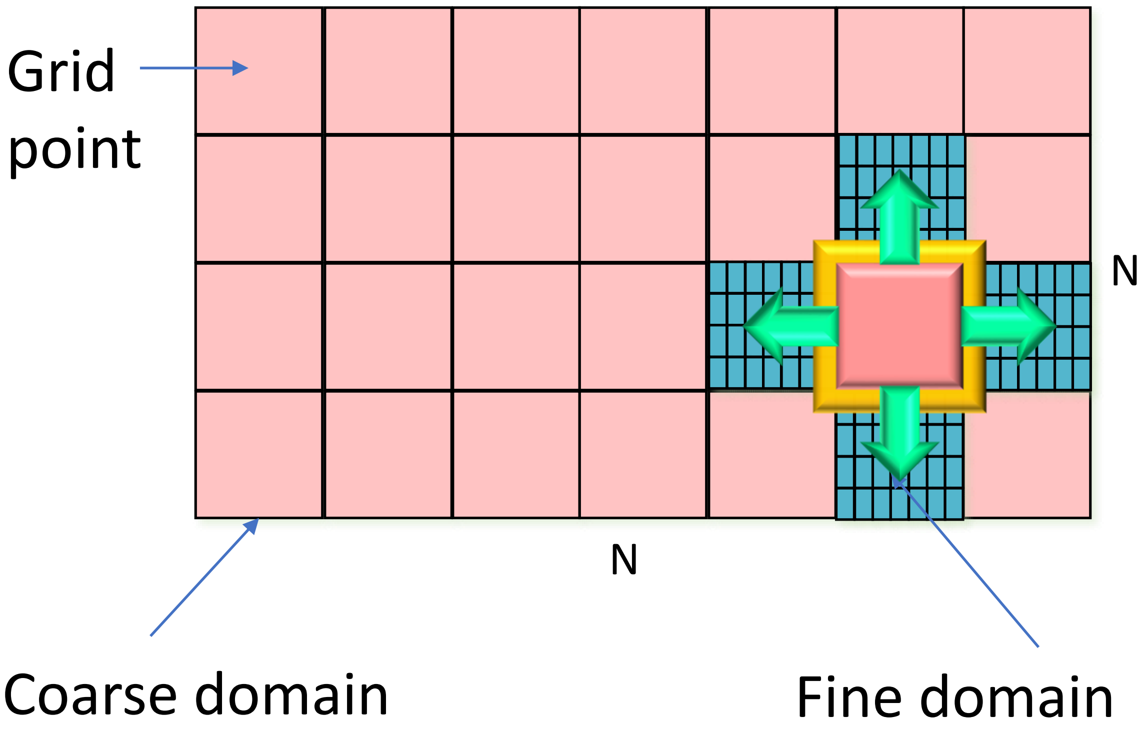
Parallelization-II

Jan 29, 2019

Domain Refinement



2D domain

Halo exchange

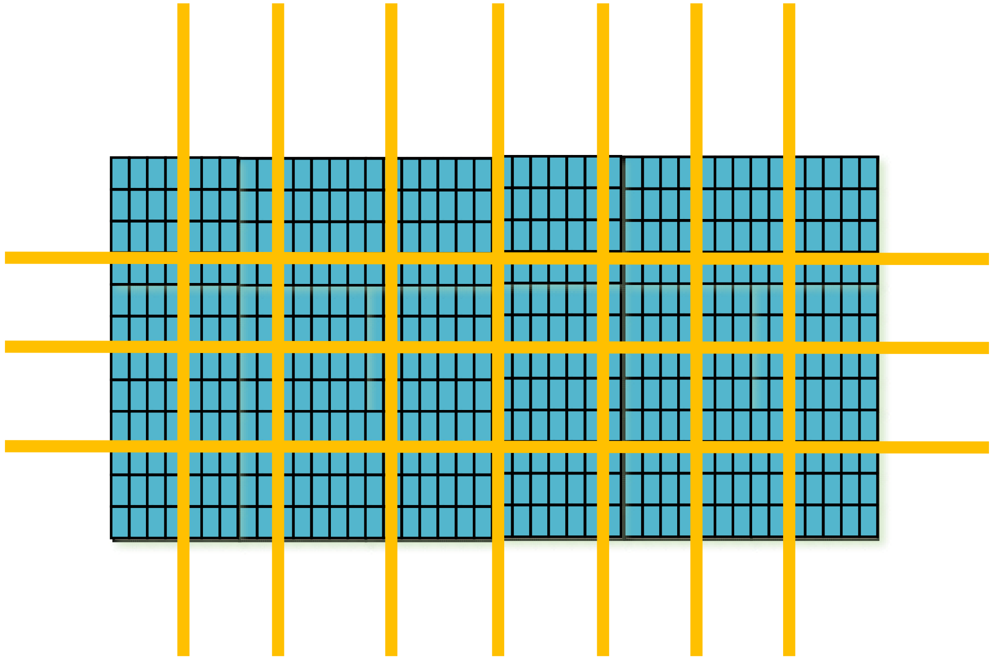
- Each cell has some ghost regions
- Communication with four neighbors

