15-319 / 15-619
Cloud Computing

Recitation 14

November 26th 2019

Overview

- **Last week's reflection**
  - Team Project - Phase 3 - Live Test
- **This week's schedule**
  - Phase 3 report
    - Deadline **TODAY** Nov 26, 23:59:00 ET
  - Project 4.3
    - Deadline **FRIDAY** Dec 6, 23:59:59 ET
  - Project 4.3 Reflection Feedback
    - Deadline **SUNDAY** Dec 8, 23:59:59 ET
  - Course survey (2% bonus!)
    - Deadline Saturday Dec 6, 23:59:59 ET

Project 4

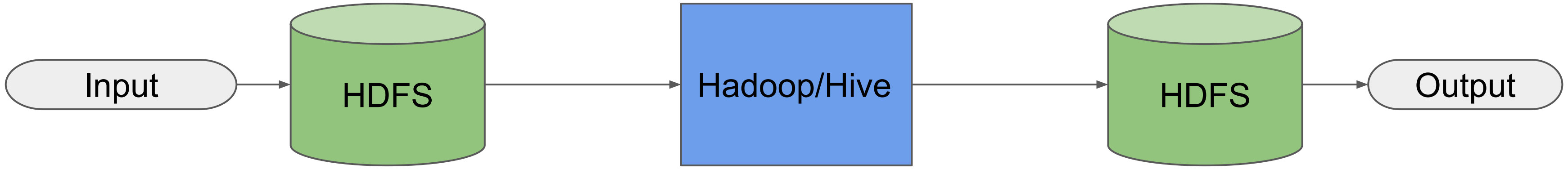
- Project 4.1
  - Iterative Programming Using Apache Spark
- Project 4.2
  - Machine Learning on the Cloud
- Project 4.3
  - Stream Processing using Kafka & Samza

Stream vs Batch Processing

- Batch processing
    - Data parallel, graph parallel
    - Iterative, non-iterative
    - Runs once in few hours/days
    - Historical data analysis
    - Not well suited for real time events streams
- Stream processing
    - Process events as they come
    - Real time decision making
    - Sensor streams/web event data

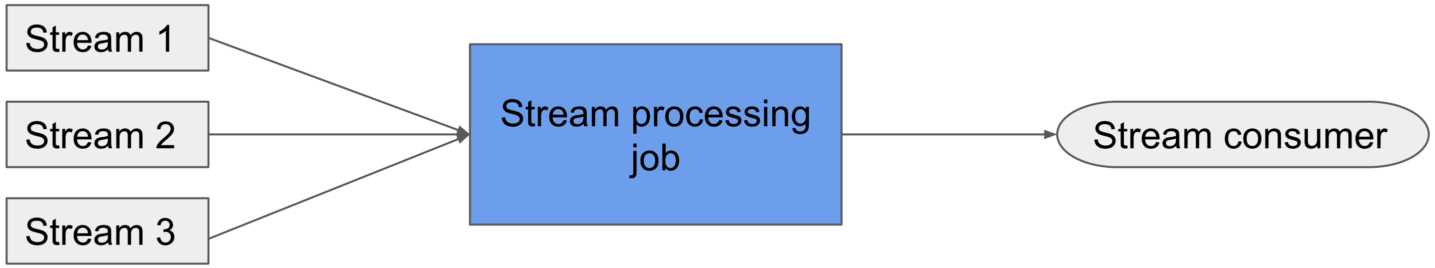
Typical batch processing job

- Input is collected into batches and processing is done on the input data
- Output is consumed later at any point of time - the data does not lose much of its “value” with time



