The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms

Trever Schirmer
TU Berlin & ECDF
Berlin, Germany
ts@mcc.tu-berlin.de

Nils Japke
TU Berlin & ECDF
Berlin, Germany
nj@mcc.tu-berlin.de

Sofia Greten
TU Berlin & ECDF
Berlin, Germany
sog@mcc.tu-berlin.de

Tobias Pfandzelter
TU Berlin & ECDF
Berlin, Germany
tp@mcc.tu-berlin.de

David Bermbach
TU Berlin & ECDF
Berlin, Germany
db@mcc.tu-berlin.de

ABSTRACT

Function-as-a-Service is a popular cloud programming model that supports developers by abstracting away most operational concerns with automatic deployment and scaling of applications. Due to the high level of abstraction, developers rely on the cloud platform to offer a consistent service level, as decreased performance leads to higher latency *and* higher cost given the pay-per-use model. In this paper, we measure performance variability of Google Cloud Functions over multiple months. Our results show that diurnal patterns can lead to performance differences of up to 15%, and that the frequency of unexpected cold starts increases threefold during the start of the week. This behavior can negatively impact researchers that conduct performance studies on cloud platforms and practitioners that run cloud applications.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Applied computing** → Service-oriented architectures; • **Software and its engineering** → *Software performance*; • **Networks** → *Cloud computing*.

KEYWORDS

Serverless Computing, Function as a Service, Performance Variation, Resource Contention

ACM Reference Format:
Trever Schirmer, Nils Japke, Sofia Greten, Tobias Pfandzelter, and David Bermbach. 2023. The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms. In *The 1st Workshop on SErverless Systems, Applications and MEthodologies (SESAME '23), May 8, 2023, Rome, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3592533.3592808

1 INTRODUCTION

Function-as-a-Service (FaaS) is a serverless cloud computing delivery model where developers compose their applications from event-driven stateless functions and all operational tasks are managed by the cloud provider [5, 17, 23]. Functions are billed on a pay-per-use basis at second or even microsecond granularity and offer rapid elasticity and scale [5, 11, 23]. The abstraction from operational concerns has made FaaS a popular cloud execution model, with offerings by all major cloud providers, e.g., Amazon Web Services Lambda[1] and Google Cloud Functions[2] [2, 5, 11].

The flip side of high levels of resource sharing in the cloud and abstracting from resource management is that developers must rely on the cloud platform provider to offer stable and consistent performance. Somewhat counterintuitively, FaaS users must actually pay *more* when a FaaS platform underperforms and latency is higher, as billed function execution is also longer [2, 36]. Most cloud FaaS platforms do not offer service level agreements beyond limited guarantees regarding general uptime [1, 21, 44].

Cloud computing is subject to performance variations [7, 33], and FaaS is no exception, as previous studies on the long-term (day-to-day) performance changes of FaaS platforms have shown [16, 27, 40]. A general improvement of FaaS services over time is expected as platform providers update and advance there infrastructure [42], yet there are also much finer effects in the short term.

In this paper, we benchmark and analyze these effects using highly frequent (every 40s) cloud FaaS benchmarks over the course of two months against Google Cloud Functions (GCF), which has received fewer attention in previous studies. Our main finding is that performance varies greatly during the course of the day, with an increase of request-response latency of up to 15% and more than three times as many unexpected cold starts from day to night *within the same day*, and that these effects are most noticeable at the start of the week. These findings impact how we interpret the results of cloud FaaS benchmarks and are significant for cloud FaaS developers. In summary, we make the following contributions:

- Based on a number of existing serverless benchmarks, we propose a methodology for evaluating temporal performance variations in cloud FaaS platforms (§3).

[1] https://aws.amazon.com/lambda/
[2] https://cloud.google.com/functions/

SESAME '23, May 8, 2023, Rome, Italy

ACM ISBN 979-8-4007-0185-6/23/05…$15.00
https://doi.org/10.1145/3592533.3592808

- We execute our benchmark with frequent (every 40s) runs on Google Cloud Functions over the course of two months and show how request-response latency exhibits strong (up to 15% difference) diurnal variation (§4.1).
- We further uncover an increase in unexpected cold starts at specific times of the day that suggest increased rates of instance recycling in GCF during times of high demand (§4.2).
- We evaluate long-term trends in our data and identify possible causes (§4.3).
- We survey existing cloud FaaS benchmarking studies and find that almost two thirds provide insufficient information on how the performance variability effects we identify are controlled for (§5.1).
- We discuss implications of our findings for practitioners that run applications on cloud FaaS platforms (§5.2).

In order to enable other researchers and practitioners to extend and replicate our experiments, we make the artifacts used to produce this paper available as open-source[3].

2 RELATED WORK

Existing research on FaaS performance variability focuses on *either* short-term *or* long-term (i.e., more than a week) variability. Short-term studies include a report by Lambion et al. [27], who have benchmarked the performance of various functions on AWS Lambda over the course of a day. The authors execute their functions in different time zones and using different hardware architectures, and find a 6% shorter function duration during the night. Mahmoudi et al. [29] propose an analytical performance model to predict performance metrics of functions. To validate their model, they repeat the same one-hour experiment ten times, and find that request arrival rate, average response times, and function timeout can be used to predict performance up to five minutes in advance. Ginzburg and Freedman [20] analyze performance variations on AWS Lambda over the course a week. They call 1000 functions every two hours and show that the daily performance of the same function inside the same region and between regions can vary significantly, which is mainly caused by local inactivity and lack of performance isolation between tenants. Since they only measure for one week, their analysis focuses on daily variations, which they measure at 1-2%.

