

Scalable and Live Trace Processing with Kieker Utilizing Cloud Computing

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

2013-11-28

ExplorViz

1. Introduction
2. ExplorViz
3. Scalable Trace Processing Architecture
4. High-Throughput Tunings for Kieker
5. Preliminary Performance Evaluation
6. Related Work
7. Future Work and Conclusions
8. References

- ▶ 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

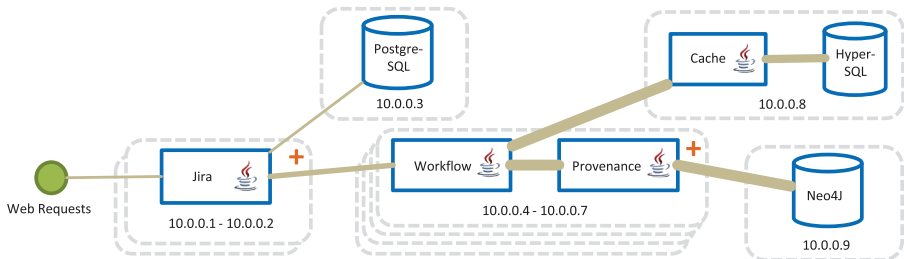
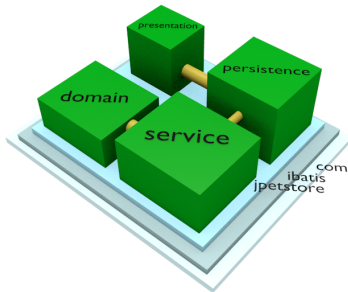
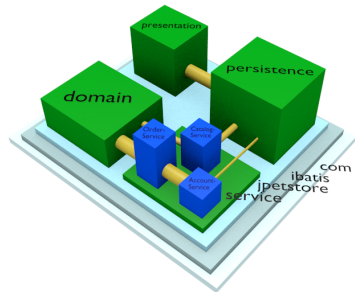


Figure 1 : Macro view on landscape level showing the communication between applications in the PubFlow (<http://pubflow.de>) software landscape [FWWH13]



(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]

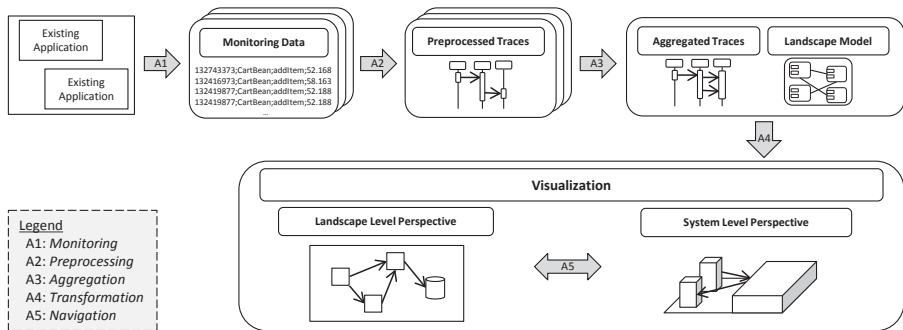


Figure 3 : Activities in our ExplorViz approach for live trace visualization of large software landscapes [FWWH13]

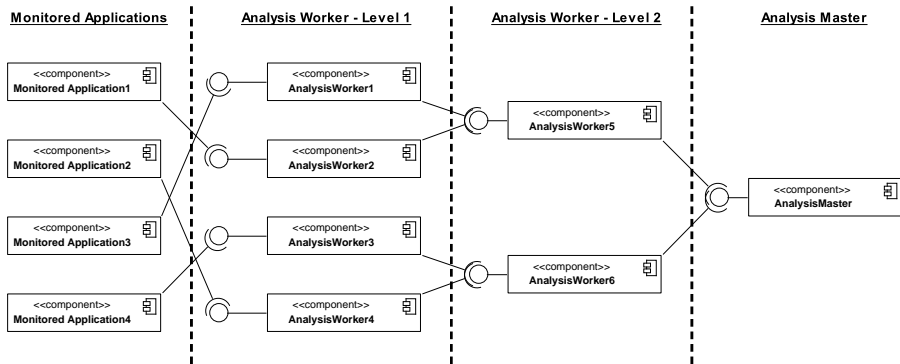


Figure 5 : Example for chaining of analysis workers

- ▶ Levels of chaining are not restricted to one or two
- ▶ On each level, the number of analysis workers should be lower than before
- ▶ SLAstatic can be used to scale each group of analysis workers
- ▶ SLAstatic can be extended to decide whether a new analysis worker level should be opened

