istemi Distribuiti 1 Marzo 2020

S

DaveRhapsody

1

Indice

Introduzione

Un sistema distribuito è un insieme di componenti hardware e software che comunicano tramite scambi di messaggi in rete. Il concetto di componente è fondamentale, può essere per l'appunto sia hardware che software.

Nel dettaglio: E' un sistema di componenti **autonome**, nel senso che le componenti sono indipendenti tra loro MA concorrono tutte allo stesso scopo. Per apparire come unico componente occorre generare una sorta di collaborazione **SENZA** memoria condivisa.

Autonomia: Ognuno svolge il proprio compito in modo indipendente (sia in termini di tempo che dati) da ogni altro. Pertanto occorre che le componenti siano **SINCRONIZZATE** (Reti e Sistemi insegna ->). Questo insieme di nodi appaiono all'utente finale come un unico sistema **coerente** senza che lui sappia dove vengono processati (e nemmeno come) i dati. Il sistema è un blocco unico agli occhi dell'utente MA obv no.

Concetto di trasparenza: Il livello di trasparenza lo si decide in fase di progettazione del sistema, fondamentalmente è il livello tale per cui l'utente sappia come vengono processati i dati, e cosa e come viene eseguito dal sistema. (Un sistema trasparente, è un sistema che ti fa vedere vita, morte, miracoli, luci, coriandoli, sassi e Babbo Natale).

0.1 Caratteristiche fondamentali

- Non c'è memoria condivisa
- L'esecuzione è concorrente (allo stesso istante)
- Non ci sono stati del processo o della memoria, ogni nodo è a sè un processo!
- NON C'E' UN CLOCK GLOBALE, niente scheduling. controllo (globale), si fa tutto per scambio di messaggi
- Non esiste un fallimento globale ma un fallimento della singola componente.

  Sistemi Realtime: Noi non utilizziamo questo genere di dispositivi, studiamo e prendiamo in esame dispositivi best-effort

- Architetture software

0.2 Architetture software

E' la struttura del sistema, di come sono organizzate le componenti, quali sono i protocolli, le interfacce, etc.

- Architetture a Livelli (tier)
- Architetture a oggetti
- Architetture centrate sui dati
- Architetture event-based (ajax)

0.2.1 Stratificazione

L'idea è formare degli strati di complessità, fondamentalmente nulla vieta di fare tutto a livello application, lì si impreca per davvero, ma soprattutto non ci sarebbe una specializzazione. Il motto è "Fai una cosa, ma falla bene".

0.3 Tipi di sistemi operativi

In base al tipo ho un livello diverso di trasparenza

- DOS: Distributed OS
- NOS: Network OS: L'OS ti dà delle librerie e supporti per supportare delle applicazioni MA non nasconde le comunicazioni tra le varie applicazioni.
- Middleware: Hai un insieme di sistemi di rete che forniscono primitive per sostenere comunicazione tra sistemi E costruirci delle logiche che consentano di vedere questo come unico sistema (Sì, il middleware è a sua volta un'applicazione distribuita)

0.4 Servizi del middleware

- Naming
- Accesso trasparente
- Persisenza: avviene una memorizzazione di dati persistente
- Transazioni distribuite: Va garantita la consistenza dei dati
- Sicurezza dei dati: integrità computazionali

Riporto il riassunto presente nelle slides

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

Capitolo 1

Modello Client-Server

E' un'interazione basata su richiesta-risposta tra una componente e l'altra.

Funzionamento:

1. Client invia la request
2. Server la riceve, client aspetta
3. Server elabora
4. Server risponde alla richiesta

Quando client aspetta, quest'ultimo è in standby, in attesa, fermo, inchiodato. Il server si attiva ad ogni richiesta che arriva. Richiesta e risposta hanno un tempo ovviamente proprio, che deriva dalla rete di riferimento, chiaramente se si fa tutto sulla stessa macchina si parla di microsecondi.

Osservazione: Un dispositivo è sia client che server, può essere sia uno che l'altro.

Tipi di accesso:

- Accesso a server multipli (nel senso che sono duplicati i server)
- Accesso via proxy

1.1 Problemi dei Sistemi Distribuiti

- Identificazione della controparte: Identifico la risorsa, tipo con address etc.
- Accesso alla controparte: Dove accedo? Chi è il mio access point
- Comunicazione 1: Definisco il formato dei messaggi scambiati
- Comunicazione 2: Capisco il contenuto del messaggio in seguito all'estrazione

  Esempio: I tipi per i dati, che specificano un insieme di valori ed operazioni fattibili su un determinato dato.

1.2 Trasparenza

Significa nascondere all'utente **come** viene ottenuto un determinato risultato. E' per intenderci ciò che salva coloro che programmano tutto nel main. Come si fa?

- **Naming**: non mi connetto per indirizzo IP ma per indirizzo web, link.
- **Accesso trasparente**: Come accedo in maniera trasparente? Anche qui nomi simbolici, qualcosa che non mi faccia capire la locazione della controparte
- **Location Trasparency**: Hai una stampante a casa e devi stampare una foto di gattini? Ti serve sapere dove si trova, non puoi cercare a caso su google, quindi devi sapere fisicamente dove hai la stampante. Invece se ti serve una calcolatrice online tipo wolfram alpha, digiti l'equazione, la risolve, ma tu non sai una bega di dove è stata risolta.
- **Relocation o trasparenza mobile**: Posso spostare le risorse mentre le uso (Cellulari, dispositivi wifi, ci siamo capiti).
- **Migrazione**: Posso spostare un sito web da un pc del 91 ad un pc del 2020, il servizio quello è, cambierà la velocità (tempi di elaborazione + request + response)
- **Replicazione**: Hai una serie di server identici, tu ti connetti ad uno o l'altro, cambia niente
- **Concorrenza**: In più utenti accediamo allo stesso servizio (spotify)
- **Trasparenza del fallimento**: Ciò che resta dopo un fallimento di una componente deve colmare i danni di una componente morta.
- **Persistenza**: Non mi accorgo di quando un sistema è riavviato o no

In alcuni casi è impossibile nascondere un fail o lo stato di una applicazione, se ti crasha Word, vedi che ti è crashato Word, quindi in qualche modo va comunicato all'utente "Ueh, fa schifo Uord, usa OpeNoFFiSsszsxxs". Ma tra l'altro non sempre è utile nascondere, prendete l'esempio di prima della stampante.

1.3 Separazione tra interfaccia e realizzazione

Costruisco un'astrazione logica che nasconde i dettagli implementativi di ciò che sta diestro. Ogni componente pubblica il What cioè COSA fa, ma non ti spara fuori anche l'HOW, cioè COME lo fa, è il concetto di information hiding di Java.

Esempio: Gestione delle stringhe nei vari linguaggi. Una stringa è una concatenazione di caratteri, la gestione di append, copy, concat etc. sono how, il risultato finale è il what

1.4 Politiche versus mechanism

Un Sistema distribuito è composto da.. Va beh avete capito. Progressivamente si va verso un unico enorme (utopico) sistema complesso organizzato con delle policy stabilite. Le politiche banalmente sono una serie di regole ad esempio prendete il context switch di Unix.

- Il CS è un meccanismo
- Il Round Robin invece è la policy di come è gestito un comportamento

1.5 Concetto di protocollo

Un protocollo è un insieme di regole che definiscono un formato, l'ordine, payload, operazioni da compiere alla ricezione od all'invio di un messaggio.