DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics - Proposal

Carbon-Aware Scheduling for Serverless Computing

Author: B.Tech. Thandayuthapani Subramanian
Supervisor: Prof. Dr. Michael Gerndt
Advisor: M.Sc. Mohak Chadha

1 Introduction

By 2030, it has been estimated that data centers will consume anywhere between 3% to 13% of global electricity [1]. The recent report [2] from "The Intergovernmental Panel on Climate Change [3]" warns that the world is set to reach the 1.5°C level in next 20 years [4], which would result in 4°C increase by end of century, which is 2°C over the limit agreed on the Paris Agreement on Climate Change [5]. So it is important to come up with new methodologies for increasing the data center energy efficiency to limit it's environmental impact [6], [7]. Serverless Computing is a new cloud computing paradigm which abstracts software completely away from the hardware that it runs on. Many predict that next big evolution in cloud system is serverless [8]. So, it is important to ensure the energy efficiency of serverless cloud offerings.
Major cloud providers like Google Cloud Platform intends to be completely carbon-free by 2030 and since 2017, they have matched its global electricity use with renewable sources like wind and solar purchases [9]. As well as in case of Microsoft, they have been carbon-neutral since 2012 and to become carbon negative by 2030 [10]. Similarly many cloud providers have made vast commitments, like in case of Oracle Cloud their data centers in Europe are 100% carbon-neutral. But only 59% of their electricity use in other parts of the world are renewable [11]. Few of the above mentioned cloud providers' claims come with admonition that their carbon emissions not zero, but are offset by their carbon offset programs, which makes their net operational emissions to zero [12].
Cloud adoption has been increasing over the years [13], it is important for us to turn our focus on sustainable development by establishing energy-aware design strategies. 50% of energy used by data centers are by idle resources [14]. Serverless computing will be one of the main solutions to solve the energy wastage by idle resources. Main aim of this work will be to propose and develop a scheduling policy for intelligent placement of serverless workloads in different regions depending on the availability of renewable resources at given point of time. To achieve this, we will exploit Kubernetes' container orchestration capabilities and Knative's capability of building serverless applications.

2 Background

This section deals with the background information of the concepts that are used. Section 2.1 deals with information about serverless computing and section 2.2 provides information with how kubernetes does scheduling.

2.1 Serverless Computing

Serverless is the fastest-growing and the most preferred cloud service model with annual increase of 75% increase in adoption [15]. Serverless computing offers a platform in cloud where developers simply have to upload their code, and the platform has capabilities to execute on their behalf as needed at any scale. Provisioning resources is responsibility of the platform. Developers only have to pay for the resources only when the code is invoked [16]. Serverless platforms promise new capabilities like event processing, API composition and data flow control that make writing scalable microservices easier and effective. All the major cloud providers have their own solution for serverless computing, like AWS Lambda [17], GCP's Cloud functions [18] and Azure functions [19]. And many open source solutions like OpenWhisk [20] from Apache Software Foundation [21] and Knative [22] from Cloud Native Computing Foundation [23].

2.2 Scheduling in Kubernetes

Kubernetes was developed and open sourced by Google in 2015. It was initially used internally in Google for running and maintaining their containers [24]. Kubernetes was then donated to Cloud Native Computing Foundation, which is part of Linux Foundation. Default scheduling algorithm used in Kubernetes can be understood from kubernetes official documentation [25]. Excerpt from the official documentation as follows: There are two steps before a destination node of a Pod is chosen. The first step is filtering all the nodes and the second is ranking the remaining nodes to find a best fit for the Pod. Scheduler first evaluates all the nodes in the cluster based on number of rules called predicates, which filters out unqualified nodes. Next, all the qualified nodes goes through another scheduling rules called priorities, which ranks the nodes according to preferences. In general, a scheduling policy is combination of predicates and priorities. List of all supported scheduling policies of predicates and priorities are explained in official Kubernetes documentation [25].

