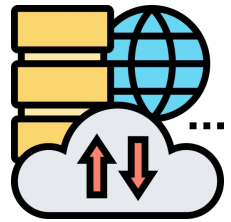
COEN 241
Introduction to Cloud Computing

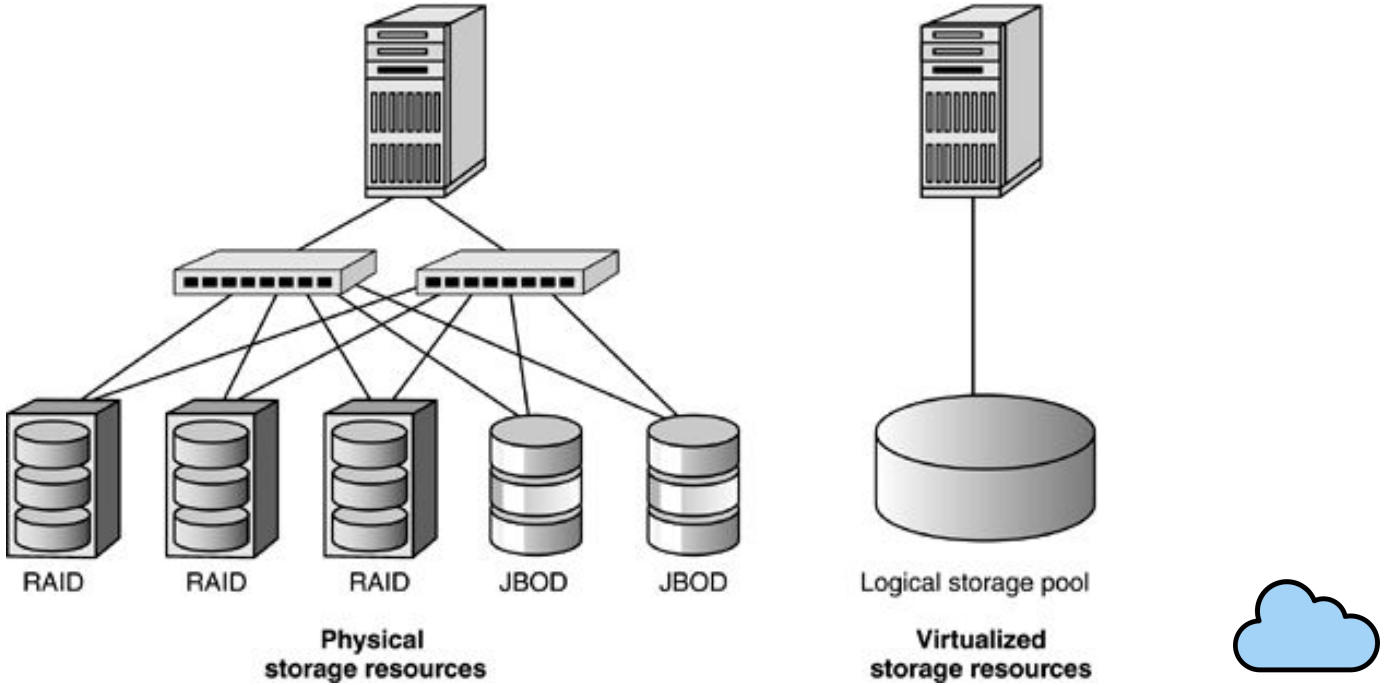
Lecture 15 - Big Data I



Lecture 14 Recap

- Final Presentation and Report Overview
- Storage Virtualization
- Cloud Storage
- Cloud Databases
- Readings
  - Recommended: CCSA Chapter 5,6,7
  - Optional: None

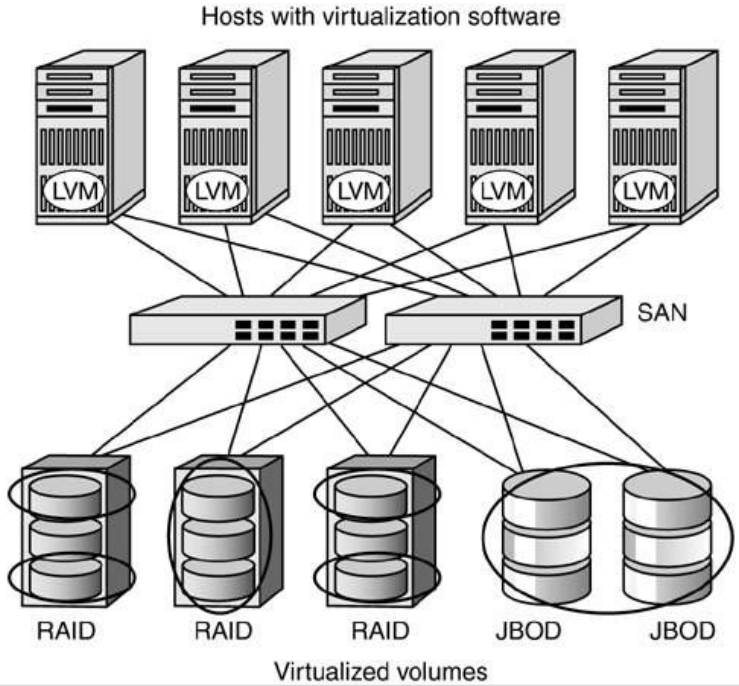
What is Storage Virtualization?



