Energy efficiency in Data Reduction for Imaging in a Radio Astronomy pipeline

Giovanni Lacopo, PhD in Physics

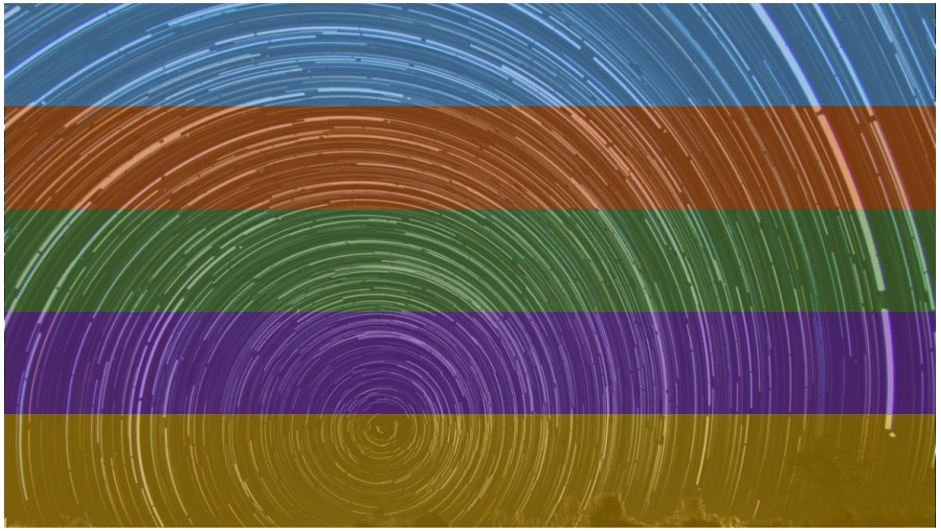
Supervisors: Luca Tornatore, Giuliano Taffoni, Stefano Borgani

Radio-interferometry data processing

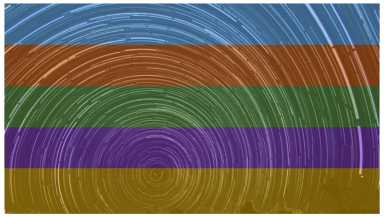
- The code (see C. Gheller’s talk for details) is a stage in a Radio Imaging pipeline;
- Each MPI task has a fraction of a timeline series, that contains the observed sources in chronological order;
  as always, the sources apparently move in the sky due to the Earth rotation.

Each MPI task has a fraction of the time-ordered log of the observations
(kind of it has some of the tracks in the picture)

What we need is
to translate from
a **time-domain**
to a **sky-domain**



What we need is to translate from a **time-domain** to a **sky-domain**



1) every MPI task stacks all of its data that lie in a given sector
2) all the tasks perform a reduce operation having as target the task that owns the sector

