DIDATTICA dell' INFORMATICA di BASE

CALCOLO PARALLELO

Architettura tradizionale per il calcolo parallelo e distribuito e possibile implementazione tramite l'uso di personal computer

di **Renato Agati**

INDICE

PREMESSA

Cosa si intende con il termine "**calcolo parallelo e distribuito**"?

Nella sua definizione più semplice intendiamo l'uso simultaneo di molteplici risorse di calcolo per eseguire un programma.

Oggi il software è scritto per un'esecuzione di calcolo sequenziale: il software viene infatti eseguito da un singolo computer con una singola CPU,
Il calcolo sequenziale eseguito su una singola macchina non è però sufficiente quando si devono velocizzare i calcoli e/o si devono eseguire calcoli complessi.
In questi casi si deve ricorrere all'utilizzo contemporaneo di più risorse di calcolo che può essere ottenuto:

- utilizzando un computer singolo con più processori;
- utilizzando più computer collegati in rete;
- implementando una combinazione di computer multi-processori collegati in rete.

Il calcolo non è più sequenziale bensì è un calcolo:

PARALLELO
(il programma viene eseguito in contemporanea da più processori)
e
DISTRIBUITO
(il programma viene diviso su più processori).

. . . .

In questo lavoro vengono dati riferimenti concettuali per lo sviluppo di un'applicazione di calcolo parallelo e distribuito, esaminando gli aspetti e le problematiche principali delle risorse hardware e software.

L'ultima sezione tratta le possibili applicazioni del calcolo parallelo e distribuito, inclusa una rapida descrizione di due importanti progetti avviati già da tempo.

Glossario

Prima di procedere, vediamo alcune terminologie ricorrenti quando si parla di "parallel computing"

Task
Sessione di lavoro, tipicamente un programma o un set di istruzioni eseguito dal processore.

Parallel Tasks
Applicazioni la cui esecuzione è indipendente una dall'altra, in modo che possono essere eseguite simultaneamente da processori multipli .

Esecuzione sequenziale
Esecuzione di un programma in maniera sequenziale, ovvero un'istruzione alla volta.

Esecuzione parallela
Esecuzione contemporanea di un programma diviso in più task

Memoria condivisa
Architettura hardware tale per cui ogni processore accede ad una porzione della memoria che hanno in comune

Memoria distribuita
Architettura hardware tale per cui ogni processore accede ad una propria memoria fisica.

Comunicazione
Scambio di dati tra task parallele

Sincronizzazione
Coordinazione dello scambio di dati tra processori coinvolti in task differenti.

Granularità
Misura qualitativa del rapporto tra calcolo e scambio dati.

Latenza (Parallel Overhead)
Quantità di tempo necessaria per coordinare sessioni di lavoro parallele. Tiene conto di:

- Tempo di avvio della sessione
- Sincronizzazioni
- Scambio dati
- Overhead del software
- Tempo di chiusura di una sessione di lavoro

Scalabilità
Identifica la capacità di un sistema parallelo di incrementare la potenza di calcolo (velocità di esecuzione) in maniera proporzionale all'aumentare del numero di processori.

PARTE 1

ARCHITETTURA DI UN CALCOLATORE PARALLELO

1. Le piattaforme hardware
2. Organizzazione logica: la tassonomia di Flynn
3. Organizzazione logica: la memoria
4. Organizzazione fisica: le reti di interconnessione

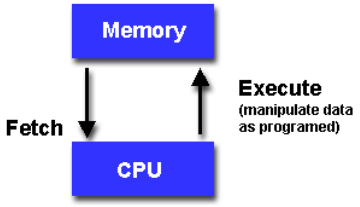
L'organizzazione fisica e' indipendente da quella logica

1. Le piattaforme hardware

Negli ultimi anni la velocità e la potenza di calcolo delle CPU è cresciuta notevolmente.
Nel 1965 Moore affermò che "*Il numero di transistor su un chip raddoppia ogni 18 mesi*" (Legge di Moor) con le conseguenze di aumento delle capacità di calcolo delle CPU (nel 1980 il processore 8088 lavorava a 4.77 MHz, oggi un Pentium IV lavora a oltre 3 GHz, ovvero a una velocità 650 volte superiore. Cresce anche il set di istruzioni) e aumento delle capacità dei chip di memoria.

Ma all'aumentare della memoria disponibile le dimensioni del software sono sempre cresciute in proporzione: Nathan afferma che "*Il software è come un gas: riempie sempre completamente qualsiasi contenitore in cui lo si metta*".

Si è instaurato un circolo vizioso (spinta tecnologica, nuove applicazioni, nuova spinta tecnologica…) che porta ad avere applicazioni sempre più pesanti e complesse ma pur sempre progettate per un calcolo sequenziale, ovvero calcolo eseguito da un computer avente un singolo processore (CPU), basato sul modello di macchina di Von Neumann, modello tuttora attuale per rappresentare il processo di elaborazione che si basa su:
fase di **fetch**, che comprende tutte le operazioni necessarie per il reperimento dell'istruzione da parte della CPU:
fase di **decode**, in cui l'istruzione viene interpretata;
fase di **execute**, che prevede l'invio in memoria, da parte della CPU, dei risultati del'elaborazione eseguita.
Nell'esecuzione di un programma, queste fasi si ripetono ciclicamente fino a quando il programma viene terminato.



Nonostante le evoluzioni tecnologiche, quando si devono velocizzare i calcoli e/o si devono eseguire calcoli complessi l'architettura di Von Neumann rappresenta un collo di bottiglia per varie ragioni.

