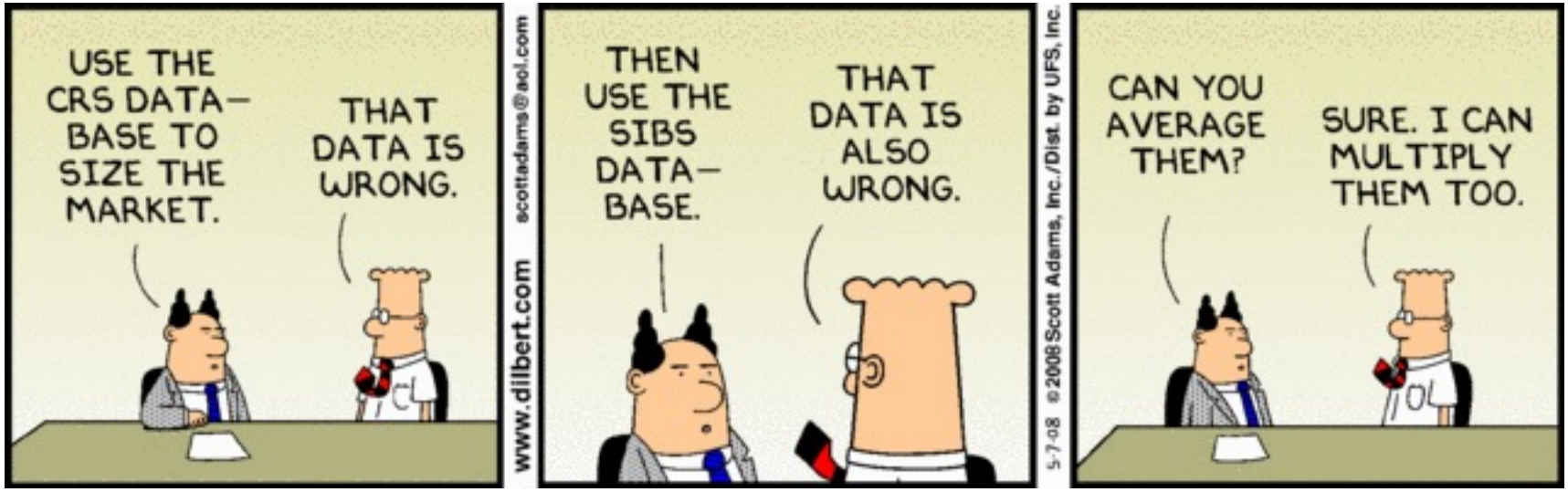
Introduction to Database Systems

Intro to Big Data Management

Eleni Tzirita Zacharatou

Based on slides from Björn Þór Jónsson

Week 12



First Things First: Rest of Class

- Trial exam next Week 13 (May 7)
  - 08:15 – 12:15: Quiz on LearnIT
    - Piazza/email (elza@itu.dk) if you have questions
- Homework 4 is out (deadline: May 7)
  - Individual work
- Questions: Piazza/email
- Course Evaluation Survey is open

Lecture Outline

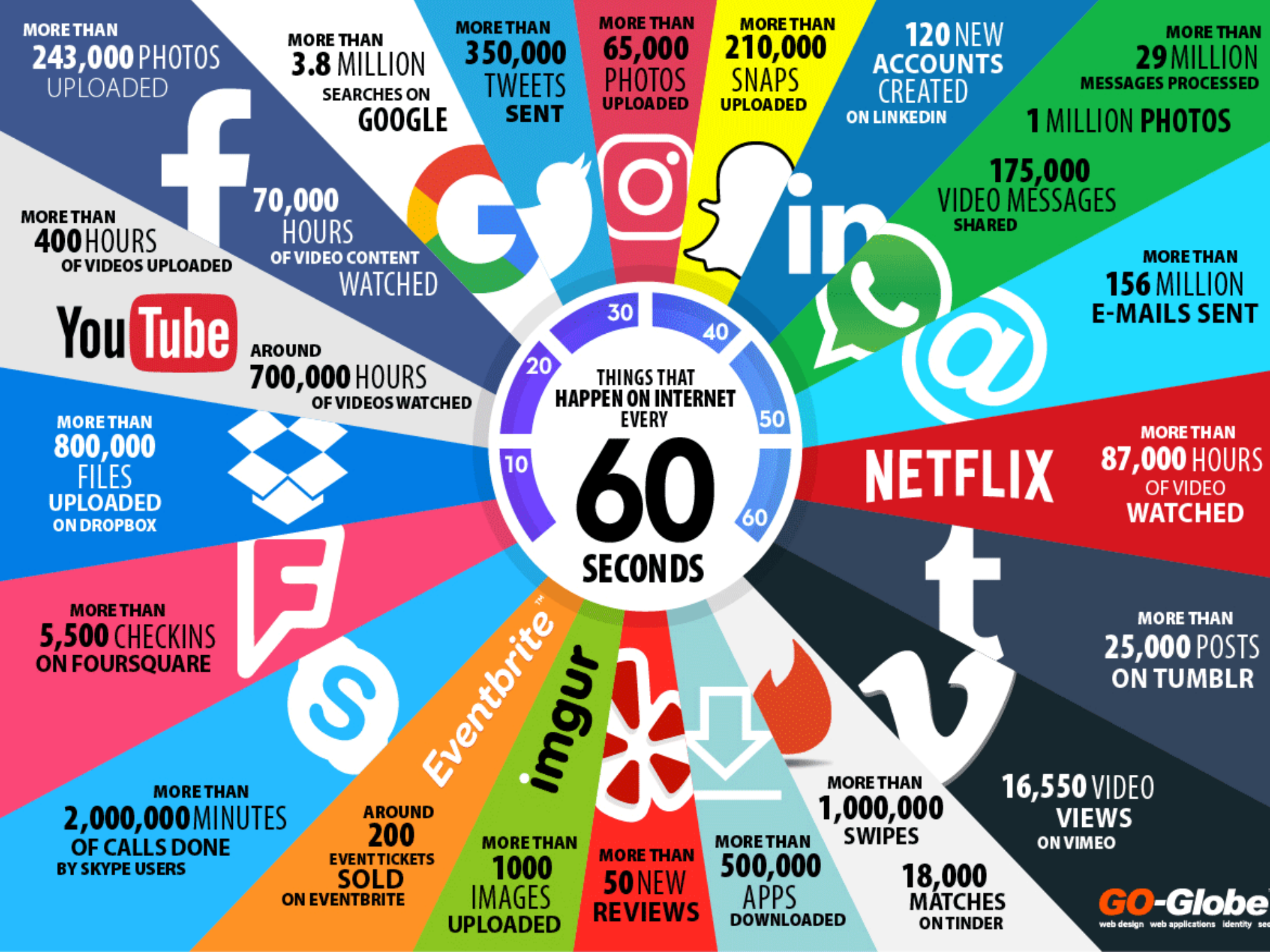
Learning Outcome 6: Discuss the pros and cons of different classes of data systems for modern analytics and data science applications.

- What are modern analytics and data science applications?
- What are their access patterns and processing requirements?
- How well/badly do the classes of data systems covered so far support these access patterns?
- MapReduce/Spark: Their properties and their support for these access patterns

Lecture Outline

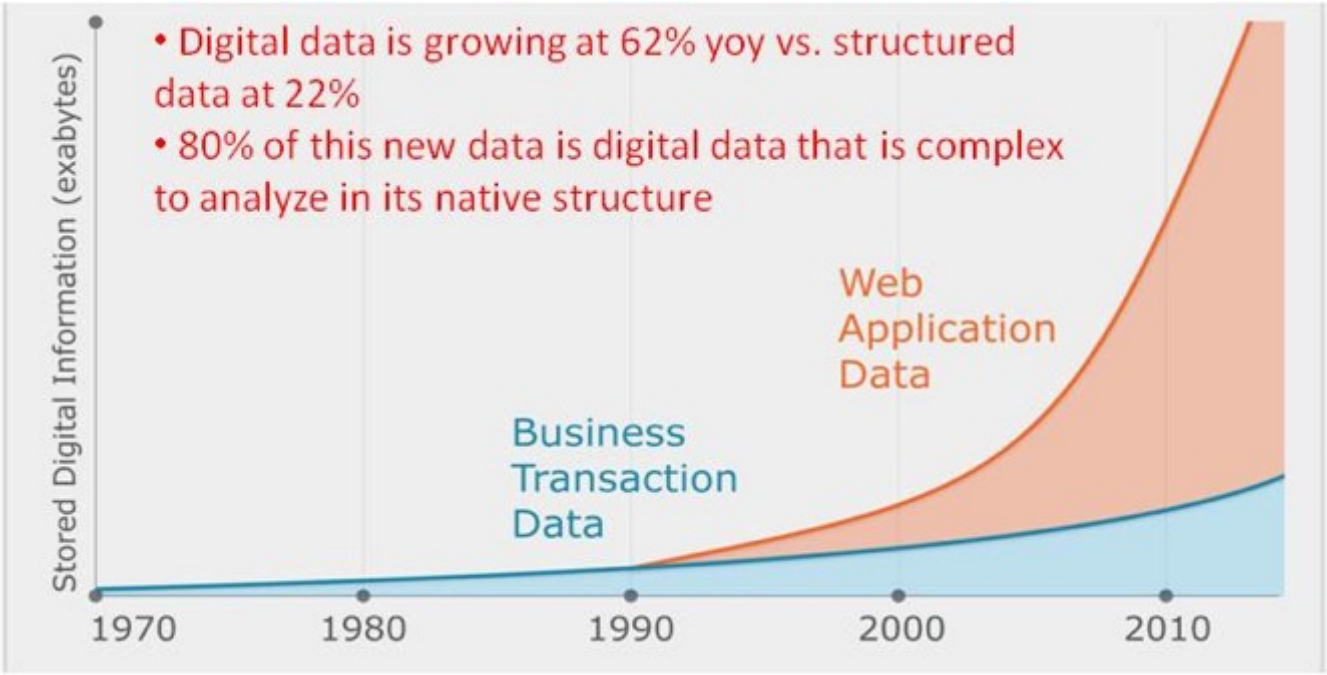
Learning Outcome 6: Discuss the pros and cons of different classes of data systems for modern analytics and data science applications.

- What are modern analytics and data science applications?
- What are their access patterns and processing requirements?
- How well/badly do the classes of data systems covered so far support these access patterns?
- MapReduce/Spark: Their properties and their support for these access patterns

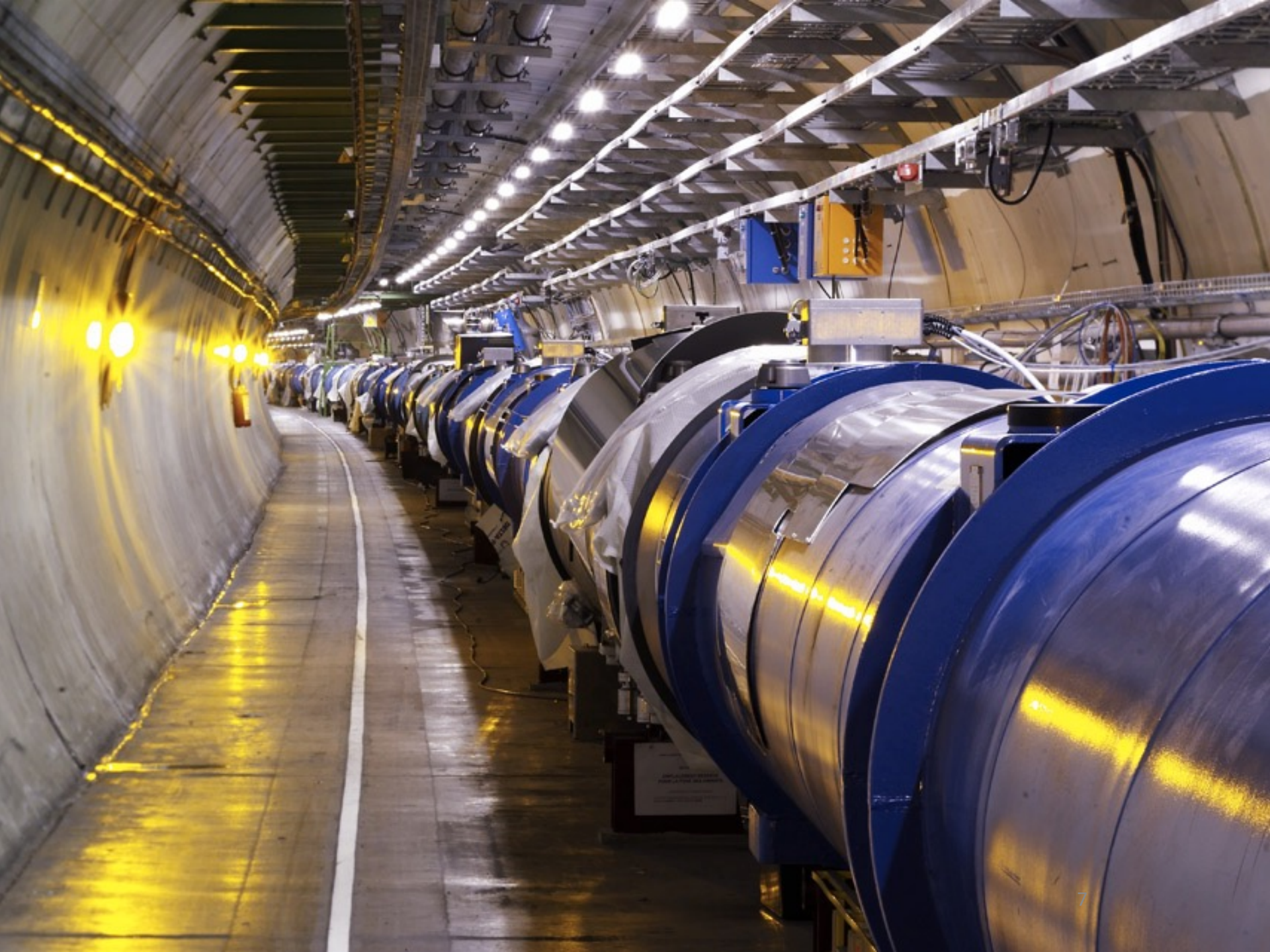

MORE THAN 243,000 PHOTOS UPLOADED
MORE THAN 3.8 MILLION SEARCHES ON GOOGLE
MORE THAN 350,000 TWEETS SENT
MORE THAN 65,000 PHOTOS UPLOADED
MORE THAN 210,000 SNAPS UPLOADED
120 NEW ACCOUNTS CREATED ON LINKEDIN
MORE THAN 29 MILLION MESSAGES PROCESSED
1 MILLION PHOTOS
175,000 VIDEO MESSAGES SHARED
MORE THAN 156 MILLION E-MAILS SENT
70,000 HOURS OF VIDEO CONTENT WATCHED
MORE THAN 400 HOURS OF VIDEOS UPLOADED
YouTube
AROUND 700,000 HOURS OF VIDEOS WATCHED
THINGS THAT HAPPEN ON INTERNET EVERY 60 SECONDS
10
20
30
40
50
60
MORE THAN 800,000 FILES UPLOADED ON DROPBOX
NETFLIX
MORE THAN 87,000 HOURS OF VIDEO WATCHED
MORE THAN 5,500 CHECKINS ON FOURSQUARE
MORE THAN 25,000 POSTS ON TUMBLR
MORE THAN 2,000,000 MINUTES OF CALLS DONE BY SKYPE USERS
AROUND 200 EVENT TICKETS SOLD ON EVENTBRITE
Eventbrite
imgur
MORE THAN 1000 IMAGES UPLOADED
MORE THAN 50 NEW REVIEWS
MORE THAN 500,000 APPS DOWNLOADED
MORE THAN 1,000,000 SWIPES
18,000 MATCHES ON TINDER
16,550 VIDEO VIEWS ON VIMEO
GO-Globe
web design web applications identity

Beyond Relational Data!