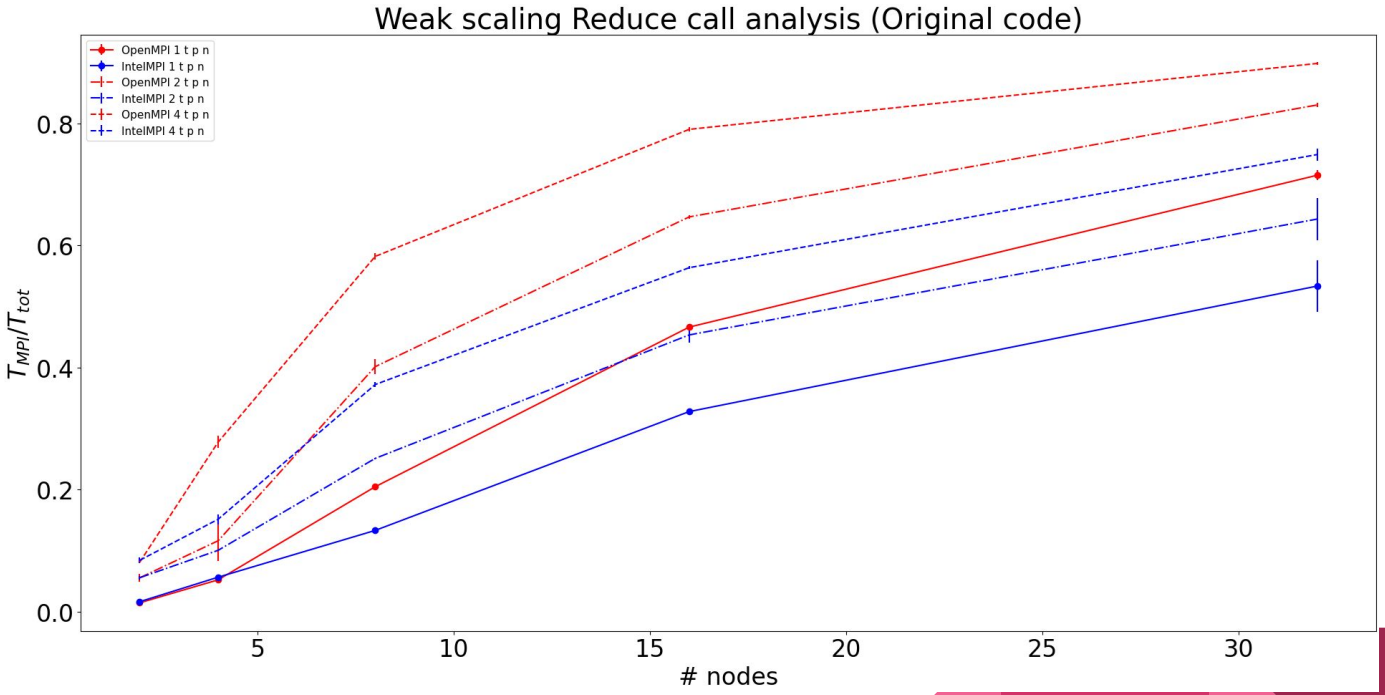
Obviously, there are as many sectors as MPI tasks.

⇒ There are as many `MPI_Reduce()` calls than MPI tasks

The MPI_Reduce is the bottleneck

We find the the **MPI_Reduce operation takes a very significant amount of time**, which grows fast with the number of MPI tasks

Up to 80-90 % for big problems
(which are the targets)

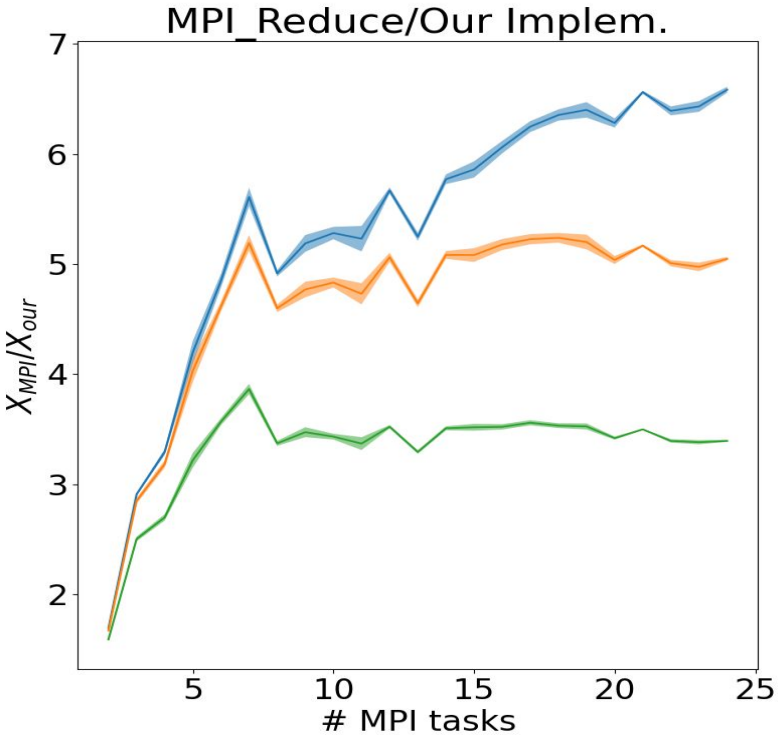


Reduce in Shared memory

We keep the reduce in-node under control **by implementing a by-hand reduce in shared memory by exploiting the NUMA awareness** of the architectures (see L.Tornatore's talk)

- New in-node communicator in which each task knows which are its siblings;
- Ring algorithm in which each task sums 1/P of the data.