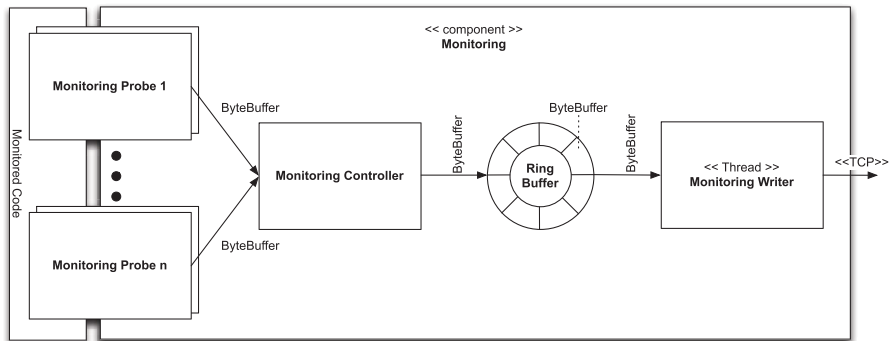


Figure 6 : Our high-throughput tuned version of *Kieker.Monitoring*

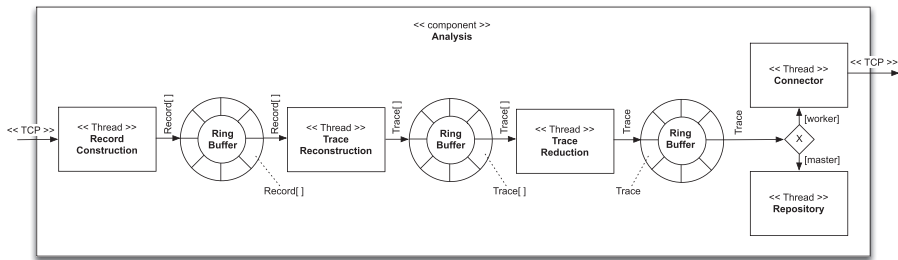


Figure 7 : Our high-throughput tuned version of *Kieker.Analysis*

- ▶ 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

	No inst.	Deactiv.	Collecting	Writing	Reconst.	Reduction
Mean	2 500.0k	1 176.5k	141.8k	39.6k	0.5k	0.5k
95% CI	$\pm 371.4k$	$\pm 34.3k$	$\pm 2.0k$	$\pm 0.4k$	$\pm 0.001k$	$\pm 0.001k$
Q ₁	2 655.4k	1 178.0k	140.3k	36.7k	0.4k	0.4k
Median	2 682.5k	1 190.2k	143.9k	39.6k	0.5k	0.5k
Q ₃	2 700.4k	1 208.0k	145.8k	42.1k	0.5k	0.5k

Table 1 : Throughput for Kieker 1.8 (traces per second)

	No inst.	Deactiv.	Collecting	Writing	Reconst.	Reduction
Mean	2 688.2k	770.4k	136.5k	115.8k	116.9k	112.6k
95% CI	± 14.5k	± 8.4k	± 0.9k	± 0.7k	± 0.7k	± 0.8k
Q ₁	2 713.6k	682.8k	118.5k	102.5k	103.3k	98.4k
Median	2 720.8k	718.1k	125.0k	116.4k	116.6k	114.4k
Q ₃	2 726.8k	841.0k	137.4k	131.9k	131.3k	132.4k

Table 2 : Throughput for our high-throughput tuned Kieker version (traces per second)

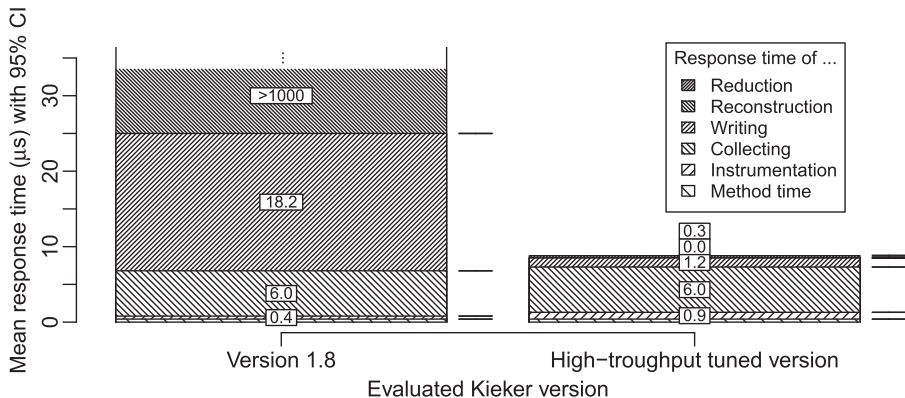


Figure 8 : Comparison of the resulting response times

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

- ▶ Dapper
- ▶ Magpie
- ▶ X-Trace

- ▶ 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

- ▶ Enabling scalable monitoring in the cloud
- ▶ Live trace processing for ExplorViz¹
- ▶ 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.

In Proceedings of the 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013). IEEE Computer Society, 2013.



Jan Waller and Wilhelm Hasselbring.

A comparison of the influence of different multi-core processors on the runtime overhead for application-level monitoring.

In Multicore Software Engineering, Performance, and Tools (MSEPT 2012), pages 42–53. Springer, 2012.