MPI Tool Interfaces
A role model for other standards!?

Martin Schulz
Lawrence Livermore National Laboratory
The MPI 1.0 Team Had a Lot of Foresight

People using MPI might care about performance
  – After all, it’s called High Performance Computing

Hence, people may want to measure performance
  – Communication & synchronization is wasted time for computation
  – Want to measure how much we waste

Why not add an interface
to MPI to enable this?
  – Sounds trivial, right?

Still today very uncommon!
The MPI Profiling Interface

Simple support for interception of all MPI calls
- Enforced throughout the whole standard
- Coupled with name shifted interface

Easy to implement profiling tools
- Start timer on entry of MPI routine
- Stop timer on exit of MPI routine
The mpiP tool: Example of the Intended Effect

Intercepts all MPI API calls using PMPI
- Records number of invocations
- Measures time spent during MPI function execution
- Gathers data on communication volume
- Aggregates statistics over time

Several analysis options
- Multiple aggregations options/granularity
  - By function name or type
  - By source code location (call stack)
  - By process rank
- Adjustment of reporting volume
- Adjustment of call stack depth that is considered

Provides easy to use reports

http://mpip.sourceforge.net/
The mpiP tool: Example of the Intended Effect

bash-3.2$ srun --n4 smg2000
mpiP:
mpiP:
mpiP: mpiP V3.1.2 (Build Dec 16 2008/17:31:26)
mpiP: Direct questions and errors to mpiP-help@lists.sourceforge.net
mpiP: Running with these driver parameters:
(nx, ny, nz) = (60, 60, 60)
(Px, Py, Pz) = (4, 1, 1)
(bx, by, bz) = (1, 1, 1)
(cx, cy, cz) = (1.000000, 1.000000, 1.000000)
(n_pre, n_post) = (1, 1)
dim = 3
solver ID = 0
============================================= Struct Interface:  
============================================= Struct Interface:
wall clock time = 0.075800 seconds
cpu clock time = 0.080000 seconds
mpiP:
mpiP: Storing mpiP output in [./smg2000-p.4.11612.1.mpiP].
mpiP:
bash-3.2$
mpiP 101 / Output – Metadata

@ mpiP
@ Command : ./smg2000-p -n 60 60 60
@ Version : 3.1.2
@ MPIP Build date : Dec 16 2008, 17:31:26
@ Start time : 2009 09 19 20:38:50
@ Stop time : 2009 09 19 20:39:00
@ Timer Used : gettimeofday
@ MPIP env var : [null]
@ Collector Rank : 0
@ Collector PID : 11612
@ Final Output Dir : .
@ Report generation : Collective
@ MPI Task Assignment : 0 hera27
@ MPI Task Assignment : 1 hera27
@ MPI Task Assignment : 2 hera31
@ MPI Task Assignment : 3 hera31
### mpiP 101 / Output – Overview

<table>
<thead>
<tr>
<th>Task</th>
<th>AppTime</th>
<th>MPITime</th>
<th>MPI%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9.78</td>
<td>1.97</td>
<td>20.12</td>
</tr>
<tr>
<td>1</td>
<td>9.8</td>
<td>1.95</td>
<td>19.93</td>
</tr>
<tr>
<td>2</td>
<td>9.8</td>
<td>1.87</td>
<td>19.12</td>
</tr>
<tr>
<td>3</td>
<td>9.77</td>
<td>2.15</td>
<td>21.99</td>
</tr>
<tr>
<td>*</td>
<td>39.1</td>
<td>7.94</td>
<td>20.29</td>
</tr>
</tbody>
</table>
## mpiP 101 / Output – Callsites

