Stream Benchmarks

Jeyhun Karimov
DFKI GmbH, Germany

Synonyms

Stream performance evaluation

Definitions

Stream benchmarks deal with performance evaluation techniques and define related metrics for stream data processing systems.

Historical background

Lee et al (1997) initiated one of the first works in the related area, MediaBench, by evaluating and synthesizing multimedia and communications systems. Abadi et al (2003) and Motwani et al (2003) pioneered one of the first stream data processing systems, Aurora and STREAM respectively. The need to compare the performance characteristics of streaming systems relative to each other and to alternative (e.g., Relational Database) systems endorsed the development of Linear Road Benchmark, Arasu et al (2004).

Foundations

Stream data processing is key when the large volumes of input data has to be processed fast to quickly adapt and react to changes. Therefore, stream data processing has gained significant attention.

The main intuition behind stream benchmarks is to define a standard to compare streaming systems, which has different characteristics, in a various use-cases. Stream benchmarks simulate an environment with different workloads and analyze the behavior of the systems to be tested. The more similar the benchmark to the real production

environment, the more realistic and valuable it is.

In the high level, stream benchmark framework consists of two main components: system under test and driver. Driver is responsible for simulating the production environment w.r.t. a given use-case. System under test is actual tested streaming system which can be out-of-the-box or tuned.

Accurately representing the system under test is important when designing benchmarks. Schroeder et al (2006) categorize benchmarks into closed, open and partly-open models. In a closed system model, the input arrivals are are triggered after the completion of the previous input processing and some thinking time delay. In an open system model, on the other hand, new input arrivals and the completion of the previous input processing are independent. In a partly-open system model, we specify the settings for which partly-open model behaves like a closed or an open model.

Below, we categorize existing stream benchmarks in literature. Firstly, we talk about the main metrics in stream benchmarks and their definitions. Secondly, we analyze stream benchmarks which concentrate on specific features such as fault tolerance and state management. Lastly, we analyze stream benchmarks built for specific industrial use-cases.

Metrics

Main metrics for stream benchmarks are latency and throughput. Achieving high throughput while preserving low latency is the main goal for streaming systems.

Latency

Defining a standard latency metric definition for streaming systems is a challenging task. One reason is significant architectural and computational differences among stream data processing engines. Another reason is calculating latency for stateful operators in non-trivial.

Chintapalli et al (2016a) use Redis for stateful computations as part of the system under test. The authors calculate the event-time latency by subtracting the window start time and duration time from the last updated time of a particular input record. Perera et al (2016) propose reproducible benchmarks for Apache Spark and Flink on public clouds. The authors does not use a standard latency definition, as, in this case, the latency measurement is experiment and application specific. Lu et al (2014) propose a new stream benchmark by separating the data generator and system under test. The authors put a mediator layer between the two components. They define latency as an average time span from the arrival of a record till the end of processing of the record. Qian et al (2016) also adopted a similar approach. Karimov et al (2018) develop stream benchmark framework that overcome overhead of a mediator layer. The authors show how processing-time latency might mislead when compared with event-time latency.

Throughput

Throughput is another essential metric for stream data processing systems. Similar to latency, measuring and defi-

ning a standard throughput metric for all streaming systems is non-trivial.

Chintapalli et al (2016a) separate the data generator and system under test with an intermediate layer between them. The authors calculate the throughput by configuring the speed of data generator for a specific workload. Lopez et al (2016a), on the other hand, rely on Kafka's sampled throughput rates. Lu et al (2014) measure the overall system under test throughput and throughput per node. The authors define overall throughput metric by count (average count of records per second) and size (average data size in terms of bytes processed per second). Shukla and Simmhan (2016) define throughput as the rate of output tuples emitted from the output operators in a unit time. Dayarathna and Suzumura (2013) define job throughput in two ways. First, the authors measure the time required to process a specific amount of events. Second, the authors measure the number of tuples processed in a given amount of time. The throughput computation is performed based on both time periods. Samosir et al (2016), on the other hand, adopt the throughput metric used in batch processing systems. Karimov et al (2018) propose maximum sustainable throughput throughout the whole experiment.

Benchmarking the energy consumption of stream data processing engines is another important aspect of stream benchmarks. Dayarathna et al (2017) adopt Linear Road benchmark for testing the energy efficiency of S4, Storm, ActiveMQ, Esper, Kafka, and Spark Streaming. The key finding of this work is that better power consumption behaviors in the context of data stream processing systems can be achieved by setting tuple sizes to be moderate and scheduling plans to have balanced system overhead.

