

# Evaluation of Integrated Frameworks for Optimizing QoS in Serverless Computing

Anisha Kumari[1], Bibhudatta Sahoo[1], Ranjan Kumar Behera[2(✉)], Sanjay Misra[3], and Mayank Mohan Sharma[4]

[1] Department of CSE, National Institute of Technology, Rourkela, India
bdsahu@nitrkl.ac.in
[2] School of Computer Science and Engineering, XIM University, Bhubaneswar, India
ranjanbehera@xim.edu.in
[3] Department of Electrical and Information Engineering, Covenant University, Ota1023, Nigeria
sanjay.misra@covenantuniversity.edu.ng
[4] Zillow Inc., San Francisco, USA
mayankmohans@zillowgroup.com

**Abstract.** Serverless computing is an emerging cloud deployment model where developers can concentrate on developing application logic without worrying about the underlying architecture. It is similar to the platform as a service (PaaS) but at the functional level. Applications are usually deployed in the form of a set of functions independently and each function may be executed at separate servers thus also named as function as a service (FaaS). Serverless at the edge can handle thousands of concurrent functions invocations to process various kinds of events generated from resources like database, system logs, and other storage units, etc. A number of serverless frameworks like Openfaas, Openwhisk, Microsoft Azure, Amazon AWS allow dynamic scaling to handle the parallel request of stateless functions from the client-side. A separate container manager may be provisioned to handle distributed load for data processing. In this paper, we have evaluated the performance of serverless frameworks for parallel loads in terms of response time and throughput. In this paper, we have shown that the serverless framework is suitable for handling dynamic applications that can be executed on a number of stateless functions. An extensive comparison of the performance of serverless frameworks in handling concurrent invocations in terms of response time and throughput is also presented. It has been observed that Openwhisk is found to be the better serverless framework in terms of elasticity and scalability.

**Keywords:** Serverless computing · Function-as-a-service · Orchestration · OpenWhisk · Openfaas

## 1 Introduction

Serverless computing has attracted a lot of attention to small and medium scale industries for fulfill customer demands. Unlike traditional computing,

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12955, pp. 277–288, 2021.
https://doi.org/10.1007/978-3-030-87007-2_20

where users are required to maintain on-premises infrastructure, cloud computing abstains from these requirements as well as provides services as per user's demands. There is no need to purchase and maintain hardware, storage, and processing units etc. by the users. The data are accessed and stored over the Internet so that at any time from anywhere cloud users could able to access their stored data. Generally, cloud systems deliver three types of deployment models such as Infrastructure as a Service (IaaS) [1], Platform as a Service (PaaS) [2] and Software as a Service (SaaS). The environment of cloud computing is scalable and flexible which provides the users for leasing or releasing services as per the requirement. Users may rent the services with short-term dynamic plans or long-term preservation plans. Several cloud service providers facilitate the same type of services like network services, storage and computing. These services are approximating to non-functional Quality of Service (QoS) parameters for instance energy consumption, time, availability, cost, etc. Meanwhile, several cloud service providers, like Google Cloud Platform (GCP) [3], Amazon Web Services (AWS), Microsoft Azure, and IBM Bluemix showed up offering various cloud infrastructures and services in the cloud computing marketplace. Function-as-a-service (FaaS) leverages the best functionalities of Infrastructure as a Service (IaaS) and Container as a Service (CaaS), where services are delivered in the form of bundles known as containers. The basic architecture for FaaS also include service oriented and event driven features, which allow the users to develop Service oriented applications. Several cloud providers having started their FaaS platforms such as Azure Functions, AWS Lambda, IBM Openwhisk, and Google Cloud Functions, which are based upon container-orchestration systems such as Kubernetes. Over the evolution of FaaS platforms as well as the advancement of the cloud computing archetype, the serverless computing framework, whereupon the application is abstracted as a combined form of functions hosted on FaaS platforms orchestrated in a workflow has emerged [12].