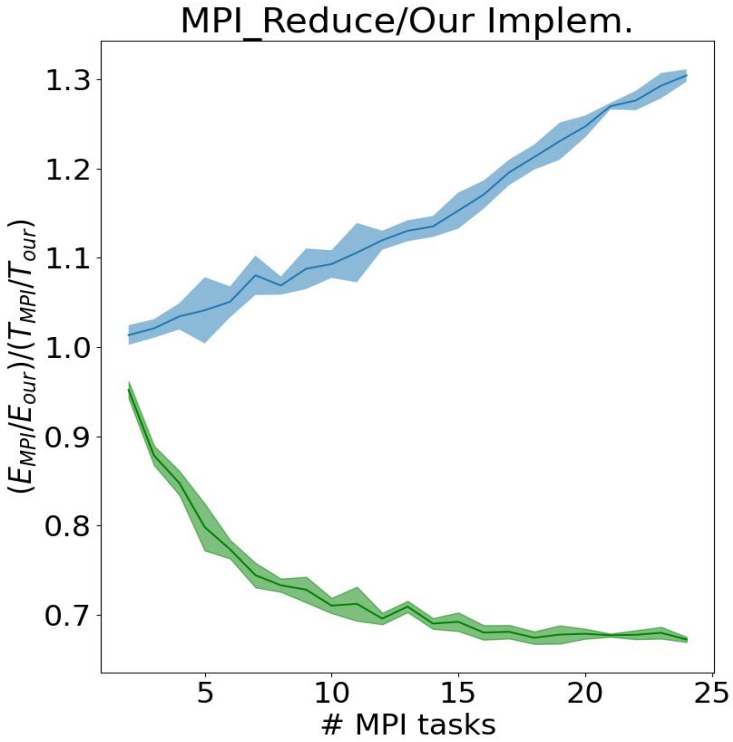
Our reduce vs OpenMPI (paper in prep.)



Key message

- Our reduce is **2 to 5 times faster on a node**
- Our reduce requires **less energy**
  - 2 to 7 times less CPU energy
  - ~3 times less memory energy

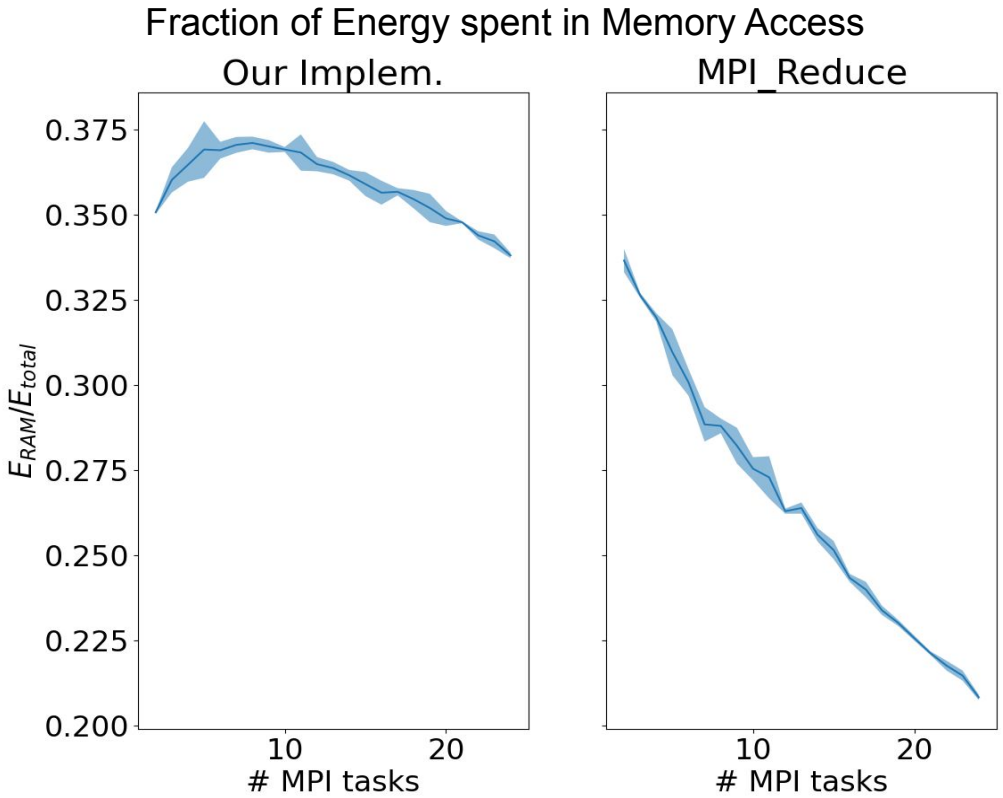
Our reduce vs OpenMPI (paper in prep.)



