Optimizing distributed data stream processing by tracing

Zoltán Zvara, Péter G.N. Szabó, Barnabás Balázs, András Benczúr

PII: S0167-739X(17)32514-1
DOI: https://doi.org/10.1016/j.future.2018.06.047
Reference: FUTURE 4309

To appear in: *Future Generation Computer Systems*

Received date : 3 November 2017
Revised date : 21 June 2018
Accepted date : 25 June 2018

Please cite this article as: Z. Zvara, P.G.N. Szabó, B. Balázs, A. Benczúr, Optimizing distributed data stream processing by tracing, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.06.047

Optimizing Distributed Data Stream Processing by Tracing[☆]

Zoltán Zvara[a,*], Péter G.N. Szabó[a], Barnabás Balázs[a], András Benczúr[a]

[a]*Hungarian Academy of Sciences*
Institute for Computer Science and Control (MTA SZTAKI)
13-17 Kende u., H-1111 Budapest, Hungary

Abstract

Heterogeneous mobile, sensor, IoT, smart environment, and social networking applications have recently started to produce unbounded, fast, and massive-scale streams of data that have to be processed "on the fly". Systems that process such data have to be enhanced with detection for operational exceptions and with triggers for both automated and manual operator actions. In this paper, we illustrate how tracing in distributed data processing systems can be applied to detecting changes in data and operational environment to maintain the efficiency of heterogeneous data stream processing systems under potentially changing data quality and distribution. By the tracing of individual input records, we can (1) identify outliers in a web crawling and document processing system and use the insights to define URL filtering rules; (2) identify heavy keys, such as NULL, that should be filtered before processing; (3) give hints to improve the key-based partitioning mechanisms; and (4) measure the limits of overpartitioning if heavy thread-unsafe libraries are imported.

By using Apache Spark as illustration, we show how various data stream processing efficiency issues can be mitigated or optimized by our distributed tracing engine. We describe and qualitatively compare two different designs, one based on reporting to a distributed database and another based on trace piggybacking. Our prototype implementation consists of wrappers suitable for JVM environments in general, with minimal impact on the source code of the core system. Our tracing framework is the first to solve tracing in multiple systems across boundaries and to provide detailed performance measurements suitable for automated optimization, not just debugging.

Keywords: Distributed Data Processing, Data Stream Processing, Distributed Tracing, Data Provenance, Apache Spark

1. Introduction

Big data analytics is difficult to realize using today's state-of-the-art technologies, given the flood of data generated from various sources that has to be processed "on the fly". Fast data can flood from network measurements, call records, web page visits, sensor readings, and so on [1]. We can also observe a shift from traditional data stream processing performed in a central data center to a geo-distributed processing environment as represented by Fog computing and multi-clouds [2]. Traditional data processing assumes that data is available for multiple access, even if in some cases it resides on disk and can only be processed in larger chunks. In this case, we say that the data is *at rest*, and we can perform *batch processing*. Fast data, or *data in motion*, is closely connected to and in certain cases used as a synonym of the *data stream* computational model [3]. In this model, data arrives continuously in a potentially infinite stream that has to be processed by a resource-constrained system.

High-performance data streaming solutions need to be equipped with smart logics to adapt the framework and the application to rapidly changing execution conditions and workloads [4]. Towards this end, in [5] we designed a tracing framework that can monitor cloud and fog systems across subsystem boundaries, with particular emphasis on some key properties of Fog computing systems, including real-time data streaming, low latency, location awareness, and heterogeneous software architectures [2]. In this paper, we give an overview of our system and demonstrate its applicability for monitoring and optimizing on the fly.

Monitoring cloud computing platforms is a complex task of high practical relevance [6]. In a cloud environment, one of the main challenges lies in understanding and troubleshooting distributed data processing systems (*DDPS*), and detecting causes of performance degradation. Compared to batch jobs, distributed data streaming applications are even harder to optimize by performance measurements for a variety of reasons: The distribution of the data is unknown, the velocity may rapidly change, new types of outliers may appear in the data, and service level agreements require adaptive intervention to the execution graph of a running job. Figure 1 shows an example heterogeneous Fog compute topology where our system can trace records from data source through an edge, some cloud, and again an edge application.

[☆]Research was supported from the EIT Digital Activity 17186: HopsWorks and the "Momentum - Big Data" grant of the Hungarian Academy of Sciences.
[*]Corresponding author
Email addresses: `zoltan.zvara@sztaki.mta.hu` (Zoltán Zvara), `peter.szabo@sztaki.mta.hu` (Péter G.N. Szabó), `balazs.barnabas@sztaki.mta.hu` (Barnabás Balázs), `andras.benczur@sztaki.mta.hu` (András Benczúr)