Short Course:

Apache Spark for Social Scientists

October 25, 2018

Presented by :
Ian Thomas

Submitted by:
RTI International

3040 Cornwallis Rd
RTP, NC 27709

RTI International is a registered trademark and a trade name of Research Triangle Institute.

RTI International

delivering **the promise of science**
for global good

RTI International is an independent, nonprofit research institute dedicated to improving the human condition. We combine scientific rigor and technical expertise in social and laboratory sciences, engineering, and international development to deliver solutions to the critical needs of clients worldwide.

Introduction - Who am I

2015-Present: **RTI International: Center for Data Science**

- Data Scientist/Data Engineer
- RTI's first Spark pipeline for processing tweets
  - in production 2016

2009-2015: **Epic Games**

- Data Engineer/Analyst
- Fortnite Analytics Process
- Unreal Enigne Data Pipeline
- Gears of War 3 Data pipeline

Ian Thomas
Center for Data Science
RTI, International

Course Objectives

- Demystify the jargon
- What is Spark?
- When to use it
- Why it’s useful for Social Science
- How to get started

Schedule

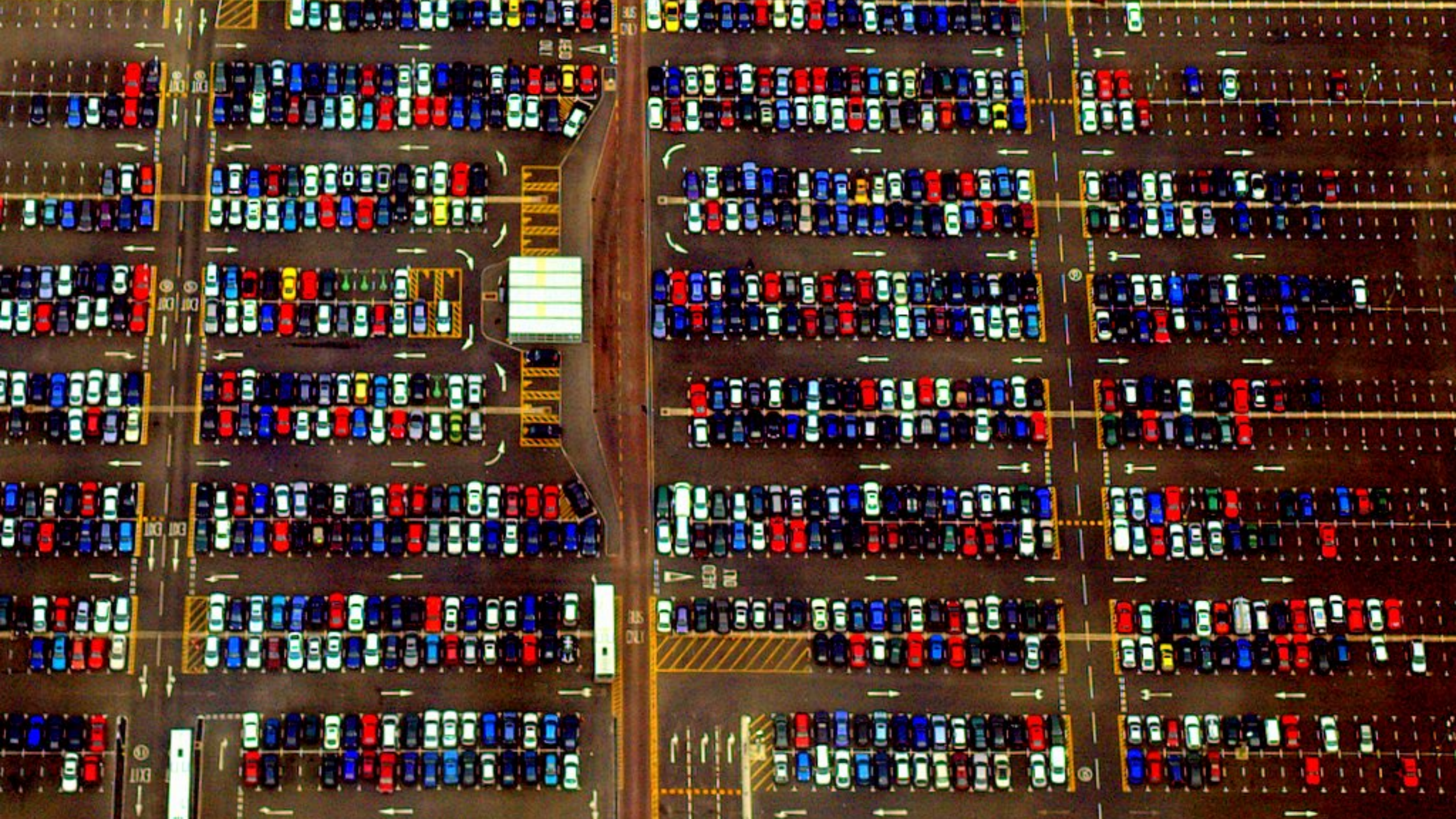
- 13:00 – 13:50 Intro & Concepts
- 14:00 – 14:50 Spark Up and Running
- 15:00 – 15:50 Programming in Spark
- 16:00 – 17:00 Advanced Features

Why Spark?

