Comunicazione tra processi

Processi cooperanti

Coordinamento

Un insieme di processi all'interno del calcolatore puo' operare in modo coordinato in modo tale che le varie attivita' avvengano rispettando certi ordinamenti nel tempo.

A tal fine deve essere attivata una **sincronizzazione** della computazione dei processi che garantisca l'accesso a risorse condivise oppure per assicurare un'evoluzione congiunta della computazione dei processi diretta a raggiungere uno scopo applicativo commune.

Cooperazione

La collaborazione tra processi puo' essere piu' intensa fino a spingersi alla cooperazione.

Si parla di collaborazione quando i processi operano in maniera congiunta per il raggiungimento di scopi applicativi comuni condividendo e scambiando informazioni.

Concetto di cooperazione tra processi o thread

Processi indipendenti

Possiamo modellare la nostra computazione dei processi applicativi come processi indipendenti i quali:

- Non hanno scopi comuni con altri processi;
- Non influenzano e non sono influenzati da altri processi;
- Non hanno informazioni condivise;
- In modo mirato tendono a risolvere una specifica applicazione;
- Competono per l'uso del processore con altri processi, ed eventualmente di periferiche condivise.

In questo caso il coordinamento della computazione avra' come obiettivo la sincronizzazione dei processi perche' devono usare le risorse condivise in modo consistente. In particolare quando le periferiche e le risorse condivise possono essere usate coerentemente solo se usate in mutua esclusione.

Processi cooperanti

La computazione puo' essere elaborata anche come processi cooperanti, non piu' come un solo processo che da solo affronta e risolve il problema, ma come un gruppo di processi che operano congiuntamente.

Questi processi cooperanti:

- Hanno un unico scopo (applicazione) comune;
- Possono condividere le informazioni l'un l'altro (**comunicazione**);
- Possono influenzarsi l'un l'altro tramite un coordinamento della computazione (**sincronizzazione**).

Vantaggi della cooperazione

La cooperazione rispetto alla risoluzione di un proble a applicativo con un unico processo ha dei vantaggi perche' permette di scomporre il problema applicativo in parti e di affrontare ciascuna parte con un processo mirato. Questo consente di avere:

- Modularita';
- Parallelizzazione;
- Scalabilita';
- Specializzazione;
- Qualita' del progetto e della realizzazione.

Esempi di processi cooperanti sono i sistemi produttore-consumatore come:

- Client-Server;
- Compilatore-Assemblatore-Linker-Loader.

Comunicazione tra processi

Concetto di comunicazione

La comunicazione tra processi e' l'insieme dei meccanismi e delle politiche che permettono a due processi di scambiare informazioni per operare in modo cooperativo.

L'invio e la ricezione di informazioni tra i processi consente di coordinare l'evoluzione della computazione.

Necessita'

Si ha la necessita' di realizzare una comunicazione quando si vuole trasferire informazioni dal mittente al ricevente oppure quando si vuole far condividere informazioni a due o piu' processi.

Entita' coinvolte

La comunicazione vede coinvolte diverse entita':

- Un processo produttore dell'informazione;
- L'informazione che viene trasferita;
- Un processo ricevente, utilizzatore dell'informazione;
- L'insieme delle attivita' di trasferimento dell'informazione (dal mittente al ricevente e dei meccanismi coinvolti) costituiscono il **canale di comunicazione** a livello delle procedure di sistema.

Il canale puo' essere:

- Mono-direzionale;
- Bi-direzionale.

Caratteristiche

Le comunicazioni sono caratterizzate da vari aspetti:

- La quantita' di informazioni che si vuole trasmettere:
  - In modo da poter identificare quale canale di comunicazione e' piu' utile utilizzare;
- La velocita' di esecuzione:
  - Definisce quanto devono essere complesse le operazioni di comunicazione, anche qui per poter scegliere accuratamente il canale di comunicazione;
- Scalabilita':
  - Quando si vuole poter aumentare il numero di comunicazioni da gestire, quindi il numero di processi da gestire in comunicazione;
- Semplicita' di uso nelle applicazioni;
- Omogeneita' delle comunicazioni;
- Integrazione nel linguaggio di programmazione;
- Affidabilita';
- Sicurezza;
- Protezione.