Example / Types of Storage Virtualization

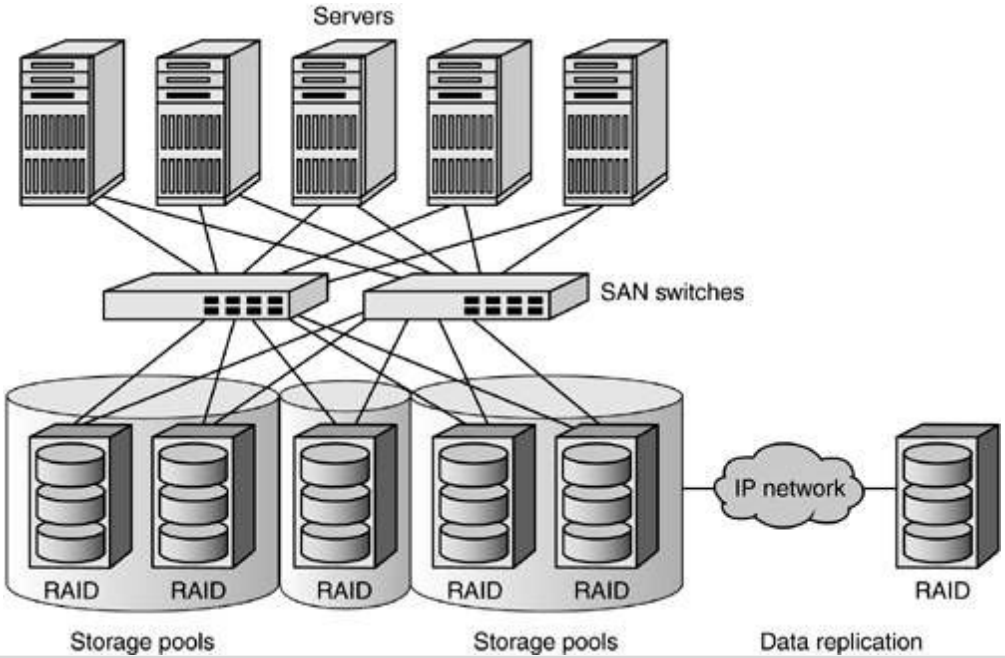
- Host based
  - Running additional software on each host
- Storage device based
  - Creating disk arrays of storages for virtualization
- Network based
  - Combining storage over a computer network

