An Overview of Profiling

Jerome Vienne

Texas Advanced Computing Center
Disclaimer: counting flops

• Lonestar: Cannot count FLOPS accurately, but can make reasonable estimates
• Stampede (Sandy Bridge): Cannot count FLOPS accurately
  – Results always overcount, by amounts that depend on how busy the memory system is.
• Stampede (Xeon Phi): Cannot count FLOPS easily.
Why do we need to profile?

- To know how the code is behaving
- To detect performance issues
- To know where you need to focus to increase the speed of your code.

There are plenty of tools to profile an application. Be sure to select the right one!
Lonestar != Stampede
Version could be different!

**Lonestar**
- Command line timing
- gprof
- Perfexpert (fairly reliable)
- IPM
- mpiP
- Tau
- Papi
- HPCToolkit

**Stampede**
- Command line timing
- gprof
- Perfexpert (ignore metrics relying on flops)
- IPM
- mpiP
- Tau
- Papi (Cpu+MIC)
- HPCToolkit
- Vtune Amplifier
- perf
Outline

1. Tools provided with Linux:
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
Outline

1. Tools provided with Linux
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
Timers: Command Line

• The command `time` is available in most Unix systems.
• It is simple to use (no code instrumentation required).
• Gives total execution time of a process and all its children in seconds.

```bash
% /usr/bin/time -p .exeFile
real 9.95
user 9.86
sys 0.06
```

Leave out the `-p` option to get additional information:

```bash
% time ./exeFile
% 9.860u 0.060s 0:09.95 99.9% 0+0k 0+0io 0pf+0w
```
Timers: Code Section

```plaintext
INTEGER :: rate, start, stop
REAL    :: time

CALL SYSTEM_CLOCK(COUNT_RATE = rate)
CALL SYSTEM_CLOCK(COUNT = start)

! Code to time here

CALL SYSTEM_CLOCK(COUNT = stop)
time = REAL( ( stop - start )/ rate )

#include <time.h>

double start, stop, time;
start = (double)clock()/CLOCKS_PER_SEC;

/* Code to time here */

stop = (double)clock()/CLOCKS_PER_SEC;
time = stop - start;
```
Outline

1. Tools provided with Linux:
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
About GPROF

GPROF is the GNU Project PROFiler.  

• Requires recompilation of the code.

• Compiler options and libraries provide wrappers for each routine call and periodic sampling of the program.

• Provides three types of profiles
  - Flat profile
  - Call graph
  - Annotated source

gnu.org/software/binutils/
Types of Profiles

• Flat Profile
  – CPU time spent in each function (self and cumulative)
  – Number of times a function is called
  – Useful to identify most expensive routines

• Call Graph
  – Number of times a function was called by other functions
  – Number of times a function called other functions
  – Useful to identify function relations
  – Suggests places where function calls could be eliminated

• Annotated Source
  – Indicates number of times a line was executed
Profiling with gprof

Use the -pg flag during compilation:

% gcc -g -pg ./srcFile.c
% icc -g -p ./srcFile.c
% pgcc -g -pg ./srcFile.c

Run the executable. An output file gmon.out will be generated with the profiling information.

Execute gprof and redirect the output to a file:

% gprof ./exeFile gmon.out > profile.txt
% gprof -l ./exeFile gmon.out > profile_line.txt
% gprof -A ./exeFile gmon.out > profile_annotated.txt
In the flat profile we can identify the most expensive parts of the code (in this case, the calls to matSqrt, matCube, and sysCube).

