Mapping Applications with Collectives over Sub-communicators on Torus Networks

Abhinav Bhatele, Todd Gamblin, Steven H. Langer, Peer-Timo Bremer, Erik W. Draeger, Bernd Hamann, Katherine E. Isaacs, Aaditya G. Landge, Joshua A. Levine, Valerio Pascucci, Martin Schulz, Charles H. Still

SC '12 ◆ November 15, 2012

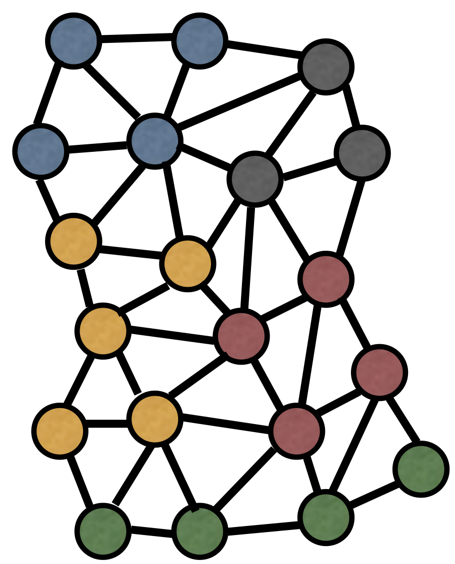
LLNL-PRES-605452

SC12
Salt Lake City, Utah

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

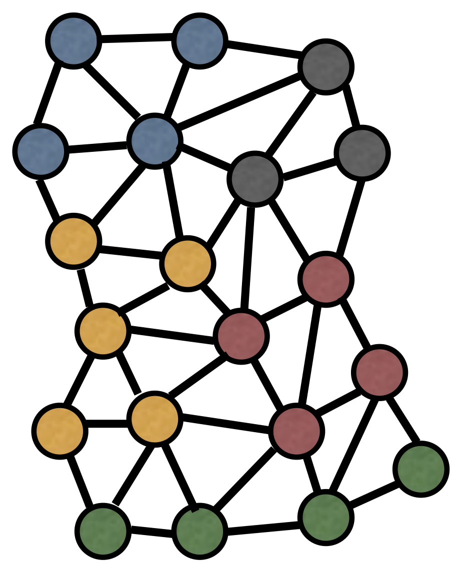
Topology aware task mapping

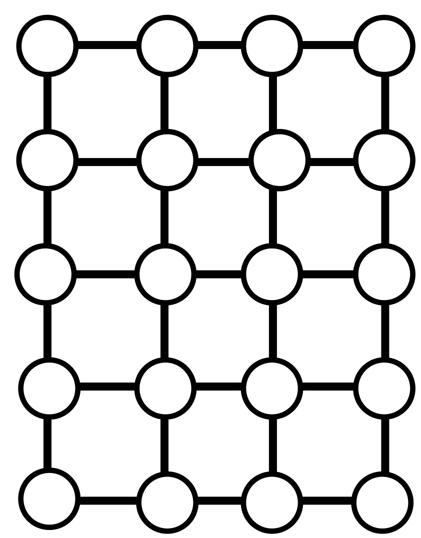
- What is mapping - layout/placement of tasks/processes in an application on the physical interconnect
- Does not require any changes to the application



Topology aware task mapping

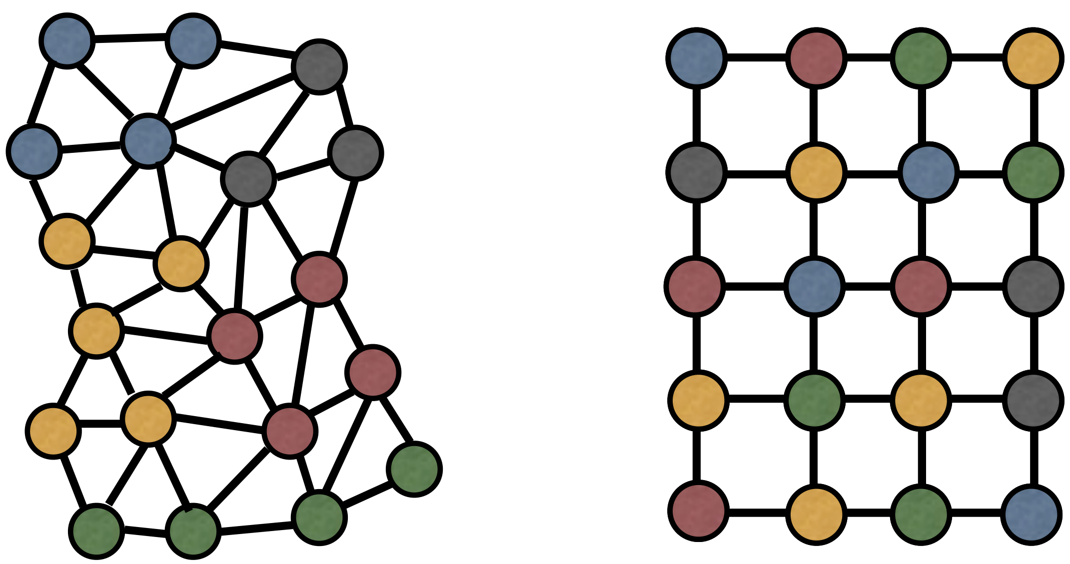
- What is mapping - layout/placement of tasks/processes in an application on the physical interconnect
- Does not require any changes to the application





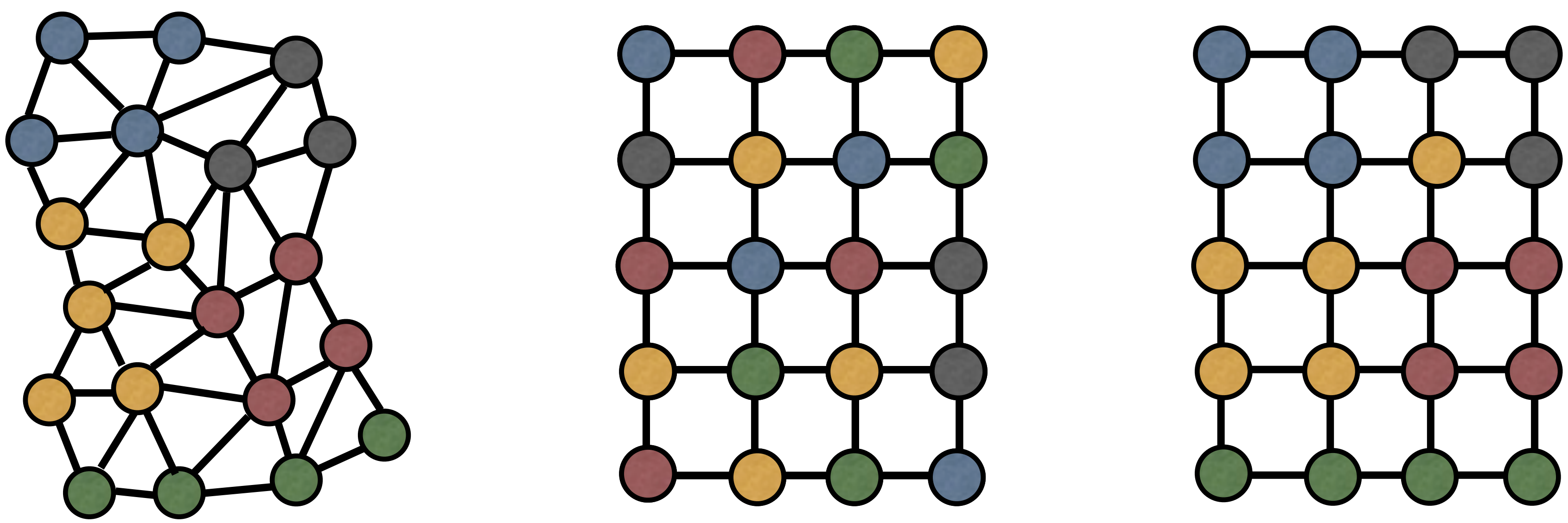
Topology aware task mapping

- What is mapping - layout/placement of tasks/processes in an application on the physical interconnect
- Does not require any changes to the application



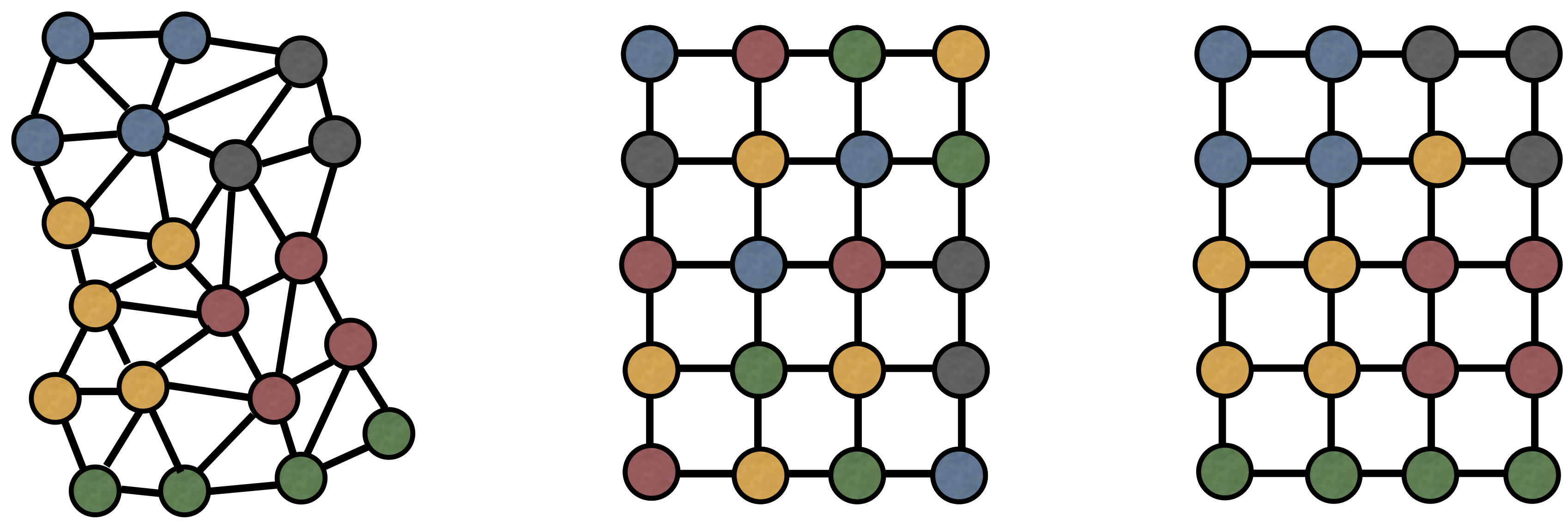
Topology aware task mapping

- What is mapping - layout/placement of tasks/processes in an application on the physical interconnect
- Does not require any changes to the application



Topology aware task mapping

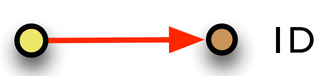
- What is mapping - layout/placement of tasks/processes in an application on the physical interconnect
- Does not require any changes to the application



- Goals:
  - Balance computational load
  - Minimize contention (optimize latency or bandwidth)

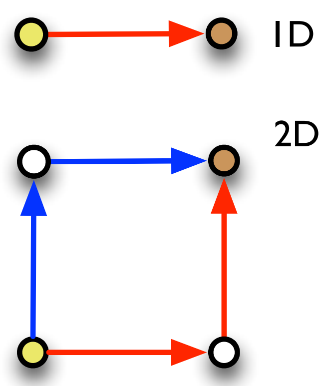
Maximize bandwidth?

