Introduction

CSCI 6830

BIG DATA ANALYTICS WITH HADOOP AND R

Who's Generating Big Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

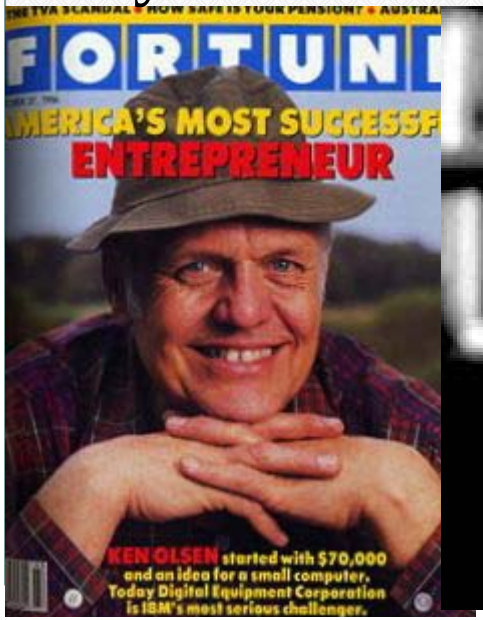
Mobile devices
(tracking all objects all the time)

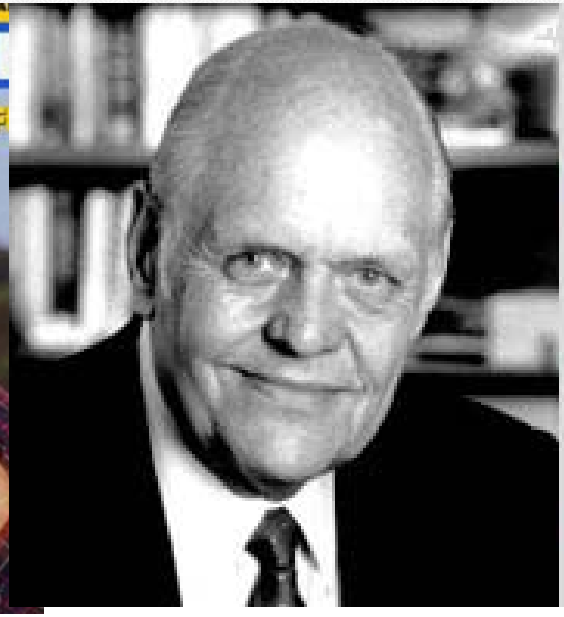
Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

How much data?

- Google processes 20 PB a day
- Wayback Machine has 3 PB + 100 TB/month
- Facebook has 2.5 PB of user data + 15 TB/day
- eBay has 6.5 PB of user data + 50 TB/day
- CERN's Large Hydron Collider (LHC) generates 15 PB a year



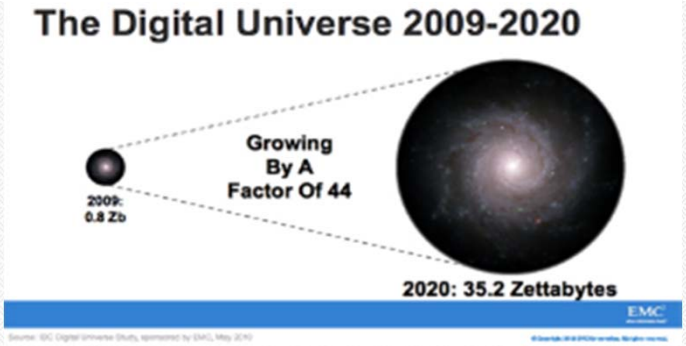


"There is no reason for any individual to have a computer in his home."

Ken Olsen

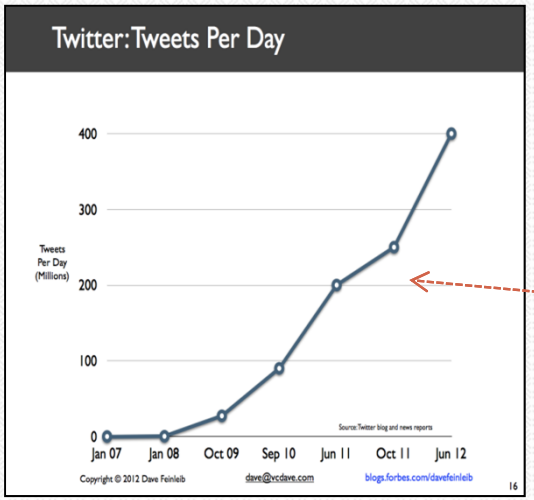
Characteristics of Big Data:
1-Scale (Volume)

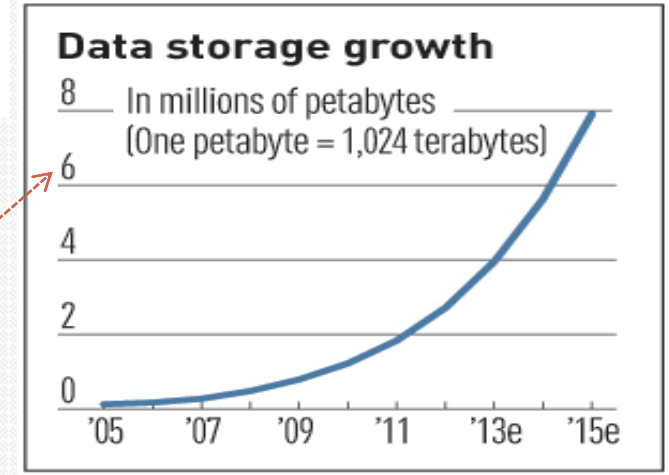
- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially



terabytes | petabytes | exabytes | zettabytes

the amount of data stored by the average company today

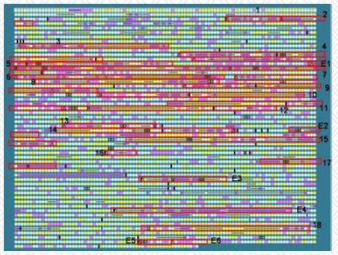


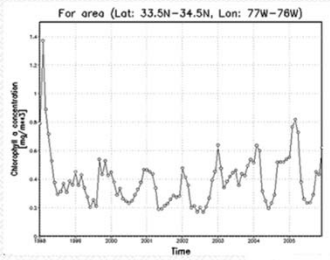


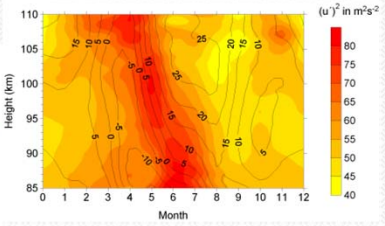
Exponential increase in collected/generated data

