

LINMA2710 - Scientific Computing

Distributed Computing with MPI

P.-A. Absil and B. Legat

Full Width Mode Present Mode

≡ Table of Contents

Single Program Multiple Data (SPMD)

Collectives

Distributed sum

Point-to-point

Consortium des Équipements de Calcul Intensif (CÉCI)

Topology

[Eij10] V. Eijkhout. *Introduction to High Performance Scientific Computing*. 3 Edition, Vol. 1 (Lulu.com, 2010).

[Eij17] V. Eijkhout. *Parallel Programming in MPI and OpenMP*. 2 Edition, Vol. 2 (Lulu.com, 2017).

Single Program Multiple Data (SPMD)

Message Passing Interface (MPI)

- MPI  is an open standard for distributed computing
- Many implementations:
 - MPICH, from Argonne  and  MISSISSIPPI STATE UNIVERSITY.
 - Open MPI 
 - commercial implementations from , , Microsoft, and 

MPI basics

Initializes MPI, remove mpiexec , etc... from argc and argv .

```
MPI_Init(&argc, &argv)□
```

Get the number of processes. nprocs is the **same** on all processes.

```
int nprocs;  
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);□
```

Get the id of processes. procid is the **different** for **different** processes.

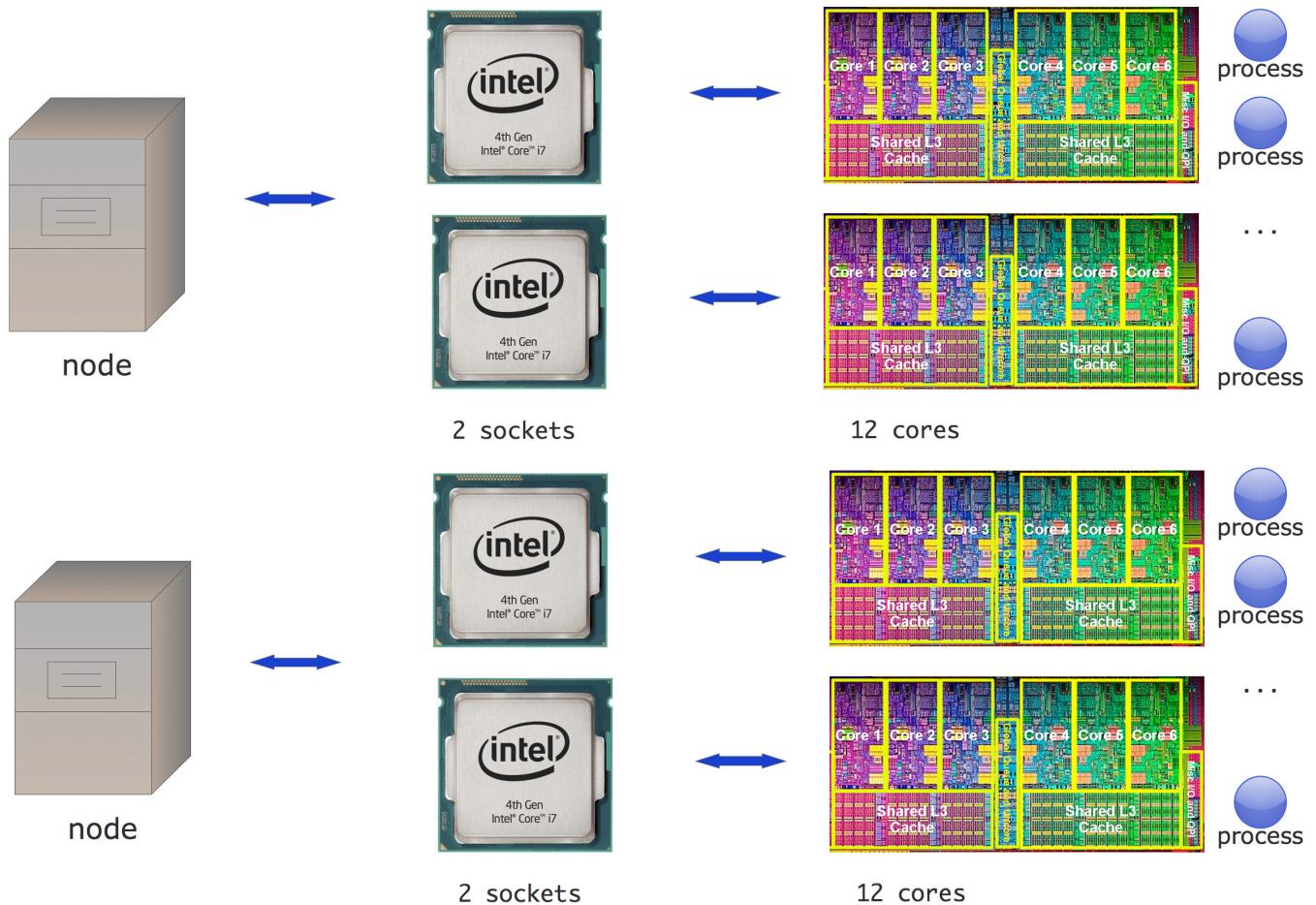
```
int procid;  
MPI_Comm_rank(MPI_COMM_WORLD, &procid);□
```

