Wexner Medical Center

Task Mapping in Today's Supercomputers: Geometry vs. Connectivity

Ümit V. Çatalyürek

The Ohio State University

Collaborators:

Mehmet Deveci[1], Sivasankaran Rajamanickam[2], Vitus Leung[2], Kevin Pedretti[2], Stephen L. Olivier[2], David Bunde[3], Karen Devine[2], Kamer Kaya[4]

[1]The Ohio State University

[2]Sandia National Laboratories

[3]Knox College

[4]Sabanci University

9th Scheduling for Large Scale Systems Workshop

July 1-4, 2014 - Lyon, France

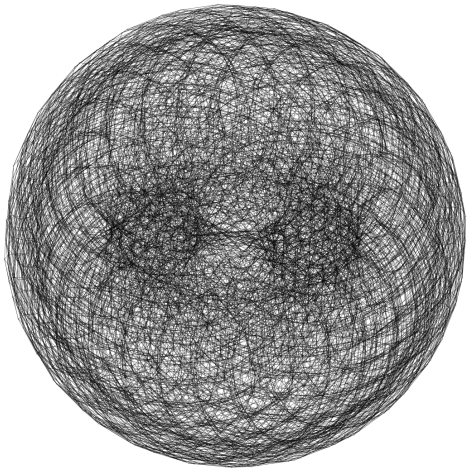
Task Mapping

Wexner Medical Center

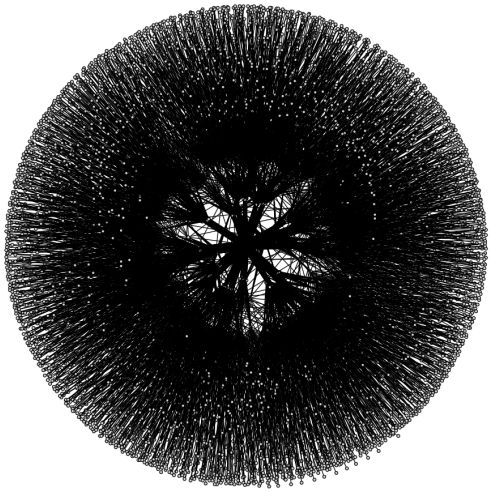
- **Problem: Mapping an application's tasks to the processors of a parallel computer**
  - Increasingly important as the number of processors grows from O(100K) to O(1M)
    - Large-diameter processor networks
    - More users submitting applications with various task sizes
    - Processor allocations are more sparse and spread further across the network
  - Communication messages can travel long routes
  - Network links might be congested by heavy traffic
- **Aim: Maintaining scalability on large-scale machines with sparse processor allocations**

Todays Machines' Network Topologies

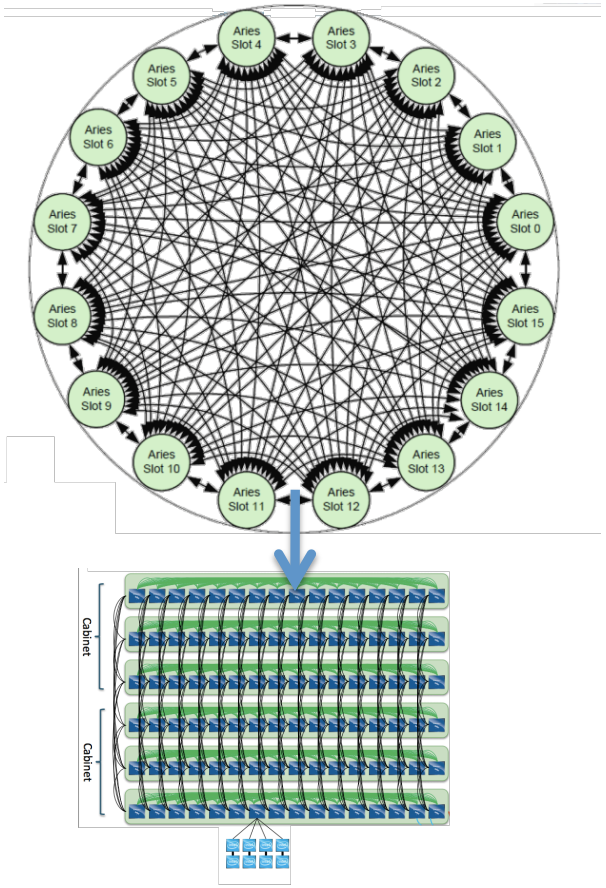
- Most of the network topologies are structural graphs
- Top10 of Top500:
  - 1 x 3D Torus
  - 4 x 5D Torus
  - 1 x 6D Torus
  - 3 x Fat-Tree
  - 1 x Dragonfly



3D Torus
Source:http://htor.inf.ethz.ch/research/topologies/



Fat-Tree
Source:http://htor.inf.ethz.ch/research/topologies/



Dragonfly
Source: https://www.nersc.gov/assets/Uploads/Edison-Overview.pdf

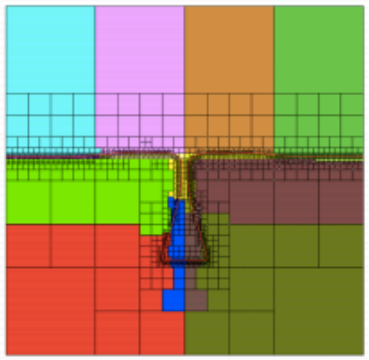
Todays Machines' Network Topologies

- For structured architectures, much of recent research has focused on mapping tasks to block- based allocations (e.g., IBM's BlueGene)
- Our focus is on non-contiguous (i.e., sparse) allocations (e.g., Cray, clusters)
  - Mapping strategies developed can be used automatically for the more restricted case of block allocations

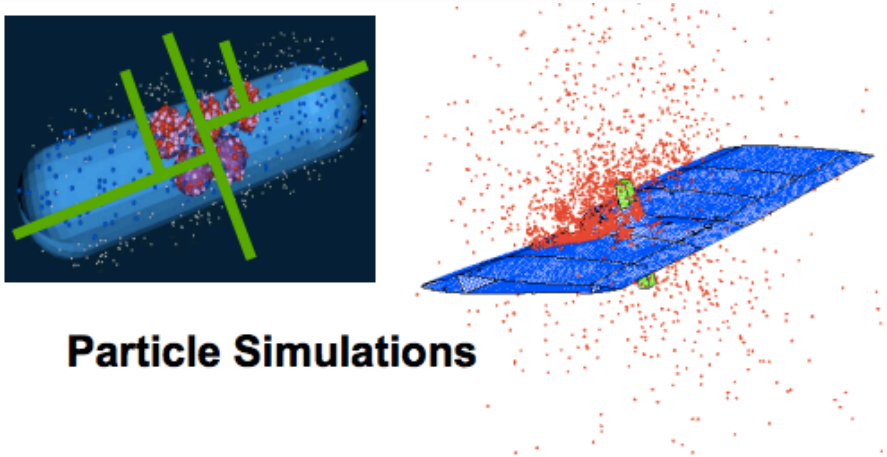
Task Mapping using Geometry

- **Geometric Models**: The machine topology and the application are represented by coordinates
  - Assumes the coordinates of the applications provide an estimation on the communication requirement
    - Not applicable to irregular applications
    - Or for the applications where coordinates do not exist.
- Fast and effective method to map regular applications.

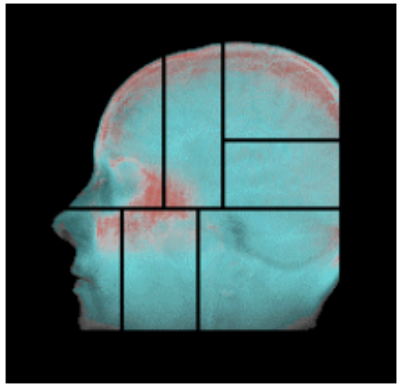
(Regular) Applications with Geometry



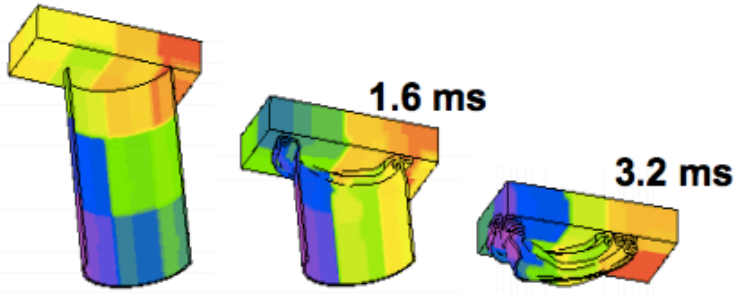
Adaptive Mesh Refinement



Particle Simulations



Parallel Volume Rendering



Crash Simulations and Contact Detection

Source: http://www.cs.sandia.gov/~kddevin/papers/Zoltan_Tutorial_Slides.pdf

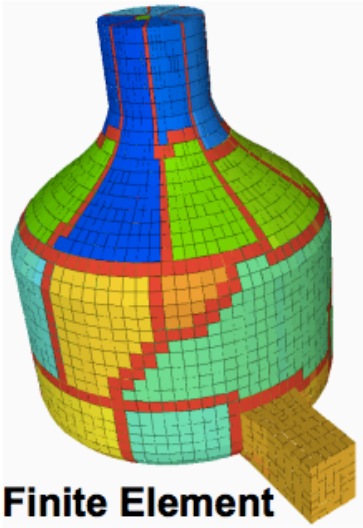
Task Mapping using Connectivity

Wexner Medical Center

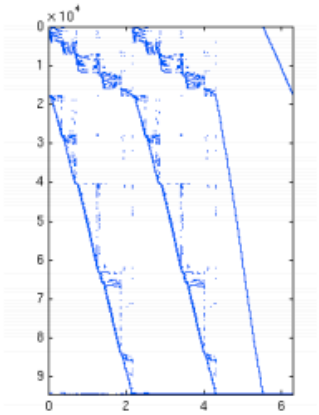
- **Graph Models:** The machine topology and the application are represented using graph models.
  - More accurate at representing the communication requirement for complex and hierarchical networks (well… not as accurate as hypergraph models but..)
  - But the mapping techniques are usually more expensive than those using geometric models
- Graph bi-partitioning
  - Recursively partitions both task and network graph, and then performs mapping [Pellegrini95]
- Heuristics for quadratic assignment problem used in Jostle: [Walshaw01]
- Graph Mapping heuristics such as: Recursive Bisection, Greedy, RCM (LibTopoMap [Hoefler & Snir 11])

(Irregular) Applications with Connectivity

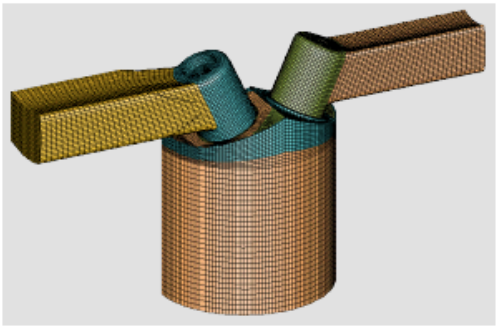
Wexner Medical Center



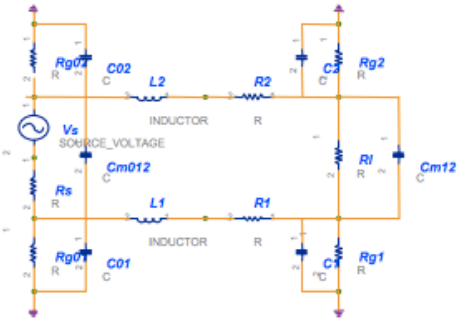
Finite Element Analysis



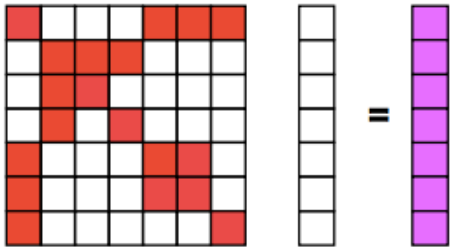
Linear programming for sensor placement



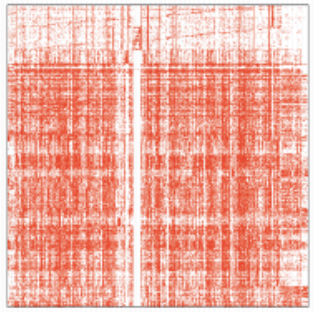
Multiphysics and multiphase simulations



Circuit Simulations



Linear solvers & preconditioners



Data Mining

Source: http://www.cs.sandia.gov/~kddevin/papers/Zoltan_Tutorial_Slides.pdf

The Question(s)

- How to perform task mapping better?
  - What are the bottlenecks and limitations we have?
- Geometry or connectivity?
  - Obviously, the answer depends on the available information.
  - What happens if both information are available?

