Big Data Overview

Agenda

- What is “Big Data” all about?
- Fundamental Database Concepts
- The “Problem”: Relational Databases and Big Data
- The “Solution”: Scaling Techniques, NoSQL Products
- Hadoop

Current Trends in Computing

- Massive Amounts of data being generated & collected
  - Terabytes (TB 10^{12}) PetaBytes (PB 10^{15}), ExaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billions of people all over the world
- Lots of Different Kinds of data
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance
  - Real-time data

Current Trends in Computing

- Ma[illegible]ed & collected
  - [illegible]kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billions of people all over the world
- Lots of Different Kinds of data
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance

Volume

Current Trends in Computing

- Ma[illegible] ed & collected
  - [illegible] kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billion[illegible]
- Lots of D[illegible]
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance

Volume

Variety

Current Trends in Computing

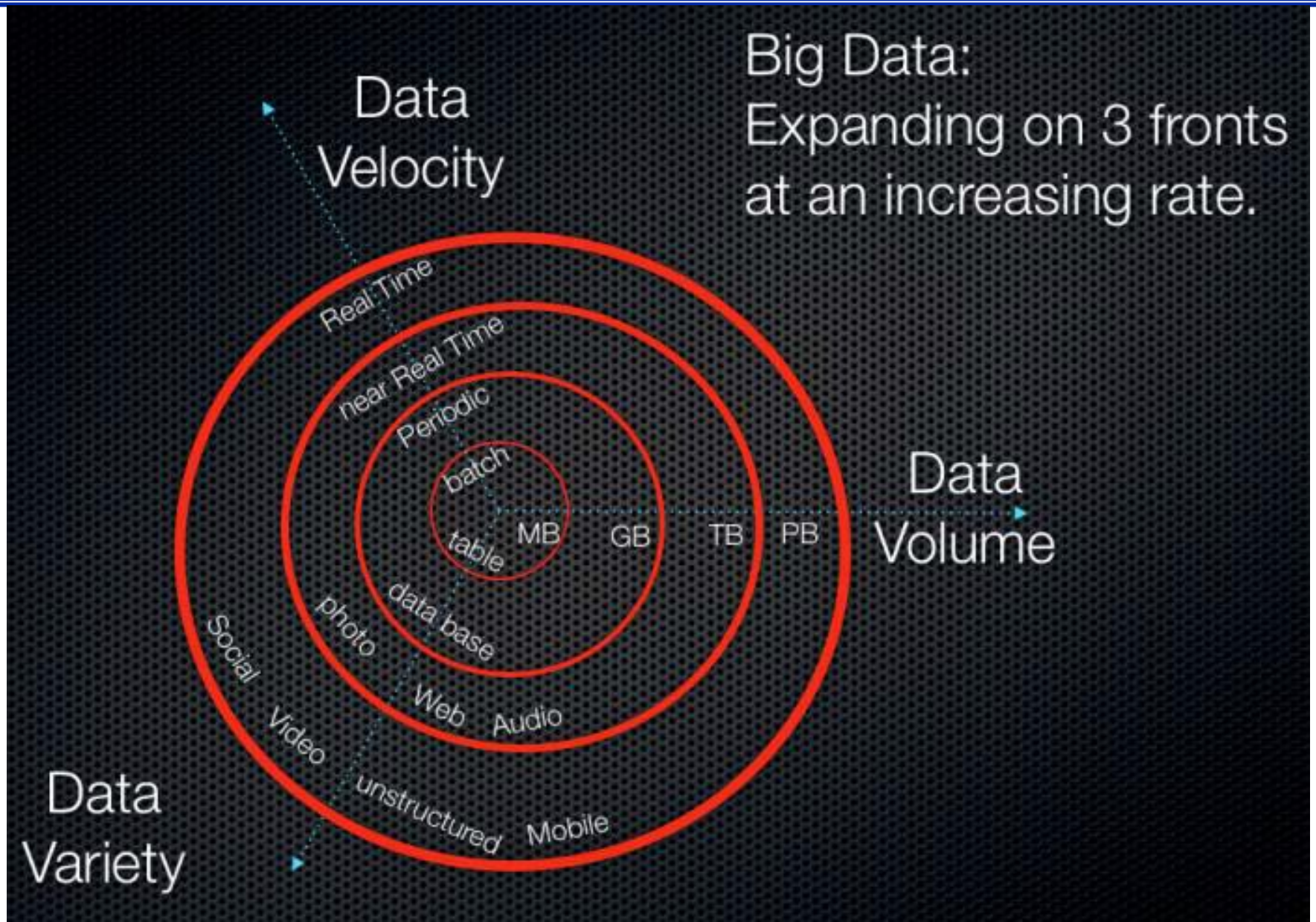
- Ma… ed & collected
  - …kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billion…
- Lots of D…
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of …
  - Competitior…
  - No tolerance for downtime, slow performance

Volume

Variety

Velocity

Big Data Overview



A Few Amazing Facts (1)

- Annual global IP traffic passed the zettabyte threshold by the end of 2016, and will reach 2.3 ZB per year by 2020.
- Smartphone traffic will exceed PC traffic by 2020.
- Traffic from wireless and mobile devices will account for two-thirds of total IP traffic by 2020.
- The number of devices connected to IP networks will be more than three times the global population by 2020.

Cisco The Zettabyte Era—Trends and Analysis http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

What is driving this data explosion?

- Smart Phones
- Apps
- The Internet of Things
- Digital Commerce
- Online Entertainment
- Cloud Computing
- Social Media

"Social media facilitate the most intimate of discussions among users, who seem to forget that they're communicating on a public forum."

Data Crush, Christopher Surdak

Smart Phones

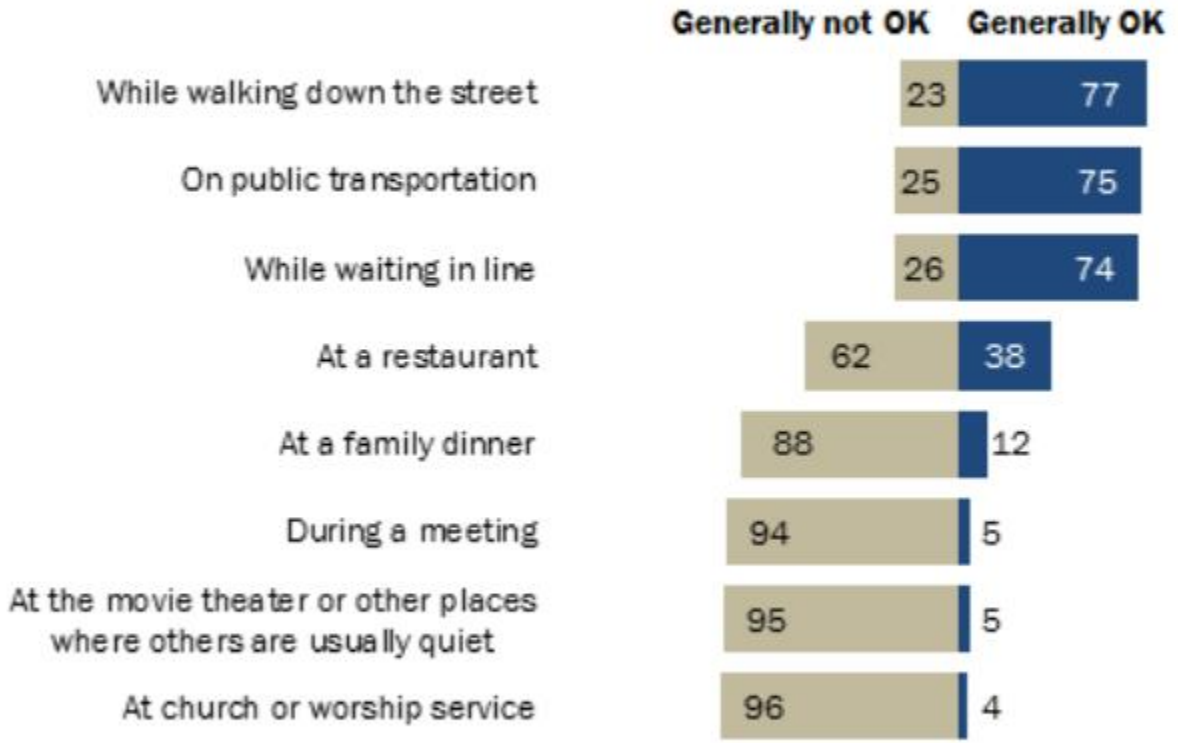
- Smartphones -- Only here for a decade: the first iPhone came out in January 2007
- In 2010, 4.5 B people owned a mobile phone
- By 2017, there were 6.8 B mobile phone users (90% of humanity)
  - Of which 2.5 B are “smartphones”
- The mobile phone has become the individual’s dominant point of interaction with society

Data Overview

"On the Internet, nobody knows you're a dog."

People Have Varying Views About When It Is OK Or Not OK To Use Their Cellphones

% of adults who believe it is OK or not to use a cellphone in these situations



Source: Pew Research Center American Trends Panel survey, May 30-June 30, 2014.
N=3,217 adults.

PEW RESEARCH CENTER

Apps

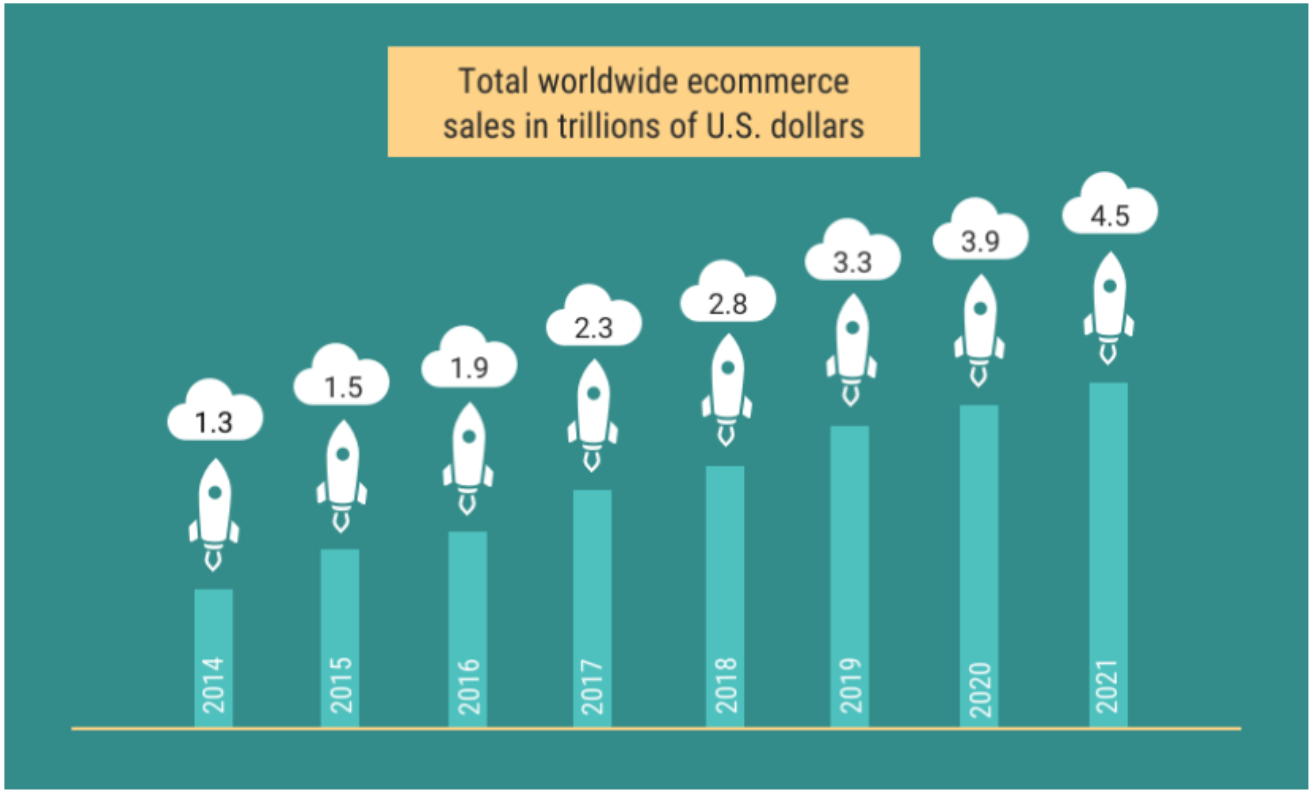
- 2012 – Apple reached 25 B annual downloads of apps
- 2013 – Apple reached 40 B annual downloads of apps
- Android apps – 50 B downloads in 2015
- Android apps – 90 B downloads in 2016
- By December 2017 – combined downloads = 197 B
- Google Play provides over a million downloadable apps & games

IOT – The internet of things

- Internet connected devices that are collecting and transmitting data
  - TVs with wifi connectivity
  - Cars with OnStar
  - Appliances (refrigerators, thermostat, etc.)
  - Medical devices
  - “Wearables”
  - FitBit
  - Apple Watch
  - Robots
  - Assistants (Alexa, Google Home, Siri)

eCommerce

1. Global Retail Ecommerce Sales Will Reach $4.5 Trillion by 2021



eCommerce

Data from Statista forecasts a 246.15% increase in worldwide ecommerce sales, from $1.3 trillion in 2014 to $4.5 trillion in 2021.

Nearly threefold lift in online revenue.

Online Entertainment

- Online video has a 90 percent penetration rate among internet users on any device.
- In the United States, current data indicates more than 180 million digital video viewers with a penetration rate of approximately 75 percent of the online population.
- The majority of mobile video revenues are generated via subscriptions, followed by advertising.
- Current US mobile video subscription revenues have already surpassed 1 billion US dollars annually.

