

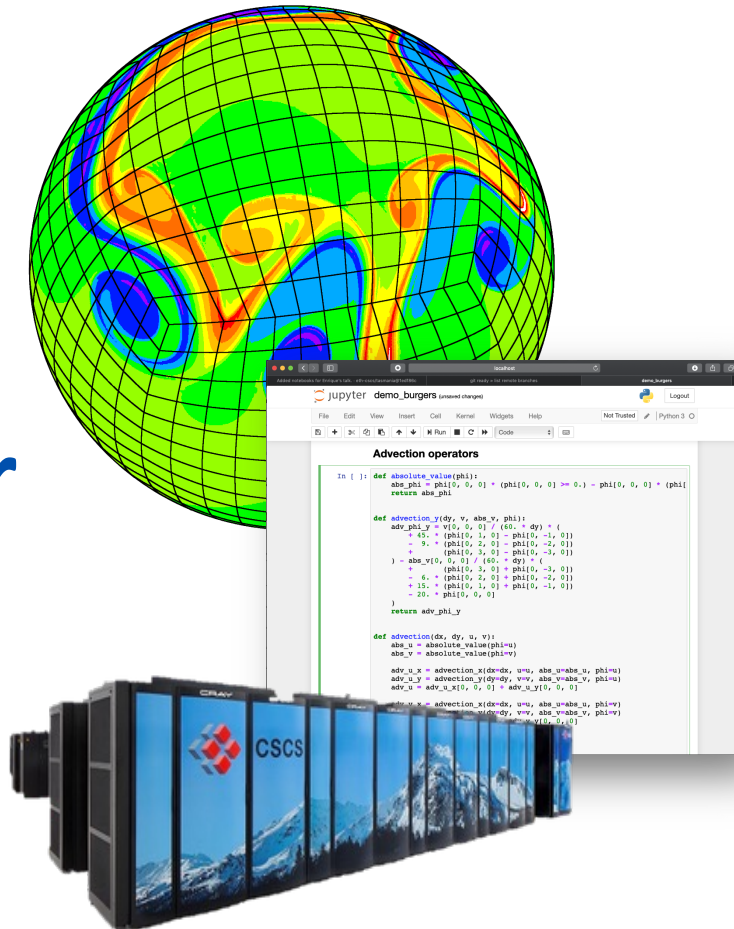
High Performance Computing for Weather and Climate (HPC4WC)

Content: Distributed Memory Parallelism / MPI

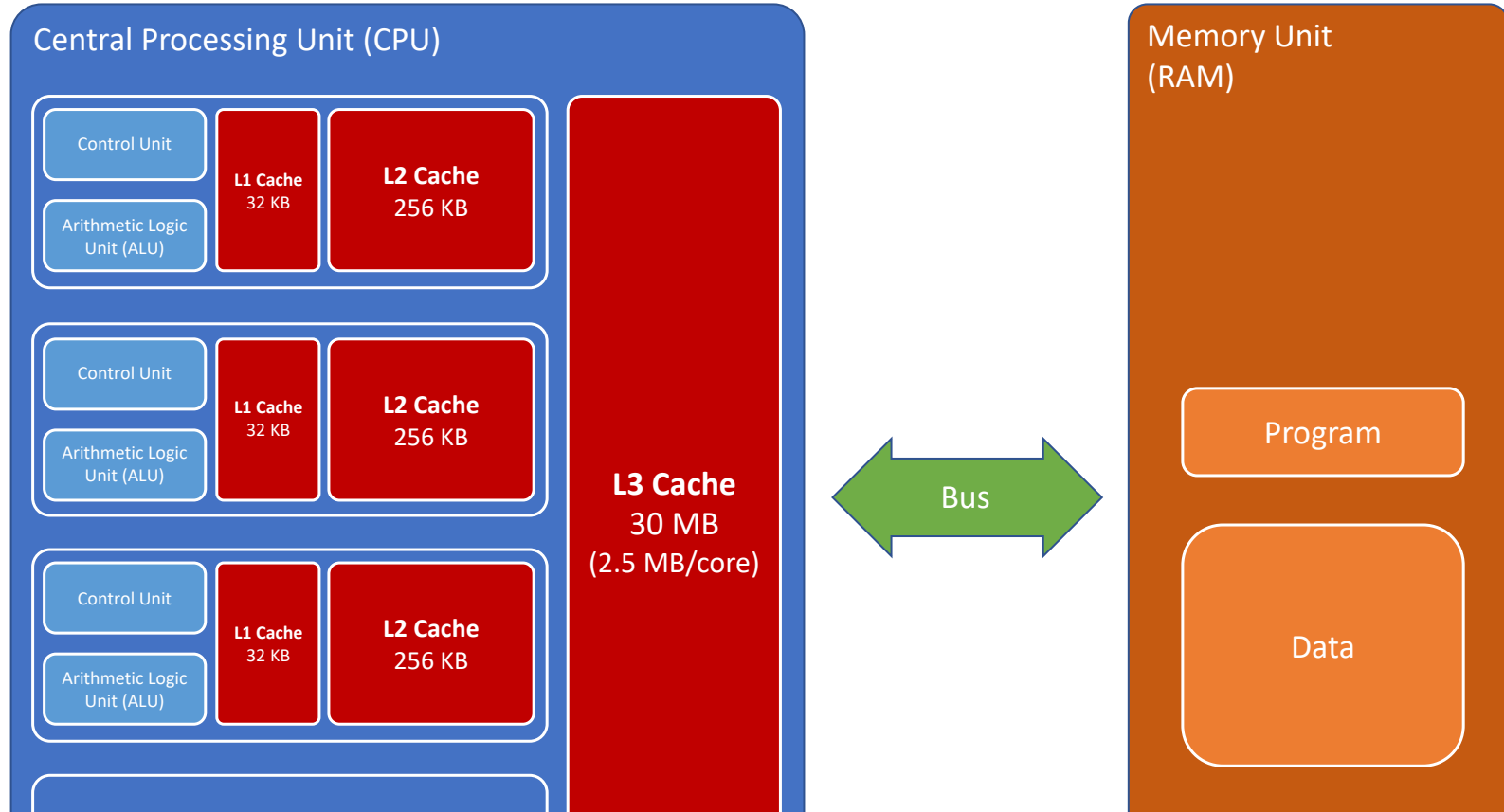
Lecturers: Oliver Fuhrer

Block course 701-1270-00L

Summer 2022

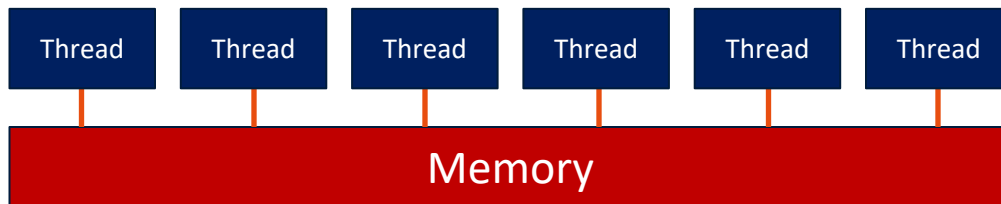


Memory hierarchy (L1, L2, L3, DRAM)

