

Curs 12

Programare Paralela si Distribuita

Strategii de Partitionare/Descompunere

Sabloane de proiectare

Partitionarea

- Problema partitionarii are in vedere impartirea problemei de programare in componente care se pot executa concurrent.
- Aceasta nu implica o divizare directa a programului intr-un numar de componente egal cu numarul de procesoare disponibile.
- Cele mai importante scopuri ale partitionarii sunt legate de:
 - scalabilitate,
 - abilitatea de a ascunde intarzierea(*latency*) datorata retelei sau accesului la memorie si
 - realizarea unei granularitati cat mai mari.
- Sunt de preferat partitionarile care furnizeza mai multe componente decat procesoare, astfel incat sa se permita **ascunderea intarzierii**:
 - Un task poate fi blocat, sau va astepta, pana cand
 - un mesaj care contine informatia dorita va ajunge;
 - o conditie care este necesara ajunge sa fie adevarata;
 - ...
 - Daca exista si alte componente program disponibile, procesorul poate continua calculul
=> multiprogramare => executie concurenta

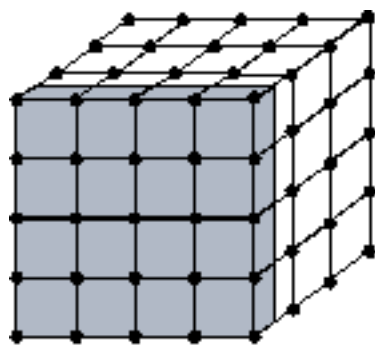
Strategii de partitionare

- Exista doua strategii principale de partitionare:
 - descompunerea domeniului de date si
 - descompunerea functionala.
- In functie de acestea putem considera aplicatii paralele bazate pe:
 - descompunerea domeniului de date – paralelism de date, si
 - aplicatii paralele bazate pe descompunerea functionala.
- Cele doua tehnici pot fi folosite insa si impreuna:
 - de exemplu se incepe cu o descompunere functionala si dupa identificarea principalelor functii se poate folosi descompunerea domeniului de date pentru fiecare in parte.

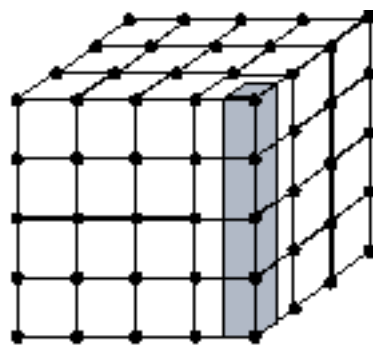
Descompunerea domeniului de date

- Este aplicabila atunci cand domeniul datelor este mare si regulat.
- Ideea centrala este de a divide domeniul de date, reprezentat de principalele structuri de date, in componente care pot fi manipulate independent.
- Apoi se partitioneaza operatiile, de regula prin asocierea calculelor cu datele asupra carora se efectueaza.
- Astfel, se obtine un numar de activitati de calcul, definite de un numar de date si de operatii.
 - Este posibil ca o operatie sa solicite date de la mai multe activitati. In acest caz, sunt necesare comunicatii.

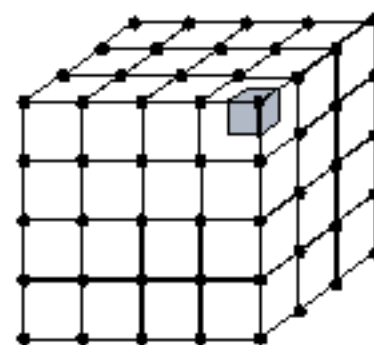
Exemplificare



1-D



2-D



3-D

Descompunerea domeniului de date

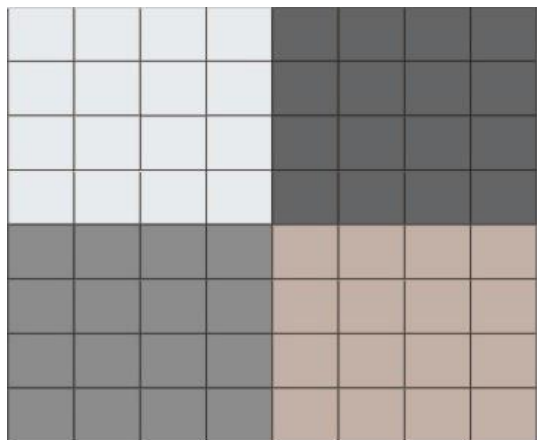
- Sunt posibile diferite variante, in functie de structurile de date avute in vedere.
- Datele care se partitioneaza sunt in general datele de intrare, dar pot fi considerate deasemenea si datele de iesire sau intermediare.
 - Trebuie avute in vedere in special structurile de date de dimensiuni mari sau cele care sunt accesate in mod frecvent.
- Faze diferite ale calculului pot impune partitionari diferite ale aceleiasi structuri de date
 - =>redistribuirii ale datelor
 - in acest caz trebuie avute in vedere de asemenea si costurile necesare redistribuirii datelor.

Distributii de date

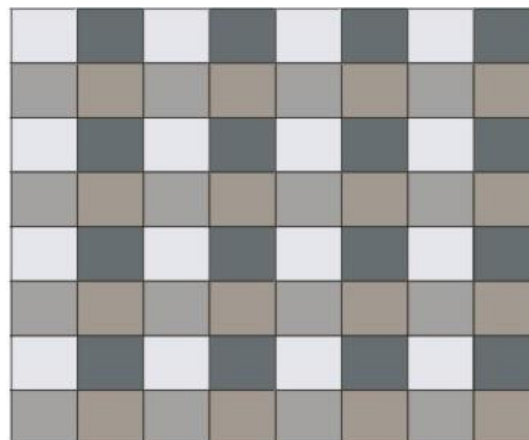
- Partitionarea datelor conduce la anumite distributii ale datelor per procese si de aici implicit si per procesoare.
- Exista mai multe tehnici de partitionare a datelor, care pot fi exprimate si formal prin functii definite pe multimea indicilor datelor de intrare cu valori in multimea indicilor de procese.
- Cele mai folosite tehnici de partitionare sunt prin “taiere” si prin “incretire” care corespund distributiilor liniara si ciclica.

Distributii de date

- De exemplu, pentru o matrice A de dimensiune 8×8 , partitionarea sa in $p = 4$ parti:
 - prin tehnica taierii (distr. liniara) conduce la partitionarea aratata de Figura (a), iar
 - prin tehnica incretirii (distr. ciclica) conduce la partitionarea aratata de Figura (b).



