Sistemi distribuiti

Funzioni di un sistema distribuito

Obiettivi e funzioni

Architetture di elaborazione distribuite

Un'architettura di elaborazione distribuita e' costituita da un'insieme di sistemi di elaborazione interconnessi attraverso la rete.

Ciascuna macchina avra':

- Almeno un processore;
- Della memoria locale;
- Periferiche locali;
- Periferiche condivise tra i vari sistemi;
- Risorse globali accessibili alle altre macchine;
- La rete di comunicazione stessa costituisce una periferica.

La caratteristica principale e' l'eterogeneita' dei singoli sistemi, comportando delle difficolta' di gestione per il sistema operativo.

Un'architettura di elaborazione distrivuita e' costituita da un'insieme di **siti** in cui vengono installate delle **macchine** (host, computer, nodi) che possono fungere da **server e/o client**.

I vantaggi di un'architettura di elaborazione distribuita sono:

- La possiblita' di integrare sottosistemi: esistono dei sottosistemi gia' acquisiti in vari siti se decidiamo di interconnetterli con gli altri sottosistemi nella rete possiamo creare un'interazione a livello dell'intero sistema;
- La condivisione delle risorse;
- Parallelismo della computazione: avendo piu' sistemi di elaborazione, una computazione puo' essere 'spezzata' e distribuita su altri sistemi di elaborazione in modo da avere un parallelismo fisico nella computazione ottenendo un aumento della velocita' di calcolo (**computation speedup**);
- Interazione con utente: permette di aumentare l'astrazione e avvicinarsi di piu' agli utenti;
- Riduzione della complessita' e del costo dei singoli componenti per adattarli alle specifiche esigenze (**downsizing**);

- Aumenta l'affidabilita' totale del sistema (**reliability**), la tolleranza ai guasti (**fault tolerance**) e la sua disponibilita' (**dependability**);
- Forte scalabilita' del sistema (**scalability**) poiche' e' possibile aggiungere funzioni, servizi e componenti senza dover gettare cio' che era stato precedentemente acquisito.

Sistemi operativi per architetture di elaborazione distribuite

Esistono due famiglie di sistemi operativi:

- Sistemi operativi di rete - **Network Operating Systems**: gestiscono l'integrazione tra i singoli componenti del sistema distribuito ma lasciano ancora vedere agli utenti la struttura e le caratteristiche dei componenti e della rete. Un processo sa dove si trovano le risorse che gli servono.
- Sistemi operativi distribuiti - **Distributed Operating Systems**: coprono l'intera architettura distribuita, nasconde le differenze tra le singole macchine e fa apparire l'insieme della macchine interconnesse in rete come un unico sistema di elaborazione. Il fatto che sia distribuito anche geograficamente diventa un dettaglio marginale, tutto e' trasparente ai processi. In questo modo e' piu' semplice organizzare l'uso delle risorse da parte degli utenti.

Sistema operativo di rete

Un sistema operativo di rete deve consentire l'accesso alle risorse informative e fisiche che sono condivise nella rete. Per questo le attivita' che deve supportare sono:

- Collegamento per eseguire un'elaborazione remota:
  - Login remoto: Telnel/SSH;
  - Comunicazione tra processi remoti: Socket;
  - Attivazione di procedure remote: RPC;
- Consentire l'uso di risorse fisiche remote:
  - Stampanti remote: spooling di stampa remoto.
- Accesso ai file remoti:
  - Trasferimento remoto di file: FTP;
  - File server remoto: montaggio di file system;
- Servizi generali come la posta elettronica.

Sistema operativo distribuito

Ha un approccio completamente diverso: deve rendere trasparente tutta l'architettura fisica del sistema di elaborazione distribuito.

Deve quindi:

- Rendere accessibili le risorse remote come se fossero risorse locali;
- Consentire la migrazione di dati e processi con una gestione trasparente da parte del sistema operativo.

Si effettua una migrazione dei dati quando si vogliono spostare dei dati dalla macchina locale alla macchina remota o viceversa. Si ha:

- Il meccanismo di copia-elaborazione-salvataggio viene svolto in maniera trasparente da parte del sistema distribuito tramite FTP automatizzato;
- Una copiatura del file in modo simile a quando si effettua la paginazione;

Il problema consiste nella compatibilita' di rappresentazione dei dati tra le varie macchine e quindi delle tecniche che possono essere utilizzate per nascondere le differenze.

La **migrazione computazionale** prevede la migrazione di singole procedure o dell'intero processo, a partire dalla macchina in cui erano in esecuzione inizialmente verso macchine remote nel sistema distribuito.

La migrazione di una **procedura** avviene attraverso la chiamata di una procedura remota (RPC). In questo caso il processo locale attiva una procedura su una macchina remota come se attivasse una sua propria procedura nel sistema operativo locale. Esistera' un processo remoto che si fara' carico dell'esecuzione della procedura richiesta scambiando messaggi con il processo locale. L'obiettivo e' quello di andare ad utilizzare dell'hardware remoto perche':

- L'hardware e/o il software remoto sono preferibili rispetto a quello locale;
- In remoto esistono delle risorse specifiche che non esistono localmente;
- La quantita' di dati su cui si vuole operare e' particolamente grande rendendoli non-trasportabili all'interno della rete;
- Non si vuole spostare alcun dato per garantire la protezione e la riservatezza.

La migrazione di un **processo** puo' essere fatto con:

- La migrazione effettiva del processo;
- Tramite la tecnologia degli **agenti mobili**.

Le motivazioni di tale migrazione sono:

- Effettuare un bilanciamento del carico di lavoro sui processori delle varie macchine;
- Velocizzare l'elaborazione (valutando il tradeoff con l'overhead generato dallo spostamento di un processo da una macchina ad un'altra);
- Preferibilita' dell'hardware e/o software remoti;

- Disponibilita' di risorse specifiche;
- Per accedere a dei dati che devono rimanere in remoto.

Un sistema operativo distribuito deve inoltre fornire un **file system distribuito**. Cio' puo' essere effettuato montando i file system remoti in modo omogeneo e trasparente come se fossero file system di un altro disco locale. Cio' deve essere fatto in modo tale che il file system globale che si venga a creare venga visto in modo identico qualunque sia la macchina.

Inoltre deve essere possibile usare risorse fisiche nell'ambiente distribuito come le stampanti (non deve essere necessaria alcuna configurazione su nessuna macchina). Analogamente devono essere forniti i servizi distribuiti come la posta elettronica.

Robustezza

Un sistema operativo di rete o distribuito deve farsi carico della gestione della robustezza dell'architettura. Un'architettura distribuita e' estremamente robusta rispetto ai guasti e ai malfuzionamenti rispetto ad un singolo sistema di elaborazione.

Il sistema operativo dovra' introdurre delle tecniche di gestione per:

- Rilevazione dei guasti;
- Mascheramento degli errori eventualmente indotti;
- Riconfigurazione del sistema per escludere i componenti guasti;
- Ripristino e reinclusone dei sistemi che si erano guastati.

Il **rilevamento dei guasti** puo' essere effettuato tramite:

- Monitoraggio periodico delle macchine mediante meccanismi di handshaking: se non viene ottenuta una risposta si puo' dedurre quali macchine non sono disponibili e provvedere ad escluderle dalla computazione attiva;
- Rilevamento di un tempo massimo di attesa (timeout): se non viene risposto ad una richiesta entro un tempo massimo, la macchina in questione verra' considerata come non funzionante;
- Computazioni duplicate su macchine diverse nella rete e confronto dei risultati: una discordanza dei risultati porta a rilevare un possibile guasto all'interno del singolo sistema

In caso di criticita' dell'applicazione puo' diventare necessario **mascherare gli errori**:

- Se si hanno computazione moltiplicate su piu' macchine, attraverso un meccanismo di votazione si potra' decidere a maggioranza quale e' il risultato corretto (consensus algorithm);
- E' possibile duplicare le risorse garantendo un mascheramento nel caso di guasti.

Deve essere garantita la **riconfigurazione** del sistema in caso di interruzione di collegamento o guasto ad una macchina:

- Riconfigurando le tabelle di instradamento per escludere il collegamento o la macchina guasti;
- Sostituzione la macchina guasta nei suoi compiti con altre risorse di elaborazione del sistema (eventualmente accettando delle funzionalita' degradate).

Infine, una volta riparato il componente guasto, il **ripristino** del sistema distribuito consente di:

- Riconfigurare l'archittura distribuita reintegrando i componenti riparati:
  - Informando le macchine tramite handshaking;
  - Aggiornare nuovamente le tabelle di instradamento;
  - Provvedere alla garanzia della consistenza delle informazioni nei vari siti (in particolare in quelli ripristinati).
- Deve essere garantito il completamento di quei servizi che non erano andati a buon fine a causa dell'interruzione di collegamento o guasto (es. gestione posta non consegnata).

