Data Distribution and Exploitation in a Global Microservice Artefact Observatory

Panagiotis Gkikopoulos[1]

***Abstract*—Cloud computing and specifically the microservice architecture pattern is becoming an increasingly prominent paradigm in computer science. Many modern cloud applications are composed of a variety of different microservices, each potentially built in different languages, using different technologies and a different software artefact structure. What is needed is the capability to monitor this rapidly expanding field and leverage the data to enable further research and development of microservice architectures. Drawing inspiration from the global observatories used in geoscience and astronomy, the aim of this research initiative is the establishment of a global observatory for microservice artefacts, allowing the aggregation of data from different hubs and the execution of dynamic analysis on them.**

***Index Terms*—microservices, software artefacts, data science, monitoring**

I. Motivation

Microservices are becoming a prevalent model for modern software development as technology is headed more toward cloud native systems. As is usually the case with new developments, new ways of implementing and exploiting the technology are now being released at a rapid pace.

One area of particular interest are the emerging marketplaces for microservice software artefacts. In this context the term artefact entails any pre-made service or even set of services made publicly available by its developer as a package to be cloned and deployed through a marketplace.

More and more such packages are made available from multiple vendors, with leading examples being Amazon's AWS, Kubernetes and Docker. However, as with any such new trending technology, the early steps of development are usually somewhat chaotic.

Types of ready made deployment blocks like these are appearing fast and sometimes even failing before receiving traction and even for established marketplaces quality control is sometimes problematic or impossible.

There is currently little effort to systematically monitor these hubs and marketplaces and their contents. However, this endeavor is scaling up in requirements fast. Where there used to be a handful of vendors and a few hundred artefacts to check, there are now several thousand in many cases, with Docker images being a prime example, with DockerHub boasting 'over 100000 images' [1].

For this reason the MAO informal consortium was envisioned, as a scientific community effort to monitor and document these artefact marketplaces, to provide insight on them through a combination of metadata checks, code quality control and dynamic testing.

[1]P. Gkikopoulos is a research assistant and doctoral candidate at the Service Prototyping Lab (blog.zhaw.ch/splab/) of the Zurich University of Applied Sciences, Winterthur Switzerland under the supervision of Dr.-Ing. habil. Josef Spillner pang@zhaw.ch

II. Background

Cloud applications based on novel microservice architectures are drawing ever closer to becoming the norm in many sectors of modern software engineering. Microservices allow for a complex application to be developed and operated in a distributed way as well as increasing its resilience and scalability. Additionally, teams can select the languages and technologies to be used for each component service with more freedom, afforded to them by the loosened coupling between services. These services are loosely connected and typically interact with each other within their specific architecture using Representational State Transfer (REST) Application Programming Interfaces (APIs) [2], message queues or the newer service meshes. A completely separate team of developers can work independently on individual component services of a much larger application [3] and some of the resulting software artefacts can be reusable and thus placed in a hub or marketplace for other developers to adapt and integrate to other projects. Hubs, such as Docker Hub [1], Kube Apps Hub [4] and the AWS Serverless Application Repository [5], offer a growing collection of available deployment-ready packages. These repositories of microservice artefacts are still relatively young, but already contain a wealth of data both in terms of metadata and runtime characteristics. Some of this data is versioned, but some can be lost through change, causing an irreversible loss of valuable data on the evolution of microservices.

The motivation of this doctorate is exploring mining and exploitation techniques for taking advantage of this data, as well as providing a medium for data distribution and storage, that in unison would act as a catalyst for collaboration and further research. The main inspiration of this endeavor are existing global observatories in other fields. Such observatories are the primary reference point for the entire field, with access to the latest scientific developments at a glance, as well as a wealth of documents and data. Such an observatory is needed to preserve, study and further the rapidly expanding landscape of mircoservice architecture.

III. State-of-the-Art

A. Related Work

Various works have delved into the monitoring, testing or benchmarking of various aspects of microservices and the

application architectures they can be used in. Some approaches focus on the metadata and logs generated by the microservices [6], some others are built to benchmark specific architectures [7], [8] or test dependencies and service to service interactions [9] and others on runtime monitoring of microservice based applications [10]. More recently, attempts are being made for benchmarking architectural models based on microserivces to study the implications of the design pattern on real world applications. [11]. Work has been done to extract usable metrics for developers [12] and corporate managers [13] from the data obtained by monitoring these software artefacts. Effort has also been made to set a standard set of requirements for the orchestration of microservices [14]. From a slightly different perspective, some surveys have been made to detect the trends in microservice development [15].

