Curs 9

Programare Paralela si Distribuita

Metode de evaluare a performatei programelor paralele

Granularitate

Scalabilitate

Complexitate – consideratii generale

- Daca in cazul algoritmilor secventiali performanta este masurata in termenii complexitatiilor timp si spatiu, in cazul algoritmilor paraleli se folosesc si alte masuri ale performantei, care au in vedere toate resursele folosite.
  - ***Numarul de procesoare*** in cazul programarii paralele =>

    o resursa importanta

- Pentru compararea corecta a variantei paralele cu cea seriala, trebuie
  - sa se precizeze arhitectura sistemului de calcul paralel (***sistem paralel = program + arhitectura pe care se executa***)
  - sa se aleaga algoritmul serial cel mai bun si
  - sa se indice daca exista conditionari ale performantei algoritmului datorate volumului de date.

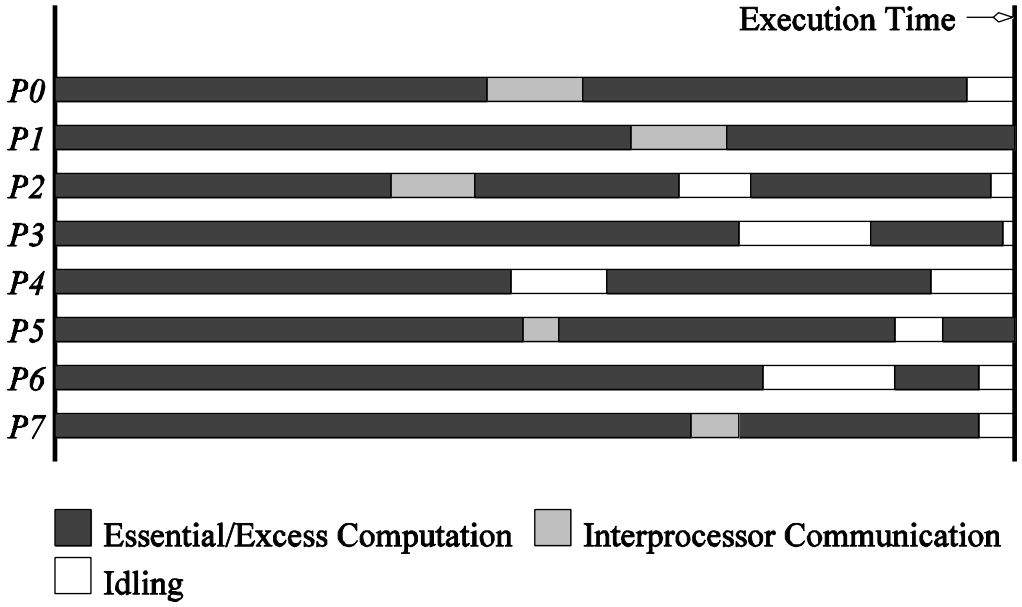
Observatii

- In calculul paralel, obtinerea unui timp de executie mai bun nu inseamna neaparat utilizarea unui numar minim de operatii, asa cum este in calculul serial.

- Factorul memorie nu are o importanta atat de mare ın calculul paralel (relativ).
- In schimb, o resursa majora ın obtinerea unei performante bune a algoritmului paralel o reprezinta numarul de procesoare folosite.

- Daca timpul de executie a unei operatii aritmetice este mult mai mare decat timpul de transfer al datelor intre doua elemente de procesare, atunci intarzierea datorata retelei este nesemnificativa, dar, in caz contrar, timpul de transfer joaca un rol important ın determinarea performantei programului.

Timp de executie



Timpul de executie al unui program paralel masoara perioada care s-a scurs ıntre momentul initierii primului proces si momentul cand toate procesele au fost terminate.

Timp de executie vs Complexitate timp

- In timpul executiei fiecare procesor executa
  - operatii de calcul,
  - de comunicatie, sau
  - este in asteptare.
- Timpul total de executie se poate obtine din formula:

$$t_p = (max\ i : i \in \overline{0, p-1} : T^i_{calcul} + T^i_{comunicatie} + T^i_{asteptare}))$$

- sau in cazul echilibrari perfecte ale incarcarii de calcul pe fiecare procesor din formula:

$$t_p = \frac{1}{p} \sum_{0}^{p-1} (T^i_{calcul} + T^i_{comunicatie} + T^i_{asteptare})$$

Evaluarea teoretica a complexitatii-timp

- Ca si in cazul programarii secventiale, pentru a dezvolta algoritmi paraleli eficienti trebuie sa putem face o evaluare a performantei inca din faza de proiectare a algoritmilor.

- Complexitatea timp pentru un algoritm paralel care rezolva o problema **P(n)** cu dimensiunea **n** a datelor de intrare este o functie **T** care depinde de **n**, dar si de numarul de procesoare **p** folosite.

- Pentru un algoritm paralel, un pas elementar de calcul se considera a fi o multime de operatii elementare care pot fi executate in paralel de catre o multime de procesoare.

- Complexitatea timp a unui pas elementar se poate considera a fi O(1).