(a) Taiere (distributie bloc)



(b) Incretire (distributie grid)

Variante

1D



BLOCK



CYCLIC

2D



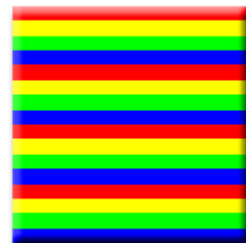
BLOCK, *



*, BLOCK



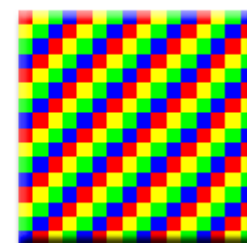
BLOCK, BLOCK



CYCLIC, *



*, CYCLIC

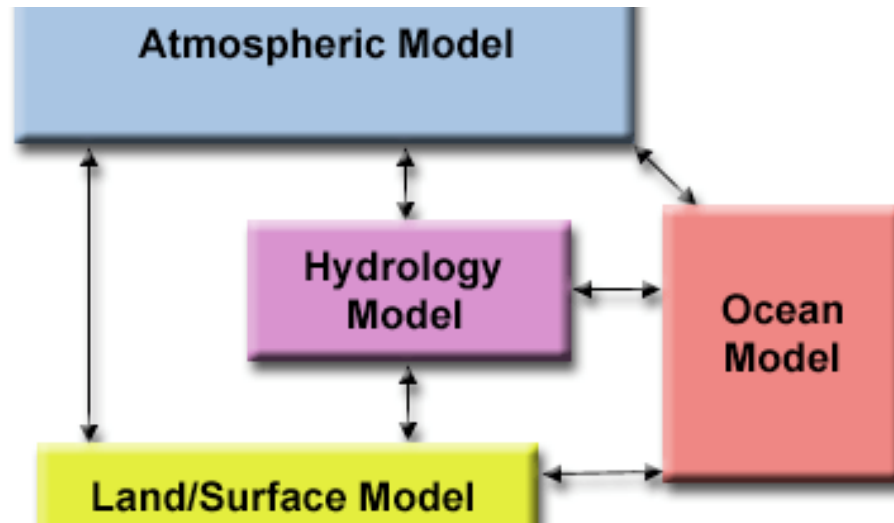


CYCLIC, CYCLIC

Descompunerea functionala

- Descopunerea functionala este o tehnica de partitionare folosita atunci cand aspectul dominant al problemei este functia, sau algoritmul, mai degraba decat operatiile asupra datelor.
- Obiectivul este descompunerea calculelor in activitati de calcul cat mai fine.
- Dupa crearea acestora se examineaza cerintele asupra datelor.
- **Focalizarea asupra calculelor** poate revela uneori o anumita structura a problemei, de unde oportunitati de optimizare, care nu sunt evidente numai din studiul datelor.
- In plus, ea are un rol important ca si tehnica de structurare a programelor.
- Aceasta varianta de descompunere nu conduce in general la o granulatatie fina a sarcinilor de calcul, care se executa in paralel.

Exemplificare



Cerinte generale pentru partitionare

- Task-urile obtinute sunt de dimensiuni comparabile.
- Scalabilitatea poate fi obtinuta.
 - Aceasta inseamna ca numarul de sarcini de calcul sunt definite in functie de dimensiunea problemei;
 - deci cresterea dimensiunii datelor implica cresterea numarului de sarcini de lucru.
- Intazierile pot fi reduse prin multitasking.
- Granularitatea aplicatiei este suficient de mare astfel incat sa poata fi implementata cu succes pe diferite arhitecturi.

SABLOANE DE PROIECTARE

... in general sabloanele de proiectare se bazeaza pe...

- Descompunere functionala
- Descompunere domeniului de date
- Fluxul de date

Observatie:

In literature exista mai multe clasificari de sabloane de proiectare paralela dar sunt cateva sabloane de baza care se regasesc in majoritatea referintelor.

Sabloane de baza

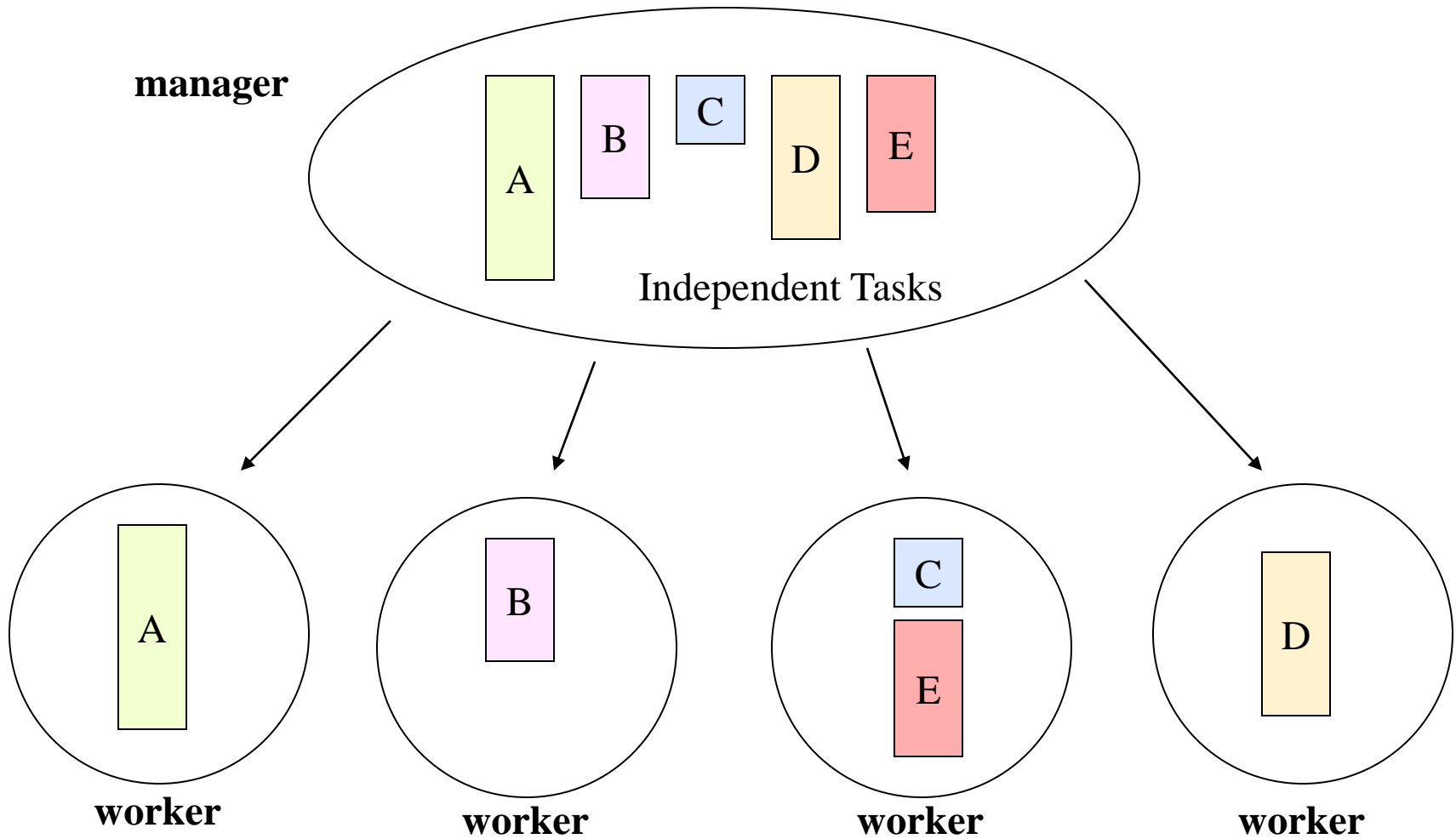
Parallel Programming Patterns
Eun-Gyu Kim 2004

