



Better Methods. Better Outcomes.

# Webinar Series

## TMIP VISION

TMIP provides technical support and promotes knowledge and information exchange in the transportation planning and modeling community.



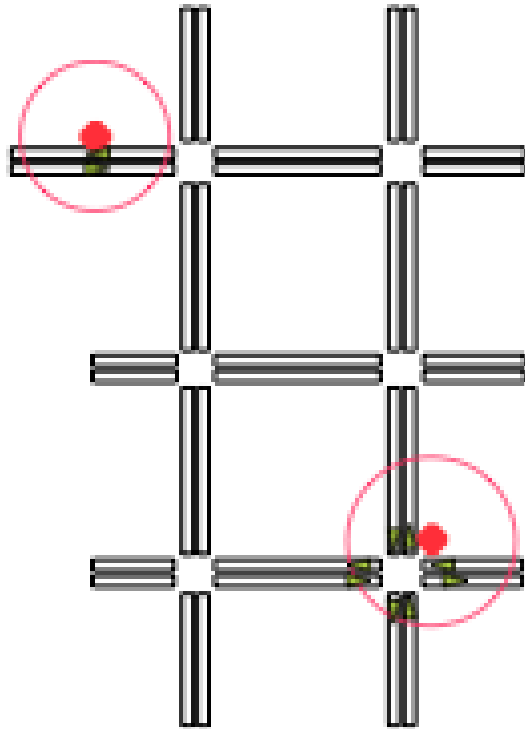
U.S. Department of Transportation  
**Federal Highway Administration**

# Today's Goals

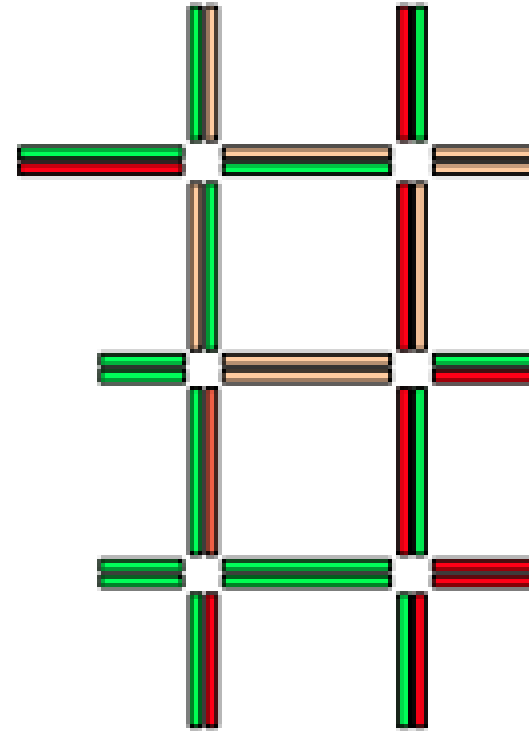
## To Consider:

- Parallel Processing
  - More on Hadoop
- Data Management
  - Distributed File Systems
  - Storage (SQL/NoSQL)
- Algorithms
  - Processing algorithms
  - Prediction algorithms (ML)

# Algorithm Teaser / Crowd Pleaser



(a) GPS points and candidate projections



(d) Traffic estimation algorithm

Algorithms get you from (a) to (d) ... Big Data infrastructure does it at scale

# Webinar Roadmap

Big Data definitions

Infrastructure

Metal

Management

Models

(part 1)

(part 2)

Algorithms

Data Processing

Machine Learning

Implications

Privacy

Smart Cities

Technology Tradeoffs

Session 2

Big Data Infrastructure

# **MODELS: RESOURCE PROVISIONING**

# Big Data Infrastructure: Hadoop

Hadoop is

- A data processing programming model
- A resource management framework
- A distributed file system

