SAND2014-17767PE

Exceptional service in the national interest

Sandia National Laboratories

Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping

J. Brandt, K. Devine, *A. Gentile*, K. Pedretti

Sandia National Laboratories,

Albuquerque, NM, USA

U.S. DEPARTMENT OF ENERGY

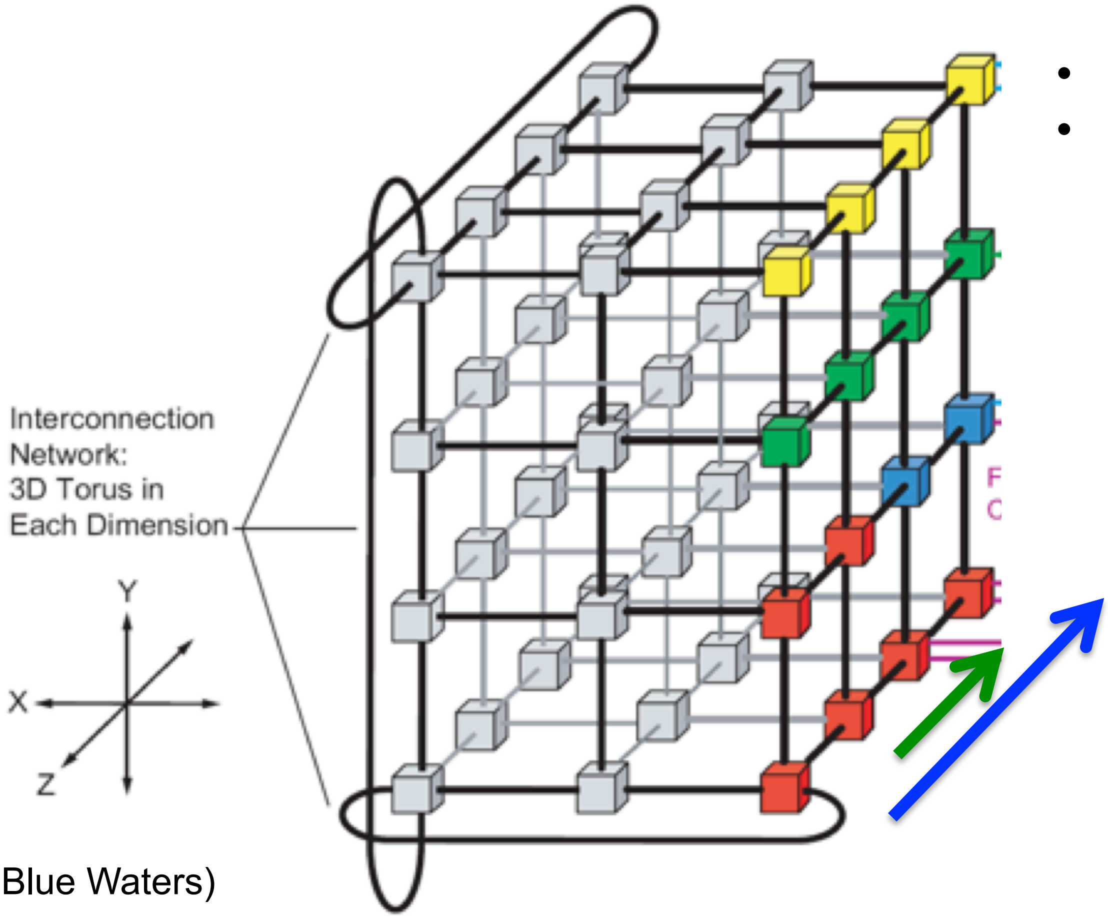
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Outline

- Motivation
- Approach
- Framework for delivering system state data to applications
  - Monitoring
  - Assessing and Presenting Dynamic State Information
  - Using Dynamic State Information for Task Mapping
- Application Performance, Congestion, and Mitigation
- Conclusions and Future Work

Shared Resources in the Gemini Network



(Blue Waters)

