10. Distributed Computing

Sistemes Distribuïts en Xarxa (SDX)
Facultat d’Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC)
2024/2025 Q2

Contents

- **Volunteer computing**
- Grid computing
- Cloud computing

Motivation

- Supercomputers are too expensive
  - e.g. Jaguar (#1 TOP500 2010) cost $104 million
  - Few institutions can afford this level of investment
- More than 1 billion PCs around the world
  - Some as powerful as early 90s supercomputers
  - They are idle most of the time (60% to 90%), even when being used (typing, printing,...)

⇒ Exploit spare computing power on end-user PC computers for computationally-intensive problems

Target applications

- Work can be partitioned in independent units
- Work-units can be executed in parallel
- Large computation to communication ratio
  - No communication or coordination between nodes while they are processing the work-units
  - Send results to server in a single short message whenever the client and server are available
- e.g. computationally-intensive scientific applications: biology, astronomy, …

➢ http://en.wikipedia.org/wiki/List_of_distributed_computing_projects

SETI@home project

- http://setiathome.ssl.berkeley.edu
- SETI: Search for Extraterrestrial Intelligence
- Started in 1999 to enlist PCs to analyze data from the Arecibo radio telescope
- First popular distributed computing project
- Ended in March 2020
- Stats (Mar 2020)
  - More than 1,8 million users (91.000 active)
  - More than 165.000 active hosts
  - Over 1 PFLOPS

Berkeley Open Infrastructure for Network Computing

- http://boinc.berkeley.edu/
- Open-source middleware for volunteer computing
- Started on 2002 to support SETI@home project
- Freed to support computationally-intensive projects in a wide range of disciplines
  - Mathematics (Collatz Conjecture, Primaboinca), Physics & Astronomy (SETI@home, Einstein@home, LHC@home, Milkyway@home), Earth Science (Climateprediction.net, Quake-Catcher Network), Biology (Rosetta@home)
- Stats (May 2019): 50 active projects, 28 PFLOPS

BOINC architecture

- Some project-operated servers and a client program running on volunteer's computer
  - Volunteer trusts that the client's program will not damage their computer or invade their privacy
- BOINC uses a pull model
  - Client periodically contacts the server to request **work-units** (i.e. computations to be run) and report the results of completed ones
    - Application has to be adapted to be partitioned
  - Server validates each volunteer's work (through redundant executions) and grants some "credit"

Folding@home project

- http://foldingathome.org/
- Enlist PCs to work on the protein folding and related diseases
- Launched on October 2000, not BOINC-based
- Clients can harness the power of PlayStation 3s (2007/12) and GPGPUs (2006/-)
- First computing system to attain 1 PFLOPS (2007) and 1 EFLOPS (2020)
  - For several years, it exceeded the performance of TOP500's fastest supercomputer

Contents

- Volunteer computing
- **Grid computing**
- Cloud computing

Motivation

- Requirements of science in the Internet era
  - Construct and access large databases
  - Develop efficient large simulations & analyses
  - Access specialized devices remotely
  - Exchange information within distributed multidisciplinary teams
- Need for a distributed platform that supports seamless and efficient <u>data sharing and coordination of computation on a large scale</u>

Typical Grid applications

- Complex problems with several organizations collaborating and sharing resources

1. Computationally-intensive applications
   - e.g. climate/weather modeling, galaxy formation, fault diagnosis, financial modeling, earthquake simulation
     - NEESgrid Earthquake Engineering Collaboratory
     - Rolls-Royce Distributed Aircraft Engine Diagnostics
2. Data-intensive applications
   - e.g. multimedia DB, medical/scientific data federation
     - The World-Wide Telescope
     - US Biomedical Informatics Research Network
3. Apps using special devices (e.g. lab equipment)

Grid definition

- Distributed system that enables **seamless aggregation** and **sharing** among **dynamic collections** of individuals and/or institutions of **geographically distributed** resources and services owned by **different organizations** administered with different policies **for some common purpose** in a flexible, secure & efficient manner

Grid resources

- Grid resource: computers, clusters, servers, storage, data, utilities, applications, etc.

