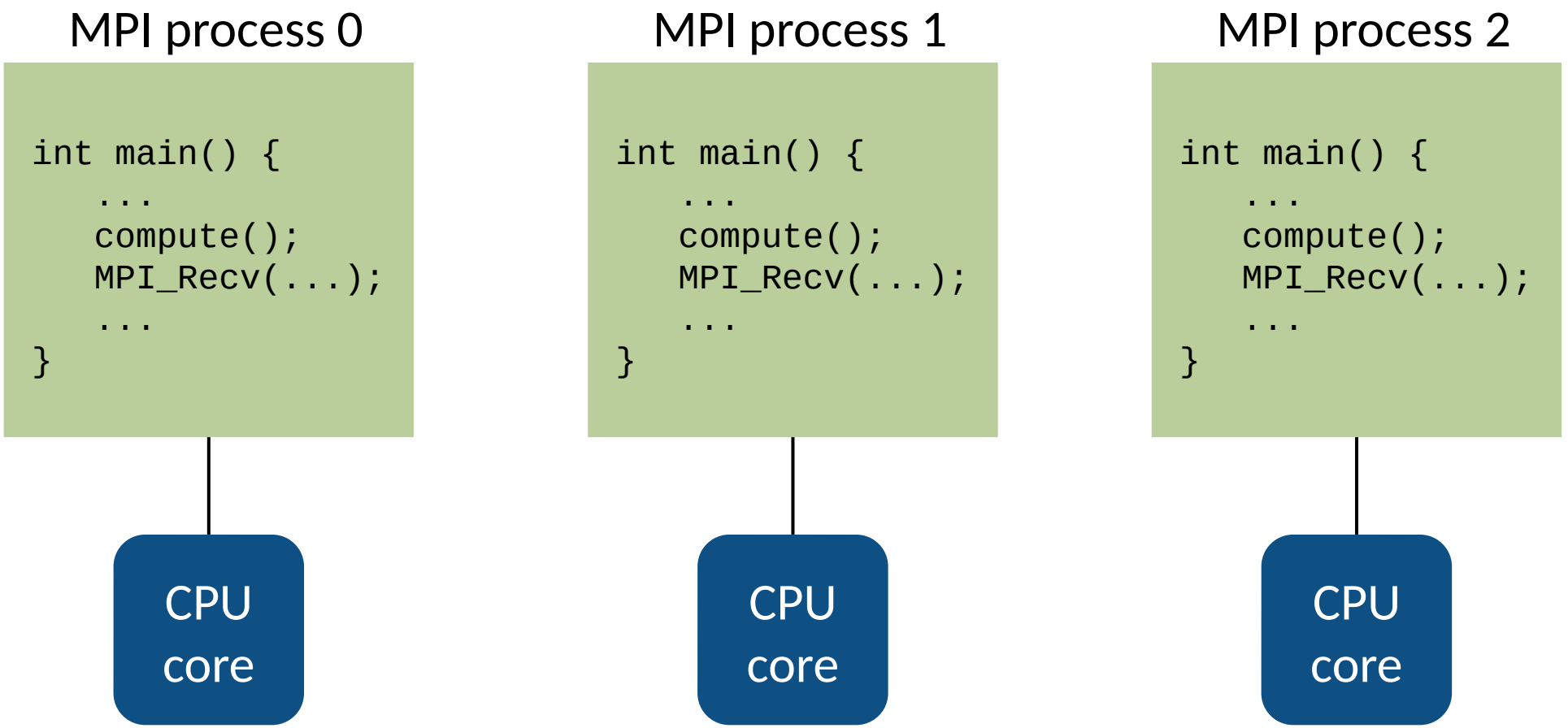
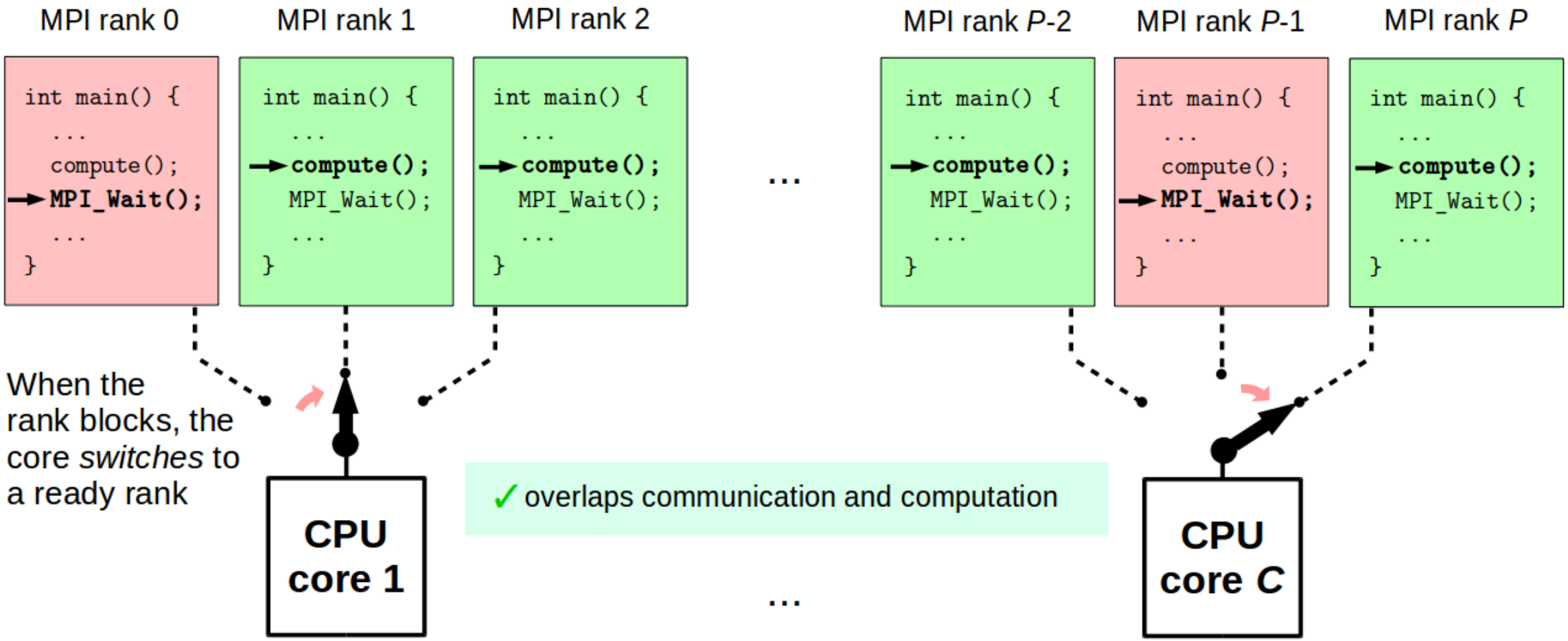
ALEXANDR NIGAY, LUKAS MOSIMANN, TIMO SCHNEIDER, TORSTEN HOEFLER

