Automated Mapping of Regular Communication Graphs on Mesh Interconnects

Abhinav Bhatele, Gagan Gupta, Laxmikant V. Kale and
I-Hsin Chung

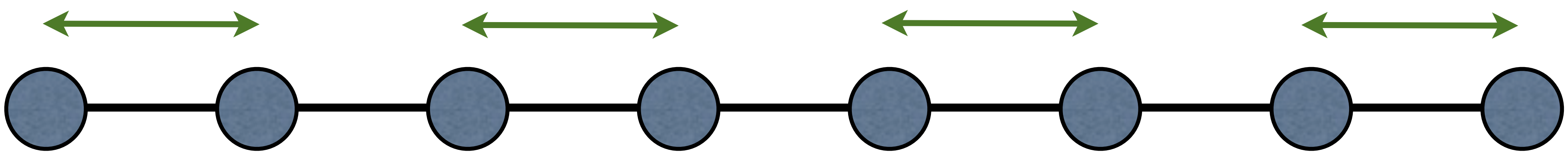
December 20th, 2010

हाइपीसी
HiPC

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC

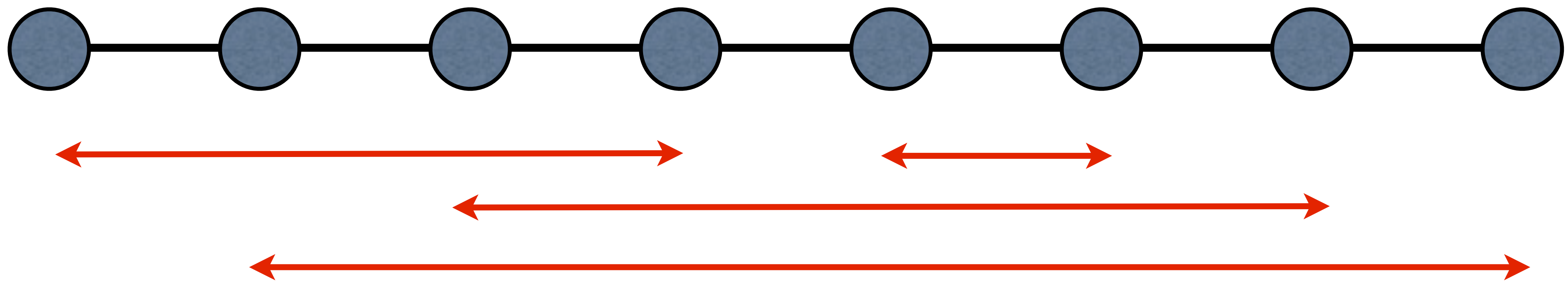
Motivation

- Running a parallel application on a linear array of processors:



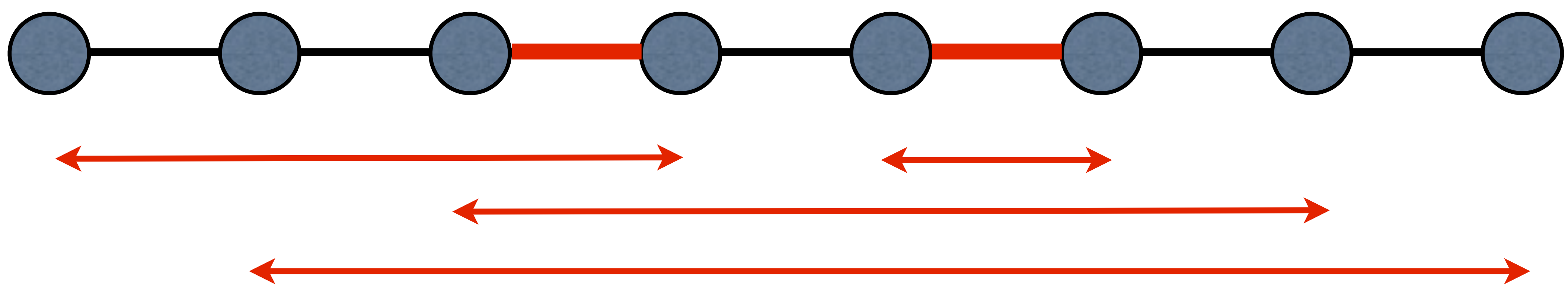
Motivation

- Running a parallel application on a linear array of processors:



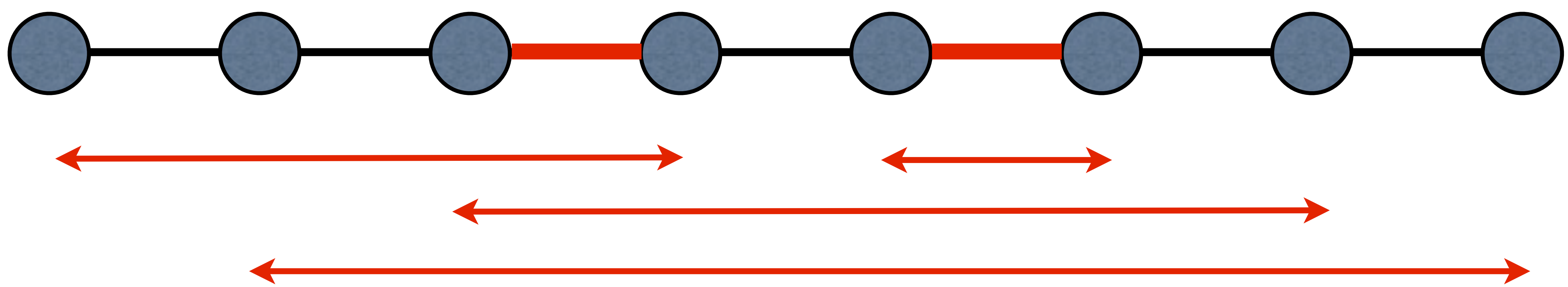
Motivation

- Running a parallel application on a linear array of processors:



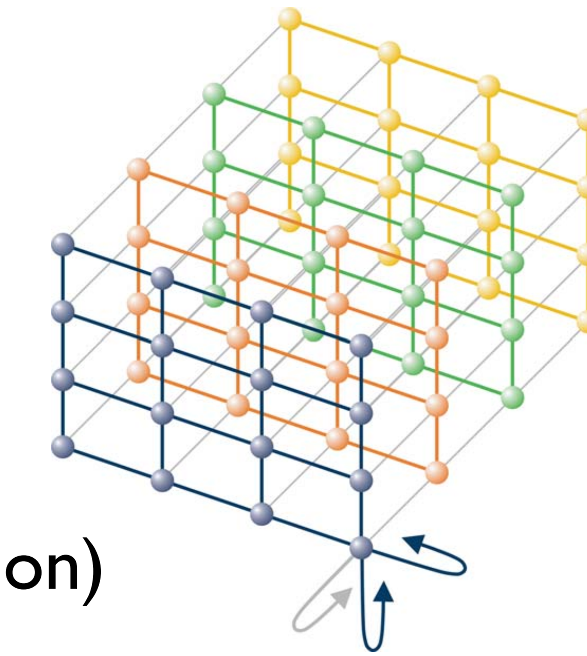
Motivation

- Running a parallel application on a linear array of processors:

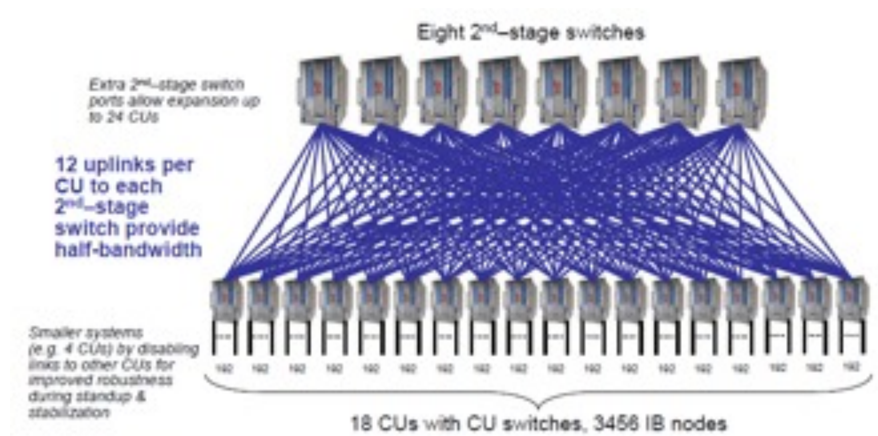


- Typical communication is between random pairs of processors simultaneously

Interconnect Topologies



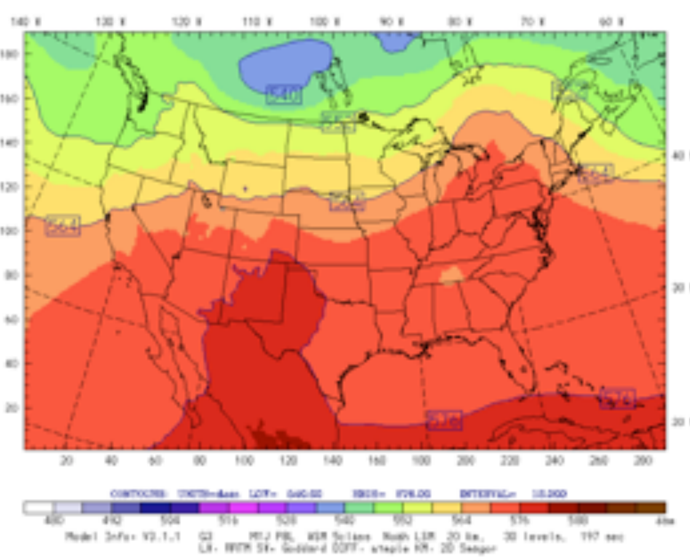
- Three dimensional meshes
  - 3D Torus: Blue Gene/L, Blue Gene/P, Cray XT4/5
- Trees
  - Fat-trees (Infiniband) and CLOS networks (Federation)
- Dense Graphs
  - Kautz Graph (SiCortex), Hypercubes
- Future Topologies?
  - Blue Waters, Blue Gene/Q

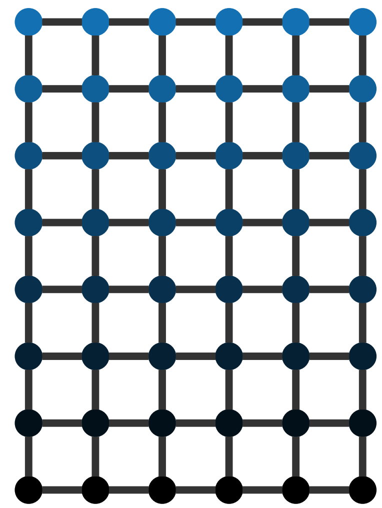


Roadrunner Technical Seminar Series, March 13th 2008, Ken Koch, LANL

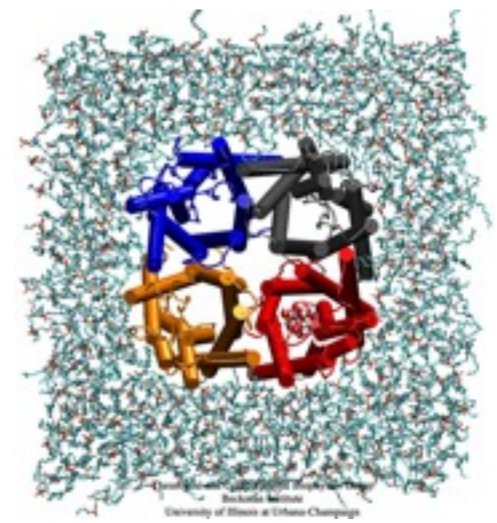
Application Topologies

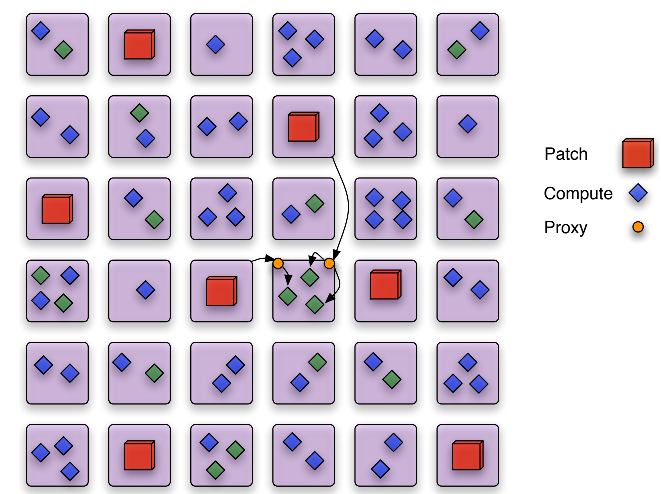
http://wrf-model.org/plots/realtime_main.php

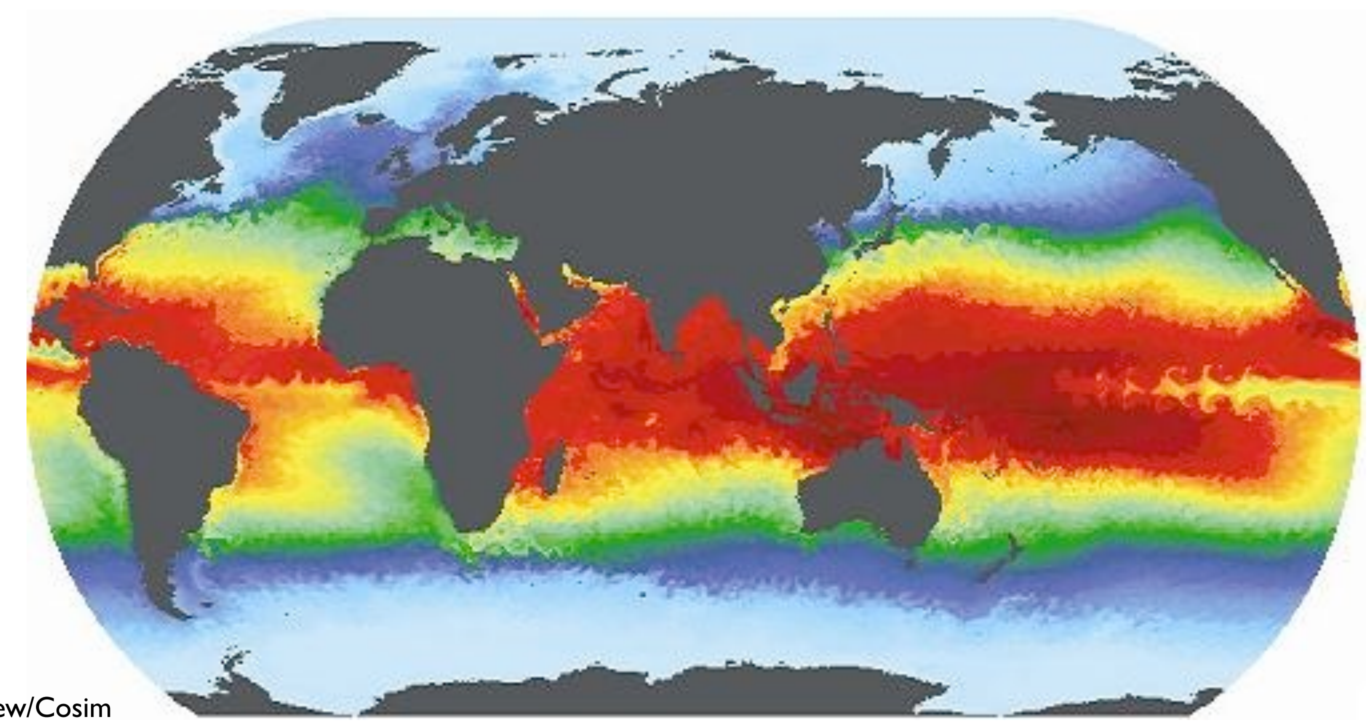




http://www.ks.uiuc.edu/Gallery/Science/



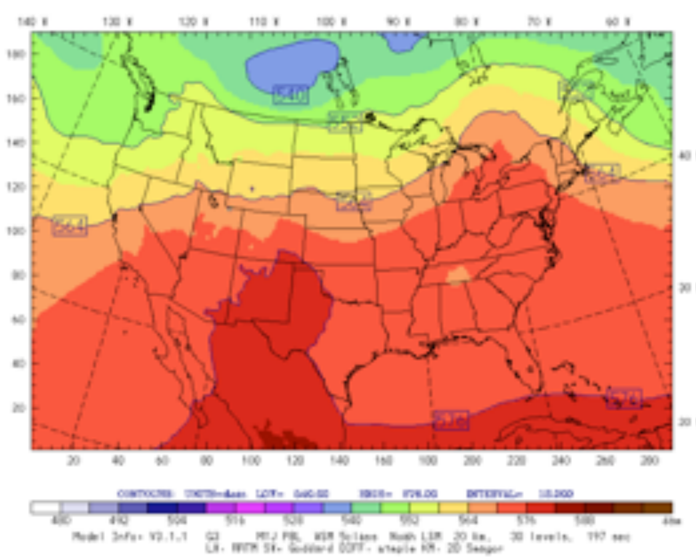


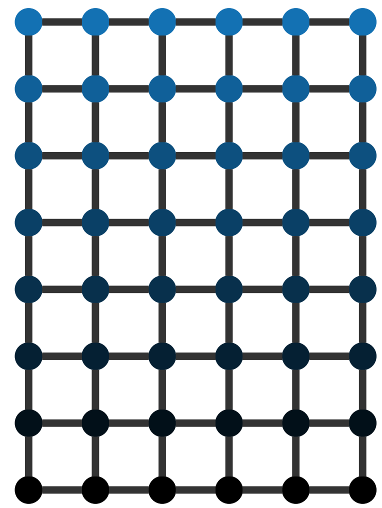


http://oceans11.lanl.gov/twiki/bin/view/Cosim

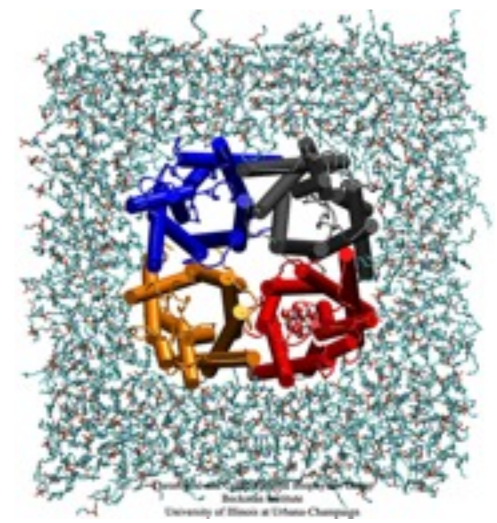
Application Topologies

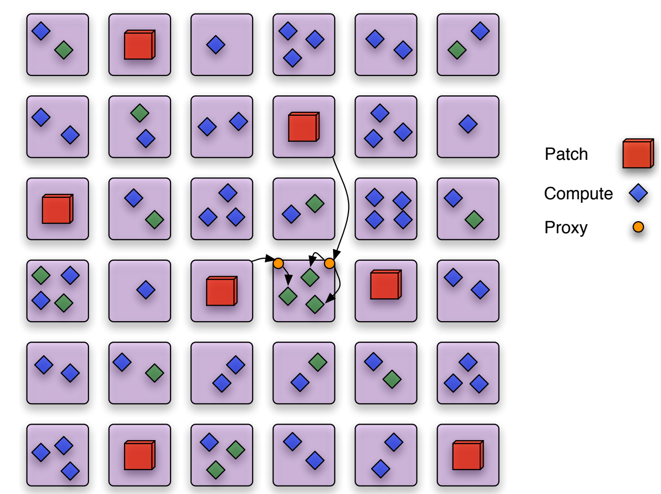
http://wrf-model.org/plots/realtime_main.php



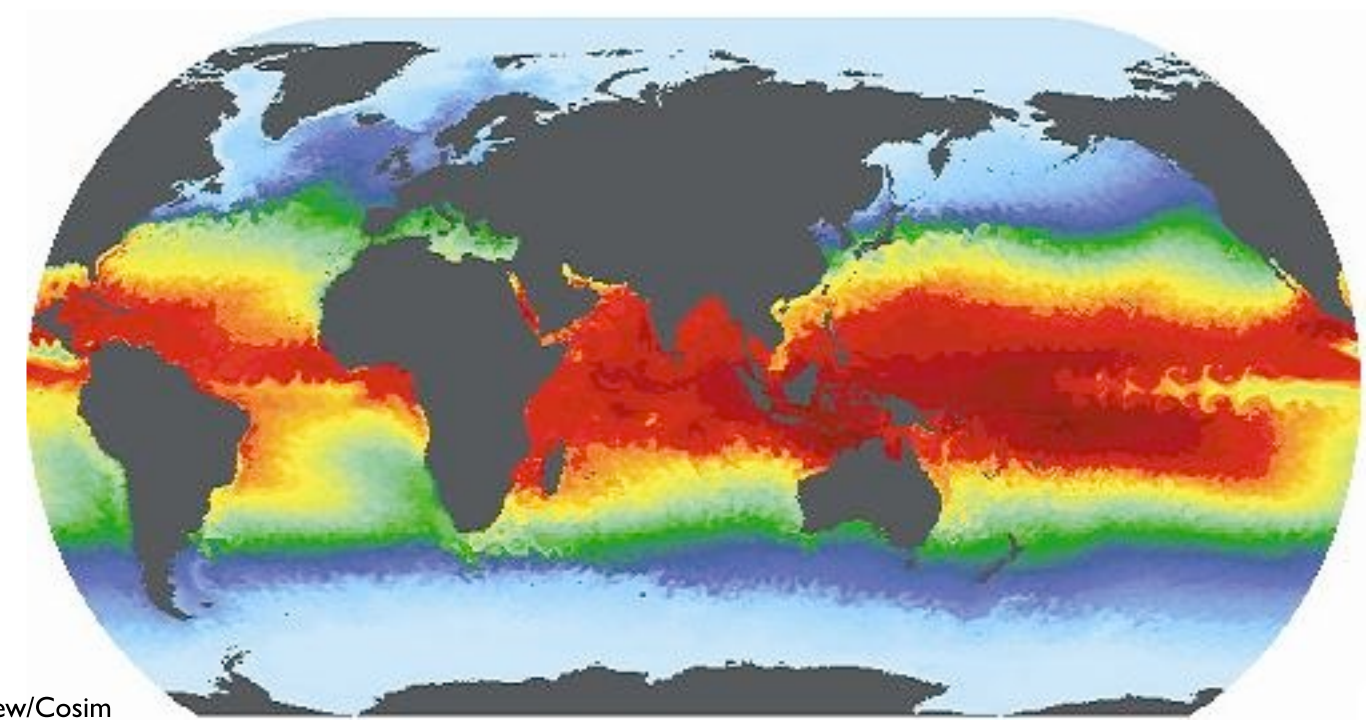


http://www.ks.uiuc.edu/Gallery/Science/





We want to map communicating objects closer to one another



http://oceans11.lanl.gov/twiki/bin/view/Cosim

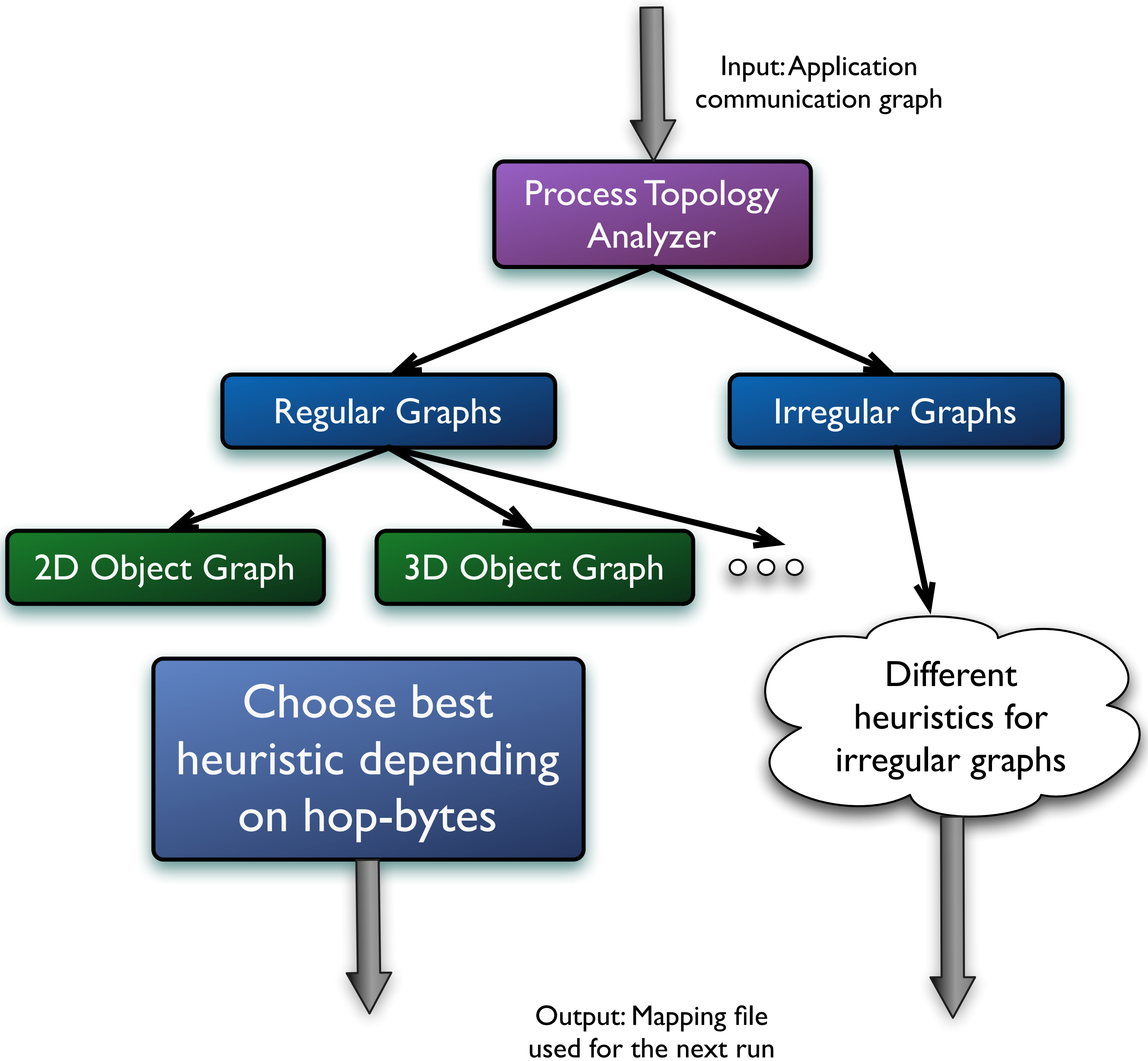
The Mapping Problem

- 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:
  - Minimize communication traffic and hence contention
  - Balance computational load (when n > p)

