Introduction to Data Science with Hadoop

Glynn Durham | Senior Instructor
glynn@cloudera.com

May 2017

Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

cloudera

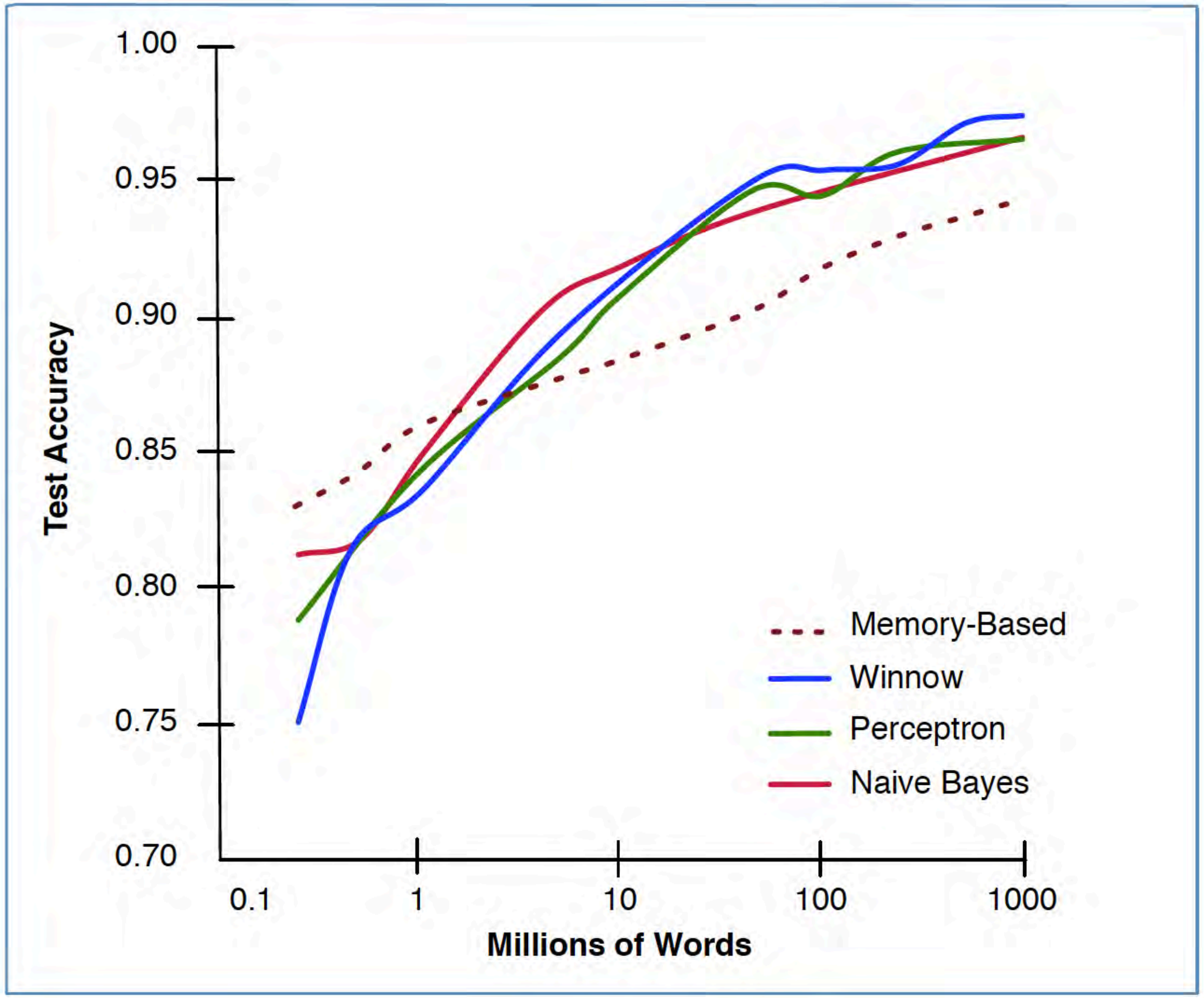
Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

Data Science is …

- gathering data,
  potentially of many types and from many sources,
- wrangling that data into useful forms,
  and
- applying statistical programming and machine learning,
  to gain new information from the data.

cloudera

Machine Learning and Data Volume



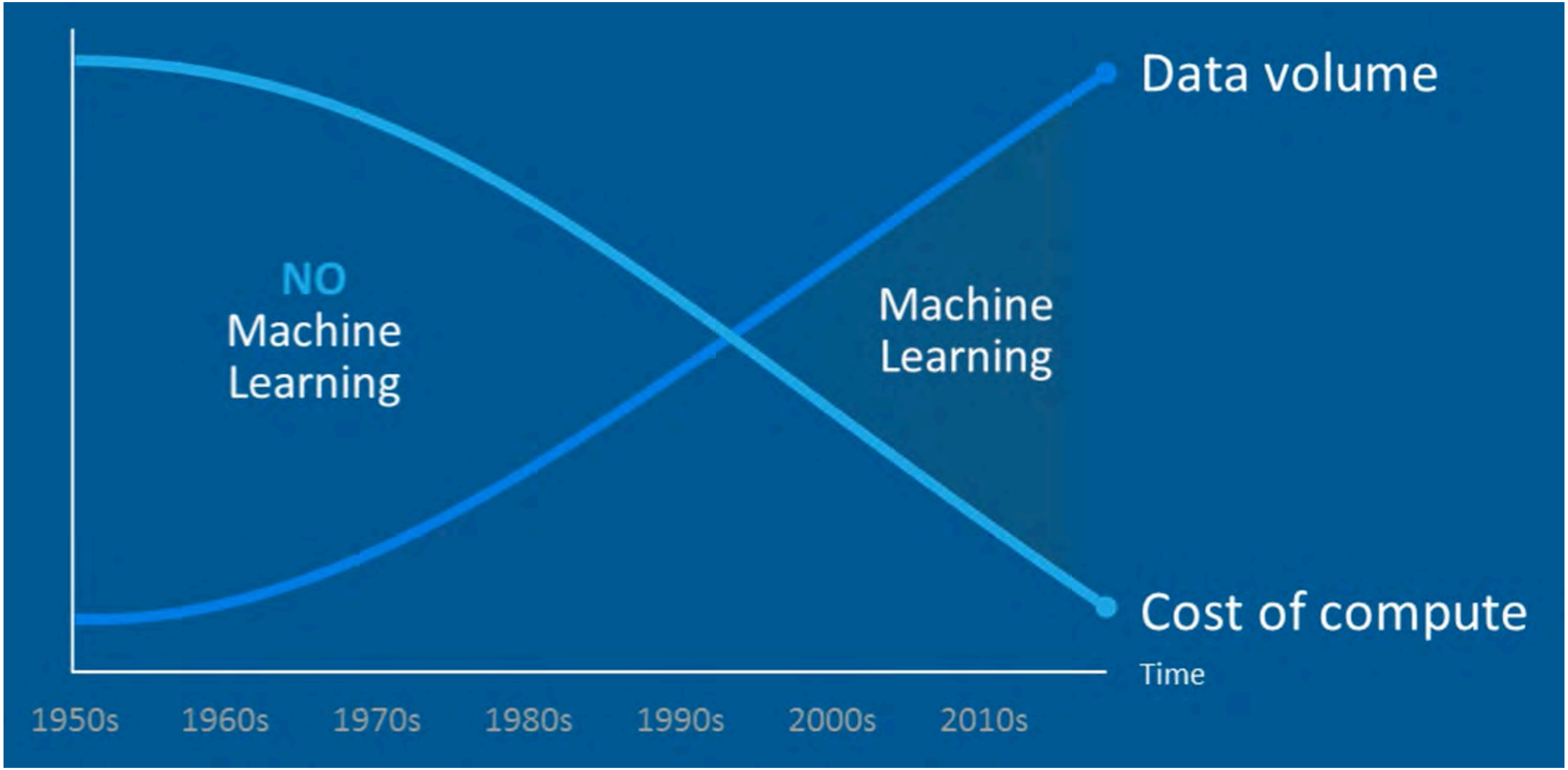
"It's not who has the best algorithms who wins. It's who has the most data."

[Banko and Brill, 2001]

Hadoop is …

- an open source software platform for
- acquiring, storing, and processing massive volumes of data,
- economically.

The Age of Machine Learning



Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

cloudera

The word “Hadoop” means

- a child’s toy

  or
- Hadoop Core

  or
- the Hadoop Ecosystem.

cloudera

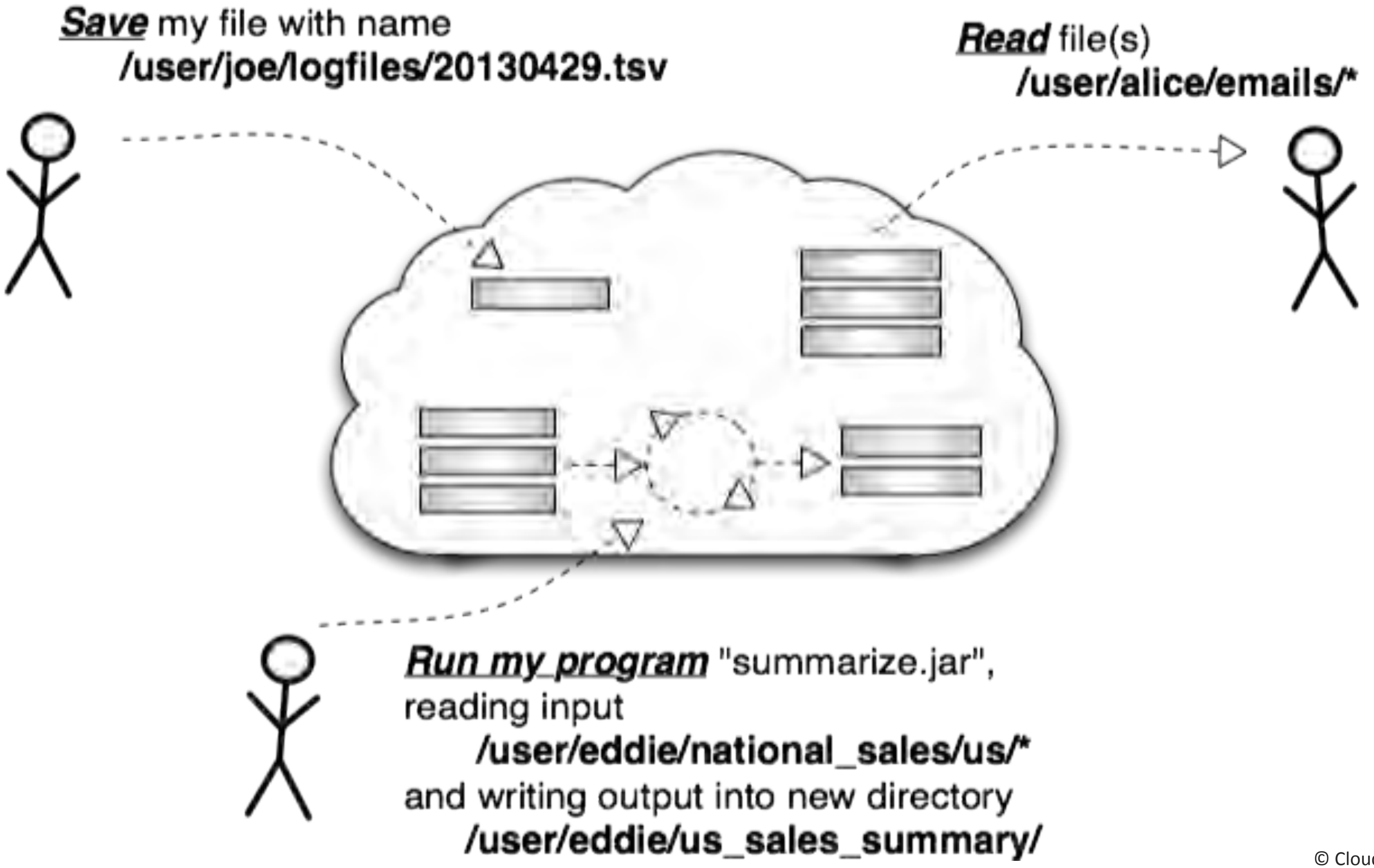
Hadoop Core

hadoop.apache.org

- A free open source software software project
- Managed transparently online, at the Apache Software Foundation (ASF), apache.org
- The project was started in 2006, based on papers from Google, in 2003 and 2004

