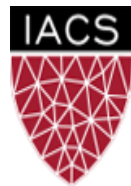


“In God we Trust, all others bring data.”

**W. Edwards Deming, Professor at UColumbia,
1980s**

Lecture C.3: Stream Data Processing

CS205: Computing Foundations for Computational Science
Bill Richmond
AI/ML Evangelist
Amazon Web Services
Spring Term 2020



**INSTITUTE FOR APPLIED
COMPUTATIONAL SCIENCE**
AT HARVARD UNIVERSITY



HARVARD
School of Engineering
and Applied Sciences

Before We Start

Where We Are

Computing Foundations for Computational and Data Science

How to use modern computing platforms in solving scientific problems

Intro: Large-Scale Computational and Data Science

A. Parallel Processing Fundamentals

B. Parallel Computing

C. Parallel Data Processing

C.1. Batch Data Processing

C.2. Dataflow Processing

C.3. Stream Data Processing

Wrap-Up: Advanced Topics

CS205: Contents

APPLICATION SOFTWARE

APPLICATION
PARALLELISM

PARALLEL
PROGRAM DESIGN



Optimization

PROGRAMMING MODEL

OpenACC

Spark

OpenMP

Map-Reduce

MPI

B. BIG COMPUTE

PLATFORM

C. BIG DATA



CLOUD COMPUTING



PARALLEL ARCHITECTURES

Context

Stream Data Processing



Stream

Computes a function of **one data element**, or a **small window of recent data**

Computes something relatively **simple**

Optimize for **latency**, complete each computation in **near-real-time**

Computations are generally **independent**, like **trends** over time

Batch

Computes on **entire data**, usually extremely **large sets of data**

Might compute something **big** and **complex**

Optimize for **throughput**, data processed per second

Calculations access to a **complete set of records**, like **totals** and **averages**



Epic Games continually improves Fortnite for 250+ million players globally



Challenge

They needed a way to process and analyze over 100 PB of data (125M events/min) ingested from game clients and game servers to understand and adapt to player engagement.

Solution

Epic Games turned to AWS for an Amazon S3 data lake in combination with Amazon EMR, Amazon EC2, and Amazon Kinesis.

Benefits

The data provides a constant feedback loop for designers, and an up to the minute analysis of gamer satisfaction to drive gamer engagement.

Data is a strategic asset for every organization

“ The world’s most valuable resource is no longer oil, but **data**. ”*



*Copyright: The Economist, 2017, David Parkins

The pace of transformation is increasing...

BY 2025 THE WORLD WILL SEE:

41 B
IoT CONNECTED DEVICES

79 ZB
DATA CREATED PER YEAR

Source: IDC, Worldwide Global DataSphere IoT Device and Data Forecast, 2019-2023

Users want more value from their data



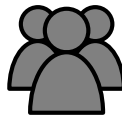
Growing exponentially



From new sources



Increasingly diverse



Used by many people



Analyzed by many applications

Types of analytics users



Architects

Application developers

Business intelligence (BI) analysts

CxO

Data engineers, operations

Data modelers

Data scientists

Data warehouse admins

Database admins (DBAs)