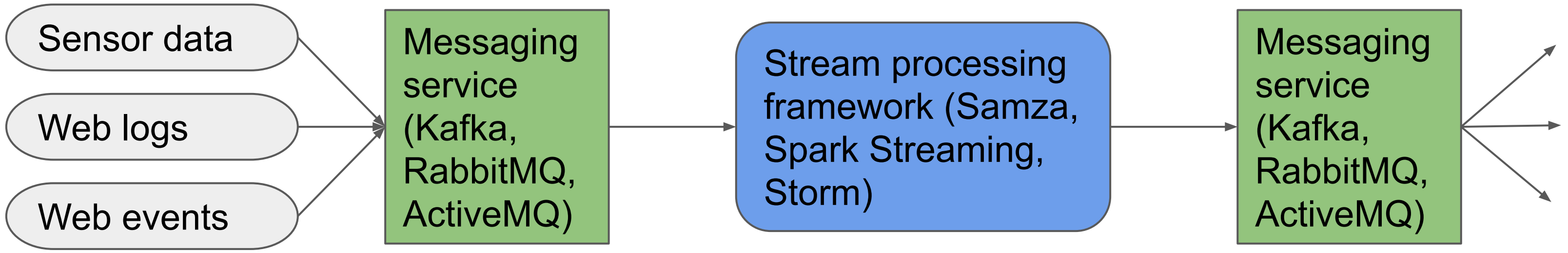
Typical stream processing job

- Data is processed immediately (few seconds)
- The processed data is used by downstream consumers for real time decision/analytics immediately



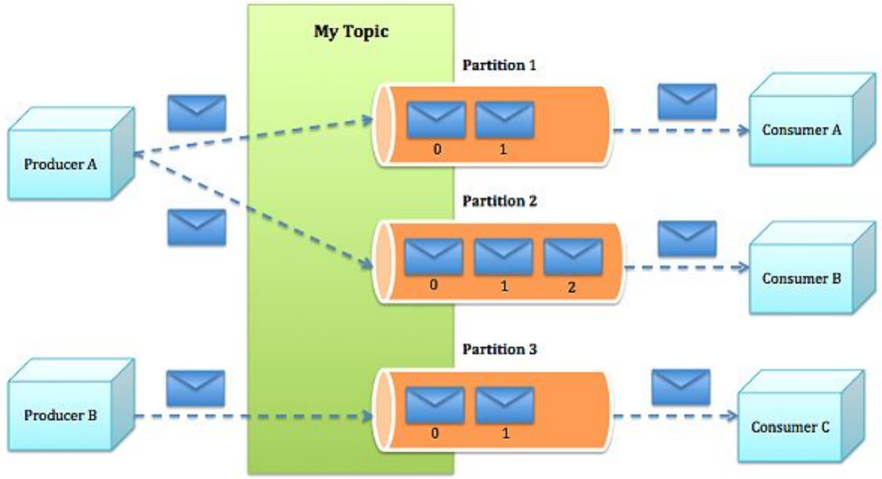
Typical stream processing components

- An event producer - Sensors, web logs, web events
- A messaging service - Kafka, RabbitMQ, ActiveMQ
- A stream processing framework - Samza, Storm, Spark Streaming


Sensor data
Web logs
Web events
Messaging service (Kafka, RabbitMQ, ActiveMQ)
Stream processing framework (Samza, Spark Streaming, Storm)
Messaging service (Kafka, RabbitMQ, ActiveMQ)

Apache Kafka

- A distributed messaging system developed at LinkedIn.

