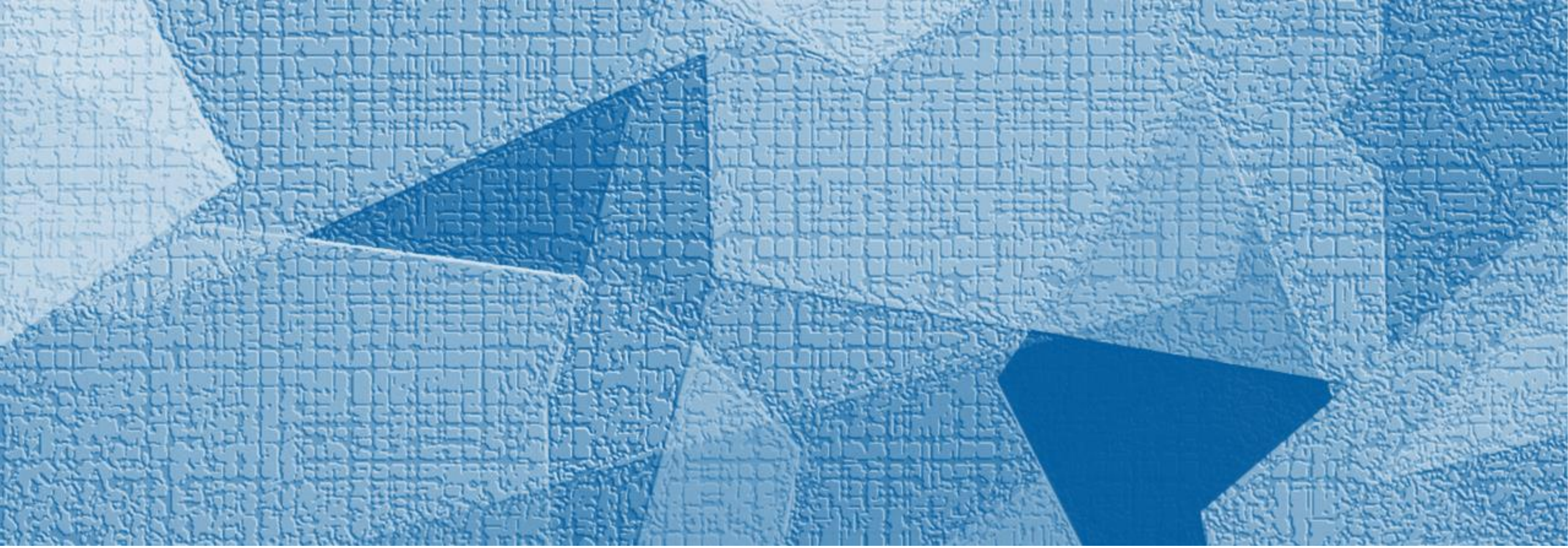


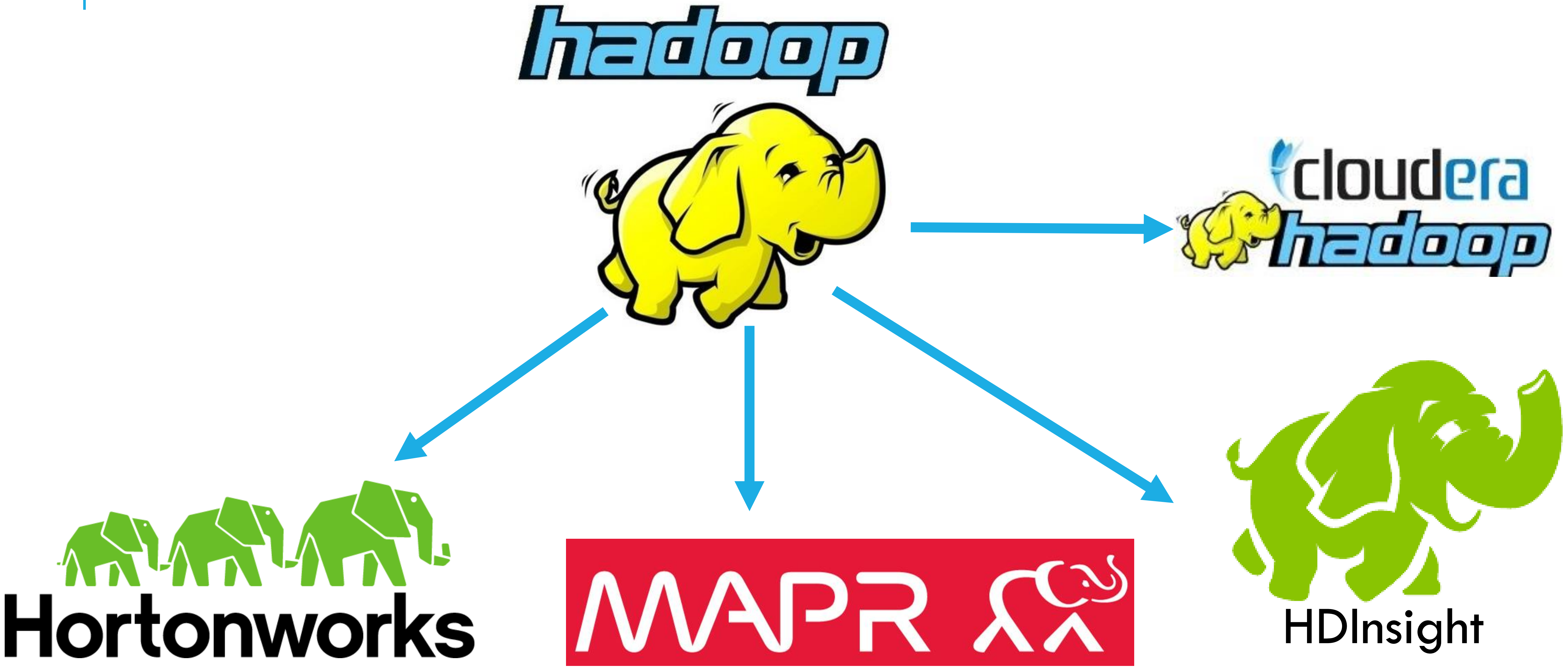
BIG DATA ENGINEERING WITH MAPREDUCE & HIVE

ITINERARY

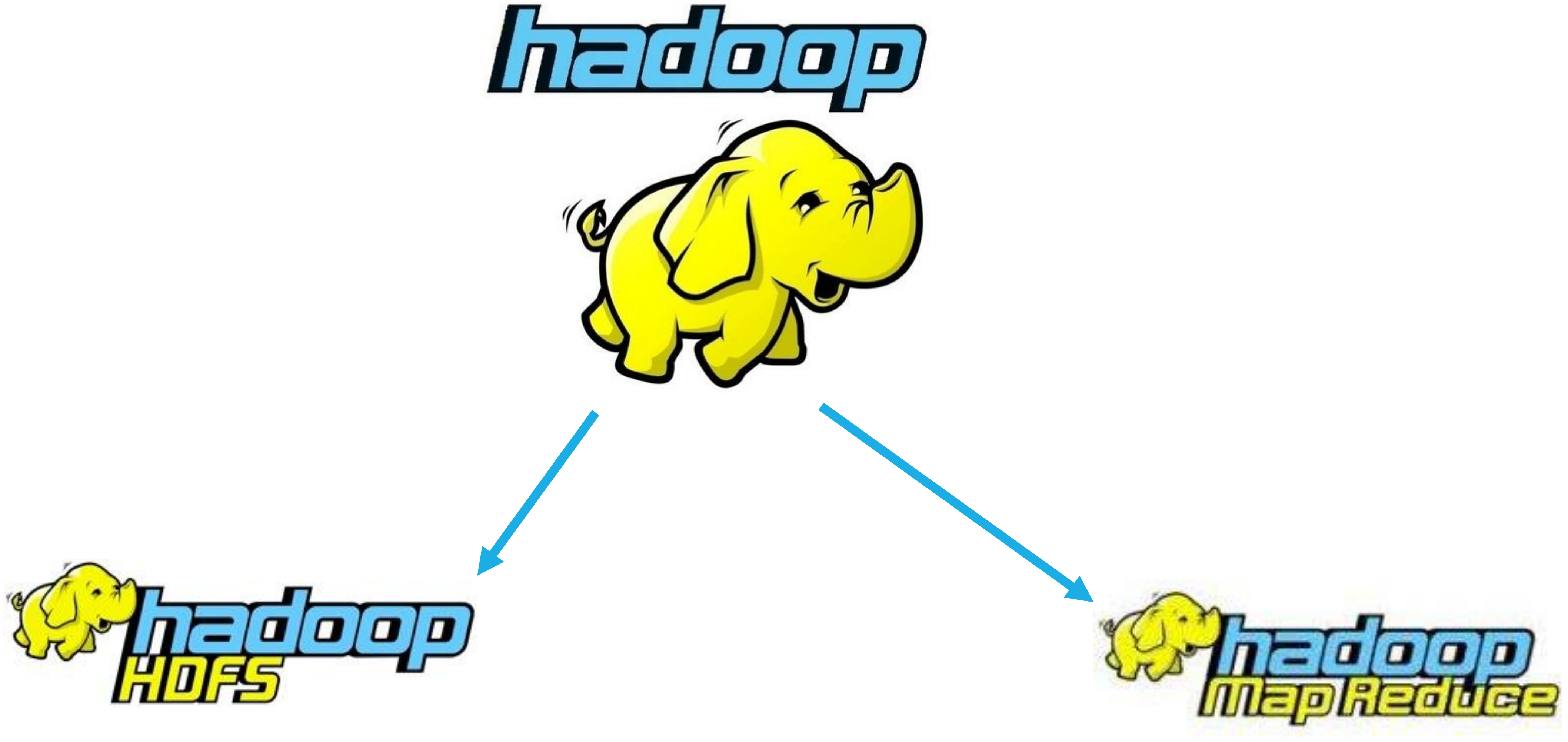
hadoop