3 Related Work

Some research have been done on the area of energy-efficient cloud computing and few more on green data centers and renewable energy. One of the important works is by Patros et al [26], which enumerates potential strategies to reduce the energy overheads of serverless computation. This paper dissects potential strategies into three directions: 1) Changes to design and development of serverless platform level such as power capping, scheduling, etc., 2) By studying workload patterns and try to avoid peak power consumption which forces to use high-emission, fossil-fuel driven generators, 3) Combination of above two strategies. But their analysis is intentionally biased towards machine learning workloads and there is no data regarding more generic serverless workloads in their research.
Data center energy efficiency can be achieved by copious number of ways, which is compiled by Zakarya and Gillam [27]. Some of the methods listed will be further considered in connection with serverless computing in this research, especially ones related to 'renewables' section in their research.
GreenSlot [28] is a scheduler, which predicts the amount of solar energy that will be available in the near future and schedules the workload to maximize the green energy consumption, thus resulting in reduction of energy from fossil fuel, which was proposed by Goiri et al. GreenSlot operates only within a data center rather than between distributed data centers. And it has been implemented only for two specific schedulers - SLURM and MapReduce scheduler in Hadoop. In our research, we will predominantly work with Knative to deploy serverless workloads.
GreenWorks by Li et al [29] is a framework for HPC data centers running on a renewable energy mix. GreenWorks features a cross-layer power management scheme tailored to the timing behaviors and capacity constraints of different energy sources.
Green-Aware Virtual Machine Migration Strategy proposed by Wang et al [30] presents a green-aware virtual machine (VM) migration strategy in such datacenters powered by sustainable energy sources. They define their solution as an optimization problem from an energy-aware point of view and try to solve it using statistical searching approaches. The purpose is to utilize green energy sufficiently while guaranteeing the performance of applications hosted by the datacenter.
GreenSlot [28] and GreenWorks [29] consider only scheduling with regards to renewable energy sources but within a single data center instead of taking global view of distributed data centers. But, our research is distinct in a way, we will consider the global view of distributed data centers and specifically for serverless workloads. Instead of dealing with mix of renewable and non-renewable energy sources, we will consider variable renewable sources exclusively as discussed in [30], but for serverless workload running as containers.

4 Methodology

Since there is not much work done in carbon-aware scheduling for serverless computing, the main focus of this research would be to propose and implement a novel solution for sustainably scheduling serverless workloads across distributed clusters spread out in different regions of the world. In this research, it is planned to utilize capabilities of Knative to develop and deploy serverless applications on top of Kubernetes. In the above mentioned scenario, bulk of work to be done with respect to proposing and implementing scheduling policies will be done on Kubernetes level. In Kubernetes, it is possible to make necessary changes to scheduling policies in three ways [31]. Those are: 1) adding required rules in scheduler code and recompiling, 2) implementing custom scheduler process that runs instead of or along with Kubernetes scheduler, 3) implementing a scheduler extender. Recently Kubernetes community has also proposed scheduling framework for Kubernetes scheduler [32], which defines new extension points and provides APIs in Kubernetes scheduler for use by plugins. Proper investigation should be done for each of the potential way to extend scheduler and decision should be reached on extension of scheduler.
It is important for the scheduler to access data regarding carbon intensity for electrical grids or data centers. There are multiple possibilities to acquire such data in real-time, like WattTime [33]. WattTime provides access to real-time, forecast and historical marginal emission rate for the electric grids around the world, it is possible to access those data using their publicly exposed APIs [34]. Similarly there are other alternatives like - electricityMap [35, 36]. Even cloud providers like GCP also provides cloud service which provides carbon footprint data [37, 38]. It should be possible to make the most of publicly available carbon intensity data to aid scheduler to make scheduling decisions. There are also easily available simulated datasets [39] from the research conducted on temporal workload shifting to reduce carbon emission by Wiesner et al [40].
When working with multiple clusters of Kubernetes distributed geographically, it is important for the research to have capability of scheduling the workload across different clusters. It is feasible with the help of Admiralty [41] and Virtual Kubelet [42]. Admiralty is a set of Kubernetes controllers which can prudently schedule workloads across different cluster. Virtual Kubelet is an masqueraded version of Kubernetes Kubelet [43], which enables connecting Kubernetes to other APIs like Admiralty, AWS Fargate, Tensile Kube, etc. In short, Virtual Kubelet can be described as "Kubernetes API on top, programmable back".
Multi-cluster scheduling is done by Admiralty with the help of extending kubernetes scheduler using extensible scheduler framework and virtual kubelet. Admiralty supports multiple cluster topologies like [44] - Central control plane, Cloud bursting and Decentralized federation. Admiralty provides lot of flexibility in terms of desired federated cluster topology. Admiralty works as following [41] - It mutates the pods into proxy pods scheduled on