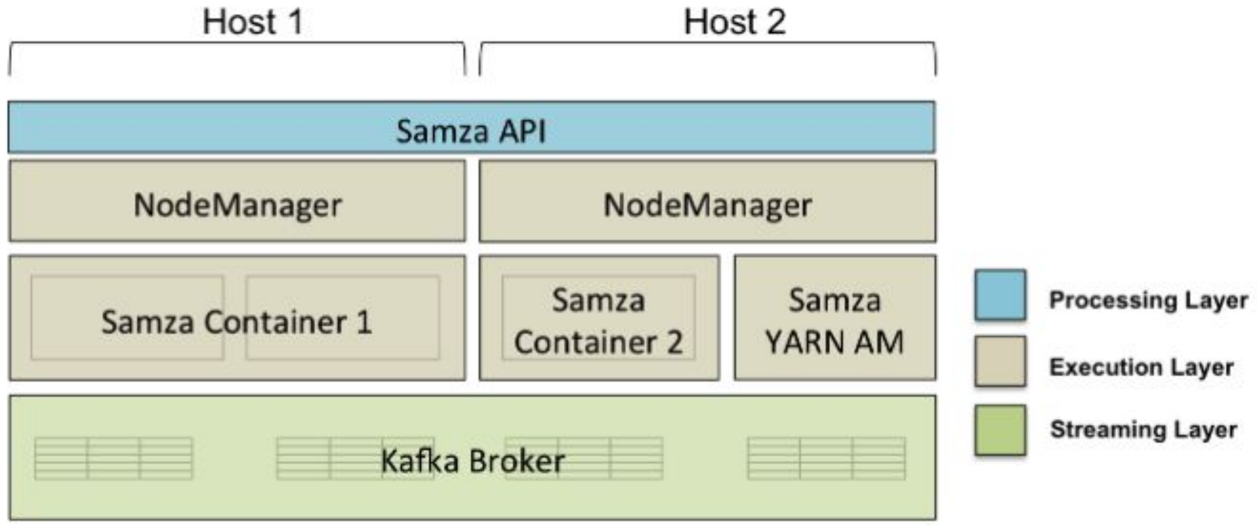


Semantic partitioning in Kafka

- Each topic (stream) is partitioned for scalability across all nodes in the Kafka cluster
- Default partitioning attempts to load balance the messages
- Streams can also be partitioned semantically by user - key of the message
- All messages with the same key arrive to the same partition
- Fault-tolerance: Replication
  - One leader and zero/more followers
  - Replication factor
  - ISR (in-sync replicas)

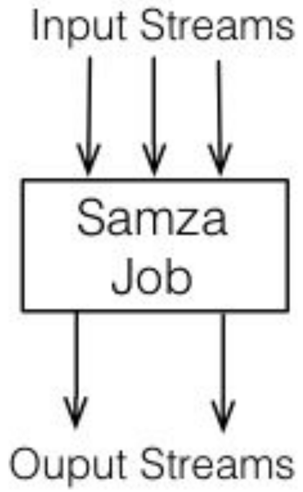
Apache Samza

- Stream processing framework developed at LinkedIn
- Consists of 3 layers:
  - streaming, execution and processing (Samza) layer
- Most common use: Kafka for streaming, YARN for execution



Apache Samza

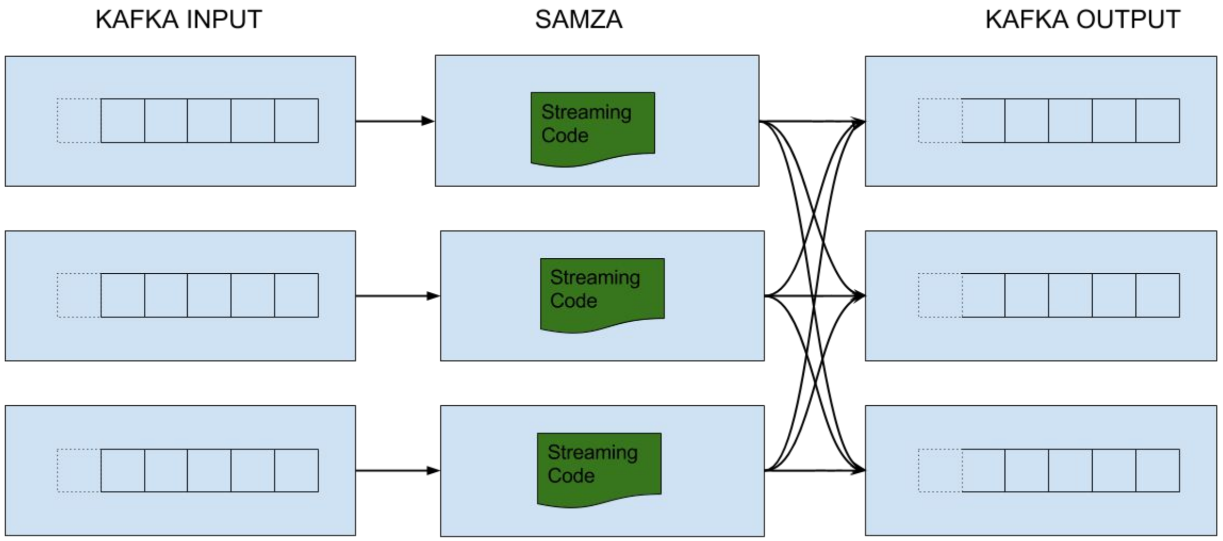
- Programmer uses the Samza API to perform stream processing
- Each partition in Kafka is assigned to a <u>single</u> Samza task instance