Communication and Timing Issues with MPI Virtualization

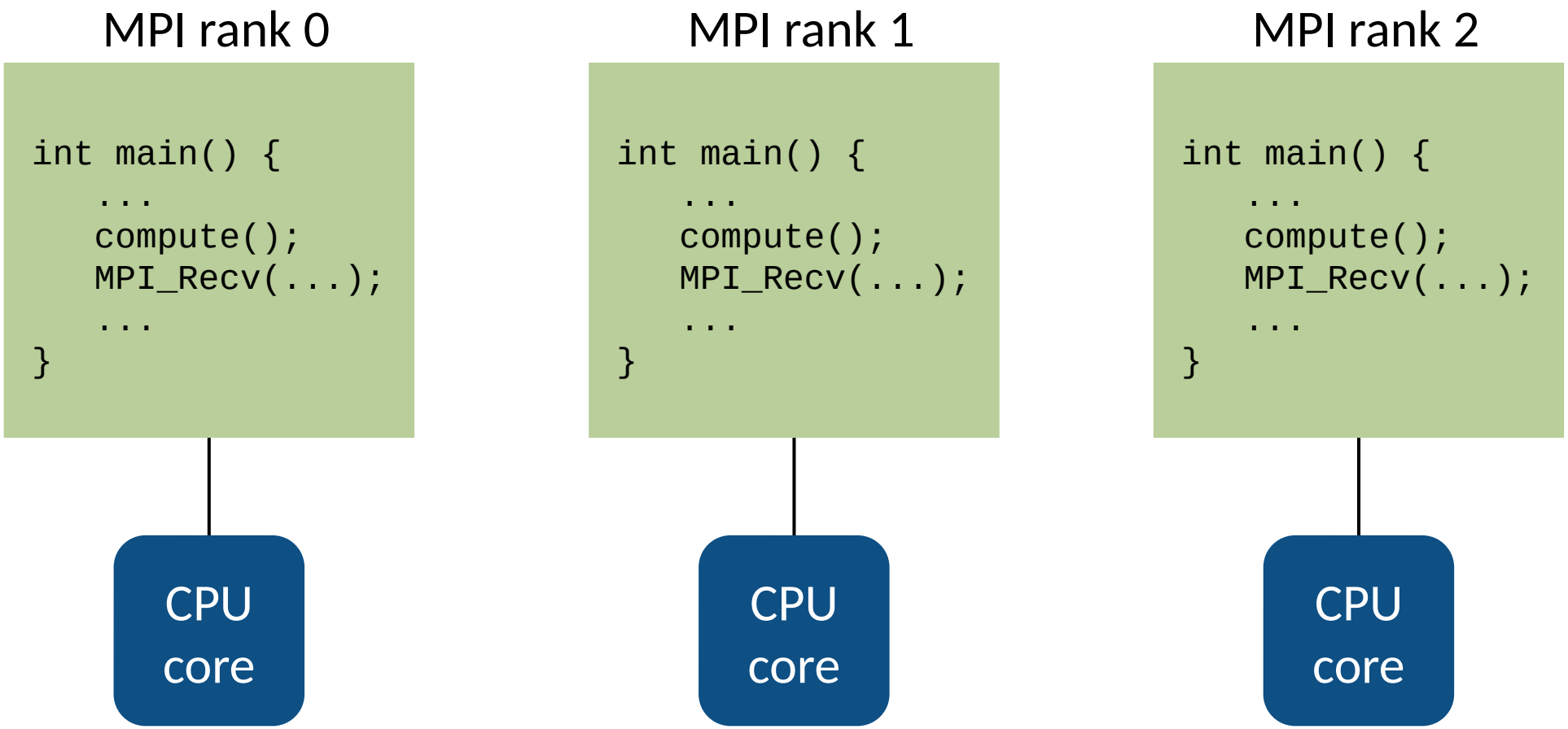
What is MPI Virtualization?



