Lambda++

the power of cluster computing,
at your fingertips

Ananya Kumar, Jake Zimmerman

Functional Programming in C++ is A Good Thing™

- Same language that teams are already using
- Fast, compiled language
- Lambda functions in C++11 are concise

```
bool paren_match(Sequence<int> &seq) {
  auto plus = [](int a, int b) { return a + b; };
  auto min  = [](int a, int b) { return a < b ? a : b; };
  seq.scan(plus, 0);
  return seq.get(seq.length() - 1) == 0 &&
         seq.reduce(min, INT_MAX) >= 0;
}
```

The **Sequence** Abstraction

2	5	1	-3	4	8

- Feels like an array on steroids
- Supports any data type
  - ints, floats, pairs, structs, and more (types!)
- Thrives with higher order functions
  - map, scan, reduce, tabulate

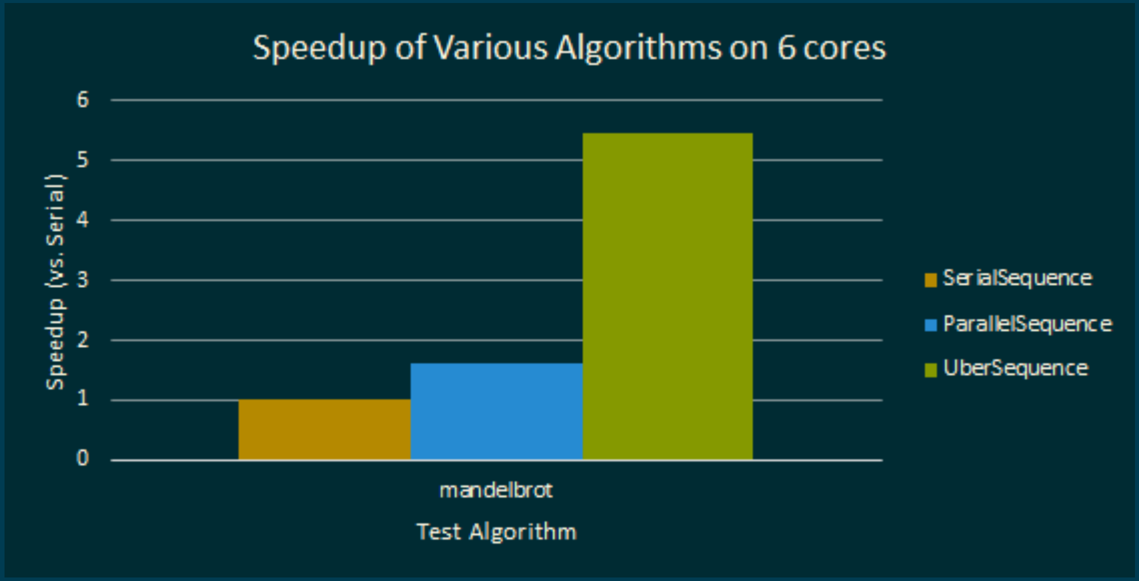
Some Implementation Jargon

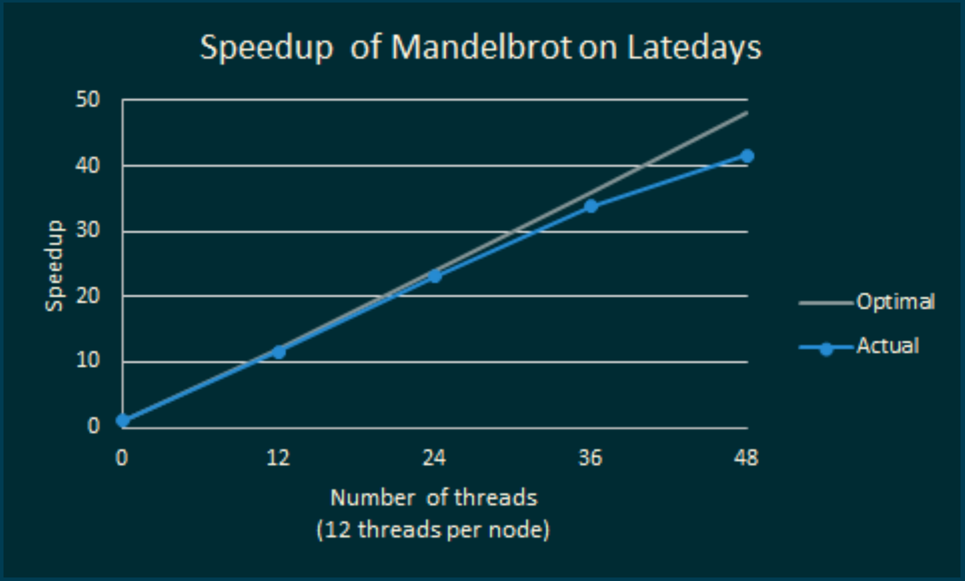
- `mandelbrot`
  - compute elements of mandelbrot set
- `paren_match`
  - determine whether parentheses are well-matched