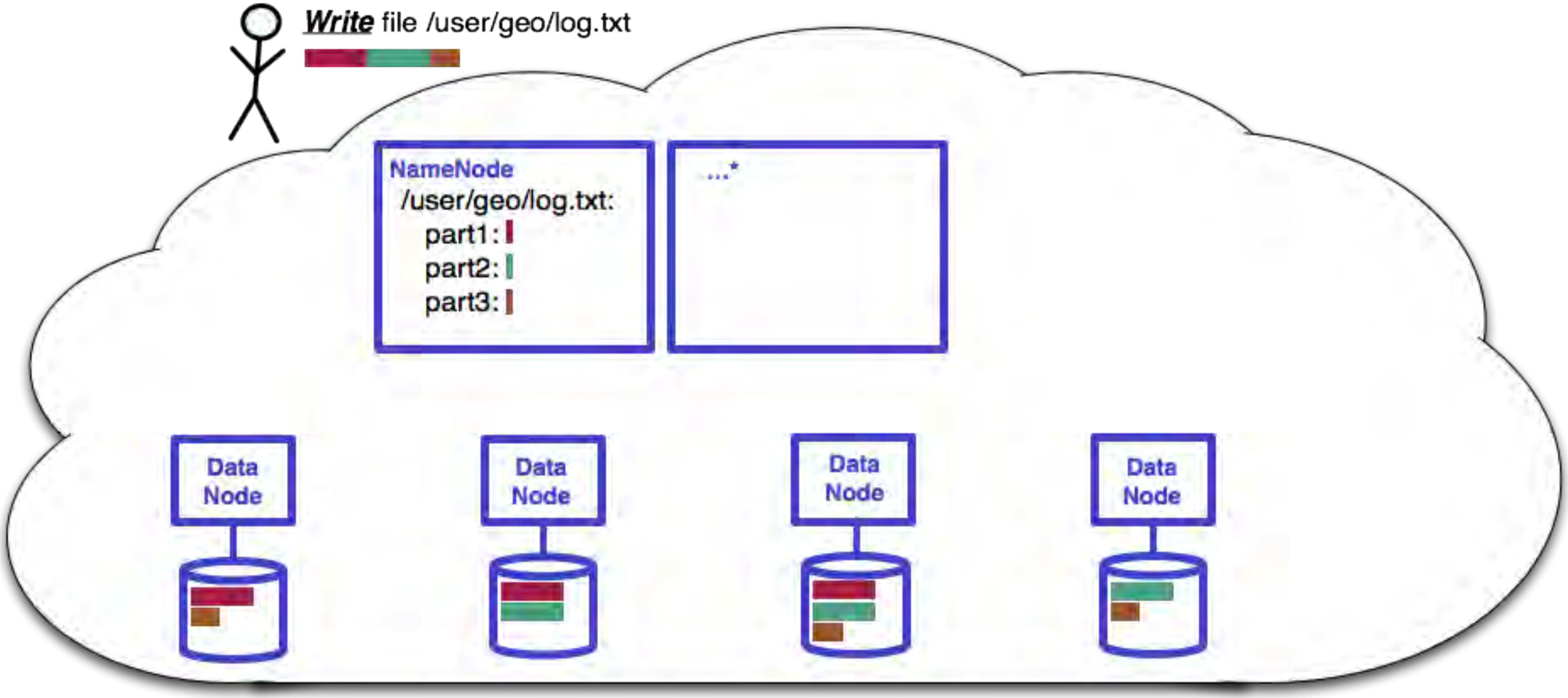
- Consists of:
  - HDFS (Hadoop Distributed File System), for storage
  - Hadoop MapReduce, for processing
  - YARN (Yet Another Resource Negotiator)

Hadoop Core main features: File storage and batch programming



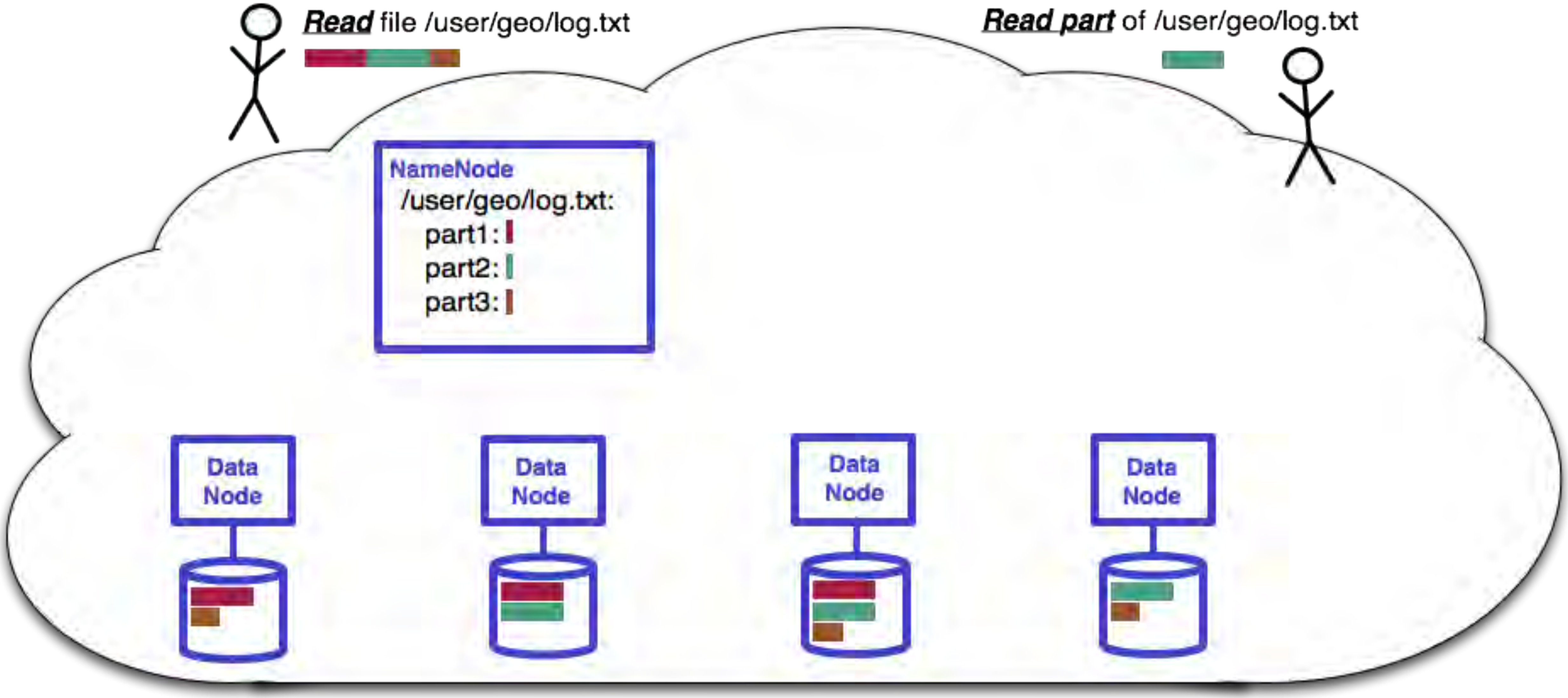
cloudera

HDFS Writes



Write file /user/geo/log.txt
NameNode
/user/geo/log.txt:
part1 :
part2:
part3:
...*
Data Node
Data Node
Data Node
Data Node

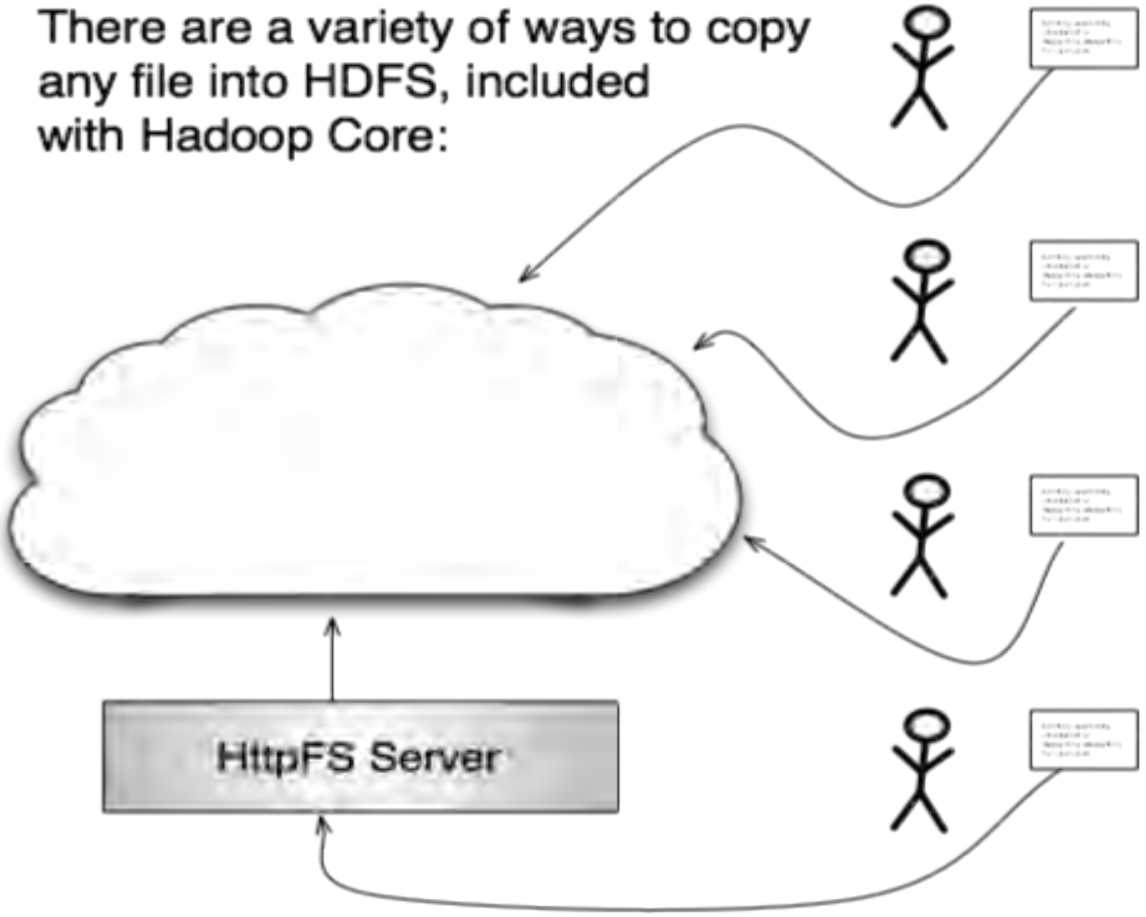
HDFS Reads



cloudera

General File Input/Output

There are a variety of ways to copy any file into HDFS, included with Hadoop Core:



Single line command:

```
hadoop fs
  -put myfile hadfile
```

FUSE

```
cp myfile /hdfs/hadfile
```

Java program using HDFS API

HttpFS

```
curl "http://host:14...
```

Each of these has a complement for copying a file from HDFS

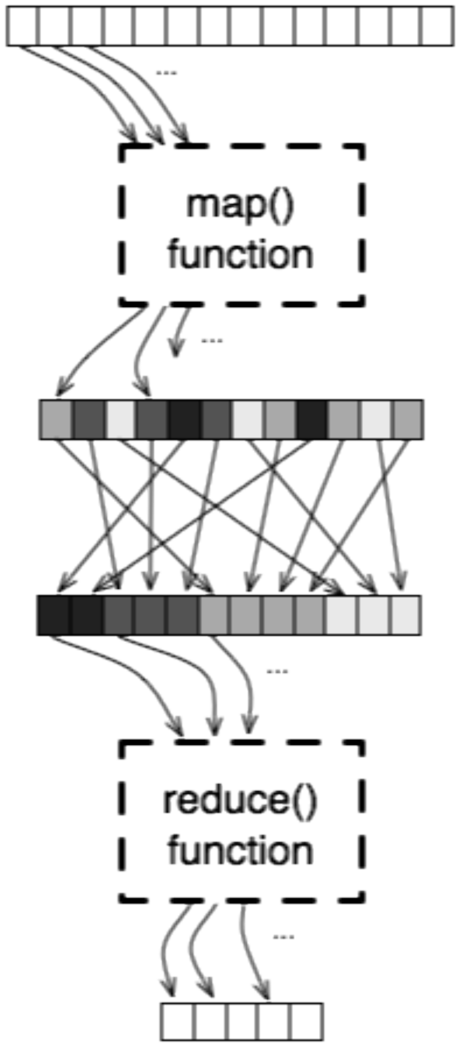
cloudera

HDFS Strengths and Weaknesses

- HDFS is good at:
  - storing enormous files
  - storing lots of data reliably
  - throughput on sequential writes
  - throughput on sequential reads of a file or part of a file
- HDFS is not good at:
  - high speed (low latency) random reads of parts of a file
- HDFS cannot:
  - update any part of a file once written*

  * but you can always write a new file and/or delete, move, and rename files and directories

MapReduce: Programming with simple functions



Input records

Map function takes one input record, returns 0 or more intermediate records

Intermediate records must be of the form (key, value)

Shuffle sorts records by key

Reduce function takes records of one key, returns 0 or more output records

Output records

MapReduce Example: Word Count

Count the number of occurrences of each word over a large amount of input data

- This is the 'hello world' of MapReduce programming

```
map(String input_key, String input_value)
    foreach word w in input_value:
        emit(w, 1)
```

```
reduce(String output_key,
                    Iterator<int> intermediate_vals)
    set count = 0
    foreach v in intermediate_vals:
        count += v
    emit(output_key, count)
```

Word Count, continued

Input to the Mapper:

```
(3414, 'the cat sat on the mat')
(3437, 'the aardvark sat on the sofa')
```

Output from the Mapper:

```
('the', 1), ('cat', 1), ('sat', 1), ('on', 1),
('the', 1), ('mat', 1), ('the', 1), ('aardvark', 1),
('sat', 1), ('on', 1), ('the', 1), ('sofa', 1)
```

cloudera

Word Count, continued

Intermediate data sent to the Reducer:

```
('aardvark', [1])
('cat', [1])
('mat', [1])
('on', [1, 1])
('sat', [1, 1])
('sofa', [1])
('the', [1, 1, 1, 1])
```

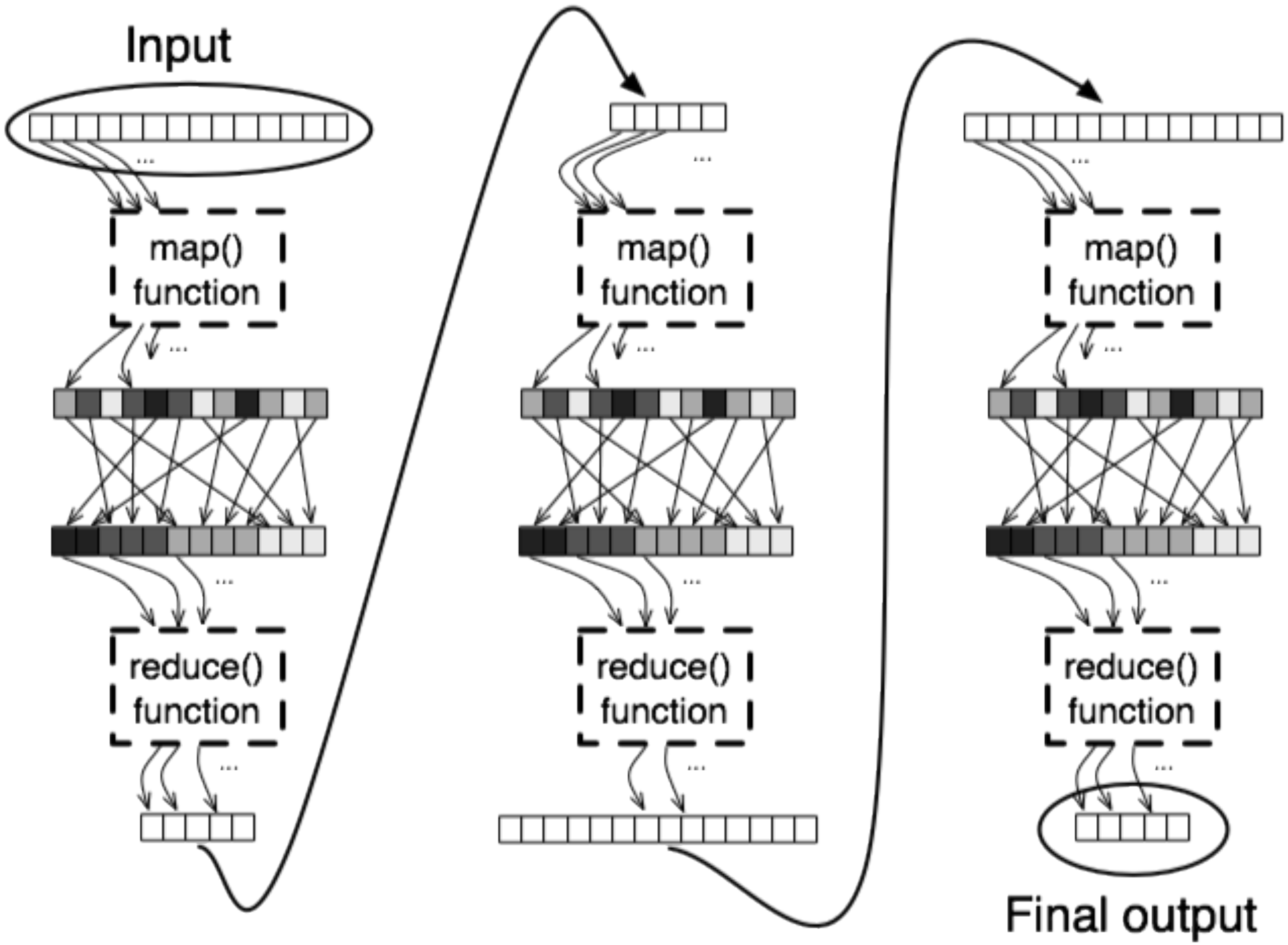
Final Reducer output:

```
('aardvark', 1)
('cat', 1)
('mat', 1)
('on', 2)
('sat', 2)
('sofa', 1)
('the', 4)
```

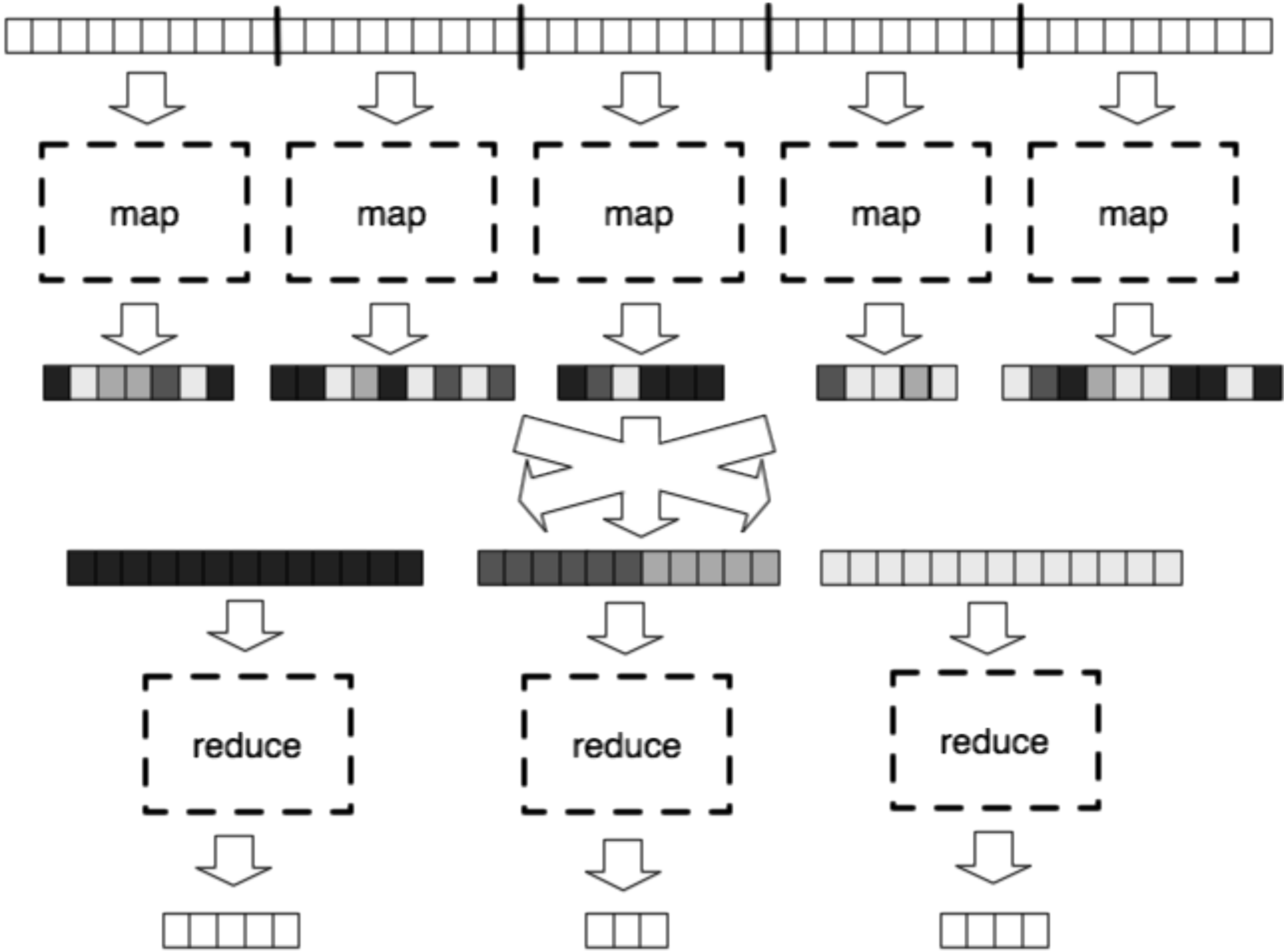
So we just counted words. So what?

