Università degli Studi di Roma “Tor Vergata”

Facoltà di Ingegneria

# Introduzione ai Sistemi Distribuiti

## Corso di Sistemi Distribuiti

Valeria Cardellini

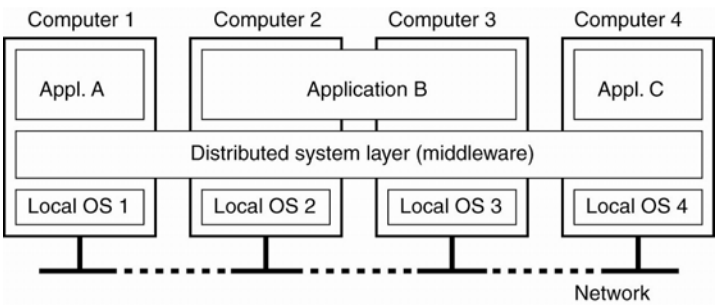
Anno accademico 2008/09

## Definizioni di SD

- Molteplici definizioni di **sistema distribuito (SD)**, scarsamente coerenti tra loro
- [Tanenbaum & van Steen] Un insieme di *calcolatori indipendenti* che appare agli utenti ed alle applicazioni come un *singolo sistema coerente*
  - Sistema costituito da componenti autonomi
  - Dall’esterno visione di un’unica entità

Necessaria la collaborazione tra componenti del sistema!

**Middleware**



# Definizioni di SD (2)

- [Coulouris & Dollimore] Un sistema distribuito è formato da componenti hw e sw localizzati su computer in rete che *comunicano* e *coordinano* le loro azioni attraverso *scambio di messaggi*
  - Se componenti = CPU è la definizione di macchina MIMD (Multiple Instruction stream Multiple Data stream)
- [Lamport] Si apprende l'esistenza di un SD quando il crash di un computer di cui non si è mai sentito parlare impedisce di portare a termine qualunque lavoro
  - Enfasi sulla gestione dei malfunzionamenti



# Obiettivi dei SD

- Accessibilità delle risorse
- Trasparenza
- Apertura
- Scalabilità

# Accessibilità delle risorse

- Risorsa = computer, stampanti, dati, file, pagine Web, reti, …
- Facilitare l'accesso alle risorse remote per utenti e applicazioni
- Condividere le risorse in modo efficiente e controllato
- Problemi di sicurezza (non esaminati in questo corso)



# Trasparenza

- Trasparenza della distribuzione: il SD è percepito nella sua interezza anziché come una collezione di componenti indipendenti
- Diversi tipi di trasparenza in un SD (ISO 10746, Reference Model of Open Distributed Processing)

Trasparenza all'accesso
  - Nasconde le differenza nella rappresentazione dei dati e nelle modalità di accesso alle risorse
    - L'accesso alle risorse locali e remote avviene usando le stesse identiche operazioni, in modo unico ed uniforme

Trasparenza all'ubicazione
  - Nasconde dove è localizzata una risorsa
    - Ad es. l'URL nasconde l'indirizzo IP
  - Trasparenza all'accesso ed all'ubicazione: trasparenza di rete

Trasparenza alla migrazione
  - Nasconde l'eventuale spostamento (logico o fisico) di una risorsa senza interferire sulla sua modalità di accesso

# Trasparenza (2)

## Trasparenza al riposizionamento

- Nasconde la possibilità di spostare una risorsa *mentre* è in uso

## Trasparenza alla replica

- Nasconde la replica di una risorsa
  - Ogni istanza della risorsa deve avere lo stesso nome
  - Supporto di trasparenza alla replica e trasparenza all'ubicazione

## Trasparenza alla concorrenza

- Nasconde la condivisione di una risorsa da parte di molti utenti contemporaneamente
  - Ad es. accesso contemporaneo di più utenti alla stessa tabella di una base di dati condivisa
  - L'accesso concorrente ad una risorsa condivisa deve lasciarla in uno stato consistente: ad es. meccanismi di *locking* (blocco)

