Sistemi distribuiti: introduzione

SCD

Anno accademico 2013/14
Sistemi Concorrenti e Distribuiti

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

Definizione

- **Un sistema distribuito è un insieme di nodi di calcolo indipendenti capaci di apparire all'applicazione come un sistema unitario e coerente**
  - **La comunicazione di coordinamento tra essi deve restare trasparente all'applicazione**
  - **L'interazione tra applicazione e sistema deve essere indipendente dal tempo e dalla locazione in cui avviene**

Caratteristiche di trasparenza

Trasparenza di	Per nascondere
Accesso	Differenze nella - rappresentazione dei dati (per HW eterogeneo) - modalità di accesso a 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	Esistenza di copie multiple di una risorsa Coordinamento di attività per gestire una configurazione di risorse
Malfunzionamento	Guasto ed eventuale ripristino delle risorse
Persistenza	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 – 1

- ***Openness***
  - **Portabilità e interoperabilità**
  - **Sintassi di invocazione definita da regole note e garantite**
    - Servizi sintatticamente specificati in termini di **interfacce** espresse in linguaggio neutro (***Interface Definition Language***, IDL)
    - **Completezza**: la specifica dell'interfaccia non nasconde alcun dettaglio essenziale alla sua realizzazione da parte di terzi
    - **Neutralità:** la specifica dell'interfaccia non impone una particolare realizzazione

Altre caratteristiche desiderabili – 2

- **Conviene separare 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 – 3

- ***Scalability*** **(dimensionabilità)**
  - **Rispetto alla cardinalità dei componenti del sistema**
    - Per aggiunta o rimozione di utenti, nodi, risorse
  - **Rispetto all'estensione spaziale**
    - 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**

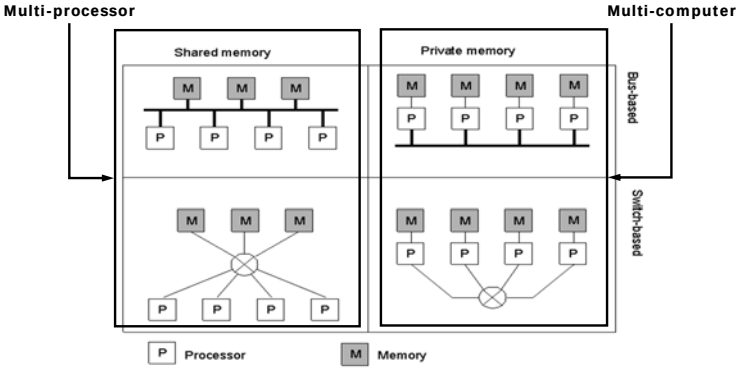
Fattori di centralizzazione

- **Centralizzazione dei servizi**
  - **Singolo servente per tutti gli utenti del sistema**
    - Collo di bottiglia
- **Centralizzazione dei dati**
  - **Tutte le informazioni significative in un unico luogo**
    - Dimensioni e complessità gestionale
- **Centralizzazione degli algoritmi**
  - **Conoscere lo stato corrente dell'intero sistema**
    - Onere di raccolta e ricostruzione

Prerequisiti di distribuzione

- **Un algoritmo è distribuito se**
  - **Non richiede informazione completa sull'intero sistema**
  - **Sa prendere 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 necessaria**
- **Il paradigma di comunicazione asincrona**
  - **Nasconde i ritardi di rete e quindi è più naturalmente adatto alla distribuzione**

Distribuzione HW



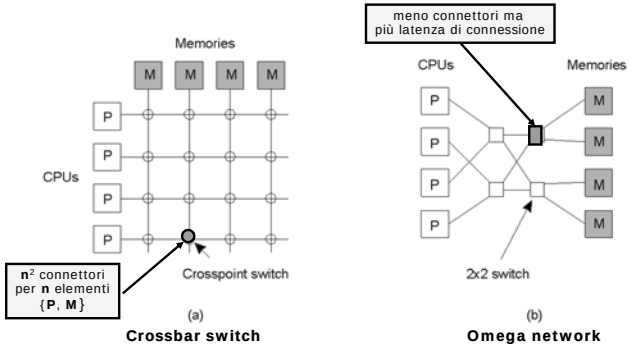
Architettura di memoria

- **Uniforme (UMA) ⇒ *multi-processor***
  - **Spazio di indirizzamento unico e comune**
    - Assunzione base delle architetture SMP (***Symmetric Multi-Processor***)
  - ***Cache* coerente**
  - **Accesso uniforme a tutta la memoria**
    - Ma ogni singolo accesso blocca tutte le CPU
- **Non uniforme (NUMA) ⇒ *multi-computer***
  - **Spazio di indirizzamento comune**
  - ***Cache* coerente**
  - **Accesso non uniforme alla memoria comune**
    - Costo di accesso ottimizzabile ma con maggiore complessità organizzativa

