Automated Mapping of Regular Communication Graphs on Mesh Interconnects

Abhinav Bhatle, Gagan Gupta, Laxmikant V. Kale and I-Hsin Chung

December 20th, 2010
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://oceans11.lanl.gov/twiki/bin/view/Cosim
We want to map communicating objects closer to one another.

http://wrf-model.org/plots/realtime_main.php
http://www.ks.uiuc.edu/Gallery/Science/
http://oceans11.lanl.gov/twiki/bin/view/Cosim
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)
Solution - Mapping Framework

- Input - communication graph of the application and processor topology of the allocated job partition
- Output - mapping of processes/objects to physical processors
- Parallel applications can be classified into:
  - regular/structured: n-dimensional near-neighbor (e.g. POP, WRF)
  - irregular: arbitrary communication
- We focus on regular communication in this paper
Automatic Mapping Framework

Process Topology Analyzer

Input: Application communication graph

Regular Graphs
Irregular Graphs

2D Object Graph
3D Object Graph

Choose best heuristic depending on hop-bytes

Different heuristics for irregular graphs

Output: Mapping file used for the next run
Automatic Mapping Framework

Relieve the application developer of the mapping burden

Input: Application communication graph

Process Topology Analyzer

Regular Graphs

Irregular Graphs

2D Object Graph

3D Object Graph

Choose best heuristic depending on hop-bytes

Different heuristics for irregular graphs

Output: Mapping file used for the next run
Relieve the application developer of the mapping burden

Process Topology Analyzer

Input: Application communication graph

Regular Graphs

2D Object Graph

3D Object Graph

Irregular Graphs

Choose best heuristic depending on hop-bytes

Output: Mapping file used for the next run

No change to the application code

Different heuristics for irregular graphs
Machine 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, jump several hoops to get this information and make it available through the same API
Application communication graph

• Several ways to obtain the graph
• MPI applications:
  • Profiling tools (IBM’s HPCT tools)
  • Collect information using the PMPI interface
  • Manually provided by the application end user
• Charm++ applications:
  • Instrumentation at runtime
  • Profiling tools (HPCT): when n = p
Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

**Input:** $CM_{n,n}$ (communication matrix)

**Output:** $isRegular$ (boolean, true if communication is regular)
- $dims[]$ (dimensions of the regular communication graph)

**Pseudocode for identifying regular communication graphs**

```
for i = 1 to n do
    find the maximum number of neighbors for any rank in $CM_{i,n}$
end for
if max neighbors ≤ 5 then
    // this might be a case of regular 2D communication
    select an arbitrary rank $start_{pe}$ find its distance from its neighbors
    $dist = $ difference between ranks of $start_{pe}$ and its top or bottom neighbor
for i := 1 to n do
    if distance of all ranks from their neighbors == 1 or $dist$ then
        $isRegular = true$
        $dim[0] = dist$
        $dim[1] = n/dist$
    end if
end for
end if
```
Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

**Input:** $CM_{n,n}$ (communication matrix)

**Output:** $isRegular$ (boolean, true if communication is regular)

- $dims[\ ]$ (dimensions of the regular communication graph)

```plaintext
for i = 1 to n do
    find the maximum number of neighbors for any rank in $CM_{i,n}$
end for
if max neighbors $\leq$ 5 then
    // this might be a case of regular 2D communication
    select an arbitrary rank $start_{pe}$ find its distance from its neighbors
    $dist$ = difference between ranks of $start_{pe}$ and its top or bottom neighbor
    for $i := 1$ to $n$ do
        if distance of all ranks from their neighbors == 1 or $dist$ then
            $isRegular = true$
            $dim[0] = dist$
            $dim[1] = n/dist$
        end if
    end for
end if
```

The algorithms for identifying 3D and 4D near-neighbor patterns are similar. Once the information about communicating neighbors has been extracted and identified, mapping algorithms can use it to map communicating neighbors on nearby physical processors.

The pattern matching algorithms were tested with three different applications which are known to have regular communication: MILC, POP, and WRF. The communication patterns and the size of each dimension were correctly identified as shown in Table 8.1.

**Table 8.1: Pattern identification of communication in MILC, POP, and WRF**

<table>
<thead>
<tr>
<th>Application</th>
<th>No. of cores</th>
<th>Dimensionality</th>
<th>Size of dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILC</td>
<td>256</td>
<td>4-dimensional</td>
<td>$4 \times 4 \times 4 \times 4$</td>
</tr>
<tr>
<td>POP</td>
<td>256</td>
<td>2-dimensional</td>
<td>$8 \times 32$</td>
</tr>
<tr>
<td>POP</td>
<td>512</td>
<td>2-dimensional</td>
<td>$32 \times 16$</td>
</tr>
<tr>
<td>WRF</td>
<td>256</td>
<td>2-dimensional</td>
<td>$16 \times 16$</td>
</tr>
<tr>
<td>WRF</td>
<td>512</td>
<td>2-dimensional</td>
<td>$32 \times 32$</td>
</tr>
</tbody>
</table>
Process Topology Discovery

