IJRECE VOL. 7 ISSUE 1 (JANUARY- MARCH 2019) ISSN: 2393-9028 (PRINT) | ISSN: 2348-2281 (ONLINE)

A Serverless Model for Scheduling Scientific Work Flows

Salini Suresh[1], L. Manjunath Rao [2]

[1]*Global Institute of Management Sciences*

[2]*Dr. Ambedkar Institute of Technology*

***Abstract* -** Serverless computing is an emerging model that excludes the apprehensions linked to server provisioning and management. This computing paradigm employs isolated, stateless functions that are event triggered to handle definite tasks. In recent times, the Function-as-a-Service (FaaS) has attained significant importance in scientific and high performance computing. Serve less computing is a potential model for scientific workflows as challenges in resource provisioning and auto scaling are naturally addressed here. Even though serverless computing delivers a substantial effect in scheduling process of scientific workflows, prevalence of concurrent tasks, dependency constraints and inter data dependency should be considered as significant parameters for the scheduling process. To this end we propose an approach for scientific workflow scheduling that aims to minimize the execution time of entire workflow and optimal resource usage in server less environments.

***Keywords* -** Server less computing, scientific work flows, FaaS, execution time,

I. INTRODUCTION

Server less computing is an evolving cloud computing service model that delivers application deployment architecture as Function as a service (FaaS) where the service provider is accountable for Infrastructure, resource provisioning and scaling according to the application logic. The ownership and management of the servers are abstracted from the clients and task specific functions serve as event handler while the clients are charged only for the time of function execution. The sever less computing paradigm and FaaS has gained importance in business as well as scientific and engineering research domains. The rest of the manuscript is organised as follows: Section II enumerates an over view of server less computing. Section III details how server less computing can be used for scientific and academic research. In Section IV problem description is discussed .The proposed approach is elaborated in section V. The result and analysis is discoursed in Section VI.

II. OVER VIEW OF SERVER LESS COMPUTING

The emergence of server less platforms enables ephemeral, stateless computation and event triggered application instances that can scale up and down robustly and levy the users with minuscule granularity [1]. In a traditional cloud computing model any service oriented application necessitates a Virtual machine hosting and application residency in the virtual machine.

In IaaS model of cloud the user have the complete control over the operating system with even root access privileges, where as in Function as service set free the user from configuration and management of server resources.

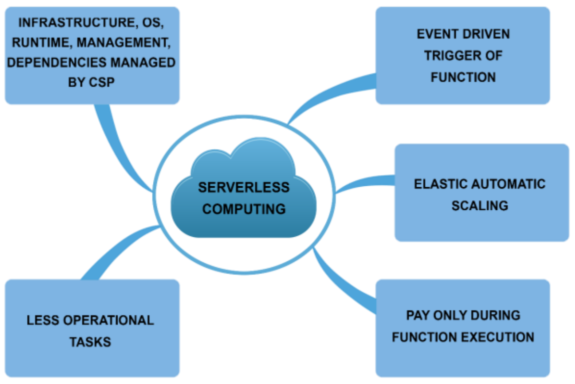
INFRASTRUCTURE, OS, RUNTIME, MANAGEMENT, DEPENDENCIES MANAGED BY CSP
EVENT DRIVEN TRIGGER OF FUNCTION
SERVERLESS COMPUTING
ELASTIC AUTOMATIC SCALING
LESS OPERATIONAL TASKS
PAY ONLY DURING FUNCTION EXECUTION

Figure 1: Serverless Computing

In customary cloud computing model it necessitates the application hosting on virtual machine and user need to pay although the application hosting irrespective of the service is availed or not. A server less architecture excludes the concern of managing the servers, application logic and thus decreasing set-up as well as provisioning costs. These Architectures enhances productivity in software development and makes whole cycle essentially responsive .The Cloud service providers are also advancing to standardize development environments to persuade the usage of server less architectures

Even though cloud virtualization and containerization are effective and has led to better provisioning in response to dynamic requirements, the management of the infrastructure is still a concern for the user. . Server less computing visualizes computing model that efficiently pools all resources like hardware, operating systems and runtime environments [2].These platforms provide execution and hosting environments that are seamless and lightweight stateless functions which do not employ any resources till execution are delivered on demand through an API.

