Serverless Query Processing on a Budget

William Ma
University of California, Berkeley
williamma@berkeley.edu

ABSTRACT

Relational query processing is an ideal candidate for serverless computation with its stateless, idempotent, and short-lived properties. However, current serverless offerings for query processing neither provide millisecond-based pricing nor allow users to optimize the cost of their queries. To have tradeoffs between the cost and performance of their queries, users are limited to using traditional serverful approaches, which we demonstrate to be 50% slower at approximately the same cost as serverless approaches. We propose a model that will allow service providers to dynamically provision clusters to achieve their users' desired time-cost tradeoffs.

CCS CONCEPTS

• **Information systems** → **Query planning**; **MapReduce-based systems**.

KEYWORDS

Serverless, Query Planning, Spark

ACM Reference Format:
William Ma. 2020. Serverless Query Processing on a Budget. In *2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3318464.3384410

1 INTRODUCTION

Cloud computing achieved widespread adoption over the past decade, which has freed users from the burden of maintaining their own clusters and allowed them to provision their clusters on-demand. Similar to cloud computing, *serverless computation* has freed users from provisioning their own clusters and allows them to achieve "web-scale" without a thought for how to optimally provision their clusters. As serverless computation gains momentum, service providers are introducing new services that leverage serverless, such as AWS Athena [1] and GCP BigQuery [3], which allow users to query remote stores without provisioning their own compute clusters.

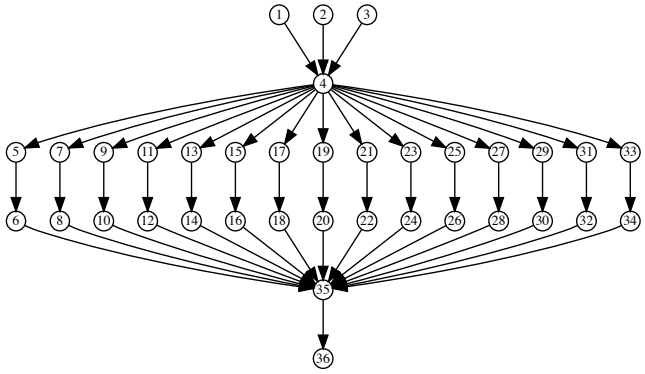

Figure 1: Spark execution graph for a sample TPC-DS query. Clearly there is an opportunity for large performance gains and cost savings with elasticity.

Unlike other serverless offerings (e.g., AWS Lambda [2], GCP Function [4]) that offer millisecond-based pricing, these services are priced on the amount of data accessed by the query, which is not optimal for the service provider or the user. The only way for the user to reduce the amount of data accessed is to approximate queries, which is not always desirable. Furthermore, pricing based on the amount of data accessed does not reflect the actual cost, which is the product of the wall-clock time, the cost of the compute node used, and the number of compute nodes used. Thus, the user pays the same amount for running two select queries (e.g., `SELECT ... FROM TABLE_1` and `SELECT ... FROM TABLE_2`) and one join query (e.g., `SELECT ... FROM TABLE_1, TABLE_2`).

Our key observation is that a pricing scheme based on wall-clock time will allow the service provider to have a fairer pricing scheme. Additionally, this will allow users to trade off run time for cost.

We propose a model based on Spark [12] that provides the optimal time-dollar cost tradeoff to the user and show preliminary results of a 50% run time reduction with only a 2% increase in cost from using serverless computation.

2 METHODS

In order to determine the time-cost tradeoff of user's queries, we developed a model that takes the user's queries and a

SIGMOD '20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06.
https://doi.org/10.1145/3318464.3384410

sample of their data and returns the time-cost tradeoff curve to the user and the dynamic cluster provisioning sizes of the query execution to the service provider.

Because Spark is not serverless, we build our model based on a simulated version of Spark that always has access to nodes ready to load remote data and execute tasks without warming up the JVM and can have multiple Spark drivers running simultaneously, following the standard of many cloud service providers. We believe that these are reasonable requirements for cloud service providers to meet.

With this, we build a model by simulating the query execution using the data gathered when running the user's query on sampled data. Without a detailed discussion on the internals of Spark, the nodes in the Spark execution graph in figure 1 represents a different stage. Each stage in a Spark query can be executed independently given that its parents have finished executing. From figure 1, stages 5, 7, . . ., 33 can be run in parallel, which Spark does but it is clear that additional resources during the execution would improve performance. Additionally, each stage consists of multiple tasks that can all be run independently and in parallel. The number of tasks that can be run in parallel is limited by the number of CPUs available in the Spark cluster. We assume that the duration of task t (denoted t_d) follows $t_d \sim Gamma(a, b)$, where a and b are estimated using the data we collected on the sample run. We model task runtimes as Gamma random variables since the Gamma distribution is nonnegative and exhibits long-tail behavior. Thus, this model can efficiently simulate the execution of the Spark query on a cluster that can dynamically change at any step of the execution.

