TEXAS TECH UNIVERSITY

# CS4379: Parallel and Concurrent Programming
# CS5379: Parallel Processing

## Lecture 11

**Dr. Yong Chen**

**Associate Professor**

**Computer Science Department**

**Texas Tech University**

# Course Info

- **Lecture Time**: TR, 12:30-1:50
- **Lecture Location**: ECE 217
- **Sessions**: CS4379-001, CS4379-002, CS5379-001, CS5379-D01
- **Instructor:** Yong Chen, Ph.D., Associate Professor
- **Email:** yong.chen@ttu.edu
- **Phone:** 806-834-0284
- **Office**: Engineering Center 315
- **Office Hours**: 2-4 p.m. on Wed., or by appointment
- **TA:** Mr. Ghazanfar Ali, Ghazanfar.Ali@ttu.edu
- **TA Office hours:** Tue. and Fri., 2-3 p.m., or by appointment
- **TA Office:** EC 201 A
- **More info:**
  - http://www.myweb.ttu.edu/yonchen
  - http://discl.cs.ttu.edu; http://cac.ttu.edu/; http://nsfcac.org

# Outline

- Questions?

- Mapping techniques
- Parallel algorithm model
- Matrix-multiplication parallel algorithms
- Gaussian Elimination algorithm of solving linear equations

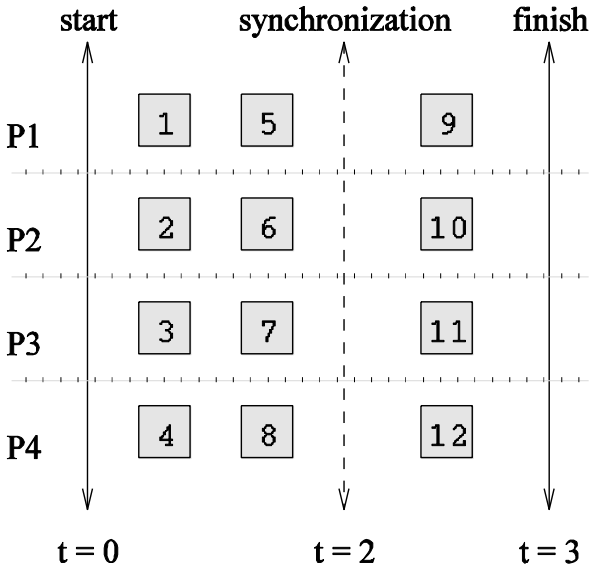
# Mapping Techniques

- Once a problem has been decomposed into concurrent tasks, these tasks must be mapped to processes (that can be executed on a parallel platform)

- Mappings must minimize overheads
  - Primary overheads are communication and idling
  - Minimizing these overheads often represents contradicting objectives
  - Assigning all work to one processor trivially minimizes communication at the expense of significant idling.

# Mapping Techniques (cont.)

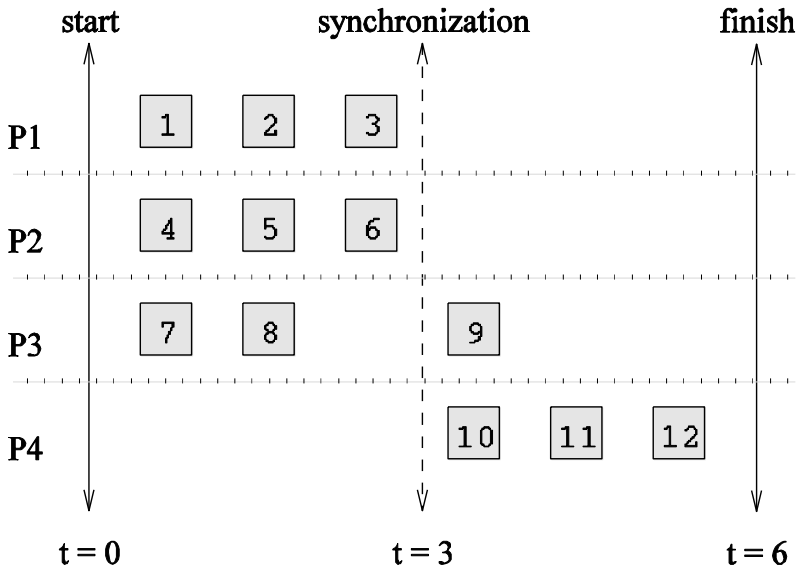
Mapping must simultaneously minimize idling and load balance.
Merely balancing load does not minimize idling.