A long-term study of serverless systems is presented by Eismann et al. [16], who have executed the same serverless application on Lambda once a day over ten months. They find long-term performance changes that are likely to be caused by platform changes, and short-term variations between days. Figelia et al. [19] measure the performance of various functions running on Lambda over seven months, but do not analyze their dataset for regular variability.

The focus of our paper is to close the gap between long-term and short-term studies by collecting frequent measurements over a longer period of time. Additionally, we go beyond the focus on AWS Lambda and present a general methodology for experiments that are applicable to all FaaS platforms. We collect our results on Google Cloud Functions, which has received fewer attention in previous studies despite its popularity.

3 METHODOLOGY

To assess performance variations in FaaS, we repeatedly execute a FaaS function in short intervals across a large time span. By controlling for execution region, resource parameters (through the memory option on cloud FaaS providers), and function type, we can focus on cloud platform performance.

Functions. We use three FaaS functions from existing serverless benchmarks in our experiments. All functions perform isolated computations that do not rely on external services, the performance of which could influence our results [22]. The *float* workload of Kim et al. [26] that performs floating point operations. The *matrix* function of Werner et al. [41] that performs matrix multiplication. Finally, we adapt the face detection model of Barosum et al. [4] for the *ml* function. To minimize the impact of external fluctuations on our measurements, we embed all inputs directly into the functions.

The resources available to a function instance are determined by the memory configured for that function: On GCF, the amount of vCPUs allocated to a function instance is tiered and increases with every multiple of 128MB memory, while AWS Lambda scales vCPUs linearly with memory [14]. To capture effects of resource configurations, we deploy *float* and *matrix* with 128MB, 256MB, and 512MB of memory, while *ml* is deployed with 512MB and 1024MB as it has higher resource requirements.

Execution. Cloud function invocations can be both "warm" and "cold": When a function is invoked for the first time, a new function instance is created. This is called a "cold start" and incurs a creation overhead [3, 6, 31]. Subsequent (but possibly not parallel) invocations of the same function can reuse the existing instance and avoid this overhead, the "warm starts". Typically, cloud platforms will keep existing instances for future invocations for a limited amount of time and then evict them to reclaim resources [13].

To capture both cold and warm start latencies, we invoke functions in loops: We first call the function once, creating a cold start, and then call the function again. In theory, this second invocation should be served by the existing function instance. We then wait 20 minutes to make sure that the next function call is a cold start again and restart the same loop. To collect more measurement points than twice every 20 minutes, we deploy parallel copies of a function that we cycle through.

Metrics. We consider three main metrics: request-response latency, unexpected cold starts, and long-term trends. For request-response latency, we use the billed duration that is output for every function execution by the FaaS platform. Unexpected cold starts are cold starts that occur directly after a function has already been called once, so that they should be warm. Unexpected cold starts imply that a platform was unable to find a warm function instance, possibly because it has been evicted due to resource contention.

Finally, we conduct a seasonal trend decomposition using LOESS (STL) [12], which can handle complex seasonal patterns. We use the following model for our data:

$$y_t = T_t + S_t + I_t$$

, where T_t represents the trend, S_t represents a seasonal component (days in our case), and I_t represents the remaining noise. We fit our data to this model using a method to create smoothed estimates

[3] https://github.com/umbrellerde/night-shift-code

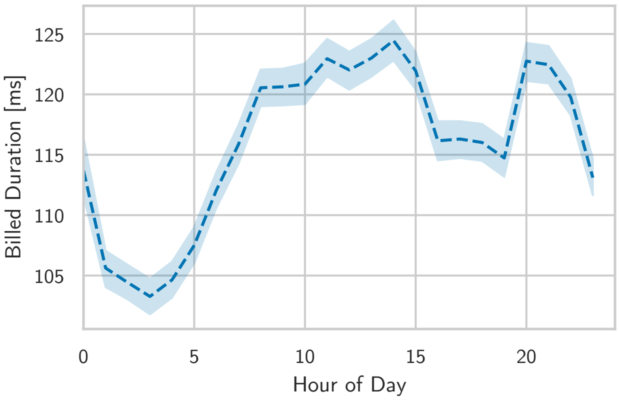


Figure 1: Billed Duration of warm calls to *float* with 128MB memory. During working hours, the billed duration increases by up to 15%. The area around the dashed line shows the 95% confidence interval.