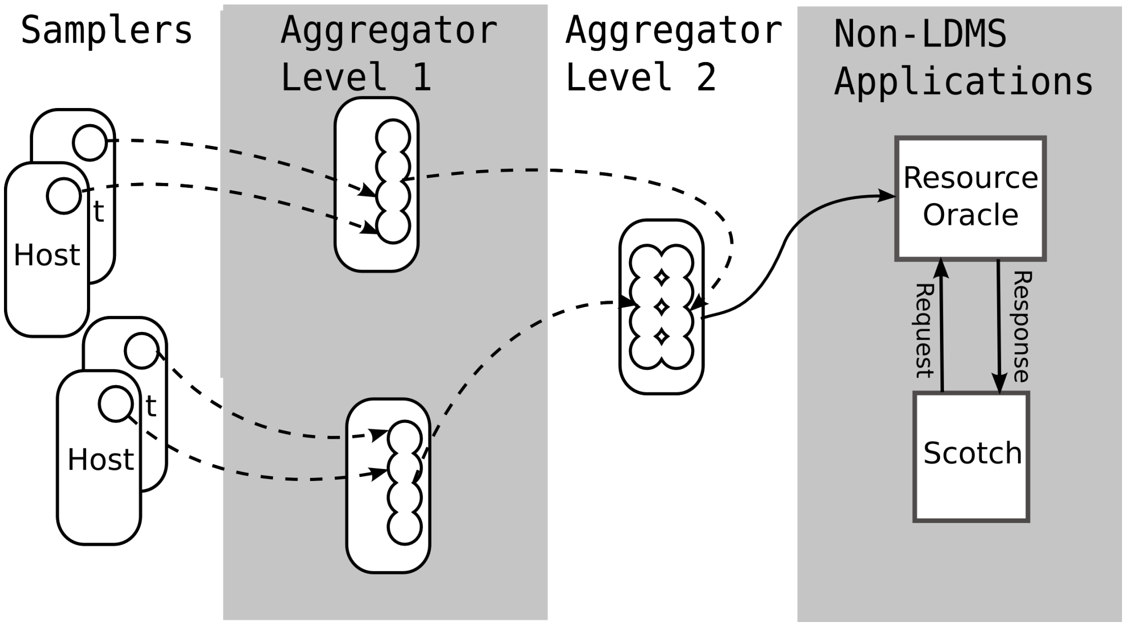
- 2 nodes share a Gemini router
- Routing algorithm:
  - X, Y, Z in order
  - Tie breaking
  - Forward and reverse routes may not be the same

- An application may be impacted by the traffic of other applications.
- An application cannot get a measure of contention from its view alone.
- In practice, 40% variation in the messaging rate (Bhatele et al SC13).

Architecture-Aware Mapping

- Static system topology information for allocation decisions
  - Nid reordering, shape allocations – Blue Waters
- Partitioning and Task mapping by an application within its allocation
  - Tools for mapping applications to architecture information. Application provides architecture and communication info. *Primarily node-level*.
  - Geometric Mapping based on network topology (Deveci et al IPDPS).
  - Charm++ Environment: Grid and Torus topology aware mapping approximating communication costs by hop-bytes.
- Framework for Dynamic Monitoring and Task Mapping
  - Mapping based on dynamic network contention information and known application communication patterns
  - Framework provides dynamic information in architecture-aware context.
  - Difficulty: determine meaningful architecture-aware measures of contention at run-time and deliver them on actionable time-scales *at scale*

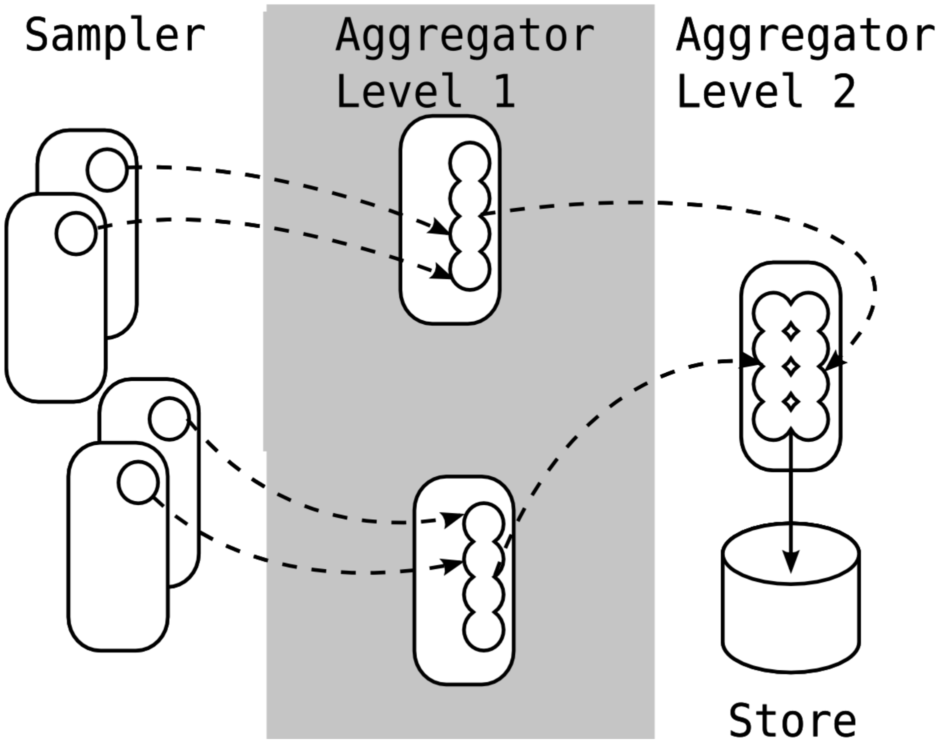
Framework for Providing State Data to Applications



