Cost-efficiency and Performance Robustness in Serverless Data Exchange

David Justen
Hasso Plattner Institute
University of Potsdam
david.justen@student.hpi.de

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

KEYWORDS

serverless, data exchange, data skew

ACM Reference Format:
David Justen. 2022. Cost-efficiency and Performance Robustness in Serverless Data Exchange. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3514221.3520248

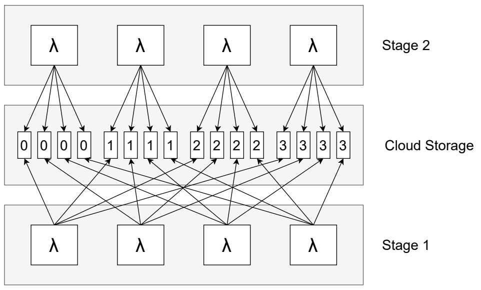

Figure 1: Fully-meshed serverless data exchange

1 PROBLEM AND MOTIVATION

Enterprises increasingly store their sporadically used data on cheap, elastic cloud storage to save costs. Analytical workloads on that data often appear in infrequent bursts presenting a high disparity in input data sizes. Using conservatively over-provisioned compute resources for this class of workloads is not cost-efficient as a fixed amount of resources only matches steady demand. Elastic query processors with a serverless architecture resolve this issue by running workers in cloud functions [3][9][10]. These systems can start thousands of functions within seconds, enabling elasticity down to query pipeline granularity. Moreover, serverless query processors come at no cost for idle times.

While cloud functions are well-suited for embarrassingly parallel tasks [7], they struggle with blocking operators such as joins, sorts, and aggregations as they require data exchange between the workers. Since cloud functions do not support direct communication, they need an intermediary storage layer to exchange ephemeral data. Elastic cloud storage services typically offer low prices to store large amounts of data but charge customers by the number of requests they make to the service [2][5][8]. These costs can quickly dominate the overall query price as data exchanges with thousands of cloud functions can induce millions of requests. Another challenge for serverless data exchange with a large number of workers is data skew, leading to severe input size imbalance and highly divergent worker processing times.

In contrast to single-node and shared-nothing systems, run duration does not solely define cost-efficiency for a serverless query processor. A serverless data exchange approach must consider pricing as a second aspect as its cloud infrastructure induces costs depending on the service usage patterns. Therefore, serverless data exchange must be fast while causing little service cost. Apart from cost-efficiency, consistent query performance (i.e. performance robustness) is critical to data analytics users [1]. Thus, the exchange must process diverse sets of data with predictable processing times and low variance.

2 RELATED WORK

Lambada [9] and Starling [10] are serverless query processors introducing intermediate file formats and multi-stage data exchange strategies to reduce request costs. However, they do not further investigate the price-performance trade-offs inherent to these approaches. Moreover, both systems only process uniform data in their evaluations, disregard skewed data, and do not address performance robustness issues. Locus [11] partially circumvents the request pricing issue by deploying in-memory stores for their data exchange. Thereby, they introduce a pre-provisioned resource with running costs, limit the storage elasticity for intermediate results, and eliminate the serverless pay-per-query quality. Boxer [12] is a system using TCP hole punching to enable direct communication between cloud functions. It allows the functions to exchange data without request costs or additional long-standing resources. On the other hand, direct communication makes cloud functions more vulnerable to failure due to load imbalance. Furthermore, it impedes a system's ability to rescale compute resources spontaneously within a query. Wawrzoniak et al. also admit that their system is at a prototypical stage and still lacks stability.

3 APPROACH

To address the challenges above, we introduce a cost-efficient serverless data exchange approach with robust performance. The approach is based on the exchange operator [6], in which producer

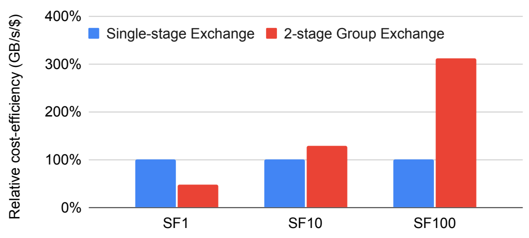

Figure 2: Relative cost-efficiency in GB/s/$ for different methods of data exchange on the TPC-H lineitem table with different scale factors. SF1 uses ten concurrent cloud functions, SF10 uses 100, and SF100 uses 400.

tasks pre-partition an input table and assign each partition to one consumer task. Afterwards, the consumer tasks can concurrently execute a blocking operator on distinct data subsets. For a serverless data exchange (cf. Figure 1), a stage of cloud functions λ performs the producer tasks by executing partition operators and exporting the partitions to a shared cloud storage layer. After that, each cloud function in the second stage imports its assigned partitions and executes the blocking operator on them.

For cost-efficiency, the data exchange must minimize the number of requests to the storage service. To minimize writes, our approach introduces a data layout to store multiple partitions in one file so that each worker only must make one write request. The export operator writes one partition after the other to the file and stores the partition row-ranges in the file tail. Then, workers in the next stage execute the import operator, which reads the file tail to retrieve the row-range for their target partition. Afterwards, the operator reads that range from the input file to import the partition.

