Big Data Analytics Introduction to Hive

What is Hive?

Data warehouse infrastructure build on top of Hadoop for querying and managing large data sets

Why Hive?

Hadoop is great!

MapReduce is very low level
Lack of expressiveness

Higher level data processing languages are needed

Hive Features

Designed for OLAP

On Line Analytical Processing

SQL type language for querying

HiveQL or HQL

It is familiar, fast, scalable, and extensible

Can plug in map/reduce scripts in language of choice

Hive is NOT

Relational database

Designed for Online Transaction Processing (OLTP)

*Online transaction processing, or **OLTP**, is a class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing (Wikipedia)*

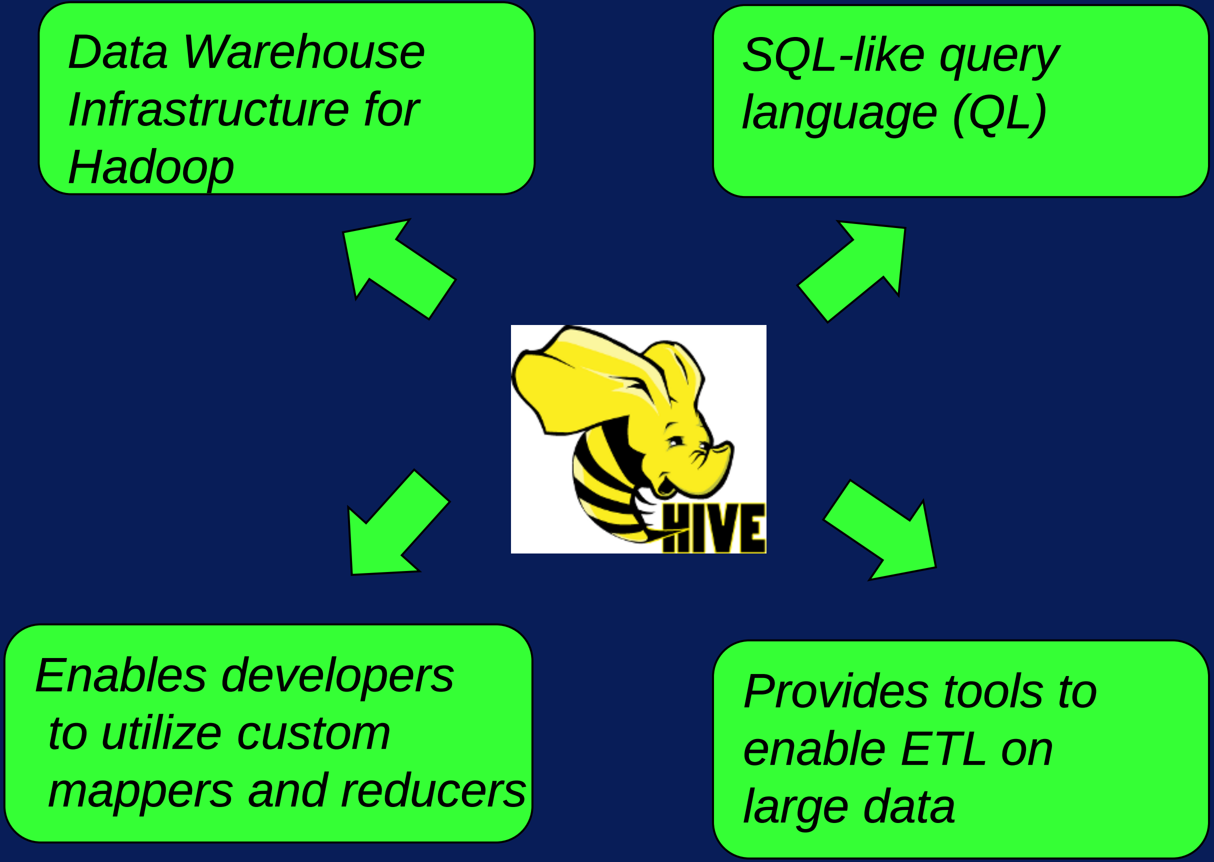
History

ETL =
Extract, Transform
and Load

Early Hive development work started at Facebook in 2007

Hive is an Apache project under Hadoop

http://hive.apache.org



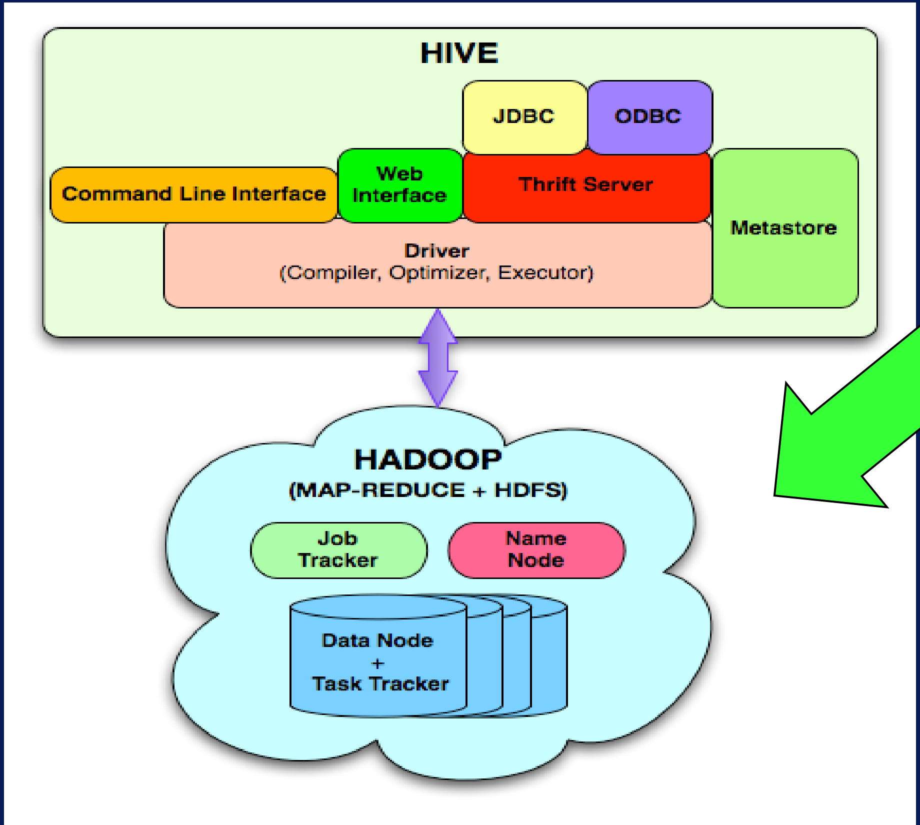
Data Warehouse Infrastructure for Hadoop

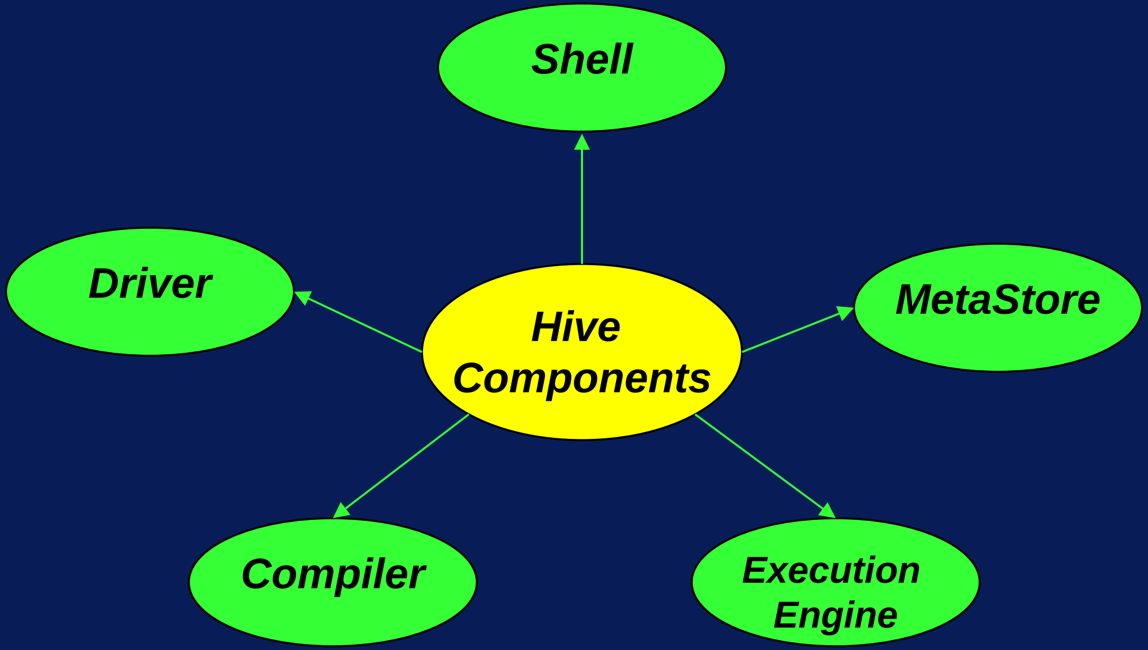
SQL-like query language (QL)

