Sistemi distribuiti: introduzione

SCD

Anno accademico 2009/10
Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Definizione

- **Un sistema distribuito è un insieme di elaboratori indipendenti capaci di apparire all'applicazione come un sistema unitario e coerente**
  - **La comunicazione di tali elaboratori tra loro deve restare nascosta all'applicazione**
  - **L'interazione tra applicazione e sistema deve essere indipendente dal tempo e dallo spazio in cui essa avviene**

Caratteristiche di trasparenza

Trasparenza di	Per nascondere
Accesso	**Eventuali differenze** **- nella rappresentazione dei dati (per *hardware* eterogeneo)** **- nelle modalità di accesso alle risorse (per organizzazioni logiche diverse)**
Collocazione	**Il luogo di residenza effettiva delle risorse (distinzione tra nome fisico e nome logico)**
Migrazione	**Che una risorsa possa cambiare collocazione nel tempo**
Spostamento	**Che una risorsa possa cambiare collocazione durante l'uso**
Replicazione / Transazione	**L'esistenza di copie multiple di una risorsa / Coordinamento di attività per gestire una configurazione di risorse**
Malfunzionamento	**Il guasto e l'eventuale ripristino delle risorse**
Persistenza	**Il grado di persistenza della risorsa logica (residente in memoria primaria oppure in memoria secondaria)**

ISO/IEC 10746-[1,4]:1996, *Open Distributed Processing*

Altre caratteristiche desiderabili

- ***Openness***
  - **Portabilità e interoperabilità**
  - **Sintassi di invocazione definita da regole note e garantite**
    - Servizi sintatticamente specificati in termini di **interfacce**, espresse in linguaggio ad hoc (*Interface Definition Language*, IDL)
    - **Completezza**: la specifica dell'interfaccia non nasconde alcun dettaglio essenziale alla sua realizzazione
    - **Neutralità:** la specifica dell'interfaccia non impone una particolare realizzazione
- **Separazione tra politiche e meccanismi**
  - **La politica deve essere facilmente modificabile, adattabile e configurabile al variare dei bisogni e delle circostanze**
    - La politica è interna al servente e trasparente al cliente
  - **I meccanismi consentono la realizzazione di diverse politiche e non dovrebbero cambiare al variare di esse**

Altre caratteristiche desiderabili

- ***Scalability* (dimensionabilità)**
  - **Rispetto alla cardinalità**
    - Per aggiunta o rimozione di utenti, nodi, risorse
  - **Rispetto all'estensione geografica**
    - Utenti e risorse possono trovarsi a distanza variabile tra loro senza che questo ne pregiudichi l'accesso e l'interazione
  - **Rispetto alle problematiche locali di gestione**
    - Ciascuna amministrazione locale non pregiudica l'amministrazione del sistema distribuito nel suo complesso
- **Obiettivi a elevato costo prestazionale**

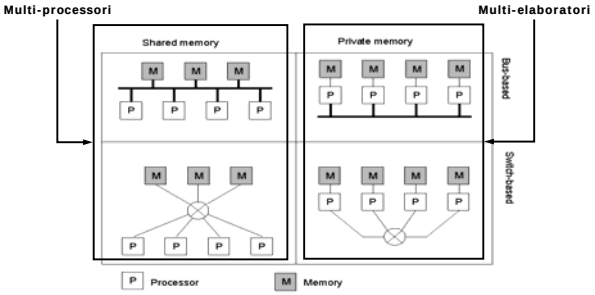
Fattori di centralizzazione

- **Centralizzazione dei servizi**
  - **Assumere un singolo servente per tutti gli utenti del sistema**
    - Pesante collo di bottiglia
- **Centralizzazione dei dati**
  - **Raccogliere tutte le informazioni significative in un unico luogo**
    - Dimensioni e complessità proibitive
- **Centralizzazione degli algoritmi**
  - **Necessitare visione completa dello stato corrente del sistema**
    - Onere di raccolta proibitivo

