Comprehensive Big Data Solution for Stock Prices and Tweets Collection

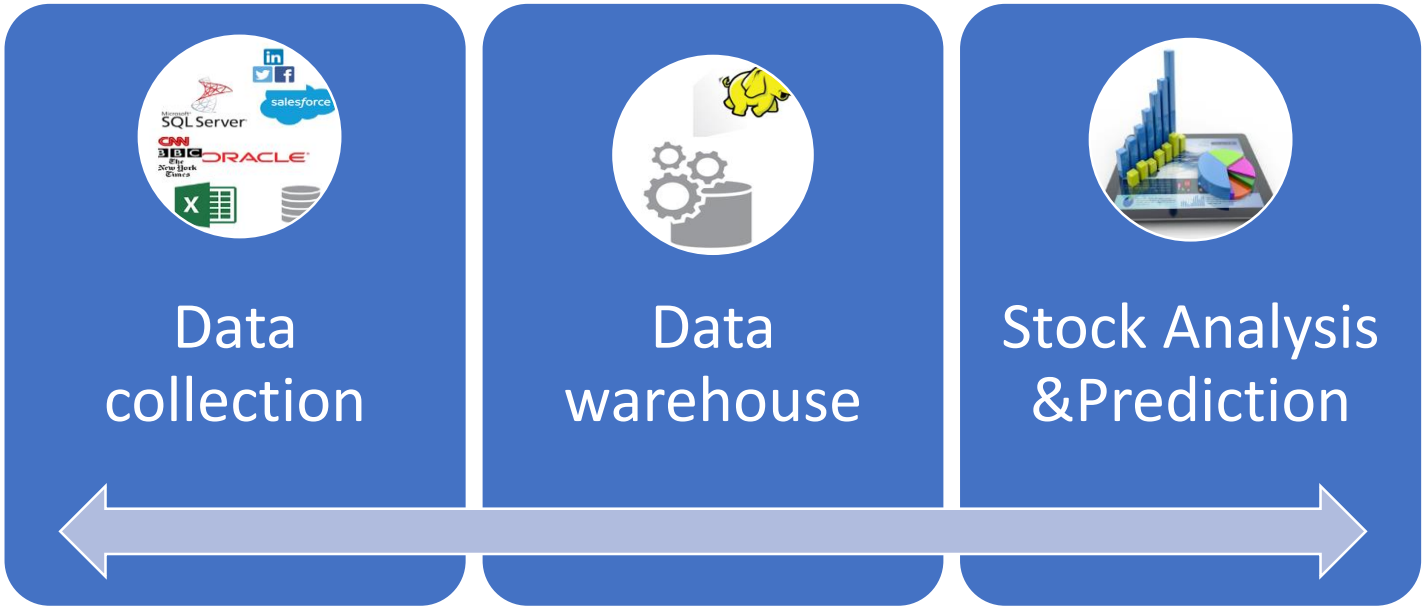
Paul Adams

Rikel Djoko

Stuart Miller

Problem

- Predicting Markets Using Big Data
  - Quick Reads Needed
    - Data-parallelism
    - Task-parallelism

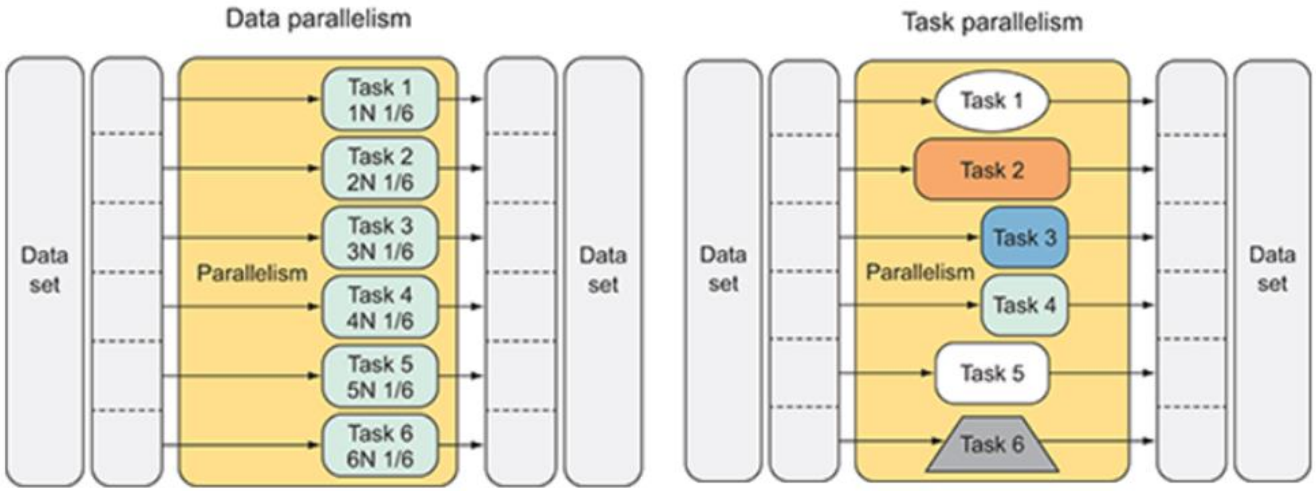


Data

- Gathered via API from Twitter, Markets
  - JSON collected using Python
  - CSV collected using R
- Large Volumes
  - Increasing rapidly
  - Horizontal scaling needed
  - Sharable across networks

