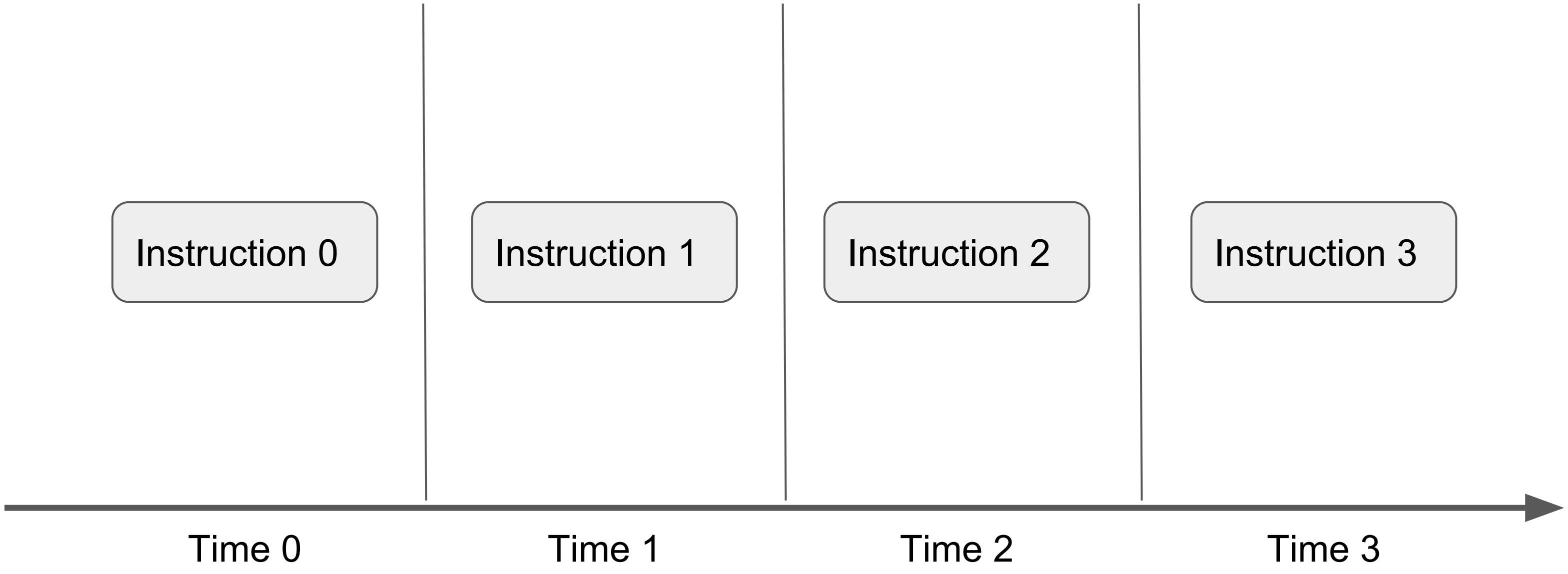
Parallel Computing and OpenMP: Terminology and Examples

