Outils informatiques pour le Big Data en astronomie

Lionel Fillatre

Université Nice Sophia Antipolis

Polytech Nice Sophia

Laboratoire I3S

École d'été thématique CNRS BasMatI

3 juin 2015

Outlines

- What is the Big Data (including Hadoop Ecosystem)
- HDFS (Hadoop Distributed File System)
- What is MapReduce?
- Image Coaddition with MapReduce
- What is NoSQL?
- What is Pig?
- What is Hive?
- What is Spark?
- Conclusion

What is the Big Data

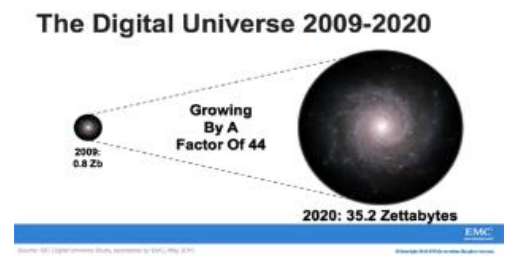
Big Data Definition

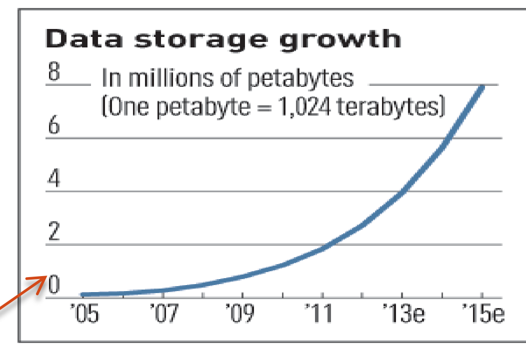
- No single standard definition…

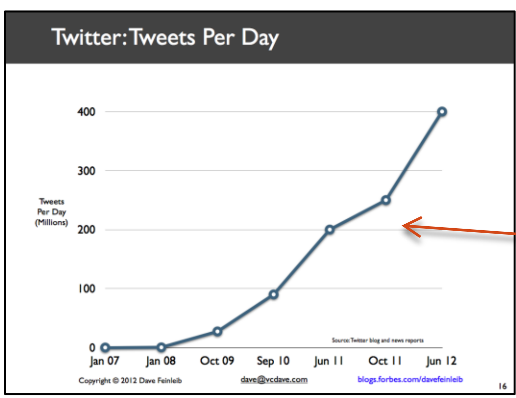
"***Big Data***" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…

Characteristics of Big Data:
1-Scale (Volume)

- **Data Volume**
  - 44x increase from 2009 to 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially



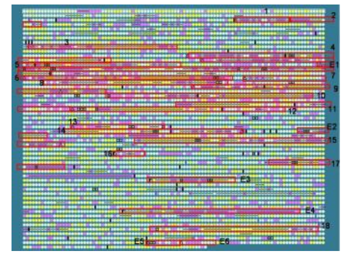


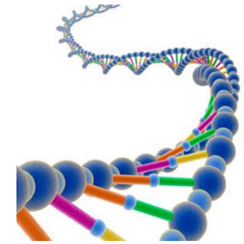


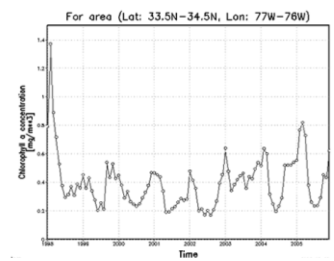
Exponential increase in collected/generated data

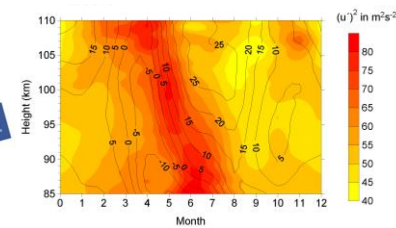
Characteristics of Big Data:
2-Complexity (Variety)

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be generating/collecting many types of data









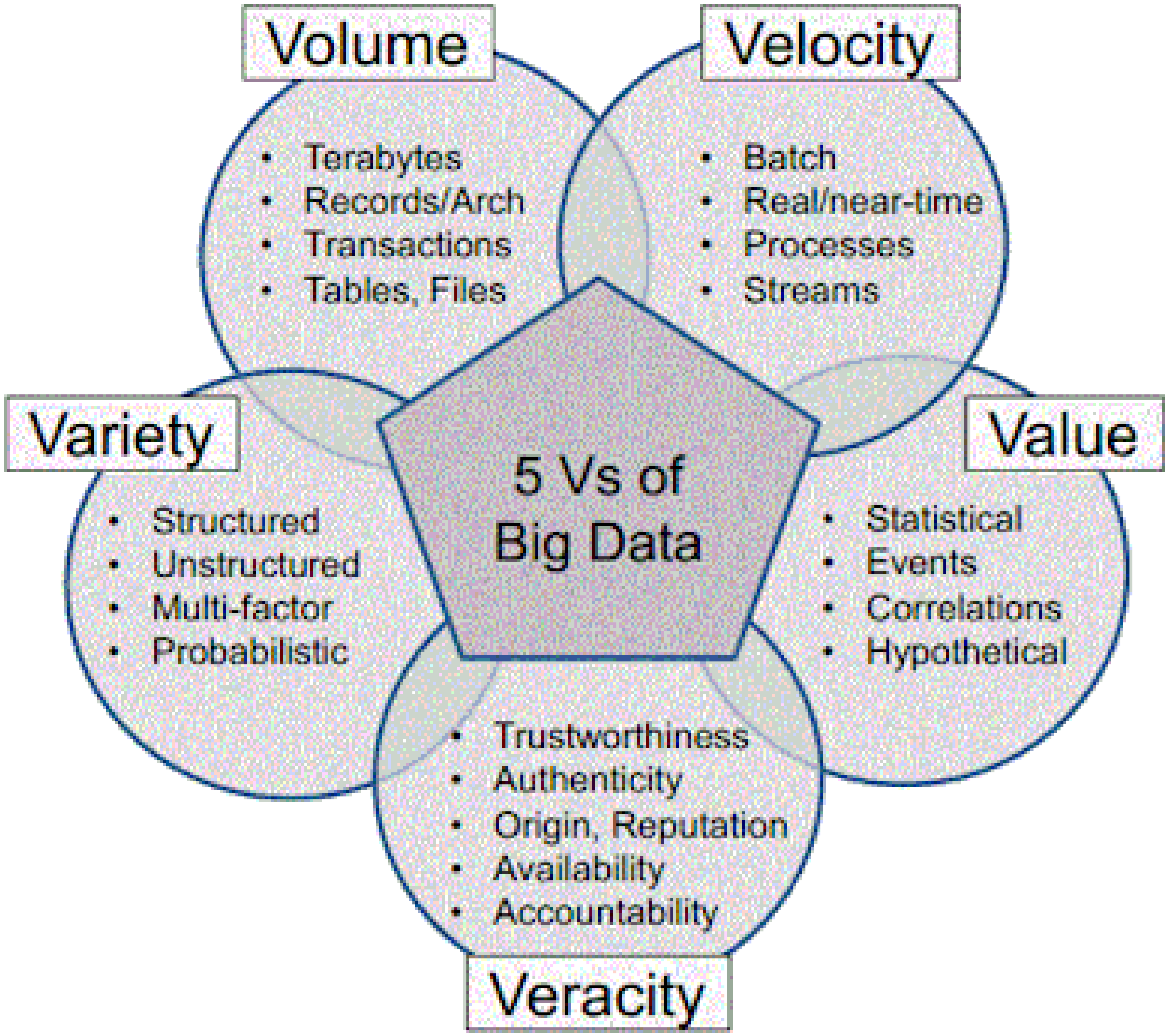
To extract knowledge
➔ all these types of data need to be linked together

Characteristics of Big Data:
3-Speed (Velocity)

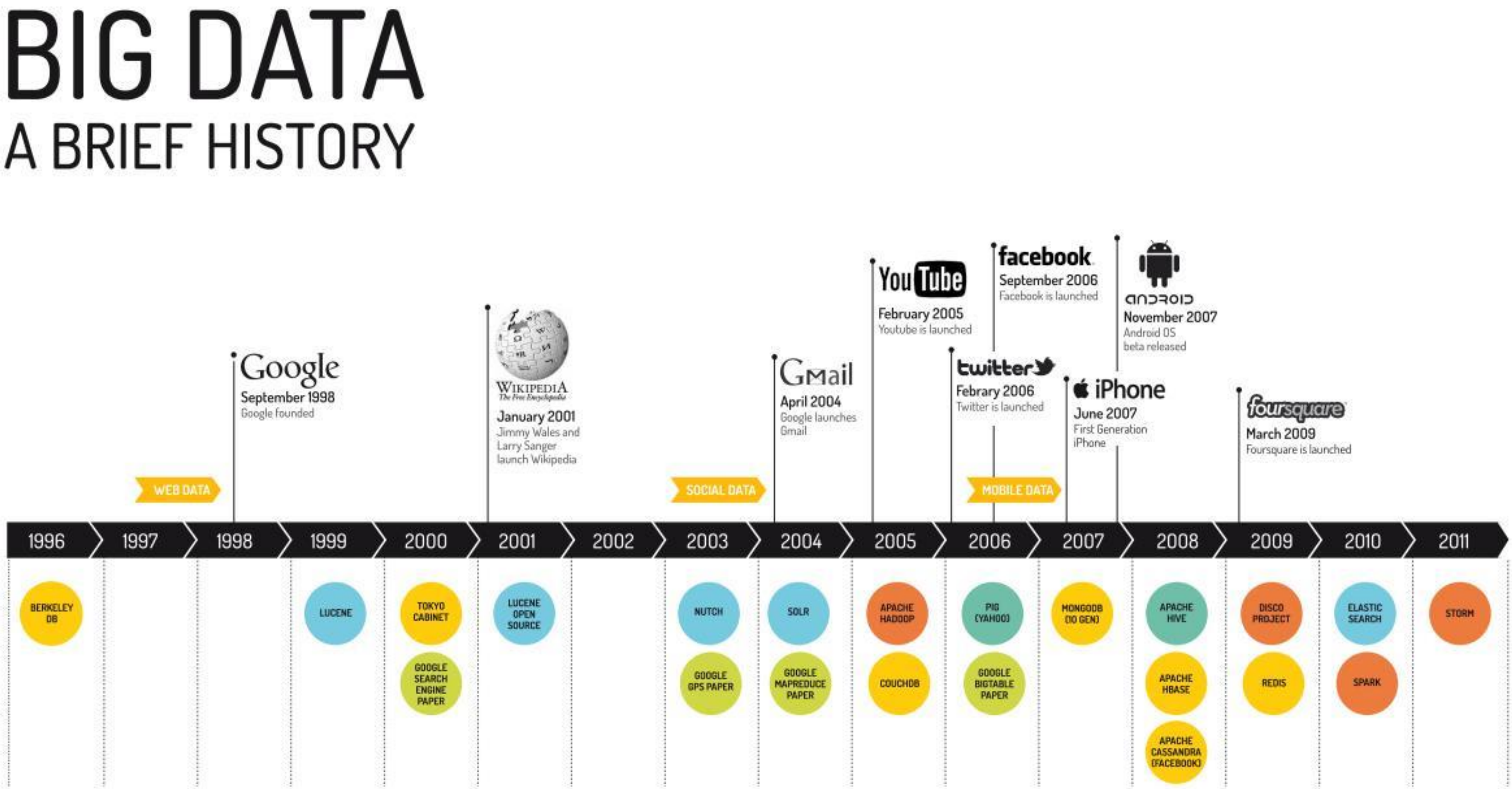
- Data is generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

Some Make it 5V's



What technology for Big Data?



Big Data Landscape

Vertical Apps
PREDICTIVE POLICING, bloomreach, MYRRIX

Log Data Apps
splunk>, loggly, sumologic

Ad/Media Apps
rocketfuel, collective[i], bluefin, Recorded Future, LuckySort, Media Science, TURN, DataXu

Business Intelligence
ORACLE | Hyperion, SAP, Business Objects, RJMetrics, Microsoft | Business Intelligence, IBM, COGNOS, birst, MicroStrategy, Autonomy, QlikView, bime, DOMO, Chart.io, GoodData

Analytics and Visualization
tableau, Palantir, OPERA, metaLayer, METAMARKETS, dataspora, centrifuge, TERADATA ASTER, sas, TIBCO, KARMASPHERE, panopticon, pentaho, Datameer, platfora, ClearStory, CIRRO, alteryx, visual.ly, AYATA

Data As A Service
factual., kaggle, knoema, GNIP, DATASIFT, Windows Azure Marketplace, INRIX, LexisNexis, SPACE CURVE, LOCATE

Analytics Infrastructure
Hortonworks, VERTICA, MAPR, cloudera, INFOBRIGHT, PARACCEL, EMC², GREENPLUM, NETEZZA, kognitio, DATASTAX, EXASOL, calpont

Operational Infrastructure
COUCHBASE, 10gen, TERADATA, HADAPT, TERRACOTTA, VoltDB, MarkLogic, INFORMATICA

Infrastructure As A Service
amazon web services, Windows Azure, infochimps, Google BigQuery

Structured Databases
ORACLE, MySQL, SQL Server, PostgreSQL, IBM DB2, SYBASE, memsql

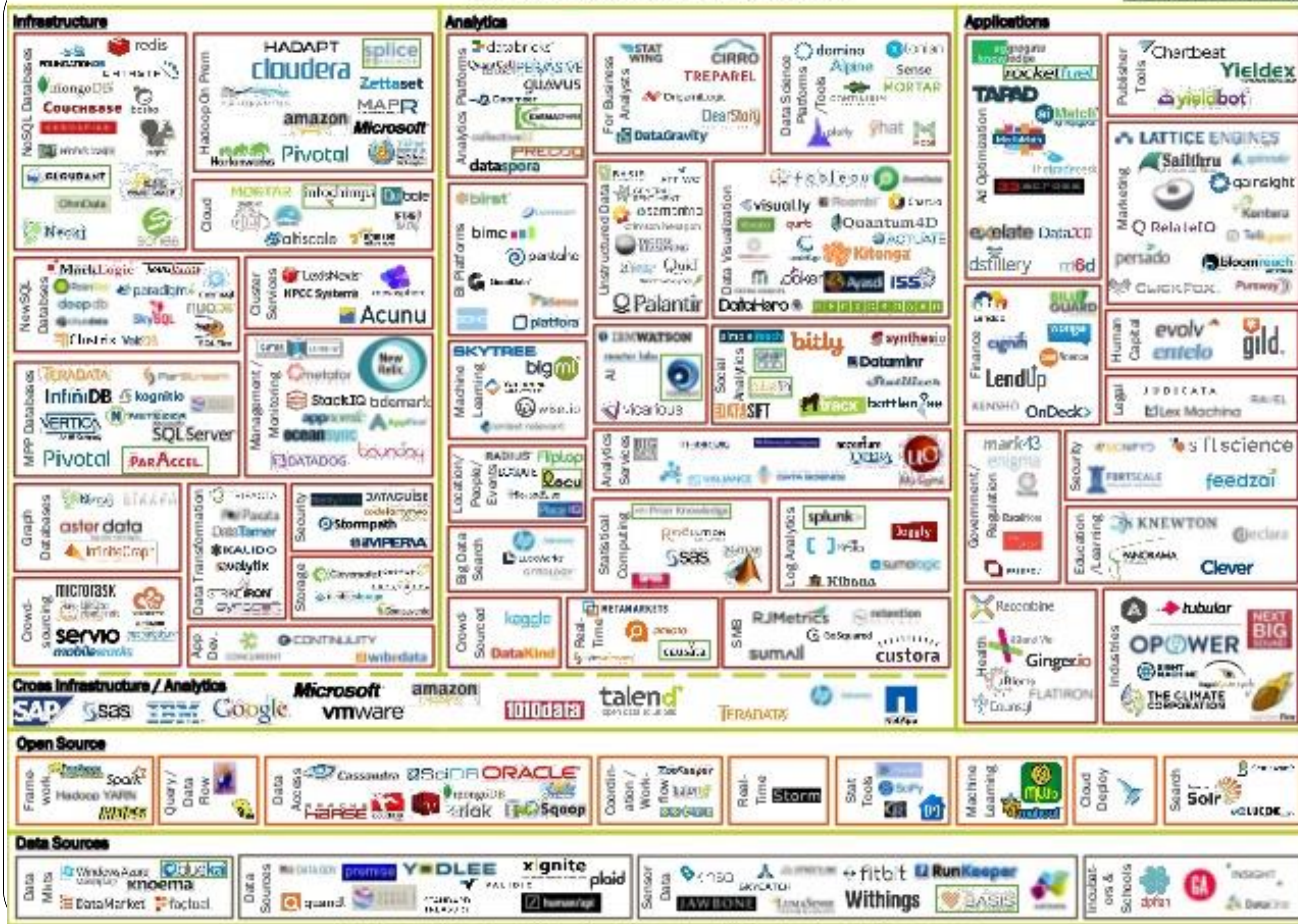
Technologies
hadoop, hadoop MapReduce, mahout, APACHE HBASE, Cassandra

 dave@vcdave.com blogs.forbes.com/davefeinleib

Big Data Landscape (Version 2.0)

© Matt Turck (@mattturck) and ShivonZilis (@shivonz) Bloomberg Ventures

BIG DATA LANDSCAPE, VERSION 3.0

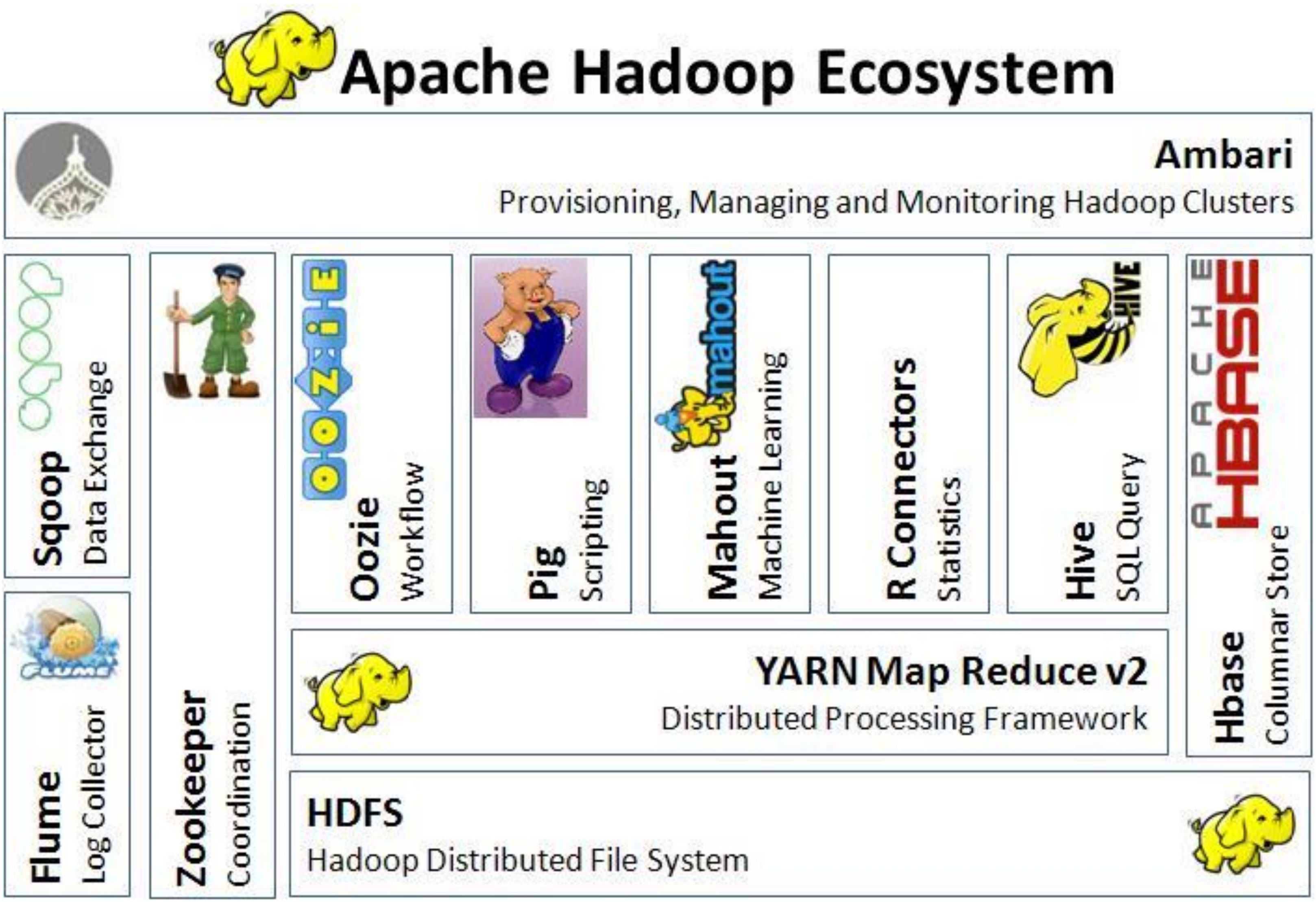


© Matt Turck (@mattturck), Sutian Dong (@sutiandong) & FirstMark Capital (@firstmarkcap)

Hadoop Origins

- Apache Hadoop is a framework that allows for the distributed processing of large data sets accross clusters of commodity computers using a simple programming model.
- Hadoop is an open-source implementation of Google MapReduce and Google File System (GFS).
- Hadoop fulfills need of common infrastructure:
  - Efficient, reliable, easy to use,
  - Open Source, Apache License.

