CT30A3401
Distributed Systems
Lecture 2

Bilal Naqvi, PhD.
syed.naqvi@lut.fi

Designing distributed systems

- Design of distributed system a key towards their desired operation.
- Four important goals
  - make resources easily accessible
  - hide the fact that resources are distributed across a network
  - be open
  - be scalable

Supporting resource sharing

- Goal: To make it easy for users (and applications) to access and share remote resources
  - Resources can be virtually anything, but typical examples include peripherals, storage facilities, data, files, services, and networks
- Example: file-sharing peer-to-peer networks like BitTorrent

Making distribution transparent

- Goal: to hide the fact that processes and resources are physically distributed across multiple computers possibly separated by large distances
- Types of distribution transparency
  - Access transparency
  - Location transparency
  - Relocation transparency
  - Migration transparency
  - Replication transparency
  - Concurrency transparency
  - Failure transparency

Making distribution transparent (continued)

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

Reference: M.V Steen, A.S Tanenbaum, (2018) Distributed Systems 3rd edition

Being open

- Goal: To ensure that a system offers components that can easily be used by, or integrated into other systems
- Three aspects:
  - Interoperability: the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other's services as specified by a common standard.
  - Portability: extent an application developed for a distributed system A can be executed, without modification, on a different distributed system B that implements the same interfaces as A.
  - Extensibility: extent to which it is easy to add new components or replace existing ones without affecting those components that stay in place.

Being scalable

- Goal: To ensure that the system could be scaled, and capacity be changed in size or scale.
- Scalability dimensions
  - Size scalability
    - easily add more users and resources to the system without any noticeable loss of performance
  - Geographical scalability
    - users and resources may lie far apart, but the fact that communication delays may be significant is hardly noticed
  - Administrative scalability
    - be easily managed even if it spans many independent administrative organizations

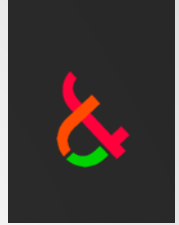
Being scalable

- Scaling up: improving their capacity (e.g., by increasing memory, upgrading CPUs, or replacing network modules)
- Scaling out: expanding the distributed system by essentially deploying more machines
  - hiding communication latencies
  - distribution of work
  - replication

Pitfalls

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

Pitfalls: The network is reliable



- Power failures
- Switches have a mean time between failures
- Other hardware and software failures

Pitfalls: The network is secure

- A hope?
- Need to build security into your applications
- Authentication and authorization are the key

Pitfalls: The network is homogeneous

- A homogeneous network today is an exception
  - a home network may connect a Linux PC and a Windows PC
- Need for interoperability.

Pitfalls: The topology does not change

- The topology doesn’t change only in the lab environments
- Servers may be added and removed often, clients (laptops, wireless ad hoc networks) are transient

Pitfalls: Latency is zero

- Latency (not bandwidth): how much time it takes for data to move from one place to another: measured in time
- Latency will be there, not observed often due to small values

Pitfalls: Bandwidth is infinite

- Internet: A packet switched network
  - Bandwidth on demand
- Compression

Pitfalls: Transport cost is zero

- Transferring a packet from source to destination and involving different networks/resources involves cost
- The cost (in terms of money) for setting and running the network is not zero

Pitfalls: There is one administrator

- Unless there is a small LAN, there will be different administrators associated with the network with different degrees of expertise
- Human factors might also come into play