- **Embarassingly Parallel** – e.g. suma de vectori
 - /Master-Slave
- Replicable
- Repository
- **Divide&Conquer**
- **Pipeline**
- Recursive Data
- **Geometric Decomposition**
- IrregularMesh

Variante pentru sabloane bazate pe descompunerea datelor sau a taskurilor

- Descompunere
 - geometrica ... -> caracter static
 - recursiva
 - exploratorie ... -> caracter dinamic
 - speculativa

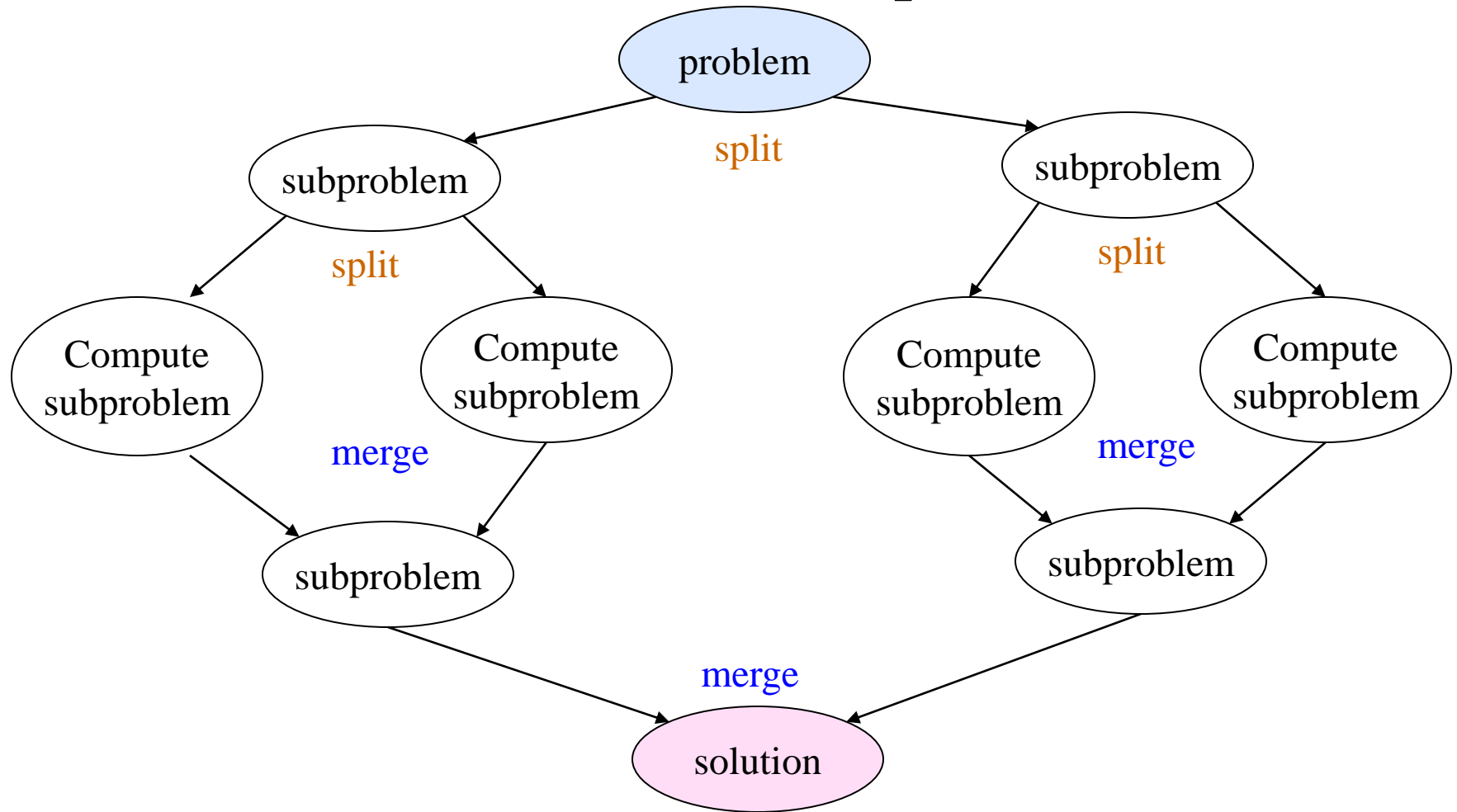
Master-slave (master-worker)