Extremely Brief History of Computing

- 1937- 1946 –Vacuum Tubes
- 1947 - 1962 – Transistors

- 1963 – Present Integrated Circuits
- 1981 – MS DOS
- 1996 – Deep Blue Wins Chess



HPC

High Performance Computing (HPC)

- Really REALLY nice hardware
- Many Fast processors
- Lots of Fast Memory
- SERIOUSLY Fast Networking
- Fast Everything!
- Bring the data to the compute as fast as possible

Found at

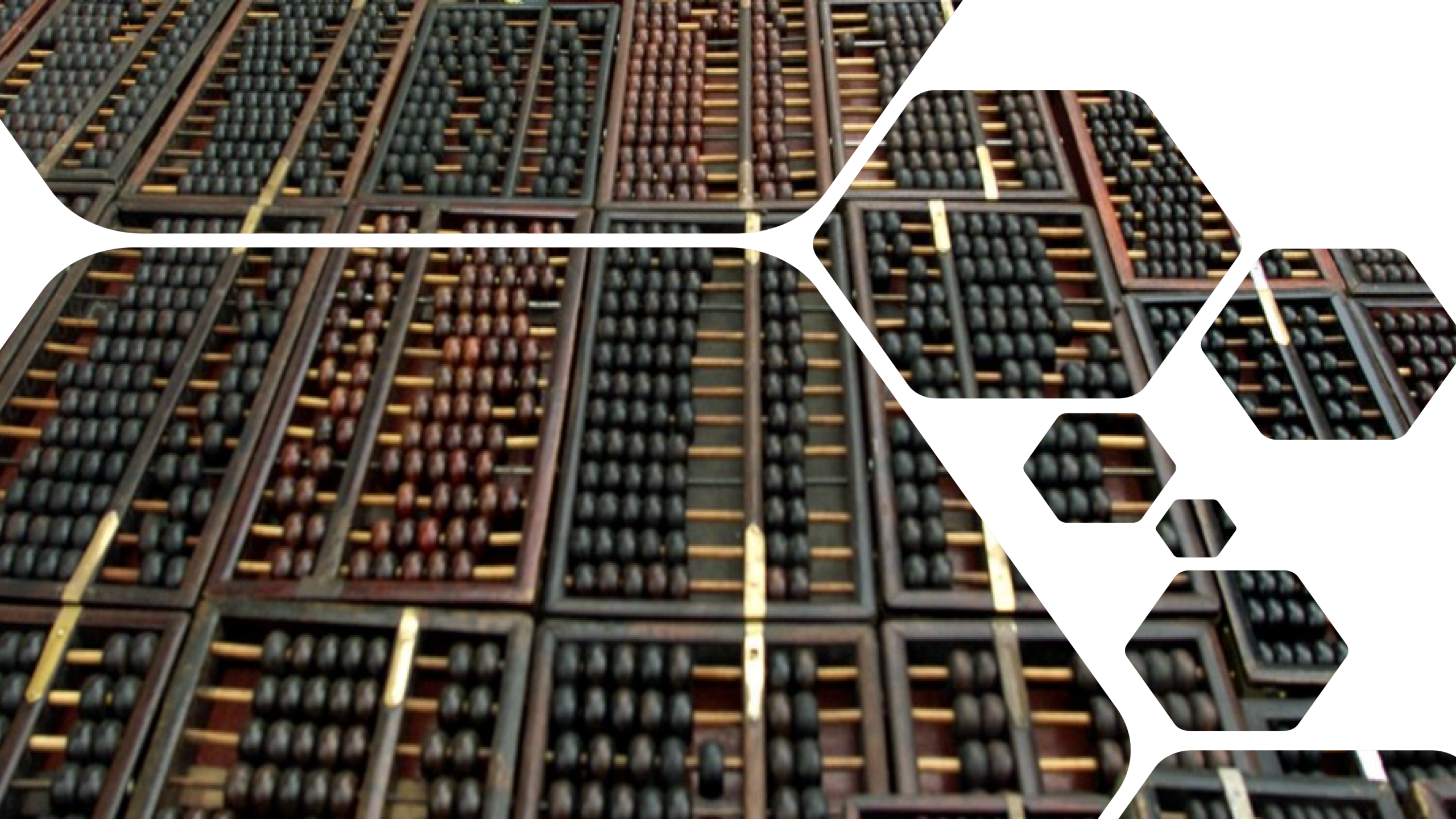
- Universities
- Large Organizations

Pros

- Intuitive Programming
- Really Fast Computation
- Can handle interdependent algorithms

Cons

- Expensive
- Bottlenecks
- Really fast hard drives and networks are still pretty slow