The Cloud providers like Amazon, Azure, Google, and IBM established and positioned server less computing platforms Amazon AWS Lambda, Azure Functions and Google Cloud Functions respectively. The rapid scalability and dynamic resource allocation in these platforms are made possible using dedicated containers with own pool of resources [4].The function instances are invoked upon requests ,serves the request and turns idle after the maximum execution time .

Server less programming enables the application programmers to convert programmed functions into readily available cloud services [4]. The application instances are launched without requiring the running of application every time which contrasts with existing execution models where applications runs continuously on the server to service a client request and billed even though the server application is futile

The areas where FaaS is gaining dominance are IoT (Internet of Things) applications, web application as well as scientific and HPC (High Performance Computing) applications that demand dynamic execution environments [5].The runtime environments of FaaS can be largely divides as those which facilitate commercial services like AWS Lambda [6], Azure Functions and Google Cloud Functions [7] or open source like open whisk [8]. Lambda@Edge [9] is server less implementation on the edge (closer to the user).

III. SERVER LESS COMPUTING FOR SCIENTIFIC AND ACADEMIC RESEARCH.

The server less computing and FaaS has major significance in scientific and engineering research .The concerns that arise in scientific data management and high performance work loads like reproducibility, cost and rapid scalability can be effectively addressed using server less implementations. Complex scientific workflows like LIGO [10] and CyberShake [11] that are utilized in fields of astronomy and high energy physics have high levels of precedence constraints and inherently consist of large number of dependent tasks.

FaaS delivers a substantial effect in scheduling process of scientific workflows. During the scheduling of virtual machines due to independency between the tasks some of the machine can be idle. This affects the efficiency of the scheduling process especially in scientific applications with heavy workloads.

In order to manage these workflow management systems are deployed which require setting up and configuration of massive clusters. The precedence constraints and heavy inter task dependency often lead to underutilization of resources. The cloud native sever less and FaaS platforms enable short-lived computations and event initiated applications that can scale up down rapid and make these platform suitable for scientific workflows. Server less computing permits developers to disintegrate large scientific applications into small functions, to enable scaling of application components independently [12].

Hybrid approaches using IaaS and FaaS are also gaining acceptance as it reduces the cost and execution time. Such blends can lead to cost effectiveness as small tasks can be executed as FaaS which also avoid resource wastage and idle time [13].

The occurrence of concurrent tasks in which demand proficient scheduling process for resource allocation is a challenge in scientific work flows. Serve less computing is an effective implementation model for scientific workflows as concerns in resource provisioning and auto scaling are automatically addressed here[14].

A. Models for execution of scientific work flows - The server less architectures for scientific work flow execution are depicted in [15].In the traditional model the workflow runs in an IaaS cloud and the master node that runs the workflow is usually set out in the cloud and accepts the tasks in a queque,and Virtual machines deployed as workers are created on demand. Pegasus [16] and Hyper-Flow [17] follows this traditional model. The queue model spawns cloud functions that realizes the tasks and executed it. This facilitate simultaneous use of cloud functions from many cloud platforms. In decentralized model for example Flowbster [18], every task is administered as separate function triggered using events or other functions.

A work flow model illustrates the connection between different tasks. Directed Acyclic Graph (DAG) and Non Directed Acyclic Graph are utilized for the workflow models. The sequence based workflow models execution of tasks occurs in series while parallel models enables synchronous execution of tasks.