communication to computation ratio
(2D decomposition) = $\frac{4N/\sqrt{P}}{N^2/P}$

communication to computation ratio
(1D decomposition) = $\frac{2N}{N^2/P}$

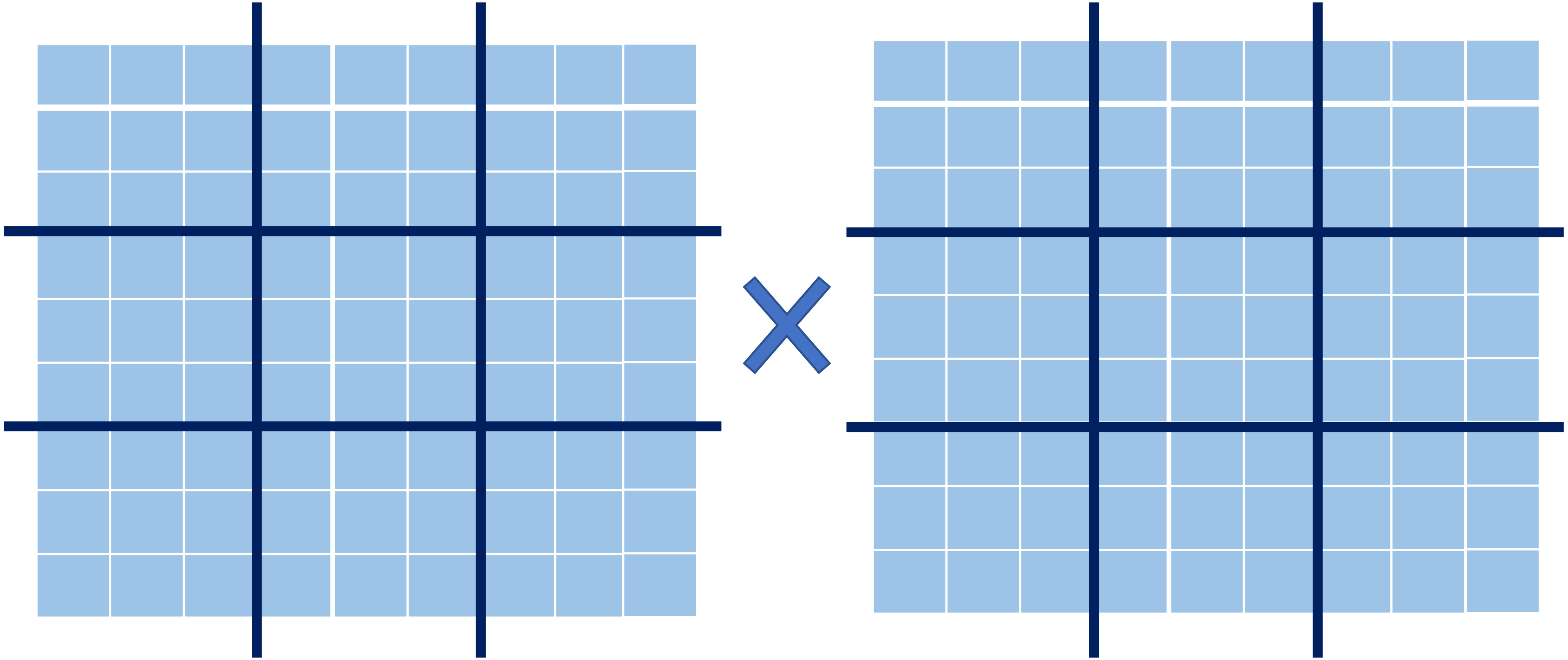
Q: Which is better?