Aspetti progettuali

Tutti questi aspetti vanno previsti nella progettazione e configurazione del sistema operativo per le architetture distribuite.

Si deve garantire:

- Trasparenza di allocazione delle risorse (processori, dispositivi di memoria, periferiche, file, ecc) per non dover sovraccaricare i programmatori e gli utenti di conoscenze non strettamente necessarie al raggiungimento dei loro fini (analogamente a quanto avviene con il file system);
- Il supporto alla mobilita' dell'utente, alla computazione e ai dati;
- Tolleranza ai guasti;
- Scalabilita' delle architetture e delle funzioni.

Comunicazioni in rete

Protocolli di comunicazione

Problemi

I problemi che si vogliono affrontare sono:

- La necessita' di effettuare comunicazioni di tipo asincrono in un ambiente distribuito per garantire

il successo della comunicazione rimanendo non-bloccati nelle comunicazioni;
- Tenere conto della probabilita' di errori durante la comunicazione e quindi trovare un modo di inviare i messaggi con una buona probabilita' di sucecsso;
- L'interazione tra ambienti eterogenei che devono essere integrati.

Obiettivi

- Semplificare la prograttazione delle applicazioni creando uno strato di progettazione della rete sul quale le operazioni di comunicazione si possono appoggiare vedendolo come un ambiente omogeneo tra componenti eterogenei;
- Astrazione della visione delle comunicazioni in rete e quindi una virtualizzaizone virtualizzazione delle stesse;
- Gestione efficiente degli errori e dei guasti.

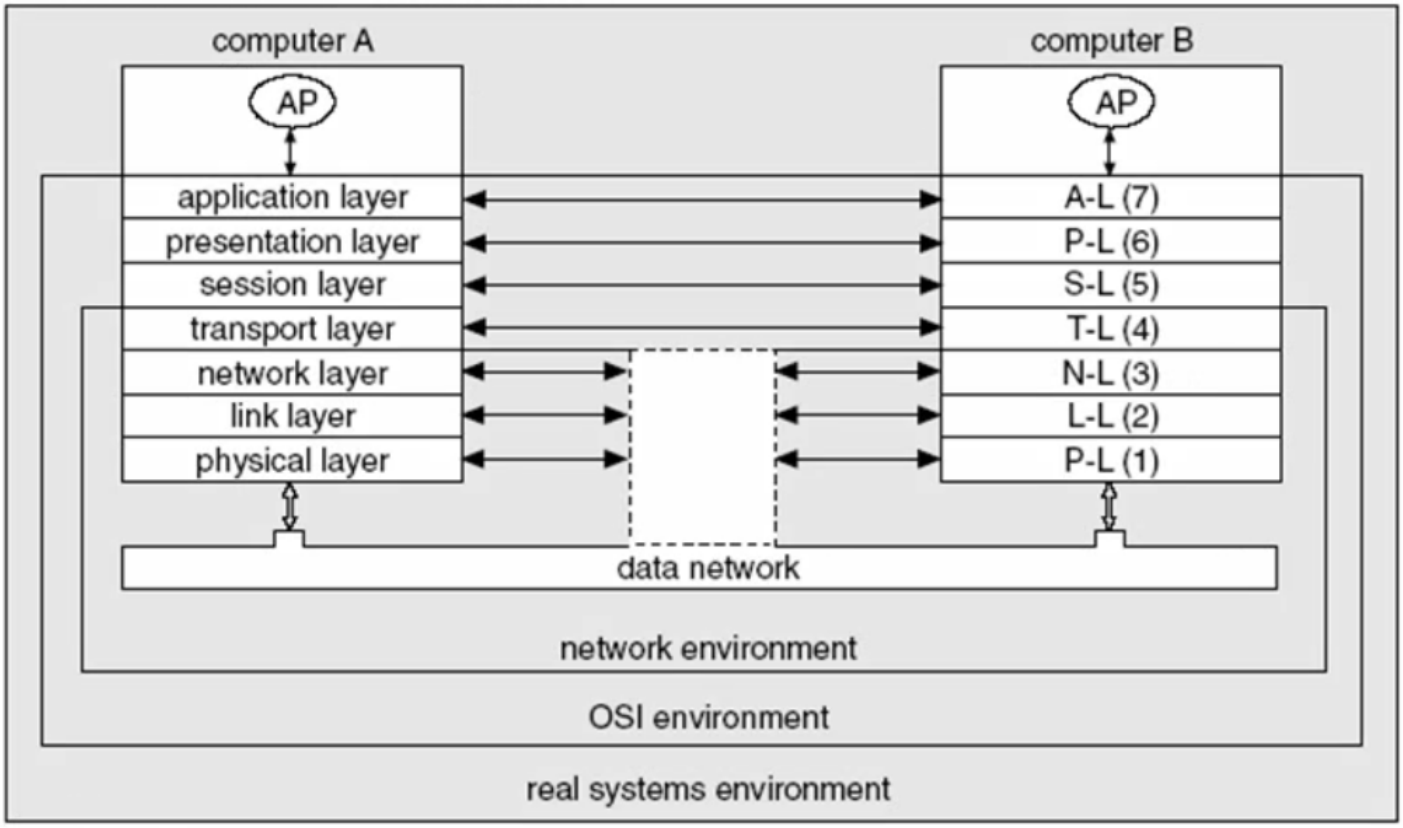
Soluzione

Introduzione di protocolli di comunicazione come **driver** di gestione della 'periferica' rete.

Il sottosistema dedicato alle comunicazioni in rete sara' suddiviso in strati e la comuincazione verra' gestita tra strati equivalenti tramite protocolli specifici.

Modello teorico

Il modello teorico che ben descrive la comunicazione in maniera astratta e virtuale è il modello ISO/OSI.Questo modello è stato introdotto dalla società internazionale degli standard (ISO) per permetterel’interconnessione tra sistemi aperti eterogenei (Open System Interconnection).



Questo modello prevede una pila di strati che provvede a rendere astratti e trasparenti i vari aspetti di complessità crescente (da livello fisico fino alla gestione dell'interazione a livello di applicazione).

Gli strati più bassi (fisico - trasporto) creano un ambiente di rete omogeneo ed indipendente dalle strutture fisiche che effettivamente si vanno ad utilizzare.In questo modo i processi possono colloquiare tra loro senza sapere come è realizzata fisicamente laconnessione al di sotto.

Al di sopra di questi viene creata una serie di strati che aumenta il livello di astrazione, fornendo una modalità standard ed omogenea per le applicazioni di colloquiare tra di loro, introducendo anche delle applicazionistandard a livello di rete.

Strato fisico

Lo strato fisico si occupa della gestione della trasmissione dei bit a livello meccanico ed elettrico, specificando come devono essere realizzate fisicamente le interconnessioni.

Strato data-link

Lo strato del collegamento tra i dati gestisce l'invio e la ricezione del singolo pacchetto, ossia della singola porzione di messaggio di lunghezza fissa, e provvede a gestire la rilevazione e la correzione di errori.

Strato rete

Lo strato di rete è quello che si occupa della connessione e dell'instradamento dei pacchetti all'interno dellarete (definendo l'indirizzo al quale deve essere inviato un pacchetto, e decodificando l'indirizzo dei pacchetti inentrata per capire se il pacchetto è destinato a quella macchina).

Strato trasporto

Lo strato di trasporto si occupa di partizionare i messaggi in pacchetti di lunghezza fissa e attivarne l'invio,mantenendo l'ordine degli stessi.Gestisce quindi la ricostruzione ad alto livello, gestendo gli errori a livello di messaggio.

Strato sessione

Lo strato di sessione provvede a realizzare le sessioni di comunicazione, e quindi a garantire la comunicazione a livello di processi (e quindi lo scambio dei messaggi dall'inizio alla fine della sessione,l'ordinamento di tali messaggi e la gestione dei relativi errori).

Strato presentazione

Lo strato di presentazione ha come obiettivo quello di risolvere le differenze di formato tra le varie macchine. Provvede quindi a convertire i vari formati, e provvede a gestire l'invio in modalità semi-duplex o full-duplex(una direzione o due direzione contemporaneamente).

Strato applicazione

Lo strato di applicazione gestisce l'interazione a livello di standard di applicazione, ad esempio con il trasferimento di file, connessioni remote, posta elettronica, basi di dati distribuite ecc.

Pila del protocollo ISO/OSI

La pila del protocollo ISO/OSI provvede a creare per ogni livello di astrazione, una visione specifica per le caratteristiche della comunicazione tra processi.

I dispositivi fisici di connessione realizzano l'interconnessione fisica tra le apparecchiature in rete.

Al di sopra dello strato di trasporto si riesce a vedere un servizio di comunicazione e scambio messaggi indipendente dalle caratteristiche della rete.

Al di sopra dello strato di presentazione si riesce a vedere un servizio di scambio dei messaggi indipendente dalla sintassi usata all'interno delle singole macchine della rete.