Enables developers to utilize custom mappers and reducers

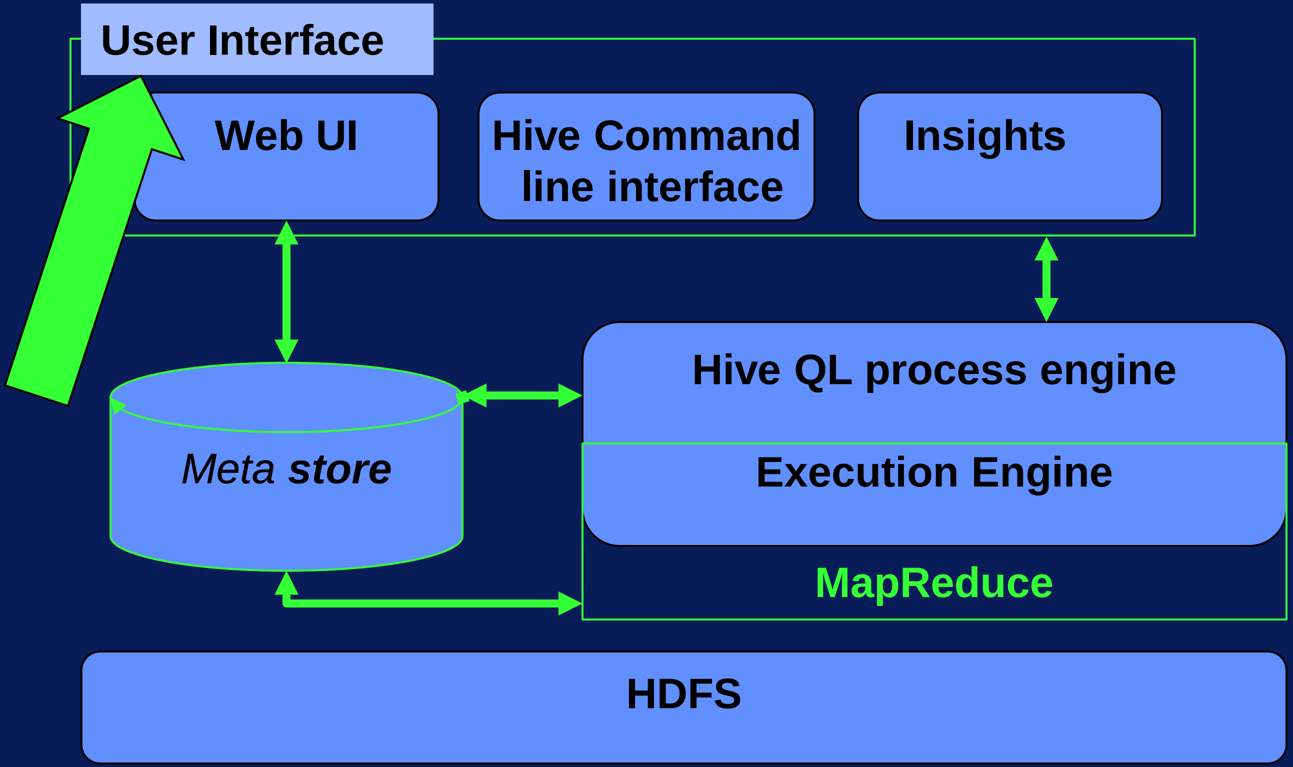
Provides tools to enable ETL on large data

