K8-Scalar: a workbench to compare autoscalers for container-orchestrated services (Artifact)

Wito Delnat
Dept. of Computer Science, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
wito.delnat@gmail.com

Thomas Heyman
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
thomas.heyman@cs.kuleuven.be

Wouter Joosen
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
wouter.joosen@cs.kuleuven.be

Davy Preuveneers
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
davy.preuveneers@cs.kuleuven.be

Ansar Rafique
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
ansar.rafique@cs.kuleuven.be

Eddy Truyen
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
eddy.truyen@cs.kuleuven.be

Dimitri Van Landuyt
imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
dimitri.vanlanduyt@cs.kuleuven.be

Abstract

This artifact is an easy-to-use and extensible workbench exemplar, named K8-Scalar, which allows researchers to implement and evaluate different self-adaptive approaches to autoscaling container-orchestrated services. The workbench is based on Docker, a popular technology for easing the deployment of containerized software that also has been positioned as an enabler for reproducible research. The workbench also relies on a container orchestration framework: Kubernetes (K8s), the de-facto industry standard for orchestration and monitoring of elastically scalable container-based services. Finally, it integrates and extends Scalar, a generic testbed for evaluating the scalability of large-scale systems with support for evaluating the performance of autoscalers for database clusters.

The associated scholarly paper presents (i) the architecture and implementation of K8-Scalar and how a particular autoscaler can be plugged in, (ii) sketches the design of a Riemann-based autoscaler for database clusters, (iii) illustrates how to design, setup and analyze a series of experiments to configure and evaluate the performance of this autoscaler for a particular database (i.e., Cassandra) and a particular workload type, (iv) and validates the effectiveness of K8-scalar as a workbench for accurately comparing the performance of different auto-scaling strategies. Future work includes extending K8-Scalar with an improved research data management repository.

2012 ACM Subject Classification General and reference → Experimentation, Information systems → Middleware for databases, Software and its engineering → Application specific development environments, Software and its engineering → Software creation and management, Software and its engineering → Software post-development issues

Keywords and phrases Container orchestration, autoscalers, experimentation exemplar

Digital Object Identifier 10.4230/DARTS.4.1.2

Related Article Wito Delnat, Eddy Truyen, Ansar Rafique, Dimitri Van Landuyt and Wouter Joosen, "K8-Scalar: a workbench to compare autoscalers for container-orchestrated database clusters", in Proceedings of the 2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2018), ACM, pp.7, 2018.
https://doi.org/10.1145/3194133.3194162

Related Conference 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2018), May 28-29, 2018, Gothenburg, Sweden

1 Scope

The K8-Scalar artifact is an easy-to-use and extensible workbench exemplar for implementing and evaluating different self-adaptive approaches to autoscaling container-orchestrated services. Container technology such as Docker [5] and container orchestration frameworks such as Kubernetes [11] have been pushed by the Linux Foundation as the contemporary set of platforms and tools for deploying and managing cloud-native applications [16]. Docker is a popular tool for deploying software using fast and light-weight Linux containers [21, 19]. Moreover, Docker's approach to image distribution, which enables developers to store a specific configuration of a software component as a portable and light-weight Docker image that can be downloaded from a local or central Docker registry. The latter advantage also helps to address the above mentioned necessity for repeating experiments in identically configured testing environments to improve reproducible research [2]. Container technology also has the potential to improve consistency of resource allocation parameters across different computers with the same CPU clock frequency.

K8-Scalar has been used and validated in the context of autoscalers for database clusters. The motivation is that effective autoscaler development for databases is a field that is heavily based on heuristics and domain knowledge. The development of autoscalers for databases must thus be finetuned according to the concrete application and scenario in which it will be applied, and therefore it is important to have automated solutions that allow the evaluation of diverse alternatives.

