Workload Characterization of a Software-as-a-Service Web Application Implemented with a Microservices Architecture

Harold Aragon
Escuela Superior Politecnica del Litoral, ESPOL. Guayaquil, Ecuador.
haragon@espol.edu.ec

Samuel Braganza
Escuela Superior Politecnica del Litoral, ESPOL. Guayaquil, Ecuador.
sbraganz@espol.edu.ec

Edwin F. Boza
Escuela Superior Politecnica del Litoral, ESPOL. Guayaquil, Ecuador.
eboza@espol.edu.ec

Jonathan Parrales
Escuela Superior Politecnica del Litoral, ESPOL. Guayaquil, Ecuador.
jfparral@espol.edu.ec

Cristina L. Abad
Escuela Superior Politecnica del Litoral, ESPOL. Guayaquil, Ecuador.
cabadr@espol.edu.ec

ABSTRACT

We study the workload of an Online Invoicing application with clients in the Andean region in South America. The application is offered to clients with a Software-as-a-Service model, has a microservices architecture and runs on a containerized environment on a public cloud provider. The cloud application workload described in this paper can be used as part of a workload suite comprised of different application workloads, when evaluating microservices architectures. To the best of our knowledge, this is a novel workload in the web domain and it complements other workloads publicly available. Though we make no claim of the general applicability of this workload as a "microservices benchmark", its inclusion in evaluations could aid researchers and practitioners enrich their evaluations with tests based on a real microservices-based web application. Finally, we provide some insights regarding best-practices in microservice design, as a result of the observed workload.

CCS CONCEPTS

• **General and reference** → **Measurement**; • **Software and its engineering** → *Cloud computing*.

KEYWORDS

microservices; workload characterization; software-as-a-service

ACM Reference Format:
Harold Aragon, Samuel Braganza, Edwin F. Boza, Jonathan Parrales, and Cristina L. Abad. 2019. Workload Characterization of a Software-as-a-Service Web Application Implemented with a Microservices Architecture. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3308560.3316466

1 INTRODUCTION

In a microservice architecture, an application is formed by a series of loosely coupled services, each performing a single functionality. These services are fine grained and communicate using lightweight protocols. Each microservice should be elastic and resilient, and have clearly defined APIs for ease of composability [7, 12, 17].

Microservices architectures have been increasingly adopted by enterprises [4, 20], mostly due to: (1) the success of the high-performance microservice-based applications of giants like Netflix, Amazon and eBay, and (2) the benefits of these architectures, which include enabling services that are independently developed and deployable, ease of horizontal scalability in Infrastructure-as-a-Service (IaaS) clouds, and support for efficient development team structures [19].

In the quest to increase the adoption of microservices architectures, many enhancements to the cloud infrastructure are being proposed. These modifications and enhancements seek to improve the performance, scalability and fault tolerance of microservice architectures under dynamic workloads (e.g., see [14, 16, 22]).

To better support these proposed enhancements, we need a better understanding of the types of workloads that microservice architectures impose on the computing infrastructure. Sadly, there is no freely available repository with microservice workloads.

We study and characterize the workload of a web application implemented with a microservices architecture. The workload comes from an invoicing-as-a-service application [8]. The results of our workload characterization can help others better understand how real microservice architectures work. Furthermore, our results can be used to parameterize simulations and benchmarks used to evaluate proposed infrastructure performance enhancements. To the best of our knowledge, this is a novel workload in the web domain and it complements other workloads publicly available. In addition, we provide insights supporting existing best-practices in microservice design, as a result of the observed workload.

2 DATASET

We studied a set of traces provided by Dátil (https://datil.co), a company with clients in the Andean region of South America. Specifically, they provided us with traces recording all requests received by the microservices supporting their Invoicing-as-a-Service product, during February, March and the first 18 days of April (2017). Dátil is a Software-as-a-Service provider that enables businesses to perform their invoicing online, by connecting to Dátil's web application. The businesses are charged a monthly subscription, depending on pre-defined service tiers (e.g., based on number of transactions per month). This web application is implemented using a microservices

