Performance Analysis on Blue Gene/P

Tulin Kaman

Department of Applied Mathematics and Statistics
Stony Brook University
From microprocessor to the full Blue Gene P/system
IBM XL Compilers

The commands to invoke the compiler

- on **BG/L**: `blrts_xlc, blrts_xlC, blrts_xlf, ..`
- on **BG/P**: `bgxlc, bgxlc++,bgxl f, ..`

How to compile MPI Programs on BG/P

**Option1:** Using `bgx` prefix, the programmer explicitly identifies all the libraries and include files.

```bash
-L/bgsys/drivers/ppcfloor/comm/lib -lmpich.cnk -ldcmfcoll.cnk -ldcmf.cnk
-lpthread -lrt -L/bgsys/drivers/ppcfloor/comm/runtime/SPI -ISPI.cna
```

**Option2:** `mpixlc, mpixlcxx, mpixlf`,… These scripts take care of the proper order of in which libraries should be called.
Shared-memory parallelism

- BG/P system supports shared memory parallelism on single nodes.

- Compiler can automatically locate and countable loop is automatically parallelized if:
  - The order in which loop iterations start and end doesn’t affect the result
  - The loop doesn’t contain I/O operations
  - The code is compiled with a thread-safe version of compiler (_r suffix) `mpixlc_r, mpixlcxx_r, mpixlf77_r`...
  - `-qsmp = auto option is in effect.`
OpenMP pragmas

-qsmp=omp parallelizes based on OpenMP.

```c
#pragma omp parallel private(i,tid)
    
    tid = omp_get_thread_num();
    if (tid == 0)
        nthreads = omp_get_num_threads();
    printf("Thread %d starting...\n",tid);
#pragma omp for
    for (i=0; i<N; i++)
        c[i] = a[i] + b[i];

/* end of parallel section */
```

- For loop the master thread creates additional threads. The loop is executed by all threads.
- Variables are by default shared.
Compiler optimization options

For specific architecture

- BG / L → **two** 32-bit PowerPC440
- BG / P → **four** 32-bit PowerPC450

- **qarch**: generates parallel instruction, double precision floating point multiply add unit (double hummer)
- **qtune**: optimizes object code. Single FPU only.

- O3 -qarch=450d -qtune=450

To list all options in effect, specify the -qlistopt
Mathematical Acceleration Subsystem Libraries (MASS)

- MASS libraries offer improved performance over the standard mathematical library routines, are thread-safe.

- MASS Libraries for Blue Gene v.4.4[^1] is installed to $home$ directory.

- On the linker command it is explicitly specified.
  -L$home/mass/4.4/bgp/lib -lmass

EXECUTION PROCESS MODES

Symmetrical MultiProcessing (SMP) Node Mode

- Each compute node executes a single MPI task per node with a maximum of 4 threads

Virtual Node (VN) Mode

- In this mode, kernel runs four MPI task on each compute node.

Dual Node (DUAL) Mode

- Each compute node executes two MPI tasks per node. Each task in this mode get the half memory. It runs two threads per task.
Specify the mode in the script file

SMP (1x4)
- mode smp

VN (4x1)
- mode vn

DUAL (2x2)
- mode dual
TOTAL PERFORMANCE

Computation
  - Xprofiler
  - HPM

Communication
  - MPI Profiler

I/O
  - MIO Library

High Performance Computing Toolkit (HPCT)
MPI Profiler and Tracer

- Compiling and linking
  - The application must be compiled with the -g option.
  - To link the application with the trace library, those options should be added to the command line.
    -L</path/to/libmpitrace.a> -lmpitrace -llicense
  - -lmpitrace should be before the -lmpich

- Set environment variables

- Outputs
  - Plain text file (mpi_profile.0)
  - The VIZ file  (mpi_profile_0.viz)
  - Trace file    (single_trace)
We will get profile data for rank 0 and ranks with min-max-median MPI communication time.

<table>
<thead>
<tr>
<th>MPI Routine</th>
<th>#calls</th>
<th>avg.bytes</th>
<th>time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_Comm_size</td>
<td>37</td>
<td>0.0</td>
<td>0.000</td>
</tr>
<tr>
<td>MPI_Comm_rank</td>
<td>327</td>
<td>0.0</td>
<td>0.000</td>
</tr>
<tr>
<td>MPI_Bsend</td>
<td>400</td>
<td>22384.9</td>
<td>0.022</td>
</tr>
<tr>
<td>MPI_Recv</td>
<td>400</td>
<td>22384.9</td>
<td>0.123</td>
</tr>
<tr>
<td>MPI_Buffer_attach</td>
<td>1</td>
<td>0.0</td>
<td>0.000</td>
</tr>
<tr>
<td>MPI_Barrier</td>
<td>103</td>
<td>0.0</td>
<td>74.211</td>
</tr>
<tr>
<td>MPI_Allgather</td>
<td>12</td>
<td>40.0</td>
<td>17.557</td>
</tr>
<tr>
<td>MPI_Allreduce</td>
<td>744</td>
<td>15.8</td>
<td>69.269</td>
</tr>
</tbody>
</table>

Total communication time = 161.182 seconds.
Total elapsed time = 313.848 seconds.
mpi_profile.0 contains timing summaries from each tasks.

