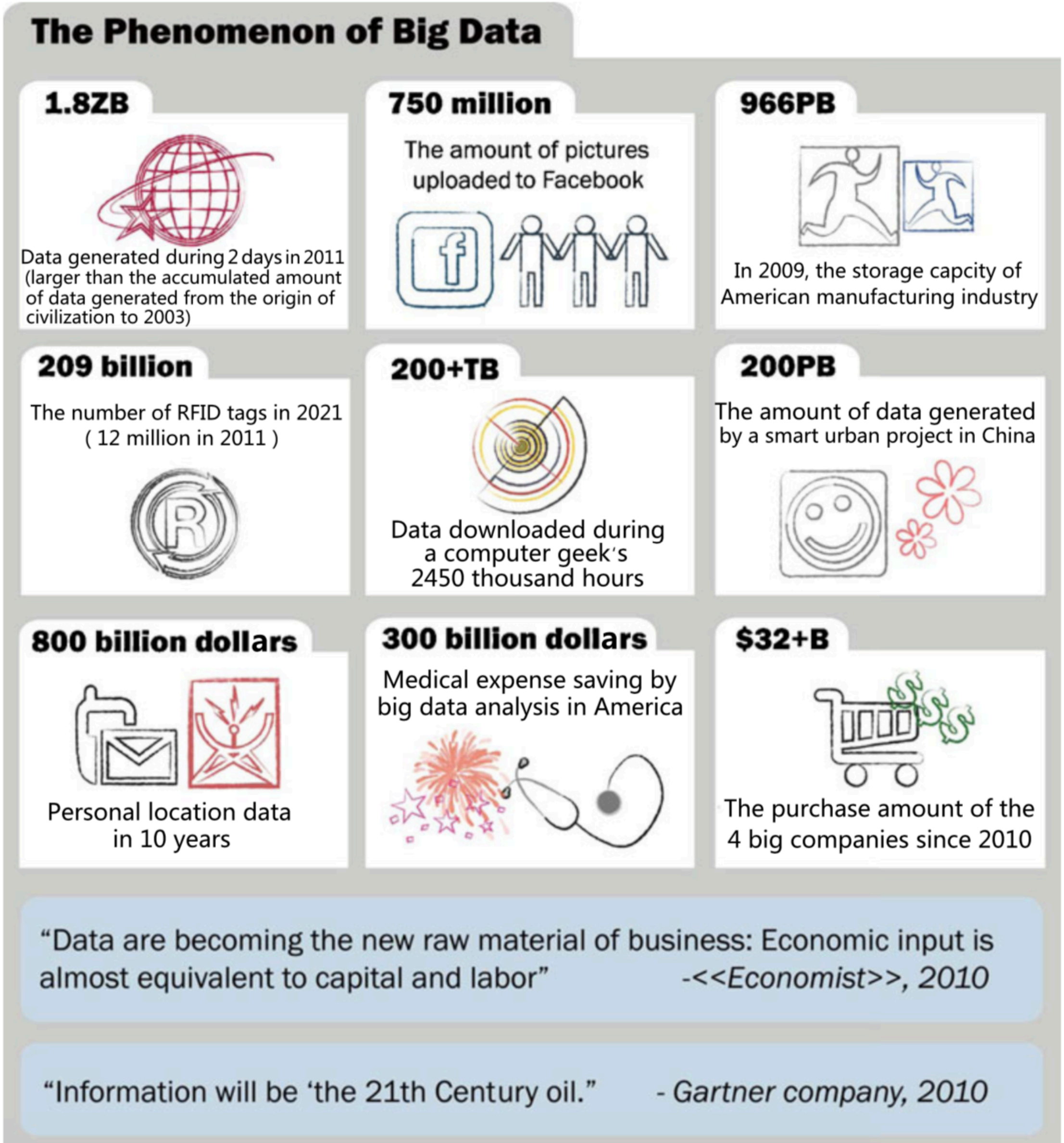
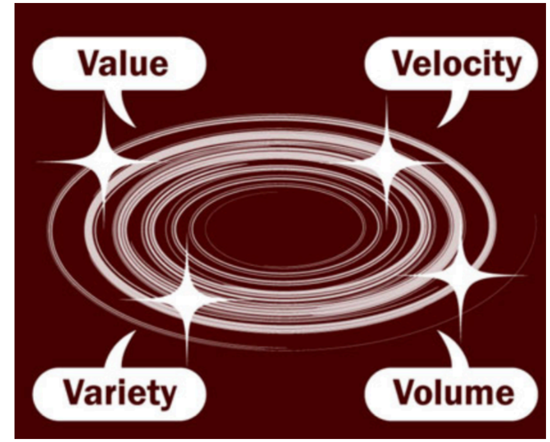
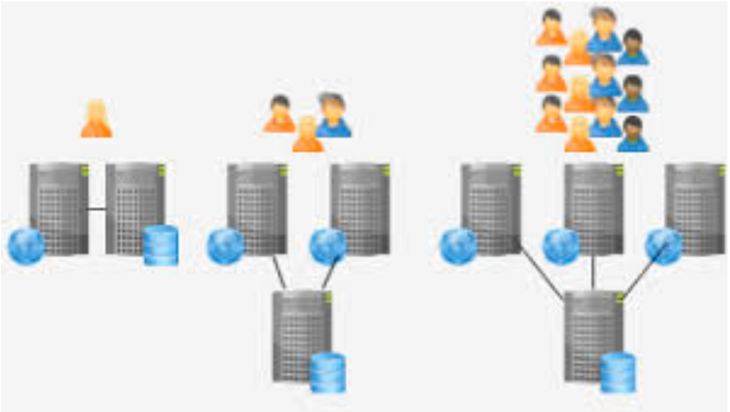
Big data

