Frequently asked questions from the previous class survey

- Streaming:
  - Significance of minimum delay
  - Interleaving of packets in live streaming
  - Who handles spacing in the buffer
  - Dealing with major packet loss
  - Buffering is primarily for syncing up audio and video

- Publish subscribe
  - Difference between AMQP and JMS
  - Wire-frames vs. interfaces

- Popular type based pub/sub?

- Dealing with major packet losses?

- Buffering is primarily for syncing up audio and video

- Can message queuing handle multiple recipients?

- Circuit-switched networks

Topics covered in this lecture

- Spark
  - Software stack
  - Interactive shells in Spark
  - Core Spark concepts
  - Resilient Distributed Datasets

Gossips: Variant of the epidemic scheme

- Also called rumor spreading
- If P has been updated for data item x
- Finds Q and tries to update it
- If Q was already updated:
  - P may lose interest in spreading the update with \( p = \frac{1}{k} \)

Gossiping: An excellent way to spread news

- But cannot guarantee that all nodes will actually be updated
- When a large number of nodes participate in epidemics:
  - A fraction of users (s) can miss updates
  - \( s = e^{-(k+1)(1-s)} \)
Managing Web Content Delivery

Akamai

- Websites redirect users to Akamaized URLs
- IP address associated with client used to select server-farm closest to client.
- Most popular content served up from caches
  - Benefits of caching and network proximity
- Server farms sync up with managed websites to track content changes.

Apache Spark

Spark: What is it?

- Cluster computing platform
  - Designed to be fast and general purpose
- Speed
  - Extends MapReduce to support more types of computations
    - Interactive queries, iterative tasks, and stream processing
- Why is speed important?
  - Difference between waiting for hours versus exploring data interactively

Key enabling idea in Spark

- Memory resident data
  - Spark loads data into the memory of worker nodes
  - Processing is performed on memory-resident data

A look at the memory hierarchy

<table>
<thead>
<tr>
<th>Item</th>
<th>Time</th>
<th>Sealed time in human terms (2 billion times slower)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor cycle</td>
<td>0.5 ns (2 GHz)</td>
<td>1 second</td>
</tr>
<tr>
<td>Cache access</td>
<td>1 ns (1 GHz)</td>
<td>2 seconds</td>
</tr>
<tr>
<td>Memory access</td>
<td>70 ns</td>
<td>140 seconds</td>
</tr>
<tr>
<td>Context switch</td>
<td>5,000 ns (5 μs)</td>
<td>167 minutes</td>
</tr>
<tr>
<td>Disk access</td>
<td>7,000,000 ns (7 ms)</td>
<td>162 days</td>
</tr>
<tr>
<td>Quantum</td>
<td>100,000,000,000 ns (100 ms)</td>
<td>6.3 years</td>
</tr>
</tbody>
</table>


Spark covers a wide range of workloads

- Batch applications
- Iterative algorithms
- Queries
- Stream processing
- This has previously required multiple, independent tools
APIs

- Java, Python, Scala, and SQL
- Integrates well with other tools
  - Can run in Hadoop clusters
  - Access Hadoop data sources, including Cassandra

At its core, Spark is a computational engine

- Spark is responsible for several aspects of applications that comprise
  - Many tasks across many machines (compute clusters)
- Responsibilities includes:
  1. Scheduling
  2. Distributions
  3. Monitoring

The Spark stack

**THE SPARK SOFTWARE STACK**

- Spark SQL: structured data
- Spark Streaming: real-time
- MLlib: machine learning
- GraphX: Graph processing

Spark Core

- Basic functionality of Spark
- Task scheduling, memory management, fault recovery, and interacting with storage systems
- Also, the API that defines Resilient Distributed Datasets (RDDs)
  - Spark’s main programming abstraction
  - Represents collection of data items dispersed across many compute nodes
  - Can be manipulated concurrently (parallel)

