PC to HPC:
Parallel Computing

Xiaoge Wang

ICER

Jan 24, 2017

MICHIGAN STATE UNIVERSITY

ICER

Outline

- Moving from PC to HPC
- Principles of HPC
- Parallel computing examples

Outline

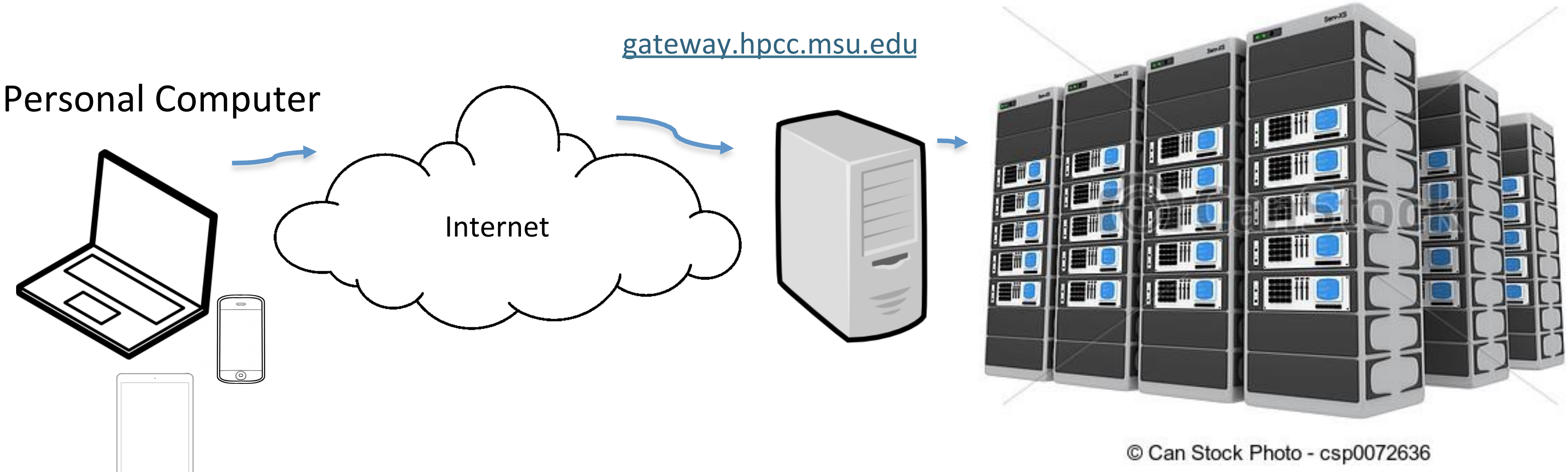
- **MOVING FROM PC TO HPC**
- Principles of HPC
- Parallel computing examples

Moving from PC to HPC

- Differences between PC and HPC
- When to move from PC to HPC?
- How to move from PC to HPC?