a) Can **dynamically** join/leave the Grid

b) Are **heterogeneous** in every aspect

c) Are **geographically distributed** and connected by a wide-area network

d) Can be accessed **on demand** by the users

e) May be owned by **diverse organizations**

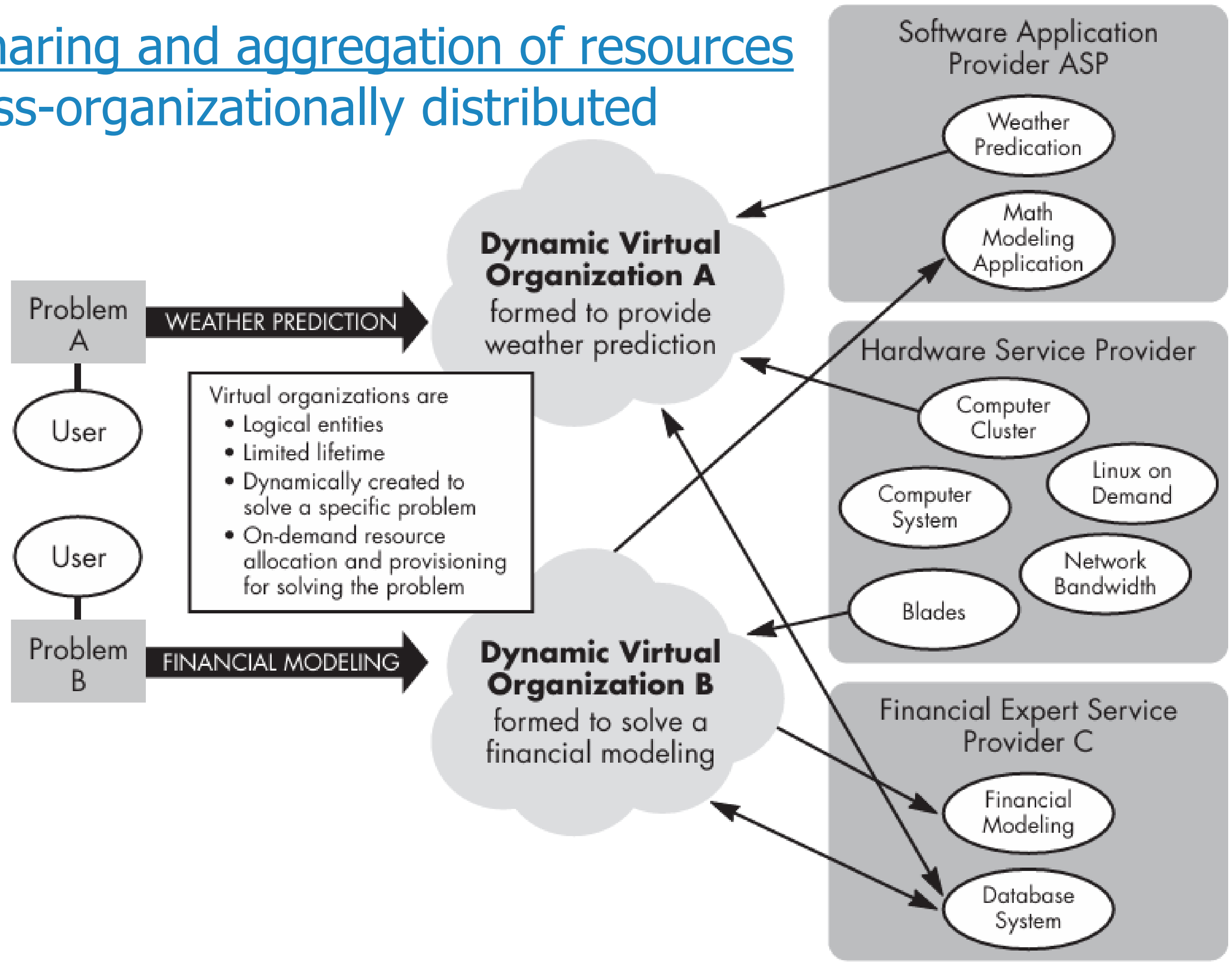
f) Can be **transparently** accessed

Power Grid analogy

- The term ‘Grid’ in computing is an analogy to electrical power grids
- Grid provides computing on demand to users like the power grid provides electricity
  - Seamless, high-quality, dependable, ubiquitous
    - Transparent access with respect to the source through uniform interfaces
  - Some differences:
    - Wider spectrum of services and performance
    - Access governed by more complicated issues: security, performance, multi-ownerships, funding & political issues

Virtual organizations (VO)

VOs enable sharing and aggregation of resources which are cross-organizationally distributed



Grid architecture

- Grid has gradually shifted toward a Service-Oriented Architecture (SOA)
  - 'Everything is a service'
    - Grid resources become available as Web services
    - Use WSRF, SOAP, XML to describe/access Grid services
- Open Grid Services Architecture (OGSA)
  - Specification defining the overall structure & capabilities to be supported in a Grid system
    - Defines the functionality and interfaces of common services for Grid applications: e.g. Infrastructure, Execution management, Data, Information, Security, Self-management, Resource management

Grid related paradigms

1. **Cluster computing**
   - Essentially a group of workstations connected through a LAN
   - Tightly coupled and homogeneous
     - Same OS, near-identical hardware
   - Physically contained in a single location
2. **Volunteer computing**
   - Loosely coupled, heterogeneous, and geographically distributed
   - Tend to be specialized systems intended for a single purpose or user group

Grid related paradigms

3. Grid computing

- Large scale (connected over a WAN)
- <u>Heterogeneous</u> computer hardware, operating systems, programming languages and applications
- <u>Tightly coordinated</u> to allow the collaboration of groups of people or institutions
- Dispersed across several organizations
- Geographically distributed (not physically coupled)
- <u>Transparent</u>: User has no knowledge about the underlying topology or any individual node