Hadoop is

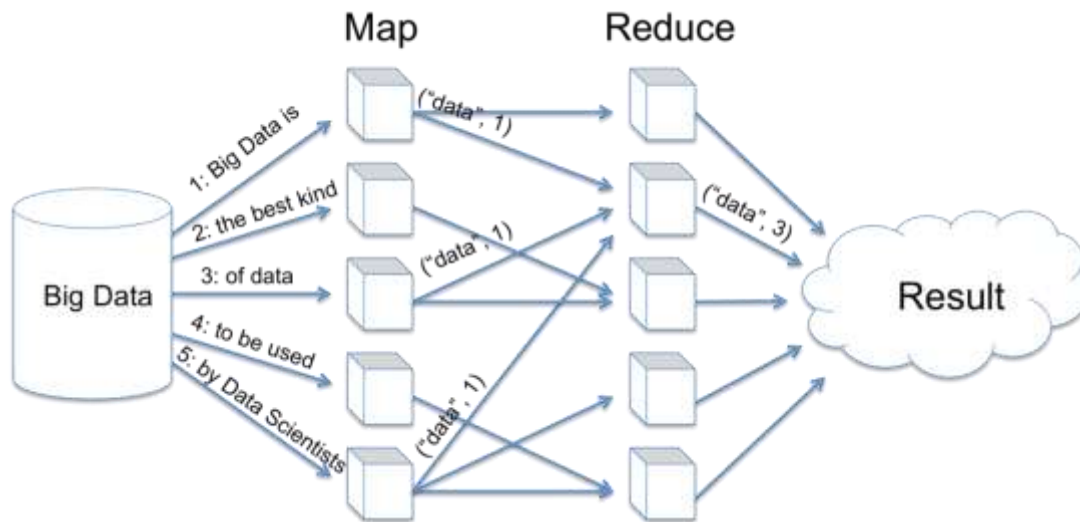
- A popular, open source parallel processing framework
- An implementation of the MapReduce algorithm

It

# Data Processing Model and Resource Management

## MapReduce\* is

- A programming model for parallel data processing
- A cluster resource management framework

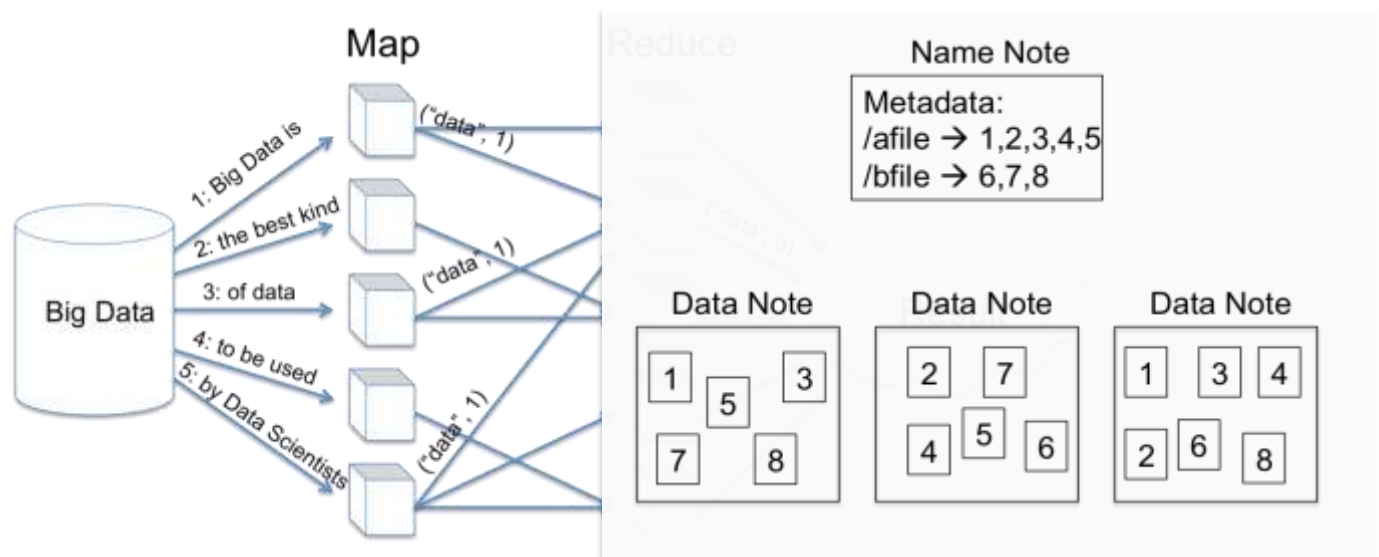


\* Hadoop 1.0

# Distributed File System

HDFS is

- A redundant, reliable storage framework
- HBase is a key-value store built on HDFS