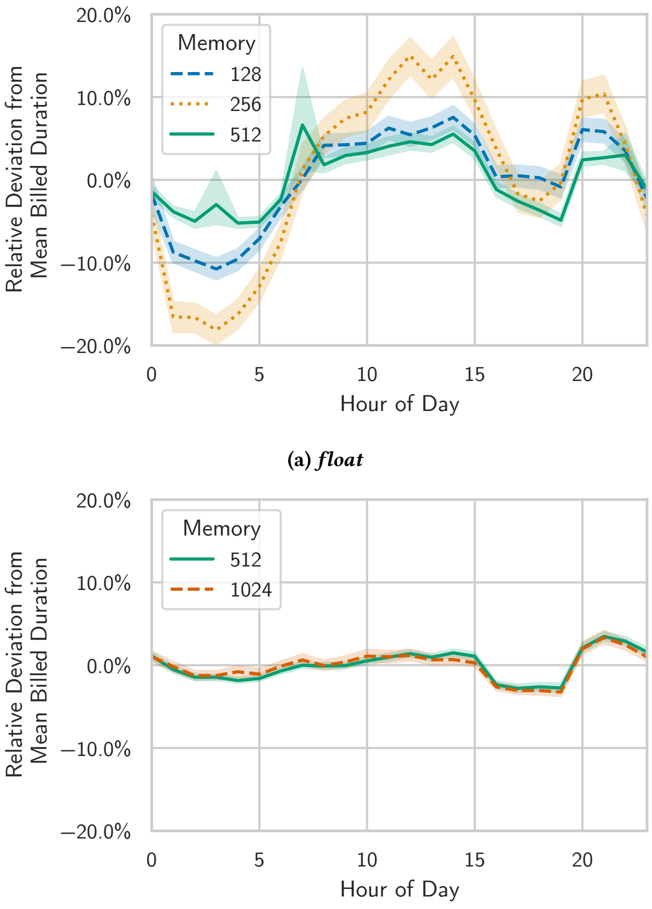
with a seasonality of one day. The trend component T_t shows the overall progress in the billed duration over the whole duration of the experiment. A non-flat trend line indicates that longer-term changes to the platform have occurred, e.g., long-term seasonal changes or updates to the platform that influenced performance. The seasonal component S_t shows periodic recurring deviations in the data from the trend. The I_t component is random noise centered around 0. Outliers in the noise indicate that the performance at specific times could not be explained by the previous two components. The trend component from the STL is also used for Change Point Detection [39], which can detect structural changes in data.

4 EXPERIMENTS

In this section, we present initial results of our experiments with Google Cloud Functions in the `europe-west3` region. The measurement period started on Dec 12, 2022 and ended on Feb 27, 2023. We report all execution times in local time (CET). We first analyze the performance variability of the platform (§4.1). Afterwards, we explore unexpected cold starts (§4.2) and outliers as well as change points and long-term trends in our data (§4.3).

4.1 Performance Variability

To analyze performance variability, we analyze the billed duration of comparable invocations. We show the billed duration of the *float* function with 128MB memory in Figure 1. We observe a clear performance increase during the night, with a noticeable latency spike during working hours. The average billed duration between 23:00 and 06:00 was 106ms, and increased by 15% to 122ms between 07:00 and 16:00. When aggregated by the day of the week, the average billed duration fluctuates between 113.86ms on Saturdays and 117.28ms on Mondays. Overall, billed duration is slightly lower on the average weekend compared to the start of the week. The weekly trend is much smaller than the daily trend, as the billed duration only decreases by ~4% during the weekend.



(a) *float*

(b) *ml*

Figure 2: Performance change over a day of warm instances ordered by memory size. The y-Axis is normalized to the average and shows the relative change, e.g., bigger values show a bigger deviation from the average billed duration. The area around the curves shows the 95% confidence interval.

When looking at larger memory sizes, the relative performance change over time becomes smaller. As shown in Figure 2, the *float* function with 128MB memory changes ~10% during a day, while 512MB only changes up to 5%. The *ml* functions with 512MB and 1024MB memory equally changed ~4% during an average day. Noticeably, the average billed duration of the *float* function with 256MB of memory differed more than 15% during a day. This can be explained by looking at the distribution of latency values over all invocations (Figure 3): Around 50% of functions with a memory size of 256MB follow the same distribution as functions with 512MB of memory, and the other half follows the same distribution as the 128MB functions. The results for the *matrix* function exhibit similar results but are omitted due to space constraints. The high variability in performance and uneven distribution of billed durations indicates that GCF internally uses 128MB and 512MB

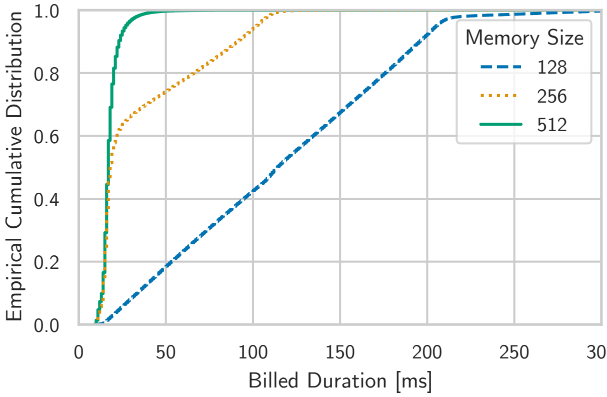


Figure 3: Cumulative distribution of billed durations of the float function without cold starts. We argue that GCF uses 128MB and 512MB containers to execute 256MB functions.