HIVE

HADOOP IMPLEMENTATIONS

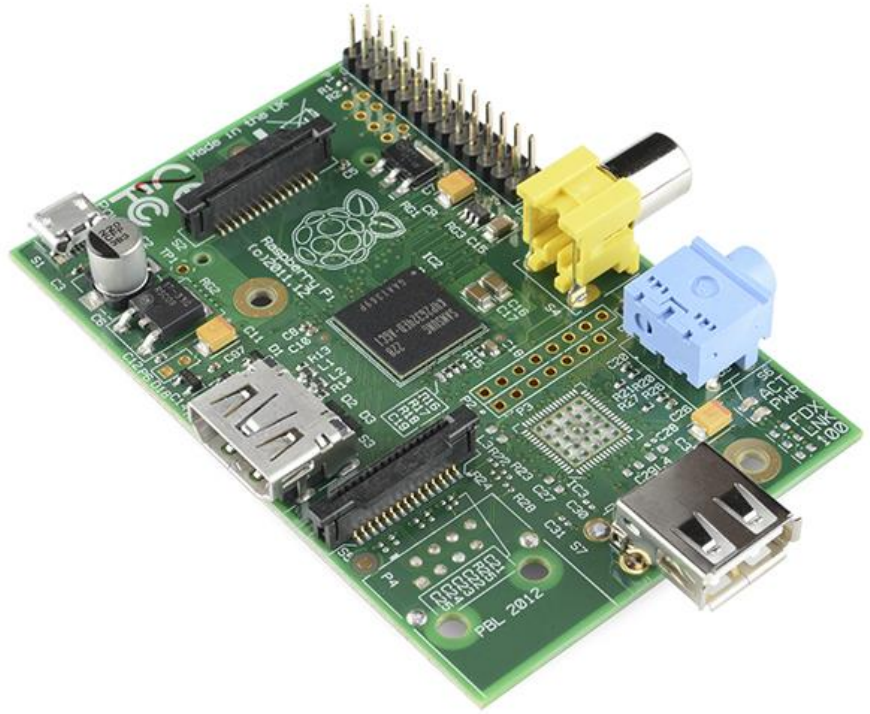


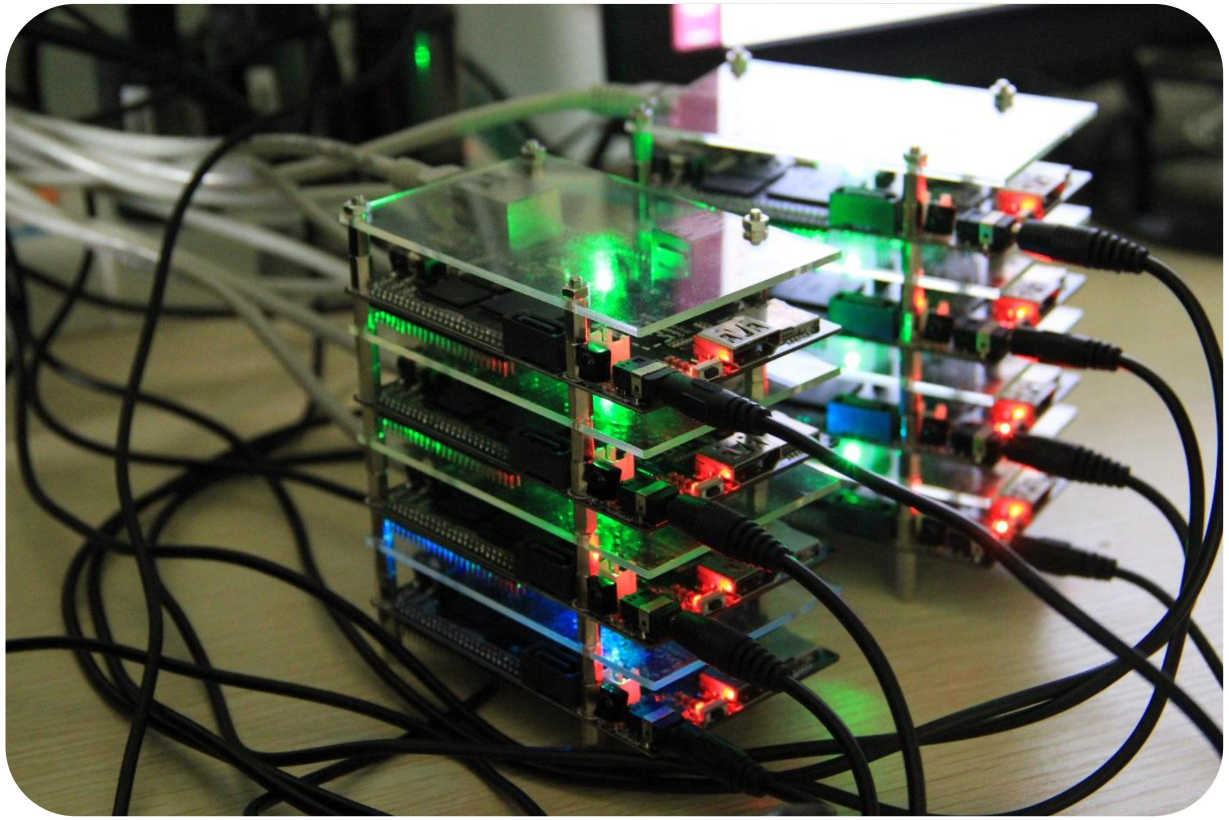
HADOOP



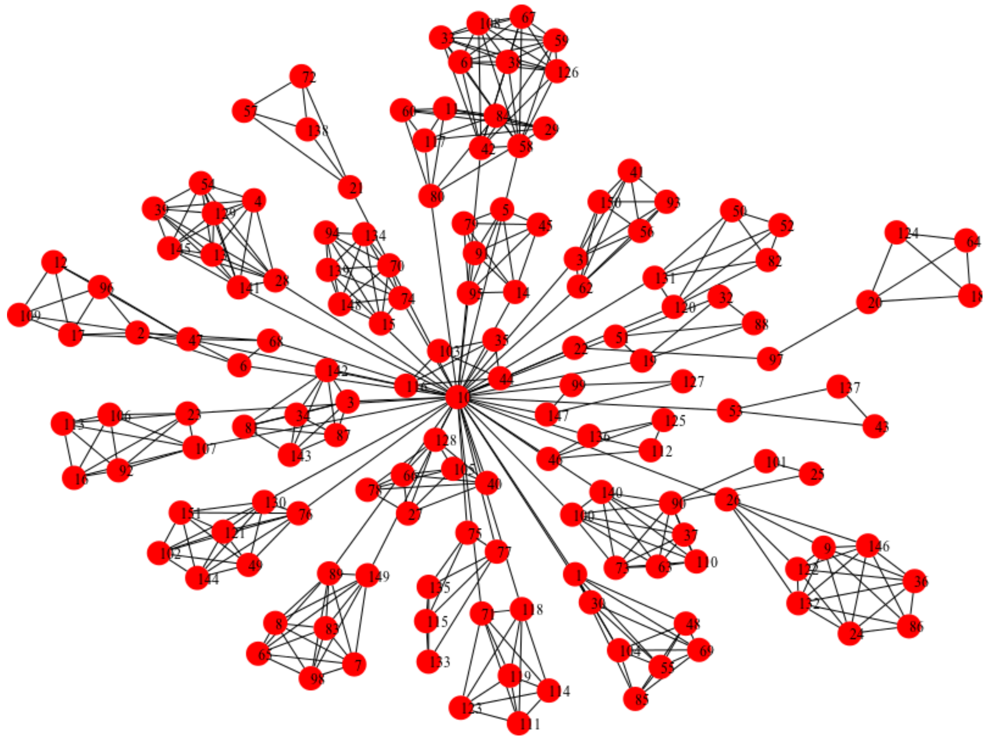
