

Sistemi distribuiti: processi e concorrenza

Processi e concorrenza

SCD

Anno accademico 2008/9
Sistemi Concorrenti e Distribuiti

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

Corso di Laurea Magistrale in Informatica, Università di Padova 1/31

Sistemi distribuiti: processi e concorrenza

Cliente e servente concorrenti – 1

□ Lato cliente

- La concorrenza interna può mitigare l'effetto del ritardo di rete nelle comunicazioni distribuite
 - In un *Web browser* (lato cliente) conviene eseguire in parallelo
 - Attivazione della connessione TCP/IP → operazione bloccante
 - Lettura ed elaborazione dei dati in ingresso → può operare in *pipeline*
 - Trasferimento su video → può operare in *pipeline*
- Grazie alla concorrenza interna il cliente può anche supportare più sessioni parallele con le medesime caratteristiche
 - P.es.: i "tab" di Mozilla

Corso di Laurea Magistrale in Informatica, Università di Padova 2/31

Sistemi distribuiti: processi e concorrenza

Cliente e servente concorrenti – 2

□ Lato servente

- La concorrenza interna offre
 - Maggiore efficienza prestazionale
 - Ancor più utile e desiderabile che nel cliente
 - Maggiore modularità (specializzazione, semplicità) architetturale

Corso di Laurea Magistrale in Informatica, Università di Padova 3/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato cliente – 1

Trasparenza	Ruolo del MW di lato cliente
Accesso	Fondamentale – <i>stub</i> (RPC) o <i>proxy</i> (RMI)
Collocazione	Fondamentale
Migrazione / Spostamento	Desiderabile – gestione dinamica delle corrispondenze nome-indirizzo (<i>naming</i>)
Replicazione	Utile per nascondere la possibile interazione con più repliche del servente
Transazione	Utile (ma fondamentale dal lato servente)
Malfunzionamento	Desiderabile – p.es. il <i>caching</i> del <i>Web browser</i>
Persistenza	Non significativa (ma fondamentale dal lato servente)

Corso di Laurea Magistrale in Informatica, Università di Padova 4/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato cliente – 2

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

Corso di Laurea Magistrale in Informatica, Università di Padova 5/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato servente – 1

□ Due possibili organizzazioni di servente

- Iterativa o ricorsiva → distribuzione verticale
 - Il servente utilizza i servizi di altri serventi (interni o esterni)
 - La richiesta successiva potrà essere accolta **solo dopo** il completamento di quella corrente
 - Per soddisfare più richieste in parallelo bisogna replicare l'intero servente
- Concorrente → distribuzione orizzontale
 - Il servente si occupa solamente di accogliere richieste demandandone il soddisfacimento a un *thread* o processo distinto
 - Nuove richieste possono essere accolte **subito dopo** che quella corrente sia stata affidata all'esecutore selezionato
 - Per soddisfare più richieste in parallelo basta replicare gli esecutori (1 *dispatcher* – N *worker*)

Corso di Laurea Magistrale in Informatica, Università di Padova 6/31

Sistemi distribuiti: processi e concorrenza

Distribuzione verticale – 1

The diagram illustrates the vertical distribution of a request. It shows three layers: User interface (presentation), Application server, and Database server. The process starts with a 'Request operation' from the user interface to the application server. The application server then sends a 'Request data' to the database server. The database server returns 'Return data' to the application server, which then sends a 'Return result' back to the user interface. There are 'Wait for result' and 'Wait for data' periods indicated by dashed lines.

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

Corso di Laurea Magistrale in Informatica, Università di Padova 7/31

Sistemi distribuiti: processi e concorrenza

Distribuzione verticale – 2

The diagram shows a workstation sending a 'Richiesta ricorsiva' (recursive request) to a 'Preferred DNS Server'. This server then sends a 'Richiesta iterativa' (iterative request) to a 'Root DNS Server', which in turn queries a 'COM DNS Server', and finally a 'CONTOSO.COM DNS Server'.

Corso di Laurea Magistrale in Informatica, Università di Padova 8/31

Sistemi distribuiti: processi e concorrenza

Distribuzione orizzontale

The diagram shows a 'Front end handling incoming requests' connected to 'Replicated Web servers each containing the same Web pages'. Requests are handled in a 'round-robin fashion'. The servers are connected to the 'Internet' and have 'Disks'.

