Improving Task Placement for Applications with 2D, 3D, and 4D Virtual Cartesian Topologies on 3D Torus Networks with Service Nodes

CUG 2013

Robert Fiedler & Stephen Whalen
Cray, Inc.

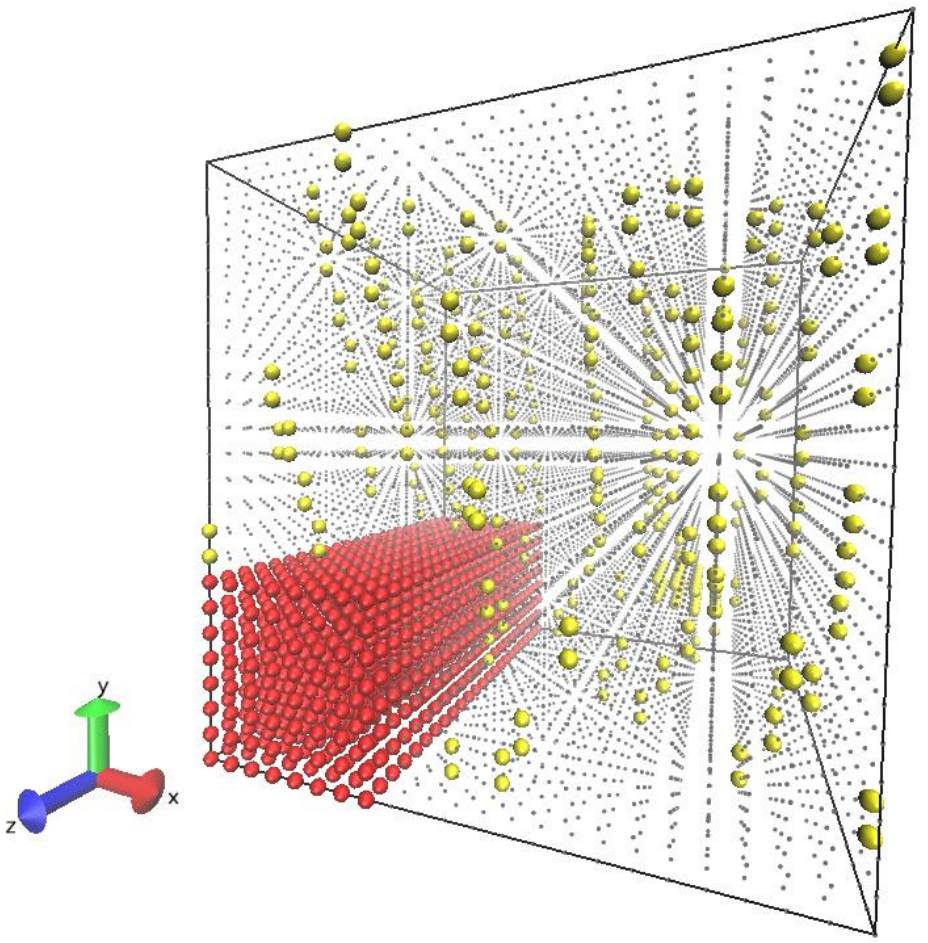
Outline

- **Background**
  - Cray XE/XK 3D torus
- **Sensitivity of applications to layout**
- **Placing neighboring tasks on each node**
  - Craypat
  - Grid_order
- **Placing groups of neighboring tasks onto nearby nodes**
  - Adaptive Layout – topology-aware version of MILC
  - Topaware – node selection & task placement tool
- **Results for applications with 4D, 3D, & 2D topologies**
- **Conclusions**