Components:

- Monitoring – LDMS
- Assessing and Presenting Global Dynamic Data – Resource Oracle
- Determining Task Mapping - Scotch

Monitoring: LDMS



- Low overhead: 2MB, 0.01% CPU, O(100s) metrics/node
- Large-scale collection: RDMA over Gemini fan-in 16000:1
- High-fidelity: O(seconds)
- Complete system snapshots: resource allocation decisions based on a consistent global picture
  - within .25 sec on *Blue Waters* 27648 nodes

Congestion Measures in the Gemini Network

Sandia National Laboratories

U64 1 nettopo_mesh_coord_X
U64 1 nettopo_mesh_coord_Y
U64 6 nettopo_mesh_coord_Z
U64 511796170434 X+_traffic (B)
U64 11550455465 X+_packets (1)
U64 279915898696 X+_inq_stall (ns)
U64 53317089003 X+_credit_stall (ns)
U64 48 X+_sendlinkstatus (1)
U64 48 X+_recvlinkstatus (1)
U64 13 X+_SAMPLE_GEMINI_LINK_USED_BW (%)
U64 0 X+_SAMPLE_GEMINI_LINK_INQ_STALL (%)
U64 0 X+_SAMPLE_GEMINI_LINK_CREDIT_STALL (%)

- USED_BW - % of total theoretical bandwidth on an incoming link over the last sample interval.
- INQ_STALL - % of time the input queue of the Gemini spent stalled due to lack of credits.
- CREDIT_STALL - % of time that traffic could not be sent from the output queue due to lack of credits.

Credit based flow control: source can only send if it has credits

Architecture-Aware Dynamic Information: Resource Oracle

- Build the entire route between all pairs of nodes:
  - rtr --phys-routes: complete listing of routes between any 2 gemini

    rtr --phys-routes:
    {23,24,33,34,43,44,53,54}c0-0c0s0g0{00,01,10,11,25-27,35} ->
    {06,07,16-22,32}c0-0c0s1g0{00,01,10,11,25-27,35} ->
    {06,07,16-22,32}c0-0c0s2g0{00,01,10,11,25-27,35} ->
    {06,07,16-22,32}c0-0c0s3g0{23,24,33,34,43,44,53,54}

  - rtr --interconnect: link directions between any 2 directly connected gemini

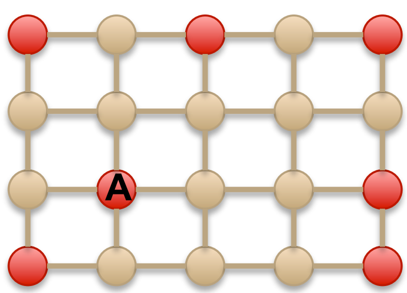
    rtr --interconnect:
    c0-0c0s0g0l00[(0,0,0)] Z+ -> c0-0c0s1g0l32[(0,0,1)] LinkType: backplane
    c0-0c0s0g0l02[(0,0,0)] X+ -> c0-0c1s0g0l02[(1,0,0)] LinkType: cable11x

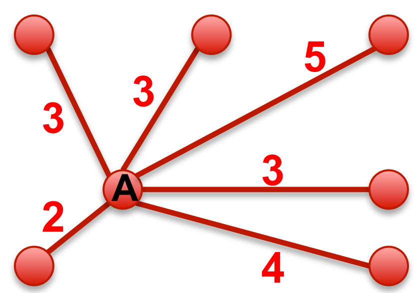
- Combine the route and monitoring information to calculate measures of congestion to characterize the entire route between any pairs of nodes
- API to query for static and functions of dynamic information (e.g., Max(USED_BW)) between any pairs of nodes

