Evaluation of a deep learning model on Kubeless.

Dhaval Rajeshbhai Bhanderi
Dept. of Computer Science
University of Georgia
Athens, Georgia
drb34163@uga.edu

Vishakha Vijay Atole
Dept. of Computer Science
University of Georgia
Athens, Georgia
va45686@uga.edu

Anuj Mehul Panchmia
Dept. of Computer Science
University of Georgia
Athens, Georgia
ap83035@uga.edu

***Abstract*—Recent advancements in virtualization and cloud technology have led to a compelling paradigm of serverless computing, which allows development and deployment of a wide range of event -based cloud applications as stateless functions. At the same time, Machine learning and Artificial intelligence are also heavily being adopted by many enterprises and organizations. With this heavy increase in the adoption of the Function-as-a-Service (FaaS) platform and machine learning applications, the suitability of machine learning models needs to be evaluated on a serverless platform. However, ML models have not been evaluated on any open source serverless platform. In this work, we will analyze the suitability of Kubeless—Kubernetes -native serverless platform, for large neural network models. In our work, we evaluate the performance of an open source serverless platform Kubeless and containerized application serving large deep learning models. Additionally, our work aims at rendering a performance comparison of response time for serverless platform and that for containerized application. Our experimental results show that there is no major change and overhead in the response time for Kubeless in comparison with containerized application which is deployed on Kubernetes cluster.**

I. Introduction

Over the past few years, cloud businesses have become very popular and have given rise to new approaches and technologies. Now-a-days, "Serverless" is a new buzzword in the cloud industry and has created a hype. Serverless computing is an emerging cloud service wherein the applications are decomposed into independent stateless functions. These functions are executed in response to triggers (For ex, HTTP triggers, Database triggers or any messaging events). They can be scaled independently as they are completely stateless. This is considered as a very promising approach for several applications like data analytics, scientific computing and mobile computing. Serverless platforms provide the capabilities that make writing and deploying scalable applications in an easier, simpler and a cost-effective way.

In serverless computing, the underlying infrastructure and configurations are managed by a third-party service provider or an operations team while using private cloud services. The first serverless platform was presented by Amazon. At this time, a platform for serverless computing was offered by all the major cloud service providers. The major cloud service providers are AWS Lambda, Azure Functions, Google Cloud Functions and IBM Cloud Functions. There are several open source FaaS platforms available like Kubeless, OpenFaaS, Apache OpenWhisk and Fission.

Also, there has been a heavy rise in the use of machine learning algorithms by enterprises and organizations. Our work primarily focuses on the applications of Deep learning - a part of broader field of machine learning that utilizes neural network to provide services like image classification and recognition, text analysis, natural language processing and speech recognition. Here, our key idea is to amalgamate both the evolving fields of serverless computing and deep learning. Some of the evaluations are measured on third-party service providers like AWS Lambda and Azure Functions [3].But there is no such work that has been done for an open source

serverless platform providers. In this work, we aim at evaluating the suitability of serving large neural networks on Kubeless - an open source serverless computing platform[1].

A. An Open source serverless platform: Kubeless

Kubeless —Kubernetes –native serverless framework, allows to deploy functions (small bit of code) without having to worry about the infrastructure. The Kubeless framework is designed to be deployed on top of the Kubernetes cluster and takes advantage of all the Kubernetes primitives [2] [23]. Kubeless clones with the major serverless platform provider AWS Lambda, Azure Functions and Google Cloud Functions. Kubeless provides runtimes in Python, Node.js, Ruby, Go and Java. It uses Custom Resource Definitions (CRDs) to extend the Kubernetes API and to create functions as custom Kubernetes resources. The Kubeless controller continuously monitors for any changes to function objects and takes necessary action. For example, if a function object is created, the controller creates a pod for the function. A function's runtime is encapsulated as a container image and Kubernetes config maps are used to deploy a function's code in the runtime.

B. Deep Learning

Deep learning, a part of broader field machine learning, is driven by neural networks. It is overtaking the traditional machine learning methods for understanding perceptual and unstructured data. Deep learning models are used for applications of natural language processing, image classification, speech and image recognition. Users can develop and train the models with the help of several deep learning frameworks like TensorFlow, Theano, Caffe etc. Training of the neural networks can consume a large amount of time. Once the training is complete, it can be used for inferencing on new data.

In this work, experiments were conducted with the help of Yelp's dataset in order to build a deep learning model which was capable of classifying an image uploaded by an user [5] [6]. After which, the model was trained in such a way that it was capable of classifying the image into five categories namely food, drinks, menu, inside view and outside view. We have built and trained a convolutional neural network (CNN) for our image classification model[7]. We have also used Keras -an open source neural network library running on TensorFlow[4] backend to initialize the neural network [8] [9].

