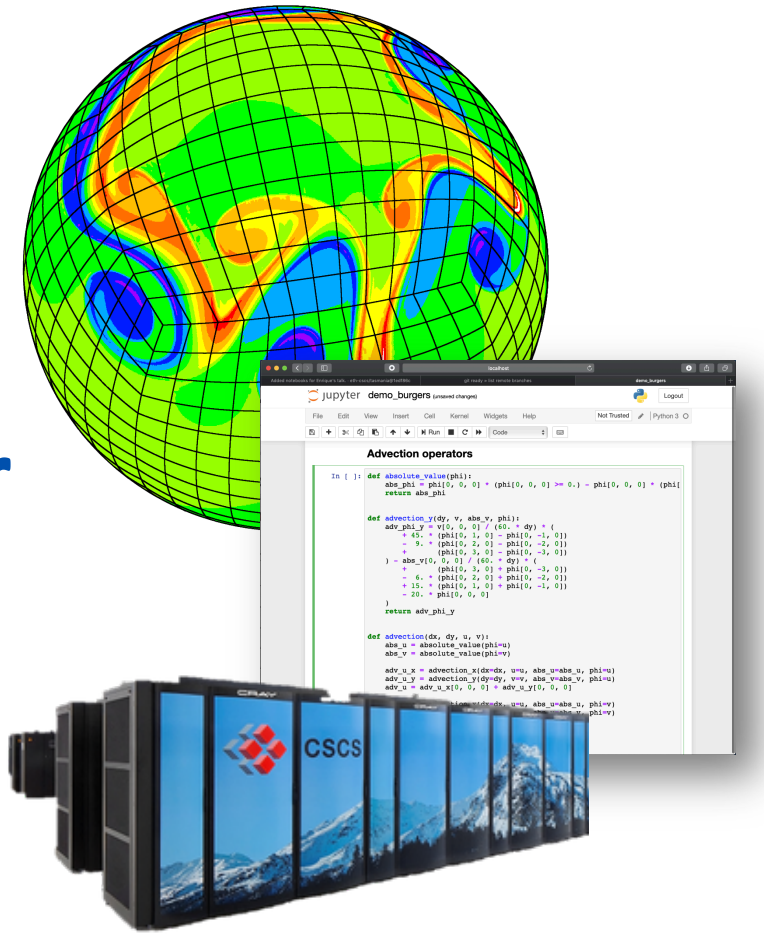
# High Performance Computing for Weather and Climate (HPC4WC)

Content: Distributed Memory Parallelism / MPI
Lecturers: Oliver Fuhrer
Block course 701-1270-00L
Summer 2020

