Simulation-as-a-Service with Serverless Computing

Kyriakos Kritikos*
*ICS-FORTH, Crete, Greece
kritikos@ics.forth.gr

Paweł Skrzypek†
*AI Investments, Warsaw, Poland
pawel.skrzypek@aiinvestments.pl

***Abstract*—Simulations are the greatest means for evaluating systems and producing knowledge related to their optimal configuration for production. Simulation systems support the execution of simulations. These can be installed and executed internally to an organisation or can be offered as a service in the cloud. Current simulation-as-a-service (SimaaS) offerings rely on VM or container-based deployments which lead to additional costs due to the charging in an hourly basis. Further, such offerings cannot be easily adapted at runtime to still be able to sustain their promised service level. To resolve these issues, this paper proposes a novel SimaaS architecture and solution which exploits the serverless computing paradigm for reducing the simulation cost based on the actual usage of resources as well as accelerating the simulation time through the limitless, parallelised invocation of functions. Further, this solution relies on the MELODIC/Functionizer multi-cloud platform which enables adapting the simulation execution at runtime in order to sustain the right service level according to the user requirements and preferences. For the validation of our solution, a real business application provided by AI Investments has been used. It aims to optimise investment portfolio using the most advanced AI-based methods and requires heavy computational power to accomplish the respective tasks.**

***Index Terms*—simulation, serverless, cloud, deployment, asynchronous communication, massive parallelisation, task computing, queues**

I. Introduction

Systems can be evaluated before entering the production by utilising simulations, which enable to assess them under varied conditions. The outcome of these simulations is an important knowledge that can assist in improving these systems and their configuration. The simulations are usually executed via simulation tools and frameworks. Such tools can be deployed in different ways. On-premise in case an organisation owns and has capacity to deploy and run such tools. They can also be deployed in the cloud, which not only offers the right, powerful resources to execute them but also promises to scale them on demand through the supply of additional resources.

Due to the advantages that cloud computing offers, a certain line of work also considered realising the concept of Simulation-as-a-Service (SimaaS) [1]–[5]. This can be a SaaS offered to organisations in order to run their simulations without caring about resource management details for them. However, the current realisations of this concept suffer from two main drawbacks. First, they are not always capable of automatically exploiting additional resources to run simulations, when the respective need arises. This means that either simulations are run over a static resource set or this set needs to be manually enhanced with the extra resources needed. Second, such realisations usually exploit either virtual machines (VMs) or containers as resource units. However, such units are usually charged per hour and not based on the amount of their usage leading to unnecessary, extra costs. Further, their scaling is limited due to the overall resource limit posed by providers per user account while it can lead to the occupation of extra resources not actually needed for the computation (in case of VM resource units).

Fortunately, cloud technologies evolve over time and new ones come into play, giving rise to new resource management opportunities and computing types. This is essentially true also for serverless computing [6]. In this computing kind, users can deploy and execute small pieces of software called functions while they do not care about their scaling as this is handled transparently by the provider operating the serverless platform. This computing kind comes with several advantages [6], which apart from zero administration also include flexible cost models based on the actual function usage, leading to cost-efficient solutions, as well as capabilities of infinite scaling with much higher function limits with respect to other types of resources. In this respect, service computing currently takes a momentum and various applications have been migrated or are currently under migration towards this new computing form.

Based on this latest cloud development, this paper proposes to take SimaaS to the next level. In particular, it proposes a novel SimaaS architecture which is centred around the notion of functions. This architecture is quite flexible as it promises to use message queues for the asynchronous orchestration of simulation execution functions while it also caters for two adaptation forms: (a) scaling the message queue to handle the additional messaging load; (b) increasing the number of function deployments to handle the extra simulation load. To support both adaptation forms, the Functionizer platform is exploited. This platform enables to deploy applications across multiple clouds while it relies on the use of utility functions for continuously finding the best deployment plans for these applications. Adaptation is realised through the application's global reconfiguration which does include the identification of the right number of instances per application component.

