TeaStore - A Micro-Service Reference Application for Performance Engineers

Demonstration Paper

Jóakim v. Kistowski
University of Würzburg
Würzburg, Germany
joakim.kistowski@uni-wuerzburg.de

Simon Eismann
University of Würzburg
Würzburg, Germany
simon.eismann@uni-wuerzburg.de

Johannes Grohmann
University of Würzburg
Würzburg, Germany
johannes.grohmann@uni-wuerzburg.de

Norbert Schmitt
University of Würzburg
Würzburg, Germany
norbert.schmitt@uni-wuerzburg.de

André Bauer
University of Würzburg
Würzburg, Germany
andre.bauer@uni-wuerzburg.de

Samuel Kounev
University of Würzburg
Würzburg, Germany
samuel.kounev@uni-wuerzburg.de

ABSTRACT

Performance engineering researchers propose and employ various methods to analyze, model, optimize and manage the performance of modern distributed applications. In order to evaluate these methods in realistic scenarios, researchers rely on reference applications. Existing testing and benchmarking applications are usually difficult to set up and either outdated, designed for specific testing scenarios, or do not offer the necessary degrees of freedom.

In this paper, we present the TeaStore, a micro-service-based reference application. The TeaStore offers multiple services with various performance characteristics and a high degree of freedom regarding its deployment and configuration to be used as a cloud reference application for researchers. The TeaStore is designed for the evaluation of performance modeling and resource management techniques. We invite researchers to use the TeaStore and provide it open-source, extendable, easily deployable and monitorable.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **Software and its engineering** → *Software architectures*;

KEYWORDS

Micro-service Architecture, Reference Application

ACM Reference Format:
Jóakim v. Kistowski, Simon Eismann, Johannes Grohmann, Norbert Schmitt, André Bauer, and Samuel Kounev. 2019. TeaStore - A Micro-Service Reference Application for Performance Engineers: Demonstration Paper. In *Tenth ACM/SPEC International Conference on Performance Engineering (ICPE '19), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3302541.3311966

ICPE '19 Companion, April 7–11, 2019, Mumbai, India

ACM ISBN 978-1-4503-6286-3/19/04.
https://doi.org/10.1145/3302541.3311966

1 INTRODUCTION

Modern distributed micro-service applications have complex performance characteristics, as the constituent services feature different performance properties and are short-lived compared to traditional software components. These applications are usually deployed in container orchestration frameworks, such as Kubernetes, Docker Swarm or cloud provider specific solutions. These frameworks add another layer of complexity by autoscaling the service containers, regenerating failing containers or by the introduction of monitoring or service mesh sidecars.

Current research employs many analysis, modeling, optimization, and management approaches that aim to tackle this challenging performance behavior. Verifying, comparing, and evaluating the results of such research is difficult. To enable practical evaluation, researchers need a software application that they can deploy as reference and that offers realistic degrees of freedom. The reference application must also feature sufficient complexity regarding performance behavior to warrant optimizing it in the first place. Finding such an application and performing the necessary experiments is often difficult. The software in question should be open source, available for instrumentation, and should have reproducible results, all while being indicative of how the evaluated research would affect applications in production use.

Real-world distributed software is usually proprietary and cannot be used for experimentation. Existing test and reference software, on the other hand, is usually explicitly created for evaluating a single contribution, which makes comparisons difficult. Other existing and broadly used test software does not offer the necessary degrees of freedom and is often manually adapted. Some of the most widely used test and reference applications, such as RUBiS [1] or Dell DVD Store [2], are outdated and therefore not representative of modern real-world applications. Reference applications from industry vendors, such as the Sock Shop by WeaveWorks [3], usually use a modern technology stack but are built to showcase a specific software solution and do not pose the performance challenges that current research focuses on.

[1] OW2 Consortium. 2008.RUBiS User's Manual.
[2] https://linux.dell.com/dvdstore/
[3] https://github.com/microservices-demo/microservices-demo

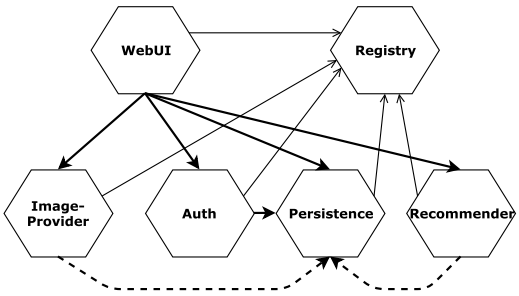

Figure 1: TeaStore Architecture.