Hive Architecture and Components

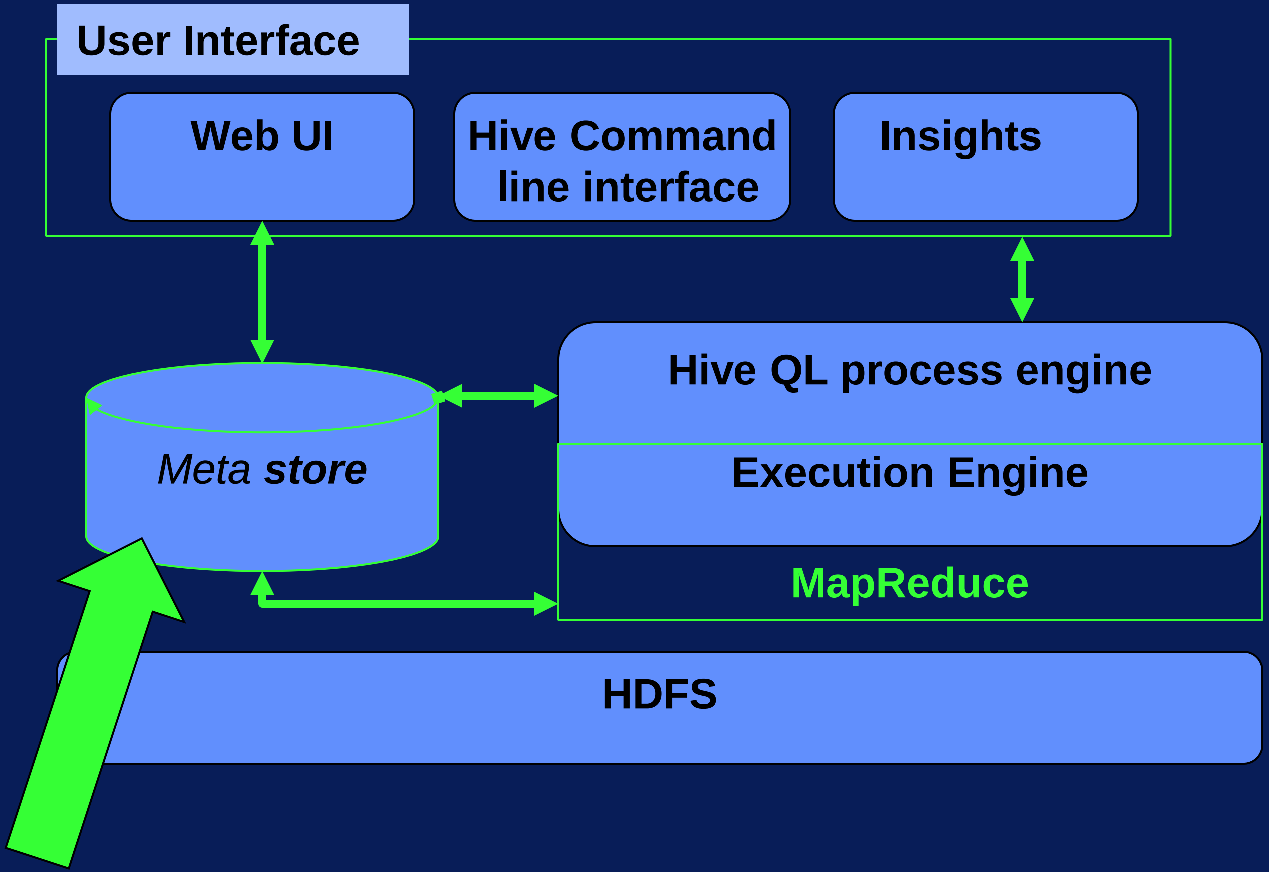



Shell
Driver
Hive Components
MetaStore
Compiler
Execution Engine

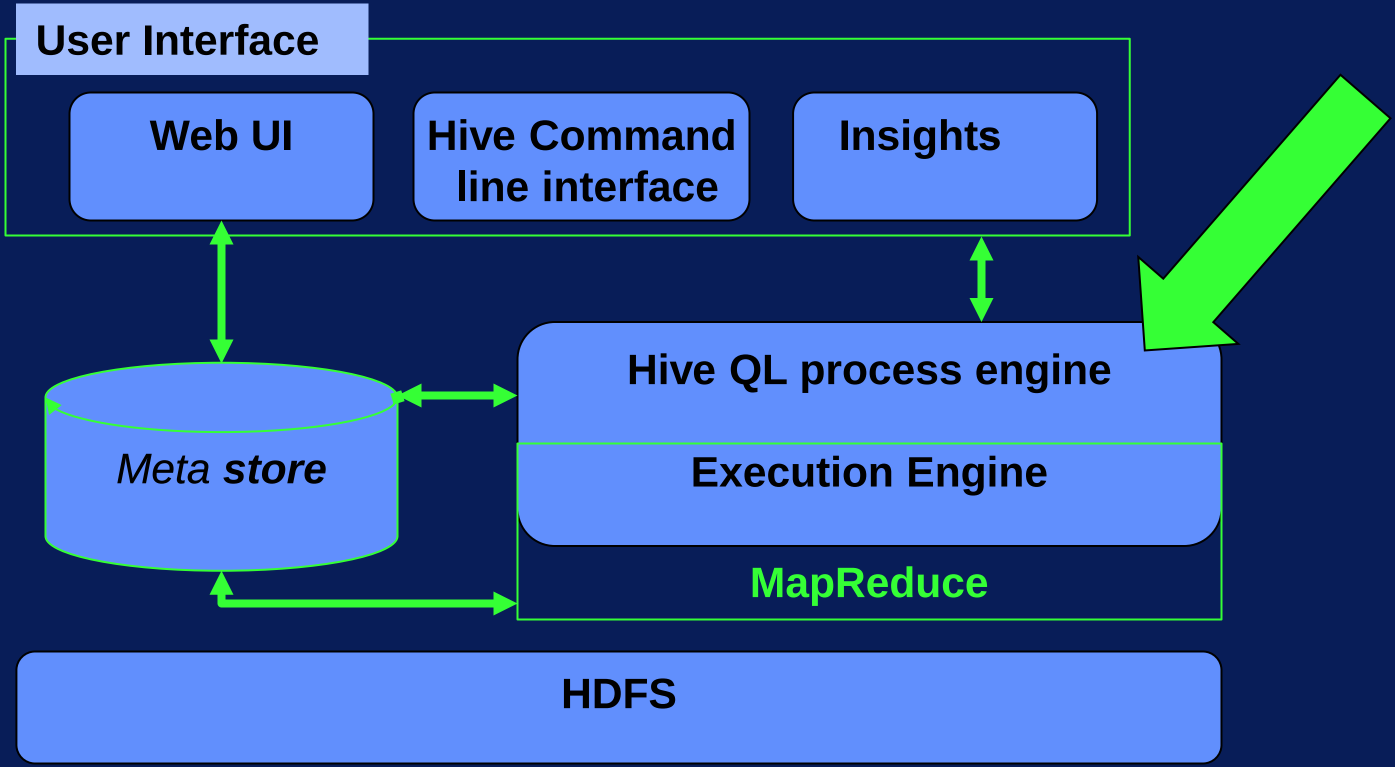
Hive Architecture



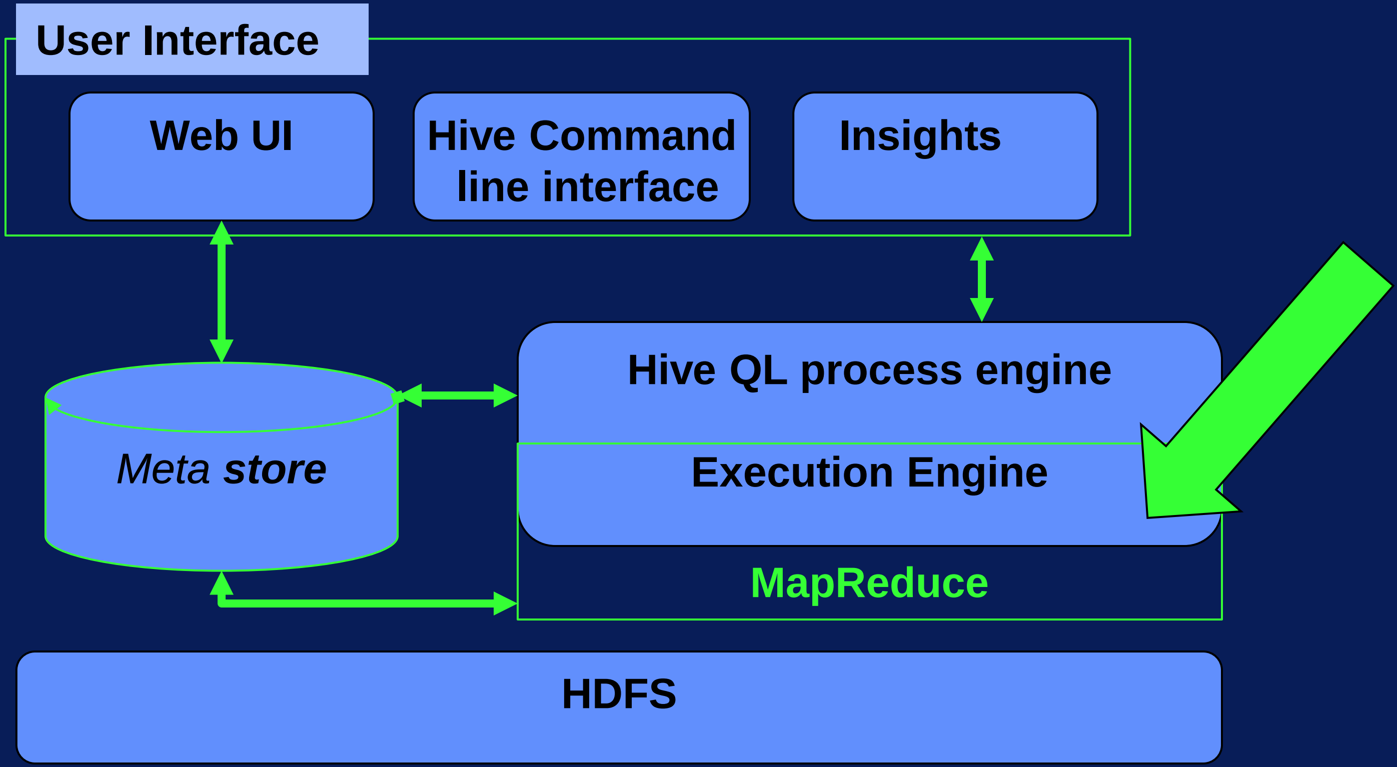
Hive Architecture

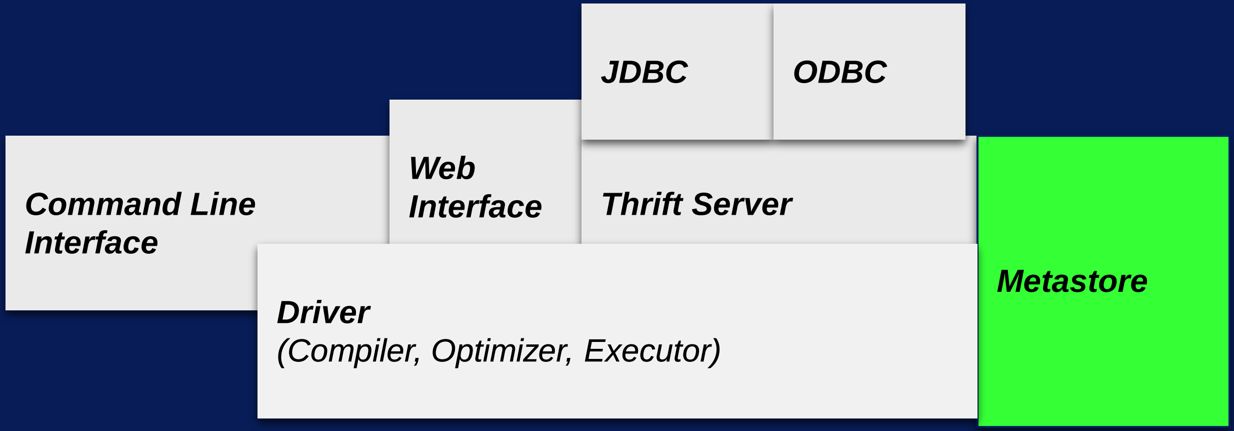


Hive Architecture



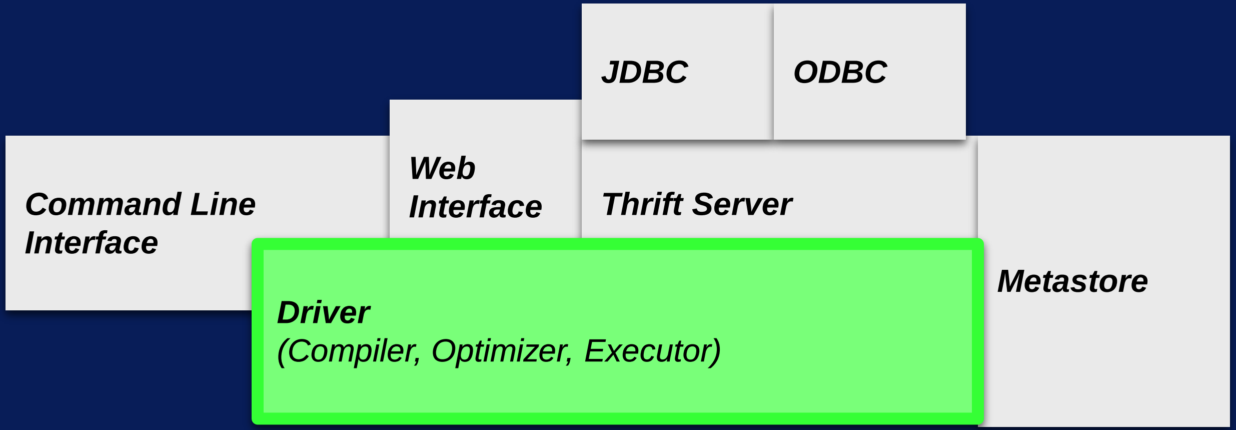
Hive Architecture





Metastore

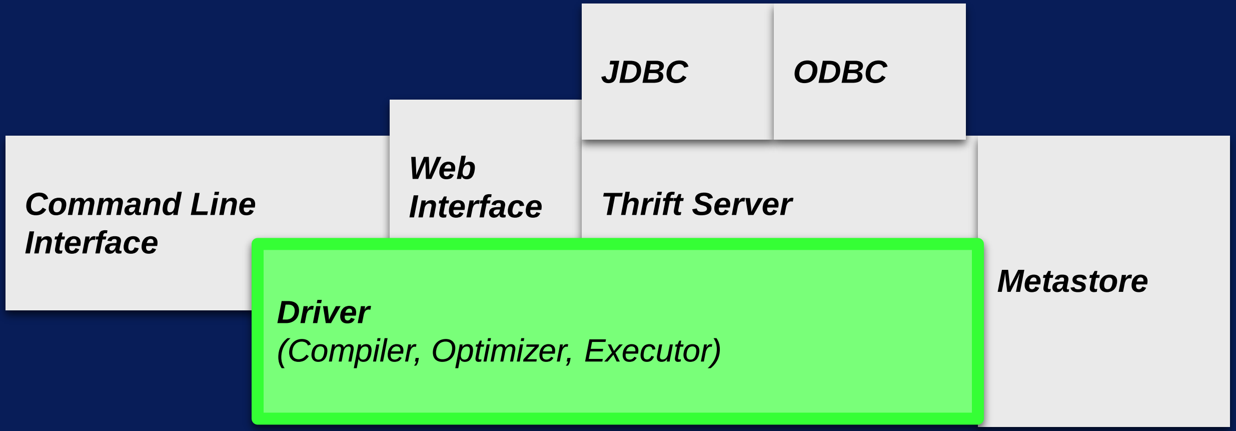
Stores the system catalog and meta data about tables, columns, partitions etc.