II. Related Work

The first serverless platform was AWS Lambda [10]. AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the Amazon Web Services. Other related works include OpenWhisk and OpenLambda which are open source serverless platforms [11]. Static websites can be built on AWS S3 using serverless framework [12]. Previous work has been conducted on how to leverage AWS Lambda and MXNet to build a scalable prediction pipeline [13]. This experiment however, failed in drawing specific execution times and eventually included a delay while downloading the model from S3. This experiment was incomplete because it did not consider most of the dimensions like cold and warm starts, rate of requests and memory.

Bila et al. talk about leveraging the serverless architecture for securing Linux containers [14]. This work facilitated continuous scanning for containers which upon detection of vulnerabilities takes the required actions using an automated threat mitigation framework. This analysis of threat and mitigating it was entirely based upon user -defined policies for the compromised containers. A related research mentions about presenting a generic architecture for a chatbot framework on top of a serverless computing framework [15]. Garrett McGrath and Paul R. Brenner have presented the design of performance-oriented serverless computing platform implemented in .Net, deployed in Microsoft Azure and which utilized containers as function execution environments [16]. Their work also proposed the metrics in order to evaluate the execution performance of serverless platforms and also conducts various experiments on various service providers. This included tests that were designed to measure latency of serverless platforms thereby, showing container expiration thresholds.

Geoffrey C. Fox et al present their work on the status of serverless computing and function-as-a-service in industry and research while, they had also conducted a workshop on the same topic [17]. Serverless computing also provides an on-demand high performance computing for biomedical research [18]. This case study demonstrates high chances for leveraging computing computational resources in bio-medical research thereby making it easy to use, scalable and cost effective. Serverless frameworks can be used with TensorFlow for deep learning and image recognition [19]. This work is highly optimized which focuses on maximizing throughput and minimizing the latency with the help of proper adaptive model selection techniques. An important related work is that of Vatche Ishakian et al. who present their work

on serving deep learning models on a serverless platform [20].

III. Approach

For this project work, our aim is to check the feasibility of serving deep learning models on a serverless platform -Kubeless. Our approach includes building a deep learning model using Convolutional Neural Network (CNN) running on the TensorFlow backend and trained on the GPU separately. After which, we deploy this trained network on Kubeless as a function and on Kubernetes as containerized application.

Our work focuses on rendering results for the following metrics:

- Response Time: The latency as observed by the end users.
- Prediction time: The total time taken by the model for prediction.

A. System Design:

To measure the performance metrics, we have deployed our trained network on both Kubeless and Kubernetes cluster on separate virtual machines. We have used Minikube which provides a single node Kubernetes cluster. For our experiment, we have used the latest Minikube version v0.29.0. After setting up a working cluster on both the virtual machines, we installed Kubeless on one cluster and docker container on the other virtual machine. We have used the latest the version v1.0.0-alpha of Kubeless for our system.

We kept all the system configuration same for both the clusters as we wanted to compare performance of both the systems. We assigned two CPUs and 2 GB RAM for both the clusters. For Kubeless, we have used a serverless framework to write and deploy our function on Kubeless. Serverless framework is an open-source web framework which provides a simple CLI that makes it easy to write and deploy function on various FaaS platforms. We can invoke the functions using the HTTP triggers on Kubeless. We have used Kafka for internal message service in a Kubeless environment.

B. Implementation:

Our project work implementation is categorized into two major parts:

1. Model Training:

As we are evaluating a deep learning model on Kubeless, we needed a large deep learning model to deploy on Kubeless. For that, we have used Yelp's image dataset to train our model. We have built a convolutional neural network (CNN) and have trained this model on the Yelp's dataset. We have trained the model on GPU separately and saved the model with the highest weights. The training dataset has 19309 images classified in five different classes. We then, have built and trained the model using Keras neural network libraries and TensorFlow. After the training, we obtained a model with size of around 11 MB. The accuracy of the model is 0.79 with the loss value of 0.72.

1. Deployment:

After the model is trained, we deployed this model for inferencing on Kubeless and Kubernetes cluster. We used Kubectl CLI to call the Kubernetes APIs and Kafka for internal messaging service. For the Kubeless, we deployed the model as an event based function in Python runtime. Here, one of the major challenge that we faced was handling the dependencies for our models at runtime. As we are using python 3 as a runtime environment which does not support the PIL library (which is a requirement of image.keras), we had to install Pillow-a PIL fork. We have written function using http-trigger. For Kubeless, we do not have to configure about any system resource.

We also deploy this model as a docker image on Kubernetes cluster. Since the Kubernetes cluster has inbuilt docker runtime it becomes an easy task to pull the image and run it as a container. Creating a docker image makes the program OS independent as the dependencies are taken care by the docker image. It makes the necessary installations like TensorFlow, Keras and Pillow and set up the python 3.6 runtime environment. The image is deployed as a container residing in a Kubernetes pod. The deployed container is exposed to a http port for taking the handling subsequent requests.

