Teaching Distributed-Memory Parallel Concepts with MPI

Joel Adams, Calvin College

Libby Shoop, Macalester College

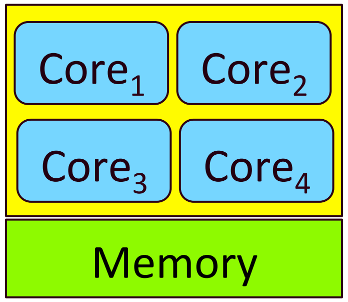
(Dick Brown, St. Olaf College)

Outline

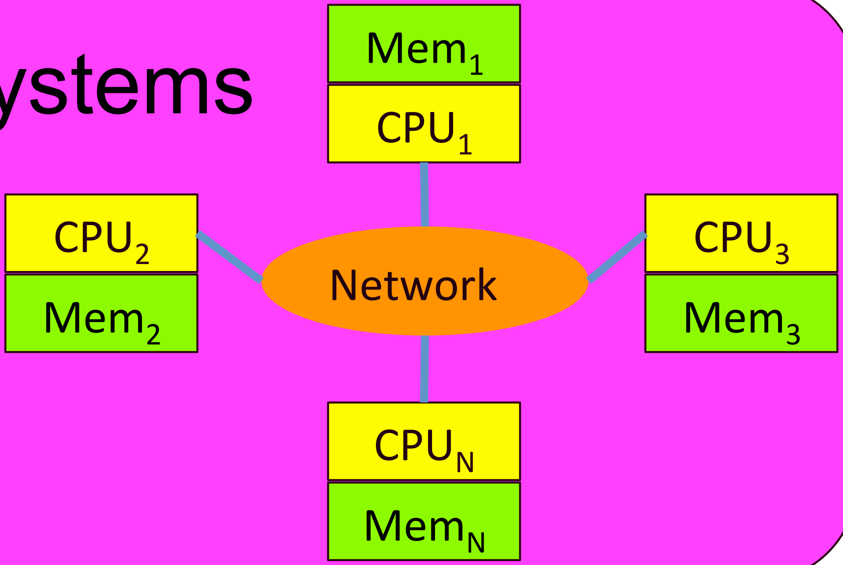
- Welcome and Introductions
- Part I: MPI Patternlets
  - Introduction to MPI (Joel)
  - Connecting to *cder.gsu.edu*(Joel)
  - The Patternlets module (Libby)
  - Self-paced exploration (You!)
- Break
- Part II: MPI Exemplars
- Wrap-up: Curricular discussion (Joel)

Hardware: A Diverse Landscape

- Shared-memory systems



- Distributed-memory systems



- Hybrid systems

Software: Multi*processing*

- Software *processes* run on each computer and *pass messages* via the network to communicate.
- Two basic options:
  1. Message-Passing *Libraries*:
     - The Message Passing Interface (MPI)
     - Language independence via multi-language bindings
  2. Message-Passing *Languages*:
     - Scala, Erlang, …

MPI …

- is an industry-standard library for distributed-memory parallel computing in C, C++, Fortran, with 3rd party bindings for Java, Python, R, …
- was designed by a large consortium:
  - 12 companies: *Cray, IBM, Intel, …*
  - 11 national labs: *ANL, LANL, LLNL, ORNL, Sandia, …*
  - representatives from 16 universities
- has many parallel design patterns "built in"

Typical MPI Program Structure

```
#include <mpi.h>                                 // MPI functions

int main(int argc, char** argv) {
    int id = -1, numProcesses = -1;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numProcesses);
    MPI_Comm_rank(MPI_COMM_WORLD, &id);

    // program body, usually including communication
    // calls (e.g., MPI_Send() and MPI_Receive())

    MPI_Finalize();
    return 0;
}
```

The 6 MPI Basic Functions

1. `MPI_Init(&argc, &argv);`
   – Set up **`MPI_COMM_WORLD`**, a “communicator”
     (The set of processes that make up the distr. computation)
2. `MPI_Comm_size(MPI_COMM_WORLD, &numProcesses);`
   – How many of us processes are there to attack the problem?
3. `MPI_Comm_rank(MPI_COMM_WORLD, &id);`
   – Which of the *n* processes am I?

The 6 MPI Basic Functions (2)

4. `MPI_Send(sendBuffAddress, numItems, itemType, destinationRank, tag, communicator);`
   - Send the item(s) at *sendBuffAddress* to *destinationRank*

5. `MPI_Recv(recvBuffAddress, bufferSize, itemType, senderRank, tag, communicator, status);`
   - Receive up to *bufferSize* items from *senderRank*

6. `MPI_Finalize();`
   - Shut down the distributed computation

These 6 are all you need to do useful work in MPI!