Free up memory.

```
MPI_Finalize();□
```

► **Each process runs the same executable. So how can we make them do different things ?**

Different processes may be on the same node



Processor name identifies the node

Processes that are on the same node share the same processor_name (the hostname).

```
int name_length = MPI_MAX_PROCESSOR_NAME;
char proc_name[name_length];
MPI_Get_processor_name(proc_name,&name_length);
printf("Process %d/%d is running on node <<%s>>\n",
    procid,nprocs,proc_name);
```

① Compiling : `mpicc -O3 /tmp/jl_hX5QAw/main.c -o /tmp/jl_hX5QAw/bin`

① Running : `mpiexec -n 2 /tmp/jl_hX5QAw/bin`

Process 1/2 is running on node <<pkrvmf6wy0o8zjz>>
Process 0/2 is running on node <<pkrvmf6wy0o8zjz>>

num_processes = 2

Compiling

You could simply add `lmpi` bu using `mpicc` and `mpic++` is easier.

```
Process('mpicc -show', ProcessExited(0))
```

1 run(`mpicc -show`)

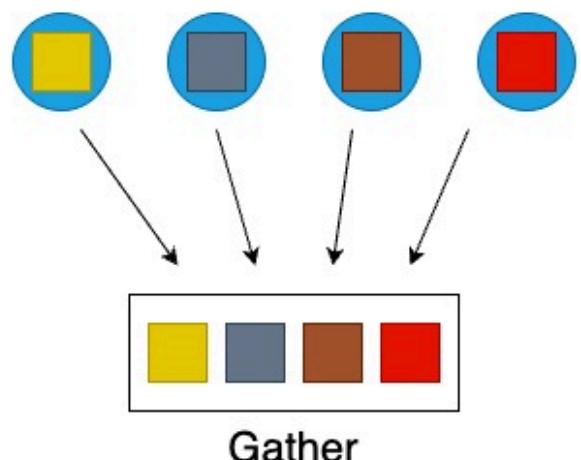
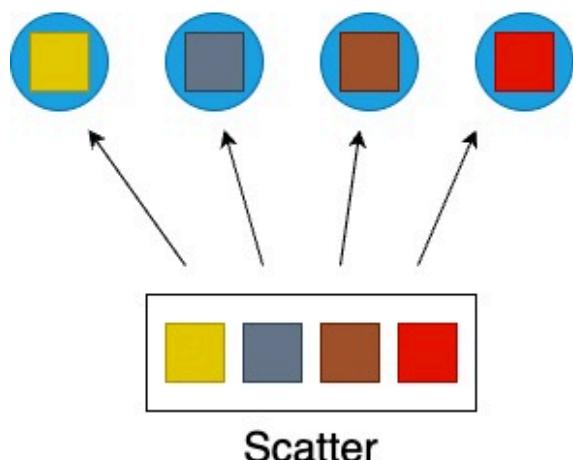
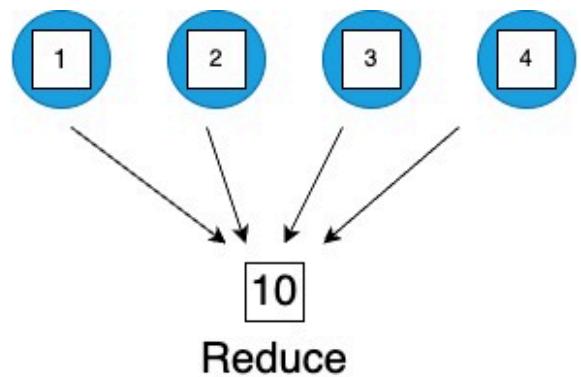
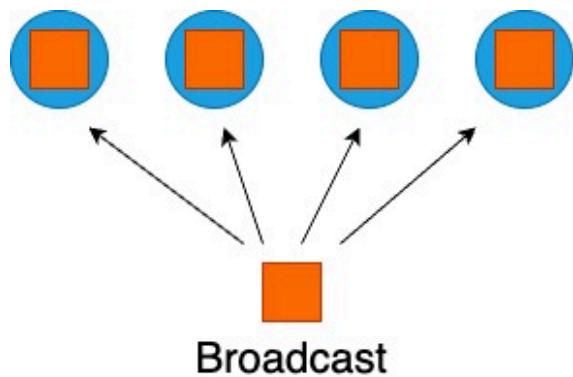
gcc -I/usr/lib/x86_64-linux-gnu/openmpi/include -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lmpi

```
Process('mpic++ -show', ProcessExited(0))
```

1 run(`mpic++ -show`)

g++ -I/usr/lib/x86_64-linux-gnu/openmpi/include -I/usr/lib/x86_64-linux-gnu/openmpi/include/openmpi -L/usr/lib/x86_64-linux-gnu/openmpi/lib -lmpi_cxx -lmpi

Collectives



Broadcast

Before

| procid | 1 | 2 | 3 | 4 |
|--------|---|---|---|---|
| x | | | | |

After

| procid | 1 | 2 | 3 | 4 |
|--------|-----|-----|-----|-----|
| x | x | x | x | x |

- ▶ Lower bound complexity for n bytes with p processes ?