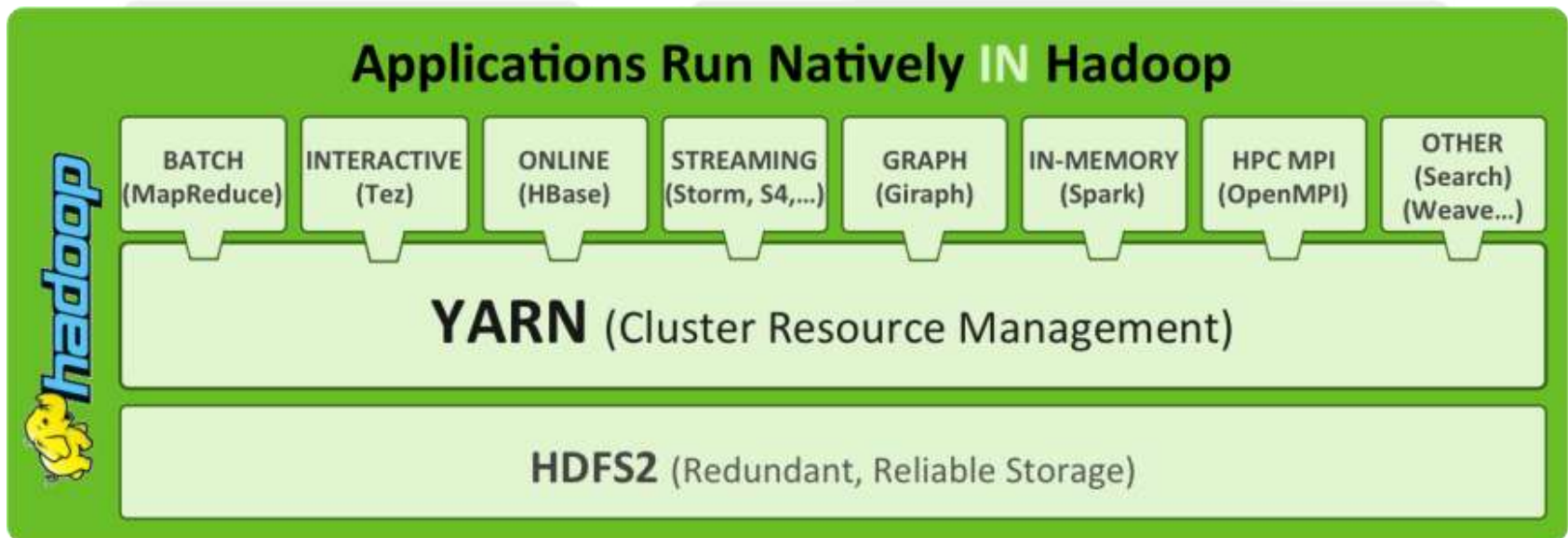




# Resource Management Framework

## Hadoop 2.0

- Released in October 2013
- MapReduce was split in two
  - MapReduce – A parallel data processing model
  - YARN – A cluster resource management system



# What is Hadoop Good For?

Hadoop is

- **MapReduce**: A data processing programming model
- **HDFS**: A distributed file system
- **YARN**: A resource management framework

**Most Hadoop Sys Admins (e.g., Data Scientists)**

- **Will need to be familiar with YARN and alternatives such as MESOS**

# The MapReduce Programming Model

## MapReduce is...

- A programming model for parallel data processing
- Several Variants: Map Only, Single Reducer, ...
- MapReduce can be useful even if data is not BIG
  - Example: Processing Prime Numbers