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316466

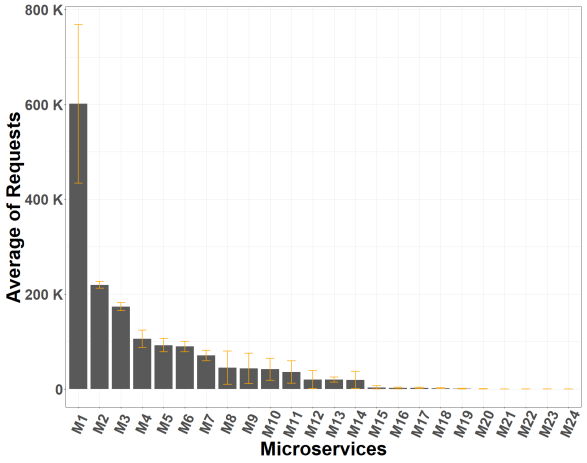


Figure 1: Average per-hour requests for each microservice, for the period between February and March 2017. The associated error bars correspond to one standard deviation.

architecture, and runs on a containerized environment (Docker) on a public cloud provider (Amazon Web Services).

The common log format of the traces included information about HTTP requests. For each request, the logs contain the request time, origin IP, destination IP and target microservice.

A preliminary analysis of this dataset was used to motivate prior work on simulation-based budgeting for web applications running on public cloud providers [8]. However, this analysis was limited to one microservice. In this paper, we extend this analysis by studying all the microservices supporting one web application.

An important limitation of our dataset is that the traces provided by Dátil do not contain information about dependencies between the microservices. For this reason, we do not study these dependencies or the workflows that they represent. Nevertheless, the information that can be analyzed without the dependency information is of use to experiments that seek to test aspects like load balancing, scalability, fault-tolerance, and performance.

Data processing. The data was preprocessed in Linux using standard command-line tools and converted to CSV (comma-separated values) files. The analysis and figure generation was done in R.

3 WORKLOAD CHARACTERIZATION

Unless otherwise noted, the results presented in this section correspond to the analysis of the complete dataset (Feb 01, 2017 through April 18, 2017). We focus our efforts to obtain statistical information that can be used by others to parameterize their benchmarks and simulators. We study common workload features like popularity, interarrival rates, and service times.

Microservice popularity. A common way to generate requests to target different "entities" (files, blocks, services, etc.) in a system, is to sample from the popularity distribution of these objects. This is referred to as the *Independence Reference Model (IRM)* [3], and it is frequently used for simulations [3] and benchmarking [13].

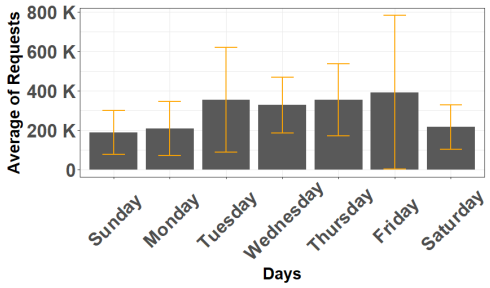


Figure 2: Average requests to M1 microservice per day of week. The error bars correspond to one standard deviation.

Figure 1 shows average per-hour requests for each microservice, for the period between February and March of 2017. The distribution is skewed, with a single microservice almost 3 times more popular than the second one, and a tail of unpopular microservices; however, the data is not well represented with a Zipf distribution (the gold standard when modeling popularity).

The data shows little variability in the hourly rate, except for the most popular microservice (M1). This behaviour should be considered for resource pre-allocation; if the service is elastically provisioned, tail latency due to cold starts triggered by scale-up decisions will increase [18]. The less popular microservice was only requested 23 times during the analyzed period.

Insight 1: Some microservices are extremely unpopular. Companies could consider migrating them to a serverless platform [1], e.g. AWS Lambda, instead of using a traditional container-based implementation; this approach could lead to significant savings without a noticeable decrease in performance [8].

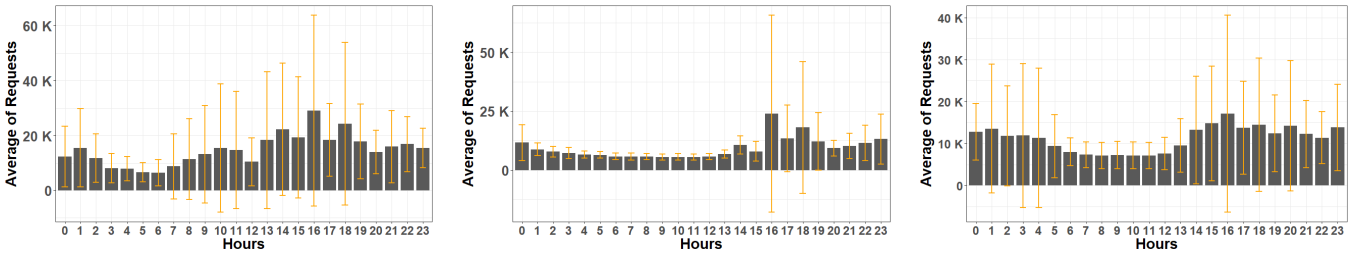
Request arrival rates. Figure 2 shows the average requests to the most popular microservice, received during each day of the week. The highest load is sustained Tuesday through Friday. This is the result of the higher commercial activity during these days.

