Parallel and Distributed Computing

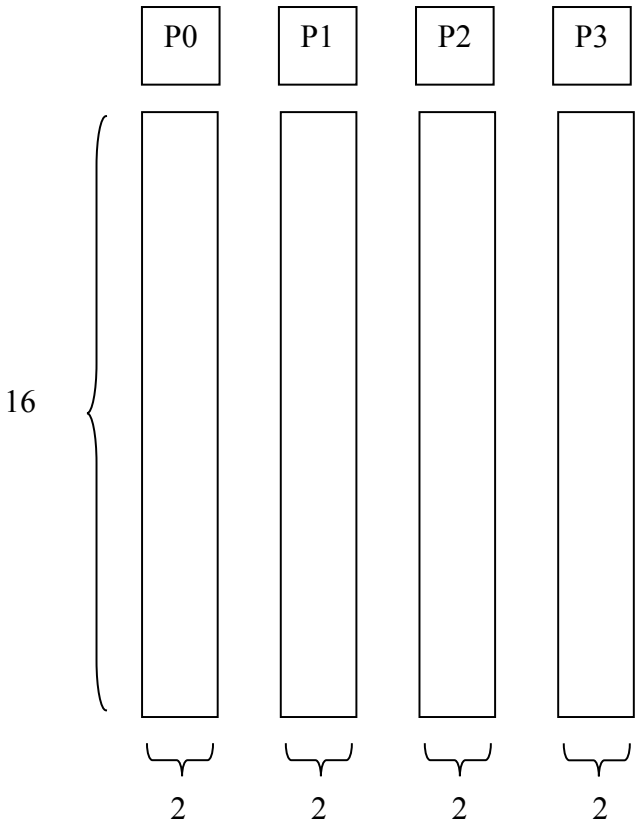
Prova di Autovalutazione - 20DIC2023

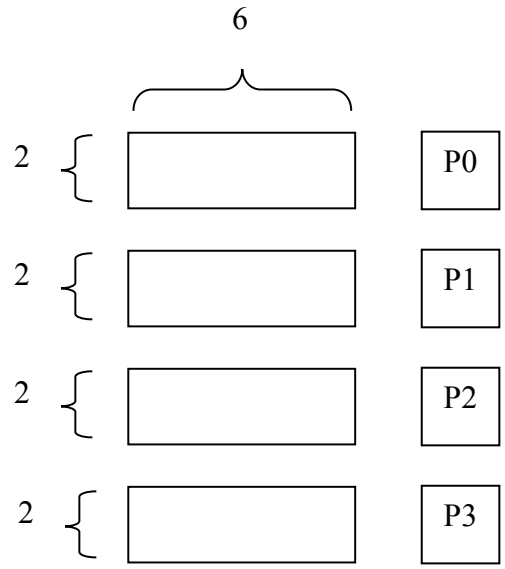
Prof. G. Laccetti

(A.A. 2023/2024)

POSSIBILE SOLUZIONE

1. ***Si consideri il prodotto matrice per matrice*** $A \cdot B = C$ ***, con*** $A \in \Re^{16\times 8}, B \in \Re^{8\times 6}$ ***, su un'architettura MIMD-DM, costituita da 4 processori. La matrice A è distribuita per blocchi di colonne, mentre la matrice B è distribuita per blocchi di righe.***
   a. ***Con i dati così distribuiti cosa possono calcolare i singoli processori?***