Over-decomposition

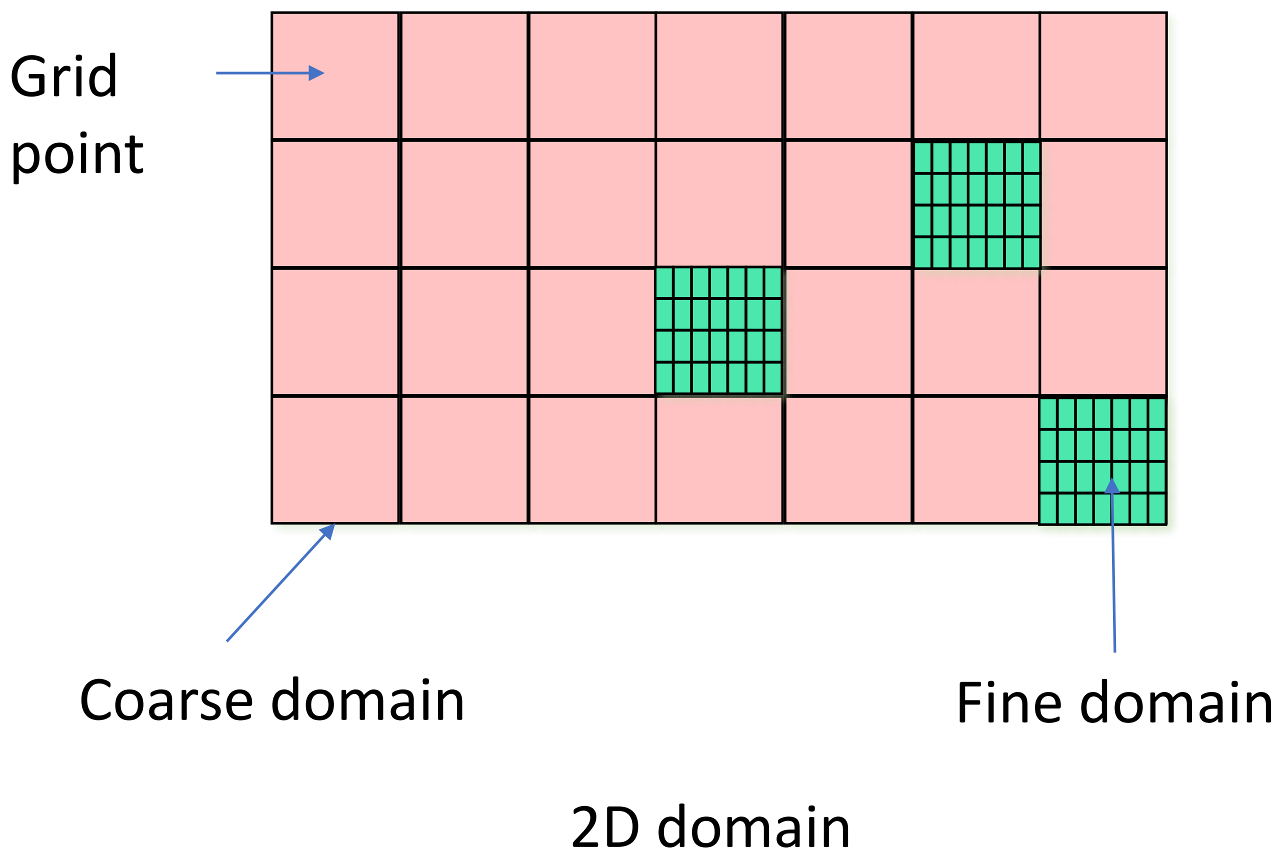


- There must always be sufficient work for a process!
- Balance between reducing communication volume and reducing computations/process such that processes do not idle
- Communication to computation ratio

