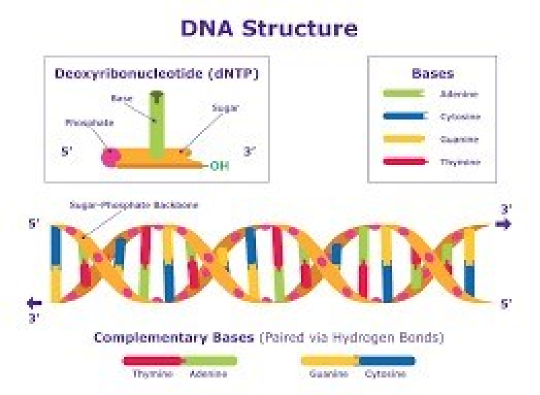
Big Data and MapReduce

Data Sources



Big Data

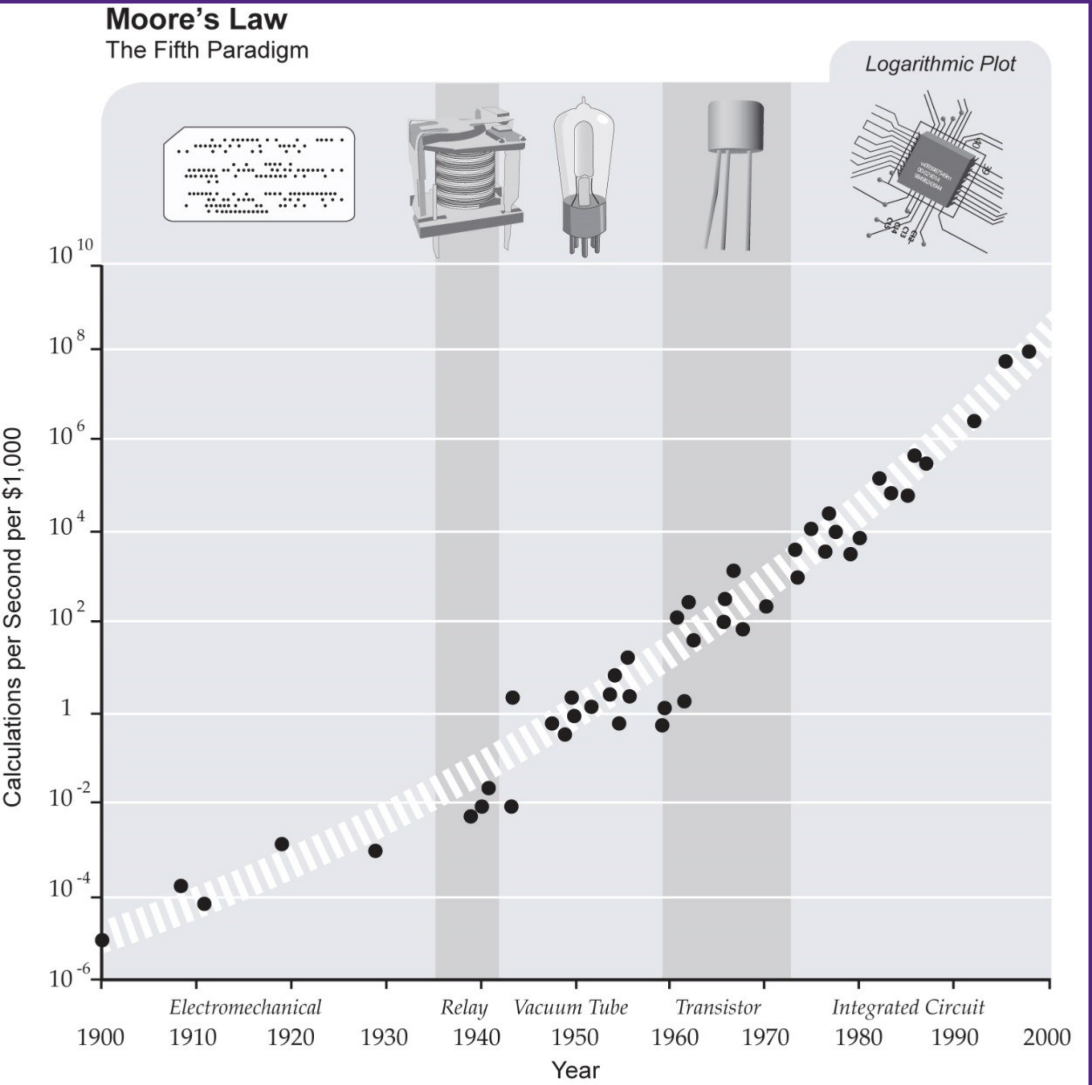
- “How much data is there?”
  - 2010: 1,200,000,000 TB
  - 2020: 38,500,000,000 TB
  - 2020: It would take 181 years to download all the data from the Internet
- Internet users generate about 2,500,000,000 TB each day
- Using big data, Netflix saves 1 billion dollars per year on customer retention