The Metrics

- $G_t(V_t, E_t)$ is the application communication graph
  - Vertices V_t are MPI processes and edges E_t represent the amount of communication between processes
- $G_n(V_n, E_n)$ is the topology graph
  - V_n is the set of nodes and E_n is the set of links
  - Each processor has a coordinate (x,y,z) on a 3D torus network
- Γ is the mapping function from V_t -> V_n

$$hopcount(t_1,t_2) = ShortestPathLength(\Gamma(t_1),\Gamma(t_2))$$

$$TotalHopCount(\Gamma) = \sum_{(t_1,t_2)\in E_t} hopcount(t_1,t_2)$$

$$WeightedHopCount(\Gamma) = \sum_{(t_1,t_2)\in E_t} Vol(t_1,t_2)\times hopcount(t_1,t_2)$$

$$AverageHopCount(\Gamma) = TotalHopCount(\Gamma) / |E_t|$$

Mapping Metrics

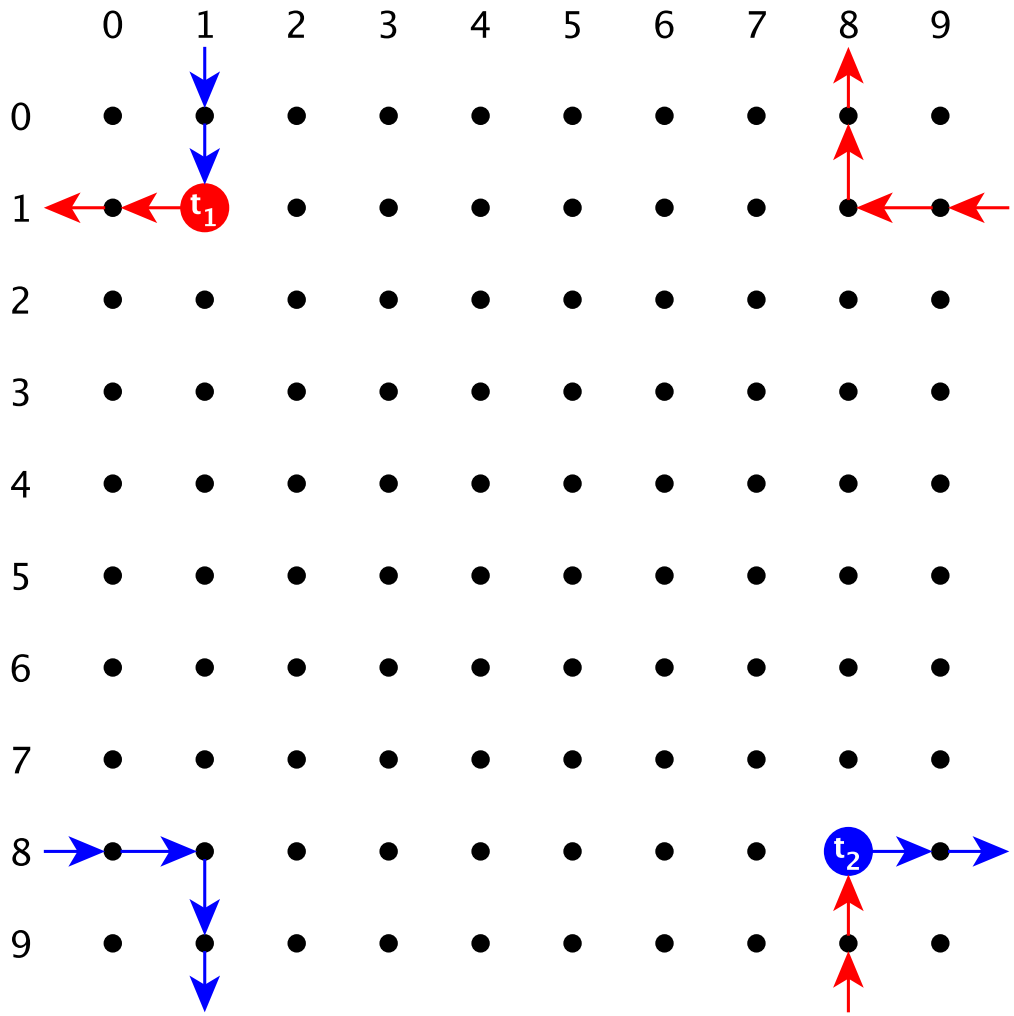
- Link congestion: The ratio of the volume goes through the link to its capacity (bandwidth)

$$Congestion(e) = \sum_{(t_1,t_2)\in E_t} \frac{Vol(t_1,t_2)}{Bandwidth(e)} \times inShortestPath(e,\Gamma(t_1),\Gamma(t_2))$$

$$MaxCongestion(\Gamma) = \max_{e\in E_n}(Congestion(e))$$

- We assume:
  - Messages take one of the shortest path(s)
  - Static routing of messages
  - Messages are transferred via single path
    - E.g., in 3D torus: first X, then Y, then Z

Mapping Metrics



$$hopcount(t_1, t_2) = 6$$

$$hopcount(t_2, t_1) = 6$$

$$TotalHopCount(\Gamma) = 12$$

$$AverageHopCount(\Gamma) = 6$$

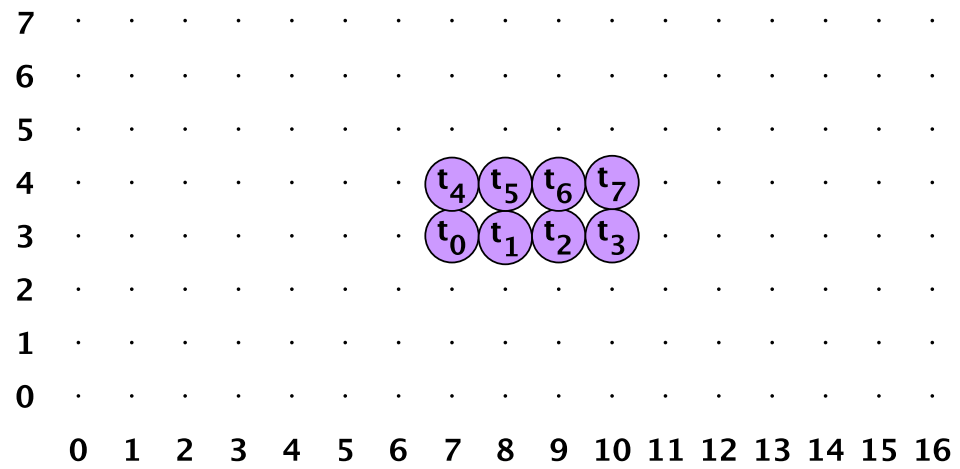
$$MaxCongestion(\Gamma) = 1$$

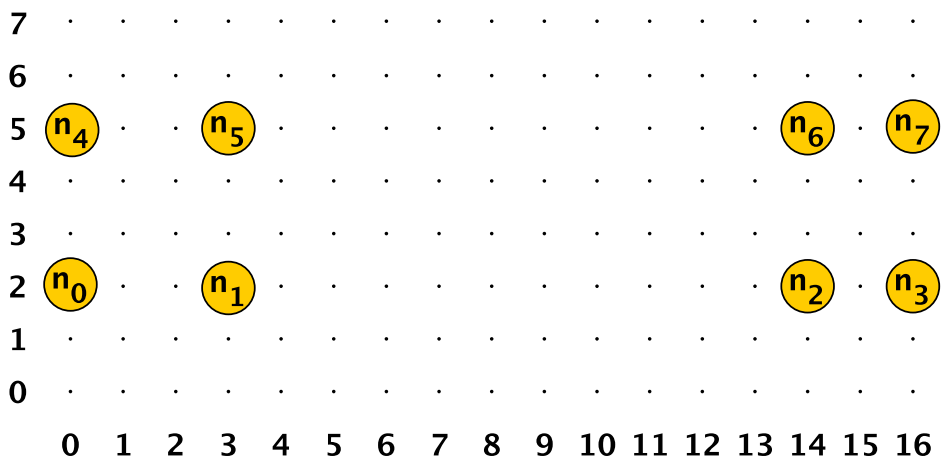
Geometric Task Mapping [IPDPS14]

- **Geometric Model:**
  - The machine topology is described only by the cores' coordinates, rather than a topology graph
  - Application's MPI processes are also represented by coordinates
    - the center of the process' application domain
    - the average coordinate of its application data
- **A geometric partitioning algorithm** is used to consistently reorder both the MPI processes and the allocated cores
  - The ordering is used to construct the mapping

[IPDPS14] M. Deveci, S. Rajamanickam, V. Leung, K. T. Pedretti, S. L. Olivier, D. P. Bunde, Ü. V. Çatalyürek, K.D. Devine, "Exploiting Geometric Partitioning in Task Mapping for Parallel Computers", IPDPS, 2014.

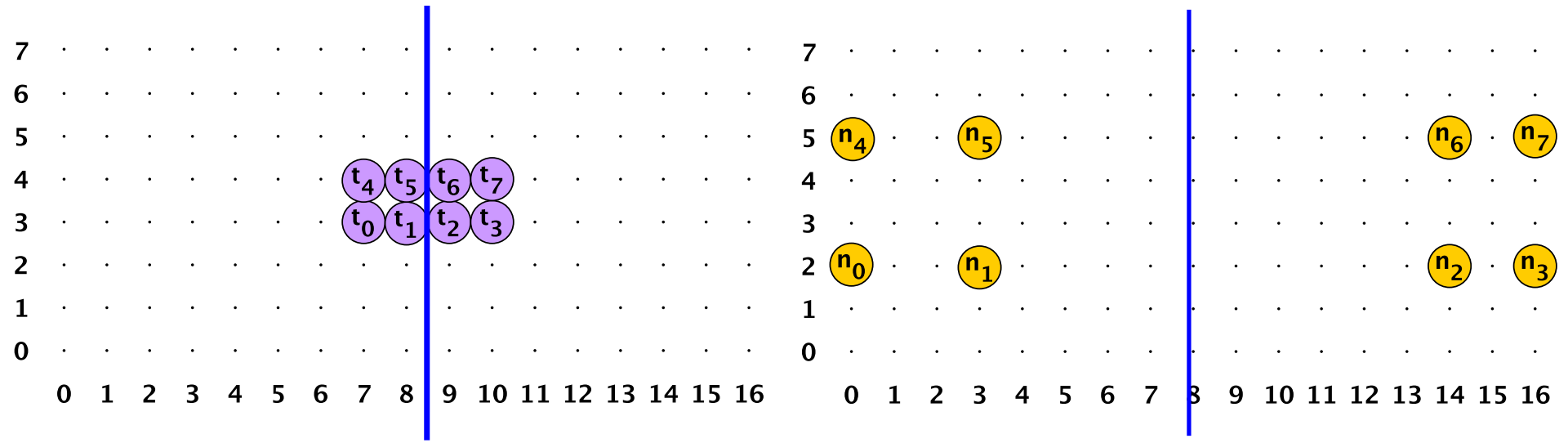
Task Mapping Example



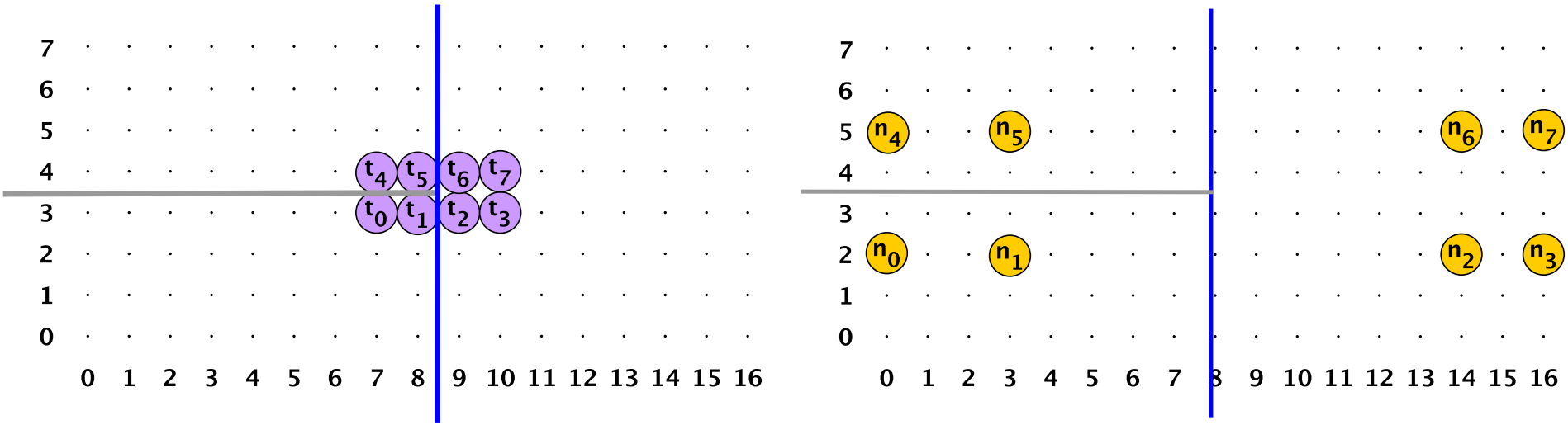


- Given the task coordinates
  - tasks communicate with their immediate neighbors
- And the machine coordinates
  - 17x8 2D Torus with a wrap-around