Prerequisiti di distribuzione

- **Un algoritmo è distribuito se**
  - **Non richiede informazione completa sull'intero sistema**
  - **Prende decisioni sulla base di conoscenza locale**
  - **Non viene pregiudicato da guasti locali**
  - **Non necessita di un tempo di sistema unico e globale**
  - **Consente ripartizione dei compiti e replicazione delle risorse e ne garantisce la consistenza necessario**
- **Il paradigma di comunicazione asincrona**
  - **"Nasconde" i ritardi di comunicazione sulla rete e dunque si presta alla distribuzione**

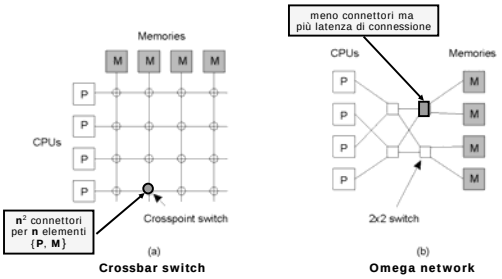
Distribuzione *hardware*



Sistemi multi-processore – 1

- **Spazio di memoria unico per tutte le CPU**
  - **La comunicazione su *bus* diventa rapidamente collo di bottiglia**
  - **La connessione punto a punto (*switch*) ripartisce le comunicazioni ma aumenta la complessità strutturale**
    - Connessione a matrice (***crossbar switch***): **N×M** punti di collegamento
      - Comunicazione veloci per grande costo strutturale
    - Combinazioni di sottoreti semplici di connessione (p.es. **2×2**, ***omega network***)
      - Basso costo strutturale per collegamenti più complicati
- **NUMA → *non-uniform memory access***
  - **Gerarchia di memoria più articolata (locale, globale)**
    - Costo di accesso ottimizzabile ma maggiore complessità organizzativa

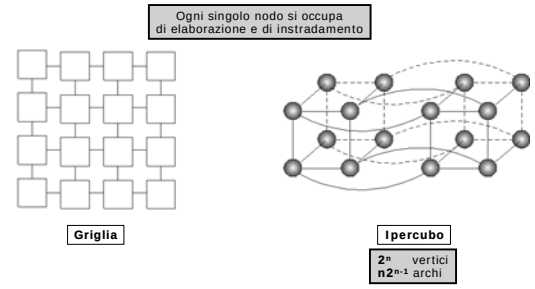
Sistemi multi-processore – 2



Sistemi multi-elaboratore – 1

- **Sistemi omogenei**
  - **Per ambiti di elaborazione specifici e specializzati**
  - **Nessuno spazio di memoria globale**
  - **Comunicazione tra elaboratori sia per diffusione (*bus*) che punto a punto (*switch*)**
    - Cammino determinato da instradatori (*router*)
    - Diversa scalabilità
  - **Topologie classiche punto a punto**
    - A griglia (*grid*)
    - A ipercubo (*hypercube*)
      - Cubi **n**-dimensionali con 2^n vertici e $n2^{n-1}$ archi diretti tra vertici
      - Ciascun vertice è un elaboratore e ciascun arco una connessione punto a punto

Sistemi multi-elaboratore – 2



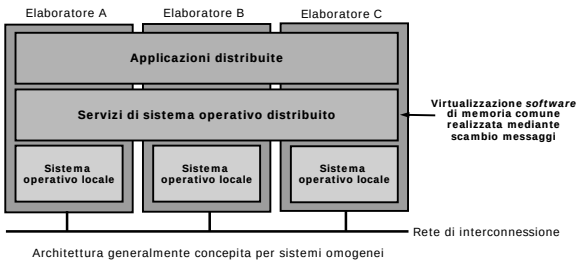
Sistemi multi-elaboratore – 3

- **Sistemi eterogenei**
  - **Per ambiti di elaborazione non necessariamente specializzati**
  - **Eterogenei sia rispetto alla tipologia degli elaboratori che alla topologia della rete di interconnessione**
  - **La base corrente dei sistemi distribuiti attuali**
  - **I sistemi omogenei multi-elaboratore sono più spesso visti come sistemi a parallelismo massiccio**

