Corso di Calcolo Parallelo e Distribuito mod. A

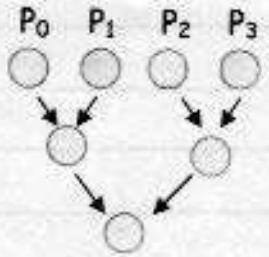
Esame

Prof. G. Laccetti (A.A. 2010/2011 – 26/01/2011)

1. Si consideri il prodotto matrice per vettore $A \cdot b = c$, con $A \in \Re^{16\times8}$, $b \in \Re^{8}$ su un'architettura MIMD-DM, costituita da 4 processori, disposti secondo una griglia bidimensionale 2×2. La matrice A è distribuita per blocchi di righe e di colonne.
   a. Che dimensione hanno i blocchi A_{loc} e come deve essere distribuito il vettore b?
   b. Si descriva l'algoritmo relativo alla risoluzione del problema.
   c. Si calcolino speedup ed efficienza, secondo la definizione classica.

2. Si enunci la Legge di Ware Generalizzata, descrivendo nel dettaglio tutti i parametri.

3. Si consideri l'algoritmo della somma di N=64 numeri in parallelo, con P=4 processori, schematizzato come segue:



   a. Si calcoli il valore dello Speed-up utilizzando sia la definizione classica ($T_{com}+T_{calc}$) sia la legge di Ware.
   b. Si calcoli l'Overhead di comunicazione.

4. Si consideri la seguente routine della libreria MPI:
   MPI_Bcast (void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm).
   a. Cosa è possibile effettuare utilizzando tale routine?
   b. Descrivere, in dettaglio, i parametri.

5. Si descrivano le problematiche dell'algoritmo della somma di N numeri su un'architettura MIMD a memoria condivisa, spiegando come si risolvono utilizzando OpenMP.

6. Si consideri la classificazione di Flynn relativamente alle architetture parallele.
   a. Illustrare la differenza fra le sottoclassi MIMD-DM e MIMD-SM.

Corso di Calcolo Parallelo e Distribuito mod. A

Esame

Prof. G. Laccetti (A.A. 2010/2011 – 22/02/2011)

1. Si consideri il prodotto matrice per vettore $A \cdot b = c$, con $A \in \Re^{24\times12}$, $b \in \Re^{12}$ su un'architettura MIMD-DM, costituita da 4 processori. La matrice A è distribuita per blocchi di colonne.

- **a.** Che dimensione hanno i blocchi A_{loc} e come deve essere distribuito il vettore b ?
- **b.** Si descriva l'algoritmo relativo alla realizzazione del prodotto, affinchè ogni processore abbia l'intero vettore risultato c.
- **c.** Si calcolino speed-up ed efficienza, secondo la legge di Ware-Amdhal generalizzata.

2. Si consideri il prodotto Matrice-Matrice, da eseguire su un'architettura MIMD a memoria distribuita. Si descrivano due possibili strategie di approccio alla soluzione del problema, tenendo conto della distribuzione dei dati e della ricompilazione del risultato finale, evidenziando le differenze e le analogie tra i due approcci scelti.

3. Si enuncino le definizioni di:

- Efficienza
- Overhead Totale

e si esprima la relazione tra di essi, riscrivendo la prima in funzione del secondo.

4. Si consideri la seguente routine della libreria MPI:

MPI_Cart_create (MPI_Comm comm_old, int dim, int *ndim, int *period,
int *reorder, MPI_Comm *new_comm).

- **a.** Cosa è possibile effettuare utilizzando tale routine?
- **b.** Descrivere, in dettaglio, i parametri.

5. Si consideri l'API OpenMP.

- **a.** Si descrivano i costrutti di Work-Sharing.
- **b.** Si descrivano tre clausole che siano applicabili ognuna ad almeno un costrutto descritto al punto precedente.

6. Si descrivano i tipi di parallelismo studiati, evidenziando le differenze e servendosi di esempi.

Corso di Calcolo Parallelo e Distribuito mod.A

Esame

Prof. Giuliano Laccetti (A.A. 2015/2016 – 25/01/2016)

1. Si enunci la legge di Ware-Amdhal, descrivendo nel dettaglio tutti i parametri;
2. Si consideri la somma di 642 numeri con 8 processori:
    a. Si consideri la 2°strategia:
        i. Spiegare a parole (facendosi aiutare da uno schema) la strategia;
        ii. Calcolare lo Speed-up con la legge di Ware generalizzata;
    b. Si consideri la 3°strategia:
        i. Spiegare a parole (facendosi aiutare da uno schema) la strategia;
        ii. Calcolare lo Speed-up con la legge di Ware generalizzata;
3. Come si crea una regione parallela con OpenMP?
4. Si descrivano i tre tipi di parallelismo studiati, evidenziando le differenze e servendosi di esempi.

[AUTOVALUTAZIONE]

1- Si consideri il prodottto matrice x matrice A*B=C, con A € R^{16x8}, B € R^{8x6}, su un'architettura MIMD-DM costituita da 4 processori. La matrice A è ditribuita per blocchi di colonne, mentre B è distribuita per blocchi di righe.

a) Con i dati così distribuiti cosa possono calcolare i singoli processori?