Features

Besides focusing on metrics computations, stream benchmarks also concentrate on specific features of stream data processing engines.

Fault tolerance

Lopez et al (2016a) study streaming systems' tolerance to failures by analyzing the system behavior after detecting the failure. The system behavior includes the message losses and latency/throughput change during node failure. Gradvohl et al (2014) categorize fault tolerance behavior in stream data processing systems into replication components, upstream backup, checkpoint, and recovery and analyze the system utilization of these strategies. Mohamed et al (2017) propose a driver which allows programmatic specification of complex fault scenarios. Chauhan et al (2012) measure the systems' tolerance to faults by *i)* measuring the number of events handled and *ii)* checking the number of events that were missed when nodes go down in cluster. Qian et al (2016) adopt Identity workload and consider only one node failure at a time. The benchmark suite collects the performance metrics in node-failure workload and compares it with non-faulty workload.

State management

Linear Road benchmark, Arasu et al (2004), Jain et al (2006), consists of continuous queries which update operator state by processing the incoming stream. Kipf et al (2017) analyze the limitation of efficiently exposing the state to analytical queries for stream data processing systems. The authors compare main-memory databases and streaming engines and propose new methods to advance state management in streaming systems.

Key applications

In this section we categorize stream benchmarks based on different industrial use-cases.

Data mining. Zhang et al (2012) and Le-Phuoc et al (2012) are the pioneers to propose SRBench and LSBench, the first benchmarks for RDF streaming and Linked Stream Data processing. The authors adopt wide range of queries including simple graph pattern matching queries and the ones with complex reasoning tasks. DellAglio et al (2013) propose CSRBench, an extension for SRBench. The authors overcome the main shortcoming of SRBench and LSBench, which is inability to assess the correctness of query evaluation results, by analyzing the operational semantics of the particular processors. Ali et al (2015) address another limitation of SRBench and LSBench, addressing the dynamic application requirements and data-dependent properties. The authors propose the workloads which include fluctuating streaming rates during query execution and changing the application requirements over a some time duration. Implementing, modeling and evaluating the provisioning algorithms for stream processing applications is another related work, in which authors propose VISP Testbed Hochreiner (2017). The toolkit provides a common runtime for stream processing applications.

E-commerce. Teng et al (2017) analyze streaming systems behavior in e-commerce scenarios. The authors provide a data generator, as part of the benchmark suite, with certain user models, which adopt a certain user habits in e-commerce platforms. Tucker et al (2008) propose NEXMark, an extension of XMark Schmidt et al (2001), based on online auction system. Currently, NEXMark is used as a benchmark suite in Apache Beam Buzzwords (2017).

IoT. Shukla and Simmhan (2016) develop a benchmark suite for streaming systems for IoT applications. The authors classify 13 common IoT tasks with functional categories. Moreover, the benchmark suite provides two IoT applications being statistical summarization and predictive analytics. CityBench is the benchmark to evaluate RDF stream processing systems in IoT scenarios Ali et al (2015). The authors use traffic vehicles, parking, weather, pollution, cultural, and library events, with changing event rates and playback speeds as part of the data generator. Shukla et al (2017) extend the existing stream benchmarks in IoT proposing RIoTBench. The benchmark includes 27 common IoT tasks. Moreover, the authors propose four IoT application benchmarks composed from the proposed tasks.

Network. Nazhandali et al (2005) propose stream benchmark for sensor network systems, suitable for sensor

processors. The authors propose new metrics being EPB (Energy Per Bundle) and CFP (Composition Footprint) to evaluate and compare systems under test. Lopez et al (2016b) analyze the performance of Virtualized Network Function for realtime thread detection using stream processing. Wolf and Franklin (2000) propose telecommunication benchmark for network processors. The authors adopt four workloads for data stream processing in telecommunications scenario. Trivedi et al (2016) analyze yet another interesting aspect, the (ir)relavance of network bandwidth to modern streaming engines. The key finding of this paper is that, current streaming engines need significant architectural improvements as they cannot benefit from high bandwidth networks.

Multi-Core Processors. Zhang et al (2017) benchmark the current design of stream data processing engines on multi-core processors analyzing the possible bottlenecks of massively parallel JVM based streaming engines.

CEP. Mendes et al (2009) were among the pioneers to propose a benchmark for CEP systems. The authors provide series of queries to exercise factors such as window size and policy, selectivity and event dimensionality. Mendes et al (2013) propose BiCEP, the domain-specific benchmark suite, to evaluate different performance aspects of event processing platforms. Alevizos and Artikis (2014) adopt existing techniques to analyze widely used Esper system which employs a SQL-based language and RTEC which is a dialect of the Event Calculus.