Although autoscalers for database clusters [22, 4, 20] or multi-tier applications [1] have been researched, developing an effective autoscaler for databases is still an art, rather than a science. First, there is no one-size-fits-all solution as autoscalers must be specifically customized for specific databases. For example, auto-scaling makes only sense when a database cluster is load balanced such that adding a new database instance will reduce the load of an existing instance. The load balancing algorithms that are specifically designed for that purpose are database-specific, however, and therefore the autoscaler's design must take into account how fast load balancing algorithm adapt to requested cluster reconfigurations. Secondly, adding or removing an instance should be cost-efficient in the sense that it should not require extensive (re)shuffling of data across existing and new database instances. Thirdly, detecting imminent SLA violations accurately requires multiple type of metrics to be monitored and analyzed simultaneously so that no wrong scaling decisions are triggered by temporary spikes in resource usage or performance metrics. Such wrong scaling decisions can be very costly an actually hurt elasticity instead of improving it.
The artifact fully supports the following contributions of the associated scholarly paper:

- K8-Scalar integrates and customizes Scalar [8], a generic platform for evaluating the scalability of large-scale systems, with support for **evaluating autoscalers for container-orchestrated database clusters**.
- K8-Scalar extends Kubernetes with an advanced autoscaler for database clusters, based on the Riemann event processor [18] that allows for simultaneous analysis of multiple metrics and composition of multiple event-based conditions. This Advanced Riemann-Based Autoscaler (ARBA) comes with a set of elastic scaling policies, which are based on the Universal Law of Scalability [6, 7] and the Law of Little [15], and that have been implemented and evaluated in the context of a case study on using Cassandra in a Log Management-as-a-Service platform [17].
- The paper illustrates how to design, setup and analyze a series of experiments to configure and evaluate the performance of a Riemann-based autoscaler for a particular database (i.e., Cassandra [3]) and a particular workload type

The artifact partially supports the following contribution of the scholarly paper:

- The paper validates the effectiveness of K8-scalar as a workbench for accurately comparing the performance of different auto-scaling strategies. This validation has been performed as part of a master thesis. The artifact includes all the code for running the experiments, but it does not include the collected research data, which is lost because the student did not commit the research data to our research data repository.

2 Content

The artifact is stored and publicly available on GitHub at the following URL: `https://github.com/k8-scalar/k8-scalar`. The current release includes:

1. a detailed, step-by-step hands-on tutorial that relies on Helm, a command-line interface and run-time configuration management server for creating and managing the *Helm charts* [10]. A Helm chart is a highly-configurable deployment package that encapsulates inter-dependent Kubernetes objects such as services, configuration setting or authentication credentials. The tutorial presents extensive and easy-to-following instructions for the following steps:
   - Setting up Kubernetes in a development environment using Minikube [12] or across a cluster of machines using the universal kubeadm deployment tool [14].
   - Deploying the Kubernetes's monitoring service Heapster [9] with a Grafana visual monitoring dashboard.
   - Deploying the Cassandra database as a Kubernetes StatefulSet object [13].
   - Developing and configuring a specific Scalar experiment for the Cassandra database and a specific type of workload (e.g 90% reads and 10% writes).
   - Building a Docker image of the Scalar experiment.
   - Deploying the Scalar experiment as a StatefulSet.
   - Running the Scalar experiment with a different linearly increasing workload profiles for determining the appropriate resource thresholds where elastic scaling actions should be triggered in order to avoid imminent SLA violations.
   - Configuring a concrete autoscaling strategy of the included Advanced Rieman-Based Autoscaler (ARBA) and deploying it.
   - Running a Scalar experiment to evaluate the behavior of a particular autoscaling strategy. The Grafana visual monitoring dashboard allows to monitor resource usage and scaling actions at run-time. Scalar's statistical results can be further analyzed off-line. Scalar also monitors the performance of the experiment itself in order to detect any undesired scalability or performance bottlenecks in the Scalar code itself.