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

Corso di Laurea Magistrale in Informatica, Università di Padova 9/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato server – 2

Localizzazione del server

- Al server corrisponde una **porta (end-point)** del nodo sulla quale un processo apposito si pone in ascolto
- Porta preassegnata e nota**
 - IANA (Internet Assigned Numbers Authority)** attribuisce porte a specifici protocolli di livello Applicazioni
 - HTTP-80, FTP-20-1, SMTP-25, ...
- Porta assegnata dinamicamente**
 - Un **daemon** ascolta su porta **preassegnata** le richieste in per alcuni servizi
 - Richieste per lo stesso servizio girate su porta **assegnata dinamicamente** di cui viene informato il server corrispondente
- Super-Server**
 - Per richieste sporadiche non conviene mantenere serveri o **daemon** attivi
 - Un super-server ascolta **tutte** le porte corrispondenti e per ogni richiesta in arrivo risveglia (o crea dinamicamente) il server corrispondente
 - P.es. **inetd** di UNIX e GNU/Linux

Corso di Laurea Magistrale in Informatica, Università di Padova 10/31

Sistemi distribuiti: processi e concorrenza

Localizzazione del server

The first diagram, 'Assegnazione dinamica di porta server (interazione tramite daemon)', shows a client asking for an endpoint (1. Ask for end point) and the server machine's daemon registering the endpoint (2. Request service). The second diagram, 'Attivazione dinamica di server (interazione tramite super-server)', shows a client requesting a service (1. Request service) and the super-server creating a server for the requested service (2. Continue service).

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

Corso di Laurea Magistrale in Informatica, Università di Padova 11/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato server – 3

Interrompibilità del server

- Modello TCP/IP**
 - La rottura della connessione (p.es. per abbandono del cliente) comporta interruzione del servizio
 - Non immediata ma garantita senza confusione con richieste successive
- Dati "out-of-band"**
 - Il cliente può chiedere di dare precedenza a dati fuori sequenza ma di maggiore urgenza
 - Cliente e server devono intrattenere più di una sottoconnessione logica entro una stessa connessione di servizio
 - Una porta distinta per ogni sottoconnessione
 - Designazione di urgenza nell'intestazione dei pacchetti dati (p.es. TCP)

Corso di Laurea Magistrale in Informatica, Università di Padova 12/31

Sistemi distribuiti: processi e concorrenza

Problematiche di lato server – 4

- **Server senza stato (*stateless*)**
 - Non ricorda lo stato del cliente e non deve informarlo di eventuali cambi di stato di lato server
 - **Esempio:** un *Web server* accoglie richieste HTTP sulla porta 80, le soddisfa, dimentica il cliente subito dopo, e può cambiare locazione, stato ed esistenza dei propri *file* senza doverne informare alcun cliente
- **Server con stato (*stateful*)**
 - Il server ricorda lo stato del cliente
 - **Esempio:** il *MW* di lato cliente deposita *cookie* per fornire al server informazioni sullo stato di servizio del cliente
 - *Cookie* validi entro o tra sessioni

Corso di Laurea Magistrale in Informatica, Università di Padova 13/31

Sistemi distribuiti: processi e concorrenza

Server di oggetto – 1

- **Non fornisce alcun servizio di per se ma agisce come tramite di invocazione locale per conto di clienti remoti**
- **La realizzazione concreta del server determina se e come l'interfaccia e l'oggetto a lui associati possano essere separati**
- **È desiderabile che il server di oggetto sia capace di supportare diverse politiche di attivazione (accesso e gestione) dell'oggetto remoto**

Corso di Laurea Magistrale in Informatica, Università di Padova 14/31

Sistemi distribuiti: processi e concorrenza

Server di oggetto – 2

- **Politiche di attivazione determinano le modalità con le quali un oggetto remoto può essere invocato**
 - **Riguardo al caricamento dell'oggetto (in generale il *servant*) nella memoria ospite**
 - Al caricamento dell'oggetto segue la sua attivazione
 - L'oggetto può essere transitorio o persistente
 - **Conviene disaccoppiare il MW dalle politiche di servizio dell'oggetto**

Corso di Laurea Magistrale in Informatica, Università di Padova 15/31

Sistemi distribuiti: processi e concorrenza

Server di oggetto – 3

- **Una politica di attivazione comune per nodo viene realizzata da un singolo *object adapter***
 - **Concetto recepito dai *design pattern* della GoF**
 - **Fornisce metodi per**
 - Ricevere invocazioni remote in arrivo dal MW di distribuzione
 - Servizio funzionale
 - Inviare le invocazioni ai *servant* destinatari
 - Servizio funzionale
 - Registrare o rimuovere *servant* e influenzare le politiche di servizio
 - Servizio amministrativo

Corso di Laurea Magistrale in Informatica, Università di Padova 16/31

Sistemi distribuiti: processi e concorrenza

Server di oggetto – 3

Il diagramma illustra l'architettura di un server di oggetto. In alto, sono mostrati tre oggetti: Oggetto C, Oggetto A e Oggetto B. Ogni oggetto è associato a un'interfaccia. Sotto ogni oggetto c'è un *Skeleton*. In basso, ci sono due *Object Adapter*: *Object Adapter 1* e *Object Adapter 2*. *Object Adapter 1* è collegato all'interfaccia di Oggetto C, mentre *Object Adapter 2* è collegato alle interfacce di Oggetto A e Oggetto B. Un'interfaccia standard (entro il middleware) è condivisa tra i due adapter. In basso, un *Daemon dispatcher di RMI* è collegato ai *Skeleton* e ai *Object Adapter*. Un'annotazione indica che il *Daemon dispatcher di RMI* può essere generico per più oggetti perché non deve conoscere le specifiche interfacce degli oggetti che attiva.