<table>
<thead>
<tr>
<th>% cumulative</th>
<th>self</th>
<th>self</th>
<th>total</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>seconds</td>
<td>seconds</td>
<td>calls</td>
<td>s/call</td>
</tr>
<tr>
<td>50.00</td>
<td>2.47</td>
<td>2.47</td>
<td>2</td>
<td>1.24</td>
</tr>
<tr>
<td>24.70</td>
<td>3.69</td>
<td>1.22</td>
<td>1</td>
<td>1.22</td>
</tr>
<tr>
<td>24.70</td>
<td>4.91</td>
<td>1.22</td>
<td>1</td>
<td>1.22</td>
</tr>
<tr>
<td>0.61</td>
<td>4.94</td>
<td>0.03</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>0.00</td>
<td>4.94</td>
<td>0.00</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>4.94</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>4.94</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>index</td>
<td>% time</td>
<td>self</td>
<td>children</td>
<td>called</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>&lt;hicore&gt;</td>
<td>(8)</td>
</tr>
<tr>
<td>[1]</td>
<td>100.0</td>
<td>0.03</td>
<td>4.91</td>
<td>1</td>
</tr>
<tr>
<td>0.00</td>
<td>1.24</td>
<td>1/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2]</td>
<td>50.0</td>
<td>2.47</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>1.24</td>
<td>1/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[3]</td>
<td>25.0</td>
<td>0.00</td>
<td>1.24</td>
<td>1</td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Visual Call Graph

main

sysSqrt → matSqrt → vecSqrt → matCube → vecCube → sysCube
## Call Graph Profile

<table>
<thead>
<tr>
<th>index</th>
<th>% time</th>
<th>self</th>
<th>children</th>
<th>called</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>&lt;hicore&gt; (8)</td>
<td></td>
</tr>
<tr>
<td>[1]</td>
<td>100.0</td>
<td>0.03</td>
<td>4.91</td>
<td>1</td>
<td>main [1]</td>
</tr>
<tr>
<td>0.00</td>
<td>1.24</td>
<td>1/1</td>
<td></td>
<td>sysSqrt [3]</td>
<td></td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td>matSqrt [2]</td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>sysCube [5]</td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>matCube [4]</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td>vecSqrt [6]</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>vecCube [7]</td>
<td></td>
</tr>
</tbody>
</table>

| 1.24  | 0.00   | 1/2  |          | main [1] |
| 1.24  | 0.00   | 1/2  |          | sysSqrt [3] |

| [2]   | 50.0   | 2.47 | 0.00     | 2      | matSqrt [2] |

| 0.00  | 1.24   | 1/1  |          | main [1] |

| [3]   | 25.0   | 0.00 | 1.24     | 1      | sysSqrt [3] |
| 1.24  | 0.00   | 1/2  |          | matSqrt [2] |
| 0.00  | 0.00   | 1/2  |          | vecSqrt [6] |
Visual Call Graph

main

sysSqrt
vecSqrt
matSqrt
matCube
vecCube
sysCube
### Call Graph Profile

<table>
<thead>
<tr>
<th>Index</th>
<th>% Time</th>
<th>Self</th>
<th>Children</th>
<th>Called</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td>&lt;hicore&gt; (8)</td>
<td></td>
</tr>
<tr>
<td>[1]</td>
<td>100.0</td>
<td>0.03</td>
<td>4.91</td>
<td>1</td>
<td>main [1]</td>
</tr>
<tr>
<td>0.00</td>
<td>1.24</td>
<td>1/1</td>
<td></td>
<td>sysSqrt [3]</td>
<td></td>
</tr>
<tr>
<td>1.24</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td>matSqrt [2]</td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>sysCube [5]</td>
<td></td>
</tr>
<tr>
<td>1.22</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>matCube [4]</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/2</td>
<td></td>
<td>vecSqrt [6]</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>1/1</td>
<td></td>
<td>vecCube [7]</td>
<td></td>
</tr>
</tbody>
</table>

---

| [2]   | 50.0   | 2.47 | 0.00     | 2      | matSqrt [2] |

---

| [3]   | 25.0   | 0.00 | 1.24     | 1      | sysSqrt [3] |
| 1.24  | 0.00   | 1/2  |          | matSqrt [2] |
| 0.00  | 0.00   | 1/2  |          | vecSqrt [6] |
Outline

1. Tools provided with Linux:
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
IPM: Integrated Performance Monitoring

• “IPM is a portable profiling infrastructure for parallel codes. It provides a low-overhead performance summary of the computation and communication in a parallel program”

• IPM is a quick, easy and concise profiling tool

• The level of detail it reports is smaller than TAU, PAPI or HPCToolkit
IPM: Integrated Performance Monitoring

- IPM features:
  - easy to use
  - has low overhead
  - is scalable
- Requires no source code modification, just adding the “-g” option to the compilation
- Produces XML output that is parsed by scripts to generate browser-readable html pages
IPM: Integrated Performance Monitoring