TURN BACK THE CLOCK, THE MAINFRAME

DISTRIBUTED COMPUTING

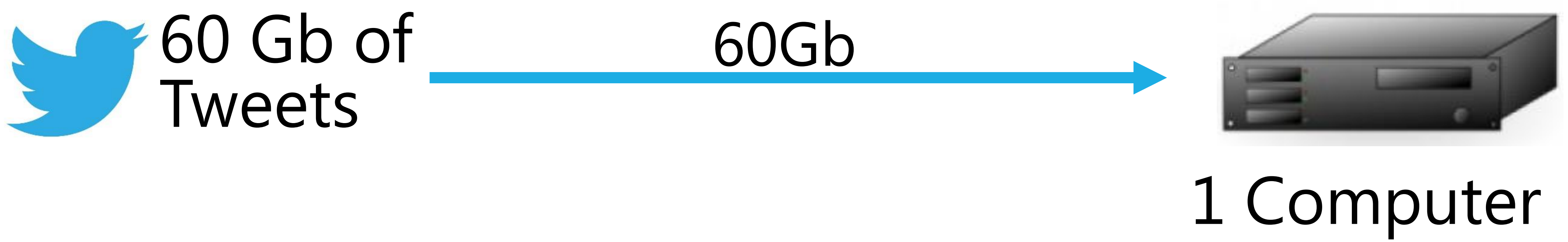




CLOUD COMPUTING

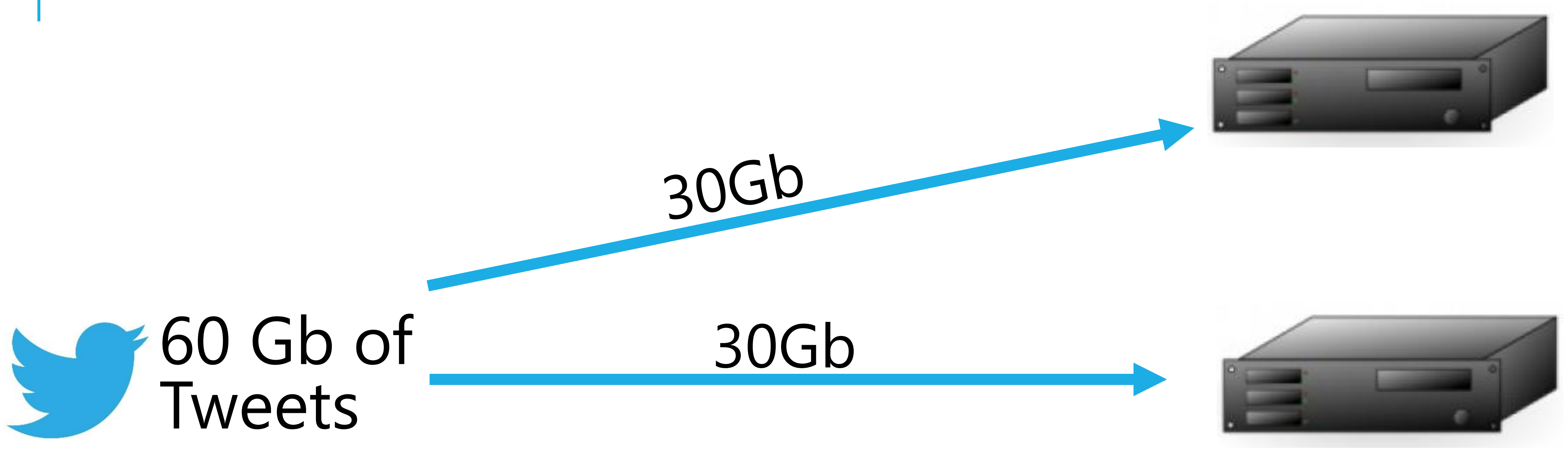


HDFS & MAPREDUCE



Processing: 30 hours

