Introduction to Distributed Systems (DS)

INF5040/9040 Autumn 2015

Lecturer: Amir Taherkordi (ifi/UiO)

August 24, 2015

UiO : University of Oslo

Outline

1. **Definition** of a distributed system
2. **Goals** of a distributed system
3. **Implications** of distributed systems
4. **Pitfalls** in developing distributed systems
5. **Types** of distributed systems

From a Single Computer to DS

- 1945-1985: Computers
  - Large and expensive
  - Operated independently
- 1985-now: two advances in technology
  - Powerful microprocessors
  - High-speed computer networks
- Result: **Distributed Systems**

> Putting together computing systems composed of a large number of computers connected by a high-speed network

3

What Is a Distributed System?

Operational perspective:

> A distributed system is one in which hardware or software **components,** located at **networked** computers, **communicate and coordinate** their actions only by **passing messages**.

[Coulouris]

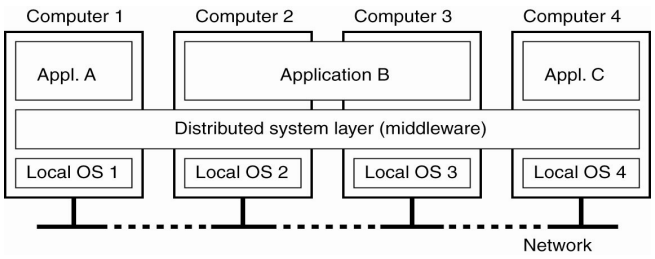
User perspective:

> A distributed system is a collection of independent computers that **appears** to its **users** as a **single coherent system**.

[Tanenbaum]

4

What Is a Distributed System?



- A DS organized as middleware
  - extending over multiple machines
  - offering each application the same interface

5

Examples of Distributed Systems

- Web search
  - Indexing the entire contents of the Web
- Massively multiplayer online games
  - Very large number of users sharing a virtual world.
- Financial trading
  - Real time access and processing of a wide rage of information sources.
  - Delivery of items of interest in a timely manner

6

Goals of Distributed Systems

- Resource sharing
- Distribution transparency
- Openness
- Scalability
- Fault tolerance
- Allow heterogeneity

Resource Sharing

- Making resources **accessible**:
  - accessing remote resources
  - sharing them in a controlled and efficient way
- Examples: printers, storages, files, etc.
- One reason to share: economics
  - e.g., 1-to-many printer access, rather than 1-to-1
- **Resource managers** control access, offer a scheme for naming, and control concurrency
- A **resource sharing model** describes how
  - resources are made available
  - resources can be used
  - service provider and user interact with each other

Models for Resource Sharing

- **Client-server** resource model
  - Server processes act as resource managers, and offer services (collection of procedures)
  - Client processes send requests to servers
  - (HTTP defines a client-server resource model)
- **Object-based** resource model
  - Any entity in a process is modeled as an object with a **message based interface** that provides access to its operations
  - Any shared resource is modeled as an object
  - Object based middleware (CORBA, Java RMI) defines object-based resource models

9

Distribution Transparency

- An important goal of a DS:
  - hiding the fact that its processes and resources are physically distributed across multiple computers
- Definition:

> A distributed system that is able to present itself to its users and applications as if it were only a single computer system is said to be **transparent.**

- What kind of transparency?

10

Forms of Transparency

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

- Degree of transparency
  - Situations in which full transparency is not good
  - Trade-off between a high degree of transparency and performance
  - In location and context-aware systems, e.g., ubiquitous DS?

11

Openness

- Definition:

  An **open** distributed system is a system that offers **services** according to **standard rules** that describe **syntax** and **semantics** of those services.

