Big Data Foundations

- Sai Kiran Krishna Murthy

Agenda

- Day 1: Introduction
- Day 2: HDFS, Kafka, Spark
- Day 3: Practical Showcase
- Day 4: Free day

What is Big Data?

The 5 V’s of Big Data

- Volume
- Velocity
- Variety
- Veracity
- Value

Big Data is a Paradigm shift

- Schema on write vs Schema on read
- Scale up vs Scale out
  - Horizontal Scaling vs Vertical Scaling
  - Single Node vs Cluster Computing
- Specialized vs Commodity hardware

Schema on write

- Create table
- Write data (validation happens here)
- Read data

Schema on read

- Create file (Write data)
- Read data (validation happens here)

Scale up vs Scale out

Scale up

- Get a bigger machine
  - More RAM
  - Bigger CPU/GPU
  - More Cores

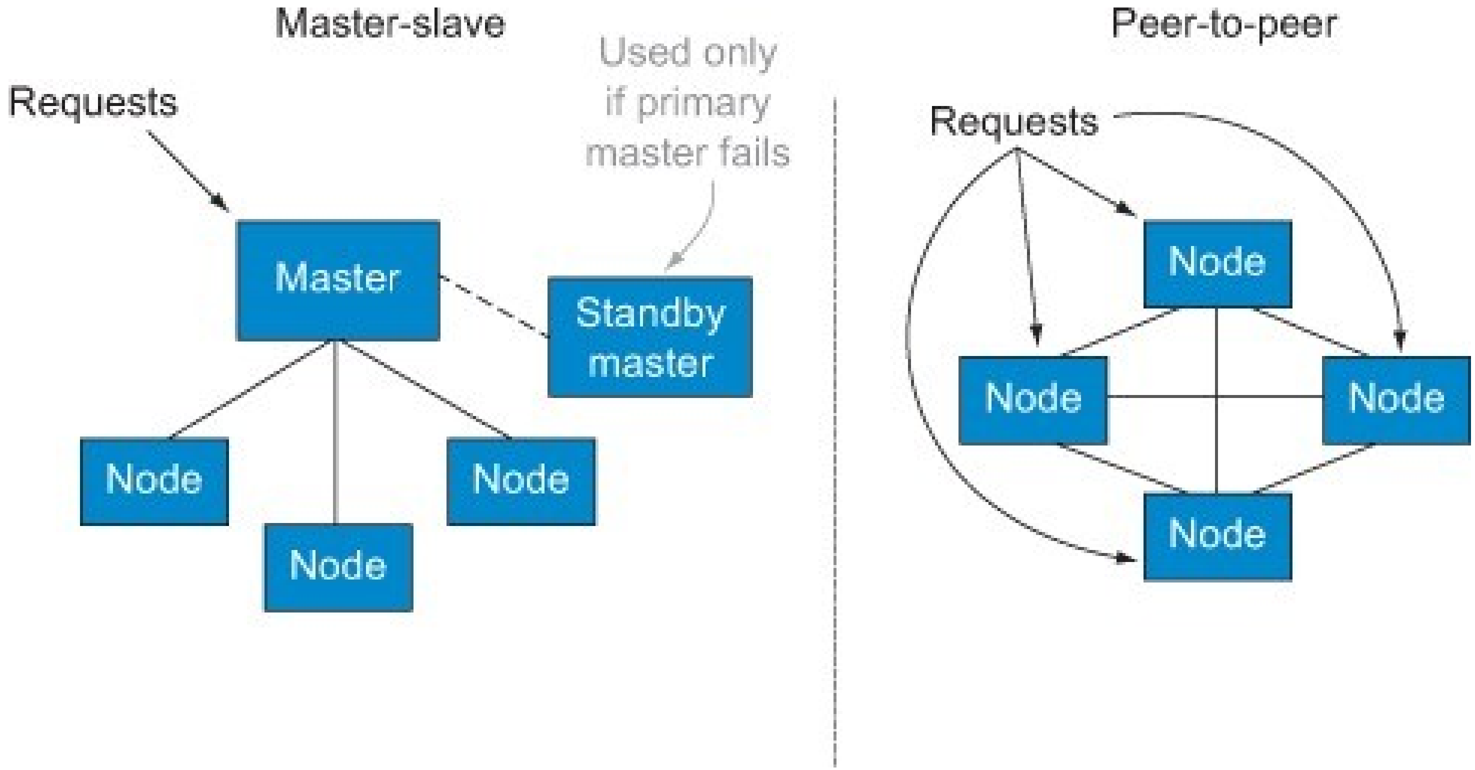
Problems?

