Analisi dei risultati

Prove su ricette semplici

Il che seguiamo prima di giungere all'effettiva simulazione è quello di verificare funzionamento del modello, prendendo come spunto una semplificazione della realtà che simuleremo con il programma jVEFrame.

Questa semplificazione consiste nello sperimentare il modello prendendo in considerazione schemi che riproducono il più fedelmente possibile la formalizzazione che è stata fatta delle sequenze produttive ed egli schemi di approvvigionamento materiali e componenti per la ricerca che sta alla base di questo lavoro.

Di seguito vengono presentate le tabelle riportanti:

1. esemplificazione di ricette

```
# comment ;
# comment ;
    alfa  100 1 s 2  2 s 3 3 s 4 4 s 3 5 s 1 e 100 ;
    beta  200 6 s 1  2 s 1 7 s 2 8 s 3 4 s 2 e 200 ;
    teta  300 p 1 100 9 s 10                       ;
# comment ;
    sigma 400 p 1 200 10 s 1                       ;
```

2. esemplificazione dello schema di approvvigionamento materiali e componenti

```
1  100 * 500 200 * 500                          ;
2  100 * 500 200 * 500                          ;
3  100 * 500 200 * 500                          ;
4  100 * 500 200 * 500                          ;
5  100 * 500 200 * 500                          ;
6                        300 * 200 400 * 300    ;
7  100 * 500 200 * 500                          ;
8  100 * 500 200 * 500 300 * 3000 400 * 1000    ;
9                                               ;
```

10	100	*	500	200	*	500							;
11	100	*	500	200	*	500	300	*	1200	400	*	1200	;
12													;
13	100	*	500	200	*	500							;
14	100	*	500	200	*	500							;
15							300	*	1100	400	*	300	;

Impostiamo la simulazione con i seguenti parametri:

useOrderDistiller true

attiviamo l'utilizzo delle classi orderDistiller.java è Recipes.java.;

ticksInATimeUnit 28800

consideriamo un numero di passi che possa essere almeno rappresentativo dell'ordine di grandezza che vorremmo osservare in un turno di lavoro dell'azienda presa in considerazione. Se volessimo osservare la realtà completamente dovremmo avere a disposizione degli strumenti in grado di calcolare 28800 tick in ogni ciclo, rappresentativi dei secondi di un turno di otto ore;

totalUnitNumber 97

il numero di unità lavorative in grado di effettuare le lavorazioni necessarie per completare le ricette di passi produttive;

totalEndUnitNumber 49

il numero di unità di transito che contengono ciò che, nell'applicazione ad un'azienda del settore meccanico, si considerano semilavorati o componenti;

maxStepNumber 30

non va considerato nella simulazione in questione;

maxStepLength 0

non va considerato nella simulazione in questione;

useWarehouses false

la gestione del magazzino e non è simulata, infatti ogni unità lavora ciò che le viene assegnato da un incisore centrale e non vi è creazione di scorte presso le singole unità, la gestione di componenti semilavorati è fatta con la metodologia dei procurement;

useNewses false

la trasmissione di informazioni fra le unità non avviene, come conseguenza della mancanza di magazzini;

maxInWarehouses 10

non va considerato nella simulazione in questione;

minInWarehouses 3

non va considerato nella simulazione in questione;

infDeepness 30

non va considerato nella simulazione in questione; pt

inventoryFinancialRate 0.05

determina il tasso al quale la permanenza delle scorte magazzino senza che siano utilizzate produce un costo;

inventoryEvaluationCriterion 1

abbiamo stabilito di attribuire unicamente i costi variabili di competenza;

revenuePerEachRecipeStep 3

il profitto da attribuire ad ogni passo della ricetta;

nOfNewsesToProduce 1

non va considerato nella simulazione in questione;

nOfNewsesToBeCleared 100000

non va considerato nella simulazione in questione;

nOfOrdersInNewses 1

non va considerato nella simulazione in questione;

orCriterion 0

la scelta effettuata indica di considerare tutti i percorsi alternativi possibili che si presentano nella ricetta

Breve descrizione dei risultati

Prendendo in considerazione il parametro ticksInTimeUnit pari a 28800 il risultato che si ottiene è quello di occupare tutta la memoria e di bloccare per intero l'unità di CPU del computer, in grado di lavorare su una frequenza di 1,34 GigaHertz.