Serverless computing has a number of advantages over the traditional cloud computating models by abstracting the management of underlying infrastructure. In serverless computing, FaaS platforms taking over entire operational and computational responsibilities for resource management, scaling, function deployment, and monitoring [16]. Developers can primarily concentrate on the business logic of functions, rather than expediting the application development. The execution model for serverless computing is shown in Fig. 1. The serverless computing models are found to be more effective due to its reduced cost model which is more flexible and the bill is based on exact usages of resources. Even though serverless architecture eases resource management and cost-effectiveness, some challenges are hindering its use by potential users due to its tradeoff between performance and cost [6]. Cost modeling, performance and optimization are non-trivial as well as the necessary measures, which needs to be taken care in SLA of computation model in serverless platform [4].

As shown in Fig. 1, there is no single traditional back end for serverless execution model. The application's front end communicates straightaway with the database, compute functions, or services through an API gateway. However, several services are hidden backside compute service functions, in which additional security steps and validation could take place.

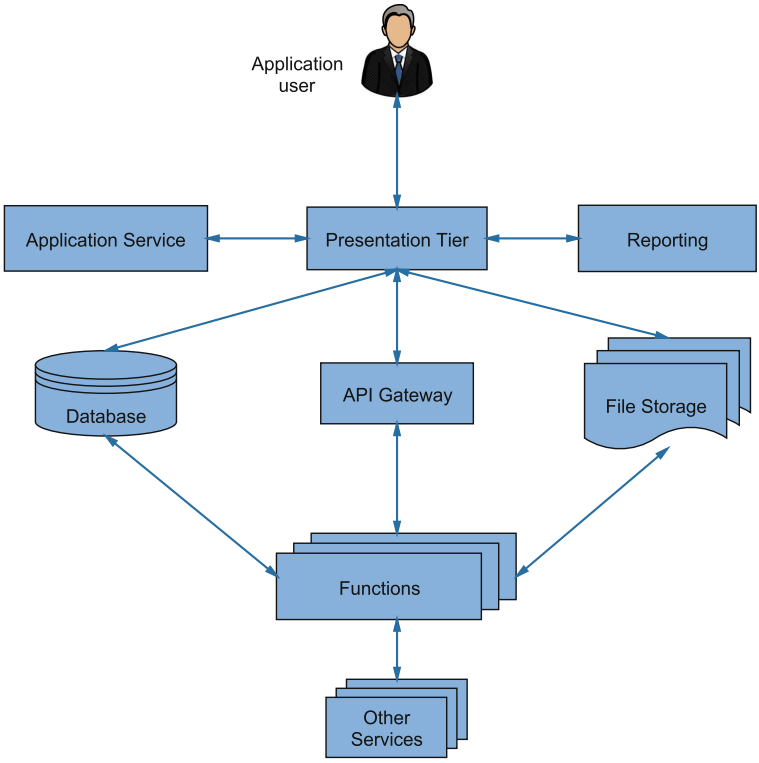



**Fig. 1.** Execution model for serverless computing

Serverless frameworks may help with the problem of layering and having to upgrade too much ground. There is room for developers to minimize or remove layering by breaching the system as a group of functions, also authorizing the front end to communicate with services securely and even the database directly, as illustrated in Fig. 1. All these things might be done in a structured manner to prevent complex implementations as well as dependency nightmares by defining service boundaries clearly, which allow functions to be autonomous, and planning how services and functions will interact.

The rest of this paper is organized as follows: Sect. 2 describes the motivation toward the work. In Sect. 3, some of the related work in this area have been discussed. Section 4 focuses on various aspect of the serverless computing framework. Section 5 describes the evaluation of various serverless frameworks and Sect. 6 conclude the paper.

## 2 Motivation

The motivation towards the research work may be described as these folds:

- The most suitable advantages of serverless computing are aligning computational costs with actual business requirements and reducing costs in general, but also more secondary benefits are identified like diminished lead time, enhanced agility, etc.
- Currently, several businesses are not utilizing serverless computing techniques or are underutilization due to several unknowns for instance performance challenges, complex cost structures, and uncertain facts regarding the implementation of this concept towards the current application architecture.
- The motivation in this research work is such as a serverless environment has more to provide to software application developers, architects, and providers, despite that, because of the novelty of the technology and economic and the relatively high architectural impact, its acceptance is inadequate behind or at the very least, is not highly utilized. Switching towards a serverless framework needs a considerate switch, comparable with the one architects as well as developers could make when micro-services will be suitable for a common architectural style. It certainly is not a one-size-fits-all kind of approach. The adoption of successful serverless computing depends on several factors and also needs many important non-arbitrary considerations.
- We hope this research will be able to contribute to the successful adoption of serverless technology by facilitating developers, software architects, and cloud providers with the insights, handlebars, references, etc. for making prior decisions of their particular applications and also for maximizing the potentials of this interesting new computing technology towards software deployment in the cloud.