DevOps engineers

LOB knowledge workers

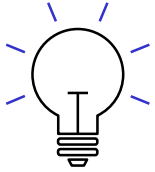
Product managers

IT operations

IT security and governance

VP/director analytics

Common analytics use cases



Data warehouse modernization

Big data and data lakes

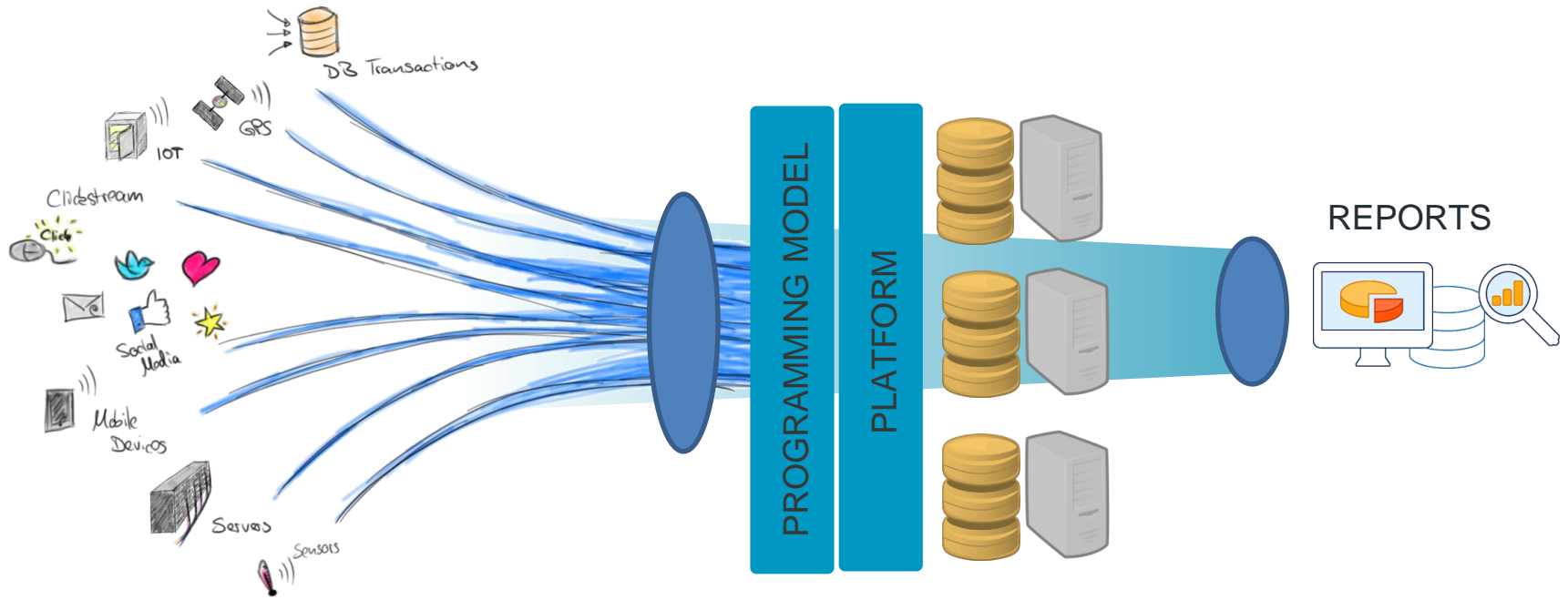
Real-time streaming and analytics

Operational and search analytics

Self-service business analytics

Big Streaming Data

Big Picture



Data Collection
and Cleaning

Data Integration,
Processing and Analysis

Prediction and Statistical
Learning

Big *Streaming* Data

Streaming Data

Stream **Datasets**

- Unbounded: The total dataset is only defined as the amount of data that has entered the system so far.
- Non-Persistent: The working dataset is perhaps more relevant, and is limited to a single item at a time.
- Processing is event-based and does not "end" until explicitly stopped. Results are immediately available and will be continually updated as new data arrives.



Big Streaming Data

Big Streaming

Stream Processing

- Results based on current data, typically one data record or small window
- Perform simple analysis on data in motion
- Optimize for latency: average time taken for a record
- Processing with near real-time requirements where you must respond to changes or spikes and where you're interested in trends over time



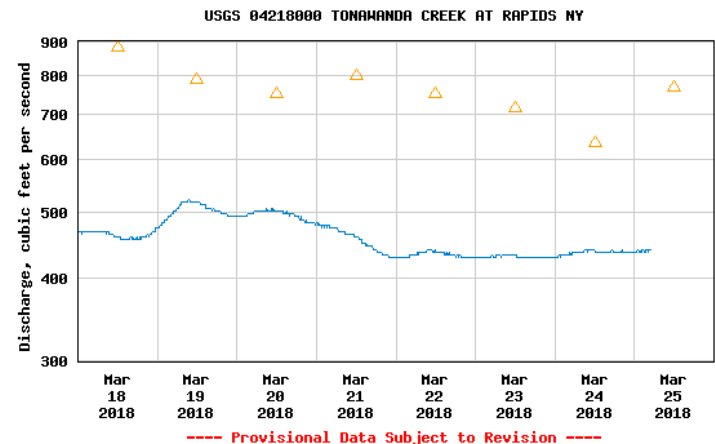
USGS Water Resource

Current Co
PROVISIONA
Predefined d
Current Custom Di

Station
Number
● Erie County, N
04218518
42552007853560
42554307853500

Discharge, cubic feet per second

Most recent instantaneous value: 441 03-25-2018 05:00 EST



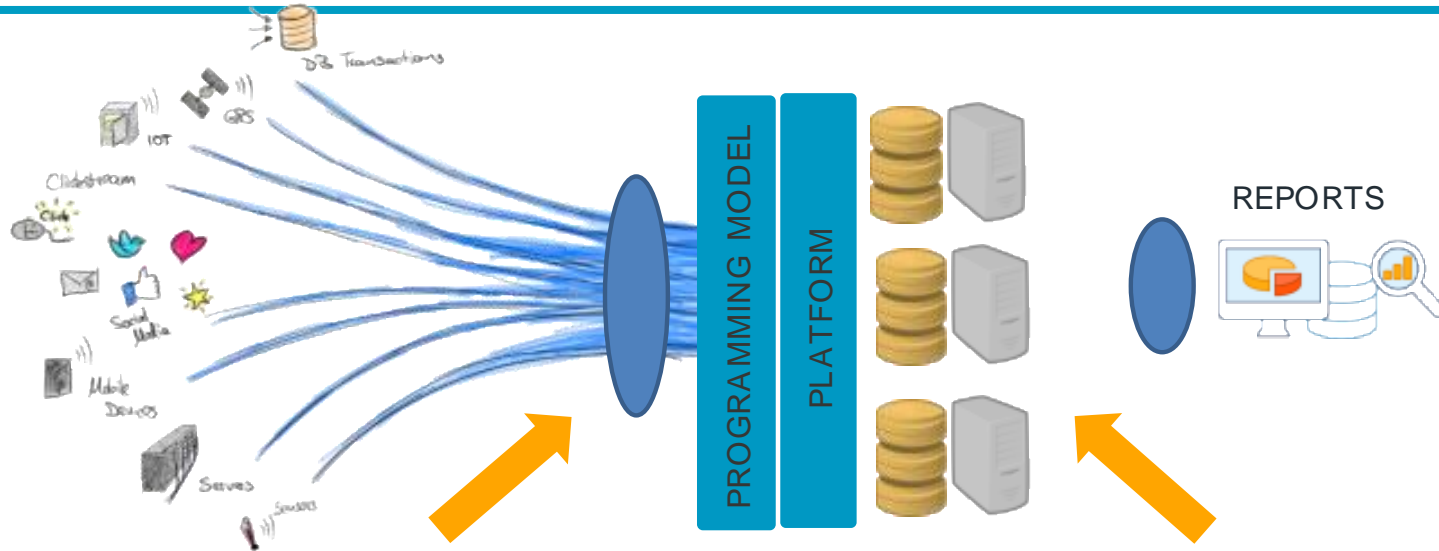
△ Median daily statistic (47 years) — Discharge

Big Streaming Data

Streaming Infrastructure

Parallel Processing Needs

- Achieves very low latency (real-time) => 1 second is too long!
- Integrate with big compute or big data resources
- Close to users (edge)
- Requires elasticity due to address variable demands (cloud)
- Recovers from failures (cloud)
- Scales to thousand of nodes (cloud)



Number of cores/nodes streams to be able to receive all data within the interval

Number of cores/nodes streams to be able to process all data within the interval

Big Streaming Data – Use Case

Tens of millions of users search Zillow daily for information on 100+ million homes and apartments across the U.S.