Cloud Computing

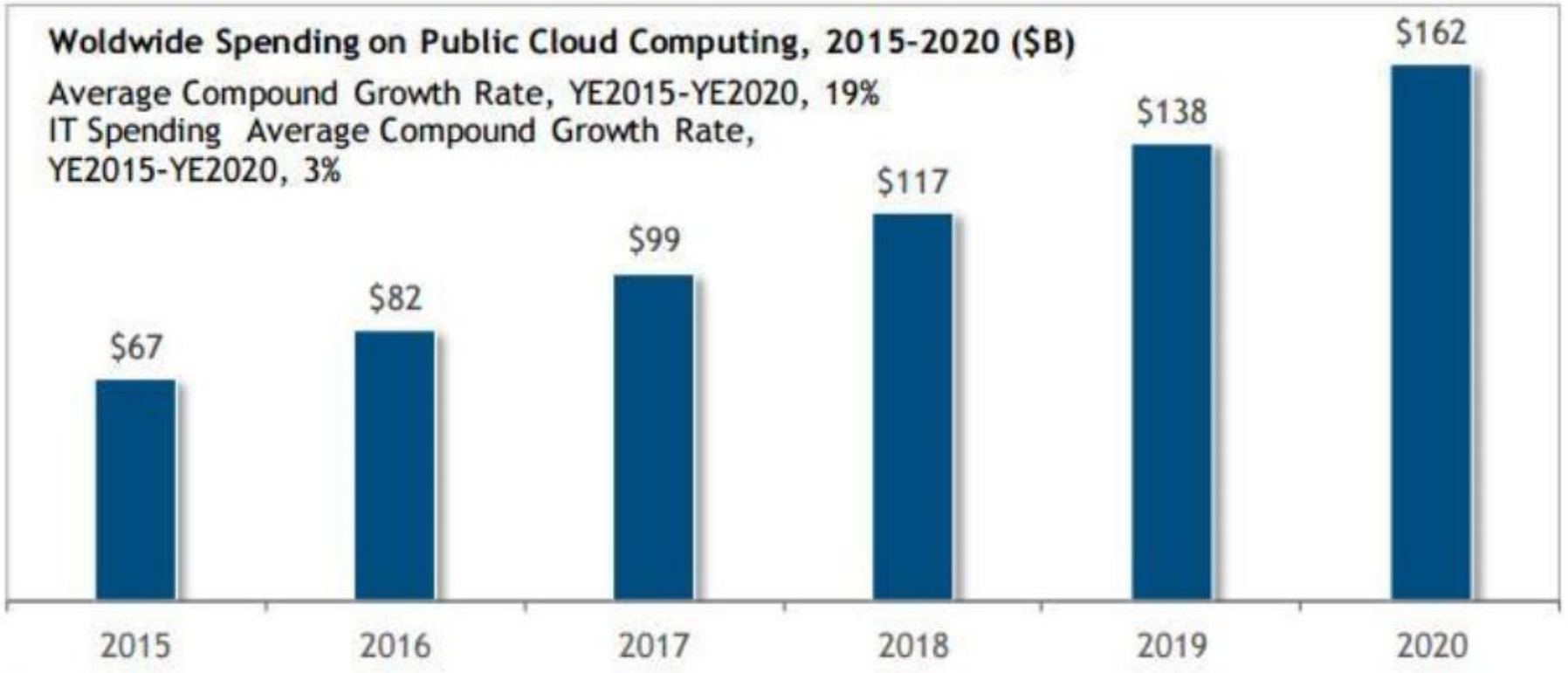
- Cloud computing is projected to increase from $67B in 2015 to $162B in 2020 attaining a compound annual growth rate (CAGR) of 19%.
- Gartner reports the worldwide public cloud services market grew 18% in 2017 to $246.8B, up from $209.2B in 2016.
- 74% of Tech Chief Financial Officers (CFOs) say cloud computing will have the most measurable impact on their business

Cloud Computing

Cloud computing spending is growing at 4.5 times the rate of IT spending since 2009 and is expected to grow at better than 6 times the rate of IT spending from 2015 through 2020.

According to IDC, worldwide spending on public cloud computing will increase from $67B in 2015 to $162B in 2020 attaining a 19% CAGR.

The Rapid Growth of Cloud Computing, 2015-2020



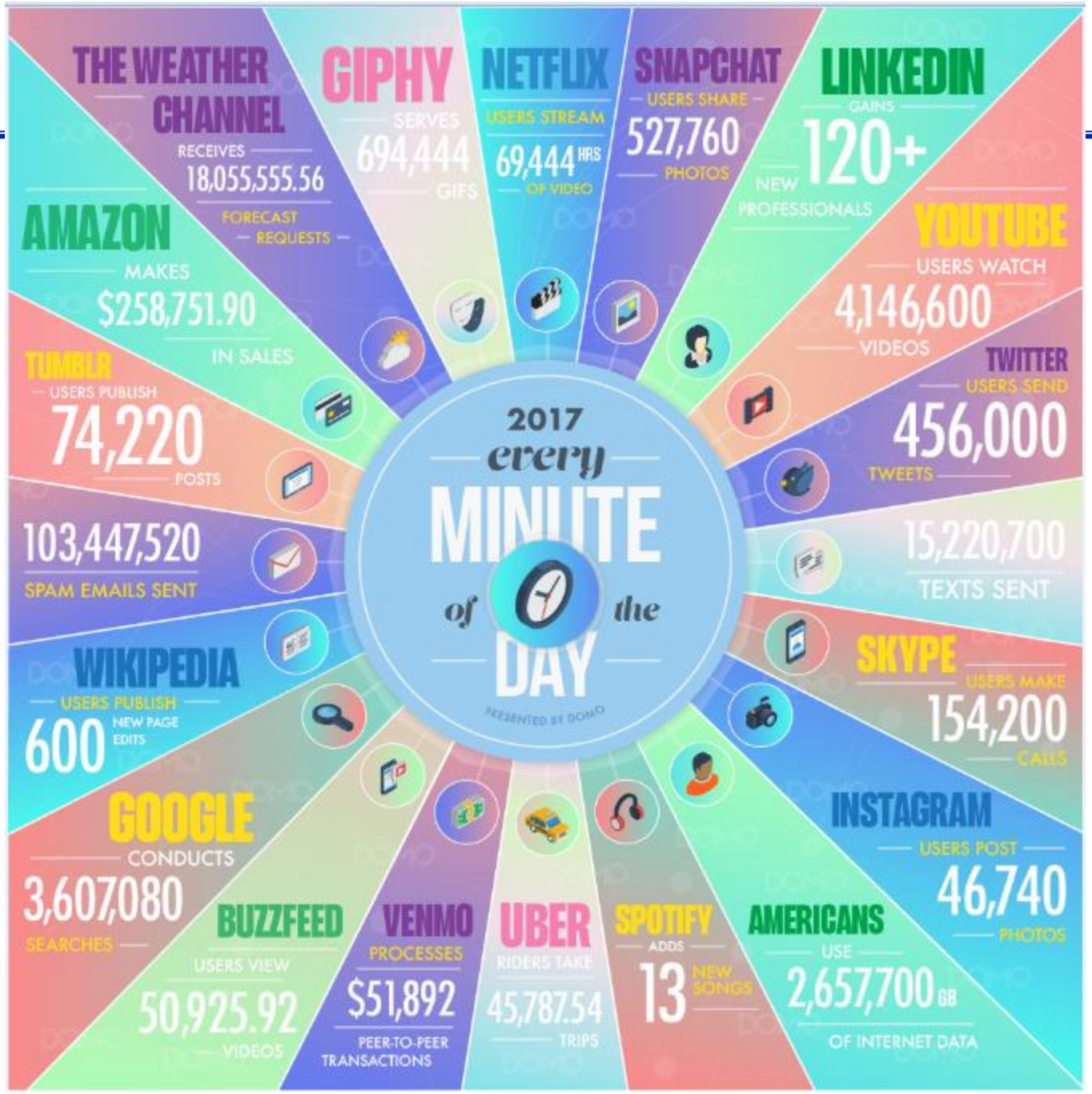
Source: IDC, 2016

Big Data Overview

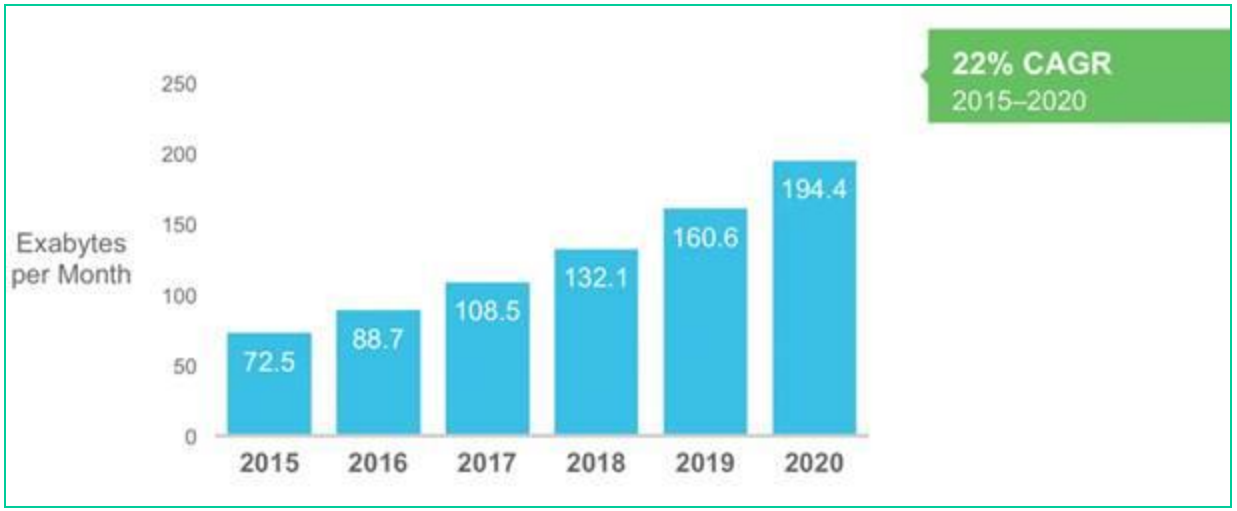
- 1,209,600 new data producing social media users each day.
- 656 million tweets per day!
- More than 4 million hours of content uploaded to Youtube every day, with users watching 5.97 billion hours of Youtube videos each day.
- 67,305,600 Instagram posts uploaded each day
- There are over 2 billion monthly active facebook users, compared to 1.44 billion at the start of 2015 and 1.65 at the start of 2016.
- Facebook has 1.32 billion daily active users on average as of June 2017
- 4.3 BILLION Facebook messages posted daily!
- 5.75 BILLION Facebook likes every day.
- 22 billion texts sent every day.
- 5.2 BILLION daily Google Searches in 2017.

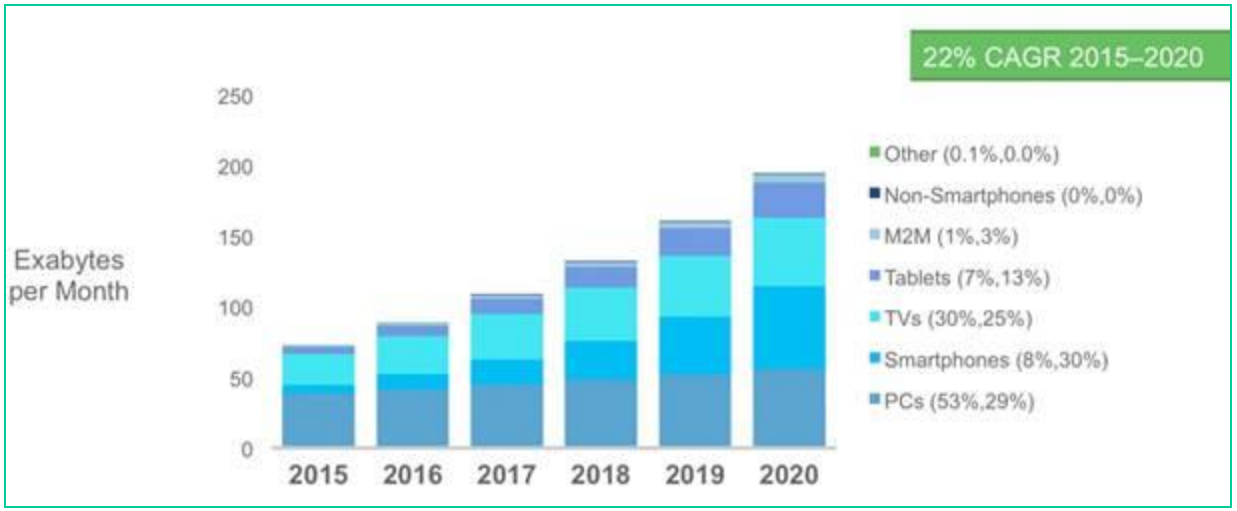
Big Data Overview

- The Mobile app Millennials used the most in 2015 – Facebook (21% of users)
- The mobile app Millennials used the most in 2017 – Amazon (35% of users)
- The time spent per user with digital media on mobile in US daily in 2017 – 2.3 hours
- The age group that spends the most time on apps monthly in US in 2015 – 18-24 (90.6 hrs on smartphone apps, 34.7 hrs. on tablet apps)
- The age group that spends the most time on apps monthly in US in 2017- 18-24 (93.5 hrs / month on smartphone apps, 27.6 hrs. / month on tablet apps)
- The major age group of users that operate a smartphone with two hands – 55+ (39% of users)


2017
every
MINUTE
of the
DAY
PRESENTED BY DOMO
THE WEATHER CHANNEL
RECEIVES
18,055,555.56
FORECAST REQUESTS
GIPHY
SERVES
694,444
GIFS
NETFLIX
USERS STREAM
69,444 HRS
OF VIDEO
SNAPCHAT
USERS SHARE
527,760
PHOTOS
LINKEDIN
GAINS
120+
NEW PROFESSIONALS
AMAZON
MAKES
$258,751.90
IN SALES
YOUTUBE
USERS WATCH
4,146,600
VIDEOS
TUMBLR
USERS PUBLISH
74,220
POSTS
TWITTER
USERS SEND
456,000
TWEETS
103,447,520
SPAM EMAILS SENT
15,220,700
TEXTS SENT
WIKIPEDIA
USERS PUBLISH
600
NEW PAGE EDITS
SKYPE
USERS MAKE
154,200
CALLS
GOOGLE
CONDUCTS
3,607,080
SEARCHES
INSTAGRAM
USERS POST
46,740
PHOTOS
BUZZFEED
USERS VIEW
50,925.92
VIDEOS
VENMO
PROCESSES
$51,892
PEER-TO-PEER TRANSACTIONS
UBER
RIDERS TAKE
45,787.54
TRIPS
SPOTIFY
ADDS
13
NEW SONGS
AMERICANS
USE
2,657,700 GB
OF INTERNET DATA

Big Data Overview





Cisco The Zettabyte Era—Trends and Analysis http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

1 THE RAPID GROWTH OF GLOBAL DATA

CSC

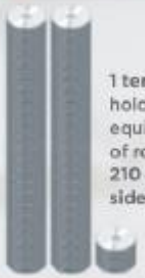
The production of data is expanding at an astonishing pace. Experts now point to a 4300% increase in annual data generation by 2020. Drivers include the switch from analog to digital technologies and the rapid increase in data generation by individuals and corporations alike.

Size of Total Data
Enterprise Managed Data
Enterprise Created Data

2020: MORE THAN 1/3 OF THE DATA PRODUCED WILL LIVE IN OR PASS THROUGH THE CLOUD.

WHAT IS A ZETTABYTE?

1,000,000,000,000	gigabytes
1,000,000,000,000	terabytes
1,000,000,000,000	petabytes
1,000,000,000,000	exabytes
1,000,000,000,000,000	zettabyte



1 terabyte holds the equivalent of roughly 210 single-sided DVDs.

It took roughly 1 petabyte of local storage to render the 3D CGI effects in Avatar.