Hadoop Ecosystem (main elements)



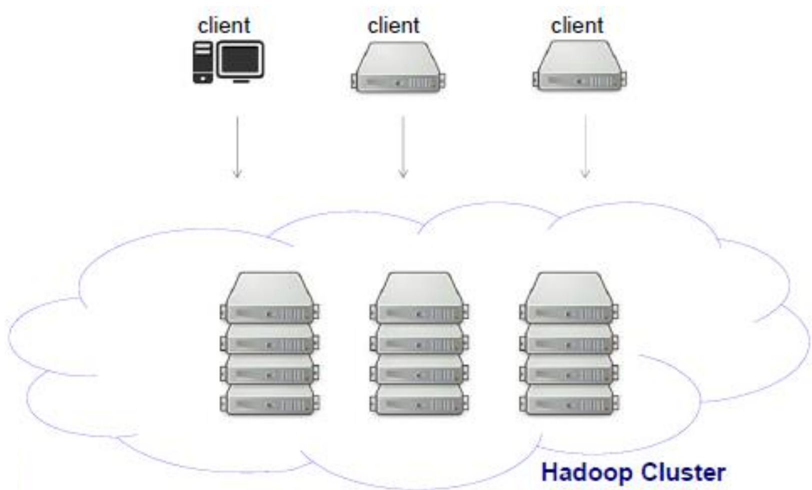
Data Storage

- **Storage capacity has grown exponentially but read speed has not kept up**
  - 1990:
    - Store 1,400 MB
    - Transfer speed of 4.5MB/s
    - Read the entire drive in ~ 5 minutes
  - 2010:
    - Store 1 TB
    - Transfer speed of 100MB/s
    - Read the entire drive in ~ 3 hours
- **Hadoop - 100 drives working at the same time can read 1TB of data in 2 minutes**

Hadoop Cluster

- **A set of "cheap" commodity hardware**
  - No need for super-computers, use commodity unreliable hardware
  - Not desktops
- **Networked together**
- **May reside in the same location**

– Set of servers in a set of racks in a data center

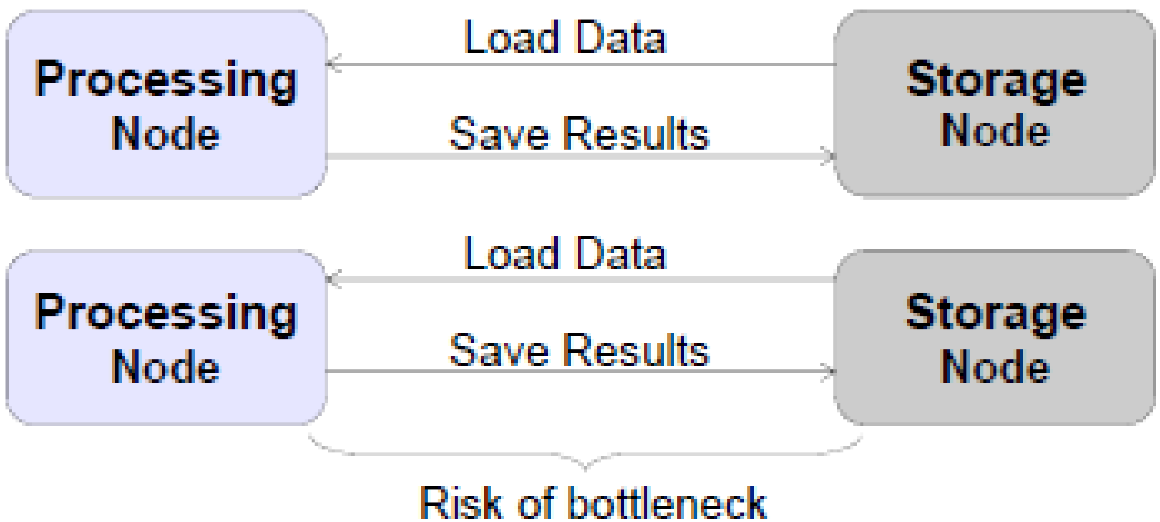


Scale-Out Instead of Scale-Up

- **It is harder and more expensive to scale-up**
  - Add additional resources to an existing node (CPU, RAM)
  - Moore's Law can't keep up with data growth
  - New units must be purchased if required resources can not be added
  - Also known as scale vertically
- **Scale-Out**
  - Add more nodes/machines to an existing distributed application
  - Software layer is designed for node additions or removal
  - Hadoop takes this approach - A set of nodes are bonded together as a single distributed system
  - Very easy to scale down as well

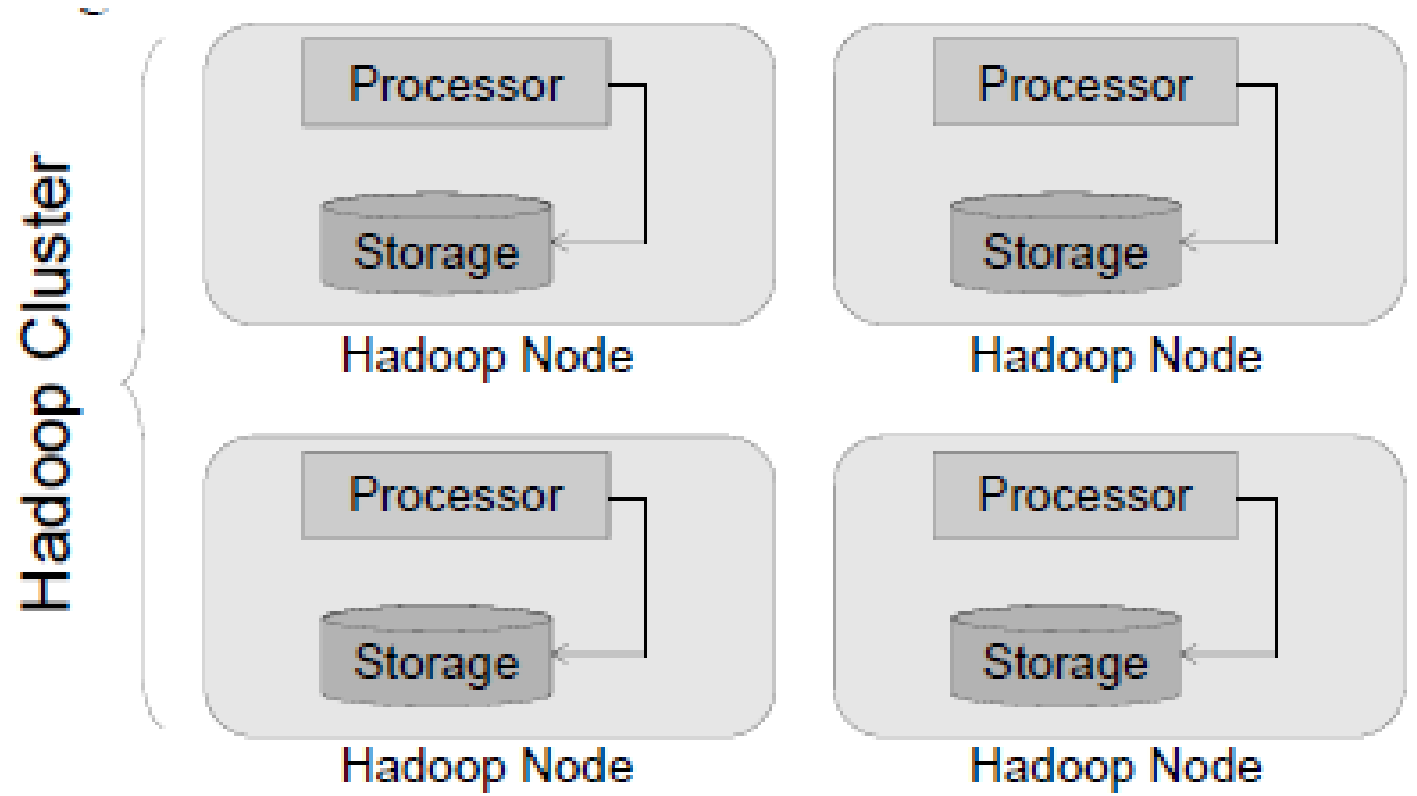
Code to Data

- **Traditional data processing architecture**
  - Nodes are broken up into separate processing and storage nodes connected by high-capacity link
  - Many data-intensive applications are not CPU demanding causing bottlenecks in network



Code to Data

- **Hadoop co-locates processors and storage**
  - Code is moved to data (size is tiny, usually in KBs)
  - Processors execute code and access underlying local storage



Failures are Common

- **Given a large number machines, failures are common**
  - Large warehouses may see machine failures weekly or even daily
- **Hadoop is designed to cope with node failures**
  - Data is replicated
  - Tasks are retried

Comparison to RDBMS

- **Relational Database Management Systems (RDBMS) for batch processing**
  - Oracle, Sybase, MySQL, Microsoft SQL Server, etc.
  - Hadoop doesn't fully replace relational products; many architectures would benefit from both Hadoop and a Relational product
  - RDBMS products scale up
    - Expensive to scale for larger installations
    - Hits a ceiling when storage reaches 100s of terabytes
  - Structured Relational vs. Semi-Structured vs. Unstructured
  - Hadoop was not designed for real-time or low latency queries

HDFS
(Hadoop Distributed File System)

HDFS

- **Appears as a single disk**
- **Runs on top of a native filesystem**
- **Fault Tolerant**
  - Can handle disk crashes, machine crashes, etc…
- **Based on Google's Filesystem (GFS or GoogleFS)**

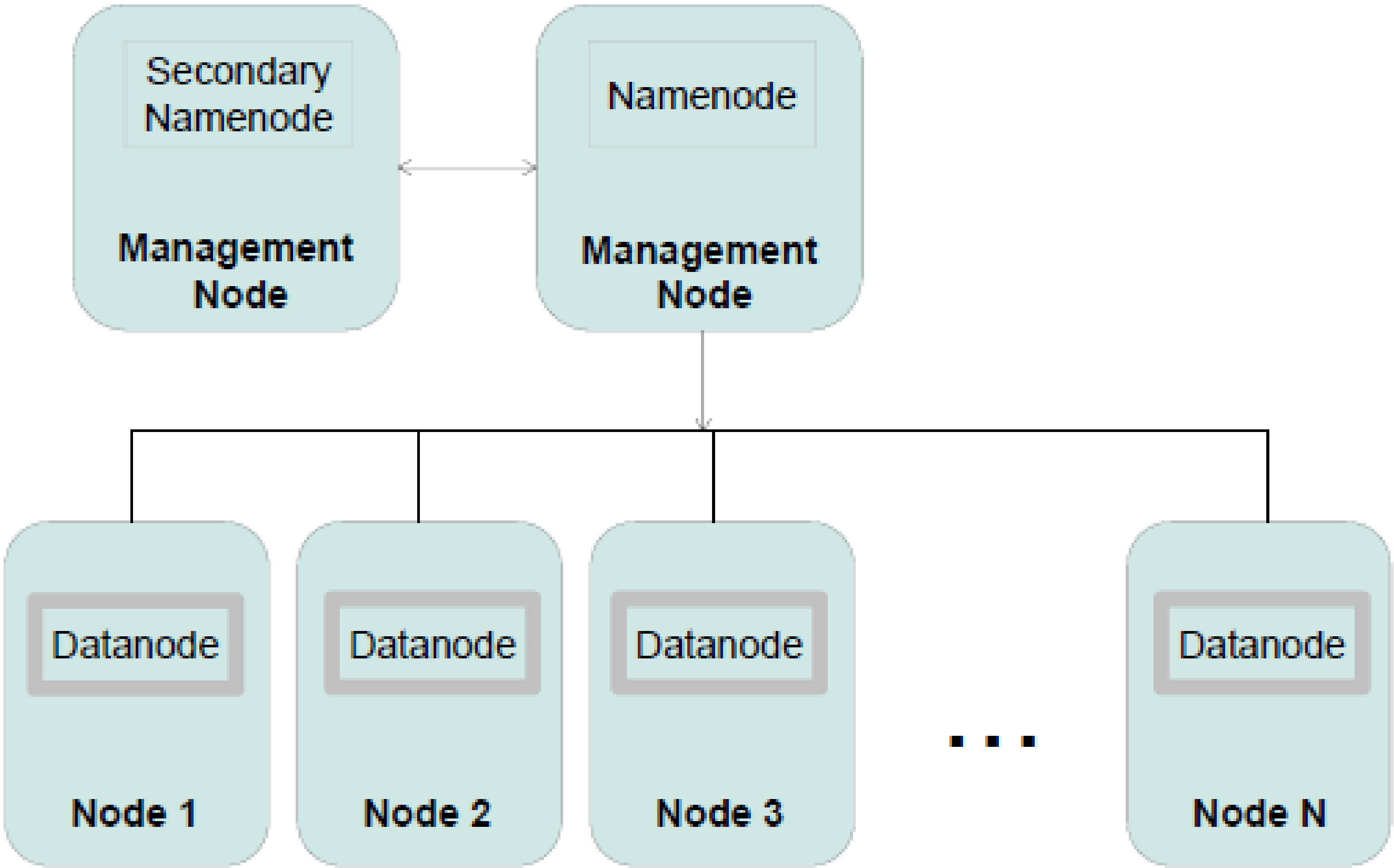
HDFS is Good for...

- **Storing large files**
  - Terabytes, Petabytes, etc...
  - Millions rather than billions of files
  - 100MB or more per file
- **Streaming data**
  - Write once and read-many times patterns
  - Optimized for streaming reads rather than random reads
- **“Cheap” Commodity Hardware**
  - No need for super-computers, use less reliable commodity hardware

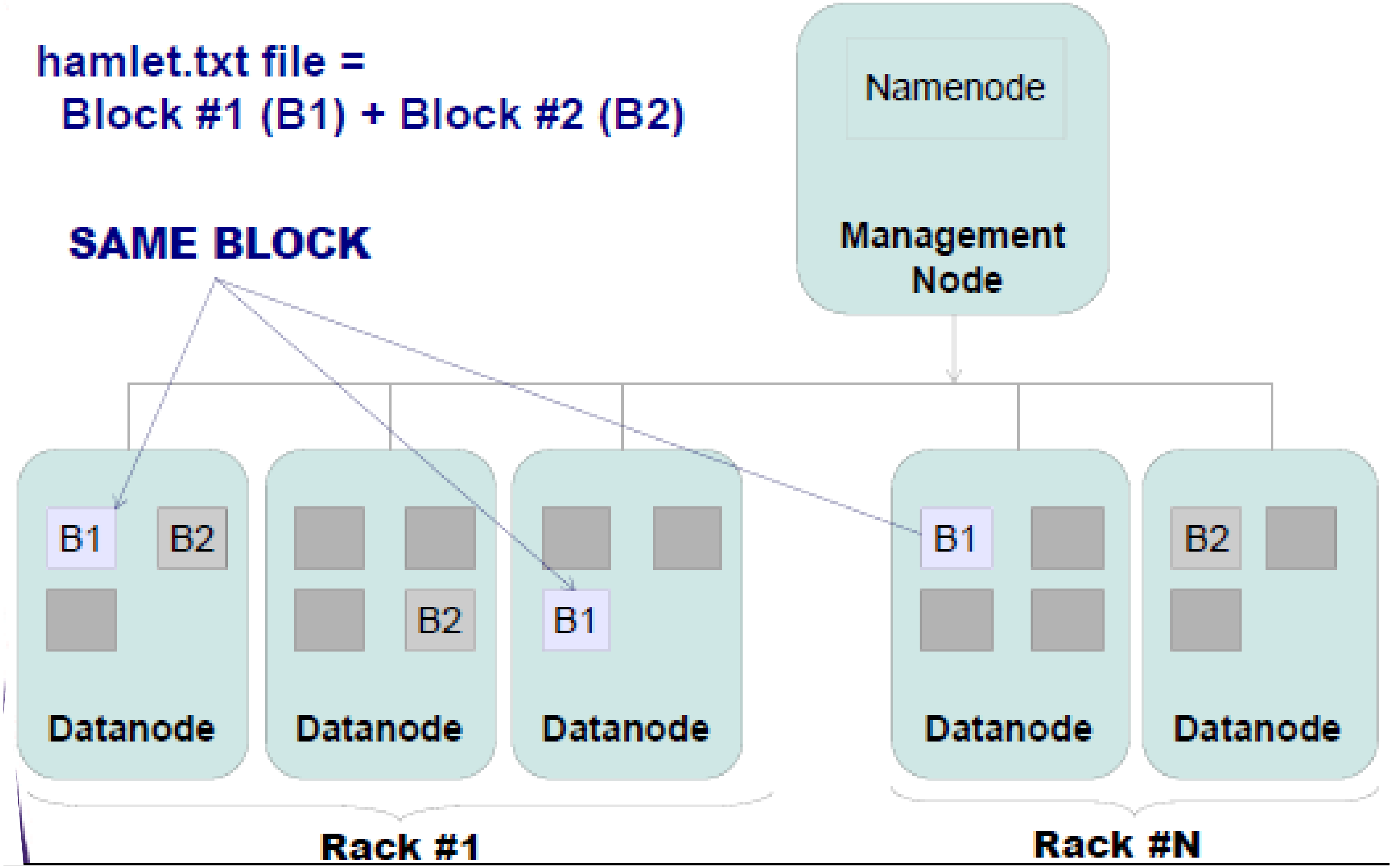
HDFS is not so good for…

- **Low-latency reads**
  - High-throughput rather than low latency for small chunks of data
  - HBase addresses this issue
- **Large amount of small files**
  - Better for millions of large files instead of billions of small files
    - For example each file can be 100MB or more
- **Multiple Writers**
  - Single writer per file
  - Writes only at the end of file, no-support for arbitrary offset

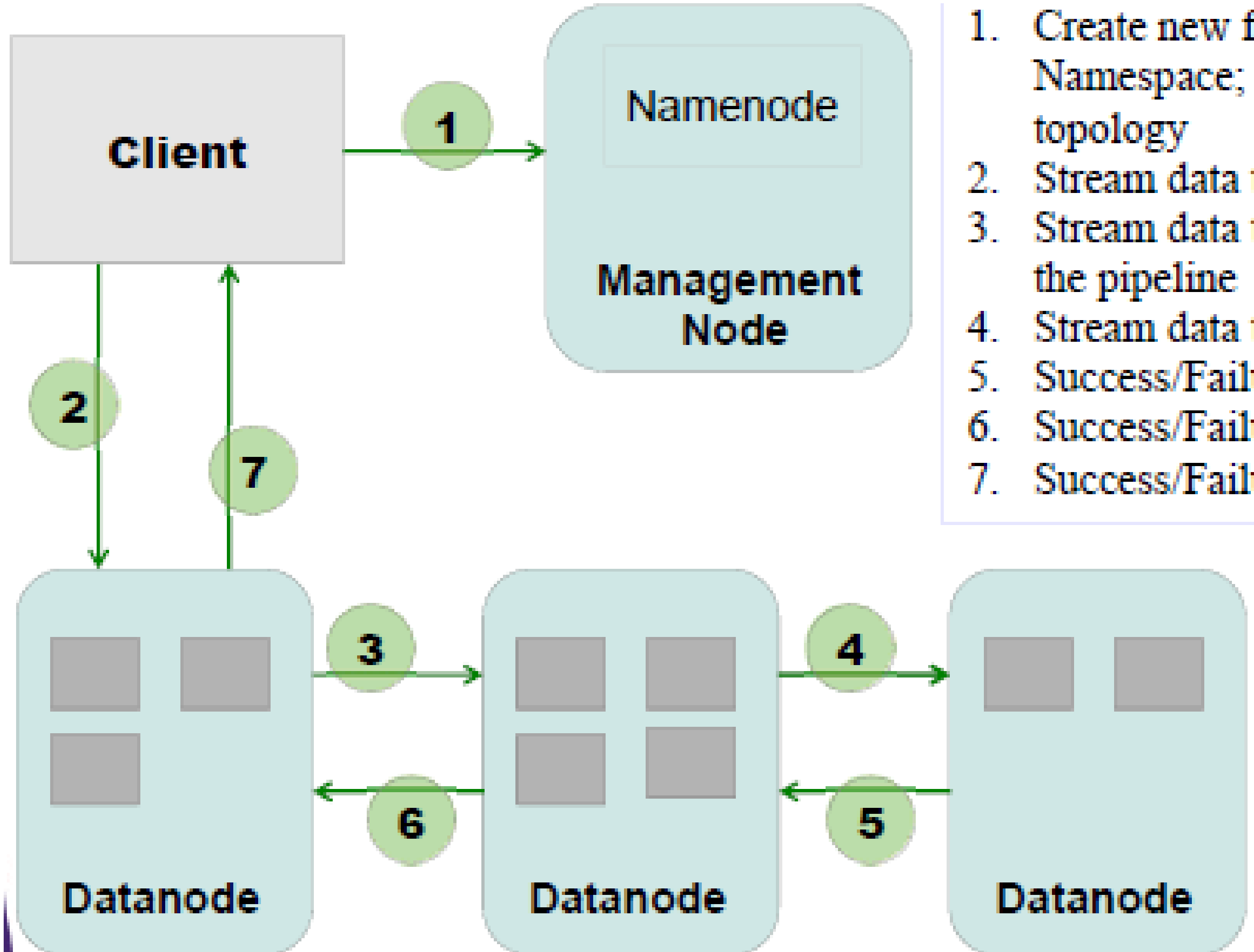
HDFS Daemons



Files and Blocks



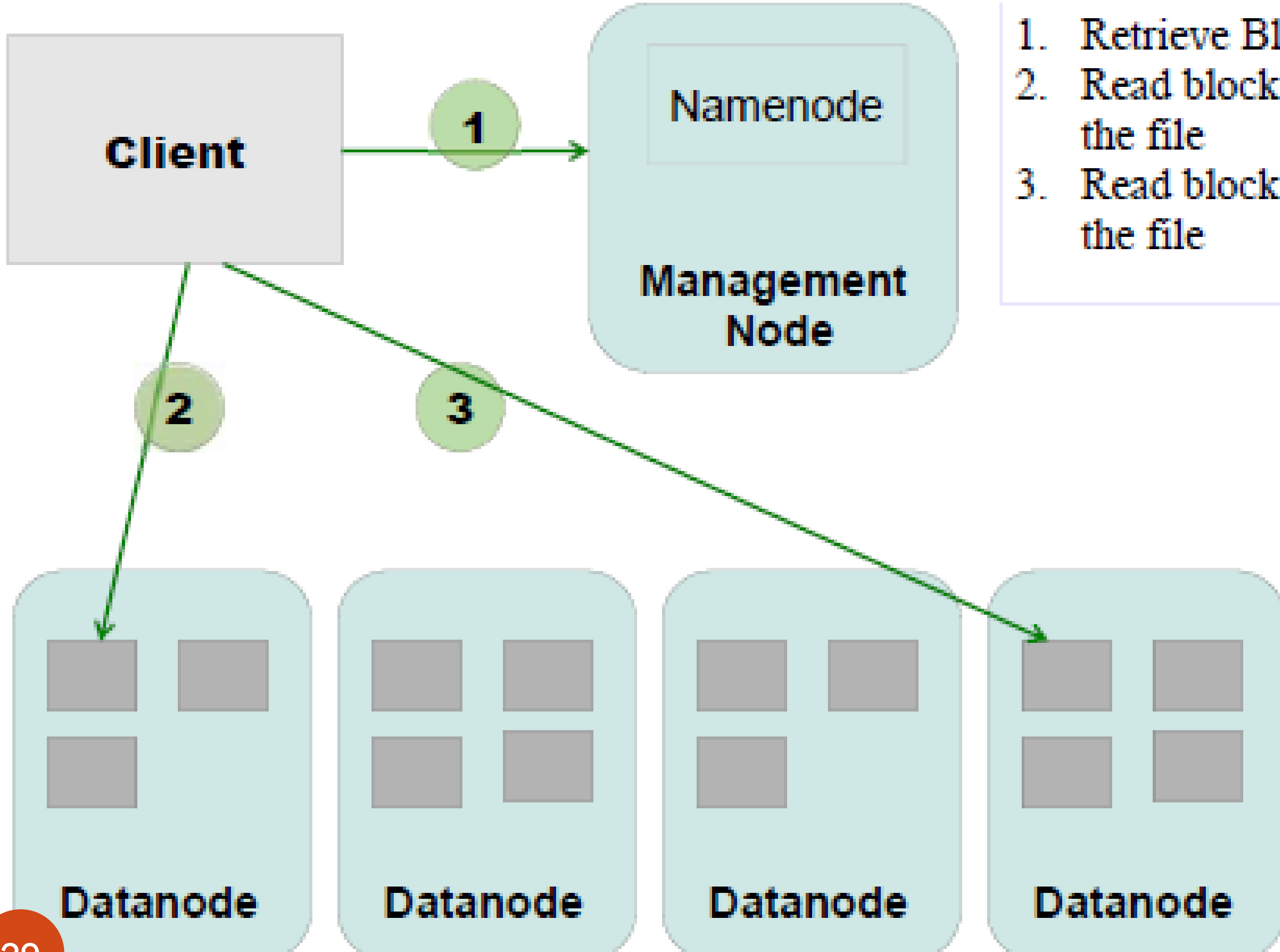
HDFS File Write



1. Create new file in the Namenode's Namespace; calculate block topology
2. Stream data to the first Node
3. Stream data to the second node in the pipeline
4. Stream data to the third node
5. Success/Failure acknowledgment
6. Success/Failure acknowledgment
7. Success/Failure acknowledgment