The representation of an application workflows are realized using directed acyclic graph where the node of the graph denotes a specific task and edges denote a precedency constraints[19,20].Server less computing is well suited for high performance computing and research applications with resource hungry jobs in constrained environments. Server less platforms can facilitate the execution of light weight FaaS functions [21].

B. Architecture for work flow execution in server less -

In IaaS cloud platforms the execution of server less workflows are different with regard to the design and scheduling methods. Time constraints and cost effectiveness are important factors to be considered during scheduling of workflows with diverse resource set.

Static and dynamic scheduling approaches are widely implemented in server less models. Static methods allocates the resources for a specific workflow tasks prior to execution whereas dynamic or runtime allocation is carried out if the workflow demands execution in priority .VM provisioning also may be required to service tasks with long duration. A typical workflow execution in server less can be depicted as in Figure 2.

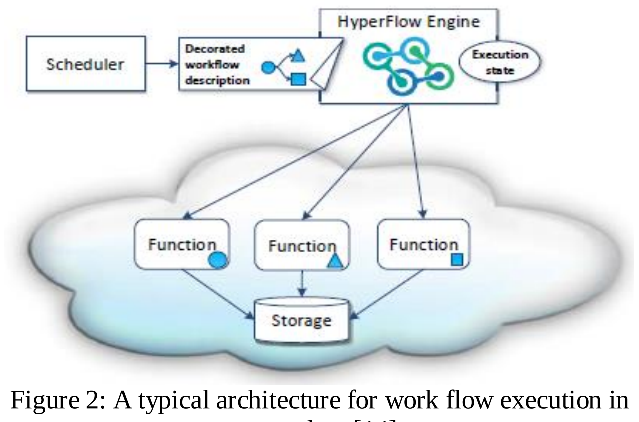

Figure 2: A typical architecture for work flow execution in serverless [14]

More unconventional cloud implementations are being used recently for scientific work flows. Web frameworks like Google App Engine are also used for computing platforms with higher workloads. AWS lambda implementations with python and Globus transfer API [22] are used to monitor high performance workloads in storage [23] .The hybrid approaches are designed to address underutilization of resources during execution of extensive scientific workflows [12].

The compute requirement of scientific domains has significant difference in terms of compute, consumption of network resources and storage requirements. The dissimilar computing tasks form different scientific domains can implement levels of FaaSificastion by in view of function, line of code, instructions as the basic units [5].Fission and kubeless [24] are kubernetes [25] based provisioning framework with support to many programming languages and custom runtimes. Kubeless also offers event triggering using Kafka [26].

IV. PROBLEM DESCRIPTION

The traditional Computing paradigms like cluster and virtual machine approaches has challenges from underutilization of resources in scientific applications with heavy workloads[27].The heterogeneous nature of scientific workflows, high levels of interdependency between the tasks and precedence constraints demands efficient scheduling approaches to reduce the delay in workflow execution and optimize resource usage[14].

V. PROPOSED APPROACH

Any scientific workflow can be represented as a Directed Acyclic Graph (DAG), which is a function of tasks, data and data dependency. Each node of the graph represents a task and edges the data dependency. Due to interdependency of the tasks execution of child tasks cannot occur unless all its parent tasks are completed. Delay in the individual child task execution may delay the execution time of complete workflow.

We propose a scheduling algorithm for scientific workflows in a server less environment. The proposed approach relies on generating a function to map the tasks to the resources and scheduling the tasks based on the dependency to other tasks. Algorithm calculates the make span of the entire workflow and scheduling the process with more number of child processes.

Each tasks is assigned a level, which is a numerical value that is the maximum path length from entry node to task node. Every process will have multiple tasks with many child tasks. The tasks are grouped according to the number of child tasks. Groups with maximum number of child tasks are given priority in execution. The wait time for execution of each tasks is optimized and minimize the execution time of entire workflow with optimal usage of resources.

We define deadline as the maximum completion time for each task at each level.

G_{MC} denotes the group with maximum child process.