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

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

a: I singoli processori possono calcolare una matrice 16x6 (16=righe matrice A e 6=colonne della matrice B) , cioè un contributo per ottenere il risultato finale che sarà ottenuto sommando tutti i contributi.

b: In teoria la seconda strategia (schema di comunicazione ad albero) sarebbe la migliore delle 3, poiché abbiamo a disposizione 4 processori. => i passi di comunicazione, con la seconda strategia, sono $\log_2 4$. Se volessimo conservare il risultato finale su tutti i processori allora ci converrebbe usare la terza strategia. Quest'ultima funziona in modo simile alla prima, solo che la differenza sostanziale è che ad ogni passo di comunicazione per ogni coppia i processori si scambieranno le somme parziali bilateralmente.

c: **speed-up S = T(1) / T(P)**. T(1) è il tempo impiegato dall'algoritmo con l'utilizzo di un solo processore. **T(1)** = 16*6 * (8+7)t_calc. (perché abbiamo 8 prodotti e 7 addizzioni, è un prodotto scalare 8x8.) dove t_calc = tempo che la macchina impiega per fare un operazione.
T(4) = t_comm + t_calc
t_calc = prodotto locale + somma dei contributi.
Prodotto locale = (16*6 * (2+1))t_calc (perché ci sono 2 prodotti e 1 somma, perché ogni processore moltiplica una matrice 16x2 ad una 2x6, essendo 4 i processori.).
Somma dei contributi = ($\log_2 4$ * (16*6))tcalc (vengono sommate matrici i16*6 ogni passo di comunicazione)
t_calc = (16*6 * (2+1)) + ($\log_2 4$ * (16*6)t_calc = (288 + 192)t_calc = 480t_calc
t_comm = passi di comunicazione * dimensione dato inviato * 2 (1 t_comm = 2 t_calc)
t_comm = $\log_2 4$ * (16*6) * 2tcalc = 384t_calc

T(4) = (480+384)t_calc = 864 t_calc
Speed-up S = 1440t_calc / 864t_calc = 1.7
Efficienza E = S(p) / P = 1.7 / 4 = 0.425

2- Si consideri il problema della somma di N=23 numeri su archiettura MIMD-DM costituita da 8 processori.

a) Calcolare speed-up, secondo la legge di Ware, per le 3 principali strategie di comunicazione.

a: inizialmente suddividiamo i numeri tra gli 8 processori, 23/8 quindi diamo 3 numeri ad ogni processore tranne ad uno a cui ne daremo solo due.

Lo Speed-Up secondo la legge di Ware viene calcolato nel seguente modo:

S(p) = $p * E(p) = \frac{1}{\alpha+\left(\frac{1-\alpha}{p}\right)}$

Dove $\boldsymbol{\alpha}$ è la parte di algoritmo non parallelizabile e **1 -** $\boldsymbol{\alpha}$ è la parte di algoritmo parallelizabile.

Calcoliamo S(p) secondo la **prima strategia di comunicazione**:
al primo passo di calcolo ogni processore esegue concorrentemente 2 somme, per quelli successivi il processre P0 eseguirà una sola somma. Il totale quindi è 15+7 = 22 somme, quindi verranno eseguite in parallelo 15 somme ed in sequenziale 7 somme.

Quindi $\boldsymbol{\alpha}$ = 7/22 e **1-** $\boldsymbol{\alpha}$ = 15/22.
$S(8) = 1/(\frac{7}{22} + \left(\left(\frac{15}{22}\right) * \left(\frac{1}{8}\right)\right)$ = 1/(0.32+0,085) = 1/0,4 = 2,5

Andiamo a calcolare la **seconda strategia di comunicazione**:
Al primo passo i processori eseguiranno concorrentemente 2 somme (15 somme), tranne P7 che ne fa una.
Al secondo passo di calcolo i processori dispari eseguiranno 1 somma (4 somme), al terzo passo P0 e p4 fanno la somma (perché P6 invia a P4 e P2 invia a P0. 2 somme), al quarto passo P0 esegue l'ultima somma (1 somma)
Il totale è 15+7=22. Quindi verranno eseguite in parallelo 21 somme ed in sequenziale 1 sola somma

Da cui $\boldsymbol{\alpha}$ = 1/22 e **1-** $\boldsymbol{\alpha}$ =21/22
S(8)=$1/(\frac{1}{22} + \left(\frac{21}{22} * \frac{1}{8}\right) = \frac{1}{0.45+0.12} = 6{,}06$

Andiamo a calcolare la **terza strategia di comunicazione**:
Al primo passo tutti fanno due somme tranne una processore, per un totale di 15 somme in parallelo.
Al secondo passo e al terzo passo tutti i processori fanno una somma, cioè 8 somme.
Alla fine, tutti i processori fanno una somma per un totale di 8.
15+(8*3)= **39 somme in totale tutte in parallelo**.

Quindi $\boldsymbol{\alpha} = 0$ e **1-** $\boldsymbol{\alpha}$ = 39/39 = 1

S(8)= 1/ (0 + ((39/39)/8)) = 8

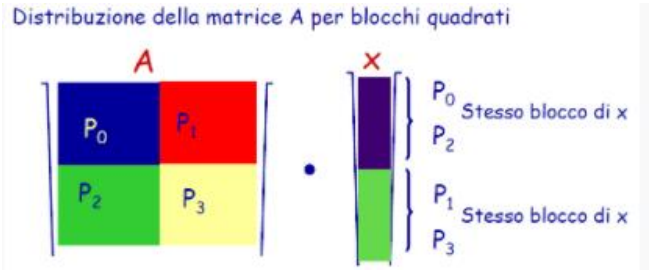
Appare chiaro che la seconda strategia porta ad uno speed-up che si avvicina molto allo speed-up ideale, ma la terza strategia lo raggiunge esattamente. Tuttavia, tra la seconda e la terza strategia c'è una differenza in termini di numero di operazioni fatte: il numero maggiore di operazioni proprio della terza strategia di comunicazione può non essere sempre necessario, ma permette un migliore sfruttamento dell'ambiente parallelo.

