Serverless Nanopore Basecalling with AWS Lambda

Piotr Grzesik[1] and Dariusz Mrozek[1]

Department of Applied Informatics, Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
dariusz.mrozek@polsl.pl

Abstract. The serverless computing paradigm allows simplifying operations, offers highly parallel execution and high scalability without the need for manual management of underlying infrastructure. This paper aims to evaluate if recent advancements such as container support and increased computing resource limits in AWS Lambda allow it to serve as an underlying platform for running bioinformatics workflows such as basecalling of nanopore reads. For the purposes of the paper, we developed a sample workflow, where we focused on Guppy basecaller, which was tested in multiple scenarios. The results of the experiments showed that AWS Lambda is a viable platform for basecalling, which can support basecalling nanopore reads from multiple sequencing reads at the same time while keeping low infrastructure maintenance overhead. We also believe that recent improvements to AWS Lambda make it an interesting choice for a growing number of bioinformatics applications.

Keywords: nanopore sequencing, bioinformatics, serverless computing, cloud computing, metagenomics, AWS Lambda, parallel computing, basecalling

1 Introduction

In recent years, we have seen the growing popularity of the serverless computing paradigm, which was popularized by AWS Lambda offering that was made available in 2014 [17]. It allows to simplify operations, abstract away the underlying servers and reduce maintenance overhead by allowing to run and scale workflows without the need to manage the infrastructure manually, while also offering support for highly parallel execution [9]. It is also often referred to as Function-as-a-Service (FaaS) [22]. So far, it gained widespread adoption for services that require high throughput with relatively low requirements for computing power, becoming a popular choice for Web APIs and asynchronous processing [14]. However, due to the nature of most bioinformatic workflows, which often require advanced computing capabilities in terms of CPU and memory, so far, it didn't gain massive adoption, and traditional computing clusters are still the most popular architecture used to run a bioinformatics analysis. Additionally, so far, only a few computing runtimes were officially supported on major platforms

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

such as AWS Lambda. However, the recent introduction of support for Docker [10] containers [1], as well as expanding maximum memory that can be used by AWS Lambda function to 10,240 MB and up to 6 vCPU cores, makes it easier to run workloads with higher computing power requirements [4].

At the same time, in the past years, we observed the rapid growth of third-generation sequencing technologies that allow for cost-effective and quick genome sequencing. Nowadays, large DNA sequencing platforms offer natural transmission bridge to Cloud environments to facilitate data storage, processing, and analysis. For example, Illumina, which delivers technological solutions for genetic and genomic data analyses, promotes efficiency by streaming the sequencing data directly to the AWS cloud with their BaseSpace Sequence Hub tool [8]. This paper aims to evaluate the feasibility, performance, and cost-effectiveness of performing computations directly on AWS Lambda to see if it can be used to carry out operations such as basecalling [23] of nanopore reads. We experimentally check whether Cloud serverless computing can serve to computationally-demanding tasks related to modern DNA sequencing technologies. The paper is organized as follows. In section 2, we review the related works in the area. In section 3, we describe the testing environment along with bioinformatic tools considered as a part of our experiments. Section 4 contains a description of the testing methodology along with performance experiments that we carried out for selected scenarios. Finally, section 5 summarizes the results and concludes our findings.

2 Related Works

In the literature, there is only a little research concerning running bioinformatics workflows with the use of serverless computing platforms. In the paper, [21], Niu et al. describe the proof of concept example of running all-against-all pairwise comparison among 20,000 human protein sequences using Striped Smith-Waterman implementation. According to their findings, it can be accomplished in about 2 minutes for a cost of less than one dollar. Authors conclude that the use of serverless cloud computing can be leveraged to dramatically speed up the execution time of certain tasks at a low cost. They also suggest that in a similar approach, serverless computing can be used for tasks such as sequence alignment, protein-folding, or deep learning.

Malawski et al. [20] focused on evaluating AWS Lambda, Google Cloud Functions, and HyperFlow in the context of executing scientific workflows in a serverless manner. The authors developed a prototype workflow executor functions based on the aforementioned technologies. They managed to deploy and run the Montage astronomy workflow. They suggest that AWS Lambda infrastructure offers good scalability. However, not all workflows are suitable to serverless computing architecture, and it might be worth considering a hybrid approach that combines serverless and traditional architectures.