# Summing Prime Numbers

Sum the First 100 Primes Between 1-100

- Sum: 1,060
- Execution Time (Sequential): 0.050 seconds

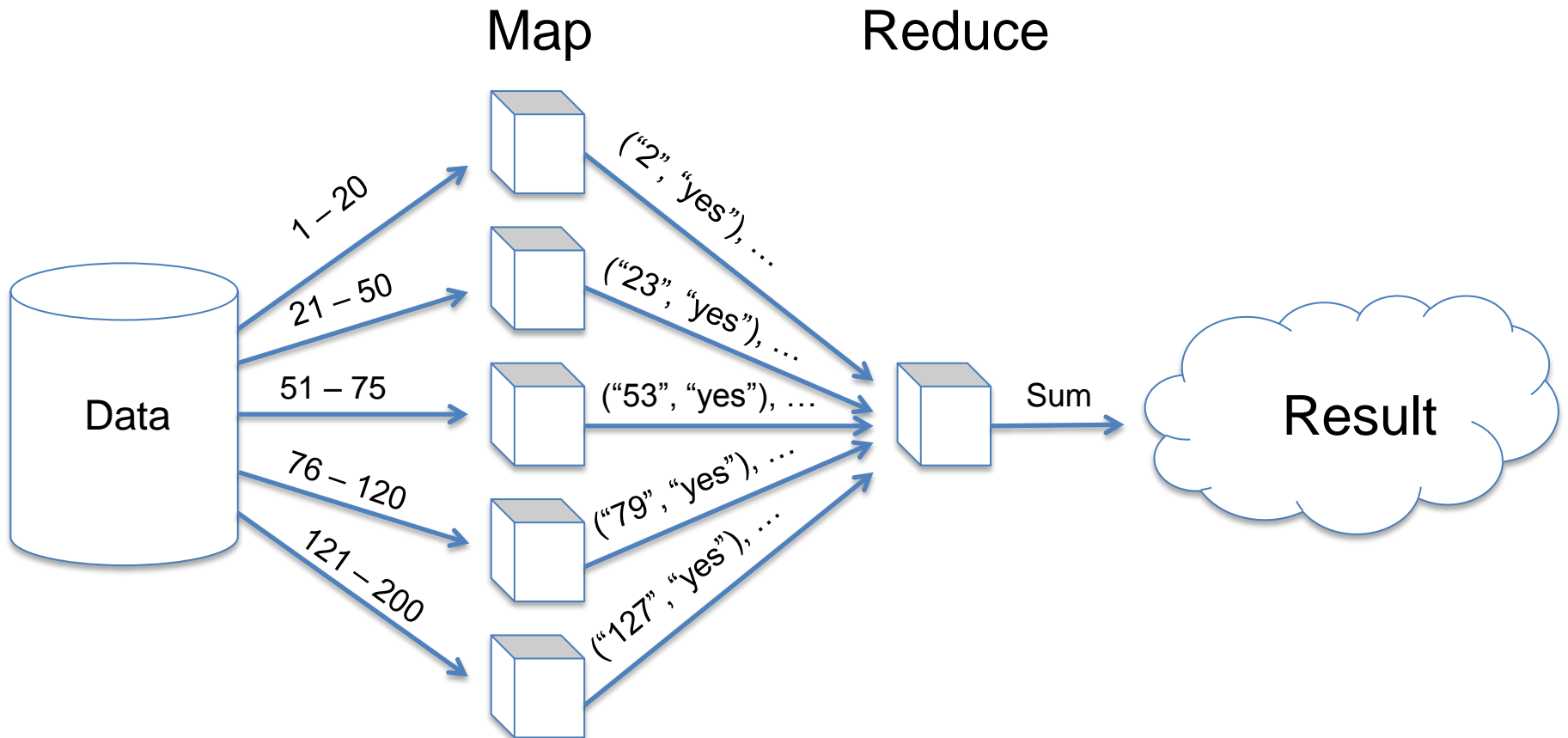
Sum the First 100 Primes Between 29,901-30,000

- Sum: 209,643
- Execution Time (Sequential): 0.111 seconds

Sum the First 100 Primes Between 299,999,901-300,000,000

- Sum: 1,199,999,796
- Execution Time (Sequential): 406 seconds

# MapReduce for Summing Prime Numbers



# Is MapReduce Good Enough?

## MapReduce Is A Batch Processing Model

- Batch:
  - Get All Data → Process Data → Stop

## What If The Data Processing Needs To Be

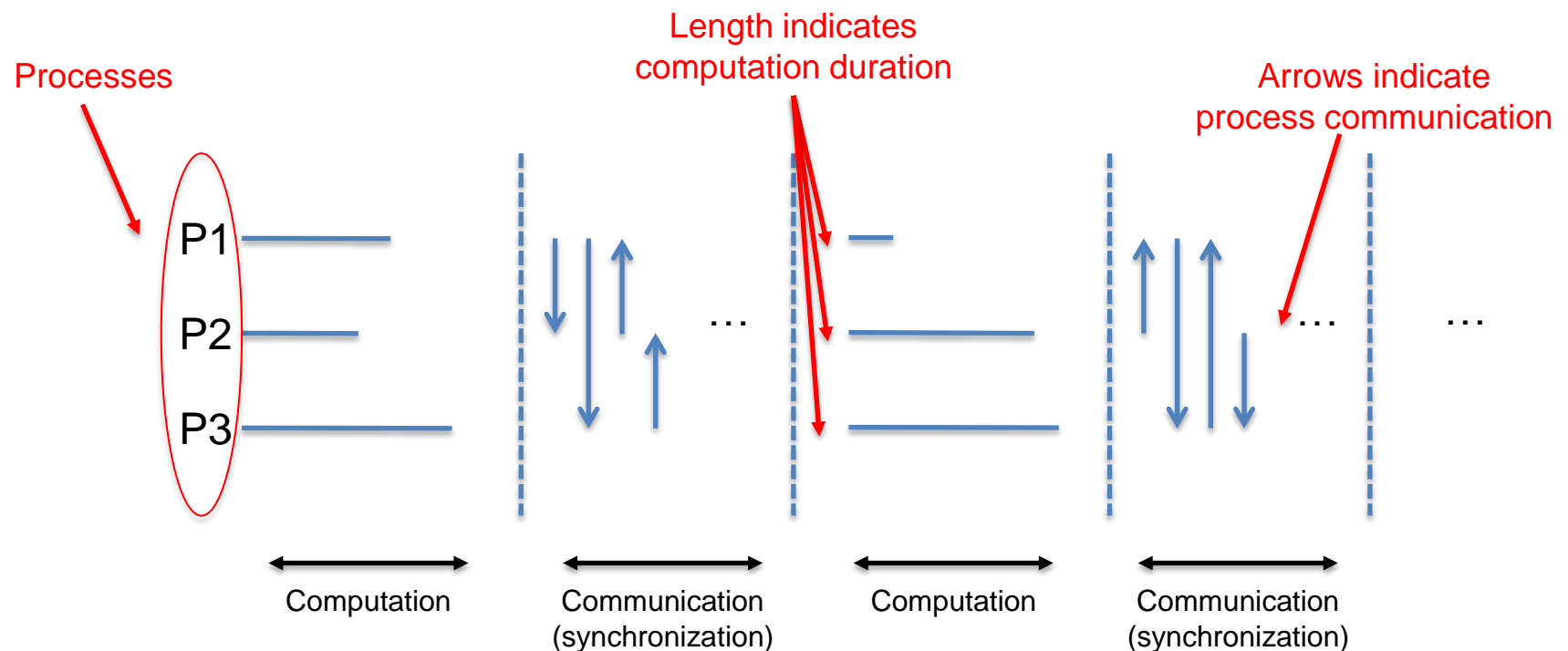
- Iterative
  - Get Data → Process Data → Repeat
- Streaming (Continual)
  - Get Data → Process Data → Get More Data → Repeat Forever