B. MAO and Predecessors

What was missing from prior works is attempts at systematic monitoring of the microservice-related marketplaces, from the data science perspective. Preceding this research is an existing collaboration of researchers in an effort to develop and operate research infrastructure to assess microservice artefacts, named the Microservice Artefact Observatory (MAO). Researchers at the Service Prototyping Lab (SPLab) of the Zurich University of Applied Sciences (ZHAW) have made attempts to crawl through the Helm charts of the Kube Apps Hub [16] and the Serverless Application Repository [17] for QA and preservation purposes. The current output of this effort includes static analysis software tools, as well as experimental datasets [18], [19] in addition to the research paper preprint. [20]

However, such a method of data crawling can never be made truly effective, as there will be breaks in its execution and data will inevitably be lost. Consequently, what would be needed is a truly distributed and decentralized architecture, that would allow the experiment to experience no downtime and grant it fault tolerance.

Though such a task has not been undertaken before is this sub-field, some examples of similar infrastructure are already there. Distributed architecture examples can be seen in Planetlab, BOINC, Cloudlab, the EGI Federated Cloud, GENI or EOSC. The existence of Data Management Platforms and the many Mining Software Repository papers show an insight in the data mining aspect of this work, though in this case what would be needed is a continuous stream of data. Experiments in the field include industry benchmarks like YCSB as well as others like CloudSim and DockerAnalyser.

IV. Objectives

A. Problem Statement

In the field of microservice development, there currently exists no global point of reference and no formalized quality standards. Moreover, from a research perspective, data on microservice artefact evolution is continuously lost, as the technology evolves and older versions of artefacts are being replaced, often without the older versions being archived. In contrast, many other fields have established observatory-like portals, which serve both to provide insight and infrastructure for current development, and to provide a historical archive, allowing in-depth analysis of evolutionary data in the field. In an effort to address this gap in improving, studying and documenting the evolution of microservice artefacts and architectures, the following goals and research topics are posed.

B. Goals

The goals in this research are outlined as follows:

1) The creation of a historical archive for the preservation of data on microservices, for the purposes of data science.
2) The setting of a reference point for microservice development and by extension the creation of quality standards.
3) The active improvement of microservice artefacts through direct feedback to their developers.
4) Gaining an understanding of the metadata-runtime relationship, in particular how quality characteristics are affected by different architectural setups.
5) Contribution to a global observatory for microservice artefacts.

C. Research Questions

The methodology of this research endeavor is structured around the following research questions:

1) **RQ1:** How can data aggregation on microservice artefacts be efficient, relevant and extensible, to accommodate new developments?
    a) **RQ1.1:** What kinds of data should be extracted from the different types of microservice artefacts, both statically and at runtime?
    b) **RQ1.2:** How should data be formatted for easier storage and querying across different ecosystems?
    c) **RQ1.3:** How to accommodate the addition of new sources in the form of new microservice ecosystems and new or improved data crawlers?
2) **RQ2:** What developer engagement can be observed by providing feedback generated through the analysis of their artefacts?
    a) **RQ2.1:** What kind of response is received by the feedback generated with the developers contacted?
    b) **RQ2.2:** Is the observatory serving one of its primary goals of presenting a quality standard and through that, improving the average quality of microservice artefacts over time?
3) **RQ3:** What insight can be obtained through the analysis of the aggregated data?
    a) **RQ3.1:** What can be learned about the current state of microservice development?
    b) **RQ3.2:** What can be learned from a historical archive that follows the evolution of microservice artefacts?

c) **RQ3.3** Can knowledge about runtime characteristics, such as security or performance, be inferred from static (meta)datasets alone?

V. Methods

The realization of this doctoral work involves both work on the establishment of the proposed observatory, as well as in-depth studying of available metrics and comparison data, in the effort of extracting as much usable data as possible, to create the most useful framework possible, both from an archival and from a data analysis perspective.

A. Computer Science Component

The extension of current metadata analysis tools to collect as much data as possible to monitor the evolution of microservice artefacts the primary concern of this research. This entails the collection and aggregation of multiple artefact types to be processed and analyzed.

One very important part of the data collection aspect that is really valuable is dynamic execution testing of the artefacts collected. This will provide a significant amount of data and information on microservice architectures that simply examining their metadata will not.

B. Data Science Component

The analysis of the acquired data, to obtain views on methods and technologies employed in application development using microservices and understand the trends and methodologies of modern cloud-first development with quantifiable, comparable data. The subjects of this study are components, applications, services, repositories, code and artefacts. The extraction of useful and reliable information on microservices for the purpose of both creating a historical archive of their evolution and assisting in furthering the technology as a whole, is the focus of the data analysis component of this proposal.