- Available on Ranger and Lonestar with the mvapich libraries
- Create ipm environment with module command before running code: "module load ipm"
- In your job script, set up the following ipm environment before the ibrun command:
  module load ipm
  export LD_PRELOAD=$TACC_IPM_LIB/libipm.so
  export IPM_REPORT=full

  ibrun <my executable> <my arguments>
IPM: Integrated Performance Monitoring

- Export LD_PRELOAD=$TACC_IPM_LIB/libipm.so
  - must be inside job script
- **IPM_REPORT**: full, terse or none are the levels of information
- **IPM_MPI_THRESHOLD**: Reports only routines using this percentage (or more) of MPI time.
  - e.g. "IPM_MPI_THRESHOLD 0.3" report subroutines that consume more than 30% of the total MPI time.
- Important details: "module help ipm"
## IPMv0.922

### command:
/work/01125/yye00/ICAC/cactus_SandTank SandTank.par

### host:
i101-309/x86_64_Linux

### mpi_tasks:
32 on 2 nodes

### start:
05/26/09/11:49:06

calculated wallclock: 2.758892 sec

### stop:
05/26/09/11:49:09

### %comm:
2.01

### gbytes:
4.38747e+00 total

gflop/sec: 9.39108e-02 total

### region:
*

<table>
<thead>
<tr>
<th>[ntasks]</th>
<th>&lt;avg&gt;</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>entries</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### wallclock:
88.2742

calculated wallclock: 2.75857 2.75816 2.75889

### user:
5.51634

calculated user: 0.172386 0.148009 0.200012

### system:
1.771

calculated system: 0.0553438 0.0536683 0.056717

### %comm:
2.00602

calculated %comm: 1.94539 2.05615

### gflop/sec:
0.0939108

calculated gflop/sec: 0.00293471 0.00293338 0.002952

### gbytes:
4.38747

calculated gbytes: 0.137109 0.136581 0.144985

### PAPI_FP_OPS:
2.5909e+08

calculated PAPI_FP_OPS: 8.09655e+06 8.09289e+06 8.14685e+06

### PAPI_TOT_CYC:
6.80291e+09

calculated PAPI_TOT_CYC: 2.12591e+08 2.02236e+08 2.19109e+08

### PAPI_VEC_INS:
5.95596e+08

calculated PAPI_VEC_INS: 1.86124e+07 1.85964e+07 1.8756e+07

### PAPI_VEC_INS:
4.16377e+09

calculated PAPI_VEC_INS: 1.30118e+08 1.0987e+08 1.35676e+08

### MPI Allreduce:
0.978938

calculated MPI Allreduce: 53248 55.28 1.11

### MPI_Comm_rank:
0.316355

calculated MPI_Comm_rank: 6002 17.86 0.36

### MPI Barrier:
0.247135

calculated MPI Barrier: 3680 13.95 0.28

### MPI Allgather:
0.16621

calculated MPI Allgather: 2848 9.39 0.19

### MPI Bcast:
0.0217298

calculated MPI Bcast: 576 1.23 0.02

### MPI Allgather:
0.0216982

calculated MPI Allgather: 672 1.23 0.02

### MPI Bcast:
0.0186796

calculated MPI Bcast: 32 1.05 0.02

### MPI Comm_size:
0.000139921

calculated MPI Comm_size: 2112 0.01 0.00

### MPI Comm_size:
0.000115622

calculated MPI Comm_size: 32 0.01 0.00

### MPI_Send:
0.000139921

calculated MPI_Send: 2112 0.01 0.00

### MPI Rx:
0.000115622

calculated MPI Rx: 32 0.01 0.00

### MPI Tx:
0.000139921

calculated MPI Tx: 2112 0.01 0.00

### MPI_time:
0.000139921

calculated MPI time: 2112 0.01 0.00

### MPI Time:
0.000115622

calculated MPI Time: 32 0.01 0.00
IPM: Integrated Performance Monitoring

command: /work/01125/yeye00/IPM/Benchmark/Defiant.exe_perurb -runpermuted -da_grid.x 50 -da_grid.y 50 -da_grid_z 50 -seed.phi 3454345 -seed.k11 56756756 -seed.k22 235759 -seed.k33 234656 -seed_flowmask 322111 -percentage.phi 0.1_percentage.k11 0.1_percentage.k22 0.1_percentage.k33 0.1_percentage_flowmask 0.15 -endtime 1.0 -ksp_type bicg -pc_type bjacobi