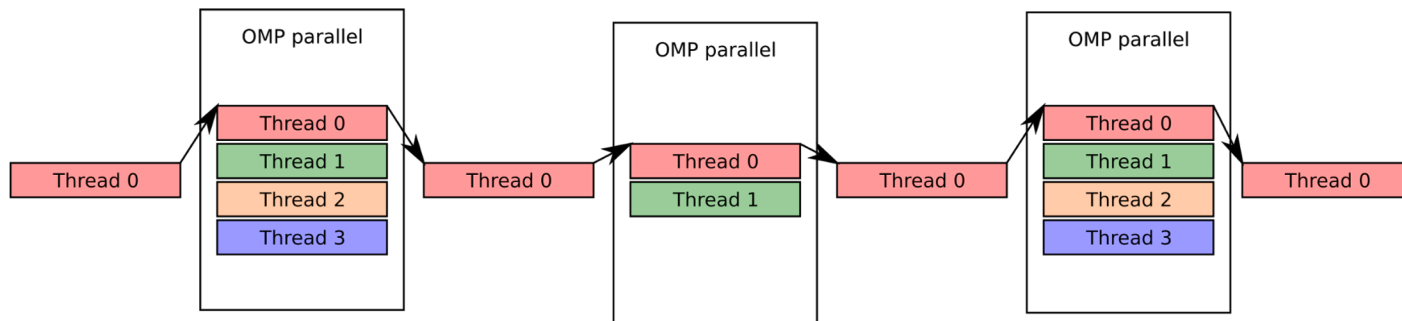


Shared memory parallelism

Parallel workers (threads) share the same view of memory



Fork-join model (threads)



OpenMP directives

For weather and climate models, most of the work is in loops over all gridpoints

```
!$omp parallel do default(none) private(tmp1_field, laplap) &
!$omp shared(nz, num_halo, ny, nx, in_field, num_iter, out_field, alpha)
do k = 1, nz
  do j = 1 + num_halo - 1, ny + num_halo + 1
    do i = 1 + num_halo - 1, nx + num_halo + 1
      tmp1_field(i, j) = -4._wp * in_field(i, j, k)      &
        + in_field(i - 1, j, k) + in_field(i + 1, j, k) &
        + in_field(i, j - 1, k) + in_field(i, j + 1, k)
    end do
  end do
end do
```

Step 0: Determine where most of the time is spent (e.g. using a profiler)

Step 1: Determine what can be run in parallel, apply OpenMP directives

Step 2: Apply data-sharing rules (e.g. specify which variables are private/shared)

Step 3: Optimize using profiler (e.g. schedule)

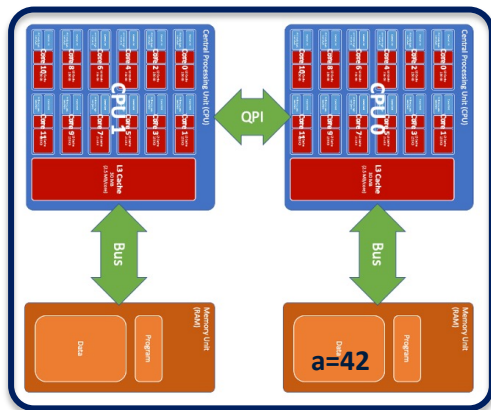
Learning goals

- 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.

Shared vs. Distributed Memory

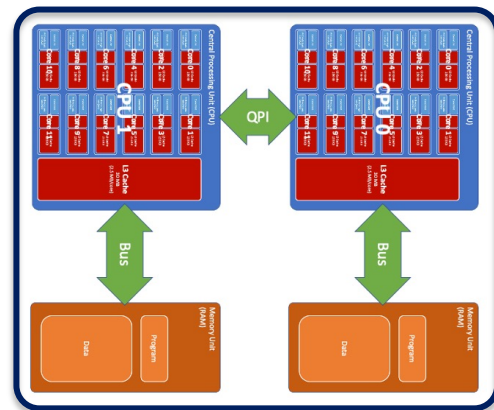
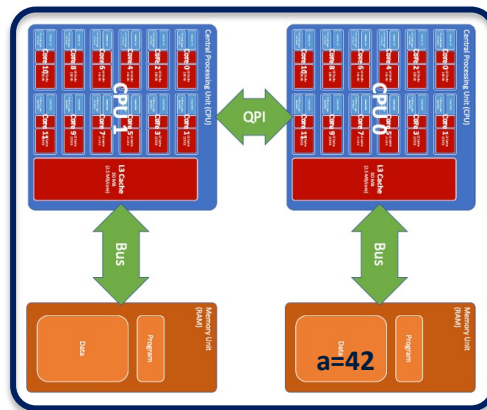
All cores on a node share the same address space / memory

