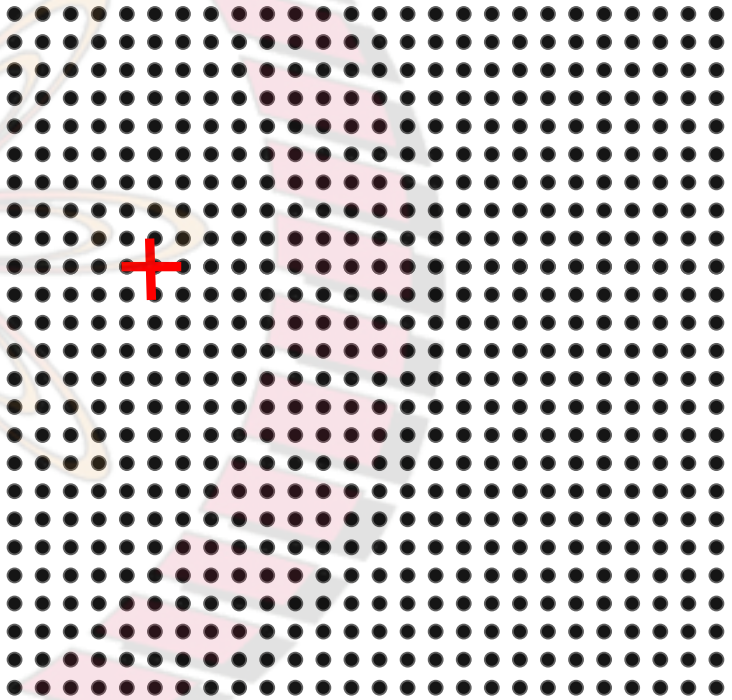
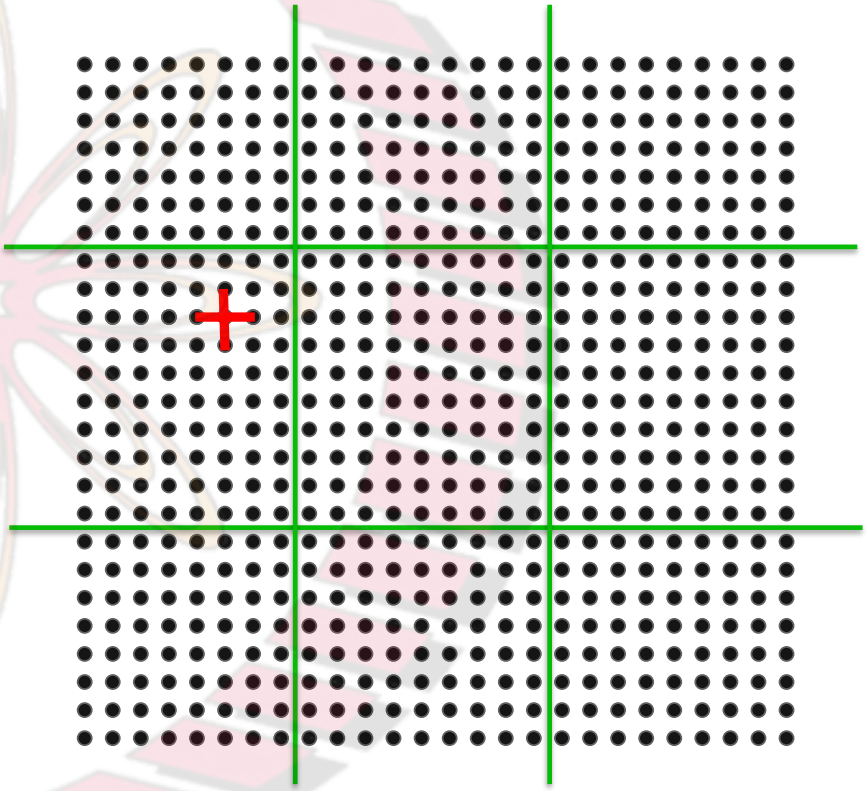
The Global Data Structure

- Each circle is a mesh point
- Difference equation evaluated at each point involves the four neighbors
- The red “plus” is called the method’s stencil
- Good numerical algorithms form a matrix equation Au=f; solving this requires computing Bv, where B is a matrix derived from A. These evaluations involve computations with the neighbors on the mesh.