virtual-kubelet nodes representing target clusters and creates delegate pods in the remote clusters. Then the pod dependencies are copied, if any, like config maps and secrets. Then an feedback loop updates the statuses and annotations of the proxy pod to reflect the statuses and annotations.
So it is established how it is planned to extend the kubernetes scheduler to make scheduling decisions which is carbon-aware. Multiple sources of the carbon intensity data is listed above and it is pertinent for this research to exploit the data available in open source and put together a scheduling scheme which gels together with Admiralty, enabling multi-cluster scheduling and managing the life cycle of the workloads across cluster a possibility. In conclusion, focus of this research is to extend Kubernetes scheduler to consider publicly available carbon intensity data to make scheduling calls, which can deploy the workloads across geographically distributed Kubernetes cluster, to make the most use of sustainable, intermittently-available renewable energy for the running serverless workloads.

Based on the above mentioned goals, this study aims to answer the following three research questions (RQs) by evaluating and implementing scheduling policies for serverless workloads:

- **Research Question 1:** *When and what kind of serverless workloads should be migrated?*
- **Research Question 2:** *How can Knative with Kubernetes be extended to support carbon-aware scheduling of serverless functions?*
- **Research Question 3:** *What kind of scheduling policy should be adopted to achieve the goal?*

4.1 RQ 1: Migrating Serverless Workloads

In general, we can safely segregate types of workloads which are commonly deployed as serverless workloads into four [45]: 1) Micro Jobs, 2) Application, 3) ML Model - Training, 4) ML Model - Serving. Micro Jobs can be relatively short running workloads in comparison with other workloads. Typically, this kind of workload involve having arithmetic operations, matrix operations and solving equations. It can be safely assumed that micro jobs take either some values or a JSON as input and either JSON or file as an output. In case of application, the workload can vary from processing a transaction and storing the state in some database to image and video processing application. So the execution time of such workloads will vary depending on the use case. Though the names look similar for next two workloads, they both have different characteristics. In ML model - Training workload, we train actual ML model, which is very much a data intensive and long running workload. In ML model - Serving workload, we deploy already trained model and use it for inference. This type of workload can also be long running one, but not as data intensive as training a ML model.
Our main goal in this research is to make carbon-aware scheduling decision. Transferring huge amount of data through network as input for serverless workload or output from serverless workload will consume energy and directly contradicts our main goal, which

is to reduce the carbon foot print for our workload execution. So it is sensible to migrate serverless workloads which are not data intensive and transferring input and output through network will not cost much in terms of energy. In conclusion, it is logical to migrate following serverless workload types: Micro jobs, Application which are not data intensive and ML model - Serving.

4.2 RQ 2: Extending Knative

In this research, Knative will be used to create serverless workloads. Knative is an open source solution to build and manage serverless workloads in kubernetes clusters. In short, the functionality involved in Knative can be explained as follows: It is required as an user to create Knative service [46] to deploy their application. Under the hood, Knative create a ReplicaSet [47] in Kubernetes. ReplicaSet is a Kubernetes object which ensures there is stable set of running pods for a specific workload. Number of pods should be defined in ReplicaSet configuration. Those pods are scheduled using scheduler process in Kubernetes. As mentioned above, it is possible to extend kubernetes schedulers [31, 32]. Real-time data carbon intensity data is required for making scheduling decisions. It is possible to access those data from services like WattTime [33, 34] and electricityMap [35, 36]. Other alternatives like simulated dataset [39] is also available. Theoretically, it is possible to extend Knative with Kubernetes with the above mentioned resources to make carbon-aware scheduling decisions.

4.3 RQ 3: Scheduling Policy

Scheduling policy is set of directives and purpose which directs the scheduler process to make optimal decision. In Kubernetes scheduler, selection of node is done in a 2-step operation [48]: 1) Filtering using Predicate scheduling policies, 2) Scoring using Priorities scheduling policies. Filtering step finds suitable set of nodes which is feasible to schedule the workload. In filtering stage, scheduler runs workload's requirement and list of nodes in cluster through predefined set of predicate policies like CheckNodeConditionPredicate, CheckNodeUnschedulablePredicate, etc., and obtains list of nodes which satisfies workload's requirement. After filtering step, in scoring stage filtered set of nodes and workload's requirement is given as input for definite set of priority scheduling policies like ImageLocality, PodTopologySpread, etc. After scoring stage, we get score assigned to each node and scheduler chooses the node which has the highest score. In this research, a new scheduling policy in scoring stage would help accomplishing the expected behaviour. Aim of this research is to implement a carbon-aware scheduling policy, which prioritizes to place the workload in a node in a region which uses energy with less carbon footprint.