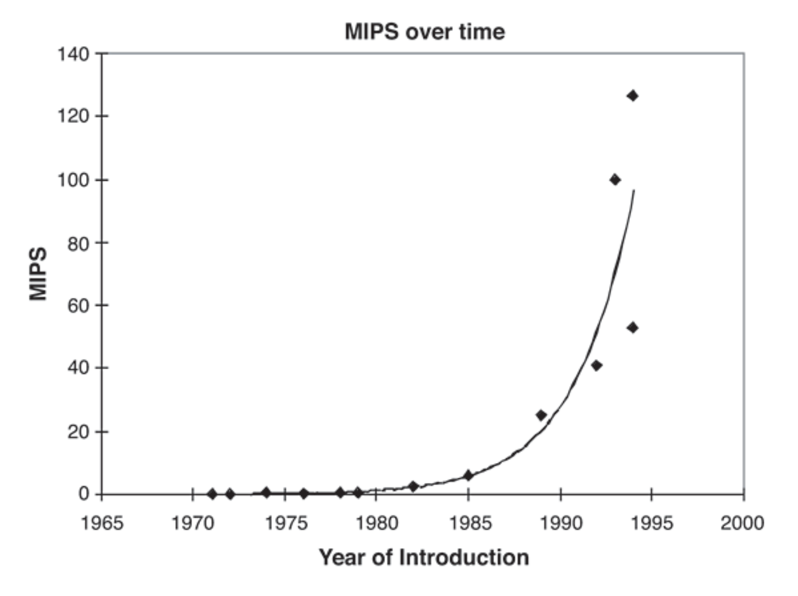
Big Data vs Moore’s Law

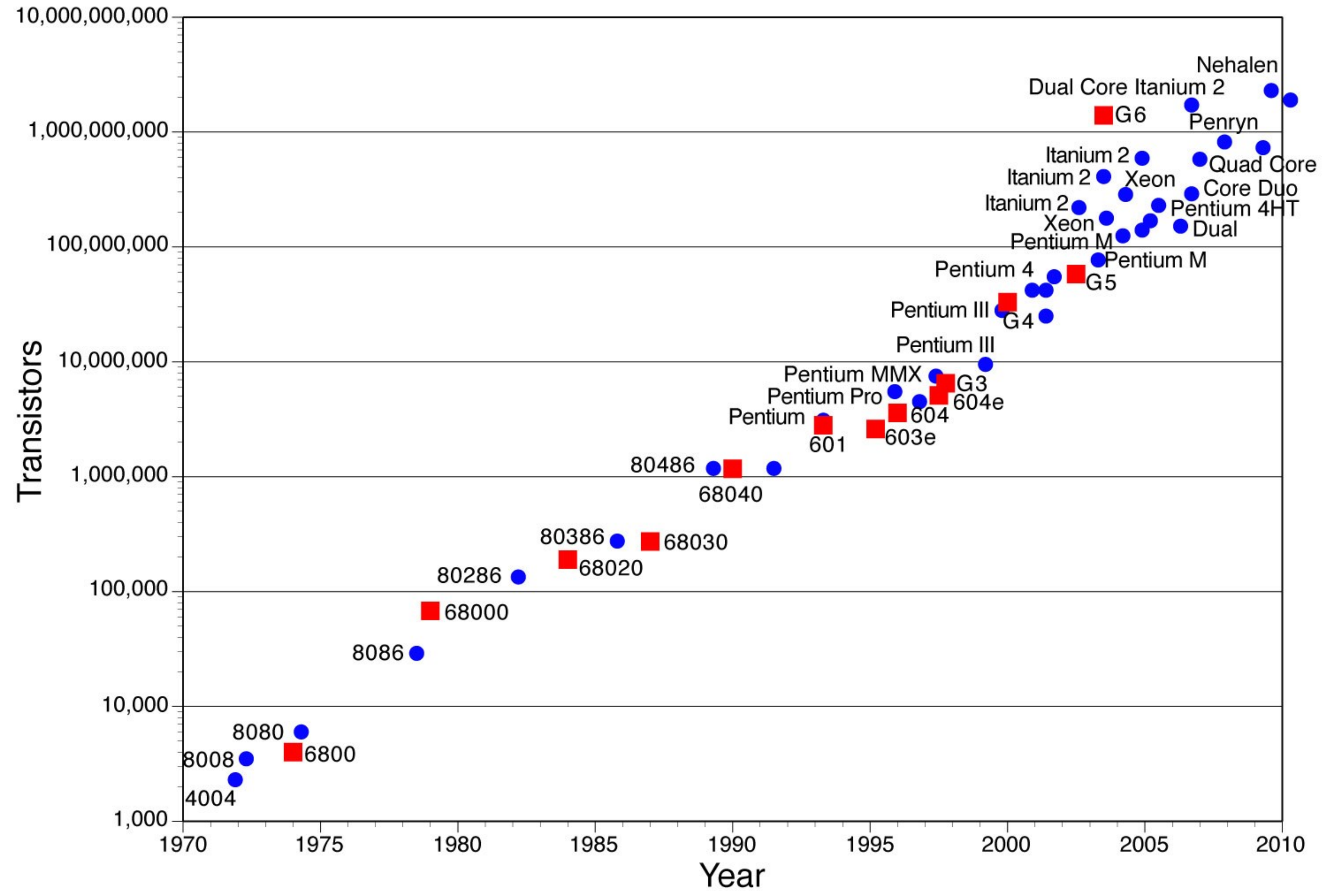
- Gordon Moore, 1965: The number of transistors on a chip will double about every two years.

