Mapping parallel applications on the machine topology: Lessons learned

Abhinav Bhavele and Laxmikant V. Kale

Parallel Programming Laboratory
University of Illinois at Urbana-Champaign
Motivation

- Running a parallel application on a linear array of processors:
Motivation

• Running a parallel application on a linear array of processors:
Motivation

• Running a parallel application on a linear array of processors:
Motivation

• Running a parallel application on a linear array of processors:

• Typical communication is between random pairs of processors simultaneously
Interconnect Topologies

• Three dimensional meshes
  • 3D Torus: Blue Gene/L, Blue Gene/P, Cray XT4/5

• Trees
  • Fat-trees (Infiniband) and CLOS networks (Federation)

• Dense Graphs
  • Kautz Graph (SiCortex), Hypercubes

• Future Topologies?
  • Blue Waters, Blue Gene/Q

Roadrunner Technical Seminar Series, March 13th 2008, Ken Koch, LANL
Application Topologies

http://wrf-model.org/plots/realtime_main.php

http://www.ks.uiuc.edu/Gallery/Science/

http://math.lanl.gov/Research/Projects/meshing.shtml
Application Topologies

We want to map communicating objects closer to one another
The Mapping Problem

• Applications have a communication topology and processors have an interconnect topology

• Definition: Given a set of communicating parallel “entities”, map them on to physical processors to optimize communication

• Goals:
  • Minimize communication traffic and hence contention
  • Balance computational load (when \( n > p \))
No Contention Runs

![Graph showing No Contention Message Latencies (Torus - 8 x 8 x 16)](image)

Blue Gene/P
No Contention Runs

No Contention Message Latencies (Torus - 8 x 8 x 16)

Message Latency

% Difference

XT3 (BigBen)
Wormhole Routing

- Ni et al. 1993; Oh et al. 1997 - Equation for modeling message latencies:

\[
\frac{L_f}{B} \times D + \frac{L}{B}
\]

\(L_f\) = length of flit, \(B\) = bandwidth, 
\(D\) = hops, \(L\) = message size

- Relatively small sized supercomputers
- It was safe to assume message latencies were independent of distance
Benchmark Creating Artificial Contention

- Pair each processor with a partner that is $n$ hops away

1 hop

2 hops

3 hops
Results: Contention

Effect of distance on latencies (Torus - 8 x 8 x 16)

Message Size (Bytes)

Latency (us)

8 hops
7 hops
6 hops
5 hops
4 hops
3 hops
2 hops
1 hop

Blue Gene/P

Bhatele A., Kale L.V., Quantifying Network Contention on Large Parallel Machines, Parallel Processing Letters (Special Issue on Large-Scale Parallel Processing), 2009. Best Poster Award, ACM Student Research Competition, Supercomputing 2008, Austin, TX.
Results: Contention

Effect of distance on latencies (Torus - 8 x 8 x 16)

- 7 hops
- 6 hops
- 5 hops
- 4 hops
- 3 hops
- 2 hops
- 1 hop

Results: Contention

Bhatele A., Kale L.V., Quantifying Network Contention on Large Parallel Machines, *Parallel Processing Letters (Special Issue on Large-Scale Parallel Processing)*, 2009. Best Poster Award, ACM Student Research Competition, Supercomputing 2008, Austin, TX.
Results: Contention

Effect of distance on latencies (Torus - 8 x 8 x 16)

Bhatele A., Kale L. V., Quantifying Network Contention on Large Parallel Machines, Parallel Processing Letters (Special Issue on Large-Scale Parallel Processing), 2009. Best Poster Award, ACM Student Research Competition, Supercomputing 2008, Austin, TX.
Obtaining Topology Information
Topology Discovery

- Topology Manager API: for 3D interconnects (Blue Gene, XT)

- Information required for mapping:
  - Physical dimensions of the allocated job partition
  - Mapping of ranks to physical coordinates and vice versa

- On Blue Gene machines such information is available and the API is a wrapper

- On Cray XT machines, there is no easy way to obtain topology information
Cray XT machines

- Get nid (node ID) corresponding to an MPI rank:
  - XT3: cnos_get_nidpid_map
  - XT4/5: PMI_Portals_get_nid

- Get physical coordinates corresponding to nid:
  - rca_get_meshcoord

- Translate the origin and provide this information through the Topology Manager API
Bigben @ PSC

- **Bigben**: The first Cray XT3 system in the world
  - Officially unveiled on July 20, 2005 (ranked 44 in the top500 list) and decommissioned on March 31, 2010
  - Initially had 2.4 GHz single core Opterons (upgraded to 2.6 GHz dual-core nodes in late 2006) - 4,180 cores 21.5 TF
  - SeaStar interconnect (3D torus of size 11 X 12 X 16)
Bigben @ PSC

