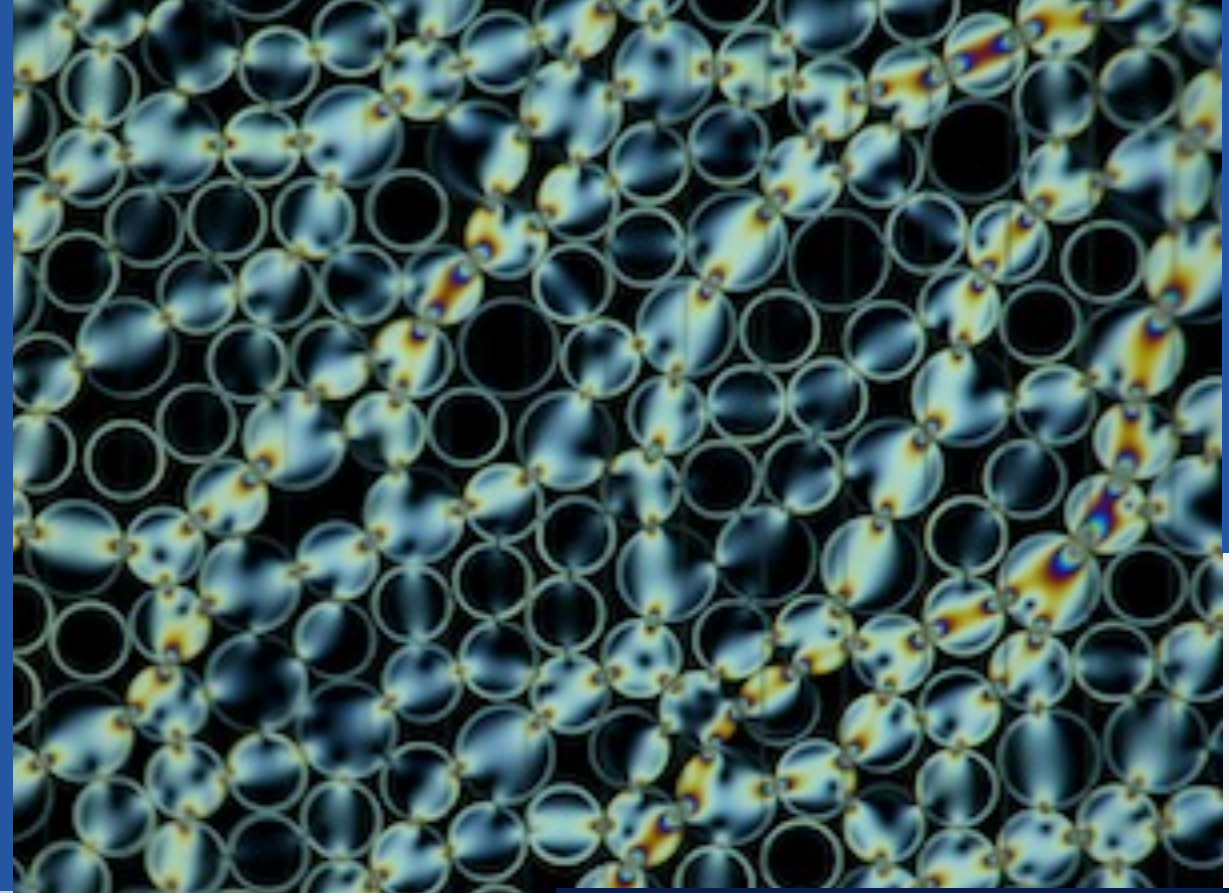
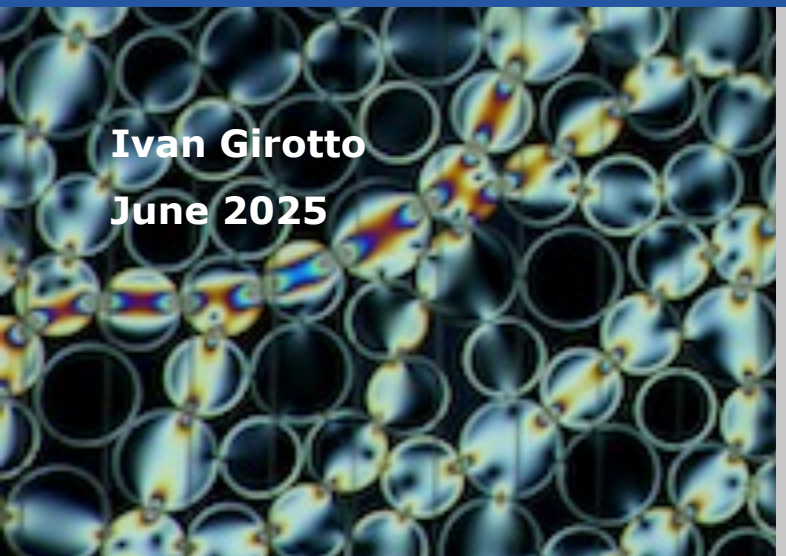
Distributed Multi-GPUs