(a)

# Mapping Techniques (cont.)

Mapping must simultaneously minimize idling and load balance.
Merely balancing load does not minimize idling.




(b)

# Mapping Techniques (cont.)

Mapping techniques can be static or dynamic.

- Static Mapping: Tasks are mapped to processes a-priori. For this to work, we must have a good estimate of the size of each task

- Dynamic Mapping: Tasks are mapped to processes at runtime. This may be because the tasks are generated at runtime, or that their sizes are not known.

Other factors that determine the choice of techniques include the size of data associated with a task and the nature of underlying domain.

# Schemes for Static Mapping

- Mappings based on a-priori decision, e.g.
  - Array distribution, 1-D block distribution schemes

| |
|---|
| $P_0$ |
| $P_1$ |
| $P_2$ |
| $P_3$ |
| $P_4$ |
| $P_5$ |
| $P_6$ |
| $P_7$ |

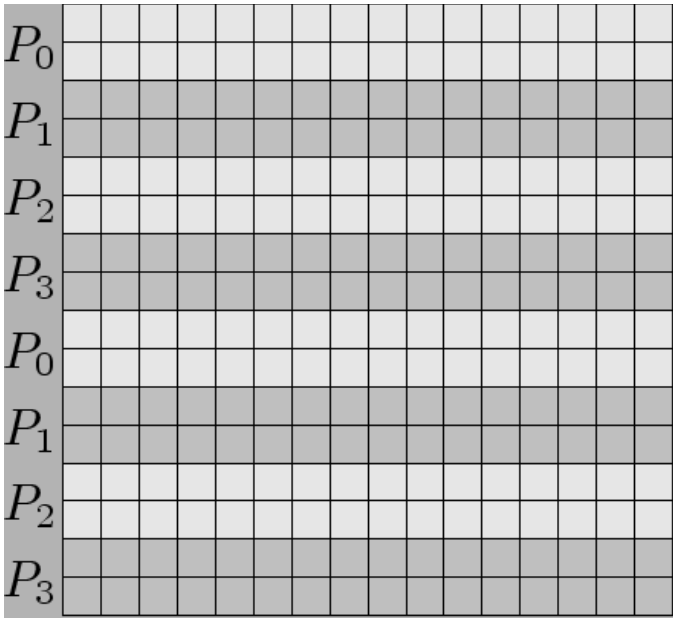
# Schemes for Static Mapping (cont.)

- Mappings based on a-priori decision, e.g.
  - Block distribution generalized to higher dimensions, e.g. 2-D block distribution

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

# Schemes for Static Mapping (cont.)

- Mappings based on a-priori decision, e.g.
  - Cyclic distribution




  - Block-cyclic distribution (hybrid)

# Schemes for Dynamic Mapping

- Dynamic mapping is sometimes also referred to as dynamic load balancing, since load balancing is the primary motivation for dynamic mapping.
  - Centralized dynamic mapping
  - Distributed dynamic mapping

# Schemes for Dynamic Mapping (cont.)

- Centralized Dynamic Mapping
  - Processes are designated as masters or slaves.
  - When a process runs out of work, it requests the master for more work.
  - Master may become the bottleneck
  - Chunk scheduling: pick up a number of tasks (a chunk) once: tradeoff
- Distributed Dynamic Mapping
  - Each process can send or receive work from other processes.
  - This alleviates the bottleneck in centralized schemes

# Minimizing Overheads

- Maximize data locality to avoid communication
  - Where possible, reuse intermediate data
  - Restructure computation so that data can be reused

- When communication is necessary, minimize the volume of data exchange
  - There is a cost associated with each word that is communicated. For this reason, we must minimize the volume of data communicated.

# Minimizing Overheads (cont.)

- Minimize the frequency of communications
  - There is a startup cost associated with each communication. Therefore, try to merge multiple communications into one, where possible
- Minimize the contention and hot-spots
  - Use decentralized/distributed techniques, replicate data where necessary

# Minimizing Overheads (cont.)

- Use group communications instead of point-to-point primitives
  - Identify the communication pattern and use the right API
  - Similarly to I/O operations
- Overlap computations with communications
  - E.g. use non-blocking communications, multithreading, and prefetching to hide latencies
- Overlap communications with other communications

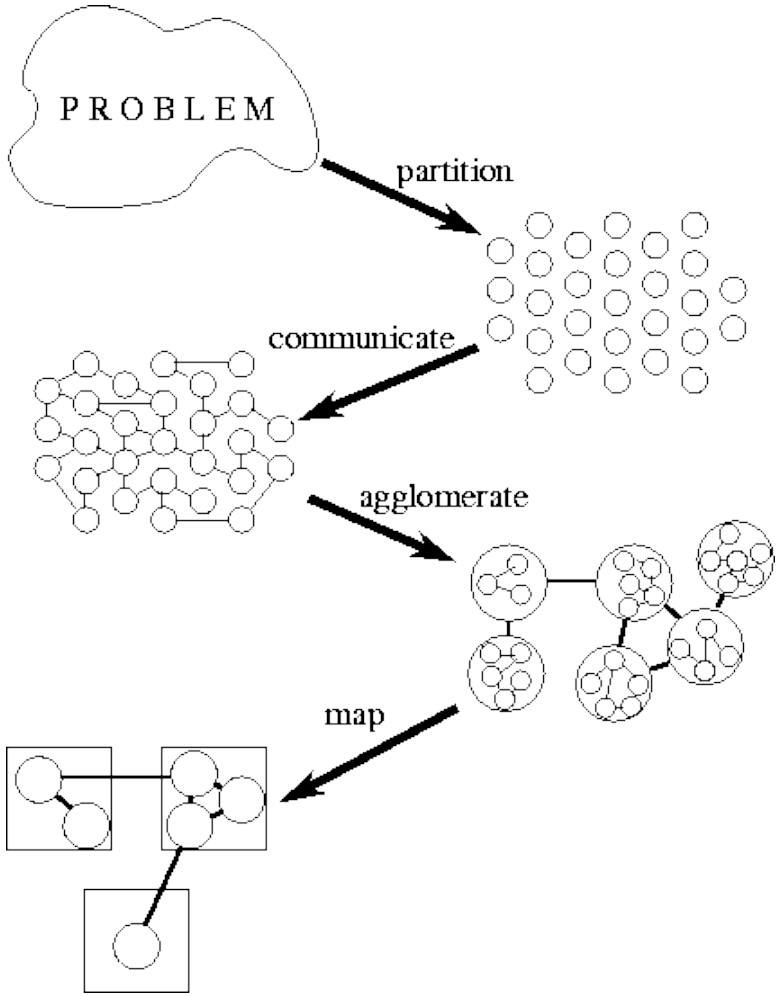
# Parallel Algorithm Model

A parallel algorithm model is a way of structuring a parallel algorithm by selecting a decomposition and mapping technique and applying the appropriate strategy to minimize overheads

# Summary: A PCAM Method

- Partitioning
- Communication
- Agglomeration
- Mapping

- http://www.mcs.anl.gov/~itf/dbpp



# Outline

- Questions?

- Mapping techniques
- Parallel algorithm model
- Matrix-multiplication parallel algorithms
- Gaussian Elimination algorithm of solving linear equations

# Matrix-Matrix Multiplication

- Consider the problem of multiplying two $n$ x $n$ dense, square matrices $A$ and $B$ to yield the product matrix $C = A$ x $B$.

- The serial complexity is $O(n^3)$.

```
1.  procedure MAT_MULT (A, B, C)
2.  begin
3.    for i := 0 to n - 1 do
4.      for j := 0 to n - 1 do
5.        begin
6.          C[i, j] := 0;
7.          for k := 0 to n - 1 do
8.            C[i, j] := C[i, j] + A[i, k] x B[k, j];
9.        endfor;
10. end MAT_MULT
```

# Parallelize Matrix-Matrix Multiplication

- A useful concept is called block operations
- In this view, an $n$ x $n$ matrix $A$ can be regarded as a $q$ x $q$ array of blocks $A_{i,j}$ ($0 \le i, j < q$) such that each block is an $(n/q)$ x $(n/q)$ submatrix.
- In this view, we perform $q^3$ matrix multiplications, each involving $(n/q)$ x $(n/q)$ matrices.