A livello di strato di applicazione si potranno vedere dei servizi distribuiti che possono essere utilizzati dai processi applicativi.

Messaggi nel protocollo ISO/OSI

I messaggi scambiati sono incapsulati, aggiungendo progressivamente degli header passando per i vari strati della pila. All'ultimo strato, verranno aggiunti un header e un footer, a garantire la correttezza dell'invio delsingolo pacchetto.

Modelli reali

I modelli reali devono essere però più semplici ed efficienti da utilizzare. Devono essere soprattutto più rapidi nell'incapsulamento dei messaggi da inviare con le info di servizio e l'invio sulla rete. Diventano più difficili da realizzare, e meno astratti del modello ISO/OSI.

I modelli reali sono:

- Internet Protocol (IP);
- Trasmission Control Protocol (TCP)
- User Datagram Protocol (UDP);
- Ulteriori protocolli applicativi.

Relazione tra TCP/IP, UDP/IP e ISO/OSI

Il livello IP equivale allo strato di rete.

Il protocollo TCP o UDP equivalgono allo strato di trasporto.

Non sono definiti strati standard per il livello fisico e collegamento, e nemmeno per i livelli di presentazione e sessione.

A livello di applicazione può essere però definita una serie di protocolli specifici, basati sull'uso di TCP/IP o UDP/IP (HTTP, FTP ecc.).

Internet Protocol (IP)

Il protocollo internet è lo strato di rete, e serve a gestire l'instradamento dei pacchetti in rete.

Protocolli di trasporto

UDP si occupa di gestire le comunicazioni senza il concetto di connessione (e quindi è inaffidabile). Le applicazioni dovranno sapere che utilizzando tale tipo di comunicazione non viene garantita la correttezza della comunicazione.

TCP invece è orientato alle connessioni, e quindi nativamente affidabile, e cerca di recuperare gli errori occorsi nella comunicazione, se possibile.

Strati TCP/IP

A livello applicazione vi sono molti altri protocolli per:

- Realizzazione delle connessioni remote (Telnet, SSH);
- File transfer (FTP/SFTP);
- Realizzare servizi web (HTTP/S);
- Posta elettronica (SMTP);
- Risoluzione dei nomi (DNS).

Computazione distribuita

Distribuzione della computazione

Motivazioni

In un sistema di elaborazione composto da calcolatori connessi in rete, la possibilita' di avere piu' sistemi elaboranti in parallelo consente di:

- Aumentare la velocita' di elaborazione tramite il parallellismo;
- Accedere in maniera efficiente a risorse informative o fisiche;
- Elaborare grandi quantita' di dati localizzati;
- Tollerare i guasti.

Obiettivi

Spostare la computazione sulla macchina su cui si trovano le risorse adatte per raggiungere l'obiettivo applicativo.

Tecniche per la distribuzione della computazione

Esistono diversi approcci:

- Chiamate a procedure remote: il processo attiva remotamente le operazioni desiderate sulla risorsa considerata;
- Allocazione dei processi: spostare la computazione sulla macchina dove si trova la risorsa

desiderata;
- Utilizzare degli agenti mobili: sapendo dove reperire le risorse necessarie, spostano autonomamente la computazione sulla macchina adatta.

Tecniche di supporto alla computazione distribuita

E' importante fare uso di tecniche per:

- La comunicazione tra processi distribuiti;
- La sincronizzazione tra processi distribuiti.

Chiamata di procedura remota

Obiettivo

Eseguire una procedura sulla macchina su cui sono disponibili le risorse informative o fisiche necessarie, lasciando il resto del processo sulla macchina che l'ha attivata.

Chiamata di procedura remota

La chiamata di procedura remota RPC e' del tutto simile alla chiamata di procedura all'interno del processo oppure al sistema operativo.

Il processo chiamante e' entita' attiva, mentre la procedura chiamata e' un'entita' passiva.

L'RPC e realizzata mediante comunicazione tra processi basata su messaggi strutturati.

Realizzazione

Nell'RPC abbiamo:

- Un processo P chiamante sulla macchina 1;
- Il corpo della procedura su macchina 2;
- Il processo P segnalera' la necessita' di usare la procedura chiedendo al sistema operativo della macchina 1 di connettersi al demone della macchina dove si trova la procedura desiderata;
- Il demone riceve la richiesta e mette in esecuzione la procedura per conto del processo P;
- Il demone provvede a ritornare al processo P il risultati della computazione (eventualmente);

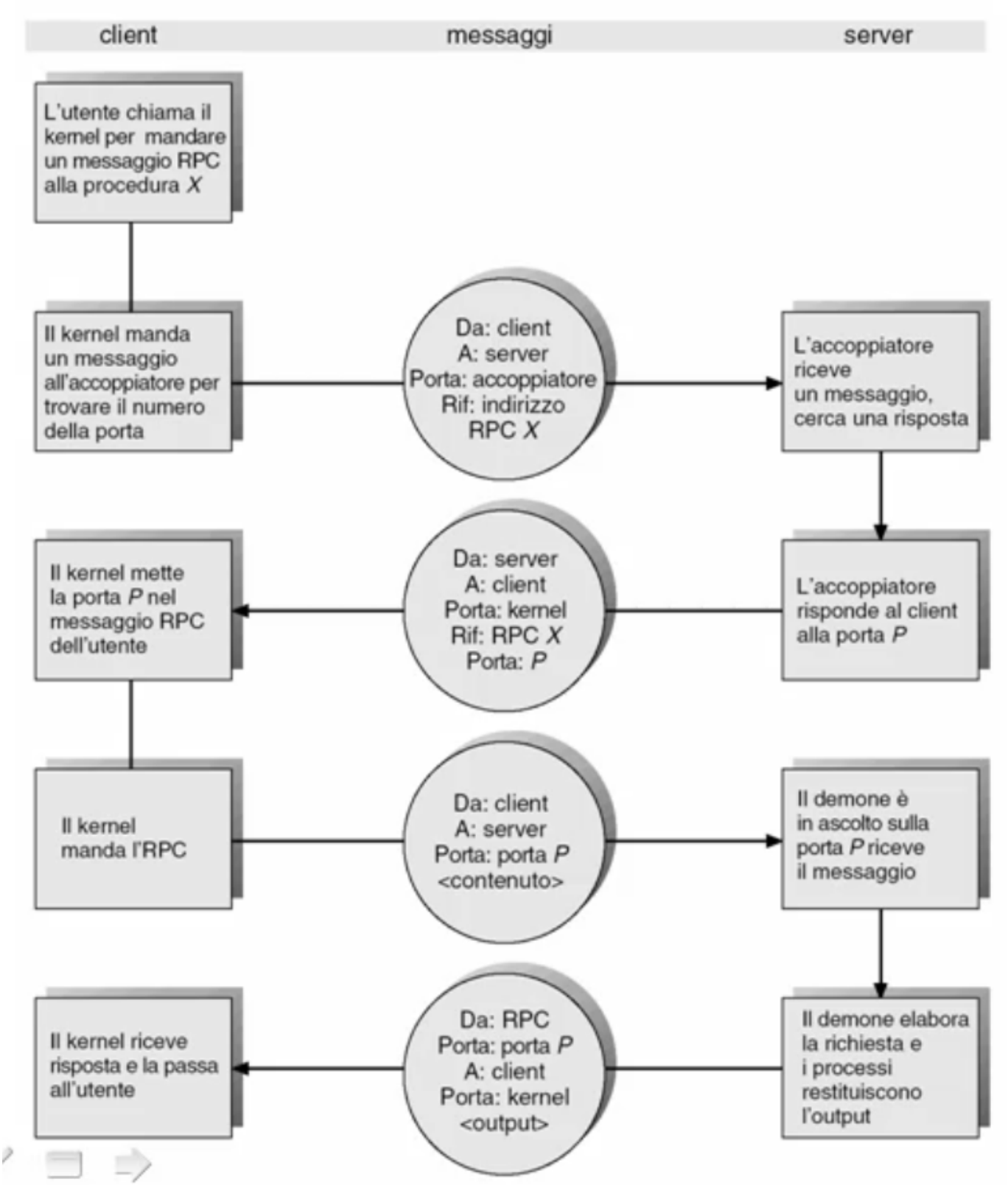
L'attivazione avviene l'invio di un messaggio strutturato:

- Identificare della funzione richiesta.

- Parametri.

La procedura desiderata e' composta da un **terminale remoto** chiamato **stub** che viene mandato in esecuzione dal demone quando un processo nella rete la richiede:

- Uno stub per ogni procedura;
- Parametri;
- Traduzione parametri (qualora non siano omogenei tra le macchine);
- Scambio messaggi.



Vantaggi

L'uso della RPC ha il vantaggio di presentare l'accesso alle funzioni esattamente come se fossero locali (nascondendo i dettagli del trasferimento delle informazioni). Mostra semplicemente la funzione remota come se fosse una funzione locale del sistema operativo.