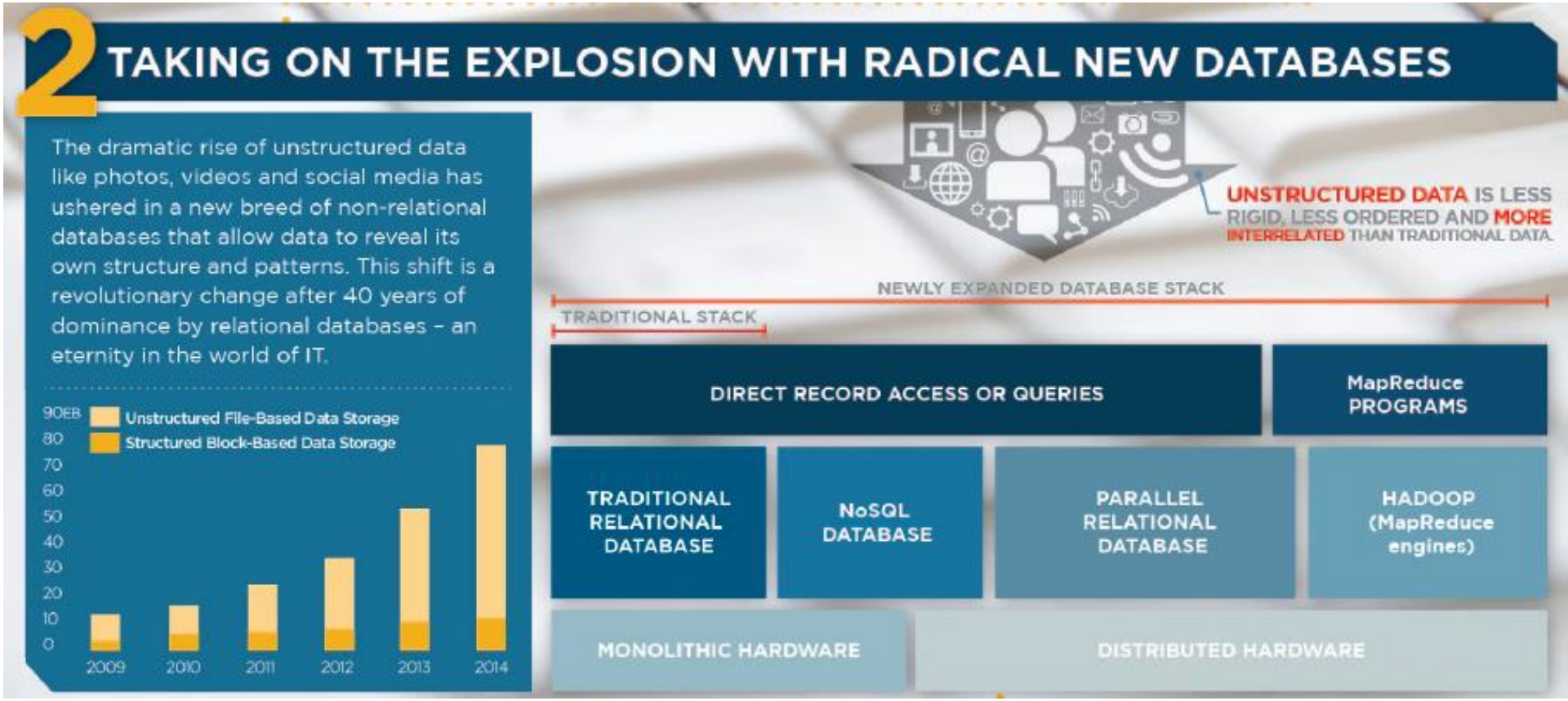
In 2007, the estimated information content of all human knowledge was 295 exabytes.

DATA PRODUCTION WILL BE 44 TIMES GREATER IN 2020 THAN IT WAS IN 2009

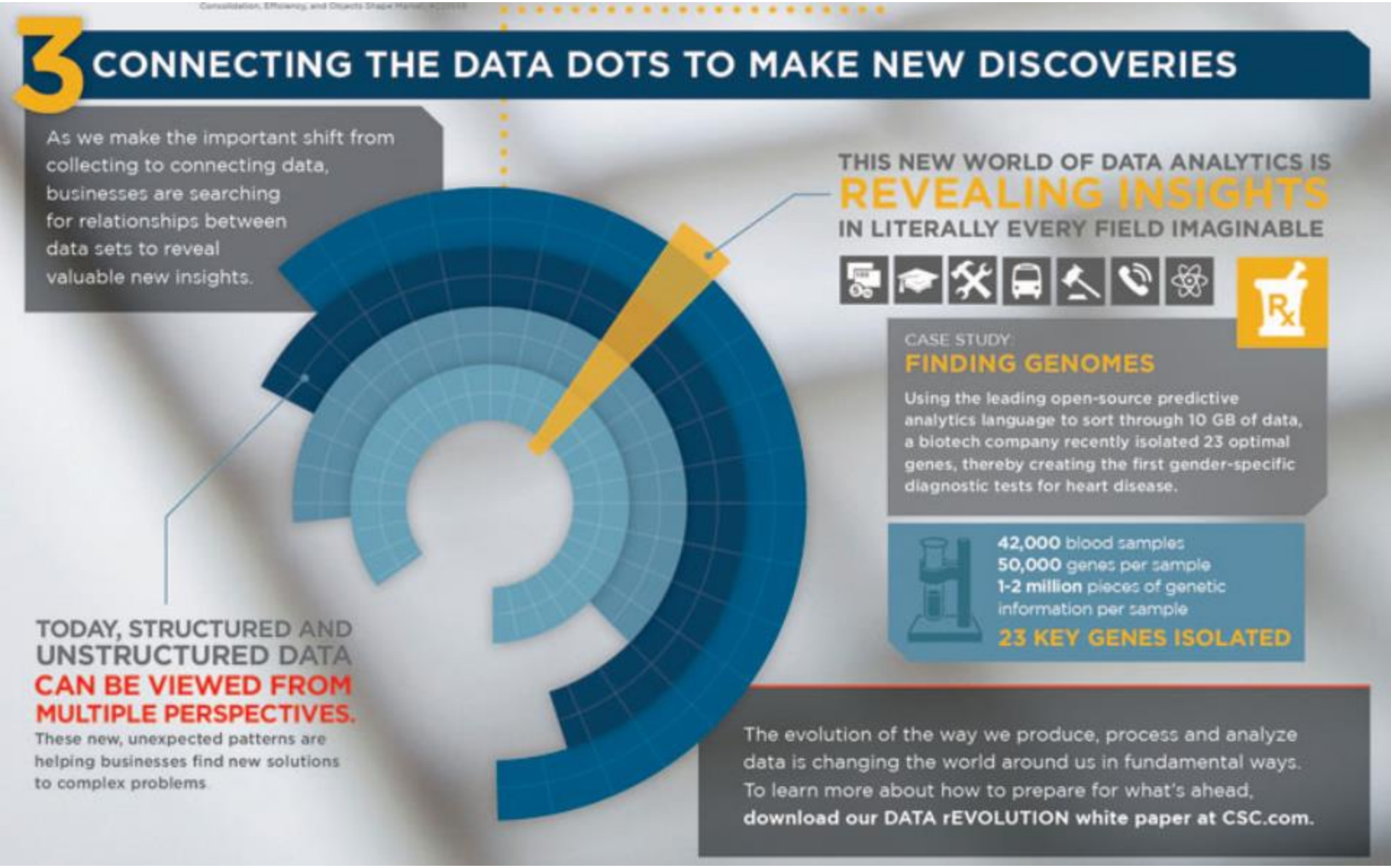
More than 70% of the digital universe is generated by individuals. But enterprises have responsibility for the storage, protection and management of 80% of it.*

(1) http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

Big Data Overview



(1) http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode



(1) http://www.csc.com/insights/flxwd/78931-big_data_universe_beginning_to_explode

Challenges

- How can we collect, process and store so much data?
- How can we wade through so much data in a timely manner?
- How can we analyze so much data and gain meaningful insights?
- Can my existing systems/architectures handle this?
  - Why not?
- Can my existing staff (skills & tools) make the transition?

The Relational Problem

- The Relational Database: 40+ year-old technology
  - Tables, Rows, Columns, Keys, Joins, Commits
  - ACID Transaction Compliance
  - RDBMS Software Choices
- Traditional Database Server Architecture
  - Memory, CPU, Storage
  - The cost of scaling “UP”

Alternatives for handling Big Data

- Scale “OUT”, not “UP”
- Use cheap, commodity hardware, local disk
- Use SSDs
- Seamless Fault Tolerance -- No Single Point of Failure
- Distribute the Data: Replication & Sharding
- Distribute the Processing: Parallelization
- Redundancy: Replication
  - Master-Master
  - Master-Slave
- Relaxed ACID compliance restrictions

Big Data Overview

- (whiteboard)

The CAP Theorem – Influences your strategy for handling Big Data

- Consistency – All nodes see the same data
- Availability – All user requests are served
- Partition Tolerance – Resilient handling of network failures
- **Theorem:** It is impossible for a distributed system to guarantee all three.
- TradeOffs must be made
- When a network failure occurs, either you sacrifice Availability or you sacrifice Consistency

"Better stale than offline"

Solutions:

- **Relational "Plus"**
  - Clustering, Parallelization, Scaling "out"
  - Examples: Teradata, Oracle RAC, MySQL Cluster
- **NoSQL** – Not Only SQL
  - Non-relational Data Model, Clustering, Parallelization, Scaling "out", Commodity Hardware, Relaxed ACID compliance
  - Examples: MongoDB, Cassandra, Neo4j, Hbase, Amazon Dynamo
- **Hadoop**
  - A Whole Different Ballgame…

- **Relational "Plus"**
  - Relational Model
  - ACID transaction compliant
  - Uses SQL
  - Leverages existing staff skills
  - Helps to handle big data without sacrificing relational advantages & prior investment

- **MySQL with Clustering**
  - Relational Model
  - ACID transaction compliant
  - Uses SQL
  - Leverages existing staff skills
  - Helps to handle big data without sacrificing relational advantages & prior investment
  - Open Source licensing
    - MariaDB
    - Oracle MySQL
    - Percona

- **MySQL with Clustering** (e.g. “Galera”)
  - Provides Horizontal Scaling and Fault tolerance
  - Can be Deployed on Cheap Commodity Hardware
  - Uses Replication to Distribute the Data, May also use Sharding
  - May include Geographic Distribution
  - Requires a “watcher” node
  - Master-Slave replication, Synchronous or Asynchronous
    - All WRITES to Master
    - READS from any node
  - Master-Master replication, Synchronous or Asynchronous
    - READ, WRITE on any node
    - Introduces Risk of Update Conflicts & Inconsistency

Big Data Overview

- **Teradata** (Typically used for Data Warehouse rather than OLTP)
  - Relational Model
  - ACID transaction compliant
  - Uses SQL
  - Leverages existing staff skills
  - Helps to handle big data without sacrificing relational advantages & prior investment
  - Very Costly, but Very Capable
  - Massively Parallel Workload Distribution

- **Teradata**
  - Shared-Nothing Architecture
  - Provides Horizontal Scaling and Fault Tolerance
    - Replication Factor across multiple nodes with Aster Analytics add-on
  - Data is Sharded via Hashing, Balanced Distribution Across Nodes
  - Re-Distributes Data as Nodes are Added
  - Robust Suite of System Management Tools
    - Workload Manager
    - Backup/Recovery, System Monitoring

Big Data Overview

- **Oracle RAC** (Real Application Clusters)
  - Provides a Form of Horizontal Scaling
  - Robust, Seamless Fault Tolerance, Automatic Failover
  - Shared Storage -- Multiple DBMS nodes share one copy of the database on disk
  - Robust Suite of System Management Tools
    - OEM – Oracle Enterprise Manager
  - Leverages existing staff skills
  - Costly, Relies on SAN
  - Data/Storage is not horizontally scalable

Big Data Overview

- **NoSQL** (“Not Only SQL”)
  - Distribute the Data and the Processing
  - Replication provides redundancy, high availability, parallel processing
  - Horizontally scalable
  - Does not store data in tables with rigid row/column structure
  - May restrict join capabilities
  - May restrict ACID transaction compliance
  - May use *non-standard* query language (typically NOT SQL)

- **NoSQL Data Schema/Models**
  - Document Store (using XML or JSON format)
  - Graph (using node/edge structures with properties)
  - Key-Value pairs
  - Wide Columnar store (rows with dynamic columns holding key-value pairs)

- **Document Database**
  - Organized around a “document” containing text
  - Can handle very large data volumes
  - Provides Speed and Scalability
  - Document format is easily understood by humans
  - No “schema”, but JSON/XML provides internal structure within a document
  - Documents are indexed and stored within “collections”
  - Supports full text search
- **Popular Implementations** (open source)
  - MongoDB
  - CouchDB

JSON

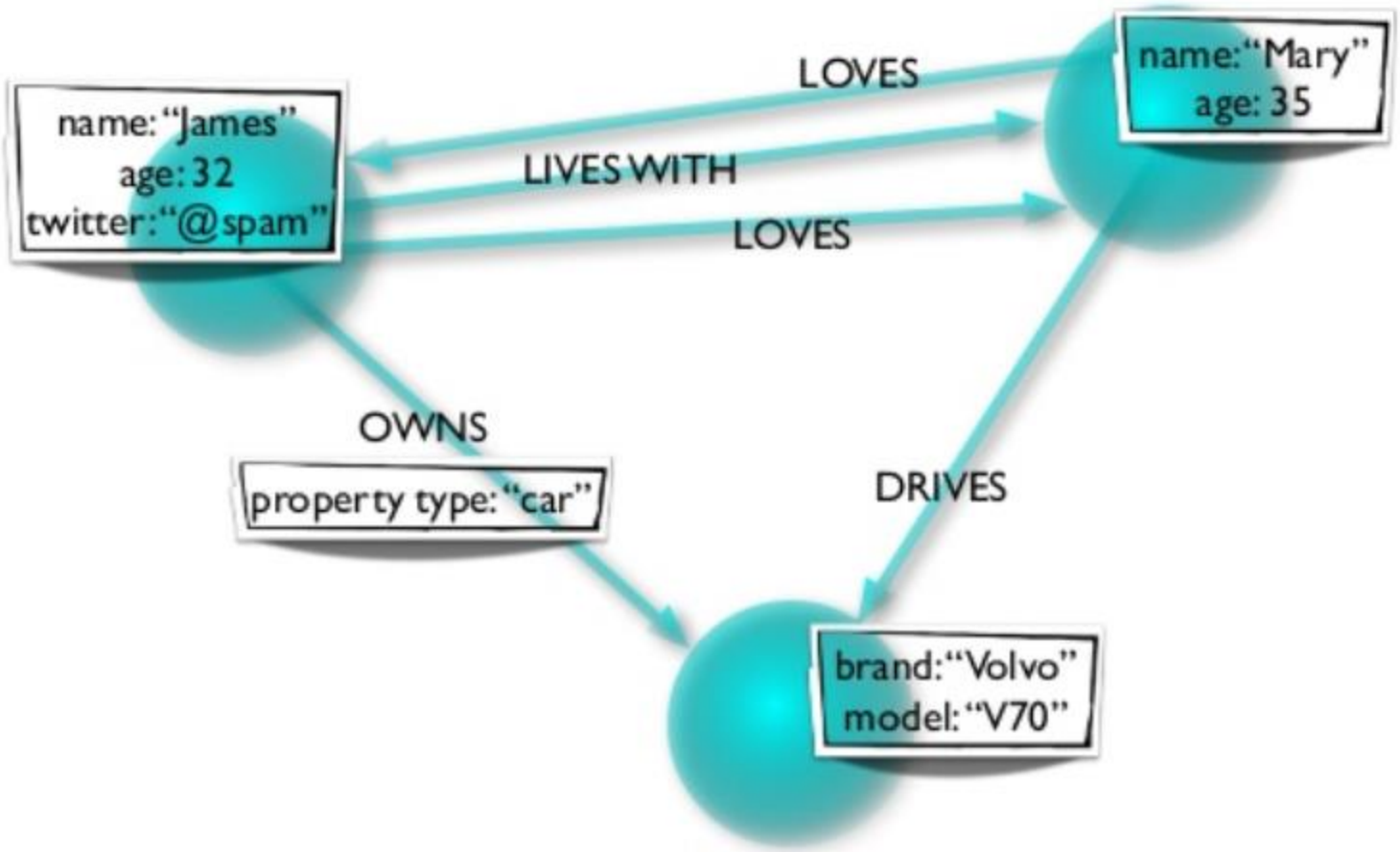
```
{

  "Subject": "I like Plankton"

  "Author": "Rusty"

  "PostedDate": "5/23/2016"

  "Tags": ["plankton", "baseball", "decisions"]

  "Body": "I decided today that I do not like baseball.
I like plankton."

}
```