- Increasing costs
- Physical Limitation
- Specialized software? → Vendor Lockin

Alternate?

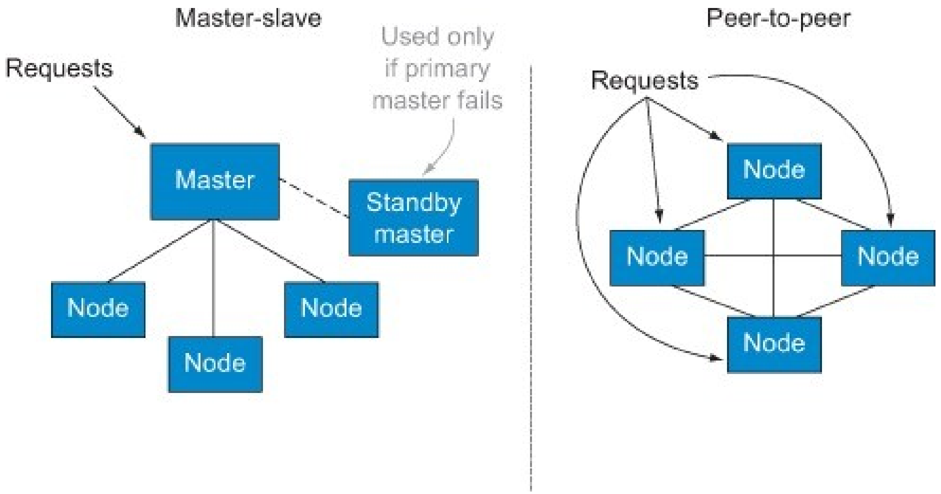
- Scale out:
  - add more machines (nodes)
  - Create a cluster of these machines (nodes)

Scale out: In practice



Features

- Concurrency
- Scalability
- Availability
- Fault Tolerance



Case Study

Problem Statement

- Most of Scrooge’s wealth is in his Gold Coins
- Total wealth ~= Number of Gold Coins * Cost of each gold coin

Introducing: The Counter

- He counts gold coins for a living

Current Approach

- Identify gold coins from the vault
- Move the gold to where the counter is
- Let the counter do the counting and give us the results
- Currently we have 1 counter

- Total time taken
  - Sorting and Filter time
  - Transportation time
  - Counting time

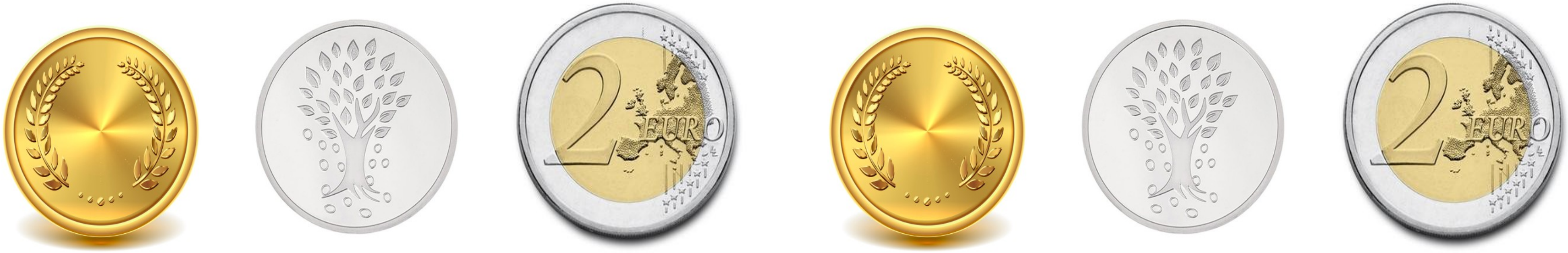
How can we Improve?

