhead.

Characteristics of virtualized environments

- Refers to the creation of a virtual version of something, whether hardware, a software environment, storage, or a network.
- three major components: guest, host, and virtualization layer
- hardware virtualization
- virtual machine manager
- virtual private network

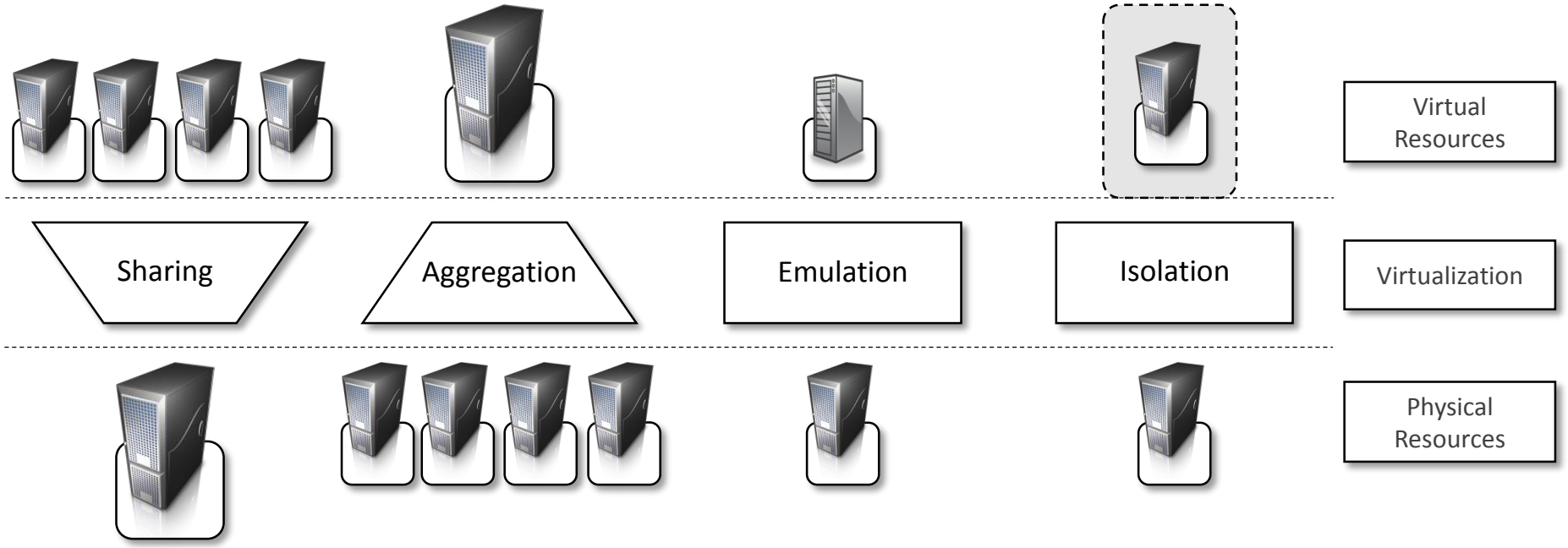


The virtualization reference model.

Advantantages of Virtualization

- Increased security
- Managed execution
  - Sharing: Higher CPU Utilization
  - Aggregation: Server Consolidation
  - Emulation: Imitating a system
  - Isolation: Using Abstraction layer
- Portability: e.g. Java Byte Code

Advantantages of Virtualization



Functions enabled by managed execution.

Emulation

- In computing, an **emulator** is hardware or software that enables one computer system (called the *host*) to behave like another computer system (called the *guest*)

- Many printers, for example, are designed to emulate Hewlett-Packard LaserJet printers because so much software is written for HP printers.

- If a non-HP printer emulates an HP printer, any software written for a real HP printer will also run in the non-HP printer emulation and produce equivalent printing.

Emulation vs Simulation

- Simulation = For *analysis* and *study*
- Emulation = For usage as a *substitute*
- A simulator is an environment which models but an emulator is one that replicates the usage as on the original device or system
- If a flight-simulator could transport you from A to B then it would be a flight-emulator
- An emulator can replace the original for *real* use
- A Virtual PC emulates a PC.
- An emulator will always have to operate close to real-time. For a simulator that is not always the case

Virtualization

- JVM (Java Virtual Machine): A engine that provides runtime environment to drive the Java Code or applications.
- JVM converts Java bytecode into machines language.
- JVM is a part of JRE(Java Run Environment).
- In other programming languages the compiler produces machine code for a particular system. However, Java compiler produces code for a Virtual Machine known as Java Virtual Machine.

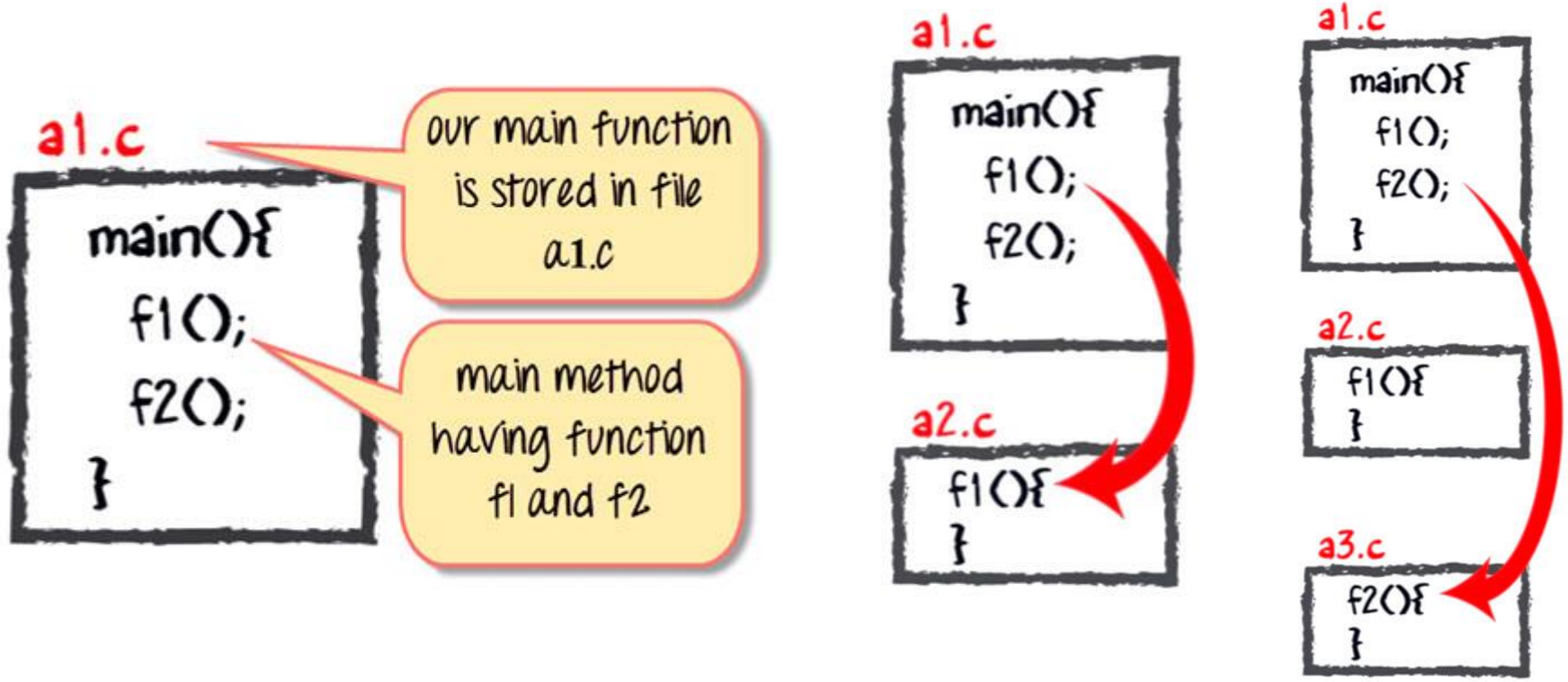
Virtualization

Software Code Compilation & Execution process: In order to write and execute a software program, you need the following

- Editor – To type your program into, a notepad could be used for this
- Compiler – To convert your high language program into native machine code
- Linker – To combine different program files reference in your main program together.
- Loader – To load the files from your secondary storage device like Hard Disk, Flash Drive, CD into RAM for execution. The loading is automatically done when you execute your code.
- Execution – Actual execution of the code which is handled by your OS & processor.

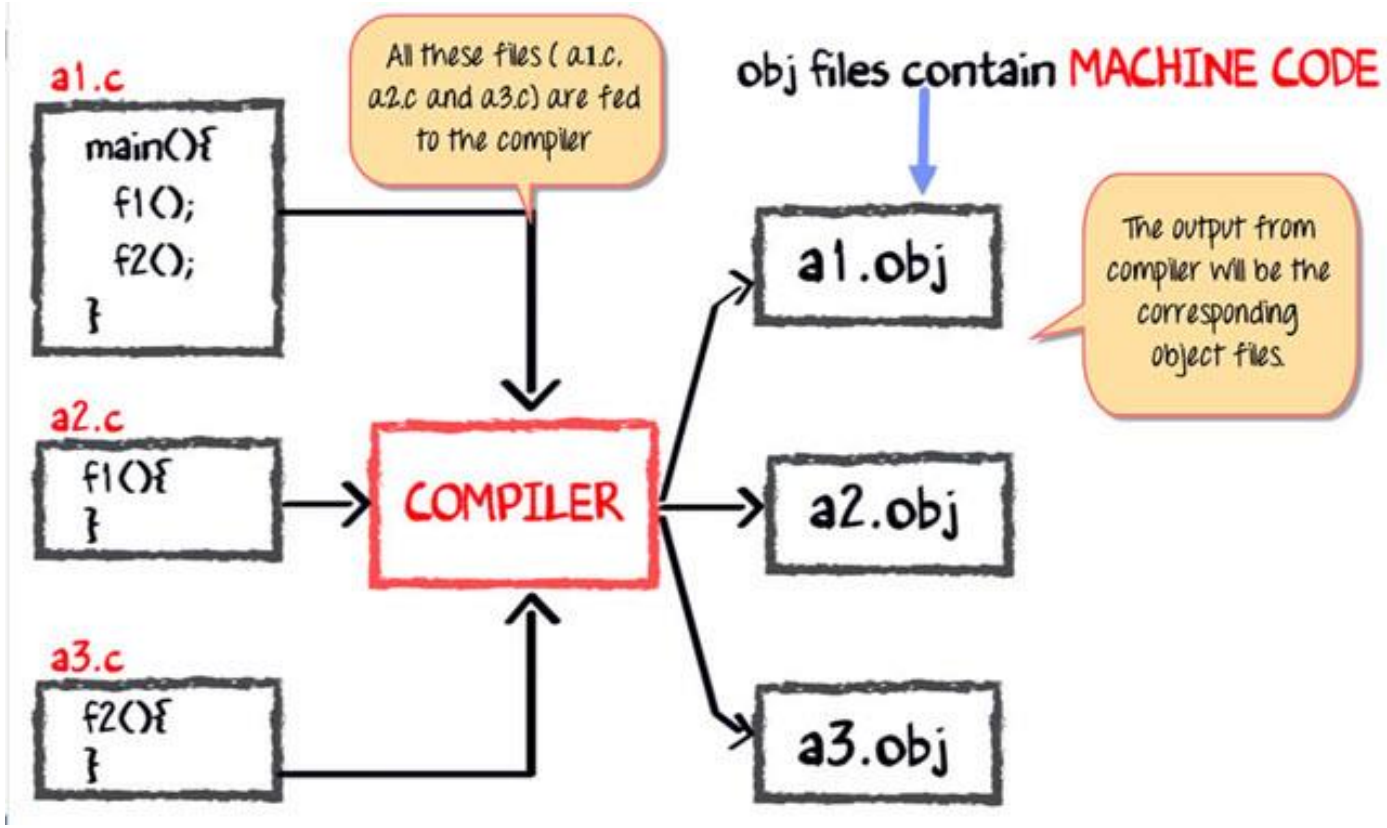
Virtualization

C code Compilation and Execution process:



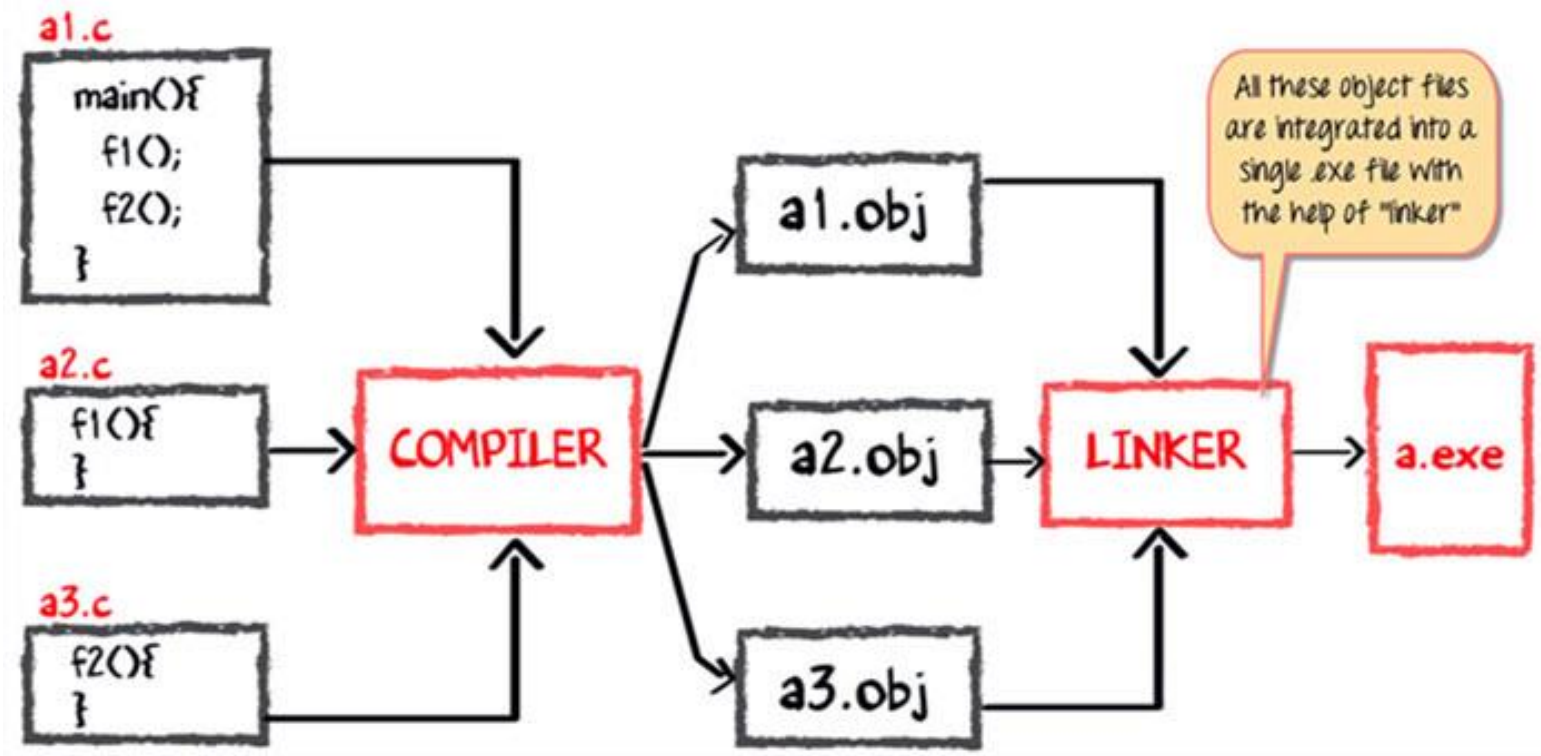
Virtualization

C code Compilation and Execution process:



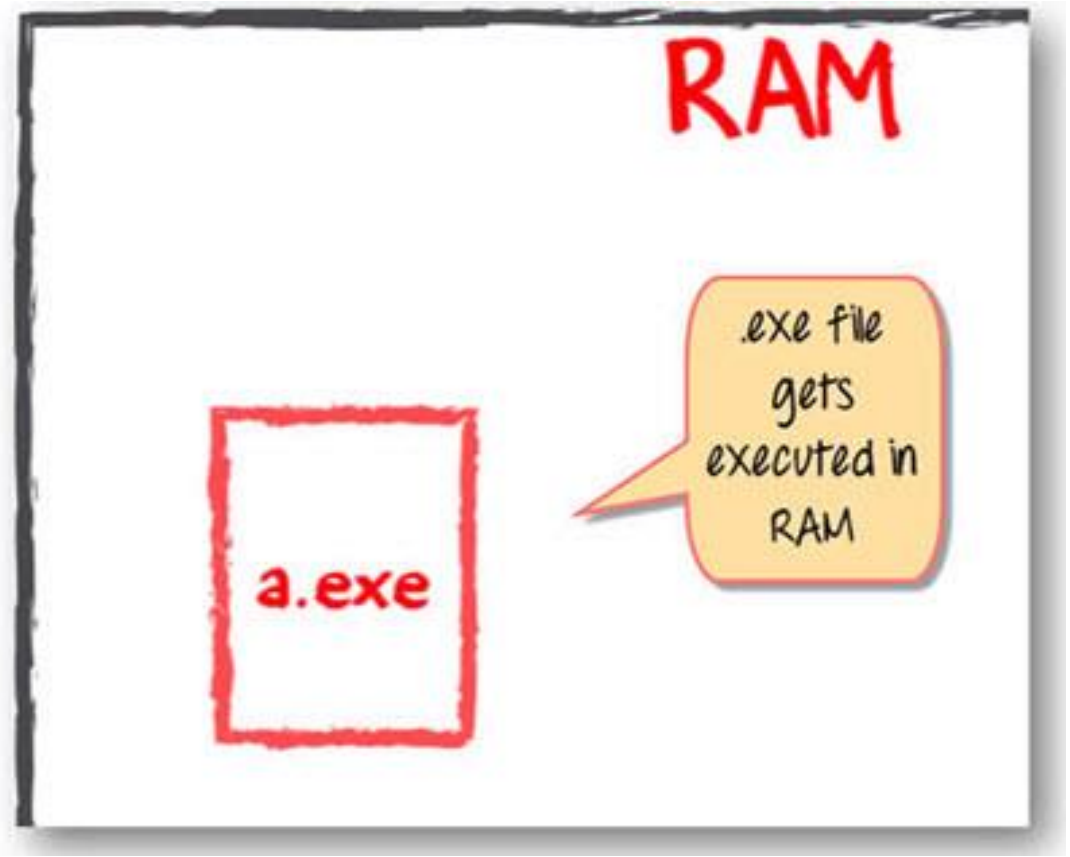
Virtualization

C code Compilation and Execution process:



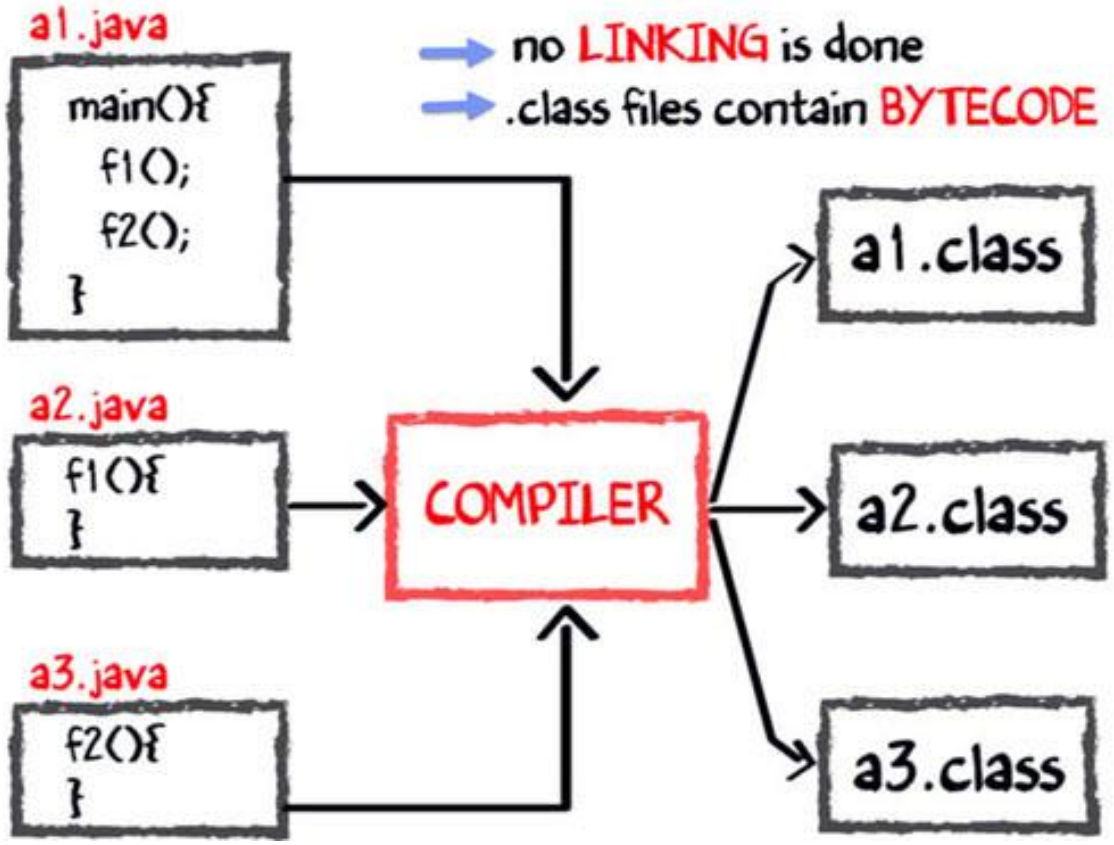
Virtualization

C code Compilation and Execution process:



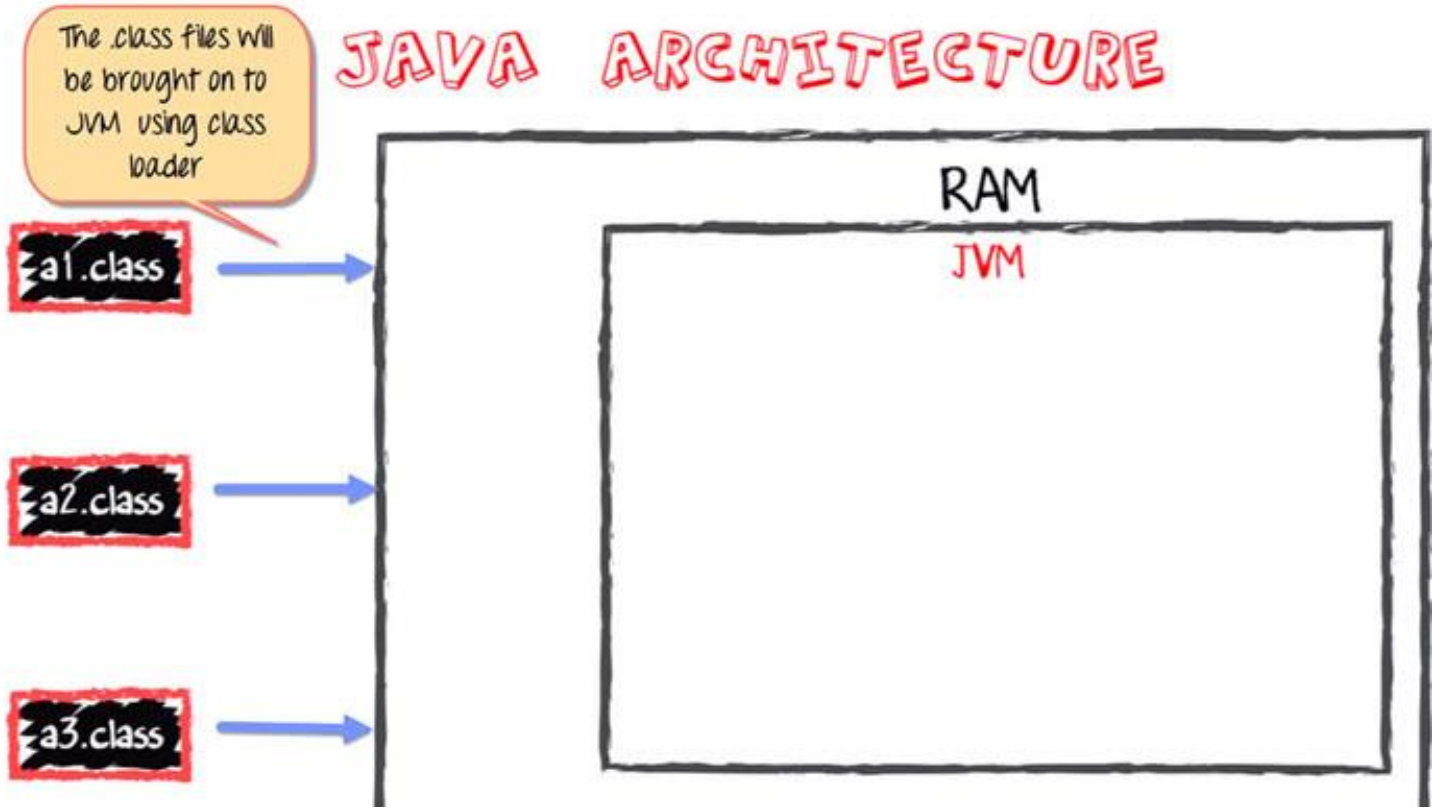
Virtualization

Java code Compilation and Execution in Java VM:



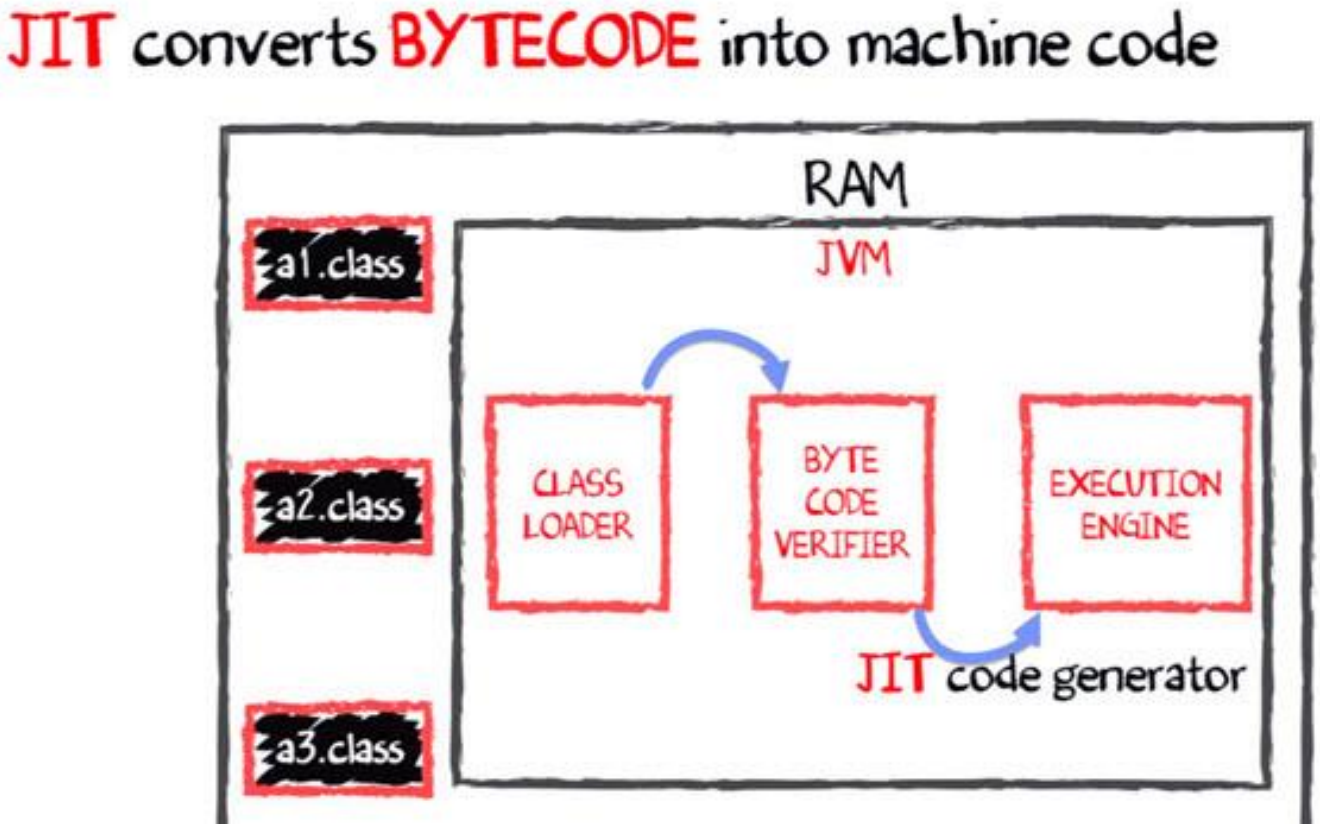
Virtualization

Java code Compilation and Execution in Java VM:

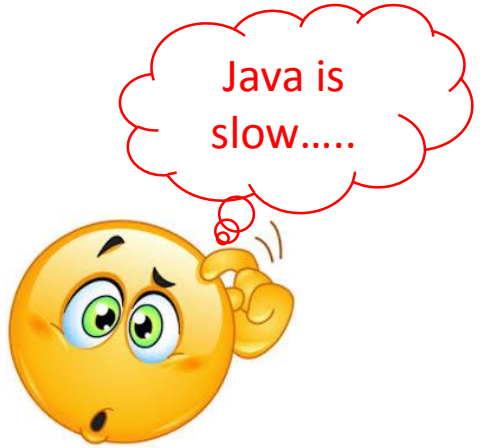


Virtualization

Java code Compilation and Execution in Java VM:



Virtualization

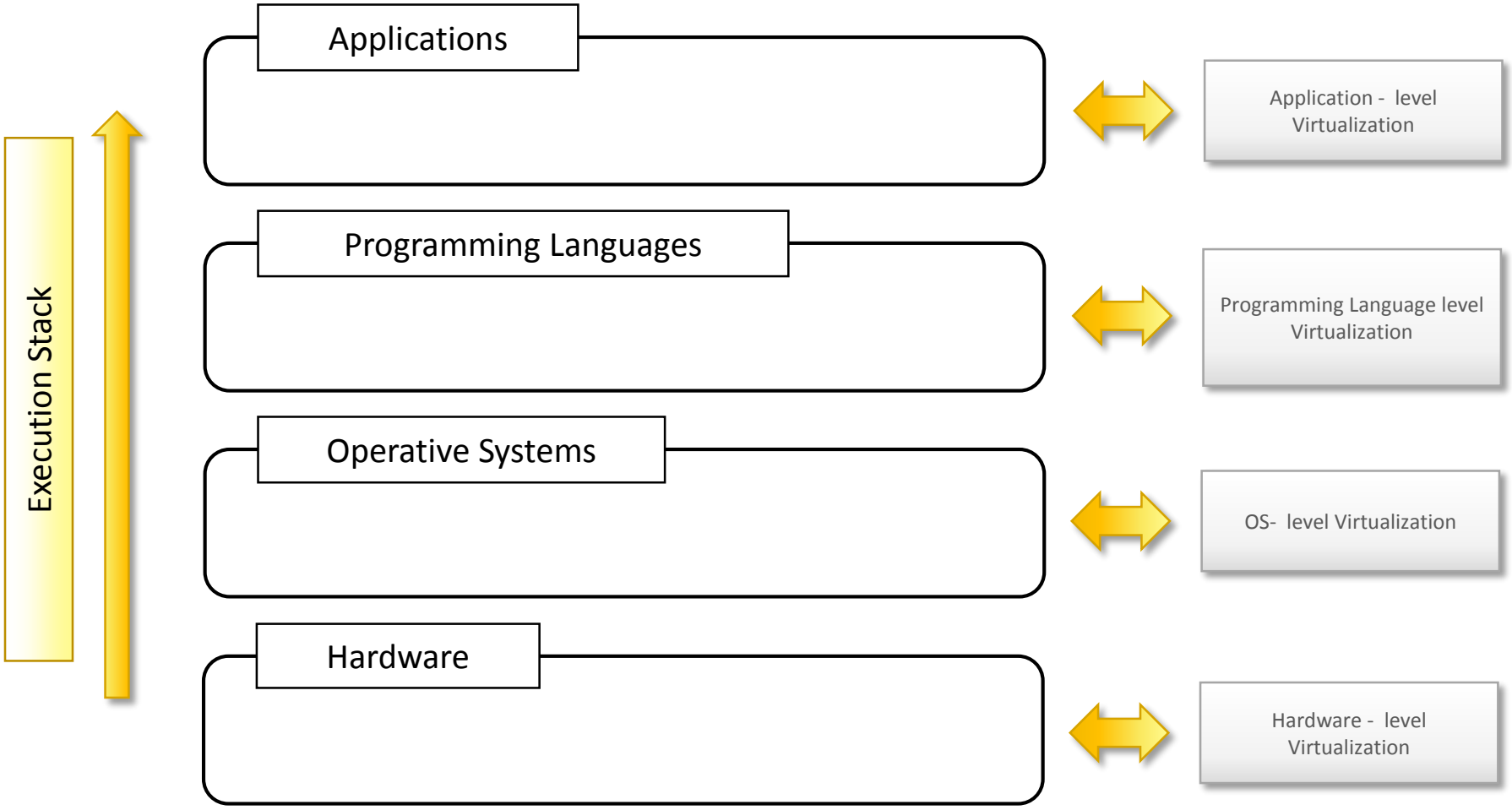


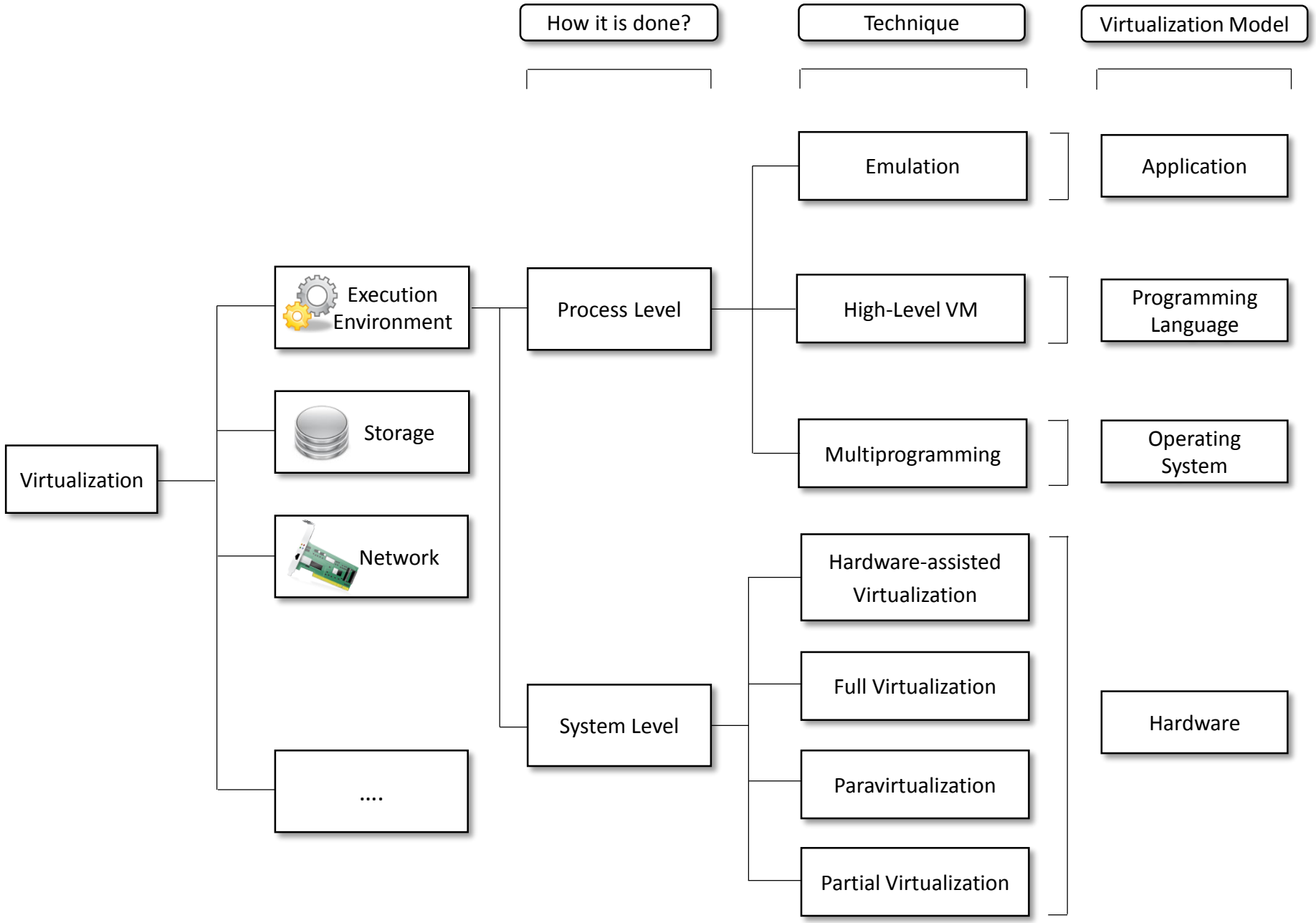
Dynamic Linking: Unlike C, linking is done at run-time, every time the program is run in Java.