codename: unknown
username: yeye00
group: G-801077
host: 1115-108 (x86_64 Linux)
mpi_tasks: 64 on 4 hosts
start: 07/13/10:00:28:10
wallclock: 3.80580e+00 sec
stop: 07/13/10:00:28:13
%comm: 11.5752798360397

total memory: 10.85705 gbytes
total gflop/sec: 1.02192900310053
switch(send): 0 gbytes
switch(recv): 0 gbytes

### Computation

<table>
<thead>
<tr>
<th>Event</th>
<th>Count</th>
<th>Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_FP_OPS</td>
<td>3889256866</td>
<td>*</td>
</tr>
<tr>
<td>PAPI_TOT_CYC</td>
<td>93055641837</td>
<td>*</td>
</tr>
<tr>
<td>PAPI_TOT_INS</td>
<td>82058705179</td>
<td>*</td>
</tr>
<tr>
<td>PAPI_VEC_INS</td>
<td>8293137711</td>
<td>*</td>
</tr>
</tbody>
</table>

### Communication

% of MPI Time

- MPI_Allreduce
- MPI_Sendrecv
- MPI_Bcast
- MPI_Barrier
- MPI_Scan
- MPI_Init
- MPI_Finalize
- MPI_InitMPI

### HPM Counter Statistics

<table>
<thead>
<tr>
<th>Event</th>
<th>Ntasks</th>
<th>Avg</th>
<th>Min(rank)</th>
<th>Max(rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_FP_OPS</td>
<td></td>
<td>60769636.53</td>
<td>532546.64 (63)</td>
<td>68822066 (21)</td>
</tr>
<tr>
<td>PAPI_TOT_CYC</td>
<td></td>
<td>1453994403.76</td>
<td>13480405.00 (23)</td>
<td>1646491900 (12)</td>
</tr>
<tr>
<td>PAPI_TOT_INS</td>
<td></td>
<td>1262167268.42</td>
<td>113430964.0 (0)</td>
<td>147779580.0 (12)</td>
</tr>
<tr>
<td>PAPI_VEC_INS</td>
<td></td>
<td>1295802789.73</td>
<td>113000200.63 (63)</td>
<td>14691553.21 (21)</td>
</tr>
</tbody>
</table>
### IPM: Event Statistics

