OPTIQ: A Data Movement Optimization Framework for Data-centric Applications on Supercomputers

Huy Bui
Electronic Visualization Lab.
Dept. of Computer Science
University of Illinois at Chicago, USA
Supercomputers and their applications

Attributing Changes in the Risk of Extreme Weather and Climate (150 Million Core-Hours)

Computational Studies of Nucleosome Stability (20 Million Core-Hours)

Toward Crystal Engineering from First Principles (12 Million Core-Hours)

Computing the Dark Universe (40 Million Core-Hours)

Simulation of combustion engine (113 Million Core-Hours)

Electronic Visualization Laboratory, University of Illinois at Chicago
Data-centric applications on supercomputers

- Most of the applications are data-centric i.e. generating a large amount of data.

![Bar chart showing total amount of read/write data per simulation in some applications.](chart.png)

Total amount of read/write data per simulation in some applications.
Supercomputers meet Big Data

• From *Exascale*($10^{18}$ PFLOP/s) computing study: *Technology challenges in achieving exascale and systems and Synergistic Challenges in Data-Intensive Science and Exascale Computing* reports:
  – HPC has been compute-intensive, but is shifting toward data-centric computing.
  ⇒ Data is a big challenge in supercomputing.
Supercomputer’s Interconnection Network

- A supercomputer includes ten thousands compute nodes and high throughput and low latency interconnect network.
## Data Movement and Optimization Input in Supercomputers

<table>
<thead>
<tr>
<th>Layers</th>
<th>Data movements</th>
<th>Optimization Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Topology</td>
</tr>
<tr>
<td>Applications</td>
<td>Data flows from sources to destinations</td>
<td>No</td>
</tr>
<tr>
<td>Middleware (MPI, PGAS...)</td>
<td>Communication routines such as MPI_Send, MPI_Broadcast</td>
<td>No</td>
</tr>
<tr>
<td>Systems</td>
<td>Packets/messages routing</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Data movement and optimization inputs at different layers

Information lost when data is moved between layers.
Data Movement in Data-Centric Application

- Separate optimizations produce local optimization.
- This thesis proposes solutions to improve data movement performance for data-centric applications in supercomputing systems.
  - Optimizing data flows: holistic approach to take system routings, interconnection topology and application communication patterns into formulation.
  - Realizing in to Data Movement Optimization framework (OPTIQ).
Current status of data movement optimization and proposed solutions

<table>
<thead>
<tr>
<th>Desired Features</th>
<th>Optimization: Input parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interconnect topology</td>
</tr>
<tr>
<td>Layers</td>
<td>System routing</td>
</tr>
<tr>
<td>Applications</td>
<td>Comm. routines</td>
</tr>
<tr>
<td>Middleware libraries</td>
<td>Comm. patterns</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Limited</td>
</tr>
<tr>
<td></td>
<td>Limited, system-specific</td>
</tr>
<tr>
<td></td>
<td>Yes (wrappers)</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>System</td>
<td>Limited</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>OPTIQ</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

Data movement and optimization inputs at different layers and OPTIQ

OPTIQ takes more inputs promising higher throughput.
Related Work
Related work

• Optimizing data movement.
  – System routing.
    • Static routing: pre-compute paths to move data.
      – Mathematical model based optimization.
      – Heuristic approach.
      – Pros: Optimized for certain routines, fast at runtime.
      – Cons: not optimized for flows, not adapt to state-of-the-art traffic.
    • Adaptive routing: compute paths instantly based on current status of traffic at local regions (sources and destination).
      – Randomized routing, Minimal routing.
      – Pros: fast to adapt to current traffic.
      – Cons: Not globally optimal.
Related work

- Optimizing data movement.
  - General work.

