Hello!

CS 744: FLINK

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

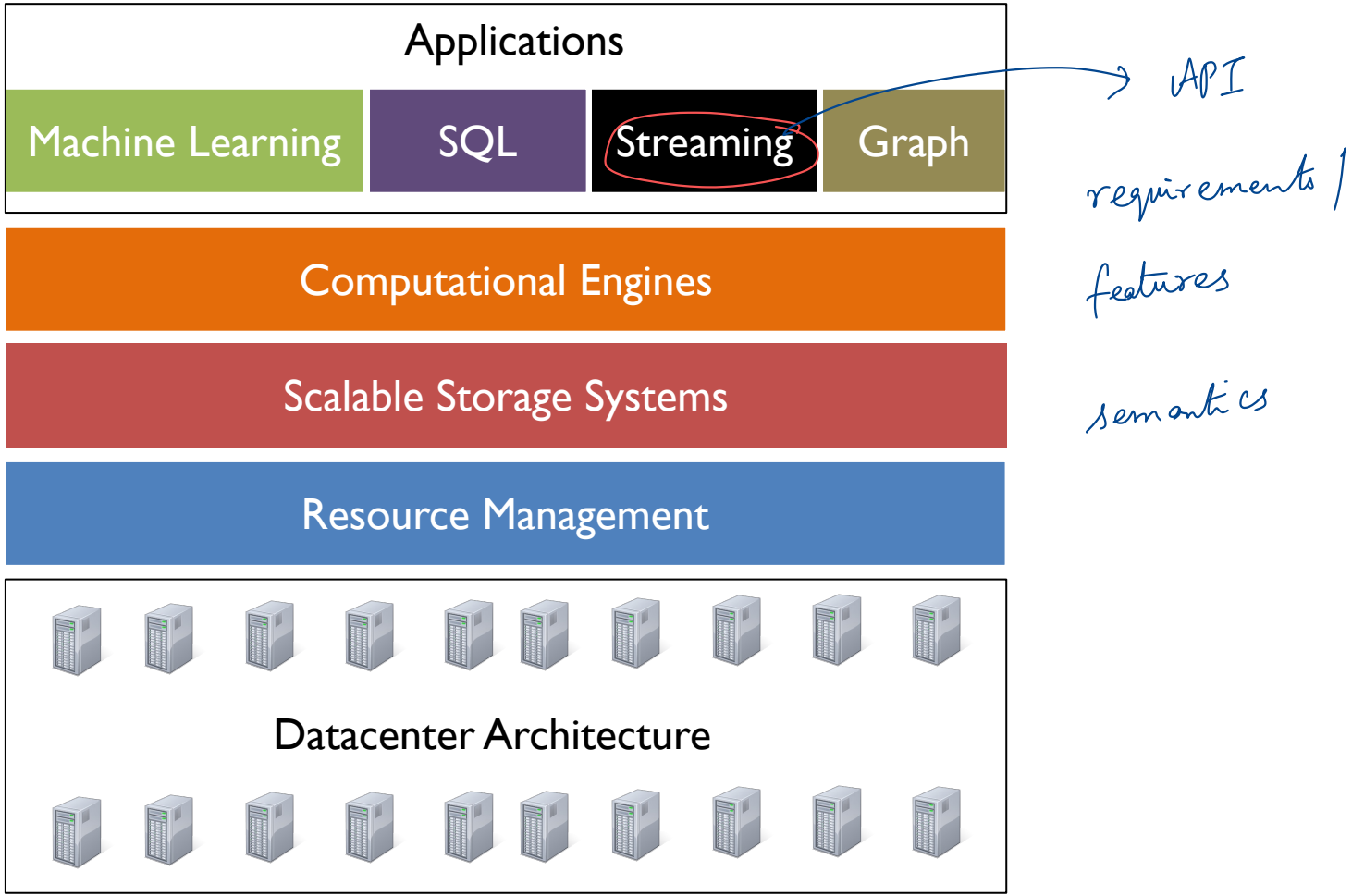
In Progress

- 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


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture
API
requirements /
features
semantics

