Getting Started with HBase: The Hadoop Database

INTRODUCING HBASE

Overview

Understand the need for HBase in a distributed environment

Understand the differences between HBase and an RDBMS

Install and set up HBase

Software for Distributed Computing

How Much Data Do Organizations Deal With?

Google

Google

Current storage = 15 exabytes

Processed per day = 100 petabytes

Number of pages indexed = 60 trillion

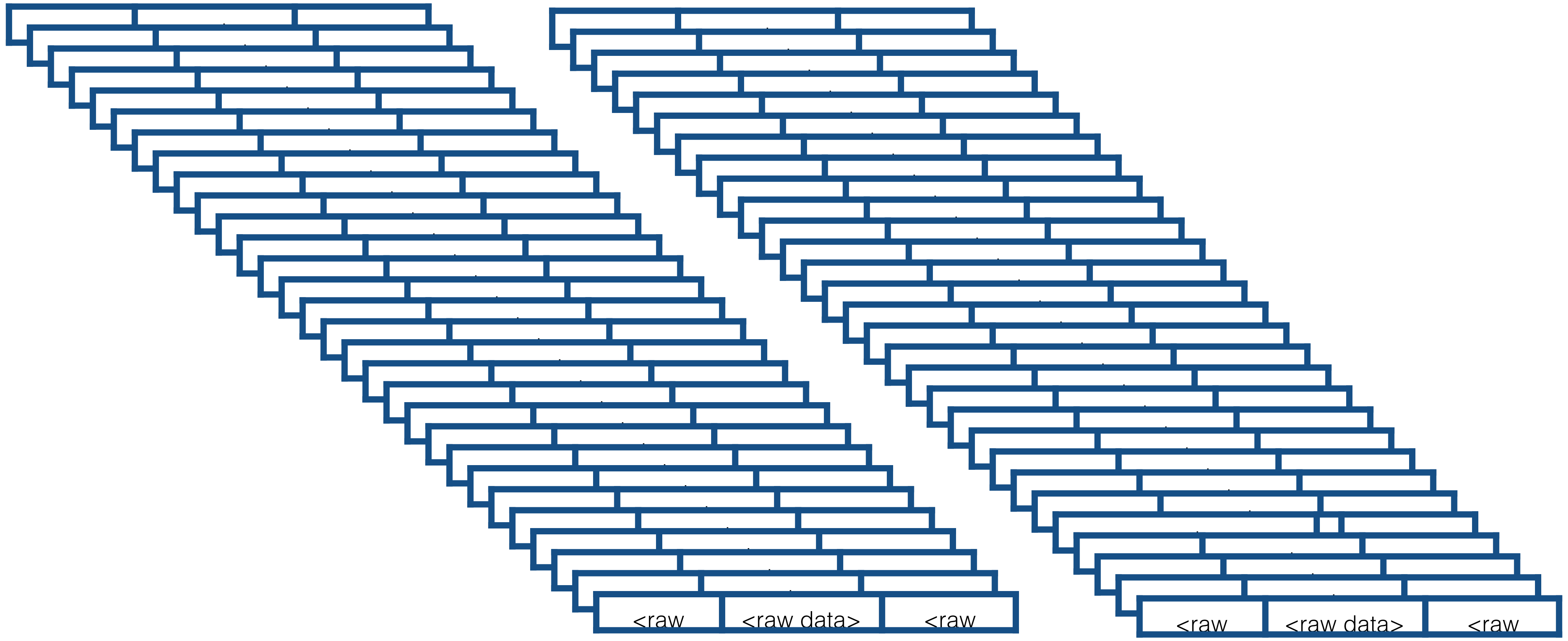
Unique search users per month > 1 billion