Source: White, Tom. Hadoop The Definitive Guide. O'Reilly Media. 2012

HDFS File Read



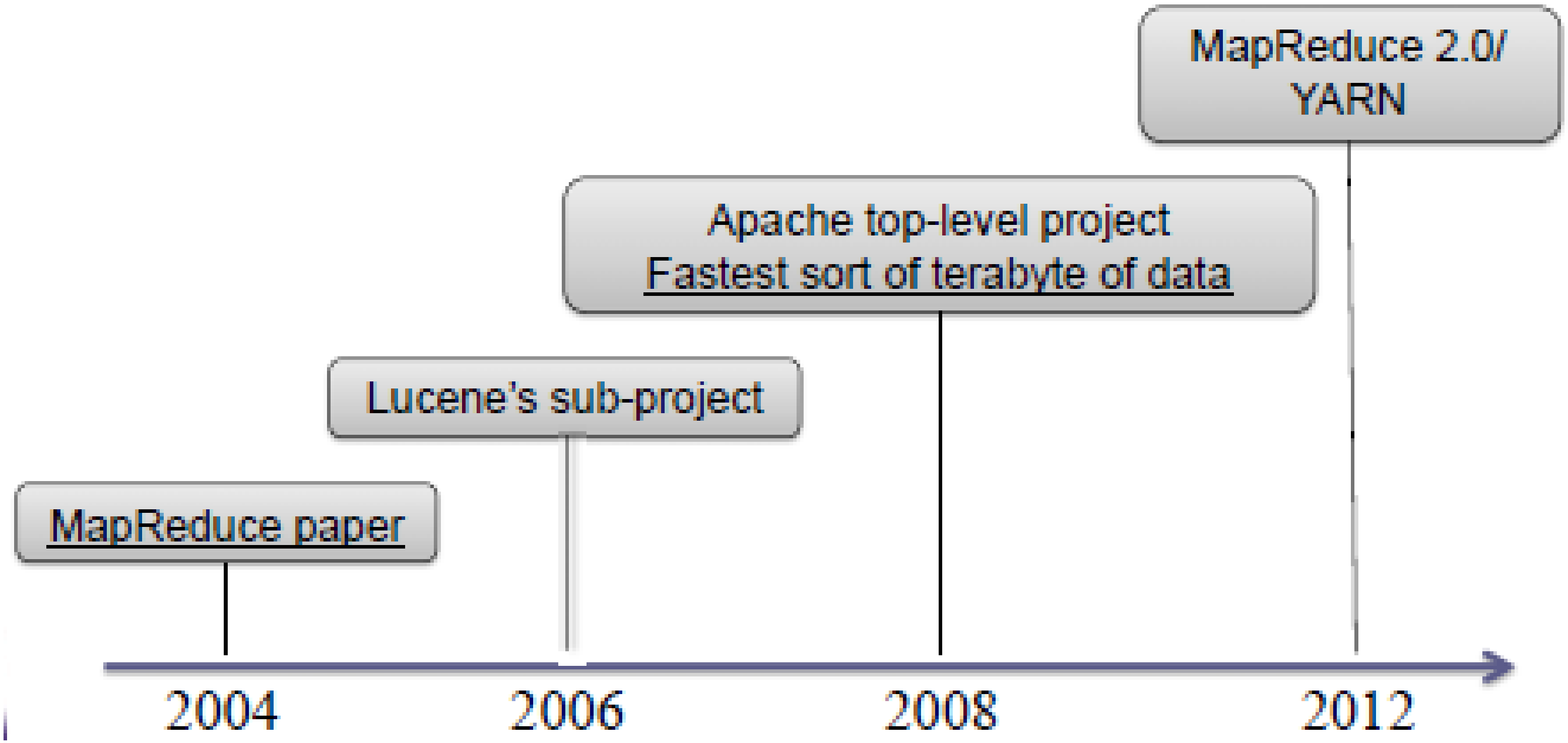
1. Retrieve Block Locations
2. Read blocks to re-assemble the file
3. Read blocks to re-assemble the file

What is MapReduce?

Hadoop MapReduce

- **Model for processing large amounts of data in parallel**
  - On commodity hardware
  - Lots of nodes
- **Derived from functional programming**
  - Map and reduce functions
- **Can be implemented in multiple languages**
  - Java, C++, Ruby, Python, etc.

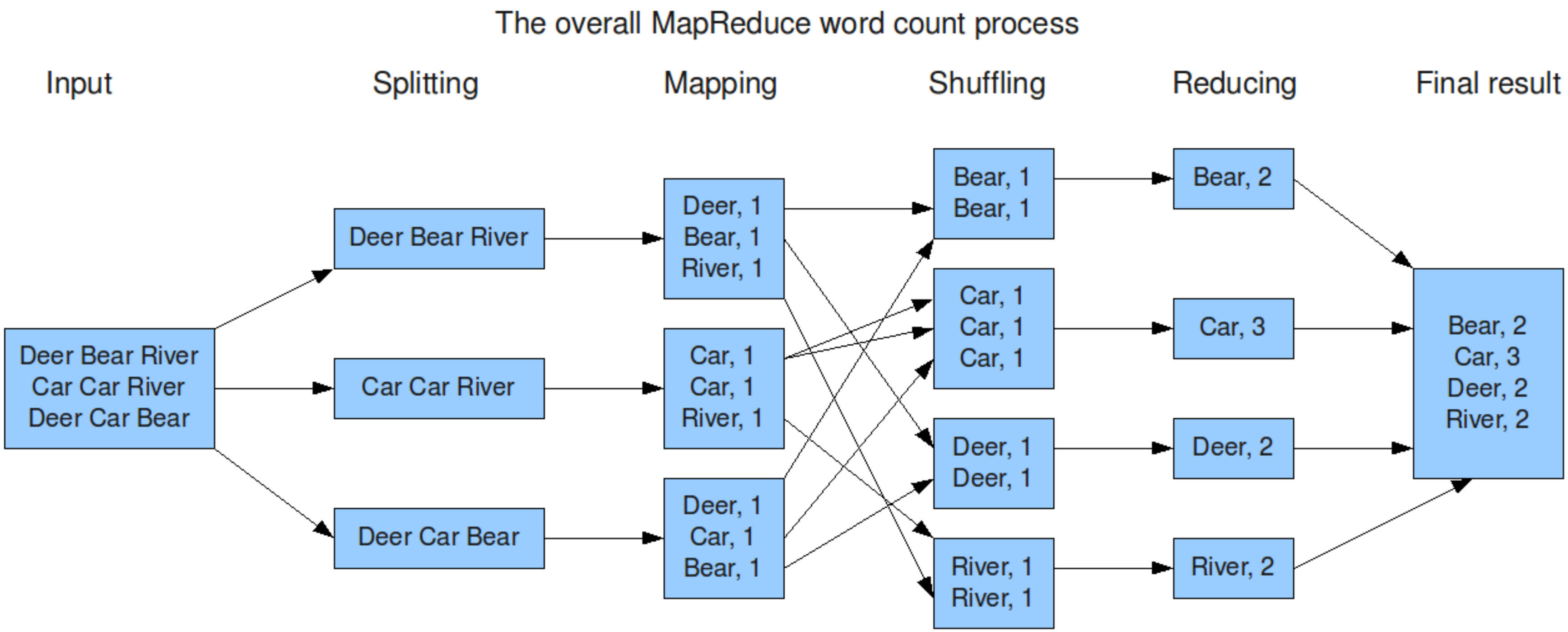
Hadoop MapReduce History



Main principle

- **Map**: (f, [a, b, c, …]) -> [f(a), f(b), f(c), …]
  - Apply a function to all the elements of a list
  - ex.: map((f: x->x + 1), [1, 2, 3]) = [2, 3, 4]
  - Intrinsically **parallel**
- **Reduce**: (g, [a, b, c, …]) -> g(a, g(b, g(c, …)))
  - Apply a function to a list recursively
  - ex.: (sum , [1, 2, 3 ,4]) = sum(1, sum(2, sum(3, 4)))
- Purely fonctionnal
  - No global variables, no side effects

WordCount example



MapReduce Framework

- **Takes care of distributed processing and coordination**
- **Scheduling**
  - Jobs are broken down into smaller chunks called tasks.
  - These tasks are scheduled.
- **Task localization with Data**
  - Framework strives to place tasks on the nodes that host the segment of data to be processed by that specific task
  - Code is moved to where the data is

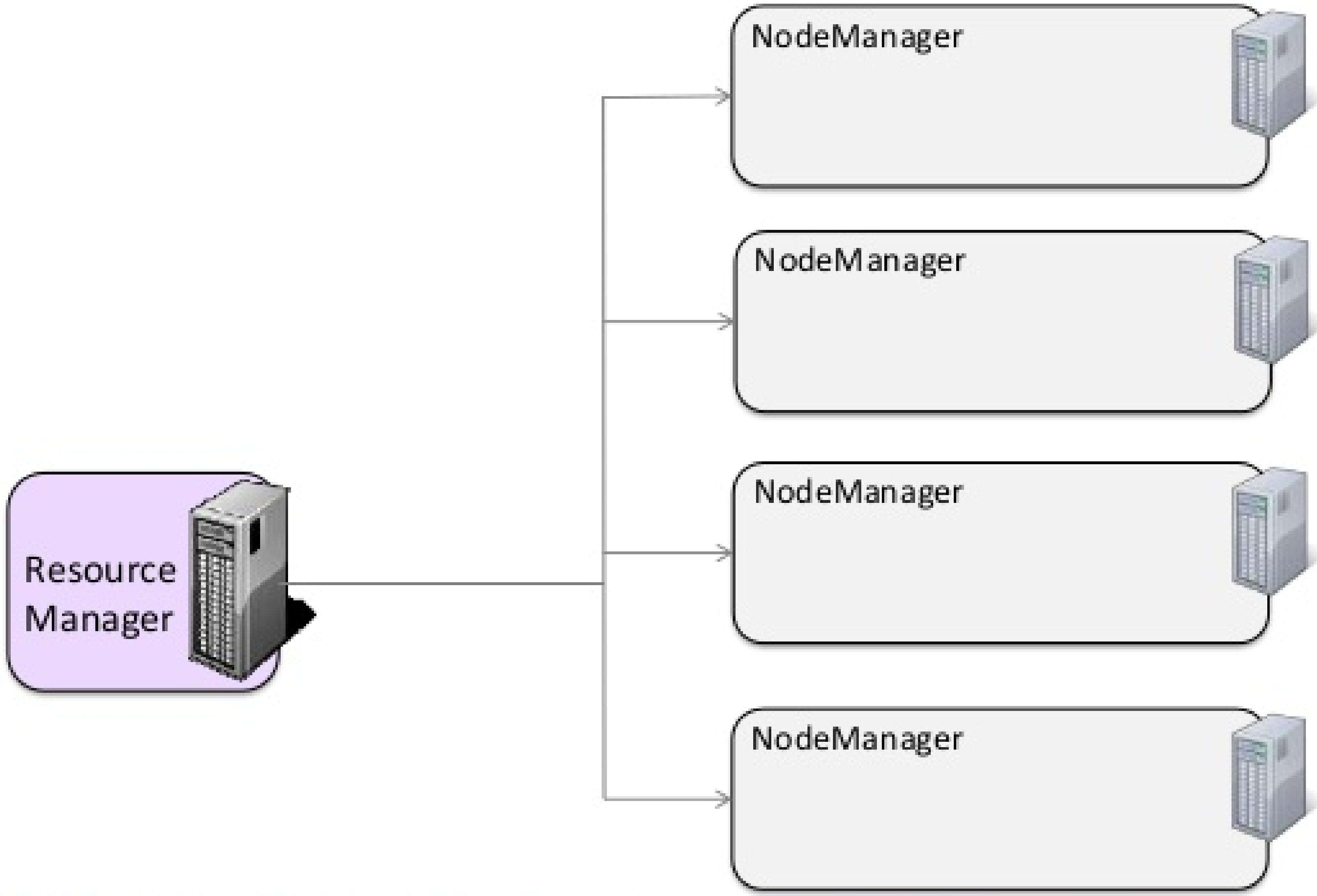
MapReduce Framework

- **Error Handling**
  - Failures are an expected behavior so tasks are automatically re-tried on other machines
- **Data Synchronization**
  - Shuffle and Sort barrier re-arranges and moves data between machines
  - Input and output are coordinated by the framework

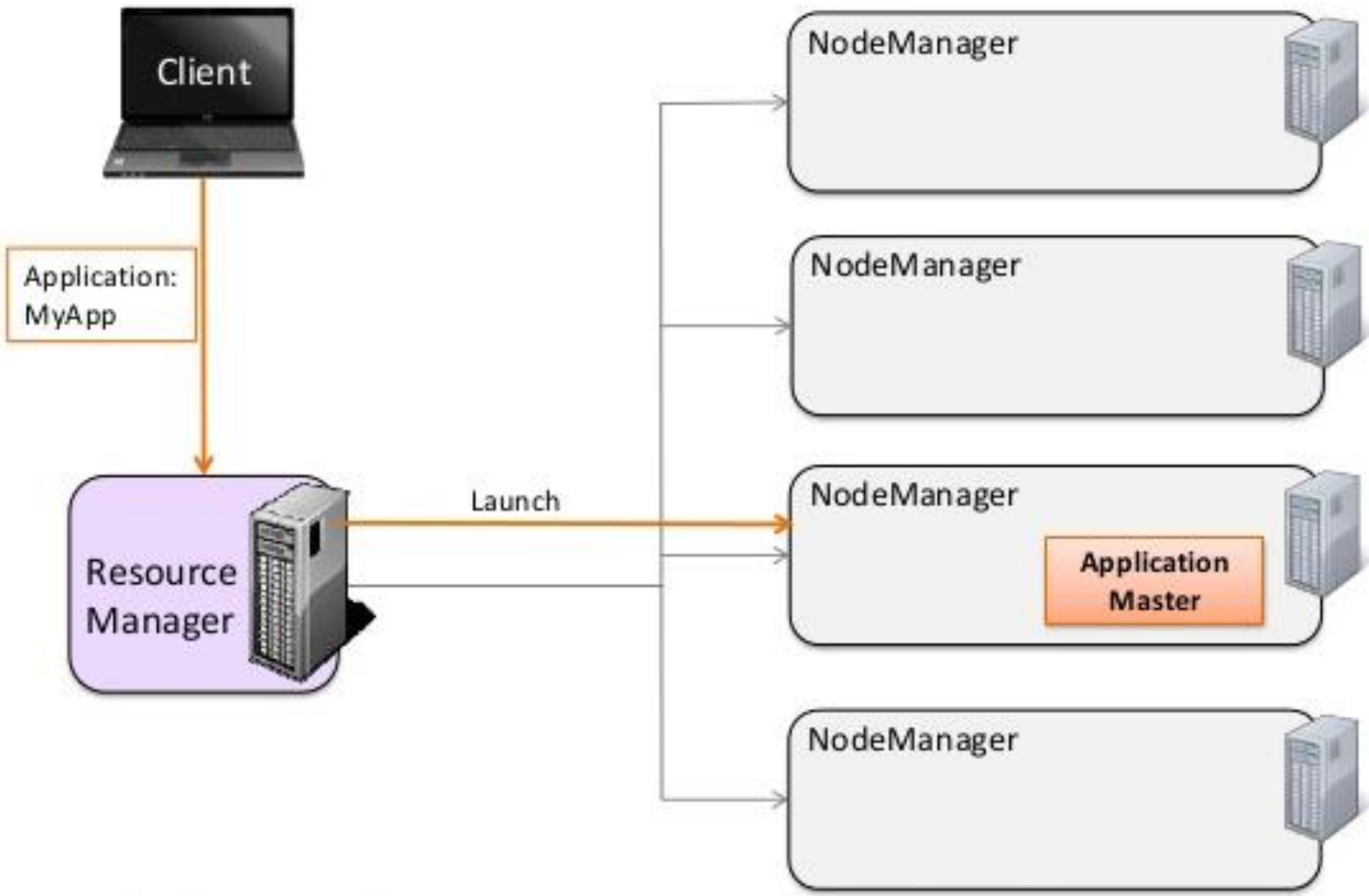
Map Reduce 2.0 on YARN

- **Yet Another Resource Negotiator (YARN)**
- **Various applications can run on YARN**
  - MapReduce is just one choice (the main choice at this point)
  - http://wiki.apache.org/hadoop/PoweredByYarn