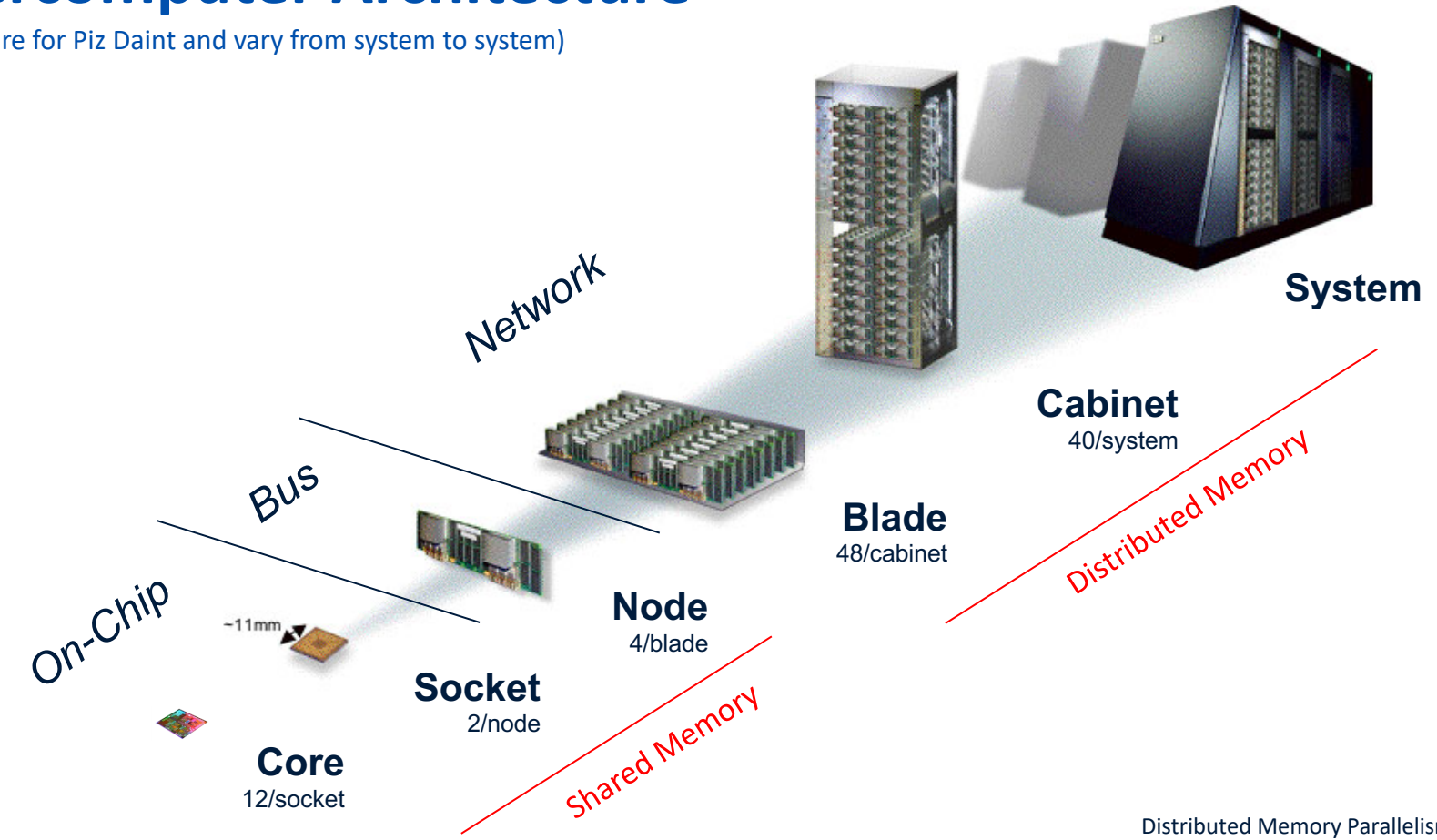


# Learning goals

- Understand distributed memory parallelism and how it is different from shared memory parallelism
- Learn basic message passing patterns using MPI
- Be able to apply domain decomposition for solving partial differential equations
- Understand the concept of halo points and able to implement a halo-update.

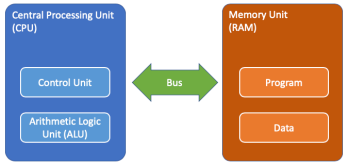
# Supercomputer Architecture

(Numbers are for Piz Daint and vary from system to system)