- This “law” has held pretty much since then; forecasters are predicting an end around 2025.

- If data grew so fast we couldn’t process it, there would be a harsher cap on the value of data, so this is intertwined with the importance of Big Data


Moore's Law
The Fifth Paradigm
Logarithmic Plot
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
1
10^{-2}
10^{-4}
10^{-6}
Calculations per Second per $1,000
Electromechanical
Relay
Vacuum Tube
Transistor
Integrated Circuit
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
Year

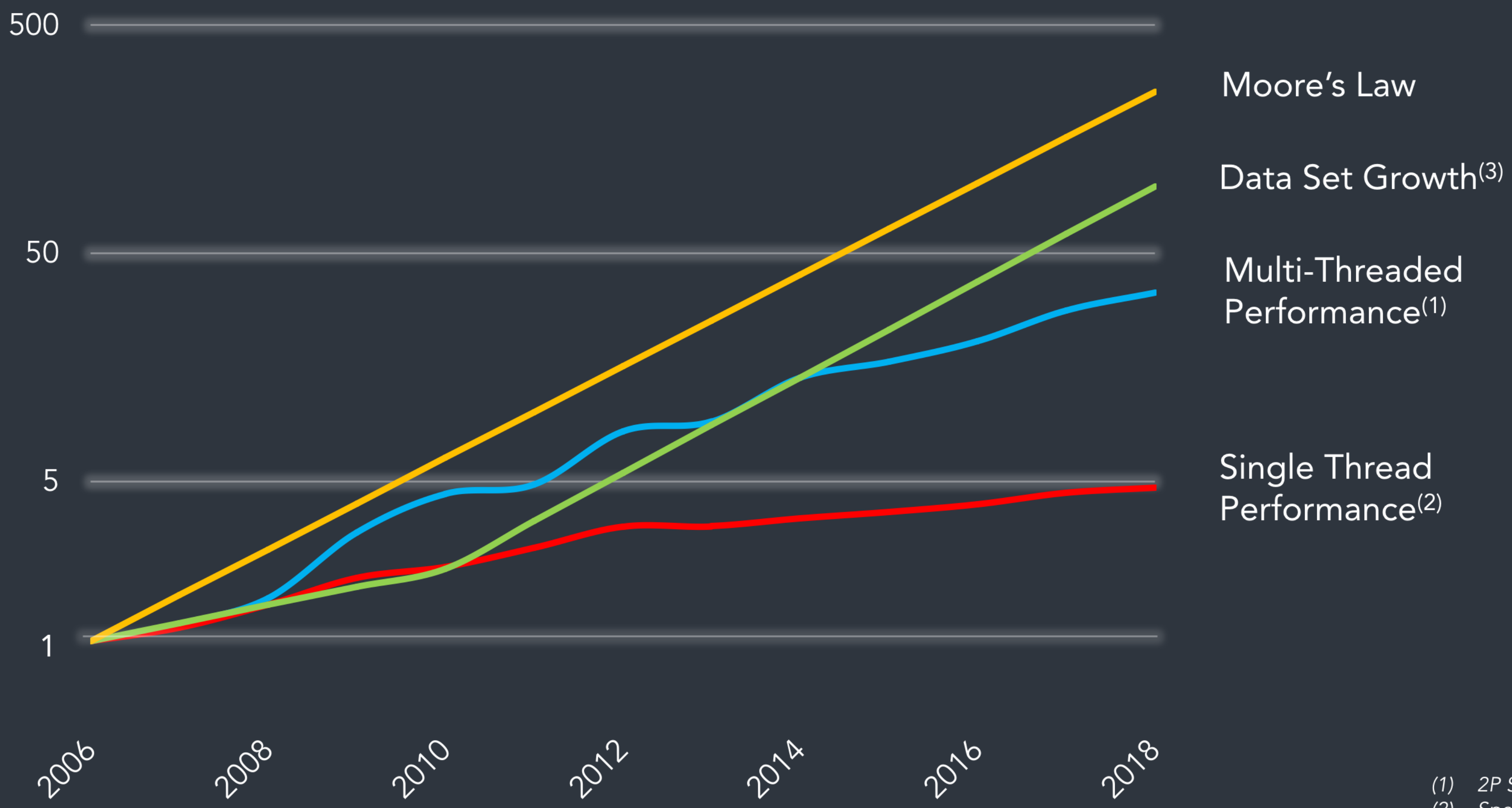

MIPS over time
140
120
100
80
60
40
20
0
MIPS
1965
1970
1975
1980
1985
1990
1995
2000
Year of Introduction



Transistors
Year
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
2005
2010
4004
8008
8080
6800
8086
68000
80286
68020
80386
68030
80486
68040
Pentium
601
603e
604
Pentium Pro
Pentium MMX
G3
604e
Pentium III
Pentium III
G4
Pentium 4
G5
Pentium M
Pentium M
Xeon
Itanium 2
Itanium 2
Itanium 2
Xeon
G6
Dual Core Itanium 2
Nehalen
Penryn
Quad Core
Core Duo
Pentium 4HT
Dual

- Maybe not over?
- https://www.technologyreview.com/s/614247/the-worlds-most-advanced-nanotube-computer-may-keep-moores-law-alive/
- **“The world’s most advanced nanotube computer may keep Moore’s Law alive”**
- **Video showing prediction vs reality: https://www.reddit.com/r/dataisbeautiful/comments/cynql1/moores_law_graphed_vs_real_cpus_gpus_1965_2019_oc**

Rate of CPU Performance Increase is Slowing



(1) 2P SPECint®_rate 2006
(2) Specint® 2006
(3) 62% CAGR

Scalable Computing

- Lots of data
- Relatively cheap to store
- Analyzing data has a lot of benefits
- However, for large amounts of data we need many computers and storage units
  - Need clusters of commodity computers

