On-line detection of large-scale parallel application's structure
>10k processes + long runs = large traces

Blind tracing is not an option

Profilers also start presenting issues

Can you even store the data?

How patient are you?
- Past methodology: Filters driven by the expert
  - Get the whole trace
  - Summarize for a global view
  - Focus on a representative region

- Goal: Transfer the expertise to the run-time
Objectives

- **Traces of “100 Mb”**
  - Best describe the application behavior
  - Trade-off: Maximize information / data ratio

- **The challenge?**
  - Intelligent selection of the information

- **How?**
  - On-line analysis framework
    - Decide at run-time what is most relevant
- **Data acquisition**
  - MPItrace (BSC)
    - PMPI wrappers

- **Data transmission**
  - MRNet (U. of Wisconsin)
    - Scalable master / worker
    - Tree topology

- **Data analysis**
  - Clustering (BSC)
    - Find structure of computing regions
- Local trace buffers
- BE threads blocked
- FE periodically collects data
  - Automatic / fixed interval
  - Reduction on tree
- Global analysis
- Propagate results
- Locally emit trace events
Clustering analysis

- Density-based clustering algorithm
  - J. Gonzalez, J. Gimenez, J. Labarta – IPDPS'09
    “Automatic detection of parallel applications computation phases”

- Characterize structure of computing regions

- Using hardware counters data
  - Instructions + IPC
    - Complexity & Performance
  - Any other metric
    - i.e. L1, L2 cache misses
Clusters Performance

<table>
<thead>
<tr>
<th>CLUSTER</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>% TIME</td>
<td>36.29</td>
<td>29.52</td>
<td>10.13</td>
<td>9.68</td>
<td>3.73</td>
<td>1.71</td>
</tr>
<tr>
<td>AVG. BURST DUR. (MS)</td>
<td>220.46</td>
<td>177.70</td>
<td>60.81</td>
<td>29.09</td>
<td>38.71</td>
<td>44.83</td>
</tr>
<tr>
<td>IPC</td>
<td>0.53</td>
<td>0.50</td>
<td>0.62</td>
<td>0.77</td>
<td>0.66</td>
<td>0.59</td>
</tr>
<tr>
<td>MIPS</td>
<td>1210.07</td>
<td>1164.36</td>
<td>1403.19</td>
<td>1743.32</td>
<td>1499.47</td>
<td>1388.24</td>
</tr>
<tr>
<td>L1/M/KINSTR</td>
<td>22.72</td>
<td>32.63</td>
<td>12.65</td>
<td>8.39</td>
<td>16.12</td>
<td>6.86</td>
</tr>
<tr>
<td>L2/M/KINSTR</td>
<td>0.59</td>
<td>1.23</td>
<td>1.08</td>
<td>0.61</td>
<td>1.23</td>
<td>1.73</td>
</tr>
<tr>
<td>MEM. BW (MB/s)</td>
<td>90.77</td>
<td>182.65</td>
<td>193.32</td>
<td>136.33</td>
<td>236.15</td>
<td>295.71</td>
</tr>
</tbody>
</table>

Clusters Distribution Over Time

Code Linking

<table>
<thead>
<tr>
<th>CLUSTER</th>
<th>CODE SECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>solve_nnm.f:[2037 - 2310]</td>
</tr>
<tr>
<td>2</td>
<td>solve_nnm.f:[1478 - 1782]</td>
</tr>
<tr>
<td>3</td>
<td>solve_nnm.f:[2030 - 1782]</td>
</tr>
<tr>
<td>4</td>
<td>solve_nnm.f:[1241 - 1345]</td>
</tr>
<tr>
<td>5</td>
<td>solve_nnm.f:[2771 - 2865]</td>
</tr>
<tr>
<td>6</td>
<td>solve_nnm.f:[2388 - 2489]</td>
</tr>
<tr>
<td>7</td>
<td>solve_nnm.f:[1478 - 1569]</td>
</tr>
<tr>
<td>8</td>
<td>solve_nnm.f:[1607 - 1633]</td>
</tr>
</tbody>
</table>
- Trigger clustering analysis periodically
  - Sequence of structure snapshots

- Compare subsequent clusterings
  - See changes in the application behavior

- Find a representative region
  - Most applications are highly iterative
Select representative

- Compare 2 clusterings, cluster per cluster
  - Inscribe clusters into a rectangle
  - Match those that overlap with a 5% variance
  - Sum of the matched clusters cover the 85% of total computing time

- Stability = N equivalent clusterings “in-a-row”
  - Keep on looking for differences
- Gradually lower requisites if can not be met
  - Best possible region based on “seen” results
60 Mb, 6 iterations
- Clustering time grows with the number of points
  - 5k pts $\rightarrow$ 10 sec, 50k pts $\rightarrow$ 10 min

- Sample a subset of data to cluster (SDBScan)
  - Space: Select a few processes. Full time sequence.
  - Time: Random sampling. Wide covering.