- **Limite della memoria**
  un singolo computer ha risorse di memoria finite.
- **Limite della velocità di trasmissione**
  La velocità di un computer sequenziale è strettamente correlata alla velocità dei dati all'interno della struttura hardware. Questa velocità è limitata, e i suoi limiti sono rappresentati dalla velocità di trasmissione attraverso conduttori di rame (9 cm/nanosecondo) e dalla velocità della luce.
- **Limite della miniaturizzazione**
  La tecnologia dei processori consente di aumentare il numero dei transistor presenti in un chip riducendone le dimensioni, ma per quanto piccoli possano essere, esiste un limite.
- **Aspetti economici.**
  Un singolo processore sempre più veloce avrebbe costi superiori a quelli di più processori "tradizionali" che, messi insieme, offrirebbero la stessa potenza e prestazioni.

La soluzione al problema di avere piattaforme per calcoli veloci e complessi è rappresentata dall'utilizzo contemporaneo di più risorse di calcolo che può essere ottenuto:

- utilizzando un computer singolo con più processori;
- utilizzando più computer collegati in rete;
- implementando una combinazione di computer multi-processori collegati in rete.

Il calcolo non è più sequenziale come sulla macchina di Von Neumann, bensì è un calcolo **PARALLELO** (eseguito in contemporanea da più processori) e **DISTRIBUITO** (il programma viene diviso su più processori).

2. Organizzazione logica: la tassonomia di Flynn

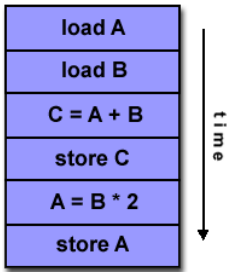
Si possono definire vari tipi di computer paralleli distinti secondo il tipo di connessione tra i vari processori e tra i processori e la memoria. La classificazione più utilizzata è la Tassonomia di Flynn.

La **tassonomia di Flynn** distingue le architetture multiprocessore in base a come i computer possono essere classificati in base alle due dimensioni indipendenti Istruzioni e Dati, in considerazione che entrambe queste dimensioni hanno solo due possibili stati: singolo o multiplo:

Istruzioni \ Dati	Singoli	Multipli
Singole	**SISD** Single Instruction, Single Data	**SIMD** Single Instruction, Multiple Data
Multiple	**MISD** Multiple Instruction, Single Data	**MIMD** Multiple Instruction, Multiple Data

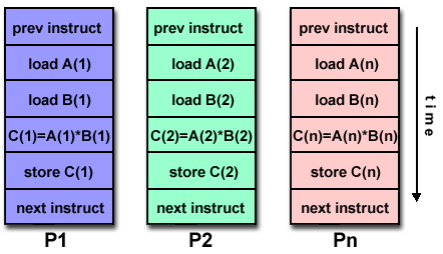
SISD – Single Instruction, Single Data

- E' un computer sequenziale (non parallelo);
- Single Instruction perché la CPU processa una sola istruzione ad ogni ciclo di clock;
- Single Data perché viene accettato solo un dato in ingresso ad ogni ciclo di clock;
- Rappresenta l'architettura di computer più vecchia ma ancora prevalente;
- Architettura utilizzata nella maggior parte di PC, workstation con singola CPU a mainframe.

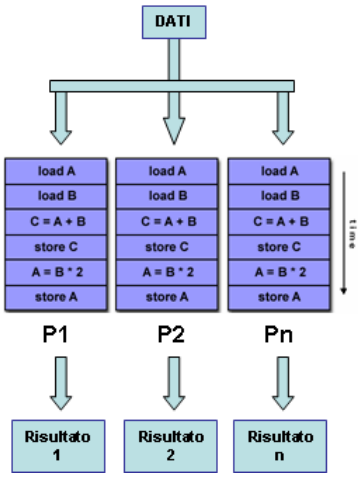


SIMD – Single Instruction, Multiple Data

- E' un computer multiprocessore
- Single Instruction perché tutte le CPU eseguono la stessa istruzione durante lo stesso ciclo di clock, in modo sincrono
- Multiple Data perché ogni CPU può operare su dati differenti;
- Esiste una unità di controllo che organizza e distribuisce le istruzioni da inviare ad ogni CPU;
- Architettura utilizzata, ad esempio, nell'IBM 9000, nel Cray C90 e nel Fujitsu VP.



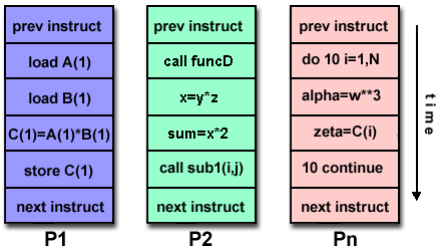
MISD – Multiple Instruction, Single Data



Architettura poco utilizzata in quanto non ha, nella realtà, applicazioni concrete.

Una di esse può essere l'applicazione di differenti algoritmi di crittografia applicati agli stessi dati in ingresso.

MIMD – Multiple Instruction, Multiple Data



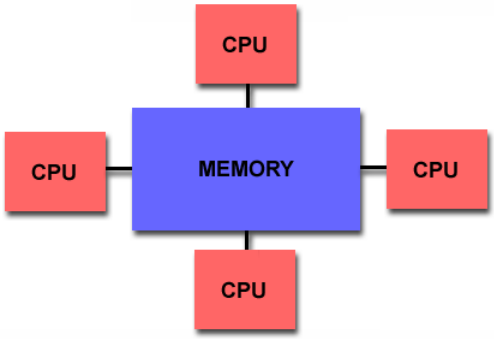
- Attualmente è l'architettura di calcolo parallelo più diffusa;
- Multiple Instruction perché ciascun processore può eseguire istruzioni differenti;
- Multiple Data perché ogni processore può lavorare su dati differenti;
- Non ha bisogno di unità di controllo:
- Architettura utilizzata nei supercomputer, nelle reti di computer in parallelo (GRID computer) e nei PC multiprocessori.