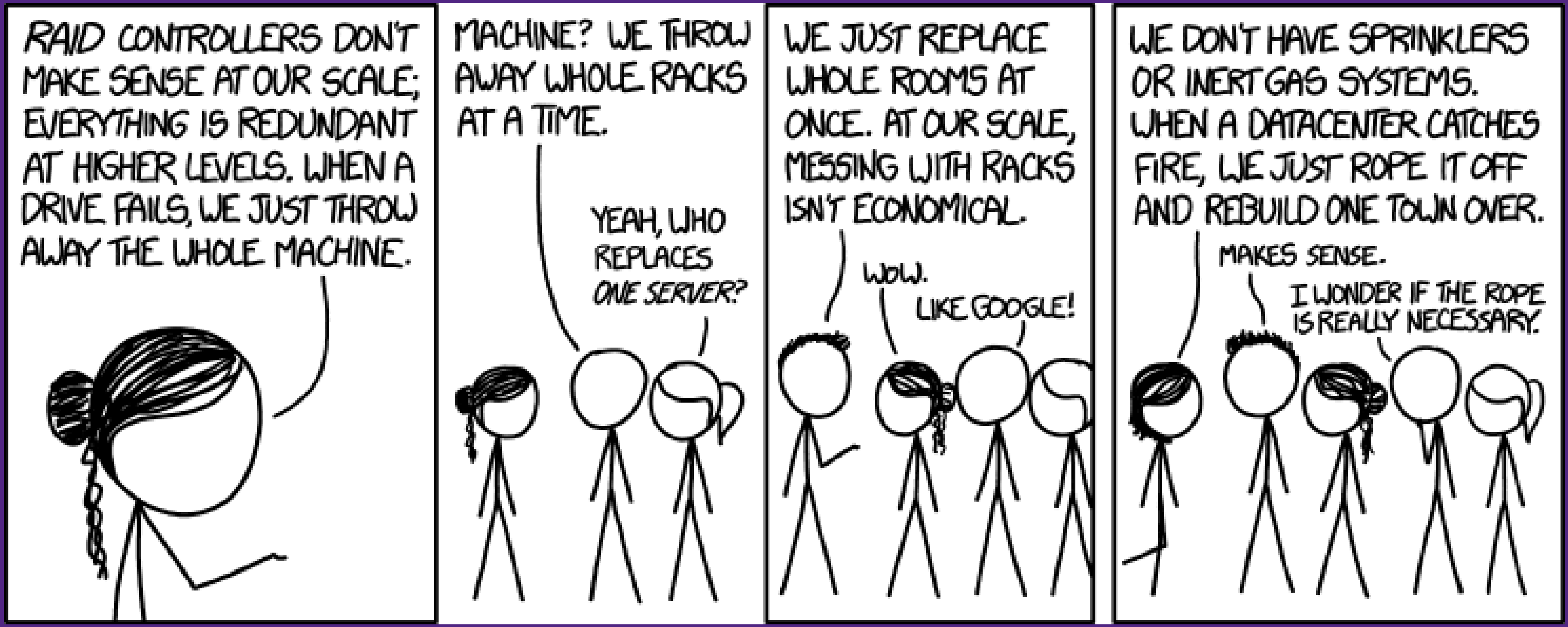
Processing Large Datasets

- Centralizing data processing will not work for huge amounts of data.
- Data and processing often needs to be distributed
- Processing platforms need to enable multiple tasks to be execute on different chunks of data

Processing Large Datasets

- How do we distribute computing tasks?
- How do we deal with the complexities of developing distributed software?
  - Data is distributed
  - Processing platforms need to enable multiple tasks to be executed on different chunks of the datasets
- What about failures?

Processing Large Datasets


RAID CONTROLLERS DON'T MAKE SENSE AT OUR SCALE; EVERYTHING IS REDUNDANT AT HIGHER LEVELS. WHEN A DRIVE FAILS, WE JUST THROW AWAY THE WHOLE MACHINE.
MACHINE? WE THROW AWAY WHOLE RACKS AT A TIME.
YEAH, WHO REPLACES ONE SERVER?
WE JUST REPLACE WHOLE ROOMS AT ONCE. AT OUR SCALE, MESSING WITH RACKS ISN'T ECONOMICAL.
WOW.
LIKE GOOGLE!
WE DON'T HAVE SPRINKLERS OR INERT GAS SYSTEMS. WHEN A DATACENTER CATCHES FIRE, WE JUST ROPE IT OFF AND REBUILD ONE TOWN OVER.
MAKES SENSE.
I WONDER IF THE ROPE IS REALLY NECESSARY.

https://xkcd.com/1737/

Simple, Large-scale computations

- Big data computation sounds fancy
- Mostly just counting stuff or adding stuff up
- TF-IDF
  - How many times does word appear in doc?
  - How many docs does a word appear in?
- Count-based language model
  - How often does “the” occur after “apple”?
- Neural network training
  - How often does my network make mistakes?

Example

- Let’s say that a retailer has a huge ledger with all of its sales representing stores in multiple cities

Date	City	Product	Price
2017-01-01	London	earrings	50
2017-05-01	Toronto	purse	150
2017-06-08	Ottawa	belt	50
2017-10-15	London	jacket	200

- You want to calculate the total sales per city

Map Reduce

Example

One task goes through each entry in the ledger in order to calculate the sales per city

Date	City	Product	Price
2017-01-01	London	earrings	50
2017-05-01	Toronto	purse	150
2017-06-08	Ottawa	belt	50
2017-10-15	London	jacket	200