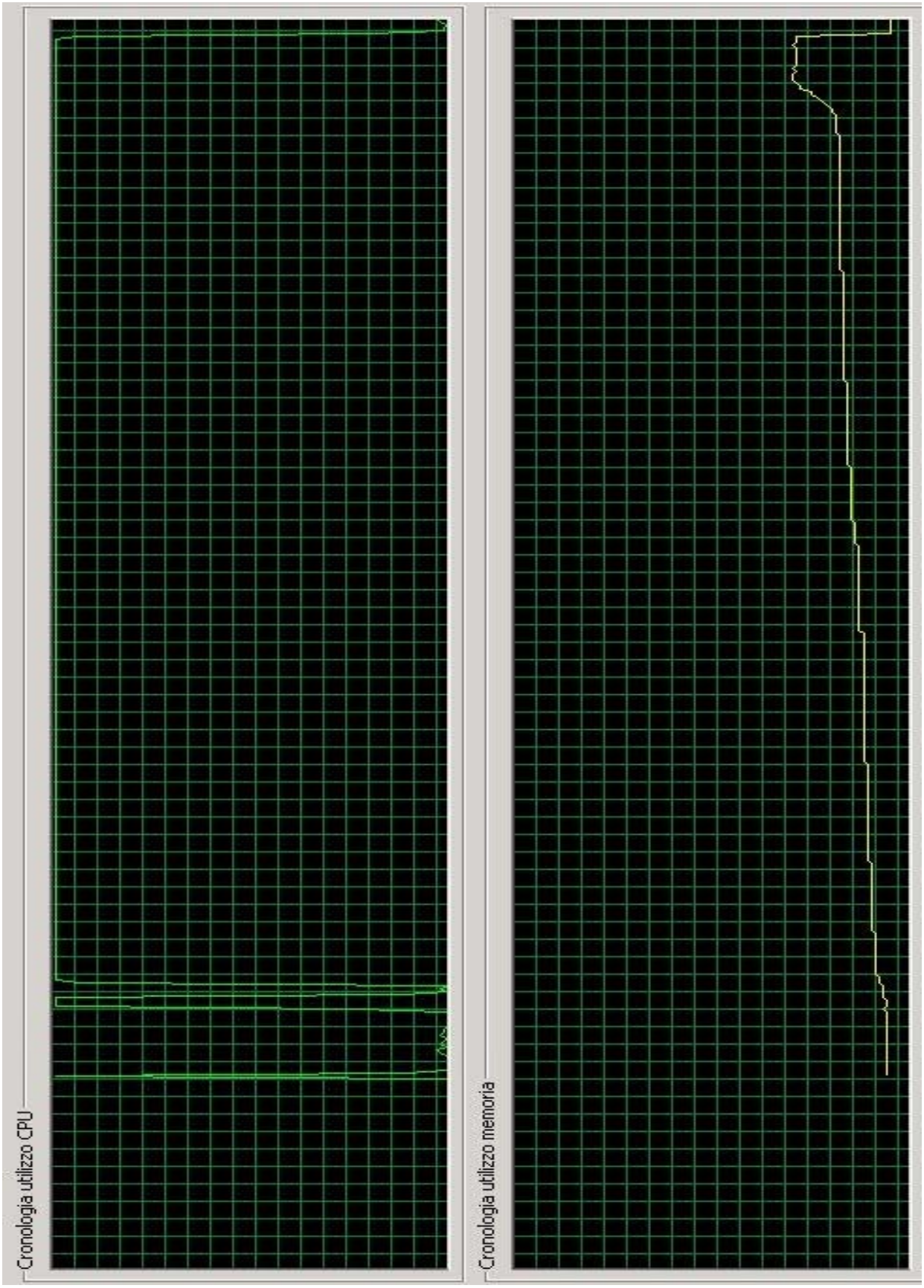


Figura 1: Utilizzo delle risorse del computer con cicli di 28800 tick

Descrizione del problema

Il tempo richiesto da una simulazione esemplificativa, come quella posta in essere sino questo momento, è decisamente elevato e non permetterebbe assolutamente di effettuare una valutazione reale dell'impresa che interessa in tempi utili, ma soprattutto sarebbe logorante l'attesa per effettuare il confronto fra simulazione con parametri differenti, che richiederebbero l'esecuzione ripetuta del programma. Il problema principale va ricollegato anzitutto alla notevole quantità di azioni che un agente in una simulazione reale si trova a dover compiere, ma anche ad un più generale problema di gestione delle risorse a disposizione e della scelta delle tecniche di ricerca o di calcolo: ogni agente ed ogni oggetto si mette in relazione con il resto dell'ambiente e con gli altri agenti mediante una tecnica di ricerca di tipo lineare ossia, per esemplificare, quando un oggetto necessita di informazioni o messaggi relativi ad altri oggetti o all'ambiente, invia dei messaggi alle liste di oggetti, scorrendole in maniera sequenziale sino a trovare ciò di cui ha bisogno. Lo sviluppo progressivo delle funzionalità degli agenti e della complessità dell'ambiente pongono in essere problemi di tipo tecnico-tecnologico.

La strategia di ricerca adottata è ben strutturata in quanto è in grado di rispondere al requisito di completezza, ossia è possibile trovare una soluzione quando esiste. Limiti di carattere temporale e spaziale impongono, d'altro canto, di tenere in considerazione altre tre caratteristiche (Russel, Norvig; 1995) che ben si adattano all'impostazione dei problemi di ricerca:

1. Complessità temporale, ossia quanto tempo occorre per trovare una determinata soluzione:
2. Complessità spaziale, ossia quanta memoria occorre per effettuare la ricerca;
3. Ottimalità, la strategia deve trovare la soluzione di qualità massima quando ci sono varie soluzioni differenti;

Nel nostro ambito, per ciò che attiene all'ultima caratteristica, la scelta che viene fatta a riguardo della soluzione è che essa deve semplicemente rispondere ai

requisiti richiesti (se mi occorre una lavorazione del tipo 5 andrà bene qualunque unità in grado di effettuare tale lavorazione, indipendentemente dalle code).

Diversa è la questione relativa alle altre due caratteristiche, infatti potrebbe rivelarsi utile in futuro effettuare un confronto fra algoritmi per verificare quale sia più veloce o richieda meno memoria.

Esistono in particolare due modi possibili:

1. il primo consiste nell'usare dei benchmark, ossia dei parametri di confronto in grado di stabilire, dopo aver eseguito i due algoritmi, quale sia il più celere o quale richieda meno memoria. Proprio in questo calcolo si comprendono anche caratteristiche specifiche della macchina e della piattaforma presente; alternativa può essere il conteggio effettivo del numero di operazioni che vengono eseguite;
2. il secondo metodo può essere un'analisi degli algoritmi di tipo matematico, indipendente dalla particolare implementazione. Tale analisi tiene conto principalmente della possibilità di confrontare un numero elevato di volte, tendente all'infinito, i passi compiuti dagli algoritmi ed a quel punto effettuare una analisi asintotica; resta comunque il problema non banale di definire la lunghezza dell'algoritmo.

Altro problema può nascondersi nella crescita esponenziale delle interrelazioni fra gli agenti e perciò nella complessità sottostante e difficilmente prevedibile.