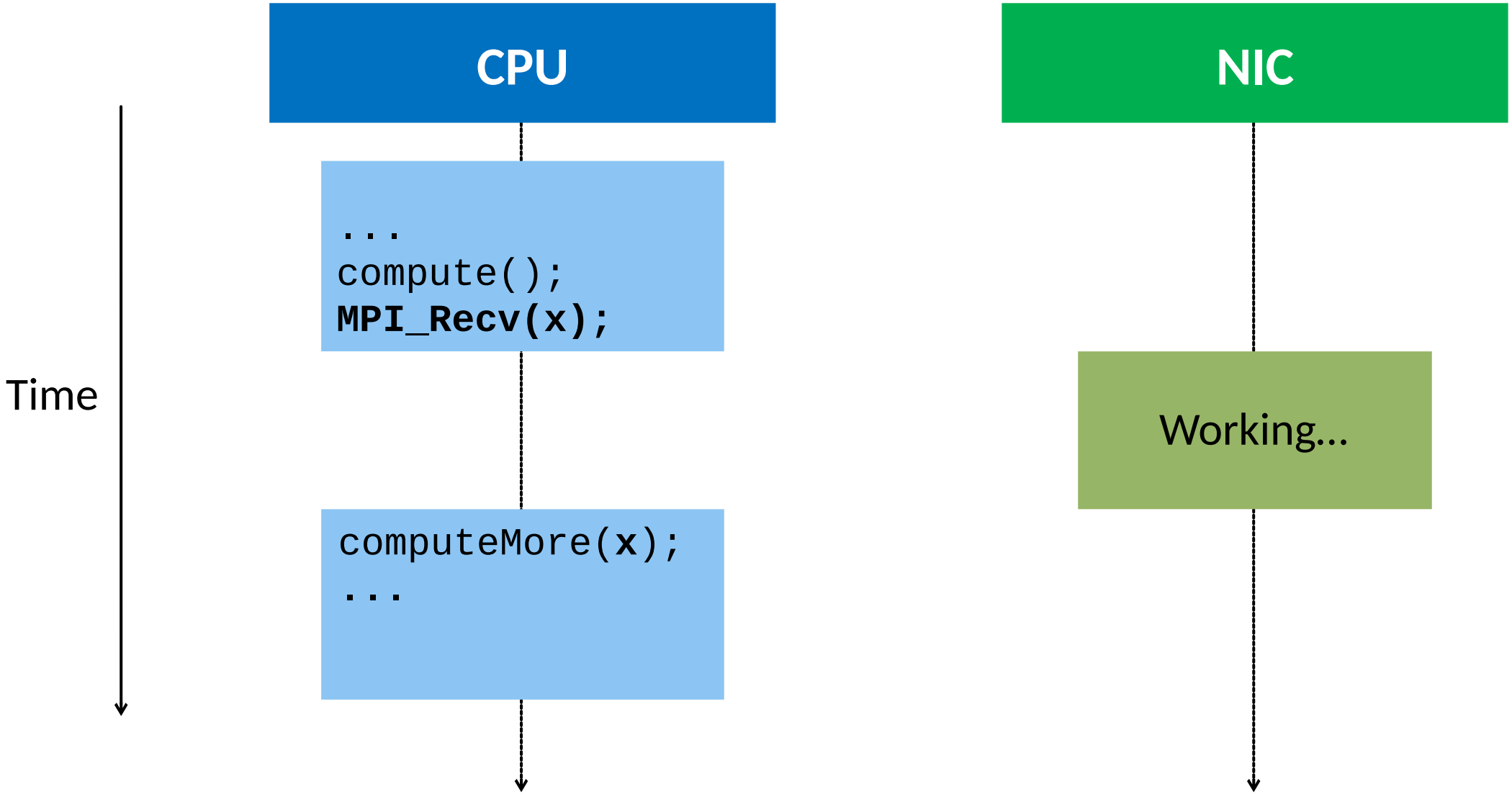
What is MPI Virtualization?



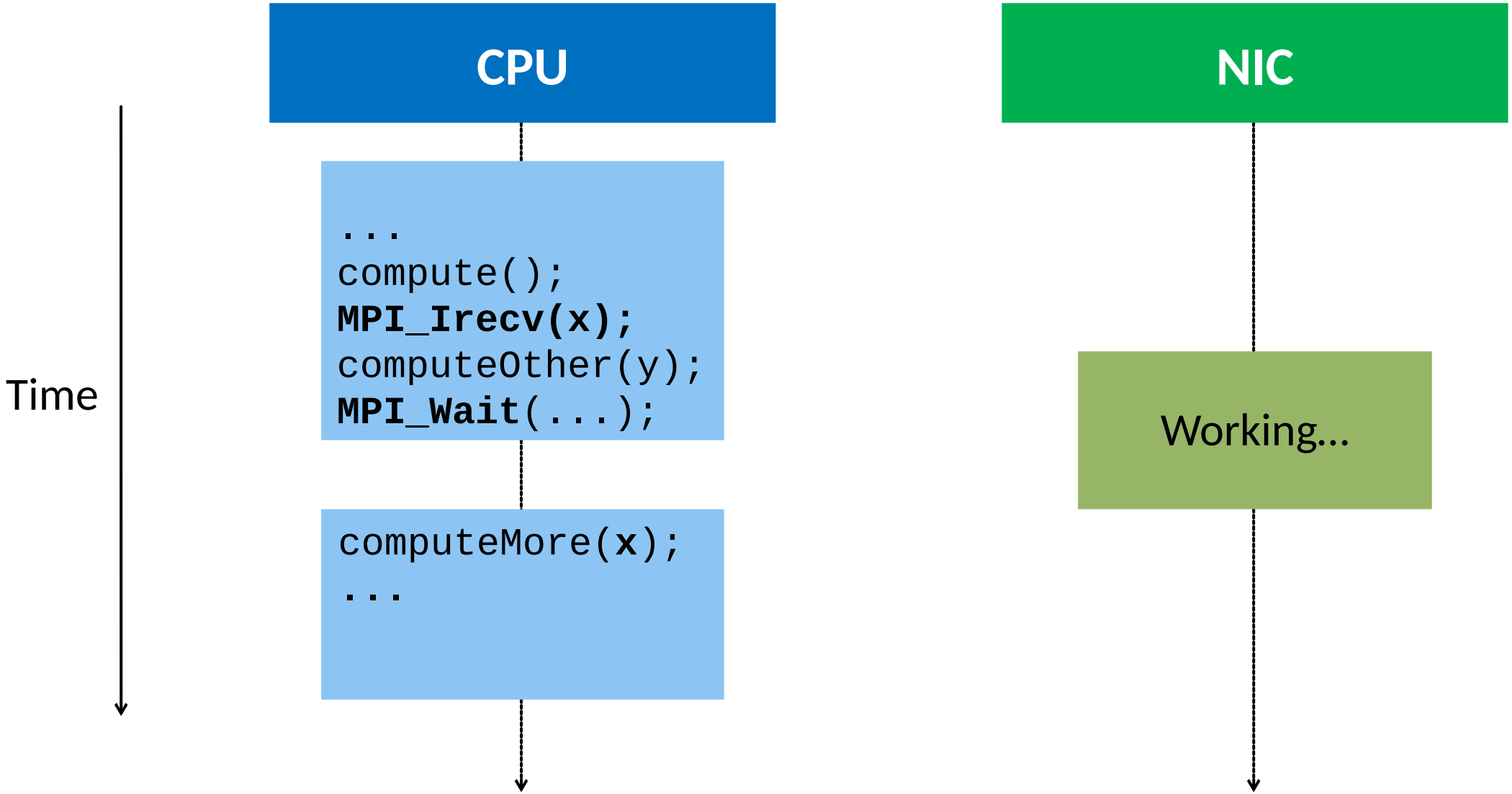
Why MPI Virtualization is a good Idea: Load Balancing



