Principle Of Parallel Algorithm Design (cont.)

Alexandre David

B2-206

Today

- Characteristics of Tasks and Interactions (3.3).
- Mapping Techniques for Load Balancing (3.4).
- Methods for Containing Interaction Overhead (3.5).
- Parallel Algorithm Models (3.6).

So Far...

- Decomposition techniques.
  - Identify tasks.
  - Analyze with task dependency & interaction graphs.
  - Map tasks to processes.
- Now properties of tasks that affect a good mapping.
  - Task generation, size, and size of data.

Task Generation

- Static task generation.
  - Tasks are known beforehand.
  - Apply to well-structured problems.
- Dynamic task generation.
  - Tasks generated on-the-fly.
  - Tasks & task dependency graph not available beforehand.

Task Sizes

- Relative amount of time for completion.
  - Uniform – same size for all tasks.
    - Matrix multiplication.
  - Non-uniform.
    - Optimization & search problems.

Size of Data Associated with Tasks

- Important because of locality reasons.
- Different types of data with different sizes
  - Input/output/intermediate data.
- Size of context – cheap or expensive communication with other tasks.

Characteristics of Task Interactions

- Static interactions.
  - Tasks and interactions known beforehand.
  - And interaction at pre-determined times.
- Dynamic interactions.
  - Timing of interaction unknown.
  - Or set of tasks not known in advance.

Characteristics of Task Interactions

- Regular interactions.
  - The interaction graph follows a pattern.
- Irregular interactions.
  - No pattern.

Example: Image Dithering

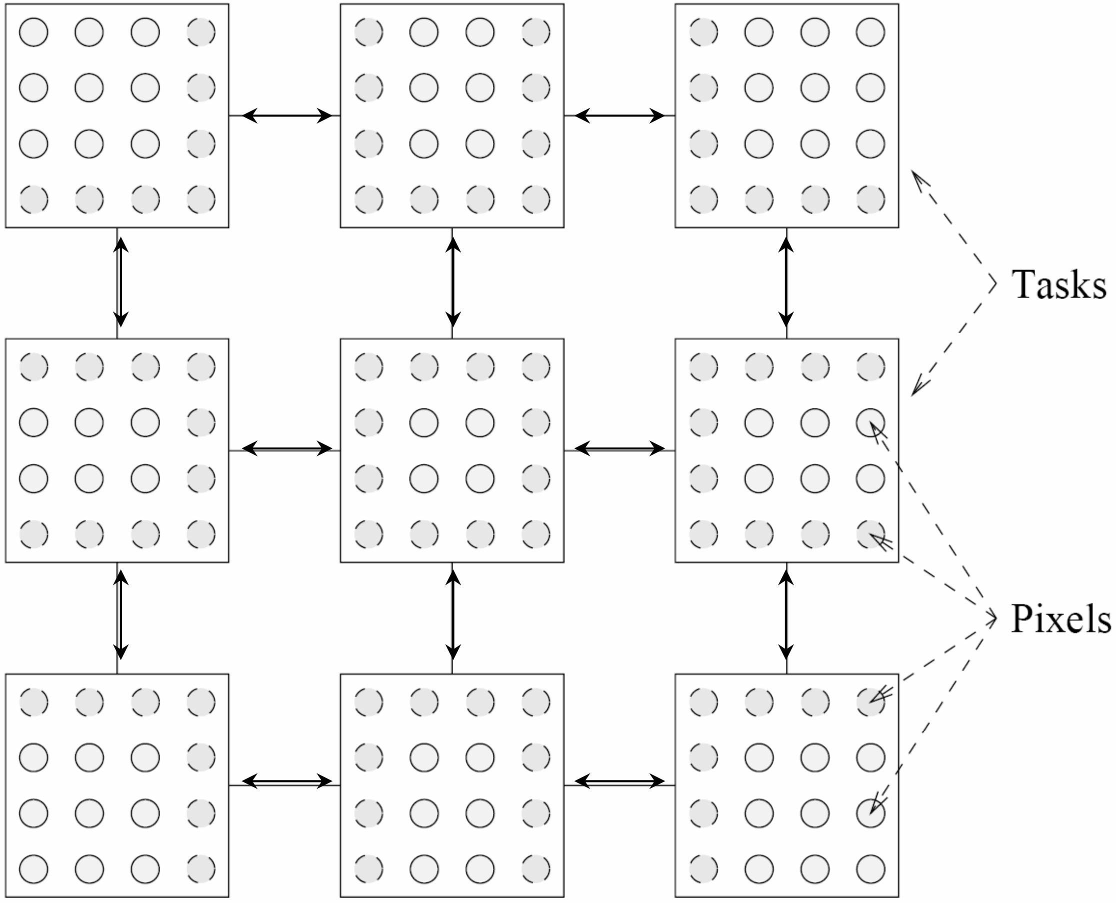


Figure 3.22 The regular two-dimensional task-interaction graph for image dithering. The pixels with dotted outline require color values from the boundary pixels of the neighboring tasks.

Example: Sparse Matrix*Vector

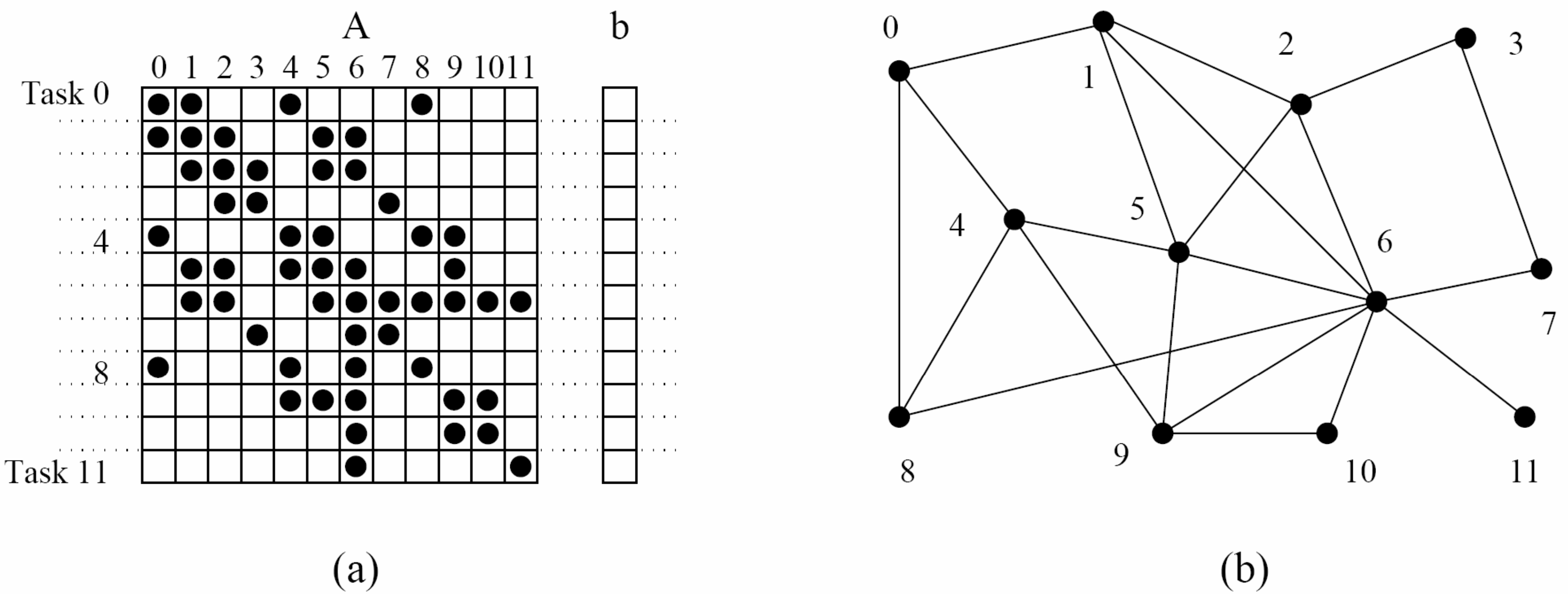


Figure 3.6 A decomposition for sparse matrix-vector multiplication and the corresponding task-interaction graph. In the decomposition Task i computes $\sum_{0 \leq j \leq 11, A[i,j] \neq 0} A[i, j].b[j]$.