- Traditionally, research has focused on bringing tasks closer to reduce the number of hops
  - Minimizes latency, but more importantly link contention
- For applications that send large messages this might not be optimal



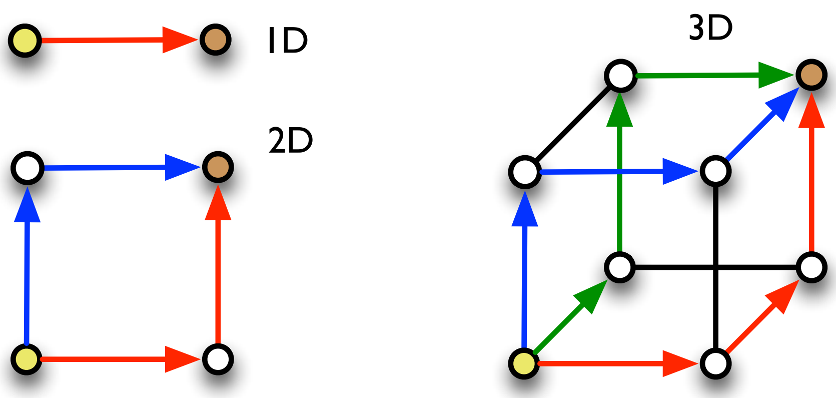
Maximize bandwidth?

- Traditionally, research has focused on bringing tasks closer to reduce the number of hops
  - Minimizes latency, but more importantly link contention
- For applications that send large messages this might not be optimal



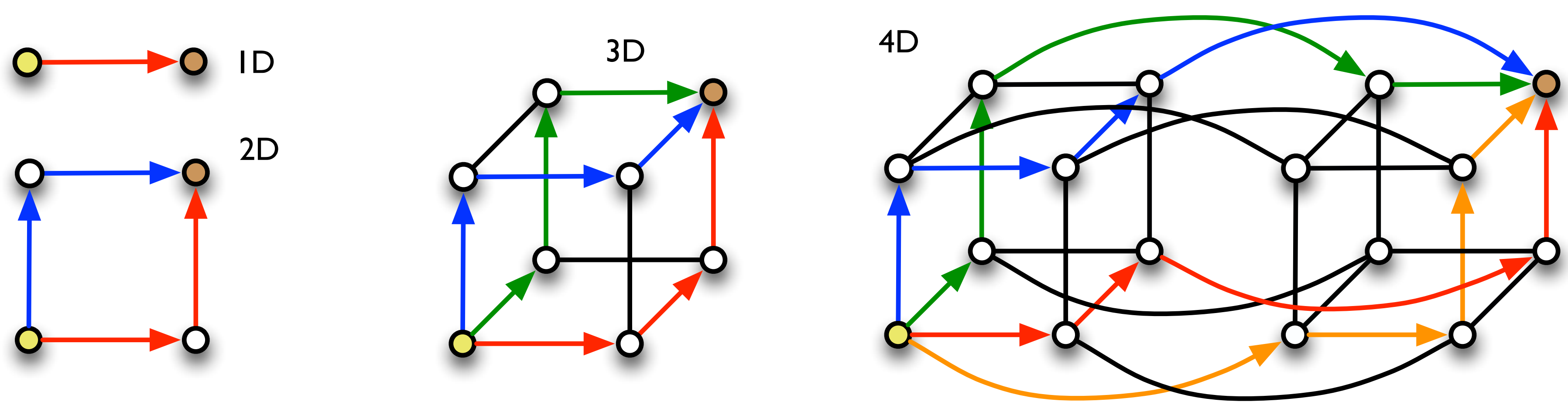
Maximize bandwidth?

- Traditionally, research has focused on bringing tasks closer to reduce the number of hops
  - Minimizes latency, but more importantly link contention
- For applications that send large messages this might not be optimal

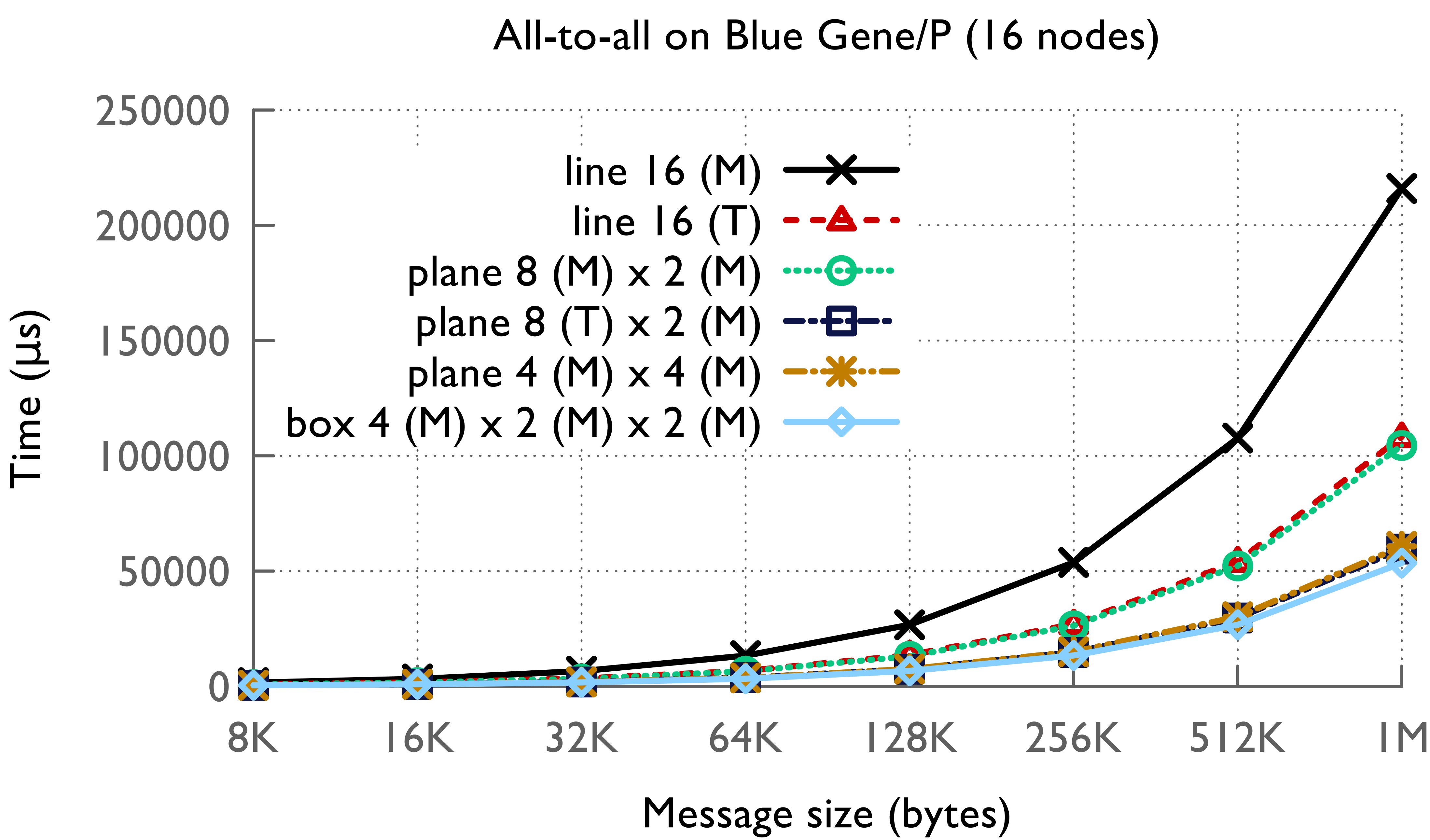


Maximize bandwidth?

- Traditionally, research has focused on bringing tasks closer to reduce the number of hops
  - Minimizes latency, but more importantly link contention
- For applications that send large messages this might not be optimal



Collectives

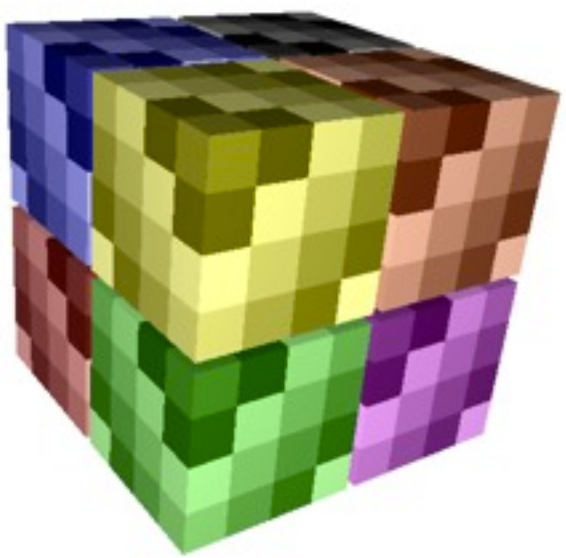


Improving bandwidth utilization

- Placing communicating pairs farther apart in multiple dimensions to increase "spare links"
- Placing processes on a cube or plane instead of a line (collectives)
- Use wraparound links for additional routes

Rubik

- We have developed a mapping tool focusing on:
  - structured applications that are bandwidth-bound, use collectives over sub-communicators
  - built-in operations that can increase effective bandwidth on torus networks based on heuristics
- Input:
  - Application topology with subsets identified
  - Processor topology
  - Set of operations to perform
- Output: map file for job launcher



Idea of partition trees

- Recursive partitioning of n-D cartesian spaces
  - n can be 2, 3, 4, 5 or any other number
- Intermediate nodes in the tree represent
  - closely communicating groups in application space, or
  - sub-domains of processors in network space
- Leaf nodes represent processes in the application or nodes on the network

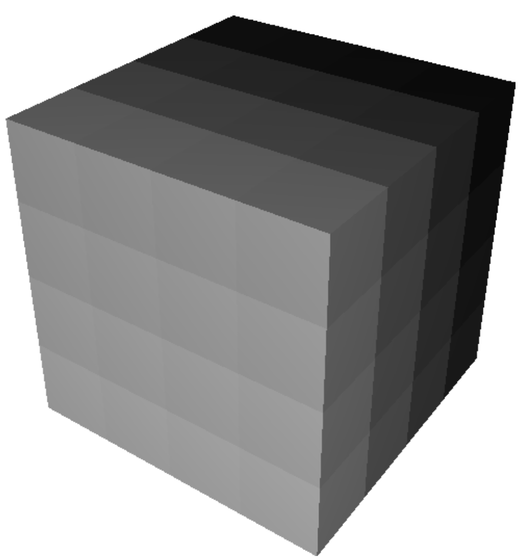
Creating a partition tree

Creating a partition tree

```
domain = box([4, 4, 4])
```

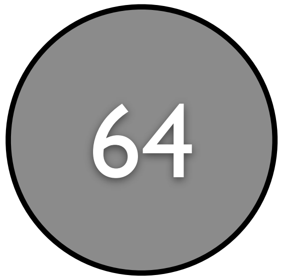
Creating a partition tree

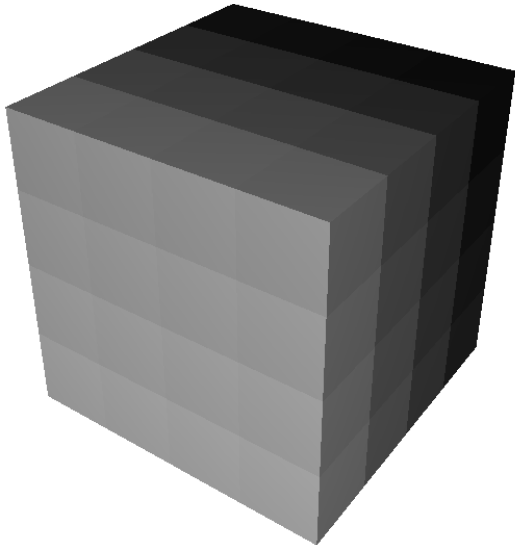
```
domain = box([4, 4, 4])
```



Creating a partition tree

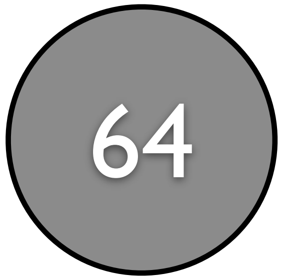
```
domain = box([4, 4, 4])
```

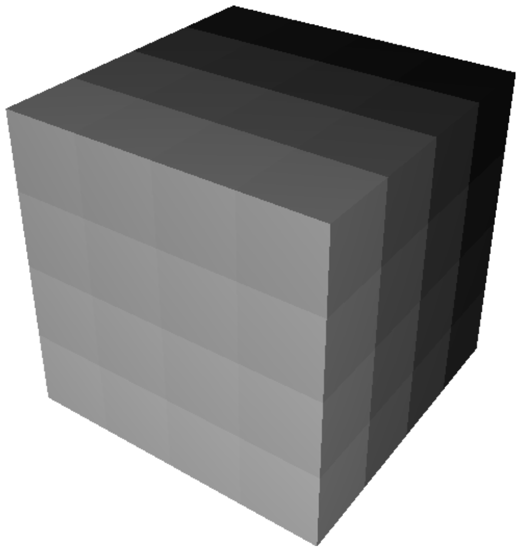




Creating a partition tree

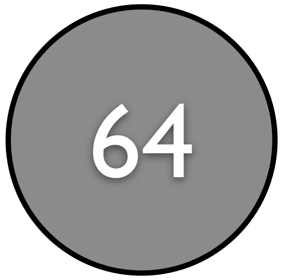
```
domain = box([4, 4, 4])

domain.tile([4, 4, 2])
```