Corso di Laurea Magistrale in Informatica, Università di Padova 17/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 1

- **Utile al bilanciamento di carico**
 - Molto importante nel passato
 - Oggi meno vista la potenza di calcolo disponibile
 - L'onere di comunicazione è diventato il vero collo di bottiglia
 - **Migrazione del cliente presso il server**
 - **Esempio:** un cliente deve effettuare una sequenza complessa di transazioni su base dati ove ciascuna transazione mobilita grandi volumi di informazione
 - Conviene che la parte coinvolta del cliente operi *localmente al server* per limitare il trasporto dati su rete
 - **Migrazione del server presso il cliente**
 - **Esempio:** un server che richiede al cliente molti dati da fornire in piccole unità di formato prefissato (p.es. *form*) come prerequisito a una transazione
 - Conviene inviare una parte del server *localmente al cliente* per predisporre più velocemente i dati trattati

Corso di Laurea Magistrale in Informatica, Università di Padova 18/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 2

- **Miglioramento delle prestazioni**
 - Esempio: parallelizzazione di ricerca su più siti inviando su ciascuno una copia dell'agente di ricerca
- **Flessibilità di configurazione del sistema**
 - Fissare staticamente una configurazione è difficile e rischioso quando le condizioni di lavoro non siano note o stimabili a priori
 - Conviene di più poterla adattare dinamicamente

Corso di Laurea Magistrale in Informatica, Università di Padova 19/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 3

- **Modelli di mobilità (1/2)**
 - **Debole (weak mobility)**
 - Solo segmento codice e dati di inizializzazione → il programma migrato riparte sempre dal suo stato iniziale (p.es. Java *applet*)
 - Richiede portabilità del codice, oppure speciale supporto dal compilatore
 - **Forte (strong mobility)**
 - Migra anche lo stato di esecuzione → l'esecuzione continua a destinazione
 - Grande complessità realizzativa
 - **Causata dal luogo di residenza iniziale (sender-initiated)**
 - Da cliente verso servernte → delicato, richiede autenticazione del cliente
 - **Causata dal luogo di destinazione (receiver-initiated)**
 - Da servernte verso cliente (p.es. Java *applet*) → più facile e sicura

Corso di Laurea Magistrale in Informatica, Università di Padova 20/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 4

- **Modelli di mobilità (2/2)**
 - **Esecuzione nel processo destinatario**
 - Esempio: le Java *applet* eseguono nello spazio di indirizzamento del processo *browser* di destinazione. (lato cliente)
 - Il processo destinatario deve essere protetto da codice malintenzionato
 - **Esecuzione in processo dedicato**
 - Maggiore protezione da intrusione per maggior onere di comunicazione locale
 - **Clonazione**
 - Simile al modello `fork()` di UNIX, con il processo clonato che eredita codice e stato dal processo clonante

Corso di Laurea Magistrale in Informatica, Università di Padova 21/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 5

Fuggetta, A., Pico, G.P., Vigna, G.: *Understanding Code Mobility*. IEEE Transactions on Software Engineering, 24(3):342-361, maggio 1998

Corso di Laurea Magistrale in Informatica, Università di Padova 22/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 6

- **Modelli di migrazione delle risorse su cui opera il codice mobile**
 - **Legame forte (binding by identifier)**
 - La risorsa ha identità fisica (p.es. indirizzo IP)
 - **Legame debole (binding by value)**
 - La risorsa non ha identità fisica ma solo specifiche caratteristiche attese e può essere copiata a destinazione
 - Esempio: riferimenti a librerie standard in linguaggi come C, C++, Java, Ada
 - **Legame flebile (binding by type)**
 - La risorsa deve solo appartenere a un tipo dato fissato che può avere più rappresentazioni
 - Esempio: una stampante PostScript, un dispositivo