Characteristics of Task Interactions

- Data sharing interactions:
  - Read-only interactions.
    - Read only data associated with *other* tasks.
  - Read-write interactions.
    - Read & modify data of *other* tasks.

Characteristics of Task Interactions

- One-way interactions.
  - Only one task initiates and completes the communication ***without*** interrupting the other one.
- Two-way interactions.
  - Producer – consumer model.

Mapping Techniques for Load Balancing

- Map tasks onto processes.
- Goal: minimize overheads.
  - Communication.
  - Idling.
- Uneven load distribution may cause idling.
  - Constraints from task dependency $\rightarrow$ wait for other tasks.

Example

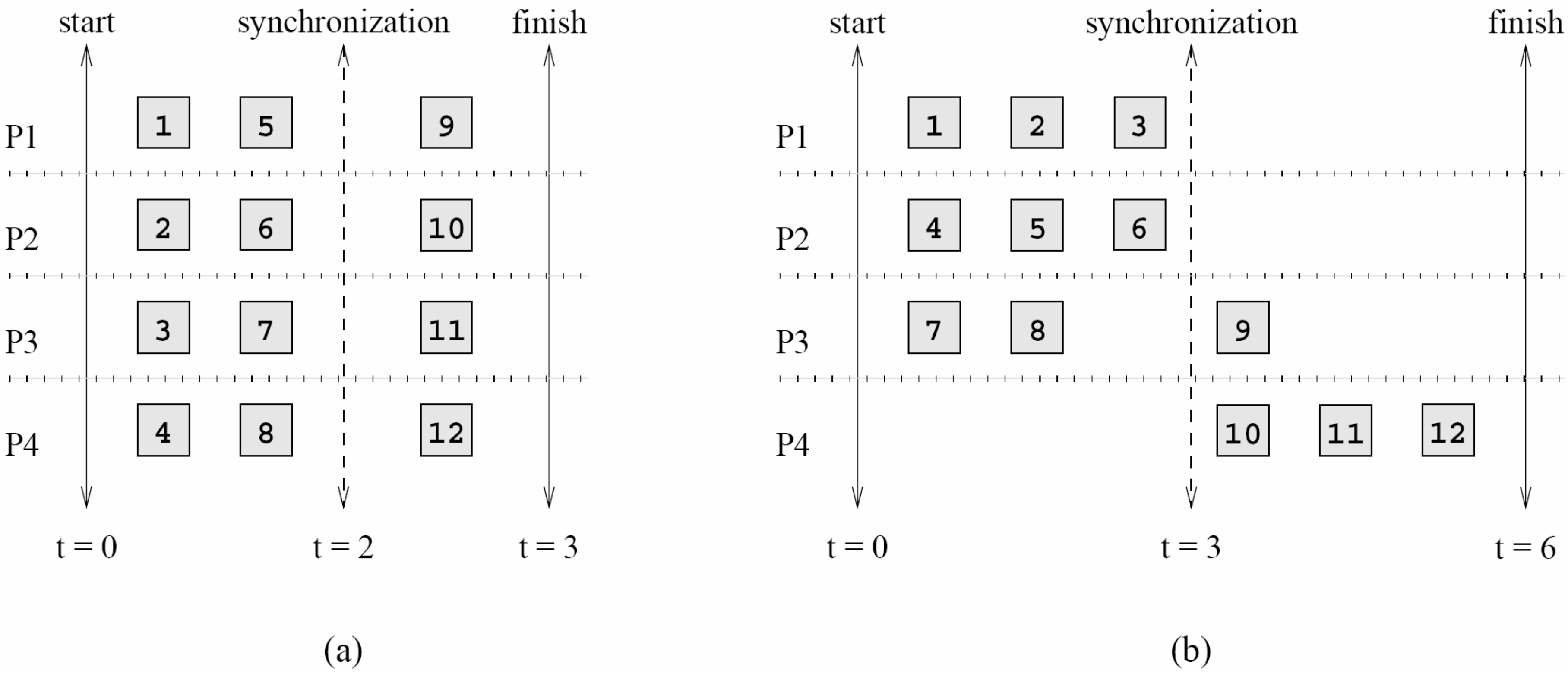


Figure 3.23 Two mappings of a hypothetical decomposition with a synchronization.

Mapping Techniques

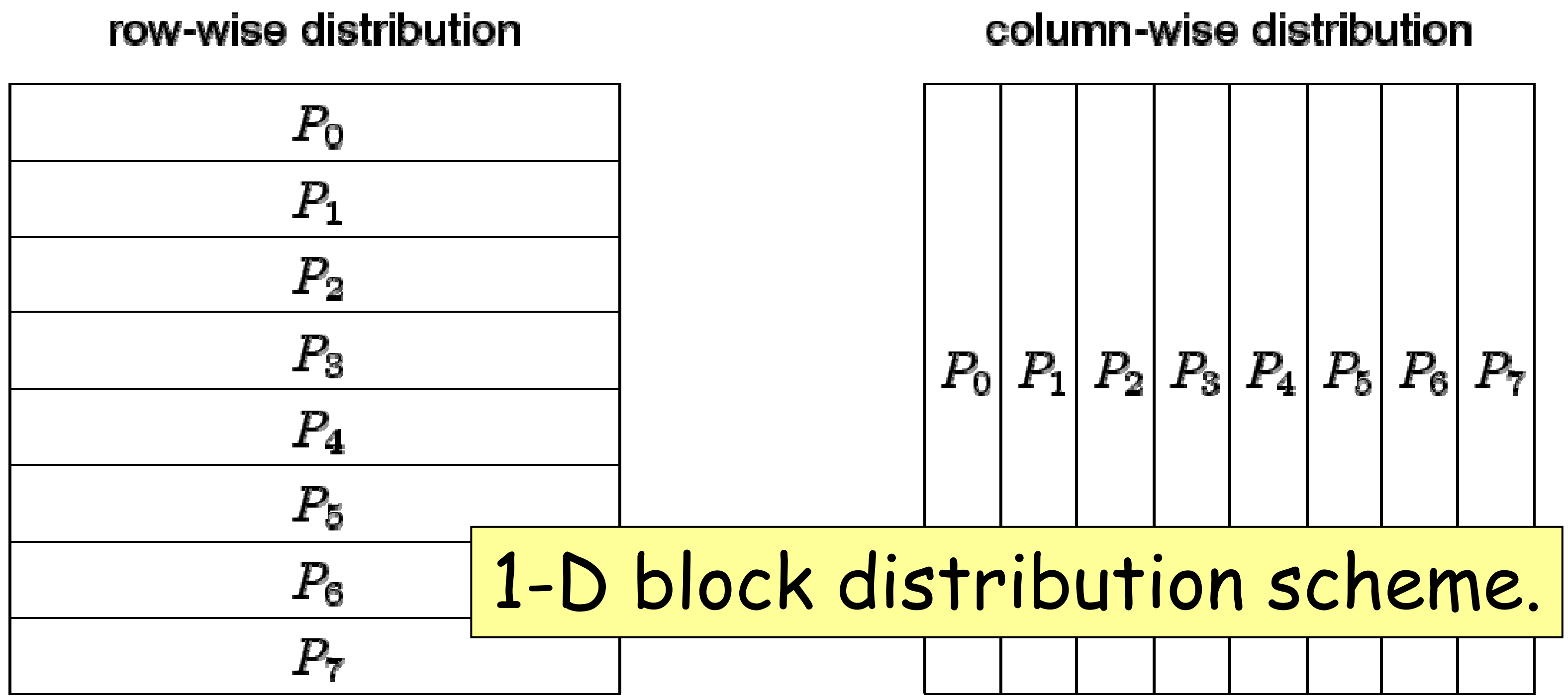
- Static mapping.
  - NP-complete problem for non-uniform tasks.
  - Large data compared to computation.
- Dynamic mapping.
  - Dynamically generated tasks.
  - Task size unknown.