## 3 Literature Survey

The serverless computing model focuses on revolutionizing the archetype and advancement of modern scalable applications, where developers can run event-driven code without managing or provisioning servers. This new technique experiments with an industry momentum around the cloud event abstraction. Very few research publications are there for the practical implementation of theoretical serverless concepts.

One of the reason behind the popularity of serverless computing is that it allows more responsiveness towards both providers and users, using market based schemes incorporation for resource allocation and pricing. Vipul et al. [19] have carried out a work, in which their approach brings about dynamic pricing strategy obtained as a result of optimization problem with dual form. They further developed mechanism of feedback which allows the cloud provider to achieve the optimal resource allocation, also when the utilities of users are unidentified. As a result of their simulation, it is shown that their approach can trail market-demand.

Maissen et al. [10] introduced a user-space application FaaSdom that facilitating benchmark tests over different types of serverless archetypes. It can be used to evaluate the serverless computing platform. An open-source standard

suite FaaSdom is easily deployed and facilitating significant perceptions over several metrics as well as serverless providers. They envisioned extending FaaSdom towards the following directions. Firstly, they have expanded the runtime environment and category of supported languages, as well as integrating the supports from the freely accessible orchestration tools. Secondly, they have planed to preserve an online, publicly accessible, and continuous deployment of the FaaSdom, to frame a large dataset of factual measurements which would be quite helpful for the researchers.

Another work has been carried out by Mistry et al. [13] in which authors proposed to investigate unikernel, an alternative to containers to build serverless platform at the edge, having security concerns and addressing in particular performance. They have presented UniFaaS, a prototype edge-serverless platform that leverages unikernels – tiny library single-address-space operating systems only containing the parts of the OS required to run a given application – for executing functions. The result is a serverless platform with less CPU footprints and extremely low memory, along with excellent performance. UniFaaS has been designed and deployed on low-powered single-board computer systems, like Arduino or Raspberry Pi, without compromising on performance.

In the public cloud, machine learning applications deployment considering prediction problems having a key challenge such as integrating the framework as well as the heterogeneity of resources towards optimizing the constraints [5]. Gunasekaran et al. [7] have proposed a model to frame a machine learning prediction model, which is self-managed and could optimize the various needs of the application, to have relied on peculiarities of the heterogeneous resource offered by the public cloud. In doing so, the authors discussed the compromises for intermixed resources such as functions besides Virtual Machines in serverless, also identified the fundamental issue closely related to the configuration of the resources. They proposed several resource management key policies making; (i) multi-dimensional SLO awareness, (ii) fluctuated request load awareness and (iii) latency awareness. These measures could be collectively utilized for cost-efficient prediction serving to have uncompromising accuracy as well as latency.

In a research paper by Sreekanti et al. [18] the authors argued about a well-suited dataflow API is preferable to the latency-sensitive task, also manageable for optimization even with faultless black-box machine learning frameworks. They presented the Cloudflow design, an archetype providing auto-scaling serverless an API, and realized it on the backend. The Cloudflow implements competitive execution and performance-critical optimizations transparently comprising operator fusion. Their analysis results that on synthetic workloads Cloudflow's optimizations having significantly improved performance, also on real-world prediction pipelines Cloudflow outperforms state-of-the-art prediction serving systems by as much as two times, achieving latency goals of trending applications such as real-time video analysis.

An event-driven Extract, Transform, and Load (ETL) pipeline serverless architecture was presented in a paper by Pogiatzis and Samakovitis [14], which provides an evaluation of its performance over a range of dataflow tasks of

changing velocity, payload size and frequency. They have designed an experiment utilizing the generated tabular data throughout varying event frequencies, processing power and data volumes in order to measure: (i) reliability on data delivery; (ii) the consistency of pipeline executions; (iii) economic scalability (cost of chargeable tasks); and, (iv) maximum payload size per pipeline.

Wen and Liu [20] have presented the first empiric survey on characterization and comparison of existing serverless workflow services, i.e., Azure Durable Functions, AWS Step Functions, Google Cloud Composer, and Alibaba Serverless Workflow. They firstly compare the characteristics from six dimensions, e.g., parallelism support, orchestration way, data payload limit, etc. After that they have measured the performance of these serverless workflow services beneath various experimental settings (i.e. data-flow complexity, function complexity, and different levels of activity complexity). On the basis of the results, some interesting findings, e.g., only under high data-flow complexity conditions, the performance of serverless workflow have a certain impact, can be useful and helpful for the developers and the cloud providers. Finally, they reported a series of findings and implications to further facilitate the current practice of serverless workflow.

## 4 Serverless Computing Framework

In this section, a brief introduction to FaaS, serverless workflow and integrated tools for serverless platform has been discussed.

### 4.1 Function as a Service (FaaS)

Serverless computing can also be called Function as a Service or FaaS. Clients upload their business logics and providers maintain the traffic routing, scaling, etc. as well as mainly focus on deployment. From a client's viewpoint, the serverless technique can facilitate the approach of deploying business logic into production. Capacity planning, maintaining the virtual machines (VM's), and Scaling are hidden from the developer. Additionally, the billing mechanism relies on the actual amount of resources consumed by an application.

Abstraction is one of the core feature in FaaS model, where set of instructions known as functions abstracts the implementation part of the business logic in an application. The unit of execution are the functions in serverless computing, which are similar to the methods in object oriented programming model or the functions in functional programming. The major task of function is to handle stateless job required by an application. Containers are used to bundle the source codes along with their dependencies. However, each of the function are executed separately with independent manner with the help of lightweight virtualization approaches such as Uni-kernels [17], Docker containers [11], or even processes. The sequence diagram for execute a function in response to a event is shown in Fig. 2.

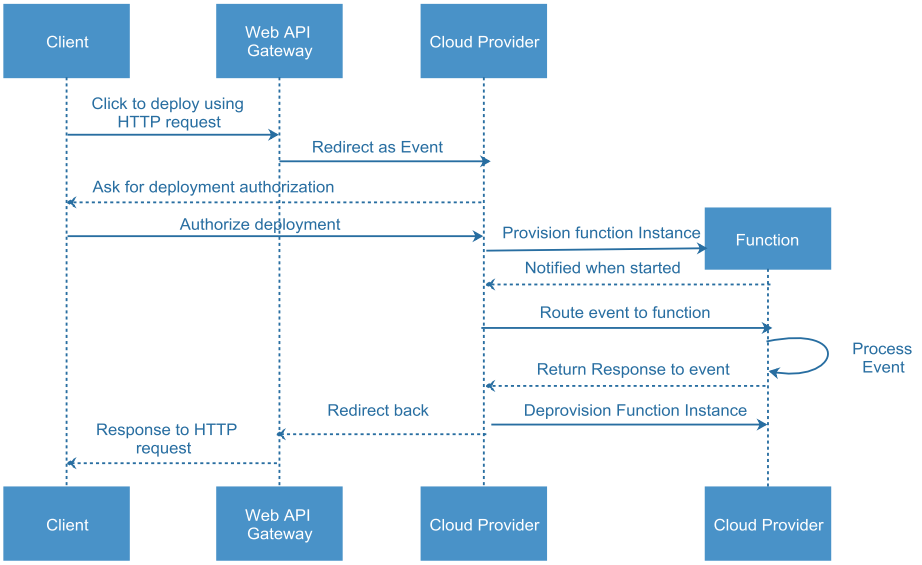



**Fig. 2.** Sequence diagram for execution of FaaS

The first FaaS model was lunch by AWS known as Lambda in 2014 which can deploy, store bundles of function and provide functions as services through event triggers. It also facilitates monitoring of resource utilization and auto scaling features. Some of the other platforms also provides the same functionalities by open-source communities and public cloud service providers such as Microsoft azure Google functions etc. In FaaS, developers have very little control over resources and the infrastructure. They are only responsible for deploying their applications along with the triggered events [9]. For example; in AWS Lambda during execution, the concurrency level and the amount of allocated memory are the only options for tuning the performances of functions. By provisioning and reserving additional instances to the host functions making high concurrency level and warm instance unavailable so that delay in container initialization provisioning has also referred to as cold starts which decreases the performance of framework as well as the number of throttles with extremely heavy requests of workloads.