Company
Industry
Country
Employees
Website

Zillow Group
Real Estate
United States
2,700-plus
www.zillow.com

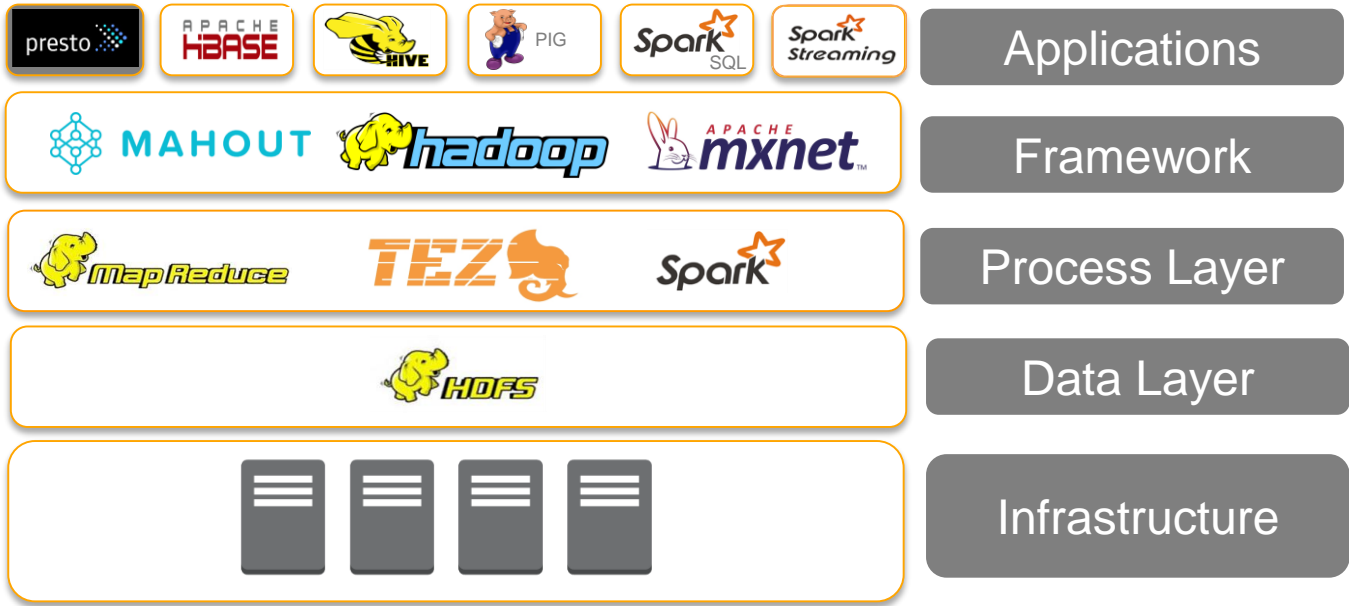
“We can compute Zestimates in seconds, as opposed to hours, by using Amazon Kinesis Streams and Spark on Amazon EMR.

As a result, the Zestimates are more up-to-date and accurate, because they’re built with the absolute latest data.

That’s a huge benefit for our users, who depend on this information to influence their buying or selling decisions.”

- Jasjeet Thind, Vice President of Data Science and Engineering, Zillow Group

Big Streaming Data Platforms

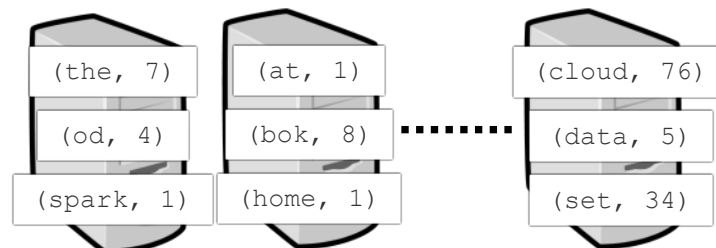


Stream Processing with Spark

The Spark Programming Model

The Fundamental Data Structure - Resilient Distributed Dataset

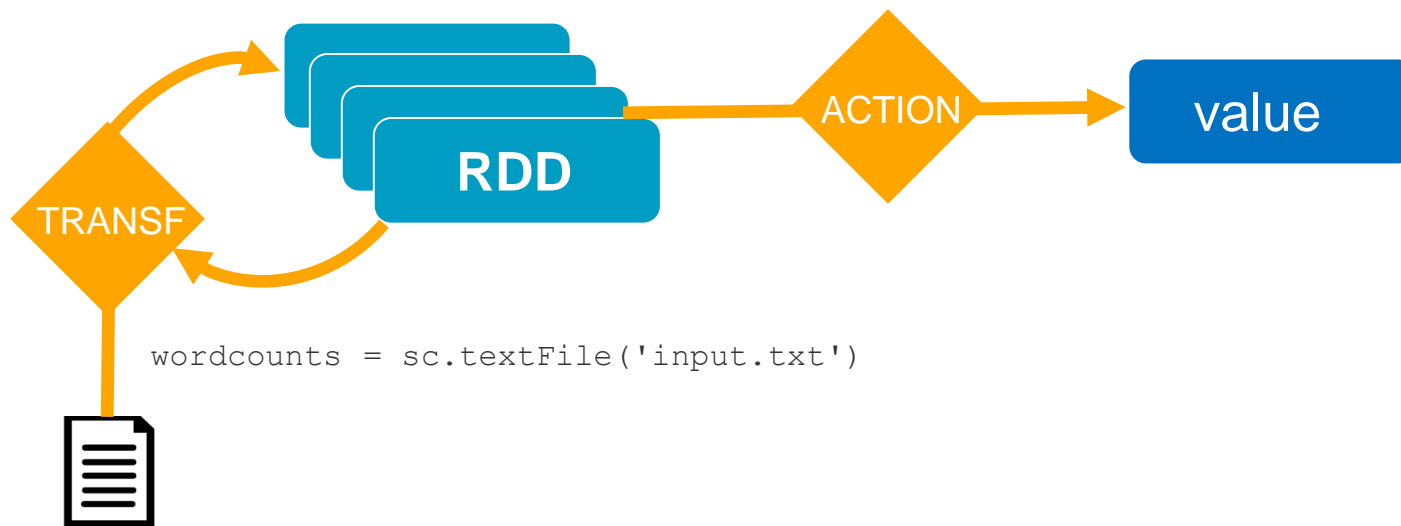
- Resilient: Fault-tolerant
- Distributed: Multiple-node
- Dataset: Collection of partitioned data organized in records