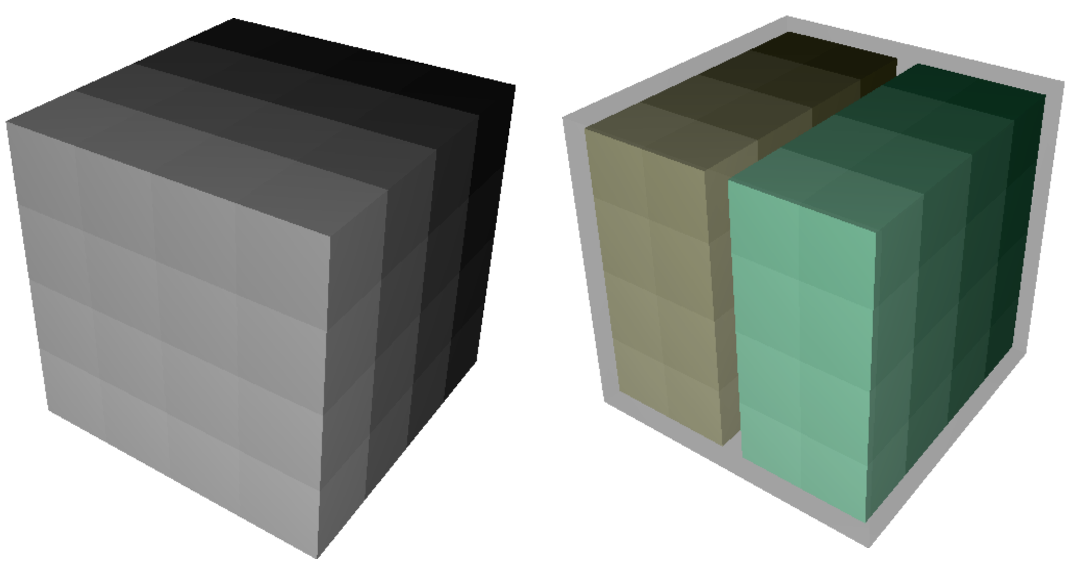




Creating a partition tree

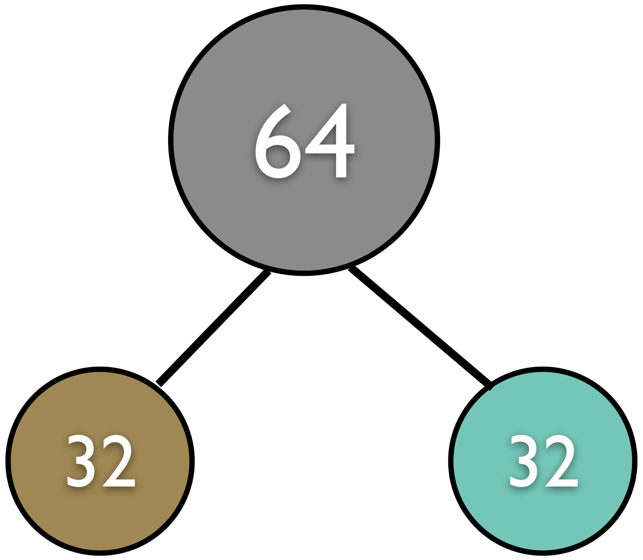
```
domain = box([4, 4, 4])

domain.tile([4, 4, 2])
```

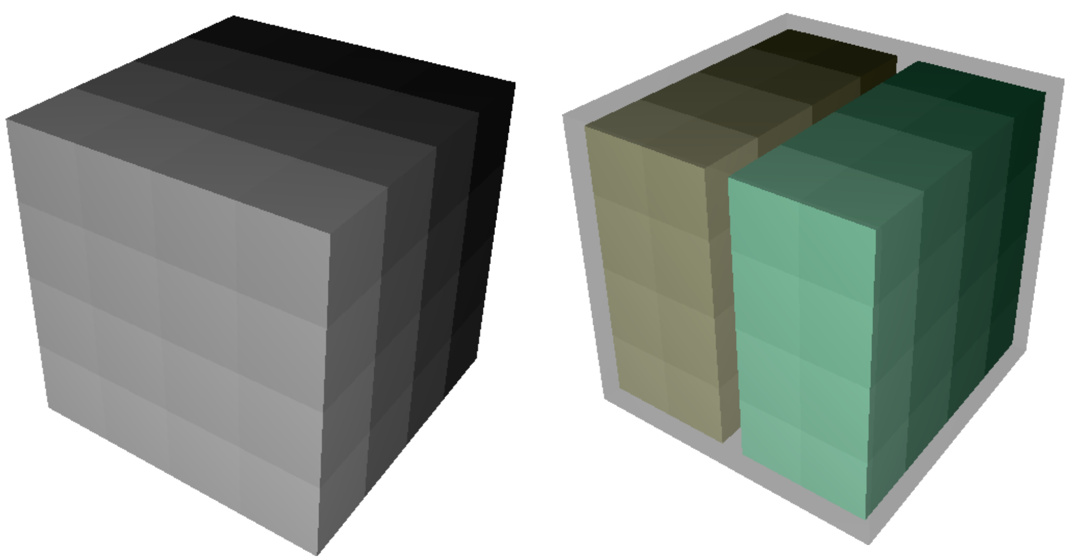




Creating a partition tree

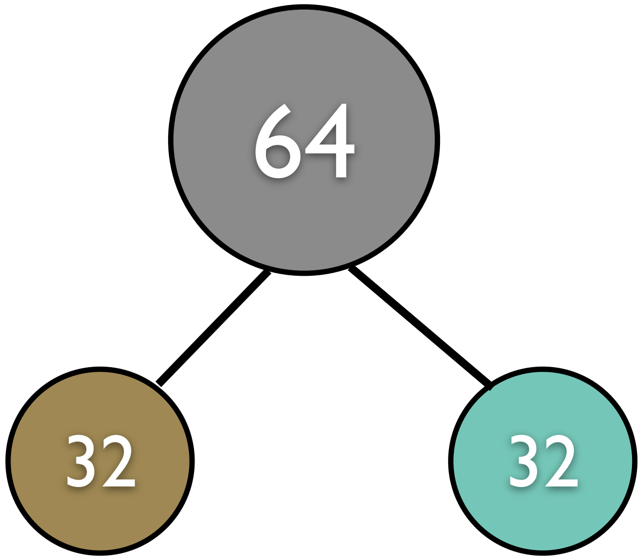
```
domain = box([4, 4, 4])
domain.tile([4, 4, 2])
```

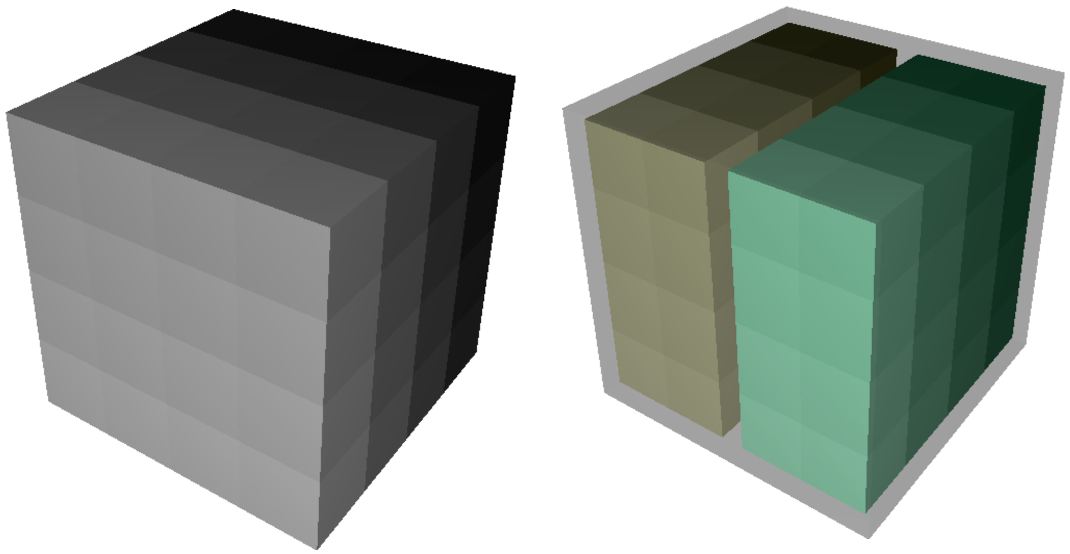




Creating a partition tree

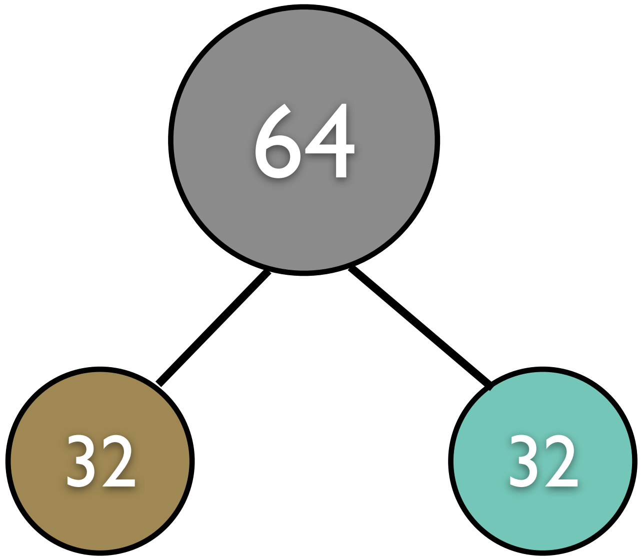
```
domain = box([4, 4, 4])
domain.tile([4, 4, 2])
for child in domain:
    child.tile([2, 4, 1])
```