Resource-Aware Task Mapping

- The *Scotch* graph-based mapping library maps tasks to nodes while attempting to minimize total cost of communication, account for both message sizes and communication cost across links.
  - (Pellegrini et al., LaBri, Inria Bordeaux)
- Input 1: Task graph (derived from the application)
  - Vertices represent MPI tasks
  - Weighted edges represent #bytes communicated between tasks
- Input 2: Architecture graph
  - Vertices represent allocated nodes
  - Weighted edges represent cost of communication between nodes
- Set edge weights using route characterizations from ResourceOracle
  - With static metrics (HOPS), distant processors have higher weights
  - With dynamic metrics (USED_BW, STALLS), heavily congested paths between processors have higher weights

Allocated nodes in a mesh network





Weighted edges (HOPS) incident to node A; similar edges exist between all pairs of allocated nodes.

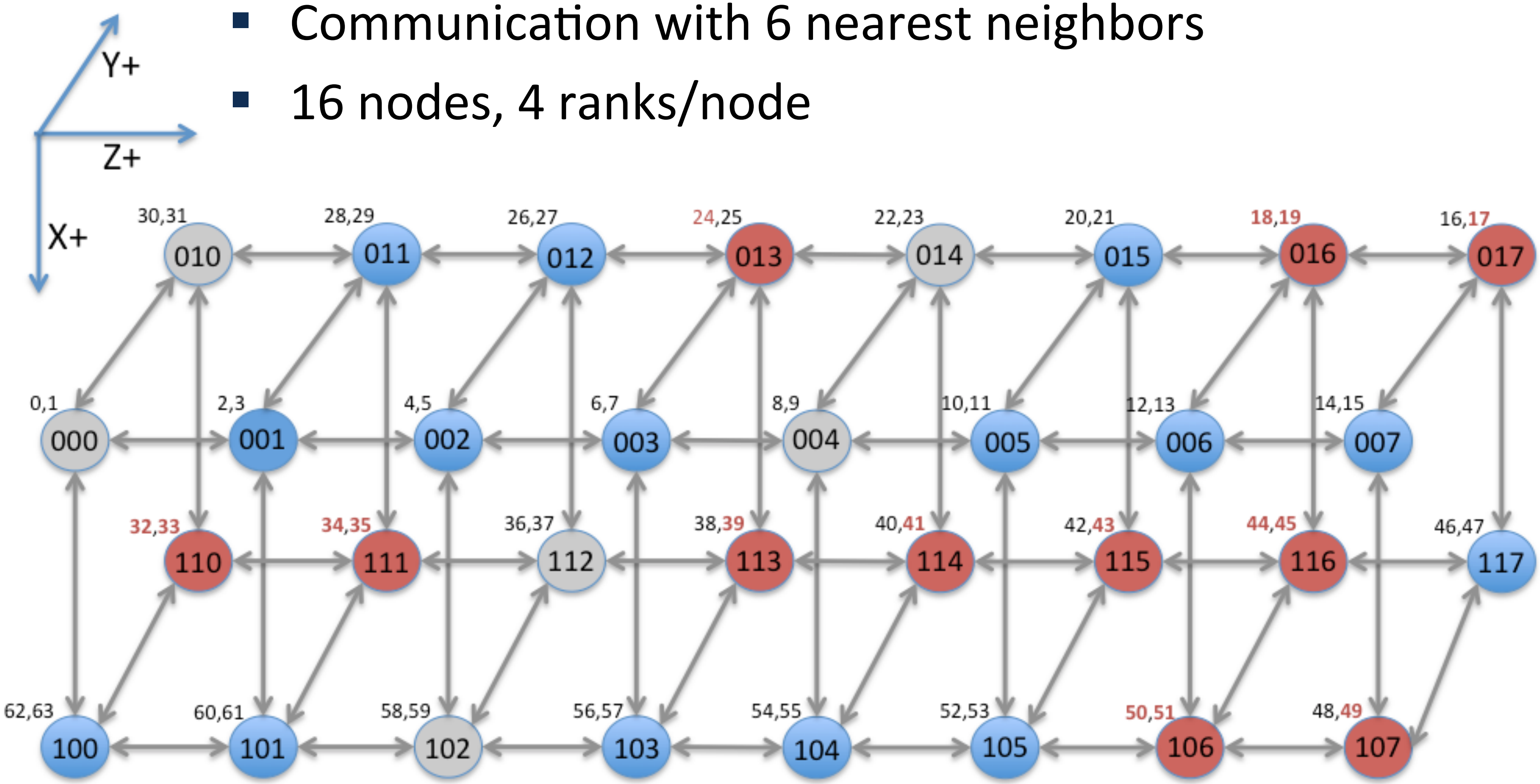
EXPERIMENT:
APPLICATION PERFORMANCE DEGRADATION DUE TO CONGESTION
AND
MITIGATING RESPONSE

