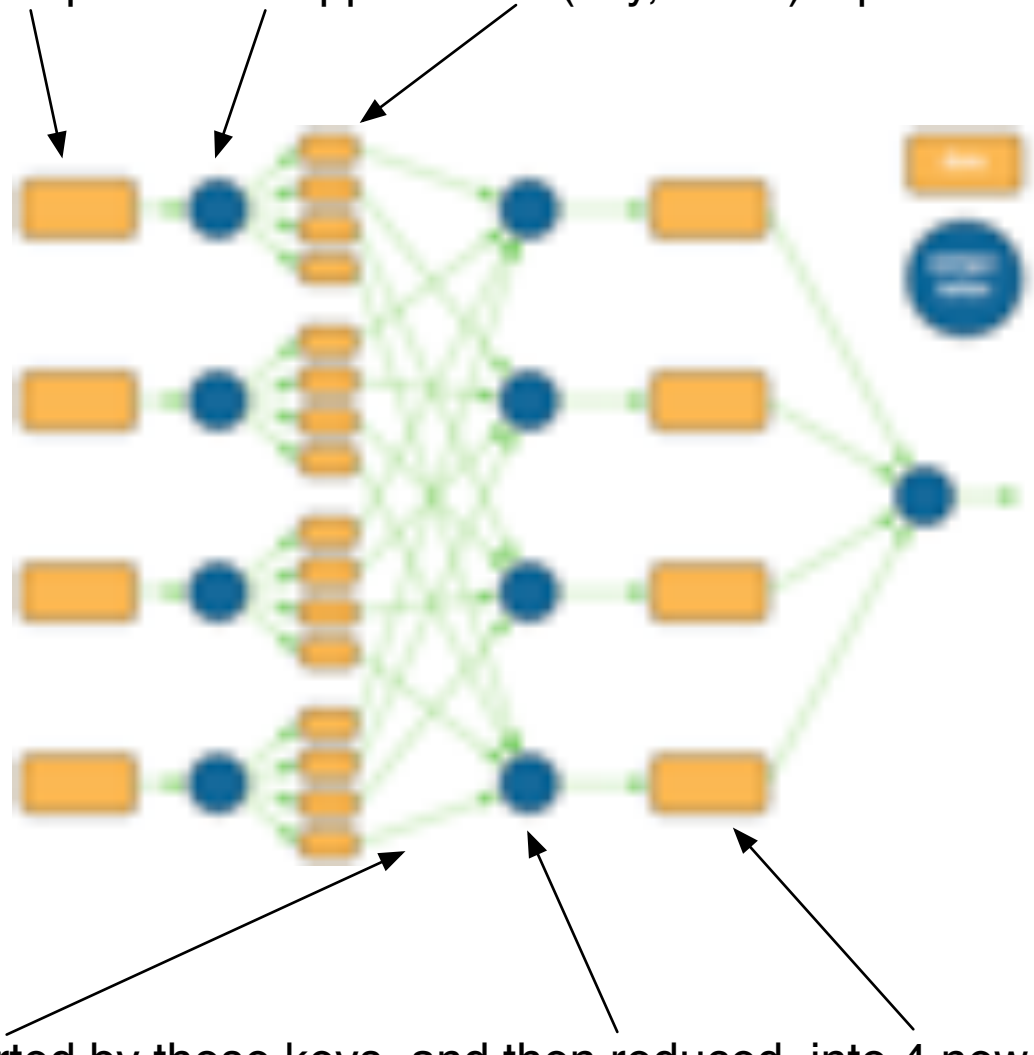
Getting started with Hadoop, Hive, and Elastic MapReduce

