Performance Evaluation Tool
TAU for NYBlue Users

Tulin Kaman

Department of Applied Mathematics and Statistics
Stony Brook University

New York Blue IBM Blue Gene Supercomputer
November, 2008
Tuning and Analysis Utilities

- Performance evaluation tool
- Support for multiple parallel programming paradigms: MPI, Multi-threading, Hybrid (MPI+Threads)
- Access to hardware counters.
- Automatically instruments your code.
How to use TAU?

- Set a couple of environment variables
  - $PATH, $TAU_MAKEFILE, $TAU_OPTIONS
- Instrument the program by inserting TAU macros or automatically.
- To take advantage of TAU's automatic instrumentation features, Program Database Toolkit (PDT) needs to be installed <pdt-dir>. PDT provides access to the high-level interface of source code for analysis tools and applications.
- For **automatic** instrumentation
  - Replace the compiler with TAU compiler script.
TAU Configuration

- Each configuration labeled with the options used.
  - ./configure -mpi -arch=bgl -pdt=<pdt-dir> -pdt=xlC -PROFILE(default)
  - OTHER OPTIONS: -PROFILECALLPATH/MULTIPLECOUNTERS/…

- Each configuration creates a unique Makefile.
  - <tau-dir>/bgl/lib for BG/L platform
  - <tau-dir>/bgp/lib for BG/P platform

- TAU compiler scripts are installed in
  - <tau-dir>/bgl/bin for BG/L platform
  - <tau-dir>/bgp/bin for BG/P platform

- Add the bin directory to your path.
Set TAU_MAKEFILE

- Set the environment variable TAU_MAKEFILE to the location of the tau makefile.
- List of TAU’s Makefile
  - Makefile.tau-mpi-pdt
  - Makefile.tau-callpath-mpi-pdt
  - Makefile.tau-mpi-pdt-mpitrace
  - Makefile.tau-mpi-pthread-pdt
  - Makefile.tau-multiplecounters-mpi-papi-pdt …
- Start with MPI instrumentation & PDT for automatic source instrumentation.

```
export TAU_MAKEFILE=<tau-dir>/bgl/lib/Makefile.tau-mpi-pdt
export TAU_MAKEFILE=<tau-dir>/bgp/lib/Makefile.tau-mpi-pdt
```
TAU Shell Scripts

- Compile your code with TAU shell scripts.

<table>
<thead>
<tr>
<th>GNU Compilers</th>
<th>IBM XL Compilers</th>
<th>TAU shell scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpicc</td>
<td>mpixlc</td>
<td>mpixlc_r</td>
</tr>
<tr>
<td>mpicxx</td>
<td>mpixlcxx</td>
<td>mpixlcxx_r</td>
</tr>
<tr>
<td>mpif77</td>
<td>mpixlf77</td>
<td>mpixlf77_r</td>
</tr>
<tr>
<td>mpif90</td>
<td>mpixlf90</td>
<td>mpixlf90_r</td>
</tr>
</tbody>
</table>

- If your Fortran code is a fixed-format Fortran code, use “tau_f90.sh -qfixed”

- **-D options to XLF**: The XL Fortran compilers require a slightly different syntax to define preprocessor macro symbols. Instead of just "-D", you should use "-WF,-D" like this: mpixlf77 -WF,-DMYFLAG myprogram.F
Analyze Performance Data

- **pprof (for text based display)**
  - sorts and displays profile data generated by TAU.
  - Execute `pprof` in the directory where profile files are located.

- **paraprof (for GUI display)**
  - TAU has Java based performance data viewer.
  - Requires Java 1.4 or above, add it to your path.
  - --pack options pack the data into packed (.ppk) format and it does not launch the paraprof GUI.
    
    ```
    paraprof --pack filename
    ```
  - To launch the GUI
    ```
    paraprof filename.ppk
    ```
pprof (Text based display)