<table>
<thead>
<tr>
<th>Works</th>
<th>Static</th>
<th>Dynamic</th>
<th>Network Domain</th>
<th>Optimization Technique</th>
<th>@Layer</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valiant’81</td>
<td></td>
<td>Yes</td>
<td>General</td>
<td>Randomized</td>
<td>System</td>
<td>Large</td>
</tr>
<tr>
<td>Khana’08</td>
<td>Yes</td>
<td></td>
<td>Grid</td>
<td>Liner Programming</td>
<td>Libraries</td>
<td>Small</td>
</tr>
<tr>
<td>Rodriguez’09</td>
<td></td>
<td>Yes</td>
<td>Fat Tree</td>
<td>Heuristics</td>
<td>System</td>
<td>Small</td>
</tr>
<tr>
<td>Prisacari’13a</td>
<td>Yes</td>
<td></td>
<td>Fat Tree</td>
<td>Heuristics</td>
<td>System</td>
<td></td>
</tr>
<tr>
<td>Prisacari’13b</td>
<td>Yes</td>
<td></td>
<td>Generalized Fat Tree</td>
<td>Integer Linear Programming</td>
<td>System</td>
<td>x1000</td>
</tr>
</tbody>
</table>

*Optimizing data movement.*
## Related work

- Optimizing data movement.
  - Related work on recent supercomputers.

<table>
<thead>
<tr>
<th>System</th>
<th>Static</th>
<th>Dynamic</th>
<th>Network Domain</th>
<th>Opt technique</th>
<th>@Layer</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/L,P</td>
<td>Yes</td>
<td></td>
<td>3D Torus</td>
<td>Heuristics</td>
<td>System</td>
<td>Large</td>
</tr>
<tr>
<td>BG/Q</td>
<td>Small net</td>
<td>Large net</td>
<td>5D Torus</td>
<td>Heuristics</td>
<td>System</td>
<td>Large</td>
</tr>
<tr>
<td>Cray</td>
<td>Yes</td>
<td></td>
<td>Fat Tree</td>
<td>Heuristics</td>
<td>System</td>
<td>Large</td>
</tr>
<tr>
<td>BG/Q</td>
<td>Yes</td>
<td></td>
<td>5D Torus</td>
<td>Heuristics</td>
<td>Middleware</td>
<td>Large</td>
</tr>
</tbody>
</table>
Inefficiencies of data movement in current systems and solutions

Multi-paths data movement can improve performance.
OPTIQ Framework
OPTIQ Framework

Communication demand: sources, destinations, data (sizes, buffers). Graph of nodes and links.

1. Path generation
   k shortest paths for each pair of communication

2. Path searching
   Optimization Heuristics

3. Schedule
   Split and put messages into queues to be sent out

4. Transport
   Transfer data from sources to destinations

OPTIQ

Reduce search space
Search paths and assign data to path
OPTIQ Framework (cont.)

• OPTIQ framework:
  – The framework exposes a simple API that can be used in applications with minimal changes.
  – New features can be added easily: algorithm to search for paths, scheduling, transport.
  – It is also extensible to different systems.
Multi-path Data Movement
Path Generation

- Using any k-shortest paths algorithms to generate k shortest paths between a pair of source and destination.
- Pruning paths with length more than a certain number of hops e.g. diameter of partition of compute nodes.
Path Generation (cont.)
Heuristic Algorithm 1

• Assumptions:
  – All pairs of communication have similar number of paths and similar data size per path.

• Goal:
  – Limit the maximum number of paths per link by a given maxload value.

⇒ Limit the data passing through any link.

• Algorithm:
  – Iterate through all pairs.
    • Pick one path per pair at a time.
    • Update load of links and selected paths.
    • If any load on links is over maxload value, terminate.
Heuristic Algorithm 1 (cont.)

Maxload = 2
Heuristic Algorithm 1 (cont.)
Heuristics Algorithm 2

- **Assumptions:**
  - Data size can be different. Number of shortest paths per pair can be different.

- **Goal:**
  - Minimize the maximum data passing through any link.

- **Algorithm:**
  - Sort all pairs by data size (demand). The job with largest demand at top.
  - Pick the job at top, assign part of its demand (chunk) to one of its paths with lowest demand. Update the demand of the jobs and loads on links.
  - Repeat until all demands are assigned.
Heuristic Algorithm 2 (cont.)