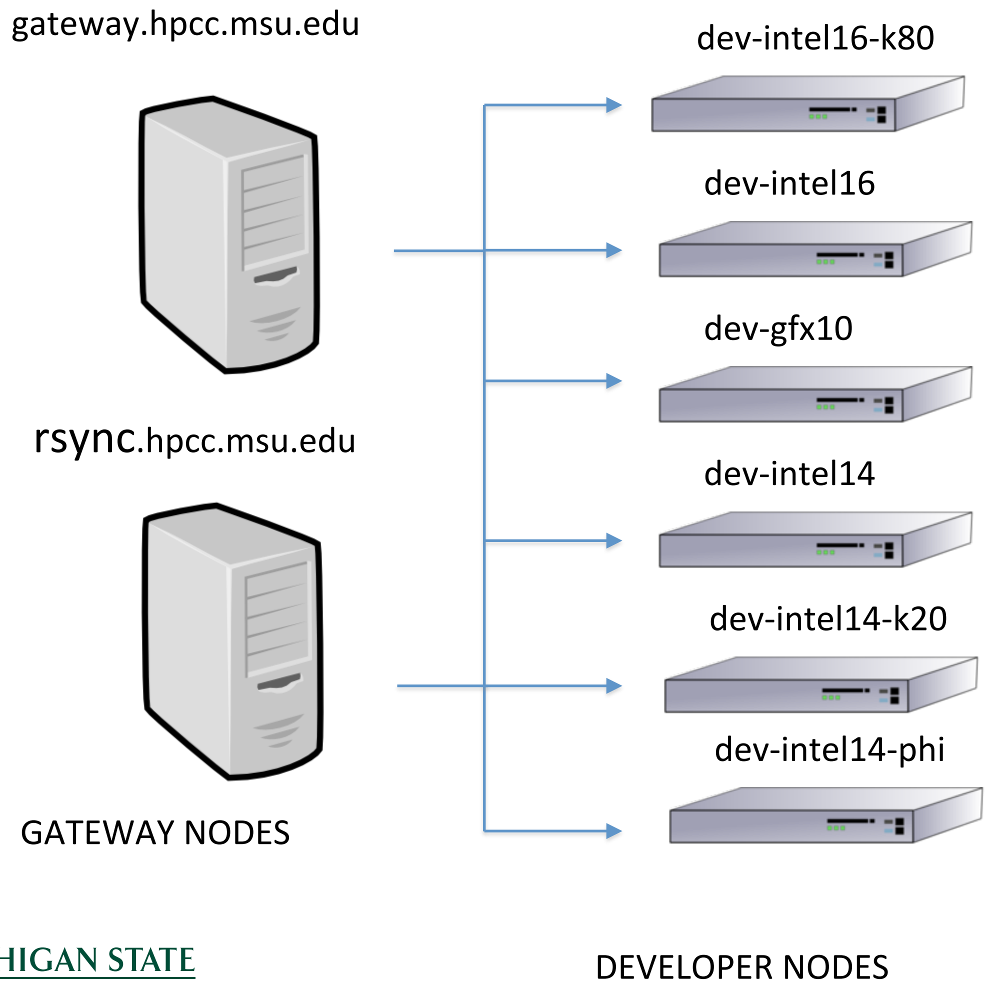
Moving From PC to HPC

- C = computing/computer



http://wiki.hpcc.msu.edu/x/DYAf

Example of HPC system



PC vs. HPC

- Capability
  - Capability is mostly from multiplicity
- User interface
  - More command line, less GUI
- Internal structure
  - Data movement become bottleneck
  - Nodes, cores, accelerators
  - SIMD, processes, threads
- Resource Sharing

When to move to HPC?

- Your PC fails to satisfy the needs
  - Speed
  - Storage
  - Style
- The resources are not directly available
  - Services
  - Software
  - Data

How to move to HPC?

- Switch to other Apps
- Run the same App on HPC
- Adapt your own program to HPC
- Develop new App for your field on HPC

Which is your way?