Aaron Weeden
Shodor Education Foundation, Inc.
2015

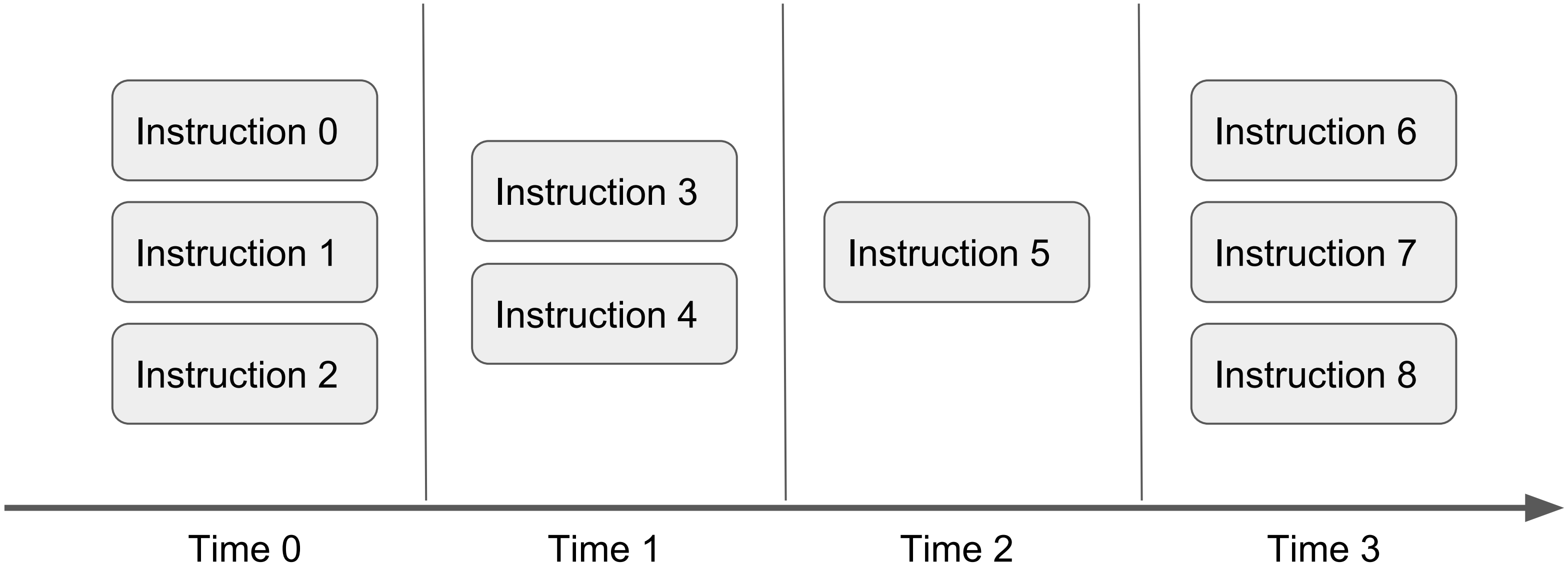
Key Term: **Serial**

Instructions are executed one at a time, in a series.



Key Term: **Parallel**

Multiple instructions are executed at the same time.



Key Term: **Core**

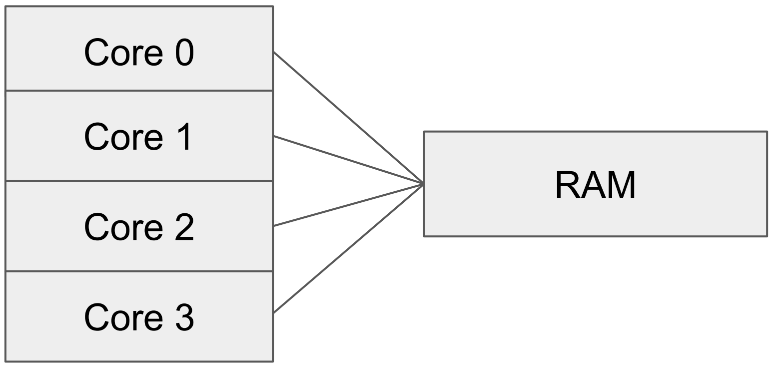
Entity on a CPU that executes instructions. Multiple cores can execute instructions in parallel.

Examples:

- dual-core CPU
- quad-core CPU
- octa-core CPU
- ...