Stateful stream processing in Apache Samza

- Calculate sum, avg, count, etc.
- State in remote data store? - slow
- State in local memory? - machine might crash
- Solution - persistent KV store provided by Samza
    - Changes to KV store persisted to a different stream (usually Kafka) - replay on failure
    - RocksDB currently supported as a persistent KV store
        - You MUST use a persistent KV store for P4.3!

Putting Kafka and Samza Together



Project 4.3-Three Tasks

- Use Kafka to produce streams and use Samza to join the streams and output client-driver match like Uber.
- Test cases are provided for the all the tasks. Your solution should pass the provided test cases.

Task 1	**Kafka producer API, Tracefile**
Task 2	**Samza API, Client-driver Match**
Task 3	**Samza API, Advertisement-rider match**
Bonus Task	**Samza API, Enhanced ad-recommendation service**

Project 4.3 - Task 1

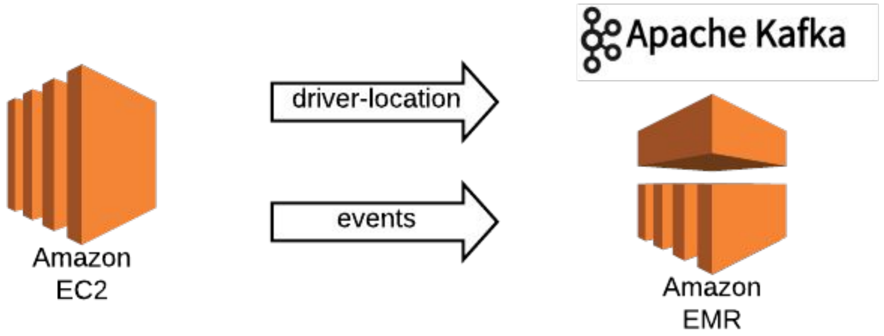
- Simulate the scenario that the **drivers** update their locations on a regular basis as they move in the city and the **clients** request rides at some time.
  - Data
    - Tracefile -> Two streams
    - Type:
      - DRIVER_LOCATION

        -> driver_locations stream
      - LEAVING_BLOCK, ENTERING_BLOCK, RIDE_REQUEST, RIDE_COMPLETE