Descompunere Recursiva

- In general pentru probleme care se pot rezolva prin **divide & impera**

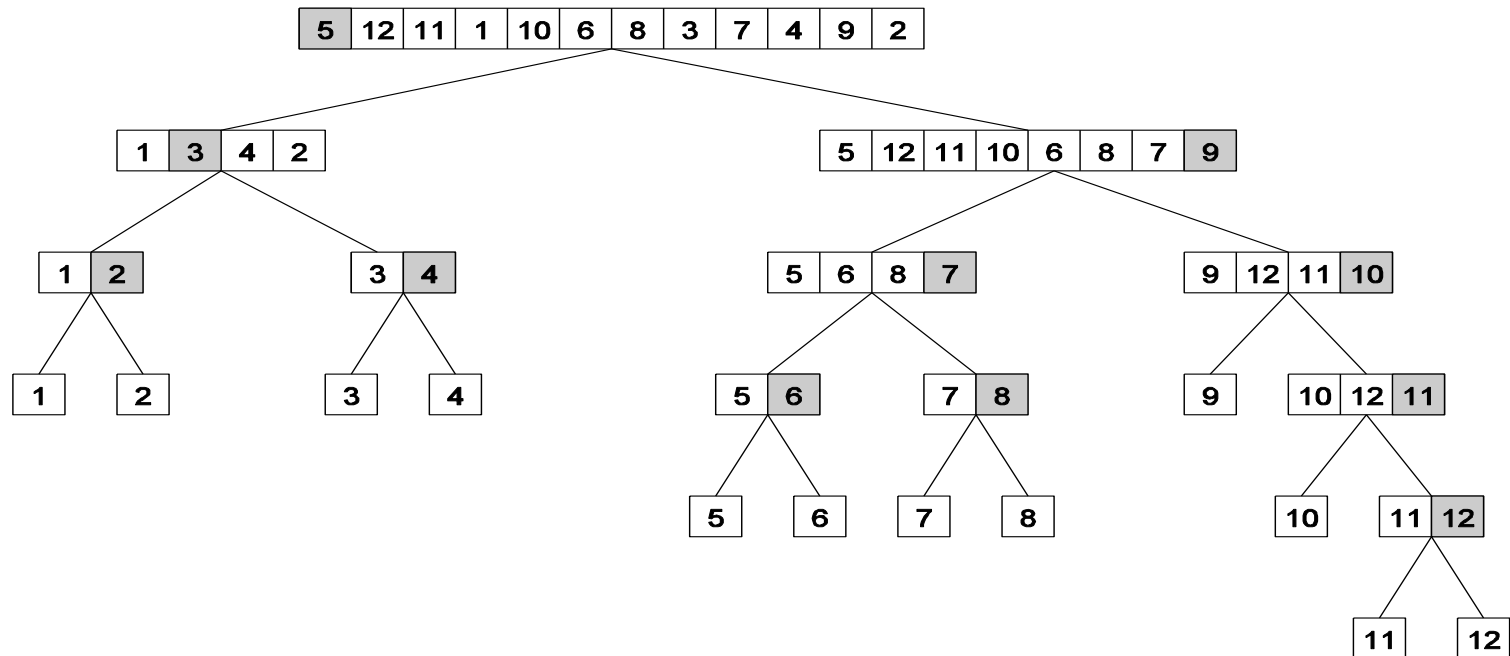
Divide & Conquer



* Nivelurile de descompunere trebuie sa fie ajustate corespunzator.

Recursive Decomposition: Exemplul 1

Quicksort



Fiecare sublista reprezinta un task.

Recursive Decomposition: Exemplul 2

Cautarea minimului:

```
1. procedure SERIAL_MIN (A, n)  
2. begin  
3. min = A[0];  
4. for i := 1 to n - 1 do  
5.           if (A[i] < min) min := A[i];  
6. endfor;  
7. return min;  
8. end SERIAL_MIN
```

Recursive Decomposition

Rescriere

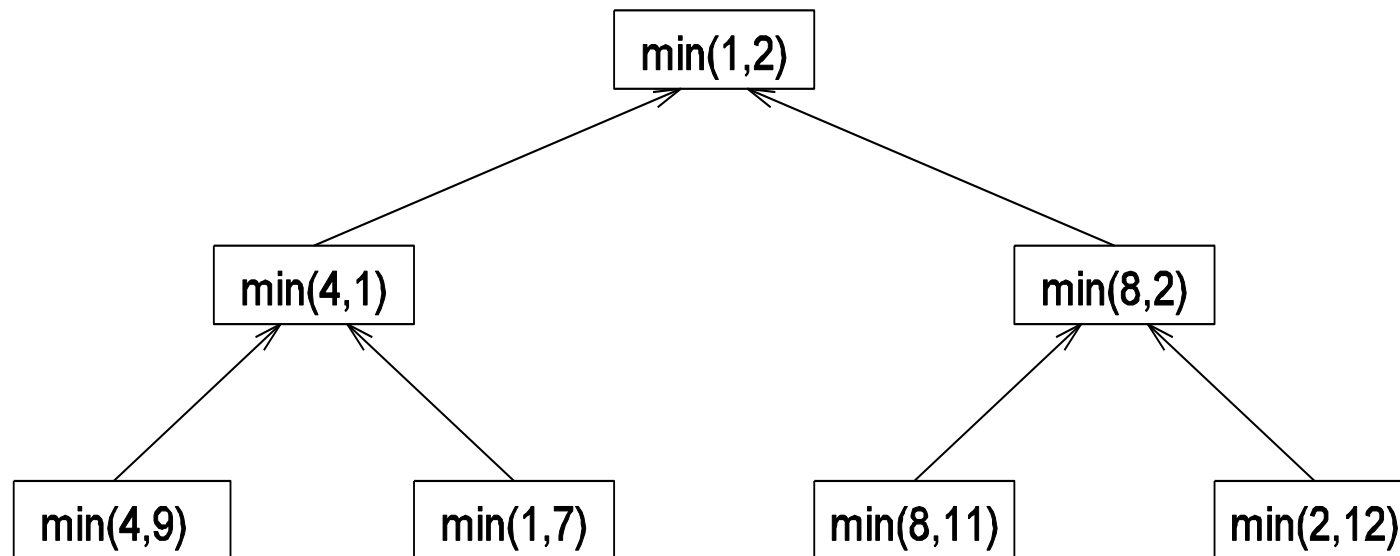
```
1. procedure RECURSIVE_MIN (A, n)
2. begin
3. if ( n = 1 ) then
4.   min := A [0] ;
5. else
6.   lmin := RECURSIVE_MIN ( A, n/2 );
7.   rmin := RECURSIVE_MIN ( &(A[n/2]), n - n/2 );
8.   if (lmin < rmin) then
9.     min := lmin;
10.  else
11.    min := rmin;
12.  endelse;
13. endelse;
14. return min;
15. end RECURSIVE_MIN
```

se pot
executa in
paralel

Recursive Decomposition

{4, 9, 1, 7, 8, 11, 2, 12}.