Parallelization – Matrix-Matrix Multiplication

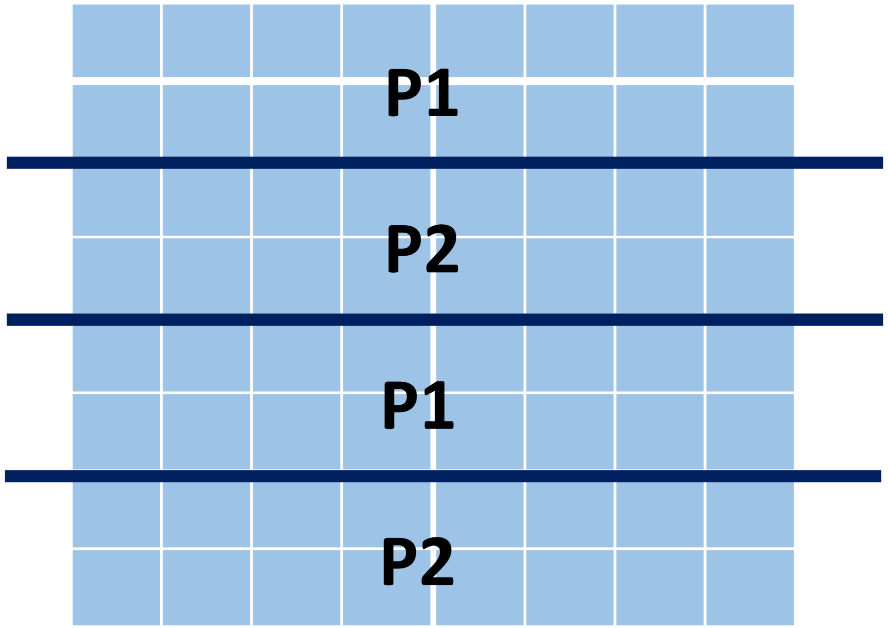


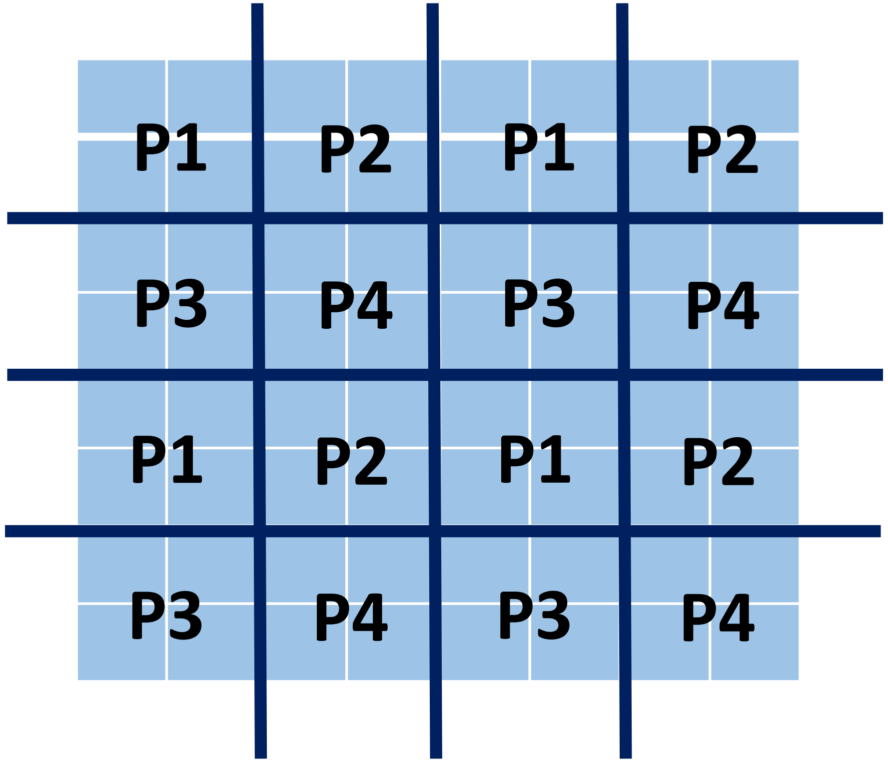
Adaptive Mesh Refinement



Q: Issue?

Block-cyclic Distribution





What may be the benefit/use case?

Performance Issues – Summary

Decomposition

Assignment

Orchestration

Mapping

Load imbalance

Division of work

Synchronization

Communication volume

Communication structure

Data distribution

Scalability

Strong scaling

- Fixed problem size
- Increase number of processes
- Efficiency decreases, in general – why?

Can you keep efficiency fixed?

Weak scaling

- Fixed problem size per process
- Increase number of processes
- Increase problem size

MPI Process Topology

0 (0,0)	**1 (0,1)**	**2 (0,2)**	**3 (0,3)**
4 (1,0)	**5 (1,1)**	**6 (1,2)**	**7 (1,3)**

Cartesian topology

int MPI_Cart_create (MPI_Comm comm_old, int ndims, int *dims, int *periodic, int reorder, MPI_Comm *comm_cart)