Based on the above analysis, our proposition surpasses current SimaaS offerings for the following reasons. First, by adopting serverless computing, it reduces the simulation cost as the simulation execution is charged only based on its actual resource usage time. Second, it enables to massively execute thousands of simulation execution functions which can lead to reduced overall simulation time. Third, through its global reconfiguration ability, it can enable to adapt the SimaaS on demand to respect the user requirements. This can allow, for instance, to scale the SimaaS when it is estimated through monitoring that it will not finish on time.

Our solution was evaluated according to a real business application, dedicated to investment portfolio optimisation, against traditional cloud computing resource management set ups. The evaluation results highlight a significantly improved simulation cost and time, highlighting the suitability and great impact potentiality of our novel solution.

The rest of the paper is structured as follows. Section II reviews the related work. Section III analyses our new SimaaS proposition. Section IV showcases and discusses our solution's evaluation results. Finally, the last section concludes the paper and draws directions for further research.

II. Related Work

Related work takes two main forms: (a) current SimaaS offerings; (b) frameworks which support data-intensive cloud application execution through serverless computing. Each of these forms is analysed in separate sub-sections.

A. SimaaS Approaches

The concept of Models and Simulation as a Service (MaSS) has been proposed in [1]. The main rationale is that simulation computation models can be shared, configured and executed through a scientific research platform like the Galaxy one. To demonstrate this, the authors have developed a tool suite which supports the simulation of the high-spatial resolution model of the cardiac Ca^{2+} spark.

The same concept is adopted in [2] with a focus on Petri-Net models. To this end, the authors have integrated the simulation tool Renew as a service and were able to simulate a traffic control system as a Petri-Net model which can adjust its parameters through reflective simulation.

The concept of SimaaS is promoted in [3] as a means to reduce costs and as well as scale on-demand respective simulations in the manufacturing domain. Based on this concept, they have realised the Polymer Portal, which is a SimaaS platform that integrates the access to modelling, simulation and training services. Interesting features of this platform include: the capability to deploy simulations over both VM and HPC resource units as well as the supply of an AAA service to secure the access to its facilities.

[4] proposes a novel layered cloud architecture for Modelling and Simulation as a Service as well as a middleware supporting this architecture. This middleware is able to support the deployment of models and simulations as services in scalable infrastructures with hierarchical resource provisioning as well as the integration of experimental frameworks.

A cloud-based simulation service is proposed in [5] which supports container-based deployment of simulation components instead of a VM-based deployment, adopted in the rest of the above solutions, which incurs substantial overhead in on-demand resource provisioning.

As it can be seen from the above analysis, no simulation framework or SimaaS has adopted serverless computing. As such, it is not able to feature the advantages of our proposition, which include less cost, less simulation time as well as a better and adaptive provisioning of the execution simulation.

B. Data-Intensive Application Frameworks

While serverless computing has been originally deemed as appropriate for handling rare or highly-bursty workloads, it has been recently advocated that it can also be used for handling data-intensive workloads. The main rationale for the latter relies on the capability to concurrently launch thousands of functions in order to process such workloads, which can really enable to better scale the data-intensive application while reducing its overall execution time.

In this respect, three main data-intensive application execution frameworks have been proposed. All of these frameworks seem to rely on Spark, which is considered as a state-of-the-art data-intensive application execution engine. In [7], Spark is extended with a new form of executor able to execute functions on-demand based on the current way Spark handles the separation of the data-intensive workload. The main innovation lies on changing the way the data shuffling phase is conducted. In particular, instead of using a distributed file system, the approach in [7] proposes the use of message queues, which allow the asynchronous delivery of the work to executors that can take care of it once they become available.