- `SerialSequence`
  - Implement abstract Sequence class serially
- `ParallelSequence`
  - Parallel Sequence class implementation (naive)
- `UberSequence`
  - Heavily optimized parallel Sequence implementation

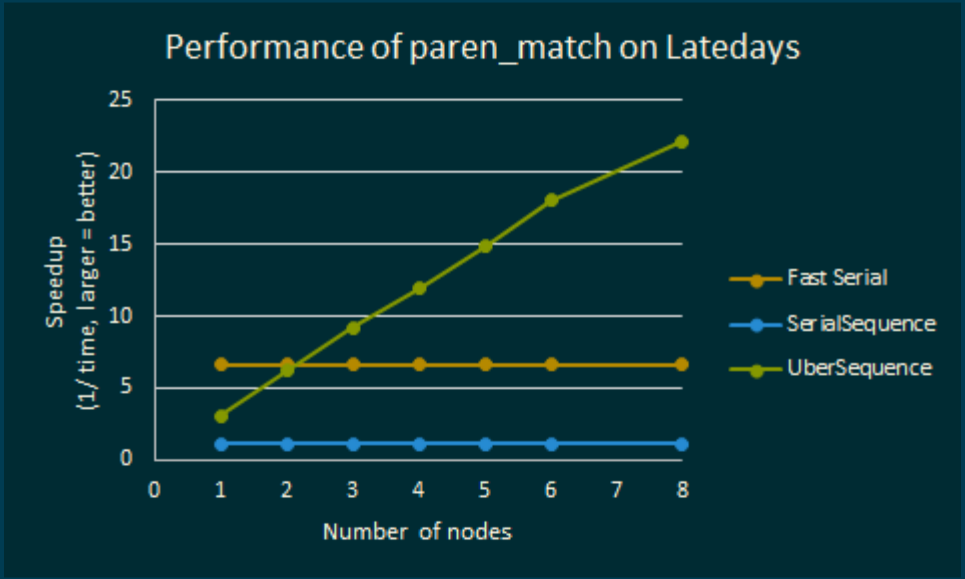
Near optimal speedup on `mandelbrot`

- Tabulate to compute mandelbrot set elements
- Data parallel, math heavy
- Reaches near-peak speedup on GHC and Latedays