- Many problems conform to this pattern:
  - Web log analysis: map() emits an IP address for each web log event; reduce() counts occurrences for each IP address
  - Indexing: For each document, map() emits each term of interest paired with the document ID; reduce() collects and emits all document IDs for each term
  - Page rank algorithm:
    - Every web page (URL) on the Web gets an initial score.
    - map() divides a page's score among all of its outlinks' URLs; reduce() sums the received scores for each URL.
    - Iterate on this procedure.

MapReduce Chains



MapReduce at Scale



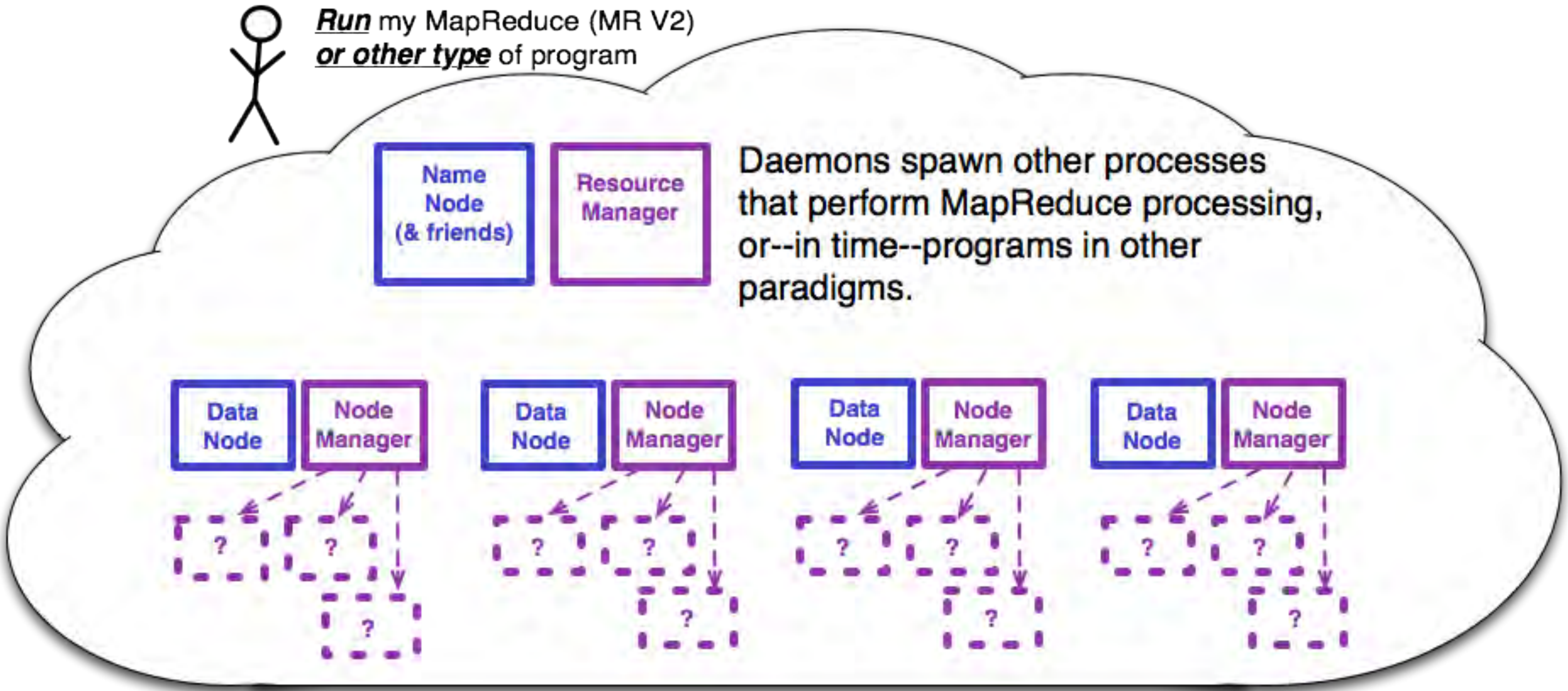
cloudera

MapReduce Strengths and Weaknesses

- MapReduce is good at:
  - processing enormous volumes of data
  - scaling out as you add more machines
  - continuing to completion, even when some machines die
- MapReduce is not good at:
  - running any algorithm you can write in pseudocode
  - algorithms that require shared state overall*

    * but maybe you can get clever with your algorithm design
- MapReduce cannot:
  - run in real time: MapReduce jobs are batch jobs