Host-based Storage Virtualization

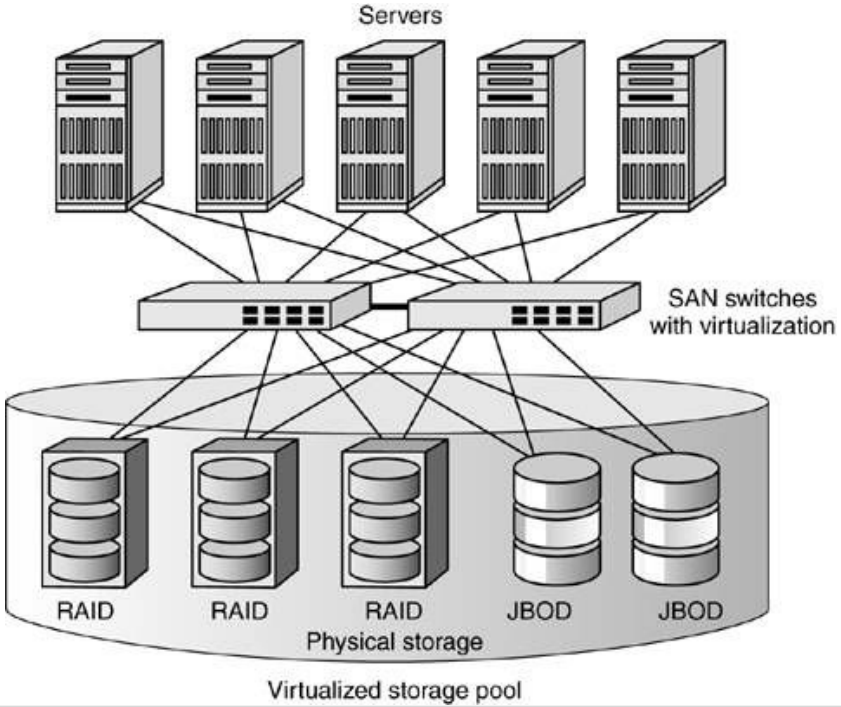


*SAN: Storage Area Network

Storage Device Based Storage Virtualization



Network Based Storage Virtualization



What is Cloud Storage?

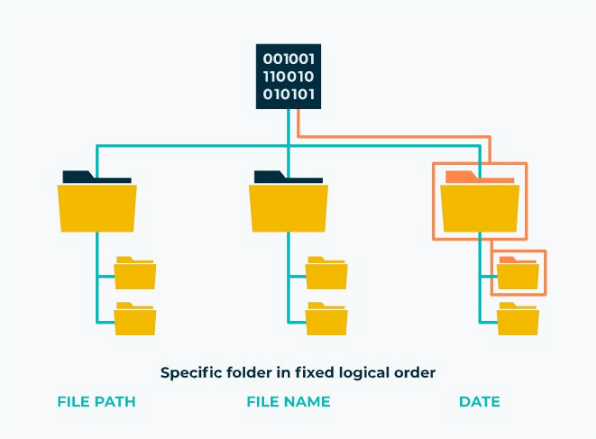
- Model of computer data storage in which the digital data is stored in logical pools over multiple servers in the internet (in the “cloud”)
- Servers are mostly managed by storage / cloud providers
  - Responsible for making storage available and accessible
  - Provides security to the data
- Data is accessed via APIs or web-based content management system

Types of Cloud Storage

- File Storage
  - Data stored as files
- Block Storage
  - Data stored as blocks in separate pieces
- Object Storage
  - Data is broken into pieces and spread out among hardware

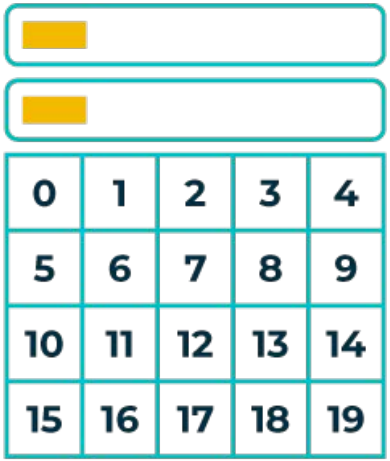
File Storage

- The traditional, old-school, approach to storage.
  - Each file gets a name
  - Store files in folders/directories and sub-directories
- Use Cases
  - File sharing / Collaboration
  - Backup and Recovery
- Pros
  - Easy to understand and manage at small scale
  - Users can manage their own files
- Cons
  - Hard and expensive to manage at larger scale
  - Hard to work with unstructured data



