A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms

Conference Paper · December 2017

DOI: 10.1109/CloudCom.2017.15

4 authors, including:

Theodore Gerard Lynn
Dublin City University
285 PUBLICATIONS **2,997** CITATIONS

SEE PROFILE

Pierangelo Rosati
University of Galway
119 PUBLICATIONS **1,065** CITATIONS

SEE PROFILE

Vincent C. Emeakaroha
Munster Technological University
46 PUBLICATIONS **1,301** CITATIONS

SEE PROFILE

2017 IEEE 9th International Conference on Cloud Computing Technology and Science

A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms

Theo Lynn, Pierangelo Rosati, Arnaud Lejeune
Irish Center for Cloud Computing and Commerce
Dublin City University, Ireland
{theo.lynn, pierangelo.rosati, arnaud.lejeune}@dcu.ie

Vincent Emeakaroha
Department of Computer Science
Cork Institute of Technology, Ireland
vincent.emeakaroha@cit.ie

***Abstract*— In line with cloud computing emergence as the dominant enterprise computing paradigm, our conceptualization of the cloud computing reference architecture and service construction has also evolved. For example, to address the need for cost reduction and rapid provisioning, virtualization has moved beyond hardware to containers. More recently, serverless computing or Function-as-a-Service has been presented as a means to introduce further cost-efficiencies, reduce configuration and management overheads, and rapidly increase an application's ability to speed up, scale up and scale down in the cloud. The potential of this new computation model is reflected in the introduction of serverless computing platforms by the main hyperscale cloud service providers. This paper provides an overview and multi-level feature analysis of seven enterprise serverless computing platforms. It reviews extant research on these platforms and identifies the emergence of AWS Lambda as a *de facto* base platform for research on enterprise serverless cloud computing. The paper concludes with a summary of avenues for further research.**

Keywords—Serverless Computing; Function-as-a-Service; FAAS; AWS Lambda; Google Cloud Functions; Azure Functions; IBM OpenWhisk; Iron.io; Auth0 Webtask; Gestal Laser.

I. INTRODUCTION

Cloud computing is now widely accepted as the dominant paradigm in computing. By 2020, 67% of all enterprise IT infrastructure and software spending is predicted to be on cloud-based offerings [1]. Fundamentally, cloud computing remains, as originally defined, "[…] a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [2, p. 2]

While this standard definition of cloud computing has remained the same, the search for increased sharing, improvements in the speed of provisioning, and reduction in management effort has been driven by the emergence and dominance of hyperscale cloud service providers (CSPs) operating warehouse-scale machines [3]. Due to the scale of investment and operations of such CSPs, research suggests that there are relatively few of them [4]. Reference [4] suggests that 24 such CSPs operated 295 data centers in 2015 and that they will represent 47% of all installed datacenter servers and will account for 83% of the public cloud server installed base (86% of public cloud workloads).

In addition and as a result of scale in several orders of magnitude than traditional datacenters, hyperscale datacenters are characterized by their high levels of virtualization and standardization. The role of virtualization and specifically the pooling of resources evolved dramatically in recent years. Figure 1, as adapted from [5], visualizes this evolution. Initially, cloud datacenters made use of virtualization as a means of greater software and service consolidation on servers thereby greatly reducing underutilization and increasing manageability. In this first phase of sharing evolution, virtualization enabled the sharing of common hardware. In this scenario, although multiple virtual machines (VMs) may run on a server, each VM still runs a full copy of an operating system (OS). Containerization evolved resource-sharing further through OS-level virtualization. Containers hold all the components necessary to run a specific software program and a minimal subset of an OS. As a result, access to physical resources is constrained and containers are less resource-intensive than VMs. Furthermore, new services can be provisioned faster than VMs.