YARN, Yet Another Resource Negotiator



cloudera

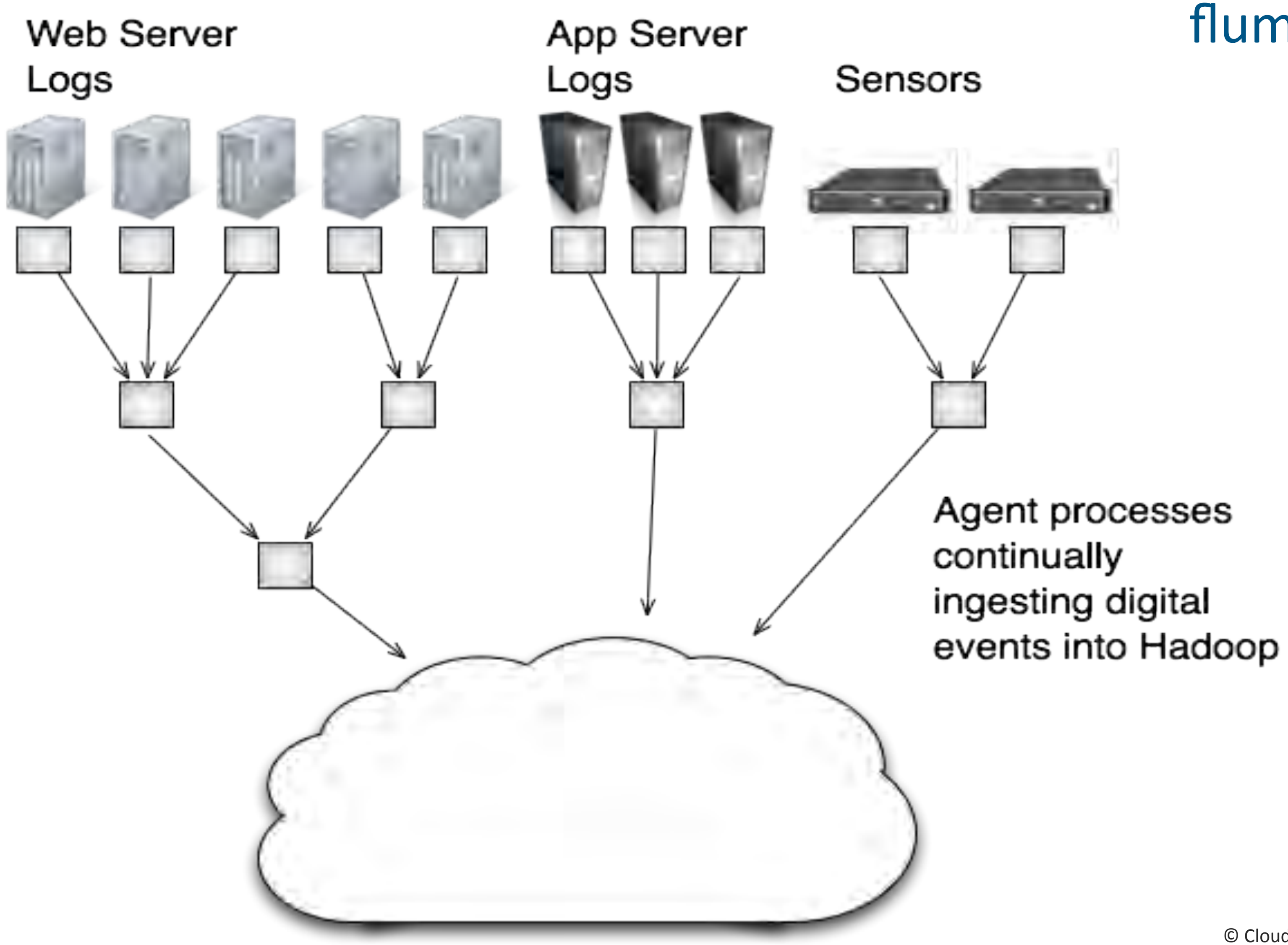
Sqoop: RDBMS to Hadoop and Back

sqoop.apache.org

- Uses MapReduce to run concurrent database queries that extract or insert data

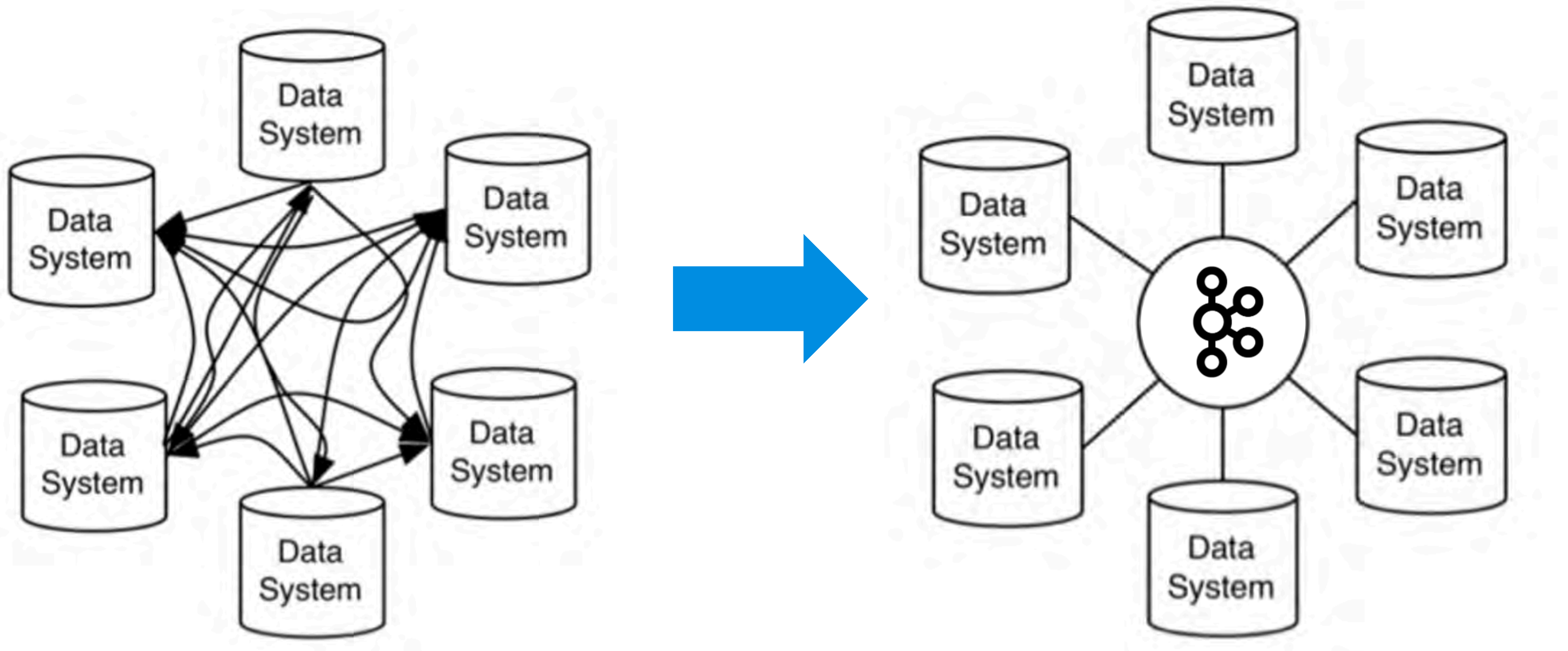
Flume: Ingesting Ongoing Event Data

flume.apache.org



Kafka: General Data Streaming

kafka.apache.org



cloudera

HBase: A NoSQL Database System

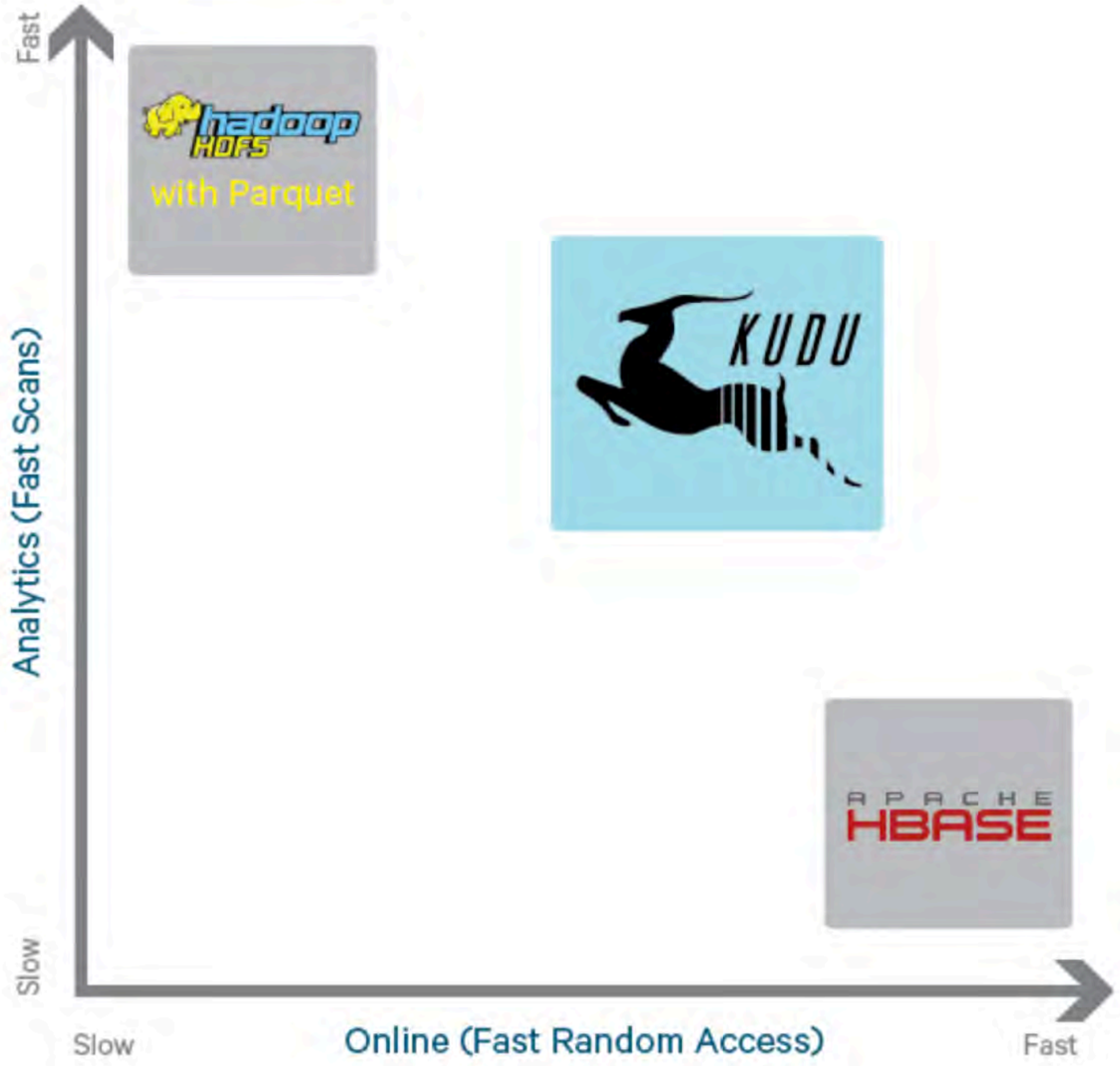
APACHE HBASE

hbase.apache.org

- A scalable key/value store
- Accommodates general binary data
- High volume, high performance access to individual items
- Random reads and writes
- Weaker query language than SQL (put, get, scan, delete)
- Lacks ACID-compliant transactions

Kudu: Scalable storage for structured data

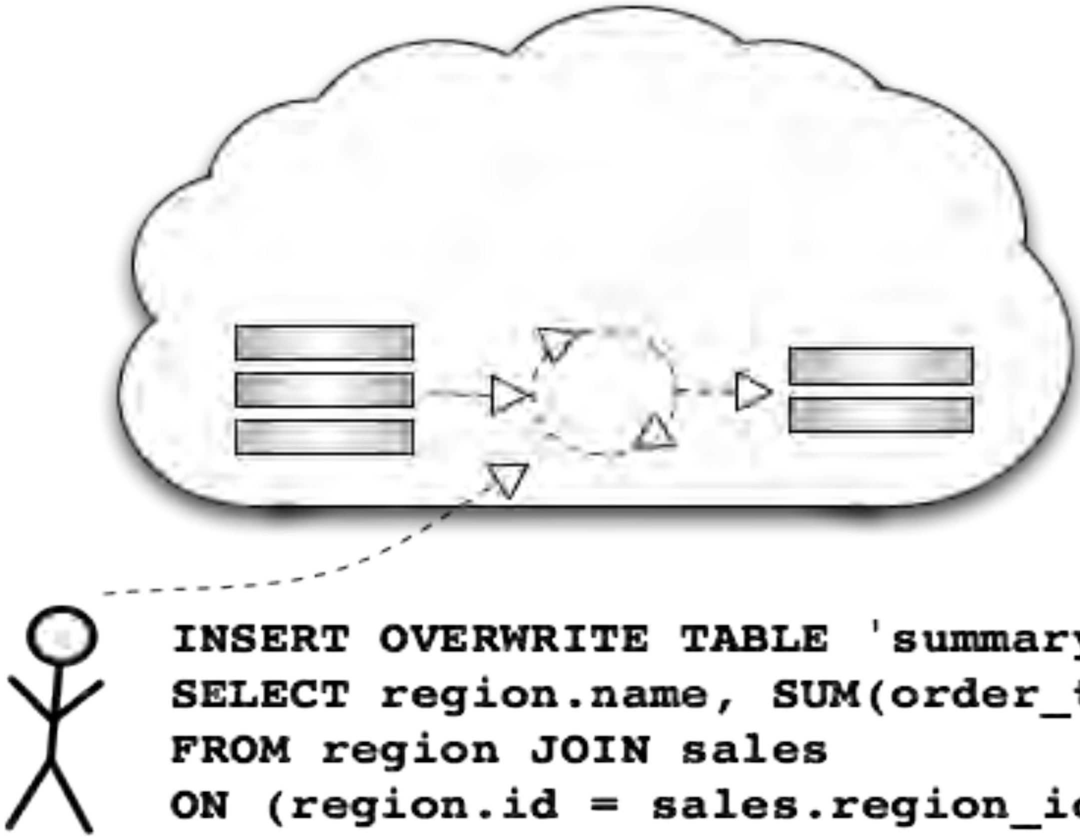
kudu.apache.org



Hive: MapReduce (or Spark) as “SQL”

hive.apache.org

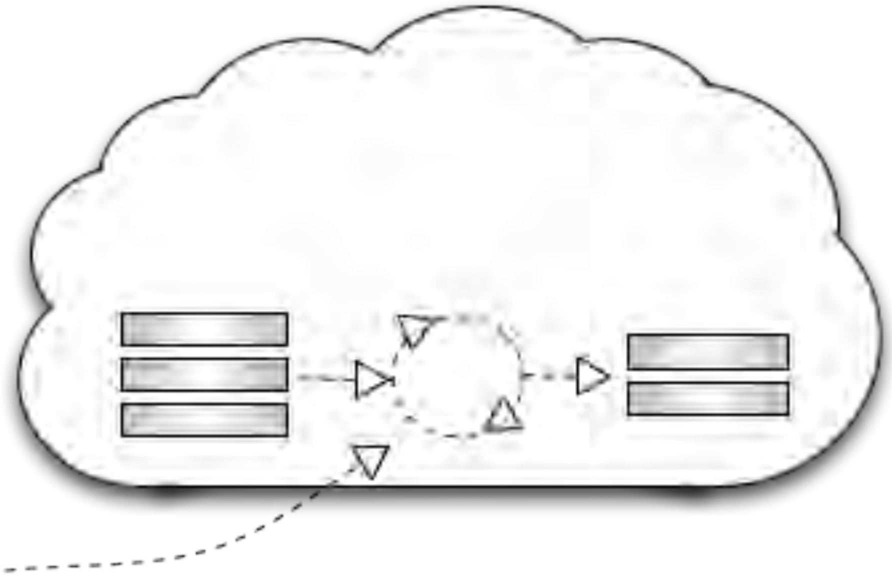
- Familiar language and programming paradigm
- Provides interface to many SQL-compliant tools



```
INSERT OVERWRITE TABLE 'summary'
SELECT region.name, SUM(order_total) region_sales
FROM region JOIN sales
ON (region.id = sales.region_id)
WHERE sales.sale_date > "20121231"
GROUP BY region.name
ORDER BY region_sales DESC;
```