Adattamento del problema

Per porre in essere una simulazione funzionante ed in grado di fornire dati utili in tempi ragionevoli occorre perciò trovare una soluzione al problema della potenzialità dello strumento. Tale soluzione può seguire due strade opposte:

1. nel primo caso è possibile adottare una convenzione ed effettuare prove che considerino una granulosità del tempo differente da quella

dei secondi, consderandone decine o centinaia per ogni tick, valutando però la perdita di dati che si ha in questo modo;

2. la seconda soluzione attiene invece alla possibilità di sviluppare le potenzialità dei calcolatori seguendo la strada dei super calcolatori o cluster.

La prima soluzione consiste nel considerare un numero inferiore di tick all'interno del singolo ciclo di simulazione stabilendo, per convenzione, che ogni istante sia rappresentativo di tanti tick quanto è il denominatore dell'operazione di riduzione effettuata.

Allo stesso tempo occorrerà ridurre proporzionalmente il numero di ordini in programmazione, mentre non sarà necessario effettuare modifiche, salvo che per gli ordini di tipo *stand alone batch,* le cui quantità saranno anch'esse ridotte di conseguenza.

Di seguito sono riportati i risultati di alcune prove:

ticksInATimeUnit 2880;
quantità di secondi occorrenti per ottenere una determinata fornitura di ricette
batch stand alone 1/10;
numero di lavorazioni interne o richieste di forniture di ricette del tipo *sequential batch* 1/10.

Purtroppo anche in questo caso il risultato non è stato utilizzabile; la situazione è simile alla precedente, infatti, anche se i grafici vengono disegnati, lo scorrere del tempo è troppo lento per giungere a risultati utili.

Figura 2: Utilizzo delle risorse del computer con cicli da 2880 tick

Per confronto si può osservare la figura 2, dalla quale si osserva che il problema sia dovuto alla limitata capacità di calcolo del computer, problema

superabile con l'adozione di computer più potenti o con lo sviluppo di nuove tecnologie.

Problemi di hardware ed evoluzione tecnologica

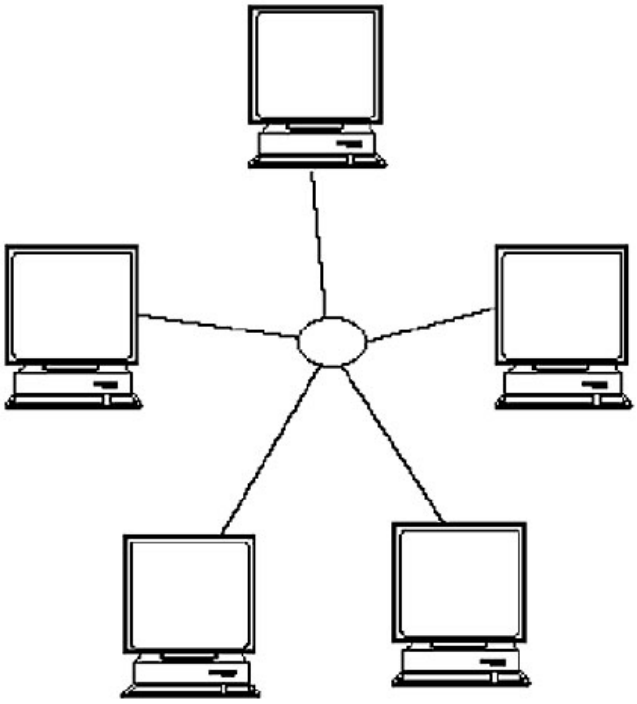
Nel corso degli ultimi anni si è verificato uno sviluppo costante nelle prestazioni offerte dai personal computer tale da fare sì che un moderno PC possa essere tranquillamente assimilato ad una workstation, una stazione di lavoro dotata di sistemi con buona qualità di grafica, con video di più elevate prestazioni, con CPU di media potenza, spesso attrezzata con unità opzionali orientate alla grafica, quali coprocessori specializzati, scanner e tavolette grafiche e destinata ad attività di studio, professionali o di sviluppo di programmi. L'incremento di prestazioni è senz'altro dovuto allo sviluppo di componenti hardware sempre più sofisticati ed in grado di operare ad altissime velocità. In particolare per quanto riguarda i PC questo incremento di prestazioni è avvenuto senza compromettere la tradizionale economicità di queste macchine, che vantano dunque un sorprendente rapporto prestazioni/costo.

