How Conviva used Spark to Speed Up Video Analytics by 30x

Dilip Antony Joseph (@DilipAntony)
Conviva monitors and optimizes online video for premium content providers
What happens if you don't use Conviva?
65 million video streams a day
Conviva data processing architecture

- Video Player
- Live data processing
- Hadoop for historical data
- Spark
- Reports
- Ad-hoc analysis
- Monitoring Dashboard
Group By queries dominate our Reporting workload

```
SELECT videoName, COUNT(1)
FROM summaries
WHERE date='2012_08_22' AND customer='XYZ'
GROUP BY videoName;
```

10s of metrics, 10s of group bys
Group By query in Spark

```scala
val sessions = sparkContext.sequenceFile[SessionSummary, NullWritable](
    pathToSessionSummaryOnHdfs,
    classOf[SessionSummary], classOf[NullWritable])
  .flatMap {
    case (key, val) => val.fieldsOfInterest
  }

val cachedSessions = sessions.filter(
    whereConditionToFilterSessionsForTheDesiredDay)
  .cache
val mapFn : SessionSummary => (String, Long) = { s => (s.videoName, 1) }

val reduceFn : (Long, Long) => Long = { (a,b) => a+b }

val results = cachedSessions.map(mapFn).reduceByKey(reduceFn).collectAsMap
```
Spark is 30x faster than Hive

45 minutes versus 24 hours
for weekly Conviva Geo Report
How much data are we talking about?

- 150 GB / week of compressed summary data
- Compressed ~ 3:1
- Stored as Hadoop Sequence Files
- Custom binary serialization format
Spark is faster because it avoids reading data from disk multiple times
Hive

Group By Country
Read from HDFS
Decompress
Deserialize

Group By State
Read from HDFS
Decompress
Deserialize

Group By Video
Read from HDFS
Decompress
Deserialize

10s of Group Bys ...

Spark

Group By Country
Read from HDFS
Decompress
Deserialize
Cache data in memory

Group By State
Read data from memory

Group By Video
Read data from memory

10s of Group Bys ...

Hive/MapReduce startup overhead

Overhead of flushing intermediate data to disk

Cache only columns of interest
Why not <some other big data system>?

- Hive
- Mysql
- Oracle/SAP HANA
- Column oriented dbs
Spark just worked for us

- 30x speed-up
- Great fit for our report computation model
- Open-source
- Flexible
- Scalable
30% of our reports use Spark
We are working on more projects that use Spark

- Streaming Spark for unifying batch and live computation

- SHadoop – Run existing Hadoop jobs on Spark
Problems with Spark

- Video Player
- Live data processing
- Hadoop for historical data
- Spark
- Monitoring Dashboard
- Reports
- Ad-hoc analysis

No Logo
Hadoop for historical data
Live data processing
Spark
Reports
Ad-hoc analysis
Hadoop
Spark
Video Player
Monitoring Dashboard
Hive
Spark queries are not very succinct

```
SELECT videoName, COUNT(1)
FROM summaries
WHERE date='2012_08_22' AND customer='XYZ'
GROUP BY videoName;
```

```
val sessions = sparkContext.sequenceFile[SessionSummary, NullWritable](
    pathToSessionSummaryOnHdfs,
    classOf[SessionSummary], classOf[NullWritable])
  .flatMap {
    case (key, val) => val.fieldsOfInterest
  }

val cachedSessions = sessions.filter(
    whereConditionToFilterSessionsForTheDesiredDay)
  .cache
val mapFn : SessionSummary => (String, Long) = { s => (s.videoName, 1) }
val reduceFn : (Long, Long) => Long = { (a,b) => a+b }

val results = cachedSessions.map(mapFn).reduceByKey(reduceFn).collectAsMap
```
There is a learning curve associated with Scala, but ...

- Type-safety offered by Scala is a great boon
  - Code completion via Eclipse Scala plugin

- Complex queries are easier in Scala than in Hive
  - Cascading IF()s in Hive

- No need to write Scala in the future
  - Shark
  - Java, Python APIs
Additional Hiccups

• Always on the bleeding edge – getting dependencies right

• More maintenance/debugging tools required
Spark has been working great for Conviva for over a year
We are Hiring

jobs@conviva.com

http://www.conviva.com/blog/engineering/
using-spark-and-hive-to-process-bigdata-at-
conviva