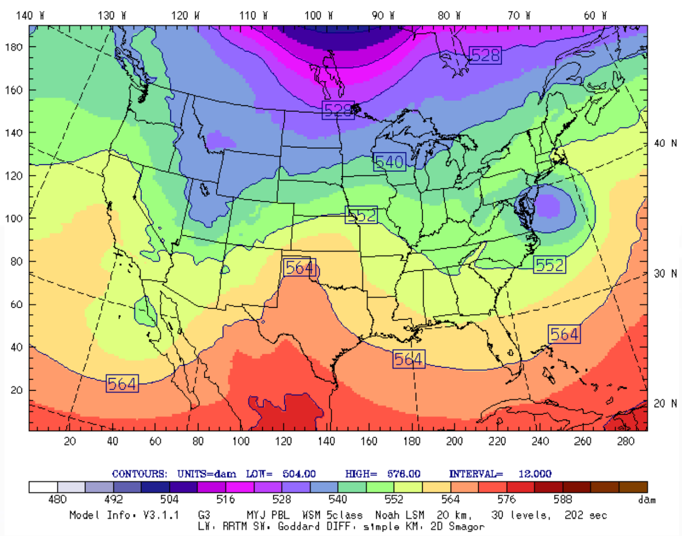
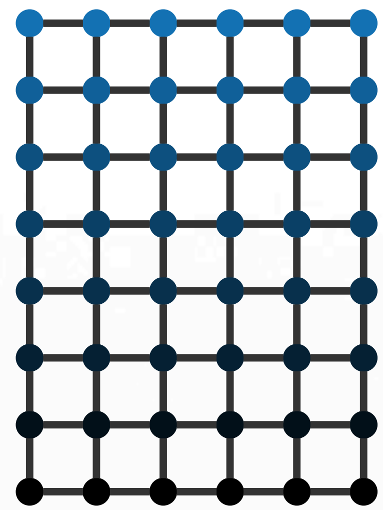
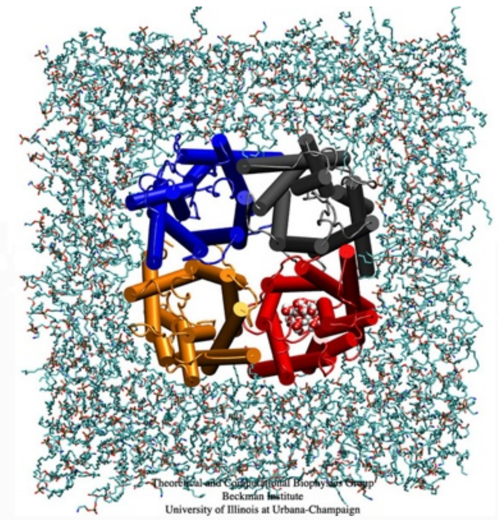
Automating Topology Aware Mapping for Supercomputers

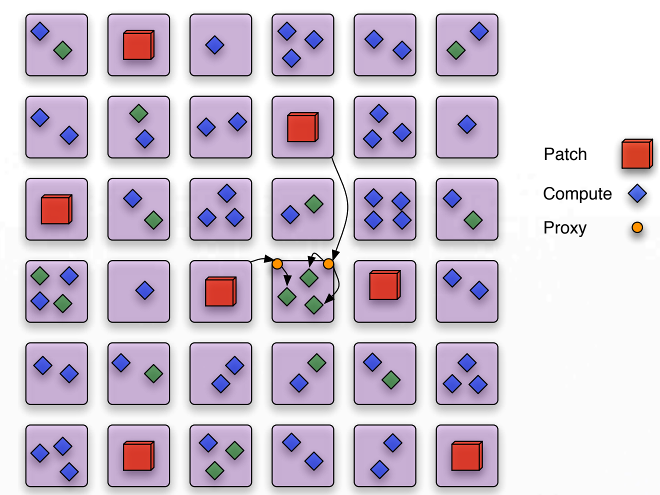
Abhinav Bhatele, Gagan Gupta
Laxmikant V. Kale

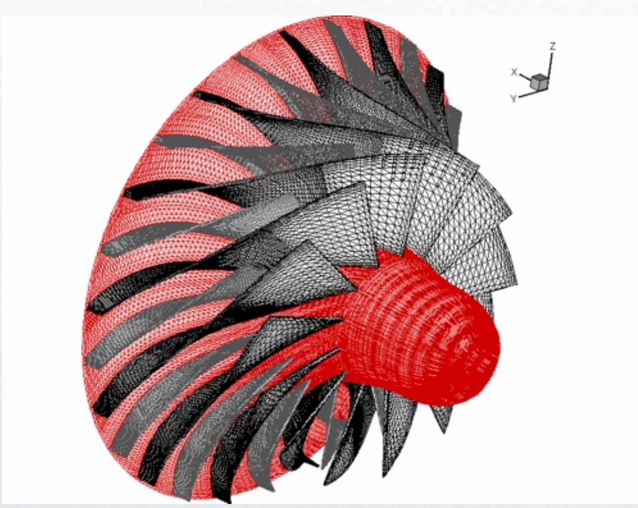
Application Topologies

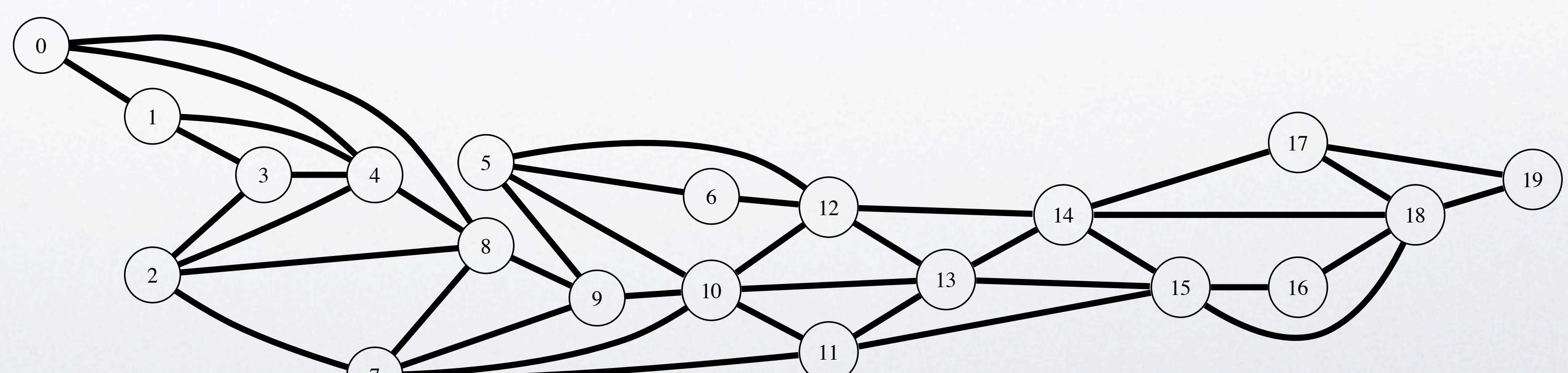




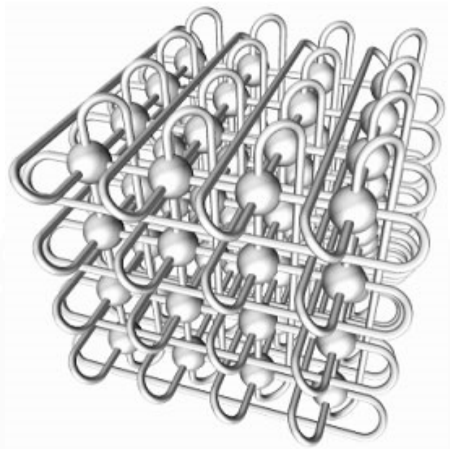




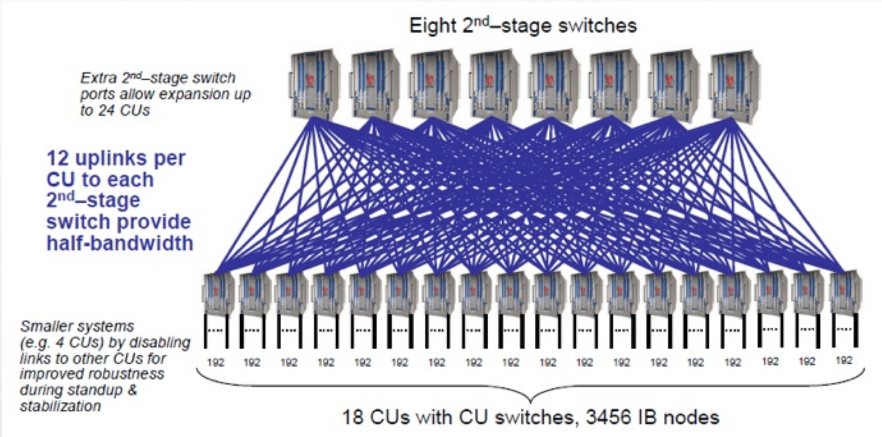




Interconnect Topologies



- Three dimensional meshes
  - 3D Torus: Blue Gene/L, Blue Gene/P, Cray XT4/5
- Trees
  - Fat-trees (Infiniband) and CLOS networks (Federation)
- Dense Graphs
  - Kautz Graph (SiCortex), Hypercubes
- Future Topologies?
  - Blue Waters, Blue Gene/Q

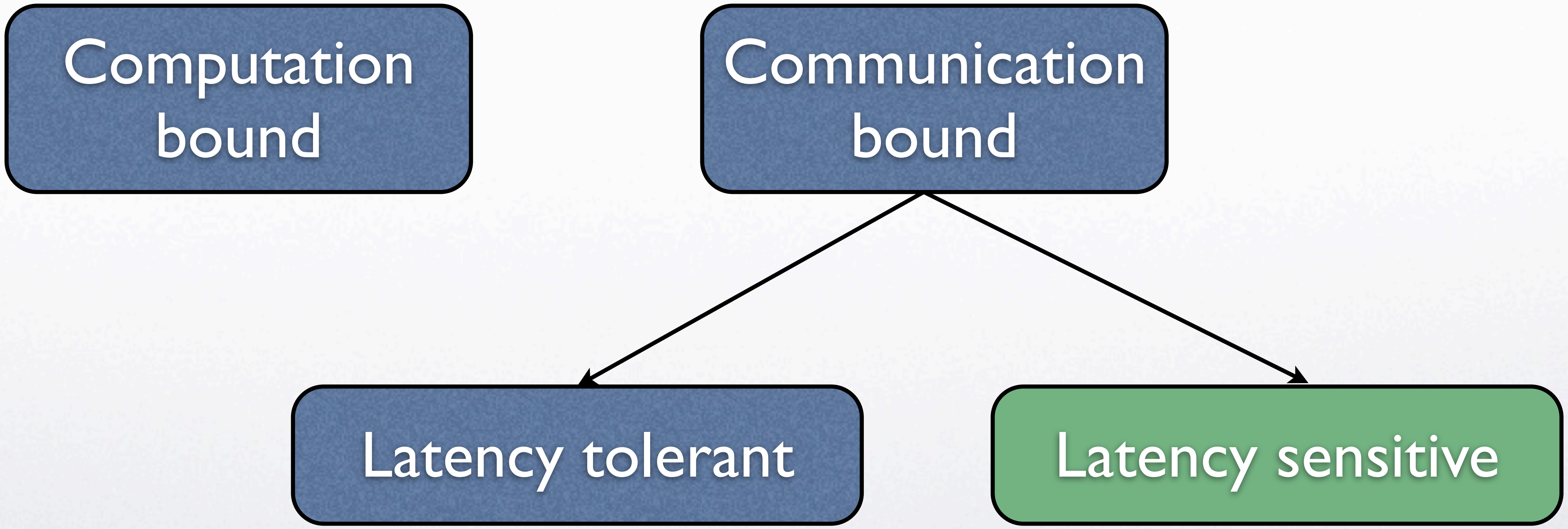


The Mapping Problem

- Applications have a communication topology and processors have an interconnect topology
- Definition: Given a set of communicating parallel "entities", map them on to physical processors to optimize communication
- Goals:
  - Balance computational load
  - Minimize communication traffic and hence contention

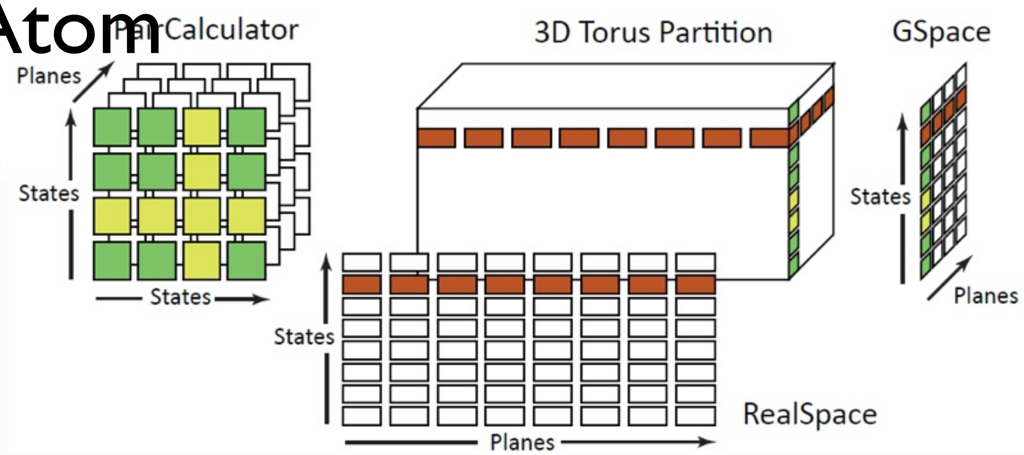
Scope of this work

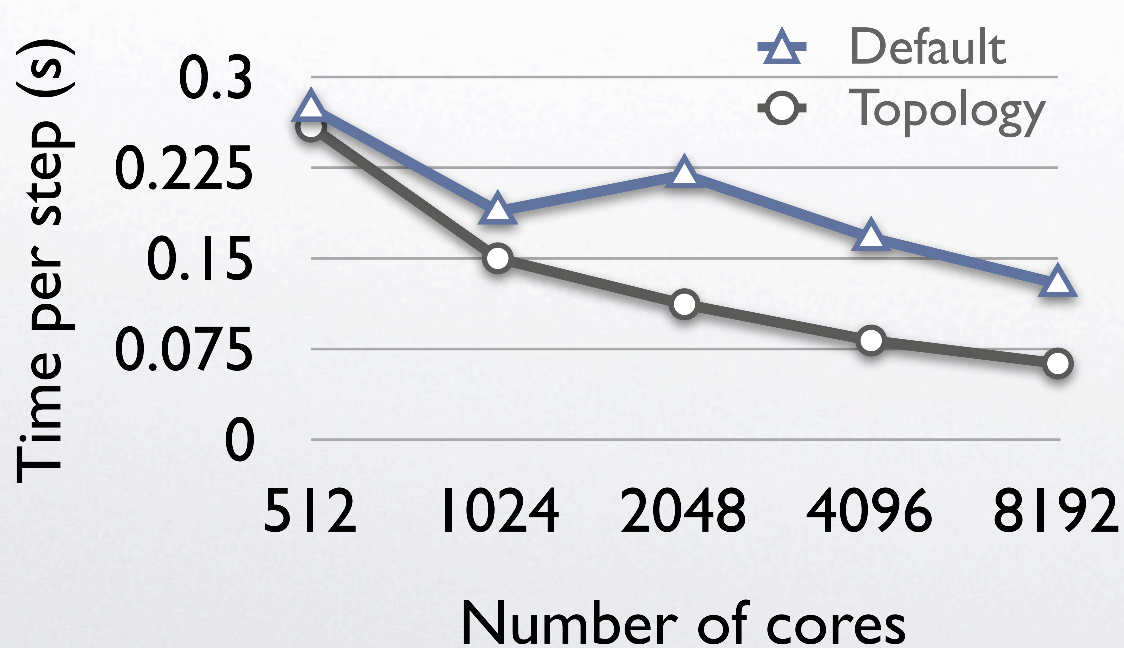
- Currently we are focused on 3D mesh/torus machines
- For certain classes of applications