Problemi

I dati possono essere rappresentati in modo diverso sulle varie macchine a seconda delle convenzioni, ed e' spesso necessario utilizzare una rappresentazione esterna dei dati in modo da avere uno standard di riferimento unico per garantire la comprensibilita' dei dati (XDR - XML Data Reduced).

Un altro problema e' definire la semantica della RPC. Esistono diversi approcci:

- Si puo' considerare che la procedura remota venga chiamata al piu' una volta. In questo caso e' necessario tenere uno storico delle richieste, in modo da capire se la procedura viene richiesta piu' di una volta da parte di un processo. Lo storico permette di verificare se una richeista pendente e' stata servita o meno;
- Si puo' considerare che la procedura remota venga chiamata esattamente una volta. In questo caso bisognera' garantire l'unicita' delle richieste e l'esecuzione remota (la responsabilita' dell'esecuzione dell'RPC e' del sistema operativo).

Esempio di uso delle RPC

Le RPC sono usate in molti casi tra cui la realizzazione di file system distribuiti:

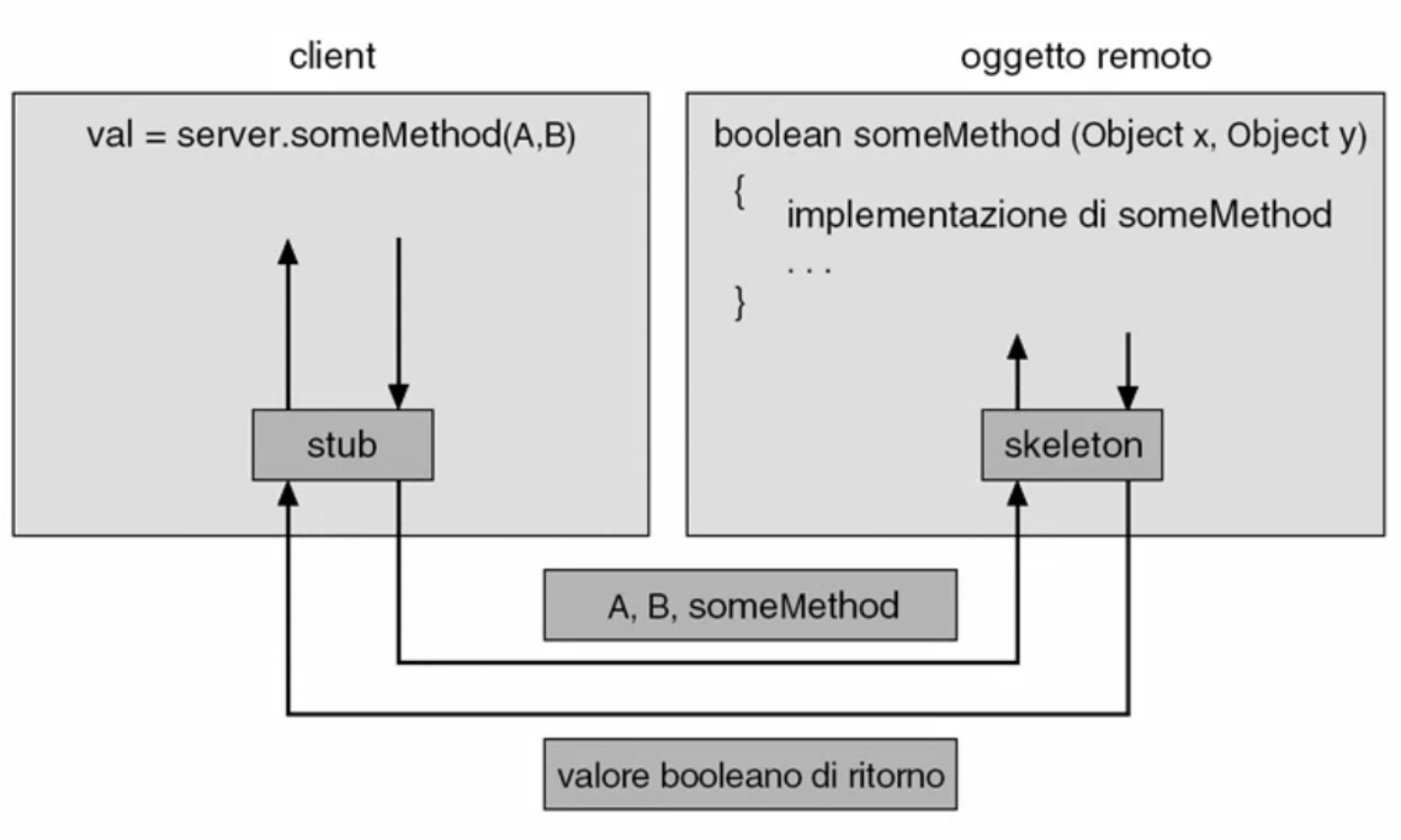
- Ci sono i demoni per gestire le RPC;
- Quando un client vuole accedere ad un file remoto, la richiesta di accesso al file viene mandata sulla porta associata al file system distribuito e il file server provvede a restituire il file desiderato. In caso di operazioni di lettura o scrittura remote queste operazioni potranno essere gestite direttamente sul file remoto.

Invocazione di un metodo remoto

La RPC e' una chiamata di procedura remota tipica di un qualunque linguaggio di programmazione imperativo.

L'estensione ad un linguaggio OOP viene usualmente chiamato **Remote Method Invocation - RMI**.

L'RPC diventa l'attivazione di un metodo appartenente ad un oggetto remoto. L'RMI permettera' di richiamare questi metodi per ottenre l'acesso ai dati desiderati.



Il client vedra' localmente l'interfaccia (stub) del metodo, e sara' il sistema di gestione dell'ambiente ad oggetti distribuito che si fara' carico di attivare il corpo della procedura remota sulla macchina desiderata e di riprendere i risultati della computazione, fornendoli al processo chiamante in maniera trasparente.

Dunque:

- In RPC: la gestione della connessione e' responsabilita' del programmatore;
- In RMI: la gestione e' trasparente al programmatore, gestita dall'ambiente di programmazione.

Allocazione dei processi

Obiettivi

In un ambiente distribuito si considera l'allocazione dei processi come una tecnica che mira a raggiungere diversi obiettivi:

- Attivare i processi sulla macchina piu' adatta ad effettuare la computazione tenendo conto del carico computazionale e delle risorse informative e fisiche necessarie;
- Migliorare lo sfruttamento dei processori: bilanciamento del carico;

- Migliorare lo sfruttamento delle risorse;
- Gestire in modo efficiente l'uso delle risorse minimizzando il carico computazionale di gestione;
- Realizzare la tolleranza ai guasti.

Allocazione statica

L'allocazione dei processi viene definita all'atto della loro attivazione ottimizzando uno o piu' obiettivi.

Una volta caricari i processi sui nodi della rete, l'allocazione e' permanente e i processi vengono attivati.

La tecnica di allocazione statica puo' essere:

- **Completa**: tutti i processi che devono essere attivati sono allocati contemporaneamente e poi sono attivati;
- **Incrementale**: quando un gruppo di procesi deve essere attivati, viene prima allocato tenendo fissa l'allocazione dei processi gia' attivati e poi viene attivato.

Per l'allocazione statica e' necessario definire una **funzione obiettivo** che si vuole ottimizzare:

- Sfruttamento dei processori:
  - Tempo di idle totale dei processori;
  - Distribuzione del tempo di idle dei processori;
  - Latenza media dei processi;
  - Throughput dei processi;
  - Tempo di risposta in sistemi in tempo reale;
  - ...
- Efficienza della gestione:
  - Minimizzazione del tempo di accesso alle informazioni;
  - Minimizzazione del tempo di accesso alle periferiche;
  - Minimizzazione del tempo di accesso ai servizi del sistema operativo;
  - ...
- ...

Quando si esegue l'ottimizzazione dell'allocazione statica bisogna tener conto dei vincoli:

- Locazione dei processi;
- Locazione delle risorse informative o fisiche necessarie con accesso solo locale:
  - Interazione con utente;
  - Basi dati;

    - Sensori e attuatori;
    - Sicurezza ed autenticazione;
    - ...
- Incompatibilita' con la locazione:
    - Hardware;
    - Software;
    - Sistema operativo;
- Incompatibilita' tra processi;
- ...

Gli algoritmi di allocazione sono vari e si distinguono per alcune caratteristiche:

- Modalita' di ricerca della soluzione:
    - Algoritmi deterministici;
    - Algoritmi euristici;
- Modalita' di esecuzione:
    - Algoritmi centralizzati;
    - Algoritmi distribuiti;
- Qualita' della soluzione:
    - Soluzione ottima;
    - Soluzione sub-ottima.