DASHBOARDS

Input unbounded stream of data

Users express queries. Output

Sales Dashboard

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

Week of Date Closed
December 6, 200 December 25, 20

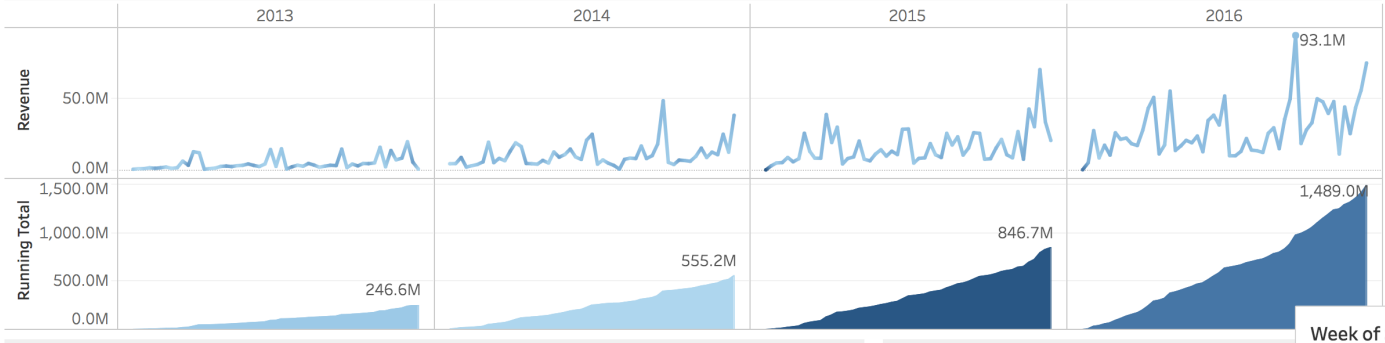
Region
(All)

Country
(All)

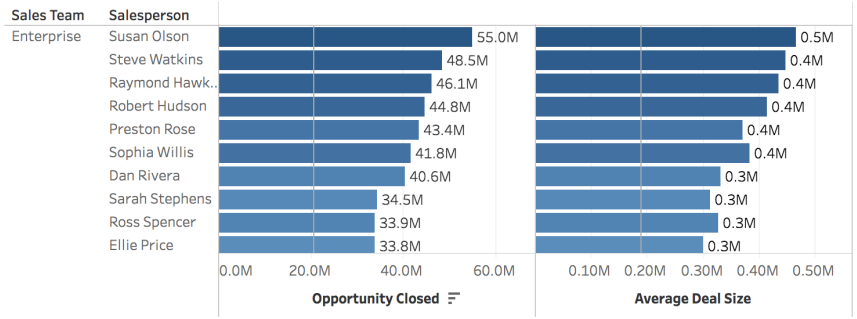
Sales Team
(All)
Small and Midmarket
Enterprise