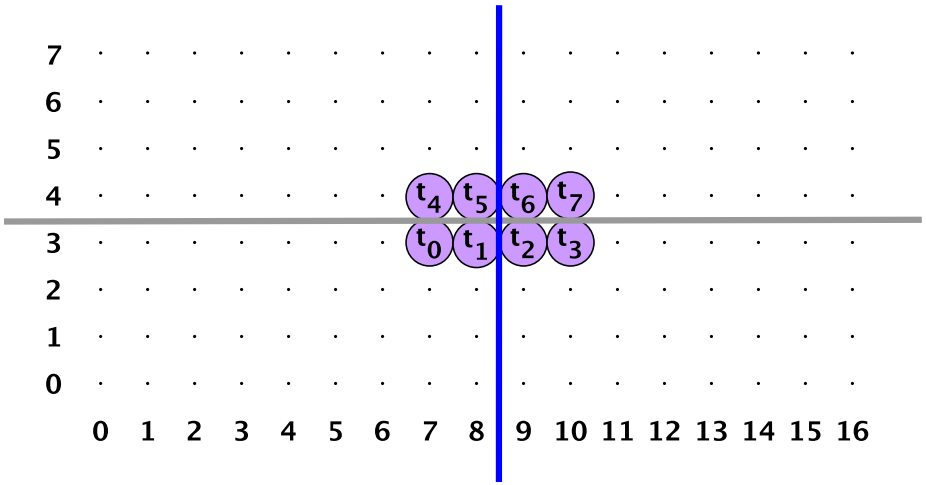
Task Mapping Example

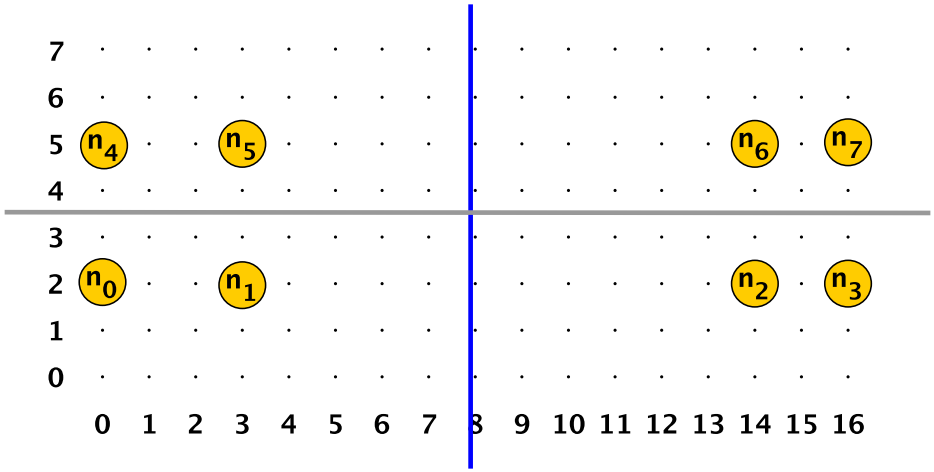


Task Mapping Example

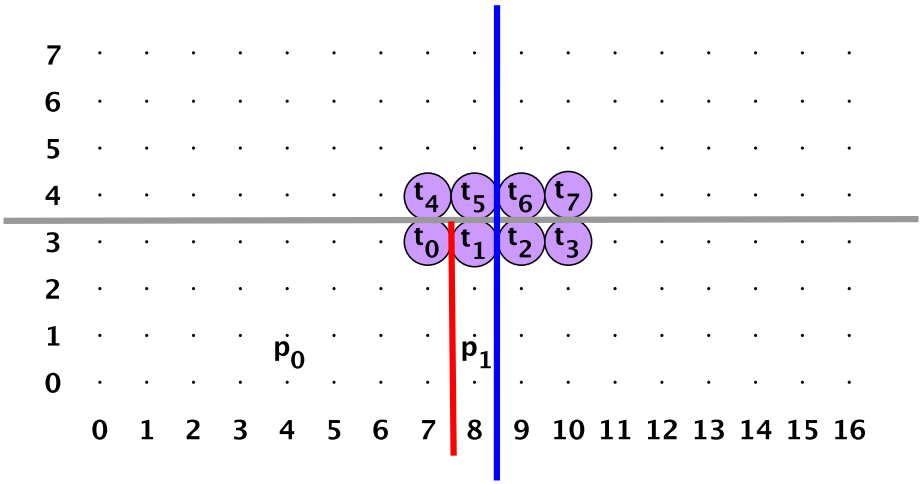


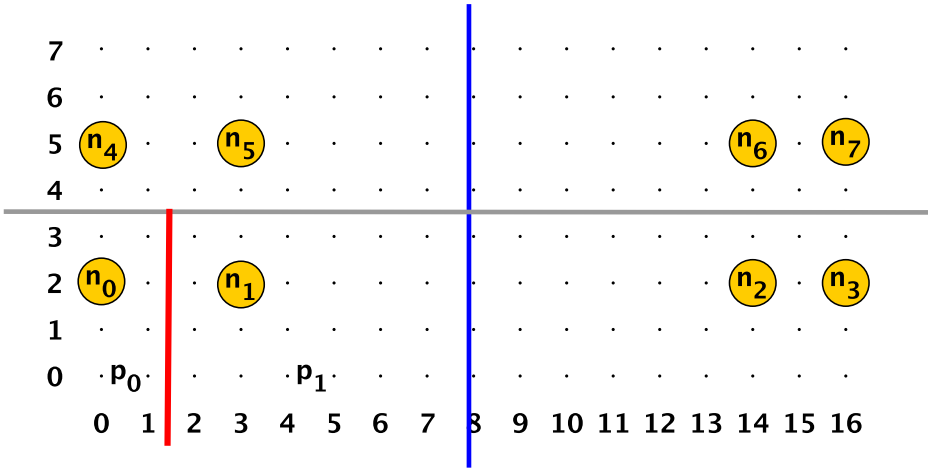
Task Mapping Example



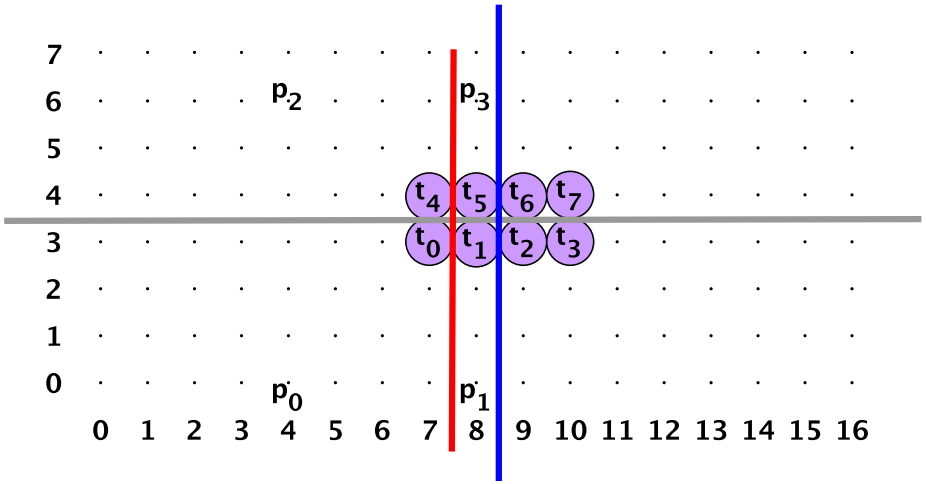


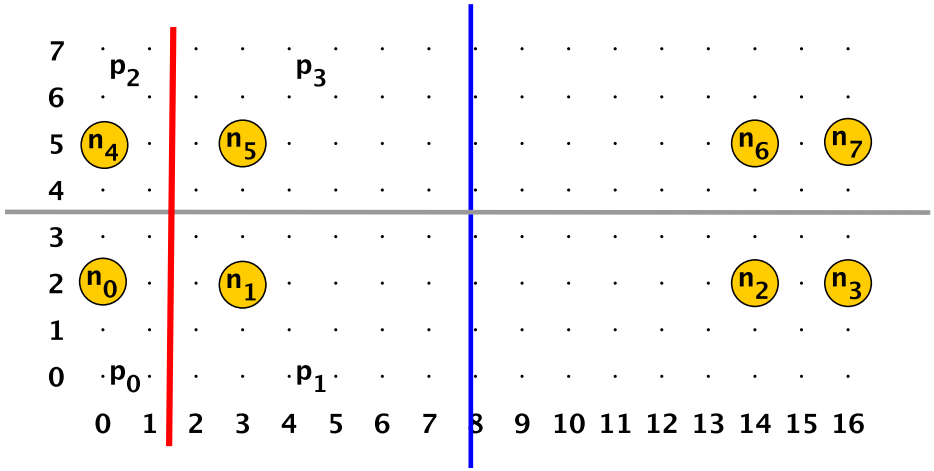
Task Mapping Example Partitioning Tasks



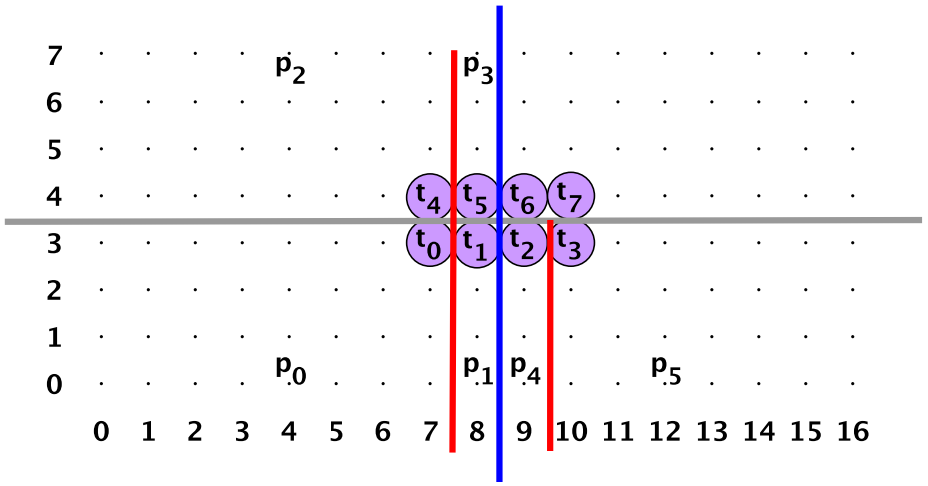


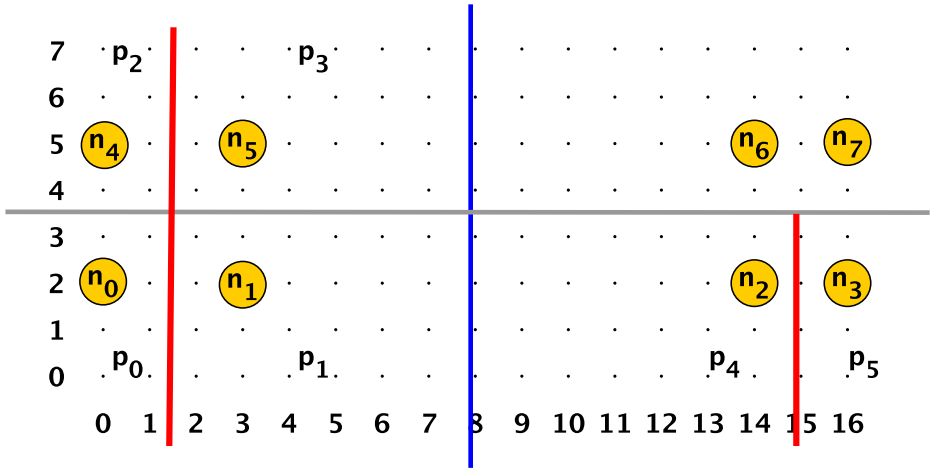
Task Mapping Example Partitioning Tasks



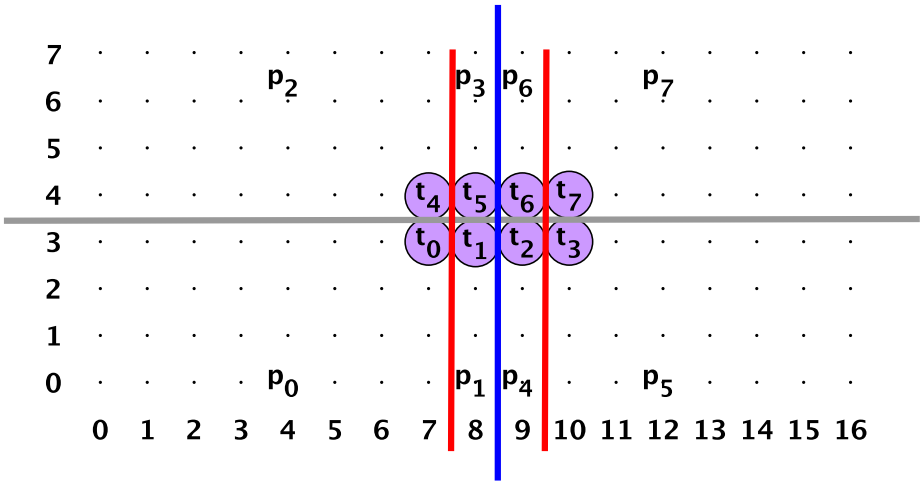


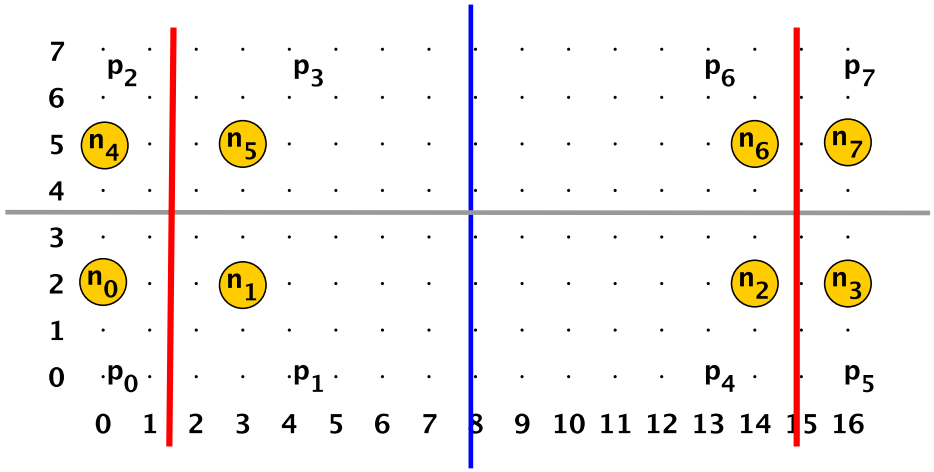
Task Mapping Example Partitioning Tasks



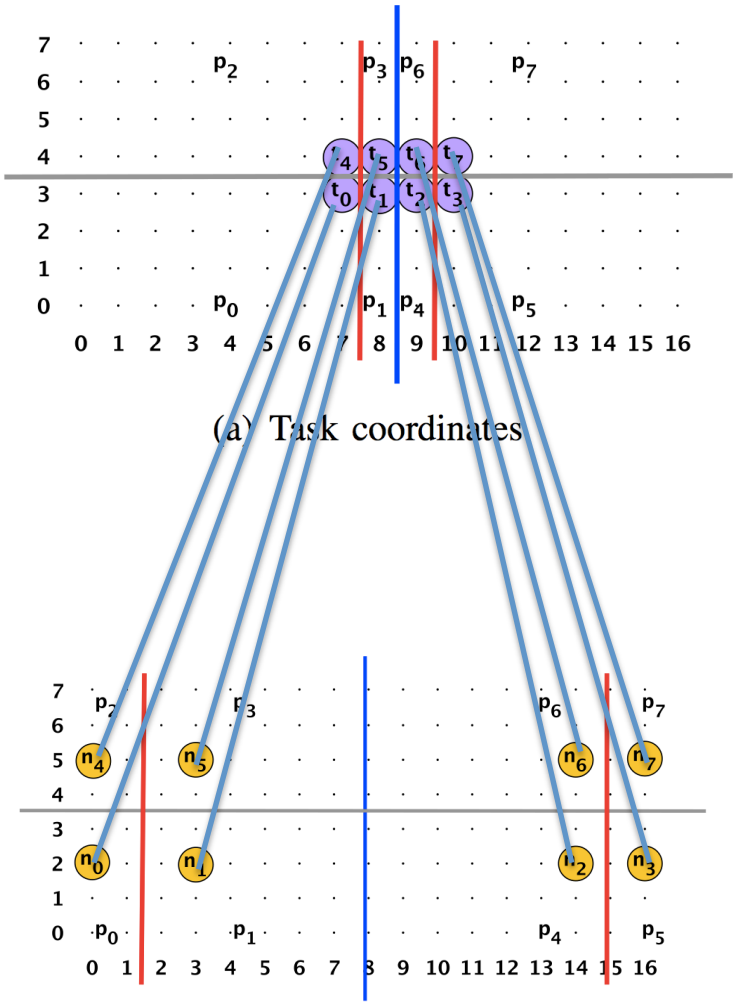


Task Mapping Example Partitioning Tasks





Task Mapping Example Partitioning Tasks



(a) Task coordinates

(b) Node coordinates

- 17x8 2D Torus with a wrap-around
- Total hop count is 34
  - (22 and 12 along x and y).

$hop(T_4\text{-}T_5) = (3,0)$
$hop(T_5\text{-}T_6) = (6,0)$
$hop(T_6\text{-}T_7) = (2,0)$

$hop(T_0\text{-}T_1) = (3,0)$
$hop(T_1\text{-}T_2) = (6,0)$
$hop(T_2\text{-}T_3) = (2,0)$

$hop(T_4\text{-}T_0) = (0,3)$
$hop(T_5\text{-}T_1) = (0,3)$

$hop(T_6\text{-}T_2) = (0,3)$
$hop(T_7\text{-}T_3) = (0,3)$

Heuristics to Improve the Quality

- The ability to reduce communication costs depends on the results of the partitioner
- It is possible to improve the quality of the mapping by modifying the input data provided to partitioner
  - Shifting the machine coordinates
  - Rotating the machine and/or task coordinates

Graph Mapping Methods [ongoing work]

Wexner Medical Center

- We have also studied a graph model with a single level greedy graph mapping algorithm and a refinement mechanism.
  - First, we use greedy graph growing methods