Figures 3a– 3c show the average hourly load for the most popular service (M1), at day, week, and month levels. Load peaks at the 4-5pm slot, possibly due to some people leaving work at 4:30pm and increasing commercial transactions around this time. Least loaded hours are between 3am and 6am, when most businesses, other than pharmacies and gas stations, are closed. We can observe that the data from the third week of February is not representative of the hourly distribution observed at a monthly scale.

Insight 2: The high variability in request intensity at the daily and weekly levels, combined with the unpredictability of some of the variability, calls for the use of responsive auto-scalers that minimize costs while maintaining a target performance level.

Burstiness in arrivals. Figure 4 shows the burstiness of the requests received by M1 at different time scales: requests per second, per minute, per 10 minutes, and per hour. The load is very bursty, with peaks of high activity followed by plateaus with low load.

Insight 3: Requests are bursty at several levels, suggesting self-similar behavior; this has implications related to queuing theory at



(a) Per-hour requests in Tuesday days. **(b) Per-hour requests in 3rd week of February.** **(c) Per-hour requests in February.**

Figure 3: Average per-hour requests to the most popular microservice (M1). Error bars correspond to one standard deviation.

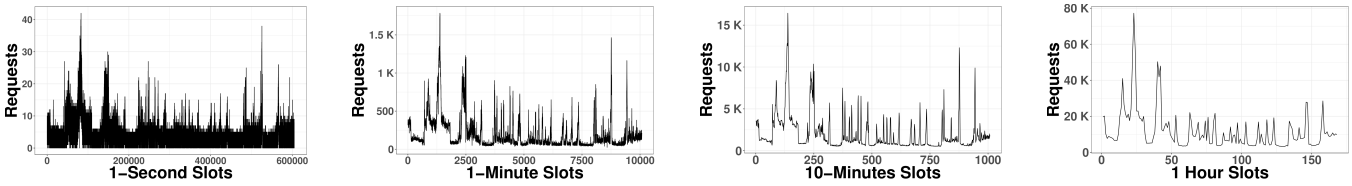


Figure 4: Requests in slots of different size in the first week of March for M1 microservice.

any place where requests may be queued (e.g., lost requests due to overflowing buffers).

Interarrival times. Figure 5 shows the cumulative distribution function (CDF) of the interarrival times for requests received by microservice M1. The tail, i.e. interarrivals greater than 10 seconds, was omitted to be able to show the most significant data in detail.

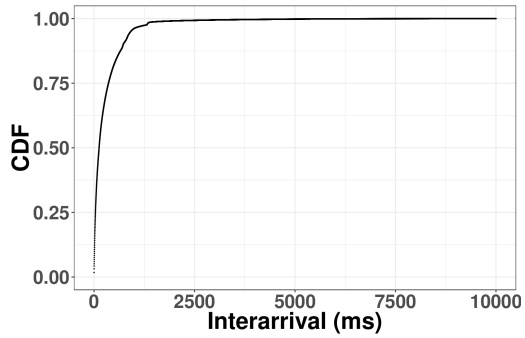


Figure 5: Interarrival distribution for M1 microservice.

Service Time. Figure 6 shows the backend processing time, which the load balancer defines as "The total time elapsed, in seconds, from the time the load balancer sent the request to a registered instance until the instance started to send the response headers." [5]. In other words, this records the time that it takes a microservice to process a request (i.e., the *service time*, using queuing theory terminology). We can observe that most microservices have a service time of less than 0.5 seconds. Furthermore, only two microservices, M8 and M9, have a service time that exceeds 3 seconds (on occasion); the largest observed service time was 60 seconds.

Insight 4: The combination of microservices with low and high response times in a single application calls for the use of diverse communication models (e.g., distributed message queues, pub-sub, and lightweight RPCs) that best suits each use case.