IV. Evaluation Plan

The primary goal of our project is to check the suitability of a serverless computing platform for large neural networks. We have used Kubeless deployed on a Kubernetes cluster and a deep learning model using TensorFlow as back-end to classify the images. We have evaluated Kubeless on the basis

of response time, I/O performance and concurrent function invocation.

The model size is an important factor that affects the performance of inferencing. We have evaluated an Image classification model using open source neural network libraries namely Keras and TensorFlow. The size of our trained neural network model is 11 MB and comprises of 5 layers.

A. Evaluation setup:

All the experiments involved in this work have been performed on a custom Kubernetes cluster which is deployed on a virtualized instance. Kubeless interacts directly with the Kubernetes cluster manager. We have used Minikube, which provides a single-node Kubernetes cluster. Kubernetes version 1.10.4 has been used in our work. The Kubeless version we have used is 1.0.0-alpha. We have also used the Nginx Ingress controller to provide routing for functions.

The virtual machine that we have used has two CPUs, 2 GB RAM and runs on Linux 2.6/3.x OS. We have also used LoadRunner and ApacheBench to generate HTTP requests that invoke functions.

B. Impact on Concurrency:

Firstly, we measure the mean response time of different number of concurrent requests (that is 1, 5, 10, 20 concurrent requests). The purpose of this experiment is to identify performance issues which may arise while serving at the production level. For this, we have written a simple function in python runtime that takes an image file as input and returns the category of image as string. We deployed this function on Kubeless and invoked it through http-trigger.

Figure (1) depicts the mean response time for different number of groups of concurrent requests on Kubeless. For the one request only, kubeless takes 0.2 s to respond.

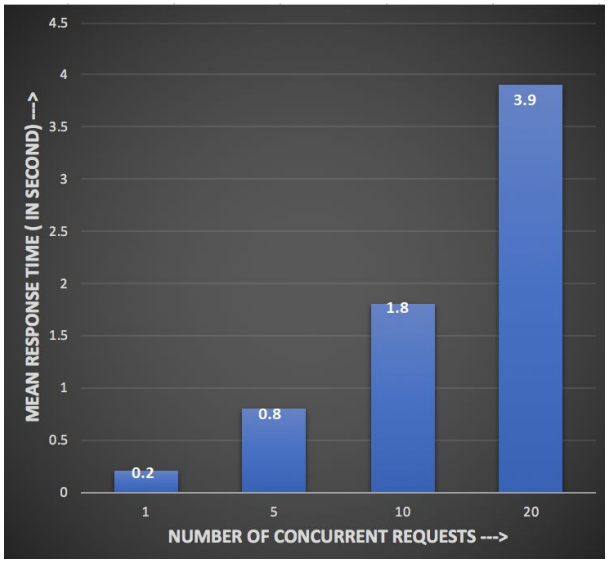

Fig. (1): Mean response time for different number of concurrent requests.

We repeated each experiment up to five times and then, calculated their mean. Up to 20 concurrent requests are considered for evaluation. Here, for the 20 requests that were executed concurrently, the average response time is 965s, the quickest response is 472 ms and the longest response time takes up to 1s.

During this experiment, we did not configure the horizontal pod auto scalar technique provided by Kubeless. Despite of this, we received all the responses successfully without getting an error for any requests.

C. Comparison with containerized application:

This work focuses on the comparison of measurements for serverless platform and containerized application. We have deployed our trained neural network on a single-node Kubernetes cluster. We also deployed this neural network on Kubeless as a function. We used the virtual machines with same specification of two CPUs, 2 GB RAM and Linux OS. We then, invoked the function by HTTP trigger event with an image file as an input. We noted the response time and prediction time for both the platform.

	Prediction Time	Response Time
Kubeless	0.12s	0.2s
Kubernetes	0.10s	0.13s

Table (1): Performance metrics for Kubelesss and containerized application

We have calculated an average of 10 requests for both the platforms. A closure examination as shown in the above table (1) depicts that response time and prediction time are slightly higher for Kubeless than that for Kubernetes. However, the prediction time for Kubeless is 0.02s more than that of containerized application. Kubeless takes around 35% more response time than that of the containerized application.

Here for this experiment, we do not have consider the latency because of cold start while invoking an idle function.

D. CPU utilization:

We do not find any meaningful results of the CPU utilization while triggering concurrent requests while performing same experiment given in Impact on Concurrency with enabling Horizontal Pod Auto-scaling.

V. Discussions and Learning

Our experimental results show that Kubeless has a higher response time and prediction time for our deep learning model. However, these results are still acceptable as they provide an easy way to deploy and maintain applications working on large neural networks. Kubeless also gives satisfactory results for rapidly scaling applications. This is because of its simple architecture and the use of native Kubernetes APIs. On this basis, we considered Kubeless as a feasible serverless platform for inferencing and serving trained neural networks. In addition, Kubeless has recently released version v1.0.0-alpha. We feel that it is still not fully stable and with upcoming releases, we can expect it will achieve adequate performance and features.