- Complexitatea timp a unui algoritm paralel este data de numararea atat a pasilor de calcul necesari dar si a pasilor de comunicatie a datelor/acces la memorie.

Overhead

- T_{all} = timpul total (insumarea timpului pentru toate elementele de procesare).
- T_S = timp serial
- $T_{all} - T_S$ = timp total in care toate procesoarele sunt implicate in operatii care nu sunt strict legate de scopul problemei
  non-goal computation work
  -> *total overhead*.
- $T_{all} = p\ T_P$ (p = nr. procesoare).
- $T_o = p\ T_P - T_S$

Accelerarea(“speed-up”),

- Accelerarea notata cu Sp, este definita ca raportul dintre timpul de executie al celui mai bun algoritm serial cunoscut, executat pe un calculator monoprocesor si timpul de executie al programului paralel echivalent, executat pe un sistem de calcul paralel.
- Daca se noteaza cu t_1 timpul de executie al programului serial, iar tp timpul de executie corespunzator programului paralel, atunci:

$$S_p(n) = \frac{t_1(n)}{t_p(n)}.$$

  - n reprezinta dimensiunea datelor de intrare,
  - p numarul de procesoare folosite.

Variante

- ***relativa***, cand ts este timpul de executie al variantei paralele pe un singur procesor al sistemului paralel;

- ***reala***, cand se compara timpul executiei paralele cu timpul de executie pentru varianta seriala cea mai rapida, pe un procesor al sistemului paralel;

- ***absoluta,*** cand se compara timpul de executie al algoritmului paralel cu timpul de executie al celui mai rapid algoritm serial, executat de procesorul serial cel mai rapid;

- ***asimptotica,*** cand se compara timpul de executie al celui mai bun algoritm serial cu functia de complexitate asimptotica a algoritmului paralel, ın ipoteza existentei numarului necesar de procesoare;

- ***relativ asimptotica,*** cand se foloseste complexitatea asimptotica a algoritmului paralel executat pe un procesor.

Analiza asimptotica (considera dimensiunea datelor n si numarul de procesoare p foarte mari) ignora termenii de ordin mic, si este folositoare ın procesul de constructie al programelor performante.

Eficienta

- Eficienta este un parametru care masoara gradul de folosire a procesoarelor.
- Eficienta este definita ca fiind:

  $E = Sp/p$

- Daca accelerarea este cel mult egala cu p se deduce ca valoarea eficientei este subunitara.

Legea lui Amdahl

- Afirma ca accelerarea procesarii depinde de raportul partii secventiale fata de cea paraleizabila:

seq = fractia calculului secvential; (e.g 20%=> seq=20/100)
par = fractia calculului paralelizabil; (e.g 80%=> par=80/100)

Se considera calculul serial T_s= seq+par = 1 unitate

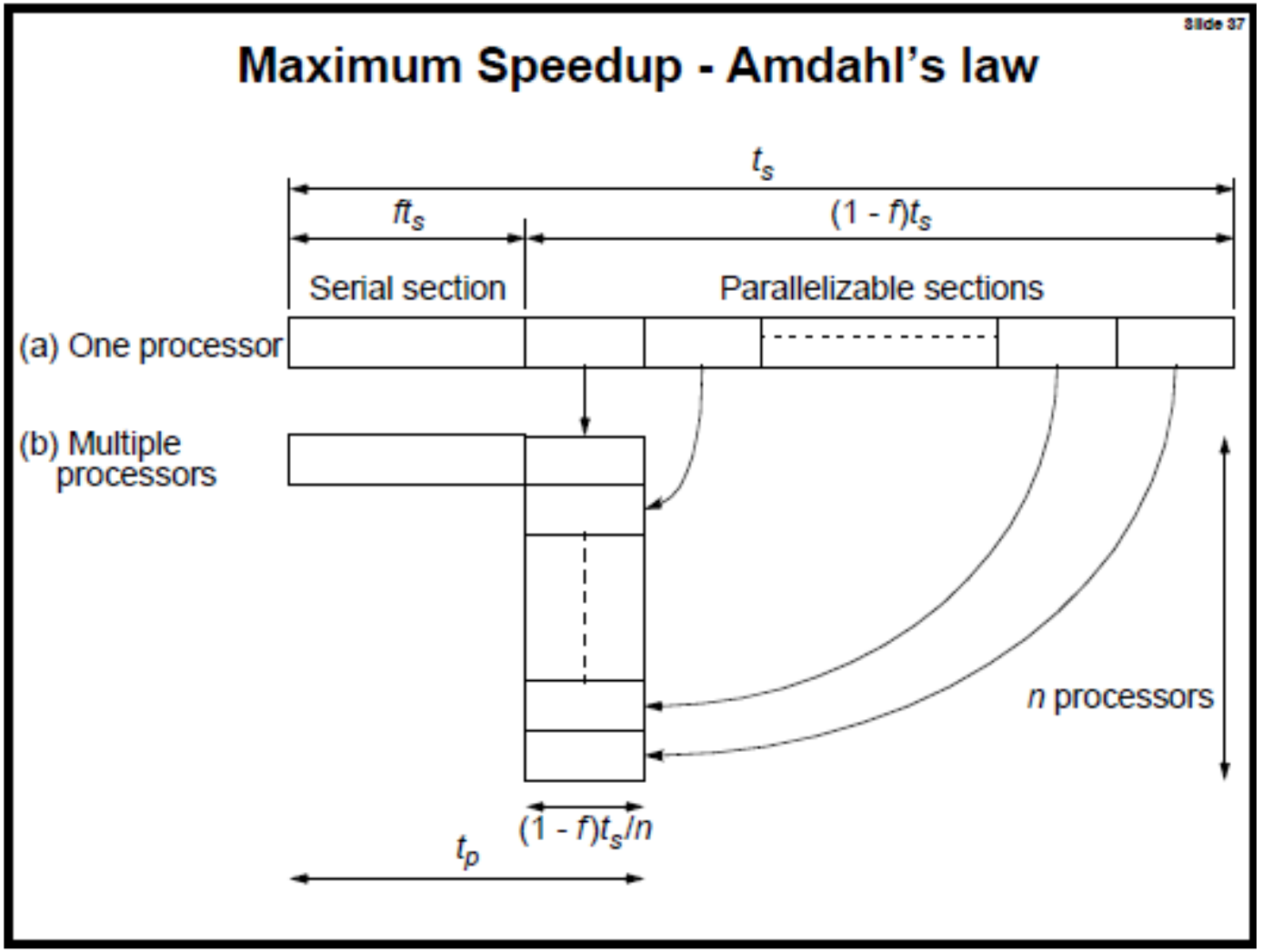
Speedup = 1/(seq + par/p)