Run-time Interpreter: The conversion of byte code into native machine code is done at run-time in Java which furthers slows down the speed

Note: Latest version of Java has addressed the performance bottlenecks to a great extent

Taxonomy of Virtualization

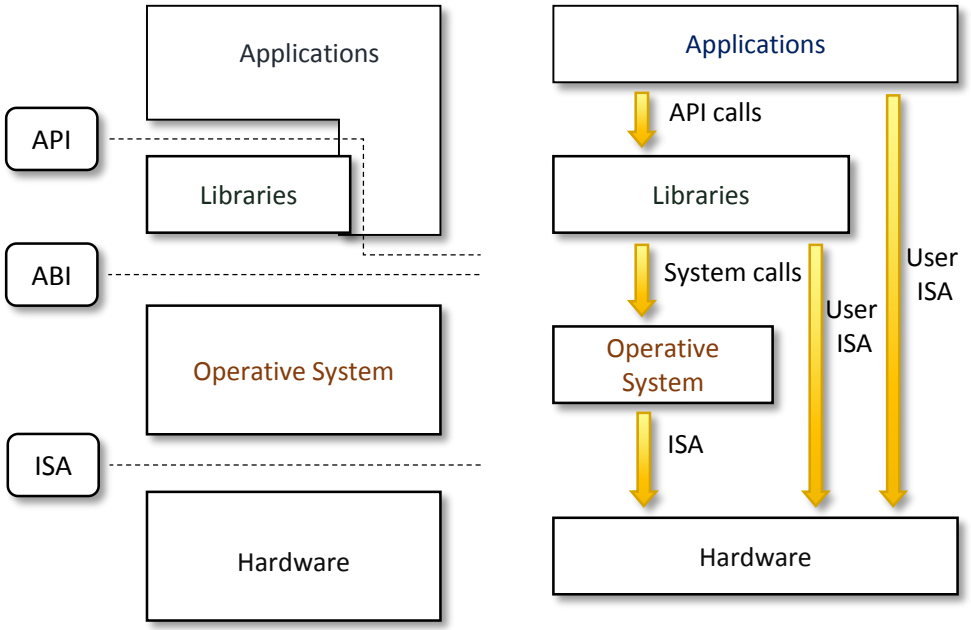




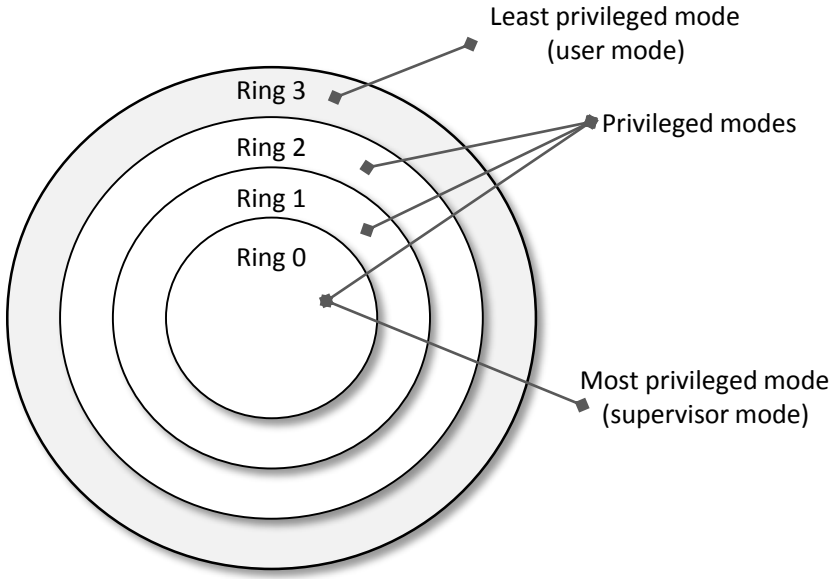
Taxonomy of Virtualization

Taxonomy of Virtualization

- Execution virtualization
  - Machine reference model



A machine reference model

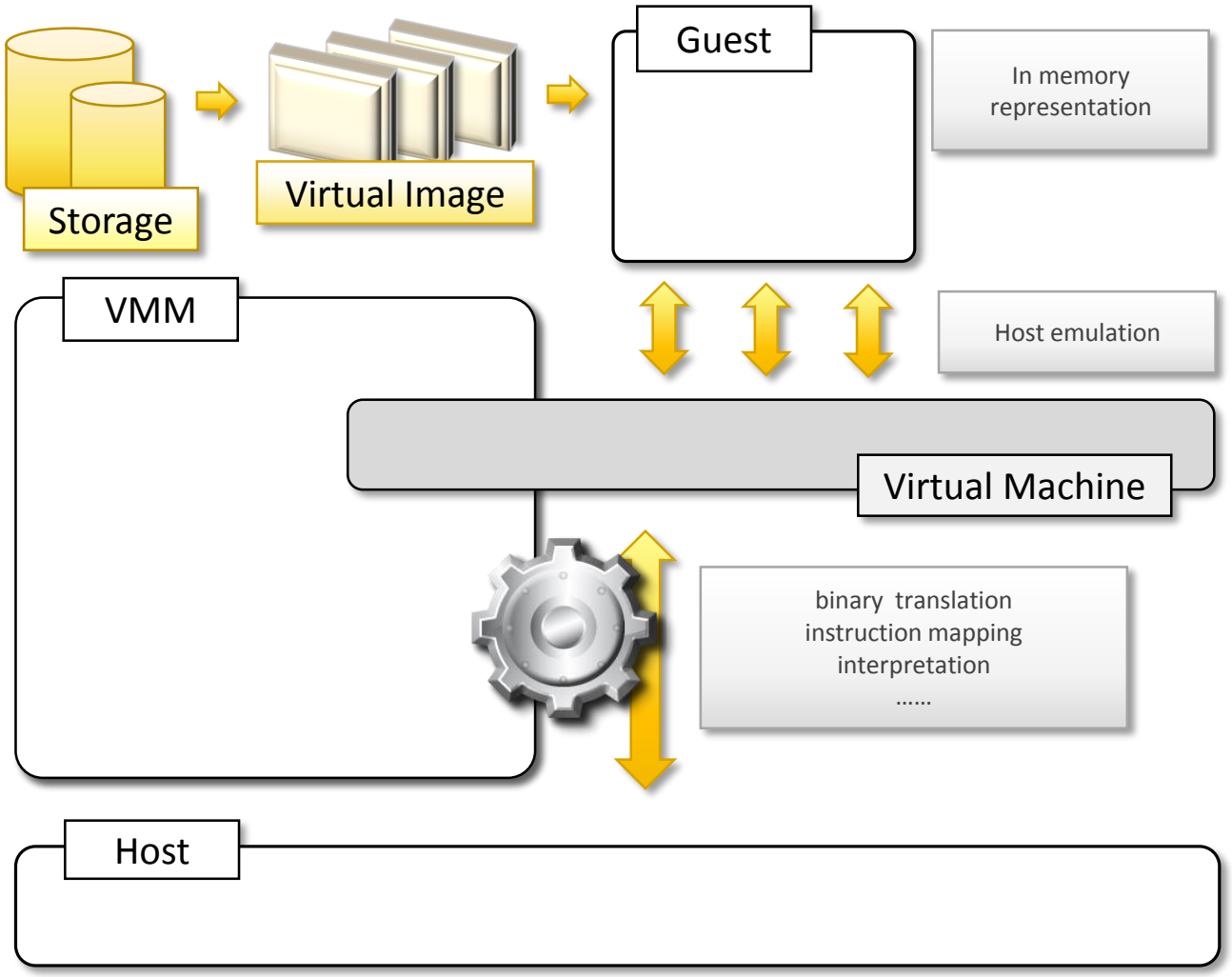


Security rings and privilege modes.

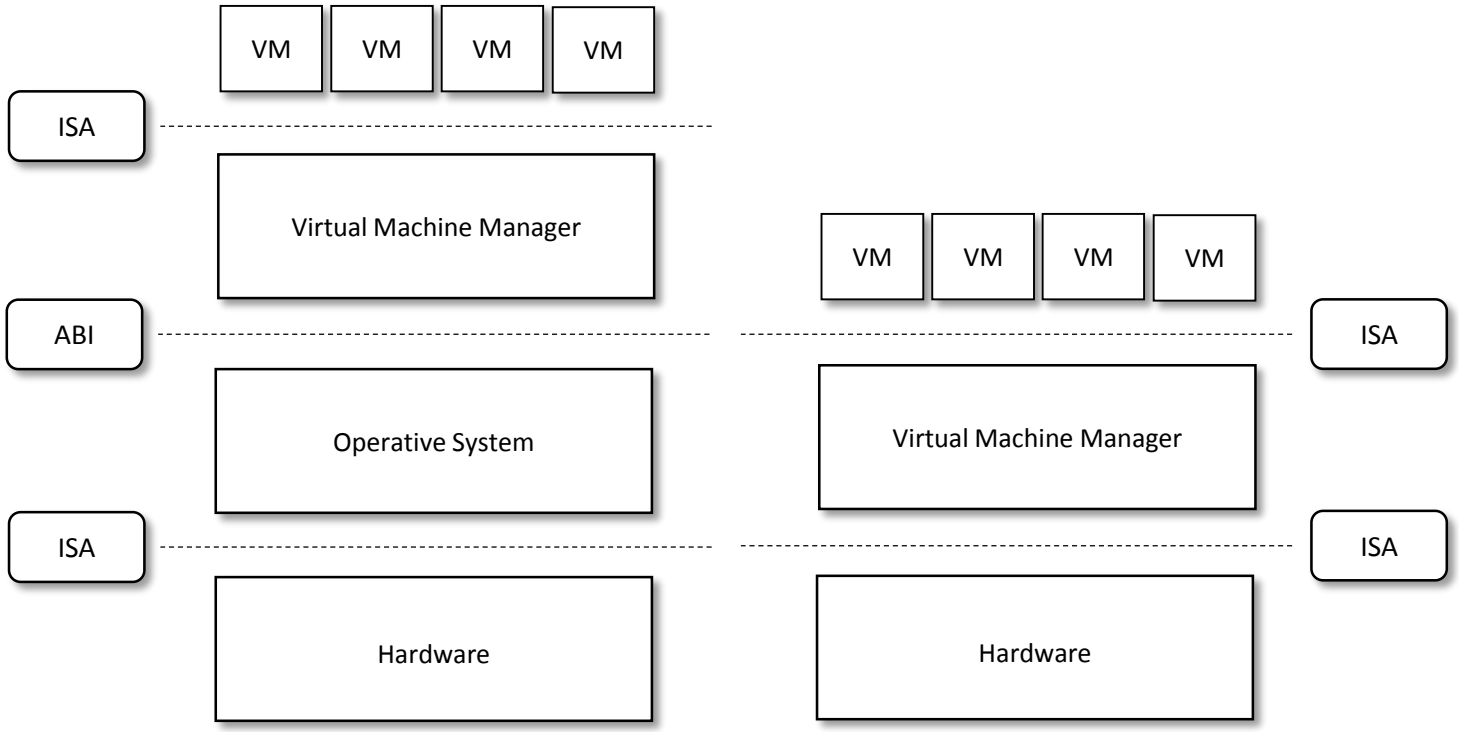
Taxonomy of Virtualization

- Execution virtualization
  - Hardware-level virtualization

Hardware virtualization or *platform virtualization* refers to the creation of a virtual machine that acts like a real computer with an operating system.



Hypervisors

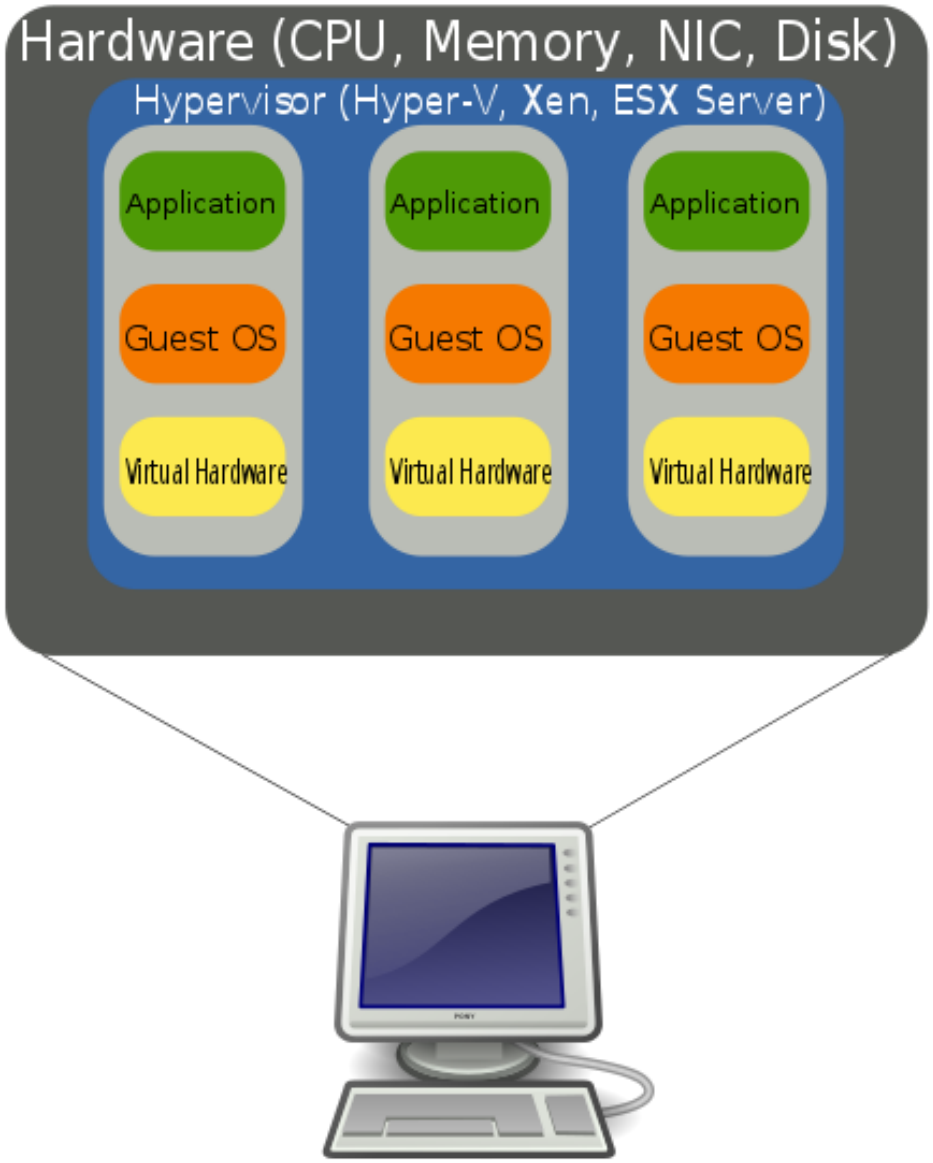


Hosted (left) and native (right) virtual machines. This figure provides a graphical representation of the two types of hypervisors.

Hardware virtualization techniques

- **Full Virtualization**
  - Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and **without any modification**, as though it were run on the raw hardware.
  - Example: Oracle's Virtualbox , VMware server, Microsoft Virtual PC

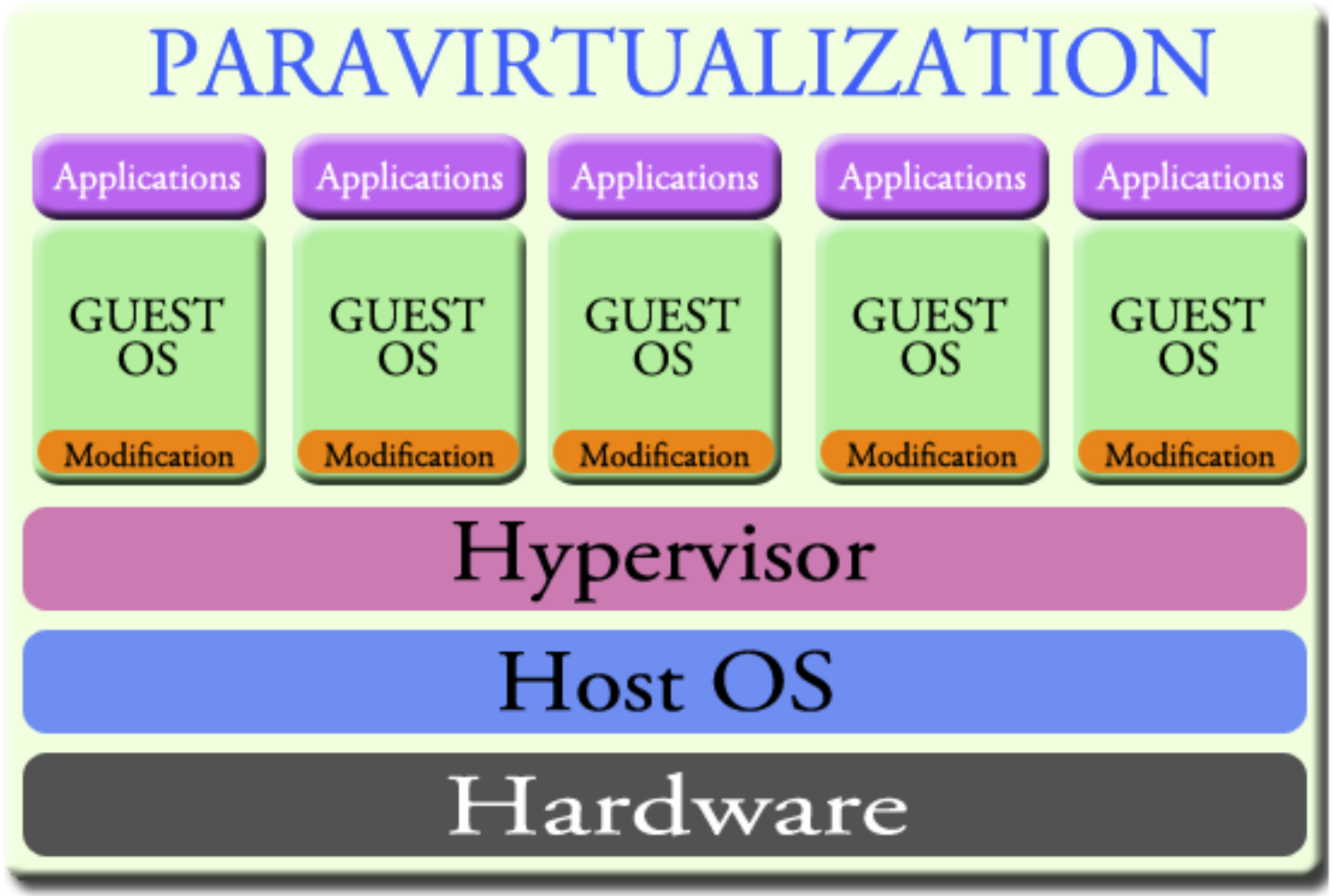
- **Partial virtualization**
  - Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation. – Limited Address space



Logical diagram of full virtualization
Sources: Wikipedia

Hardware virtualization techniques

- **Paravirtualization**
  - In computing, **Paravirtualization** is a virtualization technique that presents a software interface (like an OS) to the virtual machines which is similar, yet not identical to the underlying hardware–software interface.
  - Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.
  - The guest apps are executed in their own isolated domains, as if they are running on a separate system, but a hardware environment is not simulated. Guest programs need to be specifically modified to run in this environment.


PARAVIRTUALIZATION
Applications
Applications
Applications
Applications
Applications
GUEST OS
GUEST OS
GUEST OS
GUEST OS
GUEST OS
Modification
Modification
Modification
Modification
Modification
Hypervisor
Host OS
Hardware

Programming language-level virtualization

- Programming language-level virtualization is mostly used to achieve ease of deployment of applications, managed execution, and portability across different platforms and operating systems.

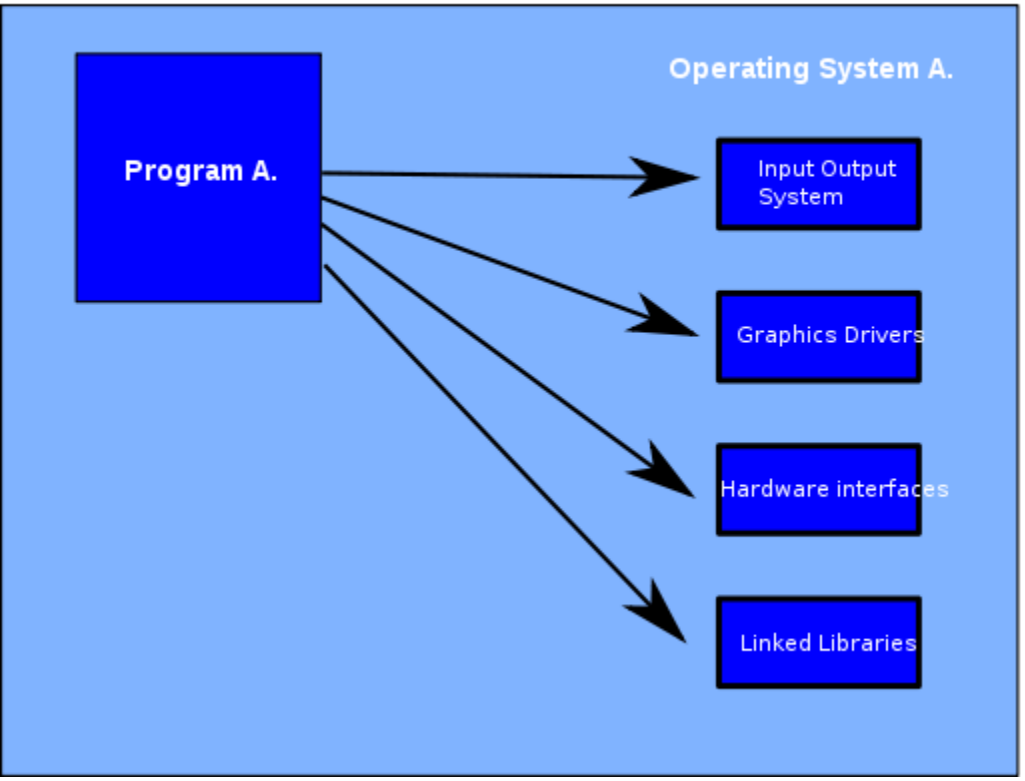
- It consists of a virtual machine executing the byte code of a program, which is the result of the compilation process.