Operations: Transformations and Actions

```
.filter(lambda line: "spark" in line)
```

```
.count()
```

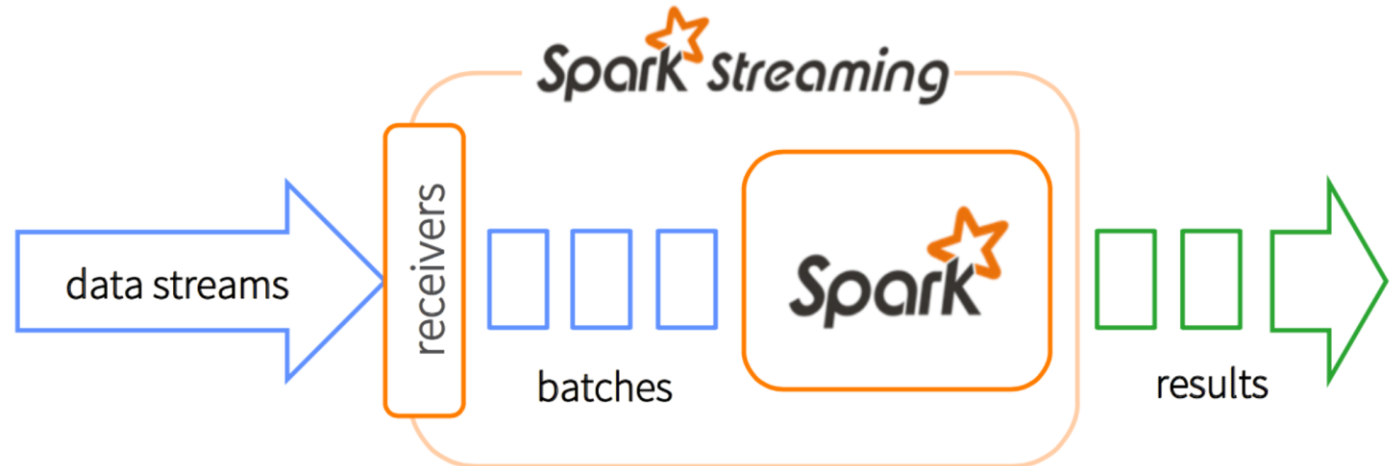


```
wordcounts = sc.textFile('input.txt')
```

Stream Processing with Spark

How Does It Work?

New file in dir
New content in file
TCP socket
Flume
Kafka
Kinesis
Twitter
...



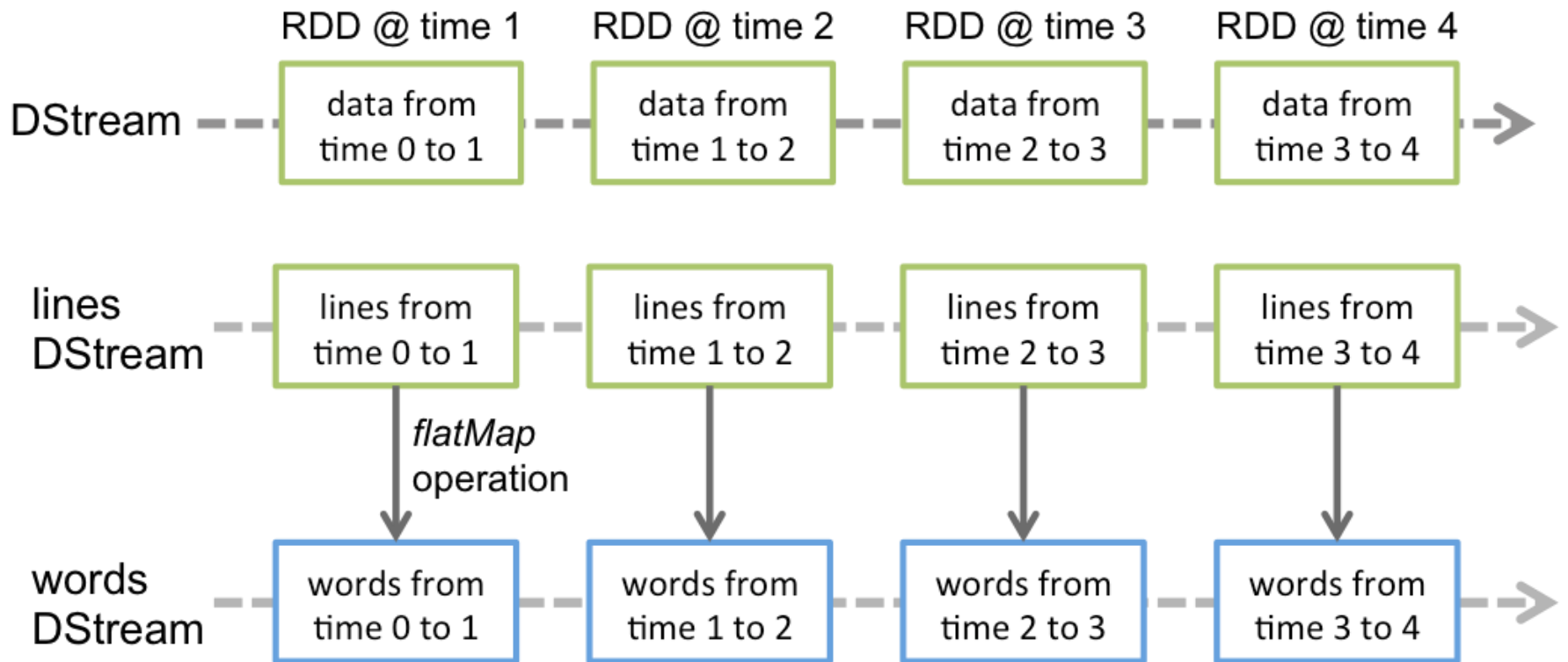
- Split data streams into batches within a time window
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Processed results are pushed out in batches