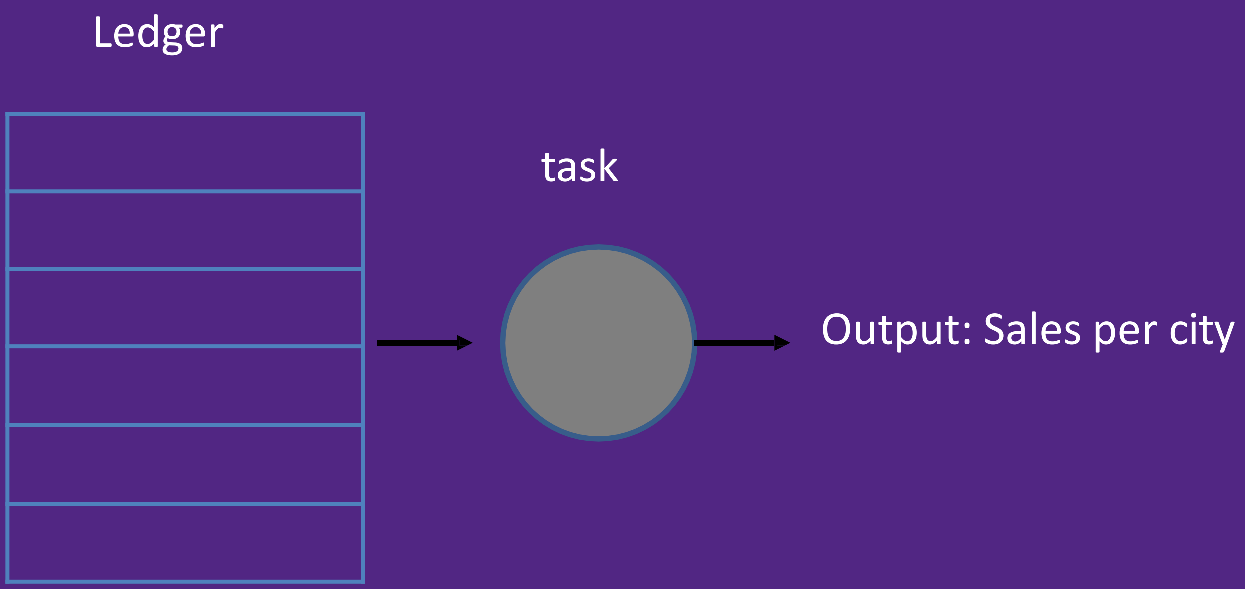
London 50

London 50, Toronto 150

London 50, Toronto 150, Ottawa 50