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



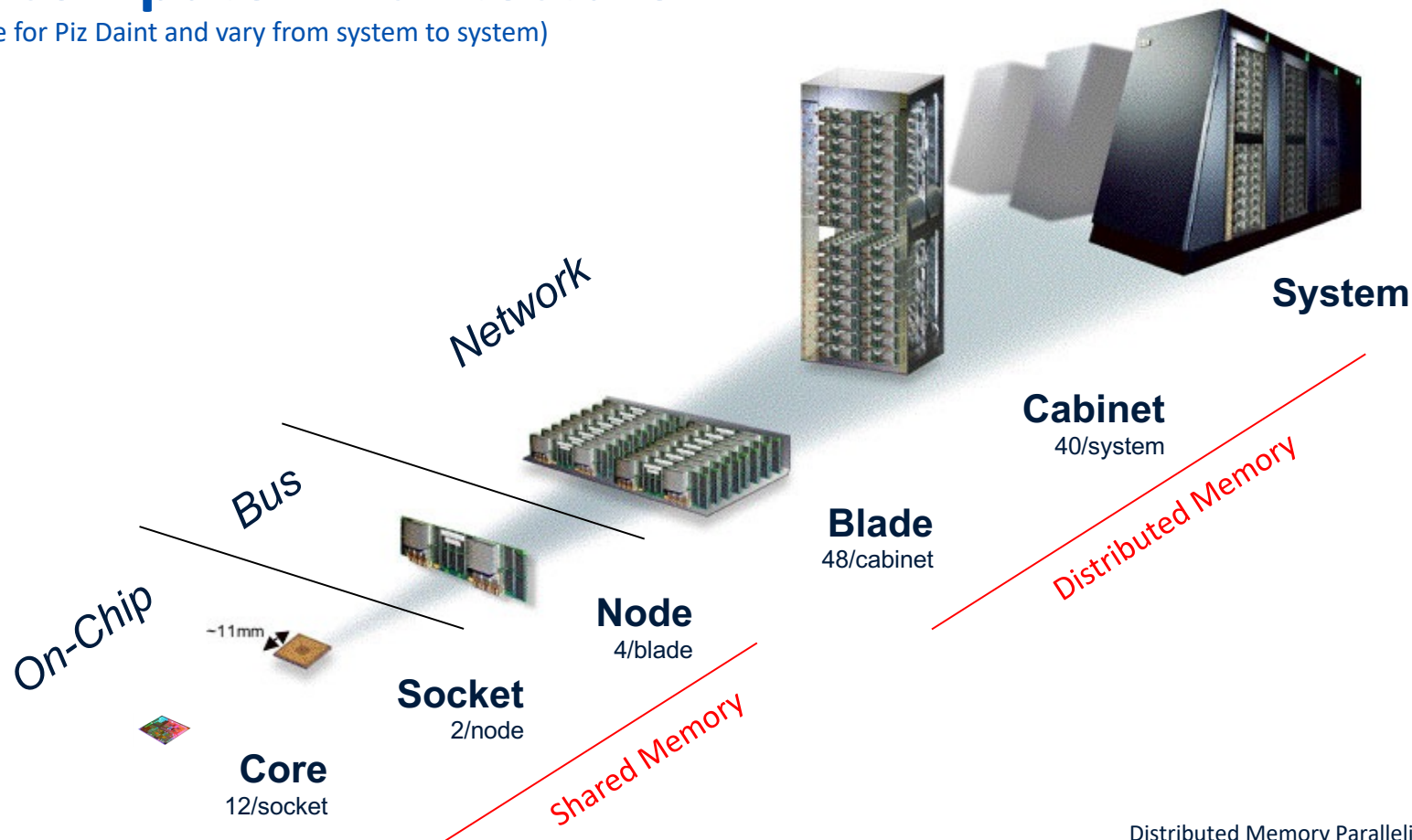
Nodes have different address spaces / memories. Variables are not shared.

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



Supercomputer Architecture

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

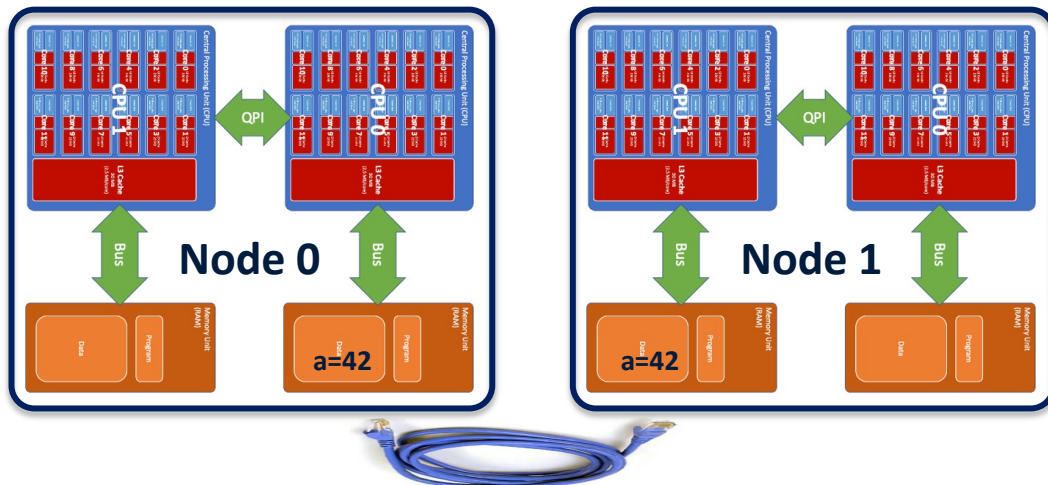


Message Passing

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

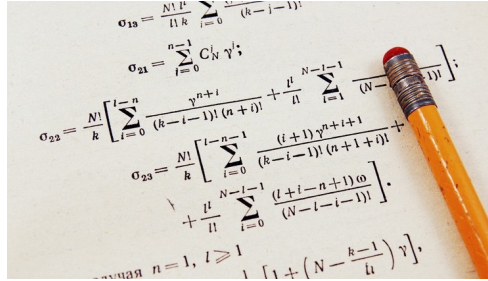
```
>>> a = 42
>>> address(a)
0x001a947e3211
>>> send(a, destination=1)
```

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



Parallel Computing (shared memory)

Problem



Handwritten mathematical formulas on a piece of paper, with a yellow pencil resting on it. The formulas include:

$$\sigma_{10} = \frac{N!}{k!} \sum_{j=0}^k \frac{1}{(k-j)!}$$
$$\sigma_{21} = \sum_{j=0}^{n-1} C_N^j \gamma^j$$
$$\sigma_{22} = \frac{N!}{k} \left[\sum_{i=0}^{l-n} \frac{\gamma^{n+i}}{(k-i-1)! (n+i)!} + \frac{l!}{k} \sum_{i=1}^{N-l-1} \frac{1}{(N-i-1)!} \right]$$
$$\sigma_{23} = \frac{N!}{k} \left[\sum_{i=0}^{l-n-1} \frac{(i+1) \gamma^{n+i+1}}{(k-i-1)! (n+i+1)!} + \frac{l!}{k} \sum_{i=0}^{N-l-1} \frac{(l+i-n+1) \omega}{(N-l-i-1)!} \right]$$

Below the formulas, it says: $n=1, l \geq 1$

Worker 1



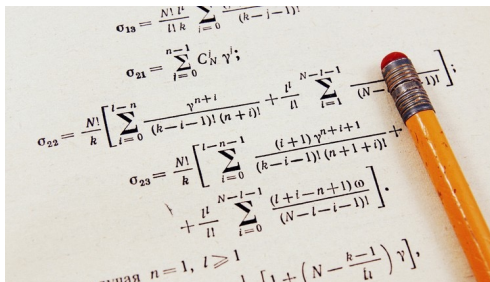
Worker 2



Notebook

Parallel Computing (distributed memory)

Problem



Handwritten mathematical formulas on a piece of paper, with an orange pencil resting on it. The formulas include:

$$\sigma_{10} = \frac{N!}{l! k} \sum_{i=0}^l \frac{\gamma^i}{(k-i-1)!}$$
$$\sigma_{21} = \sum_{i=0}^{n-1} C_N^i \gamma^i$$
$$\sigma_{22} = \frac{N!}{k} \left[\sum_{i=0}^{l-n} \frac{\gamma^{n+i}}{(k-i-1)! (n+i)!} + \frac{l!}{l!} \sum_{i=1}^{N-l-1} \frac{N-l-1}{(N-i-1)!} \right]$$
$$\sigma_{23} = \frac{N!}{k} \left[\sum_{i=0}^{l-n-1} \frac{(i+1) \gamma^{n+i+1}}{(k-i-1)! (n+i+1)!} + \frac{l!}{l!} \sum_{i=0}^{N-l-1} \frac{(l+i-n+1) \omega}{(N-l-i-1)!} \right]$$

Below the formulas, it says: $n=1, l \geq 1$

Worker 1



Notebook

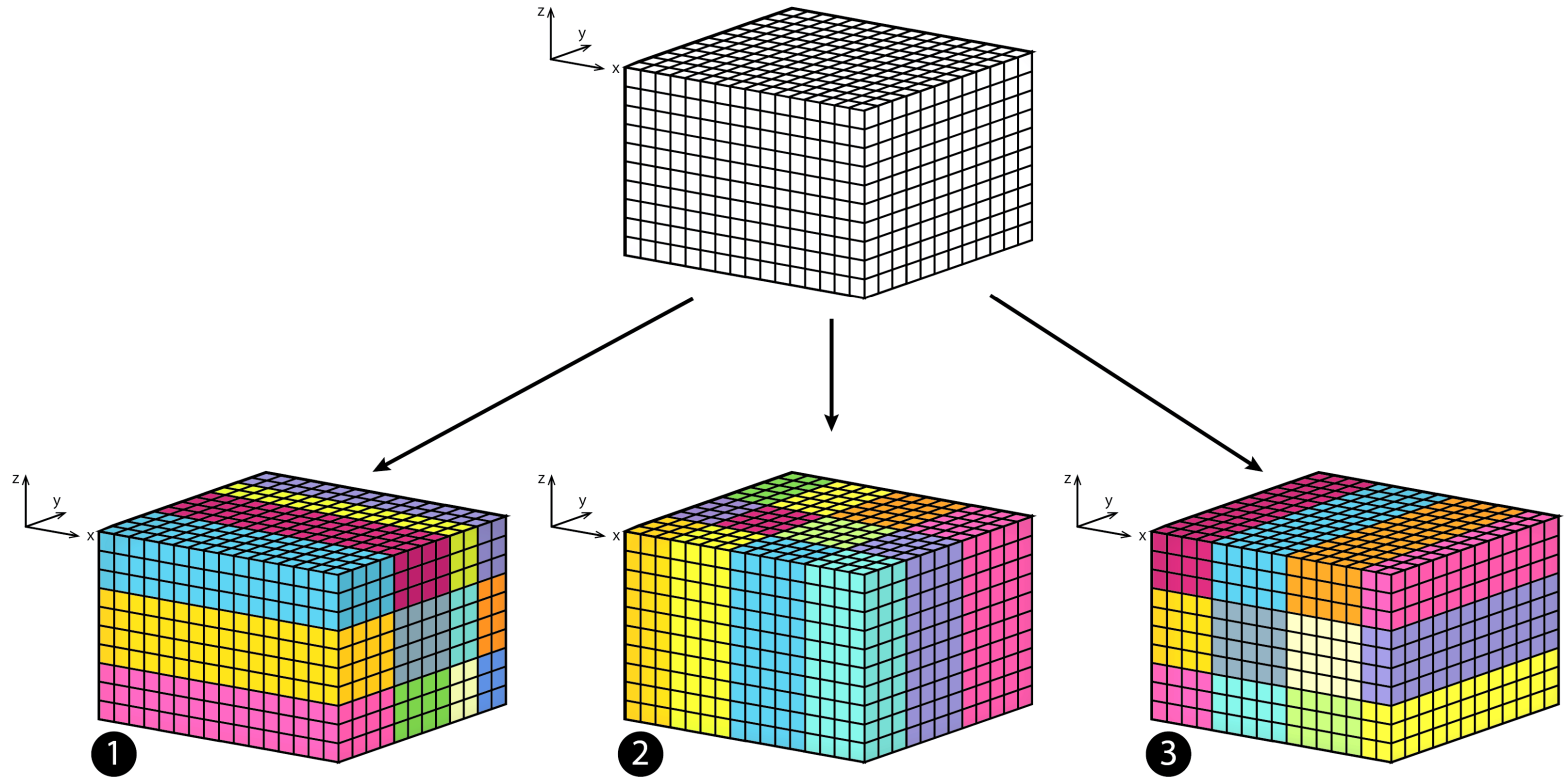


Worker 2



Notebook

Domain Decomposition



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
 - ...

Lab Exercises

01-test-MPI-setup.ipynb

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

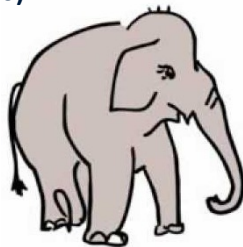
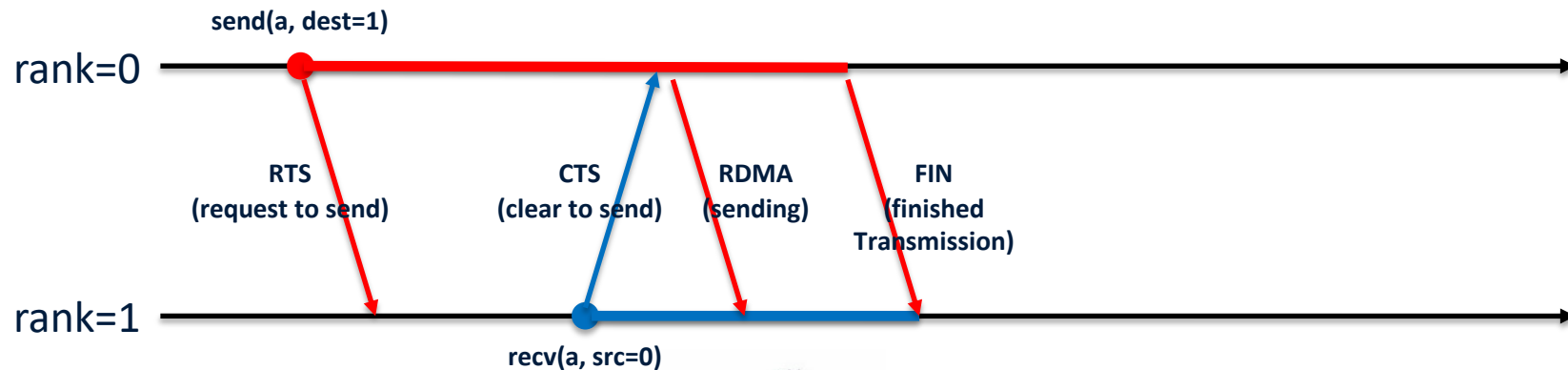
02-MPI-introduction.ipynb