Implementazione

Esistono varie tecniche per implementare la comunicazione tra processi da utilizzare a seconda del canale di comunicazione scelto:

- Comunicazione diretta: dove ogni processo conosce l'altro (memoria condivisa, messaggi);
- Comunicazione indiretta: dove non si conosce il processo destinatario ma si ha una struttura dati in cui vengono depositate le informazioni da scambiare (mailbox, file/pipe, socket).

Memoria condivisa

E' la tecnica piu' semplice e consiste nel far condividere a due processi una zona di memoria condivisa (possibilmente nello spazio di indirizzamento dei processi). Dove il processo P_1 puo' scrivere e il processo P_2 puo' leggere.

Messaggi

Due processi P_1 e P_2 possono comunicare attraverso delle funzioni di sistema operativo appoggiandosi a dei blocchi di memoria posti all'interno del sistema operativo (**buffer**).

Il processo mittente depone il suo messaggio in un buffer e il processo ricevente lo estrae.

Mailbox

Due processi P_1 e P_2 comunicano sfruttando il sistema operativo depositando i propri messaggi in una struttura apposita chiama Mailbox.

Il processo ricevente acquisisce il messaggio ricevuto prendendolo dalla Mailbox. In questo modo il mittente non deve per forza conoscere il ricevente.

File e pipe

E' possibile estendere il concetto di memoria condivisa passando dalla memoria centrale alla memoria di massa. Si puo' avere la realizzazione della comunicazione di due processi P_1 e P_2 attraverso i supporti del sistema operativo ponendo le informazioni su un **file** sul disco.

Una variante di questa architettura prevede che la comunicazione avviene attraverso buffer ordinati sequenzialmente posti in memoria centrale del sistema operativo (**pipe**).

Socket

Un altro approccio che generalizza la tecnica delle pipe per i sistemi distribuiti: due processi P_1 e P_2 che possono risiedere anche su macchine diverse, con due sistemi operativi diversi tra di loro.

Il processo P_1 deposita le informazioni in un estremo della pipe e il processo P_2 lo estrae dall'estremo opposto sull'altra macchina.

La comunicazione viene garantita tra le due macchine e quindi tra i due estremi della pipe dalla rete, che in modo trasparente prende i messaggi da un estremo e li pone sull'altro estremo. Questo e' il modello di comunicazione dei **socket**.

Tale struttura funziona sia in ambiente distribuito sia su macchina singola.

Comunicazione con memoria condivisa

Condivisione di parte dei dati in memoria centrale

Consiste nell'avere un processo P_1 con un suo spazio di indirizzamento in cui vengono posti:

- Il codice del programma da eseguire;
- I dati globali che non vengono condivisi;
- Le variabili globali che si vogliono condividere;
- Heap;
- Stack.

Questo spazio di indirizzamento e' separato da quello del processo P_2 che vuole ricevere le informazioni, che analogamente avra' il suo spazio di indirizzamento.

Dunque il processo P_1 potra' scrivere le informazioni che vuole condividere nello strato di memoria locale che ospita le variabili globali. Poiche' la memoria e' condivisa, le variabili scritte in quello strato vengono lette dal processo P_2.

Problemi della condivisione di variabili globali

E' necessario:

- Identificare i processi comunicanti (comunicazione diretta);
- Garantire la consistenza delle operationi (lettura e scrittura sono incompatibili tra di loro e quindi vanno eseguire in mutua esclusione);
- Sincronizzare l'accesso in mutua esclusione all'area dati comune ai due processi.

Realizzazione con area comune copiata dal sistema operativo

Il sistema operativo avra' uno spazio di memoria che contiene una **copia** dello strato di memoria condivisa tra i processi.

Quando il processo P_1 vuole avere accesso alla memoria per scrivere le informazioni da condividere compete per avere l'accesso mutuamente esclusivo ed effettua un'operazione di scrittura nella sua memoria locale.