In their research [12], Burkat et al. evaluated serverless infrastructures in the model of Container-as-a-Service as a platform for running various scientific

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

workflows. Authors especially focused on two offerings, AWS Fargate and Google Cloud Run. For evaluation purposes, they extended the HyperFlow engine to support execution on these platforms. During experiments, the authors run four scientific workflows: Ellipsoids, Vina, KINC, and Soy-KB, which were selected due to different resource requirements. The authors conclude the paper with a claim that serverless containers can be successfully used for scientific workflows.

Joyner et al., in their article [18], propose Ripple, a dedicated programming framework that aims to allow programs that were designed for single-machine execution to take advantage of parallelism offered by serverless computing. Ripple offers an interface that allows users to express workflows of a broad spectrum of applications, such as machine learning, genomics, or proteomics. Authors port three workflows: SpaceNet building border identification, proteomics with Tide and Percolator, and DNA compression with METHCOMP, showing that using Ripple can offer significant performance benefits versus traditional cloud deployments.

John et al. [15] proposed a solution called SWEEP, which is a workflow management system that takes advantage of the serverless execution model. It is cloud-agnostic and allows users to define, run and monitor scientific workflows. The authors evaluated their system for two cases, variant calling, and satellite imagery processing. They also list the elasticity of serverless computing and lack of overhead related to cluster management in traditional computing as significant benefits of the serverless approach.

In his research [19], Lee et al. proposed a DNAVisualisation.org, a fully serverless web tool dedicated to DNA sequence visualizations. With this tool, the authors wanted to demonstrate the applicability of serverless computing in the field of molecular biology in addition to allowing the ability to visualize DNA sequences in a cost-effective manner quickly. They also suggest that while not all applications are a great fit for serverless computing, some of them might benefit from decreased costs, reduced development complexity, which can be a significant advantage over the traditional architectures.

Crespo-Cepeda et al., in their work [13], consider challenges and opportunities for AWS Lambda services in the context of bioinformatics. Their paper proposes an architecture for running CloudDmetMiner, focusing on simplifying the workflow as much as possible, based on AWS Lambda and AWS S3. Authors conclude successful experiments suggesting that serverless computing can ease the code execution by reducing the time it takes to manage and provision infrastructure manually.

In their paper [16], Jonas et al. suggest that stateless functions offered by serverless computing can allow for more straightforward parallel computation without complex management of clusters and configuration tools. Authors also introduce and evaluate a tool called PyWren, which is a framework that allows mapping selected calculations across multiple concurrently running AWS Lambda functions. They conclude the paper with a suggestion that stateless functions are a great fit for data processing for future applications.

Based on the above, we can conclude that there is a lot of interest in evaluat-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

ing serverless computing architectures for various scientific workflows. However, just a few of them consider bioinformatics workflows, and not a single one of them considered performing basecalling in such an environment. This paper aims to expand knowledge in that area by evaluating basecalling workflow when using AWS Lambda and serverless computing model.

3 Testing workflow and environment

For the evaluation, we selected the workflow where we split Nanopore MinION FAST5 files with nanopore reads (raw signal data) into batches. These batches are later processed separately by independent Lambda functions, enabling stateless parallelism, as the functions do not have to communicate with each other during processing. During executions, Lambda calls the basecalling function, which processes the files from an AWS S3 bucket. After processing its batch, each function saves the results in a separate folder in AWS S3 bucket. The data stored in that bucket can be used for further processing. Fig. 1 presents the considered workflow.