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

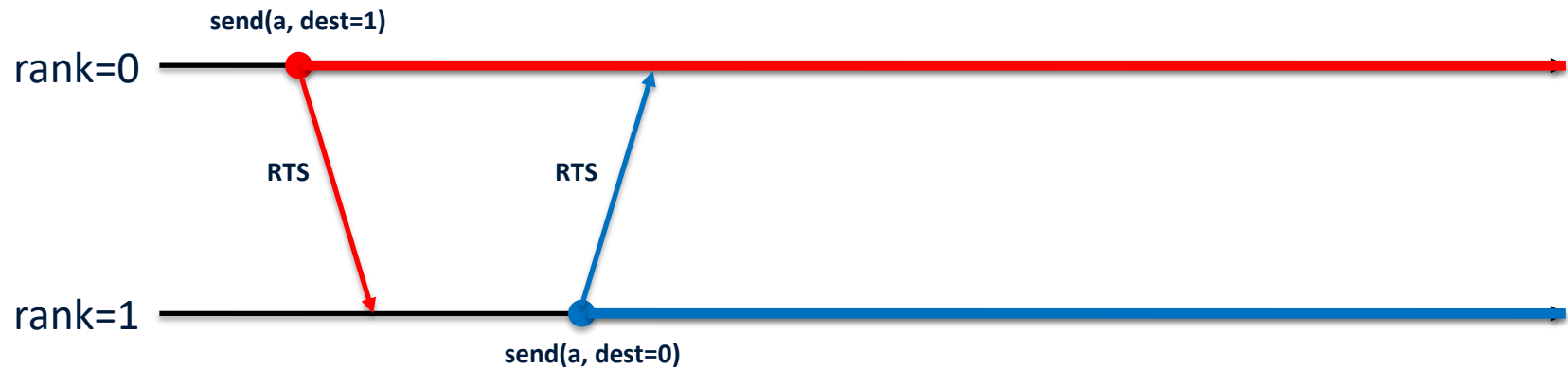
03-domain-decomposition.ipynb

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

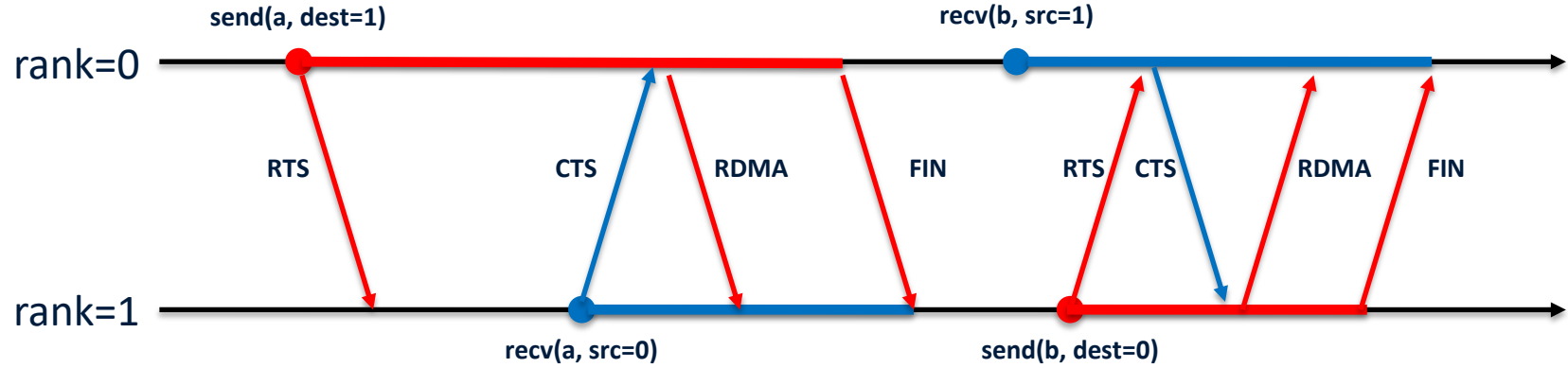
Send / Receive (Rendezvous protocol = large messages)