Pig: Another Language for MapReduce (or Spark)

pig.apache.org



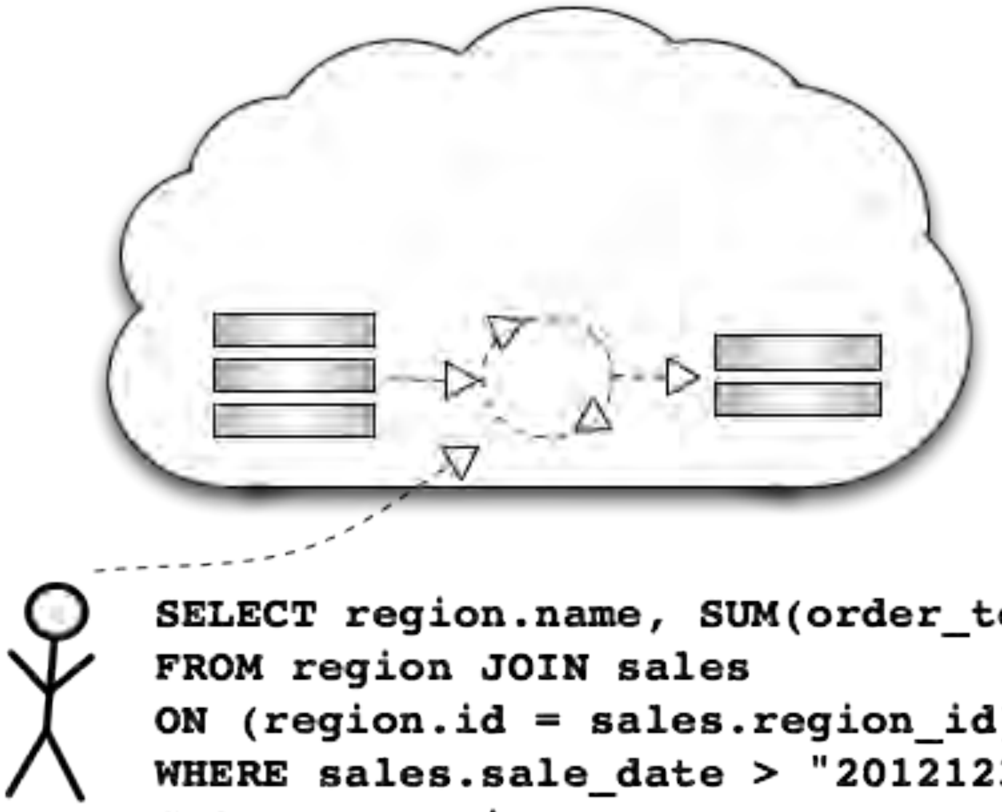
```
sales = LOAD 'sales' AS (orderId:INT, customerName:CHARARRAY,
  sale_date:INT, regionId:INT, orderTotal:FLOAT);
thisYearSales = FILTER sales BY sale_date > 20121231
region = LOAD 'region' AS (id:INT, name:CHARARRAY);
joined = JOIN region BY id, thisYearSales BY regionId;
grouped = GROUP joined BY region:id, region:name;
summary = FOREACH grouped GENERATE group.region:id,
  SUM(joined.orderTotal);
STORE summary INTO 'summary';
```

cloudera

Impala: High Speed Analytics in Hadoop

incubator.apache.org/projects/impala.html

- Purpose-built for high speed analytic queries
- Does not use MapReduce or Spark
- Usually 5 to 30 times faster—sometimes 100 times faster!



```
SELECT region.name, SUM(order_total) region_sales
FROM region JOIN sales
ON (region.id = sales.region_id)
WHERE sales.sale_date > "20121231"
GROUP BY region.name
ORDER BY region_sales DESC;
```

And More

- Serialization and efficient file storage: Avro and Parquet

avro.apache.org

parquet.apache.org

- Workflow: Oozie

oozie.apache.org

And Even More…

- Security: Sentry and Record Service

sentry.apache.org

recordservice.io

- Machine Learning in MapReduce: Mahout

mahout.apache.org

- And …

Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

Spark: An Improvement on MapReduce

spark.apache.org

- Originally a research project at UC Berkeley RAD Lab—later the AMPLab, in 2009
- Addresses some fundamental pain points of MapReduce
- The Spark Streaming subproject of 2012 adds near real-time programming
    - using “micro-batches” as an adaptation of batch programming
    - a capability altogether lacking in Hadoop MapReduce

Similarities of MapReduce and Spark

- Processes massive volumes of data with a scale-out, distributed framework
- The framework provides reliability, even in the face of machine failure
- Programming with stateless functions
- Relies on expensive shuffle to reorganize data for aggregation, joins, sorting
- Still lacks a shared state among all processes
- Can run under YARN to share processing resources

cloudera

Improved API

- First-class APIs in Scala, Java, Python and R
- Data-flow programming paradigm (like Pig)
- Interactive shell—
  great for exploratory work
- Improved support for structured data and
  SQL-like processing

```
sc.textFile(file) \
 .flatMap(lambda s: s.split()) \
 .map(lambda w: (w,1)) \
 .reduceByKey(lambda v1,v2: v1+v2)
 .saveAsTextFile(output)
```

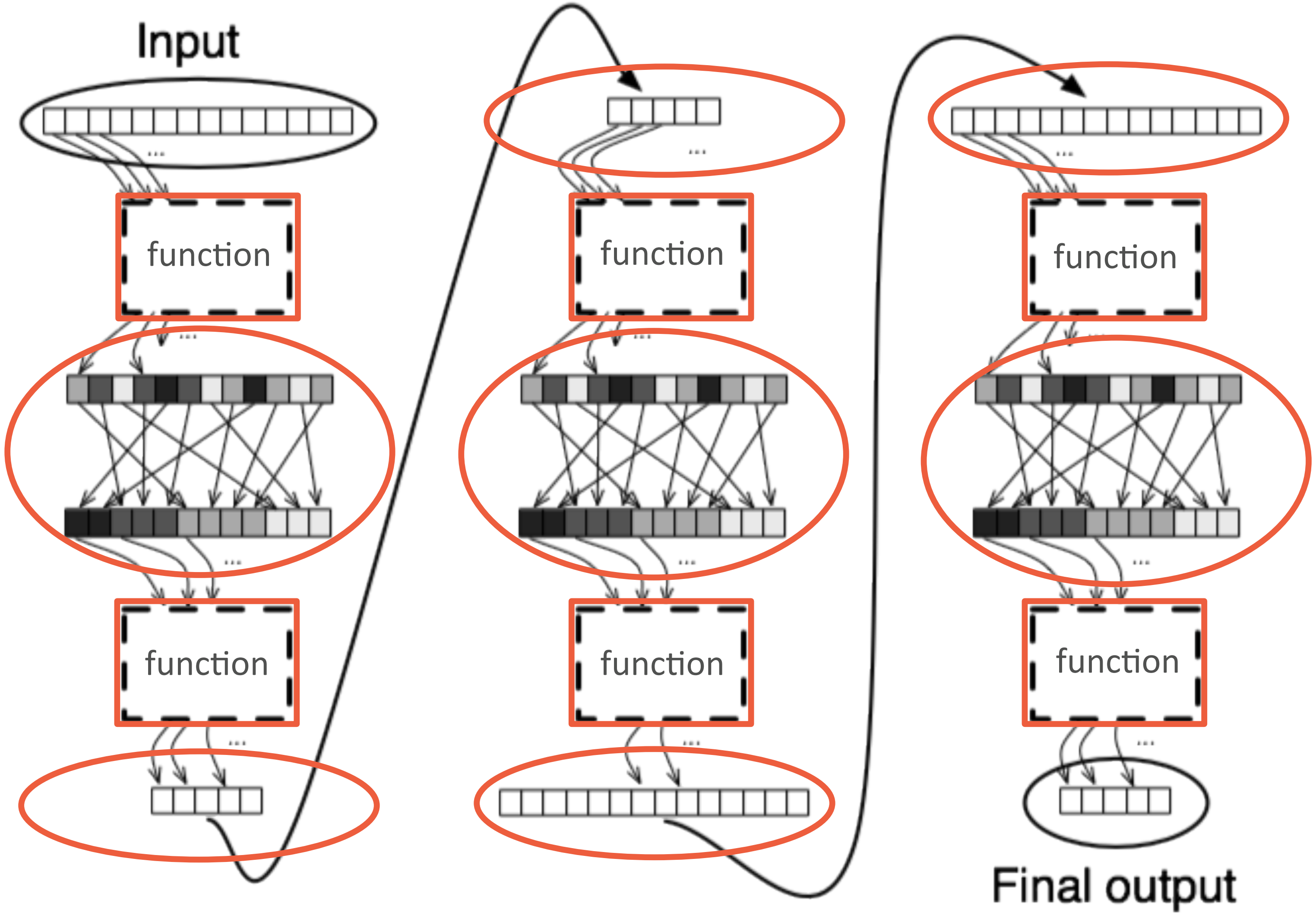
```
public class WordCount {
 public static void main(String[] args) throws
  Job job = new Job();
  job.setJarByClass(WordCount.class);
  job.setJobName("Word Count");
  FileInputFormat.setInputPaths(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  job.setMapperClass(WordMapper.class);
  job.setReducerClass(SumReducer.class);
  job.setMapOutputKeyClass(Text.class);
  job.setMapOutputValueClass(IntWritable.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  boolean success = job.waitForCompletion(true);
  System.exit(success ? 0 : 1);
 }
}

public class WordMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
public void map(LongWritable key, Text value,
Context context) throws IOException, InterruptedException {
  String line = value.toString();
  for (String word : line.split("\\W+")) {
   if (word.length() > 0)
    context.write(new Text(word), new IntWritable(1));
   }
  }
 }
}

public class SumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
 public void reduce(Text key, Iterable<IntWritable>
 values, Context context) throws IOException, InterruptedException {
  int wordCount = 0;
  for (IntWritable value : values) {
   wordCount += value.get();
  }
  context.write(key, new IntWritable(wordCount));
 }
}
```

cloudera

Processing Chains, Improved

Tasks, not new processes (JVMs)

Enhanced caching in memory



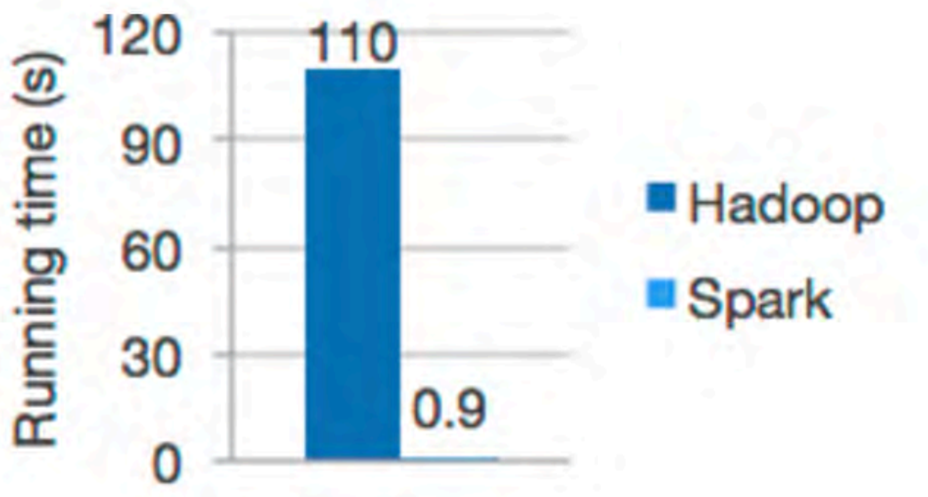
Eliminate I/O

Reduce I/O

Eliminate I/O

Spark MLlib: Machine Learning in Spark

- Subproject of Spark
- Effectively replaces Mahout for machine learning in Hadoop clusters
- From spark.apache.org, the front page:



Logistic regression in Hadoop and Spark

But just be clear what you mean by "Hadoop"!

Commercial Message # 1

Big Ecosystem

oozie.apache.org

Complete Big Data Platform

- Cloudera Manager can
  - install, monitor, manage, upgrade
    a coherent bundle of these projects and more
- Cloudera Director can
  - easily configure and deploy this platform on cloud services
    from Amazon, Google, or Microsoft
- !!!