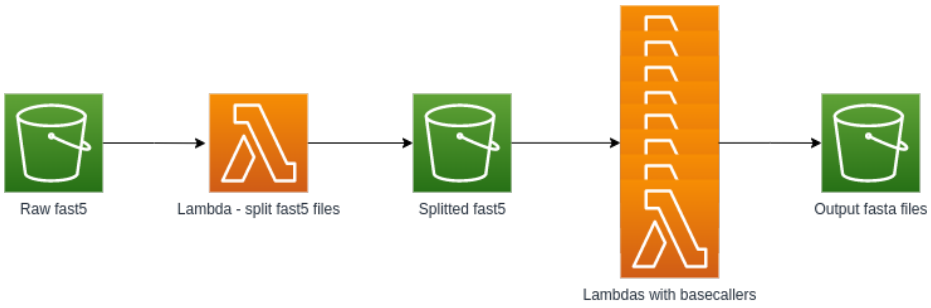

Fig. 1: Diagram of the considered workflow.

Each Lambda function used a Docker container runtime, taking advantage of a custom Docker image based on Ubuntu 16.04 operating system with a Python 3.6 wrapper script that executes a binary with basecalling software.

For basecalling, we considered several tools. The first one was Deepnano-blitz [11], an open-source basecaller, developed by Boža, V., based on bidirectional recurrent neural networks. It was very promising, as its implementation is optimized to take advantage of Intel AVX2 instruction set [7], which is also supported by Lambda environment [6]. Unfortunately, after preliminary testing in the Lambda environment, it turned out that Lambda's lack of support for shared memory between separate processes makes it impossible to run Deepnano-blitz in its current form. We consider adjusting Deepnano-blitz in the future for the AWS Lambda environment. Next considered basecaller was Guppy [23], which is a closed-source, state-of-the-art basecaller developed by Oxford Nanopore Technologies. It has support for multiple basecalling models – fast and high accuracy.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

We also considered two alternative open-source basecallers, Bonito [5], and Causcalcall [24]. However, after preliminary testing, they turned out to offer much lower performance in comparison to Guppy, so we decided to focus on Guppy in our experiments and potentially considering Deepnano-blitz in the next research.

4 Performance experiments

During experiments, we decided to measure the basecalling capabilities of the AWS Lambda computing environment with a different maximum available memory setting, which also translates to the number of virtual CPU cores that are available for the function. According to official documentation[**?**], available CPU power scales proportionally with memory with 1 full vCPU core at 1769 MB of memory to the maximum of 6 vCPU cores for 10240 MB of allocated memory. For each run, we recorded the number of samples processed per second, as well as the ratio of samples per second to available memory, to assess which setting is the most effective from a pricing perspective, as AWS Lambda is billed per GB/s [3]. The tests were run with Guppy in the 4.0.14 version, using both fast and high accuracy models for R9.4.1 chemistry. To carry out experiments, we used a subset of the Klebsiella pneumoniae reads dataset that was used for benchmarking in [23].

Based on results obtained during the experiments, we observed that when using Guppy fast model, the number of samples processed per second scales linearly from 256 to about 6,144 MBs of RAM, where after crossing that threshold, we see smaller improvements, as can be seen on Fig. 2. It is especially visible in Fig. 3 that after crossing 6,144 MB, we see a lower ratio of samples per second per one MB of memory. For Guppy fast model, we observed the highest ratio of samples per second per MB of memory for 2,048 MBs, with a ratio of 86.76, which means that it's the most efficient setting from the cost-effectiveness standpoint.

When considering Guppy with high accuracy models, we observed different patterns than for fast model. Except for a scenario with 6,144 MB of memory, we observe that the ratio of samples per second per MB of RAM is growing along with assigned memory, being the highest for 10,240 MBs of memory with the value of 3.74, which can be seen in Fig. 5. We also observed much lower values for samples processed per second in general compared to the fast model with a peak of 38,293 samples processed per second for maximum memory of 10,240 MBs, as presented in Fig. 4 (versus 622,154 in the fast mode for the same amount of memory). It is also important to note that it was impossible to run a high accuracy model with only 256 MB of memory, which was previously possible with the fast model. For all tested scenarios, we did not observe instances of failing tasks other than expected initial invocation failure related to mandatory image optimization step performed by AWS Lambda right after deployment of new version of the container image.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