Bibliography

[1] A. S. G. Andrae and T. Edler. "On Global Electricity Usage of Communication Technology: Trends to 2030". In: *Challenges* 6.1 (2015), pp. 117–157. ISSN: 2078-1547. DOI: 10.3390/challe6010117. URL: https://www.mdpi.com/2078-1547/6/1/117.

[2] e. P.R. Shukla J. Skea. "Climate Change 2022: Mitigation of Climate Change". In: *Cambridge University Press, Cambridge, UK and New York, NY, USA.* (2022). DOI: 10.1017/9781009157926. URL: https://report.ipcc.ch/ar6wg3/pdf/IPCC_AR6_WGIII_FinalDraft_FullReport.pdf.

[3] *The Intergovernmental Panel on Climate Change*. https://www.ipcc.ch/. Accessed: 2022-06-01.

[4] *IPCC climate report 2022 summary*. https://climate.selectra.com/en/news/ipcc-report-2022. Accessed: 2022-06-01.

[5] *The Paris Agreement*. https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement. Accessed: 2022-06-01.

[6] M. Dayarathna, Y. Wen, and R. Fan. "Data Center Energy Consumption Modeling: A Survey". In: *IEEE Communications Surveys and Tutorials* 18.1 (2016), pp. 732–794. DOI: 10.1109/COMST.2015.2481183.

[7] *Data centers are more energy efficient than ever*. https://blog.google/outreach-initiatives/sustainability/data-centers-energy-efficient. Accessed: 2022-06-01.

[8] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson. "What Serverless Computing is and Should Become: The next Phase of Cloud Computing". In: *Commun. ACM* 64.5 (Apr. 2021), pp. 76–84. ISSN: 0001-0782. DOI: 10.1145/3406011. URL: https://doi.org/10.1145/3406011.

[9] *Tracking Google's carbon-free energy progress*. https://sustainability.google/progress/energy/. Accessed: 2022-06-01.

[10] *Environmental sustainability - Microsoft*. https://www.microsoft.com/en-us/corporate-responsibility/sustainability. Accessed: 2022-06-01.

[11] *Sustainability - Oracle Cloud*. https://www.oracle.com/corporate/citizenship/sustainability/clean-cloud.html. Accessed: 2022-06-01.

[12] *Google environmental report*. https://www.gstatic.com/gumdrop/sustainability/google-2017-environmental-report.pdf. Accessed: 2022-06-01.

[13] *Cloud adoption statistics*. https://findstack.com/cloud-adoption-statistics. Accessed: 2022-06-01.

[14] *How much energy do data centers use?* https://davidmytton.blog/how-much-energy-do-data-centers-use/. Accessed: 2022-06-01.

[15] *Serverless Trends*. https://techbeacon.com/enterprise-it/state-serverless-6-trends-watch. Accessed: 2022-06-01.

[16] J. M. Hellerstein, J. Faleiro, J. E. Gonzalez, J. Schleier-Smith, V. Sreekanti, A. Tumanov, and C. Wu. "Serverless computing: One step forward, two steps back". In: *arXiv preprint arXiv:1812.03651* (2018).

[17] *AWS Functions*. https://aws.amazon.com/lambda/. Accessed: 2022-06-01.

[18] *GCP - Cloud Functions*. https://cloud.google.com/functions/. Accessed: 2022-06-01.

[19] *Azure Functions*. https://azure.microsoft.com/en-us/services/functions/. Accessed: 2022-06-01.

[20] *Apache OpenWhisk: An open source serverless cloud platform*. https://openwhisk.apache.org/. Accessed: 2022-06-01.

[21] *Apache Software Foundation*. https://www.apache.org/. Accessed: 2022-06-01.

[22] *Knative Has Applied to Become a CNCF Incubating Project*. https://knative.dev/blog/steering/knative-cncf-donation/. Accessed: 2022-06-01.

[23] *Cloud Native Computing Foundation*. https://www.cncf.io/. Accessed: 2022-06-01.

[24] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. "Large-scale cluster management at Google with Borg". In: *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France, 2015.

[25] *Scheduling algorithm in Kubernetes*. https://github.com/kubernetes/community/blob/master/contributors/devel/sig-scheduling/scheduler_algorithm.md. Accessed: 2022-06-01.

