PerSyst Monitoring: A scalable Monitoring Tool using performance properties of HPC Systems

Carla Guillen, Wolfram Hesse, Matthias Brehm
Leibniz Supercomputing Centre

Scicomp 17
Thursday, 12th of May 2011
Paris, France.
Agenda

PerSyst Monitoring
- Motivation
- Background (Periscope)
- Implementation and Features
- Results and Interpretation
- Outlook: SuperMUC
Massive parallel systems are an expensive resource

Many users, many different projects...

System Level

Inefficient applications

Continuous system wide monitoring

Bottleneck detection (first hints)

Application Level

Improve codes using fine grain analysis

Motivation | Background | Implementation | Results | SuperMUC
Periscope

- Developed at the Technische Universität München.
- Automatic performance analysis for highly parallel applications.
- Online performance analysis during program execution.
- Based on
  - Instrumented code,
  - Agent systems (hierarchy from Agents),
  - Specification of Properties and
  - Specification of search Strategies.

Periscope and PerSyst Monitoring

PerSyst Monitoring: “Periscope System Monitoring”

- Like Periscope it uses
  - Agent system (hierarchy of agents)
  - Specification of formal *Properties* based on performance hardware counters
  - Specification of *Strategies*

- Main difference: We make system wide monitoring and not application specific monitoring.
  - No instrumented code. This means that the information about application code is not available.
  - Agents measure currently every 10 mins. all cores.
PerSyst Monitoring

Developed in Altix 4700, a system with 9728 Cores. A partition consists of 512 cores that can profit from shared memory.

**High Level Agent:**
- In charge of fault tolerance of other agents
- Synchronises analysis agents, aggregates and
- Stores data.

**Partition Agents:**
- Make the analysis and the synchronisation within a partition.
- Possibility of integrate other agents as plugins.
Properties

- A property is a function of hardware counters which allows us to interpret deficiencies in a prompt manner.
- Based on one or more hardware counters.

Example:

\[
\text{DataCacheStalls} = \text{BE\_EXE\_BUBBLE\_GRALL} - \text{BE\_EXE\_BUBBLE\_GRGR} + \text{BE\_L1D\_FPU\_BUBBLE\_L1D} - \text{BACK\_END\_BUBBLE\_ALL}
\]

Each Property has a

- Property Value: reference hardware counter or property function
- Threshold: \( \theta \)
- Severity:
  \[
  s = \text{funct}(hwc1, hwc2, \ldots, \theta)
  \]
- \( 0 \leq s \leq 1 \)
Strategies

- A strategy is a tree of Properties.
- Search into tree with Properties with \( s > 0 \).
- Refines to more specific information.
Data Selection and Reduction

First reduction through selection of Properties.

Cores
Data Selection and Reduction

Second reduction is to have statistical data per Job. The motivation is to detect applications at system wide level with inefficiencies.

Per job, no matter the size of the job, we calculate meaningful representative data:

- Average
- 11 percentiles, including minimum and maximum
## Excursus: Percentiles

### Example data:

| 7 | 8 | 10 | 7 | 6 | 3 | 6 | 7 | 12 | 8 | 8 | 9 | 4 | 8 | 9 | 8 | 5 | 11 | 10 | 7 |

### Sort your data:

| 3 | 4 | 5 | 6 | 6 | 7 | 7 | 7 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 9 | 9 | 10 | 10 | 11 | 12 |

### Get the percentiles:

<table>
<thead>
<tr>
<th>0th</th>
<th>10th</th>
<th>20th</th>
<th>30th</th>
<th>40th</th>
<th>50th</th>
<th>60th</th>
<th>70th</th>
<th>80th</th>
<th>90th</th>
<th>100th</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

![Value intervals (bins)](value_intervals.png)

![Number of observations](number_of_observations.png)

![Value](value.png)

![Percentiles](percentiles.png)
Interface to Periscope

- Provides data that is able to be read by Periscope.
- Periscope will thus have a hint where to start looking for deficiencies. Periscope will show in the code, where the bottlenecks are.
Overview for administrators

