



## 8. Sistemi Distribuiti e Middleware

Andrea Polini

Ingegneria del Software  
Corso di Laurea in Informatica

# Sommario

- 1 Sistemi distribuiti - generalità
- 2 Middleware - generalità
- 3 Middleware - Tecnologie e Paradigmi
- 4 Modelli di calcolo distribuito

# Sommario

- 1 Sistemi distribuiti - generalità
- 2 Middleware - generalità
- 3 Middleware - Tecnologie e Paradigmi
- 4 Modelli di calcolo distribuito

# Sistemi distribuiti

## una definizione

Un sistema distribuito è una collezione di processori che **non condividono** memoria o clock. Ogni processore ha invece una sua memoria locale e la comunicazione tra processi avviene attraverso **linee di comunicazione**. I processori in un sistema distribuito variano in dimensioni e funzionalità.

...

Un sistema distribuito deve fornire **meccanismi per la sincronizzazione e la comunicazione**, per gestire il problema del **deadlock**, ed infine per gestire un insieme di **tipologie di fallimento** che non possono occorrere in un sistema centralizzato.

(Silberschatz - Galvin, *Operating System Concepts*)

# Sistemi distribuiti

in evidenza

- no memoria comune
- eterogeneità
- presenza dei problemi tipici della concorrenza e necessità di meccanismi di supporto alla loro gestione
- più punti di fallimento e nuove tipologie di fallimento
- problemi di sicurezza ed integrità

## CONCLUSIONE

La realizzazione di un sistema distribuito è tipicamente estremamente complessa e costosa rispetto alla realizzazione di un sistema centralizzato. Dunque se non è effettivamente necessario è in generale meglio non pianificare e progettare un sistema distribuito.

# Sistemi distribuiti

in evidenza

- no memoria comune
- eterogeneità
- presenza dei problemi tipici della concorrenza e necessità di meccanismi di supporto alla loro gestione
- più punti di fallimento e nuove tipologie di fallimento
- problemi di sicurezza ed integrità

## CONCLUSIONE

La realizzazione di un sistema distribuito è tipicamente estremamente complessa e costosa rispetto alla realizzazione di un sistema centralizzato. Dunque se non è effettivamente necessario è in generale meglio non pianificare e progettare un sistema distribuito.

# Sistemi distribuiti

in evidenza

- no memoria comune
- eterogeneità
- presenza dei problemi tipici della concorrenza e necessità di meccanismi di supporto alla loro gestione
- più punti di fallimento e nuove tipologie di fallimento
- problemi di sicurezza ed integrità

## CONCLUSIONE

La realizzazione di un sistema distribuito è tipicamente **estremamente complessa e costosa** rispetto alla realizzazione di un sistema centralizzato. Dunque se non è **effettivamente necessario** è in generale meglio non pianificare e progettare un sistema distribuito.

# Sistemi distribuiti

quando?

Alcune caratteristiche sono però difficilmente ottenibili da un sistema centralizzato. Quando dunque queste proprietà diventano particolarmente importanti si dovrà ricorrere alla progettazione di un sistema distribuito.

- scalabilità
- eterogeneità
- condivisione delle risorse
- tolleranza ai guasti
- apertura (openness)

Riguardo la performance? È questa una proprietà che dovrebbe suggerire di implementare un sistema distribuito?



# Sistemi distribuiti

quando?

Alcune caratteristiche sono però difficilmente ottenibili da un sistema centralizzato. Quando dunque queste proprietà diventano particolarmente importanti si dovrà ricorrere alla progettazione di un sistema distribuito.

- scalabilità
- eterogeneità
- condivisione delle risorse
- tolleranza ai guasti
- apertura (openness)

Riguardo la performance? È questa una proprietà che dovrebbe suggerire di implementare un sistema distribuito?

# Sommario

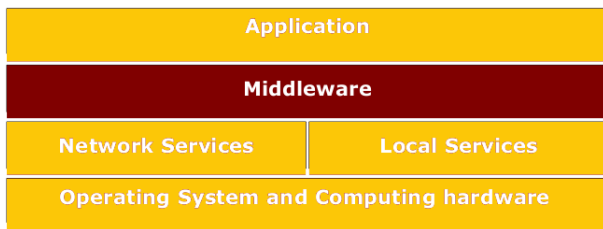
- 1 Sistemi distribuiti - generalità
- 2 Middleware - generalità**
- 3 Middleware - Tecnologie e Paradigmi
- 4 Modelli di calcolo distribuito