Gather

Before

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | | | |
| | | x_2 | | |
| | | | x_3 | |
| | | | | x_4 |

After

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | | | |
| | | x_2 | | |
| | | | x_3 | |
| | | | | x_4 |

- Lower bound complexity with p processes if x_i has length n/p bytes ?

Reduce

Before

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | x_2 | x_3 | x_4 |

After

| procid | 1 | 2 | 3 | 4 |
|-------------------------|---|---|---|---|
| $x_1 + x_2 + x_3 + x_4$ | | | | |

- Lower bound complexity for n bytes with p processes and arithmetic complexity γ ?

All gather

Before

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | | | |
| | | x_2 | | |
| | | | x_3 | |
| | | | | x_4 |

After MPI_Allgather

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | x_1 | x_1 | x_1 |
| | x_2 | x_2 | x_2 | x_2 |
| | | x_3 | x_3 | x_3 |
| | | | x_4 | x_4 |

► Can MPI_Allgather be implemented by combining existing collectives ?

► Would it be more efficient to have a specialized implementation instead of combining existing collectives ?

Reduce scatter

Before

| procid | 1 | 2 | 3 | 4 |
|--------|-----------|-----------|-----------|-----------|
| | $x_{1,1}$ | $x_{1,2}$ | $x_{1,3}$ | $x_{1,4}$ |
| | $x_{2,1}$ | $x_{2,2}$ | $x_{2,3}$ | $x_{2,4}$ |
| | $x_{3,1}$ | $x_{3,2}$ | $x_{3,3}$ | $x_{3,4}$ |
| | $x_{4,1}$ | $x_{4,2}$ | $x_{4,3}$ | $x_{4,4}$ |

After MPI_Reduce_scatter

| procid | 1 | 2 | 3 | 4 |
|--------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| | $x_{1,1} + \dots + x_{1,4}$ | | | |
| | | $x_{2,1} + \dots + x_{2,4}$ | | |
| | | | $x_{3,1} + \dots + x_{3,4}$ | |
| | | | | $x_{4,1} + \dots + x_{4,4}$ |

► Can MPI_Reduce_scatter be implemented by combining existing collectives ?

► Would it be more efficient to have a specialized implementation instead of combining existing collectives ?

Allreduce

Before

| procid | 1 | 2 | 3 | 4 |
|--------|-------|-------|-------|-------|
| | x_1 | x_2 | x_3 | x_4 |

After MPI_Reduce_scatter

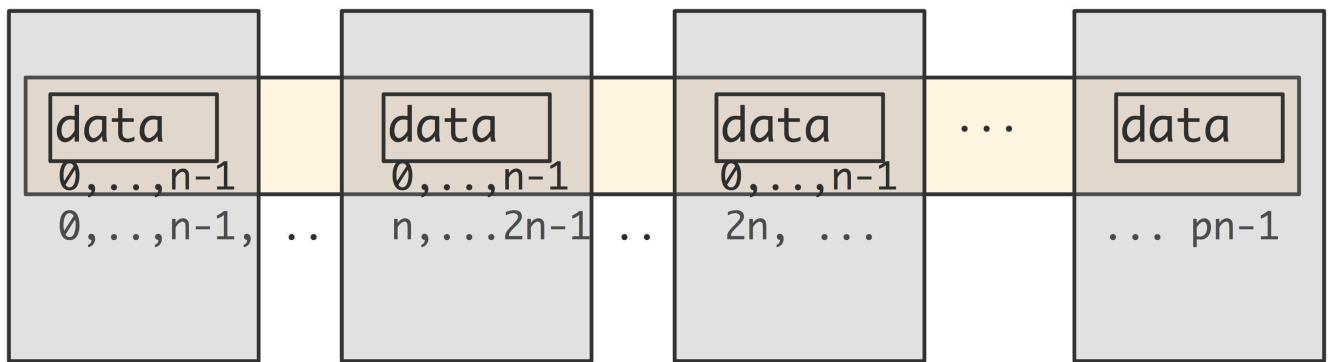
| procid | 1 | 2 | 3 | 4 |
|--------|---------------------|---------------------|---------------------|---------------------|
| | $x_1 + \dots + x_4$ |

► Can MPI_Allreduce be implemented by combining existing collectives ?