Background

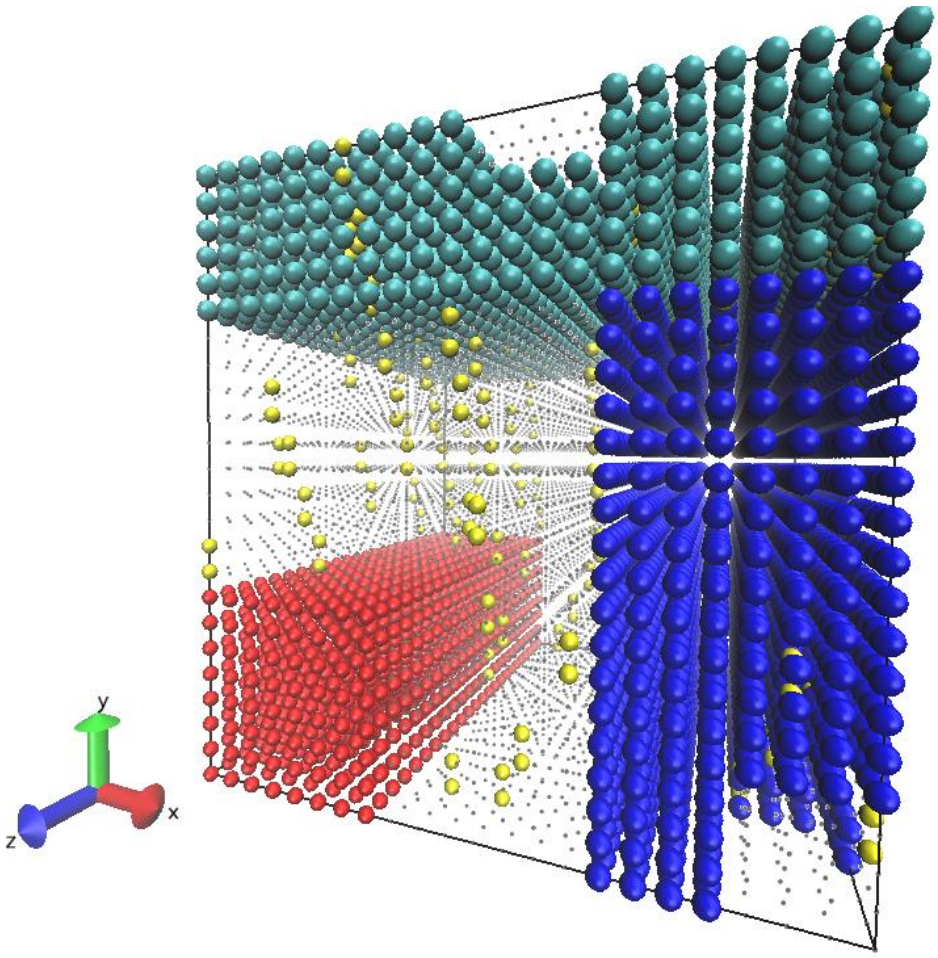
Blue Waters Interconnect

- Topology is 23x24x24 gemini routers
- 2 nodes per gemini, 2 geminis per blade
- 8x8x24 XK geminis (red)
- Service blades randomly distributed (yellow)
- Y-links between blades have 1/2 bandwidth of X- or Z-links
  - 2 nodes on same gemini don't use interconnect to exchange messages
- Routing algorithm is X, then Y, then Z



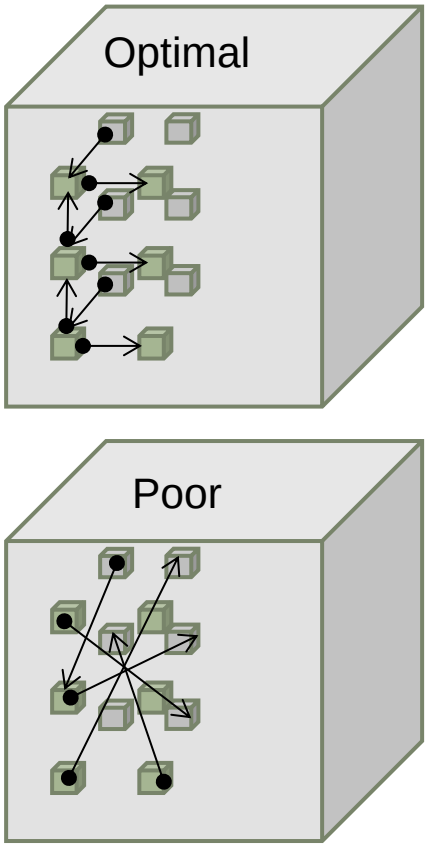
Background

- Routing takes shortest path
- If using > 1/2 of geminis in any dimension, traffic may wrap around the torus through geminis not assigned to job
- Jobs share interconnect for application communication, IO
- Run times affected by task placement, other running jobs



Task Placement and Interference

- **Applications that perform more communication are more sensitive to placement and interference**
  - Applications with All-to-All communication patterns compete more with other jobs
- **Applications with only nearest-neighbor communication in their virtual topology, if poorly placed, actually perform pairwise communication between randomly located nodes**
- **Even applications with All-to-All patterns can benefit significantly from topology-aware node selection**