Solution - Mapping Framework

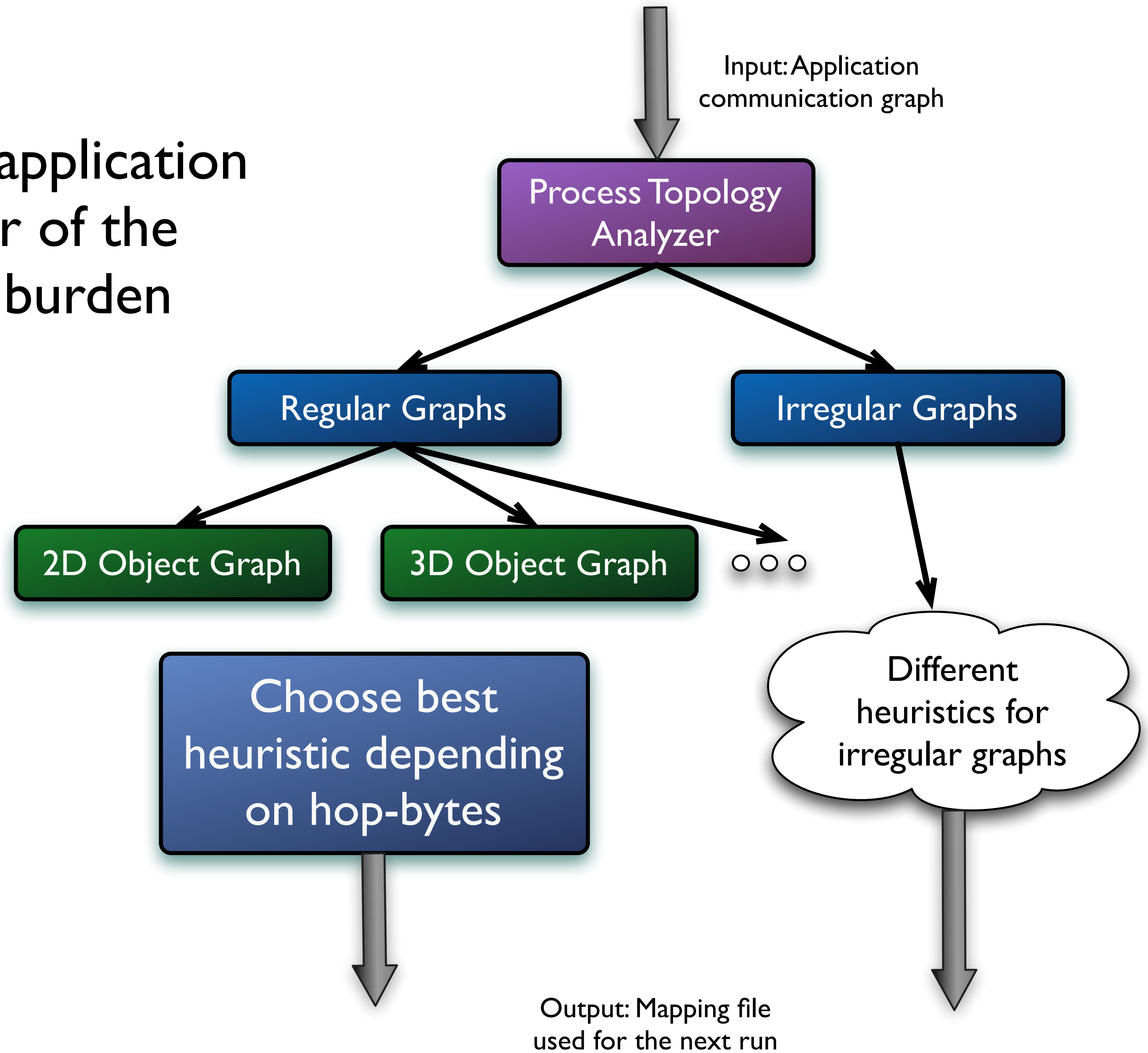
- Input - communication graph of the application and processor topology of the allocated job partition
- Output - mapping of processes/objects to physical processors
- Parallel applications can be classified into:
  - regular/structured: n-dimensional near-neighbor (e.g. POP, WRF)
  - irregular: arbitrary communication
- We focus on regular communication in this paper

Automatic Mapping Framework



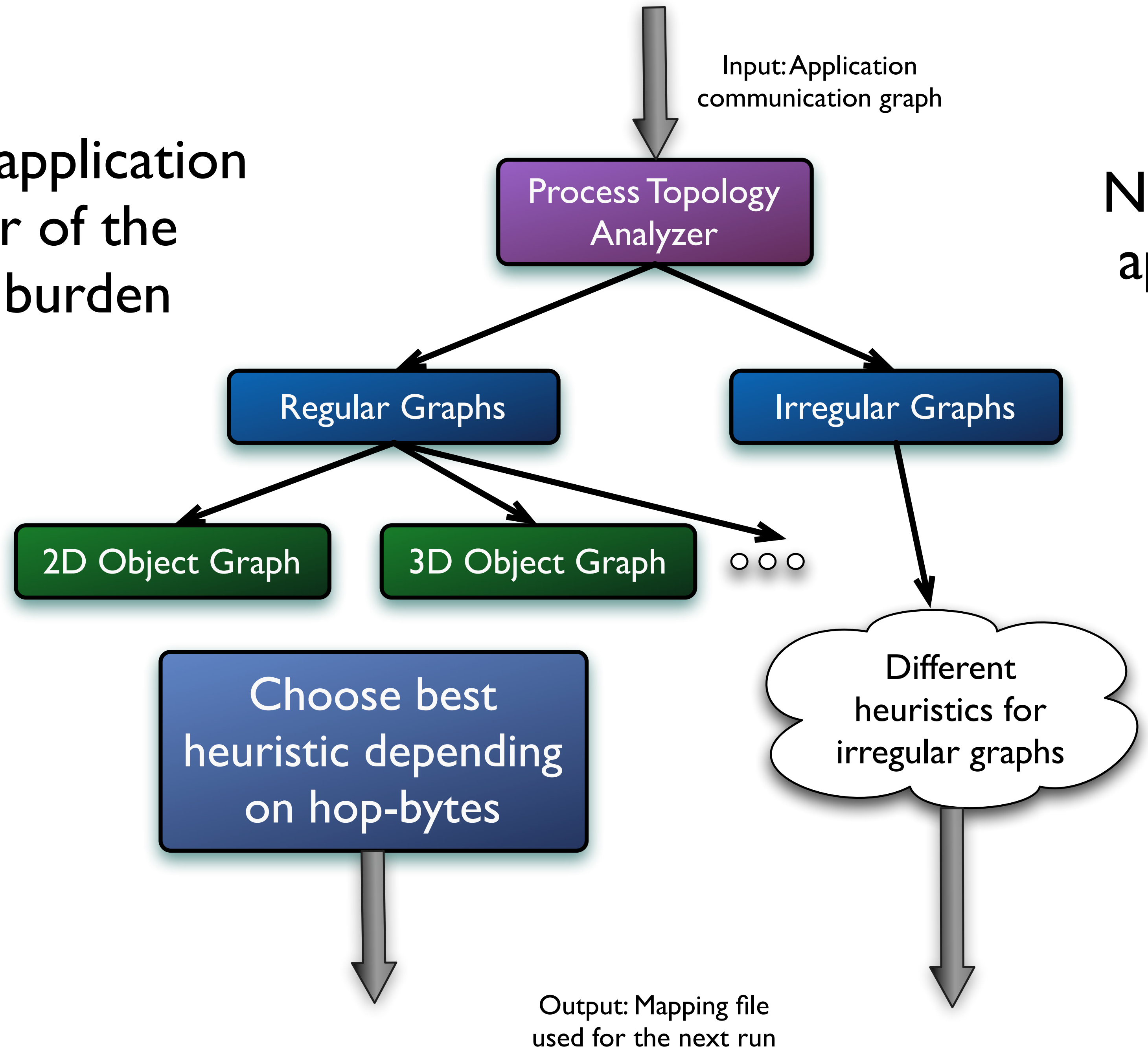
Automatic Mapping Framework

Relieve the application developer of the mapping burden



Automatic Mapping Framework

Relieve the application developer of the mapping burden



No change to the application code

Machine 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

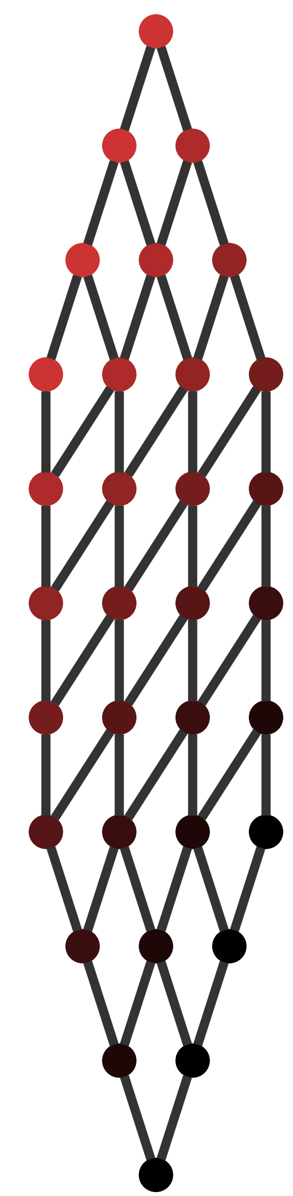
Application communication graph

- Several ways to obtain the graph
- MPI applications:
  - Profiling tools (IBM's HPCT tools)
  - Collect information using the PMPI interface
  - Manually provided by the application end user
- Charm++ applications:
  - Instrumentation at runtime
  - Profiling tools (HPCT): when n = p

Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

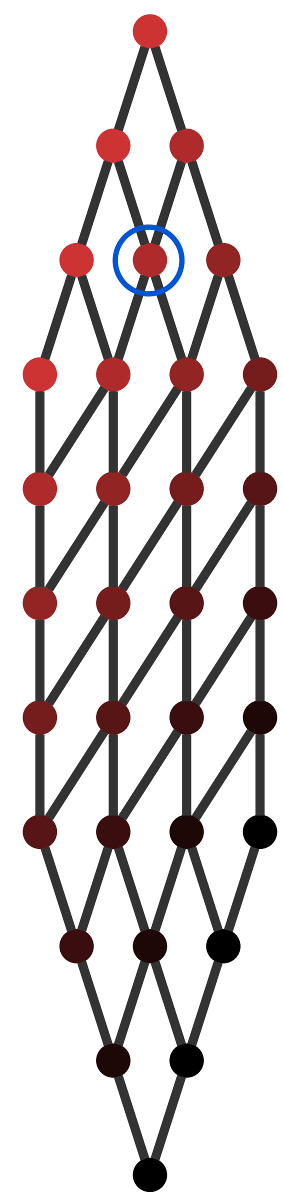
Input: $CM_{n,n}$ (communication matrix)
Output: $isRegular$ (boolean, true if communication is regular)
dims[ ] (dimensions of the regular communication graph)
for $i = 1$ to n **do**
find the maximum number of neighbors for any rank in $CM_{i,n}$
end for
if max neighbors ≤ 5 **then**
// this might be a case of regular 2D communication
select an arbitrary rank $start_{pe}$ find its distance from its neighbors
$dist$ = difference between ranks of $start_{pe}$ and its top or bottom neighbor
for $i := 1$ to n **do**
if distance of all ranks from their neighbors == 1 or $dist$ **then**
$isRegular$ = true
dim[0] = $dist$
dim[1] = $n/dist$
end if
end for
end if



Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

Input: $CM_{n,n}$ (communication matrix)
Output: $isRegular$ (boolean, true if communication is regular)
dims[ ] (dimensions of the regular communication graph)
for $i = 1$ to n **do**
find the maximum number of neighbors for any rank in $CM_{i,n}$
end for
if max neighbors ≤ 5 **then**
// this might be a case of regular 2D communication
select an arbitrary rank $start_{pe}$ find its distance from its neighbors
$dist$ = difference between ranks of $start_{pe}$ and its top or bottom neighbor
for $i := 1$ to n **do**
if distance of all ranks from their neighbors == 1 or $dist$ **then**
$isRegular$ = true
dim[0] = $dist$
dim[1] = $n/dist$
end if
end for
end if

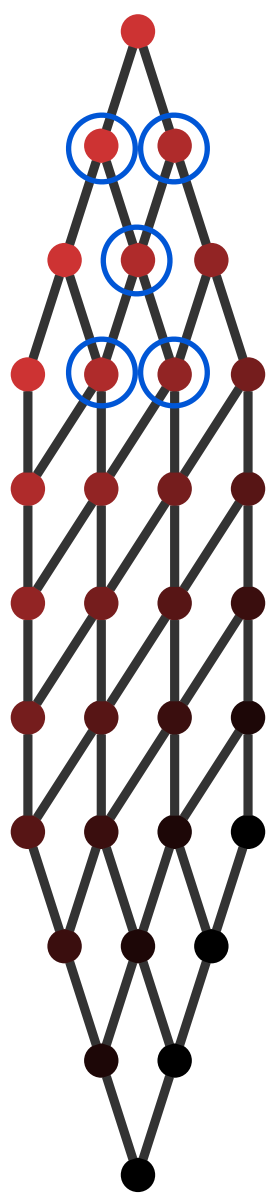


Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

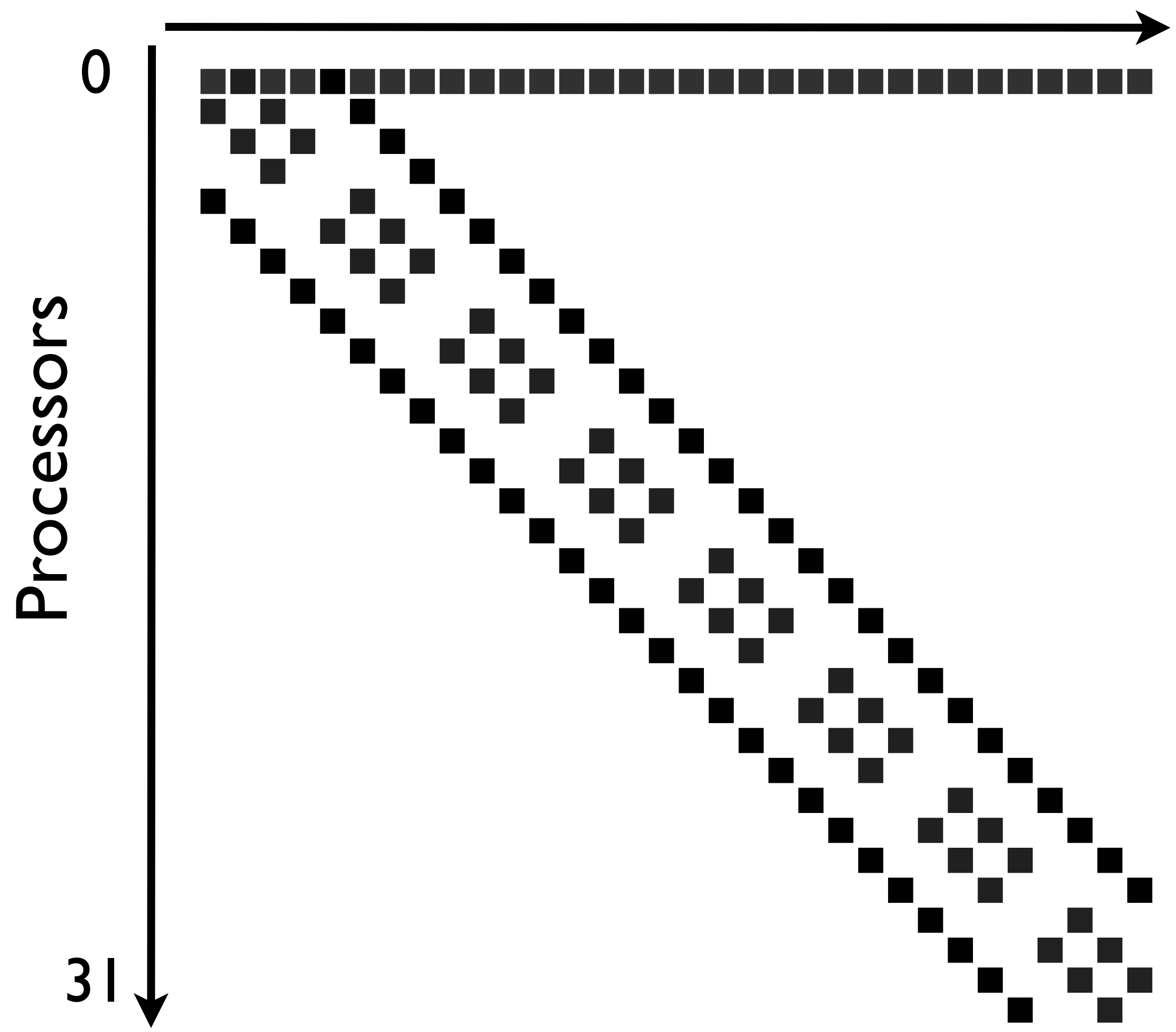
Input: $CM_{n,n}$ (communication matrix)
Output: $isRegular$ (boolean, true if communication is regular)
dims[ ] (dimensions of the regular communication graph)

for $i = 1$ to n **do**
  find the maximum number of neighbors for any rank in $CM_{i,n}$
end for
if max neighbors ≤ 5 **then**
  // this might be a case of regular 2D communication
  select an arbitrary rank $start_{pe}$ find its distance from its neighbors
  $dist$ = difference between ranks of $start_{pe}$ and its top or bottom neighbor
  for $i := 1$ to n **do**
    if distance of all ranks from their neighbors == 1 or $dist$ **then**
      $isRegular$ = true
      dim[0] = $dist$
      dim[1] = $n/dist$
    end if
  end for
end if



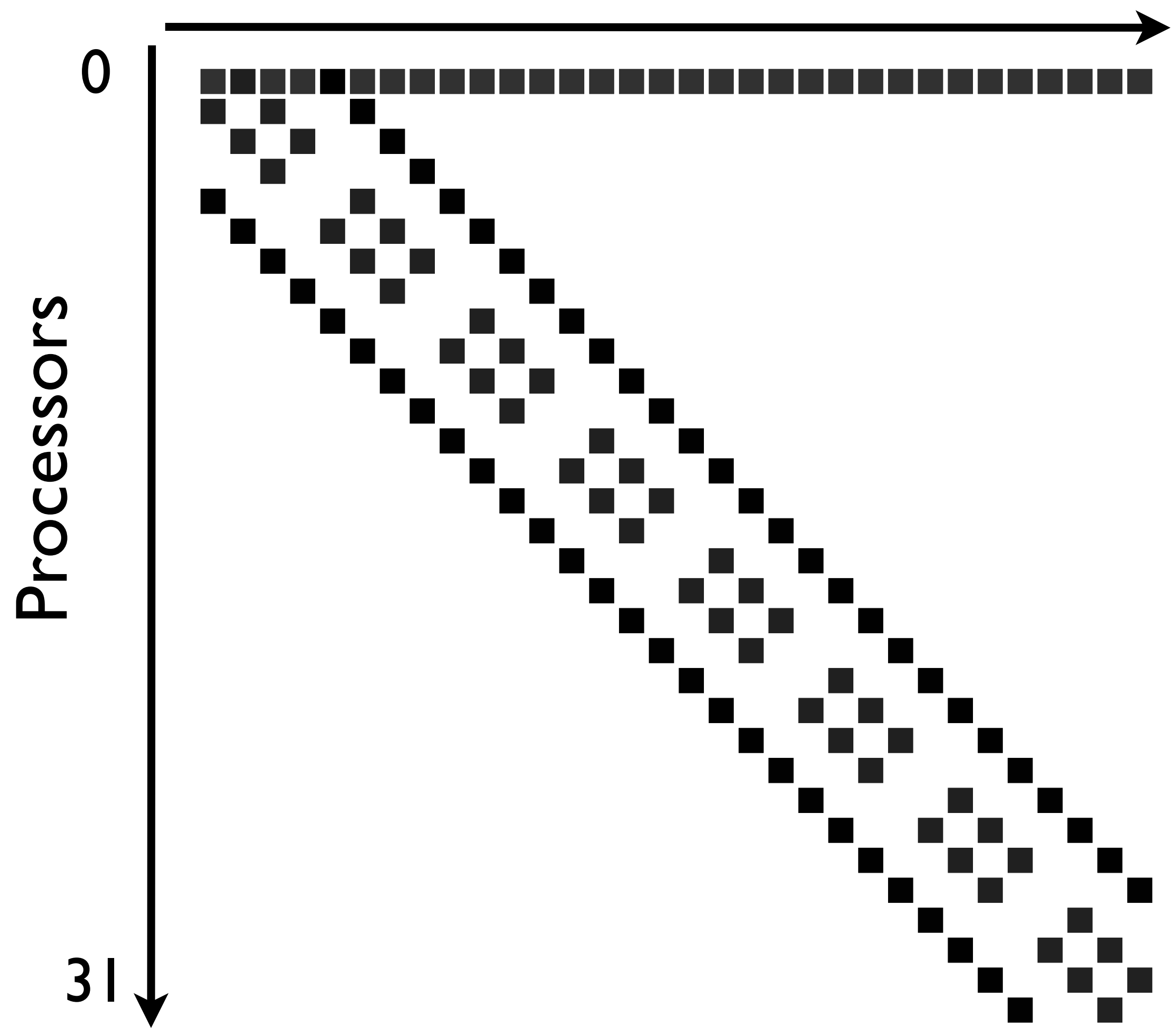
Example

- WRF running on 32 cores of Blue Gene/P



Example

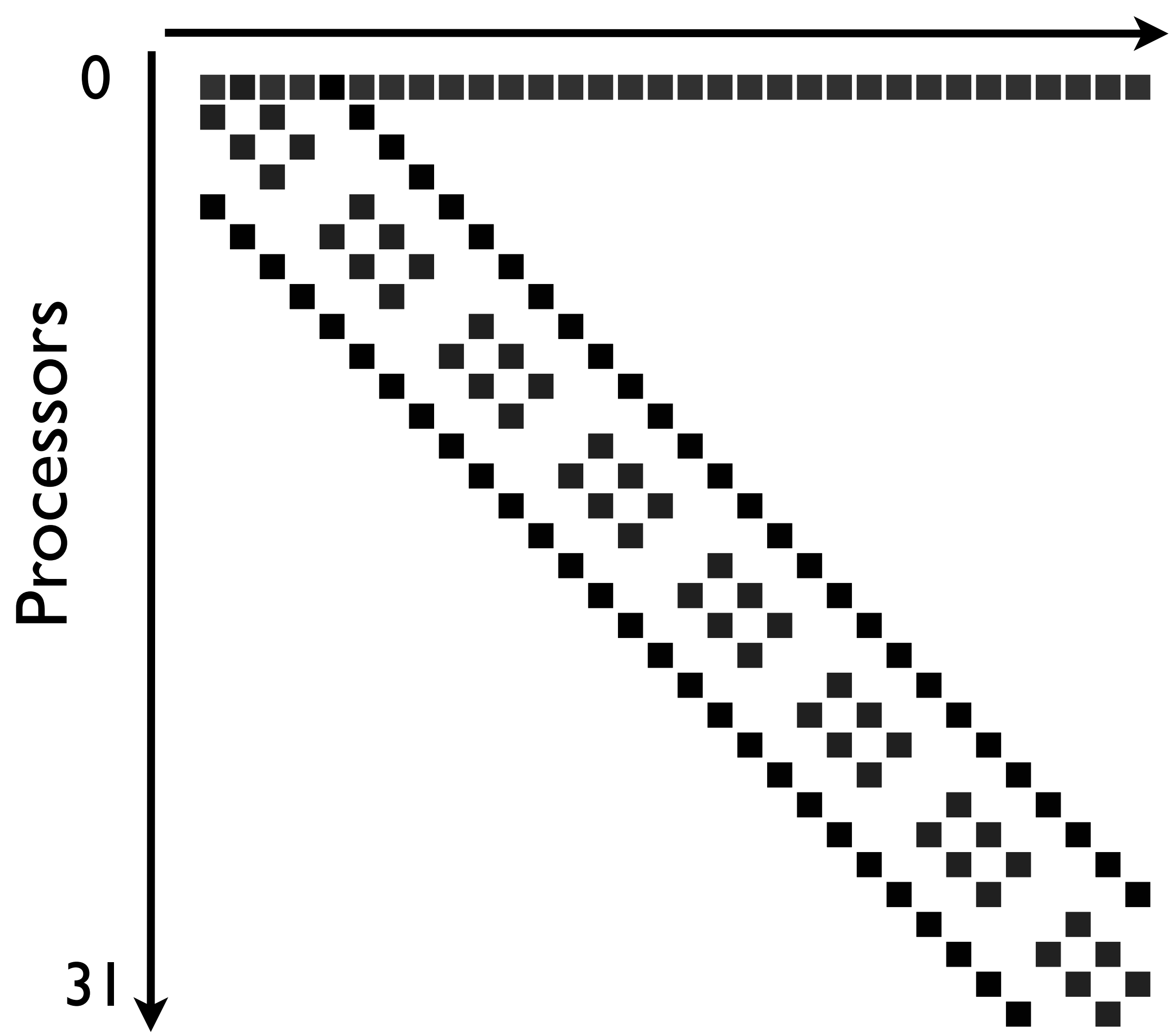
- WRF running on 32 cores of Blue Gene/P

