Other Big Data Tools

Big Data Analysis with Scala and Spark

Heather Miller

What we've learned, and what there is to learn...

Now you've learned Spark.

What else is out there?

What we've learned, and what there is to learn...

Now you've learned Spark.

What else is out there?

Spark is just a drop in the bucket. See:

- Big-Data Ecosystem Table
- Apache "big data" projects
- Amazon Web Services in Plain English

What else is out there?

We can't hope to really see a significant fraction of these in a semester (or even a degree). What follows is a summary of some more.

Proposed take-away messages:

- There are tools for big data outside of the Spark/Hadoop universe
- The big data world is evolving extremely fast
- There are many things out there, and you should know them enough to think "I'm pretty sure I heard about a tool for that already..."

Let's look at some categories of large-scale parallel processing tools

Spark is one example of **batch** processing. There is also **stream** processing.

Let's look at the kinds of applications out there in big Data

Big-Data Ecosystem Table. Apache "big data" projects. Amazon Web Services in Plain English.

Making Computations Happen

The goal: get the compute work to some computer with processor/memory to do it, and get the results back.

Resource managers:

- Apache YARN: Hadoop's resource manager.
- Mesos: resource manager more closely aligned with Spark.

Making Computations Happen

Resources to compute on:

- Amazon EC2: get VMs to do work with.
- Amazon EMR: EC2 + Hadoop set up automatically.
- Google Compute Engine: get VMs from Google.
- Amazon Lambda: give Amazon functions; it runs them when you call.
- Google AppEngine: give Google functions; it runs them when you call.

Expressing Computation

The goal: Describe your computation in a way that can be run on a cluster.

- MapReduce.
- Spark.
- Flink: competitor to Spark. Scala/Java. Streaming first.
- Pig: high-level language to analyze data. Produces MapReduce jobs.

Data Warehousing

The goal: Take lots of data from many sources, and do reporting/analysis on it.

- Spark SQL.
- Hive: take varied data and do SQL-like queries on it.
- Apache Impala: massively-parallel SQL queries on Hadoop, against varied inputs.
- Apache Drill: take data from many sources (SQL, NoSQL, files, ...) and query it.
- Google BigQuery: Google's data warehouse tool.
- Amazon RedShift: Amazon's data warehouse tool.

Storing Files

The goal: Store files or file-like things in a distributed way.

- HDFS.
- Amazon S3: Amazon's file storage.
- Gluster: distributed network filesystem.
- Alluxio: (formerly known as Tachyon) in-memory distributed storage system.

Databases

The goal: store records and access/update them quickly. I don't need SQL/relations.

- Cassandra: Good clustering. Secondary keys, but no joins.
- HBase: Good clustering, fast. Otherwise very manual.
- MongoDB. Clustered, but questionable reliability. **
- Amazon SimpleDB and DynamoDB.
- Amazon Aurora: Amazon's scalable relational database.

Serialization/Storage

The goal: read/write data efficiently for memory/disk/network.

- Parquet: efficient columnar storage representation. Supported by Spark, Pandas (new), Impala.
- HDF5: on-disk storage for columnar data.
- CSV, JSON: well-understood interchange formats.
- Arrow: in-memory representation for fast processing. Available in Spark 2.3+.

Streaming

The goal: deal with a continuously-arriving stream of data.

- Spark Streaming (DStreams, Streaming DataFrames).
- Apache Storm.
- Apache Flume.
- Amazon Kinesis.

ML Libraries

The goal: use machine learning algorithms (at scale) without having to implement them.

- Spark MLlib.
- Apache Mahout.
- Amazon Machine Learning.

Visualization

The goal: take the data you worked so hard to get, and let people understand it and interact with it.

- Tableau.
- Qlik.
- Power BI.

Extract-Transform-Load

The goal: Extract data from the source(s); transform it into the format we want; load it into the database/data warehouse.

- Apache Sqoop.
- Amazon Data Pipeline.
- MapReduce, Spark, programming.

Message Queues

The goal: pass messages between nodes/processes and have somebody else worry about reliability, queues, who will send/receive, etc.

- Apache Kafka.
- RabbitMQ.
- ZeroMQ/ØMQ.
- Amazon SQS.

All designed to scale out and handle high volume.

Message Queues

The idea:

- Some nodes publish messages into a queue.
- The message queue makes sure that they are queued until they can be processed; ensures each message is processed once.
- Some nodes subscribe to the queue(s) and consume messages.

Or other interactions with the queues. Freely switch languages between publisher/consumer too.

Message Queues

These things are fast: RabbitMQ Hits One Million Messages Per Second. For comparison,

- Tweets max spike: 143k/sec.
- Tweets average: >6k/sec.
- Google search average: 35k/sec.

Realistic streaming scenario with message queues

Spark streaming takes in the data stream, filters/processes minimally, and puts each record into a queue for more processing. Then many nodes subscribe to the queue and handle the data out of it.

Or without Hadoop, just generate a bunch of work that needs to be done, queue it all, then start consumer processes on each computer you have.

Either way: you can move around the bottleneck (and hopefully then fix it).

Task Queues

The goal: get some work on a distributed queue. Maybe wait for results, or maybe don't.

- Celery (Python).
- Resque, Sidekiq (Ruby).
- Google AppEngine Task Queues (Python, Java, Go, PHP).
- Amazon Simple Workflow Service.
- Any message queue + some code.

Task Queues

With a task queue, you get to just call a function (maybe with slightly different syntax). You can then retrieve the result (or just move on an let the work happen later).

Where the work happened is transparent.

Task Queues

Need a lot of work done without Hadoop? Run task queue workers on many nodes; make all the asynchronous calls you want; let the workers handle it.

Need nightly batch analysis done? Have a scheduled task start a Spark task.

Have a spike in usage? Let tasks queue up and process as possible.

Text Search

The goal: index lots of data so you (or your users) can search for records they want.

- Apache Solr/Apache Lucene.
- Elasticsearch.
- Amazon CloudSearch.