Sistemi *multi-processor* – 1

- **Unico spazio di indirizzamento ∀ CPU**
  - **La comunicazione su *bus* causa collo di bottiglia**
  - **La connessione punto a punto (*switched*) bilancia meglio il carico al costo di maggiore complessità strutturale**
  - **Connessione completa (*crossbar switch*) con matrice $P \times M$**
    - Comunicazione veloci per alto costo strutturale
  - **Combinazione di sottoreti connessione più semplici (p.es. 2×2, *omega network*)**
    - Basso costo strutturale per collegamenti più complicati

Sistemi *multi-processor* – 2



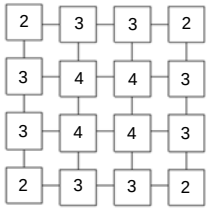
Sistemi *multi-computer* – 1

- **Omogenei**
  - **Nessuno spazio di indirizzamento comune**
  - **Comunicazione via *router* con interconnessione a diffusione (*bus*) o punto a punto (*switch*)**
    - L'interconnessione a *bus* non scala quella a *switch* si
- **Interconnessione punto a punto**
  - **Topologia 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

Sistemi *multi-computer* – 2

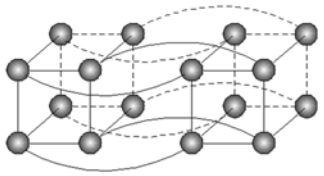
Ogni singolo nodo si occupa di elaborazione e di instradamento

2^n vertici
$n2^{n-1}$ archi



Griglia

Qui la posizione del nodo determina il suo numero di vicini quindi il *routing* va specializzato



Ipercubo

Qui il numero di vicini è invariante e il *routing* non va specializzato

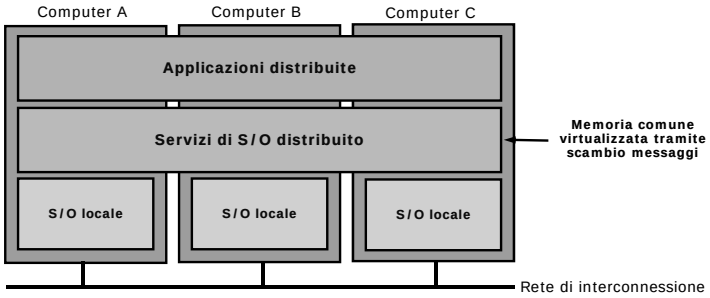
Sistemi *multi-computer* – 3

- **Eterogenei**
  - **Sia rispetto alla tipologia degli elaboratori che alla topologia di interconnessione**
  - **Sono il modello architetturale più generale**
    - E quindi il termine di riferimento dei sistemi distribuiti
- **Nota storica**
  - **I sistemi omogenei erano visti come architetture a parallelismo massiccio per applicazioni specializzate**
    - L'avvento dei processori *multi-core* ne ha cambiato la percezione
    - I nuovi processori *many-core* sono *multi-computer* eteronegei

Distribuzione SW

- **Secondo la struttura del S/O**
- **Accoppiamento stretto → S/O distribuito**
  - **Gestione uniforme delle risorse di sistema**
    - In analogia con le funzioni di S/O per *mono-processor*
- **Accoppiamento lasco → S/O di rete (NOS)**
  - **Per offrire a utenti remoti l'accesso ad alcune risorse e servizi locali**
  - **Le funzionalità di gestione della distribuzione possono essere arricchite da un livello SW interposto tra NOS e applicazioni → *middleware***

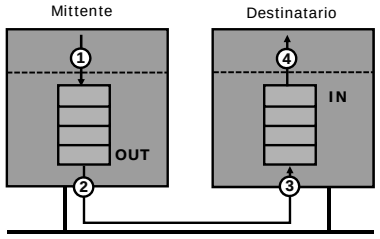
Sistemi operativi distribuiti – 1



Architettura generalmente concepita per sistemi omogenei

Sistemi operativi distribuiti – 2

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!

