

Sistemi Distribuiti - Lezioni

Sara Angeretti

@Sara1798

2022/2023

Indice

1	Contenuti della lezione	3
1.1	Parte1 - Concetti e modelli base	3
2	Introduzione ai Sistemi Distribuiti	4
2.1	Alcune definizioni	4
2.1.1	Definizione 1	4
2.1.2	Definizione 2	5
2.1.3	In sintesi	5
2.2	Sistemi come collezioni di nodi autonomi	5
2.2.1	Comportamenti	5
2.2.2	Collezioni di nodi	6
2.3	Sistemi coerenti	6
2.3.1	Essenzialmente	6
2.3.2	Trasparenza di come caratteristica fondamentale dei sistemi distribuiti	6
2.4	Sintesi delle caratteristiche di un sistema distribuito	6
3	Architetture Software	8
3.1	Definizione di architetture software	8
3.2	Modello base: Architetture a strati (layered)	8
3.2.1	Definizione	8
3.2.2	Sistemi Operativi Distribuiti	9
3.3	Architetture a livelli (tier)	11
3.4	Architetture basate sugli oggetti	11
3.5	Architetture centrate sui dati	11
3.6	Architetture basate su eventi	11
4	Il modello Client-Server	12

Capitolo 1

Contenuti della lezione

1.1 Parte1 - Concetti e modelli base

Cominciamo ad introdurre i seguenti argomenti:

- Definizione di sistema distribuito
- Architetture software
- Il modello client-server
- Proprietà e caratteristiche fondamentali

Capitolo 2

Introduzione ai Sistemi Distribuiti

2.1 Alcune definizioni

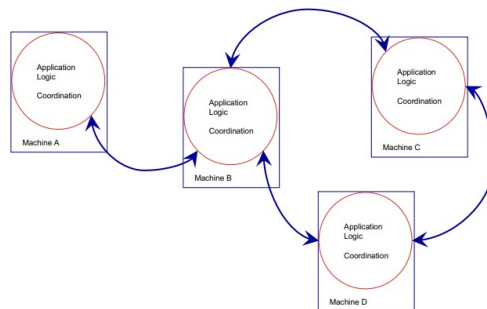
2.1.1 Definizione 1

Ci sono diverse definizioni con alcuni aspetti in particolare in comune.

Il libro di testo definire un *sistema distribuito* come un sistema in cui componenti hardware o software che sono localizzati in un sistema collegato alla rete, **comunicano** e **coordinano** le loro azioni solamente (**"only by"**) passandosi messaggi.

In inglese: "We define a distributed system as one in which **hardware or software components** located at **networked computers** **communicate** and **coordinate** their actions *only by passing messages*."

Ricorda che stiamo parlando di **processi**, anche se ci sono processi che operano senza una rete ma sono eccezioni.

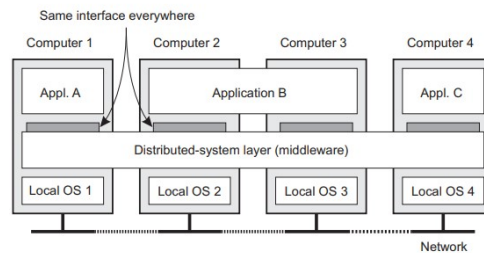


2.1.2 Definizione 2

Un'altra definizione è:

Un *sistema distribuito* è definito come un sistema di elementi computativi autonomi (che coesistono) che appaiono ad un utente come un'unica applicazione, un unico sistema coerente.

In inglese: "A distributed system is a collection of **autonomous computing elements** that appears to its users as a **single coherent system**."



2.1.3 In sintesi

Definition

- A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.

Characteristic features

- Autonomous computing elements, also referred to as nodes, be they hardware devices or software processes.
- Single coherent system: users or applications perceive a single system

⇒ nodes need to collaborate.

2.2 Sistemi come collezioni di nodi autonomi

2.2.1 Comportamenti

Sono **autonomi e indipendenti**, ovvero possono progredire come vogliono, ognuno ha la propria concezione del tempo ⇒ non c'è un clock globale, non è tutto sincronizzato.

Tutto ciò porta a *fondamentali problemi* di **sincronizzazione** e **coordinazione**.

2.2.2 Collezioni di nodi

Come gestire **appartenenze di gruppo** (group membership)?

I **gruppi** possono essere **aperti/dinamici** (qualsiasi nodo può partecipare) o **chiusi/fissi** (solo nodi ben selezionati possono entrare nel sistema, questa nozione verrà commentata diversamente e comunque non troppo a fondo).

Ma quindi, come faccio a sapere se il nodo con cui sto comunicando è effettivamente **autorizzato**?

2.3 Sistemi coerenti

2.3.1 Essenzialmente

La collezione di nodi opera nello stesso modo indipendentemente da dove, quando e come avvengono le interazioni fra l'utente e il sistema.

Examples

- An end user cannot tell where a computation is taking place
- Where data is exactly stored should be irrelevant to an application
- If or not data has been replicated is completely hidden

2.3.2 Trasparenza di come caratteristica fondamentale dei sistemi distribuiti

”**Trasparenza di distribuzione (?)**” come parola chiave. Da tradurre. The snag: partial failures

- It is inevitable that at any time only a part of the distributed system fails.
- Hiding partial failures and their recovery is often very difficult and in general impossible to hide.

2.4 Sintesi delle caratteristiche di un sistema distribuito

Caratteristiche fondamentali per tutti i sistemi distribuiti:

2.4. SINTESI DELLE CARATTERISTICHE DI UN SISTEMA DISTRIBUITO 7

Gestione della memoria?

- **Non c'è memoria condivisa**
- Comunicazione via scambio messaggi
- Non c'è stato globale: ogni componente (nodo, processo) conosce solo il proprio stato e può sondare lo stato degli altri.

Gestione dell'esecuzione?

- **Ogni componente è autonomo** ⇒ esecuzione concorrente
- Il coordinamento delle attività è importante per definire il comportamento di un sistema/applicazione costituita da più componenti

Gestione del tempo (temporizzazione)?

- **Non c'è un clock globale**
- Non c'è possibilità di controllo/scheduling globale
- Solo coordinamento via scambio messaggi

Tipi di fallimenti?

- **Fallimenti indipendenti** dei singoli nodi (independent failures)
- Non c'è fallimento globale

Capitolo 3

Architetture Software

3.1 Definizione di architetture software

Una architettura software definisce la struttura del sistema, le interfacce tra i componenti e i pattern di interazione (i protocolli).

I sistemi distribuiti possono essere organizzati secondo diversi stili architeturali.

3.2 Modello base: Architetture a strati (layered)

- Sistemi operativi
- Middleware

3.2.1 Definizione

Un'*architettura a strati* è un'architettura software che organizza il software in **strati**.

Ogni strato è

Each layer is built on top of another more general layer.

A layer can loosely be defined as a set of (sub)systems with the same degree of generality.

Upper layers are more application specific and lower are more general.

Quella al centro è da sapere benissimo in quanto oggetto di questo insegnamento.

3.2.2 Sistemi Operativi Distribuiti

Diversi tipi:

- DOS (Distributed Operating Systems)
- NOS (Network Operating Systems)
- Middleware

DOS (Distributed Operating Systems)

Users not aware of multiplicity of machines

- Access to remote resources like access to local resources

Data Migration

- Transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task

Computation Migration

- Transfer the computation, rather than the data, across the system

Process Migration - execute an entire process, or parts of it, at different sites

- Load balancing - distribute processes across network to even the workload
- Computation speedup - subprocesses can run concurrently on different sites
- Hardware preference - process execution may require specialized processor
- Software preference - required software may be available at only a particular site
- Data access - run process remotely, rather than transfer all data locally

NOS (Network Operating Systems)

Users are aware of multiplicity of machines

NOS provides explicit communication features

- Direct communication between processes (socket)
- Concurrent (i.e., independent) execution of processes that form a distributed application
- Services, such as process migration, are handled by applications

Access to resources of various machines is done explicitly by:

- Remote logging into the appropriate remote machine (telnet, ssh)
- Remote Desktop (Microsoft Windows)
- Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism

Middleware

Distributed Operating Systems

- Make services (e.g., data storage and process execution) **transparent** to applications
- Rely on homogeneous machines (since they need to run the same software)

Network Operating Systems

- Services (e.g., data storage and process execution) **are explicitly managed** by applications
- Do not require homogeneous machines (since they may run different software)
- E.g., MacOSX, Windows10, Linux

Middleware

- Implements services (one or more) to **make them transparent** to applications
- E.g., Java/RMI

Servizi Middleware

Services can address several issues, from general to domain specific.

Naming (il più importante) ovvero come faccio ad identificare un sistema operativo: astrazione

- Symbolic names are used to identify entities that are part of a DS
- They can be used by registries to provide the real addresses (e.g., DNS, RMI registries), or implicitly by the middleware

Access transparency (il più importante)

- ... defines and offers a communication model that hides details on message passing

Persistence

- ... defines and offers an automatic service for data storage (on file system or DB)

Distributed transactions (non vedremo tanto a fondo)

- ... defines and offers a persistence models to automatically ensure consistency on read/write operations (usually on DBs)

Security (non vedremo tanto a fondo)

- ... defines and offers models to protect access to data and services (with different levels of permissions) and computation integrity

3.3 Architetture a livelli (tier)

- Le applicazioni client server (2-tier, 3-tier)

3.4 Architetture basate sugli oggetti

- Java-Remote Method Invocation (RMI)

3.5 Architetture centrate sui dati

- Il Web come file system condiviso

3.6 Architetture basate su eventi

- Applicazioni Web dinamiche basate su callback (AJAX)

Capitolo 4

Il modello Client-Server

Il modello Client-Server è il modello di interazione tra un processo client e un processo server.