MPI Runtime

- To run an MPI *program* from the command line:

$$\underbrace{\texttt{mpirun}}\quad \underbrace{\texttt{-np } N}_{\text{Launch } N \text{ processes (each will get a unique rank) Vary } N \text{ to test scalability}}\quad \underbrace{\texttt{-machinefile } hostFile}_{\text{Launch those } N \text{ processes on the computers listed in } hostFile \text{ (optional on many clusters)}}\quad \underbrace{\texttt{./}program}_{\text{Each process runs this same } program \text{ (SPMD pattern)}}$$

Launch *N* processes
(each will get a unique rank)
Vary *N* to test scalability

Each process runs
this same *program*
(SPMD pattern)

Launch those *N* processes
on the computers listed in *hostFile*
(optional on many clusters)

Parallel Patterns

… are strategies that practitioners have repeatedly found to be useful in parallel problem-solving.

- Industry-standard best practices
  - These originated in *industry*, not academia
- Accumulated wisdom of decades of experience

When solving problems, experts *think* in patterns, so the more we can get our students to think in patterns, the more like experts they will be.

Categorizing Patterns

- *Algorithmic* Strategies:
  - *Data Decomposition, Task Decomposition, …*
- *Implementation* Strategies:
  - *SPMD, Master-Worker, Parallel Loop, …*
- *Concurrent Execution* Strategies:
  - *Barrier, Message Passing, Broadcast, Reduction, Scatter, Gather, …*

Higher level

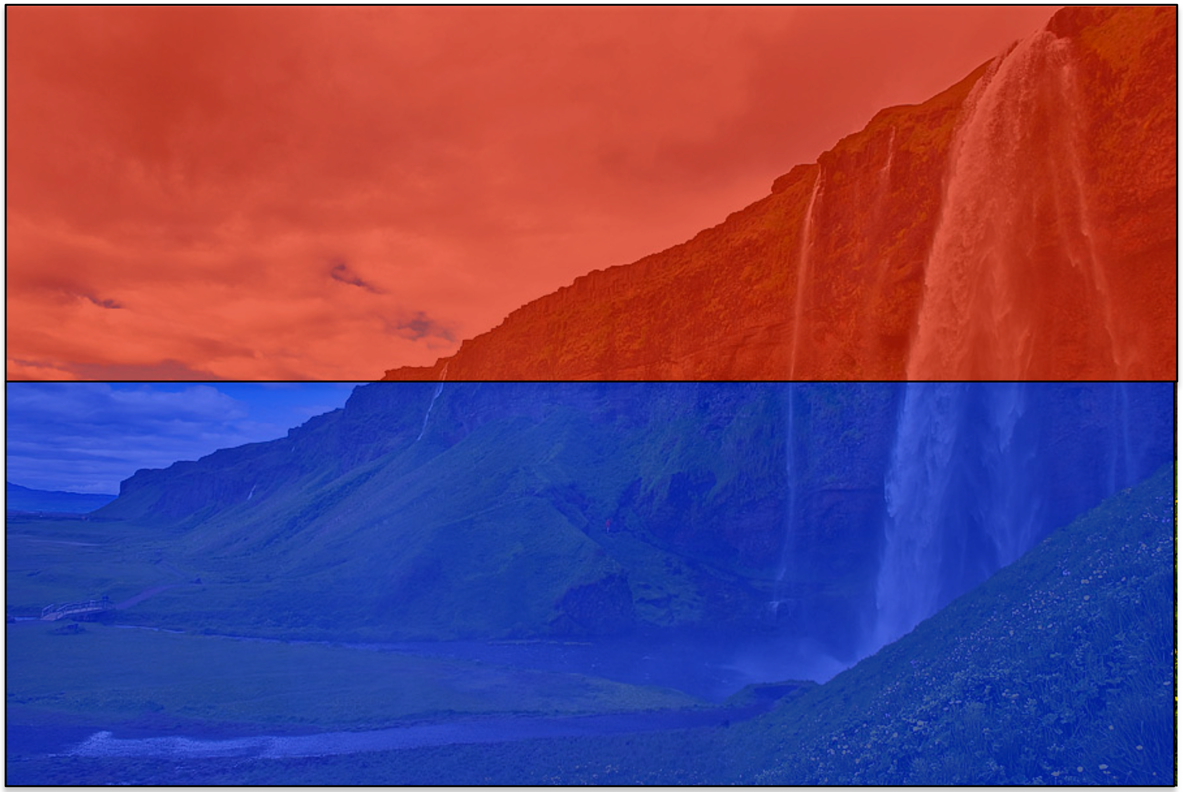
Lower level

Most MPI programs employ multiple patterns.