        -> events stream

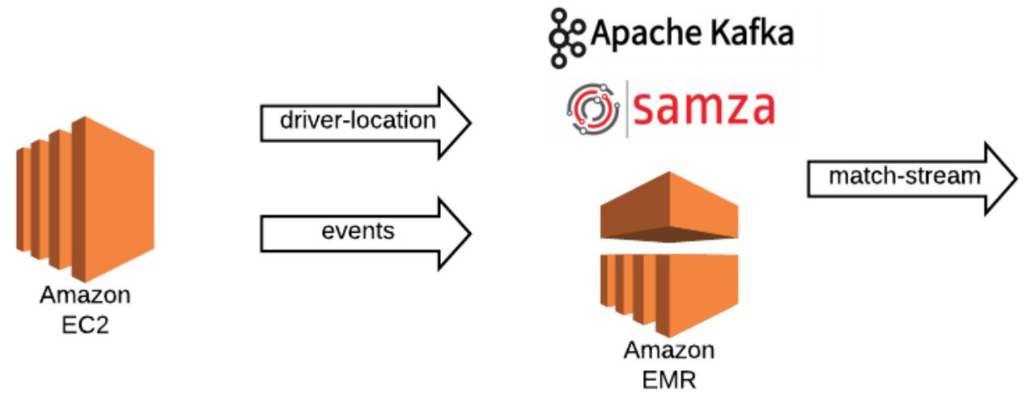
Project 4.3 - Task 1

- Task 1
  - You will run your producer program on your AMI instance.
  - The producer program will publish the data into Kafka brokers.
  - The submitter for Task 1 is located on the AMI instance.



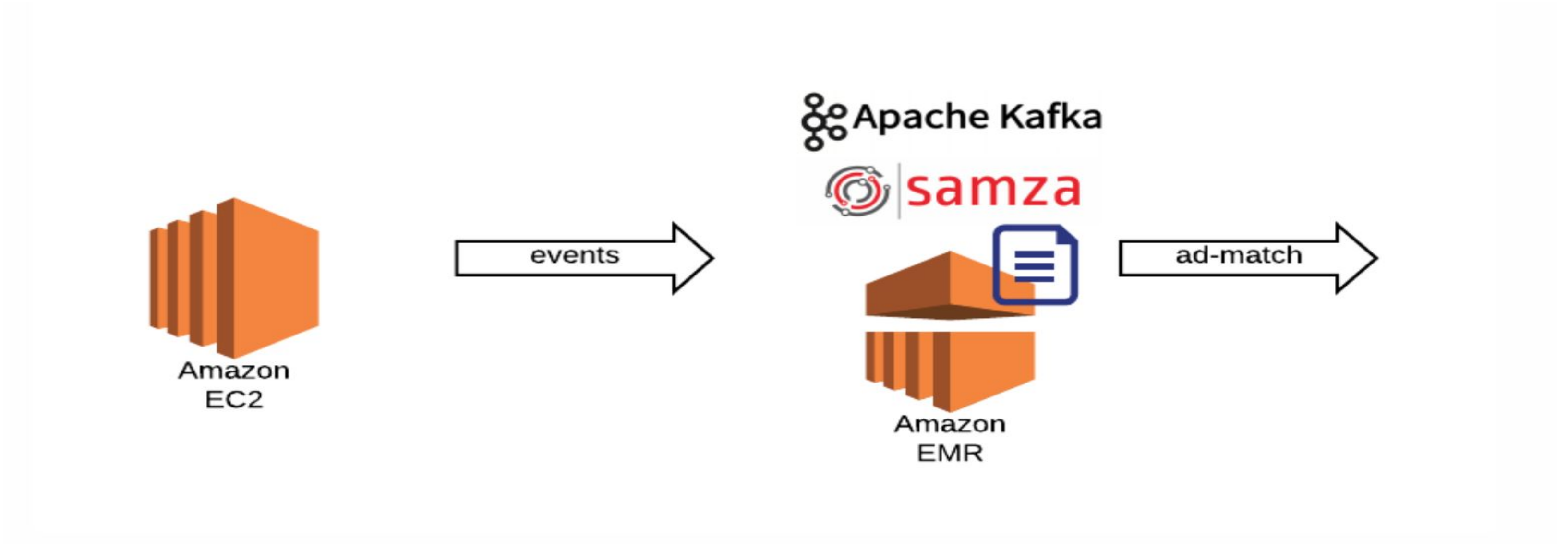
Project 4.3 - Task 2

- Task 2
  - Use the same producer program used in Task 1.
  - You need to find the best match of a ride request with a driver located in the same block as the rider based on published data.



Project 4.3 - Task 3

- Task 3
  - You need to find the best advertisement to place for a specific user.
  - You need to utilize static data(user profile, health status and interests) and stream data to make this decision.



Hints for Task 2 & Task 3

You need to:

1. Change the Network Address in **config/*.properties** files, and ***TaskApplication.java** files. The Network Address is given when you deploy Samza to your cluster.

2. Configure the low level Task Application in ***TaskApplication.java** files:
   a. Create inputDescriptor and outputDescriptor
   b. Attach systemDescriptor in taskApplicationDescriptor
   c. Attach inputDescriptor and outputDescriptor in taskApplicationDescriptor

3. Make sure that you **do not overwrite** the tracefile when copying your code files to the workspace instance or the EMR cluster.

Project 4.3 - Debugging

- Debugging (IMPORTANT!)
  - Use the YARN UI
  - Output a kafka stream for debugging
  - Yarn application commands
    - yarn application -list
  - YARN container logs
    - on the machine where the YARN container is running
- Read the debugging section in the write-up carefully!
- Include the error message when you post on Piazza!

Project 4.3 - Bonus Task

- Bonus task - Advanced Ad Matching.
- Change your ad match rule as described in the writeup.
- **<u>Task</u>**: Consider the destination direction when you perform ad matching.
- The logic in this bonus task will be manually graded, so make sure that you submit your code.

P4.3 Grading

- Skeleton code also provides the submitters
- Follow the instructions in the submitter
  - Prompts for starting the Kafka Producer and Samza job
- We will look for the usage of KV stores and reasonably efficient and well-tested code
  - Do not iterate through ALL drivers or businesses to find the best match!

TEAM PROJECT
Twitter Data Analytics

Team Project Time Table

Phase (and query due)	Start	Deadlines	Code and Report Due
Phase 1 • Q1, Q2	Monday 10/07/2019 00:00:00 ET	**Checkpoint 1, Report: Sunday 10/13/2019 23:59:59 ET** **Checkpoint 2, Q1: Sunday 10/20/2019 23:59:59 ET** **Phase 1, Q2: Sunday 10/27/2019 23:59:59 ET**	Phase 1: Tuesday 10/29/2019 23:59:59 ET
Phase 2 • Q1, Q2,Q3	Monday 10/28/2019 00:00:00 ET	Sunday 11/10/2019 15:59:59 ET	
Phase 2 Live Test (Hbase AND MySQL) • Q1, Q2, Q3	Sunday 11/10/2019 17:00:00 ET	Sunday 11/10/2019 23:59:59 ET	Tuesday 11/12/2019 23:59:59 ET
Phase 3 • Q1, Q2, Q3 (Managed services)	Monday 11/11/2019 00:00:00 ET	Sunday 11/24/2019 15:59:59 ET	
Phase 3 Live Test • **Q1, Q2, Q3 (Managed services)**	Sunday 11/24/2019 17:00:00 ET	Sunday 11/24/2019 23:59:59 ET	**Tuesday 11/26/2019 23:59:59 ET**

Team Project, Overall Winners

- Attend the **Thursday** (12/5) cupcake party recitation
  - To see the winners of the Team Project
  - To listen to the top teams and their implementations
  - Eat a lot of cupcakes
  - Have fun!

Upcoming Deadlines

- Team Project - Phase 3 report
  - **Due: TODAY 11/26/2019 23:59 PM Pittsburgh**
- Project 4.3 : Stream Processing with Kafka/Samza
  - **Due: *FRIDAY* 12/6/2019 11:59 PM Pittsburgh**
- Apply for the S20 TA job, there is still time
  - https://forms.gle/3ddaeYqa2d8qNwcs5
- Complete the course survey (to be announced on Piazza)
  - 2% bonus for the overall course grade (Don’t miss it!!!)
- Cupcake Party (GHC 4307 Pittsburgh and SV 109)
  - Thursday 12/5/2018 4:30 PM ET Pittsburgh, 1:30 PM PT SV

Questions?