Dynamic Monitoring and Task Mapping Response

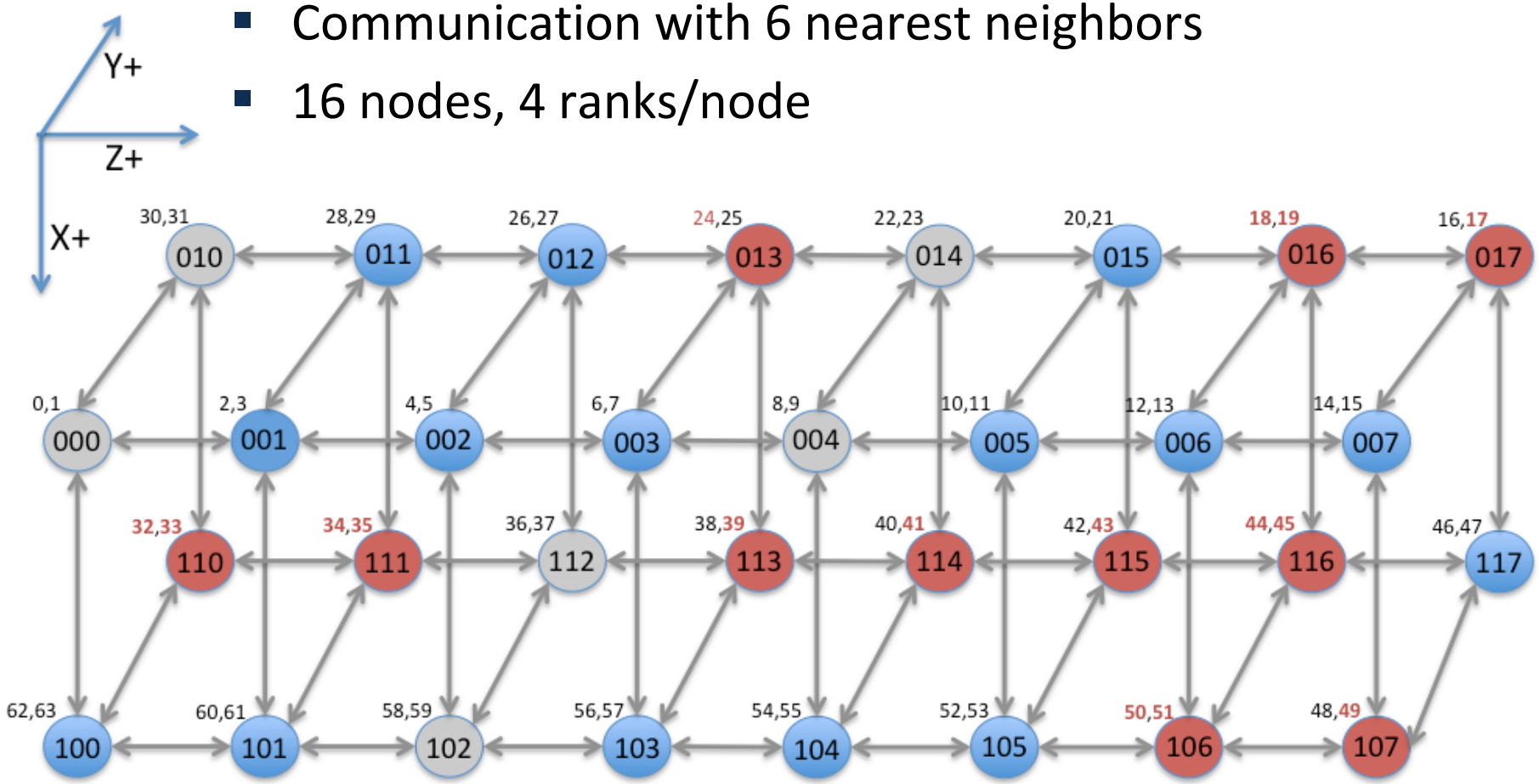
- Use *dynamic* information about *system* traffic to preferentially assign communicating tasks to cores to reduce the impact of competing traffic.
- Utilize dynamic information in architecture-aware context.
- Requires: determine meaningful architecture-aware measures of contention at run-time and deliver them on actionable time-scales *at scale*