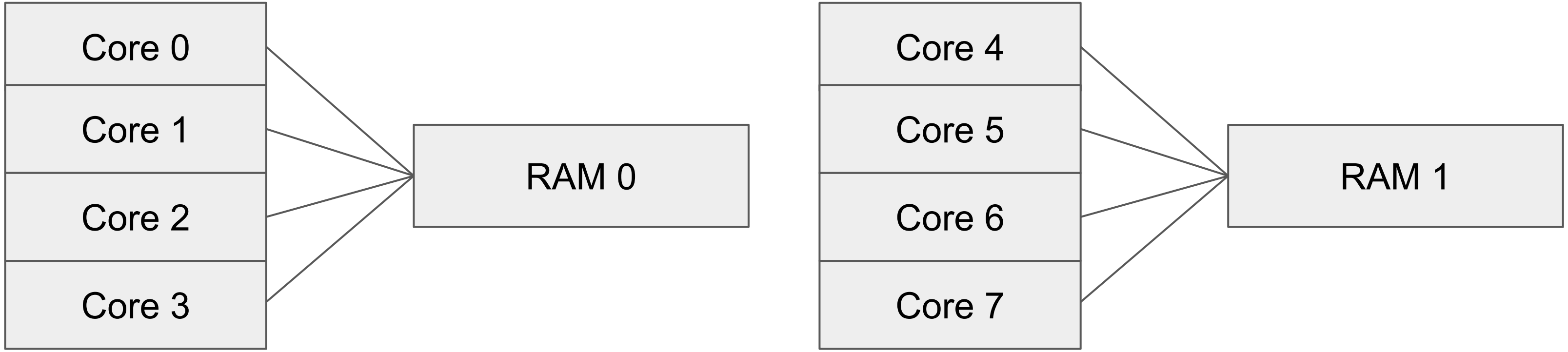
Key Term: **Shared Memory**

Multiple cores can share RAM, reading from and writing to memory in parallel.



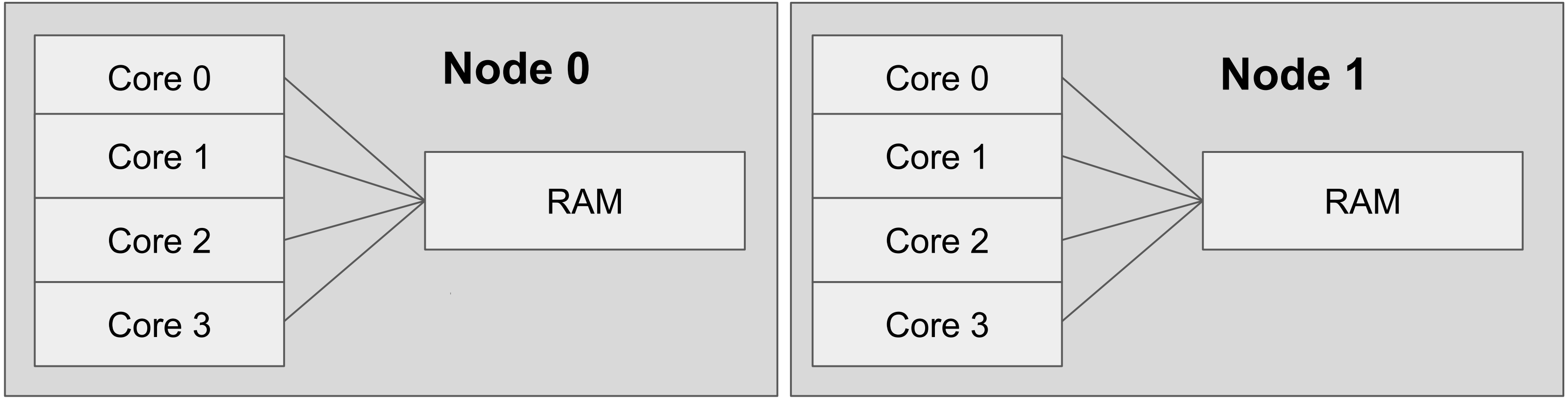
Key Term: **Distributed Memory**

Cores can also have RAM separate from other cores, unable to read from and write to each other's memory directly.