Application specific mapping

OpenAtom



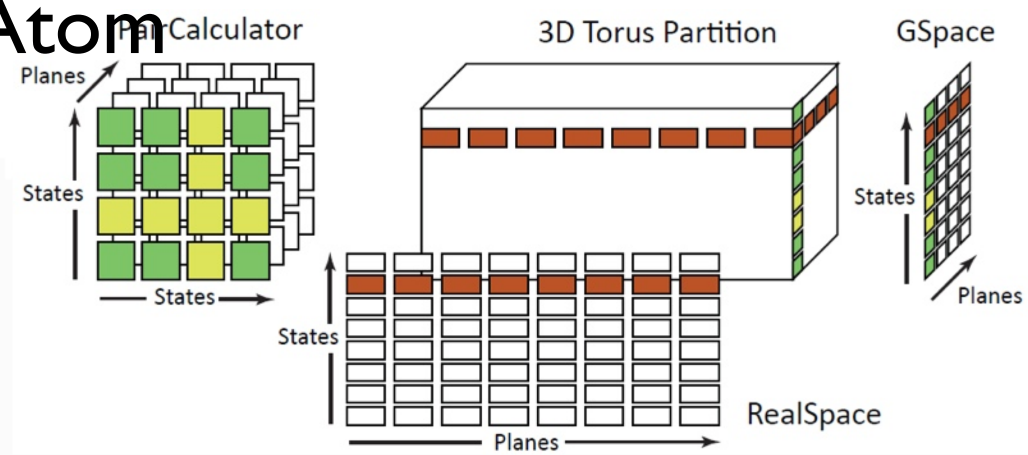


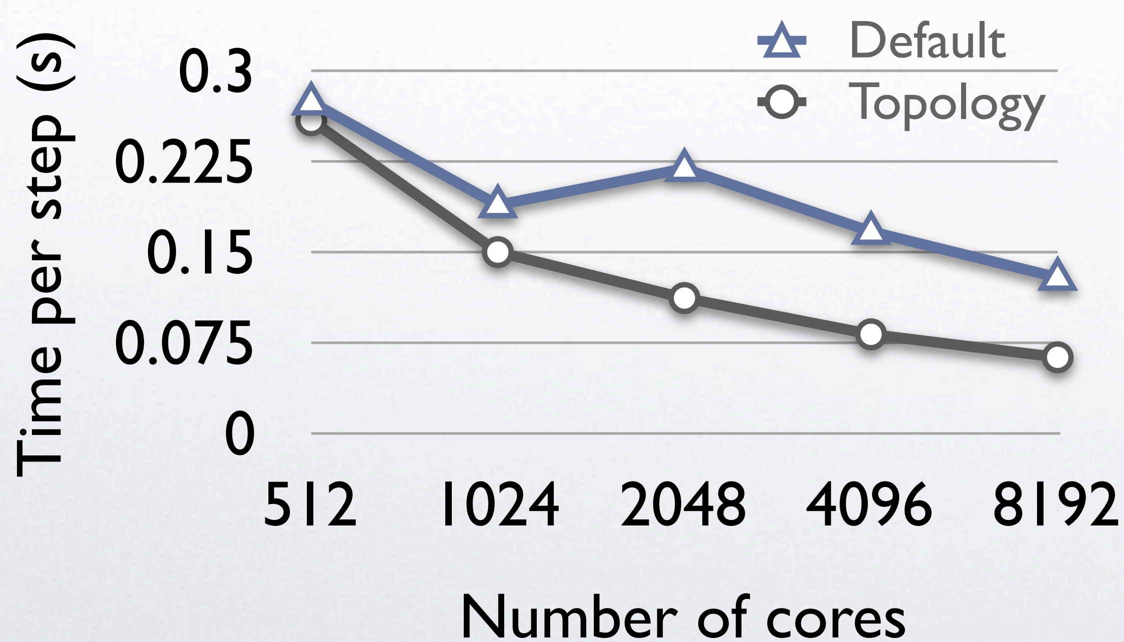
A. Bhatele, E. Bohm, and L.V. Kale. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In Euro-Par, LNCS 5704, pages 1015–1028, 2009. Distinguished Paper Award.

A. Bhatele, L.V. Kale and S. Kumar, Dynamic Topology Aware Load Balancing Algorithms for Molecular Dynamics Applications, In 23rd ACM International Conference on Supercomputing (ICS), 2009.

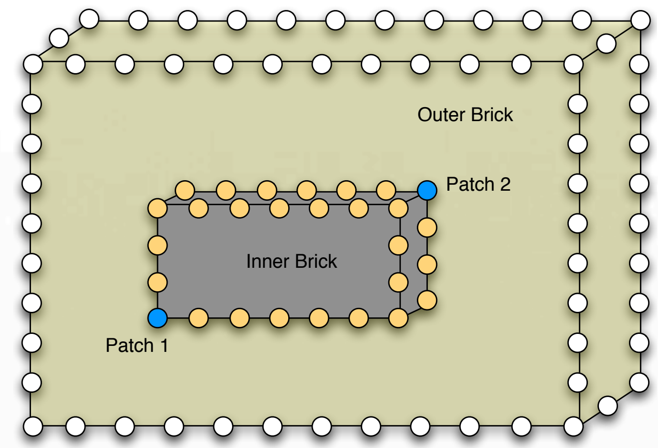
Application specific mapping

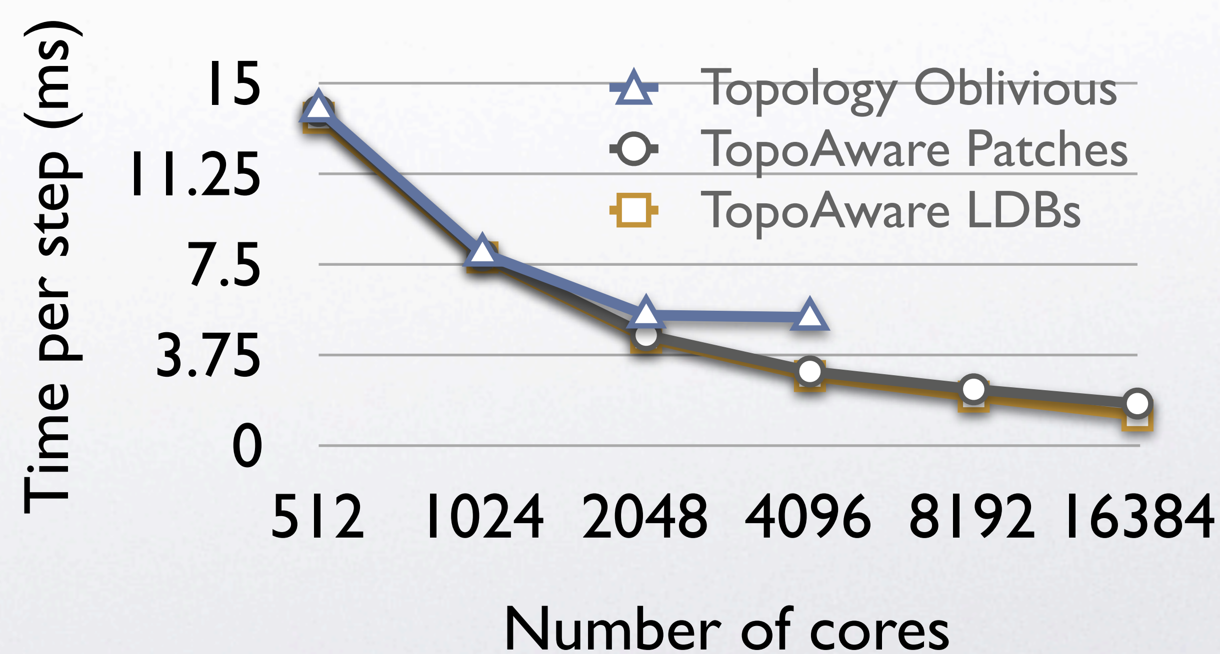
OpenAtom





NAMD





A. Bhatele, E. Bohm, and L.V. Kale. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In Euro-Par, LNCS 5704, pages 1015–1028, 2009. Distinguished Paper Award.

A. Bhatele, L.V. Kale and S. Kumar, Dynamic Topology Aware Load Balancing Algorithms for Molecular Dynamics Applications, In 23rd ACM International Conference on Supercomputing (ICS), 2009.

Automatic Mapping

- Obtaining the processor topology and the application communication graph
- Pattern matching to identify regular patterns
  - 2D/3D near-neighbor communication
- A suite of heuristics: the right strategy invoked depending on the communication scenario:
  - Regular communication
  - Irregular communication

Topology Discovery

- Topology Manager API: for 3D interconnects (Blue Gene, XT)
- Information required for mapping:
  - Physical dimensions of the allocated job partition
  - Mapping of ranks to physical coordinates and vice versa
- On Blue Gene machines such information is available and the API is a wrapper
- On Cray XT machines, jump several hoops to get this information and make it available through the same API

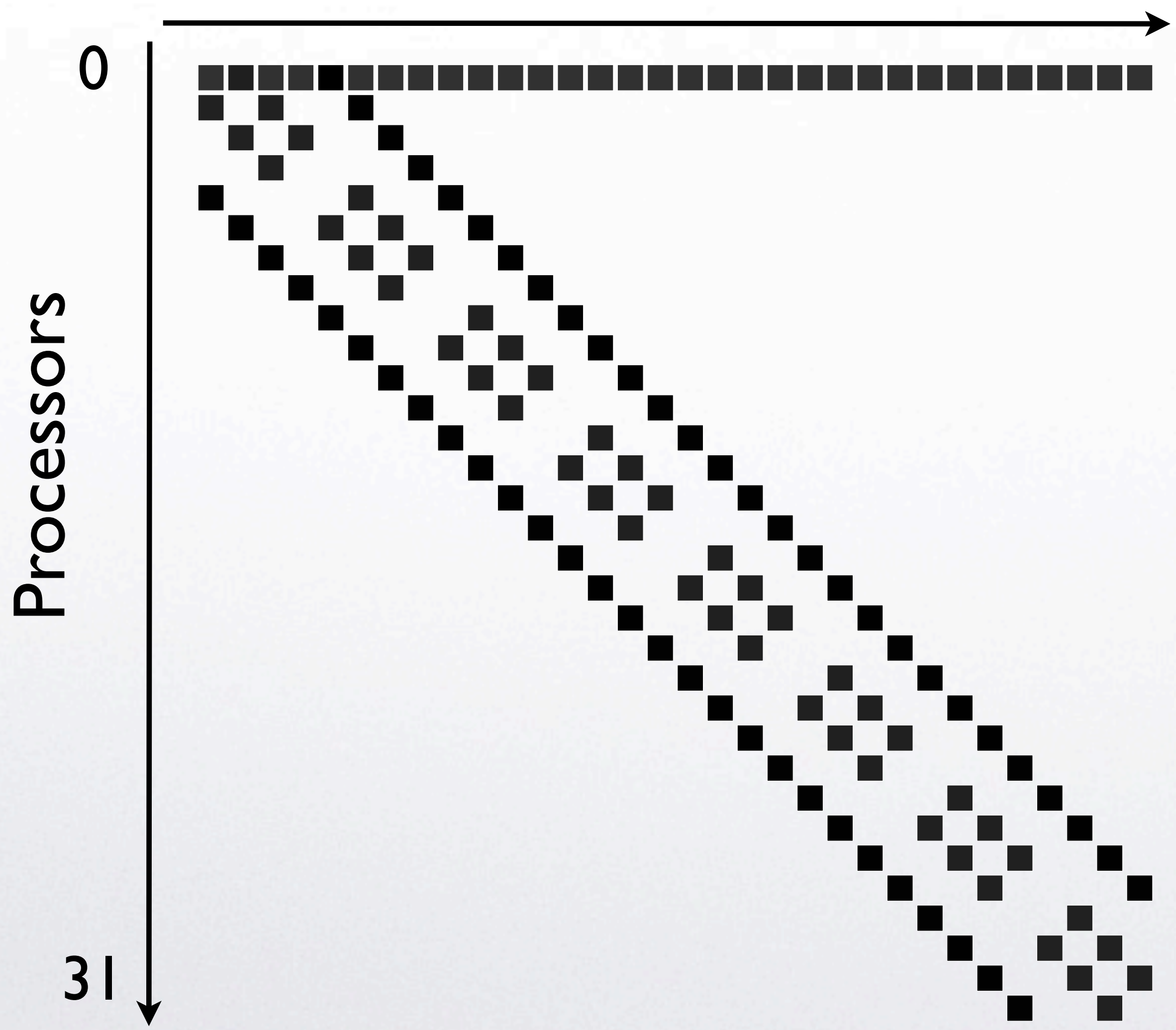
http://charm.cs.uiuc.edu/~bhatele/phd/TopoMgrAPI.tar.gz

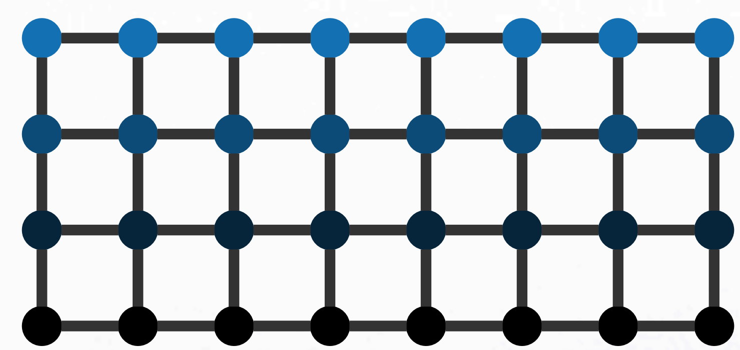
Application communication graph

- Several ways to obtain the graph
- MPI applications:
  - Graph obtained from a run can only be used in a subsequent run
  - Profiling tools (IBM's HPCT tools)
- Charm++ applications:
  - Instrumentation at runtime
  - Enables dynamic mapping for changing communication graphs