Searches per second = 2.3 million

Single Coordinating Software

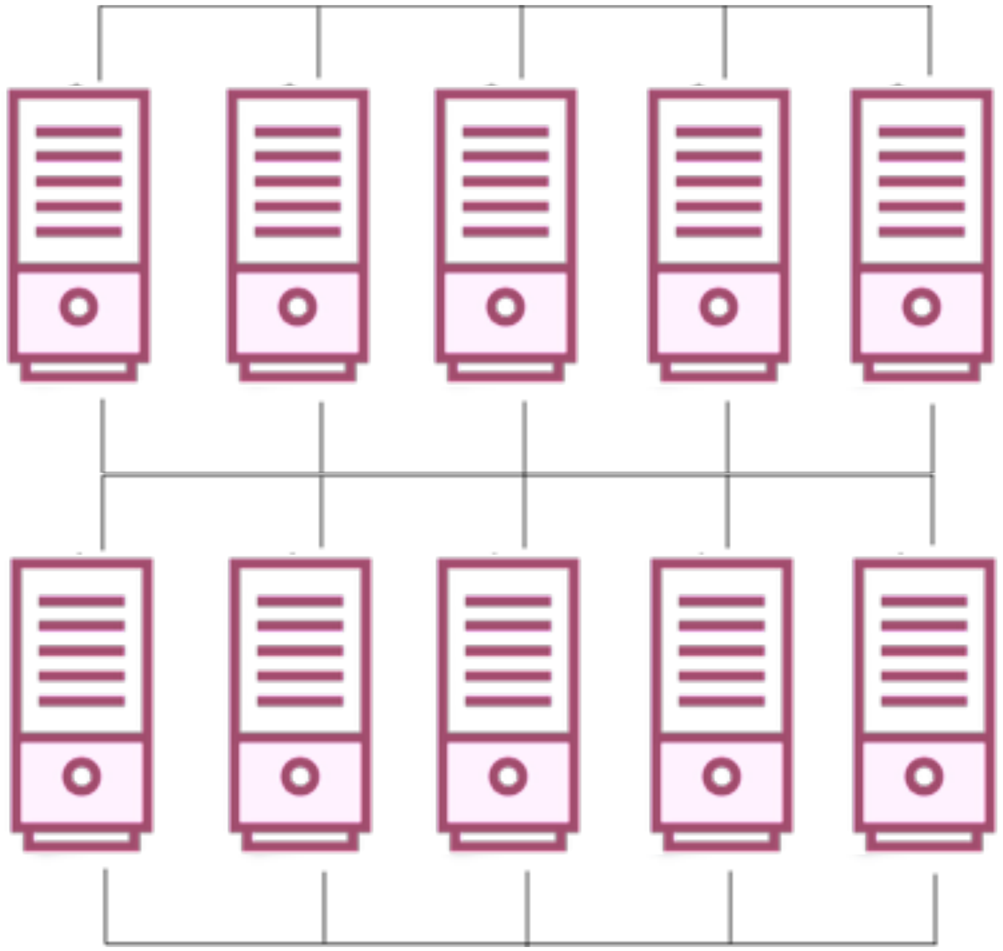
Google realized that running web search algorithms on