- Bigben: The first Cray XT3 system in the world
  - Officially unveiled on July 20, 2005 (ranked 44 in the top500 list) and decommissioned on March 31, 2010
  - Initially had 2.4 GHz single core Opterons (upgraded to 2.6 GHz dual-core nodes in late 2006) - 4,180 cores 21.5 TF
  - SeaStar interconnect (3D torus of size 11 X 12 X 16)

Thanks to Chad Vizino and Shawn Brown
Application Case Studies
Case Study I: OpenAtom

Performance on Blue Gene/L

- Default Mapping

Time per step (s)

Number of cores

- 512
- 1024
- 2048
- 4096
- 8192
Diagnosis

Timeline view (OpenAtom on 8,192 cores of BG/L) using the performance visualization tool, Projections
Mapping of OpenAtom Arrays

A. Bhathele, E. Bohm, and L. V. Kale. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In Euro-Par, LNCS 5704, pages 1015–1028, 2009. Distinguished Paper Award, Feng Chen Memorial Best Paper Award
Mapping of OpenAtom Arrays

PairCalculator and GSpace have plane-wise communication

RealSpace and GSpace have state-wise communication

A. Bhatele, E. Bohm, and L. V. Kale. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In Euro-Par, LNCS 5704, pages 1015–1028, 2009. Distinguished Paper Award, Feng Chen Memorial Best Paper Award
Mapping of OpenAtom Arrays

A. Bhatele, E. Bohm, and L. V. Kale. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In Euro-Par, LNCS 5704, pages 1015–1028, 2009. Distinguished Paper Award, Feng Chen Memorial Best Paper Award

Paircalculator and GSpace have plane-wise communication

RealSpace and GSpace have state-wise communication
Performance Benefits from Mapping

Performance on Blue Gene/L

- Default Mapping
- Topology Mapping

Time per step (s)

Number of cores

512 1024 2048 4096 8192
Diagnosis of Improvement

Timeline view using the performance visualization tool, Projections

Timeline of 1 iteration of OpenAtom running WATER_256M_70Ry on 8192 cores of BG/L
OpenAtom Performance on Blue Gene/P

Application Performance

- Default Mapping
- Topology Mapping

Time per step (s)

Number of cores

1024  2048  4096  8192

11  8.25  5.5  2.75
OpenAtom Performance on Blue Gene/P

Application Performance

- Default Mapping
- Topology Mapping

Performance Counters

- System Bandwidth (GB/step)

Time per step (s)

Number of cores

1024 2048 4096 8192

11 8.25 5.5 2.75

1100 825 550 275

0

1024 2048 4096 8192

Number of cores

1100 825 550 275

0
OpenAtom Performance on Blue Gene/P

Application Performance

- Default Mapping
- Topology Mapping

Performance Counters

- System Bandwidth (GB/step)

Time per step (s)

Number of cores

System Bandwidth (GB/step)

Number of cores
OpenAtom Performance on Cray XT3
OpenAtom Performance on Cray XT3

- Cray XT3:
  - Link bandwidth - 3.8 GB/s (XT3), 0.425 (BG/P), 0.175 (BG/L)
  - Bytes per flop - 8.77 (XT3), 0.375 (BG/P and BG/L)
OpenAtom Performance on Cray XT3

- Cray XT3:
  - Link bandwidth - 3.8 GB/s (XT3), 0.425 (BG/P), 0.175 (BG/L)
  - Bytes per flop - 8.77 (XT3), 0.375 (BG/P and BG/L)
- Job schedulers on Cray are not topology aware
OpenAtom Performance on Cray XT3

- Cray XT3:
  - Link bandwidth - 3.8 GB/s (XT3), 0.425 (BG/P), 0.175 (BG/L)
  - Bytes per flop - 8.77 (XT3), 0.375 (BG/P and BG/L)
- Job schedulers on Cray are not topology aware
- Performance Benefit at 2048 cores: 40% (XT3), 45% (BG/P), 41% (BG/L)
Case Study II: NAMD

Communication between patches and computes

Topology aware placement of computes

NAMD Performance on Blue Gene/P

Measured Hop-bytes

- Topology Oblivious
- TopoAware Patches
- TopoAware Computes

• Evaluation Metric: Hop-bytes

\[ HB = \sum_{i=1}^{n} d_i \times b_i \]