- task dependency graph

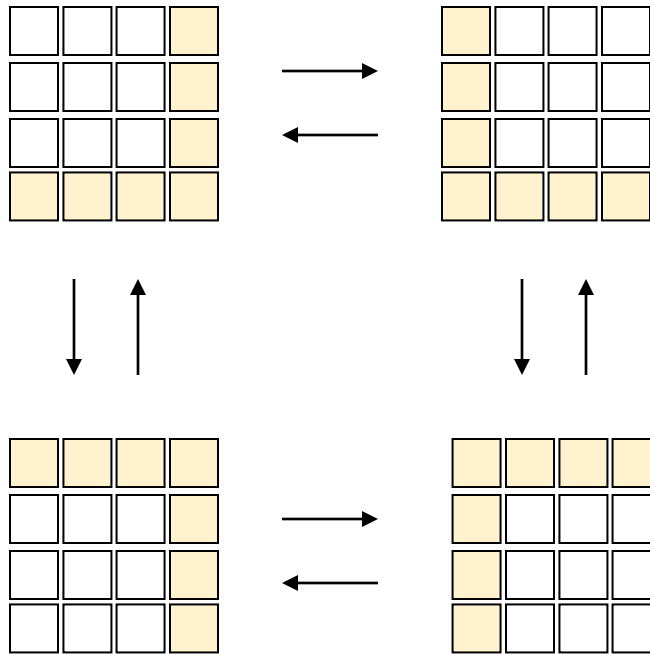


Data Decomposition (geometric)

- Identificarea datelor implicate in calcul.
- Partitionarea datelor pe taskuri.
 - Diferite modalitati care implica impact important pt performanta.

Descompunere Geometrica

Exista dependente dar comunicarea se face intr-un mod predictibil (geometric) -> vecini.



Neighbor-To-Neighbor communication

Output Data Decomposition: Exemplu

inmultire de matrice

$n \times n$ matricele $\mathbf{A} \times \mathbf{B} = \mathbf{C}$.

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Task 1: $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$

Task 2: $C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$

Task 3: $C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$

Task 4: $C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$

Output Data Decomposition: Exemplu

- nu exista doar o descompunere unica – variante!

Decomposition I	Decomposition II
Task 1: $\mathbf{C}_{1,1} = \mathbf{A}_{1,1} \mathbf{B}_{1,1}$	Task 1: $\mathbf{C}_{1,1} = \mathbf{A}_{1,1} \mathbf{B}_{1,1}$
Task 2: $\mathbf{C}_{1,1} = \mathbf{C}_{1,1} + \mathbf{A}_{1,2} \mathbf{B}_{2,1}$	Task 2: $\mathbf{C}_{1,1} = \mathbf{C}_{1,1} + \mathbf{A}_{1,2} \mathbf{B}_{2,1}$
Task 3: $\mathbf{C}_{1,2} = \mathbf{A}_{1,1} \mathbf{B}_{1,2}$	Task 3: $\mathbf{C}_{1,2} = \mathbf{A}_{1,2} \mathbf{B}_{2,2}$
Task 4: $\mathbf{C}_{1,2} = \mathbf{C}_{1,2} + \mathbf{A}_{1,2} \mathbf{B}_{2,2}$	Task 4: $\mathbf{C}_{1,2} = \mathbf{C}_{1,2} + \mathbf{A}_{1,1} \mathbf{B}_{1,2}$
Task 5: $\mathbf{C}_{2,1} = \mathbf{A}_{2,1} \mathbf{B}_{1,1}$	Task 5: $\mathbf{C}_{2,1} = \mathbf{A}_{2,2} \mathbf{B}_{2,1}$
Task 6: $\mathbf{C}_{2,1} = \mathbf{C}_{2,1} + \mathbf{A}_{2,2} \mathbf{B}_{2,1}$	Task 6: $\mathbf{C}_{2,1} = \mathbf{C}_{2,1} + \mathbf{A}_{2,1} \mathbf{B}_{1,1}$
Task 7: $\mathbf{C}_{2,2} = \mathbf{A}_{2,1} \mathbf{B}_{1,2}$	Task 7: $\mathbf{C}_{2,2} = \mathbf{A}_{2,1} \mathbf{B}_{1,2}$
Task 8: $\mathbf{C}_{2,2} = \mathbf{C}_{2,2} + \mathbf{A}_{2,2} \mathbf{B}_{2,2}$	Task 8: $\mathbf{C}_{2,2} = \mathbf{C}_{2,2} + \mathbf{A}_{2,2} \mathbf{B}_{2,2}$

Output Data Decomposition: Exemplan

Problema: numararea instantelor unor itemi intr-o baza de date de tranzactii.

Output= itemset frequencies

se partitioneaza intre taskuri.

(a) Transactions (input), itemsets (input), and frequencies (output)

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,		C, D		1
	A, E, F, K, L		D, K		2
	B, C, D, G, H, L		B, C, F		0
	G, H, L		C, D, K		0
	D, E, F, K, L				
	F, G, H, L				

(b) Partitioning the frequencies (and itemsets) among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 1

Database Transactions	A, B, C, E, G, H	Itemsets	C, D	Itemset Frequency	1
	B, D, E, F, K, L		D, K		2
	A, B, F, H, L		B, C, F		0
	D, E, F, H		C, D, K		0
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 2

Output Data Decomposition: Exemplu

Analiza:

- Daca baza de tranzactii este replicata pe procese fiecare task se poate executa fara comunicatii.
- Daca baza de tranzactii este distribuita pe procese (memory ...) atunci fiecare task calculeaza frecvente partiale care trebuie ulterior agregate.

Input Data Partitioning

- Exemple: minim intr-o lista, sortare,....
- Task \Leftrightarrow partitie input
- Processe ulterioare pot agrega/ combina rezultatele partiale.