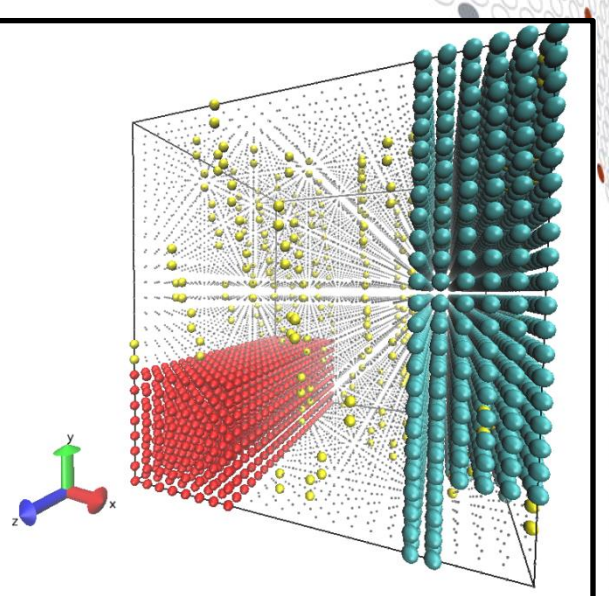


Example: PSDNS Turbulence Application

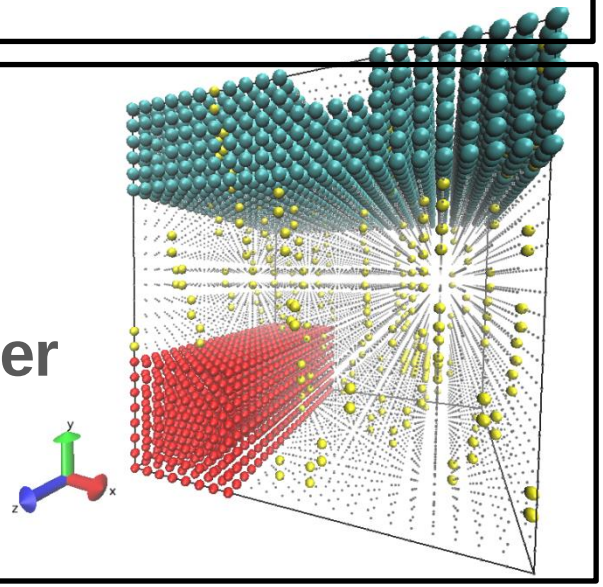
Pseudo-spectral method

3D FFTs → All-to-All

- **6k XE node job with 2 different shapes**
- **6x24x24 XE gemini region**
  - Ave max time per step: 35.3 s

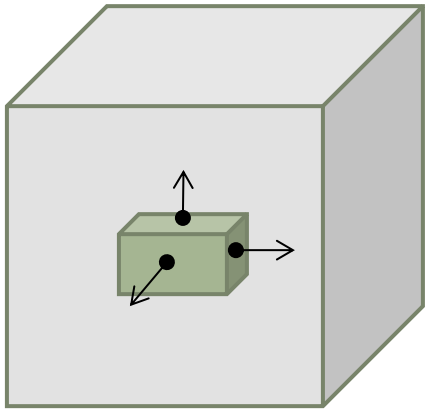


- **23x6x24 XE gemini region**
  - 2X more bisection bandwidth per node
  - Ave max time per step: 21.5 s
- **Job in slab normal to X takes 1.64X longer than job in slab normal to Y**



Choosing Tile Sizes

- **Consider applications that perform nearest-neighbor communication w/3D virtual Cartesian topology**
  - Assume same amount of communication in each direction per cell
- **Elongated tiles take advantage of faster links in x and z**
  - T_comm_x ~ X_face_area / X-link_bw
  - T_comm_y ~ Y_face_area / Y-link_bw
  - T_comm_z ~ Z_face_area / Z-link_bw
- **These three times are equal if**
  - X_face_area = Z_face_area = 2*Y_face_area
  - L_y = 2 * L_x
  - V = L^3 from cubic case ➔ L_x = L / 2^(1/3)
  - T_comm_x = 2^(1/3) T_comm_cubic_x
- **If communication is concurrent for all 3 directions**
  - T_comm = T_comm_cubic * 2^(1/3) / 2 = 0.63 * T_comm_cubic
- **If 3 directions done in sequence**
  - T_comm_seq = T_comm_cubic_seq * 2^(1/3) * (3/4)
    = T_comm_cubic_seq * 0.945



Virtual Topologies and Task Placement

- **Many applications define Cartesian grid virtual topologies**
  - MPI_CartCreate
  - Roll your own (i, j, …) virtual coordinates for each rank
- **Craypat rank placement**
  - Automatic generation of rank order based on detected grid topology
- **grid_order tool**
  - User specifies virtual topology to obtain rank order file
  - Node list by default is in whatever order ALPS/MOAB provide
- **These tools can be very helpful in reducing off-node communication, but they do not explicitly place neighboring groups of partitions in virtual topology onto neighboring nodes in torus**

Example: 4D Virtual Topology

MILC (lattice quantum chromodynamics)