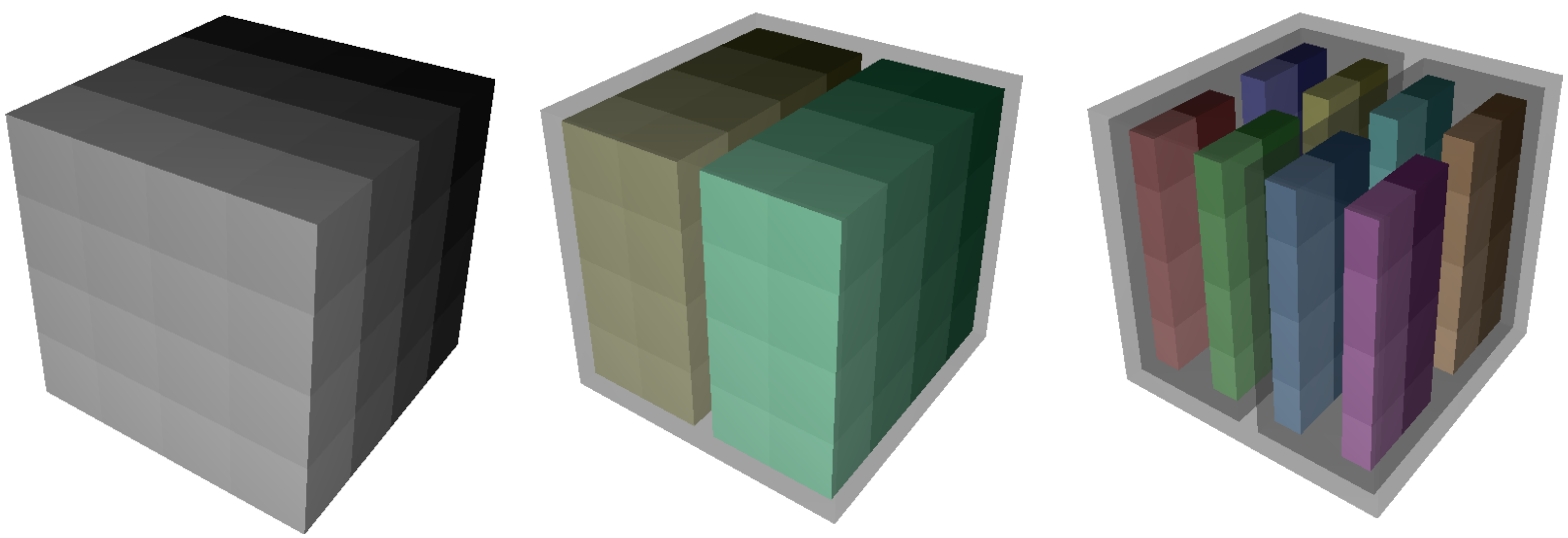




Creating a partition tree

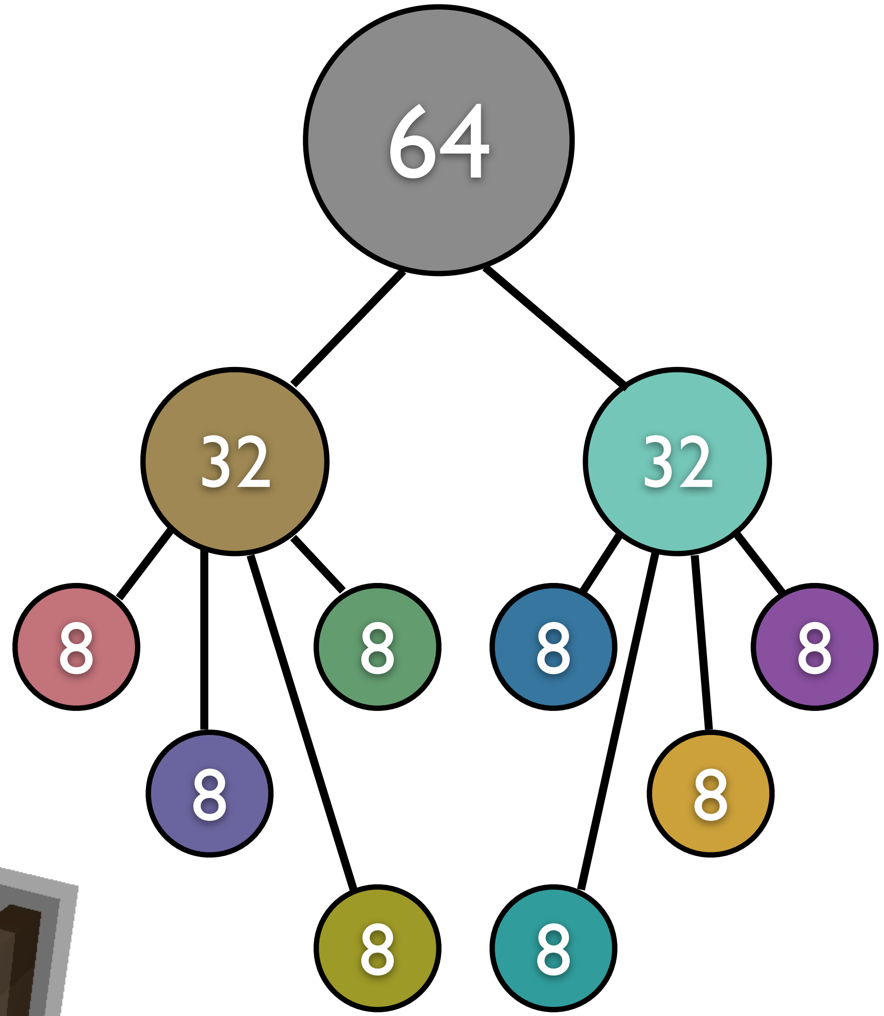
```
domain = box([4, 4, 4])
domain.tile([4, 4, 2])
for child in domain:
    child.tile([2, 4, 1])
```

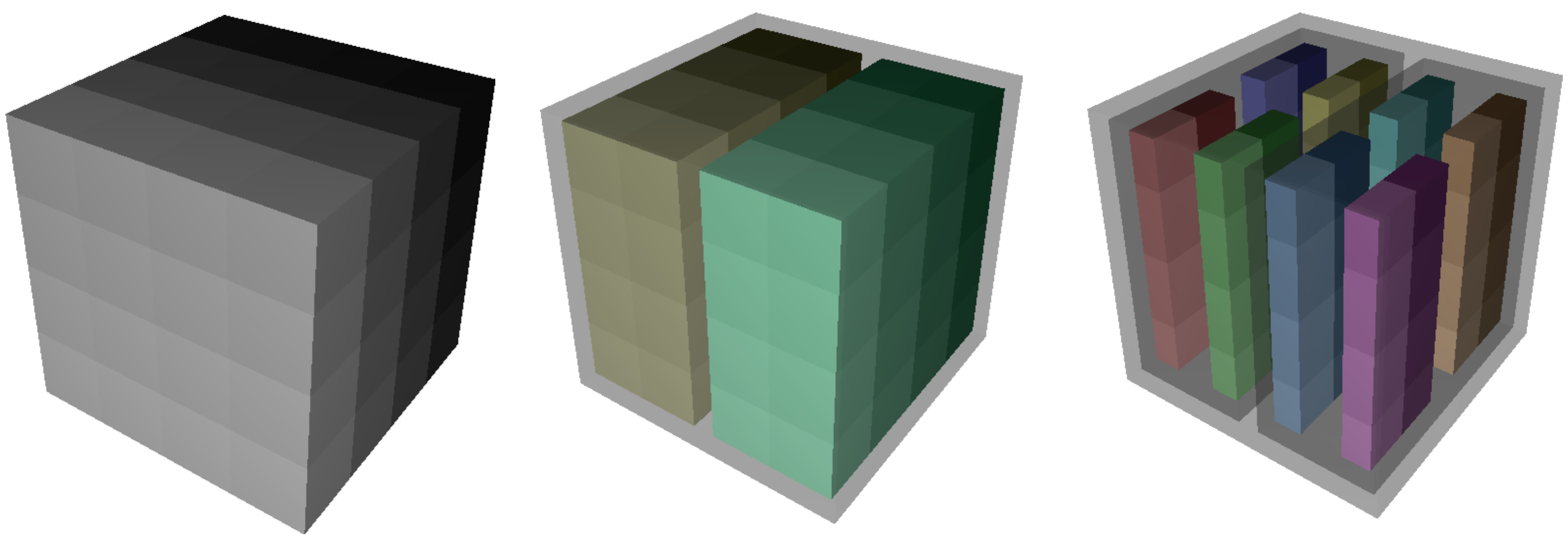




Creating a partition tree

```
domain = box([4, 4, 4])
domain.tile([4, 4, 2])
for child in domain:
    child.tile([2, 4, 1])
```

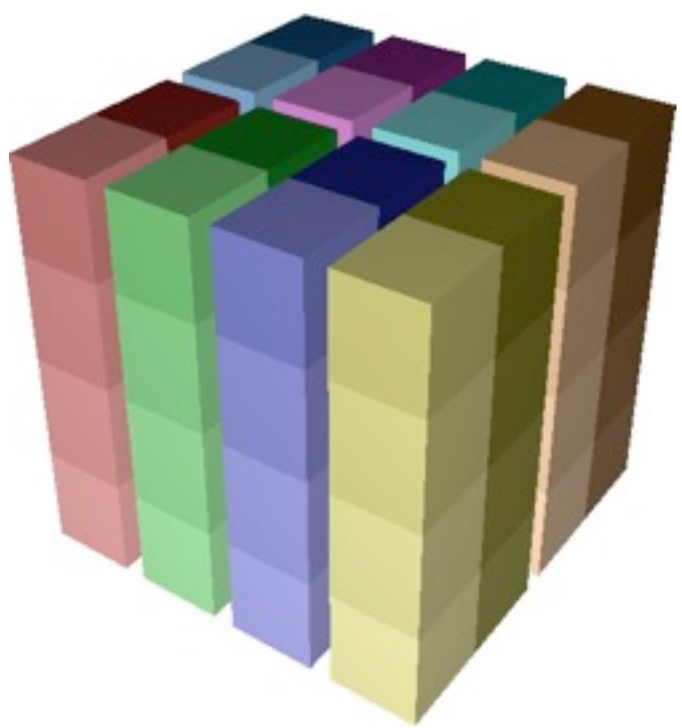




Monday, February 11, 13

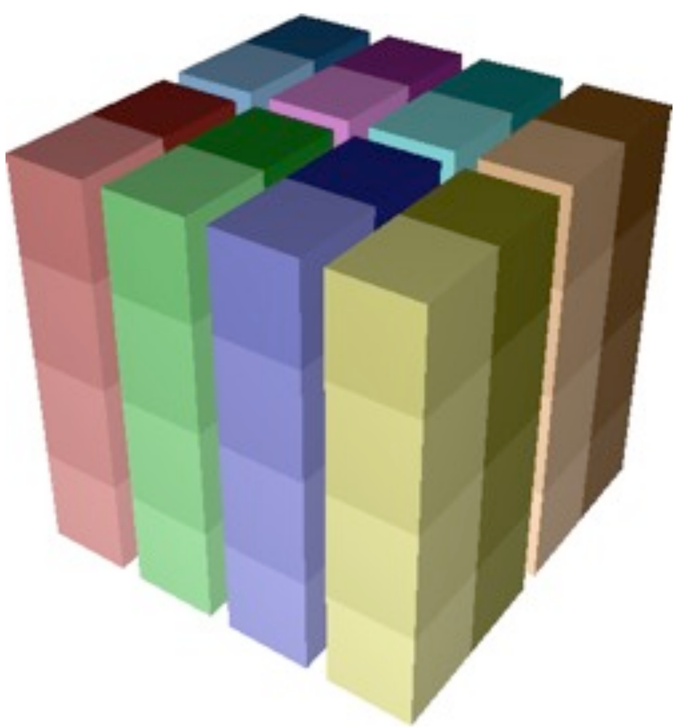
Partitioning operations

```
app = box([4, 4, 4])
app.div([2, 1, 4])
```