We present TeaStore, a micro-services-based test and reference application that can be used as a benchmarking framework by researchers. TeaStore consists of five different services, each featuring unique performance characteristics and bottlenecks. Due to the services' different resource usage profiles, the TeaStore poses interesting challenges in the area of performance modeling, autoscaling, and energy efficiency. It is designed to be scalable and to support both distributed and local deployments. Additionally, its architecture supports run-time scalability as services and service instances can be added, removed, and replicated at run-time.

2 THE TEASTORE

The TeaStore is an online store for tea and tea-related utilities. Users can browse the available products by category and look at individual products. After logging in, the user can add items to the shopping cart, modify the content of the shopping cart and checkout by entering shipping and payment information. Previous orders can be tracked on the user's profile page. After the user is finished, he can log out. The TeaStore displays advertisements for other products based on the user's previous orders, his current shopping cart and the item/category he is currently looking at.

In addition to these regular operations, the TeaStore's user interface provides an overview of all running service instances and the option to populate the database with an adjustable number of categories, products per category, users, and average number of previous orders per user. These operations should not be part of any benchmark but simplify measurement setup.

The TeaStore consists of five distinct services and a Registry service as shown in Figure 1. All services communicate with the Registry. Additionally, the WebUI service issues calls to the Image-Provider, Authentication, Persistence and Recommender services.

The Image provider and Recommender both connect to a provided interface at the Persistence service. However, this is only necessary on startup (dashed lines). The Image provider must generate an image for each product, whereas the Recommender needs the current order history as training data. Once running, only the Authentication and the WebUI access, modify, and create data using the Persistence.

All services communicate via representational state transfer (REST) calls, as REST has established itself as the de-facto industry standard in the micro-service domain. In order to distribute these REST calls between the available service instances, the TeaStore uses the client-side load balancer Netflix Ribbon. The TeaStore uses a custom service registry, which is similar to Netflix Eureka that supplies service instances with target instances of a specified target service type. To enable this, all running instances register and unregister at the registry. This allows for dynamic addition and removal of service instances during run-time. Each service also sends heartbeats to the registry. In case a service is overloaded or crashes and therefore fails to send the heartbeat messages, it is removed from the list of available instances. Subsequently, it will not receive further requests from other services. This mechanism ensures good failure recovery and minimizes the number of requests sent to unavailable service instances.

Generally, all requests to the WebUI by a user or load generator are handled similarly. The WebUI always retrieves information from the Persistence service. If all information is available, images for presentation are fetched from the Image provider and embedded into the page. Finally, a Java Server Page (JSP) is compiled and returned. This behavior ensures that even non-graphical browsers and simple load generators that otherwise would not fetch images from a regular site cause image I/O in the TeaStore, ensuring comparability regardless of the load generation method. As the TeaStore is primarily a benchmarking and testing application, it is open source and available to instrumentation using available monitoring solutions. Pre-instrumented Docker images are available for each service that include the Kieker monitoring probes [1] as well as a central trace repository service- We choose Kieker, as it is a commonly used application level monitoring solution in academia. System-level monitoring for micro-service applications is usually managed by container management frameworks, such as Kubernetes.

For additional information about the TeaStore and initial case studies we refer to [2]. The TeaStore source code and detailed documentation can be found on GitHub[4], ready to deploy Docker containers are available on DockerHub[5]. A short video showcasing how to deploy and use the TeaStore in Kubernetes is available online[6].

3 CONCLUSION

We present the TeaStore, a test and reference cloud application intended to serve as a benchmarking framework for researchers evaluating their work. The TeaStore is designed to offer the degrees of freedom and performance characteristics required by cloud software management, prediction, and analysis research. Specifically, the TeaStore facilitates research focused on model-based software and performance engineering. Researchers may use the TeaStore together with its pre-packaged testing tools and profiles for their evaluation scenario.

REFERENCES

[1] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. 2012. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*. ACM, 247–248.

[2] Jóakim von Kistowski, Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, and Samuel Kounev. 2018. TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In *Proceedings of the 26th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '18)*.

[4] https://github.com/DescartesResearch/TeaStore/
[5] https://hub.docker.com/u/descartesresearch/
[6] https://www.youtube.com/watch?v=6OcSNrErzGE