Why MPI Virtualization is a Good Idea: Communication Overlap

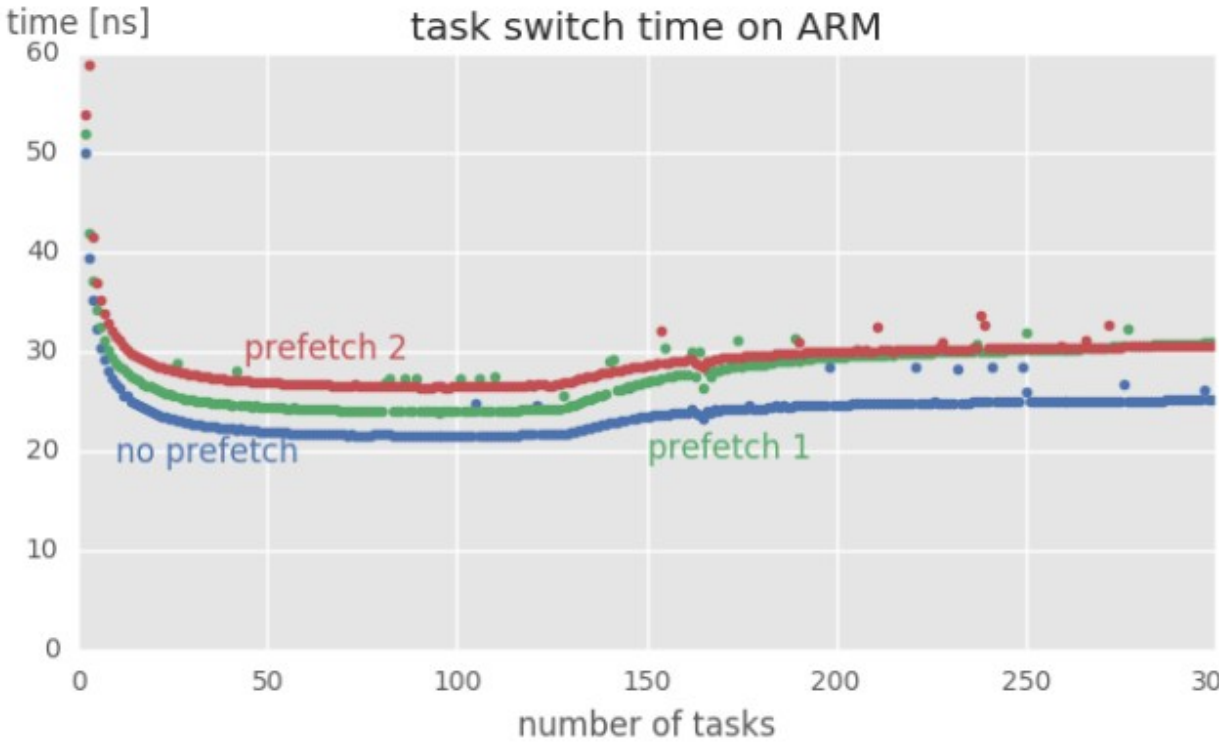


Why MPI Virtualization is a Good Idea: Communication Overlap

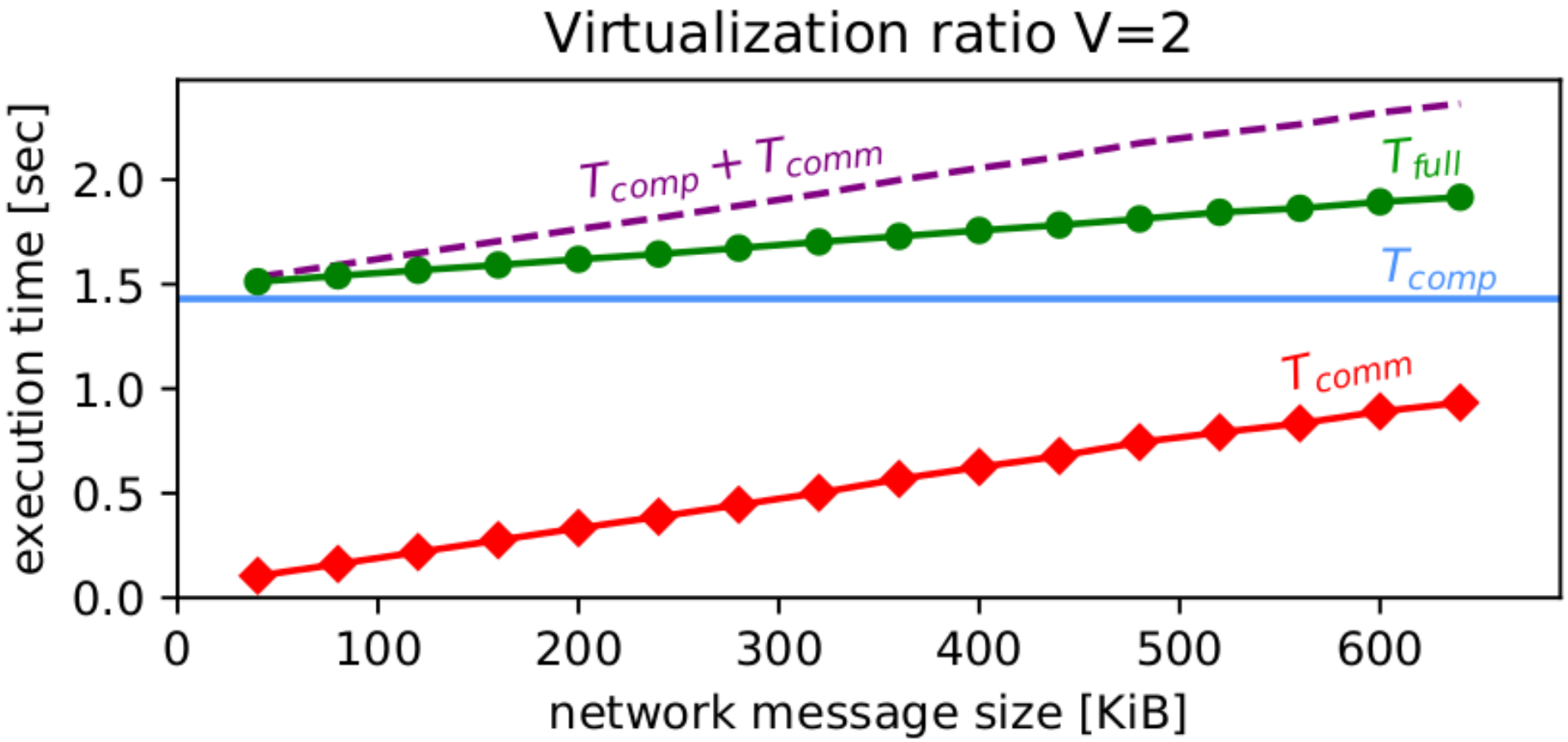


MPI Virtualization Basics

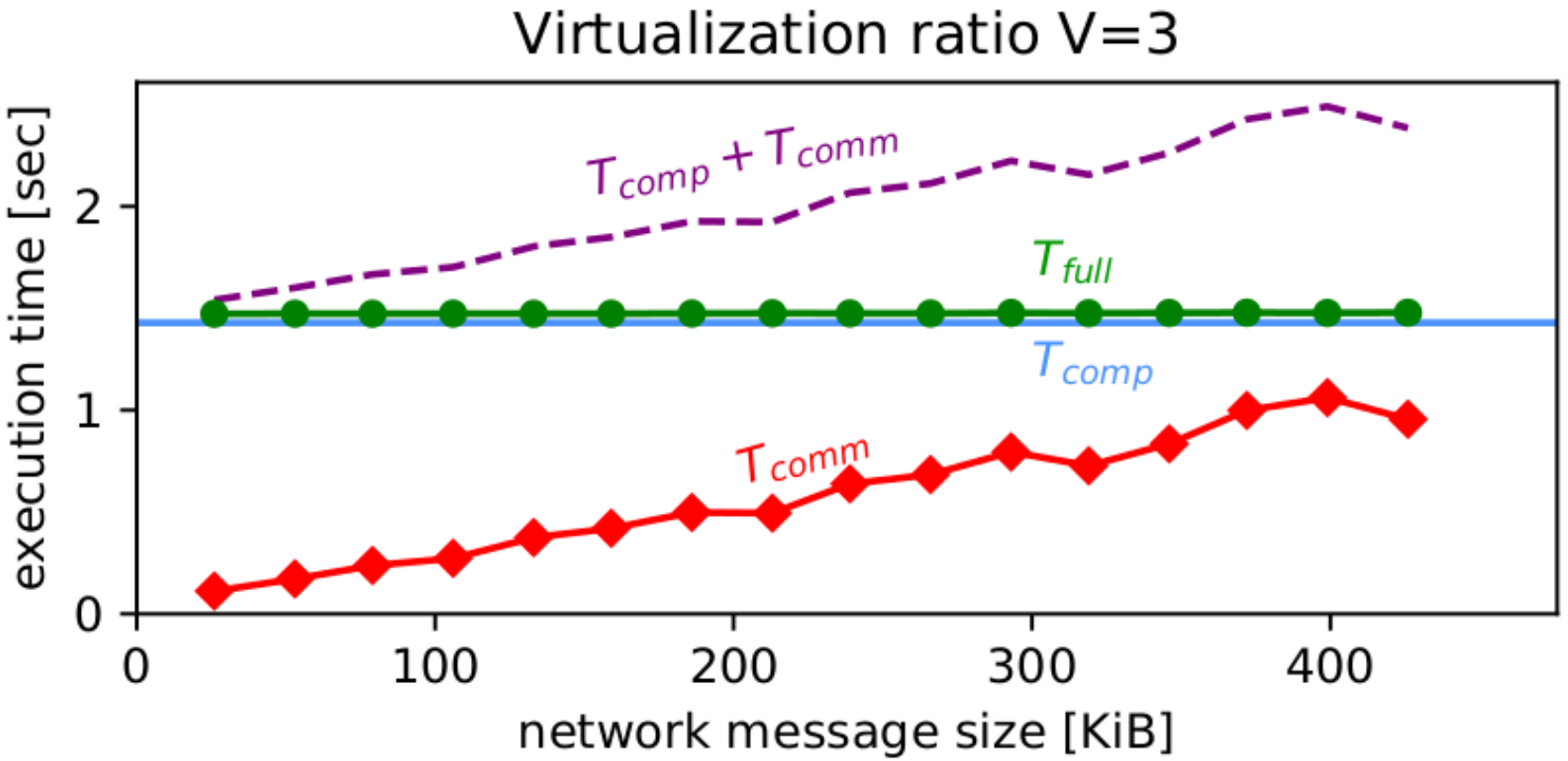
- Minimize overhead between switching between MPI Processes
- Similar idea as Adaptive MPI [3], MPC-MPI [4], FG-MPI [5], TMPI [6], Toucan [7], TOMPI [8]
- Implementation (http://spcl.inf.ethz.ch/Research/Parallel_Programming/TinyMPI/) close to Grappa [17]