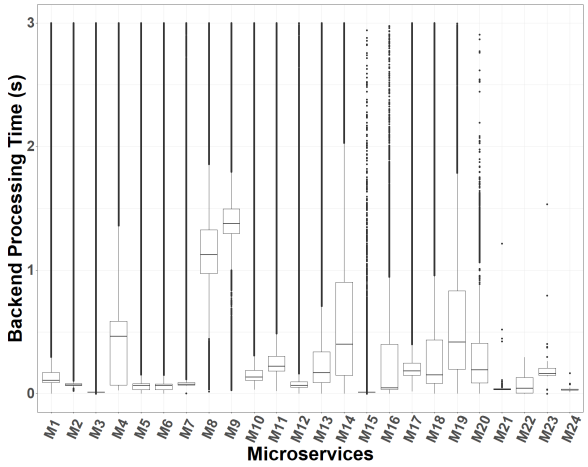
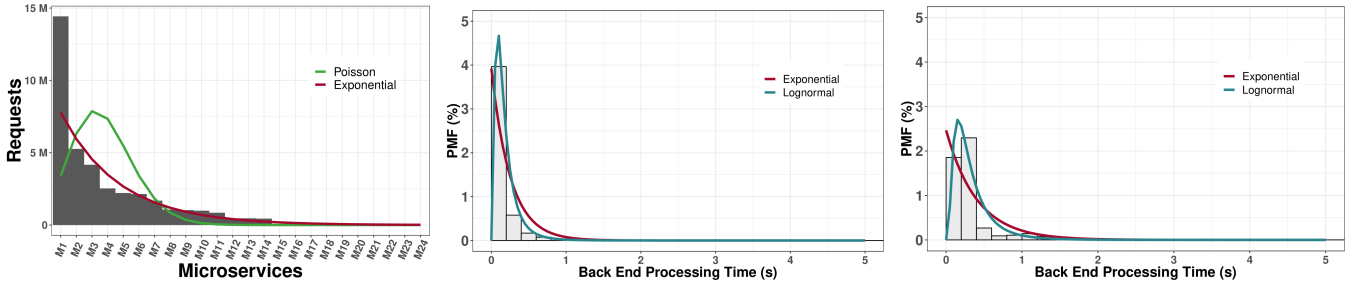


Figure 6: Boxplots of the microservice's service times.

Distribution Fitting. To facilitate the tasks of generating data for simulations and benchmarks, we provide the best fit of some of the studied data to common statistical distributions.

The graphical results are shown in Figure 7, while the Tables 1 and 2 quantify the difference between the observed data and the best fits, using the Kolmogorov-Smirnov distance. Their corresponding lognormal and exponential fitting parameters are $meanlog = -1.32$, $sdlog = 0.72$, and $rate = 2.46$ for M1; and $meanlog = -1.93$, $sdlog = 0.76$, and $rate = 3.92$ for M11. For microservice popularity, Poisson and exponential fitting parameters are $lambda = 0.73$ and $rate =$



(a) Best fit for microservices popularity (b) Best fit for service times of microservice M1 (c) Best fit for service times of microservice M11

Figure 7: Best fit for microservices popularity, and service times.

Table 1: Kolmogorov-Smirnov statistic for the best fit of the distribution of microservices popularity, to the exponential and poisson distributions.

	exponential	poisson
Kolmogorov-Smirnov statistic (D)	0.23493	0.21107

Table 2: Kolmogorov-Smirnov statistic for the best fit of the Service Times of microservices M1 and M11, to the exponential and lognormal distributions

	exponential	lognormal
M1	0.26217	0.20965
M11	0.27874	0.19094

0.26. We should note that the Zipf distribution was a very poor fit for our popularity data. The exponential distribution was not a good fit for the service times, but we nevertheless provide the best fit to this distribution, as it is frequently used in performance (queuing theory) models. Finally, due to space constraints we do not show the best fit of the interarrival times to the exponential distribution: $\lambda = 0.003573037$, with $D = 0.15421$ (Kolmogorov-Smirnov goodness of fit). For all presented data, we tried fitting to other common distributions, but do not present the results due to obtaining very poor results.

4 RELATED WORK

Workload characterization studies are important as they can be used as the basis for performance modeling, simulations and benchmarking [15]. In the past, others have studied in detail workloads from several related domains, like web servers [6], media servers [11], storage [2], MapReduce clusters [10], among others. In the microservices domain, a few application-based synthetic benchmarks exist, most notably, Acme Air [21], *μSET* [9] and Sock Shop[1]. Our work can be used to parameterize benchmarks like these, and also as the basis of simulation-based studies. However, a good evaluation should not be based on one single workload, so the need for public information and traces regarding real microservices workloads remains an open problem that hinders the evaluation of microservice architectures as well as emerging improvements to the infrastructure that supports them.