- E.g., in computer networks: format, content, and meaning of messages
- In DS: services specified through interfaces
  - Interface Definition Language (IDL): capturing syntax
  - Semantics? the hard part to specify
- Extensibility: an open DS can be extended and improved incrementally
  - add or replace components

12

Scalability

- A system is **scalable** if it remains effective when there is a significant increase in the amount of resources (data) and number of users
  - Internet: number of users and services has grown enormously
  - Google: scaled over the years to handle *O(100)* billion queries a month, expected query time 0.2 secs.
- **Scalability** denotes the ability of a system to handle an increasing future load
- **Dimensions**:
  - Scalable in size
  - Geographically scalable
  - Administratively scalable

13

Scalability Problems

- Problems with **size** scalability:
  - Often caused by centralized solutions

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

- Problems with **geographical** scalability:
  - traditional synchronous communication in LAN
  - unreliable communications in WAN
- Problems with **administrative** scalability:
  - across multiple domains: e.g., conflicting policies for resource usage

14

Scaling Techniques

- **Distribution**
  - splitting a resource (such as data) into smaller parts, and spreading the parts across the system (cf DNS)
- **Replication**
  - replicate resources (services, data) across the system
  - increases availability, helps to balance load
  - caching (special form of replication)
- **Hiding communication latencies**
  - avoid waiting for responses to remote service requests (use asynchronous communication or design to reduce the amount of remote requests)

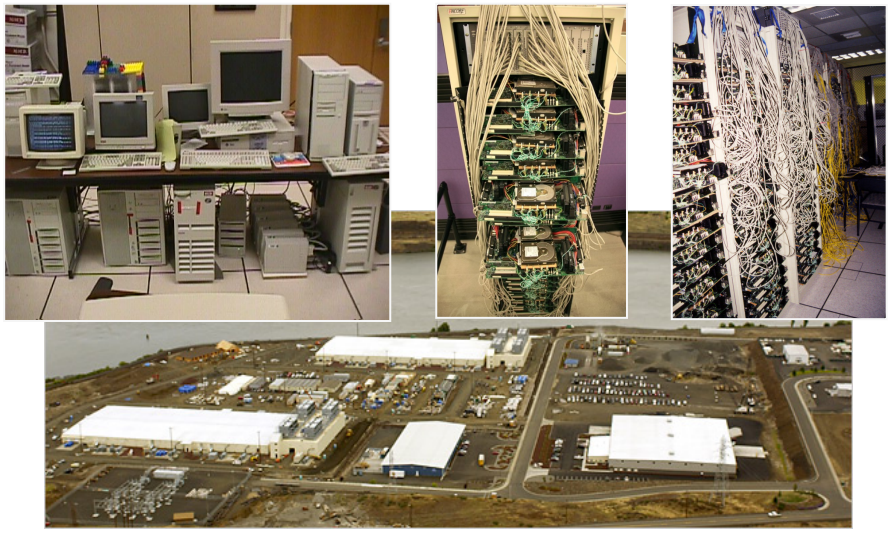
15

Fault Tolerance

- Hardware, software and network fail!!
- DS must maintain availability even in cases where hardware/software/network have low reliability
- Failures in distributed systems are partial
  - makes error handling particularly difficult
- Many techniques for handling failures
  - Detecting failures (checksum)
  - Masking failures (retransmission in protocols)
  - Tolerating failures (as in web-browsers)
  - Recovery from failures (roll back)
  - Redundancy (replicate servers in failure-independent ways)

16

Example: Google File System



Implications of Distributed Systems

- **Concurrency**
  - components execute in concurrent processes that read and update shared resources. Requires coordination
- **No global clock**
  - makes coordination difficult (ordering of events)
- **Independent failure of components**
  - "partial failure" & incomplete information
- **Unreliable communication**
  - Loss of connection and messages. Message bit errors
- **Unsecure communication**
  - Possibility of unauthorised recording and modification of messages
- **Expensive communication**
  - Communication between computers usually has less bandwidth, longer latency, and costs more, than between independent processes on the same computer

Pitfalls When Developing DS

- False assumptions made by first time developer:
  - 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**.