# Middleware

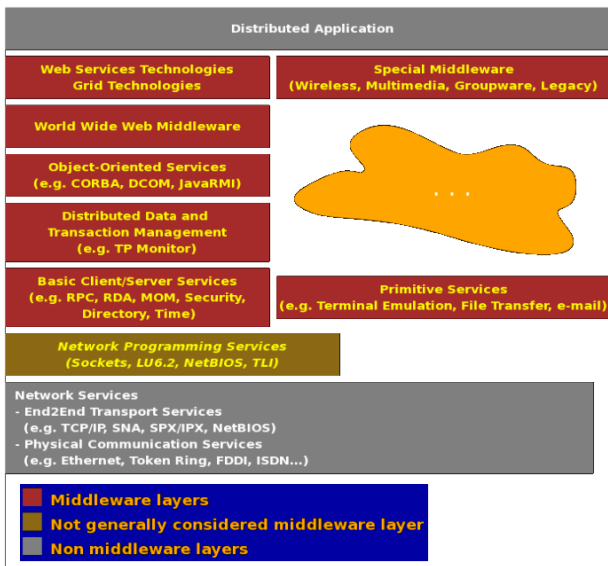
## definizione

Il **Middleware** è un insieme di servizi di supporto alla distribuzione indipendenti dalle applicazioni. In definitiva il middleware è il software che risiede al di sopra della rete ed al di sotto delle applicazioni software.

(Umar, *Object-Oriented Client/Server Internet environments*)



# Middleware - "vista d'insieme"

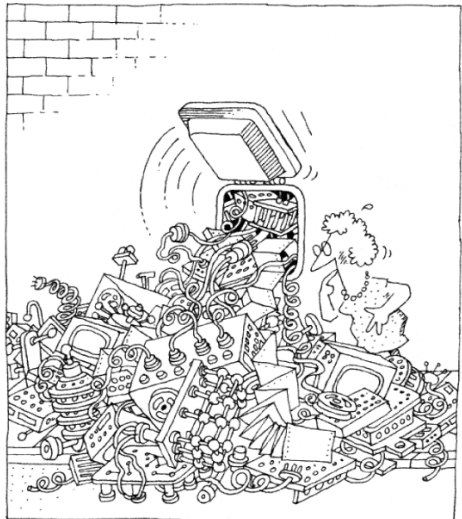
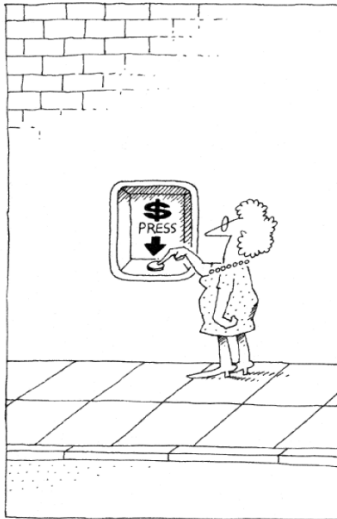


# Perché ne abbiamo bisogno

Sviluppo di ambiente distribuito eterogeneo complesso perché:

- scambio di dati complessi
- differenti tipi di codifica
- parametri di ritorno possono rappresentare altri componenti distribuiti
- attivazione e deattivazione di componenti distribuiti
- necessità di azioni atomiche
- sincronizzazione e parallelismo reale
- ...

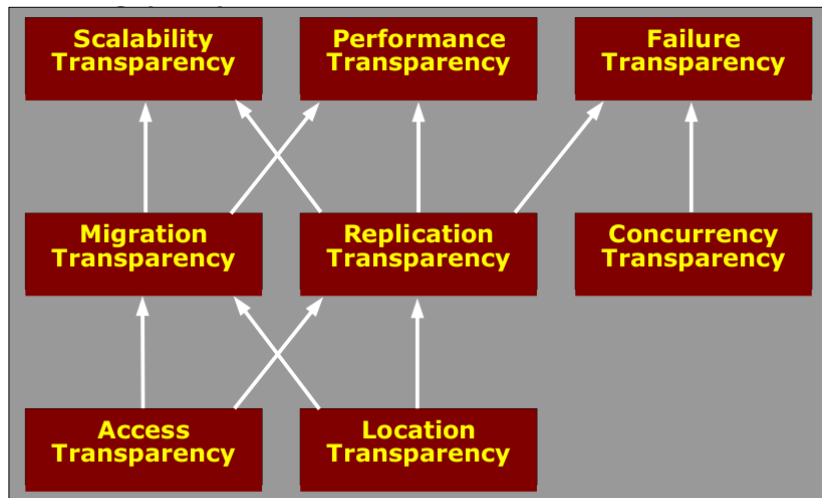
# Situazione tipica



# Middleware e trasparenza

Una tecnologia è **trasparente** se nasconde la complessità sottostante. Obiettivo del middleware è dunque far “**scompare**” la complessità introdotta dalla distribuzione.