Driver

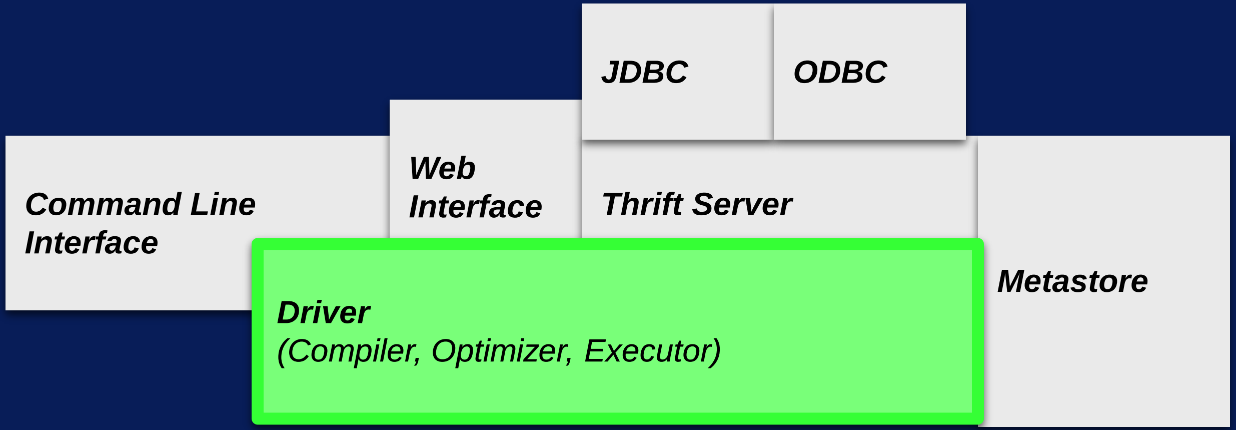
Manages the lifecycle of a HiveQL statement

Maintains a session handle and any session statistics



Query Compiler

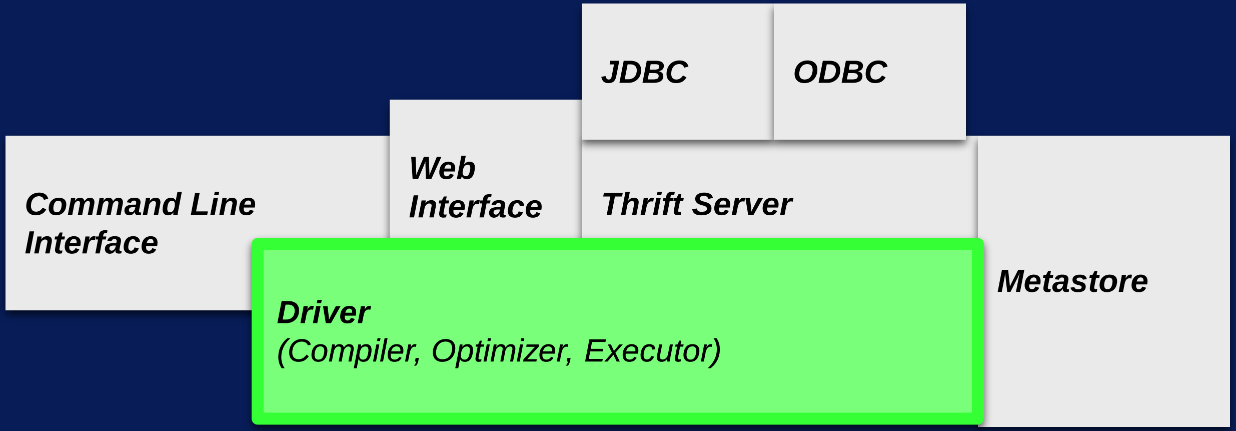
The component that compiles HiveQL into a directed acyclic graph of map/reduce tasks



Optimizer

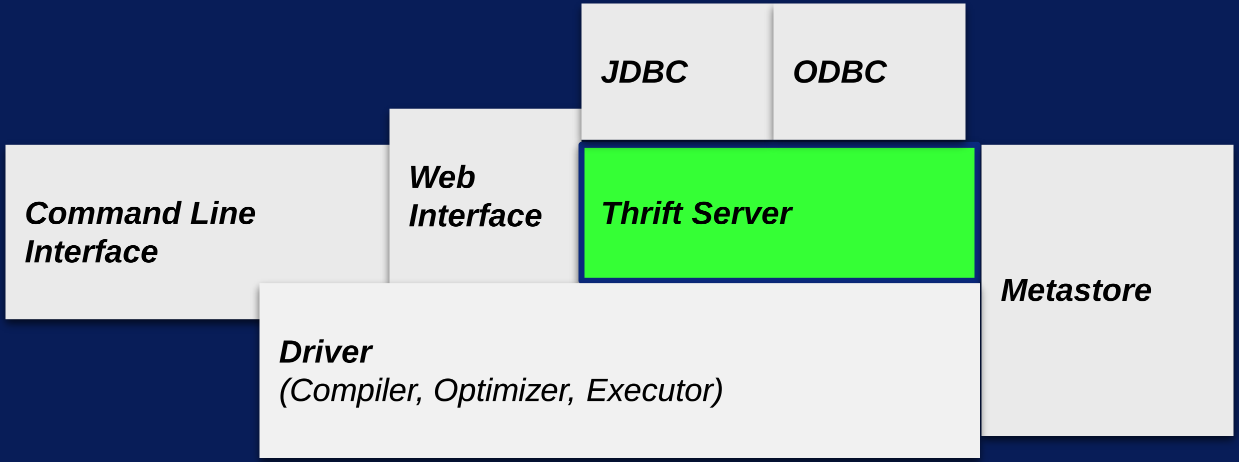
Consists of a chain of transformations

Performs Column Pruning , Partition Pruning, Repartitioning of Data



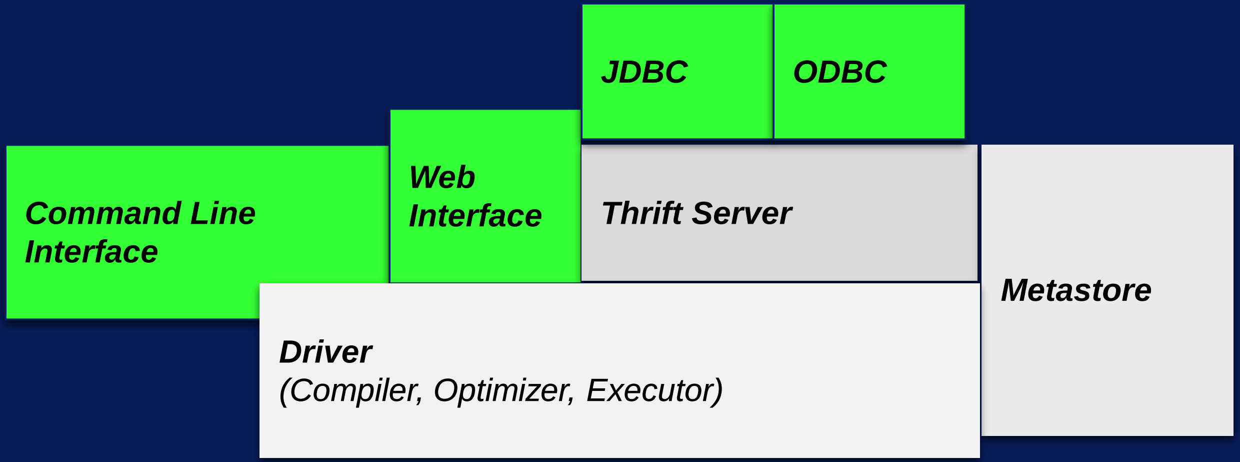
Execution Engine

- Executes the tasks produced by the compiler in proper dependency order
- Interacts with the underlying Hadoop instance