MPI_Cartdim_get (comm, ndims)

MPI_Cart_rank (comm, coords, rank)

MPI_Cart_coords (comm, rank, maxdims, coords)

MPI_Cart_shift (comm, dir, disp, source, dest)

MPI_Cart_sub (comm, dims, newcomm)

Code Snippet

```
ndims = 2
dim[0] = 3, dim[1] = 3
wrap_around[0] = 0, wrap_around[1] = 0
reorder = 0
MPI_Cart_create(MPI_COMM_WORLD, ndims, dims, wrap_around,
reorder, &comm2D)

free_coords[0] = 0, free_coords[1] = 1   //column dimension
MPI_Cart_sub (comm2D, free_coords, &comm1D_row)
free_coords[0] = 1, free_coords[1] = 0
MPI_Cart_sub (comm2D, free_coords, &comm1D_col)

MPI_Reduce(&val, &rowmax, 1, MPI_INT, MPI_MAX, 0, comm1D_row);
MPI_Reduce(&rowmax, &max, 1, MPI_INT, MPI_MAX, 0, comm1D_col);
```

0 (0,0) val=78	**1 (0,1)** 72	**2 (0,2)** 70
3 (1,0) 81	**4 (1,1)** 77	**5 (1,2)** 80
6 (2,0) 77	**7 (2,1)** 78	**8 (2,2)** 75

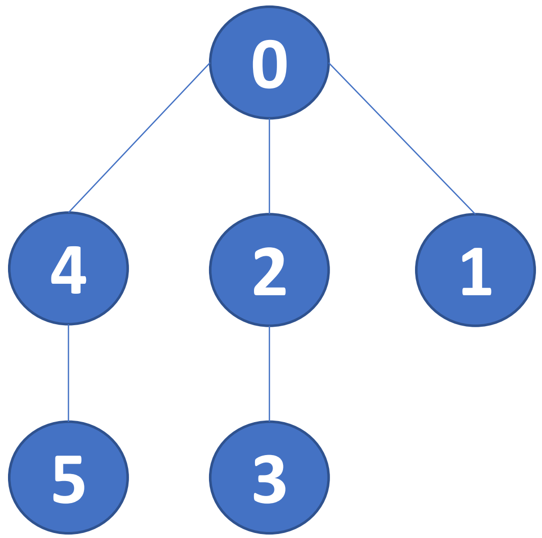
Graph Topology

MPI_Graph_create (comm_old, nnodes, index, edges, reorder, comm_graph)

