Tools for MPI Performance Analysis and Debugging

Rolf Rabenseifner

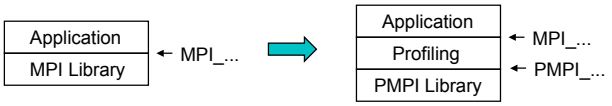
University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de

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* **P**MPI_... for each MPI routine



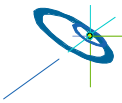
MPI Profiling, the three techniques

- User profiling via PMPI interface

- 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]

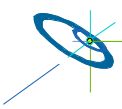
MPI Profiling, the three techniques

- 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)

```
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
!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
```

[...continue 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
```

[...continue on next slide]

Output of manual counter profiling (3)

Efficiency of the computation

```
..Hardware performance counters:

.. part  event name  events/PE,         Mega_event/sec/PE, per appl_time + mpi_time
.. appl: INSTR       247923915, per appl_time: 348.648730,  57.678614 Mevent/sec/PE
.. appl: FP_INSTR     79053259, per appl_time: 111.138585,  18.391458 Mevent/sec/PE
.. appl: L2CACHE_MISS   558677, per appl_time:   0.785427,   0.129974 Mevent/sec/PE
.. mpi:  INSTR       500624505,  per mpi_time: 139.563836, 116.468504 Mevent/sec/PE
.. mpi:  FP_INSTR       146134,  per mpi_time:   0.040739,   0.033998 Mevent/sec/PE
.. mpi:  L2CACHE_MISS  4246743,  per mpi_time:   1.183905,   0.987990 Mevent/sec/PE

.. 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 : L2CACHE_READWRITE : L1DCACHE_MISS : L1DCACHE_HIT :
.. L1DCACHE_READWRITE : L1ICACHE_MISS : L1ICACHE_HIT : L1ICACHE_READWRITE : JUMP_UNSUCCES :
.. ITLB_MISS : DTLB_MISS : ATOMIC_SUCCESS : CYCLES

..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 mptrun):    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 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          in     2 jobs
```

Usage of the system by that user

very high MPI percentage

Analysis program: **Output of weekly e-mail (continued)**

column 1	2	3	4	5	6	7	8	9	10	11	12	13
MPI routine	number of calls	calls per PE	sum of CPU time [sec]	CPU time per call [usec]	sum of COUNT ments	sum of COUNT* P_SIZE =trans ferred bytes	trans-ferred bytes per call	input for col.11-13 latency [us]	band width [MB /s]	estimated sum of latency time [sec]	trans-CPU time [sec]	idle/ sync. time [sec]
recv	703e+5	303097	263e+1	37	403e+7	292e+8	415	10	300	703e+0	928e-1	184e+1
get_count	278e+5	119857	368e-1	1						368e-1		
ssend	703e+5	303096	**134e+4**	19072	403e+7	292e+8	415	50	300	352e+1	928e-1	**134e+4**
isend	137e+6	591252	232e+2	169	299e+9	243e10	17734	10	100	137e+1	232e+2	-14e+2
irecv	137e+6	591252	389e+1	28	677e10	542e11	394969	10	300	137e+1	233e+2	-21e+3
waitall	138e+6	592770	**403e+3**	2931	285e+6			20		275e+1		**400e+3**
reduce	346176	1492	**104e+4**	301e+4	461e+8	181e+9	524126	51	6	177e-1	288e+2	**101e+4**
allreduce	122e+5	52559	**114e+4**	93307	122e+5	975e+5	8	70	6	854e+0	155e-1	**114e+4**
comm_size	464	2	106e-5	2						106e-5		
comm_rank	464	2	685e-6	1						685e-6		
cart_create	232	1	595e-2	25649						595e-2		
wtime	274e+5	118253	549e-1	2						549e-1		
init	232	1	549e-2	23671						549e-2		
finalize	232	1	762e-5	33						762e-5		

Only rough estimate / most time is spent for waiting