Relevant to all predictive analytics, including text

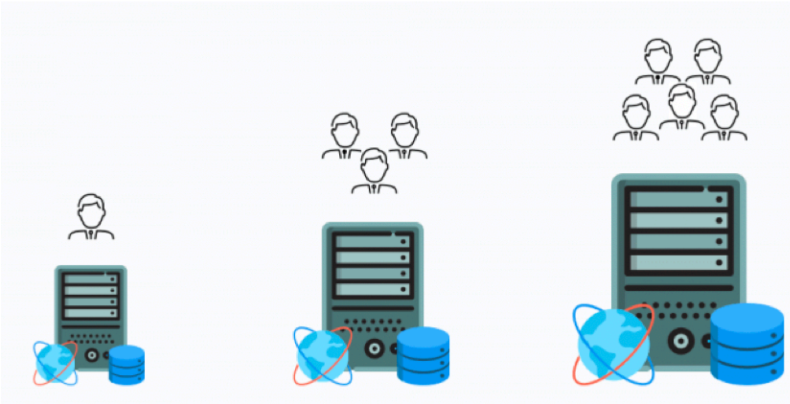

The Phenomenon of Big Data
1.8ZB
Data generated during 2 days in 2011 (larger than the accumulated amount of data generated from the origin of civilization to 2003)
750 million
The amount of pictures uploaded to Facebook
966PB
In 2009, the storage capcity of American manufacturing industry
209 billion
The number of RFID tags in 2021 (12 million in 2011)
200+TB
Data downloaded during a computer geek's 2450 thousand hours
200PB
The amount of data generated by a smart urban project in China
800 billion dollars
Personal location data in 10 years
300 billion dollars
Medical expense saving by big data analysis in America
$32+B
The purchase amount of the 4 big companies since 2010
"Data are becoming the new raw material of business: Economic input is almost equivalent to capital and labor" -<<Economist>>, 2010
"Information will be 'the 21th Century oil." - Gartner company, 2010