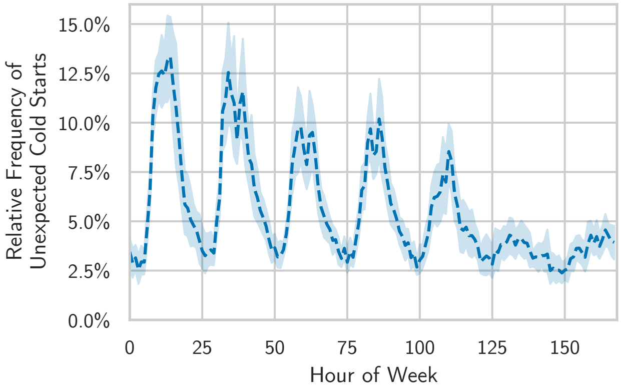


Figure 4: Relative frequency of unexpected cold starts, averaged by the hour of the week they happened in. On Mondays during the day, up to 13% of invocations can be unexpected cold starts, compared to less than 5% during the night and on weekends. The area around the curve shows the 95% confidence interval.

function instances to handle requests to the 256MB functions, as also shown by Malawski et al. [30].

While cold start durations also follow the daily patterns shown in §4.1, the configured memory size has no impact on the duration of cold starts. The billed durations of cold starts follow a normal distribution and are on average around 9-10x longer than the average warm latency, but all memory sizes follow the same distribution. This indicates that the cold start overhead is dependent on resources that can be configured by changing the function configuration.

4.2 Unexpected Cold Starts

We show the relative frequency of unexpected cold starts in Figure 4. While the billed duration of warm instances seems to only follow a daily trend, the frequency of unexpected cold starts has a weekly seasonality, with clear trends of increased cold starts during working hours. On average, the frequency of unexpected cold starts was 3.7% during the night (20:00—08:00), 3.6% during the weekend, 9.8% during working hours (09:00—17:00 Mon—Fri), and 12.3% during working hours on Monday. For comparison, there were less than 0.15% unexpected warm starts, where a function instance was still warm after more than 20 minutes.

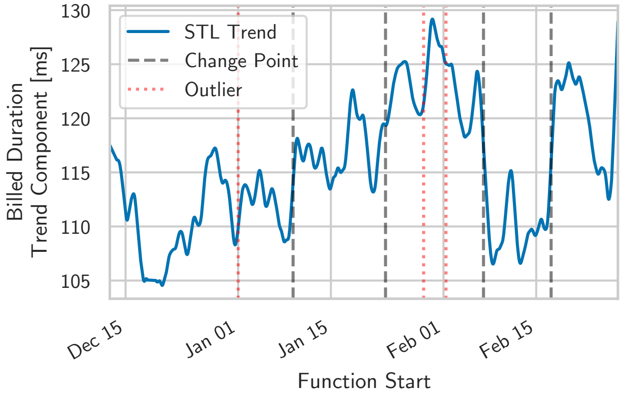


Figure 5: Trend Component of the STL, with Change Points and Outliers marked as horizontal lines.

4.3 Outliers & Long-Term Trend

Based on the STL introduced in §3, we show change points and outliers in Figure 5. We define an outlier as every hour during which the average execution duration is outside the fourth interquartile range, i.e., more than four times the difference between the first and third quartile, away from the average. Our data contains four outliers, which were all within three days of the turn of the month. This indicates to us that the platform is under unusual load at these times, possibly due to additional load from monthly jobs.

All change points, i.e., points when the average execution duration changed, occurred during the night, indicating that they coincide with scheduled updates to the platform.

Over our whole measurement period, there is no clear permanent trend towards better or worse overall performance. Based on the long-term study by Eismann et al. [16], we only expect to find permanent trends in longer measurement periods. Compared to the authors' study of AWS, which finds statistical trends below 10%, our trend component changes between 105ms during December to 128ms during February, a 21% increase.

5 IMPLICATIONS

In this section, we discuss how our findings impact serverless systems. First, we focus on the implications for benchmarking. Afterwards, we describe implications on serverless applications.

5.1 Validity of Benchmarks

When running benchmarks, researchers want to minimize effects of external factors to their measurements, which otherwise might confound results [9]. In the case of benchmarking serverless systems,

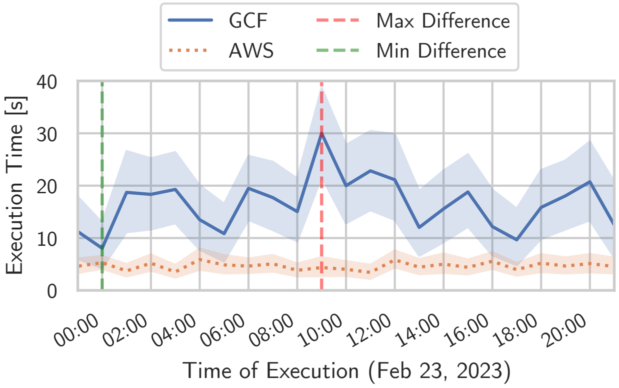


Figure 6: Execution Times of the `110.dynamic-html` Benchmark on GCF and Lambda (excerpt). The minimum and maximum measured difference of the whole experiment are marked as horizontal lines.

we have shown that the number of cold starts and the performance of functions undergoes changes within a single day. For comparative performance studies, benchmark results are only comparable if experiments are conducted at comparable times. If an experiment is short, i.e., does not capture the performance variation of an entire day, daily variations can skew results. Similarly, long-term performance changes can impact measurements taken over longer periods of time, e.g., a three-day study with one group benchmarked over the weekend and another benchmarked during the week. A possible remediation for such experiments is adopting parallel benchmarking techniques such as duet benchmarking [10].