Hunter Blanks, hblanks@monetate.com / @tildehblanks
github.com/hblanks/talks/

PhillyAWS. January 12, 2012

Overview

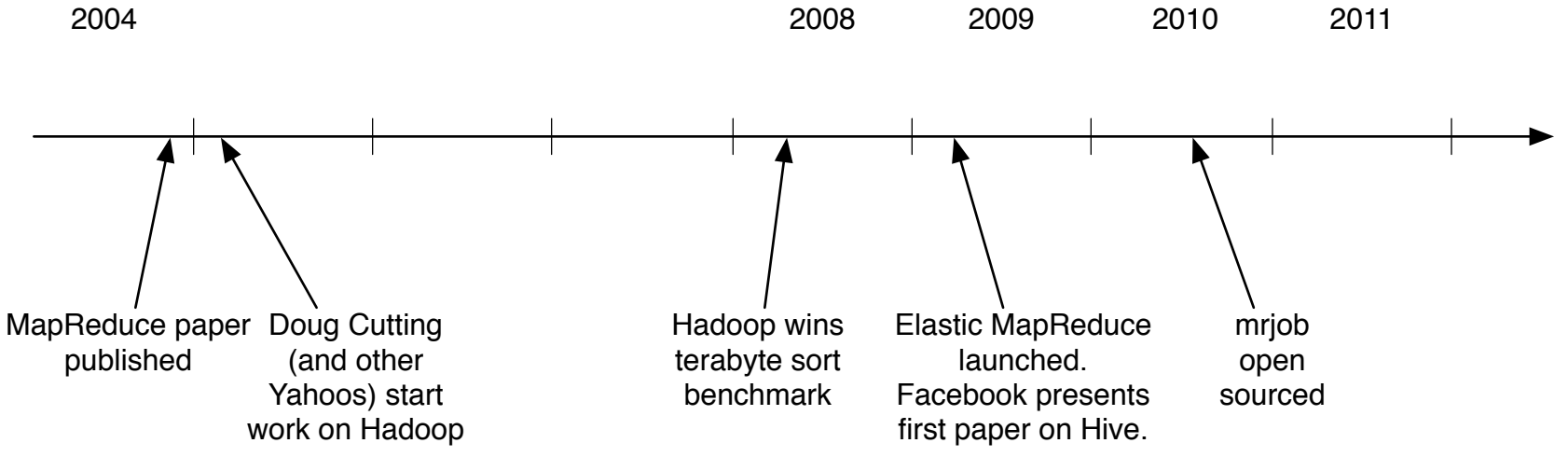
In the figure below, 4 inputs are mapped to 16 (key, value) tuples.



These tuples are sorted by those keys, and then reduced, into 4 new (key, value) tuples.
Finally, these new tuples are once again being sorted and reduced, and so on...

source: http://code.google.com/p/mapreduce-framework/wiki/MapReduce

Timeline


2004
2008
2009
2010
2011
MapReduce paper published
Doug Cutting (and other Yahoos) start work on Hadoop
Hadoop wins terabyte sort benchmark
Elastic MapReduce launched. Facebook presents first paper on Hive.
mrjob open sourced

An overview of available (and not available) tools

1) **MapReduce**. Google's C++ implementation, which sits atop GFS, chubby, BigTable, and who knows what else. See also sawzall.

2) **Hadoop**. Apache's/Facebook's/Yahoo's Java implementation, which furthermore includes, or else integrates with: HDFS (a reimplementation of GFS), Zookeeper (a reimplementation of chubby), and HBase (a reimplementation of BigTable). See also Pig (not exactly sawzall, but another take on it).

3) **Hive**. Apache's/Facebook's data warehousing system, which allows users to query large datasets using SQL-like syntax, over Hadoop.

4) **Elastic MapReduce**. Amazon's API for spinning up temporary Hadoop clusters (aka "Jobflows"), typically reading input and writing output to S3. It is much, much easier than bringing up your own Hadoop cluster.

5) **mrjob**. Yelp's Python framework for running MapReduce jobs on Hadoop clusters, with or without ElasticMapReduce.

Terms for Elastic MapReduce

1) **job flow**. A transient (or semi-transient) Hadoop cluster. Jobflows typically terminate after all their steps are done, but they can also be flagged as "permanent," which just means they don't shut down on their own.

