SPARK SUMMIT 2017

Scaling Data Science Capabilities with Spark at Stitch Fix

Derek Bennett

Overview

- About us
- Our usage of Spark
- Details of our Spark Architecture
- Challenges and Ongoing Work

About Stitch Fix

- Personalized online retail company based in San Francisco, CA USA
- Stylists create shipments (“fixes”) based on algorithmic recommendations and requests
- Customer pays for what they keep and returns the rest, along with feedback on checkout
- “Empowering people to find what they love”

About The Algorithms Team

- Over 80 people dedicated to analytics and data science capabilities
  - Three vertically focused teams of data scientists
  - Data platform team providing infrastructure, services and tools to all vertical teams
- Data scientists own their pipelines
  - Platform team has a self service charter
  - “Engineers shouldn’t write ETL”

Our Usage of Spark

How we Leverage Spark

- Heavy use of Spark SQL along with `DataFrame` APIs
- Hive Metastore and Spark `HiveContext`
- Mostly PySpark with some Scala
- Some initial use of Spark ML

What’s different about us

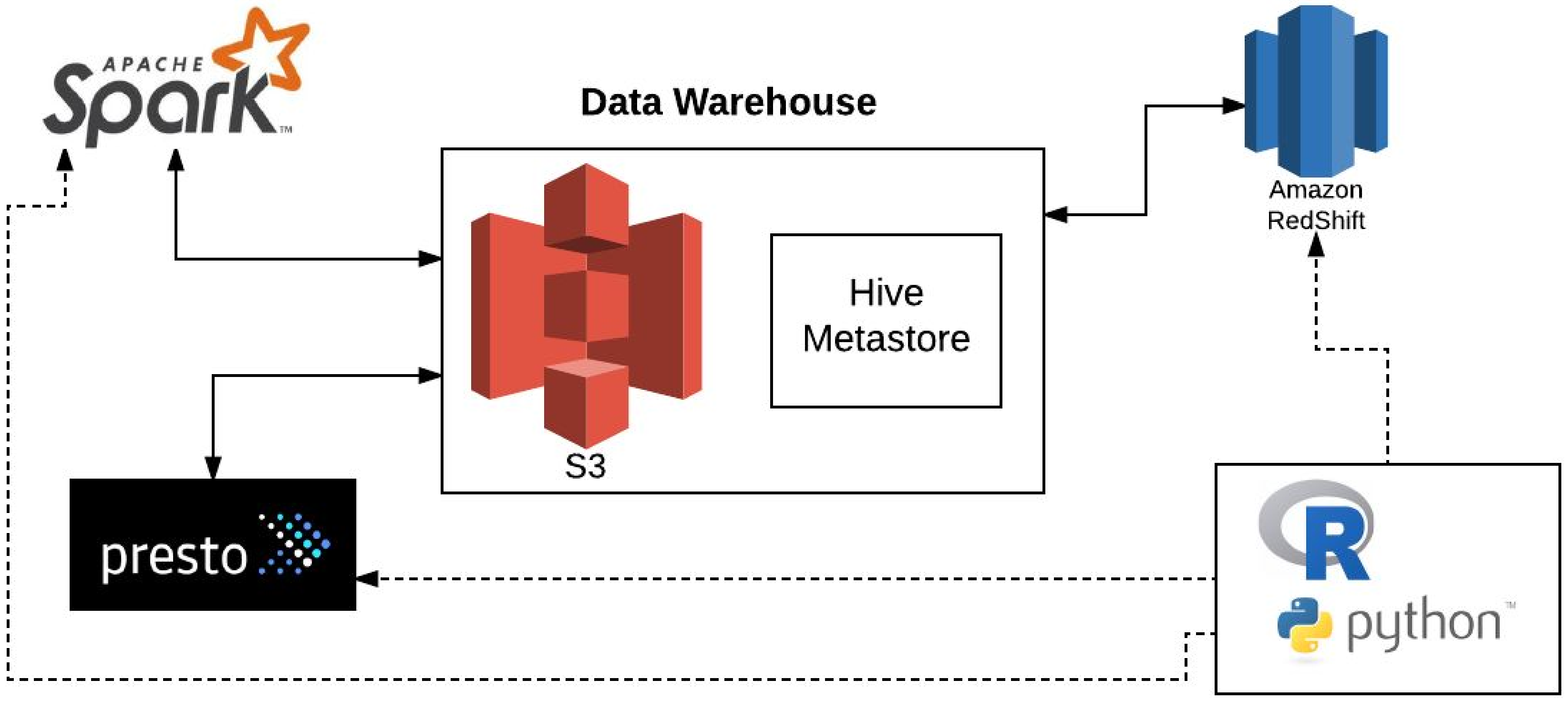
- Vast majority of data is in structured tables
- Instead of a small number of large jobs, large number of diverse and varying-size jobs
- Scalability is by capability not amount of data
- Many people used to SQL, so Spark SQL is an easy way to transition in

More Interesting Use Cases

- Analyzing website visitors
  - Several queries mixed with use of Python API
  - Reuse of existing python code in spark job
- Client “states” : chains of multiple queries
- Examples using 3rd party NLP libraries
- Analyzing client segments with loop of multiple similar jobs
- Machine Learning : feature selection example