[26-01-2011]

1 - Si consideri il prodotto matrice per vettore A*x = c, con A € R^{16x8}, x € R^8 su un'archiettura MIMD-DM costituita da 4 processori, disposti secondo una griglia bidimensionale 2x2. La matrice A è distribuita per blocchi di righe e colonne.

a) Che dimensione hanno i blocchi A_loc e come deve essere distribuito il vettore x?
b) Si descriva l'algoritmo relativo alla risoluzione del problema
c) Si calcolino speed-up ed efficienza secondo la definizione classica

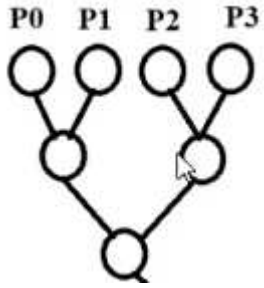
a: I sottoblocchi di matrice locali a ciascun processo avranno dimensione 8x4 (16/2 e 8/2). Il vettore x viene suddiviso in due parti ed affinchè sia possibile eseguire il prodotto locale la prima sarà assegnata alla prima colonna di processori (p0 e p2), la seconda alla seconda colonna di processori (p1 e p3)



b: ciascun processore effettuerà un prodotto matrice vettore tra la sua porzione di A e la sua porzione di x, in tal modo ogni processore avrà calcolato un contributo per il risultato finale sottoforma di vettore di dimensione 8. Per ottenere il risultato finale i processori sulla stessa riga sommeranno elemento per elemento i propri contributi, una volta fatto ciò i risultati saranno concatenati per colonna ottenendo così il risultato finale.

c: **Speed-up S** = T(1)/T(P).
T(1) = 16 * (8+7)t_calc = 240t_calc (8 prodotti e 7 somme).
T(4) = t_comm + t_calc.
t_calc = prodotto locale + somma dei contributi. Al primo passo, ogni processore moltiplica una matrice 8x4 ad un vettore di dimensione 4. => t_calc = 8*(4+3)t_calc = 56t_calc (8 sono le righe, 4 prodotti e 3 somme). Dopo il calcolo dei contributi, sommiamo i contributi: 56t_calc + 8t_calc = 64t_calc (perché sono 8 somme, $log_2 4$=2 moltiplicato r=4 gli elementi del vettore da sommare).
t_comm = dimensione dato * 2 = 8t_comm * 2 = 16t_calc (t_comm = 2t_calc)
T(4) = 64t_calc + 16t_calc = 80t_calc
Speed-up S = 240t_calc / 80t_calc = 3
Efficienza E = S/p = 3/4 = 0.75

2 – Si consideri l'algoritmo della somma di N=64 numeri in parallelo, con P=4 processori schematizzato ad albero.



a) Si calcoli il valore dello speed-up utilizzando sia la definizione classica, sia la legge di Ware
b) Si calcoli l'overhead di comunicazione

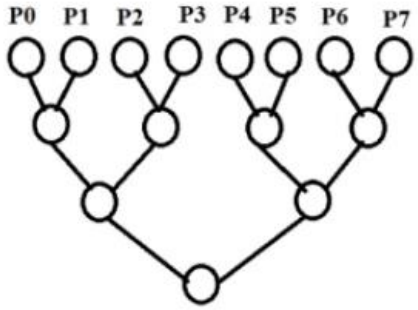
a: T(1) = tempo di esecuzione dell'algoritmo con un solo processore. **T(1)** = (N-1)t_calc = 63t_calc
T(4) = tempo di esecuzione dell'algoritmo con 4 processori. Calcoliamo R = tetto(N/P) = tetto(64/4) = 16. Ciascun processore avrà un numero pari a 16 numeri che dovrà sommare in parallelo agli altri. Supponendo che t_comm = 2t_calc, **T(4)** = (R - 1)t_calc + $(log_2 4)$t_calc + $(log_2 4)$t_comm = 15 + 2 + 4 = 21t_calc.
Speed-up S = T(1)/T(P) = 63t_calc/21t_calc = 3t_calc (quasi speed-up ideale).
Per la legge di Ware, abbiamo che: **Speed-up S** = $\frac{1}{\alpha+\left(\frac{1-\alpha}{p}\right)}$ dove $\boldsymbol{\alpha}$ corrisponde alla parte dell'algoritmo eseguita in sequenziale, e **1 – $\boldsymbol{\alpha}$** la parte dell'algoritmo eseguito in parallelo.
Al primo passo i processori calcolano 60 somme, al secondo passo fanno 2 somme, al terzo passo fanno 1 somma. **Per un totale di 63 somme**. $\boldsymbol{\alpha}$ = 1/63 = 0.0158 e **1- $\boldsymbol{\alpha}$** = 62/63 = 0.984
Speed-up S = 1 / ((1/63) + ((62/63)/4) = 3.82

b: **Overhead di comunicazione Oh** = $T_p$comm / $T_p$calc = 4/17 ($T_p$calc = (R-1)+2, $T_p$comm = $log_2 4$ * 2. **Se la strategia è la seconda o quella ad albero!**) Oh = 0.24

[09-01-2018]

1- Si consideri l'algoritmo della somma di N=62 vettori, definiti in R^4, con P = 8 processori. Si supponga che lo schema di comunicazione sia ad albero.

a) Si calcolino Speed-up ed efficienza secondo la legge di Ware generalizzata
b) Opzionalmente, calcolare il tempo di esecuzione secondo la definizione classica



a: Sia N=62 vettori, definiti in R^4, con P = 8 processori. => $\boldsymbol{r_k}$ = (62/8) -1 se k >= 62%8, altrimenti (62/8 +1) -1 se k < 62%8. Visto che il mdulo è != 0, r_0=8 r_1=8 r_2=8 r_3=8 r_4=8 r_5=8 r_6=7 r_7=7 vettori.