Outline

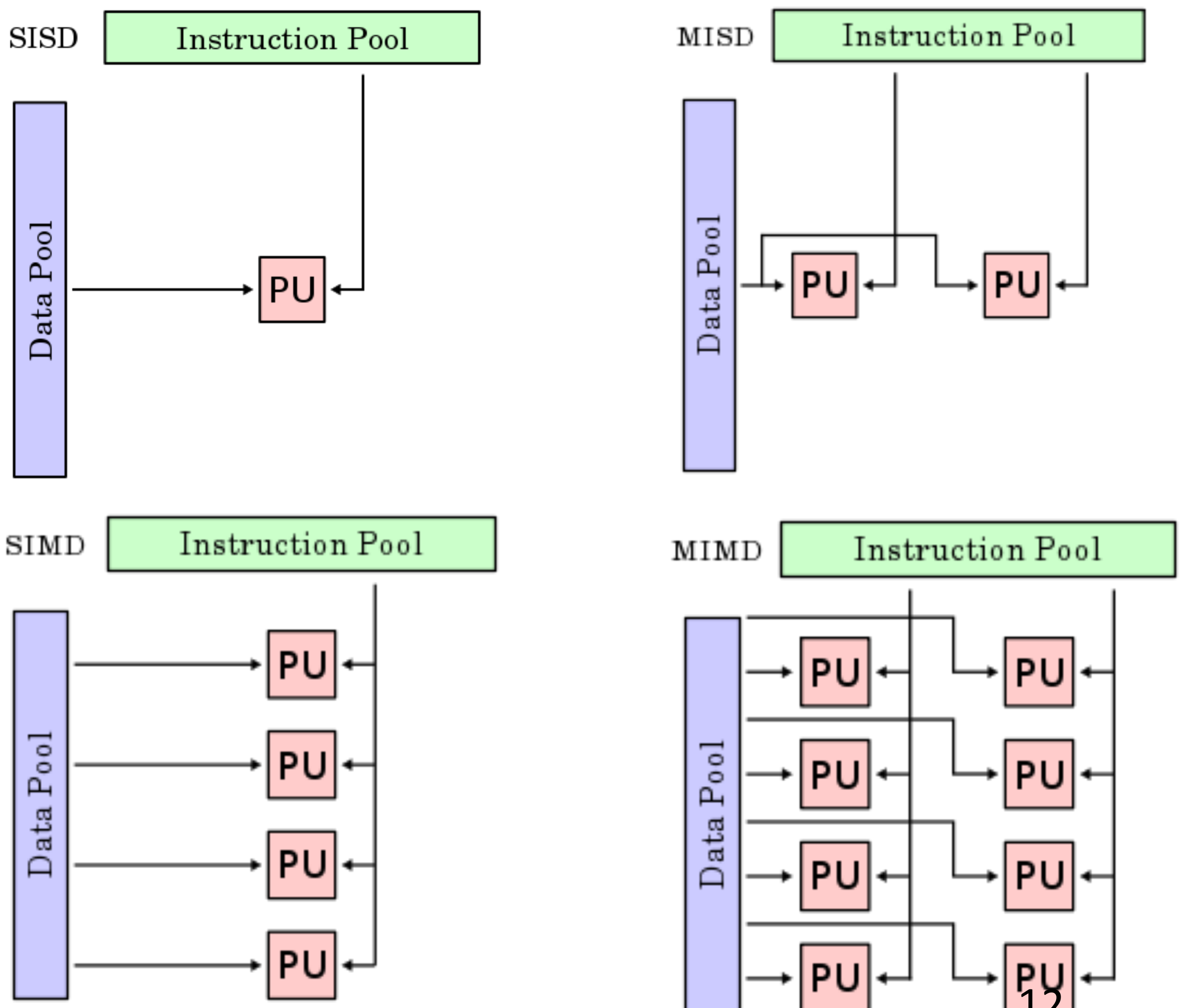
- Moving from PC to HPC
- **PRINCIPLES OF HPC**
- Parallel computing examples

Principles of HPC

- Classification of machines
- Parallel programming models
- Methodical strategy of design
- Fundamentals of parallel programming