Key message

- The energy advantage is NOT simply due to the smaller time-to-solution: **there is a specific algorithmic imprint**
- we consume slightly more energy in DRAM access
  (our algorithm is more memory intense)

Memory accesses

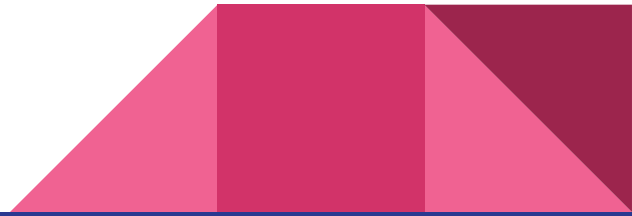


Key message

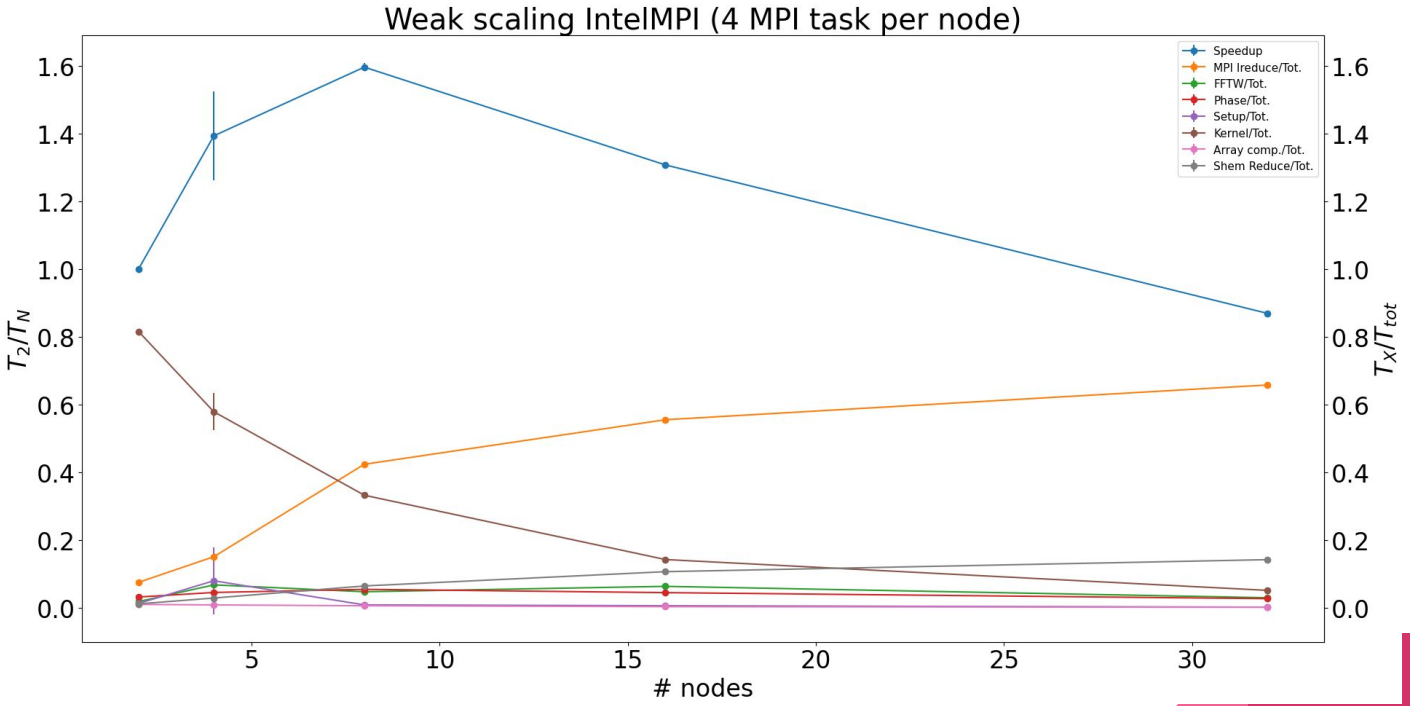
Relatively, our reduce is significantly more memory-intense

