

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Module 4

4.1 Hive

Introduction

Apache Hive is an **open-source data warehouse software** built on top of the Hadoop ecosystem. It provides an SQL-like language called **Hive Query Language (HQL)** that enables users to query, summarize, and analyze large datasets stored in Hadoop Distributed File System (HDFS).

Hive – Suitable For		
Data warehousing applications	Processes batch jobs on huge data that is immutable (data whose structure cannot be changed after it is created is called immutable data)	Examples: Web Logs, Application Logs

Figure 9.1 Hive – a data warehousing tool.

Hive was originally developed by **Facebook** to process enormous volumes of structured data and is now maintained by the Apache Software Foundation.

Unlike traditional relational databases, Hive is designed for **batch processing** rather than real-time transactions.

Definition

Apache Hive is a data warehouse infrastructure built on Hadoop that facilitates querying, managing, and analyzing large datasets using SQL-like commands.

Features of Hive

- SQL-like Query Language (HQL)
- Supports structured and semi-structured data
- Scalable for petabytes of data
- Fault tolerant

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Integrates with Hadoop ecosystem
- Supports User Defined Functions (UDF)
- Partitioning and Bucketing
- Metadata management using Metastore

Advantages

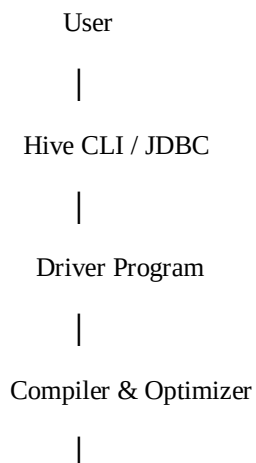
- Easy for SQL users
- Handles huge volumes of data
- Highly scalable
- Cost-effective
- Supports ETL operations
- Provides data summarization

Limitations

- Not suitable for OLTP
- High query latency
- No row-level updates
- Batch-oriented processing

Hive Architecture

Components



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Execution Engine

|

MapReduce / Tez / Spark

|

Hadoop HDFS

|

Data Storage

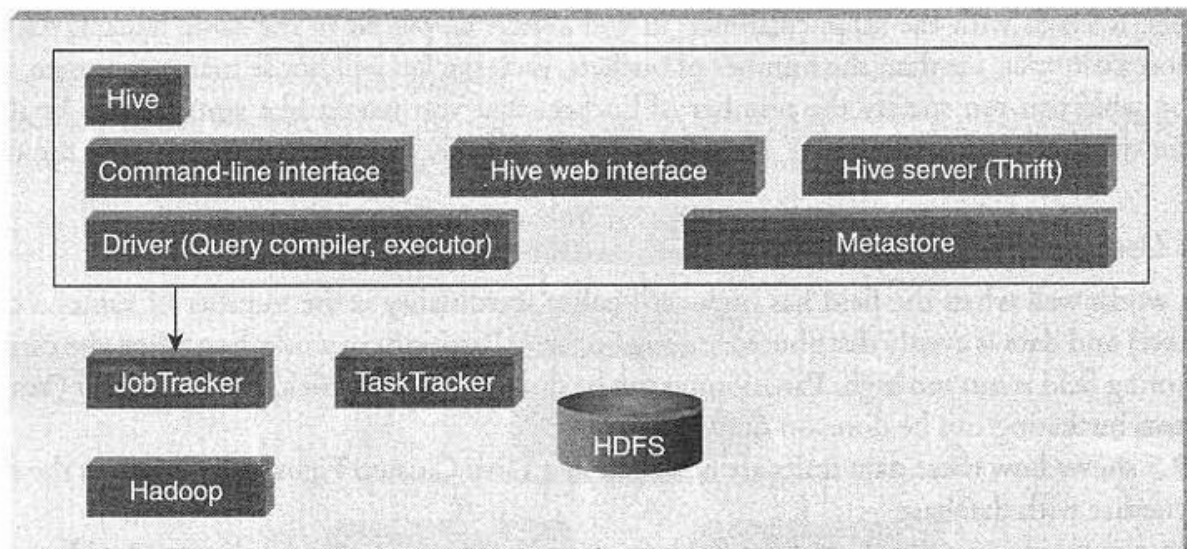


Figure 9.7 Hive architecture.

Components Explanation

1. User Interface

Allows users to interact using

- Hive CLI
- Beeline
- JDBC
- ODBC
- Web Interface

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

2. Driver

Responsible for

- Creating sessions
- Executing queries
- Managing lifecycle

3. Compiler

Converts HQL into execution plans.