Spark SQL

- Package for working with structured data
- Allows querying data using SQL and HQL (Hive Query Language)
  - Data sources: Hive tables, Parquet, and JSON
- Allows intermixing queries with programmatic data manipulations support by RDDs
  - Using Scala, Java, and Python
Spark Streaming
- Enables processing of live streams of data
- Sources
  - Logfiles generated by production webservers
  - Messages containing web service status updates

MLib
- Library that contains common machine learning functionality
- Algorithms include:
  - Clustering, classification, regression, clustering, and collaborative filtering
  - Low-level primitives
  - Generic gradient descent optimization algorithm
- Alternatives?
  - Mahout, sci–kit learn, VW, WEKA, and R among others

Graph X
- Library for manipulating graphs
- Graph-parallel computations
- Extends Spark RDD API
  - Create a directed graph, with arbitrary properties attached to each vertex and edge

Cluster Managers
- Spark runs over a variety of cluster managers
- These include:
  - Hadoop YARN
  - Apache Mesos
  - Standalone Scheduler
  - Included within Spark

Storage Layers for Spark
- Spark can create distributed datasets from any file stored in HDFS
- Plus, other storage systems supported by the Hadoop API
  - Amazon S3, Cassandra, Hive, HBase, etc.

Interactive Shells in Spark
Spark Shells

- Interactive [Python and Scala]
  - Similar to shells like Bash or Windows command prompt
- Ad hoc data analysis
- Traditional shells manipulate data using disk and memory on a single machine
  - Spark shells allow interaction with data that is distributed across many machines
  - Spark manages complexity of distributing processing

Several software were designed to run on the Java Virtual Machine

- Languages that compile to run on the JVM and can interact with Java software packages but are not actually Java
- There are a number of non-Java JVM languages
  - The two most popular ones used in real-time application development: Scala and Clojure

Scala

- Has spent most of its life as an academic language
  - Still largely developed at universities
  - Has a rich standard library that has made it appealing to developers of high-performance server applications
- Like Java, Scala is a strongly typed object-oriented language
  - Includes many features from functional programming languages that are not in standard Java
  - Interestingly, Java 8 incorporate several of the more useful features of Scala and other functional languages.

What is functional programming?

- When a method is compiled by Java, it is converted to instructions called byte code and ...
  - Then largely disappears from the Java environment
    - Except when it is called by other methods
- In a functional language, functions are treated the same way as data
  - Can be stored in objects similar to integers or strings, returned from functions, and passed to other functions

What about Clojure?

- Based on Lisp
- Javascript?
  - Name was a marketing gimmick
  - Closer to Clojure and Scala than it is to Java
Core Spark Concepts

- Drivers
- SparkContext
- Executors

Drivers

- Every Spark application consists of a driver program
- Driver launches various parallel operations on the cluster
- Constituent elements
  - Application's main function
  - Defines distributed datasets on the clusters
  - Applies operations to these datasets

SparkContext

- Driver programs access Spark through a SparkContext object
- Represents a connection to a computing cluster
- Within the shell?
  - Created as the variable sc
    - You can even print out sc to see the the type
- Once you have a SparkContext, you can use it to build RDDs
  - And then run operations on the data ...

Executors

- Driver programs manage a number of nodes, called executors
- Executors are responsible for running operations
- For example:
  - If we were running a count() operation on cluster
  - Different machines might count lines in different ranges of the file

Components for distributed execution in Spark

Lot of Spark’s API revolves around passing functions to its operators

```python
def hasPython(line):
    return "Python" in line

pythonLines = lines.filter(hasPython)
```

```
pythonLines = lines.filter(line => line.contains("Python"))
```

Also known as the lambda or => syntax
Lot of Spark’s API revolves around passing functions to its operators

```java
JavaRDD<String> pythonLines = lines.filter(
    new Function<String, Boolean>() {
      Boolean call(String line) {
        return line.contains("Python");
      }
    })
;
```

```java
JavaRDD<String> pythonLines = lines.filter(line => line.contains("Python") );
```

The contents of this slide-set are based on the following references

- Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data. Byron Ellis. Wiley. [Chapter 2]