Distributed sum

Distributed vector

```
int n;  
double data[n];
```



► How to collect the partial sums ?

Let's try it

```
for (int i = stride * procid; i < last; i++)  
    local_sum += vec[i];  
float total = 0;  
MPI_Reduce(&local_sum, &total, 1, MPI_FLOAT, MPI_SUM, 0, comm);  
if (verbose >= 1)  
    fprintf(stderr, "proc id : %d / %d : [local = %f] : [total = %f]\n", procid,  
nprocs, local_sum, total);
```

Compiling : `mpicc -O3 /tmp/jl_bAd1mY/main.c -o /tmp/jl_bAd1mY/bin`

Running : `mpiexec -n 2 /tmp/jl_bAd1mY/bin`

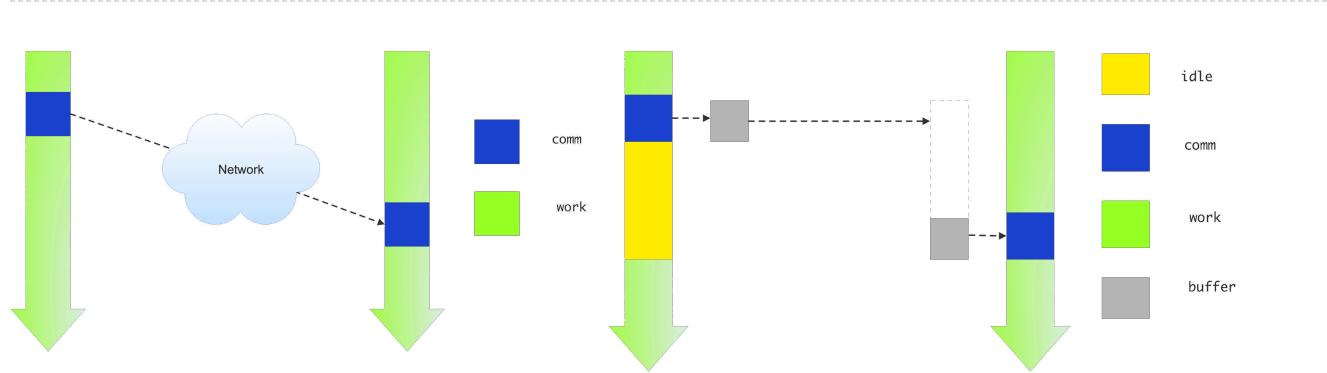
```
> proc id : 0 / 2 0:3  
proc id : 1 / 2 4:8  
proc id : 1 / 2 : [local = 35.000000] : [total = 0.000000]  
proc id : 0 / 2 : [local = 10.000000] : [total = 45.000000]
```

num_processes = 2

► Why is it the first process that gets the sum ?

Point-to-point

Blocking communication



Blocking send/received with `MPI_Send` and `MPI_Recv`.

The network cannot buffer the whole message (unless it is short). The sender need to wait for the receiver to be ready and then transfer its copy of the data.

Example

```
int tag = 0;
for(int size = 1; size <= (1<<20); size <= 1){
    char* buf = malloc(size);
    if (procid == 0) {
        MPI_Send(buf, size, MPI_CHAR, procid + 1, tag++, comm);
    }
    else {
        double tic = MPI_Wtime();
        MPI_Recv(buf, size, MPI_CHAR, procid - 1, tag++, comm, MPI_STATUS_IGNORE);
        double toc = MPI_Wtime();
        printf("[%d] I have received %d B in %f sec\n", procid, size, (toc-tic));
    }
}
```

```
[1] I have received 1 B in 0.000032 sec
[1] I have received 2 B in 0.000001 sec
[1] I have received 4 B in 0.000000 sec
[1] I have received 8 B in 0.000000 sec
[1] I have received 16 B in 0.000000 sec
[1] I have received 32 B in 0.000000 sec
[1] I have received 64 B in 0.000000 sec
[1] I have received 128 B in 0.000000 sec
[1] I have received 256 B in 0.000000 sec
[1] I have received 512 B in 0.000000 sec
[1] I have received 1024 B in 0.000001 sec
[1] I have received 2048 B in 0.000001 sec
[1] I have received 4096 B in 0.000014 sec
[1] I have received 8192 B in 0.000011 sec
[1] I have received 16384 B in 0.000012 sec
[1] I have received 32768 B in 0.000025 sec
[1] I have received 65536 B in 0.000053 sec
[1] I have received 131072 B in 0.000082 sec
[1] I have received 262144 B in 0.000148 sec
[1] I have received 524288 B in 0.000285 sec
[1] I have received 1048576 B in 0.000582 sec
```

► Is this timing bandwidth accurately ?