Despite the efficiencies and increased provisioning speeds introduced by virtualization and containerization, further improvements are limited by the management of underlying infrastructural components, in this case, servers. Serverless computing envisages a model of computing whereby effectively all resources are pooled including hardware, operating systems and runtime environments. Reference [6, p.1] defines serverless computing as "a software architecture where an application is decomposed into 'triggers' (events) and 'actions' (functions), and there is a platform that provides a seamless hosting and execution environment." The developer only concerns itself with relatively lightweight, single purpose stateless functions that can be executed on-demand, typically through an API, without consuming any resources until the point of execution. As the developer does not pay for resources until they are executed, it is therefore referred to as 'serverless' [7]. Similarly, responsibility for datacenter management, server management and the runtime environment are now transferred to the CSP [8]. When compared with other cloud service models, serverless computing introduces a higher separation of concerns from the CSP and the developer, which in effect transfers more responsibility to the CSP while reducing complexity and maintenance overhead. This separation of concerns combined with the lightweight nature of functions dramatically reduces the cost of deployment. This inherent cost attractiveness is further enhanced by new business models e.g. pricing at the level of execution runtime for computer code rather than how long an instance is running [9].

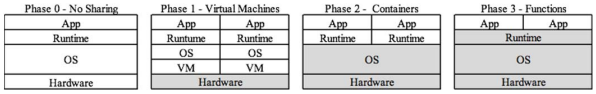

Fig. 1. Evolution of Sharing (adapted from [5]). Gray layers are shared.

Some of the popular use cases for serverless cloud computing include event processing, API composition, API aggregation to reduce API calls, and flow control for issue tracking [10]. The market for such services is not insignificant; it is forecast to grow from US$1.88 Billion in 2016 to US$7.72 Billion by 2021, at an estimated compound annual growth rate (CAGR) of 32.7% [11]. Despite the size of

2330-2186/17 $31.00 © 2017 IEEE
DOI 10.1109/CloudCom.2017.15

the market, research on serverless cloud computing and Function-as-a-Service is still at a nascent level reflecting the maturity of the technology. Only one survey article [10] reviewing enterprise platforms was identified in our literature search on enterprise serverless computing platforms. This paper addresses a gap in the literature by (i) updating, extending and providing a more detailed high-level technical comparison of the current enterprise serverless computing platforms and (ii) presenting indicative use cases cited by these platforms and those use cases featured in extant research.

The remainder of the paper is organized as follows: Section II introduces the enterprise serverless cloud computing platforms studied and proposed use cases by such vendors. Section III presents a multi-level feature analysis of the selected platforms. Section IV provides an overview of related work on serverless computing and Function-as-a-Service with a specific emphasis on the platforms identified. The paper concludes with a discussion of key findings and avenues for future research.

II. Enterprise Serverless Cloud Computing Platforms

Seven enterprise serverless cloud computing platforms were identified for the purpose of this review, namely: AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Bluemix OpenWhisk, Iron.io Ironworker, Auth0 Webtask, and Galactic Fog Gestal Laser.

A. AWS Lambda

AWS Lambda is Amazon Web Service's serverless computing compute service built on AWS various cloud offerings. Cited trigger events included image uploads, in-app activities, website clicks, outputs from connected devices, or other custom requests. *Ab initio*, Amazon Web Services emphasized that AWS Lambda would eliminate the need to provision or manage virtual servers and auto-scaling across multiple jurisdictions (Availability Zones). Amazon Web Services cited a number of use cases including data processing (real-time file processing, real-time stream processing, and Extract Trasform and Load - ETL) and serverless backends (IoT, Mobile and Web). Cited high profile users include Netflix (transcoding, monitoring, disaster recovery, and compliance), SPS Commerce (information processing), Earth Network (sensor data detection, monitoring and prediction), Vidroll (real-time ad bidding), Localytics (stream analytics), Seattle Times (image resizing), Zillow (real-time mobile metrics), Bustle (mobile and web backend) and Major League Baseball Advanced Media (data analysis, player and game metrics).

B. Microsoft Azure Functions

Microsoft Azure Functions was made available in limited release in March 2016 and on general release in November 2016. It was designed to extend the existing Azure application platform with capabilities to implement code triggered by events occurring in Azure or third-party service as well as on-premises systems. Microsoft cite the following use cases for Azure Functions – time based processing (Cron workloads), Azure triggers, Software-as-a-Service (SaaS) event processing, mobile backends, real-time stream processing (IoT), and real-time bot messaging. Cited high profile users include Accuweather (Cron workloads) and Plexure (software auto-scaling).