Characteristics of Big Data:
2-Complexity (Varity)

- 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







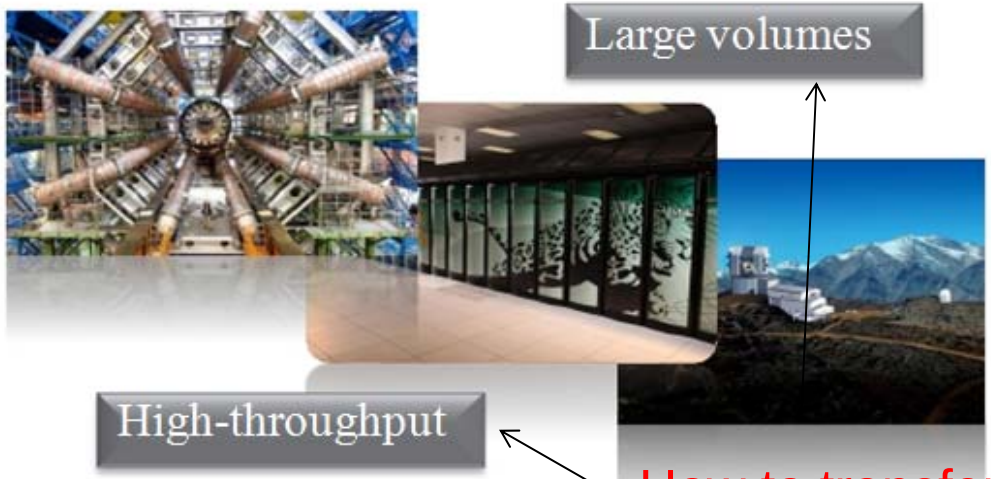
To extract knowledge➔ all these types of data need to linked together

Characteristics of Big Data:
3-Speed (Velocity)

- Data is begin 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

Challenges

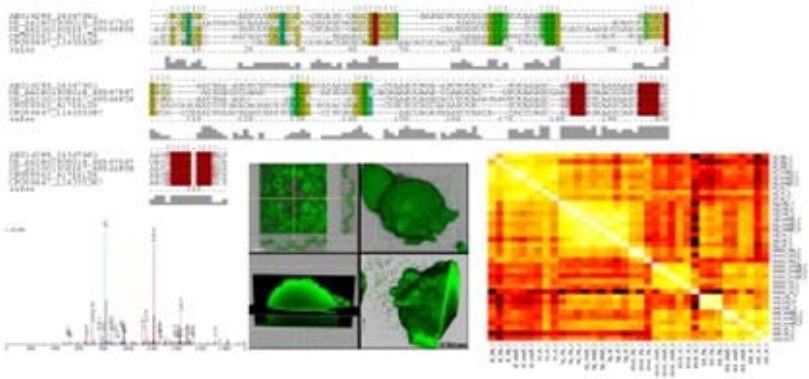


Large volumes

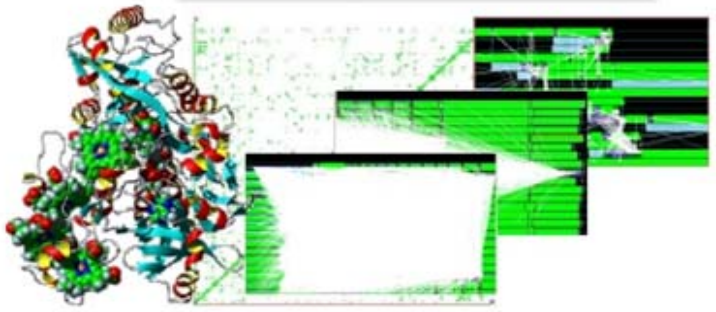
High-throughput

How to transfer Big Data?

Relating and Linking

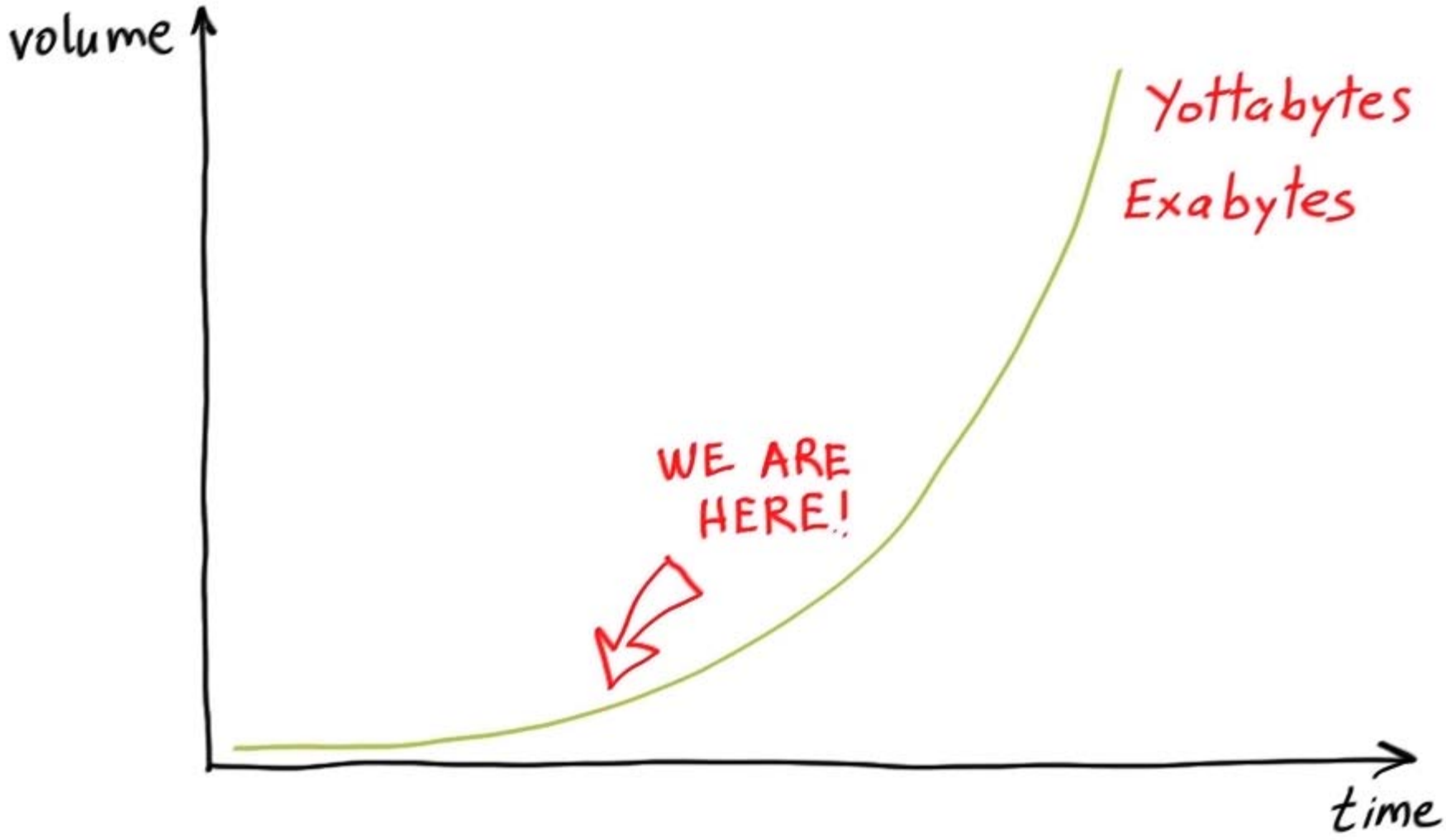


Heterogeneity

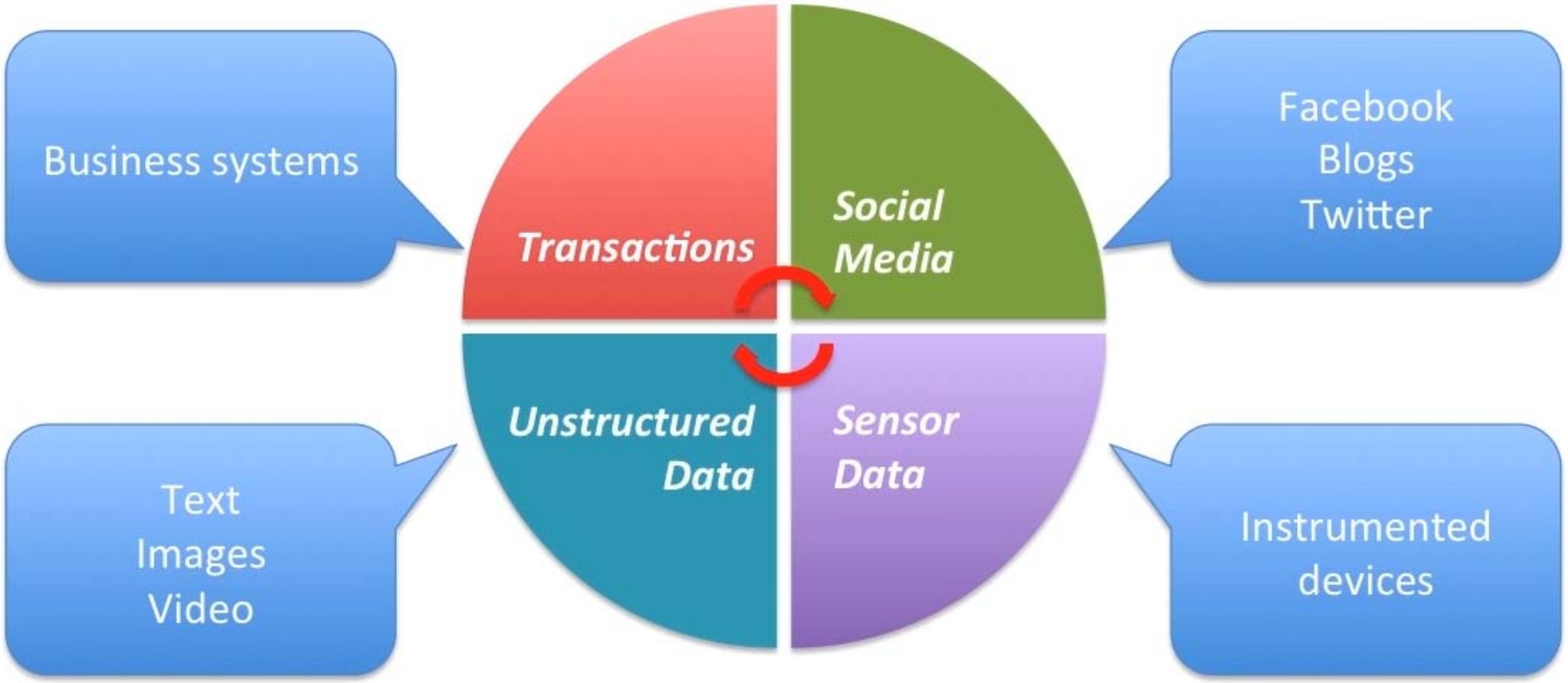


Complexity

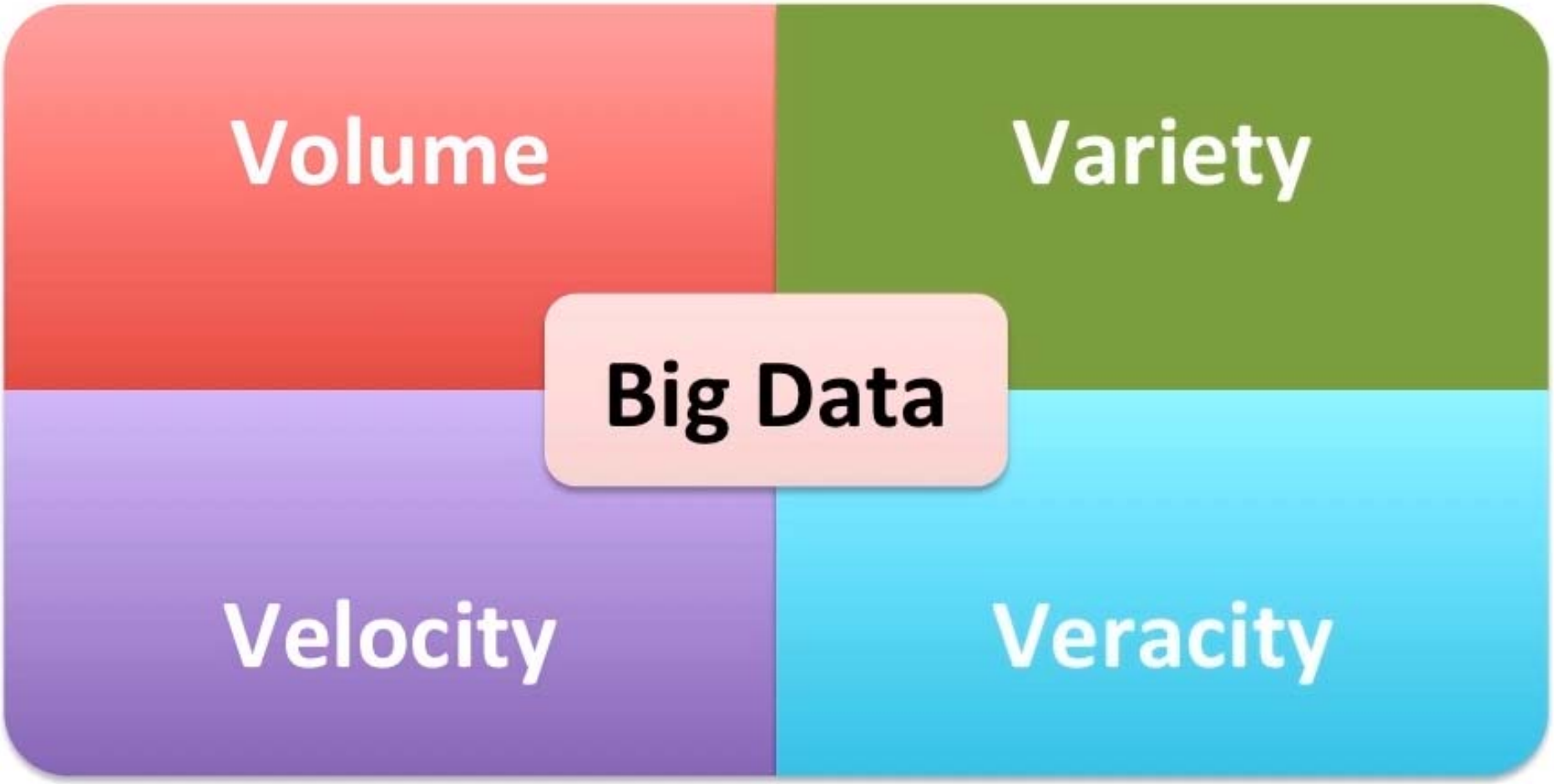
Big Data



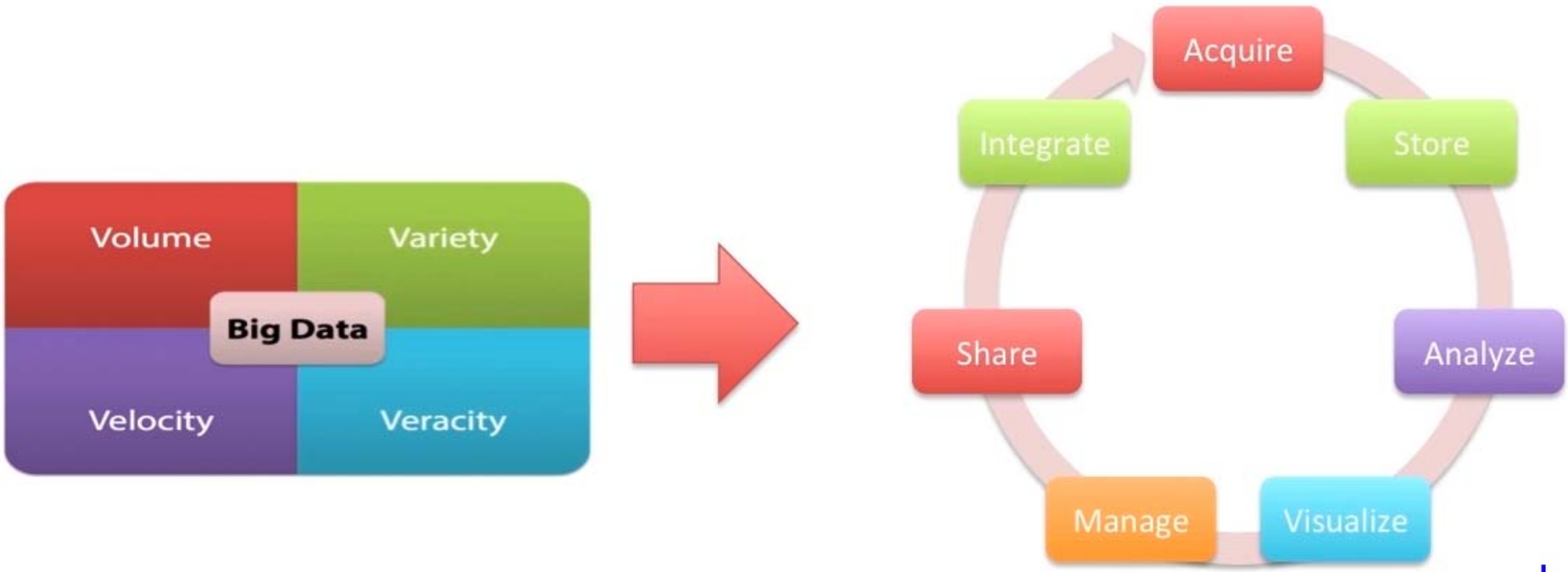
Big Data Sources



Big Data Characteristics



Big Data Workflow



- Process workflow timely and cost-effectively

Big Data-Related Problems

- More data than traditional system can handle
  - Not always extremely high volumes
- Unstructured data
  - Data does not fit in conventional storage
- Data arrives quickly
  - May have very narrow usefulness window

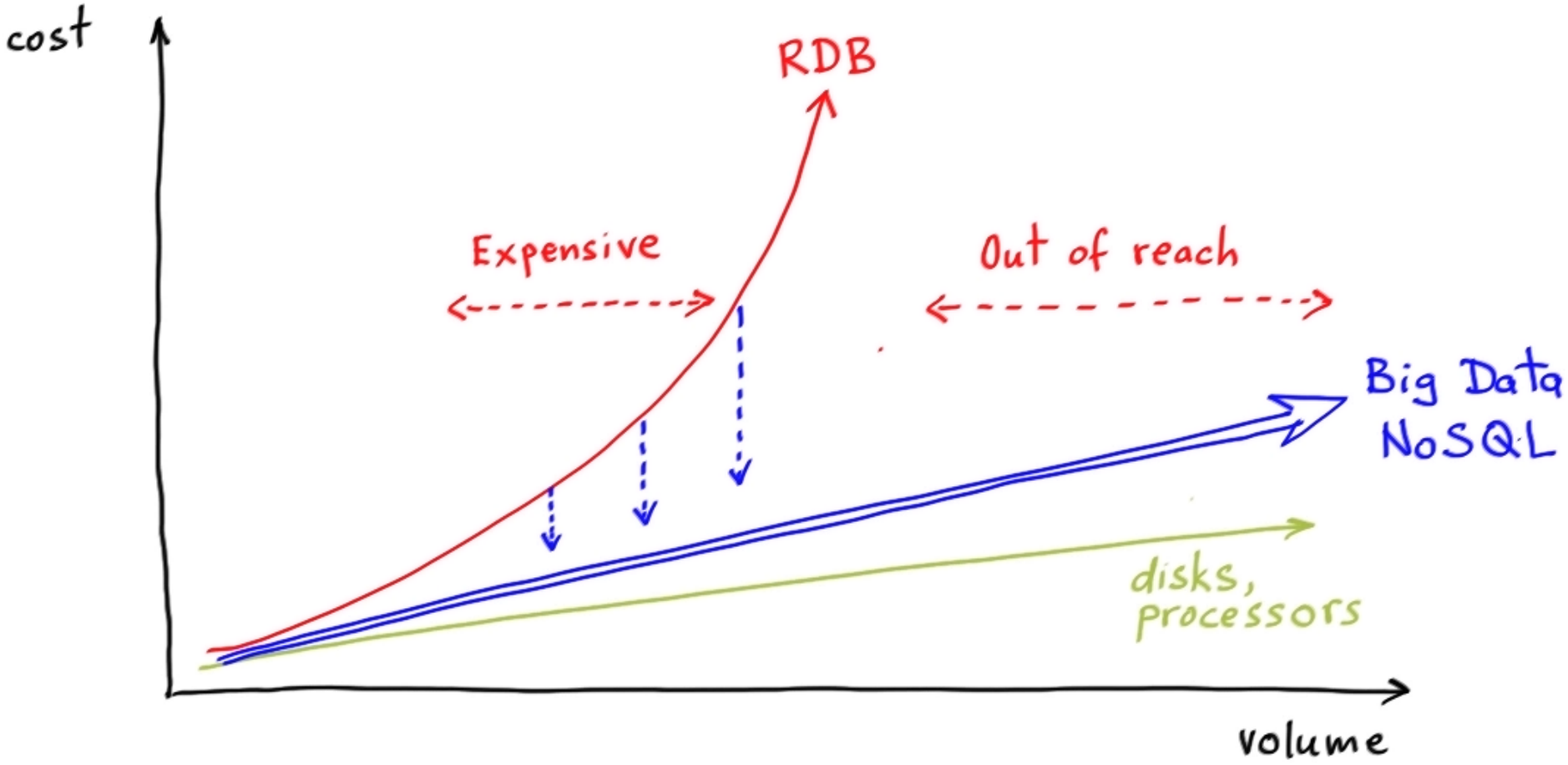
More Big Data-Related Problems

- So much data – not clear what to analyze
- Integrate new data with conventional systems
- Lack of experts to process, manage, and analyze data
- Organizational silos

Solving Big Data Problems

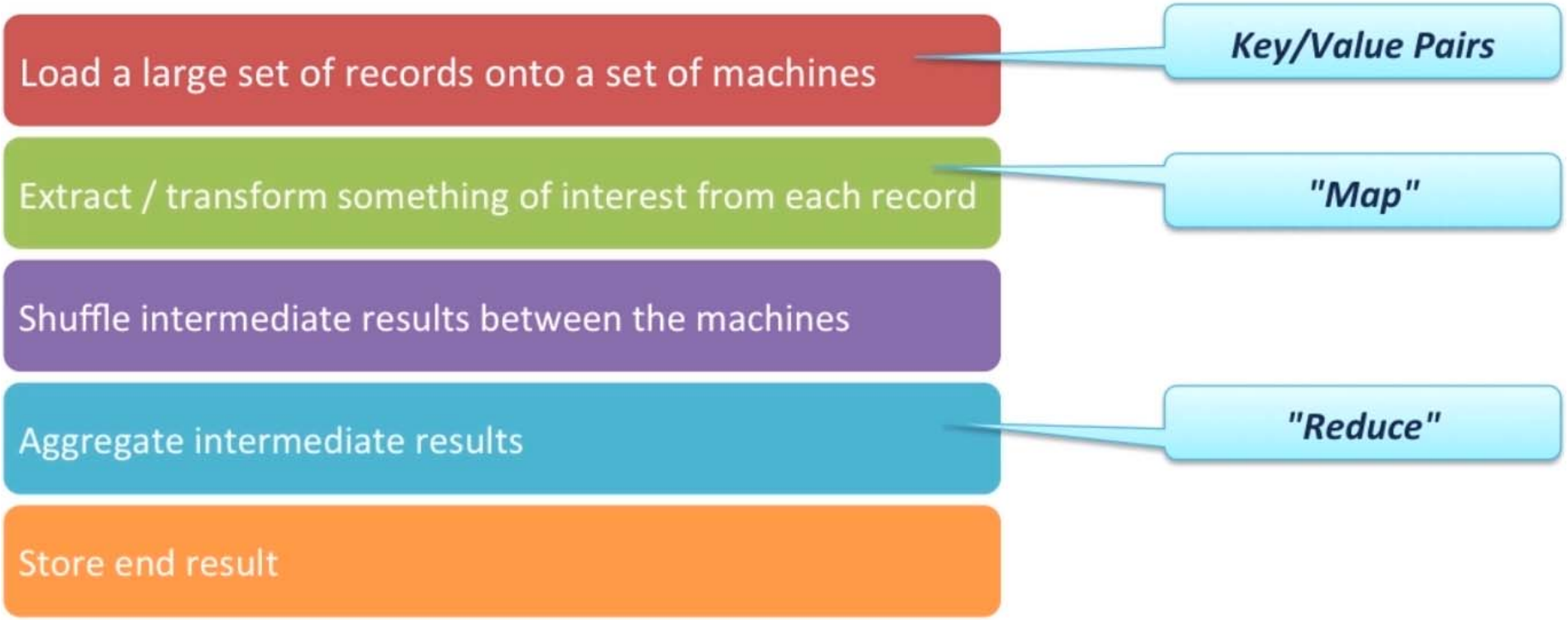
- New processing approaches
  - MapReduce algorithm
  - Other algorithms
- New data technologies
  - Hadoop and Hadoop family
  - NoSQL databases
  - Stream processing
- Tools must be combined, no single solution

Challenges of Relational Databases

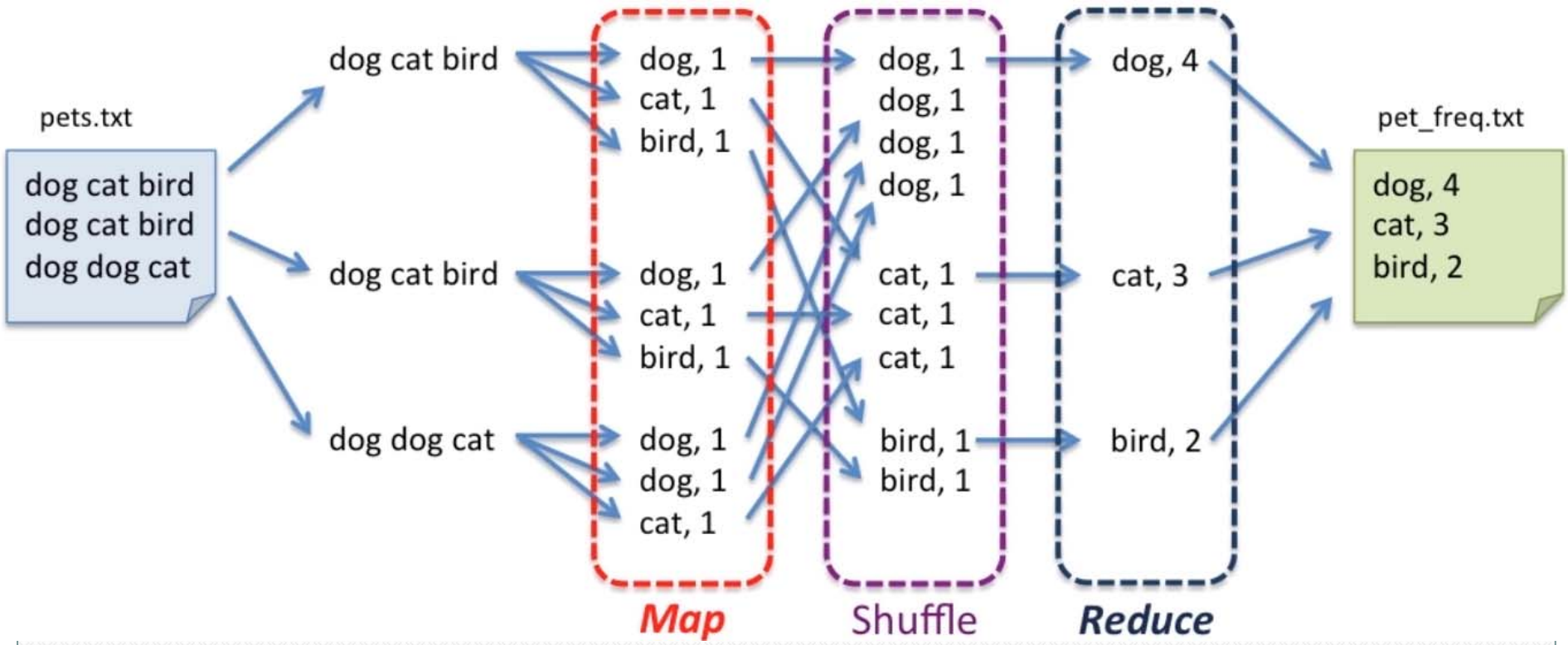


MapReduce and Hadoop

- Process large volumes of data with many machines



MapReduce Algorithm



MapReduce—**Map Step**

- "Map" Step:
  - Input split into pieces
  - Worker nodes process individual pieces in parallel
  - Each worker node stores its result in its local file system
  - Reducer accesses files produced by workers

MapReduce—**Reduce Step**

- "Reduce" step:
  - Data is aggregated by worker nodes
    - This is called "reduced" after the map steps
  - Multiple reduce tasks can be executed in parallel

Separation of Work

Programmers

- Map
- Reduce

Framework

- Deals with fault tolerance
- Assign workers to map and reduce tasks
- Moves processes to data
- Shuffles and sorts intermediate data
- Deals with errors

Apache Hadoop

- Open-source framework for processing large amounts of data, spread across multiple machines
- Designed to work on clusters of machines
  - Cheap and unreliable clusters
- Inspired by Google technologies
- Implemented in Java
- The system is designed to scale
- Originally, employed by companies with big data from the Web
- Great solution for data of regular scales (less than Petabyte)

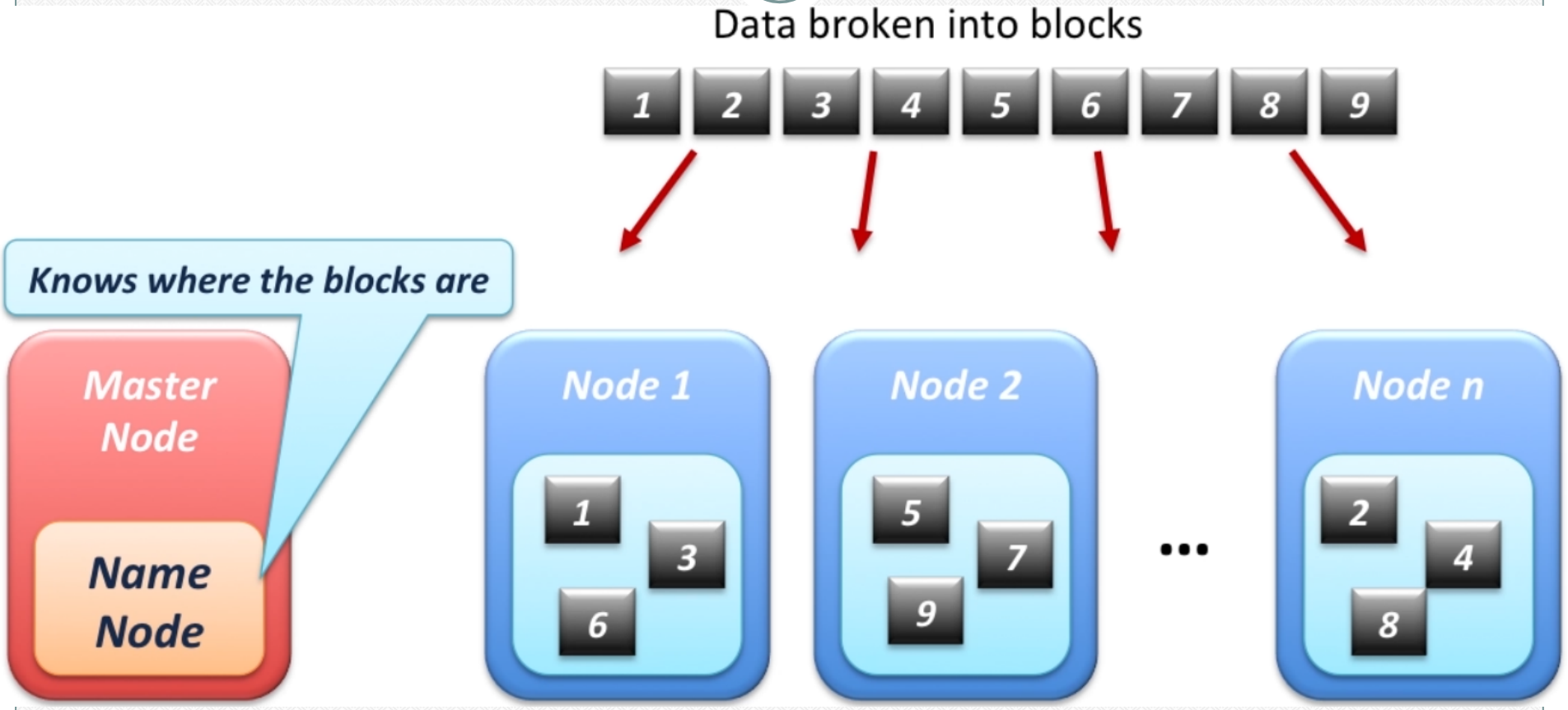
Two key aspects of Hadoop

- MapReduce framework
  - How Hadoop understands and assigns work to the nodes
- HDFS—Hadoop Distributed File System
  - Where Hadoop stores data
  - Data stored redundantly
  - HDFS spans all the nodes in Hadoop cluster
  - Links together local nodes to make one big system
  - System can be self-healing
  - For a regular user HDFS looks as an ordinary file system

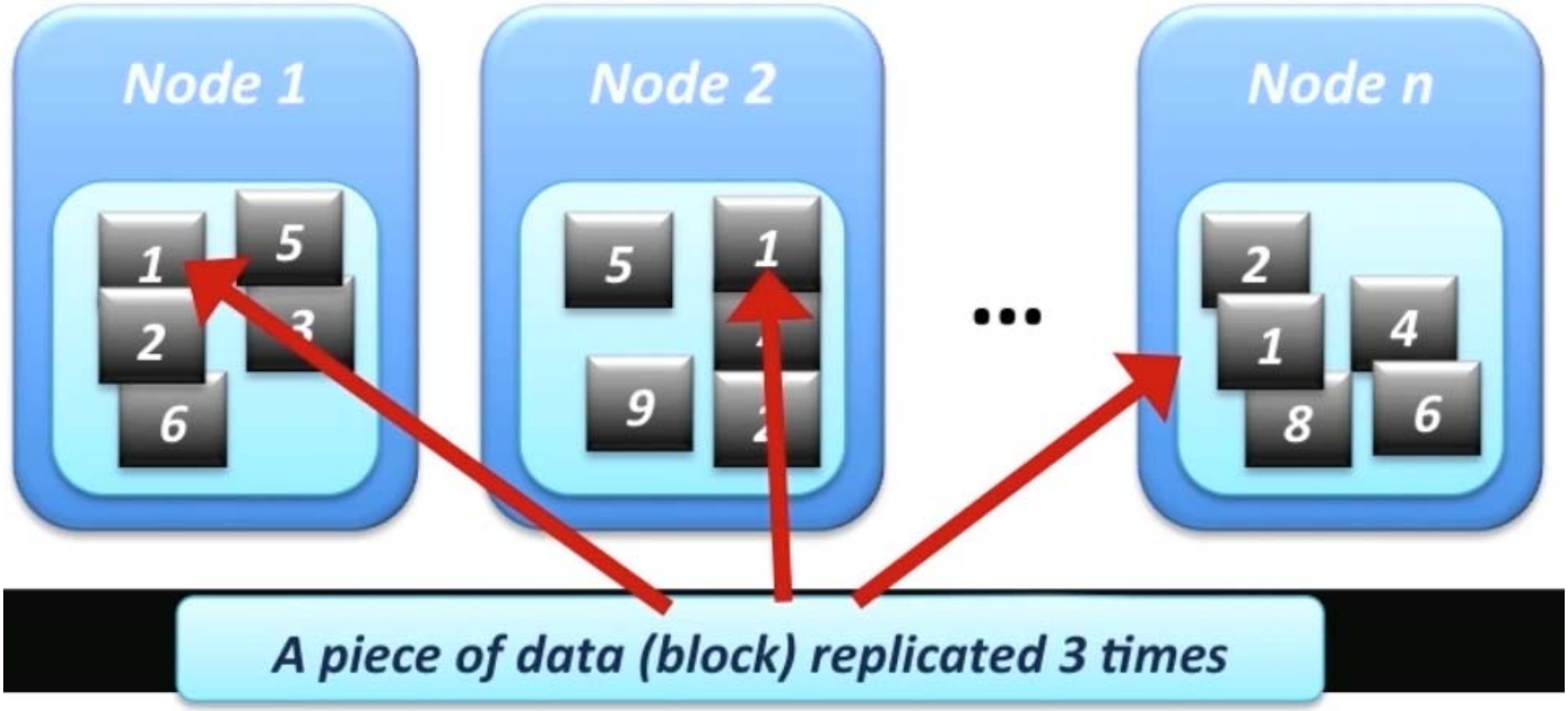
HDFS for MapReduce

- Hadoop Distributed File System
- Data must be stored on multiple machines
- Should allow storing any kind of data
  - Data does not have to be fit into a table as a relational database
- A machine in a cluster can fail!
- Data should not be lost
- **Idea**: distributed reliable knowledge
- **Solution**: Hadoop Distributed File System

Hadoop Distributed File System



HDFS Replication



HDFS

- Data is:
  1. Split into blocks
  2. Distributed across the machines in the cluster
  3. Each block is replicated 3 times

- Machines in the cluster are cheap and unreliable
- HDFS resides on top of a native file system
- Block size is typically 64 or 128 MB
- Follows the idea of the Google File System (GFS)

HDFS Features

- Smaller number of large files
  - Files typically > 100MB
- Ideal applications read the data from the beginning to the end
  - Minimize the cost of *seek*
- Files are typically not updated
  - No random access!
- Default replication: 3

Hadoop Infrastructure: **Name Node**

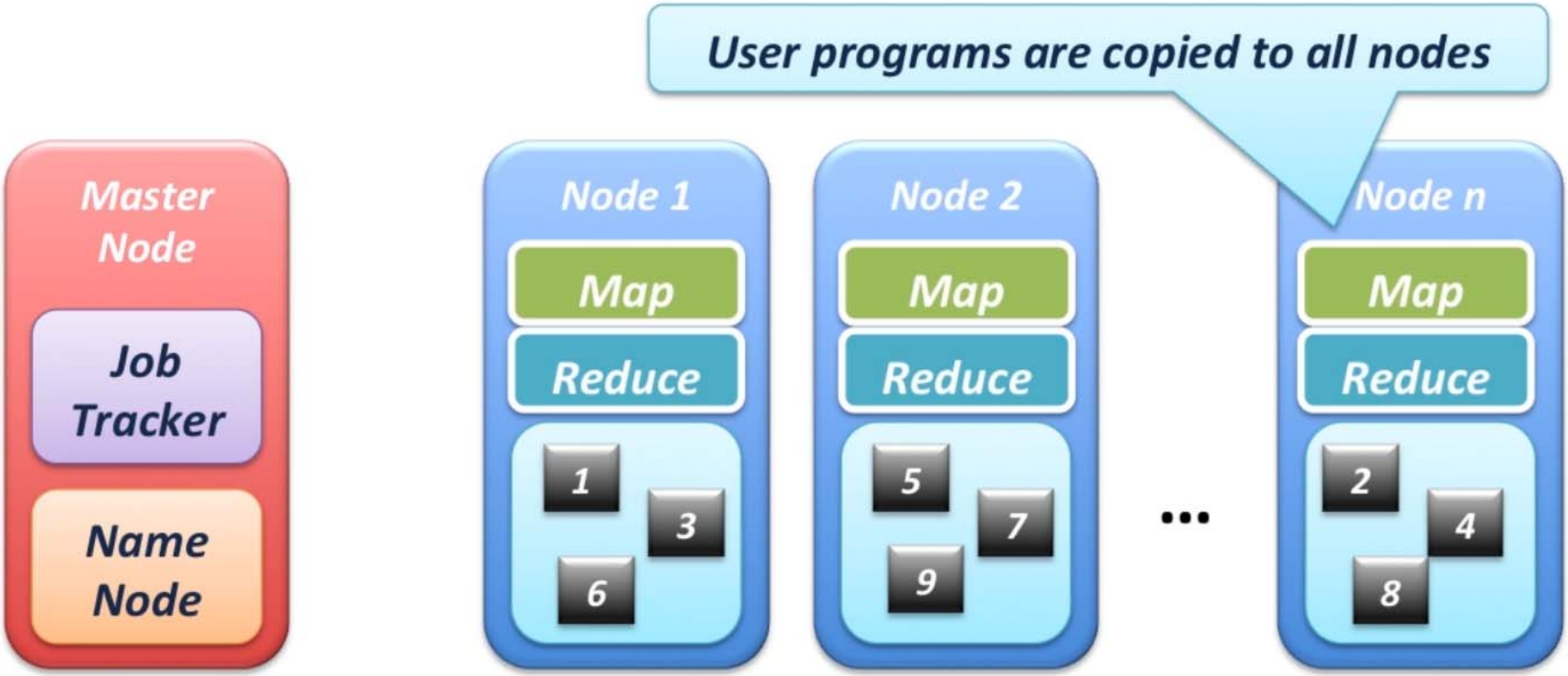
- Namenode is installed on a more reliable machine

- Manages the metadata for the HDFS
- Must remain accessible
  - Install on a more reliable machine than the data notes
- Metadata is held in RAM
  - Make sure the NameNode machine has a lot of RAM!
- Secondary NameNode
  - It is not a backup!
  - It handles some housekeeping tasks

Hadoop Infrastructure: **Data Node**

- Stores data in standard files
- NameNode used to access the data in the HDFS cluster
  - NameNode knows which blocks are needed to assemble a file
  - Programs read data from DataNode directly
- Files are split into 64 or 128MB blocks
  - Blocks make for easier management
- Map and Reduce jobs run on data nodes
  - We send the programs to the data, instead of the other way around!

Map and Reduce Tasks on Nodes



Hadoop Management Tools

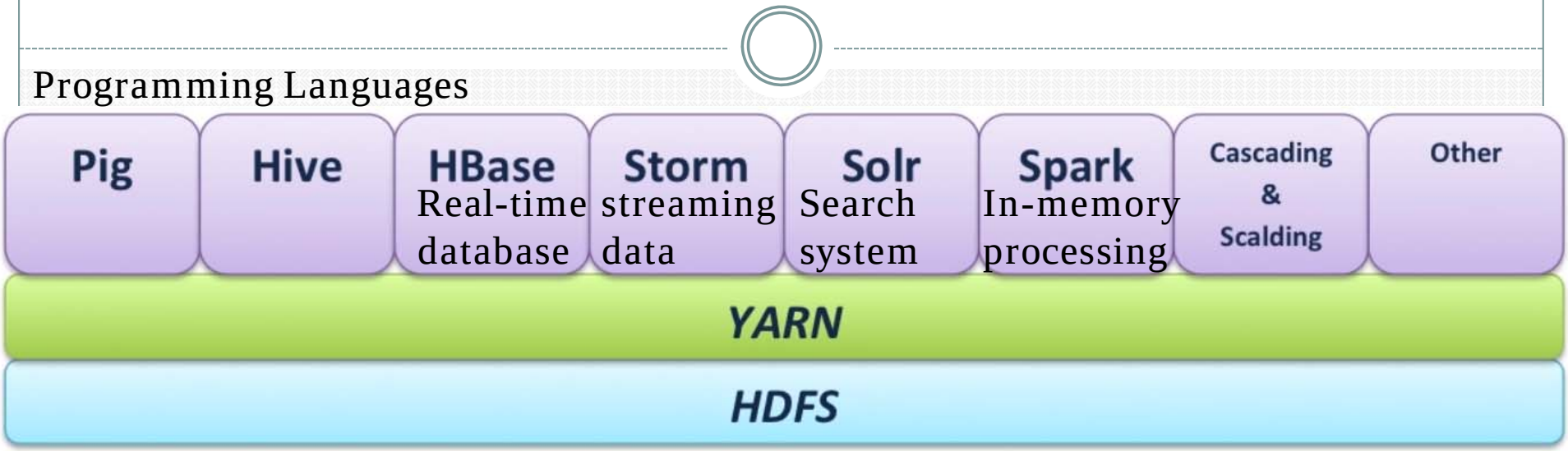
- Graphically manage cluster, jobs, HDFS
- Sample administration tasks
  - Start/Stop servers
  - Add/Remove servers
  - Server status details (log)

YARN

- Yet Another Resource Negotiator

- Hadoop MapReduce V1 issues:
  - Multi-tenancy
    - MapReduce V1 has a very simple approach to assigning tasks to nodes
    - We wish to manage multiple jobs on a cluster
  - Difficult to scale beyond 4,000 nodes
    - Cascading failures, network flooding
- YARN is an internal reorganization in Hadoop V2
  - API compatible with Hadoop V1

Resource Management with Yarn



- Yarn: A common resource management for applications
  - Scalability
  - Improved cluster utilization
  - Workloads other than MapReduce