Deadlock



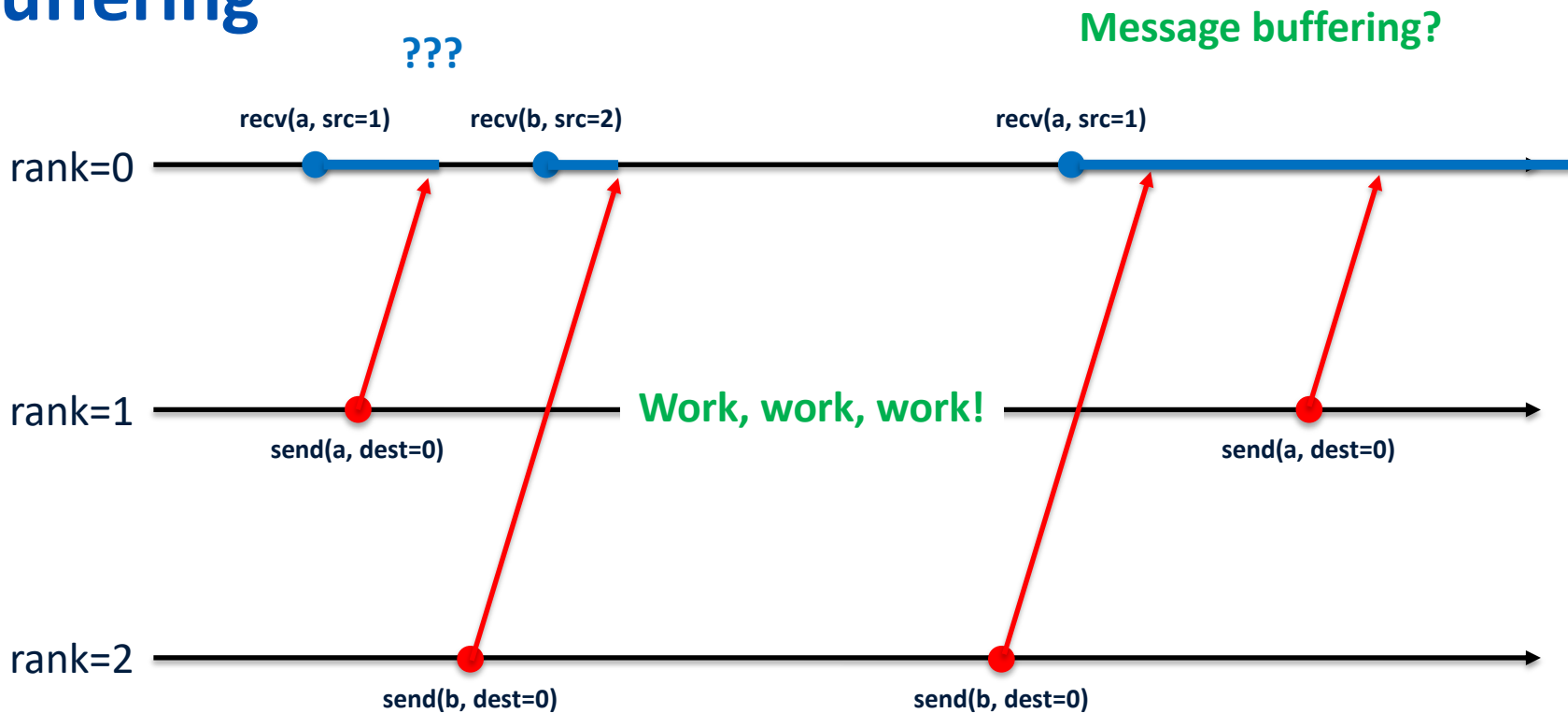
Matching Send / Recv



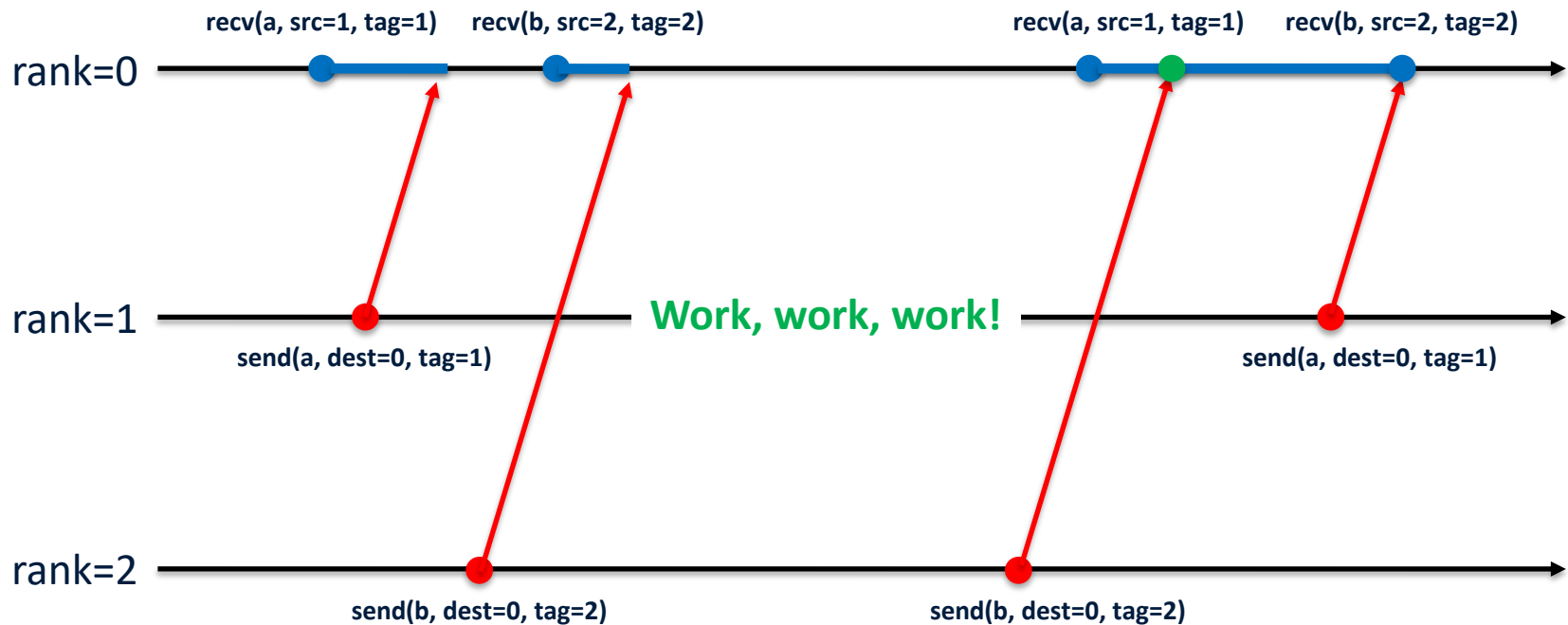
Order!



