Parallel Programming

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

WS 16/17

Communicators

```
MPI_Comm_split(
    MPI_Comm comm,          <- EVERYBODY involved!
    int color,              <- where do I belong
    int key,                <- for the new rank
    MPI_Comm* newcomm)
```

Communicators

```
MPI_Comm_split(
    MPI_Comm comm,          <- EVERYBODY involved!
    int color,              <- where do I belong
    int key,                <- for the new rank
    MPI_Comm* newcomm)
```

```
MPI_Comm_dup
MPI_Comm_create
MPI_Comm_create_group
MPI_Group_incl
...
MPI_Group_union, MPI_Group_intersection
```

Topologies: Who are my neighbors?

Cartesian topology

```
MPI_Cart_create(
    old_communicator,
    ndims,                  <- number of dimensions of the grid
    dims[],                 <- size of each dimension
    periods[],              <- periodic boundaries?
    reorder,
    *cart_communicator      <- new communicator
    )
```

Topologies: Who are my neighbors?

Cartesian topology

```
MPI_Cart_create(
    old_communicator,
    ndims,                  <- number of dimensions of the grid
    dims[],                 <- size of each dimension
    periods[],              <- periodic boundaries?
    reorder,
    *cart_communicator      <- new communicator
    )
```

Who am I?

```
MPI_Comm_rank(   cart_COMM, &rank );
MPI_Cart_coords( cart_COMM, rank, ndims, coords[] );
```

Topologies: Who are my neighbors?

Cartesian topology

```
MPI_Cart_create(
    old_communicator,
    ndims,                  <- number of dimensions of the grid
    dims[],                 <- size of each dimension
    periods[],              <- periodic boundaries?
    reorder,
    *cart_communicator      <- new communicator
    )
```

Who am I?

```
MPI_Comm_rank(   cart_COMM, &rank );
MPI_Cart_coords( cart_COMM, rank, ndims, coords[] );
```

My neighbors? No neighbor: `MPI_PROC_NULL`

```
MPI_Cart_shift( cart_COMM, direction, displacement, *rank_source, *rank_dest )
```

Topologies: Who are my neighbors?

Cartesian topology

```
MPI_Cart_create(
    old_communicator,
    ndims,                  <- number of dimensions of the grid
    dims[],                 <- size of each dimension
    periods[],              <- periodic boundaries?
    reorder,
    *cart_communicator      <- new communicator
    )
```

Who am I?

```
MPI_Comm_rank(   cart_COMM, &rank );
MPI_Cart_coords( cart_COMM, rank, ndims, coords[] );
```

My neighbors? No neighbor: `MPI_PROC_NULL`

```
MPI_Cart_shift( cart_COMM, direction, displacement, *rank_source, *rank_dest )
```

Other topologies: `MPI_Graph_create`

Patterns for parallel computing

MPI's perspective

- Pipeline
  $\rightarrow$ Ok, but what if the different stages differ significantly in cost?

Patterns for parallel computing

MPI's perspective

- Pipeline
  $\rightarrow$ Ok, but what if the different stages differ significantly in cost?