    - In order to map the highly communicating tasks to closer nodes.
      - Minimizing weighted hops
  - Then, a swap-based refinement method to reduce the maximum and average congestion

Experiments

- Real experiments using geometric mapping methods on a proxy application: MiniGhost
  - Weak scaling experiments to evaluate the effect of mapping on communication time and execution time
  - We compared our geometric methods with
    - The applications' default task layout
    - Application-specific grouping of tasks for multicore nodes
    - The graph-based task mapping library LibTopoMap
- Simulated experiments using graph mapping methods on irregular applications: e.g., SpMV

Computing Environment

Wexner Medical Center

- DOE Cielo Cray XE6 at Los Alamos National Laboratory (16 cores per node)
- 3D torus with wrap-around
  - Two computing nodes are in a single Gemini Node having (x,y,z) coordinates
- Network Dimensions:
  - 16x12x24
  - 4,608 Gemini Nodes, 9,216 Computing Nodes, 147,456 cores
- gcc 4.7.2 compilers and Cray's MPICH2

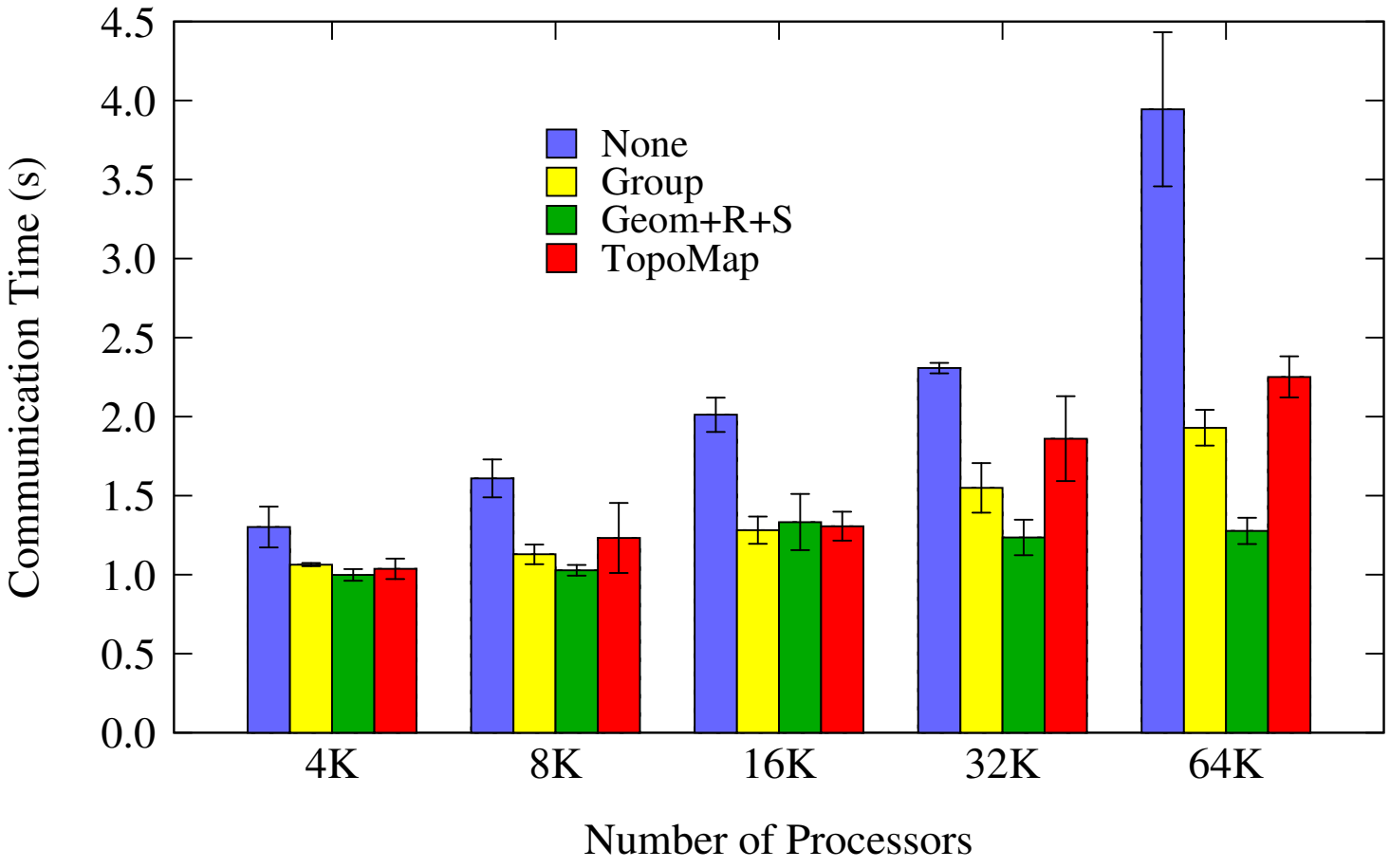
Minighost Weak Scaling Experiments

- A finite-difference proxy application
  - implements a finite difference stencil and explicit time-stepping scheme across a three-dimensional uniform grid
  - each task communicates with two neighbors in each dimension
  - does not scale to high core counts in weak scaling tests
  - goal is to improve scalability through task mapping
- Weak Scaling with 4,096–65,536 processors
- For each experiment, we obtained a node allocation of the requested size
  - All mapping methods run within same allocation
- Each experiment was repeated five times with different allocations