par = (1 – seq), p= # procesoare

- *p -> infinit* => S ~ 1/seq (e.g. S~ 100/20=5)
- Limita superioara a accelerarii este data de fractia partii secventiale.
- *Nu se face analiza in functie de dimensiunea problemei!*

Prezentare alternativa

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1+(n-1)f}$$



Legea lui Gustafson

- Considera ca atunci cand dimensiunea problemei creste partea seriala se micsoreaza in procent :
- m = dimensiunea problemei, p = # procesoare,
  seq(m) = fractia calculului secvential;
  par(m) = fractia calculului paralel;

Considerand ca programul paralel se executa intr-o unitate de timp :

$T_p = \underline{seq(m) + par(m) = 1}$

$T_s = seq(m)+p*par(m)$

Atunci

- $speedup = T_s / T_p = seq(m)+p*par(m)$

$speedup = seq(m) + p(1 – seq(m))$

Daca $seq(m) \to 0$ atunci cand $m \to \infty$ => se obtine ~ accelerare liniara.

Legea lui Gustafson – optimista
Legea lui Amdahl - pesimista

- Legea lui Gustafson -> presupune ca partea seriala (costul ei) ramane constanta – nu creste odata cu cresterea problemei.

- Legea lui Amdahl -> presupune ca **procentul** partii secventiale este constant - nu depinde de dimensiunea problemei

Evaluare timp de Comunicatie

- Pentru a estima timpul de comunicatie trebuie sa se tina seama de urmatorii factori (timpi):
  - *Startup time* (t_s): Timpul necesar pentru trimitere si receptionare la nivelul fiecarui nod Ti (executing the routing algorithm, programming routers, etc.).
  - *Per-hop time* (t_h): acest timp este evaluat printr-o functie care are ca si variabila numarul de ‘hops’ (noduri) si include factori precum *switch latencies, network delays, e*tc.
  - *Per-word transfer time* (t_w): Acest timp include toate ‘overheadurile’ determinate de lungimea mesajului. Acesta depinde de latimea de: *bandwidth of links, error checking and correction,* etc.

Simplified Cost Model for Communicating Messages

- Daca se tine cont de numarul de hops

$$t_{comm} = t_s + lt_h + t_w m.$$

- Varianta simplificata (se poate accepta deoarece intr-o varianta de transmitere bazata pe ‘cut-through routing’ (t_h) este foarte mic comparativ cu (t_s) si (t_w) si se poate ignora

$$t_{comm} = t_s + t_w m.$$

- Se poate accepta doar daca se considera ‘uncongested networks!