- **4D Lattice, 84x84x84x144**
- **4116 nodes, 16 tasks per node, 65856 tasks**
- **6x6x6x6 lattice points per task**
- **Found best performance with**

grid_order –R -c 2,2,2,2 -g 14,14,14,24

- 1.9X speedup over SMP ordering!
- Difficult to map 4D virtual topology onto 3D torus using 2x2x2x2 blocks
- Possible to improve performance further by selecting which nodes to use

Selecting Nodes to Use

- **Very desirable to place tasks so that virtual neighbors are nearby on torus**
  - Difficult problem for arbitrary node lists
- **Sometimes practical to select which nodes to use in addition to placing tasks via rank order**
  - Dedicated system (or node pool)
  - Reservation with specified node list
- **Two approaches taken to quantify benefit**
  1. Adaptive Layout
     - Topology-aware decomposition scheme for MILC
     - Assumes compact, regular prism allocation
     - Allocation need not evenly divide lattice
  2. Topaware node selection and task placement tool
     - User specifies desired allocation shape
     - Generates node list and rank order for near-optimal layout

Adaptive Layout – Topology-Aware MILC

- **Find bounding box of node allocation**
  - Fit 4th lattice dimension entirely on each node
    - Map 3 space dimensions to 3D torus
  - Decompose lattice as if all enclosed nodes can be used
    - Some nodes outside allocation or service nodes
- **Some partitions assigned to unavailable nodes**
  - Relocate in torus along x, z
  - Place unassigned partitions on neighboring useable nodes
    - Split into 4 pieces before hand-off to neighbors (+/- x, +/- z)
  - Move excess partitions off busiest nodes onto neighbors
- **Results for 4116-node job in 23x4x24 allocation**
  - 2.7X faster than default placement
    - 1.42X faster than grid_order w/2x2x2x2
    - Some benefit from allocation shape (more bisection bandwidth)

Topaware: Node Selection and Task Placement

Purpose

- Given application w/2-, 3-, or 4-D grid communication graph
- Given particular input deck and decomposition
- Find near-optimal layout on given Cray XE/XK system
- Explore best possible performance and scaling

Limitations

- Presence of service nodes limits max node count
- Not all decompositions can be placed ideally
  - Number of usable nodes along each torus direction
  - Number of partitions per node
- Easier to get desired nodes on dedicated system

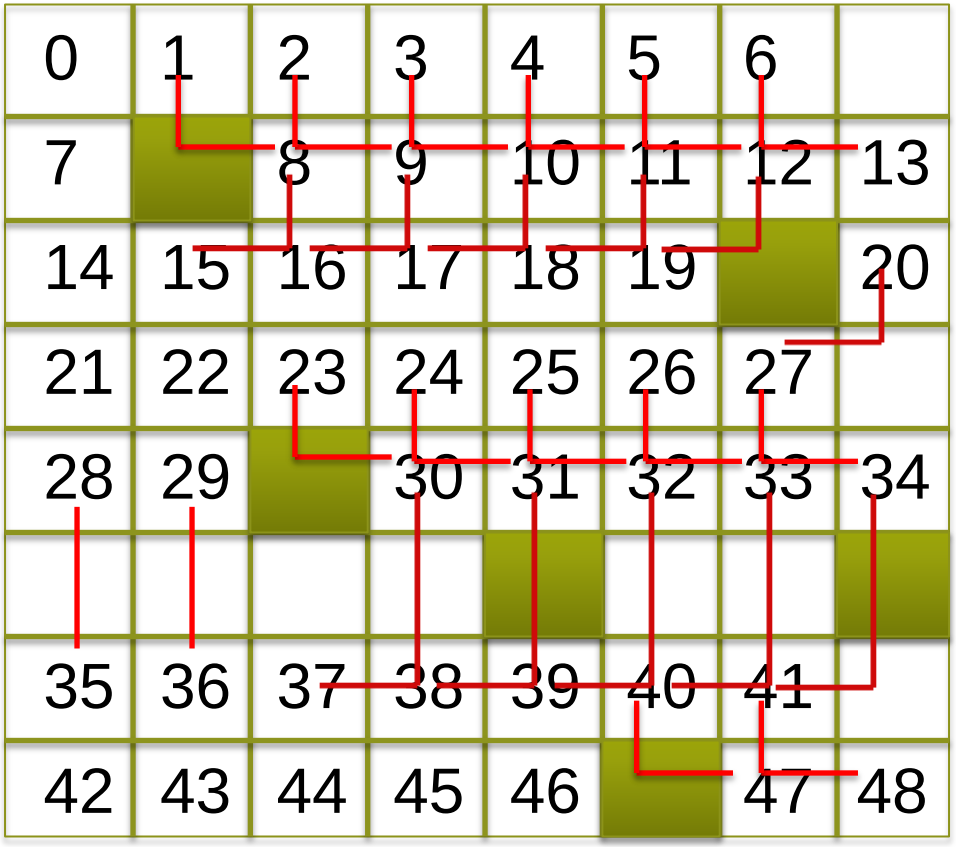
Topaware Node Selection Scheme

- One XZ plane shown
- Most rows and columns have 0 or 1 service node (green)
- Can fit up to a 7x7 gemini layout onto this 8x8 torus cross section
  - Selects 7 geminis in same rows they would have w/o service nodes
  - All selected geminis are also in same plane as w/o service nodes
- Scan in Y to find enough usable XZ planes