Ogni processore da 0 a 5 avrà 8 vettori, e quindi effettuerà (8-1)*4 somme (4 somme perché ogni vettore è in R^4). Stessa cosa per i processori da 6 a 7, che avranno 7 vettori. Quindi effettueranno (7-1)*4 somme. Al passo 0 avremo un numero di operazioni pari a 6*((8-1)*4) + 2*((7-1)*4) (2* perché sono solo i processori P6 e P7, 6* perché sono solo i processori da P0 a P5). = 168+48 = 216.

Secondo lo schema di comunicazione ad albero, abbiamo un numero di operazioni pari a $log_2 8$ = 3. Al primo passo di comunicazione, lavorano solo 4 processori ed effettuano 4 somme, per un totale di 4*4 = 16. Al secondo passo, lavorano solo 2 processori, per un totale di 8 somme. Al terzo passo lavorano solo 1 processore, per un totale di 4 somme. Quindi abbiamo un totale di opreazioni pari a: 216 + 16 + 8 + 4 = 244.

Legge di Ware generalizzata: $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$

a_1 = operazioni svolte da un solo processore / opereazioni totali = $\frac{4}{244}$ = 0.016

Ora vediamo quali processori lavorano: lo sappiamo perché prima abbiamo visto quali e quanti solo i passi di comunicazione. Sono 2, 4, 8 processori.

$\alpha 2^1 = \frac{8}{244}$ =0.032

$\alpha 2^2 = \frac{16}{244}$ =0.065

$\alpha p = \frac{216}{244}$ =0.885

$s(p) = \frac{1}{0.016 + \frac{0.032}{2} + \frac{0.65}{4} + \frac{0.885}{8}} = \frac{1}{0.15} = 6.7$ (lontano dallo speed-up ideale)

Efficienza E = S(8) / 8 = 6.7/8 = 0.83

b: T(p) = T(8) = ((8-1)*4)t_calc + 4*($log_2 8$)t_calc + 2*($4 * (log_2 8)$)t_calc = 54t_calc

2 - Si consideri l'algoritmo della somma di N=193 numeri in parallelo, con P = 8 processori.

a) Si calcoli il valore dello speed-up ed efficienza con tutte e tre le strategie
b) Si confrontino i risultati e si facciano delle considerazioni

a: **Speed-up S** = T(1)/T(P). Calcoliamo r = N/P = 25. I primi 7 processori prendono 24 numeri, il processore 8 ne prende 25. **T(1)** = tempo di esecuzione dell'algoritmo con un solo processore. => T(1) = (N-1)t_calc = 192t_calc. **T(8)** = tempo di esecuzione dell'algoritmo con 8 processori. Per la **prima strategia**, al primo passo abbiamo (25-1)t_calc. Visto che nella strategia uno abbiamo P-1 passi di comunicazione, => T(p) = (r-1)t_calc + (p-1 * k)t_calc + (p-1 * k)t_comm, con k=1. (k=1 perché non abbiamo vettori). Sia t_comm = 2t_calc, => T(8) = 24t_calc + 7t_calc + (7*2)t_calc = 45t_calc.

Speed-up S = 192t_calc/45t_calc = 4.3. **Efficienza E** = S(8)/P = 4.3/8 = 0.53

Per la **seconda strategia**, **T(1)** = tempo di esecuzione dell'algoritmo con un solo processore. => T(1) = (N-1)t_calc = 192t_calc. T(8) = tempo di esecuzione dell'algoritmo con 8 processori. In questo caso T(8) = (r-1)t_calc + $log_2 8$t_calc + ($log_2 8$)t_comm*2 = 24 + 3 + 6 = 33t_calc.

Speed-up S = 192t_calc/33t_calc = 5.81. **Efficienza E** = S(8)/8 = 0.72

Per la **terza strategia**, **T(1)** = tempo di esecuzione dell'algoritmo con un solo processore. => T(1) = (N-1)t_calc = 192t_calc. T(8) = tempo di esecuzione dell'algoritmo con 8 processori. In questo caso, la stretegia tre è analoga alla seconda strategia.

b: Notiamo che lo speed-up "migliore" è quello che si avvicina allo speed-up ideale. => le strategie 2 e 3 sono le migliori per l'esecuzione dell'algoritmo presentato. Se volessimo conservare il risultato finale in ogni processore => utilizzeremo la terza strategia, cosicchè ad ogni passo di comunicazione per ogni coppia i processori si scambieranno le somme parziali bilateralmente.

