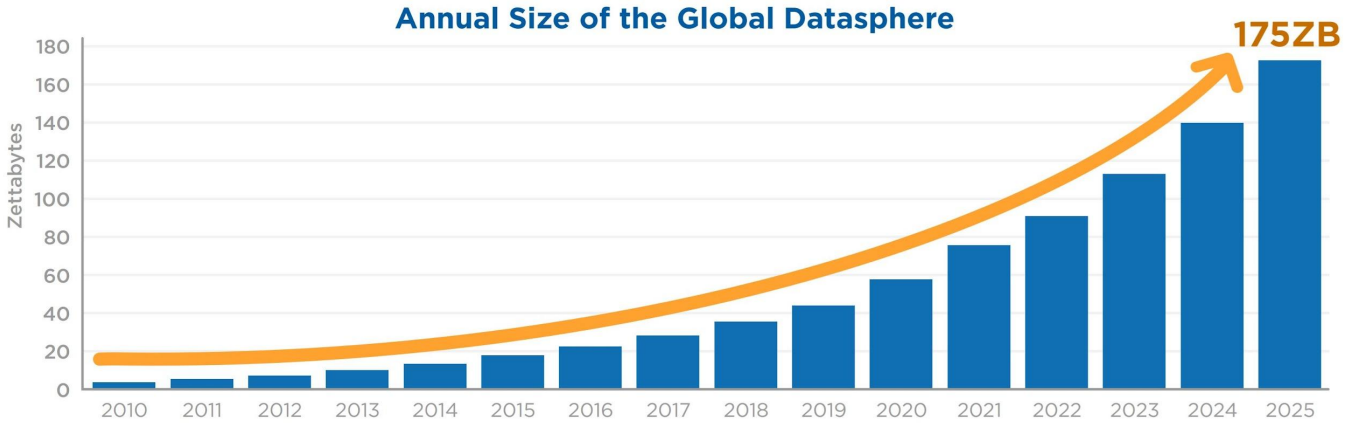
Distributed Technology for Big Data

ISTA 322 - Data Engineering

Reminder - Tons of data these days!

- As we talked about at the start of class, there are two issues that lead to the rise of data engineering.
  - 1 - Data are often messy and need lots of processing to be useful
  - 2 - The *amount* of data has boomed



So what does ‘big’ mean?

- Simplest version - too big to fit in memory of one machine
- Current machine has 16gb - could add more and churn through it
  - Could also batch process
- Local processing isn’t practical for many situations
  - Netflix’s 10bn events a day
  - FB processing 10’s of petabytes a day - ref.
    - 10,000,000gb

Solution - many machines and clever computing

- At some point it’s easier to just distribute the workload over many smaller machines vs. trying to make one do it all.

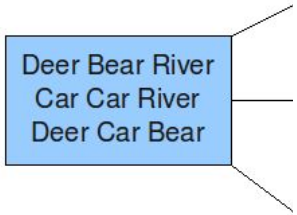
Instance	GPUs	vCPU	Mem (GiB)	GPU Mem (GiB)	GPU P2P	Storage (GB)	Dedicated EBS Bandwidth	Networking Performance
p3.2xlarge	1	8	61	16	-	EBS-Only	1.5 Gbps	Up to 10 Gigabit
p3.8xlarge	4	32	244	64	NVLink	EBS-Only	7 Gbps	10 Gigabit
p3.16xlarge	8	64	488	128	NVLink	EBS-Only	14 Gbps	25 Gigabit
p3dn.24xlarge	8	96	768	256	NVLink	2 x 900 NVMe SSD	19 Gbps	100 Gigabit

Solution - many machines and clever computing

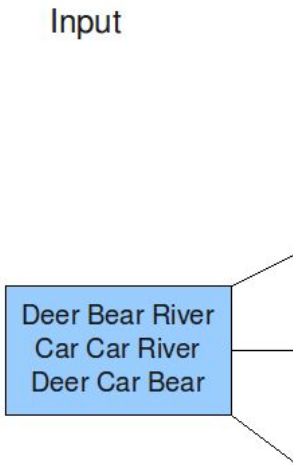
- Enter MapReduce
- MapReduce is a computing framework for dividing up computation across the machines on a cluster
- Two main stages to MapReduce processing
  - Map
  - Reduce

