Addressing transient workload spikes with Cloud bursting in Apache Flink

Introduction

Due to the push-based nature of streaming inputs from external data sources, stream processors have no control over the rate of incoming events. Satisfying quality of service (QoS) under workload variations has been a long-standing research challenge in stream processing systems. To avoid performance degradation when input rates exceed system capacity, the stream processor needs to take actions that will ensure sustaining the load. For example, systems like Flink employ flow control mechanisms, like *back-pressure*. In a network of consumers and producers such as a streaming execution graph with multiple operators, back-pressure propagates to upstream operators, eventually reaching the data stream sources.

Unfortunately, back-pressure mechanisms have two major disadvantages. First, back-pressure has the effect that all operators slow down to match the processing speed of the slowest consumer. Second, to ensure no data loss, a persistent input message queue, such as Apache Kafka, and adequate storage space are required. In this project, you will investigate an alternative approach that relies on *serverless computing*.

Background

Serverless computing is a cloud computing model where the cloud provider is responsible for the server management and the user can run their code without provisioning, scaling, and maintaining servers. Instead, the user is charged based on the number of *requests* and the duration of the execution. The most common use case for serverless computing is running event-driven, stateless, and short-lived functions. These functions are often called "serverless functions" or "lambda functions."

“Cloud bursting” is a technique used to handle high traffic or computational load of an application. It allows an organization to run their application on-premises, and when the load exceeds the capacity of the on-premises infrastructure, additional resources are automatically acquired from a cloud provider. When the load on the on-premises infrastructure exceeds a predefined threshold, the application automatically starts using the “burst” capacity. The cloud resources are used until the load drops below the threshold, at which point the application stops using the cloud resources and goes back to using the base capacity.

Project goal

The goal of this project is to design and implement an adaptive Flink application that leverages the "cloud bursting" technique as an alternative to back-pressure. Your application should automatically detect workload variations and offload excess load to AWS lambda functions. You will first develop a controller that will continuously collect execution metrics and monitor the dataflow performance and input rate. If excess workload is detected, the controller will have to decide whether cloud bursting is necessary. In that case, it will be responsible for spawning lambda functions, collecting their results, and merging them with the results produced by the dataflow running on your local machine.

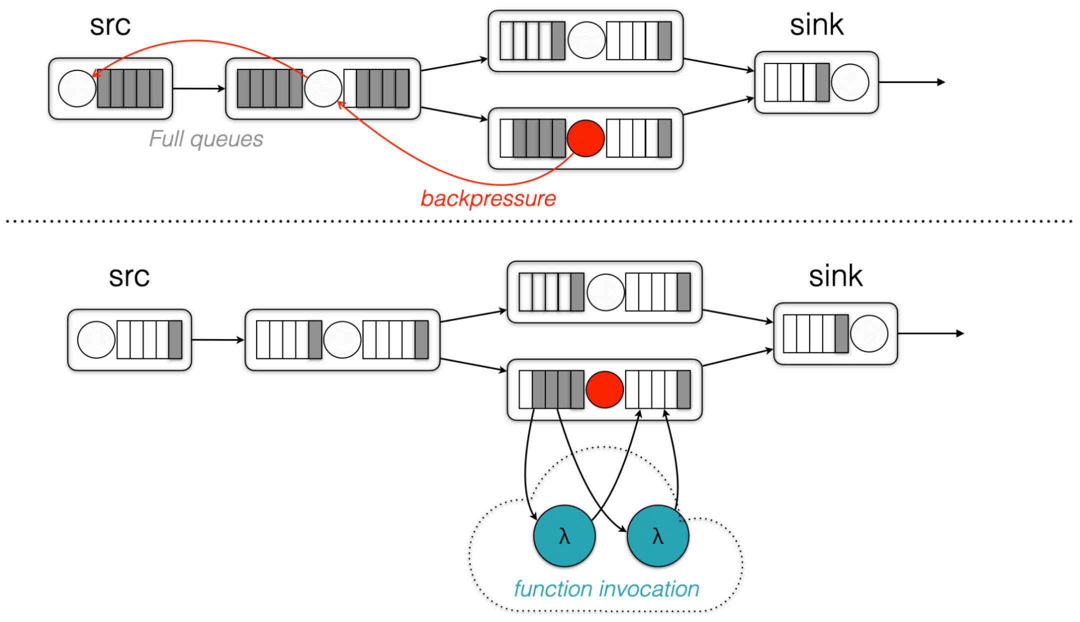

Fig 1: Backpressure propagates from the bottleneck operator to the sources, slowing down the entire dataflow. Cloud bursting can offload the excess computation to the cloud.

Required skill set

- Experience with Java programming.
- Experience with AWS is a plus.

Where to start

- Read about serverless computing and lambdas https://aws.amazon.com/lambda/ and figure out how to spawn functions.
- Write a simple Flink benchmark that allows you to change the input rate.