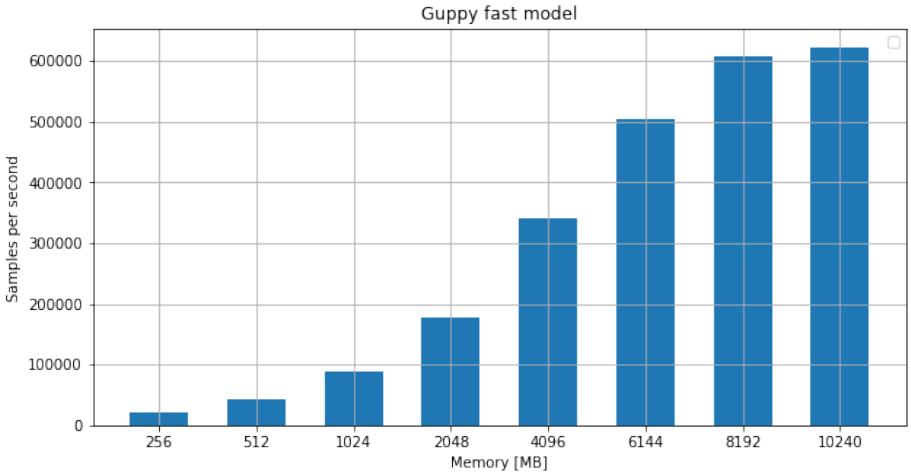

Fig. 2: Samples processed per second for Guppy fast model.

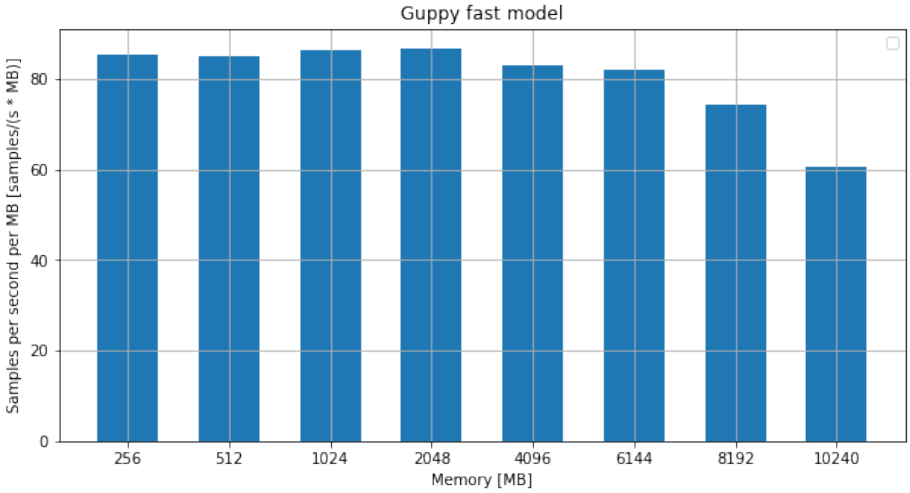

Fig. 3: Samples per second per MB of memory for Guppy fast model.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

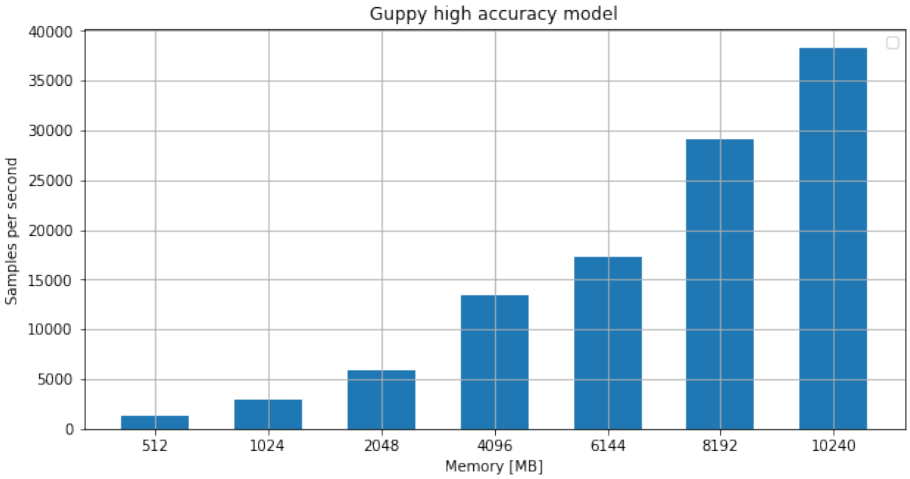

Fig. 4: Samples processed per second for Guppy high accuracy model.