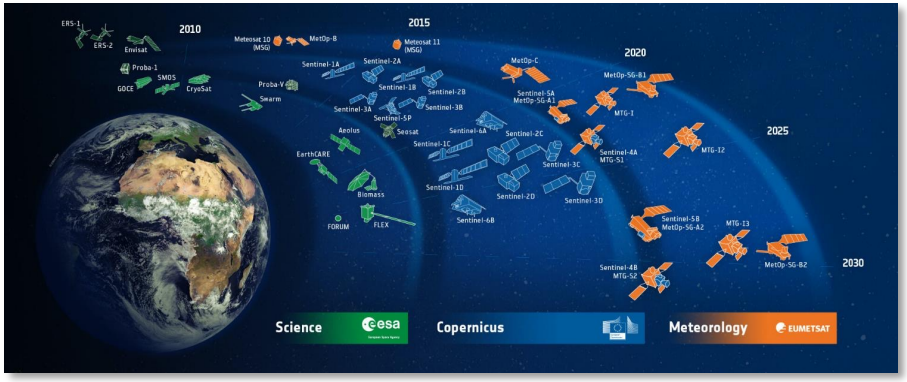
- Keep track of **all** the history
- Keep track of **all** interactions, also low-level
- Keep track of **all** data: media, text data, logs, …



Source: https://practicalanalytics.wordpress.com/2011/11/06/explaining-hadoop-to-management-whats-the-big-data-deal/



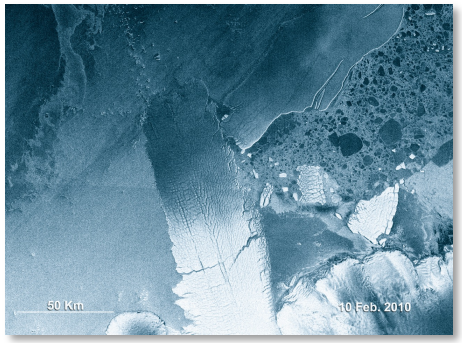
Earth Observation Big Data Analytics



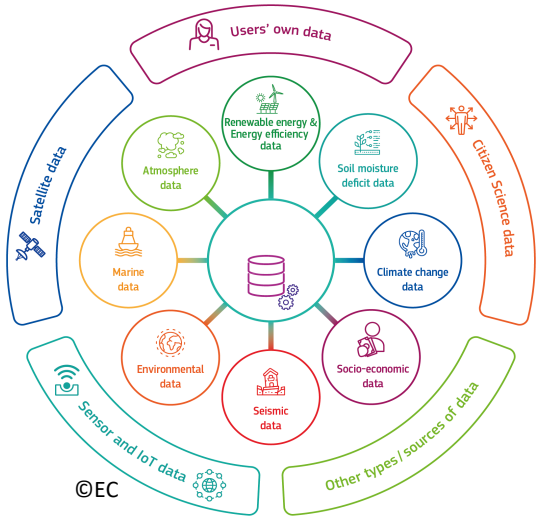
©ESA

- Growing EO data (petabytes!)
- Diverse EO applications
- Growing heterogeneity

Agricultural monitoring

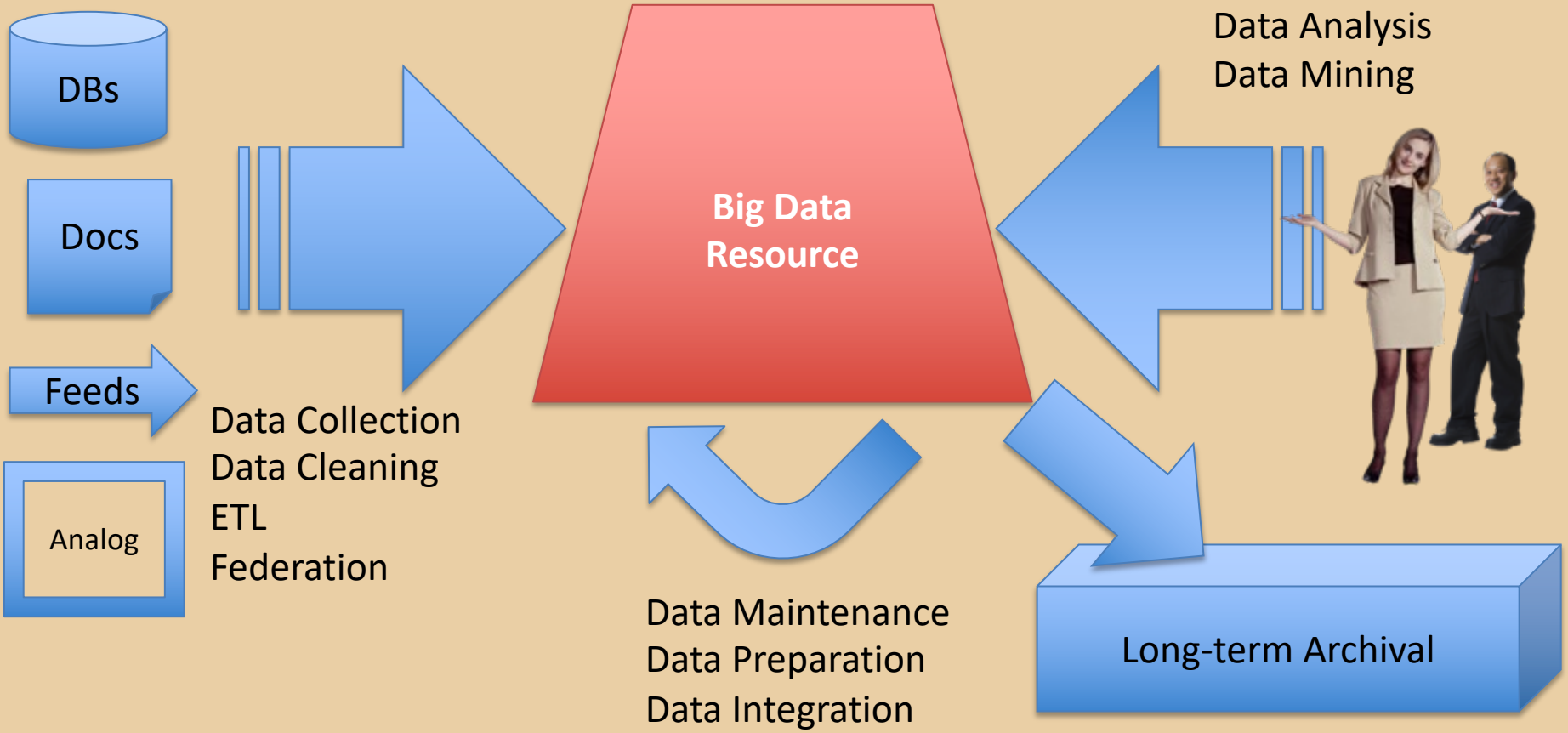


Climate change monitoring



©EC

High Data Heterogeneity



Big data is not a product, but a collection of **processes**

Big Data and Five "V"s

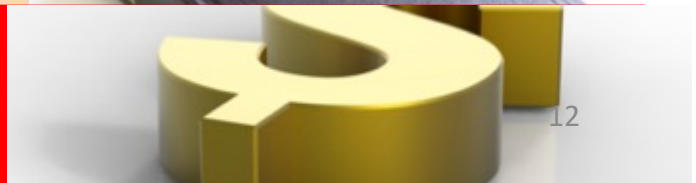
Volume

Velocity

Variety

Veracity

Value



Space is big. You just won't believe how vastly, hugely, mind-bogglingly big it is. I mean, you may think it's a long way down the road to the chemist's, but that's just peanuts to space.

Douglas Adams
The Hitch Hiker's Guide to the Galaxy

Volume

Image credit: https://www.123rf.com/photo_25981580_a-collage-of-travel-photos-from-around-the-world.html

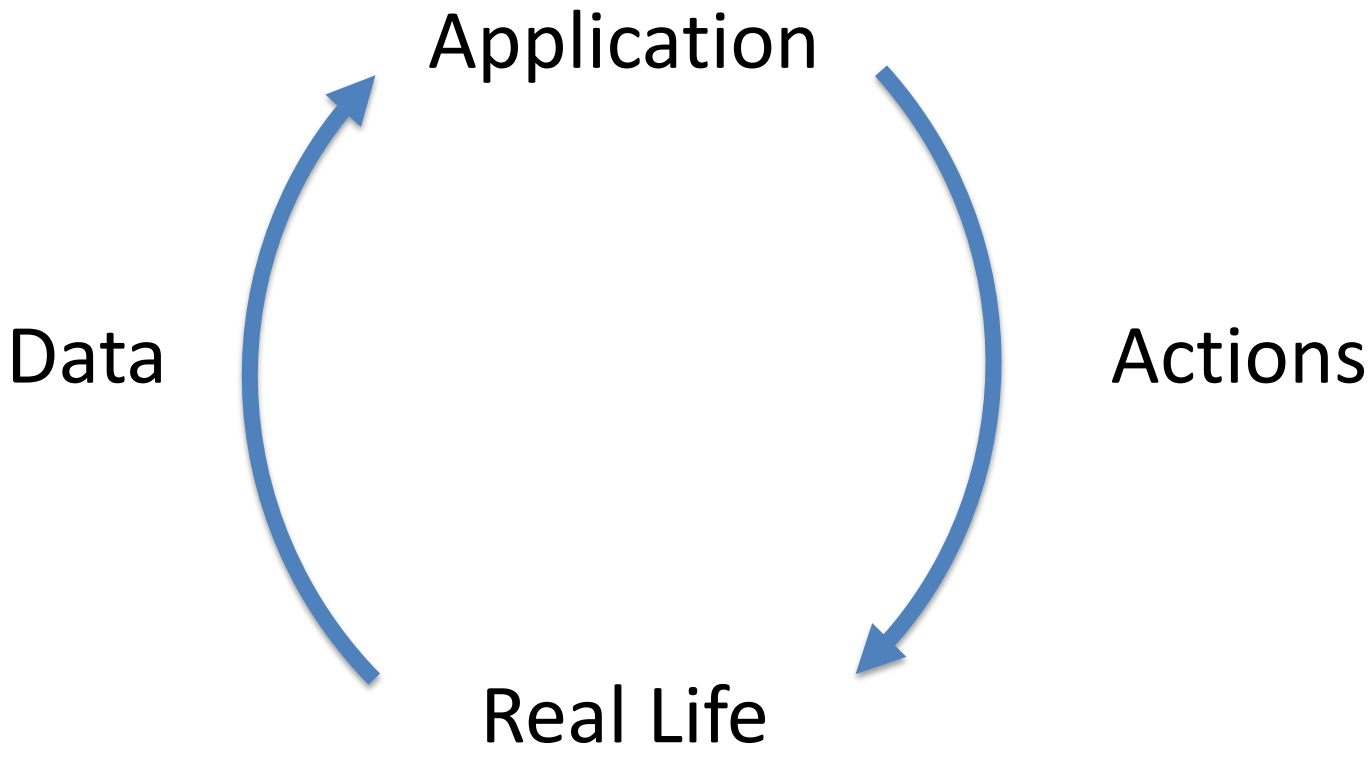
Volume

“Yesterday”:
10K images

Today:
100M images

Volume

Two Sides of Velocity



- Structured Data
  - Business data in RDBMSs, metadata (often)
  - Highly regular and repeating patterns
- **Semi-Structured Data**
  - Log data, literature (books), comment threads (?)
  - Some structure, but also much data w/o structure
- Unstructured Data
  - Multimedia content, free text (?), comment threads (?)
  - No (or little) structure
  - Try to extract some structure from this data to make it semi-structured!

