CS 398 ACC
Streaming

Prof. Robert J. Brunner
Ben Congdon
Tyler Kim
How’s it going?

Final Autograder run:
  - Tonight ~9pm
  - Tomorrow ~3pm

● Due tomorrow at 11:59 pm.
● Latest Commit to the repo at the time will be graded.
● Last Office Hours today after the lecture until 7pm.
Outline

- Streaming Overview
- Spark Streaming
- Spark Streaming Programming
- Final Project Announcement
Outline

- Streaming Overview
- Spark Streaming
- Spark Streaming Programming
- Final Project Announcement
Streaming

- Why streaming?
  - Lots of data is not fixed, in practice
  - We have new data coming in all the time; it makes sense to respond in real time

- What is streaming?
  - Clients push “events” in real time to an interface with our streaming system
  - The streaming system distributes input across the cluster (like batch processing)
  - Some resultant data is generated, which can be saved or streamed out of the system
## Batch Processing vs Stream Processing

<table>
<thead>
<tr>
<th></th>
<th>Batch Processing</th>
<th>Stream Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Size</strong></td>
<td>Large batches of data; Most data in a data set</td>
<td>“An event”; Micro-batches of records</td>
</tr>
<tr>
<td><strong>Nominal Latency</strong></td>
<td>Minutes to Hours</td>
<td>In the order of milliseconds</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>Complex Algorithm/Analytics</td>
<td>Simple functions, aggregation, rolling metrics</td>
</tr>
</tbody>
</table>
Application of Streaming Data

- Real-Time Machine Learning
  - e.g. Twitter’s “trending” topics, disaster monitoring, etc.

- Tracking changes in the finance markets in real-time

- Processing sensor data in large industrial settings
  - (Also scientific settings)
Stream Sources

- File System
- Internet of Things
- Network Traffic
- Embedded devices on a radio frequency
Apache Storm - A pure streaming system

- Purpose-built for real-time stream computation

- Three main concepts:
  - "Stream": An unbounded sequence of tuples (like an infinite RDD)
  - "Spout": A source of tuples
  - "Bolt": A transformation operation on a stream
Apache Storm - A pure streaming system

- Spouts, Bolts, and Streams define a “Topology”
Apache Storm

● Powerful language-agnostic tool/framework

● Open-sourced by Twitter
  ○ Used to power Twitter’s real-time tweet analytics
  ○ Now Twitter uses Heron

● Handles fault tolerance
  ○ Keeps track of which tuples have been fully processed
  ○ If a “Bolt” fails, unprocessed / partially processed tuples are reprocessed
Outline

- Streaming Overview
- **Spark Streaming**
- Spark Streaming Programming
- Final Project Announcement
Spark Streaming

- Kafka
- Flume
- HDFS/S3
- Kinesis
- Twitter

- HDFS
- Databases
- Dashboards
Spark Looks a lot like Apache Storm...
Spark Looks a lot like Apache Storm...

Spark Streaming

Data Source A

Transformation

Transformation

Transformation

Data Source B

Transformation

Action

Output
How Spark Handles Streaming

- Spark Core has a robust way for creating computation graphs on batch data
- How can we extend this to streaming data?
How Spark Handles Streaming

input data stream → Spark Streaming → Spark Engine → batches of processed data

batches of input data
How Spark Handles Streaming
How Spark Handles Streaming

DStream

Stream Source

RDD (t=1)

Transformation

RDD (t=1)

Transformation

RDD (t=1)

Transformation

DStream

DStream

DStream

Action

Output
How Spark Handles Streaming

Stream Source → DStream → Transformation → DStream → Transformation → DStream → Transformation → DStream → Output

- RDD (t=1)
- RDD (t=2)
How Spark Handles Streaming

Stream Source

Transformation

RDD (t=3)

Transformation

RDD (t=3)

Transformation

RDD (t=3)

RDD (t=2)

Transformation

RDD (t=2)

RDD (t=3)

Action

RDD (t+1)

Output
How Spark Handles Streaming

- **DStream**
  - “Discretized Stream” which represents an infinite stream of data
  - In actuality, it’s a (endless) sequence of RDDs

- **Spark Streaming Context**
  - Similar to the Spark Context, except it handles DStreams
  - Has a user-defined batch interval
    - Defines the window size for RDDs

- **Job Processing**
  - Executed in multiples of the batch interval
Spark Stream Sources

- **Built-in Data Sources**
  - File Stream - Load new files in a given directory
  - Socket Stream - Listen on a TCP connection for new data

- **Additional Supported Stream Sources**
  - Kafka (Apache)
  - Flume (Apache, in the Hadoop ecosystem)
  - Kinesis (AWS)
Outline

- Streaming Overview
- Spark Streaming
- **Spark Streaming Programming**
- Final Project Announcement
Spark Streaming Programming

- **Transformations**
  - Operations on DStreams (mostly) identical to RDDs
  - Functions: Map, Filter, Join, etc.