Chunk size = 1MB

Source Nodes

Destination Nodes
# Heuristic 1 vs. Heuristic 2

<table>
<thead>
<tr>
<th>Comparison factor</th>
<th>Heuristic 1</th>
<th>Heuristic 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load on physical link</td>
<td>Number of paths that used a link.</td>
<td>Actual amount of data passing through a link.</td>
</tr>
<tr>
<td>Pair iteration</td>
<td>Each pair 1 time to get 1 path.</td>
<td>The pair that has the largest amount of remaining data.</td>
</tr>
<tr>
<td>Need to know amounts of data in advance</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Comparison of Heuristic 1 vs. Heuristic 2.
Model-based Data Movement Optimization

• Problem modeling:
  – Given a set of Jobs and set of Paths for each job in Jobs. Each job has Demand[job], flow on path p flow[job, p].
  – Each edge (i,j) has capacity c(i, j).
  – Objective function:
    • Minimize the transfer time t.
  – Capacity constraint:
    \[ \sum_{\forall job \in Jobs} \sum_{\forall p \in kpaths_{job}} flow[job, p]_{i,j} \leq c(i, j) \]
  – Throughput constraint:
    \[ \sum_{\forall p \in kpaths_{job}} flow[job, p] = \frac{Demand[job]}{t} \]
Evaluations
Blue Gene/Q supercomputers

- Mira: 5th in top 500, 48K nodes, 10PF/S, 5D torus.
Communication Patterns

- 3 main communication patterns.

<table>
<thead>
<tr>
<th>Disjoint</th>
<th>Overlap</th>
<th>Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="disjoint.png" alt="disjoint" /></td>
<td><img src="overlap.png" alt="overlap" /></td>
<td><img src="subset.png" alt="subset" /></td>
</tr>
</tbody>
</table>

3 main communication patterns that used by most of applications.

- 91 experiments with Optimization, Heuristics 1 & 2 and MPI default mechanism MPI_Alltoallv.
Set of Experiments

- Scaling total number of nodes.
- Varying sources-destinations distance.
- Varying sources/destinations ratio.
- Random sources-destinations pairing.
- Paths searching time.
- Experiments on 2 applications.
Scaling total number of nodes

- Same message size – Disjoint pattern.

OPTIQ outperforms MPI at scale for disjoint pattern.
Scaling total number of nodes (cont.)

- Same message size – Overlap pattern.

**OPTIQ outperforms MPI at scale for overlap pattern.**
Scaling total number of nodes (cont.)

- Same message size – Subset pattern.

OPTIQ outperforms MPI at scale for subset pattern.
Scaling total number of nodes (cont.)

- Same message size – Data distribution on physical network links.

The better data distribution the higher performance.
Scaling total number of nodes (cont.)

- Random message sizes: Overlap pattern.

OPTIQ outperforms MPI with various message sizes.
Varying Source-Destination Distance

- Increase source-dest distance: Disjoint pattern.

Increasing distance can improve throughput.
Varying Source-Destination Ratio

- Increasing source/destination ratio – Disjoint pattern.

OPTIQ outperforms MPI when increasing ratio.
Random Source-Destination Pairing

- Experiment on 512 nodes: Random pairing sources and destinations for disjoint patterns (32 source nodes to 256 destination nodes) for 5 times. Collect and report average performance.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPT</td>
</tr>
<tr>
<td>Disjoint</td>
<td>193</td>
</tr>
<tr>
<td>Overlap</td>
<td>208</td>
</tr>
<tr>
<td>Subset</td>
<td>207</td>
</tr>
</tbody>
</table>

Performance (GB/s) when random source-destination pairing.

OPTIQ still outperforms well with random pairing.
Paths searching time

- Search times are significantly different (~10X)

Trade-off between search time and search quality.
Community Earth System Model (CESM)

- Simulate the earth climate. There are 4 models: Atmosphere, Ocean, Ice, Land.
- Coupling models: 4 models communicate via Coupler.
- 512 nodes, 4 ranks/node (total 2048 ranks).

CESM has 3 subset and 1 disjoint patterns.
Community Earth System Model (CESM) (cont.)

- Data movement throughput: 512 nodes, 32KB to ~2MB/node.

OPTIQ outperforms MPI (20%-46%) with different message sizes, random pairing.