Classification

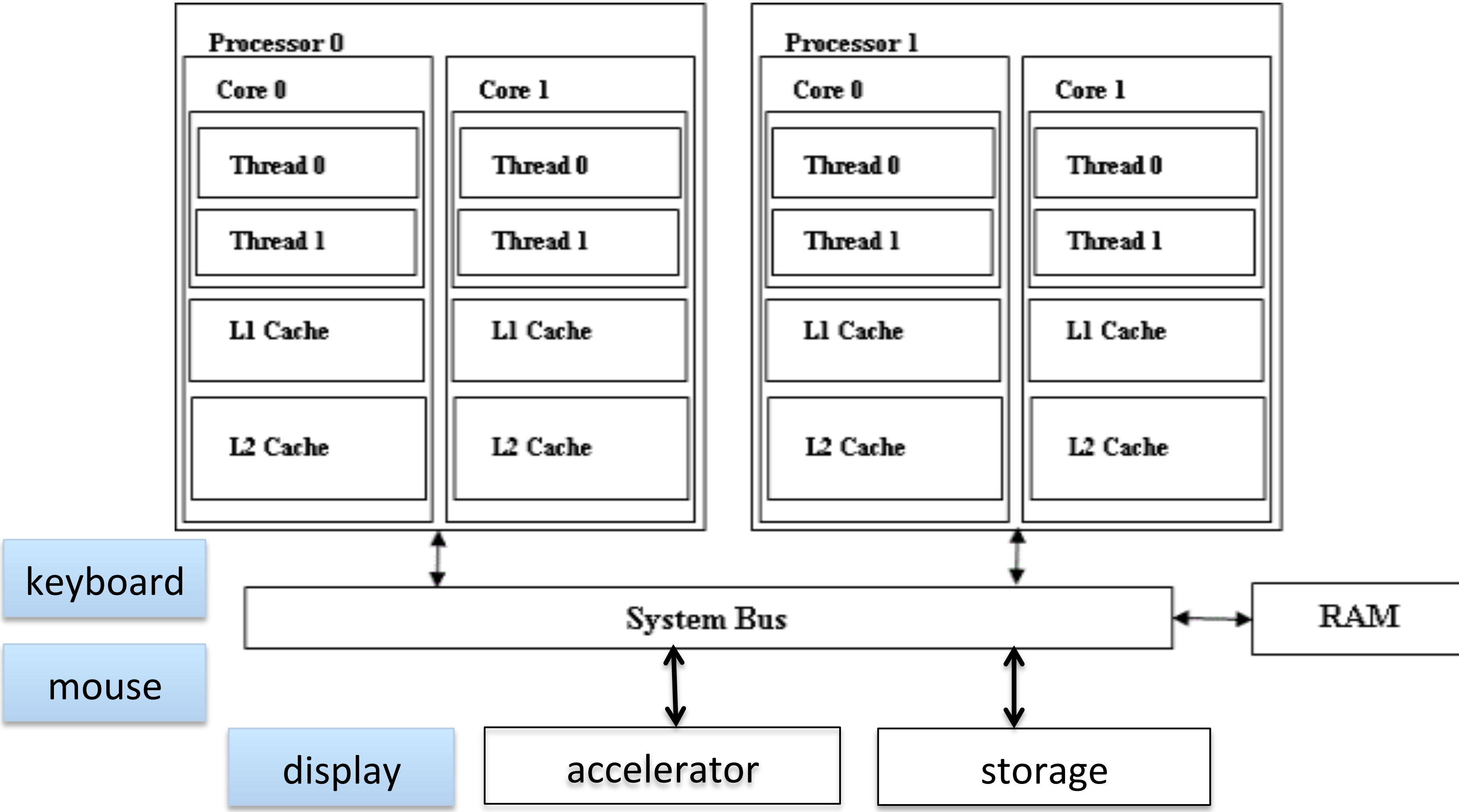
Flynn's Taxonomy



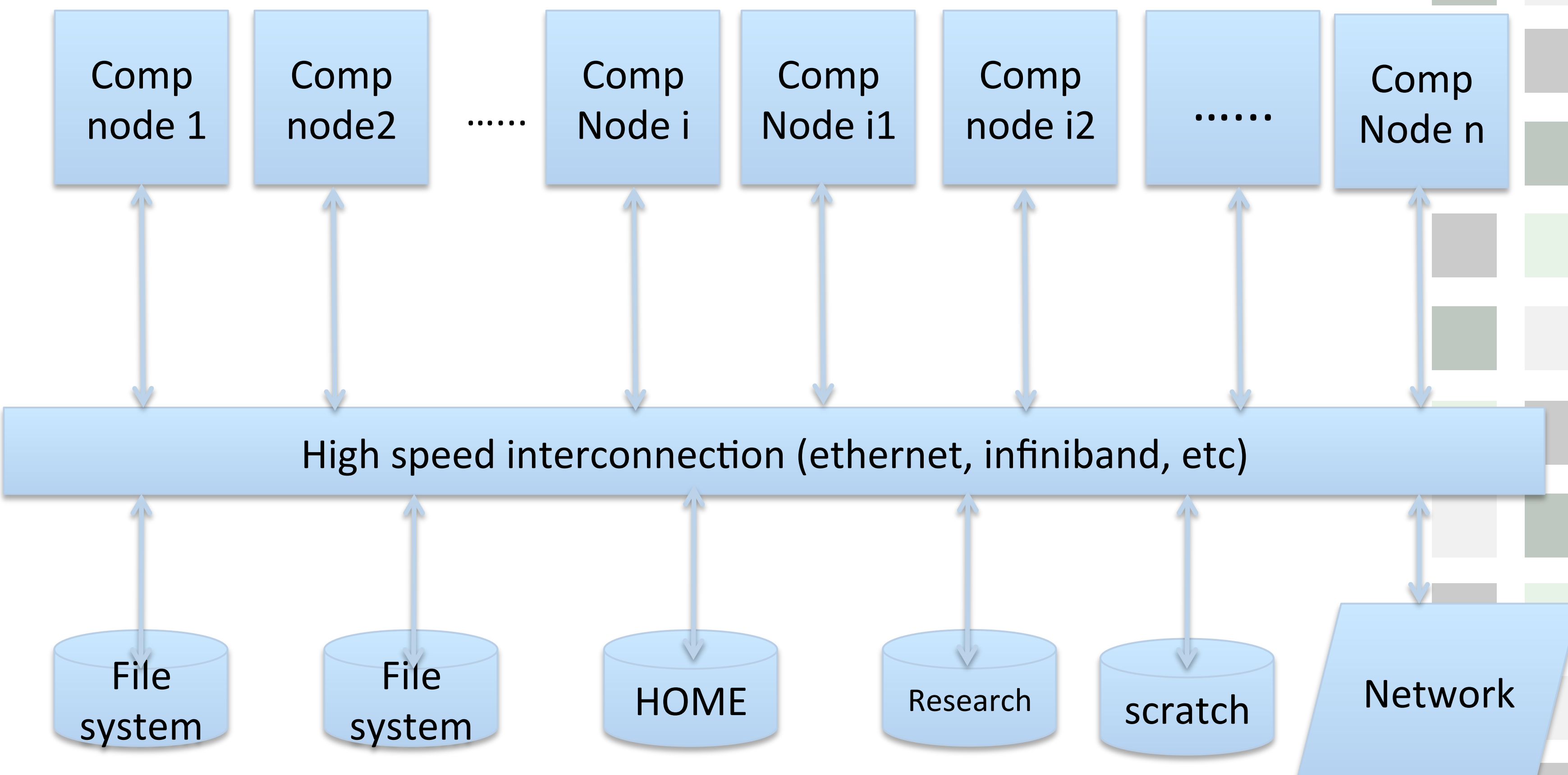
MICHIGAN STATE UNIVERSITY

ICER

Architecture of PC



Architecture of HPC

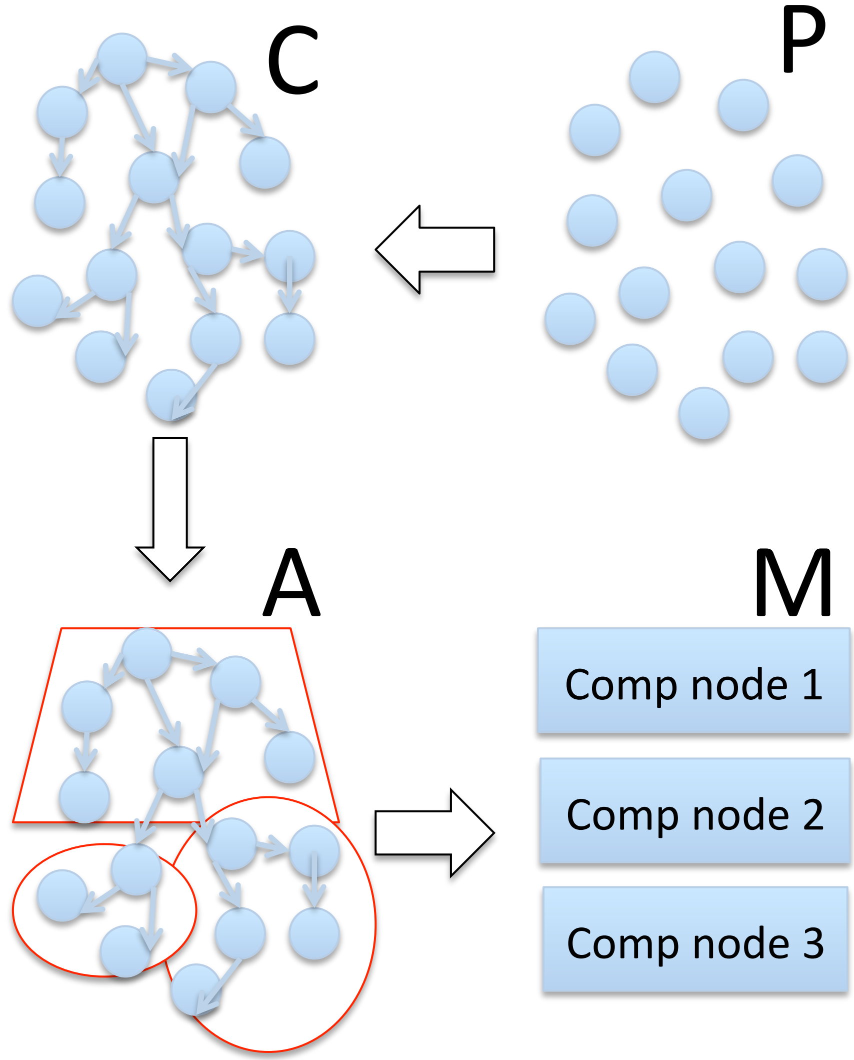


Parallel Programming Models

- Shared memory
- Message passing
- Map-reduce