19

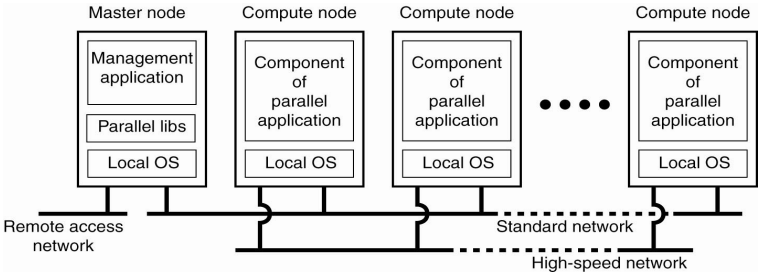
Types of Distributed Systems

- Distributed **Computing** Systems
  - Used for high performance computing tasks
  - Cluster and Cloud computing systems
  - Grid computing systems
- Distributed **Information** Systems
  - Systems mainly for management and integration of business functions
  - Transaction processing systems
  - Enterprise Application Integration
- Distributed **Pervasive** (or **Ubiquitous**) Systems
  - Mobile and embedded systems
  - Home systems
  - Sensor networks

20

Cluster Computing Systems

- Collection of similar PCs, closely connected, all run same OS, e.g.:

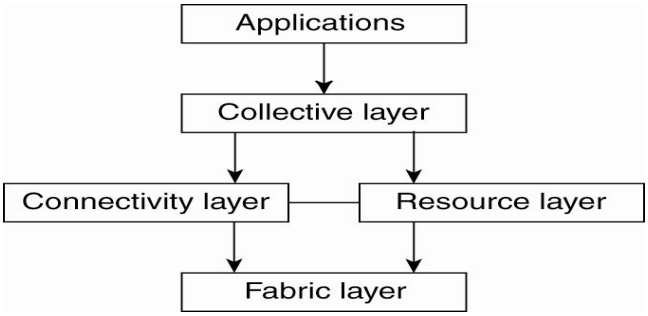


- A collection of **computing** nodes + **master** node
- Master runs **middleware**: parallel execution and management

21

Grid Computing Systems

- Federation of autonomous and heterogeneous computer systems (HW,OS,...), several adm domains

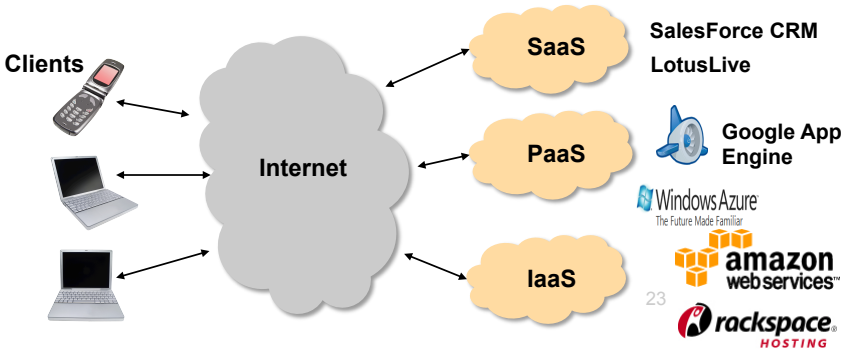


A layered architecture for grid computing systems.

22

Distributed Computing as a Utility

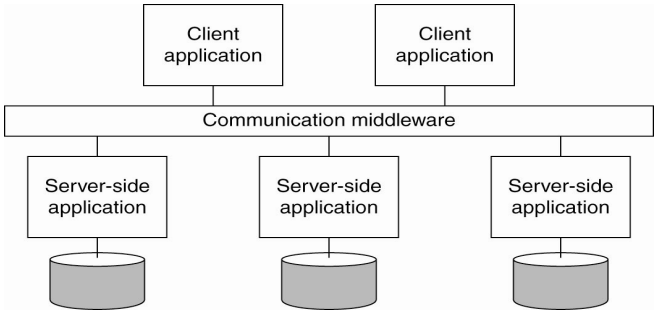
- View: Distributed resources as a commodity or utility
- Resources are provided by service suppliers and effectively rented rather than owned by the end user.
- The term **cloud computing** capture the vision of computing as a utility



23

Enterprise Application Integration

- Allowing existing applications to directly exchange information using communication middleware

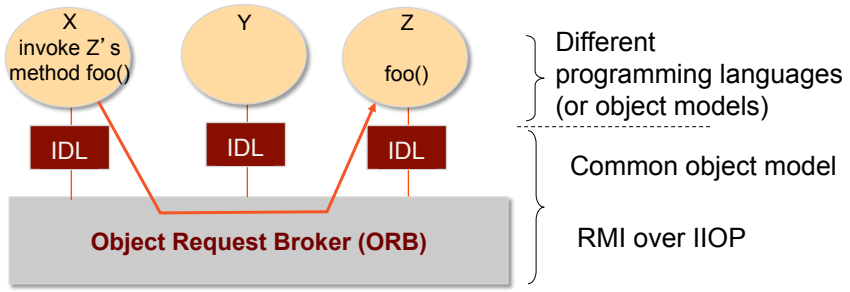


- Middleware as a communication facilitator in enterprise application integration

24

Example Communication Middleware: CORBA

- Clients may invoke methods of remote objects without worrying about:
  - object location, programming language, operating system platform, communication protocols or hardware.



25

Distributed Pervasive Systems

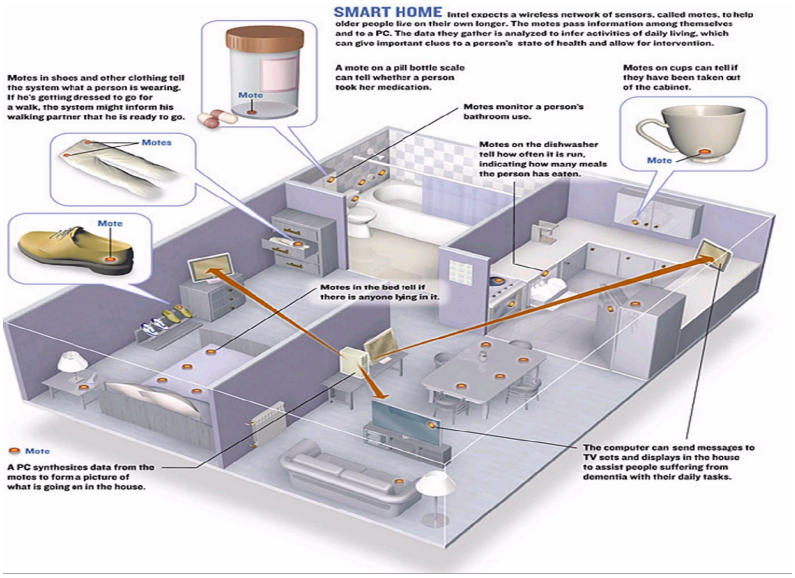
- Pervasive systems:

  exploiting the increasing integration of **services and (small/tiny) computing devices** in our everyday **physical** world

- (Mobile) Devices in distributed pervasive systems discover the environment (its services) and establish themselves in this environment as best as possible.
- Requirements for pervasive applications
  - Embrace contextual changes.
  - Encourage ad hoc and dynamic composition.
  - Recognize sharing as the default.

26

Example: Smart Home System



27

Summary

- Distributed systems:
  - components located in a network that communicates and coordinates their actions exclusively by sending messages.
- Goals like resource sharing, distribution transparency, openness, scalability, fault tolerance and heterogeneity can be satisfied by distributed systems
- Consequences of distributed systems
  - Independent failure of components
  - Unsecure communication
  - No global clock
- Many pitfalls when developing distributed systems
- Novel applications in pervasive systems: e.g., smart homes

28