Aspects we can improve?

- Storage Layout: How we store?
- Locality: Where we count?
- Processing Paradigm: How we count?

Storage Layout

Scenario #1: Row based



Discussion #1

- Advantages and disadvantages

Storage Layout

Scenario #2: Column based

Discussion #2

- Advantages and disadvantages

Storage Layout

Scenario #3: Hybrid

2 EURO
2 EURO

Locality

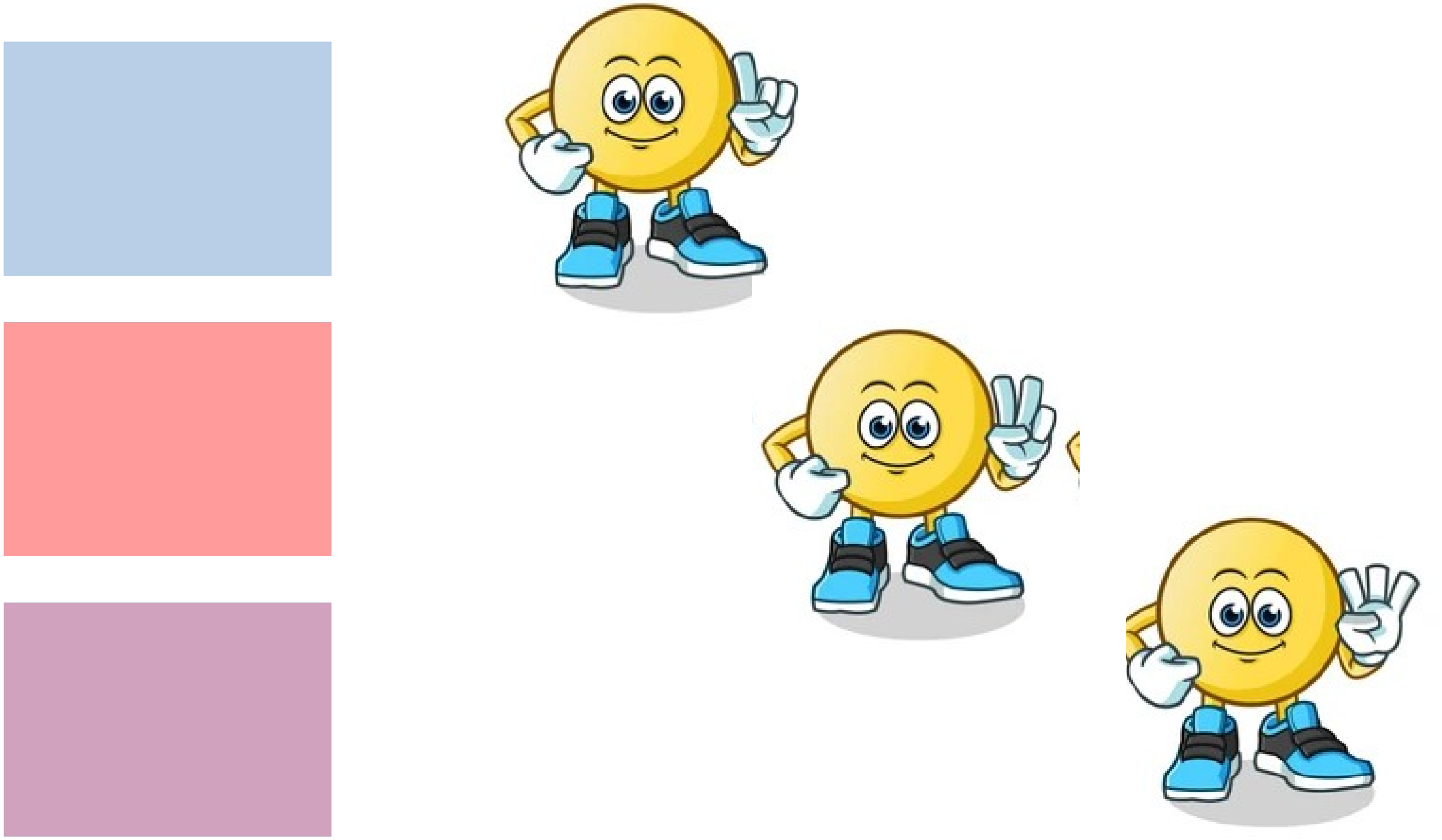
- Move Gold to where the Counter is?
- Move the Counter to the Vault?