[22-02-2011]

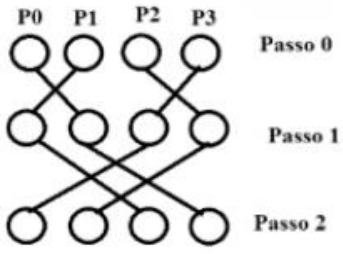
1 - Si consideri il prodotto matrice per vettore A*b = c, con A € R^{24x12}, b € R^{12} su architettura MIMD-DM, costituta da 4 processori. La matrice A è distribuita per blocchi di colonne.

a) Che dimensione hanno i blocchi A_loc e come deve essere distribuito il vettore b?
b) Si descriva l'algoritmo relativo alla realizzazione del prodotto, affinchè ogni processore abbia l'intero vettore risultato c.
c) Si calcoli speed-up ed efficienza secondo la legge di Ware generalizzata

a: Poiché la matrice A è divisa per blocchi di colonne, avremo che ogni processore riceverà n righe (quindi 24 righe) e m/p colonne, con m = colonne di A e p = numero di processori (quindi 12/4 =3. riceve 3 colonne). Quindi A_loc € R^{24x3}. Il vettore b sarà distribuito ai vari processori in m/p parti. Quindi ogni processore riceverà una parte del vettore b € R^3.

b: Una volta che ogni processo avrà ricevuto il suo blocco di A_loc € R^{24x3} e il suo vettore b € R^3, potrà calcolare parallelamente agli altri processori (3 prodotti scalari + 2 somme) tutto * 24 righe. Ogni processore potrà calcolare solo un contributo delle componenti del vettore risultante. Al termine di questi calcoli parziali, è necessario che i processori si scambino i risultati per poterli sommare e ottenere il risultato finale. Poiché si vuole che ogni processore abbia il risultato finale, si può utilizzare la strategia 3, cioè si avrà un numero di passi di comunicazione pari a $log_2 4 = 2$ e la comunicazione delle proprie componenti avverrà per coppie di processori. Ad ogni i-esimo passo di comunicazione i due processori si scambieranno il proprio vettore di componenti per poi poterlo sommare al proprio. Alla fine dei passi avremo che pur avendo eseguito delle operazioni aggiuntive di somma rispetto ad una strategia 2, il risultato del vettore c lo avranno tutti i processori.

c: Al passo 0, ogni processore calcolerà un contributo delle componenti del vettore risultato, quindi effettuerà un numero di operazioni pari a 4*((3+2)*24) (24 sono le righe) = 480 operazioni.
Per ottenere il vettore c risultato in tutti i processori dovremmo effettuare un numero di operazioni pari a $log_2 4 = 2$. Al passo 1 avremo che ogni processo effettuerà 24, per un totale di operazioni pari a 4*24 = 96. Stessa cosa al passo 2. Quindi avremo un totale di operazioni pari a 480+96+96 = 672 operazioni totali



Legge di Ware generalizzata: $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$ dove a_i rappresentano le operazioni eseguite da i processori parallelamente.

Poiché in ogni passo lavorano parallelamente sempre tutti e 4 processori insieme si ha che tutti gli a_i con i != 4 sono pari a 0. Quindi avremo che S(4) = $\frac{1}{1/4}$ = 4. Questo perché a_4 = $\frac{672}{672}$ = 1.

Efficienza E = S(4)/p = 4/4 = 1

2 – Si consideri il prodotto matrice-matrice, da eseguire in parallelo su architettura MIMD a memoria distribuita. Si descrivano due possibili strategie di approccio alla soluzione del problema, tenendo contro della distribuzione dei dati e della ricompilazione del risultato finale, evidenziando le differenze e le analogie tra i due approcci scelti.

Se volessimo calcolare il prodotto di una matrice *A* di dimensione n*m per una matrice B di dimensione m*r su un calcolatore monoprocessore dovremmo effettuare un numero di operazioni pari a *(m prodotti + (m-1) somme)*n*r.* Sfruttando il parallelismo su un calcolatore MIMD con p processori si può dividere il problema in sottoproblemi; ogni sottoproblema viene assegnato a uno dei p processori che lavorerà in parallelo agli altri; in fine si può unire i risultati ottenuti dai p processori in un unico risultato.
Esistono 3 strategie di calcolo delle operazioni parziali e delle comunicazioni di esse, tra i vari Processori. Supponiamo di utilizzare la prima e la seconda strategia:

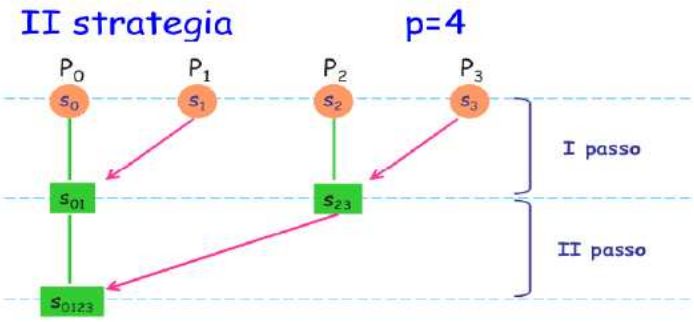
Strategia I
La matrice *A* è divisa per blocchi di righe e la matrice B è divisa per blocchi di colonne, ogni blocco viene assegnato a un processore. Ogni processore avrà, quindi, n/p righe e m colonne di *A* e m righe e r/p colonne di B. A questo punto ogni processore può calcolare, parallelamente agli altri, n/p * r/p prodotti scalari, che corrispondono a n/p * r/p componenti del vettore risultato. A questo punto è indispensabile che i processori si scambino i propri blocchi di colonne di B (ogni processore dovrà inviare il proprio blocco di colonne di B ai p-1 processori rimanenti, per un totale di p-1 passi di comunicazione). Dopo gli scambi, ogni processore sarà in grado di calcolare altre componenti. Al termine ogni processore avrà calcolato n/p * r componenti della matrice risultato.
Al termine è possibile, eventualmente, scambiare le componenti ottenute in modo che uno o tutti i processori abbiano il risultato finale (ovvero tutte le componenti), per un totale di log(p) passi di comunicazione.
Strategia II
La matrice *A* è divisa per blocchi di colonne e la matrice B è divisa per blocchi di righe, ogni blocco viene assegnato a un processore. Ogni processore avrà, quindi, n righe e m/p colonne di *A* e m/p righe e r colonne di B. Così come è diviso il problema ogni processore può calcolare, parallelamente agli altri, solo un contributo per ogni componente del vettore risultato (n*r). Al termine dei calcoli parziali occorre che i processori si scambino i contributi così da poterli sommare e ottenere le componenti del vettore risultato, per un totale di log(p) passi di comunicazione.
Lo scambio e la somma dei contributi possono avvenire in maniera analoga alla strategia II della somma se si vuole che il risultato sia posseduto da un solo processore prestabilito: coppie di processori saranno tali da lavorare in parallelo; per ogni coppia un processore invierà i propri contributi, l'altro riceverà e sommerà i contributi ricevuti ai suoi.
Se si vuole che le componenti finali siano possedute da tutti i processori, lo scambio sarà analogo alla **strategia III** della somma: coppie di processori saranno tali da lavorare in parallelo; in ogni coppia entrambi i processore invieranno i propri contributi all'altro processore, ed entrambi riceveranno e sommeranno i contributi ricevuti ai propri.

[25-01-2016]

1- Si consideri la somma di 642 numeri con 8 processori:

a) Si consideri la 2° strategia:
   - Spiegare a parole (facendosi aiutare da uno schema) la strategia
   - Calcolare lo speed-up con la legge di Ware generalizzata
b) Si consideri la 3° strategia:
   - Spiegare a parole (facendosi aiutare da uno schema) la strategia
   - Calcolare lo speed-up con la legge di Ware generalizzata
c) Si consideri la 1° strategia:
   - Spiegare a parole (facendosi aiutare da uno schema) la strategia
   - Calcolare lo speed-up con la legge di Ware generalizzata

a: Ogni processore calcola la propria somma parziale in parallelo agli altri. La comunicazione delle somme parziali avverranno per coppie di processori; le coppie lavoreranno in parallelo: ad ogni i-esimo passo di comunicazione in ogni coppia un processore riceverà la somma parziale dall'altro che invierà la propria somma parziale. I passi di comunicazione sono pari a $log_2 P$. Al termine, anche in questo caso, solo un processore prestabilito avrà il risultato finale. Rispetto alla strategia I è evidente come i processori vengano sfruttati meglio.

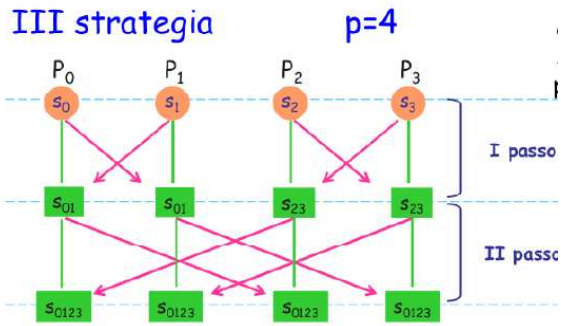


Speed-up secondo la **legge di ware generalizzata**: $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$ dove a_i rappresentano le operazioni eseguite da i processori parallelamente.

Calcoliamo r = N/P = 80.25 => $\boldsymbol{r_k}$ = (642/8) -1 se k >= 642%8, altrimenti (642/8 +1) -1 se k < 642%8. i primi due processori (P0 e P1) avranno 81, i restanti 6 processori avranno 80 numeri da sommare. I passi di comunicazione sono $log_2 8$ = 3. Al passo 0, i primi 2 processori fanno 80 somme e i restanti ne fanno 79 => al passo 0 avremo (80*2)+(79*6) operazioni = 634 operazioni. Al passo 1 l'algoritmo calcola 4 somme, al passo 2 l'algoritmo calcola 2 somme e al passo 3 l'algoritmo calcola 1 somma. => **le operazioni totali sono 634+4+2+1 = 641 operazioni totali**. Ora dobbiamo calcolarci i diversi α per la legge generalizzata di Ware. $\boldsymbol{\alpha_8}$ = T(8)/T(1) = 634/641 = 0.98. $\boldsymbol{\alpha_4}$ = T(4)/T(1) = 4/641 = 0.0062. $\boldsymbol{\alpha_2}$ = 2/641 = 0.0031. $\boldsymbol{\alpha_1}$ = 1/641 = 0.0015.