Pattern Matching

- We want to identify simple communication patterns





Pattern matching to identify simple communication patterns such as 2D/3D near-neighbor graphs

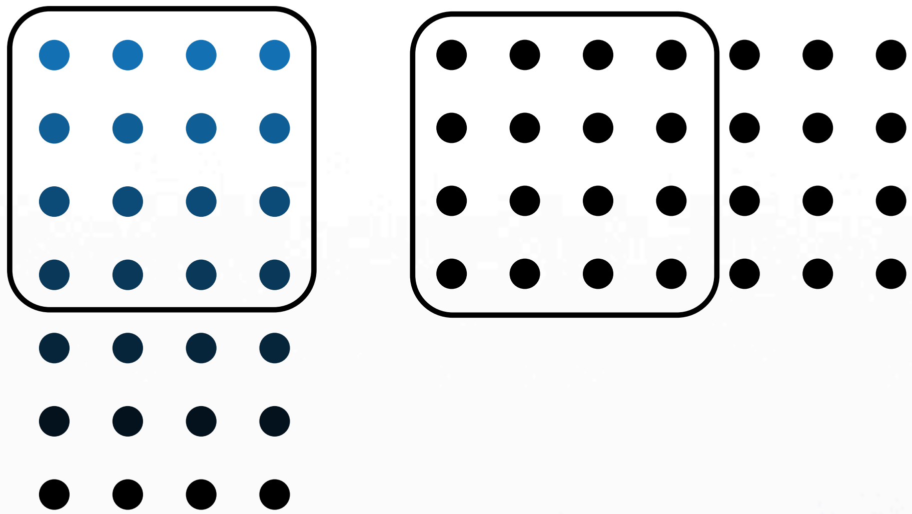
Communication Graphs

- Regular communication:
  - POP (Parallel Ocean Program): 2D Stencil like computation
  - WRF (Weather Research and Forecasting model): 2D Stencil
  - MILC (MIMD Lattice Computation): 4D near-neighbor
- Irregular communication:
  - Unstructured mesh computations: FLASH, CPSD code
  - Many other classes of applications

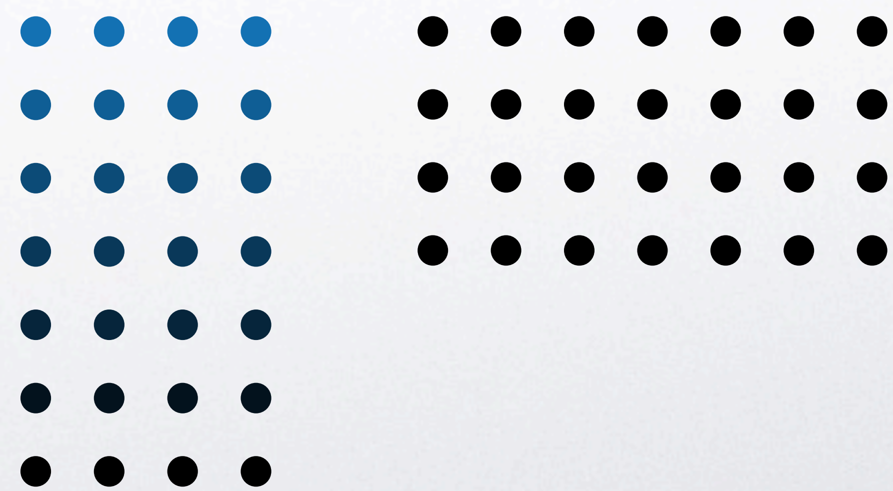
Mapping Regular Graphs

- Maximum Overlap (MXOVLP)

Object Graph: 7 x 4
Processor Graph: 4 x 7



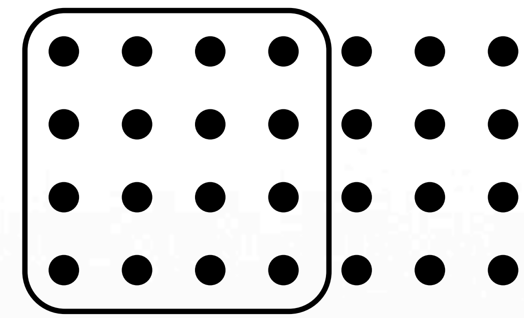
- Expand from Corner (EXCO)



- Affine Mapping (AFFN)

Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



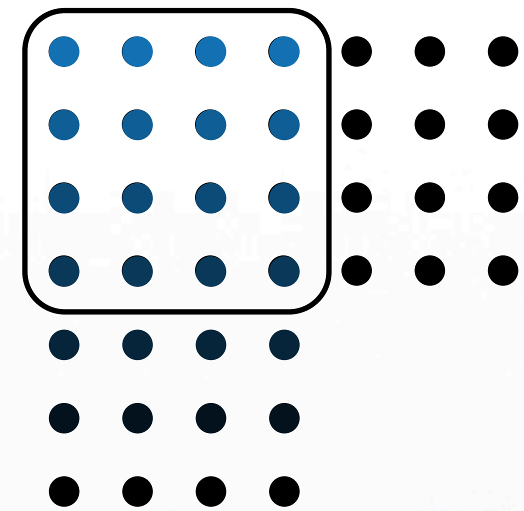
Object Graph: 7 x 4
Processor Graph: 4 x 7

- Expand from Corner (EXCO)

- Affine Mapping (AFFN)

Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



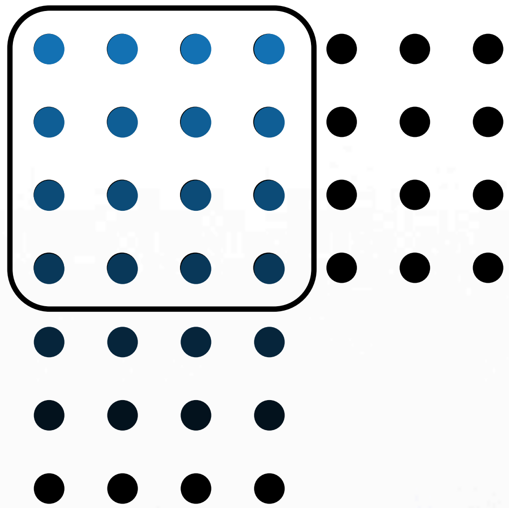
Object Graph: 7 x 4
Processor Graph: 4 x 7

- Expand from Corner (EXCO)

- Affine Mapping (AFFN)

Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



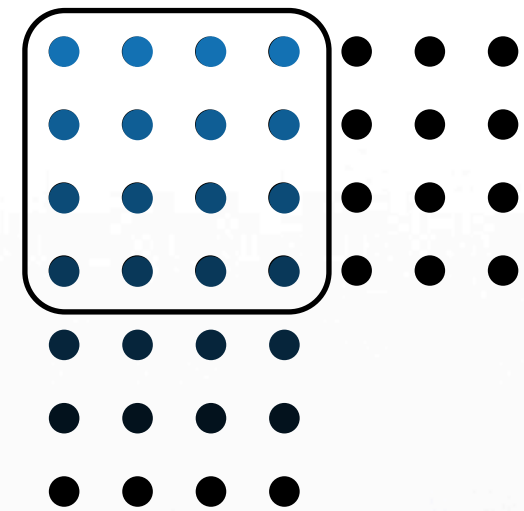
Object Graph: 7 x 4
Processor Graph: 4 x 7

- Expand from Corner (EXCO)

- Affine Mapping (AFFN)

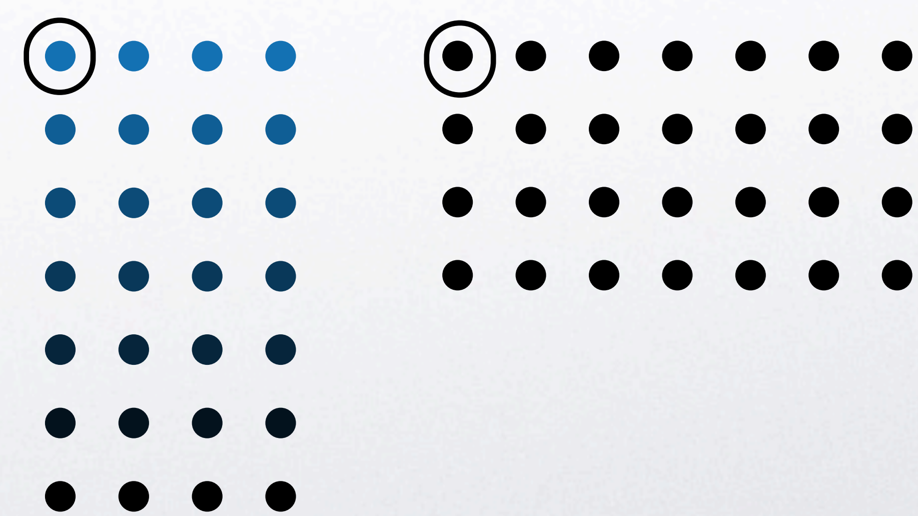
Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



Object Graph: 7 x 4
Processor Graph: 4 x 7

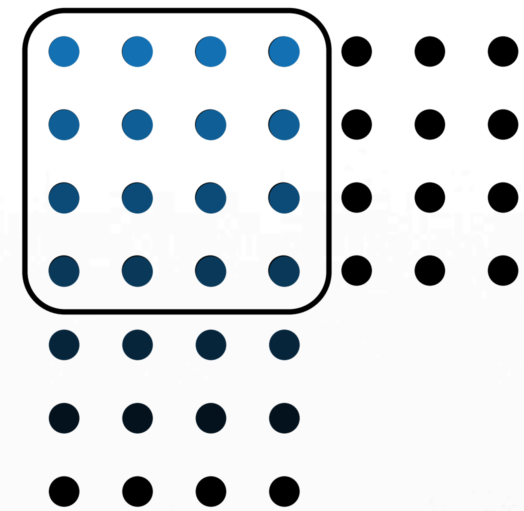
- Expand from Corner (EXCO)



- Affine Mapping (AFFN)

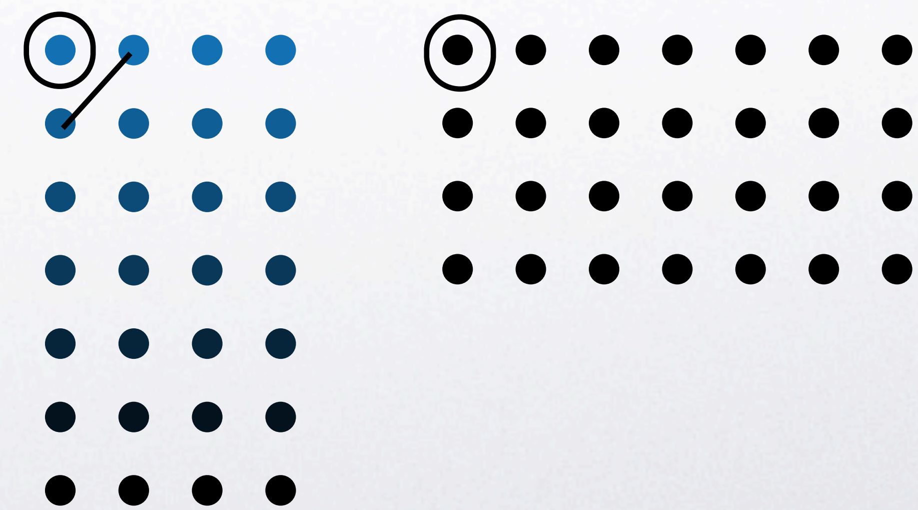
Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



Object Graph: 7 x 4
Processor Graph: 4 x 7

- Expand from Corner (EXCO)

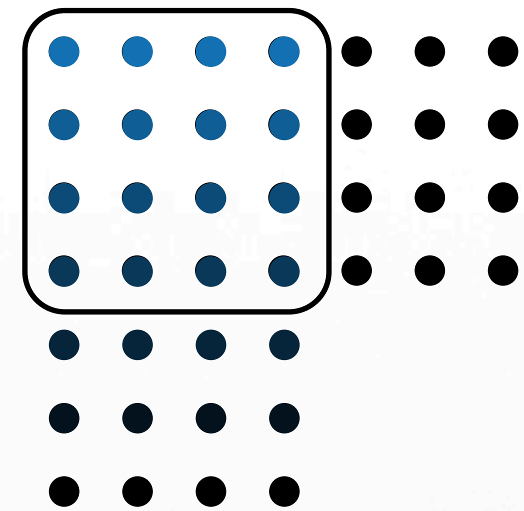


- Affine Mapping (AFFN)

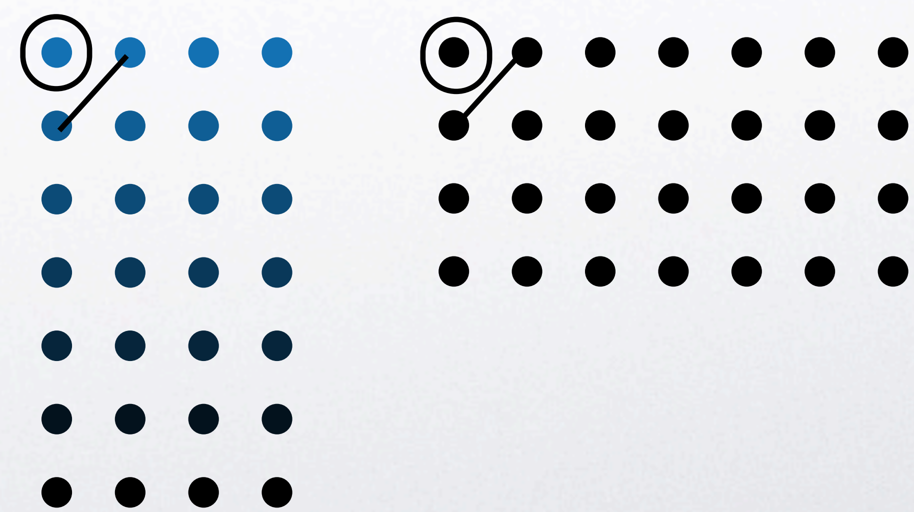
Mapping Regular Graphs

- Maximum Overlap (MXOVLP)