Value
Velocity
Variety
Volume

Horizontal vs. Vertical Scaling

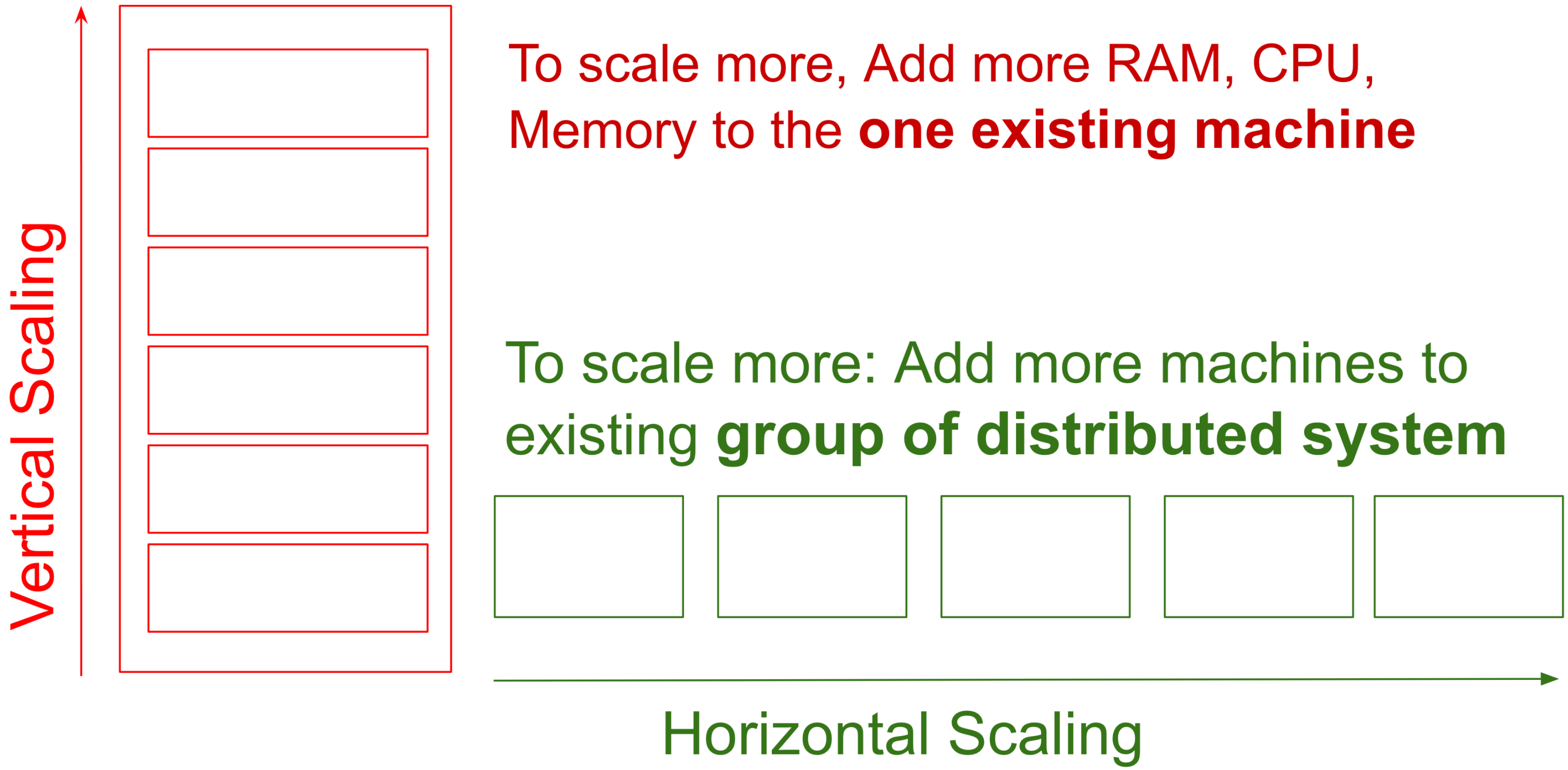


Horizontal Scaling



Vertical Scaling

Put differently



More precisely,

- **Horizontal scaling:** Horizontal scaling involves distributing the workload across many servers which may be even commodity machines. It is also known as "scale out" , where multiple independent machines are added together in order to improve the processing capability. Typically, multiple instances of the operating system are running on separate machines.
- **Vertical scaling:** Vertical Scaling involves installing more processors, more memory and faster hardware, typically, within a single server. It is also known as "scale up" and it usually involves a single instance of an operating system.