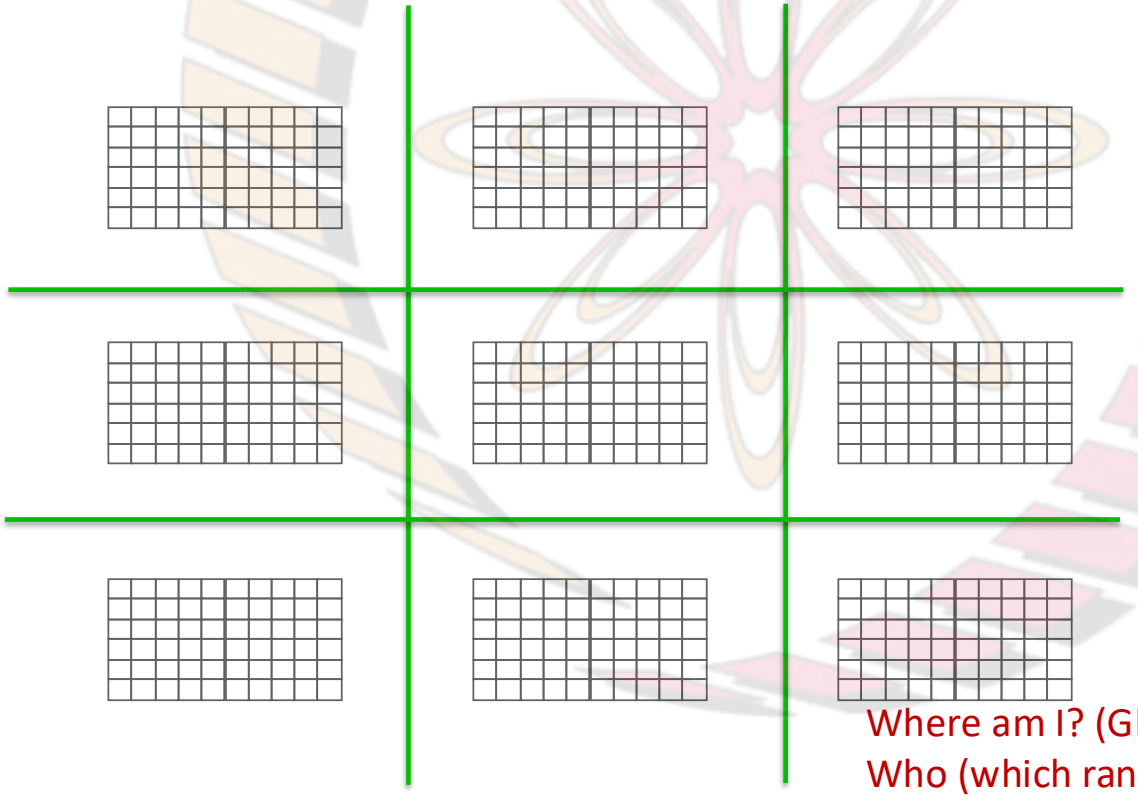
The Global Data Structure

- Each circle is a mesh point
- Difference equation evaluated at each point involves the four neighbors
- The red “plus” is called the method’s stencil
- Good numerical algorithms form a matrix equation Au=f; solving this requires computing Bv, where B is a matrix derived from A. These evaluations involve computations with the neighbors on the mesh.
- Decompose mesh into equal sized (work) pieces