- Data-driven workflow

Methodical Strategy of Design

- Partition
- Communication
- Agglomeration
- Mapping



Fundamentals

- Partitioning:
  - Data partition
  - Task partition
- Communication
  - Data sharing
  - Message passing
- Coordination between parallel tasks.
  - Dependency analysis
- Performance evaluation
- Bug or feature?
  - Non-deterministic
  - Race condition

Partition

- Task partition
  – Program
  – Module
  – Function
  – loop
- Data partition
  – 1D, 2D, 3D array
  – Domain decomposition
  – Data set partition

Granularity

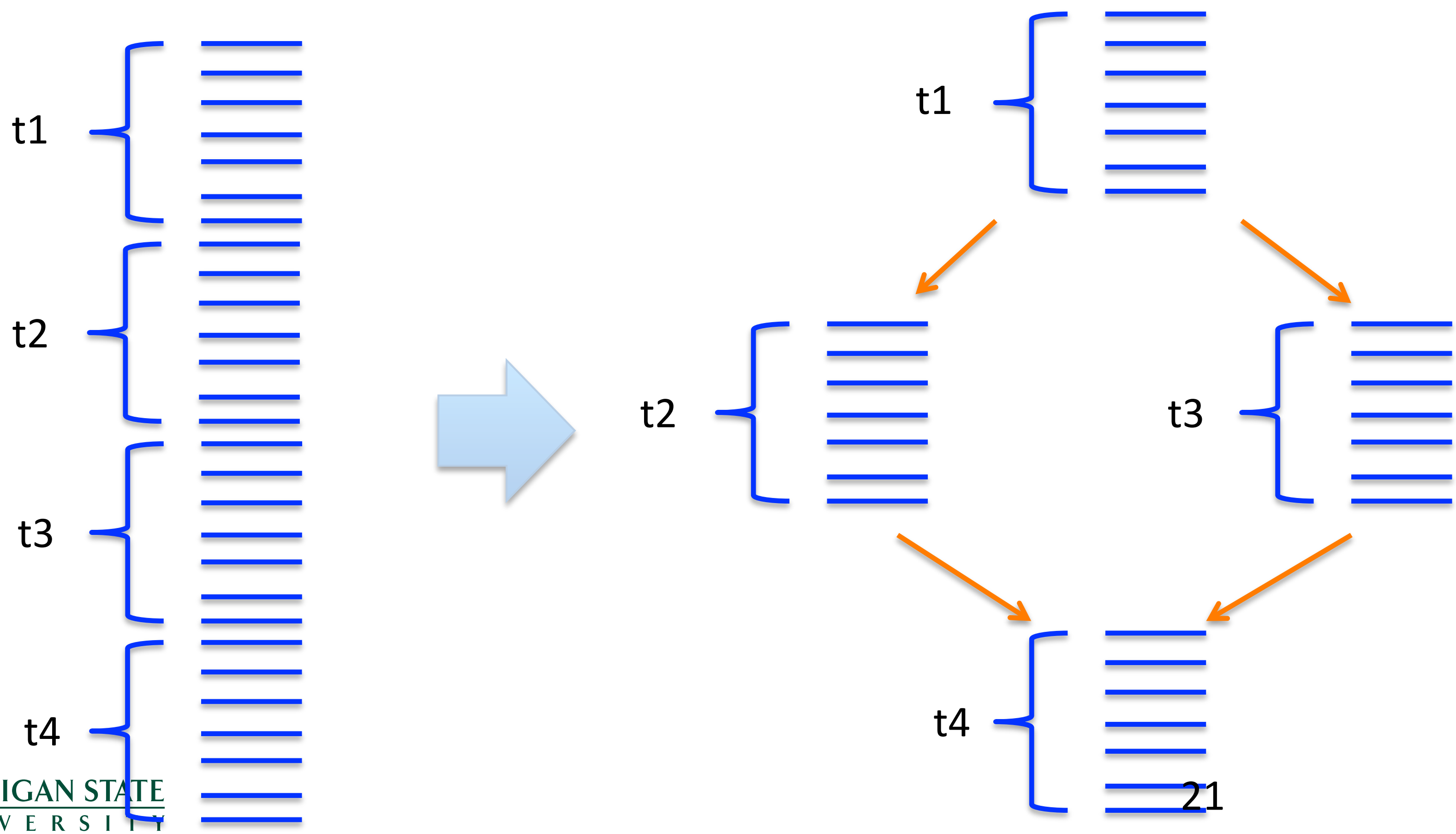
- Instruction
- Thread
- Process
- Program
- Application