Alla terminazione della scrittura rilascera' l'uso mutuamente esclusivo della zona di memoria di modo che il sistema operativo puo' prendere i valori contenuti nella zona di memoria locale al processo e copiarli nello spazio riservato al suo interno.

Quando il processo P_2 chiedera' di accedere alla zona di memoria condivisa, il sistema operativo provvedera' a copiare i dati del suo spazio di memoria nello spazio di memoria locale del processo.

Questo comporta una doppia operazione di copiatura dei dati, e se la porzione di dati da copiare e' molto grande puo' portare a rallentamenti per la gestione della condivisione delle informazioni. Ancora peggio accade quando si vanno a modificare poche informazioni nello spazio di indirizzamento comune, in quanto molte delle informazioni che sono state copiate non sono effettivamente state modificate.

Realizzazione con area comune fisicamente condivisa

Se l'HW del sistema supporta la condivisione di memoria tra processi, ossia permette di rilasciare i vincoli su alcune porzioni di memoria per il possesso in modo univoco per quella porzione di memoria a un processo, possiamo creare un meccanismo di condivisione molto semplice.

Al di fuori dello spazio di indirizzamento dei processi e del sistema operativo, esistera' una zona di memoria usata in modo mutuamente esclusivo per condividere le informazioni comuni. Entrambi i processi vedranno questa zona di memoria come una loro zona di memoria locale, logicamente lo spazio di indirizzamento dei processi viene esteso per inglobare anche questa zona di memoria.

L'accesso alla zona di memoria condivisa richiede comunque la mutua esclusione, ma evita la doppia copiatura delle informazioni.

Condivisione di un'area di memoria centrale per le comunicazioni

Un altro modo di gestire la cooperazione evitando di trasferire le informazioni e' quella che utilizza i buffer. In questo caso il mittente e il ricevente non si scambiano tutto l'insieme di variabili, ma solo il delta che si vuole trasferire.

Il processo P_1 depositera' le variabili che vuole inviare al processo P_2 in un'area di memoria condivisa tra i due processi, ma quello che viene trasferito e' solo la porzione di informazione che viene appositamente costruita dal processo mittente per il ricevente.

Problemi:

- Anche in questo caso e' necessario identificare i due processi comunicanti (comunicazione diretta);
- Anche in questo caso e' necessario garantire la consistenza degli accessi;
- Anche in questo caso e' necessario sincronizzare l'accesso in mutua esclusione.

La realizzazione puo' essere effettuata:

- Tramite dei buffer che vengono copiati dal sistema operativo;
- Tramite dei buffer che risiedono ina zona di memoria fisicamente condivisa.

Esempio produttore-consumatore

Le comunicazioni avvengonno utilizzando un buffer condiviso, con una capacita' limitata o illimitata.

Le attivita' della comunicazione sono:

- Il produttore genera le informazioni elementari per il consumatore;
- Il produttore memorizza ciascuna informazione elementare nel buffer di comunicazione condiviso;
- Il consumatore acquisisce ciascuna informazione elementare prendendola dal buffer di comunicazione codiviso e rimuovendola da questo;
- Il consumatore legge ed usa ciascuna informazione elementare generata dal produttore.

Comunicazione con scambio di messaggi

Modello della comunicazione a messaggi

L'obiettivo e' quello di innalzare l'astrazione rispetto al modello di condivisione di memoria per modellare soltanto il trasferimento di informazioni tra due processi.

Un processo P_1 gestisce l'invio di messaggi al processo P_2 attraverso il supporto di alcune primitive del sistema operativo.

Caratteristiche dei messaggi

I messaggi contengono un insieme di informazioni atte a trasferire i dati desiderati:

- Identificativo del processo mittente;
- Identificativo del processo destinatario;
- Informazioni da trasmettere;
- Eventuali altre informazioni di gestione per garantire lo scambio di messaggi.

I messaggi possono avere una dimensione fissa o variabile.