Input Data Partitioning: Exemplu

Partitioning the transactions among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		2
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		1
	F, G, H, K,		C, D		0
			D, K		1
			B, C, F		0
			C, D, K		0

task 1

Database Transactions		Itemsets	A, B, C	Itemset Frequency	0
			D, E		1
			C, F, G		0
	A, E, F, K, L		A, E		1
	B, C, D, G, H, L		C, D		1
	G, H, L		D, K		1
	D, E, F, K, L		B, C, F		0
	F, G, H, L		C, D, K		0

task 2

Partitioning Input *and* Output Data

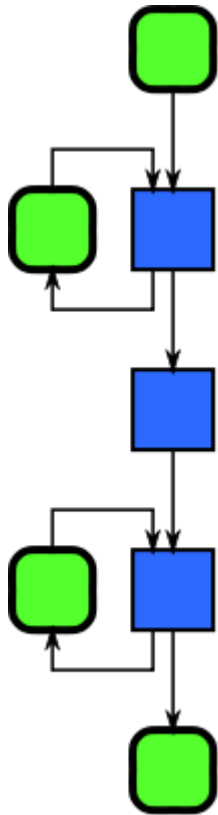
Partitioning both transactions and frequencies among the tasks

Database Transactions A, B, C, E, G, H B, D, E, F, K, L A, B, F, H, L D, E, F, H F, G, H, K,	Itemsets A, B, C D, E C, F, G A, E	Itemset Frequency 1 2 0 1
task 1		
Database Transactions A, B, C, E, G, H B, D, E, F, K, L A, B, F, H, L D, E, F, H F, G, H, K,	Itemsets C, D D, K B, C, F C, D, K	Itemset Frequency 0 1 0 0
task 2		
Database Transactions A, E, F, K, L B, C, D, G, H, L G, H, L D, E, F, K, L F, G, H, L	Itemsets A, B, C D, E C, F, G A, E	Itemset Frequency 0 1 0 1
task 3		
Database Transactions A, E, F, K, L B, C, D, G, H, L G, H, L D, E, F, K, L F, G, H, L	Itemsets C, D D, K B, C, F C, D, K	Itemset Frequency 1 1 0 0
task 4		

The Owner Computes Rule

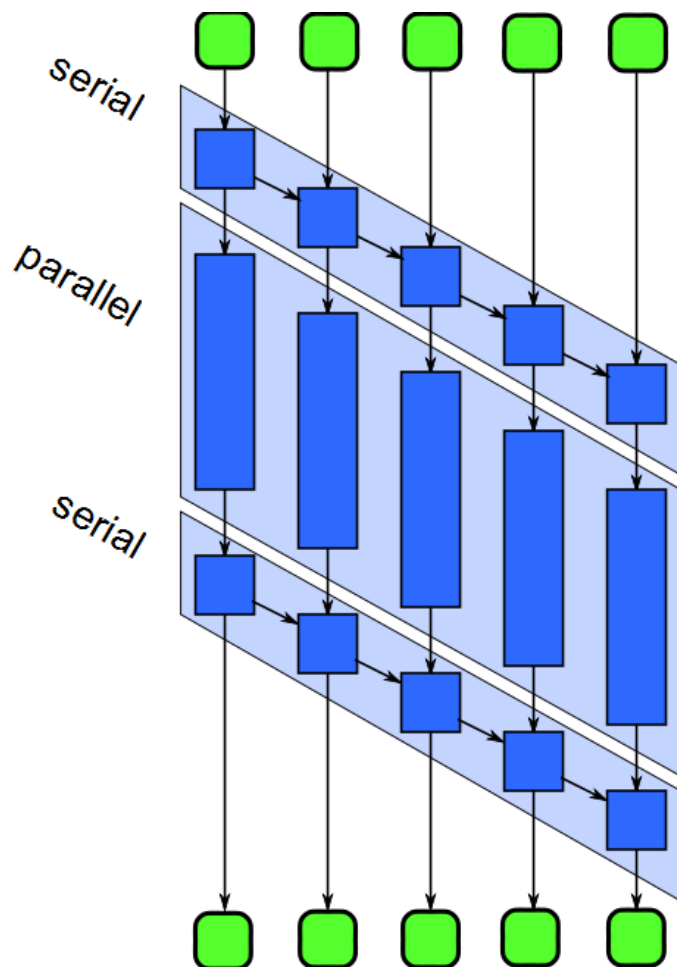
- Procesul care are data asignata lui este responsabil de calculele asociate acelei date
- Diferente:
 - input data decomposition
 - output data decomposition
- Obs: similar sablon GRASP: Expert

Pipeline - sablon de programare paralela



- *Pipeline* – o secventa de stagii care transforma un flux de date
- Unele stagii pot sa stocheze stare
- Datele pot fi “consumate” si produse incremental.