- We want to identify regular 2D/3D communication patterns

**Input:** $CM_{n,n}$ (communication matrix)

**Output:**
- $isRegular$ (boolean, true if communication is regular)
  - dims[] (dimensions of the regular communication graph)

```plaintext
for $i = 1$ to $n$ do
  find the maximum number of neighbors for any rank in $CM_{i,n}$
end for

if max neighbors $\leq 5$ then
  // this might be a case of regular 2D communication
  select an arbitrary rank $start_{pe}$ find its distance from its neighbors
  $dist =$ difference between ranks of $start_{pe}$ and its top or bottom neighbor
  for $i := 1$ to $n$ do
    if distance of all ranks from their neighbors $== 1$ or $dist$ then
      $isRegular = true$
      $dim[0] = dist$
      $dim[1] = n/dist$
    end if
  end for
end if
```
Example

• WRF running on 32 cores of Blue Gene/P
Example

- WRF running on 32 cores of Blue Gene/P

Pattern matching to identify regular communication patterns such as 2D/3D near-neighbor graphs
Example

- WRF running on 32 cores of Blue Gene/P

Pattern matching to identify regular communication patterns such as 2D/3D near-neighbor graphs
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8  
Processor Graph: 12 x 6
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

Object Graph: 9 x 8
Processor Graph: 12 x 6
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)
  - Object Graph: 9 x 8
  - Processor Graph: 12 x 6

- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)

  Object Graph: 9 x 8
  Processor Graph: 12 x 6

- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call

- Expand from Corner (EXCO)
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)
  - Object Graph: 9 x 8
  - Processor Graph: 12 x 6

- Maximum Overlap with Alignment (MXOV+AL)
  - Alignment at each recursive call

- Expand from Corner (EXCO)
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)
  
  Object Graph: 9 x 8  
  Processor Graph: 12 x 6

- Maximum Overlap with Alignment (MXOV+AL)
  
  Alignment at each recursive call

- Expand from Corner (EXCO)
Mapping Regular Graphs (2D)

- Maximum Overlap (MXOVLP)
  Object Graph: 9 x 8
  Processor Graph: 12 x 6

- Maximum Overlap with Alignment (MXOV+AL)
  Alignment at each recursive call

- Expand from Corner (EXCO)
More heuristics ...

- Corners to Center (COCE)
- Start simultaneously from all corners
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners
More heuristics ...

- Corners to Center (COCE)
- Start simultaneously from all corners
More heuristics ...

- Corners to Center (COCE)
- Start simultaneously from all corners
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners

- Affine Mapping (AFFN)
More heuristics ...

- Corners to Center (COCE)
- Start simultaneously from all corners

- Affine Mapping (AFFN)
More heuristics ... 

- **Corners to Center (COCE)**
- **Start simultaneously from all corners**

- **Affine Mapping (AFFN)**

\[
(x, y) \rightarrow (\left\lfloor Px \cdot \frac{x}{Ox} \right\rfloor, \left\lfloor Py \cdot \frac{y}{Oy} \right\rfloor)
\]
More heuristics ...

- Corners to Center (COCE)
  - Start simultaneously from all corners

- Affine Mapping (AFFN)

\[(x, y) \rightarrow ([Px \ast \frac{x}{Ox}], [Py \ast \frac{y}{Oy}])\]
More heuristics ...

- **Corners to Center (COCE)**
  - Start simultaneously from all corners

- **Affine Mapping (AFFN)**

\[(x, y) \rightarrow ([P_x \times \frac{x}{O_x}], [P_y \times \frac{y}{O_y}])\]
Running Time

- Pairwise Exchanges (PAIRS)
  - Bokhari, Lee et al.

![Graph showing Hop per byte vs Time (s) for 4k nodes]
Running Time

- Pairwise Exchanges (PAIRS) - Bokhari, Lee et al.

Hops per byte vs. Time (s)

- Hops for 4k nodes

Time (ms) vs. Number of nodes

- AFFN
- COCE
- MXOVLP
- MXOV+AL
- EXCO

Hops per byte:

- Time (s):
  - 1k
  - 4k
  - 16k
  - 64k

Number of nodes:

- Time (ms):
  - 0.01
  - 0.1
  - 1
  - 10
  - 100
Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Example Mapping

Object Graph: 9 x 8
Processor Graph: 12 x 6

Mapping of 9x8 graph to 12x6 mesh

MXOVLP: 1.66  
MXOV+AL: 1.65  
EXCO: 2.31  
COCE: 1.91
Mapping of 9x8 graph to 12x6 mesh