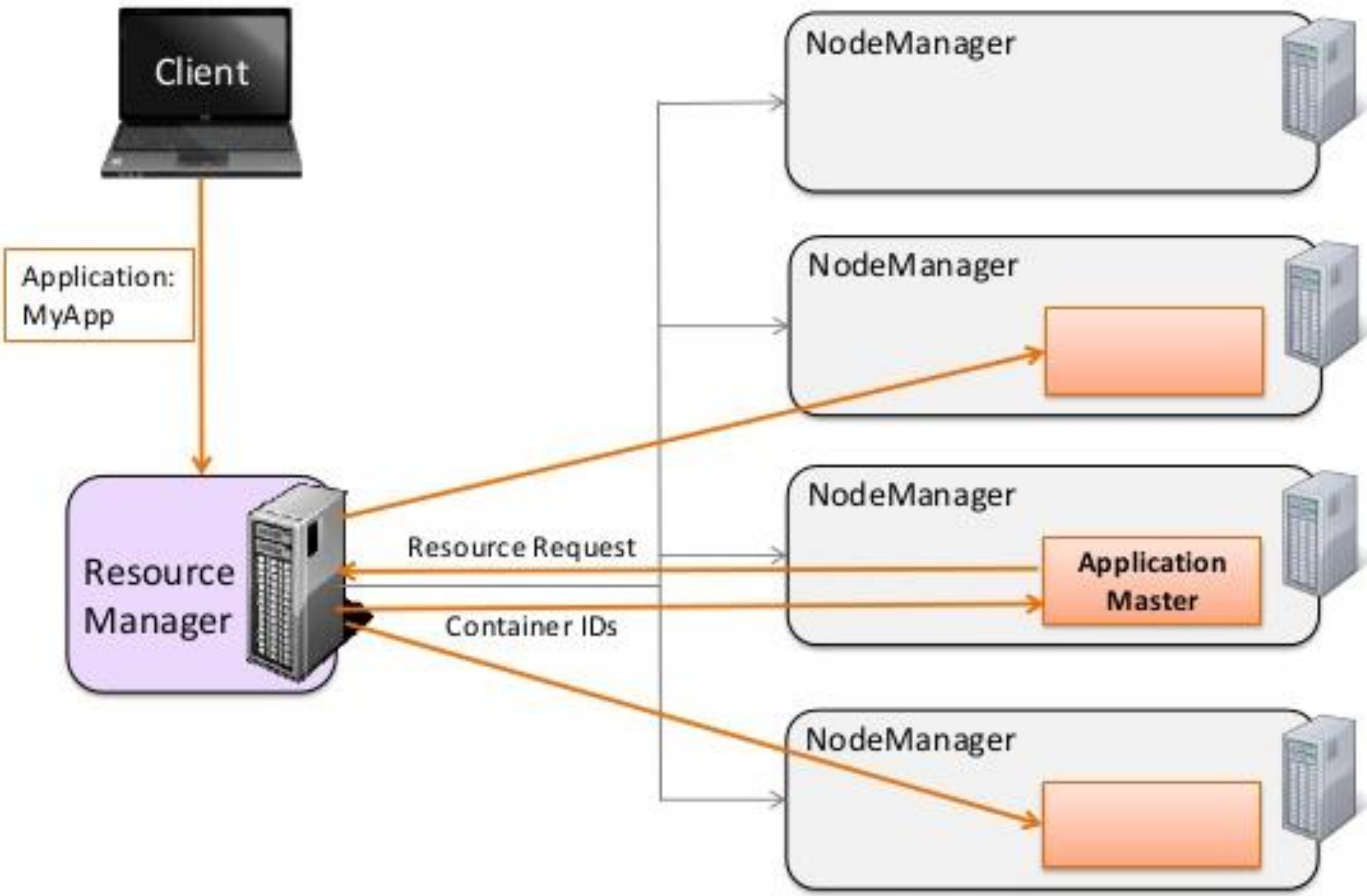
YARN Cluster



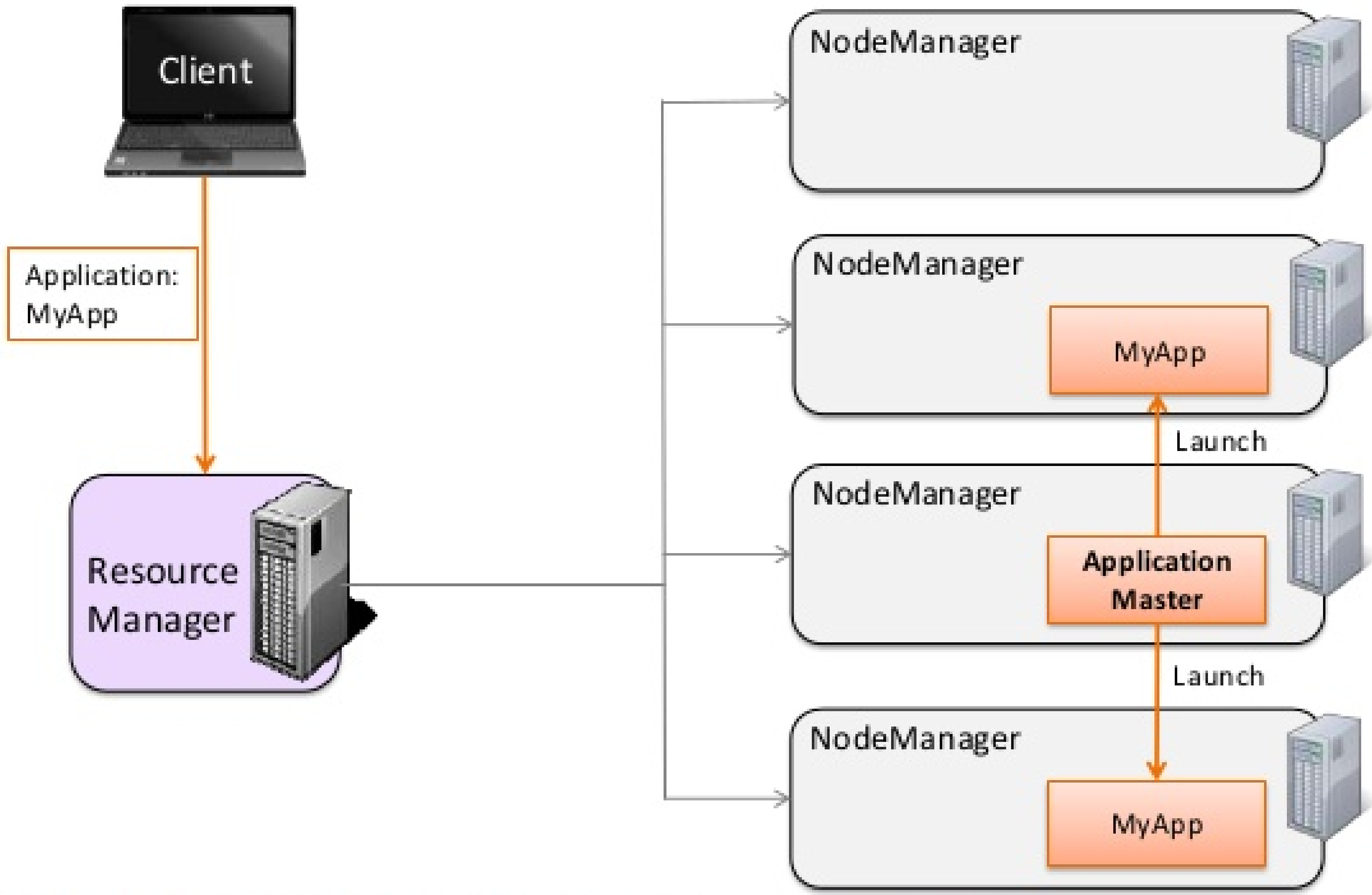
YARN: Running an Application



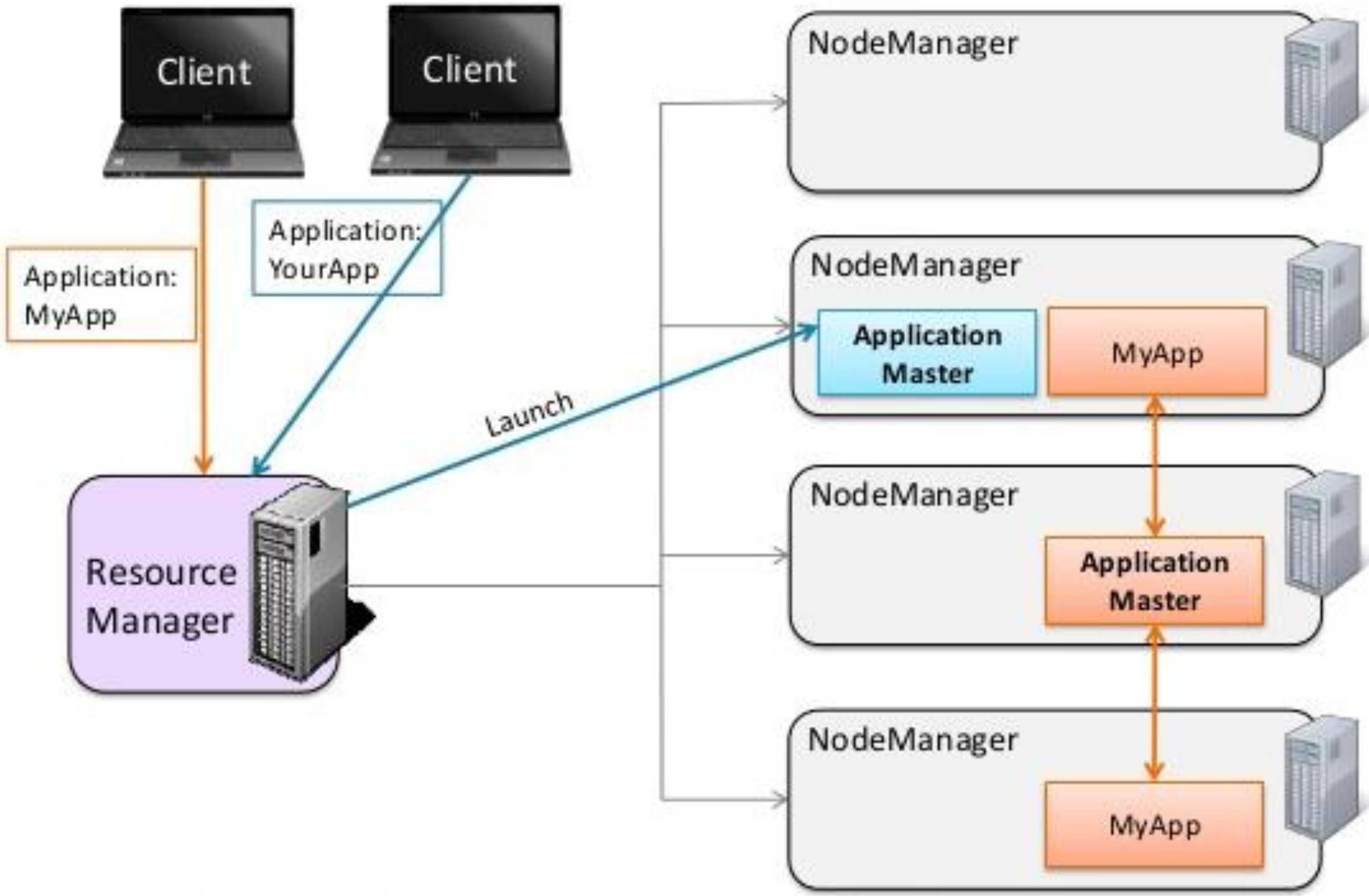
YARN: Running an Application



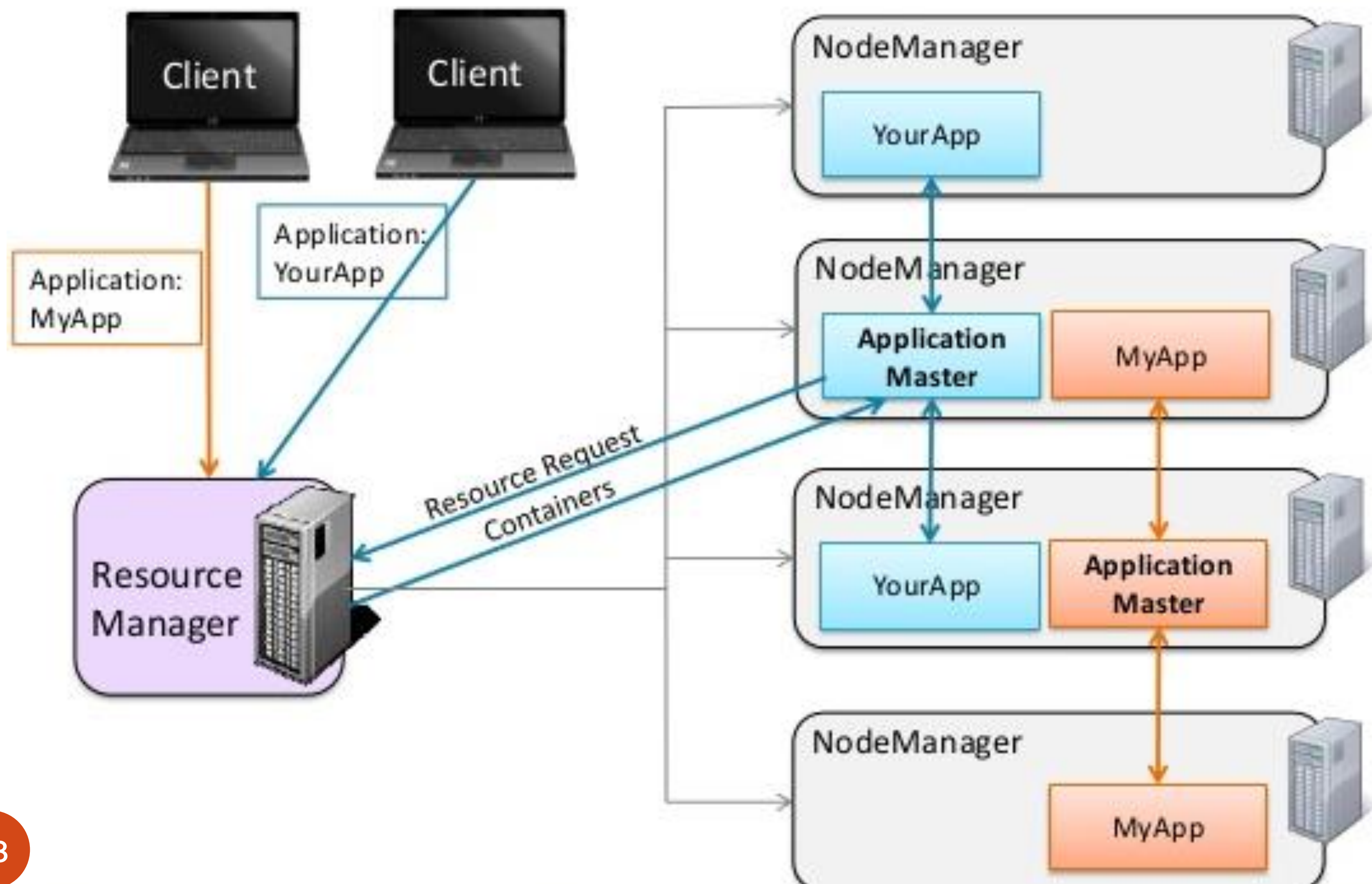
YARN: Running an Application



YARN: Running an Application

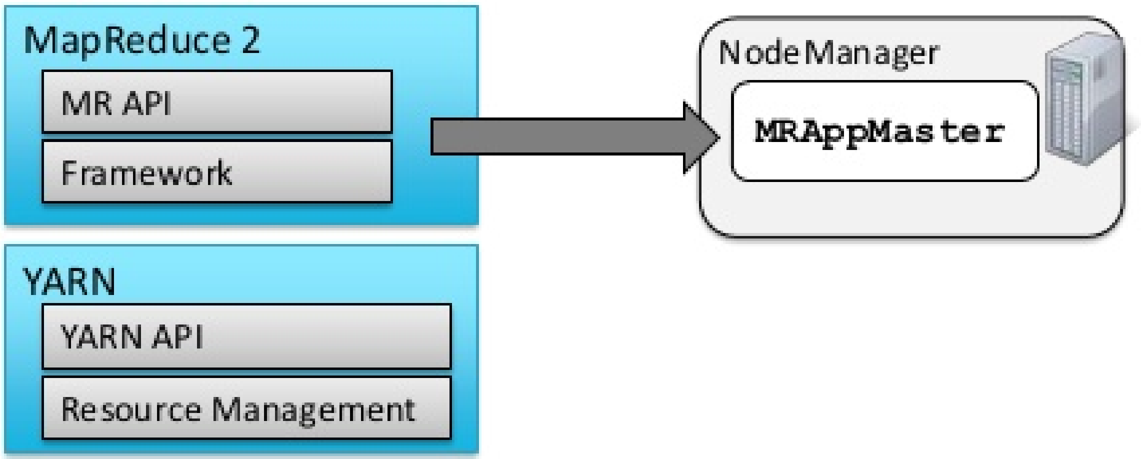


YARN: Running an Application

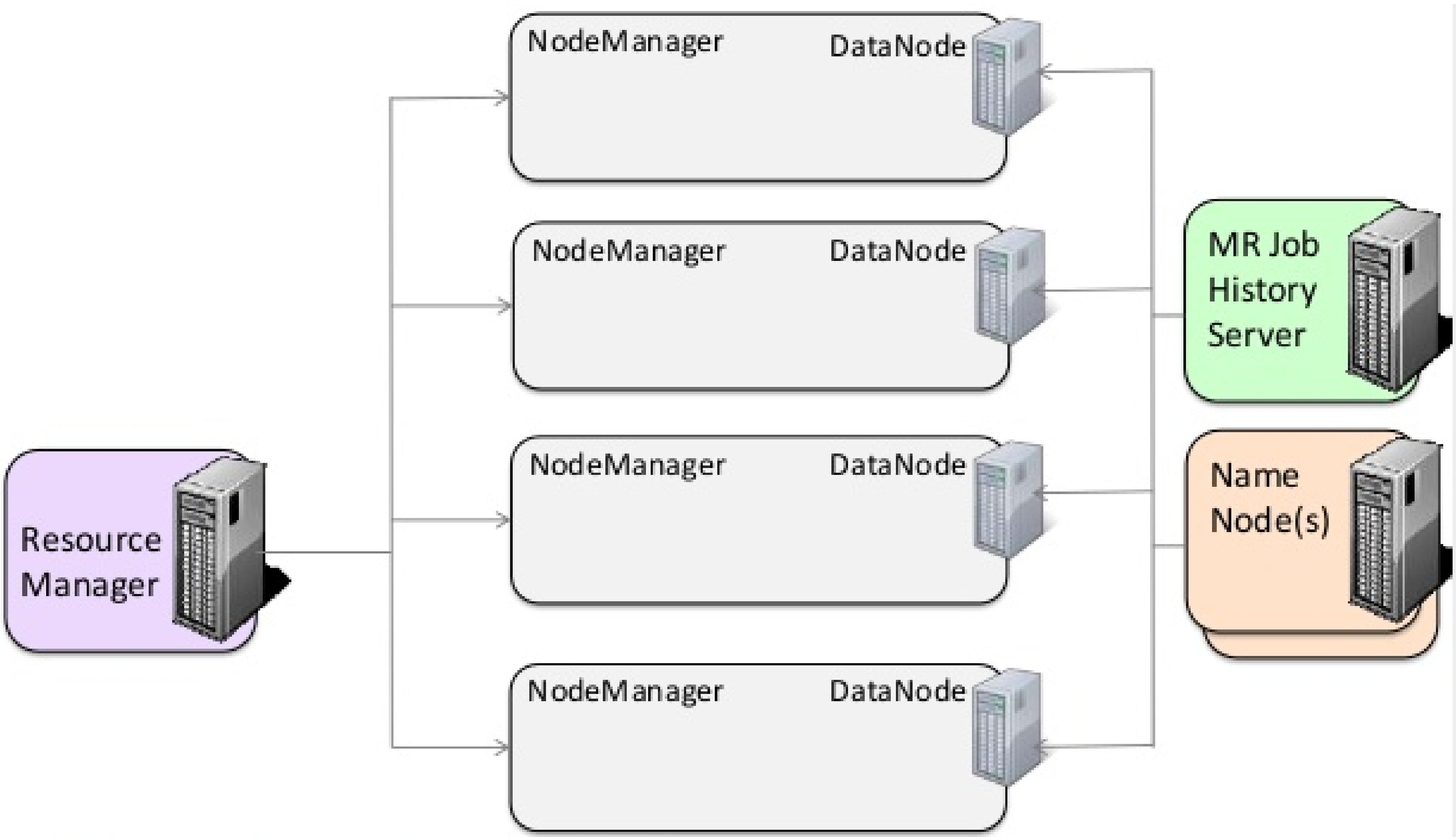


YARN and MapReduce

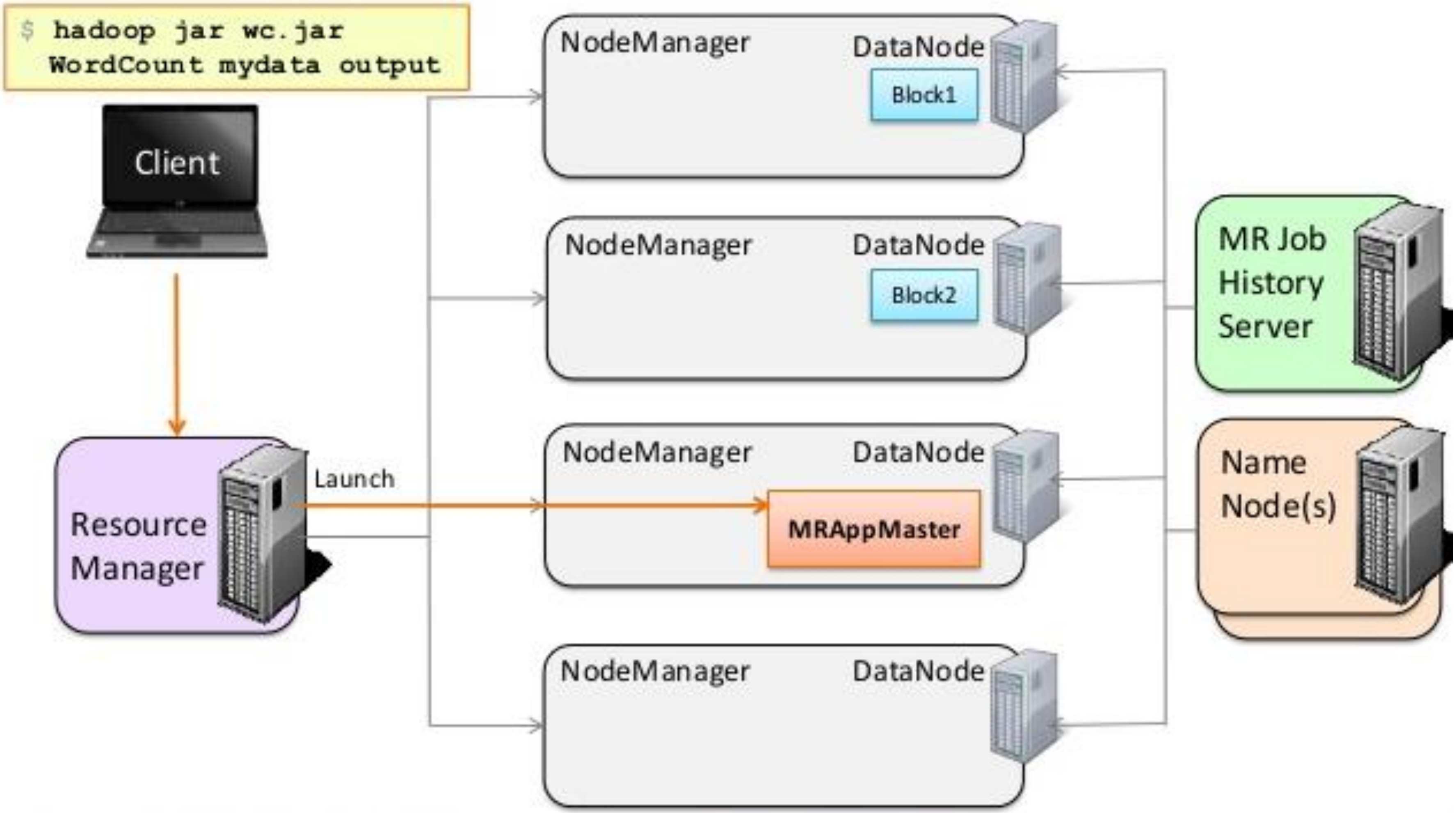
- YARN does not know or care what kind of application it is running
- MapReduce uses YARN
  - Hadoop includes a MapReduce ApplicationMaster to manage MapReduce jobs
  - Each MapReduce job is an instance of an application



