Faster ETL Workflows using Apache Pig & Spark

- Praveen Rachabattuni, Sigmoid Analytics

@praveenr019

About me

- Apache Pig committer and Pig on Spark project lead.

OUR CUSTOMERS

CAPILLARY™
Intelligent Customer Engagement

fusionops

PubNub®

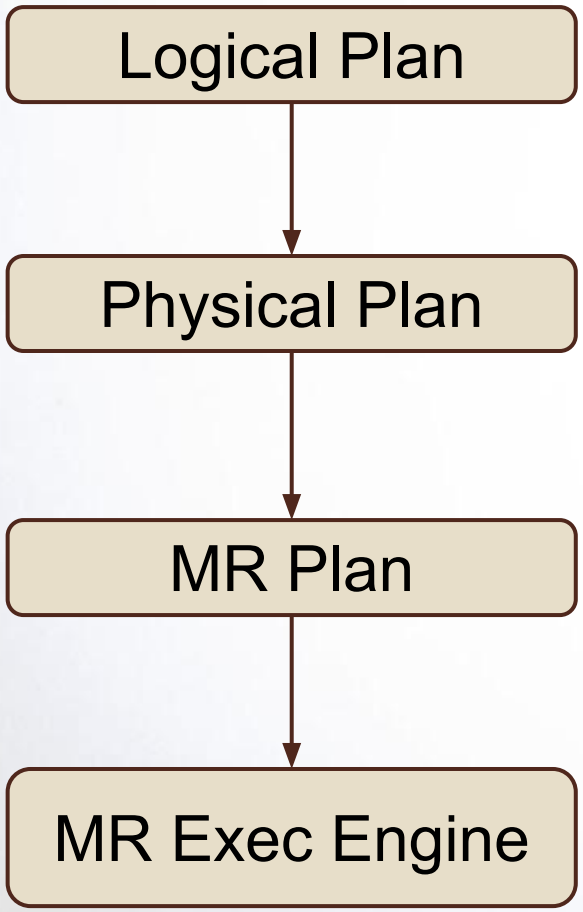
Why pig on spark ?

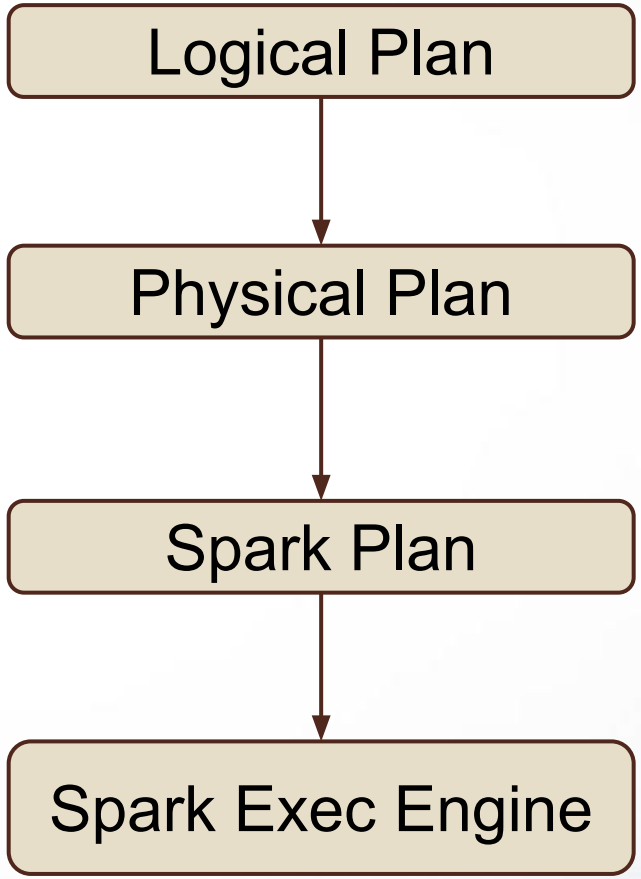
- Spark shell (scala), Spark SQL, Dataframes API
- Large mapreduce clusters running Pig on mapreduce jobs
- Much familiar language with developers/analysts and easier to debug

What steered us to Pig ?