<table>
<thead>
<tr>
<th>ID</th>
<th>Lev</th>
<th>File/Address</th>
<th>Line</th>
<th>Parent_Funct</th>
<th>MPI_Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>communication.c</td>
<td>1405</td>
<td>hypre_CommPkgUnCommit</td>
<td>Type_free</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>timing.c</td>
<td>419</td>
<td>hypre_PrintTiming</td>
<td>Allreduce</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>communication.c</td>
<td>492</td>
<td>hypre.InitializeCommunication</td>
<td>Isend</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>struct_innerprod.c</td>
<td>107</td>
<td>hypre_StructInnerProd</td>
<td>Allreduce</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>timing.c</td>
<td>421</td>
<td>hypre_PrintTiming</td>
<td>Allreduce</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>coarsen.c</td>
<td>542</td>
<td>hypre_StructCoarsen</td>
<td>Waitall</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>coarsen.c</td>
<td>534</td>
<td>hypre_StructCoarsen</td>
<td>Isend</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>communication.c</td>
<td>1552</td>
<td>hypre_CommTypeEntryBuildMPI</td>
<td>Type_free</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>communication.c</td>
<td>1491</td>
<td>hypre_CommTypeBuildMPI</td>
<td>Type_free</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>communication.c</td>
<td>667</td>
<td>hypre_FinalizeCommunication</td>
<td>Waitall</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>smg2000.c</td>
<td>231</td>
<td>main</td>
<td>Barrier</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>coarsen.c</td>
<td>491</td>
<td>hypre_StructCoarsen</td>
<td>Waitall</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>coarsen.c</td>
<td>551</td>
<td>hypre_StructCoarsen</td>
<td>Waitall</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>coarsen.c</td>
<td>509</td>
<td>hypre_StructCoarsen</td>
<td>Irecv</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>communication.c</td>
<td>1561</td>
<td>hypre_CommTypeEntryBuildMPI</td>
<td>Type_free</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>struct_grid.c</td>
<td>366</td>
<td>hypre_GatherAllBoxes</td>
<td>Allgather</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>communication.c</td>
<td>1487</td>
<td>hypre_CommTypeBuildMPI</td>
<td>Type_commit</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>coarsen.c</td>
<td>497</td>
<td>hypre_StructCoarsen</td>
<td>Waitall</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>coarsen.c</td>
<td>469</td>
<td>hypre_StructCoarsen</td>
<td>Irecv</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>communication.c</td>
<td>1413</td>
<td>hypre_CommPkgUnCommit</td>
<td>Type_free</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>coarsen.c</td>
<td>483</td>
<td>hypre_StructCoarsen</td>
<td>Isend</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>struct_grid.c</td>
<td>395</td>
<td>hypre_GatherAllBoxes</td>
<td>Allgatherv</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>communication.c</td>
<td>485</td>
<td>hypre.InitializeCommunication</td>
<td>Irecv</td>
</tr>
</tbody>
</table>
mpiP 101 / Output – per Function Timing

--- Aggregate Time (top twenty, descending, milliseconds) ---

<table>
<thead>
<tr>
<th>Call</th>
<th>Site</th>
<th>Time</th>
<th>App%</th>
<th>MPI%</th>
<th>COV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waitall</td>
<td>10</td>
<td>4.4e+03</td>
<td>11.24</td>
<td>55.40</td>
<td>0.32</td>
</tr>
<tr>
<td>Isend</td>
<td>3</td>
<td>1.69e+03</td>
<td>4.31</td>
<td>21.24</td>
<td>0.34</td>
</tr>
<tr>
<td>Irecv</td>
<td>23</td>
<td>980</td>
<td>2.50</td>
<td>12.34</td>
<td>0.36</td>
</tr>
<tr>
<td>Waitall</td>
<td>12</td>
<td>137</td>
<td>0.35</td>
<td>1.72</td>
<td>0.71</td>
</tr>
<tr>
<td>Type_commit</td>
<td>17</td>
<td>103</td>
<td>0.26</td>
<td>1.29</td>
<td>0.36</td>
</tr>
<tr>
<td>Type_free</td>
<td>9</td>
<td>99.4</td>
<td>0.25</td>
<td>1.25</td>
<td>0.36</td>
</tr>
<tr>
<td>Waitall</td>
<td>6</td>
<td>81.7</td>
<td>0.21</td>
<td>1.03</td>
<td>0.70</td>
</tr>
<tr>
<td>Type_free</td>
<td>15</td>
<td>79.3</td>
<td>0.20</td>
<td>1.00</td>
<td>0.36</td>
</tr>
<tr>
<td>Type_free</td>
<td>1</td>
<td>67.9</td>
<td>0.17</td>
<td>0.85</td>
<td>0.35</td>
</tr>
<tr>
<td>Type_free</td>
<td>20</td>
<td>63.8</td>
<td>0.16</td>
<td>0.80</td>
<td>0.35</td>
</tr>
<tr>
<td>Isend</td>
<td>21</td>
<td>57</td>
<td>0.15</td>
<td>0.72</td>
<td>0.20</td>
</tr>
<tr>
<td>Isend</td>
<td>7</td>
<td>48.6</td>
<td>0.12</td>
<td>0.61</td>
<td>0.37</td>
</tr>
<tr>
<td>Type_free</td>
<td>8</td>
<td>29.3</td>
<td>0.07</td>
<td>0.37</td>
<td>0.37</td>
</tr>
<tr>
<td>Irecv</td>
<td>19</td>
<td>27.8</td>
<td>0.07</td>
<td>0.35</td>
<td>0.32</td>
</tr>
<tr>
<td>Irecv</td>
<td>14</td>
<td>25.8</td>
<td>0.07</td>
<td>0.32</td>
<td>0.34</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
But then something happened …