distributed systems required special software

Single Coordinating Software



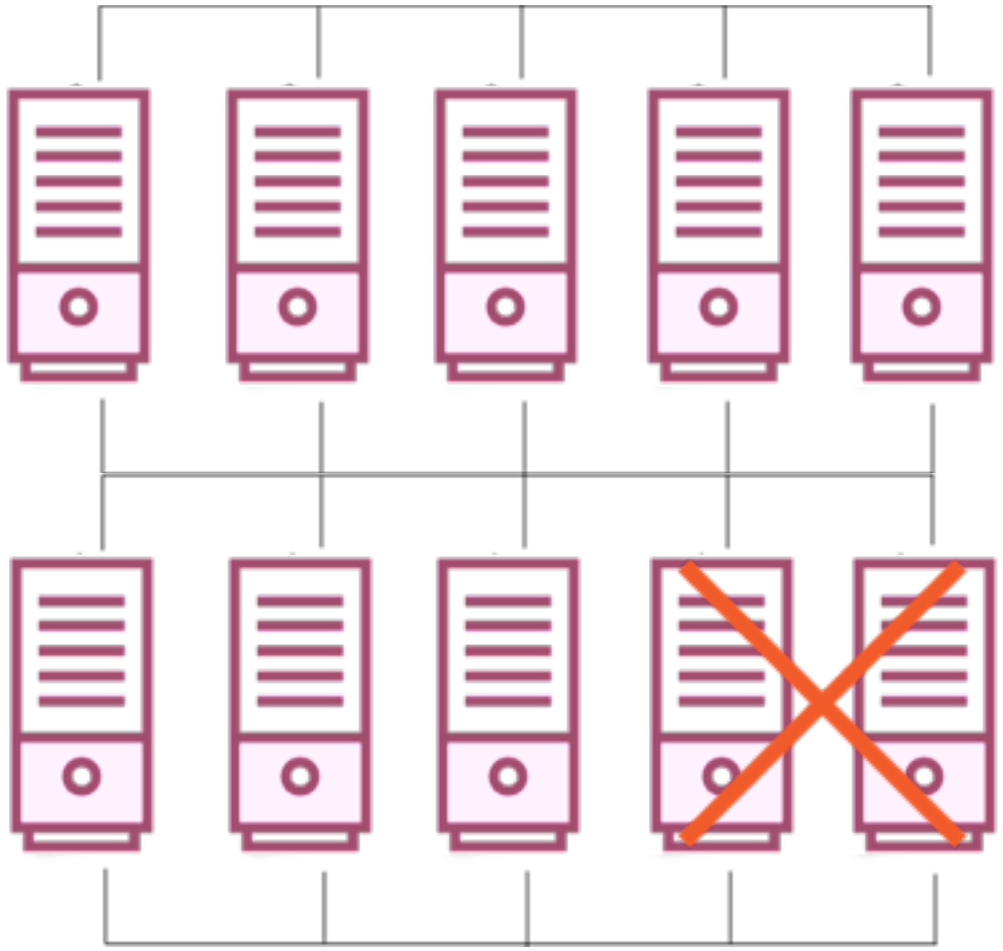
1: Store millions of records on multiple machines