Distribuzione *software*

- **Visione secondo la struttura del sistema operativo**
  - **Ad accoppiamento stretto → sistema operativo distribuito**
    - Gestione uniforme delle risorse complessive del sistema
      - In totale analogia con le funzioni di un sistema operativo per mono-processore
    - Per sistemi multi-processore (a memoria condivisa) e per sistemi omogenei multi-elaboratore
  - **Ad accoppiamento lasco → sistema operativo di rete (NOS)**
    - Per offrire a utenti remoti l'accesso ad alcune risorse e servizi locali
    - Con funzionalità di gestione della distribuzione che possono essere arricchite da un livello *software* interposto tra NOS e applicazioni → ***middleware***

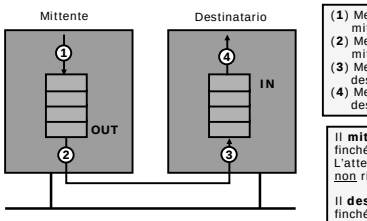
Sistemi operativi distribuiti – 1



Architettura generalmente concepita per sistemi omogenei

Sistemi operativi distribuiti – 2

Possibili punti di sincronizzazione nello scambio messaggi



(**1**) Messaggio depositato in *buffer* OUT mittente
(**2**) Messaggio prelevato da *buffer* OUT mittente e inviato su rete
(**3**) Messaggio depositato in *buffer* IN destinatario
(**4**) Messaggio prelevato da *buffer* IN destinatario per ricezione

Il **mittente** può bloccarsi su (**1**) finché il *buffer* OUT è pieno
L'attesa del mittente ai punti (**2–4**) non richiede *buffer* dal suo lato!

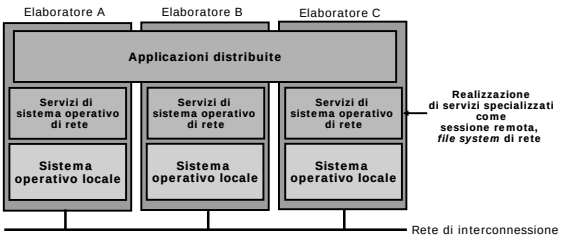
Il **destinatario** può bloccarsi su (**3**) finché il *buffer* IN è vuoto

L'attesa del mittente ai punti (**3–4**) ha senso solo in presenza di una rete di comunicazioni affidabile

Sistemi operativi distribuiti – 3

- **La programmazione di sistemi distribuiti per multi-elaboratore è molto più complicata di quella per sistemi multi-processore**
- **La comunicazione basata su memoria condivisa e primitive di sincronizzazione (semafori, *monitor*, risorse protette) è molto più facile di quella basata esclusivamente su scambio messaggi**
  - **Lo scambio messaggi è potenzialmente scalabile ma complicato dalle problematiche di accodamento, sincronizzazione e affidabilità della rete di interconnessione**

Sistemi operativi di rete

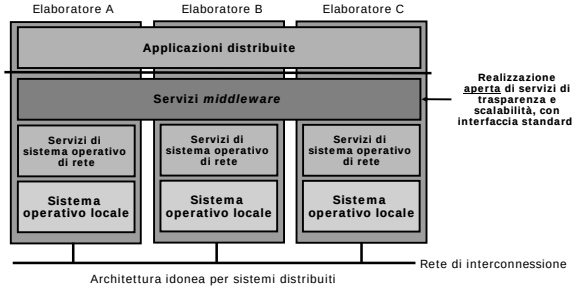


Architettura idonea per sistemi eterogenei

Sistemi distribuiti: *middleware* – 1

- **Né i sistemi operativi distribuiti né quelli di rete aderiscono alla definizione di sistema distribuito!**
  - **I sistemi operativi distribuiti hanno caratteristiche di trasparenza ma non coordinano un insieme di elaboratori indipendenti**
  - **I sistemi operativi di rete hanno caratteristiche di *openness* e *scalability* ma non forniscono la visione di un sistema unitario e coerente**
- **I sistemi distribuiti moderni sono realizzati per aggiunta di un livello *software* di astrazione chiamato *middleware* posto al livello NOS**