Communication

- Data sharing (ex. OpenMP)
  - Traffic signal, billboard, signs, etc.
  - Access control: critical region
  - Shared space vs. private space
- Messages passing (ex. MPI)
  - Blocking/unblocking message passing
  - Point-to-point : send, receive
  - Collective
  - Overhead of massage passing

Coordination

- Determine the order of execution
- Enforce the order of execution



Dependency Analysis

Given 2 tasks, t1 and t2. t2 is dependent on t1 if

- Control dependency: t2's execution is guarded by the execution result of t1
- Data dependency: the data used in t2 is the results of t1, or vice versa. Or both t1 and t2 will write to the same output.
- Loop dependency: Take loop index into analysis

Performance evaluation

- Speedup
- Efficiency
- Amdahl's Law

Speedup and Efficiency:

Let

T_1: the execution time on one processor,

T_p: the execution time on p processor,

P: number of processors used,

$E_{relative}$: relative efficiency,

$E_{relative} = T_1/(pT_p)$

$S_{relative}$: relative speedup,

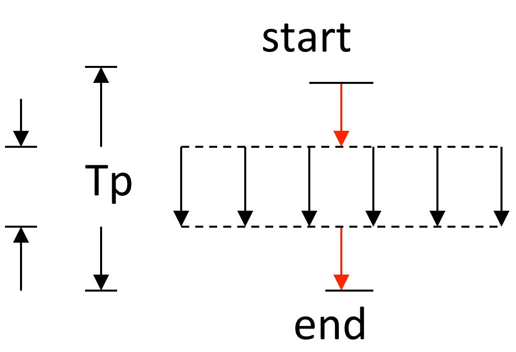
$S_{relative} = pE_{relative} = T_1/T_p$

Amdahl's Law:

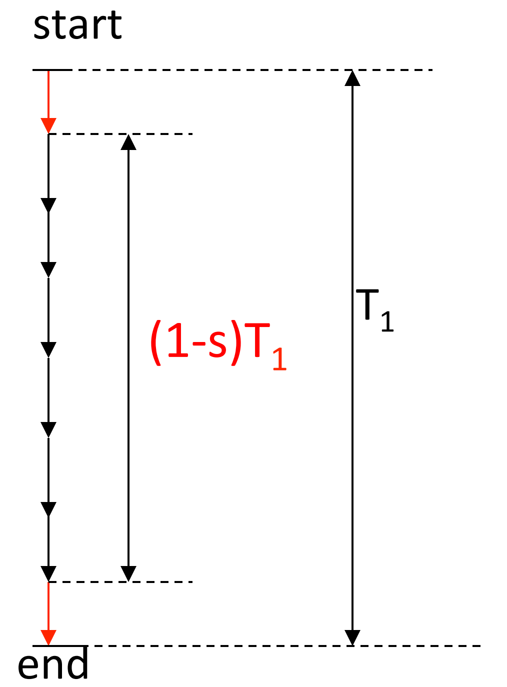
Assumption: for a given program,

serial fraction = s, $0 \le s \le 1$,

p-fold parallel fraction = 1-s.