Data Decomposition (1 task)

Task 0

Data Decomposition (2 Tasks)



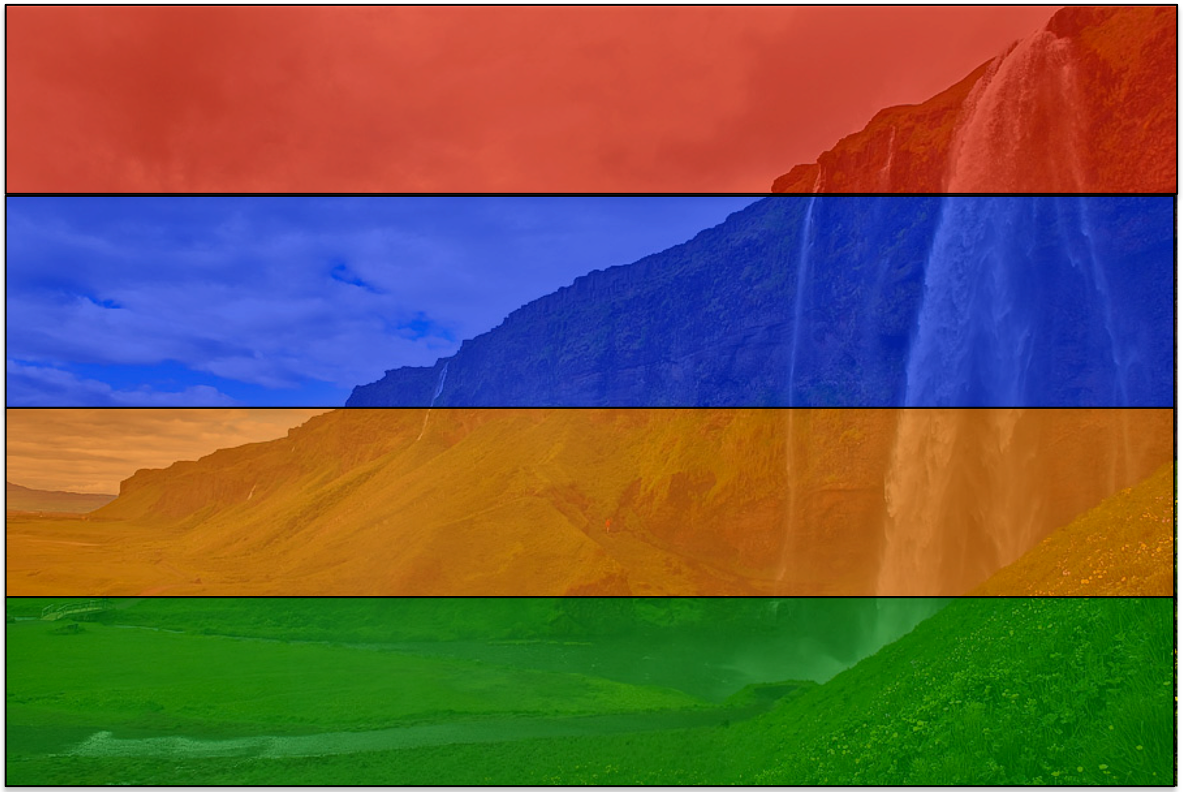
Data Decomposition (4 Tasks)