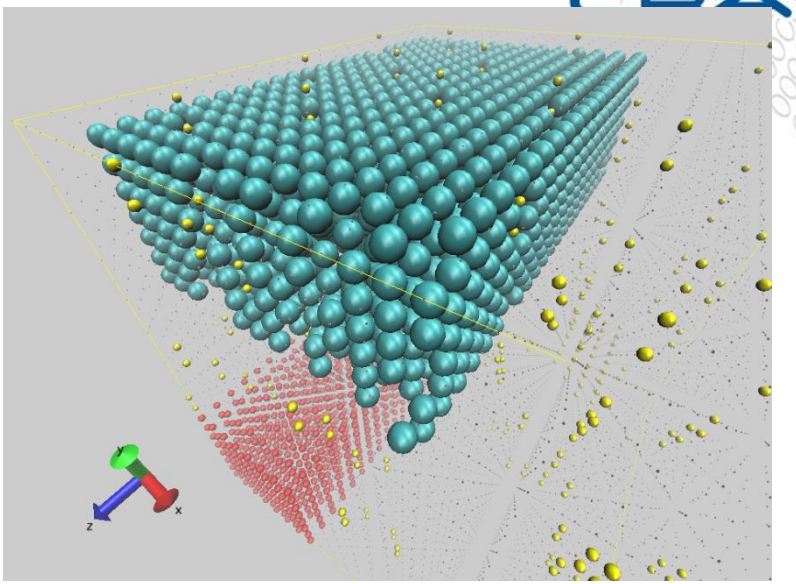
0	1	2	3	4	5	6	
7		8	9	10	11	12	13
14	15	16	17	18	19		20
21	22	23	24	25	26	27	
28	29		30	31	32	33	34
35	36	37	38	39	40	41	
42	43	44	45	46		47	48

Extra hops for North/South exchange

- Many hubs require second hop to reach some neighbors
- Density of multiple hops does not increase with scale, nor does # hops
- Should enable nearly ideal weak scaling, despite extra hops



Results on Blue Waters



MILC

- 4D Lattice, 84x84x84x144
- 4116 nodes, 16 tasks per node
- 6x6x4x9 lattice points per task
- Entire 4th dimension on each node pair
  - Remaining 3 dimensions mapped like any 3D virtual topology
- ~fiedler/bin/pick_nodes.csh **14 7 21** **1 2 1 16** **32** **0**
  - 14x7x21 geminis
  - 1x2x1x16 partitions per node pair
  - XE nodes only
  - Consider all “up” nodes, even if others using it
- 3.7X faster than default SMP placement
  - 1.9X faster than when using grid_order –c 2x2x2x2 …