How we count? (Processing Paradigm)

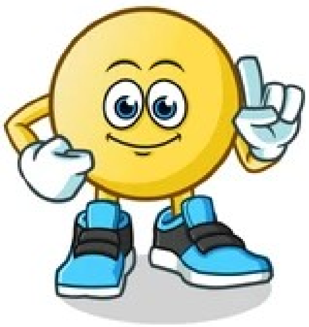
- Divide and Conquer
  - Divide the data
  - Add more counters
  - Let the counters work in parallel

Bucketing / Partitioning

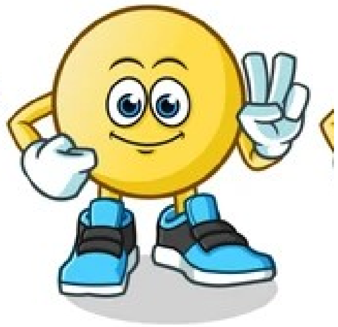
Mapping



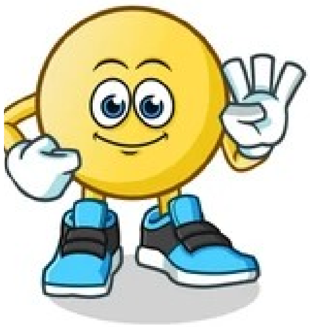
Reducing



+



+



= 6

Map + Reduce

- Split a big problem in smaller pieces
- Map each task to a counter (worker)
- Reduce the results of each worker, until you have only one result

Discussion

- What happens with too few partitions?
- What happens with too many partitions?
- What is the ideal number of partitions?

Discussion

- What happens with too few counters?
- What happens with too many counters?
- What is the ideal number of counters?

Examples

Exercise

- Take 10 mins and come up with an example problem you can solve with map reduce
- Present it to the group

Wait a minute

- Were we talking about Big Data all along?
  - Gold == Data
  - Scrooge has a lot of data (aka volume) == Big Data
  - Counter == A program
  - Vault == Storage
  - MapReduce == processing framework
  - Different Binary File formats == avro, parquet

Recap Lessons learned

- Data Locality : Move the program to where the data resides
- Program Paradigm: Map Reduce
- Data Storage: Depending on the use case, Row Vs Column Storage

Introducing Hadoop

- Hadoop = Storage + Processing Framework + Scheduler
  - Storage = HDFS
  - Processing Framework = Map Reduce
  - Scheduler = Yarn

Note on Map Reduce

- Map Reduce has been replaced with Apache Spark

Putting it all together

- Storage = HDFS
- Processing Framework = Spark
- Scheduler = Yarn
- Synchronization = Zoo Keeper
- SQL interface for HDFS = Hive
- NoSQL Database = HBase
- Message Queue = Kafka
- Notebook = Zeppelin