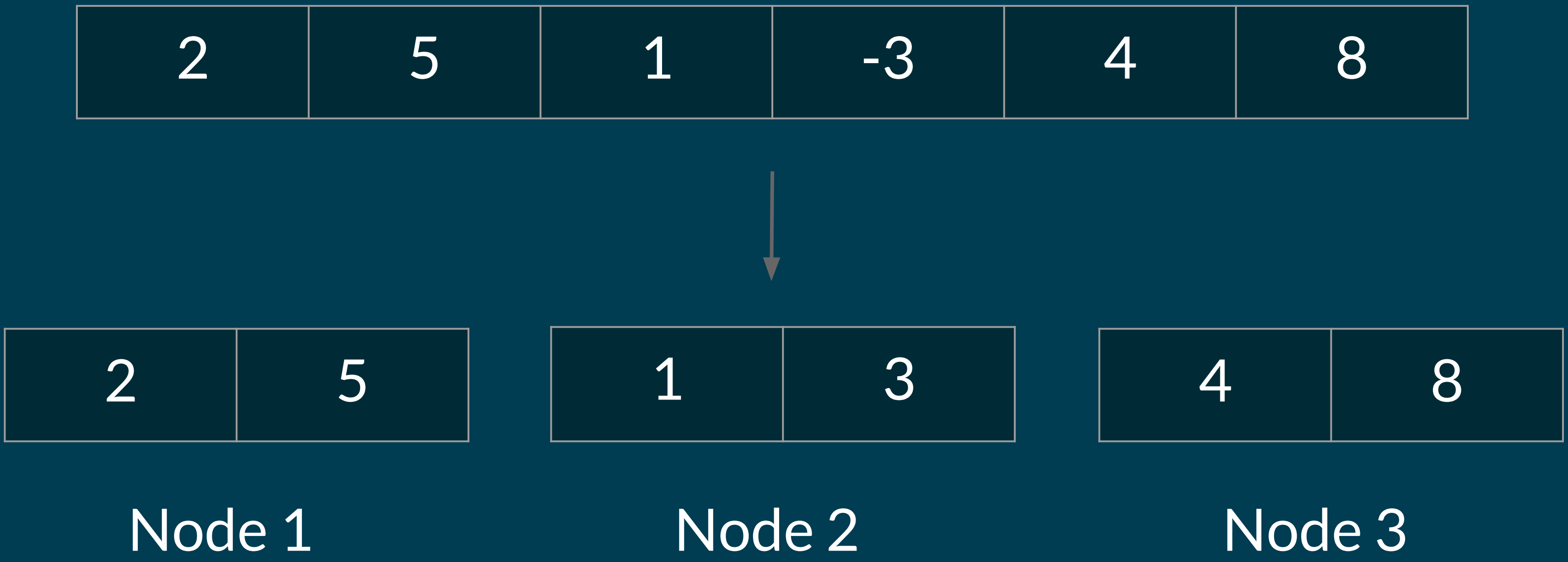


Speedup on `paren_match`



- scan + reduce to determine well-matchedness
- much harder to parallelize (not data parallel)
  - Fast Serial wins on one and two nodes
- Linear speedup with number of nodes

