Tools for MPI Performance Analysis and Debugging

Rolf Rabenseifner

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de

Outline

• MPI profiling
  – methods
    • manual profiling
    • counter based profiling
    • trace based profiling, e.g., with VAMPIR
  – goals
    • performance profiling --> to find performance bottlenecks
    • debugging the execution --> to verify the correctness

• Parallel debugging with TotalView
MPI Profiling, the three techniques

- User profiling via PMPI interface
  - User writes profiling wrapper routines
    - named same as original MPI\_... routines
    - and calling internally the original MPI routine via PMPI\_...
  - The MPI library (or with \texttt{-lpmpi}) includes an alias PMPI\_... for each MPI routine

Counter based profiling,
  - examples:
    - MPI on HP
    - most vendors support a counter based profiling tool
    - with the counter based profiling of HLRS on T3E and IRIX
  - counting data about calls to each MPI routine
    - number of calls, time spent in the MPI routines, ...
  - [see next slides]
MPI Profiling, the three techniques

- User profiling via PMPI interface
- Counter based profiling
- Trace based profiling, e.g., with VAMPIRtrace and VAMPIR
  - writing timestamp and details at begin/end of each MPI call to a trace-file
  - off-line analysis of this trace-file
  - [see next talk]

MPI Counter profiling on T3E

- Using default mpt = mpt.1.4.0.2.p
- automatic mail each weekend about all MPI jobs during the week
- manually for each job by setting
  - for sh, ksh, bash:
    export MPIPROFOUT=stdout or
    export MPIPROFOUT=stderr or
    export MPIPROFOUT=filename
  - for csh, tcsh:
    setenv MPIPROFOUT stdout or
    setenv MPIPROFOUT stderr or
    setenv MPIPROFOUT filename
Output of manual counter profiling (1)

rus0598 hwt3e 459$ f90 -o heat-mpi0-big heat-mpi0-big.f
rus0598 hwt3e 460$ export MPIPROFOUT=stdout
rus0598 hwt3e 461$ mpirun -np 16 ./heat-mpi0-big

!size iter- wall clock time communication part abort criterion
! ations [seconds] method [seconds] meth. stride [seconds]
! 16 14320 4.491 1 3.633 1 10 0.2440

-------- BEGIN of PROFILING --------

C <91911>

.. Summary (avg. on each PE):
.. Number of PEs: 16
.. MPI percentage = mpi_time / (appl_time + mpi_time): 83% <<<<<<
.. mpi time per PE: 3.59e+00 sec = 0.001 hours
.. application+mpi time per PE: 4.30e+00 sec = 0.001 hours,
.. sum on all PEs: 6.88e+01 sec = 0.019 hours

.. Hardware performance counters
-- application events per (appl_time + mpi_time) and PE:
.. INSTR events: 57.678614 Mega_event/sec/PE
.. FP_INSTR events: 18.391458 Mega_event/sec/PE
.. L2CACHE_MISS events: 0.129974 Mega_event/sec/PE

.. This output is explained in:
.. http://www.hlrs.de/structure/support/parallel_computing/models/mpi/
.. profiling_explain.html

[...continuation on next slide]

Output of manual counter profiling (2)

Details:
.. PEs: n=16, uid=843, MPI-Release mpt.1.2.1.2 / avg. on each PE:
.. isend = 57276 calls, 1.20e+00 sec, cnt= 4.3e+04, transf.bytes= 6.79e+06
.. irecv = 57276 calls, 5.99e-01 sec, cnt= 4.3e+04, transf.bytes= 6.79e+06
.. waitall = 28638 calls, 1.58e+00 sec, cnt= 1.1e+05
.. type_vector = 2 calls, 1.69e-05 sec, cnt= 2.1e+01, transf.bytes= 1.66e+02
.. type_commit = 2 calls, 2.07e-05 sec
.. allreduce = 1432 calls, 1.97e-01 sec, cnt= 1.4e+03, transf.bytes= 1.15e+04
.. comm_size = 1 calls, 2.53e-06 sec
.. comm_rank = 2 calls, 3.71e-06 sec
.. cart_create = 1 calls, 1.52e-04 sec
.. cart_coords = 1 calls, 9.38e-06 sec
.. cart_shift = 2 calls, 2.40e-05 sec
.. init = 1 calls, 7.96e-03 sec
.. finalize = 1 calls, 2.00e-03 sec

[...continuation on next slide]
Output of manual counter profiling (3)

Hardware performance counters:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Events/PE</th>
<th>MicroEvents/sec/PE</th>
<th>appl_time</th>
<th>mpi_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>appl: INSTR</td>
<td>247023915</td>
<td>247023915</td>
<td>57.8536810</td>
<td>57.786419</td>
</tr>
<tr>
<td>appl: FP_INSTR</td>
<td>79053259</td>
<td>79053259</td>
<td>57.8536810</td>
<td>57.786419</td>
</tr>
<tr>
<td>appl: L2CACHE_MISS</td>
<td>558677</td>
<td>558677</td>
<td>57.8536810</td>
<td>57.786419</td>
</tr>
<tr>
<td>mpi: INSTR</td>
<td>500624055</td>
<td>500624055</td>
<td>116.468504</td>
<td>116.468504</td>
</tr>
<tr>
<td>mpi: FP_INSTR</td>
<td>146134</td>
<td>146134</td>
<td>116.468504</td>
<td>116.468504</td>
</tr>
<tr>
<td>mpi: L2CACHE_MISS</td>
<td>4246744</td>
<td>4246744</td>
<td>116.468504</td>
<td>116.468504</td>
</tr>
</tbody>
</table>