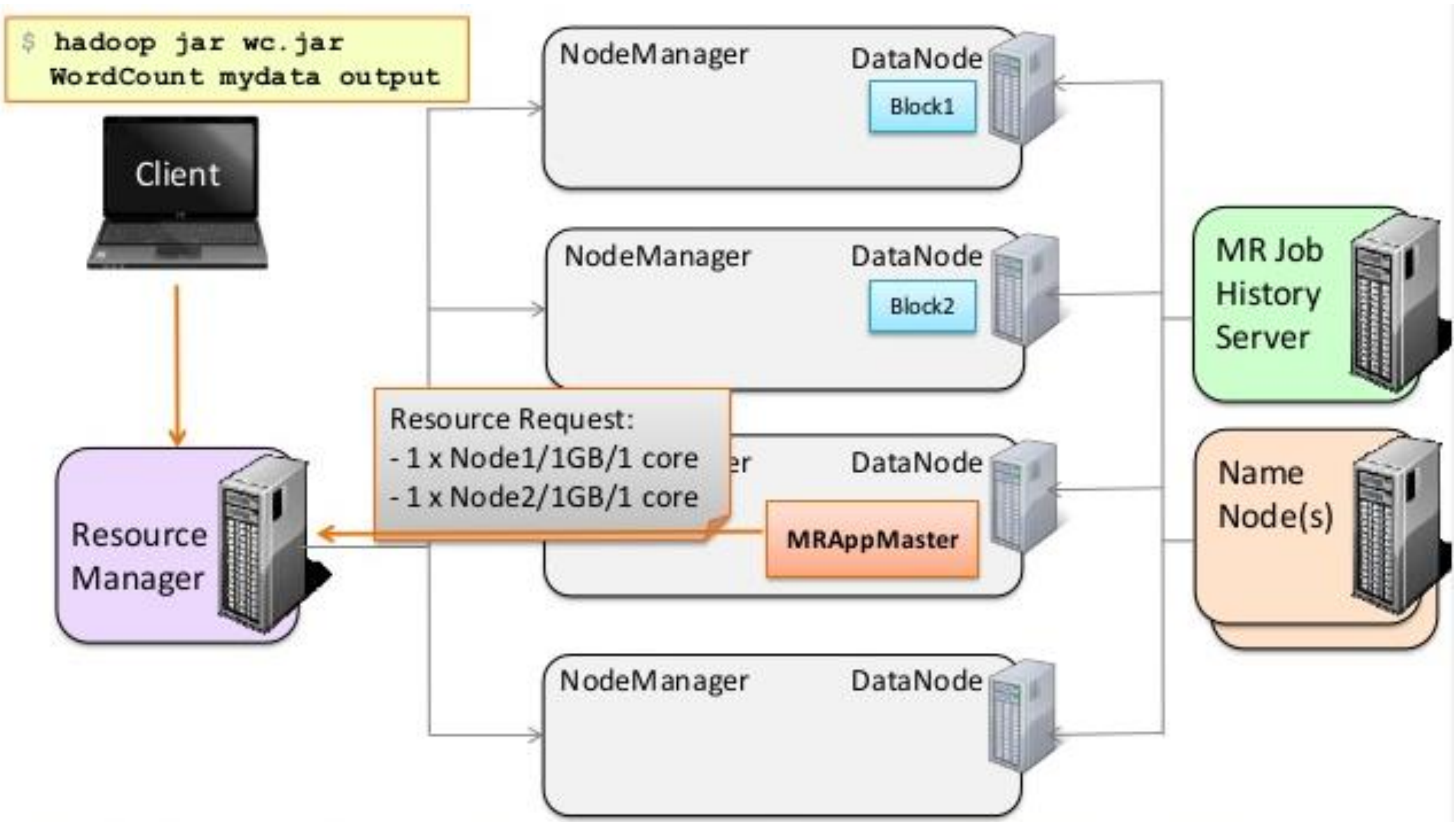
Running a MapReduce2 Application



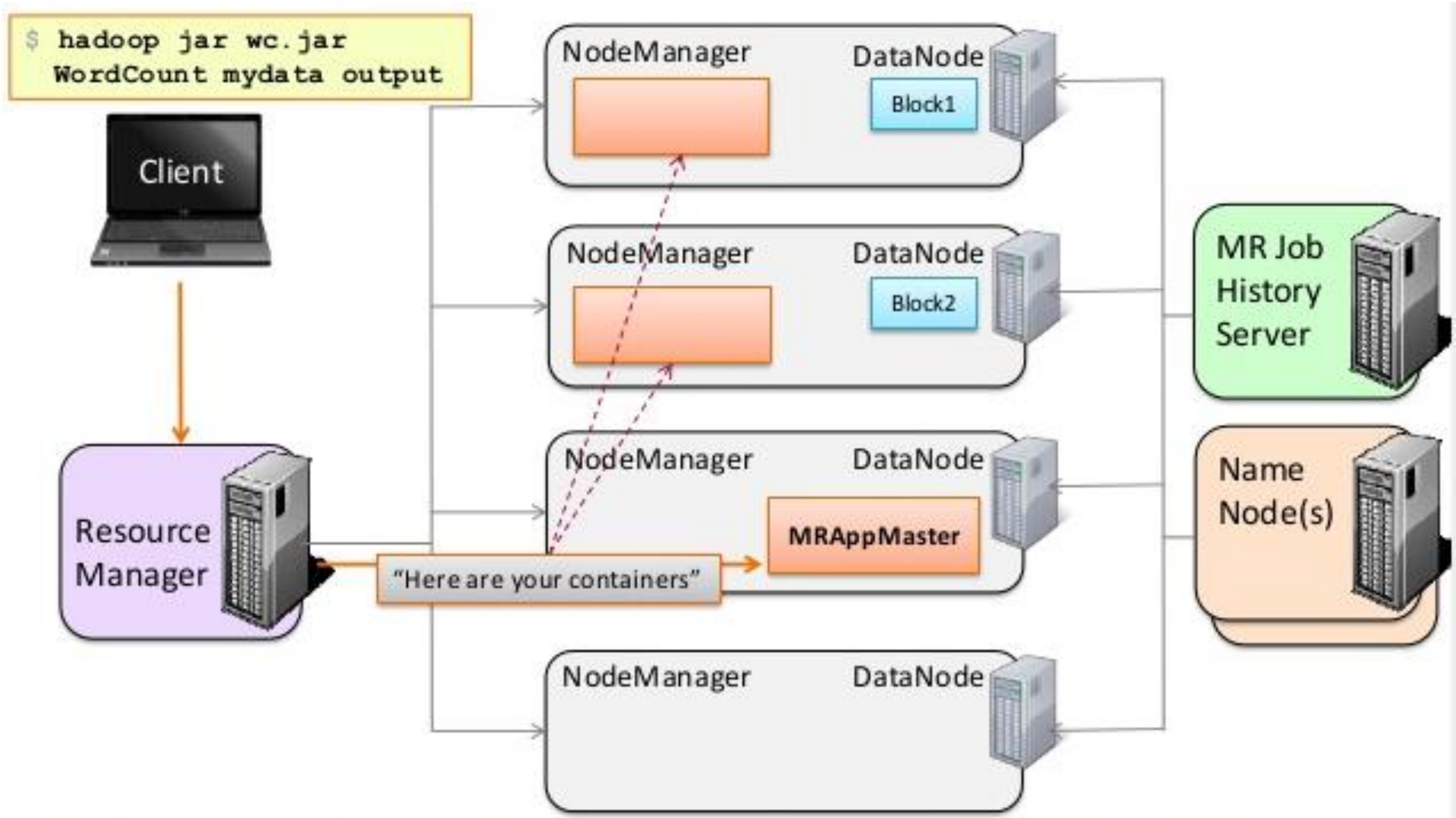
Running a MapReduce2 Application



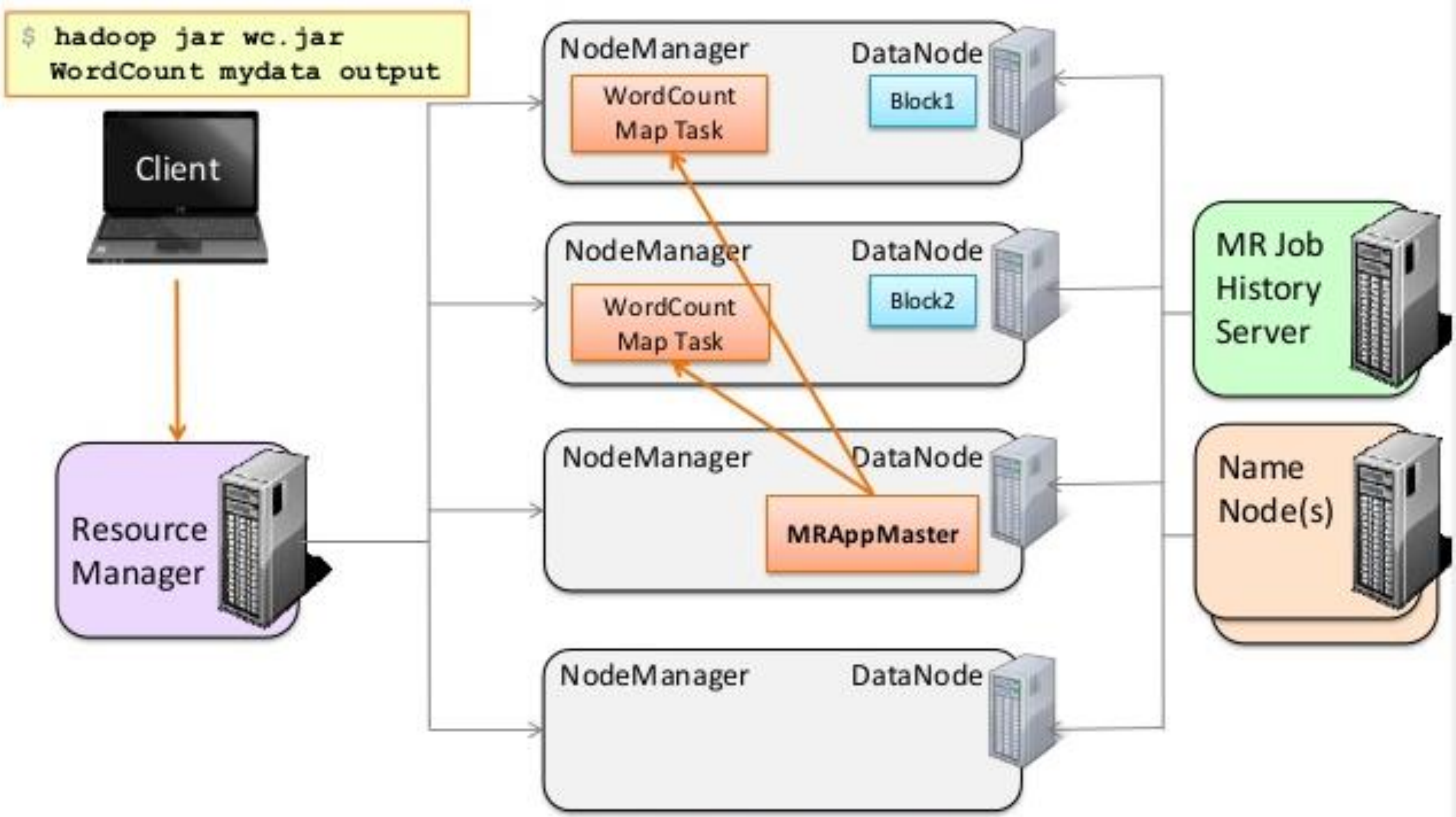
Running a MapReduce2 Application



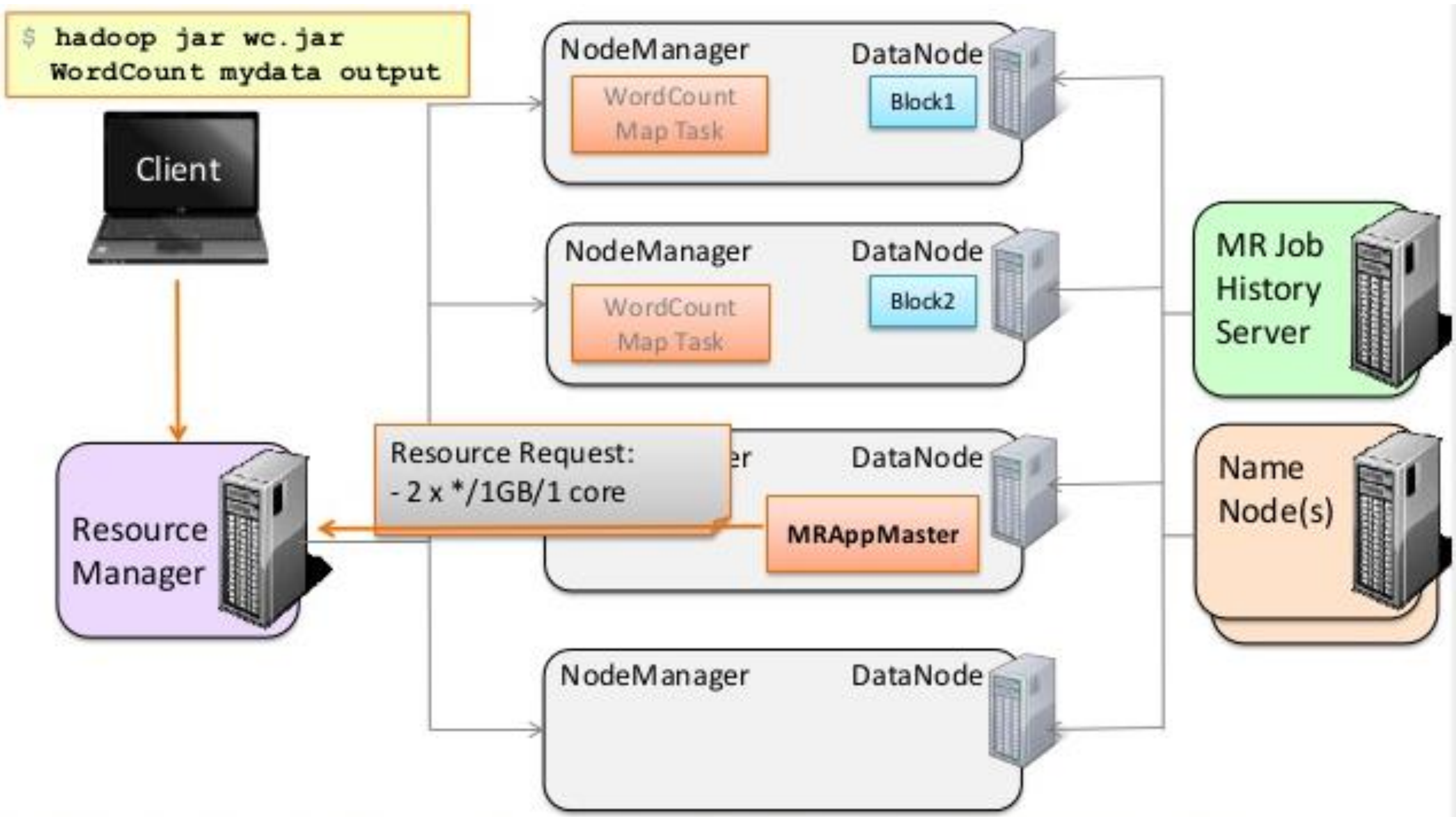
Running a MapReduce2 Application



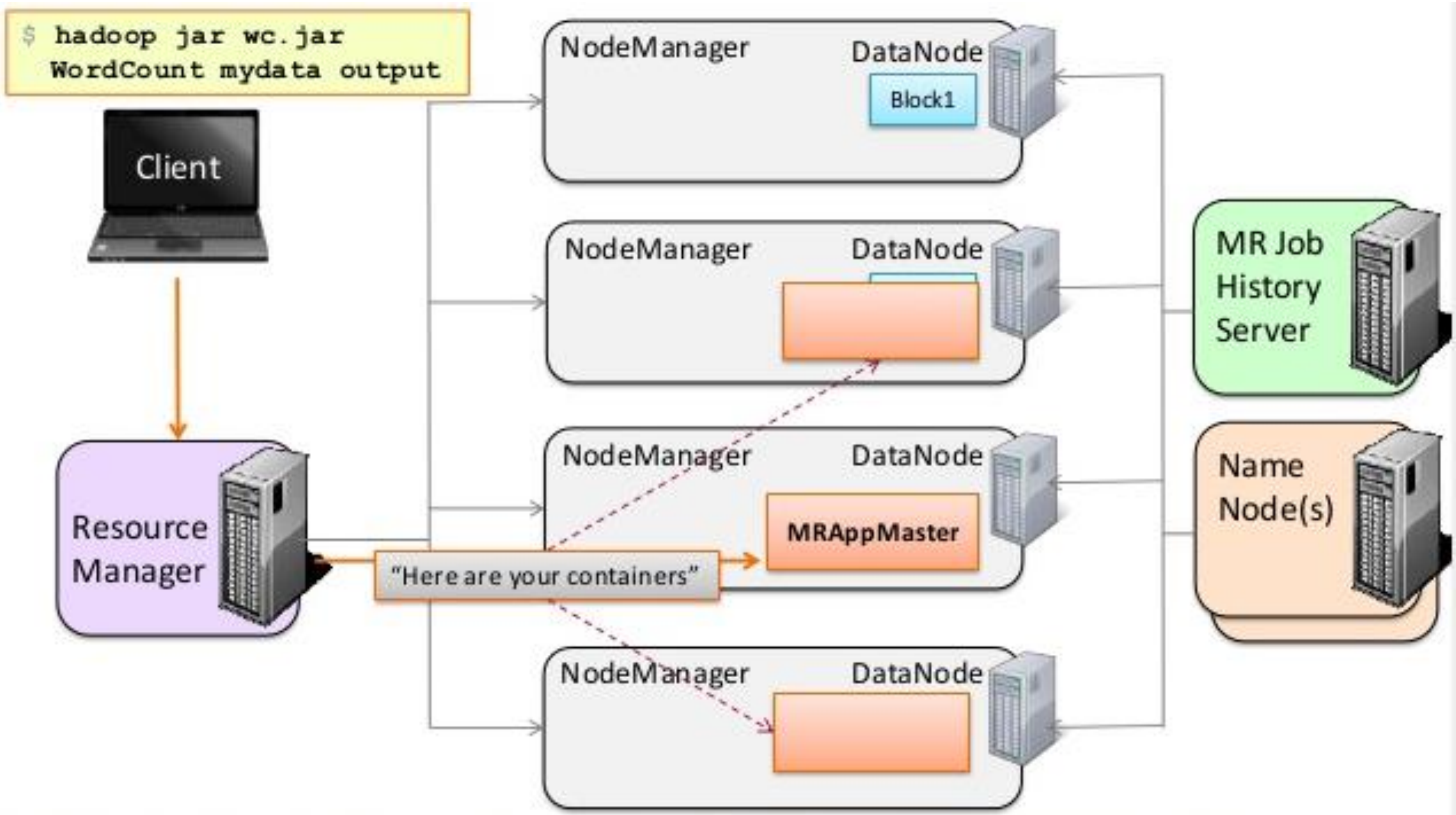
Running a MapReduce2 Application



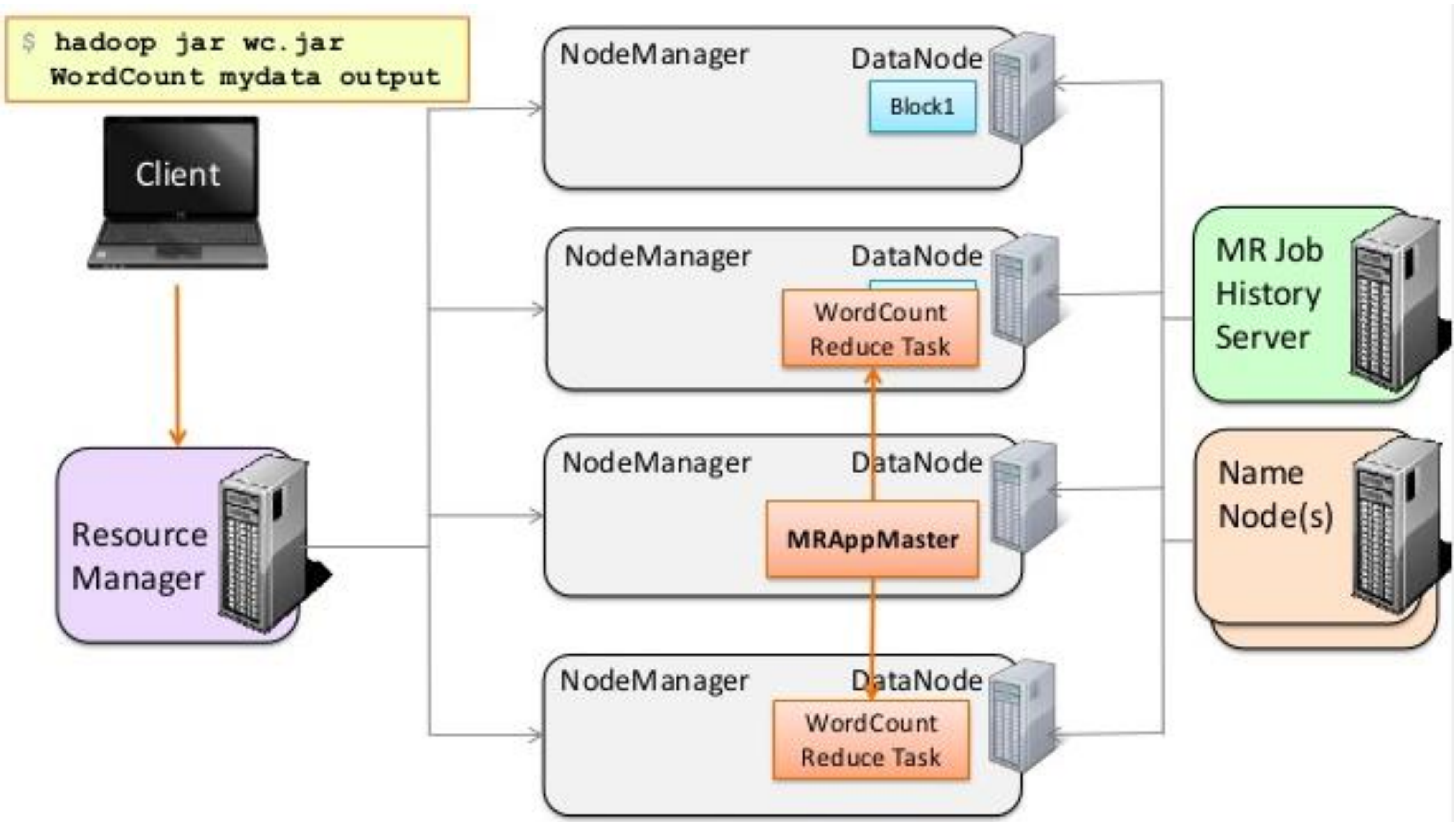
Running a MapReduce2 Application



Running a MapReduce2 Application



Running a MapReduce2 Application



Running a MapReduce2 Application

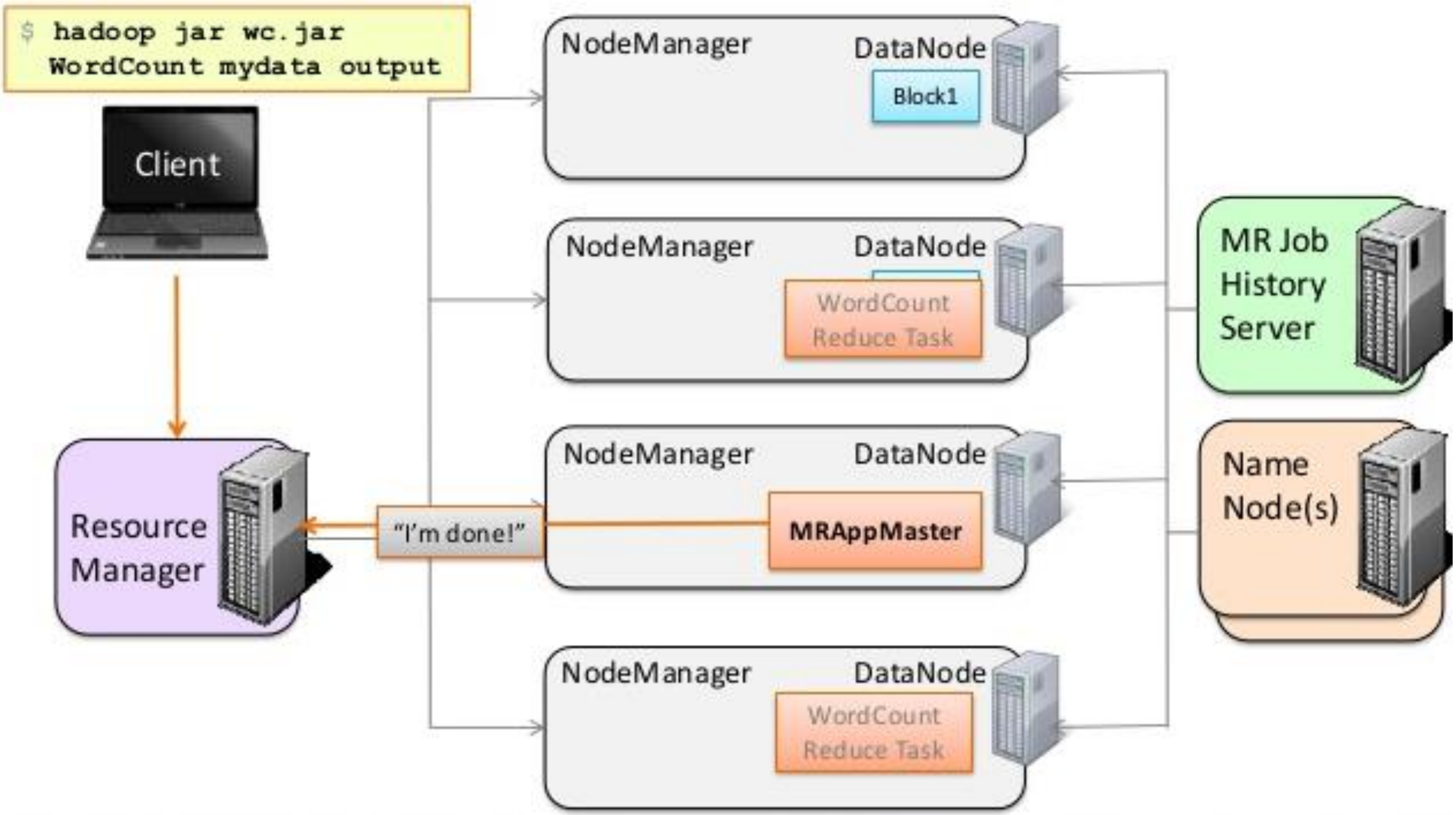


Image Coaddition with MapReduce

What is Astronomical Survey Science from Big Data point of view ?