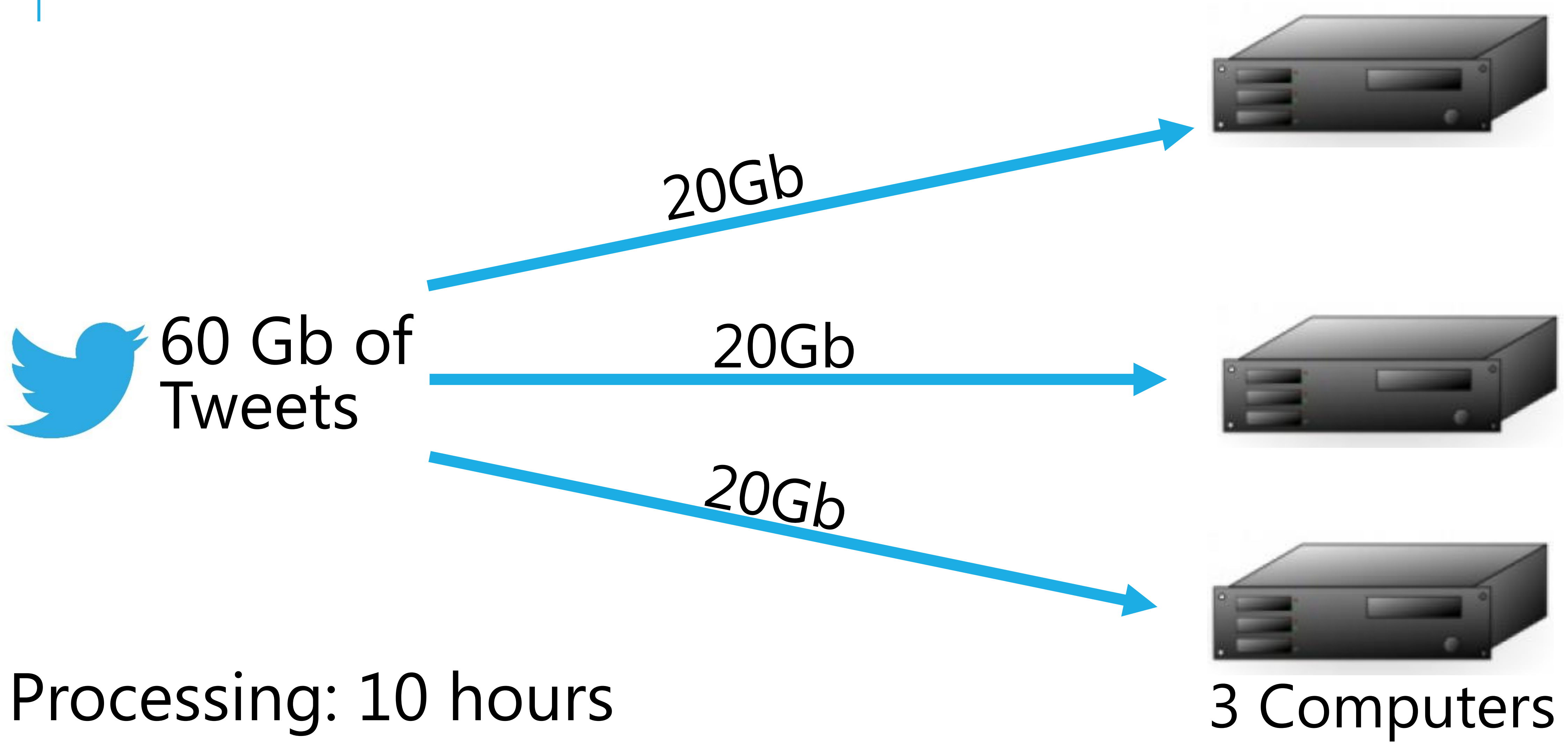
HDFS & MAPREDUCE



Processing: 15 hours

2 Computers

HDFS & MAPREDUCE

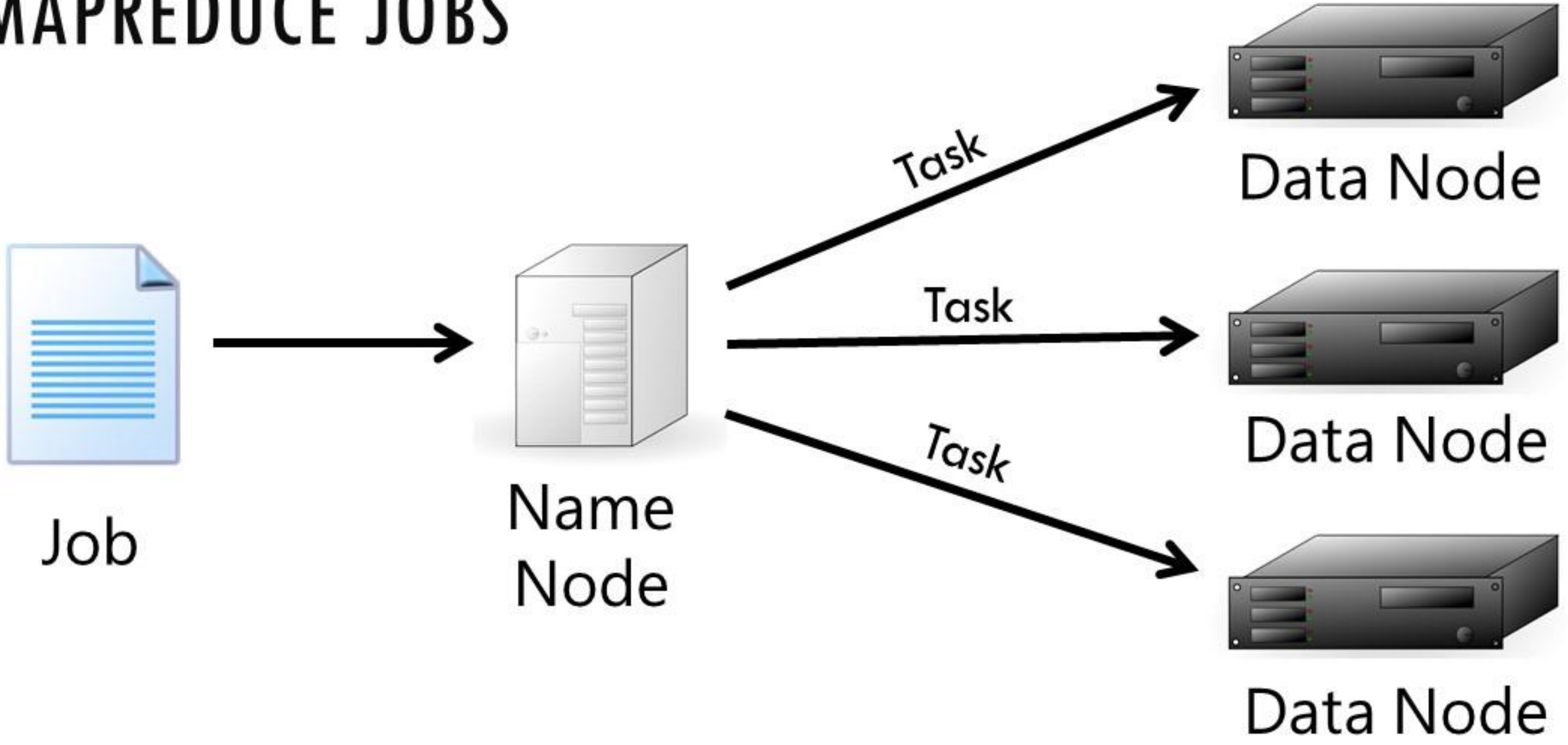


Processing: 10 hours