Task 0

Task 1

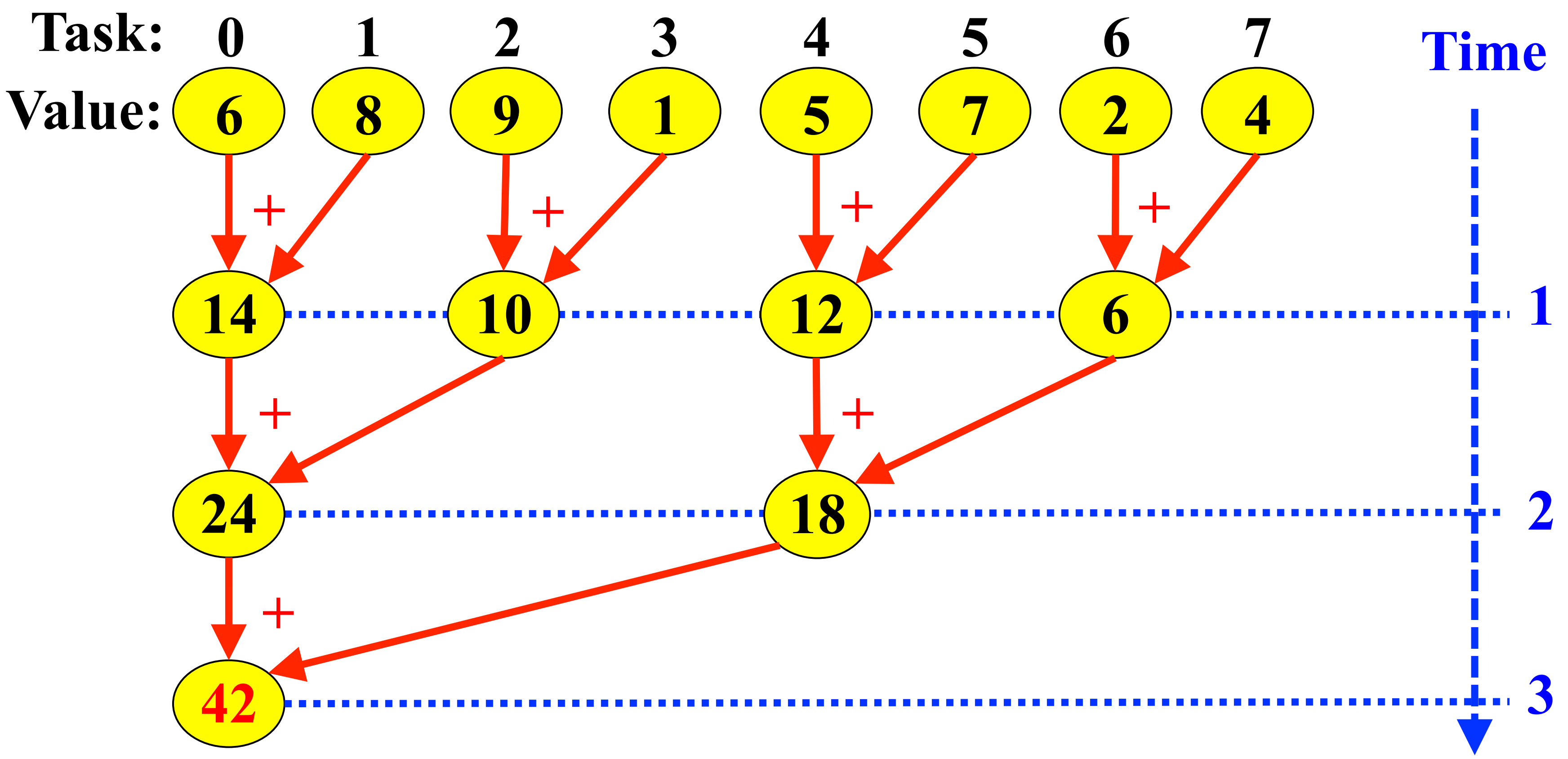
Task 2

Task 3



Reduction (8 Tasks)

To sum the local value-results of *N* parallel tasks:



Terminology: *Patternlets…*

are minimalist, scalable, and complete programs, each illustrating one or more parallel patterns:

- *Minimalist* to help students understand the pattern by eliminating non-essential details
- *Scalable* so that students can vary the number of processes and see the pattern's behavior change
- *Complete* for flexible use:
  - Instructors can use them in a 'live coding' lecture
  - Students can explore them in a hands-on exercise, and use them as models for their own programs.