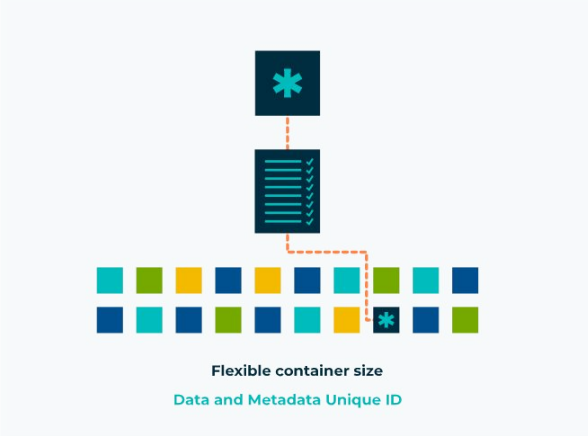
Block Storage

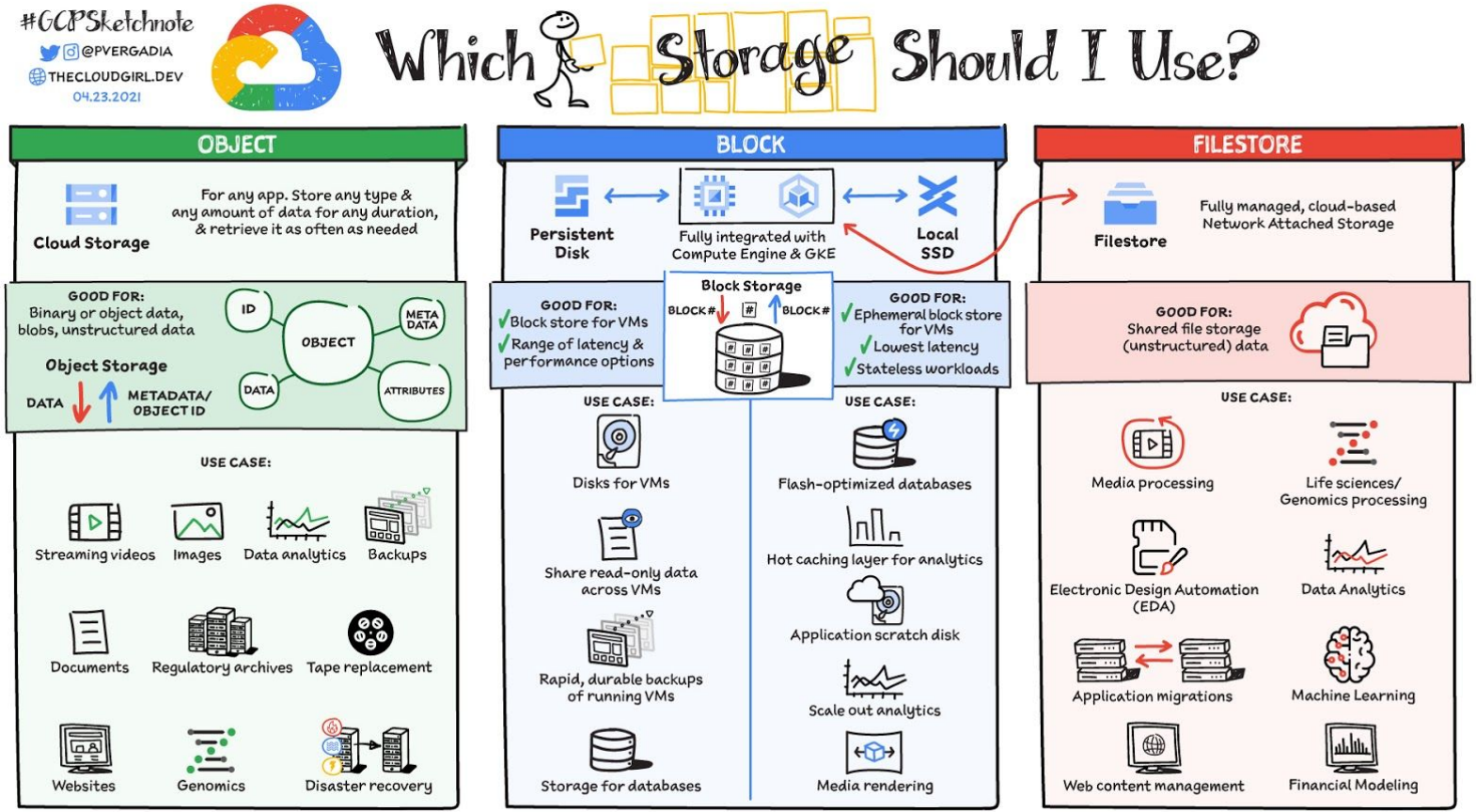
- Data split into fixed blocks of data and store separately
  - Blocks are given an ID
  - Reassembled at retrieval
- Use Cases
  - Databases
  - Virtual machine file system
- Pros
  - Fast and reliable
  - Easy to modify per block
- Cons
  - Lack of metadata, less usable for unstructured data
  - Not searchable and expensive



Object Storage

- Divides data into self-contained units stored in flat environment
  - All objects are at the same level, no sub-directories
  - Data is split up and requires metadata to access
- Use Cases
  - Store unstructured data (even large data set)
  - Store database dumps and logs
- Pros
  - Unlimited Scalability (Best for Machine Learning)
  - Easy to search
- Cons
  - Cannot change the object once created
  - Slow performance