MXOVL: 1.66
MXOV+AL: 1.65
EXCO: 2.31
COCE: 1.91
Mapping of 9x8 graph to 12x6 mesh

STEP: 1.39

AFFN1: 1.77

AFFN2: 1.53

AFFN3: 1.91
Mapping of 9x8 graph to 12x6 mesh

STEP: 1.39
AFFN1: 1.77
AFFN2: 1.53
AFFN3: 1.91
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

\[ d(e) = \max \{d_i | e_i \in E\} \]
Evaluation

Hops per byte

- MXOVLMP
- MXOV+AL
- EXCO
- COCE
- AFFN
- PAIRS

27x44 to 36x33
~1k nodes

100x40 to 125x32
~4k nodes

128x128 to 512x32
~16k nodes

320x200 to 125x512
~64k nodes
Mapping 2D Graphs to 3D

- Map a two-dimensional object graph to a three-dimensional processor graph
- Divide object graph into subgraphs once each for the number of planes
  - Stacking
  - Folding
- Best 2D to 2D heuristic chosen based on hop-bytes
Results: 2D Stencil on Blue Gene/P

Hop-bytes

- Default Mapping
- Topology Mapping

Hops per byte vs. Number of cores

- 512, 1024, 2048, 4096, 8192, 16384 cores
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>0</td>
</tr>
<tr>
<td>1024</td>
<td>5</td>
</tr>
<tr>
<td>2048</td>
<td>10</td>
</tr>
<tr>
<td>4096</td>
<td>15</td>
</tr>
<tr>
<td>8192</td>
<td>20</td>
</tr>
<tr>
<td>16384</td>
<td>20</td>
</tr>
</tbody>
</table>

### Performance

- **Default Mapping**
- **Topology Mapping**

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Time per step (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>400</td>
</tr>
<tr>
<td>1024</td>
<td>417.5</td>
</tr>
<tr>
<td>2048</td>
<td>435</td>
</tr>
<tr>
<td>4096</td>
<td>452.5</td>
</tr>
<tr>
<td>8192</td>
<td>470</td>
</tr>
<tr>
<td>16384</td>
<td>512</td>
</tr>
</tbody>
</table>

December 20th, 2010

HiPC 2010 © Laxmikant Kale
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

2D Stencil on BG/P (4,096 cores)

- Default Mapping
- Topology Mapping

Time per step (s)

Message size
Results: WRF on Blue Gene/P

Hops from IBM HPCT

- Default
- Topology
- Lower Bound

Average hops per byte vs. Number of nodes:
- 256
- 512
- 1024
- 2048
- 4096
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

Hops from IBM HPCT

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Default</th>
<th>Topology</th>
<th>Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td><img src="256" alt="Default" /></td>
<td><img src="256" alt="Topology" /></td>
<td><img src="256" alt="Lower Bound" /></td>
</tr>
<tr>
<td>512</td>
<td><img src="512" alt="Default" /></td>
<td><img src="512" alt="Topology" /></td>
<td><img src="512" alt="Lower Bound" /></td>
</tr>
<tr>
<td>1024</td>
<td><img src="1024" alt="Default" /></td>
<td><img src="1024" alt="Topology" /></td>
<td><img src="1024" alt="Lower Bound" /></td>
</tr>
<tr>
<td>2048</td>
<td><img src="2048" alt="Default" /></td>
<td><img src="2048" alt="Topology" /></td>
<td><img src="2048" alt="Lower Bound" /></td>
</tr>
<tr>
<td>4096</td>
<td><img src="4096" alt="Default" /></td>
<td><img src="4096" alt="Topology" /></td>
<td><img src="4096" alt="Lower Bound" /></td>
</tr>
</tbody>
</table>

Average hops per byte
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by: 63%
  - 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

256 512 1024 2048 4096
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by: 63%
  - Time for communication reduces by 11%
  - Performance improves by 17%

Hops from IBM HPCT

- Default
- Topology
- Lower Bound

![Average hops per byte](chart)

Number of nodes:
- 256
- 512
- 1024
- 2048
- 4096

December 20th, 2010
Results: WRF on Blue Gene/P

- Performance improvement negligible on 256 and 512 cores

- On 1024 nodes:
  - Hops reduce by 63%
  - 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
- 512
- 1024
- 2048
- 4096

17%
5%
Summary

• Contention in modern day supercomputers can impact performance: makes mapping important

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

• Topology discovery: Topology Manager API

• Object Communication Graph: Profiling, Instrumentation

• Pattern matching: regular and irregular graphs

• Suite of heuristics for mapping
Future Work

- More sophisticated algorithms for process topology discovery and mapping
  - Multicast and many-to-many patterns
- Handling multiple communication graphs
  - Simultaneous or occurring in different phases
- Extension to irregular communication graphs (in progress)
Thanks

Questions?