MOST CASES, LINEAR SCALING OF PROCESSING POWER

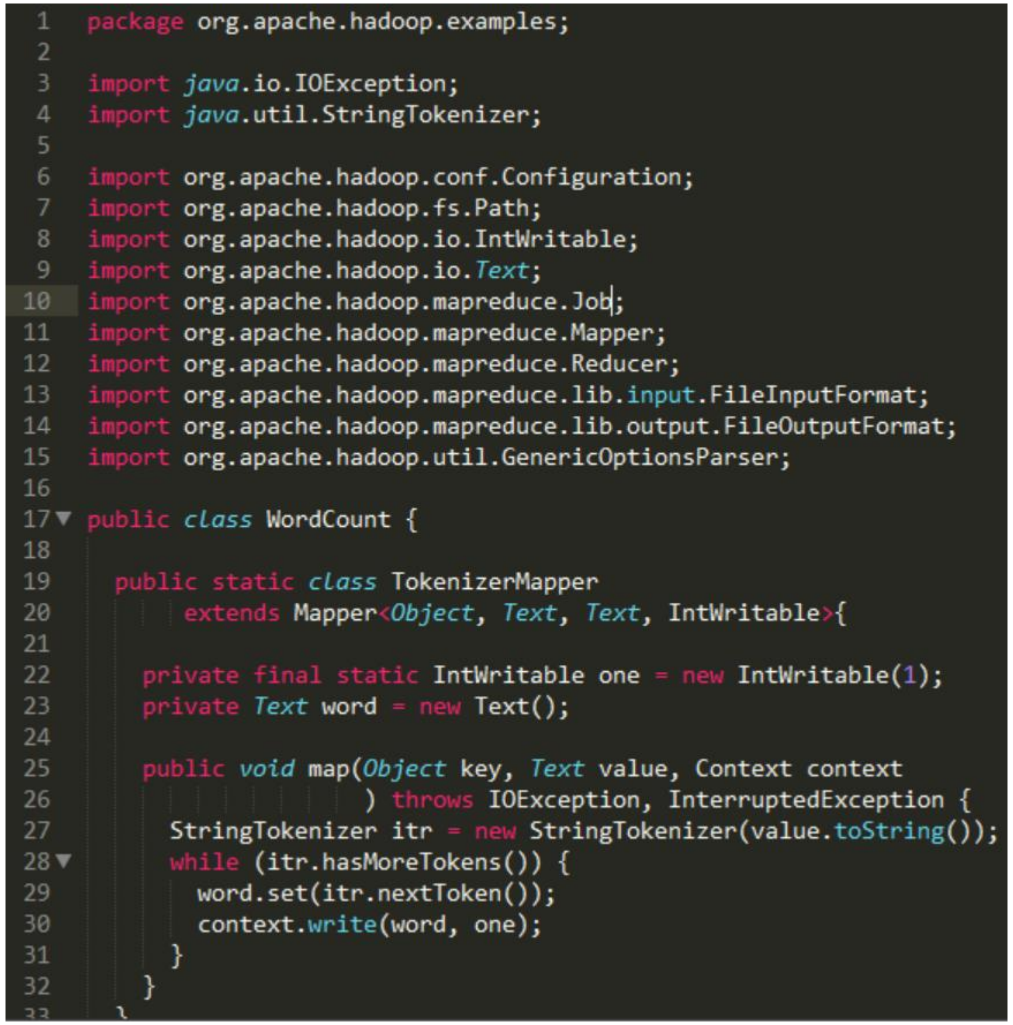
Number of Computers	Processing Time (hours)
1	30
2	15
3	10
4	7.5
5	6
6	5
7	4.26
8	3.75
9	3.33