Functions:

- Syntax checking
- Semantic analysis
- Optimization

4. Metastore

Stores metadata such as

- Table names
- Database names
- Column information
- Partitions
- Data location
- File formats

Usually stored in MySQL or Derby database.

5. Execution Engine

Executes optimized query plans.

Communicates with

- MapReduce
- Apache Tez

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Apache Spark

6. Hadoop

Stores actual data in

- HDFS

Processes data using

- MapReduce
- YARN

Hive Workflow

User

|

Writes HQL Query

|

Hive Driver

|

Compiler

|

Optimizer

|

Execution Engine

|

MapReduce / Tez

|

HDFS

|

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Results Returned

Working of Hive

Step 1

User writes an HQL query.



Step 2

Driver receives the query.



Step 3

Compiler checks syntax.



Step 4

Execution plan is created.



Step 5

Optimizer improves the execution plan.



Step 6

MapReduce/Tez jobs are executed.



Step 7

Results are returned.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Hive Data Model

Hive organizes data into several logical units.

Hierarchy:

Database



Tables



Partitions



Buckets



Rows



Columns

Database

A collection of related tables.

Example

```
CREATE DATABASE Company;
```

Table

A collection of rows and columns.

Example

Employee Table

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

EmpID Name Salary

101	Raj	50000
102	Anita	65000

Types of Tables

Managed Table

Hive controls both

- Data
- Metadata

Deleting table removes both.

External Table

Hive controls only metadata.

Actual data remains in HDFS.

Useful when multiple applications share the same data.

Hive Data Types

Primitive Data Types

- INT
- BIGINT
- FLOAT
- DOUBLE
- BOOLEAN
- STRING
- CHAR
- VARCHAR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- DATE
- TIMESTAMP

Primitive Data Types

Numeric Data Type	
TINYINT	1-byte signed integer
SMALLINT	2-byte signed integer
INT	4-byte signed integer
BIGINT	8-byte signed integer
FLOAT	4-byte single-precision floating-point
DOUBLE	8-byte double-precision floating-point number
String Types	
STRING	
VARCHAR	Only available starting with Hive 0.12.0
CHAR	Only available starting with Hive 0.13.0
Strings can be expressed in either single quotes (') or double quotes (")	
Miscellaneous Types	
BOOLEAN	
BINARY	Only available starting with Hive

Complex Data Types

- ARRAY
- MAP
- STRUCT
- UNIONTYPE

Example

ARRAY<int>

MAP<string,int>

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

STRUCT<Name:string, Age:int>

Comparison: Hive vs Traditional RDBMS

Hive	RDBMS
Batch processing	Transaction processing
Works on Hadoop	Works on local storage
HQL	SQL
High latency	Low latency
Petabyte scale	Gigabyte/Terabyte scale
No row-level updates	Supports updates

Applications of Hive

- Data Warehousing
- Log Analysis
- Business Intelligence
- Data Mining
- Report Generation
- Customer Analytics
- Financial Analysis
- Retail Analytics

Advantages of Hive in Business Intelligence

- Simplifies big data analysis
- Supports SQL queries
- Integrates with Hadoop
- Low-cost storage
- High scalability
- Easy report generation

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

4.2 Hive Partitioning

Introduction

As the size of data increases, scanning an entire table for every query becomes time-consuming. **Partitioning** divides a table into smaller logical parts based on the values of one or more columns, enabling Hive to read only the relevant partition instead of the entire table.

Definition

Partitioning is the process of dividing a Hive table into multiple directories based on the values of partition columns.

Each partition corresponds to a separate folder in HDFS.

Advantages of Partitioning

- Reduces query execution time.
- Minimizes data scanning.
- Improves query performance.
- Simplifies data management.
- Efficient handling of large datasets.
- Easy deletion and archiving of old data.

Example

Suppose an employee table stores data for multiple departments.

EmpID Name Department Salary

101	Raj	CSE	50000
102	Anita	ISE	60000
103	Rahul	ECE	55000

The table can be partitioned by **Department**.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Employee

└─ Department=CSE

└─ Department=ISE

└─ Department=ECE

When the query is:

```
SELECT * FROM Employee  
WHERE Department='CSE';
```

Hive scans only the **CSE partition**, improving performance.

Types of Partitioning

1. Static Partition

Partition value is specified manually.

Example:

```
INSERT INTO TABLE Employee  
PARTITION(Department='CSE')  
SELECT * FROM TempEmployee;
```

2. Dynamic Partition

Hive automatically creates partitions based on incoming data.