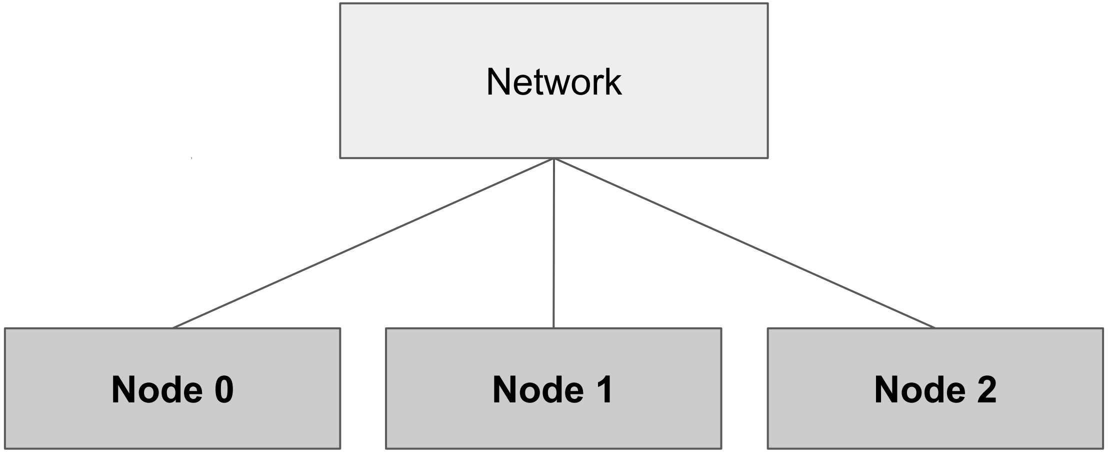
Key Term: **Node**

A grouping of cores and their shared memory.

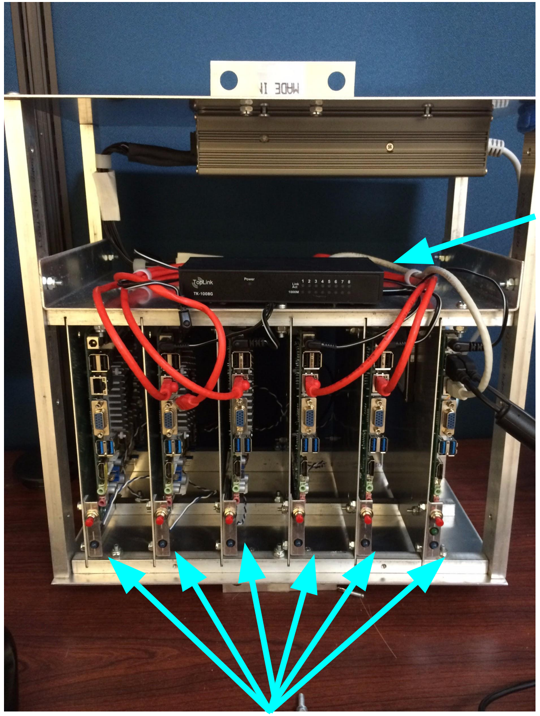


Key Term: **Cluster**

A grouping of nodes and the network that connects them.

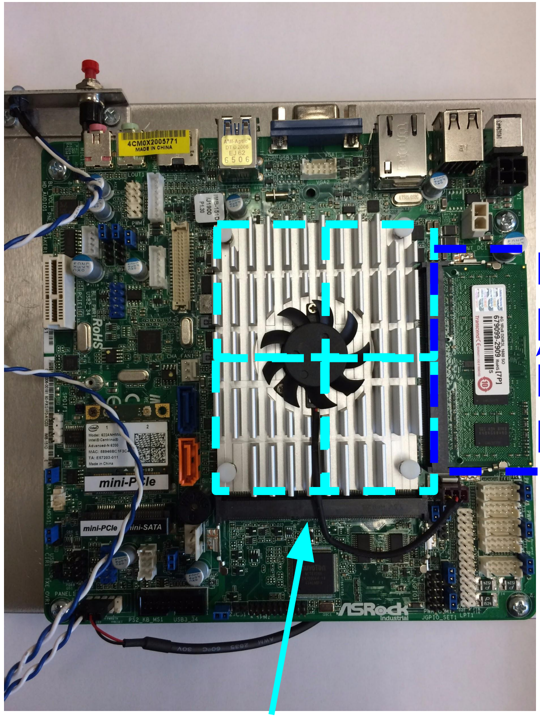


Cluster Example: **LittleFe** (http://littlefe.net)



Ethernet network

6 nodes per cluster



4 GB shared memory (RAM)

4 cores per node (quad-core CPU)

Key Term: **Supercomputer**

A really big, really fast cluster.