Test kernel

- Sparse Matrix-Vector Multiplication (SpMV): *Ax*
  - Key kernel of many scientific applications
- Communication is primarily point-to-point communication to obtain needed off-processor *x* values
- Task graph is determined by matrix's non-zero pattern and parallel distribution of matrix/vector
  - *A* is distributed row-wise; matching distribution of *x*
  - For chosen matrix, each rank communicates with at most six neighbors

Application Allocation

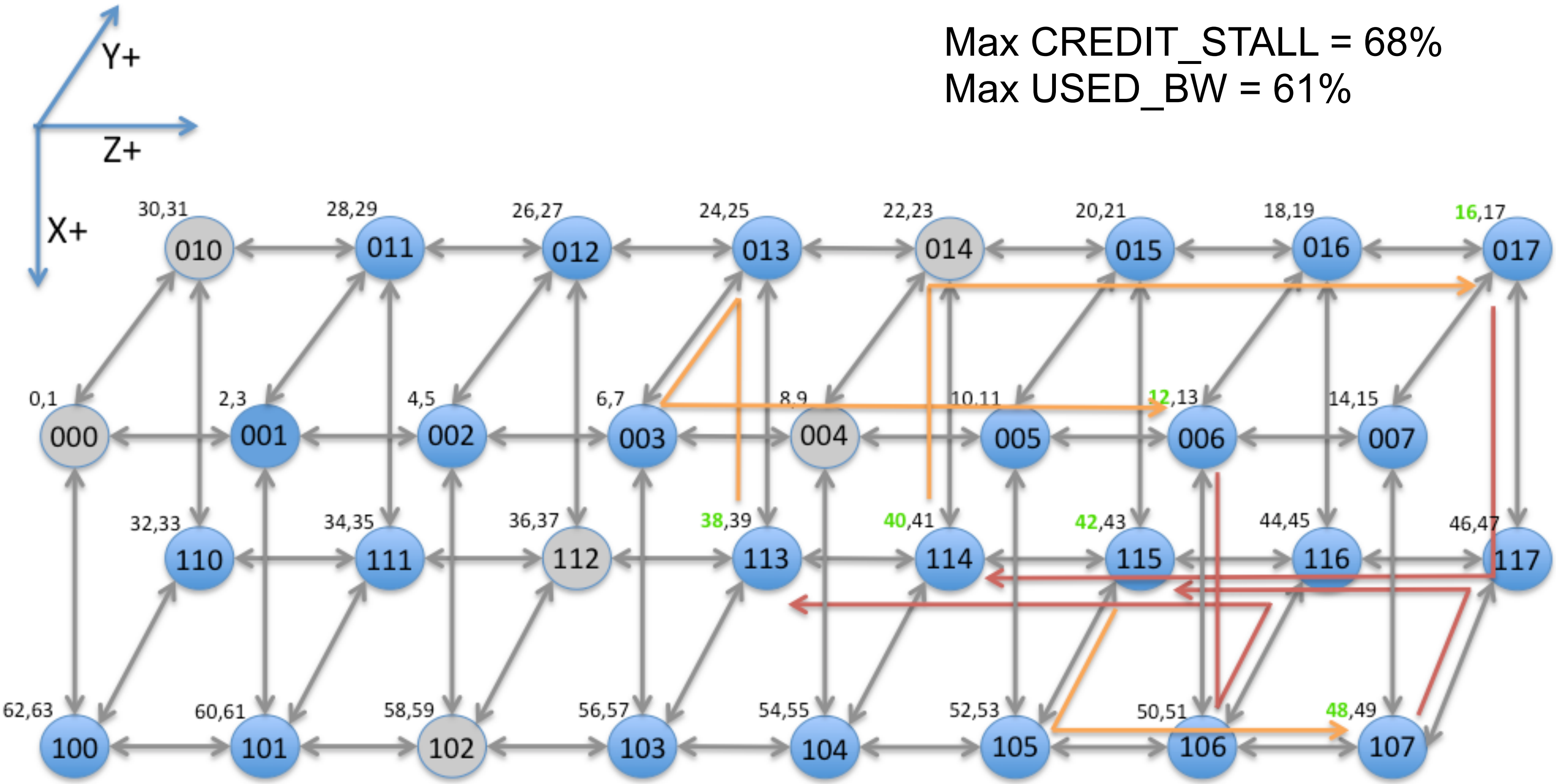
- Sparse Matrix Vector Computation.
- Communication with 6 nearest neighbors
- 16 nodes, 4 ranks/node