Il sistema operativo e' responsabile della memorizzazione delle informazioni dei messaggi in spazi di memoria temporanea, in buffer dedicati alla comunicazione. I buffer possono essere assegnati a ciascuna coppia di processi oppure possono essere di uso generale ed assegnati di volta in volta alla coppia di processi che ne fa richiesta attivando una procedura di comunicazione.

Il numero dei buffer che vengono assegnati a ciascun processo puo' essere illimitato, limitato o nullo.

Funzioni

Per gestire le comunicazioni il sistema operativo mette a disposizione un insieme di primitive.

Invio

L'invio avviene attraverso la primitiva `send(recipient, message)`:

- La primitiva deposita il messaggio in un buffer libero del sistema operativo;
- La primitiva e' bloccante nel caso in cui non ci siano buffer liberi per completare la comunicazione.

Ricezione

Il messaggio puo' essere ricevuto attraverso la funzione `receive(sender, message)`:

- La primitiva riceve il messaggio da un buffer;
- La primitiva e' bloccante fintanto che non c'e' un messaggio ricevibile.

Invio condizionale

L'invio condizionale tramite `cond_send(recipient, message): error_status`:

- Deposita il messaggio in un buffer libero del sistema operativo;
- E' una primitiva non bloccante in quanto se non ci sono buffer liberi per poter effettuare la comunicazione col destinatario, ritorna una condizione di errore non bloccando il mittente e non depositando piu' il messaggio.

Ricezione condizionale

La ricezione condizionale tramite `cond_receive(sender, message): error_status`:

- Riceve il messaggio da un buffer;
- E' una primitiva non bloccante in quanto se non ci sono messaggi ricevibili segnala una condizione di errore e non si pone in attesa di alcun messaggio.

Sincronizzazione dei processi comunicanti

Quando i processi accedono alle operazioni di comunicazione mediante scambio di messaggi, le comunicazioni possono essere sincronizzate in vari modi.

Comunicazioni asincrone

Bloccanti solo se l'operazione non puo' essere completata.

Comunicazioni sincrone

La comunicazione avviene solo con la presenza contemporanea dei due processi. Il processo mittente aspetta sempre che il processo destinatario esegua la funzione di ricezione.

Identificazione dei processi comunicanti

I processi comunicanti devono identificarsi nella comunicazione con messaggi a seconda del tipo di comunicazione.

Comunicazione simmetrica

Mittente e destinatario sono sempre univocamente identificati (comunicazione diretta).

Comunicazione asimmetrica

Il mittente o il destinatario possono non essere identificati univocamente.

Se il destinatario non e' specificato univocamente allora puo' essere il processo di un gruppo (specificato) oppure un processo qualunque.

Se il mittente non e' specificato univocamente allora puo' essere il processo di un gruppo (specificato) oppure un processo qualunque.

Caratteristiche e problemi

Nella comunicazione mediante l'utilizzo di messaggi:

- E' necessaria l'identificazione dei processi comunicanti;
- Non si ha piu' la memoria condivisa tra i processi;
- Per garantire la consistenza del messaggio inviato e' necessaria la sincronizzazione per l'accesso ai messaggi stessi. La sincronizzazione viene gestita in maniera implicita dal sistema operativo.

Comunicazione con mailbox

Modello della comunicazione a messaggi con mailbox

Il modello prevede che un processo P_1 mittente voglia scambiare un messaggio contenente informazioni attraverso una struttura del sistema operativo chiamata mailbox (o porta). Questa struttura contiente dei buffer in cui i messaggi possono essere accumulati

Quando un processo P_2 desidera ricevere un messaggio dalla mailbox, vi accede attraverso una funzione di sistema operativo ed estrae il messaggio.

Caratteristiche dei messaggi

I messaggi conterranno le informazioni:

- Processo mittente;
- Mailbox destinataria;
- Informazioni da trasmettere;
- Eventuali altre informazioni a supporoto della gestione dei messaggi nella mailbox.

I messaggi possono essere di dimensione fissa o variabile.