# Iterative Model:

## Bulk Synchronous Processing (BSP)

BSP Model - Created by Leslie Valiant (1980s)

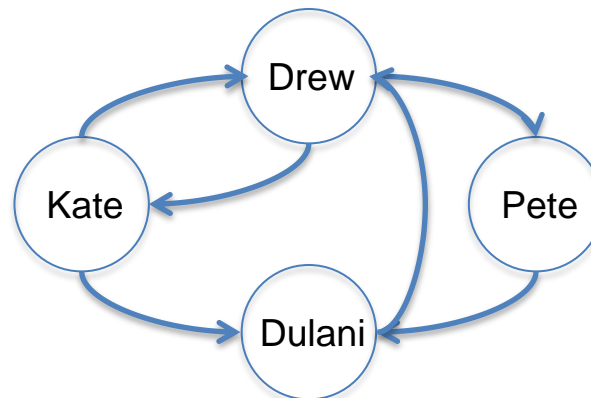
- Parallel processing via synchronized “supersteps”



# Iterative Model: When Is It Useful?

## Example: Graph Processing

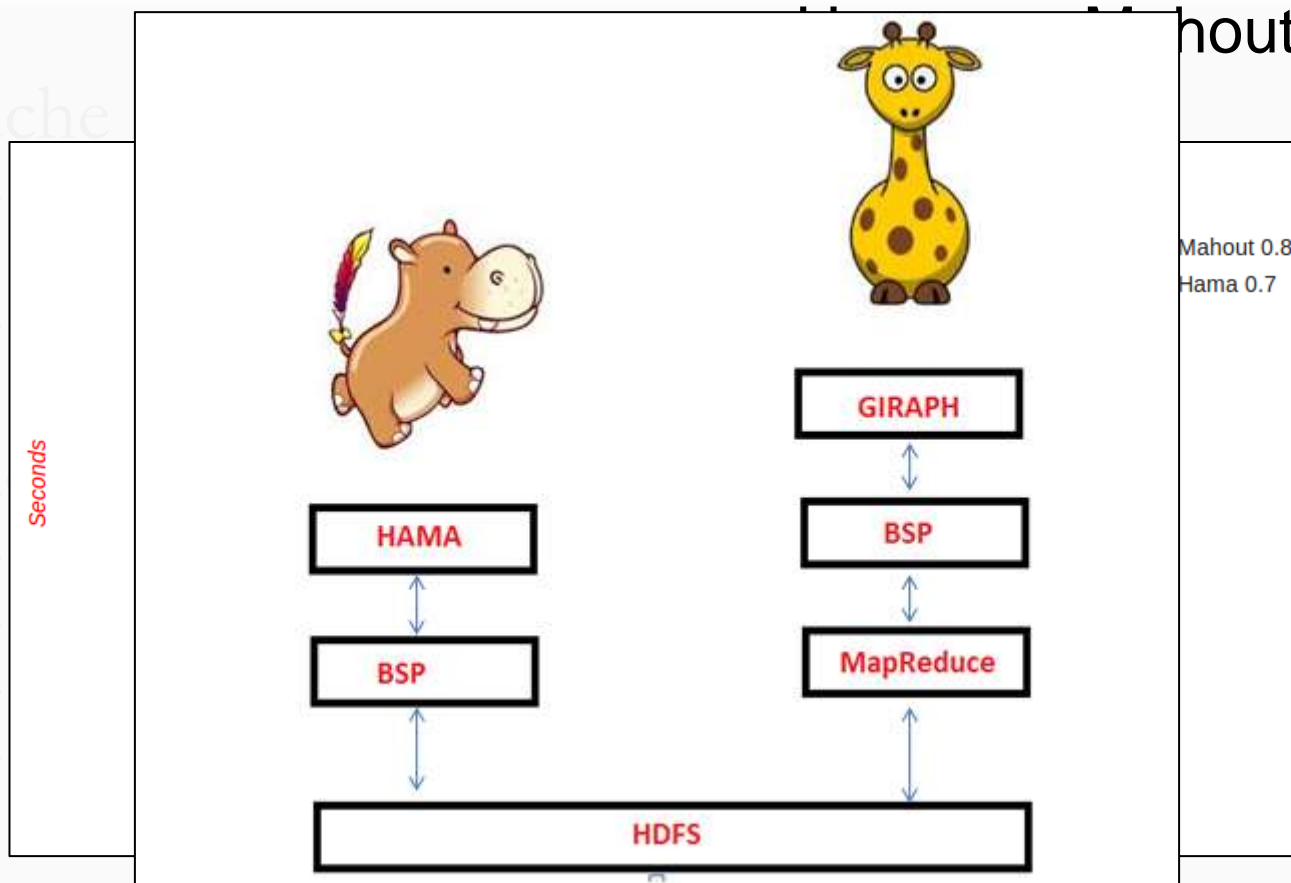
- Social networks are modeled as graphs
- Iterative Steps
  - Process Each Node Based on Neighbor's Info
  - Send New Calculation To Neighbors
  - Repeat