XML

```
<contact>
        <firstname>Bob</firstname>
        <lastname>Smith</lastname>
        <phone type="Cell">(123)555-0178</phone>
        <phone type="Work">(890)555-0133</phone>
        <address>
                <type>Home</type>
                <street>123 Black St.</street>
                <city>Big Rock</city>
                <state>AR</state>
                <zip>32225</zip>
                <country>USA</country>
        </address>
</contact>
```

- **Document Database**
  - Keeps related information together (not normalized into tables)
  - Access to a document is fast (index/key/URL)
  - ACID compliance is maintained within a document
  - Cannot easily “join” across documents

- **Graph Database**
  - Uses a graph structure consisting of
    - Nodes – Represents an entity (like a person)
    - Edges – Represents a relationship between entities
    - Properties – Attributes associated with Nodes and Edges
  - Supports navigation along edges from a starting point node
  - Designed for applications tracking the inter-connections among entities
    - Who is friends with whom? (In a social network application)
    - Who is following me, who am I following?
  - Uses a pattern matching query language to navigate nodes & edges
- **Popular Implementations** (open source)
  - Neo4j


name: "James"
age: 32
twitter: "@spam"
LOVES
LIVES WITH
LOVES
name: "Mary"
age: 35
OWNS
property type: "car"
DRIVES
brand: "Volvo"
model: "V70"

- **Key-Value Pairs Database**
  - "Schemaless"
  - Maps a key to an opaque value, like an associative array
  - Simple operations (put, get, remove, modify)
  - Keys are unique in a collection
  - Most useful when access is by the primary key
  - May be a building block for other data models (such as key-value pairs within a wide-columnar store like Cassandra)
- **Popular Implementations**
  - Amazon Dynamo (available via AWS in the cloud)
  - Riak, Redis (open source)

Product Catalog Example

```
{
    Id: 206,
    Title: "20-Bicycle 206",
    Description: "206 description",
    BicycleType: "Hybrid",
    Brand: "Brand-Company C",
    Price: 500,
    Color: ["Red", "Black"],
    ProductCategory: "Bike",
    InStock: true,
    QuantityOnHand: null,
    RelatedItems: [
        341,
        472,
        649
    ],
    Pictures: {
        FrontView: "http://example.com/products/206_front.jpg",
        RearView: "http://example.com/products/206_rear.jpg",
        SideView: "http://example.com/products/206_left_side.jpg"
    },
    ProductReviews: {
        FiveStar: [
                "Excellent! Can't recommend it highly enough!  Buy it!",
                "Do yourself a favor and buy this."
        ],
        OneStar: [
                "Terrible product!  Do not buy this."
        ]
    }
}
```

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.CaseStudy.html

- **Key-Value Pairs Database Example (Amazon DynamoDB)**
- The key value (Id) is 206.
- Most of the attributes have simple data types, such as String, Number, Boolean and Null.
- One attribute (Color) is a String Set.
- The following attributes are document data types:
  - A List of RelatedItems. Each element is an Id for a related product.
  - A Map of Pictures. Each element is a short description of a picture, along with a URL for the corresponding image file.
  - A Map of ProductReviews. Each element represents a rating and a list of reviews corresponding to that rating. Initially, this map will be populated with five-star and one-star reviews.

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.CaseStudy.html

- **Wide-Columnar (Column Family) Store Database**
  - Data is stored within “collections” of dynamic related columns
  - Similar to key/value with the value having columnar structure
- **Popular Implementations** (open source)
  - Cassandra
  - HBase

- **Wide-Columnar Store Example**
- Consider an application that needs to store stock trades and retrieve them by stock symbol
- Some data that might be stored includes the following:
  - symbl: The stock symbol
  - id: A unique trade id
  - date: Date of the trade
  - price: Trade price
  - quant: Number of shares traded
  - notes: General comment field.

- Below, we illustrate trade data in a wide column store
  - The stock symbol (symb) is the row key
  - Generally the complete row is stored on a single node
  - The trades are stored as columns in the row
  - If there were many trades, the row could get very wide
  - Note also, that Trade 2 has no notes column
  - Contrast this to a normalized relational model with a trade table which would generally store trades in separate rows

symb	id	date	price	quant	notes	id	…	quant	id	…	notes
GM	1c3f	2014-09-15	32.96	100	blah	2a5f		400	2e99		bar

Benefits of a Wide Column Store

- Lookup of rows by row key can be **very fast**
- In a distributed system, the complete row is stored on one node
- May be stored redundantly across the cluster
- The can be retrieved with one access
- Good fit for **scale-out** systems
- Can scale to large capacity and high availability
- The columns are "**sparse**"
  - That is, column with no values are absent, saving space
  - As shown in Trade 2 having no Notes column
- Flexible access to column data in a row
  - You can flexibly query on them
  - e.g. - get all trades, get the last 10 trades, or 1st ten trades, etc.

Issues with a Wide Column Store

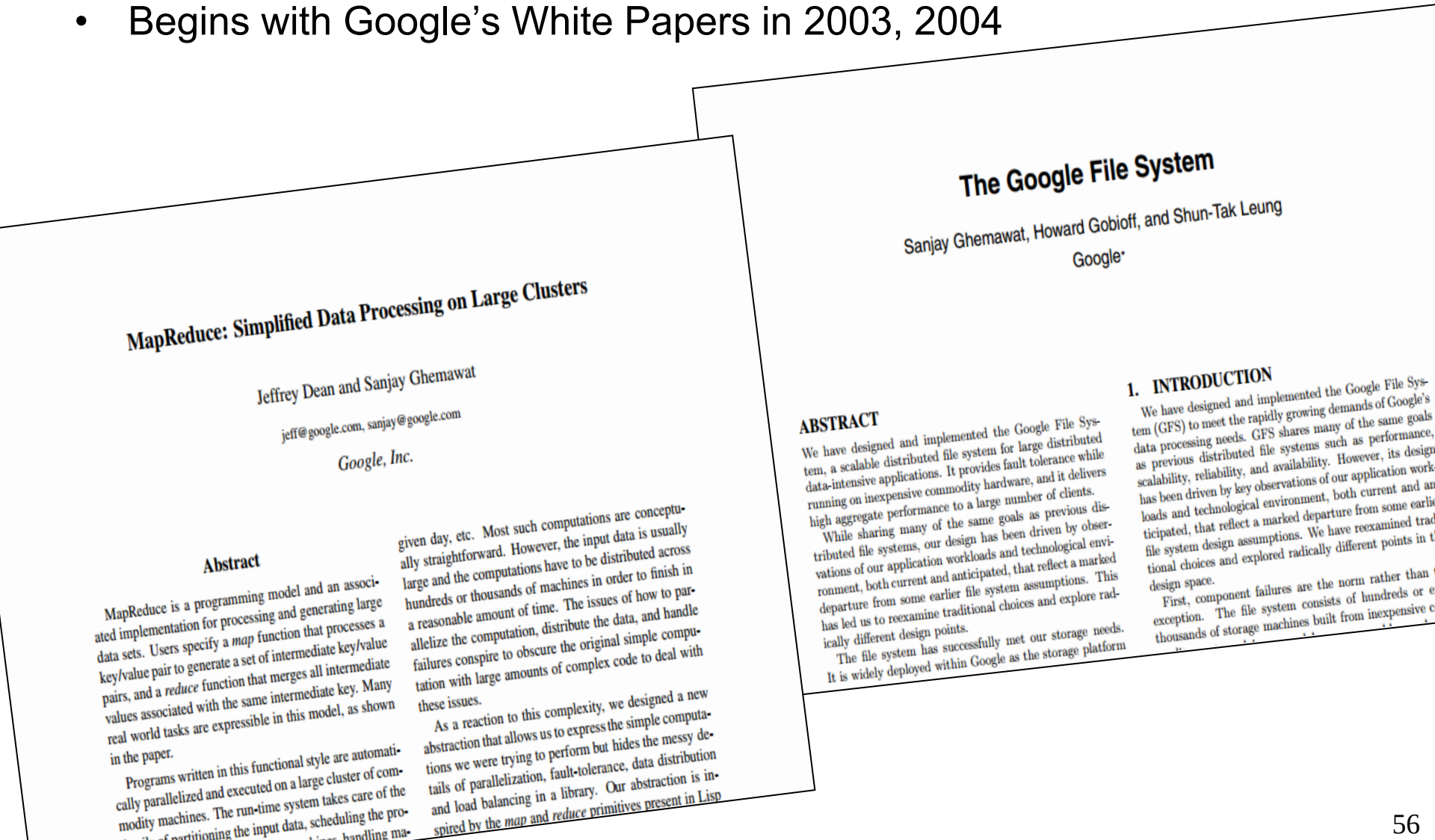
- The data model is **complex**, and **not very intuitive**
- Generally, you create your data model **based on your queries**
  - Possibly taking into account how data is distributed
- For example, querying for trades by stock symbol is very fast for the trade table shown earlier -- it's just one query to one node
- Consider finding all trades (for all stocks) on a particular date
  - The previous data model is not very good for this
  - You'd need to query every node, get the trades from that node,then combine them all for the result
- In practice, that may not scale
- You could, however, store data optimized for retrieval by date
- This would require another table / column layout

Issues with a Wide Column Store

- Client code **may be extensive**
  - For example, to keep track of trades by both stock symbol and trade date, you may need to maintain two sets of data
  - Which needs to be done via application code on a client
- Different from relational model where you maintain one set of data, and just write queries as needed
- However, wide column stores tend to have much more scale out capability -- Which is why you use them in the first place

Hadoop -- History and evolution

- Begins with Google's White Papers in 2003, 2004



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

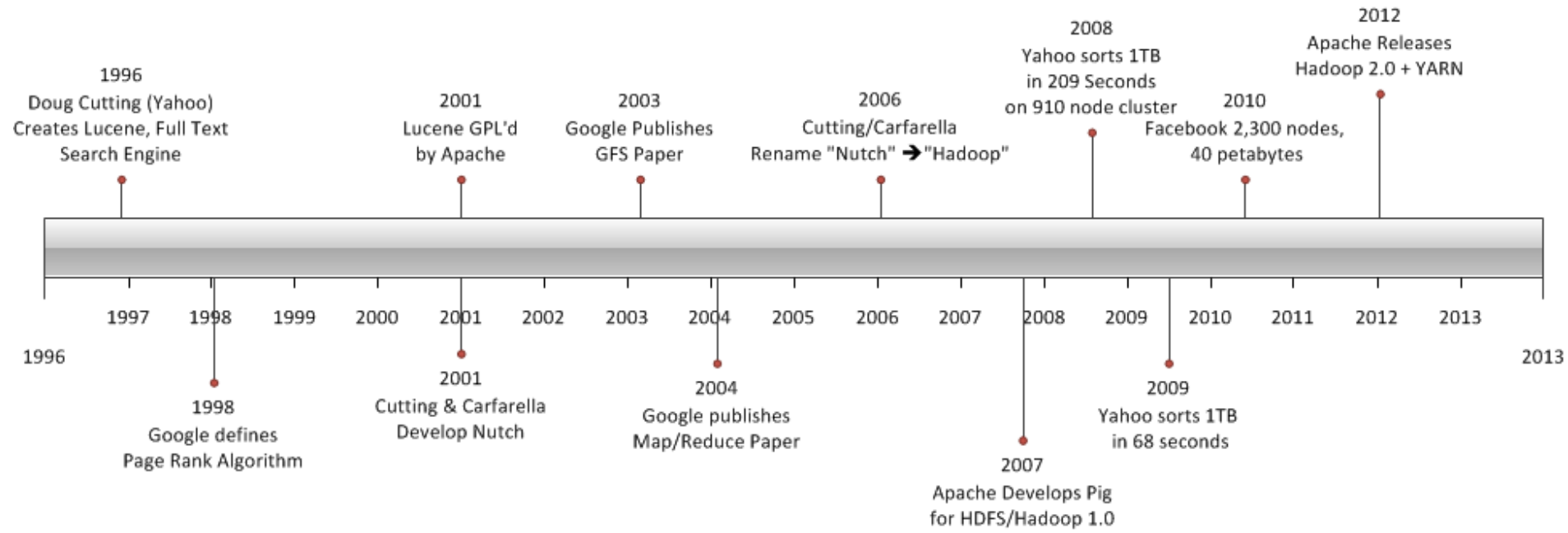
The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and an ticipated, that reflect a marked departure from some earli file system design assumptions. We have reexamined trad tional choices and explored radically different points in t design space.

First, component failures are the norm rather than exception. The file system consists of hundreds or thousands of storage machines built from inexpensive c

Hadoop – Timeline



Hadoop – Principles

- Parallelization (for throughput)
- Distribution of Data and Compute Tasks
- On cheap commodity hardware (physical, not virtual)
- Local dedicated disk
- Fault Tolerance
- Redundancy Factor
- Relaxed ACID compliance
- Chunks = Large Blocks
- Linear Scalability
- Write Once Read Many
- Open Source

Hadoop Provides

- **A distributed filesystem (HDFS) that can store data across thousands of servers**
- **A means of running work (MapReduce & YARN) across those machines, running the work near the data**

Apache

http://wiki.apache.org/hadoop/ProjectDescription

"A distributed filesystem that can store data across thousands of servers"

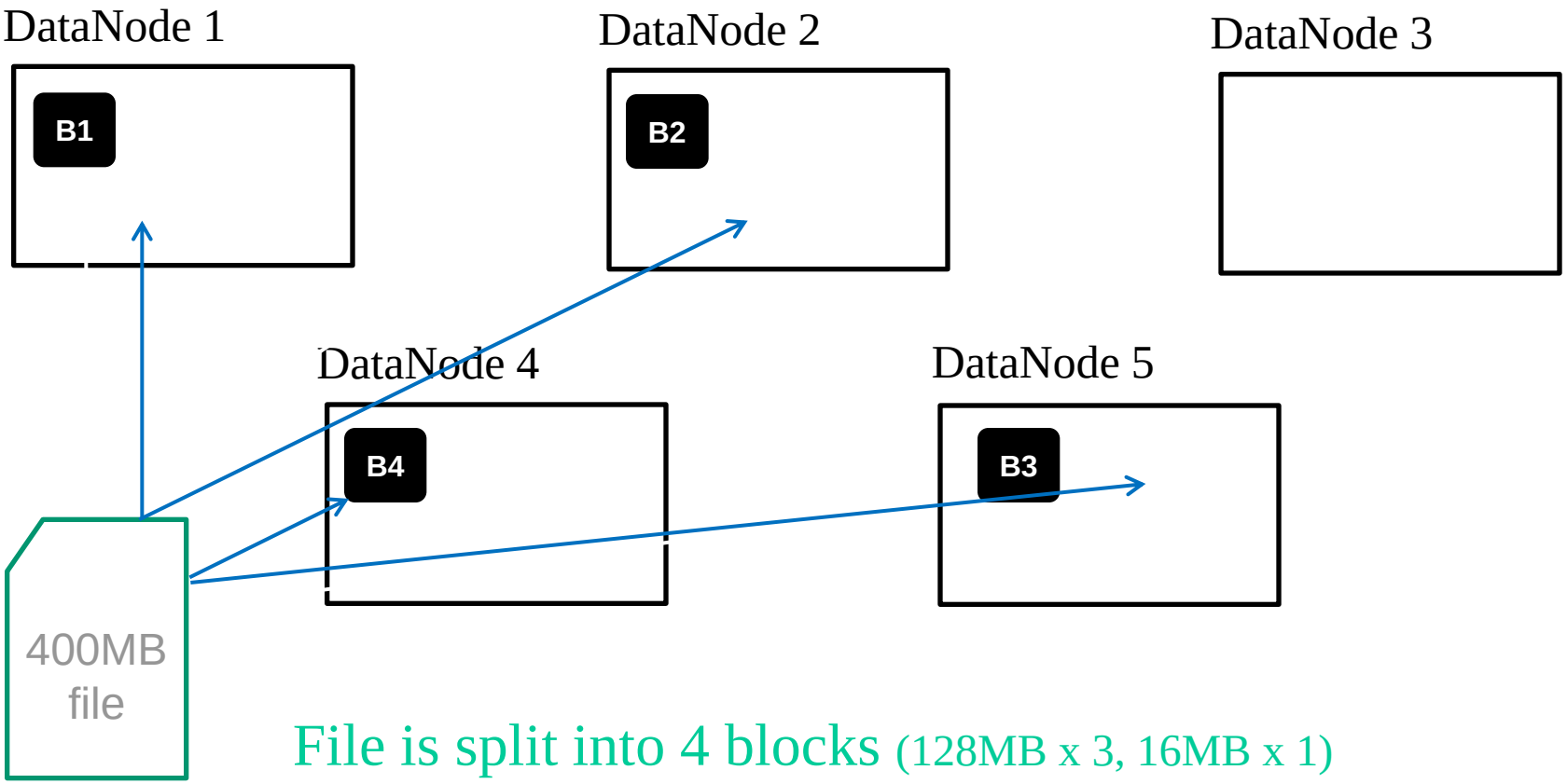
- **Cluster of Commodity Servers**
  - (Physical Boxes or in the Cloud)
- **LOTS of Memory**
- **LOTS of Inexpensive unprotected storage**

Key Features

Write Once Read Many

- Logical filesystem – sits on top of native OS
- Files split into Blocks
- Blocks spread across the cluster

400 MB file to write to the cluster

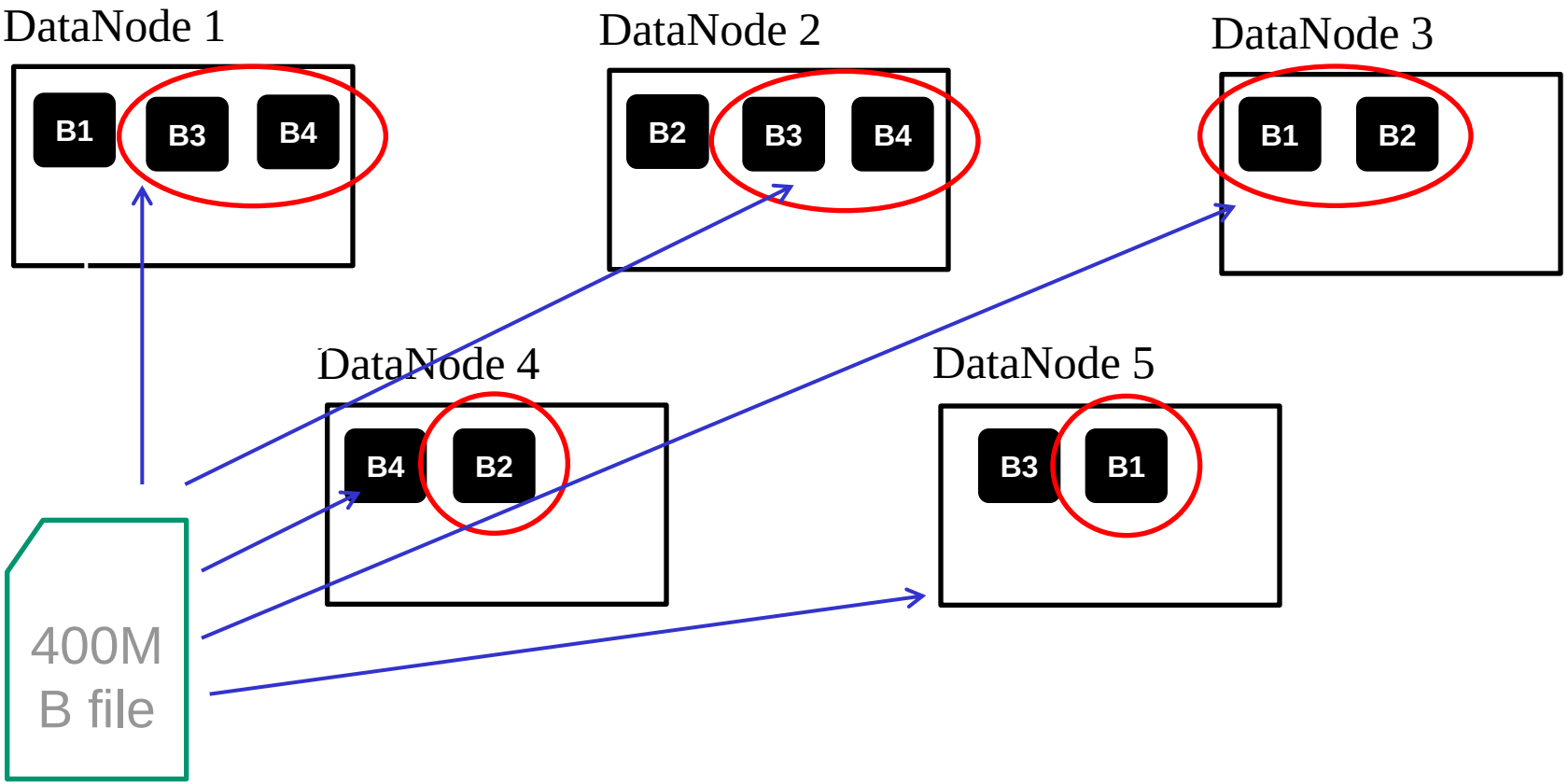


File is split into 4 blocks (128MB x 3, 16MB x 1)

Data Blocks copied to 3 different nodes

- **Fault-Tolerance**
- **Multiple copies provide performance boost**
- **Replication Level is configurable**
- **Full checksums**
- **Rack awareness**

Data Blocks are copied to three nodes



Looks just like a conventional filesystem…

bpreachuk — root@sandbox:~ — ssh — 112×29

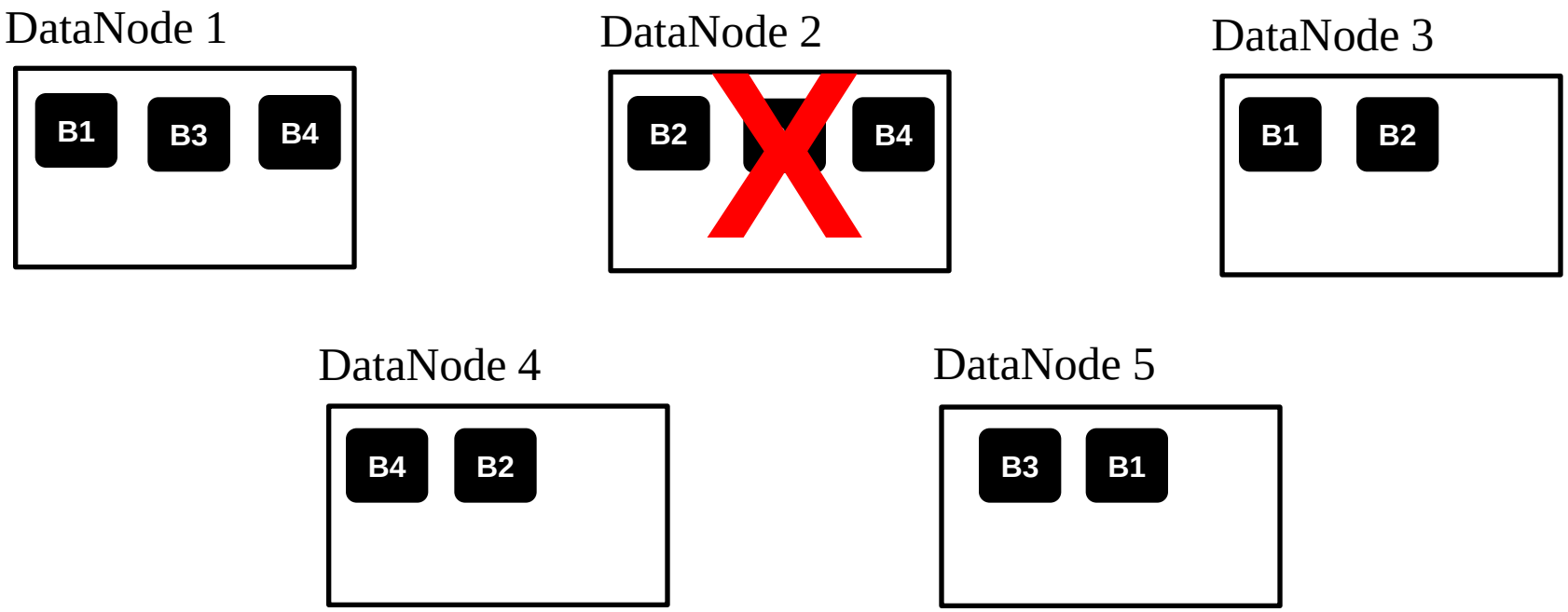
```
[root@sandbox ~]# hadoop fs -ls /apps
Found 5 items
drwxrwxrwx   - falcon hdfs          0 2014-04-21 07:15 /apps/falcon
drwxr-xr-x   - hdfs   hdfs          0 2014-04-21 07:16 /apps/hbase
drwxr-xr-x   - hdfs   hdfs          0 2014-04-21 07:17 /apps/hive
drwxr-xr-x   - tez    hdfs          0 2014-04-21 07:17 /apps/tez
drwxr-xr-x   - hcat   hdfs          0 2014-04-21 07:23 /apps/webhcat
[root@sandbox ~]#
[root@sandbox ~]#
[root@sandbox ~]# hadoop fs -ls -R /apps/hive/warehouse/sample_07
-rw-r--r--   1 hue hdfs      46055 2014-04-22 07:21 /apps/hive/warehouse/sample_07/sample_07.csv
[root@sandbox ~]#
[root@sandbox ~]#
[root@sandbox ~]# hadoop fs -ls  /apps/hive/warehouse
Found 5 items
drwxr-xr-x   - root hdfs          0 2014-08-23 00:18 /apps/hive/warehouse/generic_dw.db
drwxr-xr-x   - hue  hdfs          0 2014-04-22 07:21 /apps/hive/warehouse/sample_07
drwxr-xr-x   - hue  hdfs          0 2014-04-22 07:21 /apps/hive/warehouse/sample_08
drwxr-xr-x   - root hdfs          0 2014-08-22 14:55 /apps/hive/warehouse/tpcds_bin_partitioned_orc_5.db
drwxr-xr-x   - root hdfs          0 2014-08-22 15:26 /apps/hive/warehouse/tpcds_text_5.db
[root@sandbox ~]#
```

HDFS – Fault-Tolerance

Node failure is **Expected**

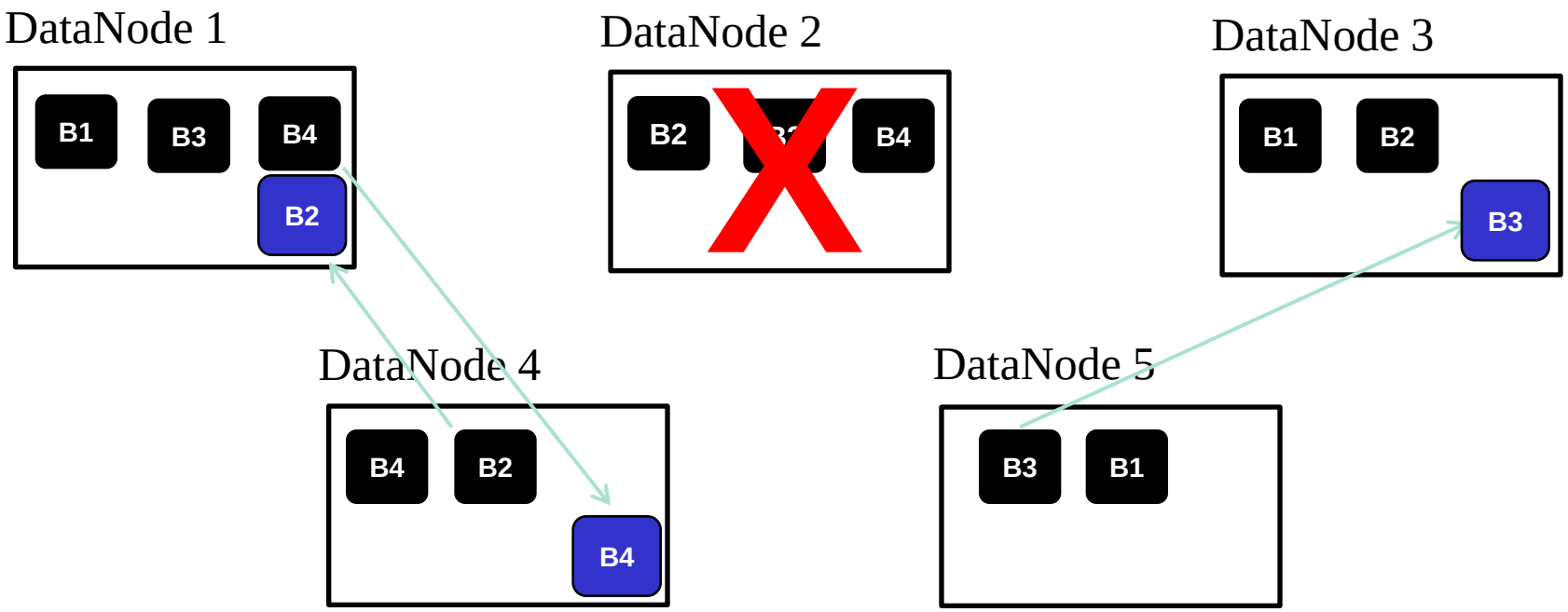
- Cluster detects missing/under-replicated blocks
- Cluster detects failed/unreachable nodes
- HDFS re-replicates data to maintain 3-copy protection level

Node Failure is Expected



When a node fails (or is unreachable)…

HDFS re-replicates data



Maintains 3-copy protection level

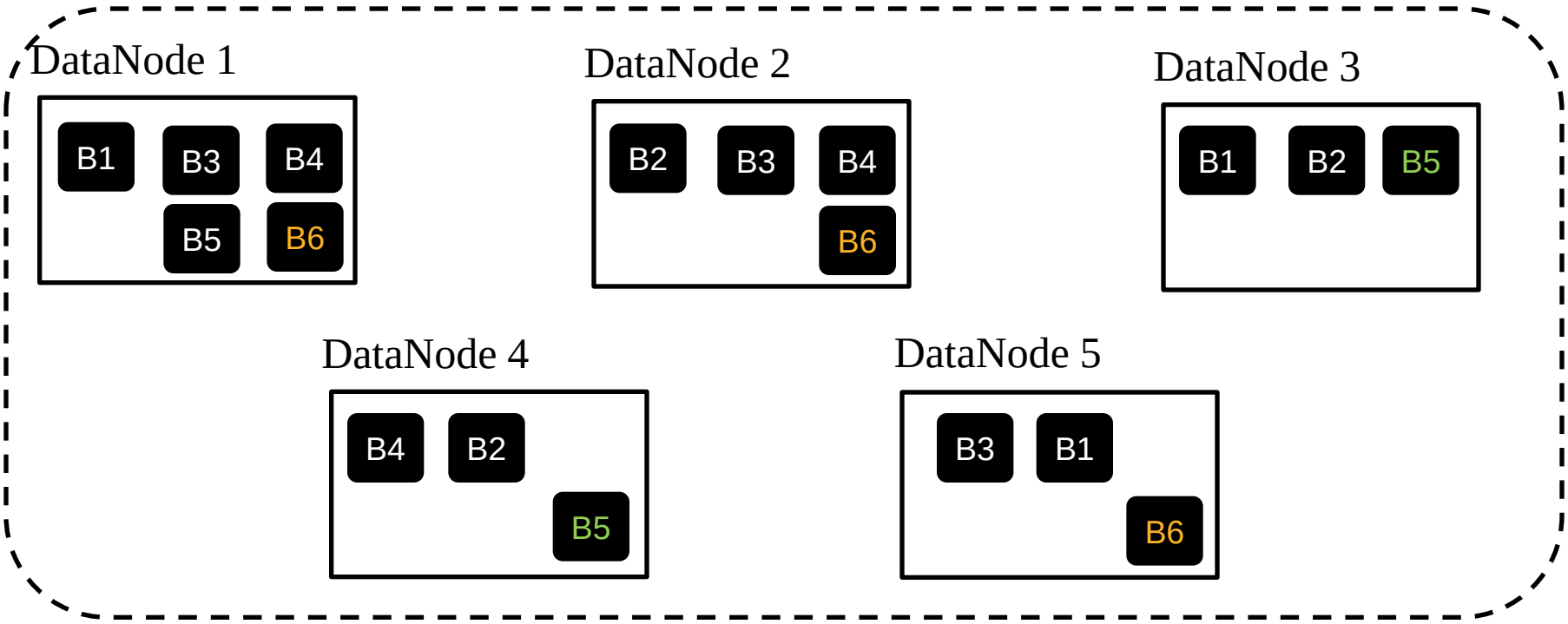
How does the cluster know:

- **What to replicate**
- **Where data is stored?**
- **If a block has been corrupted?**
- **If a block is over or under-replicated?**
- **If a data node has failed or is unreachable?**

The NameNode(s)

- **Critical High-Performance Node**
- **Contains all metadata for data blocks**
- **Keeps metadata in-memory**
- **Controls re-replication of missing data**

HDFS – NameNode



NameNode

StoreSales.txt - blocks B1, B2, B3, B4
Promos.csv - blocks B5, B6

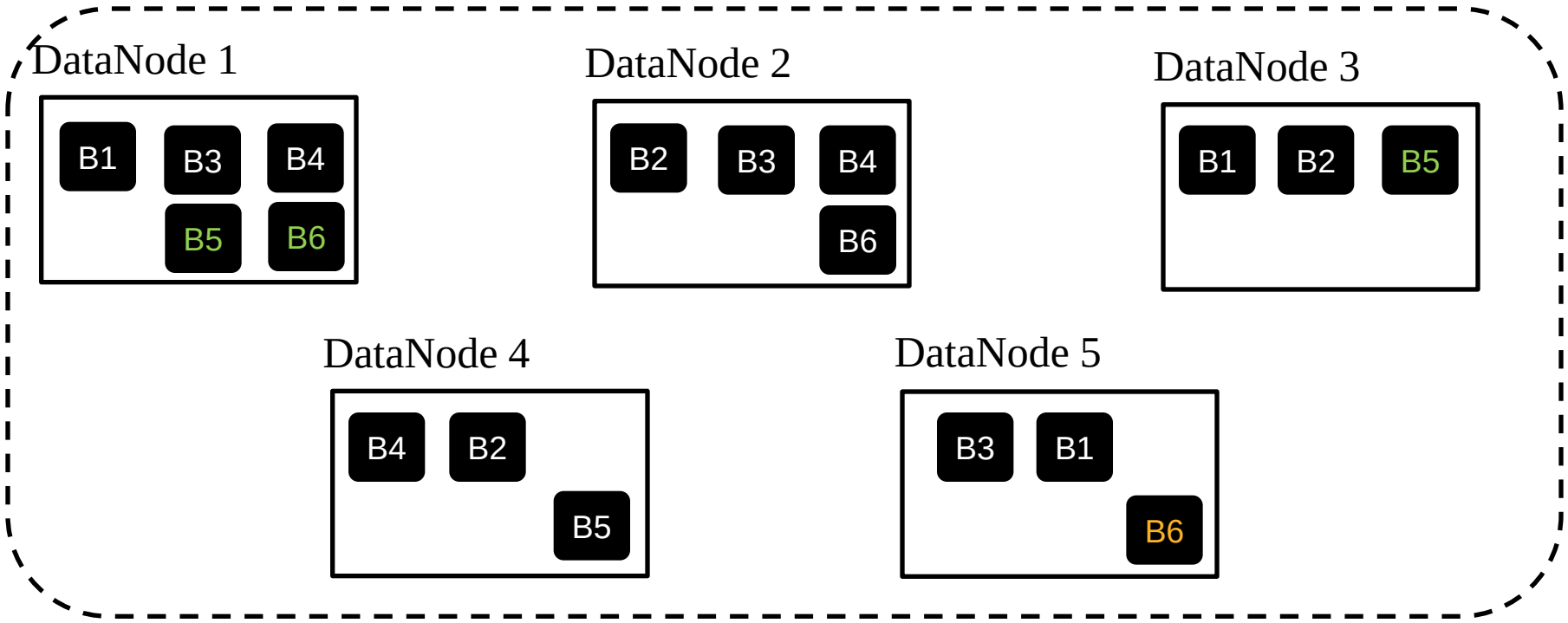
- Critical High-Performance Node
- Contains all metadata for data blocks
- Keeps metadata in-memory
- Controls re-replication of missing data

HDFS - Communications

- **Data Node Heartbeats**
- **Data Node Block Reports**
- **Rack Awareness**

If Name Node is down… Hadoop is down!

HDFS – Standby NameNode



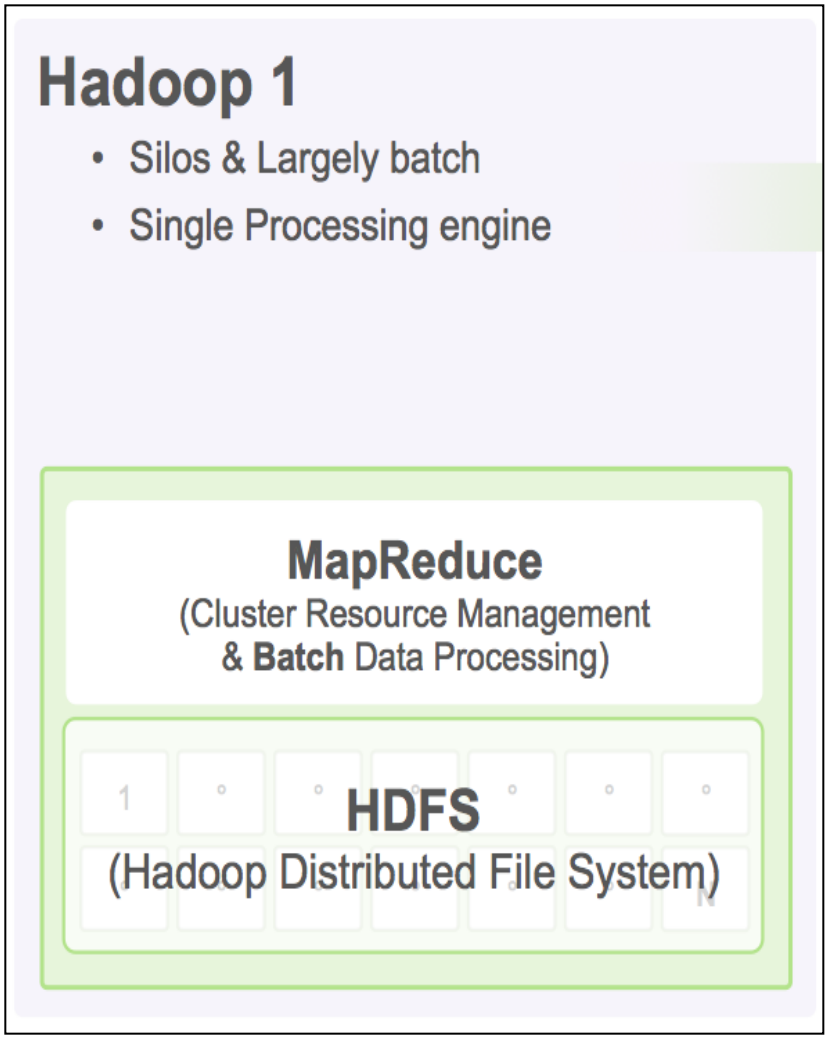
NameNode

StoreSales.txt - blocks B1, B2, B3, B4
Promos.csv - blocks B5, B6

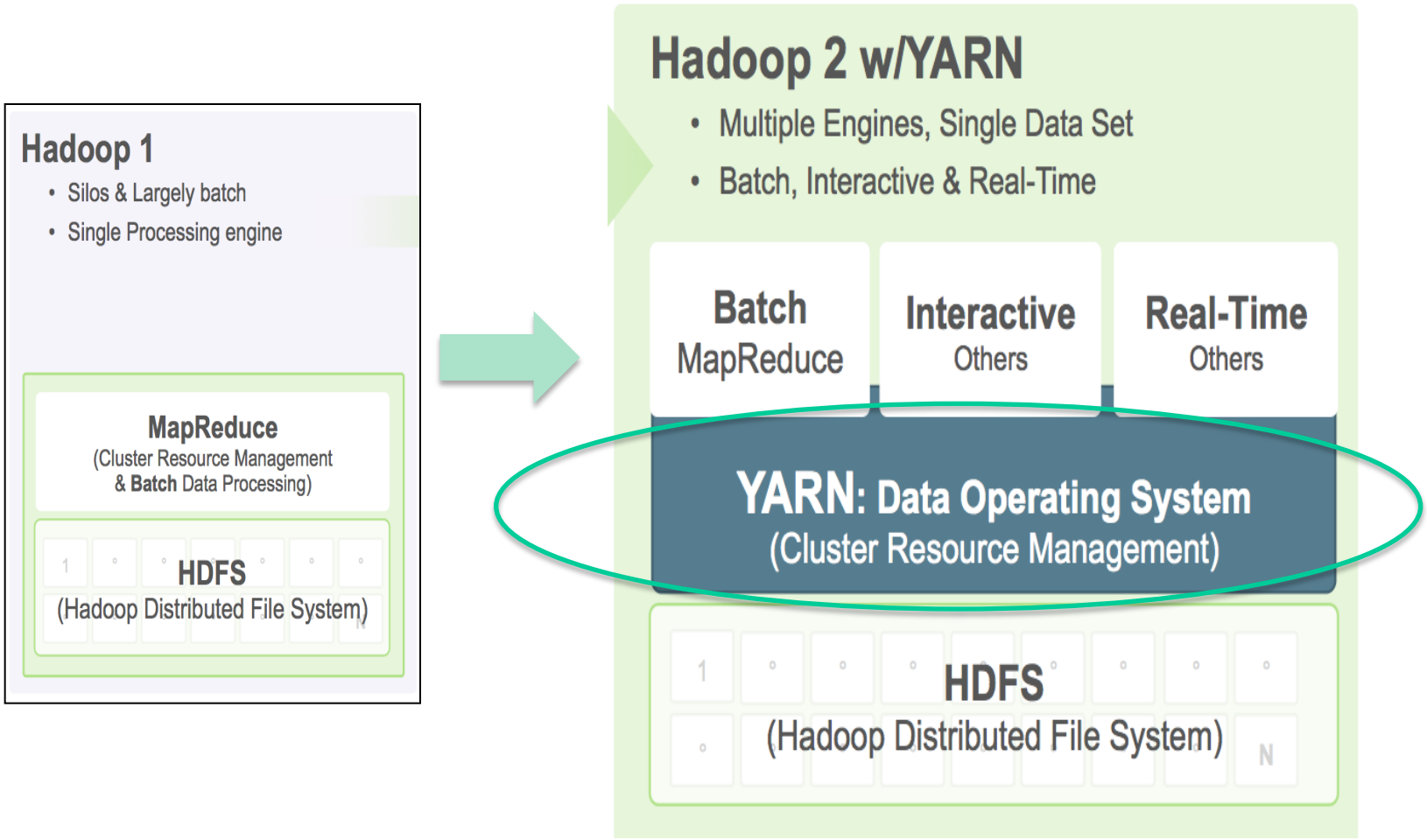
Standby NameNode

Redundant Name Space
Metadata Checkpoints to disk

MapReduce – Hadoop 1.0



YARN/Tez – Hadoop 2.0



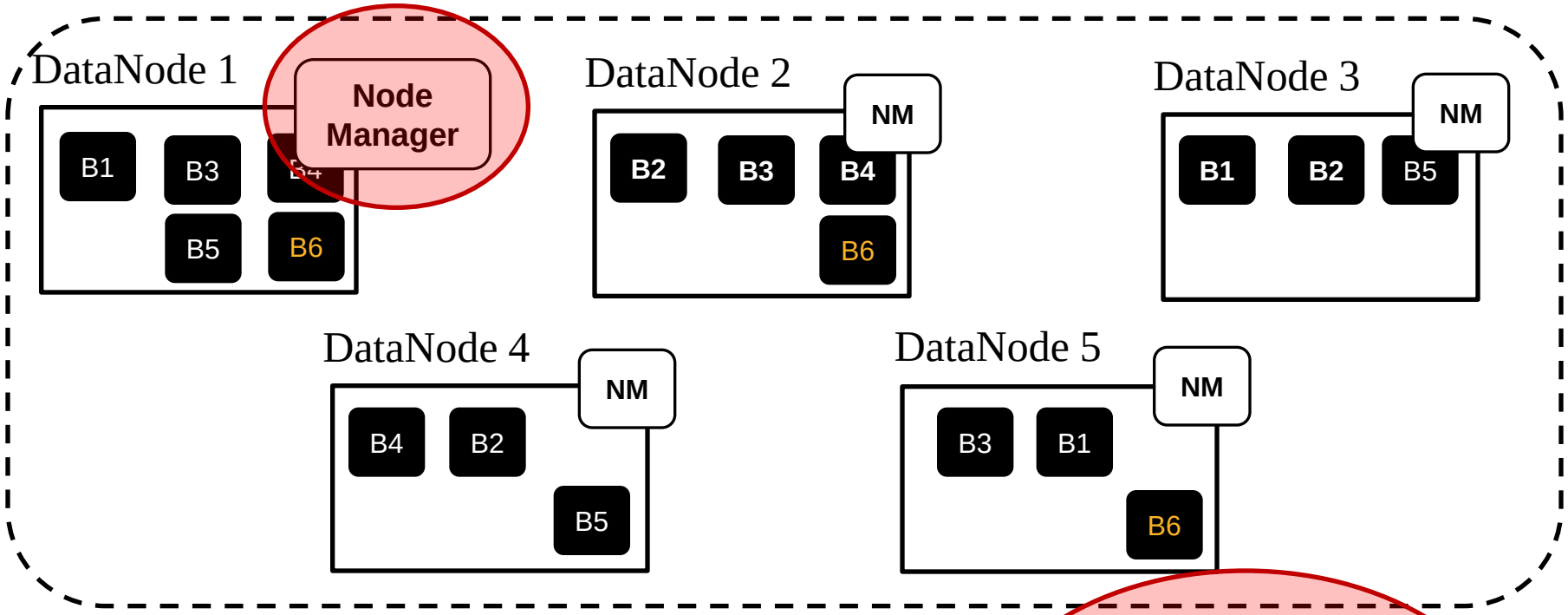
The Data Operating System

- More ways to process data than just MapReduce - Tez
- Improved Scalability & cluster Resource Utilization
- Mixed Workloads – Batch, Streaming, Online

So how do we run 'work' against a Hadoop cluster?

- Jobs are split into Tasks
- Tasks are sent to the Data ***
- Tasks are performed in parallel

Hadoop Job Execution



NameNode

StoreSales.txt
B1, B2, B3, B4
Promos.csv - B5, B6

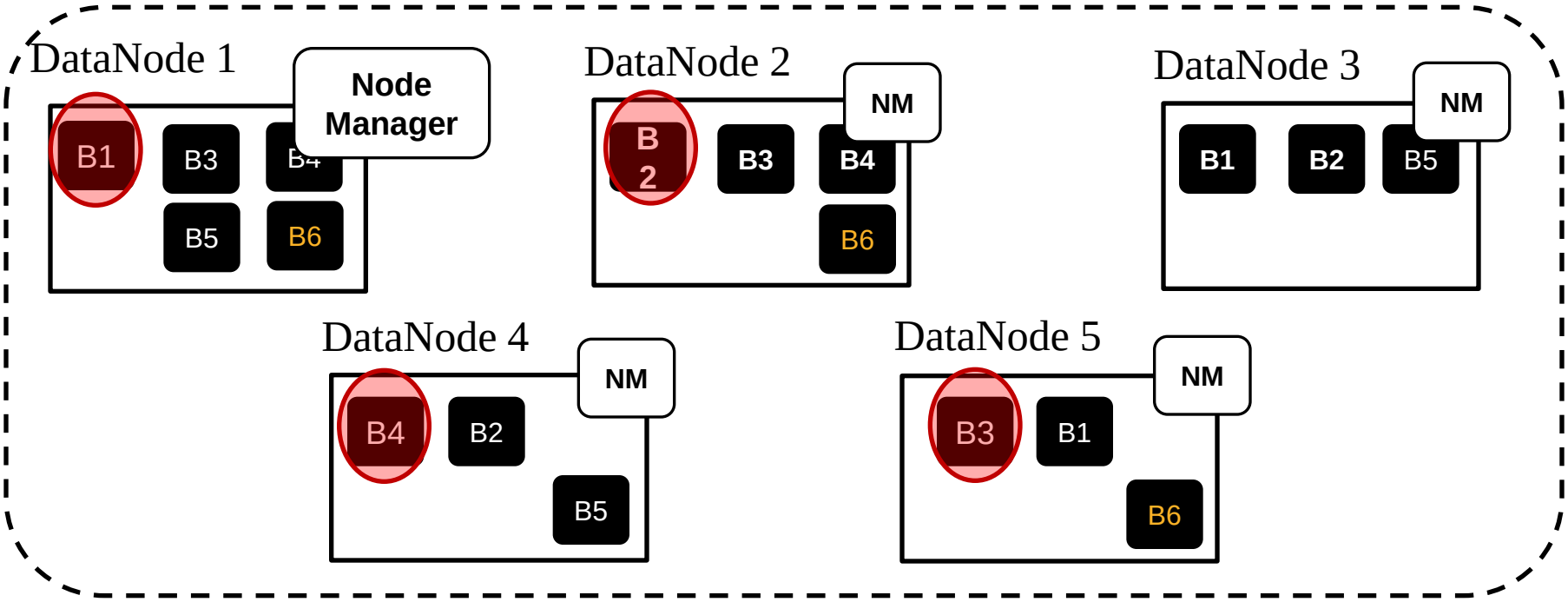
Resource Manager

Resource Management Scheduling

Client

Job #1
Store Sales

Hadoop Job Execution



NameNode

StoreSales.txt
B1, B2, B3, B4
Promos.csv - B5, B6

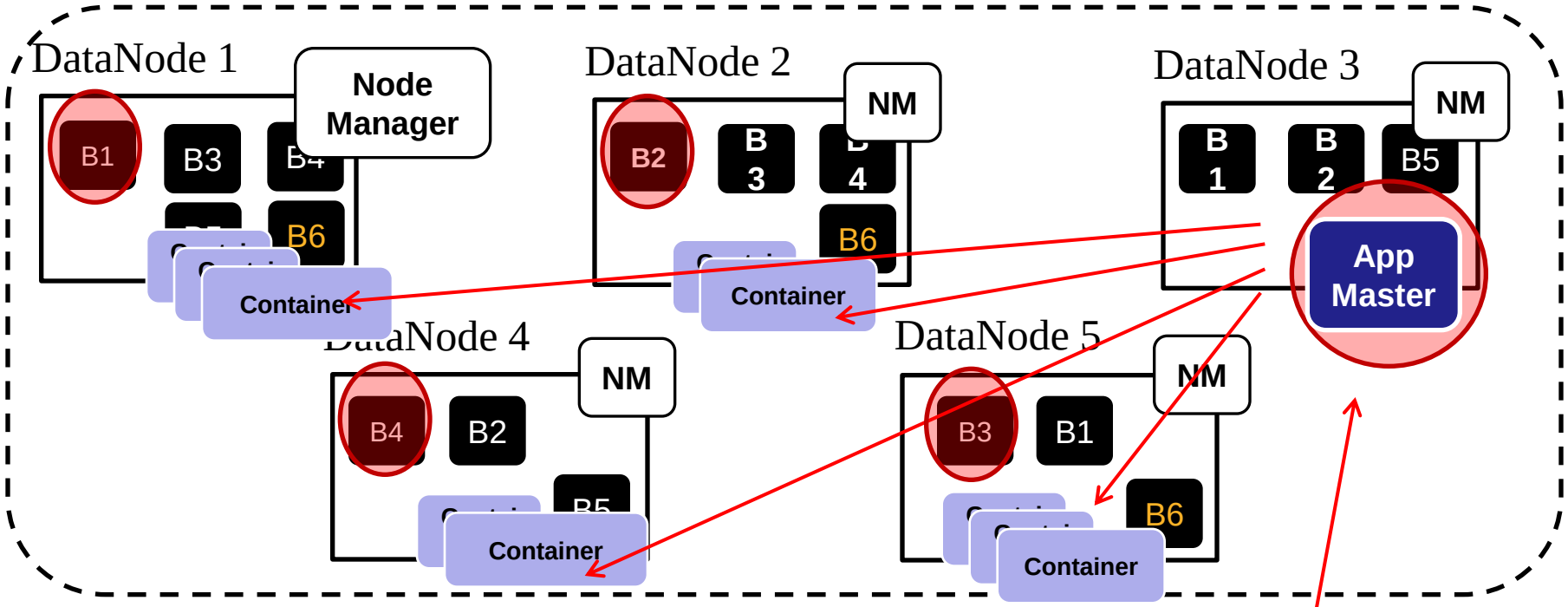
Resource Manager

Resource Management Scheduling

Client

Job #1
Store Sales

Hadoop Job Execution



NameNode

StoreSales.txt - B1, B2, B3, B4
Promos.csv - B5, B6

Resource Manager

Resource Management Scheduling

Client

Job #1 Store Sales

Job Execution – High Points

Resource Manager

- **Application Management and Scheduling (at least one, go HA)**

Node Managers

- **Worker Nodes (1 per DataNode)**

Application Masters

- **Controller for each YARN application (1 per Job)**

Containers

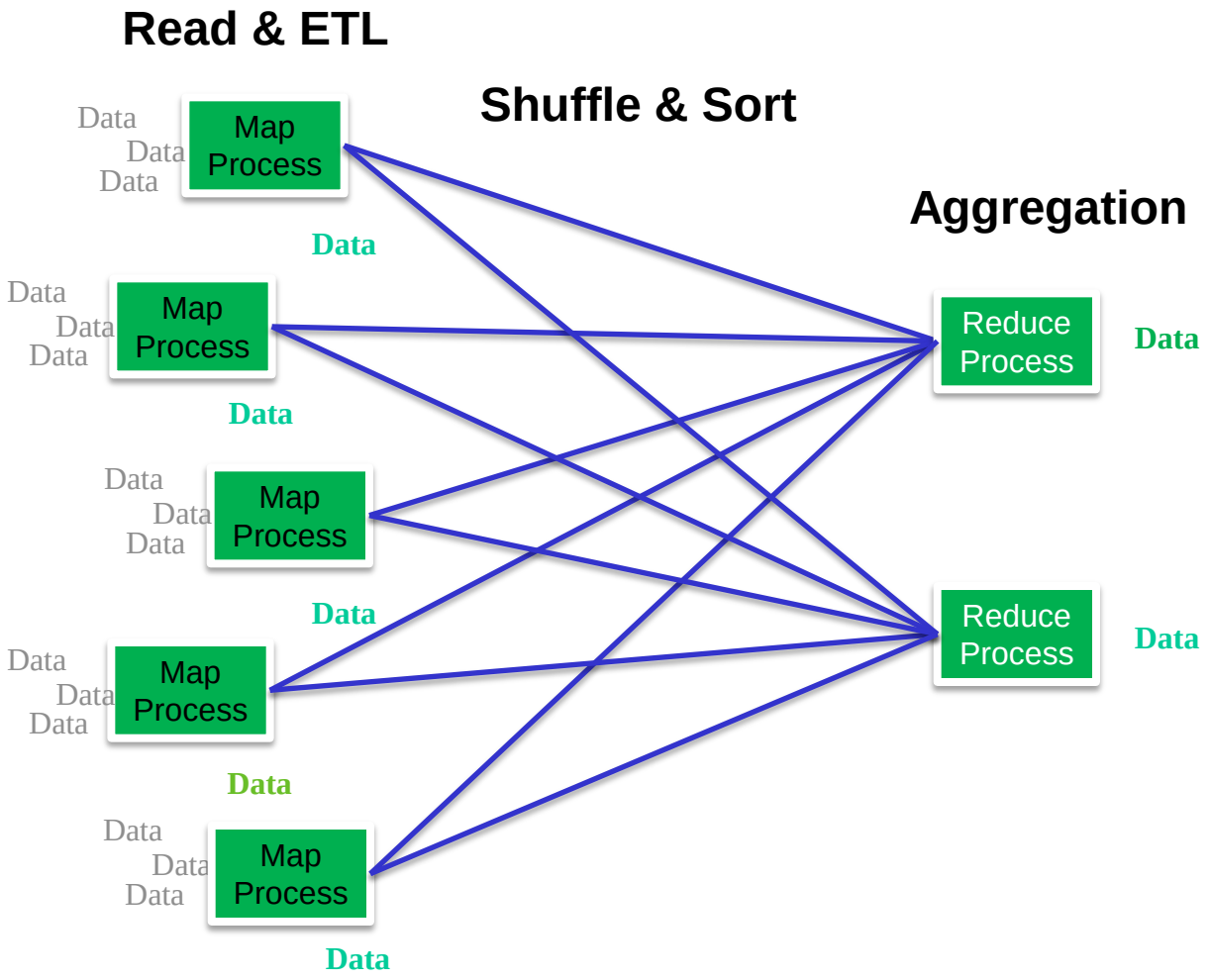
- **Generic Resource Unit (1 per Task)**

Tolerant of Task Failures

In fact, Task failures are expected

- Speculative Execution
- Task Resubmission

MapReduce breaks a large problem into sub-solutions



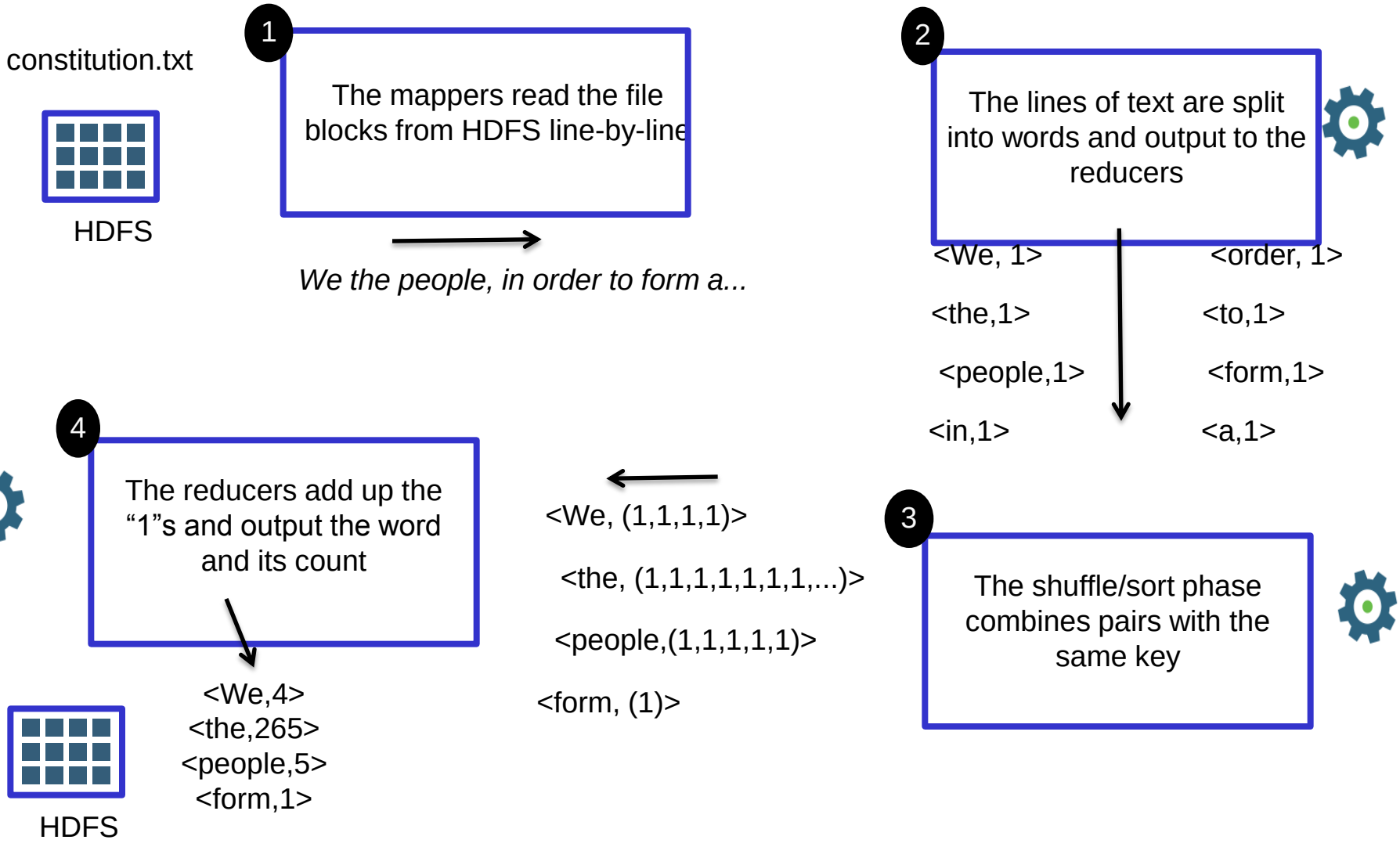
Map (SQL Analogy: Select – From - Where)