#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
04.23.2021
Which Storage Should I Use?
OBJECT
Cloud Storage
For any app. Store any type & any amount of data for any duration, & retrieve it as often as needed
GOOD FOR:
Binary or object data, blobs, unstructured data
Object Storage
DATA
METADATA/ OBJECT ID
ID
META DATA
OBJECT
DATA
ATTRIBUTES
USE CASE:
Streaming videos
Images
Data analytics
Backups
Documents
Regulatory archives
Tape replacement
Websites
Genomics
Disaster recovery
BLOCK
Persistent Disk
Fully integrated with Compute Engine & GKE
Local SSD
Block Storage
BLOCK#
BLOCK#
GOOD FOR:
Block store for VMs
Range of latency & performance options
GOOD FOR:
Ephemeral block store for VMs
Lowest latency
Stateless workloads
USE CASE:
Disks for VMs
Share read-only data across VMs
Rapid, durable backups of running VMs
Storage for databases
USE CASE:
Flash-optimized databases
Hot caching layer for analytics
Application scratch disk
Scale out analytics
Media rendering
FILESTORE
Filestore
Fully managed, cloud-based Network Attached Storage
GOOD FOR:
Shared file storage (unstructured) data
USE CASE:
Media processing
Life sciences/ Genomics processing
Electronic Design Automation (EDA)
Data Analytics
Application migrations
Machine Learning
Web content management
Financial Modeling

Types of Databases

- SQL
  - Traditional vs Cloud
  - Self-managed vs Fully Managed
- NoSQL
  - Key-value store
  - Document
  - Columnar
  - Graph

SQL Database on Cloud

- Amazon RDS, Cloud SQL, Azure SQL
  - Fully-managed Relational Database Services
  - Scales horizontally and vertically automatically
- Amazon Redshift
  - Columnar database
  - Automatically scales to petabytes
  - Supports SQL for analytical queries
- Amazon Athena, Google BigQuery (?)
  - Runs SQL over Object Storage
  - Used for queries over large data
  - Serverless

NoSQL Database

- Databases that do not support SQL
- Various ways to read data
- Examples: MongoDB, Cassandra, Neo4J, HBase
- What is NoSQL Database good for?
  - Unstructured Data
  - Hierarchical data storage
  - Fast insertion, large amount of data
  - Simple queries