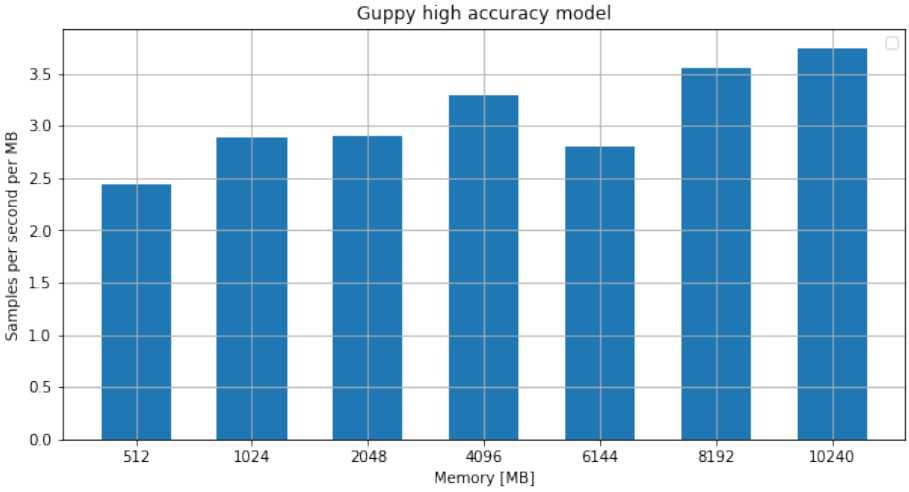

Fig. 5: Samples per second per MB of memory for Guppy high accuracy model.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

5 Results Summary and Concluding Remarks

Considering successful experiments and results presented in the previous chapter, we conclude that thanks to recent advancements in AWS Lambda offering, it is now possible to run basecalling workflows in a serverless manner. For example, given the fact that the theoretical maximum of MinION Nanopore is around 2,300,000 signals per second (512 pores with around 4,500 signals read per second per pore), with real-world scenarios resulting in less than 2,000,000 signals per second, which means that 3 to 4 function instances would be able to keep up with processing that data in near real-time. Given the ability of AWS Lambda to scale up to hundreds of thousands of concurrently running functions [2], we gain the capability to quickly basecall data from multiple sequencing experiments while keeping the infrastructure maintenance overhead as low as possible. We expect even better results with CPU-optimized basecallers such as Deepnano-blitz [11]. We believe that serverless computing architectures will continue to gain more features and enable even more bioinformatic workflows in the future and that there is still a lot of room for improvement and development in that area.

Acknowledgments

The research was supported by the Polish Ministry of Science and Higher Education as a part of the CyPhiS program at the Silesian University of Technology, Gliwice, Poland (Contract No. POWR.03.02.00-00-I007/17-00) and by Statutory Research funds of Department of Applied Informatics, Silesian University of Technology, Gliwice, Poland (grant No BK-221/RAu7/2021).

References

1. AWS Lambda container image support (accessed on February 5th, 2021), https://aws.amazon.com/blogs/aws/new-for-aws-lambda-container-image-support/
2. AWS Lambda limits (accessed on February 5th, 2021), https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html
3. AWS Lambda pricing (accessed on February 5th, 2021), https://aws.amazon.com/lambda/pricing/
4. AWS Lambda support for 10240 MB and 6 vCPU cores (accessed on February 5th, 2021), https://aws.amazon.com/about-aws/whats-new/2020/12/aws-lambda-supports-10gb-memory-6-vcpu-cores-lambda-functions/
5. Bonito basecaller repository on Github (accessed on February 5th, 2021, https://github.com/nanoporetech/bonito
6. Creating faster AWS Lambda functions with AVX2 (accessed on February 5th, 2021), https://aws.amazon.com/blogs/compute/creating-faster-aws-lambda-functions-with-avx2/
7. How Intel® Advanced Vector Extensions 2 improves performance on server applications (accessed on February 5th, 2021), https://software.intel.com/content/www/us/en/develop/articles/how-intel-avx2-improves-performance-on-server-applications.html