## Trasparenza al guasto

- Nasconde il malfunzionamento e la riparazione di una risorsa
- Vedi definizione di SD di Lamport



# Grado di trasparenza

- Può essere eccessivo pretendere di ottenere una trasparenza *completa* della distribuzione
  - Gli utenti possono essere localizzati in continenti diversi; la distribuzione è apparente e non sempre si vuole nasconderla
  - E' impossibile (teoricamente e praticamente) nascondere completamente i guasti di nodi e reti
    - E' molto difficile distinguere tra una risorsa morta ed una estremamente lenta
    - Non si può essere certi che un server abbia eseguito un'operazione prima del crash
  - La trasparenza completa costa in termini di prestazioni
    - Tenere le copie cache esattamente aggiornate con la copia master
    - Effettuare il flushing immediato di un'operazione di scrittura per tolleranza ai guasti

# Apertura

- Un SD aperto offre servizi rispettando le regole standard che descrivono la sintassi e la semantica dei servizi stessi
  - Il sistema deve conformarsi ad interfacce standard
    - Spesso scritte in IDL (Interface Definition Language)
    - Spesso specificano solo la sintassi dei servizi
    - Definizione di interfaccia completa e neutrale
  - Il sistema deve supportare la portabilità delle applicazioni
    - Un'applicazione sviluppata per il SD A può essere eseguita, senza modifiche, su un SD B che implementa le stesse interfacce di A
  - Il sistema dovrebbe interoperare facilmente
    - Due implementazioni di sistemi o componenti di due diversi produttori possono coesistere e collaborare basandosi unicamente sui reciproci servizi specificati da uno standard comune



# Apertura (2)

- Ottenere l'apertura: rendere il SD indipendente almeno dall'eterogeneità dell'ambiente sottostante
  - Hardware
  - Piattaforme
  - Linguaggi

# Separazione di politiche e meccanismi

- L'implementazione dell'apertura richiede il supporto di differenti politiche specificate dalle applicazioni e dagli utenti
  - Quale livello di consistenza è richiesto per i dati nella cache del client?
  - Quali operazioni vengono concesse di essere eseguite al codice scaricato?
  - Quali requisiti di QoS (Quality of Service) vengono modificati a fronte di una variazione nella banda di rete?
  - Quale livello di sicurezza viene richiesto ad una comunicazione?
- Idealmente, un SD fornisce soltanto i meccanismi
  - Consente la configurazione (dinamica) di politiche di caching, preferibilmente per singola risorsa
  - Supporta diversi livelli di sicurezza per il codice mobile
  - Fornisce diversi parametri modificabili di QoS per ciascun flusso di dati
  - Offre diversi algoritmi di crittografia



# Scalabilità

- Indica la capacità di un sistema (distribuito) di migliorare le proprie prestazioni all'aumentare:
  - delle risorse che lo compongono e degli utenti
    scalabilità rispetto alla dimensione
  - della distanza tra le risorse e gli utenti
    scalabilità geografica
  - del numero di domini amministrativi coinvolti
    scalabilità amministrativa
- E' una proprietà difficile da ottenere
- La maggior parte dei SD si occupa soltanto e in modo parziale della scalabilità rispetto alla dimensione
  - Due direzioni per la scalabilità rispetto alla dimensione: verticale (scale-up) ed orizzontale (scale-out)
  - La classica non soluzione: macchine più potenti (scale-up)
- La sfida è nel progettare e realizzare sistemi con scalabilità geografica ed amministrativa

# Tecniche di scaling

## Nascondere le latenze nella comunicazione

Evitare l'attesa delle risposte alle richieste di servizi remoti (e distanti) e far fare altro lavoro utile al richiedente

- Soluzione: usare solo comunicazione asincrona
  - Si usa un handler (gestore) specifico per completare la richiesta
- Problema: comunicazione asincrona non adatta per tutte le tipologie di applicazioni (ad es. applicazioni interattive)

