May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

Comparison &Adoption of FOSS Serverless Computing for Enterprise Openstack Cloud Platform

Rajesh Rompicharla[1], Bhaskar Reddy P.V[2]

[1]PG Student, [2]Professor,
[1,2]School of C and IT, REVA University, Bangalore-India
[1]rajesh.rom@gmail.com, [2]bhaskarreddy.pv@reva.edu.in

Article Info
Volume 83
Page Number: 4844-4850
Publication Issue:
May-June 2020

Abstract
Cloud Computing paradigm accelerated the phase of development & deployment of software applications over on-demand api-enabled programmable infrastructure. However, the decision factor of hosting the applications over On-Premises or Public Cloud generally dependent on the features of Cloud Service offerings like IaaS, PaaS, CaaS and SaaS from the respective Cloud Platforms. Pubic Clouds are always on top with their taxonomy of service offering list, this worries enterprise IT departments about future platform dependency and cost aspects, hence phenomenon of Open Source Cloud Computing platforms like Openstack [25] are encouraged by IT Enterprises to offer Cloud Services similar to Public Clouds. Serverless Computing is an emerging cloud service construct wherein software applications decompose into multiple independent stateless functions, which are run only when invoked or trigger by events and killed when functions session expired. There are multiple FOSS Serverless Computing frameworks available, comparison & adoption of suitable framework for Openstack based On-Premises Cloud platform with appropriate design and implementation procedure is the objective of this paper.

Keywords: *hosting the applications, Cloud Computing paradigm*

Article History
Article Received:*19 November 2019*
Revised: *27 January 2020*
Accepted: *24 February 2020*
Publication: *16 May 2020*

1. Introduction

The evolution of cloud computing simplified the datacenter physical computing machines management, which were in the "bare-metal" state. The operation and maintenance of the datacenter and time to offer the compute resources to the end-customer improved with the adoption of Cloud Computing and Virtualization. The main reasons of the Cloud success are resources configured for API accessibility, enablement of broad network access and services made available on-demand through the portal namely Infrastructure as a service (IaaS).The deployment of Cloud IaaS in enterprise is called Private Cloud, consuming resources through Internet from external vendor is called Public Cloud, enabling connectivity between Private & Public and offering an option to choose services between them is called Hybrid Cloud.

The transformation of software engineering from Monolithic to Service Oriented and Microservices Architectures fueled the need of different Cloud Services apart from IaaS. The flexibility for developer is consideration to offer new cloud service called Platform as Service (PaaS). PaaS offers the readily available software packaged Virtual Machines (VM) that simplifies developer job to concentrate on application development. Since Microservices adoption gradually increased, the requirement of

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

server form-factor shift from VM to Container had become essential. The new service offering Container as a Service (CaaS) prevailed in competition to IaaS. Container offers virtualization at the Operating system level using abstractions like chroot, namespace and cgroups, required application packaging offers by container engine like Docker and orchestration for containers delivered through Container Orchestrator like Kubernetes [24]. Free and Open-Source (FOSS) adoption helped enterprises to avoid vendor lock-in, de-facto standard softwares like Openstack (IaaS), Docker (Container Engine) and Kubernetes (Container Orchestrator) emerged as main players and gathered the trust of enterprise to adopt for their production needs. Since Cloud and Software engineering transformed at rapid speeds, developer difficulty to accustom to multiple service environments had also grown. Complexity that developer feels to inculcate knowledge of multiple layers of cloud services like IaaS, PaaS and CaaS to leverage them for better software development addressed with Serverless Architecture patterns, Figure 1 provides the

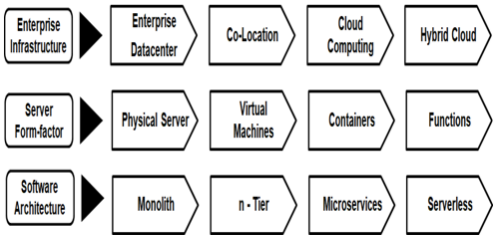


Figure 1: Transformation of Enterprise Infrastructure, Server Form-factor and Software Architecture

overview of overall technology transformation that influences Enterprise Cloud Computing.