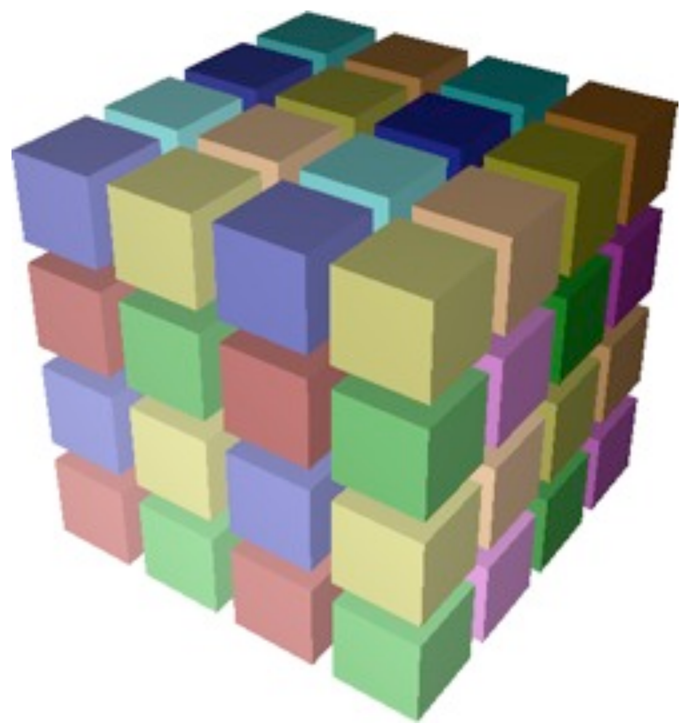


Partitioning operations

```
app = box([4, 4, 4])
app.div([2, 1, 4])
```

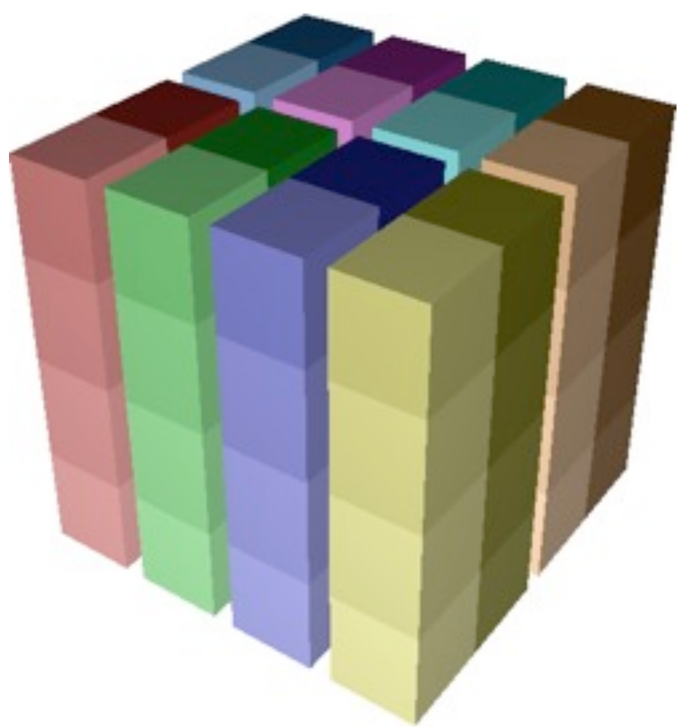


```
app = box([4, 4, 4])
app.mod([2, 2, 2])
```

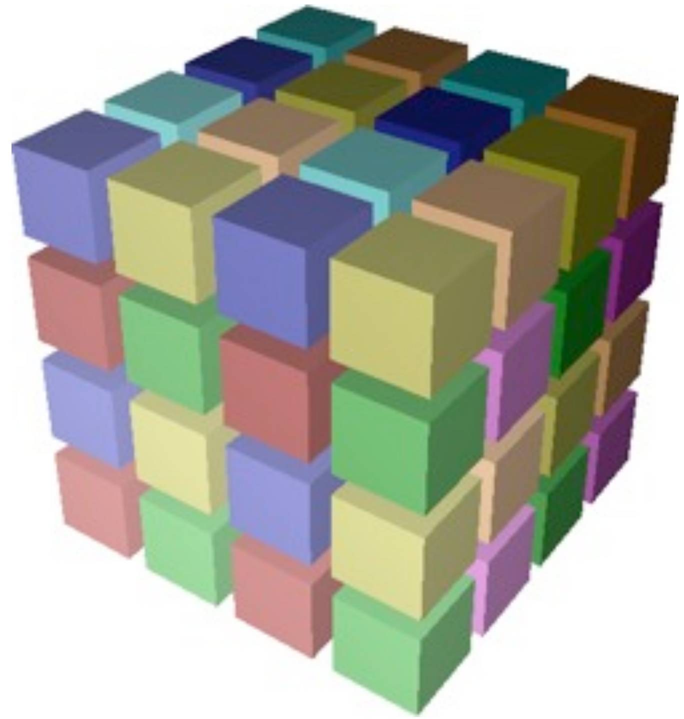


Partitioning operations

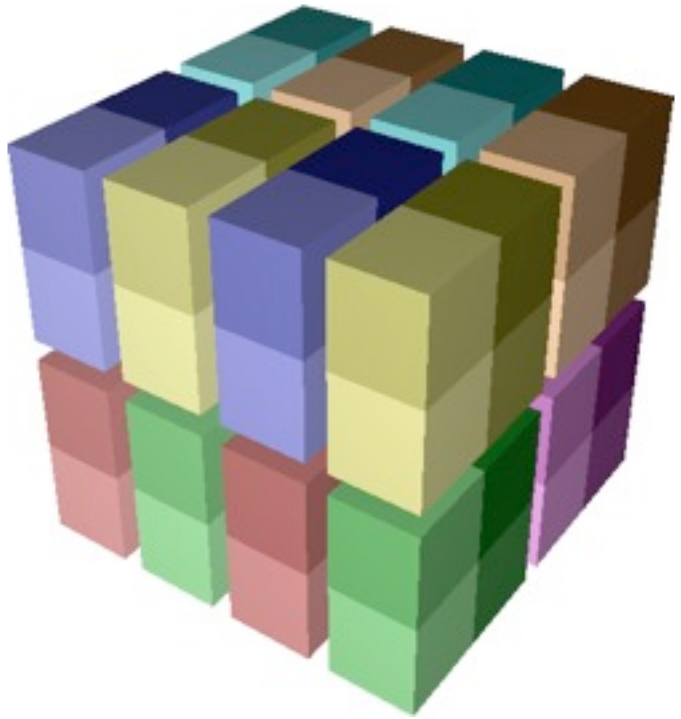
```
app = box([4, 4, 4])
app.div([2, 1, 4])
```



```
app = box([4, 4, 4])
app.mod([2, 2, 2])
```



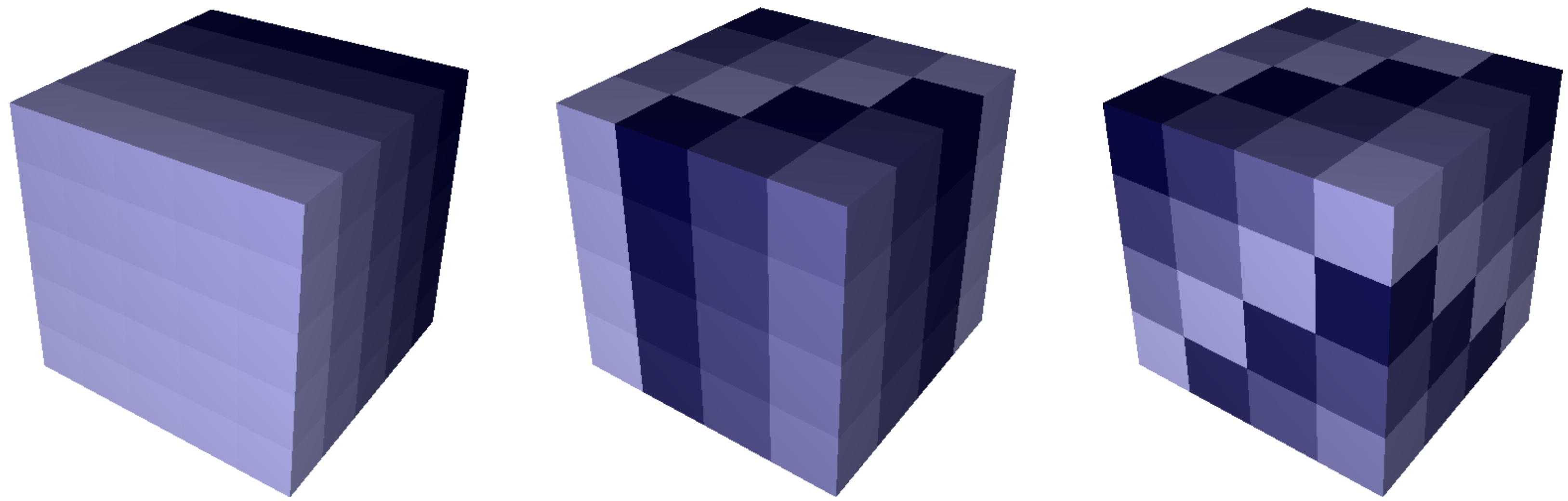
```
app = box([4, 4, 4])
app.cut([2, 2, 2],
    [div, div, mod])
```