<table>
<thead>
<tr>
<th>Coupling</th>
<th>Type</th>
<th>Performance</th>
<th>Improv %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPL-ATM</td>
<td>OPT</td>
<td>352</td>
<td>46%</td>
</tr>
<tr>
<td></td>
<td>HEU 2</td>
<td>350</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>MPI</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>CPL-LND</td>
<td>OPT</td>
<td>343</td>
<td>23%</td>
</tr>
<tr>
<td></td>
<td>HUE 2</td>
<td>332</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>MPI</td>
<td>278</td>
<td></td>
</tr>
<tr>
<td>CPL-OCN</td>
<td>OPT</td>
<td>135</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>HEU 2</td>
<td>136</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>MPI</td>
<td>104</td>
<td></td>
</tr>
</tbody>
</table>

Performance (GB/s) of data movement between models.
Hardware/Hybrid Accelerated Cosmology Code (HACC)

- Simulating universe from beginning ~15 billion years ago. Writing data to storage after each phase.

- In this experiment:
  - Aggregate data from compute nodes to bridge nodes.
  - Data size ~ 6MB/node.
Scaling number of compute nodes.

<table>
<thead>
<tr>
<th>Partition Size</th>
<th>Total Throughput (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>16</td>
</tr>
<tr>
<td>1024</td>
<td>32</td>
</tr>
<tr>
<td>2048</td>
<td>64</td>
</tr>
<tr>
<td>3072</td>
<td>128</td>
</tr>
<tr>
<td>4096</td>
<td>256</td>
</tr>
<tr>
<td>5120</td>
<td>512</td>
</tr>
<tr>
<td>6144</td>
<td>1024</td>
</tr>
</tbody>
</table>

Transfer bandwidth for data aggregation in I/O for HACC

- OPTIQ Heuristics 2
- MPI_Alltoallv

OPTIQ outperforms MPI at scale.
Conclusions
Dissertation Contributions

• Proposed a holistic approach to improve data movement for data-centric applications on supercomputers.
• Realized the approach in a framework OPTIQ with an easy to use API, requiring minimal changes to integrate into applications.
• Provided multi-path data movement with a number of algorithms.
• Implemented and demonstrated results on BGQ supercomputer with 2 applications, from ~2X-3X up to 5X improvement.
Publications/Submissions

• Submissions:

• Publications:
Thank you!
Efficacy of \textit{maxload} value

- Partition of 1024 nodes, 64 sources and 512 destinations, 1 rank/node, 8 MB/pair.
- Varying the \textit{maxload} value: 1, 2, 4, 8, 16, 32.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>MPI</th>
<th>Maxload value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Disjoint</td>
<td>45</td>
<td>31</td>
</tr>
<tr>
<td>Overlap</td>
<td>42</td>
<td>66</td>
</tr>
<tr>
<td>Subset</td>
<td>74</td>
<td>69</td>
</tr>
</tbody>
</table>

Performance (GB/s) when increasing \textit{maxload} value.

In general, OPTIQ performs best with \textit{maxload} = 16.
Efficacy of Number of Shortest Paths Fed into Solvers

- Partition: 2048 nodes, 128 sources and 1024 destinations, 1 rank/node, 8 MB/pair.

- Number of shortest paths: 4, 16, 32, 50.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>MPI</th>
<th>Number of paths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Disjoint</td>
<td>61</td>
<td>29</td>
</tr>
<tr>
<td>Overlap</td>
<td>59</td>
<td>82</td>
</tr>
<tr>
<td>Subset</td>
<td>111</td>
<td>99</td>
</tr>
</tbody>
</table>

Performance (GB/s) when increasing number of paths fed into solvers.

OPTIQ outperforms better with more paths fed.
Future work

• Further improve performance by investigating multiple solutions produced by solvers.
• Extend the work to different supercomputers.
• Reduce the solving time of solvers by graph-partitioning approaches.
• Provide the QoS for the supercomputers.
Efficacy of Number of Ranks per Node
Efficacy of Chunk Size

- Data is split into chunk to send out.
- Chunk size can affect performance.