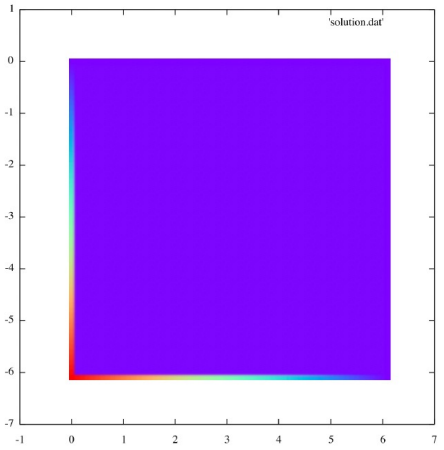


Ivan Girotto

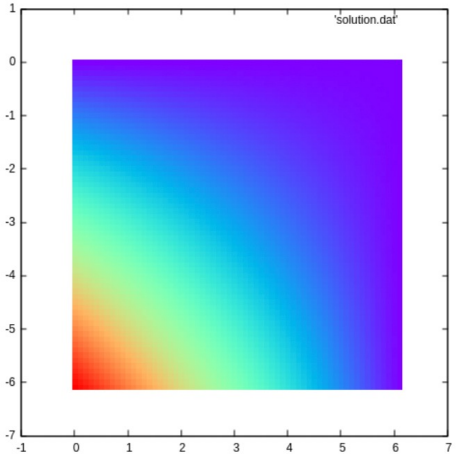
June 2025

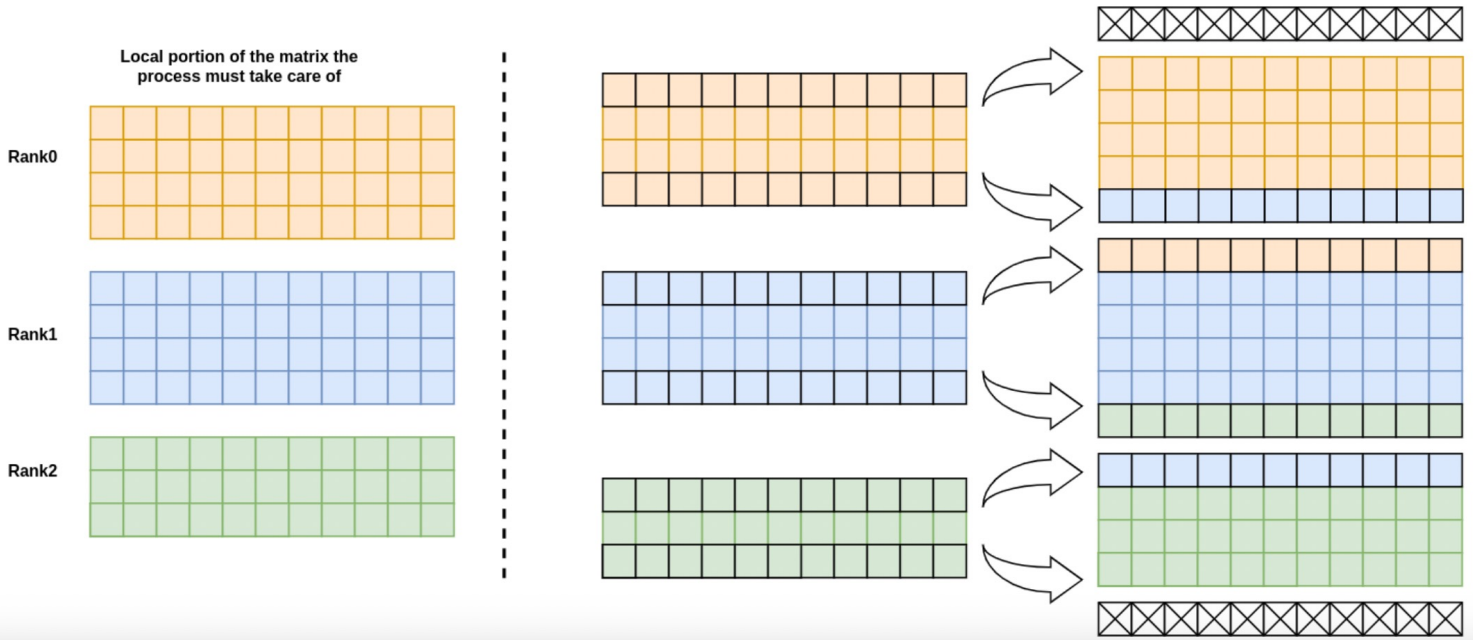
The Abdus Salam International Centre for Theoretical Physics

Parallel Efficiency: Jacoby example



$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$





$$V_{i,j}^{new} = 0.25(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1})$$

0	0	0	0	0	0	0	0	0	0	0
10										0
20										0
30			$V_{i+1,j}$							0
40		$V_{i,j-1}$	$V_{i,j}$	$V_{i,j+1}$						0
50			$V_{i-1,j}$							0
60										0
70										0
80										0
90										0
100	90	80	70	60	50	40	30	20	10	0

Figure 1: A diagram of the Jacobi Relaxation for Solving the Laplace's Equation on an evenly spaced 9x9 grid with the boundary conditions outlined in the text above.