- Read data one Key-Value Pair at a time
- Apply computation
- Emit <key, value> pairs

Reduce (SQL Analogy: Group By - Having - Aggregation - Sort)

- Read Map Task results
- Apply computation
- Emit <key, value> pairs – usually 1 row/key

WordCount in MapReduce

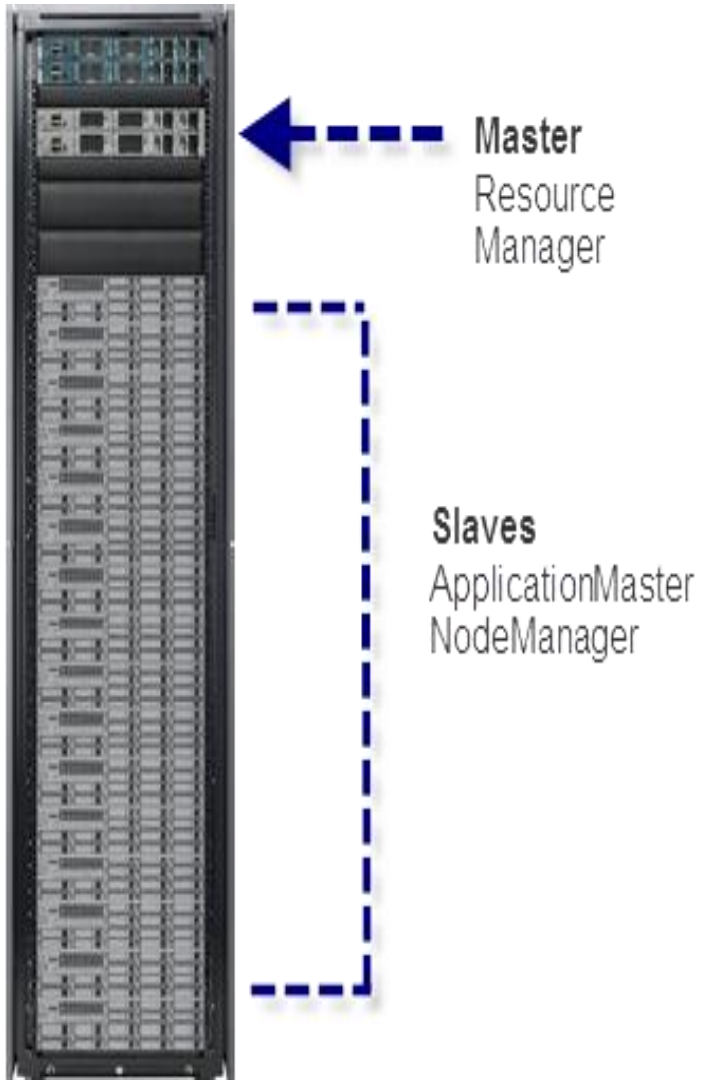


YARN

ResourceManager allocates cluster resources, starts ApplicationMaster

ApplicationMaster divides job into tasks, assigns task to NodeManagers

NodeManager runs on all slave nodes, starts and monitors task



BIG Improvements over MapReduce

Mixed Workloads

- **Batch, Interactive, and Real-Time processing**

Greater Task Parallelism

- **No longer successive Maps & Reduce loops**

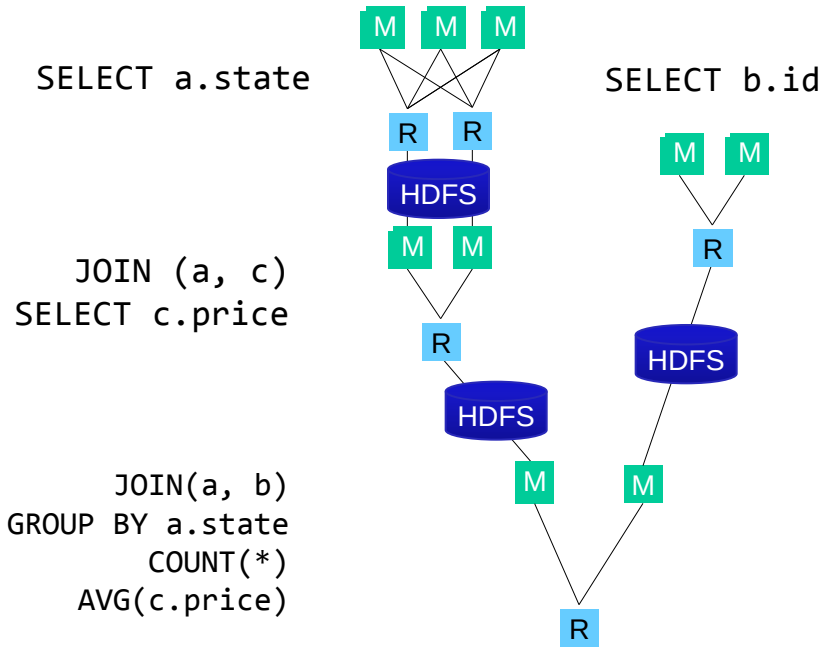
In-Memory & Caching Optimizations

- **No writes or spill to disk unless necessary**
  - Reduced process overhead (startup costs)
  - Data remains in-memory

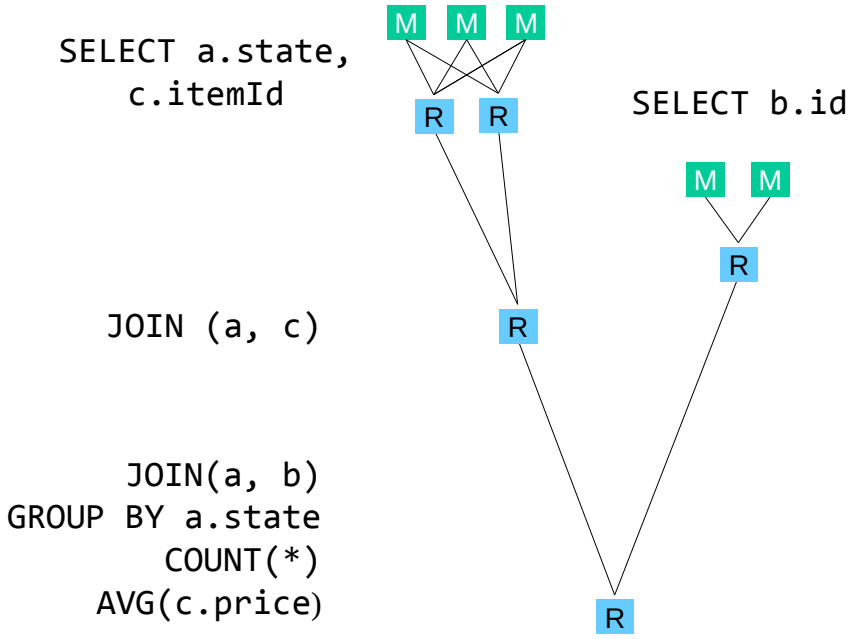
```
SELECT a.state, COUNT(*), AVG(c.price)
FROM a
   INNER JOIN b ON (a.id = b.id)
   INNER JOIN c ON (a.itemId = c.itemId)
GROUP BY a.state
```

Tez avoids unneeded writes to HDFS

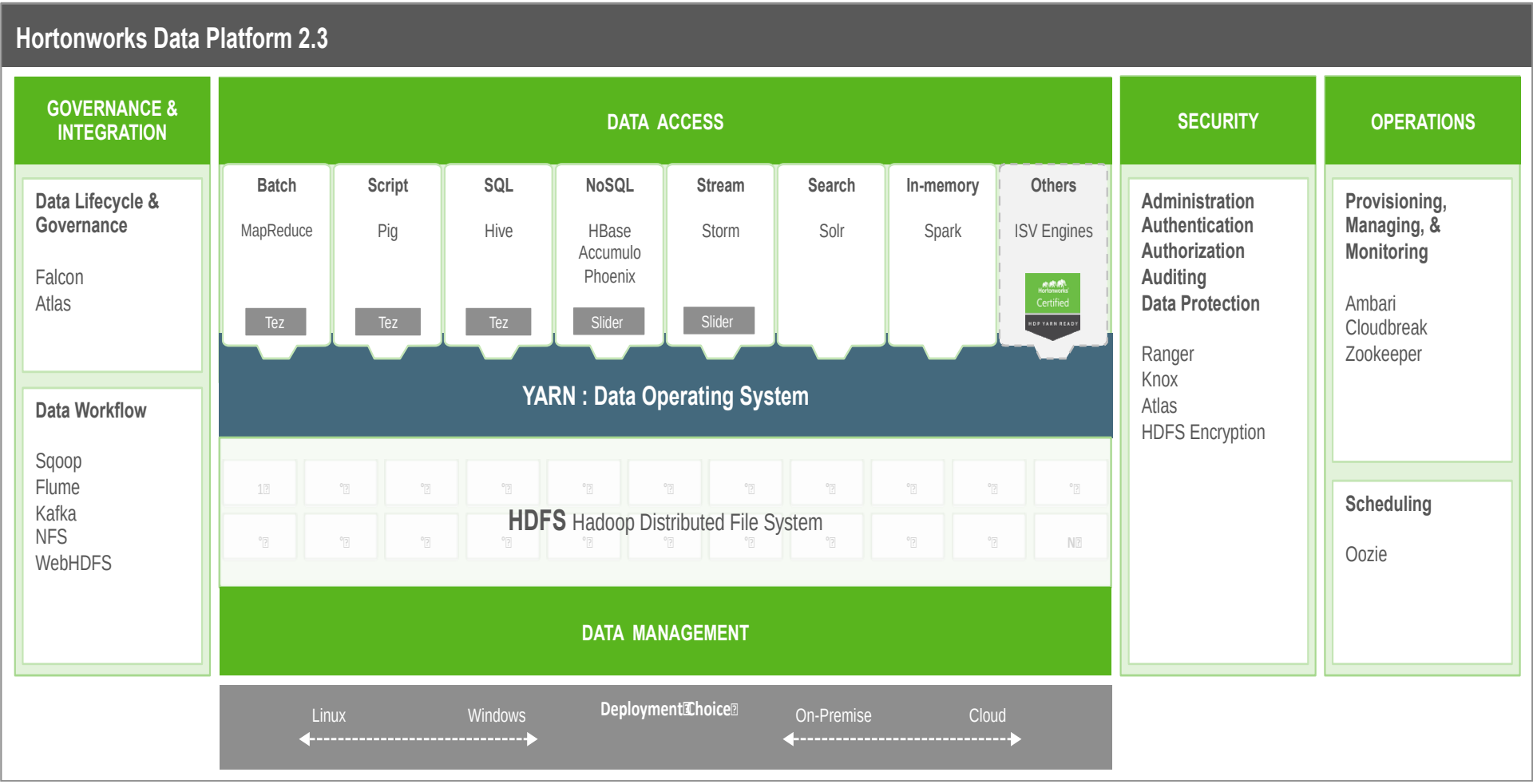
Hive – MapReduce



Hive – Tez



The Hadoop Ecosystem



The Hadoop Ecosystem

HDFS – The Hadoop Distributed File System.

Map/Reduce – A programming model for large datasets, Parallel & Distributed.

Pig – A data processing engine using scripting language. Uses M/R & Tez.

Hive – A data processing engine using SQL-like language. Uses M/R & Tez.

Hcatalog – Provides a relational view of HDFS data .

YARN – An OS-like layer that manages resources.

SPARK – Data analytics engine that uses in-memory data stores.

Tez – Workload optimizer for HDFS.

Kafka – Message Broker (Publish/Subscribe -- similar to MQSeries, MSMQ).

Hbase – A wide-column store NoSQL database engine.

Flume – Ingestion of streaming data (e.g. logs).

Sqoop – Ingestion of data from relational sources.

Solr – full text search

Oozie – Job scheduler.

Ambari – Sysadmin console

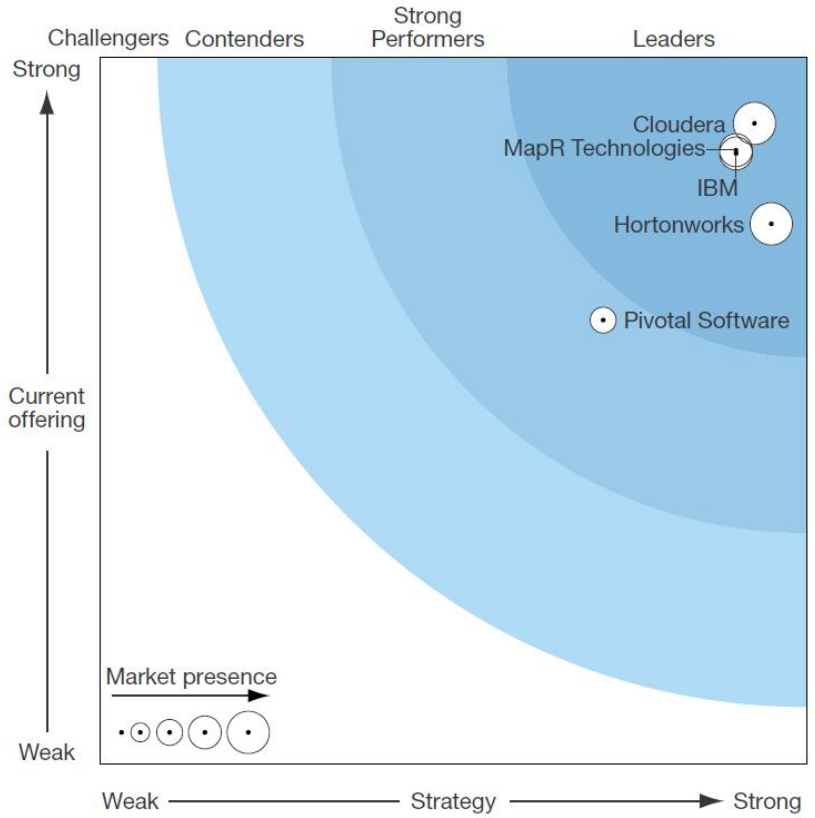
Ranger – Security manager

https://hadoopecosystemtable.github.io/

Providers

Five top commercial Hadoop distribution vendors:

- Cloudera
- Hortonworks
- IBM
- MapR Technologies
- Pivotal Software



https://www.cloudera.com/content/dam/www/static/documents/analyst-reports/forrester-wave-big-data-hadoop-distributions.pdf