G_{MIN} denotes the group with minimum completion time.

r denotes the resource type.

DL_{user} denotes deadline for a user.

DL_l denotes deadline for a level.

FT_{max} denotes Maximum finish time for the task.

FT_{min} denotes Minimum finish time for the task.

ST denotes start time of the task.

Algorithm 1: Server less optimal scheduling algorithm

```
Input: user, deadline, tasks

1. Sort all the tasks based on levels.
2. Group the tasks with maximum number of child tasks.
3. Group the tasks with minimum completion time.
4. If $DL_{user} < FT_{max}$ then no allocation
5. Else $G_{MC} \leftarrow r$.
6. $G_{MC} > FT_{min}$.
7. $G_{MC} \leftarrow r_{++}$
8. If $G_{MIN} < DL_l$ then allocate resources
9. $DL < FT_{max} - ST$ go to step 7.
```

VI. RESULT AND ANALYSIS

A scientific workflow of 30 tasks was executed in serverless platform AWS Lambda by varying the configurations and execution time was evaluated .The prototype of the approach was implemented in same serverless platform for executing the scientific workflow of 30 tasks and execution time was evaluated for functions with 256 MB, 512 MB, 1024 MB and 2048 MB of memory allocated respectively. The comparative analysis shows that proposed approach made the execution time faster by 182%, 11% and 29 % respectively .The execution of function with 2048 MB was slower by 11%.

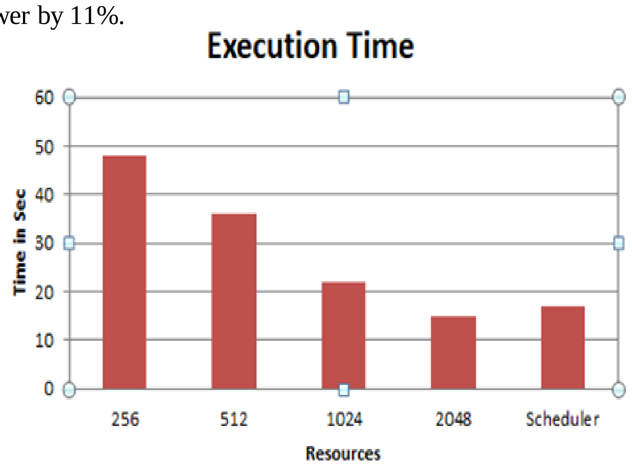

Figure 3: Comparison of execution time

VII. CONCLUSIONS

The paper has proposed an approach for scientific workflow scheduling which intends to minimize the workflow execution time and achieve optimal resource usage in server less environments. Significant decrease in execution time and finish time of the individual tasks were achieved.

VIII. References

[1]. Castro P, Ishakian V, Muthusamy V, Slominski," A. Serverless Programming (Function as a Service)". 2017 IEEE

37th International Conference on Distributed Computing Systems. (ICDCS). 2017. pp. 2658–2659. Doi: 10.1109/ICDCS.2017.305.

[2]. A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms Theo; Pierangelo Rosati ; Arnaud Lejeune ; Vincent Emeakaroha2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) DOI: 10.1109/CloudCom.2017.15.

[3]. Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking Behind the Curtains of Serverless Platforms. In 2018 USENIX Annual Technical Conference (USENIX ATC 18). USENIX Association, 2018

[4]. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., and Muthusamy, V., Rabbah, R., Slominski, Suter, and P.: Serverless Computing: Current Trends and Open Problems. ar_iv:1706.03178 (June 2017)

[5]. Josef Spillner, Cristian Mateos, and David A Monge. 2017. FaaSter, Better, Cheaper: The Prospect of Serverless Scientific Computing and HPC. In Latin American High Performance Computing Conference. Springer, 154–168.

[6]. AWS Lambda: How It Works, September 2017. URL http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction.html.

[7]. Cloud Functions Overview, September 2017. URL https://cloud.google com/functions/docs/concepts/overview.