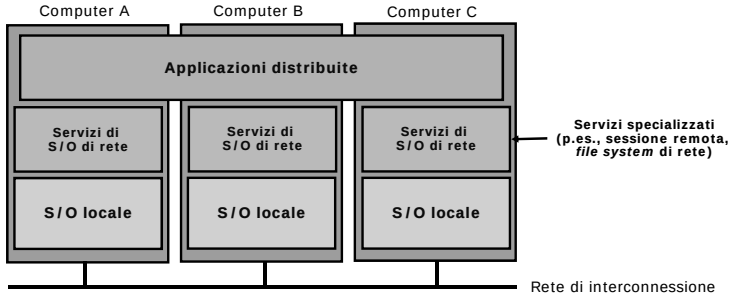
Il **destinatario** può bloccarsi su (**3**) finché il *buffer* IN è vuoto

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-computer* è molto più complessa di quella per sistemi *multi-processor***
  - **Vale invece il contrario per le problematiche di *scheduling* (!)**
- **La comunicazione basata su memoria condivisa e primitive di sincronizzazione è molto più facile di quella basata su scambio messaggi**
  - **Lo scambio messaggi è potenzialmente scalabile ma complicato dalle problematiche di accodamento, sincronizzazione (coordinamento) e affidabilità della rete di interconnessione**
  - **Per la sincronizzazione nella condivisione di risorse in presenza di parallelismo non è ovvio scegliere tra *suspend lock* e *spin lock***

Sistemi operativi di rete

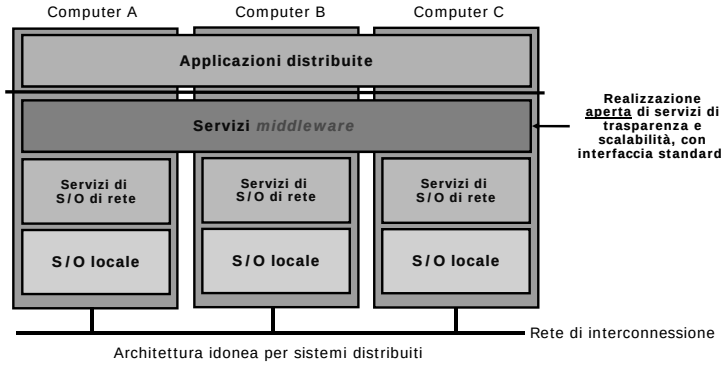


Architettura idonea per sistemi eterogenei

Sistemi distribuiti: *middleware* – 1

- Né i S/O distribuiti né i S/O di rete aderiscono alla definizione di sistema distribuito
  - S/O distribuiti hanno caratteristiche di trasparenza ma non coordinano un insieme di nodi indipendenti
  - S/O di rete hanno caratteristiche di *openness* e *scalability* ma non forniscono la visione di un sistema unitario e coerente
- I sistemi distribuiti moderni aggiungono a (o rimpiazzano) lo strato NOS con un livello di astrazione SW chiamato *middleware*

Sistemi distribuiti: *middleware* – 2



Architettura idonea per sistemi distribuiti

Sistemi distribuiti: *middleware* – 3

- Esistono svariati paradigmi di *middleware*
- *File system* distribuito → NFS su UNIX
  - Trasparenza limitata a *file* di tipo tradizionale
- Chiamate di procedura remota (RPC)
  - Trasparenza estesa alla comunicazione distribuita
- Oggetti distribuiti
  - Interazioni come tra oggetti rappresentati da interfacce semplici

Sistemi distribuiti: *middleware* – 4

- E poi ancora ...
- Documenti distribuiti → WWW
- Risorse distribuite → paradigma REST
- Servizi distribuiti → paradigma SOA
- Ma tutti i diversi paradigmi hanno alcune problematiche comuni
  - Trasparenza, *naming*, sicurezza

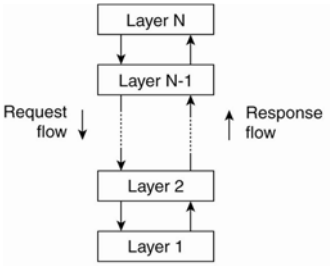
Sistemi distribuiti: *middleware* – 4

	S/O distribuito		S/O di rete	Sistema distribuito basato su *middleware*
	Multi-processor	Multi-computer		
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

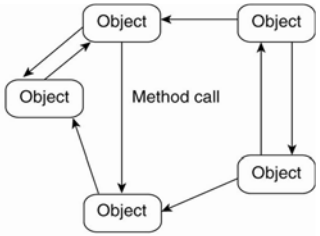
Stili architetturali – 1

- **Espressi in termini di definizione e uso di**
  - **Componenti**
    - Unità modulare coesa dotata di interfacce fornite e richieste ben definite
  - **Connettori**
    - Mezzo per comunicazione, coordinamento e cooperazione tra componenti
- **Alternative comuni**
  - **A livelli**
  - **A oggetti**
  - **Orientate ai dati**
  - **Basate su eventi**