<table>
<thead>
<tr>
<th>Event</th>
<th>Buffer Size</th>
<th>Ncalls</th>
<th>Total Time</th>
<th>Min Time</th>
<th>Max Time</th>
<th>% MPI</th>
<th>% Wall</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_Allreduce</td>
<td>8</td>
<td>79680</td>
<td>4.178</td>
<td>8.225e-06</td>
<td>8.882e-04</td>
<td>14.82</td>
<td>1.72</td>
</tr>
<tr>
<td>MPI_Bcast</td>
<td>4</td>
<td>1024</td>
<td>4.047</td>
<td>5.914e-06</td>
<td>6.413e-02</td>
<td>14.35</td>
<td>1.66</td>
</tr>
<tr>
<td>MPI_Allreduce</td>
<td>512</td>
<td>39936</td>
<td>3.803</td>
<td>1.660e-05</td>
<td>1.170e-01</td>
<td>13.49</td>
<td>1.56</td>
</tr>
<tr>
<td>MPI_Allreduce</td>
<td>4</td>
<td>25472</td>
<td>2.250</td>
<td>6.012e-07</td>
<td>1.552e-03</td>
<td>7.98</td>
<td>0.92</td>
</tr>
<tr>
<td>MPI_Barrier</td>
<td>0</td>
<td>64</td>
<td>1.176</td>
<td>1.814e-02</td>
<td>1.865e-02</td>
<td>4.17</td>
<td>0.48</td>
</tr>
<tr>
<td>MPI_Isend</td>
<td>8</td>
<td>630</td>
<td>1.028</td>
<td>3.427e-07</td>
<td>1.647e-02</td>
<td>3.65</td>
<td>0.42</td>
</tr>
<tr>
<td>MPI_Isend</td>
<td>4</td>
<td>4556</td>
<td>0.943</td>
<td>2.738e-07</td>
<td>1.833e-02</td>
<td>3.34</td>
<td>0.39</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>14976</td>
<td>144</td>
<td>0.722</td>
<td>1.030e-03</td>
<td>7.308e-03</td>
<td>2.56</td>
<td>0.30</td>
</tr>
<tr>
<td>MPI_Comm_rank</td>
<td>0</td>
<td>106948</td>
<td>0.620</td>
<td>3.725e-06</td>
<td>9.872e-03</td>
<td>2.20</td>
<td>0.25</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>0</td>
<td>6093</td>
<td>0.542</td>
<td>4.615e-07</td>
<td>1.358e-02</td>
<td>1.92</td>
<td>0.22</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>1248</td>
<td>27462</td>
<td>0.519</td>
<td>5.183e-07</td>
<td>2.723e-04</td>
<td>1.84</td>
<td>0.21</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>16224</td>
<td>144</td>
<td>0.517</td>
<td>4.283e-04</td>
<td>7.129e-03</td>
<td>1.83</td>
<td>0.21</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>1352</td>
<td>20370</td>
<td>0.496</td>
<td>5.197e-07</td>
<td>5.783e-03</td>
<td>1.76</td>
<td>0.20</td>
</tr>
<tr>
<td>MPI_Start</td>
<td>0</td>
<td>269196</td>
<td>0.396</td>
<td>3.623e-07</td>
<td>3.685e-03</td>
<td>1.40</td>
<td>0.16</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>13824</td>
<td>48</td>
<td>0.298</td>
<td>4.935e-03</td>
<td>7.227e-03</td>
<td>1.06</td>
<td>0.12</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>1152</td>
<td>10980</td>
<td>0.243</td>
<td>5.383e-07</td>
<td>2.310e-04</td>
<td>0.86</td>
<td>0.10</td>
</tr>
<tr>
<td>MPI_Bcast</td>
<td>216</td>
<td>576</td>
<td>0.231</td>
<td>2.502e-06</td>
<td>5.843e-03</td>
<td>0.82</td>
<td>0.09</td>
</tr>
<tr>
<td>MPI_Allgath</td>
<td>4</td>
<td>9088</td>
<td>0.215</td>
<td>5.118e-07</td>
<td>1.733e-03</td>
<td>0.76</td>
<td>0.09</td>
</tr>
<tr>
<td>MPI_Waitall</td>
<td>184</td>
<td>11</td>
<td>0.210</td>
<td>1.633e-02</td>
<td>2.135e-02</td>
<td>0.74</td>
<td>0.09</td>
</tr>
<tr>
<td>MPI_Scan</td>
<td>4</td>
<td>384</td>
<td>0.144</td>
<td>2.259e-05</td>
<td>1.600e-03</td>
<td>0.51</td>
<td>0.06</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>147</td>
<td>453</td>
<td>0.141</td>
<td>4.866e-07</td>
<td>1.406e-02</td>
<td>0.50</td>
<td>0.06</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>4</td>
<td>448</td>
<td>0.132</td>
<td>4.345e-07</td>
<td>5.805e-03</td>
<td>0.47</td>
<td>0.05</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>320</td>
<td>18</td>
<td>0.120</td>
<td>3.002e-06</td>
<td>1.284e-02</td>
<td>0.42</td>
<td>0.05</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>17576</td>
<td>42</td>
<td>0.108</td>
<td>6.682e-05</td>
<td>6.406e-03</td>
<td>0.38</td>
<td>0.04</td>
</tr>
<tr>
<td>MPI_Waitall</td>
<td>72</td>
<td>6</td>
<td>0.103</td>
<td>1.547e-02</td>
<td>2.038e-02</td>
<td>0.36</td>
<td>0.04</td>
</tr>
<tr>
<td>MPI_Waitall</td>
<td>96</td>
<td>30</td>
<td>0.091</td>
<td>2.882e-06</td>
<td>1.563e-02</td>
<td>0.32</td>
<td>0.04</td>
</tr>
<tr>
<td>MPI_Waitany</td>
<td>624</td>
<td>140</td>
<td>0.088</td>
<td>1.126e-06</td>
<td>7.880e-03</td>
<td>0.31</td>
<td>0.04</td>
</tr>
<tr>
<td>MPI_Recv</td>
<td>8</td>
<td>9</td>
<td>0.085</td>
<td>8.373e-07</td>
<td>7.696e-02</td>
<td>0.30</td>
<td>0.03</td>
</tr>
</tbody>
</table>
IPM Buffer Size Distribution: % of Comm Time