Variety

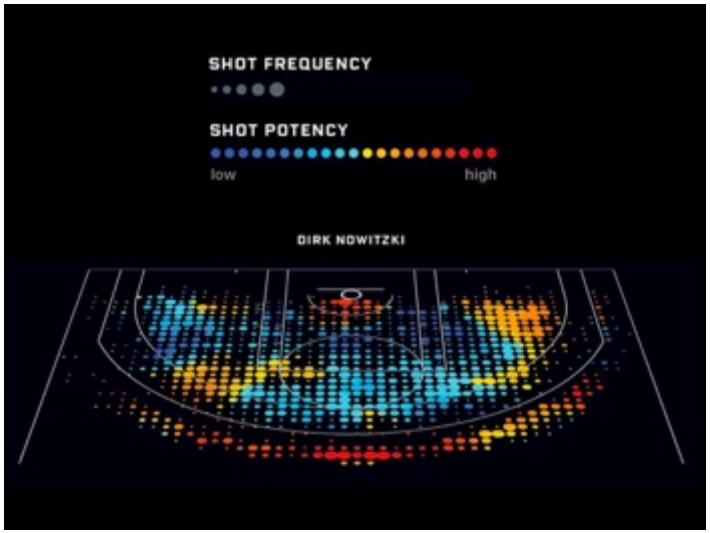
The Blind Men and the Elephant

Six blind men were asked to determine what an elephant looked like by feeling different parts of the elephant's body. The blind man who feels a leg says the elephant is like a **pillar**; the one who feels the tail says the elephant is like a **rope**; the one who feels the trunk says the elephant is like a **tree branch**; the one who feels the ear says the elephant is like a **hand fan**; the one who feels the belly says the elephant is like a **wall**; and the one who feels the tusk says the elephant is like a **solid pipe**.

http://en.wikipedia.org/wiki/Blind_men_and_an_elephant

Know Your Data

- Take time to understand the **whole** data
  - Value ranges, distributions
  - Plots, descriptive statistics, ...



- Estimate the outcome of your analysis
  - Best way to know whether the final outcome is correct

Veracity

Data is of Low Quality!

Impact of poor data quality?

- Erroneous data costs US businesses $600 billion/year
- In analytics projects, data cleaning takes 30-80% of time and budget
- Data quality tools market is growing at 16% annually, way over the 7% average for other IT segments

How much data is erroneous?

- Enterprise data error rates: average of 1-5%, some > 30%
- Only 1/3rd of XML Web documents with XSD/DTD are valid, 14% even lack well-formedness

Veracity

Two Sides of Value

Value

Lecture Outline

Learning Outcome 6: Discuss the pros and cons of different classes of data systems for modern analytics and data science applications.

- What are modern analytics and data science applications?
- What are their access patterns and processing requirements?
- How well/badly do the classes of data systems covered so far support these access patterns?
- MapReduce/Spark: Their properties and their support for these access patterns

Access Patterns to Data on Disk

- Data collections typically consist of large files of semi-structured data
- Too large for a single server
  → **distributed storage**
- Read the whole collection
  → **large sequential scans**
  - Typically in any order
  - Sometimes repeatedly (e.g., clustering)

Processing Requirements

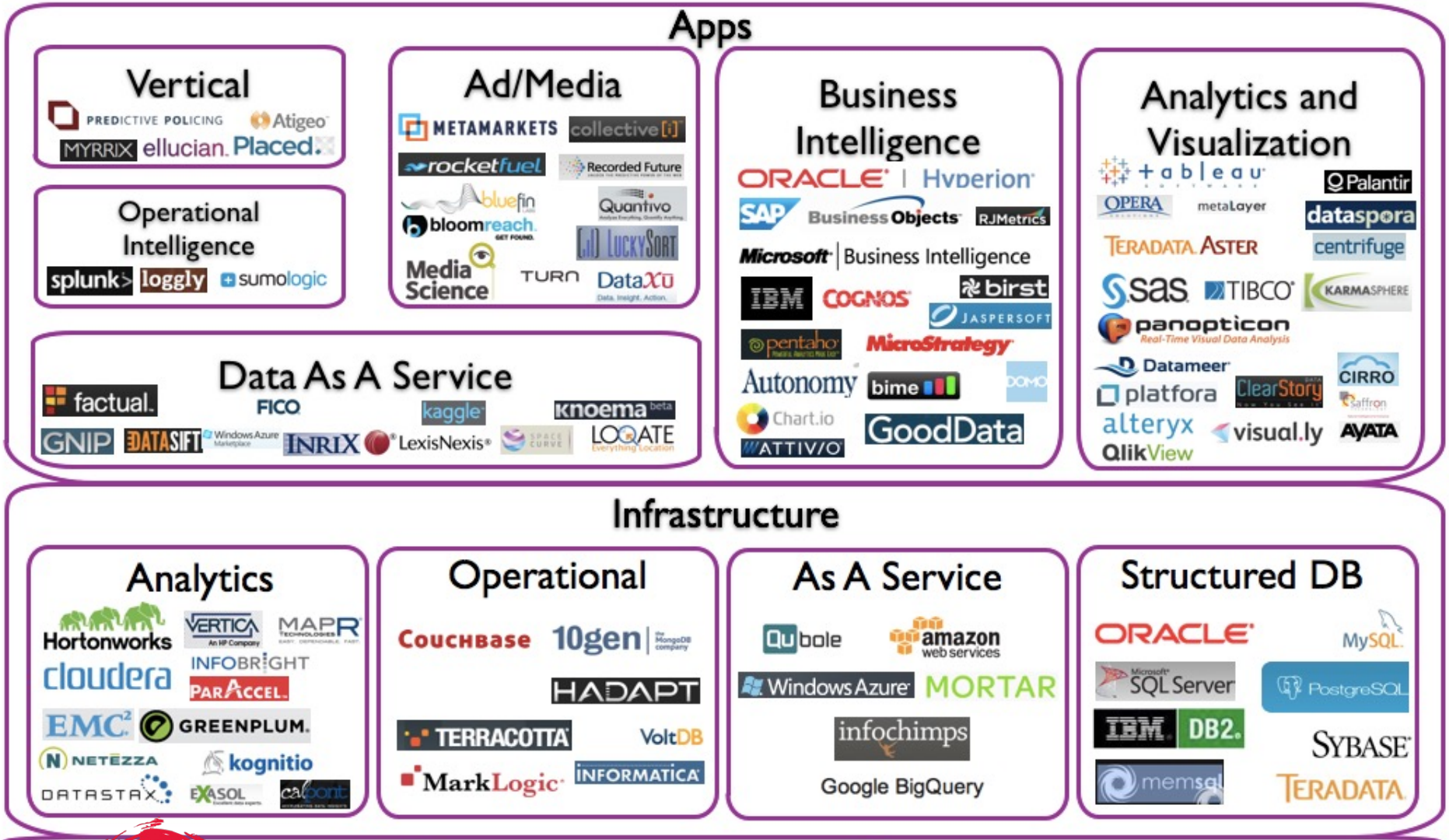
- Apply filters to reduce data quantity
  - May be similar to SQL or based on regular expressions
- Run complex processing pipelines
  - Filtering + algorithms, sometimes loops
- Too large for a single server
  → **distributed processing**

Lecture Outline

Learning Outcome 6: Discuss the pros and cons of different classes of data systems for modern analytics and data science applications.

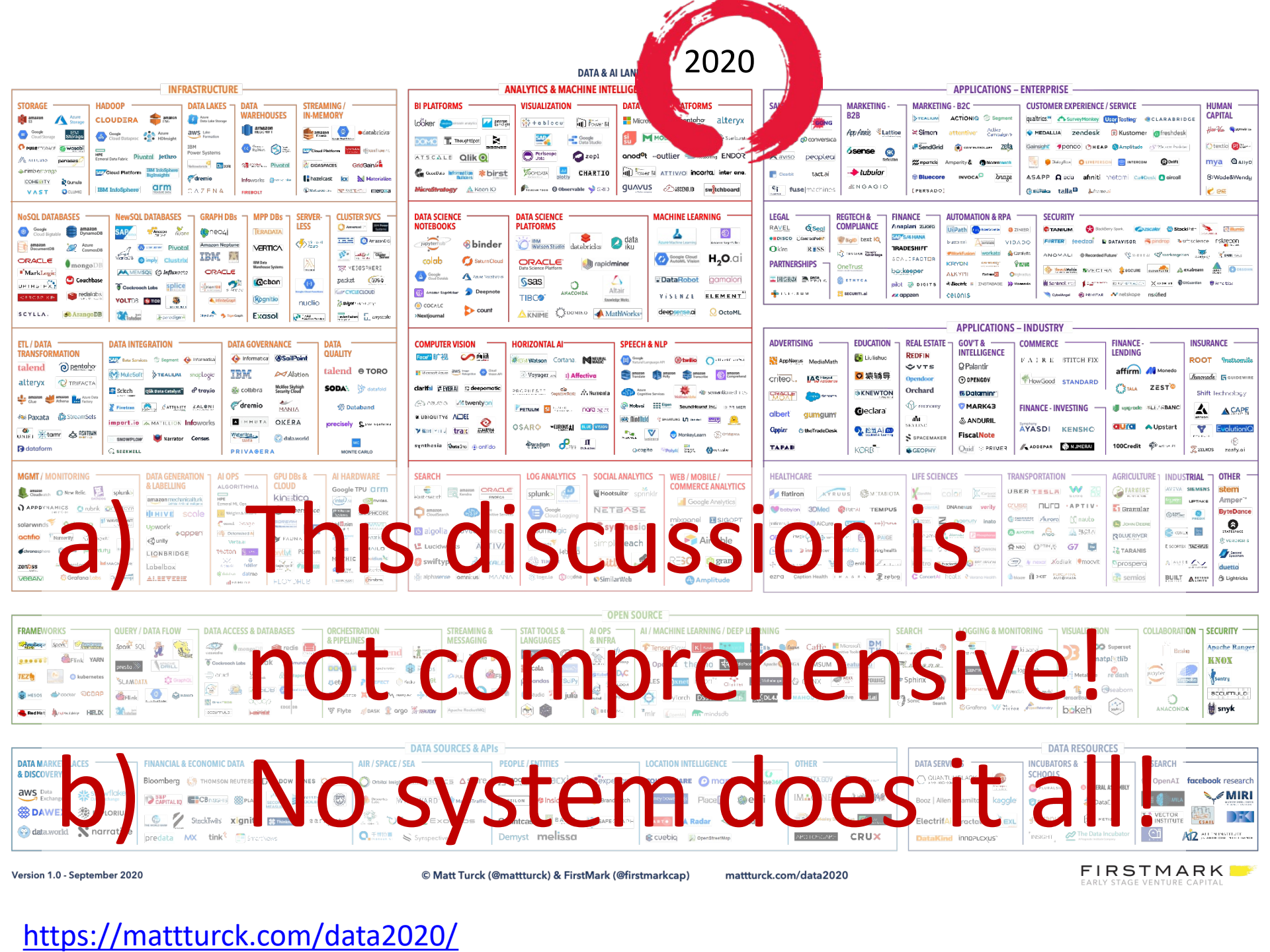
- What are modern analytics and data science applications?
- What are their access patterns and processing requirements?
- How well/badly do the classes of data systems covered so far support these access patterns?
- MapReduce/Spark: Their properties and their support for these access patterns

The Big Data Landscape



http://mattturck.com/2012/06/29/a-chart-of-the-big-data-ecosystem/


Big Data Landscape 2016
Infrastructure
Analytics
Applications
Hadoop On-Premise
Hadoop in the Cloud
Spark
Cluster Services
Analyst Platforms
Analytics Platforms
Visualization
Sales & Marketing
Customer Service
Human Capital
Legal
NoSQL Databases
NewSQL Databases
BI Platforms
Statistical Computing
Log Analytics
Social Analytics
Ad Optimization
Security
Vertical AI Applications
Graph Databases
MPP Databases
Cloud EDW
Data Transformation
Data Integration
Real-Time
Machine Learning
Speech & NLP
Horizontal AI
Publisher Tools
Govt/ Regulation
Finance
Management / Monitoring
Security
Storage
App Dev
Crowd-sourcing
Search
Data Services
For Business Analysts
SMB / Commerce
Education/ Learning
Life Sciences
Industries
Cross-Infrastructure/Analytics
Open Source
Framework
Query / Data Flow
Data Access
Coordination
Real-Time
Stat Tools
Machine Learning
Search
Security
Visualization
Data Sources & APIs
Health
IOT
Financial & Economic Data
Air / Space / Sea
Location/People/Entities
Other
Incubators & Schools
© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark Capital (@firstmarkcap)
FIRSTMARK
27


2020
DATA & AI LAN
INFRASTRUCTURE
STORAGE
HADOOP
DATA LAKES
DATA WAREHOUSES
STREAMING / IN-MEMORY
ANALYTICS & MACHINE INTELLIG
BI PLATFORMS
VISUALIZATION
APPLICATIONS – ENTERPRISE
MARKETING - B2B
MARKETING - B2C
CUSTOMER EXPERIENCE / SERVICE
HUMAN CAPITAL
NoSQL DATABASES
NewSQL DATABASES
GRAPH DBs
MPP DBs
SERVER-LESS
CLUSTER SVCS
DATA SCIENCE NOTEBOOKS
DATA SCIENCE PLATFORMS
MACHINE LEARNING
LEGAL
REGTECH & COMPLIANCE
FINANCE
AUTOMATION & RPA
SECURITY
PARTNERSHIPS
APPLICATIONS – INDUSTRY
ETL / DATA TRANSFORMATION
DATA INTEGRATION
DATA GOVERNANCE
DATA QUALITY
COMPUTER VISION
HORIZONTAL AI
SPEECH & NLP
ADVERTISING
EDUCATION
REAL ESTATE
GOV'T & INTELLIGENCE
COMMERCE
FINANCE - LENDING
FINANCE - INVESTING
INSURANCE
MGMT / MONITORING
DATA GENERATION & LABELLING
AI OPS
GPU DBs & CLOUD
AI HARDWARE
SEARCH
LOG ANALYTICS
SOCIAL ANALYTICS
WEB / MOBILE / COMMERCE ANALYTICS
HEALTHCARE
LIFE SCIENCES
TRANSPORTATION
AGRICULTURE
INDUSTRIAL
OTHER
OPEN SOURCE
FRAMEWORKS
QUERY / DATA FLOW
DATA ACCESS & DATABASES
ORCHESTRATION & PIPELINES
STREAMING & MESSAGING
STAT TOOLS & LANGUAGES
AI OPS & INFRA
AI / MACHINE LEARNING / DEEP LEARNING
SEARCH
LOGGING & MONITORING
VISUALIZATION
COLLABORATION
SECURITY
DATA SOURCES & APIs
DATA MARKETPLACES & DISCOVERY
FINANCIAL & ECONOMIC DATA
AIR / SPACE / SEA
PEOPLE / ENTITIES
LOCATION INTELLIGENCE
OTHER
DATA RESOURCES
DATA SERVICES
INCUBATORS & SCHOOLS
RESEARCH
a) This discussion is
not comprehensive!
b) No system does it all!
Version 1.0 - September 2020
© Matt Turck (@mattturck) & FirstMark (@firstmarkcap)
mattturck.com/data2020
FIRSTMARK
EARLY STAGE VENTURE CAPITAL
https://mattturck.com/data2020/

RDBMSs (OLTP and OLAP)

Strengths

- Transactions
- Complex query processing
  - Especially join processing
  - … but also filtering data

Weaknesses

- Neither strength is relevant to big data
  - … except for filtering
- Little support for unstructured data
- Little support for distributed processing

NoSQL: Key-Value Stores

Strengths

- Distributed storage
- Short latency for single items
- Handles any data equally well/badly

Weaknesses

- No means to scan the entire collection
- No support for filtering data
- No support for distributed processing pipelines

NoSQL: Document Stores

Strengths

- Distributed storage
- Short latency for single documents
- Deals with semi-structured data
- Support for filtering documents

Weaknesses

- Not designed for scanning the entire collection
- No support for distributed processing pipelines

NoSQL: Graph Stores

Strengths

- Good for certain graph-related applications
- May be strongly relational or more NoSQL-like and inherit strengths of each...