3. Organizzazione logica: la memoria

In un'architettura parallela, la memoria può essere:
- condivisa (piattaforma shared-address-space)
- distribuita (piattaforma message-passing)
- condivisa e distribuita (hybrid)

Memoria condivisa (piattaforma shared-address-space)

- La memoria è in comune ed è accessibile a tutti i processori;
- I cambiamenti effettuati da un processore in una locazione di memoria, sono visibili a tutti gli altri processori (cache coherency);
- L'accesso alla memoria può essere:
  - uniforme (**UMA**)
  - non uniforme (**NUMA**)



Accesso alla memoria uniforme (**UMA** - Uniform Memory Access):
- Processori identici
- tempo di accesso di ogni processore alla memoria è uguale per tutti;
- utilizzo nelle macchine SMP – Symmetric MultiProcessor

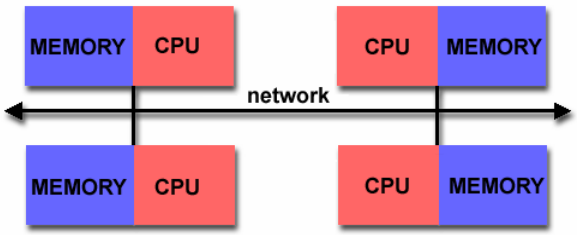
Accesso alla memoria non uniforme (**NUMA** – Non Uniform Memory Access):
- tempo di accesso differente per ogni processore
- un processore può accedere direttamente alla memoria di un altro processore.

Vantaggi della memoria condivisa:
- la vicinanza della memoria con i processori garantisce uno scambio dati veloce e uniforme.

Svantaggi della memoria condivisa:
- l'aggiunta di nuovi processori aumenta il traffico sul bus della memoria condivisa;
- aggiungendo nuovi processori è necessario rivedere l'applicazione per risincronizzare gli accessi alla memoria.

Le prestazioni di una piattaforma con memoria condivisa possono migliorare se ogni CPU ha una propria cache.

Memoria distribuita (piattaforma message-passing)



- Ogni processore ha una sua propria memoria locale che non può essere condivisa con un altro processore.
- Ogni processore lavora in maniera indipendente dagli altri processori;
- i cambiamenti eseguiti nella memoria da un processore non influenzano il calcolo degli altri processori.
- Architettura di memoria presente nel caso in cui il calcolo parallelo viene implementato con più sistemi collegati tra loro attraverso una rete locale.

Vantaggi della memoria distribuita:

- l'aumento del numero di processori è accompagnato da un aumento proporzionale della memoria;
- ogni processore può accedere alla propria memoria senza interferenze di altri processori.

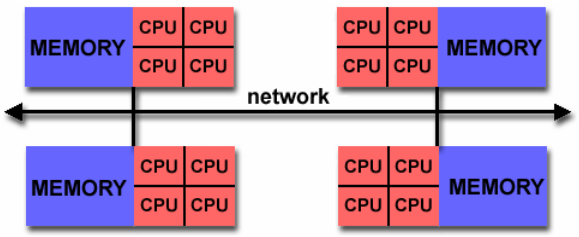
Svantaggi della memoria distribuita:

- una maggiore complessità del software che deve garantire una corretta sincronizzazione tra le task di due processori quando uno di essi deve accedere ai dati elaborati dall'altro.
- Accesso di tipo NUMA (tempi non uniformi)

Memoria condivisa-distribuita (hybrid)

L'architettura di memoria che oggi prevale su tutte e che secondo le previsioni continuerà a prevalere sulle altre è l'architettura ibrida di memoria condivisa e distribuita:

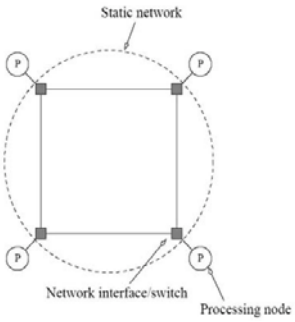
In quest'architettura, sistemi SMP lavorano ciascuno con la propria memoria condivisa, e la connessione in rete viene utilizzata solo quando è necessario muovere dati da un sistema SMP a un altro.



I **vantaggi** e **svantaggi** offerti da questa architettura ibrida sono gli stessi visti nelle architetture di memoria condivisa e memoria distribuita.

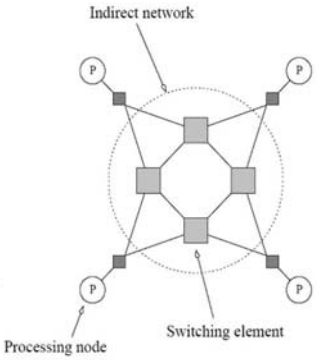
4. Organizzazione fisica : le reti di interconnessione

Le reti di interconnessione per computer paralleli forniscono i collegamenti processore-processore e processore-memoria.
Le reti sono classificate in statiche e dinamiche.



Interconnessioni statiche

- Consistono di un numero di collegamenti punto-punto.
- Ogni nodo è connesso ad altri due tramite un'interfaccia di rete.
- La configurazione dei nodi è "a maglie"



Interconnessioni dinamiche

- Ogni nodo è connesso agli altri in modo dinamico tramite switch.
- Gli switch stabiliscono i percorsi tra i processori e i banchi di memoria

Diverse tipologie di rete

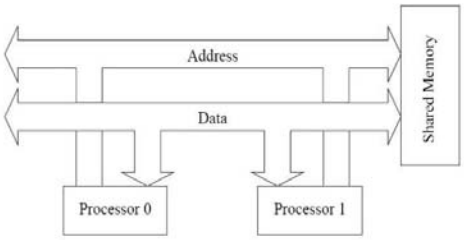
Il calcolo parallelo può essere implementato utilizzando diverse tipologie di rete.