Eager vs rendezvous protocol

There are two protocols:

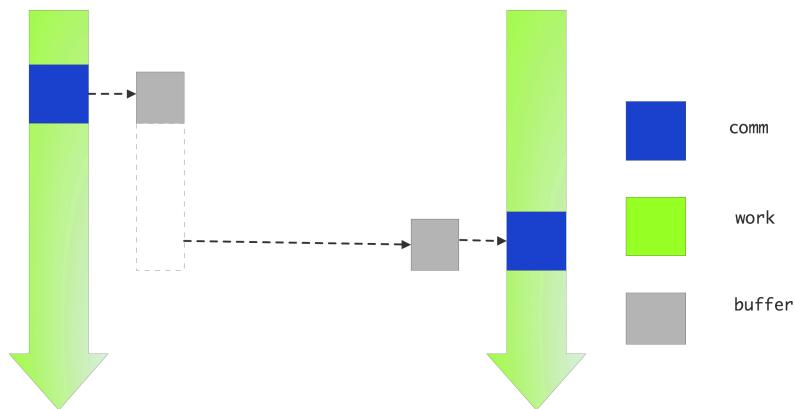
- Rendezvous protocol

1. the sender sends a header;
 2. the receiver returns a ‘ready-to-send’ message;
 3. the sender sends the actual data.
- Eager protocol the message is buffered so MPI_Send can return eagerly, before the receiver is even ready

Eager protocol is used if the data size is smaller than the *eager limit*. To force the rendezvous protocol, use MPI_Ssend.

See [Eij17; Section 4.1.4.2]

Nonblocking communication



MPI_Isend and MPI_Irecv where I stands for immediate or incomplete. MPI_Wait can be used to wait for the send and receive to finish.

Example

```
for(int size = 1; size <= (1<<20); size <= 1){
    char* buf = malloc(size);
    if (procid == 0) {
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Send(buf, size, MPI_CHAR, procid + 1, 0, comm);
    }
    else {
        MPI_Irecv(buf, size, MPI_CHAR, procid - 1, 0, MPI_COMM_WORLD, &rqst);
        MPI_Barrier(MPI_COMM_WORLD);
        double tic = MPI_Wtime();
        MPI_Wait(&rqst, MPI_STATUS_IGNORE);
        double toc = MPI_Wtime();
        printf("[%d] I have received %d B in %f sec\n", procid, size, (toc-tic));
    }
}
```

```
[1] I have received 1 B in 0.000002 sec
[1] I have received 2 B in 0.000001 sec
[1] I have received 4 B in 0.000001 sec
[1] I have received 8 B in 0.000003 sec
[1] I have received 16 B in 0.000001 sec
[1] I have received 32 B in 0.000001 sec
[1] I have received 64 B in 0.000001 sec
[1] I have received 128 B in 0.000008 sec
[1] I have received 256 B in 0.000000 sec
[1] I have received 512 B in 0.000004 sec
[1] I have received 1024 B in 0.000001 sec
[1] I have received 2048 B in 0.000001 sec
[1] I have received 4096 B in 0.000020 sec
[1] I have received 8192 B in 0.000017 sec
[1] I have received 16384 B in 0.000015 sec
[1] I have received 32768 B in 0.000024 sec
[1] I have received 65536 B in 0.000049 sec
[1] I have received 131072 B in 0.000086 sec
[1] I have received 262144 B in 0.000162 sec
[1] I have received 524288 B in 0.000289 sec
[1] I have received 1048576 B in 0.000594 sec
```

Consortium des Équipements de Calcul Intensif (CÉCI)

- Follow README instructions to create an account and setup your computer
 - Don't wait the last minute, if you get into trouble it's easier to get this setup before you actually need it
- Select CÉCI cluster from the list + manneback for GPU. You only have access to Tier-2 clusters.
This sadly leaves out:
 - Tier-1 clusters such as Lucia
 - Tier-0 cluster such as LUMI from  EuroHPC Joint Undertaking
- Connect with SSH using ssh lemaître4 or ssh manneback.



Lmod

```
[local computer]$ ssh lemaître4
```



```
▶ [blegat@lm4-f001 ~]$ module list
```

```
[blegat@lm4-f001 ~]$ mpicc  
-bash: mpicc: command not found
```



```
[blegat@lm4-f001 ~]$ module load gompi/2023a
```

```
[blegat@lm4-f001 ~]$ mpicc  
gcc: fatal error: no input files  
compilation terminated.
```

```
▶ [blegat@lm4-f001 ~]$ module list
```

Tip

```
▶ Use module spider to see which version are available
```

Launching a job

```
[laptop]$ ssh lemaître4  
[blegat@lm4-f001 ~]$ cd LINMA2710/examples  
[blegat@lm4-f001 examples]$ mpicc procname.c  
-bash: mpicc: command not found
```

```
▶ How to fix it ?
```

Slurm

- `srun` : Synchronous (blocked) job

```
[blegat@lm4-f001 ~]$ srun --time=1 pwd  
srun: job 3491072 queued and waiting for resources  
srun: job 3491072 has been allocated resources  
/home/users/b/l/blegat
```