Normal laptop: 10^9 **FLOP/S** (Floating Point Operations per Second)

Terascale: 10^12 FLOP/S

Petascale: 10^15 FLOP/S (http://www.shodor.org/petascale)

Exascale: 10^18 FLOP/S

Supercomputer Example: **Blue Waters**

Image Source

Image Source

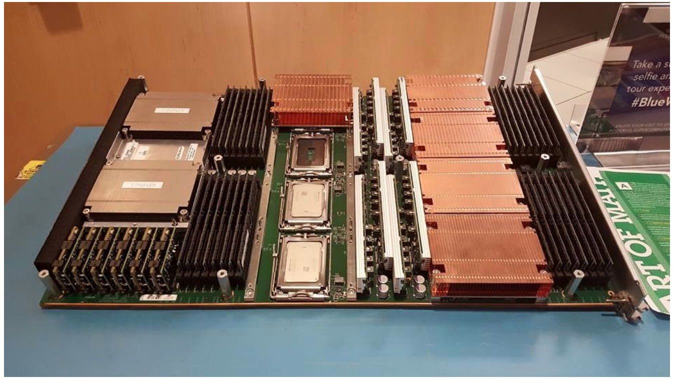


Photo credit: Erik Saathof

- Fastest supercomputer on a university campus (13 Petaflops)
- Over 23,000 nodes, almost 800,000 cores
- https://bluewaters.ncsa.illinois.edu/

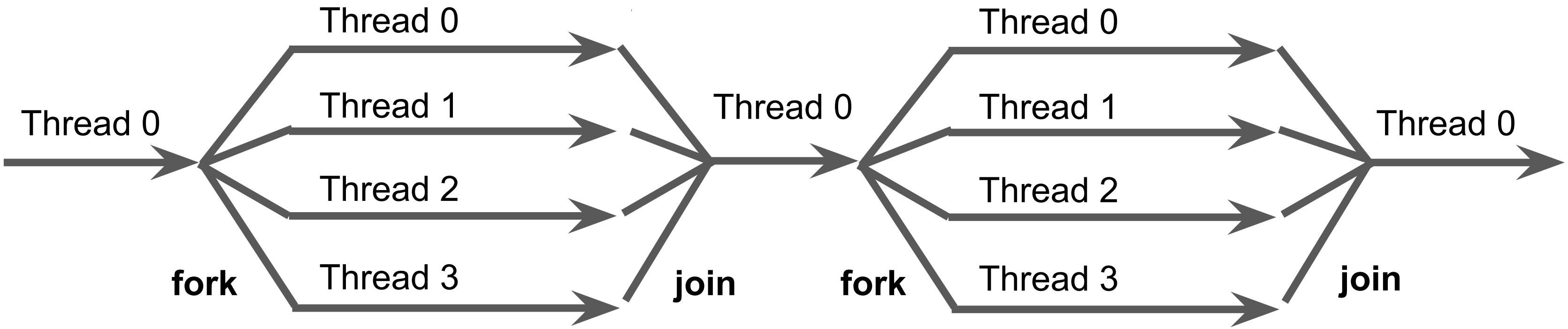
OpenMP

- API for shared memory programming in C, C++, and Fortran.
- Uses compiler directives to parallelize code.
- Syntax example: run iterations of a `for` loop in parallel:

```
#pragma omp parallel for
for (i = 0; i < n; i++) {
  ...
```

