02 - Distributed Systems

- Definition
- (Dis)advantages
- Challenges
-

- Coulouris 1
- Coulouris 2
- Saltzer_84.pdf

Definition

- Distributed Systems
  - Distributed System is a collection of independent computers that appear to its users as a single coherent system
- Examples of Distributed Systems:
  - Computer cluster in a university
  - Air lines database and reservation system
  - Web
  - Cloud

Network vs Distributed

- Remote components
- Communication
- Addressing

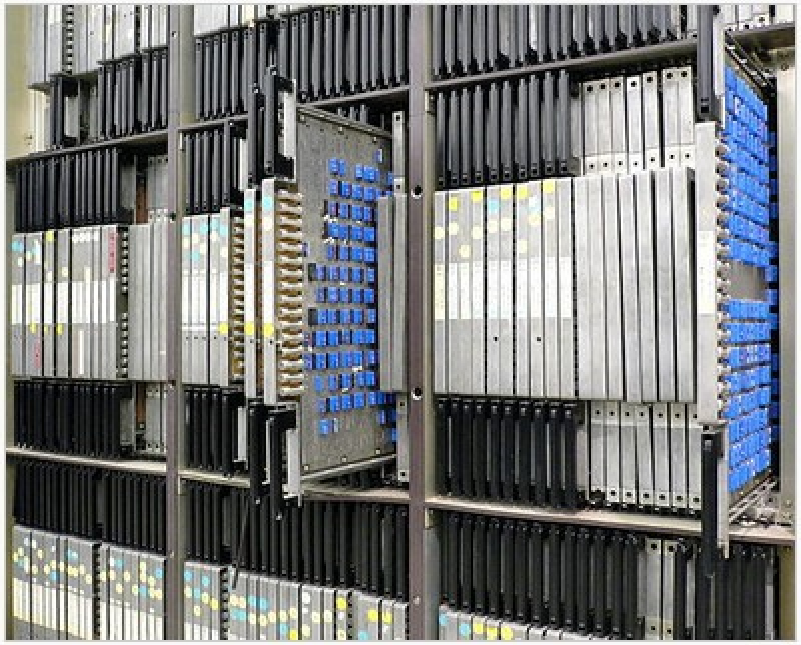
- Network system
  - Explicit
- Distributed system
  - Mostly implicit

- FileZilla
- Sshfs
- DNS ?
- WEB ?

Parallel vs Distributed Systems

- A concurrent system could be Parallel or Distributed:
- Two possible Views to make the distinction
- View 1:
  - Parallel System : A particular tightly-coupled form of distributed computing
  - Distributed System: A loosely-coupled form of parallel computing
- View 2:
  - Parallel System:processors access a shared memory to exchange information
  - Distributed System: uses a “distributed memory”. Massage passing is used to exchange information between the processors as each one has its own private memory.

Parallel vs Distributed Systems



ILLIAC IV parallel computer's CU

Advantages

Over Centralized Systems

- Economics:
  - Lower price/performance ratio
- Speed:
  - May have a more total computing power than a centralized system
  - Enhanced performance through load distributing.
- Inherent Distribution:
  - Some applications are inherently distributed
- Availability and Reliability:
  - No single point of failure.
  - The system survives even if a small number of machines crash
- Incremental Growth:
  - Can add computing power on to your existing infrastructure

Advantages vs networked PCs

- Computation:
  - can be shared over multiple machines
- Shared management of system:
  - backups & maintenance...
- Data Sharing:
  - many users can access the same common database
- Resources Sharing:
  - can share expensive peripherals
- Flexibility:
  - Spreading workload over the system CPUs.

Disadvantages

- Software:
  - Developing a distributed system software is hard
  - Creating OSs / languages that support distributed systems concerns
- Network:
  - When network is overloaded/messages lost, rerouting/rewiring the network is costly/difficult
- Security :
  - more sharing leads to less security especially in the issues of confidentiality & integrity
- Incremental growth is hard in practice
  - due to changing of hardware and software

Distributed Systems

- Challenges
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Failure handling
  - Concurrency
  - Transparency
  - Quality of service

Heterogeneity

- Applies to the following elements:
  - Networks
  - Hardware
  - Operating Systems
  - Programming languages
  - Multiple implementations by different developers

Openness

- Capability of a system to be:
  - Extended
  - Implemented in various ways
- Determined by degree of
  - How new services can be added
  - How can be accessed by multiple clients
- Open systems
  - Have interfaces published
  - Are based on uniform communication mechanisms
  - Can be built from heterogenous components
    - But components must be conform published standards

Security

- Security for resources has three components:
  - Confidentiality
    - Protection agains disclosure to unauthorized individuals
  - Integrity
    - Protection agains alteration and corruption
  - Availability
    - Protection agains interference with the means to access the resources

Scalability

- A system is scalable if
  - Remain effective when there is an increase number of users
- Challenges
  - Control the cost of physical resources
  - Control performance lost
  - Prevent SW resources starvation
  - Avoid performance bottlenecks

Failure handling

- Fails produce incorrect results or stop services
- Failures handling techniques:
  - Detecting failures
  - Masking failures
  - Tolerating failures
  - Recovering from failures
  - Redundancy

Concurrency

- Resources can be accessed simultaneously
  - By multiple clients
- Serialization of requests limits throughput
- Concurrent processing should be allowed
  - Shared resources should operate correctly in concurrent environment
    - Server
    - Services
    - Objects
  - Operations should be guarded

Transparency

- Concealment
  - Of separation/distributions of components
  - From the user and programmer
- System is perceived as a whole
  - Rather than a collection of components

Transparency

- Access transparency
  - Local and remote resources are accessed using the same operations
- Location transparency
  - Resources can be accessed without knowledge of physical and network location
- Concurrency transparency
  - Processes can operate concurrently using shared resources without interference between them

Transparency

- Replication transparency
  - Multiple instances of a resource can be used without knowledge of the replicas by the users or application programmers
- Failure transparency
  - Faults should be concealed
  - Users and programs should complete their tasks despite failures of HW or components
- Mobility transparency
  - Resources and clients can move within the system without affecting the operation of users and programs

Transparency

- Performance transparency
  - Systems can be reconfigured to improve performance as loads vary
- Scaling transparency
  - System and application can scale without change to the system structure and algorithms
- Network transparency
  - Access
    - Local vs remote
  - Location
    - Location independent addresses

Quality of service

- Users are provided with a functionality with a certain quality level
- Quality of service is affected by non-functional properties:
  - Reliability
  - Security
  - Performance
- Adaptability to changing configuration and resources
  - Important aspect to Quality of service
- Performance
  - Usually defined in terms of responsiveness and throughput
- QoS
  - Capability of a system to to meet pre-defined deadlines
  - Reliability, security or performance