Mapping Methods

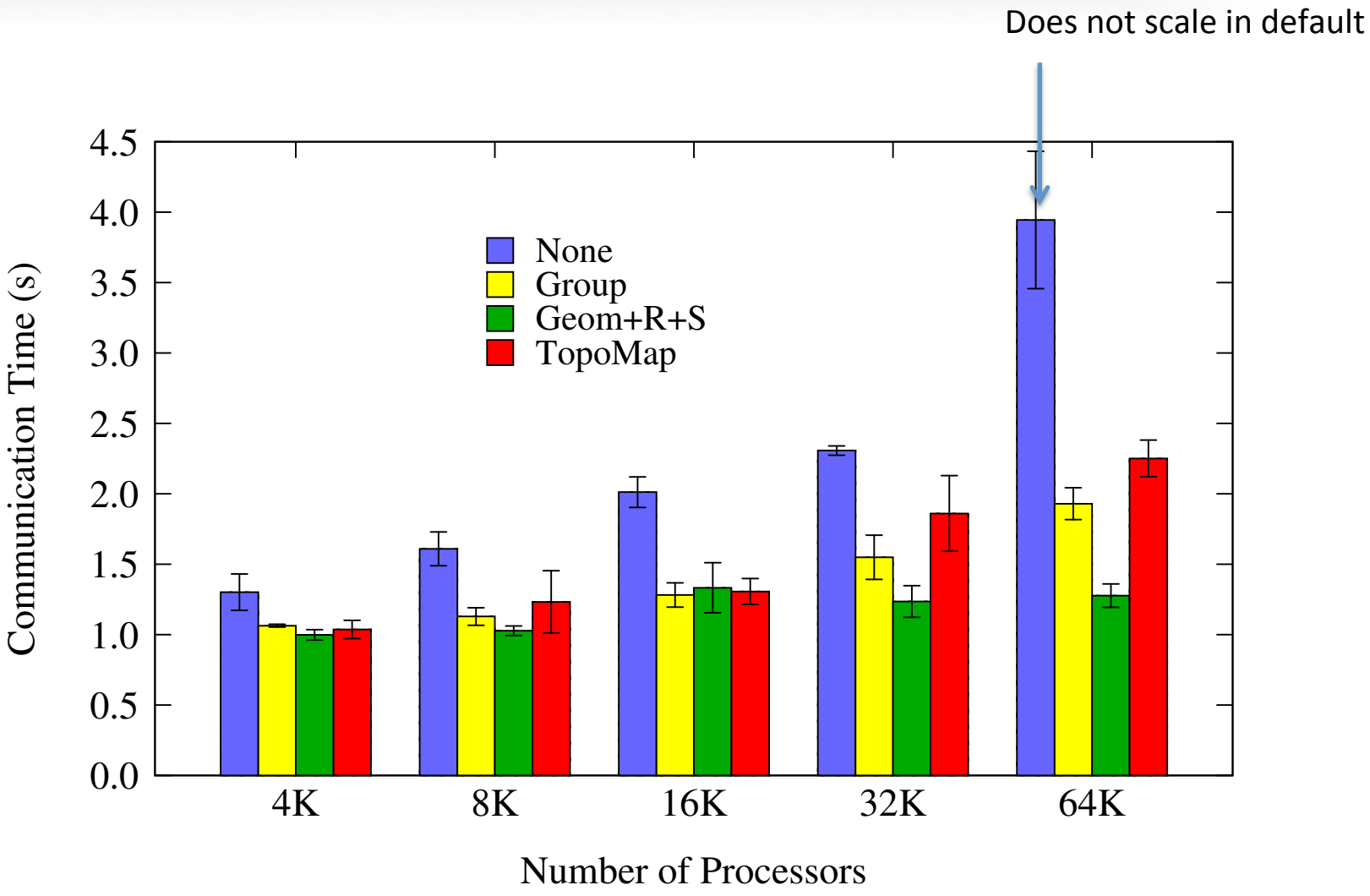
- Multicore Grouping (Group):
  - Tasks reordered into 16-task blocks: 2x2x4 tasks per block
  - A block, which contains frequently communicating tasks, is then assigned to cores within the same node
  - Does not account for inter-node communication
- LibTopoMap (TopoMap):
  - Graph-based mapping in LibTopoMap
  - The best of Greedy, Recursive Bisection, and Reverse Cuthill-McKee (RCM)
- Geometric (Geom + R + S):
  - Recursive coordinate bisection
  - 3! x 3! = 36 different rotations/solutions in 3D (computed parallel)
  - Coordinate shifting