Examples

- Horizontal scaling: https://simplicable.com/new/horizontal-scale
  - Load balancing, cloud databases, service architecture…
  - Also, peer-to-peer networks, MapReduce/Hadoop…
- Vertical scaling: relational databases mostly use it (e.g., Oracle, but also MySQL and Amazon RDS), also advanced kinds of neural network training using lots of GPUs, HPC clusters, multicore CPUS, FPGAs…

A comparison of advantages and drawbacks of horizontal and vertical scaling

Scaling	Advantages	Drawbacks
Horizontal scaling	➔ Increases performance in small steps as needed ➔ Financial investment to upgrade is relatively less ➔ Can scale out the system as much as needed	➔ Software has to handle all the data distribution and parallel processing complexities ➔ Limited number of software are available that can take advantage of horizontal scaling
Vertical scaling	➔ Most of the software can easily take advantage of vertical scaling ➔ Easy to manage and install hardware within a single machine	➔ Requires substantial financial investment ➔ System has to be more powerful to handle future workloads and initially the additional performance in not fully utilized ➔ It is not possible to scale up vertically after a certain limit

Source: *A survey on platforms for big data analytics*, Singh and Reddy

Case study: MApReduce

MapReduce and Hadoop

- MapReduce is a ‘framework’ for embarrassingly parallel programming
- Popularized in the article by Google researchers Dean and Ghemawat (https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf)
  - Highly recommended reading
- Implemented as Apache Hadoop, available on all cloud platforms!