Anche l'hardware di rete si è sviluppato con analoga rapidità ed economicità, permettendo il diffondersi di reti locali veloci ed affidabili. In particolare grande diffusione hanno conosciuto in questi ultimi anni gli switch, ad esempio Fast Ethernet, che permettono di realizzare reti di interconnessione fra PC e workstation analoghe a quelle dei multiprocessori.

Parallelamente allo sviluppo dell'hardware, c'è stato quello del software: ad esempio, per i moderni PC sono oggi disponibili sistemi operativi dotati di tutte quelle caratteristiche che per anni sono state disponibili esclusivamente per calcolatori di livello superiore (multitasking, multithreading), oltre che dei servizi di rete standard basati sul TCP, UDP e IP. Tali sistemi sono sia commerciali (ad es. Windows 95, Windows NT) che liberamente distribuiti come Linux, una versione di Unix per PC particolarmente interessante perché nella distribuzione sono compresi i sorgenti, modificabili secondo i termini della licenza GNU .

Tutto ciò ha contribuito a determinare l'incredibile diffusione dei PC in ogni ambiente, facendone dunque una realtà di cui non si può non tener conto.

Ben diversa è stata la situazione per quanto riguarda i calcolatori paralleli, siano essi multicalcolatori o multiprocessori . La loro diffusione è stata piuttosto modesta, e i loro costi rimangono tutt'ora elevatissimi, sopportabili solo da un numero piuttosto limitato di enti specializzati. In particolare, ben di rado multicalcolatori sono disponibili nei centri di calcolo universitari, sia per il loro costo che per la necessità di personale specializzato alla loro manutenzione. Inoltre il mercato piuttosto limitato di questi sistemi di calcolo ne rende antieconomico un aggiornamento frequente.



Super cluster

In altri termini spesso essi usufruiscono delle innovazioni nel campo della componentistica con un notevole ritardo.

Proprio per questi motivi è in atto già da alcuni anni in diverse Università e centri di calcolo la tendenza a sviluppare sistemi alternativi ai calcolatori paralleli ``convenzionali'', e particolare sviluppo in questo senso hanno avuto i concetti di rete di workstation (Network Of workstation, NOW) e di cluster di PC (analogo alle NOW ma con PC invece di workstation). L'idea è quella di utilizzare le dotazioni normalmente presenti nei centri di calcolo o comunque acquistabili a prezzi