Sistemi distribuiti: *middleware* – 2



Architettura idonea per sistemi distribuiti

Sistemi distribuiti: *middleware* – 3

- **Vari modelli (paradigmi) di *middleware* si sono susseguiti nell'ultimo decennio**
  - ***File system* distribuito (estensione della piattaforma UNIX)**
    - Trasparenza limitata a *file* di tipo tradizionale
  - **Chiamate di procedura remota (RPC)**
    - Trasparenza estesa alla comunicazione distribuita
  - **Oggetti distribuiti**
    - Interazioni tra oggetti rappresentati da interfacce (dettagli implementativi nascosti)
  - **Documenti distribuiti → WWW**
  - **Paradigma SOA (*service-oriented architecture*)**
- **Problematiche comuni a ogni modello**
  - **Supporto alle varie forme di trasparenza, denominazione delle entità (*naming*), gradi di sicurezza**

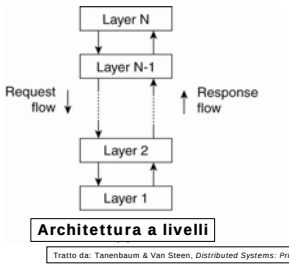
Sistemi distribuiti: *middleware* – 4

	Sistema operativo distribuito		Sistema operativo di rete	Sistema distribuito basato su *middleware*
	Multi-processore	Multi-elaboratore		
Grado di trasparenza	**Eccellente**	**Buono**	**Scarso**	**Buono**
Stesso sistema operativo su ogni nodo	**Si**	**Si**	**No**	**No**
Istanze di sistema operativo	**1**	**N**	**N**	**N**
Paradigma di comunicazione	**Memoria condivisa**	**Scambio messaggi**	**NFS**	**Svariati**
Gestione delle risorse	**Centralizzata per risorse globali**	**Distribuita per risorse globali**	**Per nodo**	**Per nodo**
Scalability	**Nulla**	**Modesta**	**Buona**	**Dipende dal paradigma**
Openness	**Nulla**	**Nulla**	**Buona**	**Buona**

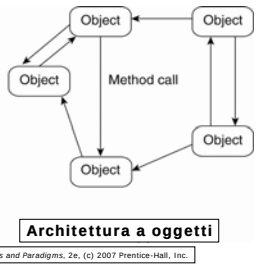
Stili architetturali – 1

- **Espressi in termini di definizione e uso di**
  - **Componenti**
    - Unità modulare dotata di interfacce **fornite** e **richieste** ben definite
      - Completamente rimpiazzabile nel suo ambiente
  - **Connettori**
    - Ciò che consente comunicazione, coordinamento e cooperazione tra componenti
- **Stili prevalenti**
  - **Architetture a livelli**
  - **Architetture basate su oggetti**
  - **Architetture orientate ai dati**
  - **Architetture basate su eventi**

Stili architetturali – 2



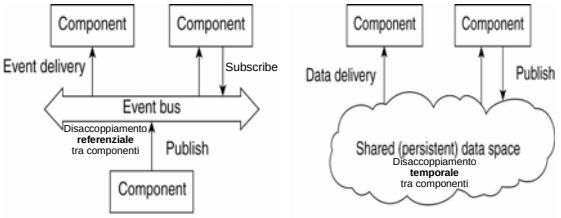
Architettura a livelli



Architettura a oggetti

Tratto da: Tanenbaum & Van Steen, *Distributed Systems: Principles and Paradigms*, 2e, (c) 2007 Prentice-Hall, Inc.