Single Coordinating Software



2: Run processes on all these machines to crunch data

Single Coordinating Software



3: Handle fault tolerance and recovery when nodes crash

Single Coordinating Software

Google File System

To solve distributed storage

MapReduce

To solve distributed computing

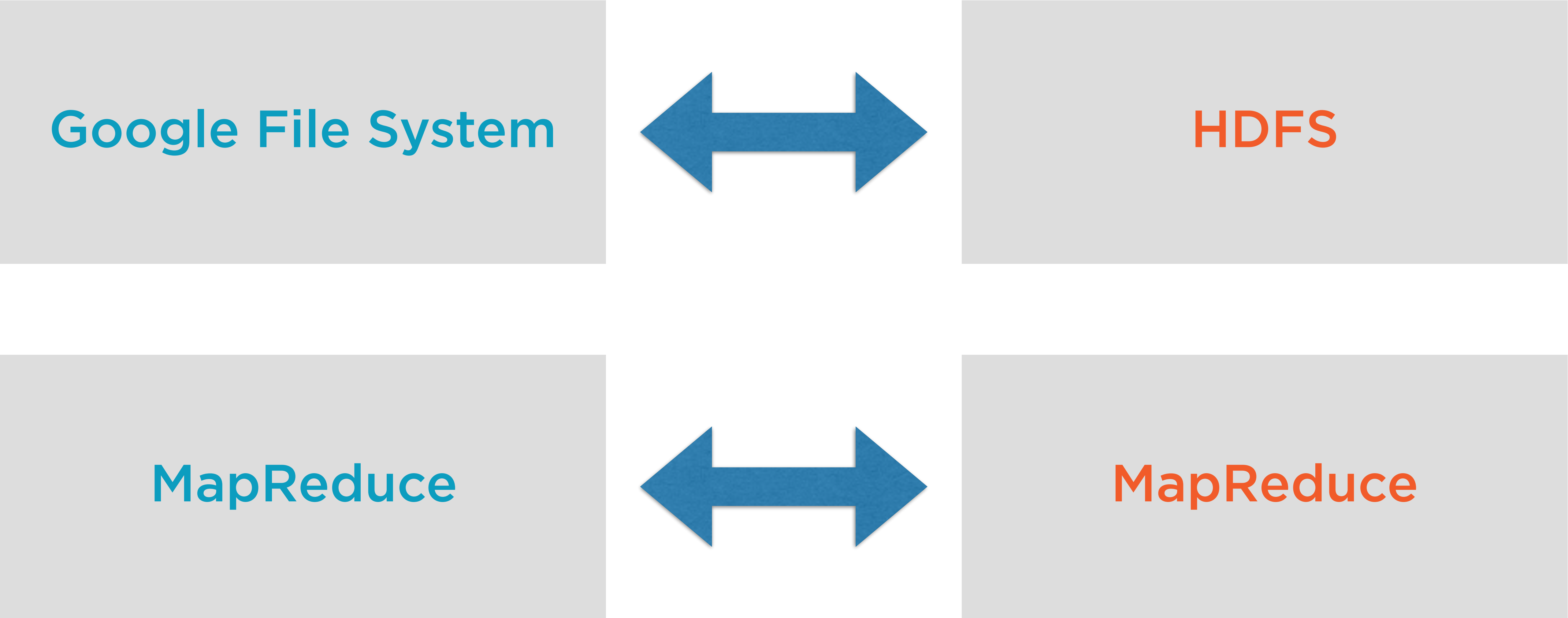
Single Coordinating Software

Google File System

MapReduce

Apache developed open source versions of these technologies

Single Coordinating Software



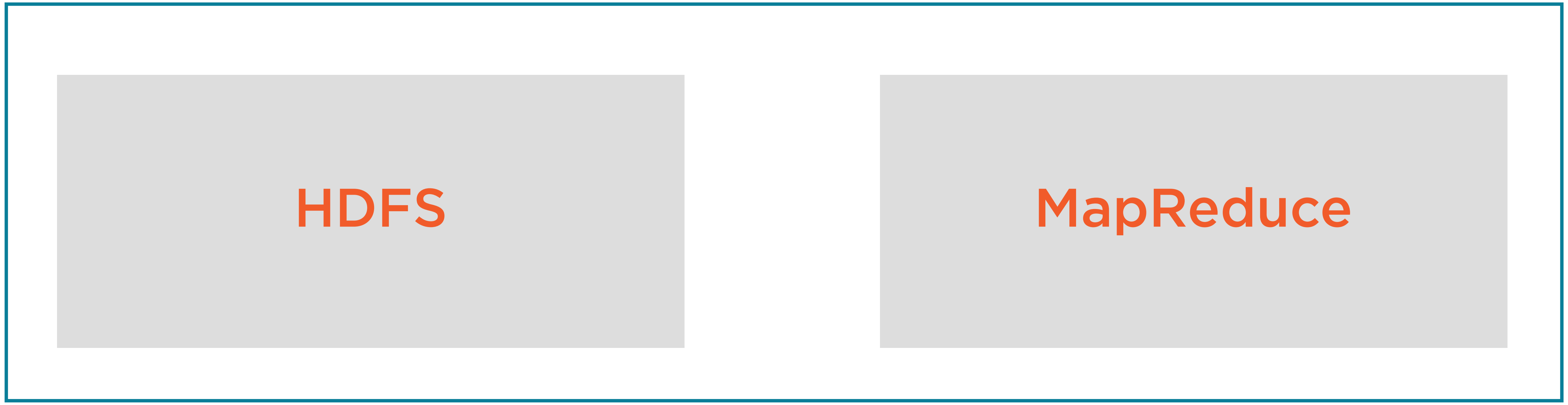
Google File System

HDFS

MapReduce

MapReduce

Hadoop



HDFS

MapReduce

A file system to manage
the storage of data

A framework to process data
across multiple servers

Hadoop is a big data processing framework

Hadoop is **not** a database!

The Importance of Databases

What Kind of Data Do Organizations Store?

Order Management

An e-commerce site stores order information

Payroll

A company stores employee payroll details

Accounts

A bank stores account and transaction information

Requirements of a Database

Structured: Rows and columns

Random access: Update one row at a time

Low latency: Very fast read/write/ update operations

ACID compliant: Ensure data integrity

What Are ACID Properties?