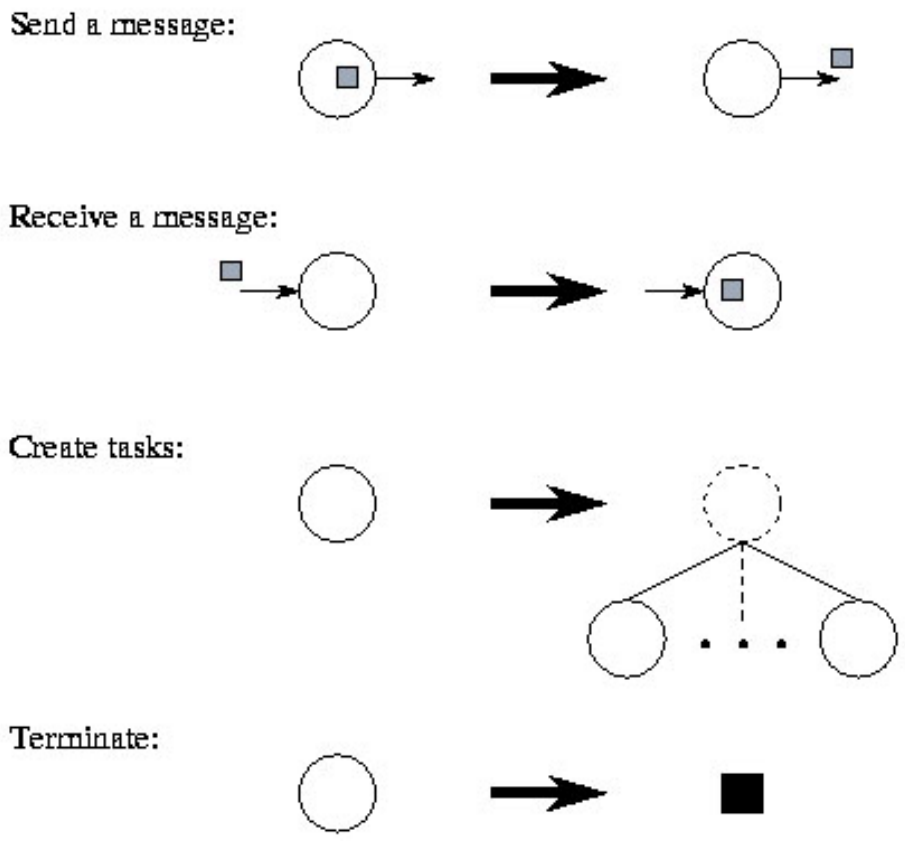
contenuti, e cioè reti locali di PC, dotandole di un sistema operativo che ottimizzi le comunicazioni fra i vari processori, offrendo così una potenza di calcolo il più possibile vicina a quella degli specializzati multicalcolatori e multiprocessori, ma a prezzi decisamente più bassi.

Gli algoritmi che determinano le allocazioni dei task possono essere più o meno efficienti a seconda che operino allocazioni statiche o dinamiche. Le allocazione statiche si rivelano più efficaci nel caso in cui i processori che compongono il cluster siano tra loro omogenei e le attività da distribuire siano tra loro uguali.

Nel caso di allocazione dinamica si distingue tra:

- data-oriented in cui al processore che ha esaurito il proprio task viene assegnato uno dei task in dotazione al processore con la coda maggiore;
- task-oriented in cui al processore che eseguito il proprio task ne viene assegnato uno nuovo.

Un calcolo parallelo consiste di una o più mansioni che si eseguono simultaneamente. Il numero di mansioni può variare durante l'esecuzione di programma; un'operazione incapsula un programma sequenziale e una memoria locale. Come evidenziato in figura 2 un'operazione può realizzare quattro azioni di base oltre che la lettura e scrittura della relativa memoria locale: trasmettere i messaggi sui relativi outports, ricevere i messaggi sui relativi inports, generare le nuove mansioni e terminare.



Operazioni generate da un'azione

Un funzionamento di trasmissione è asincrono: termina immediatamente. Un funzionamento di ricezione è sincrono: causa l'esecuzione dell'operazione ostruire fino a che un messaggio non sia disponibile. Gli accoppiamenti di Outport/inport possono essere collegati dalle code di messaggio denominate scanalature.

I primi cluster di PC realizzati per il calcolo parallelo erano poco più che dei “giocattoli”, con prestazioni molto limitate. L'interfaccia verso il programmatore era comunque costituita in genere da MPI (Message Passing Interface) o PVM (Parallel Virtual Machine), le stesse di cui la maggior parte delle piattaforme parallele ``vere'' sono dotate.