Data is Spread across Nodes



2	5	1	-3	4	8

2	5

Node 1

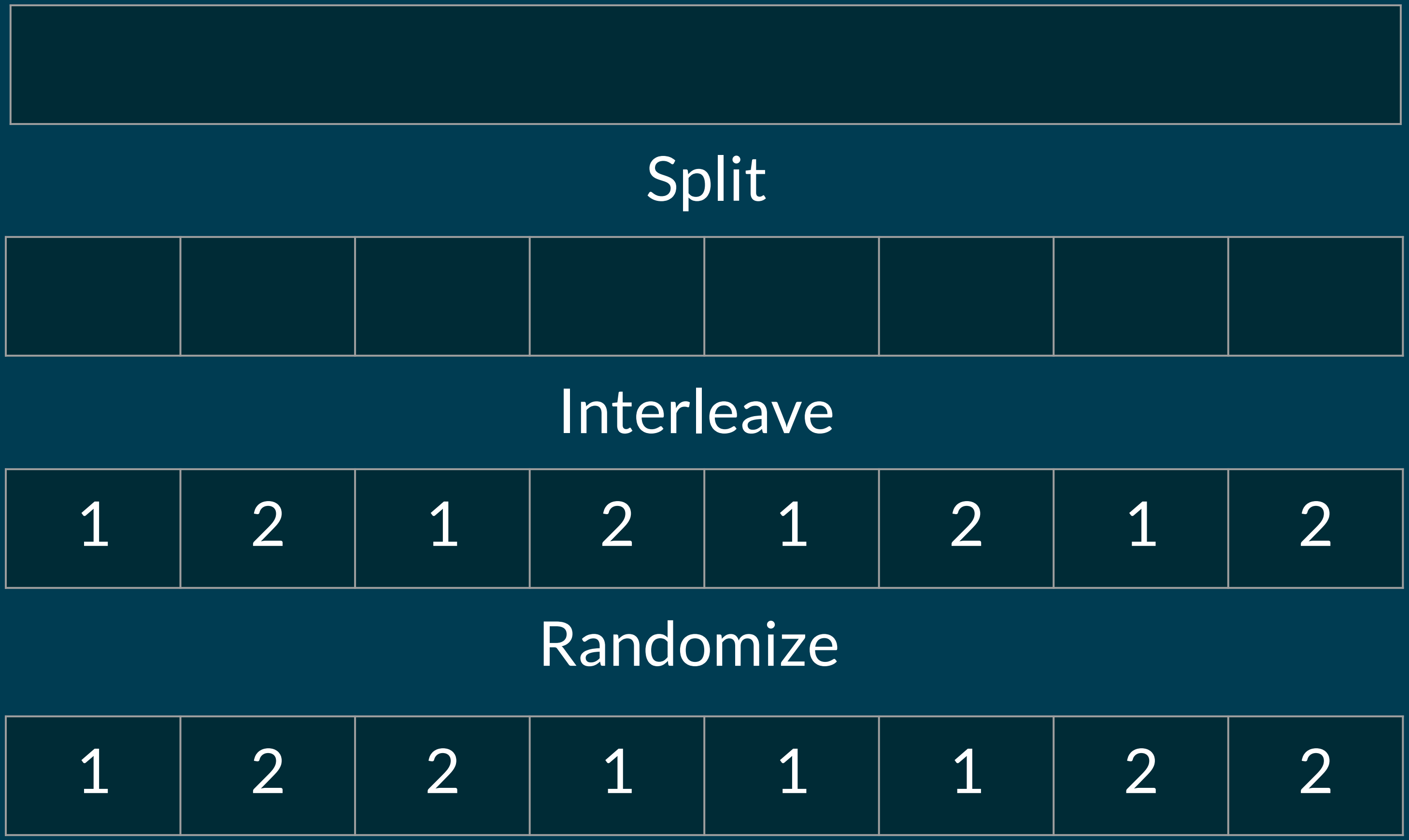
1	3

Node 2

4	8

Node 3

Randomized Load Distribution



Split

Interleave

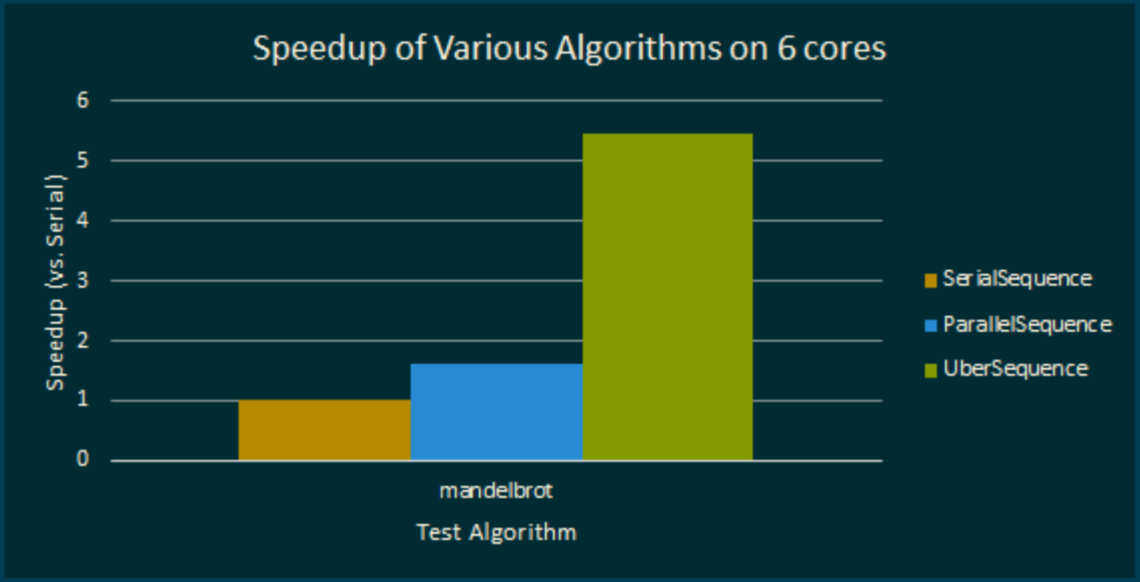
1 2 1 2 1 2 1 2

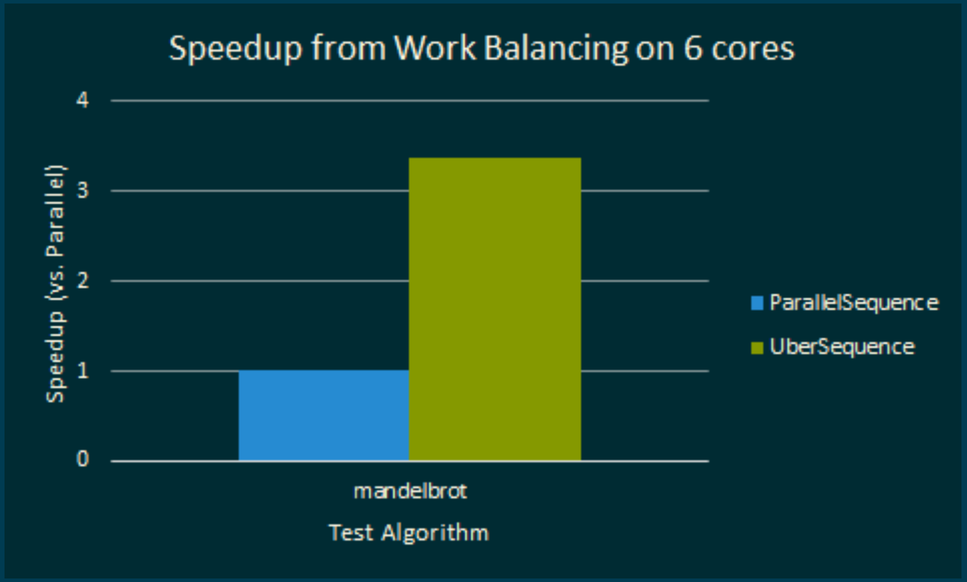
