the SUT. It provides a set of search strategies to evaluate possible combinations of resources and loads based on configurable service level objectives (SLOs). To store and analyze results, it utilizes a persistent volume, a Grafana server, and a set of Jupyter notebooks.

6 FUTURE WORK AND CONCLUSION

The platform we have presented in this submission, is still a work in progress and there are several features and improvements in the works.

One of these improvements touches on the currently used threshold based alerts via assertions: these tend to be flaky, and we have considered using one of the change-point detection algorithms to introduce outlier detection and reduce false positives, e.g. by implementing E-Divisive with Means or similar approaches [15].

In addition, we plan to collect aggregated statistics on application-side metrics such as CPU and memory utilization, profiling and tracing data, etc. and bundle them with the results summary to provide resource utilization comparisons and regression analysis.

Developer experience with the platform can be further enhanced by introducing an interactive integration with Slack, e.g. by providing an easy way to re-run a failed load test directly from the Slack message (party implemented), or by introducing a chat bot functionality to manage load tests without the need for the Argo WebUI or CLI.

Another improvement to the developer experience is planned integration with the internal development portal based on Backstage [2]. This would allow performance measurement data to be embedded into a service health scorecard, to track the progress on "Performance Engineering Framework" action items and provide a single point of entry for all interactions with the platform directly from Backstage.

To address the scheduling delays, completely separate load generation from the system under test, and have a way to stress all components via external endpoints, we had considered setting up an additional Kubernetes cluster dedicated to load testing. However, this would add significant fixed costs and maintenance overhead. At this point, we don't have a specific use case that would justify this effort, but we may revisit this idea in the future. A more efficient solution would be to have a dedicated pool of nodes for running load tests and introduce a synchronization checkpoint.

In this submission, we have presented our solution for a self-service performance testing platform for microservices. This is a scalable, fully integrated performance testing framework built from open-source components by a platform performance engineering team that has been widely adopted in a large engineering organization.

REFERENCES

[1] Argo. 2018. *Argo Workflows.* https://argo-workflows.readthedocs.io

[2] Backstage. 2020. *What is Backstage?* https://backstage.io/docs/overview/what-is-backstage

[3] Chris Baeckstrom. 2021. *Comparing K6, Gatling and JMeter.* https://www.redline13.com/blog/2022/09/comparing-k6-gatling-and-jmeter/

[4] Jeremy J. Carroll, Pankaj Anand, and David Guo. 2021. Preproduction Deploys: Cloud-Native Integration Testing. arXiv:2110.08588 [cs.NI]

[5] Adrian Cockcroft. 2016. *Microservices workshop.* https://www.slideshare.net/adriancockcroft/microservices-workshop-craft-conference

[6] Melvin E. Conway. 1968. How Do Committees Invent? *Datamation* (April 1968). http://www.melconway.com/research/committees.html

[7] Datadog. 2015. *DogStatsD.* https://docs.datadoghq.com/developers/dogstatsd/

[8] Gatling. 2013. *Gatling.* https://gatling.io/docs/gatling/

[9] Mohammad Hajjat, Ruiqi Liu, Yiyang Chang, T. S. Eugene Ng, and Sanjay Rao. 2015. Application-specific configuration selection in the cloud: Impact of provider policy and potential of systematic testing. In *2015 IEEE Conference on Computer Communications (INFOCOM).* 873–881. https://doi.org/10.1109/INFOCOM.2015.7218458

[10] Sen He, Glenna Manns, John Saunders, Wei Wang, Lori Pollock, and Mary Lou Soffa. 2019. A statistics-based performance testing methodology for cloud applications. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019).* Association for Computing Machinery, New York, NY, USA, 188–199. https://doi.org/10.1145/3338906.3338912

[11] Sören Henning and Wilhelm Hasselbring. 2021. Theodolite: Scalability Benchmarking of Distributed Stream Processing Engines in Microservice Architectures. *Big Data Research* 25 (July 2021), 100209. https://doi.org/10.1016/j.bdr.2021.100209

[12] Henrik Ingo and David Daly. 2020. Automated system performance testing at MongoDB. In *Proceedings of the workshop on Testing Database Systems (SIGMOD-/PODS '20).* ACM. https://doi.org/10.1145/3395032.3395323

[13] Kubernetes. 2020. *Kubernetes Documentation Concepts Workloads Workload Management Jobs.* https://kubernetes.io/docs/concepts/workloads/controllers/job/

[14] James Lewis and Martin Fowler. 2014. *Microservices.* https://martinfowler.com/articles/microservices.html

[15] Mark Leznik, Md Shahriar Iqbal, Igor Trubin, Arne Lochner, Pooyan Jamshidi, and André Bauer. 2022. Change Point Detection for MongoDB Time Series Performance Regression. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering* (Bejing, China) *(ICPE '22).* Association for Computing Machinery, New York, NY, USA, 45–48. https://doi.org/10.1145/3491204.3527488

[16] MongoDB. 2017. *Evergreen.* https://www.mongodb.com/blog/post/testing-linearizability-jepsen-evergreen-call-me-continuously

[17] Vishnu Murty. 2023. Distributed WorkLoad Generator for Performance & Load Testing Using Opensource Technologies. https://raw.githubusercontent.com/ltb2023/ltb2023.github.io/master/slides/LTB23_VMurty.pdf

[18] Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim von Kistowski, Ahmed Ali-Eldin, Cristina L. Abad, José Nelson Amaral, Petr Tůma, and Alexandru Iosup. 2021. Methodological Principles for Reproducible Performance Evaluation in Cloud Computing. *IEEE Transactions on Software Engineering* 47, 8 (2021), 1528–1543. https://doi.org/10.1109/TSE.2019.2927908

[19] Matt Ranney. 2016. *What I Wish I Had Known Before Scaling Uber to 1000 Services.* https://www.youtube.com/watch?v=kb-m2fasdDY

[20] Tanya Reilly. 2022. *The Staff Engineer's Path: A Guide For Individual Contributors Navigating Growth and Change.* O'Reilly Media.

[21] M. Skelton, M. Pais, and R. Malan. 2019. *Team Topologies: Organizing Business and Technology Teams for Fast Flow.* IT Revolution. https://books.google.fi/books?id=oFdRuAEACAAJ

[22] Cindy Sridharan. 2018. *Testing in Production, the safe way.* https://copyconstruct.medium.com/testing-in-production-the-safe-way-18ca102d0ef1

[23] Rafael Winterhalter. 2014. *Why runtime code generation?* https://bytebuddy.net

[24] Ailin Yang. 2022. *"Noisy Neighbors" Problem in Kubernetes.* https://www.intel.com/content/www/us/en/developer/articles/technical/noisy-neighbors-problem-in-kubernetes.html