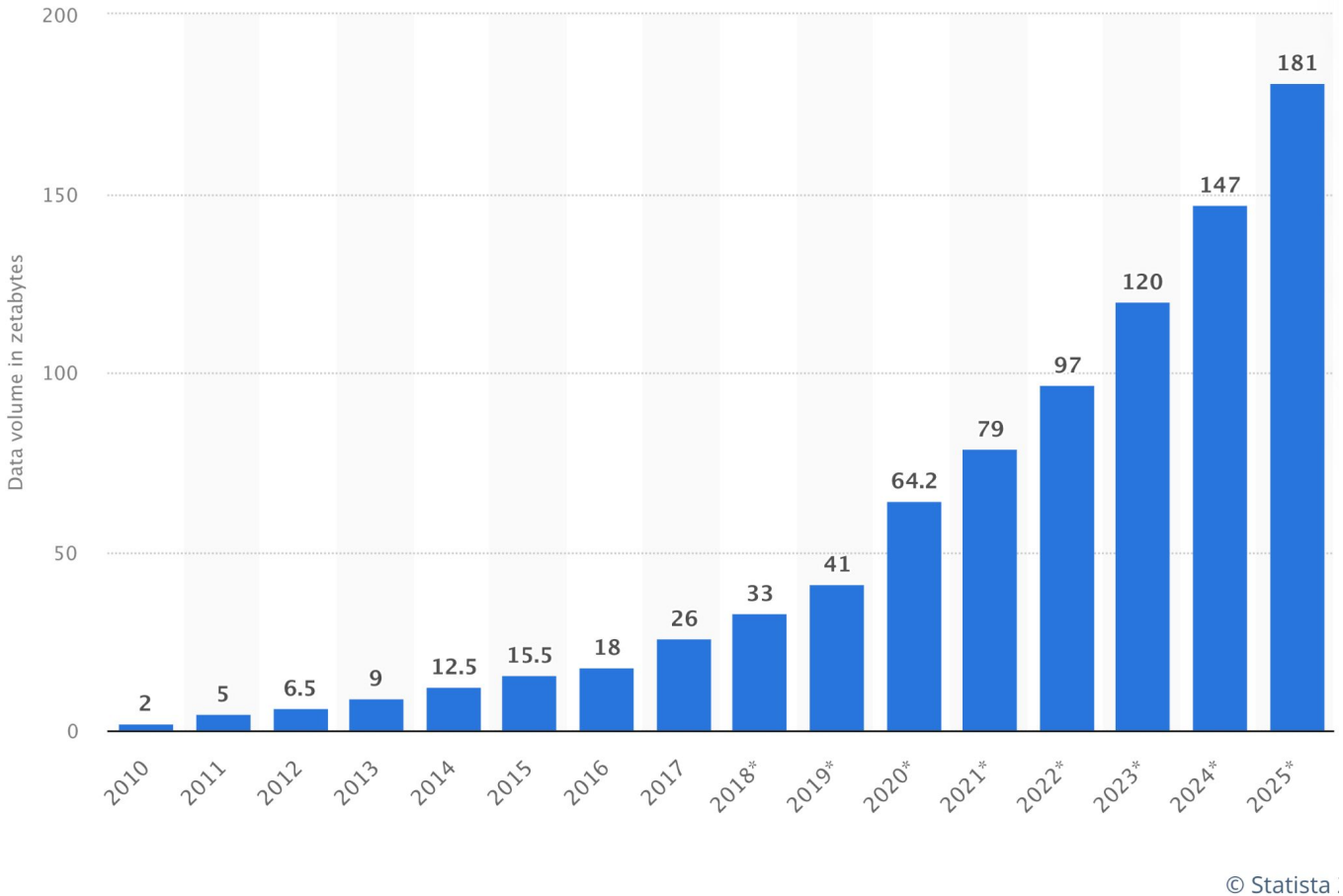
NoSQL Database on Cloud

- Amazon DynamoDB, Google BigTable, Azure CosmoDB, MongoDB
  - Key-value
  - Document store
    - Data stored in loosely structured format
- Cassandra and Hbase (Managed on EMR)
  - Wide column store
  - Schema free
- AWS Neptune
  - Graph Database
  - Best for storing relations

Agenda for Today

- Big Data
  - What is Big Data
  - Big Data Platforms
- Online Data Processing
  - Message Queue Systems
  - Online Analytics
- Readings
  - Recommended: CCSA Chapter 8, 10.1~10.8
  - Optional: None

Big Data


Data volume in zetabytes
200
150
100
50
0
2
5
6.5
9
12.5
15.5
18
26
33
41
64.2
79
97
120
147
181
2010
2011
2012
2013
2014
2015
2016
2017
2018*
2019*
2020*
2021*
2022*
2023*
2024*
2025*
© Statista

What is Big Data?

- What do you think big data is?
- Is it only determined by size?
- What other factors determine what a big data is?

Interesting Facts about Data

- From the beginning of time until 2003, the entire world only had five billion gigabytes of data
- The same amount of data was generated over only two days in 2011
- By 2013, this volume was generated every ten minutes
- 90% of all the data in the world has been generated in the past few years
  - This is accelerating as well!

What is Big Data?

- Definition: Data that contains greater **variety**, arriving in increasing **volumes** and with more **velocity**.
  - **Three Vs**
- Variety: Many types of data that are available
  - Traditional data types were structured and fit neatly in a relational database
  - Now, data comes in various unstructured formats
- Volume: Amount of data, which varies across the users
  - ~ terabytes for some, hundreds of petabytes for others
- Velocity: Rate at which the data arrives
  - Data often arriving in streams

What is Big Data?

- More recently there are two new ‘V’s that emerged
- Value: Data has intrinsic value, but not useful until you analyze it
  - Need good methods and **systems** for big data analytics
- Veracity: How truthful is your data?
  - Garbage in - Garbage out

Digression: History of Big Data

- Origin of large data goes back to 1960~70s
  - Emerged along with relational databases
- Big data became a hot topic from 2005 with Facebook, Google, ...
  - Hadoop invented the same year
  - NoSQL started gaining popularity
  - Cassandra: 2008, MongoDB: 2009
- Cloud computing allows scaling big data analytics to another level

What can we do with Big Data?

- Store it nicely
  - Storing and organize data is an asset!
- **Analyze** it to find interesting patterns
  - Healthcare
    - Outbreaks of pandemic
    - Evidence-based medicine
  - Ads / Recommendations
    - User preferences
  - Weather Forecast
  - Self-Driving and many many many more to come

Cloud Computing's Role in Big Data

- Cloud Computing allows Big Data to be available, scalable and fault-tolerant
- Hard to deploy & manage clusters for Big Data storage and analytics
- Cloud computing provides
  - Agility
  - Elasticity
  - Cost saving
  - Reduced complexity
  - **Platforms for big data processing**

Big Data Platforms

Big Data Platforms

- Storage (We have covered this)
  - Distributed SQL/NoSQL Databases
  - OLAP: Redshift, BigQuery
- Offline (Batch) Analytics
  - **MapReduce**
    - Hadoop
    - Spark
    - Pig, Hive
- Online (Real-Time) Analytics
  - Apache Storm
    - **Message Queue**