- Gather millions of images and TBs/PBs of storage.
- Require high-throughput data reduction pipelines.
- Require sophisticated off-line data analysis tools
- The following example is extracted from

Wiley K., Connolly A., Gardner J., Krughoff S., Balazinska M., Howe B., Kwon Y., Bu Y.

Astronomy in the Cloud: Using MapReduce for Image Co-Addition. Publications of the Astronomical Society of the Pacific,

2011, vol. 123, no. 901, pp. 366-380.

FITS (Flexible Image Transport System)

- An image format that knows where it is looking.
- Common astronomical image representation file format.
- Metadata tags (like EXIF):
  - Most importantly: Precise astrometry (position on sky)
- Other:
  - Geolocation (telescope location)
  - Sky conditions, image quality, etc.

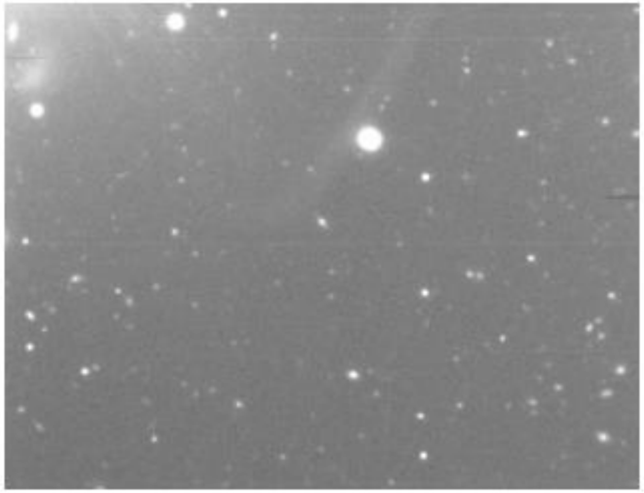


Image Coaddition

- Give **multiple** partially overlapping images and a **query** (color and sky bounds):
  - Find images' intersections with the query bounds.
  - Project bitmaps to the bounds.
  - Stack and mosaic into a final product.

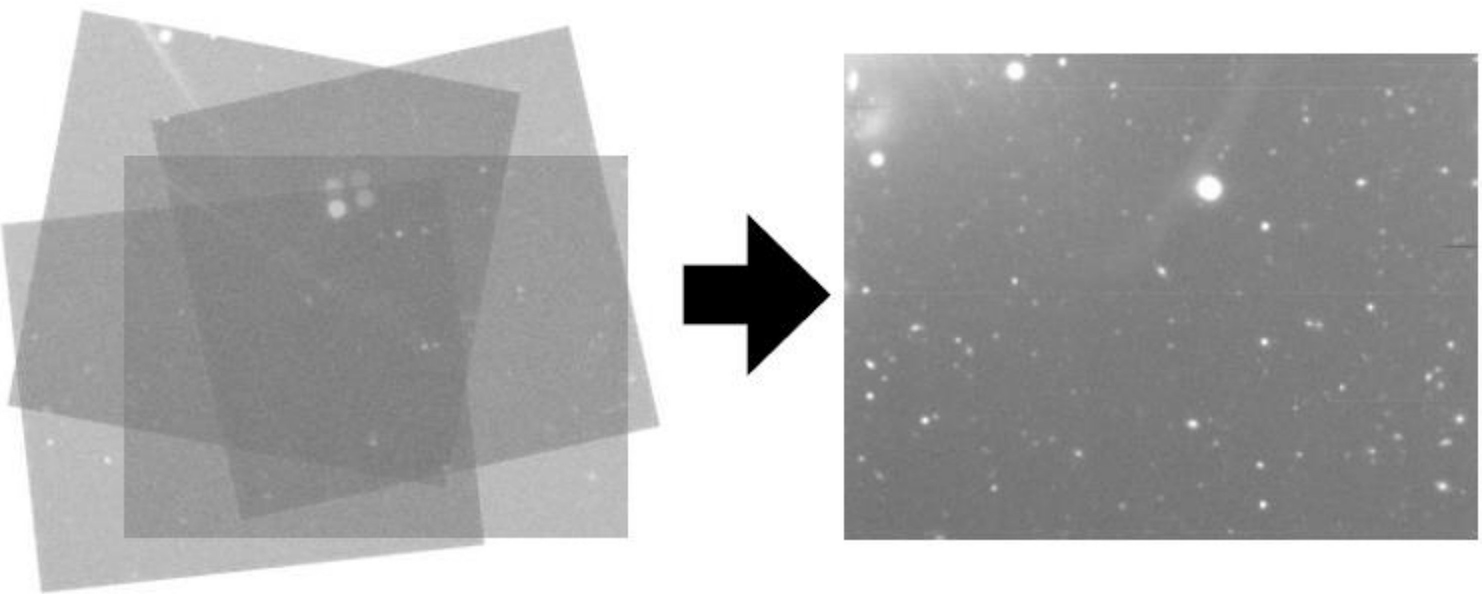
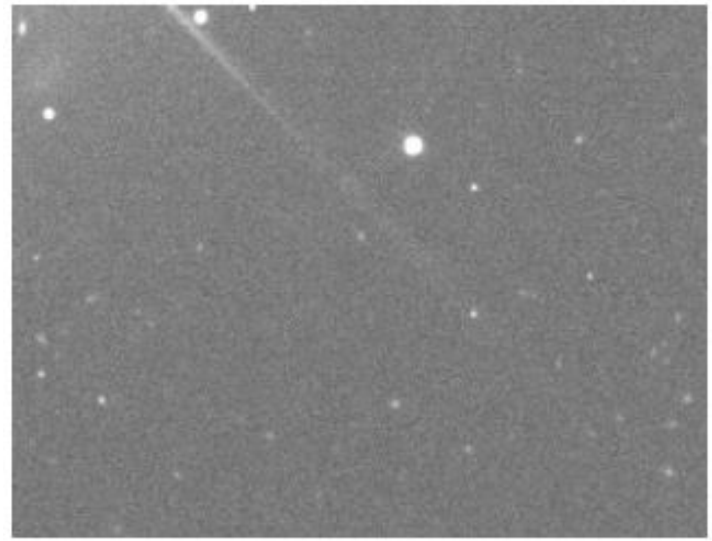
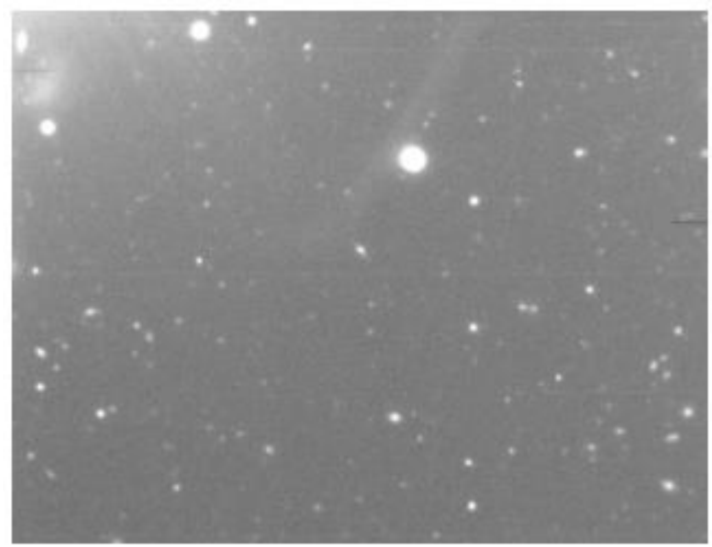


Image Stacking (Signal Averaging)

- Stacking improves SNR: makes fainter objects visible.
- Example (SDSS, Stripe 82):
  - Top: Single image, R-band
  - Bottom: 79-deep stack (~9x SNR improvement)
- Variable conditions (e.g., atmosphere, PSF, haze) mean stacking algorithm complexity can exceed a mere sum.

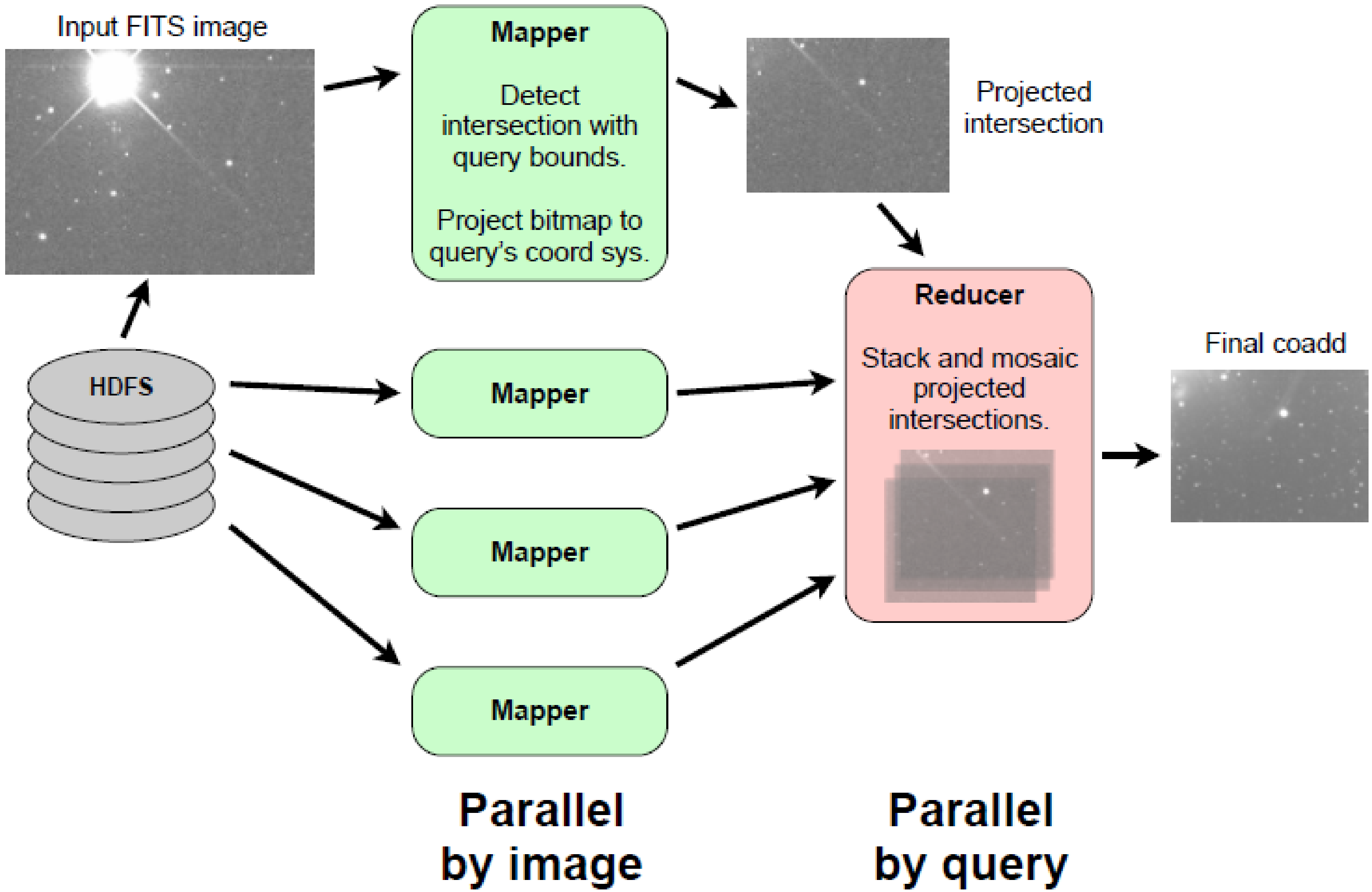




Advantages of MapReduce

- High-level problem description. No effort spent on internode communication, message-passing, etc.
- Programmed in Java (accessible to most science researchers, not just computer scientists and engineers).
- Runs on cheap commodity hardware, potentially in the cloud, e.g., Amazon's EC2.
- Scalable: 1000s of nodes can be added to the cluster with no modification to the researcher's software.
- Large community of users/support.

Coaddition in Hadoop

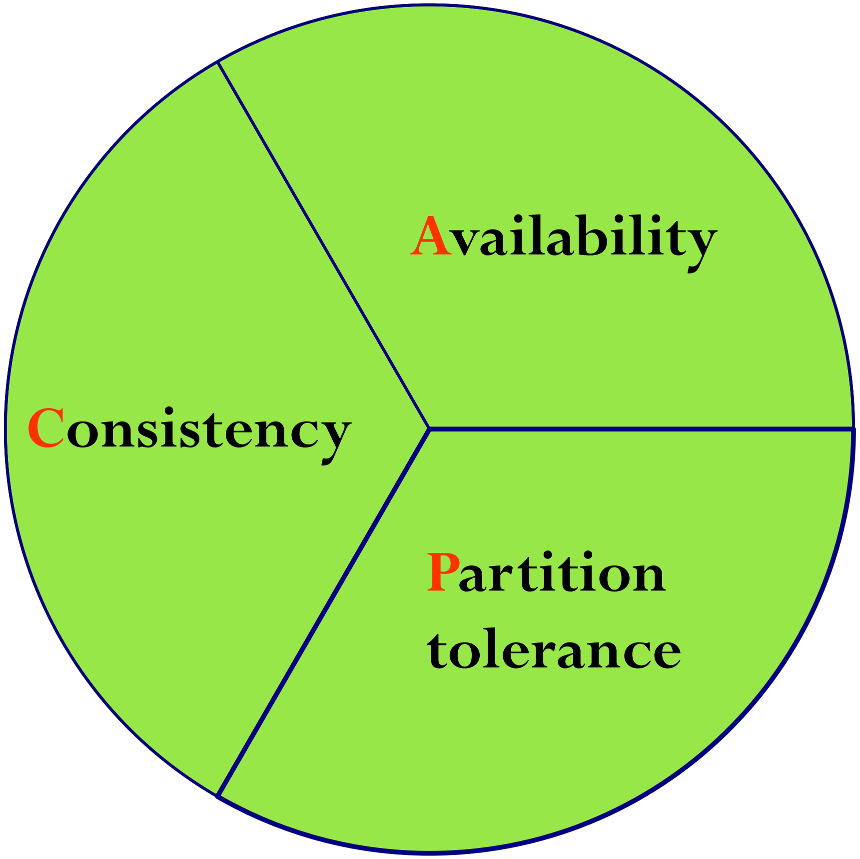


What is NoSQL?

What is NoSQL?

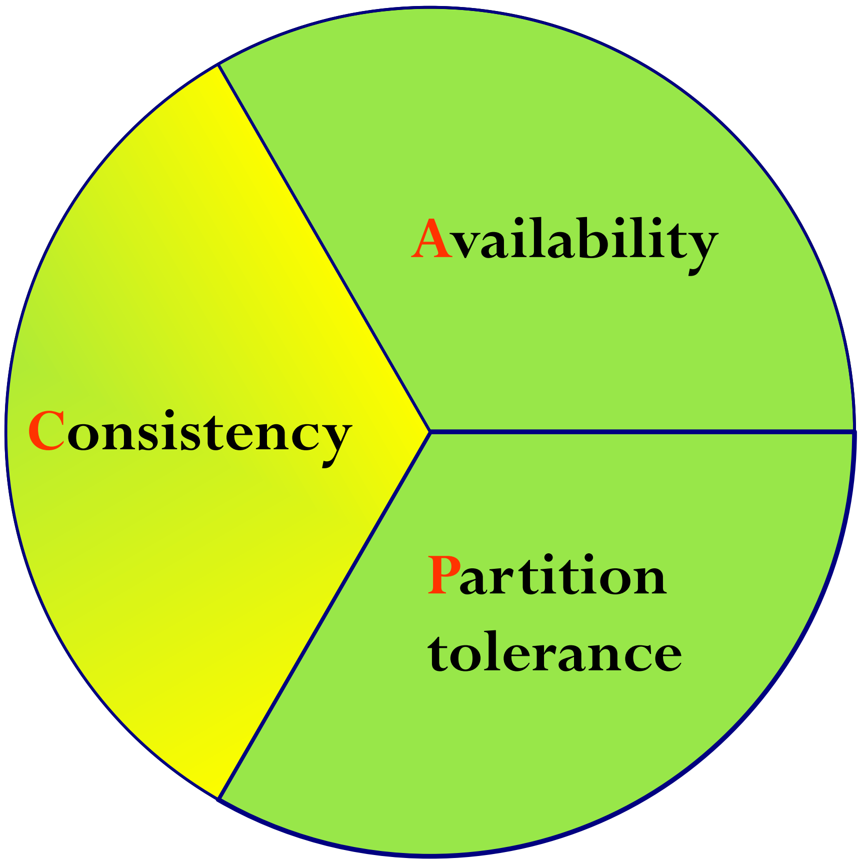
- Stands for **N**ot **O**nly **SQL**
- Class of non-relational data storage systems
- Usually do not require a fixed table schema nor do they use the concept of joins
- All NoSQL offerings relax one or more of the ACID properties (CAP theorem)
- For data storage, an RDBMS cannot be the be-all/end-all
- Just as there are different programming languages, need to have other data storage tools in the toolbox
- A NoSQL solution is more acceptable to a client now

The CAP Theorem



Theorem: You can have at most **two** of these properties for any shared-data system

The CAP Theorem

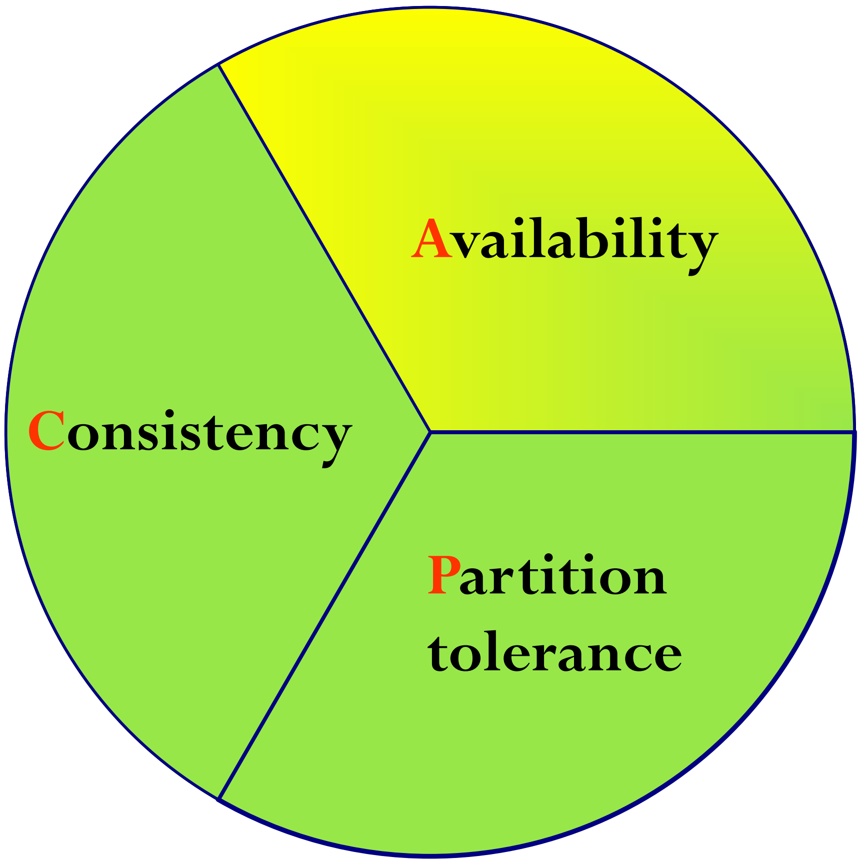


- Once a writer has written, all readers will see that write

Consistency

- Two kinds of consistency:
  - strong consistency – ACID (Atomicity Consistency Isolation Durability)
  - weak consistency – BASE (Basically Available Soft-state Eventual consistency)
    - Basically Available: The database system always seems to work!
    - Soft State: It does not have to be consistent all the time.
    - Eventually Consistent: The system will eventually become consistent when the updates propagate, in particular, when there are not too many updates.