Corso di Laurea Magistrale in Informatica, Università di Padova 23/31

Sistemi distribuiti: processi e concorrenza

Migrazione di codice – 7

- **Modelli di migrazione delle risorse**
 - **Risorse mobili (unattached resources)**
 - Il legame tra risorsa e nodo è lasco e lo spostamento ha basso costo
 - Esempio: un file dati
 - **Risorse legate (fastened resources)**
 - Il legame può essere lasco ma lo spostamento è oneroso
 - Esempio: una intera base dati, un intero sito *Web*
 - **Risorse immobili (fixed resources)**
 - Il legame è così stretto da non poter essere sciolto
 - Esempio: un dispositivo locale

Corso di Laurea Magistrale in Informatica, Università di Padova 24/31

Sistemi distribuiti: processi e concorrenza
Migrazione di codice – 8

Legame risorsa – nodo

		+ debole		+ forte
Legame processo – risorsa		Unattached		
	+ forte	By identifier	MV (GR se condivisa)	GR (o MV)
		By value	CP (MV o GR se condivisa)	GR (o CP)
	+ debole	By type	RB (GR o CP se non disponibile a destinazione)	RB (o GR o CP)
			Fixed	
			GR	GR (memoria globale)
			RB (GR o CP se non disponibile a destinazione)	RB (o GR)

Cosa fare con le risorse associate a codice che migra

MV: la risorsa può essere spostata
 CP: il valore della risorsa può essere copiato a destinazione
 GR: la risorsa può essere univocamente riferita da ogni parte del sistema
 RB: la risorsa può essere rappresentata a destinazione da una risorsa analoga

Corso di Laurea Magistrale in Informatica, Università di Padova 25/31

Sistemi distribuiti: processi e concorrenza
Migrazione di codice – 9

Diagram illustrating migration types based on process-resource and resource-node binding strength. The vertical axis represents 'Legame processo – risorsa' (process-resource binding) from 'debole' (weak) to 'forte' (strong). The horizontal axis represents 'Legame risorsa – nodo' (resource-node binding) from 'debole' to 'forte'. Migration types are categorized as Unattached, Fastened, and Fixed.

- By identifier:** MV (GR se condivisa), GR o MV, GR
- By value:** CP (MV o GR se condivisa), GR o CP, GR (memoria globale)
- By type:** RB (GR o CP se non disponibile a destinazione), RB (GR se non disponibile a destinazione)

Corso di Laurea Magistrale in Informatica, Università di Padova 26/31

Sistemi distribuiti: processi e concorrenza
Agenti software – 1

- Definizione 1**
 - Unità autonome capaci di eseguire compiti collaborando con altri agenti anche remoti
- Definizione 2**
 - Processo autonomo capace di reagire a cambiamenti nel proprio ambiente e di provocarne eventualmente in collaborazione con utenti o altri agenti
- Caratteristiche salienti**
 - Capacità di lavoro autonomo e di collaborazione
 - Agenti collaborativi

Corso di Laurea Magistrale in Informatica, Università di Padova 27/31

Sistemi distribuiti: processi e concorrenza
Agenti software – 2

- Agenti mobili**
 - Con mobilità forte (più spesso) ma anche debole
- Agenti di interfaccia**
 - Assistono uno o più utenti nell'uso di applicazioni
 - Esempio: gli *assistants* di molte applicazioni da ufficio
 - Hanno capacità di apprendimento
- Agenti di informazione**
 - Stazionari: agiscono su informazione in ingresso
 - Esempio: un filtro di posta elettronica
 - Mobili: cercano attivamente informazione ove essa risiede

Corso di Laurea Magistrale in Informatica, Università di Padova 28/31

Sistemi distribuiti: processi e concorrenza
Agenti software – 3

Proprietà	Di tutti?	Capacità implicata
Autonomia	Si	Prendere iniziative proprie
Reattività	Si	Rispondere prontamente a cambiamenti nell'ambiente
Proattività	Si	Intraprendere azioni che producano cambiamenti nell'ambiente
Comunicazione	Si	Scambiare informazioni con utenti e altri agenti
Continuità	No	Avere un tempo di vita medio-lungo
Mobilità	No	Poter migrare da un sito all'altro
Adattività	No	Apprendimento

Proprietà comportamentali degli agenti software

Corso di Laurea Magistrale in Informatica, Università di Padova 29/31

Sistemi distribuiti: processi e concorrenza
Agenti software – 4

```

    graph TD
      AA[Autonomous Agents] --> BA[Biological Agents]
      AA --> RA[Robotic Agents]
      AA --> CA[Computational Agents]
      CA --> SA[Software Agents]
      CA --> ALA[Artificial Life Agents]
      SA --> TSA[Task-specific Agents]
      SA --> EA[Entertainment Agents]
      SA --> V[Viruses]
    
```

Tratto da: <http://www.msc.memphis.edu/~franklin/AgentProg.html>

Corso di Laurea Magistrale in Informatica, Università di Padova 30/31