Maximum Communication Time

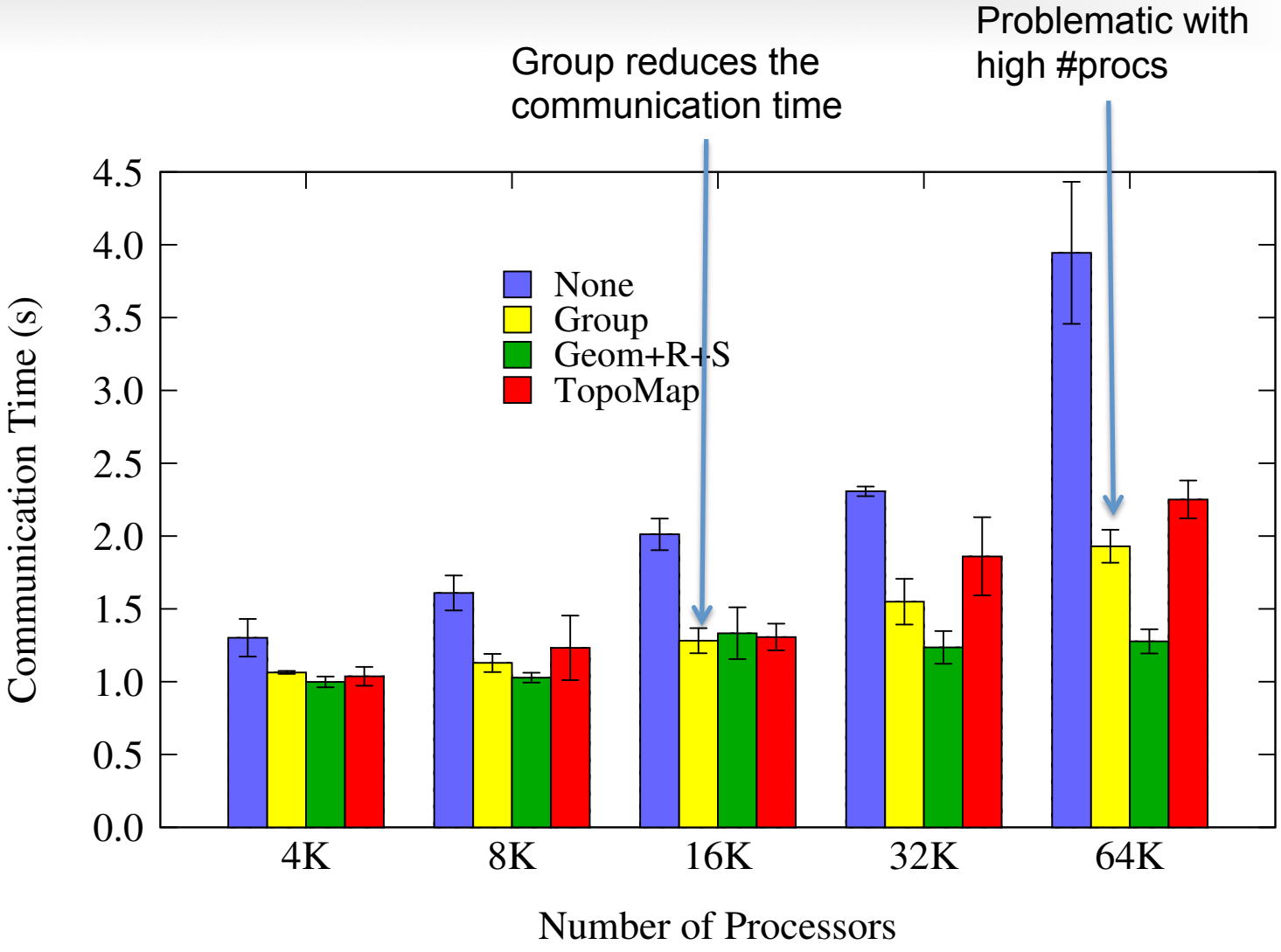


Maximum Communication Time

Wexner Medical Center



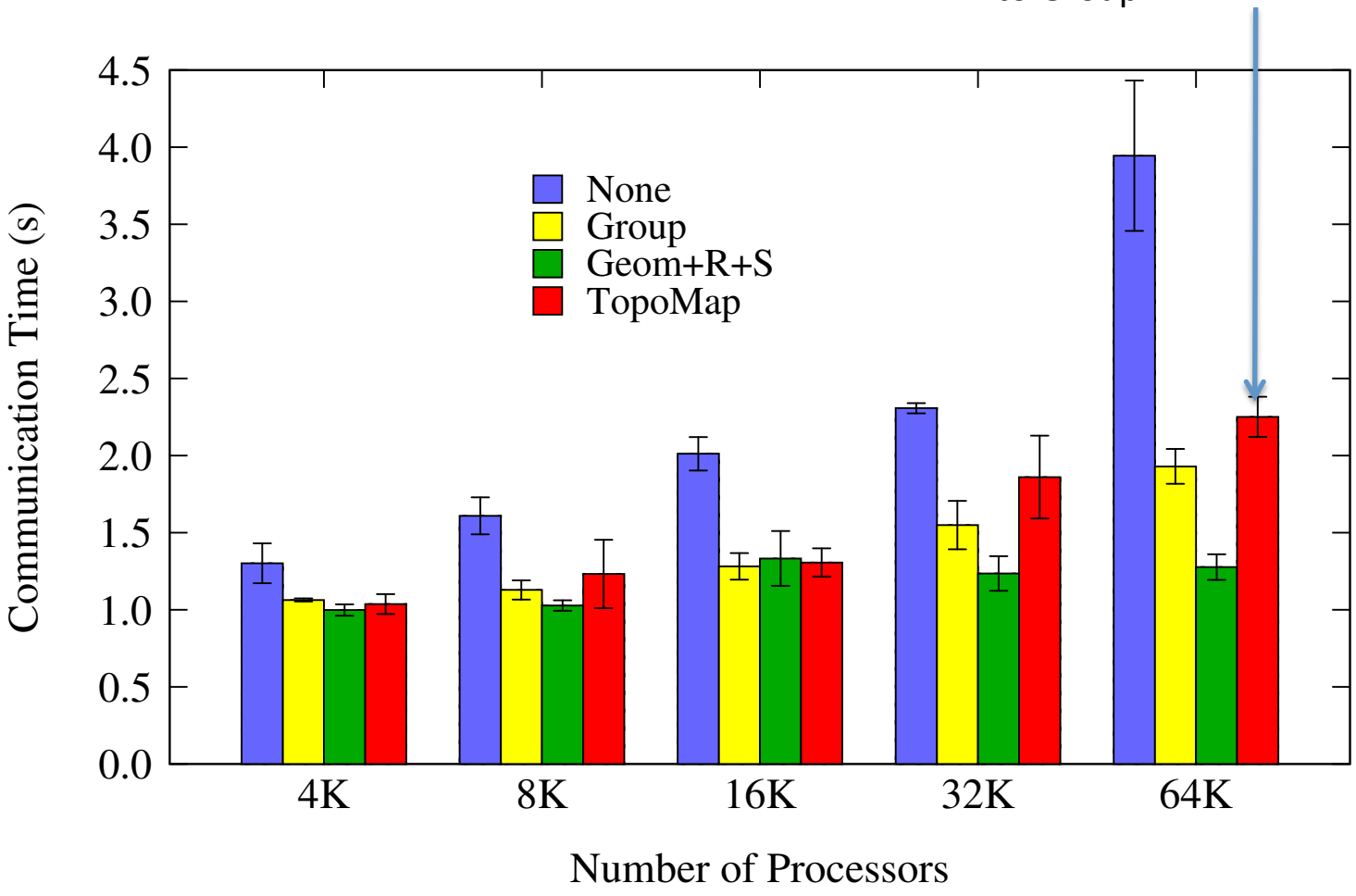
Maximum Communication Time



Maximum Communication Time

Wexner Medical Center

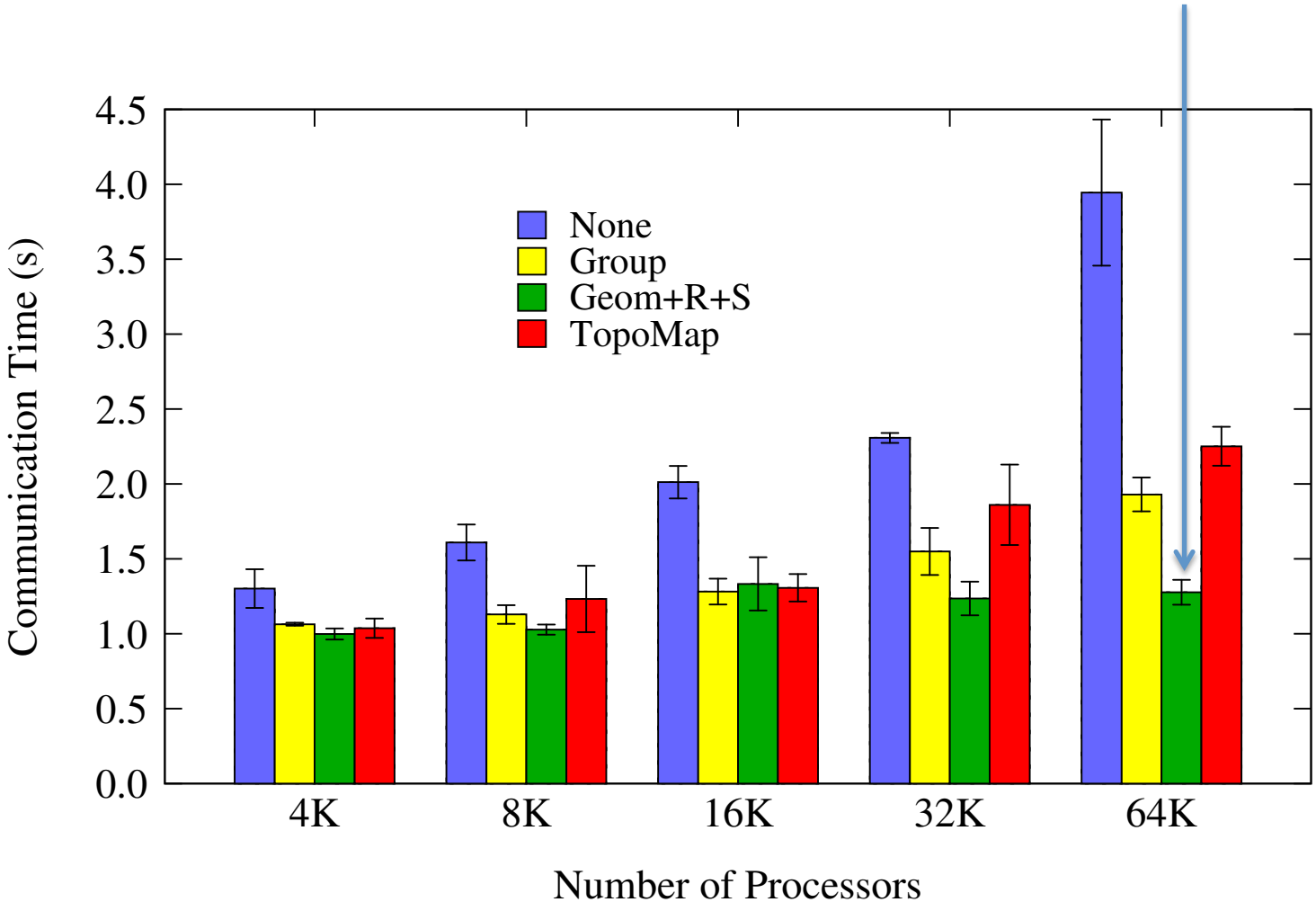
TopoMap performance is similar to Group



Maximum Communication Time

Wexner Medical Center

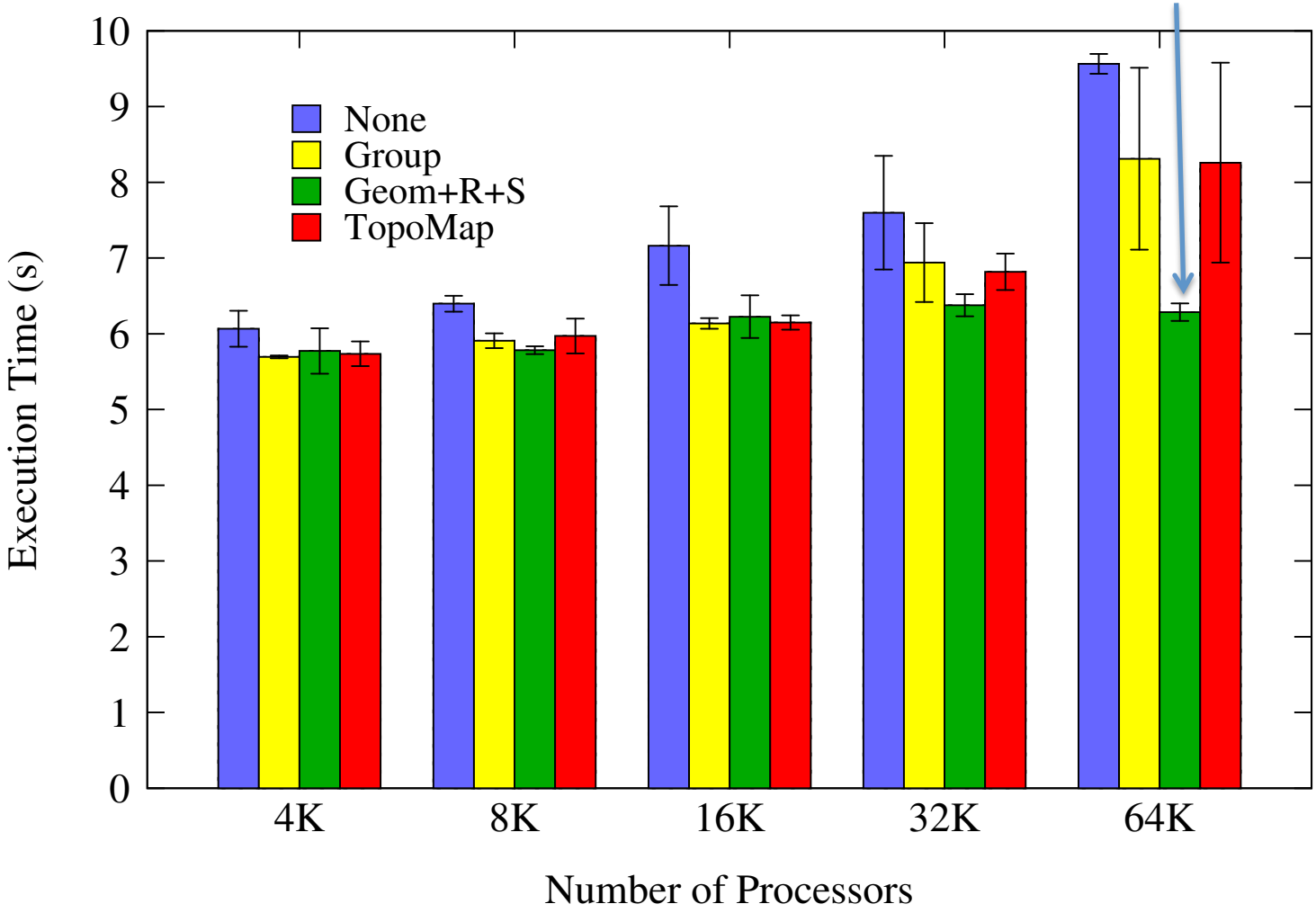
Geom + R + S obtains the best performance:



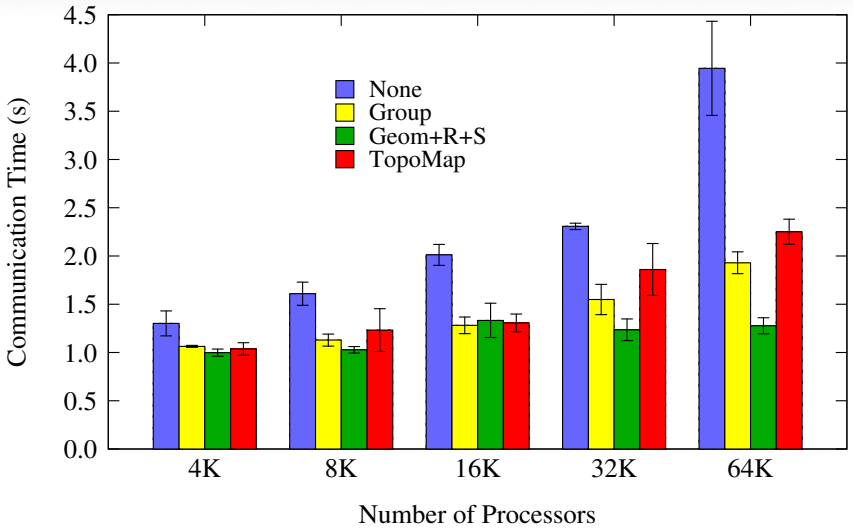
Total Execution Time

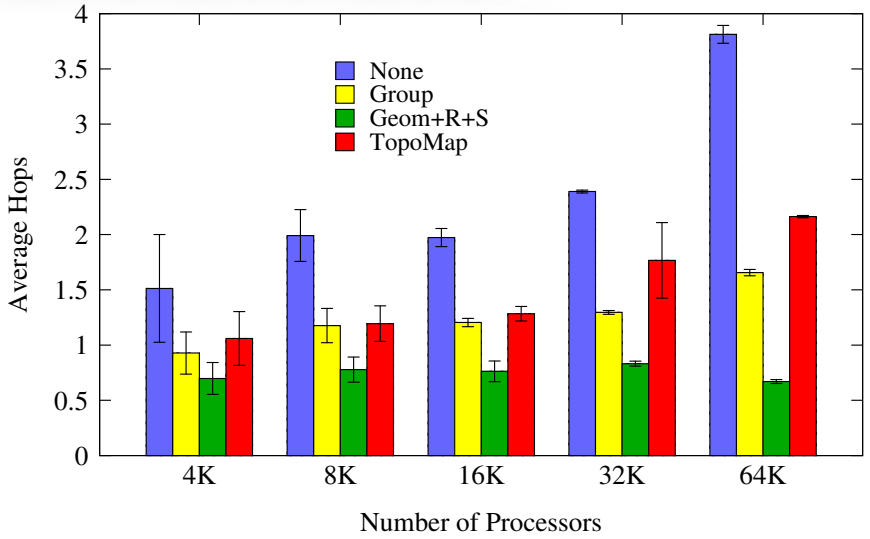
Wexner Medical Center

Geom + R + S obtains the best performance:
34% reduction

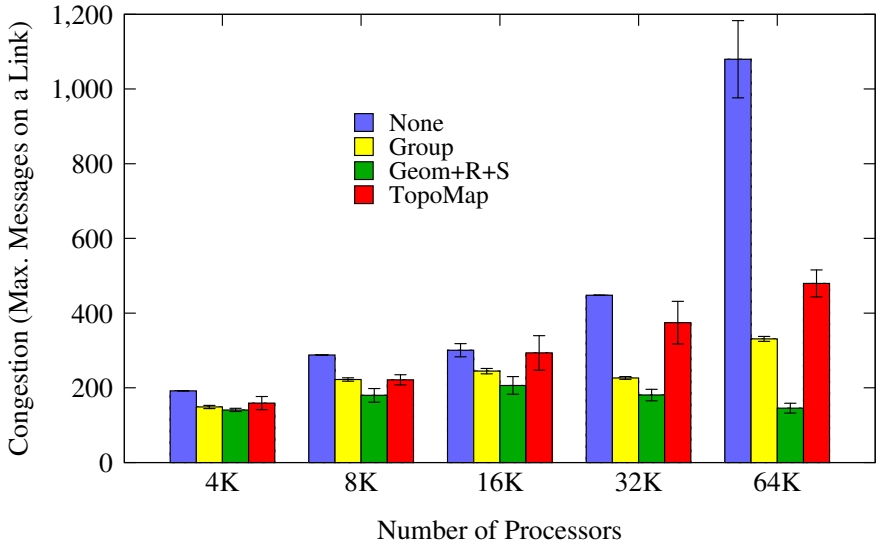


Max Communication Time, Average Hop Count & Max Congestion

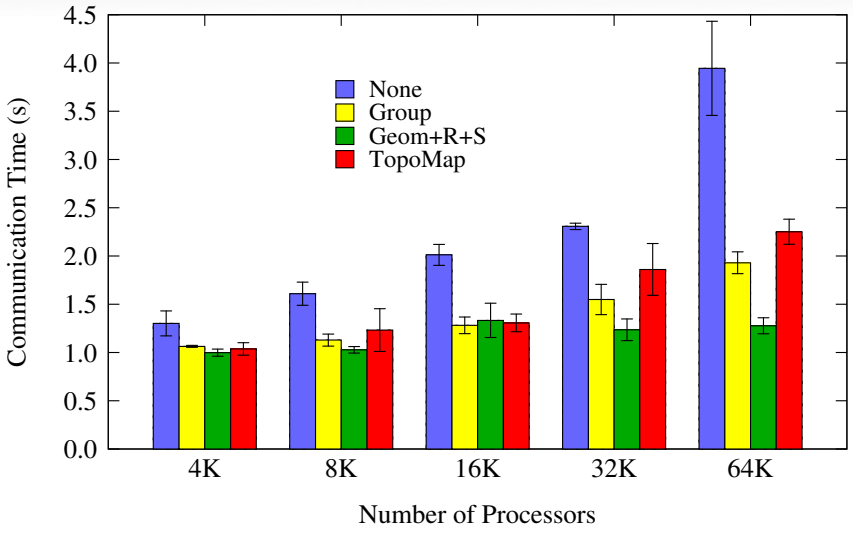


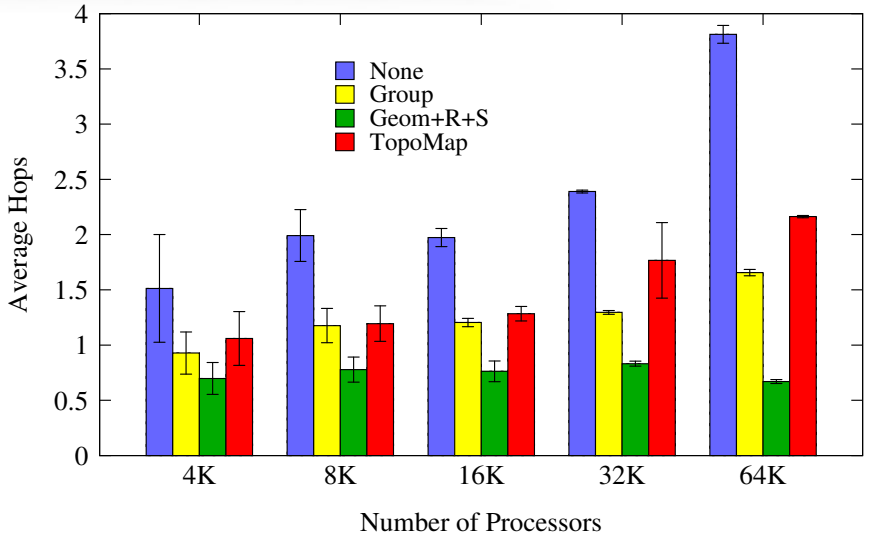


- Group does not account for inter-node communication
  - Avg hop count and max congestion increase with increasing # procs
- TopoMap's average hop count and max congestion also increase

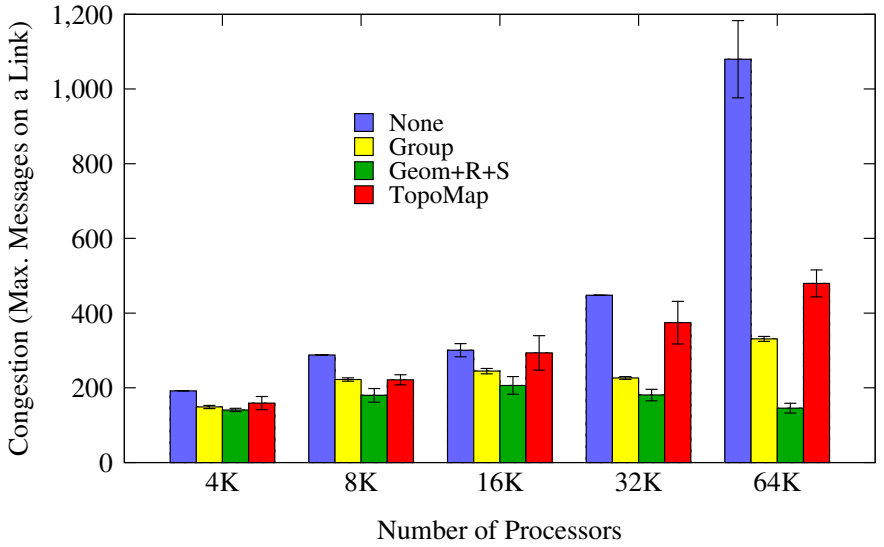


Max Communication Time, Average Hop Count & Max Congestion





- Avg hop count and maximum congestion for geometric method remain nearly unchanged
  - Furthermore, they decrease on 64K, even though the allocation of the nodes become denser.



Metric Correlations

Wexner Medical Center

PCC	Avg Hop Count	Max Congestion	Measured Max Stall
Max Comm Time	0.835	0.915	0.940
Total Time	0.760	0.872	0.885

- We used Cray Gemini's performance counters to measure network congestion empirically using the Gemini's per-link stall cycle counters.
  - Stall counter is incremented whenever a message can not move towards its destination due to network congestion.
  - Maximum stall count among the all links
- Max Stall metric is found to have the best correlation to maximum communication time.
- The calculated maximum congestion metric correlates slightly less well to the measured communication times;
  - interference from other jobs
  - all messages are not transferred simultaneously
  - heterogeneous link speeds in the network, for which we do not account
- Congestion metrics are more accurate for MiniGhost and it should be preferred over the average hop count metric.

MiniGhost Mapping using Graph Models

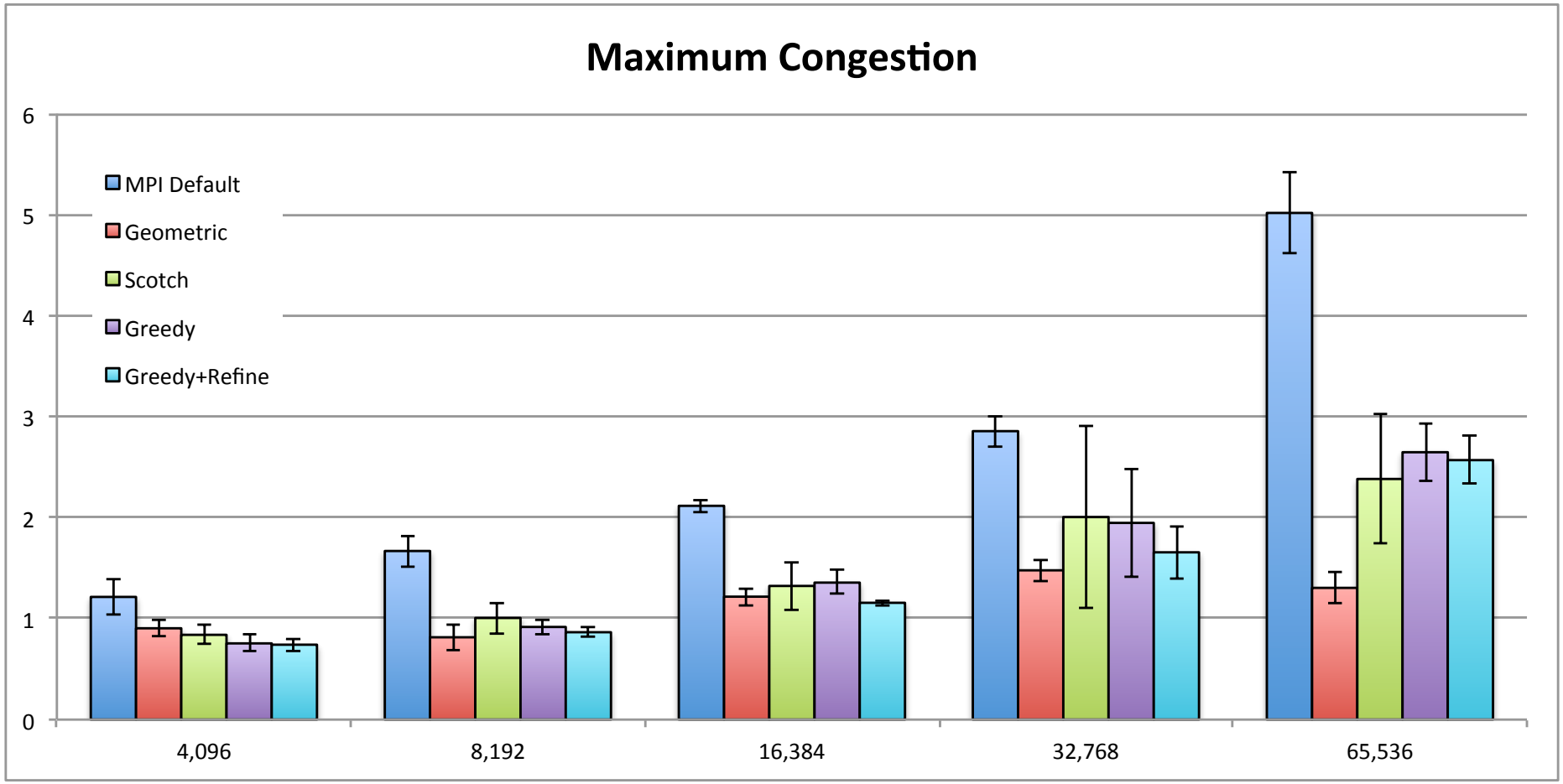
Wexner Medical Center

- We simulate the MiniGhost experiments, using the
  - The developed graph mapping methods
    - Greedy graph graph growing
    - Greedy graph graph growing + Refinement method
  - Scotch's bi-partitioning mapping.

on Cielo supercomputer topology by randomly generated allocations.