Ciascuna tipologia è caratterizzata da:

diametro	distanza massima tra due nodi *(meglio piccoli diametri)*
connettività	minimo numero di archi che devono essere rimossi per dividere la rete in due reti disconnesse *(meglio alta connettività)*
bisection width	minimo numero di archi che devono essere rimossi per dividere la rete in due parti uguali *(meglio alta)*
bisection bandwidht	minimo volume di comunicazione permesso tra due metà di una rete *(meglio grande)*

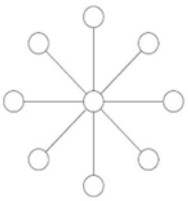
Bus-based network

- Un bus interfaccia tutti i nodi.
- Il diametro è 1
- La connettività è 1



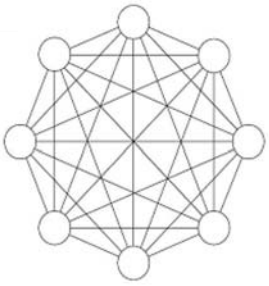
Rete a stella

- Simile alla rete bus-based
- Il nodo centrale svolge il ruolo del bus



Reti completamente connesse

- Ogni nodo è direttamente collegato con ogni altro nodo nella rete
- Il diametro è 1
- La connettività è n (numero dei nodi)



Routing

Qualunque sia la tipologia di rete, si deve sviluppare un algoritmo per determinare il percorso che un messaggio prende per andare dal nodo sorgente al nodo di destinazione.

Il routing può essere:

- **Minimo**
  quando viene selezionato sempre il percorso più breve, con il rischio di congestione della rete

- **Non minimo**
  quando per evitare di congestionare la rete prende percorsi più lunghi

- **Deterministico**
  quando viene determinato un unico percorso tra sorgente e destinazione

- **Adattivo**
  quando usa informazioni riguardo lo stato della rete per scegliere il percorso.

Parte 2

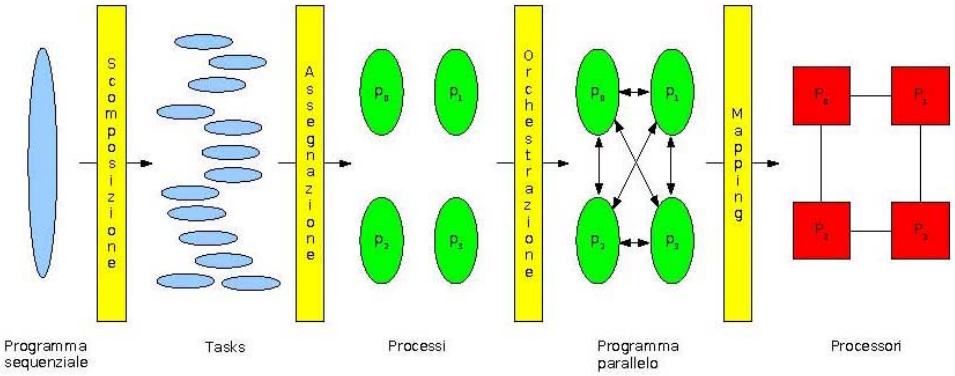
Il software dei sistemi paralleli

1. Progettazione di algoritmi paralleli
2. Modelli di programmazione parallela
3. Parallelizzazione automatica o manuale

1. Progettazione di algoritmi paralleli

La progettazione di un algoritmo parallelo si basa principalmente su:

- scomposizione in task
- assegnazione dei task
- orchestrazione
- mapping

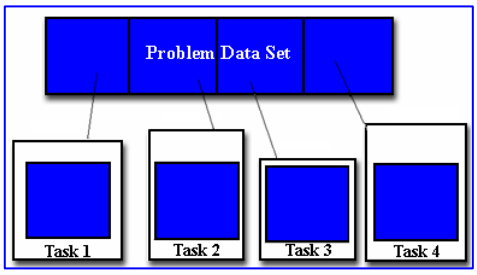


Scomposizione
In questa fase il programma viene scomposto in un insieme di task.

Ci sono due modi di scomporre un programma in task parallele:
scomposizione "domain"
scomposizione funzionale

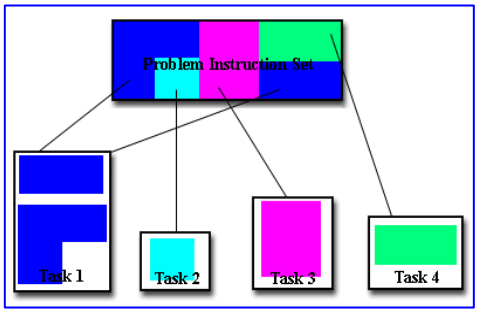
Nella **scomposizione domain** vengono scomposti i dati associati al problema.
L'applicazione è comune a tutti i processori che lavorano ciascuno su una porzione di dati

Esempio: elaborazione di una gran quantità di dati



Nella scomposizione funzionale invece è il problema ad essere scomposto in task.

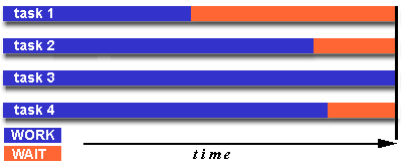
Esempio: un segnale audio viene processato applicando quattro distinti filtri. Ogni filtro rappresenterà una task dell'algoritmo parallelo.