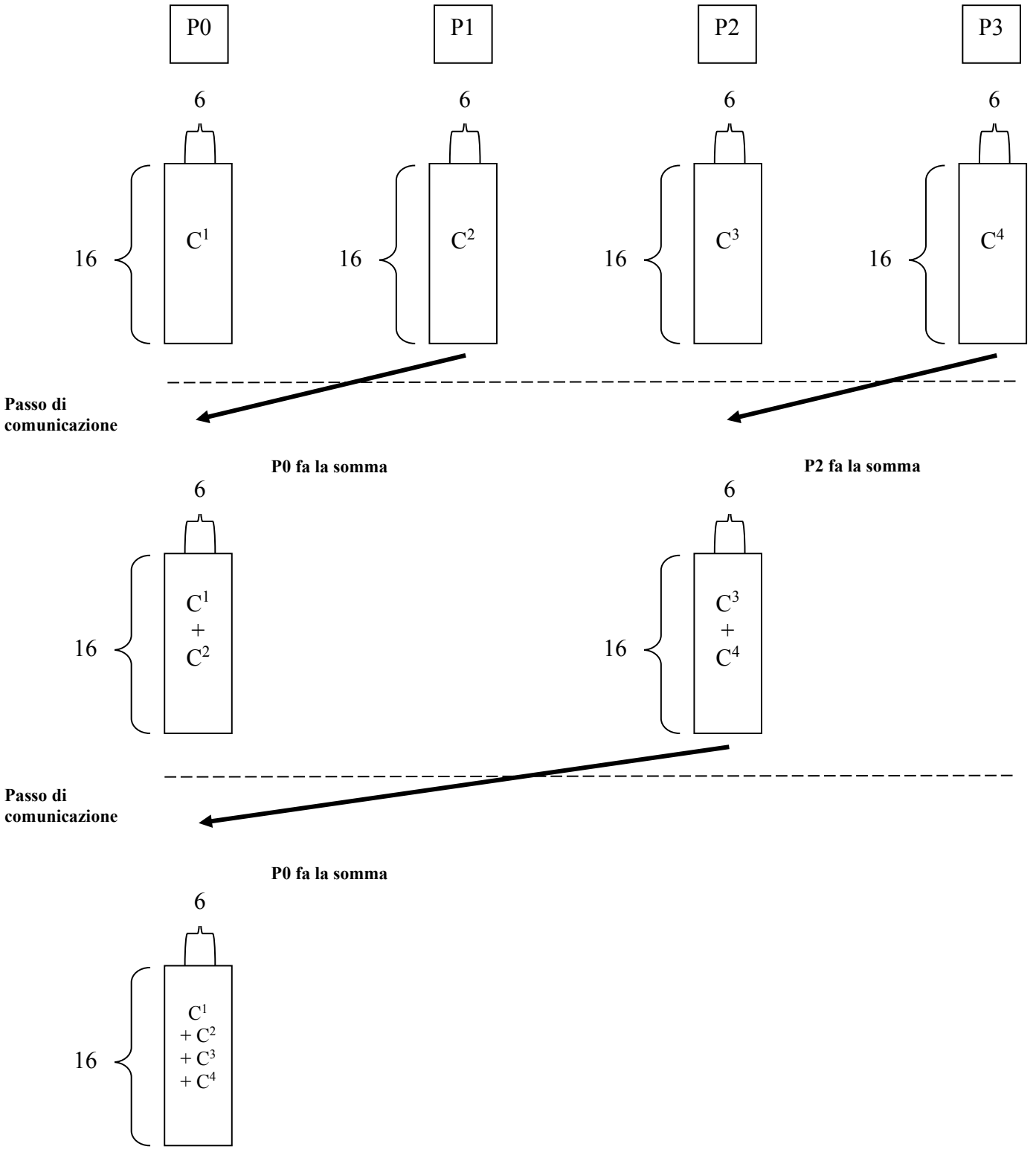




Notiamo che la matrice C risultato del nostro prodotto deve avere dimensione 16x6. Con i dati distribuiti in questo modo ogni processore può calcolare una matrice 16x6, cioè può calcolare un contributo per ogni elemento della matrice C.

b. ***Dopo la fase di calcolo, avvenuta al primo passo dell'algoritmo parallelo, occorre effettuare delle comunicazioni per ottenere il risultato finale? Il candidato illustri uno schema di comunicazione possibile giustificando la sua scelta.***

Avendo al primo passo ogni processore calcolato un contributo per ogni elemento della matrice C, perché il risultato sia completo tali contributi dovranno essere sommati elemento per elemento. Perché questo possa avvenire sono dunque necessarie delle comunicazioni. Un possibile schema è quello ad albero:



In questo modo, in soli $2=\log_2 4$ passi di comunicazione abbiamo calcolato il risultato, che sarà contenuto in P0 e potrà essere stampato. Se fosse stato necessario che tutti i processori avessero il risultato finale nella propria memoria, o se fosse stato possibile fare la somma delle matrici, per qualche motivo, in un solo processore, avremmo usato altre strategie di comunicazione.

c. ***Si calcoli speed-up ed efficienza, secondo la definizione classica.***

Prima di tutto osserviamo che, in caso di un prodotto matrice-matrice eseguito in sequenziale su un solo processore avremo il seguente tempo di calcolo:

$T(1) = (8t_{calc}+7t_{calc})*16*6 = 15t_{calc}*16*6 = 1440t_{calc}$

Infatti:

- Si deve fare il prodotto scalare di ogni riga di A per ogni colonna di B: 16*6 prodotti scalari.
- Ogni prodotto scalare comporta: 8 prodotti e 7 somme.
- Supponiamo che ogni operazione elementare (somma o prodotto) venga eseguita in un tempo t_{calc} che consideriamo come unità di tempo.