Lo scopo di questi cluster era essenzialmente quello di permettere, a costi ridotti, lo sviluppo ed il debugging di applicazioni da eseguire poi su calcolatori paralleli ben più potenti e costosi, nonché l'addestramento, sempre a costi assai contenuti, alla programmazione parallela. Nei cluster di PC le comunicazioni fra i

vari host utilizzavano i servizi di rete standard, tipicamente TCP o UDP, su reti locali di vecchia concezione quali Ethernet.

La complessità dei protocolli e la scarsa banda passante della rete fisica rende alta la latenza di comunicazione (tempo occorrente per comunicare un messaggio di lunghezza minima) e limita il throughput (velocità di spedizione dei messaggi vista dall'applicazione) del sistema di comunicazione.

In pratica, i cluster di PC e le NOW erano dei “giocattoli” sia per la limitata potenza di calcolo delle loro CPU sia per la scarsa efficienza delle comunicazioni tra processi, essendo la rete il vero collo di bottiglia di tali sistemi. In pochi anni la situazione è radicalmente cambiata.

Le prestazioni di calcolo delle moderne CPU sono enormemente aumentate. Inoltre, l'aumento della banda passante offerta dalle moderne LAN e il rapido sviluppo degli switch di rete hanno reso disponibili interconnessioni di rete locale a costo contenuto ma con prestazioni confrontabili con quelle delle interconnessioni inter-processore dei calcolatori paralleli commerciali. Vista l'attuale disponibilità di reti con bande di 1 Gigabit/s ed oltre, e la possibilità di dotare ciascun PC di più interfacce di rete, il collo di bottiglia di una NOW o di un cluster di PC è oggi costituito dal bus PCI (banda 133 MByte/s), almeno dal punto di vista hardware. L'atteso sviluppo di bus PCI a 64 bit (banda 266 MByte/s) dovrebbe risolvere tale situazione.

Tuttavia a tale impressionante sviluppo dell'hardware non ha corrisposto un analogo progresso dei protocolli di comunicazione, oggi del tutto inadeguati alle moderne LAN. Di fatto, il collo di bottiglia di una NOW o di un cluster di PC continua ad essere il sistema di comunicazione, ma stavolta più sul piano software che hardware.

Nell'ambito dello studio dell'elaborazione parallela massiccia (*MPP*) svolto alla *NASA* nasce nel 1994 il progetto Beowulf con lo scopo di studiare le potenzialità dei clusters di PC in attività di calcolo proprie delle workstation contemporanee, ma ad un costo non superiore. Nell'ottobre del 1996 è stato annunciato che un sistema Beowulf ha raggiunto una performance di regime di 1 Gigaflops in un'applicazione scientifica, per un costo totale inferiore i $50000,

non paragonabile con i costi dei calcolatori paralleli tradizionali, con cui invece la performance è paragonabile.

I sistemi operativi impiegati sono Linux e BSD che supportano un ampio insieme di periferiche e di piattaforme.

Accanto alla necessità di ottimizzare i tempi di calcolo si è detto dell'obiettivo di impiegare tecnologie economiche, per questa ragione molti di questi sistemi di classe Beowulf sono realizzati "riciclando" hardware dai laboratori, o comunque con processori non più attuali e che pertanto non sono utilizzati a causa dell' "evoluzione" del software.

Inoltre le principali librerie per la programmazione parallela basate sullo scambio di messaggi sono facilmente reperibili e permettono di realizzare un sistema "*open source*" basato su questa tecnologia in modo estremamente economico.

Ricorrere all'utilizzo di queste "*pile di pc*" presenta quindi evidenti vantaggi in termini di materiali, tecnologicamente collaudati e facilmente reperibile a costi contenuti, ed in termini di *scalabilità*: se le reti di comunicazione sono efficienti e, almeno inizialmente, sottosfruttate è possibile incrementare le prestazioni aggiungendo nuove risorse.