Components of Analysis and Feedback

- Congestion monitoring. Weight potential communication routes based on congestion measures (max stalls along a route).
- Response determination via graph partitioning:
  - Graph-based mapping library (Scotch) maps tasks to nodes while attempting to minimize total cost of communication, accounting for both message sizes and communication cost across links.
  - Inputs: Task graph of application communication routes, weighted architecture graphs
- Response implementation via feedback of partitioning result to application

Example: Application Allocation

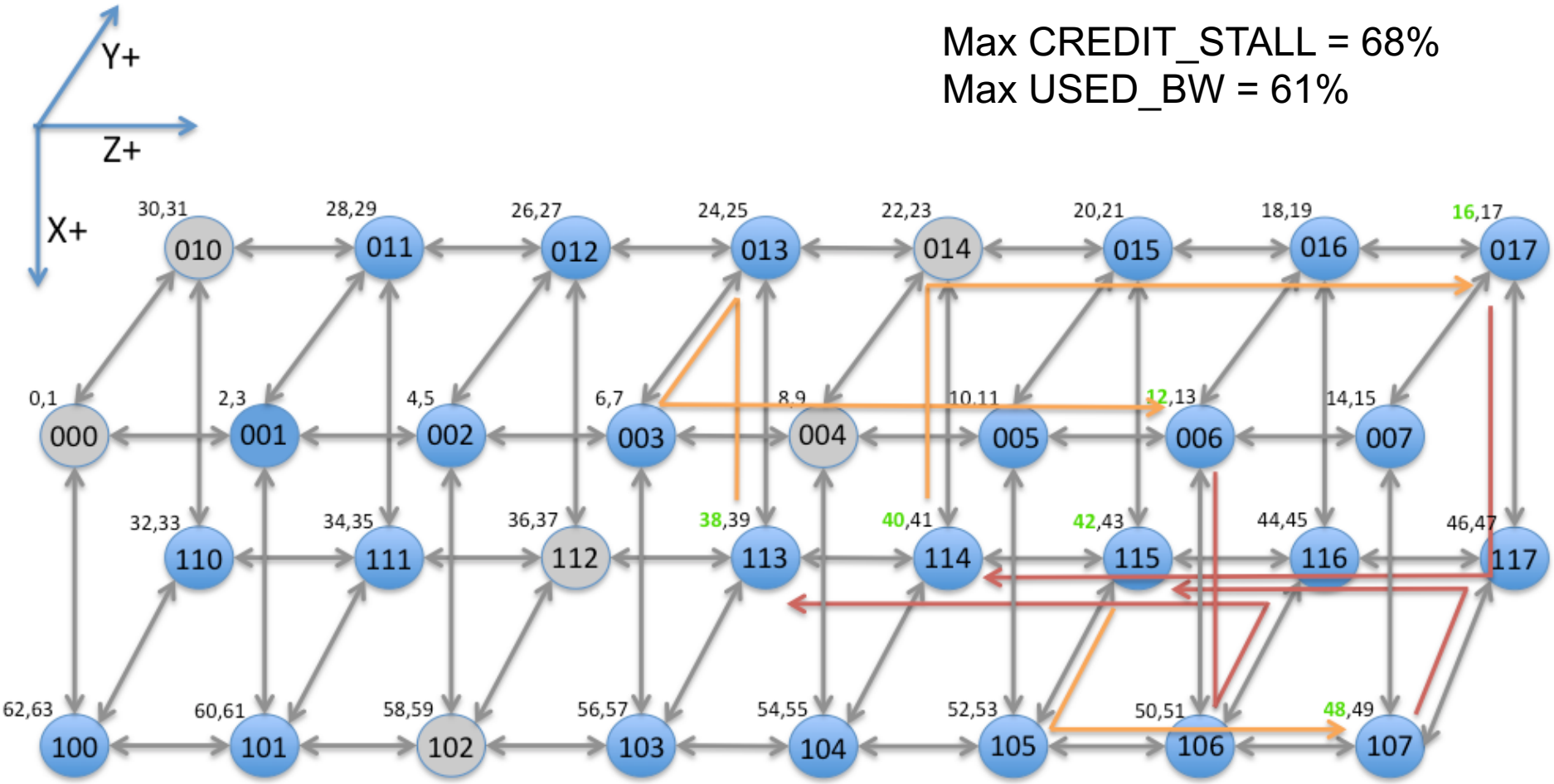
- Sparse Matrix Vector Computation.
- Communication with 6 nearest neighbors
- 16 nodes, 4 ranks/node