Per la legge di ware generalizzata: $\frac{1}{0.0015 + \frac{0.0031}{2} + \frac{0.0062}{4} + \frac{0.98}{8}} = 7.9$

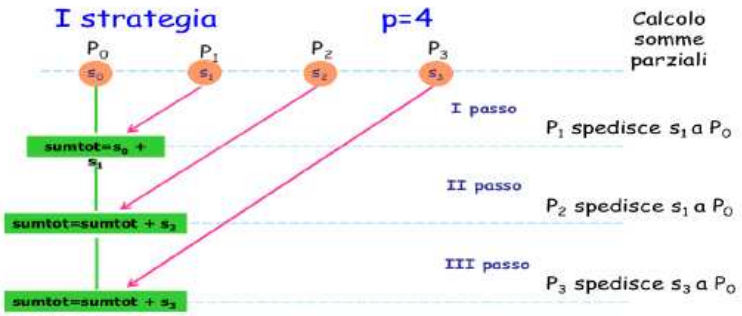
b: Ogni processore calcola la propria somma parziale in parallelo agli altri. Anche in questo caso, come per la strategia II, la comunicazione avverrà tra coppie di processori; le coppie, anche in questo caso, lavoreranno in parallelo. La differenza sostanziale è che ad ogni passo di comunicazione per ogni coppia i processori si scambieranno le somme parziali bilateralmente. I passi di comunicazione, anche in questo caso, sono pari a $log_2 P$. Al termine, quindi, ogni processore avrà la somma finale. E' evidente come in questa strategia, sebbene vengano fatte più operazioni, il vantaggio è che ogni processore ha già il risultato finale e quindi se mai ad uno di questi servisse non ci sarà bisogno di ulteriori comunicazioni.



Speed-up secondo la **legge di ware generalizzata**: $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$ dove a_i rappresentano le operazioni eseguite da i processori parallelamente.

Calcoliamo r = N/P = 80.25 => . i primi due processori (P0 e P1) avranno 81, i restanti 6 processori avranno 80 numeri da sommare. I passi di comunicazione sono $log_2 8$ = 3. Al passo 0, i primi 2 processori fanno 80 somme e i restanti ne fanno 79 => al passo 0 avremo (80*2)+(79*6) operazioni = 634 operazioni. I passi succesivi (passo 1, passo 2, passo 3) fanno 8 somme. Per un totale di operazioni pari a 634+24 = 658. Le operazioni sono tutte in parallelo. => α = 0, e di conseguenza S(8) = 8.

c: Ogni processore calcola la propria somma parziale in parallelo agli altri. Ognuno invierà tale somma parziale ad un unico processore prestabilito, il quale sarà il solo ad avere il risultato finale. I passi di comunicazione sono pari a p-1. E' evidente come in tale strategia non si sfrutta al meglio ogni processore: una volta inviata la propria somma parziale ogni processore terminerà di lavorare.



Speed-up secondo la **legge di ware generalizzata**: $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$ dove a_i rappresentano le operazioni eseguite da i processori parallelamente.

Calcoliamo r = N/P = 80.25 . i primi due processori (P0 e P1) avranno 81, i restanti 6 processori avranno 80 numeri da sommare. I passi di comunicazione sono in questo caso p-1. Al passo 0, i primi 2 processori fanno 80 somme e i restanti ne fanno 79 => al passo 0 avremo (80*2)+(79*6) operazioni = 634 operazioni. Successivamente gli altri 7 passi faranno ognuno 1 somma, per un totale di 7 somme. => le operazioni totali saranno 634+7 = 641. $\boldsymbol{\alpha_8}$ = T(8)/T(1) = 634/641 = 0.99. $\boldsymbol{\alpha_{1\ldots7}}$ = (1/641)*7 = 0.011. => Per la legge di ware generalizzata: $\frac{1}{0.011 + \frac{0.99}{8}}$ = 7.42

[04-02-2019]

1- Si consideri l'algoritmo della somma di N = 122 vettori definiti in R^{10}, con P = 8 processori. Si supponga che lo schema di comunicazione sia ad albero.

a) Si descriva l'algoritmo alla risoluzione del problema
b) Calcolare il tempo di esecuzione dell'algoritmo, lo speed up secondo la legge di ware generalizzata e l'efficienza

a: Ogni processore calcola la sua somma parziale in parallelo. La comunicazione avviene per coppie di processori, le coppie lavorano in parallelo, ad ogni passo di comunicazione ogni coppia di processori riceverà la somma parziale dell'altro che invierà la propria somma parziale. I passi di comunicazione sono uguali a log_2P. Al termine, solo un processore avrà il risultato finale.

b: Dividiamo i vettori per bilanciare l'algoritmo: N/P = 122/8 = 15,25 => i primi 6 processori avranno 15 vettori e i restanti 2 avranno 16 vettori. Al passo 0 avremo 10*(15-1*6) + 10*(16-1*2) = 1140. Al passo 1 saranno attivi solo 4 processori, che invieranno le proprie somme parziali nelle rispettive coppie: => 4*10 = 40. Al passo 2 saranno attivi solo 2 processori che invieranno le proprie somme parziali: => 2*10 = 20. Al terzo passo sarà attivo solo un processore che farà l'ultima somma: => 1*10 = 10.

Secondo la **legge di ware generalizzata**,S (p) = $\frac{1}{a_1 + \sum_{k=2}^{p} \frac{a_k}{k}}$ dove a_i rappresentano le operazioni eseguite da i processori parallelamente. La somma sequenziale dei rispettivi vettori è: 1140+40+20+10 = 1210. Ora calcoliamo i vari α: α_8=1140/1210 = 0.94, α_4=40/1210 = 0.033, α_2=20/1210 = 0.016, α_1=10/1210 = 0.008

Ora calcoliamo con la Legge di ware generalizzata lo speed-up: 1/(0.008+(0.016/2)+(0.033/4)+(0.94/8)) = 7.05. L'efficienza E: S(p)/p = 7.05/8 = 0.88

Il tempo di esecuzione dell'algoritmo: T(p) = (Ts + ($\frac{T(p)}{p}$)t_calc) + (Tc)t_comm, dove Ts è il tempo impiegato per la parte sequenziale dell'algoritmo e T(p)/p è il tempo impiegato per eseguire la parte parallelizabile.