<table>
<thead>
<tr>
<th>Time</th>
<th>Exclusive</th>
<th>Inclusive</th>
<th>#Call</th>
<th>#Subrs</th>
<th>Inclusive Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>msec</td>
<td>total msec</td>
<td></td>
<td></td>
<td>usec/call</td>
</tr>
<tr>
<td>100.0</td>
<td>0.081</td>
<td>21:16.904</td>
<td>1</td>
<td>2</td>
<td>1276904738 int main(int, char **) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.083</td>
<td>21:16.904</td>
<td>1</td>
<td>1</td>
<td>1276904059 int dmain(int, char **, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.083</td>
<td>21:16.904</td>
<td>1</td>
<td>1</td>
<td>1276904859 int main(int, char **) C =&gt; int dmain(int, char **, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.125</td>
<td>21:16.903</td>
<td>1</td>
<td>3</td>
<td>1276903975 int dmain(int, char **, INIT_DATA *, INIT_PHYSICS *) C =&gt; void perform_initialization(int, char **, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.125</td>
<td>21:16.903</td>
<td>1</td>
<td>3</td>
<td>1276903975 void perform_initialization(int, char **, INIT_DATA *, INIT_PHYSICS *) C =&gt; void perform_initialization(int, char **, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.071</td>
<td>21:16.903</td>
<td>1</td>
<td>1</td>
<td>1276903684 void g_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS, S **) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.071</td>
<td>21:16.903</td>
<td>1</td>
<td>1</td>
<td>1276903684 void perform_initialization(int, char **, INIT_DATA *, INIT_PHYSICS *) C =&gt; void perform_initialization(int, char **, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.127</td>
<td>21:16.903</td>
<td>1</td>
<td>3</td>
<td>1276903813 void d_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>100.0</td>
<td>0.127</td>
<td>21:16.903</td>
<td>1</td>
<td>3</td>
<td>1276903813 void g_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS, S **) C =&gt; void d_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>99.9</td>
<td>0.571</td>
<td>21:16.174</td>
<td>1</td>
<td>20</td>
<td>1276174530 void d_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS *) C =&gt; void d_init_run(int *, char ***, INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>99.9</td>
<td>0.571</td>
<td>21:16.174</td>
<td>1</td>
<td>20</td>
<td>1276174530 void set_up_cauchy_data(INIT_DATA *, INIT_PHYSICS *) C =&gt; void set_up_cauchy_data(INIT_DATA *, INIT_PHYSICS *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.211</td>
<td>21:06.620</td>
<td>1</td>
<td>6</td>
<td>12686620615 void states(INIT_DATA *, INIT_PHYSICS, S **) C =&gt; void states(INIT_DATA *, INIT_PHYSICS, S **) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.211</td>
<td>21:06.620</td>
<td>1</td>
<td>6</td>
<td>12686620615 void states(INIT_DATA *, INIT_PHYSICS, S **) C =&gt; void states(INIT_DATA *, INIT_PHYSICS, S **) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.134</td>
<td>21:06.481</td>
<td>1</td>
<td>6</td>
<td>1268481343 void clip_front_to_subdomain(Front *) C =&gt; void clip_front_to_subdomain(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.134</td>
<td>21:06.481</td>
<td>1</td>
<td>6</td>
<td>1268481343 void clip_front_to_subdomain(Front *) C =&gt; void clip_front_to_subdomain(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.115</td>
<td>21:06.480</td>
<td>69</td>
<td>276</td>
<td>18352796 bool scatter_front(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>0.115</td>
<td>21:06.480</td>
<td>69</td>
<td>276</td>
<td>18352796 bool scatter_front(Front *) C =&gt; bool scatter_front(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>1</td>
<td>21:06.480</td>
<td>68</td>
<td>84</td>
<td>18624708 bool form_subintfca_communication(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>1</td>
<td>21:06.480</td>
<td>68</td>
<td>84</td>
<td>18624708 bool scatter_front(Front *) C =&gt; bool form_subintfca_communication(Front *) C</td>
</tr>
<tr>
<td>99.2</td>
<td>3</td>
<td>21:06.480</td>
<td>68</td>
<td>385</td>
<td>18624707 bool form_subintfca_communication(Front *) C =&gt; bool g form_subintfca_communication2d(Front *) C</td>
</tr>
</tbody>
</table>
|--More--
Generate a Flat Profile

- Set a couple of environment variables.
  
  ```
  > export PATH=/bgl/apps/TAUL/tau-2.18/bgl/bin:$PATH
  > export TAU_MAKEFILE= PATH=/bgl/apps/TAUL/tau-2.18/bgl/lib/Makefile.tau-mpi-pdt
  ```

- Compile your code with TAU shell scripts.
  
  ```
  > make CC=tau_cc.sh CXX=tau_cxx.sh F90=`tau_f90.sh -qfixed`
  ```

- Provide the full path to the directory where you want to store the profile files (profile.x.0.0). In your batch job script file, set the environment variable PROFILEDIR.
  
  ```
  # @ arguments = -np 16 -env PROFILEDIR=<profile-dir> -exe ...
  ```

- Submit your batch job script.
- Go to the directory where you store the profile files. Pack the data into packed (.ppk) format.
  
  ```
  > paraprof --pack filename.ppk
  ```

- Launch the GUI to analyze the data.
  