Avg Deal Size/Salesperson
$147,043 $336,519

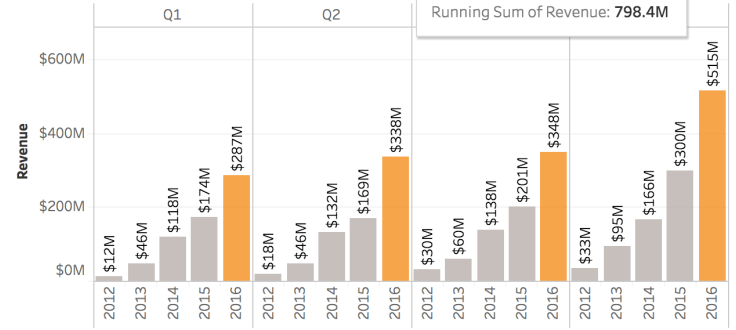
Revenue Over Time



Sales Team Performance

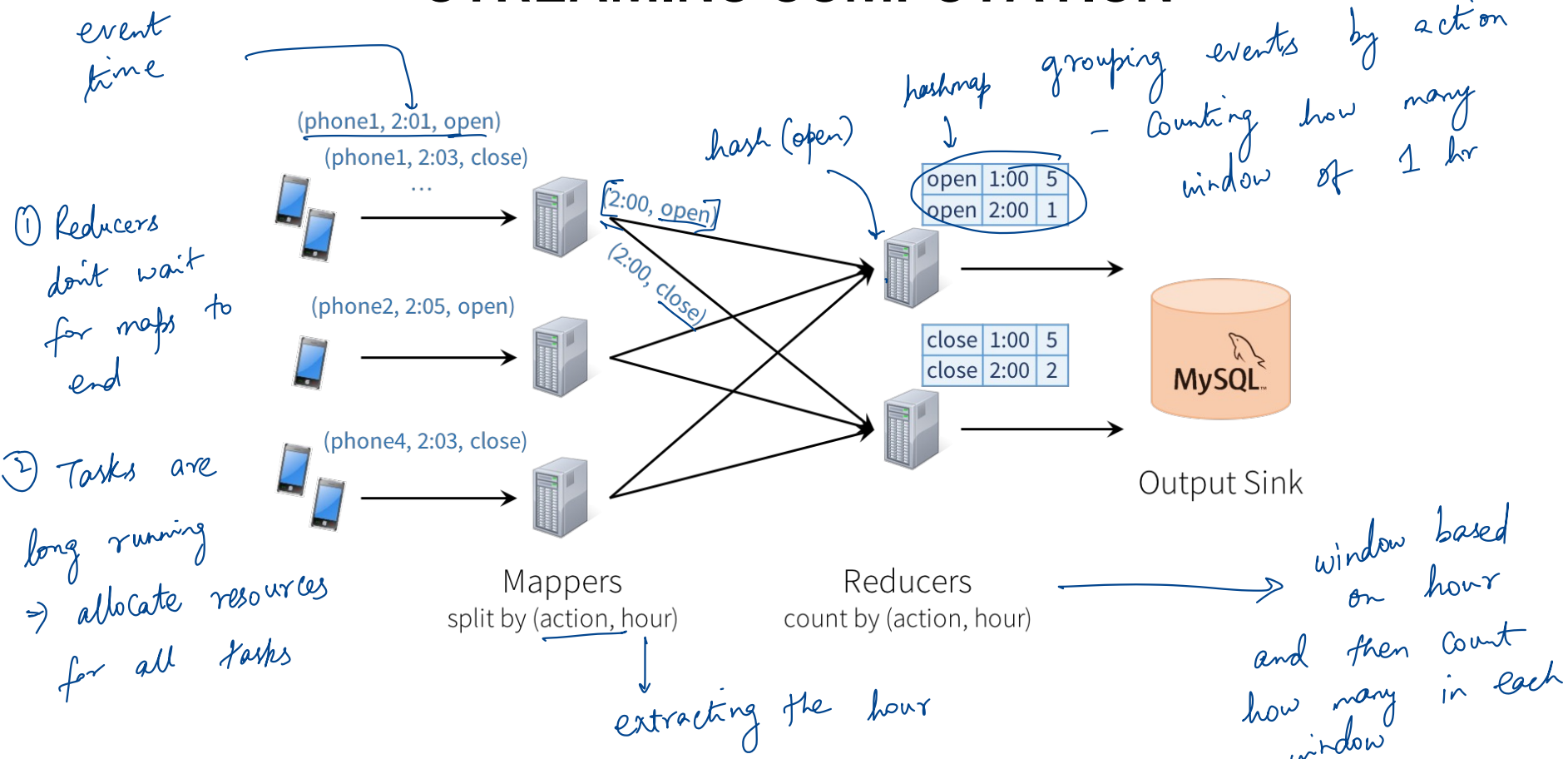


Revenue by Quarter

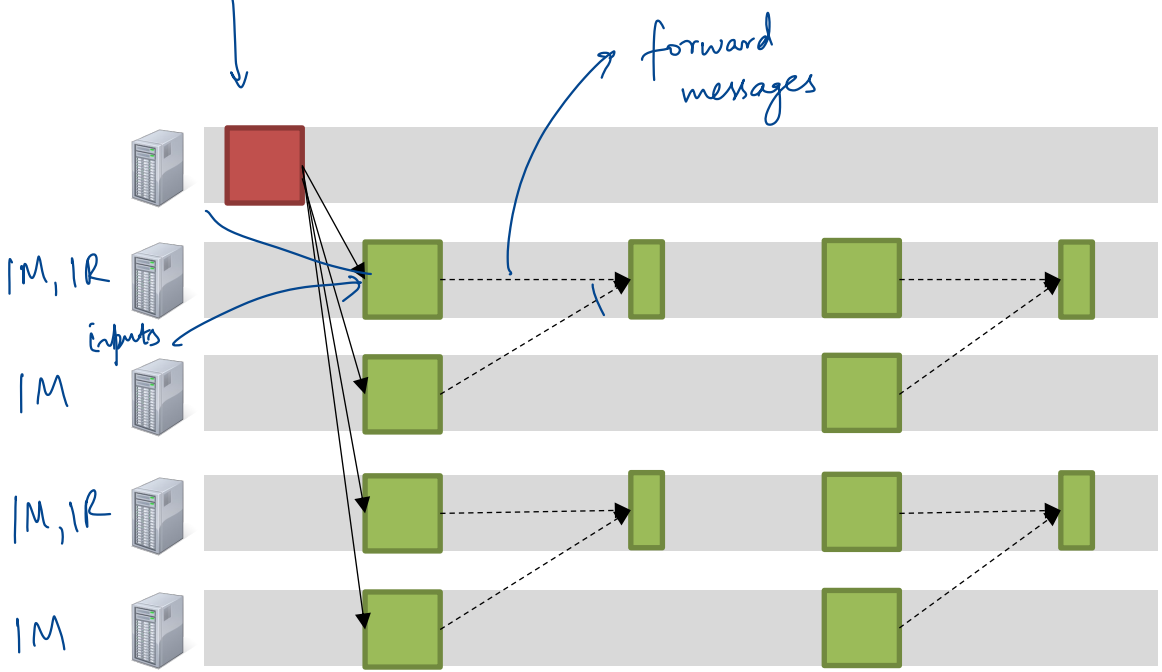


Week of September 4, 2016
Revenue: 14.6M
Running Sum of Revenue: 798.4M

STREAMING COMPUTATION



FLINK: COMPUTATION MODEL



Operators implement Map / Reduce / Windows