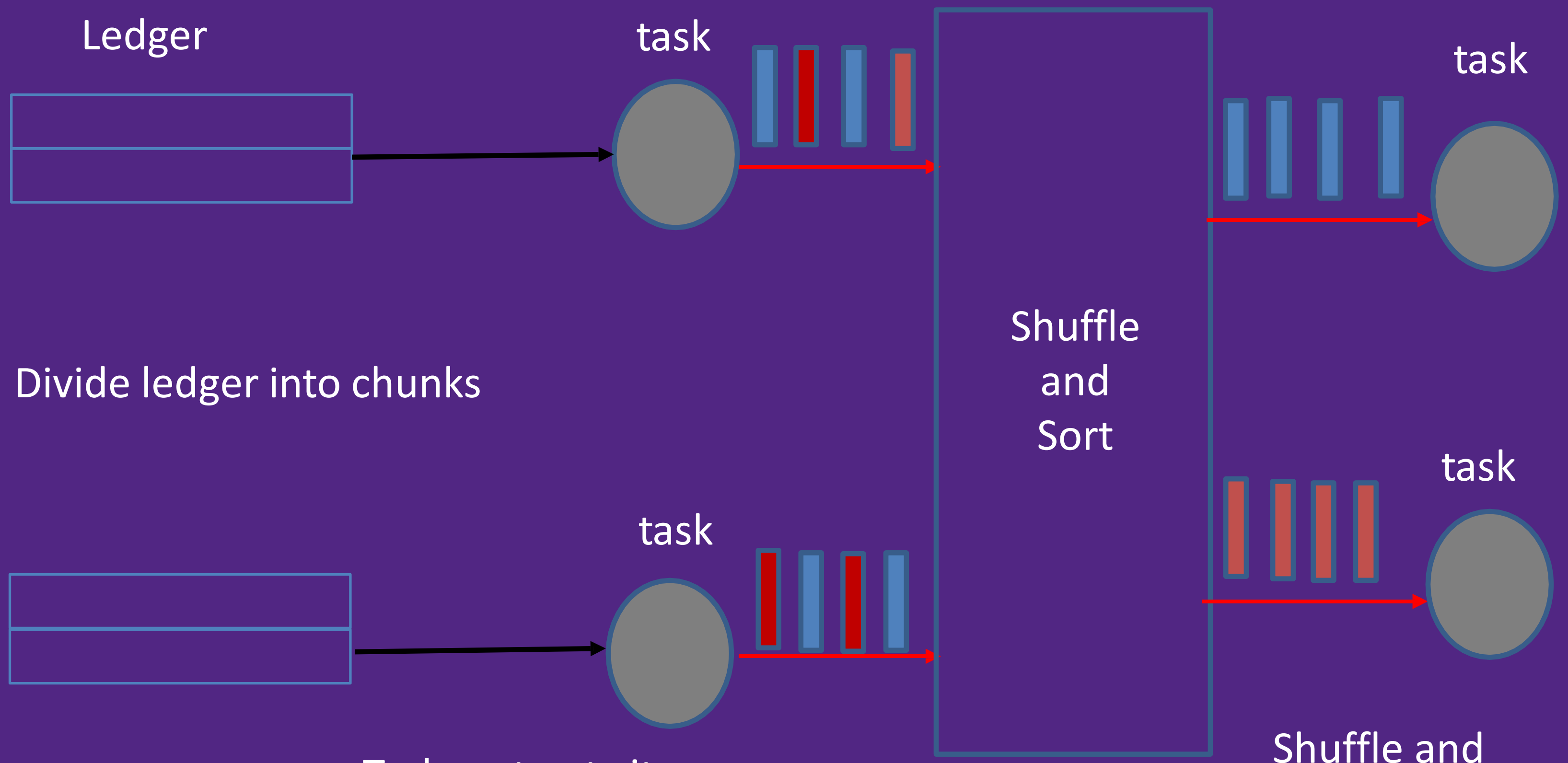
London 250, Toronto 150, Ottawa 50

Example



What if we could have multiple tasks running?

