Università degli studi di Napoli Federico II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELLINFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Parallel and Distributed Computing
Elaborato 1

Professore:
Giuliano Laccetti

Matricola:
Alessandro Schiavo
N97000423

ANNO ACCADEMICO 2022 / 2023

Indice

Definizione ed analisi del problema

1.0.1 Analisi ambiente

Si intende definire un algoritmo in grado di utilizzare appieno i calcolatori MIMD a memoria distribuita per eseguire una somma di numeri reali. I calcolatori MIMD (Multiple Instruction Multiple Data) a memoria distribuita si basano su architetture a memoria condivisa virtuale dove le diverse unità di elaborazione eseguono istruzioni su dati diversi. Tali unità sono associate ad una memoria non condivisa, mentre i dati da condividere sono trasmessi con messaggi sincroni e asincroni. L'architettura MIMD quindi presenta caratteristiche ottimali per un ambiente di calcolo parallelo dove :

- le memorie non condivise non presentano problemi di sincronizzazione
- solo i dati da condividere sono trasmessi
- facile modifica del numero di unità di elaborazione

Al contempo gli svantaggi sono multipli e derivano dal tipo di architettura che demanda molteplici responsabilità allo sviluppatore, come: bilanciare il carico di lavoro tra i vari nodi (coppia memoria-unità) e distribuire i dati necessari per l'elaborazione. L'operazione deve quindi sfruttare l'ambiente di sviluppo individuando la soluzione migliore in base ai nodi presenti e agli input forniti. In ambiente parallelo la complessità di tempo non è in grado di misurare l'efficienza degli algoritmi al variare del numero di processi, dato che non è proporzionale al numero di passi compiuti. Difatti ogni operazione è scomponibile in una componente sequenziale e una parallelizzabile.

$$T(n) = T_s + \frac{T_c}{p} + T_o(p) \text{ , con } T_o > 0 \text{ se } p > 1$$

Dove T_s risulta l'insieme delle operazioni esclusivamente sequenziali, T_c l'insieme delle istruzioni eventualmente simultanee sul numero di processi p e

$T_o(p)$ il costo della comunicazione che è strettamente correlato al numero di processi.

Dall'analisi dei tempi si evidenzia che al crescere del numero di processi, a causa della presenza di $T_o(p)$, non necessariamente corrisponde una riduzione del tempo di esecuzione totale. Risulta quindi inevitabile analizzare l'efficienza con strumenti adeguati all'ambiente parallelo.

La funzione *speed-up* $S(p)$ indica la riduzione del tempo di esecuzione da sequenziale a parallelo dimostrando quanto l'implementazione si presti alla parallelismo; mentre la funzione di *efficienza* $E(p)$ indica la percentuale con la quale l'implementazione impiega le risorse a disposizione.

1.0.2 Caratteristiche del problema

In particolare, l'implementazione deve leggere i seguenti input:

- N : numero di elementi da sommare :
  - se $N > 20$ l'algoritmo genera un insieme di numeri reali di cardinalità N oppure
  - se $N \leq 20$ l'algoritmo legge i numeri in input
- P strategia di comunicazione tra processi da applicare
- ID identificativo del processo che stampa il risultato :
  - se $ID = -1$ tutti i processi stampano il risultato
  - se $0 \leq ID \leq (P - 1)$ il processo indicato stampa il risultato

Definizione dell'agoritmo

L'operazione somma di numeri reali presenta una struttra del tipo : lettura degli addendi e un ciclo di somme parziali fino ad ottenere il risultato finale. L'operazione assume dunque le caratteristiche di un albero binario, dove le foglie sono gli addendi e i nodi dello stesso livello sono somme parziali ricavate dal livello precedente; segue che processi diversi possono eseguire rami diversi dell'albero e scambiare le informazioni solo quando necessario.

La comunicazione tra processi è gestita con la libreria MPI ed è impiegata per inizializzare l'ambiente per i processi disponibili e lo scambio di messaggi secondo varie strategie di comunicazione.