Long-lived operators

Mutable State

Operators internally maintain state

Google MillWheel

Flink

Streaming DBs: Borealis, Flux etc

Naiad

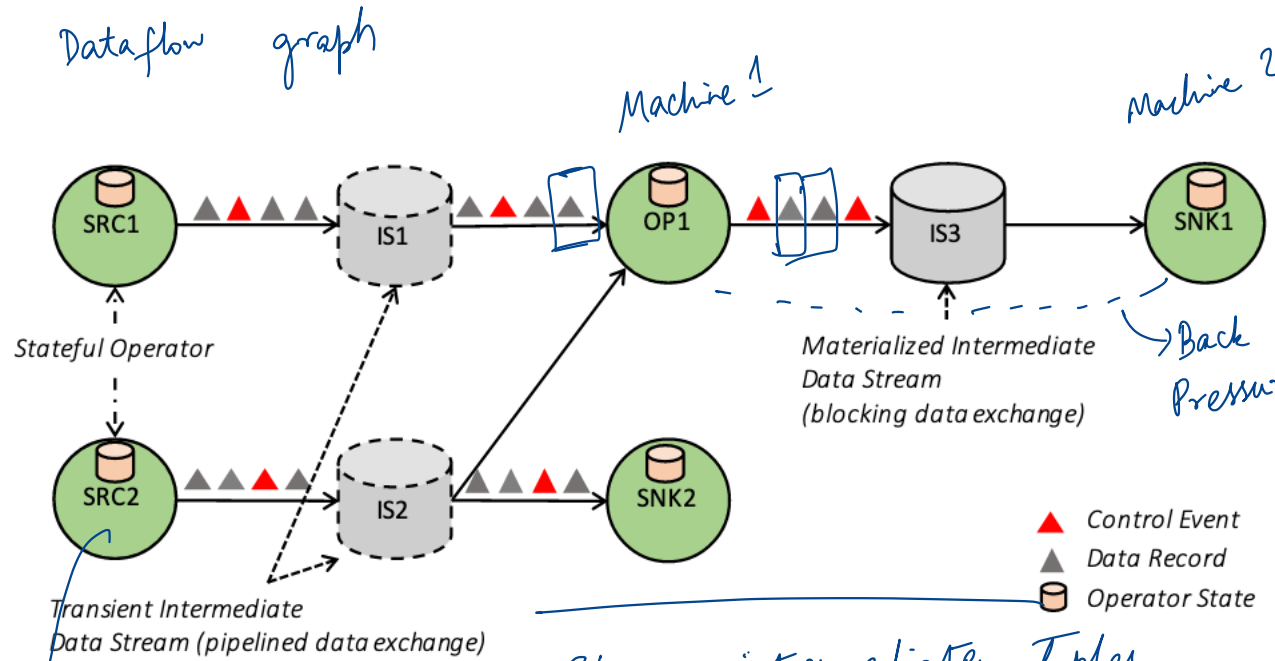
Manager/Driver

Task/Computation

Control Message

Network Transfer

INTERMEDIATE DATA STREAMS



Dataflow graph

Machine 1

Machine 2

- Operators can be partitioned
- Streams can be 1-1 or 1-many
- Pipelined stream
  - Comp. & comm. pipelined

