adapters guarantee the reusability of the same driver for different types of performance test [4].

The benchmark life cycle starts with SUT's deployment by exploiting the Docker containerization technology. Thus it provides for the replicability of the experiments by ensuring the SUT is always deployed in the exact same initial state. While containerization technologies introduce some overhead on system's performance, a recent reliable performance analysis of Docker [1] indicated that, if carefully configured, Docker reaches near-zero overhead. During the performance test execution, the BenchFlow framework monitors the experiment's execution state to gather resource utilization (e.g., CPU, RAM, Network) data, using the lightweight monitors of the Docker stats API. When the performance test execution is complete, a set of collectors gather the raw performance data from the distributed infrastructure on which the performance test has been performed, and send them to a central storage service, e.g., Amazon S3, or Minio[5]. We rely on a storage service, since the data have to be efficiently accessed from the components computing the metrics. To abstract from the different data formats that different SUTs might have, the performance data are then transformed to a canonical meta-model. After the transformation, the performance data are stored in a Cassandra[6] database (DB), and the performance metrics and KPIs are computed. Cassandra is a powerful DB for storing and accessing performance data from the service that computes metrics on top of them. The computation is performed by relying on Apache Spark[7], a fast, general-purpose engine for large-scale data processing. Before the computation of the metrics is triggered, the BenchFlow framework checks the logs collected from the SUT to identify execution errors and validate the experiment.

The orchestration of the performance test execution, the data collection, and the performance data analysis, is delegated to Apache Kafka[8], a publish-subscribe messaging framework. We have introduced this state-of-the-art framework to decouple the benchmark execution managed by the Faban Harness, from the performance metrics computation, and thus pipeline the gathering of performance data with the corresponding analytics, which can be performed offline.

3. GOALS OF THE DEMONSTRATION

The BenchFlow framework is currently used for benchmarking Workflow Management Systems, and has been successfully applied in different experiments [2]. In the demo we will present a walk-through use case that shows BenchFlow framework's capabilities, both from the perspective of performance researchers and performance testers. During the demo, the framework will be pre-installed on multiple servers, in a dedicated, reliable and controlled environment, which should be accessible over VPN from the conference venue. We will go through the end-to-end performance test process, by defining, submitting and monitoring the performance test, describing the SUT deployment definition, and accessing the automatically calculated performance metrics.

Defining the performance test: the framework simplifies the definition of load drivers' behavior with Faban, through load driver definition descriptors. We will define a performance test, involving a distributed system of multiple microservices and their DBs, by means of a benchmark definition file, where we can define all the performance test parameters (e.g., a load test). The framework will take care of translating the performance test definition to the actual format used internally by Faban. **Describing the SUT deployment definition**: the deployment is performed by relying on the Docker Compose tool. This ensures that each SUT deployment configuration (e.g., different amounts of RAM) and its initial state can be precisely described and executed obtaining the exact same initial conditions for the experiment. Paired with Docker Swarm[9] it is possible to automate SUT's deployment on a distributed infrastructure. **Submitting and monitoring the performance test**: we will submit the performance test we have previously defined, and monitor its execution status. The test will last 1 minute and will be repeated 3 times. **Accessing the automatically calculated performance metrics**: when the benchmark execution is complete and the performance data are ready to be explored, we will visualize them at the end of the demo.

During the demo, we aim to demonstrate the flexibility and the simplicity of using the BenchFlow framework. Once defined by means of the BenchFlow benchmark definition file and the Docker Compose deployment definition file, the experiments can be replicated multiple times in a fully automated way to obtain reliable and verified results.

4. CONCLUSIONS AND FUTURE WORK

The BenchFlow framework greatly simplifies and accelerates the definition and execution of reliable, repeatable and reusable performance tests. It does so by integrating two powerful technologies: Faban and Docker, and building on top of their functionality to provide a complete framework for automated performance test execution over a distributed environment. The next planned steps concern enabling the simplified definition of additional performance test types (e.g., spike testing), as well as providing stronger performance test validation and means for performance test result analysis and exploration. Moreover we also plan to collaborate with the ICPE and the SPEC community, to drive the framework's development and strengthen its functionality.

5. ACKNOWLEDGMENTS

This work is partially funded by the Swiss National Science Foundation with the BenchFlow - A Benchmark for Workflow Management Systems (Grant Nr. 145062) project.

6. REFERENCES

[1] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. Technical report, IBM, July 2014.

[2] V. Ferme, A. Ivanchikj, and C. Pautasso. A framework for benchmarking BPMN 2.0 workflow management systems. Proc. of BPM '15, pages 251–259, 2015.

[3] K. Huppler. The art of building a good benchmark. TPCTC 2009, pages 18–30. Springer, 2009.

[4] I. Molyneaux. *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly, 2nd edition, 2014.

[5] https://www.minio.io

[6] http://cassandra.apache.org

[7] http://spark.apache.org

[8] http://kafka.apache.org

[9] https://www.docker.com/docker-swarm