Parallel execution



Amdahl's Law:

Then

$$T_p = sT_1 + (1-s)T_1 / p;$$

$$S_p = 1/(s + (1-s)/p);$$

$$E_p = 1/(sp + (1-s)).$$

When $p \to \infty$, $S_p \to 1/s$, $E_p \to 0$!

Where is the hope for parallel computing?

Is the Amdahl's Law Correct?

- Too optimistic?

  Assume that 1-s of total computation could be perfectly parallelizable.

- Too pessimistic?

  Assume that number of processors is unlimited, we have upper bound for the speedup.

  $s = 0.1 \quad S(p) < 10$

  $s = 0.01 \quad S(p) < 100$

  $s = 0.001 \quad S(p) < 1000$

What is the problem with Amdahl's Law?

Answer: *s* could be the function of problem size n !

Example: *outer product of vector $v*v^T$ with one I/O port and p processors.*

Time for data input (distribution): O(n)

Time for computation: $O(n^2)$

Serial fraction s is O(1/n).

s decreases when problem size increase!

Bugs or Features?

- Non-deterministic
  - Reproducibility?
  - Different Nodes -> different speed -> different operation order
  - Association law may not true (ex. +, *, …)
- Race condition
  - Lock
  - Atomic operation

Outline

- Moving from PC to HPC
- Principles of HPC
- **PARALLEL COMPUTING EXAMPLES**

Parallel Computing Examples

1. Parallel computing with shell script
2. Parallel computing with qsub script
3. Parallel computing with data partition

Example 1: Baking Cakes

Task: Need to make a cakes for a party
(input: ingredients, output: cakes)

Sequential program (single person's task):

```
make_a_cake {
    • Measure dry ingredients
    • Measure liquid ingredients
    • Grease a baking pan
    • Mix ingredients
    • baking
    }
end
```

Example 1: Baking Cakes

Task: Need to make a 5 cakes for a party

(input: ingredients, output: cakes)

(1) Sequential program (single person):

```
for i = 1: 5
    making_a_cake { }
end
```

(2) Parallel program (5 people team)

```
par for i = 1: 5
    making_a_cake { }
end
```

Example 1: Baking Cakes

Task: Need to make a 5 cakes for a party
(input: ingredients, output: cakes) We could partition task further.

(3) Parallel program (25 people team)

```
par for I = 1: 5  (in 5 groups)
    make_a_cake_p { }
End
```

(4) make_a_cake_p: 5 people work together to make a cake

- Measure dry ingredients
- Measure liquid ingredients
- Grease pans
- Mix ingredients
- baking

What We Learn

- Interactive mode
- Multi-process parallel computing
- Partition
- Communication
- Dependency
- Non-deterministic
- Timing
- Speedup
- Efficiency

Example 2: Using Job Script

Task: Make 500 cakes.

- What is the difference between 5 and 500?
- How to run on compute nodes?
- How to run many jobs in parallel?
- How to enforce the dependency?

Example 2: Using Job Script

Task: Make 500 cakes

- How to run a jobs on compute nodes?
  - Wrap .sh up into .qsub file: resource?
- How to run 500 jobs on compute nodes?
  - Use "-t" option (array job): if all tasks are independent
- How to run 2500 (5x500) jobs in parallel?
  - Use "-w" option for dependency control

What We Learn

- Job level parallel computing
- How do subscribe resources
- Run many jobs in parallel
- Run jobs with dependency

Example 3: Data Partition

Task: process a large data set in parallel.

What we learn:

- How to partition a input file into many?
- How to run these task in parallel?
- How to get optimum result among all results of 100 tasks?

Summary

- Moving from PC to HPC
  - Difference between PC and HPC
  - Programming models
- Principles of HPC
  - Partition
  - Communication
  - Coordination
  - Performance evaluation
  - Characteristics of parallel program
- Parallel computing examples
  - Task and data partition
  - Tools: shell script, job script

Questions?

Thanks!

MICHIGAN STATE UNIVERSITY

ICER