One of the major challenge for serverless computing platform is the latency due to cold starting. The latency of warm requests is within a reasonable range. However, this is true when the allocated memory sizes are greater than 1024 MB. The latency of cold requests can be significantly higher. One way to offset the delay is to provide access of resources like GPUs for inferencing because they have shown to better performance ratios than CPUs. There is also need for tools that analyze previous function executions and suggest changes in declared resources.

Apart from that, serverless computing cannot be considered for applications that require high-end computing power with intensive I/O operations. The reason behind this is that serverless computing imposes some restrictions on resources. Few of the restrictions include 5 minutes execution timeout and no GPU support.

We consider this project as full of learnings. This work helped us take our first step towards the world of Serverless computing. This work helped us explore an open source serverless platform Kubeless and a container orchestration system Kubernetes. This project work helped us understand where serverless computing would be a good option and where it would not. We also became familiar with some concepts of deep learning. In particular, we learnt how to develop and train a neural network using Keras and TensorFlow. We deployed a large neural network for serving as functions on Kubeless and as containerized application on Kubernetes.

VI. CONCLUSION

In this work, we have evaluated the suitability of deep learning models on a serverless platform. In particular we have analyzed the suitability of Kubeless - Kubernetes native serverless platform for large neural network models. For that, we evaluated the performance comparison between an open source platform- Kubeless and containerized application serving deep learning models. We have measured the response time and prediction time for both Kubeless and containerized applications. We also measure the performance degradation while triggering concurrent requests for Kubeless function. Our results indicate that there is no significant change or overhead in the prediction time for serverless platform and that for containerized applications. However, we noted around 35% increase in the response time for Kubeless.

In future, this work can be extended by evaluating different deep learning models on several open source serverless computing platforms like Fission, OpenFaaS and OpenWhisk.

VII. References

[1] https://github.com/kubeless/kubeless
[2] https://en.wikipedia.org/wiki/Kubernetes
[3] https://arxiv.org/pdf/1710.08460.pdf

Vatche Ishakian, Vinod Muthusamy, Aleksander Slominski - Serving deep learning models in a serverless platform

[4] https://www.tensorflow.org/

[5] https://engineeringblog.yelp.com/2015/10/how-we-use-deep-learning-to-classify-business-photos-at-yelp.html

[6] http://blog.kaggle.com/2016/04/28/yelprestaurant-photo-classification-winners-interview-1st-place-dmitrii-tsybulevskii/

[7] https://pdfs.semanticscholar.org/2067/09bf8177e9d76f56189c29959d4e37833a9e.pdf

Yixin Cai - Yelp Restaurant Photo Classification

[8] https://en.wikipedia.org/wiki/Keras

[9] https://www.tensorflow.org/tfx/

[10] AWS Lambda https://aws.amazon.com/lambda/

[11] Hendrickson, Scott, et al. "Serverless Computation with OpenLambda." 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16). 2016

[12] https://www.serverlessops.io/blog/static-websites-on-aws-s3-with-serverless-framework

[13] https://aws.amazon.com/blogs/compute/seamlessly-scale-predictions-with-aws-lambda-and-mxnet/

[14] http://www.cs.umd.edu/class/fall2017/cmsc414/readings/serverless.pdf

[15] https://www.researchgate.net/publication/311755516_Building_a_Chatbot_with_Serverless_Computing

[16] https://www.serverlesscomputing.org/wosc17/presentations/garrett.pdf

[17] https://arxiv.org/pdf/1708.08028.pdf

[18] https://arxiv.org/pdf/1807.11659.pdf

[19] https://serverless.com/blog/using-tensorflow-serverless-framework-deep-learning-image-recognition/

[20] https://arxiv.org/pdf/1710.08460.pdf

[21] https://core.ac.uk/download/pdf/159414228.pdf

Josef Spillner, Cristian Mateos, and David A. Monge - FaaSter, Better, Cheaper: The Prospect of Serverless Scientific Computing and

[22] https://kubernetes.io/

[23] https://www.virtualbox.org/

[24] https://cloud.google.com/sdk/

[25] https://engineeringblog.yelp.com/2015/10/how-we-use-deep-learning-to-classify-business-photos-at-yelp.html

[26] http://blog.kaggle.com/2016/04/28/yelprestaurant-photo-classification-winners-interview-1st-place-dmitrii-tsybulevskii/

[27] [23] S. T. predictions with Python and A. Lambda, "https://medium.com/tooso/serving-tensorflow-predictions-with-pythonand-aws-lambda-facb4ab87ddd," 2017.