Stream Processing with Spark

Discretized Streaming

Discretized Stream (DStream)

- Represents a stream of data
- Implemented as a sequence of RDDs



Stream Processing with Spark

Discretized Streaming – Application Anatomy

- Define the `streamingContext`
- Define the input sources by creating input `DStreams`
- Define the streaming computations by applying transformation and output operations to `DStreams` (RDD-like)
- Start receiving data and processing it using `streamingContext.start()`
- Wait for the processing to be stopped (manually or due to any error) using `streamingContext.awaitTermination()`
- The processing can be manually stopped using `streamingContext.stop()`

Stream Processing with Spark

Streaming Word Count Example

- Create a local StreamingContext with two threads and batch interval of 1 second

```
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
```

- Create a DStream that will connect to hostname:port, like localhost:9999

```
lines = ssc.socketTextStream("localhost", 9999)
```

- Split each line into words

```
words = lines.flatMap(lambda line: line.split(" "))
```

- Count each word in each batch

```
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)
```

- Print the first ten elements of each RDD generated in this DStream to the console

```
wordCounts.pprint()
```

- Start the computation and wait for the computation to terminate

```
ssc.start()
ssc.awaitTermination()
```

Hands-on

Streaming Word Count – Supplementary Material Requirements

1. Unix-like shell (Linux, Mac OS or Windows/Cygwin)
2. Python installed
3. Installation of Spark (see guide “Install Spark in Local Mode”)

Full examples can be found at:

<https://spark.apache.org/docs/latest/streaming-programming-guide.html>

Hands-on

Streaming Word Count – Supplementary Material

- Access to your Spark instance
- Copy `network_wordcount.py` example to your home directory

```
$ cp /usr/local/spark/examples/src/main/python/streaming/network_wordcount.py .
```

- Check its contents
- Change interval to 5 seconds
- Change logging level to `ERROR` by adding this attribute just after spark context creation

```
sc.setLogLevel("ERROR")
```

- Parallelize execution to use 2 threads

```
sc = SparkContext("local[2]", appName="PythonStreamingNetworkWordCount")
```

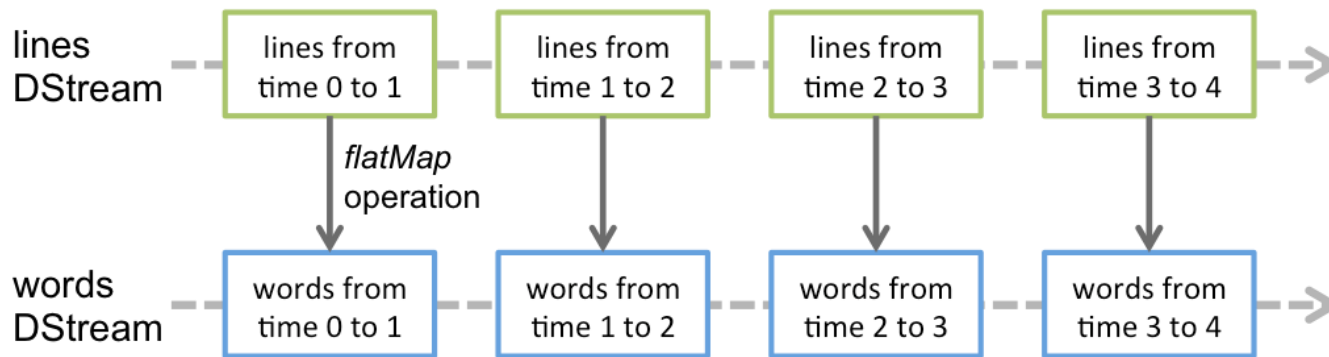
- When running a Spark Streaming program locally, do not use “local” or “local[1]” because these mean that only one thread will be used for running tasks locally.
- If you are using an input DStream based on a receiver (e.g. sockets), then the single thread will be used to run the receiver, leaving no thread for processing the received data.
- Hence, when running locally, always use “local[n]”, where $n >$ number of receivers to run

Hands-on

Streaming Word Count - Streaming Word Count – Supplementary Material

```
# TERMINAL 1: Running Netcat
```

```
$ nc -lk 9999  
hello world  
...
```



```
# TERMINAL 2: Running network_wordcount.py
```

```
$ spark-submit network_wordcount.py localhost 9999  
-----  
Time: 2014-10-14 15:25:21  
-----  
(hello,1)  
(world,1) ...
```

Stream Processing with Spark

DStreams Sources and Transformations

Discretized Stream (DStream)

- DStreams representing the stream of input data received from streaming sources
- Every input DStream (except file stream) is associated with a Receiver object which receives the data from a source and stores it in Spark's memory for processing

Types of DStream Sources

- Basic sources: Sources directly available in the StreamingContext API. Examples: file systems, and socket connections.
- Advanced sources: Sources like Kafka, Flume, Kinesis, etc. are available through extra utility classes.

