From MapReduce to Spark with Apache Crunch

Micah Whitacre
@mkwhit
Invested in learning
Invested in **learning**

**Setup** production clusters
Invested in learning

Setup production clusters

Tuned everything
Current Strategy

1. Build MR Jobs as needed
2. ????
3. Profit
We should switch to Spark
Umm what would it take to switch?
Learn Spark’s API and processing patterns, ...
Refactor all our code, ...
Experiment with how to tune it all again, ...
We don’t use plain MR it won’t be that bad...
● Crunch ([http://crunch.apache.org/user-guide.html#sparkpipeline](http://crunch.apache.org/user-guide.html#sparkpipeline))
● Summingbird (will [https://github.com/twitter/summingbird/issues/387](https://github.com/twitter/summingbird/issues/387))
How Spark is Known..
In Memory

How Spark is Known..
In Memory

100x Faster than MapReduce

How Spark is Known..

SQL, streaming, and complex analytics
A fast and general engine for large-scale data processing.
Spark has an advanced *Directed Acyclic Graph* execution engine that supports cyclic data flow and in-memory computing.
Spark has an advanced Directed Acyclic Graph execution engine that supports cyclic data flow and in-memory computing.
RDD
RDD

Resilient Distributed Dataset
Locality Aware Scheduling
Locality Aware Scheduling

Scalability
Locality Aware Scheduling

Scalability    Fault Tolerant
Locality Aware Scheduling

Scalability          Fault Tolerant

Applications with working sets
(Parallel ops on intermediate results)
Locality Aware Scheduling

Scalability  Fault Tolerant

Applications with working sets
(Parallel ops on intermediate results)
Options?

Distributed
Shared Memory
+ Checkpointing

Log Updates
Options?

Distributed
Shared Memory
+ Checkpointing

Log Updates
Log (coarse-grained) Updates
Immutable/Read Only

Partitioned

**Bad** for async updates to shared state
RDDs lifecycle in memory tied to Spark Application
Transformations

Actions
Transformations
map, filter, flatmap, union, groupByKey, sample

Actions
reduce, collect, count, take
Transformations

lazily executed

Actions

return values to driver
val sc = new SparkContext(new SparkConf())
val charCounts = sc.textFile(args(0))
  .flatMap(_.split(" "))
  .flatMap(_.toCharArray).map((_, 1))
charCounts.collect()
// (‘a’, 1)(‘a’, 1)(‘b’, 1)(‘c’, 1)(‘e’, 1)
Apache Crunch Review
Process Reference Data

Process Raw Data using Reference

Filter Out Invalid Data

Group Data By Person

Create Person Record

Avro

CSV

CSV
Pipeline

CSV

Process Reference Data

Process Raw Data using Reference

Filter Out Invalid Data

Group Data By Person

Create Person Record

Avro

CSV

Process Raw Person Data
Pipeline $p = \ldots$
Sources

- Process Reference Data
- Process Raw Data using Reference
- Process Raw Person Data

Pipeline

- Filter Out Invalid Data
- Group Data By Person
- Create Person Record

Target
PCollection<String> values = p.read(source);
...
values.write(target);
Process Reference Data

Process Raw Data using Reference

Filter Out Invalid Data

Group Data By Person

Create Person Record

CSV

PCollection Pipeline

Avro
PCollection<String> values = ...
PTable<String, Integer> counts = values.parallelDo(fn, ptype);
MRPipeline

DoFn

Join

FilterFn

Group By

Key

MapFn

DoFn
Pipeline \( p = \) new MRPipeline(
    Driver.class, hadoopConfig);
PCollection<String> values = 
   p.read(...);
<do processing>
   p.write(...);
   p.done();
MRPipeline

DoFn → Join → FilterFn → Group By Key → MapFn

Map → Reduce → Reduce
Here’s what we need to do to switch...
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.10</artifactId>
  <version>${sparkVersion}</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>org.apache.crunch</groupId>
  <artifactId>crunch-spark</artifactId>
  <version>${crunchVersion}</version>
  <scope>compile</scope>
</dependency>
Pipeline p = new MRPipeline(
    Driver.class, hadoopConfig);
Pipeline \( p = \)

```
new SparkPipeline(
  "spark://localhost:7077",
  "Spark App Name"
);
```
hadoop jar myjar.jar com.example.Driver ...
spark-submit

--class com.example.Driver

--master spark://localhost:7077

...
That’s not too bad...
Well there are some differences to account for...
Crunch with MRPipeline minimizes I/O

Crunch with SparkPipeline defers planning to Spark
MRPipeline  SparkPipeline
MRPipeline  SparkPipeline

Supports multiple writes
MRPipeline

Supports multiple writes

SparkPipeline

Performs multiple writes in same task/stage
MRPipeline

Supports multiple writes

Performs multiple writes in same task/stage

SparkPipeline

Serial writes in separate task/stages
SparkPipeline

Job 1

DoFn

Job 2

DoFn

FilterFn

Group By Key

MapFn
SparkPipeline

Stage 1

DoFn

Stage 1

DoFn → FilterFn → Group By Key → MapFn

Job 2

Job 3
DoFn → Compute Expensive DoFn → FilterFn

DoFn
Spark is lazy

Action needed for something to happen
DoFn

Compute Expensive DoFn

FilterFn

Job 1
Job 2

DoFn

Compute Expensive DoFn

FilterFn

DoFn
DoFn

Compute Expensive DoFn

FilterFn

DoFn
Limit expensive computations

Keep RDDs around for reuse
Spark supports persisting RDDs in memory

rdd.persist()
rdd.persist(
    StorageLevel.MEMORY_ONLY)

    DISK_ONLY, DISK_ONLY_2,
    MEMORY_AND_DISK, MEMORY_AND_DISK_2,
    MEMORY_AND_DISK_SER,
    MEMORY_AND_DISK_SER_2, MEMORY_ONLY,
    MEMORY_ONLY_2, MEMORY_ONLY_SER,
    MEMORY_ONLY_SER_2, NONE, OFF_HEAP
PCollection<String> values =
    //expensive computation
values.cache();
PCollection<String> values =
    //expensive computation
CacheOptions opts = new CacheOptions.Builder()
    .useDisk(true).useMemory(true)
    .build();
values.cache(opts);
Spark needs to be able to serialize data

Send data between workers

Persist data in memory or disk
Spark supported serialization

Java Serializable
(and Externalizable)

Kyro Serialization
Spark recommends Kryo

Extra config on the SparkConfig

Custom serializer registration
Spark on Crunch

Hides serialization behind PTypes

Handles complex records like Avro
Spark on Crunch

Hides serialization behind **PTypes**

Handles complex records like Avro
Additional Topics to Explore

Aggregation sort behaviors

Reusing Crunch Functions in Spark
With Crunch, we’ll be able to ...
minimize significant code refactoring,
shorten learning curve by reusing concepts and API already used to...
incrementally switch from Spark,
overall experiment to find where Spark fits best.
Links:

- [http://spark.apache.org/docs/latest/](http://spark.apache.org/docs/latest/)
- Examples: [https://github.com/mkwhitacre/simplesparkapp](https://github.com/mkwhitacre/simplesparkapp)
Special Thanks...

- **Josh Wills** - helped come up with content
- **Sean Owen & Sandy Ryza** whose repo I forked to build examples and experiment