Rhino: Efficient Management of Very Large Distributed State for Stream Processing Engines [Abstract]

Bonaventura Del Monte[1], Steffen Zeuch[1,2], Tilmann Rabl[3] and Volker Markl[1,2]

[1]Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany
[2]DFKI GmbH, Alt-Moabit 91c, 10559 Berlin, Germany
[3]Hasso Plattner Institute, Campus II, Building F, 1st Floor, August-Bebel-Str. 88, 14482 Potsdam, Germany

Abstract

Scale-out stream processing engines (SPEs) are powering large big data applications on high velocity data streams. Industrial setups require SPEs to sustain outages, varying data rates, and low-latency processing. Thus, SPEs need to transparently reconfigure stateful queries during runtime. However, state-of-the-art SPEs are not ready yet to handle on-the-fly reconfigurations of queries with terabytes of state due to three problems. First, *network overhead*: a reconfiguration involves state migration between workers over a network, which results in more resource utilization and latency proportional to state size. Second, *consistency*: a reconfiguration has to guarantee exactly-once processing semantics through consistent state management and record routing. Third, *processing overhead*: a reconfiguration must have minimal impact on performance of query processing.

Today, several industrial and research solutions provide state migration. However, these solutions restrict their scope to small state sizes or offer limited on-the-fly reconfigurations. Apache Flink [1], Apache Spark [2], and Apache Samza [3], enable consistency but at the expense of performance and network throughput. They support large, consistent operator state but they restart the running query upon its reconfiguration. Research prototypes, e.g., Chi [4], Megaphone [5], and SEEP [6] address consistency and performance but not network overhead. They enable fine-grained reconfiguration but support smaller state sizes (i.e., tens of gigabytes).

To bridge the gap between stateful stream processing and operational efficiency via on-the-fly query reconfigurations and state migration, we propose *Rhino*. Rhino is a library for efficient management of very large distributed state compatible with SPEs based on the streaming dataflow paradigm [7]. Rhino enables on-the-fly reconfiguration of a running query to provide resource elasticity, fault tolerance, and runtime query optimizations, such as load balancing, in the presence of very large distributed state (i.e., up to TBs). To this end, Rhino applies a state-centric, proactive replication protocol to asynchronously replicate the state of a running operator on a set of SPE workers through incremental checkpoints. Furthermore, Rhino applies a handover protocol that smoothly migrates processing and state of a running operator among workers. This does not impact query execution and guarantees exactly-once processing.

Overall, our evaluation shows that Rhino scales with state sizes of up to TBs, reconfigures a running query fifteen times faster than baseline solutions, and reduces latency by three orders of magnitude upon a reconfiguration. Rhino was originally published as full research paper at the 2020 ACM SIGMOD conference [8]. Currently, we use Rhino in our NebulaStream data management platform for unified Cloud and Internet-of-Things environments [9].

Keywords

Distributed Stream Processing, Query Optimization, Fault Tolerance

Acknowledgments

This work was funded by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and 01IS18037A).

References

[1] A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. Sax, S. Schelter, M. Höger, K. Tzoumas, D. Warneke, The stratosphere platform for big data analytics, The VLDB Journal (2014).

[2] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, I. Stoica, Discretized streams: Fault-tolerant streaming computation at scale, in: ACM SOSP, 2013.

[3] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, R. H. Campbell, Samza: Stateful scalable stream processing at linkedin, PVLDB (2017).

[4] L. Mai, K. Zeng, R. Potharaju, L. Xu, S. Venkataraman, P. Costa, T. Kim, S. Muthukrishnan, V. Kuppa, S. Dhulipalla, S. Rao, Chi: A scalable and programmable control plane for distributed stream processing systems, VLDB (2018).

[5] M. Hoffmann, A. Lattuada, F. McSherry, V. Kalavri, T. Roscoe, Megaphone: Latency-conscious state migration for distributed streaming dataflows, VLDB (2019).

[6] R. Castro Fernandez, M. Migliavacca, E. Kalyvianaki, P. Pietzuch, Integrating scale out and fault tolerance in stream processing using operator state management, in: ACM SIGMOD, 2013.

[7] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, et al., The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing, PVLDB 8 (2015) 1792–1803.

[8] B. Del Monte, S. Zeuch, T. Rabl, V. Markl, Rhino: Efficient management of very large distributed state for stream processing engines, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2471–2486.

[9] S. Zeuch, A. Chaudhary, B. Monte, H. Gavriilidis, D. Giouroukis, P. Grulich, S. Breß, J. Traub, V. Markl, The nebulastream platform: Data and application management for the internet of things, in: Conference on Innovative Data Systems Research (CIDR), 2020.

LWDA'22: Lernen, Wissen, Daten, Analysen. October 05–07, 2022, Hildesheim, Germany

delmonte@tu-berlin.de (B. Del Monte); steffen.zeuch@dfki.de (S. Zeuch); tilmann.rabl@hpi.de (T. Rabl); volker.markl@tu-berlin.de (V. Markl)