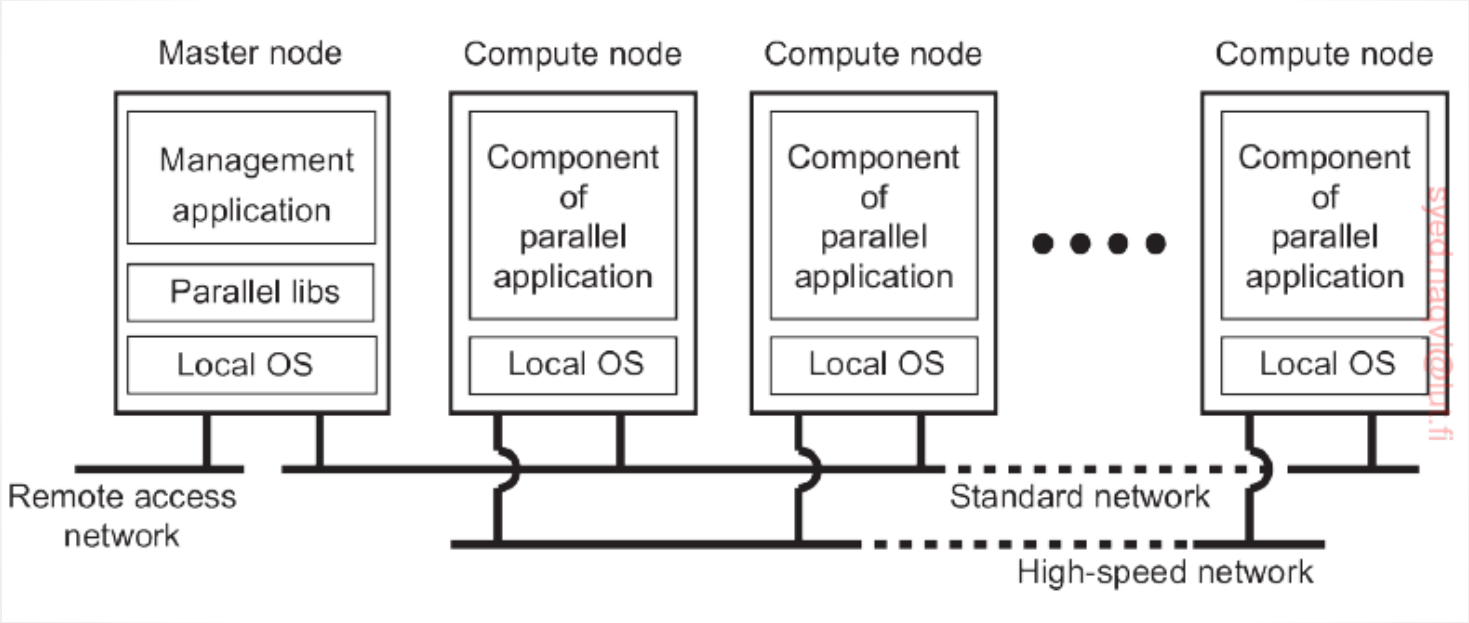
Types of distributed systems

- Distributed computing systems
  - cluster, grid, cloud
- Distributed information systems
- Pervasive systems

Cluster Computing

- A computer cluster is a set of loosely or tightly connected computers that work together so that, in many aspects, they can be viewed as a single system.
- Clusters have each node set to perform the same task, controlled and scheduled by software.

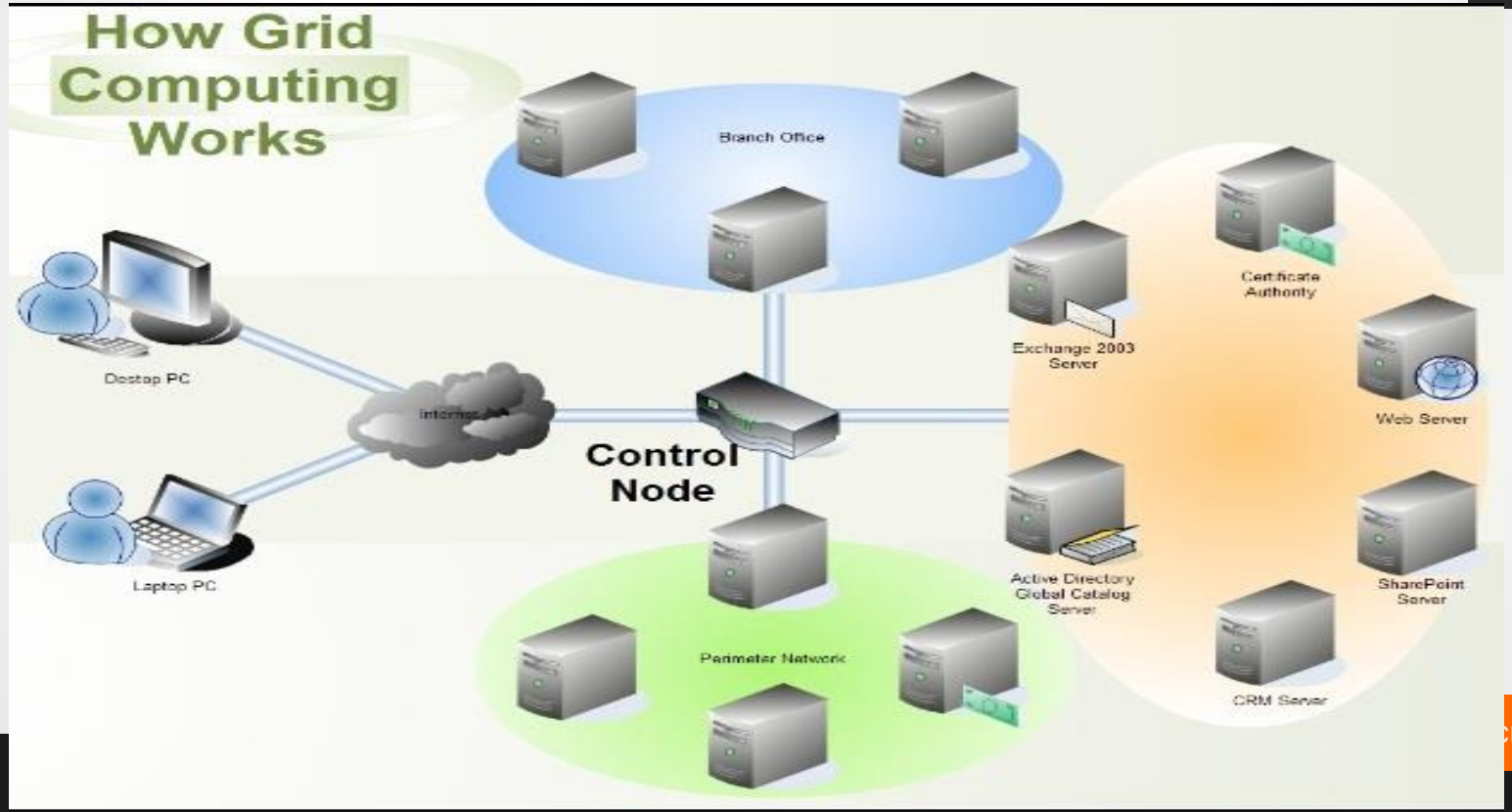
Cluster Computing



Grid computing

- A processor architecture that combines computer resources from various domains to reach a main objective.
- Computers on the network can work on a task together, thus functioning as a supercomputer