MapReduce - Get word count

Input

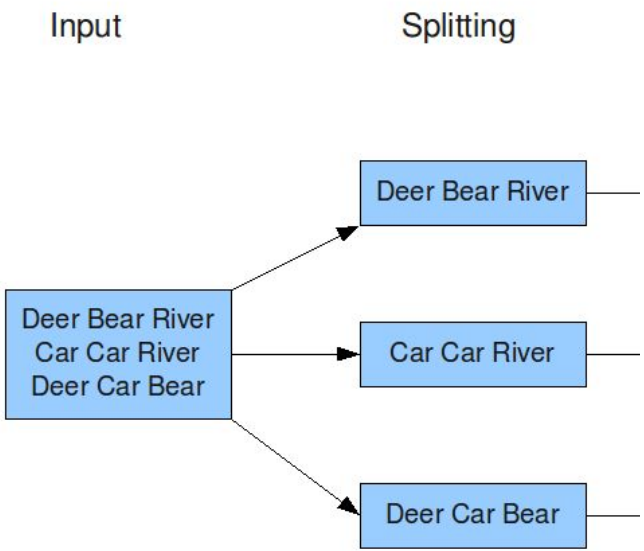


MapReduce - Input



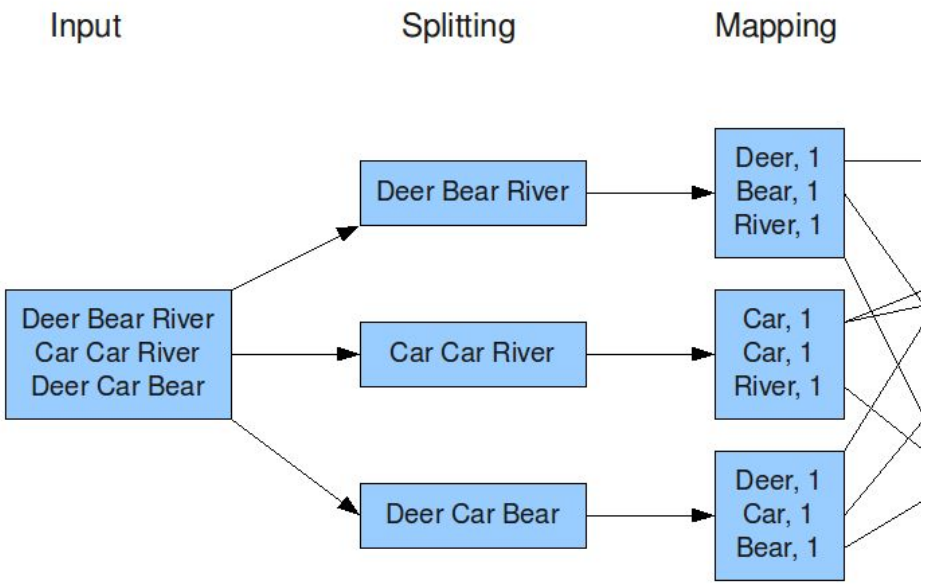
- Data stored on HDFS - Hadoop Distributed File System
- Fault tolerant
- 64-128mb blocks
- Replicated 3x

MapReduce - Splitting



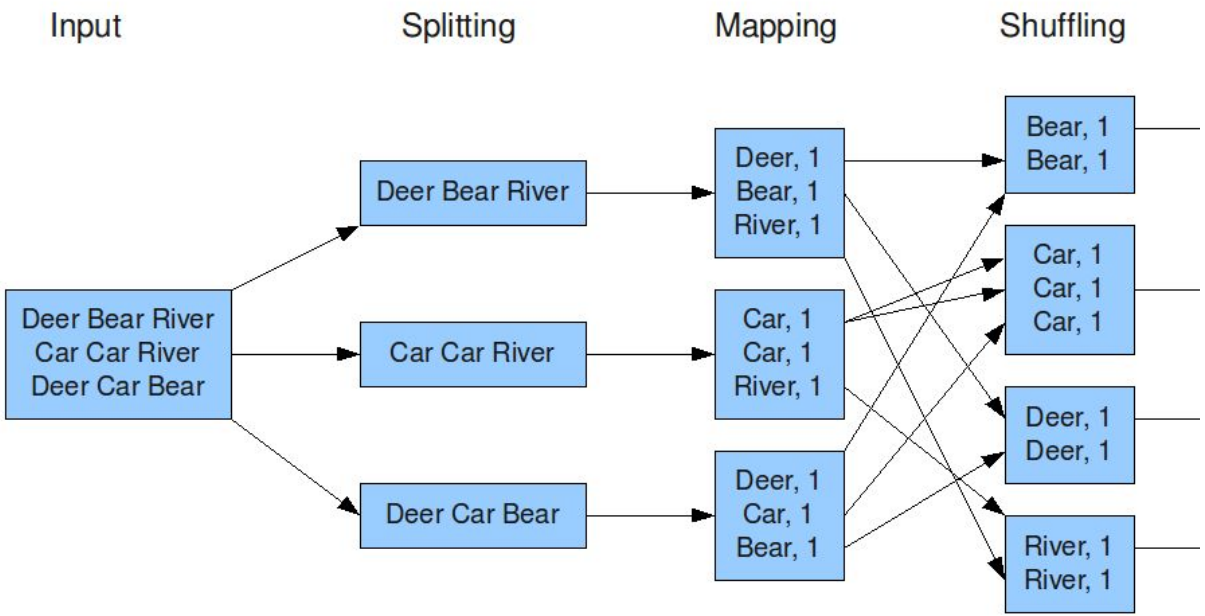
- Input data split up
- 64-128mb
- Sent to different worker nodes