Message Buffer Size Distributions: time

Buffer size (bytes)

% comm time <= buffer size

cumulative values, values

TACC
THE UNIVERSITY OF TEXAS AT AUSTIN
TEXAS ADVANCED COMPUTING CENTER
Buffer Size Distribution: Ncalls

Message Buffer Size Distributions: Ncalls

 Cumulative values, values

TACC
THE UNIVERSITY OF TEXAS AT AUSTIN
TEXAS ADVANCED COMPUTING CENTER
Communication Topology: point to point data flow

- data sent, data recv, time spent, map_data_file, map_adjacency_file

TACC
TEXAS ADVANCED COMPUTING CENTER
IPM: Integrated Performance Monitoring

• When to use IPM?
  – To quickly find out where your code is spending most of its time (in both computation and communication)
  – For performing scaling studies (both strong and weak)
  – When you suspect you have load imbalance and want to verify it quickly
  – For a quick look at the communication pattern
  – To find out how much memory you are using per task
  – To find the relative communication & compute time
IPM: Integrated Performance Monitoring

• When IPM is NOT the answer
  – When you already know where the performance issues are
  – When you need detailed performance information on exact lines of code
  – When want to find specific information such as cache misses
Outline

1. Tools provided with Linux:
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
Advanced Profiling Tools

• Can be intimidating:
  – Difficult to install
  – Many dependences
  – Could require kernel patches

  } Not your problem!!

• Useful for serial and parallel programs

• Extensive profiling and scalability information

• Analyze code using:
  – Timers
  – Hardware registers (PAPI)
  – Function wrappers
Outline

1. Tools provided with Linux
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
PAPI

PAPI is a Performance Application Programming Interface

icl.cs.utk.edu/papi

- API to use hardware counters
- Behind Tau, HPCToolkit
- Multiplatform:
  - Most Intel & AMD chips
  - IBM POWER 4/5/6
  - Cray X/XD/XT
  - Sun UltraSparc I/II/III
  - MIPS
  - SiCortex
  - Cell
- Available as a module on Lonestar and Stampede (mind the hardware counters issue with sandybridge xeons).
Outline

1. Tools provided with Linux
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
Tau

TAU is a suite of Tuning and Analysis Utilities
www.cs.uoregon.edu/research/tau

• 12+ year project involving
  – University of Oregon Performance Research Lab
  – LANL Advanced Computing Laboratory
  – Research Centre Julich at ZAM, Germany

• Integrated toolkit
  – Performance instrumentation
  – Measurement
  – Analysis
  – Visualization
Tau: Measurements

• Parallel profiling
  – Function-level, block (loop)-level, statement-level
  – Supports user-defined events
  – TAU parallel profile data stored during execution
  – Hardware counter values (multiple counters)
  – Support for callgraph and callpath profiling

• Tracing
  – All profile-level events
  – Inter-process communication events
  – Trace merging and format conversion
PDT is used to instrument your code.

**Replace mpicc and mpif90 in make files with tau_f90.sh and tau_cc.sh**

It is necessary to specify all the components that will be used in the instrumentation (mpi, openmp, profiling, counters [PAPI], etc. However, these come in a limited number of combinations.)

Combinations: First determine what you want to do (profiling, PAPI counters, tracing, etc.) and the programming paradigm (mpi, openmp), and the compiler. PDT is a required component:

<table>
<thead>
<tr>
<th>Instrumentation</th>
<th>Parallel Paradigm</th>
<th>Collectors</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDT</td>
<td>MPI</td>
<td>PAPI</td>
<td>Intel</td>
</tr>
<tr>
<td>Hand-coded</td>
<td>OMP</td>
<td>Callpath</td>
<td>PGI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GNU (gcc)</td>
</tr>
</tbody>
</table>
Tau: Instrumentation

You can view the available combinations
    (alias tauTypes 'ls -C1 $TAU | grep Makefile ').