- `$ sbatch submit.sh` : Asynchronous job, get status with
- `$ squeue --me`



See [CÉCI documentation](#)

- More details on the [README](#)

Profiling with NVIDIA Nsight Systems

- NVIDIA Nsight Systems  can profile CUDA code but also MPI
- Available on manneback after loading CUDA with [Lmod](#)

```
[laptop]$ ssh manneback
[blegat@mbackf1 ~]$ nsys
-bash: nsys: command not found
[blegat@mbackf1 ~]$ ml CUDA
[blegat@mbackf1 ~]$ nsys
```

Topology

- Specializing on topology is important for communication libraries like MPI/NCCL. For instance, Deepseek-V3 by-passed NCCL and used PTX directly to hardcode how the hardware should be used.
- Specified in [Slurm's topology.conf file](#).
- Source : [Eij10; Section 2.7]

Graph diameter

- Consider graph G with nodes v corresponding to computer nodes or switches.
- There is an edge $(u, v) \in E$ if there is an ethernet cable **directly** connecting u and v .
- $e \in E$ are ethernet cables of bandwidth w_e
- Distance (unweighted) from node $u \in V$ to node $v \in V$ is $d(G, u, v)$
 - Does not depend on bandwidth w_e of edges of the path

Def: Graph diameter

Graph diameter is $d(G) := \max_{u, v \in V} d(G, u, v)$

Bisection bandwidth

Bandwidth $\text{bw}(u, v)$ is the bandwidth of the cable if $(u, v) \in E$ or 0 otherwise. Given $S, T \subseteq V$,

$$\begin{array}{ll}\text{Width} & w(S, T) = |\{(u, v) \in E \mid u \in S, v \in T\}| \\ \text{Bandwidth} & \text{bw}(S, T) = \sum_{u \in S, v \notin S} w(u, v)\end{array}$$

The *bisection width* is:

The *bisection bandwidth* is:

$$\min_{S \subset V: \lfloor |V|/2 \rfloor \leq |S| \leq \lceil |V|/2 \rceil} w(S, V \setminus S) \quad \min_{S \subset V: \lfloor |V|/2 \rfloor \leq |S| \leq \lceil |V|/2 \rceil} \text{bw}(S, V \setminus S)$$

- Worst case pairwise communication of two groups S and $V \setminus S$ of *almost* (± 1) equal size.
- NP-hard to compute for general graphs

► What are the differences with Min-Cut ?

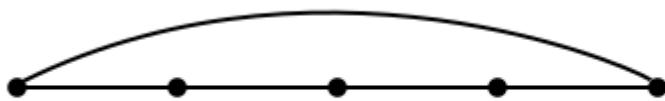
Linear array



► What is the graph diameter ?

► What is the bisection width ?

Rings

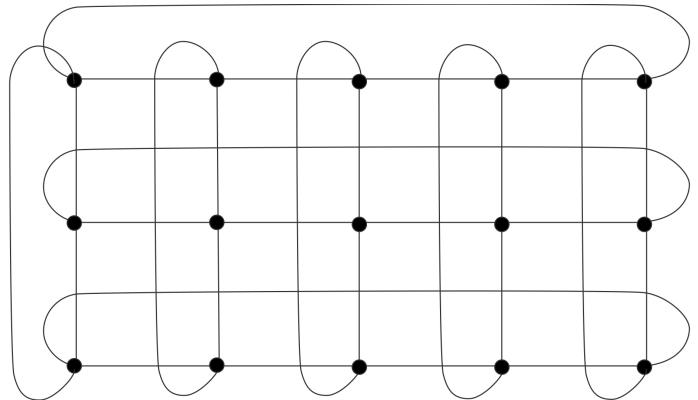
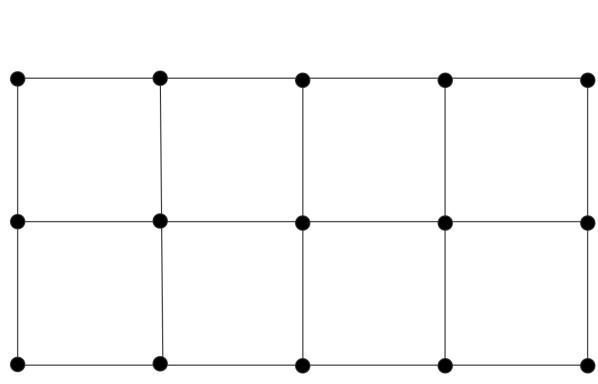


► What is the graph diameter ?

► What is the bisection width ?