Big Data Solution: Parallelism

- Data-Parallelism
  - Different data processed together
- Task-Parallelism
  - Different tasks performed together



Images from
https://livebook.manning.com/book/concurrency-in-dot-net/chapter-7/10

Big Data Solution: MapReduce

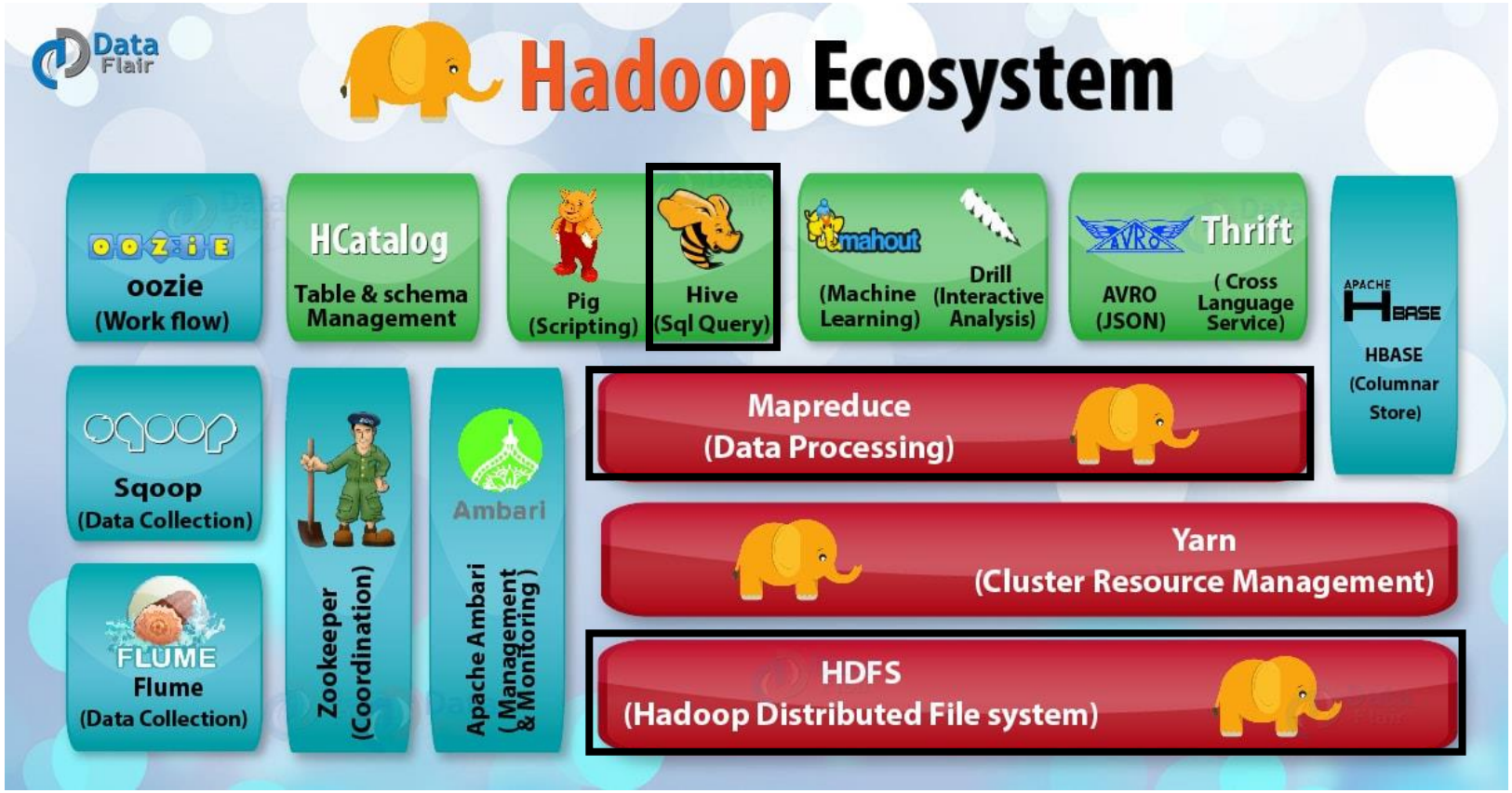
- Hadoop MapReduce
  - Hadoop Distributed File System
  - Manage data-parallel tasks
    - Mapping repartitions into byte code
    - Reducing randomly sorts, compiles data