<table>
<thead>
<tr>
<th>Id</th>
<th>Cores</th>
<th>#VALUES</th>
<th>FP_OPS_REI</th>
<th>IA32_INST</th>
<th>NGOPS_REI</th>
<th>USERPCT</th>
<th>SYSTEMPCT</th>
<th>IOWAITPCT</th>
<th>STALL_CYC</th>
</tr>
</thead>
<tbody>
<tr>
<td>492756</td>
<td>512</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492443</td>
<td>512</td>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>489766</td>
<td>510</td>
<td>269</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492575</td>
<td>512</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>489776</td>
<td>510</td>
<td>249</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492759</td>
<td>2300</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>490668</td>
<td>510</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492542</td>
<td>1024</td>
<td>163</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>490967</td>
<td>508</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492202</td>
<td>508</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492344</td>
<td>508</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492667</td>
<td>508</td>
<td>237</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492759</td>
<td>1024</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492715</td>
<td>1024</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492540</td>
<td>1024</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492543</td>
<td>1024</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492777</td>
<td>500</td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491504</td>
<td>768</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491774</td>
<td>500</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492077</td>
<td>768</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491627</td>
<td>500</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492250</td>
<td>500</td>
<td>283</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491717</td>
<td>903</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>491671</td>
<td>768</td>
<td>144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492334</td>
<td>576</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492805</td>
<td>750</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492333</td>
<td>768</td>
<td>144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492086</td>
<td>576</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492485</td>
<td>576</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>492970</td>
<td>512</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average of severity for the entire job for this property (w.r.t. time, cores)
Results and Interpretation: Job summary

List of Properties

- Every box is the average w.r.t. cores at one timestamp
- Average of the entire job (w.r.t. timestamps and cores)

At a timestamp, display of severity distribution of cores
Detecting an application with problems

**Motivation**

**Background**

**Implementation**

**Results**

**SuperMUC**

![Diagram](image-url)
Detecting an application with problems.

Legend: Percentile

Motivation | Background | Implementation | Results | SuperMUC

Floating point operations

System calls (%)

Time
# Altix 4700 to SuperMUC

<table>
<thead>
<tr>
<th></th>
<th>Altix 4700</th>
<th>SuperMUC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>Intel Itanium 64</td>
<td>Westmere, Sandybridge</td>
</tr>
<tr>
<td><strong>Number of cores</strong></td>
<td>~10,000</td>
<td>&gt;130,000</td>
</tr>
<tr>
<td><strong>Shared memory</strong></td>
<td>Partition (512 Cores)</td>
<td>Nodes (40/16 Cores)</td>
</tr>
<tr>
<td><strong>Monitoring group</strong></td>
<td>Partition</td>
<td>Islands</td>
</tr>
<tr>
<td><strong>Monitoring counters at a time</strong></td>
<td>12 Hardware counters</td>
<td>4 Hardware counters</td>
</tr>
<tr>
<td><strong>Cycles</strong></td>
<td>Constant</td>
<td>Variable per Job</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
<td>pfmon</td>
<td>IBM Tools (TEAL, SCI, hpctool kit)</td>
</tr>
</tbody>
</table>

---

12th of May, 2011. Scicomp 17, Paris, France
Targeting SuperMUC and Petascale monitoring

In our case raw data has to be sent to a centralised agent in order to calculate percentiles per Job.

Problem: This doesn’t scale!

In order to calculate percentiles of Job 1, we have to get the detailed information from Agent 1, 2 and 3.
Possible solutions to this problem:

- Aggregation of a Job’s data subset “in situ”. Instead of using percentiles, use discrete intervals (histogram bins).
- Use a layer of agents that collect job information. Communication is targeted to reduce job information transfer per agent.

SuperMUC
On top of the layer of collector agents, we would need a tree hierarchy structure. (MRNet, SCI)
Thank you for your attention!

Questions?