Network Dimensions: 2x2x8

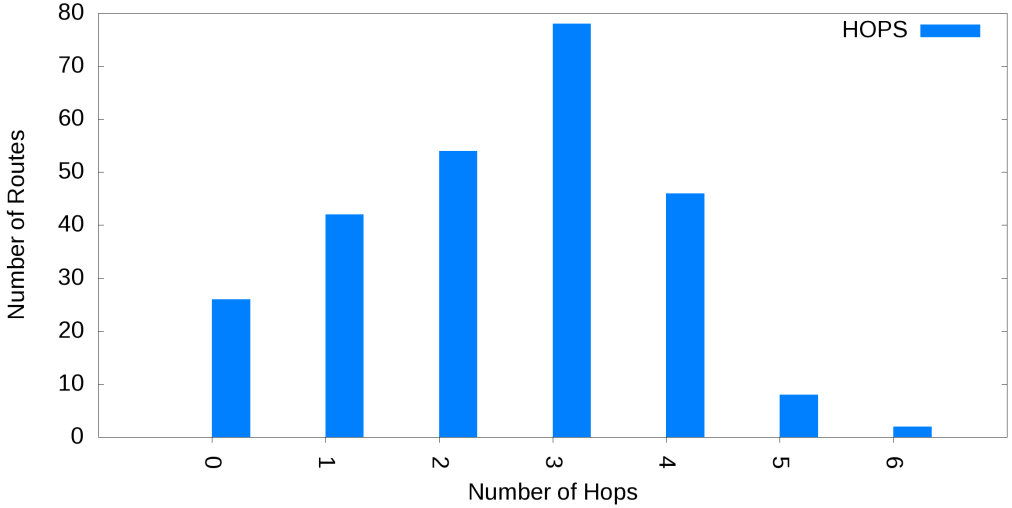
Competing Application with Network Traffic Demands

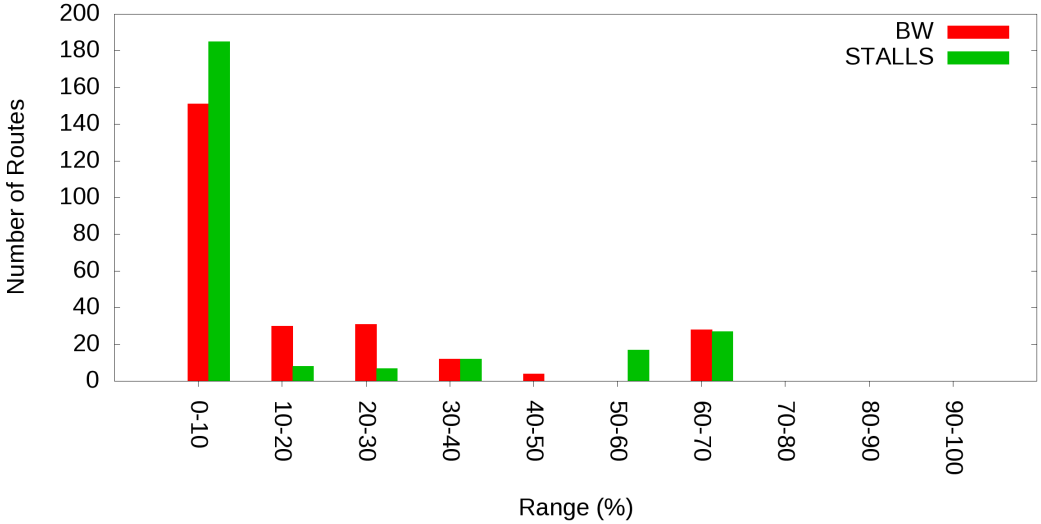
Max CREDIT_STALL = 68%
Max USED_BW = 61%



