Monitoring Data Distribution and Exploitation in a Global-Scale Microservice Artefact Observatory

Position Paper

Panagiotis Gkikopoulos
Zurich Univ. of Applied Sciences
Winterthur, Switzerland
pang@zhaw.ch
University of Neuchâtel
Neuchâtel, Switzerland
panagiotis.gkikopoulos@unine.ch

Josef Spillner
Zurich Univ. of Applied Sciences
Winterthur, Switzerland
josef.spillner@zhaw.ch

Valerio Schiavoni
University of Neuchâtel
Neuchâtel, Switzerland
valerio.schiavoni@unine.ch

ABSTRACT

Reusable microservice artefacts are often deployed as black or grey boxes, with little concern for their properties and quality, beyond a syntactical interface description. This leads application developers to chaotic and opportunistic assumptions about how a composite application will behave in the real world. Systematically analyzing and tracking these publicly available artefacts will grant much needed predictability to microservice-based deployments. By establishing a distributed observatory and knowledge base, it is possible to track microservice repositories and analyze the artefacts reliably, and provide insights on their properties and quality to developers and researchers alike. This position paper[1] argues for a federated research infrastructure with consensus voting among participants to establish and preserve ground truth about the insights.

1 INTRODUCTION

Software is increasingly delivered as a service through clouds and other scalable platforms. Microservices allow for a software application to be developed in a distributed way as well as increasing its resilience and scalability. They typically interact with each other using REST APIs [15], message queues or service meshes. Separate teams of developers can work on individual component services of a much larger application [29] and some of the resulting software artefacts can be reusable and thus placed in a marketplace for other developers to integrate to other projects. Beyond existing repositories specific to programming languages such as Maven Central [5], RubyGems [7] and the Python Package Index [6], microservice-specific repositories, hubs and marketplaces have shown growth over recent months. These include Docker Hub [2], Helm Hub [4] and the Amazon Serverless Application Repository [1] among others.

The widespread adoption of these marketplaces also causes concern as little quantitative data is available to developers, leading to opportunistic design decision with potentially unpredictable results.

We envision the globally operated Microservice Artefact Observatory (MAO [10]) as a scientific community effort to monitor and analyze these artefact marketplaces and provide insights through a combination of metadata monitoring, static checks and dynamic testing. For this purpose, we contribute a first working prototype of the corresponding federated research infrastructure for resilient tracking and analysis of marketplaces and artefacts. More importantly, we contribute and outlook of how researchers and developers can benefit from such an approach. A key feature of the infrastructure is a novel method of continuously generating ground truth data resulting from consensus voting over the individual observations and insights. This ground truth is usable by the software engineering and distributed systems communities for further studies.

Roadmap. The reminder of this paper is organized as follows. First, we survey related work in §2. Then, we clarify the problem statement and methodology in §3. Some preliminary results are given in §4. Finally, we describe our future work in §5.

2 RELATED WORK

Various works have delved into the monitoring and testing of quality aspects of microservice architectures. Some approaches focus on the metadata and logs generated by the microservices [14], rather than benchmarking specific architectures [19, 23] or testing dependencies and service interactions [17]. Researchers looked also at runtime monitoring of microservice-based applications [20]. More recently, there were attempts [16] for benchmarking architectural models based on microservices to study the implications of the design pattern on real world applications. SIEVE [28] extracts

---

[1]This paper extends the presentation given at EuroDW/Eurosys'20.

usable metrics for developers by monitoring these software artefacts. A similar approach was proposed in [18] for corporate managers. Finally, there are efforts to define a standard set of requirements for the orchestration of microservices [13] as well as surveys to detect trends in microservice development [30].

We observe the lack of an approach to systematic monitoring of microservice-related marketplaces. Current tools and platforms are aimed at analyzing and monitoring specific case studies and architectures to a significant depth, but the publicly available reusable artefacts are, for the most part, not scrutinized for their quality properties nor tracked with respect to potential improvements or regressions. We argue that such an approach is necessary for providing preemptive quality metrics to developers that aim to reuse publicly available artefacts, as well as researchers studying the evolution of the field and emerging trends and issues.