Stili architetturali – 2



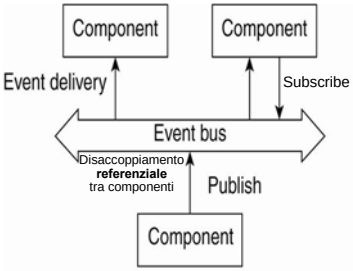
Architettura a livelli



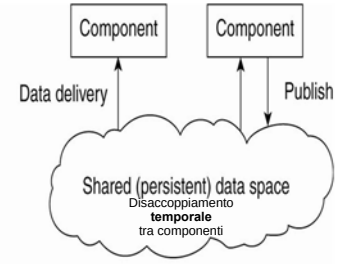
Architettura a oggetti

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

Stili architetturali – 3



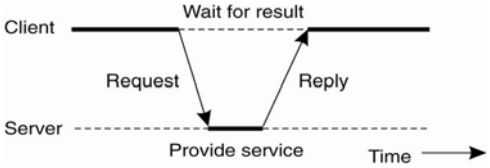
Architettura basata su eventi



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 "*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 affidabile una interconnessione fisica inaffidabile ha costo molto elevato

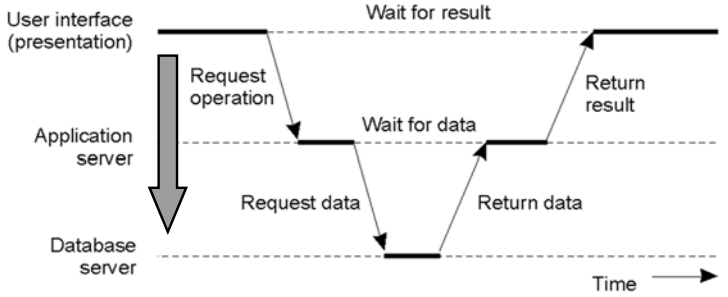
Architetture distribuite – 1

- **Due le varianti principali di architetture distribute di tipo cliente-servente**
  - **In relazione all'organizzazione del servizio e dei suoi dati**
- **Distribuzione verticale**
  - **Con ripartizione di autorità**
- **Distribuzione orizzontale**
  - **Con ripartizione del carico di lavoro**

Architetture distribuite – 1

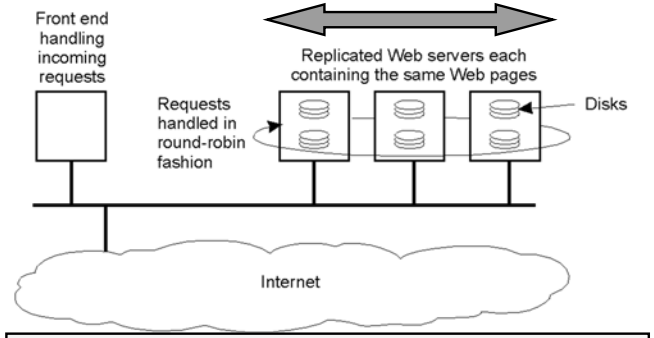
- **Distribuzione verticale**
  - **Componenti diversi dello stesso servizio possono essere assegnati a elaboratori distinti**
    - Sia sul lato servente che sul lato cliente (delegazione parziale)
  - **Il servizio richiede cooperazione articolata di componenti distribuiti**
- **Distribuzione orizzontale**
  - **Servente e cliente possono essere partizionati ma ogni loro componente può operare da solo**
  - **Ogni componente sa fornire "il" servizio richiesto**

Architetture distribuite – 2



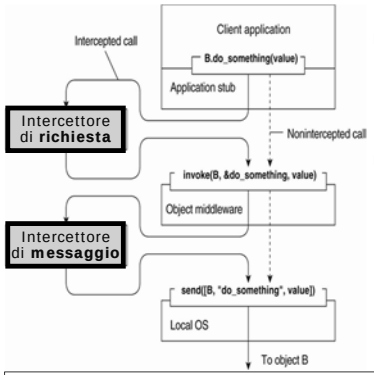
Nell'architettura a **distribuzione verticale** il servente visto dal cliente può essere esso stesso cliente di un componente servente cui sia stata demandata parte del servizio

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

Middleware moderno



- **Un approccio architetturale al *middleware* offre**
  - **Semplicità progettuale**
  - **Scarsa adattabilità**
- **Un approccio più flessibile si basa su**
  - **"*Separation of concerns*"**
  - **"*Computational reflection*"**
  - **Progettazione per componenti e connettori**
    - Gli intercettori in figura mostrano il posizionamento logico dei connettori

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