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

8. Augustyn, D.R., Wyciślik, L., Mrozek, D.: Perspectives of using cloud computing in integrative analysis of multi-omics data. Briefings in Functional Genomics (in press), 1–23 (2021)
9. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., Suter, P.: Serverless Computing: Current Trends and Open Problems, pp. 1–20. Springer Singapore, Singapore (2017), https://doi.org/10.1007/978-981-10-5026-8_1
10. Bashari Rad, B., Bhatti, H., Ahmadi, M.: An introduction to Docker and analysis of its performance. IJCSNS International Journal of Computer Science and Network Security Vol. 17, No. 3, 228–235 (2017)
11. Boža, V., Perešíni, P., Brejová, B., Vinař, T.: Deepnano-blitz: A fast base caller for minion nanopore sequencers. Bioinformatics (Oxford, England) 36, 4191–4192 (05 2020)
12. Burkat, K., Pawlik, M., Balis, B., Malawski, M., Vahi, K., Rynge, M., da Silva, R.F., Deelman, E.: Serverless containers - rising viable approach to scientific workflows. ArXiv abs/2010.11320 (2020)
13. Crespo-Cepeda, R., Agapito, G., Vazquez-Poletti, J.L., Cannataro, M.: Challenges and opportunities of amazon serverless lambda services in bioinformatics. In: Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. p. 663–668. BCB '19, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3307339.3343462
14. Eismann, S., Scheuner, J., van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., Abad, C.L., Iosup, A.: A review of serverless use cases and their characteristics. arXiv 2008.11110 (2021)
15. John, A., Ausmees, K., Muenzen, K., Kuhn, C., Tan, A.: Sweep: Accelerating scientific research through scalable serverless workflows. In: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. p. 43–50. UCC '19 Companion, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3368235.3368839
16. Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., Recht, B.: Occupy the cloud: Distributed computing for the 99Cloud Computing. p. 445–451. SoCC '17, Association for Computing Machinery, New York, NY, USA (2017), https://doi.org/10.1145/3127479.3128601
17. Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N.J., Gonzalez, J.E., Popa, R.A., Stoica, I., Patterson, D.A.: Cloud programming simplified: A berkeley view on serverless computing. CoRR abs/1902.03383 (2019), http://arxiv.org/abs/1902.03383
18. Joyner, S., MacCoss, M., Delimitrou, C., Weatherspoon, H.: Ripple: A practical declarative programming framework for serverless compute. CoRR abs/2001.00222 (2020), http://arxiv.org/abs/2001.00222
19. Lee, B., Timony, M., Ruiz, P.: DNAvisualization.org: a serverless web tool for DNA sequence visualization. Nucleic acids research 47 (06 2019)
20. Malawski, M., Gajek, A., Zima, A., Balis, B., Figiela, K.: Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions. Future Generation Computer Systems 110, 502–514 (2020), https://www.sciencedirect.com/science/article/pii/S0167739X1730047X
21. Niu, X., Kumanov, D., Hung, L.H., Lloyd, W., Yeung, K.Y.: Leveraging serverless computing to improve performance for sequence comparison. In: Proceedings of the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44

10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. p. 683–687. BCB '19, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3307339.3343465
22. Scheuner, J., Leitner, P.: Function-as-a-service performance evaluation: A multivocal literature review. Journal of Systems and Software 170, 110708 (2020), https://www.sciencedirect.com/science/article/pii/S0164121220301527
23. Wick, R.R., Judd, L.M., Holt, K.E.: Performance of neural network basecalling tools for oxford nanopore sequencing. Genome Biology 20(1), 129 (Jun 2019), https://doi.org/10.1186/s13059-019-1727-y
24. Zeng, J., Cai, H., Peng, H., Wang, H., Zhang, Y., Akutsu, T.: Causalcall: Nanopore basecalling using a temporal convolutional network. Frontiers in Genetics 10, 1332 (2020), https://www.frontiersin.org/article/10.3389/fgene.2019.01332

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_44