[1]https://microservices-demo.github.io/

5 CONCLUSIONS

We presented the results of a workload characterization study of a small Online Invoicing application implemented using a microservices architecture. The cloud application workload described in this paper can be used as part of a workload suite comprised of different application workloads to evaluate microservice architectures, or as the basis of simulation-based performance studies. We urge others to publish similar studies of real applications with microservices architectures, so that the community can evaluate their systems using a wide variety of representative workloads.

REFERENCES

[1] Cristina Abad, Edwin Boza, and Erwin van Eyk. 2018. Package-Aware Scheduling of FaaS Functions. In *Companion of ACM/SPEC ICPE*.

[2] Cristina Abad, Nathan Roberts, Yi Lu, and Roy Campbell. 2012. A storage-centric analysis of MapReduce workloads: File popularity, temporal locality and arrival patterns. In *IEEE IISWC*.

[3] Cristina Abad, Mindi Yuan, Chris Cai, Yi Lu, Nathan Roberts, and Roy Campbell. 2013. Generating request streams on Big Data using clustered renewal processes. *Performance Evaluation* 70, 10 (2013).

[4] Carlos Aderaldo, Nabor Mendonça, C Pahl, and P Jamshidi. 2017. Benchmark requirements for microservices architecture research. In *IEEE/ACM ECASE*.

[5] Inc. Amazon Web Services. 2018. Access Logs for Your Classic Load Balancer. In *AWS Documentation*. Amazon Web Services, 96.

[6] Martin F Arlitt and Carey L Williamson. 1996. Web server workload characterization: The search for invariants. *SIGMETRICS Perf. Eval. Rev.* 24, 1 (1996).

[7] SAMAN Barakat. 2017. Monitoring and Analysis of Microservices Performance. *Journal of Computer Science & Control Systems* 10, 1 (2017), 19.

[8] Edwin Boza, Cristina Abad, Mónica Villavicencio, Stephany Quimba, and Juan Plaza. 2017. Reserved, on demand or serverless: Model-based simulations for cloud budget planning. In *IEEE Ecuador Technical Chapters Meeting (ETCM)*.

[9] Antonio Brogi, Andrea Canciani, Davide Neri, Luca Rinaldi, and Jacopo Soldani. 2017. Towards a reference dataset of microservice-based applications. In *International Conference on Software Engineering and Formal Methods*. Springer.

[10] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. 2011. The Case for Evaluating MapReduce Performance Using Workload Suites. In *IEEE MASCOTS*.

[11] L. Cherkasova and Minaxi Gupta. 2004. Analysis of enterprise media server workloads: Access patterns, locality, content evolution, and rates of change. *IEEE/ACM Transactions on Networking* 12, 5 (2004).

[12] Ghyzlane Cherradi, Adil El Bouziri, Azedine Boulmakoul, and Karine Zeitouni. 2017. Real-time hazmat environmental information system: A micro-service based architecture. *Procedia Computer Science* 109 (2017).

[13] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *ACM SoCC*.

[14] Maria Fazio, A Celesti, R Ranjan, C Liu, L Chen, and M Villari. 2016. Open issues in scheduling microservices in the cloud. *IEEE Cloud Comp.* 3, 5 (2016).

[15] Dror G Feitelson. 2002. Workload modeling for performance evaluation. In *IFIP Intl. Symp. Computer Performance Modeling, Measurement and Evaluation*.
[16] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. 2018. Pocket: Elastic ephemeral storage for serverless analytics. In *USENIX Symp. Op. Sys. Design and Impl. (OSDI)*.
[17] Holger Knoche. 2016. Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices. In *ACM/SPEC ICPE*.
[18] Wes Lloyd, Shruti Ramesh, Swetha Chinthalapati, Lan Ly, and Shrideep Pallickara. 2018. Serverless computing: An investigation of factors influencing microservice performance. In *IEEE IC2E*.
[19] Genc Mazlami, Jurgen Cito, and Philipp Leitner. 2017. Extraction of microservices from monolithic software architectures. In *IEEE Intl. Conf. Web Services (ICWS)*.
[20] Takanori Ueda, Takuya Nakaike, and Moriyoshi Ohara. 2016. Workload characterization for microservices. In *IEEE IISWC*.
[21] T. Ueda, T. Nakaike, and M. Ohara. 2016. Workload characterization for microservices. In *IEEE IISWC*.
[22] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *European Conf. Comp. Sys. (EuroSys)*.