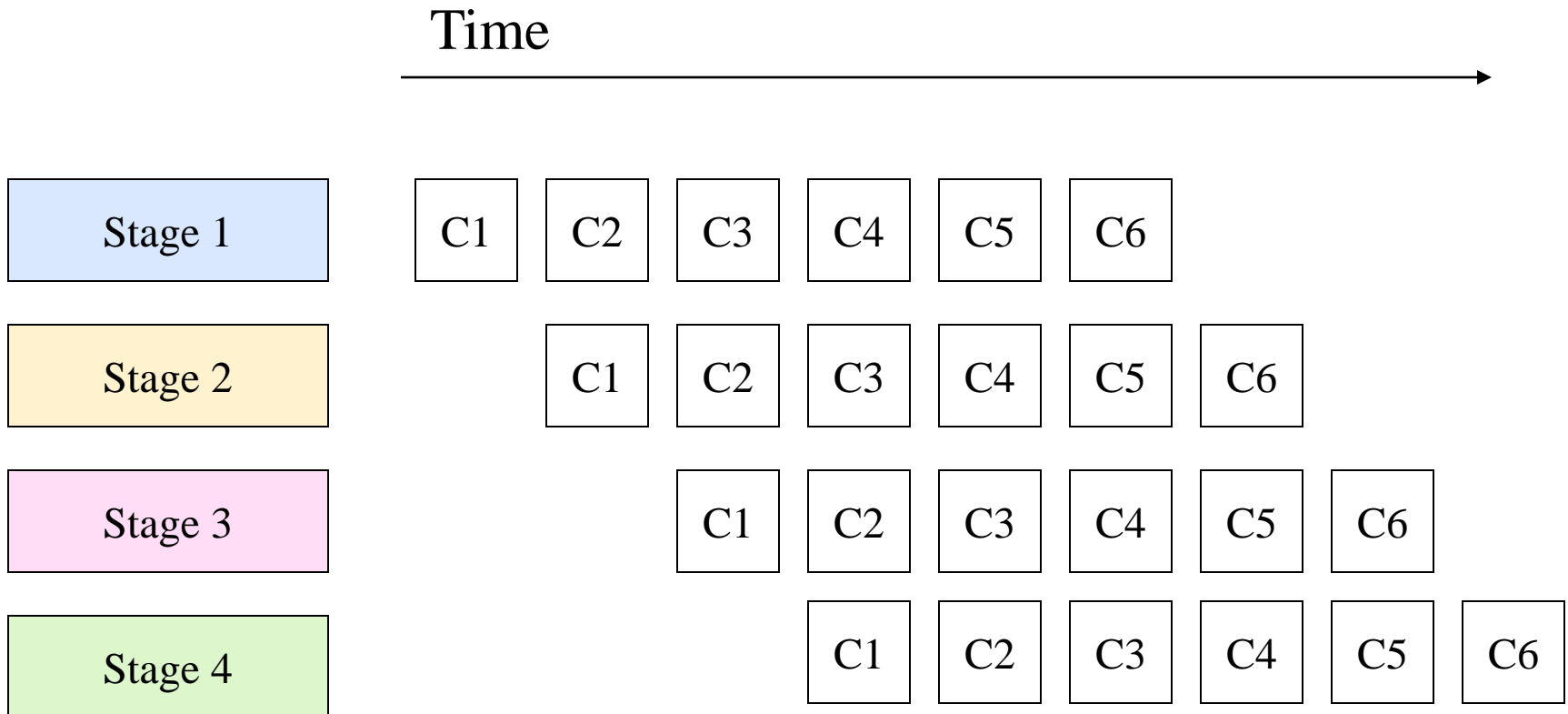
Pipeline



- Paralelizarea pipeline se face prin
 1. Executia diferitelor stagii in paralel
 2. Executia multiplelor copii ale stagiilor fara stare in paralel

Pipeline

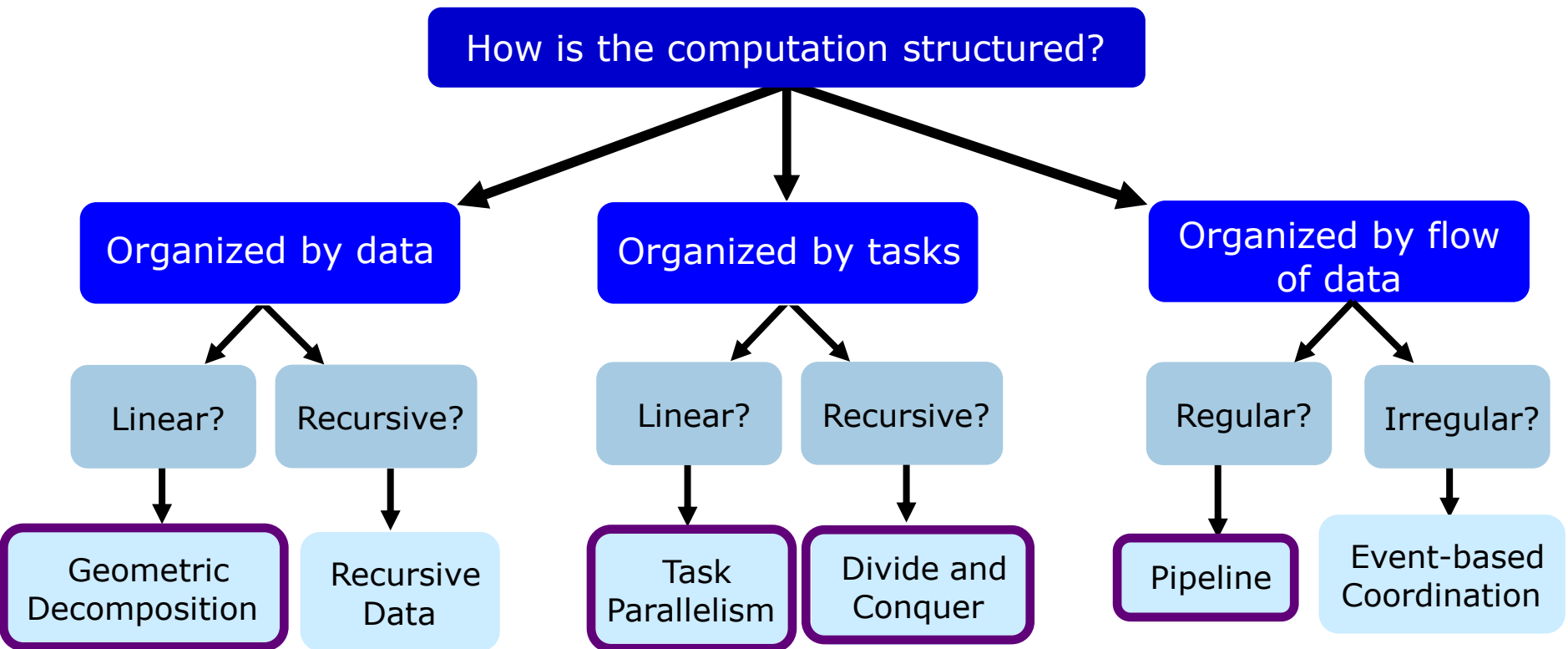
A series of ordered but independent computation stages need to be applied on data, where each output of a computation becomes input of subsequent computation.