  ```
  > paraprof filename.ppk
  ```
Identify the routines that use the most time
Show Thread Statistics Text Window

<table>
<thead>
<tr>
<th>Name</th>
<th>Total Time</th>
<th>Exclusive</th>
<th>Inclusive</th>
<th>Calls</th>
<th>Child Calls</th>
<th>Inclusive/Call</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFP_Barrier()</td>
<td>24.6</td>
<td>1.9972E8</td>
<td>1.9972E8</td>
<td>193</td>
<td>0</td>
<td>1034799.373</td>
<td></td>
</tr>
<tr>
<td>HELMBO</td>
<td>23.9</td>
<td>1.9469E8</td>
<td>1.9469E8</td>
<td>3716690</td>
<td>0</td>
<td>52.304</td>
<td></td>
</tr>
<tr>
<td>HELMRSO</td>
<td>13.8</td>
<td>1.1231E8</td>
<td>1.1231E8</td>
<td>1332</td>
<td>0</td>
<td>80822.916</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>31.0</td>
<td>5.6953E7</td>
<td>5.6953E7</td>
<td>551373</td>
<td>0</td>
<td>457.544</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>3.1</td>
<td>1.6852E7</td>
<td>1.6852E7</td>
<td>486329</td>
<td>1369562</td>
<td>51.821</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>3.2</td>
<td>1.0768E7</td>
<td>1.0768E7</td>
<td>703273</td>
<td>892230</td>
<td>32.679</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>32.2</td>
<td>2.621E8</td>
<td>2.621E8</td>
<td>582272</td>
<td>471.337</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>1.8</td>
<td>8674457</td>
<td>2.522E8</td>
<td>551373</td>
<td>0</td>
<td>457.544</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>4.2</td>
<td>752951</td>
<td>3.3792E7</td>
<td>792319</td>
<td>886241</td>
<td>171.918</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>1.2</td>
<td>696672</td>
<td>9371128</td>
<td>9371128</td>
<td>113198</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>12.8</td>
<td>6794483</td>
<td>1.0386E9</td>
<td>437860</td>
<td>564384</td>
<td>227.001</td>
<td></td>
</tr>
<tr>
<td>HELMSTATE</td>
<td>1.8</td>
<td>814309</td>
<td>2.8745E7</td>
<td>427681</td>
<td>427681</td>
<td>67.725</td>
<td></td>
</tr>
</tbody>
</table>
Function Data and Comparison Windows
-PROFILECALLPATH

- Generate call path profiles.
- Show the time spend in a routine when it is called by another routine in the calling path.
  - f1 => f2 shows the time spent in f2 when it is called by f1
- Set the environment variable TAU_MAKEFILE to Makefile.tau-callpath-mpi-pdt

[Table and diagram showing call path metrics and timings]
paraprof → Windows → Threads → Call Graph
Blue Gene modern CPUs provide on-chip hardware performance counters that can record several events.
- The number of instructions issued
- The number of L1, L2 and L3 data and instruction cache misses, hits, access, read, write.
- Floating point operations executed.

TAU uses the Performance Data Standard and API (PAPI-Performance Application Programming Interface) to access these performance counters.
Generate Hardware Counter Profile

- Set the environment variable TAU_MAKEFILE to Makefile.tau-multiplecounters-mpi-pdt

- Set the COUNTERx environment variables to specify the type of counter to profile in your job script file.
  
  ```
  # @ arguments = -np 16 -env PROFILEDIR=<profile-dir>
  -env COUNTER1=GET_TIME_OF_DAY \ 
  -env COUNTER2=PAPI_FMA_INS -exe ...
  ```

- It will produce profile files in directories.
  <profile-dir>/MULTI__GET_TIME_OF_DAY
  <profile-dir>/MULTI__PAPI_FMA_INS
Performance Counters
Fast Blue Gene Timers

- Blue Gene systems have a special clock cycle counter that can be used for low overhead timings,
  - BGLTIMERS
    Use fast low-overhead timers on IBM BG/L
  - BGPTIMERS
    Use fast low-overhead timers on IBM BG/P
PerfExplorer

- Framework for parallel performance data mining.
- Enables the development and integration of data mining operations that will be applied to large-scale parallel performance profiles.
- Requires Java Run Time Environment 5
- Requires PerfDMF (Performance Data Management Framework) from TAU.
Running PerfExplorer

- Make sure you have Java5 or better in your PATH.
- Configure PerfDMF. To configure PerfDMF, run `perfdmf_configure`
- Generate .ppk files.
  - `llsubmit tau_app32.run`
    - `paraprof --pack tau_np32.ppk`
  - ...
  - `llsubmit tau_app512.run`
    - `paraprof --pack tau_np512.ppk`
- `paraprof`
  - Add trial to the DB.
    - Trial type: Paraprof Packed Profile
    - Select File(s) -> OK.
    - Uploading Trial.
- `perfexplorer`
  - Choose Experiments.
  - The options under the Chart menu provide analysis.
perfexplorer → Charts → Relative Speedup

Relative Speedup – New Application:Time

Speed Up $S = \frac{t_s}{t_p}$
perfexplorer → Charts → Relative Efficiency

Relative Efficiency - New Application: Time

Efficiency $E = \frac{S}{p}$
New York Blue Visualization Cluster

- Login NYBlue Visualization Cluster name: vis1.bluegene.bnl.gov to vis4.bluegene.bnl.gov
- Add the <tau-dir>/ppc64/bin to your path
- Launch the GUI.

>paraprof filename.ppk
References

- http://www.cs.uoregon.edu/research/tau/home.php
- http://www.cs.uoregon.edu/research/tau/docs/newguide/index.html