2019 IEEE 12th International Conference on Cloud Computing (CLOUD)

FunctionBench : A Suite of Workloads for Serverless Cloud Function Service

Jeongchul Kim
College of Computer Science
Kookmin University, South Korea
kjc5443@kookmin.ac.kr

Kyungyong Lee
College of Computer Science
Kookmin University, South Korea
leeky@kookmin.ac.kr

***Abstract*—Serverless computing is attracting considerable attention recently, but many published papers use micro-benchmarks for evaluation that might result in impracticality. To address this, we present FunctionBench, a suite of practical function workloads for public services. It contains realistic data-oriented applications that utilize various resources during execution. The source codes customized for various cloud service providers are publicly available. We are positive that it suggests opportunities for new function applications with lessen experiment setup overheads.**

***Index Terms*—workload; serverless; benchmark; FaaS; cloud**

I. Introduction

Serverless computing with a Function-as-a-Service (FaaS) execution model is rapidly gaining popularity. The FaaS model allows programmers to focus on the core application development without overhead from server provisioning and run-time management. In the FaaS execution model, containers launched from virtual machines are utilized to run user-defined functions. It is well-known that many cloud service vendors provide serverless computing services with *proprietary-library* attached to a FaaS model [2]. For example, the Lambda service by AWS, which is the first public FaaS provider, provides a well-integrated service with AWS S3 (object storage), DynamoDB (key-value storage), SNS (notification), and SQS (message queueing).

Due to its popularity, the FaaS model has been employed in the industry and academia to achieve several applications and research breakthroughs, respectively, resulting in the coverage of a wide range of topics such as opportunities and limitation of serverless computing [2], new applications [4, 1], function run-time environment optimization [6], and public service comparison [8, 5].

Though many research results based on the FaaS execution model have been published in the literature, they lack common workloads that are vital in the accurate comparison of many systems and algorithms. Consequently, most published results are based on micro-benchmarks that emphasize specific computer resources such as CPU, memory, disk, or network. Though such micro-benchmarks provide a way to evaluate each resource exclusively, they are different from the requirements of real FaaS applications that are widely used recently. To overcome the limitation, we propose FunctionBench, which provides a suite of workloads to evaluate various aspects of a FaaS execution model in realistic application scenarios[1].

FunctionBench contains a micro-benchmark and an application workload; the micro-benchmark uses simple *system calls* to exclusively measure the performance of the target resource, while the application-benchmark represents realistic data-oriented applications that generally utilize various resources together. We are positive that the introduction of FunctionBench will enable researchers to easily deploy function applications in the accurate evaluation of various systems and algorithms. As far as we know, the proposed FunctionBench is the publicly available FaaS workload suites that can be deployed on public cloud services.

II. Proposed Function Workloads

Table I shows FuntionBench workloads. The column labeled *amount of loads* represents the relative overhead intensity of each workload, and the input/output columns define how they should be prepared to run the workloads. The current version of FunctionBench supports Python run-time, and we plan to support other programming languages. To make the workloads widely applicable, we prepare each function to be deployable in AWS, Microsoft, and Google's cloud function service. In the AWS and Google cloud, users can configure the allocated RAM of the function run-time. Moreover, based on the configured memory size, we prepared the workload input size to be adjustable.

Micro-benchmark The FunctionBench contains the *float* workload (floating point arithmetic operations - squareroot, sin, and cos); *matrix multiplication* (two N-dimensional square matrix); and *Linpack* (solving linear equations), which is used to mainly measure the CPU and memory bound performance. To measure the performance of the disk IO, we add a micro-benchmark that performs the *dd* system command, which creates a file in the */tmp/* directory of the function run-time. Furthermore, to measure the network performance, we added the *cloud storage* (download the object from the input bucket and upload the object to the output bucket); and an *iperf3* workload that initiates a direct connection between sender and receiver for a test. Since most of current function execution environments do not provide direct connection between function run-times [2], we used a dedicated cloud instance to serve

