Scalable and Live Trace Processing with Kieker Utilizing Cloud Computing

Florian Fittkau, Jan Waller, Peer Brauer, and Wilhelm Hasselbring

2013-11-28
Knowledge of the internal behavior often gets lost
Application-level monitoring
Can cause large impact on the performance
High-throughput trace processing reducing the overhead
Cloud infrastructures
Figure 1: Macro view on landscape level showing the communication between applications in the PubFlow (http://pubflow.de) software landscape [FWWH13]
System Level Perspective

ExplorViz

(a) Macro view visualizing four components of jPetStore
(b) Relationship view with opened service component

Figure 2: Mockup of system level perspective on the example of jPetStore for demonstrating the exploration concept [FWWH13]
Figure 3: Activities in our ExplorViz approach for live trace visualization of large software landscapes [FWWH13]
Basic Approach

Scalable Trace Processing Architecture

Figure 4: Overview on our general trace processing architecture
Chaining of Analysis Workers

Figure 5: Example for chaining of analysis workers
Chaining of Analysis Workers

Scalable Trace Processing Architecture

- Levels of chaining are not restricted to one or two
- On each level, the number of analysis workers should be lower than before
- SLAStic can be used to scale each group of analysis workers
- SLAStic can be extended to decide whether a new analysis worker level should be opened
Figure 6: Our high-throughput tuned version of *Kieker.Monitoring*
Figure 7: Our high-throughput tuned version of **Kieker.Analysis**
Experimental Setup

Preliminary Performance Evaluation

- Extended version of the monitoring overhead benchmark MooBench [WH12]
- 2 virtual machines (VMs) in our OpenStack private cloud
- Each physical machine in our private cloud contains two 8-core Intel Xeon E5-2650 (2 GHz) processors, 128 GiB RAM, and a 10 Gbit network connection
### Results for Kieker 1.8

#### Preliminary Performance Evaluation

<table>
<thead>
<tr>
<th></th>
<th>No inst.</th>
<th>Deactiv.</th>
<th>Collecting</th>
<th>Writing</th>
<th>Reconst.</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean</strong></td>
<td>2 500.0k</td>
<td>1 176.5k</td>
<td>141.8k</td>
<td><strong>39.6k</strong></td>
<td>0.5k</td>
<td>0.5k</td>
</tr>
<tr>
<td><strong>95% CI</strong></td>
<td>± 371.4k</td>
<td>± 34.3k</td>
<td>± 2.0k</td>
<td>± 0.4k</td>
<td>± 0.001k</td>
<td>± 0.001k</td>
</tr>
<tr>
<td><strong>Q₁</strong></td>
<td>2 655.4k</td>
<td>1 178.0k</td>
<td>140.3k</td>
<td>36.7k</td>
<td>0.4k</td>
<td>0.4k</td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td>2 682.5k</td>
<td>1 190.2k</td>
<td>143.9k</td>
<td>39.6k</td>
<td>0.5k</td>
<td>0.5k</td>
</tr>
<tr>
<td><strong>Q₃</strong></td>
<td>2 700.4k</td>
<td>1 208.0k</td>
<td>145.8k</td>
<td>42.1k</td>
<td>0.5k</td>
<td>0.5k</td>
</tr>
</tbody>
</table>

**Table 1**: Throughput for Kieker 1.8 (traces per second)
## Results for Our Tuned Kieker Version

### Preliminary Performance Evaluation

<table>
<thead>
<tr>
<th></th>
<th>No inst.</th>
<th>Deactiv.</th>
<th>Collecting</th>
<th>Writing</th>
<th>Reconst.</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean</strong></td>
<td>2688.2k</td>
<td>770.4k</td>
<td>136.5k</td>
<td>115.8k</td>
<td>116.9k</td>
<td>112.6k</td>
</tr>
<tr>
<td><strong>95% CI</strong></td>
<td>± 14.5k</td>
<td>± 8.4k</td>
<td>± 0.9k</td>
<td>± 0.7k</td>
<td>± 0.7k</td>
<td>± 0.8k</td>
</tr>
<tr>
<td><strong>Q₁</strong></td>
<td>2713.6k</td>
<td>682.8k</td>
<td>118.5k</td>
<td>102.5k</td>
<td>103.3k</td>
<td>98.4k</td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td>2720.8k</td>
<td>718.1k</td>
<td>125.0k</td>
<td>116.4k</td>
<td>116.6k</td>
<td>114.4k</td>
</tr>
<tr>
<td><strong>Q₃</strong></td>
<td>2726.8k</td>
<td>841.0k</td>
<td>137.4k</td>
<td>131.9k</td>
<td>131.3k</td>
<td>132.4k</td>
</tr>
</tbody>
</table>

**Table 2**: Throughput for our high-throughput tuned Kieker version (traces per second)
Figure 8: Comparison of the resulting response times
Threats to Validity

Preliminary Performance Evaluation

- Only on one type of virtual machine/hardware
- Virtualized cloud environment might resulted in unfortunate scheduling effects
- Minimized this threat by prohibiting over-provisioning
Related Work

- Dapper
- Magpie
- X-Trace
Future Work

Future Work and Conclusions

- Evaluate the scalability and performance of our trace processing architecture in our private cloud environment
- Search for guidelines which number of levels of analysis workers is suitable in which situation
- Feedback our high-throughput tunings into Kieker
Conclusions

Future Work and Conclusions

- Enabling scalable monitoring in the cloud
- Live trace processing for ExplorViz\(^1\)
- Improved the analysis performance of Kieker by a factor of 250

\(^1\)http://www.explorviz.net
Florian Fittkau, Jan Waller, Christian Wulf, and Wilhelm Hasselbring. Live trace visualization for comprehending large software landscapes: The ExplorViz approach.

Jan Waller and Wilhelm Hasselbring. A comparison of the influence of different multi-core processors on the runtime overhead for application-level monitoring.