PAM: Fast reactive reconfiguration for stateful stream processing

Pritish Mishra
University of Toronto
pritish@cs.toronto.edu

Oana Balmau
McGill University
oana.balmau@cs.mcgill.ca

Eyal de Lara
University of Toronto
delara@cs.toronto.edu

ABSTRACT

Stream processing frameworks such as Flink and Storm are powerful tools to manage applications on the cloud that require high throughput and real-time data processing. However, new applications like traffic monitoring, autonomous driving, and augmented/virtual reality-based interactive environments rely on distributed data generated by remote sensors or cameras. Processing this data in a single cloud data center may be infeasible as it requires sending the data from the remote sources to the cloud, running up against cost, bandwidth, and latency limits.

To address these requirements, such modern streaming applications are being deployed on a combination of geo-distributed cloud and edge datacenters. In this cloud-edge infrastructure, while nodes closer to the edge reduce the data transferred to the cloud and result in lower latency, the cost of running applications on edge resources is very high. Hence, to balance the objectives of cost and performance, the application operators are initially deployed on the cloud and only when there is sufficient increase in user demand, the application is reconfigured to deploy specific operators on the edge nodes to bring processing closer to the data producers. Such a reconfiguration of the application is usually predicated by a change in user traffic and the amount of data generated. Another cause for application reconfiguration could be the mobility of data sources. When a data source moves from one edge to another, the operators processing the data for this source must also be migrated accordingly.

Reconfiguration of application operators is a challenging problem, especially for stateful operators. Stateful operators usually store information that is used for future computation, e.g. for a traffic monitoring application, state stored for a car could be its position and current average speed. Reconfiguration of such operators requires moving the state of the operator along with this processing and needs to address four key challenges: (i) maintaining guarantees of in-order and exactly-once processing of tuples produced by data sources; (ii) preserving state correctness, since tuples that can modify the state are being processed concurrently with the state migration; (iii) avoiding global coordination, which is expensive in geographically distributed deployments due to the high latency of network links; and (iv) avoiding overheads while the application is not being reconfigured.

Since scenarios demanding reconfiguration cannot always be predicted well in advance, two key goals of a stream processing framework are - a) to minimize the time taken to perform the application reconfiguration and b) to minimize its impact on application performance. Existing reconfiguration approaches incur application stoppage ranging from a few seconds to hundreds of milliseconds which is not tolerable for edge applications requiring real-time processing.

We present PAM - a Push-based Adaptive Migration technique to perform reconfiguration of stateful applications experiencing sudden spikes in user demand. Two key ideas utilized in our migration protocol are - a) the state of one data source is migrated at a time and once this migration is completed, processing of tuples belonging to this data source is resumed on the destination instance, and b) it looks ahead the backlog of tuples awaiting processing and leverages patterns in this backlog to the schedule the order of transfer of state for data sources. This adaptive scheduled transfer for data sources using patterns like tuple arrival order and skewness in the traffic of some data sources reduces the amount of time that the destination instance has to wait to receive the state stored for each data source and resume the processing. Specifically, by adaptively moving the state, our algorithm ensures that the transfer cost of the moving state of a data source is interleaved by the processing cost of tuples for data sources whose state has already been transferred.

The benefit of using a push-based mechanism where the source instance decides the transfer order of the data source state eliminates the need for message passing between the source and destination instances to co-ordinate the transfer.

Our theoretical evaluation shows that PAM can achieve shorter reconfiguration duration than existing state reconfiguration protocols. Our empirical evaluation on a hierarchical network composed of geographically distributed Amazon datacenters shows that PAM can achieve 45-60% reduction in the time taken to stabilize the application performance and minimizes the application stoppage to 10–15 milliseconds. We also show that this performance remains stable even with the increase of state stored per data source, increase in the number of data sources, and increase in the number of edges. Finally, we show that this approach supports the mobility of data sources across edges and migrates state of the moving data source seamlessly across the edges.