- Fork-join
  $\rightarrow$ Not really

Patterns for parallel computing

MPI's perspective

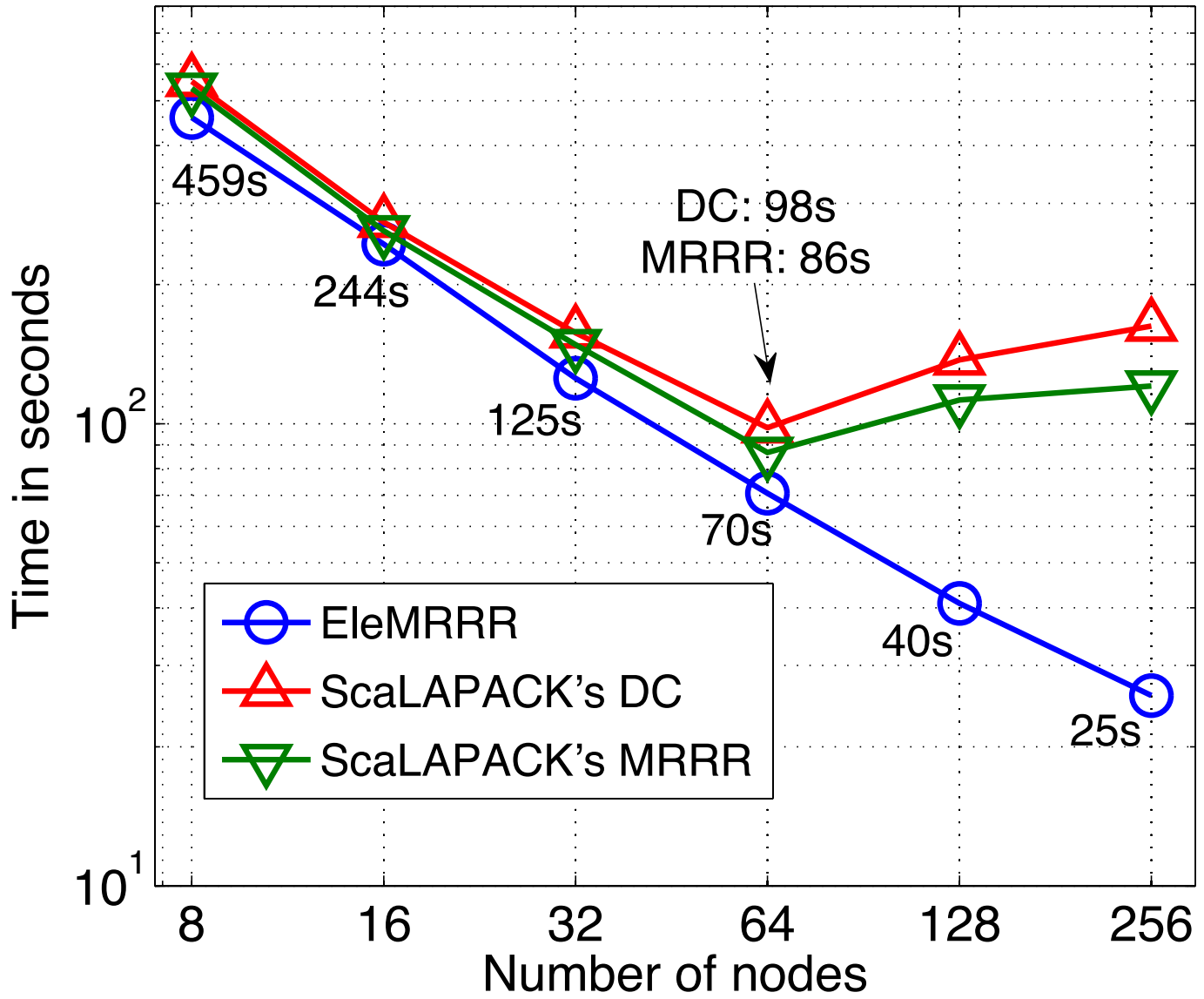
- Pipeline
  $\rightarrow$ Ok, but what if the different stages differ significantly in cost?