Example



Ledger

task

task

Shuffle
and
Sort

task

task

Divide ledger into chunks

Tasks extract city name and sales amount

Output: pairs (city, sales)

Shuffle and sort gives all pairs from same city to same task

Large Data Set Analysis

- Iterate over a large set of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate interim results
- Generate final output

Data Analytics in the Cloud

- Need a lot of servers
- These can come from a cloud provider

https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-the-cloud

Topics

- We will discuss a popular programming model and show examples of how it can be used.
- We will discuss the execution environment that includes a discussion of failure management

MapReduce

MapReduce History

- Google’s invention (2003)
- Became known with a 2004 paper

Example: MapReduce Applications

- Netflix: discover the most popular movies based on your viewing in order to provide suggestions
- LinkedIn: Discover who visited each member’s profile
- E-Commerce providers: Identify favorite products based on users’ interests or buying behavior.
  - Used by Amazon, Walmart, eBay

Example: MapReduce Applications

- Financial Industries: Fraud detection
- Search Providers: Ranking content
- Google Maps: Locating roads linked to a given intersection; finding nearest feature to a given address

MapReduce – What is it?

- *Programming model* for processing large data sets
- An *execution framework* that is able to run multiple tasks

MapReduce Overview

- MapReduce is highly scalable and can be used across many computers.
- Many small machines can be used to process jobs that normally could not be processed even by a large machine.

Before MapReduce

- Large scale data processing was difficult
  - Managing hundreds or thousands of processors
  - Managing parallelization and distribution
  - I/O scheduling
  - Status and monitoring
  - Fault/crash tolerance

- Programming models: MPI (Message-passing Interface)

Programming Model

- Programmers specify two functions
  - Map
  - Reduce
- Inspired from map and reduce operations commonly used in functional programming languages like Lisp
- Have multiple workers (processes) on multiple machines run either map or reduce

Map Operation

- Map: $(key_i,value_i) \text{ -> } (key_j,value_j)$
  - Input: A key/value pair
  - Output: A key/value pair
- Evaluation
  - – Function defined by user
  - Might need to parse input and extract relevant data
- Produces a new list of key/value pairs
  - – Can be of different type from input pair

Example

Reduce Operation

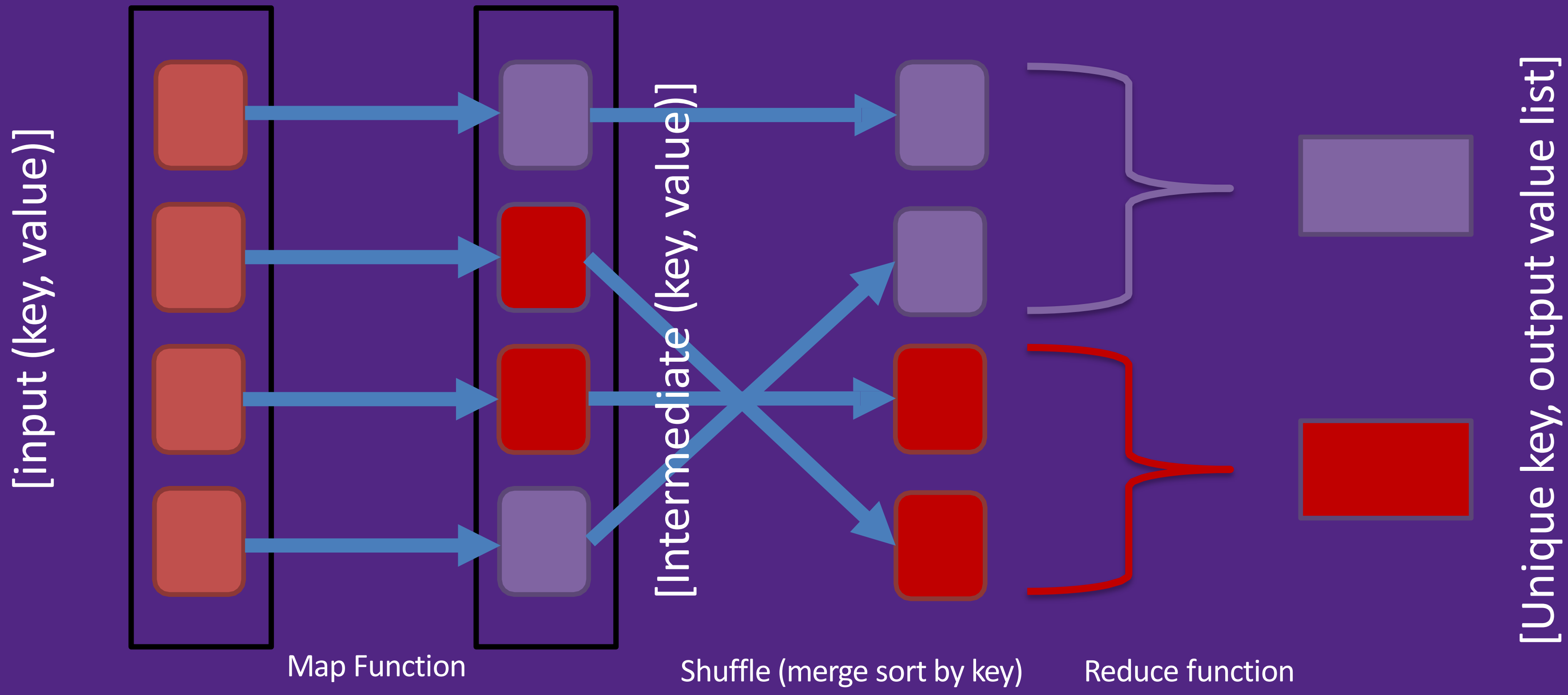
- Reduce: $(key_j,[val]_j)$ -> $[val_k]$