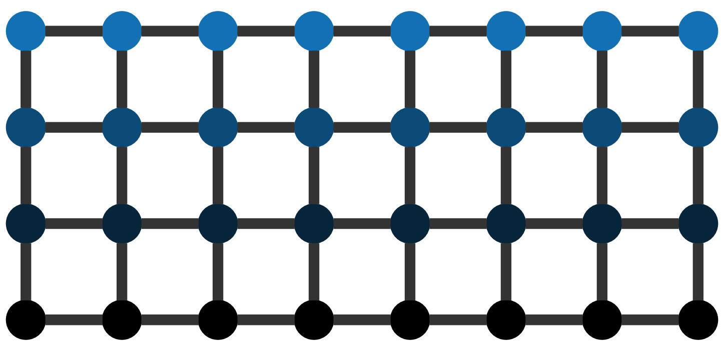


Pattern matching to identify regular communication patterns such as 2D/3D near-neighbor graphs

Example

- WRF running on 32 cores of Blue Gene/P



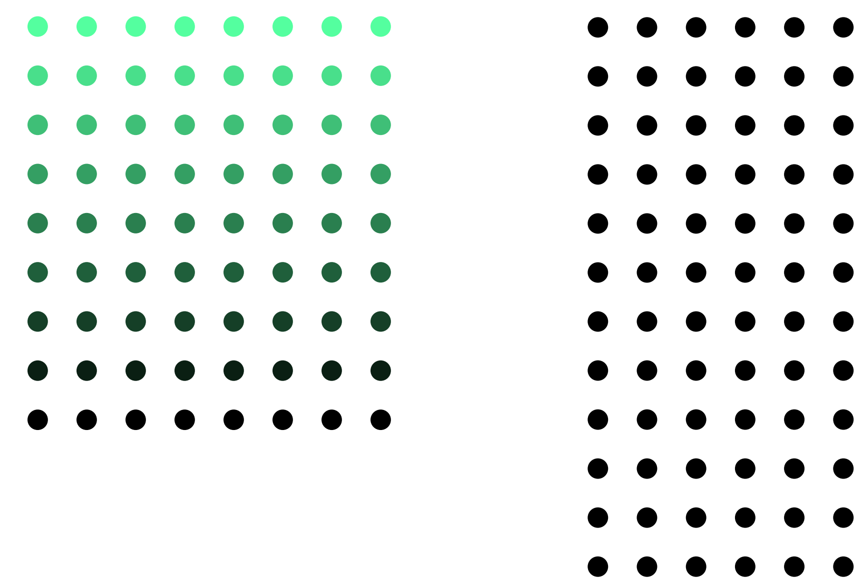


Pattern matching to identify regular communication patterns such as 2D/3D near-neighbor graphs

Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

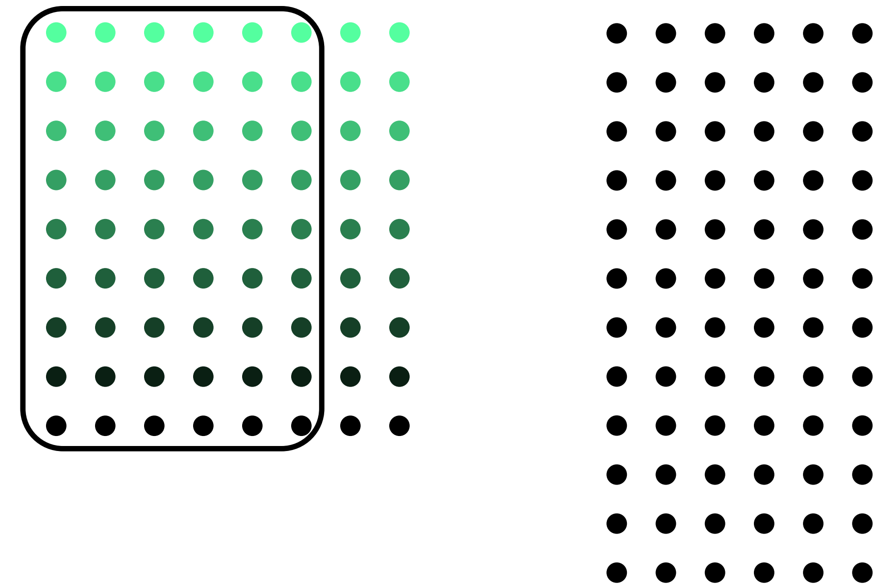
Object Graph: 9 x 8
Processor Graph: 12 x 6



Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

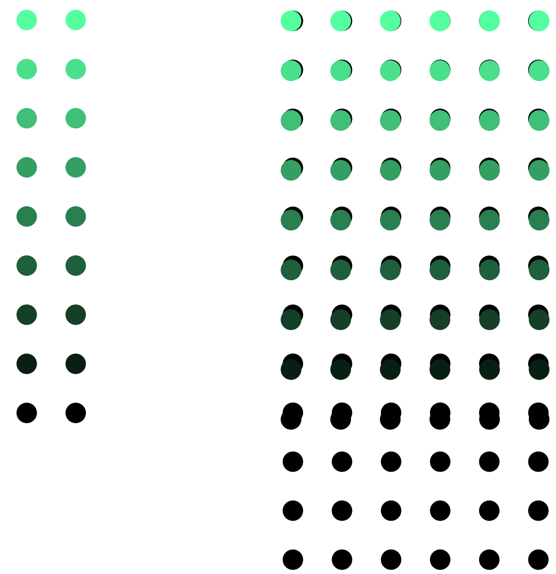
Object Graph: 9 x 8
Processor Graph: 12 x 6



Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

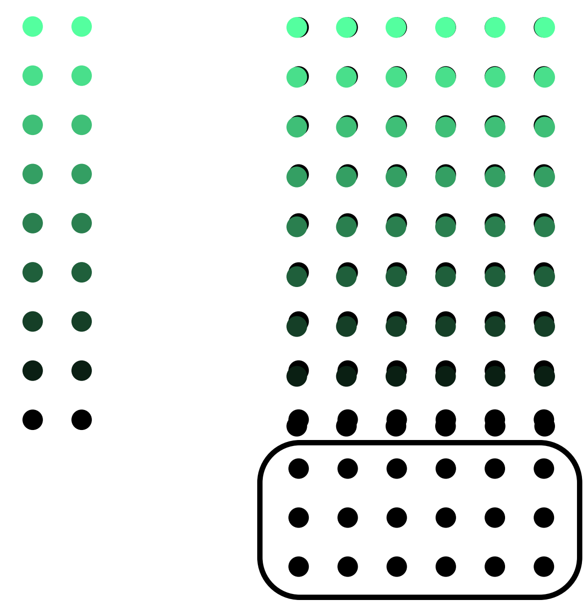
Object Graph: 9 x 8
Processor Graph: 12 x 6



Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

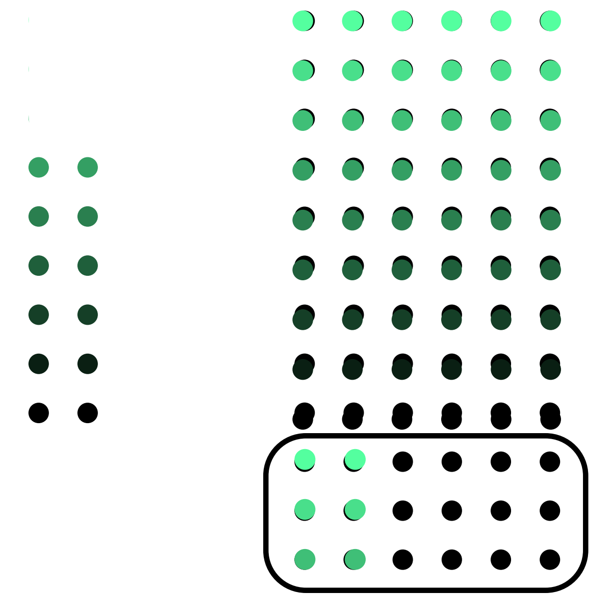
Object Graph: 9 x 8
Processor Graph: 12 x 6



Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

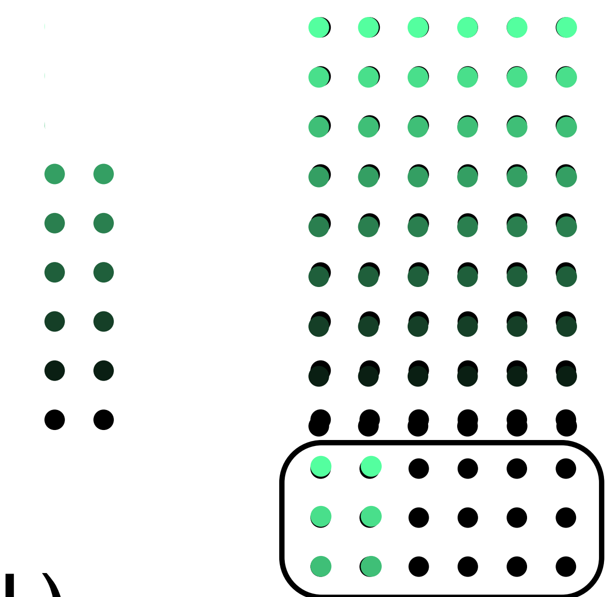
Object Graph: 9 x 8
Processor Graph: 12 x 6



Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6

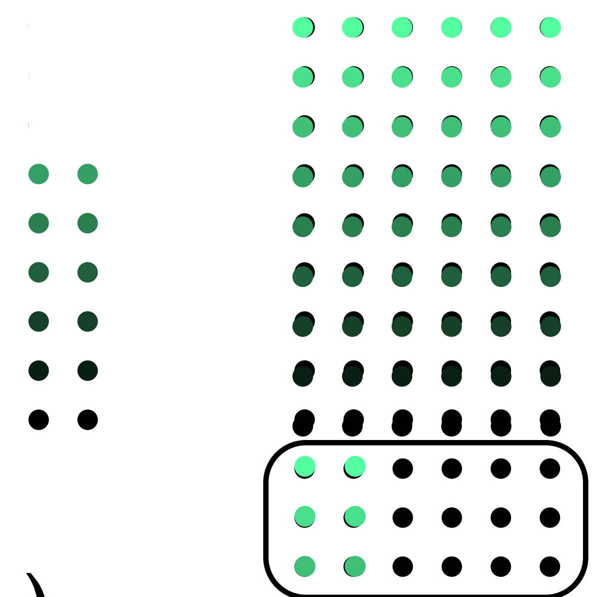


- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call

Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

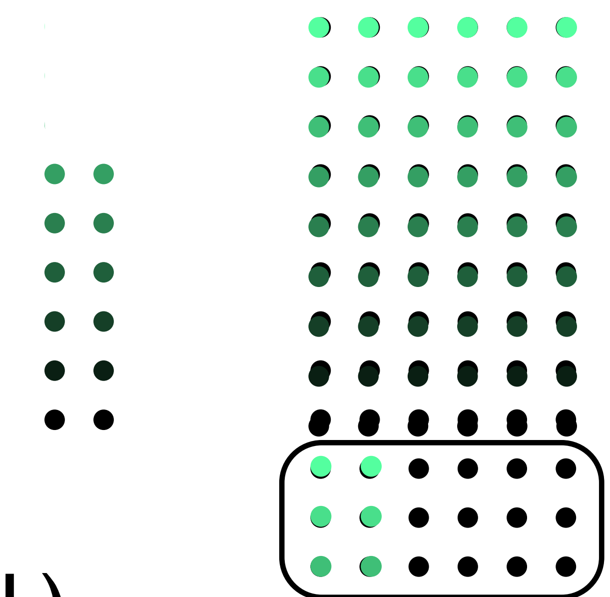
Object Graph: 9 x 8
Processor Graph: 12 x 6



- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call
- Expand from Corner (EXCO)

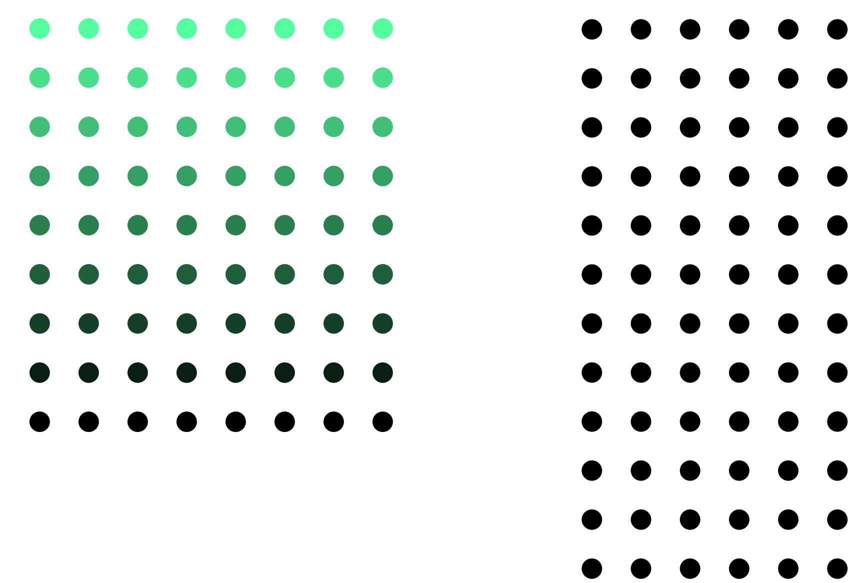
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)



Object Graph: 9 x 8
Processor Graph: 12 x 6

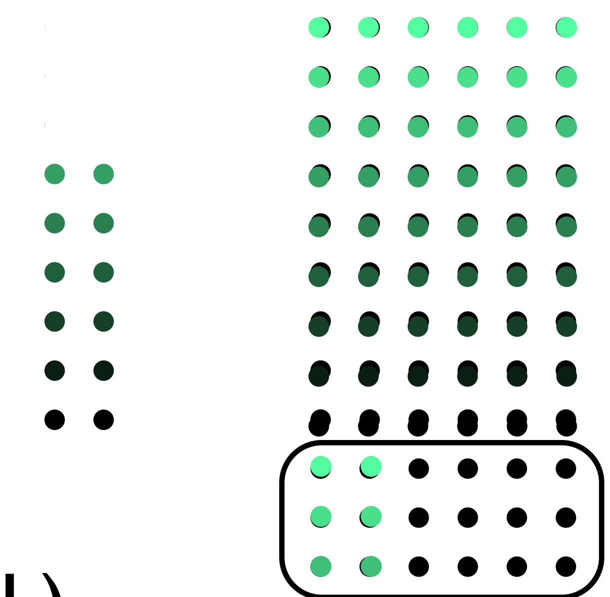
- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call
- Expand from Corner (EXCO)



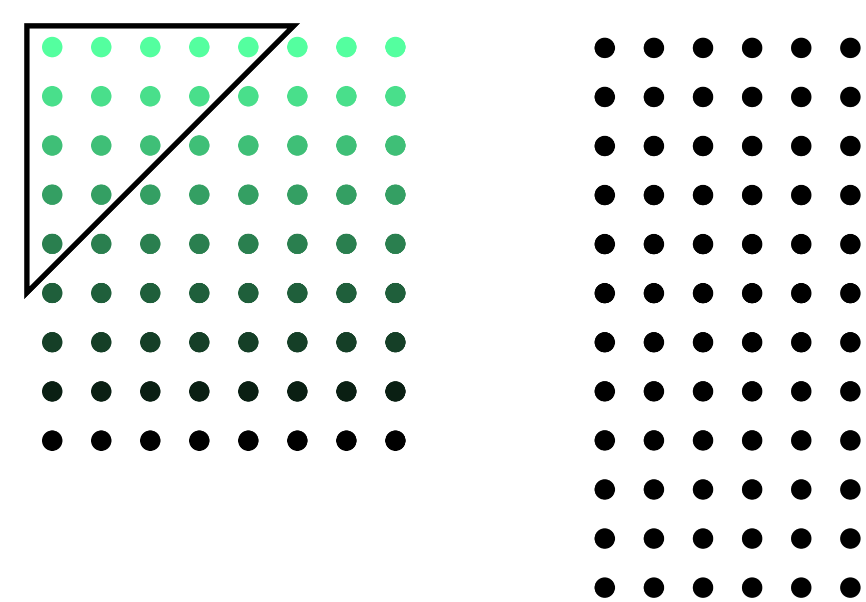
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6

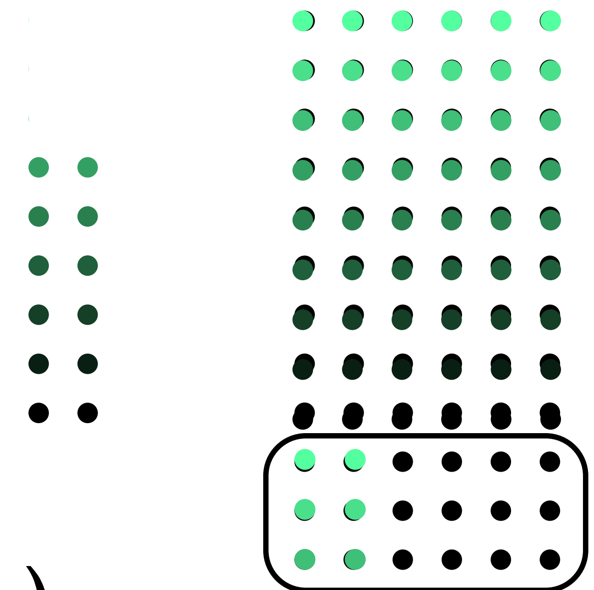


- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call
- Expand from Corner (EXCO)



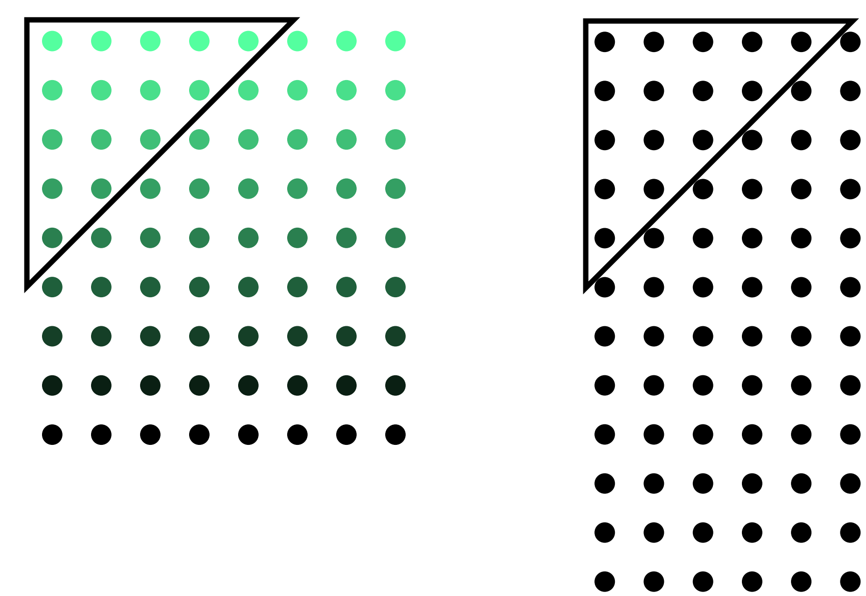
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)



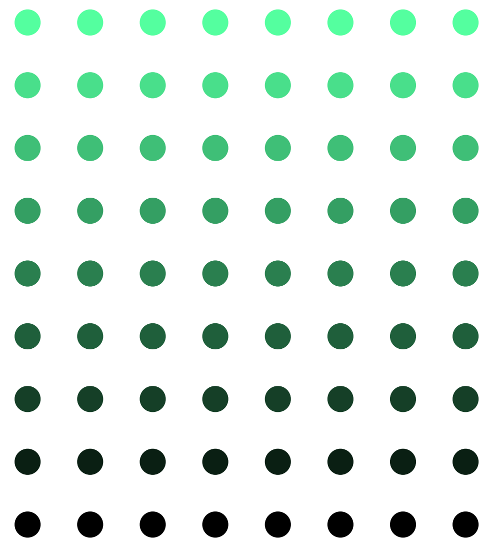
Object Graph: 9 x 8
Processor Graph: 12 x 6

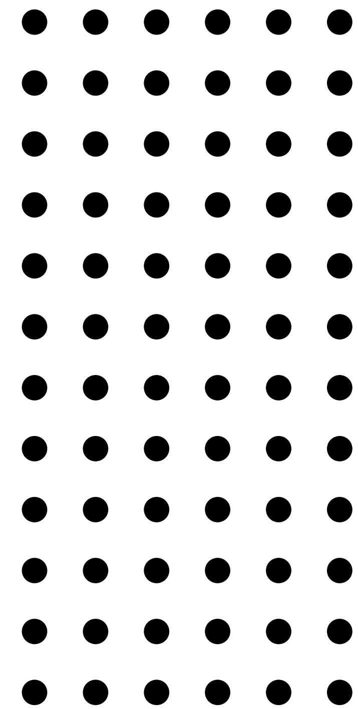
- Maximum Overlap with Alignment (MXOV+AL)
    - Alignment at each recursive call
- Expand from Corner (EXCO)



More heuristics ...

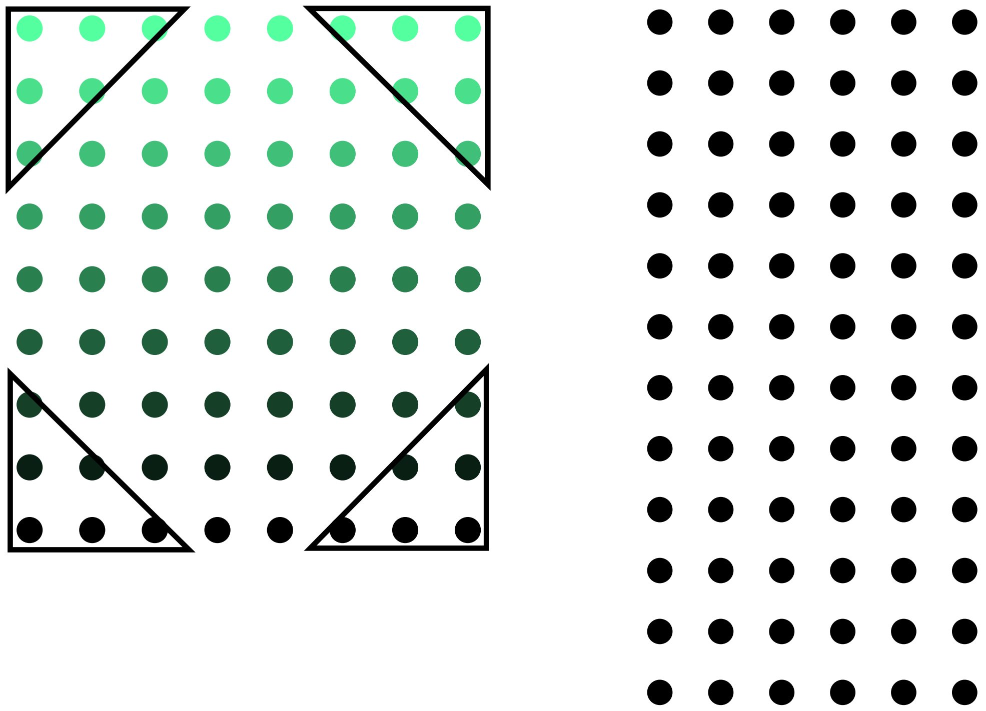
- Corners to Center (COCE)
    - Start simultaneously from all corners





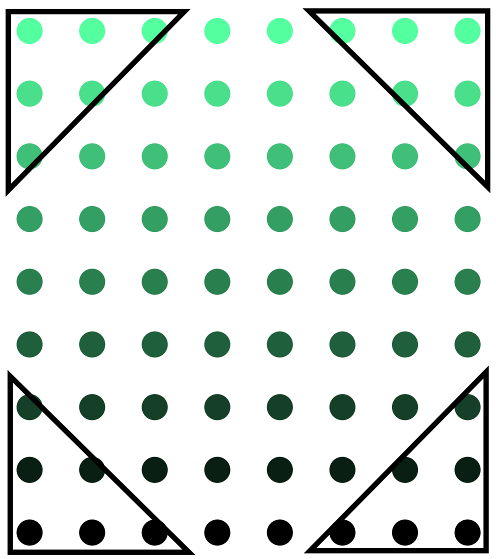
More heuristics ...