Image from
https://livebook.manning.com/book/concurrency-in-dot-net/chapter-7/10

Big Data Solution: Storing and Accessing

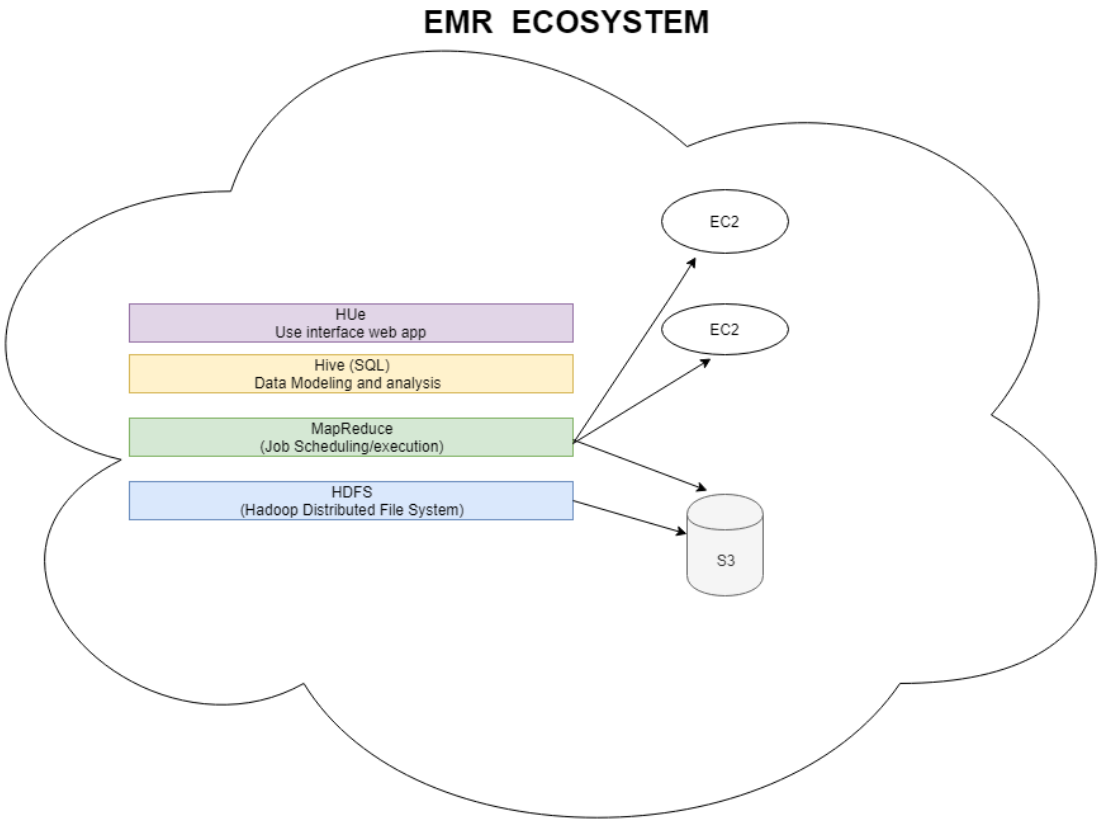
- Apache Hadoop Ecosystem
  - Large data storage
  - Open-source tools
- Apache Hive Data Warehouse
  - NoSQL data warehouse
- Cloudera Hue
  - Graphical interface for data lake, data warehouse