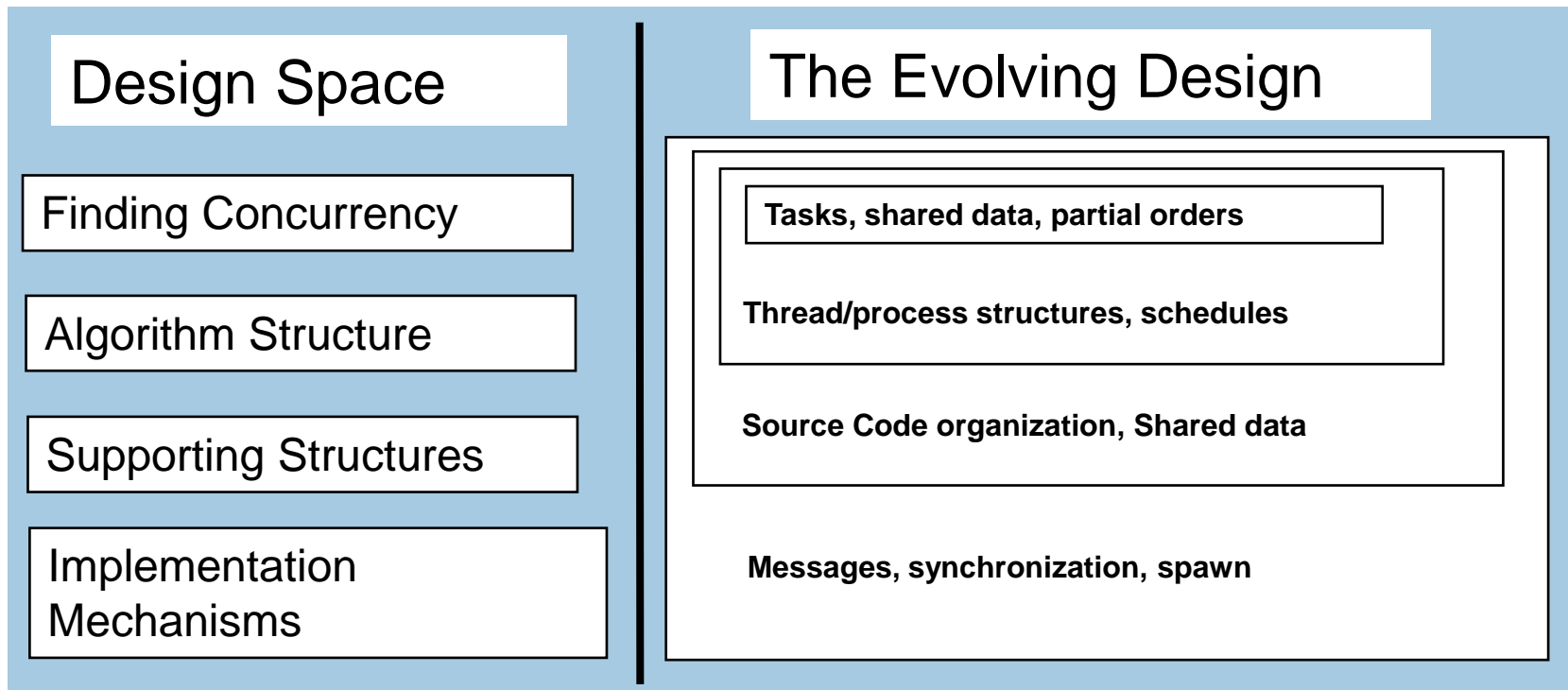
Sumar - descompunere

- Nu exista doar o singura reteta pentru descompunere
 - Se pot aplica un set de tehnici comune pe o clasa de probleme mai vasta.
-
- *data decomposition (geometric decomposition)*
 - *recursive decomposition*
 - *exploratory decomposition*
 - *speculative decomposition*
-
- *Pipelines... (se poate obtine prin descompunerea fluxului de date)*

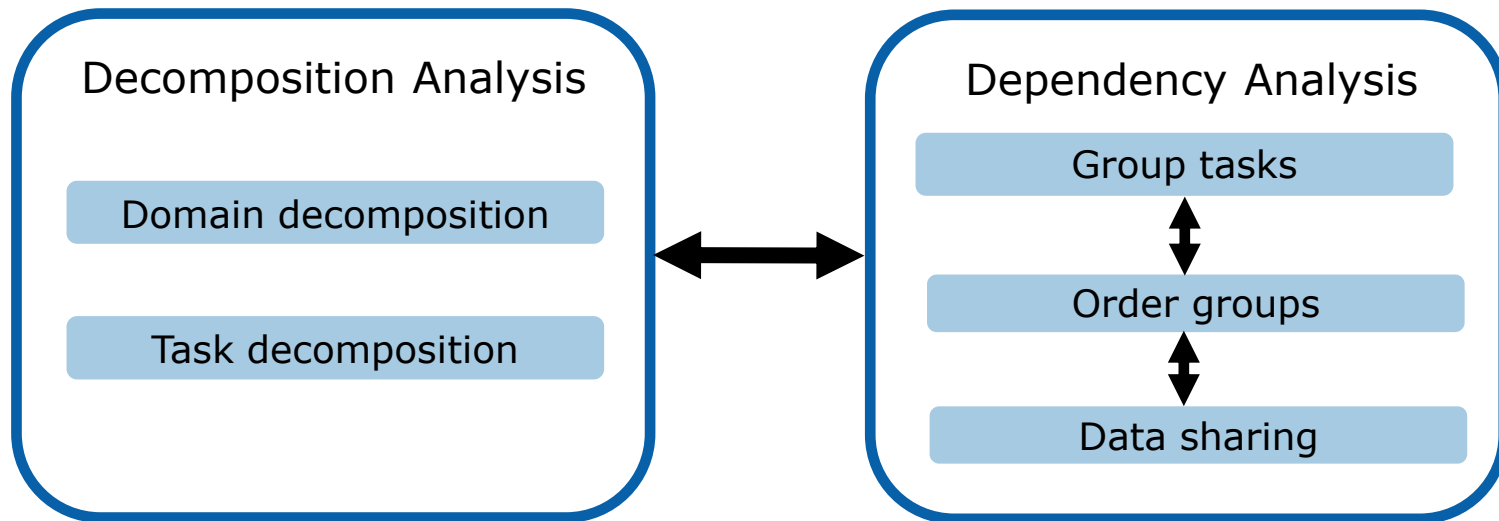
Algorithm Structure Design Space



Patterns for Parallel Programming. Mattson, Sanders, and Massingill (2005).



Cautare in spatiul de proiectare concurenta



Applications

Structural Patterns

Pipe-and-Filter

Agent-and-Repository

Process-Control

Event-Based/Implicit-Invocation

Arbitrary-Static-Task-Graph

Model-View-Controller

Iterative-Refinement

Map-Reduce

Layered-Systems

Puppeteer

Computational Patterns

Graph-Algorithms

Dynamic-Programming

Dense-Linear-Algebra

Sparse-Linear-Algebra

Unstructured-Grids

Structured-Grids

Graphical-Models

Finite-State-Machines

Backtrack-Branch-and-Bound

N-Body-Methods

Circuits

Spectral-Methods

Monte-Carlo

Parallel Algorithm Strategy Patterns

Task-Parallelism

Divide and Conquer

Data-Parallelism

Pipeline

Discrete-Event

Geometric-Decomposition

Speculation

Implementation Strategy Patterns

SPMD

Fork/Join

Program structure

Kernel-Par.

Loop-Par.

Vector-Par.

Actors

Work-pile

Shared-Queue

Shared-Map

Shared-Data

Partitioned-Array

Partitioned-Graph

Data structure

Parallel Execution Patterns

Coordinating Processes

Stream processing

Shared Address Space Threads

Task Driven Execution

Referinte:

``Introduction to Parallel Computing''

Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar
2003

Ian Foster

Designing and Building Parallel Programs, Addison Wesley, 2, Addison-Wesley Inc.,
Argonne National Laboratory (<http://www.mcs.anl.gov/~itf/dbpp/>)

Parallel Programming Patterns

Eun-Gyu Ki, 2004

Patterns for Parallel
Programming. Mattson,
Sanders, and Massingill
(2005).