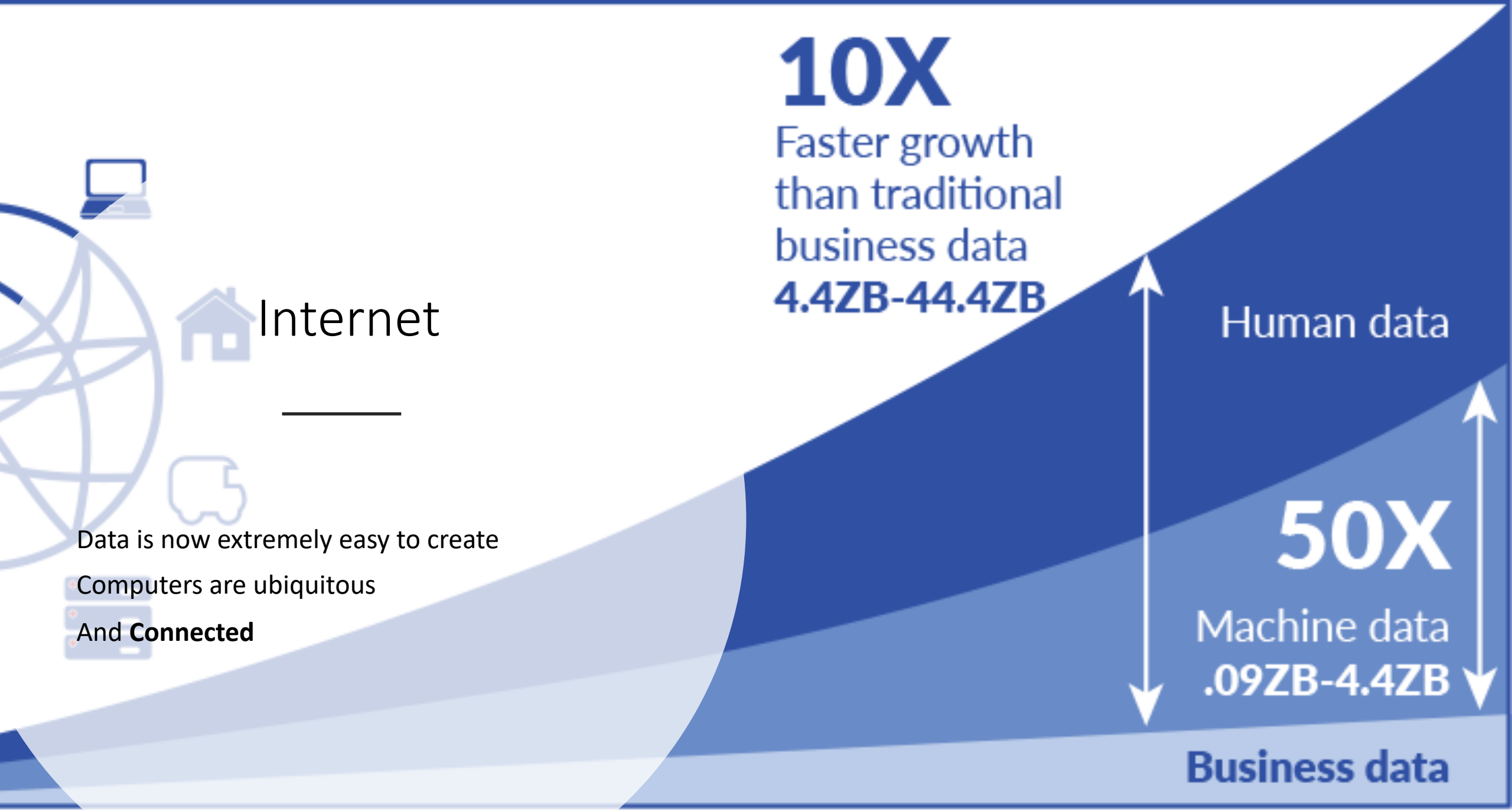


Internet

Data is now extremely easy to create

Computers are ubiquitous

And **Connected**



ebigdata.com/2017/02/16/the-exponential-growth-of-data/)

Distributed Cluster Computing

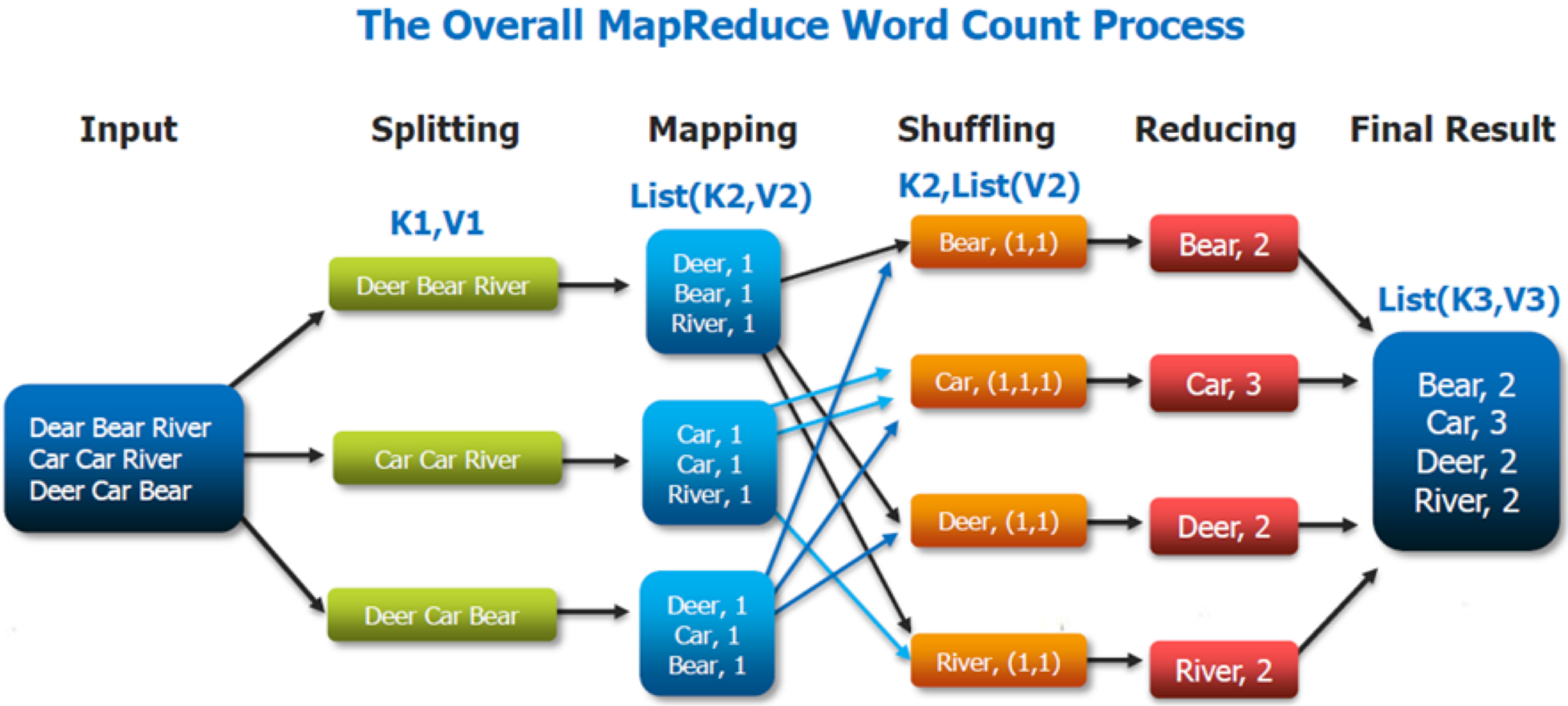
Expect failure

- Every computer has a failure rate
  - Even expensive servers
- $: uptime ratio better with consumer grade computers
- Programs that run on multiple computers can expect some of them to be broken