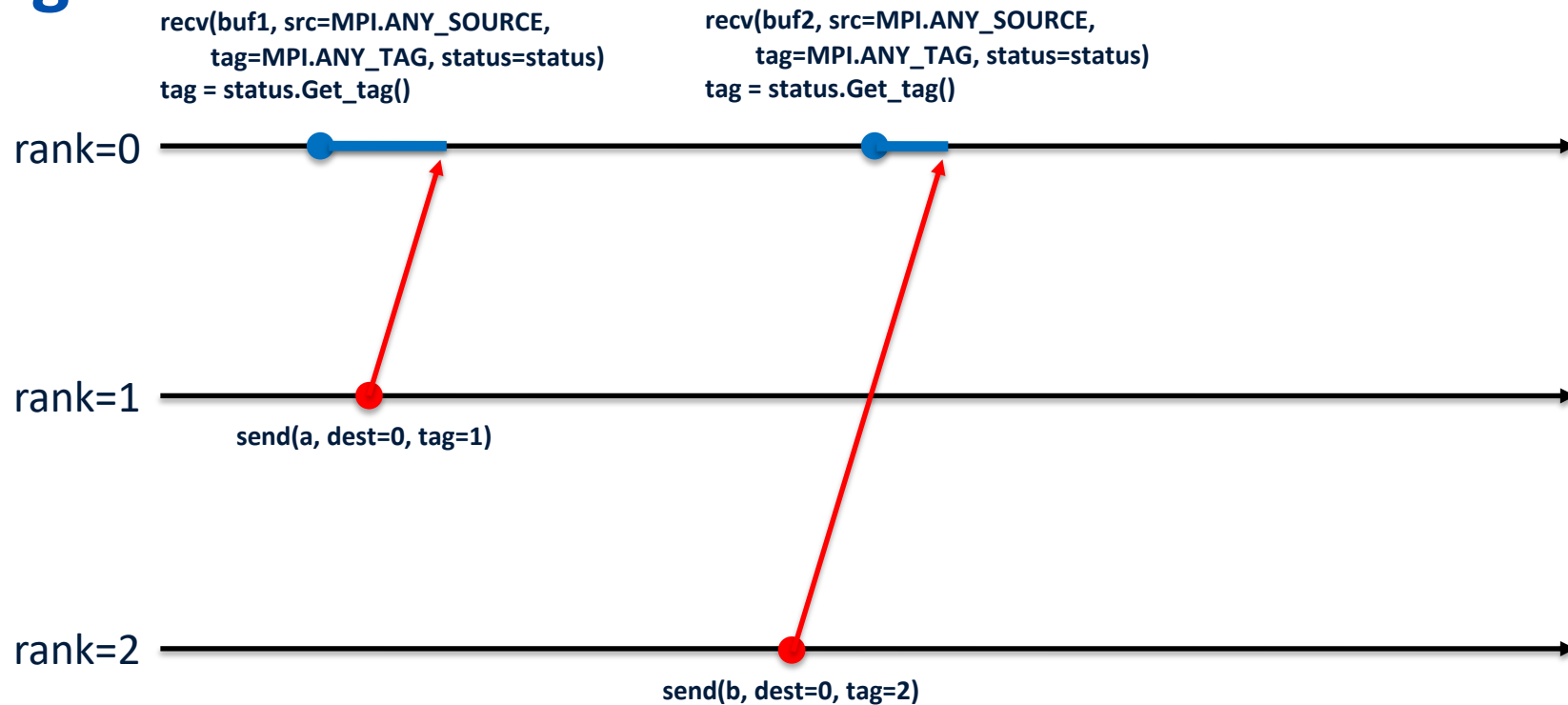
Buffering



Tags

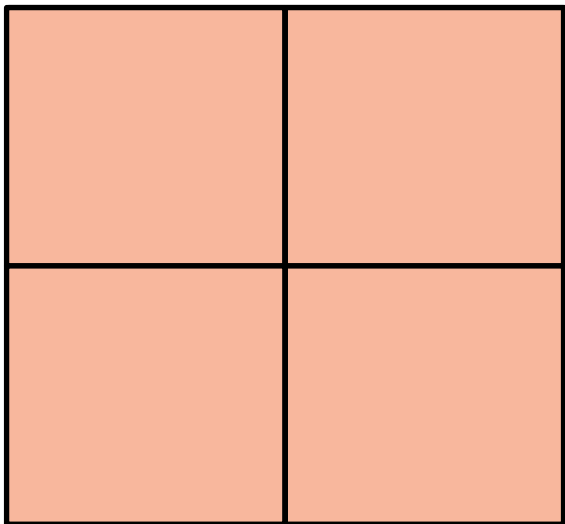


Tags

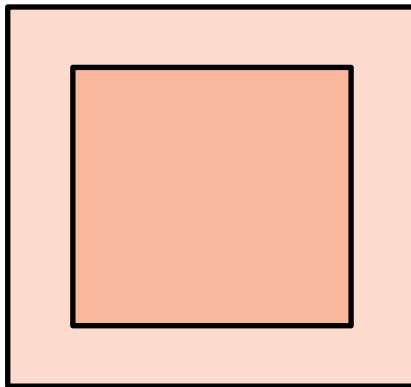


Corners

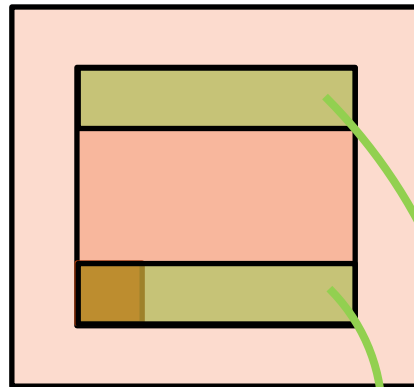
global domain



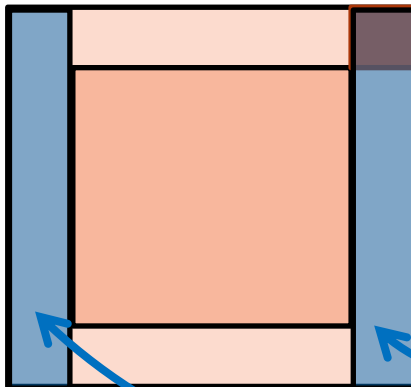
rank = 2



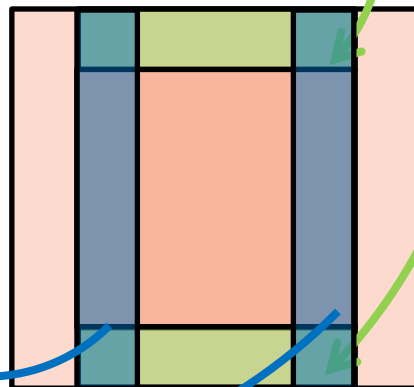
rank = 3



rank = 0



rank = 1

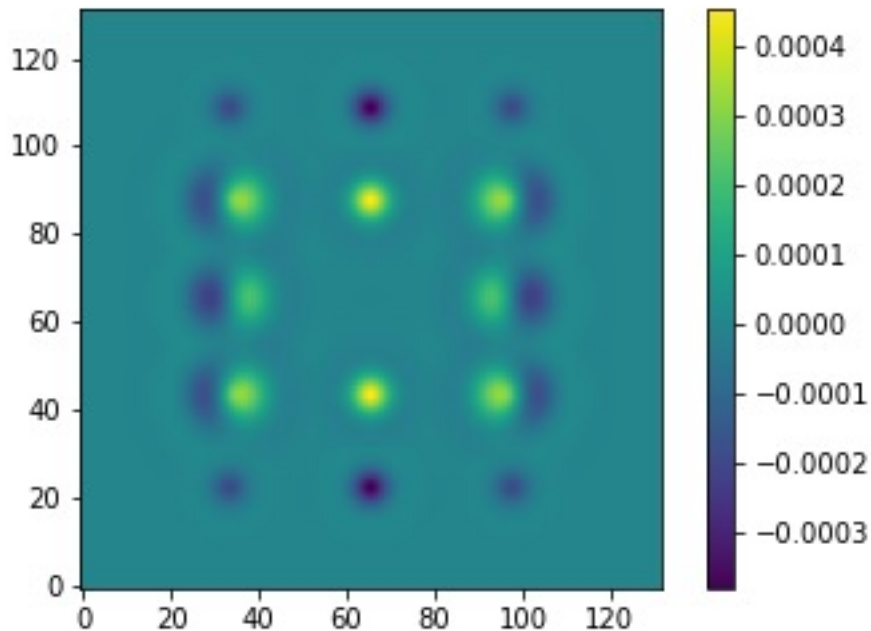
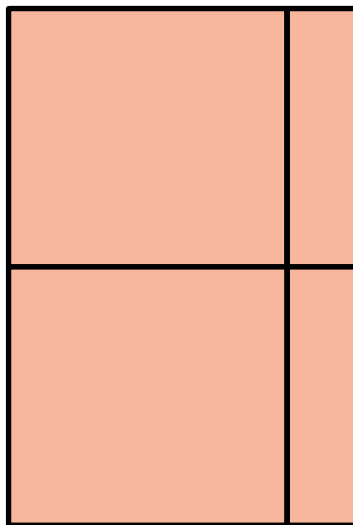


top-bottom

left-right

Error without sync

