ID2221 - Reading Assignment 3

Authors:

Miquel Larsson Corominas,
Dmytro Siniukov

1 Summary

In this third and final reading assignment of the course, our task consists of reviewing two papers: *"Discretized Streams: Fault-Tolerant Streaming Computation at Scale"* and *"Apache Flink: Stream and Batch Processing in a Single Engine"*. From now on, these two papers will be reffed to in the present document as the "DS" and the "Flink" paper.

The DS paper focuses on introducing a new processing model: Discretized streams (D-Streams), which claims to overcome a lot of the different challenges that appear then attempting to process real time big data.

The other paper, however, focuses on a framework (Apache Flink) that expresses and executes data processing applications as pipelined fault-tolerant dataflows.

It is important to highlight the fact that the Flink paper is not written by the inventors or developers behind Apache Flink, but by a group of enthusiasts who want to contribute to the framework by providing and working on proper documentation for it.

2 Motivation

A big fraction of "big data" is data that is received in real time, and has an analytic value that is at peak when immediately received. This value decays over time, so for many applications, being able to continuously process this new data and obtain insights (or process it) as it arrives is critical. The DS paper gives us some examples of these real life applications that require real time processing: Social media networks that want to notice new conversation trends or spam attacks as soon as possible, operators trying to manage huge distributed clusters and detect any errors in the massive logs immediately, etc. In order to achieve this, scalable streaming low-latency computation models are required.

Distributing such a model brings several issues due to the own nature of processing clusters: faults and stragglers. Meanwhile a half a minute recovery time for a node error is acceptable for many batch distributed processing tasks, for streaming this might be unacceptable. The proposed D-Stream model tries to overcome these challenges and introduce a new system that scales well (in the order of hundreds of nodes), has a low latency and fast recovery (in the second scale) and minimizes the hardware cost (by avoiding the replication overhead).

Behind the creation of Apache Flink there was the following motivation. Nowadays, there are a lot of applications that operate on the data that is being produced continuously over time (logs, sensors, transaction, etc), which requires using another data processing mechanism rather than conventional batch data processing and stream processing apart.

3 Contributions

The DS paper mentions other streaming models and systems (Storm, TimeStream, MapReduce Online, etc.), arguing that their design based on a *continuous operator* model makes it difficult to manage faults, as the recovery is managed by *replication* or *upstream backup*, which are not very attractive when working on big clusters do to the need of steeply increasing the amount of hardware (and investment). Furthermore, neither of these approaches handles stragglers well (slow nodes in the system).

The proposed D-Streams achieve a fast

recovery and have a improved performance over other streaming processing frameworks by batching the real time data into small timesteps, which allows for parallel recovery through time and data partitions. Then, low latency and other improvements are obtained by implementing the model as am extension of Spark.

The main contribution of the Apache Flink developers is implementing a unified architecture of stream and batch data processing, including some optimisations for static data sets.

Apart from that, they have shown how streaming, batch, iterative, and interactive analytics can be represented as fault-tolerant streaming dataflows.

4 Solution

Regarding the DS paper, Discretized Streams (D-Streams) solves the mentioned streaming challenges by structuring the streams of data into *stateless, deterministic batch computations* in small timesteps. By doing this, every small timestep has a deterministic state given the input, which allows it to be independent from older data and bypasses the complexity of synchronization protocols, enabling a fast and powerful recovery mechanism, that perform substantially better than replication and upstream backup recoveries.

Deterministic batch computations make it possible to enable a parallel recovery mechanism that was not possible before, as when there is an error, each node in the cluster can recompute part of the lost nodes data.

Furthermore, low latency (the importance of which cannot be underestimated, as explained in the summary) is achieved thanks to using Resilient Distributed Datasets (RDDs), which are saved on memory and are fault tolerant, compared to other batch processing systems that depend on replication on hard disks, that are considerably slower.

The proposed D-Stream model is implemented into Spark Streaming, an extension of the well known distributed cluster computing framework Spark. Numerous optimizations and changes had to be implemented to the batch engine to support streaming, which are described in the paper.

Regarding Apache Flink, it treats data-stream processing as the common model for real-time analysis, continuous streams, and batch processing. Different types of computations (e.g. processing historical data, aggregating data periodically, or in real-time) start their processing at different points in the durable stream, and maintain different forms of state during execution.

Also, Flink can treat batch computations specially, by optimizing their execution using a query optimizer and by implementing blocking operators.

5 Opinion

The first paper, about Discretized Streams is extremely interesting, as the approach is quite novel compared to other distributed stream processing models that were available at the moment of publishing.

The improvements over other systems are substantial, and on a personal note we have to highlight the well written section where implementations in real applications is done, as this really makes it easy to understand what uses and capabilities the proposed model has in real industry problems.

Both papers are extremely related, and give us a nice overview of the available frameworks when it comes to processing real time data. Although the Flink paper is not written by the developers and creators, it is a recommended read.