their solutions to the serverless architecture to benefit from these advantages and provide better services to their costumers. Such companies include Reuters, iRobot, Autodesk and many more that reportedly use AWS Lambda to better serve their customers and internal processes [6]. In Toader et al. [23], the authors proposed a detailed implementation of a serverless graph processing framework implemented with AWS Lambda which benefits from automated resource autoscaling and provisioning. Graphless abstracts away resource management and configuration from the users. This will benefit end users who are not familiar with HPC concepts. However, the authors have shown that because of the variability of network characteristics under certain communication intensive workloads, the Graphless efficiency degrades.

In [24], the authors proposed a serverless framework for machine learning tasks, called Siren, which is fully asynchronous and achieves different levels of parallelism and elasticity. Siren, through stateless serverless functions, much like Graphless eliminates the burden of resource management and scaling machine learning algorithms imposed on the end users by the current well established serverfull frameworks. The Siren framework is deployed on the AWS Lambda serverless platform. The authors in [3] discussed economic and architectural benefits of serverless computing and study two real world services, namely MindMup and Yubl [3], that have been migrated to and adopted serverless computing. They discussed how MindMup and Yubl could benefit from serverless deployment; these benefits include reduced time to feature delivery and time to market for developers and faster request processing for customers. However, it is mentioned that serverless platforms are not well suited for mission critical and time sensitive tasks [3].

Moreover, a recent innovative step and improvement in the serverless computing research and development is the replacement of functions in the FaaS serverless architecture with containers and images from the developers. These containers are stateless and their autoscaling and all the other deployment details are handled by the cloud providers. Such container-oriented serverless solutions include products such as the Google Cloud Run [15] or the Amazon Fargate [5], both of which are serverless solutions for containers. The upside of serverless containers compared to functions is the level of concurrency introduced by containers which refers to the number of user requests processed by each container. The developers are free to assign concurrency levels higher than 1 to each container, for example, a concurrency level of 4 indicates that 4 user requests are processed by each container. The concurrency level in each function in the FaaS serverless solutions is set to 1, meaning that each function processes 1 request at a time [14].

7 CONCLUSION

With the advent of serverless computing, several monolith applications which were previously developed for a single-core or multicore execution environment are implemented from scratch following the serverless paradigm to take advantages of auto-scaling and automated resource provisioning. In this article, by observing the gap in the literature and the lack of studies concerning the performance of the serverless implementation of financial services, we present the migration journey of a FinTech application from its monolithic form to a high-performance serverless implementation. Based on our evaluations, the proposed serverless implementation outperforms the previous monolith serverfull implementation by 93 times, while increasing the cost by 50%. Our cost calculation does not take into account the under utilization of the serverfull infrastructure and the investment required to build the serverless system. In conclusion, the serverless implementation provides unparalleled speedup and performance improvement over the serverfull implementation without making drastic changes to the software design.

REFERENCES

[1] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Martin L Abbott and Michael T Fisher. 2009. *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*. Pearson Education.

[3] Gojko Adzic and Robert Chatley. 2017. Serverless computing: economic and architectural impact. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 884–889.

[4] Amazon. 2019. *Amazon AWS Lambda.* https://aws.amazon.com/lambda/

[5] Amazon. 2019. *Amazon Fargate.* https://aws.amazon.com/fargate/

[6] Amazon. 2019. *AWS Lambda Costumer Case Study.* https://aws.amazon.com/lambda/resources/customer-case-studies/

[7] Lixiang Ao et al. 2018. Sprocket: A serverless video processing framework. In *Proceedings of the ACM Symposium on Cloud Computing*. 263–274.

[8] Apache. 2019. *OpenWhisk.* https://openwhisk.apache.org/

[9] Ioana Baldini et al. 2017. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*. Springer, 1–20.

[10] Belval et al. 2020. A python module that wraps the pdftoppm utility to convert PDF to PIL Image object. https://github.com/Belval/pdf2image.

[11] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[12] François Chollet et al. 2015. Keras. https://keras.io.

[13] Google Cloud. 2019. *Cloud Functions.* https://cloud.google.com/functions/

[14] Google. 2019. *Cloud Run Concurrency Concept.* https://cloud.google.com/run/docs/about-concurrency

[15] Google. 2019. *Google Cloud Run.* https://cloud.google.com/run/

[16] Google. 2019. *What is serverless?* https://cloud.google.com/serverless-options

[17] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. [n.d.]. Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*.

[18] Joseph M Hellerstein et al. 2018. Serverless computing: One step forward, two steps back. *arXiv preprint arXiv:1812.03651* (2018).

[19] Eric Jonas et al. 2019. Cloud programming simplified: a berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383* (2019).

[20] Nima Mahmoudi, Changyuan Lin, Hamzeh Khazaei, and Marin Litoiu. 2019. Optimizing serverless computing: introducing an adaptive function placement algorithm. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. 203–213.

[21] Garrett McGrath and Paul R Brenner. 2017. Serverless computing: Design, implementation, and performance. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 405–410.

[22] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.

[23] Lucian Toader et al. 2019. Graphless: Toward serverless graph processing. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 66–73.

[24] Hao Wang, Di Niu, and Baochun Li. 2019. Distributed Machine Learning with a Serverless Architecture. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1288–1296.

[25] Jianlong Zhong and Bingsheng He. 2013. Medusa: Simplified graph processing on GPUs. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (2013), 1543–1552.