Network Dimensions: 2x2x8

Competing Application with Network Traffic Demands

Max CREDIT_STALL = 68%
Max USED_BW = 61%



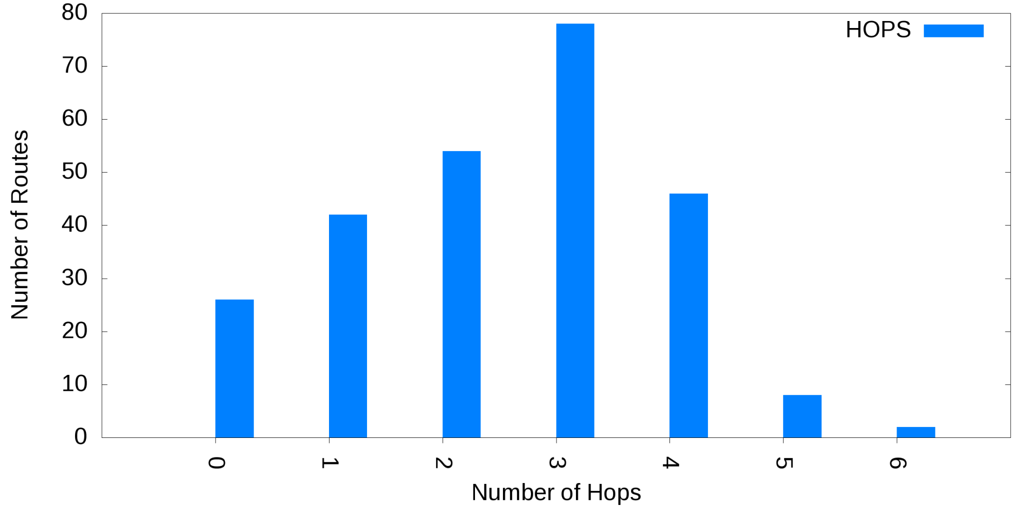
Affect of Congestion on Application Performance

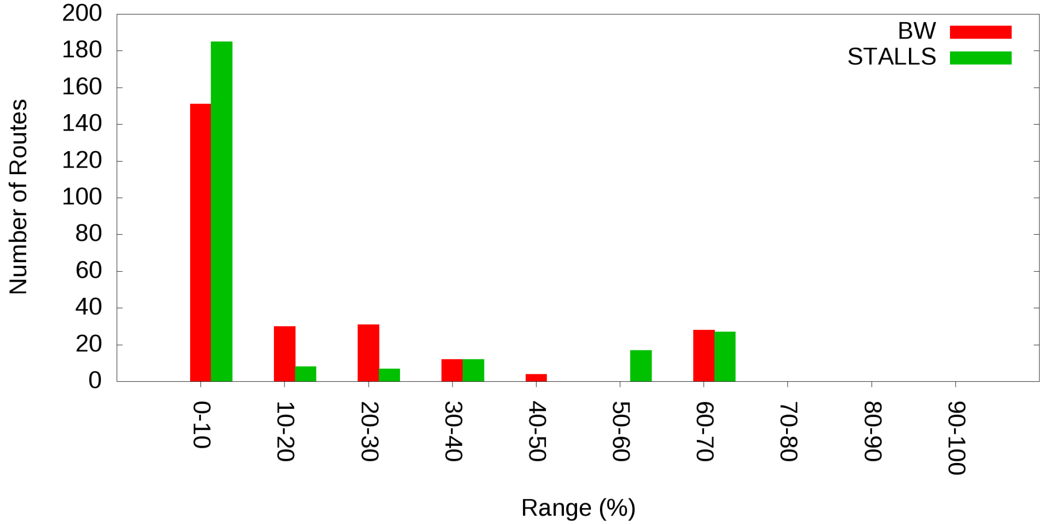
- Average SpMV time (sec) for 10K MatVecs:
  - Without congestion: 5.07 sec
  - With congestion: 6.03 sec
- Contention from competing application increases the average execution time by ~ 20%