- Fork-join
  $\rightarrow$ Not really

- Master-slave
  $\rightarrow$ Yes, absolutely

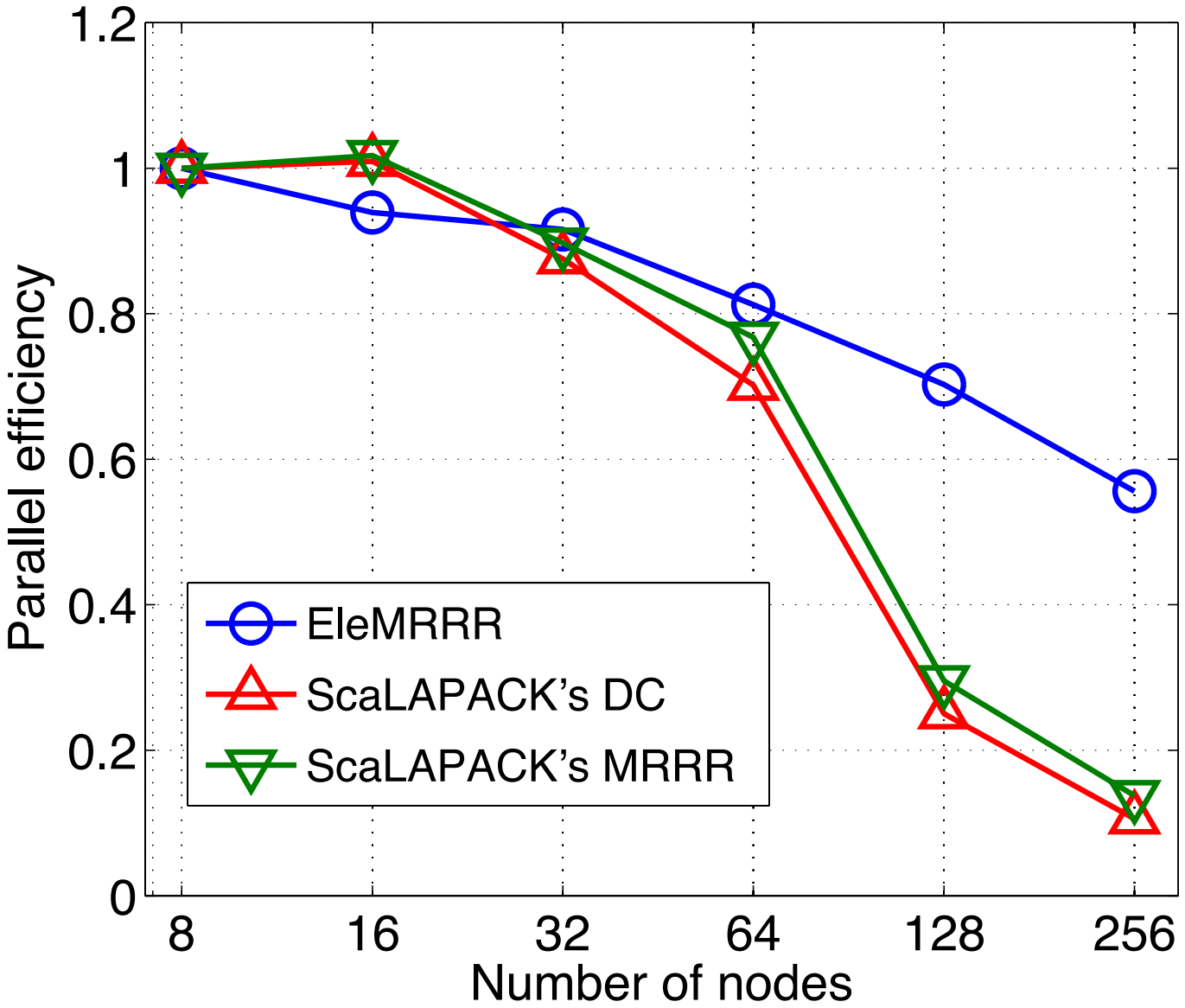
Strong scalability: timings

$n = 20.000$



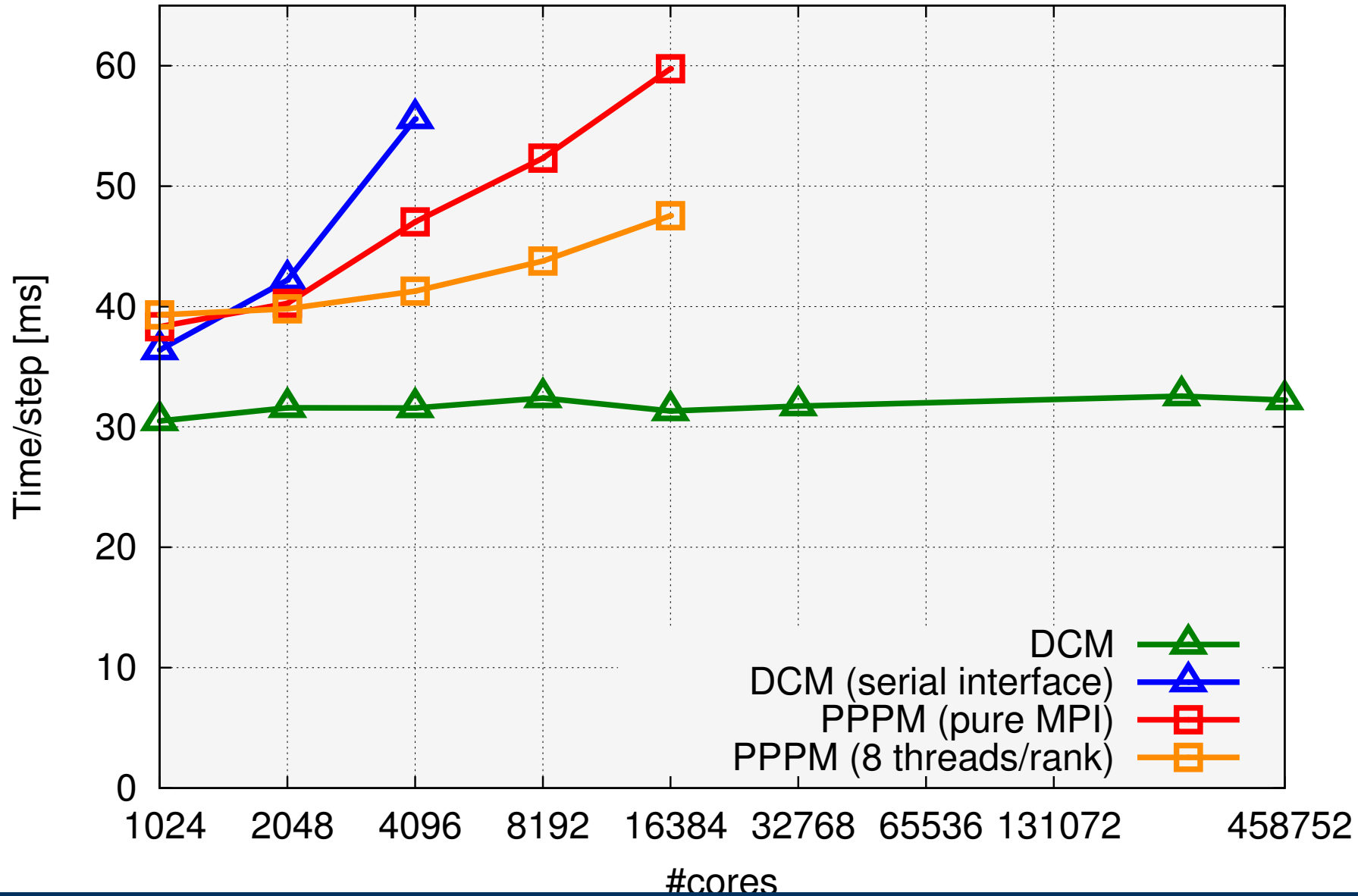
Strong scalability: efficiency

$n = 20.000$



Weak scalability: timings

10.000 particles per rank



Back to Broadcast

- Lower bound: $\log_2(p)\alpha + n\beta$
- MST algorithm: $\log_2(p)\alpha + \log_2(p)n\beta$

  Can we do better?

Back to Broadcast

- Lower bound: $\log_2(p)\alpha + n\beta$
- MST algorithm: $\log_2(p)\alpha + \log_2(p)n\beta$

  Can we do better?

- Idea for large n: first `Scatter`, then `Allgather`

Back to Broadcast

- Lower bound: $\log_2(p)\alpha + n\beta$
- MST algorithm: $\log_2(p)\alpha + \log_2(p)n\beta$

  Can we do better?

- Idea for large n: first `Scatter`, then `Allgather`
- Cost: $\log_2(p)\alpha + \frac{p-1}{p}n\beta \quad + \quad \log_2(p)\alpha + \frac{p-1}{p}n\beta$

  $\approx 2\log_2(p)\alpha + 2n\beta$

- Weight shifted from β to α; a factor of 2 from optimal

Back to Broadcast

- Lower bound: $\log_2(p)\alpha + n\beta$
- MST algorithm: $\log_2(p)\alpha + \log_2(p)n\beta$

  Can we do better?

- Idea for large n: first `Scatter`, then `Allgather`
- Cost: $\log_2(p)\alpha + \frac{p-1}{p}n\beta \quad + \quad \log_2(p)\alpha + \frac{p-1}{p}n\beta$

  $\approx 2\log_2(p)\alpha + 2n\beta$

- Weight shifted from β to α; a factor of 2 from optimal

- **In practice:** $p = r \times c \quad \Rightarrow$ 2-stage algorithms

Proposed exercise: Allgather

- Implement an `Allgather` as a sequence of broadcasts.
- Time and compare your implementation with MPI's `Allgather`; test different message lengths and different numbers of ranks.

Proposed exercise: master-slave

- Rank `root` has access to an input device.
  `root` performs an infinite loop, reading from the device, sending tasks to all other ranks, and collecting (and printing) the results.
  `root` does not execute tasks itself.

- Simulate the input device with a text file, containing a long list of integers (one per row); each integer denotes the "length" (size) of a chunk of input data.
  The idea is that `root` either reads or creates such data, and sends it to one worker, for processing.

- The workers execute an infinite loop, waiting for an incoming chunk data, processing it, and sending the result to `root`. (Alternatively, the results are handled by another designated process `dest`).

- Objective: minimize (eliminate) wait times.