Reproducibility Report for ACM SIGMOD 2023 Paper: "Using Cloud Functions as Accelerator for Elastic Data Analytics"

JUNCHANG WANG, WEIJIAN GUO, Nanjing University of Posts and Telecommunications, China
DOUGLAS B. RUMBAUGH, The Pennsylvania State University, United States
HUAN LI, Zhejiang University, China
HAOQIONG BIAN, Renmin University of China, China

We have reproduced the key findings of the paper as well as the key performance trends reported in its evaluation section. Our reproduction is performed on an AWS testbed provided by the authors.

1 INTRODUCTION

The original paper [1] studies the pros and cons of virtual machines (VMs) and cloud functions (CFs) when processing queries. The former is cost-effective, and the latter is more elastic. Based on the gained insight, the authors present Pixels-Turbo, a hybrid query engine that leverages both VMs and CFs. Pixels-Turbo by default operates on a scalable VM cluster, and leverages CFs on-demand to handle unpredictable workload spikes.

2 SUBMISSION

The paper is accompanied by the source code and a detailed guide for deploying the code. Even though the authors do not provide a single script to reproduce the experiments of the original paper (because cloud platforms vary in manipulation details), they provide the following resources.

- Source code that is also available at https://github.com/pixelsdb/pixels,
- README.md that describes the software and hardware requirements,
- INSTALL.md that describes how to build and deploy Pixels (the storage engine and metadata service) and Pixels-Turbo (the query engine on top of Pixels) on AWS EC2, and
- TPC-H.md that describes how to evaluate the performance of Pixels-Turbo on TPC-H.

3 HARDWARE AND SOFTWARE ENVIRONMENT

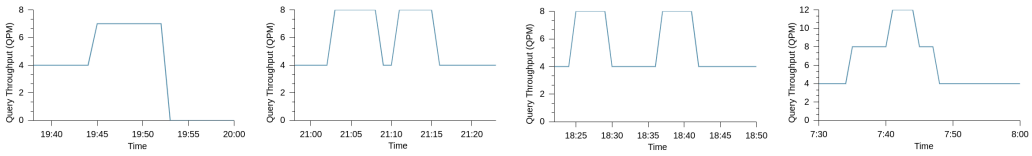
Due to the expense of running the required platform, we obtained an AWS account from the authors for reproduction. In our evaluation, we used m5.8xlarge EC2 VMs as the coordinator and configured the auto-scaling group to use m5.8xlarge. For CFs, each lambda instance is equipped with 10GB of memory.

To reduce costs, we used a 100GB TPC-H dataset. In the experiments, we set the VM scaling-out threshold (of query concurrency) to 3, and the scaling-in threshold to 0.75 (average in 5 minutes). For the VM cluster, we started one worker on the coordinator node, such that there is at least one VM worker in the cluster, even if the scaling manager scales to zero.

4 REPRODUCIBILITY EVALUATION

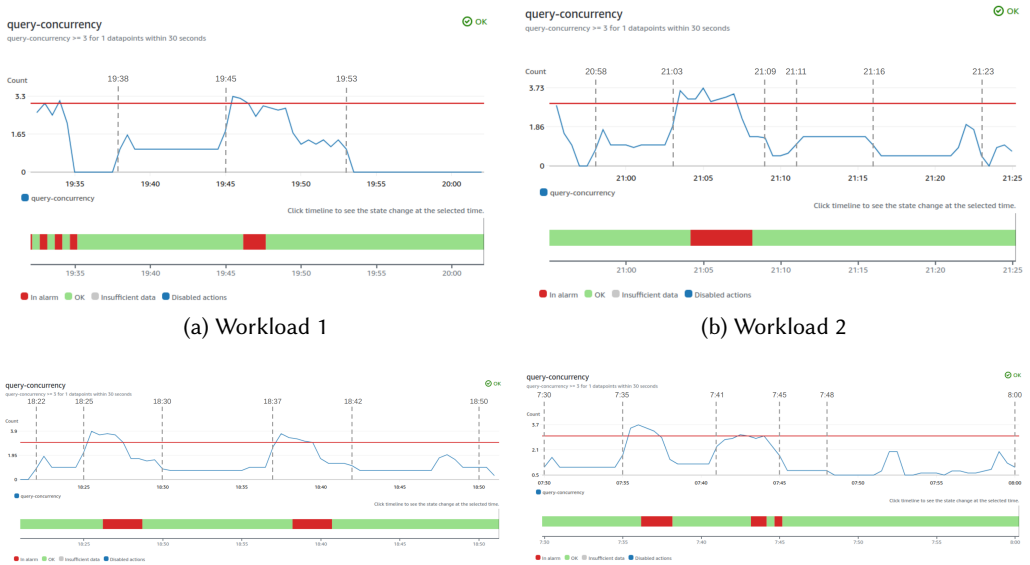
4.1 Process

We first initiated an etcd instance to record metadata and launched a Pixels daemon process to provide metadata service for query processing. Then, we utilized Pixels-Turbo as the query engine to handle user query inputs. In Pixels-Turbo, Trino is used as the query frontend and the MPP query executor in the VM cluster.



(a) Workload 1 (b) Workload 2 (c) Workload 3 (d) Workload 4

Fig. 1. Workloads used in the evaluation.



(a) Workload 1 (b) Workload 2

(c) Workload 3 (d) Workload 4

Fig. 2. Query concurrency when evaluating the workloads.