Big Data Solution: Hadoop Ecosystem

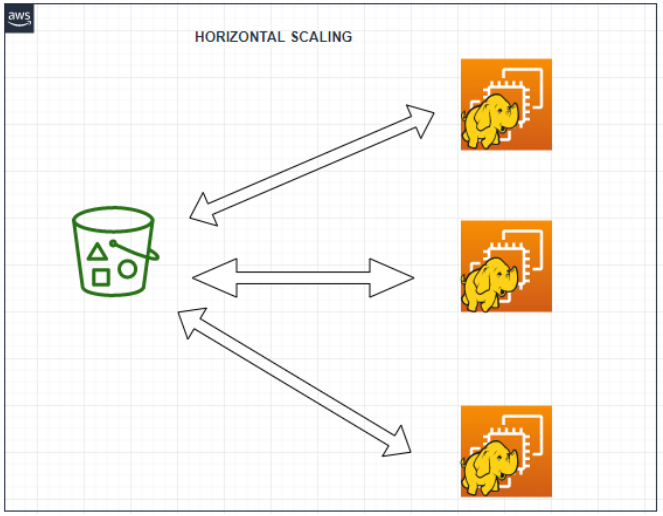


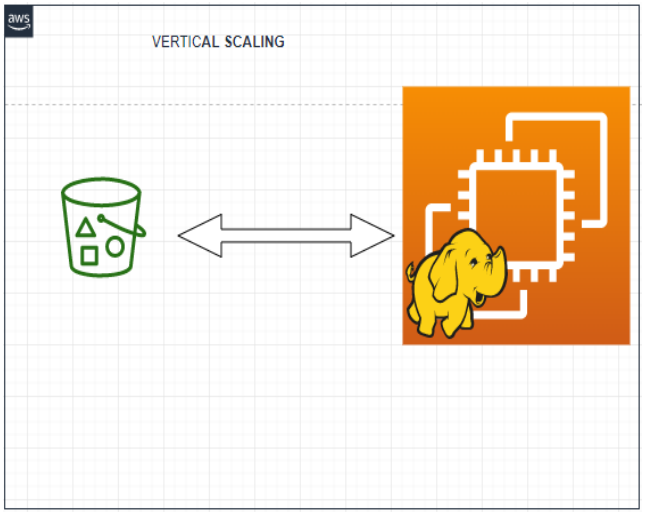
https://data-flair.training/blogs/hadoop-ecosystem-components/

Big Data Tools and Technique:



Scalability: Vertical vs Horizontal

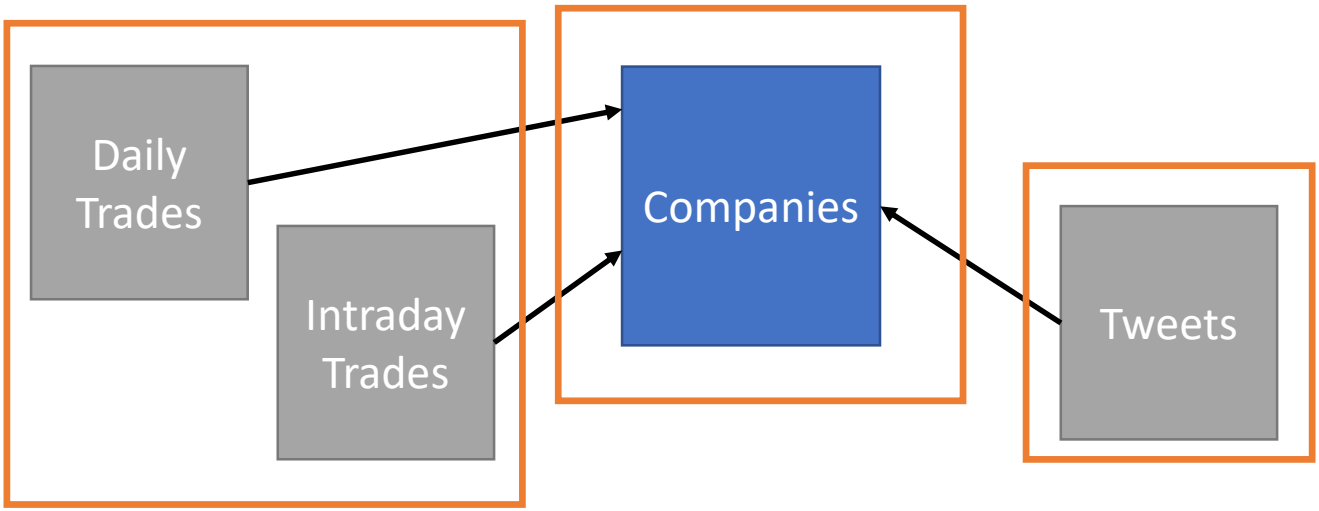




	Horizontal	Vertical
Pros	• Fault tolerance • Low Cost/Better performance	• Low Power consumption • Easy to manage
Cons	• High power consumption • Data inconsistency	• Single point of failure • Hardware limit

Schema Design

- Schemas often designed as **stars**
  - Central table: Fact table
  - Adjacent tables: Dimension tables
- Normalized star schema is a **snowflake**



