#7: Smart Operator Placement for Stream Processing at the Edge

Edge computing applications are growing rapidly in a wide range of domains, including smart cities, healthcare, and manufacturing. One of the common edge computing devices is Raspberry Pi (RPi). Raspberry Pi is a small single-board computer that can connect with sensors (e.g. Temperature, Distance, etc.). Consider a streaming application that ingests data streams from RPis at different locations to the cloud via Wide Area Network (WAN). The application collects sensor data and needs to perform continuous aggregations and queries. The WAN has limited bandwidth and high latency, so it might be challenging to ship data from the sensors to the cloud in a timely manner.

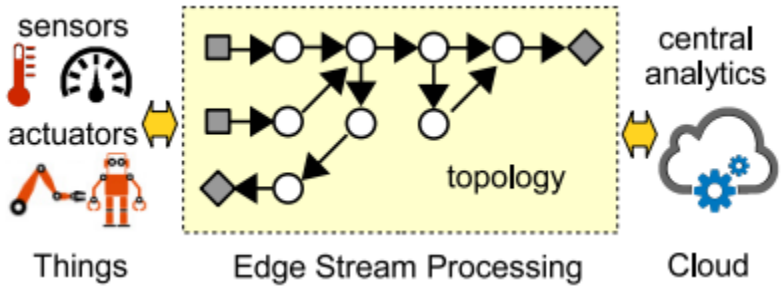

Figure 1: The Edge connects Sensors to the Cloud, and can perform local data stream processing [1]

The application would benefit if data streams could be pre-processed (e.g. filtered) at the Edge, i.e. near the data source, to reduce data traffic flowing from the sensors to the cloud. However, existing stream processing systems like Flink are mainly designed for datacenter servers with homogeneous hardware resources and cannot automatically offload tasks to the edge.

Project goal

In this project, you will develop a data stream processing system whose computations can span edge and cloud resources. You will be given a prototype Flink scheduler that can offload query operators to the edge (RPi) and a simple query implementation as a proof of concept. You will also be provided with a computing cluster consisting of 1 server and several RPis. Your task is to extend the prototype into a full-fledged edge stream processing system that can automatically decide which operators should be executed on edge resources and which ones in the cloud. Your system should be able to adapt to varying workloads and bandwidth conditions.

Tasks

1. Get started by trying out the prototype and running an experiment using the RPi cluster.
2. Read the code and become familiar with the current implementation and performance model.
3. Generalize the prototype so that it can execute any Flink query. To do that, you will need to develop a query parser to identify different types of operators, decide which operators may need to be moved to edge resources, and transform the dataflow graph accordingly.
4. Extend and improve the prototype cost model to consider different types of operators and their characteristics.

Required skill set

- Experience with Java, Python.
- Experience with RPis will be a plus.

Resources

[1] Xinwei Fu, Talha Ghaffar, James C. Davis, and Dongyoon Lee. 2019. Edgewise: a better stream processing engine for the edge. In Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '19). USENIX Association, USA, 929–945.