Data Bottleneck

- The fastest network is slower than a hard drive
- The fastest hard drive is slower than memory
- **Bring the compute to the data**

Map Reduce


The Overall MapReduce Word Count Process
Input
Splitting
Mapping
Shuffling
Reducing
Final Result
K1,V1
List(K2,V2)
K2,List(V2)
List(K3,V3)
Dear Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Map Reduce

Pros

- Made data-to compute possible
- Cluster computing easier than ever before
- Rich ecosystem of software for production environments

Cons

- Difficult to program
- Specialized skill
- Chaining Map->Reduce gets tedious
- Dependency Accounting

What is Spark?

- Computation Engine For Distributed Computing
  - Fast
  - In-Memory
- Software for your Software
- Run modern analytical tools on Big Data

- 2009 - Research project at UC Berkley
- Apache 2.0 License
  - Open Source
- 2.3.2 – Latest Stable Release
  - September 24, 2018

100TB Daytona Sort Competition 2014

	Hadoop MR Record	Spark Record
Data Size	102.5 TB	100 TB
Elapsed Time	72 mins	23 mins
# Nodes	2100	206
# Cores	50400 physical	6592 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s
Sort Benchmark Daytona Rules	Yes	Yes
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**

- Spark sorted the same data 3X faster using 10X fewer machines than Hadoop MR in 2013
- All sorting took place on disk (HDFS) without using Spark's in-memory cache!

More info:
http://sortbenchmark.org
http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

databricks

Work by Databricks engineers: Reynold Xin, Parviz Deyhim, Xiangrui Meng, Ali Ghodsi, Matei Zaharia

Is Spark a Language?

- No
- It's a Computing Engine Written in Scala
- Runs on the Java Virtual Machine (JVM)
- Orchestrates operations on cluster
  - a.k.a. Takes your code and runs it on multiple computers

Allows you to write in

- Scala
- Java
- Python
- R
- SQL

Libraries for

- Machine Learning
- Modeling
- Network Analysis

Why Use Spark?

- Tools made for Analysis:
  - SQL
  - R
  - Python
  - Scala
- Runs on
  - Your laptop
  - Your servers
  - In the cloud

Why NOT use Spark?

- Learning Curve
  - Different way of thinking about computation
  - Basically alien technology
- Some slow things become FAST
- Some fast things become SLOW
- Debugging can be difficult

When to Use Spark?

- Good rule of thumb : 1TB+
- Frequent calculations on a large dataset take too long
- Can’t run analysis on the full dataset on your own computer
- Your company has a cluster

If your IT folks say things like:

- HDFS,
- Hive,
- Pig
- Cloudera,
- Hortonworks,
- Hadoop

Reddit

- TBs of forum data
- 10+ years of history
- Social impact
- Not Representative of the Population
- Really Big Lamp Post

- Caveat emptor, computational social science: Large-scale missing data in a widely-published Reddit corpus
- Massanari A. Participatory Culture, Community, and Play: Learning from Reddit. 2nd ed. New York: Peter Lang Publishing Inc.; 2015.
- Massanari A. # Gamergate and The Fappening: How Reddit's algorithm, governance, and culture support toxic technocultures. New Media & Society. 2015; p. 146144815608807.
- Lotan G, Graeff E, Ananny M, Gaffney D, Pearce I, et al. The Arab Spring—the revolutions were tweeted: Information flows during the 2011 Tunisian and Egyptian revolutions. International journal of communication. 2011;5:31.

Common Crawl

- Open repository of **web crawl data** that can be **accessed and analyzed by anyone**.
- Billions of pages
- Trillions of links
- 7 years of data

Prevalence of Web Advertising

https://commecica.com/2018/06/27/web-ad-prevalence/

Finding Industry Trends

https://github.com/CI-Research/KeywordAnalysis

Hyperlink Graphs

http://webdatacommons.org/hyperlinkgraph/index.html#toc1

Other Examples

- News
- Transcripts
- Hackers News\Reddit\Twitter\StackOverflow\Wikipedia
- Reviews (movies, restaurants, beer, wine)
- Emails (Enroll, ASF public ML archives)
- Census Data (US, UK, UN, Japan, etc)
- Transportation (Taxi, Flights, Bicycles)
- Climate Genome

- AWS Public Datasets https://aws.amazon.com/public-data-sets/ Yahoo Webscope https://webscope.sandbox.yahoo.com/ Stanford Network Analyser Project http://snap.stanford.edu/data/ Physics Research http://opendata.cern.ch