## Distribuzione

Suddividere dati e computazione tra molteplici macchine del sistema, ad es.:

- Servizi di naming decentralizzati (DNS)
- Servizi informativi distribuiti (Web)

## Replicazione e caching

Rendere disponibili copie dei dati su macchine diverse, ad es.:

- File server e database server replicati
- Mirroring di siti Web
- Web cache (nei browser e nei proxy)
- Caching dei file (lato server e lato client)



# Problema della scalabilità

- Non è difficile applicare le tecniche di scaling, tranne che:
  - La presenza di copie multiple (cached o replicate) causa problemi di **consistenza**
    - La modifica di una copia la rende diversa dalle altre
  - Per mantenere le copie sempre consistenti tra loro occorre una sincronizzazione globale ad ogni modifica
  - La sincronizzazione globale preclude soluzioni su larga scala
- Tuttavia, se si può tollerare un certo grado di inconsistenza, è possibile ridurre il bisogno di sincronizzazione globale
- Il grado di inconsistenza tollerabile dipende dal tipo di applicazione
  - Esempi: Web, scambi di borsa, aste on-line, …

# Tranelli nello sviluppo di SD

- Molti sistemi distribuiti sono inutilmente complessi a causa di errori di progettazione ed implementazione corretti con patch successive
- Si fanno molte false supposizioni:
  - la rete è affidabile
  - la rete è sicura
  - la rete è omogenea
  - la topologia non cambia
  - la latenza è nulla
  - l'ampiezza di banda è infinita
  - il costo di trasporto è nullo
  - c'è un solo amministratore



# Tipi di sistemi distribuiti

- Processori dual/multi-core (?)
- Sistemi di calcolo distribuiti
  - Molti SD sono configurati per il calcolo ad alte prestazioni
    - Cluster computing
    - Grid computing
- Sistemi informativi distribuiti
- Sistemi distribuiti pervasivi

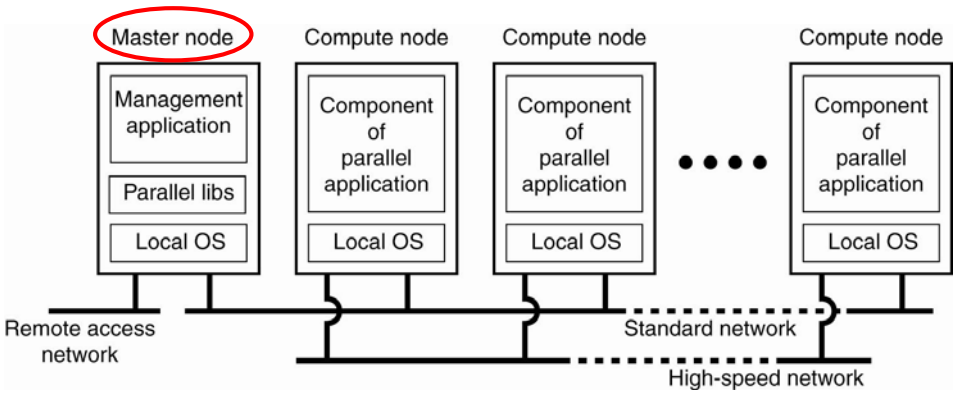
# Cluster computing

- Cluster costituito da un insieme di nodi ad alte prestazioni interconnessi tramite una rete locale ad alta velocità
  - Omogeneità: nodi con stesso sistema operativo, hardware molto simile, connessione attraverso la stessa rete
- Focus su alte prestazioni (HPC o High Performance Computing)
  - Un cluster può anche essere usato per alta affidabilità
- Organizzazione gerarchica con singolo nodo principale (ad es. Beowulf) oppure Single System Image (ad es. MOSIX)
  - Con Beowulf spesso usate librerie di message passing per il calcolo parallelo (ad es. MPI)
  - In Mosix bilanciamento automatico del carico e migrazione di processi



