Lecture 22: Topology Aware Mapping

Abhinav Bhatlele and Alan Sussman
Notes

• Graded exams returned on Tuesday
• Group Project presentations scheduled for Dec. 6, and Dec. 8 (if needed)
  • final report due Monday, December 12
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Congestion due to network sharing

• Sharing refers to network flows of different programs using the same hardware resources: links, switches

• When multiple programs communicate on the network, they all suffer from congestion on shared links
Communication is a bottleneck at scale

- GPU-based and multi-core CPU-based platforms have a large number of flop/s per node
  - Network bandwidths have not increased proportionally
- More energy is spent on sending data across the network

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (ns)</th>
<th>Energy spent (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating point operation</td>
<td>&lt; 0.25</td>
<td>30-45</td>
</tr>
<tr>
<td>Access DRAM</td>
<td>50</td>
<td>128</td>
</tr>
<tr>
<td>Get data from another node</td>
<td>&gt; 1000</td>
<td>128-576</td>
</tr>
</tbody>
</table>

Different approaches to mitigate congestion

• At the system level
  • Network topology aware job scheduler — attempts to assign compact allocation to jobs
  • Congestion-mitigating routing algorithms - this is the hotspot paper, assuming the job scheduler already assigned a set of nodes to the job

• At the individual job level
  • Users can try to optimize the mapping of MPI processes to allocated nodes – this is the BlueGene/L paper
Topology aware task mapping

• Also referred to as task placement or node mapping

• Given an allocation, decide which MPI processes are placed on which physical nodes/cores
  
  • In case of task-based models, map finer-grained tasks to cores

• Goal:
  
  • Minimize communication volume and/or number of messages on the network
  
  • Optimize “unavoidable” communication on the network
Topology aware task mapping

• Also referred to as task placement or node mapping

• Given an allocation, decide which MPI processes are placed on which physical nodes/cores
  • In case of task-based models, map finer-grained tasks to cores

• Goal:
  • Minimize communication volume on the network
  • Optimize “unavoidable” communication on the network
Topography aware task mapping

- Also referred to as task placement or node mapping
- Given an allocation, decide which MPI processes are placed on which physical nodes/cores
  - In case of task-based models, map finer-grained tasks to cores
- Goal:
  - Minimize communication volume on the network
  - Optimize “unavoidable” communication on the network
Topology aware task mapping

- Also referred to as task placement or node mapping
- Given an allocation, decide which MPI processes are placed on which physical nodes/cores
  - In case of task-based models, map finer-grained tasks to cores
- Goal:
  - Minimize communication volume on the network
  - Optimize “unavoidable” communication on the network
Graph embedding problem

• Inputs: Application communication graph, network topology graph (of one’s job allocation)

• Output: Process-to-node/core mapping – BlueGene/L paper describes and evaluates a simulated annealing algorithm for the optimization problem of mapping processes to nodes

• Most mapping algorithms do not consider that communication patterns might evolve over time
Metrics to evaluate mapping

- Hop-count

\[ \sum_{(i,j)} H(i,j) \]

- Hop-bytes

\[ \sum_{(i,j)} C(i,j) \times H(i,j) \]
Different techniques

- Heuristics-based
  - Recursive bi-partitioning
  - Random pairwise swaps
- Physical optimization algorithms
  - Simulated annealing
  - Genetic algorithms
Rubik: Python tool for mapping

- Define various operations on prisms
  - Partitioning or blocking
  - Permuting operations

https://github.com/LLNL/rubik
Rubik: Python tool for topology-aware mapping from LLNL

• Define various operations on prisms
  • Partitioning or blocking
  • Permuting operations

https://github.com/LLNL/rubik
Global link bottleneck in dragonfly systems

- Relatively few global links when building a smaller than full-sized system
- And that can create hotspots/bottlenecks for some communication patterns

- \# LL+LR links = 32 \times 31 \times n
- \# D links = n \times (n-1)

![Graph showing the ratio of LL+LR to D links and number of links vs. number of supernodes]