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 `-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)

```bash
rus00598 hwwt3e 459$ f90 -o heat-mpi0-big heat-mpi0-big.f
rus00598 hwwt3e 460$ export MPIPROFOUT=stdout
rus00598 hwwt3e 461$ mpirun -np 16 ./heat-mpi0-big
```

```bash
<table>
<thead>
<tr>
<th>size</th>
<th>iter</th>
<th>wall clock time</th>
<th>communication part</th>
<th>abort criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>14320</td>
<td>4.491</td>
<td>3.633</td>
<td>10.2440</td>
</tr>
</tbody>
</table>
```

--- BEGIN of PROFILING ---

```bash
C <91911>
```

- 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
  - 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

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
    - 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, 7.94e-03 sec
    - finalize = 1 calls, 2.00e-03 sec
```

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

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Events/PE</th>
<th>Mega_event/sec/PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>appl: INSTR</td>
<td>247923915</td>
<td>348.548730</td>
</tr>
<tr>
<td>appl: FP_INSTR</td>
<td>79053259</td>
<td>111.138585</td>
</tr>
<tr>
<td>appl: L2CACHE_MISS</td>
<td>558677</td>
<td>0.785427</td>
</tr>
<tr>
<td>mpi: INSTR</td>
<td>500624505</td>
<td>139.563836</td>
</tr>
<tr>
<td>mpi: FP_INSTR</td>
<td>146134</td>
<td>0.040739</td>
</tr>
<tr>
<td>mpi: L2CACHE_MISS</td>
<td>4246743</td>
<td>1.183905</td>
</tr>
</tbody>
</table>

Available counters and default:

- INSTR
- FP_INSTR
- INTEGER_INSTR
- LOAD_INSTR
- STORE_INSTR
- L2CACHE_MISS
- L2CACHE_HIT
- L2CACHE_READ
- L2CACHE_WRITE
- L2CACHE_READWRITE
- L1DCACHE_MISS
- L1DCACHE_HIT
- L1DCACHE_READWRITE
- L1ICACHE_MISS
- L1ICACHE_HIT
- L1ICACHE_READWRITE
- JUMP_UNSUCCESS
- 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
...profiling overhead= 6.23e-01 sec = time_all* 1.3e-01
-------------- END of PROFILING --------------
```

Analysis program: Output of weekly e-mail

- analyzed jobs (partitions started with mpirun or mpitrace): 7
- sum of used PEs: 232
- average of PEs used by one job: 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 MPI time: 3.95e+06 s = 1098.16 h
- sum of MPI time / sum of CPU time = 58.506 %
- overhead added by the profiling: 2.79e+02 s
- overhead / sum of CPU time = 4.35e-02
- used MPI language binding: C = 7 jobs
- Fortran = 0 jobs
- the following environment variables are used in ... jobs:
  - MPIPROFOUT in 2 jobs

very high MPI percentage

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

<table>
<thead>
<tr>
<th>MPI routine</th>
<th>number of calls</th>
<th>sum of CPU time</th>
<th>COUNT of argu-ments</th>
<th>COUNT*ferred calls</th>
<th>lat-ency</th>
<th>bytes transfer</th>
<th>time [sec]</th>
<th>input for latency</th>
<th>estimated sum of</th>
<th>transmitted bytes</th>
<th>idle/active CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>recv</td>
<td>703e+5</td>
<td>263e+1</td>
<td>415</td>
<td>10</td>
<td>300</td>
<td>928e-1</td>
<td>184e+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get_count</td>
<td>279e+3</td>
<td>268e+2</td>
<td>1</td>
<td></td>
<td></td>
<td>803e-2</td>
<td>184e+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isend</td>
<td>137e+6</td>
<td>232e+2</td>
<td>17734</td>
<td>10</td>
<td>100</td>
<td>232e+2</td>
<td>-14e+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>irecv</td>
<td>137e+6</td>
<td>232e+2</td>
<td>394969</td>
<td>10</td>
<td>300</td>
<td>233e+2</td>
<td>-21e+3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>waitall</td>
<td>138e+6</td>
<td>293e+2</td>
<td>275e+1</td>
<td></td>
<td></td>
<td>275e+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reduce</td>
<td>34177</td>
<td>301e+4</td>
<td>810e-1</td>
<td>51</td>
<td>6</td>
<td>208e+2</td>
<td>229e+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>allreduce</td>
<td>122e+5</td>
<td>930e+5</td>
<td>941e+5</td>
<td>8</td>
<td>6</td>
<td>954e+5</td>
<td>114e+4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm_size</td>
<td>444</td>
<td>2106e-5</td>
<td>1</td>
<td></td>
<td></td>
<td>116e-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>comm_rank</td>
<td>444</td>
<td>2106e-5</td>
<td>1</td>
<td></td>
<td></td>
<td>116e-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cart_create</td>
<td>232</td>
<td>1599e-2</td>
<td>254e+0</td>
<td></td>
<td></td>
<td>595e-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wtime</td>
<td>274e+5</td>
<td>549e-1</td>
<td>549e-1</td>
<td></td>
<td></td>
<td>549e-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>init</td>
<td>232</td>
<td>1548e-2</td>
<td>23471</td>
<td></td>
<td></td>
<td>548e-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>finalize</td>
<td>232</td>
<td>1762e-5</td>
<td>762e-5</td>
<td></td>
<td></td>
<td>762e-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Only rough estimate / most time is spent for waiting*