- **Windowed Operations**
  - Applied transformation on a time-based window of data
  - Functions: CountByWindow, ReduceByWindow, etc.
Spark Streaming Programming

- **Stateful Operations**
  - So far, we’re limited to “seeing” data within our current window
  - What if we need arbitrary state?
  - Functions: “UpdateStateByKey”
    - Uses an RDD to keep a persistent state

- **The “Transform” Function**
  - Allows you get access to the underlying DStream RDD
  - Used for “combining” DStream data with arbitrary RDDs
    - i.e. Join streamed data on precomputed data
Spark Streaming Programming Example

```python
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# local streaming context with two threads
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)  # batch interval of 1 second

# DStream that pulls from localhost:8888
lines = ssc.socketTextStream("localhost", 8888)
```
Spark Streaming Programming Example

# We can use the lines DStream almost like a normal RDD
filtered = lines.filter(lambda l: 'cloud' in l).flatMap(lambda x: x.split())
key_on_word = filtered.map(lambda w: (w, 1))

# Count in window lengths of 30 seconds, evaluated every 10 seconds
windowed = key_on_word.countByValueAndWindow(30, 10)

# We can call an action on DStreams like RDDs
windowed.pprint()

# Start stream processing
ssc.start()
MP4 - Spark Streaming

- Will be released Tonight.
- Due next Tuesday at 11:59pm (as normal)
Outline

- Streaming Overview
- Spark Streaming
- Spark Streaming Programming
- Final Project Announcement
CS398 Final Project

- **Goal:** Perform exploratory analysis on a large dataset using cloud technologies

- **High Level Overview:**
  - Choose a dataset; Perform some type of novel analysis; Document and present your findings

- **Components:**
  - Group Selection
  - Project Proposal
  - Project Presentation
  - Project Report
  - Peer Evaluation
CS398 Final Project

- **Group Selection**
  - Project groups must have 3-4 students
  - You may select your own groups
    - We will also open the Piazza group selection forum
  - Group selection will be “locked in” by submitting a form on **March 10th at 11:59pm**
- **March 9th - Drop Deadline**
CS398 Final Project

- **Dataset Selection**
  - Groups are free to pick datasets given:
    - Dataset size is between 3GB - 500GB
    - You have the rights to use it for educational purposes
  - Groups will be given cluster or S3 space to place their dataset
    - Groups are responsible for loading the dataset into S3 / the cluster
  - Final Project page lists many example datasets
    - i.e. Government data, AWS public data, etc.

  - Due in your Project Proposal
CS398 Final Project

- **Project Proposal**
  - Addresses:
    - What dataset you’ll be using
    - What technologies / frameworks you’ll be using
    - Hypotheses about the data that you plan to test
    - Briefly defend the utility / novelty of your planned work
  - Should be detailed enough to have an idea of what you’ll be working on
  - You may deviate from your proposal as you work
  - **Due March 16th at 11:59pm**
CS398 Final Project

- Project Work
  - All work on projects will be independent of the class
    - MPs will continue to be released
  - Groups are encouraged to use the course cluster
  - Groups may request additional software / resources
    - Requests will be evaluated by course staff
CS398 Final Project

- **Project Report**
  - The “final product” of your work
  - Should address:
    - What datasets / frameworks you ultimately used
    - What results / insights / knowledge you recovered from the data
    - What issues did you encounter during the project; how did you resolve them
  - All application code must be submitted
  - Should include a brief performance report
  - **Due May 2nd at 11:59pm** (Day before Reading Day)
CS398 Final Project

● Project Report Grading
  ○ Usage of Cloud Computing (40%)
    ■ Did your project make adequate use of the CC technologies discussed in class?
  ○ Treatment of Dataset (20%)
    ■ Did your project make logical use of the dataset you chose?
  ○ Application Novelty (20%)
    ■ Did your group attempt to do something new and interesting? (Not WordCount™ 2.0)
  ○ Formatting (10%)
    ■ Is your report coherent? Does it contain all components?
  ○ Code Submission (10%)
    ■ Did you submit all the code necessary to replicate your results?
CS398 Final Project

- **Project Presentation**
  - Conducted during the last 2 weeks of lecture
    - April 23, April 25, April 30
  - Present high-level findings of your project in 8-10 minutes
    - What dataset did you use?
    - What technologies did you use?
    - What were your results? What applications do your findings have?
CS398 Final Project

● Peer Evaluation

  ○ Evaluation of Group Members
    ▪ You will evaluate the contributions of your group members
    ▪ You will be graded (in part) by the evaluations of your group members

  ○ Evaluation of Other Groups
    ▪ You must attend *at least* 2 of the 3 project presentation days
    ▪ You will evaluate the presentations of the other groups that present on those days
    ▪ You will be graded (in part) by the evaluations of your peers
CS398 Final Project

- Grading Overview:
  - Group Selection: 5%
  - Project Proposal: 10%
  - Project Report: 45%
  - Project Presentation: 20%
  - Peer Evaluation: 10% + 10%
CS398 Final Project

● Deadline Overview:
  ○ **Group Selection:** March 10th (Week 8)
  ○ **Project Proposal:** March 16th (Week 9, Right before Spring Break)
  ○ **Project Report:** May 2nd (Week 16, Right before Reading Day)
  ○ **Project Presentation:** During scheduled lecture times, Weeks 15-16
  ○ **Peer Evaluation**
    ■ Evaluation of peers: During scheduled lecture times, Weeks 15-16
    ■ Group member evaluation: Due right before Reading Day