Open Execution Engines of Stream Analysis Operations

Qiming Chen and Meichun Hsu

HP Labs
Palo Alto, California, USA
Hewlett Packard Co.
(qiming.chen,meichun.hsu)@hp.com

Abstract. In this paper we describe our massively parallel and elastic stream analysis platform; it is capable of supporting the graph-structured dataflow process with each logical operator executed by multiple physical instances running in parallel over distributed server nodes. We propose the canonical dataflow operator framework to provide automated and systematic support for executing, parallelizing and granulizing the continuous operations.

We focus on the following issues: first, how to categorize the meta-properties of stream operators such as the I/O, blocking, data grouping characteristics, for providing unified and automated system support; next, how to elastically and correctly parallelize a stateful operator that is history-sensitive, relying on the prior state and data processing results; and further, how to analyze unbounded stream granularly to ensure sound semantics (e.g. aggregation). These issues are not properly abstracted and systematically addressed in the current generation of stream processing systems, but left to user programs which can result in fragile code, disappointing performance and incorrect results.

We tackle these issues by introducing the open-executors. An open executor supports the streaming operations with specific characteristics and running pattern, but is open for the application logic to be plugged-in. We illustrate the power of this approach by showing the system support in parallelizing and granulizing dataflow operations safely and correctly. The proposed canonical operation framework can be generalized to allow us to standardize various operational patterns of stream operators, and have these patterns supported systematically and automatically. We have built this platform; our experience reveals its value in real-time, continuous, elastic data-parallel and topological stream analysis process.

1 Introduction

Real-time stream analytics has increasingly gained popularity since enterprises need to capture and update business information just-in-time, analyze continuously generated “moving data” from sensors, mobile devices, social media of all types, and gain live business intelligence.

We have built a stream analytics platform with code name Fontainebleau for dealing with continuous, real-time data-flow with graph-structured topology. This platform is massively parallel, distributed and elastic with each logical operator executed by multiple physical instances running in parallel over distributed server nodes. The stream
analysis operators are defined by users flexibly. From stream abstraction point of view, our stream analytics cluster is positioned in the same space of System S(IBM), Dryad(MS), Storm(Tweeter), etc. However, this work aims to advance the state of art by providing canonical execution support for stream analysis operators.

1.1 The Challenges

A stream analytics process with continuous, graph-structured dataflow is composed by multiple operators and the pipes connecting these operators. The operators for stream analysis have certain meta-properties representing their I/O characteristics, blocking characteristics, data grouping characteristics, etc, which can be categorized for introducing unified system support. Categorizing stream operators and their running patterns to provide automatic support accordingly, can ensure the operators to be executed optimally and consistently, as well as ease user’s effort for dealing with these properties manually which is often tedious and risky. Unfortunately, this issue has been missed by the existing stream processing systems.

Next, to scale out, the data-parallel execution of operators must be taken into account, where how to ensure the correctness of data-parallelism is the key issue, and requires the appropriate system protocol to guarantee; particularly in parallelizing stateful stream operators where the stream data partitioning and data buffering must be consistent.

Further, stream processing is often made in granule. For example, to provide sound aggregation semantics (e.g. sum), the infinite input data stream must be processed chunk by chunk where each operator may punctuate data based on different chunking criteria such as in 1-minute or 1-hour time windows (certain constraints apply, e.g. the frame of a downstream operator must be the same as, or some integral number of, the frame of its upstream operator). Granulizing dataflow analytics represents another kind of common behavior of stream operators which also need to be supported systematically.

Current large-scale data processing tools, such as Map-Reduce, Dryad, Storm, etc, do not address these issues in a canonical way. As a result, the programmers have to deal with them on their own, which can lead to fragile code, disappointing performance and incorrect results.

1.2 The Proposed Solution

The operators on a parallel and distributed dataflow infrastructure are performed by both the infrastructure and the user programs, which we refer to as their *template behavior* and *dynamic behavior*. The template behavior of a stream operator depends on its meta-properties and its running pattern. For example, a map-reduce application is performed by the Hadoop infrastructure as well as the user-coded map function and reduce function. Our streaming platform is more flexible and elastic than Hadoop in handling dynamically parallelized operations in a general graph structured dataflow topology, and our focus is placed on supporting the template behavior, or operation patterns, automatically and systematically.

Unlike applying an operator to data, stream processing is characterized by the flowing of data through a *stationed* operator. We introduce the notion of *open-station* as the container of a stream operator. The stream operators with certain common