File Read Times

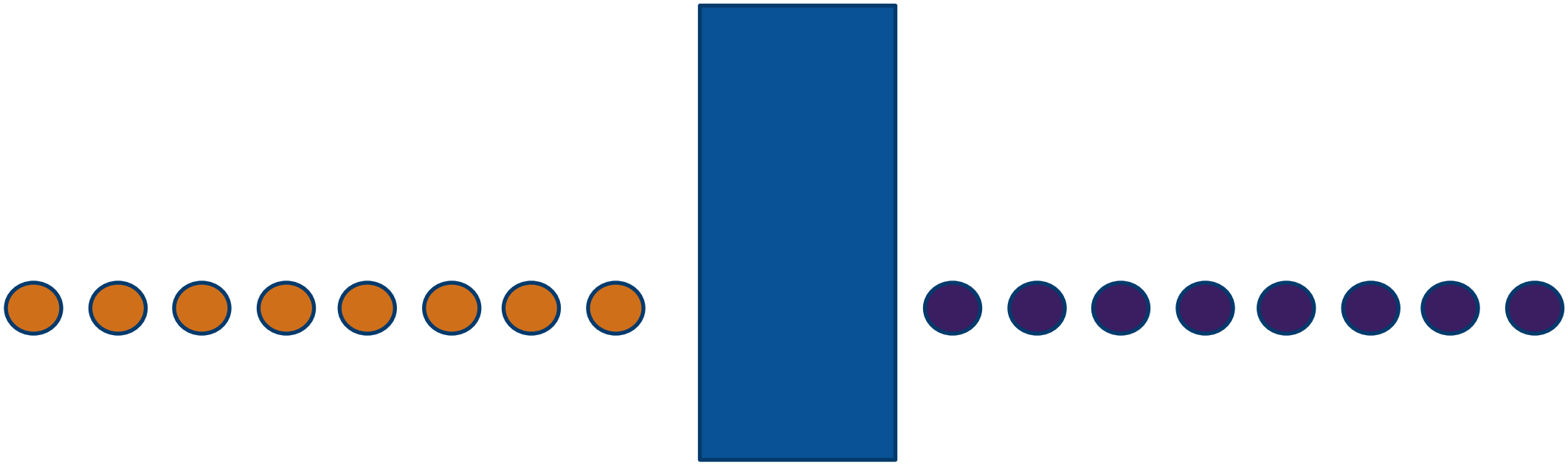
File Size	Ideal	Large Files	Small Files
1GB	2.67 Seconds	12.5 Seconds	23.8 Minutes
100GB	4.4 Minutes	20.8 Minutes	39.7 Hours
1TB	44.4 Minutes	3.5 Hours	**2 weeks**
1PB	**1 Month**	**4.8 Months**	**45.9 Years**

Things to use instead of Spark

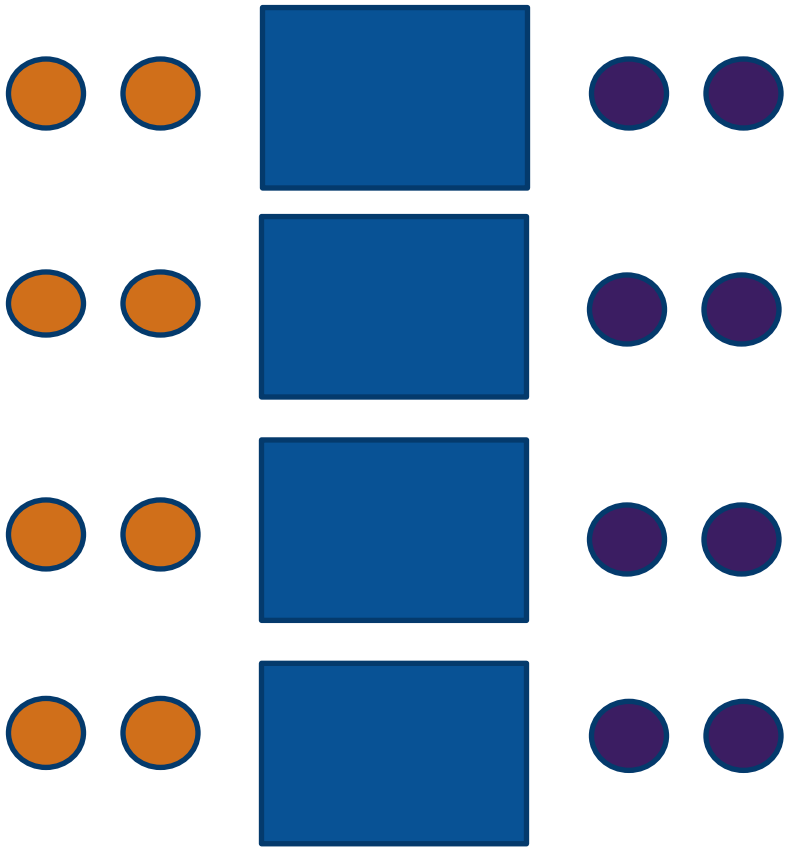
- Use LESS data
- SQL
- Specialized Libraries
  - SciPy
  - NumPy
  - Numba
- Cloud Analytic Databases
  - Amazon Redshift
  - Google Big Query
  - Azure SQL Data Warehouse
  - Snowflake
- Open Source Analytic Databases
  - MariaDB
  - Clickhouse
  - LocustDB