Votes between linked friends taken in rounds



# Iterative Model: Open Source Options



# Streaming Model

## Apache Storm

- “A system for processing streaming data in real time”
- Example Use Cases
  - Financial Services: Fraud Detection
  - Retail: Dynamic Pricing
  - Transportation: Driver Monitoring
- Key Features
  - Fast, Scalable, Fault-Tolerant, Reliable, Ease of Use
- History
  - Was previously associated with Twitter → Twitter Storm



# Streaming Model

Apache Storm

– Tuple

• Example:

– Stream

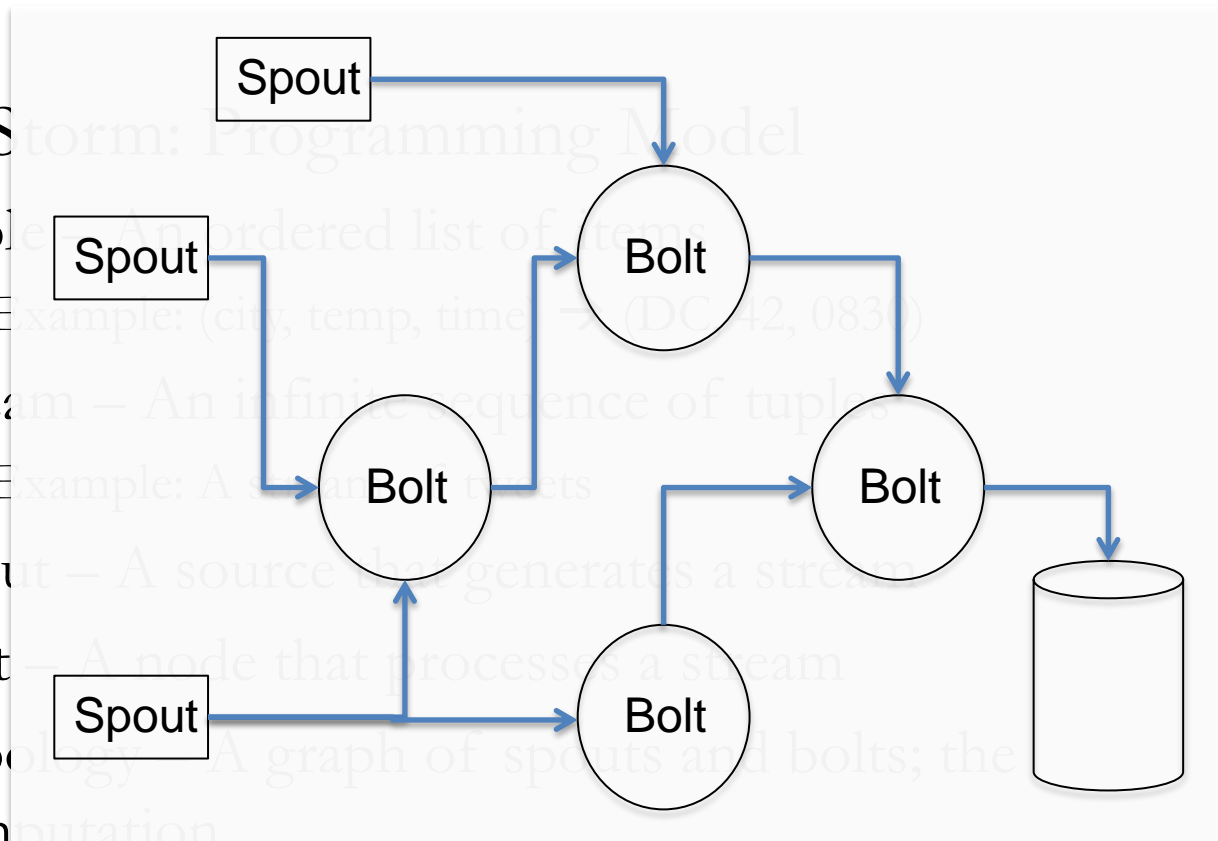
• Example:

– Spout

– Bolt

– Topology

computation



## Real World Example: Automa Systems

### Automa Systems

- Built on top of Apache Storm
- Automates shipper-trucker marketplace
  - Old Model: Compute then execute
  - Automa's New Model: Integrates real-time and historical traffic data, driver location, weather and customer delays to fully automate dispatch decisions
- Benefits for Fleet Optimization
  - Increased vehicle utilization and minimized fuel costs



# Streaming++ Model

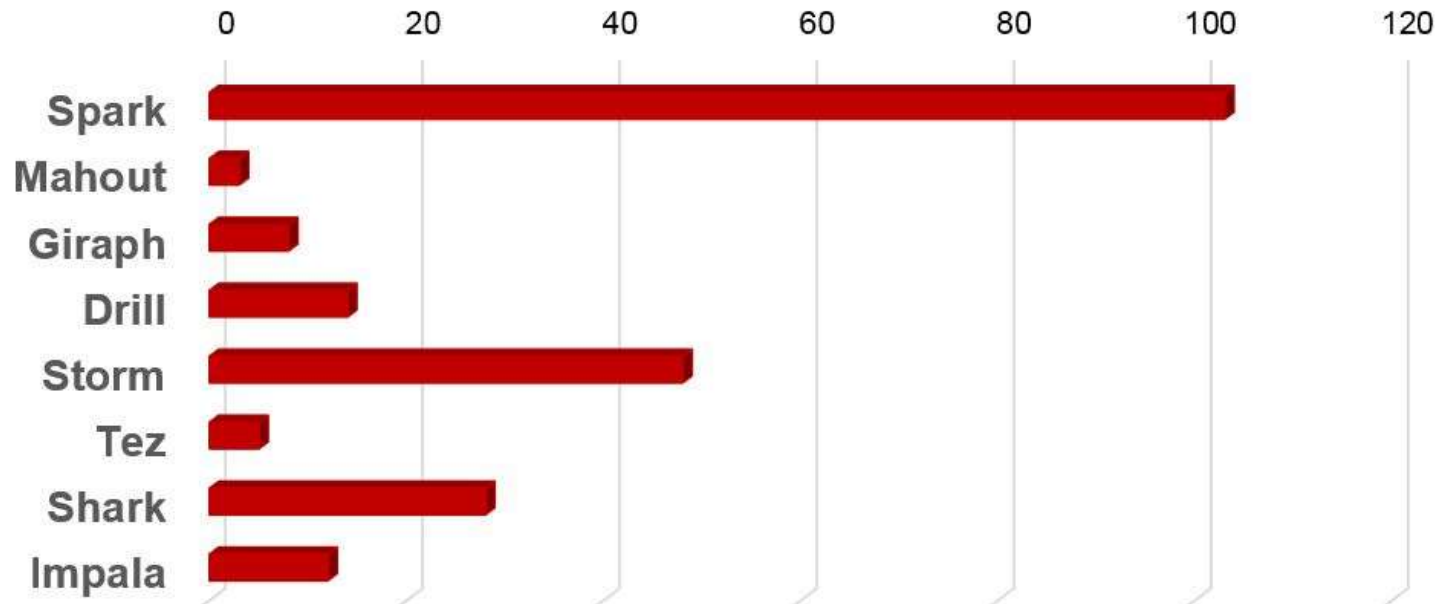


⌈

Number of Contributors

⌋

⌋

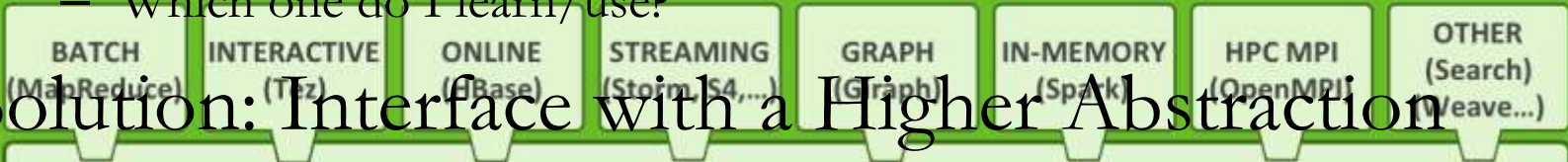


Source: GitHub Mirror

# But Wait...This Is Too Much

Hadoop, Apache, Iterative, Smiterative, ...  
Applications Run Natively IN Hadoop

- Which one do I learn/use?



Solution: Interface with a Higher Abstraction

- Instead of learning MapReduce, Spouts/Bolts, ...
  - Interact with a higher layer

YARN (Cluster Resource Management)

Two Approaches

HDFS2 (Redundant, Reliable Storage)

- Easy Programming Interfaces
- Software as a Service Interfaces



# Easier Programming Approach

## Consideration

- This approach requires infrastructure

## Examples

- Apache Hive – A SQL interface to Hadoop
  - Leverages the fact that many developers know SQL
- Apache Pig – A scripting interface for Hadoop
  - Complex programs are compiled into MapReduce jobs
- Rhadoop – R + Hadoop
  - Interact with Hadoop, HDFS and HBase
  - Developed by Revolution Analytics
- General purpose programming languages
  - Python, Java, Scala bindings for Spark

# Software-as-a-Service (SaaS) Approach

## Consideration

- This approach requires **NO** less infrastructure

## Examples

- Datameer – Excel-like Interface, Run R
- Dataminr – Twitter stream processing
- Domino – Run R, Python and Matlab
- IBM Watson API – Controlled invitations only
- IFTTT – Very simple stream processing



# Webinar Roadmap

Big Data definitions

Infrastructure

Metal

Management

Models

(part 1)

(part 2)

Algorithms

Data Processing

Machine Learning

Implications

Privacy

Smart Cities

Technology Tradeoffs

Session 2

Big Data Infrastructure

# MODELS: STORAGE

# Data Management

## Relational Databases

- Data is stored in tables (rows/columns)
- SQL – Structured Query Language
  - A language for creating, reading, updating and deleting (CRUD) table data