This critical data shuffling phase seems to be also covered by the other two frameworks. The framework in [8] uses S3 for realising this phase but, unfortunately, it can suffer from performance problems due to the throttling behaviour of S3 for high workloads. Further, S3 does not offer the best possible throughput performance. This drawback has been picked up by the framework in [9] which proposes the use of a three-level emphemeral storage architecture which is configured and dynamically provisioned according to the user requirements. The use of this storage type is justified by the fact that the internal data generated during the data-intensive computations are needed only temporarily for computation purposes and can be usually deleted once consumed by the next task in computation order.

Our proposition is similar to the above with the sole exception that it does not require the use of Spark libraries, which consume considerable memory footprint. Compared to [7] and [8], it does not depend on just one message queue or storage service but is able to function in a multi-cloud way, thus avoiding provider lock-in. Compared to [9], it is less costlier as it does not require the use of multiple levels for the storage of temporary information.

III. Proposed approach

For novel machine learning applications the crucial element is providing data for the training process. Data could be provided based on the real usage; however, it is also very important to cover a wider range of all possible cases [10]. Also, for some cases these data are not available based on the real usage and thus need to be derived from simulation. Usually, for complex problems the simulations require to be performed in thousands or even hundred thousands of iterations to cover the complete range of possible cases. Further, optimisation methods, like evolutionary algorithms [11], require to execute multiple simulations before finding the best solution. To this end, for many practical applications, the ability to execute a great number of simulations is a critical requirement.

For a typical deployment setup, based on HPC or servers with multiple cores, the cost of performing simulations is significant. In addition, HPC setups suffer from higher turnaround time. As both the turnaround and execution time for a simulation are significant, Cloud Computing accelerates the accessing to resources while allows to parallelise the computation and thus shorten the simulation time through the use of multiple VMs with many cores. However, the cost still remains an issue which is quite often blocking for many start-ups and SMEs companies. As such, such an issue does not allow them to effectively compete with large companies having higher IT budgets. Besides, there exist extra opportunities for simulation time optimisation which are restrained through current limitations in number of VMs that can be simultaneously reserved per cloud provider.

Thus, the ability to lower the cost by also extremely parallelising the execution to reduce the simulation time are crucial assets for the above type of applications and companies. The introduction of new Cloud Computing models, like the serverless[1] one, allows to prepare a more optimal solution to this problem which is both more cost-effective while bypasses the resource limitation problems mentioned above. To this end, in order to resolve the aforementioned issues, we rely on the serverless computing model and propose a novel approach towards building a SimaaS solution featuring asynchronous communication using a queue messaging broker, the aforementioned issues can be resolved.

Our approach fully supports multi-cloud deployments and can be run on any combination of cloud providers supporting both microservice and serverless components. It is based on requirements defined for a generic, multi-cloud, hybrid application provisioning method [12]. Due to this, it is possible to avoid vendor lock-in and optimise the provisioning cost by grabbing better opportunities through the use of multi-cloud environments. Such an approach also increases the solution's reliability and availability, as it is not limited to one cloud provider.

To support the multi-cloud hybrid application provisioning method, the MELODIC[2] platform with the Functionizer extension[3] [13] has been utilised. This enhanced platform follows model-driven engineering to automate the various activities in the multi-cloud, hybrid application management lifecycle. A cornerstone to this platform is the CAMEL (Cloud Application Modelling and Executing Language) [14], [15] multi-domain-specific-language (multi-DSL) which enables the rich specification of a multi-cloud application covering all related aspects to the multi-cloud application lifecycle. Another core feature of the enhanced MELODIC platform is its ability to dynamically and continuously provision an application within multi-cloud environments in the most optimal way through the use of utility functions [16], multi-level monitoring [17] and global reconfiguration methods. It also features a great repertoire in terms of the diversity in component deployment and resource provisioning, including support for provisioning virtual machines and Docker[4] containers as well as for deploying big data applications based on Spark[5] on top of cloud computing resource clusters plus serverless components. The following cloud providers are currently supported: Amazon AWS[6], Microsoft Azure[7], Google Cloud Platform[8], Oktawave[9], ProfitBricks[10] and any OpenStack[11]-based cloud provider.