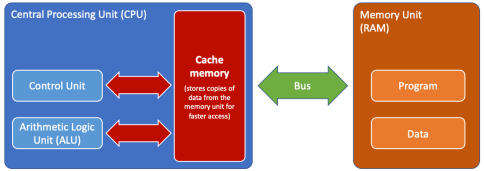


# Computer Architecture

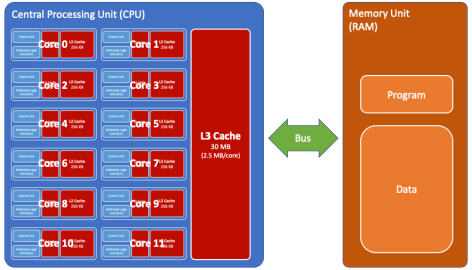
## Von Neumann



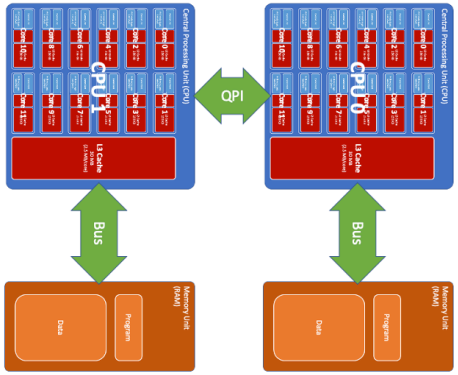

## Cache hierarchy



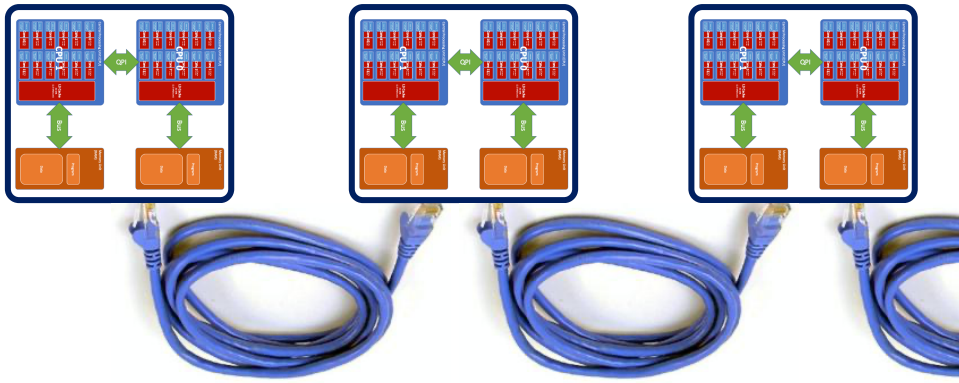

## Modern CPU




## Multicore Node



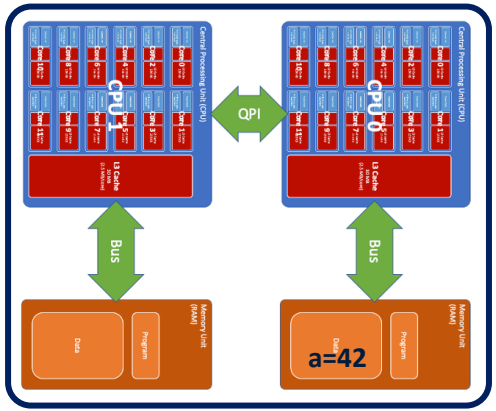

## Many nodes



# Distributed Memory

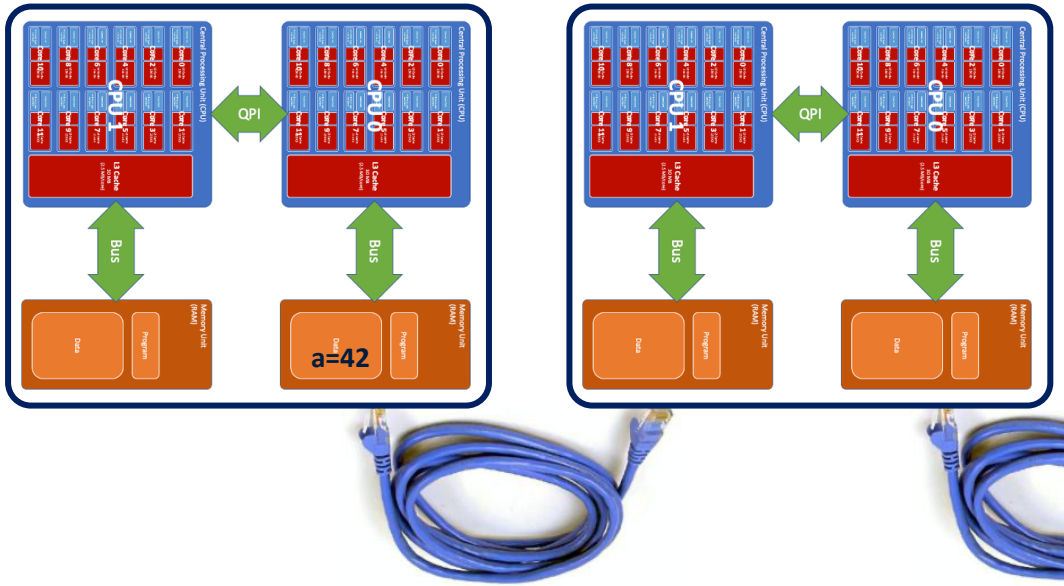
All cores on a node share the same address space / memory

```
>>> print(a)
42
```




Nodes have different address spaces / memories. Variables are not shared.

```
>>> print(a)
NameError: name 'a' is not defined
```

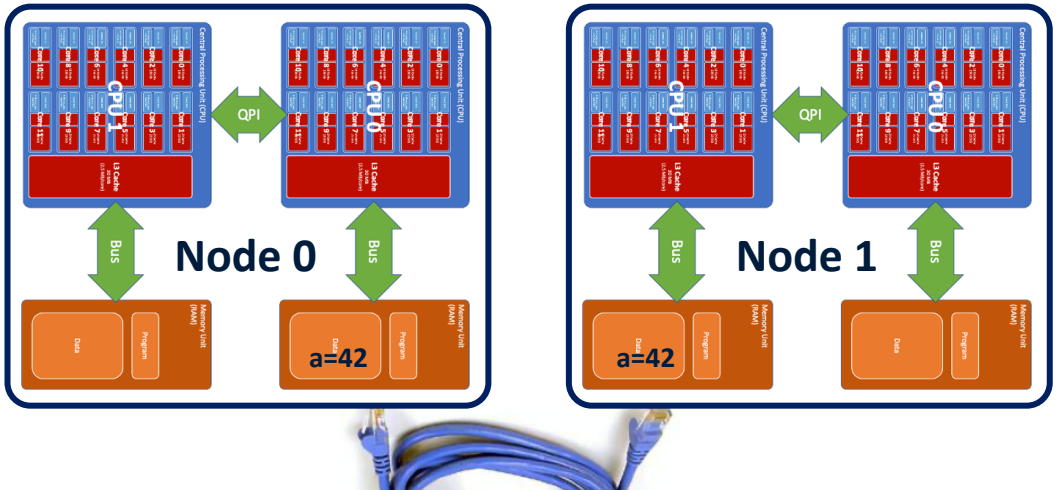


# Message Passing

- Information between nodes is transferred over a network cable using a message passing protocol.