Your selected combination is made known to the compiler wrapper through
the TAU_MAKEFILE environment variable.

E.g. the PDT instrumentation (pdt) for the Intel compiler (icpc) with MPI (mpi)
is set with this command:

    setenv TAU_MAKEFILE   $TAU/Makefile.tau-icpc-mpi-pdt

Other run-time and instrumentation options are set through TAU_OPTIONS. For verbose:

    setenv TAU_OPTIONS   ‘-optVerbose’
Tau Paraprof Overview

Raw files

PerfDMF managed (database)

Application

Experiment

Trials

HPMToolkit

Metadata

MpiP

TAU
Tau Paraprof Manager Window

Provides Machine Details
Organizes Runs as: Applications, Experiments and Trials.
Routine Time Experiment

Profile Information is in “GET_TIME_OF_DAY” metric
Mean and Standard Deviation Statistics given.
Multiply Matrices Routine Results

Function Data Window gives a closer look at a single function:

<table>
<thead>
<tr>
<th>File</th>
<th>Options</th>
<th>Windows</th>
<th>Help</th>
</tr>
</thead>
</table>

Name: MULTIPLY_MATRICES [[matmulf90] (25, 18)]
Metric Name: GET_TIME_OF_DAY
Value: Exclusive
Units: seconds

<table>
<thead>
<tr>
<th>Value</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>59.999</td>
<td>nct12.0</td>
<td>0.01</td>
</tr>
<tr>
<td>59.965</td>
<td>nct13.0</td>
<td>0.02</td>
</tr>
<tr>
<td>59.955</td>
<td>nct14.0</td>
<td>0.03</td>
</tr>
<tr>
<td>59.935</td>
<td>nct15.0</td>
<td>0.04</td>
</tr>
<tr>
<td>59.945</td>
<td>nct16.0</td>
<td>0.05</td>
</tr>
<tr>
<td>59.935</td>
<td>nct17.0</td>
<td>0.06</td>
</tr>
<tr>
<td>59.925</td>
<td>nct18.0</td>
<td>0.07</td>
</tr>
<tr>
<td>59.915</td>
<td>nct19.0</td>
<td>0.08</td>
</tr>
<tr>
<td>59.904</td>
<td>nct20.0</td>
<td>0.09</td>
</tr>
</tbody>
</table>

not from same run
Float Point OPS trial

Hardware Counters provide Floating Point Operations (Function Data view).
L1 Data Cache Miss trial

Hardware Counters provide L1 Cache Miss Operations.
Call Path

Call Graph Paths  (Must select through “thread” menu.)
Call Path

TAU_MAKEFILE =
...Makefile.tau-callpath.icpc-mpi-pdt
Derived Metrics

Select Argument 1 (green ball); Select Argument 2 (green ball); Select Operation; then Apply. Derived Metric will appear as a new trial.
Be careful ➔ even though ratios are constant, cores may do different amounts of work/operations per call.
Outline

1. Tools provided with Linux
   • Time your code
   • GPROF
2. MPI profiling:
   • IPM
3. Advanced Tools:
   • PAPI
   • TAU
4. Other tools...
Other tools

- **Valgrind** [valgrind.org]
  - Powerful instrumentation framework, often used for debugging memory problems
- **Scalasca** [www.fz-juelich.de/jsc/scalasca]
  - Similar to Tau, complete suit of tuning and analysis tools.
- **HPCToolkit** [www.hpctoolkit.org]
  - Interesting tool with a lot of promise
- **mpiP** [mpip.sourceforge.net]
  - Lightweight, Scalable MPI Profiling
- **Perf**
  - Linux profiling with performance counters
- **Intel Tools:**
  - Intel Trace Analyzer and Collector: MPI profiling tool
  - Intel Vtune: integrated performance analysis and tuning environment
  - Intel Inspector: dynamic analysis tools for memory and thread error detection.
  - Intel Advisor: design tool that helps add parallelism into your application.