- Operating Principles: ACID
  - Atomicity – Each transaction is all or nothing
  - Consistency – Each transaction will result in achieving a valid DB state
  - Isolation – Concurrent transactions are equivalent to serial transactions
  - Durability – Committed transactions are resilient to power losses
- Popular Relational Databases
  - Oracle
  - IBM DB2
  - MySQL (open source)
  - Postgres (open source)

Employees Table

Employee_Number	First_name	Last_Name	Date_of_Birth	Car_Number
10001	John	Washington	28-Aug-43	5
10083	Aavid	Sharma	24-Nov-54	null
10120	Jonas	Ginsberg	01-Jan-69	null
10005	Florence	Wojokowski	04-Jul-71	12
10099	Sean	Washington	21-Sep-66	null
10035	Elizabeth	Yamaguchi	24-Dec-59	null

# Data Management

Relational Databases are widely used

- Banks and financial systems
- HR employee data management
- Retail inventory systems

The Problem with Relational Databases

- ACID makes scaling difficult
- ACID is not necessary in many Big Data scenarios

Big Data Solution → NoSQL

# CAP Theorem

Note: Different use of “consistency” from ACID

## 2000 Eric Brewer

- Shared data system can have at most two of the three following properties: Consistency, Availability and tolerance to network Partitions

Consistency	Availability	Partition Tolerant
<ul style="list-style-type: none"><li>• Total order on all operations</li><li>• Each operation looks as if it was completed at a single instant</li><li>• Updates applied to all relevant nodes at the same time</li></ul>	<ul style="list-style-type: none"><li>• Every request results in a response, even when severe network failures occur</li></ul>	<ul style="list-style-type: none"><li>• All messages sent from nodes in one partition to another may be lost due to a network failure but system will still respond</li></ul>
Shutdown instead of inconsistent response	Respond to all, maybe stale reads and conflicting writes	Unavoidable?

# Data Management with Big Data

## Most Big Data Applications are Distributed

- Distributed applications must be Partition Tolerant
- We must assume the network can go down
  - It is beyond our control
- Due to CAP theorem, distributed applications must choose between A (availability) or C (consistency)
- Many Big Data applications need Availability and can be satisfied with Eventual Consistency
  - It is okay if returned data is stale