# Dimensioni Trasparenza



(W.Emmerich – *Engineering Distributed Objects* – John Wiley and Sons)



# Sommario

- 1 Sistemi distribuiti - generalità
- 2 Middleware - generalità
- 3 Middleware - Tecnologie e Paradigmi**
- 4 Modelli di calcolo distribuito

# Tipi di Middleware

- **Transaction processing middleware** - semplifica la progettazione di applicazioni che richiedono esecuzione di transazioni distribuite - TP monitors (IBM CICS, BEA TUXEDO, Transarc Encina, . . .)
- **Message Oriented Middleware** - supporta lo scambio di messaggi tra applicazioni distribuite (IBM MQSeries, Sun ToolTalk . . .)
- **Remote Procedure Calls (RPC)** - permette di invocare facilmente una procedura residente su di una macchina remota.  
Richiede definizione di Interface Definition Language - IDL
- **Object-Oriented Middleware** - permette di implementare un'applicazione ad oggetti distribuiti. Progetto di applicazione come se fosse locale.

**Attenzione:** distribuzione non deve essere completamente trasparente al progettista ed allo sviluppatore

# Tipi di Middleware

- **Transaction processing middleware** - semplifica la progettazione di applicazioni che richiedono esecuzione di transazioni distribuite - TP monitors (IBM CICS, BEA TUXEDO, Transarc Encina, . . .)
- **Message Oriented Middleware** - supporta lo scambio di messaggi tra applicazioni distribuite (IBM MQSeries, Sun ToolTalk . . .)
- **Remote Procedure Calls (RPC)** - permette di invocare facilmente una procedura residente su di una macchina remota.  
Richiede definizione di Interface Definition Language - IDL
- **Object-Oriented Middleware** - permette di implementare un'applicazione ad oggetti distribuiti. Progetto di applicazione come se fosse locale.

**Attenzione:** distribuzione non deve essere completamente trasparente al progettista ed allo sviluppatore

# Object-Oriented middleware

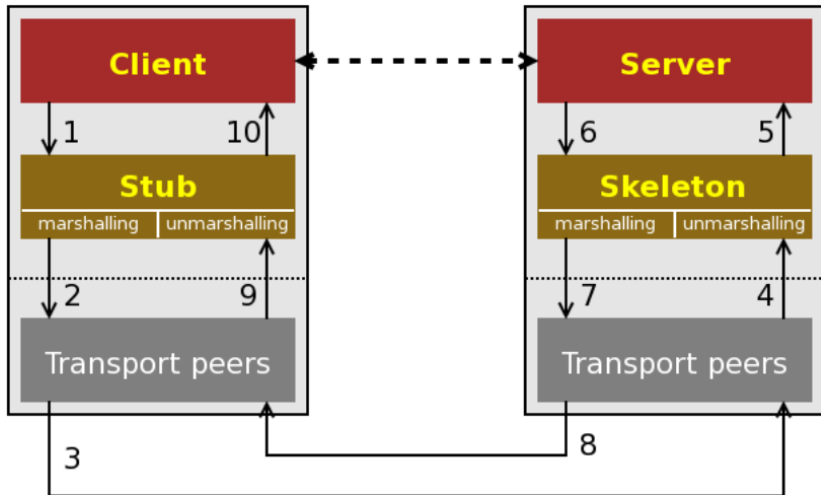
Principali elementi di un Middleware OO sono:

- Object Request Broker (ORB)
- Interface Definition Language (IDL)

Principali tipi di middleware OO sono:

- Java Remote Method Invocation (JavaRMI) - Sun
- Distributed COM (DCOM) / .Net - Microsoft
- Common Object Request Broker Architecture (CORBA) - OMG

# Meccanismi di invocazione remota



# Progettista dovrebbe sapere . . .

La progettazione di sistemi OO in distribuito presenta delle differenze che devono essere **considerate nelle varie fasi dello sviluppo**. In particolare queste riguardano:

- Ciclo di vita degli oggetti
- Riferimenti agli oggetti
- Tempi di risposta
- Attivazione e deattivazione
- Parallelismo
- Comunicazioni
- Fallimenti
- Sicurezza

# Ciclo di vita

- **Creazione:** tradizionalmente creazione avviene tramite invocazione del costruttore (il linker risolve il problema) residente nello stesso spazio di memoria). Questo non è più vero in distribuito. Meccanismo delle factory.
- **Migrazione:** è possibile che un oggetto migri dunque riferimenti ad un oggetto possono cambiare.
- **Rimozione:** implementazione di meccanismi di garbage collection in distribuito diventa particolarmente difficoltosa e molto costosa in termini di risorse. Potrebbe essere necessario gestire situazioni in cui un oggetto non esiste più.