Using the above platform with our approach allows to achieve the most optimal SimaaS solution in terms of resource cost and computation time. The proposed approach, due to its distributed and asynchronous architecture, allows for almost unlimited scalability. Thanks to multi-cloud deployment, it is possible to run multiple thousands of simulations in parallel. It also allows to fully utilise available resources in given time, to run as many instances as possible within a given cloud provider. Thanks to that the simulations execution could be massively paralellized and, thus, the total execution time of all simulations could be shortened. The usage of very lightweight component types - serverless - allows to minimise costs significantly. The above reasons demonstrate that our proposed approach is novel and features a unique way of executing simulations, which allows start-ups and SMEs to effectively compete with large enterprises and organisations.

The key elements of our approach are the following:

1) *GUI* module – The control module for operating a SimaaS application. It features a graphical UI via which user simulation requests can be formulated and issued to the *Simulation Orchestrator* as well simulation results can be presented. Each simulation request is transformed into a CAMEL model which is then fed to the MELODIC platform in order to optimally deploy a proper instance of the SimaaS application.
2) *Simulation Orchestrator* (SimOrch) – component responsible for orchestrating the execution of SimaaS instances and gathering back the results of simulations. A SimmaS instance execution orchestration involves creating appropriate work items, by dividing the simulation work, and submitting them to the *Multi-Cloud Message Queue Broker*.
3) *Simulation Launcher* (SimLauncher) – component responsible for starting the *Simulation Executor* based on a given simulation work item.
4) *Simulation Executor* (SimExecutor) – serverless component which executes the simulation work mapping to an

[1]https://serverless-stack.com/chapters/what-is-serverless.html
[2]www.melodic.cloud
[3]https://melodic.cloud/functionizer/

[4]https://www.docker.com/
[5]https://spark.apache.org/
[6]https://aws.amazon.com/
[7]https://azure.microsoft.com/pl-pl/
[8]https://cloud.google.com/
[9]https://www.oktawave.com/pl
[10]https://www.profitbricks.com/en-us/products/cloud-computing-pricing/
[11]https://www.openstack.org/

item. This component can be deployed in any serverless platform where this is determined based on the user requirements (part of the user request).
5) *Multi-Cloud Message Queue Broker* (MCBroker) – The broker configures two queues, one for sending execution requests / work items (*requestsQueue*) and another for gathering the results of the simulations (*resultsQueue*). MCBroker could be distributed through many cloud providers creating an effective multi-cloud environment for message passing.

Our approach's overall architecture is presented in Fig. 1. All components apart from the Melodic platform and the GUI are modelled in CAMEL and deployed through the Melodic platform.

The sequence action flow in our approach, which is depicted in Fig. 2, is as follows:

1) The *GUI* collects the simulation request from the user. After that prepares a CAMEL model for the MELODIC platform to optimally deploy an instance of the SimaaS solution. Final step is passing the simulation request to *SimOrch* and presenting the results after the simulation is executed.
2) *SimOrch* prepares the work items for the requested simulation, which needs to be executed according to a specific amount of times. Each work item needs to be dispatched to one *Simulation Executor*. Each work item includes a set of parameters required for executing the simulation.
3) Each work item is sent to the *requestsQueue*.
4) Work items are collected from *requestsQueue* by *Simulation Launcher*. For each collected work item, one instance of the *Simulation Executor* serverless component is invoked.
5) The *Simulation Executor* executes the simulation for a given set of parameters and returns the results to the *resultsQueue*.
6) *SimOrch* gathers the results of all conducted simulations into a coherent whole which is then supplied back to the GUI.

The listed components have been implemented using the following technologies and component types:

1) *GUI* – implemented in Java[12] and Angular[13]; deployed in a Docker container.
2) *Simulation Orchestrator* – implemented in Java, as a Spring application[14]. It is deployed in a Docker container.
3) *Simulation Launcher* – implemented also in Java as a Spring application. For general purposes, it is deployed in a Docker container. For specific deployments, like on AWS, to additional limit the number of resources used, it could be deployed using the native feature of AWS SQS[15] to trigger serverless components on each incoming SQS message.
4) *Simulation Executor* – implemented in Java and deployed as a serverless component.
5) *Multi-Cloud Message Queue Broker* – implemented using RabbitMQ[16] framework with JMS[17] extension plugin. For specific deployments on AWS, it could be replaced by native AWS's SQS messaging service. It is also considered to replace RabbitMQ by Kafka[18] in case of quite demanding, in terms of number of simulations, deployments.

[12]https://www.oracle.com/java/
[13]https://angular.io/
[14]https://spring.io/

IV. Comparative Evaluation

A. Use Case Presentation

The validation of our approach has been performed using a real business application called AI Investments[19]. It is a solution using AI for investment portfolio optimisation. This solution is offered by the AI Investments start-up, aiming at creating a complete trading solution, including a diversified way of signalling transactions, determining market conditions and managing exposition, with the goal to develop a software platform for advanced investments in the global markets. The key assumption for the AI Investments platform is diversification. Thus, the platform should be able to operate on over 200 markets (stocks, bonds, commodities, currencies) using over 10 investment strategies on 4 different time intervals (hourly, 4 hours, daily, weekly). This can create a total number of 8000 unique investment strategies (200 markets times 10 investment strategies times 4 time intervals).

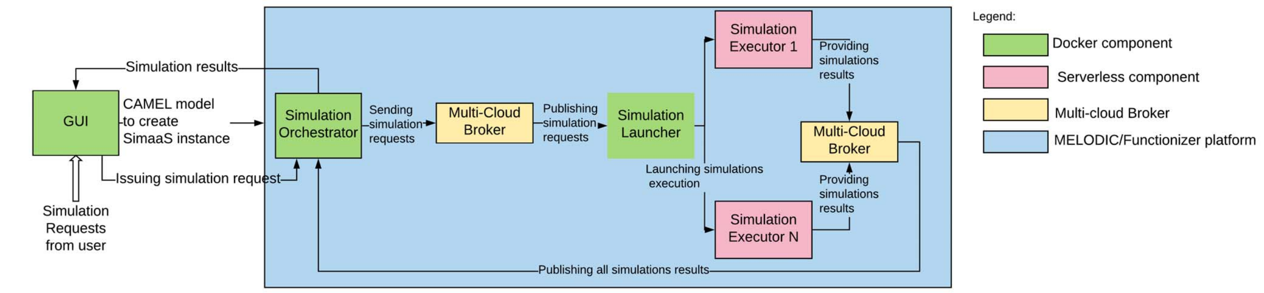
One of the critical issues for the AI Investments platform is optimisation of the investment strategies parameters. It is currently conducted by using evolutionary algorithms [18]. The use of evolutionary algorithms requires the execution of many simulations which simulate the results of the investment strategies with given parameters on the extensive time period. Based on the simulated results, a new population is created and the process is repeated. This approach requires to run multiple simulations to find the most optimal solution.

For the purposes of AI Investments, the most typical parameters for optimisation are as follows:

1) The size of population per each generation is 5000.
2) The average number of iterations is 20000.
3) The average execution time per single simulation is 0.7 seconds.

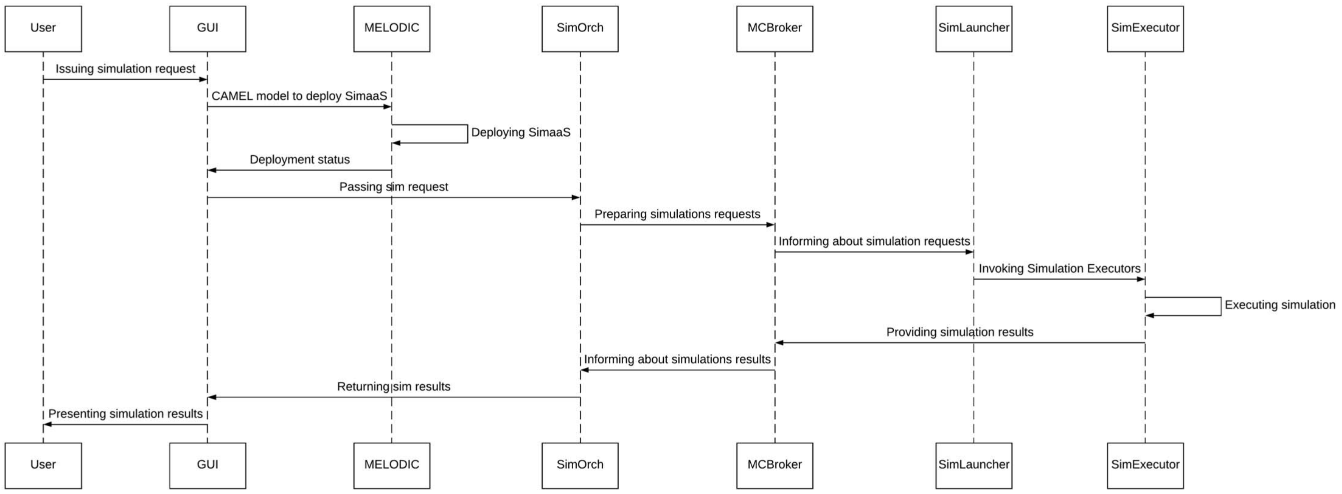
For each member of the population (set of investing strategies parameters) the simulation on the given historic period is executed. The results of the strategy are then calculated (return of the investments, maximal draw down and Sharpe ratio [19]). Based on the calculated results of the all members of current generation, the new generation is created by the evolutionary

[15]https://aws.amazon.com/sqs/
[16]https://www.rabbitmq.com/
[17]https://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html
[18]https://kafka.apache.org/
[19]https://www.aiinvestments.pl/



Simulation results
GUI
CAMEL model to create SimaaS instance
Issuing simulation request
Simulation Requests from user
Simulation Orchestrator
Sending simulation requests
Multi-Cloud Broker
Publishing simulation requests
Simulation Launcher
Launching simulations execution
Simulation Executor 1
Simulation Executor N
Providing simulations results
Multi-Cloud Broker
Providing simulations results
Publishing all simulations results
Legend:
Docker component
Serverless component
Multi-cloud Broker
MELODIC/Functionizer platform

Fig. 1. Serverless SimaaS architecture



User
GUI
MELODIC
SimOrch
MCBroker
SimLauncher
SimExecutor
Issuing simulation request
CAMEL model to deploy SimaaS
Deploying SimaaS
Deployment status
Passing sim request
Preparing simulations requests
Informing about simulation requests
Invoking Simulation Executors
Executing simulation
Providing simulation results
Informing about simulations results
Returning sim results
Presenting simulation results

Fig. 2. Serverless SimaaS sequence flow

algorithm and the process repeats. The total number of the simulations to execute is 100 000 000 per each trading strategy on a given market.

B. Evaluation

This sub-section presents results, in terms of time and cost of the execution of the proposed approach based on a certain configuration scenario according to the aforementioned use case. It covers also two other configuration scenarios utilised for comparison reasons with our solution mapping to the usage of traditional cloud resources (i.e., VMs) for the simulation execution. One of the latter scenarios is a baseline one with a single core simulation execution. The second scenario is a real example of running simulations on a powerful VM with 128 cores in the cloud.