Schemes for Static Mapping

- Mappings based on data partitioning.
- Mappings based on task graph partitioning.
- Hybrid mappings.

Array Distribution Scheme

- Combine with “owner computes” rule to partition into sub-tasks.



Block Distribution cont.

P_0	P_1	P_2	P_3
P_4	P_5	P_6	P_7
P_8	P_9	P_{10}	P_{11}
P_{12}	P_{13}	P_{14}	P_{15}

(a)

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}

(b)

Generalize to higher dimensions: 4x4, 2x8.

Example: Matrix*Matrix

- Partition output of C=A*B.
- Each entry needs the same amount of computation.
- Blocks on 1 or 2 dimensions.
- Different data sharing patterns.
- *Higher dimensional* distributions
  - means we can use *more processes*.
  - sometimes *reduces* interaction.

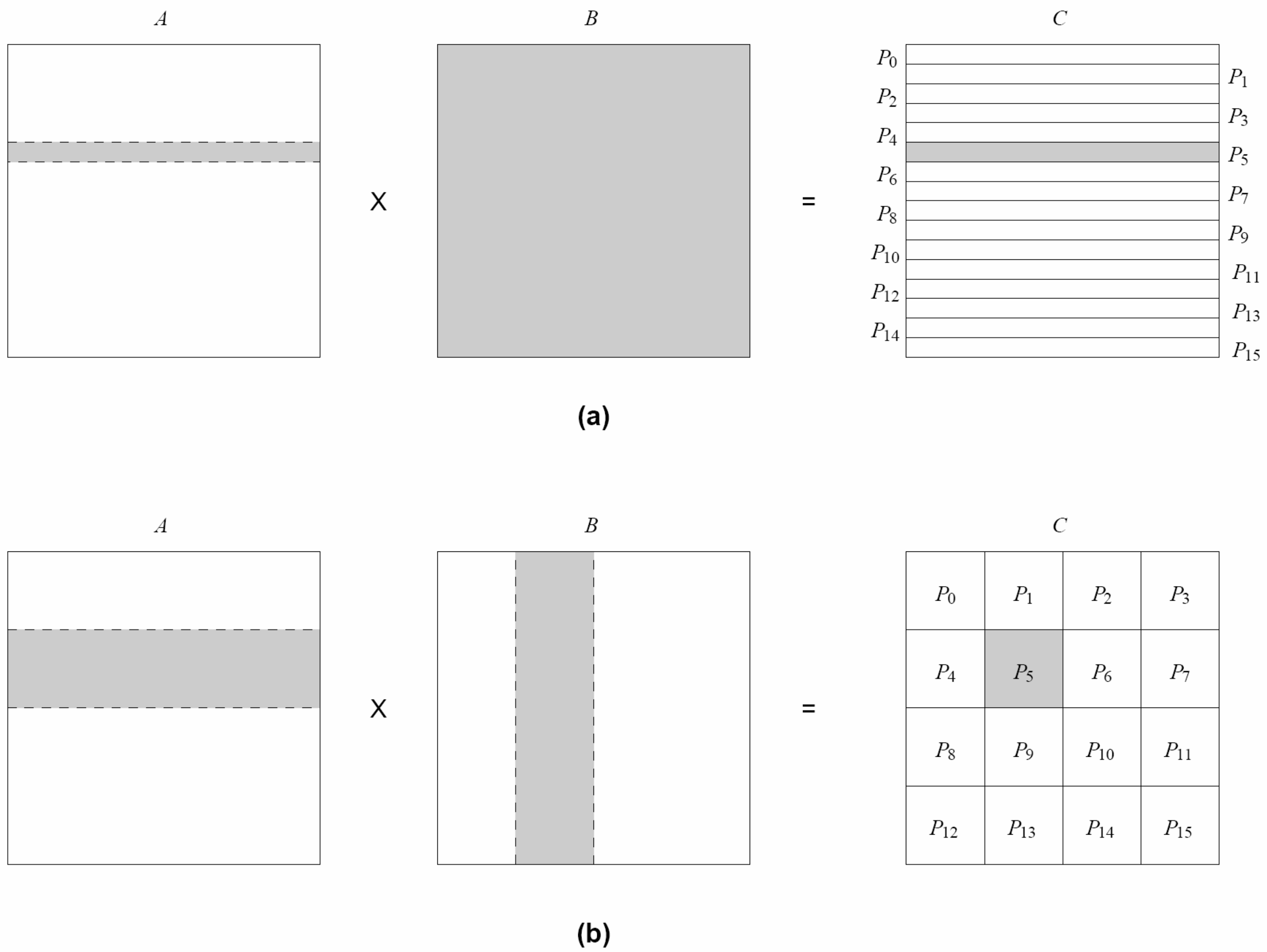
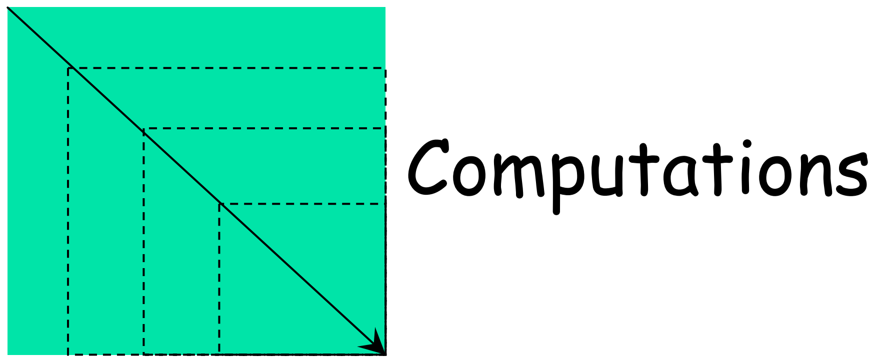


Figure 3.26 Data sharing needed for matrix multiplication with (a) one-dimensional and (b) two-dimensional partitioning of the output matrix. Shaded portions of the input matrices A and B are required by the process that computes the shaded portion of the output matrix C.

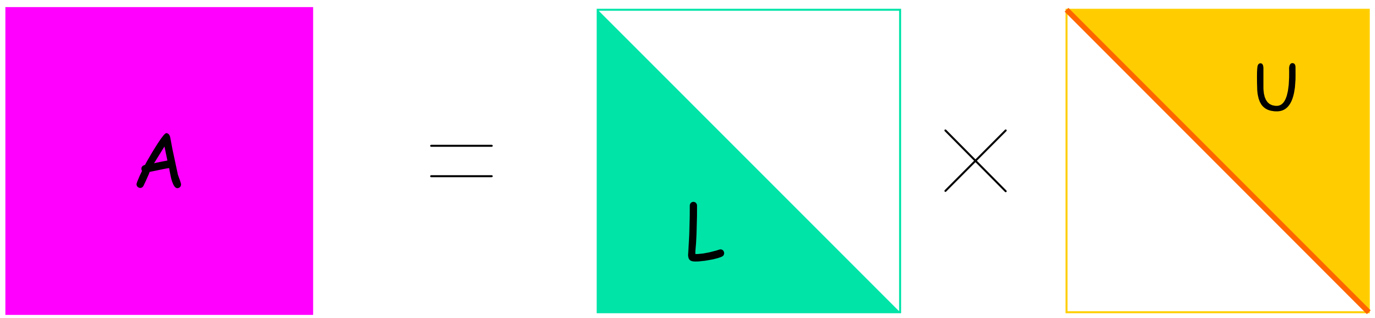
Imbalance Problem

- If the amount of *computation* associated with data *varies* a lot then *block decomposition* leads to *imbalances*.
- Example: LU factorization (or Gaussian elimination).



