Introduction to Extrae/Paraver

George S. Markomanolis
7 August 2019
Extrae/Paraver

- Developed by Barcelona Supercomputing Center
- Extrae for instrumentation
- Paraver for visualization and performance analysis
- Installed version on Summit: v3.7.1
- Module: extrae
# Capability Matrix - Extrae

<table>
<thead>
<tr>
<th>Capability</th>
<th>Profiling</th>
<th>Tracing</th>
<th>Notes/Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI, MPI-IO</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>OpenMP CPU</td>
<td>Yes</td>
<td>Yes</td>
<td>Only GNU</td>
</tr>
<tr>
<td>OpenMP GPU</td>
<td>Yes</td>
<td>Yes</td>
<td>Only with GNU compiler, no OpenACC</td>
</tr>
<tr>
<td>OpenACC</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>CUDA</td>
<td>Yes</td>
<td>Yes</td>
<td>Not advanced</td>
</tr>
<tr>
<td>POSIX I/O</td>
<td>??</td>
<td>??</td>
<td></td>
</tr>
<tr>
<td>POSIX threads</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Memory – app-level</td>
<td>Yes</td>
<td>Yes</td>
<td>Need to use dynamic allocation</td>
</tr>
<tr>
<td>Memory – func-level</td>
<td>Yes</td>
<td>Yes</td>
<td>Need to use dynamic allocation</td>
</tr>
<tr>
<td>Hotspot Detection</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Variance Detection</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Hardware Counters</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Compilation

• Extrae is a bit more complicate to start using it compared to many other tools

• We can have dynamic or static compilation
  – For static, it is required to recompile
  – For dynamic is required to compile with -g, it works even without -g but less information will be instrumented:
How does Extrae work?

• Symbol substitution through LD_PRELOAD
  – We need to use specific libraries based on programming language/model

• Dynamic instrumentation (based on DynInst)

• Static link
Trace Generation Workflow

1) Compile & link with debug info (-g)
2) Don’t strip the binary

Instrumentation

Analysis
Library Selection

- Choose a library depending on the application type
  - The suffix “f” is for Fortran codes

<table>
<thead>
<tr>
<th>Library</th>
<th>Serial</th>
<th>MPI</th>
<th>OpenMP</th>
<th>pthread</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>libseqtrace</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libmpitrace[f]&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libomptrace</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libpttrace</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>libcudatrace</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>libompitrace[f]&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>libptmpitrace[f]&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>libcudampitrace[f]&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

<sup>1</sup> include suffix “f” for Fortran codes
Trace visualization/analysis
+ trace manipulation

Timelines

Goal = Flexibility
No semantics
Programmable

2/3D tables
(Statistics)

Comparative analyses
Multiple traces
Synchronize scales
Extrae XML configuration - MPI

```xml
<mpi enabled="yes">
  <counters enabled="yes" />
</mpi>

<openmp enabled="yes" omp="no">
  <locks enabled="no" />
  <taskloop enabled="no" />
  <counters enabled="yes" />
</openmp>

<cuda enabled="no" />

<pthread enabled="no">
  <locks enabled="no" />
  <counters enabled="yes" />
</pthread>

<callers enabled="yes">
  <mpi enabled="yes">1-3</mpi>
  <sampling enabled="no">1-5</sampling>
  <dynamic-memory enabled="no">1-3</dynamic-memory>
  <input-output enabled="no">1-3</input-output>
  <syscall enabled="no">1-3</syscall>
</callers>

<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="1">
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_FP_OPS
    </set>
    <set enabled="no" domain="all" changeat-time="0">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_SR_INS, PAPI_FP_INS
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>

<buffer enabled="yes">
  <size enabled="yes">5000000</size>
  <circular enabled="no" />
</buffer>

<bursts enabled="no">
  <threshold enabled="yes">5000</threshold>
  <mpi-statistics enabled="yes" />
</bursts>

<sampling enabled="no" type="default" period="50m" variability="10m" />

<dynamic-memory enabled="no">
  <alloc enabled="yes" threshold="32768" />
  <free enabled="yes" />
</dynamic-memory>
```
Execution and Merging

- jsrun -n 64 -r 8 -a 1 -c 1 ./trace.sh ./miniWeather_mpi
- trace.sh:

```bash
#!/bin/bash
export EXTRA_E_HOME=/sw/summit/extrae/3.7.1/rhel7.5-gnu6.4.0
export EXTRA_E_CONFIG_FILE=/full_path/extrae.xml
export LD_PRELOAD=${EXTRA_E_HOME}/lib/libmpitrace.so:$LD_PRELOAD:$*
```

- jsrun -n 64 -r 8 -a 1 -c 1 mpimpi2prv -f TRACE.mpits -e miniWeather_mpi
After the execution with merging