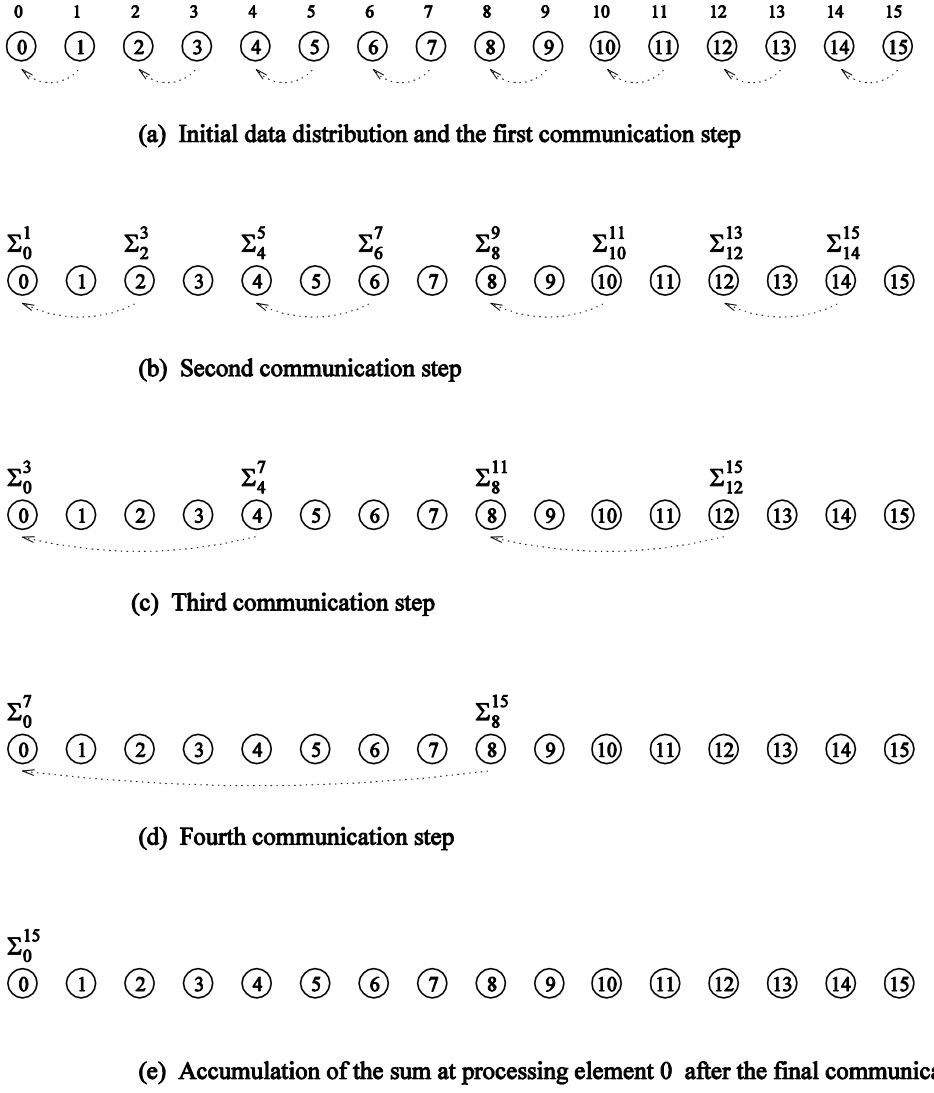
Evaluare overhead in cazul ‘multithreading’

- Timp creare threaduri
- Timp gestiune threaduri
- Timp de sincronizare
  (asteptare -> excludere mutuala; asteptare conditionala, etc…)
- Timp oprire threaduri

Exemple

- Adunarea a n numere
- Daca ***n*** = putere a lui 2 => T_p = **log *n***

Adunare – log n pasi



n=16; p=16s . .

Exemplu (continuare)

=>

- $t_{C =}$ ***timp calcul pt o operatie de adunare***
- $T_{com} = t_s + t_w$ **pt o operatie de comunicatie (per word)**
  - sunt O(n) operatii de comunicatie dar si operatiile de comunicatie se pot executa simultan $T_{\_com} = \Theta$ (**log** n)

  $T_P = \Theta$ (**log** n)

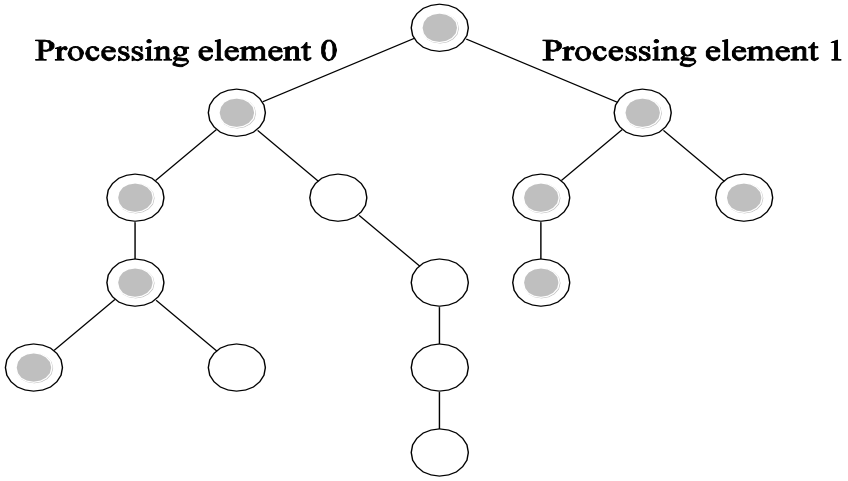
- Stim ca $T_S = \Theta$ (n)

- Speedup $S = \Theta$ (n / **log** n) (p=n => E = Θ (1 / **log** n)

Accelerare – superliniara?