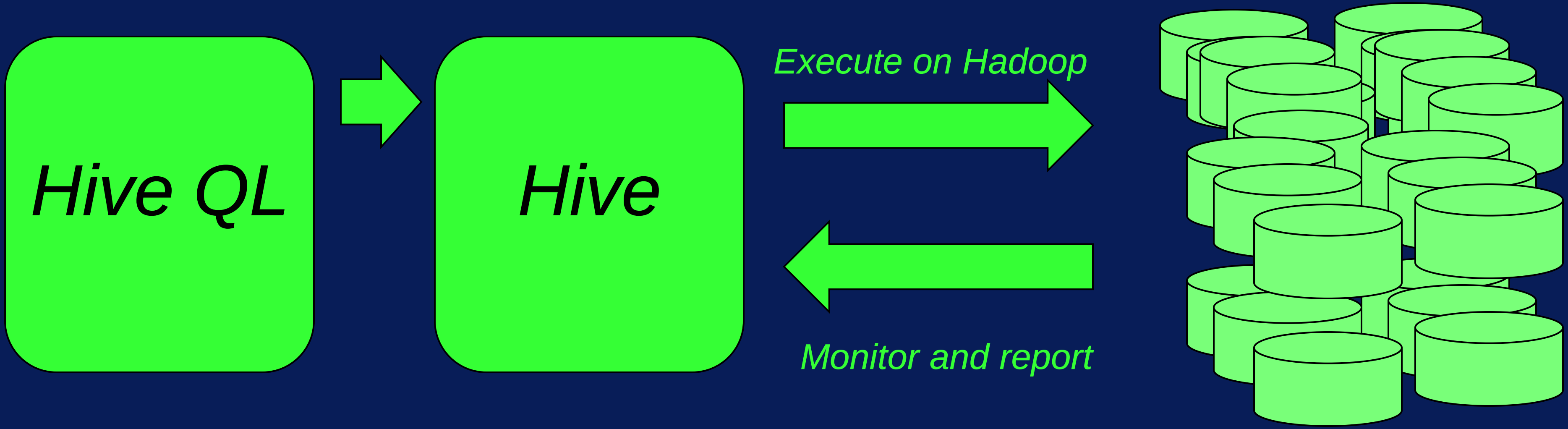
HiveServer

Provides a Thrift interface and a JDBC/ODBC server Enables Hive integration with other applications