[1]https://github.com/kmu-bigdata/serverless-faas-workbench

2159-6190/19/$31.00 ©2019 IEEE
DOI 10.1109/CLOUD.2019.00091

TABLE I: Workloads in FunctionBench

category	name	amount of loads				input	output
		CPU	Memory	Disk I/O	Network		
Micro benchmark	float	high	high	-	-	JSON (argument)	JSON (argument)
	matrix multiplication	high	high	-	-	JSON (argument)	JSON (argument)
	linpack	high	high	-	-	JSON (argument)	JSON (argument)
	dd	medium	medium	high	-	value (argument)	file (local block storage)
	iperf3	low	low	-	high	-	-
	cloud storage	low	low	medium	high	file(shared block storage)	file(shared block storage)
Application	image processing	medium	medium	low	low	image (shared block storage)	image (shared block storage)
	video processing	high	high	medium	medium	video (shared block storage)	video (shared block storage)
ML Model Training	featurization	high	high	medium	medium	text data (shared block storage)	text data (shared block storage)
	logistic regression	high	high	medium	medium	text data (shared block storage)	model (shared block storage)
ML Model Serving	face detection	medium	medium	medium	medium	video (shared block storage)	video (shared block storage)
	logistic regression	medium	medium	low	low	text data (argument)	JSON (argument)
	CNN (image classification)	medium	medium	low	low	image (shared block storage)	JSON (argument)
	RNN (words generation)	low	low	low	low	JSON (argument)	JSON (argument)

as a server where a function run-time can initiate a direct connection.

Application To represent real-world applications, we added an *Image Processing* workload, which performs image transformation tasks using Python *Pillow* library. In the workload, it fetches an input image from a shared block storage and applies ten different effects to it. The outputs are uploaded to a shared storage. The workload imposes a medium degree of CPU/memory for the image transformation and low degree of network and disk IO overhead to download an input image, upload output images, and store temporary files during computation. The *Video Processing* workload applies gray-scale effect from the OpenCV library to a video input and uploads the transformed video to a shared storage.

ML Model Training Despite the growing popularity of the FaaS execution model, most application scenarios are constrained by the orchestration of many cloud service components and some embarrassingly parallel processing jobs [2]. To extend the application scenarios, the FaaS execution run-time needs to be machine-learning jobs friendly. Furthermore, to create new possible application of the service, we introduce a few data mining tasks in FunctionBench. In a machine learning job, raw input data generally needs pre-processing to prepare the input as features for training. In the *featurization* workload, we use Amazon Fine Food Review[2] text dataset assuming that each review is transformed into a TF-IDF vector, which becomes an input to a regression model. Though the maximum input dataset size that can be executed varies according to the configured memory size, this workload has the largest input size for processing and incurs high network overhead. To run the workload on a FaaS environment with different RAM configuration in parallel, we partition the input dataset into various sizes. Also, to calculate a global TF-IDF vector from partitioned input datasets, multiple invocations of the function are necessary for parallel processing and aggregation. Public cloud service vendors provide a function orchestration feature (e.g., AWS Step function, Microsoft Azure Logic Apps), and we utilized them in the workload.

Using the outcomes of the featurization workload, the *modeling* workload applies the logistic regression algorithm to build a model that predicts reviews' sentiment scores by using the Python *scikit-learn* package. The *ML Model Training* workloads need to access large-size datasets that are available in a shared block storage; moreover, these workloads are generally CPU, memory, and network intensive.

ML Model Serving After building a model, it has to be served for arbitrary inputs to make prediction. In FunctionBench, we provided four types of inference scenario. First, the *face detection* uses the CascadeClassifier to annotate faces in a video stream utilizing the OpenCV library. In the *logistic regression* workload, we utilize a model built in the ML model training step, and it takes users' review text and predicts the sentiment score.

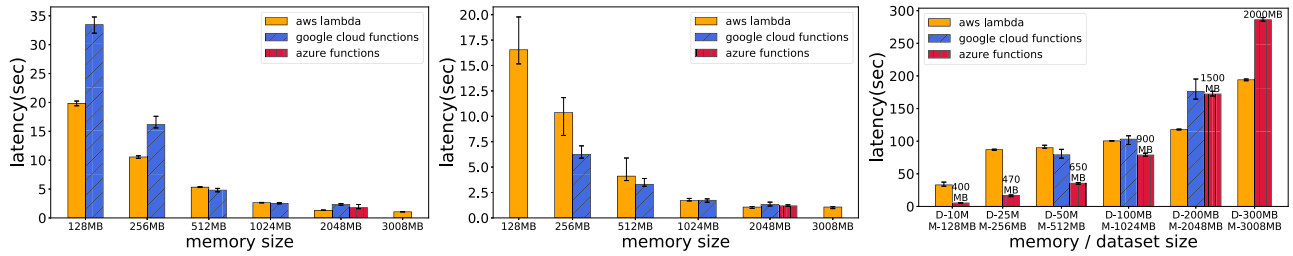
To provide a deep learning model inference, we added an image classification model of SqueezeNet [3], which achieves an impressive accuracy on an ImageNet with 50x fewer parameters than the state-of-the-art model. It is implemented with Python *Tensorflow Keras*. Attempts to import other CNN models on function run-times proved abortive due to limited memory size. Considering that the cost of using function service is proportional to the RAM usage, using a compact model with less RAM usage is recommendable in running the service on a FaaS execution environment. We also added a words generation model using a RNN implemented with *PyTorch*. Overall, the *ML Model Serving* workloads impose relatively lesser resource usage overhead than training workloads.