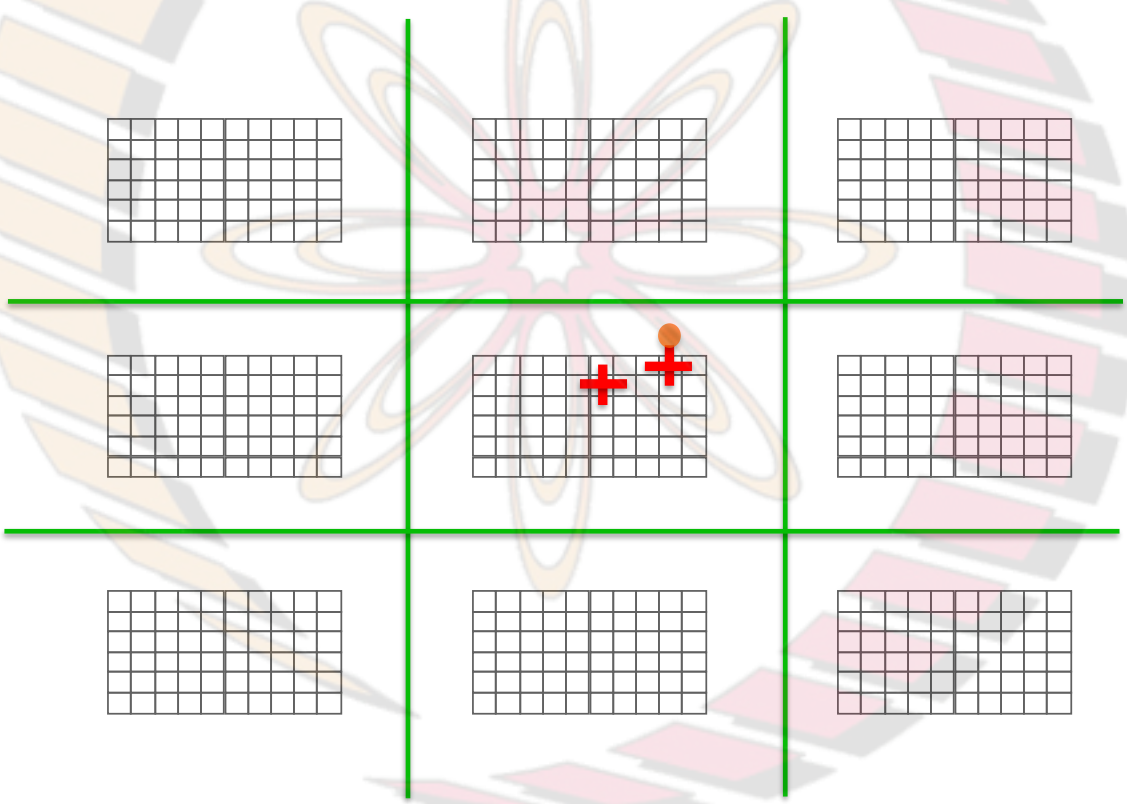
Domain Decompositioin

Parameters for domain decomposition:
N = Size of the edge of the global problem domain (assuming square)
PX, PY = Number of processes in X and Y dimension
N % PX == 0, N % PY == 0



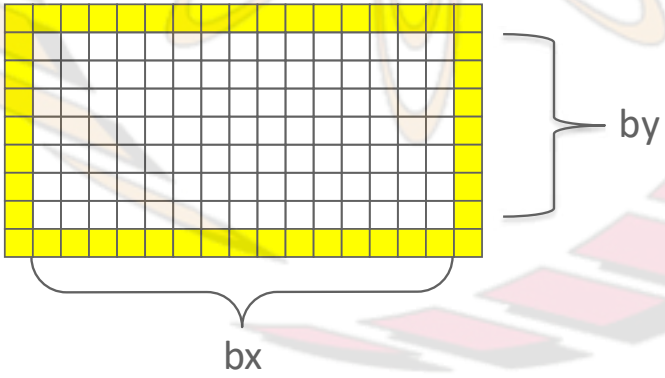
Where am I? (Global offset)
Who (which ranks) are my neigbhors?
Use MPI_PROC_NULL for boundary