VI. Current Status

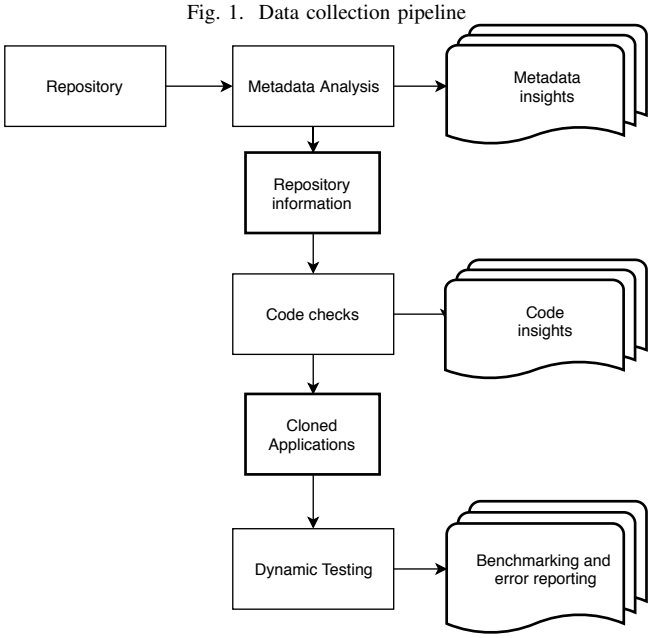
The approach to answering the research questions is 'preservation first, infrastructure later', meaning even before a global research infrastructure can be established, data collection and tests already need to be performed to avoid the loss of critical data on microservices.

A. Short Term Goal

The first stage of this research project was to extend the capabilities of an already existing tool in the MAO informal consortium, SAR Analysis. The SAR analysis tools aim to monitor the AWS Serverless Application Respository and then collect the software artefacts themselves to analyze the code. The applications contained in the SAR are based on Amazon's Serverless Application Model (SAM) and combine Function as a Service elements as well as Backend as a Service.

While the data collected by this tool is important, it lacked the capability to run and test the applications to collect dynamic runtime data. For this purpose, an extension is being made, that performs dynamic testing of all artefacts collected by the existing tool.

Fig. 1. Data collection pipeline



B. Tool Implementation

The new testing tool aims to be able to run and test as many applications of the Serverless Application Repository as possible, and collect as much useful data as possible in the process. The static code checking tools collect all repositories in the SAR, and clone them in an easily machine-traversable manner for the dynamic tool to access them. The dynamic execution implementation is based on Amazon's Serverless Application Model (SAM) local testing suite. An SAM application is defined in a template manifest, containing Lambda functions, event triggers and other backend resources. The tools of the SAM Local suite provide the capability to fire dummy events that follow the data structure of events produced by real AWS services, to trigger the functions in the application and test them, providing basic benchmarking data.

What our new tool is doing, is attempting to test these applications en masse, by parsing the template files to determine which test events to fire, and then invoking the serverless applications themselves in the SAM Local environment.

VII. Preliminary Results

The first version of the SAM testing script was ran against the code repository dataset from the SAR analysis tools. From 447 applications, 514 function invocations were made, out of which 115 (22.37%) succeeded. The identified reasons for an invocation failing can be seen in Table A.

As evident by the results, automating the process of testing hundreds of applications is a non-trivial task. Some of the failures can be addressed, for example code being compressed or simulating backend components using a a solution such as Localstack [21].

The most important next step beyond improving the success rate of function invocations, is obtaining key metrics

TABLE I
SAM MASS-TESTING FIRST RESULTS

Success	115	22.4%
Unsupported code format	96	18.7%
Missing backend component	65	12.6%
Missing template	4	0.8%
Invalid template	3	0.6%
Other failures	231	44.9%

during execution. The SAM test scripts already output some information like execution time and memory usage, but this information needs to be parsed and analyzed to differentiate a successful run (and thus usable data) from unsuccessful runs.

A. Next steps

As this doctoral work is part of a larger research effort, it is important to report on the state of the current works. Prior effort has as mentioned focused on static checks. This is currently growing and effort is being made to extend the checks to as many properties as possible, many in the form of code checks and metadata validation.

Currently a significant amount of work is being done to extend the checks to more marketplaces and more types of artefacts, including Kubernetes Operators [22], and Cloud Native Application Bundles [23]. The first steps attempting to collect static data from metadata and configuration files are already being taken, to be followed by code checks and dynamic runtime testing and benchmarking.