- Targeted users
  - Analysts
  - Large pig script codebase projects
  - Cost saving for organisations in training new frameworks
- Rich operator library

How Spark plugs into Pig ?





Operator Mapping

Pig Operator	Spark Operator
Load	newAPIHadoopFile
Store	saveAsNewAPIHadoopFile
Filter	filter transformation
GroupBy	groupby & map
Join	CoGroup
ForEach	mapPartitions
Sort	sortByKey + map

Simple script


```
A = LOAD './wiki' USING PigStorage(' ') as (hour:chararray, pcode:
chararray, pagename:chararray, pageviews:chararray, pagebytes:
chararray);
B = FILTER A BY pageviews >= 50000;
DUMP B;
```

Input data:

```
en Main_Page 242332 4737756101
ak Italy 400 73160
en Main_Page 242332 4737756101
```

Load operator


```
    @Override
public RDD<Tuple> convert(List<RDD<Tuple>> predecessorRdds, POLoad poLoad)
      throws IOException {
  JobConf loadJobConf = SparkUtil.newJobConf(pigContext);
  configureLoader(physicalPlan, poLoad, loadJobConf);

  RDD<Tuple2<Text, Tuple>> hadoopRDD = sparkContext.newAPIHadoopFile(
        poLoad.getLFile().getFileName(), PigInputFormatSpark.class,
        Text.class, Tuple.class, loadJobConf);

  // map to get just RDD<Tuple>
  return hadoopRDD.map(TO_TUPLE_FUNCTION,
        SparkUtil.getManifest(Tuple.class));
}
```

Load operator (cont..)


```
private static class ToTupleFunction extends
        AbstractFunction1<Tuple2<Text, Tuple>, Tuple> implements
        Function1<Tuple2<Text, Tuple>, Tuple>, Serializable {

    @Override
    public Tuple apply(Tuple2<Text, Tuple> v1) {
        return v1._2();
    }
}
```

Filter operator


```
@Override
public RDD<Tuple> convert(List<RDD<Tuple>> predecessors,
        POFilter physicalOperator) {
    SparkUtil.assertPredecessorSize(predecessors, physicalOperator, 1);
    RDD<Tuple> rdd = predecessors.get(0);
    FilterFunction filterFunction = new FilterFunction(physicalOperator);
    return rdd.filter(filterFunction);
}
```