Grid related paradigms

- Grid computing
  - Established target communities
    - Science, industry
    - Restricted participation
  - Resources
    - More diverse (in type)
    - More powerful
    - Good availability
    - Well connected
  - Standard protocols (Global Grid Forum) & middleware (Globus Toolkit)

- Volunteer computing
  - Anonymous individuals
  - Resources
    - Computing cycles or files
    - Less powerful
    - Intermittent participation
    - Variably connected
  - Each application defines & deploys completely independent infrastructure

Contents

- Volunteer computing
- Grid computing
- **Cloud computing**

Contents

- Volunteer computing
- Grid computing
- **Cloud computing**
  - **Cloud basics**
  - Cloud computing services
  - Enabling the Cloud
  - Obstacles for Cloud computing

Definition

- Emerging style of network-based computing
- Applications, data, and IT resources provided **on demand** to users over the Internet
  - → Scale up and down in capacity and functionalities
  - → Anytime, anywhere access delivered dynamically **as a service**
    - As with Grid computing, the Power Grid analogy applies
- Reduces requirements on users' devices
  - – Very simple desktop or portable devices can access a wide range of resources and services

Definition

- Remote facilities (a.k.a. data centers) store and compute all the data and applications
  - Elimination of the upfront commitment of users
  - Infinite computing resources available on demand
- Cloud infrastructures are transparent to users and applications
  - The infrastructure can take on many forms, but the implementation is irrelevant to the end user, hence, the "Cloud" abstraction
  - Users and applications interface with the Cloud infrastructure via APIs

Utility computing

- When computing resources in the Cloud are offered to users in a <u>pay-as-you-use</u> manner
  - Users avoid hardware acquisition costs, software licenses or upgrades to manage, new employees or consultants to hire, facilities to lease …
- New business model (a.k.a. Public Cloud)
  - Build a large infrastructure and rent out storage, computation, etc. based on user’s demand
  - Think of computing as a metered service, just like electric power, natural gas, or water
  - e.g. Amazon’s EC2

Types of Clouds

- Private Cloud
  - Provisioned for exclusive use by a single organization and operated following the Cloud principles
- Public Cloud
  - Run by third parties and made available for open use by the general public in a pay-as-you-go manner
- Community Cloud
  - Provisioned for exclusive use by a specific community of consumers from organizations with shared concerns
- Hybrid Cloud
  - Composition of distinct cloud infrastructures bound together to enable portability (e.g. cloud bursting)

Cloud vs. Grid

- Grid is a precursor of Cloud, hence they share the same vision …
  - Offer resources on demand through Internet
  - Reduce the cost of computing
  - Increase reliability and flexibility by transitioning from self-operation to third party
- … and the challenges are very similar
  - Management of large facilities
  - Methods to discover, request, and use resources
  - Parallelization/distribution of large-scale computations/data within data center facilities

Cloud vs. Grid

- However, Cloud has developed significantly due to requirements of the new scenario
  - Massive scale
    - Serve millions of users, manage huge infrastructures
    - Massive data analysis, increased demand for computing
  - We have low-cost virtualization
  - Large companies (Amazon, Google, and Microsoft) have created real commercial large-scale systems
  - Everybody can access ‘infinite’ computing resources as a utility using only a credit card

Cloud vs. Grid

- Grid computing
  - Funded by government
  - User base in academia and government labs to drive scientific computing
    - More HPC-oriented apps
  - Compute model
    - Batch scheduled
  - CPU hours per project
  - Strong security model
  - Virtualization
    - Only partial adoption

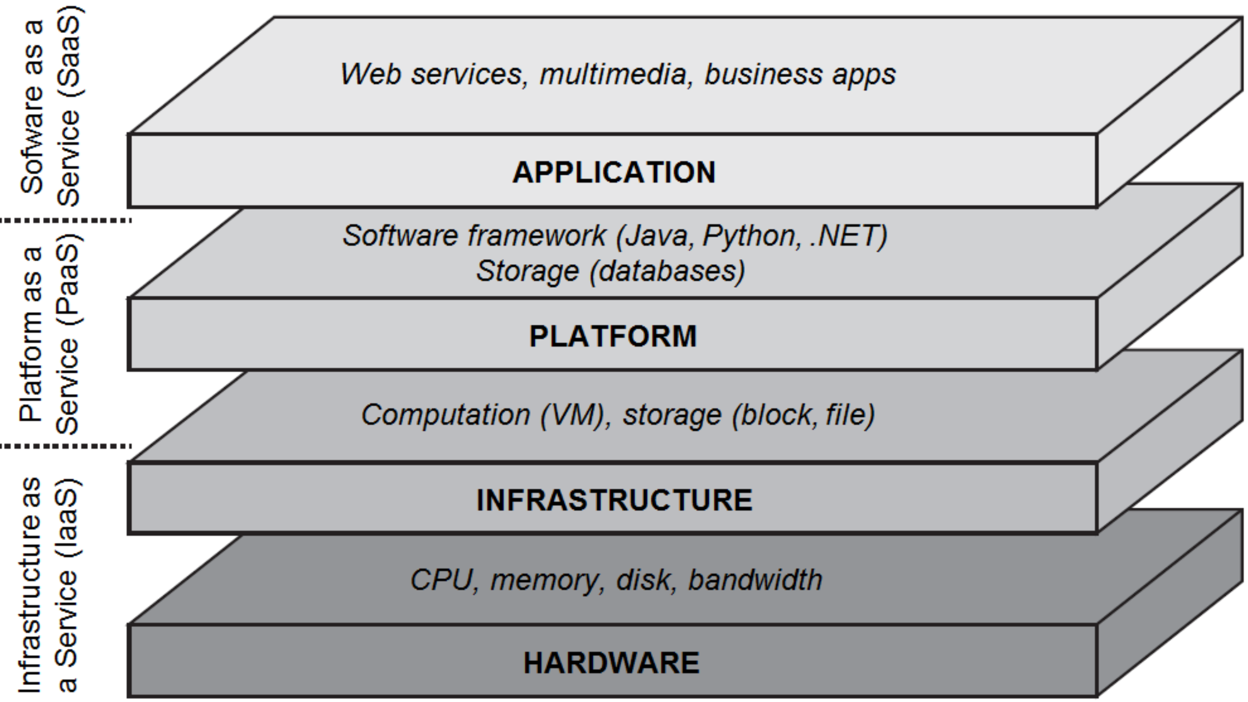
- Cloud computing
  - Funded by industry
  - User base in common people and business
    - More transactional and interactive apps
  - Compute model
    - Time sharing
  - Utility computing: money
  - Weak security model
  - Virtualization
    - Key technology for abstraction/encapsulation

