Distributed System Principles

What is a Distributed System?

- ***Definition: 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.***

Centralised System Characteristics

- *One component with non-autonomous parts*
- *Component shared by users all the time*
- *All resources accessible*
- *Software runs in a single process*
- *Single Point of control*
- *Single Point of failure*

Distributed System Characteristics

- *Multiple autonomous components*
- *Components are not shared by all users*
- *Resources may not be accessible*
- *Software runs in concurrent processes on different processors*
- *Multiple Points of control*
- *Multiple Points of failure*

Common Characteristics

- *What are we trying to achieve when we construct a distributed system?*
- *Certain common characteristics can be used to assess distributed systems*
  - *Resource Sharing*
  - *Openness*
  - *Concurrency*
  - *Scalability*
  - *Fault Tolerance*
  - *Transparency*

Resource Sharing

- *Ability to use any hardware, software or data anywhere in the system.*
- *Resource manager controls access, provides naming scheme and controls concurrency.*
- *Resource sharing model (e.g. client/ server or object-based) describing how*
  - *resources are provided,*
  - *they are used and*
  - *provider and user interact with each other.*

Openness

- *Openness is concerned with extensions and improvements of distributed systems.*
- *Detailed interfaces of components need to be published.*
- *New components have to be integrated with existing components.*
- *Differences in data representation of interface types on different processors (of different vendors) have to be resolved.*

Concurrency

- *Components in distributed systems are executed in concurrent processes.*
- *Components access and update shared resources (e.g. variables, databases, device drivers).*
- *Integrity of the system may be violated if concurrent updates are not coordinated.*
  - *Lost updates*
  - *Inconsistent analysis*

Scalability

- *Adaption of distributed systems to*
  - *accomodate more users*
  - *respond faster (this is the hard one)*
- *Usually done by adding more and/or faster processors.*
- *Components should not need to be changed when scale of a system increases.*
- *Design components to be scalable!*

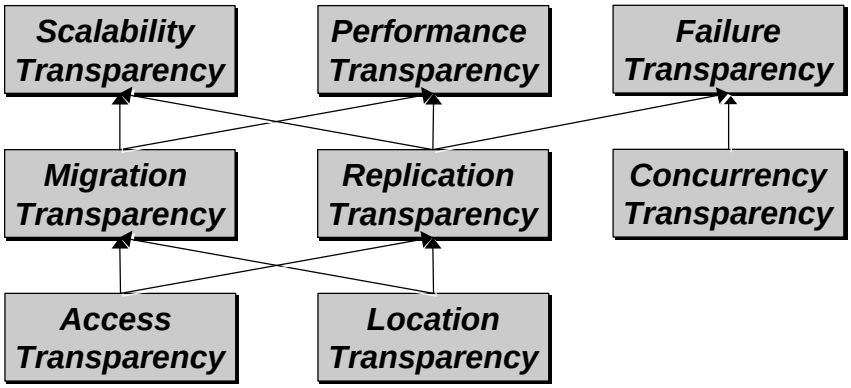
Fault Tolerance

- *Hardware, software and networks fail!*
- *Distributed systems must maintain availability even at low levels of hardware/software/network reliability.*
- *Fault tolerance is achieved by*
  - *recovery*
  - *redundancy*

Transparency

- *Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components.*
- *Transparency has different dimensions that were identified by ANSA.*
- *These represent various properties that distributed systems should have.*

Distribution Transparency



Access Transparency

- *Enables local and remote information objects to be accessed using identical operations.*
- *Example: File system operations in NFS.*
- *Example: Navigation in the Web.*
- *Example: SQL Queries*

Location Transparency

- *Enables information objects to be accessed without knowledge of their location.*
- *Example: File system operations in NFS*
- *Example: Pages in the Web*
- *Example: Tables in distributed databases*

Concurrency Transparency

- *Enables serveral processes to operate concurrently using shared information objects without interference between them.*
- *Example: NFS*
- *Example: Automatic teller machine network*
- *Example: Database management system*

Replication Transparency

- *Enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs*
- *Example: Distributed DBMS*
- *Example: Mirroring Web Pages.*

Failure Transparency

- *Enables the concealment of faults*
- *Allows users and applications to complete their tasks despite the failure of other components.*
- *Example: Database Management System*

Migration Transparency

- *Allows the movement of information objects within a system without affecting the operations of users or application programs*
- *Example: NFS*
- *Example: Web Pages*

Performance Transparency

- *Allows the system to be reconfigured to improve performance as loads vary.*
- *Example: Distributed make.*

Scaling Transparency

- *Allows the system and applications to expand in scale without change to the system structure or the application algortithms.*
- *Example: World-Wide-Web*
- *Example: Distributed Database*