# Riferimenti agli oggetti

- Riferimenti ad oggetti remoti richiedono **strutture dati complesse** (fattore anche 100).
- Dimensione influenza negativamente costo nel **passaggio dei parametri**
- Mantenere un **grosso numero di riferimenti remoti** potrebbe essere troppo costoso

Nella progettazione di applicazioni distribuite OO si dovrebbe cercare di **minimizzare il numero di oggetti**.



# Tempi di risposta

Richiesta ad oggetto remoto è tipicamente **più costosa di un fattore che oscilla tra 400 e 4000**.

Nella progettazione di applicazioni distribuite OO si dovrebbe cercare di **favorire comunicazioni tra oggetti "vicini"**. Oggetti che debbono comunicare molto dovrebbero esser posti nello stesso host.

# Attivazione/Deattivazione

Il ciclo di vita di un oggetto tipicamente include **fasi aggiuntive di attivazione/deattivazione**. Queste vanno ad incidere ancor più negativamente sui tempi di risposta.

Le motivazioni a queste due nuove fasi sono:

- le macchine ospitanti gli oggetti potrebbero aver subito **riavvii**
- le risorse disponibili su di un host potrebbero essere inferiori a quelle necessarie a tutti gli oggetti ospitati
- molti oggetti che forniscono servizi potrebbero rimanere **inattivi per lungo tempo** in attesa di nuove richieste

# Parallelismo

Certezza di parallelismo reale richiede di considerare con cautela la **sincronizzazione e l'accesso alle risorse**. Accesso a oggetti condivisi deve essere controllato

# Comunicazioni

Numerose cause che possono concorrere a ritardare le interazioni impongono l'introduzione di meccanismi non bloccanti

- Tipologie di comunicazione
  - sincrona
  - one-way
  - extended rendez-vous
  - asincrona
- Molteplicità
  - Unicast
  - Group
  - Multiple

# Fallimenti

Maggiore probabilità di fallimento, maggiori punti di fallimento

Nuove tipologie di fallimento - fallimenti parziali

Differenti livelli di affidabilità:

- Unicast
  - exactly-once
  - atomic
  - at-least-once
  - at-most-once
  - maybe
- Group e Multicast
  - k-reliability
  - totally ordered
  - best effort

# Sicurezza

Distribuzione di oggetti su reti non “controllabili”

Meccanismi di sicurezza. Ogni richiesta potrebbe essere autenticata

# Problemi comuni

Problemi ricorrenti nella progettazione di sistemi distribuiti ad oggetti non strettamente correlati alla specifica applicazione. Middleware forniscono soluzioni comuni:

- Location
- Gestione del ciclo di vita
- Persistenza
- Transazioni

# Common Object Request Broker Architecture (CORBA)

## short overview

- Standard aperto rilasciato dallo “Object Management Group” (OMG) - Ultima specifica rilasciata Marzo 2004 (1152 pagine)
- Elementi principali:
  - Object model
  - ORB
  - CORBA IDL
  - CORBA Services, CORBA Facilities, CORBA Domain Interfaces
- <http://www.corba.org.vc.html> (ca 220 organizzazioni)  
<http://www.omg.org/technology/corba/corbadownloads.htm>  
(ca 15 implementazioni free)



# Sommario

- 1 Sistemi distribuiti - generalità
- 2 Middleware - generalità
- 3 Middleware - Tecnologie e Paradigmi
- 4 Modelli di calcolo distribuito**

# Modelli di calcolo distribuito

Modelli differenti si basano su paradigmi di interazione differenti.  
Tipicamente si distingue tra:

- **File Transfer** - distribuzione riguarda i dati, basso carico e bassa concorrenza. e.g. e-mail
- **Client/Server**
- **Peer-2-Peer (P2P)** - più processi replicati su macchine differenti condividono risorse e scambiano informazioni
  - architetture decentralizzate
  - architetture semi-centralizzate

# Modello Cliente/Servente

Si distingue tra processi che richiedono servizi (Clienti) e quelli che offrono servizi (Serventi).

Nella computazione diverse problematiche devono essere affrontare:

- Interfaccia grafica
- Logica di business
- Gestione dei dati

La metodologia di gestione dei diversi “problemi” da luogo a diverse possibili architetture:

- 2-tier
  - Fat client
  - Thin client
- 3-tier
- n-tier

# Service Oriented Computing

Necessità di mettere in collegamento differenti organizzazioni (e.g. Business-to-Business - B2B).

Mettere in comunicazione differenti tecnologie “tradizionali” estremamente costosi.

Service oriented computing possibile risposta.