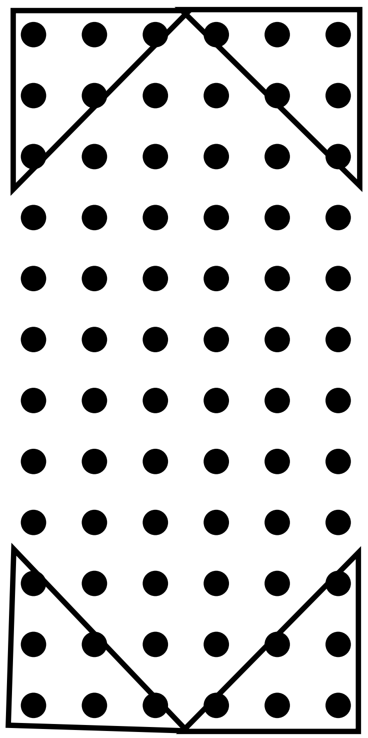
- Corners to Center (COCE)
  - Start simultaneously from all corners



More heuristics ...

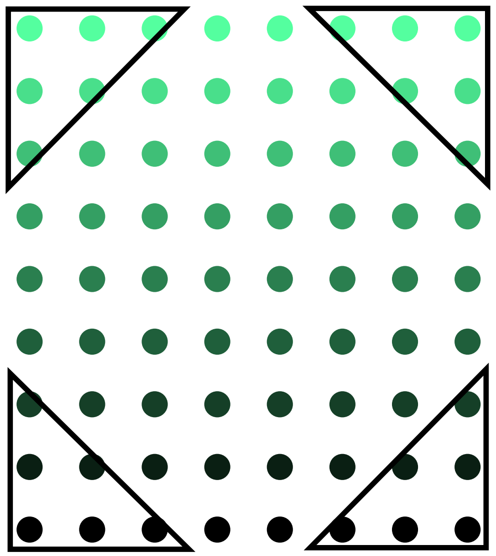
- Corners to Center (COCE)
  - Start simultaneously from all corners

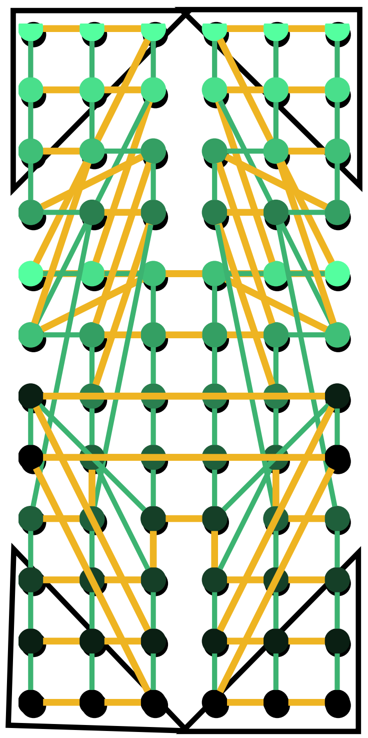




More heuristics ...

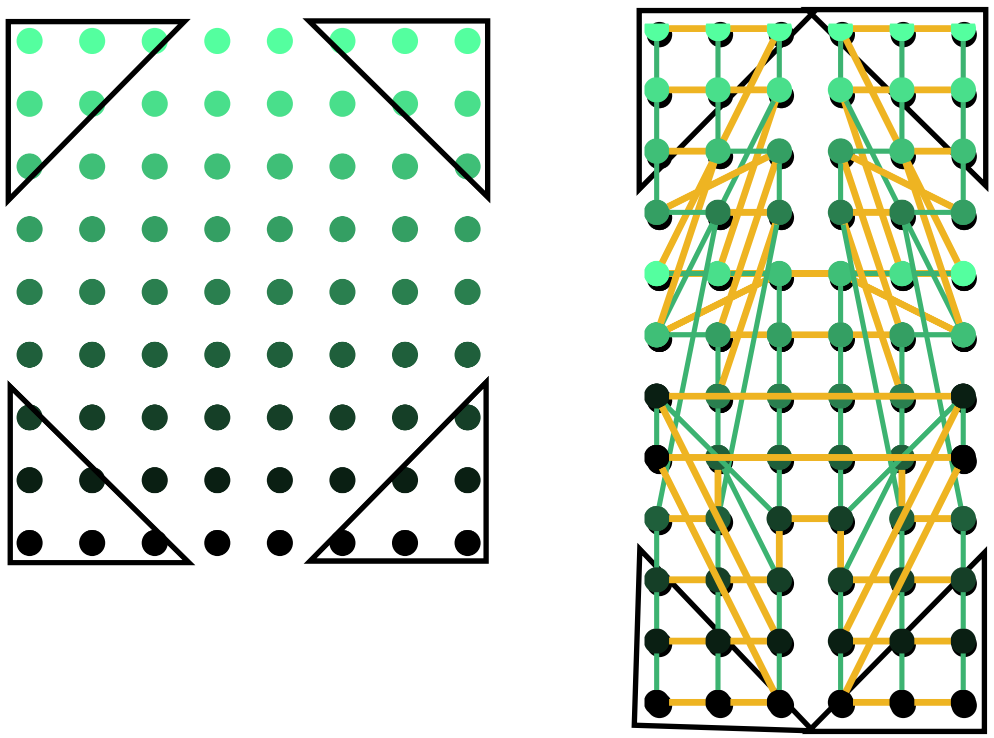
- Corners to Center (COCE)
  - Start simultaneously from all corners



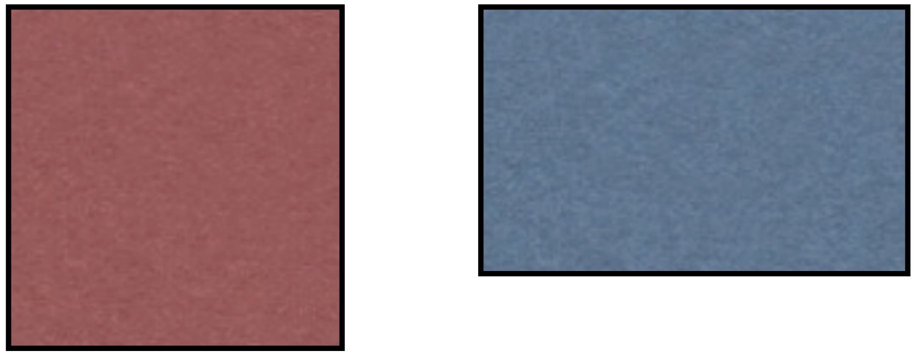


More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners

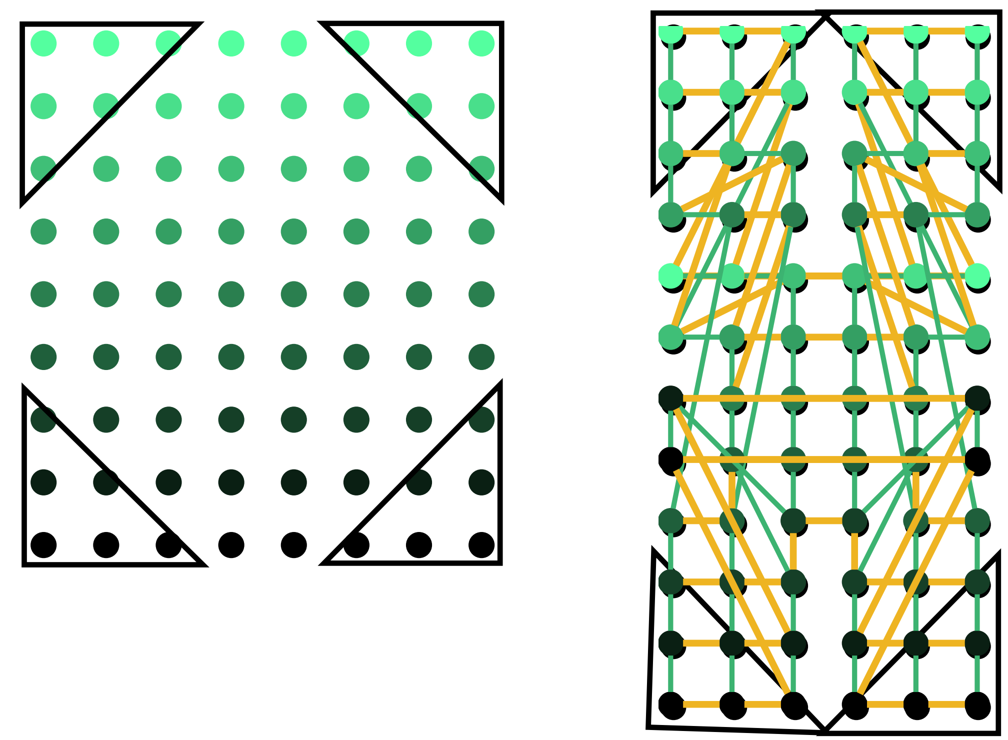


- Affine Mapping (AFFN)

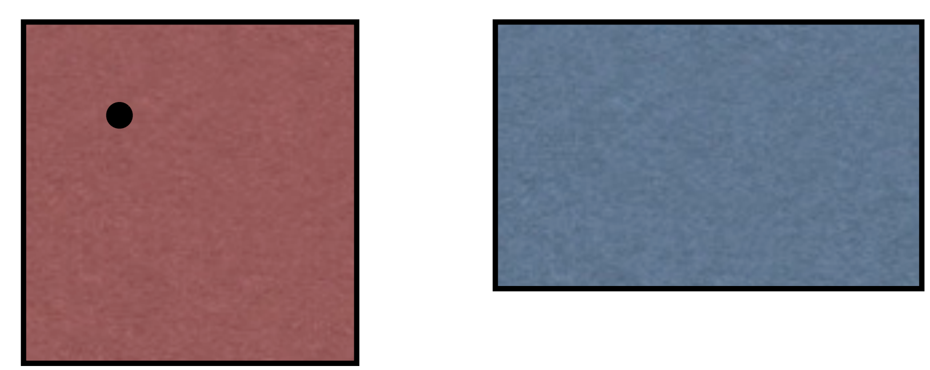


More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners

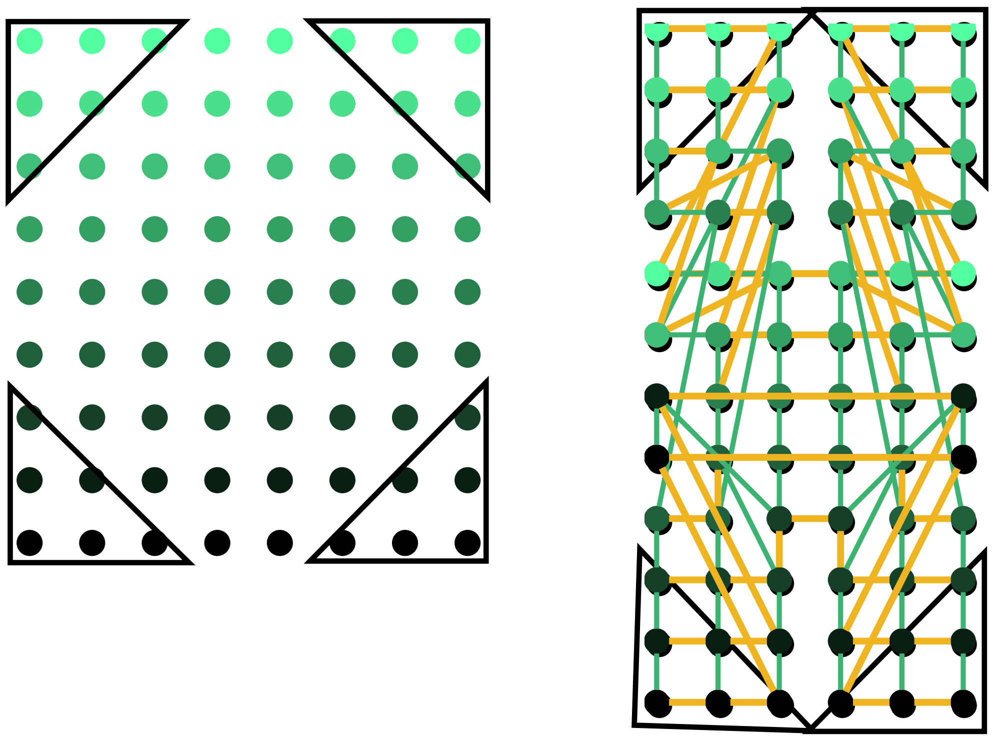


- Affine Mapping (AFFN)



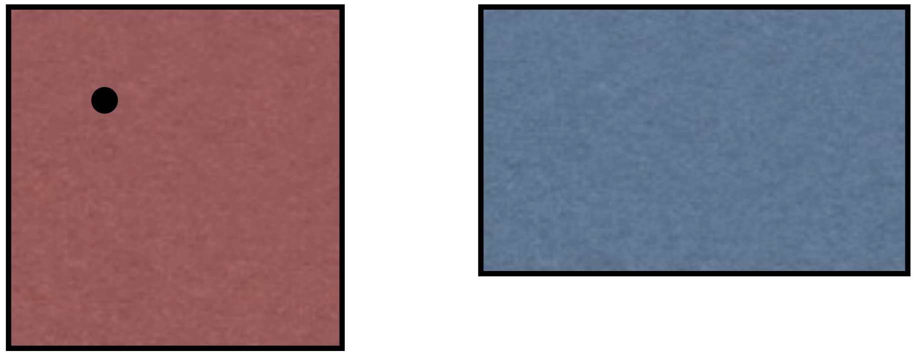
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners



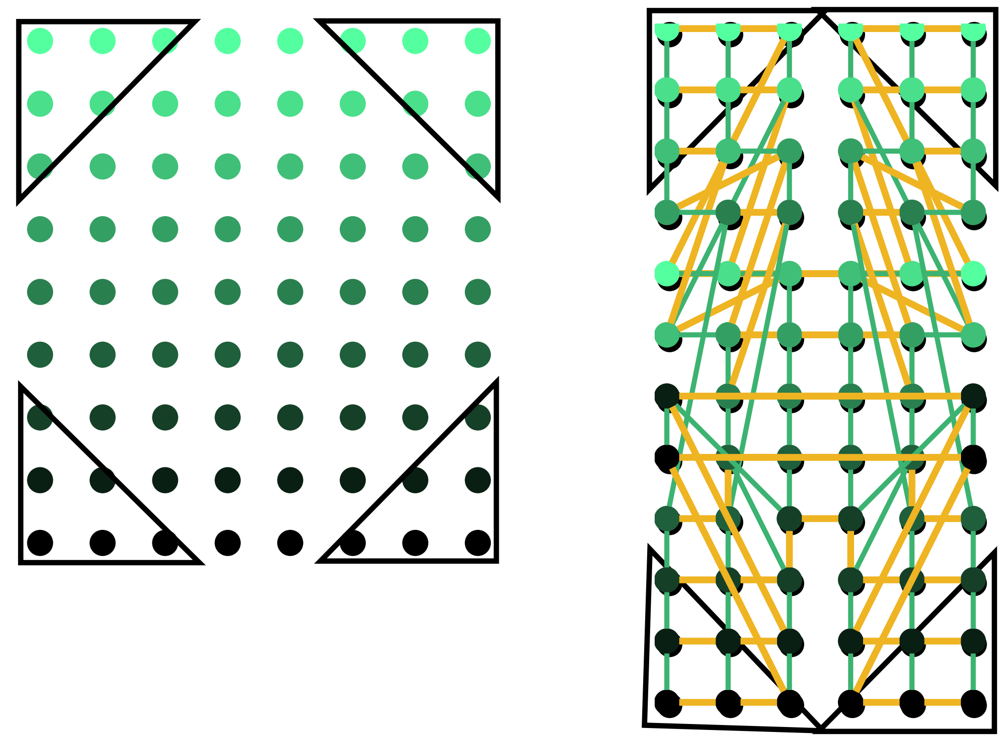
- Affine Mapping (AFFN)

$$(x, y) \rightarrow (\lfloor P_x * \frac{x}{O_x} \rfloor, \lfloor P_y * \frac{y}{O_y} \rfloor)$$



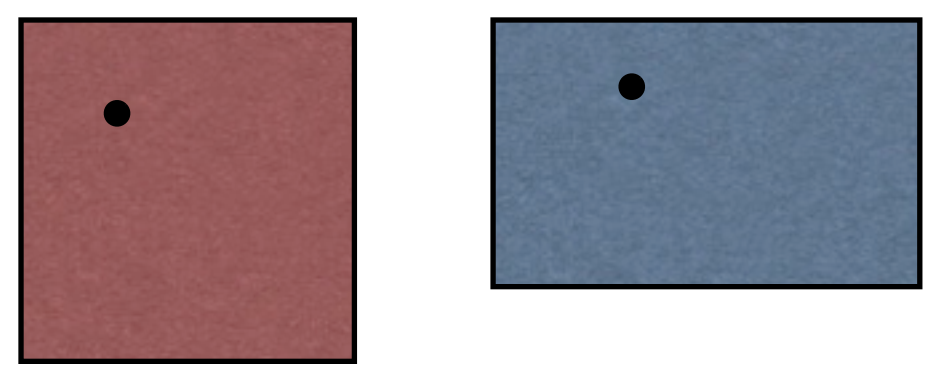
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners



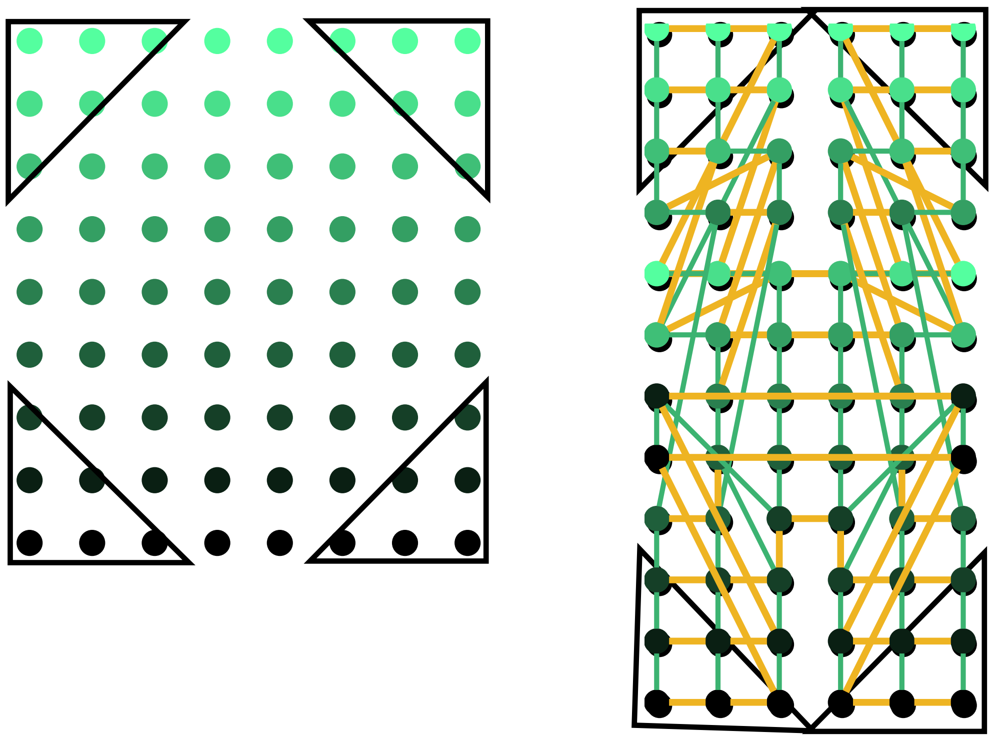
- Affine Mapping (AFFN)

$$(x, y) \rightarrow (\lfloor P_x * \frac{x}{O_x} \rfloor, \lfloor P_y * \frac{y}{O_y} \rfloor)$$



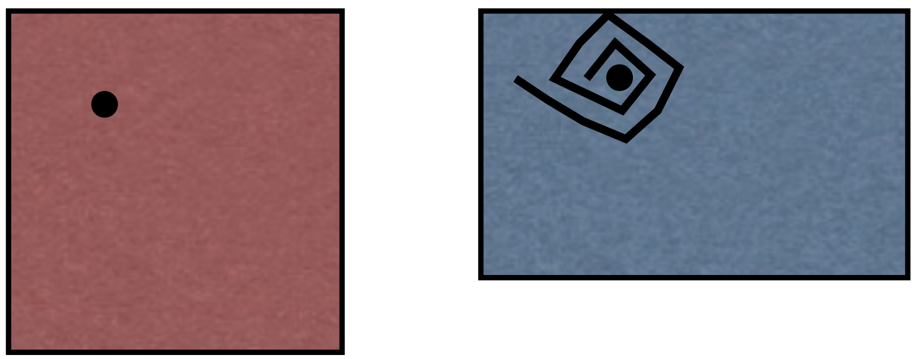
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners



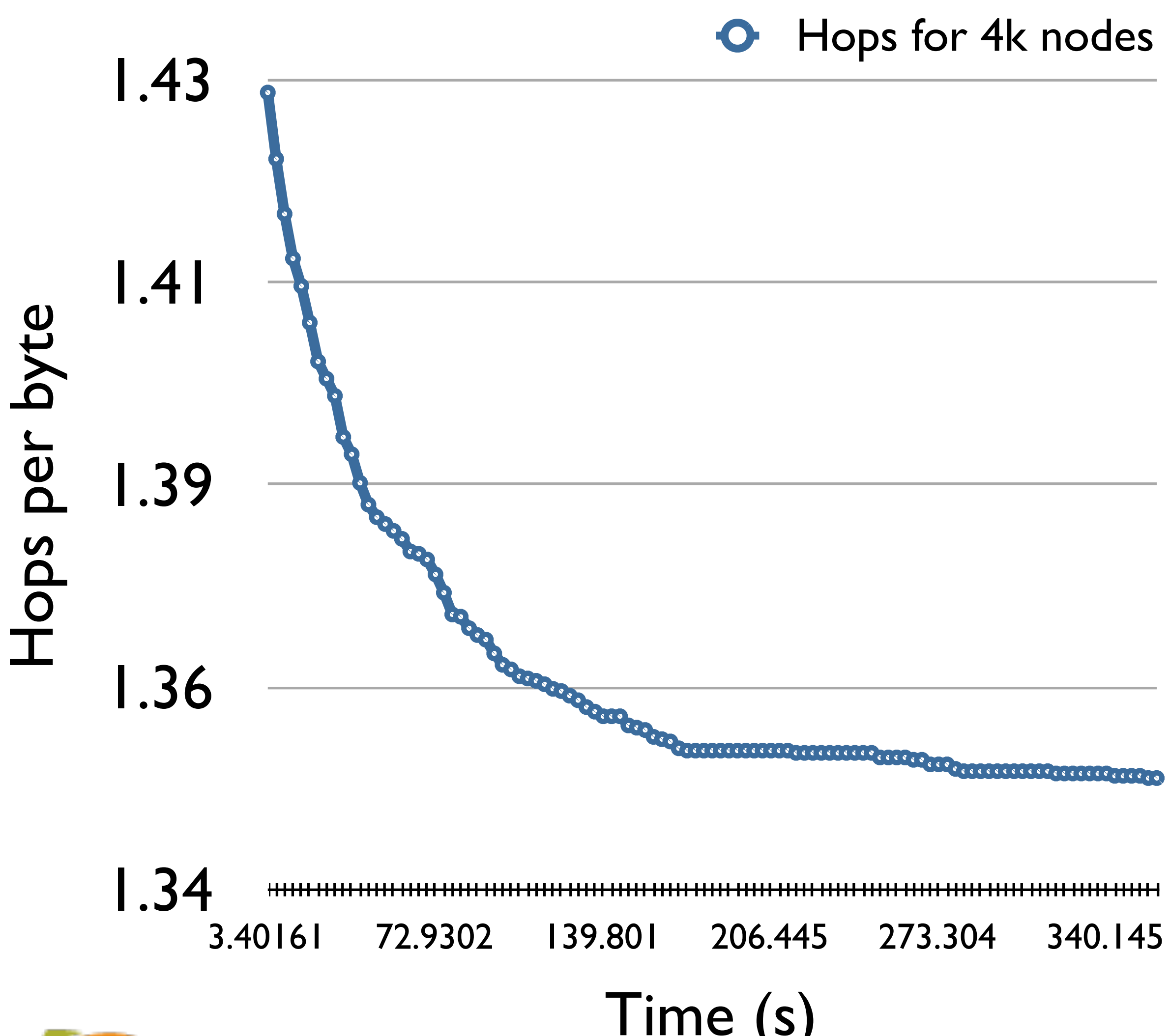
- Affine Mapping (AFFN)

$$(x, y) \rightarrow (\lfloor P_x * \frac{x}{O_x} \rfloor, \lfloor P_y * \frac{y}{O_y} \rfloor)$$