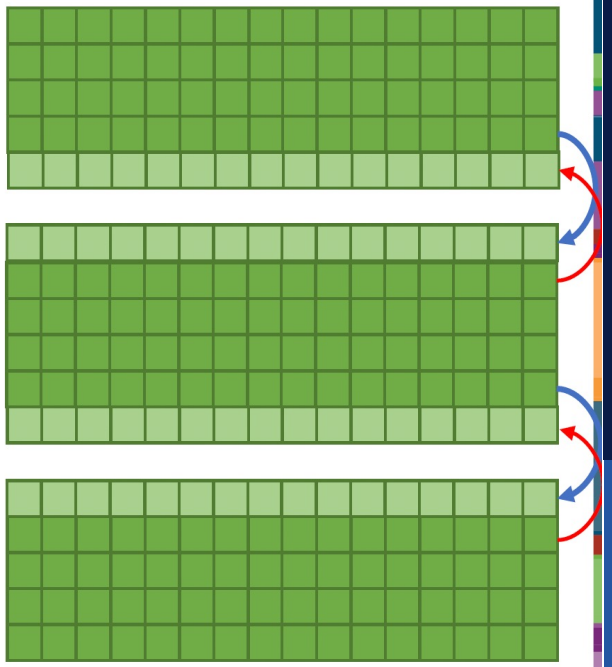
Jacobi example: Top and Bottom Boundaries

bottom neighbor

```
MPI_Sendrecv(a_new_d+offset_last_row, m-2, MPI_DOUBLE,
             b_nb, 1,a_new_d+offset_top_boundary, m-2,
             MPI_DOUBLE, t_nb, 1, MPI_COMM_WORLD,
             MPI_STATUS_IGNORE);
```

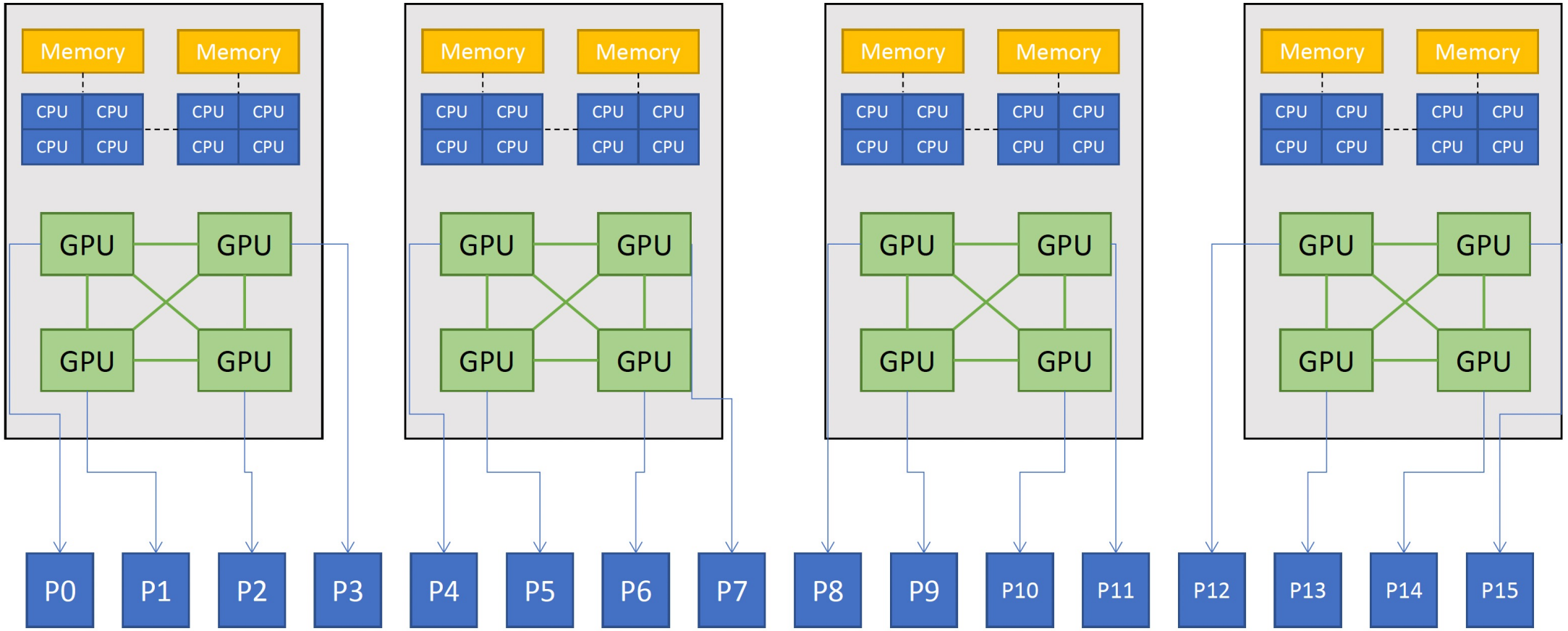
top neighbor

```
MPI_Sendrecv(a_new_d+offset_first_row, m-2, MPI_DOUBLE,
             t_nb,0,a_new_d+offset_bottom_boundary, m-2,
             MPI_DOUBLE,b_nb,  0, MPI_COMM_WORLD,
             MPI_STATUS_IGNORE);
```