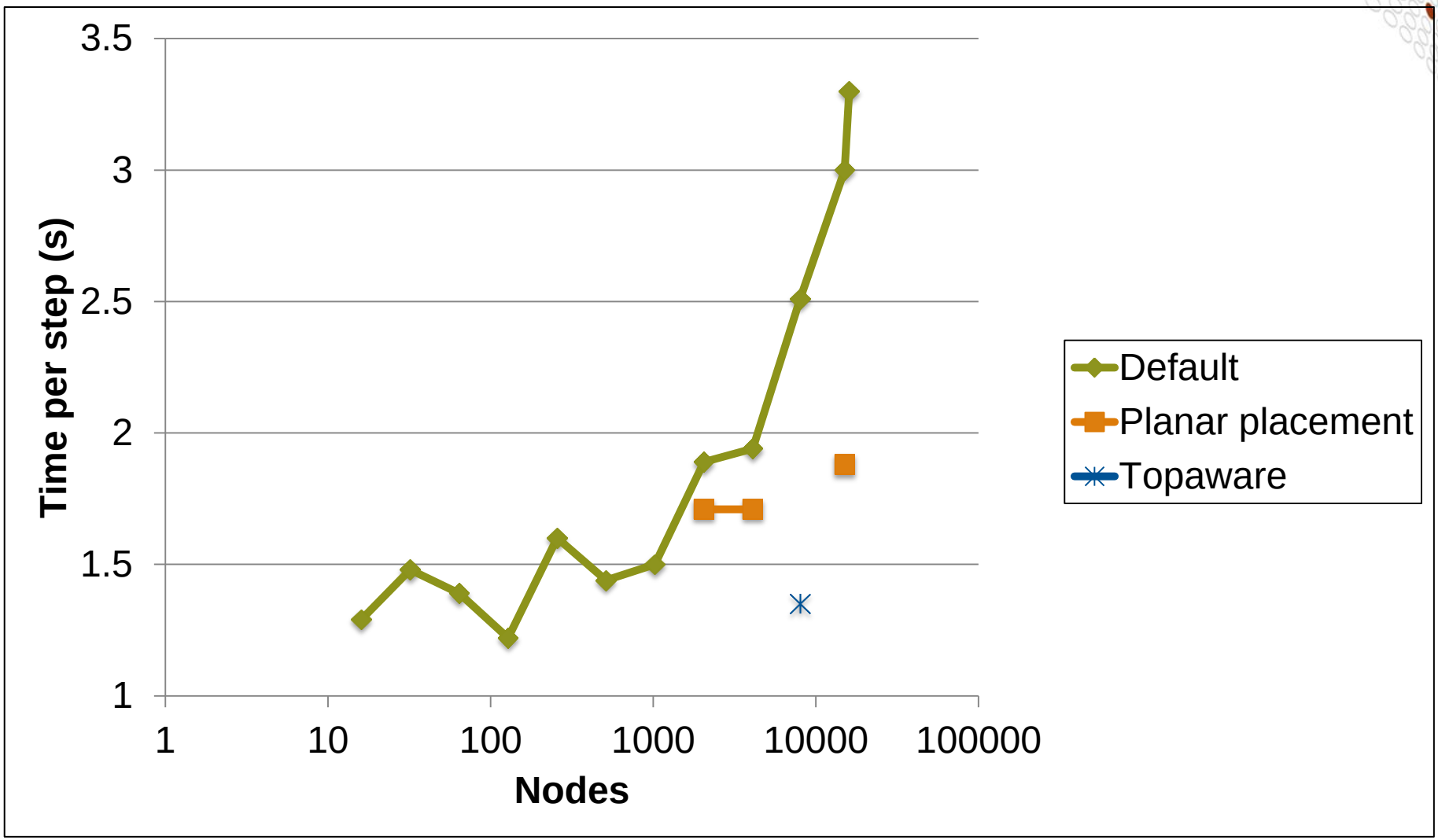
Results on Blue Waters for VPIC

- Plasma physics
- 3D virtual topology
- On 2k nodes, this code spends 8% of total run time on communication
- Ran on 4608 nodes in dedicated mode
- ~fiedler/pick_nodes.csh 12 12 16 4 4 2 1 32 0
  - 12x12x16 geminis
  - 4x4x2 partitions per node pair
  - Split first dimension between nodes
  - Use only nodes with this many cores
  - Consider all “up” nodes, even if others are using it
- Best results: 5% faster total run time than default placement

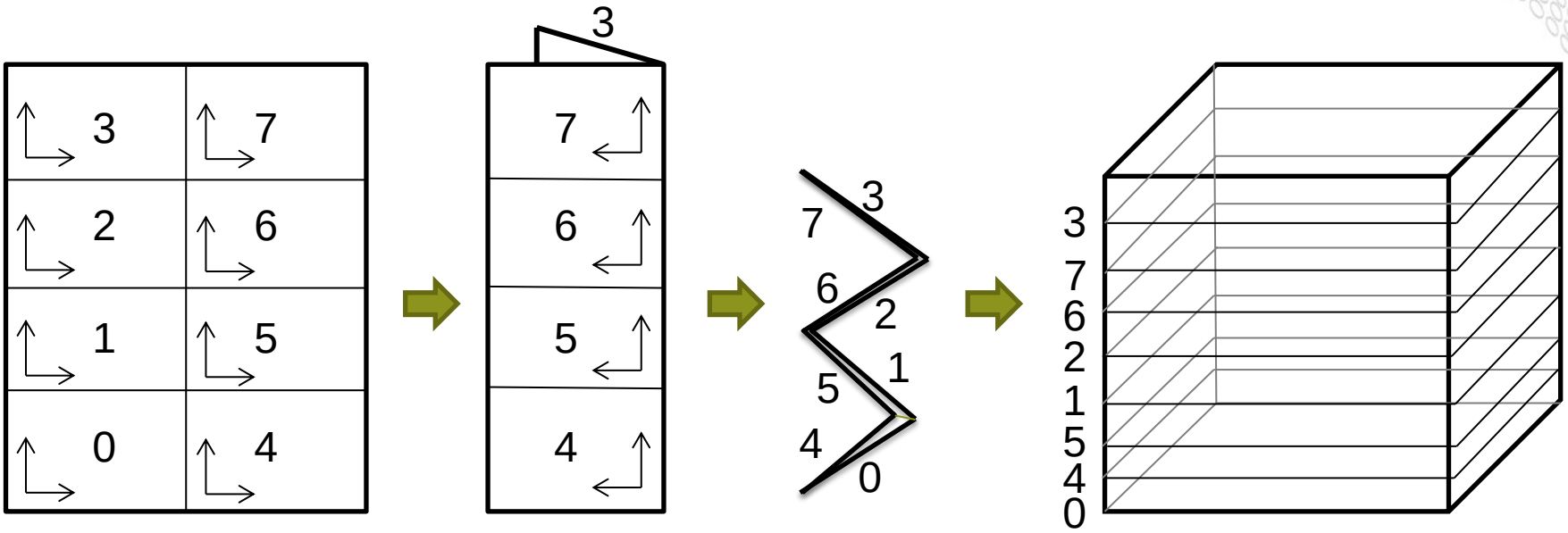
Results on Titan for S3D (R. Sankaran@ONRL)