Example:

```
INSERT INTO TABLE Employee  
PARTITION(Department)  
SELECT EmpID,Name,Salary,Department  
FROM TempEmployee;
```

Advantages

- Faster queries
- Less disk I/O
- Better scalability
- Easy maintenance

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Limitations

- Too many partitions reduce efficiency.
- Small files increase overhead.
- Wrong partition key affects performance.

4.3 Hive Bucketing

Introduction

Partitioning divides data into directories, whereas **Bucketing** divides data into a fixed number of files.

Bucketing distributes records using a hash function.

Definition

Bucketing is the process of dividing data into fixed-size buckets based on the hash value of a column.

Why Bucketing?

- Faster joins
- Efficient sampling
- Better query optimization
- Even distribution of records

Example

Suppose 1000 employee records are divided into 5 buckets.

Employee Table

↓

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Hash(EmployeeID)

↓

Bucket1

Bucket2

Bucket3

Bucket4

Bucket5

Bucket Creation

```
CREATE TABLE Employee  
(  
  EmpID INT,  
  Name STRING,  
  Salary FLOAT  
)
```

```
CLUSTERED BY(EmpID)
```

```
INTO 5 BUCKETS;
```

Partitioning vs Bucketing

Partitioning

Based on column value

Creates directories

Dynamic number

Improves filtering

Suitable for huge tables

Bucketing

Based on hash function

Creates files

Fixed number

Improves joins

Suitable for sampling

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

4.4 Hive Query Language (HQL)

Introduction

Hive Query Language (HQL) resembles SQL and is used to create databases, tables, insert records, retrieve data, and perform analysis.

Categories of HQL

1. DDL (Data Definition Language)

Used to create database objects.

Commands:

- CREATE
- ALTER
- DROP
- TRUNCATE
- SHOW

Create Database

```
CREATE DATABASE College;
```

Show Databases

```
SHOW DATABASES;
```

Use Database

```
USE College;
```

Create Table

```
CREATE TABLE Student  
(  
  USN STRING,  
  Name STRING,  
  Marks INT  
);
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Describe Table

DESCRIBE Student;

Drop Table

DROP TABLE Student;

4.5 Data Loading

Data can be loaded into Hive from HDFS or the local file system.

Example:

LOAD DATA LOCAL INPATH
'/home/student.csv'

INTO TABLE Student;

Insert Data

INSERT INTO Student

VALUES

('1RV21CS001',

'Rahul',

85);

4.6 Retrieving Data

Display Entire Table

SELECT * FROM Student;

Display Selected Columns

SELECT Name,Marks

FROM Student;

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Where Clause

```
SELECT *  
  
FROM Student  
  
WHERE Marks>80;
```

Order By

```
SELECT *  
  
FROM Student  
  
ORDER BY Marks DESC;
```

Group By

```
SELECT Department,  
  
AVG(Salary)  
  
FROM Employee  
  
GROUP BY Department;
```

Having Clause

```
SELECT Department,  
  
COUNT(*)  
  
FROM Employee  
  
GROUP BY Department  
  
HAVING COUNT(*)>10;
```

Aggregate Functions

Function Purpose

COUNT() Counts records

SUM() Adds values

AVG() Calculates average

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Function Purpose

MIN() Finds minimum

MAX() Finds maximum

Example:

```
SELECT AVG(Salary)
```

```
FROM Employee;
```

Built-in Functions

String Functions

- LENGTH()
- CONCAT()
- LOWER()
- UPPER()
- SUBSTR()

Example

```
SELECT UPPER(Name)
```

```
FROM Student;
```

Mathematical Functions

- ROUND()
- FLOOR()
- CEIL()
- RAND()

Date Functions

- CURRENT_DATE
- YEAR()
- MONTH()
- DATEDIFF()

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Hive Joins

Inner Join

Returns matching records.