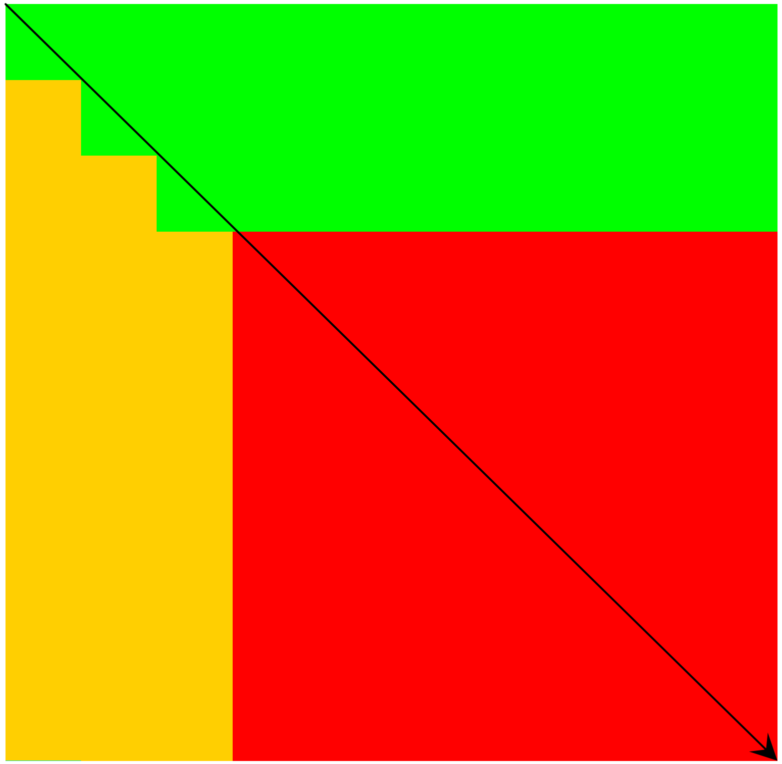
LU Factorization

- Non singular square matrix A (invertible).
- A = L*U.
- Useful for solving linear equations.



LU Factorization

In practice we work on *A*.



N steps

LU Algorithm

```
Proc LU(A)
begin
   for k := 1 to n-1 do
      for j := k+1 to n do
         A[j,k] := A[j,k]/A[k,k]
      endfor
      for j := k+1 to n do
         for i := k+1 to n do
            A[i,j] := A[i,j] - A[i,k]*A[k,j]
         endfor
      endfor
   endfor
end
```

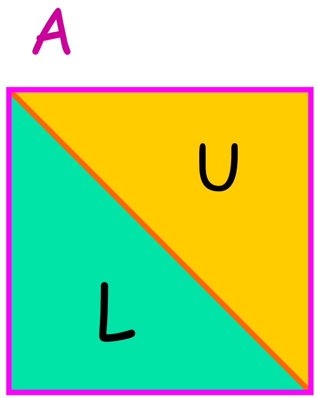
U[k,k]

L[j,k]

L[i,k] U[k,j]

Normalize L

U[k,j] := A[k,j]/L[k,k]



Another Variant

```
for k := 1 to n-1 do
   for j := k+1 to n do
      A[k,j] := A[k,j]/A[k,k]
      for i := k+1 to n do
         A[i,j] := A[i,j] - A[i,k]*A[k,j]
      endfor
   endfor
endfor
```

Decomposition

$$\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} L_{1,1} & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix} \cdot \begin{pmatrix} U_{1,1} & U_{1,2} & U_{1,3} \\ 0 & U_{2,2} & U_{2,3} \\ 0 & 0 & U_{3,3} \end{pmatrix}$$

1: $A_{1,1} \rightarrow L_{1,1}U_{1,1}$
2: $L_{2,1} = A_{2,1}U_{1,1}^{-1}$
3: $L_{3,1} = A_{3,1}U_{1,1}^{-1}$
4: $U_{1,2} = L_{1,1}^{-1}A_{1,2}$
5: $U_{1,3} = L_{1,1}^{-1}A_{1,3}$

6: $A_{2,2} = A_{2,2} - L_{2,1}U_{1,2}$
7: $A_{3,2} = A_{3,2} - L_{3,1}U_{1,2}$
8: $A_{2,3} = A_{2,3} - L_{2,1}U_{1,3}$
9: $A_{3,3} = A_{3,3} - L_{3,1}U_{1,3}$
10: $A_{2,2} \rightarrow L_{2,2}U_{2,2}$

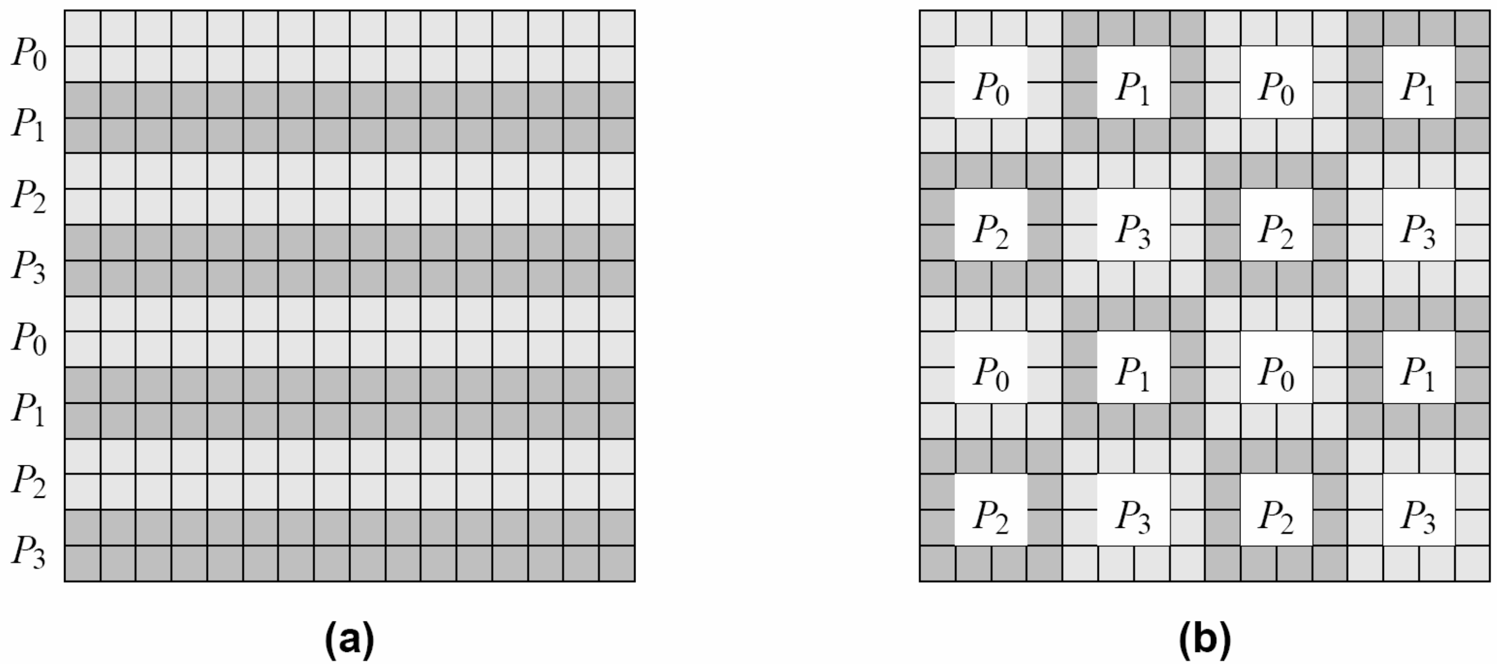
11: $L_{3,2} = A_{3,2}U_{2,2}^{-1}$
12: $U_{2,3} = L_{2,2}^{-1}A_{2,3}$
13: $A_{3,3} = A_{3,3} - L_{3,2}U_{2,3}$
14: $A_{3,3} \rightarrow L_{3,3}U_{3,3}$

Figure 3.27 A decomposition of LU factorization into 14 tasks.