The CAP Theorem

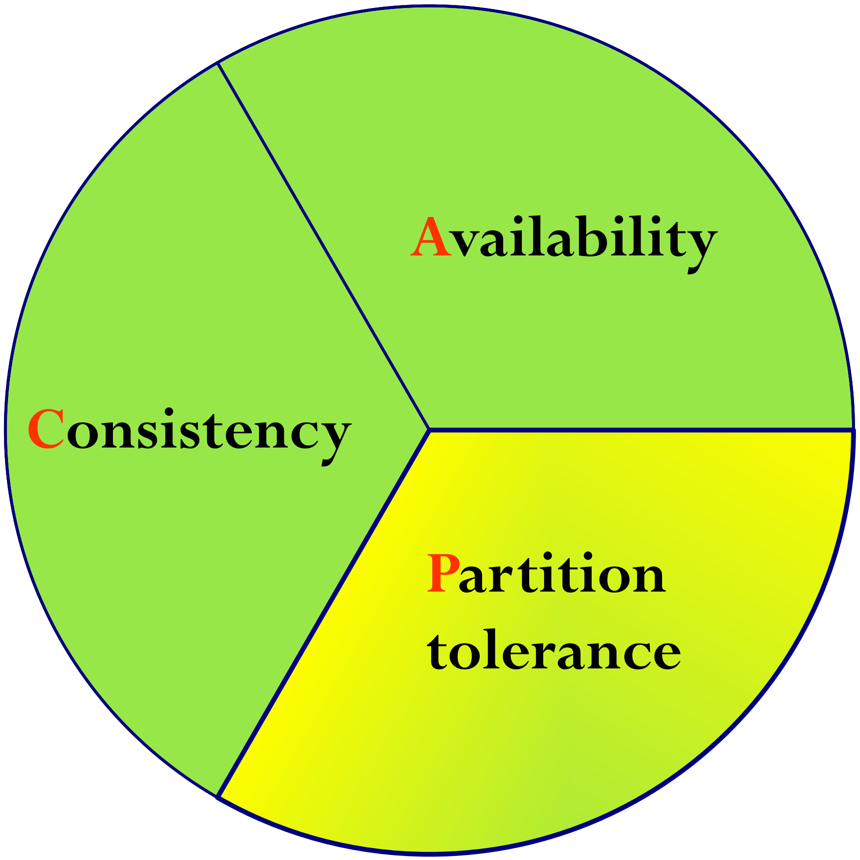


System is available during software and hardware upgrades and node failures.

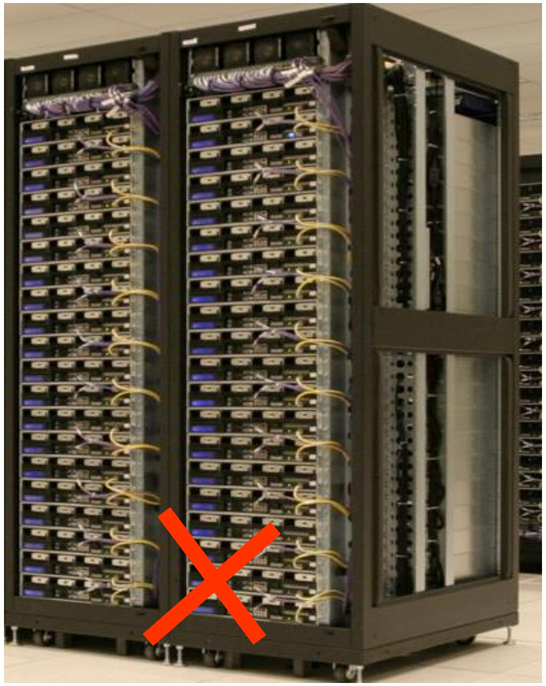
Availability

- A guarantee that every request receives a response about whether it succeeded or failed.
- Traditionally, thought of as the server/process available five 9's (99.999 %).
- However, for large node system, at almost any point in time there's a good chance that a node is either down or there is a network disruption among the nodes.

The CAP Theorem



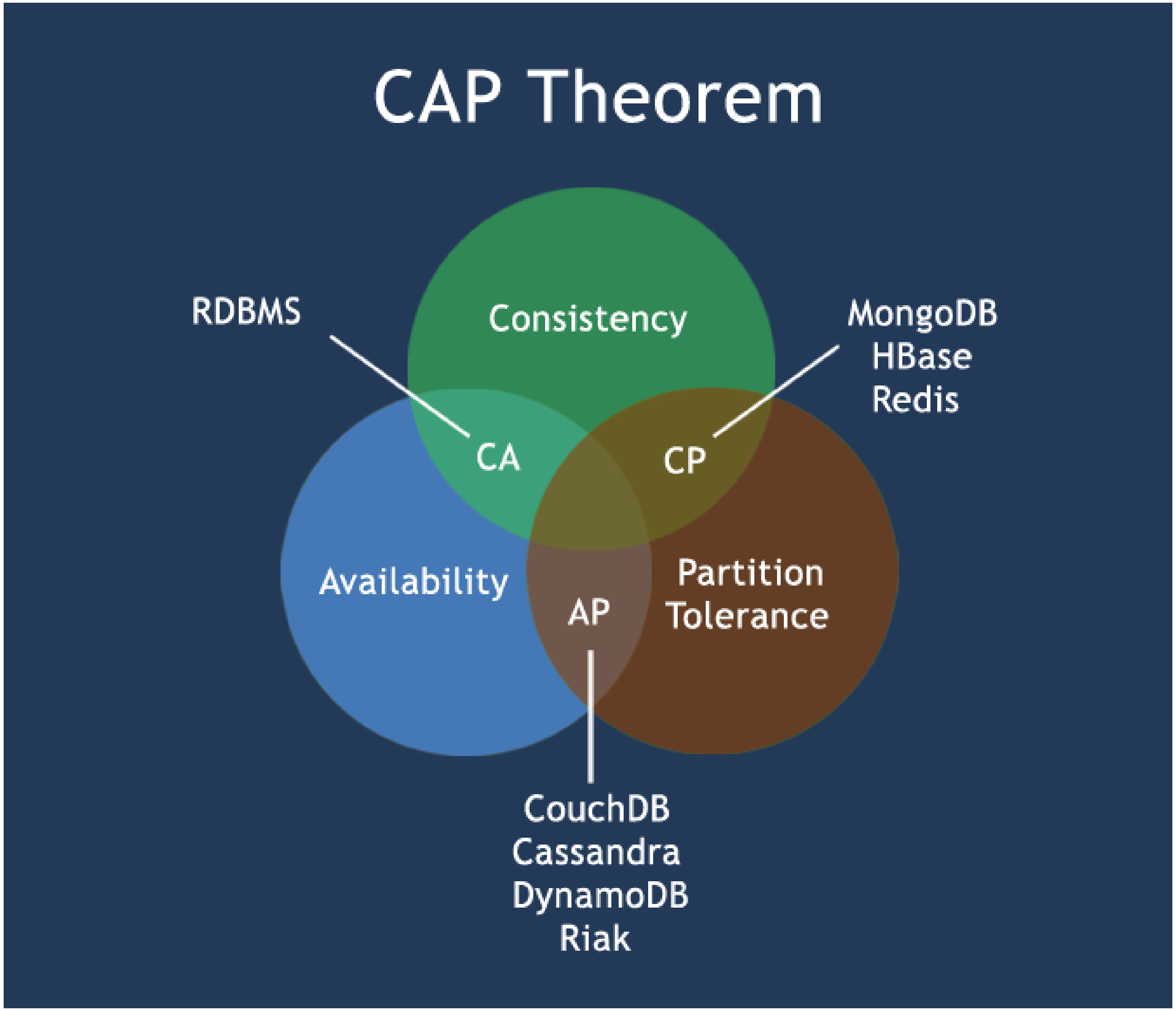
A system can continue to operate in the presence of a network partitions.



Failure is the rule

- Amazon:
  - Datacenter with 100 000 disks
  - From 6 000 to 10 000 disks fail over per year (25 disks per day)
- Sources of failures are numerous:
  - Hardware (disk)
  - Network
  - Power
  - Software
  - Software and OS updates.

The CAP Theorem



Different Types of NoSQL Systems

- Distributed Key-Value Systems - Lookup a single value for a key
  - Amazon's Dynamo
- Document-based Systems - Access data by key or by search of "document" data.
  - CouchDB
  - MongoDB
- Column-based Systems
  - Google's BigTable
  - HBase
  - Facebook's Cassandra
- Graph-based Systems - Use a graph structure
  - Google's Pregel
  - Neo4j

Use different types for different types of applications

Key-Value Pair (KVP) Stores

"Value" is stored as a "blob"

- Without caring or knowing what is inside
- Application is responsible for understanding the data

In simple terms, a NoSQL Key-Value store is a single table with two columns: one being the (Primary) Key, and the other being the Value.

Example of unstructured data for user records

Key: 1	ID: sj	First Name: Sam

Key: 2	Email: jb@gmail.com	Location: London	Age: 37

Key: 3	Facebook ID: jkirk	Password: xxx	Name: James

Each record may have a different schema

Document storage

- Records within a single table can have different structures.
- An example record from Mongo, using JSON format, might look like

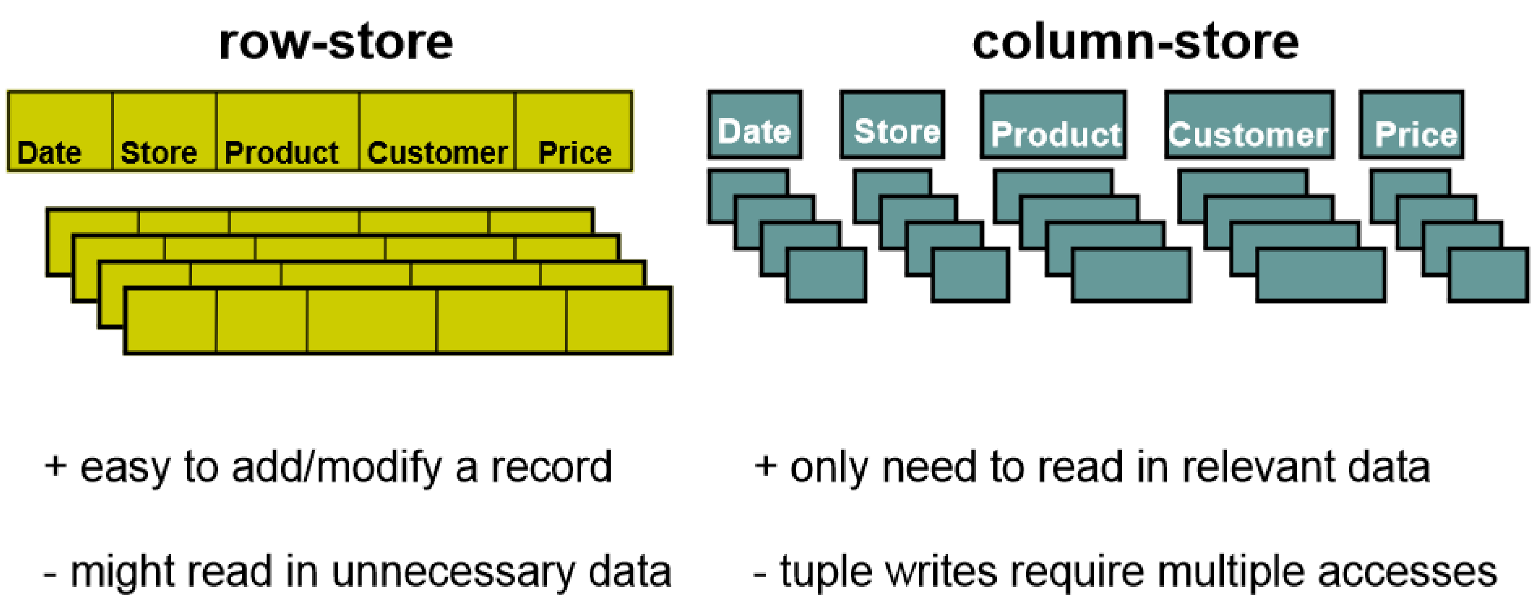
```
{
“_id” : ObjectId(“4fccbf281168a6aa3c215443″),

“first_name” : “Thomas”,
“last_name” : “Jefferson”,
“address” : {
        “street” : “1600 Pennsylvania Ave NW”,
        “city” : “Washington”,
        “state” : “DC”
}
}
```

Embedded object

- Records are called documents.
- You can also modify the structure of any document on the fly by adding and removing members from the document.
- Unlike simple key-value stores, both keys and values are fully searchable in document databases.

Column-based Stores

- Based on Google's BigTable store:
  - Each record = (row:string, column:string, time:int64)
- Distributed data storage, especially versioned data (time-stamps).
- What is a column-based store? - Data tables are stored as sections of columns of data, rather than as rows of data.



=> suitable for read-mostly, read-intensive, large data repositories

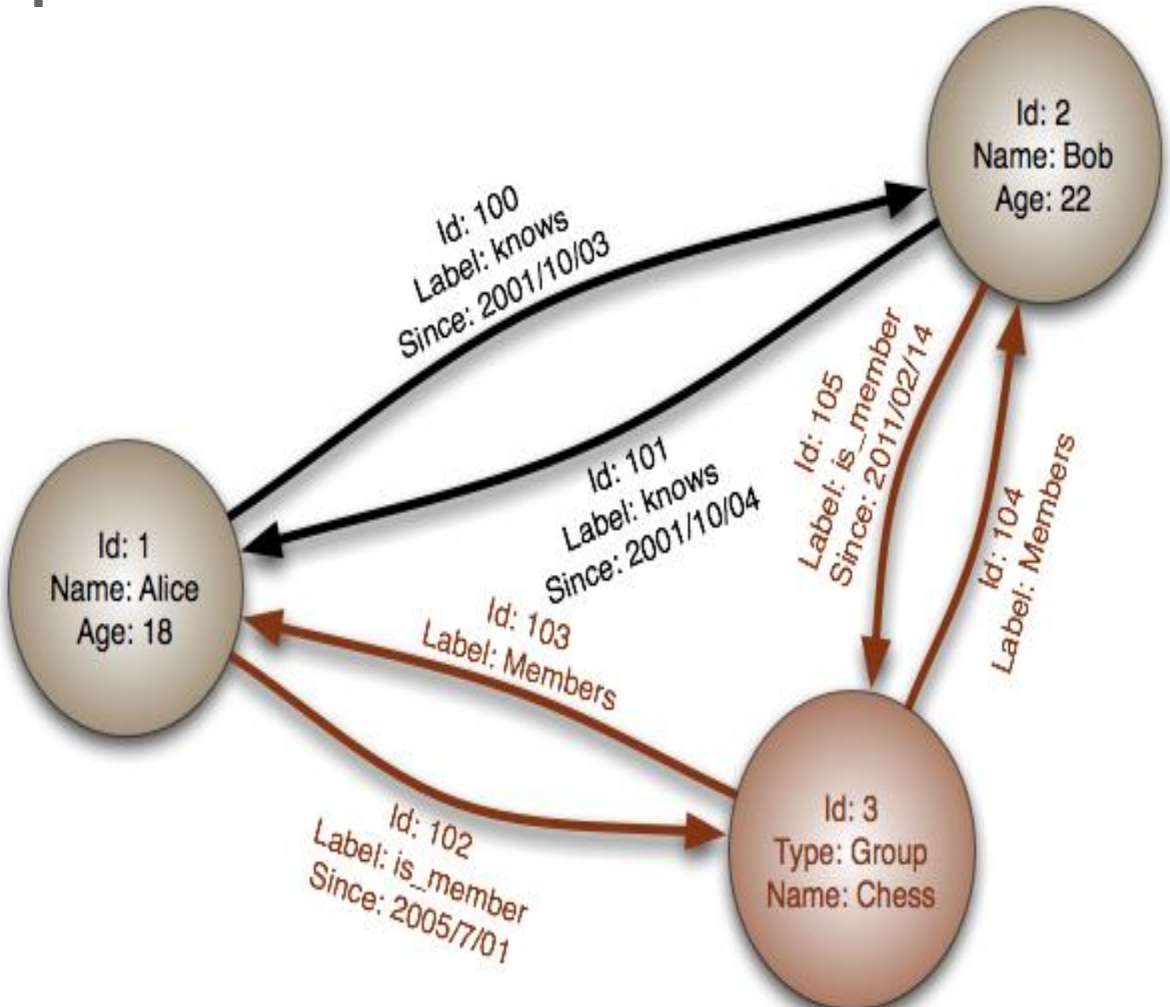
Graph Database

- Apply graph theory in the storage of information about the relationship between entries

- A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store data.

- In general, graph databases are useful when you are more interested in relationships between data than in the data itself:
  - for example, in representing and traversing social networks, generating recommendations, or conducting forensic investigations (e.g. pattern detection).

Example


Id: 2
Name: Bob
Age: 22
Id: 100
Label: knows
Since: 2001/10/03
Id: 101
Label: knows
Since: 2001/10/04
Id: 1
Name: Alice
Age: 18
Id: 105
Label: is_member
Since: 2011/02/14
Id: 104
Label: Members
Id: 103
Label: Members
Id: 102
Label: is_member
Since: 2005/7/01
Id: 3
Type: Group
Name: Chess

What is Pig?

Pig

- **In brief:**

"is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs."

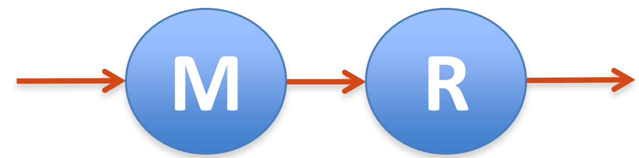
- **Top Level Apache Project**
  - http://pig.apache.org

- **Pig is an abstraction on top of Hadoop**
  - Provides high level programming language designed for data processing
  - Converted into MapReduce and executed on Hadoop Clusters

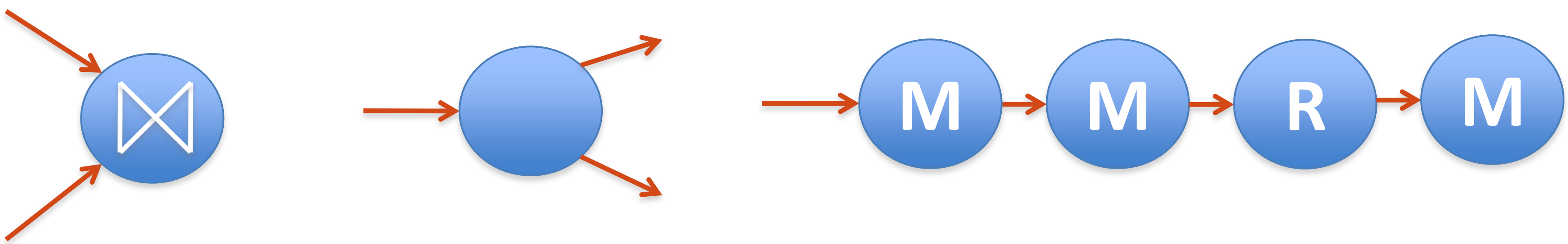
- **Pig is widely accepted and used**
  - Yahoo!, Twitter, Netflix, etc…
  - At Yahoo!, 70% MapReduce jobs are written in Pig

Disadvantages of Raw MapRaduce

1. Extremely rigid data flow



Other flows constantly hacked in



Join, Union **Split** **Chains**

2. Common operations must be coded by hand
   - Join, filter, projection, aggregates, sorting, distinct

3. Semantics hidden inside map-reduce functions
   - Difficult to maintain, extend, and optimize
   - Resulting code is difficult to reuse and maintain; shifts focus and attention away from data analysis

Pig and MapReduce

- **MapReduce requires programmers**
  - Must think in terms of map and reduce functions
  - More than likely will require Java programmers
- **Pig provides high-level language that can be used by**
  - Analysts
  - Data Scientists
  - Statisticians
  - Etc…
- **Originally implemented at Yahoo! to allow analysts to access data**

Pig's Features

- **Main operators:**
  - Join Datasets
  - Sort Datasets
  - Filter
  - Data Types
  - Group By
  - User Defined Functions
  - Etc..

- **Example:**

```
>movies = LOAD '/home/movies_data.csv' USING PigStorage(',') as
            (id,name,year,rating,duration);
>movies_greater_than_four = FILTER movies BY (float)rating>4.0;
>DUMP movies_greater_than_four;
```

What is Hive?