Client Components

- Command Line Interface(CLI)
- Web UI
- JDBC/ODBC driver

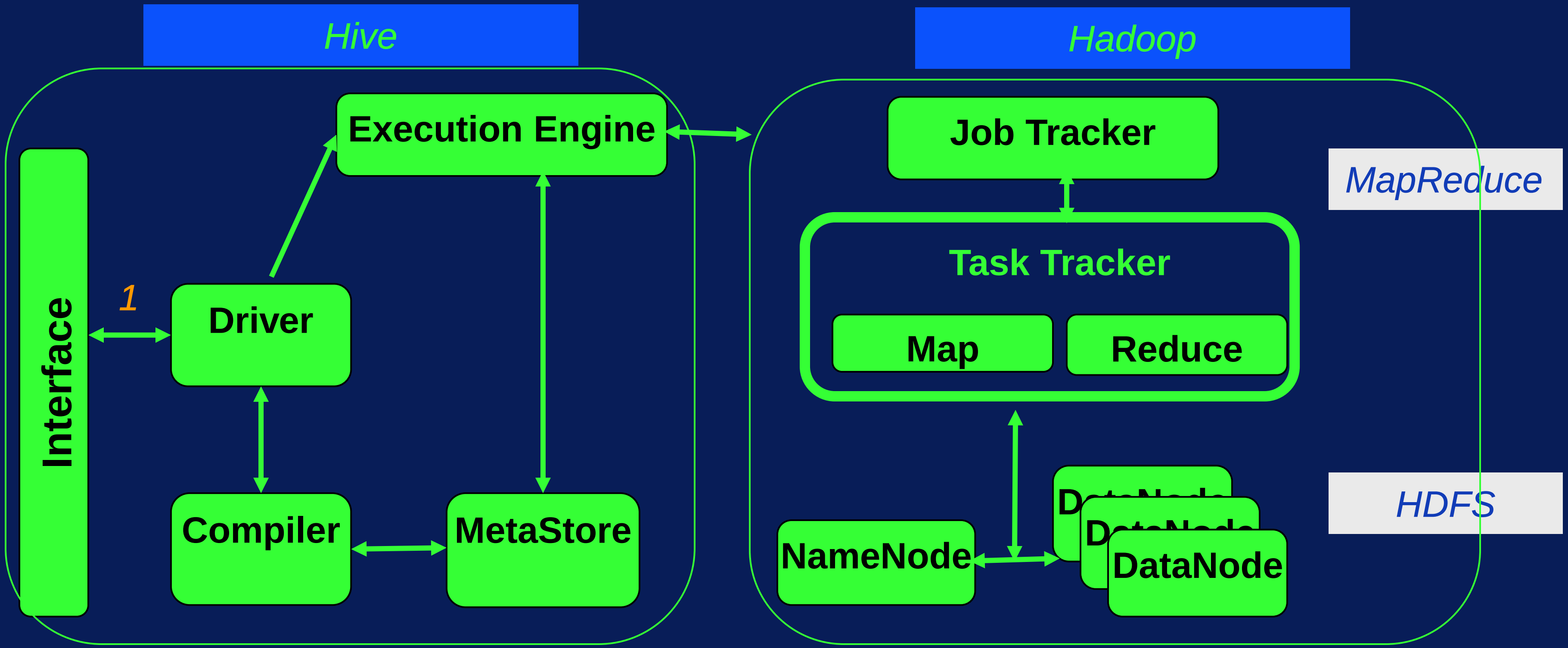


Hive QL

Hive

Execute on Hadoop

Monitor and report



Hive

Hadoop

Execution Engine

Interface

1

Driver

Compiler

MetaStore

Job Tracker

MapReduce

Task Tracker

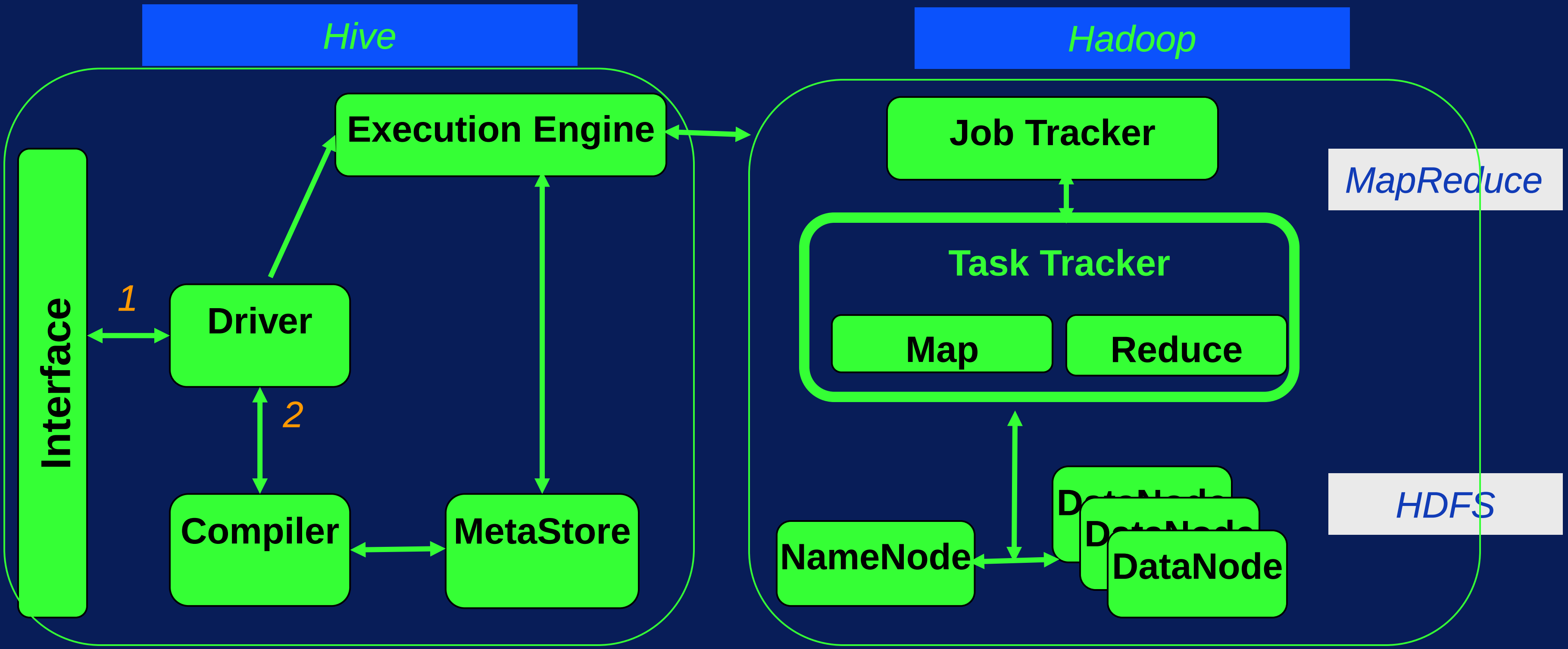
Map

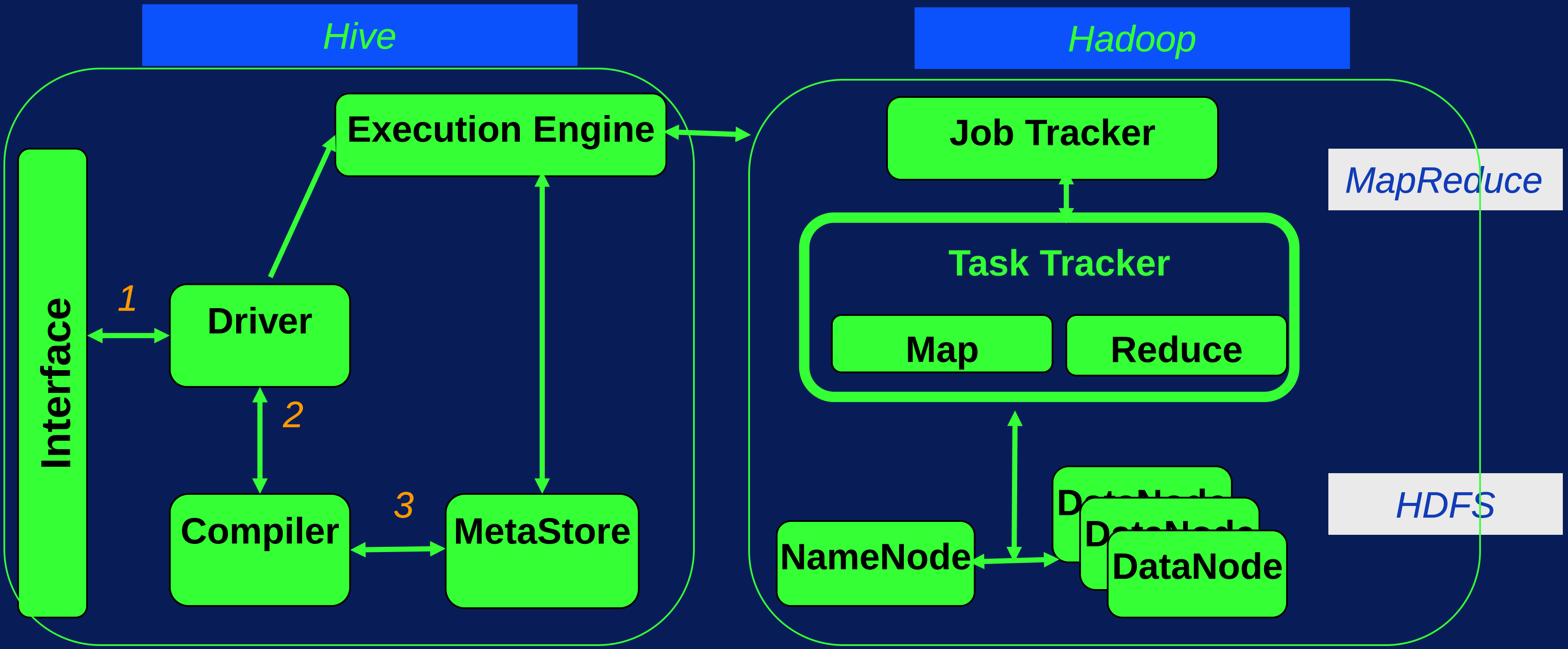
Reduce

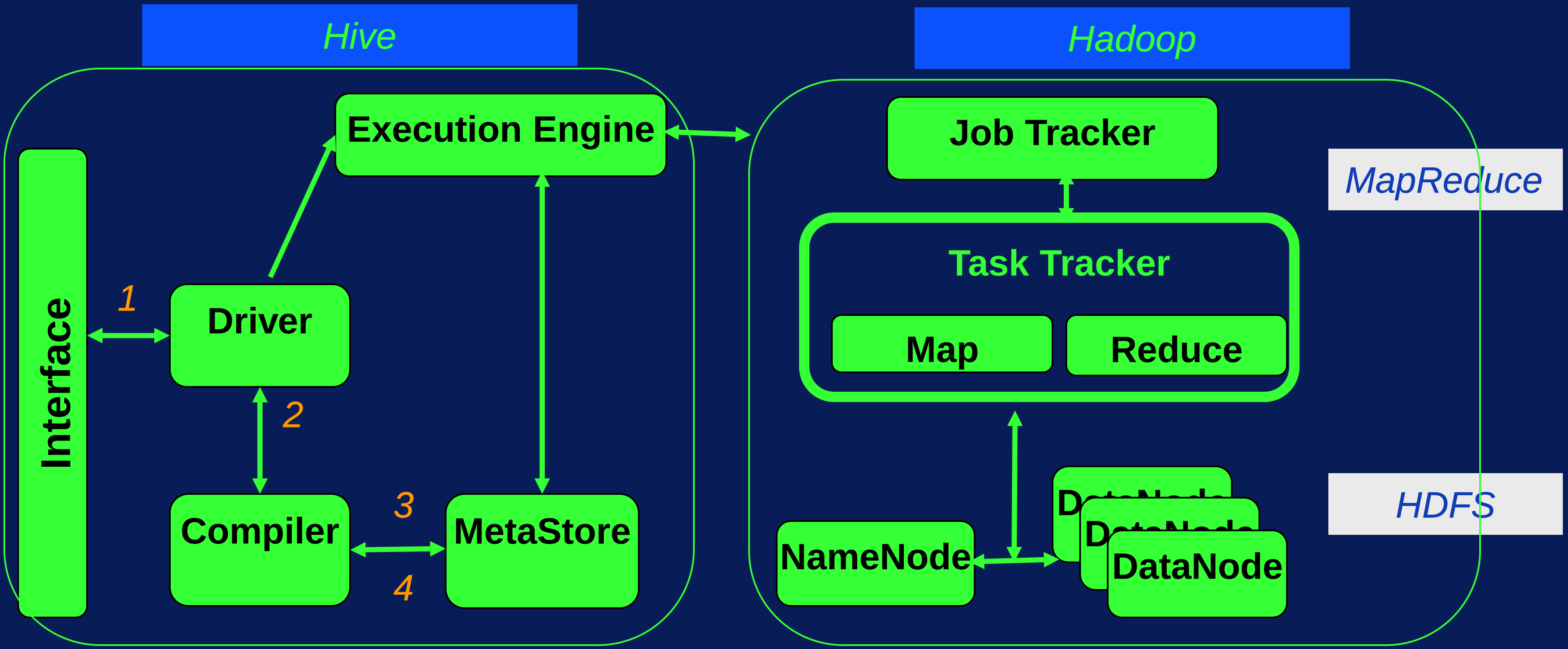
NameNode

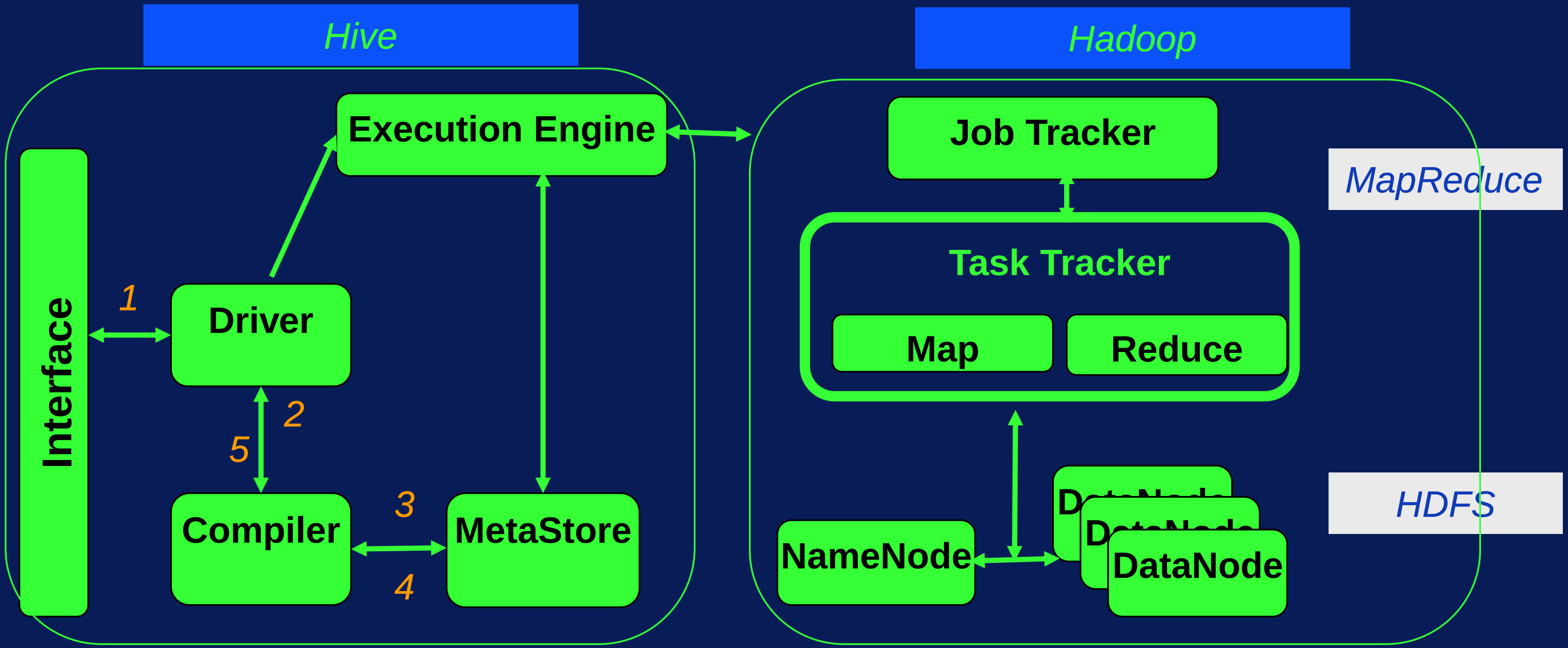
