The IBM High Performance Computing Toolkit

David Klepacki
Advanced Computing Technology
IBM T.J. Watson Research Center

March 2004
What’s New?

- One consolidated package
  “IBM High Performance Computing Toolkit”
- New Software:
  Watson Sparse Matrix Library (WSMP) now included
  http://www-users.cs.umn.edu/~agupta/wsmp.html
  Modular I/O Performance Tool (MIO)
  MPI Tracer
  SHMEM Profiler
  GUI integration tool w/ source code traceback (PeekPerf)
- New Installation Method
  RPM modules – allows query to existence and version
IBM High Performance Toolkit Software (IBM pSeries / AIX)

- **Hardware Performance**
  - HPM Toolkit
    - Catch
    - hpmcount
    - libhpm
    - hpmstat
- **Shared Memory Performance**
  - DPOMP
    - pomprof
- **Message-Passing Performance**
  - MP_Profiler, MP_Tracer
    - TurboSHMEM, TurboMPI
- **Memory Performance**
  - Sigma
- **Performance Visualization**
  - PeekPerf
- **I/O Performance**
  - MIO (modular I/O)
- **Math Performance**
  - WSMP (Watson sparse matrix)
Unified GUI with Source Code Traceback

HPM  MP_profiler/tracer  PomProf  SiGMA  MIO

PeekPerf

(work in progress)
Simultaneous Visualization (HPM+MPI+OpenMP+MIO)

```
Label      | Count | ExcSec | IncSec
----------|-------|--------|--------
BRNCHX     | 1     | 62.664 | 62.66  
Games Main | 1     | 0.037  | 62.71  
LMOINF     | 1     | 0      | 0      
LMOX       | 1     | 0      | 0      
TRUNC      | 1     | 0.013  | 0.013  

VA7FM (END) | 1     | 0      | 0      
VA7FM (INIT)| 1     | 0      | 0      

```

```
call f_hpmstop(13)
ENDIF
IF(RPAC .AND. EXETYP.NE.CHECK) THEN
    call f_hpmstart(14, "RPAC")
    CALL RPAC
    call f_hpmstop(14)
ENDIF
IF(FRIEND.NE.BLANK) CALL ZEALX
C
C
C
C
C

```

```
call f_hpmstart(9, "VA7FM (END)")
CALL VA7FM(LASTFM)
call f_hpmstop(9)
IF(LASTFM.NE.INITFM .AND. MASWRK) WRITE(IW,S50) 0
```

```
   0  0  1  4.909  0  0  2.471  0.035  0  0  0  0
   1  0  1  1.403  0  0  2.8  0.137  0  0  0  0
```

```
```

---

The IBM High Performance Computing Toolkit © 2004 IBM Corporation
HPM Visualization Using PeekPerf

The image shows a graphical user interface for visualizing high-performance computing metrics. The interface includes tabs and a main window displaying performance data for a particular loop (Loop 300) with labels such as "swim_comp.f", "calc1.f", "calc2.f", and "calc3.f". The interface also includes a metric browser and option menu with various performance counters and options for analyzing the data.
Message-Passing Performance: MP_Profiler and MP_Tracer Libraries

- **MP_Profiler**
  - Captures “summary” data for MPI and SHMEM calls with source code traceback
  - No changes to source code, but MUST compile with `-g`
  - ~1.7 microsecond overhead per MPI / SHMEM call
  - Several libraries needed:
    - mpiprof, smaprof, turbo1prof, turbo2prof

- **MP_Tracer**
  - Captures “timestamped” data for MPI and SHMEM calls with source traceback
  - ~1.4 microsecond overhead per MPI / SHMEM call
  - More libraries needed:
    - mpitrace, smatrace, turbo1trace, turbo2trace
### MP_Profiler Summary Output

<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>3</td>
<td>0.0</td>
<td>0.000</td>
</tr>
<tr>
<td>MPI_Comm_rank</td>
<td>12994</td>
<td>0.0</td>
<td>0.016</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>19575</td>
<td>11166.9</td>
<td>13.490</td>
</tr>
<tr>
<td>MPI_Isend</td>
<td>910791</td>
<td>5804.2</td>
<td>9.216</td>
</tr>
<tr>
<td>MPI_Recv</td>
<td>138173</td>
<td>2767.9</td>
<td>73.835</td>
</tr>
<tr>
<td>MPI_Irecv</td>
<td>784936</td>
<td>15891.6</td>
<td>2.407</td>
</tr>
<tr>
<td>MPI_Sendrecv</td>
<td>894809</td>
<td>352.0</td>
<td>88.705</td>
</tr>
<tr>
<td>MPI_Wait</td>
<td>1537375</td>
<td>0.0</td>
<td>288.049</td>
</tr>
<tr>
<td>MPI_Waitall</td>
<td>44042</td>
<td>0.0</td>
<td>25.312</td>
</tr>
<tr>
<td>MPI_Bcast</td>
<td>464</td>
<td>41936.8</td>
<td>3.272</td>
</tr>
<tr>
<td>MPI_BARRIER</td>
<td>1312</td>
<td>0.0</td>
<td>34.206</td>
</tr>
<tr>
<td>MPI_GATHER</td>
<td>68</td>
<td>16399.1</td>
<td>2.680</td>
</tr>
<tr>
<td>MPI_SCATTER</td>
<td>6</td>
<td>17237.3</td>
<td>0.532</td>
</tr>
</tbody>
</table>

---

total communication time = 770.424 seconds.
total elapsed time    = 1168.662 seconds.
user cpu time          = 1160.960 seconds.
system time            = 0.620 seconds.
maximum memory size    = 68364 KBytes.

To check load balance: grep "total comm" mpi_profile.*
MP_Profiler Visualization Using PeekPerf

The data shows various MPI calls with their respective call counts. For example, `MPI_Allreduce_807` has a call count of 16, and `MPI_Bcast_924` has a call count of 285.

The code snippet includes comments indicating that it processes only the total number of processes and imports necessary headers.

```
OP COMPILE
C PROCESSES ONLY, NOT THE TOTAL NUMBER OF PROCESSES.
C
C IMPLICIT NONE
INTEGER DDI_NP, DDI_ME
C
INCLUDE 'mpif.h'
```
SHMEM Profiling Capability

<table>
<thead>
<tr>
<th>Label</th>
<th>Call Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_pe_945</td>
<td>2</td>
</tr>
<tr>
<td>num_pes_944</td>
<td>2</td>
</tr>
<tr>
<td>-shmem_barrier_all_788</td>
<td>431</td>
</tr>
<tr>
<td>-shmem_barrier_all_799</td>
<td>431</td>
</tr>
<tr>
<td>-shmem_barrier_all_859</td>
<td>186</td>
</tr>
<tr>
<td>-shmem_barrier_all_871</td>
<td>186</td>
</tr>
<tr>
<td>-shmem_barrier_all_915</td>
<td>114</td>
</tr>
<tr>
<td>-shmem_barrier_all_927</td>
<td>114</td>
</tr>
<tr>
<td>shmem_broadcast8_796</td>
<td>273</td>
</tr>
<tr>
<td>shmem_int8_sum_to_all_923</td>
<td>114</td>
</tr>
</tbody>
</table>

```
WPR_START = PE_START
WLOG_STRIDE = LOG_STRIDE
WPR_SIZE = PR_SIZE

CALL SHMEM_BROADCAST8 (TARGET, SOURCE, WLENGTH)
```