2. A library with development and operational code artifacts of K8-Scalar. This archive thus consists of two logical parts:
   - A `development` directory containing the following subdirectories:
     a. The `cassandra` sub-directory that contains a Dockerfile for building the Cassandra image that we have used in our experiments.
     b. A set of sub-directories that encapsulate development code for some of the MAPE-K components of the ARBA autoscaler:
        - The `riemann` sub-directory that contains the code of ARBA's Analyzer component (which is dependent on Kubernetes' monitoring service *Heapster*[9] and is written on top of the *Riemann event processor* [18].
        - The `scaler` sub-directory that contains the code of the Executor component for performing scaling actions of the Cassandra database.
     c. The set of sub-directories with development code for K8-Scalar experiments:
        - The `scalar` sub-directory that contains all the code for extending Scalar with support for (i) evaluating autoscalers for the Cassandra databases and (ii) mapping SLAs to resource thresholds.
        - The `example-experiment` sub-directory which offers a template for configuring a specific K8-Scalar experiment (e.g. with a specific particular workload profile).
     d. A set of sub-directories with development code for configuring a number of Kubernetes tools on which K8-Scalar relies:
        - The `grafana` sub-directory that contains a slightly customized Dockerfile for building the monitoring dashboard image that visualizes monitoring data from Heapster.
        - The `helm` directory that contains configurations for a running Kubernetes cluster such that the Helm software works properly on top of that Kubernetes cluster.

     The above sub-directories, except the `helm` directory, also contain a Dockerfile for building a Docker image of the respective software component in these sub-directories.
   - An `operations` directory containing the following sub-directories with Helm packages for deploying the following Kubernetes resources:
     a. The Cassandra-cluster in the `cassandra-cluster` directory.
     b. The Heapster monitoring service in the `monitoring-core` directory.
     c. The ARBA autoscaler in the `arba` subdirectory which deploys the riemann-based Analyzer component with a particular strategy configured and the Scaler component.
     d. The Scalar experiment-controller in the `experiment-controller` directory.
3. The development and operations artifacts for all experiments presented in the scholarly paper in the `experiments/LMaaS` sub-directory.
4. The `docs` directory containing the documentation of Scalar.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://github/k8-scalar/k8-scalar`. The above mentioned hands-on tutorial is available at `https://github.com/k8-scalar/k8-scalar/blob/master/README.md`.

4 Tested platforms

K8-Scalar runs on multiple platforms: Linux VMs, Linux bare metal, OS X, and Windows. It has been tested and used extensively on Linux Ubuntu Xenial VMs and with the *kubeadm* universal

Kubernetes deployment tool [14]. The detailed hands-on experience for running K8-Scalar on a development desktop computer using the *minikube* deployment tool [12] runs on Linux bare metal, Windows 7, Windows 10, and Mac OS . It has been tested on Mac OS and Windows 10. System requirements for running the hands-on tutorial:

- Your local machine should support VT-x virtualization
- To run the Kubernetes cluster on your local machine, a VM with 1 CPU core and 2GB is memory is sufficient but the cassandra instances will not fit.
- To run 1 cassandra instance, a VM with minimally 2 virtual CPU cores and 4GB **virtual** memory must be able to run on your machine. In order to run 2 Cassandra instances, 4 virtual CPU cores and 8GB of virtual memory is needed.

Disclaimer. The minikube-based setup of the tutorial is not suitable for running scientific experiments. Minikube only supports kubernetes clusters with one worker node (i.e. the minikube VM). It is better to run the different components of the K8-Scalar architecture on different VMs as illustrated in Section 3 of the related scholarly paper. Kubeadm is needed for setting up Kubernetes clusters on multiple VMs.

5 License

The artifact is available under the Apache License version 2.0 (`http://www.apache.org/licenses/LICENSE-2.0`.

6 MD5 sum of the artifact

f7a3f4aa8cc0f64c8b8f0b162bda8816

7 Size of the artifact

0.19 GiB

Acknowledgements. We would like to thank the SEAMS 2018 reviewers for their helpful comments. This research is partially funded by the Research Fund KU Leuven and the IWT-SBO DeCoMAdS project.

References

1 Cornel Barna, Hamzeh Khazaei, Marios Fokaefs, and Marin Litoiu. Delivering Elastic Containerized Cloud Applications to Enable DevOps. *Proceedings - 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2017*, pages 65–75, 2017. `doi:10.1109/SEAMS.2017.12`.