Di seguito i macro passaggi che l'algoritmo compie.

```
somma() {
  exit_status = check_input(...);

  if (exit_status != 0) {
    exit(exit_status);
  } else if (num_processi == 1) {
    sequenziale(...);
  }
  numero_elementi_processo = calculate_elem_proc(...);
  parse_input(...);
  buffer = local_calculation(...);
  communication_strategy(...);
  print_result(...);
}
```

La funzione `communication_strategy()` definisce la strategia da utilizzare secondo l'input P e di conseguenza tutti i processi, dopo aver calcolato la propria somma parziale, eseguendo scambi di messaggi secondo tre tipi di implementazione.

2.0.1 Strategia I

La prima strategia demanda ad un singolo processo il compito di sommare le somme parziali dei vari processi. Ponendo come processo delle somme parziali $id = 0$, si ottiene che:

- i processi con $1 \leq id \leq (p-1)$ restano inutilizzati durante l'esecuzione dei livelli $l : 0 \leq l \leq (h-1)$
- a fine computazione solo il processo $id = 0$ contiene il risultato finale, aggiungendo un ulteriore scambio di messaggi in caso di lettura del risultato da altri processi
- al variare del numero di processi con numero di elementi fisso, l'altezza dell'albero rimane costante poichè la componente T_c parallelizzabile racchiude solamente il livello $l = h$

2.0.2 Strategia II

Per la seconda strategia i processi sono posti in gruppi di due e generalmente il primo riceve la somma parziale s_1 del secondo e comunica ad un altro gruppo di processi la somma parziale $s_0 + s_1$. L'albero che si ottiene è un albero bilanciato quindi con altezza h minima ottenibile (log_n).

Come per la prima strategia, il risultato è memorizzato solo in un processo.

2.0.3 Strategia III

La rappresentazione grafica della terza strategia è identica alla seconda, questo perchè le somme parziali sono calcolate secondo lo stesso principio ma con l'aggiunta che sono eseguite ulteriori comunicazioni tra rami diversi dell'albero, in modo tale che ogni foglia (processo) possegga il risultato totale.

Input e Output

Indicatori di errore

Sono presenti diversi indicatori di errore relativi al numero di input fornito, alla qualità dei valori e applicati eventuli controlli di relazione tra di essi.

Come indicato alla sezione1.0.2 gli input sono :

1. N : numero di elementi da sommare
2. p : strategia di comunicazione tra processi da applicare
3. id : identificativo del processo che stampa il risultato :
4. $s_1, s_2, .., s_N$ valori d'input se $N < 21$

Errori numero elementi

Verifica	**Descrizione**
`N > 0`	Verifica numero di elementi maggiore di zero
Posto `M` il numero di elementi effettivamente passati in input: `N != M AND N < 21`	Se indicato $N < 21$ è comito dell'utente fornire in input lo stesso numero di numeri reali
Posto `R` il numero di processori: `(N/2) < R`	Devono esser presenti almeno due numeri reali per processo

Tabella 4.1: Indicatori di errore per il numero di elementi

Errori strategia

Verifica	Descrizione
`P < 1 OR P > 3`	Verifica numero sia specificata la prima , seconda o terza strategia
Posto `P` il numero di processi e `IS_POW(P)` uan funzione che determina le potenze di due: `S != S1 AND IS_POW(P)`	Se la strategia è la seconda o la terza, il numero di processi deve esser potenza di due

Tabella 4.2: Indicatori di errore per strategia

Errori identificativo

Verifica	Descrizione
`ID < -1 OR ID > (P-1)`	Posto `P` il numero di processi, determina che l'id indicato non sia al di fuori del range

Tabella 4.3: Indicatori di errore identificativo

Errori identificativo

I *warnings* sono dei controlli che l'algoritmo esegue e il fallimento non ne implica la terminazione, ma eventualmente degli input sono modificati per proseguire col calcolo.

Verifica	Descrizione
Posto `P` il numero di processi, `P = 1`	Avvisa a video che il calcolo sarà eseguito sequenzialmente
Posto `S3` come terza strategia: `S \neq S3 AND (ID \neq 0 OR ID = -1)`	Se la prima e seconda strategia solo il primo processo contiene il risultato da stampare, per tale motivo se si è in una di queste due strategie e si è indicati un processo diverso dal primo o si è indicati una stampa di tutti i processi, allora `ID = 1`

Tabella 4.4: Indicatori di errore identificativo

Subroutine

Analisi dei tempi

Esempi d'uso

Appendice