- S = 0 (the parallel program never terminates).
- S < ***p*** (teoretic)
  - In caz contrar un procesor ar fi implicat in calcule pentru rezolvarea problemei un timp T < $\boldsymbol{T_S / p}$ *in conditiile in care* T_S *depinde de numarul de operatii care se efectueaza => se efectueaza mai putine operatii in total*

Superlinear Speedups



Cautare intr-un arbore nestructurat.

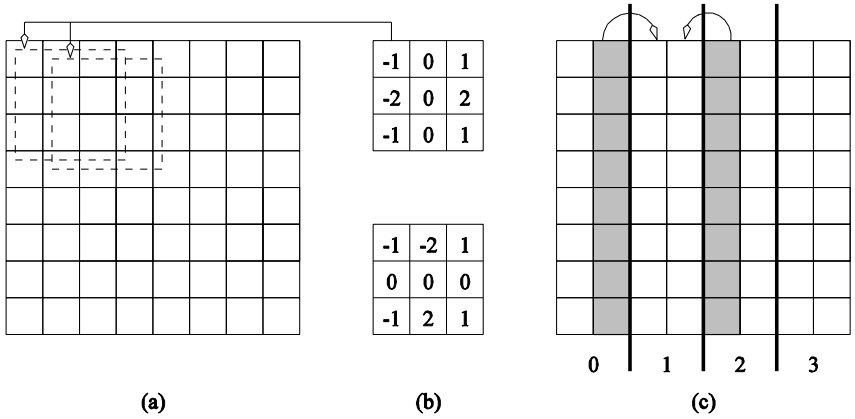
Exemplu

Problema : *filtrare imagini = model distribuit*

- modificari pe celule de ***3*** x ***3*** pixeli.

Daca o operatie aritmetica necesita pentru calcul t_c,

timpul serial pt o imagine de n x n este $T_S = 9t_c\, n^2$.



- Partitionare veriticala => $\boldsymbol{n^2}$ / $\boldsymbol{p}$ pixels.

- Marginea fiecarui segment => $\boldsymbol{2n}$ pixels.
- Nr de valori care trebuie comunicate = 2n=> $\boldsymbol{2}(\boldsymbol{t_s} + \boldsymbol{t_w n})$.

Timp de comunicatie: $t_{com} = t_s + t_w * mes\_size$

(t_s – timp de start al unei comunicatii)

calculul executat de fiecare process:

$\boldsymbol{T^S_p = 9\, t_c\, n^2 / p}$

Evaluare metrici

- Timpul paralel:

$$T_P = 9t_c\frac{n^2}{p} + 2(t_s + t_w n)$$

- Accelerarea si eficienta:

$$S = \frac{9t_c n^2}{9t_c\frac{n^2}{p} + 2(t_s + t_w n)}$$

$$E = \frac{1}{1 + \frac{2p(t_s + t_w n)}{9t_c n^2}}.$$

Costul

- Costul se defineste ca fiind produsul dintre timpul de executie si numarul maxim de procesoare care se folosesc.

  $C_p(n) = t_p(n) \cdot p$

- Aceasta definitie este justificata de faptul ca orice aplicatie paralela poate fi simulata pe un sistem secvential, situatie in care unicul procesor va executa programul intr-un timp egal cu $O(C_p(n))$.

- O aplicatie paralela este **optima** din punct de vedere al costului, daca valoarea acestuia este egala, sau este de acelasi ordin de marime cu timpul celei mai bune variante secventiale --- $C_p = O(t_s)$.

- aplicatia este **eficienta** din punct de vedere al costului daca $C_p = O(t_s \log p)$.

Costul unui sistem paralel (algoritm +sistem)

- Cost = $\boldsymbol{p}$ x $\boldsymbol{T_P}$
- Costul reflecta suma timpului pe care fiecare procesor il petrece in rezolvarea problemei.
- Un sistem paralel se numeste optimal daca costul rezolvarii unei probleme pe un calculator paralel este asimptoptic egal cu costul serial.
- $\boldsymbol{E} = \boldsymbol{T_S} / \boldsymbol{p}\ \boldsymbol{T_P}$=> pentru sisteme cost optimal => $\boldsymbol{E} = O(1)$.
- Cost ~ *work* ~ *processor-time product*.