CDP: Walk through

Extra slides

Avro file format

- Rich data types
- Row based format
- Binary
- Splittable

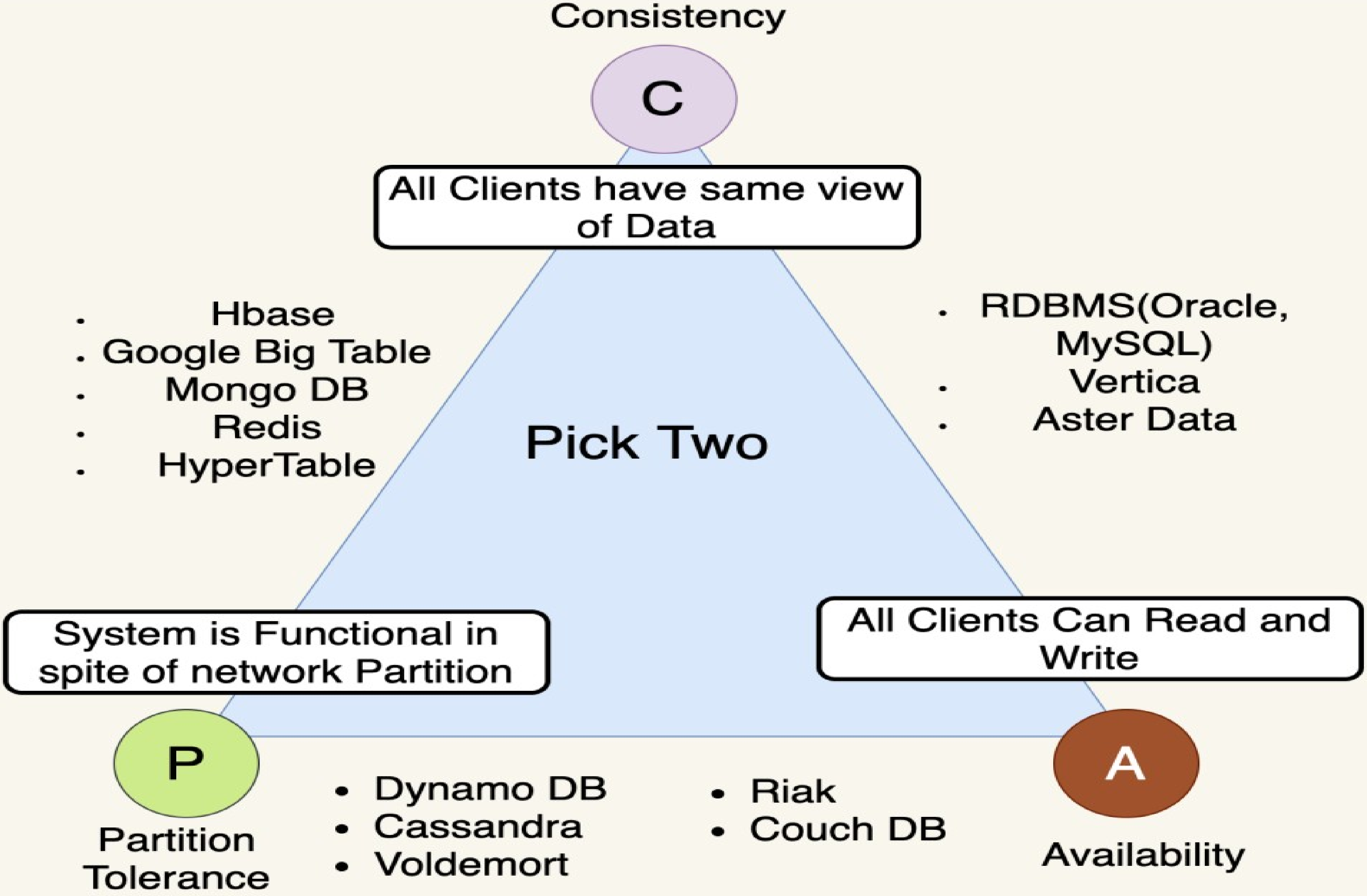
Introducing CAP theorem

C – Consistency

A – Availability

P – Partition Tolerance

Distributed System? You can only have two



Consistency

C

All Clients have same view of Data

- Hbase
- Google Big Table
- Mongo DB
- Redis
- HyperTable

Pick Two

- RDBMS(Oracle, MySQL)
- Vertica
- Aster Data

System is Functional in spite of network Partition

All Clients Can Read and Write

P

Partition Tolerance

- Dynamo DB
- Cassandra
- Voldemort

- Riak
- Couch DB

A

Availability