Necessary Data Transfers



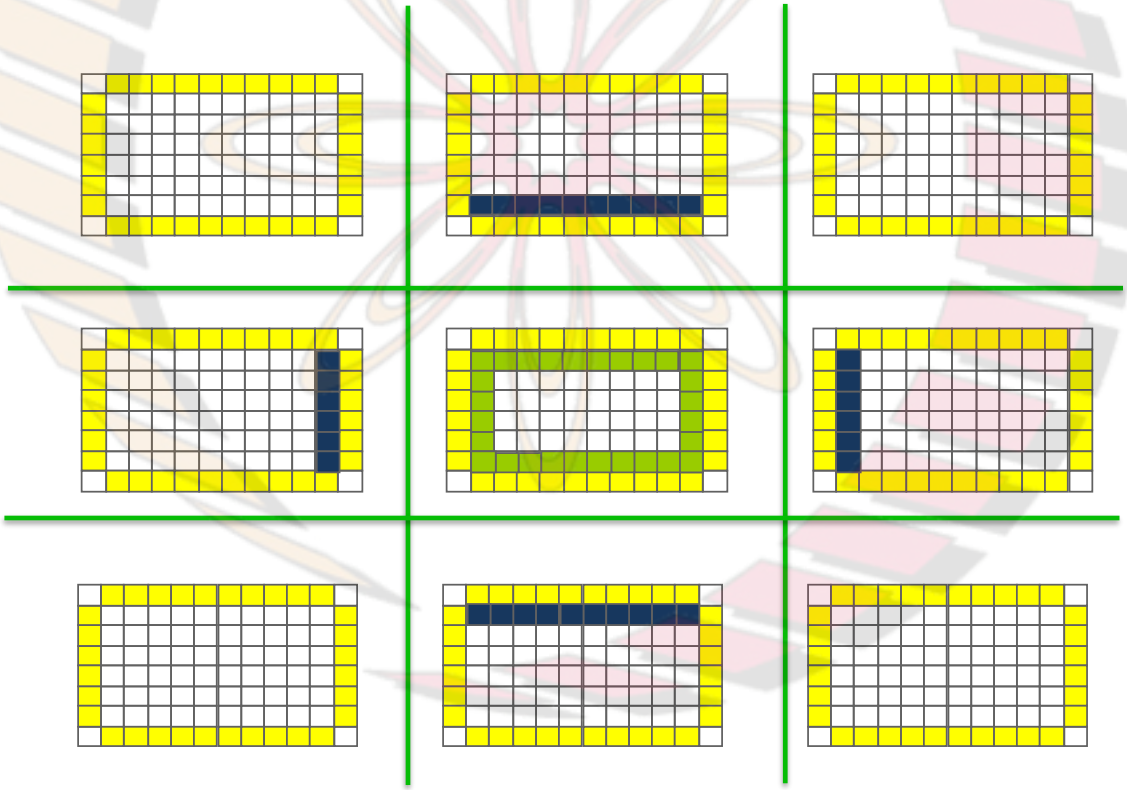
The Local Data Structure

- Each process has its local “patch” of the global array
  - “bx” and “by” are the sizes of the local array
  - Always allocate a halo around the patch
  - Array allocated of size (bx+2)x(by+2)



Necessary Data Transfers

- Provide access to remote data through a halo exchange (5 point stencil)