2) **master, core, and task nodes**. A jobflow has one master node; when it dies, your jobflow dies (or is at least supposed to). It has a fixed number of core nodes (determined when the jobflow was started) -- these core nodes can do map/reduce tasks, and also serve HDFS. It can also have a changeable number of task nodes, which do tasks but don't serve HDFS.

3) **step**. A jobflow contains one or more "steps". These may or may not be actual Hadoop "jobs" -- sometimes a step is just for initialization, other times it is an actual map/reduce job.

4) **spot instance vs on-demand**. By default, any job flow you start will use normal, "on-demand" EC2 instances, with a fixed hourly price. You may alternatively specify a spot instances by naming a maximum price (a bid price) you're willing to pay. If your bid price is below the current spot price, then you will pay the spot price for your instances (typically 50-66% cheaper than on-demand instances).

The easiest ways to get started with Elastic MapReduce, #1

If you know Python, start with mrjob.

Here's a sample MRJob class, which you might write to wordcounter.py:

```
from mrjob.job import MRJob

class MRWordCounter(MRJob):
    def mapper(self, key, line):
        for word in line.split():
            yield word, 1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    MRWordCounter.run()
```

And how you run it:

```
export AWS_ACCESS_KEY_ID=...; export AWS_SECRET_ACCESS_KEY=...
python wordcounter.py -r emr < input > output
```

These samples, and much more documentation, is at http://packages.python.org/mrjob/writing-and-running.html.

The easiest ways to get started with Elastic MapReduce, #2

If you know any other scripting language, start with the AWS elastic-mapreduce command line tool, or the AWS web console. Both are outlined at:

http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/

An example of the command line, assuming a file wordSplitter.py, is:

```
./elastic-mapreduce --create --stream \
    --mapper  s3://elasticmapreduce/samples/wordcount/wordSplitter.py \
    --input   s3://elasticmapreduce/samples/wordcount/input \
    --output  s3://mybucket.foo.com/wordcount \
    --reducer aggregate
```

A (not so) quick reference card for the command line tool is at:

http://s3.amazonaws.com/awsdocs/ElasticMapReduce/latest/emr-qrc.pdf

Getting started with Hive is not so easy

1) You're going to need to put your data into some sort of warehouse, probably in S3. (HDFS will not persist in the cloud). This may involve its own MapReduce step, but with careful attention to how you output your files to S3. Partitioning matters, but you won't get it with just the default Hadoop streaming jar! (We use oddjob's)

2) To start a jobflow with Hive programmatically (i.e., not using the command line elastic-mapreduce client), you need to add several initial steps to the jobflow -- notably ones that start Hive and possibly configure its heapsize. Fortunately, Amazon's elastic-mapreduce client is easy enough to reverse engineer...

3) You're going to need to make a schema (surprise) for your data warehouse.

4) You may have warehoused data in JSON. Hive much prefers flat, tabular files.

5) To talk to Hive programmatically from other servers, you'll need to use Hive's thrift binding, and also open up the ElasticMapReduce-master security group (yes, EMR created this for you when you first ran a jobflow) so that your application box can talk to the master node of your job flow.

6) All that said, it may beat writing your own custom MapReduce jobs.

Other pitfalls to beware

1) The JVM is not your friend. It especially loves memory.

- be prepared to add swap on your task nodes, at least until Hadoop's fork()/exec() behavior is fixed in AWS' version of Hadoop

- be prepared to profile your cluster with Ganglia (it's really, really easy to setup). Just add this bootstrap action to do so: s3://elasticmapreduce/bootstrap-actions/install-ganglia

2) Multiple output files can get you into trouble, especially if you don't sort your outputs first.

3) Hadoop, and Hive as well, much prefer larger files to lots of small files.

Further reading

- "MapReduce: Simplified Data Processing on Large Clusters" OSDI'04. http://research.google.com/archive/mapreduce.html

- "Hive - A Warehousing Solution Over a Map-Reduce Framework" VLDB, 2009. http://www.vldb.org/pvldb/2/vldb09-938.pdf

- *Hadoop: The Definitive Guide*. O'Reilly Associates, 2009.

Thank you!

(P.S. We're hiring. hblanks@monetate.com)