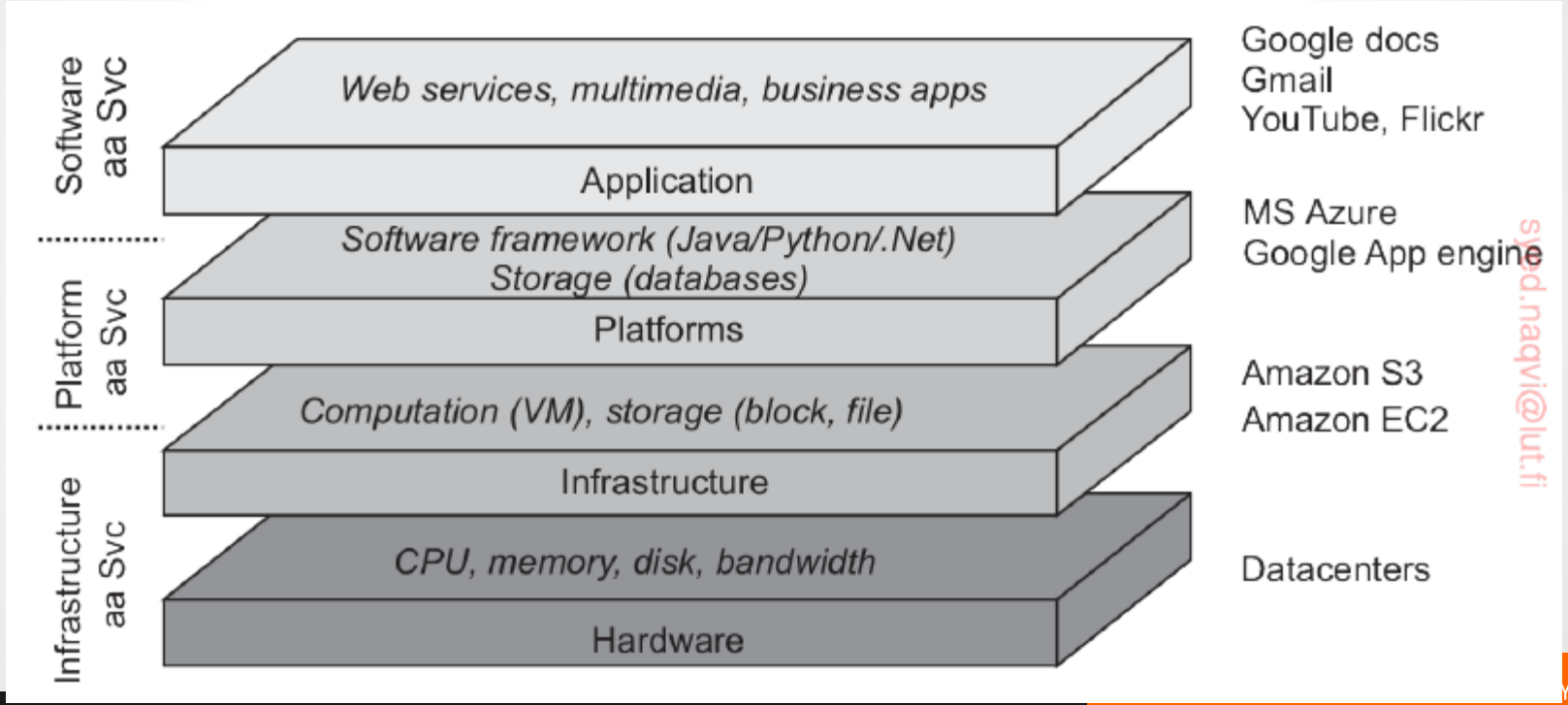
Grid Computing



Cloud computing

- Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user.

Cloud Computing



Distributed information systems

- Found in organizations which have many networked applications, and have interoperability concerns
- Middleware induced
  - remote procedure calls (RPC): an application component can effectively send a request to another application component by doing a local procedure call, which results in the request being packaged as a message and sent to the callee
  - remote method invocation (RMI): techniques were developed to allow calls to remote objects, same as an RPC, except that it operates on objects instead of functions

Pervasive systems

- naturally blend and distributed into our environment
  - Example: mobile devices
- Pervasive by nature but continuously present: ***ubiquitous systems***
  - Example: apple watch, electronic toll systems, speed checking cameras