VIII. FURTHER STUDY PLAN

The effort so far has been centered on analyzing as much data as possible from the monitored artefact marketplaces. Existing research into the metadata and code analysis of microservice artefacts is now being expanded with dynamic execution to obtain real test data. At this stage the testing capabilities are quite limited and current focus is on maximizing the success rate of the automated testing, which has proved to be a non-trivial task. Further iterations along this path will also focus on the generation of key metrics, to accurately report on the status and quality of the artefacts under examination.

Beyond the current effort on testing SAM artefacts, this doctoral study will be expanded to the other types of artefacts that are being monitored by the MAO informal consortium, such as Kubertetes Helm Charts, Docker images and Compose files, Operators and others.

We believe that preservation and study of these datasets will provide key insights on the development of the microservice architectures and the trends in their development and has the potential to exact positive influence on the development of microservice software through developer engagement and feedback.

REFERENCES

[1] Docker Hub. https://hub.docker.com/. Accessed: 14.02.2019.

[2] Alberto Lluch Lafuente Manuel Mazzara Fabrizio Montesi Ruslan Mustafin Larisa Safina Nicola Dragoni, Saverio Giallorenzo. Microservices: yesterday, today, tomorrow. *arXiv:1606.04036v4*.

[3] Johannes Thones. Microservices. *IEEE Software*, 32(1):116, 2015.

[4] Kube Apps Hub. https://hub.kubeapps.com/. Accessed: 14.02.2019.

[5] AWS Serverless Application Repository. https://aws.amazon.com/serverless/serverlessrepo/. Accessed: 14.02.2019.

[6] Stephen J. Fink Kerry Shih-Ping Chang. Visualizing Serverless Cloud Application Logs for Program Understanding. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*.

[7] Ioannis Papapanagiotou and Vinay Chella. NDBench: Benchmarking Microservices at Scale. *arXiv:1807:10792v1*.

[8] Mazedur Rahman and Jerry Gao. A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development. In *2015 IEEE Symposium on Service-Oriented System Engineering, SOSE 2015, San Francisco Bay, CA, USA, March 30 - April 3, 2015*, pages 321–325, 2015.

[9] Yen Chuang Wen-Tin Lee Shin-Jie Lee Nien-Lin Hsueh Shang-Pin Ma, Chen-Yuan Fan. Using Service Dependency Graph to Analyze and Test Microservices. *2018 42nd IEEE International Conference on Computer Software and Applications*.

[10] Chadarat Phipathananunth and Panuchart Bunyakiati. Synthetic Runtime Monitoring of Microservices Software Architecture. In *2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23-27 July 2018, Volume 2*, pages 448–453, 2018.

[11] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 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*, ASPLOS '19, pages 3–18, New York, NY, USA, 2019. ACM.

[12] Alberto Lluch Lafuente Manuel Mazzara-Fabrizio Montesi Ruslan Mustafin Larisa Safina Nicola Dragoni, Saverio Giallorenzo. Sieve: Actionable Insights from Monitored Metrics in Microservices. *arXiv:1709:0667v1*.

[13] Benjamin Mayer and Rainer Weinreich. A Dashboard for Microservice Monitoring and Management. *2017 IEEE International Conference on Software Architecture Workshops*.

[14] Antonio Brogi, Andrea Canciani, Davide Neri, Luca Rinaldi, and Jacopo Soldani. Towards a Reference Dataset of Microservice-Based Applications. In *Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, September 4-5, 2017, Revised Selected Papers*, pages 219–229, 2017.

[15] Markos Viggiato, Ricardo Terra, Henrique Rocha, Marco Tulio Valente, and Eduardo Figueiredo. Microservices in Practice: A Survey Study. *CoRR*, abs/1808.04836, 2018.

[16] Josef Spillner. Quality Assessment and Improvement of Helm Charts for Kubernetes-Based Cloud Applications. *arXiv:1901.00644v1*.

[17] Josef Spillner. Quantitative Analysis of Cloud Function Evolution in the AWS Serverless Application Repository. *arXiv:1905.04800*.

[18] Josef Spillner. AWS-SAR Dataset. https://github.com/serviceprototypinglab/aws-sar-dataset. Accessed 13.05.2019.

[19] Josef Spillner. Duplicate refuction in Helm Charts. https://osf.io/5gkxq/. Accessed 13.05.2019.

[20] MAO-MAO: Microservice Artefact Observatory. https://mao-mao-research.github.io/. Accessed: 14.02.2019.

[21] Localstack. https://github.com/localstack/localstack. Accessed 13.05.2019.

[22] OperatorHub. https://operatorhub.io/. Accessed 13.05.2019.

[23] CNAB: Cloud Native Application Bundles. https://cnab.io/. Accessed 13.05.2019.