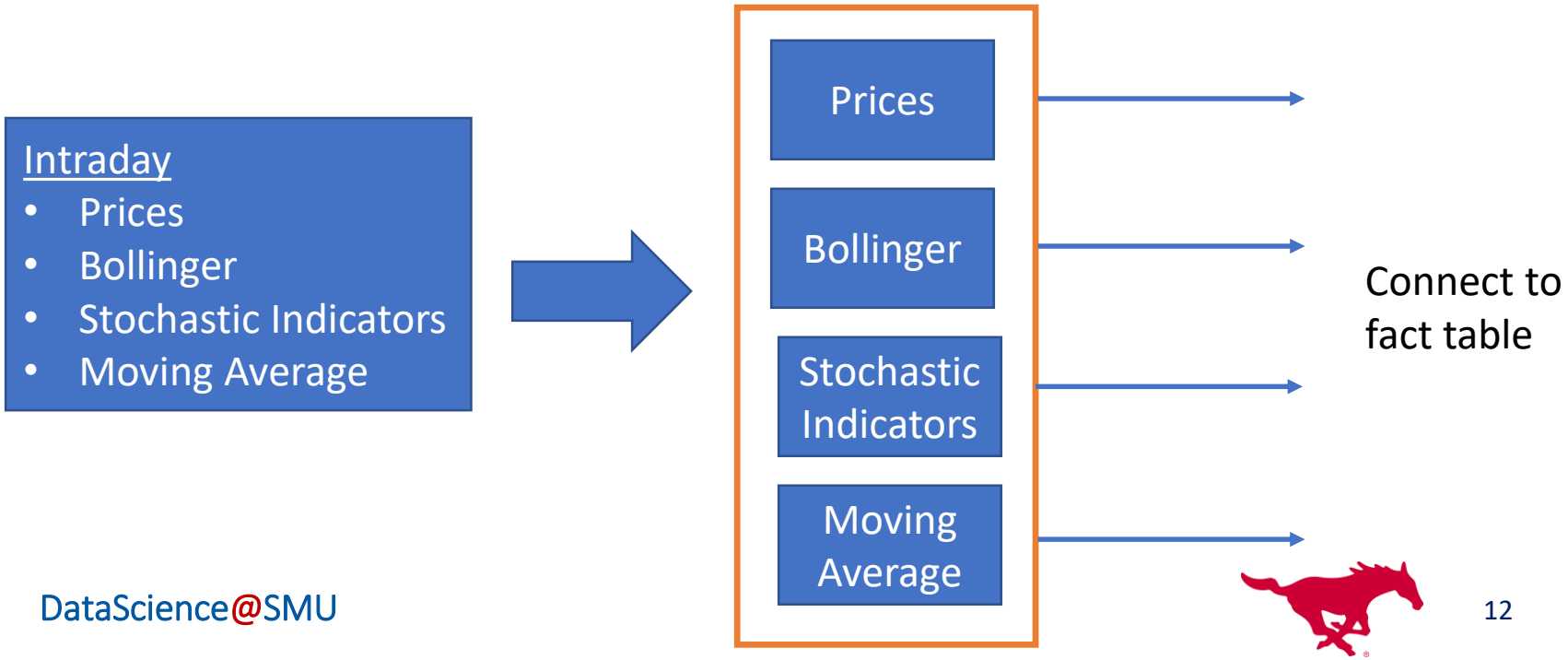
Schema Overview

- Primary key
  - Time + date + symbol
- All dimensions join by time, date, and symbol

- Two designs
  - Normalized (3NF)
  - Denormalized (1NF)

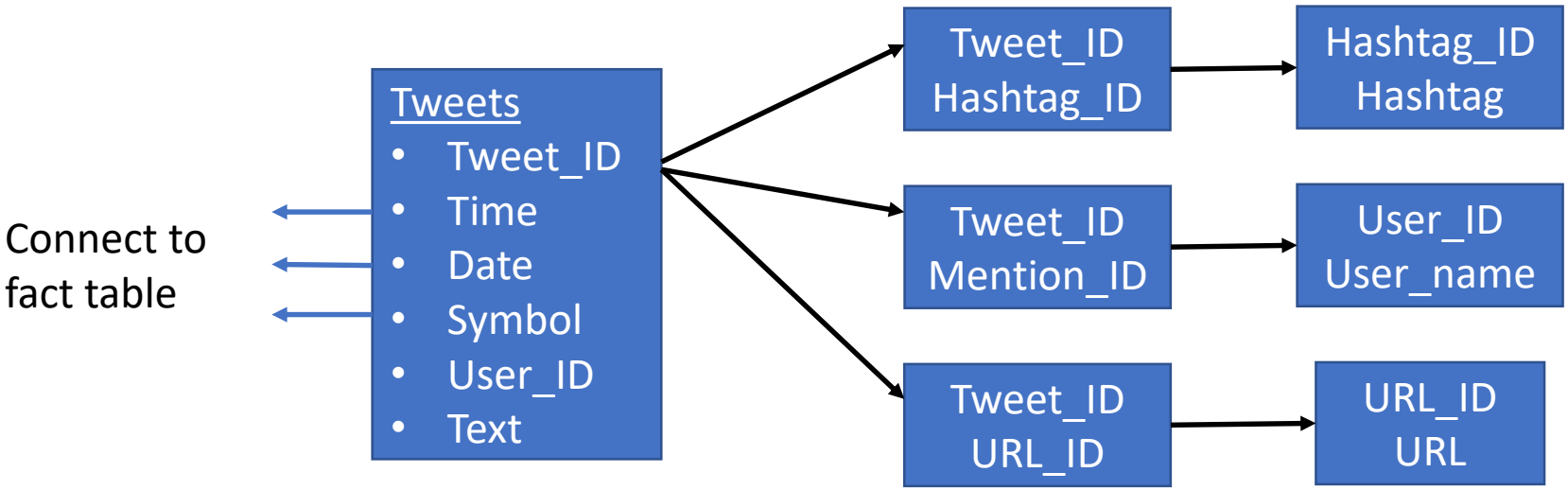
Stock Dimension

- Data is naturally in normal form
- Intraday data was split into tables categorically