III. Evaluating With FunctionBench

We run the proposed FunctionBench on various cloud computing services. We upload all source codes customized for different cloud services to the public website. It should be noted that the purpose of the experiments is not to compare the performance of various cloud function services, but to present the applicability of the proposed workloads.

Figure 1a and 1b respectively shows the latency to complete video processing and model serving workloads on AWS, Google, and Microsoft cloud services. It is well-known that the cost of using function services is proportional to the allocated memory size and running-time; moreover, we assume

[2]https://snap.stanford.edu/data/web-FineFoods.html



(a) video processing (b) model serving (c) model training - weak scalability

Fig. 1: Evaluation of FunctionBench workloads on various cloud services

users understand different behaviors of cloud services. In the experiment, we vary the allocated memory size of the function run-time, which is shown in the horizontal axis. In contrast to AWS and Google cloud service, users cannot set memory size in the Microsoft cloud service; hence, we show the latency of Microsoft service only at the pre-configured 2GB memory bar. Missing bars in the figures indicate that the cloud service could not complete the given workload with the allocated memory size (with the exception of Microsoft).

According to cloud service providers, configuring more RAM to the function run-time shortens the workload response time; however, the ratio of improvement varies for different providers. Comparing Google cloud and AWS, AWS shows better performance than Google cloud overall including experimental results that are not presented in this paper due to space limitation even when the configured RAM size is the same. Contrary to AWS, Google cloud function service could not complete a given task while AWS could do. However, for model serving workloads, when Google cloud can complete a task, it provides better performance than AWS. Different from micro-benchmarks that exclusively evaluate different resources, the proposed workloads utilizes CPU, memory, disk IO, and network resources together at different degrees, as shown in Table I. Thus, the different underlying infrastructure configurations that are abstracted from users can impact the overall performance. To accurately compare different function services, we recommend that the evaluations should be conducted using the realistic workloads provided in this paper.

Also, to demonstrate the weak-scalability of cloud services for the model training workload, we increase both the input dataset and configured memory size to measure function response time. The horizontal axis of Figure 1c shows the dataset size (D-) and the configured memory size (M-). The memory size of the Microsoft function is maxed at 2GB, but the cost is calculated based on the real usage of the memory. To compare three cloud function services, we indicated the actual memory usage of Microsoft function invocation on top of the corresponding bars. As observed from the figure, based on the performance and cost, none of the cloud function services can be considered as the best. Users can monitor the status of functions in order to better configure the functions. This observation concurs with the challenges in cloud resource configuration and necessitated autonomic configuration [7].

IV. Conclusion and Future Work

We presented FunctionBench, a suite of workloads, which targets the evaluation of various cloud function services and new algorithms. In addition to micro-benchmarks which are widely in-use in recent times, FunctionBench provides realistic data-oriented applications with its source code customized for major public cloud service vendors (AWS, Google, and Microsoft). We are positive that this contributions would broaden the applications areas of FaaS execution model and facilitate research progress in the relevant fields.

Currently, FunctionBench is publicly available, and we shall expand the workload scenarios based on the assistance and feedback received from the research community.

Acknowledgements

This work is supported by the National Research Foundation of Korea Grant funded by MSIP (No. NRF-2015R1A5A7037615 and NRF-2016R1C1B2015135), the ICT R&D program of IITP (2017-0-00396), and the AWS Cloud Credits for Research program.

References

[1] L. Feng et al. “Exploring Serverless Computing for Neural Network Training”. In: *IEEE Cloud 2018*.

[2] Joseph M. Hellerstein et al. “Serverless Computing: One Step Forward, Two Steps Back”. In: *CIDR 2019*.

[3] Forrest N. Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡1MB model size”. In: *CoRR 2016*.

[4] Youngbin Kim and Jimmy Lin. “Serverless Data Analytics with Flint”. In: *IEEE CLOUD 2018*.

[5] H. Lee, K. Satyam, and G. Fox. “Evaluation of Production Serverless Computing Environments”. In: *IEEE CLOUD 2018*.

[6] Edward Oakes et al. “SOCK: Rapid Task Provisioning with Serverless-Optimized Containers”. In: *USENIX ATC 2018*.

[7] Myungjun Son and Kyungyong Lee. “Distributed Matrix Multiplication Performance Estimator for Machine Learning Jobs in Cloud Computing”. In: *IEEE Cloud 2018*.

[8] Liang Wang et al. “Peeking Behind the Curtains of Serverless Platforms”. In: *USENIX ATC 2018*.