C. Google Cloud Functions

Google released Google Cloud Functions relatively quietly in February 2016 and it remains in beta at writing. Designed primarily for Google Cloud services, Google cite a number of specific use cases for Google Cloud Functions including mobile backend, APIs and microservice development, data processing/ETL, webhooks (for responding to third party triggers) and IoT. Only one cited high profile user could be identified – Meetup (service integration).

D. IBM Bluemix OpenWhisk

IBM Bluemix OpenWhisk is IBM's serverless cloud computing platform derived from the Apache OpenWhisk open source project to which IBM was a major contributor. It was released for general use in December 2016. Common use cases in OpenWhisk documentation include microservices, web, mobile and API backends, IoT, and data processing. It also suggests it can be used in conjunction with cognitive technologies (e.g. Alchemy and Watson) and messaging systems (e.g. Kafka and IBM Messaging Hub). While IBM provide links to numerous proof-of-concepts, no high profile users could be identified. For example, Skylink is a proof-of-concept that uses IBM Bluemix OpenWhisk with other services to analyze and tag in real-time images captured by a drone. IBM emphasize Docker container integration as a differentiating point from AWS Lambda and Google Cloud Functions.

E. Iron.io Ironworker

Iron.io describe Ironworker as an enterprise job processing system for building job-based asynchronous software. At its core, it is a serverless application platform for primarily scaling Docker-based workloads on any cloud. Ironworker has been available in various forms since 2014. Iron.io is noteworthy as it is a cloud-independent platform and thus exists on top of other clouds. Iron.io cites three primary use cases - data processing, file processing, and ETL. At a more granular level, it proposes serverless computing as a solution for job scheduling (as a replacement for Cron), job priorities, webhooks, pull queues, push queues and long polling, and failure handling. High profile users cited include Bleacher Report (near real-time news alerts), Untappd (batch processing), and HotelTonight (ETL). Iron.io also list Twitter, Google, Whole Foods Market, amongst others as customers without giving specifics on how their software is used by these customers and to what extent. Ironworker is also available on-premise which provides additional opportunities for researchers.

F. Auth0 Webtask

Webtask, the webtask.io and Auth0 Extend are serverless compute services offered by Auth0, a leading provider of authentication solutions. In March 2017, a Webtask editor was released by Auth0 to support serverless applications with a primary focus on the Node.js market. As well as Cron workloads, Auth0 provide a number of templates for integrating Webtask with common services like Stripe, Slack, Sendgrid, Github, Facebook etc. The primary use cases cited include Webtask as a code sandbox and as a webhook. Webtask was followed by the release of Auth0 Extend in May

2017, a serverless extensibility platform for SaaS with an emphasis on web hooks and authentication and identity management. Cited Auth0 Extend users include Stamplay, Meteor Development Group, and Graphcool, although no specific use cases for these customers are identified.

G. Galactic Fog Gestal Laser

Laser, short for Lambda Application Server, is the Galactic Fog's serverless service. The service was released in April 2017, as part of the fourth release of the Gestal platform, which also includes a container-as-a-service (CaaS) abstraction layer and an enterprise integration framework with policy management and security capabilities. Laser is a high-performance, low-latency, serverless engine which supports most common programming languages, and is characterized by high scalability. No specific use cases have been identified.

III. Enterprise Serverless Cloud Computing Platform Feature Matrices

Reference [10] suggests the distinguishing characteristics of serverless platforms are cost, performance and limits, programming languages, programming model, composability, deployment, security and accounting, monitoring and debugging. While [10] suggests that developers be aware of these properties when choosing a platform, they do not compare platforms in their survey against these characteristics. In our review, we found, due to (i) the early stage of the market, (ii) the release status of the platforms, and (iii) the size of the vendors, that the detail in and clarity of documentation can vary. To address these issues, we developed a list of commonly cited features based on reviewing vendor websites and platform documentation, extant literature and specialist websites offering comparisons of one or more platforms. Table I presents a comparison of general and high-level technical features based on the following items:

- Release status: the status of the platform i.e. alpha, beta or General Availability (GA).
- Runtime Environment: whether the runtime environment is open source, proprietary and/or public.
- Scalability: automatic scaling up and down or manual intervention required.
- Clouds: Cloud storage service associated on which data will be stored.
- Programming languages: the main programming languages supported by the platform.
- Authentication: the rights to access the function.
- Integration: services with native support and API that can be used/called within the functions' execution.
- Event Providers: the event providers and associated types supported.
- Orchestration: support for orchestration with other applications.
- Versioning: support for versioning using cloud repositories.
- Log Management: integrated tool to monitor functions' activity.
- Dependencies: how to install and include dependencies to a function.
- Maximum number of Functions: the total number of functions that can be deployed per project.
- Maximum deployment size: the maximum size of a single function deployment.
- Maximum concurrent functions: the maximum concurrent invocations of a single function.
- Maximum function duration - the maximum amount of time a function can run before it's forcibly terminated
- Maximum build time - the maximum time allowed for all builds. Function builds happen at deploy time.
- Default function calls per second – the default number of function calls in a second. If exceeded, all functions may be paused until the next quota period.
- Default GHz-seconds per second - the number of GHz-seconds consumed per second.

Table II presents a list of the information source associated with each feature included in our comparison.

TABLE I ENTERPRISE SERVERLESS CLOUD COMPUTING PLATFORM GENERAL AND HIGH LEVEL TECHNICAL FEATURE MATRIX

Features	**AWS Lambda**	**Microsoft Azure Function**	**Google Cloud Function**	**IBM Bluemix OpenWhisk**	**Iron.io Iron Functions**	**Auth0 WebTask**	**Galactic Fog Gestal Laser**
1. Release Status	GA	GA	Beta	Beta	Alpha	GA	GA
2. Runtime Environment	Public	Open Source	Public	Open Source	Open Source	Public	Open Source
3. Scalability	Automatic	Manual/Automatic	Automatic	Automatic	Manual	Unavailable	Automatic
4. Clouds	AWS	Microsoft Azure	Google Cloud	IBM Bluemix	Unknown	Unknown	Unknown
5. Programming Languages	Node.js, Java, C#, Python	C#, F#, JavaScript, Node.js, Window Scripting1, PowerShell1, Bash1, PHP1, Python1	JavaScript	JavaScript, Swift	PHP, Python, Ruby, Node.js, Java, .Net, Go, Scala, binary executables	Node.js	Java, Scala, JavaScript, Go, Ruby, Python and NET Core
6. Authentication	AWS Account root user, IAM user, IAM roles	Azure App Service Authentication	IAM roles	Unknown	Unknown	Auth0	Gestalt Entitlements
7. Integrations	AWS services (Through AWS SDK)	Azure Cosmos DB, Azure Event Hubs, Azure Mobile Apps (tables), Azure Notification Hubs, Azure Service Bus (queues and topics), Azure Storage (blob, queues, and tables), GitHub (webhooks), On-premises (using Service Bus), Twilio (SMS messages)	Google Cloud Storage, Google Cloud BigTable, Google Cloud Spanner, Google Cloud Datastore, Google Cloud Machine Learning Engine, Google Cloud Natural Language API, Google Cloud Speech API, Google Cloud Translation API, Google Cloud Vision API, Google Cloud BigQuery, Google Cloud Pub/Sub, Google Stackdriver Logging, Google Firebase	IBM Cloudant, IBM Message Hub, IBM Bluemix Push Notifications, IBM Watson Translator, IBM Watson Speech to Text, IBM Watson Text to Speech, Weather Company Data, Slack, GitHub, WebSockets, Serverless Framework	Unknown	Unknown	Native API Gateway (Kong)
8. Event Providers	Amazon Web Services (Refer to the reported in Table 2 for the full list), Scheduled events, HTTPS (Amazon API Gateway)	Scheduled events, HTTP (REST or Webhook), Azure Storage (blob, queues), Azure event Hubs, Azure Service Bus (Queues and topics)	HTTP, Google Cloud Storage, Google Cloud Pub/Sub, Google Firebase, Google Stackdriver Logging	HTTP, Alarms, Github Webhooks	HTTP	HTTPS	REST API

TABLE I ENTERPRISE SERVERLESS CLOUD COMPUTING PLATFORM GENERAL AND HIGH LEVEL TECHNICAL FEATURE MATRIX (CONTINUED FROM PREVIOUS PAGE)