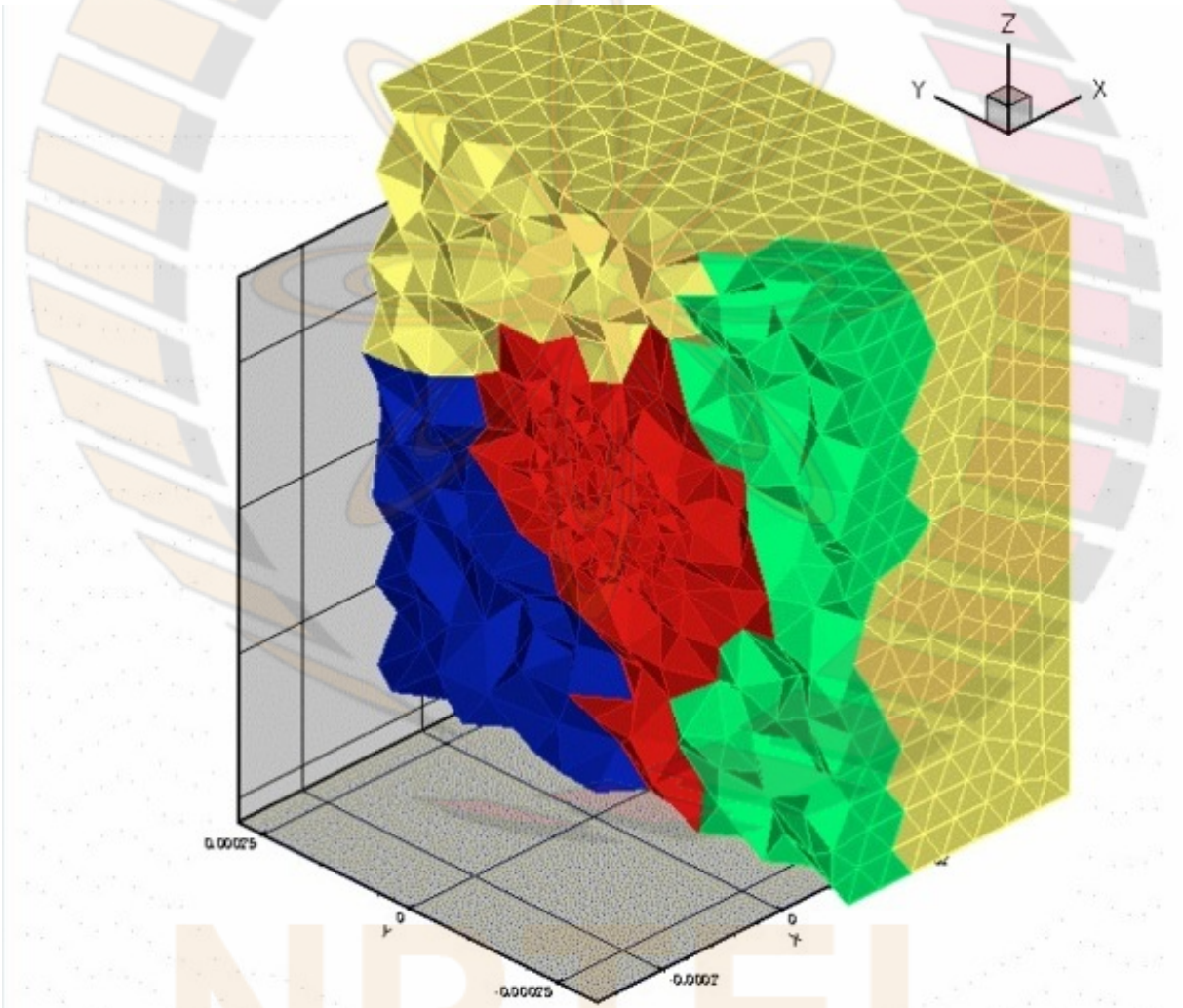


Simple/Predefined Datatypes

- Equivalents exist for all C, C++ and Fortran native datatypes
  - C int → MPI_INT
  - C float → MPI_FLOAT
  - C double → MPI_DOUBLE
  - C uint32_t → MPI_UINT32_T
  - Fortran integer → MPI_INTEGER
- For more complex or user-created datatypes, MPI provides routines to represent them as well
  - Contiguous
  - Vector/Hvector
  - Indexed/Indexed_block/Hindexed/Hindexed_block
  - Struct
  - Some convenience types (e.g., subarray)