Exemplu: Adunare n numere pe un model distribuit

- $\boldsymbol{T_P} = (t_c + t_{com})\ \mathbf{log}\ \boldsymbol{n}$ (pt $\boldsymbol{p} = \boldsymbol{n}$).

t_c timpul necesar unei operatii de adunare;
t_{com-} timpul necesar unei operatii de comunicatie;

- $C = \boldsymbol{p}\ \boldsymbol{T_P} = O(\boldsymbol{n}\ \mathbf{log}\ \boldsymbol{n})$
- $T_s = \Theta(\boldsymbol{n})$ => nu este cost optimal
- Cum se poate optimiza?
  - se micsoreaza p; **p=n/k**
  - se considera **p** segmente (**dim = n/p**) pentru care se calculeaza suma secvential
  - se foloseste calculul de tip arbore pentru insumarea celor p sume locale
  - $\mathbf{T_p = t_c\ n/p + (t_c + t_{com})\ log\ p}$
  - $C = t_c\ n + (t_c + t_{com}) p * \log p$ => daca $p * \log p = O(n)$ atunci cost optimal

Scalabilitate

- ***Scalabilitatea este un parametru calitativ care caracterizeaza atat***
  - ***sistemele paralele (numar de procesoare, unitati de memorie) – vezi curs 2***

  cat si
  - ***aplicatiile paralele!***

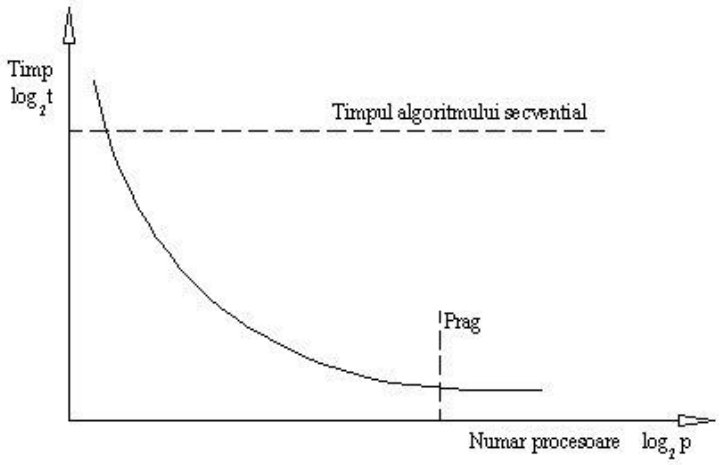
Scalabilitatea aplicatiilor paralele

- *Un program se poate scala a.i. sa foloseasca mai multe procesoare?*
  - Adica???
  - Cum se evalueaza scalabilitatea?
  - Cum se evalueaza benefiile aduse de scalabiltate?
- Evaluare performantei:
  - Daca se dubleaza nr de procesoare la ce ar trebui sa ne asteptam?
  - Cu cat trebuie sa creasca dimensiunea problemei daca dublam numarul de procesoare astfel incat sa obtinem aceeasi eficienta?
  - Este scalabilitatea liniara? (raport p1/p2= n1/n2)
- Evaluarea eficientei corespunzatoare
  - Se pastreaza eficienta pe masura ce creste dimensiunea problemei?
    - cat de mult trebuie sa creasca p?

Scalabilitate aplicatiei: abilitatea unui program paralel sa obtina o crestere de performanta proportionala cu numarul de procesoare

Scalabilitate

- **Scalabilitatea masoara modul in care se schimba(creste) performanta unui anumit algoritm in cazul in care sunt folosite mai multe elemente de procesare.**



- Un indicator important pentru aceasta este ***numarul maxim de procesoare*** care pot fi folosite pentru rezolvarea unei probleme.
- In cazul folosirii unui numar mic de procesoare, un program paralel se poate executa mult mai incet decat un program secvential.
  - Diferenta poate fi atribuita comunicatiilor si sincronizariilor care nu apar ın cazul unui program secvential. (Overhead)
- Numarul minim de componente(taskuri) pentru o anumita partitionare, poate fi de asemenea un indicator important.

Analiza scalabilitatii se face pentru un **sistem (software +arhitectura)**

Scalabilitate

Definitie

•*Scalabilitatea unui sistem parallel (aplicatie software + tip arhitectura) este o masura a capacitatii de a livra o accelerare ca si o functie crescatoare cu parametru numarul de procesoare folosite.*

- *p creste (pana la ce limita?)=> S creste (linear, logaritmic…)*

- *Evidentiaza cum se extrapoleaza performanta de la probleme si sisteme mici*
  =>
  la probleme si sisteme mai mari.

•Cu cat trebuie sa creasca dimensiunea problemei daca marim numarul de procesoare astfel incat sa obtinem aceeasi eficienta?

- **p2= k*p1 => n2= X*n1 X=? astfel incat E1=E2**

S1/p1=S2/p2 => k=p2/p1=S2/S1=T_p(n1)/ T_p(n2)

•Metrici pentru scalabilitate –
- – functia de isoeficienta (isoefficiency)
- – eficienta Isospeed –efficiency
- – Fractia seriala/Serial Fraction *f*

weak vs strong scaling analysis

- In weak scaling analysis, we evaluate the speedup, efficiency or the running time of a parallel algorithm in points (n, p) where we ensure that the problem size per processor remains constant.
  - A common practice in weak scaling analysis consists in doubling both the size of the problem and the number of processors.
  - If the running time or the efficiency remains constant, then the algorithm is scalable.
- In strong scaling analysis, we are interested in determining how far we can remain efficient given a fixed problem size.
  - Therefore, for a fixed problem size, we increase the number or processors until we observe a change in the efficiency.

