Beyond Hadoop with Apache Spark and BDAS

Khanderao Kand
Principal Technologist, Guavus
12 April GITPRO World 2014
Palo Alto, CA

Credit:
Some statistics and content came from presentations from publicly shared spark.apache.org and AmpLabs presentations.
Apache Spark

Apache Spark™ is a fast and general engine for large-scale data processing.

- Fast
- Easy to use
- Support mix of interaction models
Big Data Systems Today

- MapReduce
- Pregel
- Dremel
- Giraph
- GraphLab
- Impala
- Tez
- Drill
- Samza
- Storm
- S4

Different Models: Batch, Iterative, Interactive, and Streaming
Apache Spark

Batch

One Framework!

Interactive

Streaming

Three Programming models in one unified framework:

Batch-  Streaming-  Interactive
What is Spark?

- Open Source from Amplabs: UC Berkely
- Fast Interactive Big Data Processing engine
- Not a modified version of Hadoop
- Compatible with Hadoop’s storage APIs
  - Can read/write to any Hadoop-supported system, including HDFS, HBase, SequenceFiles, etc
- Separate, fast, MapReduce-like engine
  - In-memory data storage for very fast iterative queries
  - General execution graphs and powerful optimizations
  - Up to 40x faster than Hadoop for iterative cases
- faster data sharing across parallel jobs as required for multi-stage and interactive apps require
- Efficient Fault Tolerant Model
Spark History

• AmpLabs from UC Berkely
• Spark started as research project in 2009
• Open sourced in 2010
• Growing community since
• Entered Apache Incubator in June 2013 and now 1.0 is coming out
• Commercial Backing: Databricks
• Enhancements by current stacks: Cloudera, MapR
• Additional Contributions: Yahoo, Conviva, Oooyala, ...

Contributions

• AmpLabs - Databricks
• YARN support (Yahoo!)
• Columnar compression in Shark (Yahoo!)
• Fair scheduling (Intel)
• Metrics reporting (Intel, Quantifind)
• New RDD operators (Bizo, ClearStory)
• Scala 2.10 support (Imaginea)
How does Data Sharing work in Hadoop Map Reduce Jobs?

Result: Hadoop Map Reduce is Slow:
1. Replication
2. Serialization
3. Disk IO
Spark faster due to in-memory
Data Sharing across jobs

In some cases, $10-100\times$ faster than network and disk
Spark is Faster and Easier

• Extremely fast scheduling
  – Ms-seconds in Spark vs secs-mins-hrs in Hadoop MR

• Many more useful primitives
  – Higher level APIs for more than map-reduce-filter
  – Broadcast and accumulators variables
  – Combining programming models
Result:
Spark vs Hadoop Performance

PageRank Performance

- Hadoop: 171
- Basic Spark: 72
- Spark + Controlled Partitioning: 23
Apache Spark

- Unifies **batch, streaming, interactive** comp.
- Easy to build sophisticated applications
  - Support iterative, graph-parallel algorithms
  - Powerful APIs in Scala, Python, Java

- Batch, Interactive
- Mesos / YARN
- HDFS
- Spark Core – RDD, MR
- Spark Streaming
- Interactive
  - BlinkD
  - Shark SQL
- Graph
  - GraphX
- Machine Learning
  - MLlib
- Streaming

Components:
- Spark Streaming
- Shark SQL
- GraphX
- MLlib
RDD

• Read only Parallelized Fault tolerant Data Set
• From :
  – Files,
  – Transformation,
  – Parallelizing a collection
  – From other RDD
• Spark tracks lineage that is used for recreation upon failure
• Late-realization
RDD Fault Tolerance

RDDs track Lineage

Tracks the transformations used to build them (their *lineage*) to recompute lost data

E.g.: messages = textFile(...).filter(lambda s: s.contains("Spark")).map(lambda s: s.split("\'t\")[2])
Richer Programming Constructs