Cyclic and Block-Cyclic Distributions

- Idea:
  - Partition an array into many *more blocks than available processes*.
  - Assign partitions (tasks) to processes in a round-robin manner.
- $\rightarrow$ each process gets several non adjacent blocks.

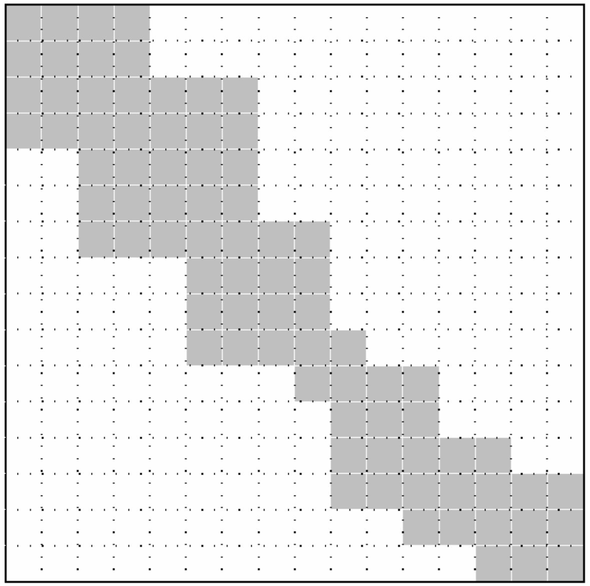
Block-Cyclic Distributions



a) Partition 16x16 into 2*4 groups of 2 rows.
   αp groups of $n/\alpha p$ rows.

b) Partition 16x16 into square blocks of size 4*4 distributed on 2*2 processes.
   $\alpha^2 p$ groups of $n/\alpha^2 p$ squares.

Randomized Distributions



(a)

P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
P_4	P_5	P_6	P_7	P_4	P_5	P_6	P_7
P_8	P_9	P_{10}	P_{11}	P_8	P_9	P_{10}	P_{11}
P_{12}	P_{13}	P_{14}	P_{15}	P_{12}	P_{13}	P_{14}	P_{15}
P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
P_4	P_5	P_6	P_7	P_4	P_5	P_6	P_7
P_8	P_9	P_{10}	P_{11}	P_8	P_9	P_{10}	P_{11}
P_{12}	P_{13}	P_{14}	P_{15}	P_{12}	P_{13}	P_{14}	P_{15}

(b)

Irregular distribution with regular mapping!
Not good.

1-D Randomized Distribution

$$V = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$$

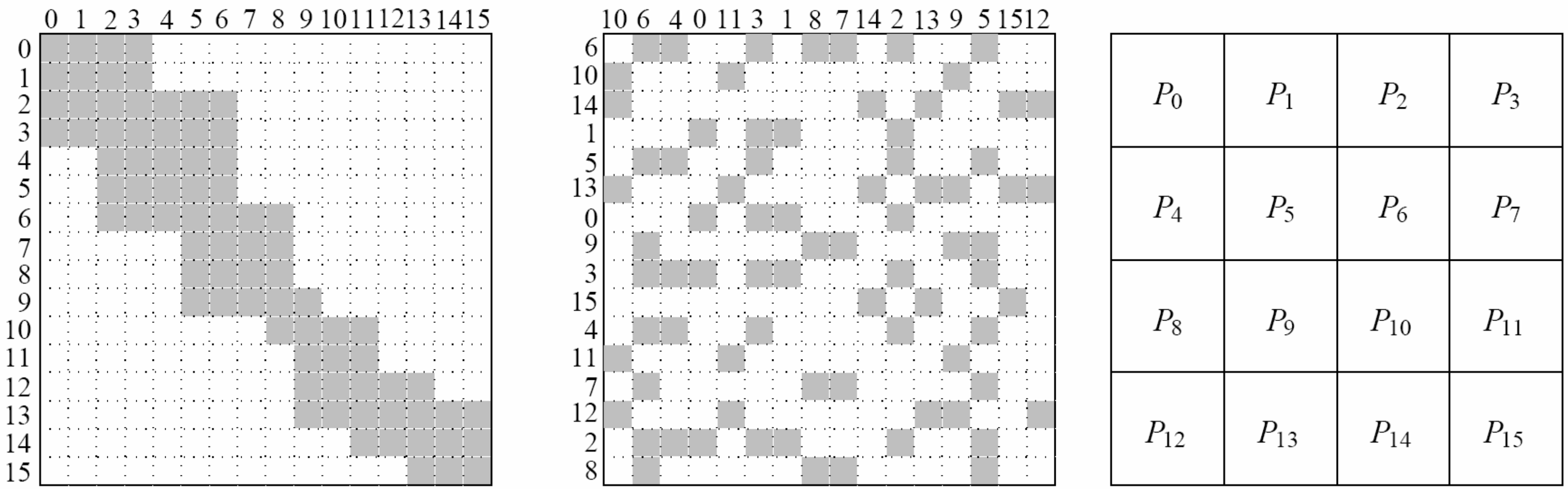
Permutation

$$\mathrm{random}(V) = [8, 2, 6, 0, 3, 7, 11, 1, 9, 5, 4, 10]$$

$$\mathrm{mapping} = \underbrace{8\ \ 2\ \ 6}_{P_0}\ \ \underbrace{0\ \ 3\ \ 7}_{P_1}\ \ \underbrace{11\ \ 1\ \ 9}_{P_2}\ \ \underbrace{5\ \ 4\ \ 10}_{P_3}$$

Figure 3.32 A one-dimensional randomized block mapping of 12 blocks onto four process (i.e., $\alpha = 3$).

2-D Randomized Distribution



(a) (b) (c)

2-D block random distribution.

Block mapping.

Graph Partitioning

- For sparse data structures and data dependent interaction patterns.
  - Numerical simulations. Discretize the problem and represent it as a mesh.
- Sparse matrix: assign equal number of nodes to processes & minimize interaction.
- Example: simulation of dispersion of a water contaminant in Lake Superior.