[Eij10; Section 2.7.3]

Multidimensional array and torus



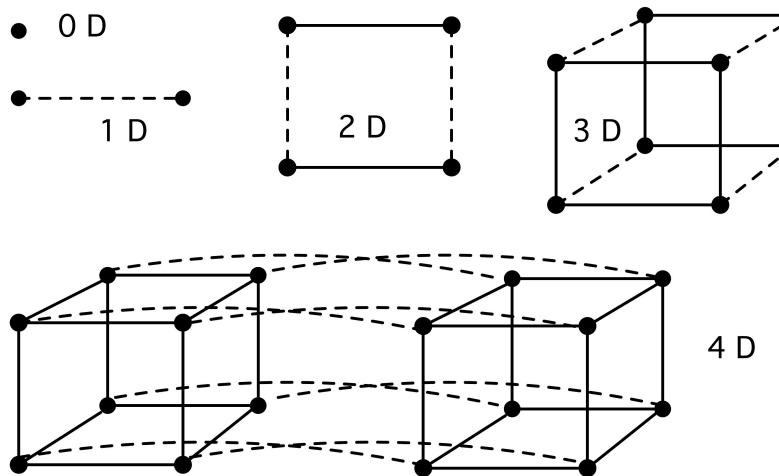
► What is the graph diameter of a $n \times n$ 2D array ?

► What is the bisection width of a $n \times n$ 2D array ?

[Eij10; Section 2.7.4]

Hypercube

Special case of multidimensional array



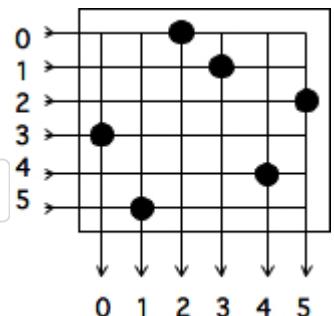
- ▶ How to order the nodes so that consecutive nodes in the order are adjacent in the graph ?

[Eij10; Section 2.7.5]

Crossbar

- Each dot is a node
- Each intersection is a switch

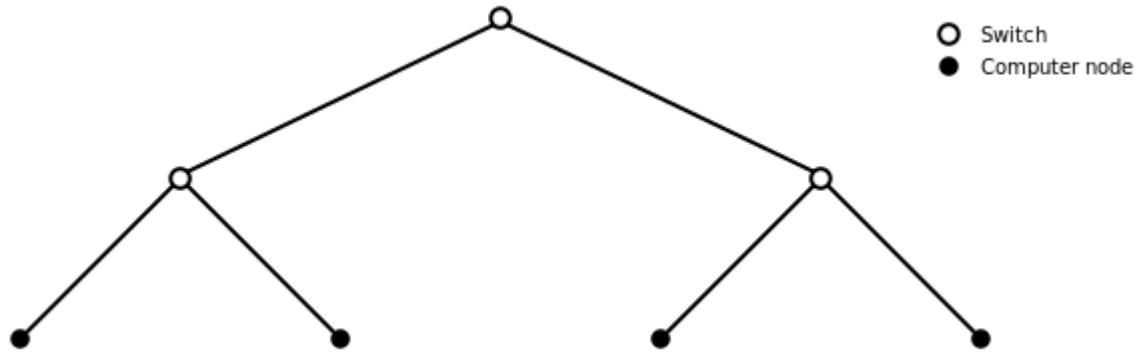
- ▶ What is the underlying graph between nodes



- ▶ What are the number of switches, edges, graph diameter and bisection width for n computer nodes ?

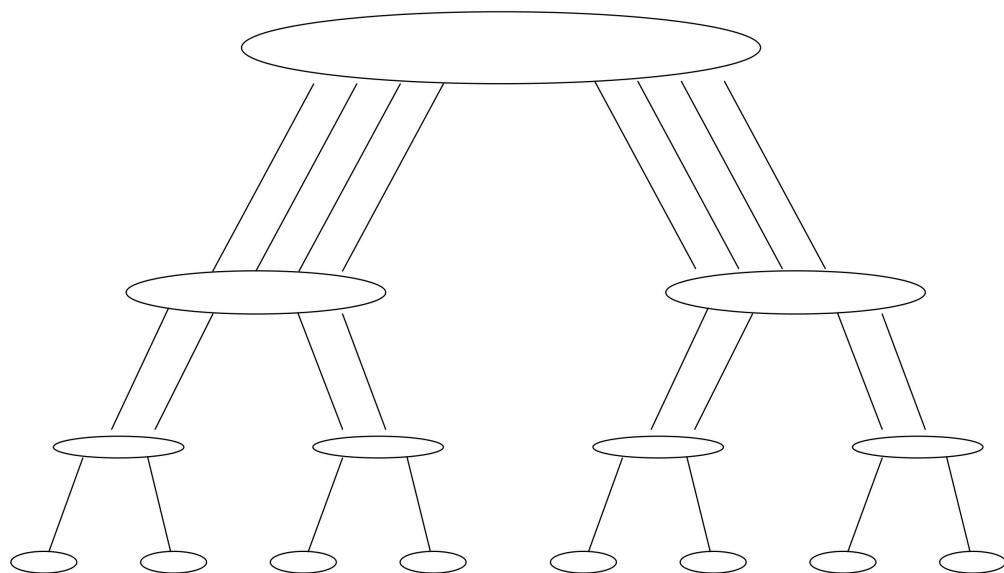
[Eij10; Section 2.7.6.1]

Tree



► What is the diameter and bisection width of n computer nodes ?