- At runtime, the byte code can be either interpreted or compiled on the fly—or jitted (Just-in-time) against the underlying hardware instruction set.

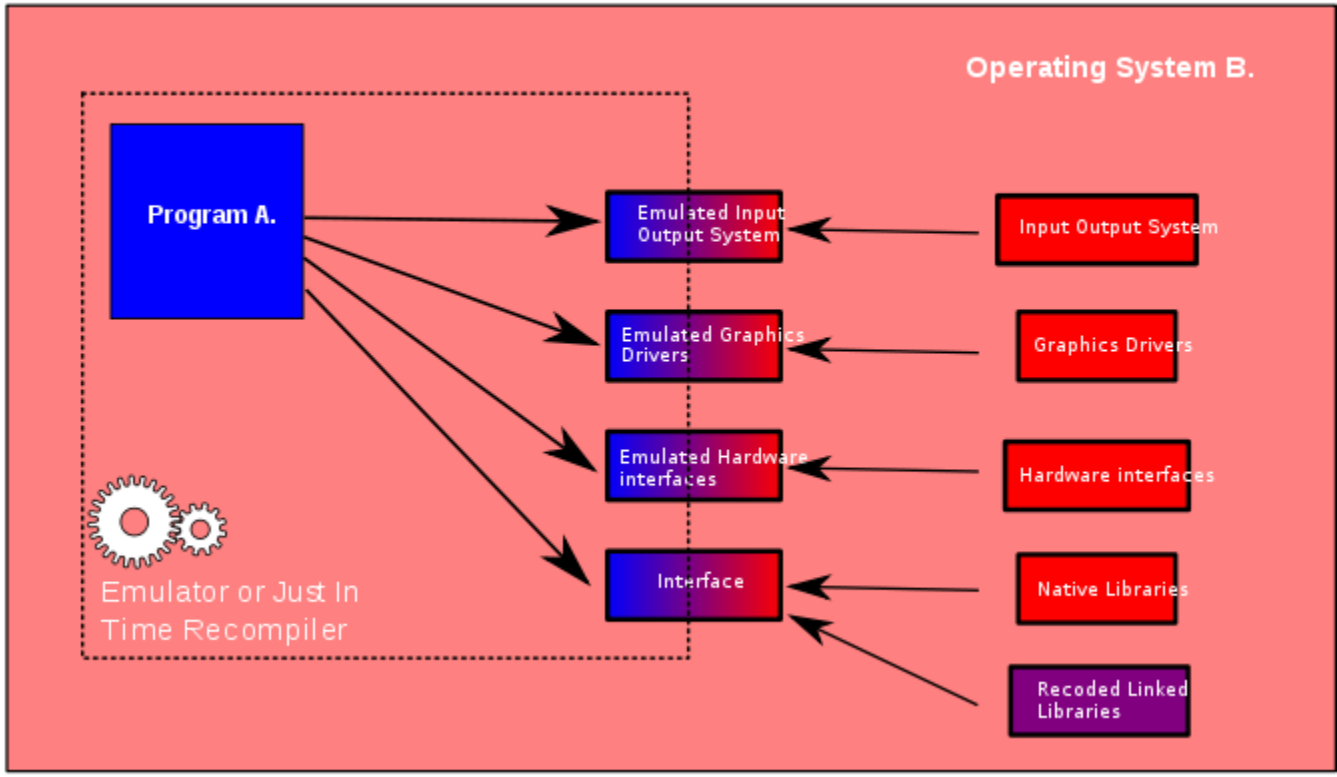
- Example: Java, .NET (CLI: Common Language Infrastructure)

Application or Process Virtualization

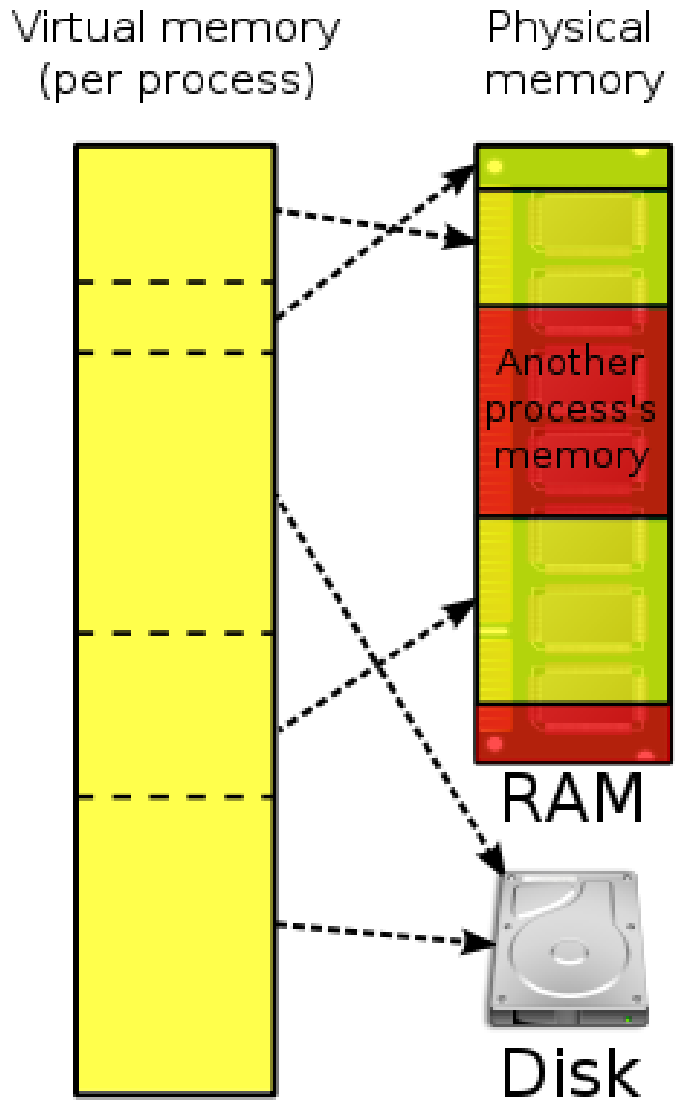
1. Application in Native Environment



2. Application in Non-Native Environment



Virtual Memory

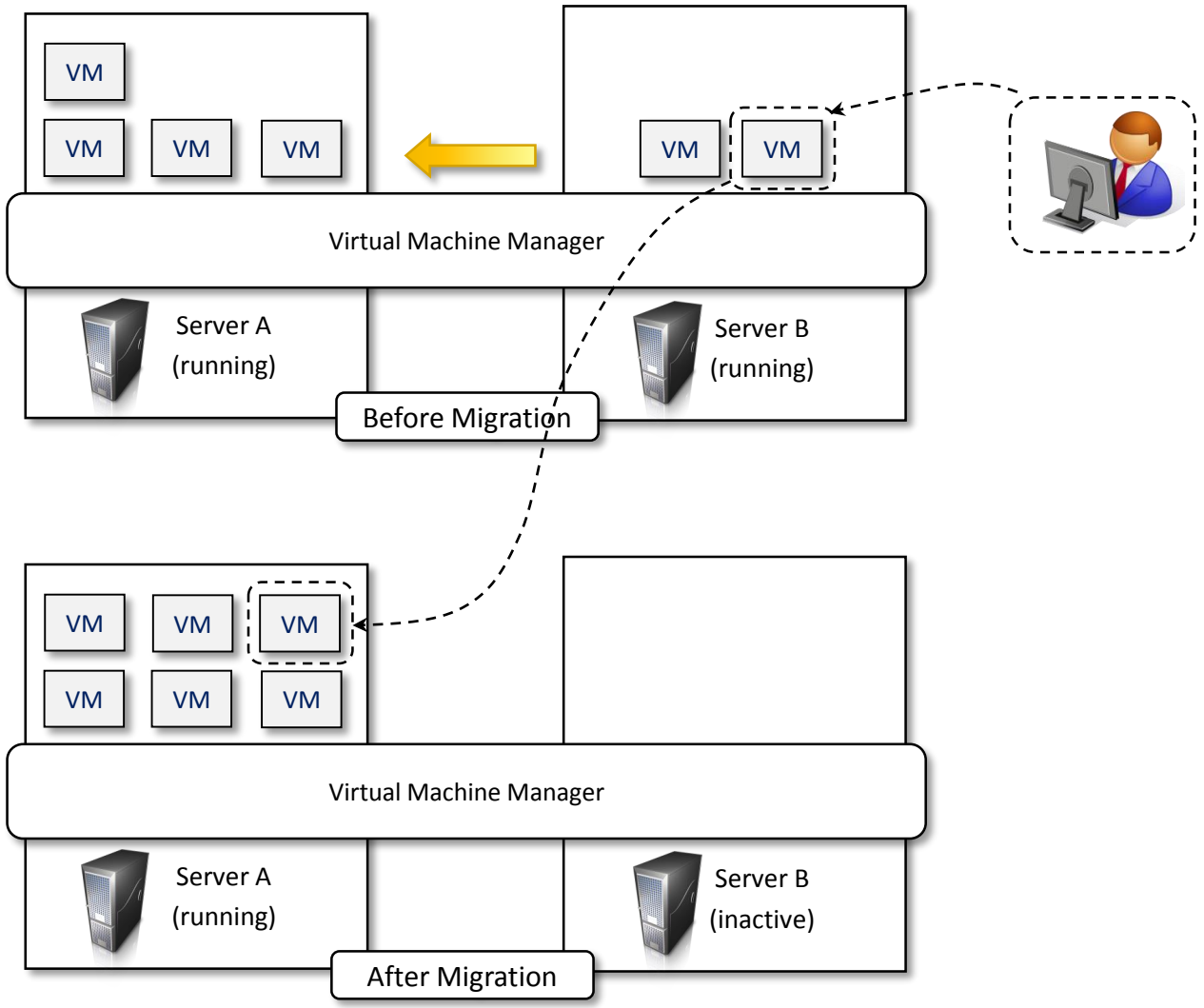


Virtualization and Cloud Computing

- Virtualization allows for the appropriate degree of customization, security, isolation, and manageability that are fundamental for delivering IT services on demand

- Hardware and programming language virtualization are the techniques adopted in cloud computing systems

  - Hardware Virtualization: **IaaS**; programming language virtualization: **PaaS**



Live migration and server consolidation.

Ref: Buyya, Rajkumar, Christian Vecchiola, and S. Thamarai Selvi. *Mastering cloud computing: foundations and applications programming*. Newnes, 2013.

Pros and cons of virtualization

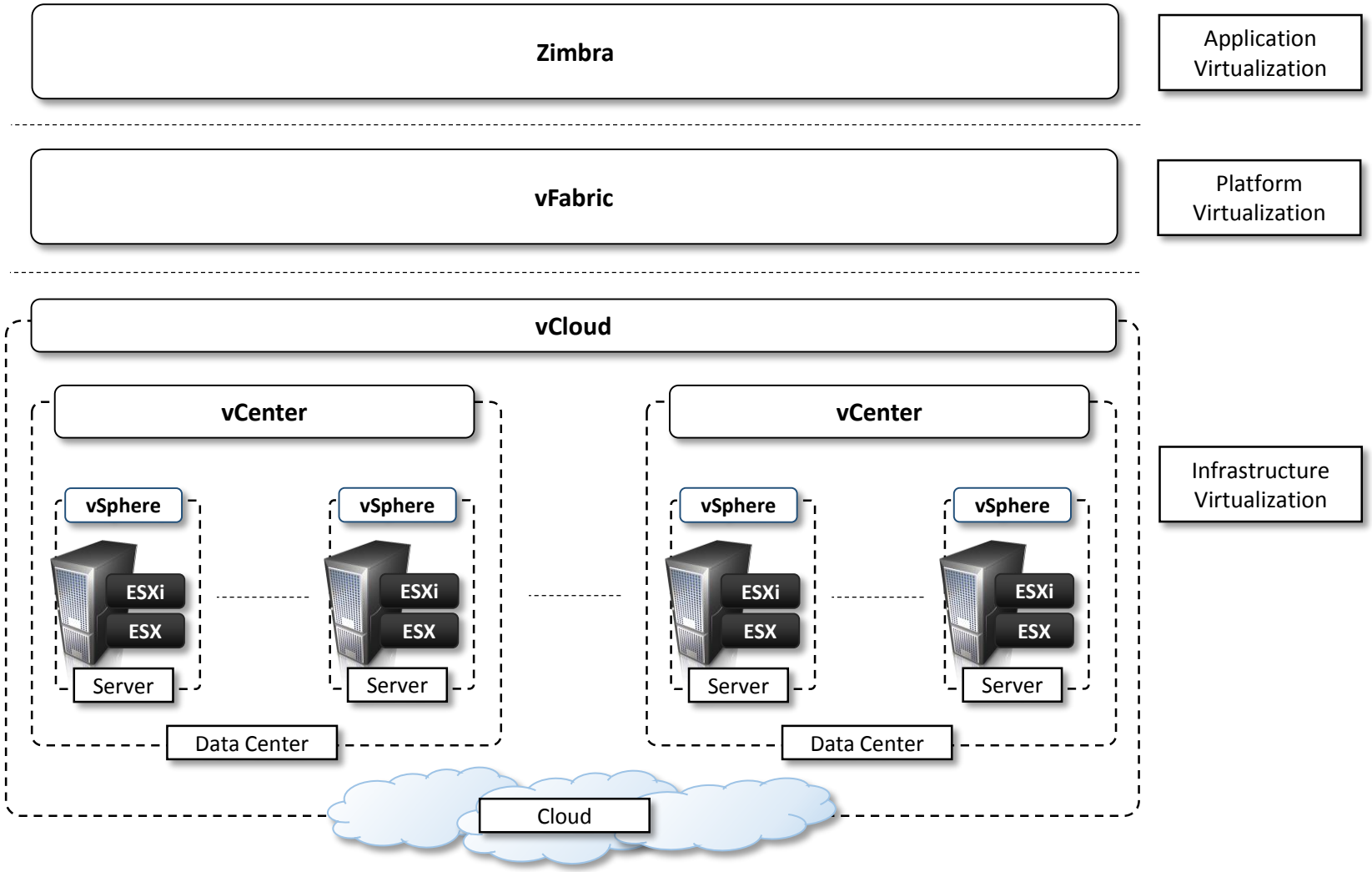
- Advantages: already discussed in previous slides

- Disadvantages:
  - Performance degradation
    - Maintaining the status of virtual processors
    - Support of privileged instructions (trap and simulate privileged instructions)
    - Support of paging within VM
    - Console functions
  - Inefficiency and degraded user experience
  - Security holes and new threats
    - Phissing etc.

Technology examples

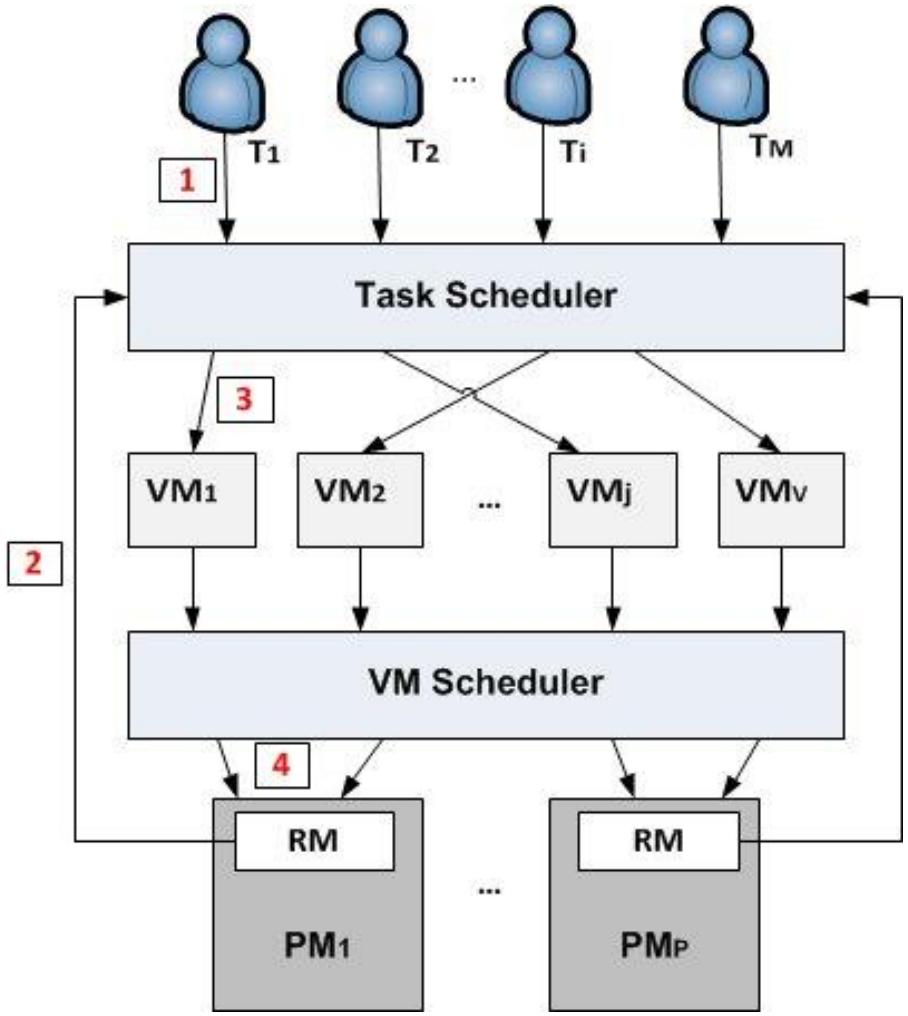
- Xen: Paravirtualization
- VMware: full virtualization
  - Full virtualization and binary translation
  - Virtualization solutions
    - End-user (desktop) virtualization
    - Server virtualization
    - Infrastructure virtualization and cloud computing solutions
- Microsoft Hyper-V

And many more …



VMware Cloud Solution stack.