Tool developers got very creative!
Record each invocation of an MPI routine
   – Lead to broad range of trace tools (e.g., Jumpshot and Vampir)

Inspect message meta-data
   – Lead to MPI correctness checkers (e.g., Marmot, Umpire, MUST)

Inspect message contents
   – Transparent checksums for message transfers

Run applications on reduced MPI_COMM_WORLD
   – Reserve nodes for support purposes (e.g., load balancers)

Replace data types to add piggybacking information
   – Useful to track critical path information

Replace MPI operations
   – Ability to modify/re-implement parts of MPI itself
Extreme example: MPIecho

Transparent cloning of MPI processes

[Barry Rountree]
Extreme Example: MPlEcho

Implemented through PMPI wrappers

- Send -> No-Op + 1 Send
- Receives -> Bcast

Enables parallelization of tools
- Fault injections
- Memory checking
Extreme example: MPlecho

Transparent cloning of MPI processes

[Barry Rountree]
The State of MPI Tools

PMPI has led to robust and extensive MPI tool ecosystem
  – Wide variety of portable tools
    • Performance, correctness and debugging tools
  – Use for application support

PMPI, however, also has problems
  – Implementation with weak symbols is often fragile
  – Allows only a single tool
  – Forces tools to be monolithic

This led to the development of PmMPI & the QMPI efforts
The Impact on the MPI Standard

The PMPI definition impacts the whole standard
  – Even where one doesn’t expect it
    • Maximal name length
    • Fortran bindings
    • Threading
  – Needs attention to be maintained

PMPI only allows to track application visible information
  – Does provide access to internal information
  – MPI_T was added to MPI 3.0 to solve this problem
    • After previous failed attempts (like PERUSE)
  – MPI can offer internal state for performance and configuration
    • But MPI can decide what to provide and under what name

New proposal on MPI_T events in the works
  – Callbacks in certain events
  – Provides better support for tracing tools
  – Again leaves freedom to MPI implementations
  – Targeted for MPI 4.0
Other standards are picking up
Goal: enable tools to gather information and associate costs with application source and runtime system

- Hooks for tracing and sampling
- Minimal overhead
- Low implementation complexity
- Mandatory vs. optional parts

Call-stack stitching

- Create user-level view
- Hide runtime impl. details

Status:

- Active API design with outside partners in OpenMP committees
- Included in OpenMP 5.0 draft

Other standards are picking up: e.g., OMPT
But are they overtaking MPI?

The wide-spread use of PMPI is still very unique
   – Combined with MPI_T interface(s) provide unprecedented options
   – Still exploring the opportunities

But:

MPI does not provide an ABI
   – Requires re-compilation of tools for MPI
   – Reduces portability and maintainability of tools
   – Other standards are specifying all types fully

New MPI interfaces are non committal
   – MPI can decide what to offer, if anything
   – Names not standardized
   – Other standards are allowing more concrete specifications
Summary

MPI provides a strong tool ecosystem
- PMPI is the cornerstone since MPI 1.0
- Developers found creative way to exploit it
- MPI_T interface(s) augment it

Wide range of tools have bee developed
- Performance analysis with Profilers and tracers
- Correctness tools (in combination with debuggers)
- Application support tools

MPI always has been a role model for tool interfaces
- Early adoption in MPI 1.0
- Generally broad support in the MPI Forum
- Strong engagement from tool and MPI developers

But other standards are catching up and MPI could learn something from these efforts as well
- ABIs would make tool maintenance and deployment easier
- More concrete requirements on tool support would be helpful