Exemple

- Suma de 2 vectori (memorie partajata) = scalabilitate foarte buna

n operatii independente => $p_{maxim} = n$

daca n creste si p poate creste => eficienta ramane la fel

Ideal

$S = n/(n/p) = p$; $E = 1$

(!!!daca se ignora timpul de creare threaduri/procese; distributia datelor!!!)

- Suma a n numere folosind p procesoare (model distribuit)

$S = n\,t_c / (t_c\,n/p + (t_c+t_{com}) \log p) =$

$np\,t_c / (t_c\,n + (t_c+t_{com})p \log p) = p\,t_c / [t_c + (t_c+t_{com})p \log p/n)]$

Acceleratia creste daca p creste pana la limita la care plog p ~n

$E = n\,t_c /(t_o\,n + (t_c+t_{com})\,p \log p)$

$p_{maxim}=n \Rightarrow E = n\,t_c / (t_c\,n + (t_c+t_{com})\,n \log n)$

Filtru pe imagine

- Timpul paralel:

$$T_P = 9t_c\frac{n^2}{p} + 2(t_s + t_w n)$$

- Accelerarea si eficienta:

$$S = \frac{9t_c n^2}{9t_c\frac{n^2}{p} + 2(t_s + t_w n)}$$

$$E = \frac{1}{1 + \frac{2p(t_s + t_w n)}{9t_c n^2}}.$$

- Daca p creste atunci eficienta scade – cat de mult?

  $(2t_w pn/9t_c n^2) = (2t_w p/9t_c n) < 1$

  - $p_{maxim} = n \Rightarrow (2t_w/9t_c) \sim 1$

Granularitate

- ***Granularitatea("grain size") este un parametru calitativ care caracterizeaza atat***
  - ***sistemele paralele (numar de procesoare, unitati de memorie)***

  cat si
  - ***aplicatiile paralele.***

- ***Granularitatea aplicatiei*** *se defineste ca dimensiunea* ***minima*** *a unei unitati secventiale dintr-un program, exprimata in numar de instructiuni.*
  - *Prin unitate secventiala se intelege o parte din program in care nu au loc operatii de sincronizare sau comunicare cu alte procese.*
- ***Granularitatea medie*** *= media tuturor grain sizes*
  - *Si aceasta reprezinta o caracteristica importanta a eficientiei aplicatiei*

- Fiecare flux de instructiuni are o anumita granularitate.

- Granularitatea medie unui algoritm poate fi aproximata ca fiind raportul dintre **timpul total calcul si timpul total de comunicare.**

Granularitatea sistemului

- *Pentru un **sistem paralel** dat, exista o valoare minima a granularitatii aplicatiei, sub care performanta scade semnificativ. Aceasta valoare de prag este cunoscuta ca si **granularitatea sistemului** respectiv.*

  – Justificarea consta in faptul ca timpul de overhead (comunicatii, sincronizari, etc.) devine comparabil cu timpul de calcul paralel.
- De dorit

  => un calculator paralel sa aiba o granularitate mica, astfel ıncat sa poata executa eficient o gama larga de programe.

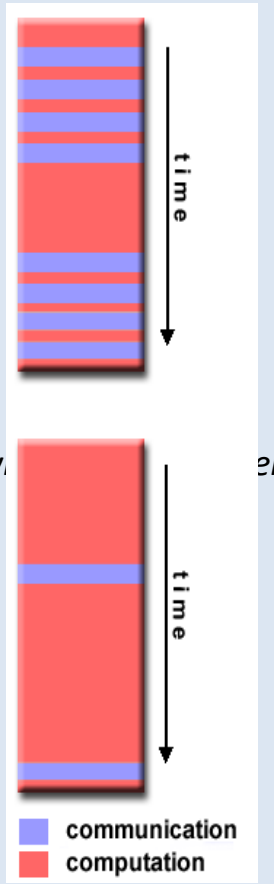
  ⇒ programele paralele sa fie caracterizate de o granularitate mare, astfel ıncat sa poata fi executate eficient de o diversitate de sisteme.

- Exceptii: clase de aplicatii cu o valoare a granularitatii foarte mica, dar care se executa cu succes pe arhitecturi specifice.
  – aplicatiile sistolice, in care in general o operatie este urmata de o comunicatie.
    - aceste aplicatii impun insa o structura a comunicatiilor foarte regulata si comunicatii doar intre noduri vecine

Granularitate…cont.