Prior work include a data crawler of Helm charts on the Kube Apps Hub [26], on the Serverless Application Repository [27] for QA and preservation purposes, as well as *DApps* across multiple marketplaces [22]. The current output of this effort includes static analysis software tools, as well as experimental datasets [21, 24, 25] in addition to the research papers and preprints. We intend to integrate and upstream the efforts and vision presented in this paper into these open initiatives with the goal that future metrics collections no longer rely on standalone, brittle and centralised tools.

A clear gap we have identified in the state of the art is a large scale monitoring system with the capability to track the evolution of these marketplaces and the artefacts within, to provide both broad and deep insight on the state of the ecosystem.

3 OVERVIEW

3.1 Problem Statement

Treating reusable microservice artefacts as black or grey boxes when designing a complex application can lead to wrong assumptions about their properties and thus unpredictable behavior of the application itself. Small mistakes in a microservice implementation or configuration, even in dependency third-party artefacts, might endanger the entire application as shown in Fig. 1. The quality of compositions is after all limited by their weakest point. Providing information in the form of metadata analysis, code quality, performance benchmarking and security evaluation for these artefacts can increase the predictability of their behaviour in production by allowing the developers to make informed design decisions and be aware of potential issues ahead of time. This would also help researchers in the field have access to insights on development trends and emerging anti-patterns in microservice architectures.
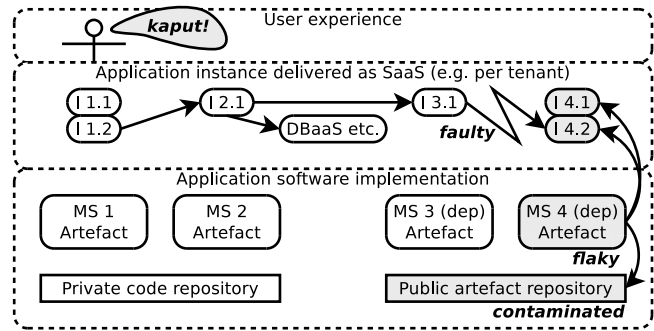

Figure 1: Propagation of weaknesses in microservice artefacts to the application-level quality of experience

Additionally, current methods used to monitor software repositories lack infrastructure to enable large scale collaborations and robust historical tracking. Yet, we argue that such infrastructure is a requirement for monitoring projects of this scale, especially since an international consortium (albeit an informal one in this case) is involved. We identify that the capability of reliable continuous tracking within a decentralized system as well as an algorithm or set of algorithms to establish ground truth when multiple data snapshots are submited by peers within the collaboration as the key features of the proposed infrastructure.

Our approach to this problem is focused around the following research questions:

- **RQ 1:** How can a distributed, federated system enable more efficient and resilient monitoring and analysis of microservice artefacts at the scale of a marketplace?
- **RQ 2:** How can cluster consensus be utilised within the federation to establish ground truth about artefact quality metrics?

3.2 Method

We envision the work to be pursued along two main dimensions: *(1)* the establishment and engineering of the proposed observatory infrastructure, and *(2)* the exploitation of said infrastructure within our primary use case of collecting artefact metrics.

The engineering component involves providing the functionality needed to assist and automate all aspects of the data management pipeline, from scheduling data acquisition tools to comparing snapshots of the data from different nodes to reach a ground truth measurement. Additionally, a main requirement is to provide resilience and reliability, to protect from hardware outages and corrupt data files.