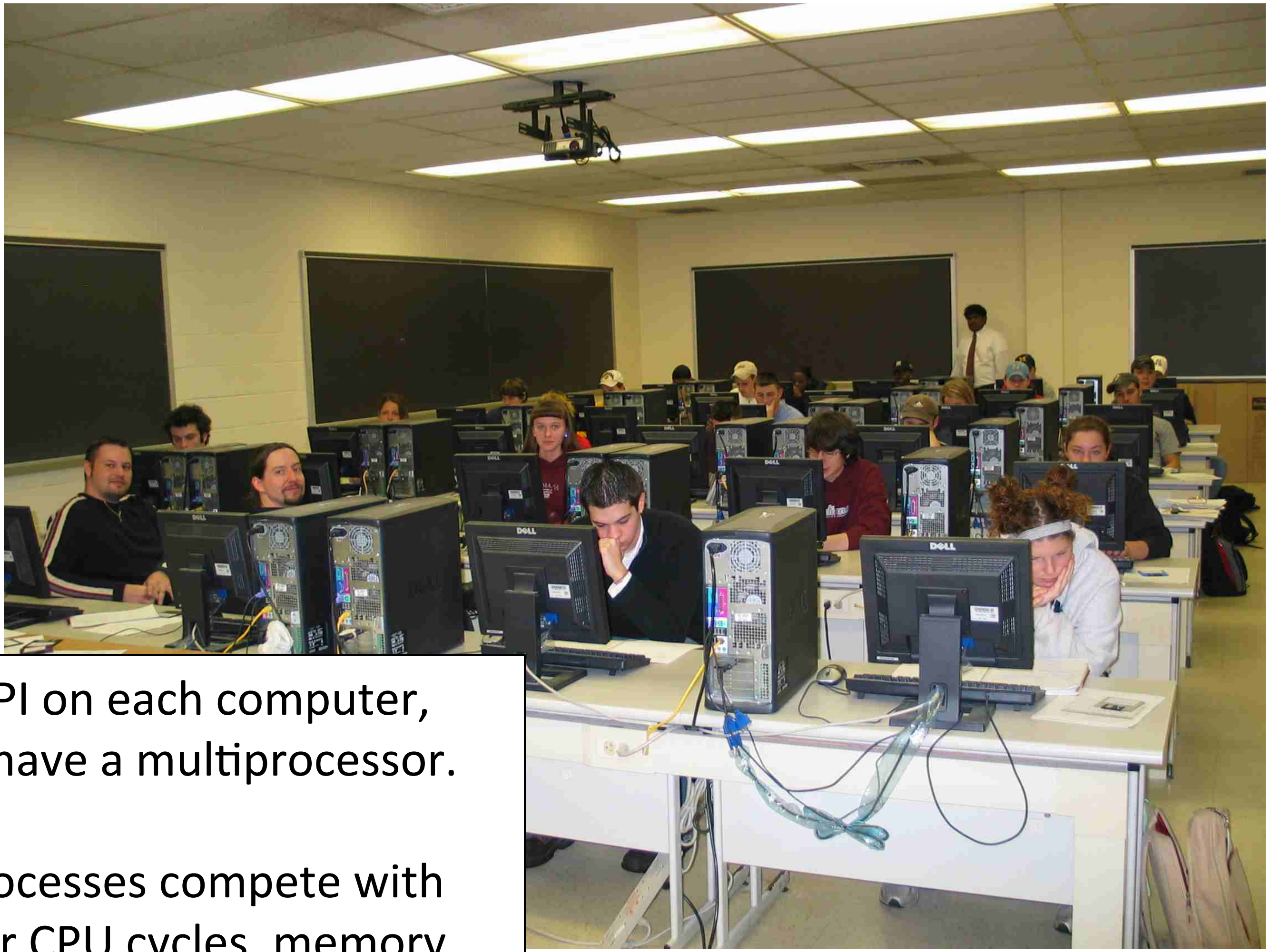
Terminology: *Exemplars…*

are programs that use the parallel patterns to solve a ‘real world’ problem.

Exemplars let students see how a pattern can be useful in a meaningful context.

A *patternlet* is useful for *introducing* students to a pattern; an *exemplar* is useful for helping students see how and why a pattern is *relevant*.

Hardware: NOW



Install MPI on each computer, and you have a multiprocessor.