**Communication summary for all tasks:**

minimum communication time = 43.102 sec for task 351
median communication time = 85.370 sec for task 57
maximum communication time = 169.488 sec for task 7

<table>
<thead>
<tr>
<th>taskid</th>
<th>xcoord</th>
<th>ycoord</th>
<th>zcoord</th>
<th>procid</th>
<th>total_comm(sec)</th>
<th>avg_hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>161.182</td>
<td>1007.00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>153.902</td>
<td>1007.00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130.799</td>
<td>1007.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>141.615</td>
<td>1007.00</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>149.342</td>
<td>1007.00</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>146.275</td>
<td>1007.00</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>146.451</td>
<td>1007.00</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>169.488</td>
<td>1007.00</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>147.396</td>
<td>1007.00</td>
</tr>
</tbody>
</table>
Output II: The VIZ file

Using the IBM HPCT Graphical User Interface (GUI), we can view the profile data presented in the form of an XML file.

- **The Viz file:** \texttt{peekperf mpi\_profile\_0.viz}
- **Trace file:** \texttt{peekview single\_trace}
Xprofiler - CPU Profiling tool

- Xprofiler provides quick access to the profiled data and helps users identify the functions that are the most CPU-intensive.

- Using the GUI, it is very easy to find the application's performance-critical areas.

- Full profiling - call graph info, statement level profiling, basic block profiling, and machine instruction profiling: `-pg -g` to the `compile and link option`

- `Xprofiler <exe_file_name> gmon.out.taskid`
The size of green, solid-filled boxes indicates CPU usage.

The height of the box: time spent on executing itself.

The width of the box: time spent on executing itself and descendent functions.

The number on arrows: call counts.
Hardware Performance Monitoring (HPM)

- Provides comprehensive reports of hardware events that are critical to performance on BG/P system.

- Hardware performance metrics:
  - the number of misses on all cache levels
  - the number of floating point instructions executed

- Libhpm is the library that provides a programming interface to start, stop and print performance counting.
Libhpm Library

- Initialize the Performance Counter unit
- Start counting in a block marked by the label
- Stop counting in a block marked by the label
- Print counters for all blocks

```
HPM_Init(void);
HPM_Start(“WorkC”);
    do_work;
HPM_Stop(“WorkC”);
HPM_Print();
```
Hardware counter report for BGP node 0, coordinates <0,0,0>. BGP counter mode = 0, trigger = level high.

<table>
<thead>
<tr>
<th>Work</th>
<th>Call Count</th>
<th>Counter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3403277641</td>
<td>BGP_PU0_JPIPE_INSTRUCTIONS</td>
</tr>
<tr>
<td>1</td>
<td>94374245446</td>
<td>BGP_PU0_JPIPE_ADD_SUB</td>
</tr>
<tr>
<td>2</td>
<td>13245536311</td>
<td>BGP_PU0_JPIPE_LOGICAL_OPS</td>
</tr>
<tr>
<td>3</td>
<td>12815801883</td>
<td>BGP_PU0_JPIPE_SHROTMK</td>
</tr>
<tr>
<td>13</td>
<td>30841</td>
<td>BGP_PU0_DCACHE_LINEFILLINPROG</td>
</tr>
<tr>
<td>14</td>
<td>77873</td>
<td>BGP_PU0_ICACHE_LINEFILLINPROG</td>
</tr>
<tr>
<td>15</td>
<td>42458</td>
<td>BGP_PU0_DCACHE_MISS</td>
</tr>
<tr>
<td>16</td>
<td>18549118224</td>
<td>BGP_PU0_DCACHE_HIT</td>
</tr>
<tr>
<td>20</td>
<td>31186315</td>
<td>BGP_PU0_ICACHE_MISS</td>
</tr>
<tr>
<td>21</td>
<td>14838898642</td>
<td>BGP_PU0_ICACHE_HIT</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>BGP_PU0_FPU_ADD_SUB_1</td>
</tr>
<tr>
<td>23</td>
<td>4932</td>
<td>BGP_PU0_FPU_MULT_1</td>
</tr>
</tbody>
</table>
Tuning and Analysis Utilities (TAU)

http://www.cs.uoregon.edu/research/tau/home.php
Performance Analysis Tool

TAU provides a suite of static and dynamic tools that provide graphical user interaction and interoperation to form an integrated analysis environment for parallel applications.
Some definitions

 Scaling studies involve changing the degree of parallelism.

- **Strong Scaling**

- **Weak Scaling**

**Speed Up**

\[ T_s / T_p(n) \]

**Efficiency**

\[ T_s / (n \times T_p(n)) \]
Fixed problem size, more processor

Strong Scaling on New York Blue BG/P System
Amount of work: 100 time steps

Wallclock Time (seconds)

Number of Processors

512 x 512
1024 x 1024
2048 x 2048
4096 x 4096
Problem size grows with more processor

Weak Scaling on New York Blue BG/P System
Amount of Work = 100 time steps

Wallclock Time (seconds) vs. Number of Processors
New York Blue

**BG/L System**
- 18 racks
- 18432 compute nodes (36864 CPUs)

**BG/P System**
- 2 racks
- 2048 compute nodes (8192 CPUs)
Argonne Leadership Computing Facility (ALCF)

BG/P Surveyor System
13.6 TF/s 1 rack BG/P
1024 compute nodes
(4096 CPUs)

BG/P Intrepid System
557.1 TF/s 40 rack BG/P
40960 compute nodes
(163840 CPUs)
IBM Redbooks:

- IBM System Blue Gene Solution: Blue Gene/P Application Development
- IBM System Blue Gene Solution: High Performance Computing Toolkit for Blue Gene/P
- IBM XL C/C++Advanced Edition for Linux, 9.0 Compiler Reference

Advanced Computing Technology Center
http://www.ibm.com/research/actc/

New York Blue
http://www.bnl.gov/newyorkblue/

Argonne Leadership Computing Facility
http://www.alcf.anl.gov/