Gli algoritmi di allocazione possono operare in modo globale, su tutta la rete, oppure locale su una singola macchina. L'attivazione dell'algoritmo di allocazione scelto puo' essere eseguita dal processore (mittente) che vuole liberarsi di un processo o dal processore (ricevente) che vede arrivarsi il processo.

Allocazione dinamica

In questo caso l'allocazione dei processi puo' essere definita durante tutta la vita dei processi, andando ad ottimizzare uno o piu' obiettivi. Sostanzialmente l'allocazione non e' piu' permanente per ogni processo, ma puo' essere messa in discussione durante tutta la vita del processo, comportando, eventualmente, una riallocazione dello stesso.

L'allocazione dinamica puo' essere definita in modo:

- Totale: l'allocazione e' definita considerando tutti i processi contemporaneamente;
- Parziale: l'alocazione e' definita considerando un sottoinsieme di processi che soddisfa una regola di candidatura alla riallocazione.

L'allocazione dinamica puo' avvenire secondo varie regole:

- Allocazione periodica: ad intervalli regolari;
- Allocazione reattiva: avviene quando si verifica una condizione di riallocazione;
- Riallocazione volontaria: quando un processo richiede esplicitamente la riallocazione di uno o piu' processi.

Anche nel caso dell'allocazione dinamica esistono delle funzioni obiettivo da ottimizzare con alcuni vincoli.

Gli algoritmi che verranno utilizzati avranno caratteristiche analoghe a quelli dell'allocazione statica.

Nel caso dell'allocazione dinamica bisogna gestire alcuni problemi:

- Come far avvenire la migrazione dei processi:
  - Tener conto dello stato di evoluzione della computazione di un processo;
  - Trasferire il processo;
  - Riattivare il processo;
  - Assicurarsi della compatibilita' e traduzione della rappresentazione dei dati e del codice (per via delle macchine eteogenee);
- Gestire il costo della migrazione dei processi:
  - Tempo di gestione dell'algoritmo di allocazione;
  - Tempo di gestione della migrazione.

Agenti mobili

Obiettivi

L'obiettivo fondamentale con gli agenti e' quello di alzare il livello di astrazione della computazione secondo i principi dell'ingegneria del software.

Anziche' vedere un programma che viene messo in esecuzione e interagisce con altri programmi attraverso meccanismi di comunicazione e sincronizzazione, la computazione con agenti mobili vuole essere descritta tramite tecnologie ad oggetti con capacita' di esecuzione di azioni.

Modello della computazione ad oggetti

Identifichiamo gli oggetti come collezione di dati e metodi per la loro gestione.

Attraverso la modellazione degli oggetti e' possibile incapsulare le caratteristiche dei dati e dei metodi all'interno dell'oggetto stesso, mostrando all'esterno solamente un'interfaccia di alto livello.

I programmi vengono realizzati scrivendo l'interazione tra i vari processi e vengono eseguiti attivandoli come processi o come thread all'interno del sistema di elaborazione.

Modello della computazione ad agenti

Un agente e' un'entita' software **autonoma** che opera in un ambiente in maniera **pro-attiva e cooperante** decidendo come evolvere ed interagire con il mondo esterno.

Agenti mobili

Un agente mobile e' un agente con la capacita' di muoversi all'interno di un sistema distribuito, interagendo con le singole macchine per scoprire dove si trovano le risorse e i servizi desiderati.

Coordinamento distribuito tra processi

Sincronizzazione dei processi

La sincronizzazione dei processi in un ambiente distribuito ha lo scopo di:

> Creare un ordinamento toale tra tutti gli eventi in un sistema distribuito.

Non si ha pero' a disposizione ne' un orologio comune ne' una memoria comune, quindi non si hanno gli strumenti fisici in grado di realizzare un ordinamento totale.

Praticamente, bisogna accontentarsi di un ordinamento parziale degli eventi affinche' si possa creare una visione globale dell'evoluzione delle attivita' e degli eventi nei sistemi.

Relazione 'accaduto prima'

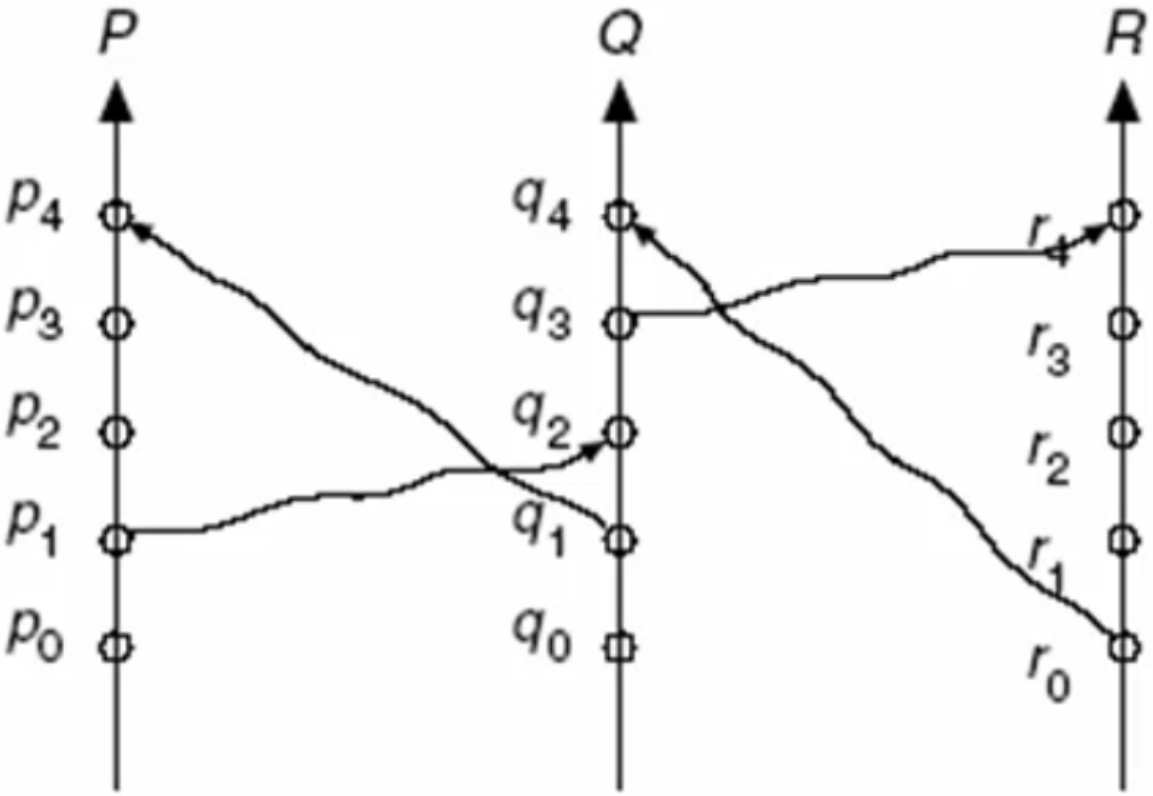
- Se A e B sono eventi dello stesso processo, ed A avviene prima di B, allora A->B (rappresenta la sequenza);
- Se A e' l'evento di trasmissione del messaggio in un processo e B e' l'evento di ricezione di quel messaggio da parte di un altro processo, allora A->B;
- Se A->B e B->C, allora A->C.

Le relazioni non sono riflessive.

Gli eventi che non sono in relazione sono concorrenti tra di loro e non si influenzano.

Se due eventi sono in relazione possono influenzarsi.

Esempio con 3 processi concorrenti:



Ordinamento globale

Per ottenere un ordinamento globale possiamo far riferimento solo ad eventi parziali e a strumenti esterni:

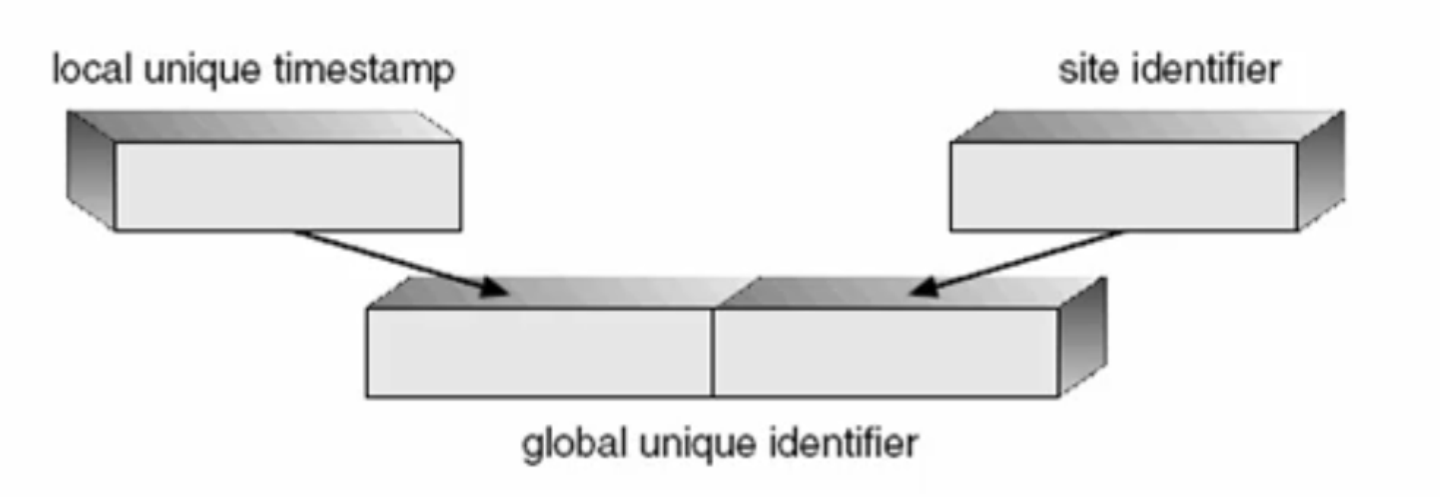
- Marca di tempo;
- Orologio logico che identifica l'evolvere del tempo parzialmente in modo che sia un orologio che incrementi in modo monotono;

Ogni volta che dei processi effettueranno una comunicazione l'orologio logico verra' fatto avanzare forzatamente.