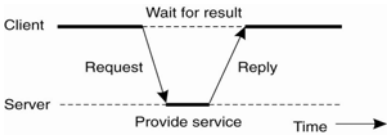
Stili architetturali – 3



Architettura basata su eventi | **Architettura orientata ai dati**

Tratto da: Tanenbaum & Van Steen, *Distributed Systems: Principles and Paradigms*, 2e, (c) 2007 Prentice-Hall, Inc.

Architetture centralizzate

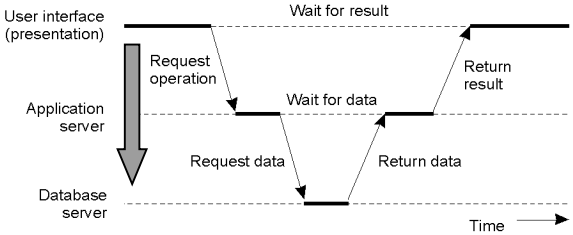


- **L'interazione tra cliente e servente implica un comportamento detto "request-reply"**
  - **Sorgente del problema prestazionale in Web 1.0**
  - **Alcune richieste (ma non tutte!) sono idempotenti**
    - Possono essere ripetute più volte senza causare danni o problemi
    - Proprietà molto importante a fronte di comunicazioni inaffidabili
    - Rendere logicamente affidabile una interconnessione fisicamente inaffidabile ha un costo molto elevato

Architetture distribuite – 1

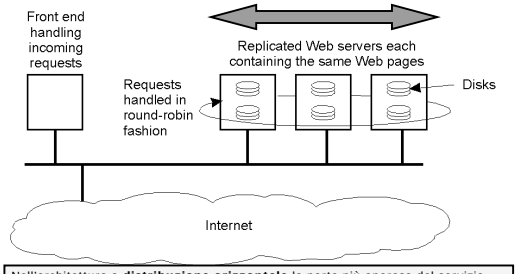
- **Varie architetture cliente-servente sono possibili**
  - **In relazione all'organizzazione del servizio e dei suoi dati**
- **Distribuzione verticale**
  - **Componenti diversi dello stesso servizio possono essere assegnati a elaboratori distinti**
    - Sia sul lato servente che sul lato cliente (delegazione parziale)
  - **Servizio reso tramite cooperazione di componenti distribuiti**
    - Ripartizione gerarchica (anche di autorità)
- **Distribuzione orizzontale**
  - **Servente e cliente possono essere partizionati ma ogni loro componente può operare da solo**
    - Bilanciamento del carico (con ripartizione del lavoro gestita da un *dispatcher*)
  - **Ogni componente sa fornire "il" servizio richiesto**

Architetture distribuite – 2



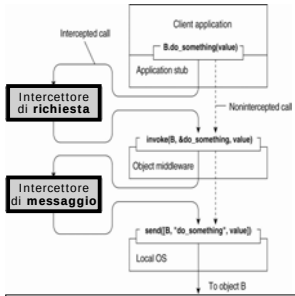
Nell'architettura a **distribuzione verticale** il servente visto dal cliente può essere esso stesso cliente di un componente servente cui sia stata demandata parte del servizio

Architetture distribuite – 4



Nell'architettura a **distribuzione orizzontale** la parte più onerosa del servizio può essere completamente replicata su più elaboratori distinti operanti in parallelo

Un nuovo concetto di *middleware*



- **Un approccio architetturale al *middleware* offre**
  - **Semplicità progettuale**
  - **Scarsa adattabilità**
- **In prospettiva è preferibile un approccio più adattabile basato su**
  - **"*Separation of concerns*"**
  - **"*Computational reflection*"**
  - **Progettazione per componenti e connettori**
    - L'intercettore illustrato in figura mostra il posizionamento logico dei connettori

Tratto da: Tanenbaum & Van Steen, *Distributed Systems: Principles and Paradigms*, 2e, (c) 2007 Prentice-Hall, Inc.