3 RESULTS

To demonstrate the performance gains achievable through serverless computation, we run a Spark script that executes common data science queries on the NASA HTTP server data [5]. Although the regular NASA HTTP data is only 200 MB, we replicated it 25× to reach a size of 5 GB and stored it in AWS S3. The executed the script on AWS EC2 m5.large (2 CPU and 4 GB RAM) instances with 2 to 64 nodes. Though the cost of a m5.large node is $0.09 per hour, we use a cost of $1 per second for ease of comprehension. To simulate the overhead associated with a serverless Spark, we assume that it takes 125 ms to set up a new driver to run tasks.

Demonstrating the effectiveness of the provisioned sized of the cluster in figure 2, we found the cost of running the script can vary anywhere from $4,800 to $2,661, with the cheapest provisioned cluster (6 nodes) running the script in 443 seconds (cost $2,661) and the fastest provisioned cluster (64 nodes) running the script in 75 seconds (cost $4,800). Thus, the ideal provisioned cluster can lead to savings of 50%. However, by exploiting dynamic scalablility of serverless computing, we find that for a budget of $2,750, we can reduce the run time by 50% (381 seconds to 190 seconds) with only a 2% increase in the cost ($2,670 to $2,734).

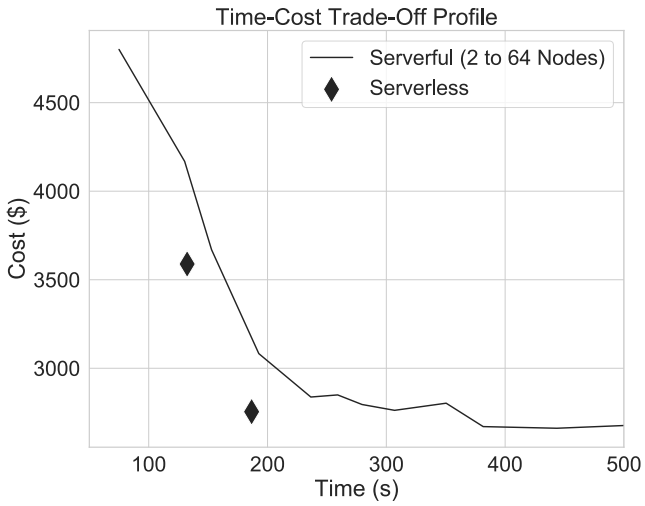

Figure 2: Time-Cost tradeoffs for serverful and serverless computation. Serverless clearly reduces the time and cost of query execution.

Despite the idealized setting, these results demonstrate the promise of the performance gains of serverless computation.

4 RELATED WORK

Other system query optimizers, like Cherrypick [6], Ernest [11], and Elastisizer [9], attempt to determine the optimal provisioned cluster. These papers neither determine the execution of the query at a task granularity nor take into consideration that the cluster can change during the execution of a query.

From the systems-side, there has been explorations of building systems for serverless computation. These either explore the map-reduce paradigm within a serverless context [8, 10] or focus making the communication of serverless computing more efficient [7].

5 FUTURE WORK AND CONCLUSION

In conclusion, we have shown that serverless offerings have the potential to be 2× faster than serverful offerings at approximately the same price. We are also developing a model that takes in an arbitrary query and sampled data to return a time-cost tradeoff profile with corresponding cluster provisioning. This will provide users with an understanding of how their queries will perform at various price points and service providers with a dynamic cluster provisioning for the user's desired performance.

REFERENCES

[1] 2019. AWS Athena. https://aws.amazon.com/athena/

[2] 2019. AWS Lambda. https://aws.amazon.com/lambda/

[3] 2019. GCP BigQuery. https://cloud.google.com/bigquery/

[4] 2019. GCP Function. https://cloud.google.com/functions/

[5] 2019. NASA-HTTP. ftp://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[6] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*. 469–482. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/alipourfard

[7] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. 2019. Cirrus: a Serverless Framework for End-to-end ML Workflows. In *Proceedings of the ACM Symposium on Cloud Computing - SoCC '19*. ACM Press, Santa Cruz, CA, USA, 13–24. https://doi.org/10.1145/3357223.3362711

[8] Joseph M. Hellerstein, Jose Faleiro, Joseph E. Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. 2018. Serverless Computing: One Step Forward, Two Steps Back. *arXiv:1812.03651 [cs]* (Dec. 2018). http://arxiv.org/abs/1812.03651 arXiv: 1812.03651.

[9] Herodotos Herodotou, Fei Dong, and Shivnath Babu. 2011. No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics. In *Proceedings of the 2nd ACM Symposium on Cloud Computing - SOCC '11*. ACM Press, Cascais, Portugal, 1–14. https://doi.org/10.1145/2038916.2038934

[10] Qifan Pu, Shivaram Venkataraman, and Ion Stoica. 2019. Shuffling, Fast and Slow: Scalable Analytics on Serverless Infrastructure. In *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019*. 193–206. https://www.usenix.org/conference/nsdi19/presentation/pu

[11] Shivaram Venkataraman, Zongheng Yang, Michael J. Franklin, Benjamin Recht, and Ion Stoica. 2016. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*. 363–378. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/venkataraman

[12] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*. https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets