

# 15-319 / 15-619

# Cloud Computing

Recitation 15

December 8<sup>th</sup> & 10<sup>th</sup> 2015

# Overview

- **Recent Tasks reflection**
  - Project 4.2
  - Quiz 12
- **This week's schedule**
  - Project 4.3
- **Demo**
- **Twitter Analytics Project: Recap**

# Reminders

- Monitor AWS expenses regularly and tag all resources
  - Check your bill (Cost Explorer > filter by tags).
- Piazza Guidelines
  - Please tag your questions appropriately
  - Search for an existing answer first
- Provide clean, modular and well documented code
  - Large penalties for not doing so.
  - **Double check** that your code is submitted!! (verify by downloading it from TPZ from the submissions page)
- Utilize Office Hours
  - We are here to help (but not to give solutions)

# Project 4.2 FAQ

- How to calculate the contributions in PageRank?  
How to deal with dangling nodes?
  - Refer to the formula in the writeup. A node receives contributions from its followers, not followee. Distribute the rank of dangling nodes equally to all nodes.
- Spark job is stuck, there are errors in my job
  - Launch more machines and use instances with enough memory and disk space.
  - Check in the web console if the job uses all the available executors and memory. If not, change your configuration.

# Project 4

- Project 4.1
  - MapReduce Programming Using YARN
- Project 4.2
  - Iterative Programming Using Apache Spark
- **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
  - Unsuitable for real time events streams
- Stream processing
  - Process events as they come
  - Real time decision making
  - Sensor streams/ web event data

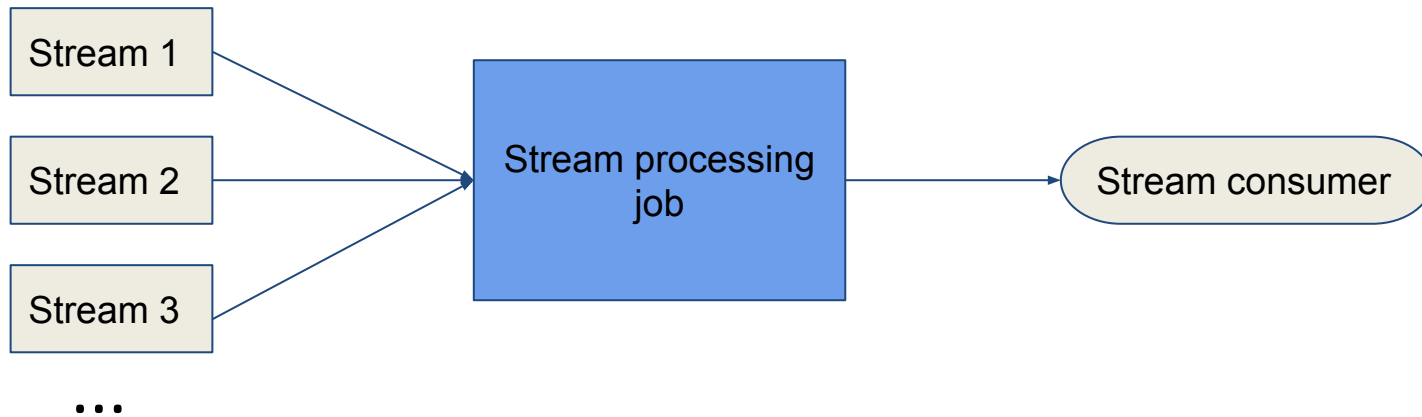
# Example of a batch processing job

- Input is collected into batches and processing is performed 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

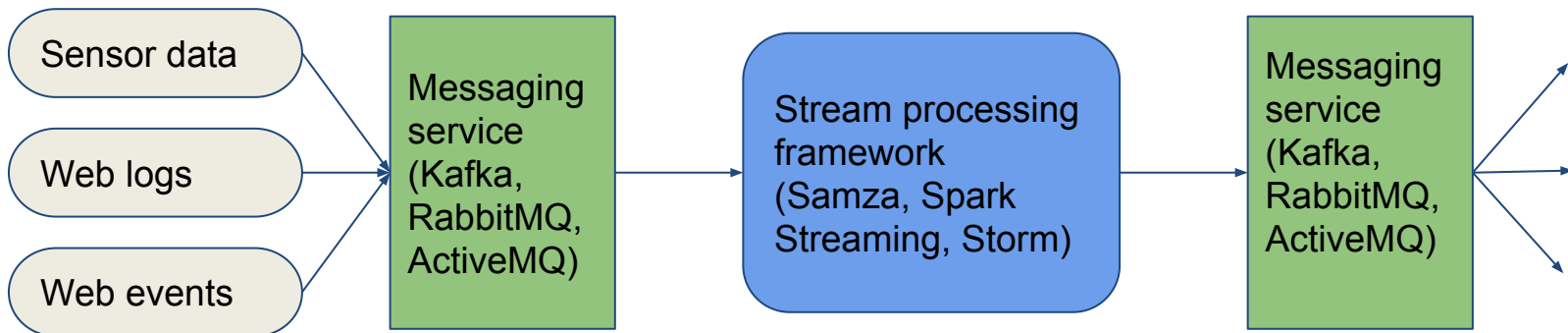
- Data is processed immediately (few seconds)
- The processed data is used by downstream consumers for real time decision/analytics immediately





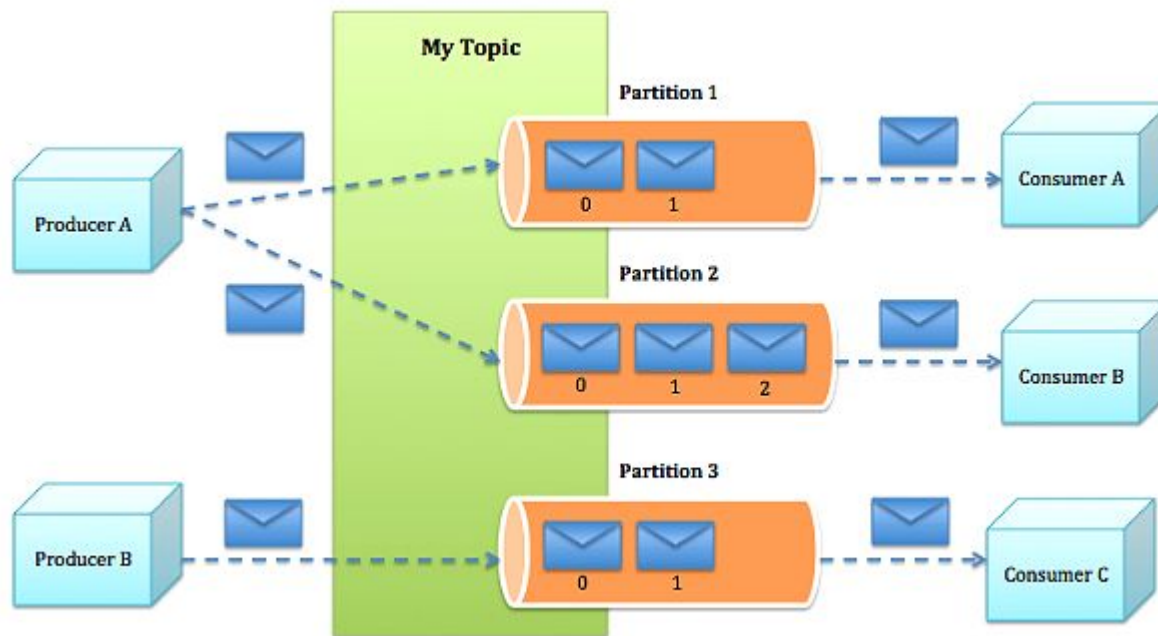
# Typical stream processing components

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



# Apache Kafka

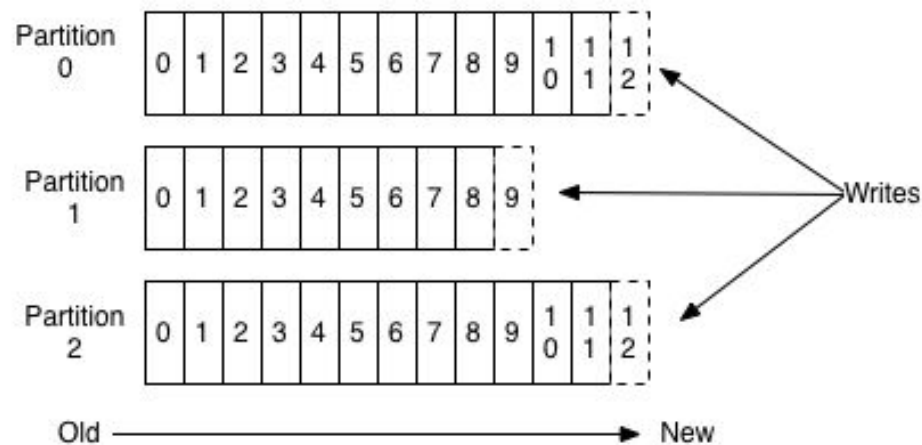
- Developed at LinkedIn as a distributed messaging system.



# Apache Kafka

- Used to integrate data from multiple sources
- Streams (or topics) in Kafka modelled as a “log”
- Different consumers read independently at different offsets in the log

## Anatomy of a Topic

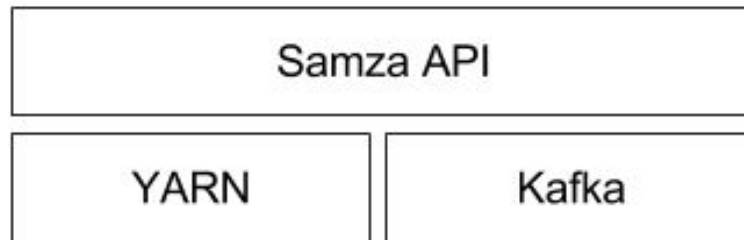


# Semantic partitioning in Kafka

- Each topic (stream) is partitioned for scalability across all nodes in the Kafka cluster
- Default partitioning attempts to load balance
- Streams can also be partitioned semantically by user
  - key of the message
- All messages with the same key come to the same partition

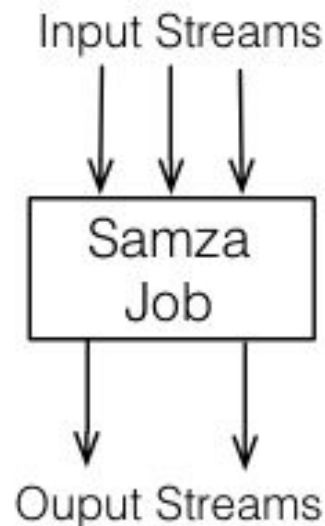
# Apache Samza

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



# Apache Samza

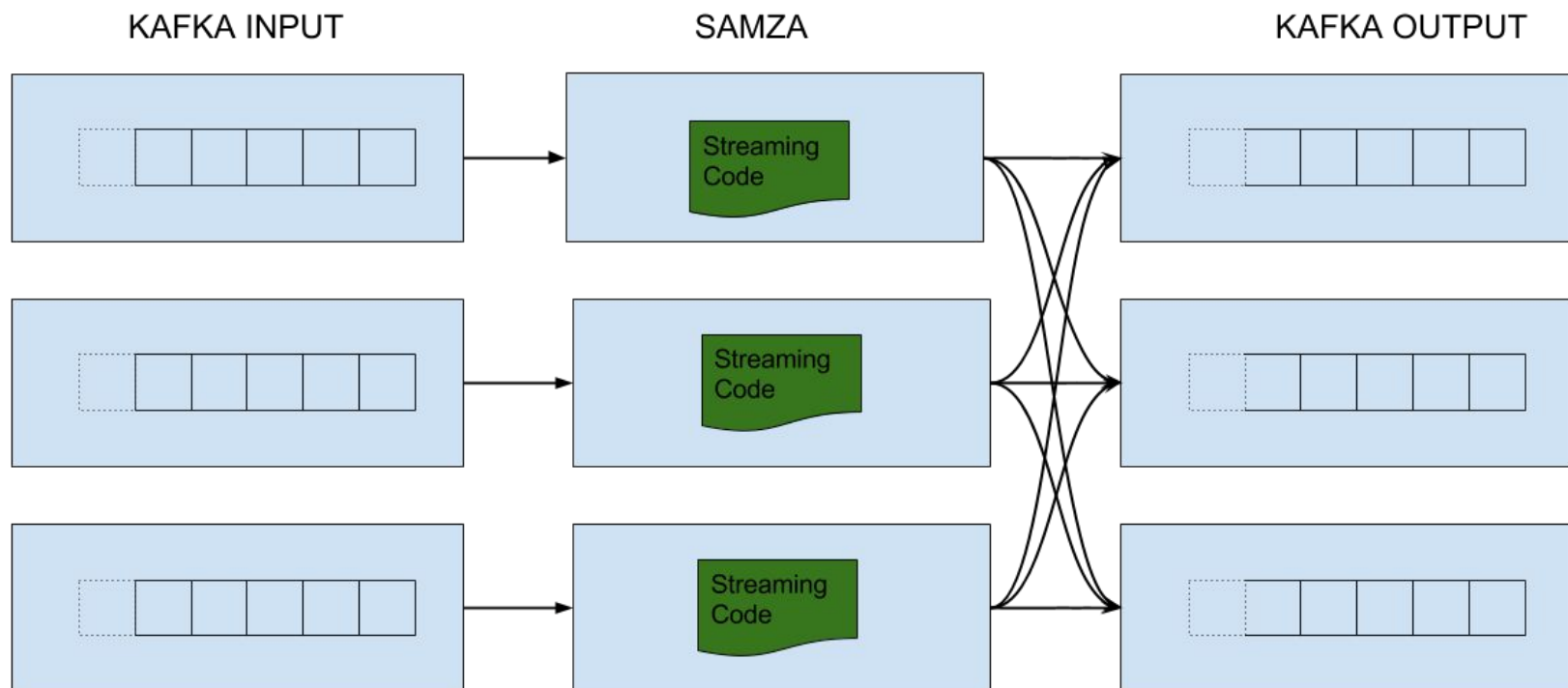
- Programmer uses Samza API to perform stream processing
- Semantic partitioning in Kafka => streaming MapReduce
- Each partition in Kafka is assigned to a single Samza task instance



# Stateful stream processing in Apache Samza

- Calculate sum, avg, count etc.
- State in remote data store? - slow
- State in memory locally? - machine crashes
- 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!

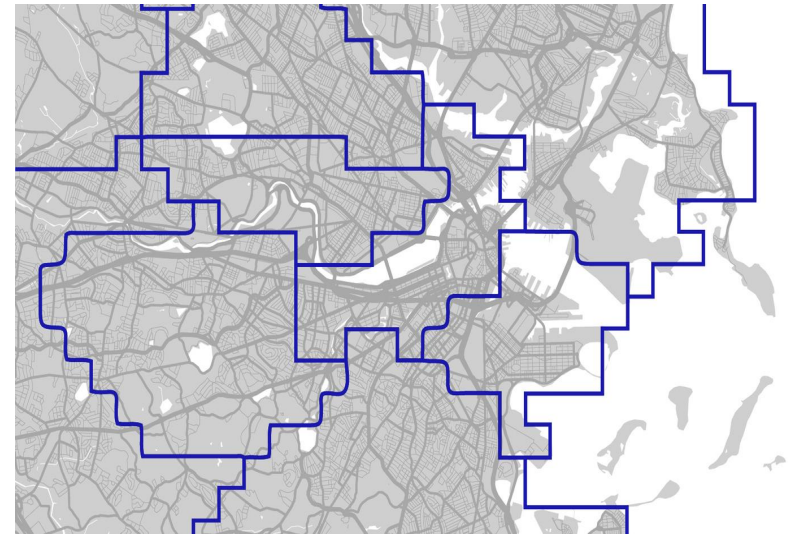
# Putting Kafka and Samza together





# Project 4.3

- Use Kafka and Samza to develop components of a ride hailing app
- Two Streams
  - Stream of driver locations
  - Stream of ride requests
- Find closest driver to a rider



# Project 4.3 - Overview

- We provide a load generator to provide Kafka streams
- Use the Samza API to maintain state and find the closest driver to a rider
- driver-locations stream - stream of driver locations as they move through city
- events stream - stream of events (ride requests, ride complete, etc.)


# Project 4.3

- Bonus task - implement dynamic (aka surge) pricing
- Same streams but different state and different calculations required
- Careful when you move drivers around blocks! - bonus grader is more sensitive to sloppy state management
  - For example: ensure that the count of drivers is not off by one
  - ...

# Grading

- Skeleton code also provides the submitters
- We will look for usage of KV stores and reasonably efficient code
  - no iterating through ALL drivers to find closest!

# Upcoming Deadlines

- Project 4.3 : Stream Processing with Kafka/Samza 
  - Due: 12/11/2015 11:59 PM Pittsburgh
- Apply for S16 of F16 TA job, there is still time
  - [link](#)
- Complete the course survey (announced on Piazza)
  - 2% bonus for the overall course grade (Don't miss it!!!)
- Cupcake Party (Pittsburgh and SV)
  - Thursday 12/10/2015 4:30 PM Pittsburgh, 1:30 PM SV

# Project 4.3

- Demo
- Create a cluster on EMR and ssh onto the master node
- wget the script that's available on S3 and run the script
- locate the logs
- the format of the sample stream
- submitter

# TWITTER DATA ANALYTICS: 15619 PROJECT




# 15619Project

Thursday, 12/10, recitation:

- Cupcakes
- Phase 3 Scores
- Breakdown of Winners
- More...



# Don't Forget!

- Project 4.3 : Stream Processing with Kafka/Samza 
  - Due: 12/11/2015 11:59 PM Pittsburgh
- Apply for S16 of F16 TA job, there is still time
  - [link](#)
- Complete the course survey (announced on Piazza)
  - 2% bonus for the overall course grade (Don't miss it!!!)
- Cupcake Party (Pittsburgh and SV)
  - Thursday 12/10/2015 4:30 PM Pittsburgh, 1:30 PM SV