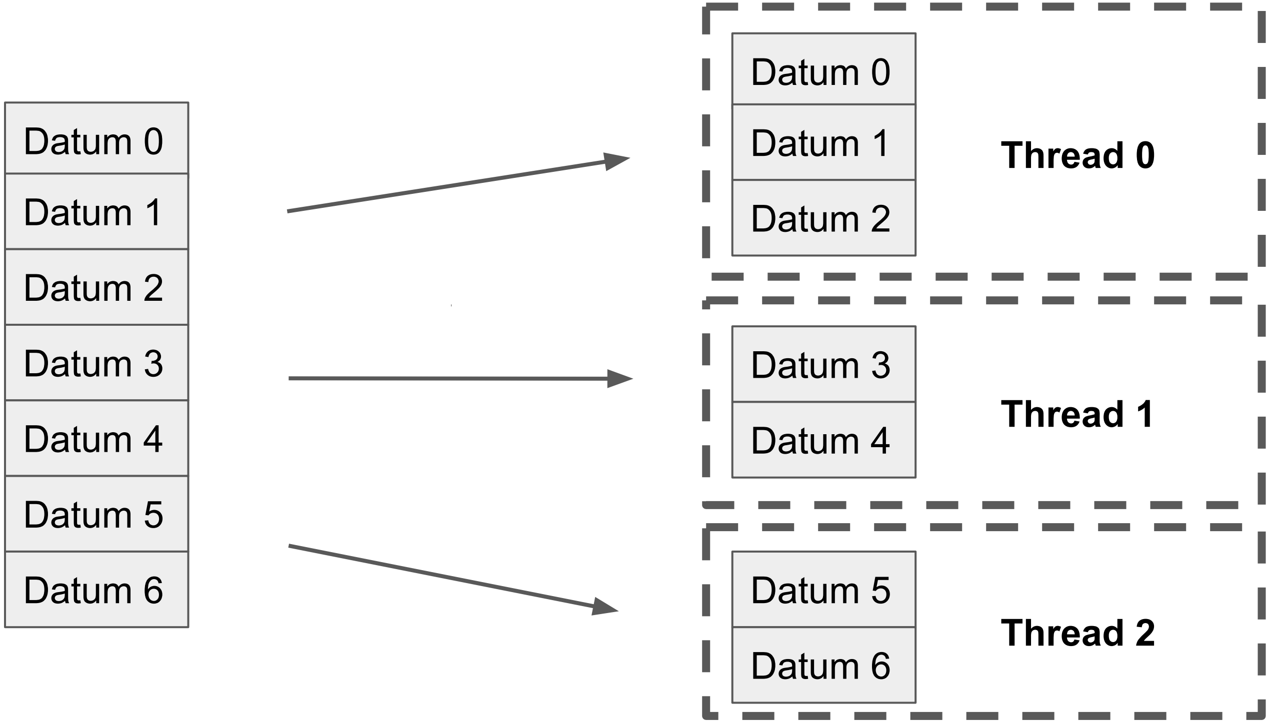
Key Term: **Thread**

- OpenMP entity that can use a core to execute instructions.
- Shares memory with other threads.
- Forked from a single, master thread at different points during program execution, then joins back into the master thread.



Key Term: **Domain Decomposition**

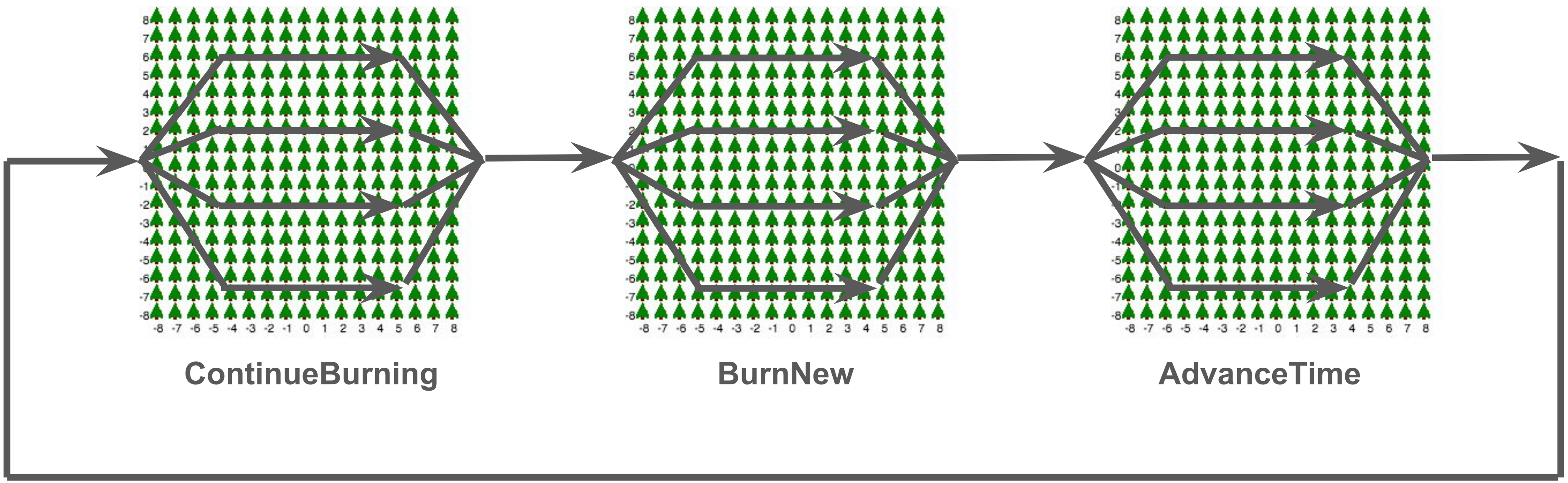
- Everyone does the same task, but on different data.