Randomize

1 2 2 1 1 1 2 2

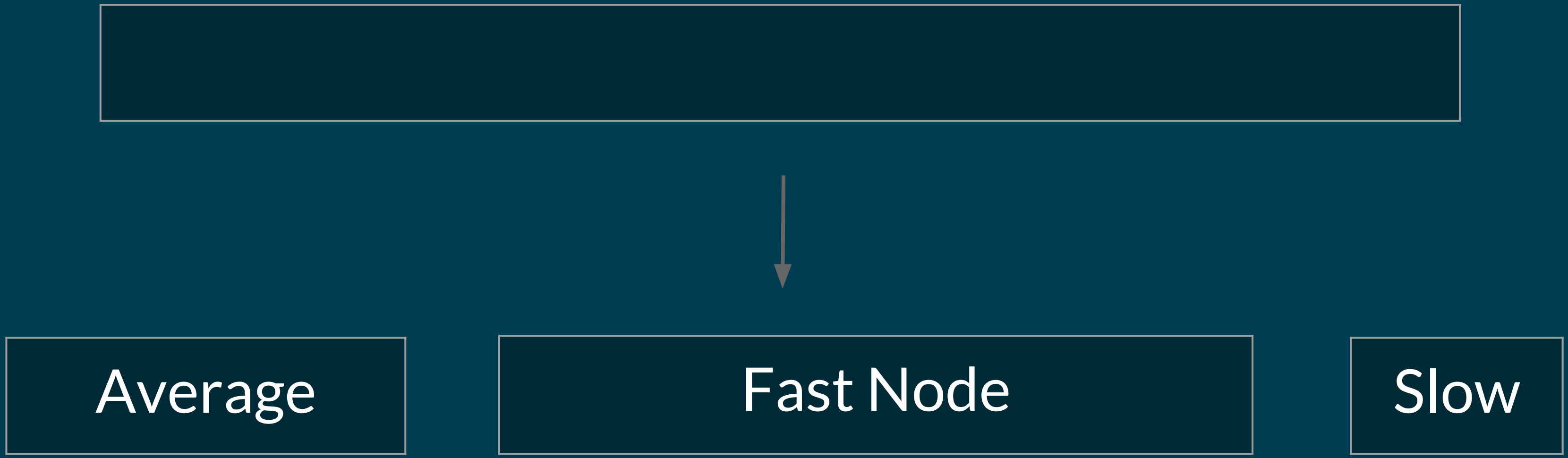
Load Distribution Improves Performance

- ParallelSequence has no load distribution
- Load distribution is responsible for a big part of the speedup in UberSequence