Assegnazione
In questa fase viene specificato il meccanismo per cui i task sono distribuiti tra i processi.

Si cerca di bilanciare il carico (load balancing) per vari motivi:

- tutti i task devono essere in esecuzione per tutto il tempo, ovvero i processori devono lavorare con continuità;
- si deve pertanto tenere in considerazione l'eterogeneità dei sistemi impiegati;
- si deve minimizzare il volume delle comunicazioni.



Orchestrazione
In questa fase vengono specificati i seguenti punti:

- organizzazione della distribuzione dei dati;
- organizzazione dell'ordine temporale dei task;
- organizzazione della sincronizzazione e comunicazione tra i processi;
- riduzione della comunicazione tra i processi:
- riduzione della serializzazione presso le risorse condivise;

Mapping
La fase di mapping consiste nell'associare i processi alle risorse fisiche.

Un aspetto fondamentale è rappresentato dalla **comunicazione tra i processi** e l'identificazione di quali task devono comunicare tra loro è un punto critico della programmazione parallela.
E' quindi necessario, sviluppando un programma parallelo, tenere in considerazione aluni fattori:

- **latenza e banda passante**
  lo scambio di piccoli messaggi può aumentare la latenza e diminuire quindi l'efficienza della comunicazione. Se la banda passante è sufficiente, può essere conveniente impacchettare tanti piccoli messaggi in un unico messaggio più grande.

- **comunicazione sincrona o asincrona**
  la comunicazione sincrona, essendo una comunicazione a blocchi, richiede che altri processi vengano fermati fino a quando la comunicazione non è terminata

- **campo della comunicazione**
  la comunicazione può coinvolgere solo due task (point-to-point) con una task "mittente" e un'altra "destinatario", così come può coinvolgere più di due task (collettiva) ed in tal caso può essere "broadcasted", "scattered", "gathered".

- **efficienza della comunicazione**
  l'architettura di memoria condivisa o distribuita, la modalità sincrona o asincrona, la piattaforma hardware e l'infrastruttura di connessione sono fattori che influenzano l'efficienza della comunicazione e quindi la "bontà" della parallelizzazione

La "**bontà**" di una parallelizzazione può essere verificata attraverso due fattori:

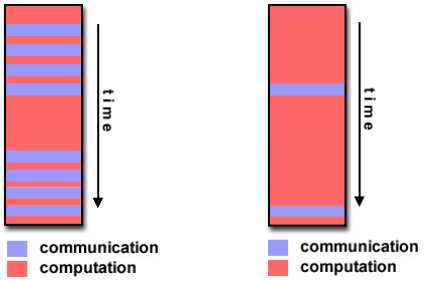
- **granularità**
- **speedup**

Granularità

Fasi di calcolo sono tipicamente alternate con fasi di comunicazione per sincronizzare gli eventi e scambiare dati.
La granularità è il rapporto tra il tempo di calcolo e il tempo di comunicazione.

Se la granularità è sottile, vuol dire che poca parte di calcolo è svolta tra eventi di comunicazione e l'overhead per eventi di comunicazione è superiore a quello per il calcolo, a discapito di miglioramenti prestazionali.

Viceversa, se la granularità è grossolana (coars grain) vuol dire che le fasi di calcolo sono prevalenti a quelle di comunicazione



Una granularità sottile facilita il bilanciamento del carico (load balancing)

Speedup

E' il rapporto tra il tempo T_1 di esecuzione di un programma sequenziale (su un solo processore) con il tempo Tn di esecuzione del programma in parallelo su *n* processori

$$S = \frac{T_1}{Tp}$$

2. Modelli di programmazione parallela

I modelli di programmazione parallela più in uso sono:

- **Shared memory**
- **Threads**
- **Message Passing**
- **Data parallel**

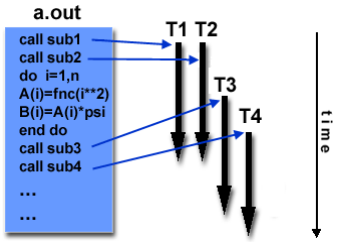
Sono modelli che non fanno riferimento a specifiche configurazioni hardware e di memoria, e che quindi possono, teoricamente, essere implementate su qualunque piattaforma hardware.

Modello "Shared memory" (memoria condivisa)
Come già visto, in questo modello le task condividono un'unica area di memoria nella quale scrivono e leggono in maniera asincrona.

Questo modello offre al programmatore il vantaggio che non deve esplicitare le comunicazioni di dati tra le task.

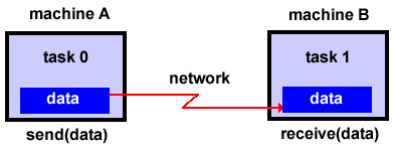
Modello "Threads"
In questo modello il processo può avere percorsi multipli di esecuzione, come se le task fossero delle "subroutine" del programma principale.

Questo modello richiede al programmatore particolare attenzione nella sincronizzazione tra i thread, poiché operano su memoria condivisa.



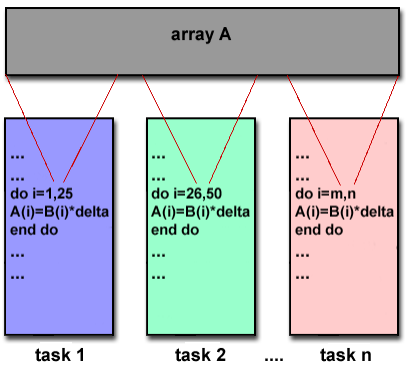
Modello "Message Passing"
Nel modello message passing ogni processore ha la sua propria memoria.

Il programmatore è responsabile di determinare il parallelismo e lo scambio di dati.