Mailbox

La mailbox potra' avere una capacita':

- Illimitata: un numero illimitato di messaggi puo' essere depositato;
- Limitata: un numero finito di messaggi puo' essere depositato;
- Nulla: nessun messaggio puo' essere depositato.

Funzioni

Il sistema operativo mette a disposizione delle funzioni per la gestione delle mailbox:

- Creazione mailbox: `create(M)`;
- Cancellazione mailbox: `delete(M)`.

Invio

La funzione di invio `send(M, message)` deposita un messaggio nella mailbox ed e':

- Non bloccante se la capacita' della mailbox e' illimitata;
- Bloccante se la mailbox e' piena;
- Bloccante se non c'e' un processo in ricezione (mailbox con capacita' nulla).

Ricezione

La funzione di ricezione `receive(M, message)` riceve un messaggio dalla mailbox e sara' sempre bloccante se non c'e' almeno un messaggio da ricevere.

Invio condizionale

La funzione di invio condizionale `cond_send(M, message): error_status` deposita un messaggio nella mailbox se l'operazione puo' essere completata (in funzione della capacita' della mailbox). Se l'invio bloccasse il mittente, ritorna una condizione di errore non bloccando il mittente e non depositando il messaggio.

Ricezione condizionale

La funzione di ricezione condizionale `cond_receive(M, message): error_status` riceve un messaggio dalla mailbox se c'e' almeno un messaggio. Se non ci sono messaggi ricevibili, ritorna una condizione di errore non bloccando il destinatario e non ricevendo il messaggio.

Sincronizzazione dei processi comunicanti

I processi che stanno comunicando vengono sincronizzati a seconda della capacita' della mailbox:

- Mailbox con capacita' illimitata: comunicazione asincrona;
- Mailbox con capacita' nulla: comunicazione sincrona;
- Mailbox con capacita' limitata: comunicazione bufferizzata.

Caratteristiche e problemi

Nel caso della comunicazione basata sulla mailbox:

- Non serve identificare i processi comunicanti (comunicazione indiretta);
- Non e' necessaria una memoria condivisa tra i processi;
- Si ha sincronizzazione per accedere ai messaggi all'interno della mailbox, ma questo e' gestito implicitamente dal sistema operativo.

Ordinamento delle code dei messaggi e dei processi in attesa

I messaggi in attesa della coda nella mailbox o i processi in attesa di ricevere dei messaggi quando la mailbox e' vuota possono essere ordinati mediante opportune politiche di attesa/ordinamento:

- First In, First Out;
- Priorita';
- Scadenza.

Proprieta' della mailbox

La proprieta' della mailbox puo':

- Essere del sistema operativo: dunque non essere correlata ad alcun processo;
- Essere specifica di un processo:
  - Solo il processo proprietario riceve da questa mailbox;
  - Altri processi possono solo inviare;
  - Se termina il processo proprietario scompare la mailbox.

Comunicazione con molti possibili mittenti o riceventi

Le mailbox permettono di realizzare comunicazioni con piu' processi mittenti o piu' processi riceventi. In ogni caso la comunicazione coinvolge sempre lo scambio di messaggi da un processo mittente ad un processo ricevente (anche se puo' non essere identificato quale tra i processi mittenti e' quello che invia il messaggio, o quale tra i processi riceventi e' quello che riceve il messaggio).

Comunicazioni molti a uno

La mailbox viene vista da piu' processi mittenti che vogliono inviare un messaggio alla mailbox. Ognuno di questi processi usera' la funzione di invio e se la mailbox ha spazio sufficiente per contenere tutti i messaggi potra' procedere con la sua computazione.

La mailbox avra' un solo processo ricevente e se questo processo usera' la funzione di ricezione, prima dell'invio dei messaggi da parte degli altri processi, si porra' in attesa.

Quando il messaggio arriva, il ricevente lo estrarra' dalla coda e si procede con la computazione.

Comunicazioni uno a molti

La mailbox viene letta da piu' processi che si porranno in attesa dell'arrivo dei messaggi, che verranno ordinati secondo un'opportuna politica.