Contents

- Volunteer computing
- Grid computing
- **Cloud computing**
  - Cloud basics
  - **Cloud computing services**
  - Enabling the Cloud
  - Obstacles for Cloud computing

Cloud computing services

- Applications, data & IT resources in the Cloud offered as services at various layers
  - **Service-Oriented Architecture** (SOA)



Infrastructure as a Service (IaaS)

- Delivers fundamental computing capabilities (storage, processing, networking, …) as standardized services over the network
- Resources are abstracted / encapsulated (usually by virtualization) so that they can be exposed to upper layer and end users as integrated resources (e.g. virtual computer)
- Allows fulfilling hardware needs of an organization in a rapid and affordable way
  - Virtual resources can be ready in short time

Infrastructure as a Service (IaaS)

- Example: Amazon Simple Storage Services (S3) and Elastic Compute Cloud (EC2)
  - S3 provides data storage at a cost of about 3 cents per GB per month. EC2 provides computing power. Pricing is based on computing power (measured in terms of EC2 compute units) and the amount of memory (measured in GB)
    - e.g. a m5.large instance is defined as 2 VCPUs, 8 GB of memory, 8 EC2 compute unit
      - This would cost 10 cents per hour of usage
  - EC2 servers are Linux-based virtual machines running on top of the Xen virtualization engine

Platform as a Service (PaaS)

- Adds the software needed to develop and deploy Cloud services to the IaaS platform
  - e.g. Web hosting environment
- Makes it easy for application developers to build and deploy their applications
  - Hide the complexity of underlying IaaS
- Example: Google App Engine
  - Lets developers write web applications in Node.js, Java, Ruby, C#, Go, Python, or PHP, and host them on Google infrastructure

Software as a Service (SaaS)

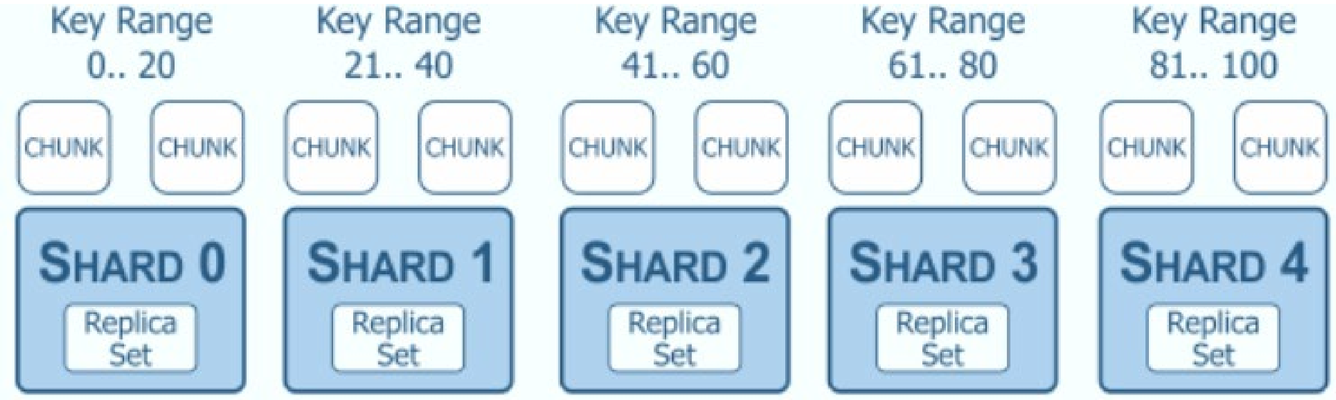
- A complete application is provided as a service to users in an on-demand fashion
- It alleviates the customer's burden of software installation and maintenance
- Users do not need to procure hardware
- Example: Google Apps
  – Docs, Gmail, Calendar