- A folder set-X where X is number 0, 1, etc. with the traces, one folder for every 256 MPI processes
- Files based on the merging output, *.prv, *.pcf, *.row, the first one is the merged trace and the rest information about the trace and the events.
- Now you need to visualize the trace for performance analysis.
- We use the tool Paraver, it is available for Linux, Mac, Windows and already pre-compiled (https://tools.bsc.es/downloads), quite difficult to be built on Power processor. Available on Rhea or your computer.
Paraver on Rhea

% ssh –Y username@rhea.ccs.ornl.gov
% module load paraver
% wxparaver

Location for configuration files: /sw/rhea/paraver/cfgs/
ls /sw/rhea/paraver/cfgs/
burst_mode clustering counters_PAPI CUDA folding General Java mpi OmpSs OpenCL OpenMP pthread sampling+folding scripts software_counters spectral uninstall.sh
Paraver – Load trace
Paraver - Filter trace

Reduce trace size

Click Yes
Paraver - Filter trace

Click Browse and load the filter.xml file
Paraver – Visualize trace

Click Browse and load the filter.xml file
Paraver – Investigating trace

Remove the communication links
Paraver - Zoom

Zoom, left click with mouse and select area moving the cursor horizontally towards right and decide which part we want to study.
Paraver – Computation configuration file

- We load h_comp_time.cfg, File -> Load configuration

Click on Open Control Window

I DON'T UNDERSTAND
Paraver – Computation configuration file

- We load `h_comp_time.cfg`, File -> Load configuration

We can see some iterations.
Paraver – Extract part of the original trace

- Select Filter Trace

- Select for Input the original trace
- Select cut for the execution chain
- Trace options: Use original time to be able to compare between traces and remove last state
- Click Select region and mark the area to cut from the original trace
- Click Apply, the trace will be created and loaded
Paraver – MPI Profile

- Select Hints -> MPI -> MPI Profile

Click Hints -> MPI profile -> Histogram

Zoom

The average under the column Outside MPI represents the parallel efficiency, the value Avg/Max is the load balance and the Max is the communication efficiency.
Paraver - Analyzing the trace - MPI Profile

Select Window properties -> Communication for Type and Maximum bytes sent for Statistic.
Paraver - Computation

- Load the 2dh_usedulduration.cfg for a histogram of the duration for the computation regions.
- A lot of areas are not constituted by vertical lines which shows load imbalance.
- We explore in the next slide what is inside the red circle.
- We select Open Filtered Window and we zoom in the area of red circle.
Paraver - Computation

- We zoom in the first area, we compare with the MPI calls and the 2dp_line_call.cfg
- Only the processes 2-8 execute this part and seems that is not instrumented, thus, it could be from an external library
Paraver Useful Instructions

• Load the 2dh_useful_instructions.cfg
Paraver – Extract part from the original trace

After we click “Select Region...” then select the area from the already opened filtered trace
Paraver – Profile per calling line

We load the 2dp_line_call.cfg, we select open control window and synchronize the new window.
Paraver – Late receivers

We load the late_receivers.cfg and the 2dp_line_call.cfg
Paraver – Late senders

We load the receiver_from_late_sender.cfg and the 2dp_line_call.cfg
MiniWeather MPI+OpenMP

- `jsrun -n 64 -r 8 -a 1 -c 2 ./trace_openmp.sh ./miniWeather_mpi_openmp`
- `Trace_opnemp.sh:
  #!/bin/bash
  export EXTRAE_HOME=/sw/summit/extrae/3.7.1/rhel7.5_gnu6.4.0
  export EXTRAE_CONFIG_FILE=/gpfs/alpine/.../c/extrae_openmp.xml
  export LD_PRELOAD=${EXTRAE_HOME}/lib/libompitrace.so:$LD_PRELOAD
  ## Run the desired program
  $*
- `jsrun -n 64 -r 8 -a 1 -c 2 mpimpi2prv -f TRACE.mpits -e ./miniWeather_mpi_openmp`
Parallel Loops

- Create a new chop file as described before
- Load the parallel_loops.cfg and zoom
Parallel Loops

• Create a new chop file as described before
• Load the parallel_loops.cfg and zoom
Load Balance

- Load OpenMP/analysis/load_balance.cfg
- Load the parallel_loops.cfg and zoom
Load Balance

- Load OpenMP/analysis/load_balance.cfg
Load Balance

- Load OpenMP/views/parallel_functions_useful.cfg and zoom
Paraver - Flush

Ctrl + Zoom
Paraver – Chop bigger area

Because of I/O rank 0, it delays to post the MPI_Irecv and MPI_ISend

Activate communication lines
Right Click -> View -> Communication lines

Ctrl + Zoom on top processes