DStream Transformations

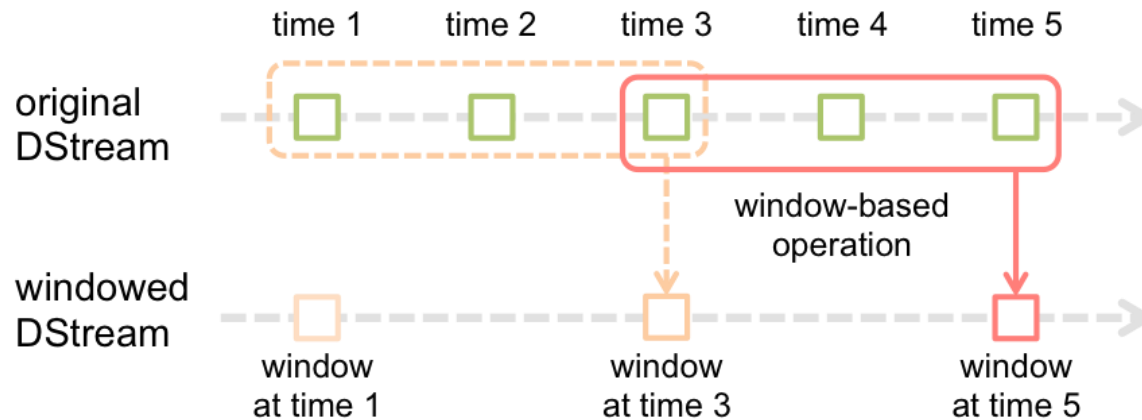
- Similar to that of RDDs, transformations allow the data from the input DStream to be modified.
- DStreams support many of the transformations available on normal Spark RDD's.

Stream Processing with Spark

Advanced Feature: Windows Operations

Windowed Computations

- Apply transformations over a sliding window of data



- Generating word counts over the last 30 seconds of data, every 10 seconds

```
windowedWordCounts = pairs.reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y:  
x - y, 30, 10)
```

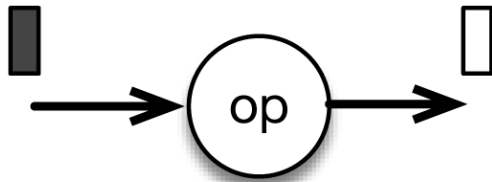
Stream Processing with Spark

Advanced Feature: Stateful Stream Processing

Computation Maintains Contextual State

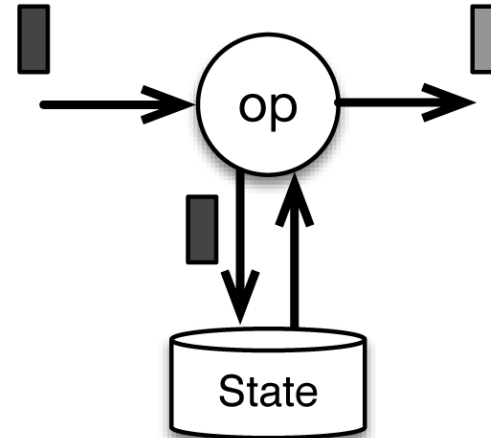
- This state is used to store information derived from the previously-seen events
- Virtually all non-trivial stream processing applications require stateful stream processing

Stateless stream processing



`updateStateByKey(func)`

Stateful stream processing



Streaming word count

Accumulated word count

Trend detection

Hands-on

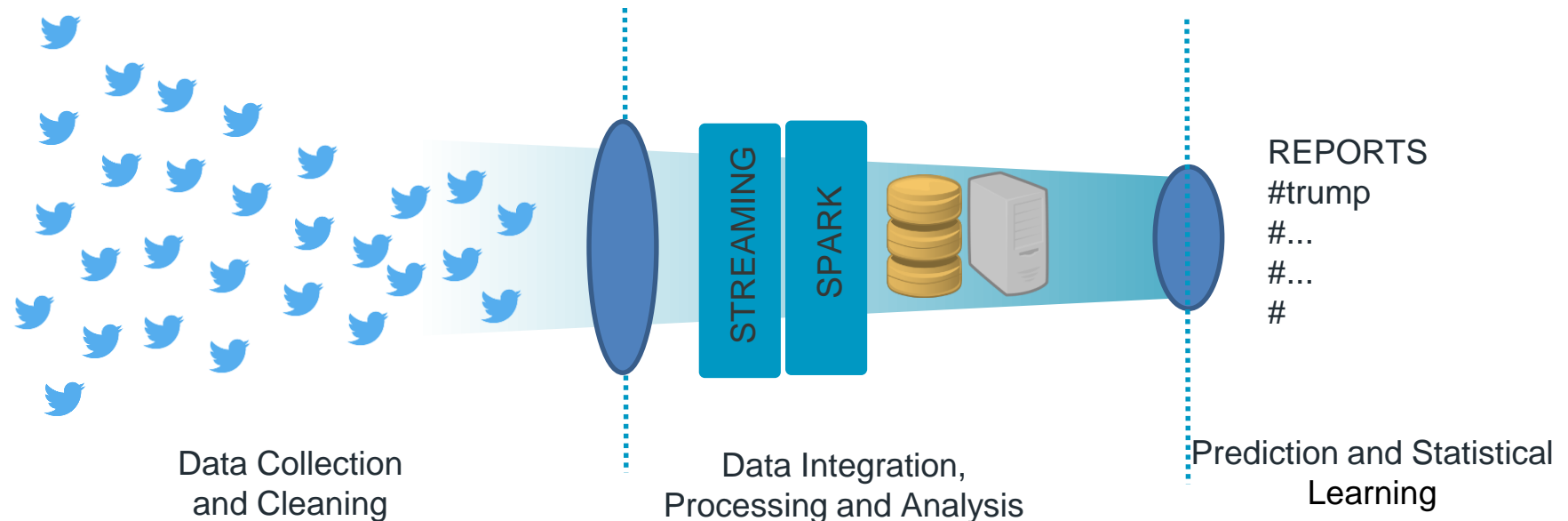
Trend Detection in Social Media

The Use Case

- Use Twitter to detect trends in real time in the Boston area

Architectural Components

- Twitter Client that collects and *cleans* tweets in the area
- Streaming Application that does real-time processing for the incoming tweets and shows hashtags in the interval and accumulated