MapReduce - Mapping



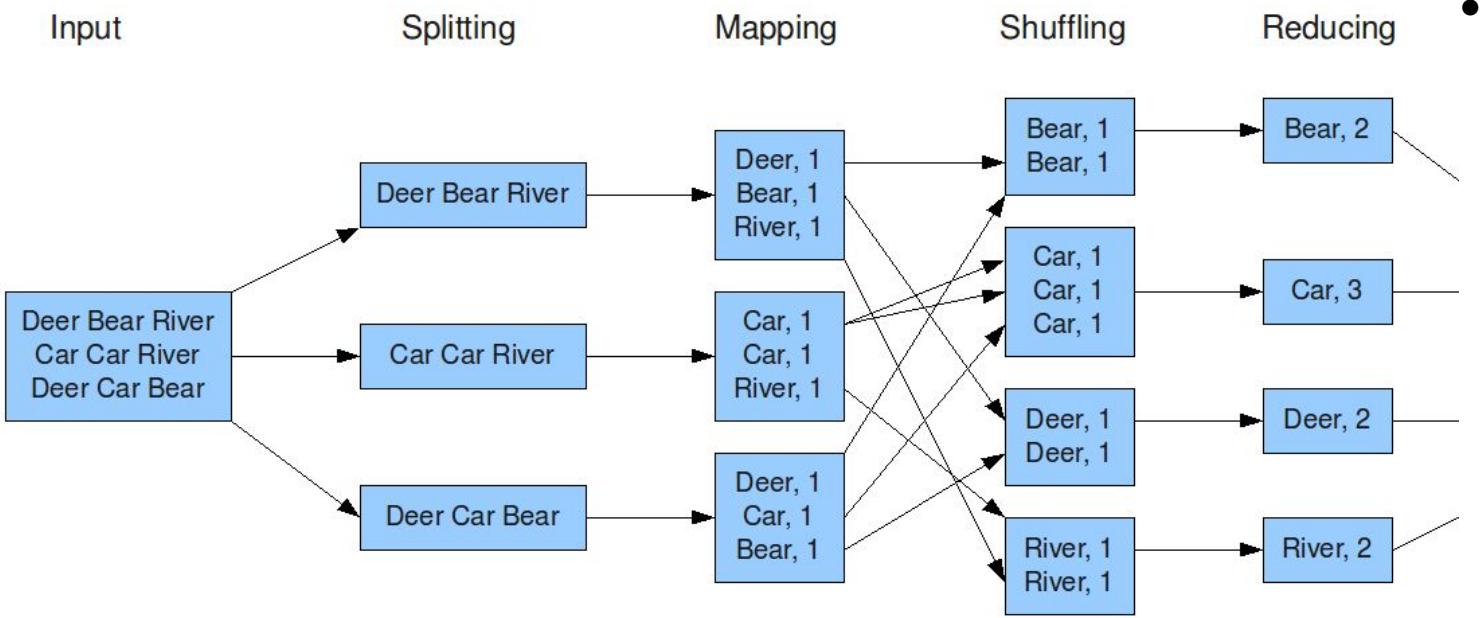
- Data are ‘mapped’ to key-value pairs
- This is the ‘Map’ in MapReduce
- Will write to disk