- All the intermediate values associated with each key_j produced by the mapper are combined together into a list, giving the pair $(key_j,[val]_j)$

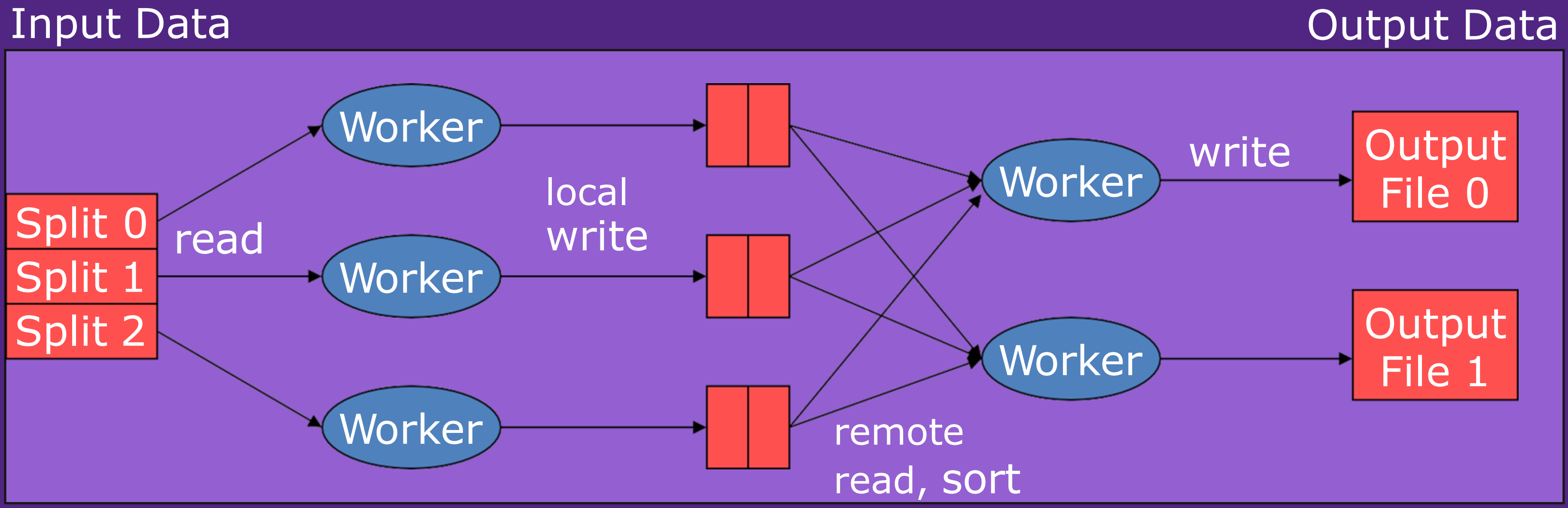
- Reduce function is applied to each of these pairs

Example

Programming Model


[input (key, value)]
[Intermediate (key, value)]
[Unique key, output value list]
Map Function
Shuffle (merge sort by key)
Reduce function

MapReduce Workflow



Map

extract something you care about from each record

Reduce

aggregate, summarize, filter, or transform

MapReduce Model

- The nice thing about the model is that a programmer writes the mapper code and the reducer code
- The Shuffle and Sort is handled by an environment like
  - Hadoop
  - Elasticsearch/Hadoop
  - MongoDB (but deprecated)
  - Riak

Summary

- Big data are big
- Distributed computing is required
- MapReduce is an elegant programming model relevant to processing big, unstructured or structured data