2 Carl Boettiger. An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79, January 2015. URL: `http://doi.acm.org/10.1145/2723872.2723882`, `doi:10.1145/2723872.2723882`.

3 Apache Cassandra. Understanding the architecture. URL: `http://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archTOC.html`, accessed 2018-01-29, 2018.

4 Ioannis Konstantinou Dimitrios Tsoumakos and Christina Boumpouka. *Automated, Elastic Resource Provisioning for NoSQL Clusters Using TIRAMOLA*. IEEE, 2013. Conference paper at Delft, Netherlands.

5 Docker. Docker engine. `https://docs.docker.com/engine/`, accessed 2018-01-24, 2018.

6 Neil J. Gunther. *Guerrilla Capacity Planning*. Springer-Verlag, Heidelberg, Germany, 2007.

7 Neil J. Gunther. A general theory of computational scalability based on rational functions. *arXiv preprint arXiv:0808.1431*, 2008.

8 Thomas Heyman, Davy Preuveneers, and Wouter Joosen. Scalar: Systematic scalability analysis with the universal scalability law. In *2014 International Conference on Future Internet of Things and Cloud*, pages 497–504, 8 2014. URL: `https://lirias.kuleuven.be/handle/123456789/460752`.

9 Kubernetes. Heapster. URL: `https://github.com/kubernetes/heapster/blob/`

master/docs/storage-schema.md, accessed 2018-01-26, 2017.

10 Kubernetes. The kubernetes package manager. URL: https://github.com/kubernetes/helm, accessed 2018-01-26, 2018.

11 Kubernetes. Production-grade container orchestration. URL: https://kubernetes.io/, accessed 2018-01-23, 2018.

12 Kubernetes. Running kubernetes locally via minikube. URL: https://kubernetes.io/docs/getting-started-guides/minikube/, accessed 2018-03-16, 2018.

13 Kubernetes. StatefulSets. https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/, 2018. Accessed: January 23 2018.

14 Kubernetes. Using kubeadm to Create a Cluster. URL: https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/, accessed 2018-01-29, 2018.

15 John DC Little and Stephen C Graves. Little's law. In *Building intuition*, pages 81–100. Springer, 2008.

16 Ron Miller. 36 companies agree to a kubernetes certification standard. https://techcrunch.com/2017/11/13/the-cncf-just-got-36-companies-to-agree-to-a-kubernetes-certification-standard/, 2017.

17 Ansar Rafique, Dimitri Van Landuyt, and Wouter Joosen. Persist: Policy-based data management middleware for multi-tenant saas leveraging federated cloud storage. *Journal of Grid Computing*, Mar 2018. URL: https://doi.org/10.1007/s10723-018-9434-6, doi:10.1007/s10723-018-9434-6.

18 Riemann. Riemann monitors distributed systems. URL: http://riemann.io/, accessed 2017-08-08, 2017.

19 Prateek Sharma, Lucas Chaufournier, Prashant Shenoy, and Y C Tay. Containers and Virtual Machines at Scale: A Comparative Study. In *Proceedings of the 17th International Middleware Conference*, Middleware '16, pages 1:1—-1:13, New York, NY, USA, 2016. ACM. URL: http://doi.acm.org/10.1145/2988336.2988337, doi:10.1145/2988336.2988337.

20 Macro Netto et al. Vincent Emeakaroha. *Towards autonomic detection of SLA violations in Cloud infrastructures.* Future Generation Computer Systems, 2012. CVolume 28, Issue 7, July 2012, Pages 1017-1029.

21 Miguel Gomes Xavier, Marcelo Veiga Neves, and Cesar Augusto Fonticielha De Rose. A Performance Comparison of Container-Based Virtualization Systems for MapReduce Clusters. *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 299–306, 2014. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6787290, doi:10.1109/PDP.2014.78.

22 L. Zhao, S. Sakr, and A. Liu. A framework for consumer-centric sla management of cloud-hosted databases. *IEEE Transactions on Services Computing*, 8(4):534–549, July 2015. doi:10.1109/TSC.2013.5.