Contention Affects Potential Application Routes

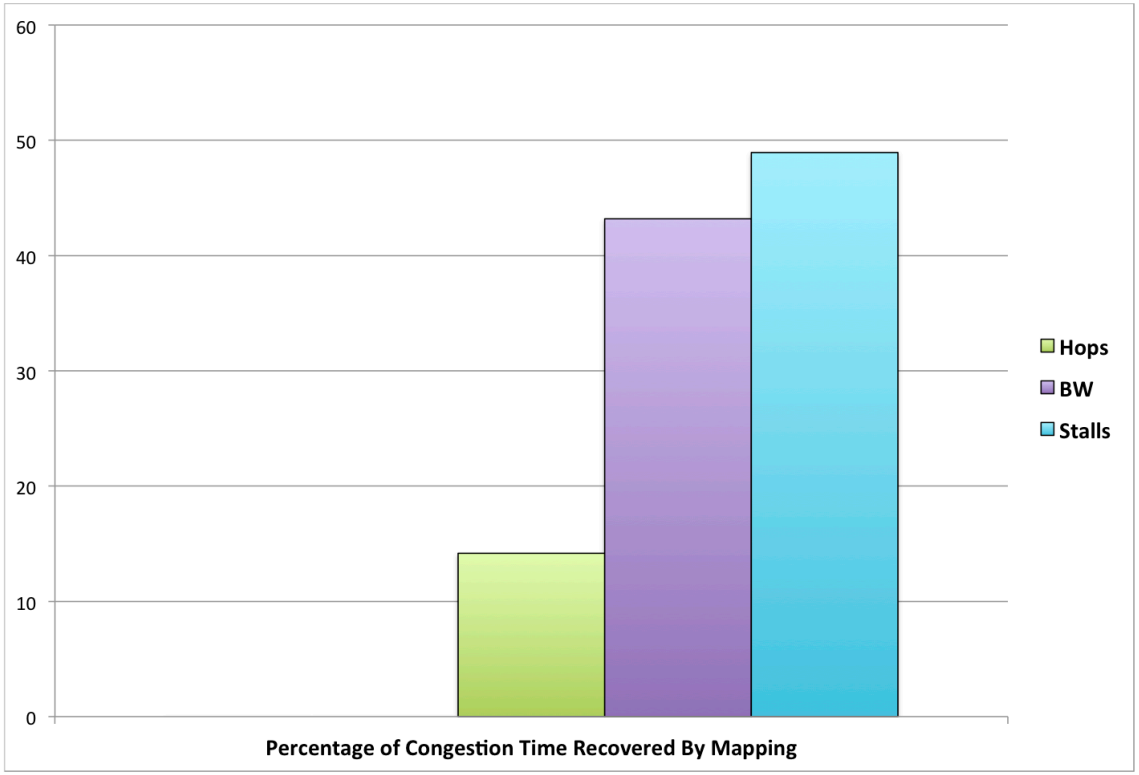
- 256 possible unique routes
- Task placement determines which routes are actually utilized during execution
- Not all combinations valid - restrictions due to the actual communication patterns
- Scotch Mapping: Minimize communication cost within the restrictions of the communication patterns





Results

- Percentage Execution Time Recovered by Performing Mapping with Various Metrics (higher is better)



Remapping based on dynamic network information in a congested environment recovered ~50% of the time lost to congestion.