Marca di tempo

Per generare delle marche di tempo uniche al livello del sistema distribuito e' necessario garantire che il valore della marca sia univocamente prodotto nell'interno sistema distribuito. Esistono due soluzioni possibili:

- Soluzione centralizzata: un unico processo distributore e' autorizzato a generare e diffondere le marche univoche;
- Soluzione distribuita: viene prodotta una marca di tempo locale, e componendola con l'identificatore del sito del sistema distribuito, e' possibile creare una marca temporale unica.



Per realizzare la marca di tempo locale si puo' utilizzare un orologio di sistema o un orologio logico locale.

Orologio logico

Gli orologi logici delle varie macchine hanno velocita' diverse, poiche' gli orologi fisici delle varie macchine possono avere velocita' diverse. Questo fa si che non e' possibile usare degli orologi fisici assoluti e tra loro sincronizzati, ma si preferisce usare degli orologi logici che vengono sincronizzati periodicamente, ogni volta che avvengono delle interazioni tra processi appartenenti a macchine con orologi diversi.

Un orologio viene aggiornato al valore n+1 quando un sito riceve una marca di valore n.

Mutua esclusione

Per realizzare la mutua esclusione esistono i metodi:

- Centralizzato:
  - Un processo coordina centralmente gli accessi alle sezioni critiche di tutti i processi nel sistema distribuito. Quando un processo vuole entrare nella sua sezione critica effettua una richiesta al coordinatore centralizzato che decide di concedere o meno l'accesso in funzione di cio' che hanno fatto precedentemente gli altri processi;
  - Verra' gestita una coda dei processi in attesa;
  - Le prestazioni sono limitate;

    - La tolleranza ai guasti e' scarsa;
- Distribuito:
    - Quando un processo P vuole entrare nella sua sezione critica
        - Genera una marca di tempo
        - Invia la richeista di entrata in sezione critica a tutti i processi;
    - Quando un processo Q riceve la richiesta di entrare in sezione critica:
        - Ritarda la risposta se e' in sezione critica;
        - Risponde immediatamente se non intende entrare in sezione critica;
        - Se desidera entrare nella propria sezione critica ma non vi e' entrato, compara la propria marca di tempo con quella di P: se la proprio e' piu' grande allora risponde immediatamente, altrimenti ritarda la risposta per entrare prima.

Il metodo distribuito e' una soluzione prima di starvation, deadlock e con una buona tolleranza ai guasti.

Un'altra tecnica e' quella del passaggio dei token. A partire da un gruppo di processi organizzati ad anello, il token indica l'autorizzazione ad accedere alla sezione critica. Quando un processo vuole entrare nella sua sezione critica puo' farlo in quel momento, trattendendo il token fino a quando non avra' completato l'uso della sezione critica. Se non desidera, o al completamento dell'uso della sezione critica, invia il token al processo successivo nell'anello, consentendo a questo di procedere ugualmente.

Atomicita'

E' necessario garantire che si possa attuare una sequenza di azioni in maniera indivisibile, atomica. Bisognera' estendere tali concetti nell'ambito dei sistemi distribuiti. Deve esiste un coordinatore delle transazioni su ciascuna macchina e gestire il coordinamento tra i coordinatori locali in modo da gestire le transazioni distribuite.

Il coordinatore locale deve:

- Iniziare la transazione;
- Dividere le transazioni in sotto-transazioni;
- Distribuire le sotto transazioni sulle macchine adatte;
- Valutare l'esito delle singole sotto-transazioni per decidere come si e' svolta la transazione complessiva (terminata correttamente o meno).

Per garantire la verifica del completamento di tutte le sotto-transazioni va introdotto un protocollo di **commit a due fasi**.

Quando la transazione inizia su una determinata macchina, tutti i siti che verranno attivati dalla transazione dovranno cooperare per definire l'esito della transazione;

Quando tutti i siti su cui sono state attivate le sotto-transazioni comunicano alla macchina d'origine della transazione che e' stata completata la porzione di transazione a loro assegnata e si attiva il protocollo di commit a due fasi.

Fase 1:

- Il coordinatore della macchina di origine aggiunge una stringa "prepare T" al suo log e invia a tutte le macchine coinvolte l'ordine di "prepare T" (preparare il completamento della transazione T);
- Quanto tale messaggio viene ricevuto dalle macchine, i relativi coordinatori decideranno se il commit sulla macchina e' possibile:
  - Se il commit non e' possibile: aggiungera' al proprio log "no T" e inviera' al coordinatore un *abort*;
  - Se il commit e' possible: aggiungera' al proprio log "ready T" e inviera' un messaggio simile al coordinatore;

Fase 2:

- Il coordinatore della macchina di partenza ricevera' tutte le risposte, oppure, nel caso di mancata risposta entro un tempo prestabilito, prendera' comunque una decisione finale;
- La transazione verra' chiusa con *commit* se vengono ricevuti messaggi "ready T" da tutte le macchine, aggiungendo "commit T" al proprio log, altrimenti la transazione verra' abortita aggiungendo "abort T" al log;
- Il risultato di tale decisione e' trasmesso a tutte le macchine;
- Quando un coordinatore rivece "commit T" o "abort T" registrera' nel loge tale rispettivo messaggio.

Un protocollo di commit a due fasi ha una buona tolleranza ai guasti, sia per le macchine in rete che per la struttura di interconnessione della rete stessa.

Concorrenza

Per gestire la concorrenza ci sara' la necessita' di:

- Un gestore delle transazioni globali e locali;
- Un file di logo in cui memorizzare le informazioni;
- Protocolli che provvedano ad effettuare la gestione dell'accesso in mutua esclusione in modo da garantirne la correttezza. Possono essere bloccanti o non bloccanti.

Protocolli bloccanti

Realizzano la gestione del blocco in ambiente distribuito garantendo l'accesso alla risorsa un processo alla volta.

Permettono eventualmente la replicazione locale dei dati per velocizzarne l'acesso.

Il modo di gestire il blocco puo' essere esclusivo o eventualmente condiviso (in caso le operazioni consentano di accedere alle informazioni senza creare problemi di consistenza delle stesse).

Coordinatore centrale dei lock:

- Non ci saranno dati replicati;
- Ci sara' un unico processo centralizzato responsabile di gestire i lock;
- La realizzazione e' semplice: ogni richiesta di lock sara' composta da una richiesta e dall'attesa di una risposta (2 messaggi);
- La gestione degli stalli e' complicata in quanto i processi si trovano distribuiti su piu' macchine e quindi si tratta di accedere alle informazioni centralizzate per ogni singolo lock;
- Le prestazioni possono diventare scarse;
- La tolleranza ai guasti e' critica per via della centralizzazione.

Coordinatori multipli dei lock:

- Non ci saranno dati replicati;
- Su ciascuna macchina esiste un gestore locale dei lock;
- La realizzazione e' semplice: ogni richiesta di lock sara' composta da una richiesta e dall'attesa di una risposta (2 messaggi);
- La gestione degli stalli e' complicata in quanto bisognera' "intervistare" piu' macchine per capire se vi e' una situazione di stallo o meno.
- Le prestazioni e la tolleranza ai guasti migliorano.