Object Graph: 7 x 4
Processor Graph: 4 x 7



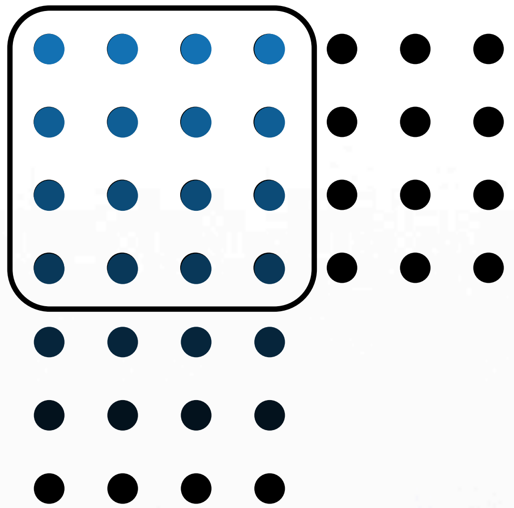
- Expand from Corner (EXCO)



- Affine Mapping (AFFN)

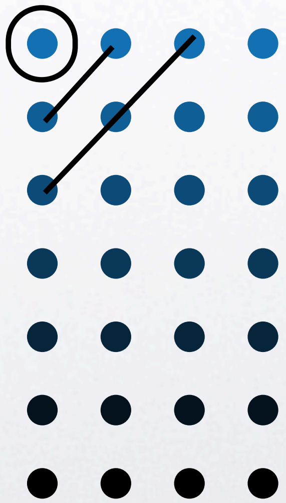
Mapping Regular Graphs

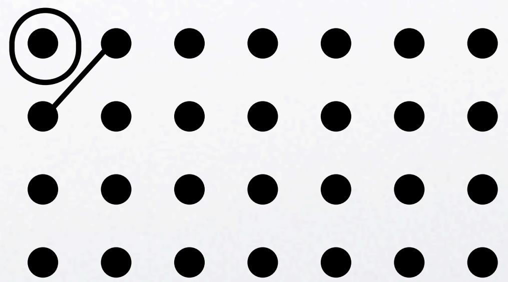
- Maximum Overlap (MXOVLP)



Object Graph: 7 x 4
Processor Graph: 4 x 7

- Expand from Corner (EXCO)

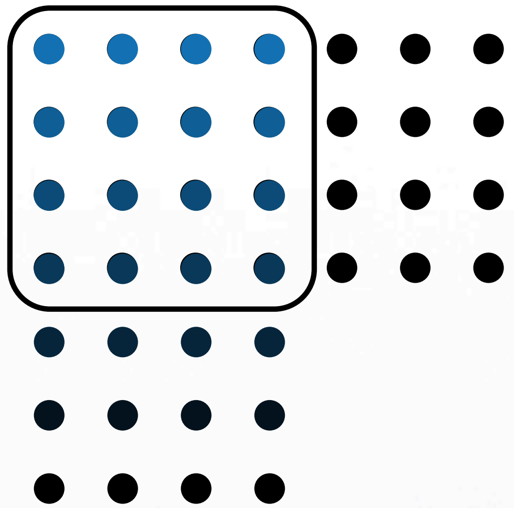




- Affine Mapping (AFFN)

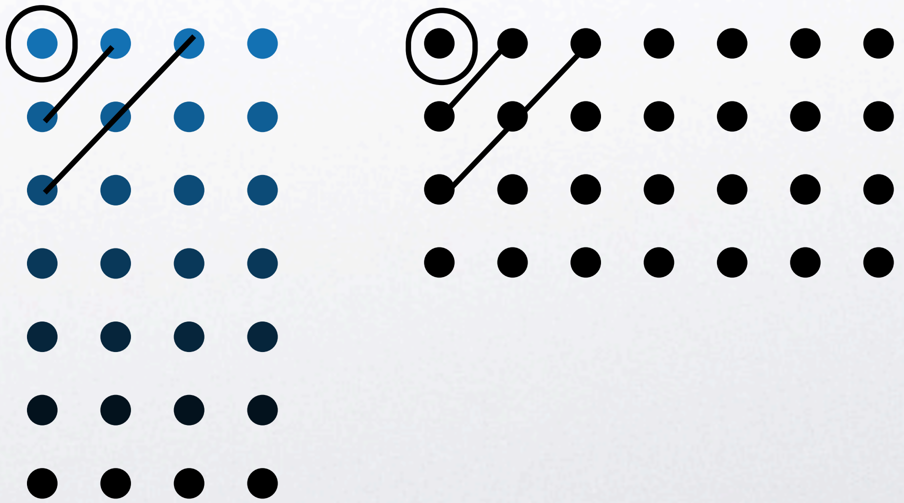
Mapping Regular Graphs

- Maximum Overlap (MXOVLP)



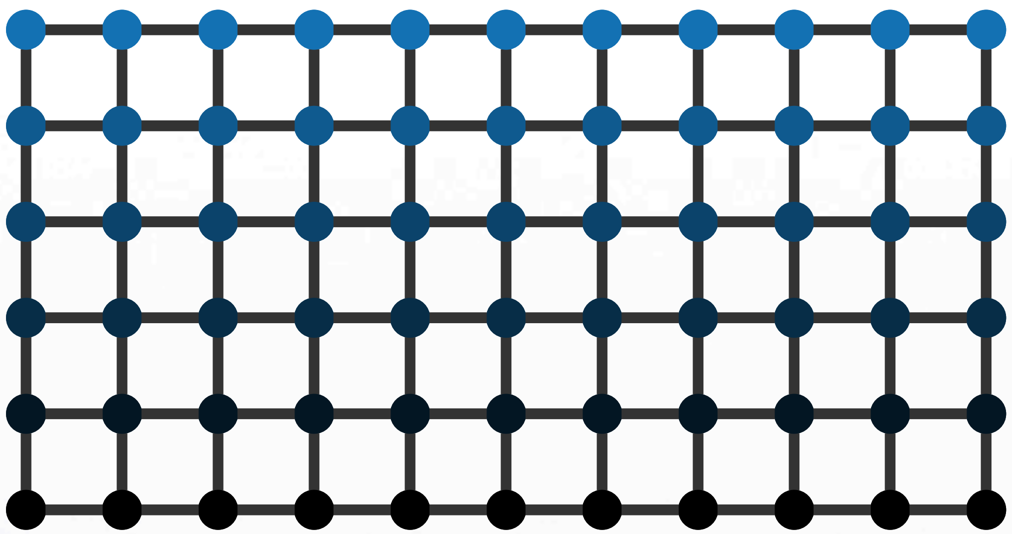
Object Graph: 7 x 4
Processor Graph: 4 x 7

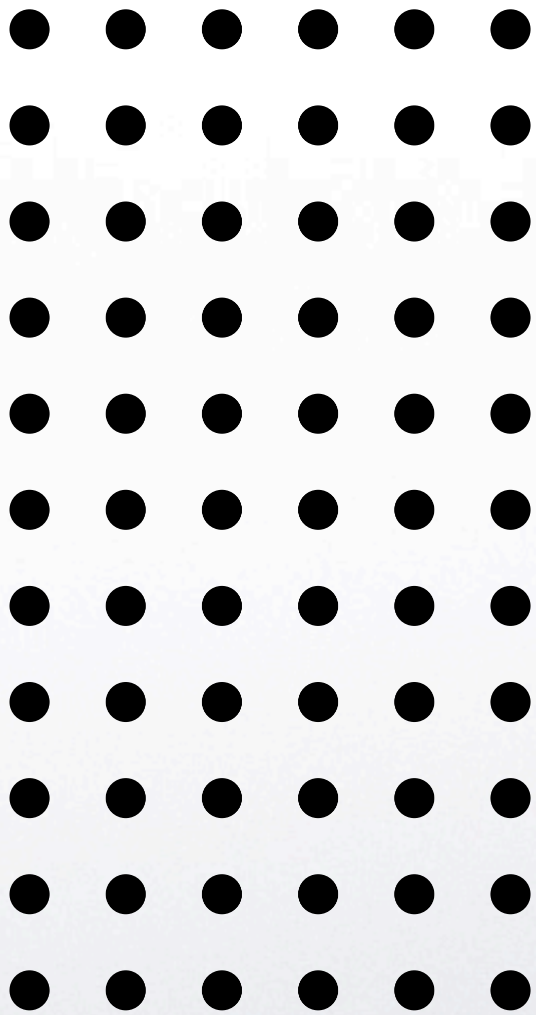
- Expand from Corner (EXCO)



- Affine Mapping (AFFN)