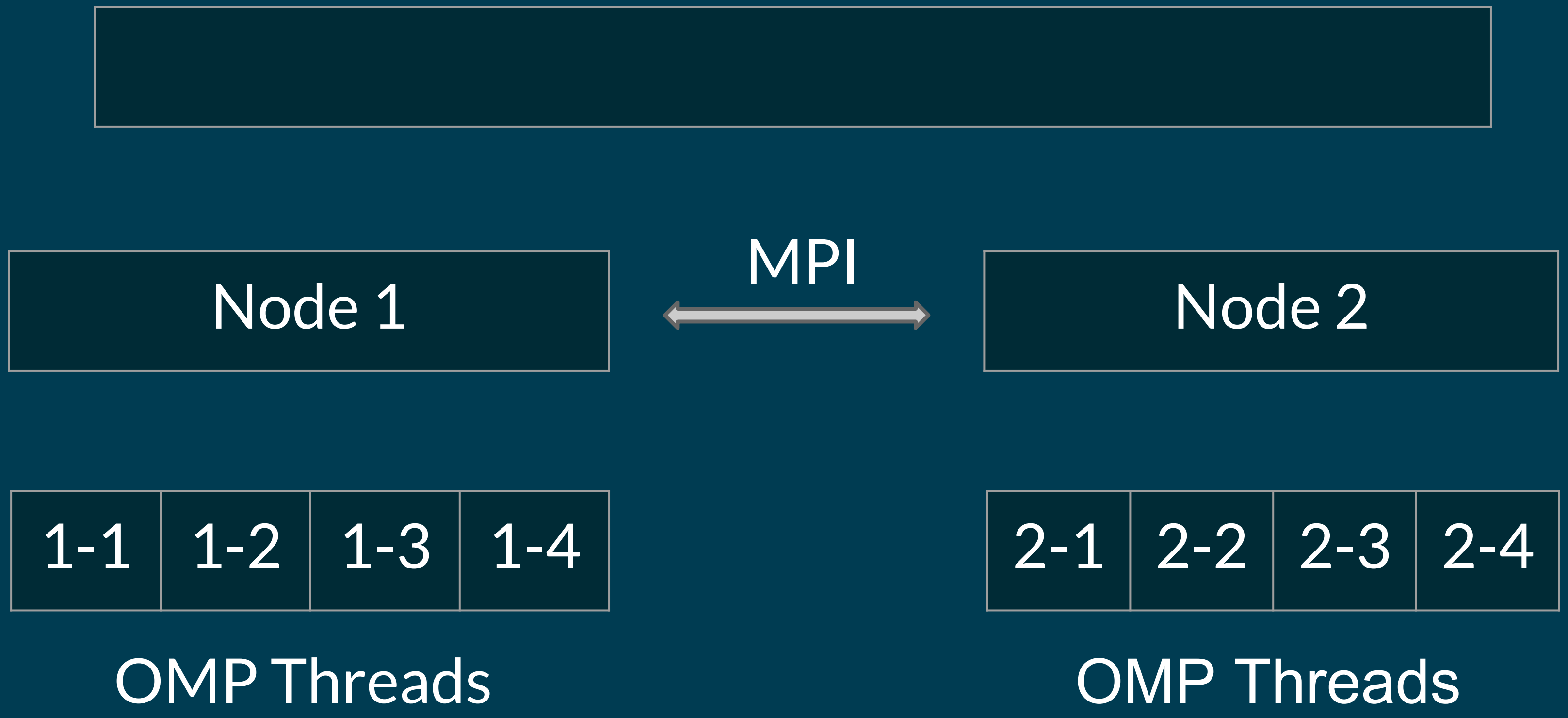




Profile cluster to determine fastest nodes



MPI Between Nodes, OpenMP within a Node



MPI

Node 1

Node 2

1-1 1-2 1-3 1-4

2-1 2-2 2-3 2-4

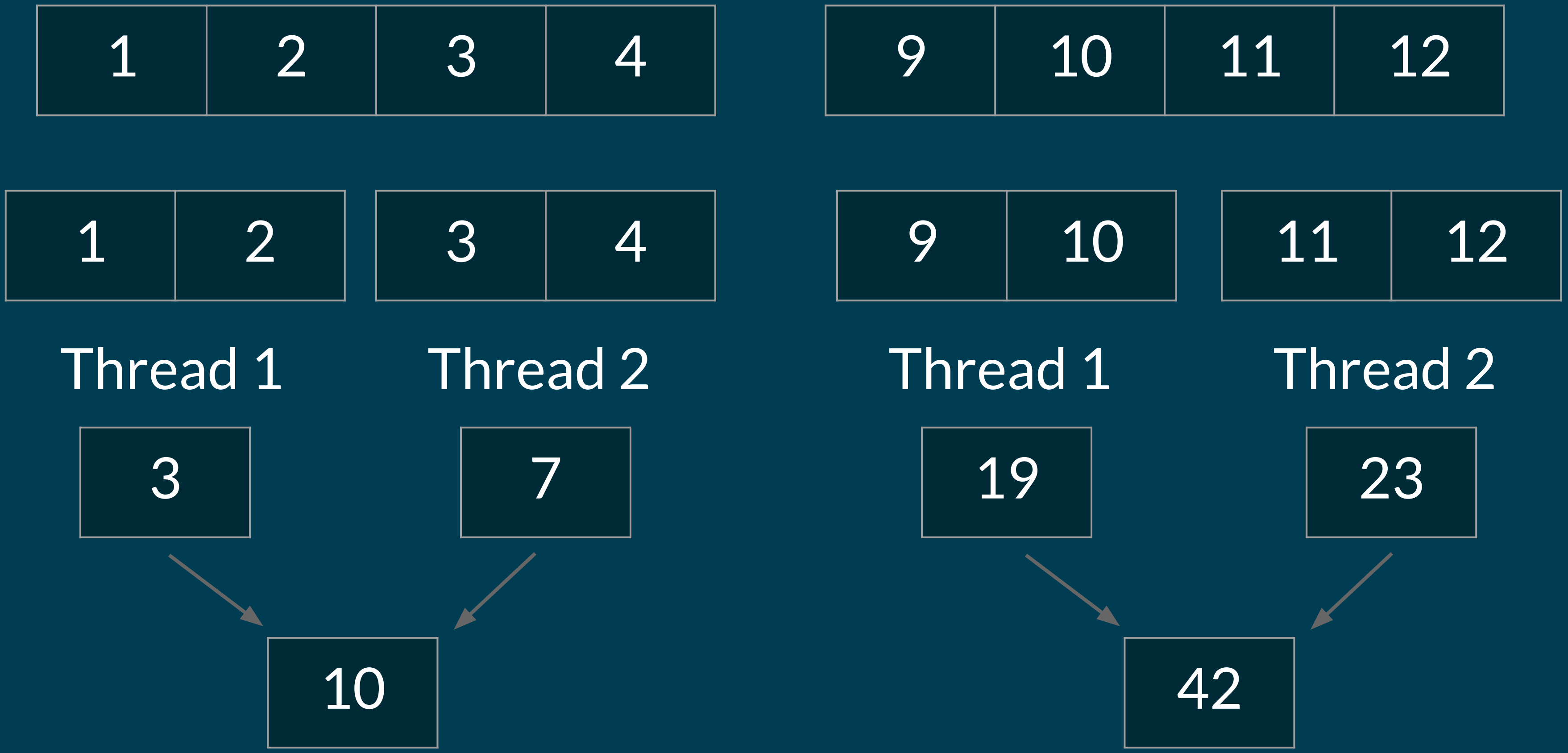
OMP Threads

OMP Threads

Maximize utilization of available resources