Atomicity

Consistency

Isolation

Durability

Atomicity

Transactions on a database should be all-or-nothing

Atomicity

Transferring Money

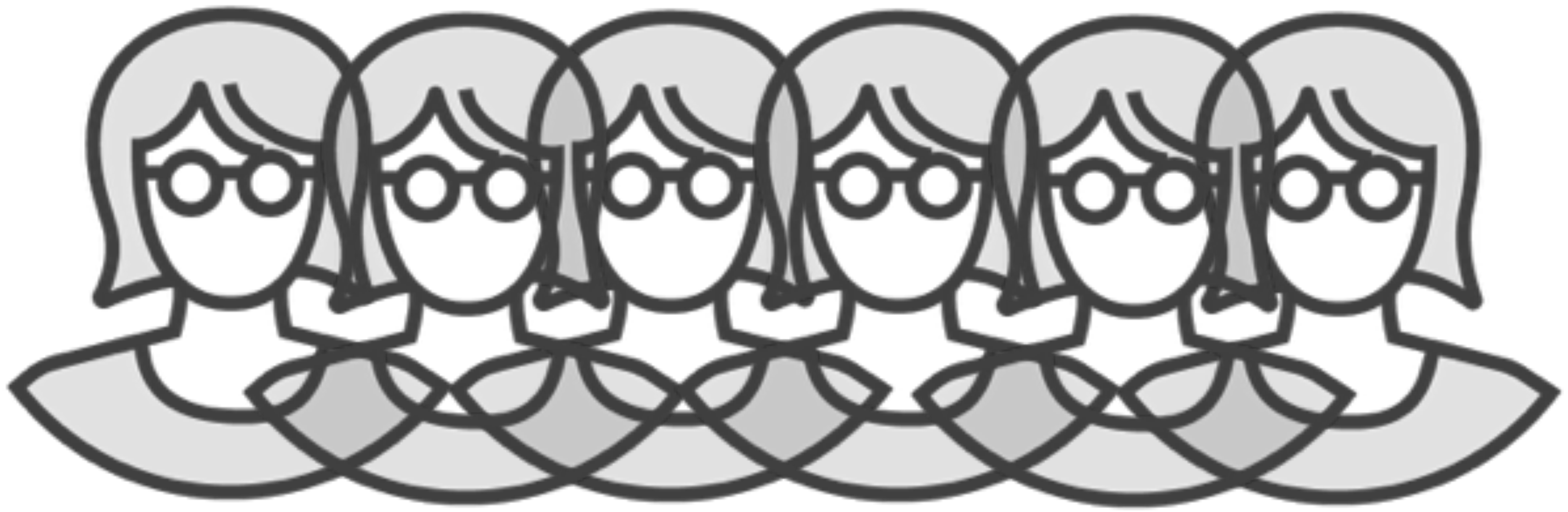
Both withdrawal and deposit should occur or none at all!

Consistency

Database updates should not violate any constraints

Consistency

Enrolling Students



Every student should have a unique student id

Isolation

Concurrent operations on the database should appear as though they were applied in some sequence

Isolation

Granularity of Updates

Can employee address be updated at the same time as employee salary?

Durability

Once changes have been made to the data they are permanent

Durability

Safety of Data

In case of power loss, crashes, errors

The Limitations of Hadoop

Unfortunately, Hadoop makes a very poor database

Limitations of Hadoop

Unstructured data

No random access

High latency

Not ACID compliant

Limitations of Hadoop

Data in HDFS has no schema, no rows and columns, no tables

Text files

Log files

Audio files

Video files

Unstructured data

Limitations of Hadoop

Basic structure exists for some file types

CSV files

XML files

JSON files

Hadoop enforces no constraints on these

Unstructured data

Limitations of Hadoop

Cannot create, access and modify individual records in a file

MapReduce parses entire files to extract information

No random access

Limitations of Hadoop

Not suited for real-time processing where a user waits for data to be retrieved

Batch processing with long running jobs

High latency

Limitations of Hadoop

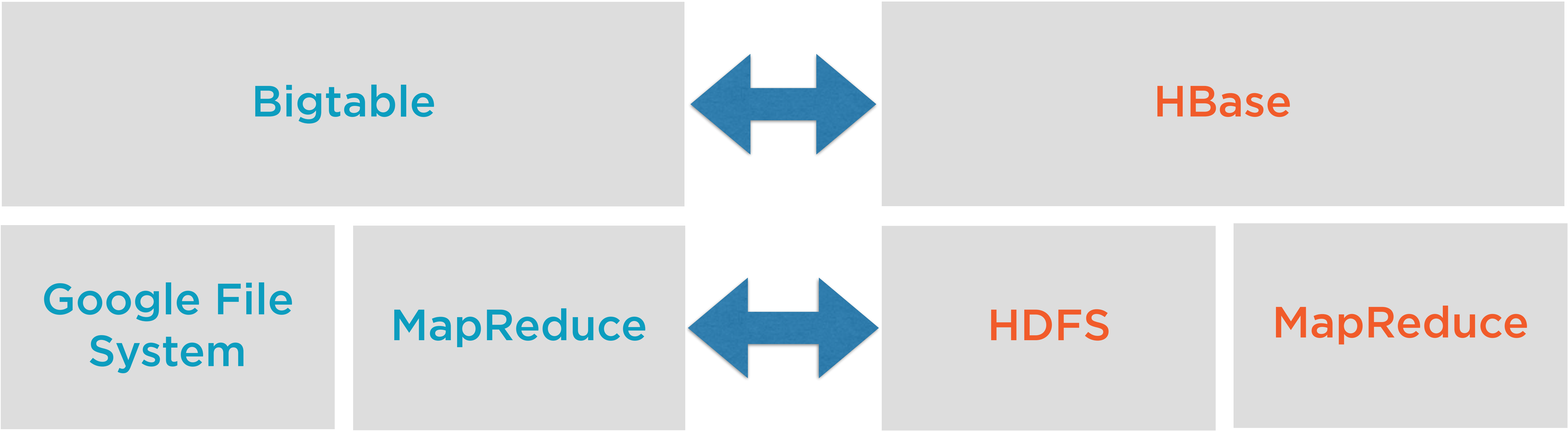
HDFS is a file storage system and provides no guarantees for data integrity