[8]. About Cloud Functions, September 2017. URL https://console.bluemix.net/docs/openwhisk/openwhisk_about.html#about-cloud- Functions.

[9]. https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html

[10]. Abramovici, A., Althouse, W.E, et. al.: LIGO: The laser interferometer gravitational-wave observatory. Science 256(5055), 325{333 (1992).

[11]. Graves, R., Jordan, T.H., et. al.: Cybershake: A physics-based seismic hazard model for Southern California. Pure and Applied Geophysics 168(3-4), 367{381 (2010).

[12]. Jiang Q., Lee Y.C., Zomaya A.Y. (2017) Serverless Execution of Scientific Workflows. In: Maximilien M., Vallecillo A., Wang J., Oriol M. (eds) Service-Oriented Computing. ICSOC 2017. Lecture Notes in Computer Science, vol 10601. Springer, Cham https://doi.org/10.1007/978-3-319-69035-3_51.

[13]. Omar Alqaryoutia , Nur Siyamb Serverless Computing and Scheduling Tasks on Cloud: A ReviewAmerican Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) ISSN (Print) 2313-4410, ISSN (Online) 2313-4402 © Global Society of Scientific Research and Researchers http://asrjetsjournal.org

[14]. Joanna Kijak, Piotr Martyna, Maciej Pawlik, Bartosz Balis, Maciej Malawski ,Challenges for Scheduling Scientific Workflows on Cloud Functions , IEEE 11th International Conference on Cloud Computing (CLOUD) 2018,**ISSN:** 2159-190 **DOI:** 10.1109/CLOUD.2018.00065.

[15]. M. Malawski, towards serverless execution of scientific workflows Hyper-Flow case study, in: Workflows in Support of Large-Scale Science Workshop, 2016.

[16]. E. Deelman, K. Vahi, G. Juve, M. Rynge,S. Callaghan, P. J. Maechling, R. Mayani, W. Chen,R. Ferreira da Silva, M. Livny, and K. Wenger.Pegasus, a workout management system for science automation. Future Generation Computer Systems, 46:17{35, May 2015.

[17]. B. Balis. HyperFlow: A model of computation, programming approach and enactment engine for complex distributed workows. Future Generation Computer Systems, 55:147{162, Sep 2016.

[18]. https://github.com/occopus/flowbster

[19]. M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds," IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, 2014

[20]. T. Ryan and Y. C. Lee, "Effective Resource Multiplexing for Scientific Workflows," 2015...

[21]. Josef Spillner, Cristian Mateos, and David A Monge. 2017. FaaSter, Better, Cheaper: The Prospect of Serverless Scientific Computing and HPC. In Latin American High Performance Computing Conference. Springer, 154–168.

[22]. https://docs.globus.org/api/transfer.

[23]. M. Malawski, M. Kuzniar, P. Wojcik, and M. Bubak.How to Use Google App Engine for Free Computing. IEEE Internet Computing, 17(1):50{59, Jan 2013.

[24]. https://kubeless.io/

[25]. https://kubernetes.io/

[26]. https://kafka.apache.org.

[27]. Suresh, Salini; Manjunatha Rao, L CCCORE: Cloud Container for Collaborative Research International Journal of Electrical & Computer Engineering (2088-8708). Jun2018, Vol. 8 Issue 3, p1659-1670. 12p.

Salini Suresh is working as Associate Professor, Department of Computer Applications, Global Institute of Management Sciences, Bangalore. She has got 13 years of teaching experience. She has obtained Bachelor of Science from Calicut University in the year 1997, Master of Computer Application from Bharathidasan University in the year 2000 and Master of Philosophy from Manonmaniam Sundurnar University in the year 2003. Now she is a Research scholar at Bharathiar University, Coimbatore, India. She has authored 3 textbooks and published research papers in International Journals and presented papers at National and International conferences.