Model driven scaling for distributed steam processing systems and micro-services

Authors

1. Thomas Cooper - t.cooper@newcastle.ac.uk (Newcastle University)
2. Paul Ezhilchelvan - paul.ezhilchelvan@newcastle.ac.uk (Newcastle University)

Abstract

Distributed stream processing systems (like Apache Storm, Heron and Flink) allow the processing of massive amounts of data with low latency and high throughput. Part of the power of these systems is their ability to scale the separate sections (operators) of a stream processing query to adapt to changes in incoming workload and maintain end-to-end latency or throughput requirements.

Whilst the mainstream stream processing systems provide the functionality to scale their query operators, none of them suggest to the operator how best to scale their queries for better performance. The operator is required to deploy the query, wait for the deployment to stabilise, assess the performance metrics from the query, alter the query's configuration and repeat this loop until the query can perform at the required level. This scaling decision loop is intensely time consuming and for complex queries, that have to deal with high incoming workloads, the process can take days to complete. The time and effort involved also means that changing the query configuration, once it is set for peak load, is discouraged and so leads to the over provisioning of resources.

There are many examples in the literature of attempts to automate the configuration of streaming queries (scheduling) and find an optimal configuration for a given set of constraints. Some of these have even been integrated into the streaming systems (see Dhalion for Apache Heron). However, all the systems proposed so far still require multiple scaling loops to be carried out in order to find the correct configuration. They remove the burden on the operator to make decisions but do not significantly reduce the time required.

Our research is focused on solving this problem. We have created a performance modelling system for distributed stream processing queries. This system is able to accept a configuration proposed by any scheduler available in the literature and use a small amount of historical data to predict how well that configuration will perform. If the prediction meets the requirements it can be deployed and if not then information can be returned to the scheduler and a new configuration tried. This removes the stabilisation and assessment sections of the scaling decision loop and allows a viable configuration to found more quickly.

A modelling system also allows the effect of different incoming workload levels can be assessed. If the traffic into a query is predicted to rise at some point in the future, modelling of potential configurations to maintain performance in the presence of that workload can be done ahead of time and the configuration deployed in time to meet the higher traffic level (pre-emptive scaling).

In this talk we will detail our modelling approach and discuss its application to Apache Storm. We will also discuss the result of a collaboration with Twitter to apply this work to their Heron system. Finally, we will highlight how this approach can be applied, at a higher level, to micro-service architectures to potentially allow faster and pre-emptive scaling of these systems.