Features	AWS Lambda	Microsoft Azure Function	Google Cloud Function	IBM Bluemix OpenWhisk	Iron.io Iron Functions	Auth0 WebTask	Galactic Fog Gestal Laser
9. Orchestration	AWS Step Functions	Azure Logic Apps	Unknown	Unknown	Kubernetes, Docker Swarn	Unknown	Unknown
10. Versioning	Versions and aliases	Cloud Source Repositories - GitHub, Visual Studio Team Services	Cloud Source Repositories - Github, Bitbucket	Unknown	Unknown	Github, Bitbucket	Unknown
11. Log Management	CloudWatch	Azure App Service	Google Stackdriver	IBM Bluemix OpenWhisk Dashboard	Logspout	Webtask Editor	Unknown
12. Dependencies	Deployment Package	NuGet, NPM	Deployment Package	Deployment Package	Unknown	npm	Unknown
13. Max number of functions	1,500	Unknown	1,000	Unknown	Unknown	Unknown	Unknown
14. Max deployment size	50MB (compressed) for Lambda function deployment package size 75GB for Total size of all the deployment packages that can be uploaded per region 250 MB (uncompressed) Size of code/dependencies that you can zip into a deployment package	Unknown	100MB (compressed) for sources 500MB (uncompressed) for sources and modules	48MB	Unknown	Unknown	Unknown
15. Max concurrent functions	10,002	Unknown	10,002	1,000	Unknown	1	1,000
16. Max function duration	300 seconds	600 sec	540 seconds	0.1 - 300 seconds3	Unknown	30 seconds	Unknown
17. Max memory per function	1.5 GB	1.5 GB	2 GB	512 MB	Unknown	Unknown	Unknown

Experimental. 2. Can be increased on request. 3. 60 seconds is default.

TABLE II INFORMATION SOURCES

Platform	Source(s)	Feature
AWS Lambda	https://aws.amazon.com/lambda/	1-5
	http://docs.aws.amazon.com/lambda/latest/dg/welcome.html	6, 7, 8, 12, 13-17
	https://aws.amazon.com/lambda/faqs/	10-11
	https://aws.amazon.com/step-functions/	9
Microsoft Azure Function	https://azure.microsoft.com/en-us/services/functions/	1, 3-5
	https://github.com/azure/azure-webjobs-sdk-script	2
	https://docs.microsoft.com/en-us/azure/azure-functions/	6-8,11, 12, 16
	https://azure.microsoft.com/en-us/pricing/details/functions/	17
Google Cloud Function	https://cloud.google.com/functions/	1-5
	https://cloud.google.com/functions/docs/	6-8, 11-12
	https://cloud.google.com/functions/quotas	13-17
	https://cloud.google.com/functions/?hl=fr	10
IBM Bluemix OpenWhisk	https://developer.ibm.com/openwhisk/	1-5
	https:/agithub.com/apache/incubator-openwhisk/blob/master/docs/catalog.md	7-8
	https://console.bluemix.net/docs/openwhisk/openwhisk_monitoring.html#monitoring-your-openwhisk-activity-with-the-openwhisk-dashboard	11
	https://console.bluemix.net/docs/openwhisk/openwhisk_actions.html#openwhisk_actions	12
	https://console.bluemix.net/docs/openwhisk/openwhisk_reference.html#openwhisk_reference	14-17
Iron.io IronFunctions	https://github.com/iron-io/functions	1-2, 5, 8
	https://github.com/iron-io/functions/blob/master/docs/operating	3, 11
	https://github.com/iron-io/functions/blob/master/docs/README.md#for-developers	9
Auth0 WebTask	https://webtask.io	1-2, 5
	https://webtask.io/docs/	6, 8, 10-12
	https://webtask.io/pricing	15
	Direct information request	3-16
Galactic Fog Gestal Laser	http://www.galacticfog.com/product/	1, 15
	https://github.com/GalacticFog	2
	http://docs.galacticfog.com/services/gestalt-laser/	3, 5
	http://docs.galacticfog.com/security/entitlements/	6

Note: Information source were last accessed on 10 July 2017.

IV. REVIEW OF RELATED REFERENCES

While there is an established but relatively limited research base on serverless computing [19, 20, 21, 22, 23], research on event-driven cloud services, and enterprise serverless cloud computing platforms in particular, is still at a nascent stage reflecting the market entry points of the various platforms.