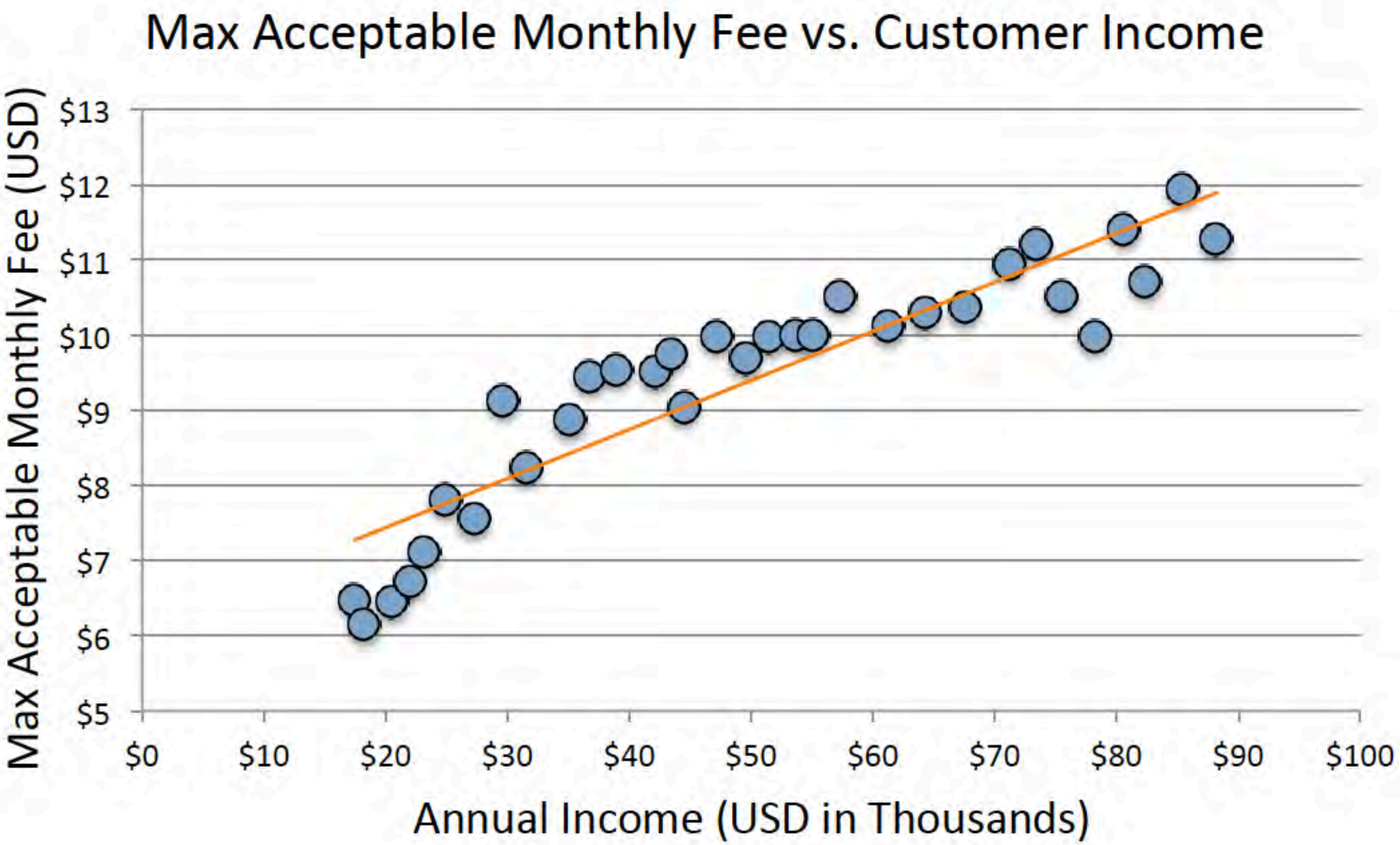
Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

cloudera

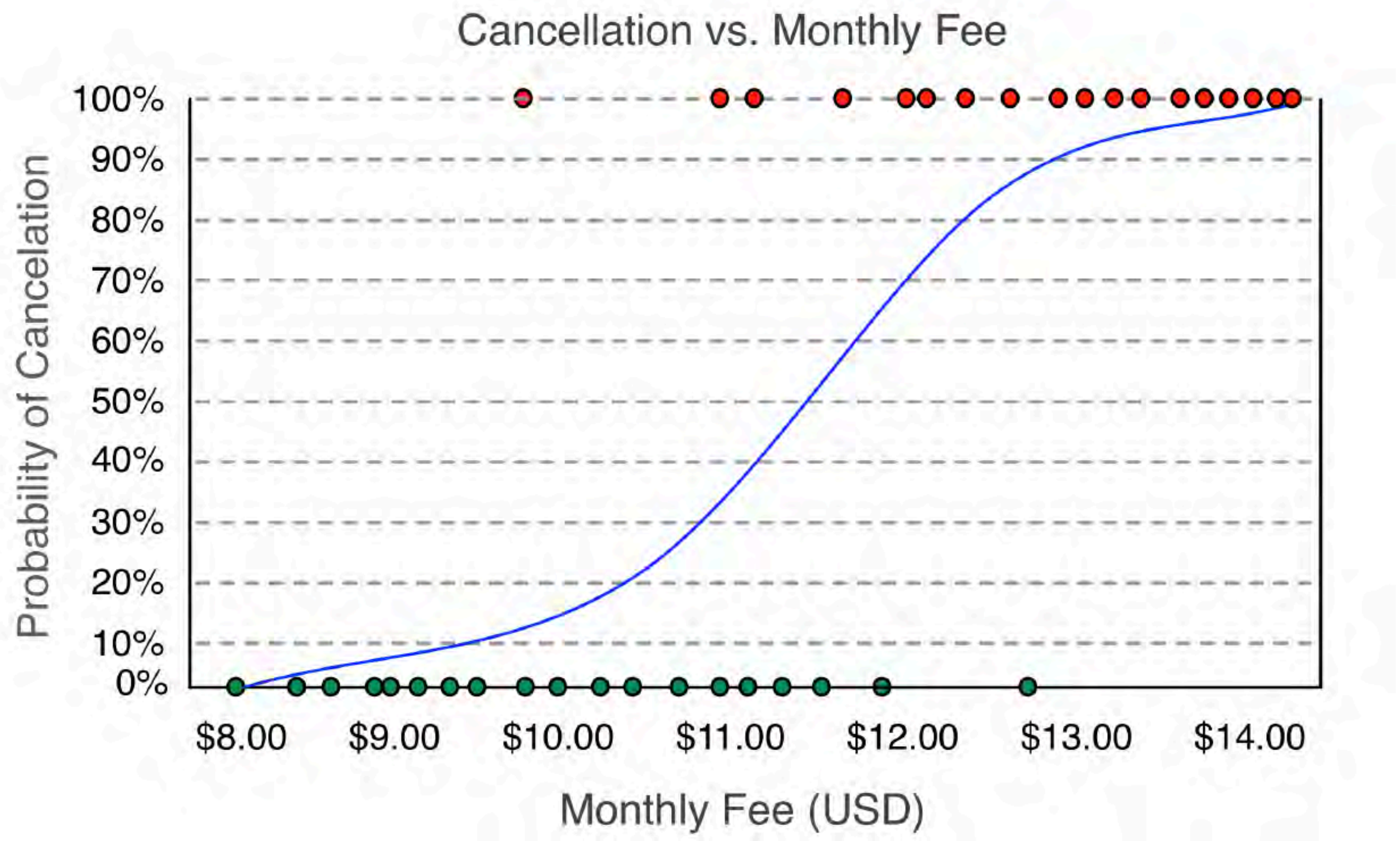
Machine Learning Algorithms

- Supervised Learning:
  - Start with correctly labeled records, and learn to estimate or predict labels for new records
  - Continuous labels: Regression
  - Discrete labels: Logistic Regression, Classifiers
- Unsupervised Learning:
  - Start with unlabeled records, try to tease patterns (labels) out of the data
  - There is not a single “correct” answer for labeling
  - Continuous labels: Collaborative Filters (Recommenders)
  - Discrete labels: Clustering

Linear Regression: Supervised Learning of a Continuous Label



Logistic Regression: Supervised Learning of a Binary Label



cloudera

Classifiers:
Supervised Learning of Discrete Labels

Training: Cat

Training: Table

Scoring: ???