We show an overview of existing publications on cloud FaaS performance measurements surveyed for this paper in Table 1. For every paper, we give an overview of cloud platforms under test, cloud regions, and time of day of the benchmark execution, if stated. Based on this metadata, we must assume that published results could be affected by daily performance variations if execution time is not given or different regions are used (implying different time zones). Overall, 10 out of a total of 16 papers do not provide sufficient information to rule out effects of performance variability. While this does not mean that the reported results are invalid, it shows that the research community has not paid enough attention to these effects in performance measurements.

As an example, we replicate an experiment from Copik et al. [13] that compares the execution time of a dynamic HTML generator (`110.dynamic-html`) between AWS Lambda and GCF. We deploy this function in the `eu-central-1` (Lambda) and `europe-west3` (GCF) regions with 128MB of memory and run 50 sequential invocations every hour over the course of five days. As shown in Figure 6, the performance on GCF exhibits temporal variations that can skew results: The smallest performance difference during our experiments happened on the 23rd of February 2023 at midnight, when the average performance difference was <2.8s (GCF 52% slower). Shortly after, at 09:00, we observe the largest performance difference with the average difference increasing to >25s (GCF 6.8× slower). These performance changes show that short-term variation in performance between serverless platforms can have a significant impact on benchmarking results and need to be controlled for. We recommend repeating experiments over the course of a day and mentioning execution time when describing experiment setup.

5.2 Application Performance

The performance variability that we have shown for GCF affects serverless applications in several ways: During daytime, functions have increased latency, suffer more cold starts, and their execution cost is increased due to the pay-by-second billing model. For low-latency, event-driven functions, it is not feasible to postpone their execution to the night or a weekend to decrease costs. A possible way forward, however, is to shift function execution to another cloud region with better performance. This may increase network latency and transmission costs, but an up to 20% reduction on function execution times and the associated decrease in costs can outweigh this overhead for long(er)-running functions. Such an approach requires constant evaluation of FaaS platform performance in different regions, possibly also based on application metrics [8].

Researchers have also proposed systems that adapt FaaS applications to improve performance and cost on cloud FaaS platforms. Such systems rely on initial performance measurements of applications on cloud platforms [14, 15, 18, 24] or a feedback loop between platform, application, and optimizer [36]. Both approaches are affected by performance variability of the FaaS platform, as the optimizer or model cannot differentiate between performance changes that are caused by deployment updates and those that are caused by platform instability. A possible way forward for these systems is to control for temporal performance variations, e.g., by deploying multiple parameter sets concurrently or performing longer initial measurements.

6 LIMITATIONS & FUTURE WORK

We have shown considerable performance variability in Google Cloud Functions and discussed how these affect applications and performance measurement research. We plan to build on this initial work in the future to arrive at a more holistic view of performance variability in cloud FaaS platforms.

FaaS Platforms. Our initial experiments are limited mostly to GCF, with some additional validation on AWS Lambda. Although we have seen that in our experiments, Lambda suffers from less performance variability than GCF, parameters such as memory size, hardware architecture, geographical region, or programming language could further influence variability. We plan to conduct additional experiments on different FaaS platforms in the future, controlling for these additional parameters.

Platform Changes. FaaS platforms are evolving quickly, and our measurements and experiments can only capture the behavior of such a platform at a specific point in time. Continuous updates could increase or even eliminate the performance variability effects we observe in the future, making continuous measurements important. Subsequently, researchers that want to account for the described behavior in their own measurements on FaaS platforms should

Table 1: Selection of serverless benchmarks and whether they *could* be impacted by short-term performance fluctuations. Bold lines are papers where a daily impact could not be ruled out.

Authors	Cloud Platform(s)	Region(s) Specified	Execution Time of Day
Copik et al. [13]	**AWS, Azure, GCP**	**US (AWS), EU (Azure, GCP)**	**–**
Eismann et al. [16]	AWS	–	daily (19:00) for 10 months
Figiela et al. [19]	**AWS, Azure, IBM, GCP**	**EU (AWS), US (GCP)**	**every 5 minutes**
Jackson et al. [25]	AWS, Azure	–	hourly over 6 days
Grambow et al. [22]	**AWS, Azure, GCP**	**EU**	**–**
Kim et al. [26]	**AWS, Azure, GCP**	**–**	**–**
López et al. [28]	**AWS, Azure, IBM**	**–**	**–**
Malawski et al. [30]	AWS, GCP	EU (AWS), US (GCP)	“permanently”
Manner et al. [31]	AWS, Azure	–	–
McGrath et al. [32]	**AWS, Azure, GCP**	**–**	**–**
Pelle et al. [34]	AWS	“multiple regions”	–
Scheuner et al. [35]	**AWS**	**US**	**–**
Shahrad et al. [37]	Azure	“entire infrastructure”	every minute
Somu et al. [38]	**AWS, GCP**	**–**	**–**
Werner et al. [41]	**AWS**	**EU**	**–**
Zhang et al. [43]	**AWS, GCP**	**–**	**–**

conduct their own experiments using our methodology, as our measurement results may be outdated by then.