- Parameters:
  - 10K Matvecs
  - 22-44 experiments
  - Each message contains 5x5x100 double precision values

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

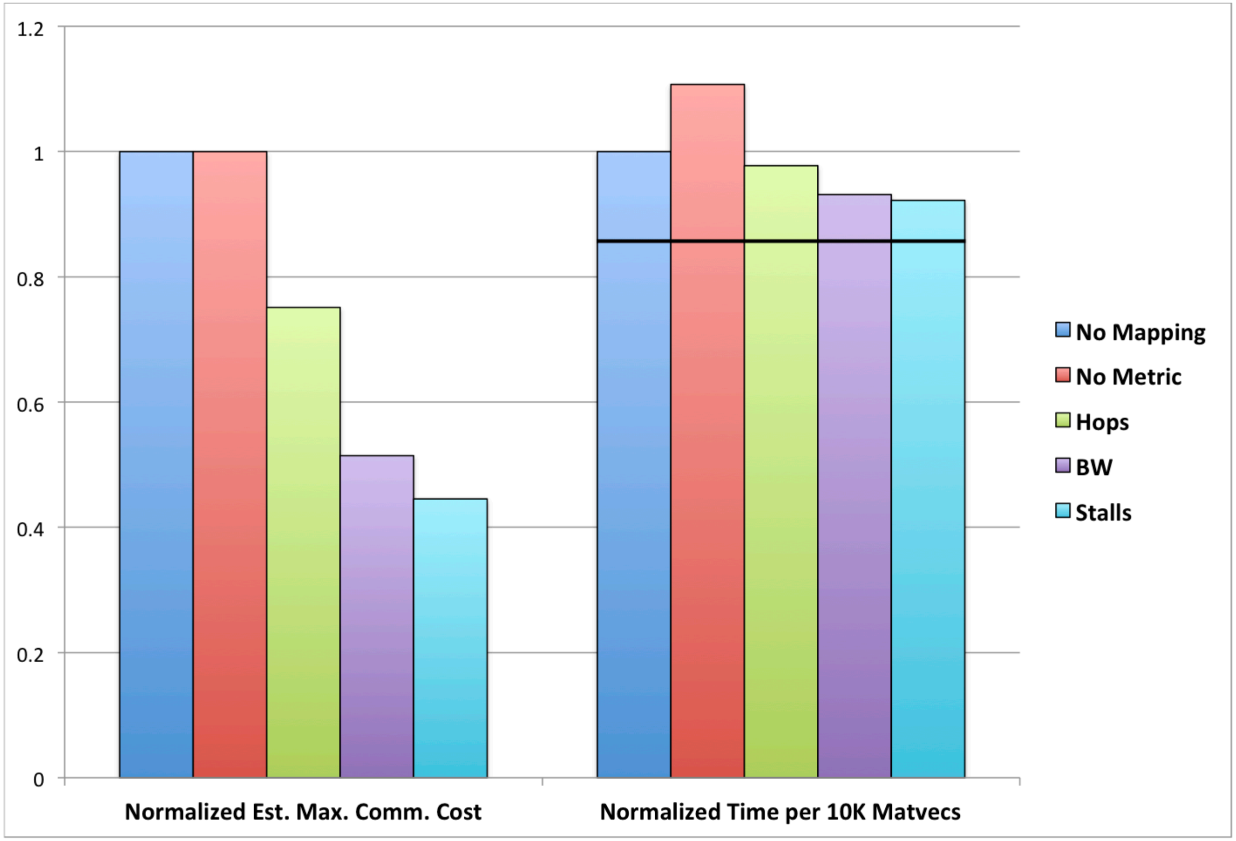




Determining Mapping Based on Static and Dynamic Information

- Simplistic weighting approach:
  - Static: HOPS -- locality but not congestion
    - Comm cost = HOPS x bytes
  - Dynamic: CREDIT_STALL, USED_BW -– congestion but not locality
    - Comm cost = (Max(STALLS) OR Max(USED_BW)) x bytes
- Compare with:
  - No mapping: RM assignments
  - No metric: uniform weights -- neither locality nor congestion
    - Comm cost = 1 x bytes (uniform)
- App migrates the matrix and vector data among processors according to the new mapping; MPI comm is unchanged
  - 0.012 sec remapping. 0.004 sec redistribution