Example Mapping

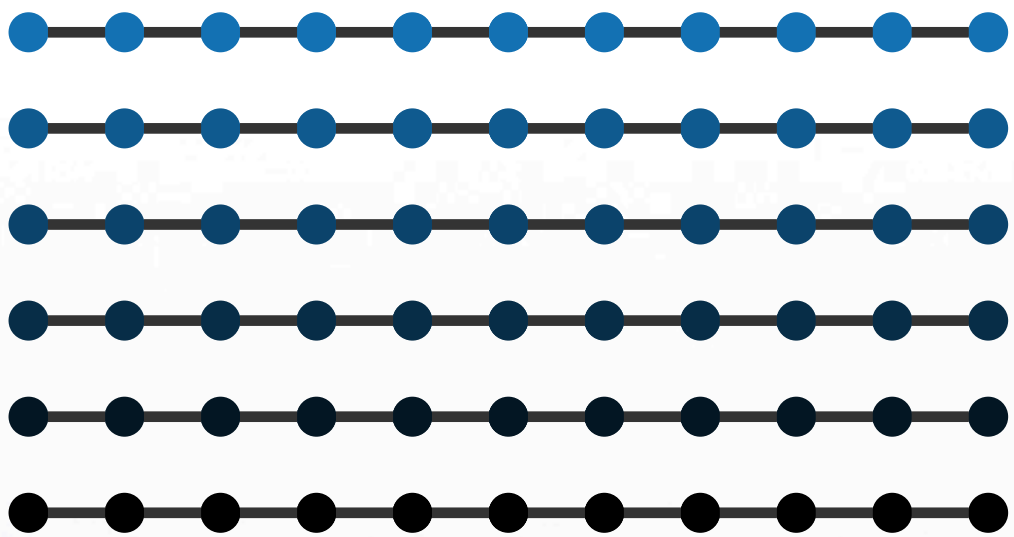


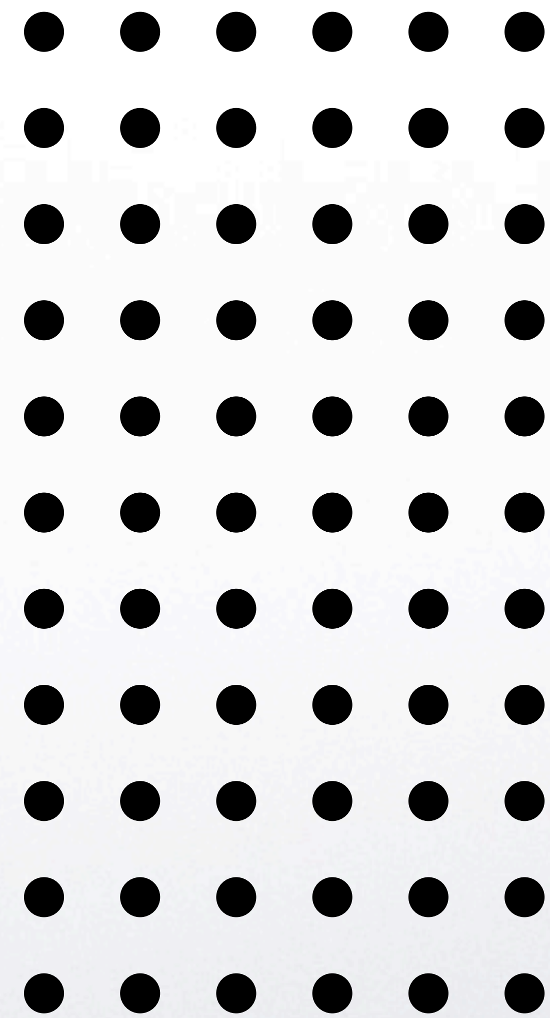


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

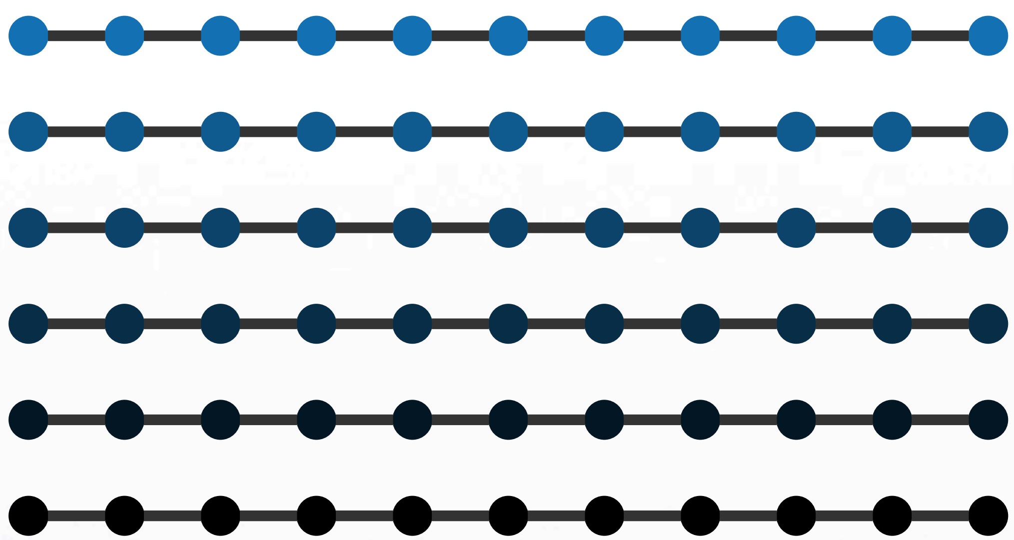


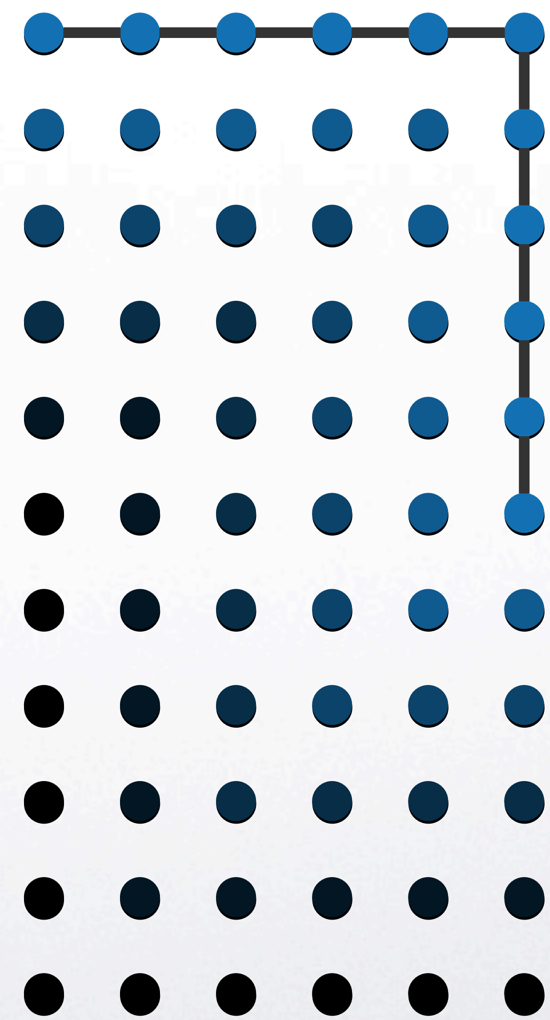


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

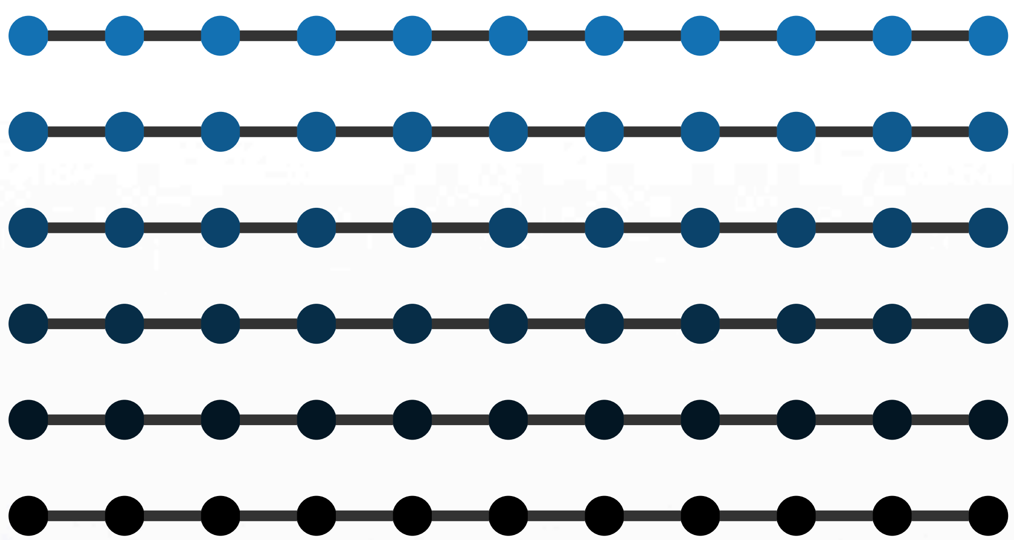




Object Graph: 6 x 11
Processor Graph: 11 x 6

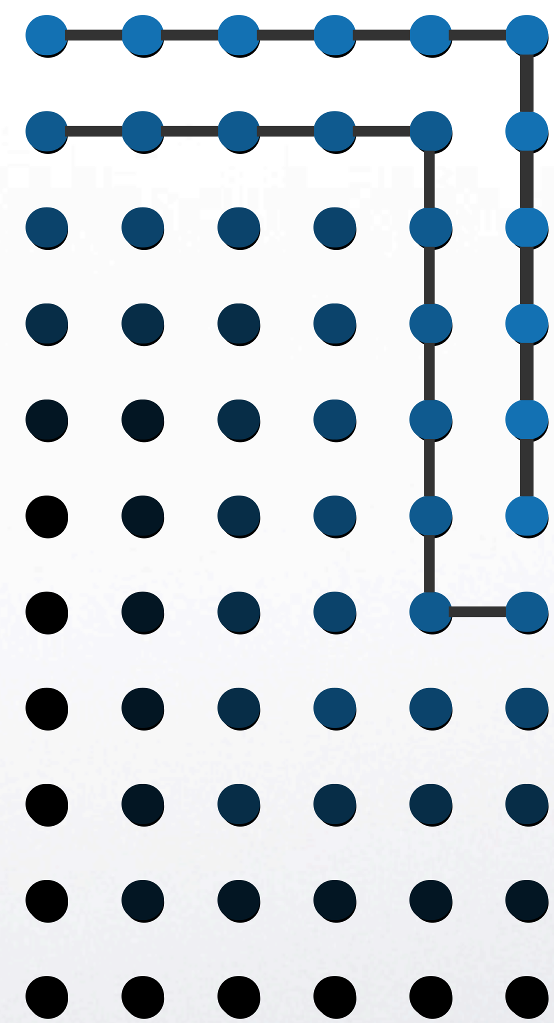
Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

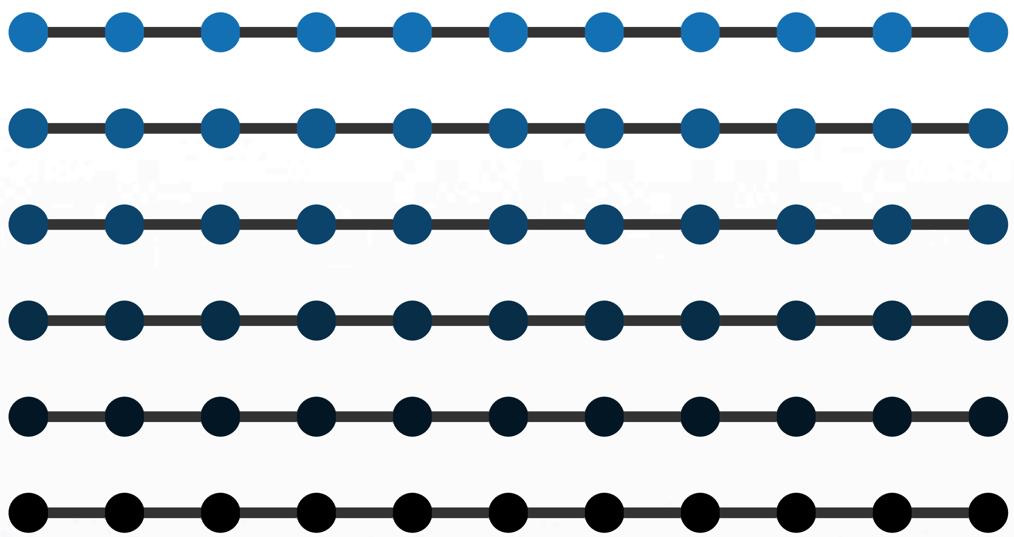


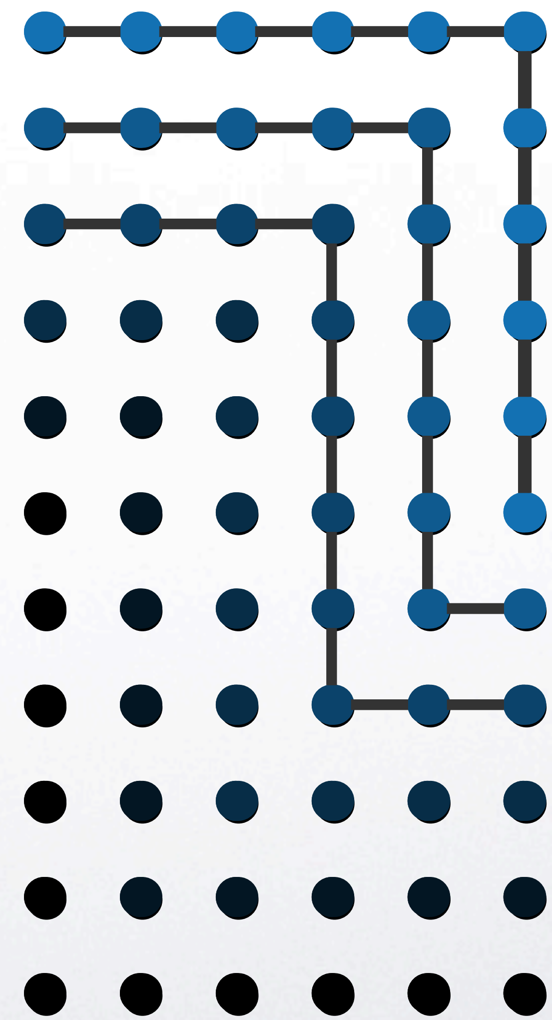
Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982