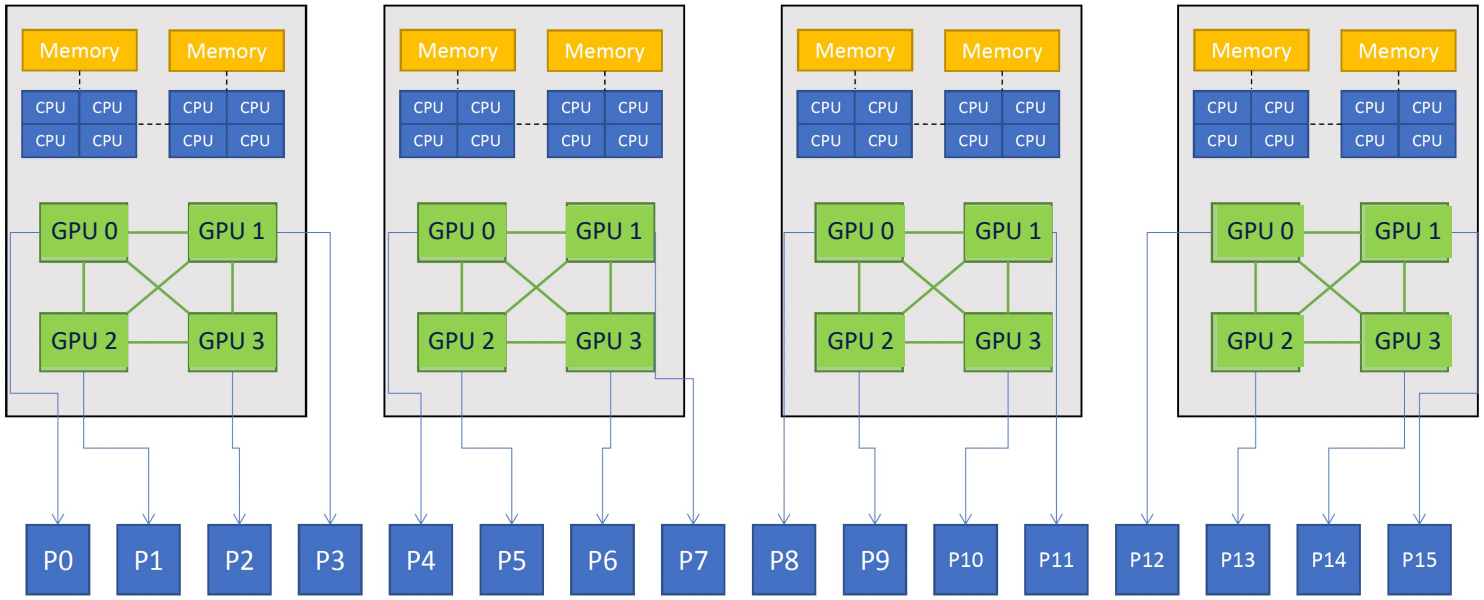
Process Mapping on Multi GPU Systems

- One GPU per Process



Process Mapping on Multi GPU Systems

- One GPU per Process




```
#ifdef _OPENACC
        const int num_dev = acc_get_num_devices(acc_device_nvidia);  //#GPUs
        const int dev_id = rank % num_dev;

        acc_set_device_num(dev_id,acc_device_nvidia); // assign GPU to one MPI process
        acc_init(acc_device_nvidia); // OpenACC call

        printf("Rank %d/%d, device %d/%d\n",rank,size,dev_id,num_dev);
#endif
```