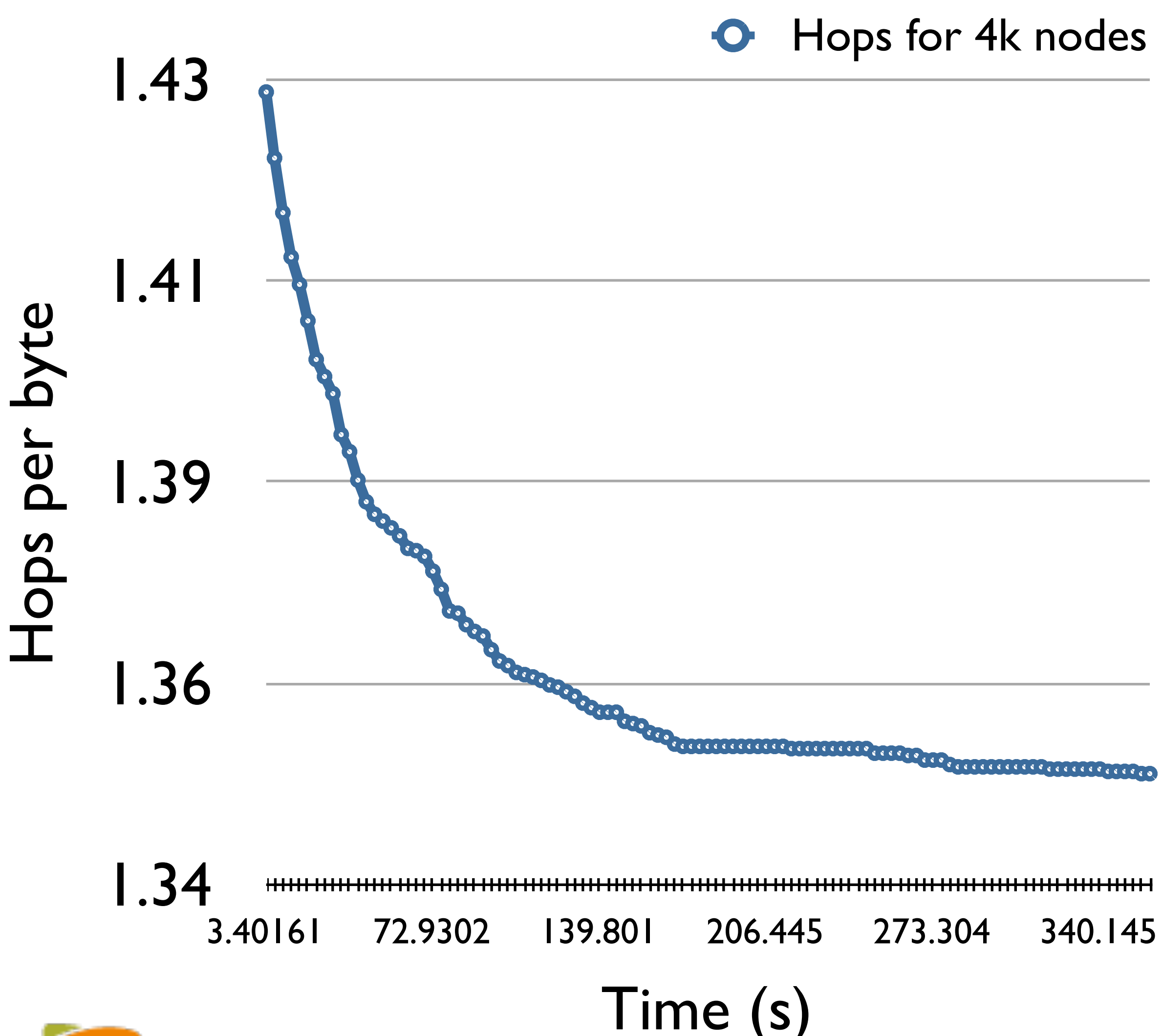
Running Time

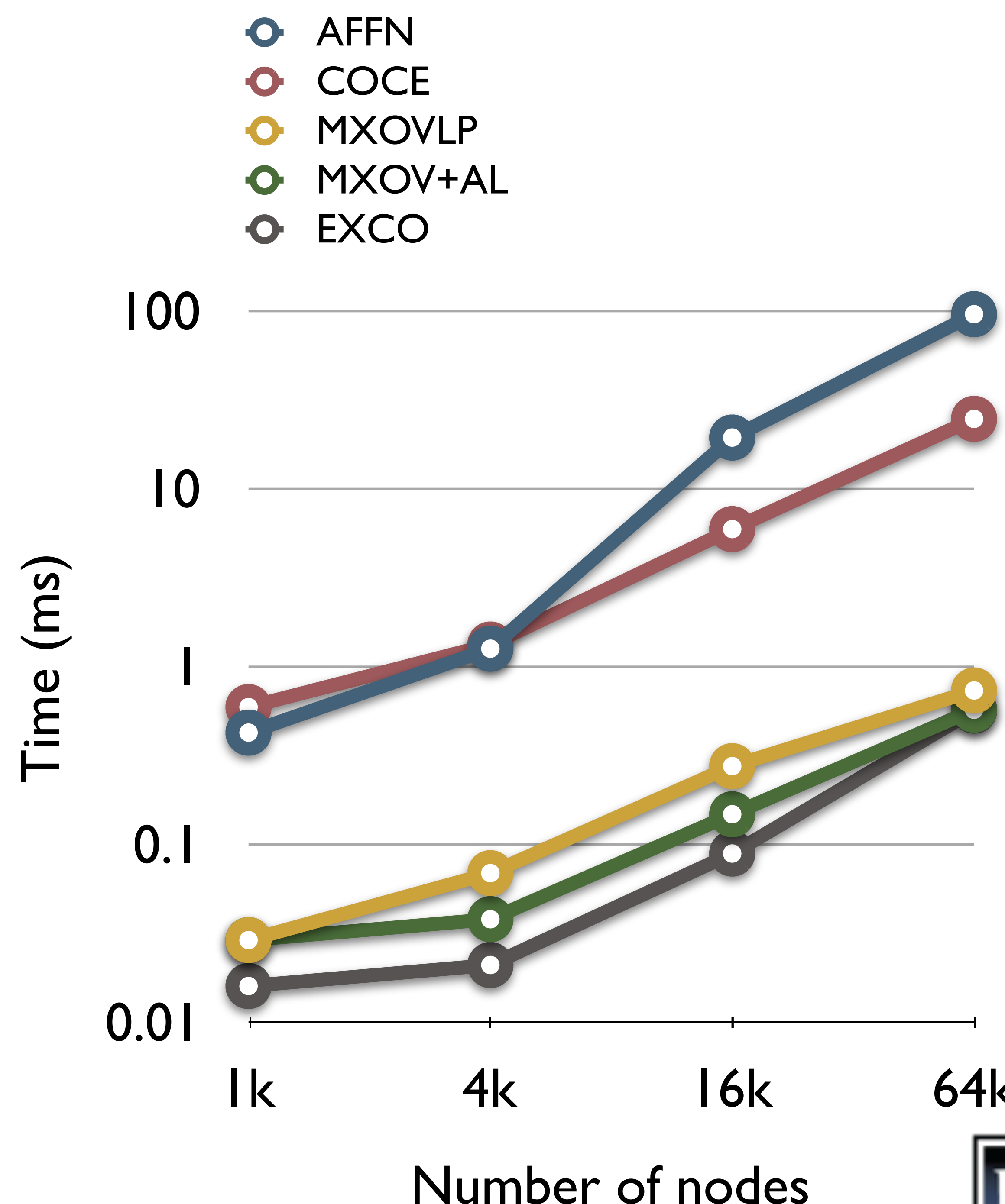
- Pairwise Exchanges (PAIRS)
  - Bokhari, Lee et al.



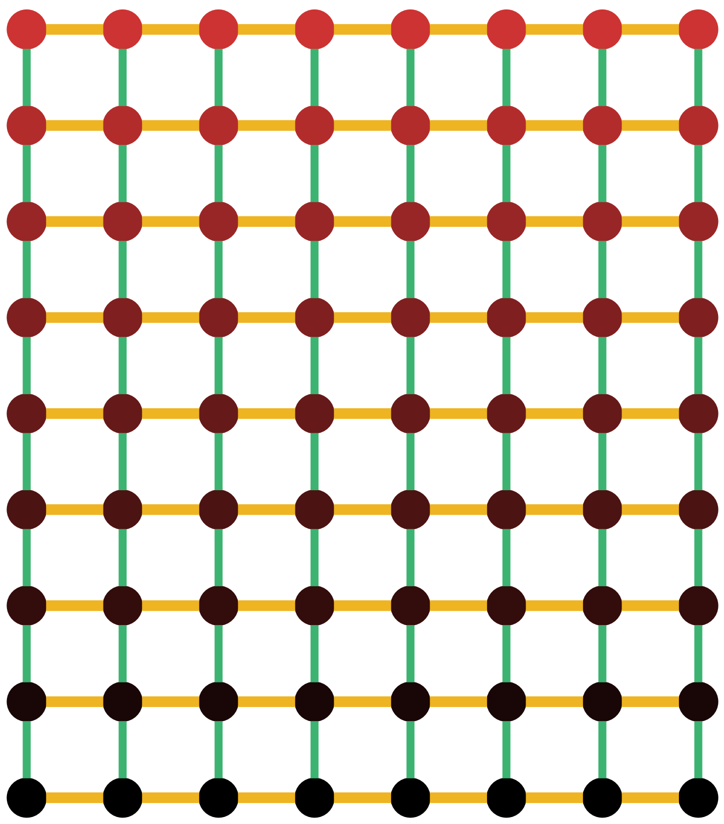
Running Time

- Pairwise Exchanges (PAIRS)
  - Bokhari, Lee et al.

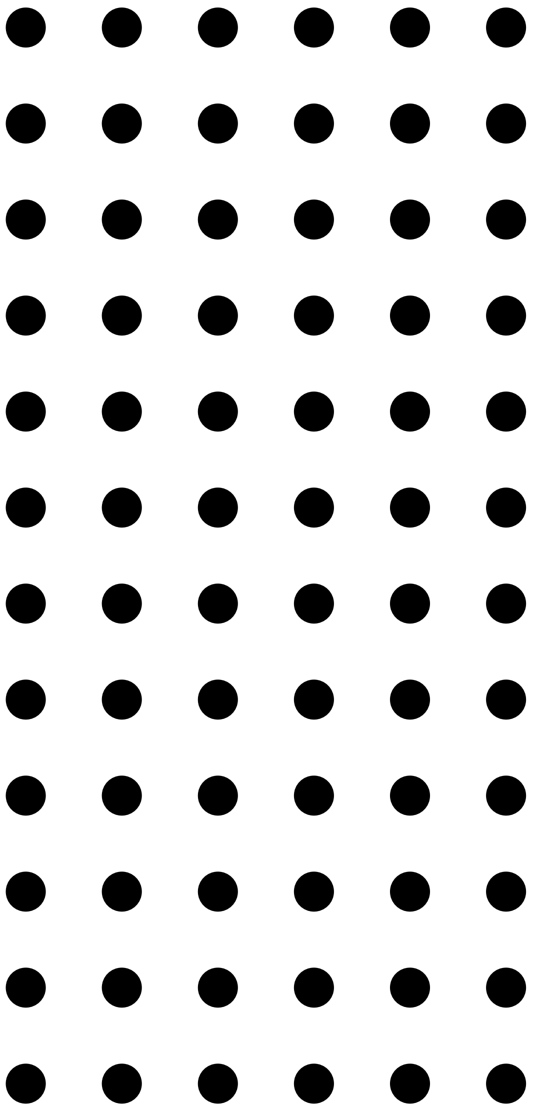




Example Mapping

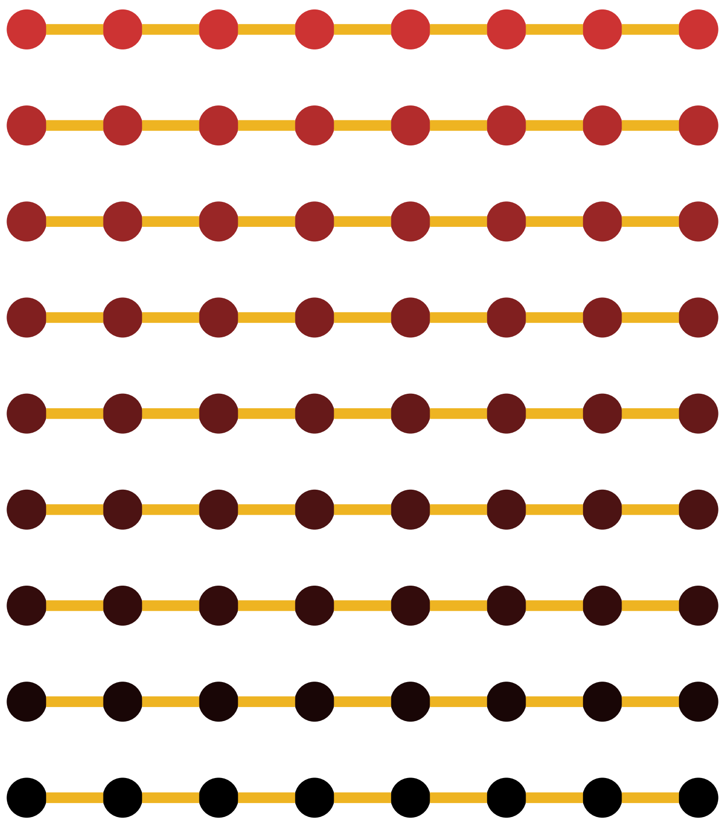


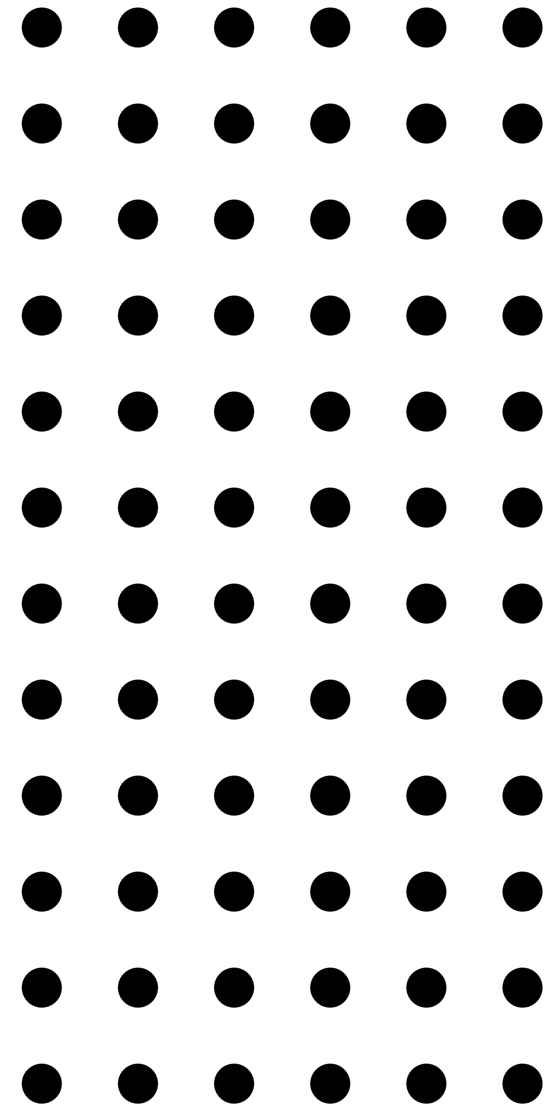
Object Graph: 9 x 8
Processor Graph: 12 x 6