Quando l'unico processo mittente inviera' un messaggio alla mailbox, questo verra' consegnato al primo processo in attesa, secondo la politica definita, mentre gli altri processi rimarranno in attesa.

Comunicazioni molti a molti

Quando ci sono piu' mittenti, i messaggi in attesa vengono scodati secondo l'ordine definito dalla politica da piu' processi riceventi, anche loro ordinati secondo la politica.

Comunicazione con file

L'obiettivo e' quello di astrarre la condivisione tramite memoria condivisa portando le informazioni su memoria di massa.

Modello della comunicazione mediante file condivisi

Nelle comunicazioni attraverso file condivisi si fa uso di particolari strutture dati che risiedono sul disco gestite dal sistema operativo.

Modello della comunicazione mediante pipe

Le pipe sono essenzialmente un file posto in memoria centrale. Il meccanismo basato sulla pipe prevede che le informazioni siano inviate in un modo strettamente sequenziale secondo la politica First In, First Out.

In questo file la scrittura avviene solo in modalita' `append` e la lettura e' sequenziale.

Caratteristiche dei messaggi

I messaggi sono composti da:

- Processo mittente;
- Informazioni da trasmettere;
- Eventuali altre informazioni a supporto della gestione dei messaggi.

Possono essere di dimensione fissa o variabile.

Funzioni

Le funzioni messe a disposizione dal sistema operativosono:

- Creazione;
- Cancellazione;
- Lettura;
- Scrittura.

Queste funzioni coincidono con quelle di gestione dei file sul FileSystem.

Caratteristiche e problemi

Le funzioni di comunicazione attraverso file o pipe permettono di sincronizzare i processi in lettura e scrittura secondo le politiche e i meccanismi previsti dal FileSystem.

I messaggi possono essere ordinati a seconda del tipo di approccio utilizzato:

- Nel file: dipende dal processo scrivente;
- Nella pipe: First In, First Out.

L'ordinamento dei processi in attesa di lettura:

- Nel file: dipende dalla gestione del FileSystem;
- Nella pipe: la comunicazione e' diretta con un unico processo ricevente.

Comunicazione con socket

L'obiettivo e' quello di generalizzare il modello di comunicazione basato sulle pipe in memoria centrale in cui la comunicazione avveniva attraverso un canale posto in memoria centrale gestito come un file in cui si scrivevano messaggi ordinatamente.

Modello della comunicazione mediante socket

Abbiamo due macchine in un ambiente distribuito con due sistemi operativi (eventualmente diversi tra loro) che supportano le operazioni dei processi applicativi.

Il processo mittente sulla prima macchina scrivera' i messaggi in un estremo della *pipe condivisa* con la seconda macchina, dove risiede il processo ricevente.

Il trasferimento dei messaggi da un estremo della pipe all'altra e' assicurato dalla rete informatica che connette le due macchine.

Architettura

La comunicazione avviene attraverso un'architettura client-server, dove:

- Il client invia richieste (messaggi) ad una porta specifica;
- Il servere riceve le richieste (messaggi) mettendosi in ascolto su una porta specifica.

Il canale di comunicazione e' dunque la **porta** ed e' individuato dall'indirizzo IP della macchina che lo ospita e il numero di porta.

Caratteristiche dei messaggi

I messaggi contengono solamente il contenuto dell'informazione da trasmettere e possono essere di dimensione fissa o variabile.

Funzioni

Le funzioni per i socket sono specifiche dei protocolli di comunicazione in rete che si vanno a considerare per il sistema operativo:

- Creazione;
- Cancellazione;
- Lettura;
- Scrittura.

Caratteristiche e problemi

L'ordinamento dei messaggi all'interno di ciascuna coda, cosi' come l'ordinamento dei processi in attesa e' strettamente First In, First Out.

La connessione puo' essere:

- Gestita: ogni sequenza di messaggi viene controllata anche nel caso di problemi di comunicazione in rete;
- Non gestita: ogni messaggio puo' essere scambiato separatamente dagli altri;
- Con meccanismi di multicast: dove un messaggio viene inviato a piu' processi.