- Store intermediate Tuples
- Back Pressure
- Increase num partition / machines (slowest operators)

operators

STATEFUL OPERATORS

Examples?

→ All windowing operators

→ Aggregation / maintaining summaries

Stateless

→ map() : input → output

→ filter() : input, filter criteria

Challenge

→ need to restore state when op fails.

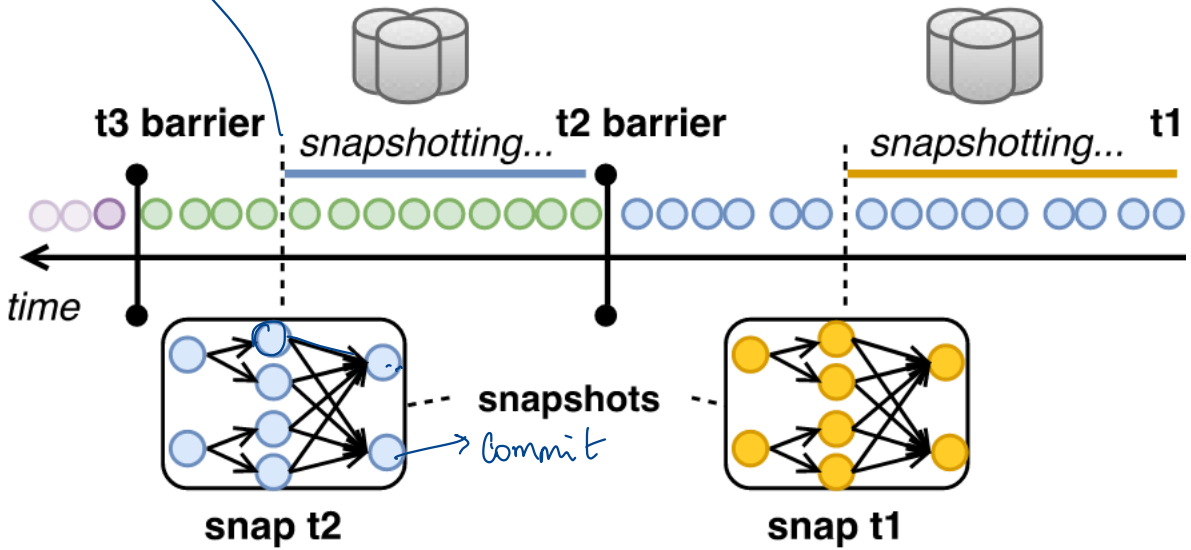
How to ensure fault tolerance?