### 4.2 Serverless Execution Model

Generally, the serverless framework is the function orchestration for implementing the complete programming instructions to run a serverless application. Several existing platforms such as AWS, IBM, Microsoft, etc. give an idea about, how a serverless framework considers as a process of grouping individual functions and coordination among those functions. The examples for deploying a serverless application are existing in various environments and it is shown that

the serverless framework is having four types of structure, which include cycle, self-loop, branch, and parallel. It is proved in prior researches that a framework appropriate to Petri net and the directed acyclic graph (DAG), is beneficial for cost as well as performance modeling of systems in parallel, distributed and scientific, computing. Despite that, each of them is unfavorable to the serverless archetype, as in DAGs loops and cycles are restrictive because they cause the problem of state explosion lacking efficient solutions for the Petri nets. Also in cloud computing, Petri nets are having their limited support and high complexity of non-functional requirements which are considered as the major shortcomings. Therefore, cost modeling and performance of serverless applications require a brief concept about a serverless archetype, that adapts to new characteristics of the serverless framework.

### 4.3 Integrated Tools for Serverless Platform

Though FaaS frameworks are beneficial for simple stateless applications likewise web-serving and ETL, several complex distributed applications could not be expressed in the context of a single function. Many existing FaaS orchestration tools offer clients for composing manifold functions to facilitate more complex application semantics.

Since 2014 Amazon Lambda's appearance, large-scale industries providing cloud services, have accepted FaaS framework to develop serverless applications using various tools. In this research work, serverless framework like IBM apache Openwhisk [15], Openfaas [8], AWS and Microsoft Azure have been considered for the implementation. Apache Openwhisk is a distributed serverless model which executes functions in response to a triggered event in a scalable manner. It inherently manages infrastructure, scalable servers with the help of a docker container which allows the users to develop applications seamlessly without worrying about the underlying infrastructure. It also supports multiple languages like Go, Java, Python, NodeJs, Ruby, Swift, etc. It is event-driven which allows the serverless execution of functional code known as actions that may be triggered from external sources like web or mobile-based applications, chat-bots, scheduled alarm, etc.

## 5 Evaluation of the Execution Model

In this paper we, have evaluated the serverless computing environment by deploying functions concurrently for various applications. The performance has been evaluated in term of throughput and response time. Each of the performance parameters are measured using concurrent function invocations. Response time corresponding to various frameworks are measured with respect to number of users.

### 5.1 Experimental Setup

Several functions from Openwhisk, Openfaas, Azure Functions were deployed having similar configuration. Each of the function are stateless and allocated with one CPU and 1.0 GB of memory for execution with respect to an event. All the events are triggered by HTTP request. Apart from the HTTP Trigger other triggers like database trigger and object storage trigger have also been used in some framework for generating event. Apache benchmark (ab) tool3 have been used to generate HTTP request to invoke the functions deployed on server.