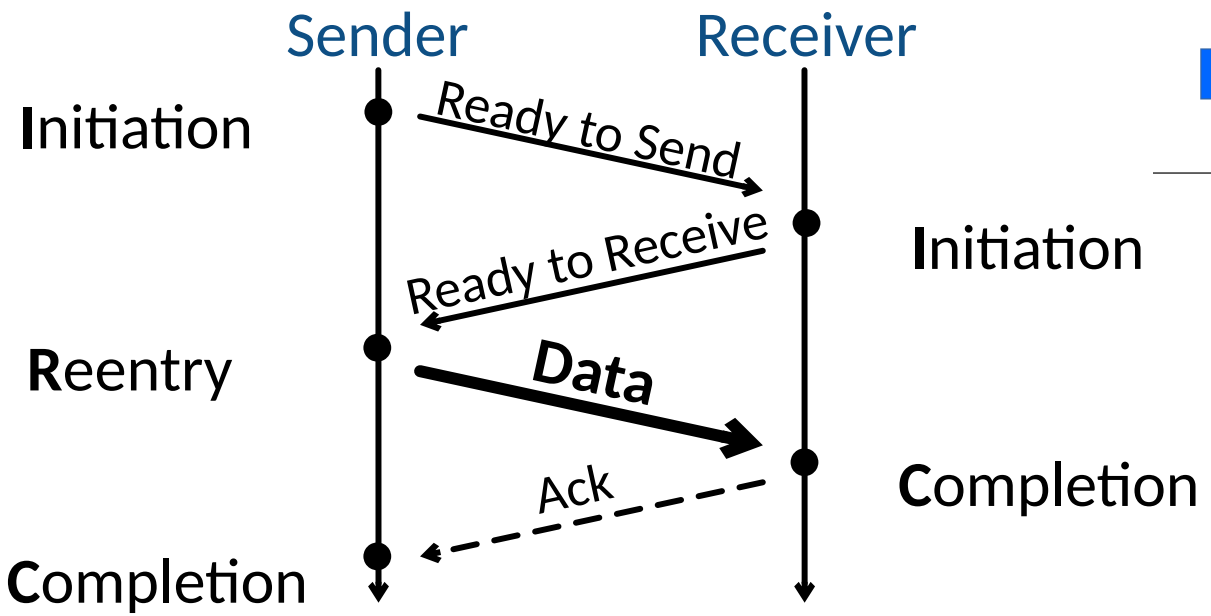
Does it Help to Overlap Communication?

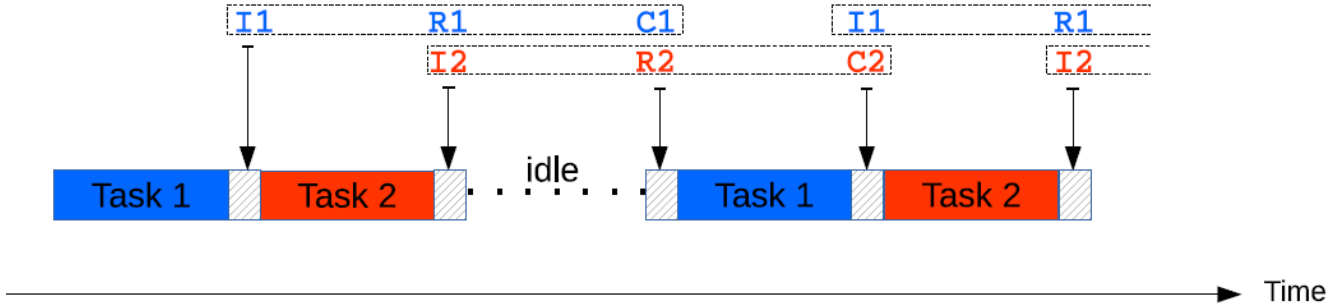


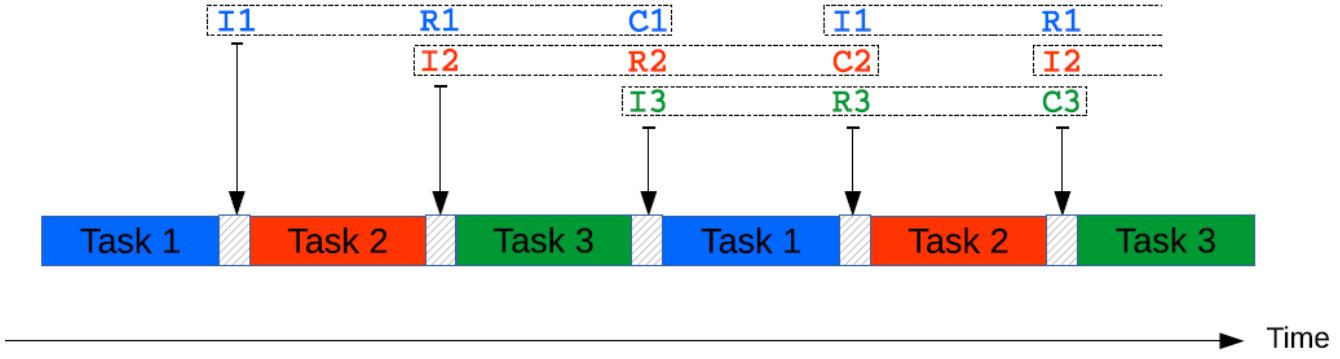
Does it Help to Overlap Communication? - Yes, with Enough Oversubscription!



Why Do We Need at Least Three Processes?