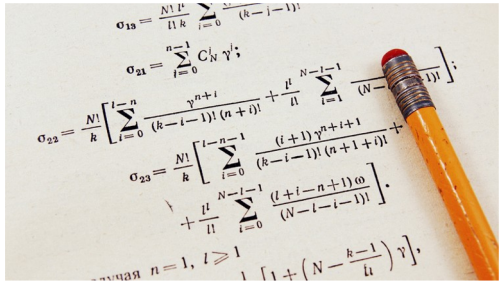
```
>>> send(a, destination=1)
>>> address(a)
0x001a947e3211
```

```
>>> a = recv(source=0)
>>> print(a)
42
>>> addres(a)
0x002f33498e77
```



# Parallel Computing (shared memory)

Problem



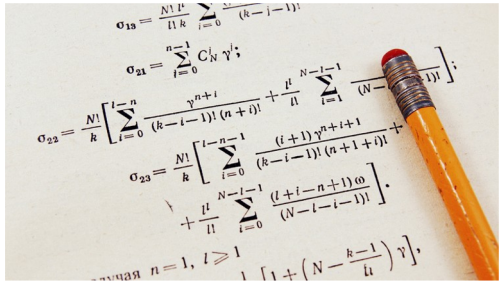
Worker 1

Notebook

Worker 2

# Parallel Computing (distributed memory)

Problem



Worker 1

Notebook

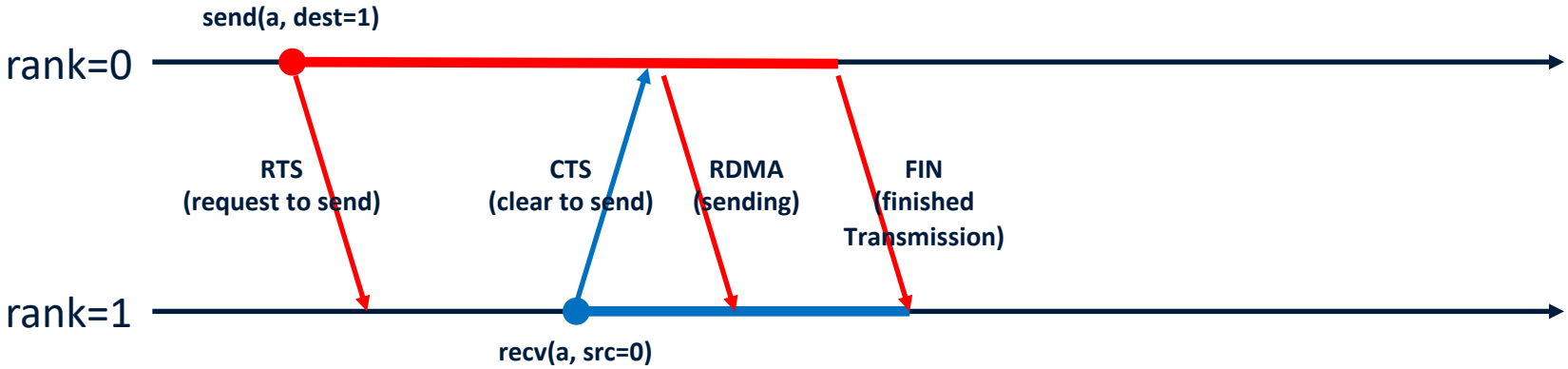
Worker 2

Notebook

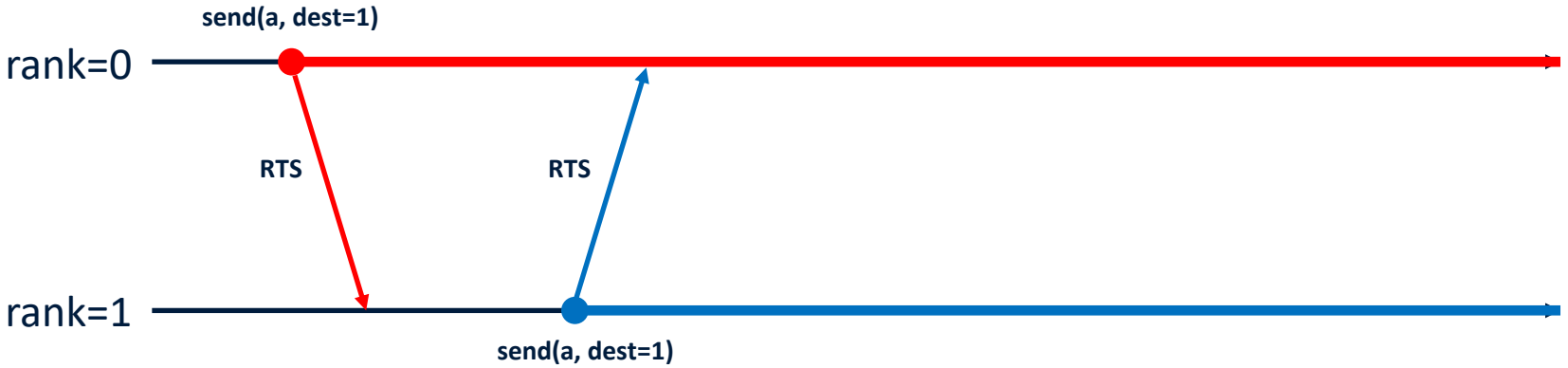
# Message Passing Interface (MPI)


- MPI is a standardized and portable message passing standard. (https://www.mpi-forum.org/ and https://github.com/mpi-forum)
- Version 1.0 in 1992, latest Version 3.1 in 2015, Version 4.0 ratification in progress
- Support for Fortran, C, C++, Python, Julia, …
- Implemented as a library that provides message passing semantics.
- Several implementations
  - MVAPICH
  - OpenMPI
  - Cray MPI
  - …
- Available on almost any architecture
  - Linux Laptop (apt-get install mpich)
  - Supercomputer
  - Google Cloud Platform
  - …