Weaknesses

- May be strongly relational or more NoSQL-like and inherit weaknesses of each...

Lecture Outline

Learning Outcome 6: Discuss the pros and cons of different classes of data systems for modern analytics and data science applications.

- What are modern analytics and data science applications?
- What are their access patterns and processing requirements?
- How well/badly do the classes of data systems covered so far support these access patterns?
- MapReduce/Spark: Their properties and their support for these access patterns

MapReduce and Hadoop

- **MapReduce**: A programming model
  - Functional style
  - Users specify Map and Reduce functions
  - Expressive
- **Hadoop**: An associated implementation
  - Automatically parallelized
  - Automatic partitioning, execution, failure handling, load balancing, communication
  - Can handle very large datasets using clusters of commodity computers

MapReduce in a Nutshell (I)

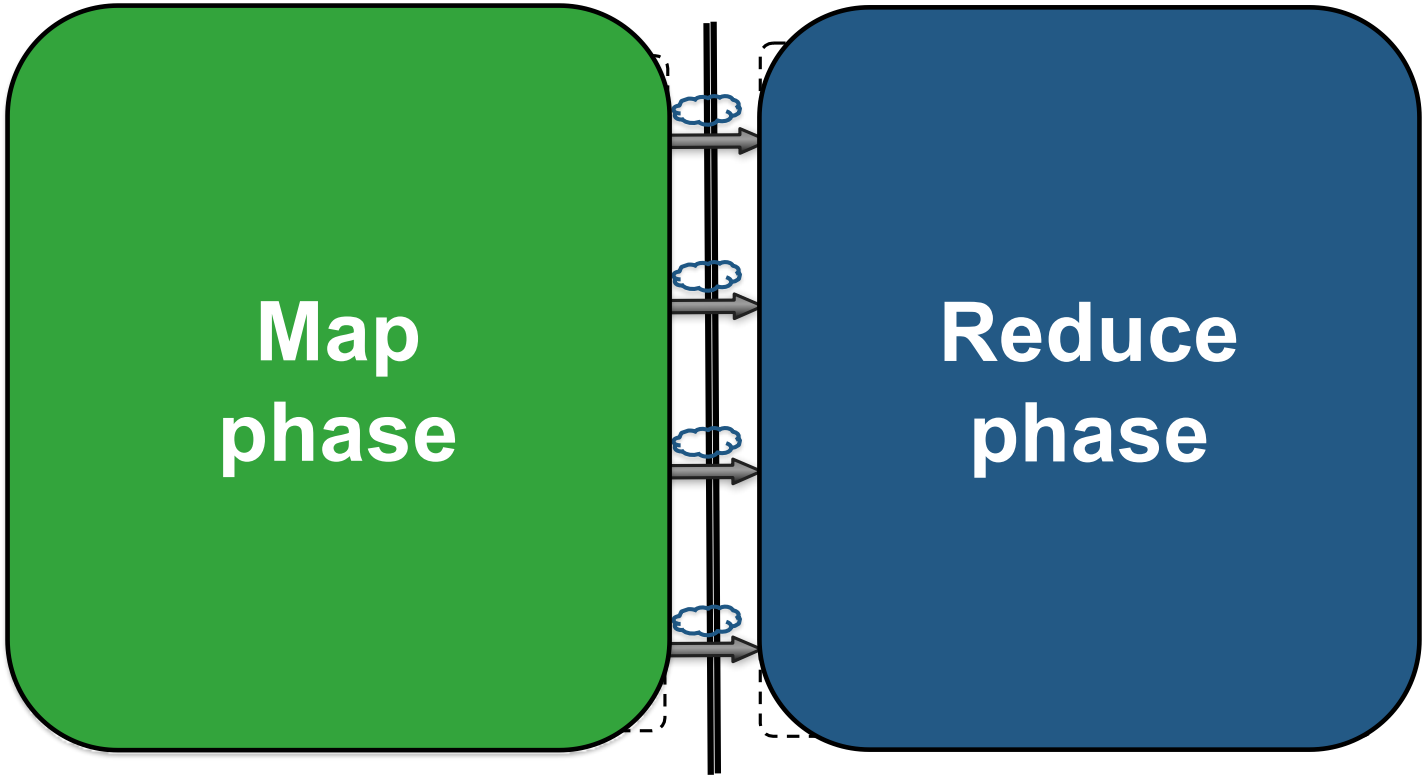
- Framework
  - Read lots of data
  - **Map**: process a data item/record
  - **Sort and shuffle**
  - **Reduce**: aggregate, summarize, filter, transform
  - Write results

- Map and Reduce are **user-specified**→ used to model given problem

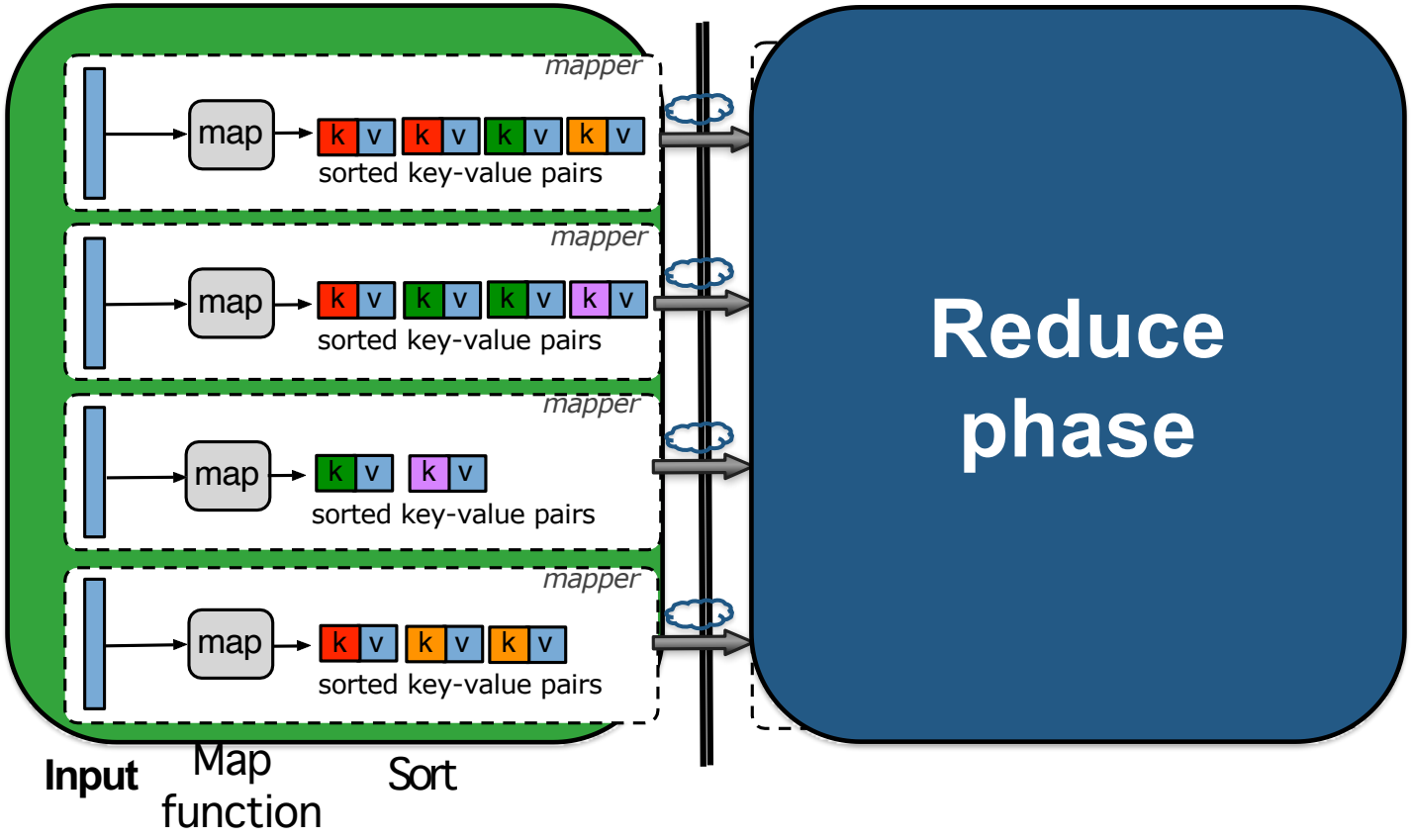
MapReduce in a Nutshell (II)

- Simple API
  - **Map** (*key*, *value*) —> {*ikey*, *ivalue*}
  - **Reduce** (*ikey*, {*ivalues*}) —> (*key'*, *value'*)
- **Map** phase: independent processes (mappers) which run in parallel
  - operate on the chunks (blocks) of input data
  - output intermediate results
- **Shuffle** phase: Intermediate results are shuffled through the network
- **Reduce** phase: independent processes (reducers) which run in parallel
  - group intermediate results of the map phase
  - operate on the groups
  - output final results