Example Mapping

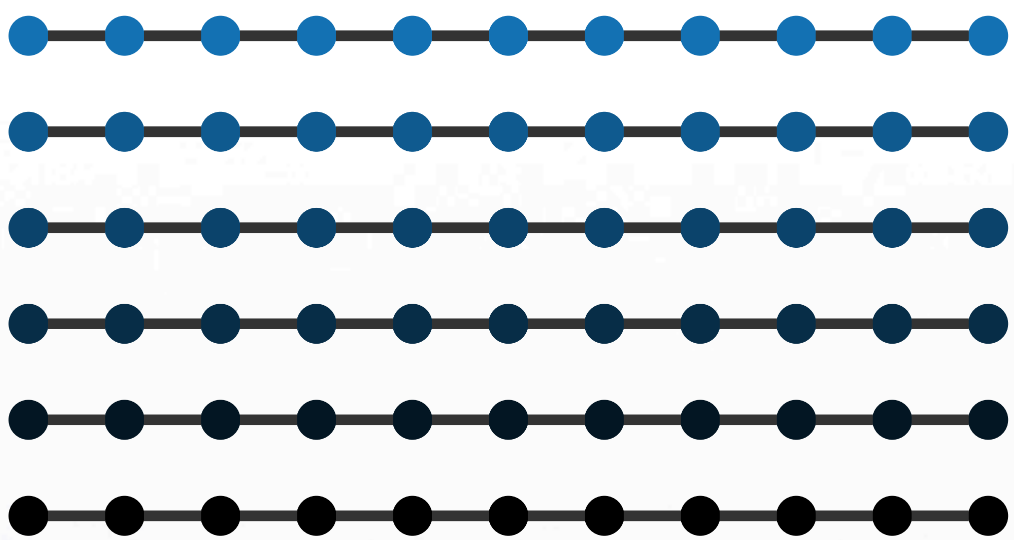


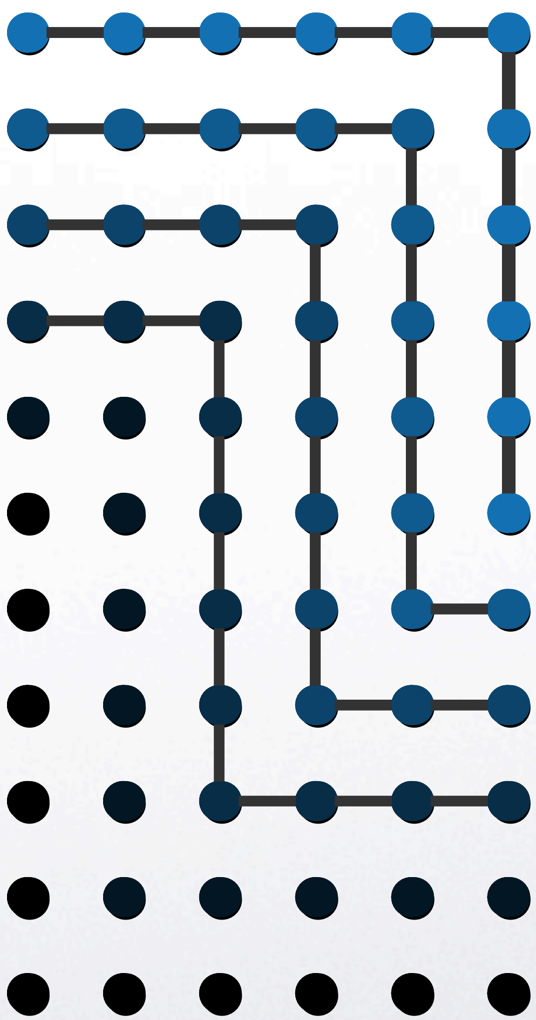


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

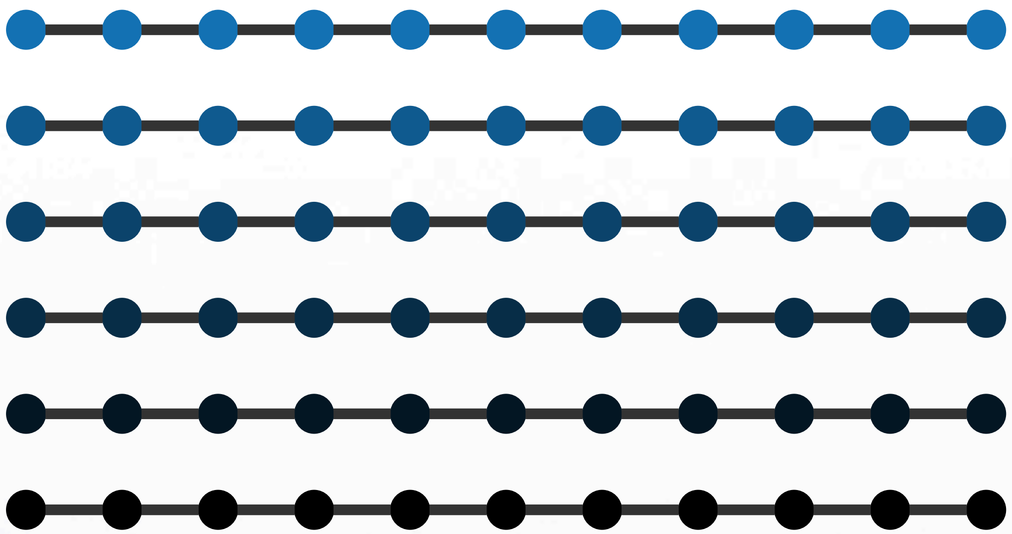


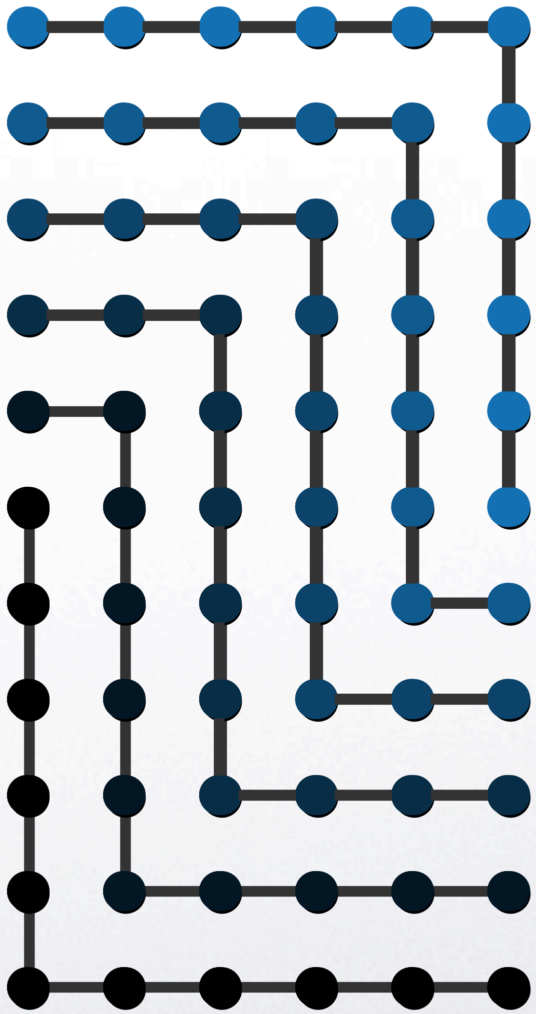


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

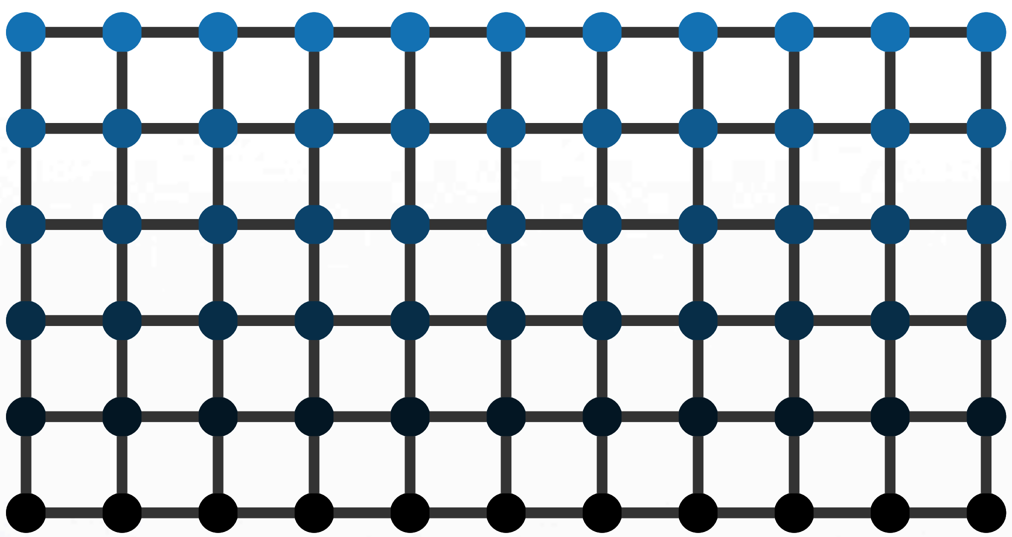


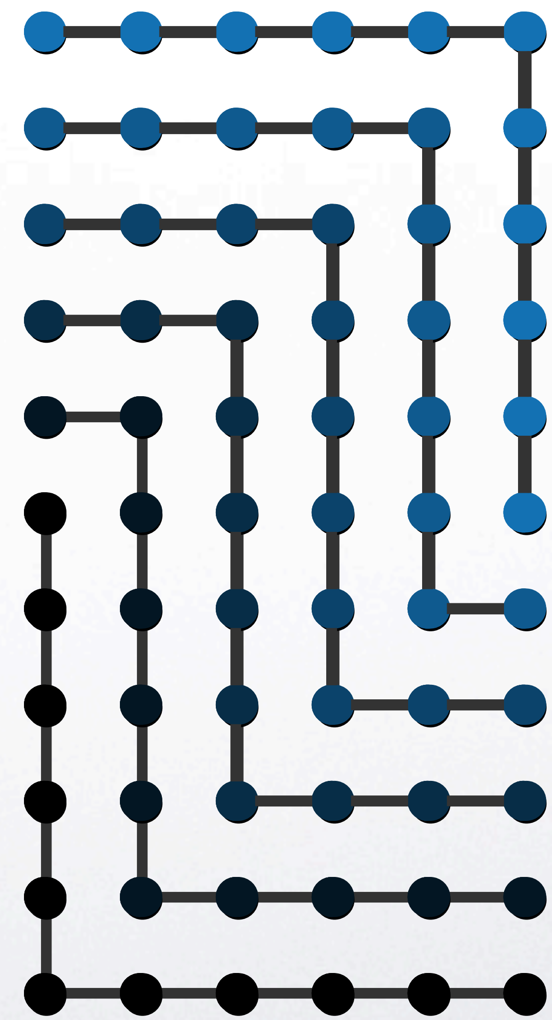


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

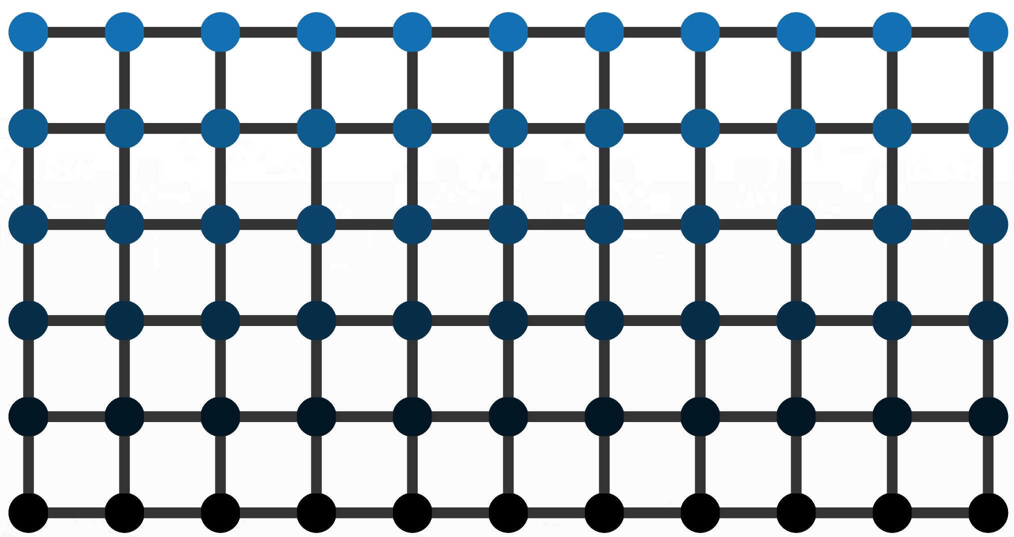


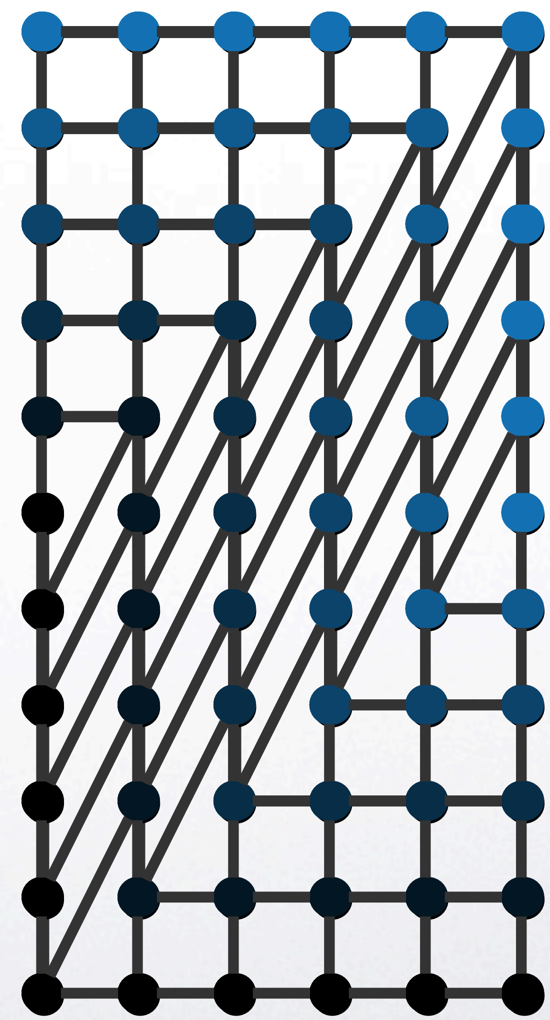


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping



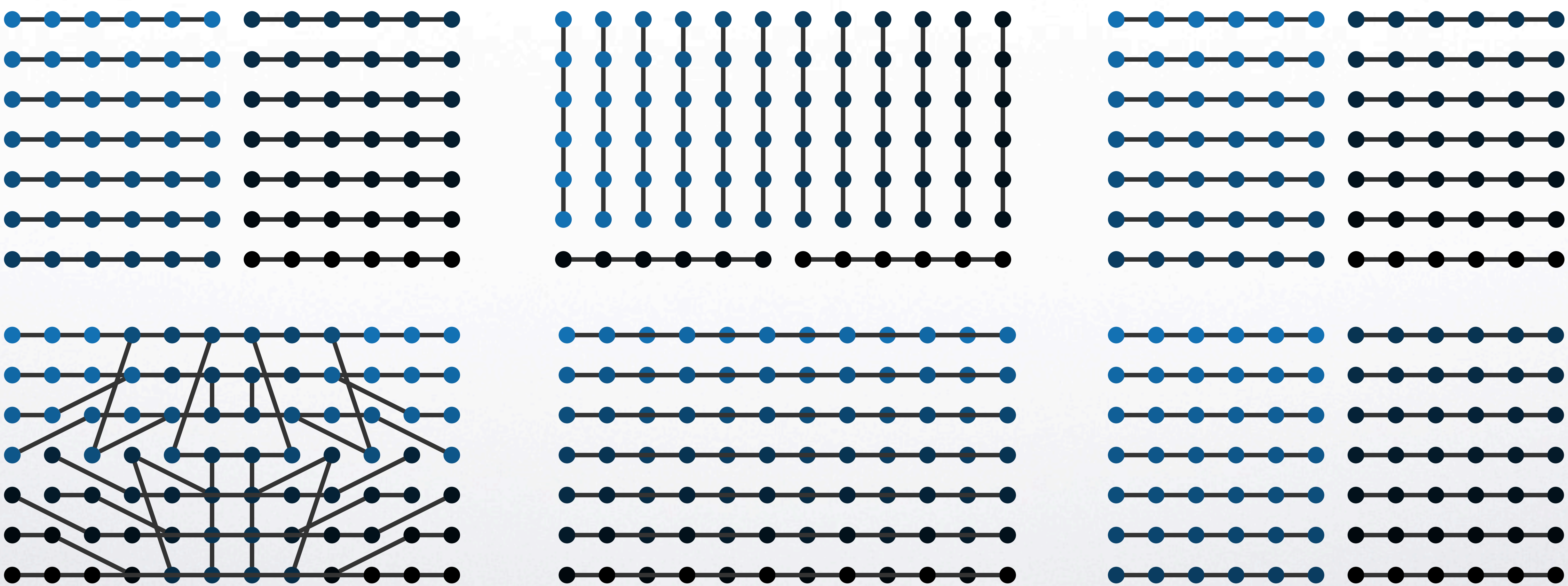


Object Graph: 6 x 11
Processor Graph: 11 x 6

Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Different mapping solutions

Object graph of 14 x 6 to processor graph of 7 x 12



Algorithms in order: MXOVLP, MXOV+AL, EXCO, COCE, AFFN, STEP

Evaluation Metric: Hop-bytes

- Weighted sum of message sizes where the weights are the number of links traversed by each message

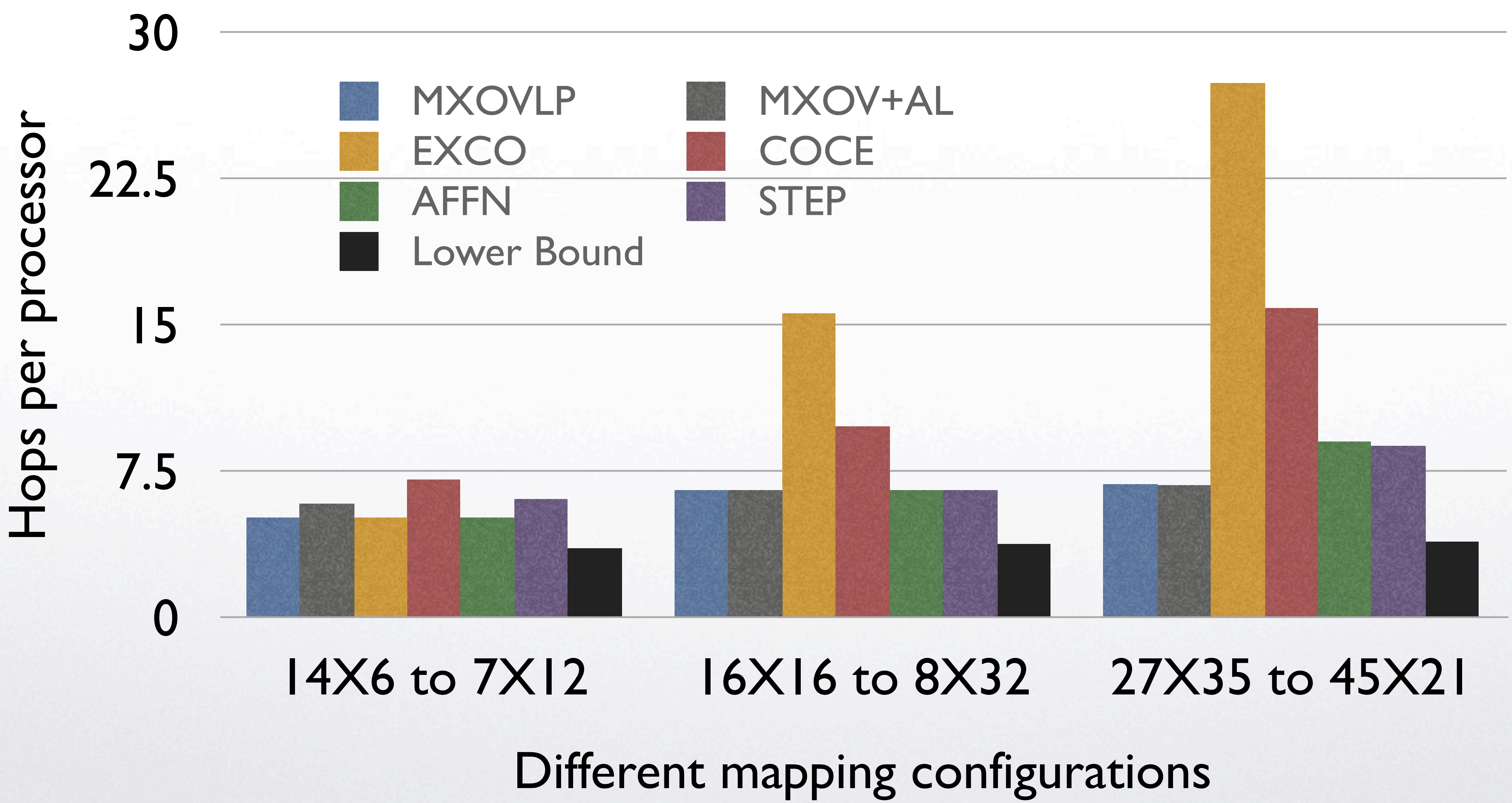
$$HB = \sum_{i=1}^{n} d_i \times b_i$$

d_i = distance
b_i = bytes
n = no. of messages

- Indicator of the communication traffic and hence contention on the network
- Previously used metric: maximum dilation