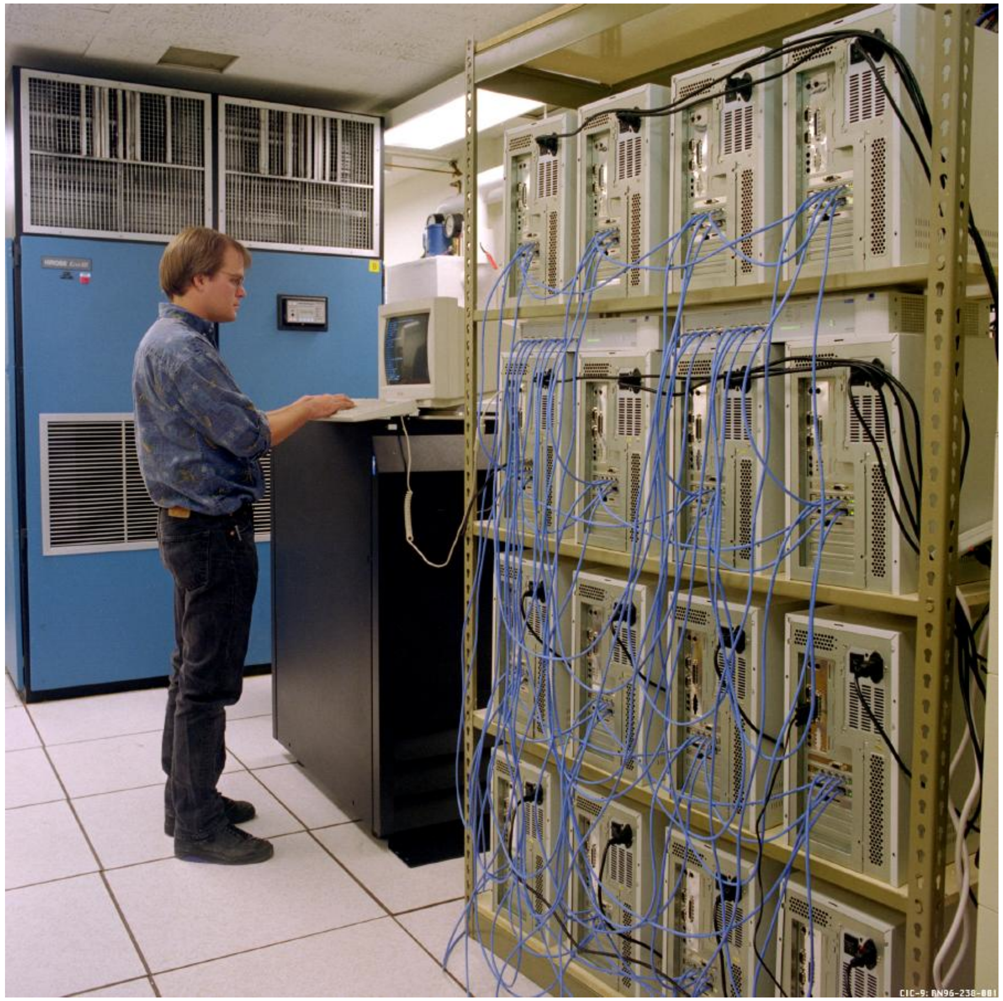
+ free!