Reduce:
Parallelizes Well with MPI + OpenMP

Stage 1 - Node 1



1 2 3 4

9 10 11 12

1 2

3 4

9 10

11 12

Thread 1

Thread 2

Thread 1

Thread 2

3

7

19

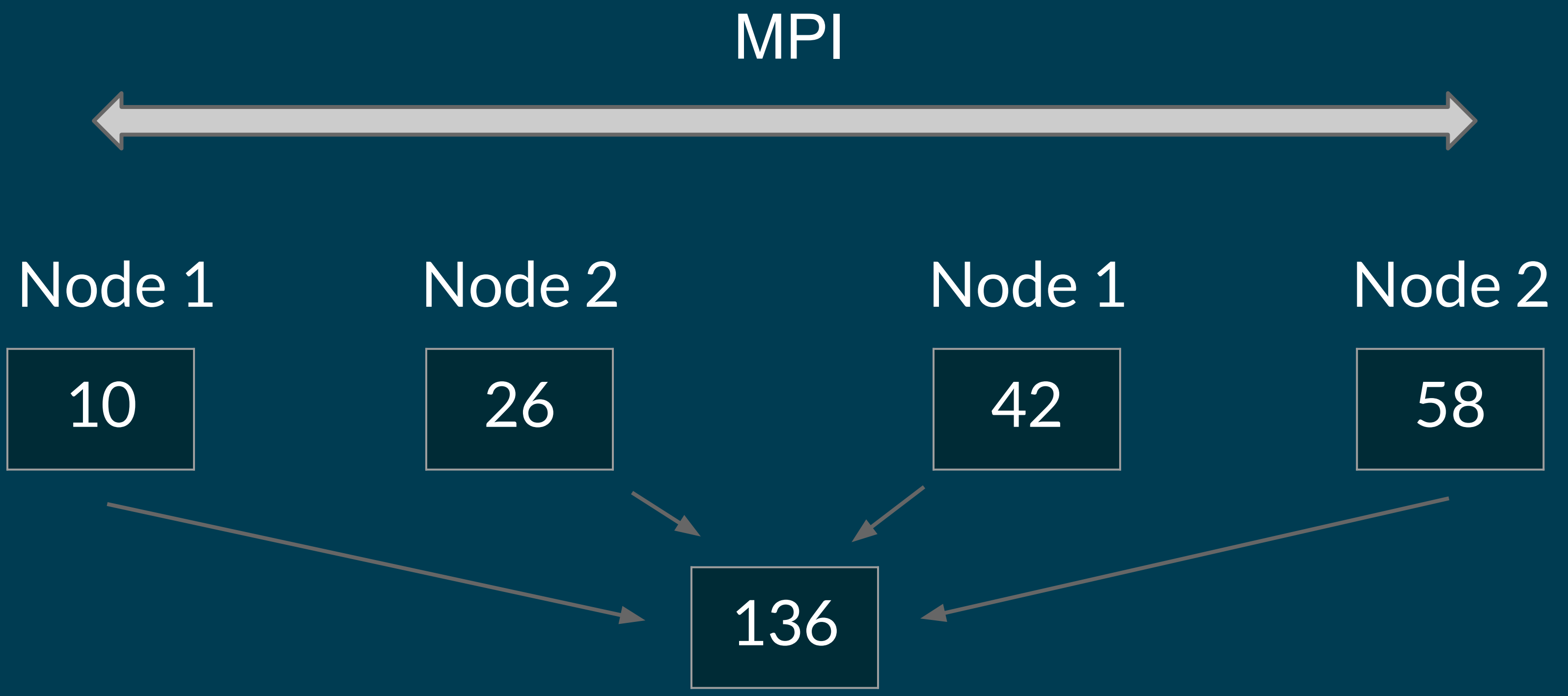
23

10

42

Reduce: Minimizes Communication

Stage 2 - Node 1



Questions

Lambda++

the power of cluster computing,
at your fingertips

Ananya Kumar, Jake Zimmerman

Lambda++ is...

a C++ functional programming library

designed for running algorithms

across a cluster

that

approaches near optimal speedups

while enabling programmers to write

concise, functional code.