Based on the application and setup presented in the previous sub-section, the total time of running the simulations using the presented approach with up to 1000 serverless components running in parallel on AWS takes on average approximately 20 hours[20]. The execution cost of that type of simulation is on average 450 usd (based on real measurements). The estimated cost of using serverless components on AWS gives very similar results[21]. The estimated execution time for the given parameters (size of population: 5000, number of iterations:20000 and average simulation time 0.7 second) is 19.44 hours. It shows that additional overheads, like the communication one, have low impact on the total computation time.

For comparison reasons, with respect to the same parameters of evolutionary algorithm and simulation time, the execution of all simulations on the single-core VM is estimated to take 810 days (5000 members of population times 20000 iterations times 0,7 seconds of execution simulation). It will cost 1866 usd, based on the cost of the smallest m5.large VM offer on AWS, which costs 0,096 usd per hour.

On the other hand, the more realistic scenario with the 128-core VM took approximately 6.3 days to finish. It was charged with, based on the current Amazon (AWS) pricing of the x1.32xlarge VM with 128 cores (the only VM offer with this number of cores), approximately 2500 usd. Summary of the estimated results for the (3) setups is presented in Table I.

[20]For the AWS deployment the *Simulation Launcher* component is replaced by the native AWS SQS serverless component launching.

[21]https://dashbird.io/lambda-cost-calculator/

TABLE I
RESULTS FOR THE THREE SETUPS

Scenario	Execution Cost (USD)	Execution Time (Hours)	Num of Parallel Executions
1-core server	1866	19440	1
128-core server	2500	151	128
Serverless solution with 1000 instances	450	19.4	1000

For comparison reasons, we also tried to deploy a similar SimaaS framework on Spark with serverless support using the version of Spark with serverless provided by Databricks[22]. Due to issues of configuration, unstable behaviour, the results were much worse than for a solution without Spark. Also, for this type of usage, the map-reduce operations, which are the core of the big data frameworks, caused additional overhead.

V. Conclusions and contributions

This paper has proposed a novel form of a simulation-as-a-service (SimaaS), which is able to exploit the main advantages of serverless computing in order to finish on time its delegated simulations. Compared to other SimaaS offerings, it is less costlier, it accelerates the simulation execution time while it is adaptive to the current workload and context. At its backbone, it exploits the MELODIC/Functionizer multi-cloud application management platform which is able to deploy and dynamically provision applications that comprise both micro-service, data-intensive and serverless components.

As proven in Section IV-A, the proposed novel form of SimaaS is very beneficial for described AI-based portfolio optimisation application. It is worth to note that our solution is applicable for a more general type of applications which use simulations at large scale. This solution is also implemented by Optimali[23], the company which optimises operations of vehicle sharing operators. It is simulating a complete environment (e.g., city, municipal area) to find the most optimal way of using sharing vehicles. It optimises a multi dimensional and multi criteria problem, based on a stochastic environment, such that the number of simulations and required computational power is even much higher than in case of AI Investments. The cost savings ratio is similar to the benefits achieved by AI Investments. Also the time of simulation is shortened significantly. The proposed solution could be also applicable to many other areas, like health, mobility, customer analysis, sales, and telecommunication, which require the execution of a significant number of simulations. The presented benefits in terms of cost savings and computation time shortening are thus meaningful also for these areas while allow to fully leverage the benefits of modern, machine learning based applications.

The following directions of work are envisioned. First, we plan to finish the realisation of the proposed SimaaS. Second, we will thoroughly evaluate it against other SimaaS and data-intensive serverless computing frameworks. Third, we will further investigate the exact conditions and requirements for scaling message queues and how these can be transformed into a certain part of the application's utility function. Fourth, we will explore how our proposition can be enhanced to cover the handling of other kinds of data-intensive applications, possibly through its integration with data-intensive workflow engines.

Acknowledgements. This work has received funding from the Functionizer Eurostars project and from AI Investments Fast Track to innovation project.

References

[1] M. A. Walker, R. Madduri, A. Rodriguez, J. L. Greenstein, and R. L. Winslow, "Models and Simulations as a Service: Exploring the Use of Galaxy for Delivering Computational Models," *Biophysical journal*, vol. 110, no. 5, pp. 1038–1043, 2016.
[2] P. Polasek, V. Janousek, and M. Ceska, "Petri Net Simulation as a Service," in *PNSE @ Petri Nets*, vol. 1160. Tunis, Tunisia: CEUR, 2014.
[3] T. Bitterman, P. Calyam, A. Berryman, D. E. Hudak, L. Li, A. Chalker, S. Gordon, D. Zhang, D. Cai, C. Lee, and R. Ramnath, "Simulation as a service (SMaaS): a Cloud-Based Framework to Support the Educational Use of Scientific Software," *IJCC*, vol. 3, pp. 177–190, 2014.
[4] S. Wang and G. A. Wainer, "Modeling and simulation as a service architecture for deploying resources in the cloud," *IJMSSC*, vol. 7, no. 1, 2016.
[5] S. Shekhar, H. Abdel-Aziz, M. E. Walker, F. Caglar, A. S. Gokhale, and X. D. Koutsoukos, "A simulation as a service cloud middleware," *Annales des Télécommunications*, vol. 71, pp. 93–108, 2016.
[6] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, *Serverless Computing: Current Trends and Open Problems*. Singapore: Springer Singapore, 2017, pp. 1–20.
[7] Y. Kim and J. Lin, "Serverless Data Analytics with Flint," *CoRR*, vol. abs/1803.06354, 2018.
[8] V. Sowrirajan, B. Bhushan, and M. Ahuja, "Qubole announces Apache Spark on AWS Lambda," Qubole, Tech. Rep., 2017. [Online]. Available: http://www.qubole.com/blog/spark-on-aws-lambda/
[9] A. Klimovic, Y. Wang, P. Stuedi, A. Trivedi, J. Pfefferle, and C. Kozyrakis, "Pocket: Elastic Ephemeral Storage for Serverless Analytics," in *OSDI*. Carlsbad, CA, USA: USENIX Association, 2018, pp. 427–444.
[10] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
[11] M. Tomassini, "Parallel and distributed evolutionary algorithms: A review," 1999.
[12] K. Kritikos and P. Skrzypek, "A review of serverless frameworks," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, 2018, pp. 161–168.
[13] ——, "Towards an Optimized, Cloud-Agnostic Deployment of Hybrid Applications," in *BIS*. Springer, 2019.
[14] A. Rossini, K. Kritikos, N. Nikolov, J. Domaschka, F. Griesinger, D. Seybold, and D. Romero, "D2.1.3—CAMEL Documentation," PaaSage project deliverable, October 2015.
[15] K. Kritikos, C. Zeginis, F. Griesinger, D. Seybold, and J. Domaschka, "A Cross-Layer BPaaS Adaptation Framework," in *FiCloud*. Prague, Czech Republic: IEEE Computer Society, 2017, pp. 241–248.
[16] G. Horn and P. Skrzypek, "Melodic: utility based cross cloud deployment optimisation," in *WAINA*. IEEE, 2018, pp. 360–367.
[17] V. Stefanidis, Y. Verginadis, I. Patiniotakis, and G. Mentzas, "Distributed Complex Event Processing in Multiclouds," in *ESOCC*, ser. Lecture Notes in Computer Science, vol. 11116. Como, Italy: Springer, 2018, pp. 105–119.
[18] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *Trans. Evol. Comp*, vol. 6, no. 5, pp. 443–462, Oct. 2002.
[19] T. H. Goodwin, "The information ratio," *Financial Analysts Journal*, vol. 54, no. 4, pp. 34–43, 1998.

[22]https://databricks.com/
[23]https://www.optimali.io/