EXAMPLE JACOBI

Top/Bottom Halo

without CUDA-aware MPI

OpenACC

```
#pragma acc update host(u_new[offset_first_row:m-2],u_new[offset_last_row:m-2])
MPI_Sendrecv(u_new+offset_first_row, m-2, MPI_DOUBLE, t_nb, 0,
             u_new+offset_bottom_boundary, m-2, MPI_DOUBLE, b_nb, 0,
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);
MPI_Sendrecv(u_new+offset_last_row, m-2, MPI_DOUBLE, b_nb, 1,
             u_new+offset_top_boundary, m-2, MPI_DOUBLE, t_nb, 1,
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);
#pragma acc update device(u_new[offset_top_boundary:m-2],u_new[offset_bottom_boundary:m-2])
```

CUDA

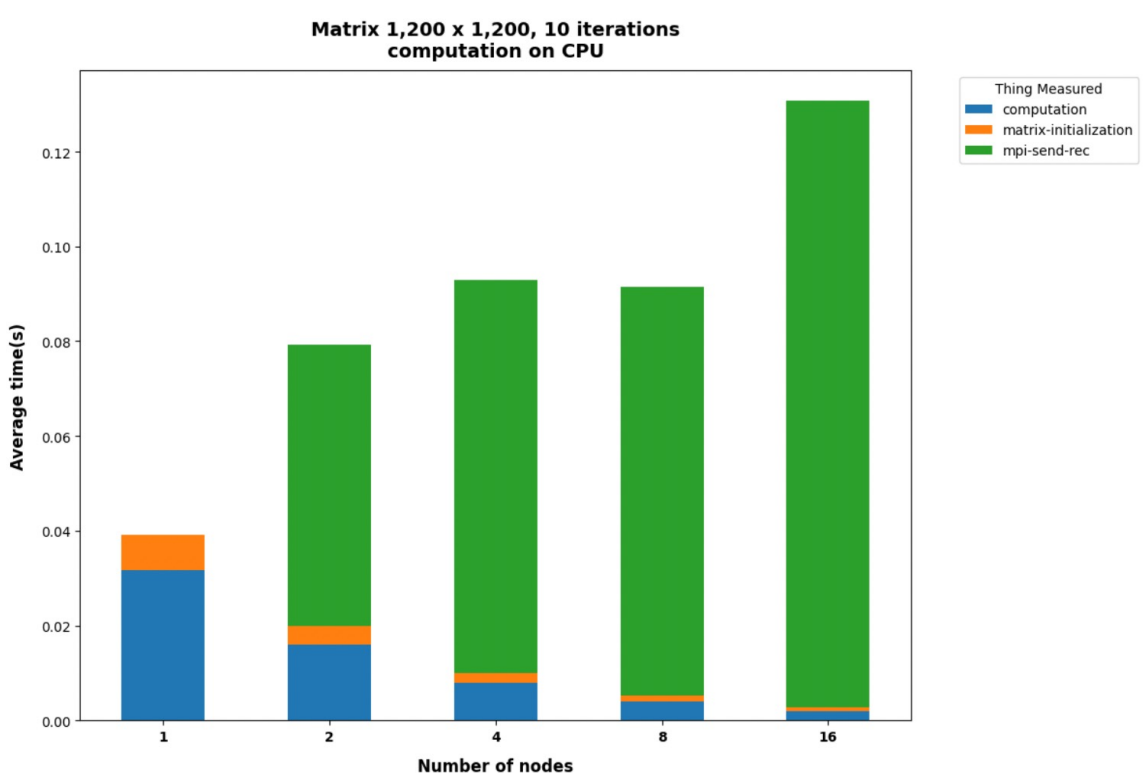
```
//send to bottom and receive from top top bottom omitted
cudaMemcpy(  u_new+offset_first_row,
             u_new_d+offset_first_row, (m-2)*sizeof(double), cudaMemcpyDeviceToHost);
MPI_Sendrecv(u_new+offset_first_row, m-2, MPI_DOUBLE, t_nb, 0,
             u_new+offset_bottom_boundary, m-2, MPI_DOUBLE, b_nb, 0,
             MPI_COMM_WORLD, MPI_STATUS_IGNORE);
cudaMemcpy(  u_new_d+offset_bottom_boundary,
             u_new+offset_bottom_boundary, (m-2)*sizeof(double), cudaMemcpyDeviceToHost);
```