Not ACID compliant

How Did Google Solve This for Search?

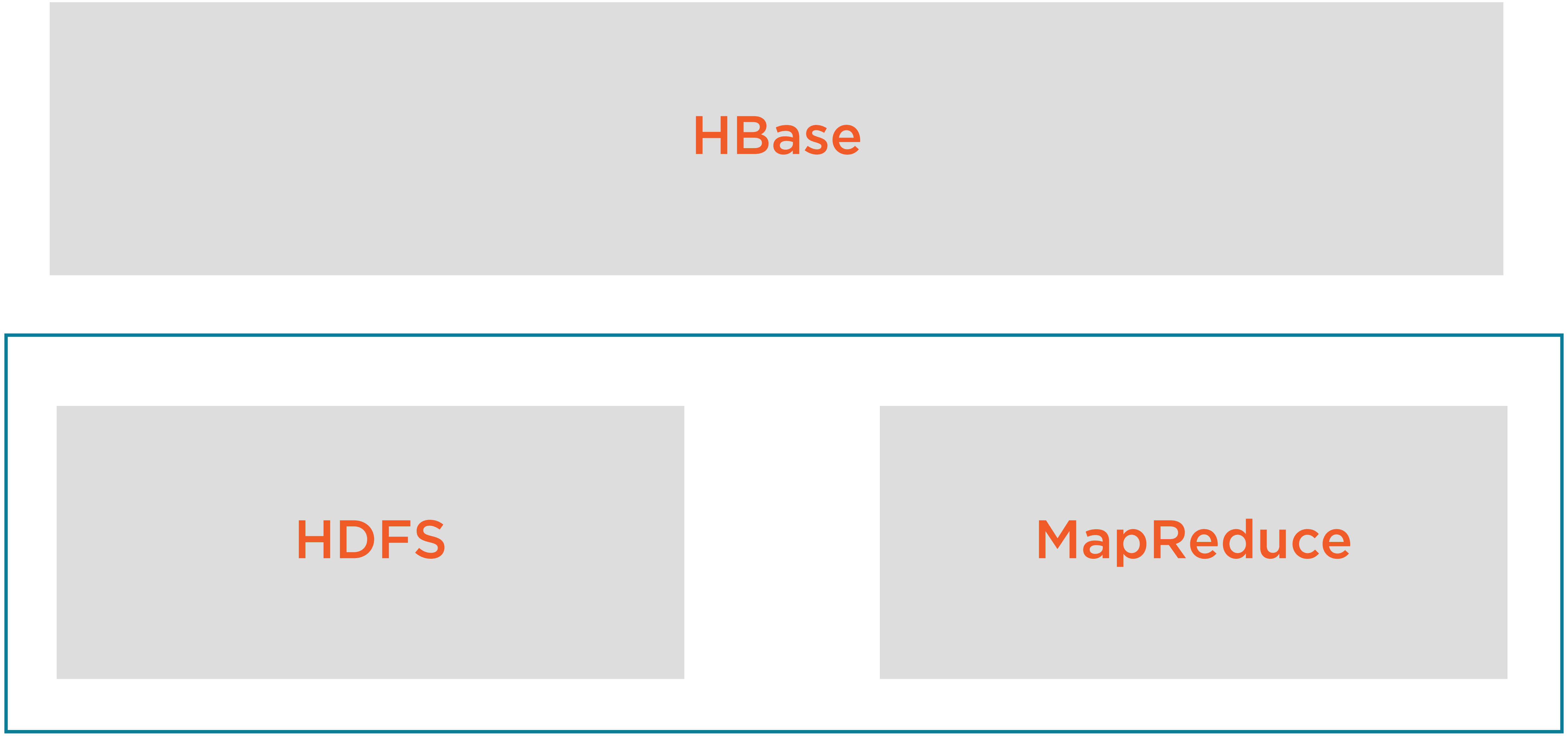
Google published a paper on Bigtable a distributed storage system for structured data

How Did Google Solve This for Search?