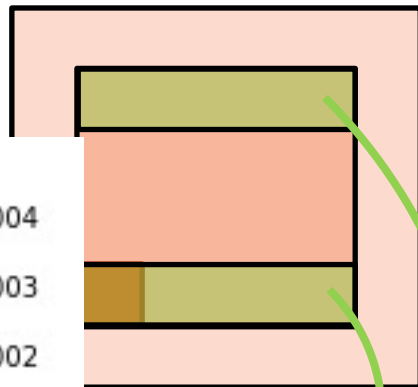
global domain



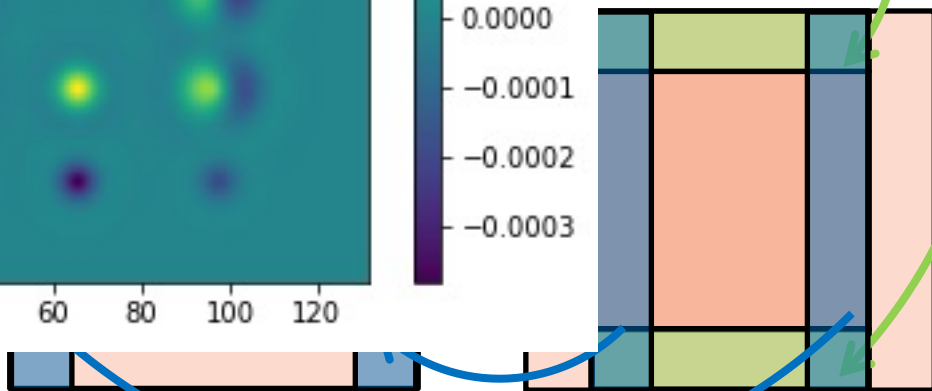
rank = 2



rank = 3



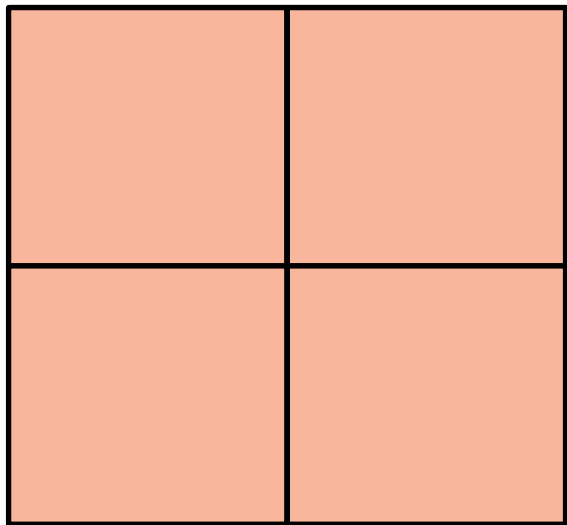
$k = 1$ top-bottom



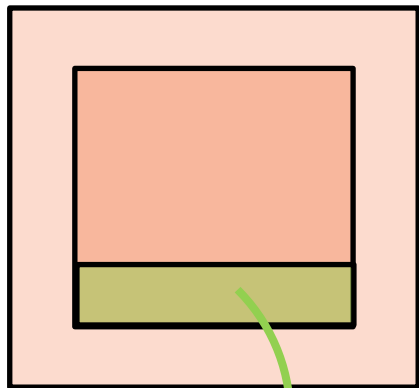
left-right

No-sync strategy

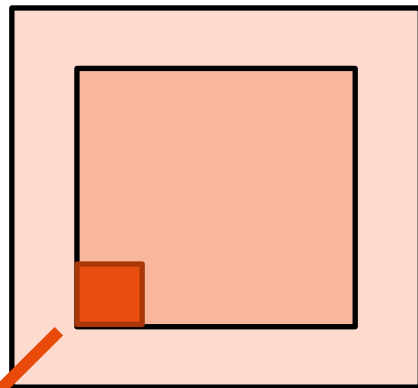
global domain



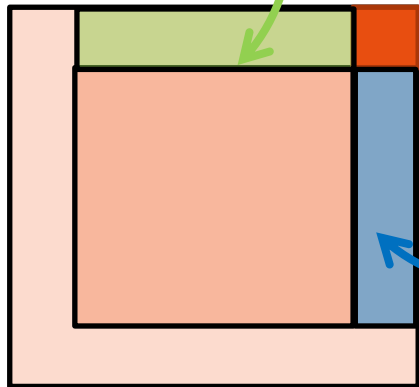
rank = 2



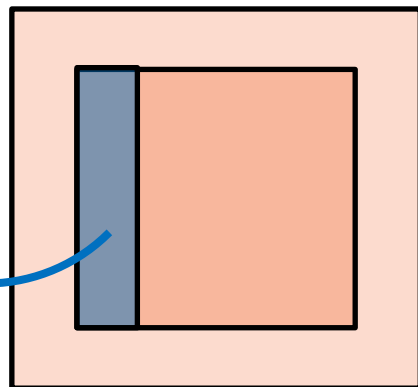
rank = 3



rank = 0



rank = 1



Domain Decomposition in Atmospheric Models