Minighost Mapping using Graph Models

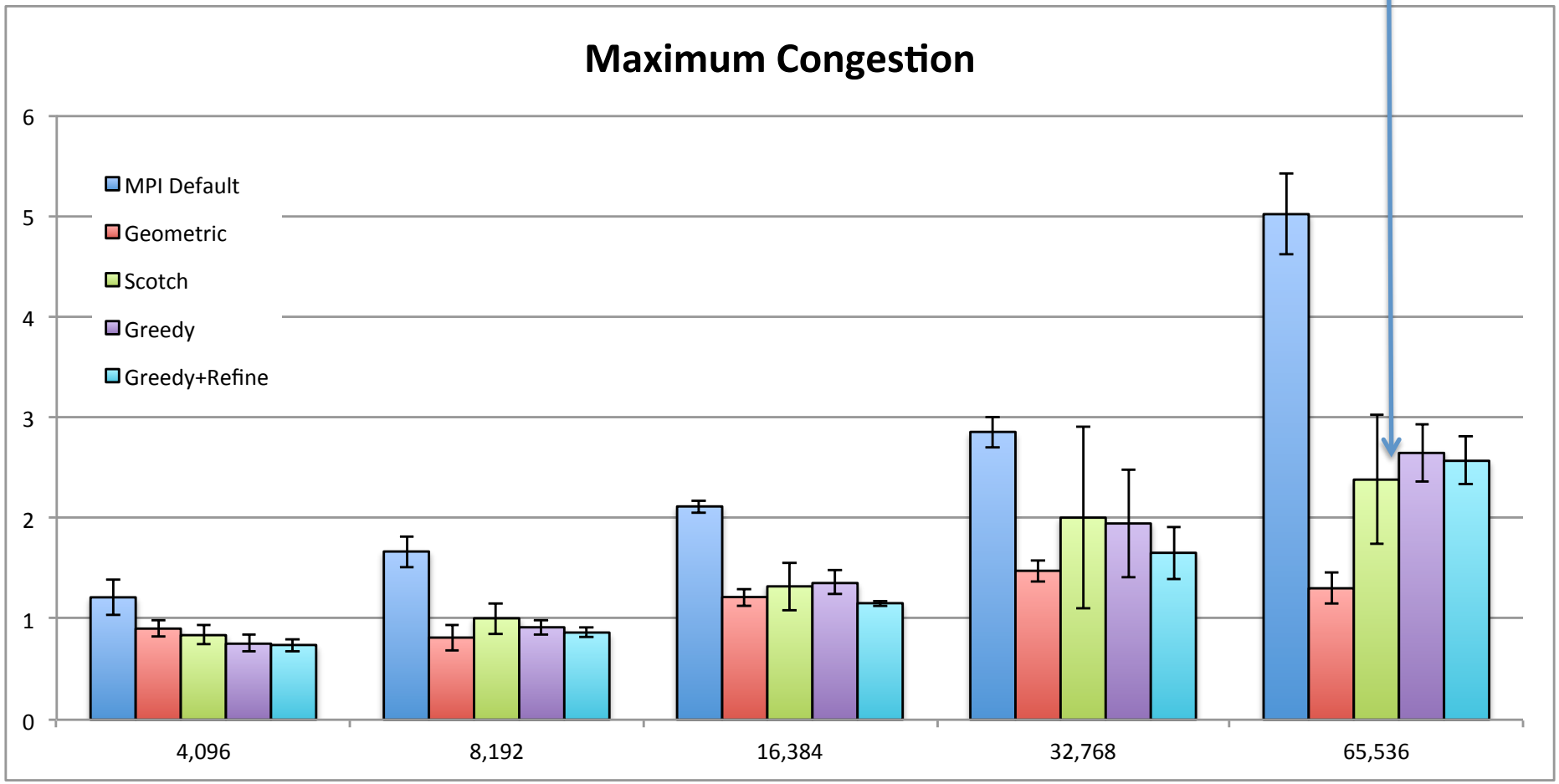
Wexner Medical Center



Minighost Mapping using Graph Models

Wexner Medical Center

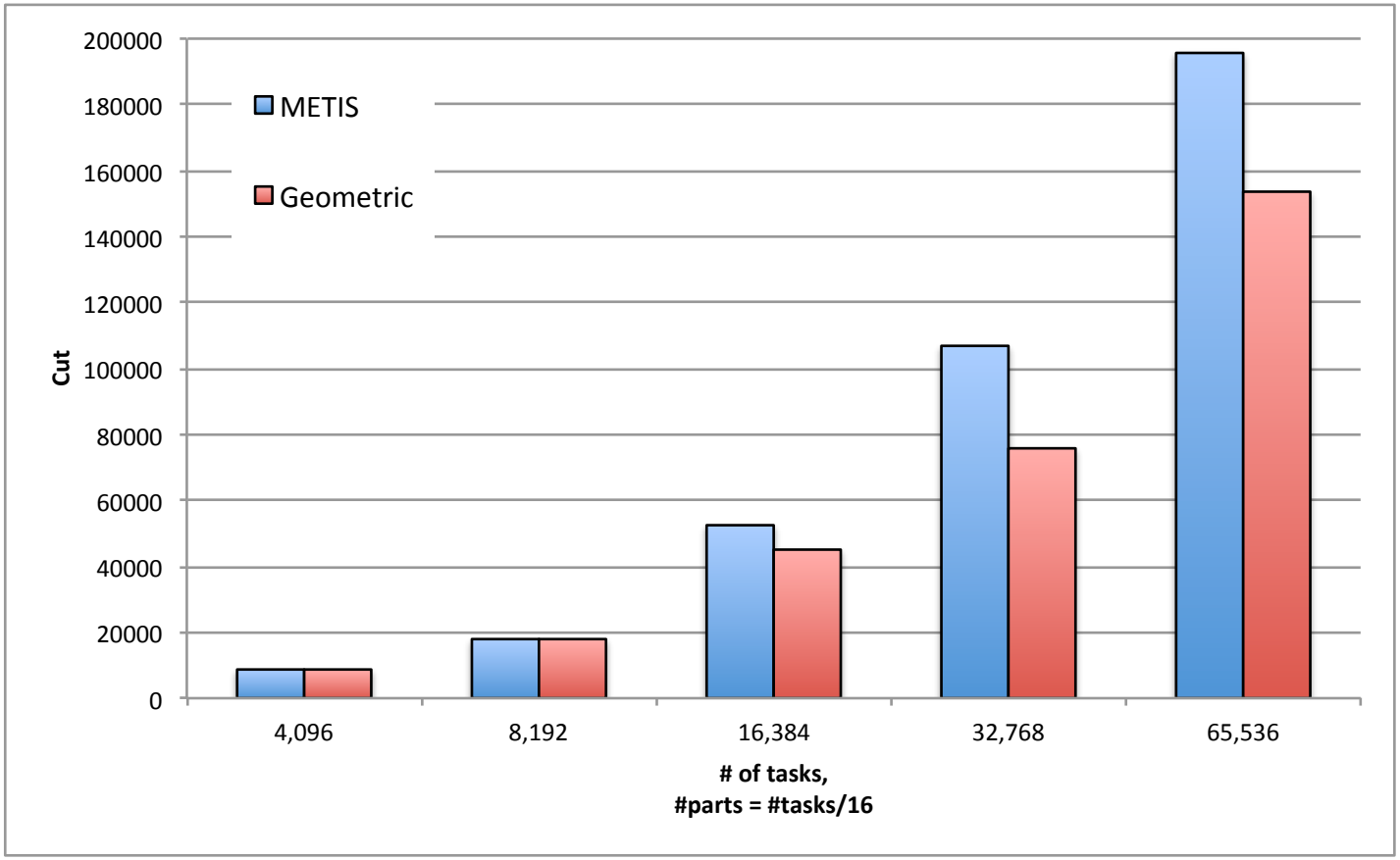
Mapping methods using graph models obtain lower quality mappings as #procs increases



Partitioning Quality on Structured Graphs

Wexner Medical Center

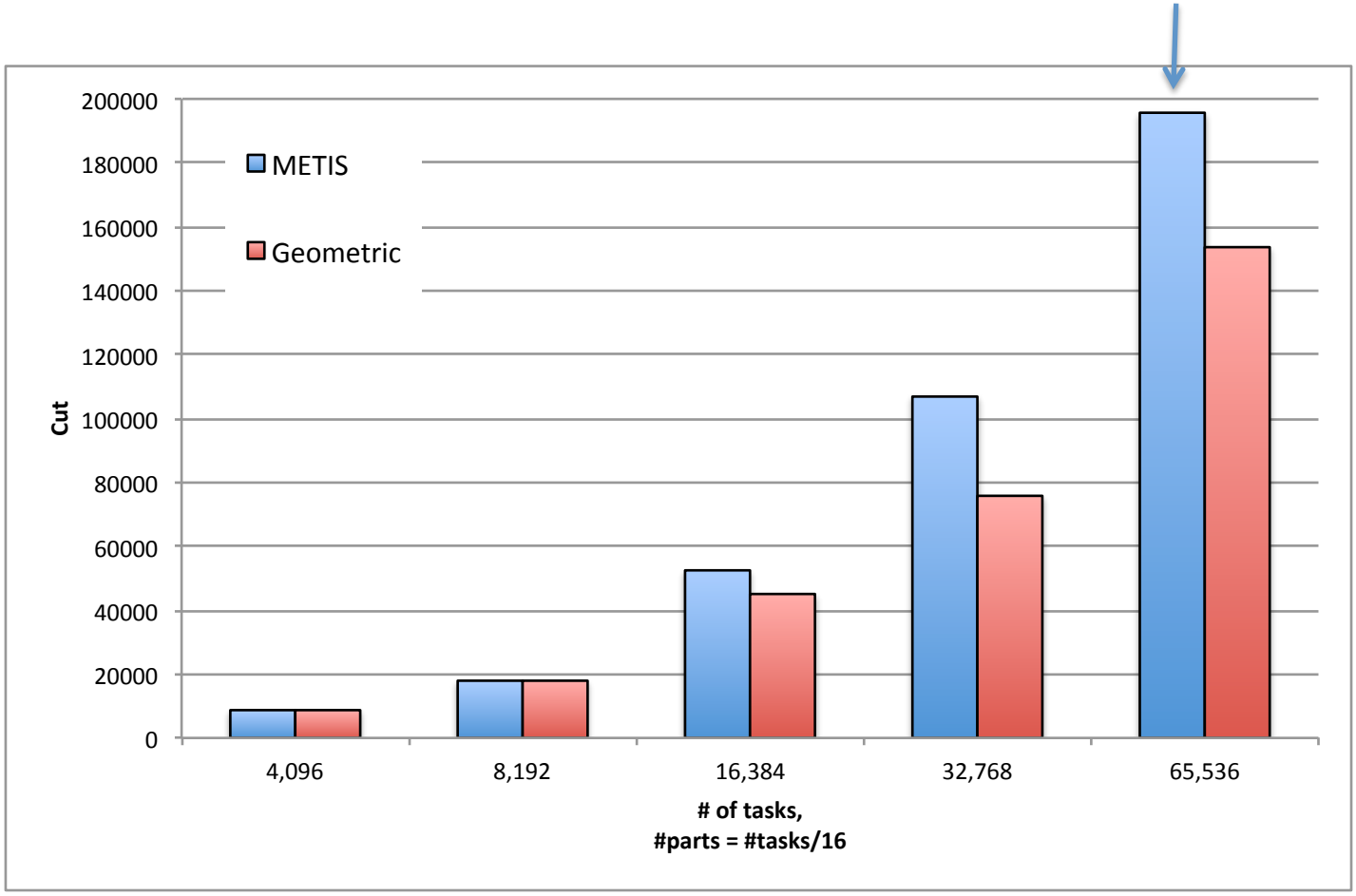
There are 16 cores per node in Cielo. Therefore, the edge-cut of the partitioning of task communication graph into (#procs / 16) parts estimates the inter-node communication volume



Partitioning Quality on Structured Graphs

Wexner Medical Center

When #procs = 65,536 and ppn=16
graph partitioning's edge cut is 22% more than geometric partitioning!



The Answer(s)

- Geometric mapping methods are useful on:
  - structured applications and/or network topologies
- Although connectivity-based models are more "accurate", they usually obtain lower quality mappings on structured graphs
  - who to blame? the model (graph), the method (partitioner/mapper) etc…
- Note: hypergraph model/methods achieve better results; but still there is room for improvements for "structured" problems [Uçar & Ç - PDP10]

The Answer(s): We need better tools!

- Better partitioner(s):
  - A hybrid ones that use both connectivity and geometry information
  - Better connectivity-based tools that will discover the structure of the problem
- Better mappers(s):
  - hybrid mapper that incorporates both graph and geometric partitioning.
  - Better mappers that will discover the structure of the network and application graph topologies.

Thanks

Wexner Medical Center

- For more information
  - Email umit@bmi.osu.edu
  - Visit http://go.osu.edu/umit or http://go.osu.edu/hpc

- Acknowledgement of Support