MAPREDUCE JOBS



LIMITATIONS WITH MAPREDUCE

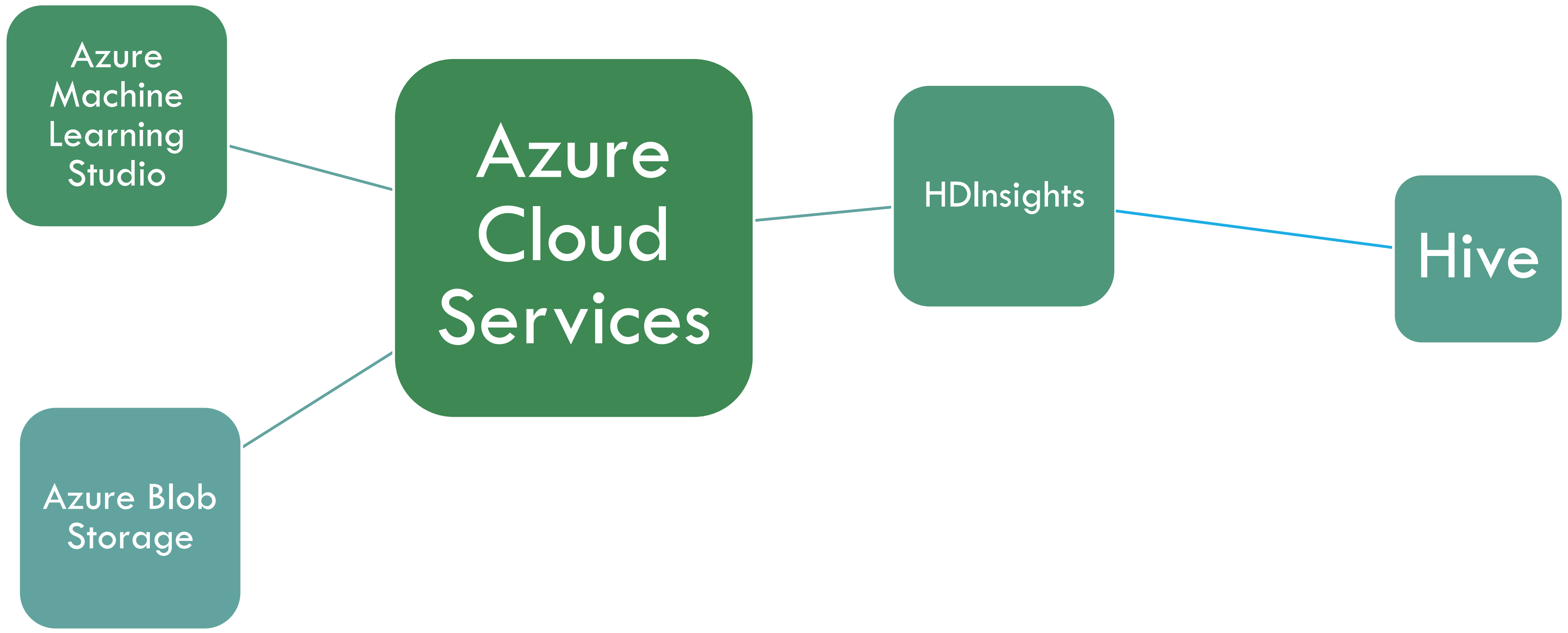
- ~200 lines of code to do anything
- Slow
- Troubleshooting multiple computers
- Good devs are scares
- Expensive certifications



```java
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

- **Ambari**: Cluster provisioning, management, and monitoring
- **Avro** (Microsoft .NET Library for Avro): Data serialization for the Microsoft .NET environment
- **HBase**: Non-relational database for very large tables
- **HDFS**: Hadoop Distributed File System