- \( d_i = \) distance
- \( b_i = \) bytes
- \( n = \) no. of messages

• Indicates amount of traffic and hence contention on the network

• Previously used metric: maximum dilation

\[ d(e) = \max\{d_i | e_i \in E\} \]
NAMD Performance on Blue Gene/P

Measured Hop-bytes

- Topology Oblivious
- TopoAware Patches
- TopoAware Computes

The number of computes on a processor and their individual computational loads determines its computational load and the number of proxies on a processor indicates its communication load. Load balancing in NAMD is measurement-based. This assumes that load patterns tend to persist over time and even if they change, the change is gradual (referred to as the principle of persistence). The load balancing framework records information about object loads for some time steps. It also records the communication graph between the patches and proxies. This information is collected on one processor and based on the instrumentation data, a load balancing phase is executed. Decisions are then sent to all processors. The current strategy is centralized and we shall later discuss future work to make it fully...
NAMD Performance on Blue Gene/P

Measured Hop-bytes

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Topology Oblivious</th>
<th>TopoAware Patches</th>
<th>TopoAware Computes</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>375</td>
<td>750</td>
<td>1125</td>
</tr>
<tr>
<td>1024</td>
<td>750</td>
<td></td>
<td>1500</td>
</tr>
<tr>
<td>2048</td>
<td>1125</td>
<td></td>
<td>1500</td>
</tr>
<tr>
<td>4096</td>
<td>1500</td>
<td></td>
<td>1500</td>
</tr>
</tbody>
</table>

Application Performance

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Topology Oblivious</th>
<th>TopoAware Patches</th>
<th>TopoAware Computes</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>7.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>3.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>3.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8192</td>
<td>1.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16384</td>
<td>1.25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The number of computes on a processor and their individual computational loads determines its computational load and the number of proxies on a processor indicates its communication load. Load balancing in NAMD is measurement-based. This assumes that load patterns tend to persist over time and even if they change, the change is gradual (referred to as the principle of persistence). The load balancing framework records information about object (compute) loads for some time steps. It also records the communication graph between the patches and proxies. This information is collected on one processor and based on the instrumentation data, a load balancing phase is executed. Decisions are then sent to all processors. The current strategy is centralized and we shall later discuss future work to make it fully.
NAMD Performance on Blue Gene/P

Measured Hop-bytes

- Topology Oblivious
- TopoAware Patches
- TopoAware Computes

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Hop-bytes (MB per iteration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0</td>
</tr>
<tr>
<td>1024</td>
<td>375</td>
</tr>
<tr>
<td>2048</td>
<td>750</td>
</tr>
<tr>
<td>4096</td>
<td>1125</td>
</tr>
</tbody>
</table>

Application Performance

- Topology Oblivious
- TopoAware Patches
- TopoAware Computes

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Time per step (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>15</td>
</tr>
<tr>
<td>1024</td>
<td>11.25</td>
</tr>
<tr>
<td>2048</td>
<td>7.5</td>
</tr>
<tr>
<td>4096</td>
<td>3.75</td>
</tr>
<tr>
<td>8192</td>
<td>6%</td>
</tr>
<tr>
<td>16384</td>
<td>13%</td>
</tr>
<tr>
<td>32768</td>
<td>28%</td>
</tr>
</tbody>
</table>

The number of computes on a processor and their individual computational loads determines its computational load and the number of proxies on a processor indicates its communication load. Load balancing in NAMD is measurement-based. This assumes that load patterns tend to persist over time and even if they change, the change is gradual (referred to as the principle of persistence). The load balancing framework records information about object computational loads for some time steps. It also records the communication graph between the patches and proxies. This information is collected on one processor and based on the instrumentation data, a load balancing phase is executed. Decisions are then sent to all processors. The current strategy is centralized and we shall later discuss future work to make it fully...
Automatic Mapping Framework

Identifying regular patterns in communication graphs.

Automatic topology aware mapping, as we shall see in the next few sections, uses heuristics for fast scalable runtime solutions. Heuristics can yield more efficient solutions if we can derive concrete information about the communication graph of the application and exploit it. For this, we need to look for identifiable communication patterns, if any, in the object graph. Many parallel applications have relatively simple and easily identifiable 2D, 3D or 4D communication patterns. If we can identify such patterns, then we can apply better suited heuristic techniques for such patterns.
Automatic Mapping Framework

8.1 Communication Graph: Identifying Patterns

Automatic topology aware mapping, as we shall see in the next few sections, uses heuristics for fast scalable runtime solutions. Heuristics can yield more efficient solutions if we can derive concrete information about the communication graph of the application and exploit it. For this, we need to look for identifiable communication patterns, if any, in the object graph. Many parallel applications have relatively simple and easily identifiable 2D, 3D or 4D communication patterns. If we can identify such patterns, then we can apply better suited heuristic techniques for such.
Automatic Mapping Framework

Figure 8.1: Schematic of the automatic mapping framework

Identifying regular patterns in communication graphs.