HBase is a distributed database management system which runs on top of Hadoop

HBase



HBase

Distributed: Stores data in HDFS

Scalable: Capacity directly proportional to number of nodes in the cluster

Fault tolerant: Piggybacks on Hadoop

HBase

Structured: A loose data structure

Low latency: Real-time access using row based indices called row keys

Random access: Row keys allow access updates to one record

Somewhat ACID compliant: Some transactions will have ACID properties

HBase

Batch processing using MapReduce

Real-time processing using row keys

HBase vs. Relational Databases

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

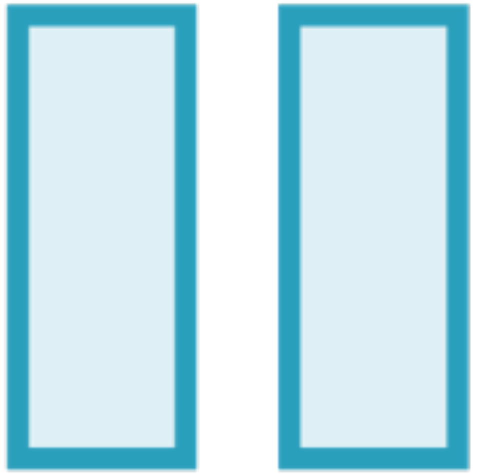
Columnar store

A Notification Service

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Layout of a traditional relational database

A Notification Service



Columnar store

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Layout of a traditional relational database

Columnar store

A Notification Service

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Row = 3
Column = To

Columnar Store

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Columnar Store

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Columnar Store

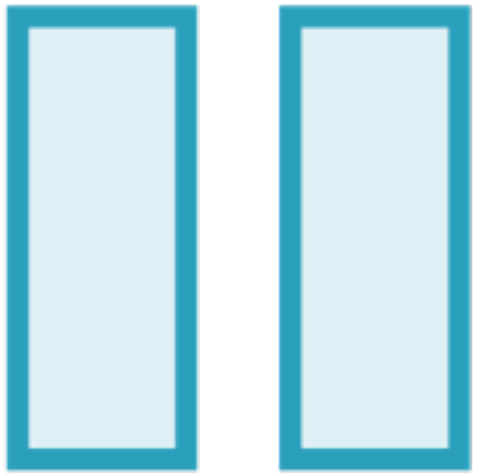
Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Columnar Store

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale



Columnar store

Advantages of a Columnar Store

Sparse tables: No wastage of space when storing sparse data

Dynamic attributes: Update attributes dynamically without changing storage structure

Sparse Tables

Id	To	Type	Content	Expiry
1	mike	offer	Offer on mobiles	2345689070
2	john	sale	Redmi sale	
3	jill	order	Order delivered	
4	megan	sale	Clothes sale	2456123989

Sale and offer notifications may have an expiry time

Sparse Tables

Id	To	Type	Content	Expiry	Order Status
1	mike	offer	Offer on mobiles	2345689070	
2	john	sale	Redmi sale		
3	jill	order	Order delivered		Delivered
4	megan	sale	Clothes sale	2456123989	

Order related notifications may have an order status

Sparse Tables

Id	To	Type	Content	Expiry	Order Status
1	mike	offer	Offer on mobiles	2345689070	
2	john	sale	Redmi sale		
3	jill	order	Order delivered		Delivered
4	megan	sale	Clothes sale	2456123989	

In a traditional database this results in a change in database structure

Sparse Tables

Id	To	Type	Content	Expiry	Order Status
1	mike	offer	Offer on mobiles	2345689070	
2	john	sale	Redmi sale		
3	jill	order	Order delivered		Delivered
4	megan	sale	Clothes sale	2456123989	

And empty cells when data is not applicable to certain rows

Sparse Tables

Id	To	Type	Content	Expiry	Order Status
1	mike	offer	Offer on mobiles	2345689070	
2	john	sale	Redmi sale		
3	jill	order	Order delivered		Delivered
4	megan	sale	Clothes sale	2456123989	

These cells still occupy space!