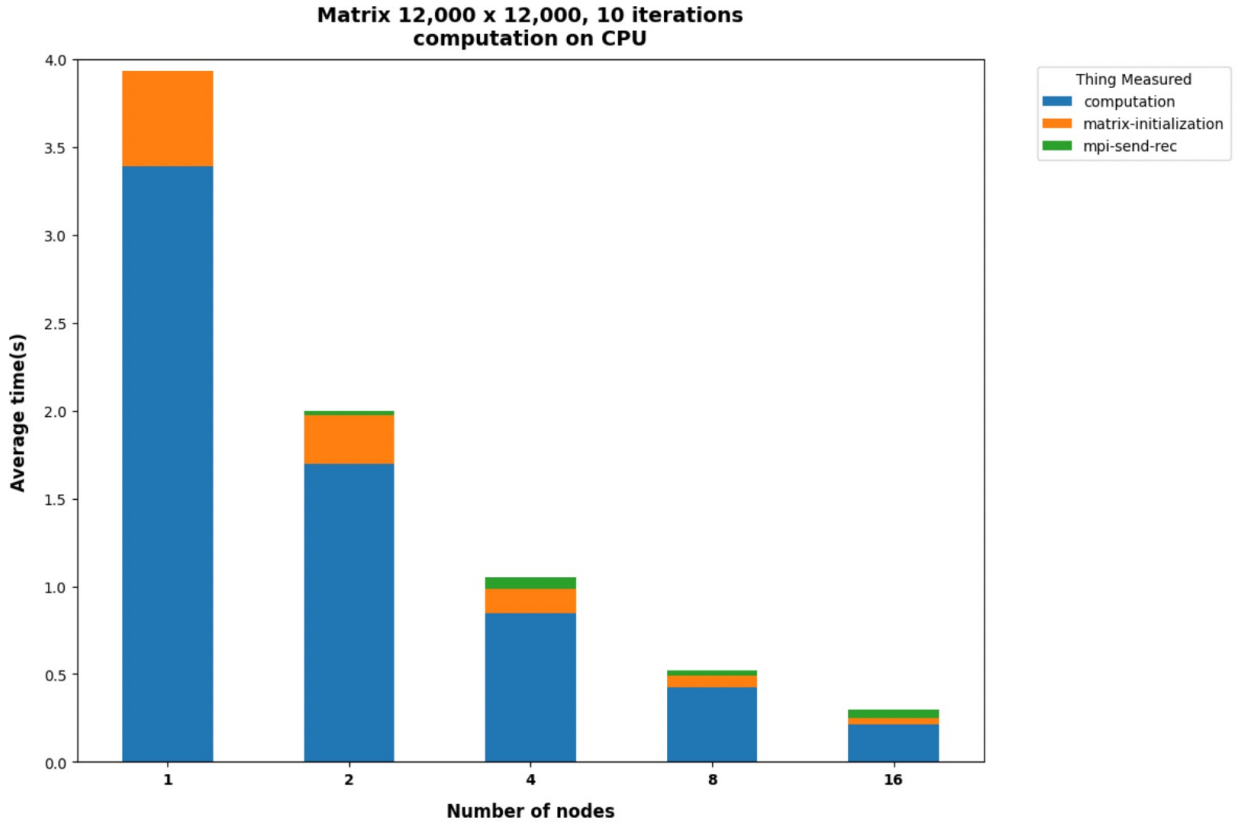
With CPU-aware MPI

- The communication is now performed directly from GPU2GPU
- Highly optimized intra-node