MPI_Graphdims_get (comm, nnodes, nedges)

MPI_Graph_get(comm, maxindex, maxedges, index, edges)

MPI_Graph_neighbors_count(comm, rank, nneighbors)

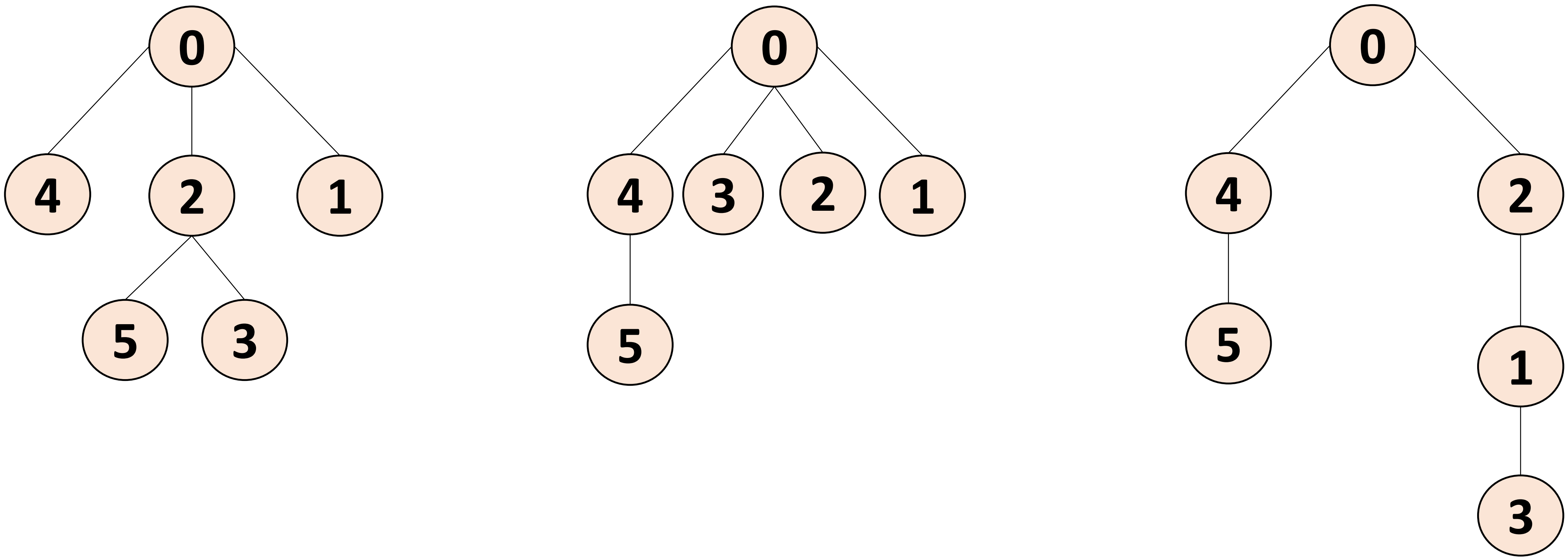


nnodes = 6

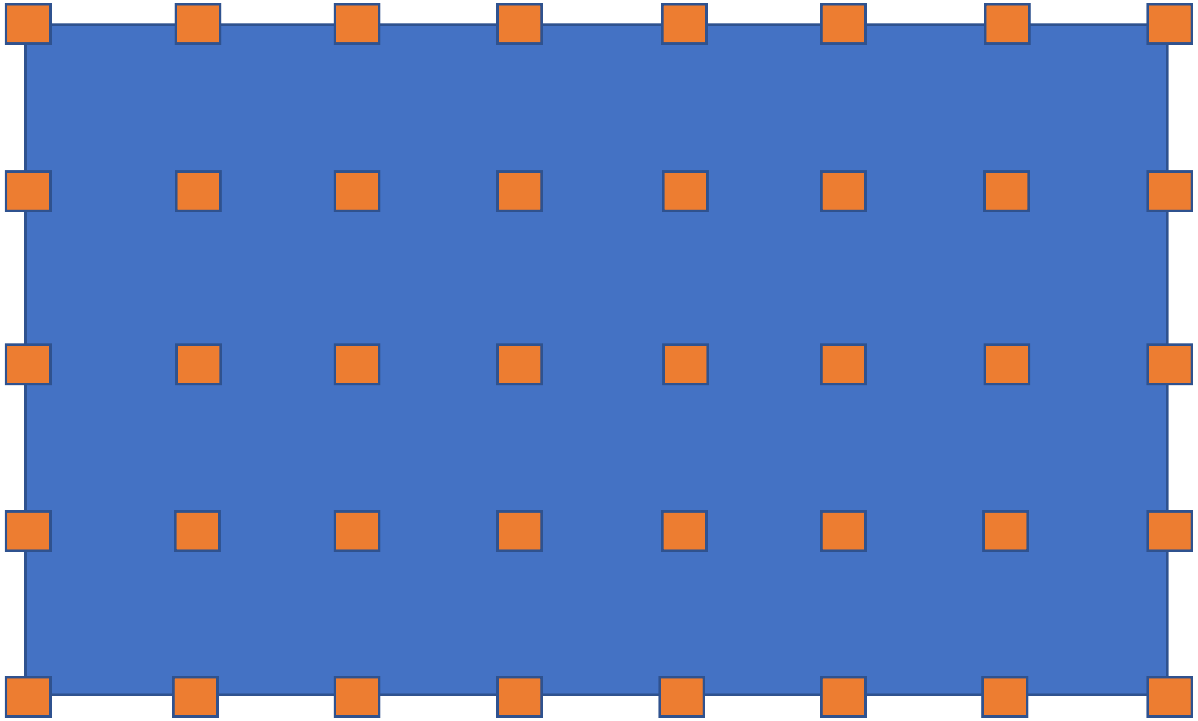
index = 3,4,6,7,9,10