How Spark Works

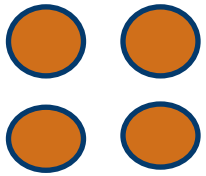
Serial Processing

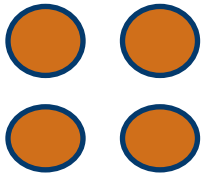


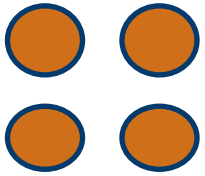
Parallel Processing

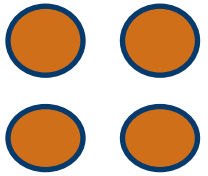


Distributed Parallel Processing

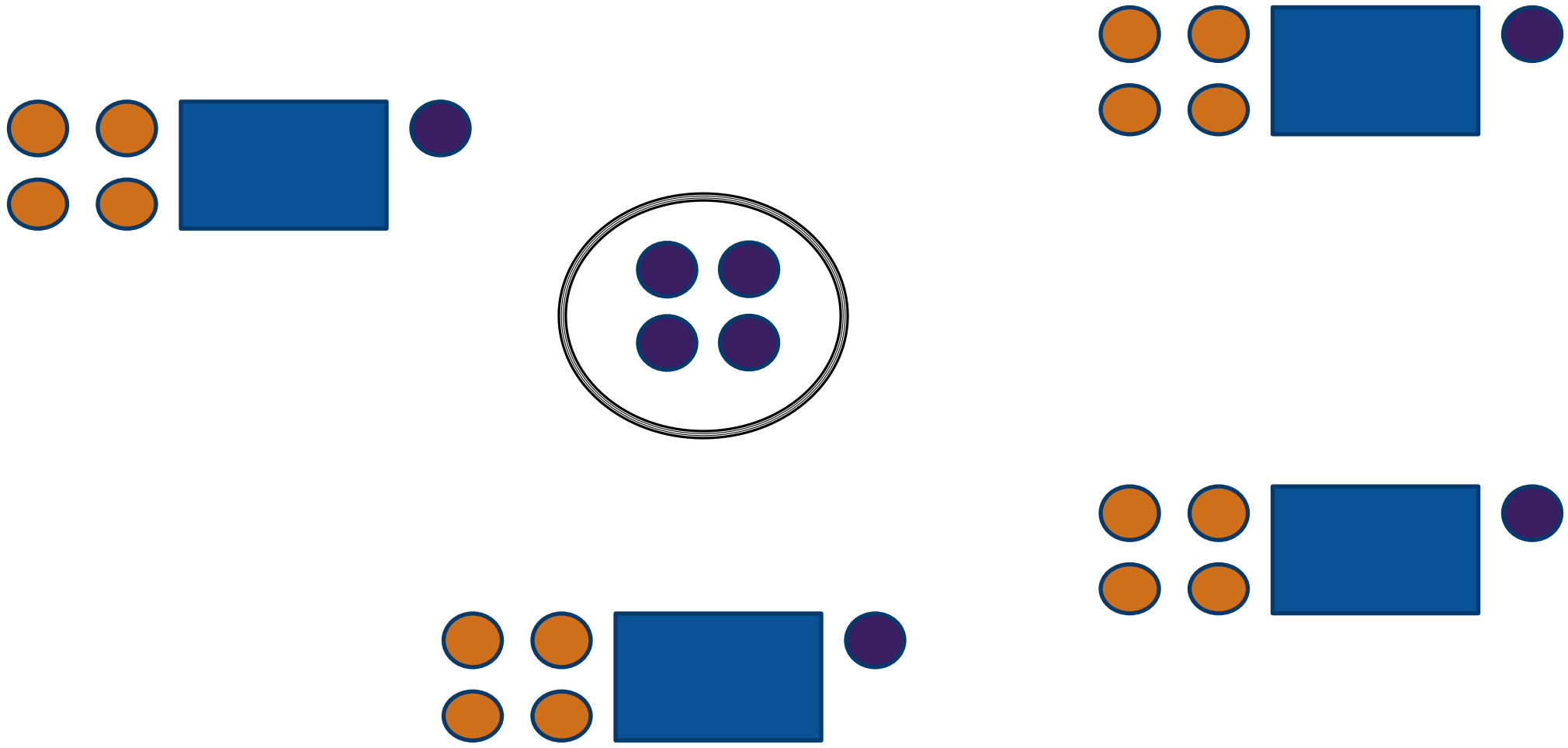








Improved Distributed Parallel Processing



Using Spark

Programming in Spark

- Goals
- Get it running
- Access the data
- Manipulating the data

Get It Running

- Docker
  - docker run --rm -p 8888:8888 -v "$PWD":/home/jovyan/work jupyter/all-spark-notebook

Spark Session

- Connection to a Spark cluster, and can be used to create RDDs, accumulators and broadcast variables on that cluster.

- Spark 1
  - Spark Context
  - Sql Context
- Spark 2
  - Spark Session

Working with Spark

1 computer	**Proof of Concept**	**Laptop**
10s of computers	**Prototype / Estimate Cost**	**AWS Spot Instances**
100s of computers	**Run on all data/ Deploy to Production**	**Dev Ops**

Debugging Logs

- The biggest frustration when working with spark
- It’s a game of telephone
- JVM->Scala->Python->Pyspark
- JVM->Scala->R-> SparkR

File Types

- CSV
- TSV
- JSON
- RDS
- Text (ASCII)
  - Fixed width
- Avro
- Parquet

Markdown

- Text only formatting
- Easy to read
- Useful for adding notes in Jupyter Notebook

- # H1
- ## H2
- *bold*
- ~strike~

RDD

Resilient Distributed Dataset not to be confused with Random Digit Dialing

RDD
Resilient Distributed Dataset

"RDDs are fault-tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operators."

- Breaks free of the Map Reduce paradigm
- Fundamental to Spark's speed and success
- Generalized computing
- Best thought of a set of instructions
- All instructions are