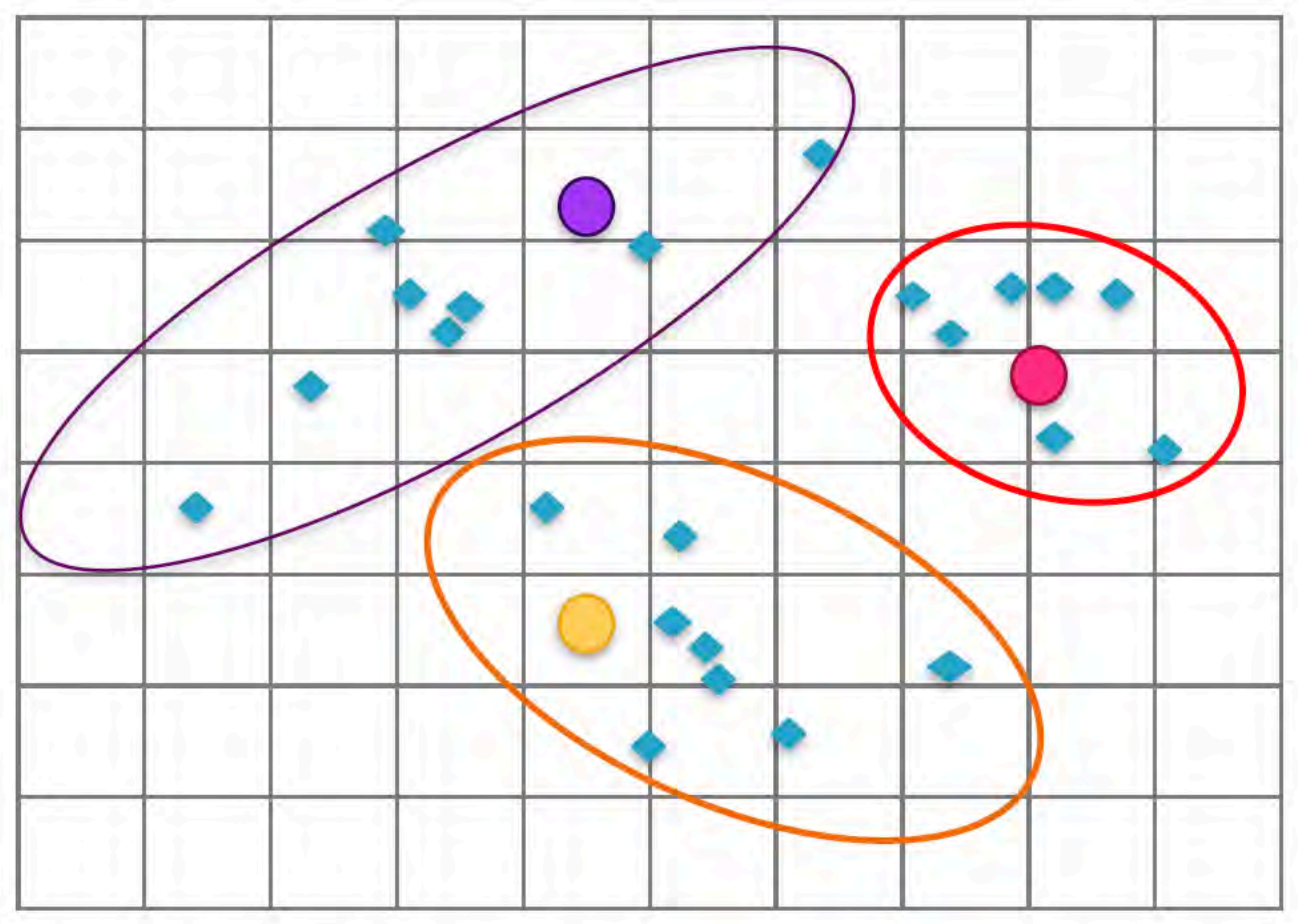
cloudera

Collaborative Filters (Recommenders): Unsupervised Learning of Continuous Labels

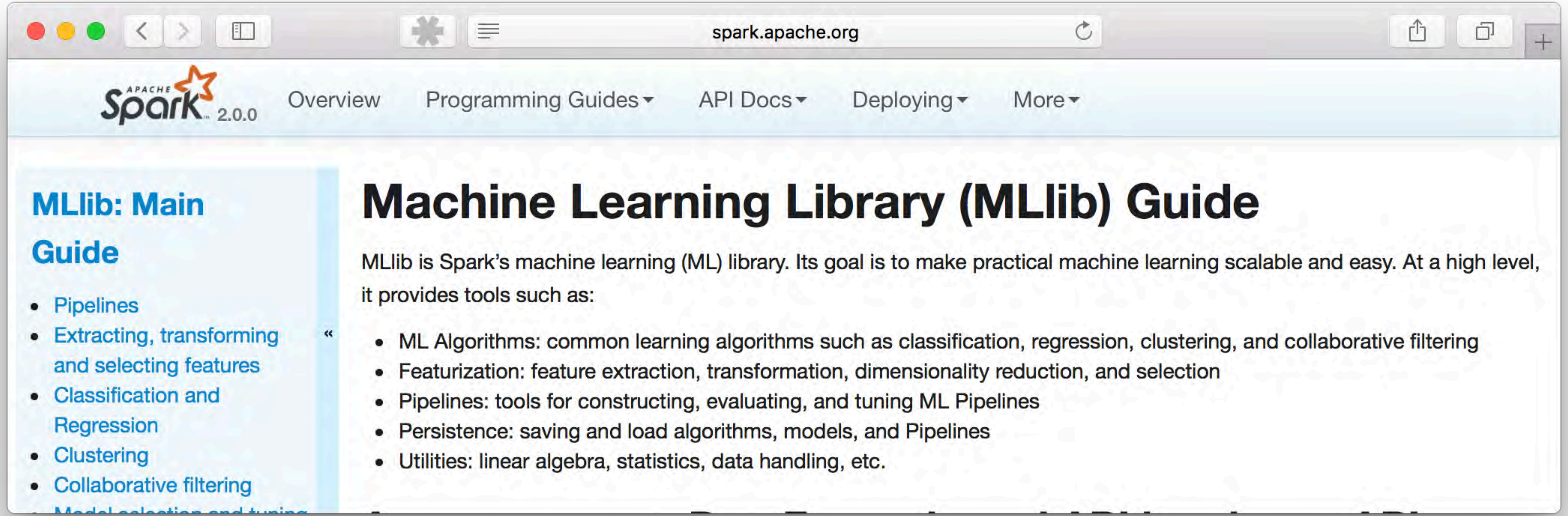
	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

cloudera

Clustering:
Unsupervised Learning of Discrete Labels



Spark MLlib: Machine Learning on Hadoop



cloudera

Short and Sweet
Hadoop
What About Spark?
Machine Learning
The Future

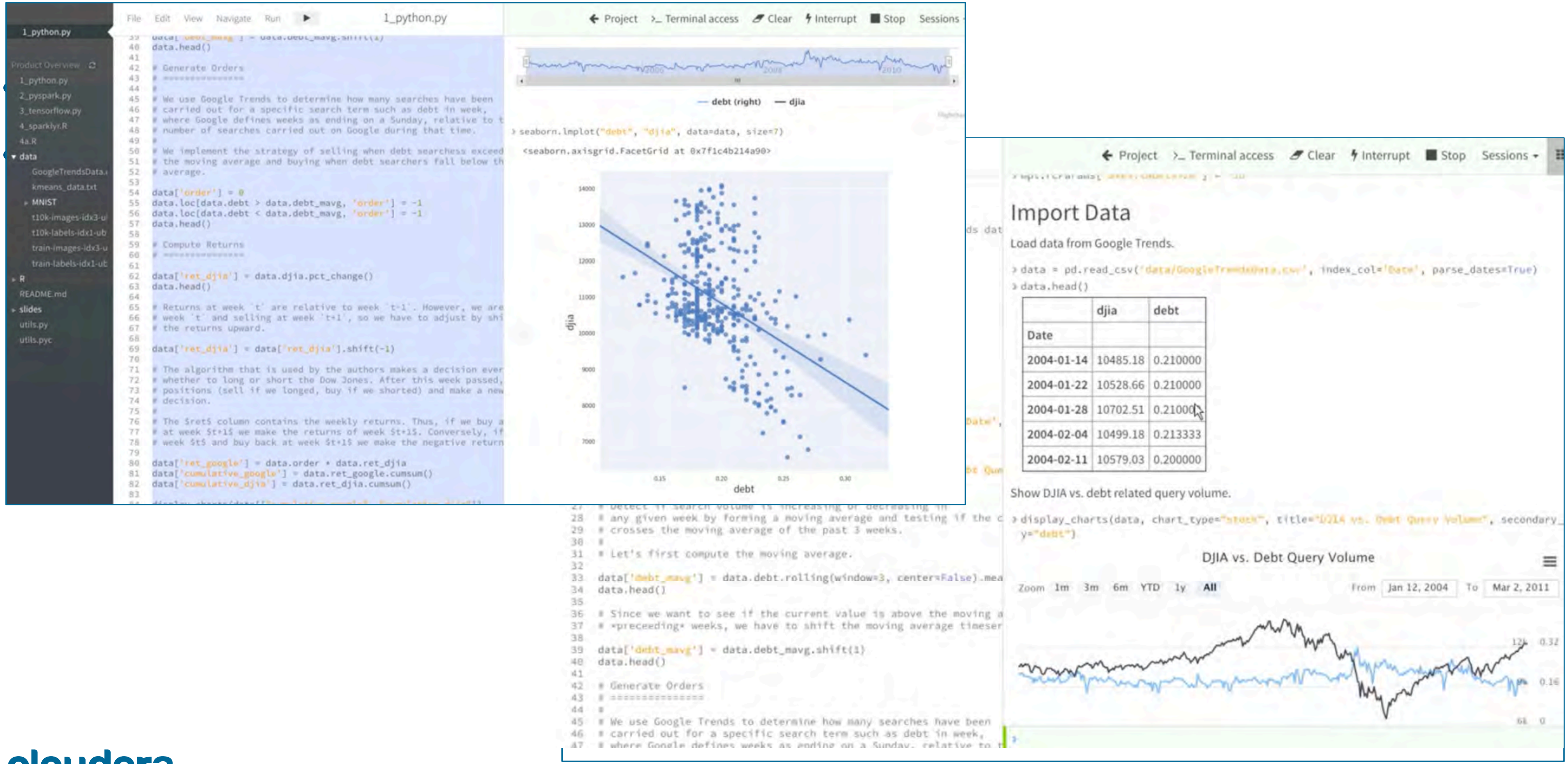
cloudera

Commercial Message # 2

More DS Teams in the Organization

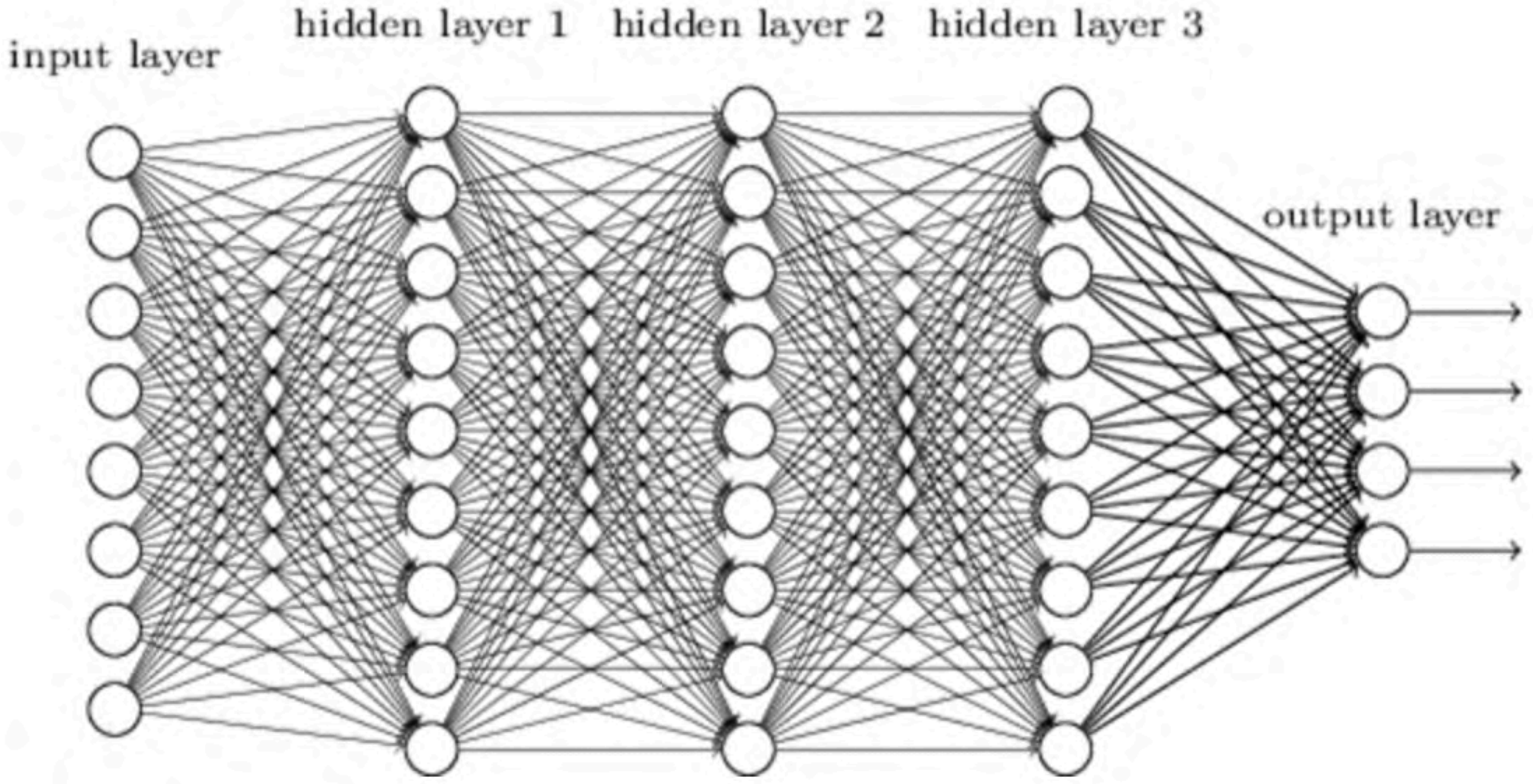
- Collaboration, repeatability within teams
- Differing security requirements
- Different preferred programing languages: Python, R, Scala
- Different software libraries: Pandas, H2O, etc.
- Even different versions of software

Cloudera Data Science Workbench



cloudera

Deep Learning



Deep Learning on Hadoop

- Deep Learning refers to a category of classifier algorithms, mostly invented in 2006.
- Spark MLlib does not have any direct implementation of DL.
- There are several additional projects that can fit DL onto Spark/Hadoop:
  - BigDL
  - Caffe
  - TensorFlow
  - DL4J

The Road—or Runway(!)—Ahead

- It is a truism that organizations today have valuable insights hidden in their data that are waiting to be uncovered.
- 90% of all data that will exist in 2020 has yet to be created.
- Open source is here to stay.
- Hadoop as a data science platform is evolving, and its use is growing exponentially.

cloudera

Thank you

glynn@cloudera.com