[26] P. Patros, J. Spillner, A. V. Papadopoulos, B. Varghese, O. Rana, and S. Dustdar. "Toward Sustainable Serverless Computing". In: *IEEE Internet Computing* 25.6 (2021), pp. 42–50. DOI: 10.1109/MIC.2021.3093105.

[27] M. Zakarya and L. Gillam. "Energy efficient computing, clusters, grids and clouds: A taxonomy and survey". In: *Sustainable Computing: Informatics and Systems* 14 (2017), pp. 13–33. ISSN: 2210-5379. DOI: https://doi.org/10.1016/j.suscom.2017.03.002. URL: https://www.sciencedirect.com/science/article/pii/S2210537917300707.

[28] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. "GreenSlot: Scheduling Energy Consumption in Green Datacenters". In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '11. Seattle, Washington: Association for Computing Machinery, 2011. ISBN: 9781450307710. DOI: 10.1145/2063384.2063411. URL: https://doi.org/10.1145/2063384.2063411.

[29] C. Li, R. Wang, D. Qian, and T. Li. "Managing Server Clusters on Renewable Energy Mix". In: *ACM Trans. Auton. Adapt. Syst.* 11.1 (Feb. 2016). ISSN: 1556-4665. DOI: 10.1145/2845085. URL: https://doi.org/10.1145/2845085.

[30] X. Wang, Z. Du, Y. Chen, and M. Yang. "A green-aware virtual machine migration strategy for sustainable datacenter powered by renewable energy". In: *Simulation Modelling Practice and Theory* 58 (2015). Special Issue on TECHNIQUES AND APPLICATIONS FOR SUSTAINABLE ULTRASCALE COMPUTING SYSTEMS, pp. 3–14. ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2015.01.005. URL: https://www.sciencedirect.com/science/article/pii/S1569190X15000155.

[31] *Scheduler extender*. https://github.com/kubernetes/design-proposals-archive/blob/main/scheduling/scheduler_extender.md. Accessed: 2022-06-01.

[32] *Scheduler extender*. https://github.com/kubernetes/enhancements/blob/master/keps/sig-scheduling/624-scheduling-framework/README.md. Accessed: 2022-06-01.

[33] *WattTime - The Power to Choose Clean Energy*. https://www.watttime.org/. Accessed: 2022-06-01.

[34] *WattTime - API reference*. https://www.watttime.org/api-documentation. Accessed: 2022-06-01.

[35] *electricityMap - The leading resource for 24/7 electricity CO2 data*. https://electricitymap.org/. Accessed: 2022-06-01.

[36] *electricityMap - API reference*. https://static.electricitymap.org/api/docs/index.html. Accessed: 2022-06-01.

[37] *Carbon Footprint | Google Cloud*. https://cloud.google.com/carbon-footprint. Accessed: 2022-06-01.

[38] *Carbon free energy for Google Cloud regions*. https://cloud.google.com/sustainability/region-carbon. Accessed: 2022-06-01.

[39] *Let's Wait Awhile - Datasets, Simulator, Analysis*. https://github.com/dos-group/lets-wait-awhile. Accessed: 2022-06-01.

[40] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen. "Let's wait awhile". In: *Proceedings of the 22nd International Middleware Conference*. ACM, Dec. 2021. DOI: 10.1145/3464298.3493399. URL: https://doi.org/10.1145%2F3464298.3493399.

[41] *Admiralty - The simplest way to deploy applications to multiple Kubernetes clusters*. https://admiralty.io/docs/. Accessed: 2022-06-01.

[42] *Virtual Kubelet*. https://virtual-kubelet.io/. Accessed: 2022-06-01.

[43] *Kubelet | Kubernetes docs*. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet. Accessed: 2022-06-01.

[44] *Kubernetes Scheduler*. https://admiralty.io/docs/concepts/topologies. Accessed: 2022-06-12.

[45] J. Kim and K. Lee. "FunctionBench: A Suite of Workloads for Serverless Cloud Function Service". In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. 2019, pp. 502–504. DOI: 10.1109/CLOUD.2019.00091.

[46] *About Knative Services*. https://knative.dev/docs/serving/services/. Accessed: 2022-06-12.

[47] *ReplicaSet | Kubernetes*. https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/. Accessed: 2022-06-12.

[48] *Kubernetes Scheduler*. https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/. Accessed: 2022-06-12.