Machine Learning. Gama et al (2009) propose the a general framework for assessing predictive stream learning algorithms. Gama et al (2013) focus on decision models and develop a benchmark suite to evaluate continuously evolving streaming and to detect and react to realtime input data. Imai et al (2017) utilize a machine learning model to predict the maximum sustainable throughput in streaming systems.

Bottlenecks. When designing a system it is important to detect and avoid bottlenecks. For streaming systems, Chintapalli et al (2016b) show the Zookeeper being a main performance bottleneck for Storm. For stream benchmarks, Friedrich et al (2017) show the limitations of existing benchmark designs and the possible biased results. Moreover, Artisans (2017) make a disclaimer for the original implementation of Chintapalli et al (2016a), showing an intermediate message layer, Kafka, and external state management system, Redis, are actually being a bottleneck for Flink's overall performance.

Cross Reference

Big Stream Processing
Distributed Systems for Big Data
Big Data Analysis

References

Abadi DJ, Carney D, Çetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik S (2003) Aurora: a new model and architecture for data stream management. The VLDB JournalThe International Journal on Very Large Data Bases 12(2):120–139

Alevizos E, Artikis A (2014) Being logical or going with the flow? a comparison of com-

plex event processing systems. In: SETN, Springer, pp 460–474

Ali MI, Gao F, Mileo A (2015) Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets. In: International Semantic Web Conference, Springer, pp 374–389

Arasu A, Cherniack M, Galvez E, Maier D, Maskey AS, Ryvkina E, Stonebraker M, Tibbetts R (2004) Linear road: a stream data management benchmark. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, pp 480–491

Artisans D (2017) Extending the Yahoo! Streaming Benchmark. https://data-artisans.com/blog/extending-the-yahoo-streaming-benchmark, [Online; accessed 1-Nov-2017]

Buzzwords B (2017) Nexmark: Using Apache Beam to create a unified benchmarking suite. https://berlinbuzzwords.de/sites/berlinbuzzwords.de/files/media/documents/nexmark_suite_for_apache_beam_berlin_buzzwords_1.pdf, [Online; accessed 1-Nov-2017]

Chauhan J, Chowdhury SA, Makaroff D (2012) Performance evaluation of yahoo! s4: A first look. In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on, IEEE, pp 58–65

Chintapalli S, Dagit D, Evans B, Farivar R, Graves T, Holderbaugh M, Liu Z, Nusbaum K, Patil K, Peng BJ, et al (2016a) Benchmarking streaming computation engines: Storm, flink and spark streaming. In: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, IEEE, pp 1789–1792

Chintapalli S, Dagit D, Evans R, Farivar R, Liu Z, Nusbaum K, Patil K, Peng B (2016b) Pacemaker: When zookeeper arteries get clogged in storm clusters. In: Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on, IEEE, pp 448–455

Dayarathna M, Suzumura T (2013) A performance analysis of system s, s4, and esper via two level benchmarking. In: International Conference on Quantitative Evaluation of Systems, Springer, pp 225–240

Dayarathna M, Li Y, Wen Y, Fan R (2017) Energy consumption analysis of data stream processing: a benchmarking approach. Software: Practice and Experience 47(10):1443–1462

DellAglio D, Calbimonte JP, Balduini M, Corcho O, Della Valle E (2013) On correctness in rdf stream processor benchmarking. In: International semantic web conference, Springer, pp 326–342

Friedrich S, Wingerath W, Ritter N (2017) Coordinated omission in nosql database benchmarking. In: BTW (Workshops), pp 215–225

Gama J, Sebastião R, Rodrigues PP (2009) Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 329–338

Gama J, Sebastião R, Rodrigues PP (2013) On evaluating stream learning algorithms. Machine learning 90(3):317–346

Gradvohl ALS, Senger H, Arantes L, Sens P (2014) Comparing distributed online stream processing systems considering fault tolerance issues. Journal of Emerging Technologies in Web Intelligence 6(2):174–179

Hochreiner C (2017) Visp testbed-a toolkit for modeling and evaluating resource provisioning algorithms for stream processing applications. strategies 1(6):9–17

Imai S, Patterson S, Varela CA (2017) Maximum sustainable throughput prediction for data stream processing over public clouds. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Press, pp 504–513

Jain N, Amini L, Andrade H, King R, Park Y, Selo P, Venkatramani C (2006) Design, implementation, and evaluation of the linear road benchmark on the stream processing core. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM, pp 431–442