Permuting operations

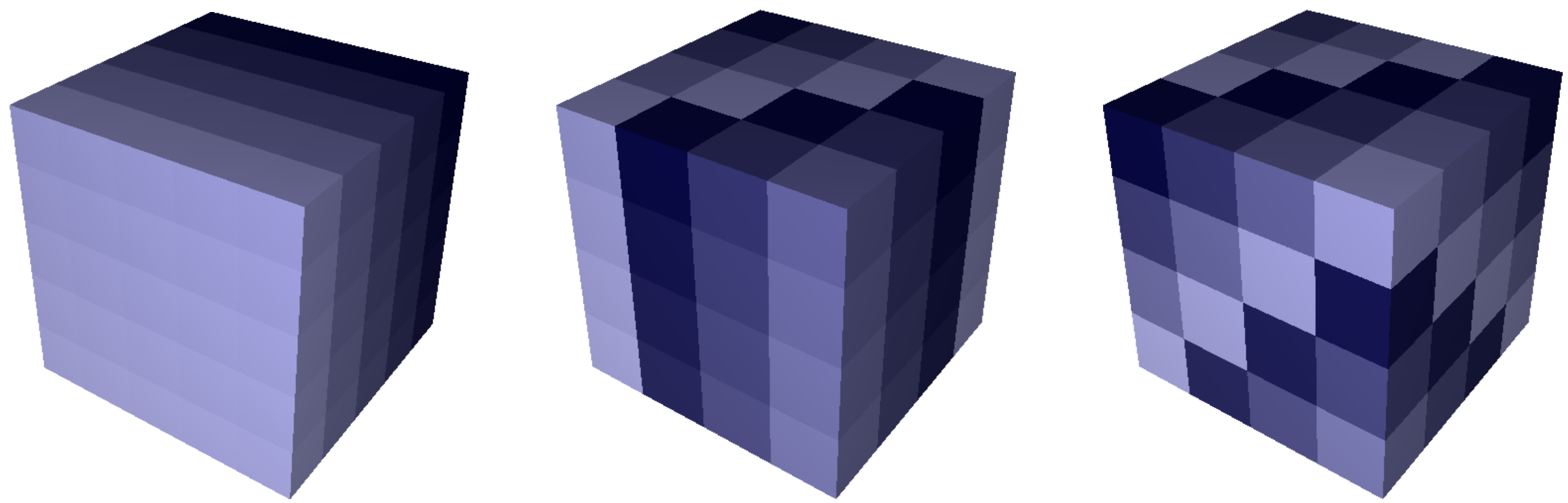
Permuting operations

- Tilt

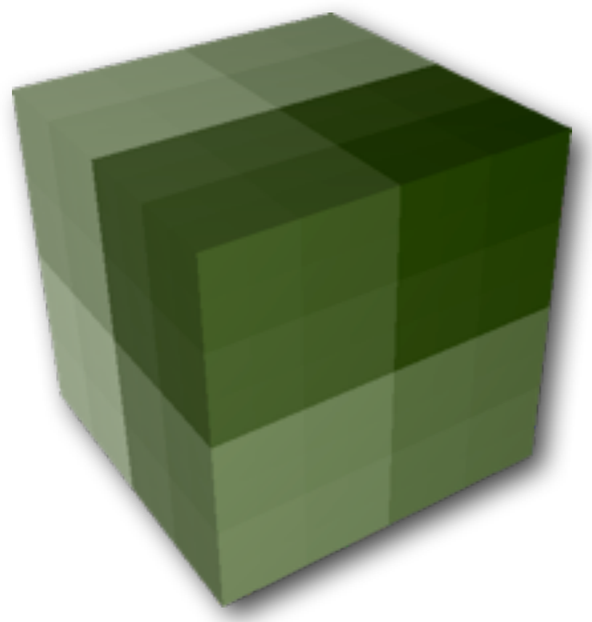


Permuting operations

- Tilt

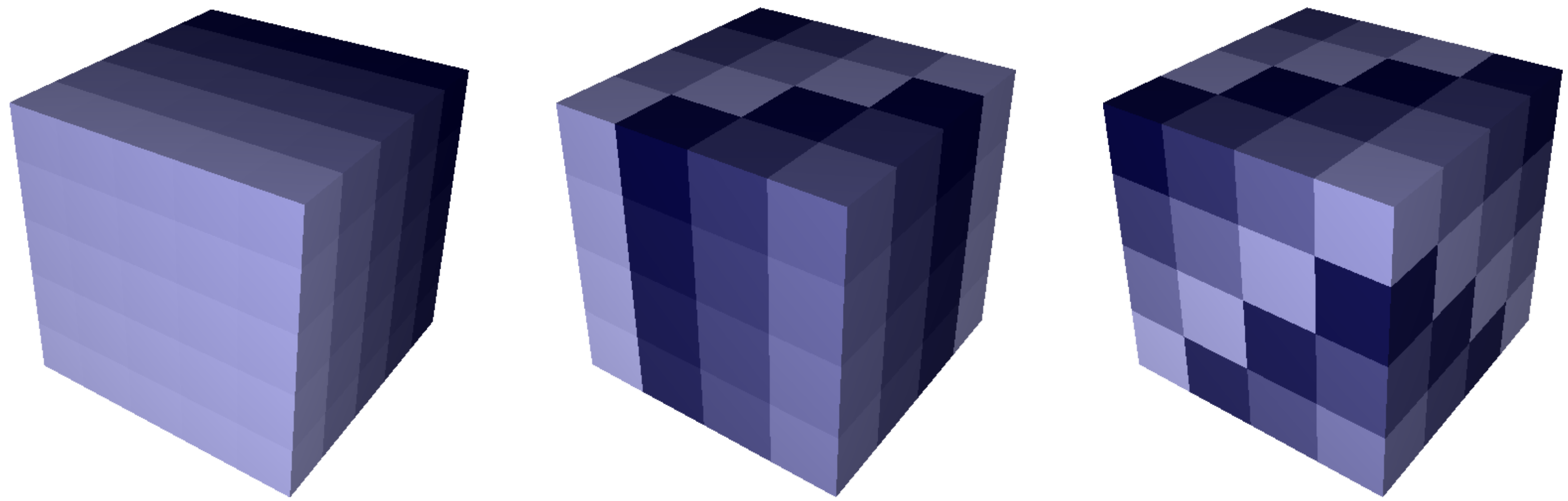


- Zorder

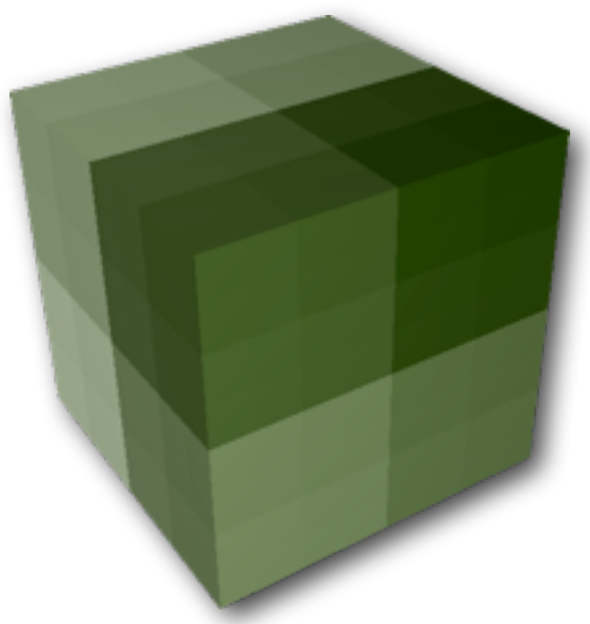


Permuting operations

- Tilt



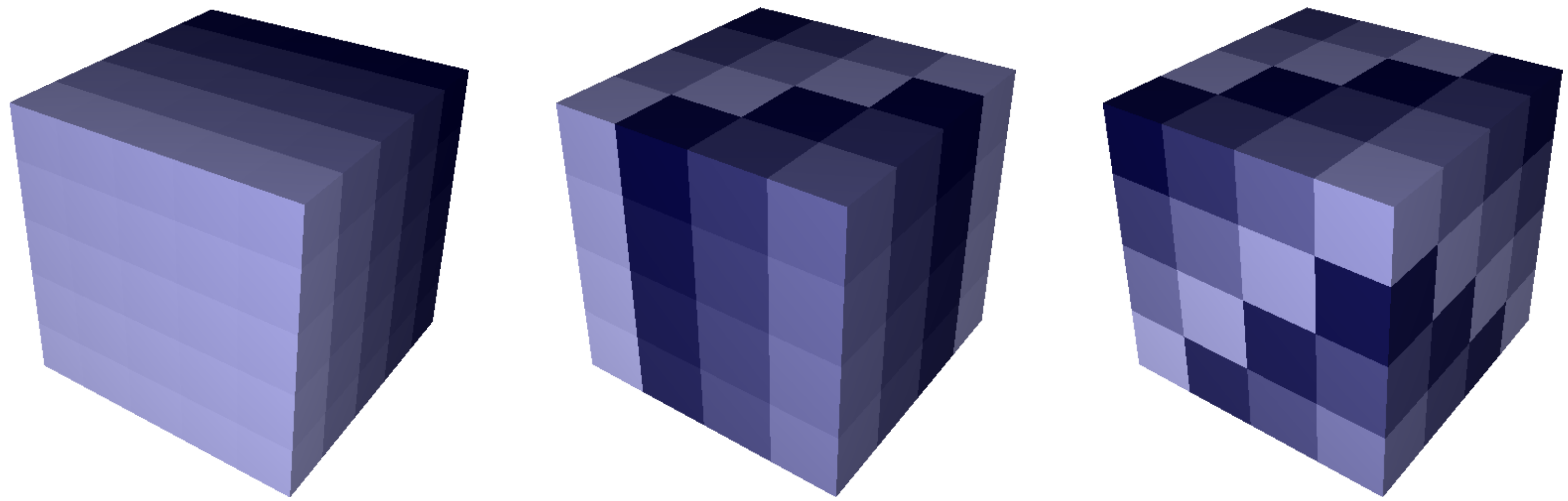
- Zorder



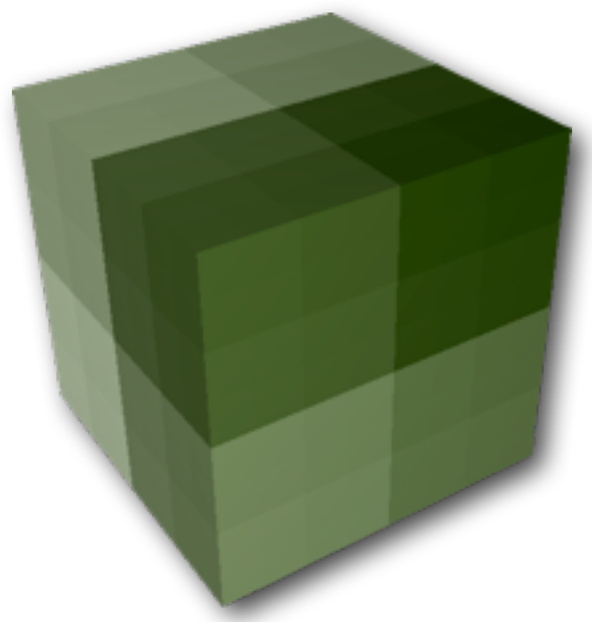
- Zigzag

Permuting operations

- Tilt



- Zorder

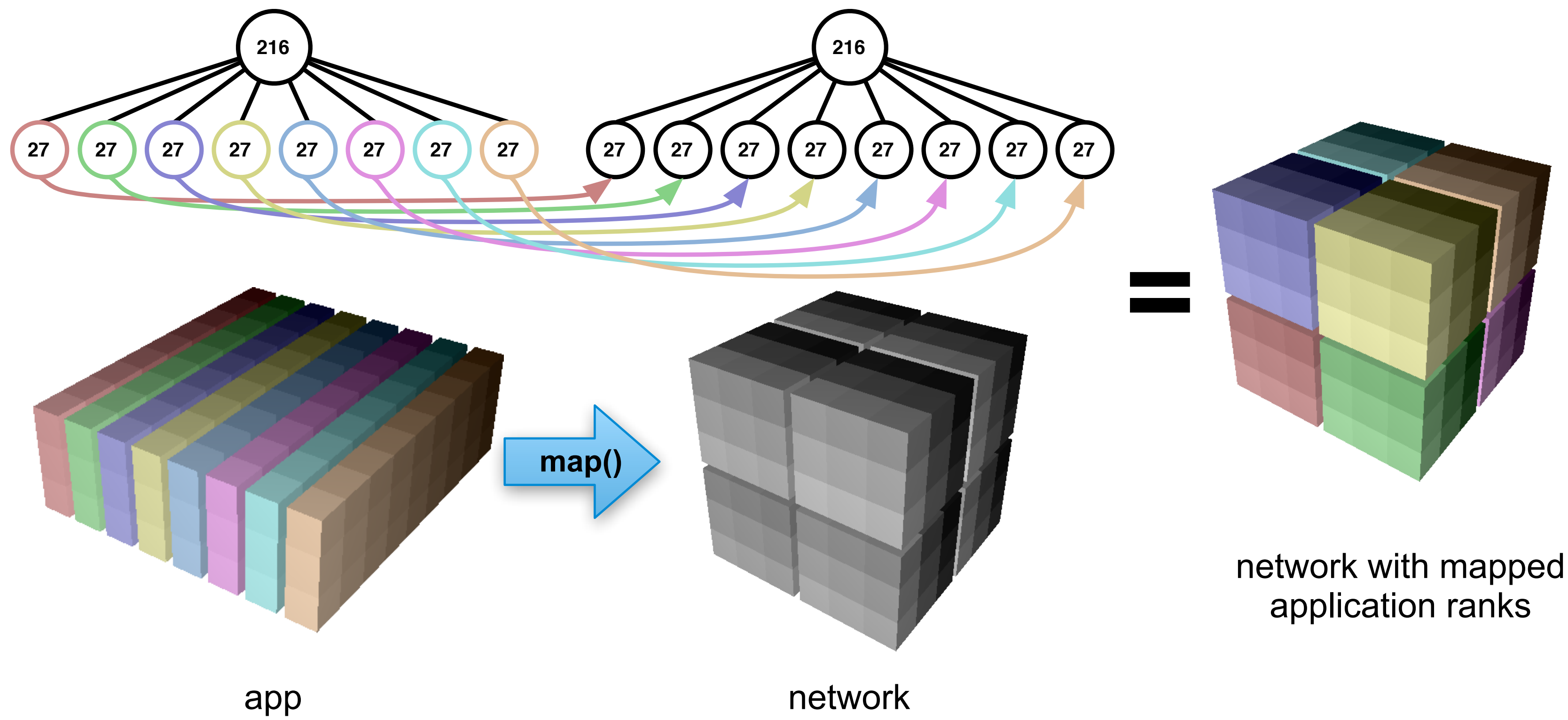


- Zigzag

- Hierarchical Operations

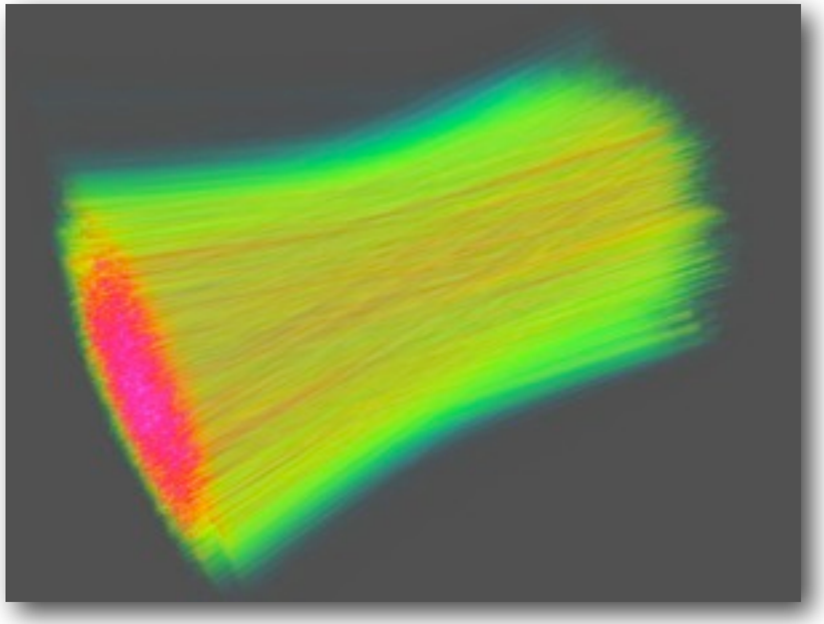
Application example

```
app = box([9,3,8]) # Create app partition tree of 27-task planes
app.tile([9,3,1])

network = box([6,6,6]) # Create network partition tree of 27-processor cubes
network.tile([3,3,3])

network.map(app)  # Map task planes into cubes
```