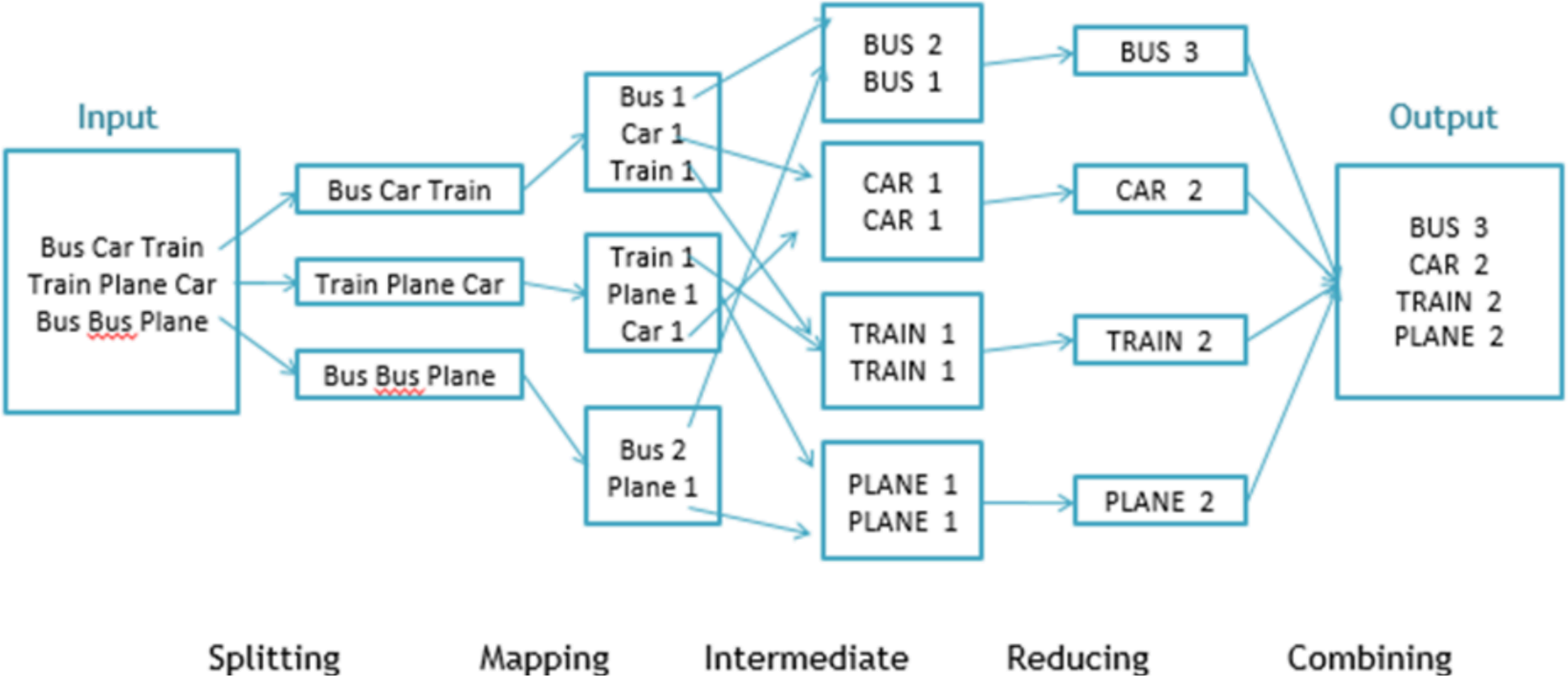
Best illustrated through an example

- (But to truly understand, you must go through the first three pages of the linked paper)
- **Example problem:** Suppose we wanted to count the number of occurrences of each word in a large collection of documents