Columnar store

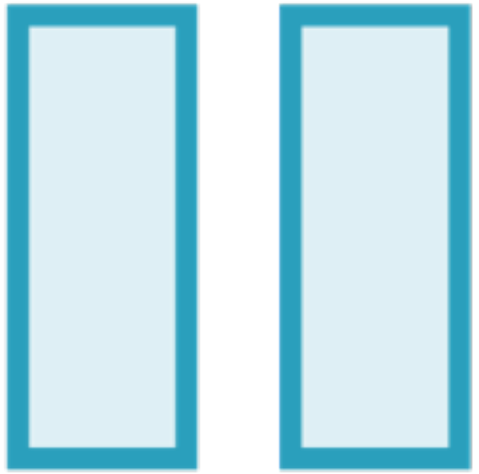
Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on
1	Expiry	2345689070
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale
4	Expiry	2456123989

Dynamically add new attributes as rows in this table

Columnar store

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on
1	Expiry	2345689070
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale
4	Expiry	2456123989

No wastage of space with empty cells!



Columnar store

Note that this is not the exact layout of how data is stored in HBase

It is a general structure of how columnar stores are constructed

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

Denormalized storage

Traditional databases use normalized forms of database design to **minimize redundancy**

Minimize Redundancy

Denormalized storage

Employee Details

Employee Subordinates

Employee Address

Denormalized storage

Employee Details

Id	Name	Function	Grade
1	Emily	Finance	6

Employee Subordinates

Id	Subordinate Id
1	2
1	3

Employee Address

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Denormalized storage

Employee Details

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

All employee details in one table

Denormalized storage

Employee Subordinates

id	Subordinate id
1	2
1	3

Employees referenced only by ids everywhere else

Denormalized storage

Employee Address

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Data is made more granular by splitting it across multiple tables

Denormalized storage

Id	Name	Function	Grade
1	Emily	Finance	6

Id	Subordinate Id
1	2
1	3

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Normalization

Normalization

Denormalized storage

Optimizes storage

But storage is cheap in a distributed system!

Denormalized storage

But storage is cheap in a distributed system!

Optimize number of disk seeks

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	Subordinate Id
1	2
1	3

Id	Name	Function	Grade	Subordinates
1	Emily	Finance	6	**<ARRAY>**
2	John	Finance	3	
3	Ben	Finance	4	

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id		City	Zip Code
1		Palo Alto	94305
2		Seattle	98101

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**<ARRAY>**	**<STRUCT>**
2	John	Finance	3		
3	Ben	Finance	4		

Denormalized Storage

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**<ARRAY>**	**<STRUCT>**
2	John	Finance	3		
3	Ben	Finance	4		

Store everything related to an employee in the same table

Denormalized Storage

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**<ARRAY>**	**<STRUCT>**
2	John	Finance	3		
3	Ben	Finance	4		

Read a single record to get all details about an employee in one read operation

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

Only CRUD operations

Traditional Databases and SQL

Joins: Combining information across tables using keys

Group By: Grouping and aggregating data for the groups

Order By: Sorting rows by a certain column

Only CRUD operations

HBase does not support SQL

NoSQL

Only CRUD operations

Only a limited set of operations are allowed in HBase

Create

Read

Update

Delete

CRUD

Only CRUD operations

No operations involving multiple tables

No indexes on tables

No constraints

Only CRUD operations

Id	Name	Function	Grade	Subordinates	Address

This is why all details need to be self contained in one row

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

ACID at the row level

Updates to a single row are atomic

All columns in a row are updated or none are

ACID at the row level

Updates to multiple rows are **not** atomic

Even if the update is on the same column in multiple rows

Traditional RDBMS vs. HBase

Traditional RDBMS	HBase
Data arranged in rows and columns	Data arranged in a column-wise manner
Supports SQL	NoSQL database
Complex queries such as grouping, aggregates, joins etc	Only basic operations such as create, read, update and delete
Normalized storage to minimize redundancy and optimize space	Denormalized storage to minimize disk seeks
ACID compliant	ACID compliant at the row level

Demo

Install and set up HBase in pseudo-distributed mode

Summary

Understood the need for a distributed database system like HBase

Know how HBase differs from a traditional RDBMS

Installed and set up HBase on your local machine