Tweet Dimension

Central table + feature tables



Snowflake Schema

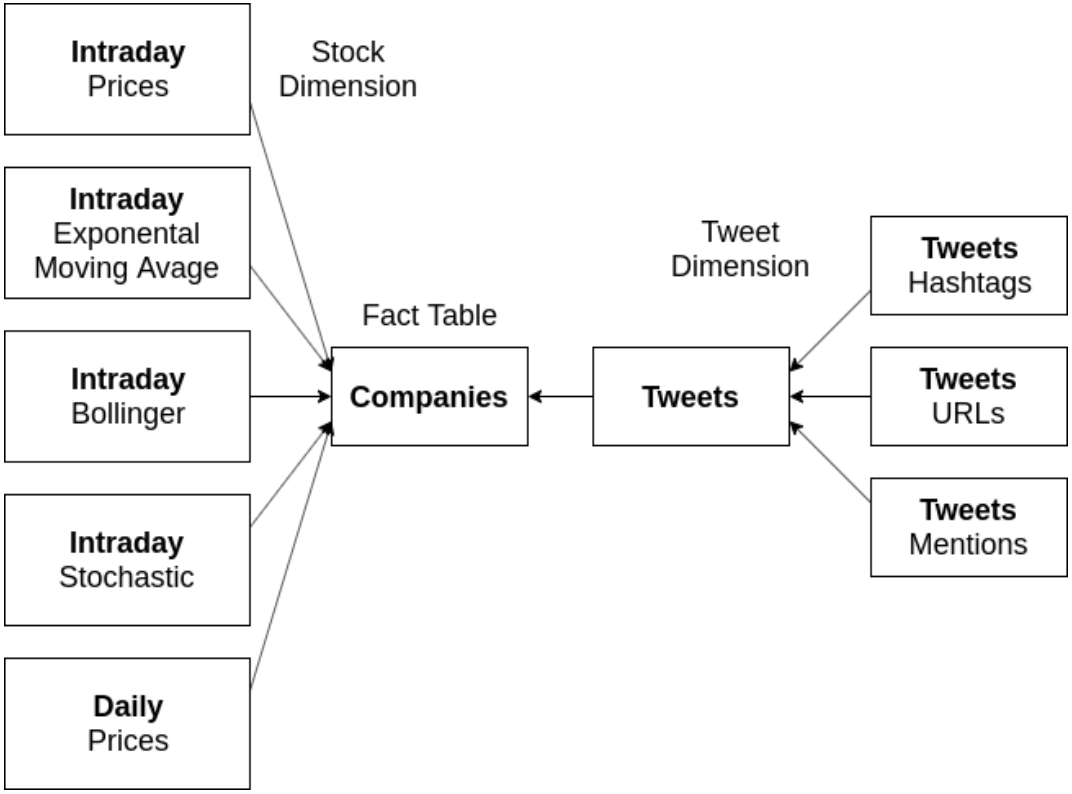


Image source:
https://github.com/sjmiller8182/DBMS_Proj/blob/master/reports/support/images/SnowFlake_Schema_Simple.png