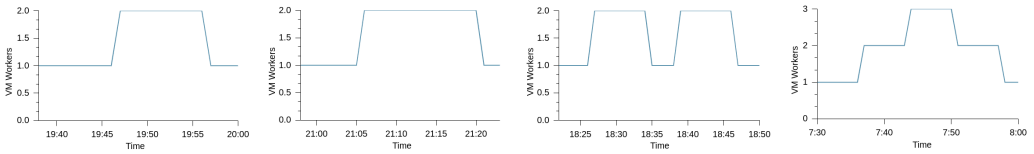
4.2 Results

We reproduced the experiments in Section 6 of the original paper [1]. Since Figure 9 of the original paper is key to the technical contribution of this paper (i.e., when workload spikes arrive, Pixels-Turbo is elastic and cost-efficient), we present our evaluation results (excerpt) as follows.

We created four workloads (as shown in Figure 1) that respectively embody the four workload patterns shown in Figure 8 of the original paper. We used a JDBC client to submit the workloads at different time points that is shown in the x-axis.

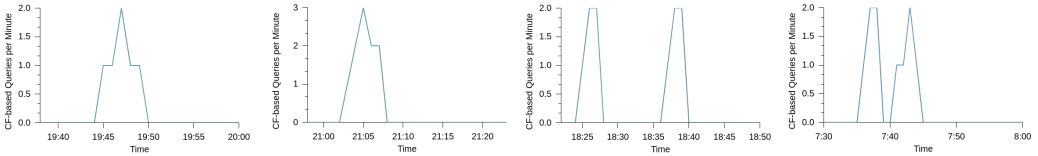
Figure 2 shows the screenshots of AWS CloudWatch showing query concurrency when different workloads are submitted. For each workload, Figures 3 and 4 show the number of VM workers in the VM cluster and the number of queries pushed down into CFs, respectively.

We use workload 1 as an example to discuss how Pixels-Turbo manipulates CFs/VMs on-demand. Figure 1a shows that a workload spike arrives at 19:45, incurring the rise of query concurrency shown in Figure 2a. When the query concurrency rises above the scaling-out threshold (i.e., the red line of value 3), Pixels-Turbo triggers the invocation of the CF workers and the scaling-out of the VM cluster. As shown in Figure 3a, in about 2-3 minutes, a new VM worker is created and added into



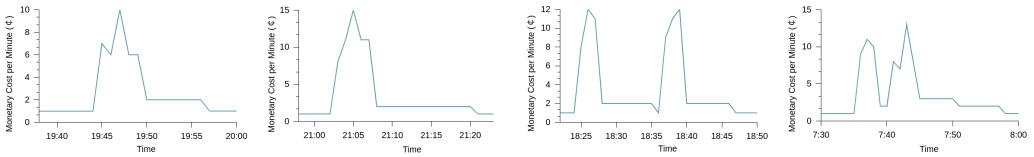
(a) Workload 1 (b) Workload 2 (c) Workload 3 (d) Workload 4

Fig. 3. Number of VMs when evaluating the workloads



(a) Workload 1 (b) Workload 2 (c) Workload 3 (d) Workload 4

Fig. 4. Number of queries executed in CFs when evaluating the workloads



(a) Workload 1 (b) Workload 2 (c) Workload 3 (d) Workload 4

Fig. 5. Monetary cost when evaluating the workloads.

the VM cluster. Meanwhile, during 19:45-19:49, as the new VM worker is in initializing status and cannot reduce the query concurrency, we observe (in the query log) 6 queries are pushed down to the CF workers running in AWS Lambda (see Figure 4a). This makes the monetary cost comparably high in these 4 minutes, as shown in Figure 5a. This is necessary to ensure reasonable query performance. After 19:53, the workload is stopped, and the query concurrency drops to 0 in a while. As shown in Figure 3a, the new VM worker created during the workload spike is released at 19:57. By dynamically creating CFs and VMs, Pixels-Turbo improves the query processing performance and reduces the query concurrency below the red line, as shown in Figure 2a. The expense of Pixels-Turbo is high only during the first few minutes of the workload spike (i.e., from 19:45 to 19:49). Such results have similar trends to Figure 9a of the original paper.

In this experiment, the average query latency is 15.85 seconds. Within the first period (19:38-19:45), in which the workload is stable, the average query latency is 14.32. Hence the invocation of the CF workers and the scaling of the VM cluster do not significantly affect the query latency.

Our evaluation results on other workloads have similar trends to Figure 9 of the original paper, which demonstrates that when workload spikes arrive, Pixels-Turbo can dynamically manipulate its VMs/CFs, improving the query processing elasticity and reducing monetary costs, as described in the original paper.

5 SUMMARY

All core findings and performance trends turned out to be reproducible on AWS. In particular, Pixels-Turbo's capability to deal with query spikes is reproducible.

ACKNOWLEDGMENTS

Huan Li would like to thank Hongwei Yuan and Lei Duan from Zhejiang University for their collaborative efforts in discussing and evaluating the reproducibility of the code.

REFERENCES

[1] Haoqiong Bian, Tiannan Sha, and Anastasia Ailamaki. 2023. Using Cloud Functions as Accelerator for Elastic Data Analytics. *Proceedings of the ACM on Management of Data* 1 (2023), 1 – 27.