Karimov J, Rabl T, Katsifodimos A, Samarev R, Heiskanen H, Markl V (2018) Benchmarking distributed stream data processing systems. In: ICDE

Kipf A, Pandey V, Böttcher J, Braun L, Neumann T, Kemper A (2017) Analytics on fast data: Main-memory database systems versus modern streaming systems. In: EDBT, pp 49–60

Le-Phuoc D, Dao-Tran M, Pham MD, Boncz P, Eiter T, Fink M (2012) Linked stream data processing engines: Facts and figures. The Semantic Web–ISWC 2012 pp 300–312

Lee C, Potkonjak M, Mangione-Smith WH (1997) Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In: Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture, IEEE Computer Society, pp 330–335

Lopez MA, Lobato AGP, Duarte OCM (2016a) A performance comparison of open-source stream processing platforms. In: Global Communications Conference (GLOBECOM), 2016 IEEE, IEEE, pp 1–6

Lopez MA, Lobato AGP, Duarte OCM, Pujolle G (2016b) Design and performance evaluation of a virtualized network function for real-time threat detection using stream processing

Lu R, Wu G, Xie B, Hu J (2014) Stream bench: Towards benchmarking modern distributed stream computing frameworks. In: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, IEEE, pp 69–78

Mendes M, Bizarro P, Marques P (2013) Towards a standard event processing benchmark. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, ACM, pp 307–310

Mendes MR, Bizarro P, Marques P (2009) A performance study of event processing systems. In: Technology Conference on Performance Evaluation and Benchmarking, Springer, pp 221–236

Mohamed S, Forshaw M, Thomas N, Dinn A (2017) Performance and dependability evaluation of distributed event-based systems: A dynamic code-injection approach. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ACM, pp 349–352

Motwani R, Widom J, Arasu A, Babcock B, Babu S, Datar M, Manku G, Olston C, Rosenstein J, Varma R (2003) Query processing, resource management, and approximation in a data stream management system. CIDR

Nazhandali L, Minuth M, Austin T (2005) Sensebench: Toward an accurate evaluation of sensor network processors. In: Workload Characterization Symposium, 2005. Proceedings of the IEEE International, IEEE, pp 197–203

Perera S, Perera A, Hakimzadeh K (2016) Reproducible experiments for comparing apache flink and apache spark on public clouds. arXiv preprint arXiv:161004493

Qian S, Wu G, Huang J, Das T (2016) Benchmarking modern distributed streaming platforms. In: Industrial Technology (ICIT), 2016 IEEE International Conference on, IEEE, pp 592–598

Samosir J, Indrawan-Santiago M, Haghighi PD (2016) An evaluation of data stream processing systems for data driven applications. Procedia Computer Science 80:439–449

Schmidt AR, Waas F, Kersten ML, Florescu D, Manolescu I, Carey MJ, Busse R (2001) The xml benchmark project

Schroeder B, Wierman A, Harchol-Balter M (2006) Open versus closed: A cautionary tale. In: Nsdi, vol 6, pp 18–18

Shukla A, Simmhan Y (2016) Benchmarking distributed stream processing platforms for iot applications. In: Technology Conference on Performance Evaluation and Benchmarking, Springer, pp 90–106

Shukla A, Chaturvedi S, Simmhan Y (2017) Riotbench: A real-time iot benchmark for distributed stream processing platforms. arXiv preprint arXiv:170108530

Teng M, Sun Q, Deng B, Sun L, Qin X (2017) A tool of benchmarking realtime analysis for massive behavior data. In: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data, Springer, pp 345–348

Trivedi A, Stuedi P, Pfefferle J, Stoica R, Metzler B, Koltsidas I, Ioannou N (2016) On the [ir] relevance of network performance for data processing. Network 40:60

Tucker P, Tufte K, Papadimos V, Maier D (2008) Nexmark–a benchmark for queries over data streams (draft). Tech. rep., Technical report, OGI School of Science & Engineering at OHSU, Septembers

Wolf T, Franklin M (2000) Commbench-a telecommunications benchmark for network processors. In: Performance Analysis of Systems and Software, 2000. ISPASS. 2000 IEEE International Symposium on, IEEE, pp 154–162

Zhang S, He B, Dahlmeier D, Zhou AC, Heinze T (2017) Revisiting the design of data

stream processing systems on multi-core processors. In: Data Engineering (ICDE), 2017 IEEE 33rd International Conference on, IEEE, pp 659–670

Zhang Y, Duc PM, Corcho O, Calbimonte JP (2012) Srbench: a streaming rdf/sparql benchmark. In: International Semantic Web Conference, Springer, pp 641–657