### 5.2 Results and Observations

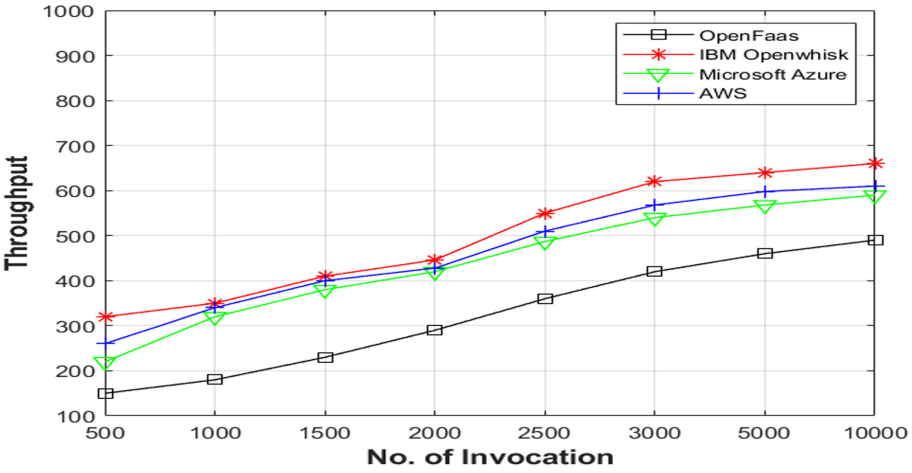



**Fig. 3.** Comparative analysis in term of throughput

The frameworks that are considered in this paper are evaluated based on the function throughput as it is one of the metric used to identify the quality of concurrent process. It deals with how many functions instances are supplied at the time of extensive request. It is important to note that maximum allowable concurrent function invocations varies from framework to framework. The resources for function invocations are also different for frameworks. For example, AWS use S3 object storage or DynamoDB database for parallel function invocations, OpenWhisk use couchDB database as the resource for function invocations. Figure 3 shows the change of throughput per second over the number of function invocations. It can be observed that IBM Openwhisk framework offers better throughput as compared to other frameworks. It attains maximum throughput at 3000 invocations. The average throughput is found to be 500 per second, which is highest as compared to other serverless frameworks. The throughput of Openfaas is very less as at all the invocations. Microsoft Azure

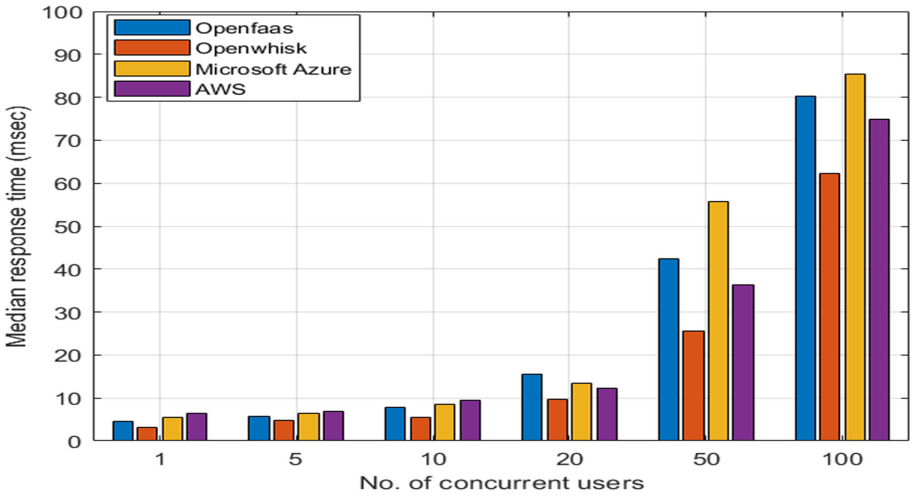



**Fig. 4.** Performance comparison in term of response time

and AWS functions shows similar bahaviour in attaining maximum throughput. The average throughput for AWS, microsoft Azure and Openfaas are found to be 465, 440 and 322 respectively. From the experimental analysis, it can be observed that the performance in term of throughput is better in IBM's Openwhisk as compared to other frameworks.

The performance of the frameworks is also evaluated in term of median response time over the no. of concurrent user request. Figure 4 shows the comparative analysis of various serverless frameworks in term of median response time. The median response has been calculated by using response time across all 10000 iterations. 10000 requests has been deployed using different level of concurrency users (1, 5, 10, 20 and 50). It can be observed from Fig. 4 that the fastest response is obtained by Openwhisk in all the scenario. The response time for microsoft Azure and AWS is almost same for request by 5 concurrent users. Microsoft Azure has median response time about 55 msec for 50 concurrent users which is highest among all the frameworks. The clear difference in mean response time is reflected at 50 number of concurrent users where Openwhisk is better as compared to others.

## 6 Conclusion

Serverless framework in cloud computing offers a number of advantages which include cost-efficiency, scalability, low latency, better responsiveness while deploying applications in servers. Developers need not to worry about the underlying architecture and the cost of service is based on pay as you go basis. Applications are deployed in the form of independent stateless functions which can be invoke to process the event triggered by user. Several framework have been

emerging to provide services in serverless platform. In this paper, we have demonstrated four open source serverless frameworks such as Openfaas, Openwhisk, AWS and Microsoft Azure to prove service in low cost and scalability manner. Their performance has been compared based on the response time and throughput on concurrent request from the client side. Each of the request may be in the form of triggers which leads to execution of functions at the server side. It has been concluded that Openwhish is found to be better severeness framework in term of median response time and throughput.