Unsurprisingly given Amazon Web Services' position in the overall cloud computing market and early mover status in enterprise serverless cloud computing, it is most prominent in academic literature. Reference [5] provides an overview of the AWS Lambda programming model and its advantages over server-based models. A comparison between AWS Lambda and AWS Elastic Beanstalk suggest AWS Lambda had a significant response time advantage, required no configuration for scaling, had faster start-up times, and was simpler to use [5]. AWS Lambda was found to be a feasible solution for a data-handling back-end for batch and real-time big data processing in smart city and sensor network use cases [24]. In both batch and real-time processing, local tests were recommended to reduce errors and capture exceptions [24]. In addition, care needs to be taken to use appropriate billing to optimize billing processes, e.g. per-minute billing, and that where dependencies exist between services, that appropriate authentication and rights are available [24]. Similarly, [25] demonstrate through a prototype implementation, PyWren, that AWS Lambda is general enough to implement a number of distributed computing models efficiently including BSP-style applications. Reference [26] presents two implementations using AWS Lambda – a web application and a media management service. The former suggests significant changes to application development and structure for efficient deployment in a serverless context

including greater emphasis on optimizing startup-times, separation of application logic and data layers, and the ability of the data layer to handle many concurrent and short-lived connections [26]. The media management service implementation suggests that the separation of concerns inherent in serverless cloud computing resulted in superior performance without the need to consider how to manage scaling, system resource constraints, queuing, cluster management and more [27]. Reference [28] describes a cloud-based video processing framework capable of running general-purpose parallel computations using AWS Lambda with low latency and an implementation for a video encoder for fine grained parallelism without impacting compression efficiency.

Given the cost advantages heralded by enterprise serverless cloud computing platforms, it is no surprise the researchers are exploring the modelling and managing of deployment costs particularly given the findings in [5] and [24]. CostHat is an approach to modelling deployment costs of micro-services deployed on conventional Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) clouds and through Function-as-a-Service, in this case AWS Lambda [28]. Reference [29] explores the economics of resilient cloud services and suggests that serverless operating environments, and AWS Lambda in particular, are attractive in terms of security, cost and architecture. This is due to a smaller attack surface, execution-time-based billing, and a loosely coupled application paradigm [29]. More recently, research by [30] suggests that a web application using AWS Lambda, exclusively designed to deploy micro-services at a more granular level (per HTTP request/function), allows companies to reduce their infrastructure costs by up to 77.08% when compared to a monolithic architecture or a micro-service architecture operated by the cloud customer.

While other platforms are referenced in academic papers as alternatives to AWS Lambda, no discrete academic publications using Azure Functions, Google Cloud Functions, Iron.io Ironworker, Auth0 Webtask or Galactic Fog Gestal Laser could be identified. Two papers were identified on IBM Bluemix OpenWhisk authored largely by IBM authors. [31] presents three demonstrators for IBM Bluemix OpenWhisk. They demonstrate event-based programming triggered by weather forecast data, Apple WatchOS2 application data, and speech utterances. Similarly [32] presents a demonstrator of a chatbot using IBM Bluemix OpenWhisk that calls on IBM Watson services including news, jokes, dates, weather, music tutor, and an alarm service.

V. Discussion and Conclusions

Enterprise serverless computing is at an early stage of conceptualization, both commercially and academically. Vendors and research suggests that it is more cost-efficient, simpler to use, and more secure [5, 29, 30] however such propositions are based on a relatively small number of use cases and published studies. Our review of cited use cases by vendor and indicative customers in Section III, and published studies in Section IV, suggest that serverless cloud computing can be applied to a wide range of general and specific use cases currently unaddressed by the academic community. In addition to addressing the gaps in proposed and actual use cases, there are opportunities for novel research in non-cloud environments. For example, [33] extend the serverless paradigm to the edge of the network, in what they call "deviceless edge computing." Even at this nascent stage, this presents a number of challenges to the existing serverless cloud computing paradigm including resource pooling and elasticity, security and provisioning and management at scale where a high tolerance to network disruption is needed. Furthermore, there is a potentially rich vein of research in a wide range of comparative research both (i) between enterprise platforms and (ii) between enterprise and open-source projects, including Apache OpenWhisk [34], OpenLambda [35], IronFunctions [36], Fission [37], Gestalt [38] and Dithen [39].