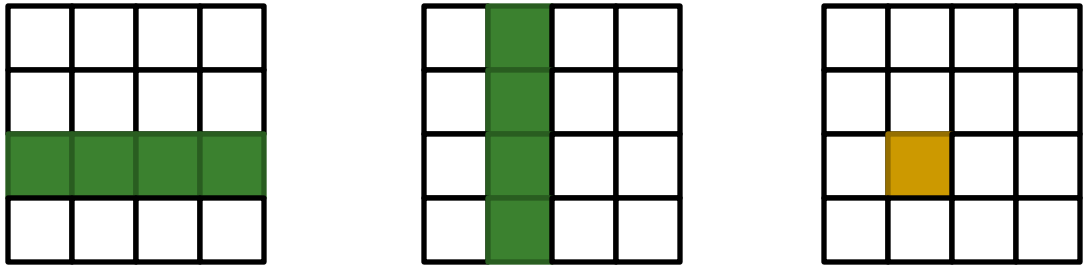
$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

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

# Parallelize Matrix-Matrix Multiplication (cont.)

- Assuming distributed-address-space
- Consider two $n$ x $n$ matrices $A$ and $B$ partitioned into $p$ blocks $A_{i,j}$ and $B_{i,j}$ ($0 \le i, j < \sqrt{p}$) of size $(n/\sqrt{p}) \times (n/\sqrt{p})$ each.
- Process $P_{i,j}$ initially stores $A_{i,j}$ and $B_{i,j}$ and computes block $C_{i,j}$ of the result matrix.
- Computing submatrix $C_{i,j}$ requires all submatrices $A_{i,k}$ and $B_{k,j}$ for $0 \le k < \sqrt{p}$.
- All-to-all broadcast blocks of $A$ along rows and $B$ along columns.
- Perform local submatrix multiplication.

Data decomposition
Static mapping (block distribution)



Calculating $C_{2,1}$

# Parallelize Matrix-Matrix Multiplication (cont.)

- Pseudo-algorithms
- Step 1: load $A_{i,j}$ and $B_{i,j}$ to $P_{i,j}$
- Step 2: all-to-all broadcast A along rows
- Step 3: all-to-all broadcast B along columns
- Step 4: each process has corresponding row and column to perform local submatrix multiplications and additions

- Total amount of computations (multiplications and additions) same as in the serial case, optimal
- Communication and memory, not optimal

# Cannon's Algorithm for Matrix-Matrix Multiplication

- In this algorithm, we reduce the communication and memory requirement
  - Each process only keeps one submatrix of A and B

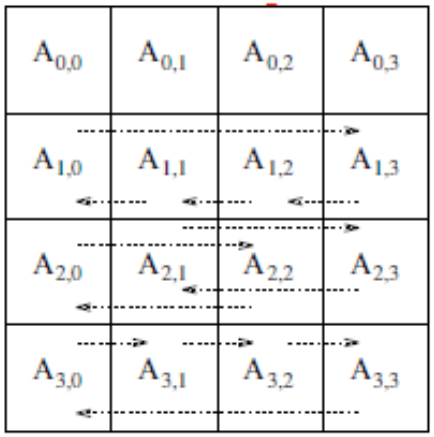
- We schedule the computations of the $\sqrt{p}$ processes of the *i*-th row such that, at any given time, each process is using a different block $A_{i,k}$

- These blocks can be systematically rotated among the processes after every submatrix multiplication so that every process gets a fresh $A_{i,k}$ after each rotation

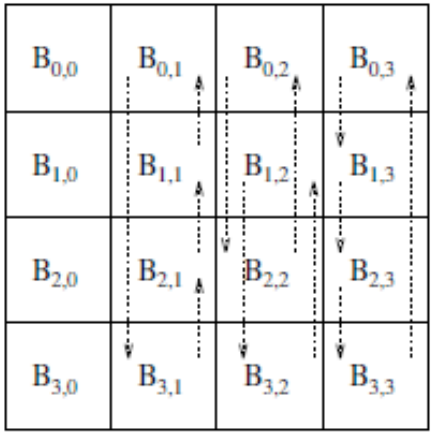
Data decomposition
Static mapping (block distribution)
Reduce communication and memory

