BlueGene Riddles Resolved

BG/Q Tipps for MPI and OpenMP

Christoph Pospiech
High Performance Computing
Lenovo Deutschland GmbH
Dresden

Contents

- OpenMP Extensions
  - Transactional Memory
  - Thread-level Speculation
- I/O Subsystem
- MPI Task Mapping
- Hybrid Parallelization

LOCKS and Deadlock

Consider the following example

```c
void move(T s, T d, Obj key){
  LOCK(s); LOCK(d);
  tmp = s.remove(key);
  d.insert(key, tmp);
  UNLOCK(d); UNLOCK(s);
}
```

This can deadlock!

Thread 0 move(a, b, key1);
Thread 1 move(b, a, key2);
Transactional Memory (TM)

- Programmer says — “I want this atomic”

```c
void move(T s, T d, Obj key)
{
    #pragma TM_ATOMIC SAFE_MODE {
        tmp = s.remove(key);
        d.insert(key, tmp);
    }
}
```

- TM system — “I'll make it so”
  - Avoids deadlock
  - Replaces fine grain locking

Unleashing TM

- Compiler flag to enable TM pragmas: −qtm
- User needs to add directives
- Use SAFE_MODE if no access to I/O
- Performance governed by tradeoff
  - Execution time of atomic region (entry and exit overhead)
  - Conflict probability (rollback overhead)

Thread-level Speculation

- Similar to Transactional Memory
  (But different usage model)
- Leverages existing OpenMP parallelization
  (However less assumptions for the compiler)
- Subdivision into work units without locking
- If work units collide in memory
  - Hardware detects collision
  - Kernel rolls back transaction
  - Runtime decides whether to retry or serialize

I/O Subsystem

I...Ooh!
**BlueGene/Q I/O**

- I/O Nodes similar to BG/L and BG/P
- All I/O is delegated to an I/O node

**I/O Considerations**

- I/O is delegated to the I/O nodes
- Compute nodes free from I/O load
- Bandwidth to I/O nodes limited by PCI/E
- Beware of I/O only from MPI rank 0
- Uses only one link to I/O nodes
- Doesn’t scale
- Consider MPI I/O

**5-dimensional Torus**

- Given by 5 coordinates ABCDE

<table>
<thead>
<tr>
<th># Node Cards</th>
<th># Nodes</th>
<th>Torus size</th>
<th>IsTorus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>2x2x2x2x2</td>
<td>00001</td>
</tr>
<tr>
<td>2(adjacent)</td>
<td>64</td>
<td>2x2x4x2x2</td>
<td>00101</td>
</tr>
<tr>
<td>4(quadrants)</td>
<td>128</td>
<td>2x2x4x4x2</td>
<td>00111</td>
</tr>
<tr>
<td>8(halves)</td>
<td>256</td>
<td>4x2x4x4x2</td>
<td>10111</td>
</tr>
</tbody>
</table>

- Maximal 10 direct neighbours
- On a midplane ≤ 2 hops in one direction
- Different message travel times

**MPI Task Mapping**

*Nail it down!*
**To Map or Not To Map**

- MPI communication should have
  - a big chunk of the execution time
  - a well understood communication pattern
- Set up MPI task mapping via
  - MPI topology (your mileage may vary)
  - runjob option –mapping
  - Set reorder=false if using both
- Necessary LoadLeveler keywords
  - #bg_shape
  - #bg_rotate = FALSE

---

**Understanding MPI**

Basic features of MPI (ordered by importance)

- Global Communication
- MPI Derived Datatypes
- Communicators
- Point to Point Communication

You should view MPI Global Communication as the "BLAS" routines of Distributed Programming. They offer a "High Level" approach to parallel programming.

---

**Routines for Global Communication (1)**

- broadcast

```
  a b c d
  1 1
  2
  3
  4
```

---

**Routines for Global Communication (2)**

- broadcast
  - gather/scatter

```
  a b c d
  1 1 2 3
  2
  3
  4
```

---

**Routines for Global Communication (1)**

```
  a b c d
  1 1
  2
  3
  4
```

---

**Routines for Global Communication (2)**

```
  a b c d
  1 1
  2
  3
  4
```

---
### Routines for Global Communication (3)

- broadcast
- gather/scatter
- allgather

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Routines for Global Communication (4)

- broadcast
- gather/scatter
- allgather
- reduce/allreduce

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Routines for Global Communication (5)

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

### Routines for Global Communication (6)

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Routines for Global Communication (7)

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>58</td>
<td>58</td>
<td>58</td>
</tr>
</tbody>
</table>
Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all

Think twice before using barrier !!

Routines for Global Communication

- broadcast
- gather/scatter
- allgather
- reduce/allreduce
- reduce and scatter
- scan
- all2all
- sendrecv

Hybrid Parallelization

Example: matrix-vector multiplication
**Choose Data Distribution**

Choose Data Distribution

1 2 3 4 5
1 2 3
1 2 3
1 2 3
1 2 3

Stride 1 access

Access with large stride

**Set Up MPI Communication**

Set Up MPI Communication

Do $j = 1, n_{\text{loc}}$

Do $i = 1, n$

\[ c(i) = c(i) + a(i, j) \times b(j) \]

end Do

end Do

call mpi_reduce_scatter

**SMP Autoparallelization**

SMP Autoparallelization

If this handles the inner loop, then $c$ is shared !!

**Timings on p690**

Timings on p690

<table>
<thead>
<tr>
<th>Threads</th>
<th>MPI</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>110</td>
</tr>
<tr>
<td>120</td>
</tr>
<tr>
<td>130</td>
</tr>
<tr>
<td>140</td>
</tr>
</tbody>
</table>

Add `-qsmp=auto`
Timings on BG/P

Time [s]

Threads 1 2 4 8 16 MPI Tasks

190 171 152 133 114 95 76 57 38 19

BG/Q Timings (SMT)

Time [s]

Threads 1 2 4 8 16 MPI Tasks

12 24 36 48 60 72 84 96 108 120

$OMP do parallel$OMP reduction(+:c)
Do $j = 1, n_{loc}$
    Do $i = 1, n$
        $c(i) = c(i) + a(i, j) * b(j)$
    end Do
end Do
$OMP end do parallel$
call mpi_reduce_scatter

Choose the same loop for SMP and MPI

Timings on p690

Time [s]

Threads 1 2 4 8 16 MPI Tasks

10 20 30 40 50 60 70 80

1 2 4 8 16

Redistribute SMP Work

$OMP do parallel$OMP reduction(+:c)
Do $j = 1, n_{loc}$
    Do $i = 1, n$
        $c(i) = c(i) + a(i, j) * b(j)$
    end Do
end Do
$OMP end do parallel$
call mpi_reduce_scatter

Choose the same loop for SMP and MPI
Timings on jugene

BG/Q Timings (SMT)

Conditions for Hybrid Parallelization
- Hybrid mode includes an MPI program
- SMP on each node has to pay off
  - Easy and effective SMP parallelization
  - MPI communication overhead explodes
  - Escape memory contention
  - Local dynamical load balancing

Mixed mode does not invalidate or escape from Amdahl’s Law!