DataNode

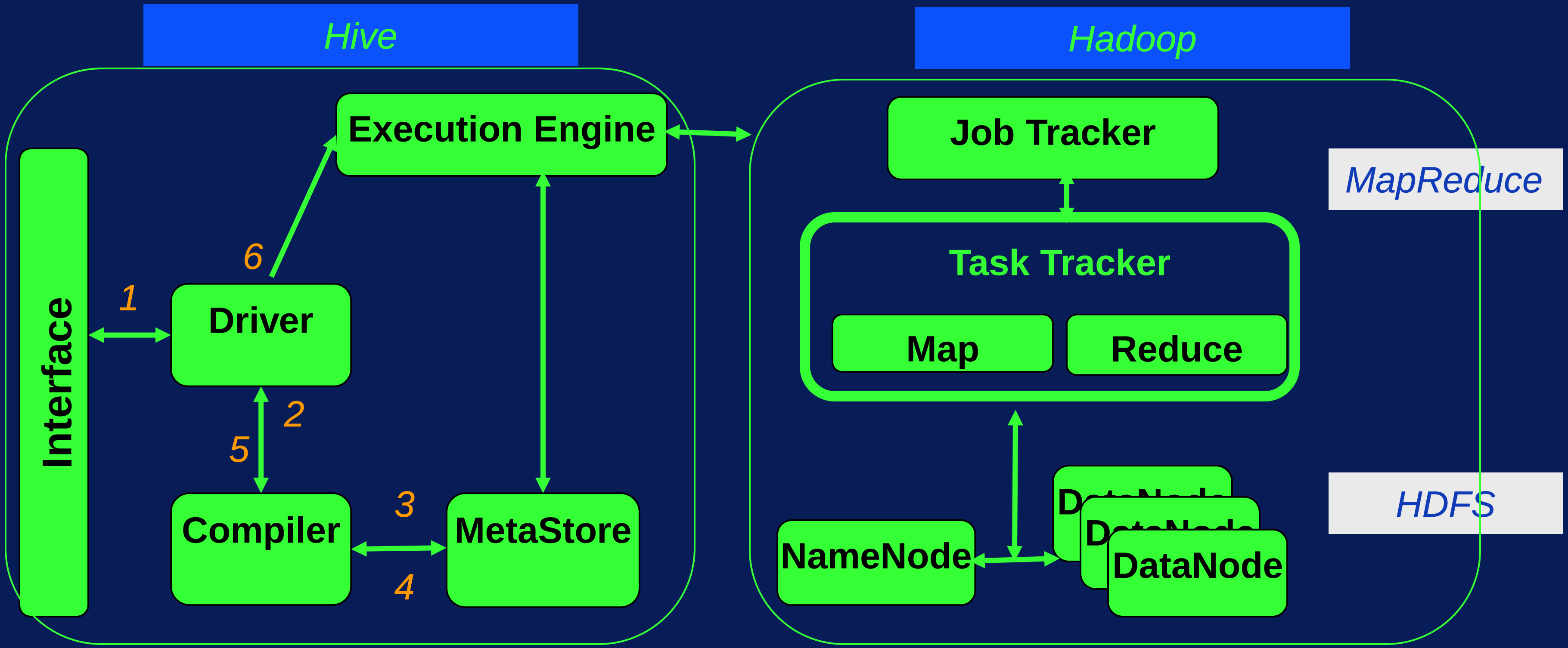
HDFS

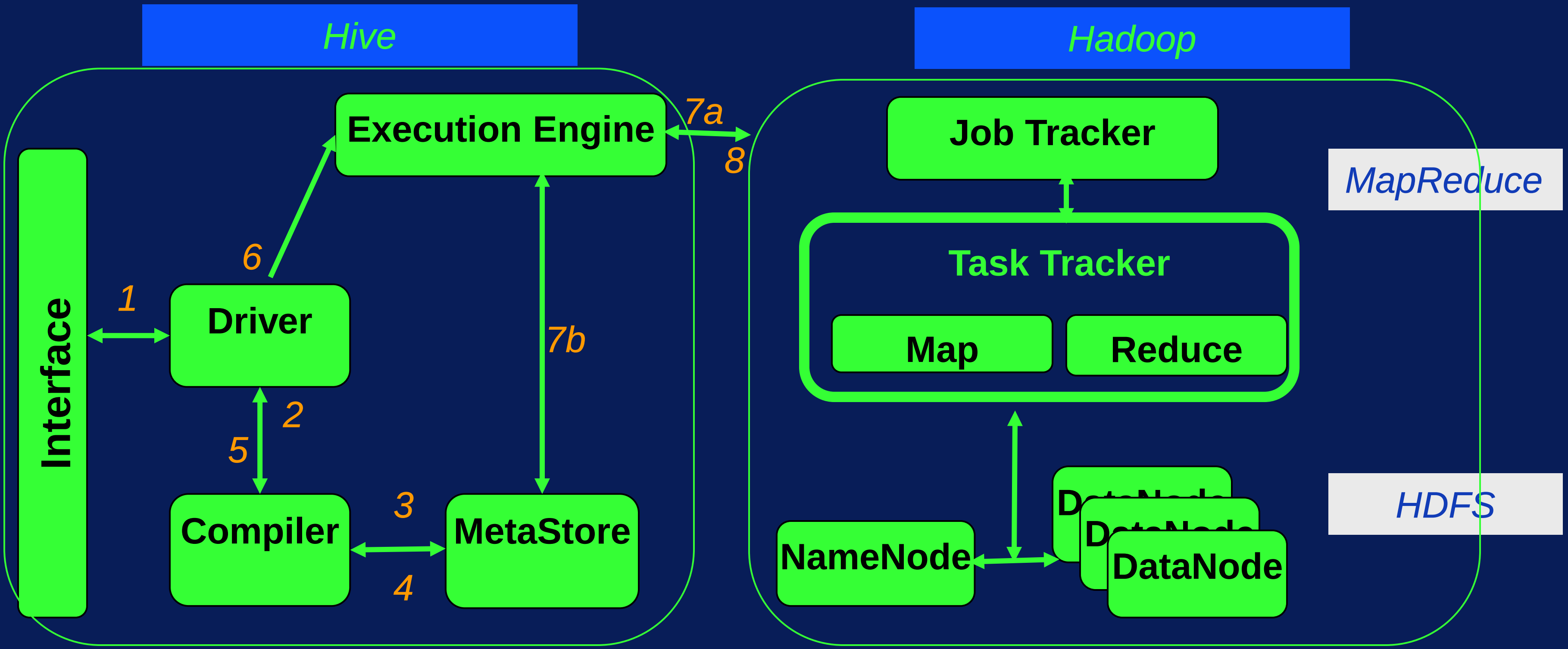
Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
Task Tracker
1
Driver
Map
Reduce
2
Compiler
MetaStore
DataNode
DataNode
NameNode
DataNode
HDFS

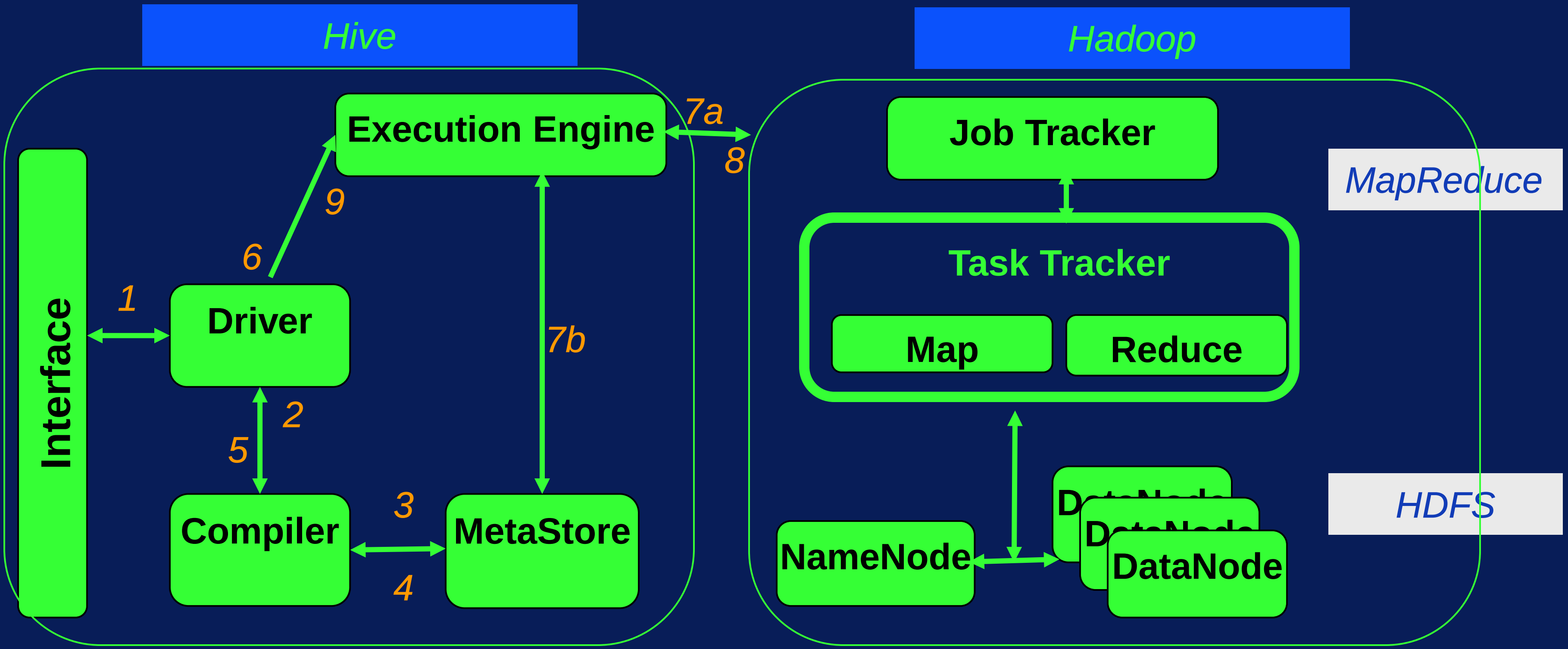

Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
1
Driver
Task Tracker
Map
Reduce
2
3
Compiler
MetaStore
DataNode
DataNode
DataNode
NameNode
HDFS