Coordinatore dei lock a maggioranza:

- I dati sono replicati su piu' macchine;
- In ogni macchina esiste un responsabile di lock;
- Lo stesso dato, essendo replicato su piu' macchine, viene gestito da piu' gestori di lock;
- Si otterra' l'acecsso alla risorsa nel momento in cui, su N copia della risorsa e N gestori dei lock, si otterra' N/2 + 1 lock locali;
- La realizzazione e' complessa, si devono usare:
  - 2(N/2 + 1) messaggi per gestire il blocco;

    - (N/2 + 1) messaggi per gestire lo sblocco;
- Il vantaggio e' che non tutti i coordinatori devono essere per forza intervistati per ottenere la risorsa, in quanto basteranno al meglio N/2 + 1 coordinatori intervistati (che concedano l'utilizzo);
- E' necessario modificare gli algoritmi per la gestione dello stallo.

Protocollo polarizzato:

- I dati sono replicati su piu' macchine;
- Su ciascuna macchina esiste un gestore locale dei lock;
- Si ha un modo diverso di trattare le richieste a seconda del fatto che il lock viene richiesto per un accesso:
    - In modo condiviso: sara' sufficiente effettuare una richiesta locale;
    - In modo esclusivo: bisognera' fare una richiesta globale in modo da ottenere l'uso escludendo ad altri processi l'accesso a tale risorsa;
- C'e' un minor sovraccarico nella lettura rispetto ai protocolli di lock a maggioranza;
- C'e' un ulteriore sovraccarico nella scrittura;
- Si avra' la stessa difficolta' per la gestione dello stallo.

Coordinatore

Per la realizzazione del coordinamento in un ambiente distribuito e' necessario introdurre processi di tal scopo. Il compito di questi processi e' quello di coordinare le attivita' tra i vari processi per:

- Realizzare la mutua esclusione;
- Rilevare gli stalli;
- Sostituire i token persi;
- Gestire l'input/output
- ...

Quando un coordinatore non e' piu' raggiungibile o usabile, bisognera' eleggere un nuovo coordinatore in modo tale che le attivita' di coordinamento continuino ad essere garantite. Esistono diversi algoritmi per l'elezione del coordinatore:

- Si puo' scegliere il processo con identificatore di priorita' piu' alta tra quelli disponibili;
- Algoritmo del bullo;
- Algoritmo dell'anello.

Algoritmo del bullo

Un processo cerca di eleggere se stesso come coordinatore.

Quando un processo si accorge che il coordinatore non funziona, manda un messaggio che attiva l'elezione a tutti i processi di priorita' piu' alta. Assume per definizione che i processi a priorita' piu' bassa non abbiano diritto a diventare coordinatori.

- Se entro un tempo massimo T non rieceve alcuna risposta, il processo elegge se stesso come coordinatore e informa tutti di cio'
- Se riceve una risposta vuol dire che c'e' qualche altro processo con una priorita' maggiore che sta cercando di diventare coordinatore:
- Attende l'identificatore del nuovo coordinatore;
- Se non riceve l'identificativo del nuovo coordinatore entro un certo tempo massimo fa ripartire l'elezione.

Algoritmo dell'anello

I processi saranno ordinati ad anello, con la comunicazione tra i processi unidirezionale lungo l'anello. Si considerera' la lista attiva come la lista dei processi attivi che coopereranno all'elezione del coordinatore.

Quando un processo si accorge che il coordinatore non funziona, generera' una nuova lista attiva vuota e invia al processo successivo l'ordine di iniziare l'elezione con la lista attiva.

In questo modo si cerchera' di creare in modo cooperativo la lista dei processi attivi, e in questa lista si potra' identificare il processo con priorita' piu' alta che sara' quello che diventera' coordinatore.

Se il processo riceve un messaggio di elezione:

- Se e' il primo messaggio, crea una lista attiva includendo se stesso e il processo che lo precede nell'anello all'interno della lista e invia il messaggio di elezione al processo successivo;
- Se il messaggio ricevuto non contiene l'identificativo del processo, allora verra' aggiunto il predecessore alla propria lista attiva e verra' inoltrato il messaggio di elezione al processo successivo;
- Se il messaggio ricevuto contiene l'identificativo del processo, la lista attiva e' stata completata e contiene tutti i processi. Il processo potra' quindi guardare nella lista chi e' il coordinatore.

Deadlock in ambiente distribuito

Prevenzione dello stallo

Puo' essere effettuata estendendo le tecniche per la macchina singla:

- Ordinamento del tempo globale delle risorse (sovraccarico minimo);

- Algoritmo del banchiere (sovraccarico elevato, basse prestazioni per via della centralizzazione).

Marche di tempo con rilascio della risorsa

Per ogni processo assegniamo un identificatore di priorita'.

Se un processo P possiede la risorsa e un processo Q ha una priorita' piu' elevata di P, allora P deve rilasciare la risorsa, effettuare un rollback e la risorsa viene assegnata a Q.

Puo' portare alla starvation, ma una soluzione puo' essere ottenuta tramite marche di tempo.

Schema wait-die

- Non si ha rilascio anticipato;
- Se al risorsa e' occupata dal processo P:
  - Se il processo richiedente Q ha marca piu' piccola, attende;
  - Se il processo richiedente Q ha marca piu' grande, fa rollback e muore.

Si evita la starvation se non si assegna una nuova marca al rollback.

Schema wound-wait

- Si ha rilascio anticipato;
- Se la risorsa e' occupata da un processo P:
  - Se il processo richiedente Q ha marca piu' piccola, fa rollback;
  - Se il processo richiedente Q ha marca piu' grande, attende.

Si evita la starvation se non si asssegna una nuova marca al rollback.

Rilevamento dello stallo

Per rilevare lo stallo si utilizzera' il grafo di allocazione delle risorse (grafo di attesa dei processi). Se tra i processi che risultano in attesa di risorse si rileva un comportamento ciclico allora si avra' uno stallo.

Grafo di attesa in ambiente distribuito

Ogni macchina ha un grafo di attesa locale.

Se non ci sono cicli nel grafo di attesa locale, non e' detto che non ci siano cilci a livello di grafo gloale.

Se pero' c'e' un ciclo locale allora si avra' sicuramente una situazione di stallo.

Se l'unione dei grafi delle singole macchine non ha cicli, non si avranno stalli.

Grafo di attesa centralizzato

Si puo' utilizzare un approccio di tipo centralizzato per la realizzazione del grafo di attesa. Si avra' dunque un coordinatore centralizzato del rilevamento degli stalli.

Il grafo di attesa globale sara' l'unione dei grafi di attesa locali. Tuttavia ci sara' un **grafo reale**, quello che il sistema vede in quel momento e il grafo che viene costruito dall'algoritmo di coordinamento centrale.

Questi grafi sono differenti perche' cambia il momento in cui vengono acquisite le informazioni per la costruzione del grafo centralizzato (potrebbe essersi gia' verificata una sitazione di stallo che viene rilevata dall'algoritmo centralizzato al momento della fusione delle porzioni locali dei grafi).

In ogni caso, il grafo costruito dell'algoritmo, dovra' garantire che venga rilevato dallo stallo e che in quel momento il sistema sia effettivamente in stallo.

Si puo' effettuare l'aggiornamento del grafo di attesa localizzato quando si inserisce un arco. In quel caso la macchina manda al coordinatore centrale un messaggio che avvisa della modifica del grafo localizzato.

Questo porta ad un traffico di messaggi, soprattutto nel caso in cui si abbiano cambiamenti frequenti.

Si puo' alternativamente inviare un messaggio solo dopo un certo numero di caricamenti, che riassuma tutti i cambiamenti avvenuti. Il sistema centrale potra' quindi cercare la presenza di cicli nel nuovo grafo unito.

Algoritmo di rilevamento centralizzato

L'algoritmo centralizzato prevede:

- La tramissione del grafi di attesa locali;
- La costruzione del grafo di attesa globale, dove ogni nodo e' un processo e gli archi sono quelli riportati dai grafi di attesa locali;
- Se c'e' un ciclo il sistema e' in stallo, altrimenti potrebbe non esserci (non c'e' garanzia che non sia in stallo in quel momento).

Algoritmo di rilevamento distribuito

In questo caso ogni macchina costruisce una parte locale del grafo di attesa globale.

In ogni grafo locale si inserisce un nodo speciale chiamato Px per indicare che l'attesa coinvolge risorse presenti su altre macchine.

Se esiste uno stallo, esistera' almeno un ciclo nei grafi di attesa locali:

- Se un grafo di attesa locale contiene un ciclo che non coinvogle il nodo Px il sistema e' in stallo;
- Se un grafo di attesa locale contiene un ciclo che coinvolge il nodo Px allora si ha la possibilita' di stallo, in tal caso bisognera' contattare le macchine per verificare l'esistenza dello stallo.

Problema di gestione della rilevazione

Sorge il problema di come rilevare contemporaneamente nei grafi le informazioni e fonderle senza provocare un grosso sovraccarico e ridondanza di messaggi.

Si puo' assegnare un unico identificare ad ogni processo, e quando una macchina scopre un ciclo che coinvolge il nodo Px del prorio grafo locale, questa macchina manda un messaggio di rilevamento dello stallo ad altre macchine solo se il processo precedente a Px nel ciclo ha come identificatore un numero minore del processo successivo a Px.