**Metric Browser: shmem_broadcast8_796**

<table>
<thead>
<tr>
<th>Task</th>
<th>Message Size</th>
<th>Call Count [Max]</th>
<th>WallClock [Max]</th>
<th>Transferred Bytes</th>
<th>Count</th>
<th>WallClock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(2) 5 ... 16</td>
<td>49</td>
<td>0.040212</td>
<td>392</td>
<td>49</td>
<td>0.040212</td>
</tr>
<tr>
<td>0</td>
<td>(3) 17 ... 64</td>
<td>7</td>
<td>0.002289</td>
<td>168</td>
<td>7</td>
<td>0.002289</td>
</tr>
<tr>
<td>0</td>
<td>(4) 65 ... 256</td>
<td>48</td>
<td>0.015485</td>
<td>3192</td>
<td>48</td>
<td>0.015485</td>
</tr>
<tr>
<td>0</td>
<td>(5) 257 ... 1K</td>
<td>273</td>
<td>0.105899</td>
<td>76440</td>
<td>273</td>
<td>0.105899</td>
</tr>
<tr>
<td>0</td>
<td>(6) 1K ... 4K</td>
<td>29</td>
<td>0.019601</td>
<td>20888</td>
<td>29</td>
<td>0.019601</td>
</tr>
<tr>
<td>0</td>
<td>(7) 4K ... 16K</td>
<td>25</td>
<td>0.008123</td>
<td>162816</td>
<td>25</td>
<td>0.008123</td>
</tr>
<tr>
<td>1</td>
<td>(2) 5 ... 16</td>
<td>49</td>
<td>0.020304</td>
<td>392</td>
<td>49</td>
<td>0.020304</td>
</tr>
<tr>
<td>1</td>
<td>(3) 17 ... 64</td>
<td>7</td>
<td>0.003646</td>
<td>168</td>
<td>7</td>
<td>0.003646</td>
</tr>
<tr>
<td>1</td>
<td>(4) 65 ... 256</td>
<td>48</td>
<td>0.029487</td>
<td>3192</td>
<td>48</td>
<td>0.029487</td>
</tr>
<tr>
<td>1</td>
<td>(5) 257 ... 1K</td>
<td>273</td>
<td>0.142042</td>
<td>76440</td>
<td>273</td>
<td>0.142042</td>
</tr>
<tr>
<td>1</td>
<td>(6) 1K ... 4K</td>
<td>29</td>
<td>0.184179</td>
<td>20888</td>
<td>29</td>
<td>0.184179</td>
</tr>
<tr>
<td>1</td>
<td>(7) 4K ... 16K</td>
<td>25</td>
<td>0.015682</td>
<td>162816</td>
<td>25</td>
<td>0.015682</td>
</tr>
</tbody>
</table>
MP_Tracer Visualization Using PeekPerf

Identifier

- MPI_Comm_size
- MPI_Comm_rank
- MPI_Bcast
- MPI_Barrier
- MPI_Recv
- MPI_Send
- MPI_All_reduce
Performance of OpenMP:  POMP

- Portable cross-platform/cross-language API to simplify the design and implementation of OpenMP tools
- POMP was motivated by the MPI profiling interface (PMPI)
  PMPI allows selective replacement of MPI routines at link time
  Used by most MPI performance tools (including MP-Profiler)
POMP Proposal

- Three groups of events
  - **OpenMP constructs and directives/pragmas**
    - Enter/Exit around each OpenMP construct
    - Begin/End around associated body
    - Special case for parallel loops:
      - ChunkBegin/End, IterBegin/End, or IterEvent instead of Begin/End
    - “Single” events for small constructs like atomic or flush
  - **OpenMP API calls**
    - Enter/Exit events around omp_set_*_lock() functions
    - “Single” events for all API functions
  - **User functions and regions**
    - Allows application programmers to specify and control amount of instrumentation
POMP Profiler (PompProf)

- Generates a detailed profile describing overheads and time spent by each thread in three key regions of the parallel application:
  - Parallel regions
  - OpenMP loops inside a parallel region
  - User defined functions

- Profile data is presented in the form of an XML file that can be visualized with PeekPerf
# PompProf Visualization Using PeekPerf

![PompProf Visualization Using PeekPerf](image)

### Metrics

<table>
<thead>
<tr>
<th>Task</th>
<th>Thread ID</th>
<th>Master Time</th>
<th>Thread Time</th>
<th>Computation Time</th>
<th>Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.109860</td>
<td>0.013279</td>
<td>0.003797</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.063225</td>
<td>0.012801</td>
<td>0.015644</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.08116</td>
<td>0.023186</td>
<td>0.028004</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.023728</td>
<td>0.013003</td>
<td>0.039373</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.054689</td>
<td>0.013006</td>
<td>0.031863</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.023176</td>
<td>0.013514</td>
<td>0.015503</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.069733</td>
<td>0.012593</td>
<td>0.03670</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.07246</td>
<td>0.013033</td>
<td>0.039173</td>
<td></td>
</tr>
</tbody>
</table>

### Code Snippet

```c
DO 100 i=1,nX
   C(i)= C(i)+p(i+1)*(|V(i)|+|V(i+1)|)
   if (f(i)) then
      C(i+1)= C(i+1)-|V(i+1)|+|V(i+1)|
      1
   endif
   if (f(i+1)) then
      C(i+1)= C(i+1)-|V(i+1)|+|V(i+1)|
      1
   endif
   if (f(i+2)) then
      C(i+1)= C(i+1)-|V(i+1)|+|V(i+1)|
      1
   endif
   if (f(i+3)) then
      C(i+1)= C(i+1)-|V(i+1)|+|V(i+1)|
      1
   endif
1 CONTINUE
```

