Modelli di programmazione parallela

Oggi sono comunemente utilizzati diversi modelli di programmazione parallela:

- Shared Memory
  - Multi Thread
- Message Passing
- Data Parallel

Tali modelli non sono specifici per una particolare tipologia di macchina parallela o di architettura della memoria: almeno in teoria, tutti possono essere implementati su qualunque modello di computer parallelo (GlobalArray su cluster). In pratica architettura della macchina e linguaggi non devono essere considerati come due modelli indipendenti poiche' l'architettura parallela ha una forte influenza, in termini di efficienza, sui costrutti utilizzati per sfruttare il parallelismo.

Shared Memory

- Oltre alla propria memoria privata, i processi condividono uno spazio di memoria, da cui leggono e scrivono asincronamente
- Attraverso lock e semafori è possibile controllare l'accesso alla memoria condivisa tra i processi

- Vantaggi:
  - Non richiede la programmazione esplicita della comunicazione di dati tra processi
  - Sovente lo sviluppo di programmi paralleli e' incrementale a partire dalla versione seriale del codice

- Svantaggi:
  - Inadatto a problemi in cui i processi concorrenti siano "particolarmente interconnessi"
  - E' complesso gestire la località dei dati al processo, con conseguenti problemi di performance dell'applicazione

Thread

- Un processo singolo puo' creare alcuni thread (unita' d'esecuzione), che il sistema operativo "schedula" in concorrenza (magari su CPU diverse)
- Ogni thread
  - condivide tutte le risorse associate al processo che lo ha generato (compreso il suo spazio di memoria)
  - ha una sua area di memoria privata
  - comunica con gli altri thread attraverso lo spazio di memoria globale
- Sono necessarie operazioni di sicronizzazione per evitare che, nello stesso istante, piu' di un thread cerchi di aggiornare una locazione di memoria globale
- Il modello multi threaded e' normalmente associato ad architetture shared memory
- Possibili implementazioni:
  - Una libreria di primitive specifiche per la programmazione multi thread
  - Un insieme di direttive di compilazione da inserire nel codice seriale

Posix thread e OpenMP

Storicamente sono stati condotti 2 tentativi di standardizzazione della programmazione multi thread che hanno condotto a 2 differenti implementazioni: POSIX Threads e OpenMP

- POSIX Threads
  - Specificato dallo standard IEEE POSIX 1003.1c (1995)
  - Comunemente indicato come Pthreads
  - E’ una libreria utilizzabile in codici C
  - Diversi vendor offrono una versione ottimizzata della libreria Pthreads
  - La gestione del parallelismo e' completamente esplicita
- OpenMP
  - Basato su direttive di compilazione
  - Standard definito congiuntamente da un gruppo di hardware e software vendor
  - L’API OpenMP Fortran e' stata rilasciata alla fine del ’97, mentre quella per il C/C++ alla fine del ‘98
  - Puo' essere molto semplice da usare, perche' consente effettuare una “parallelizzazione incrementale” a partire dal codice seriale

Posix thread

Vediamo un paio di esempi con pthread ed openmp.

Pthread: la pthread_create() crea i thread e restituisce un identificativo unico per ogni thread creato. Il chiamante provvede i vari thread di una funzione da eseguire. La join e' l'analogo del la wait nell'esempio del process forking. Il mutex (mutual exclusion) e' il meccanismo base necessario alla sincronizzazione tra i vari theads. (E' necessario sempre essere sicure della thread safety delle librerie che si usano)

OpenMP

OpenMP: OpenMP (Open Multi Processing) e' una tecnica di parallelizzazione directive-based. E' un tipo di parallelismo cosi' detto fork-join, pensato essenzialmente per sistemi shared-memory (ma vedi Cluster OpenMP). Supporta Fortran (77 and 90), C and C++. E' uno standard de facto. Attualmente molti compilatori commerciali suspportano l'OpenMP. Vediamo un esempio in Fortran (. /opt/intel/fort9/bin/ifortvars.sh)

In C/C++ in generale si usa la direttiva #PRAGMA (#pragma omp, in generale pragma e' la direttiva per passare opzioni al compilatore dal sorgente):

```
#pragma omp parallel for
```

OpenMP

Tutte le linee che iniziano con !$OMP sono direttive OpenMP. directives to specify parallel computing. Esempio, ''!$OMP PARALLEL'' e ''!$OMP END PARALLEL'' rinchiudono una regione parallela, tutto il codice in quella regione e' eseguito da tutti i thread. Il numero di thread e' di solito specificato da una variabile di enviroment OMP_NUM_THREADS.

In una regione parallela i dati possono essere sia privati che pubblici.

Un compilatore OpenMP automaticamente traduce il programma facendo uso delle pthread, puo' cosi' essere eseguito in un sistema shared memory. (Cluster OpenMP fa eccezione).

Message Passing

- Un insieme di processi usa la propria memoria locale durante la computazione
- L'insieme dei processi puo' risiedere sulla stessa macchina o su piu' macchine diverse interconnesse tramite un network di comunicazione
- La comunicazione inter-processo avviene attraverso lo scambio esplicito di messaggi
- Lo scambio dati e' un'operazione cooperativa tra processi; per esempio, ad un'operazione di send da parte di un processo deve corrispondere un'operazione di receive da parte di un altro processo
- L'implementazione del modello di programmazione message passing generalmente richiede una libreria di funzioni da chiamare all'interno del proprio codice

Data Parallel

- Un set di task lavora collettivamente sullo stesso set di dati
- Ogni task lavora su una porzione diversa dalla stessa struttura dati
- Tipicamente ogni task esegue la stessa operazione sulla propria partizione di dati; ad esempio “aggiungi 4 ad ogni elemento dell’array A”
- Sulle architetture shared memory i task accedono alla struttura dati attraverso la memoria condivisa
- Sulle architetture distributed memory la struttura dati e' divisa in “chunk” e ciascuno di essi risiede nella memoria locale al task
- Linguaggi con costrutti data parallel: Fortran 95 e HPF (High Performance Fortran)

Ibridi

Possono essere combinati insieme due o piu' modelli di programmazione parallela.

Ad esempio la combinazione, su architetture ibride, di un modello Shared Memory (OpenMP) con un modello Message Passing (MPI):

- OpenMP all'interno del nodo
- MPI all'esterno del nodo

MPI

Primo esempio di MPI (Esercizio: provate a parallelizzare, usando lo stesso schema dell'esempio la somma dei primi n numeri naturali)