Altrimenti la macchina non e' in stallo e continua la sua attivita' lasciando il compito di inziare la procedura rilevazione dello stallo ad altre macchine.

Gestione dello stallo

Per gestire lo stallo il coordinatore scegliera' una vittima tra i processi in stallo sulla quale applicare il rollback e rompere lo stallo.

Tutte le macchine vengono informate e chi stava interagendo con la vittima fa rollback a sua volta.

Si possono tuttavia verificare dei *rollback inutili* in cui esistono dei falsi cicli nel grafo glovale dovuti ai tempi di trasmissione dei messaggi di acquisizione e rilascio delle risorse.

Le richieste che provengono da macchine diverse devono essere trattate in modo da evitare questi rollback inutili tramite un'identificazione univoca delle richieste (ad esempio con marcatori temporali unici) e mettendo nel grafo globale solo le richieste non immediatamente soddisfacibili.

Comunicazione tra processi in rete

Nel caso di un ambiente distribuito non si avra' una memoria condivisa, dunque non e' possibile depositare le informazioni da trasmettere tra un processo e l'altro nella memoria condivisa. Saranno i sistemi operativi a trasferire le informazioni dalla macchina mittente alla macchina ricevente, in modo da effettuare la comunicazione.

Scambio di messaggi

Lo scambio di messaggi tra processi in un sistema distribuito avviene in modo analogo ai processi su macchina singola.

La differenza sta nel fatto che i buffer dove devono essere depositati i messaggi non si trovano sempre sulla macchina su cui si ha il processo mittente o ricevente.

Quindi un processo potra' depositare i messaggi in un buffer della propria macchina e il processo destinatario anda' a prelevare il messaggio dalla macchina mittente o viceversa.

Mailbox

Il caso della mailbox e' analogo alla gestione su singola macchina.

Quando un processo vorra' inviare un messaggio per un processo su un'altra macchina, si potra' allocare la mailbox, nella quale depositare i messaggi, sia sulla macchina mittente (deposizione locale, prelievo remoto) che sulla macchina destinataria (deposizione remota, prelievo locale).

File

Il caso dei file e' analogo alla gestione su singola macchina.

Un processo che vorra' inviare dei dati, o un messaggio, attraverso i file, potra' andare a scrivere il messaggio nel file utilizzato come mezzo di comunicazione tra processi, attraverso operazioni di lettura e scrittura.

Questo puo' essere realizzato effettuando le operazioni sul file posto nella macchina remota, oppure, se si sta utilizzando un file system distribuito, montando il file system remoto e accedendoci direttamente.

Questo puo' esere fatto sia che il file sia sulla macchina mittente, sia che sia sulla macchina destinataria, oppure che il file system dove e' depositato il file si trovi su una macchina terza di servizio.

Socket

Le comunicazioni tramite socket avvengono esattamente come nel caso di socket su singola macchina.

La differenza sara' che, invece che usare una stessa porta virtuale locale per lettura e scrittura, si utilizzeranno due porte, una per la macchina mittente (deposizione) e una per la destinataria (prelievo).

Realizzazione

Sara' il sistema operativo a farsi carico della realizzazione dei vari tipi di comunicazione. Sara' responsabile anche di usare meccanismi omogenei tra macchine diverse, cosi' che mittente e ricevente si possano capire.

File system distribuiti

Struttura e funzioni

Obiettivi

Gli obiettivi principali dell'uso di un file system distribuito sono:

- Realizzare l'accesso alle risorse informative e fisiche connesse alle varie macchine di un'architettura di elaborazione distribuita;
- Garantire una trasparenza, tramite una visione globale astratta, durante l'accesso al file system da parte degli utenti e dei processi;
- Gestire in maniera efficiente dell'accesso alle risorse.

Network File System - NFS

L'obiettivo di questo file system e' consentire la condivisione dei file delocalizzati mostrando la struttura della rete.

Si tratta di una collezione di file system delle singole macchine della rete.

E' possibile accedere alle risorse montando i file system remoti su ogni macchina.

Il problema risiede nella visibilita' della struttura della rete e dell'allocazione delle risorse, come limite di semplificazine dell'accesso da parte degli utenti e processi.

Distributed File System - DFS

L'obiettivo di questo file system e' consentire la condivisione dei file delocalizzati in maniera trasparente.

Si ha un'integrazione dei file system delle macchine in rete in un unico file system globale.

Si da una visione omogenea dell'intero file system su tutte le macchine, quindi un'allocazione trasparente delle risorse.

Nomi dei file

SI tratta del problema principale da affrontare per quanto riguarda i file system in ambito distribuito.

Si vuole assegnare un identificatore unico nel file system a ciascuna risorsa (file).

Cio' puo' essere ottenuto in diversi modi:

- In un NFS si potra' utilizzare il nome della macchina piu' il nome del file. Montando localmente il file system di una macchina remota si potra' includere nella macchina locale l'accesso alla risorsa remota utilizzando il percorso che sfrutta il meccanismo di montaggio;
- In un DFS il nome del file sara' unico all'interno del sistema complessivo, senza che si abbia una percezione di dove essos sia. Il nome del file viene mappato automaticamente su un indirizzo locale o remoto.

La locazione dei file puo' essere visibile o invisibile agli utenti (si puo' rendere trasparente la posizione e la locazione).

Un'altra gestione importante nei file system su sistemi distribuiti e' quella delle eventuali repliche dei file, in modo da migliorare prestazioni e tolleranza ai file, cosi come la migrazione dei file. Questi due aspetti avranno un forte impatto sulla trasparenza della locazione rispetto al nome del file.

Accesso ai file

L'accesso ai file puo' essere effettuato in vari modi a seconda del DFS o del NFS:

- NFS:
  - Si potra' accedere ai file usando delle RPC, cosi' da effettuare delle operazioni usuali sui file, ma in remoto;
  - E' possibile copiare localmente i file, manipolarli e quindi salvarli in remoto (previa gestione di mutua esclusione eventuale);
- DFS:
  - Tutto viene visto come un accesso ai servizi del sistema operativo, e sara' il rispettivo servizio a decidere se l'operazione dovra' essere eseguita localmente sulla macchina, oppure remotamente in maniera totalmente trasparente connettendosi alle altre macchine.

Per migliorare la gestione dei file in ambiente distribuito si possono introdurre meccanismi di **caching** che permettono di mantenere copie di file o di porzioni di file remoti, per velocizzare l'accesso.

La locazione della cache puo' essere di varia natura (singola macchina, ecc).

Le politiche di aggiornamento possono essere varie:

- Write-through: ogni volta che viene effettuata un'operazione di scrittura sulla casche viene riportata sulla memoria di massa;

- Delayed-write: viene effettuata l'operazione di scrittura in maniera ritardata, disaccoppiando le operaioni;
- Write-on-close: viene effettuata la scrittura solo alla chiusura del file, per ridurre i tempi di acccesso.

Bisognera' garantire la consistenza del contenuto della cache, la quale potra' essere verificata sia su iniziativa del server che del client.

Un altro passaggio critico sara' la scelta della dimensione (non dovra' essere troppo piccola, altrimenti sarebbe inefficace, e nemmeno occupare troppo spazio).

Stato del file server

Per migliorare la gestione dell'accesso ai file e' possibile introdurre il concetto di stato di un file server, come *insieme delle informazioni che caratterizzano l'uso di un file aperto*.

Nei **file server senza stato**, dove non verra' mantenuta alcuna informazione, ogni richiesta di lettura o scrittura viene soddisfatta in modo indipendente dalle altre (ogni volta si dovra' tradurre il nome simbolico nell'indirizzo fisico ed effettuare le operazioni). E' di semplice implementazione, ma richiede un tempo maggiore (per via della traduzione).

Nei **file server con stato**, con l'operaizone `open()` si definisce lo stato di uso del file, e si riescono a ridurre cosi' i tempi di accesso, in quanto non dovranno piu' essere eseguire le operazioni di reperimento dell'informazione su supporto fisico, avendo gia' le informazioni desiderate accessibili. E' piu' efficiente, ma puo' creare dei problemi di prestazione a causa dell'accesso parallelo alle informazioni.

Replica dei file

La replica dei file puo' essere utile in un sistema distribuito per garantire la ridondanza e quindi una maggiore tolleranza ai guasti, con un relativo aumento delle prestazioni in quanto i processi potranno accedere a copie diverse in parallelo, non andando in conglitto sulla stessa macchina. Inoltre se le copie sono piu' vicine della macchina richiedente si impieghera' meno tempo nel trasferimento.

E' importante pero' che la replicazione rimanga trasparente agli utenti, in modo da rendere semplice ed omogeneo l'accesso alle informazioni.

Inoltre sara' garantito l'aggiornamento delle informazioni, in modo da non avere copie che evolvono in modo diverso nel sistema distribuito.