In future, parameters other than throughput and response time may be taken into consideration for evaluating the performance of serverless tools.

## References


1. Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., Al-Jaroodi, J.: A survey of load balancing in cloud computing: challenges and algorithms. In: 2012 2nd Symposium on Network Cloud Computing and Applications, pp. 137–142. IEEE (2012)
2. Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., Ahmad, I.: Cloud computing pricing models: a survey. Int. J. Grid Distrib. Comput. **6**(5), 93–106 (2013)
3. Arteaga, D., Cabrera, J., Xu, J., Sundararaman, S., Zhao, M. CloudCache: on-demand flash cache management for cloud computing. In: 14th USENIX Conference on File and Storage Technologies, FAST 2016, pp. 355–369 (2016)
4. Baldini, I., et al.: Serverless computing: current trends and open problems. In: Chaudhary, S., Somani, G., Buyya, R. (eds.) Research Advances in Cloud Computing, pp. 1–20. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5026-8_1
5. Behera, R.K., Jena, M., Rath, S.K., Misra, S.: Co-LSTM: convolutional LSTM model for sentiment analysis in social big data. Inf. Process. Manage. **58**(1), 102435 (2021)
6. Behera, R.K., Shukla, S., Rath, S.K., Misra, S.: Software reliability assessment using machine learning technique. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10964, pp. 403–411. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95174-4_32
7. Gunasekaran, J.R., Mishra, C.S., Thinakaran, P., Kandemir, M.T., Das, C.R.: Implications of public cloud resource heterogeneity for inference serving. In: Proceedings of the 2020 6th International Workshop on Serverless Computing, pp. 7–12 (2020)
8. Kaewkasi, C.: Docker for Serverless Applications: Containerize and orchestrate functions using OpenFaas, OpenWhisk, and Fn. Packt Publishing Ltd. (2018)
9. Kumari, A., Behera, R.K., Sahoo, K.S., Nayyar, A., Kumar Luhach, A., Prakash Sahoo, S.: Supervised link prediction using structured-based feature extraction in social network. Concurrency Comput. Pract. Exp., e5839 (2020). https://doi.org/10.1002/cpe.5839
10. Maissen, P., Felber, P., Kropf, P., Schiavoni, V.: FaaSdom: a benchmark suite for serverless computing. In: Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems, pp. 73–84 (2020)
11. Merkel, D.: Docker: lightweight Linux containers for consistent development and deployment. Linux J. **2014**(239), 2 (2014)

12. Misra, S.: A step by step guide for choosing project topics and writing research papers in ICT related disciplines (2021). https://doi.org/10.1007/978-3-030-69143-1_55
13. Mistry, C., Stelea, B., Kumar, V., Pasquier, T.: Demonstrating the practicality of unikernels to build a serverless platform at the edge (2020)
14. Pogiatzis, A., Samakovitis, G.: An event-driven serverless ETL pipeline on AWS. Appl. Sci. **11**(1), 191 (2021)
15. Quevedo, S., Merchán, F., Rivadeneira, R., Dominguez, F.X.: Evaluating apache OpenWhisk-FaaS. In: 2019 IEEE 4th Ecuador Technical Chapters Meeting (ETCM), pp. 1–5. IEEE (2019)
16. Rodríguez, G., Mateos, C., Misra, S.: Exploring Web service QoS estimation for web service composition. In: Lopata, A., Butkienė, R., Gudonienė, D., Sukackė, V. (eds.) ICIST 2020. CCIS, vol. 1283, pp. 171–184. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59506-7_15
17. Scherb, C., Marxer, C., Tschudin, C.: Execution plans for serverless computing in information centric networking. In: Proceedings of the 1st ACM CoNEXT Workshop on Emerging in Network Computing Paradigms, pp. 34–40 (2019)
18. Sreekanti, V., Subbaraj, H., Wu, C., Gonzalez, J.E., Hellerstein, J.M.: Optimizing prediction serving on low-latency serverless dataflow. arXiv preprint arXiv:2007.05832 (2020)
19. Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uă, A., Iosup, A.: Serverless is more: from PaaS to present cloud computing. IEEE Internet Comput. **22**(5), 8–17 (2018)
20. Wen, J., Liu, Y.: An empirical study on serverless workflow service. arXiv preprint arXiv:2101.03513 (2021)