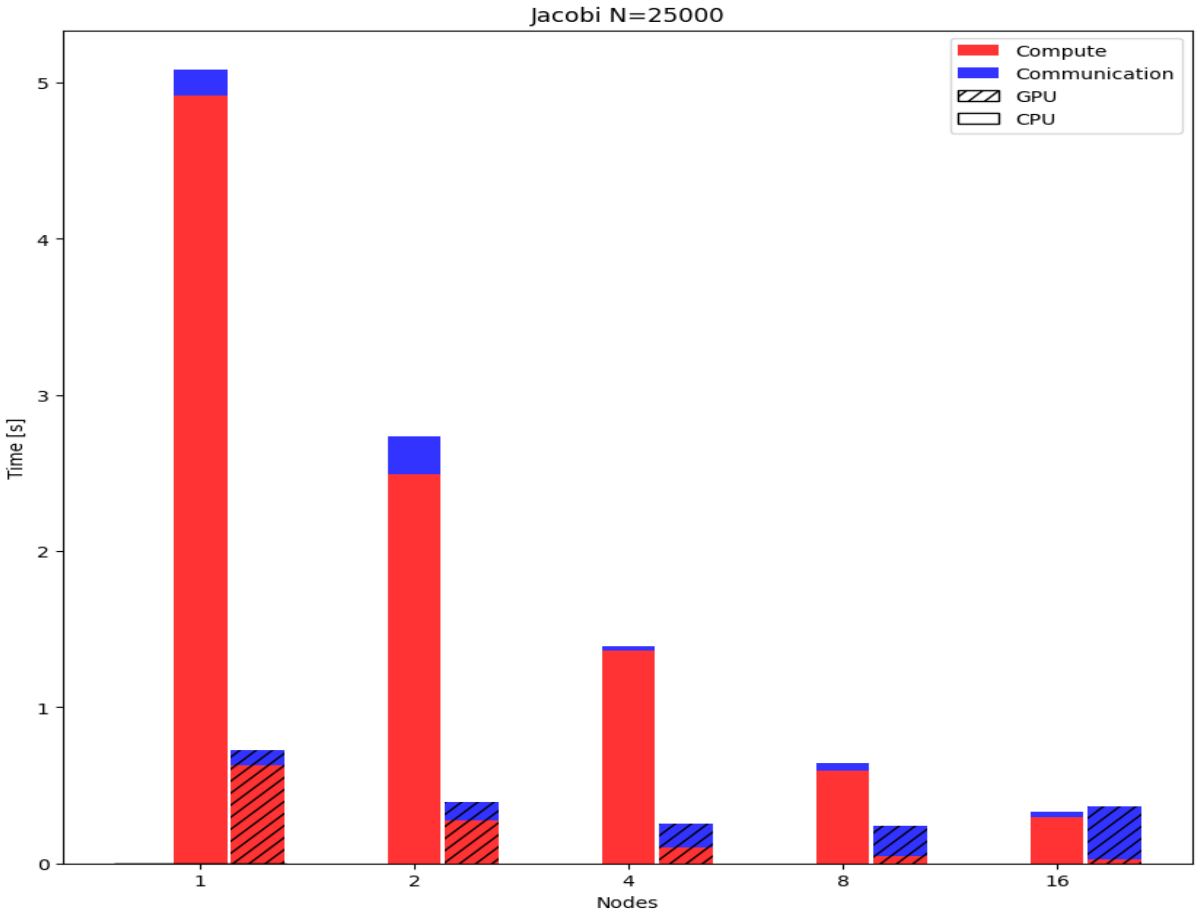
```
#pragma acc host_data use_device(matrix)
{
  MPI_Sendrecv( matrix+ width, width, MPI_DOUBLE, send_to, 0, matrix+width*(nloc+1), dimension+2,
                MPI_DOUBLE,  recv_from, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE );

  MPI_Sendrecv( matrix+width*nloc, width, MPI_DOUBLE, recv_from, 0, matrix, width,
                MPI_DOUBLE, send_to, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE );
}
```