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

Example Mapping

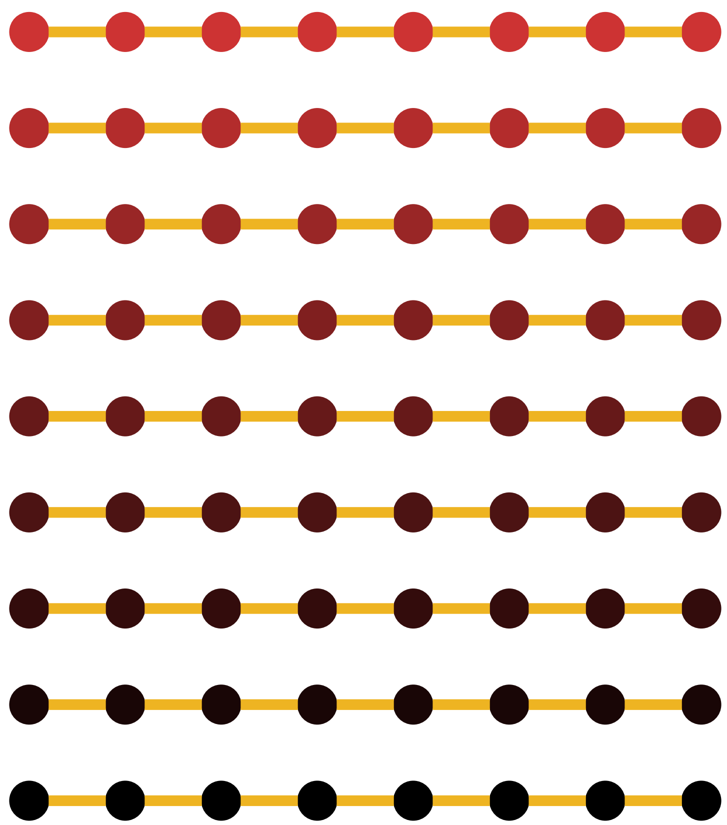


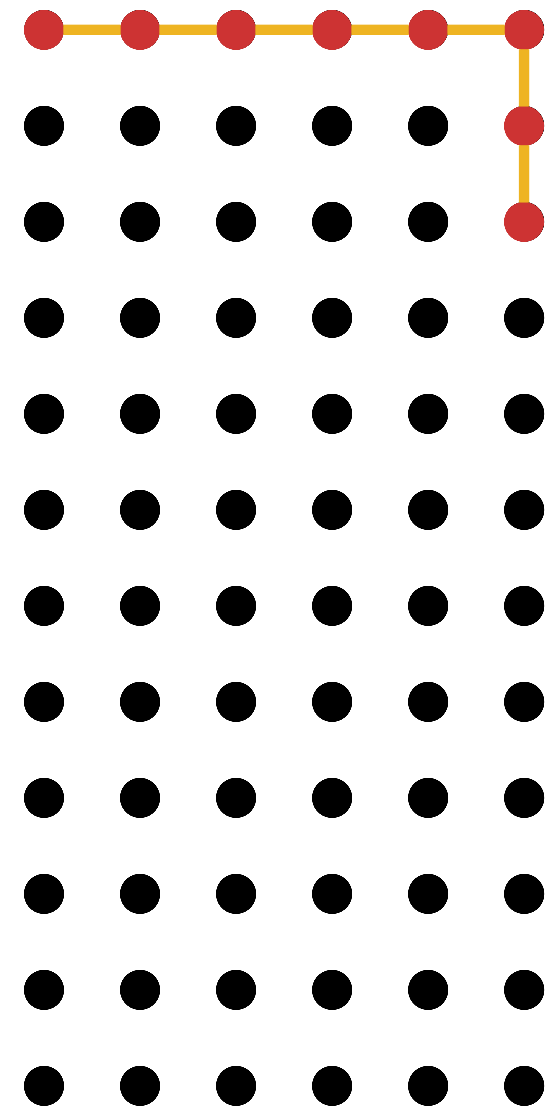


Object Graph: 9 x 8
Processor Graph: 12 x 6

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

Example Mapping

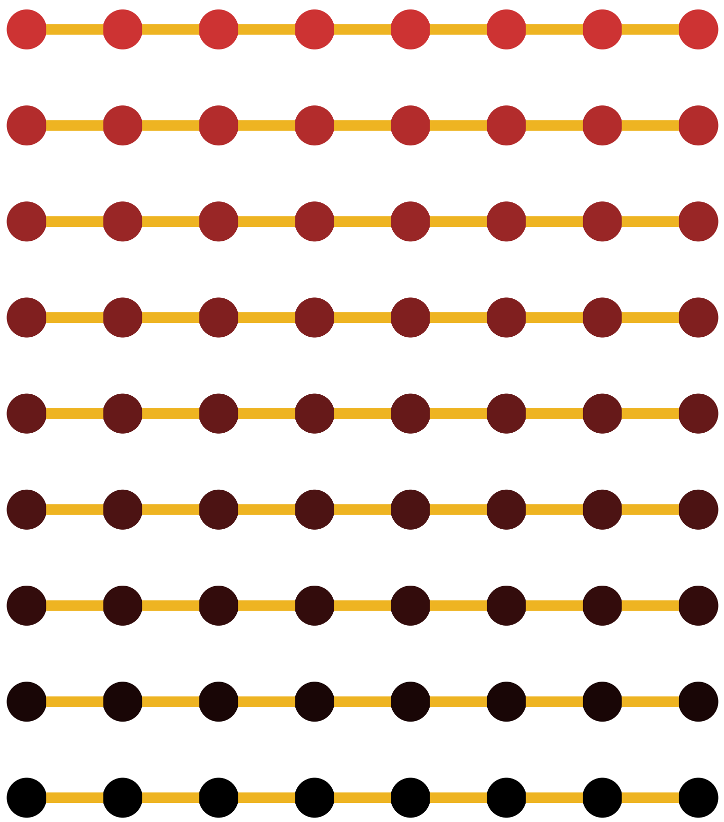


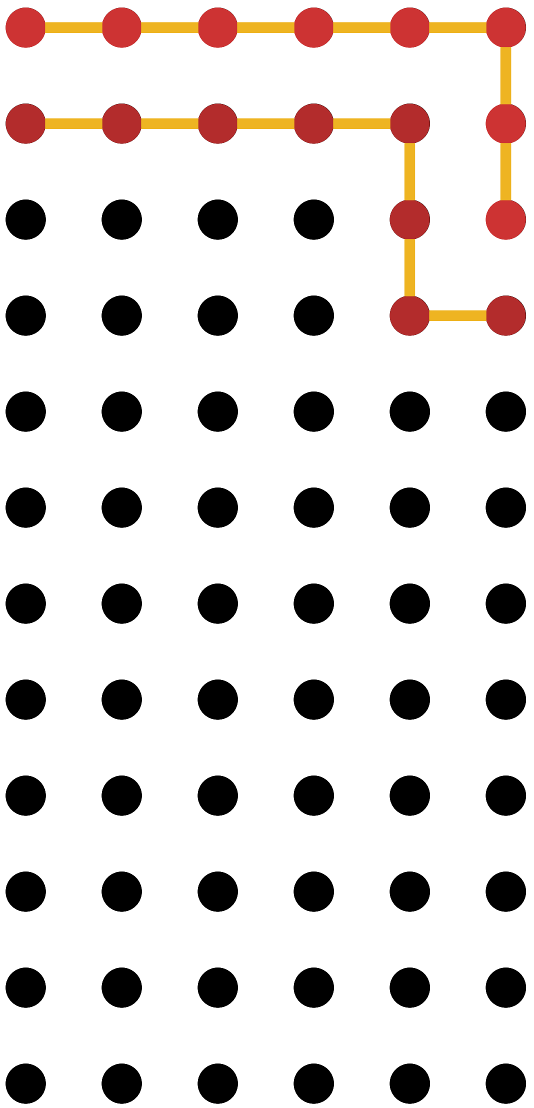


Object Graph: 9 x 8
Processor Graph: 12 x 6

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

Example Mapping

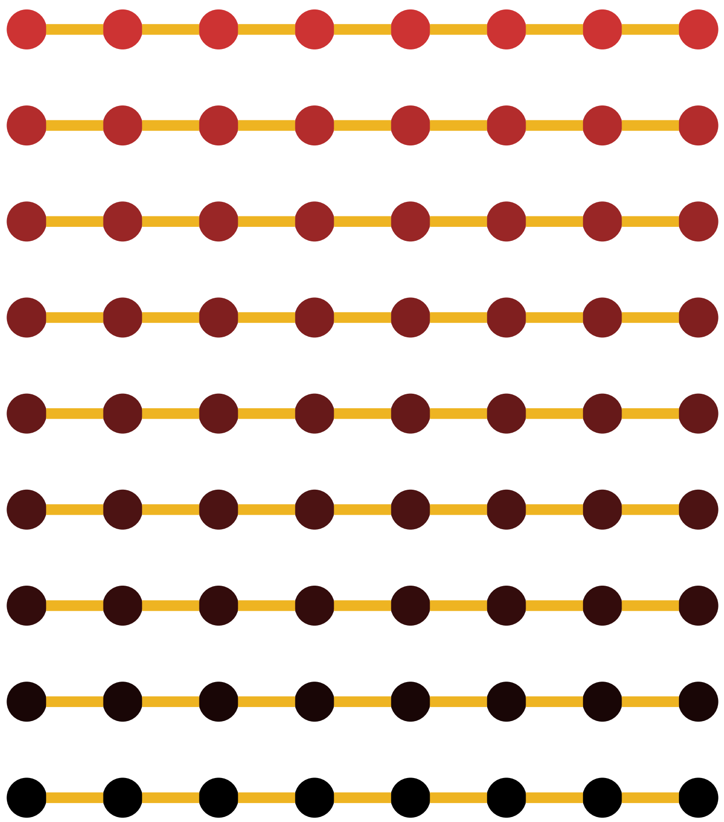


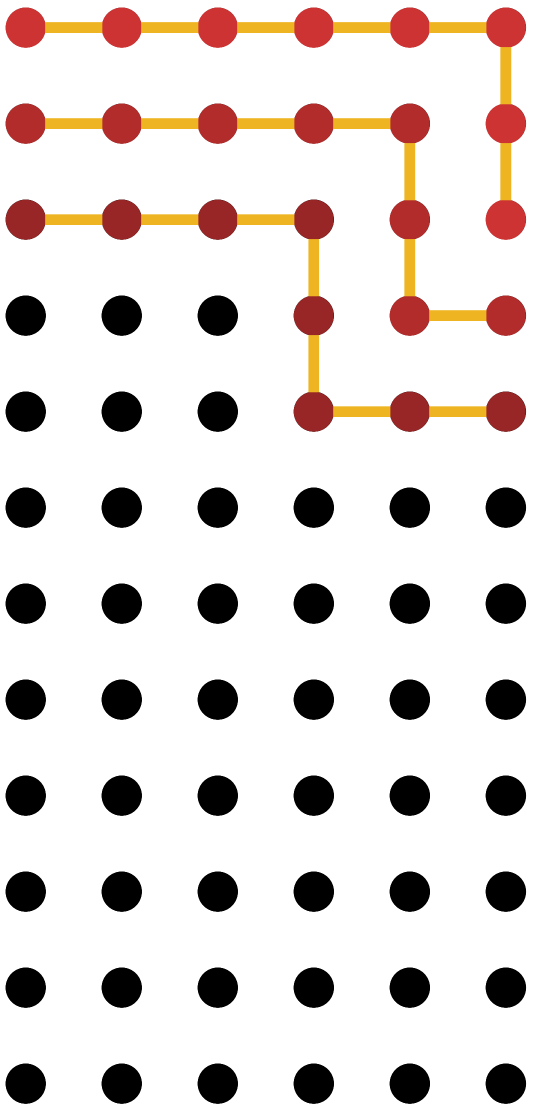


Object Graph: 9 x 8
Processor Graph: 12 x 6

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

Example Mapping

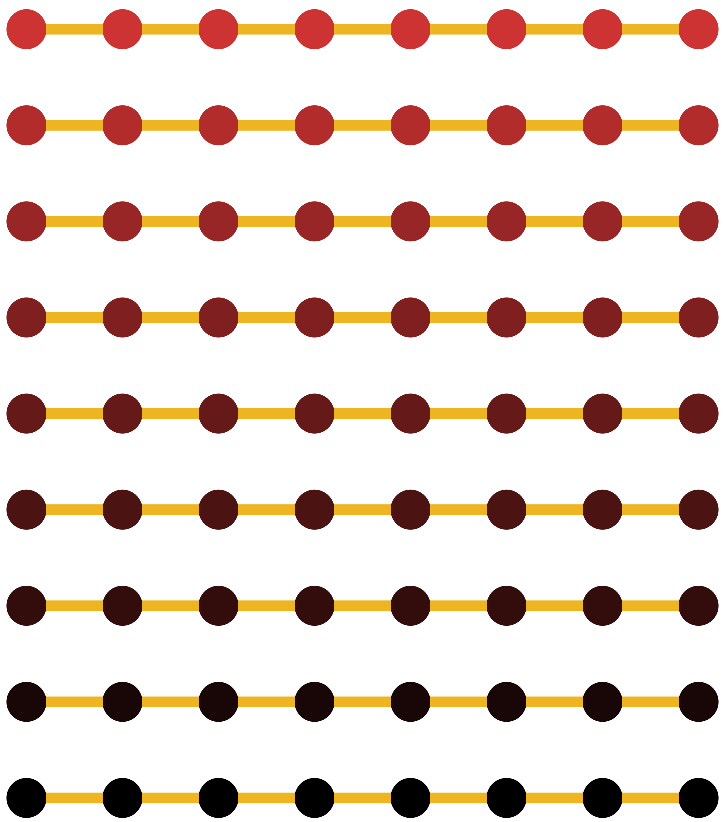




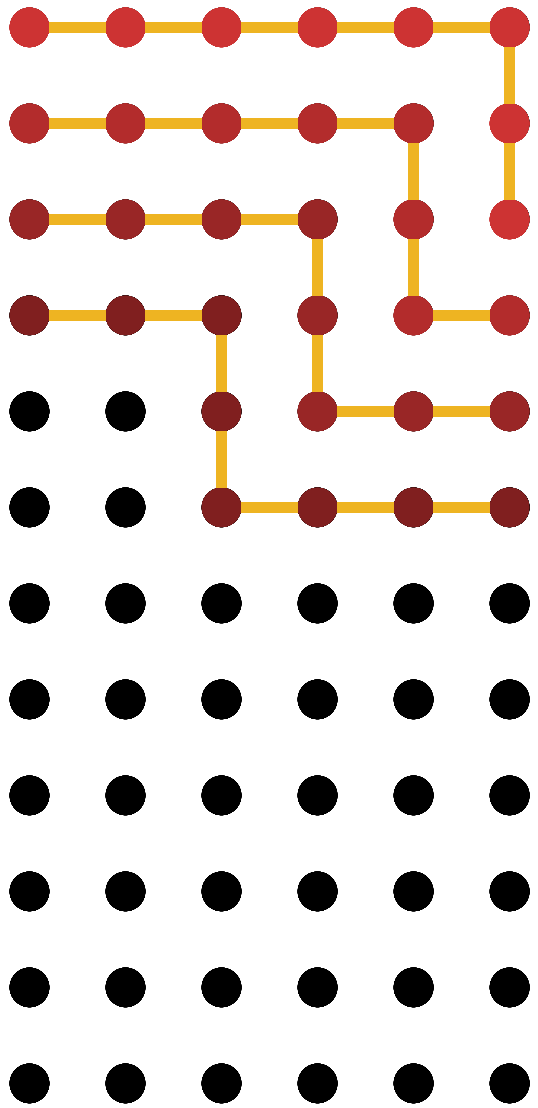
Object Graph: 9 x 8
Processor Graph: 12 x 6

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

Example Mapping

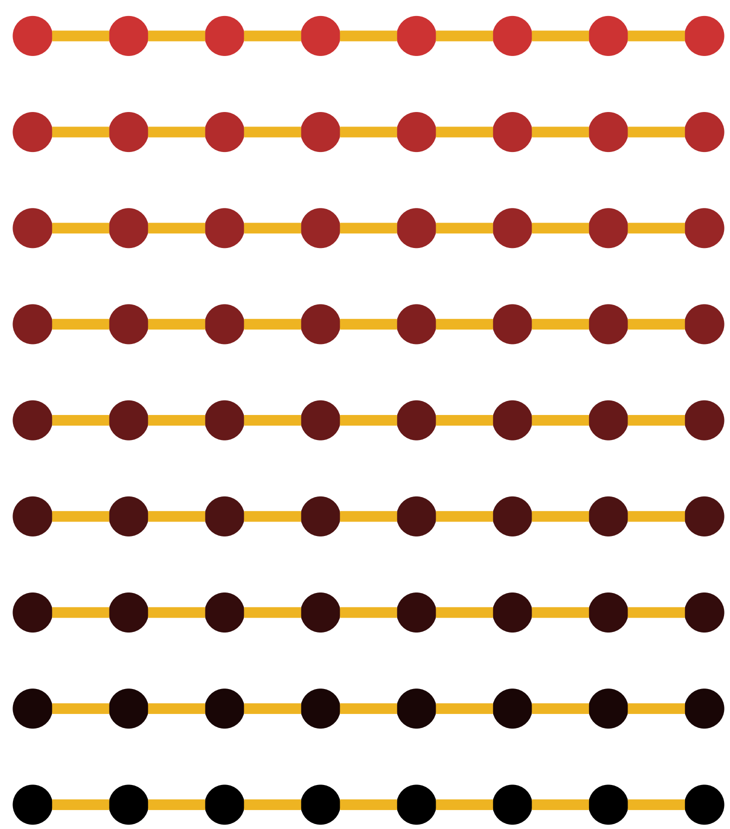


Object Graph: 9 x 8
Processor Graph: 12 x 6

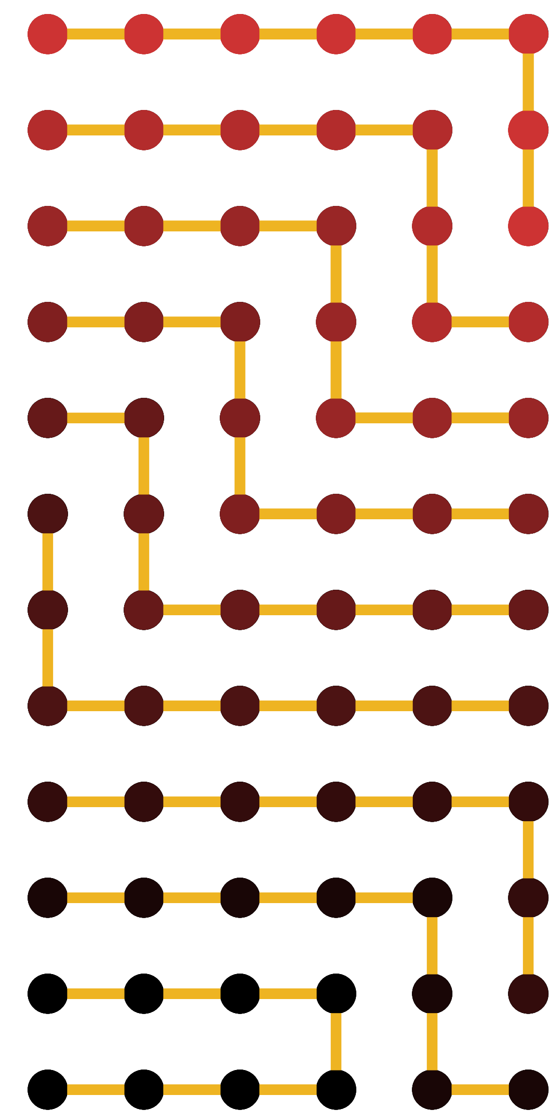


Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

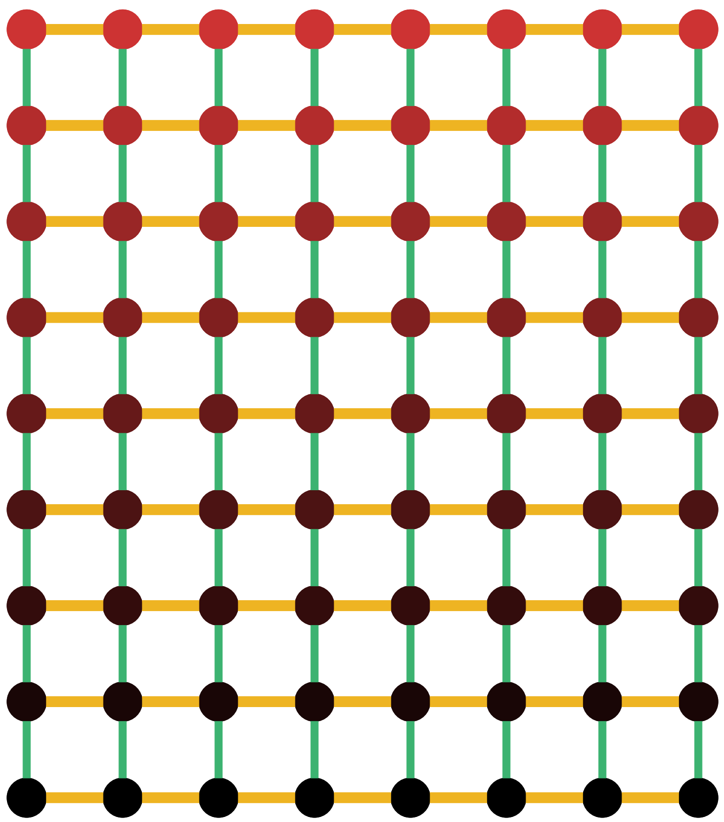


Object Graph: 9 x 8
Processor Graph: 12 x 6

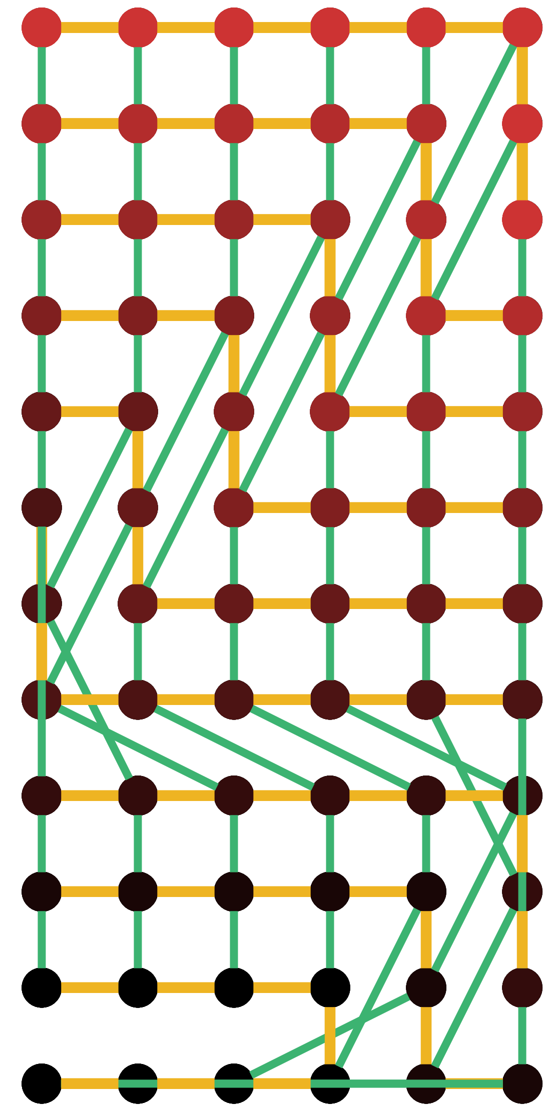


Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Example Mapping

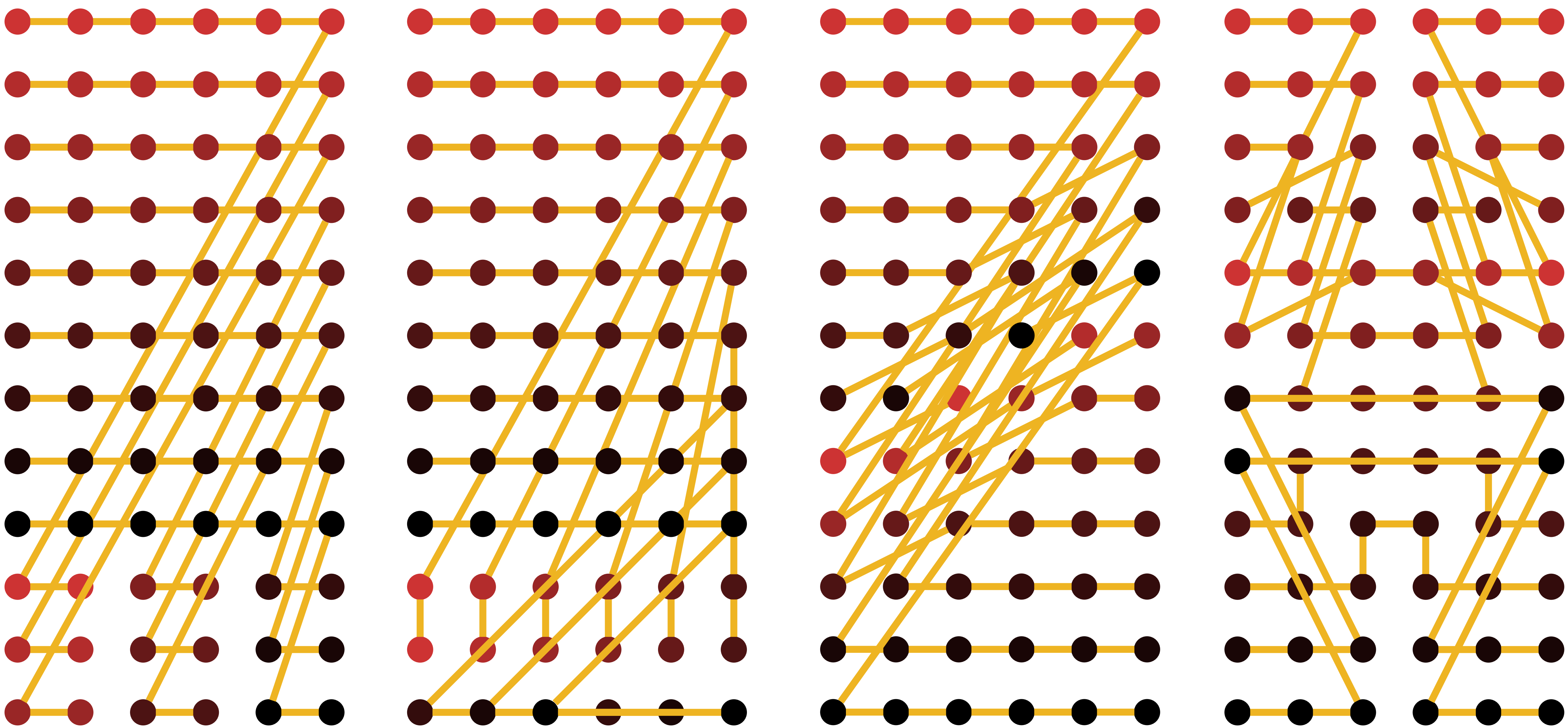


Object Graph: 9 x 8
Processor Graph: 12 x 6



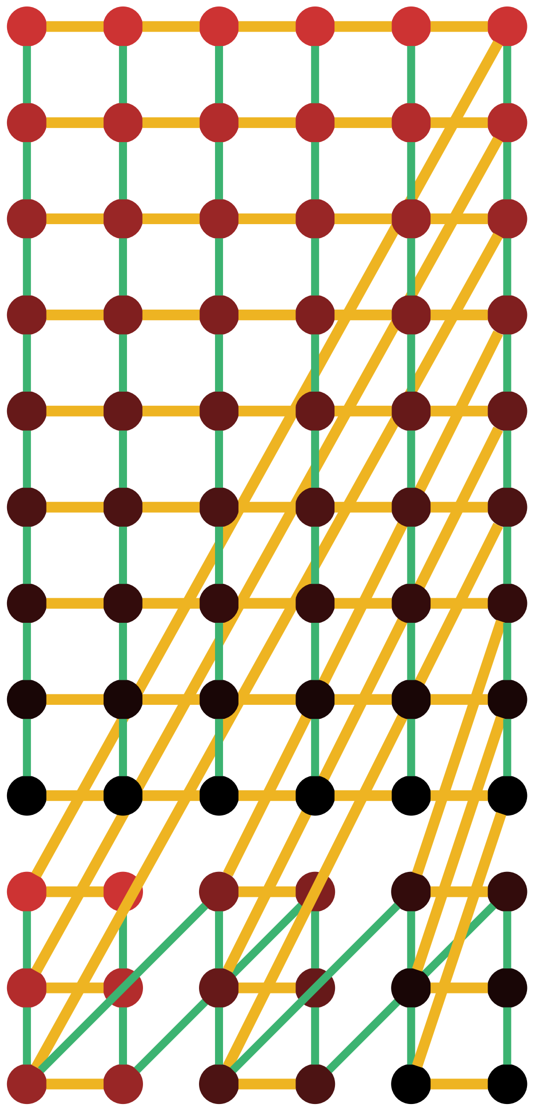
Aleliunas, R. and Rosenberg, A. L. On Embedding Rectangular Grids in Square Grids. IEEE Trans. Comput., 31(9):907–913, 1982