Mapping with Congestion



Evinced max comm cost (left):

- Scotch reduces comm cost when a variable metric is used

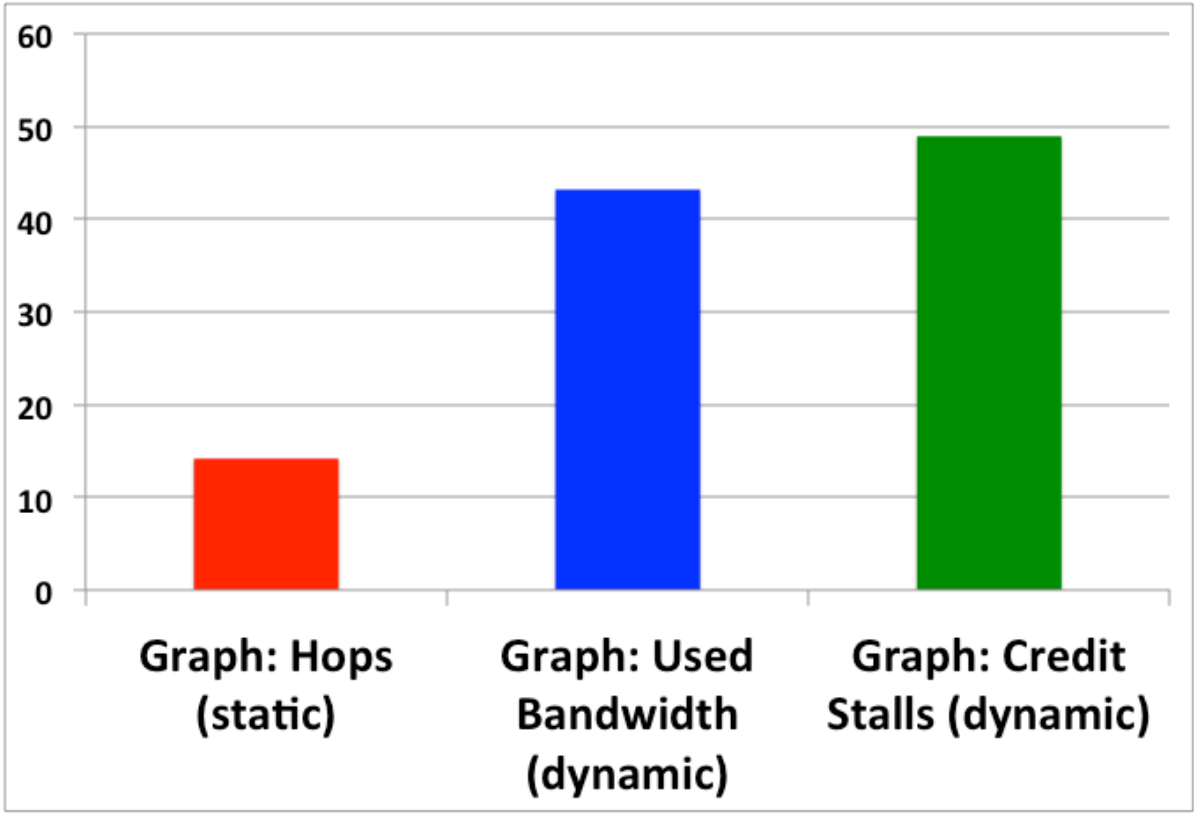
Execution Time (right):

- Black line: Uncongested result
- No Mapping – RM assignment (20% increase due to congestion)
- No Metric – Scotch with all routes equal
- HOPS
- USED_BW
- CREDIT_STALLS

Dynamic Task Mapping based on estimated communication costs due to run-time congestion monitoring reduces impact of competing application traffic

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.

Conclusions and Future Work

- Integrated framework for monitoring, analysis, and feedback to perform application-to-resource mapping that adapts to both static architecture features and dynamic resource state.
- Demonstrated significant potential benefits: recovered 50% of time lost to congestion.
- Next steps:
  - Performance optimization: Resource Oracle to directly access the LDMS aggregator data structures to reduce query overhead
  - Scalability: multiple Resource Oracles, parallel mapping
  - Metric Exploration: Metric value weighting and sensitivity (value, time)