What about the network?

Is necessary to implement a reduce operation also among the nodes or can we rely on the standard implementations given by MPI?



The bottleneck is still the MPI_reduce among nodes



Reduce operation on GPUs?

Pros:

- Relatively simple to use thanks to **NCCL**;
- RDMA in-node and inter-node for GPU-GPU communication;
- Portable on AMD GPUs.

Cons:

- Requirement of specific hardware components (Nvlink, Infinity Fabric) to achieve the best performance.

Credits: https://developer.nvidia.com/nccl

Energy profiler

Python code which recognizes the specific architecture and profiles the power consumption of the codes and their different parts:

- Intel CPUs 

- AMD CPUs 

- Nvidia GPUs 

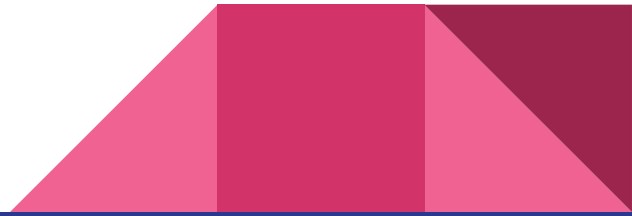
- AMD GPUs 

- ARM CPUs Not yet :(

Credits: Niccolò Tosato

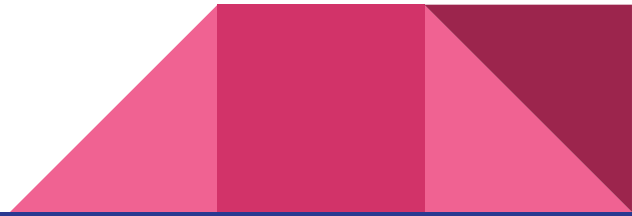
Conclusions

- In our Radio Imaging code the reduce operation is the true bottleneck, taking up to 70-80% of runtime;
- To face this issue, we have written a by-hand reduce operation which is faster by a factor of ~5-6 and more energy efficient by a factor of ~6-7 inside each computing node;
- This means that there are cheap and expensive CPU instructions. Writing an energy efficient code does not simply mean making it faster.

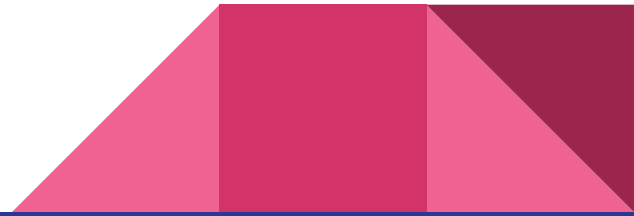


Future perspectives

- Implementation of a by-hand reduce inter-node to handle the bottleneck of communication;
- Benchmarking to compare our reduce operation with NCCL reduce on GPUs (already implemented in the code);
- Complete profiling of power consumption of single code functions with our energy profiler.