Mapping of 9x8 graph to 12x6 mesh

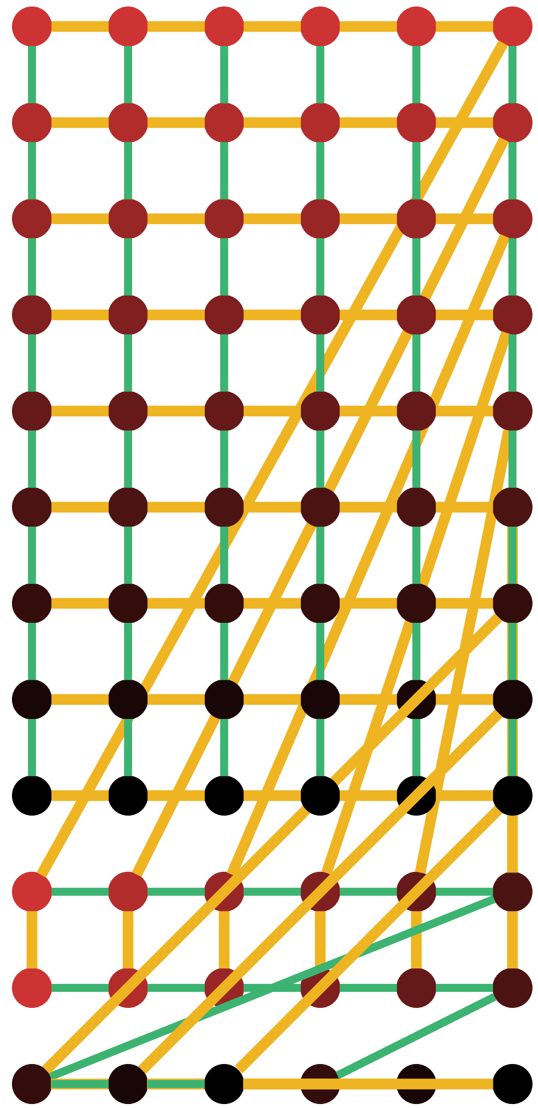


MXOVLP: 1.66 MXOV+AL: 1.65 EXCO: 2.31 COCE: 1.91

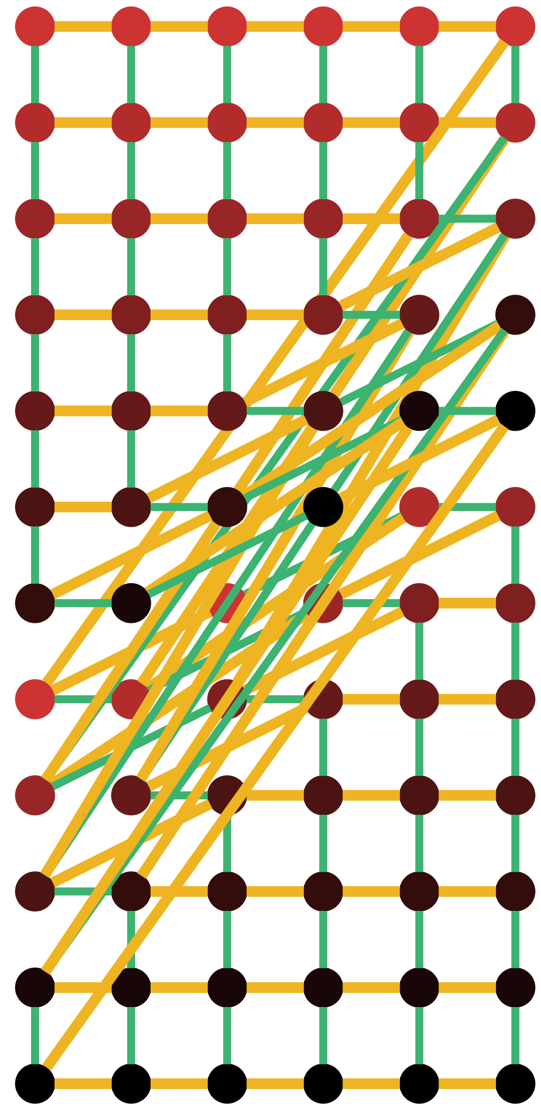
Mapping of 9x8 graph to 12x6 mesh



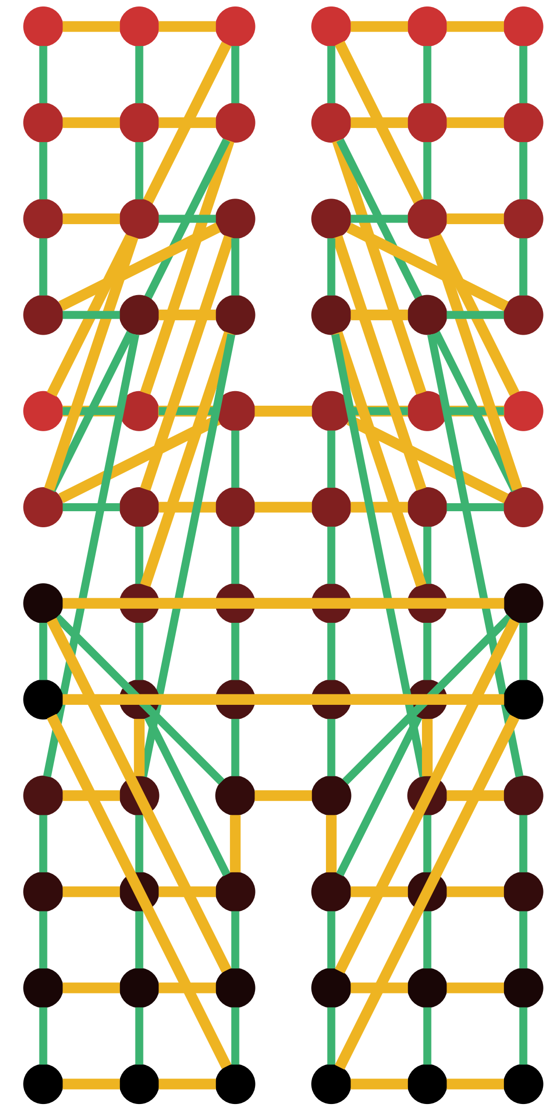
MXOVLP: 1.66



MXOV+AL: 1.65

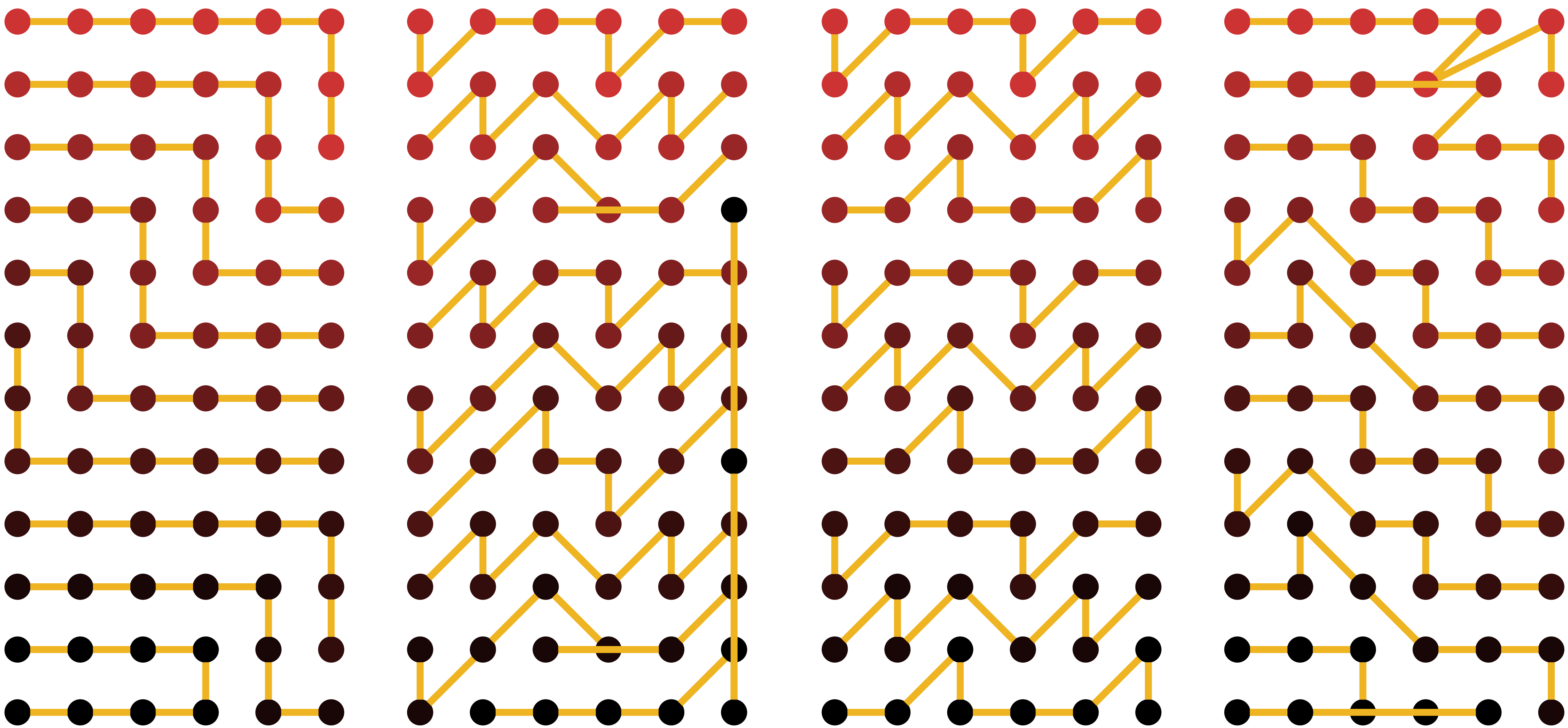


EXCO: 2.31



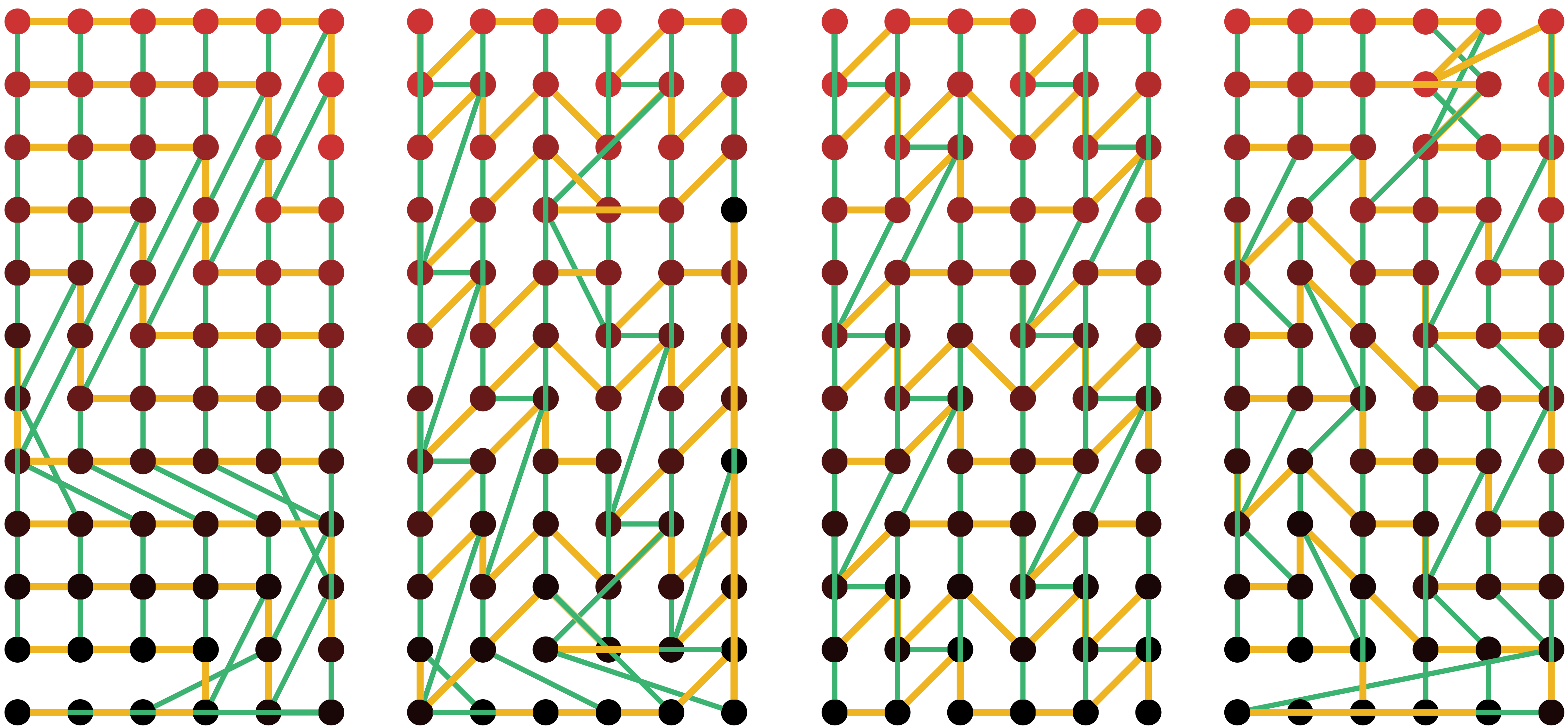
COCE: 1.91

Mapping of 9x8 graph to 12x6 mesh



STEP: 1.39 AFFN1: 1.77 AFFN2: 1.53 AFFN3: 1.91

Mapping of 9x8 graph to 12x6 mesh



STEP: 1.39 AFFN1: 1.77 AFFN2: 1.53 AFFN3: 1.91

Evaluation Metric

- Hop-bytes:

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

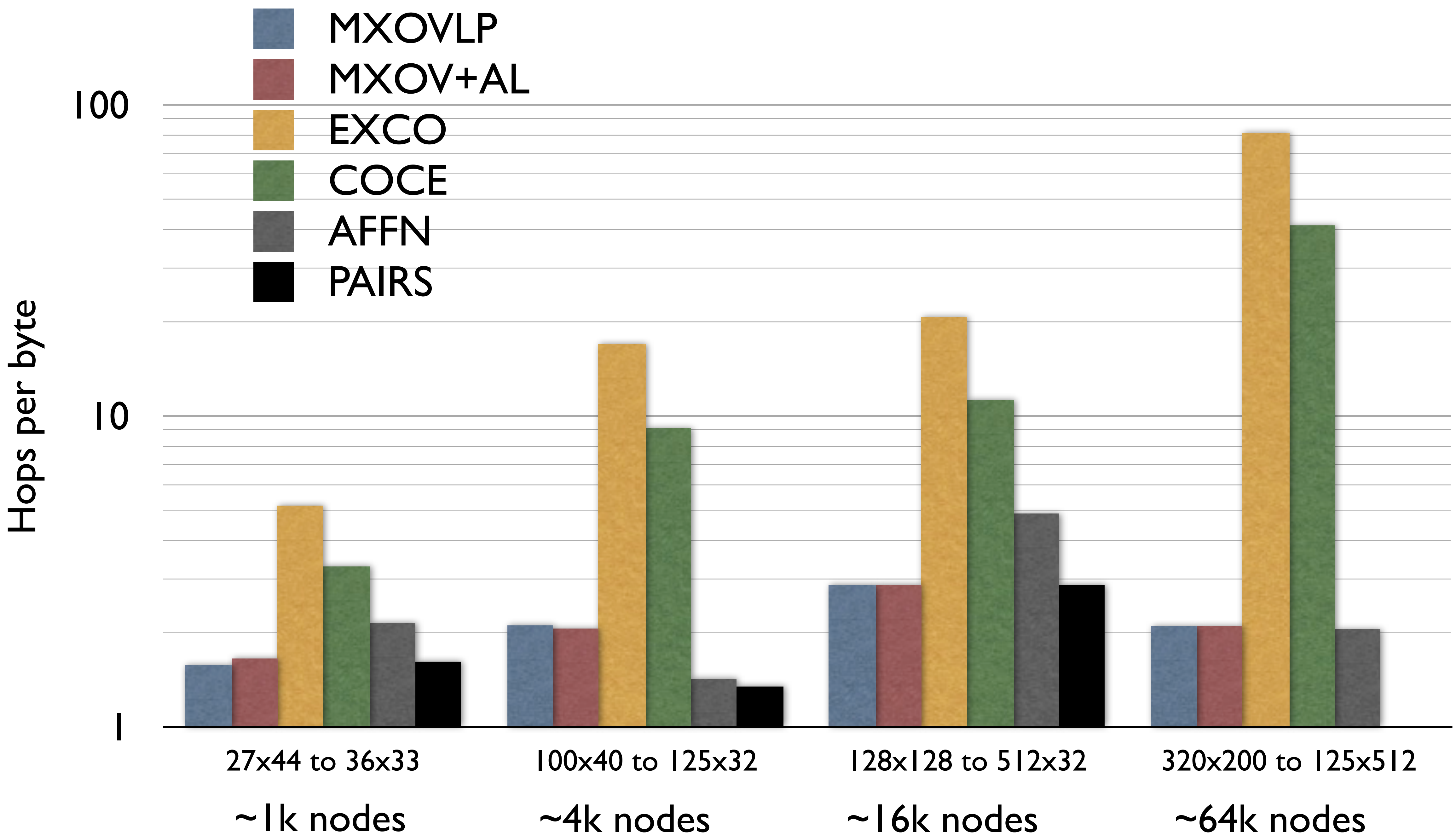
d_i = distance
b_i = bytes
n = no. of messages

- Indicates amount of traffic and hence contention on the network

- Previously used metric: maximum dilation

$$d(e) = max\{d_i | e_i \in E\}$$

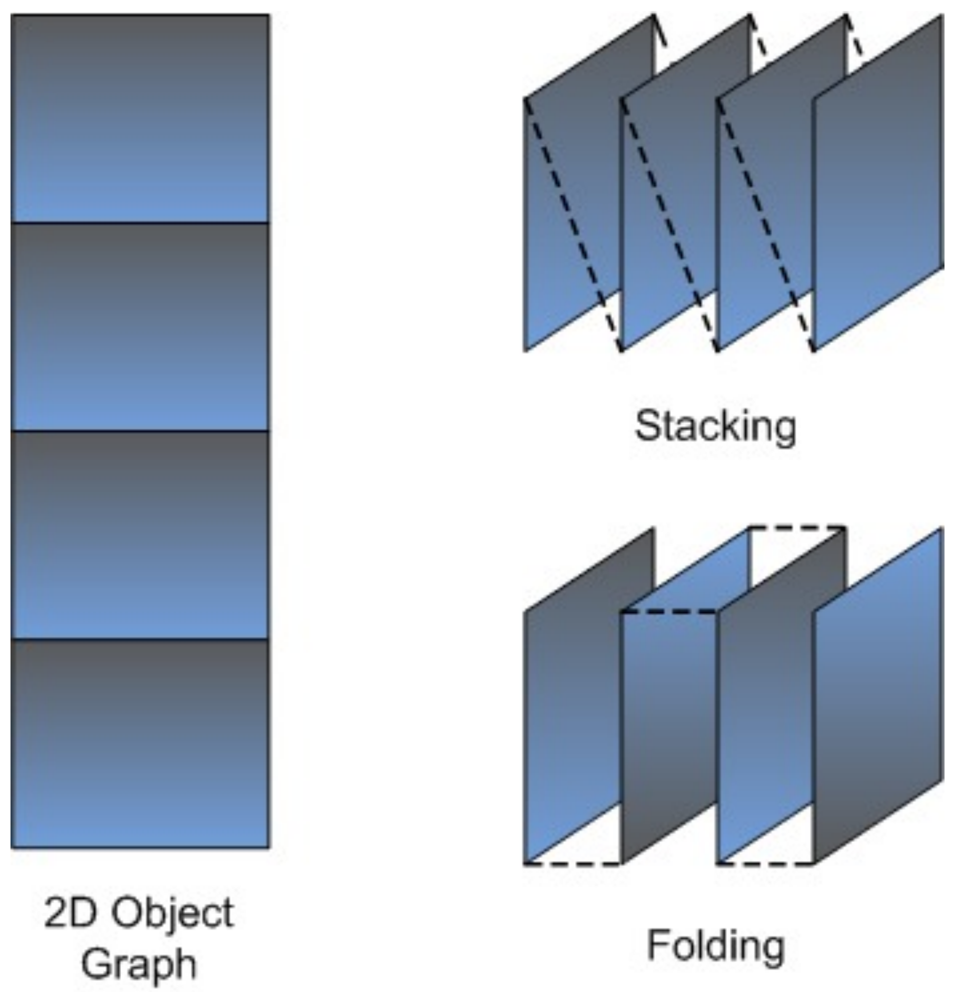
Evaluation



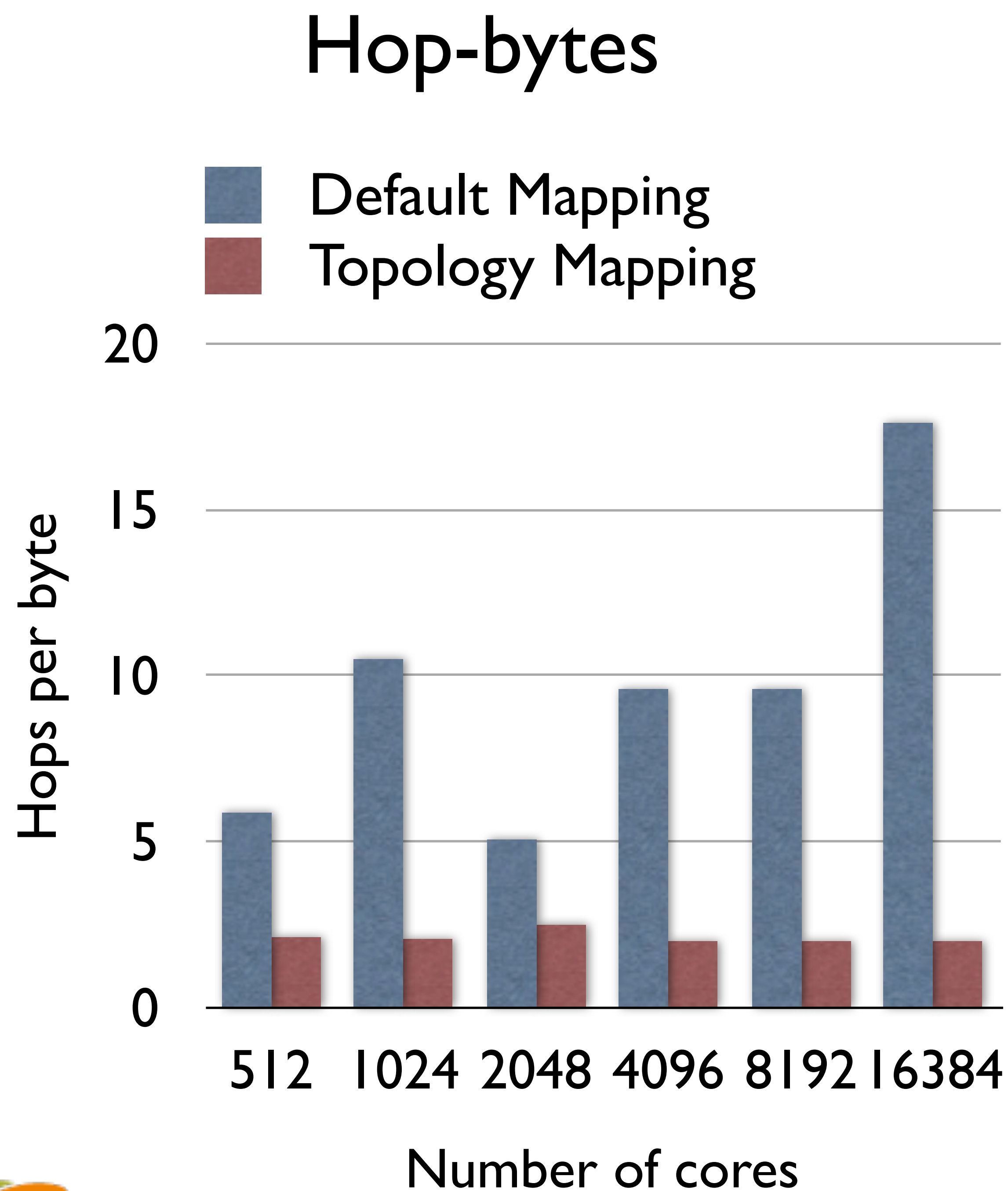
MXOVLP
MXOV+AL
EXCO
COCE
AFFN
PAIRS
100
10
1
Hops per byte
27x44 to 36x33
100x40 to 125x32
128x128 to 512x32
320x200 to 125x512
~1k nodes
~4k nodes
~16k nodes
~64k nodes

Mapping 2D Graphs to 3D

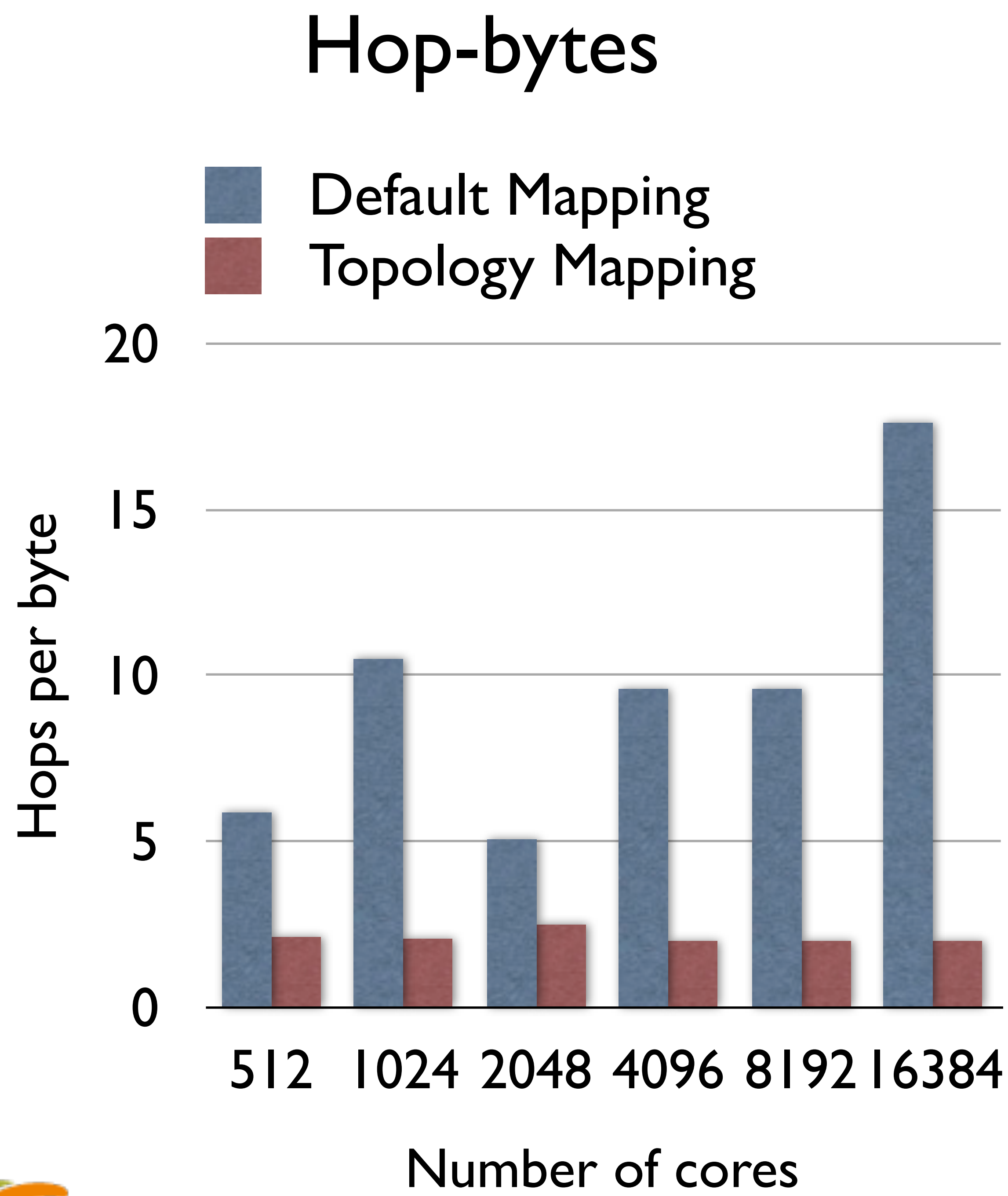
- Map a two-dimensional object graph to a three-dimensional processor graph
- Divide object graph into subgraphs once each for the number of planes
  - Stacking
  - Folding
- Best 2D to 2D heuristic chosen based on hop-bytes