Hive

- Data Warehousing Solution built on top of Hadoop
- Provides SQL-like query language named HiveQL
  - Minimal learning curve for people with SQL expertise
  - Data analysts are target audience
- Early Hive development work started at Facebook in 2007
- Today Hive is an Apache project under Hadoop
  - http://hive.apache.org

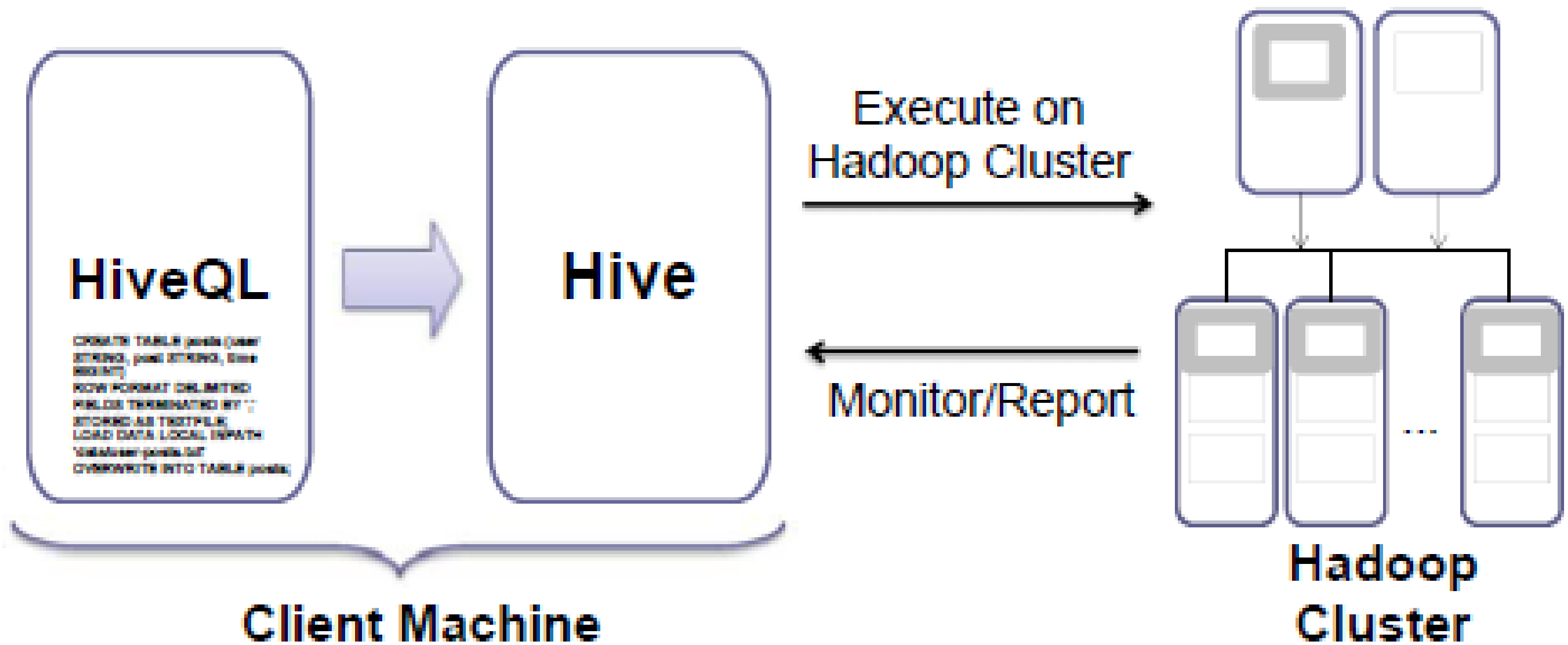
Advantages and Drawbacks

- Hive provides
  - Ability to bring structure to various data formats
  - Simple interface for ad hoc querying, analyzing and summarizing large amounts of data
  - Access to files on various data stores such as HDFS and Hbase

- Hive does not provide
  - Hive does not provide low latency or realtime queries
  - Even querying small amounts of data may take minutes
  - Designed for scalability and ease-of-use rather than low latency responses

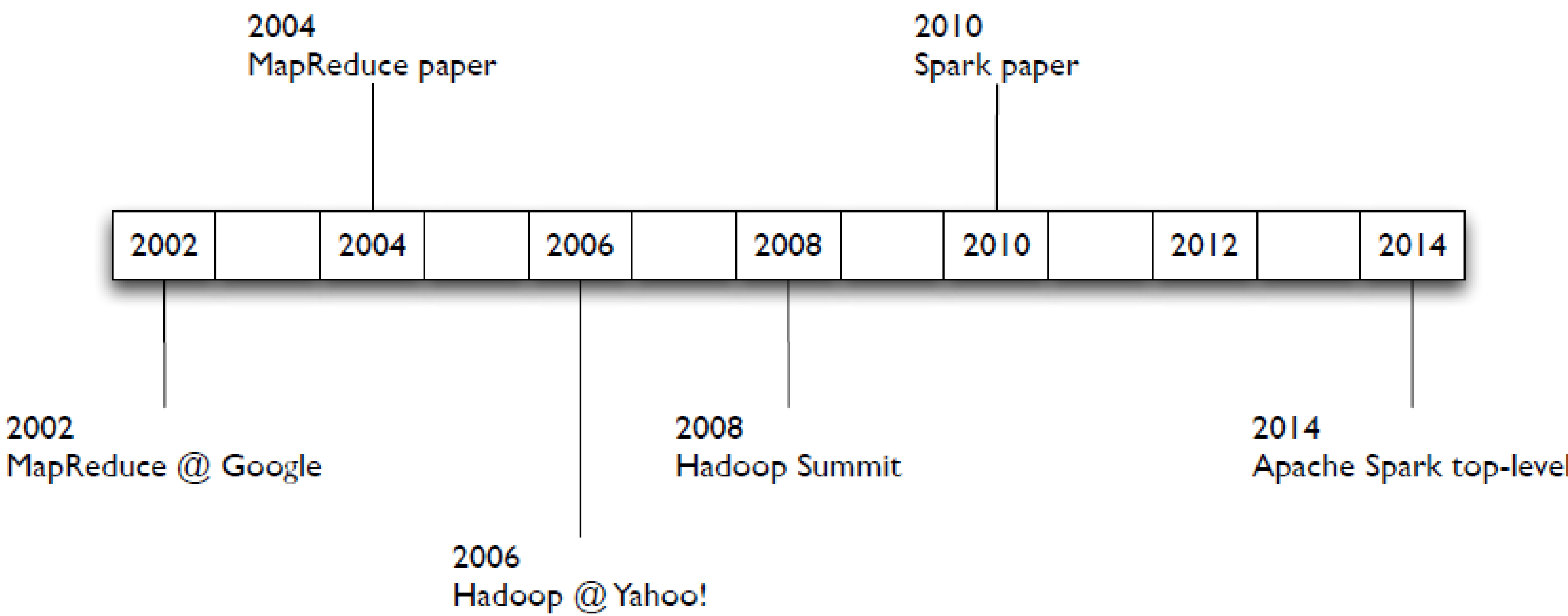
Hive

- Translates HiveQL statements into a set of MapReduce Jobs which are then executed on a Hadoop Cluster

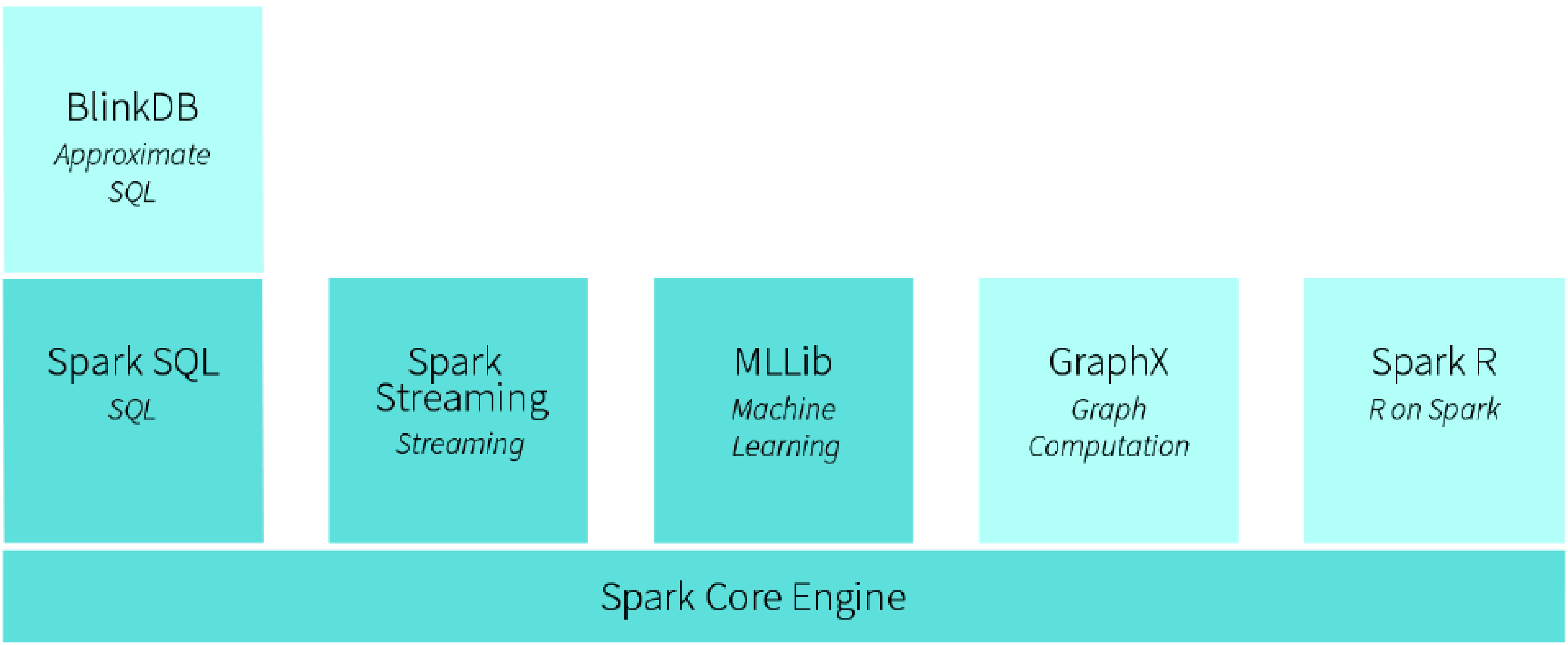


What is Spark?

A Brief History: Spark

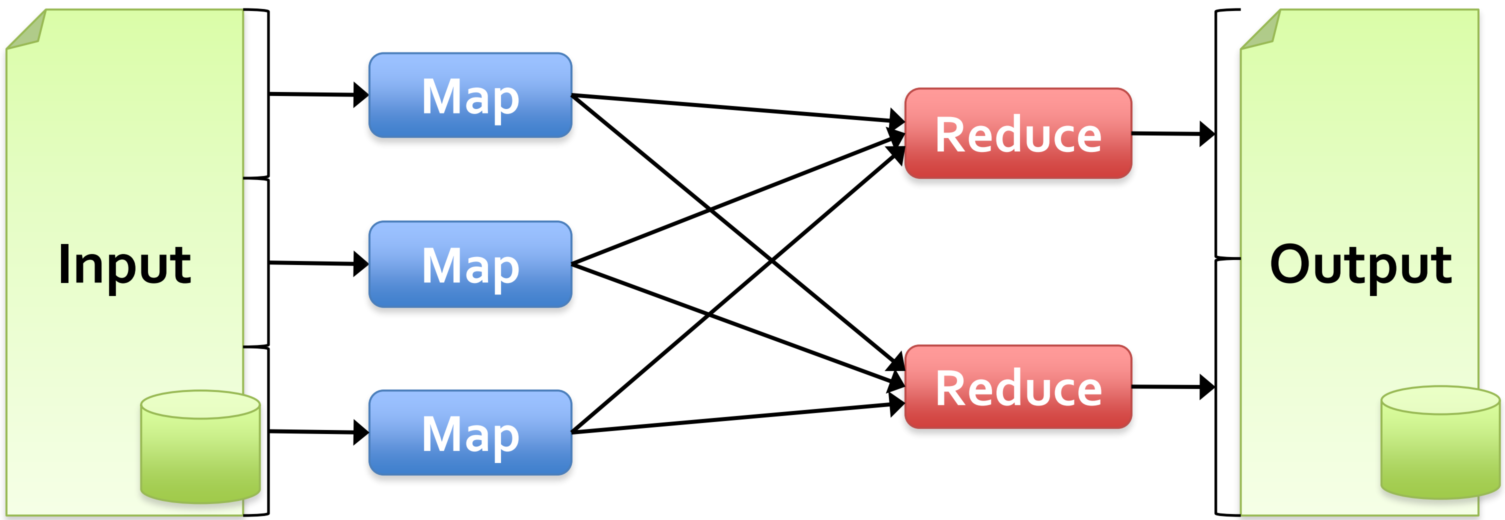


A general view of Spark



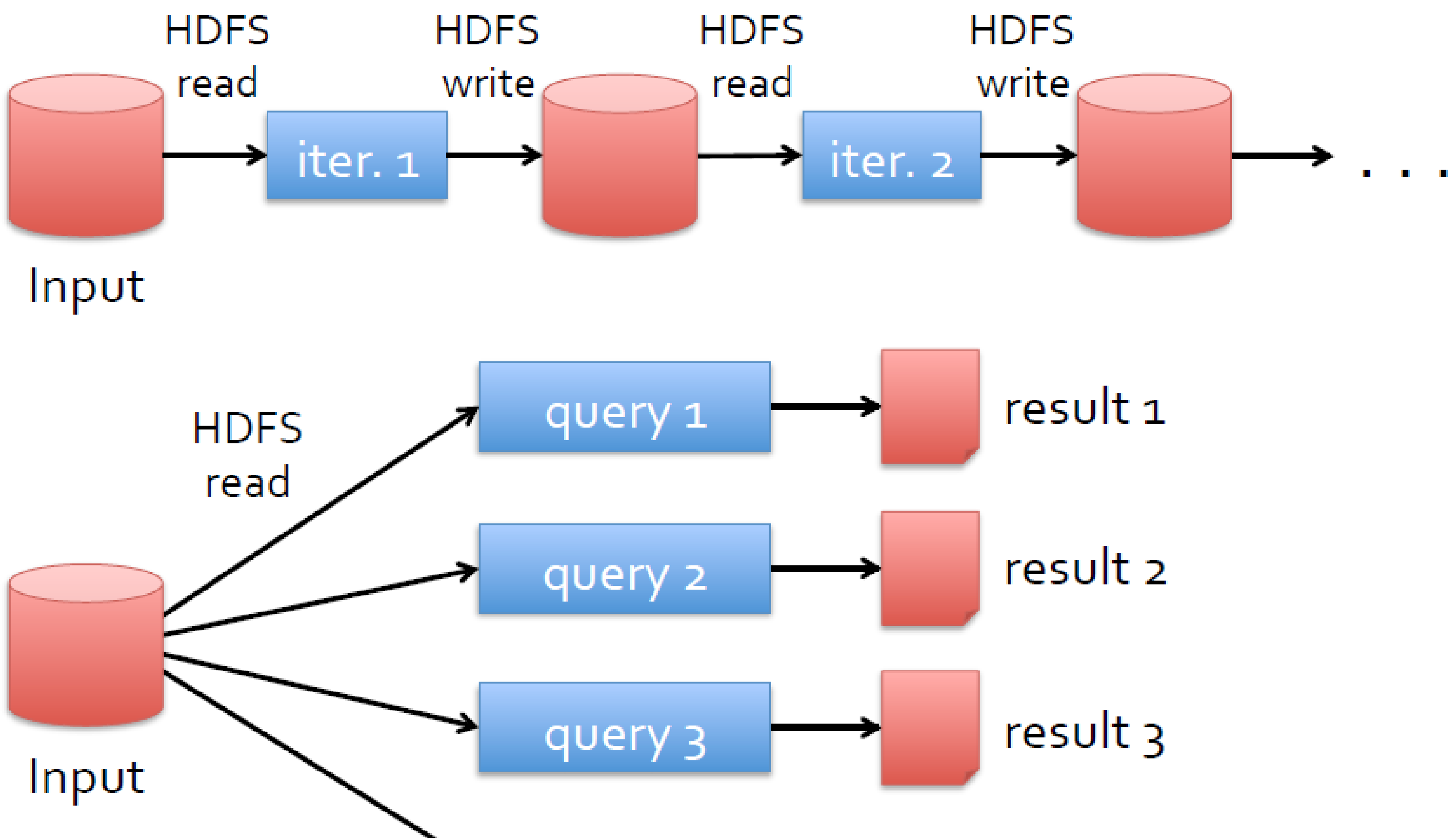
Current programming models

- Current popular programming models for clusters transform data flowing from stable storage to stable storage
- E.g., MapReduce:



Benefits of data flow: runtime can decide where to run tasks and can automatically recover from failures

MapReduce I/O



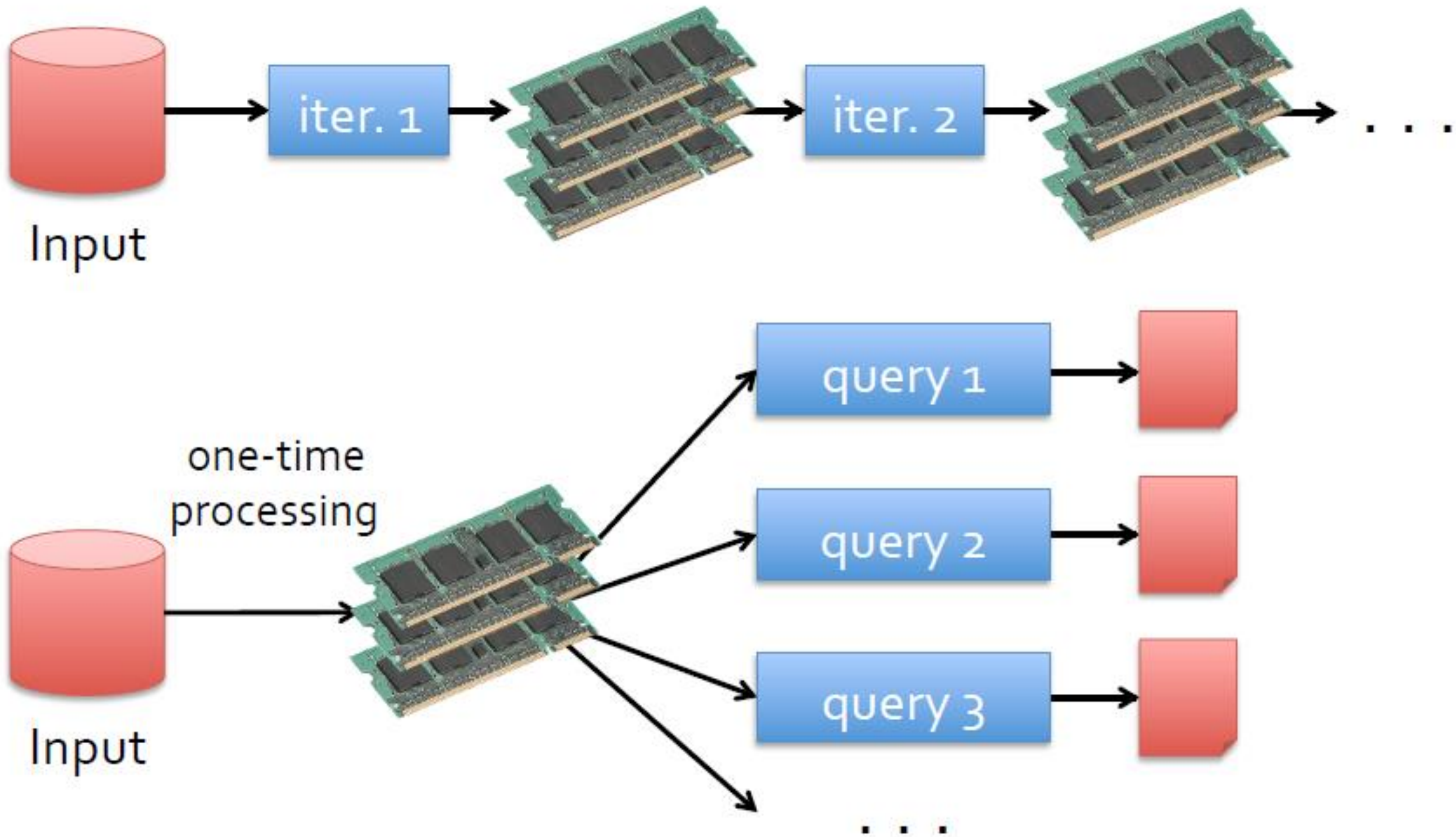
Slow due to replication and disk I/O,
but necessary for fault tolerance

Spark

- Acyclic data flow is a powerful abstraction, but is not efficient for applications that repeatedly reuse a **working set** of data:
  - Iterative algorithms (many in machine learning)
  - Interactive data mining tools (R, Excel, Python)

- Spark makes working sets a first-class concept to efficiently support these apps.

Goal: Sharing at Memory Speed



10-100× faster than network/disk, but how to get FT?

Resilient Distributed Dataset (RDD)

- Provide distributed memory abstractions for clusters to support apps with working sets.

- Retain the attractive properties of MapReduce:
  - Fault tolerance (for crashes & stragglers)
  - Data locality
  - Scalability

Solution: augment data flow model with "resilient distributed datasets" (RDDs)

Programming Model with RDD

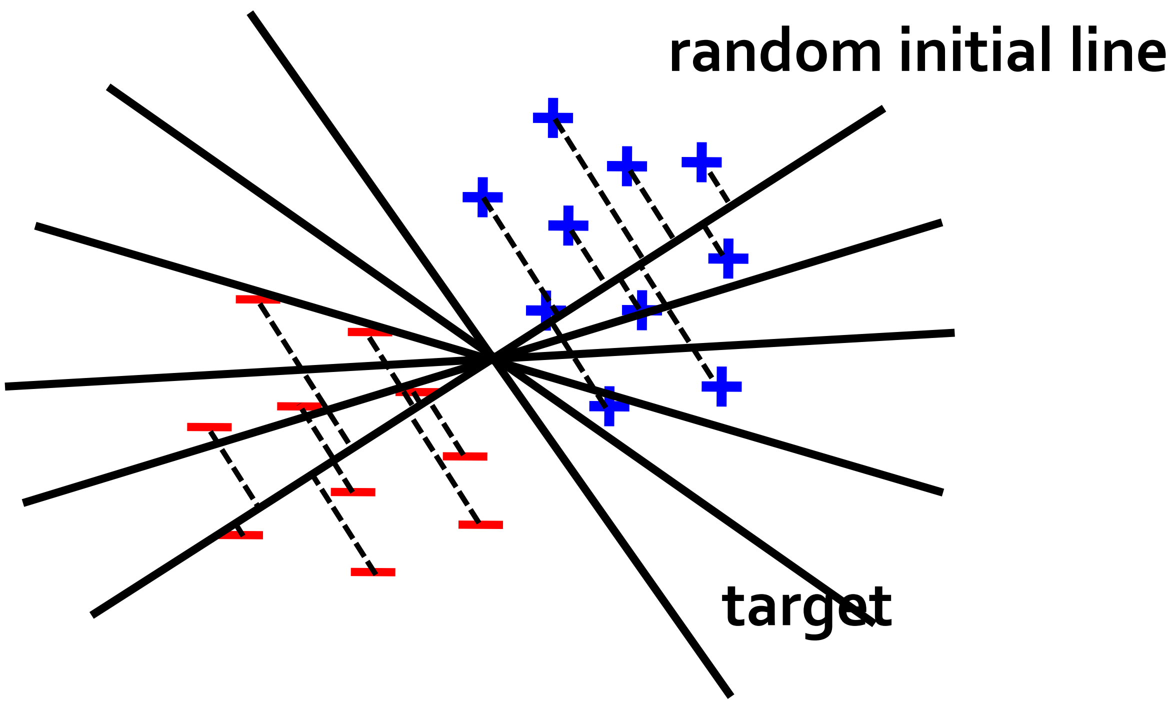
- Resilient distributed datasets (RDDs)
  - Immutable collections partitioned across cluster that can be rebuilt if a partition is lost
  - Created by transforming data in stable storage using data flow operators (map, filter, group-by, …)
  - Can be cached across parallel operations

- Parallel operations on RDDs
  - Reduce, collect, count, save, …

- Restricted shared variables
  - Accumulators, broadcast variables

Example: Logistic Regression

- Goal: find best line separating two sets of points



Logistic Regression (SCALA Code)

```
val data =
spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y *
p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Conclusion

Conclusion

- Data storage needs are rapidly increasing

→ Hadoop has become the de-facto standard for handling these massive data sets.

- Storage of Big Data requires new storage models

→ NoSQL solutions.

- Parallel processing of Big Data requires a new programming paradigm

→ MapReduce programming model.

- "Big data" is moving beyond one-passbatch jobs, to low-latency apps that need datasharing

→ Apache Spark is an alternative solution.