Performance Evaluation of Stream Data Processing Systems

ABSTRACT

Over the past years, Stream data processing is gaining compelling attention both in industry and in academia due to its wide range of applications in various use-cases. To fulfil the need for efficient and high performing Big data analytics, numerous open source stream data processing systems were developed. Processing data with high throughput while retaining low latency is key performance indicator for such systems. In this paper, we propose a benchmarking system to evaluate the performance of stream data processing systems, Storm, Spark and Flink in terms of indicators shown above. More specifically, the latency of windowed aggregation and windowed join operators is evaluated jointly with the throughput of a system.

1. INTRODUCTION

Streaming data processing has been gaining significant attention due to its wide range of uses in Big data analytics. The main reason is that processing big volumes of data periodically is not enough anymore and data has to be processed fast to enable fast adaptation and reaction to changed conditions. Several engines are widely adopted and supported by open source community, such as Apache Storm [2], Apache Spark [3] , Apache Flink [1] and etc.

2. RELATED WORK

3. PRELIMINARY AND BACKGROUND

3.1 Storm

general info, windowing, partitioning, join.

3.2 Spark

general info, windowing, partitioning, join.

3.3 Flink

general info, windowing, partitioning, join.

4. BENCHMARK SYSTEM DESIGN

We keep overall design of benchmark simple and akin to the systems under to be tested. The stream data processing system connects to predefined number of socket input data sources. Each input data source has its own data generator and queue where tuples are put after generation. When the stream data processing system pulls data from socket server, serving thread sends the data from queue rather than directly from data generator.

Figure 1 shows the overall intuition of the benchmark system.Firstly, the data generator, at the left side of the figure, spawns the tuples and appends the current timestamp as a separate field. Secondly, the generated tuples wait in queue until the stream data processing system pulls them. Queue is based on FIFO semantics. Thirdly, the stream data processing system combines the tuples from different sockets by uniting streams. Depending on the nature of operator to be tested, there can be several unions of streams. For example, windowed aggregation is single input stream operator but join operator accepts two streams as an input. Fourthly, the operator to be tested computes the data in windows. In our case, we test windowed aggregation and windowed join operators. Finally, the next operator calculates the latency of a tuple by subtracting event timestamp from current timestamp.

Proper handling tuple's timestamp fields while joining or aggregating is crucial. Lets say, t_a^m is a tuple from socket stream a, residing in window m and t_a^m denote the value of kth field. restrictions..

4.1 Key Performance Indicators

5. EVALUATION

5.1 Keyed Windowed Aggregations

scale up/down
window size increase/decrease
batch siez increase/decrease

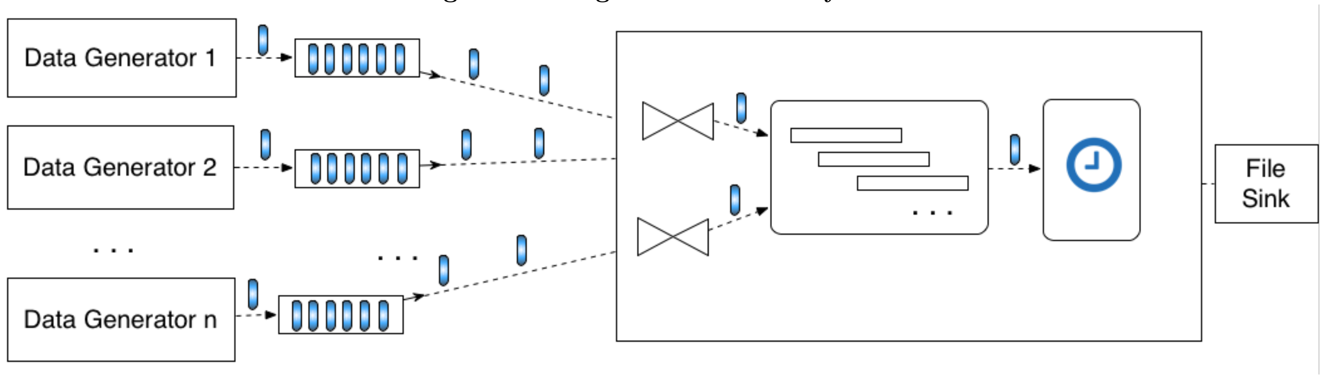
5.2 Joins

same as above

6. CONCLUSIONS

7. ACKNOWLEDGMENTS

Figure 1: Design of benchmark system.



8. ADDITIONAL AUTHORS

Additional authors: John Smith (The Thørväld Group, `jsmith@affiliation.org`), Julius P. Kumquat (The Kumquat Consortium, `jpkumquat@consortium.net`), and Ahmet Sacan (Drexel University, `ahmetdevel@gmail.com`)

9. REFERENCES

[1] P. Carbone, S. Ewen, S. Haridi, A. Katsifodimos, V. Markl, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *IEEE Data Engineering Bulletin*, 2015.

[2] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, et al. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 147–156. ACM, 2014.

[3] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In *Presented as part of the*, 2012.

9.1 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references).

APPENDIX

You can use an appendix for optional proofs or details of your evaluation which are not absolutely necessary to the core understanding of your paper.