After a decade, Berkley University had published paper on new era of Cloud Computing [1] and mentioned that Serverless computing gone be the default computing paradigm in future, largely replacing services that needs server-full knowledge for application development. Serverless narrowed down entire cloud service offerings to two services; those are Function as a Service (FaaS) & Backend as a Service (BaaS). Functions are independently deployable software constructs that are subset of Microservices. Backend services are re-suable software service like database, S3 storage bucket, authentication, analytics and e-mail service etc. Serverless computing simplifies the cloud for developers such that they have to concentrate on developing business logic in terms of functions f(x), integrate it to backend services and chose available event triggers to run the code only on need basis over Cloud Platform. Cloud provider takes care of rest and assures required infrastructure availability for applications functions to run efficiently, charges only for the time that resources consumed to run stipulated application code. In brief, Serverless computing is an event-driven, stateless, code execution, utility-based environment that enables developer to write code and consume services, rest of the operations are offload to Cloud provider.

Serverless Computing already proven its mettle with Public Cloud offerings like AWS Lambda, Microsoft Azure Functions and Google Cloud Functions. Enterprise needs to adopt Serverless Computing to leverage the benefits it offer, hence comparison and adoption of FOSS Serverless frameworks is essential with respect to Openstack (FOSS de-facto Cloud Computing Platform).Since the research in this direction is essential, the necessary work performed and this paper is the extract of the research work conducted. Below are the motivated Research Questions (RQ) that covers the extent of the Serverless domain covered as part of this research paper:

- RQ1:Which FOSS based Serverless Computing framework suits for Enterprise needs?
- RQ2: What are the economic viability and technically feasibility of implementing selected (RQ1) Serverless Computing framework on Enterprise Openstack-Cloud?
- RQ3: What are the design and implementation practices to follow to adopt Serverless Computing for enterprise cloud platforms?
- RQ4: What are the performance and efficiency measures to be taken care for Serverless Computing workloads?

The further sections of this paper organized as follows: Section-II provides the overview of literature survey on RQ's and corresponding Research Methodology steps (RMS1-4). Section-III addresses RMS5 - Serverless Architecture design framework for Enterprise Openstack Cloud Computing. The test approach, evaluation and outcomes described in Section IV addresses the RMS6-7. Finally, Section V concludes the paper and outlines the future research steps.

Design Science Research Methodology (DSRM) is adapted for this research paper. The research effort roughly divided into two parts. The first step is theoretical literature research (steps RMS 1 – 4), second part considered to be applying the knowledge to design, demonstrate, and evaluate (steps RMS 5-7).

2. Literature Survery

An extensive literature review carried out on Serverless Cloud Computing Frameworks and their applicability with respect to efficiency and performance for Enterprise Production expectations. Total Twenty research papers identified that are related to this paper RQ's; research work of those papers provided the right direction of what worked and what did not worked. Below are points that influenced the research of this paper.

From the paper [1] Berkeley view on Serverless computing, provided the glance of future Cloud Computing revolution. However, issues that needs to

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

be taken care relate to ephemeral storage [17], which must provide high IOPS & low latency, BaaS services like databases demands higher Persisted Storage performance.

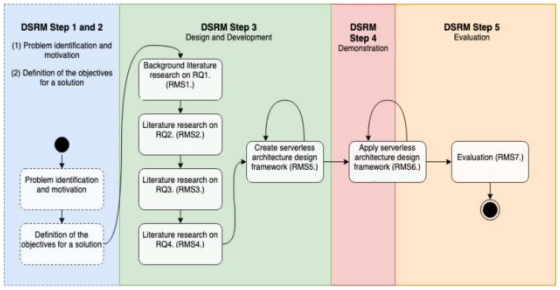


Figure 2: Design Science Research Methodology (DSRM) adoption to this research paper

From the papers [2]& [3],the clarity to Serverless fitment of FaaS& BaaS within current cloud offering and separation between Business & Operational Logicin Serverless clearly highlighted. The current uses cases of FaaS are particularly Bursty & CPU Intensive; hence, essential performance engineering tests and proper evaluation is a clear necessity.
From the papers [4], [5], Fonk-apps.io [6], provide the clear commonality among multiple FOSS Serverless Computing frameworks and their comparison on different aspects. The commonality is the FaaS offering and Kubernetes usage as Resource Manager among all the frameworks; differentiation is programming languages offered, Autoscaling performance & metric, Message queue integration and throughput & latency.

Defining FaaS platform & Comparison

The generic Serverless FaaS platforms uses two data stores for the functions those are Metadata and Function stores [18]. As name suggests for the quick retrieval of important information stores with Metadata Store and actual code stores in Function store. The reason to have two stores is two different access patterns, one for quick retrieval of information and other to run and deploy actual function.

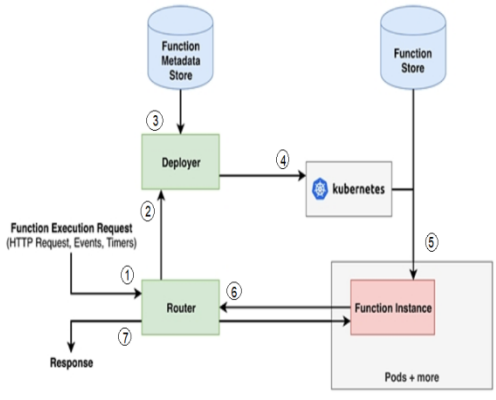


Figure 3: Anatomy of the runtime of a FaaS platform

Events triggers execution request receive by Router; the component that decides which function goes for execution cycle, also responsible for forwarding function request to the deployer. The Deployer components accepts deployment request from Router, and assess the demand for function from the metadata function store and decide how function to be deployed through the Resource Manager (Kubernetes). Resource Manager is typically a base layer below FaaS platform: manages the deployment of cloud resources, such as containers, network and storage. The Function Instance is the outcome product that accepts and receive the requests from the Router and serves basic needs of application access for the users.

The Deployment manager is the key difference between two different models; those are Events as Function Serverless Model and Container as a Function Serverless Model. The required container engine properties define the necessity of control & flexibility that models are expected to offer; it

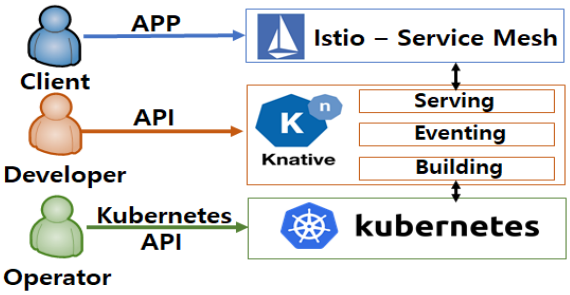
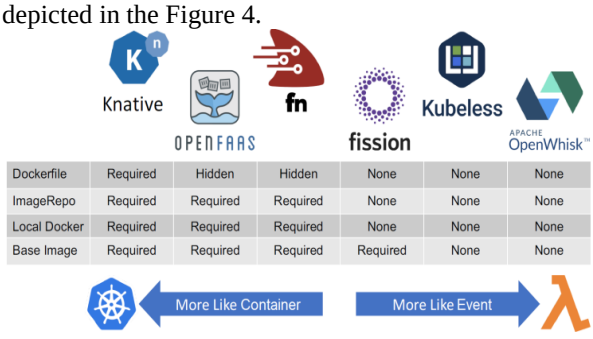


Figure 6: Knative Architecture and Components

depends on the objective of the FaaS systems to serve the defined purpose. If objective is very specific and does not change frequently then Events as Function model is the best choice, otherwise for the required dynamic objectives Container as Function model could exploit Deployment Manager that is Kubernetes feature sets.

The differentiation among the popular projects based on GitHub star popularity & key features conducted by fonk-apps.io [6], results published as depicted in the Figure 4.



Dockerfile	Required	Hidden	Hidden	None	None	None
ImageRepo	Required	Required	Required	None	None	None
Local Docker	Required	Required	Required	None	None	None
Base Image	Required	Required	Required	Required	None	None

Figure 4: FaaS on Kubernetes Landscape findings from fonk-apps.io

RMS1: From the overall comparison, Kubeless (Event as Function model) and Knative (Container as

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

Function model) suits for Enterprise needs.

RMS2: Since Kubernetes acts as common deployment manager across the FOSS-Serverless computing frameworks, also has native deployment integration with the Openstack Cloud Platform (IaaS) which is viably available as de-facto FOSS Cloud for Enterprises. Qinling [26] the native Openstack FaaS project is yet to move to its matured state.

Serverless Platform Design & Implementation requirements

Knative and Kubeless architecture and design base line requirements available from papers [4] and [7], since Kubernetes is the common factor the required abstraction and multi-tenancy are the configurable items for containers.

Knative [23] is a platform for building, deploying and running modern Serverless on Kubernetes. It offers required Middleware for building and running code on container-based applications, this helps developers to focus solely on writing code. Knative has native integration to run over multiple cloud providers. Knative itself has three building blocks, Building provide a cloud native build system for container orchestration works with Kubernetes; Eventing manage events with universal subscriptions; Serving is request-driven model that offers functions to run and scales up and down to zero, works with Istio and it models that Function is always executed and available through Istio' s Ingress Gateway.

Kubeless[22] leverages Custom Resorce Definition (CRD) feature of Kubernetes API and creates custom resoruce path for Function Objects. CRD can be namespaced and cluster-scoped, this enables Kubeless functions to be treated as normal Kubernetes constructs in the background, this enables Kubeless controller accessible through Kubernetes-API.

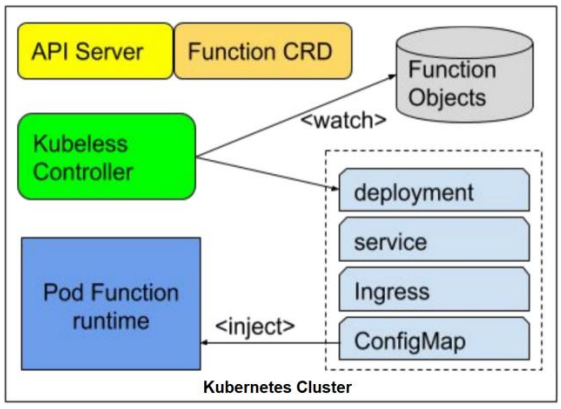


Figure 6: Kubeless Architecture and Components(https://kubeless.io/)

Kubeless Deployment manager calls the Kubernetes Container Orchestrator to invoke function runtimes. Deployment manager installs function dependencies using an Init-container, config-map feature used to inject configuration on run-time function. The required function triggers can be configured as Kubernetes Services and ingress traffic mapping for the function runtime can be influenced using Kubernetes Ingress-Controller.

RMS3: From the above the required design and implementation requirements are Kubernetes Container Orchestrator for Containers and IaaS to run Kubernetes virtual machines also called as Master & Worker Nodes.

Serverless Performance considerations

Serverless performance challenges are key considerable items to design the environments in production environments for the expected results. Several items come under scope of performance implications with respect to Serverless important items [8] are provisioning overhead, co-located Functions performance isolation, database and storage triggers. Concurrency aspects impacts for Serverless are throughput for number of connections [4], CPU Intensive workload, Disk Intensive Workload, Network Intensive workload [9].

RMS4: Performance factors highlighted above needs a consideration while designing Kubernetes and Openstack for Serverless computing frameworks.

3. Serverless Architecture & Design Over Openstack

Serverless architecture & design over the enterprise Openstack Cloud Computing platform depicted below as Figure 7. The high-level components involved are Events, Workflows, FaaS and BaaS.Cloud platform enables Function run times run as FaaS offering and Backend services run as BaaS offering. FaaS & BaaS leverages Kubernetes Container orchestration and run Docker containers. Kubernetes using native integration leverages Openstack Computing platform.

Two variants of FaaS, Kubeless for Events as Function Serverless Model and Knative for Container as a Function Serverless Model available for users. Based on the customer comfort one of the model can be selected. Customer do not want to manage underlying infrastructure can select Kubeless and others who want to explore multiple options from the underlying infrastructure can select Knative.

RMS5: Kubernetes and Openstack integrate tightly together using their native support to offer the infrastructure enablement for FaaS & BaaS workloads. The required Multi-Tenancy and Compute Resource quotas allocations configured based on the Application owner request. Storage requirement varies between FaaS and BaaS, ephemeral storage for FaaS and Persistent storage for BaaS are the requirements. Since FaaS and BaaS workloads can be CPU & I/O intensive,

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

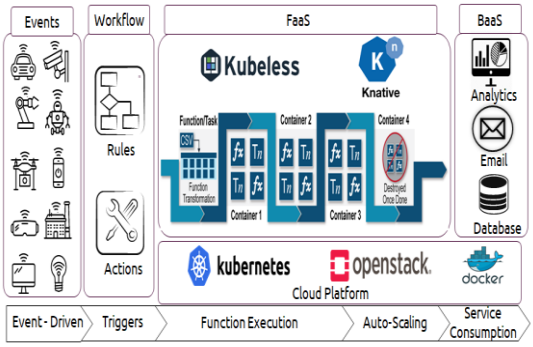


Figure7: Enterprise Serverless Computing Architecture & Design over Openstack Cloud Platform

compute optimizations and dedicated network bandwidth are required.