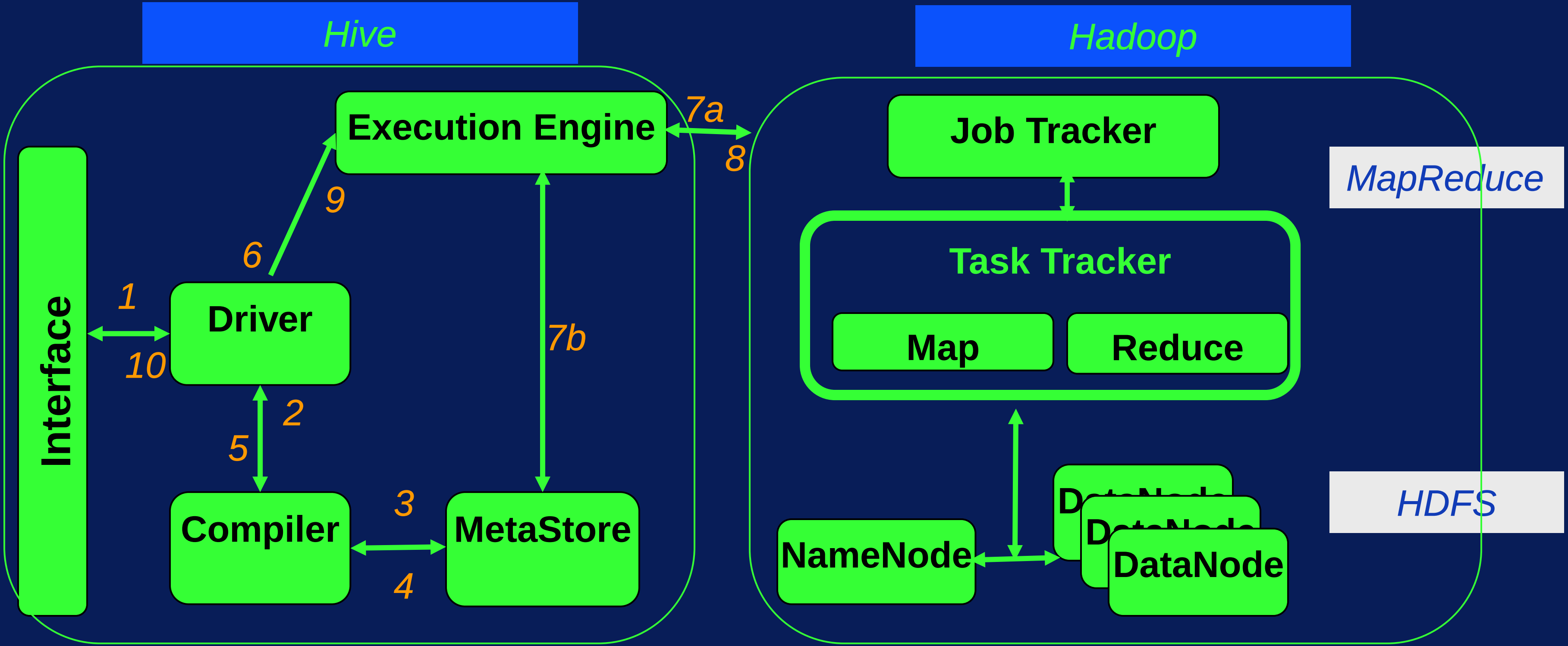

Hive
Hadoop
Execution Engine
Interface
1
Driver
2
Compiler
3
4
MetaStore
Job Tracker
MapReduce
Task Tracker
Map
Reduce
HDFS
NameNode
DataNode
DataNode
DataNode


Hive
Hadoop
Execution Engine
Interface
1
Driver
2
5
Compiler
3
4
MetaStore
Job Tracker
MapReduce
Task Tracker
Map
Reduce
HDFS
NameNode
DataNode
DataNode
DataNode


Hive
Hadoop
Execution Engine
Job Tracker
MapReduce
Interface
6
Task Tracker
1
Driver
Map
Reduce
2
5
3
Compiler
MetaStore
DataNode
DataNode
HDFS
NameNode
DataNode
4


Hive
Hadoop
Execution Engine
7a
8
Job Tracker
MapReduce
Interface
1
Driver
6
7b
Task Tracker
Map
Reduce
2
5
Compiler
3
4
MetaStore
NameNode
DataNode
DataNode
DataNode
HDFS


Hive
Hadoop
Execution Engine
7a
8
Job Tracker
MapReduce
9
6
Task Tracker
1
Interface
Driver
Map
Reduce
7b
2
5
3
Compiler
MetaStore
4
HDFS
NameNode
DataNode
DataNode
DataNode


Hive
Hadoop
Execution Engine
7a
8
Job Tracker
MapReduce
9
6
Interface
1
Driver
10
7b
Task Tracker
Map
Reduce
2
5
3
Compiler
MetaStore
4
DataNode
DataNode
NameNode
DataNode
HDFS

Hive’s Data Units

Databases

Tables

Partitions

Buckets (or clusters)

Very similar to SQL and Relational DBs

3-Levels: Tables → Partitions → Buckets

Data Model

Table maps to a HDFS directory

Partition maps to sub-directories under the table

Bucket maps to files under each partition

Tables

Similar to tables in relational DBs

Each table has corresponding directory in HDFS

Partitions

Analogous to dense indexes on partition columns

Nested sub-directories in HDFS for each combination of partition column values

Allows users to efficiently retrieve rows

Hive Data Structures

Traditional Database concepts

Supports primitive types

Additional types and structures

Hive Data Structures

Traditional database concepts

- Tables
- Rows
- Columns
- Partitions

Hive Data Structures

Basic types

- Integers
- Floats
- Doubles
- Strings

Hive File Formats

Hive enables users store different file formats

Performance improvements

- TEXTFILE
- SEQUENCEFILE
- ORC
- RCFILE

Optimized Row Columnar (ORC)

Record Columnar File - RCFILE

Hive Commands

Hive Interface

Command Line interface

Web interface or Hue

Java Database connectivity

Hive Commands

Database

Set of Tables - name conflicts resolution

Table

Set of Rows - have the same columns

Row

A single record - a set of columns

Column

Value and type for a single value

Tables Commands

- SHOW TABLES
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

Hive Commands

CREATE TABLE mytable (myint INT, bar STRING)
PARTITIONED BY (ds STRING);

SHOW TABLES '.*my';

ALTER TABLE mytable **ADD COLUMNS** (new_col INT);

DROP TABLE mytable;

A table in Hive is an HDFS directory in Hadoop

Hive Commands

Schema is known at creation time (like DB schema)

Partitioned tables have “sub-directories”, one for each partition

```
CREATE TABLE mypeople (
id                int,
name              string
)
partitioned by (date string)
```

Hive Query Language

JOIN

```
SELECT t1.a1 as c1, t2.b1 as c2
FROM t1 JOIN t2 ON (t1.a2 = t2.b2);
```

INSERTION

```
        INSERT OVERWRITE TABLE t1
       SELECT *  FROM t2;
```

Format Rows

```
CREATE TABLE mypeople (id INT, name STRING)
ROW FORMAT
DELIMETED FIELDS TERMINATED BY <output format>
LINES TERMINATED BY ‘\n’;
```

Loading Data into HIVE

HDFS

```
LOAD DATA INPATH ‘mybigdata’
        [OVERWRITE] INTO TABLE mypeople;
```

Local file system

```
LOAD DATA LOCAL INPATH ‘mybigdata’
        INTO TABLE mypeople;
```

Partitions

LOAD DATA INPATH ‘myweblogs’ INTO TABLE mypeople PARTITION (dt=12-12-2020);

BUCKETS

Set hive.enforce.bucketing property to true

```
CREATE TABLE mycustomers(id INT, purchases DOUBLE,
name STRING)
CLUSTERED BY id into 32 BUCKETS;
```

BUCKETS

```
SELECT min(cost) FROM mysales
TABLESAMPLE (BUCKET 10 OUT OF 32 ON
rand());
```

VIEWS

Similar to SQL Views

- Virtual table in Metastore
- SHOW TABLES

JOINS

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT FROM
CUSTOMERS c JOIN ORDERS o ON
(c.ID = o.CUSTOMER_ID);
```

DROP TABLE

- DROP TABLE MyCustomers;

DELETE PARTITION

```
ALTER TABLE MyCustomers DROP
PARTITION (col2=100);
```