Fat-tree



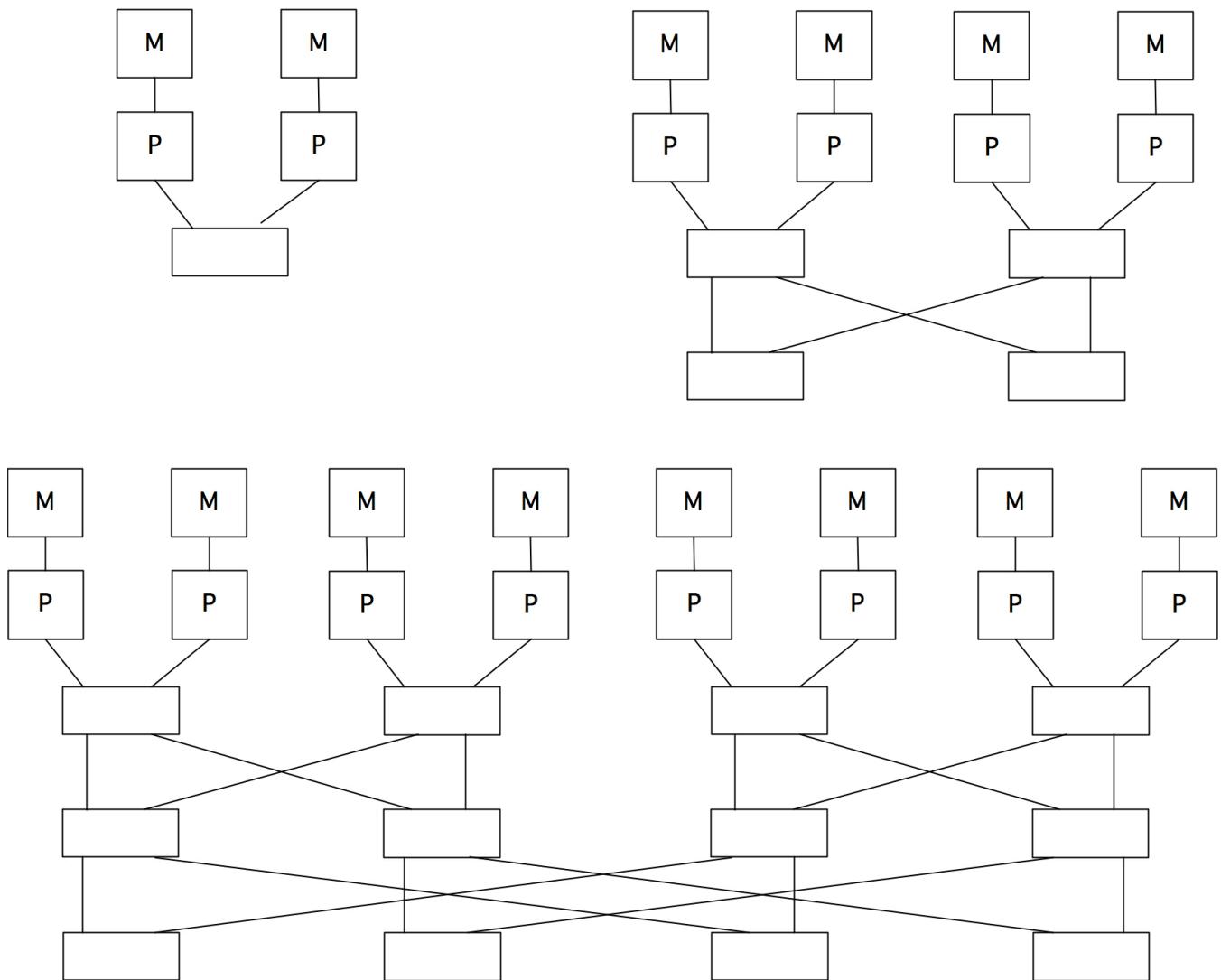
blocking factor : Ratio between upper links and lower links. Ratio is 1 for fat-tree to prevent bottlenecks if all nodes start communicating.

► What is the number of edges ? What is the bisection width ?

[Eij10; Section 2.7.6.3]

Butterfly

Fat-tree need large switches, alternative is butterfly network:



► What is the number of edges ? What is the bisection width ?

[Eij10; Section 2.7.6.2]

From [Eij10; Figure 2.27]

From [Eij10; Figure 2.30]



img1 (generic function with 1 method)