Analogous to Lambada [9] and Starling [10], our approach uses a multi-stage group exchange to reduce the number of reads. This strategy reduces the request number by adding a stage that pre-combines groups of partitions. Thereby, workers in the following stage only read one file per group instead of one file per worker in the previous stage. The multi-stage group exchange drastically reduces the number of requests but introduces cloud function run time and costs with every additional stage. Therefore, this work investigates the strategy's inherent price-performance trade-off. For example, Figure 2 displays the cost-efficiency of a single-stage exchange and a 2-stage group exchange on the TPC-H `lineitem` table with different scale factors and cloud function counts. It shows that the group exchange is only preferable for larger amounts of data as the request costs only then become significant.

Apart from the cost-efficiency aspect, the data exchange approach must also show a robust performance. During the exchange process, skewed data can lead to unevenly sized partitions resulting in unevenly balanced input load and, therefore, diverging processing times between cloud functions. As a load re-balancing system, our work introduces an over-partitioning strategy. Instead of creating one partition per consumer, each producer creates multiple partitions and returns their sizes to the scheduler. Knowing the partition sizes, the scheduler then assigns sets of partitions to each worker to even out input size differences. With the over-partitioning feature, the data exchange approach re-balances the input load and reduces the run time divergence between cloud functions. Thus, it improves the performance consistency for queries on data with different levels of skew.

Despite the over-partitioning feature, partitions still may diverge in size, e.g., if a large fraction of a column consists of a single value. Our work approaches this issue with a partitioned cartesian join using the elasticity of serverless systems to ensure robust performance. Figure 3 shows how additional cloud functions can help break up large partitions to enable distributed join processing. The approach divides a partition p_0 from the left input table R and the right input table S into n_R and n_S sub-partitions. Then it invokes $n_R * n_S$ cloud functions so that each joins one sub-partition from the left with one from the right side. Note that n_R or n_S can be 1 if only one side has a large partition. In that case, each cloud function imports a full partition from one side and joins it with a sub-partition from the other side. With the serverless partitioned cartesian join, each worker in the join stage processes a similar amount of data. Thus, it improves the performance robustness but quadratically increases the number of cloud functions, raising compute costs. Consequently, our work analyzes this trade-off between price and performance.

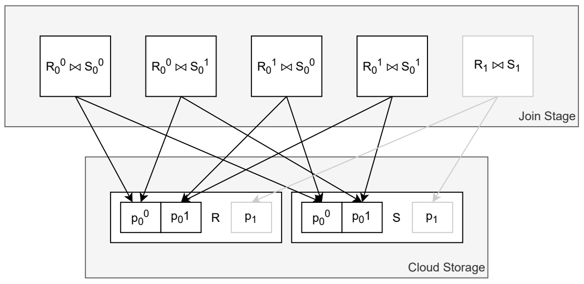

Figure 3: Serverless partitioned cartesian join. Each highlighted cloud function joins one sub-partition on the left with one on the right side.

4 CONTRIBUTION

In this work, we contribute a serverless data exchange approach built upon the Skyrise elastic query processor [3]. The approach includes a file layout for intermediate data and a multi-stage group exchange method to improve cost-efficiency. Furthermore, our work introduces the partitioned cartesian join and an over-partitioning strategy for robust performance with evenly distributed input loads. We evaluate these features by processing data exchanges on TPC-H and JCC-H [4] tables with different scale factors and synthetic data with different skew characteristics. Moreover, the final work evaluates multi-stage group exchange and over-partitioning separately and combined to indicate sweet spots for cost-efficiency and performance robustness. Finally, it investigates the price-performance trade-off inherent to the multi-stage exchange and the serverless cartesian join.

ACKNOWLEDGMENTS

I would like to thank Thomas Bodner, Stefan Halfpap, Dr. Daniel Ritter, and Dr. Michael Perscheid for their support.

REFERENCES

[1] Ankur Agiwal et al. 2021. Napa: Powering Scalable Data Warehousing with Robust Query Performance at Google. In *PVLDB*, Vol. 14 (12). 2986–2998.
[2] Amazon Web Services. 2021. Amazon S3 pricing. https://aws.amazon.com/s3/pricing/
[3] Thomas Bodner. 2020. Elastic Query Processing on Function as a Service Platforms. In *Proceedings of the VLDB PhD Workshop*.
[4] Peter Boncz, Angelos-Christos Anatiotis, and Steffen Kläbe. 2018. JCC-H: Adding Join Crossing Correlations with Skew to TPC-H. In *Performance Evaluation and Benchmarking for the Analytics Era*. 103–119.
[5] Google. 2021. Cloud Storage Pricing. https://cloud.google.com/storage/pricing
[6] Goetz Graefe. 1990. Encapsulation of Parallelism in the Volcano Query Processing System. In *SIGMOD*.
[7] Joseph M. Hellerstein et al. 2019. Serverless Computing: One Step Forward, Two Steps Back. In *9th Biennial Conference on Innovative Data Systems Research, CIDR*.
[8] Microsoft. 2021. Azure Blob Storage pricing. https://azure.microsoft.com/en-us/pricing/details/storage/blobs/
[9] Ingo Müller, Renato Marroquín, and Gustavo Alonso. 2020. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. In *SIGMOD*.
[10] Matthew Perron et al. 2020. Starling: A Scalable Query Engine on Cloud Functions. In *SIGMOD*.
[11] Qifan Pu, Shivaram Venkataraman, and Ion Stoica. 2019. Shuffling, Fast and Slow: Scalable Analytics on Serverless Infrastructure. In *NSDI*. 193–206.
[12] Michal Wawrzoniak et al. 2021. Boxer: Data Analytics on Network-enabled Serverless Platforms. In *CIDR*.