Fig. 2 shows the current architecture for the orchestration system. It primarily consists of an orchestration/scheduling service that manages the data acquisition tools. Nodes access an etcd [3] cluster to share registry information such as
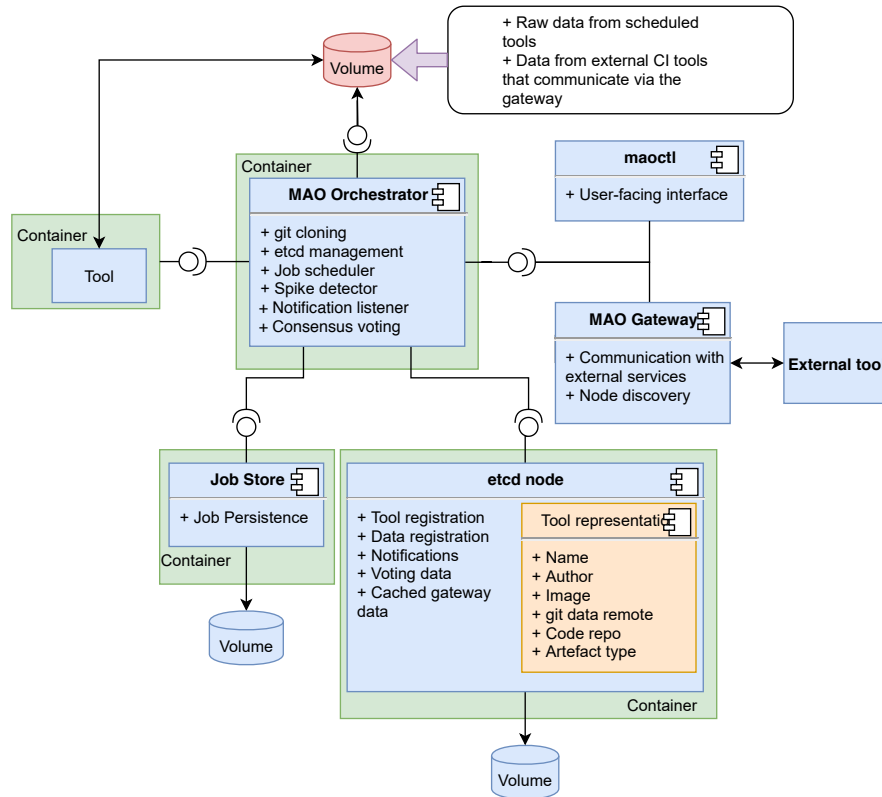
Figure 2: Current observatory architecture

available tools to deploy, dataset repositories and notifications. An additional gateway service can also be deployed to connect external tools if needed.

Apart from data acquisition automation the orchestration system aims to assist in collaborations by partially automating the replication of experiments and the verification of results. Fig. 3 shows the current concept algorithm that will be implemented in the next stages. A node that runs an experiment can announce its results to other nodes (Fig. 3-❷). If the nodes have not executed this data acquisition tool (Fig. 3-❸), they will simply accept this measurement. However, if they already have a measurement (Fig. 3-❹), they will respond to the announcement, indicating a comparable data snapshot in the `etcd` registry. The first node will then retrieve these snapshots from their respective repositories (*e.g.*, git) and run the verification algorithm (Fig. 3-❺).

The verification makes a distinction between performance metrics and quality or security metrics. In the case of benchmarking or other performance metrics (as labeled in the dataset itself) the verification will produce average, minimum and maximum values as the ground truth measurement. For metrics such as metadata or vulnerability characteristics, where average values would be meaningless, the ground truth measurement will be arrived at via clustering.

Once the verification is complete and the ground truth data snapshot is established, it will once again be announced to all nodes (see Fig. 3-❻), this time marked as verified so nodes can accept it unless a more recent measurement has surfaced.

The second dimension will focus on the exploitation of the new architecture within the MAO use case in order to both further the goals of MAO of reliable tracking of the evolution of microservice artefact repositories, as well as evaluate the improvement this federated system brings to the current state-of-the-art research practice. As the observatory grows, with more tools and metrics and more collaborators, so will out view of the system's effectiveness increase, allowing us to better gauge how it scales when applied to a large real-world scenario. For this purpose, we have initiated a free collaborative network of researchers and pilot software engineers[2].

---

[2]MAO collaboration: https://mao-mao-research.github.io/
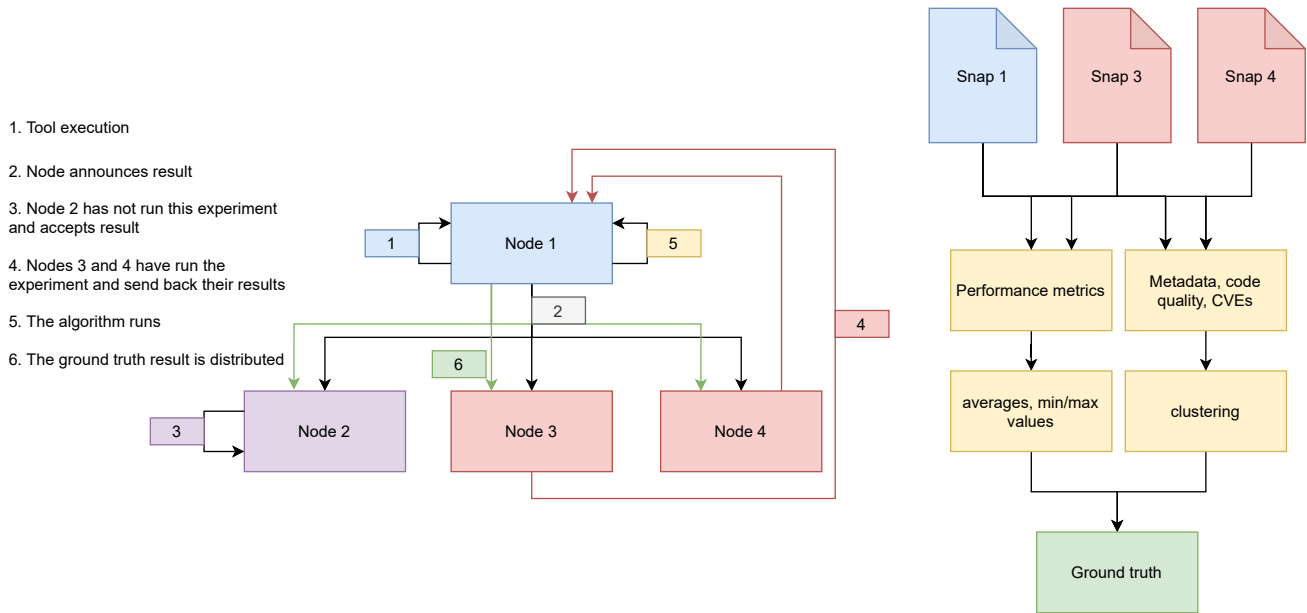
Figure 3: Consensus voting concept diagram

4 PRELIMINARY RESULTS

Early contributions in this work have been divided between improving the infrastructure of the experiments, and developing additional tooling for monitoring and testing artefacts.

We developed a distributed monitoring architecture using a geo-distributed cluster using `etcd` for metadata exchange among peers and a Docker-based scheduler-orchestrator application for members to run the monitoring tools [11]. The system is currently operated in our own servers in Switzerland, with a pilot deployment in Argentina and additional deployments being discussed with MAO-member researchers.

The first monitoring tool deployed on the system is an automatic crawler of Dockerhub's public API. It collects image metadata and provides basic insights on the development trends within the ecosystem [9]. The first version focused on OS support and CPU architecture for each image, to better understand the extent of Docker's support for heterogeneous ecosystems. Figure 4 shows the current set of data produced by the crawler. We tracked the evolution of support for different CPU architectures over time. The graph shows the number of images for ARM, x86-64 and IBM Z architectures for each date of the tracking period, along with a trend line to highlight increase/decrease over time.

A concurrent experiment focuses on extending the existing study of the AWS Serverless Application Repository with a benchmarking tool, to test the collected artefacts. To that end, an emulation based on LocalStack [8] and `sam local` [12] is currently under development. The aim is to emulate the behaviour of AWS's serverless offering to provide more accurate metrics than current unit-test tools. We also quantitatively assess Dockerfiles with multiple linters on the source level.

5 CONCLUSION AND FUTURE WORK

We proposed an architecture to analyze data from monitored artefact marketplaces. This architecture is currently being extended to support dynamic execution of the gathered artefacts, and to allow on-demand comparison of data between nodes, to establish ground truth data via cluster consensus.

The analysis will be extended towards a more diverse set of artefacts types, such as, such as Kubertetes Helm Charts, Docker images and Compose files, Kubernetes Operators and others, furthering the goals of the MAO project while simultaneously allowing us to evaluate the observatory architecture at scale.

Additionally, we aim to integrate the knowledge base we build over time into data-driven QA tooling, that can be used within CI/CD pipelines, thus providing real-time quality analysis and feedback to developers.

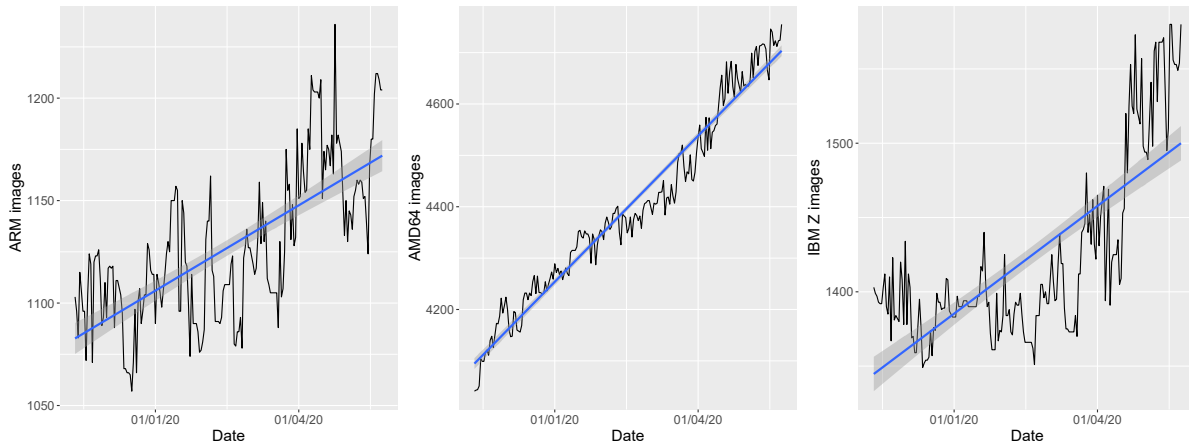**Figure 4: Dockerhub monitoring for Nov 11, 2019 to May 24, 2020**

REFERENCES

[1] AWS Serverless Application Repository. https://aws.amazon.com/serverless/serverlessrepo/. Accessed 02.06.2020.

[2] Docker Hub. https://hub.docker.com/. Accessed 02.06.2020.

[3] etcd. https://etcd.io/. Accessed 02.06.2020.

[4] Helm Hub. https://hub.helm.sh/. Accessed 02.06.2020.

[5] Maven Repository: Central. https://mvnrepository.com/repos/central. Accessed 02.06.2020.

[6] PyPi. The Python Package Index. https://pypi.org/. Accessed 02.06.2020.

[7] RubyGems. https://rubygems.org/. Accessed 02.06.2020.

[8] LocalStack. https://localstack.cloud/, 2019. Accessed: 29.02.2020.

[9] Dockerhub Collector. https://github.com/EcePanos/Dockerhub-Collector, 2020. Accessed: 29.02.2020.

[10] MAO-MAO: Microservice Artefact Observatory. https://mao-mao-research.github.io/, 2020. Accessed: 14.02.2020.

[11] MAO Orchestrator. https://github.com/serviceprototypinglab/mao-orchestrator, 2020. Accessed: 29.02.2020.