# The IBM High Performance Computing Toolkit

© 2004 IBM Corporation
TurboMP Libraries (latest version: 3.0.2)

- **TurboMPI-1 Libraries**
  - Collective Communications Enhanced for shared memory
    - AllReduce, Alltoall, Alltoallv, Bcast, Barrier, Reduce
  - No LAPI dependency (to accommodate pre-Federation switched systems)
  - 32-bit and 64-bit support, Fortran/C/C++
  - Syntax: -lturbo1

- **TurboMPI-2 Libraries**
  - All of MPI-1 above, plus MPI-2 RMA operations
    - Put, Get, Accumulate, Fence, Lock/Unlock, Start/Post/Wait/Complete, Test
  - Syntax: -lturbo2

- **TurboSHMEM Libraries**
  - Complete implementation (400+ routines)
  - 32-bit and 64-bit support, Fortran/C/C++
  - Syntax: -lsma (requires shmem_init() / shmem_finalize() in source)
Turbo MPI_Put Comparison Synchronization (SMP)
(Power 3 Data On-Node)

Bandwidth Vs. Message-Size

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Turbo Fence</th>
<th>Turbo Lock Exclusive</th>
<th>Turbo Lock Shared</th>
<th>Turbo Start…Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.224</td>
<td>0.114</td>
<td>0.271</td>
<td>0.404</td>
</tr>
<tr>
<td>8</td>
<td>0.439</td>
<td>0.227</td>
<td>0.547</td>
<td>0.768</td>
</tr>
<tr>
<td>16</td>
<td>0.882</td>
<td>0.453</td>
<td>1.097</td>
<td>1.534</td>
</tr>
<tr>
<td>32</td>
<td>1.74</td>
<td>0.904</td>
<td>2.187</td>
<td>3.242</td>
</tr>
<tr>
<td>64</td>
<td>3.652</td>
<td>1.81</td>
<td>4.357</td>
<td>6.084</td>
</tr>
<tr>
<td>128</td>
<td>7.064</td>
<td>3.615</td>
<td>8.667</td>
<td>12.44</td>
</tr>
<tr>
<td>256</td>
<td>14.33</td>
<td>7.226</td>
<td>17.26</td>
<td>24.21</td>
</tr>
<tr>
<td>512</td>
<td>28.77</td>
<td>14.39</td>
<td>34</td>
<td>45</td>
</tr>
<tr>
<td>1K</td>
<td>51.9</td>
<td>28.39</td>
<td>66.2</td>
<td>90.14</td>
</tr>
</tbody>
</table>
Turbo MPI_Put Comparison Synchronization (SMP)  
(Power 3 Data On-Node)

Bandwidth Vs. Message-Size

MB / sec

<table>
<thead>
<tr>
<th>Bytes</th>
<th>1K</th>
<th>2K</th>
<th>4K</th>
<th>8K</th>
<th>16K</th>
<th>32K</th>
<th>64K</th>
<th>128 K</th>
<th>256 K</th>
<th>512 K</th>
<th>1M</th>
<th>2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo Fence</td>
<td>51.9</td>
<td>89.2</td>
<td>171</td>
<td>263</td>
<td>232</td>
<td>432</td>
<td>727</td>
<td>1086</td>
<td>1424</td>
<td>1361</td>
<td>1377</td>
<td>1183</td>
</tr>
<tr>
<td>Turbo Lock Exclusive</td>
<td>28.4</td>
<td>56.1</td>
<td>108</td>
<td>201</td>
<td>212</td>
<td>406</td>
<td>747</td>
<td>1261</td>
<td>1918</td>
<td>2059</td>
<td>1308</td>
<td>1210</td>
</tr>
<tr>
<td>Turbo Lock Shared</td>
<td>66.2</td>
<td>128</td>
<td>234</td>
<td>397</td>
<td>294</td>
<td>553</td>
<td>988</td>
<td>1623</td>
<td>2292</td>
<td>2352</td>
<td>1350</td>
<td>1230</td>
</tr>
<tr>
<td>Turbo Start…Wait</td>
<td>90.1</td>
<td>186</td>
<td>336</td>
<td>540</td>
<td>285</td>
<td>507</td>
<td>858</td>
<td>1238</td>
<td>1586</td>
<td>1564</td>
<td>1217</td>
<td>1277</td>
</tr>
</tbody>
</table>
### ACTC Turbo MPI_Put Latency Comparison
(Power 3 Data On-Node)