Ts = 1*10 = 10 e T(8) = t_comm + t_calc.

Al passo 0 si effetuano 15*10 t_calc, cioè il tempo massimo di esecuzione svolto in parallelo da p processori. Al passo 1 si effetuano 10 t_calc + 10t_comm operazioni su 4 processori. Al passo 2 si effetuano 10t_calc 10t_comm operazioni su 2 processori e al passo 3 si effettuano 10t_calc 10t_comm operazioni su 1 processore. Consideriamo t_comm = 2t_calc

Quindi: T(8) = 150 + (30*3)t_calc = 240t_calc.

T(p) = (10+ ($\frac{240}{8}$)) t_calc + 30t_comm = 100t_calc.

2- Si consideri il prodotto matrice per vettore A·b=c, con A∈ R^{38x9} e b ∈ R^9, su un'architettura MIMD-DM, costituita da 4 processori, disposti secondo una griglia bidimensionale 2×2. La matrice A è distribuita per blocchi di righe e di colonne.

a) Che dimensione hanno i blocchi Aloc e come deve essere distribuito il vettore b?

b) Si descriva l'algoritmo relativo alla risoluzione del problema.

a: I sottoblocchi di matrice locali a ciascun processo avranno dimensione: 19x4 e 19x5 (9/2, P0 e P1). Il vettore b viene suddiviso in due parti, e per effetuare il prodotto locale la prima parte sarà assegnata alla prima colonna dei processori (P0 e P2, elementi diversi), la seconda parte alla seconda colonna di processori (P1 e P3). Sia i l'indice della colonna => avremo che la prima colonna riceve gli elementi b[4] * i, b[4] * i +1, b[4] * i +2 con i= indice colonna.

b: Ciascun processore effettua un prodotto matrice-vettore tra la sua porzione di A e la sua porzione di b, in questo modo ogni processore avrà calcolato un contributo per il risultato finale sottoforma di vettore b di dimensione 38. Per ottenere il risulato finale i processori sulla stessa riga sommeranno elemento per elemento. Fatto ciò i risultati ottenuti saranno concatenati per colonna ottenendo così il risultato finale.

Calcolo Parallelo e Distribuito Modulo A

Esame

Prof. Giuliano Laccetti (A.A. 2017-2018 – 09/01/2018)

1. Si consideri l'algoritmo della somma di $N = 62$ vettori, definiti in $\Re^4$, con $P = 8$ processori. Si supponga che lo schema di comunicazione sia ad albero.

- **a.** Si calcolino speedup ed efficienza secondo la legge generalizzata di Ware Amdahl.
- **b.** Opzionalmente, calcolare il tempo di esecuzione secondo la definizione classica.

2. Si consideri l'algoritmo della somma di $N = 193$ numeri in parallelo, con $P = 8$ processori.

- **a.** Si calcoli il valore dello speedup ed efficienza utilizzando tutte e tre le strategie di comunicazioni studiate.
- **b.** Si confrontino i risultati e si facciano delle considerazioni.

3. Si consideri la seguente routine della libreria **MPI:**

int **MPI_Gather**(const void *sendbuf, int sendcount, MPI_Datatype
sendtype, void *recvbuf, int recvcount,
MPI_Datatype recvtype, int root, MPI_Comm comm)

- **a.** Cosa è possibile effettuare utilizzando tale routine?
- **b.** Descrivere, in dettaglio, i parametri.

4. Mostrare come si apre una regione parallela in **OPEN-MP** e illustrare lo schema di **fork join**.

5. Si consideri la classificazione di Flynn relativamente alle architetture parallele.

- **a.** Illustrare la differenza fra le sottoclassi MIMD-DM e MIMD-SM.

Calcolo Parallelo e Distribuito Modulo A

Esame

Prof. Giuliano Laccetti (A.A. 2018-2019 – 04/02/2019)

1. Si consideri l'algoritmo della somma di $\mathrm{N} = 122$ vettori definiti in $\mathbb{R}^{10}$, con $\mathrm{P} = 8$ processori. Si supponga che lo schema di comunicazione sia ad albero.

   a. Si descriva l'algoritmo relativo alla risoluzione del problema.

   b. Calcolare il tempo di esecuzione dell'algoritmo, lo speed-up secondo la legge di Ware Generalizzata e l'efficienza.

2. Si consideri il prodotto matrice per vettore $\mathrm{A} \cdot \mathrm{b} = \mathrm{c}$, con $\mathrm{A} \in \mathbb{R}^{38\times 9}$ e $\mathrm{b} \in \mathbb{R}^{9}$, su un'architettura MIMD-DM, costituita da 4 processori, disposti secondo una griglia bidimensionale 2×2. La matrice A è distribuita per blocchi di righe e di colonne.

   a. Che dimensione hanno i blocchi A_{loc} e come deve essere distribuito il vettore b?

   b. Si descriva l'algoritmo relativo alla risoluzione del problema.

3. Si consideri la seguente routine della libreria MPI:

   MPI_Cart_create(MPI_Comm comm_old, int dim, int *ndim, int *period, int *reorder, MPI_Comm *new_comm).

   a. Cosa è possibile effettuare utilizzando tale routine?

   b. Descrivere, in dettaglio, i parametri.

4. Illustrare lo schema di **fork join** in **OPEN-MP** e mostrare come si apre una regione parallela.

5. Si consideri la classificazione di Flynn relativamente alle architetture parallele. Descrivere l'architettura SIMD e citare almeno un tipo di parallelismo che questa realizza.