- **Hive**: SQL-like querying
- **Mahout**: Machine learning
- **MapReduce and YARN**: Distributed processing and resource management
- **Oozie**: Workflow management
- **Pig**: Simpler scripting for MapReduce transformations
- **Sqoop**: Data import and export
- **Storm**: Real-time processing of fast, large data streams
- **Zookeeper**: Coordinates processes in distributed systems

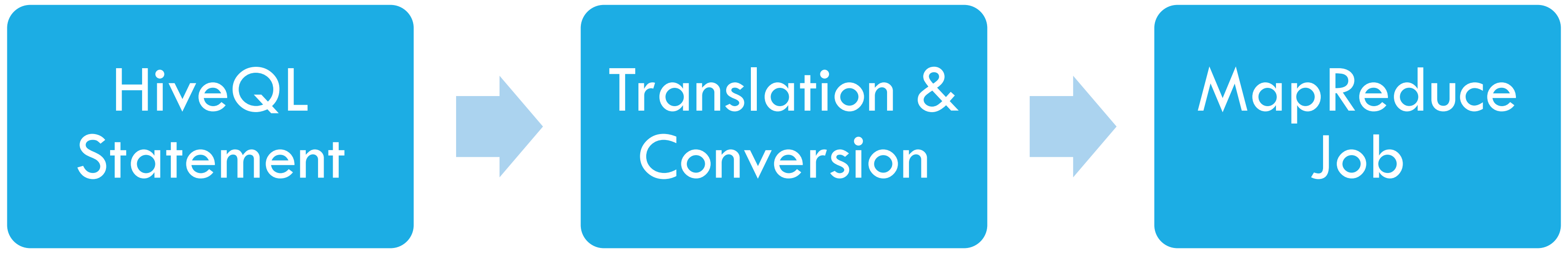
AZURE ECOSYSTEM



HIVE WITHIN AZURE STACK

Windows Azure

HIVE JOBS



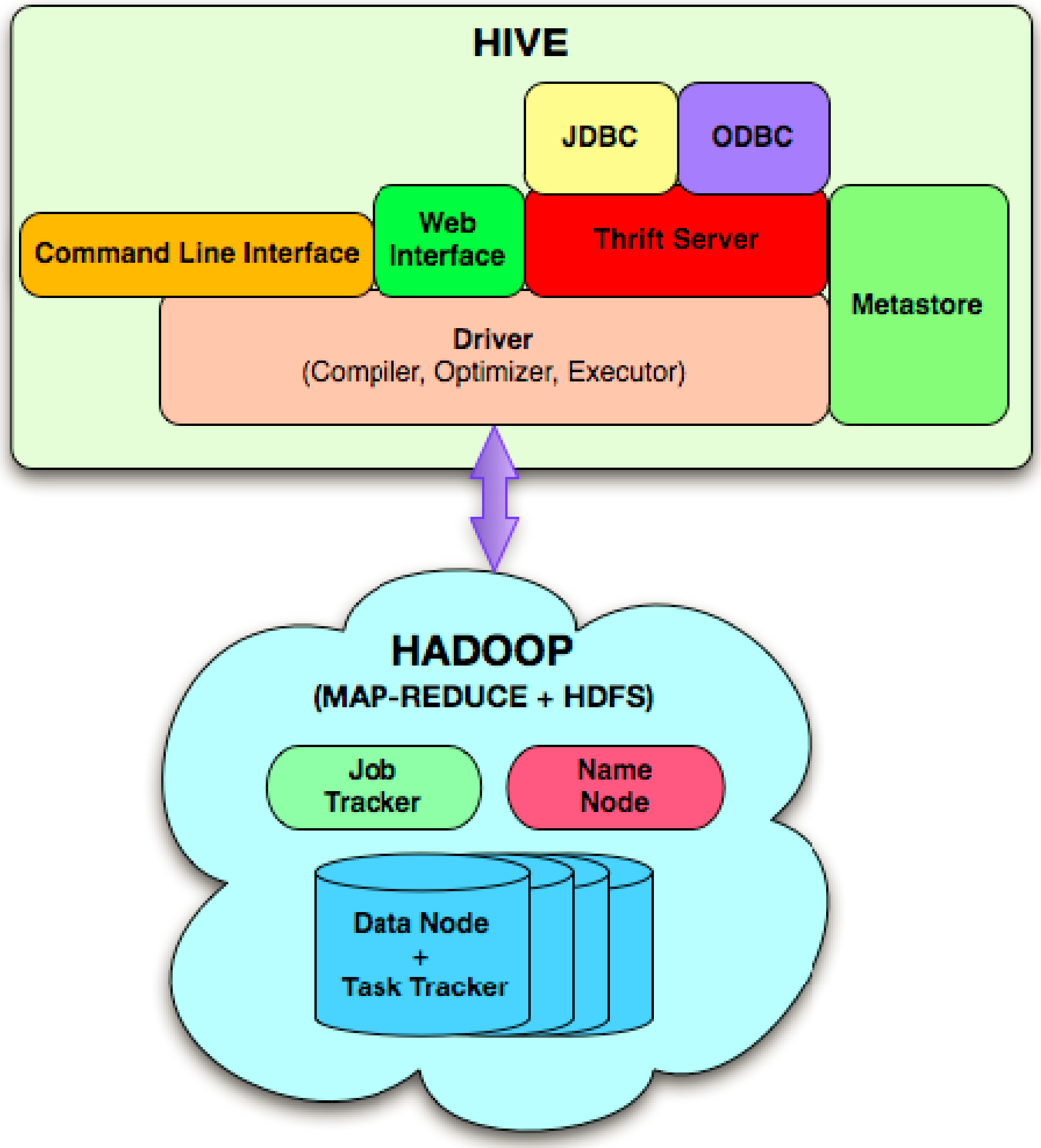