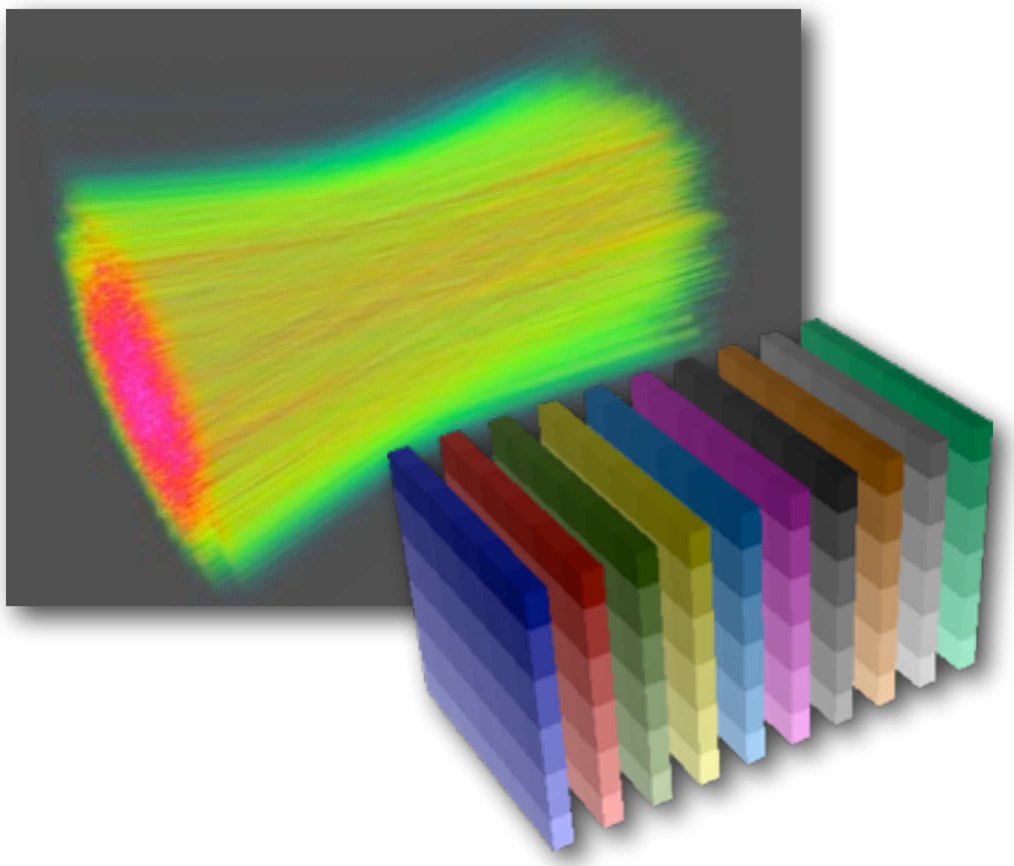
Mapping pF3D

- A laser-plasma interaction code used at the National Ignition Facility (NIF) at LLNL
- Three communication phases over a 3D virtual topology:
  - Wave propagation and coupling: 2D FFTs within XY planes
  - Light advection: Send-recv between consecutive XY planes
  - Hydrodynamic equations: 3D near-neighbor exchange



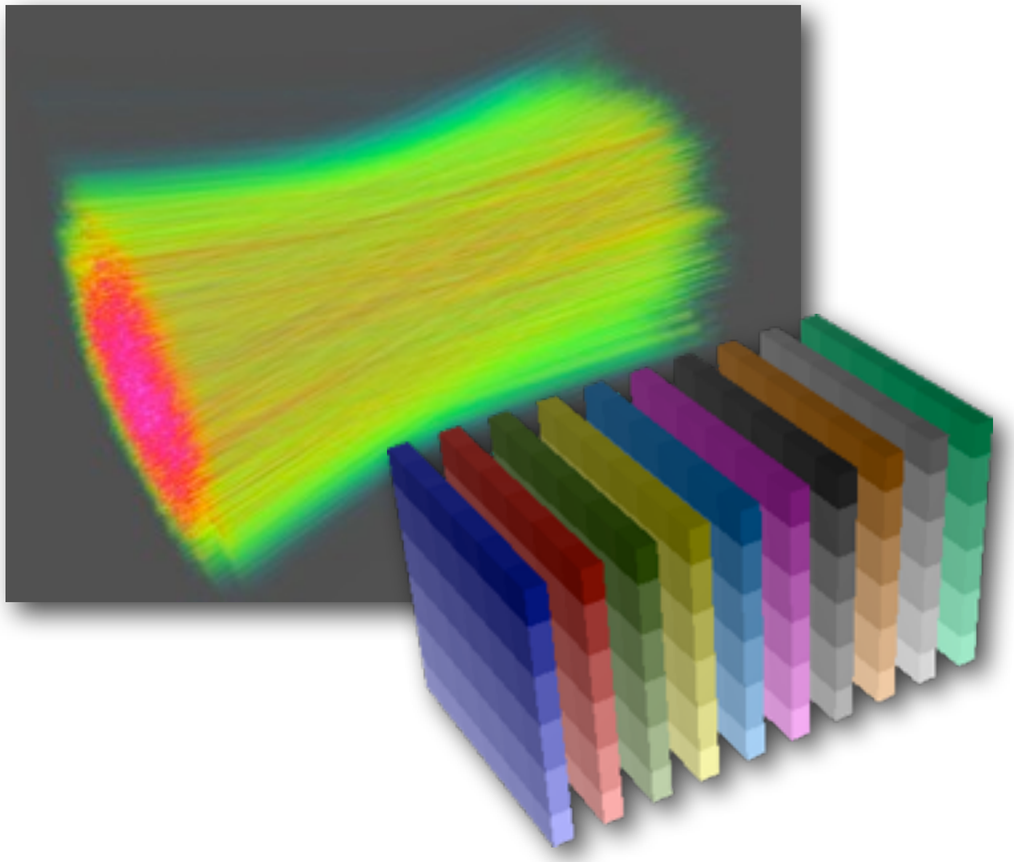
Mapping pF3D

- A laser-plasma interaction code used at the National Ignition Facility (NIF) at LLNL
- Three communication phases over a 3D virtual topology:
  - Wave propagation and coupling: 2D FFTs within XY planes
  - Light advection: Send-recv between consecutive XY planes
  - Hydrodynamic equations: 3D near-neighbor exchange



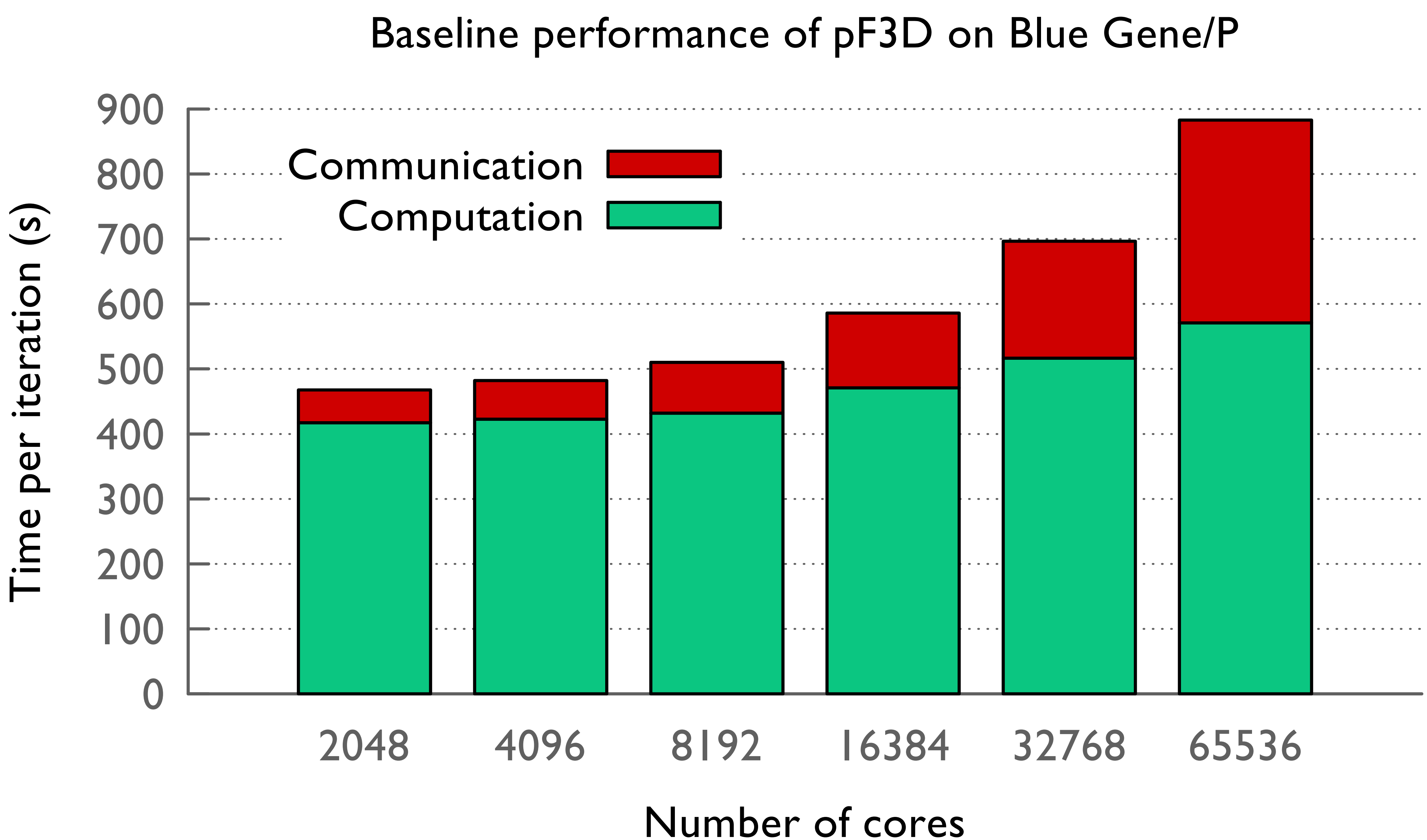
Mapping pF3D

- A laser-plasma interaction code used at the National Ignition Facility (NIF) at LLNL
- Three communication phases over a 3D virtual topology:
  - Wave propagation and coupling: 2D FFTs within XY planes
  - Light advection: Send-recv between consecutive XY planes
  - Hydrodynamic equations: 3D near-neighbor exchange