Performance Dimensions. The functions we use in our experiments are CPU-bound, which gives a good indication for general platform performance and minimizes the impact of the performance of external services. Beyond CPU performance, other resource metrics such as memory access, disk I/O, and network latency or bandwidth can be affected by platform variability. As our findings may not be unconditionally applicable to workloads that are bound in these dimensions, we will investigate their variation with additional functions in the future.

7 CONCLUSION

In this paper, we have presented the results of our multi-month performance variability benchmark measuring the performance of multiple functions on Google Cloud Functions every 40s. Our results show that the execution duration of a function varies up to 15% per day, and the frequency of unexpected cold starts varies over a week and per day. While more resources reduce daily performance variability, they do not shorten cold start durations. By looking at the long-term trend, we identify likely updates to the platform and outlier behavior around the turn of the month. These results have implications for both researchers and practitioners.

REFERENCES

[1] Amazon Web Services. 2022. *AWS Lambda Service Level Agreement.* Retrieved Feburary 24, 2023 from https://aws.amazon.com/lambda/sla/

[2] Ioana Baldini, Perry Cheng, Stephen J. Fink, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Philippe Suter, and Olivier Tardieu. 2017. The serverless trilemma: function composition for serverless computing. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (Vancover, BC, Canada) *(Onward! 2017)*. Association for Computing Machinery, New York, NY, USA, 89–103. https://doi.org/10.1145/3133850.3133855

[3] Daniel Bardsley, Larry Ryan, and John Howard. 2018. Serverless Performance and Optimization Strategies. In *Proceedings of the 2018 IEEE International Conference on Smart Cloud* (New York, NY, USA) *(SmartCloud)*. IEEE, New York, NY, USA, 19–26. https://doi.org/10.1109/SmartCloud.2018.00012

[4] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. 2016. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction* (Tokyo, Japan) *(ICMI '16)*. Association for Computing Machinery, New York, NY, USA, 279–283. https://doi.org/10.1145/2993148.2993165

[5] David Bermbach, Abhishek Chandra, Chandra Krintz, Aniruddha Gokhale, Aleksander Slominski, Lauritz Thamsen, Everton Cavalcante, Tian Guo, Ivona Brandic, and Rich Wolski. 2021. On the Future of Cloud Engineering. In *Proceedings of the 9th IEEE International Conference on Cloud Engineering* (San Francisco, CA, USA) *(IC2E 2021)*. ACM, New York, NY, USA, 264–275. https://doi.org/10.1109/IC2E52221.2021.00044

[6] David Bermbach, Ahmet-Serdar Karakaya, and Simon Buchholz. 2020. Using Application Knowledge to Reduce Cold Starts in FaaS Services. In *Proceedings of the 35th ACM Symposium on Applied Computing* (Brno, Czech Republic) *(SAC '20)*. Association for Computing Machinery, New York, NY, USA, 134–143. https://doi.org/10.1145/3341105.3373909

[7] David Bermbach and Stefan Tai. 2011. Eventual Consistency: How Soon is Eventual? An Evaluation of Amazon S3's Consistency Behavior. In *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing* (Lisbon, Portugal) *(MW4SOC '11)*. ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/2093185.2093186

[8] David Bermbach and Stefan Tai. 2014. Benchmarking Eventual Consistency: Lessons Learned from Long-Term Experimental Studies. In *Proceedings of the 2nd IEEE International Conference on Cloud Engineering* (Boston, MA, USA) *(IC2E 2014)*. IEEE, New York, NY, USA, 47–56. https://doi.org/10.1109/IC2E.2014.37

[9] David Bermbach, Erik Wittern, and Stefan Tai. 2017. *Cloud Service Benchmarking: Measuring Quality of Cloud Services from a Client Perspective.* Springer, Cham, Switzerland.

[10] Lubomír Bulej, Vojtěch Horký, Petr Tuma, François Farquet, and Aleksandar Prokopec. 2020. Duet benchmarking: Improving measurement accuracy in the cloud. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (Edmonton AB, Canada) *(ICPE '20)*. Association for Computing Machinery, New York, NY, USA, 100–107. https://doi.org/10.1145/3358960.3379132

[11] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2019. The rise of serverless computing. *Commun. ACM* 62, 12 (Nov. 2019), 44–54. https://doi.org/10.1145/3368454

[12] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics* 6, 1 (March 1990), 3–73.

[13] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. SeBS: a serverless benchmark suite for function-as-a-service computing. In *Proceedings of the 22nd International Middleware Conference* (Quebec City, QC, Canada) *(Middleware 2021)*. Association for Computing Machinery, New York, NY, USA, 64–78. https://doi.org/10.1145/3464298.3476133

[14] Robert Cordingly, Sonia Xu, and Wesley Lloyd. 2022. Function Memory Optimization for Heterogeneous Serverless Platforms with CPU Time Accounting. In *Proceedings of the 10th IEEE International Conference on Cloud Engineering* (Asilomar, CA, USA) *(IC2E 2022)*. IEEE, New York, NY, USA, 104–115. https://doi.org/10.1109/IC2E55432.2022.00019