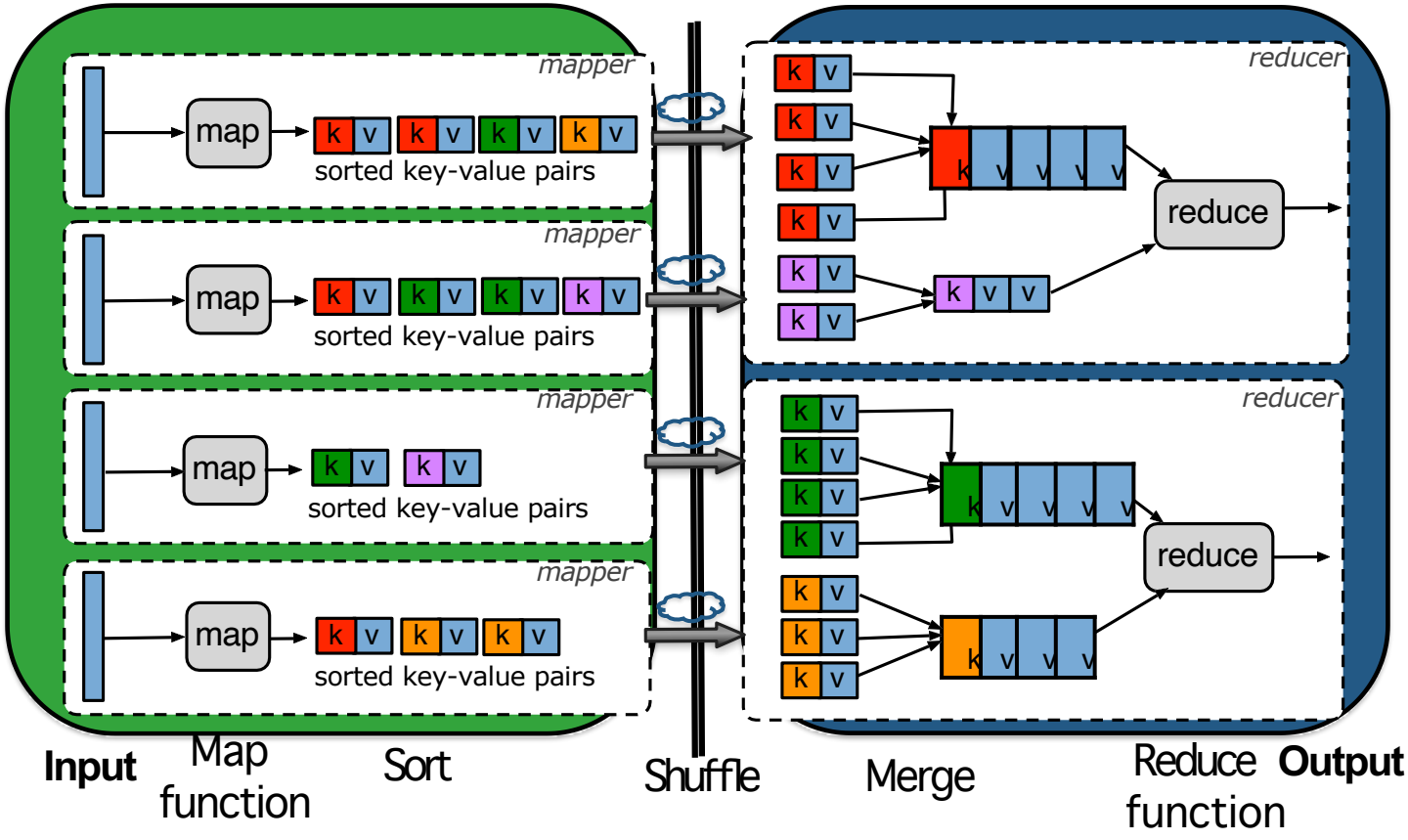
MapReduce Illustration



Map Reduce Illustration



Map Reduce Illustration

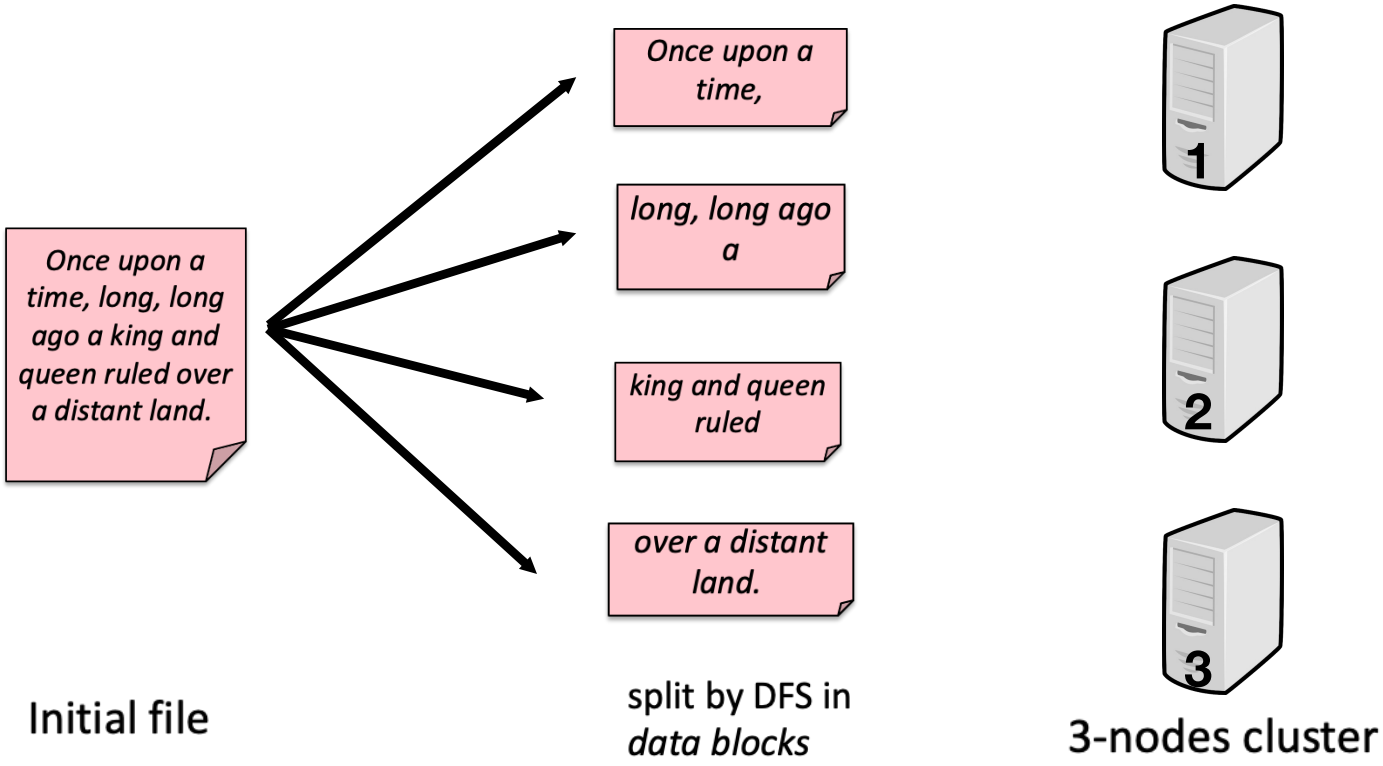


Wordcount Example

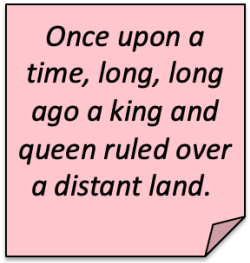
Map (*key*, *value*) —> {*ikey*, *ivalue*}
Reduce (*ikey*, {*ivalues*}) —> (*key'*, *value'*)

Map (*docID*, *text*) —> {*word*, *1*}
Reduce (*word*, {*1, 1, …*}) —> (*word*, count)