Discretization

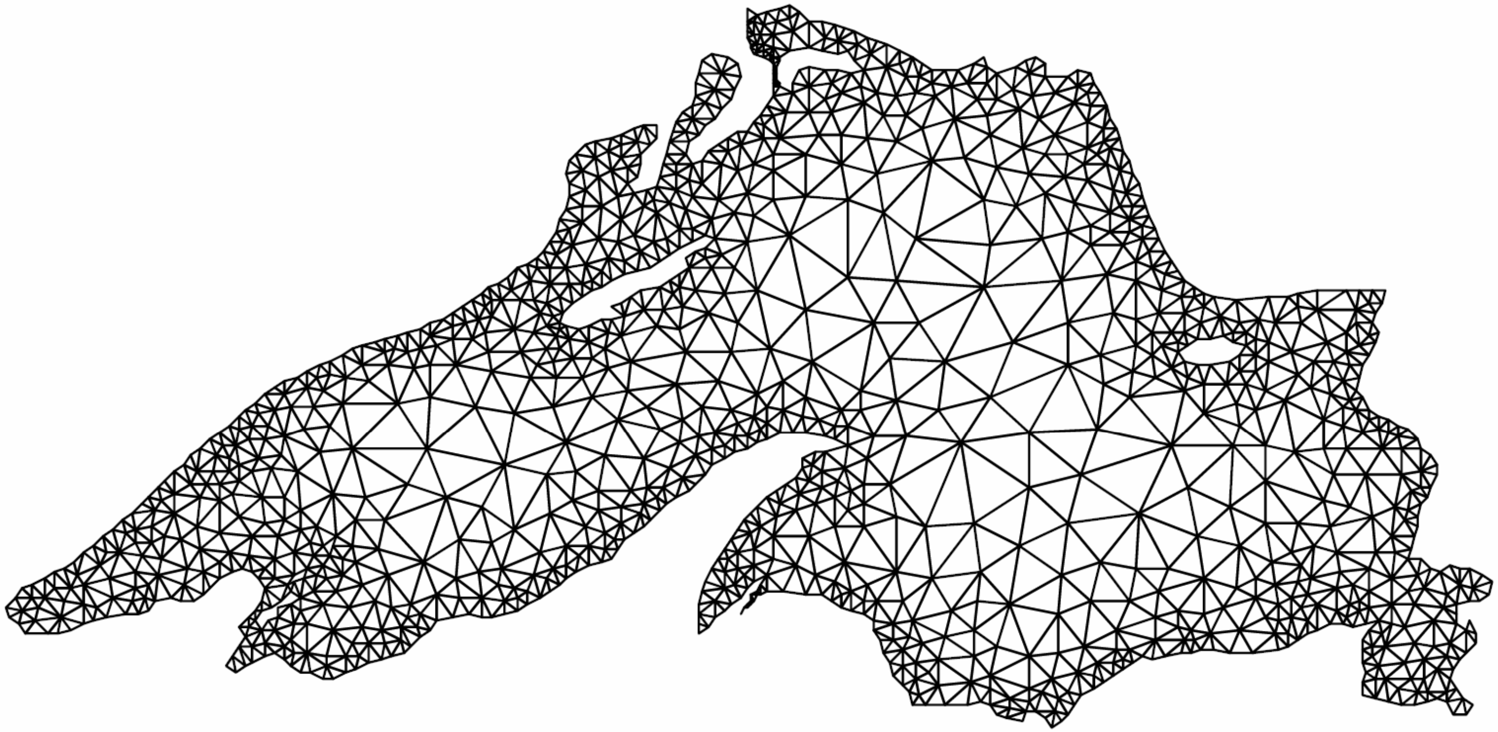
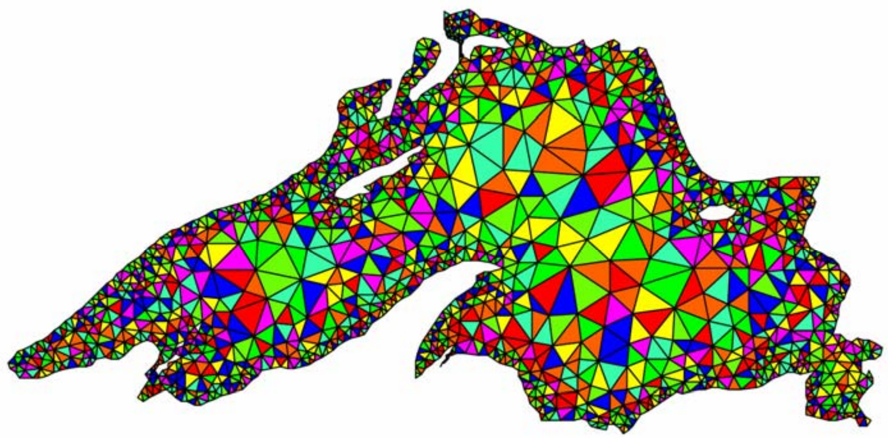
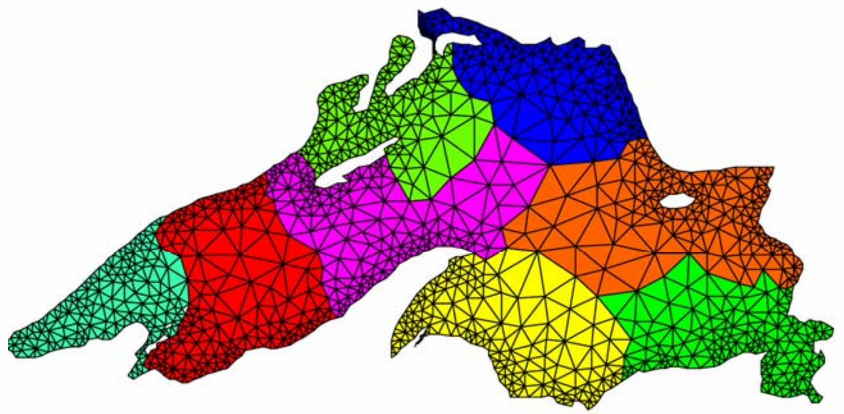


Figure 3.34 A mesh used to model Lake Superior.

Partitioning Lake Superior



Random partitioning.

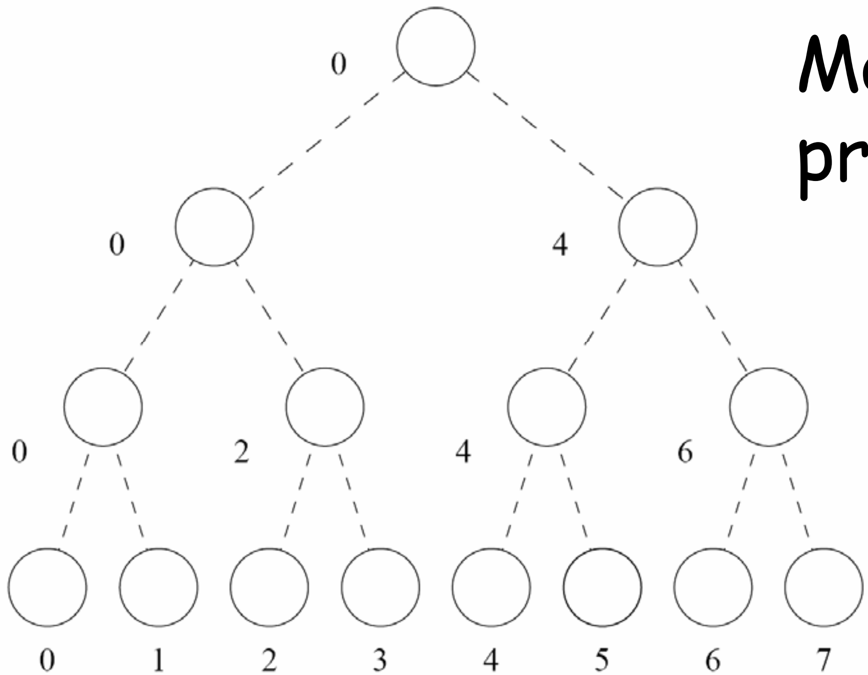


Partitioning with minimum edge cut.

Finding an exact optimal partitioning is an NP-complete problem.

Mappings Based on Task Partitioning

- Partition the task dependency graph.
  - Good when static task dependency graph with known task sizes.



Mapping on 8 processes.