[15] Janos Czentye, Istvan Pelle, Andras Kern, Balazs Peter Gero, Laszlo Toka, and Balazs Sonkoly. 2019. Optimizing Latency Sensitive Applications for Amazon's Public Cloud Platform. In *Proceedings of the 2019 IEEE Global Communications Conference* (Waikoloa Village, HI, USA) *(GLOBECOM)*. IEEE, New York, NY, USA, 1–7. https://doi.org/10.1109/GLOBECOM38437.2019.9013988

[16] Simon Eismann, Diego Elias Costa, Lizhi Liao, Cor-Paul Bezemer, Weiyi Shang, André van Hoorn, and Samuel Kounev. 2022. A case study on the stability of performance tests for serverless applications. *Journal of Systems and Software* 189, Article 111294 (July 2022). https://doi.org/10.1016/j.jss.2022.111294

[17] Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, and Alexandru Iosup. 2020. Serverless Applications: Why, When, and How? *IEEE Software* 38, 1 (Sept. 2020), 32–39. https://doi.org/10.1109/MS.2020.3023302

[18] Tarek Elgamal. 2018. Costless: Optimizing Cost of Serverless Computing through Function Fusion and Placement. In *Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing* (Seattle, WA, USA) *(SEC 2018)*. IEEE, New York, NY, USA, 300–312. https://doi.org/10.1109/SEC.2018.00029

[19] Kamil Figiela, Adam Gajek, Adam Zima, Beata Obrok, and Maciej Malawski. 2018. Performance evaluation of heterogeneous cloud functions. *Concurrency and Computation: Practice and Experience* 30, 23, Article e4792 (Aug. 2018). https://doi.org/10.1002/cpe.4792

[20] Samuel Ginzburg and Michael J. Freedman. 2021. Serverless Isn't Server-Less: Measuring and Exploiting Resource Variability on Cloud FaaS Platforms. In *Proceedings of the 2020 Sixth International Workshop on Serverless Computing* (Delft, Netherlands) *(WoSC'20)*. Association for Computing Machinery, New York, NY, USA, 43–48. https://doi.org/10.1145/3429880.3430099

[21] Google Cloud. 2021. *Cloud Functions Service Level Agreement (SLA)*. Retrieved Feburary 24, 2023 from https://cloud.google.com/functions/sla

[22] Martin Grambow, Tobias Pfandzelter, Luk Burchard, Carsten Schubert, Max Zhao, and David Bermbach. 2021. BeFaaS = An Application-Centric Benchmarking Framework for FaaS Platforms. In *Proceedings of the 9th IEEE International Conference on Cloud Engineering* (San Francisco, CA, USA) *(IC2E 2021)*. IEEE, New York, NY, USA, 1–8. https://doi.org/10.1109/IC2E52221.2021.00014

[23] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2016. Serverless computation with openlambda. In *Proceedings of the 8th USENIX Workshop on Hot Topics in Cloud Computing* (Denver, CO, USA) *(HotCloud '16)*. USENIX Association, Berkeley, CA, USA, 33–39.

[24] Shay Horovitz, Roei Amos, Ohad Baruch, Tomer Cohen, Tal Oyar, and Afik Deri. 2018. FaaStest - Machine Learning Based Cost and Performance FaaS Optimization. In *Proceedings of the International Conference on the Economics of Grids, Clouds, Systems, and Services* (Pisa, Italy) *(GECON 2018)*. Springer, Cham, Switzerland, 171–186. https://doi.org/10.1007/978-3-030-13342-9_15

[25] David Jackson and Gary Clynch. 2018. An investigation of the impact of language runtime on the performance and cost of serverless functions. In *Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion* (Zurich, Switzerland) *(UCC Companion)*. IEEE, New York, NY, USA, 154–160. https://doi.org/10.1109/UCC-Companion.2018.00050

[26] Jeongchul Kim and Kyungyong Lee. 2019. FunctionBench: A Suite of Workloads for Serverless Cloud Function Service. In *Proceedings of the 2019 IEEE International Conference on Cloud Computing* (Milan, Italy) *(CLOUD)*. IEEE, New York, NY, USA, 502–504. https://doi.org/10.1109/CLOUD.2019.00091

[27] Danielle Lambion, Robert Schmitz, Robert Cordingly, Navid Heydari, and Wes Lloyd. 2022. Characterizing X86 and ARM Serverless Performance Variation: A Natural Language Processing Case Study. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering* (Bejing, China). Association for Computing Machinery, New York, NY, USA, 69–75. https://doi.org/10.1145/3491204.3543506

[28] Pedro García López, Marc Sánchez-Artigas, Gerard París, Daniel Barcelona Pons, Álvaro Ruiz Ollobarren, and David Arroyo Pinto. 2018. Comparison of FaaS orchestration systems. In *Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion* (Zurich, Switzerland) *(UCC Companion)*. IEEE, New York, NY, USA, 148–153. https://doi.org/10.1109/UCC-Companion.2018.00049

[29] Nima Mahmoudi and Hamzeh Khazaei. 2021. Temporal Performance Modelling of Serverless Computing Platforms. In *Proceedings of the 2020 Sixth International Workshop on Serverless Computing* (Delft, Netherlands) *(WoSC'20)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3429880.3430092

[30] Maciej Malawski, Kamil Figiela, Adam Gajek, and Adam Zima. 2017. Benchmarking Heterogeneous Cloud Functions. In *Proceedings of Euro-Par 2017: Parallel Processing Workshops* (Santiago de Compostela, Spain) *(Euro-Par 2017)*. Springer, Cham, Switzerland, 415–426. https://doi.org/10.1007/978-3-319-75178-8_34