OpenMP Algorithm for Forest Fire Model

- Based on http://shodor.org/interactivate/activities/Fire/
- Data
  - **Trees** (array for checking trees)
  - **NewTrees** (array for changing trees)
- Tasks:
  - **InitData**: Light the center tree on fire
  - For each time step:
    - **ContinueBurning**: For trees already burning that haven't burnt out, burn another step.
    - **BurnNew**: For trees next to a burning neighbor, catch on fire with some probability.
    - **AdvanceTime**: Copy NewTrees into Trees.
- OpenMP threads are forked **before** each task and join back together **after** each task.

OpenMP Algorithm for Forest Fire Model

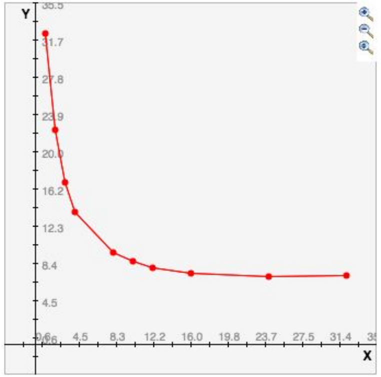


Forest Fire Model Parameters

- Input
  - Number of OpenMP threads
  - Number of rows in forest
  - Number of columns in forest
  - Probability of catching fire if next to a burning tree
  - Max # of steps a tree burns before burning out
  - Number of time steps
  - Seed for random number generator
  - Name of output file (for ASCII visualization)
- Output
  - What % of the forest burned?
  - How long did it take to run?

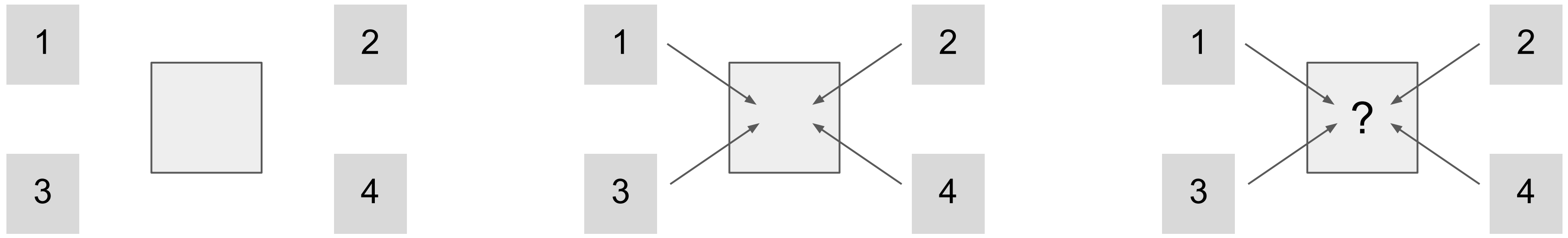
Key Term: **Strong Scaling**

- By keeping the problem size constant but increasing the number of cores, what happens to the run time?
- Example for forest fire with problem size = 1300 rows, 1300 columns, and 1300 time steps (x-axis: # of cores, y-axis: seconds of wall clock time, averaged for 5 runs):
- This is a common shape for a strong scaling curve



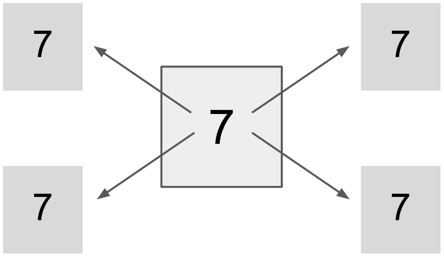
Key Term: **Race Condition**

- Multiple things happen at the same time; the result is unpredictable.
- e.g.: Collaborative spreadsheet -- what happens if multiple people try to edit the same cell? What will be the value in that cell? Whoever gets there last..
- Beware race conditions in parallel operations.

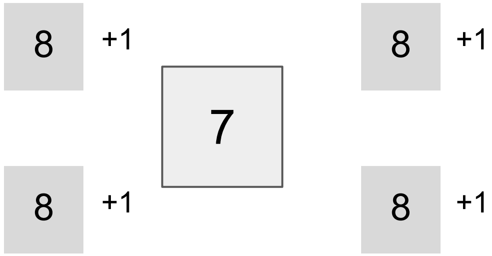


Race Condition Example: **Forest Fire**

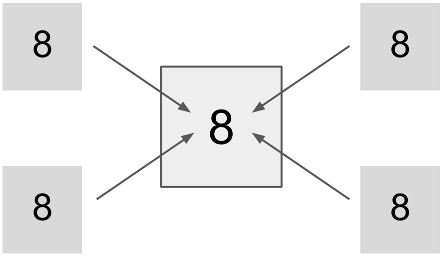
- As threads burn new trees, they update the count of burning trees.
- If multiple threads try to update the count at the same time, they might miss some trees.
- Here is an example with 4 threads. If 7 trees have burned already, and each thread wants to add 1 to the count, the end result should be 11, but:



In parallel, all threads see there are 7 trees burned so far



Each thread increments what it thinks are the number of burned trees

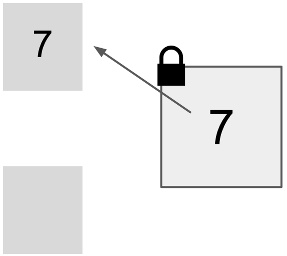


Each thread writes back the result

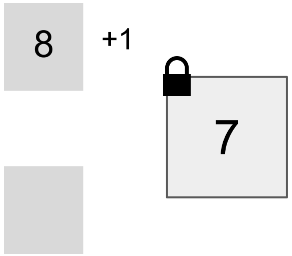
Key Term: **Lock**

A thread can prevent other threads from reading from or writing to a variable until it is finished reading/writing.

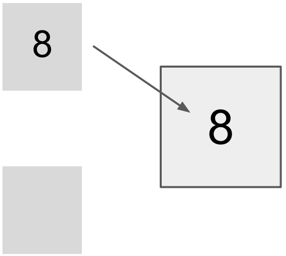
Lock Example



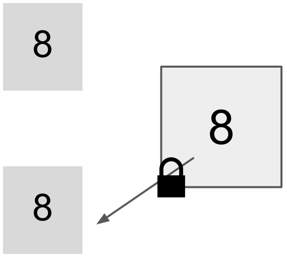
1. Thread 0 locks and reads.



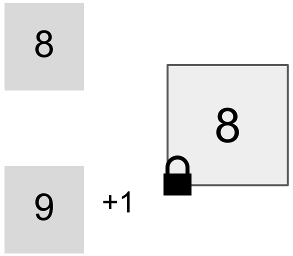
2. Thread 0 increments.



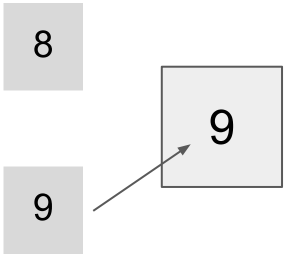
3. Thread 0 writes and unlocks.



4. Thread 1 locks and reads.



5. Thread 1 increments.



6. Thread 1 writes and unlocks.

OpenMP Lock Example: **Atomic Operation**

An atomic operation can only be executed by one thread at a time. Example: increment (++):

```
/* One thread at a time increments the burned tree count */
#pragma omp atomic
NBurnedTrees++;
```