Modello "Data Parallel"
In questo modello abbiamo più task che operano nella stessa struttura di dati, ma ciascuna singola task opera su una porzione differente di dati.

Nell'implementare questo modello, il programmatore deve provvedere a sviluppare un programma che specifica la distribuzione e l'allineamento dei dati.



3. Parallelizzazione automatica o manuale

Lo sviluppo di programmi paralleli è un'attività svolta tipicamente dal programmatore che è responsabile di identificare ed implementare la parallelizzazione.

Quest'attività, svolta manualmente, comporta notevoli perdite di tempo e non è escluso commettere errori.

Per questo sono stati resi disponibili vari tool di sviluppo per assistere il programmatore nella conversione di programmi sequenziali in paralleli.
Questi compilatori lavorano principalmente in due modi differenti:

- **automatico**
  Il compilatore analizza il codice sorgente ed identifica le opportunità di parallelizzazione, facendo anche una analisi di convenienza basata su costi e incremento di prestazioni;
- **semi-automatico**
  tramite delle direttive di compilazione, il programmatore definisce esplicitamente come parallelizzare il codice.

La modalità automatica sembra, ovviamente, essere la preferibile; tuttavia tale sistema presenta alcune limitazioni quali:

- produzione di risultati non corretti;
- prestazioni non ottimizzate;
- minore flessibilità rispetto alla parallelizzazione manuale;
- subset di codici limitato.

Qualunque sia l'approccio seguito, il **programmatore deve**:

- determinare innanzitutto se il problema può essere parallelizzato
  la serie numerica di Fibonacci (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …), ad esempio, non può essere calcolata applicando il calcolo parallelo perché il calcolo di un dato è strettamente dipendente dal calcolo del dato precedente: K = (K-1)+(K-2).
- conoscere a fondo il problema
  se si parte da un programma sequenziale, deve conoscerne la funzionalità e i codici sorgente;
- identificare i punti deboli del programma
  alcune parti del programma, come la gestione degli I/O, può risultare spropositatamente lenta e rallentare di conseguenza tutto il programma. In tal caso, il programmatore deve cercare algoritmi differenti per eliminare, o comunque ridurre, tali colli di bottiglia.
- Identificare eventuali inibitori al parallelismo
  l'inibitore più comune è la dipendenza dai dati, come nel caso della sequenza di Fibonacci.
- Verificare l'implementazione di altri possibili algoritmi
  Anche se il programma funziona, possono esistere altri algoritmi che ne migliorano l'eseguibilità e quindi l'intera efficienza.

PARTE 3

L'utilizzo dei sistemi paralleli

1. Applicazioni del calcolo parallelo e distribuito
2. Il Grid Computing
3. I progetti S.E.T.I. e SETI@home
4. Il progetto APEnext

1 Applicazioni del calcolo parallelo e distribuito

Si è detto che l'uso di architetture parallele nasce dalla necessità di trovare un modo per risolvere in tempi ragionevolmente brevi problemi che altrimenti non lo sarebbero.

L'architettura basata su cluster di PC (detta **COW** – Cluster of Workstation) per il calcolo parallelo rappresenta un'interessante alternativa ai costosi supercomputer paralleli: queste architetture permettono di avere potenza di calcolo in modo flessibile, scalabile e soprattutto molto economico.
La disponibilità di grande potenza di calcolo è un fattore fondamentale in molti settori della **ricerca scientifica**, ed assume un ruolo sempre più importante anche in vari settori applicativi che vanno dalla **progettazione industriale**, ai **grandi database** ed ai **server Internet** di ultima generazione.

Esiste peraltro un numero sempre maggiore di applicativi software che vengono utilizzati sui cluster di PC: dal calcolo scientifico (bioinformatica, meteorologia, fluidodinamica, chimica teorica, fisica delle particelle, …), alla progettazione industriale (prototipazione, calcoli agli elementi finiti, ottimizzazione, ...), ai server ad alte prestazioni (web-server, file-server, ASP, database, ...).
Tra questi menzioniamo il DBMS sviluppato da **Oracle** chiamato "10g" che sfrutta l'architettura GRID e si sta imponendo come standard di mercato per applicazioni destinate a banche e a grandi aziende.

In campo scientifico si sta oggi assistendo ad un rapido sviluppo delle conoscenze nel settore della biologia, e questo è in gran parte dovuto all'utilizzo di strumenti informatici per il calcolo e l'elaborazione dei dati. L'esigenza di memorizzare, gestire ed elaborare questi dati ha portato alla nascita della **bioinformatica**, intesa come disciplina che si occupa dello studio, dello sviluppo e dell'integrazione di strumenti informatici al servizio della ricerca scientifica in campo biotecnologico.
Fra i fattori chiave nella ricerca biotecnologica vi è la capacità di elaborare ed estrarre informazioni utili dalle grandi moli di dati oggi accumulati, attività che richiede una considerevole potenza computazionale. Le applicazioni possibili vanno dalla ricerca di omologie fra sequenze biologiche, alla predizione della struttura delle proteine, alla visualizzazione ed elaborazione di dati sperimentali, ecc...
Il **calcolo parallelo** e l'utilizzo di computer ad alte prestazioni come i **cluster di PC** permettono di assolvere a queste esigenze.

Altra applicazione che può avvantaggiarsi di architetture parallele è nel campo della computer grafica per processi di ray tracing (renderizzazione di una scena in uno spazio tridimensionale su una matrice bidimensionale).

2 Il GRID Computing

Il **GRID Computing** è un nuovo rivoluzionario concetto di collegamento tra computer destinato in futuro, secondo gli esperti, a sostituire il modello attuale di Internet.