# Cannon's Algorithm for Matrix-Matrix Multiplication

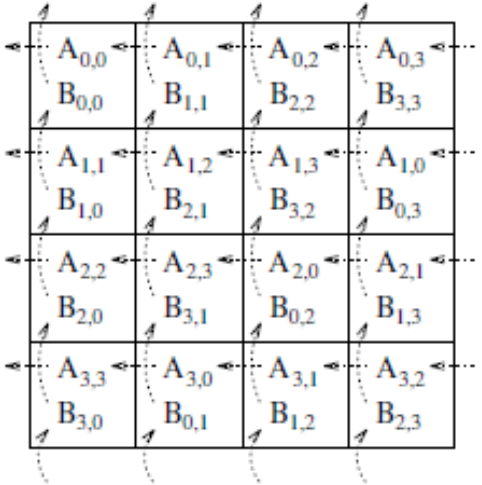
- Align the blocks of $A$ and $B$ in such a way that each process multiplies its local submatrices. This is done by shifting all submatrices $A_{i,j}$ to the left (with wraparound) by $i$ steps and all submatrices $B_{i,j}$ up (with wraparound) by $j$ steps
- Perform local block multiplication




(a) Initial alignment of A



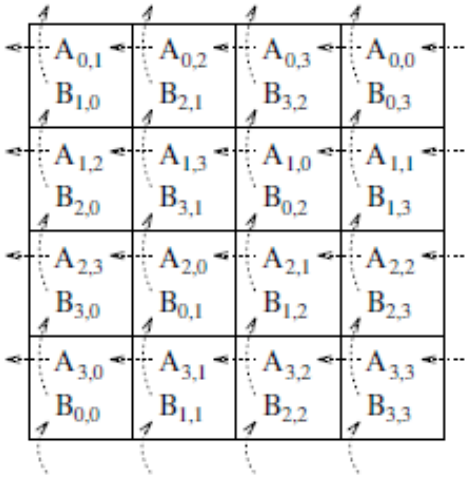

(b) Initial alignment of B



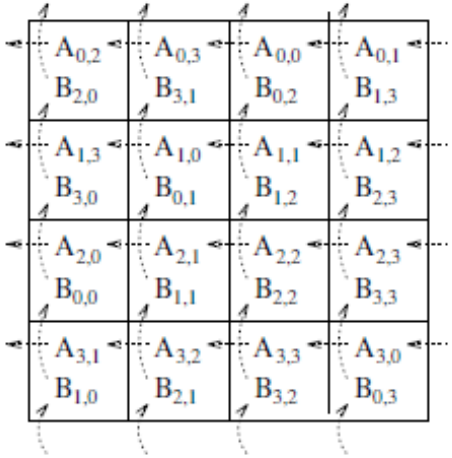

(c) A and B after initial alignment

# Cannon's Algorithm for Matrix-Matrix Multiplication

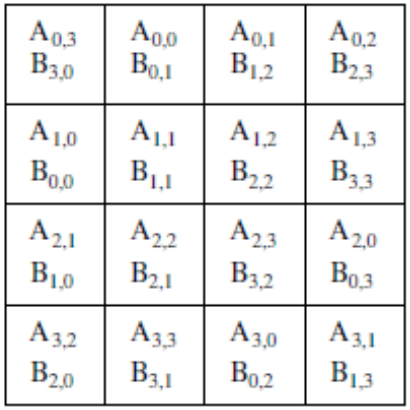
- Each block of $A$ moves one step left and each block of $B$ moves one step up (again with wraparound)
- Perform next block multiplication, add to partial result, repeat until all $\sqrt{p}$ blocks have been multiplied




(d) Submatrix locations after first shift




(e) Submatrix locations after second shift




(f) Submatrix locations after third shift

# Cannon's Algorithm for Matrix-Matrix Multiplication

- Pseudo-algorithms
- Step 1: load $A_{i,j}$ and $B_{i,j}$ to $P_{i,j}$
- Step 2: create Cartesian topology and find coordinates
- Step 3: shifting all submatrices $A_{i,j}$ to the left (with wraparound) by $i$ steps and all submatrices $B_{i,j}$ up (with wraparound) by $j$ steps
- Step 4: perform local block multiplication
- Step 5: shift each block of $A$ one step left and each block of $B$ one step up (again with wraparound)
- Step 6: perform next block multiplication, add to partial result
- Step 7: repeat until all $\sqrt{p}$ blocks have been multiplied.

# Supplemental Readings

- Reference book ITPC, Chapter 3, Chapter 8, 8.2
- Foster, DBPP, Chapter 2
  http://www.mcs.anl.gov/~itf/dbpp

# Questions?

## Questions/Suggestions/Comments are always welcome!

Write me: yong.chen@ttu.edu
Call me: 806-834-0284
See me: ENGCTR 315

*If you write me an email for this class, please start the email subject with [CS4379] or [CS5379].*