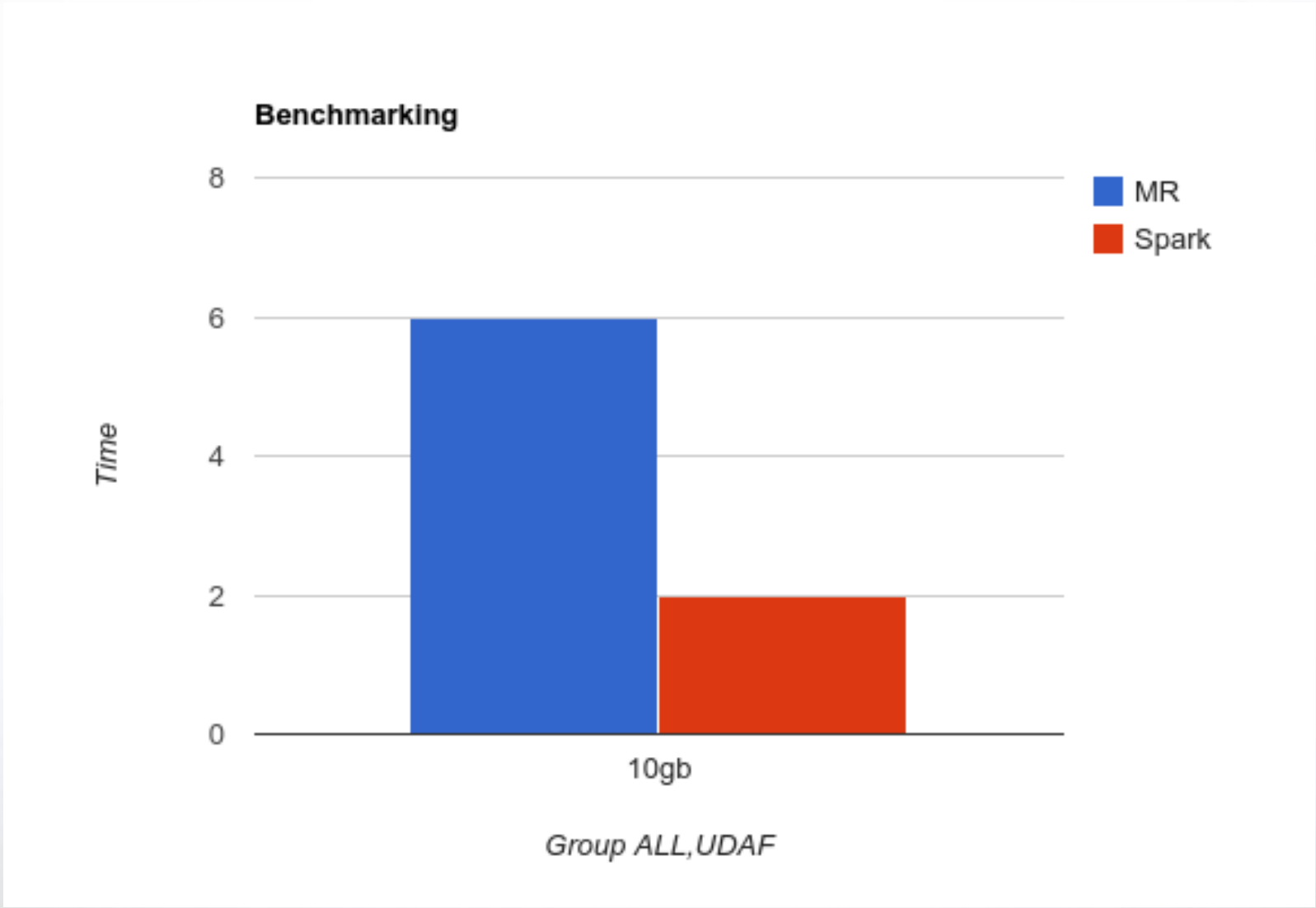
Filter operator (cont..)


```
private static class FilterFunction extends
        AbstractFunction1<Tuple, Object> implements Serializable {
    private POFilter poFilter;
    @Override
    public Boolean apply(Tuple v1) {
        Result result;
        try {
            poFilter.setInputs(null);
            poFilter.attachInput(v1);
            result = poFilter.getNextTuple();
        } catch (ExecException e) {
            throw new RuntimeException("Couldn't filter tuple", e);
        }
}
```

Spark plan

- MR Plan is structured towards mapreduce execution engine.
- Spark plan contains a sequence of transformations and more optimized towards Spark.
- Handover logical plan to Spark for much optimized flow, as Spark is pretty good at doing this.

Benchmark



Setting up Pig on Spark

1. Get the code
   a. git clone https://github.com/apache/pig -b spark
2. Building the project
   a. ant -Dhadoopversion=23 jar (assumes hadoop-2.x setup)
3. Env variables
   a. export HADOOP_USER_CLASSPATH_FIRST="true”
   b. export SPARK_MASTER=”local”
4. Start pig grunt shell
   a. bin/pig -x spark

Issues

- Spark plan to stand inline with Spark APIs
- Performance
- Functional parity with Pig on mapreduce

Contributors

- Mayur Rustagi (Sigmoid)
- Praveen Rachabattuni (Sigmoid)
- Liyun Zhang (Intel)
- Mohit Sabharwal (Cloudera)
- Xuefu Zhang (Cloudera)

References

- Apache Pig github mirror
  - https://github.com/apache/pig/tree/spark
- Umbrella jira for Pig on Spark
  - https://issues.apache.org/jira/browse/PIG-4059

Thank you

Queries ??