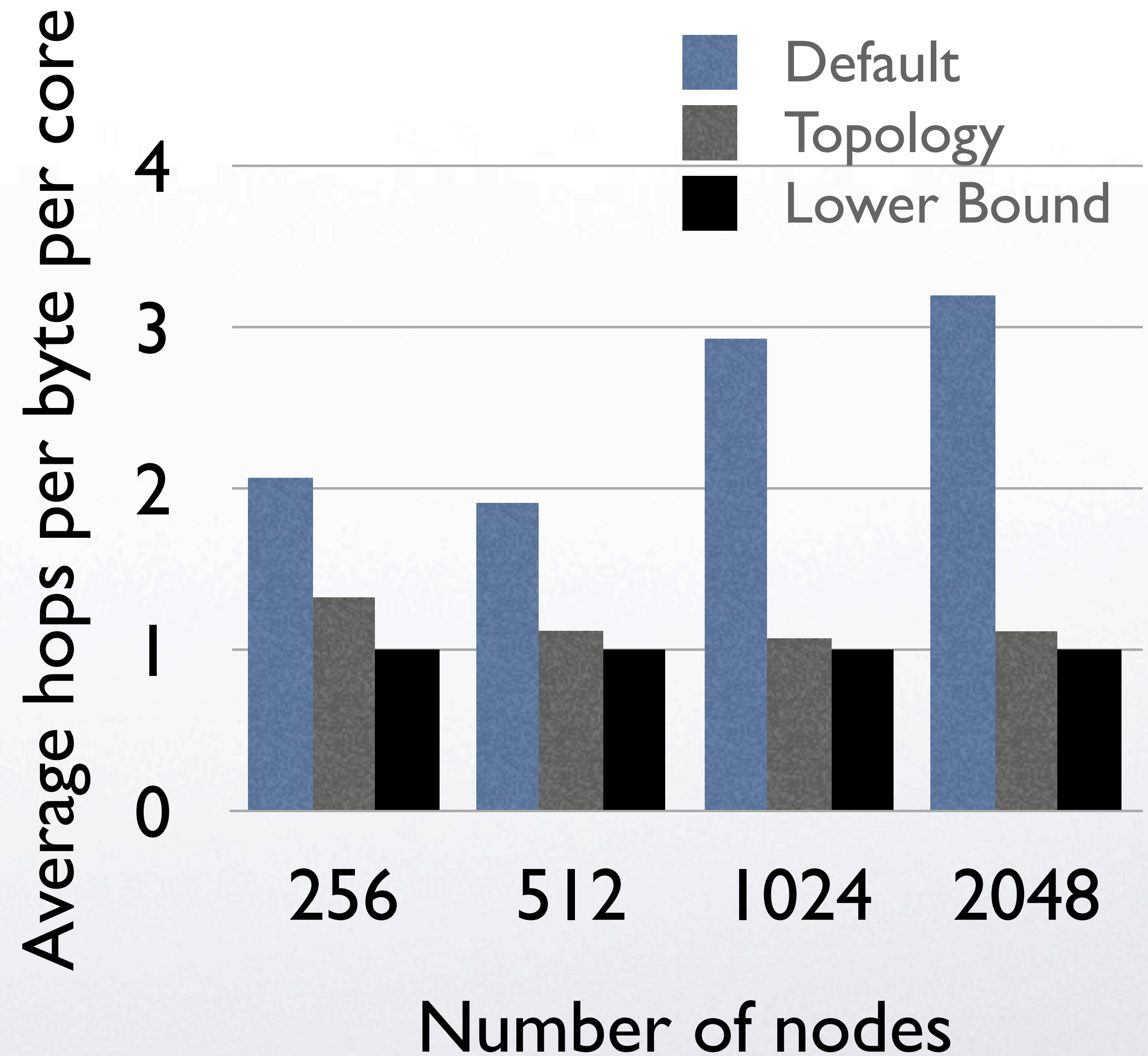
$$d(e) = \max\{|i-k| + |j-m|\}$$

Evaluation

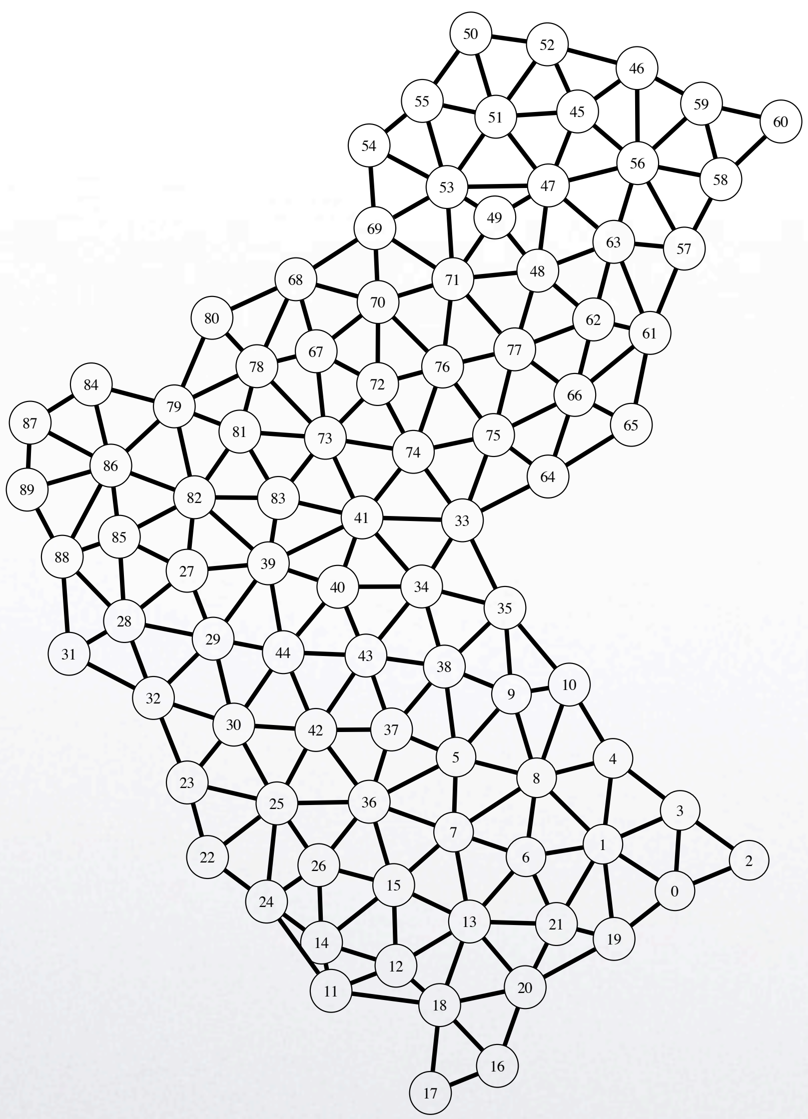

30
22.5
15
7.5
0
Hops per processor
MXOVLP
MXOV+AL
EXCO
COCE
AFFN
STEP
Lower Bound
14X6 to 7X12
16X16 to 8X32
27X35 to 45X21
Different mapping configurations

Results: WRF

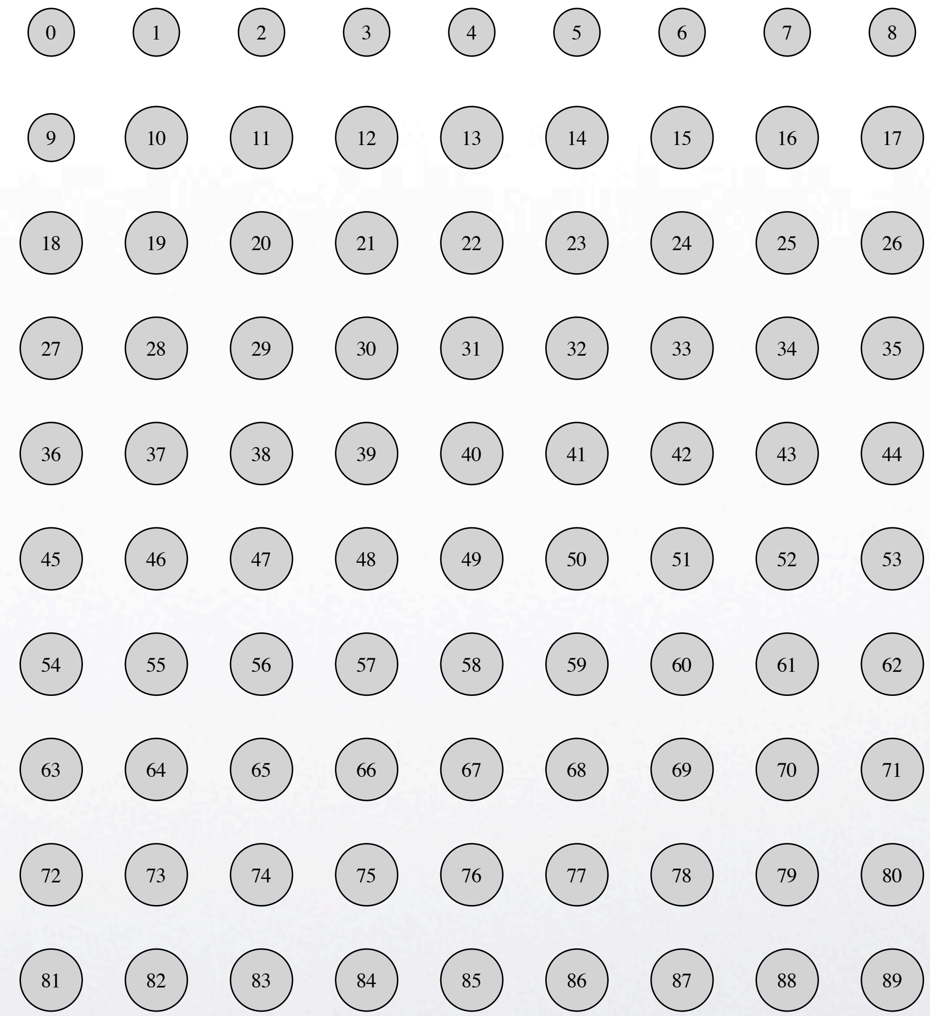
- Performance improvement negligible on 256 and 512 cores
- On 1024 cores:
  - Hops reduce by: 64%
  - Time for communication reduces by 45%
  - Performance improves by 17%



Mapping Irregular Graphs



Object graph: 90 nodes



Processor Mesh: 10 x 9

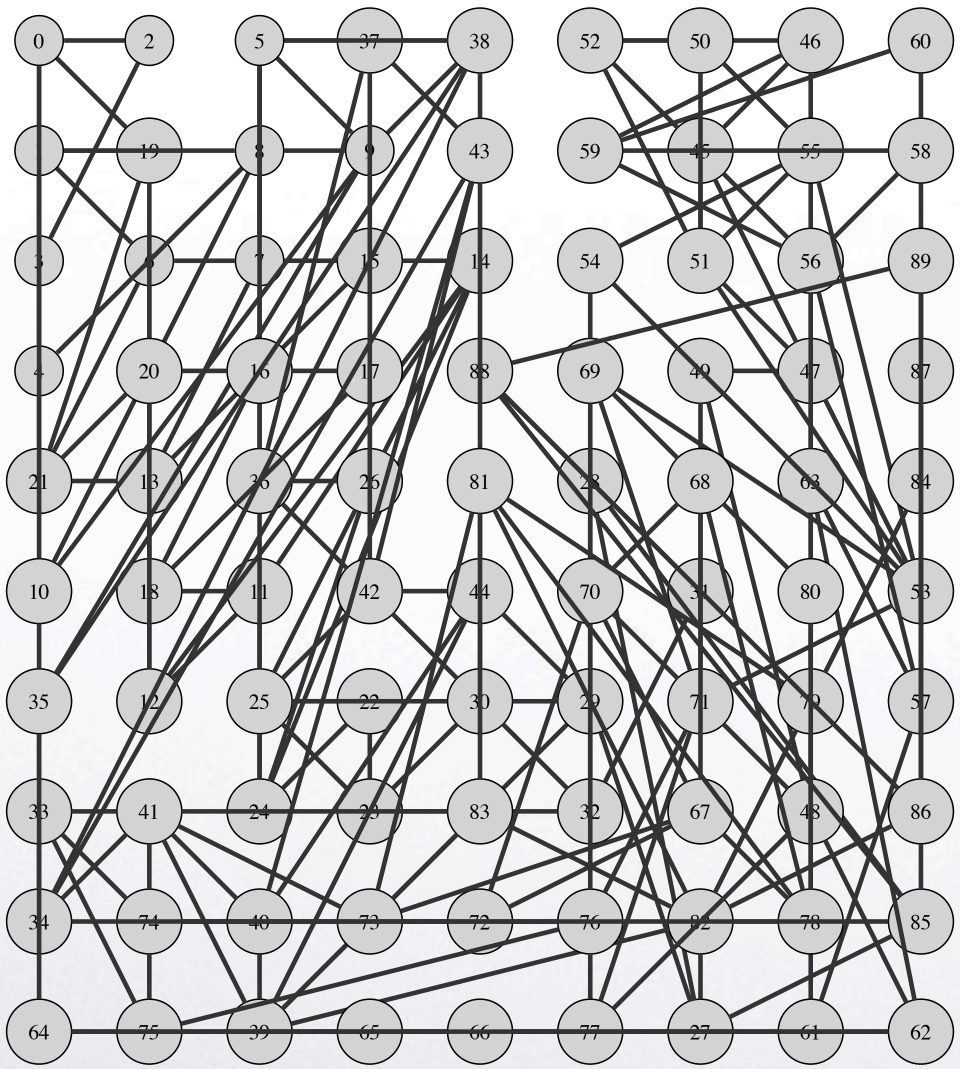
Two different scenarios

- There is no spatial information associated with the node
  - Option 1: Work without it
  - Option 2: If we know that the simulation has a geometric configuration, try to guess the structure of the graph
- We have geometric coordinate information for each node
  - Use coordinate information to avoid crossing of edges and for other optimizations