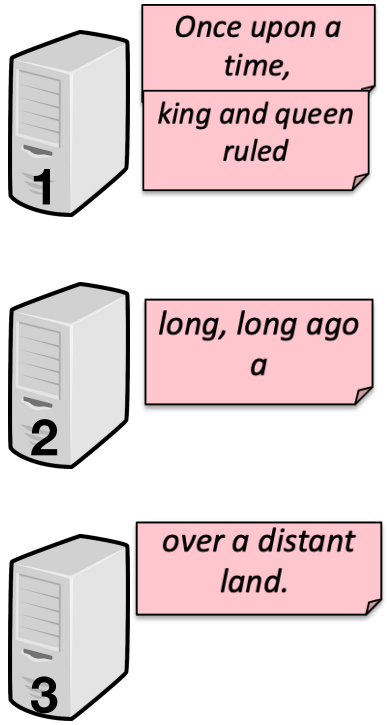
Wordcount Example



Wordcount Example

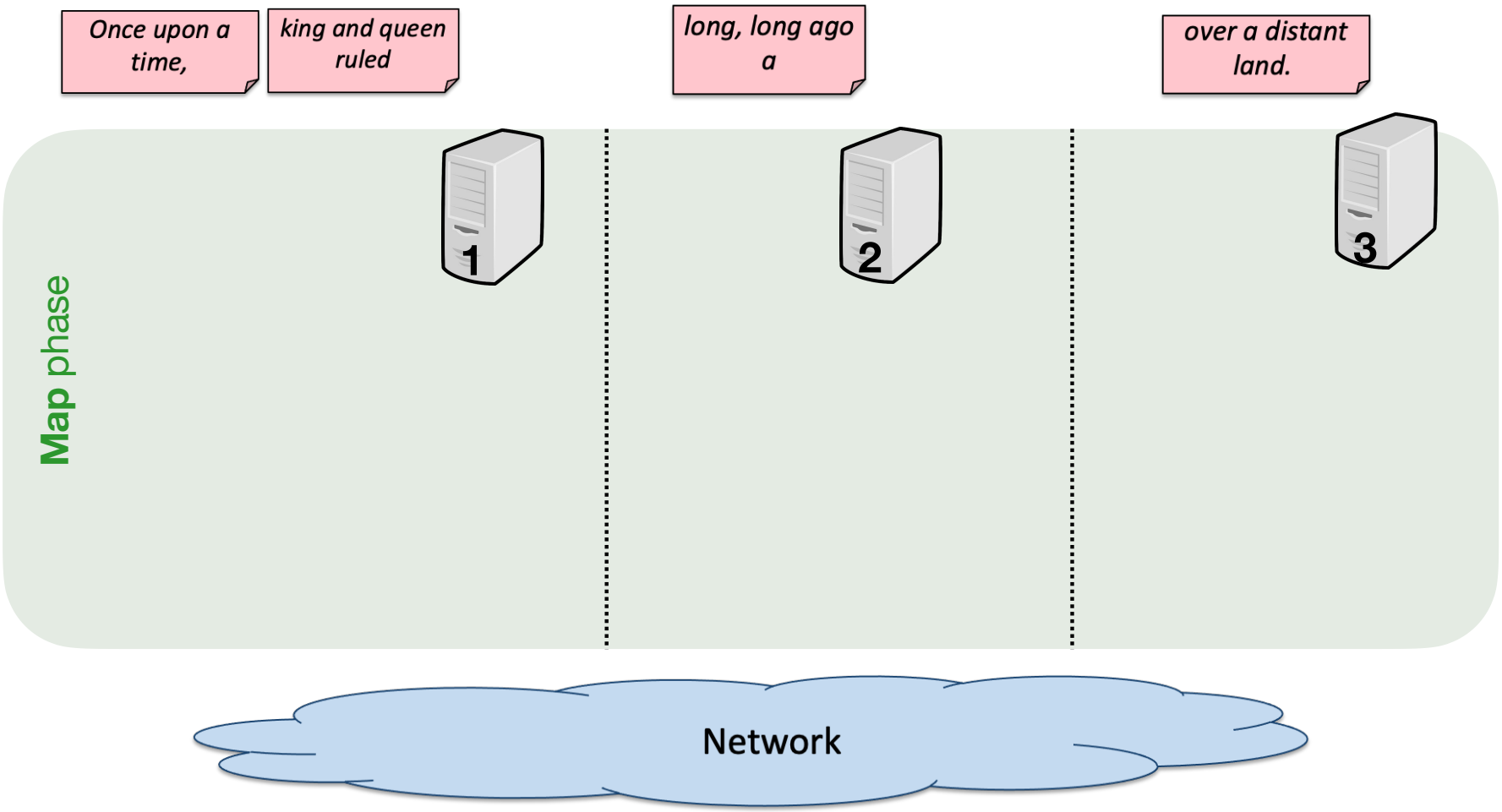


Initial file

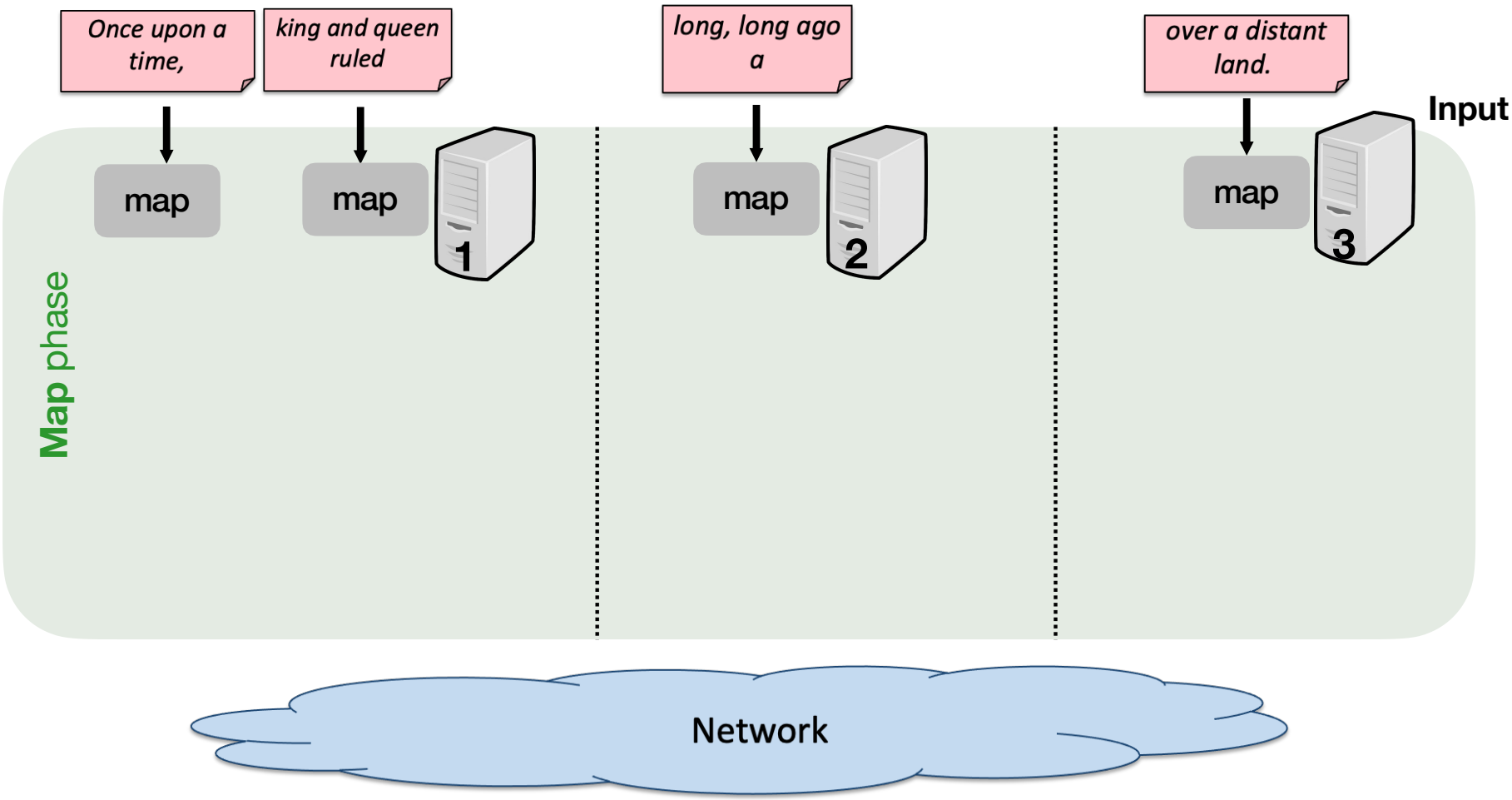


3-nodes cluster

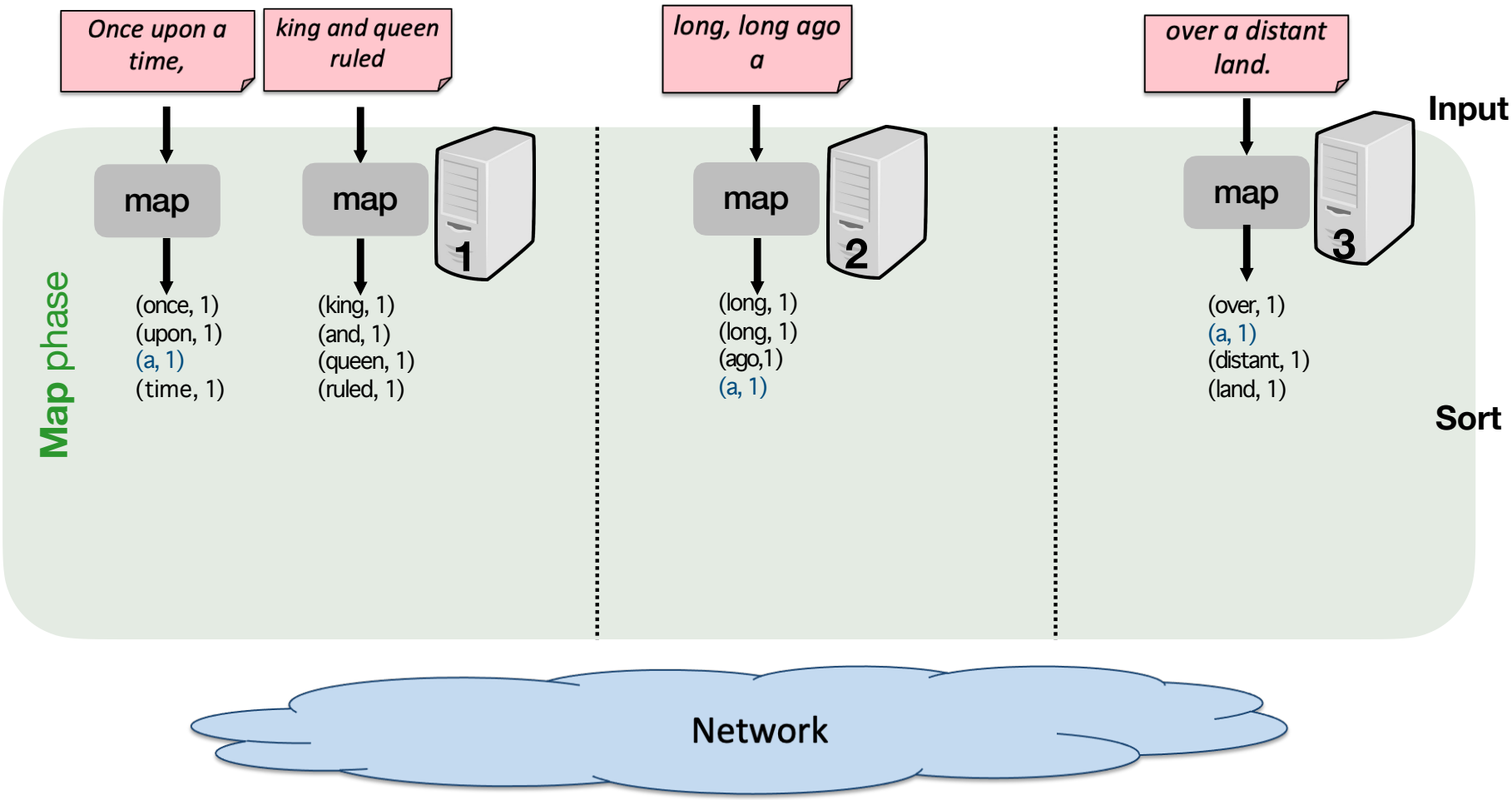
Wordcount Example (Map Phase)



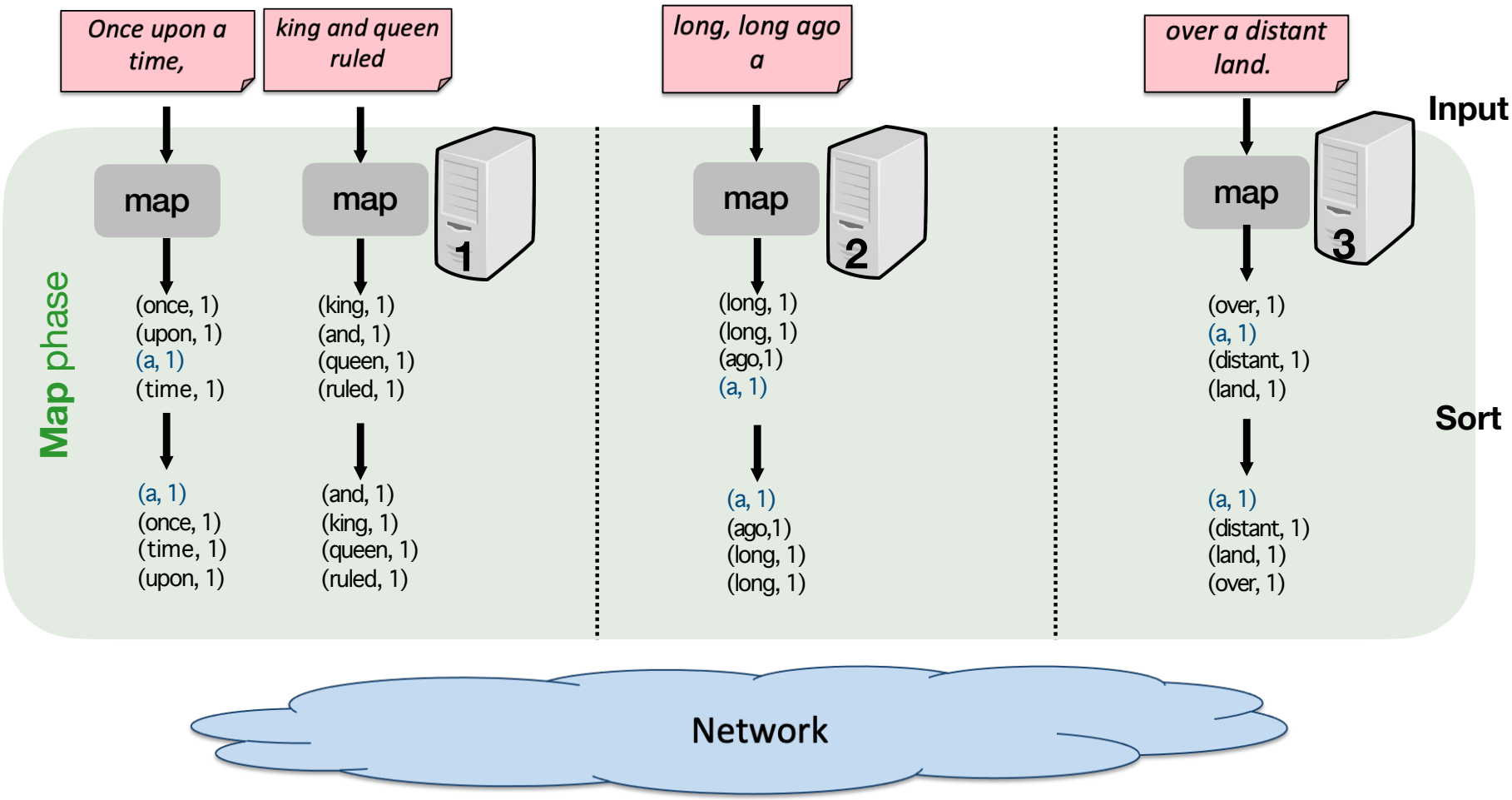
Wordcount Example (Map Phase)



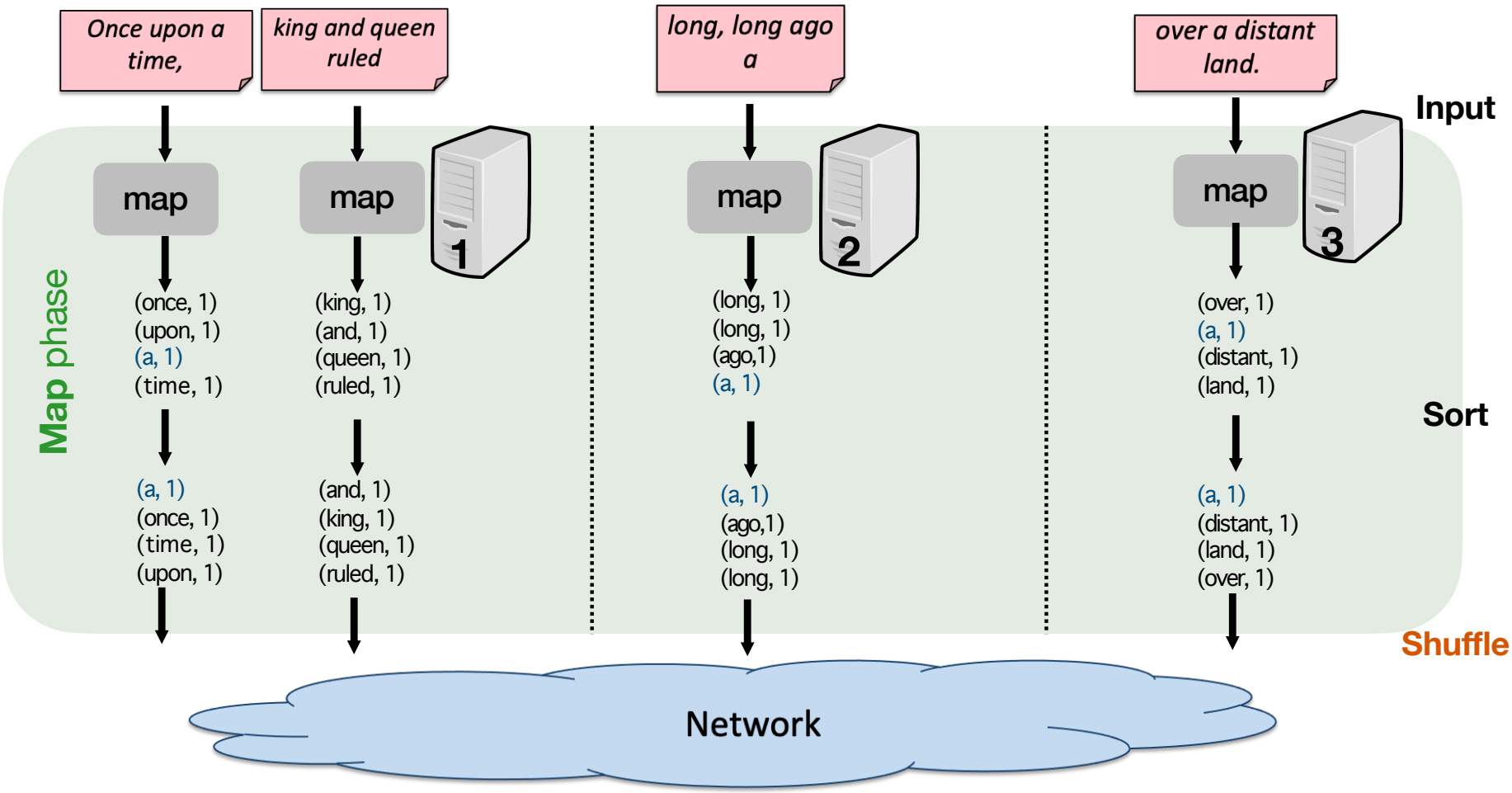
Wordcount Example (Map Phase)



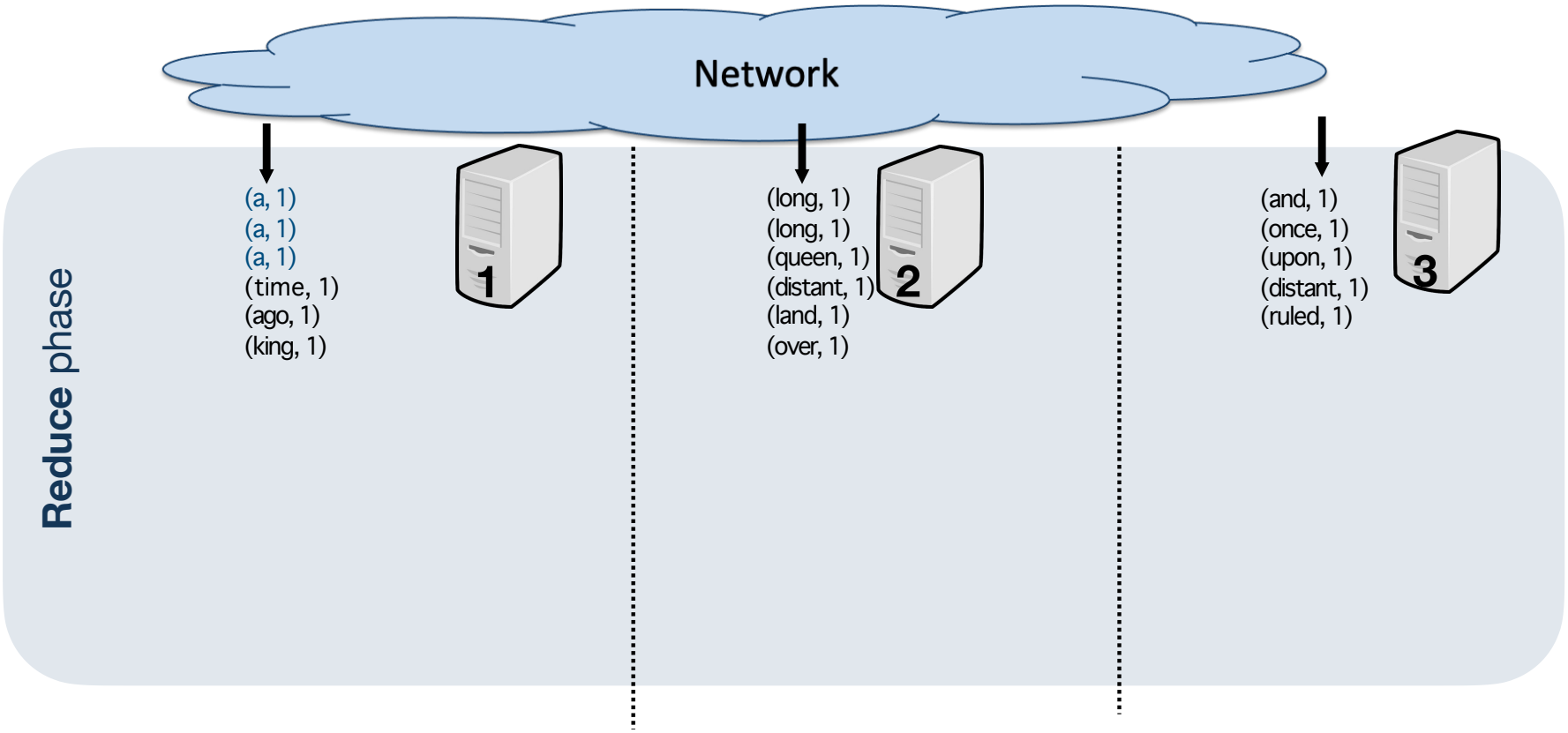
Wordcount Example (Map Phase)



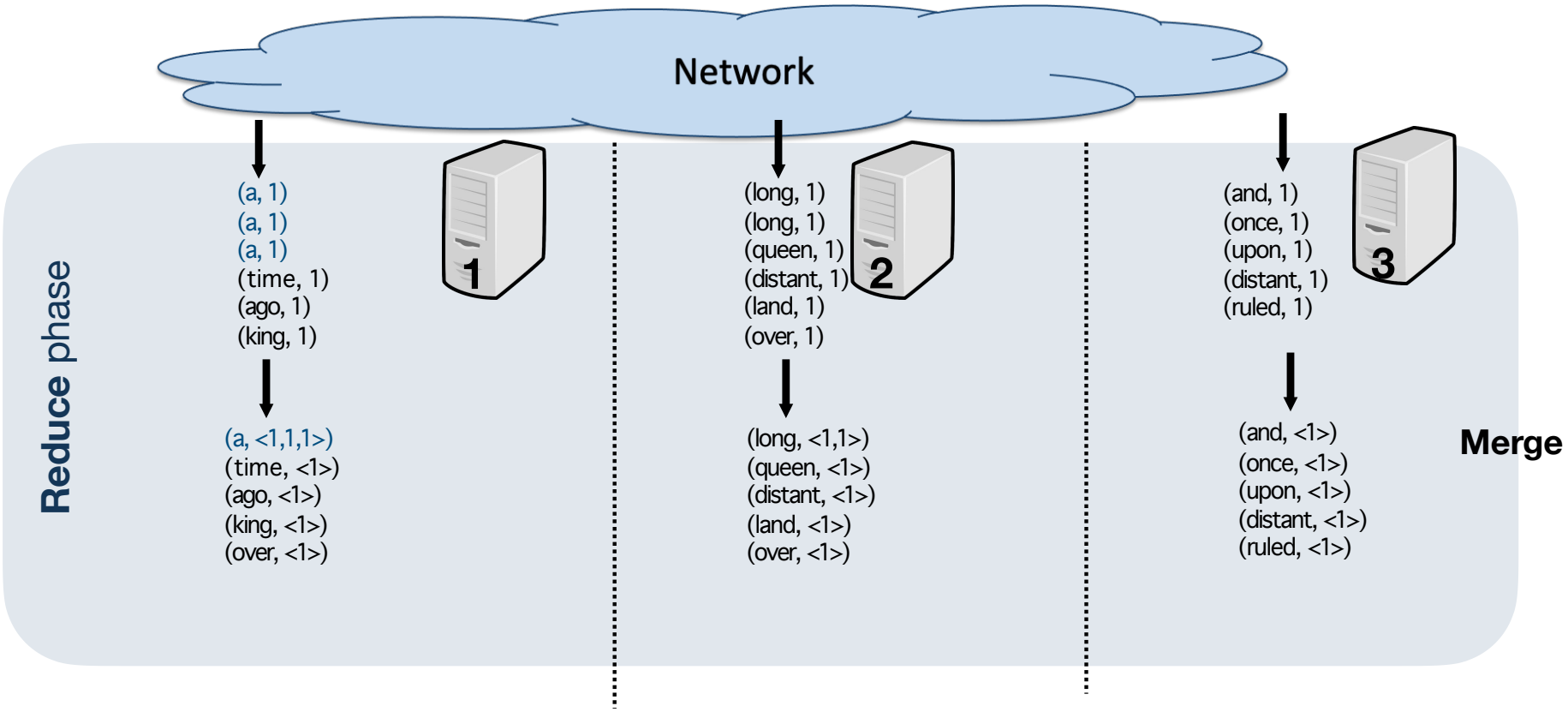
Wordcount Example (Map Phase)



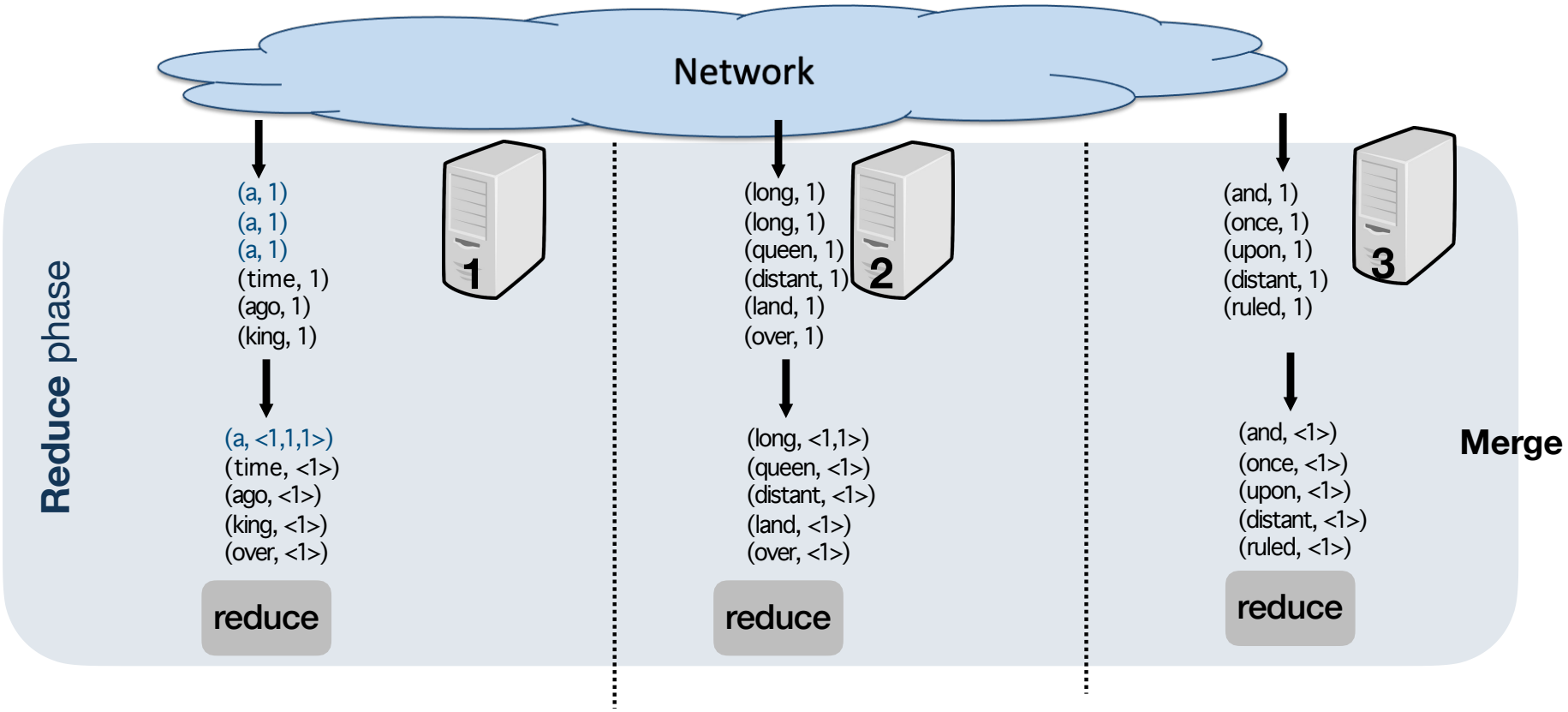
Wordcount Example (Reduce Phase)



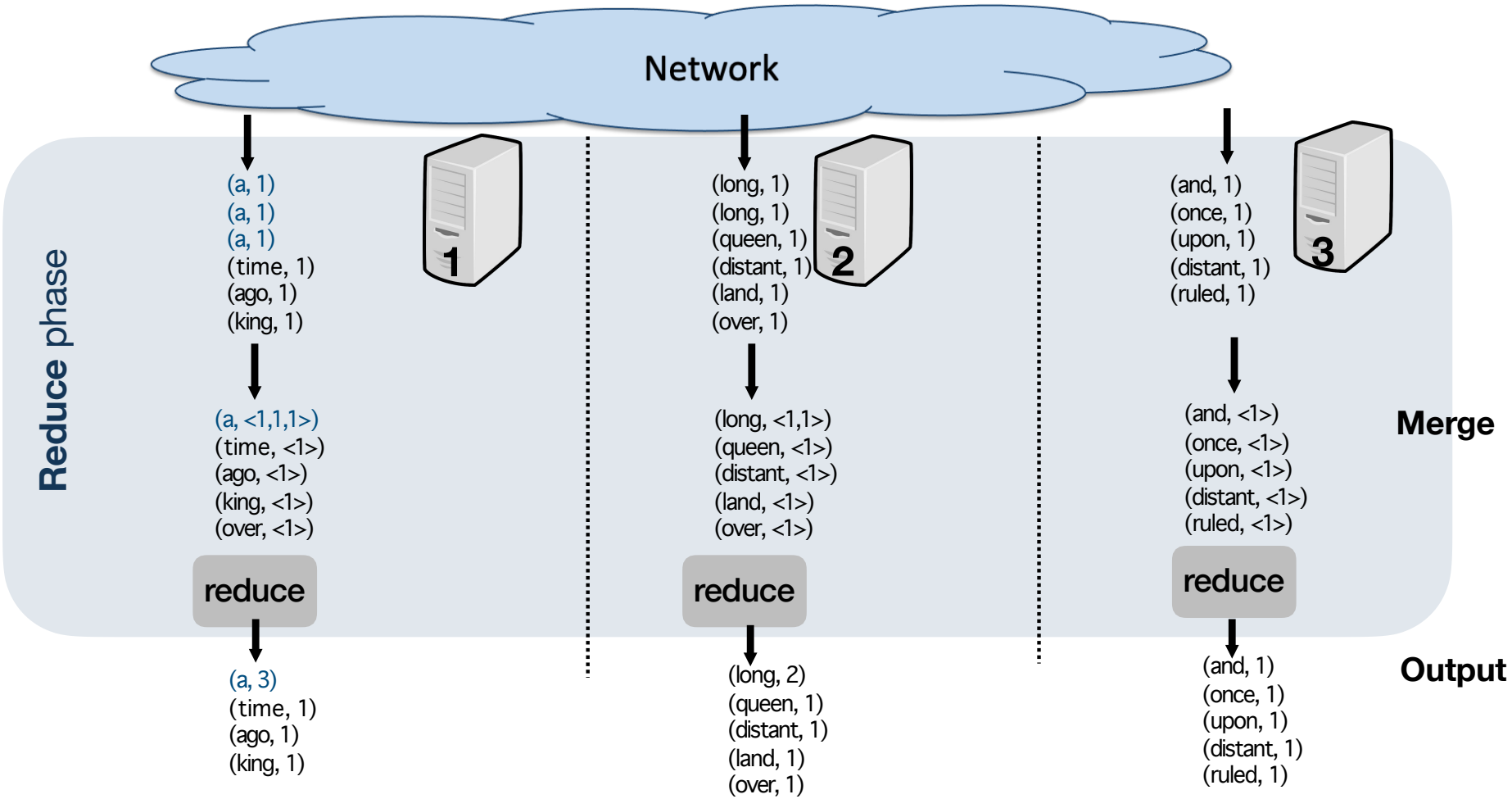
Wordcount Example (Reduce Phase)



Wordcount Example (Reduce Phase)



Wordcount Example (Reduce Phase)



Wordcount Example: Pseudocode

```
Map(Integer key, String text)
    foreach word w in text:
        emit (w, 1)

Reduce(String word, Iterator<Integer> counts)
    int result = 0
    foreach count in counts:
        result += count
    emit (word, result)
```

Hadoop Problems

- Containers prevent sharing resources
  - Large data structures → one per thread!
- Rigid MapReduce pipeline
  - Low-level abstraction
  - Map → Map → Reduce → Map → Reduce

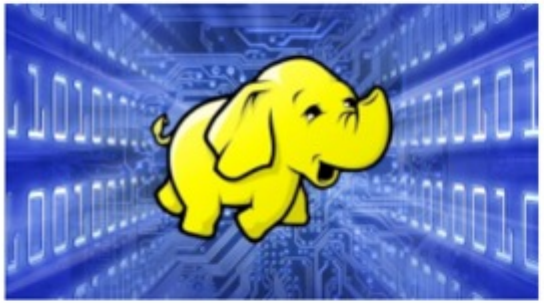
- Limited successes

Hadoop Critique

March 13, 2017

Hadoop Has Failed Us, Tech Experts Say

Alex Woodie



The Hadoop dream of unifying data and compute in a distributed manner has all but failed in a smoking heap of cost and complexity, according to technology experts and executives who spoke to *Datanami*.

"I can't find a happy Hadoop customer. It's sort of as simple as that," says Bob Muglia, CEO of Snowflake Computing, which develops and runs a cloud-based relational data warehouse offering. "It's very clear to me, technologically, that it's not the technology base the world will be built on going forward."

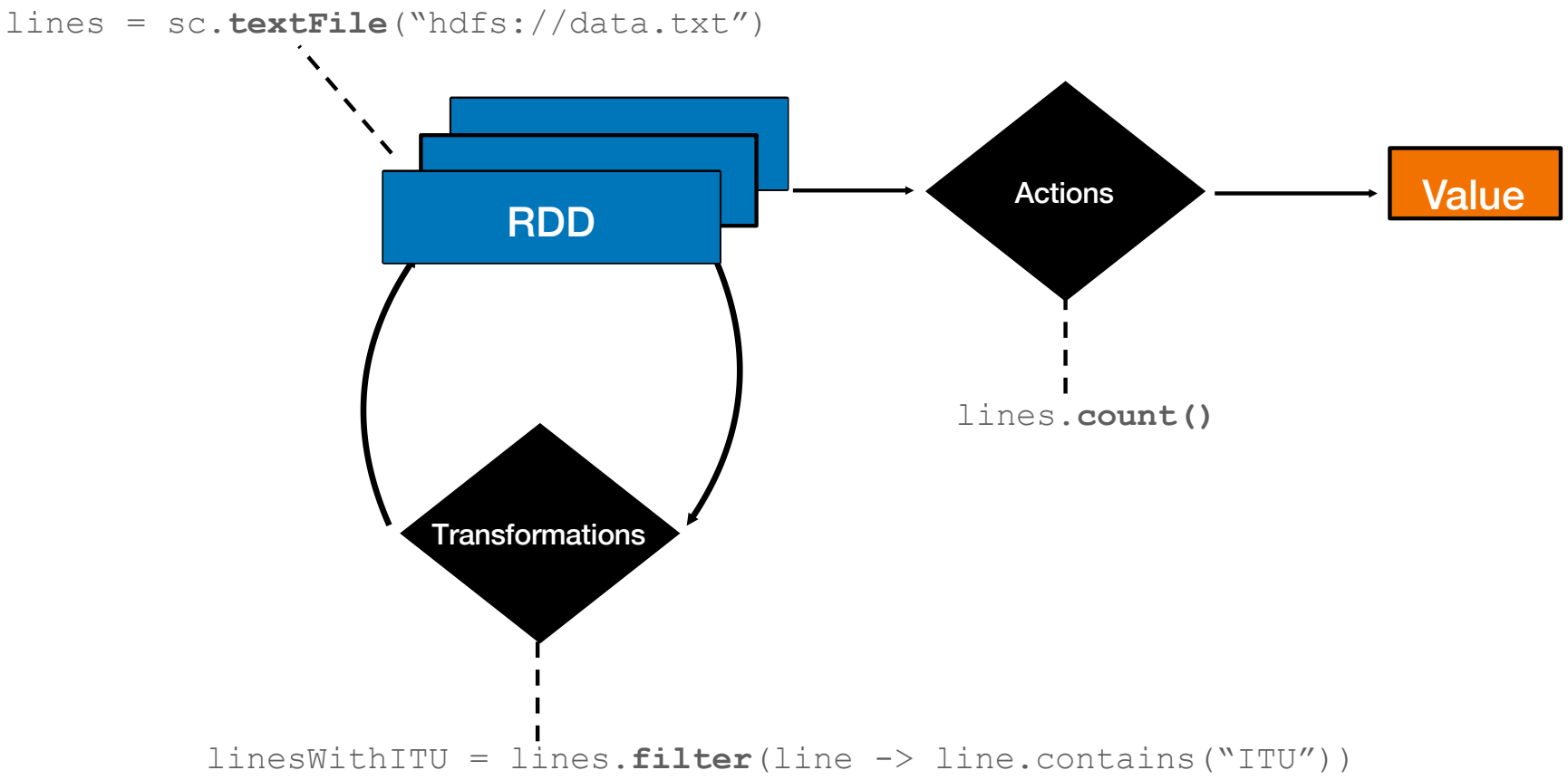
Hadoop's strengths lie in serving as a cheap storage repository and for processing ETL batch workloads, Johnson says. But it's ill-suited for running interactive, user-facing applications, he says.

Spark RDDs

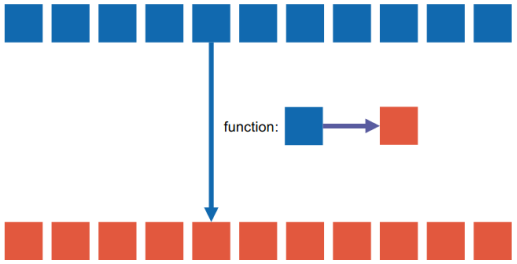
- Resilient Distributed Datasets (RDDs)
  - Transform one RDD to another via operators
  - Lazy execution – optimizations
- Supports deep pipelines
- Supports worker's memory sharing

- Write programs in terms of distributed datasets and operations on them

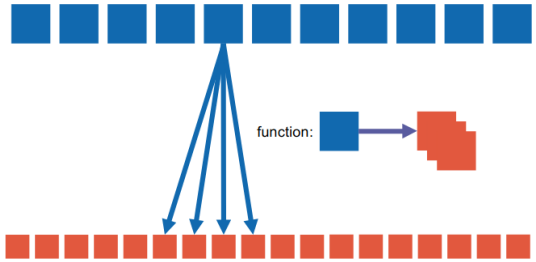
Working with RDDs (I)



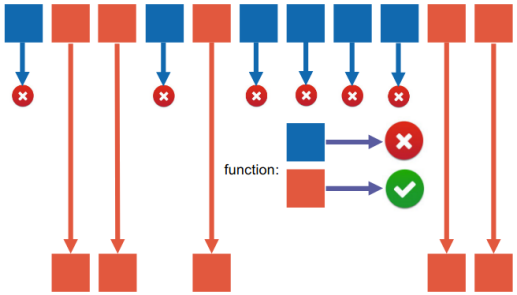
Example Transformations



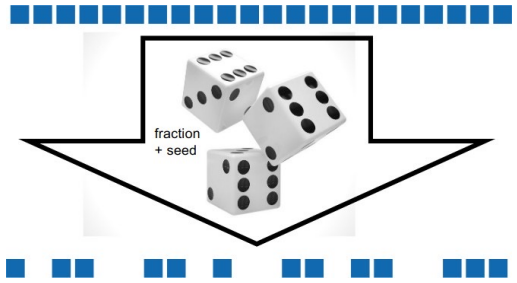
map ( )



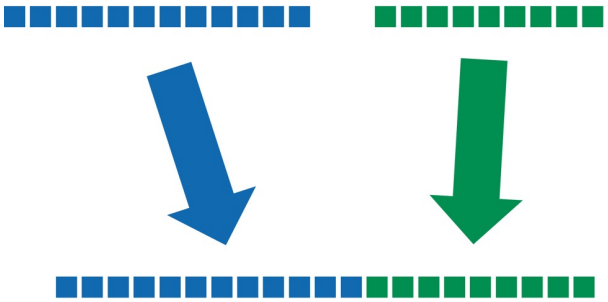
flatmap ( )



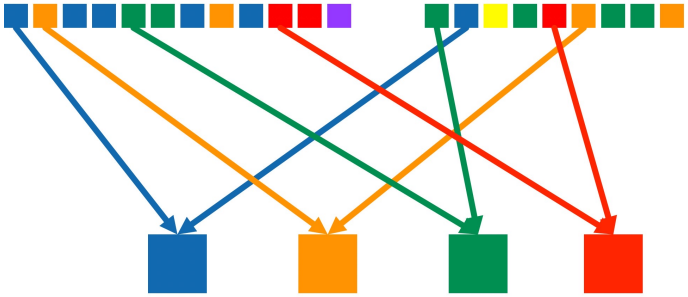
filter ( )



sample ( )

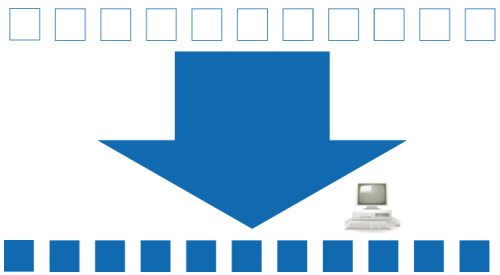


union ( )

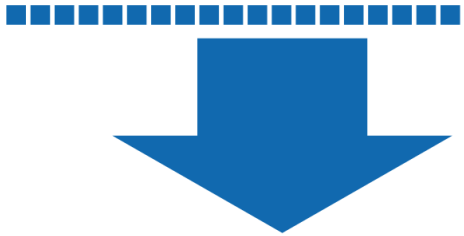


intersect ( )

Example of Actions



collect ( )

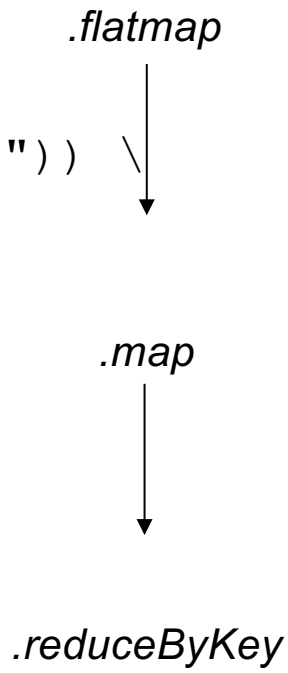


count ( )

countByValue ( )

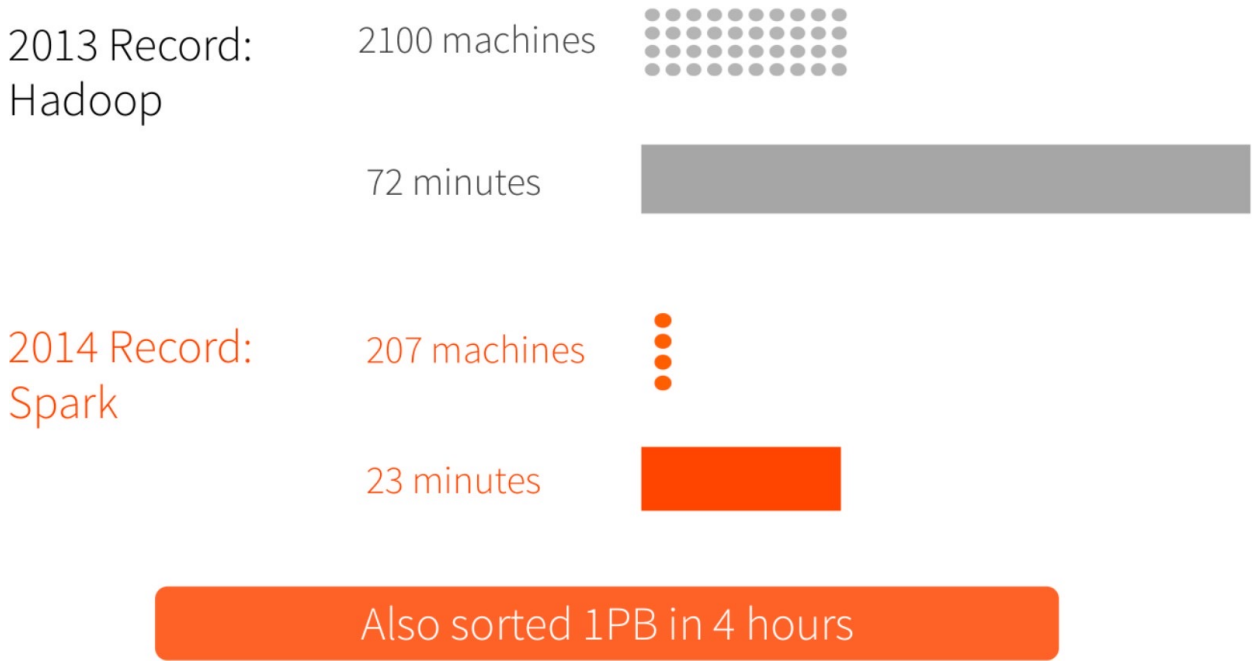
Spark Wordcount Example

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```



http://spark.apache.org/examples.html

Apache Spark: Performance



https://www.slideshare.net/rxin/stanford-cs347-guest-lecture-apache-spark

Spark SQL

- Structured data within Spark programs
  - DataSet/DataFrame as data representation
  - SQL as query language
  - Improved data parsing

```
import org.apache.spark.sql.types._

// Create an RDD
val peopleRDD = spark.sparkContext.textFile("examples/src/main/resources/people.txt")

// The schema is encoded in a string
val schemaString = "name age"

// Generate the schema based on the string of schema
val fields = schemaString.split(" ")
  .map(fieldName => StructField(fieldName, StringType, nullable = true))
val schema = StructType(fields)

// Convert records of the RDD (people) to Rows
val rowRDD = peopleRDD
  .map(_.split(","))
  .map(attributes => Row(attributes(0), attributes(1).trim))

// Apply the schema to the RDD
val peopleDF = spark.createDataFrame(rowRDD, schema)

// Creates a temporary view using the DataFrame
peopleDF.createOrReplaceTempView("people")

// SQL can be run over a temporary view created using DataFrames
val results = spark.sql("SELECT name FROM people")

// The results of SQL queries are DataFrames and support all the normal RDD operations
// The columns of a row in the result can be accessed by field index or by field name
results.map(attributes => "Name: " + attributes(0)).show()
// +-------------+
// |        value|
// +-------------+
// |Name: Michael|
// |   Name: Andy|
// | Name: Justin|
// +-------------+
```


http://spark.apache.org/docs/latest/sql-programming-guide.html

Hadoop/Spark

Strengths

- Good for distributed storage
- Good for complex distributed processing pipelines
  - Especially Spark!
- Excellent throughput!
  - Especially Spark!

Weaknesses

- Not really data management systems
- Very high latency of small requests!

→ Only suitable for distributed processing pipelines!

Take Aways

- Analytics applications typically require (repeated) processing of whole datasets of unstructured data
- This access pattern is not well served by SQL or NoSQL data systems
  - RDBMS: No need for transactions, and many analytics processes do not use SQL (but it can be useful for filtering)
  - NoSQL: Distributed storage is important, but individual records are not accessed
  - Both: Distributed (large-scale) processing is required
- MapReduce and (especially) Spark perform better in these applications