4. Openstack & Kubernetes Test Infrastructure For Serverless Computing Platforms

Openstack Cloud Computing platform offers the IaaS services from the Compute Servers, the platform needs other Servers namely Controller Servers, Storage Servers for the fulfillment of the overall requirement. As depicted below in Fig 8, Tenant projects can be created to host Knative & Kubeless and name them as FaaS (Fig 8 depicts only Kubeless), this project does not required Persistent storage, ephemeral storage suffice the requirements of FaaS. Hosting BaaS needs a separate Tenant Project along with Persistent Storage Volumes.

From the above requirements of FaaS & BaaS, the required storage and network readiness configurations needed at Openstack level. Tenant's creation with three virtual machines per tenant are required for Kubernetes installation.

Kubernetes installation needs one Master Virtual Machine to manage the environment and two Worker Nodes to run the containers with high availability. Kubernetes uses Flannel over Openstack Neutron Network for Layer 2 Network connectivity and Calico for the Network Security policies. Kubernetes uses native storage plugin to integrate with Openstack cinder for Storage requirement needs. Kubernetes Ingress-controller integration requirement supported using Openstack native load balancer Octavia.

RMS 6-Test Setup:

Hardware: Intel i7 8-Hyper-thread cores & 16 GB RAM
Openstack: Canonical Charmed Openstack, Stein
Operating System: Ubuntu 16.04.1 LTS
Kubernetes: v1.16.1, Docker v18.09.2
Knative: v0.8, Kubeless v1.0.4
Python 'Hello-world' function
Python 'HTTP' function
Workload Generator: wrk

RMS7: FaaS and BaaS workloads are CPU & Memory Intensive; Openstack Compute Optimizations likeCPU Pinning & Huge Pages [21] enabled for evaluation with Kubeless Function workloads to check the impact on performance.

Below are the test outcomes

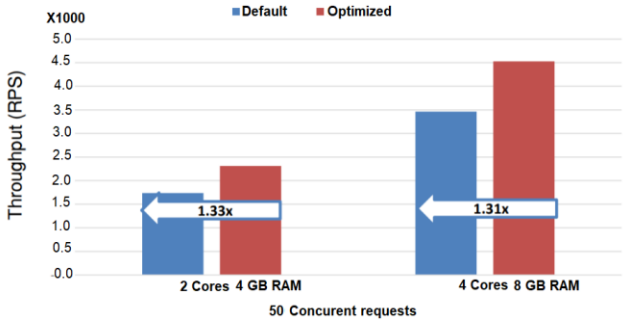


Figure 9: Kubeless Throughput Report in Requests per Second

Tests proved that Openstack optimization significantly improved the functions Throughput & Latency performance.

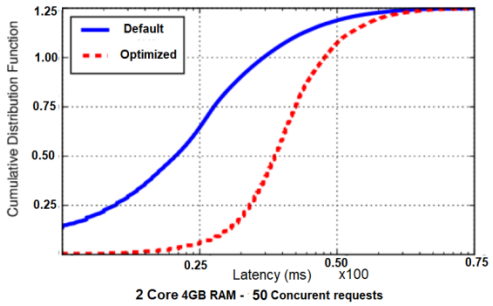


Figure 10: Kubeless Latency in Milliseconds

5. Conclusion

This research work performed a preliminary investigation to compare & adopt FOSS Serverless frameworks over Enterprise Openstack Cloud Computing Platform. As part of the research, the key objectives achieved with the selection of FOSS Serverless frameworks Knative & Kubeless, Designing & Architecting Serverless frameworks over Openstack Cloud Computing Platform, also verified and confirmed the performance improvement with Openstack Optimizations for Kubeless Function. Serverless Computing Governance & Compliance topics includes in future research.

References

[1] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, David A. Patterson "Cloud Programming Simplified: A Berkeley View

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

on Serverless Computing", arXiv:1902.03383 [cs.OS], 9 Feb 2019.

[2] Erwin van Eyk & Alexandru Iosup, "The SPEC cloud group's research vision on FaaS and Serverless architectures" WoSC '17: Proceedings of the 2nd-International Workshop on Serverless Computing, December 2017, Pages 1-4.

[3] Geoffrey C. Fox, Vatche Ishakian, Vinod Muthusamy, Aleksander Slominski "Status of Serverless Computing and Function-as-a-Service (FaaS) in Industry and Research", arXiv: 1708.08028 27, Aug 2017.