Calcoliamo ora invece il tempo che impiegano i 4 processori lavorando in parallelo.
Sappiamo che questo tempo si descrive come

$T(4) = TCALC + TCOM$

Cominciamo con l'osservare quante operazioni per ogni processore (correntemente agli altri) nell'algoritmo che abbiamo descritto, con lo schema di comunicazione scelto:

- Ogni processore deve eseguire 16*6 prodotti scalari
- Ogni prodotto scalare (nel singolo processore) richiede 2 prodotti e 1 somma.
- Dopo il primo passo di comunicazione il processore P0 e il processore P2 dovranno eseguire 16*6 somme.
- Dopo il secondo passo di comunicazione il processore P0 dovrà eseguire 16*6 somme.

Allora:

$$TCALC = \underbrace{(2t_{calc}+1t_{calc})*16*6}_{\text{Prodotti scalari}} + \underbrace{2*(16*6t_{calc})}_{\text{Somma delle matrici}} =$$

$$= 3t_{calc}*16*6 + 2*96t_{calc} = 288t_{calc}+192t_{calc} = 480t_{calc}$$

Per quanto riguarda la comunicazione, sappiamo che verranno fatti 2 passi di comunicazione:

- al primo passo due processori inviano 16*6 elementi ad altri due processori
- al secondo passo un processore invia 16*6 elementi ad un altro processore

Dunque, se consideriamo t_{com} l'unità di tempo di comunicazione:

TCOM = $2*16*6t_{com} = 192t_{com}$

Se poniamo $t_{com}=2*t_{calc}$, possiamo calcolare finalmente:

$T(4) = 480t_{calc} + 384t_{calc} = 864t_{calc}$

Ora abbiamo tutti gli elementi per calcolare lo Speed-Up

$Sp(4) = T(1)/T(4) = 1440t_{calc}/864t_{calc} = 1,7$

Mentre l'efficienza è:

$E(4) = Sp(4)/4 = 1,7/4 = 0,425$

---

2. ***Si consideri il problema della somma di N=23 numeri, su un'architettura MIMD-DM, costituita da 8 processori.***
    a. ***Calcolare lo speed-up, secondo la legge di Ware, per le 3 principali strategie di comunicazione.***

Osserviamo che, perché l'algoritmo sia bilanciato, i numeri saranno così distribuiti:

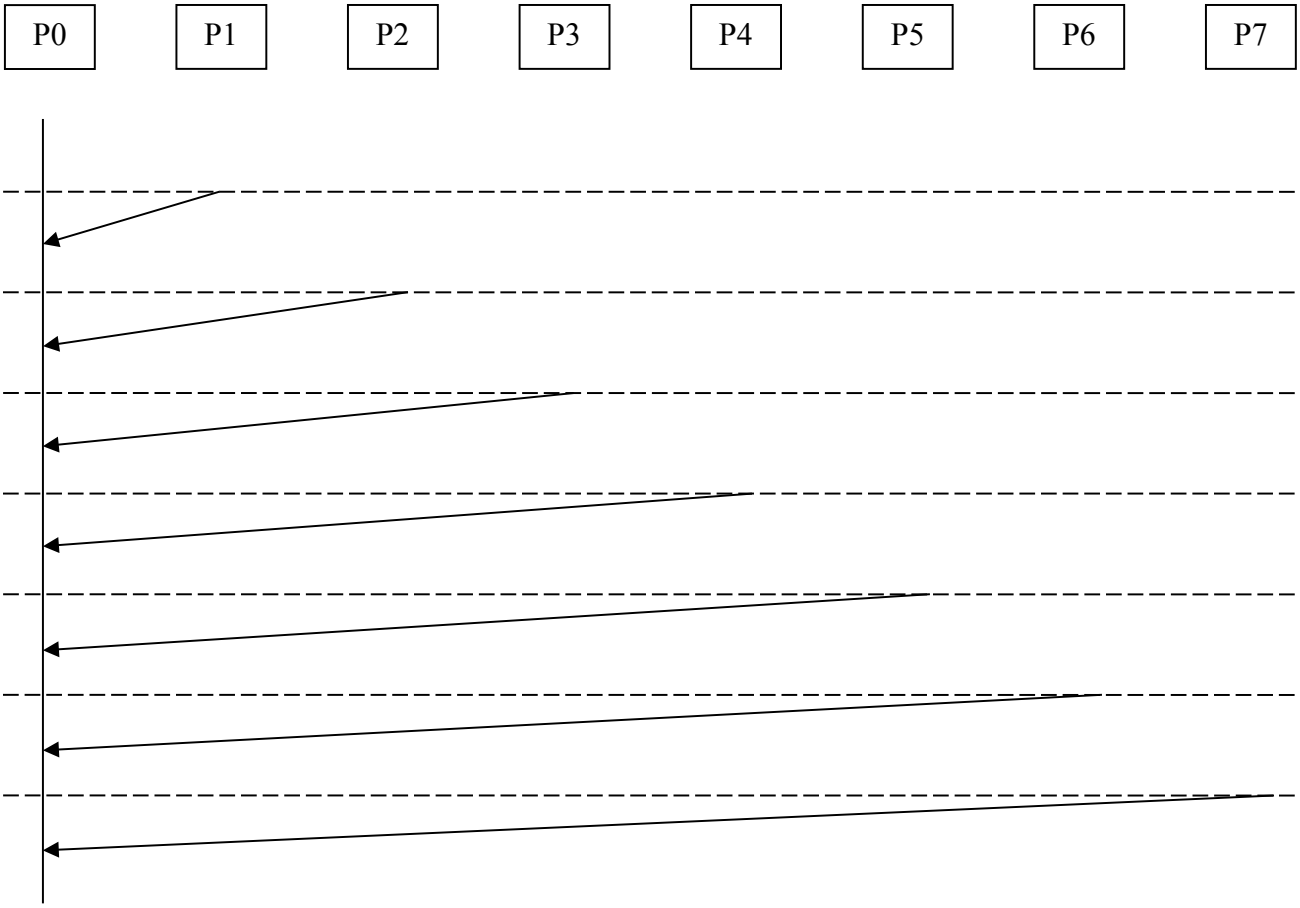
- I processori P0-P1-P2-P3-P4-P5-P6 avranno (considerando la divisione intera) 23/8+1=3 numeri
- Il processore P7 avrà (considerando la divisione intera) 23/8=2 numeri.

La legge di Ware che dobbiamo applicare ci dice che:

$$S(p) = p \cdot E(p) = \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

Dove α è la parte non parallelizzabile dell'algoritmo, e quindi $1-\alpha$ la frazione di operazioni che avvengono in parallelo.
Innanzitutto, dobbiamo identificare α, per ogni strategia.

Se utilizziamo la prima strategia di comunicazione:



- al primo passo di calcolo i processori eseguiranno concorrentemente 2 somme (P7 aspetterà dopo aver fatto la propria singola somma).
- per i successivi 7 passi di calcolo, il processore P0 eseguirà una somma.
- Il totale delle somme fatte è 15+7=22

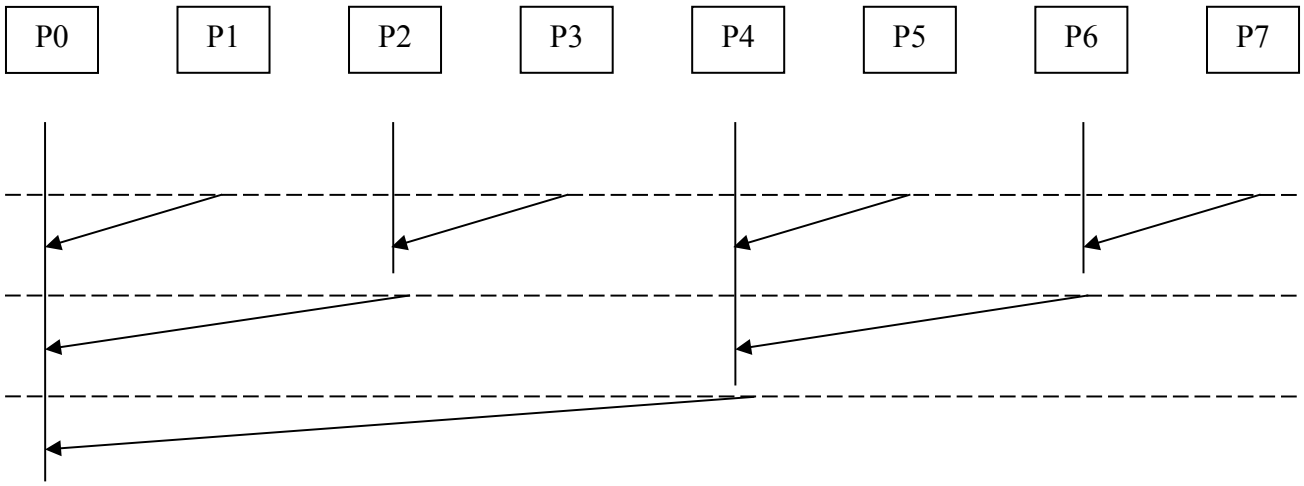
Quindi verranno eseguite in parallelo 15 somme ed in sequenziale 7 somme:

- $\alpha=7/22$
- $1-\alpha=15/22$

Da cui:

$$S(8)=\frac{1}{\frac{7}{22}+\frac{\frac{15}{22}}{8}}=\frac{1}{0{,}32+\frac{0{,}68}{8}}=\frac{1}{0{,}32+0{,}08}=\frac{1}{0{,}40}=2{,}5$$

Se invece utilizziamo la seconda strategia di comunicazione:



- al primo passo di calcolo i processori eseguiranno concorrentemente 2 somme (P7 aspetterà dopo aver fatto la propria singola somma).
- Al secondo passo di calcolo P0, P2, P4, e P6 eseguiranno con correntemente 1 somma
- Al terzo passo di calcolo P0 e P4 eseguiranno 1 somma
- Al quarto passo di calcolo P0 eseguirà 1 somma.
- Il totale delle somme fatte è 15+7=22

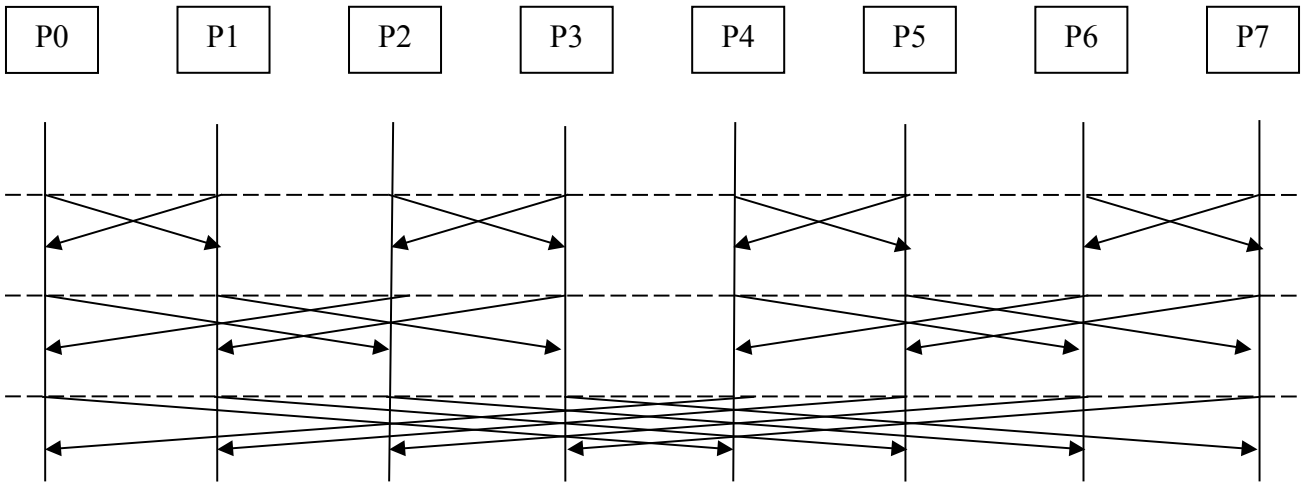
Quindi verranno eseguite in parallelo 21 somme ed in sequenziale 1 sola somma:

- α=1/22
- 1-α=21/22

Da cui:

$$S(8) = \frac{1}{\frac{1}{22} + \frac{\frac{21}{22}}{8}} = \frac{1}{0{,}045 + \frac{0{,}95}{8}} = \frac{1}{0{,}045 + 0{,}12} = \frac{1}{0{,}165} = 6{,}06$$

Se invece utilizziamo la terza strategia di comunicazione:



- al primo passo di calcolo i processori eseguiranno concorrentemente 2 somme (P7 aspetterà dopo aver fatto la propria singola somma).
- Al secondo passo di calcolo tutti i processori eseguiranno concorrentemente 1 somma
- Al terzo passo di calcolo tutti i processori eseguiranno concorrentemente 1 somma
- Al quarto passo di calcolo tutti i processori eseguiranno concorrentemente 1 somma.
- Il totale delle somme fatte è 15+3*8=39.