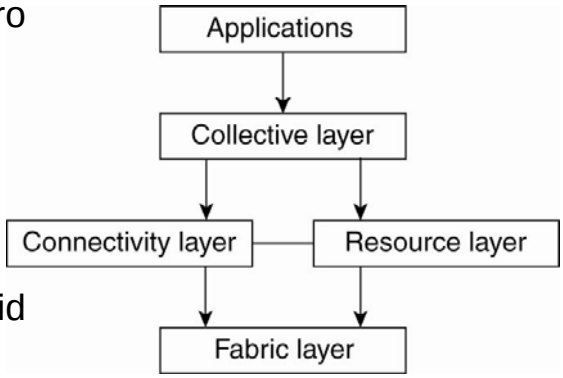
# Cluster computing (2)

- Esempio di organizzazione gerarchica: cluster Beowulf basato su Linux



# Grid computing

- Griglie di calcolo altamente decentralizzate, composte da un gran numero di nodi caratterizzati da un grado elevato di eterogeneità
  - Nodi diversi per hardware, software, tecnologia di rete, politiche di sicurezza, …
  - Nodi in diversi domini di amministrazione
- Collaborazione di gruppi di persone o istituzioni realizzata sotto forma di organizzazione virtuale
  - Un gruppo di utenti (o meglio i loro ID) con diritto d'accesso alle risorse fornite all'organizzazione virtuale
- Architettura a livelli per Grid [Foster, 2001]
  - Successiva evoluzione: Open Grid Service Architecture (OGSA)






# Sistemi informativi distribuiti

- Molti sistemi distribuiti in uso oggigiorno sono forme di sistemi informativi tradizionali, che integrano sistemi legacy
  - Ad es. sistemi transazionali

```
BEGIN_TRANSACTION(server, transaction);
READ(transaction, file1, data);
WRITE(transaction, file2, data);
newData := MODIFIED(data);
IF WRONG(newData) THEN
  ABORT_TRANSACTION(transaction);
ELSE
  WRITE(transaction, file2, newData);
  END_TRANSACTION(transaction);
END IF;
```

  - Tutte le operazioni READ e WRITE vengono eseguite; il loro effetto è permanente all'esecuzione di END_TRANSACTION
  - Le transazioni costituiscono un'operazione atomica (*principio del tutto o niente*)

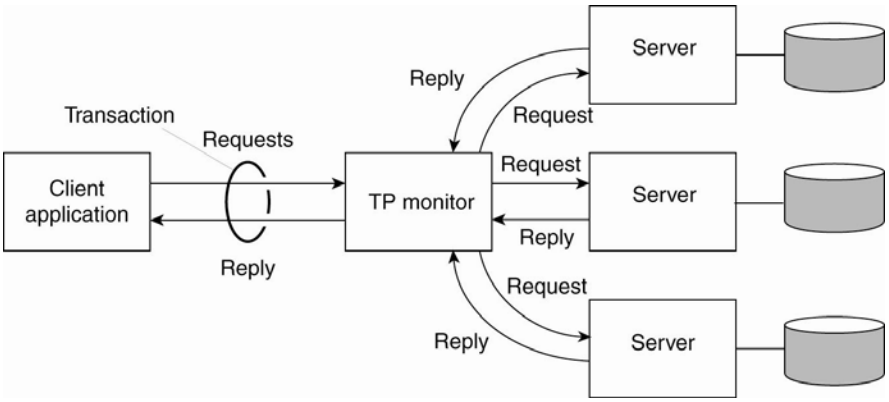
# Transazione

- Una transazione è un insieme di operazioni sullo stato di un oggetto che soddisfa le proprietà **ACID**
- Atomicità
  - La transazione o viene eseguita completamente (come un'azione singola, indivisibile ed istantanea) o non viene eseguita affatto
- Consistenza
  - La transazione non viola le invarianti del sistema; ciò non esclude la possibilità di stati intermediari non validi durante l'esecuzione della transazione
- Isolamento
  - Transazioni concorrenti non interferiscono le une con le altre
- Durabilità
  - Una volta che la transazione ha reso effettive le modifiche, esse sono permanenti