[31] Johannes Manner, Martin Endreß, Tobias Heckel, and Guido Wirtz. 2018. Cold Start Influencing Factors in Function as a Service. In *Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion* (Zurich, Switzerland) *(UCC Companion)*. IEEE, New York, NY, USA, 181–188. https://doi.org/10.1109/UCC-Companion.2018.00054

[32] Garrett McGrath and Paul R Brenner. 2017. Serverless computing: Design, implementation, and performance. In *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops* (Atlanta, GA, USA) *(ICDCSW)*. IEEE, New York, NY, USA, 405–410. https://doi.org/10.1109/ICDCSW.2017.36

[33] Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, and Martina Maggio. 2020. Towards Performance Modeling of Speculative Execution for Cloud Applications. In *Proceedings of the Companion of the ACM/SPEC International Conference on Performance Engineering* (Edmonton AB, Canada) *(ICPE '20)*. Association for Computing Machinery, New York, NY, USA, 17–19. https://doi.org/10.1145/3375555.3384379

[34] István Pelle, János Czentye, János Dóka, and Balázs Sonkoly. 2019. Towards latency sensitive cloud native applications: A performance study on AWS. In *Proceedings of the 2019 IEEE 12th International Conference on Cloud Computing* (Milan, Italy) *(CLOUD)*. IEEE, New York, NY, USA, 272–280. https://doi.org/10.1109/CLOUD.2019.00054

[35] Joel Scheuner, Marcus Bertilsson, Oskar Grönqvist, Henrik Tao, Henrik Lagergren, Jan-Philipp Steghöfer, and Philipp Leitner. 2022. TriggerBench: A Performance Benchmark for Serverless Function Triggers. In *Proceedings of the 2022 IEEE International Conference on Cloud Engineering* (Asilomar, CA, USA) *(IC2E 2022)*. IEEE, New York, NY, USA, 96–103. https://doi.org/10.1109/IC2E55432.2022.00018

[36] Trever Schirmer, Joel Scheuner, Tobias Pfandzelter, and David Bermbach. 2022. Fusionize: Improving Serverless Application Performance through Feedback-Driven Function Fusion. In *Proceedings of the 10th IEEE International Conference on Cloud Engineering* (Asilomar, CA, USA) *(IC2E 2022)*. IEEE, New York, NY, USA, 85–95. https://doi.org/10.1109/IC2E55432.2022.00017

[37] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In *Proceedings of the 2020 USENIX Annual Technical Conference* (Virtual Event, USA) *(USENIX ATC '20)*. USENIX, Berkeley, CA, USA, 205–218.

[38] Nikhila Somu, Nilanjan Daw, Umesh Bellur, and Purushottam Kulkarni. 2020. PanOpticon: A Comprehensive Benchmarking Tool for Serverless Applications. In *Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS* (Bengaluru, India) *(COMSNETS 2020)*. IEEE, New York, NY, USA, 144–151. https://doi.org/10.1109/COMSNETS48256.2020.9027346

[39] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167, Article 107299 (Feb. 2020). https://doi.org/10.1016/j.sigpro.2019.107299

[40] Erwin van Eyk, Joel Scheuner, Simon Eismann, Cristina L. Abad, and Alexandru Iosup. 2020. Beyond Microbenchmarks: The SPEC-RG Vision for a Comprehensive Serverless Benchmark. In *Proceedings of the Companion of the ACM/SPEC International Conference on Performance Engineering* (Edmonton AB, Canada) *(ICPE '20)*. Association for Computing Machinery, New York, NY, USA, 26–31. https://doi.org/10.1145/3375555.3384381

[41] Sebastian Werner, Jörn Kuhlenkamp, Markus Klems, Johannes Müller, and Stefan Tai. 2018. Serverless Big Data Processing using Matrix Multiplication as Example. In *Proceedings of the 2018 IEEE International Conference on Big Data* (Seattle, WA, USA) *(Big Data)*. IEEE, New York, NY, USA, 358–365. https://doi.org/10.1109/BigData.2018.8622362

[42] Sebastian Werner and Trever Schirmer. 2022. Hardless: A Generalized Serverless Compute Architecture for Hardware Processing Accelerators. In *Proceedings of the 10th IEEE International Conference on Cloud Engineering* (Asilomar, CA, USA) *(IC2E 2022)*. IEEE, New York, NY, USA, 79–84. https://doi.org/10.1109/IC2E55432.2022.00016

[43] Miao Zhang, Yifei Zhu, Cong Zhang, and Jiangchuan Liu. 2019. Video processing with serverless computing: A measurement study. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Amherst, MA, USA) *(NOSSDAV '19)*. Association for Computing Machinery, New York, NY, USA, 61–66. https://doi.org/10.1145/3304112.3325608

[44] Haidong Zhao, Zakaria Benomar, Tobias Pfandzelter, and Nikolaos Georgantas. 2022. Supporting Multi-Cloud in Serverless Computing. In *Proceedings of the 15th IEEE/ACM International Conference on Utility and Cloud Computing Companion* (Vancouver, WA, USA) *(UCC '22)*. IEEE, New York, NY, USA.