edges = 1,2,4,0,0,3,2,0,5,4

Design Decisions



Mathematical Modeling

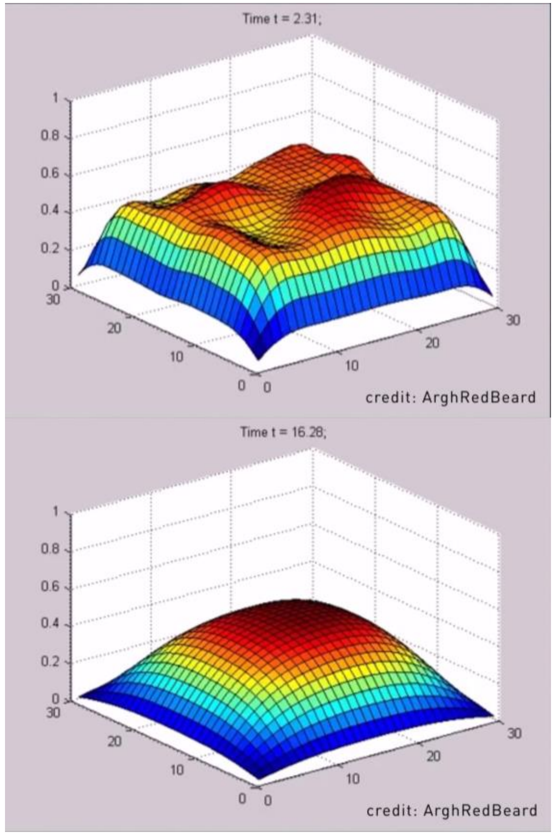


- Many real-world problems formulated using ordinary and partial differential equations
- Discretization of domain
- Finer the grid, more accurate is the solution