Denormalized Schema

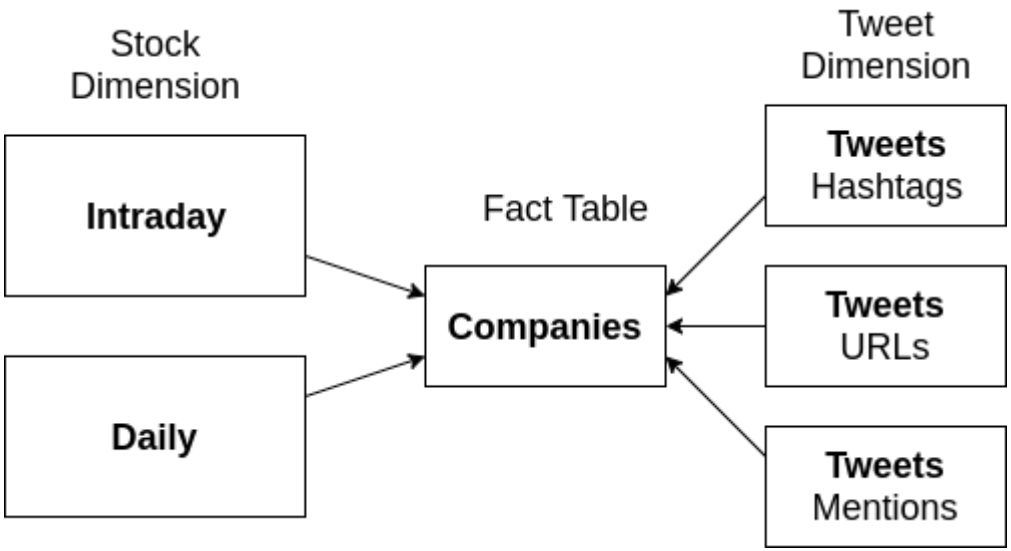
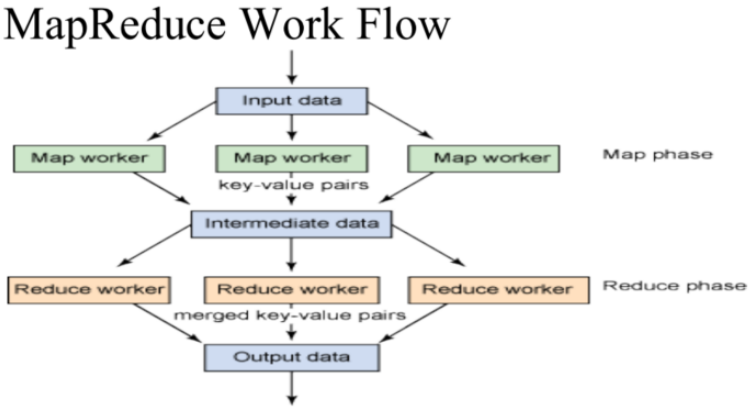
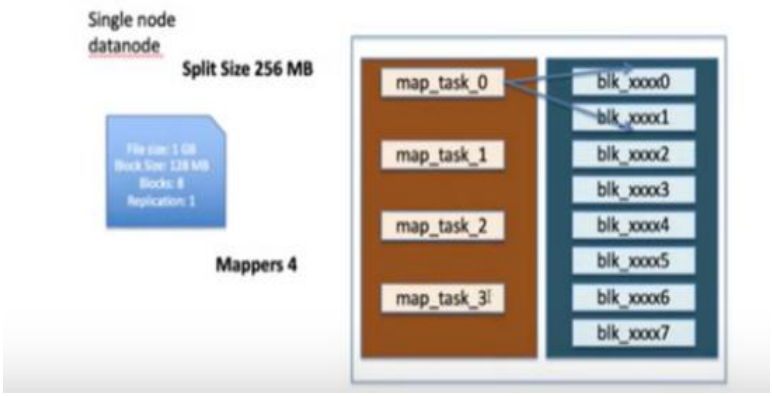


Image source:
https://github.com/sjmiller8182/DBMS_Proj/blob/master/reports/support/images/Star_Schema_Simple.png

Performance Analysis

The goal is to use Hadoop MapReduce configuration to optimize query time





File size = Block Size * number of mapper

Performance Analysis

Schema case 1: Normalized

- Map Reduce block size 64 MB
- Map Reduce block size 128MB
- Map Reduce block size 256MB

Schema case 2: Denormalized

- Map Reduce block size 64 MB
- Map Reduce block size 128MB
- Map Reduce block size 256MB

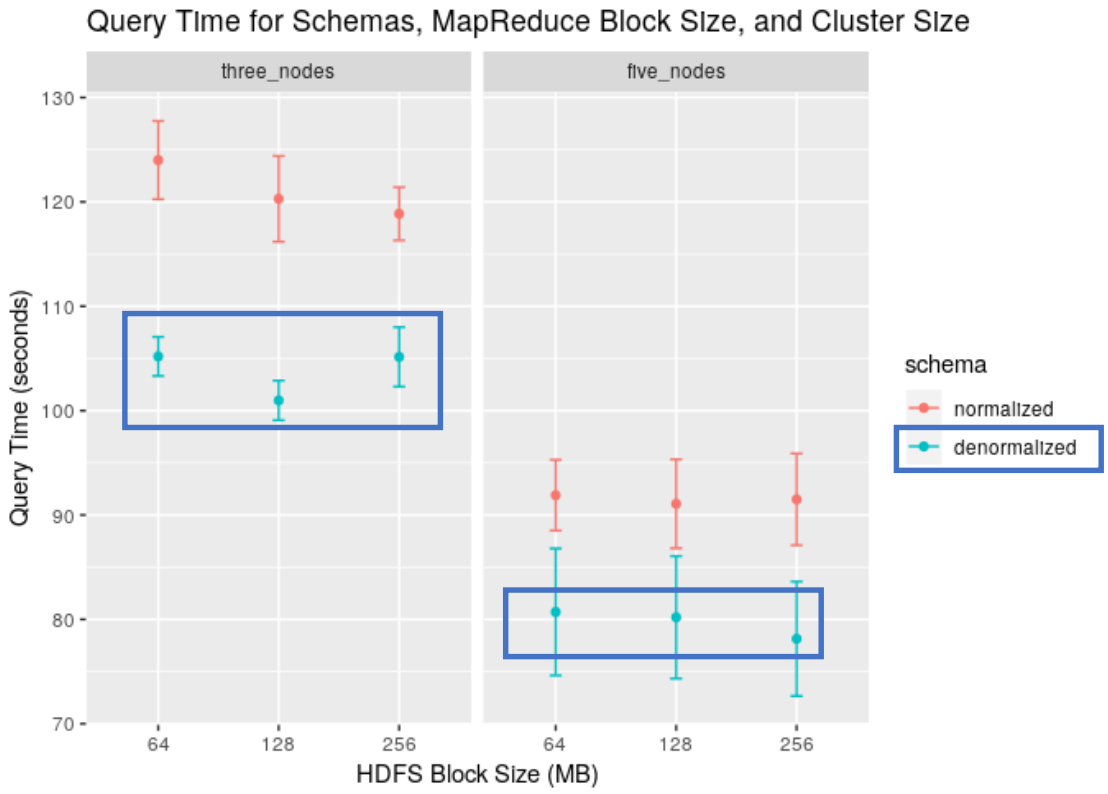
schema	block_size	time
1	64	127.783
1	64	131.555
1	64	121.676
1	64	128.487
1	64	126.547
1	64	125.911
1	64	125.17
1	64	126.655
1	64	124.491
1	64	122.644
1	64	124.284
1	64	117.619
1	64	130.326
1	64	119.305
1	64	124.632
1	64	124.857