MapReduce - Shuffling



- Key value pairs are then reorganized so they reside on the same node

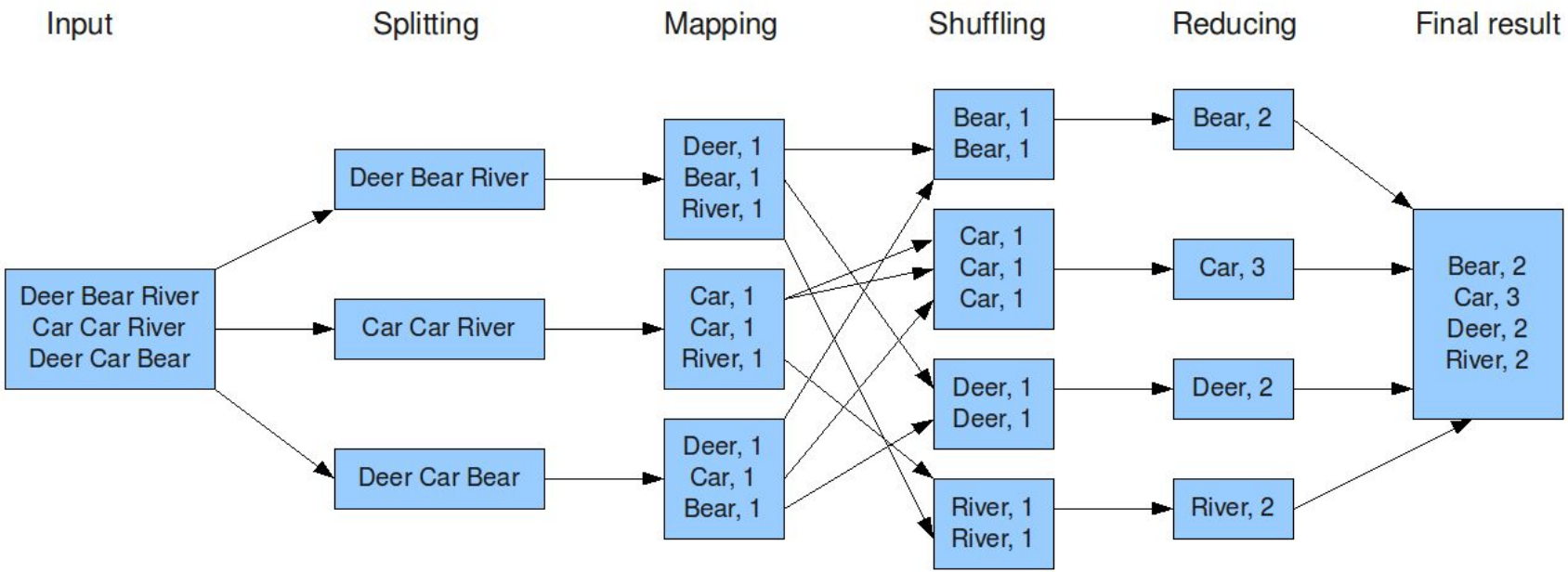
MapReduce - Reducing



- Information is Reduced by aggregating the shuffled Map results

MapReduce

- Final results brought back together
- Write to disc



MapReduce - Pros

- Can process TB’s of data in minutes
- HDFS is also fault tolerant
  - Computation happens where data are stored
- It automatically scales to different size data given resources available
- It tracks what each node is doing - fault tolerant!
- Can do SQL, data wrangling, machine learning, etc
- FYI: Hadoop ~ MapReduce

MapReduce vs. Spark

MapReduce vs. Spark

- Spark can be 100x faster than MapReduce
- Does everything in memory vs. writes to disk of MapReduce
- MapReduce needs HDFS
- Spark can take HDFS, S3, Blob, and others
- More expensive
- Really easy to code via Pyspark

Pyspark

- Python API for Spark
- Spark dataframes work really similar to Pandas
  - Syntax is similar too!
- Has expansive ML library
- Easy to use Jupyter-like services
  - Amazon EMR (Elastic Map Reduce)
  - Databricks
- We’ll be using Databricks

Databricks

- Databricks founded by creators of Spark
- Uses Jupyter style notebook
  - Code Python, R, Java, Scala
- Easily connects to clusters
- Many companies use Databricks
- Has free community edition that we'll be using
  - https://community.cloud.databricks.com/login.html