- **Fluid dynamics w/ combustion**
- **3D Virtual topology**
- **Ran on up to ~12900 nodes in dedicated mode**
  - Near linear weak scaling (unlike default placement; see next slide)
- **Topaware placement → faster run times than default**
  - 2000 nodes: 1.32X
  - 6000 nodes: 1.61X

Results on Titan for S3D (R. Sankaran@ONRL)



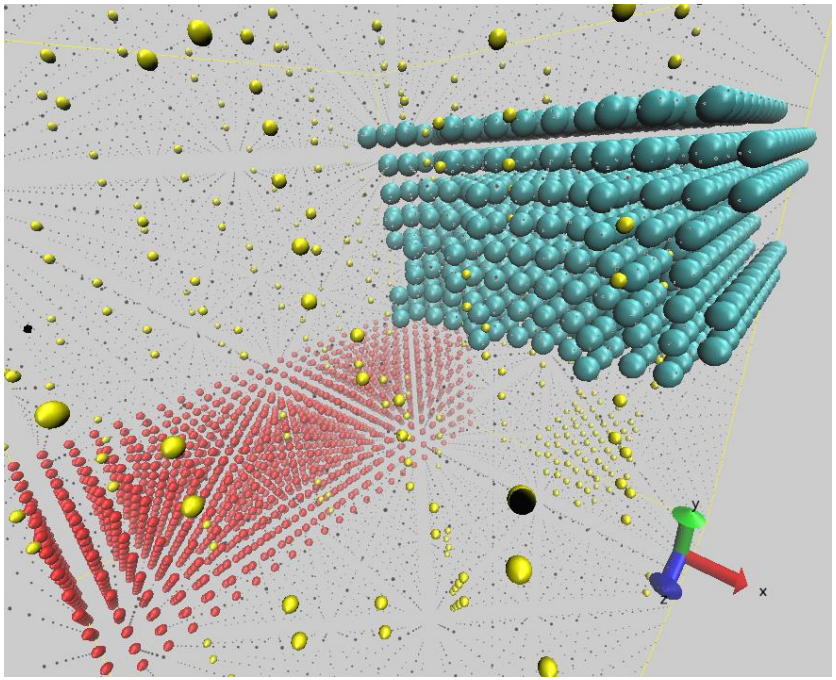
Mapping 2D Virtual Topology to 3D Torus

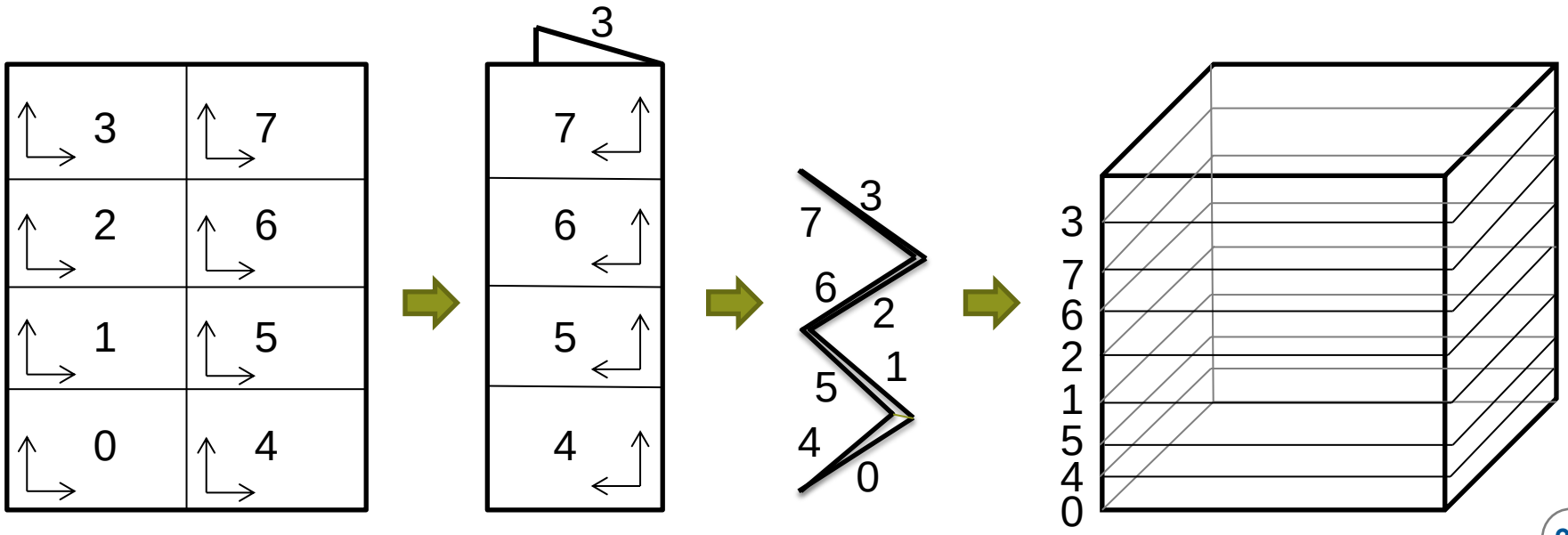


- **2D domain is folded like a sheet of paper into 8 supertiles**
  - Fold in half along one dimension, then 3 times in the other
  - No tearing – keeps neighbors close together
  - Communication between tiles is confined to super-tile edges
  - Folding in both dimensions overloads links shared by 4 supertiles
  - Optimal when folding along just one dimension
    - But results in long, thin tiles that increase “surface to volume” ratio

Staggered Supertiles

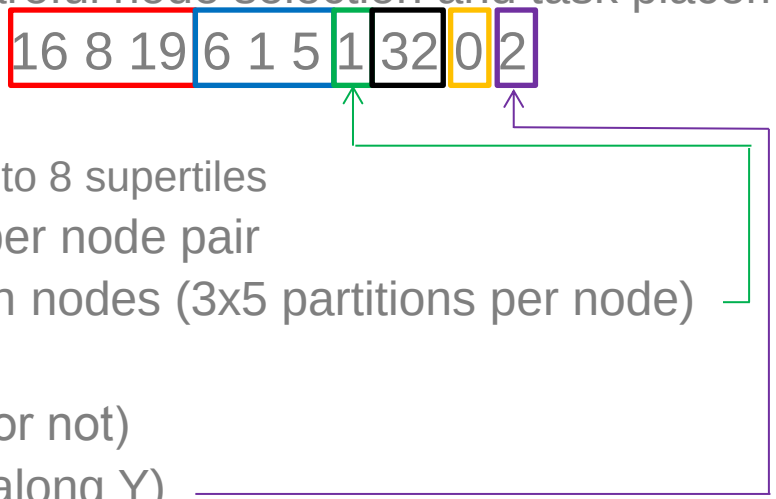
- **12x8x10 geminis**
  - 8 XZ planes
  - Stacked along Y
  - 4&5 and 6&7 staggered in X to avoid sharing links
  - Max hops = 4





Results on Blue Waters for WRF

- **Weather forecasting**
- **2D virtual topology**
- **Ran on 4864 nodes in dedicated mode**
- **Best results: 3% faster (GF/node) than grid_order placement on 4560 nodes**
  - Staggering not implemented at time of run
  - Significant benefit from core specialization
    - Greater than benefit from careful node selection and task placement
  - ~fiedler/bin/pick_nodes.csh 16 8 19 6 1 5 1 32 0 2
    - 16x8x19 geminis
      - 2D virtual topology folded into 8 supertiles
    - 6 in X and 5 in Z partitions per node pair
    - Split first dimension between nodes (3x5 partitions per node)
    - XE nodes only
    - Scan all “up” nodes (in use or not)
    - 2 supertiles along Z (and 4 along Y)



Concluding Remarks

Topaware

- **No application modifications required**
  - Set MPICH_RANK_REORDER_METHOD to 3
  - aprun –L`cat node_list` …
- **Recent enhancements**
  - Orient any virtual dimension along any torus dimension
- **Availability**
  - By request with limited support from R. Fiedler
- **Future enhancements**
  - Reduce nearest-neighbor communication path lengths for less regular node allocations from ALPS/torque/MOAB


CRAY
THE SUPERCOMPUTER COMPANY