Performance Analysis

- Design
  - Three-Way ANOVA
  - Repeated Measures
- Factors
  - Schema
  - HDFS block size
  - EMR cluster size

Findings

Query time is lower for the normalized schema



Conclusion

- Denormalized faster than Normalized
  - Controlled for:
    - MapReduce Block Size
    - Cluster Size
- Increase Cluster Size to Decrease Query Time
  - Hadoop supports horizontal scalability

Questions?

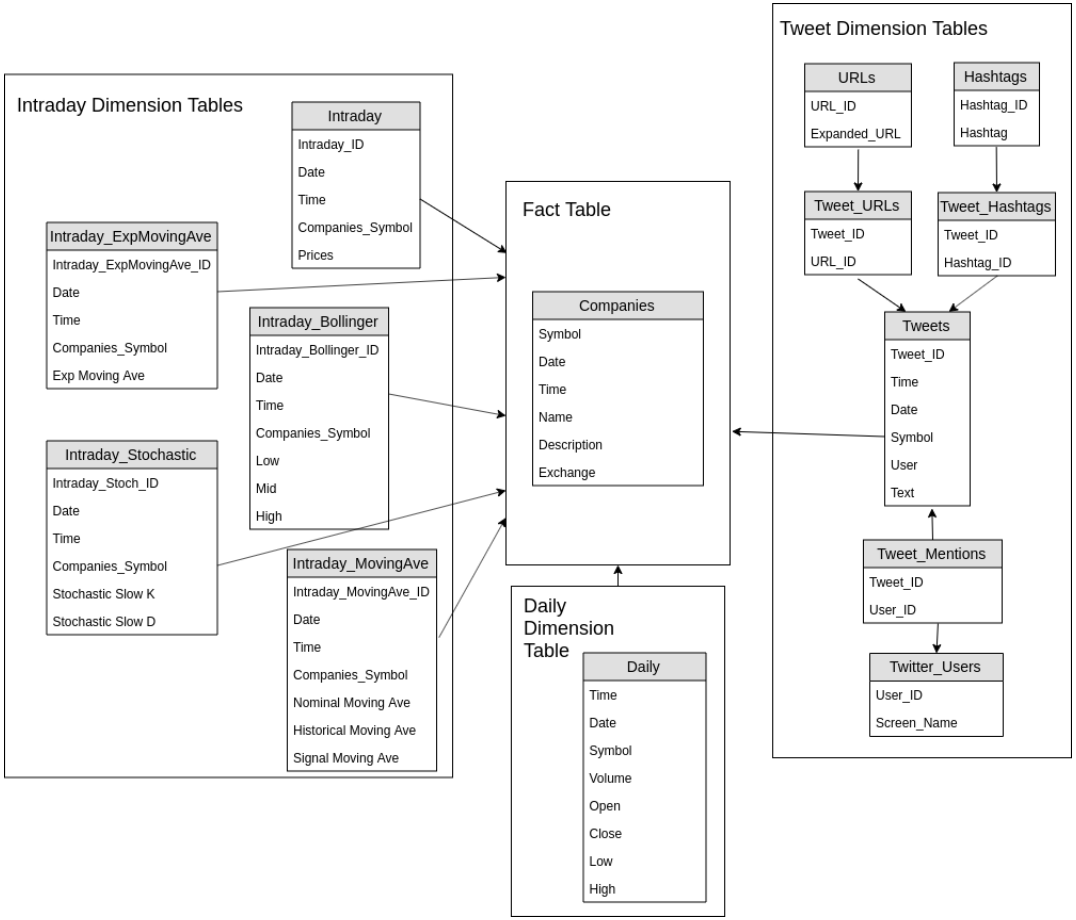
For more information

- Visit the project repo at

https://github.com/sjmiller8182/DBMS_Proj

- View the paper

Backup Schema 3NF



Backup Schema 1NF