Resilient

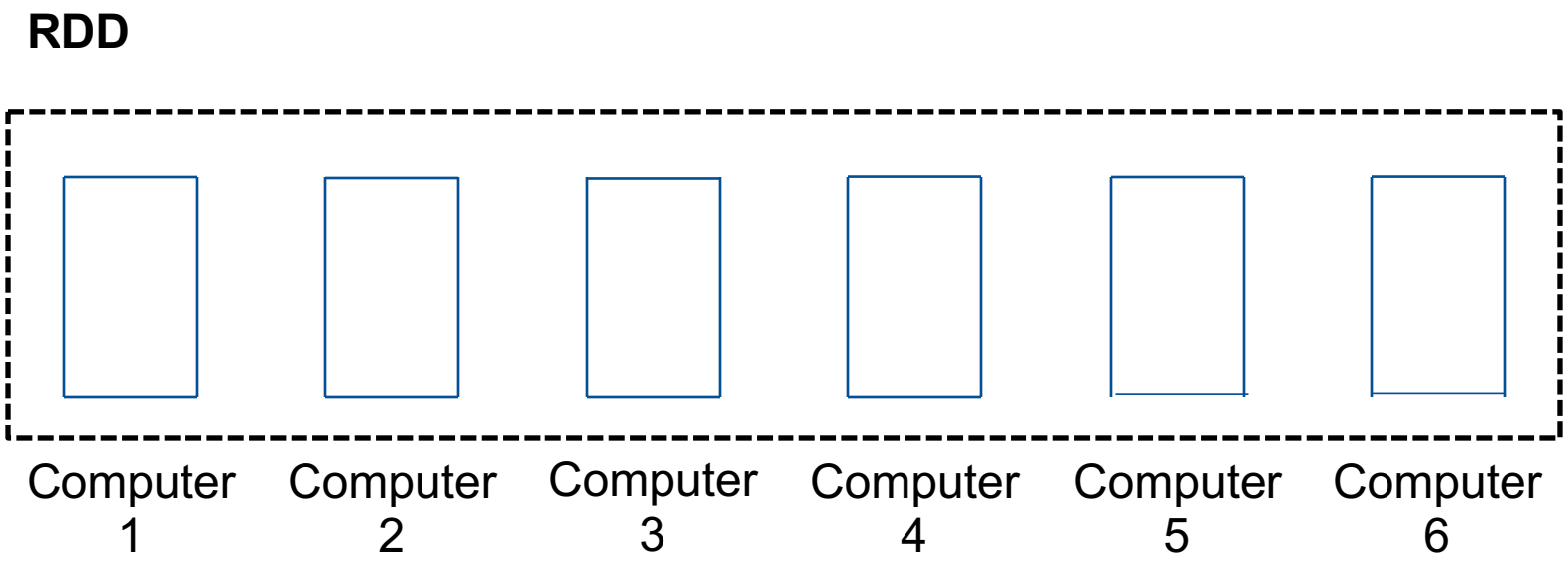
- Representation of all the data across cluster
- Lineage
  - RDD1+RDD2+RDD4+RDD5 = RDD.collect()
- Immutable
  - Changes made by copy the old set of instructions and adding new instructions
- Re-creatable
  - If a partition's RDD becomes corrupted it as enough information to recreate it.

Distributed

- Course Grained
  - Instruction applied to every element
- Transitions
  - Load data
  - Select
  - Count

- Consistency Easily Assured
  - Read Only
- Automatic Work Placement
  - based on data locality

Dataset


RDD
Computer 1
Computer 2
Computer 3
Computer 4
Computer 5
Computer 6

RDD Operations

- Transformation
  - Load data
  - Sort
  - Filter
  - Group
  - Map
- Actions
  - Collect
  - Count
  - First
  - Take
  - Save As

Dataframe

Dataframe API

- RDDs let you define HOW
- Dataframes let you define WHAT to do
  - Spark figures out HOW

A Dataframe by Any Other Name

- **Dataframe**
  - A fast and efficient object for data manipulation
  - Integrated indexing
  - Slicing
  - Aggregation

- Dataframes can be found in many places
  - Pandas
  - R
  - SQL Table

Using Dataframes

- Spark Dataframes are built on top of the RDD
- Immutable
- Lineage maintained
- Lazy
- Transitions and Action concept still useful

Basic Functions

Read

- Read
- Points to the data
- Loads some of it
- Predicts the schema

- df = spark.read.csv(‘filename.csv’)
- df = spark.read.json(‘filename.json’)
- df = spark.read.<format>(‘filename’)
- df.printSchema()

Show

- Loads some of the data
- Displays the schema and 20 rows
- Only show’s some of the data
- Great for investigating the data and testing your work

Filter and Where

- Return a new Dataset with a subset of the items in the file
- Reduce the data to be operated on
- Takes normal query operators
- Both functions are equivalent

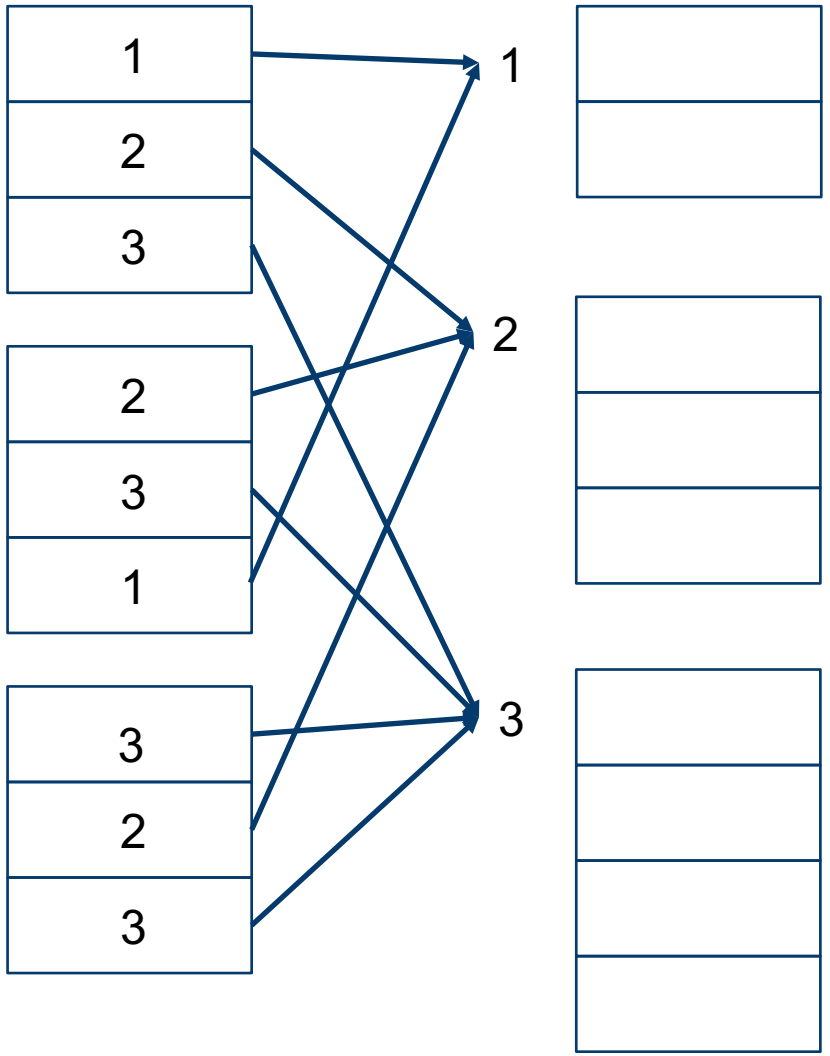
- df.firstName
- df[‘firstName’]
- ‘firstName’