Partial Differential Equations – Example

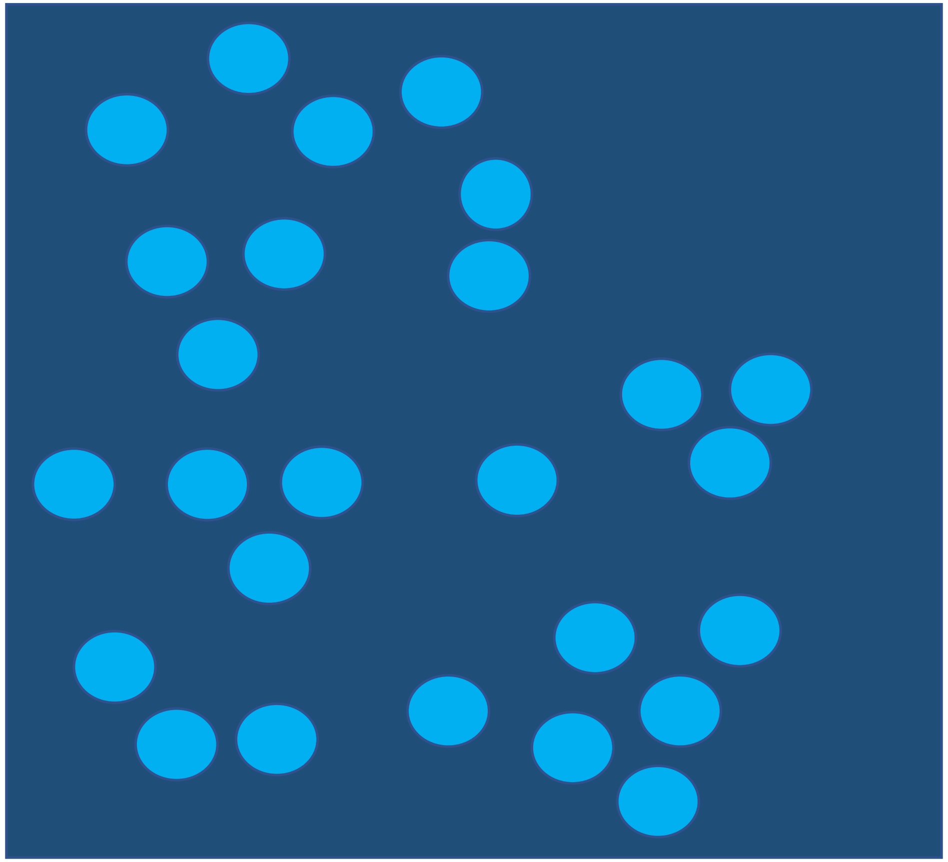
Heat equation

$$\Delta f_t (x, y) = \frac{\partial}{\partial x} f_t (x, y)$$



Visualization of solution of heat equation

N-body Simulation



Problem

- N bodies exert force on each other
- Model positions of the particles over time

Applications

- Evolution of the universe
- Crack propagation in a material

HPC2010

- 369 in top500 in June 2010
- 376 nodes – 368 compute nodes
- Intel Xeon (8 cores per node), later some more nodes were added
- Connected by Infiniband
- Home and scratch file system
- PBS scheduler
- Submit to “courses” queue

Assignment 2

On HPC2010

- Modify 1.1 and 1.2 (use many-to-one mapping)
- Ping-pong benchmark using blocking sends and receives
- Implement MPI_Reduce (MPI_PROD) functionality using point-to-point (you may use any algorithm) and compare the performance with MPI_Reduce function