On-the-Fly Calculation of Performance Metrics with Adaptive Time Resolution for HPC Compute Jobs

Konstantin Stefanov, Vadim Voevodin
cstef@parallel.ru, vadim@parallel.ru

Research Computing Center
M.V.Lomonosov Moscow State University

Russian Supercomputing Days
Moscow
24.09.2018
Performance Monitoring

• Basic object (job) is dynamic – need to select related data from the whole stream
• Data volume is huge:
  – Poll frequency – the more HZ the better
  – Many different sensors (CPU load, LoadAvg, flops, network counters etc.)
• Much of the data are used only once (calculating aggregates for finished job)
On-the-fly Calculation of Performance Metrics
## Performance Data Stream: Random IO

<table>
<thead>
<tr>
<th>Time</th>
<th>Node</th>
<th>Sensor</th>
<th>Value</th>
<th>Time</th>
<th>Node</th>
<th>Sensor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>21:34:45</td>
<td>1</td>
<td>CPU Load</td>
<td>45</td>
<td>21:34:55</td>
<td>1</td>
<td>CPU Load</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>2679</td>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>2679</td>
</tr>
<tr>
<td>2</td>
<td>CPU Load</td>
<td>49</td>
<td></td>
<td></td>
<td>2</td>
<td>CPU Load</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>IB bytes in</td>
<td>2179</td>
<td></td>
<td></td>
<td>3</td>
<td>CPU Load</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>IB bytes in</td>
<td>1629</td>
<td></td>
<td></td>
<td>4</td>
<td>CPU Load</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>1629</td>
<td>21:34:55</td>
<td>1</td>
<td>CPU Load</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>1629</td>
<td></td>
<td>2</td>
<td>CPU Load</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>2679</td>
<td></td>
<td>3</td>
<td>CPU Load</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IB bytes in</td>
<td>2679</td>
<td></td>
<td>4</td>
<td>CPU Load</td>
<td>45</td>
</tr>
</tbody>
</table>

### Jobs

<table>
<thead>
<tr>
<th>Job</th>
<th>Start</th>
<th>Finish</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20:30:07</td>
<td>21:34:50</td>
<td>1, 3</td>
</tr>
<tr>
<td>2</td>
<td>21:05:37</td>
<td>21:48:23</td>
<td>2, 4</td>
</tr>
<tr>
<td>3</td>
<td>21:34:53</td>
<td>22:08:21</td>
<td>1, 3</td>
</tr>
</tbody>
</table>
Performance Monitoring: Proposed Approach
Reconfiguring on-the-fly to Calculate per-Job Metrics: Job Start

Data
Control

System-wide metrics

Per job metrics

Transmit

Input

Output

Tag

Control

Buffer

Input

Timer

Control

Output

Sensor 1

Control

Output

Sensor N

Control

Resource Manager interaction

Set send address and tag content
Connect Tag and Transmit

Job start

Resource Manager (SLURM)

Job start

Compute node
Reconfiguring on-the-fly to Calculate per-Job Metrics: Job Running
Reconfiguring on-the-fly to Calculate per-Job Metrics: Job finish
Issue:
Restart Monitoring System Parts

• Restart is a big issue:
  – Node agent saves node ID
  – Server part saves job data after the job is finished
Issue:
Restart Node Agent

• Restart is done after a job is finished from a system-wide SLURM epilog script
Issue:
Restart Server Part
(not implemented yet)

• 2 server parts, both have the same data
  – Main part works as described above, save the data right after the job is finished
  – Standby part tries to save job data in 30s after the job is finished
    • If the record for the job is already there, the data are discarded

• Restart both parts with interval longer than the job time limit
Making Time Resolution Adaptive
Problem description and solution

• The more data points the better (especially for short jobs)
• Graphs with many data points are hard to visualize
• Need a trade-off

• Make dynamic resolution:
  – Fine for short jobs
  – Coarse for long jobs
### Short Job

<table>
<thead>
<tr>
<th>Time</th>
<th>23:00:01</th>
<th>23:00:02</th>
<th>23:00:03</th>
<th>23:00:04</th>
<th>23:16:39</th>
<th>23:16:40</th>
<th>23:16:41</th>
<th>23:16:42</th>
<th>23:16:43</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU user, %</td>
<td>100</td>
<td>94</td>
<td>90</td>
<td>90</td>
<td>50</td>
<td>55</td>
<td>50</td>
<td>55</td>
<td>50</td>
</tr>
</tbody>
</table>

...
**Not-so-Short Job**

<table>
<thead>
<tr>
<th>Time</th>
<th>23:00:01</th>
<th>23:00:02</th>
<th>23:00:03</th>
<th>23:00:04</th>
<th>...</th>
<th>23:16:39</th>
<th>23:16:40</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU user, %</td>
<td>100</td>
<td>94</td>
<td>90</td>
<td>90</td>
<td></td>
<td>50</td>
<td>55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>23:00:01</th>
<th>23:00:03</th>
<th>...</th>
<th>23:16:39</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU user, %</td>
<td>97</td>
<td>90</td>
<td></td>
<td>52,5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results:
Dynamic vs Fixed (2 min) Resolution
Results:
Dynamic vs Fixed (2 min) Resolution
Thank you for your attention