Select

- Create a new dataframe with selected columns
- Not to be confused with Where
- Similar to SQL Select

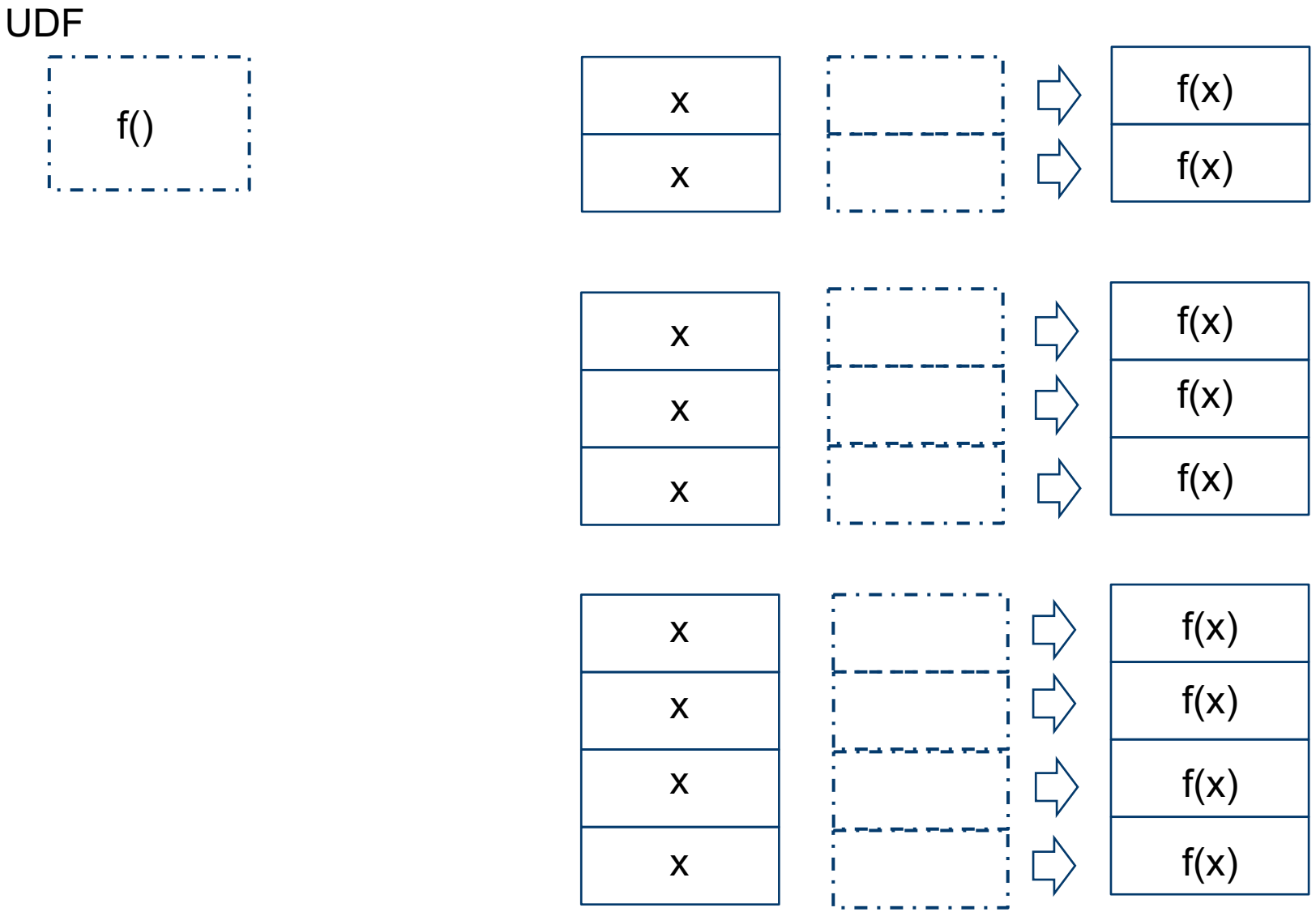
Aggregation

GroupBy OrderBy and Sort



Adding Columns

User Defined Functions



The Cloud

Document Stores

AWS S3

Azure Cosmos DB

Google Cloud Storage

- Not a database
- Key Value Store
  - If you know the key you can quickly get the value
- Great for quick access to whole documents
- Each document is read only
  - To make a change you make a new document
  - New Key New Value
- Easily Distributed

On Demand Computing

- Pay for the time you want
- Connect to Document Store
- AWS EMR
- Azure HDInsight

Beyond the Basics

- **Machine learning in Spark**
- **Clustering**
- **K-means**
- **LDA**
- **Model Selection**

- Graph Processing
- Network Analysis
- Neighborhood Aggregation