Contents

- Volunteer computing
- Grid computing
- **Cloud computing**
  - Cloud basics
  - Cloud computing services
  - **Enabling the Cloud**
  - Obstacles for Cloud computing

Cloud enabling technologies

A. Sophisticated data storage systems for handling huge amounts of data
   - (Key-based) sharding and (eventually-consistent) replication to achieve scalability and availability
     - Split a large dataset into smaller chunks and distribute them across a large number of nodes (shards)
     - Replicate each shard in a small number of nodes

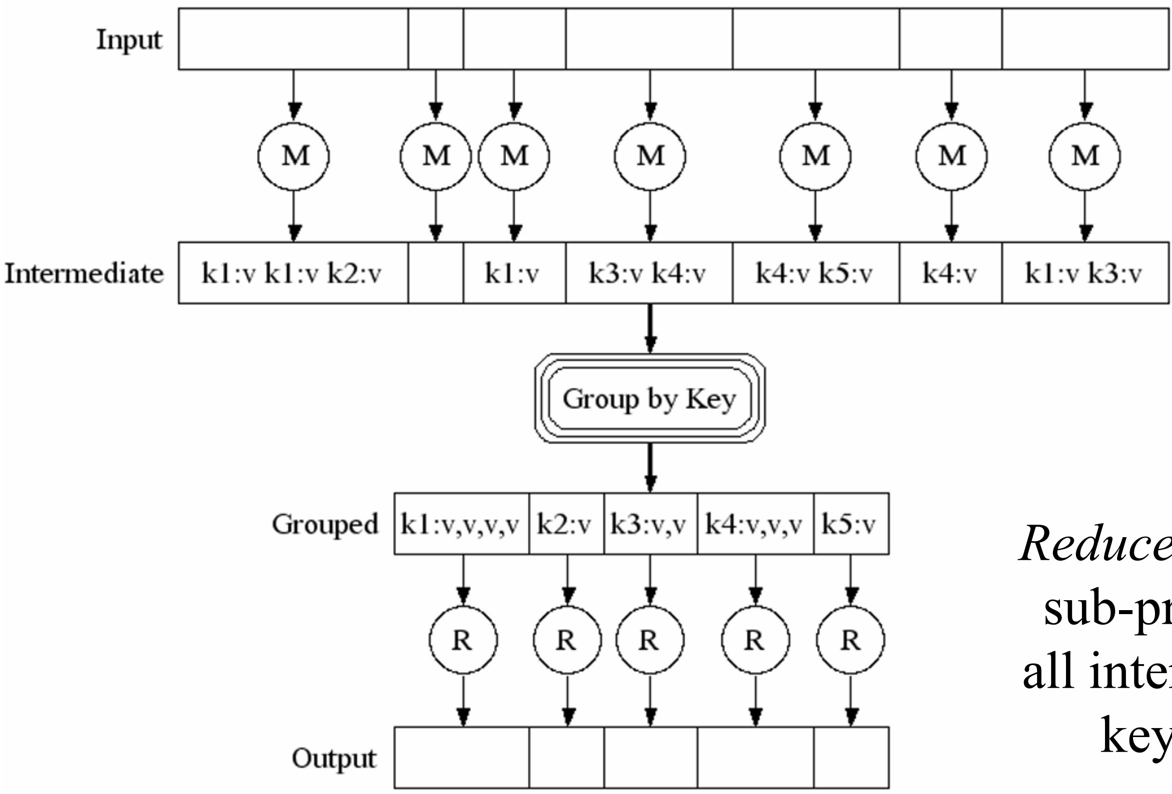


Cloud enabling technologies

A. Sophisticated data storage systems for handling huge amounts of data
   - Sharding allows parallel access to different chunks
   - Replicas maintain data availability on node crash
   - Eventual consistency is enough when applications can tolerate inconsistent data for limited periods
     - Replicas see the same updates in the same order, hence the system eventually converges to a consistent state
   - e.g. File systems: Google FS, Hadoop FS
   - e.g. NoSQL stores: Cassandra, Dynamo, MongoDB

Cloud enabling technologies

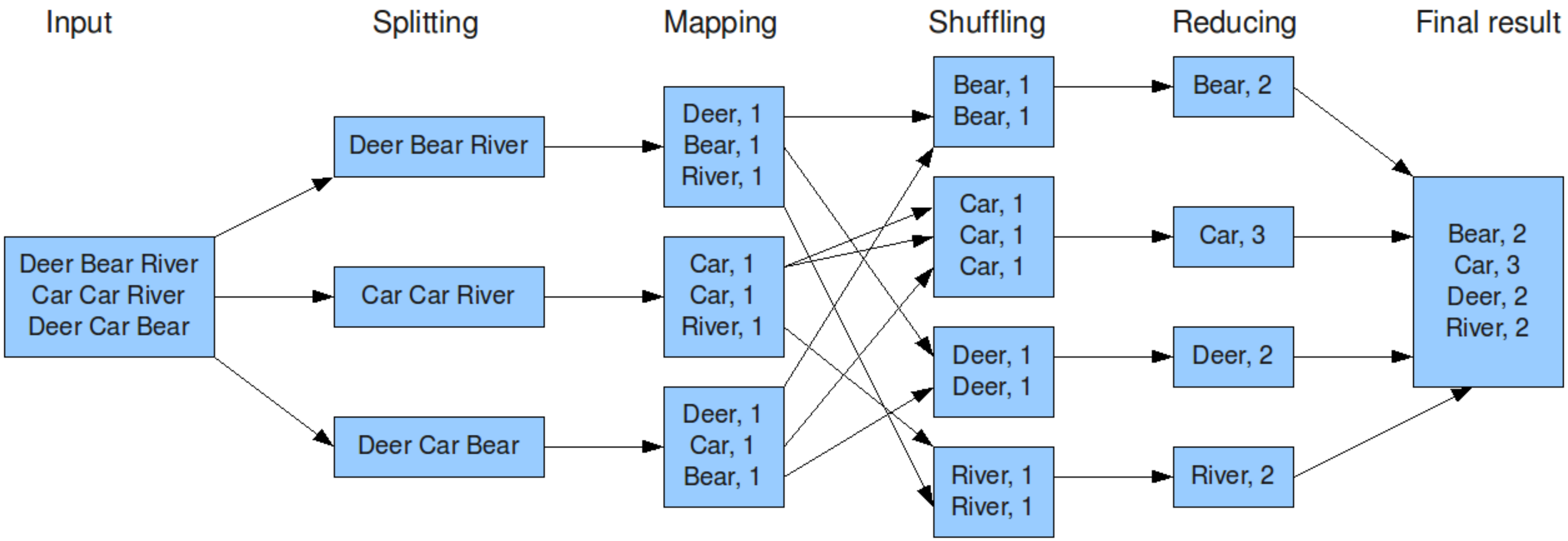
B. Programming models for processing large data sets (e.g. **MapReduce**)



Map: Divide input into smaller sub-problems and process them in parallel to generate intermediate key/value pairs

Reduce: Collect the answers to all the sub-problems and merge in parallel all intermediate values associated per key to form the global answer

MapReduce: word count example



- map(key=doc name, value=doc contents):
  - For each word in contents, emit {word, 1}
- reduce(key=word, values=list of {word, 1}):
  - Sum word occurrences in values list
  - Emit {word, sum}

MapReduce execution

- A Resource Manager (e.g. YARN) assigns Map and Reduce tasks to nodes
  - They can be easily distributed to different nodes because they are independent, but it should:
    - Be fair across the different users/jobs
    - Favor data locality (e.g. attempt to schedule a task on a node with a replica of corresponding input data)
    - Balance the load to equalize the amount of work per node and the response time of each task
    - Run some tasks redundantly to deal with stragglers (slow tasks which slow the entire job)
    - Deal with failures (e.g. restart failed tasks)

Cloud enabling technologies

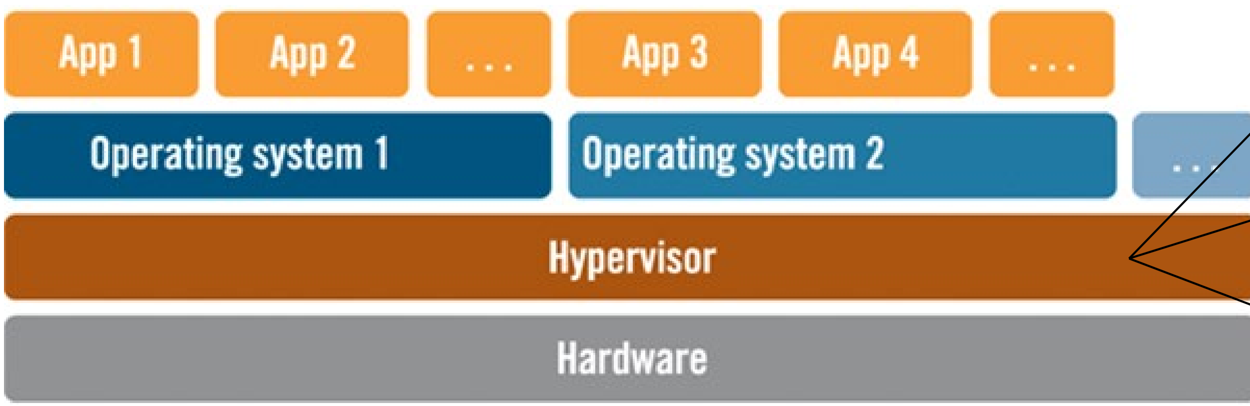
C. Virtualization technology
   - Creates the illusion of multiple dedicated (and customized) systems on the same physical system
   - Isolates programs from the underlying system and from other programs
   - Virtual environments can be created and destroyed readily and with little overhead
   - Facilitates the portability of programs
   - Allows agile and fine-grain dynamic resource provisioning (including migration)
   - Enables fault tolerance for programs

Hardware virtualization

- Add a software hypervisor to emulate enough hardware to allow an unmodified guest OS to run in isolation
  - Hypervisor a.k.a. Virtual Machine Monitor (VMM)
- The hypervisor …
  - Virtualizes all the physical systems for each VM
  - Multiplexes running VMs across the physical resources of the machine
  - Catches and emulates sensitive instructions issued by guest OS to control the HW
    - Non-sensitive instructions can be executed directly

Hardware virtualization

I. Bare-metal (native) hypervisors (Type I)
   - VMM runs directly on top of the hardware and acts as OS: provides virtualization & OS functionality
   - Provides better performance and greater flexibility and can support a large number of instances
   - Typically deployed on server systems and production environments



VMware ESX

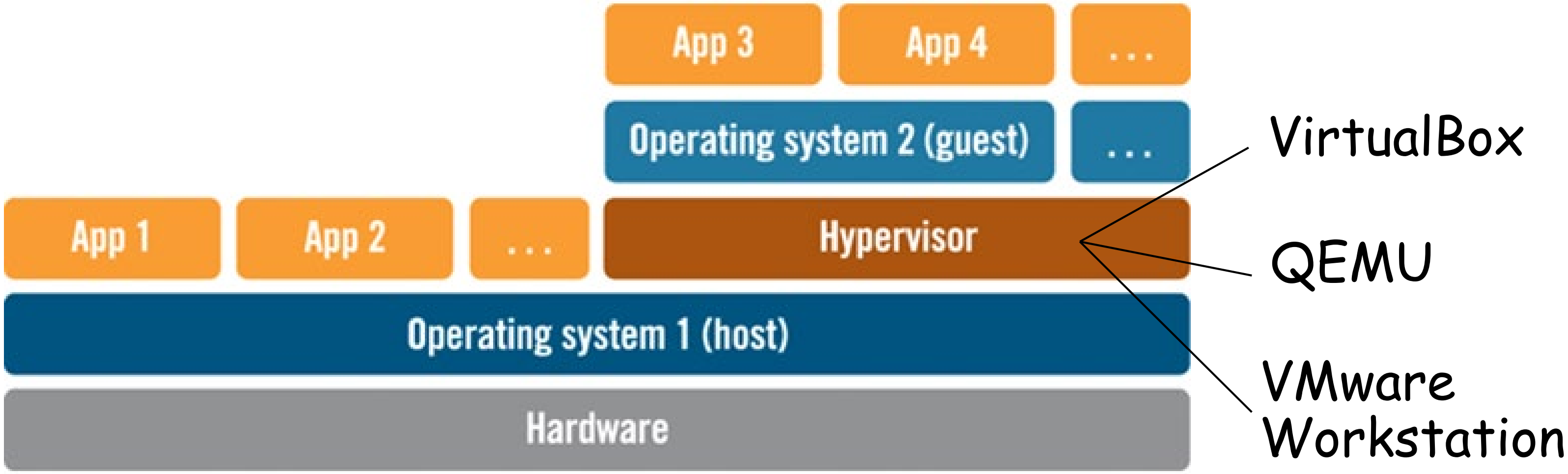
Xen

Hyper-V

Hardware virtualization

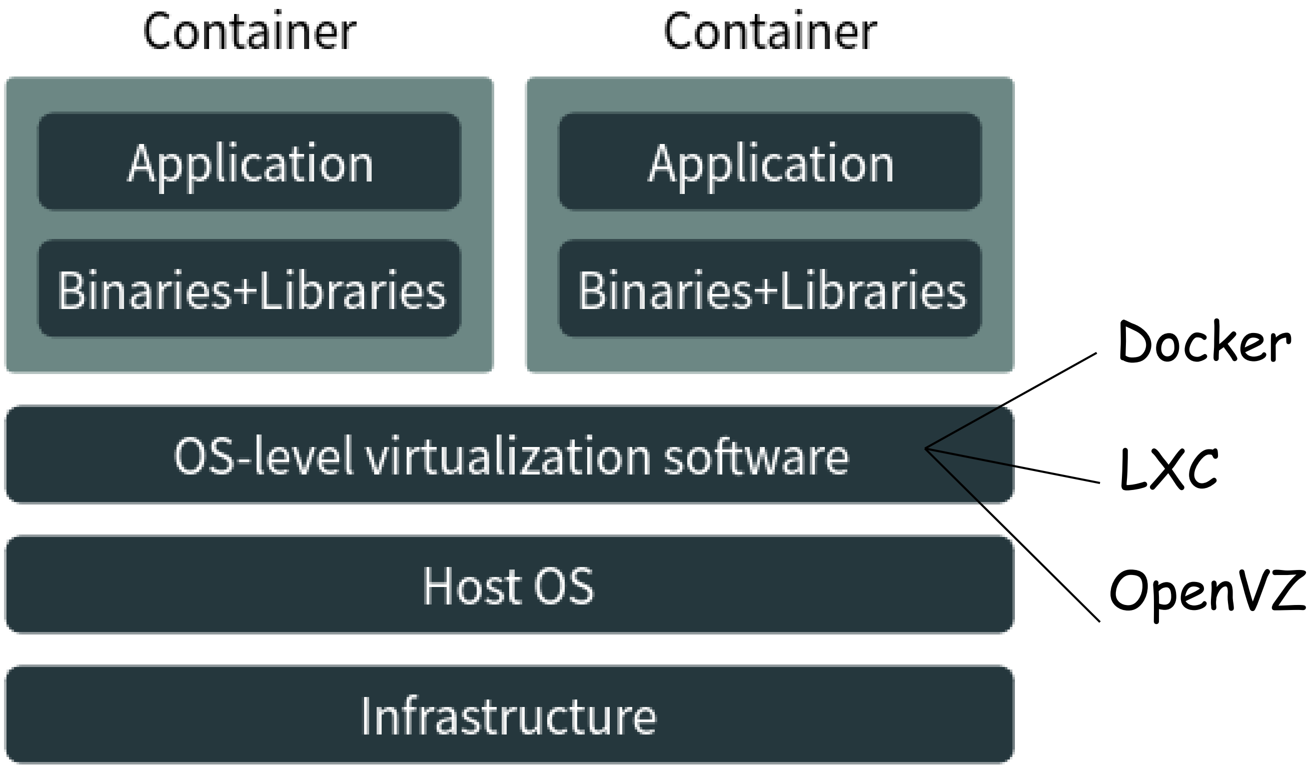
II. Hosted hypervisors (Type II)

- VMM runs as a user-level process on top of an existing OS which provides part of its functionality
- Host OS layer adds latency: less performance
- Typically deployed on commodity devices and development/testing environments



OS-level virtualization

- Multiple isolated user-space instances (a.k.a. **containers**) share a single operating system



OS-level virtualization

- A container 'feels' like a virtual machine …
  - Each container has its own process space
  - Each container has its own network interfaces
  - A container can run stuff as root
  - A container can install packages
  - A container can get a shell on it
- … but all of them share the same OS
  - Little overhead as no emulation is needed
  - Isolation is achieved by changing what containers can see about the OS: it is a **namespace** game

Cloud research at BSC-UPC

- Energy-aware computing for Edge/Cloud
  - Goal: Manage Edge/Cloud providers to maximize energy and ecological efficiency when dealing with heterogeneous workloads and hardware

- Virtualization for HPC / BD / AI convergence
  - Goal: Leverage virtualization technologies to manage supercomputers aiming to improve the performance of HPC, Big Data, and AI workloads

- For more information, contact me!

Contents

- Volunteer computing
- Grid computing
- **Cloud computing**