<table>
<thead>
<tr>
<th></th>
<th>IBM MPI_Put (usec)</th>
<th>Turbo MPI_Put (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fence</td>
<td>62</td>
<td>18</td>
</tr>
<tr>
<td>Lock - Exclusive</td>
<td>179</td>
<td>35</td>
</tr>
<tr>
<td>Lock Shared</td>
<td>174</td>
<td>15</td>
</tr>
<tr>
<td>Start/Post/Wait/Complete</td>
<td>99</td>
<td>21</td>
</tr>
<tr>
<td>TurboSHMEM Put</td>
<td>N/A</td>
<td>5</td>
</tr>
<tr>
<td>ISend/IRecv</td>
<td>11</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Modular I/O Performance Tool (MIO)

- **I/O Analysis**
  - Trace module
  - Summary of File I/O Activity + Binary Events File
  - Low CPU overhead
- **I/O Performance Enhancement Library**
  - Prefetch module (optimizes asynchronous prefetch and write-behind)
  - System Buffer Bypass capability
  - User controlled pages (size and number)
- **Recoverable Error Handling**
  - Recover module (monitors return values and error + reissues failed requests)
- **Remote Data Server**
  - Remote module (simple socket protocol for moving data)
- **Shared object library for AIX**
MIO User Code Interface

```c
#define open64(a,b,c) MIO_open64(a,b,c,0)
#define read MIO_read
#define write MIO_write
#define close MIO_close
#define lseek64 MIO_lseek64
#define fcntl MIO_fcntl
#define ftruncate64 MIO_ftruncate64
#define fstat64 MIO_fstat64
```
MIO Trace Module (sample partial output)

Trace close : program <-> pf : /bmwfs/cdh108.T20536_13.SCR300 :
(281946/2162.61)=130.37 mbytes/s
  current size=0   max_size=16277
  mode =0777  sector size=4096
  oflags =0x302=RDWR  CREAT  TRUNC
  open             1     0.01
  write          478193   462.10      59774      59774     131072 131072
  read          1777376  1700.48     222172     222172     131072 131072
  seek           911572     2.83
  fcntl            3     0.00
  trunc            16     0.40
  close            1     0.03
  size           127787
Performance Visualization (work in progress)

JFS performance
vmtune -p20 -P80 -f120 -F128 -r2 -R8

writes
reads

file position (bytes)

0 2+3 4+3 6+3 8+3 10+3 12+3 14+3 16+3
0 2+3 4+3 6+3 8+3 10+3 12+3 14+3 16+3

4500 15500

(time (seconds))

slope=115.6+6
MSC.Nastran V2001

Benchmark:
SOL 111, 1.7M DOF, 1578 modes, 146 frequencies, residual flexibility and acoustics. 120 GB of disk space.

Machine:
4-way, 1.3 GHz p655, 32 GB with 16 GB large pages, JFS striped on 16 SCSI disks.

MSC.Nastran:
V2001.0.9 with large pages, dmp=2 parallel=2 mem=700mb
The run with MIO used mio=1000mb

6.8 TB of I/O in 26666 seconds is an average of about 250 MB/sec
ABAQUS Standard v6.3-4

Engine models
Parallel direct solver
16 POWER4 processors

Elapsed Time (seconds)

- 5 M dof, 36 GB fct file
- 11.5 M dof, 80 GB fct file

- with MIO
- w/o MIO
How to Obtain the IBM HPC Toolkit?

- **Current Possibilities:**
  1. Acquire as part of new procurement.
  2. Acquire as part of ACTC Performance Tuning Workshop.
  3. Purchase license directly from IBM Research

- **Future Possibility:**
  Purchase from third-party vendor (e.g., Absoft)
Why a Fee?

- In previous years, ACTC tools costs were factored into pricing of pSeries systems. Today, Research software is no longer accounted for in system pricing, and costs must be accounted for directly through licensing. The result is a loss of transparency regarding the costs of these tools.
IBM HPC Toolkit Software Maintenance Cycle

- Major releases at SciComp meetings (twice a year)
- Minor releases (bug fixes) as-needed
- “New Technology” (e.g., DPOMP) placed on Alphaworks server:
  - http://alphaworks.ibm.com
  - 6 month lifetime
  - 90-day evaluation licenses