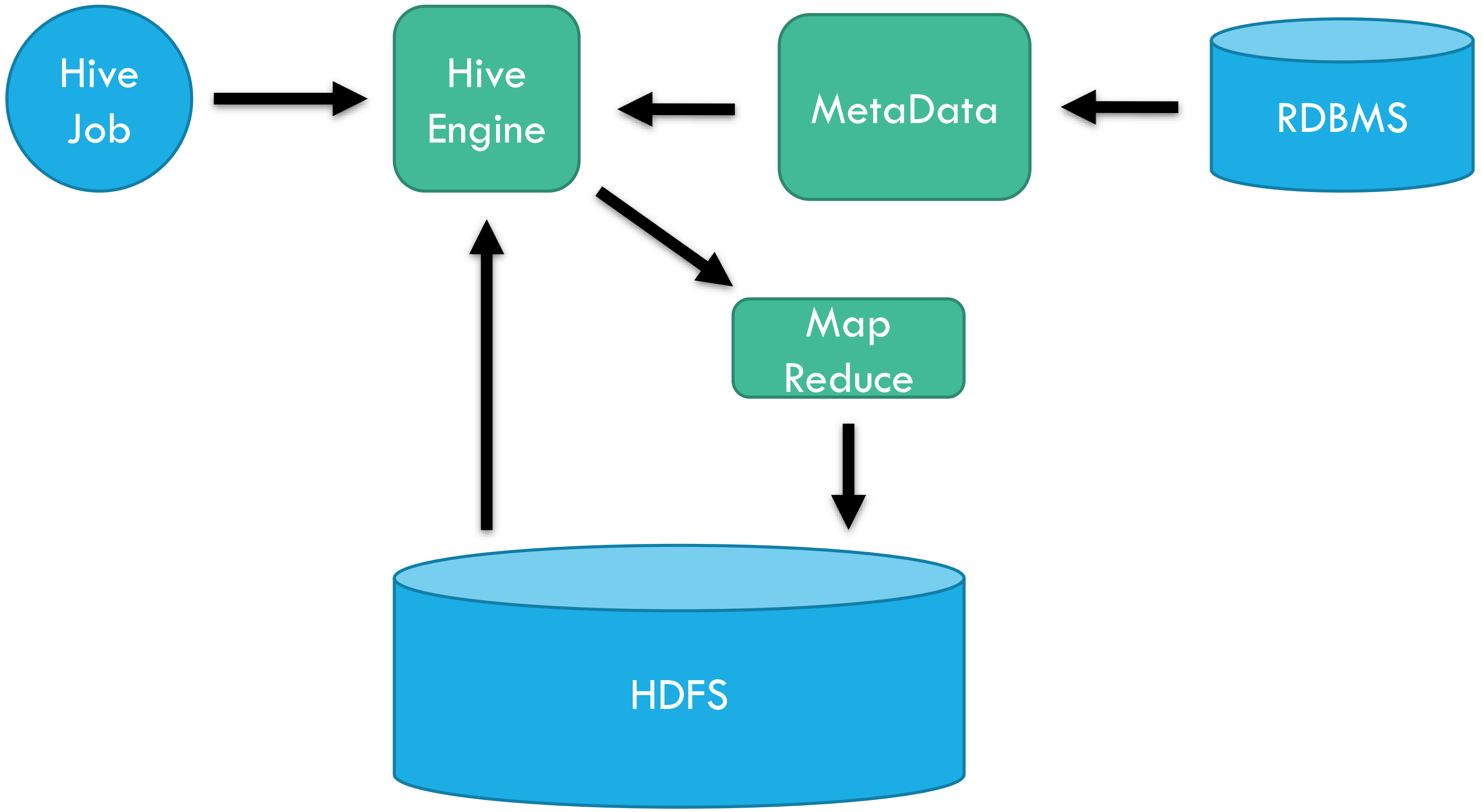


Fig. 1: Hive System Architecture

HIVE ARCHITECTURE



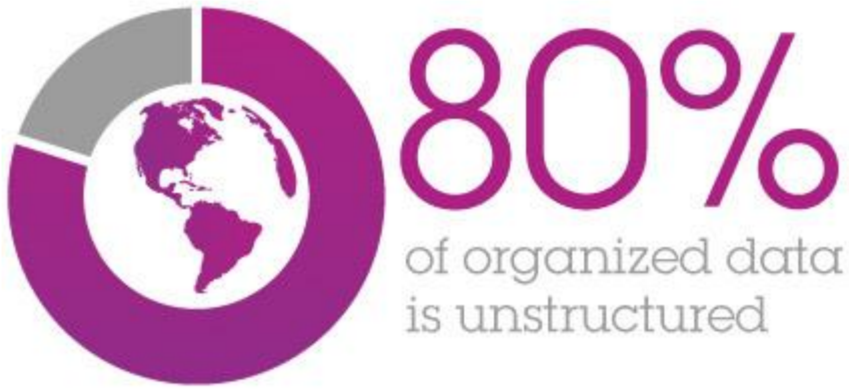
Data File = Unstructur
ed Data

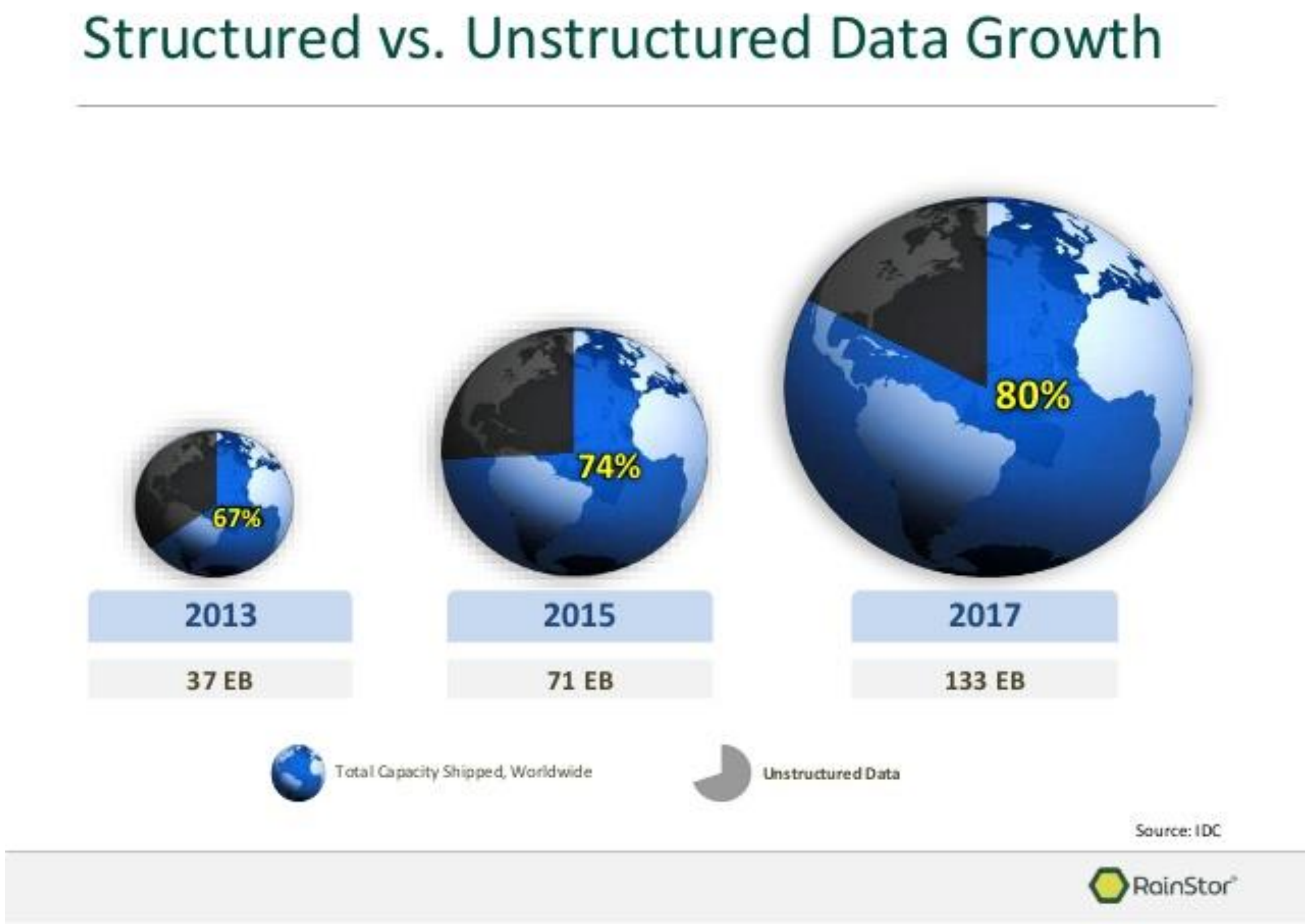
Data File + Metadata File/DB = Structured
Data

WHY HIVE?

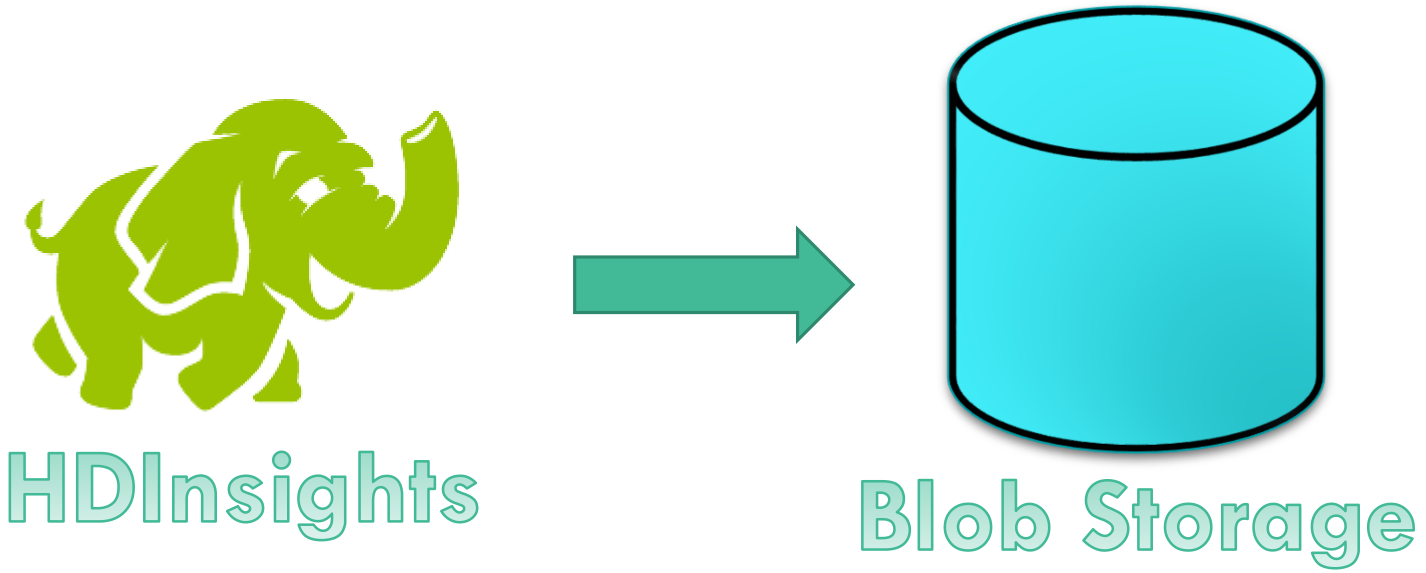
- SQL spoken here (HiveQL)
- ODBC driver
- BI Integration
- Supports only Structured Data

LIMITATIONS

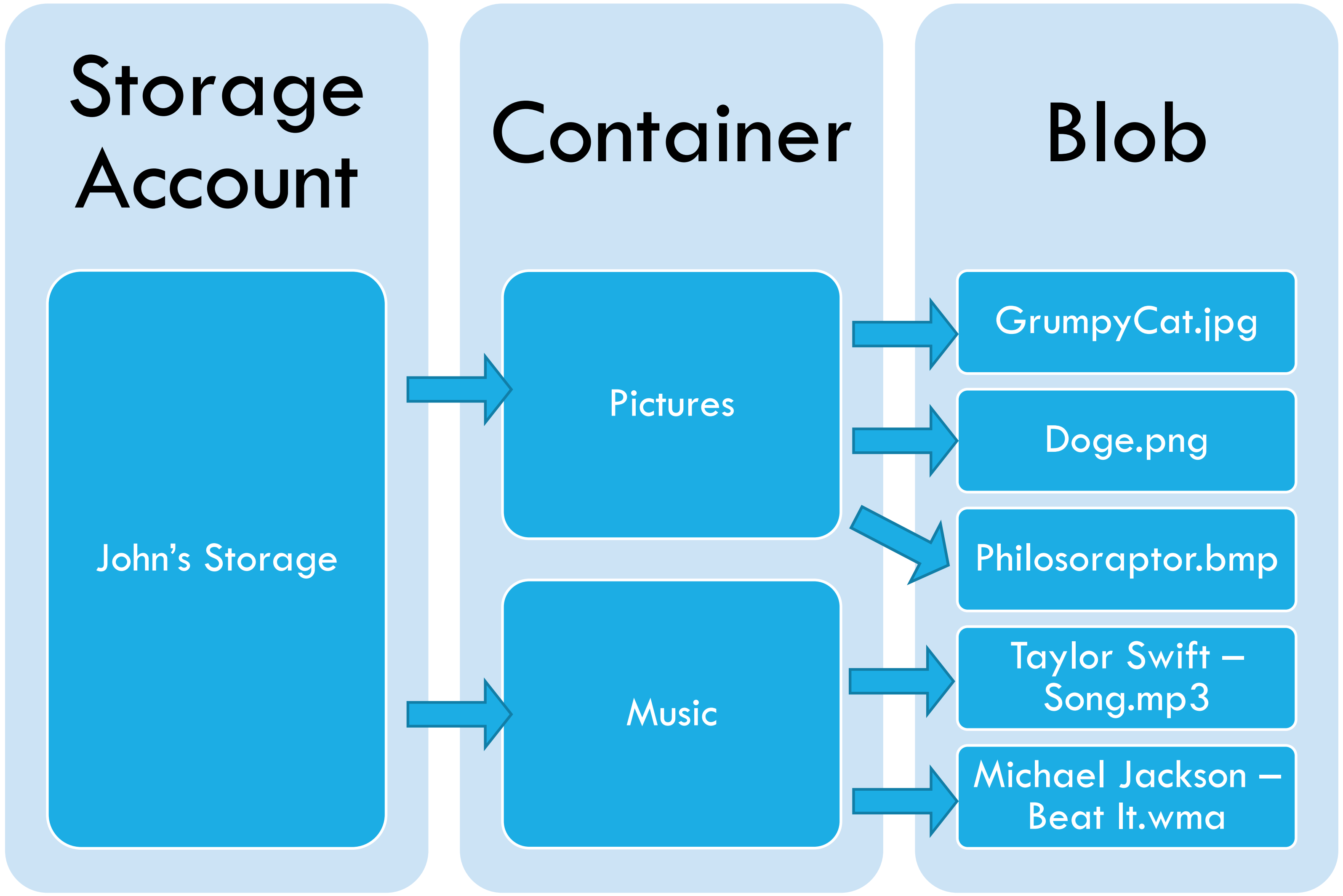




AZURE BLOB STORAGE



AZURE BLOB STORAGE



WHEN TO USE EACH?

C#

!=

Java

Vs.

HIVE

Vs.

C#/Java MapReduce

Pig