Hello!

CS 744: FLINK

Shivaram Venkataraman
Fall 2022
In Progress
- Course Project Proposal feedback (~ end of this week)
- Midterm, Assignment 2 grading

Resources for Course Projects
- Cloudlab reservations (Check Piazza)
- GCP credits (Email Roger and me)

Google Cloud $50
Scalable Storage Systems

Datacenter Architecture

Applications

Machine Learning | SQL | Streaming | Graph

Computational Engines

Scalable Storage Systems

Resource Management

Applications

API requirements
features
semantics

Datacenter Architecture
Input: unbounded stream of data

Output: 

Sales Dashboard

Total Sales: $3,256.8M
Number of Deals: 17,164
Avg Deal Size: $189,545
Rev. per Salesperson: $20.5M

Revenue Over Time:
- 2013: 246.6M
- 2014: 555.2M
- 2015: 846.7M
- 2016: 1,480.0M

Sales Team Performance:

Sales Team: Enterprise
Salesperson: Susan Olson, Steve Watkins, Raymond Hawk, Robert Hudson, Preston Ross, Sophia Willis, Dan Rivera, Sarah Stephens, Ross Spencer, Ellie Price

Revenue by Quarter:
- Q1: $12M, $46M, $174M, $387M
- Q2: $338M, $1,095M, $3,38P, $4,38P

Week of September 4, 2016
Revenue: $14.6M
Running Sum of Revenue: $798.4M
Streaming Computation

- Event time
- Hashmap grouping events by action
  - Counting how many window of 1 hr
- Window based on hour and then count how many window in each
- Output Sink

1. Reducers don't wait for maps to end
2. Tasks are long running
   - Allocate resources for all tasks

Mappers
- Split by (action, hour)

Reducers
- Count by (action, hour)
  - Extracting the hour

Examples:
- (phone1, 2:01, open)
- (phone1, 2:03, close)
- (phone2, 2:05, open)
- (phone4, 2:03, close)

Hashmap:
- open 1:00 5
- open 2:00 1
- close 1:00 5
- close 2:00 2
FLINK: COMPUTATION MODEL

Query → Streaming Word Count

**Manager/Driver**

**Task/Computation**

- Control Message
- Network Transfer

**Long-lived operators**

**Mutable State**

- Operators implement Map/Reduce/Windows
- Operators internally maintain state

**Google MillWheel**

**f^∞**

**Naiad**

**Streaming DBs:** Borealis, Flux etc
INTERMEDIATE DATA STREAMS

Dataflow graph

Stateful Operator

Transient Intermediate Data Stream (pipelined data exchange)

Materialized intermediate Data Stream (blocking data exchange)

Control Event

Data Record

Operator State

Operators can be partitioned

Streams can be 1-1 or 1-many

Pipelined stream

Comp. & Comm. pipelined

Store intermediate Tuples

BackPressure

Increase num partition / machines (slowest operators)
STATEFUL OPERATORS

Examples?
- All windowing operators
- Aggregation / maintaining summaries

Challenge
- How to ensure fault tolerance?
- Explicitly register local variables
- StateBackends that are automatically saved/recovered

Stateless
- $\text{map}(\cdot) : \text{input} \rightarrow \text{output}$
- $\text{filter}(\cdot) : \text{input}, \text{filter criteria}$

-> need to restore state when $\text{op}$ fails.
Fault Tolerance: Checkpointing

- Cannot replay all the data from beginning
- Apache Kafka

Fault Tolerance:
- "exactly once semantics"
- at least once
- at most once

Input is "replayable"

Data Streams: Control messages
Asynchronous Barrier Snapshotting

In-order delivery

control message

2 inputs. Needs to wait for both

Reset all operators to snapshot!

Only saw 1 barrier so far

No tuple after barrier should be in snapshot

- Preshot records src-1
- Preshot records src-2
- Postshot records
- Operator snapshot
- Snapshot barrier
- Blocked channel
WATERMARKS, WINDOWS

Implements similar model as Dataflow

“Watermarks originate at the sources of a topology”
Propagate through the other operators of dataflow

Windows based on event-time, processing time, ingest time(?)

stream
  .window(SlidingTimeWindows.of(
      Time.of(6, SECONDS), Time.of(2, SECONDS))
  .trigger(EventTimeTrigger.create())
COMBINING BATCH, STREAMING

Blocked DataStreams → Intermediate data in MR

Turn off periodic snapshots → Restart job

Blocking operators (e.g., sort) → New operators batch specific
Summary

Stream processing → Increasingly important workload trend

Flink: Distributed streaming dataflow to run streaming, batch, iterative

Distributed streaming dataflow
- Stateful operators
- Checkpointing based FT
DISCUSSION
https://forms.gle/idnKJWoBFHpRNC2j7
Data Streams
- Small buffer timeout
  - Low throughput ❌
  - Low 99th percentile latency ✔
- Very large timeout
  - Saturate at some throughput
  - Latency might keep going up?

![Graph showing latency and throughput against buffer timeout (milliseconds)](image)

- msg_to_ingest
- window
- reflected
Consider you are implementing a micro-batch streaming API on top of Apache Spark. What are some of the bottlenecks/challenges you might have in building such a system?

- Batch size \( bs = 1s \)
- Tasks are stateless, windowing needs state!
- Caching is less useful? Data changes every time

\[
\text{batch size is small} \implies \text{lots of tasks, high overhead for launching tasks}
\]
SUMMARY

Next class: Spark Streaming