- MPI processes compete with others for CPU cycles, memory, …

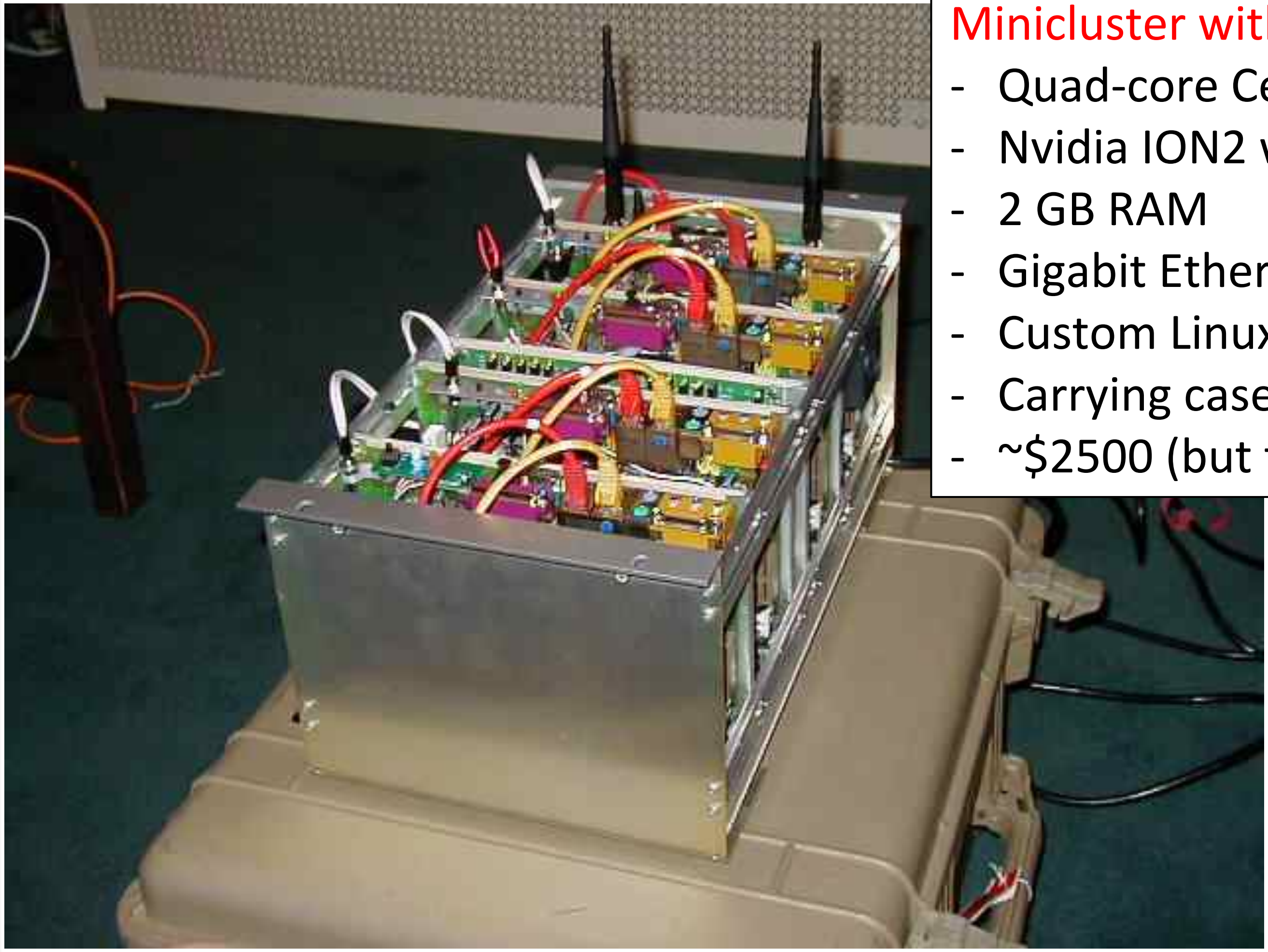
Hardware: Beowulf Clusters

Dedicated system;
you just need:
- Computers (nodes)
- A network through which they can communicate

We'll be using:
cder.gsu.edu



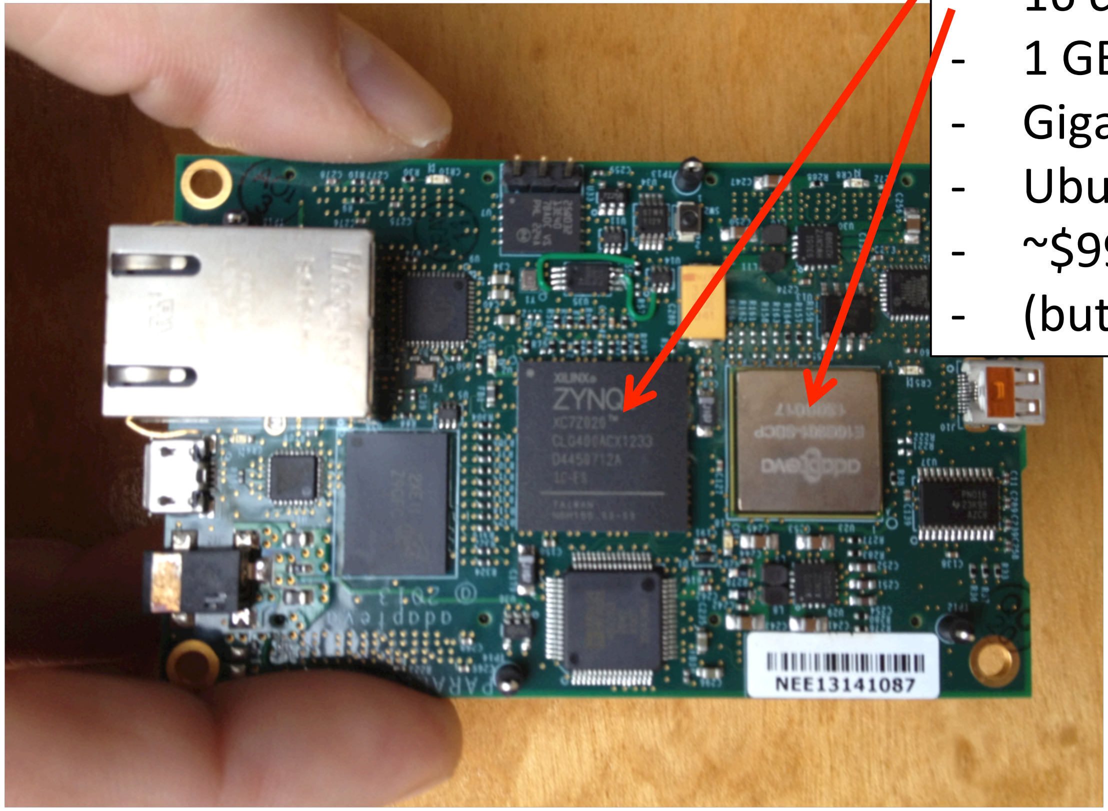
Hardware: LittleFe



Minicluster with 6 nodes, each with

- Quad-core Celeron
- Nvidia ION2 w/ 16 CUDA cores
- 2 GB RAM
- Gigabit Ethernet, USB, …
- Custom Linux distro (BCCD)
- Carrying case
- ~$2500 (but free at “Buildouts”!)