Explicitly register local variables → annotations

StateBackends that are automatically saved/recovered

FAULT TOLERANCE: CHECKPOINTING

Start snapshot → Control msg to first op
→ Fwd to downstream



Data streams : Control messages

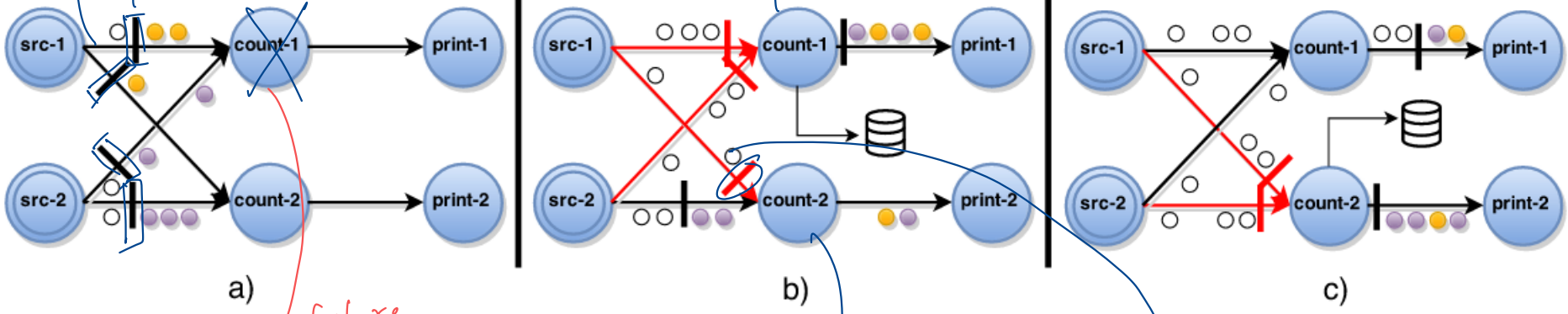
- Cannot replay all the data from beginning
- Input is "replayable"
  - Apache Kafka

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

ASYNCHRONOUS BARRIER SNAPSHOTTING → arXiv

In-order delivery

control message



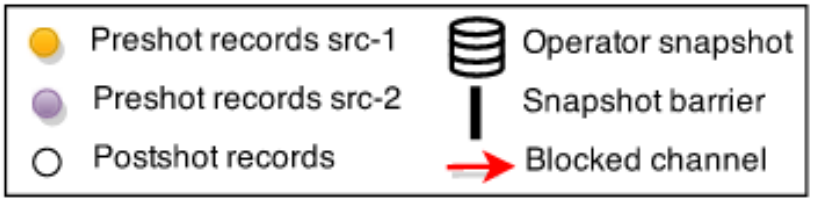
failure

reset all operator to snapshot!

2 inputs. needs to wait for both

only saw 1 barrier so far

no tuple after barrier should be in snapshot



WATERMARKS, WINDOWS

Implements similar model as Dataflow

"Watermarks originate at the sources of a topology" → External to Flink

Propagate through the other operators of dataflow

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

Same idea

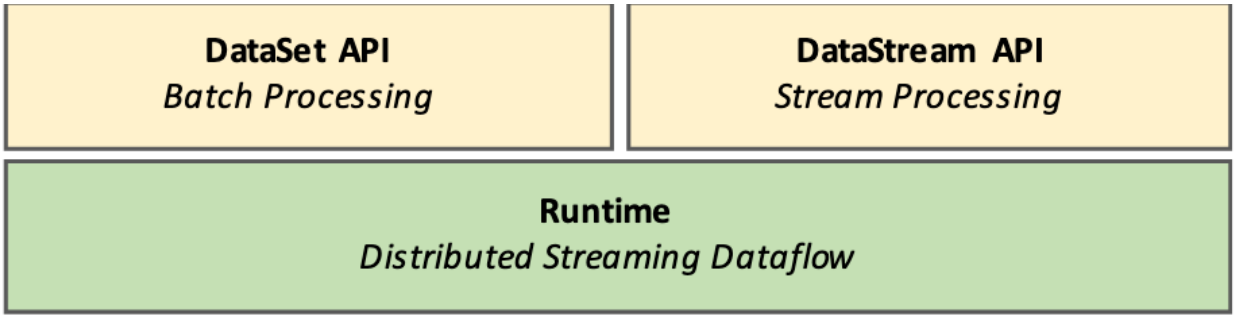
```
stream
.window(SlidingTimeWindows.of(
        Time.of(6, SECONDS), Time.of(2, SECONDS))
.trigger(EventTimeTrigger.create())
```