Our Spark Architecture

Data Architecture



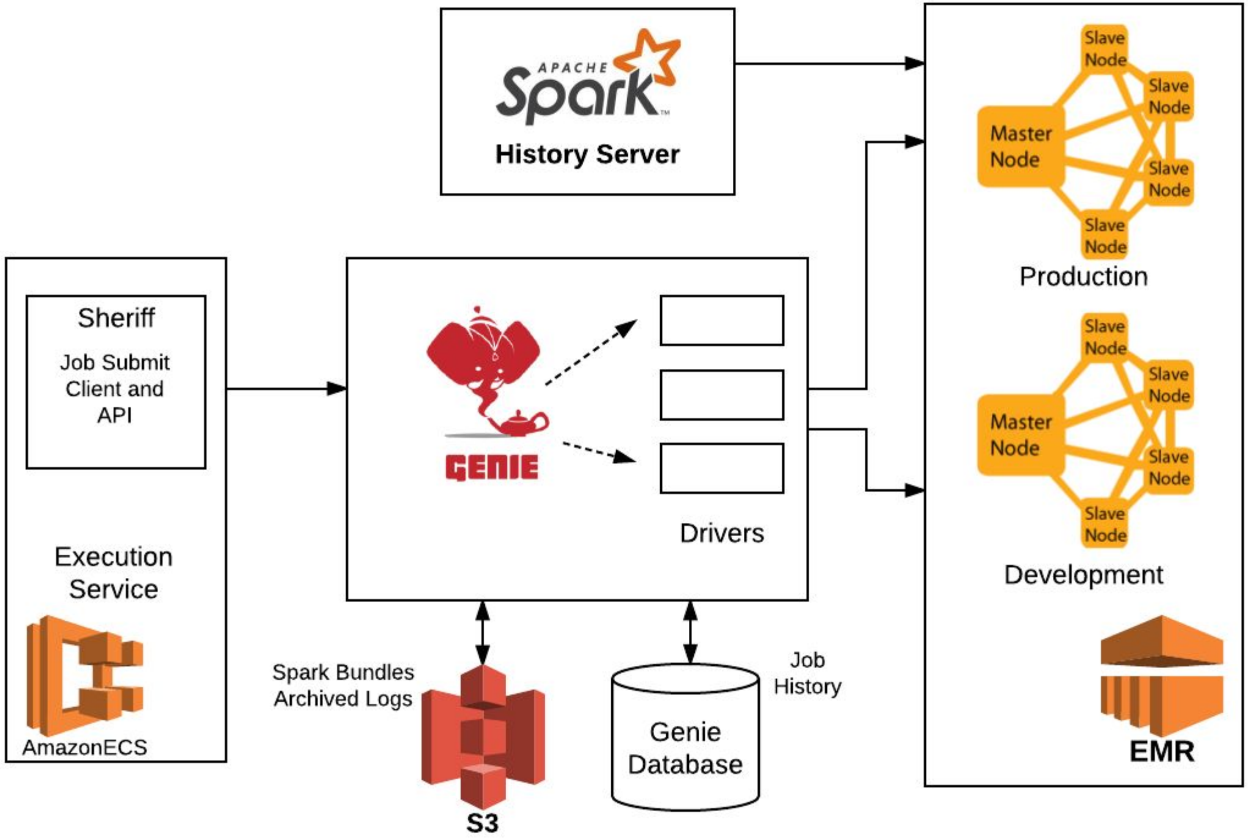
Architecture Notes

- Amazon S3
  - Read, Write, Delete (no append/update)
  - Data ingested from databases and log events
- Hive Metastore
  - Schema and partitions
  - Data location of most recent data
- Spark primarily works with data on S3
- Additional query engines: Presto and Redshift

Architecture Principles

- S3 as the source of truth
- Batch Id pattern
  - Hidden inner directory - timestamp based
  - Overwrite : write new batch and update location
- Hive metastore manages schemas and latest data location
- No data permanently kept in HDFS

Spark Setup in Detail



Spark Setup in Detail

- Genie service and its benefits:
  - Execution service isolating EMR details
  - Administration and supporting different versions of Spark and multiple clusters
- EMR and our clusters: dev / prod / ad-hoc
- Clusters are transient - no data in HDFS
- Jupyter IPython notebooks and related tools
- EMR File System (more efficient S3 interaction)

Spark Builds and Deployments

- Supported versions: 1.6.x, 2.0.x, 2.1.x
- Maintaining our own fork of Spark
- Snapshot builds for testing
- Additional “applications” and “commands” in genie to experiment with different settings

Additional Libraries and Utilities

- SFS3 : Output and Validation
  - Writing while enforcing batch id paradigm
  - Utilize Amazon EMRFS for efficient read/write
- Hive utilities
  - Allow table creation or adding partitions
  - Table analyzing
- Tracer : library interface to time series database
- Diagnosis script to find root cause of failures

Challenges and Ongoing Work

Challenges

- Porting monolithic queries and using the API
  - Education that Spark is more than just SQL!
- Education on use of Spark and various settings
  - Find useful defaults and override when needed
  - Provide reference for starting tuning
- Challenging errors related to joins, metastore, memory usage, and contention
- Contention in the cluster; scheduling; priorities

Ongoing Projects

- Users & YARN queue settings
- Centralized history server
- Data reader/writer project
- More extensive use of different file formats
- Logging integration
- Future use of streaming