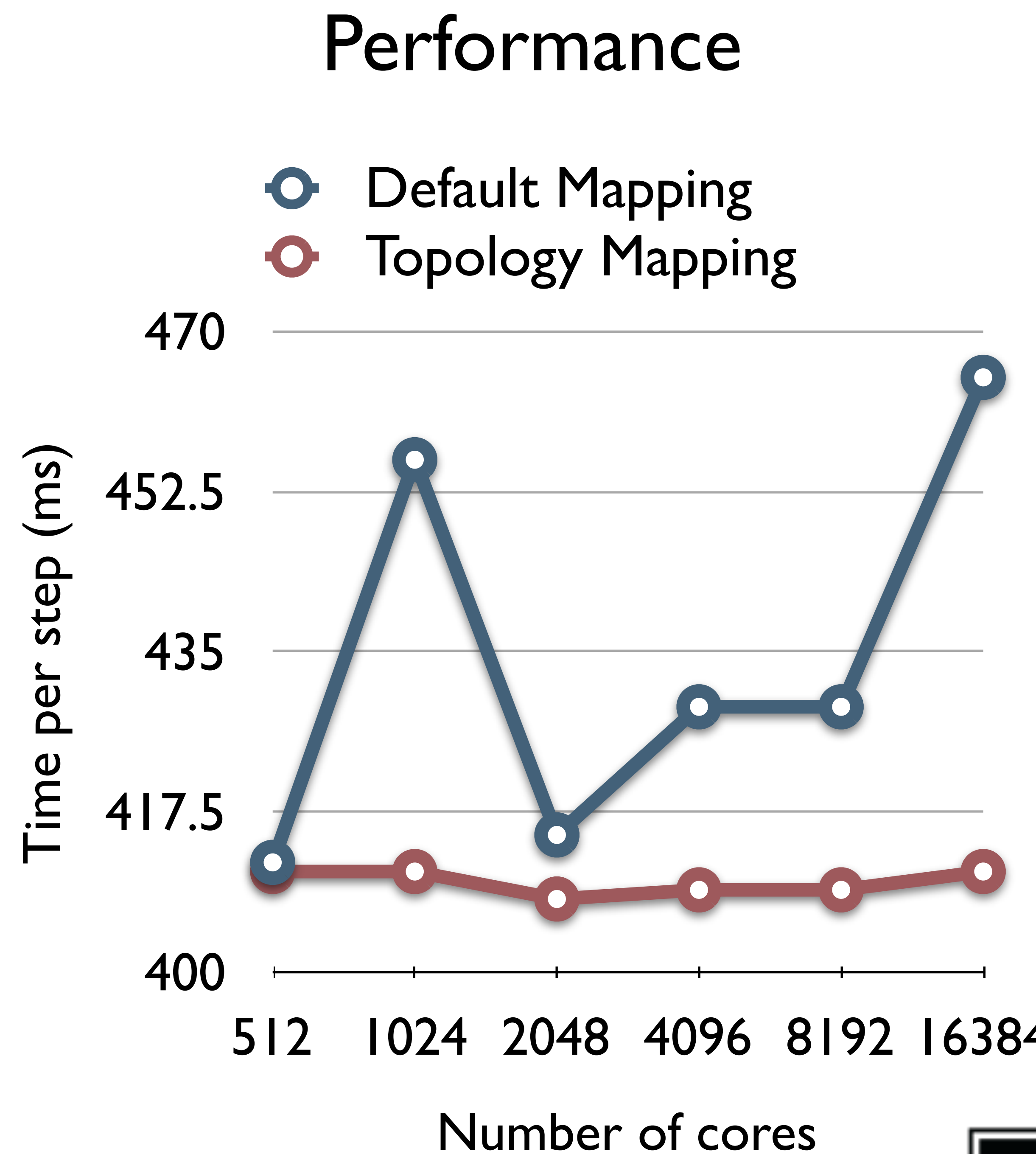


Results: 2D Stencil on Blue Gene/P



Results: 2D Stencil on Blue Gene/P

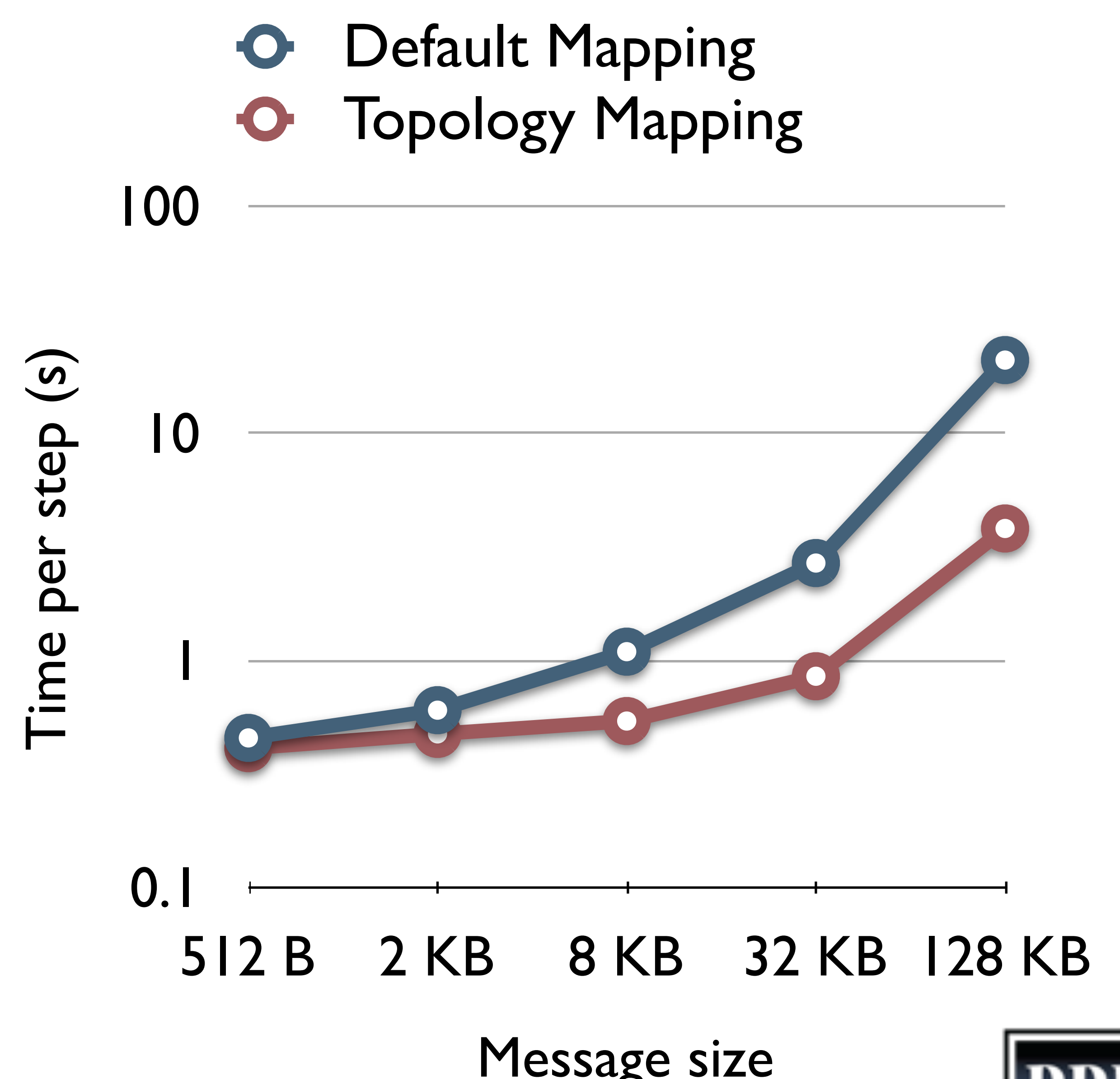




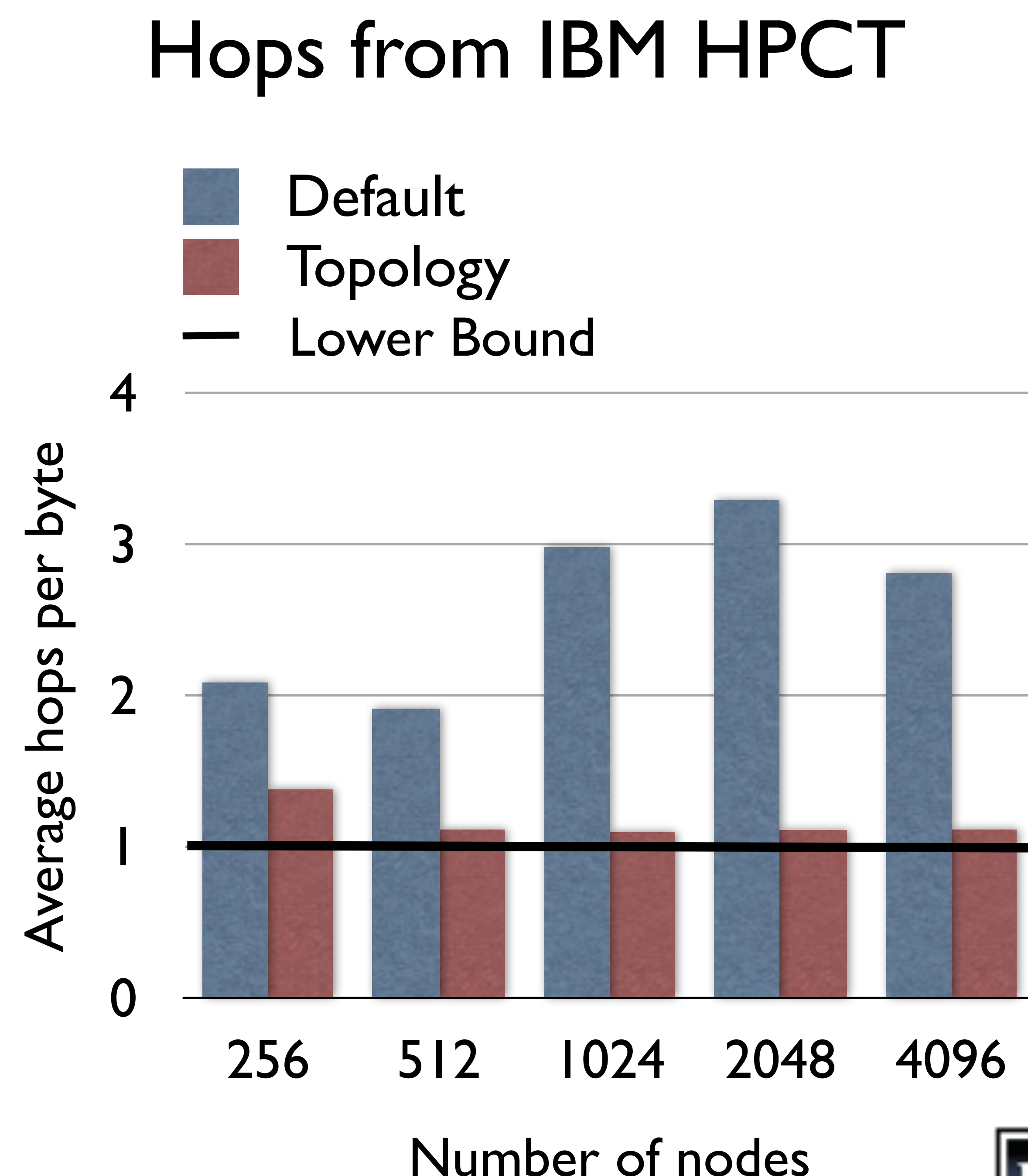
Increasing communication

- With faster processors and constant link bandwidths
  - computation is becoming cheap
  - communication is a bottleneck
- Trend for bytes per flop
  - XT3: 8.77
  - XT4: 1.357
  - XT5: 0.23

2D Stencil on BG/P (4,096 cores)

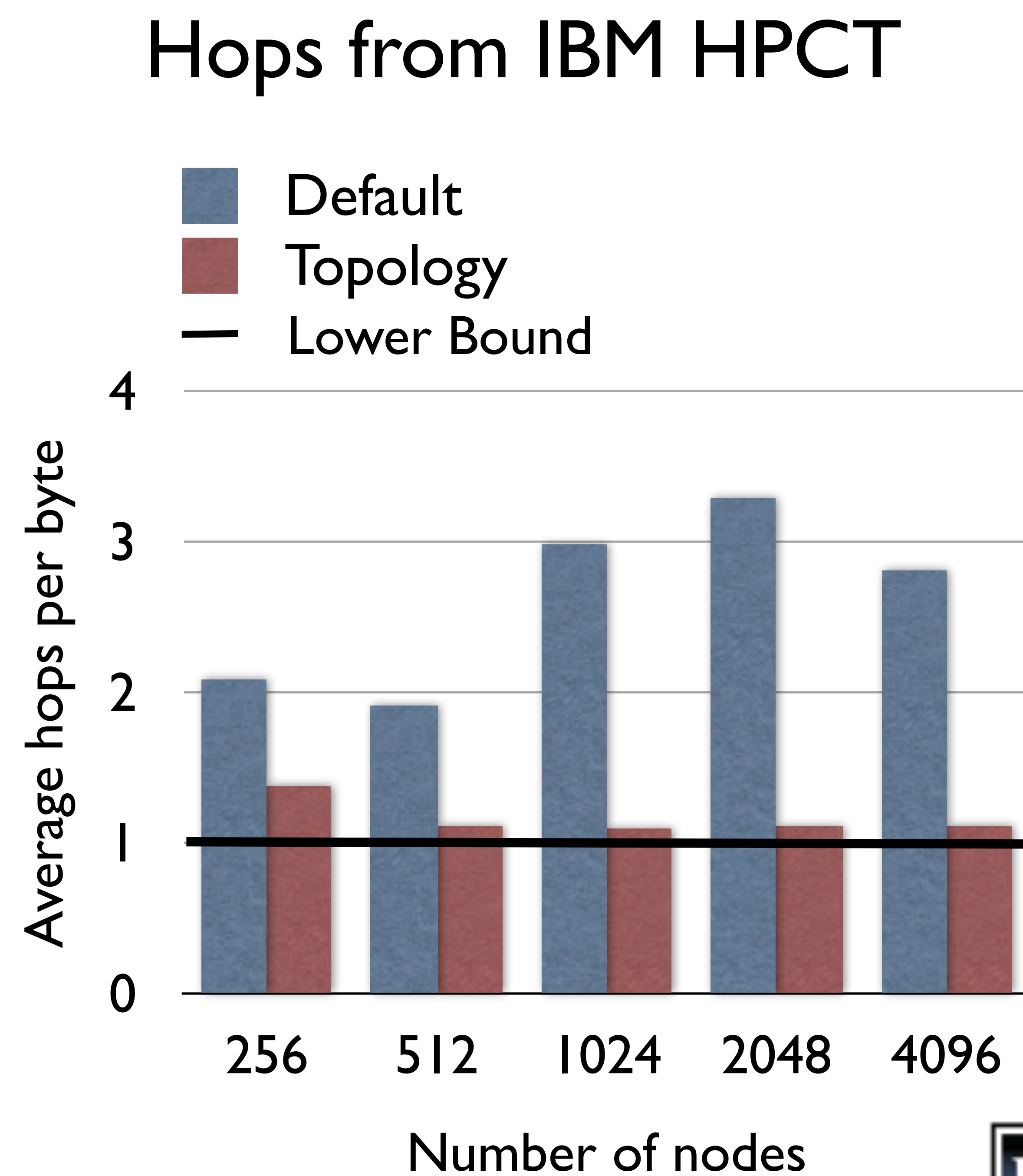


Results: WRF on Blue Gene/P



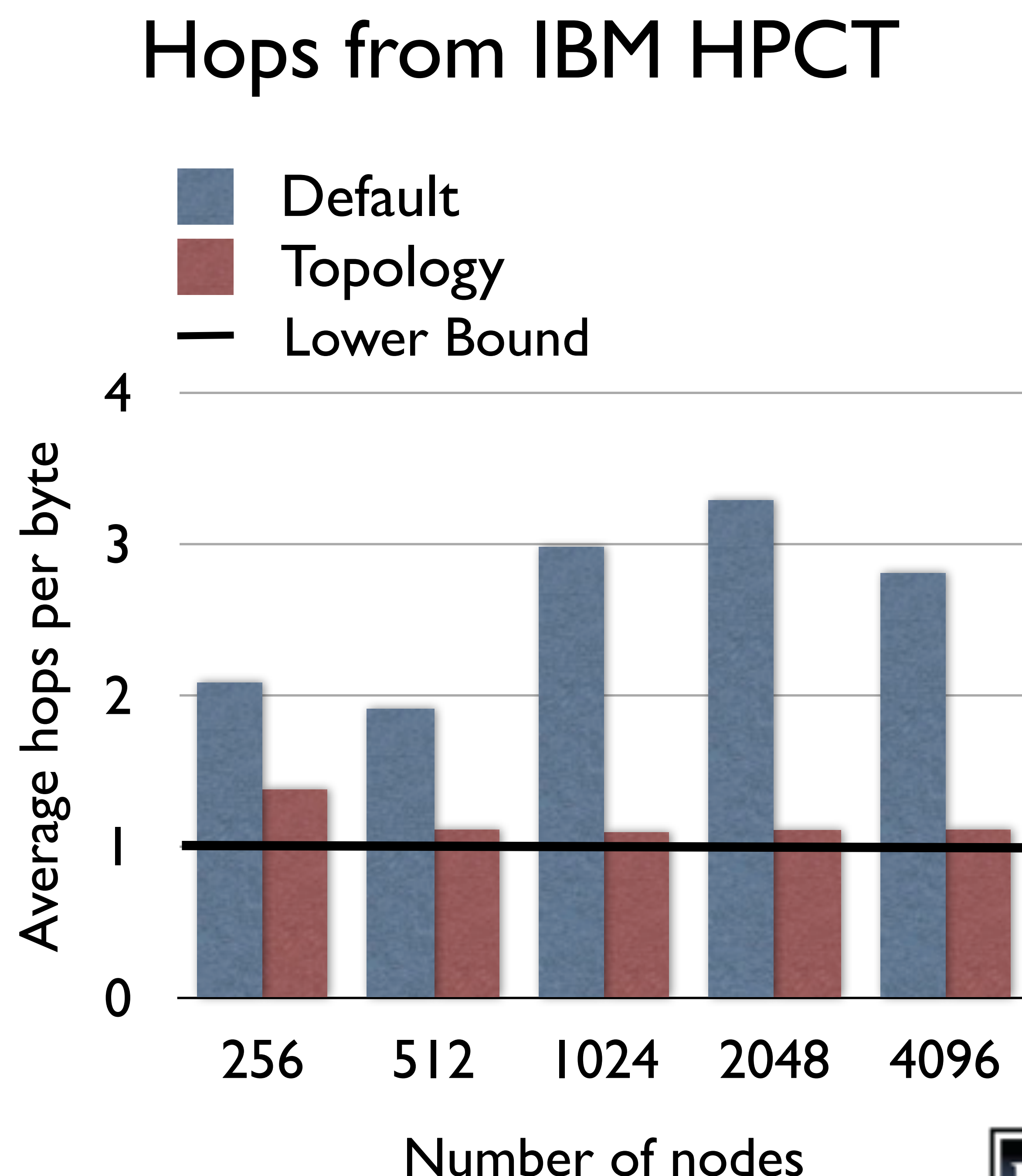
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores



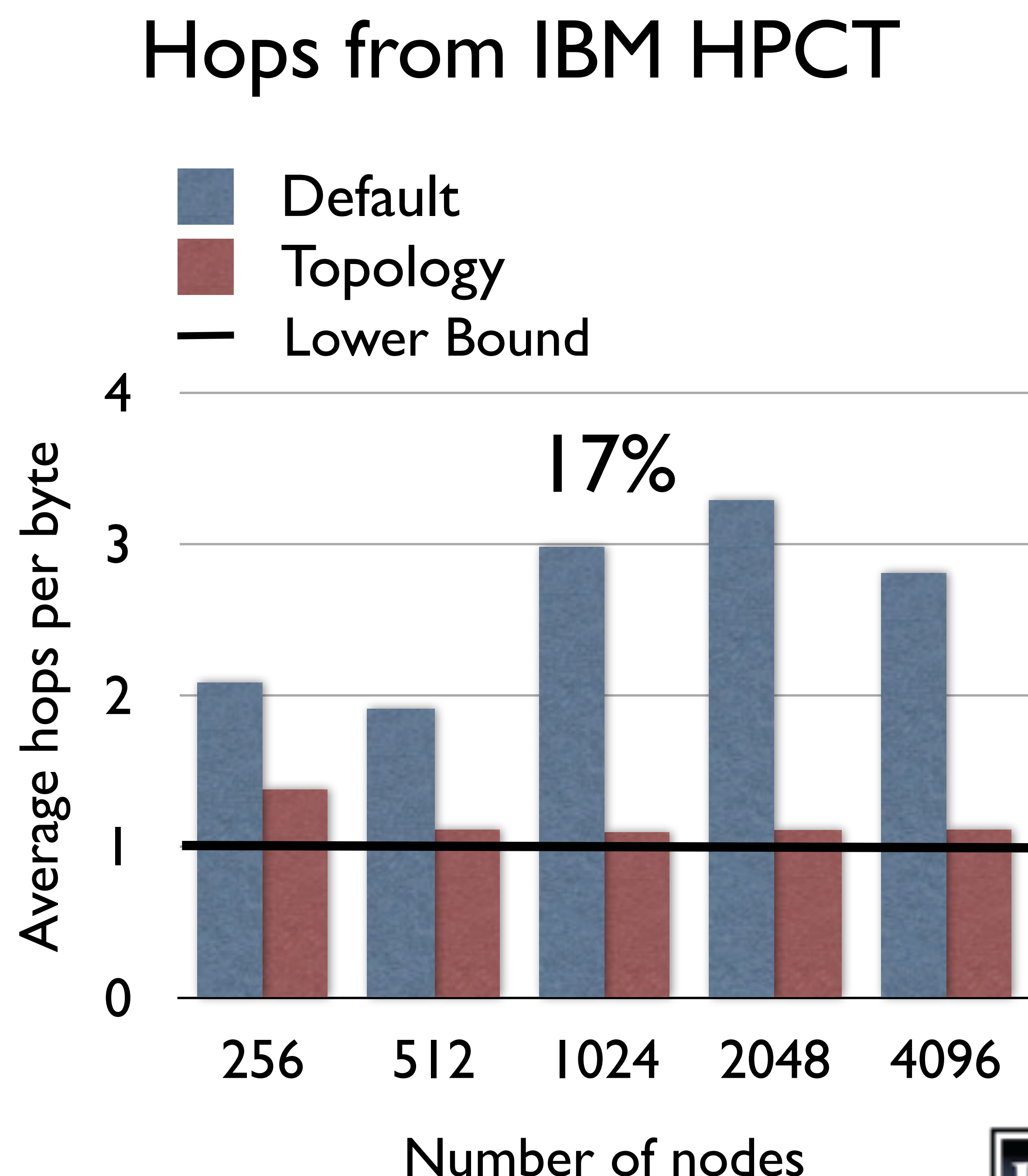
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores
- On 1024 nodes:
  - Hops reduce by: 63%
  - Time for communication reduces by 11%
  - Performance improves by 17%



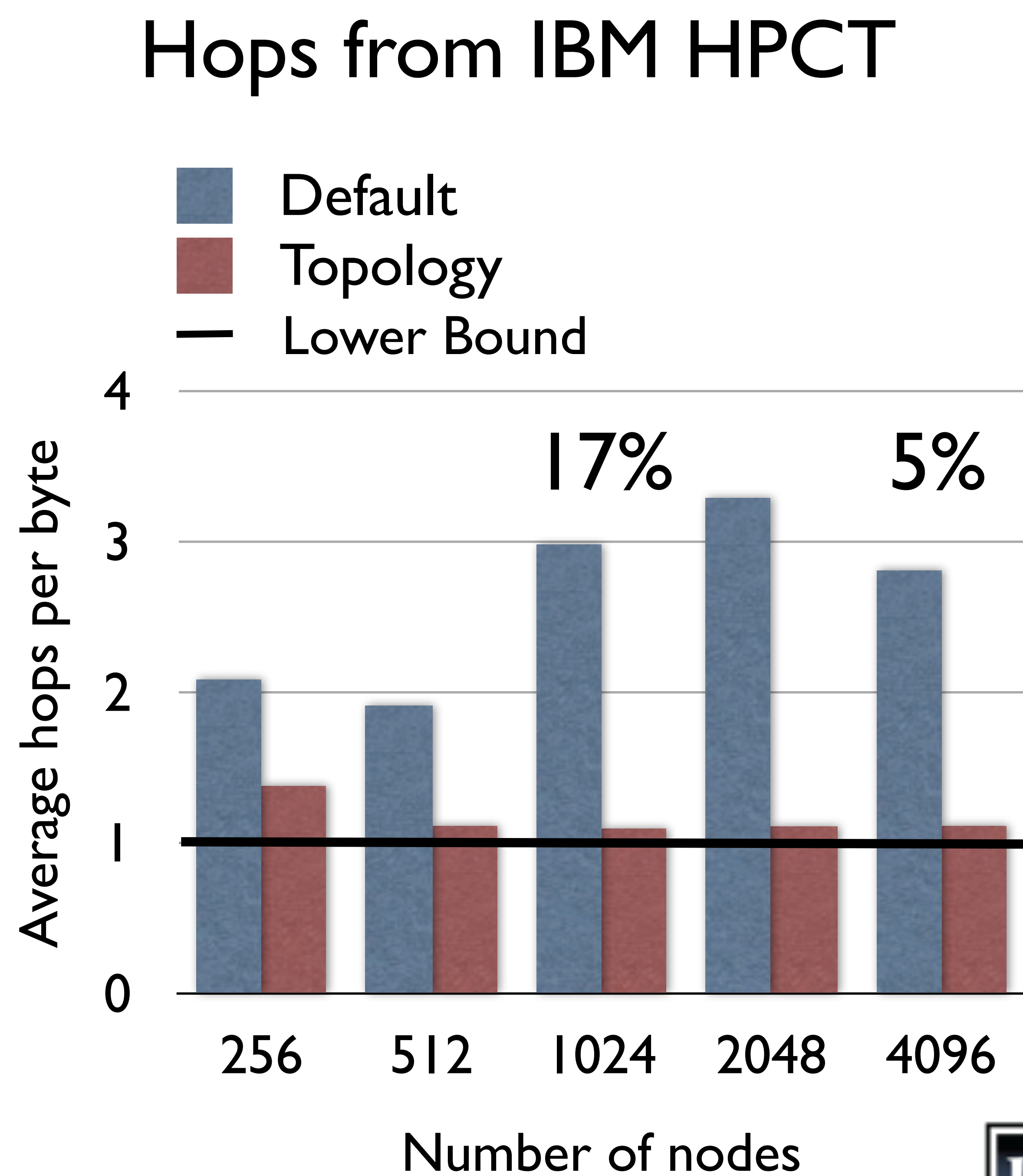
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores
- On 1024 nodes:
  - Hops reduce by: 63%
  - Time for communication reduces by 11%
  - Performance improves by 17%



Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores
- On 1024 nodes:
  - Hops reduce by: 63%
  - Time for communication reduces by 11%
  - Performance improves by 17%



Summary

- Contention in modern day supercomputers can impact performance: makes mapping important
- Developing an automatic mapping framework
  - Relieve the application developer of the mapping burden
- Topology discovery: Topology Manager API
- Object Communication Graph: Profiling, Instrumentation
- Pattern matching: regular and irregular graphs
- Suite of heuristics for mapping

Future Work

- More sophisticated algorithms for process topology discovery and mapping
  - Multicast and many-to-many patterns
- Handling multiple communication graphs
  - Simultaneous or occurring in different phases
- Extension to irregular communication graphs (in progress)

Thanks

Questions?

हाइपीसी HiPC

PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS
PPL UIUC