Sparse Matrix*Vector

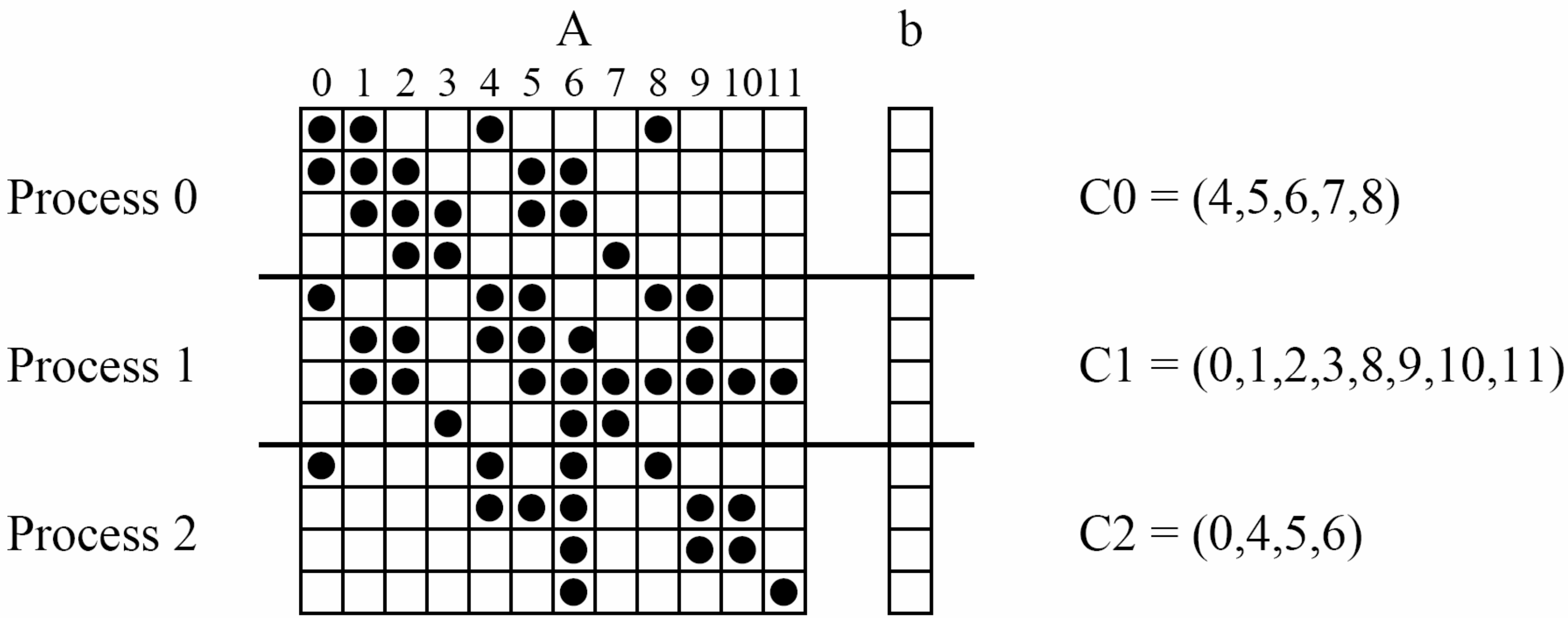


Figure 3.38 A mapping for sparse matrix-vector multiplication onto three processes. The list Ci contains the indices of b that Process i needs to access from other processes.

Sparse Matrix*Vector

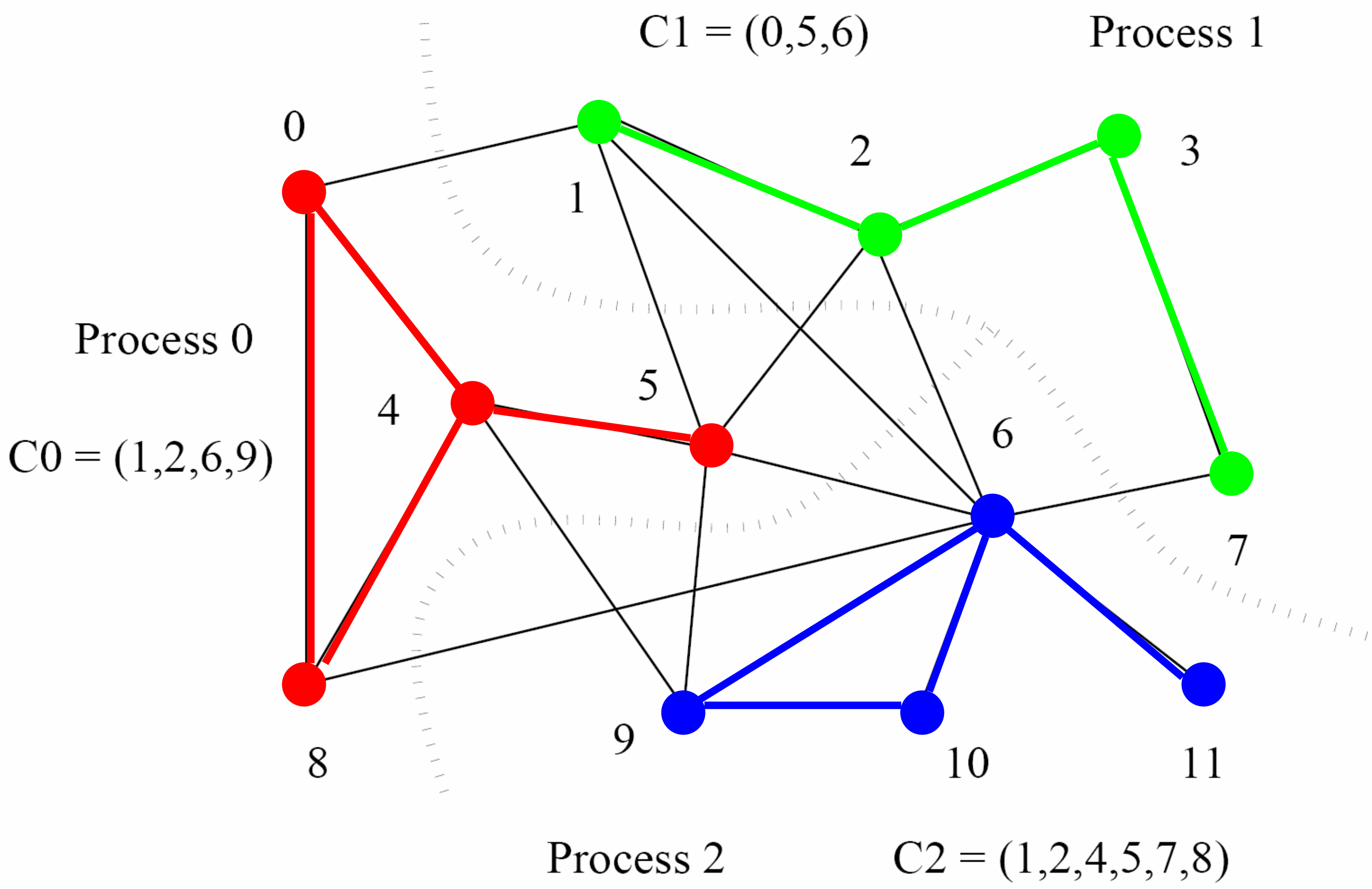


Figure 3.39 Reducing interaction overhead in sparse matrix-vector multiplication by partitioning the task-interaction graph.

Hierarchical Mappings

- Combine several mapping techniques in a structured (hierarchical) way.
- Task mapping of a binary tree (quicksort) does not use all processors.
  - Mapping based on task dependency graph (hierarchy) & block.