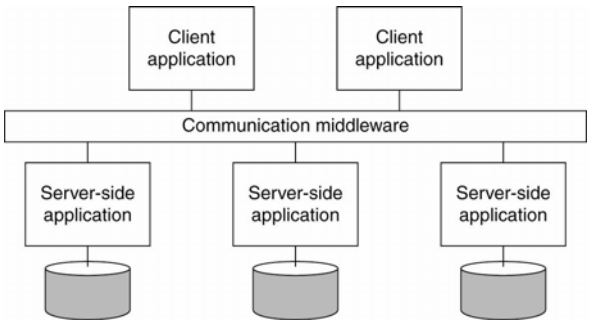
# TP monitor

- Transazione annidata (o distribuita): costruita a partire da un certo numero di sottotransazioni
  - I dati sono distribuiti su molteplici macchine
  - Il Transaction Processing (TP) monitor è responsabile di coordinare l'esecuzione della transazione annidata



# Integrazione di applicazioni aziendali

- Un TP monitor è adatto per applicazioni basate su database, ma in molti casi le applicazioni devono essere separate dalle basi di dati su cui sono state costruite
- Necessaria la comunicazione diretta tra le applicazioni (middleware di comunicazione)
  - Remote procedure call (RPC)
  - Remote method invocation (RMI)
  - Middleware orientato ai messaggi (MOM)



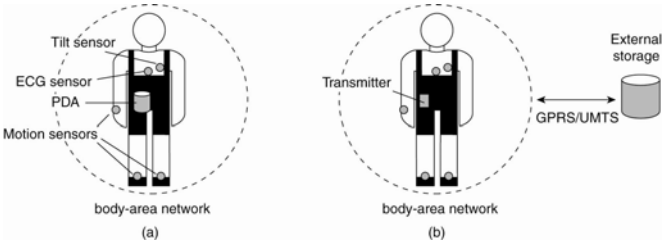



# Sistemi distribuiti pervasivi

- Una nuova generazione di sistemi distribuiti i cui nodi sono piccoli, mobili, con connessioni di rete wireless e spesso facenti parte di un sistema più grande
  - Sistemi domestici, sistemi elettronici per l'assistenza sanitaria, reti di sensori
- Alcuni requisiti per sistemi pervasivi:
  - Cambi di contesto: il sistema è parte di un ambiente che può cambiare in ogni momento
  - Composizione ad hoc: ogni nodo può essere usato in modi molto diversi da utenti differenti; richiesta la facilità di configurazione
  - Condivisione come default: i nodi vanno e vengono, fornendo informazioni e servizi da condividere
- Pervasività e trasparenza della distribuzione non sono facilmente coordinabili
  - E' preferibile esporre la distribuzione piuttosto che cercare di nasconderla

# Esempi di sistemi pervasivi

- Sistemi domestici
  - Obiettivi: sistemi completamente autoconfiguranti ed autogestiti
  - No amministratore di sistema
  - Gestione dello spazio privato di ciascun utente
  - Soluzione più semplice: un home box centralizzata?
- Sistemi elettronici per l'assistenza sanitaria
  - Dispositivi fisicamente vicini alla persona
  - Dome e come salvare i dati monitorati?
  - Come prevenire la perdita di dati cruciali?
  - Come generare e propagare gli allarmi?
  - Come garantire la sicurezza e la robustezza?






# Reti di sensori

- Caratteristiche: i nodi a cui i sensori sono collegati sono:
  - Molti (10-1000)
  - Semplici
  - Spesso alimentati a batteria (o persino senza)
- Le reti di sensori sono sistemi distribuiti
  - Consideriamoli come basi di dati distribuite
  - (a) Memorizzazione ed elaborazione dei dati in una base di dati centralizzata
  - (b) Memorizzazione ed elaborazione dei dati solo nei sensori