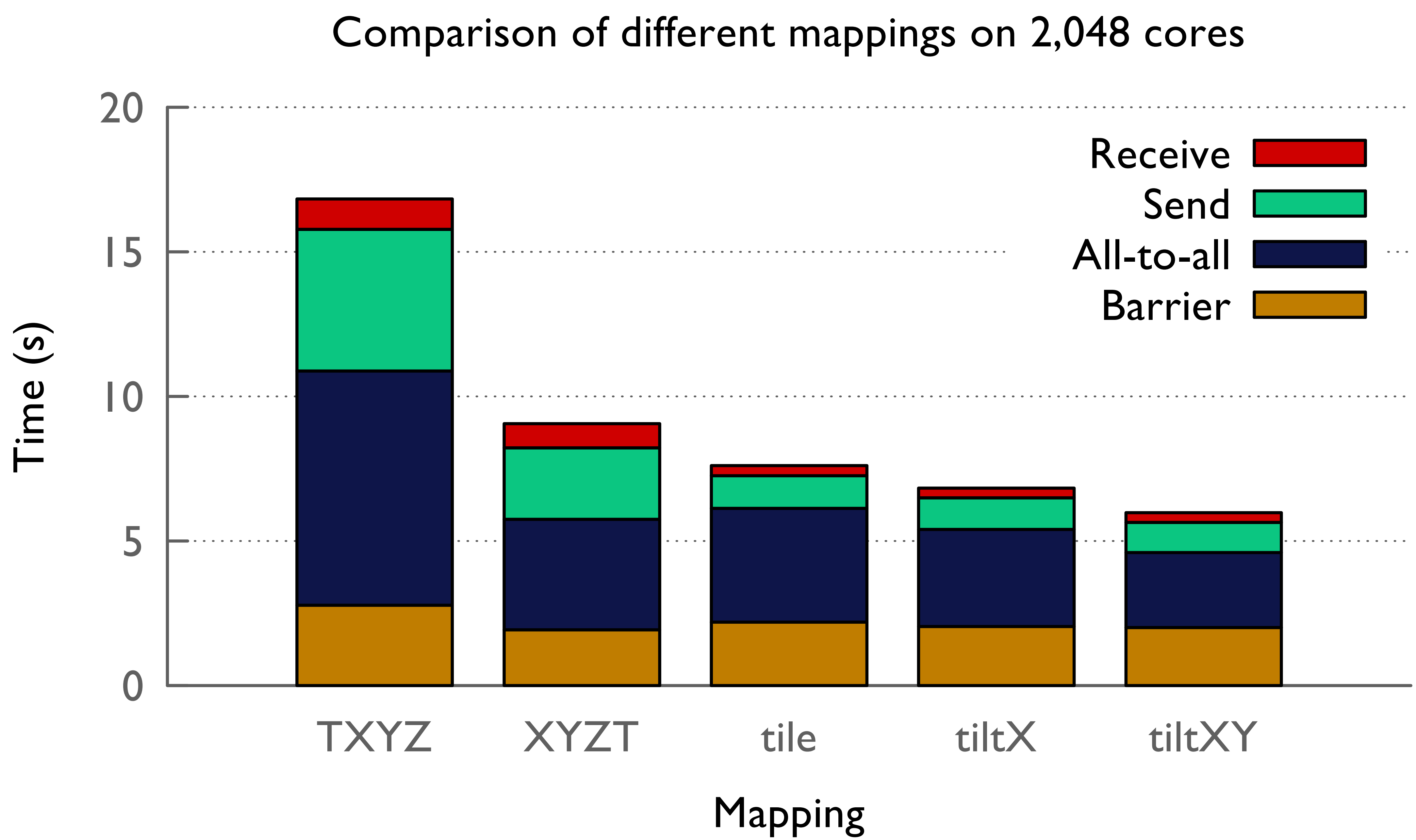


	2048 cores		16384 cores	
MPI call	**Total %**	**MPI %**	**Total %**	**MPI %**
Send	4.90	28.45	23.10	57.21
Alltoall	8.10	46.94	7.30	18.07
Barrier	2.78	16.10	8.13	20.15

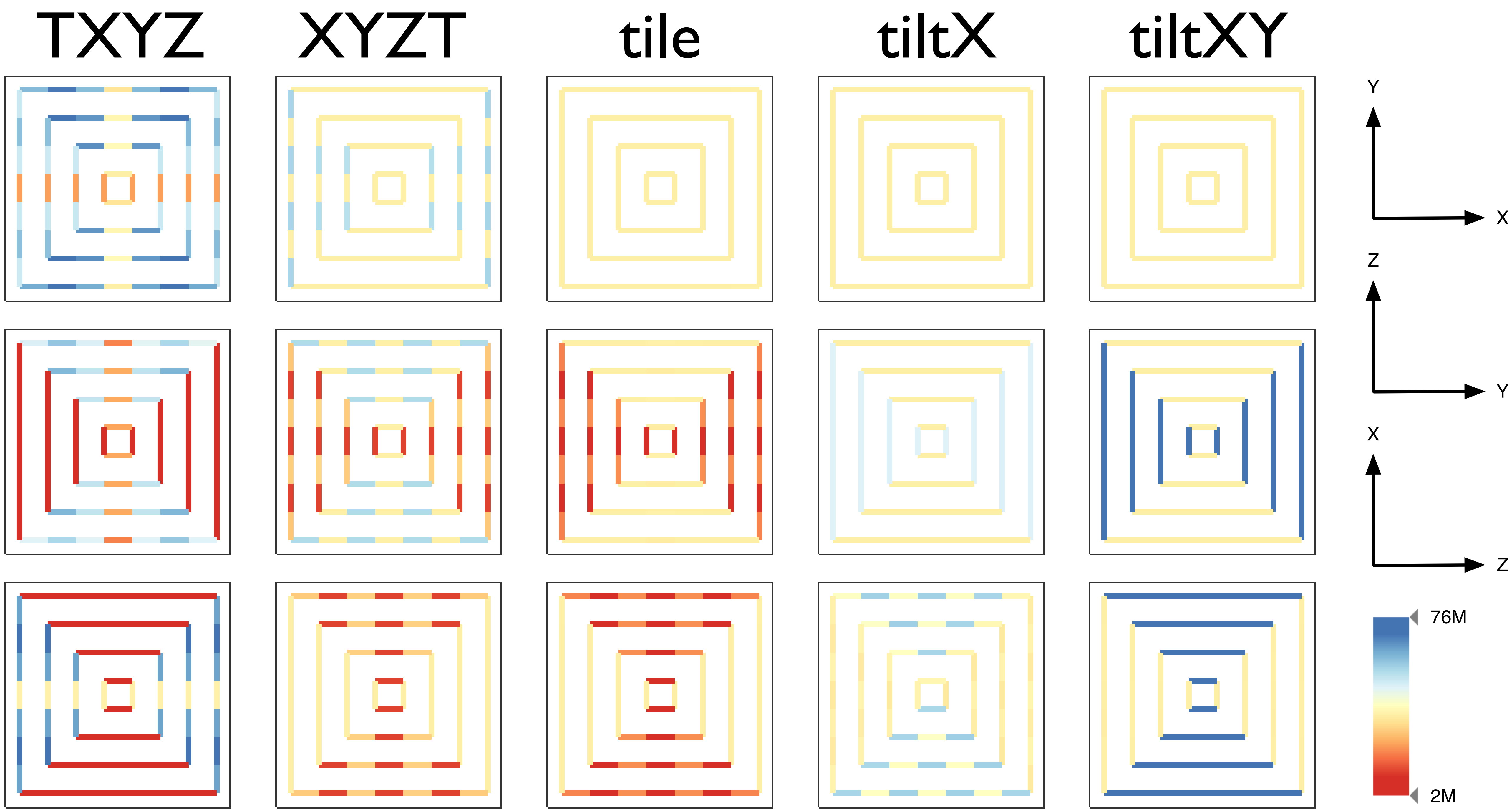
Default mapping



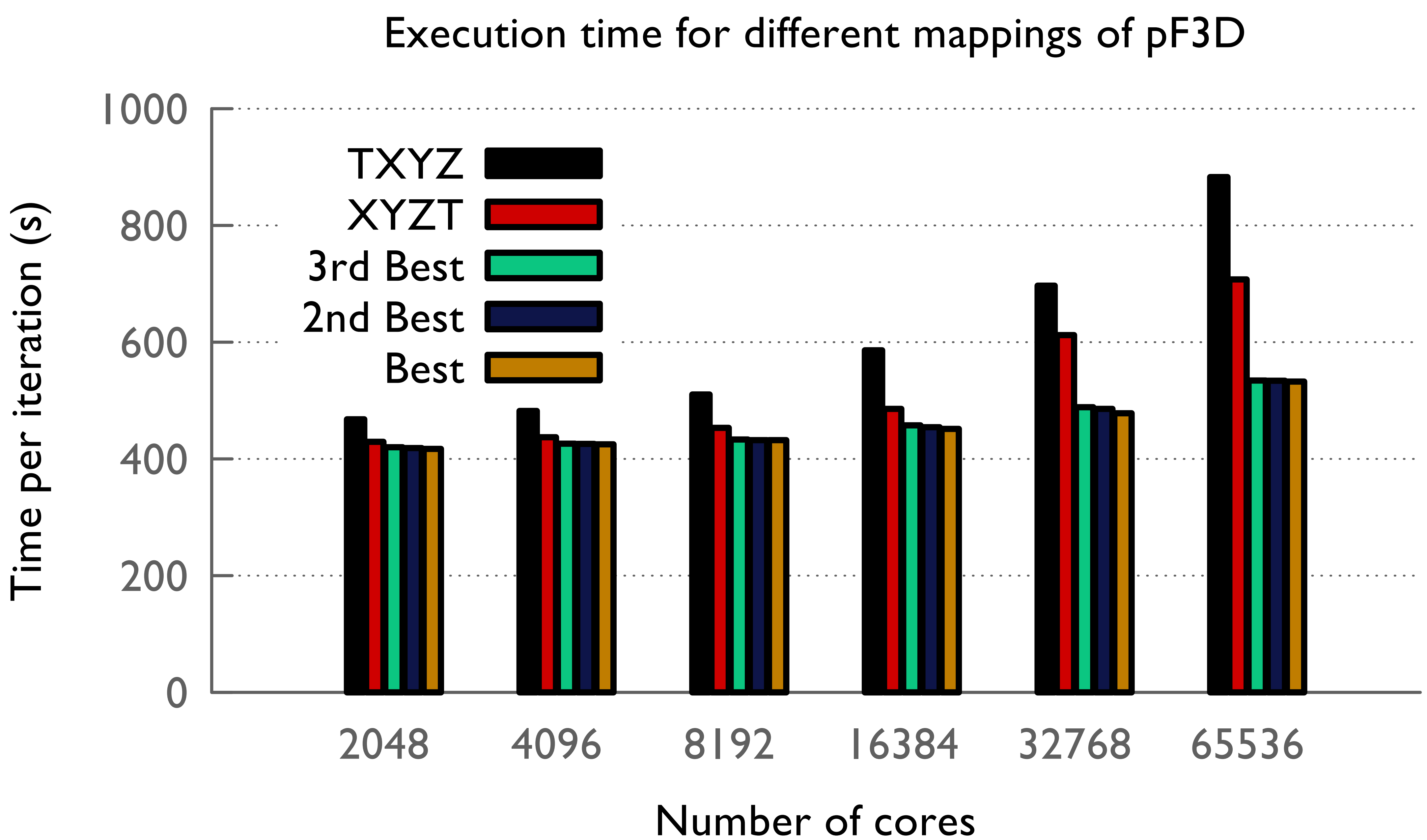
Communication breakdown



Visualizing network traffic using Boxfish



Scaling performance of pF3D



Summary

- Lots of time and energy spent in communication
- Bandwidth can't be optimized the same way as latency
- Rubik provides intuitive operations for quickly creating optimized task mappings
  - Close to 50% improvement for pF3D application
- Congestion and unstructured applications are still open problems

Download Rubik!

http://github.com/tgamblin/rubik