![Graph showing total networking throughput of OPTIQ vs. MPI_Alltoallv for 512 nodes, 1 rank/node, ratio=1/8, different chunk sizes. The x-axis represents bytes ranging from 4K to 1M, and the y-axis represents total throughput (GB/s) ranging from 4 to 128. Different message sizes are plotted with distinct markers and colors.]
Efficacy of Message Size

- Due to overheads, OPTIQ shows better performance for messages larger than 512KB.
Efficacy of Solvers

• SNOP vs. CPLEX at 2K, 91 experiments.
• Measure AMPL time, solving time and throughput.
Quality of Service (QoS) via Multiple Routing Classes

• The systems treat all packets in the same ways i.e. best-effort routing/first come first serve. Thus, different data flows have the same priority/no priority at all. But:
  – Some flows are more critical in time/bandwidth.
  – Application developers/scientists can further optimize their applications given their understanding of communication patterns.

⇒ Provide capability of giving priorities for different data flows.
Quality of Service (QoS) via Multiple Routing Classes

- Still best-effort data movement.
- For each data pattern we create a routing class.
- Each routing class has its own routing priority.
- Users can assign priority for each class (class-based) or each flow (flow-based).
- Reserve resources for classes based on priority.
- Dynamically control scheduling priorities.
Quality of Service (QoS) via Multiple Routing Classes

• $l$ routing classes index $i = 1$ to $l$ and each requires having $\alpha_i$ portion of the total achievable bandwidth. We add 2 constraints for
  – Total throughput is 1:
  \[ \sum_{i=1}^{l} \alpha_i = 1 \]
  – Total throughput of class $i$ with flows $j$:
  \[ \sum_{j=1}^{h} f_j(s_j, w) = \alpha_i * T \]
  with $j$ is index of flows in routing class $i$ and $T$ being the total throughput that we need to maximize.
  – $\alpha_i$ can be assigned by users or dynamically adapt by the framework.
Optimizing Data Movement at Scale

• Proposed directions:
  – Symmetry of interconnect
  – Multiple level graph.
  – Giving options (offline, online, combined) depends on each problem.

(a) Graph reduction for a simple 2D mesh  (b) Graph reduction for a simple tree
Model-based Data Movement Optimization

• Problem modeling:
  – Given a set of Jobs and set of Paths for each job in Jobs. Each jobs has Demand[job], flow on path p flow[job, p].
  – Each edge (i,j) has capacity c(i, j).
  – Objective function:
    • Minimize the transfer time t.
  – Capacity constraint:
    \[ \sum_{\forall job \in Jobs} \sum_{\forall p \in kpaths_{job}} flow[job, p]_{i,j} \leq c(i, j) \]
  – Throughput constraint:
    \[ \sum_{\forall p \in kpaths_{job}} flow[job, p] = \frac{Demand[job]}{t} \]
Supercomputer’s Interconnection Network

- A supercomputer includes ten thousands compute nodes and high throughput and low latency interconnect network.
# Data movement in supercomputers

<table>
<thead>
<tr>
<th>Layers</th>
<th>Data movements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>Data flows from sources to destinations</td>
</tr>
<tr>
<td>Middleware (MPI, PGAS...)</td>
<td>Communication routines such as MPI_Send, MPI_Recv, MPI_Broadcast, MPI_Gather ....</td>
</tr>
<tr>
<td>Systems</td>
<td>Packets/messages routing</td>
</tr>
</tbody>
</table>

Electronic Visualization Laboratory, University of Illinois at Chicago
Communication Patterns

<table>
<thead>
<tr>
<th>Disjoint</th>
<th>Subset</th>
<th>Overlapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>n</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n</td>
</tr>
</tbody>
</table>
Community Earth System Model

- Coupling models
- 512 nodes, 4 ranks/node.

![Diagram of Coupling Models](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>Location</th>
<th>Num. of pairs of communication with Coupler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Between Ranks</td>
</tr>
<tr>
<td>ATM</td>
<td>0 - 1791</td>
<td>5227</td>
</tr>
<tr>
<td>LND</td>
<td>0 - 515</td>
<td>10390</td>
</tr>
<tr>
<td>ICE</td>
<td>516 - 1791</td>
<td>3018</td>
</tr>
<tr>
<td>OCN</td>
<td>1792 - 2047</td>
<td>2001</td>
</tr>
<tr>
<td>CPL</td>
<td>0 - 1791</td>
<td></td>
</tr>
</tbody>
</table>