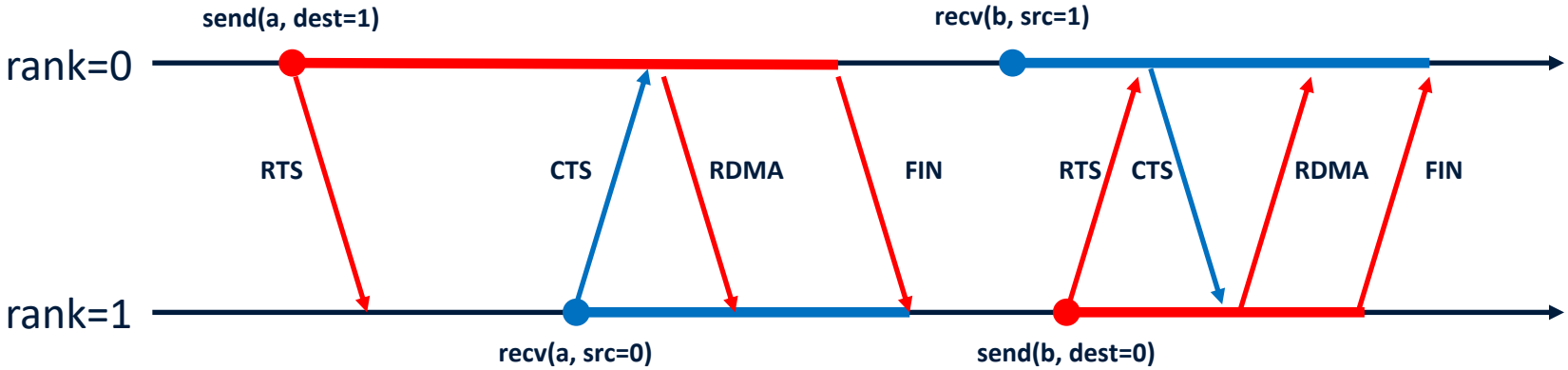
# Send / Receive



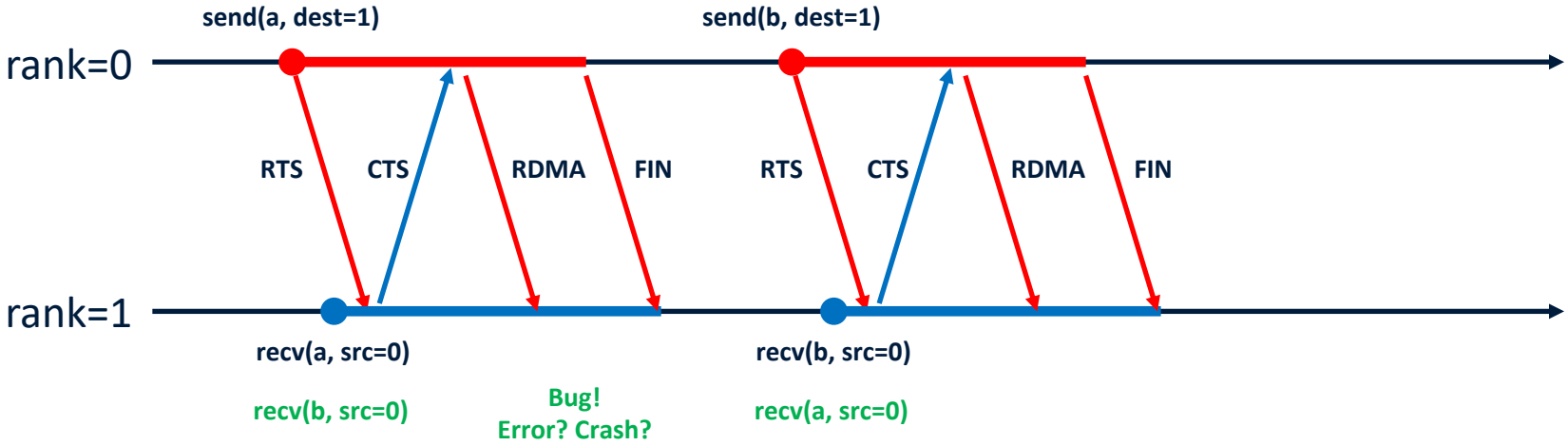
# Deadlock



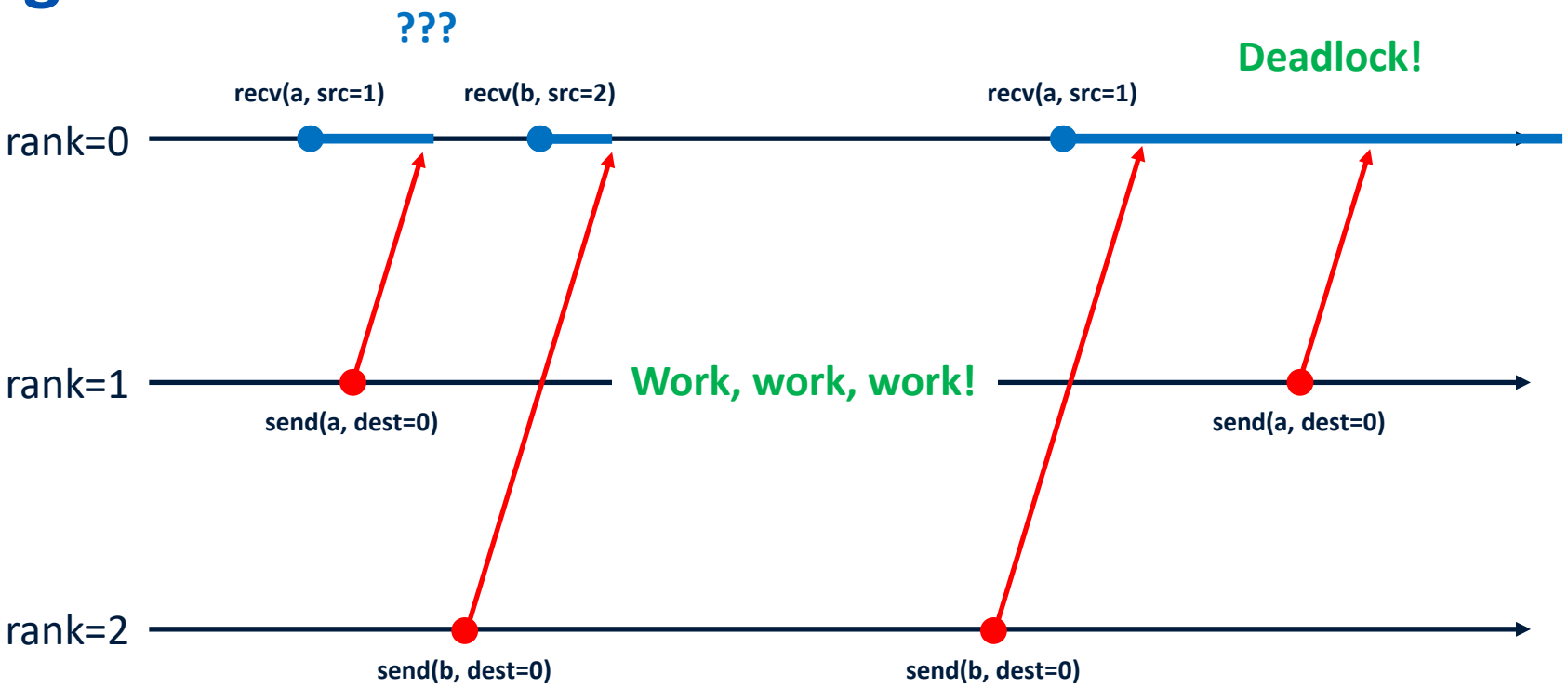
# Matching Send / Recv



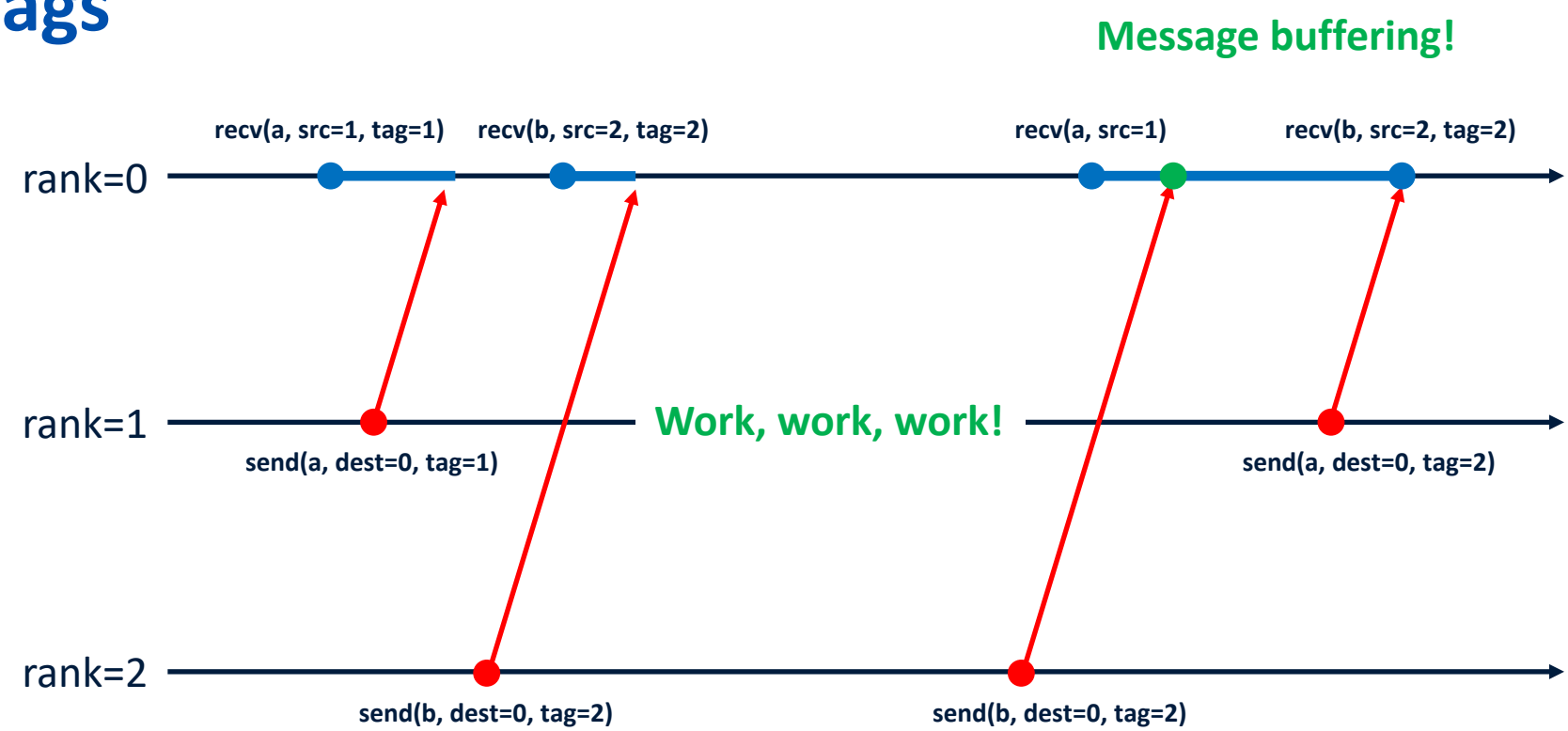
# Message Order



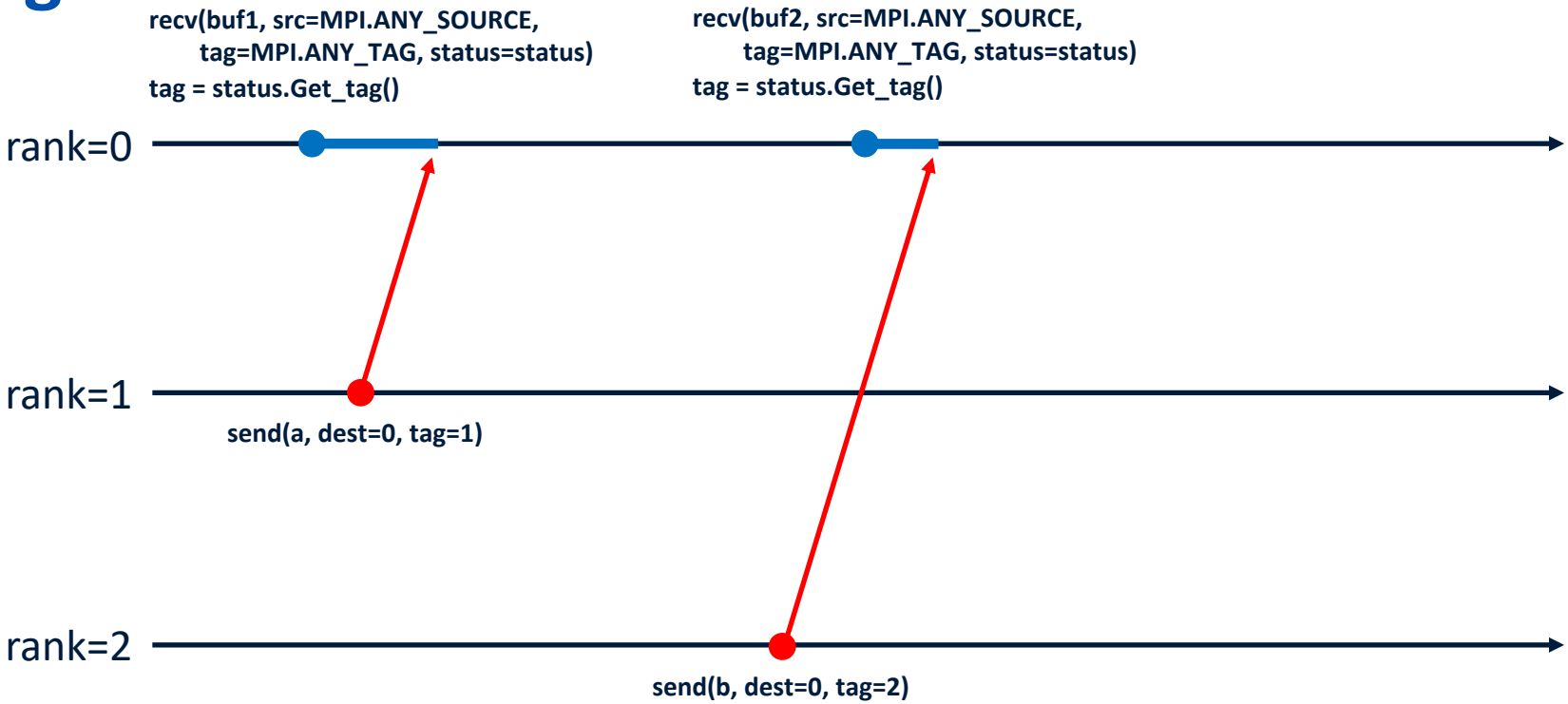
# Tags


???
Deadlock!
recv(a, src=1)
recv(b, src=2)
recv(a, src=1)
rank=0
rank=1
rank=2
Work, work, work!
send(a, dest=0)
send(a, dest=0)
send(b, dest=0)
send(b, dest=0)

# Tags


Message buffering!
recv(a, src=1, tag=1)
recv(b, src=2, tag=2)
recv(a, src=1)
recv(b, src=2, tag=2)
rank=0
rank=1
rank=2
Work, work, work!
send(a, dest=0, tag=1)
send(a, dest=0, tag=2)
send(b, dest=0, tag=2)
send(b, dest=0, tag=2)

# Tags


recv(buf1, src=MPI.ANY_SOURCE,
tag=MPI.ANY_TAG, status=status)
tag = status.Get_tag()
recv(buf2, src=MPI.ANY_SOURCE,
tag=MPI.ANY_TAG, status=status)
tag = status.Get_tag()
rank=0
rank=1
rank=2
send(a, dest=0, tag=1)
send(b, dest=0, tag=2)

# Lab Exercises

**01-test-MPI-setup.ipynb**

- Test the setup of your JupyterHub Server to make sure that MPI is working correctly.

**02-MPI-introduction.ipynb**

- Step-by-step introduction to MPI concepts in Python (mpi4py).

**03-domain-decomposition.ipynb**

- Learn about domain-decomposition.
- Apply domain-decomposition to a simple 1d example.
- Apply domain-decomposition to the stencil2d.py program.