MapReduce ‘hello world’: word counting in large corpus

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```



CLASSIC MAPREDUCE 101 EXAMPLE: WORDCOUNT

https://dzone.com/articles/word-count-hello-word-program-in-mapreduce

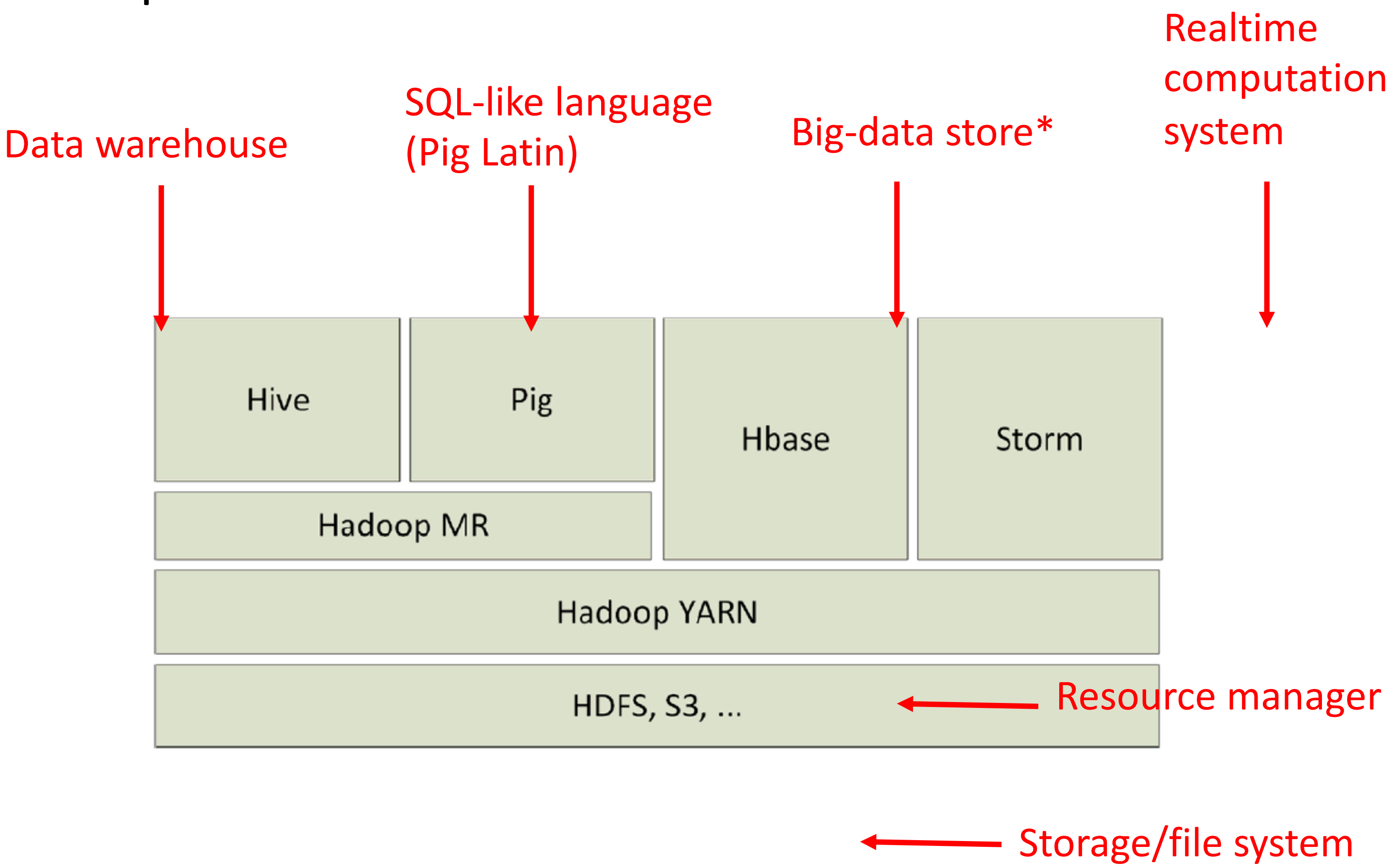
Notes

- **Splitting** –splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
- **Mapping** – takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair)
- **Intermediate splitting/partitioning:** from https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Partitioner : Partitioner controls the partitioning of the keys of the intermediate map-outputs. The key (or a subset of the key) is used to derive the partition, typically by a *hash function*. The total number of partitions is the same as the number of reduce tasks for the job. Hence this controls which of the m reduce tasks the intermediate key (and hence the record) is sent to for reduction.
- **Reduce** – group by key
- **Combining** – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

Disadvantages

- Can be very complex (this can, and does, impede adoption in more conservative or non-technical organizations)
- Only suited for certain kinds of problems (shared-nothing parallelism)
  - Can you think of problems that it's not suited to?
- Slower than a vertical cluster because of distribution of data, latency and processing speed
  - Apache Spark was basically invented to deal with this problem
- Hadoop only ensures that the data job is complete, but it's unable to guarantee when the job will be complete
- Hadoop can encounter security issues (mainly because it's written in Java)
  - Kerberos authentication supported by Hadoop is hard to manage
- Real-time and iterative processing are not feasible

Hadoop ‘stack’



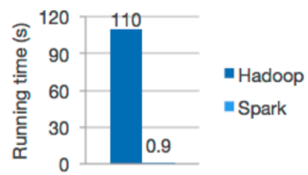
*Modeled after Google BigTable: https://research.google/pubs/pub27898/

Apache Spark™ is a unified analytics engine for large-scale data processing.

Speed

Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Logistic regression in Hadoop and Spark

Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

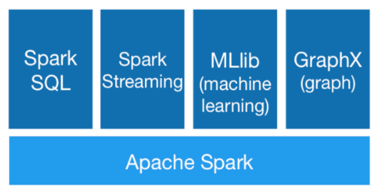
```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



Runs Everywhere

Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.

You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. Access data in HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources.

Latest News

Spark 3.0.0 released (Jun 18, 2020)

Spark+AI Summit (June 22-25th, 2020, VIRTUAL) agenda posted (Jun 15, 2020)

Spark 2.4.6 released (Jun 05, 2020)

Spark 2.4.5 released (Feb 08, 2020)

Archive

APACHECON @Home Sep 29 - Oct 1, 2020

Download Spark

Built-in Libraries:

- SQL and DataFrames
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)

Third-Party Projects

Original author(s)	Matei Zaharia
Developer(s)	Apache Spark
Initial release	May 26, 2014; 6 years ago
Stable release	3.0.0 / June 18, 2020; 2 months ago
Repository	Spark Repository
Written in	Scala[1]
Operating system	Microsoft Windows, macOS, Linux
Available in	Scala, Java, SQL, Python, R
Type	Data analytics, machine learning algorithms
License	Apache License 2.0
Website	spark.apache.org