Research direction in Cloud Computing



VM Scheduling in Cloud Computing

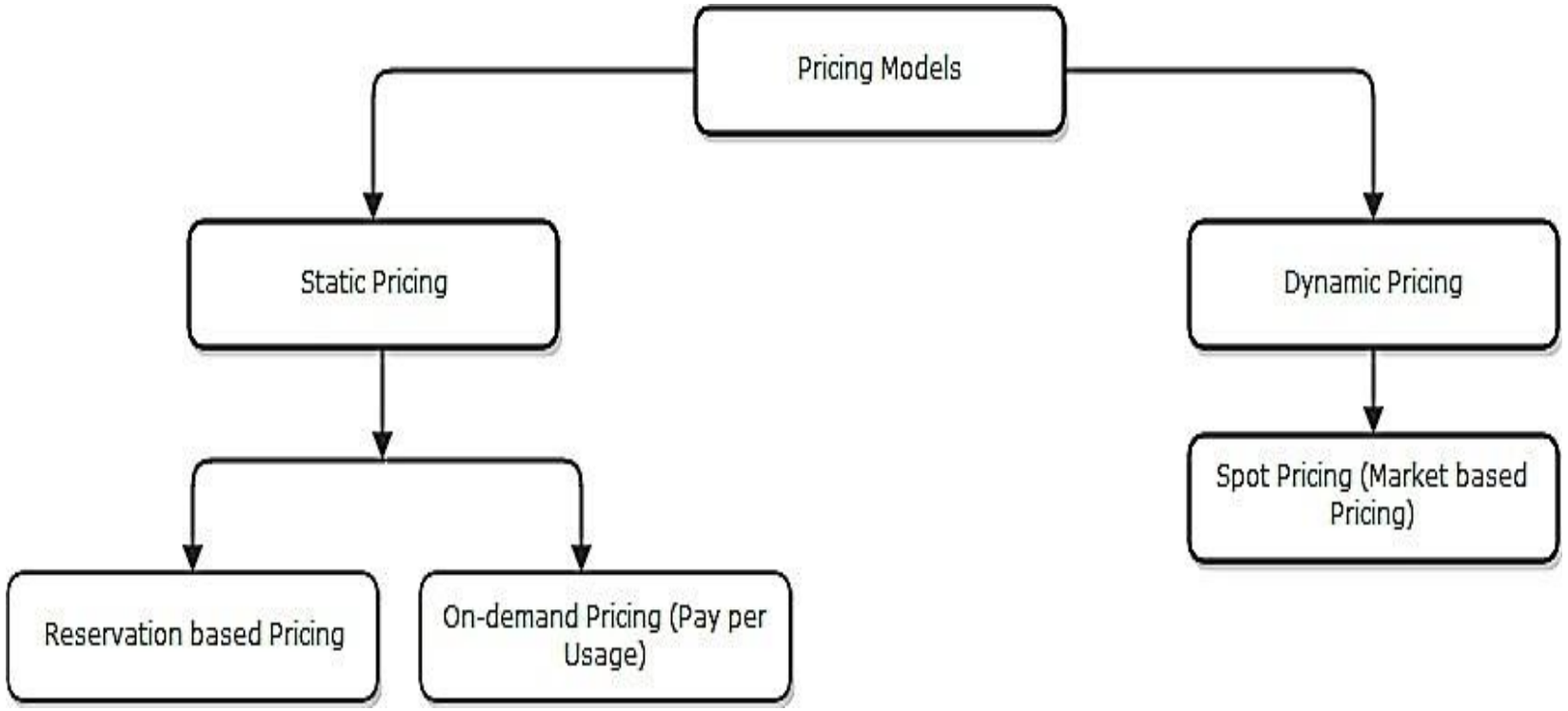
Cloud Economics and Market

- Why Economics so much important to cloud?
- Market Participants: *Cloud Providers* and *Cloud Users*
- Objectives of Cloud Provider
  - *To provide resources, To maximize revenue, To remain in business etc.*
- Objectives of Cloud User
  - *To get resources at minimum cost, To maximize the satisfaction etc.*
- **Main Aim: Designing of effective and efficient auction based resource provisioning and pricing models**
  - *For Provider: Revenue Maximization and Utilization of Resources*
  - *For Users: Best Pricing and Satisfaction*

Issues

- **Issues with Resource Provisioning and Pricing**
  - *Dynamic Pricing*
  - *Utilization of Spare resources*
  - *Resource Provisioning in Competitive and Strategic Cloud Market*
  - *Truthful Revelation in Cloud Market*
  - *Healthy Competition in Cloud Market*
  - *Spot Pricing in Cloud Ecosystem*

Pricing of Cloud Resources



Auction based Resource Allocation

- A market mechanism with a set of rules determining prices and resource allocations on basis of bids submitted from participants
- Aims to optimize the payoff of the participants
- Winners praise their winning abilities and losers blame other bidders not the auctioneer for the loss
- Sponsored Search Auction: Allocation of limited number of sponsored slots.
- Entities in auction
  - Auctioneer (Market Maker) and Bidder
- Types of auction
  - Forward Auction
  - Reverse Auction
  - Double Auction

Auction based Resource Allocation

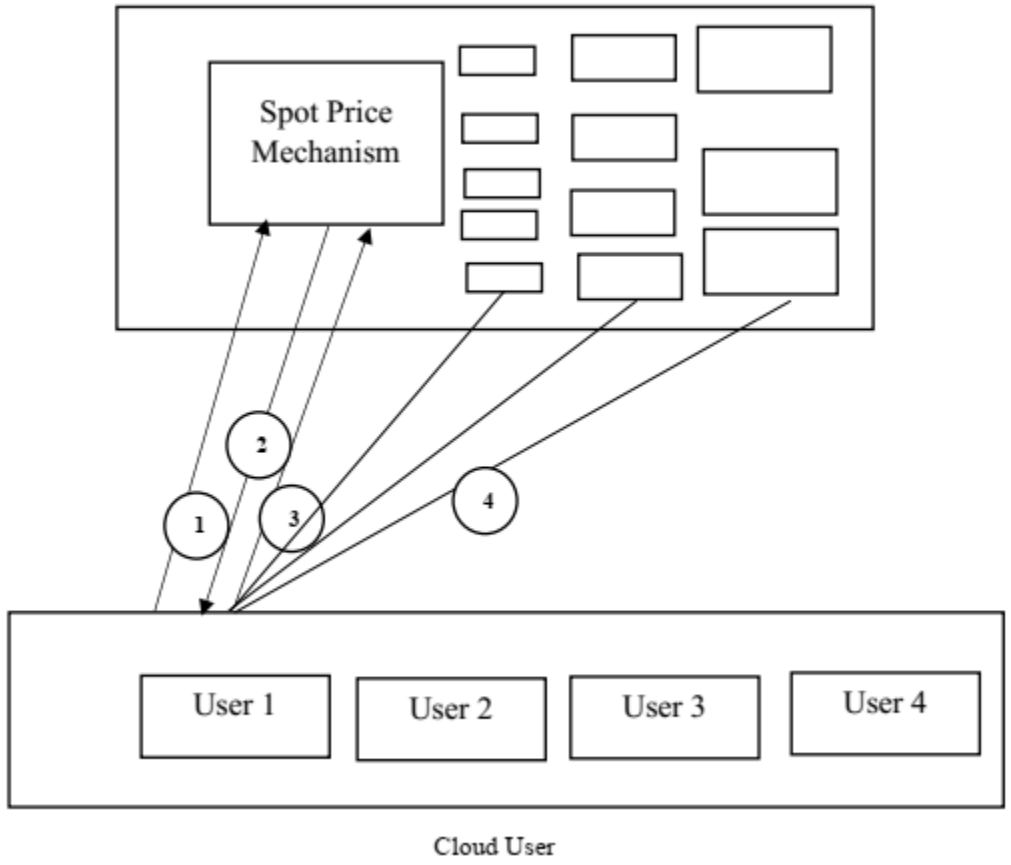
- Auction properties:
  - Efficiency: There is no loss of social welfare.
  - Incentive Compatibility: Encourage truthful bidding
  - Bidders' Optimality: Best possible price
  - Individual Rationality: It is no harmful to participate
  - Budget-balance: Sum of money transfer of all the participants is equal to zero
  - Egalitarian social welfare (fairness): Reduces the differences among the bidders (rich bidder and poor bidder)

Auction based Resource Allocation

- Why Auction in Cloud Computing
  - Utilization of spare resources
  - Dynamic pricing implementation
  - Prevents market manipulations

- Real implementation of auction in Cloud (Motivation)
  - **Amazon Spot Instances**: Allocation of unutilized resources using auction
    - https://aws.amazon.com/ec2/spot/
    - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html
  - **SpotCloud (Acquired by Virtustream)**: Provides a generalized structured framework for the creation of a Compute Spot Market where compute centric commodities are traded
  - **MODAClouds**: Model-driven Approach for design and execution of applications on multiple Clouds

Cloud Spot Market Mechanism



Step 1: Cloud Provider collects the bids (instance type, number of instances, availability zone bid value) from all Cloud users.

Step 2: Provider calculates allocation and payment and inform to the user.

Step 3: Winning User pays to the Cloud provider.

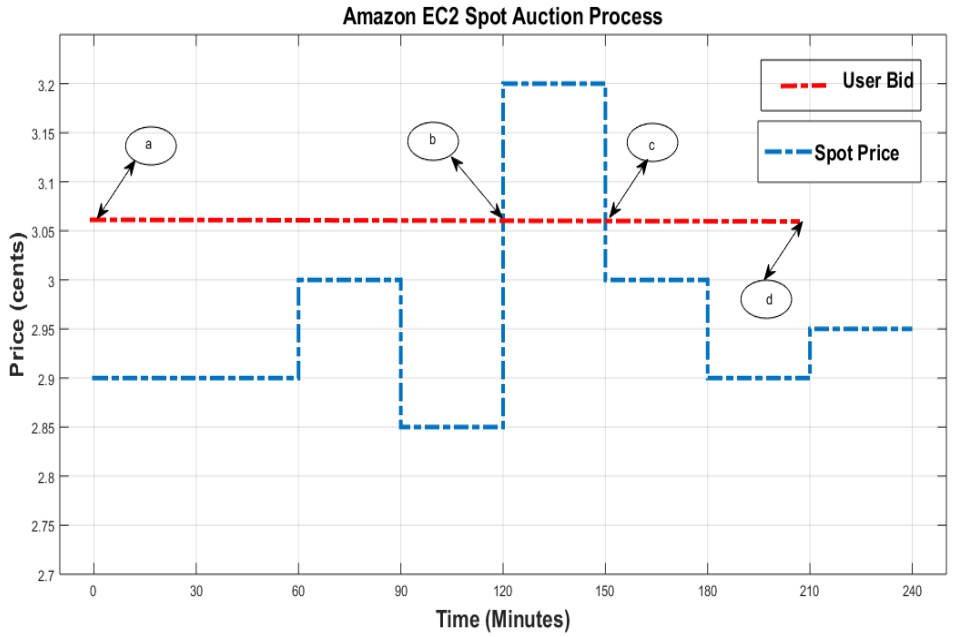
Step 4: Provider provides access to the spot instances.

Spot Pricing in Cloud Computing

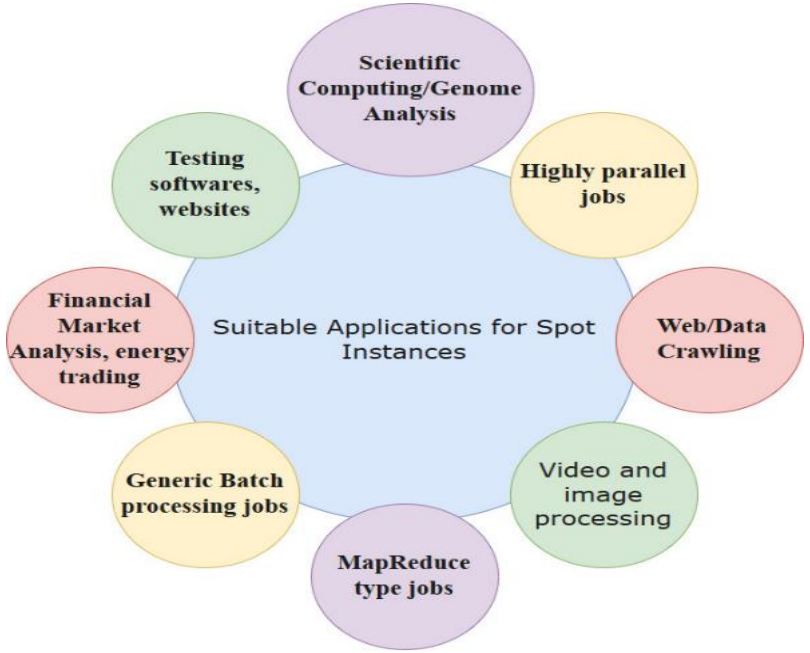
- Good for time-flexible and interruption tolerant tasks
- Amazon categorizes time-flexible and interruption-tolerant tasks in four groups:
  - **Optional tasks**: Tasks which are not strictly required and can be executed on SIs when prices are low. It can be stopped in case of higher prices.
  - **Delayable tasks**: Tasks with deadline provide flexibility about when they are executed.
  - **Acceleratable tasks**: Tasks that can be speeded up by adding more SIs in case of availability of SIs at low price.
  - **Large scale tasks**: Tasks that may require computing power that one can't access any other way, SIs can cost-effectively run them.

Spot Pricing in Cloud Computing

Amazon Spot Pricing Mechanism

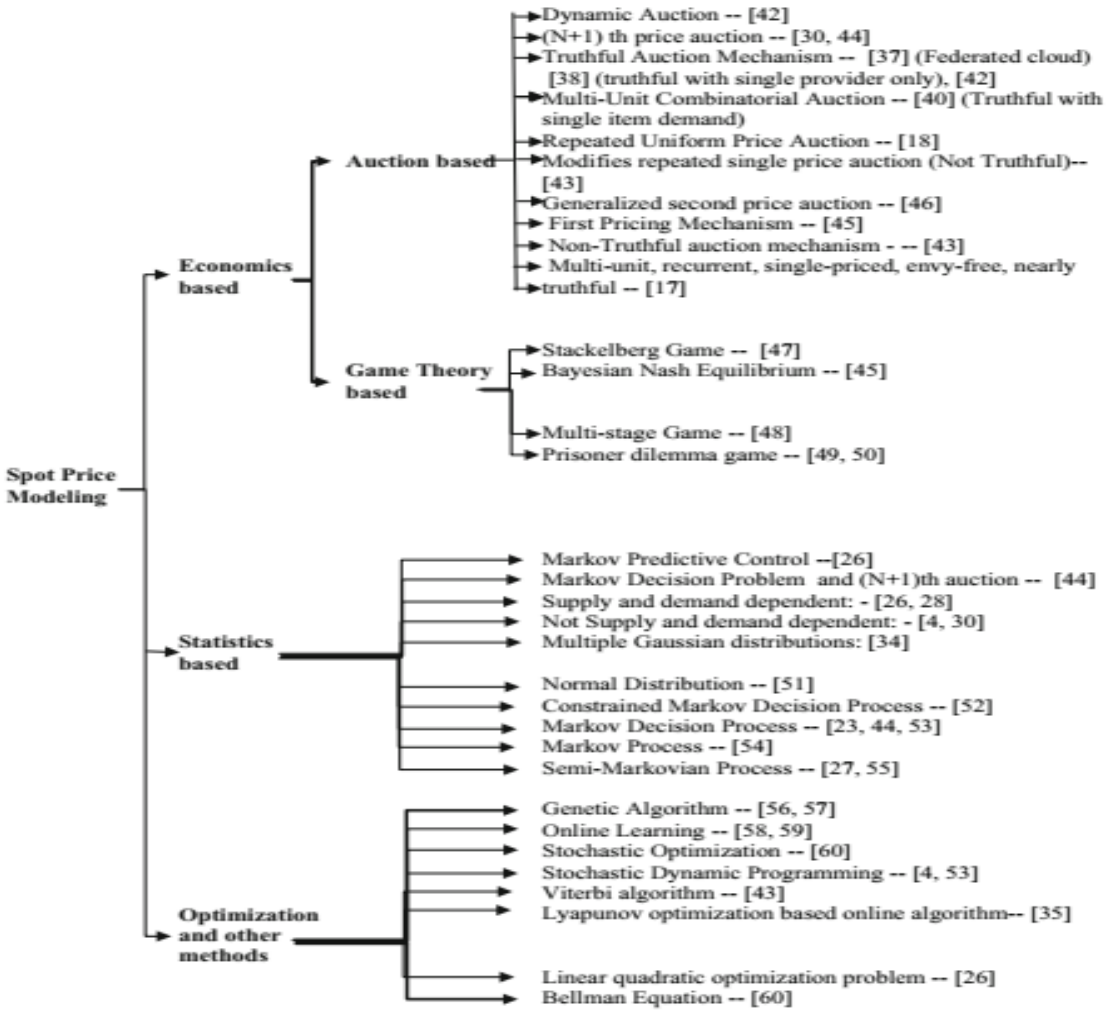


Suitable Applications for Spot Instances



Reference: Baranwal, Gaurav, Dinesh Kumar, Zahid Raza, and Deo Prakash Vidyarthi. *Auction Based Resource Provisioning in Cloud Computing*. Springer, 2018.

Spot Price Modelling and Bidding Strategy



Reference: Kumar, Dinesh, Gaurav Baranwal, Zahid Raza, and Deo Prakash Vidyarthi. "A survey on spot pricing in cloud computing." *Journal of Network and Systems Management* 26, no. 4 (2018): 809-856.

Auction

- Auction is a market-based mechanism to buy/sell resources.
- Entities:
  - Buyer
  - Seller
  - Bidder (Private or Public information)
  - Auctioneer

Auction

- Classification
  - Single-dimensional or multi-dimensional
  - One-sided or double-sided
  - Open-cry or sealed-bid
  - First-price or kth price (Vickrey auction)
  - Single-unit or multiple-unit
  - Single-item or multiple-item
  - Forward, reverse, or double

Auction

- Desirable Properties
  - **Solution Equilibrium**: no bidder wishes to change its bid
  - **Incentive Compatibility**: bidders receive maximum utility on revealing their preferences truthfully
  - **Allocative Efficiency**: maximizes the total social welfare
  - **Individual Rationality**: every participant gains non-negative utility
  - **Budget Balance**: payment done by the buyers is equal to payment received by the sellers
  - **Revenue Maximization or Cost Minimization**
  - **Fairness**: egalitarian approach minimizes the welfare difference among the participants
  - **Computational Efficiency**: Time Complexity
  - **Cheatproofness**: Robust to false bidding, collusion, etc.

Auction

- Combinatorial Resource Allocation

$$\text{minimze } \sum_{i=1}^{n} s_i pp_i$$

subject to

$$\sum_{i=1}^{n} s_i q_j^i \geq q_j^c, j = 1,2, \dots, l$$

where, $s_i \in \{0,1\}$ and $s_i = 1$, if provider p_i wins and otherwise.

Auction

- Payment Mechanism

Let $S = \{s_1, s_2, \dots, s_n\}$ be the obtained solution of winner determination problem and S^{-i} be the obtained solution in the absence of provider p_i, i.e., $S^{-i} = \{s_1, s_2, \dots, s_{i-1}, s_{i+1}, \dots, s_n\}$. Final traded price (ftp^i), received by each provider p_i

$$ftp^i = pp_i S_i + \sum_{j \neq i} pp_j S^{-i} - \sum pp_j S_j$$

Payment received by each provider is the sum of quoted cost of winner and difference between procurement cost without the winner and procurement cost with the winner.

Resource Provisioning in Cloud Computing with CloudSim

Outline

- To introduce you the basic concepts of Resource/Task Scheduling
- To give you the knowledge about various Resource Allocation mechanisms in Cloud Computing
- To motivate you for using CloudSim Simulator as a tool in research
- Essentials to start with Cloudsim.
- Insight on Cloudsim modeled components.
- Insights on different packages available in CloudSim project
- Insight on task scheduling policies in CloudSim.