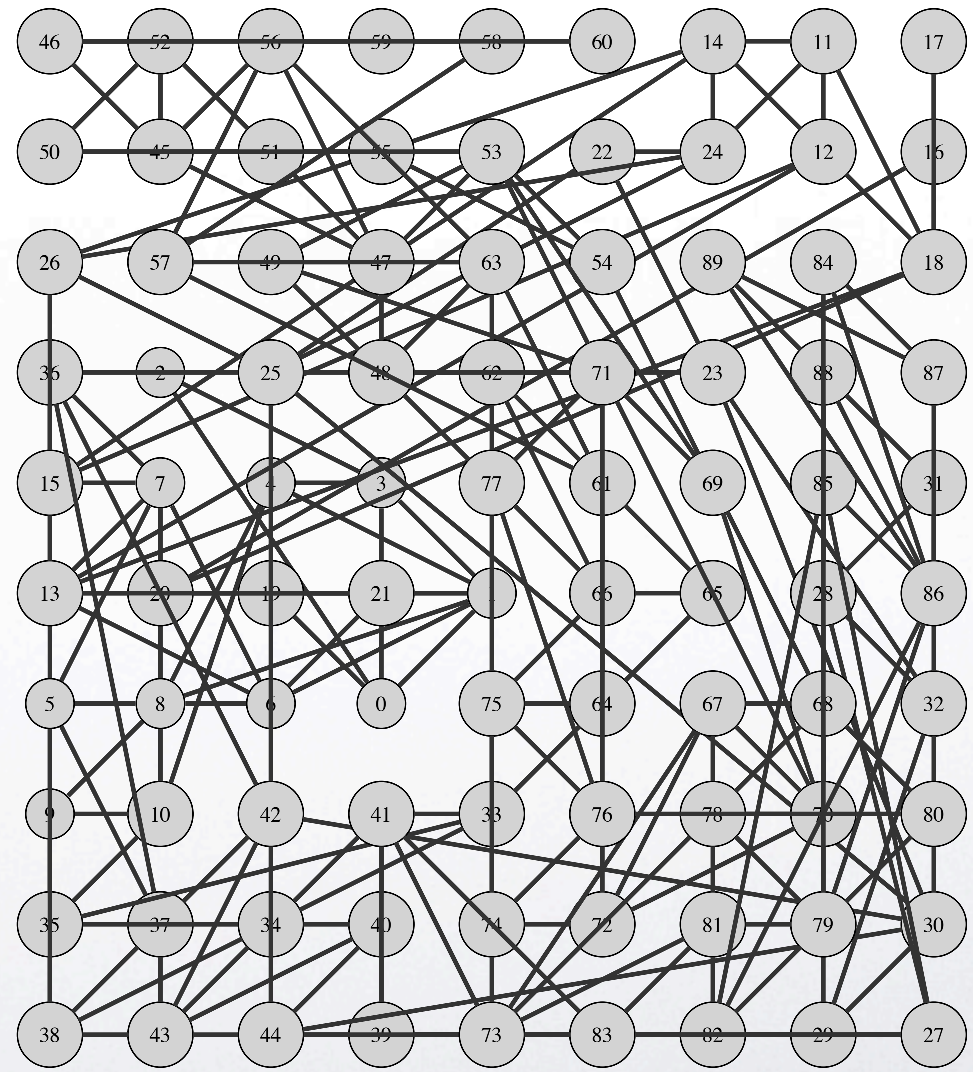
No coordinate information

- Breadth first traversal (BFT)
  - Start with a random node and one end of the processor mesh
  - Map nodes as you encounter them around the centroid of their mapped neighbors
- Max heap traveral (MHT)
  - Start with a random node and one end/center of the mesh
  - Put neighbors of a mapped node into the heap (node at the top is the one with maximum mapped neighbors)
  - Map elements in the heap one by one around the centroid of their mapped neighbors

Mapping visualization



BFT



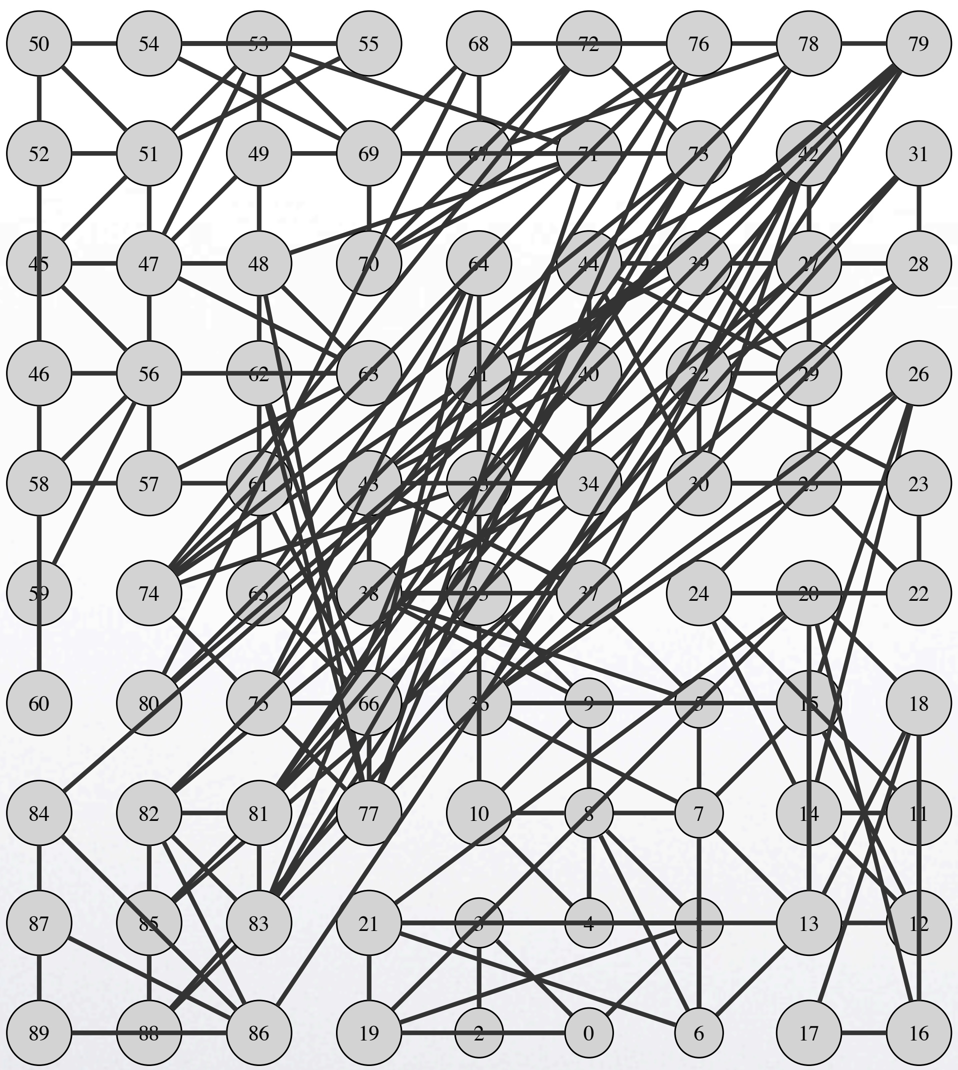
MHT

21

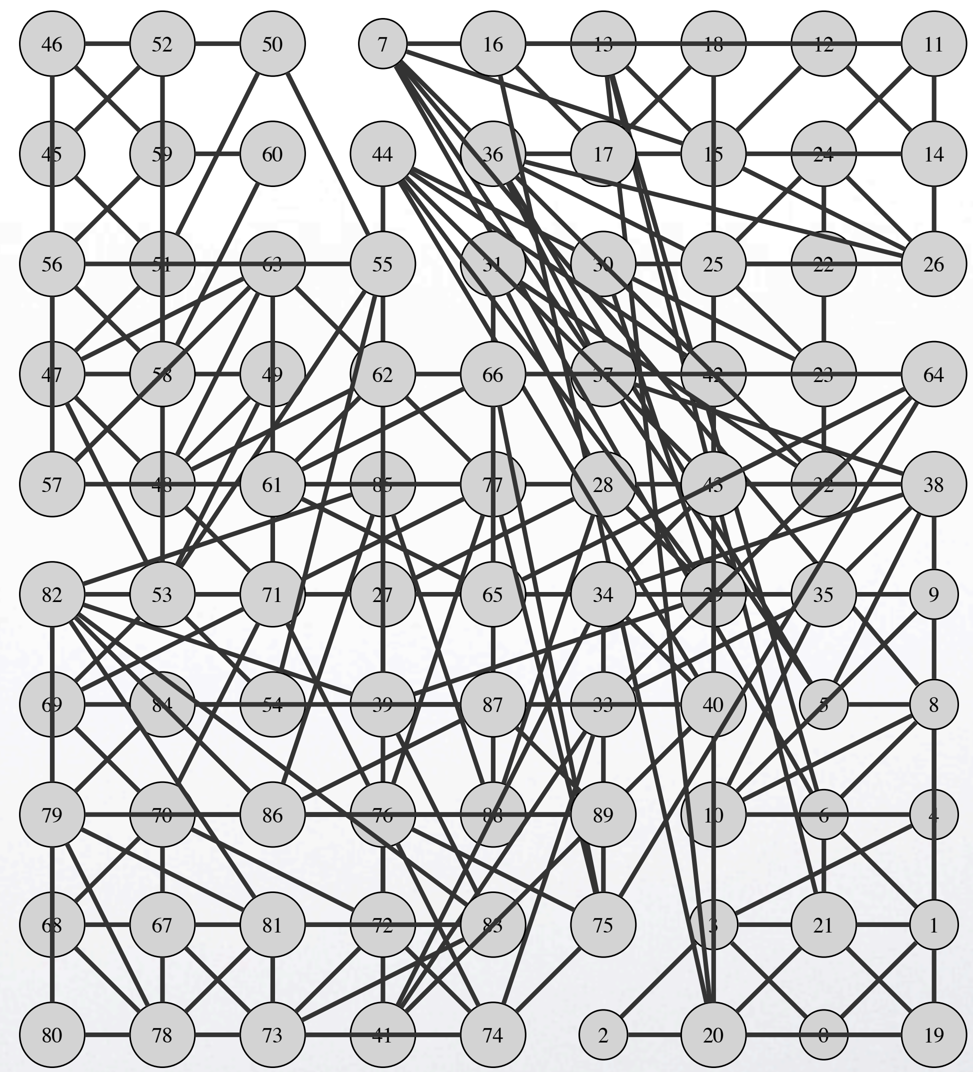
With coordinate information

- Affine Mapping (AFFN)
  - Stretch/shrink the object graph (based on coordinates of nodes) to map it on to the processor grid
  - In case of conflicts for the same processor, spiral around that processor
- Corners to Center (COCE)
  - Use four corners of the object graph based on coordinates
  - Start mapping simultaneously from all sides
    - Either a simple BFT-type scheme
    - Or a MHT-style heuristic

Mapping visualization

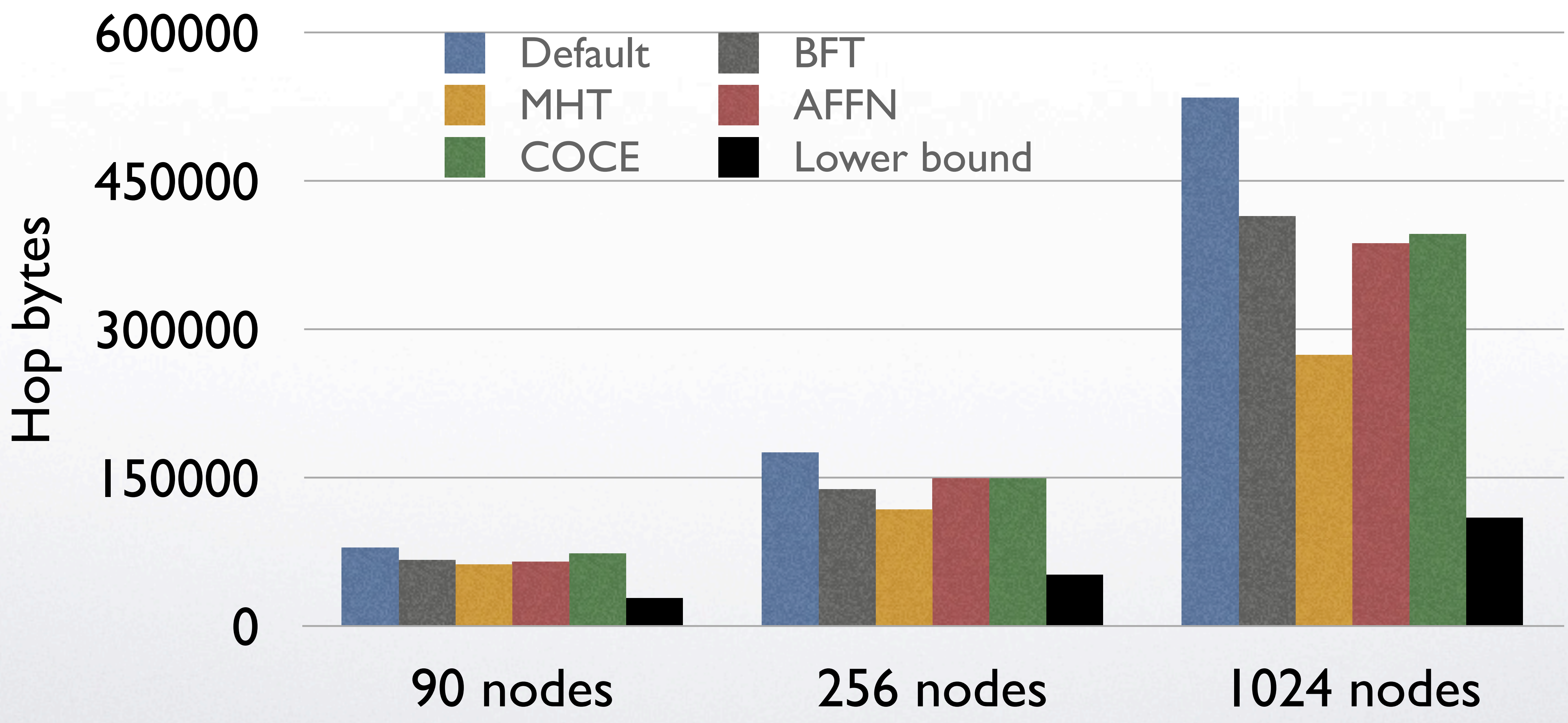


AFFN



COCE

Results: simple2D



Completely Distributed Mapping

- Problem (in content of Charm++):
  - n objects to be placed on p processors (n much greater than p)
  - Computational loads of objects are distributed
  - Each object should make its decision by itself
- Start with simple cases:
  - 1D ring communication
  - 2D stencil communication

Distributed strategies

- 1D ring to a line:
  - Perform a parallel prefix sum between chares and send total load to all objects (chares)
  - Each chare now decides which processor it should be on
- 2D stencil to a 2D mesh:
  - Linearize using Hilbert ordering
  - Perform 1D parallel prefix
- Or perform a parallel prefix in 2D (on all rows and columns)
  - Gives (x, y) coordinates for processor on which the node should go

Summary and Future Work

- Developing an automatic mapping framework
    - Topology discovery: Topology Manager API
    - Pattern matching
        - Regular graphs
        - Irregular graphs
    - Suite of heuristics for mapping
- Completely distributed strategies
- Topology aware hierarchical load balancers (NAMD)