COMBINING BATCH, STREAMING

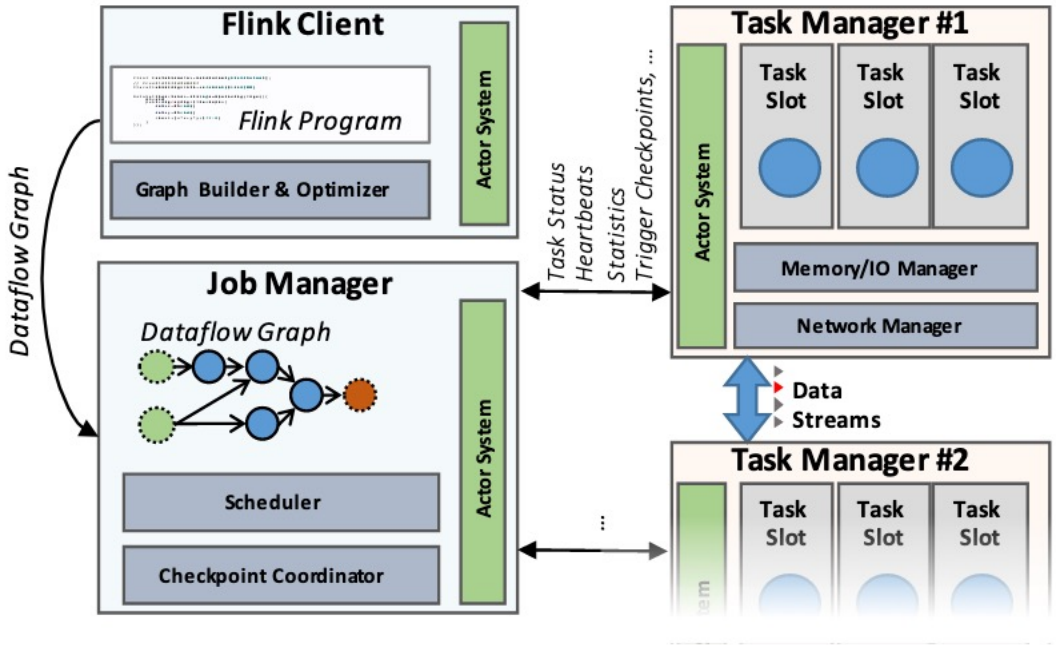
Blocked DataStreams → Intermediate data in MR

Turn off periodic snapshots → Restart job

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



OVERALL ARCHITECTURE



SUMMARY

Stream processing → Increasingly important workload trend

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

Distribtuted streaming dataflow

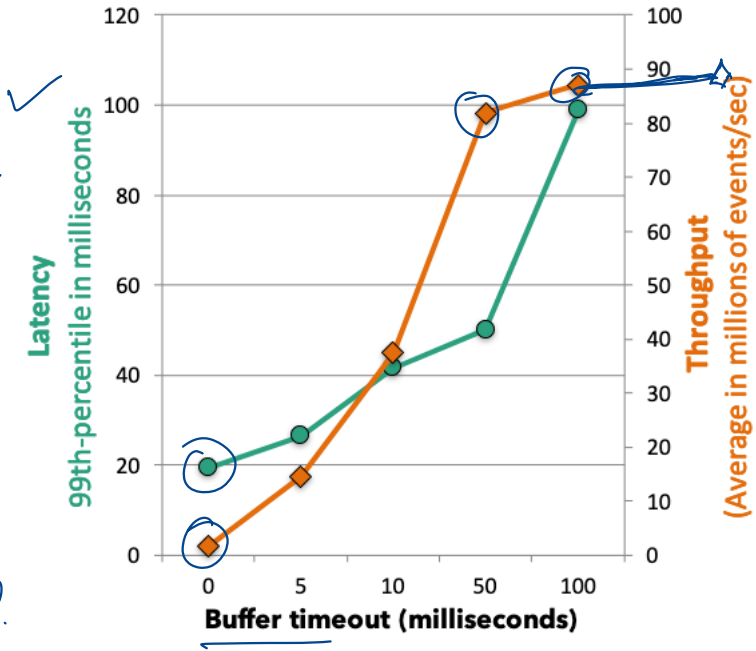
- Stateful operators
- Checkpointing based FT