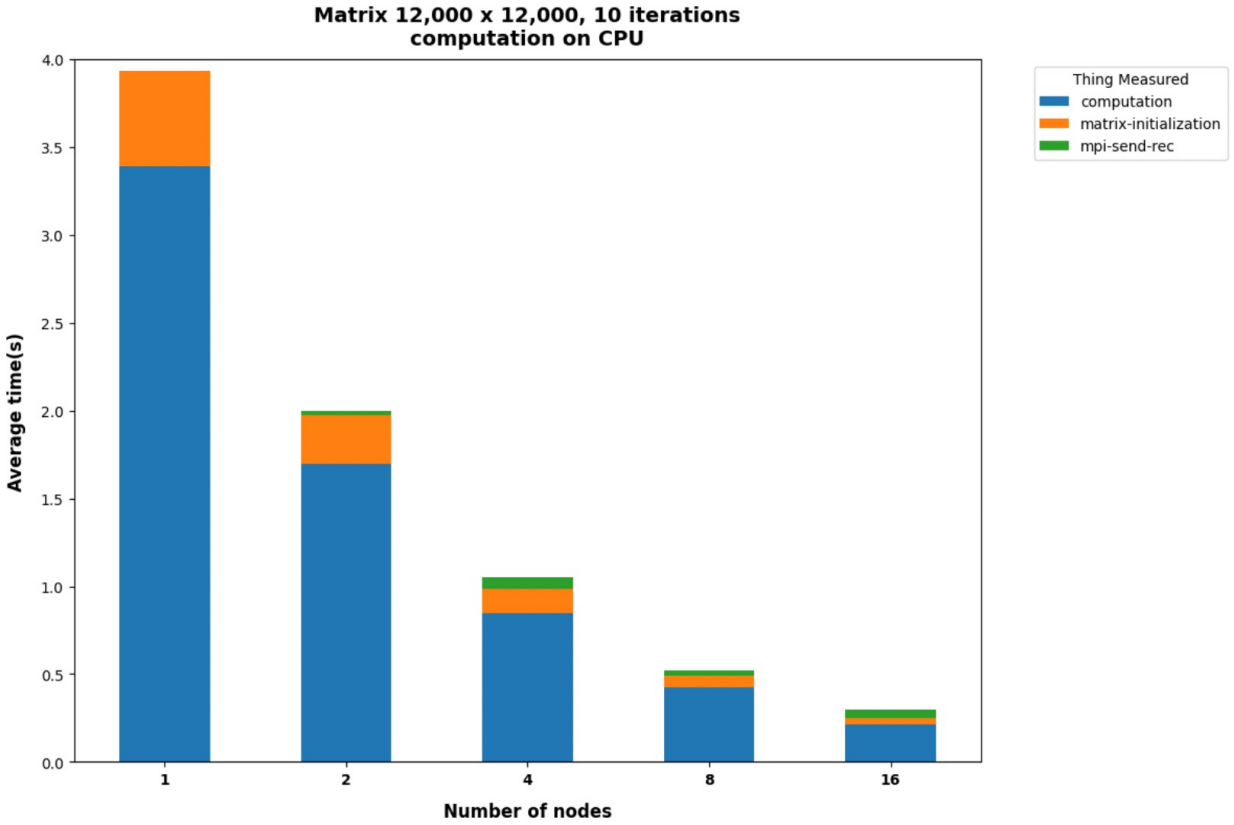
Parallel Efficiency: Jacoby example

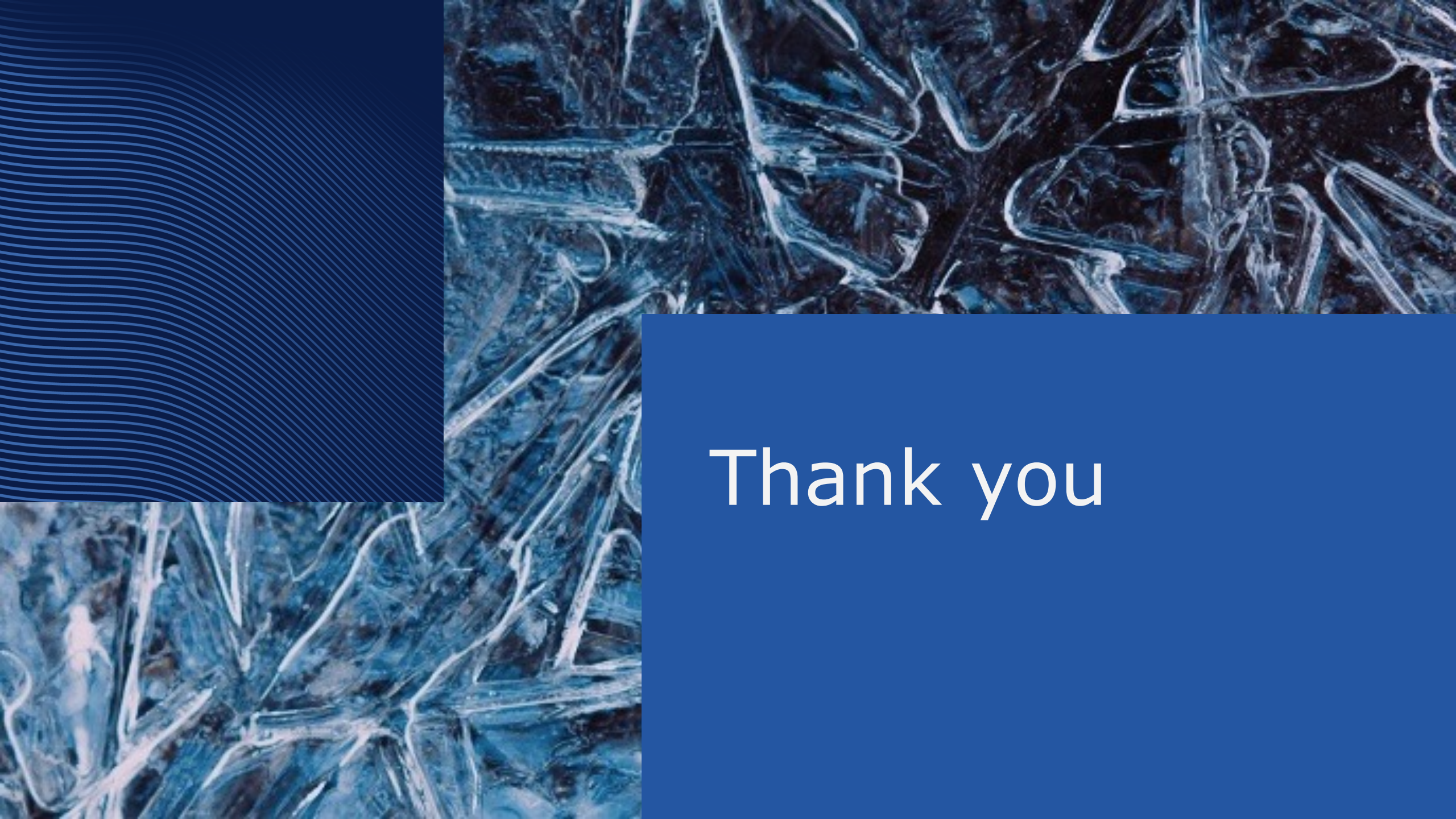




Parallel Efficiency: Jacoby example







Thank you