La differenza fondamentale tra Internet e GRID è che mentre il primo resta sostanzialmente un collegamento indiretto tra computer isolati - in cui ogni utente è collegato a un dato punto di traffico gestito dal provider, da cui parte la connessione verso altri utenti - il secondo applica all'interconnessione un concetto simile a quello usato per la distribuzione dell'energia elettrica.
In questo modo e' possibile collegare gli utenti direttamente tra loro, creando una maxi-rete di computer capace di mettere a disposizione di ogni singolo appartenente l'immensa capacità di memoria e di calcolo dell'intero sistema.
Un'altra differenza è che mentre Internet consente di condividere documenti (attraverso il WWW) e applicazioni (Web Service), GRID consente di condividere tutte le risorse eterogenee disponibili (server, storage, applicazioni) per fornire un unico sistema di calcolo in grado di soddisfare le esigenze di elaborazione di ingenti quantità di calcolo.

L'idea di utilizzare la potenza di calcolo dei computer distribuiti nel mondo (il cui utilizzo è mediamente il 5-10% della reale potenzialità) nasce all'inizio degli anni '90 negli ambienti di ricerca scientifica che hanno la necessità appunto di elaborare ingenti quantità di dati.

Tra gli artefici del GRID Computing c'è il CERN di Ginevra che ha l'esigenza di gestire i 15 petabyte di dati (ossia i 15 milioni di miliardi di byte) prodotti dall'acceleratore LHC, attualmente in costruzione a Ginevra. Per il 2007, anno in cui l'LHC dovrebbe entrare in attività, la rete GRID avrà raggiunto una potenza equivalente a quella di 100.000 attuali computer superveloci: lavoreranno tutti insieme, formando un unico supercomputer virtuale.

La piattaforma prevede uno strato di middleware fra sistema operativo e applicazione GRID. Il disaccoppiamento fra sistema operativo e GRID consente a quest'ultimo di non richiedere installazione e integrazione col sistema operativo e dunque di funzionare su Windows, Linux e Mac. Il middleware supporta richieste di invio e ricezione dei dati senza richiedere una conoscenza dei protocolli di rete all'utente-programmatore, indipendentemente dal sistema operativo utilizzato.
Gli sviluppi più importanti hanno riguardato la messa a punto di un protocollo di comunicazione efficiente per far dialogare tra loro i calcolatori e lo studio delle modalità più efficaci per scomporre i problemi e i calcoli complessi in una miriade di elaborazioni elementari adatte ad essere distribuite sui nodi del GRID.

Per il futuro è già in corso un'iniziativa di finanziamenti della UE rivolta a far uscire GRID dall'ambito della ricerca scientifica e portarlo, come prima tappa, nel mondo dell'industria e delle imprese, e trasformarlo successivamente in uno strumento a disposizione del maggior numero di utenti possibile, fino a sostituire gradualmente il sistema (rivelatosi lento e macchinoso) che regola le attuali connessioni via Internet. I progetti sponsorizzati hanno tutti come obiettivo l'applicazione pratica della nuova maxi-rete, destinata a favorire la ricerca industriale nel campo della protezione dei pedoni, dei prodotti farmaceutici e dell'industria aerospaziale.

3 I progetti S.E.T.I. e SETI@home

Il progetto SETI

SETI, acronimo di "*Search for Extra Terrestrial Intelligence*", cioè ricerca di intelligenza extraterrestre, è un progetto internazionale che si pone come obiettivo la ricerca di vita intelligente nella profondità dello spazio.

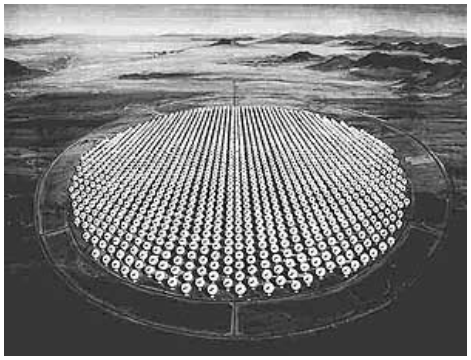
Essendo impossibile l'utilizzo di sonde (le distanze da coprire per una missione interstellare sono proibitive e per raggiungere la stella più vicina, alla modesta distanza di 4.3 anni luce - circa 40.000 miliardi di chilometri - si impiegherebbe più di 70.000 anni), per venire a sapere della loro esistenza potremmo cercare qualcosa che loro abbiano creato, qualcosa di indiscutibilmente artificiale che sia giunto fino a noi, e questa prova possono essere le onde radio.

I primi a ipotizzare una ricerca di questo tipo furono nel 1959 Giuseppi Cocconi e Philip Morrison, della Cornell University, che suggerirono agli astronomi di puntare i loro radiotelescopi verso le stelle di classe solare e cercare eventuali segnali provenienti da civiltà ET alla frequenza di 1420 MHz. Questa è la linea di emissione radio dell'idrogeno neutro ed è importante in radioastronomia in quanto permette di studiare l'estensione ed il moto della nostra galassia. Infatti, negli spazi che esistono fra le stelle non vi è completamente il vuoto, bensì vi sono alcuni atomi di idrogeno.
Una civiltà con semplici rudimenti di radioastronomia avrebbe già scoperto questa emissione radio e sarebbe quindi già in possesso di ricevitori dedicati all'analisi di questa frequenza.

Il primo progetto di ricerca al fine di scoprire eventuali segnali di chiara origine artificiale.fu avviato all'inizio degli anni '60 da Frank Drake. Il progetto era assai modesto e non durò molto: l'analisi veniva fatta solo sui 1420 MHz ed il target erano solo 2 stelle (Tau Ceti e Epsilon Erinadi). Si trattava, comunque, dell'inizio vero e proprio della ricerca SETI.