Computing Trends

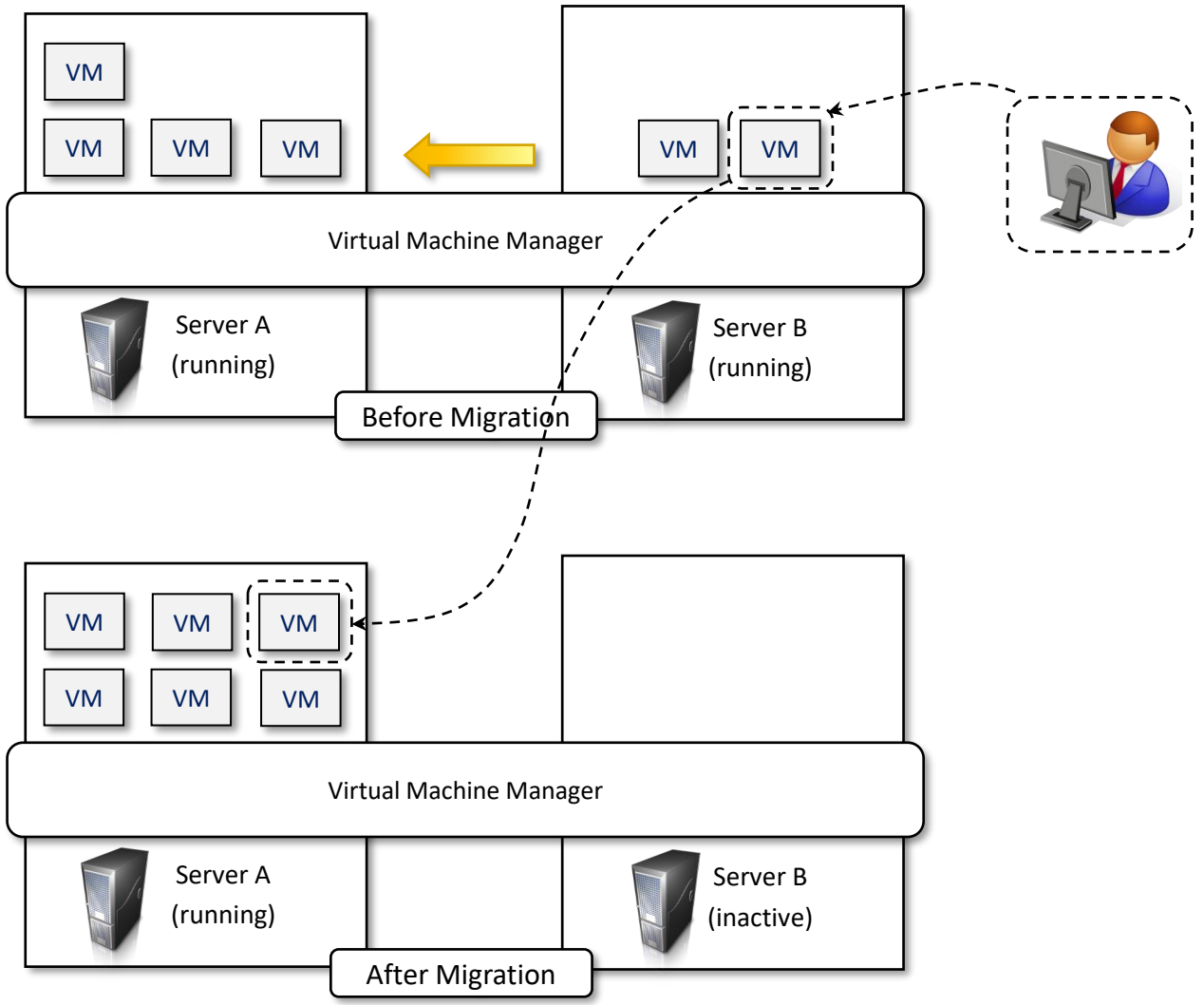
- Virtualization technology is key
  - allows a single physical machine to run many independent virtual machine thus increases utilization of physical servers
  - Enables multiple types of OSs to run in isolation of other OSs
  - Separating applications from the underlying infrastructure
  - Enables portability of virtual machines between physical machines

Virtualization and Cloud Computing

- Virtualization allows for the appropriate degree of customization, security, isolation, and manageability that are fundamental for delivering IT services on demand

- Hardware and programming language virtualization are the techniques adopted in cloud computing systems

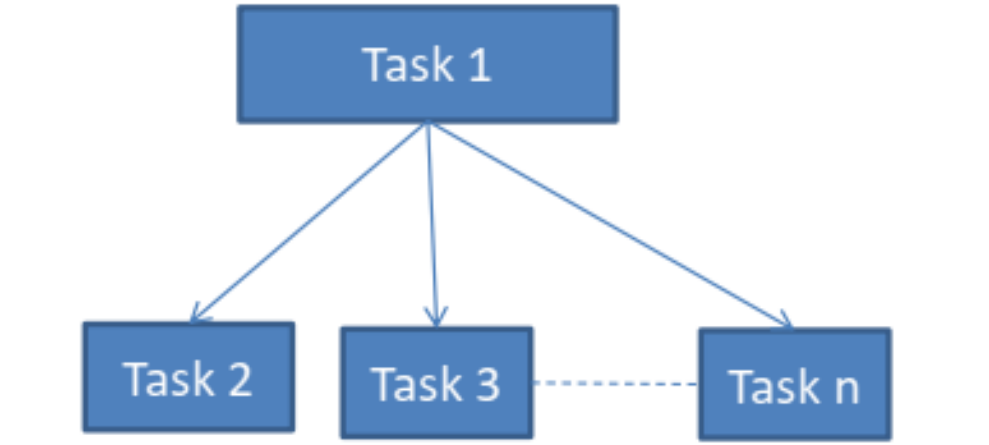
  - Hardware Virtualization: **IaaS**; programming language virtualization: **PaaS**



Live migration and server consolidation.
Ref: Mastering Cloud Computing by Rajkumar Buyya

Workload Types

- Divisible jobs (an application can be arbitrarily partitioned into smaller tasks)
  - Example: matrix multiplication application

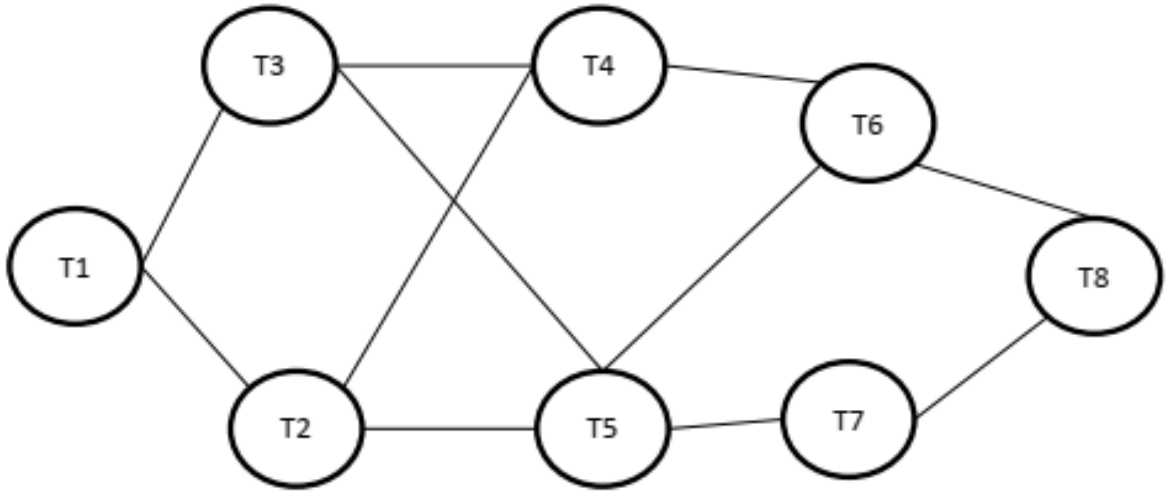


Structure of matrix multiplication application

- Indivisible processes (entire process must be assign to a single processor)

Workflow Applications

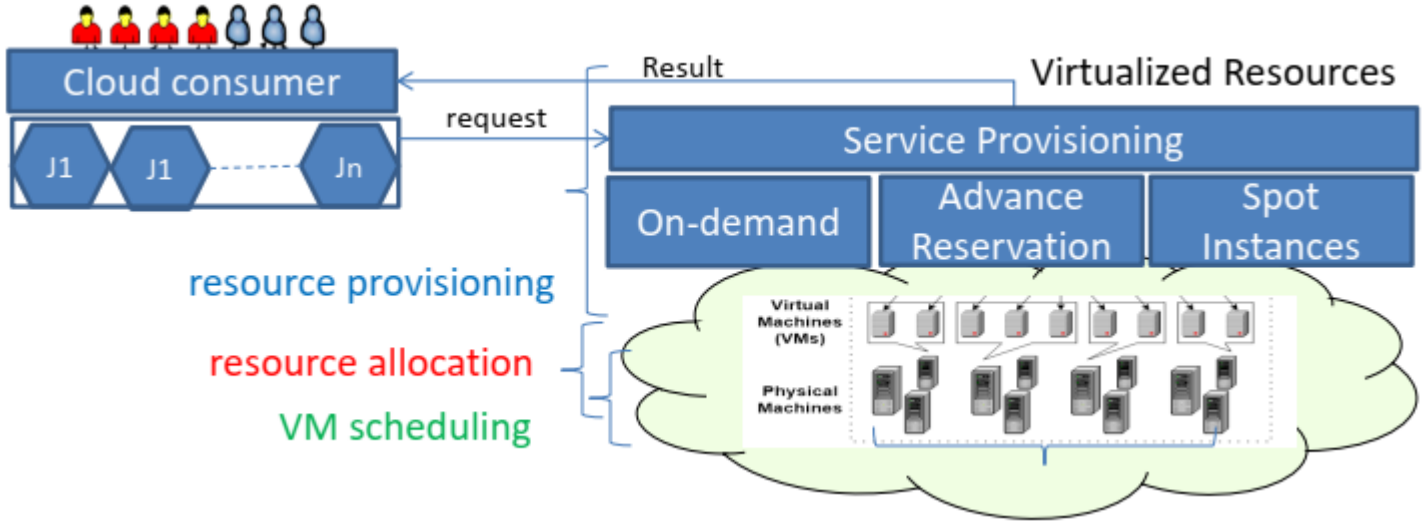
- Workflow applications are represented by a directed acyclic task graph.



Structure of divide-and-conquer programs

Cloud Data Center Model

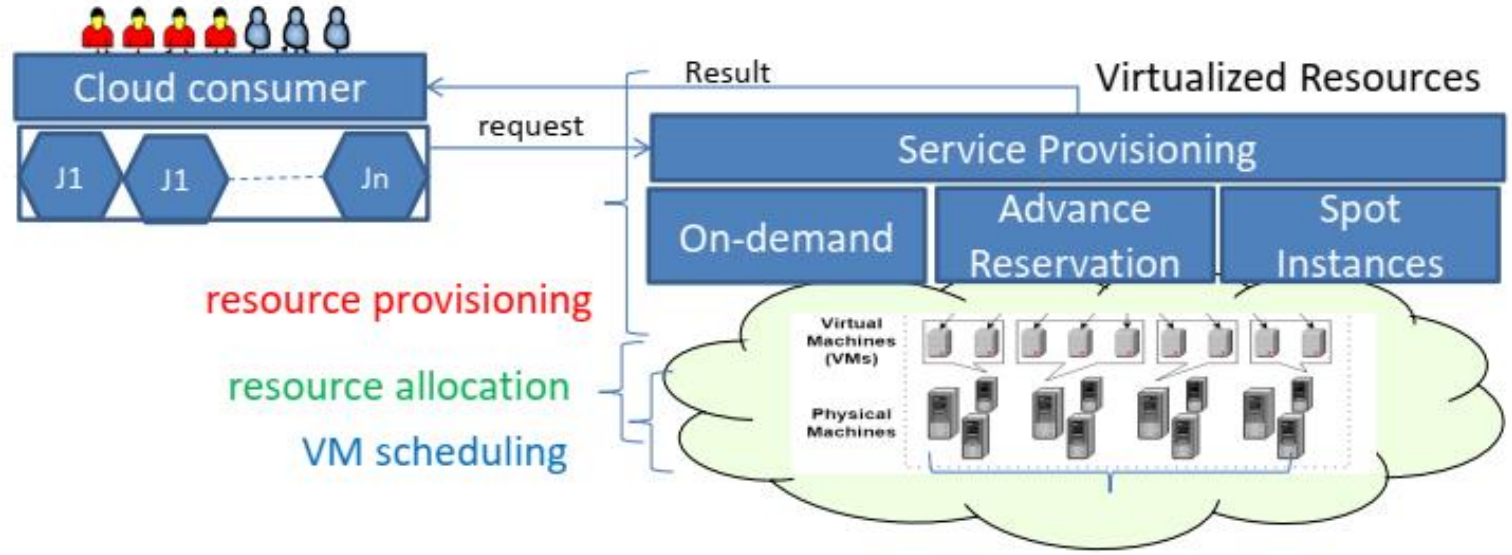
- Cloud services and basic functions are provided based on Virtual Resources which are abstracted from Physical Resources.



- The Cloud resource provisioning enables virtualized resources to be allocated to Cloud consumers based on three provisioning plans.

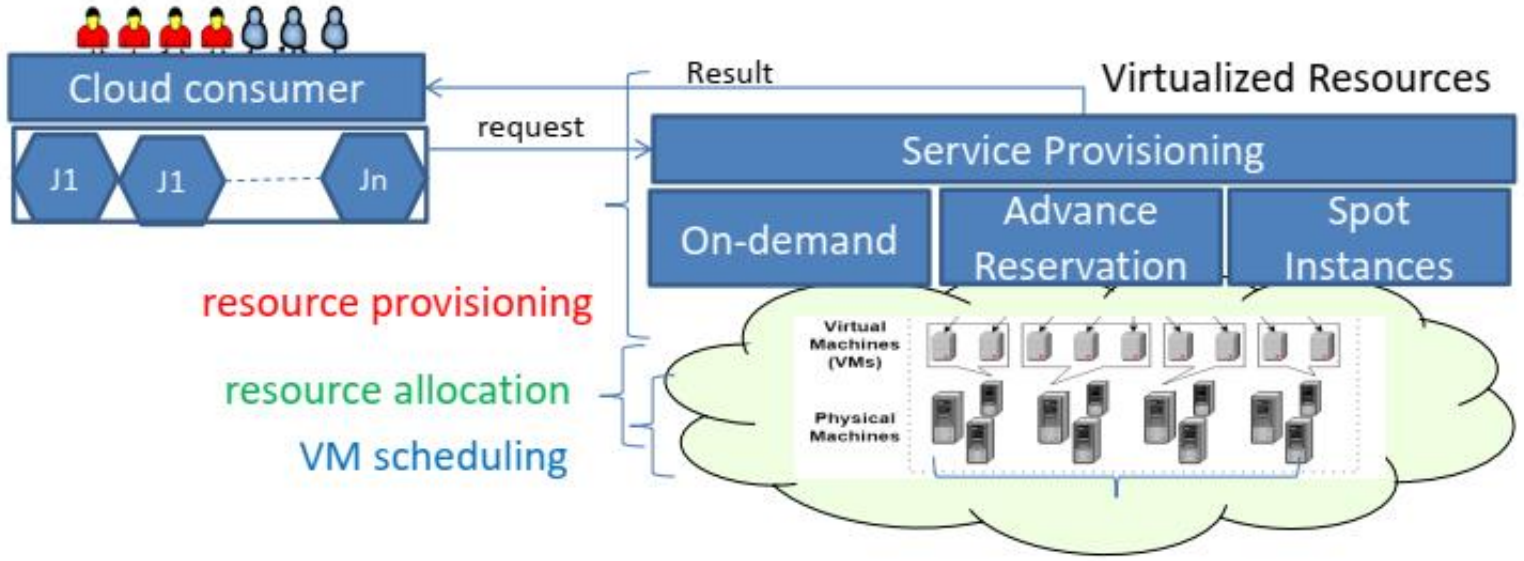
Resource provisioning

- Resource provisioning is responsible
  - To understand the user needs
  - To prepare VMs with appropriate resources to match the workloads and QoS requirements
  - SLA Negotiation
  - Etc.



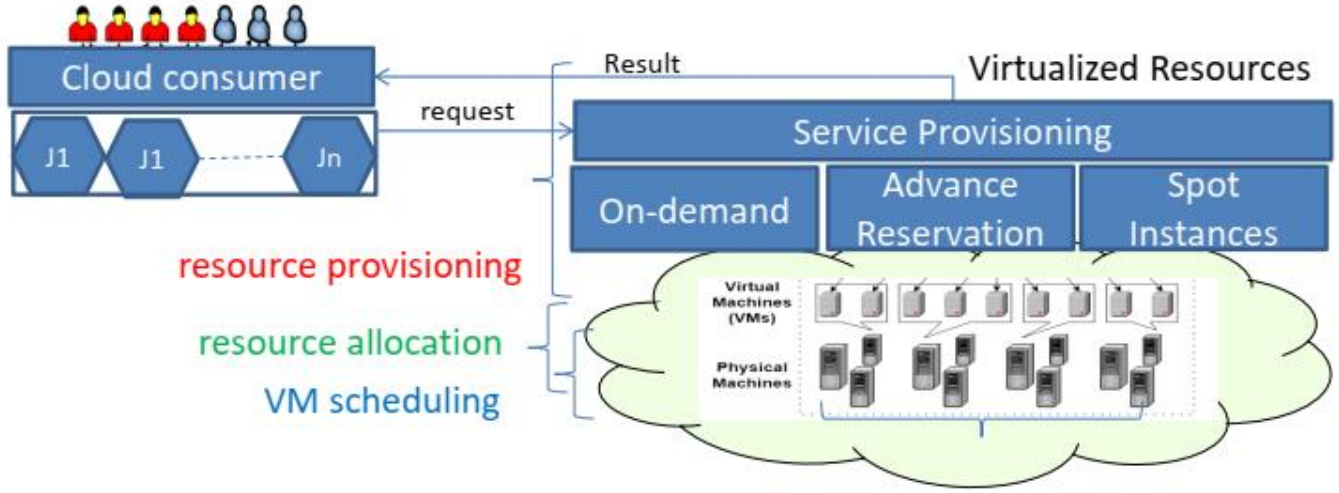
Resource Allocation

- Resource allocation is responsible
  – To select an optimal set of physical machines to host the received services (VMs),
  – To ensure the resource and QoS constraints are met.
  – To manage changes in resources availability through VMs restore or migration



VM Scheduling

- VM scheduling is responsible
  – A resource can be used either in shared (time sharing) mode or exclusive mode (space sharing).
  – To ensure that the VMs receive the required services based on the rule (policy) description.



Cloud resource provisioning problem

- The Cloud resource provisioning problem involves tasks that must be scheduled on cloud resources subject to some constraints to optimize some objective function.

- The growing scale of Cloud computing and the increasing complexity of users' requirements introduce additional constraints and make allocation decisions more challenging with difficult tradeoffs between user satisfaction and profit maximization.

Resource Provisioning and Allocation

- A wide variety of resource provisioning goals exist:
  – High resource utilization
  – Energy efficiency
  – Reliability of services
  – High Availability

- Optimally achieving the above goals in cloud computing environment has been proved to be an NP-Hard problem due to its combinatorial optimization nature.

- There are no algorithms which may produce optimal solution within polynomial time for such kind of problems.

Optimization algorithms

- Optimization algorithms can be roughly divided into two categories: exact algorithms and heuristics.
- Exact algorithms are designed in such a way that it is guaranteed that they will find the optimal solution in a finite amount of time.
  - For scheduling optimization problems (e.g. NP-hard or global optimization) this "finite amount of time" may increase exponentially in respect to the dimensions of the problem.
- Heuristics do not have this guarantee, and therefore generally return solutions that are near to optimal.
    - Heuristic algorithms usually find "good" solutions in a "reasonable" amount of time

Heuristic algorithms

- Heuristic algorithms usually are

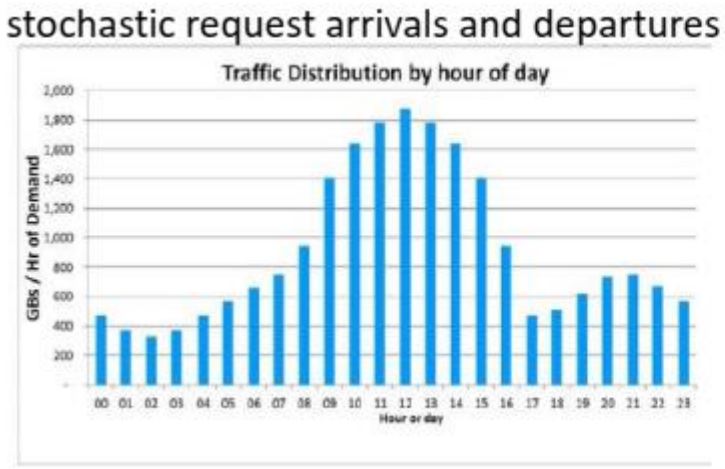
  – adapted to the problem at hand and they try to take full advantage of the particularities of this problem.