  - Cloud basics
  - Cloud computing services
  - Enabling the Cloud
  - **Obstacles for Cloud computing**

Obstacles for Cloud computing

- Requires a constant Internet connection
  - Does not work well with slow connections
  - Can be slow, even with a fast connection
- Stored data might not be secure
  - Who can access your data?
  - How trustable is your provider?
- Stored data can be lost
  - What about data recovery and data integrity?
- Political and legal issues
  - Who owns the data? ; Where is your data?
  - Who uses your personal data?

Obstacles for Cloud computing

Armbrust, M., et al., *A View of Cloud Computing*, Communications of the ACM, Vol. 53, No. 4, April 2010, pp. 50-58

1. Ensure high availability for services
2. Avoid data lock-in
   - Easily extract customers' data from one site to run on another, e.g. by using standard APIs
3. Ensure data confidentiality and auditability
4. Enhance data transfers (in time and cost)
5. Reduce performance unpredictability
   - Because of network and disk I/O sharing among VMs and lack of control on VM scheduling

Obstacles for Cloud computing

6. Scalable storage
7. Bugs in large distributed systems
   – Support for large-scale distributed debugging
8. Scaling quickly
   – Automatically scale up and down in response to load to save money, but without violating SLA
9. Reputation fate sharing
   – Bad behavior can affect the reputation of others
   – Transfer of legal liability for customer's behavior
10. New pay-for-use software licensing models

Summary

- Volunteer computing applies the P2P concept for distributed computation
  - e.g. SETI@home, BOINC, Folding@home

- Grid paradigm to solve large-scale problems in science, engineering, and commerce
  - Enables different partners within a virtual organization to share geographically distributed resources owned by different organizations for some common purpose

Summary

- Cloud computing provides IT capabilities as a service to users over the Internet
- Generally comes with Utility computing
  - Rent services to clients in a pay-as-you-use way
- Several abstraction layers: IaaS, PaaS, SaaS
- Novel enabling technologies: virtualization, data storage, programming models
- Further details:
  - [Tanenbaum]: chapters 1.3.1, 2.5.1, and 3.2
  - [Coulouris]: chapters 7.7, 9.7, and 21.6