Available counters and default; also possible values for environment variable PCL_PRIORITY:

- INSTR
- FP_INSTR
- INTEGER_INSTR
- LOAD_INSTR
- STORE_INSTR
- L2CACHE_MISS
- L2CACHE_HIT
- L2CACHE_READ
- L2CACHE_WRITE
- L1DCACHE_MISS
- L1DCACHE_HIT
- L1DCACHE_READWRITE
- L1ICACHE_MISS
- L1ICACHE_HIT
- L1ICACHE_READWRITE
- JUMP_UNSUCC
- ITLB_MISS
- DTLB_MISS
- ATOMIC_SUCCESS
- CYCLES

Efficiency of the computation

..time [sec] mpi= 3.59e+00 / all= 4.92e+00 = 7.3e-01
..time / PEs mpi= 5.74e+01 / all= 7.87e+01
...overhead added by the profiling: 6.23e-01 sec = time_all* 1.3e-01

Analysis program: Output of weekly e-mail

- analyzed jobs (partitions started with mpirun or mptrun): 7
- sum of used PEs: 232
- average of PEs: 33.1
- average of PEs (weighted by CPU time): 95.9
- usage of different MPI routines: 20
- sum of application CPU time: 6.76e+06 s = 1876.99 h
- sum of application CPU time / available CPU time = 2.186 %
- sum of MPI time: 3.95e+06 s = 1098.16 h
- sum of MPI time / sum of CPU time = 58.506 %
- overhead added by the profiling: 2.97e+02 s
- overhead / sum of CPU time = 4.39e-05
- used MPI language binding: C = 7 jobs
- Fortran = 0 jobs
- the following environment variables are used in ... jobs:
  - MPIPROFOUT
- very high MPI percentage

Usage of the system by that user
### Analysis program: Output of weekly e-mail (continued)

<table>
<thead>
<tr>
<th>Column 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPi routine</td>
<td>calls</td>
<td>sum of</td>
<td>sum of</td>
<td>Estimated sum of</td>
<td>sum of</td>
<td>sum of</td>
<td>transfer</td>
<td>input for</td>
<td>estimated sum of</td>
<td>sum of</td>
<td>sum of</td>
<td>sum of</td>
</tr>
<tr>
<td></td>
<td>of</td>
<td>per</td>
<td>sum of</td>
<td>CPU time</td>
<td>of</td>
<td>of</td>
<td>time</td>
<td>COUNT</td>
<td>COUNT*</td>
<td>transfered</td>
<td>call</td>
<td>of</td>
</tr>
<tr>
<td></td>
<td>PE</td>
<td>time</td>
<td>bytes</td>
<td>bytes</td>
<td>seconds</td>
<td>seconds</td>
<td>call</td>
<td>time</td>
<td>time</td>
<td>seconds</td>
<td>time</td>
<td>seconds</td>
</tr>
<tr>
<td>recv</td>
<td>703e+5</td>
<td>303097</td>
<td>263e+1</td>
<td>37</td>
<td>403e+7</td>
<td>292e+8</td>
<td>415</td>
<td>10</td>
<td>300</td>
<td>703e+0</td>
<td>928e-1</td>
<td>184e+1</td>
</tr>
<tr>
<td>get_count</td>
<td>276e+5</td>
<td>119657</td>
<td>368e-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tsend</td>
<td>137e+6</td>
<td>591252</td>
<td>232e+2</td>
<td>169</td>
<td>299e+9</td>
<td>243e10</td>
<td>17734</td>
<td>10</td>
<td>100</td>
<td>137e+1</td>
<td>232e+2</td>
<td>-14e+2</td>
</tr>
<tr>
<td>irecv</td>
<td>137e+6</td>
<td>591252</td>
<td>389e-1</td>
<td>28</td>
<td>64e+10</td>
<td>542e11</td>
<td>394969</td>
<td>10</td>
<td>300</td>
<td>137e+1</td>
<td>232e+2</td>
<td>-21e+3</td>
</tr>
<tr>
<td>waitall</td>
<td>138e+6</td>
<td>592770</td>
<td>403e+3</td>
<td>293</td>
<td>285e+6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reduce</td>
<td>122e+5</td>
<td>52559</td>
<td>104e+4</td>
<td>301e+4</td>
<td>461e+4</td>
<td>338126</td>
<td>51</td>
<td>6</td>
<td>114e-1</td>
<td>101e+4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm_size</td>
<td>464</td>
<td>2</td>
<td>106e-5</td>
<td>106e-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm_rank</td>
<td>464</td>
<td>2</td>
<td>65e-6</td>
<td>65e-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cart_create</td>
<td>232</td>
<td>1</td>
<td>595e-2</td>
<td>254e4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>124e+6</td>
<td>124253</td>
<td>54e-1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>finalsize</td>
<td>232</td>
<td>1</td>
<td>742e-5</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only rough estimate / most time is spent for waiting.