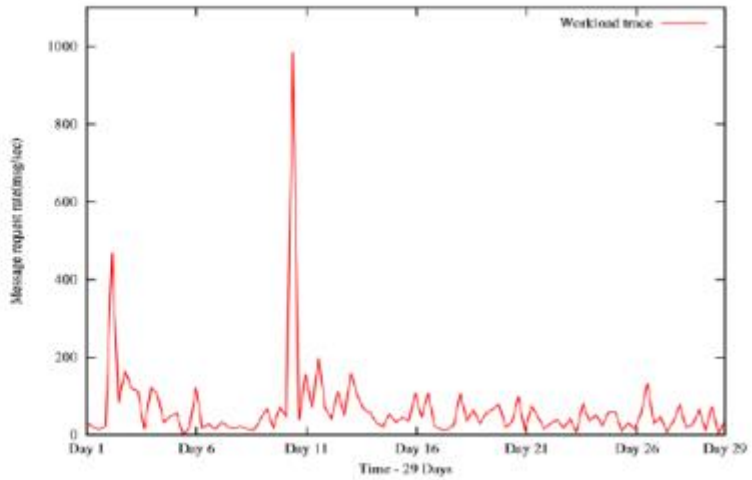
  – However, because they are often too greedy, they usually get trapped in a local optimum and thus fail, in general, to obtain the global optimum solution.

  – Heuristic algorithms are generally make simplifying assumptions to relax constrains.

Relevant Information

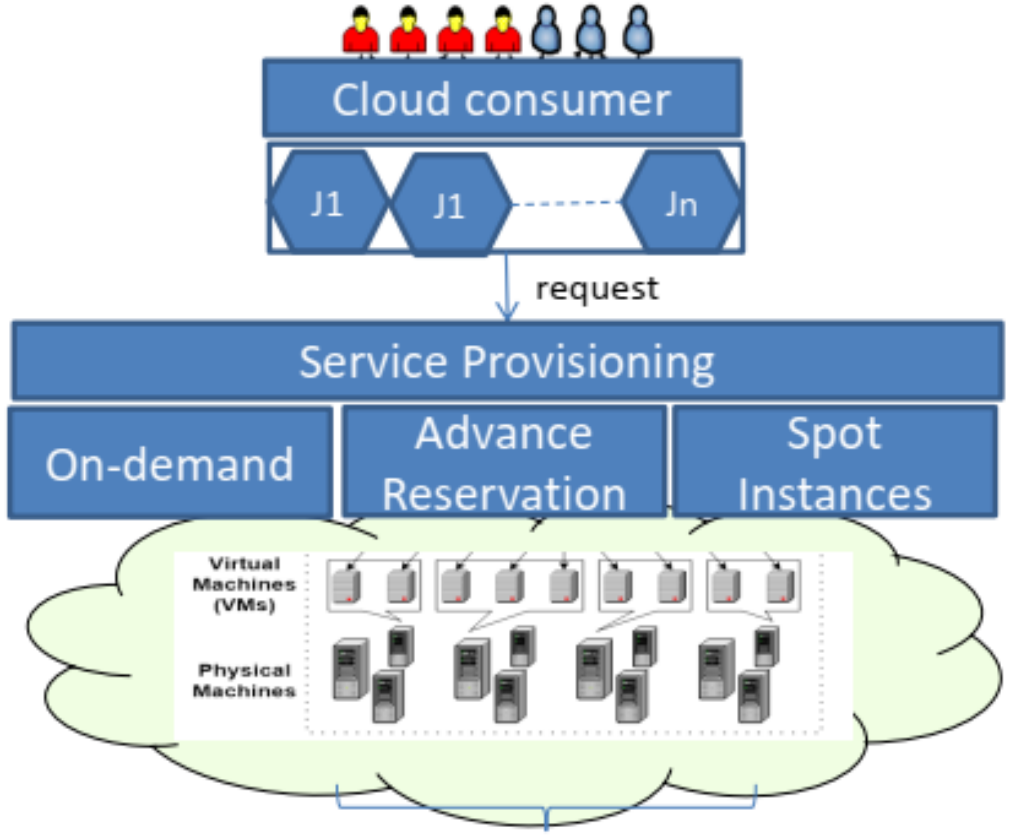
- When developing resource provisioning algorithms, it is imperative to consider information such as
  - highly heterogeneous and time-varying workloads
  - daily demand distribution of a typical Internet application
  - request arrivals and departures statistics





- https://gigaom.com/2012/02/11/which-is-less-expensive-amazon-or-self-hosted/
  https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md

Relevant Information



Cloud providers are faced with stochastic request arrivals and departures, which generates highly heterogeneous and time-varying workloads.

What are the workload characteristics?
What are the VM capability ?

What do we want to minimize: *power consumption, SLA violation, completion time,* cost, *etc.?*

What do we want to maximize:
resource usage, revenue, etc.?

Resource provisioning classification

- Resource provisioning can be generally classified as
- Static Resource Provisioning
  - Given information such as highly heterogeneous and time-varying workloads , allocation solutions based on static resource provisioning lead to poor performance and hinder providers from achieving expected profits.
- Dynamic Resource Provisioning
  - With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not.
- Hybrid Resource Provisioning
  - Combines other resource provisioning approaches

Example of Heuristic Algorithms

- Rule-based heuristics
  - First Come First Serve (FCFS),
  - Minimum Completion Time (MCT),
  - Minimum Execution Time (MET),

Meta-heuristic Approaches

- There are many meta-heuristic approaches that range from simple local search procedures to complex learning processes
  - simulated annealing (SA), [e.g., single solution approach]
  - evolutionary algorithms (EA), [e.g., population-based]
  - ant colony optimization (ACO), [e.g., of swarm intelligence]
  - particle swarm optimization(PSO) [e.g., population-based and swarm intelligence approaches]
- Many different new variants are continually being proposed.

Preliminaries

- Given a set of applications (Jobs), each can be processed on any machine
  - J_i: job i, $i \in N = \{1, \cdots, n\}$
  - p_i : processing time of job J_i
  - C_i : completion time of job J_i
- System
  - M_j: machine j, $j \in M = \{1, \cdots, m\}$
  - All machines are identical
  - p_{ij}: processing time of job J_i on machine M_j
- Find an assignment of the jobs to the machines such that the makespan (maximum completion time of all jobs) is minimized.

$$C_{max} = \max(C_1, C_2, \cdots, C_n)$$

Problem Formulation

A Mixed Integer Programming (MIP) formulation of the minimum makespan problem follows:

$$\boxed{minimize\ C_{max}}$$

$$such\ that$$

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad 1 \le i \le n$$

$$\sum_{i=1}^{n} (p_i \cdot x_{ij}) \le C_{max}, \qquad 1 \le j \le m$$

$$x_{ij} = \begin{cases} 1 & \text{if the job i is assigned to machine j} \\ 0 & \text{if the job i is not assigned to machine j} \end{cases}$$

NP Hardness

- Even though this problem has been intensively investigated, exact polynomial algorithms have not been found yet.

- Furthermore, we can verify that the problem is NP-Hard by reduction to the partition set problem.

- Meta-heuristic optimization algorithm can be used to solve NP-Hard problems.

Cloud Resource Allocation

- The aim is to assign tasks to machines with objective of optimizing some measures (eg total execution time).
- Example
  - $J = \{T_1\ [2min], T_2[10min], T_3[7min]\}, \quad S = \{M_1, M_2, M_3\}$
- Same possible task to machine assignments are

	M_1	M_2	M_3
T_1	1	0	0
T_2	0	1	0
T_3	0	0	1

	M_1	M_2	M_3
T_1	1	0	0
T_2	0	0	1
T_3	0	1	0

	M_1	M_2	M_3
T_1	0	0	1
T_2	1	0	0
T_3	1	0	0

- To find the best solution, we need to explore all possible search space.
- We can use meta-heuristics approaches that constitute a swarm moving around in the search space looking for the best solution.

Simulating Cloud With CloudSim

Outline

- Introduction – Cloudsim
- Essentials to start with Cloudsim.
- Insight on Cloudsim modeled components.
- Insights on different packages available in CloudSim project
- Insight on task scheduling policies in CloudSim.

Why CloudSim?

- To evaluate the performance of Cloud application on real cloud environments such as Amazon EC2, Google App Engine, Microsoft Azure etc. is tedious, challenging and time consuming process.
  - Cloud providers exhibits varying supply and demand patterns and contains heterogeneous resources
  - Cloud users have heterogeneous and competing resource requirements.
  - Cloud applications run with different load configurations and settings
  - Testing is not flexible due to heterogeneous cloud services provided by many cloud providers.

Why CloudSim?

- Further, it is very complicated and time consuming process to re-configure the benchmarking parameters across cloud infrastructure over multiple test runs.
- Therefore, the performance evaluation is constrained by infrastructure's rigidity.
- Alternative is Simulation
  - CloudSim: well-known java based simulator proposed by Prof. Rajkumar Buyya, University of Melbourne.

CloudSim Benefits

- Hassle-free extensible modeling and event based simulation of large scale cloud infrastructure with support of virtualization engine.
- Self contained platform for modeling:
  - Clouds
  - Service brokers
  - Provisioning and allocation policies
- Flexibility to switch between
  - Space-shared
  - Time-shared allocation at all levels
- Simulation of network connections among the simulated systems elements
- Support for federated cloud environment

Features of CloudSim

- Various cloud computing data centers
- Different data center network topologies
- Message-passing applications
- Virtualization of server hosts
- Allocation of virtual machines (VMs)
- User defined policies for allocation of host resources to VMs
- Energy-aware computational resources
- Dynamic addition/removal of simulation components
- Stop and resume of simulation

Cloudsim - Essentials

- **Eclipse Java IDE:**
  https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplersr2

- **CloudSim Project Link:**
  http://code.google.com/p/cloudsim/downloads/list
  or
  https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3

- **Common math jar Link:**
  http://commons.apache.org/proper/commons-math/download_math.cgi

Important Resources

- http://www.buyya.com/papers/CloudSim2010.pdf (CloudSim Journal Paper)
- http://www.cloudbus.org/papers/CloudSim-HPCS2009.pdf (CloudSim Conf. Paper)
- https://code.google.com/p/cloudsim/wiki/FAQ (FAQs)
- https://www.superwits.com/library/cloudsim-simulation-framework (Online Course on CloudSim by Superwits Academy founded by Mr. Anupinder Singh)

Cloudsim-Directory structure

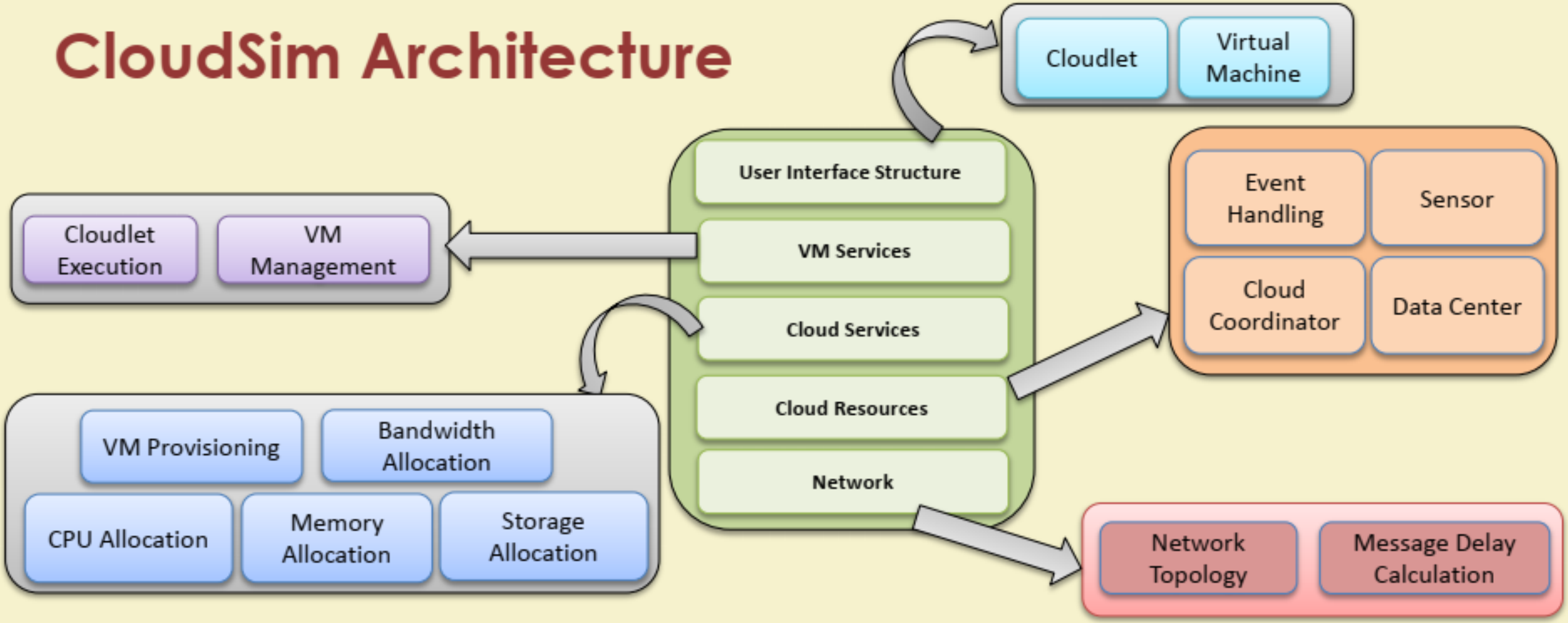
- CloudSim/ -- top level CloudSim directory

  docs/ -- CloudSim API Documentation
  examples/ -- CloudSim examples
  jars/ -- CloudSim jar archives
  sources/ -- CloudSim source code
  tests/ -- CloudSim unit tests

CloudSim jar Files

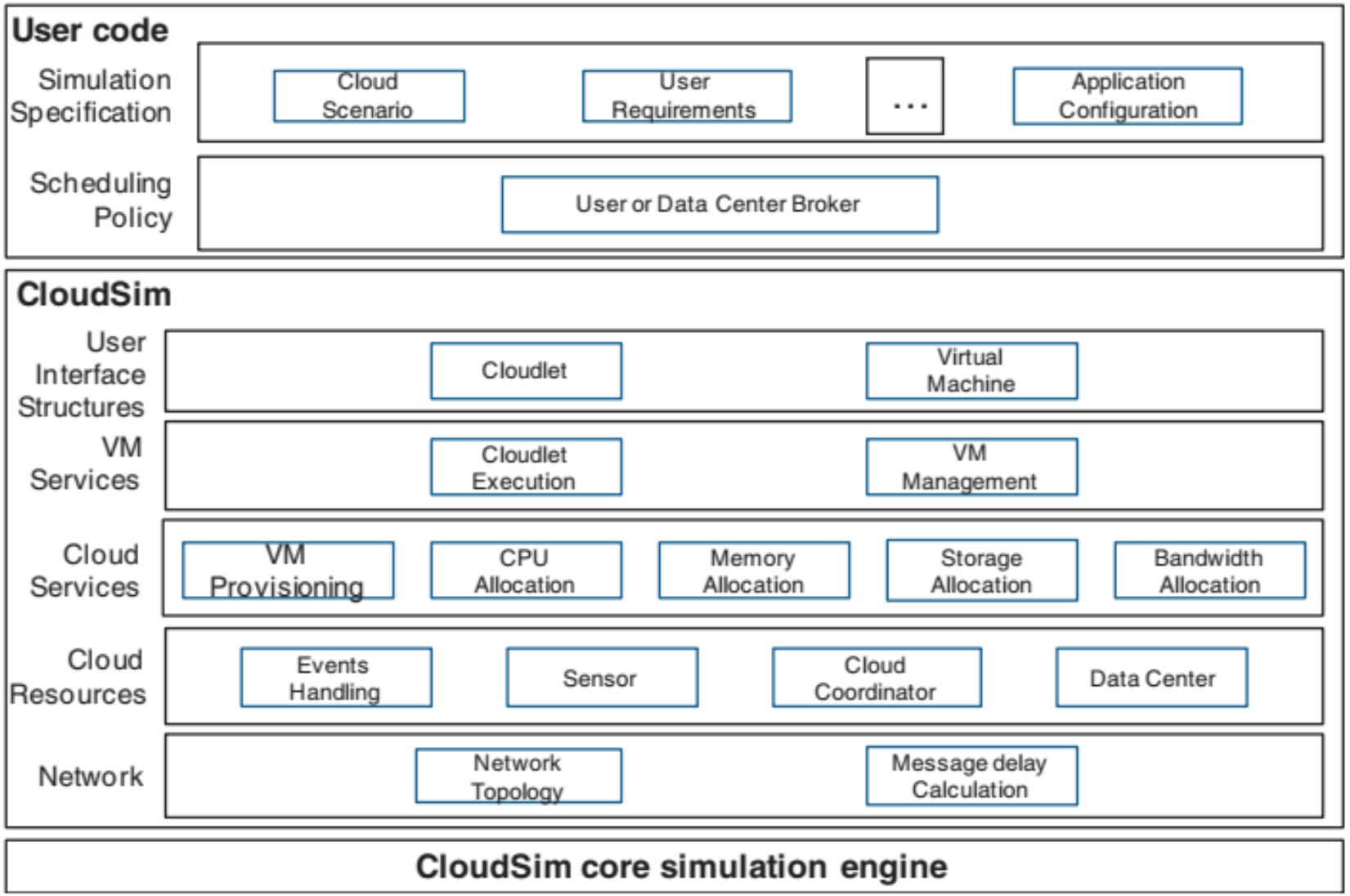
- * jars/cloudsim-<VERSION>.jar -- contains the CloudSim class files
- * jars/cloudsim-<VERSION>-sources.jar -- contains the CloudSim source code files
- * jars/cloudsim-examples-<VERSION>.jar -- contains the CloudSim examples class files
- * jars/cloudsim-examples-<VERSION>-sources.jar -- contains the CloudSim examples source code files

CloudSim Architecture



Source : Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 2011; 41(1):23–50

Cloudsim - Layered Architecture



Cloudsim - Time/Space shared models

a) VM-space, Task-space

b) VM-space, Task-time

c) VM-time, Task-space

d) VM-time, Task-time

Cloudsim - Time/Space shared models

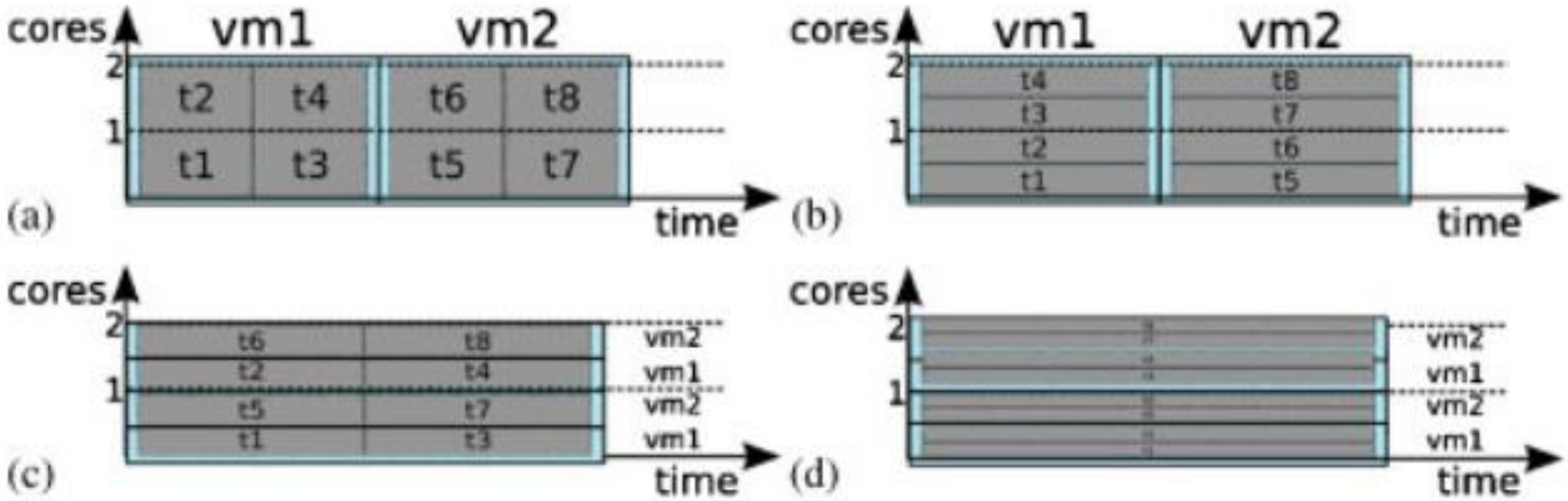


Figure 4. Effects of different provisioning policies on task unit execution: (a) space-shared provisioning for VMs and tasks; (b) space-shared provisioning for VMs and time-shared provisioning for tasks; (c) time-shared provisioning for VMs, space-shared provisioning for tasks; and (d) time-shared provisioning for VMs and tasks.