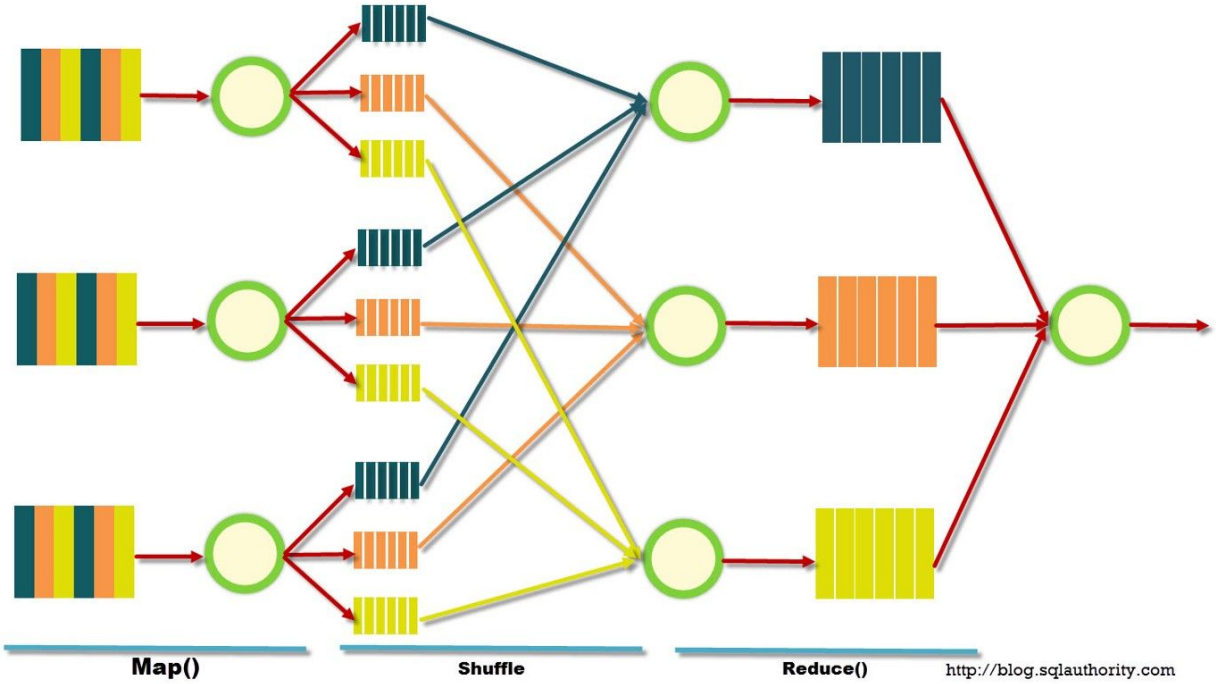
Online vs Offline Analytic System

- Online: Real-time, interactive workloads where data is ingested, stored and analyzed. Addressing **Velocity** in Big Data.
- Offline: Retrospective, sophisticated analyses that may touch most or all of the data. Addressing **Volume** in Big Data.
- Both types of systems can integrate with one another
  - E.g., Online system gathers, processes and stores data for offline system
- Most companies run both types of analytics

MapReduce Introduction

- Definition: A **programming model** and an associated implementation for processing and generating big data sets in a parallel manner
- A MapReduce program comprises of three steps:
  - Map: Filtering and sorting of data
  - Shuffle: Worker nodes redistribute data based on the output keys
    - All data with the same key goes to same worker
  - Reduce: Performing a summary operation for data with the same key
    - Counting

MapReduce Introduction



MapReduce Introduction

- MapReduce is for addressing the **Volume** in Big Data
- What MapReduce allows
  - Easy distributed computing
  - Simple algorithms
  - Ability to run on commodity servers
- What MapReduce is NOT
  - Fast compared to traditional programming models
  - Low overhead
  - Built for single server

MapReduce 5-Step Computation

- Prepare input: Each map processor gets assigned the input key K1 that each processor would work on, gets all the input data associated with that key.
- Map: Map function is run exactly once for each K1 key, generating output organized by key K2
- Shuffle: Each reduce processors gets assigned the K2 key each processor should work on, gets all the Map-generated data associated with that key.
- Reduce: Reduce function is run exactly once for each K2 key.
- Produce the final output: Collects all the Reduce output, and sorts it by K2 to produce the final outcome.