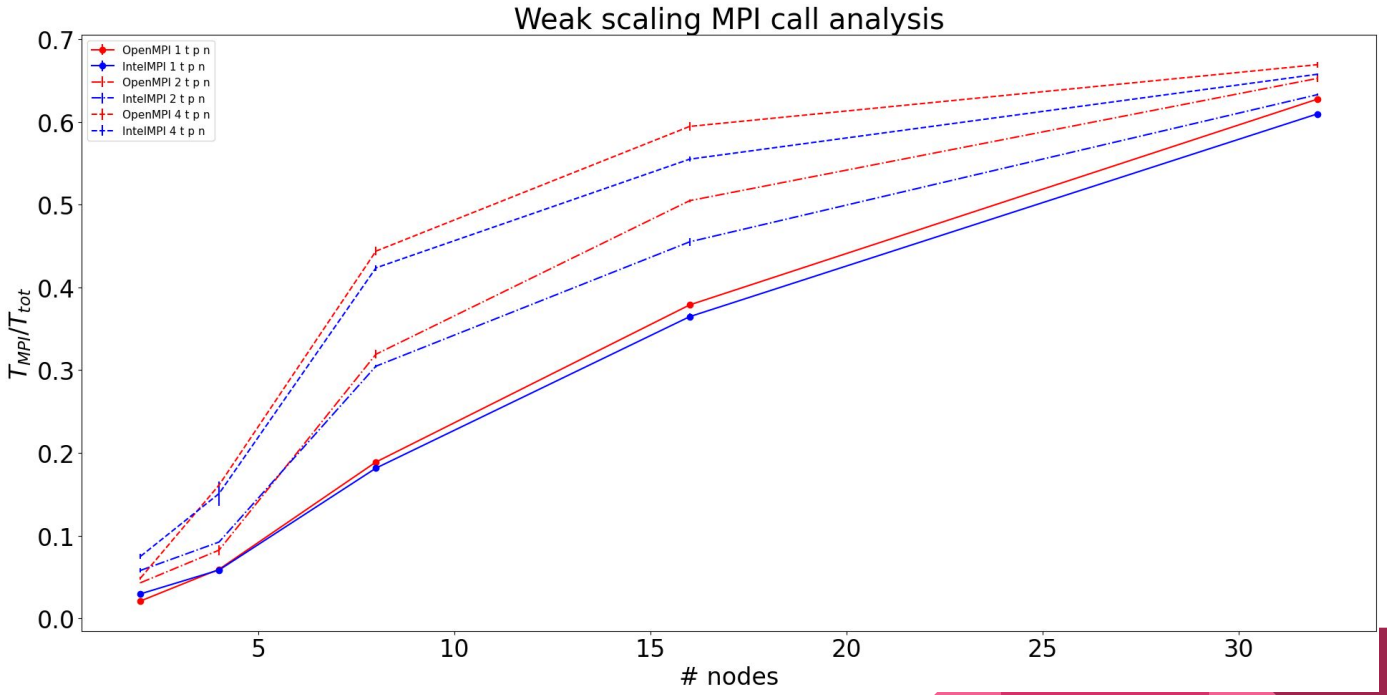


MPI_Ireduce

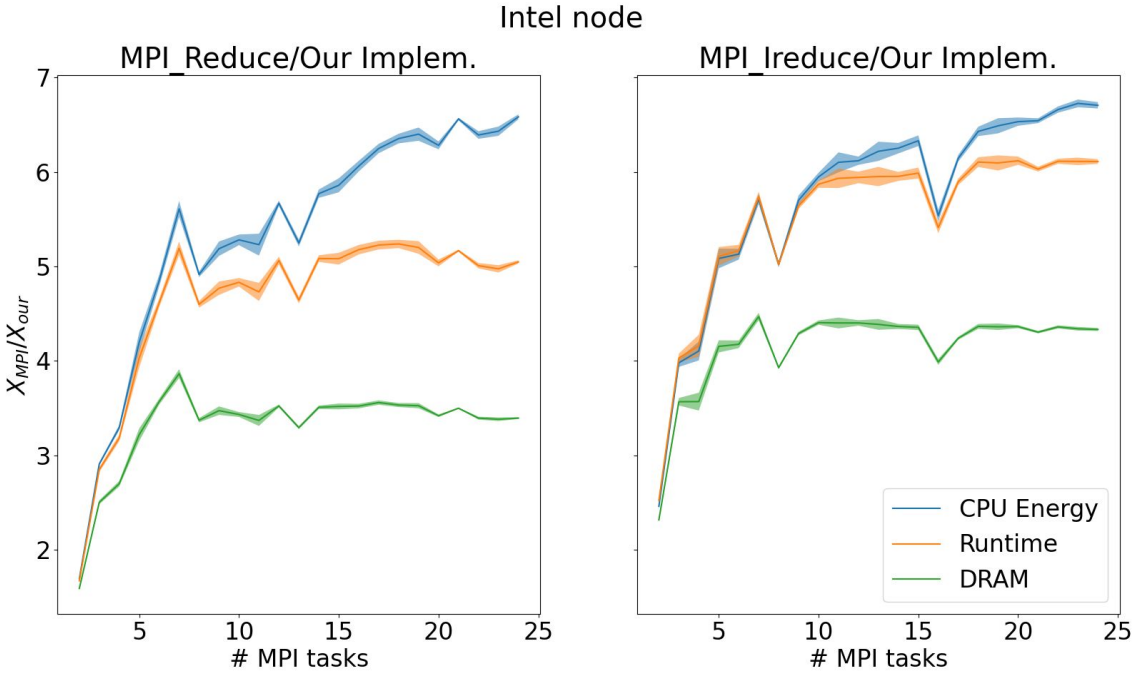


The MPI_Reduce is the bottleneck

We find the the MPI_Reduce operation takes a very significant amount of time, which grows fast with the number of MPI tasks



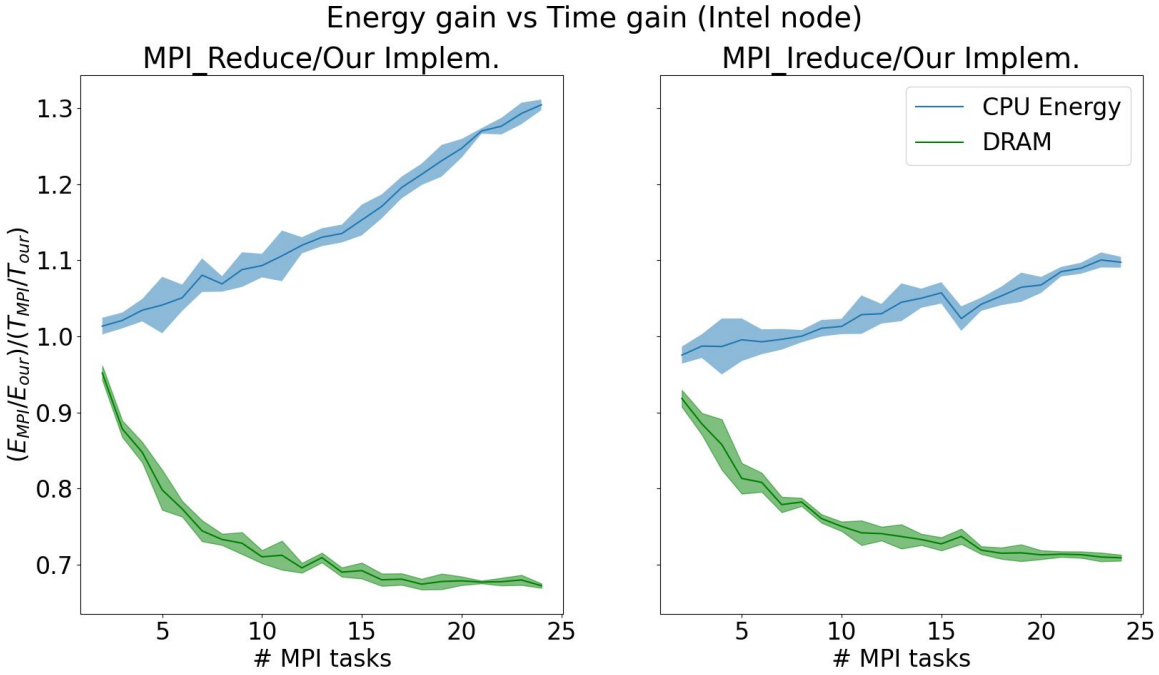
Our reduce vs OpenMPI (paper in prep.)



Key message

- Our reduce is **2 to 5 times faster on a node**
- Our reduce requires **less energy**
  - 2 to 7 times less CPU energy
  - ~3 times less memory energy

Our reduce vs OpenMPI (paper in prep.)



Key message

- The energy advantage is NOT simply due to the smaller time-to-solution
- we consume slightly more energy in DRAM access (our algorithm is more memory intense)