[12] Sam Local. https://github.com/thoeni/aws-sam-local, 2020. Accessed: 29.02.2020.

[13] BROGI, A., CANCIANI, A., NERI, D., RINALDI, L., AND SOLDANI, J. Towards a reference dataset of microservice-based applications. In *Software Engineering and Formal Methods* (Cham, 2018), A. Cerone and M. Roveri, Eds., Springer International Publishing, pp. 219–229.

[14] CHANG, K. S., AND FINK, S. J. Visualizing serverless cloud application logs for program understanding. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (Oct 2017), pp. 261–265.

[15] DRAGONI, N., GIALLORENZO, S., LAFUENTE, A. L., MAZZARA, M., MONTESI, F., MUSTAFIN, R., AND SAFINA, L. *Microservices: Yesterday, Today, and Tomorrow.* Springer International Publishing, Cham, 2017, pp. 195–216.

[16] GAN, Y., ZHANG, Y., CHENG, D., SHETTY, A., RATHI, P., KATARKI, N., BRUNO, A., HU, J., RITCHKEN, B., JACKSON, B., HU, K., PANCHOLI, M., HE, Y., CLANCY, B., COLEN, C., WEN, F., LEUNG, C., WANG, S., ZARUVINSKY, L., ESPINOSA, M., LIN, R., LIU, Z., PADILLA, J., AND DELIMITROU, C. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2019), ASPLOS '19, ACM, pp. 3–18.

[17] MA, S., FAN, C., CHUANG, Y., LEE, W., LEE, S., AND HSUEH, N. Using service dependency graph to analyze and test microservices. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (July 2018), vol. 02, pp. 81–86.

[18] MAYER, B., AND WEINREICH, R. A dashboard for microservice monitoring and management. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)* (April 2017), pp. 66–69.

[19] PAPAPANAGIOTOU, I., AND CHELLA, V. Ndbench: Benchmarking microservices at scale. *CoRR abs/1807.10792* (2018).

[20] PHIPATHANANUNTH, C., AND BUNYAKIATI, P. Synthetic runtime monitoring of microservices software architecture. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* (July 2018), vol. 02, pp. 448–453.

[21] QASSE, I., AND SPILLNER, J. DApps Quality Characteristics Dataset. Zenodo dataset at https://doi.org/10.5281/zenodo.3382126, August 2019.

[22] QASSE, I. A., SPILLNER, J., TALIB, M. A., AND NASIR, Q. A Study on DApps Characteristics. In *2nd IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)* (August 2020). To appear - conference shifted due to coronavirus.

[23] RAHMAN, M., AND GAO, J. A reusable automated acceptance testing architecture for microservices in behavior-driven development. In *2015 IEEE Symposium on Service-Oriented System Engineering* (March 2015), pp. 321–325.

[24] SPILLNER, J. AWS-SAR Dataset. https://github.com/serviceprototypinglab/aws-sar-dataset, 2019. Accessed 13.05.2019.

[25] SPILLNER, J. Duplicate refuction in Helm Charts. https://osf.io/5gkxq/, 2019. Accessed 13.05.2019.

[26] SPILLNER, J. Quality assessment and improvement of helm charts for kubernetes-based cloud applications. *CoRR abs/1901.00644* (2019).

[27] SPILLNER, J. Quantitative analysis of cloud function evolution in the AWS serverless application repository. *CoRR abs/1905.04800* (2019).

[28] THALHEIM, J., RODRIGUES, A., AKKUS, I. E., BHATOTIA, P., CHEN, R., VISWANATH, B., JIAO, L., AND FETZER, C. Sieve: Actionable insights from monitored metrics in microservices. *CoRR abs/1709.06686* (2017).

[29] THÖNES, J. Microservices. *IEEE Software 32*, 1 (Jan 2015), 116–116.

[30] VIGGIATO, M., TERRA, R., ROCHA, H., VALENTE, M. T., AND FIGUEIREDO, E. Microservices in Practice: A Survey Study. *CoRR abs/1808.04836* (2018).