Hands-on

Trend Detection in Social Media

`https://harvard-iacs.github.io/2020-CS205/lectures/C3/`

Hands-on

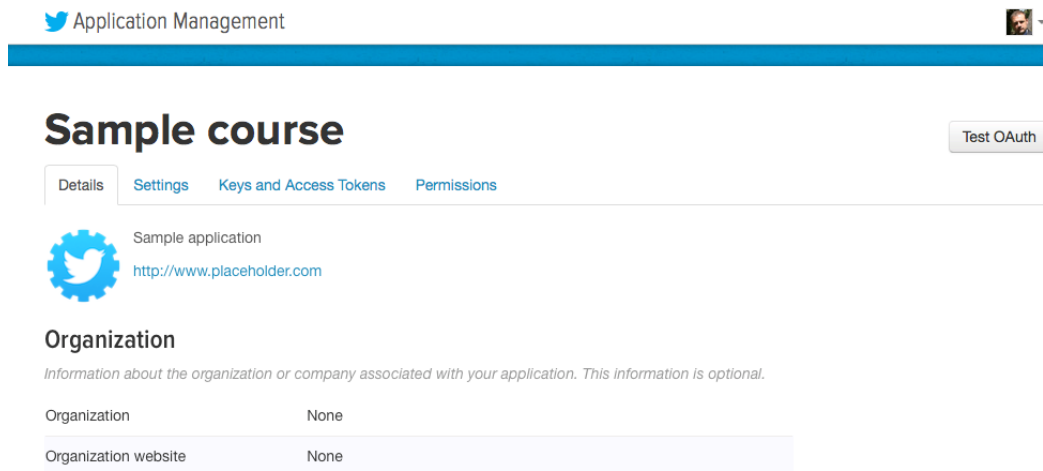
Trend Detection in Social Media

The Twitter Client

- Collect: Call the Twitter API URL and return a stream of tweets
- Clean: Extract the tweet text from the JSON structure
- Transmit: Send each tweet text through a TCP port to the application

Creation of Credential for Twitter App

- Go to <https://apps.twitter.com> and create new app
- Go to “Keys and Access Tokens” tab and then click on “Generate my access token.”



The screenshot shows the Twitter Application Management page for a "Sample course". The page has a blue header with the Twitter logo and "Application Management" text. Below the header, there is a "Sample course" title and a "Test OAuth" button. The main content area has four tabs: "Details", "Settings", "Keys and Access Tokens", and "Permissions". Under the "Keys and Access Tokens" tab, there is a "Sample application" section with a Twitter logo icon and the URL "http://www.placeholder.com". Below this is an "Organization" section with a note: "Information about the organization or company associated with your application. This information is optional." There are two rows of information: "Organization" with the value "None" and "Organization website" with the value "None".

Hands-on

Trend Detection in Social Media

The Streaming Application

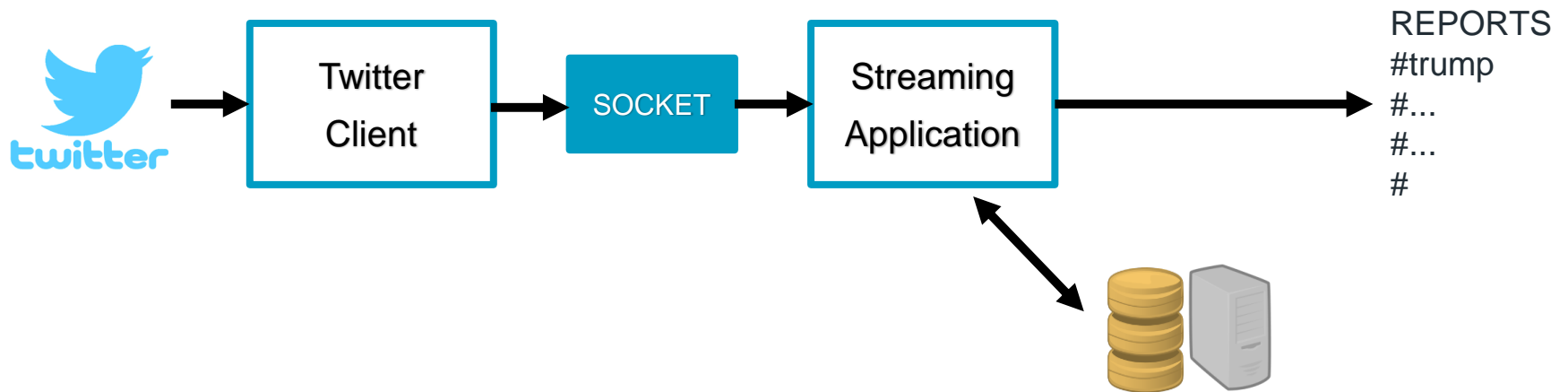
- Define Stream: Stream process with window size of 10 seconds
- Define Source: Socket port
- Stream Processing: Split DStream into words, filter to get words with hashtags, and reduce by key
- Interval Reporting: Trend in the interval
- Global State Update: Update global state
- Global State Reporting: Accumulated trend

Exercise

- Adapt code to know in real time the frequency of references to Harvard and the words used to describe it (sentiment analysis)