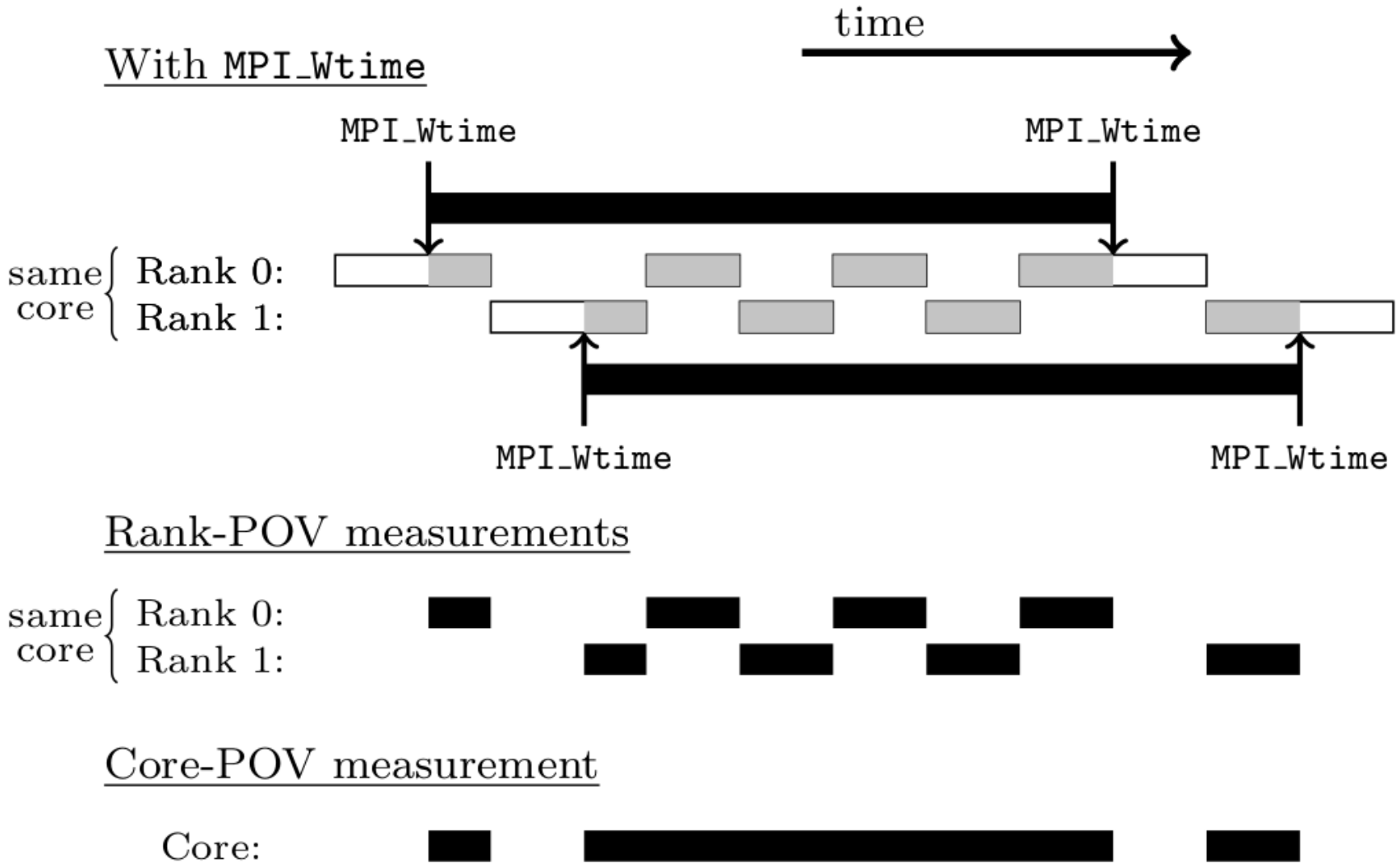


But All Benchmarks are Wrong!?

	MPC-MPI `MPI_Wtime`	TinyMPI `MPI_Wtime`	TinyMPI `MPIX_Rtime`
Tot. time [s]	11.1	10.9	2.85
DDOT time [s]	3.8	4.9	0.16
Tot. MFLOP/s	1755	1792	6846
DDOT MFLOP/s	319	245	7513

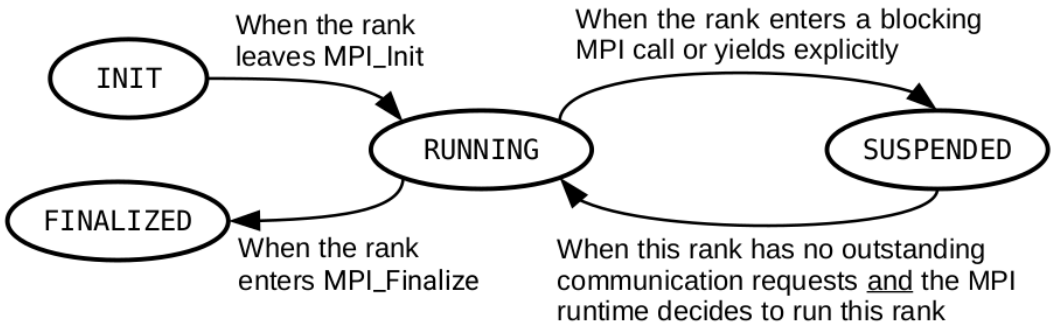
HPCCG reports measurements for rank 0. In these runs, 4 MPI processes share the same CPU core and collectively take 11.5 seconds for TinyMPI and 12 seconds for MPC-MPI.

But All Benchmarks are Wrong!? - And Here is Why.