Cloudsim - Component model classes

CloudInformationService.java

Datacenter.java,Host.java,Pe.java

Vm.java,Cloudlet.java

DatacenterBroker.java

Storage.java,HarddriveStorage.java, SanStorage.java

Cloudsim - Major blocks/Modules

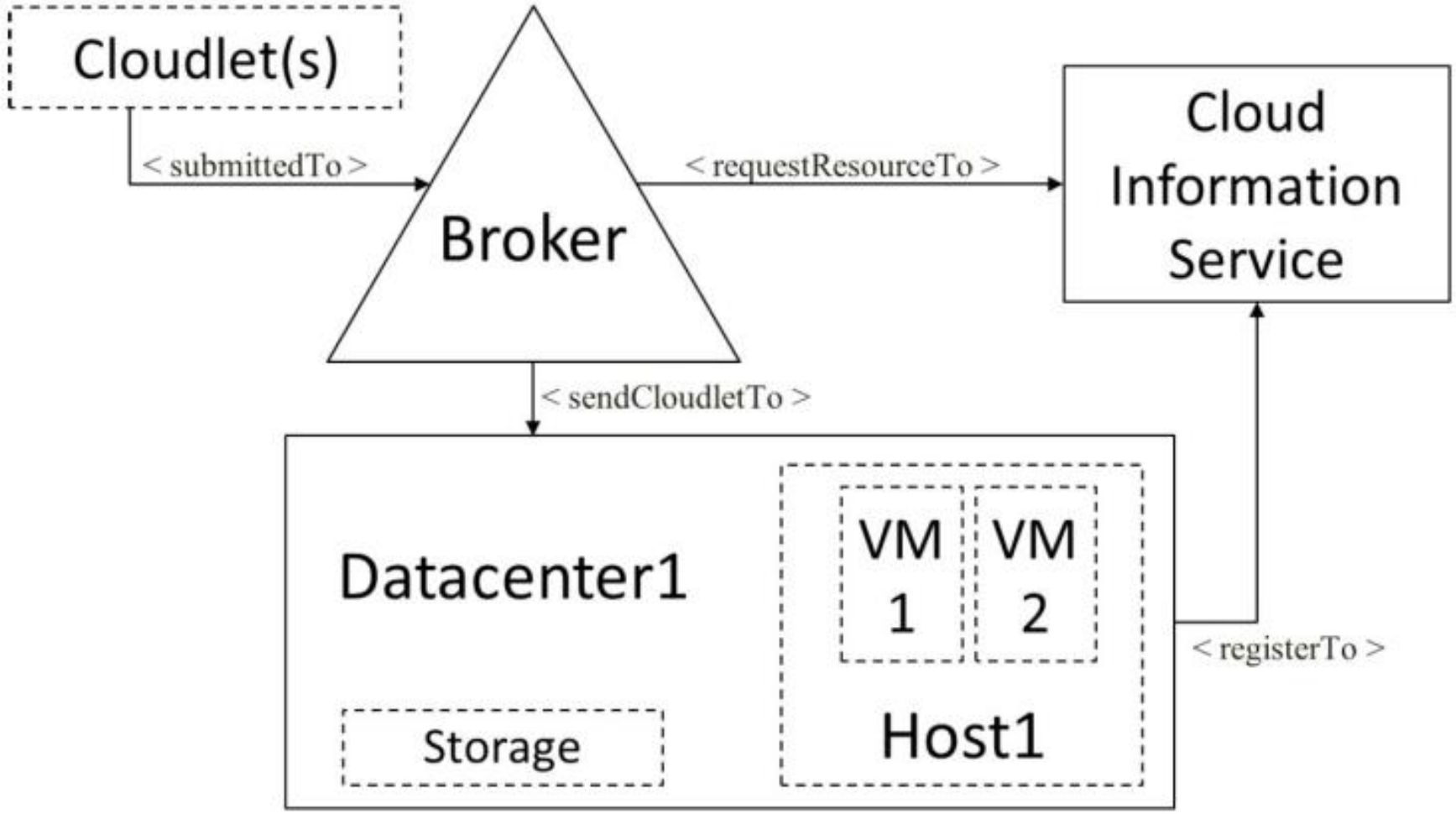
- **org.cloudbus.cloudsim**
- **org.cloudbus.cloudsim.core**
- **org.cloudbus.cloudsim.core.predicates**
- **org.cloudbus.cloudsim.distributions**
- **org.cloudbus.cloudsim.lists**
- **org.cloudbus.cloudsim.network**
- **org.cloudbus.cloudsim.network.datacenter**
- **org.cloudbus.cloudsim.power**
- **org.cloudbus.cloudsim.power.lists**
- **org.cloudbus.cloudsim.power.models**
- **org.cloudbus.cloudsim.provisioners**
- **org.cloudbus.cloudsim.util**

Cloudsim - key components

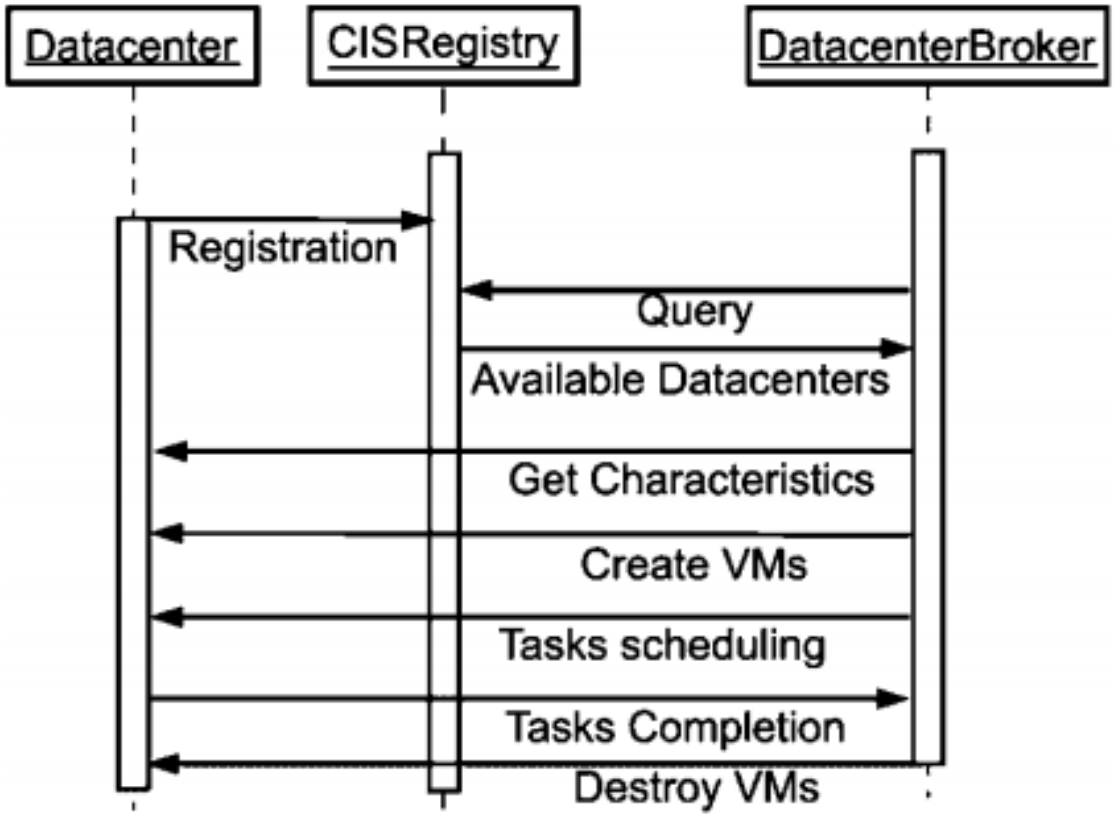
- Datacenter
- DataCenterCharacteristics
- Host
- DatacenterBroker
- RamProvisioner
- BwProvisioner
- Storage
- Vm
- VMAllocationpolicy
- VmScheduler
- Cloudlet
- CloudletScheduler
- CloudInformationService
- CloudSim
- CloudSimTags
- SimEvent
- SimEntity
- CloudsimShutdown
- FutureQueue
- DefferedQueue
- Predicate and associative classes.

Ok… so how exactly this system works?

Simulation flow for basic scenario



Simulation flow for basic scenario



Simulation data flow

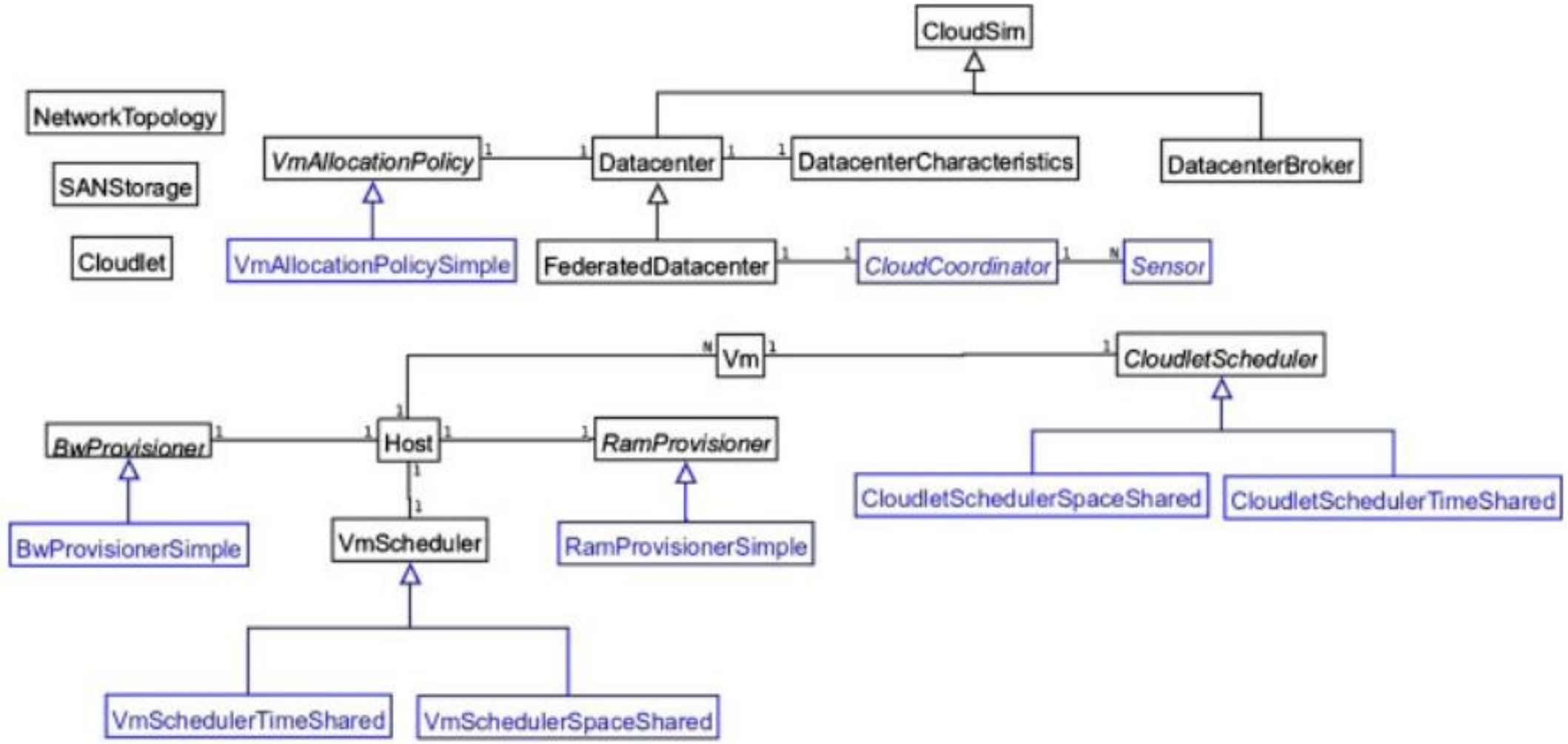
Policy

- VMAllocation Policy: used by Data Centre
  - Allocation of VM to host
  - Default: FCFS
  - Can be extended for custom allocation policies
- VMScheduler Policy: used by Host
  - Allocation of processing cores to VMs using either time-shared or space-shared policy
  - Can be extended for custom allocation policies
- CloudletScheduler policy: used by VM
  - Assigning available processing power to tasks units (Cloudlet)
- All Policies are time shared and space shared.

Cloudsim - General steps to follow

- set number of users (Brokers count)
- initialize the Common variables
- create CIS (Object of CIS is created).
- create a datacenter (Host + their Characteristics e.g. PE, RAM, BW, storage).
- create datacenter broker instance (datacenter ↔VM/Cloudlets)
- create VMs/cloudlet add it to respective lists.
- submit VM and cloudlet list to broker.
- start the simulation process.
- stop the simulation process.
- print the end results.

Design and Implementation of CloudSim



- ***BwProvisioner***
  - abstract class that models the policy for provisioning of bandwidth to VMs
  - allocation of network bandwidths to a set of competing VMs that are deployed across the data center
  - can extend this class with their own policies (priority, QoS) to reflect the needs of their applications
  - *BwProvisioningSimple* allows a VM to reserve as much bandwidth as required

- ***Cloudlet***
  - models the Cloud-based application services (such as content delivery, social networking, and business workflow)
  - orchestrates the complexity of an application in terms of its computational requirements
  - can also be extended to support modeling of other performance and composition metrics for applications such as transactions in database-oriented applications

- ***CloudletScheduler***
  - This abstract class is extended by the implementation of different policies that determine the share of processing power among Cloudlets in a VM
  - two types of provisioning policies are offered:
    - space-shared (***CloudetSchedulerSpaceShared***)
    - time-shared (***CloudletSchedulerTimeShared***).

- ***Datacenter***

  - models the core infrastructure-level services (hardware) that are offered by Cloud providers (Amazon, Azure, App Engine)
  - Both homogeneous or heterogeneous hosts
  - instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices to hosts and VMs

- ***DatacenterBroker or Cloud Broker***
  - models a broker, which is responsible for mediating negotiations between SaaS and Cloud users; and such negotiations are driven by QoS requirements
  - extend this class for evaluating and testing custom brokering policies
- ***DatacenterCharacteristics***
  - This class contains configuration information of data center resources

- ***Host***
  - models a physical resource such as a compute or storage server
  - Encapsulates important information such as:
    - amount of memory and storage
    - a list and type of processing cores (to represent a multi-core machine)
    - an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs.

CloudSim Extensions

iFogSim

- iFogSim enables modelling and simulation of **Fog computing environments** for evaluation of resource management and scheduling policies across edge and cloud resources under different scenarios.
- supports evaluation of resource management policies focusing on their impact on latency (timeliness), energy consumption, network congestion and operational costs
- simulates **edge** devices, **cloud** data centers, and **network** links to measure performance metrics.
- **Sense-Process-Actuate model**

CloudSimEx

- **Currently CloudSimEx features:**
  - Web session modeling;
  - Better logging utilities;
  - Utilities for generating CSV files for statistical analysis;
  - Automatic id generation;
  - Utilities for running multiple experiments in parallel;
  - MapReduce simulation.
- **Note!** These extensions are not officially supported by the CloudSim team until they are integrated with CloudSim.

EdgeCloudSim

- simulation environment specific to Edge Computing scenario
- adds some additional functionalities to **CloudSim** such as network modeling specific to WLAN and WAN, device mobility model, realistic and tunable load generator.

WorkflowSim

- extends the CloudSim simulation toolkit by introducing the support of workflow preparation and execution with an implementation of a stack of
  - workflow parser
  - workflow engine
  - job scheduler
- popular workflow scheduling algorithms (e.g., HEFT, Min-Min, and Max-Min) and task clustering algorithms have been implemented in WorkflowSim.

CloudAuction

- enables CloudSim to handle auction-based services
- Implements Combinatorial Double Auction based market mechanisms that allocated cloud resource to cloud users via broker.
- Considers both the user and provider.

FederatedCloudSim

- a versatile and flexible extension to the CloudSim framework.
- allows for a multitude of cloud federation experiments
- supports SLAs and offers a three level scheduling approach for VMs
  - in data centers
  - between data centers of the same cloud service provider (CSP)
  - between CSPs in a federation

CloudAnalyst

- Simulation tool designed based on CloudSim
- Provides GUI
- Supports geographically distributed large-scale Cloud applications
- The purpose is to study the behavior of such applications under various deployment configurations

References

- Thyagaraj Thanalapati, Sivarama P. Dandamudi: An Efficient Adaptive Scheduling Scheme for Distributed Multicomputers. IEEE Trans. Parallel Distrib. Syst. 12(7): 758-768 (2001)

- A. Iosup and D.H.J. Epema, Grid Computing Workloads, IEEE Internet Computing 15(2): 19-26 (2011)

- Feitelson DG. Workload modeling for computer systems performance evaluation. Cambridge University Press; 2015

References

- Zahra Pooranian · Mohammad Shojafar · Jemal H. Abawajy · Ajith Abraham, An efficient meta-heuristic algorithm for grid computing
  • Habib Shah, Tutut Herawan, Rozaida Ghazali, Rashid Naseem, Maslina Abdul Aziz, Jemal H. Abawajy: An Improved Gbest Guided Artificial Bee Colony (IGGABC) Algorithm for Classification and Prediction Tasks. ICONIP (1) 2014: 559-569
  • Nazri Mohd Nawi, Abdullah khan, M.Z. Rehman, Maslina Abdul Aziz, Tutut Herawan, and Jemal H. Abawajy," Neural Network Training by Hybrid Accelerated Cuckoo Particle Swarm Optimization Algorithm", Springer International Publishing Switzerland 2014 .
  • Nazri Mohd Nawi, Abdullah Khan, M. Z. Rehman, Maslina Abdul Aziz, Tutut Herawan, Jemal H. Abawajy: An Accelerated Particle Swarm Optimization Based Levenberg Marquardt Back Propagation Algorithm. 2014: 245-253
  • Nazri Mohd Nawi, Abdullah Khan, M. Z. Rehman, Maslina Abdul Aziz, Tutut Herawan, Jemal H. Abawajy: Neural Network Training by Hybrid Accelerated Cuckoo Particle Swarm Optimization Algorithm. 2014: 237-244

Thank You

Questions…?