Stream Processing with Spark

Application Architecture



Next Steps

- HWC due on Monday 4/20!
 - Final Project (next milestones):
 - Team formation and tentative topic: 3/30
 - Project proposal (4/14 and 4/16)
 - Project design (4/21 and 4/23)
 - Project presentation (5/11)
- More info at:

<https://harvard-iacs.github.io/2020-CS205/>

Project Requirements

- Demonstrate the need for big compute and/or big data processing, and what can be achieved thanks to large-scale parallel processing.
- Solve a problem for a non-trivial computation graph and with hierarchical parallelism.
- Be implemented on a distributed-memory architecture with either a many-core or a multi-core compute node, and evaluated on at least 8 compute nodes (note: each compute node on Cannon is a multi-core with 32, or 64 cores or with a many-core GPU with hundreds of cores)
- Use a hybrid parallel program in either, for example: MPI + OpenMP, MPI + OpenACC (or OpenCL), Spark or MapReduce + OpenACC (or OpenCL) or MPI + Spark or MapReduce
- Be evaluated on large data sets or problem sizes to demonstrate both weak and strong scaling using appropriate metrics (throughput, efficiency, iso-efficiency...).

Project Proposal Presentation

You will have **5, and ONLY 5, minutes** to briefly summarize your proposal answering bellow questions. You have to prepare 4 slides for your proposal. We will enforce the 5-minute time limit.

What is the **problem** you are trying to solve with this application?

What is the **need for big compute and/or big data processing** and what can be achieved thanks to large-scale parallel processing?

Describe your **model and/or data** in detail: where does it come from, what does it mean, etc.

Which **tools and infrastructures** you are planning to use to build the application?

Zoom Presentation Guidelines

Appoint 1 group member to share their screen on Zoom.

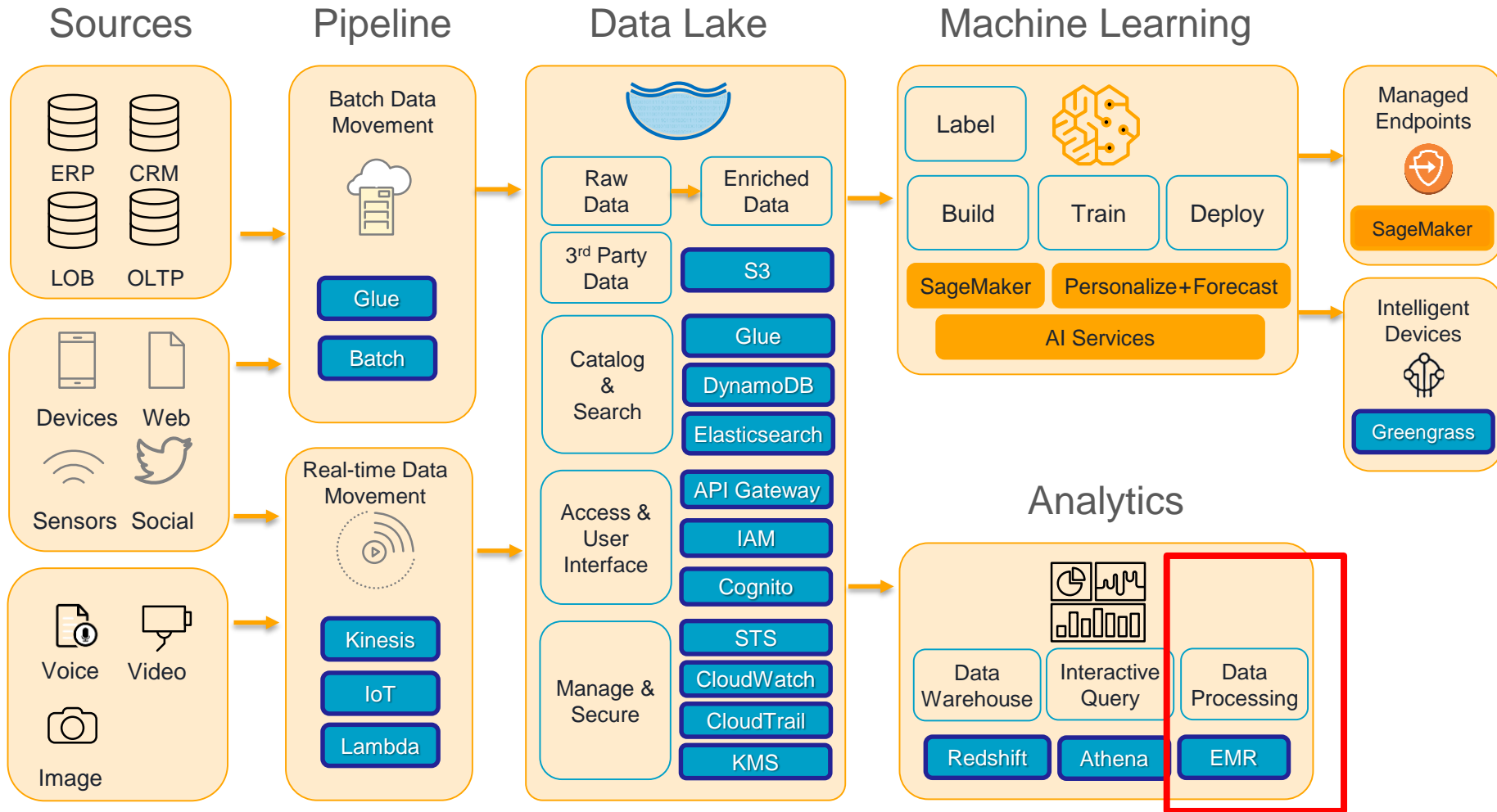
Each group member should present.

Practice ahead of time!

Make sure your mics are muted when you are not presenting.

Don't forget to stop sharing your screen when your group is done!

Bigger Picture



Questions

Stream Data Processing

<https://harvard-iacs.github.io/2020-CS205/>