• Not just map, reduce, filter

• But
  • map, reduce,
  • flatMap, filter, count, reduce,
  • groupByKey, reduceByKey, sortByKey,
  • join
Shared Variables

• Two types:
  – Broadcast
  – Accumulate

• Broadcast:
  – Lookup type shared dataset
  – Large (can go 10s-100s GB)

• Accumulators
  – Additives
  – Like counters in reduce
Scala Shell : Interactive Big Data Programming

scala> val file = sc.textFile("mytext.data")
scala> val words = file.map( line => line.split(" "))
scala> words.map(word=>(word, 1)).reduceByKey(_+_))
Spark Streaming

Micro-batches: a series of very small, deterministic batch jobs

- Microbatch is small (seconds) batches of the live stream
- Microbatches are treated as RDD
- Same RDD Programming model and operations
- Results are returned as batches
- Recovery based on lineage & Checkpoint

live data stream

batches of X seconds

Spark Streaming

Spark
Comparison with Spark and S4

Higher throughput than Storm

– Spark Streaming: 670k records/second/node
– Storm: 115k records/second/node
– Apache S4: 7.5k records/second/node

Ref: AmpLabs : Tathagata Das
DStream Data Sources

• Many sources out of the box
  – HDFS
  – Kafka
  – Flume
  – Twitter
  – TCP sockets
  – Akka actor
  – ZeroMQ

• Extensible
Transformations

Build new streams from existing streams

– RDD-like operations
  • map, flatMap, filter, count, reduce,
  • groupByKey, reduceByKey, sortByKey, join
  • etc.

– New window and stateful operations
  • window, countByWindow, reduceByWindow
  • countByValueAndWindow, reduceByKeyAndWindow
  • updateStateByKey
  • etc.
Sample Program:
Find trending top 10 hashtags in last 10 min

// Create the stream of tweets
val tweets = ssc.twitterStream(<username>, <password>)

// Count the tags over a 10 minute window
val tagCounts = tweetsflatMap (statuts => getTags(status))
  .countByValueAndWindow (Minutes(10), Second(1))

// Sort the tags by counts
val sortedTags = tagCounts.map { case (tag, count) => (count, tag) }
  .transform(_.sortByKey(false))

// Show the top 10 tags
sortedTags.foreach(showTopTags(10) _)
Storm + Fusion Convergence – Twitter Model
Shark

- High-Speed In-Memory Analytics over Hadoop (HDFS) and Hive Data
- Common HiveQL provides seamless federation between Hive and Shark
- Sits on top of existing Hive warehouse data
- Direct query access from UI like tableau
- Direct ODBC/JDBC from desktop BI tools
Shark

• Analytic query engine compatible with Hive
  – Hive QL,
  – UDFs,
  – SerDes,
  – scripts, types

• Makes Hive queries run much faster
  – Builds on top of Spark
  – Caching data
  – Optimizations
Shark Internals

• A faster execution engine than Hadoop Map Reduce
• Optimized Storage format
  – Columnar memory store
• Various other optimizations
  – Fully distributed sort
  – data co-partitioning
  – partition pruning
Shark Performance Comparison

SELECT pageURL, pageRank FROM rankings WHERE pageRank > X

Query 1A
32,888 results

Query 1B
3,331,851 results

Query 1C
89,974,976 results
Performance

Throughput (MB/s/node)

- **Streaming**
  - Storm
  - Spark

Response Time (s)

- **SQL**
  - Hive
  - Impala (disk)
  - Impala (mem)
  - Shark (disk)
  - Shark (mem)

Response Time (min)

- **Graph**
  - Giraph
  - GraphX

- **Hadoop**
MLBase Inventory

• Classifications
  – NaiveBayes
  – SVM
  – SVMWithSGD
• Clustering
  – Kmeans
• Regression
  – Linear
  – Lasso
  – RidgeRegressionWithSGD
• Optimization:
  – GradientDiscent
  – LogisticGradient
• Recommendation
  – CollaborativeFiltering / ALS
• Others
  – MatrixFactorization