Automated Code Distribution for Distributed Stream Processing

Konstantinos Kallas, Filip Niksic, Caleb Stanford, Rajeev Alur

Motivation

Internet of Things

The amount of data produced by IoT devices in coming years is expected to grow orders of magnitude faster than the capacity of the cloud to process it. This has motivated the edge computing paradigm, where applications process data streams at the source, as they arrive, with minimal communication over the network.

We view this emerging paradigm from the perspective of formal specification and programming languages. Given a description of the topology of nodes and a formal specification of the computation to be performed, how can we automatically distribute correct, efficient code across the edge nodes?

Flumina: System Overview

Flumina features:
- **Correct**: The distributed code implements the computation specification, regardless of the order in which events occur.
- **Efficient**: The optimizer tries to minimize a cost metric. Possible cost metrics include communication costs, i.e. network load (our current optimizer), response time, i.e. latency; and total time, i.e. throughput.

Related Work

- **Edge computing and IoT applications**: most code is low-level and task-specific.
- **Distributed stream processing**: cloud-computing alternative, based on data-flow programming:

  - **Apache Storm**: A scalable and fault-tolerant distributed real-time computation system
  - **Spark**: A high-level system for distributed computing
  - **Apache Flink**: A streaming system for fault-tolerant processing

Computation Specification

Implementation-independent programming model with three components:
1. **Update function**: describe how to sequentially update the state
2. **Logical dependencies** on the input events
3. **Parallelization primitives**: a fork function and a join function which describe how to fork and join state for parallelization

State = (int, int). // Current counter; max so far
Event = i, d, f

State update(Event e. State (x, y)) =
match e with
i ⇒ return (x+1, max(y,x+1))
d ⇒ return (x-1, y)
f ⇒ update y; return (x, y).

State init() = (0, 0).

Cost of a distribution strategy: number of messages over network per second

Problem: Given a specification and a target topology with rates, what distribution strategy has the best cost?

- Maliciously written specifications: NP-complete
- Specifications where no possible forks/joins are missing?

Experiments

Sequential: Sequential implementation at a single node
Greedy hybrid: Cloud-computing simulation; parallelize at one high-powered data center node
Greedy: Optimize for network load

Research Agenda and Future Work

Compile high-level queries to Flumina specifications (SQL; regular-expression based)
Leverage other representations: e.g. dataflow-based [PLDZ98], automata-based [ICA117, PPG119]

Acknowledgements

This research was supported in part by NSF Award CCF 1763514.