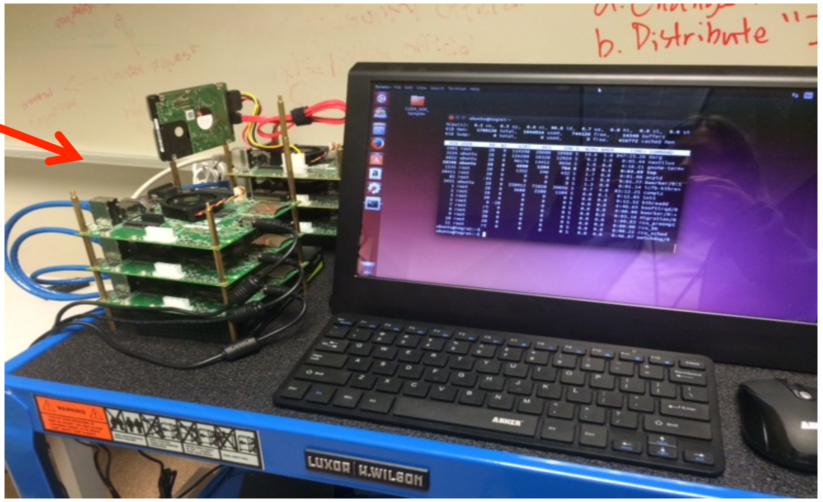
Hardware: Adapteva Parallella



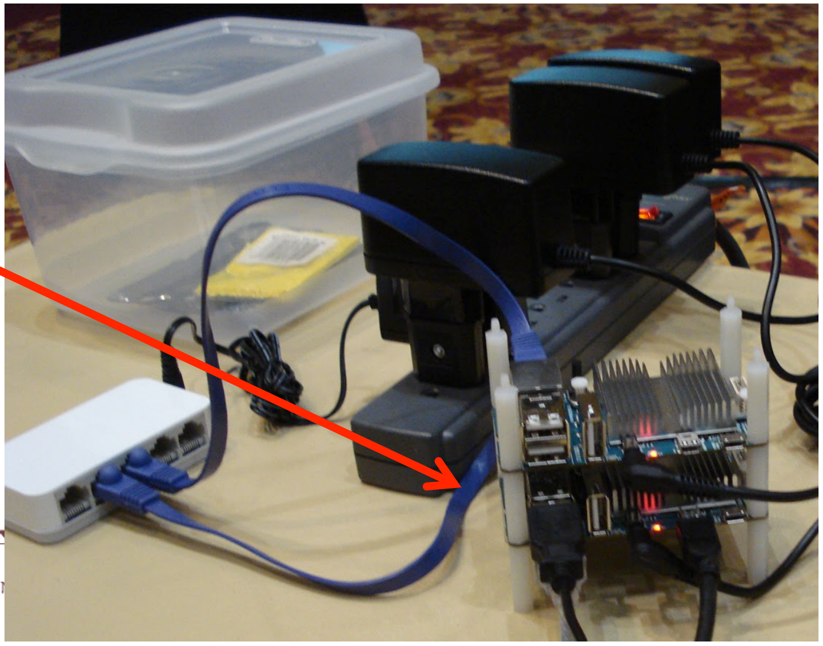
18-node "cluster on a board"
- Dual-core ARM A9
- 16 core Epiphany Coprocessor
- 1 GB RAM
- Gigabit Ethernet, USB, HDMI, …
- Ubuntu Linux
- ~$99
- (but free via university program!)

Hardware: Microclusters

Rosie (Libby Shoop, Macalester)
- 6 Nvidia Jetson TK-1 nodes
  - Quad-core ARM
  - 192 CUDA cores
  - 2 GB RAM
- Gigabit Ethernet
- ~$1345



HSC6 (Dave Toth, Centre)
- 2 ODROID XU4 nodes
  - 2 Quad-core ARM CPUs
  - 2 GB RAM
- Gigabit Ethernet
- ~$200



Outline

- Welcome and Introductions
- Part I: MPI Patternlets
  - Introduction (Joel) ✔
  - Connecting to *cder.gsu.edu* (Joel) ✔
  - The Patternlets module (Libby) ✔
  - Self-paced exploration (You!) ✔
- Break
- Part II: MPI Exemplars
- Wrap-up: Curricular discussion (Joel)

Outline

- Welcome and Introductions
- Part I: MPI Patternlets ✓
- Break ✓
- Part II: MPI Exemplars (Libby)
  - Concept: Data Decomposition Pattern ✓
  - Distributed Computing Fundamentals ✓
    - Area Under The Curve
    - Matrix Multiplication
  - Pandemic ✓
  - Self-paced exploration of Exemplars ✓
- Wrap-up: Curricular discussion + Evaluation (Joel) ✓

Thank you!