Binary Tree -> Hierarchical Block Mapping

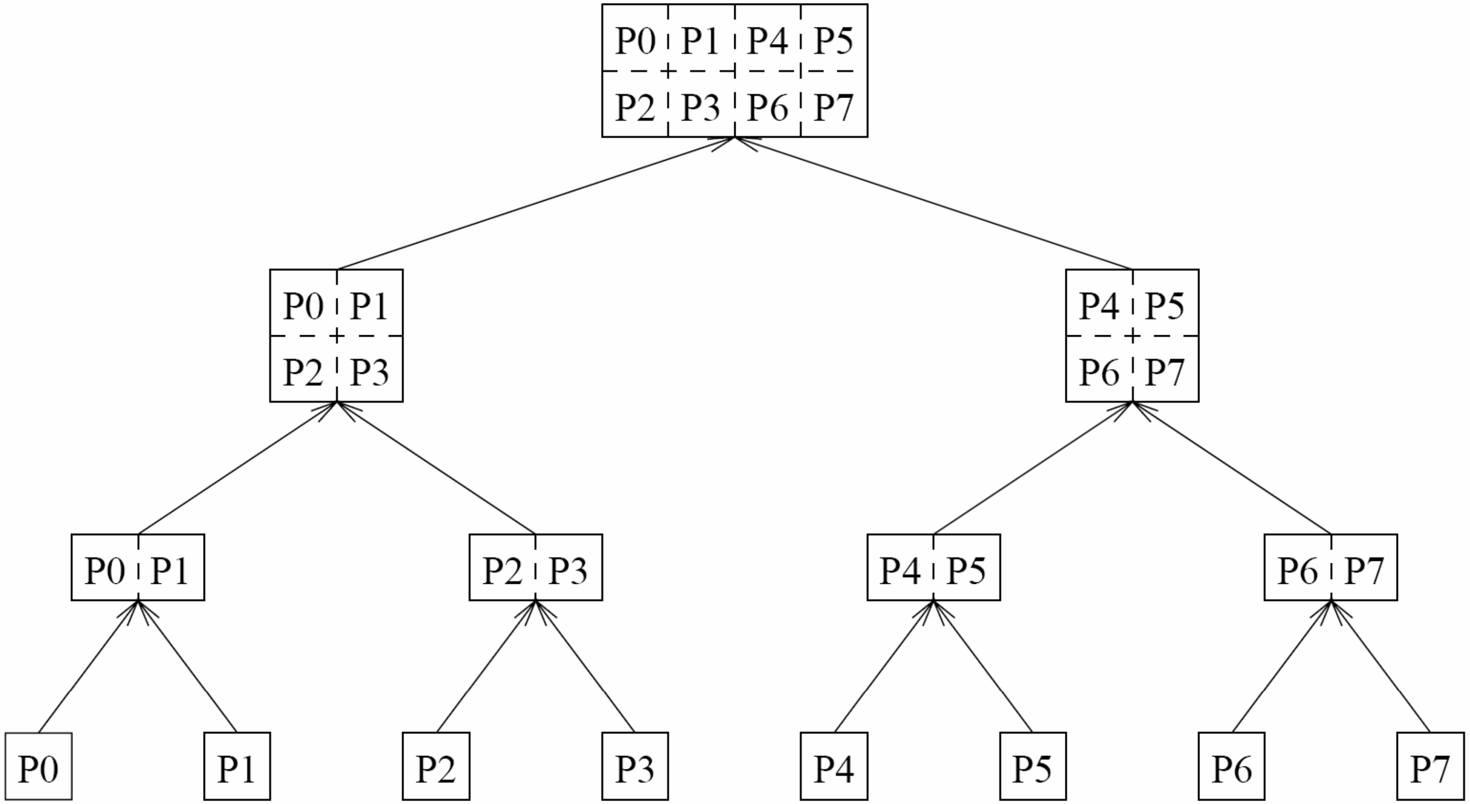


Figure 3.40 An example of hierarchical mapping of a task-dependency graph. Each node represented by an array is a supertask. The partitioning of the arrays represents subtasks, which are mapped onto eight processes.

Schemes for Dynamic Mapping

- Centralized Schemes.
  - Master manages pool of tasks.
  - Slaves obtain work.
  - Limited scalability.
- Distributed Schemes.
  - Processes *exchange tasks* to balance work.
  - Not simple, many issues.

Minimizing Interaction Overheads

- Maximize data locality.
  - Minimize volume of data-exchange.
  - Minimize frequency of interactions.
- Minimize contention and hot spots.
  - Share a link, same memory block, etc...
  - Re-design original algorithm to change the interaction pattern.

Minimizing Interaction Overheads

- Overlapping computations with interactions – to reduce idling.
  - Initiate interactions in advance.
  - Non-blocking communications.
  - Multi-threading.
- Replicating data or computation.
- Group communication instead of point to point.
- Overlapping interactions.

Overlapping Interactions

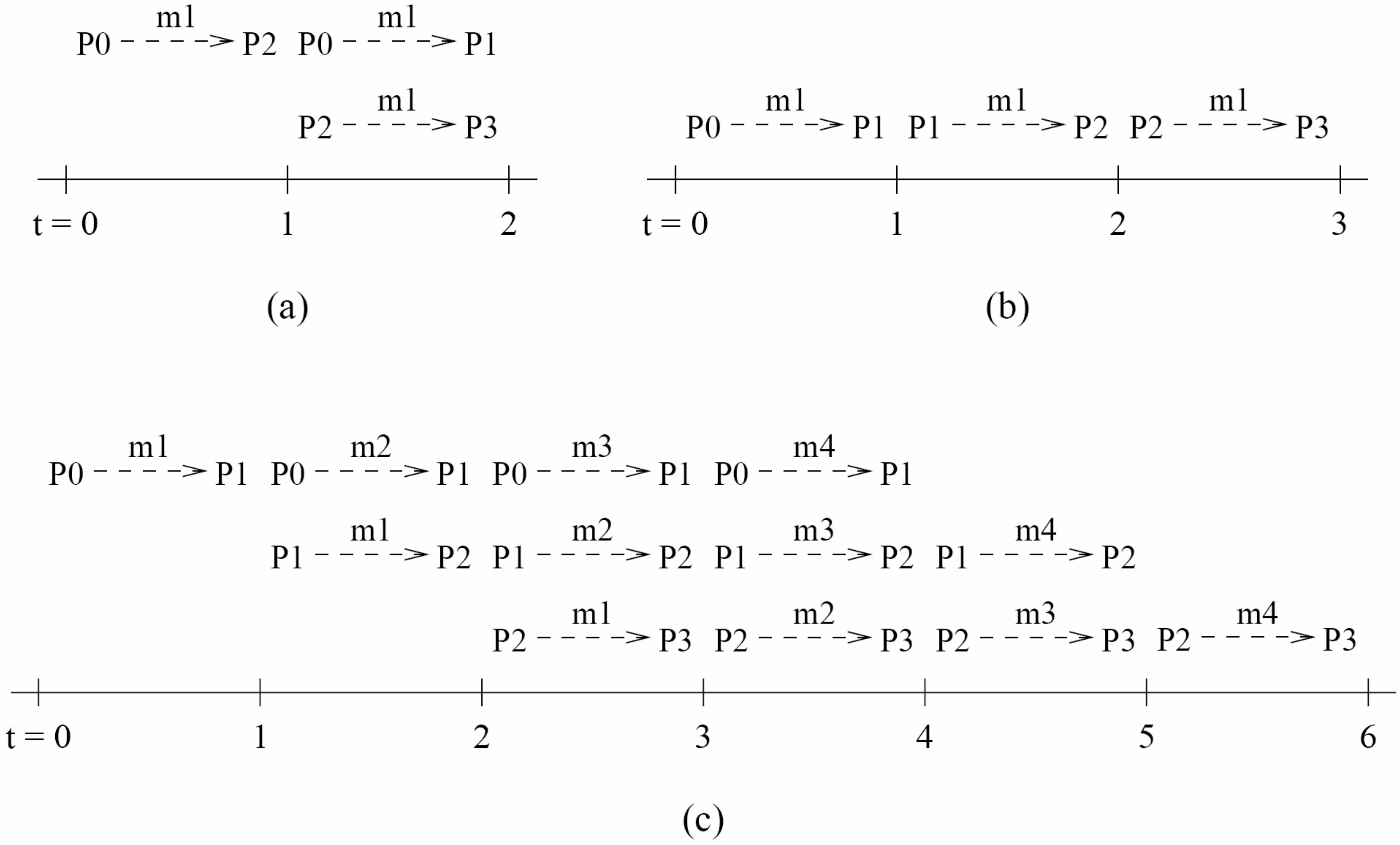


Figure 3.41 Illustration of overlapping interactions in broadcasting data from one to four processes.

Parallel Algorithm Models

- Data parallel model.
  - Tasks statically mapped.
  - Similar operations on different data.
    - SIMD.
- Task graph model.
  - Start from task dependency graph.
  - Use task interaction graph to promote locality.

Parallel Algorithm Models

- Work pool (or task pool) model.
  - No pre-mapping – centralized or not.
- Master-slave model.
  - Master generates work for slaves – allocation static or dynamic.
- Pipeline or producer – consumer model.
  - Stream of data traverses processes – stream parallelism.