MapReduce Example

- What does this program do?

```
#!/usr/bin/env python
"""mapper.py"""
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

MapReduce Example

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word, word = None, None
current_count = 0

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word,
current_count)
        current_count = count
        current_word = word

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

MapReduce Example

- Running your MapReduce Program

```
$  echo "foo foo quux labs foo bar quux" | mapper.py | sort -k1,1 | reducer.py
bar     1
foo     3
labs    1
quux    2
```

MapReduce System (1): Hadoop

- Hadoop
  - Collection of Open-Source software for running MapReduce
    - Started in 2006
  - Hadoop Distributed File System: Used for input, output files. Block Storage
  - Hadoop YARN: Resource manager
  - Hadoop MapReduce: Implementation of MapReduce
- Pros
  - Easy to run, scale MapReduce
  - Cheap and Fast
- Cons
  - Not fast enough
  - Slow for small files

MapReduce System (2): Spark

- Spark
  - Another Open-Source software for running MapReduce
    - Started in 2014
  - Requires a distributed storage, such as HDFS, S3 and so on.
- Pros
  - 100x faster than Hadoop for smaller workloads
  - Ideal for real-time processing
- Cons
  - Costly, requires machine with larger memory

MapReduce Implementation Examples

- https://spark.apache.org/examples.html
- PageRank in Hadoop
  - https://medium.com/swlh/pagerank-on-mapreduce-55bcb76d1c99

MapReduce Systems on the Cloud

- Amazon Elastic MapReduce
  - On-demand processing power
  - Auto-scaling
  - Easy to set up
- Google MapReduce
- Azure Serverless MapReduce via Durable Functions
  - https://docs.microsoft.com/en-us/samples/azure-samples/durablefunctions-mapreduce-dotnet/big-data-processing-serverless-mapreduce-on-azure/

MapReduce Add-Ons

- Apache Pig
  - Runs MapReduce using a language called PigLatin

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
filtered_words = FILTER words BY word MATCHES '\\w+';
word_groups = GROUP filtered_words BY word;
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;
ordered word count = ORDER word count BY count DESC;
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

Apache Pig

MapReduce Add-Ons

- Apache Hive
  - Supports SQL on Hadoop and Spark
  - Similar in usage to AWS Athena
- Apache Sqoop
- Apache Oozie
- Apache Storm
- Many more...

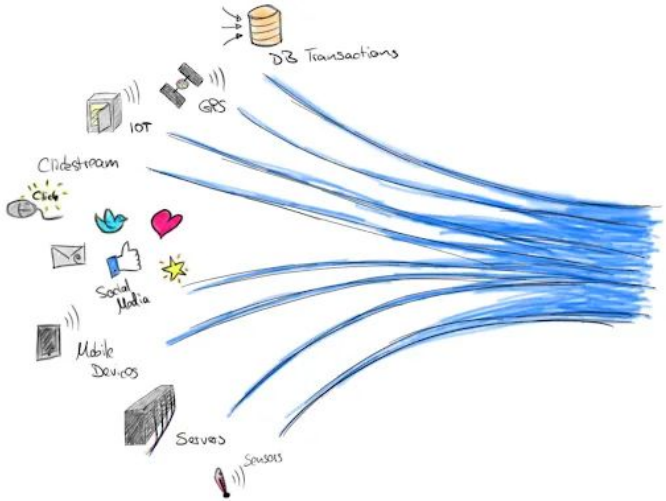
When to use MapReduce?

- MapReduce is not a one-stop solution
  - Often need to rethink your algorithm
  - Require a large cluster
  - May cost a lot more
  - May even be slower, each MapReduce step is costly
- Criteria for using MapReduce
  - **Easy to parallelize**
  - Big Data
  - If intermediate outputs are useful

Online Data Processing

How to Address the Velocity?

- We now have data coming in “Streams”
  - Data that is generated continuously from thousands of data sources
  - E.g., IoT devices / sensors continuously sending data



How to Address the Velocity?

- Cannot just store all the data as it comes in
  - Lot of garbage can come in
  - Storage is costly
  - May be just too much data to handle
- Need for a system to receive the data streams, order the data and process them nicely

How to Address the Velocity?

- There are now multiple solutions to handle the velocity issue
- Ingestion Mechanisms
- Stream Processing Frameworks
- Data Analytics Framework

TODOs!

- HW 3
- Quiz 4 will be out next week
- Final Project

Agenda for Today

- Big Data
  - What is Big Data
  - Big Data Platforms
- Online Data Processing
  - Message Queue Systems
  - Online Analytics
- Readings
  - Recommended: CCSA Chapter 8, 10.1~10.8
  - Optional: None

Questions?