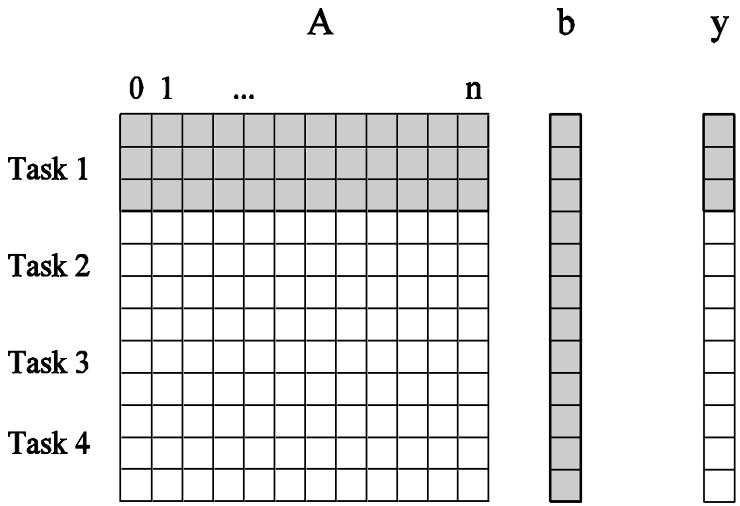
- ***Fine-grain Parallelism:***
  - *Relatively small amounts of computational work are done between communication events*
  - *Low computation to communication ratio*
  - *Facilitates load balancing*
  - *Implies high communication overhead and less opportunity for performance enhancement*
  - *If granularity is too fine it is possible that the overhead required for communications and synchronization between tasks takes longer than the computation.*

- ***Coarse-grain Parallelism:***
  - *Relatively large amounts of computational work are done between communication/sy ents*
  - *High computation to communication ratio*
  - *Implies more opportunity for performance increase*
  - *Harder to load balance efficiently*



Granularitate <= descompunere in taskuri

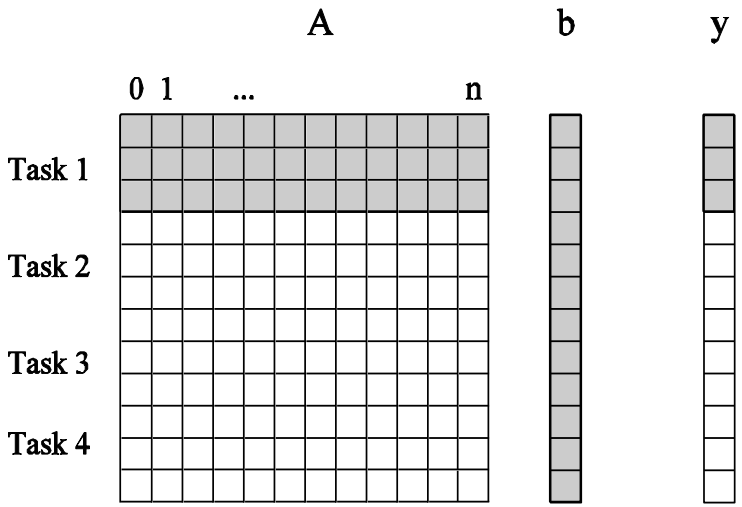
- Numarul de task-uri in care o problema se descompune determina granularitatea.
- numar mare => fine-grained decomposition
- numar mic => coarse grained decomposition



Exemplu: inmultire matrice

Granularitea decompunerii taskurilor

- Granularitatea este ***determinata*** de numarul de taskuri care se creeaza pt o problema.
- Mai multe taskuri => granularitate mai mica



Efectul granularitatii asupra performantei

- De multe ori, folosirea a mai putine procesoare imbunatateste performata sistemului parallel (system paralele= ansamblu aplicatie+sistem).
- Folosind mai putine procesoare decat numarul maxim posibil se numeste ***scaling down*** (for a parallel system).
- Modalitatea naiva de scalare este de a considera fiecare element de procesare initiala fi unul virtual si sa se atribuie fiecare procesor virtual unuia real (fizic).
- Daca numarul de procesoare scade cu un factor ***n*** / ***p***, calculul efectuat de catre fiecare procesor va creste cu acelasi factor.
- Costul comunicatiei nu creste pentru ca comunicatia intre unele procesoare virtuale se va face in cadrul aceluiasi procesor (real).

DOP Gradul de paralelism *(Degree Of Parallelization)*

- DOP al unui algoritm este dat de numarul maxim de operatii care pot fi executate simultan.

  - fina – numar mare de operatii executate in paralel
  - medie
  - bruta

DOP(Degree of Parallelism)

- *Gradul de paralelism DOP* („degree of parallelism”) =
  - (al unui program)
    - Numarul de procese care se executa in paralel intr-un anumit interval de timp.
    - Numarul de operatii care se executa in paralel intr-un anumit interval de timp
  - (al unui sistem)
    - numarul de procesoare care pot fi in executie in paralel intr-un anumit interval de timp

- *Profilul paralelismului* = graficul *DOP* in functie de timp (pentru un anumit program).

Depinde de:

-structura algoritmului;

-optimizarea programului;

-utilizarea resurselor;

-conditiile de rulare.

In concluzie:

-accelerarea indica castigul de viteza de calcul intr-un sistem paralel;

-eficienta masoara partea utila a prelucrarii (lucrului) efectuate de *n* procesoare;

-costul masoara efortul necesar in obtinerea unui viteze de calcul mai mare

-scalabilitatea masoara gradul in care o aplicatie poate folosi eficient un numar mai mare de procesoare

-granularitatea depinde de raportul intre operatiile de calcul si cele de comunicatie/sincronizare

Referinte:

Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar
``Introduction to Parallel Computing'',