Automatic topology aware mapping, as we shall see in the next few sections, uses heuristics for fast scalable runtime solutions. Heuristics can yield more efficient solutions if we can derive concrete information about the communication graph of the application and exploit it. For this, we need to look for identifiable communication patterns, if any, in the object graph. Many parallel applications have relatively simple and easily identifiable 2D, 3D or 4D communication patterns. If we can identify such patterns, then we can apply better suited heuristic techniques for such...
Automatic Mapping Framework

Relieve the application developer of the mapping burden
8.1 Communication Graph: Identifying Patterns

Automatic topology aware mapping, as we shall see in the next few sections, uses heuristics for fast scalable runtime solutions. Heuristics can yield more efficient solutions if we can derive concrete information about the communication graph of the application and exploit it. For this, we need to look for identifiable communication patterns, if any, in the object graph. Many parallel applications have relatively simple and easily identifiable 2D, 3D or 4D communication patterns. If we can identify such patterns, then we can apply better suited heuristic techniques for such patterns.

Relieve the application developer of the mapping burden

No change to the application code
Results: 2D Stencil on Blue Gene/P

Hop-bytes

- Default Mapping
- Topology Mapping

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Hops per byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>5</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
</tr>
<tr>
<td>2048</td>
<td>5</td>
</tr>
<tr>
<td>4096</td>
<td>10</td>
</tr>
<tr>
<td>8192</td>
<td>10</td>
</tr>
<tr>
<td>16384</td>
<td>15</td>
</tr>
</tbody>
</table>
Results: 2D Stencil on Blue Gene/P

**Hop-bytes**

- Default Mapping
- Topology Mapping

**Performance**

- Default Mapping
- Topology Mapping

---

- **Hops per byte**

- **Time per step (ms)**
Increasing communication

- With faster processors and constant link bandwidths
  - computation is becoming cheap
  - communication is a bottleneck
- Trend for bytes per flop
  - XT3: 8.77
  - XT4: 1.357
  - XT5: 0.23

![Graph showing time per step for 2DStencil on BG/P with Default Mapping and Topology Mapping.](image)
Results: WRF on Blue Gene/P

Hops from IBM HPCT

- Default
- Topology
- Lower Bound

Average hops per byte

Number of nodes

- 256
- 512
- 1024
- 2048
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by 64%
  - Time for communication reduces by 11%
  - Performance improves by 17%
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by: 64%
  - Time for communication reduces by 11%
  - Performance improves by 17%

![Graph showing hop reduction and performance improvement](image-url)
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by 64%
  - Time for communication reduces by 11%
  - Performance improves by 17%

Hops from IBM HPCT

- Default
- Topology
- Lower Bound

Average hops per byte

Number of nodes

256 17% 8%
512
1024
2048
Results: POP on Blue Gene/P

<table>
<thead>
<tr>
<th>VN mode</th>
<th>SMP mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZT</td>
<td>XYZ</td>
</tr>
<tr>
<td>TXYZ</td>
<td>TopoMap</td>
</tr>
<tr>
<td>TopoMap</td>
<td>TopoMap</td>
</tr>
</tbody>
</table>

![Bar chart showing average hops per byte for 128 nodes and 512 nodes in VN and SMP modes.](chart.png)
Results: POP on Blue Gene/P

- In VN mode (using all 4 cores per node):
  - Reduction in hops: 60%
  - No improvement in overall performance
Results: POP on Blue Gene/P

- In VN mode (using all 4 cores per node):
  - Reduction in hops: 60%
  - No improvement in overall performance

- In spite of POP spending 55% time in communication
  - MPI_Waitall and MPI_Allreduce

![Bar chart showing average hops per byte for VN and SMP modes across different node counts.]

**VN mode**
- XYZT
- TXYZ
- TopoMap

**SMP mode**
- XYZ
- TopoMap

---

TeraGrid '10 © Abhinav Bhatele

August 3rd, 2010
Summary

• Contention in modern day supercomputers can impact performance: makes mapping important
  • Even for high bandwidth interconnects such as Cray

• Certain classes of applications (latency sensitive, communication bound) benefit most
  • OpenAtom shows performance improvements of up to 50%
  • NAMD - improvements for > 4k cores

• Developing an automatic mapping framework
  • Relieve the application developer of the mapping burden
Questions?

Acknowledgements:

IBM Watson Research Center (Blue Gene/L): Fred Mintzer, Glenn Martyna
Pittsburgh Supercomputing Center (Cray XT3): Chad Vizino, Shawn Brown
Argonne National Laboratory (Blue Gene/P): Pete Beckman, Charles Bacon
Oak Ridge National Laboratory (Cray XT4/5): Donald Frederick, Patrick Worley

Funded in part by the Center for Simulation of Advanced Rockets (Univ. of Illinois) through DOE Grant B341494

E-mail: bhatele@illinois.edu