Tuttavia queste "pile di pc" non sostituiscono i calcolatori paralleli tradizionali, ma costituiscono un mezzo di calcolo complementare alle workstations di fascia alta, multiprocessori simmetrici e sistemi a memoria distribuita scalabili. Il software per questi sistemi è derivato dal lavoro di collaborazione di un ampio insieme di persone in aree quali i sistemi operativi, linguaggi, compilatori e librerie di calcolo parallele, ed è complessivamente noto con il termine di "Grendel Software Architecture".

La progettazione di software per questa piattaforma non può essere semplicemente basata sui principi della programmazione parallela, ma deve tenere conto anche delle latenze di comunicazione, bilanciando opportunamente l'elaborazione con la comunicazione e del problema che deve essere affrontato. Spesso le comunicazioni di rete sono ottimizzate impiegando più schede di rete su ciascuna macchina, in modo da realizzare un sistema connesso praticamente su più reti, ma che vengono collegate insieme in modo da funzionare come un'unica

rete con ampiezza di banda superiore (channel bonding). E' stato provato che per supportare il traffico dati dovuto a trasferimenti di files concorrenti su un sistema di classe Beowulf con sedici processori è sia *necessario che sufficiente* l'uso di due Fast Ethernet a 100 Mbps collegate in *channel bonding*.

Per ogni elemento del cluster possiamo ragionare in termini di *livello e* il modello di riferimento si chiama ISO/OSI.

pila ISO/OSI

Alla base della pila troviamo il *livello fisico* (physical layer) che rappresenta gli aspetti hardware del collegamento tra un computer e l'altro: a questo livello appartengono standard relativi ai mezzi trasmissivi, ai segnali elettrici ed ottici, ai connettori etc.etc.
Sopra il livello fisico troviamo il *livello di collegamento dati* (link layer): è il primo strato software e si occupa di trasferire una sequenza di bit tra due computer adiacenti; fornisce agli strati superiori gli strumenti (*primitive di collegamento*) per inviare blocchi di dati (frame) con una correzione trasparente degli errori di trasmissione.
Sopra il livello di collegamento c'è il *livello di rete* (network layer) che ha il compito

di rendere trasparente ai livelli superiori del protocollo la complessità della struttura della rete.

Sopra il livello di rete c'è il *livello di trasporto* (transport layer) il cui compito principale è di offrire, indipendentemente dalla complessità della rete sottostante, un canale di trasmissione virtuale tra due computer connessi, un canale semplice da usare e con un grado negoziabile di affidabilità nello scambio dei dati. Sopra il livello di trasporto c'è il *livello di sessione* (session layer) che fornisce le primitive per un dialogo ordinato tra due computer. Sopra il livello di sessione c'è il *livello di presentazione* (presentation layer) che offre la possibilità di uno scambio di dati indipendententemente dai formati dei dati usati dalla piattaforma sulla quale si opera.

Infine sopra il livello di presentazione c'è il *livello di applicazione* (application layer) che ha il compito di fornire i servizi applicativi, concretizzando l'utilizzo della rete da parte degli utenti.

Attualmente si utilizzano sistemi server ad alte prestazioni o proprio i cluster per gestire gli aspetti di livello più alto delle comunicazioni.

Rinvio al sito

Per osservare i risultati della simulazione, seppur con notevoli differenze dall'impostazione più realistica data all'inizio, proviamo ad effettuare una analisi dei risultati considerando:

ticksInATimeUnit 29;
quantità di secondi occorrenti per ottenere una determinata fornitura di ricette
batch stand alone 1/1000;
numero di lavorazioni interne o richieste di forniture di ricette del tipo *sequential batch* 1/1000.

Ciò significa essenzialmente che ogni tick della simulazione è rappresentativo di 1000 secondi nella realtà.

I risultati e l'analisi degli stessi è riportata sul sito ospitato dalla facoltà di Economia e Management "Luigi Einaudi" eco83.econ.unito.it/tesive

Bibliografia

Russel, S.J. (1998), Norvig, P., *Artificial Intelligence, a modern approach,* Prentice Hall International