Quindi verranno eseguite in parallelo 15+3*8=39 somme:

- $\alpha=0$
- $1-\alpha=39/39=1$

Da cui:

$$S(8) = \frac{1}{0 + \frac{1}{8}} = \frac{1}{\frac{1}{8}} = 8$$

Appare chiaro che la seconda strategia porta ad uno speed-up che si avvicina molto allo speed-up ideale, ma la terza strategia lo raggiunge esattamente. Tuttavia, tra la seconda e la terza strategia c'è una differenza in termini di numero di operazioni fatte: il numero maggiore di operazioni proprio della terza strategia di comunicazione può non essere sempre necessario, ma permette un migliore sfruttamento dell'ambiente parallelo, soprattutto considerando che se vogliamo il risultato in tutti i processori, avendo scelto una delle prime due strategie di comunicazione, alla fine dovremmo aggiungere ulteriori passi.

3. Si consideri la seguente routine della libreria MPI:
MPI_Reduce(void *operand, void *result, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_comm comm.).

a. Cosa è possibile effettuare utilizzando tale routine?

Tramite questa funzione si può realizzare un'operazione collettiva scelta (per esempio la somma dei valori contenuti in una determinata variabile presente in tutti i processori), facendo in modo che collaborino alla realizzazione tutti i processori, in maniera ottimizzata.

b. Descrivere, in dettaglio, i parametri.

- void *operand: variabile presente in tutti i processori, contenente i valori da combinare
- void *result: variabile presente nel processore root che conterrà il risultato dell'operazione
- int count: intero rappresentante la dimensione di operand (uguale per tutti i processori)
- MPI_Datatype datatype: tipo di operand e quindi di result
- MPI_Op: op operazione che si intende realizzare. Per esempio, in caso si tratti di somma, sarà MPI_SUM.
- int root: identificativo del processore che conterrà il risultato
- MPI_comm comm: communicator che comprende tutti e soli i processori che devono partecipare all'operazione

4. Si consideri un'architettura parallela di tipo MISD

a. ***Indicare il significato ed il tipo di parallelismo che è possibile implementare per tale classe.***

MISD è l'acronimo di Multiple Instruction Single Data, e si usa per intendere la classe delle architetture in cui si possono eseguire flussi di istruzioni diversi sullo stesso dato contemporaneamente. Presa la definizione alla lettera, sembra non esistere né essere immaginabile un'architettura reale che appartenga a questa classe, ma molti fanno rientrare in questa categoria quelle architetture che implementano la pipeline: questo significa che con questo tipo di architetture si può realizzare un parallelismo di tipo temporale, ovvero quello che si può esemplificare in una catena di montaggio. Un esempio di implementazione della pipeline in architettura lo troviamo spesso (oggi ormai possiamo dire sempre) a livello di unità aritmetiche (ALU) del processore: l'ALU è generalmente divisa in segmenti, ognuno preposto ad una specifica operazione elementare, e quindi in grado di lavorare concorrentemente agli altri.

b. ***Specificare la differenza tra questo tipo di architettura e quella di tipo SIMD.***

SIMD è l'acronimo di Single Instruction Multiple Data, e si usa per intendere la classe delle architetture in cui si può eseguire lo stesso flusso di istruzioni contemporaneamente su dati diversi. Dunque, a questa categoria appartengono quelle architetture in cui ad una sola unità di controllo (CU) corrispondono più unità aritmetiche (ALU), che agiscono su dati diversi: l'istruzione decisa dalla CU viene eseguita contemporaneamente dalle diverse ALU, ognuna sul proprio insieme di dati. Dunque, con questo tipo di architetture si può realizzare un parallelismo di tipo spaziale. Eventualmente le macchine pipelined possono essere fatte rientrare anche in questa categoria, ed se teniamo conto di questo con le architetture SIMD possiamo realizzare anche parallelismo di tipo temporale.