- Classify remaining data
  - Nearest neighbor algorithm
    - Reusing clustering structures
Clustering vs. Classification

All processes

25% random records

32 representatives

15% random records

16 representatives

10% random records

8 representatives + 15% random

75% less data
6s down from 2m

Good quality
Fast analysis
## Experiments

<table>
<thead>
<tr>
<th>Metric</th>
<th>SPECFEM3D</th>
<th>Gromacs</th>
<th>MILC</th>
<th>Zeus-MP</th>
<th>Leslie3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>64</td>
<td>64</td>
<td>245</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Requested trace size</td>
<td>100 Mb</td>
<td>200 Mb</td>
<td>200 Mb</td>
<td>350 Mb</td>
<td>600 Mb</td>
</tr>
<tr>
<td>Full run time</td>
<td>58 m</td>
<td>8 m</td>
<td>5.5 m</td>
<td>10 m</td>
<td>56 m</td>
</tr>
<tr>
<td>Full trace size</td>
<td>3 Gb</td>
<td>20 Gb</td>
<td>5.5 Gb</td>
<td>22 Gb</td>
<td>82 Gb</td>
</tr>
<tr>
<td>Analysis steps</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Clustering time/step</td>
<td>0.6 s</td>
<td>4.5 s</td>
<td>5.85 s</td>
<td>1 s</td>
<td>60 s</td>
</tr>
<tr>
<td>Classification time</td>
<td>1 s</td>
<td>1 s</td>
<td>1 s</td>
<td>1 s</td>
<td>3 s</td>
</tr>
<tr>
<td>Time to get results</td>
<td>15 m</td>
<td>2.5 m</td>
<td>3 m</td>
<td>3 m</td>
<td>12 m</td>
</tr>
<tr>
<td>Traced iterations</td>
<td>35</td>
<td>10</td>
<td>20</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Time/iteration</td>
<td>3 s</td>
<td>500 ms</td>
<td>60 ms</td>
<td>1 s</td>
<td>100 ms</td>
</tr>
</tbody>
</table>

- Important trace size reductions
- Results before the application finishes
- Final trace is representative
Compared vs. Profiles for the whole run

- TAU Performance System (U. of Oregon)

Same overall structure

- Same relevant functions, Avg. HWC’s & Time %
- Most measurement differences under 1%

<table>
<thead>
<tr>
<th>GROMACS user functions</th>
<th>Full run profile (TAU)</th>
<th>Trace segment (MPItrace)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% Time</td>
<td>Kinstr</td>
</tr>
<tr>
<td>do_nonbonded</td>
<td>23.72%</td>
<td>24,709</td>
</tr>
<tr>
<td>solve_pme</td>
<td>10.47%</td>
<td>6,795</td>
</tr>
<tr>
<td>gather_f_bsplines</td>
<td>5.69%</td>
<td>5,286</td>
</tr>
</tbody>
</table>
Example: GROMACS evolution

$\sum$% time matched clusters

Step 1

Step 2

Step 3

Step 4

Step 5

Step 6

Step 7

Step 8
Example: GROMACS structure

- Study load balancing
Initial development

- All data centralized
- Sampling, clustering & classification at front-end
- Bad scaling at large processor counts

>10k tasks

- Sampling at leaves
- Only put together the clustering set
- Broadcast clustering results, classify at leaves
On-line automatic analysis framework
Identify structure and see how evolves
Determine a representative region
Detailed small trace + Periodic reports
Reductions in the time dimension
Scalable infrastructure supports other analyses

Current work
• Spectral analysis (M. Casas): Better delineate the traced region
• Parallel clustering in the tree
• Finer stability heuristic
Thank you for your attention!