Nonostante Drake non trovò alcun segnale, ricerche simili furono portate avanti da altri astronomi in altre parti del mondo che utilizzarono il tempo libero dei loro radiotelescopi, per puntare nelle direzioni concesse dalla posizione dell'antenna in quel momento. Non si trattava di progetti organizzati sistematicamente: non erano sintonizzati alle frequenze migliori, non utilizzavano i ricevitori adatti e non puntavano nelle zone di cielo più interessanti.

Negli anni '70 la NASA iniziò ad interessarsi alla ricerca SETI avviando un proprio progetto che prevedeva, tra l'altro, la costruzione di un osservatorio dedicato unicamente alla ricerca SETI.



Tale progetto, conosciuto come il progetto Cyclops, non fu mai realizzato sia per la maestosità dell'impresa, un elevatissimo numero di antenne disposte su di un cerchio dal diametro di 16 chilometri, che per il suo l'unico fine, il SETI.

Dopo tante polemiche, studi, ore di lavoro e progettazione, il NASA SETI partì nel 1992, il suo budget consisteva di 12 milioni di dollari all'anno, una piccolissima percentuale del budget della NASA, ma dopo solo un anno il congresso americano tagliò tutti i fondi per continuare la ricerca. I dollari per le attrezzature d'avanguardia erano ormai stati spesi e solo il 0.1% delle analisi previste erano state eseguite.

SETI@home

Il progetto SETI è ora portato avanti, con il nome Phoenix, dal Seti Institute. Questo è un Istituto che si sostiene con donazioni provenienti da privati e dalle molte industrie hi-tech che si trovano nella Silicon Valley (California) dove lo stesso ha sede. Per l'eventuale rivelazione di questi debolissimi segnali nel campo radio, sono necessari sofisticati analizzatori di spettro digitali, tipo il SERENDIP IV, collegati a grandi antenne dei radiotelescopi presenti nel mondo.

La ricerca di intelligenza extraterrestre consiste essenzialmente nel campionare mediante enormi parabole (dell'ordine di centinaia di metri) segnali provenienti dallo spazio, i quali devono essere accuratamente analizzati per verificare se tali segnali sono dovuti a fenomeni naturali (ad esempio la morte di una stella) oppure a forme di vita intelligenti.

E' proprio quest'ultima fase che mette in crisi i laboratori di ricerca SETI, poiché in un solo secondo di campionamento viene raccolta una grande quantità di dati che richiede molto tempo per essere analizzata.

Nel 1996 David Gedye, con Craig Kasnoff, concepisce l'idea di **SETI@home**, una architettura di calcolo distribuito che utilizza milioni di computer sparsi in tutto il mondo: ogni computer scarica piccole "unità" di dati campionati (~348Kb), le analizza ed invia i risultati dell'analisi al laboratorio.

Il tempo per analizzare una unità è strettamente vincolato al tipo di computer utilizzato ed in media si aggira normalmente intorno alle 10 ore. Il programma che analizza le unità è in realtà uno screen-saver che lavora nei tempi "morti" (idle time) della CPU, ossia negli intervalli di tempo in cui il microprocessore non elabora informazioni (in questo modo l'utente non rileva alcun calo di prestazioni).

SETI@home fornisce quindi la possibilità a chiunque possieda un PC di dare il proprio contributo alla scienza.

4 Il progetto APEnext

L'Istituto Nazionale di Fisica Nucleare (INFN), attraverso il Dipartimento di Fisica dell'Università La Sapienza di Roma, ha realizzato una potente piattaforma di calcolo la cui struttura si rifà al calcolo parallelo.

L'obiettivo dell'INFN è quello di affrontare la barriera delle conoscenze in fisica teorica: scoprire la struttura intima di protoni e neutroni e trovare risposta agli interrogativi quali l'asimmetria tra materia e antimateria.

Per raggiungere tali obiettivi sono necessari computer capaci di produrre ed elaborare una mole gigantesca di dati.

Nel 2001 l'INFN avvia, insieme all'azienda italiana Exadron, lo sviluppo di un super computer basato su un tipo di processore disegnato appositamente ottimizzato per le simulazione della QCD (Cromo Dinamica Quantistica).
Il processore utilizzato è il J&T, prodotto da IBM, che ha la caratteristica di avere un'interfaccia per la realizzazione di array di processori, ideale quindi per applicazioni di **calcolo parallelo**.

Il supercomputer viene chiamato **APEnext**, e rappresenta la quarta generazione della famiglia APE (Array Processor Experiment) nata negli anni '80.

Presentato nel gennaio 2005, i primi prototipi di APEnext stanno lavorando già presso i laboratori INFN di Roma e Ferrara.

APEnext occupa lo spazio di una quindicina di armadi, ha un basso consumo di energia e, soprattutto, offre una potenza di calcolo di 12 Teraflops (dodicimila miliardi di operazioni al secondo), potenza che, a detta del direttore tecnico di Exadron, "può sempre essere aumentata, entro certi limiti, mettendo sempre più **macchine in parallelo**".

APEnext si basa su un modello di programmazione parallela tipo **message passing**. Infatti la sua struttura base è costituita da un array di unità di calcolo, ciascuna avente un processore (floating point in doppia precisione da circa 1.6 GFlops) e i propri banchi di memoria (da 256MB a 1 GB)

Il laboratorio APENEXT è il quinto laboratorio di calcolo del pianeta ed è comparabile ai maggiori laboratori della NASA.
Per potenza di calcolo, i primati appartengono ancora ad USA e Giappone con l'IBM Blue Gene/L (capace di oltre 70 teraflops) e Nec SX-8 (65 teraflops).