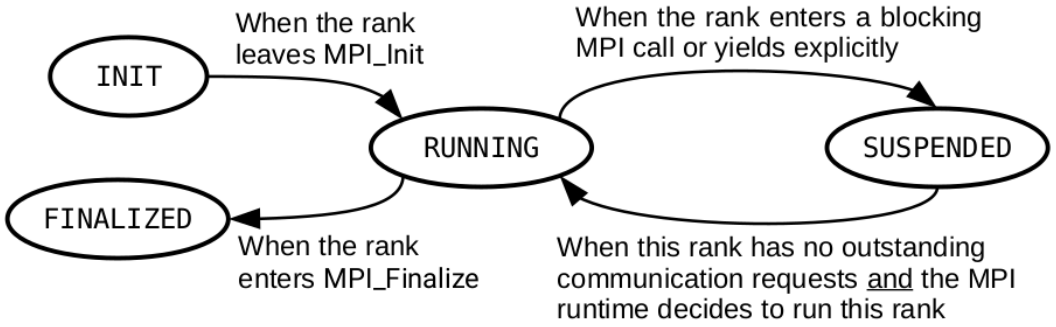


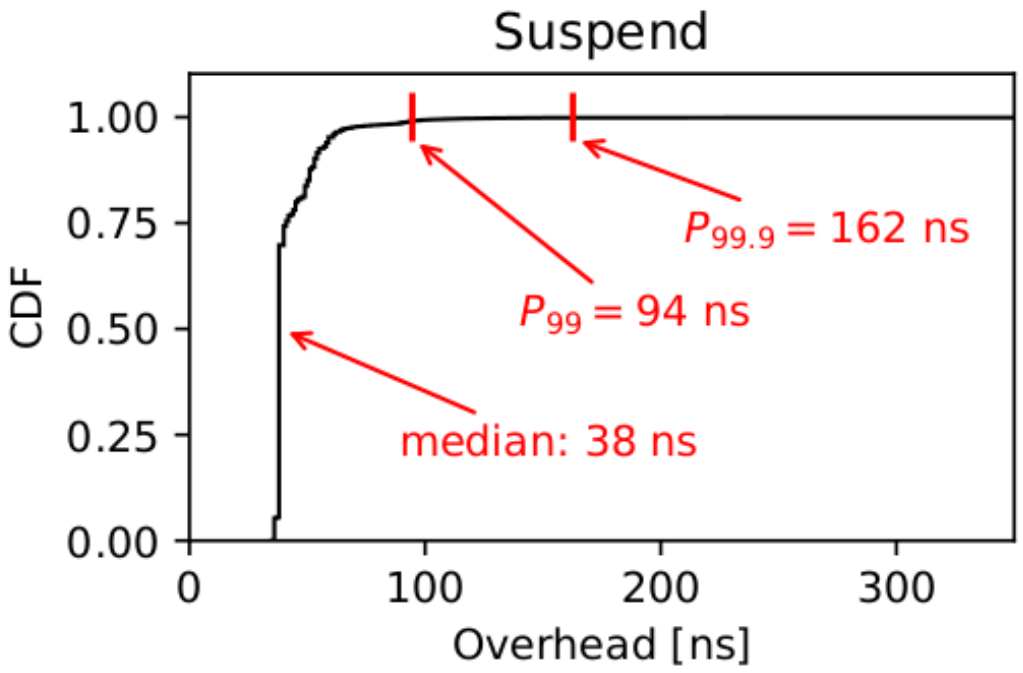
Process-POV Timers

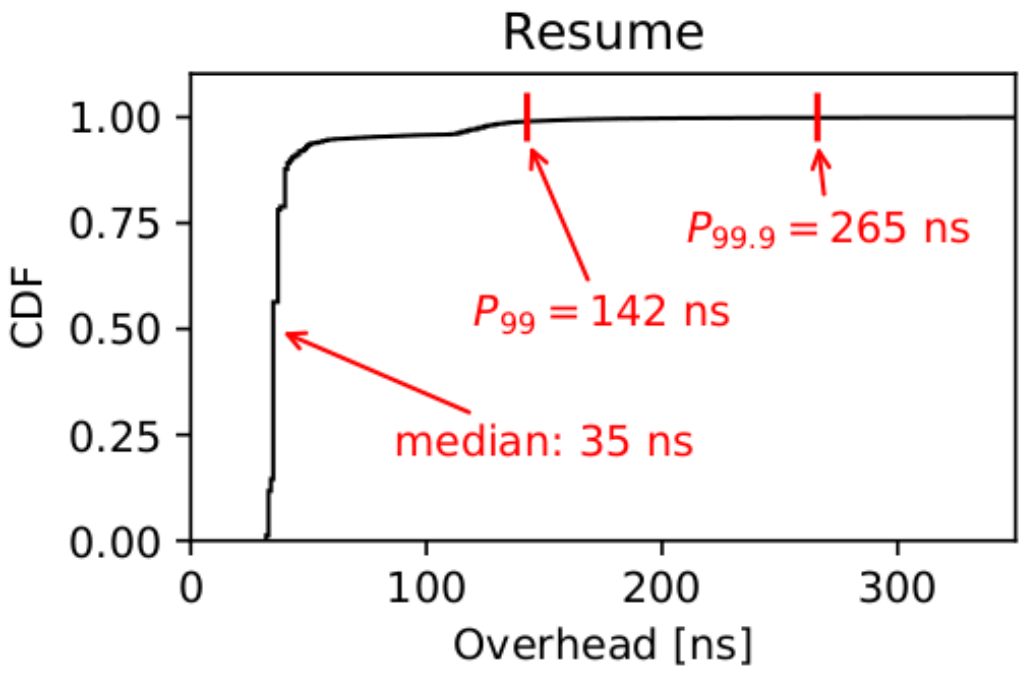
```
// There is one instance of a process_local
// variable for each process.
proc_local timestamp R; //process-POV clock
proc_local timestamp T; //helper

on_proc_init():  // Called for each proc in MPI_Init.
    R = 0;
    T = now();

on_proc_suspend(): // Advance the process-POV clock.
    R += now() - T;

on_proc_resume():
    // The process-POV time did not change.
    // Only the helper must update.
    T = now();

MPIX_Rtime():
    // Advance the clock and return the value.
    R += now() - T;
    T = now();
    return R;
```



Cost of Process-POV Timers







Conclusions

- MPI Virtualization helps to make “poorly” written codes perform well
- Choosing the right factor of oversubscription is important but depends on environment parameters
- MPIs timing infrastructure is a poor match for virtualized MPI environments
  - We propose adding per-process / per-core timers
  - Which add ~30ns of switching overhead