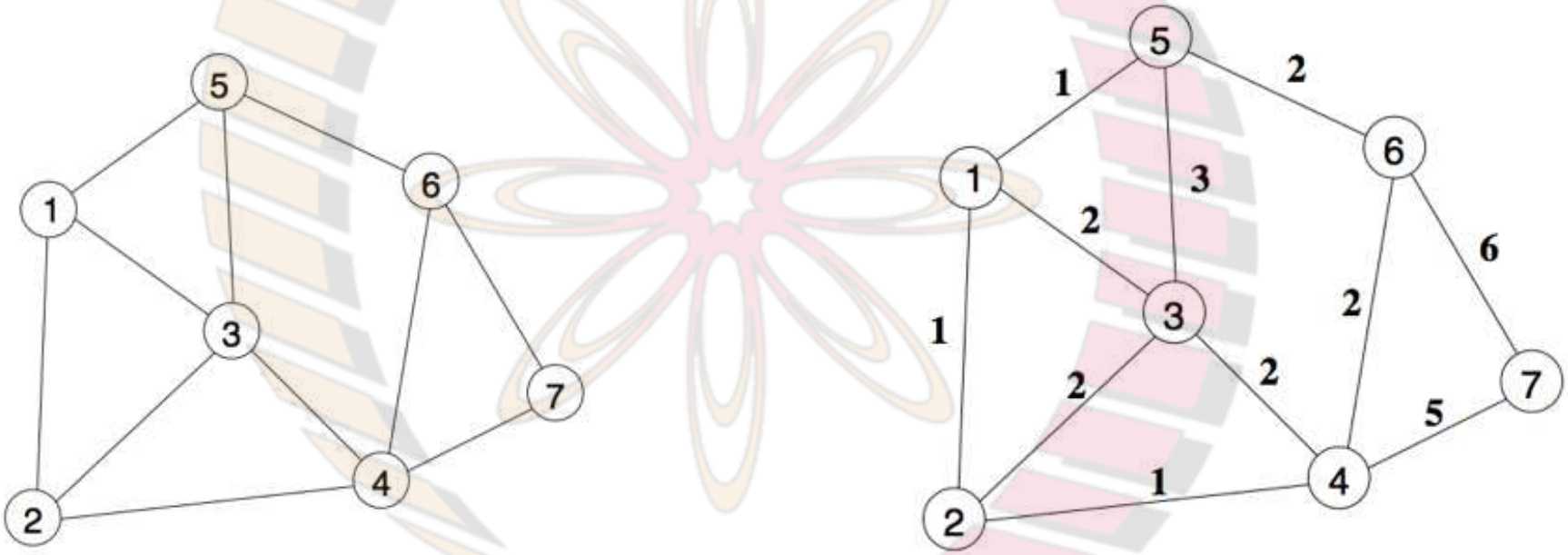
Irregular Domain Decomposition

Domain decomposed by METIS



Metis is a Library for partitioning unstructured graphs / meshes

Mesh representation as Connected graphs



Unweighted graph

Weighted graph

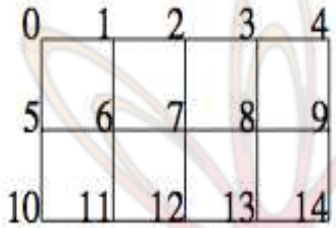
*Images: From METIS manual

CSR format and adjacency graph

"The adjacency structure of the graph is stored using the compressed storage format (CSR). The CSR format is a widely used scheme for storing sparse graphs. In this format the adjacency structure of a graph with **n** vertices and **m** edges is represented using two arrays **xadj** and **adjncy**. The **xadj** array is of size **n + 1** whereas the **adjncy** array is of size **2m** (this is because for each edge between vertices **v** and **u** we actually store both **(v, u) and (u, v))**."*Q

Quote: From METIS manual

CSR format and adjacency graph



(a) A sample graph

xadj	0 2 5 8 11 13 16 20 24 28 31 33 36 39 42 44
adjncy	1 5 0 2 6 1 3 7 2 4 8 3 9 0 6 10 1 5 7 11 2 6 8 12 3 7 9 13 4 8 14 5 11 6 10 12 7 11 13 8 12 14 9 13

(b CSR format

*Images: From METIS manual

Domain decomposed by METIS

Computational domain of a automotive engine cylinder. S. Menon [2011]