Chi: A Scalable and Programmable Control Plane for Distributed Stream Processing Systems

Samhith Venkatesh
11/06/2018
Agenda

- Introduction
- Challenges
- Motivation
- Problem
- Background
- Design
- Implementation
- Evaluation
Introduction
Characteristics

Spatial Variability

Temporal Variability
Challenges

- Different Service Level Objectives
- Different expectations
- Usability vs Flexibility
Problem

Meet various objectives

1. Dynamic Scaling
2. Auto – Tuning
3. Data Skew Management

Heron and Flink lack flexibility
How to solve?

1. Efficient and extensible feedback-loop controls
2. Easy control interface
3. Minimal impact on the process
**Background**

**Control plane:** The control plane is the part of a network that carries signalling traffic and is responsible for routing. Functions of the control plane include system configuration and management.

**Data plane:** The data plane is the part of a network that carries user traffic. Data plane traffic travels through routers, rather than to or from them.
Streaming solutions: Naiad, StreamScope and Apache Flink

Dataflow Computation Model:

A dataflow program is a graph, where nodes represent operations and edges represent data paths.

Each node in the graph is represented by triples $(s_v, f_v, p_v)$

$s_v$ : states of the vertex
$f_v$ : defines the function which captures computation
$p_v$ : properties associated with the vertex
Design

- Installable controller and operator API
- Define new custom control operations
- Minimum effort
Embedding the control plane into the data plane

- Uses existing efficient data plane infrastructure
- No need of global synchronization
- Facilitate development of various asynchronous control operations
Overview

Control Operation: We can consider this as one feedback cycle comprising of a dataflow controller and the dataflow topology

Stages involved

- Control decision and instantiation
- Propagation of control messages along with data
- Control message reaches back to controller for post processing
Example: Word Count

- Two map operators \{M1, M2\}
- Two reduce operators \{R1, R2\}
- R1 maintains the counts for all words starting with [‘a’-‘l’], and R2 maintains those for [‘m’-‘z’].
- Controller monitors the memory usage

What happens when we have to scale the service?
Control Decision and Instantiation

- Controller detects and makes reconfiguration decision
- Start new reducer R3
  - R1 - [‘a’–‘h’]
  - R2 - [‘i’–‘p’]
  - R3 - [‘q’–‘z’]
- Broadcast control message to all source nodes
Control message propagation

- M1 and M2 receive and they block input channel and update their routing table.
- R1 and R2 receive and splits data
  - R1 - ['a’-‘h’] and ['i’-‘l’]
  - R2 - ['m’-‘p’] and ['q’-‘z’]
- Passes the information along with the control message
  - R1 - ['i’-‘l’]
  - R2 - ['m’-‘p’]
Control message lifecycle
Graph Transition

Introduce a meta topology $G'$, to complete the transformation asynchronously.

State Invariance: No change in node’s state, hence we collapse and merge.

Acyclic Invariance: Aggressive merge old and new topology.

- Check for loops before and after.
Operating at scale

- Multiple Controllers - concurrently run on multiple controllers at various stages. Also facilitate global controller
- Aggregation (Spanning trees) to avoid bottlenecks at source and sinks
- To deal with deadlocks we have separate queues
- Fault tolerance
  - Retransmission until acknowledgement
  - Timeout and restart mechanism in-case of network failure
  - Checkpoint and replay mechanism for operator and controller failures
Implementation
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Synchronous Global Control Models</th>
<th>Asynchronous Local Control Models</th>
<th>Chi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>Barrier</td>
<td>None</td>
<td>Barrier / None</td>
</tr>
<tr>
<td>Semantic</td>
<td>Simple</td>
<td>Hard</td>
<td>Simple</td>
</tr>
<tr>
<td>Latency</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Overhead</td>
<td>High</td>
<td>Implementation – dependent</td>
<td>Low</td>
</tr>
<tr>
<td>Scalability</td>
<td>Implementation – dependent</td>
<td>Implementation – dependent</td>
<td>High</td>
</tr>
</tbody>
</table>
Thank You