```
SELECT *  
FROM Employee e  
JOIN Department d  
ON e.DeptID=d.DeptID;
```

Left Outer Join

Returns all left table records.

Right Outer Join

Returns all right table records.

Full Outer Join

Returns all records from both tables.

Advantages of HQL

- Easy SQL syntax
- Supports Big Data
- Highly scalable
- Easy learning curve
- Supports complex queries
- Integrates with Hadoop

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Limitations

- High latency
- Not suitable for real-time processing
- No transaction support
- No row-level updates
- Batch-oriented execution

Applications of Hive

- Data Warehousing
- Banking Analytics
- Healthcare Data Analysis
- Retail Sales Analysis
- E-Commerce Recommendation Systems
- Log Processing
- Customer Behaviour Analysis
- Business Intelligence Reporting

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

10.1 What is Pig? (Apache Pig)

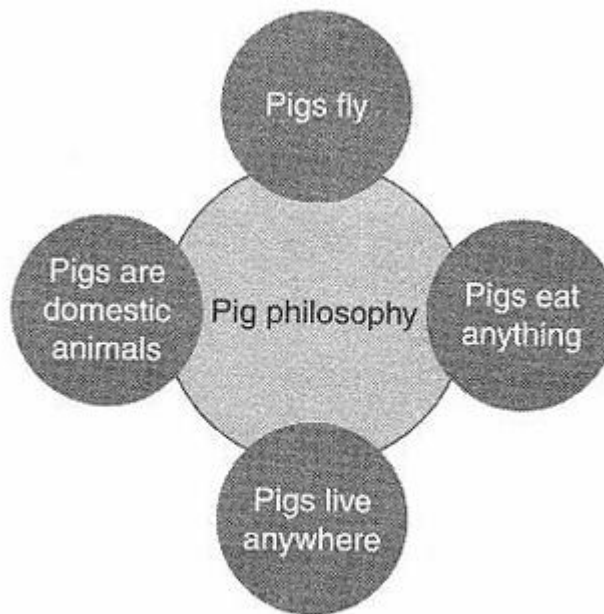


Figure 10.2 Pig philosophy.

10.1 What is Pig?

Definition

Apache Pig is a platform used to analyze and process large datasets stored in Hadoop. It provides a high-level scripting language called **Pig Latin**, which simplifies writing MapReduce programs.

Why was Pig Developed?

Writing MapReduce programs in Java is complex and time-consuming. Apache Pig was developed to make data processing easier by allowing programmers to write simple scripts instead of lengthy Java code.

Advantages

- Simplifies Hadoop programming.
- Reduces the amount of code.
- Suitable for processing huge datasets.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Automatically converts Pig scripts into MapReduce jobs.
- Supports structured, semi-structured, and unstructured data.

Key Features of Pig

According to the textbook, Pig provides the following features:

- High-level scripting language (Pig Latin).
- Automatically generates MapReduce programs.
- Easy to learn and use.
- Handles large volumes of data efficiently.
- Extensible using Java, Python, and other languages.
- Supports parallel processing on Hadoop clusters.
- Works well for ETL (Extract, Transform, Load) operations.

Pig Latin

Definition

Pig Latin is the scripting language of Apache Pig used to process large datasets in Hadoop.

Example:

```
A = LOAD 'student.txt';  
B = FILTER A BY marks > 50;  
DUMP B;
```

Here,

- LOAD → Reads data
- FILTER → Selects required records
- DUMP → Displays output

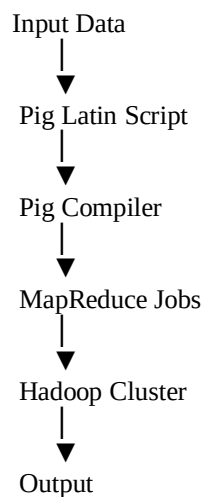
Applications of Pig

Apache Pig is widely used for:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Data analysis
- Log processing
- Data transformation
- ETL operations
- Data filtering
- Report generation
- Big data analytics

Pig Processing Flow



Anatomy of Pig

The textbook explains that the main components of Pig are:

1. Pig Latin Script

The user writes commands using Pig Latin.



2. Pig Interpreter / Compiler

Checks the syntax and converts Pig Latin into execution plans.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

3. Execution Engine

Optimizes the execution plan and generates MapReduce jobs.



4. Hadoop MapReduce

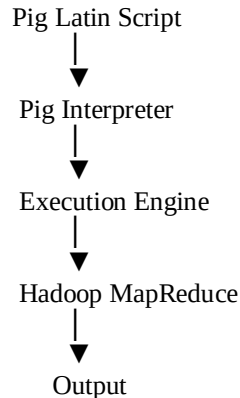
Executes the generated jobs on the Hadoop cluster.



5. Output

Results are stored in HDFS or displayed to the user.

Anatomy of Pig Diagram



Advantages of Apache Pig

- Easy programming model
- Less coding than Java MapReduce
- Automatic optimization
- Fault tolerant
- Highly scalable
- Supports parallel execution
- Good for batch processing
- Saves development time

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Limitations

- Not suitable for real-time processing.
- Depends on Hadoop.
- Slower than hand-written MapReduce in some cases.
- Mainly intended for batch analytics.

Difference between Pig and MapReduce

Pig	MapReduce
High-level language	Low-level Java programming
Easy to learn	Complex
Less code	More code
Faster development	Longer development time
Automatic optimization	Manual optimization