[4] Junfeng Li, Sameer G. Kulkarni, K. K. Ramakrishnan, Dan Li "Understanding Open Source Serverless Platforms: Design Considerations and Performance", arXiv: 1911.07449, 13 Dec 2019.

[5] Sunil Kumar Mohanty, Gopika Premsankar "An Evaluation of Open Source Serverless Computing Frameworks", 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), December 2018, ISSN: 2330-2186.

[6] Pete Johnson "examining-the-faas-on-k8s-market", November 2, 2018
https://blogs.cisco.com/cloud/examining-the-faas-on-k8s-market
Nima Kaviani, Dmitriy Kalinin, Michael Maximilien "Towards Serverless as Commodity: a case of Knative" WOSC '19: Proceedings of the 5th International Workshop on Serverless Computing,December 2019, Pages 13–18.

[7] Hyungro Lee, Kumar Satyam, Geoffrey Fox "Evaluation of Production Serverless Computing Environments" 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), July 2018, ISSN: 2159-6190.

[8] Wes Lloyd, Shruti Ramesh, Swetha Chinthalapati, Lan-Ly, and Shrideep Pallickara "Serverless Computing, An Investigation of Factors Influencing Microservice Performance" 2018 IEEE International Conference on Cloud Engineering (IC2E), April 2018, ISBN: 978-1-5386-5008-0.

[9] Jashwant Raj Gunasekaran, Prashanth Thinakaran, Mahmut Taylan Kandemir, Bhuvan Urgaonkar, George Kesidis,Chita Das "Spock: Exploiting Serverless Functions for SLO and Cost Aware Resource Procurement in Public Cloud" 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), July 2019, ISSN: 2159-6190.

[10] Louis Racicot, Nicolas Cloutier, Julien Abt, Fabio Petrillo "Quality Aspects of Serverless Architecture: An Exploratory Study on Maintainability" 14th International Conference on Software Technologies, January 2019, DOI: 10.5220/0007842000600070.

[11] Gojko Adzic, Robert Chatley "Serverless computing: economic and architectural impact" the 2017 11th Joint Meeting, August 2017, DOI: 10.1145/3106237.3117767

[12] Andreas Christoforou, Andreas S. Andreou "An effective resource management approach in a FaaS environment", Published in ESSCA@UCC 2018, Corpus ID: 85444026

[13] Aleksi Pekkala " Migrating a web application to serverless architecture", Published in ESSCA@UCC 2019, Corpus ID: 198329300.

[14] Ana Klimovic, Yawen Wang, Christos Kozyrakis, Patrick Stuedi profile imagePatrick Stuedi, Jonas Pfefferle profile imageJonas Pfefferle, Animesh Trivedi profile imageAnimesh Trivedi " Understanding Ephemeral Storage for Serverless Analytics",
USENIX ATC '18: Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference, July 2018, Pages 789–794.

[15] Alfonso Perez, Sebastián Risco, Diana María Naranjo Delgado, Miguel Caballer "On-Premises Serverless Computing for Event-Driven Data Processing Applications" 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), July 2019, DOI: 10.1109/CLOUD.2019.00073.

[16] Paul Castro, Vatche Ishakian, Vinod Muthusamy, Aleksander Slominsk "The Rise of Serverless Computing" Communications of the ACM, December 2019, Vol. 62 No. 12, Pages 44-54 10.1145/3368454.

[17] Erwin van Eyk "Four Techniques Serverless Platforms Use to Balance Performance and Cost"
https://www.infoq.com/articles/serverless-performance-cost/

[18] Hai Duc Nguyen, Chaojie Zhang, Zhujun Xiao, Andrew A. Chien "Real-time Serverless: Enabling Application Performance Guarantees" WOSC '19: Proceedings of the 5th International Workshop on Serverless Computing, December 2019, Pages 1–6.

[19] Serverless Architecture Conference 2019, April 8 – 10, 2019, The Hague, Netherlands https://serverless-architecture.io/wp-content/uploads/2019/02/SLA_Whitepaper_v3.pdf

May-June 2020
ISSN: 0193-4120 Page No. 4844-4850

[20] https://docs.openstack.org/nova/pike/admin/cpu-topologies.html
[21] https://kubeless.io/
[22] https://knative.dev/
[23] https://kubernetes.io/docs/concepts/overview/components/
[24] https://www.openstack.org/software/
[25] https://docs.openstack.org/qinling/latest/

Published by: The Mattingley Publishing Co., Inc.