In addition to extending this new paradigm to additional use cases, extant literature suggests that there are a number of outstanding challenges to be addressed. Reference [10] categorizes these in to two major categories:

- System-level challenges: cost, pricing, cold starts, resource limits, security, predictive scaling without application-level knowledge, multi- and hybrid cloud, and access to legacy cloud and non-cloud systems from serverless code running in serverless platforms.
- Programming model and DevOps challenges: serverless tools including monitoring and debugging tools, declarative deployment, higher level developer capabilities in IDEs (e.g. refactoring functions), composability and maintenance of functions, long-running functions, accommodating applications that require state in stateless serverless functions, concurrency semantics, recovery semantics, and code granularity.

Similarly, Reference [26] suggests six main categories of systems aspects for future research, which are generalizable: resource allocation and balancing, pricing, scalable scheduling, distributed storage, launch overhead optimization, and support for heterogeneous hardware e.g. GPUs and FPGAs.

Serverless cloud computing presents the opportunity for researchers to revisit the major themes and topics of research in cloud computing from a new perspective not least migration, interoperability, optimization, virtualization management, fault tolerance, simulation and much more. This paper presents the major enterprise serverless cloud computing platforms and their cited use cases as well as a review of extant research on these platforms.

ACKNOWLEDGMENT

The research work described in this paper was supported by the Irish Centre for Cloud Computing and Commerce, an Irish National Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

REFERENCES

[1] IDC, "IDC FutureScape: Worldwide IT Industry 2017 Predictions", IDC #US41883016. MA: IDC, 2016.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing, National Institute of Standards and Technology", Version 15, vol. 53, no. 6, p. 50. MD: NIST, 2009.

[3] L.A. Barroso, J. Clidaras and U. Holzle. "The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines," Synthesis Lectures on Computer Architecture, vol. 8(3), pp. 1-154, 2013.

[4] Cisco, "Cisco global cloud index: Forecast and methodology, 2015-2020,". CA: Cisco Public. 2016.

[5] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Serverless Computation with OpenLambda", Elastic, 60, 80, 2016.

[6] A. Glikson, S. Nastic, and S. Dustdar, "Deviceless Edge Computing: Extending Serverless Computing to the Edge of the Network," In Proceedings of the 10th ACM International Systems and Storage Conference (SYSTOR), p. 28, 2017. ACM.

[7] R. Koller and D. Williams, "Will Serverless End the Dominance of Linux in the Cloud", 16th Workshop on Hot Topics in Operating Systems (HTOS XVI), 2017.

[8] M. Yan, P. Castro, P. Cheng and V. Ishakian. "Building a Chatbot with Serverless Computing," In Proceedings of the 1st International Workshop on Mashups of Things and APIs, p. 5. ACM.

[9] A. Eivy, "Be Wary of the Economics of "Serverless" Cloud Computing," IEEE Cloud Computing, vol. 4(2), p.6, 2017.

[10] 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". arXiv preprint arXiv:1706.03178. 2017.

[11] Market and Markets. "Function-as-a-Service Market by User Type (Developer-Centric and Operator-Centric), Application (Web & Mobile Based, Research & Academic), Service Type, Deployment Model, Organization Size, Industry Vertical, and Region - Global Forecast to 2021," Pune: Market and Markets. 2017.

[12] Amazon Web Services, "AWS Lambda Developer Guide", Accessed on 10 July 2017 http://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf

[13] Microsoft, "Azure Functions Documentation", Accessed on 10 July 2017 https://docs.microsoft.com/en-us/azure/azure-functions/

[14] Microsoft, "Azure Functions Pricing", Accessed on 10 July 2017 https://azure.microsoft.com/en-us/pricing/details/functions/

[15] Google, "Google Cloud Functions Documentation". Accessed on 10 July 2017 https://cloud.google.com/functions/

[16] IBM, "Getting Started with OpenWhisk", Accessed on 10 July 2017 https://console.bluemix.net/docs/openwhisk/index.html

[17] Iron.io, "IronWorker Reference", Accessed on 10 July 2017 http://dev.iron.io/worker/

[18] WebTask, "Getting Started", Accessed on 10 July 2017 https://webtask.io/docs/101

[19] T.E. Anderson, M.D. Dahlin, J.M. Neefe, D.A. Patterson, D.S. Roselli, and R.Y. Wang, "Serverless Network File Systems", ACM SIGOPS Operating Systems Review, vol. 29, no. 5, p. 109.