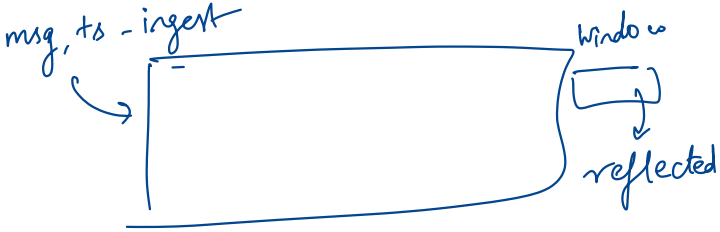
DISCUSSION

https://forms.gle/idnKJWoBFHpRNC2j7

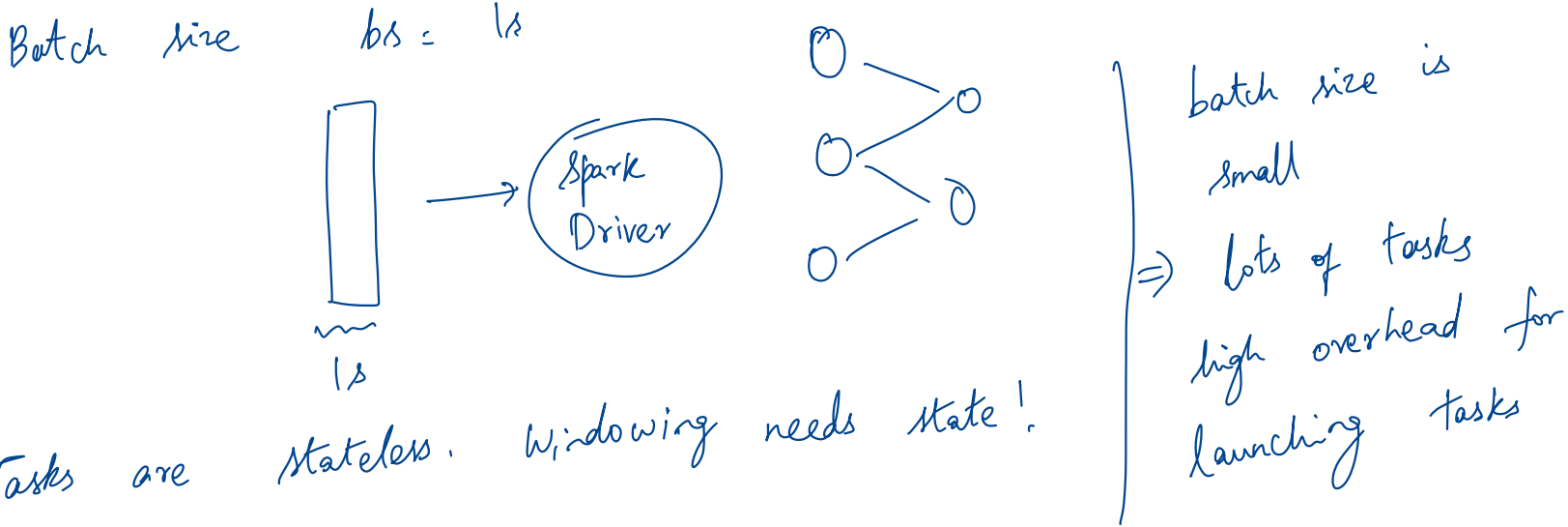
Data Streams

- Small buffer timeout
  - low tput ✗
  - low 99th-ile latency ✓
- Very large timeout
  - saturate at some tput
  - latency might keep going up?





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?



SUMMARY

Next class: Spark Streaming