[20] M. King, "Breaking the server and data communications barrier with serverless guaranteed quality of service (GQoS) compliant communications." In Proceedings of the First IEEE International Conference on Peer-to-Peer Computing, p.36 2001.

[21] V.M. Baskaran, Y.C. Chang, J. Loo and L. Wong. "Software-Based Serverless Endpoint Video Combiner Architecture For High-Definition Multiparty Video Conferencing," Journal of Network and Computer Applications, vol. 36, p.336. 2013.

[22] E.S. Jung and R. Kettimuthu, "High-Performance Serverless Data Transfer over Wide-Area Networks," In the Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), p. 557. 2015.

[23] D. Saha, "Service Mining from Legacy Database Applications," In Proceedings of the IEEE International Conference on Web Services (ICWS) 2015, p. 448. 2015. IEEE.

[24] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S.S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," In Proceedinsg of the IEEE International Conference on Big Data (Big Data) 2015, pp. 2785-2792. 2015. IEEE.

[25] G. McGrath, J. Short, S. Ennis, B. Judson and P. Brenner. "Cloud Event Programming Paradigms: Applications and Analysis," In Proceedings of the 9th IEEE International Conference on Cloud Computing (CLOUD), pp. 400-406. IEEE.

[26] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Ocuupy the Cloud: Distributed Computing for the 99%," arXiv preprint arXiv:1702.04024. 2017.

[27] S. Fouladi, R. Wahby, B. Shacklett, K. Balasubramaniam, W. Zeng, R. Bhalerao, A. Sivaraman, G. Porter, and K. Winstein, "Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads," In NSDI, pp. 363-376 2017.

[28] P. Leitner, J. Cito, and E. Stöckli, "Modelling and managing deployment costs of microservice-based cloud applications," In Proceedings of the 9th International Conference on Utility and Cloud Computing, p.165. 2016.

[29] B. Wagner and A. Sood. "Economics of Resilient Cloud Services," In the Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), p. 368. 2016.

[30] M. Villamizar, O. Garcés, L. Ochoa, H. Castro, L. Salamanca, M. Verano, R. Casallas, S. Gil, C. Valencia, A. Zambrano and M. Lang, M., "Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures," Service Oriented Computing and Applications, vol. 11(2), p233. 2017.

[31] I. Baldini, P. Castro, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, and P. Suter, "Cloud-native, event-based programming for mobile applications," In the Proceedings of the International Workshop on Mobile Software Engineering and Systems, pp. 287. 2017.

[32] M. Yan, P. Castro, P. Cheng, and V. Ishakian, "Building a Chatbot with Serverless Computing," In the Proceedings of the 1st International Workshop on Mashups of Things and APIs, p. 5. 2016.

[33] E. de Lara, C.S. Gomes, S. Langridge, S.H. Mortazavi, and M. Roodi. "Hierarchical Serverless Computing for the Mobile Edge," In Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC), pp. 109-110. 2016. IEEE.

[34] Openwhisk. "Homepage", Accessed on 10 July 2017 http://openwhisk.incubator.apache.org/

[35] OpenLambda, "Background", Accessed on 10 July 2017 https://open-lambda.org/

[36] OpenIron, "Open Source Serverless Computing", Accessed on 10 July 2017 http://open.iron.io/

[37] Fission, "Homepage", Accessed on 10 July 2017, http://fission.io/

[38] Gestalt, Homepage, Accessed on 10 July 2017 http://galacticfog.github.io/

[39] J. Doyle, V. Giotsas, M.A. Anam, and Y. Andreopoulos, "Dithen: A Computation-as-a-Service Cloud Platform For Large-Scale Multimedia Processing," IEEE Transactions on Cloud Computing. 2016.