

بسم الله الرحمن الرحيم

Apache Drill



امیرحسین بنایی خلیل آباد

درس مباحث ویژه ۱

نام استاد

ابوالفضل گندمی

What is Apache Drill

- Drill is an Apache open-source SQL query engine for Big Data exploration. Drill is designed from the ground up to support high-performance analysis on the semi-structured and rapidly evolving data coming from modern Big Data applications, while still providing the familiarity and ecosystem of ANSI SQL, the industry-standard query language. Drill provides plug-and-play integration with existing Apache Hive and Apache HBase deployments.

Apache Drill Key Features

- Low-latency SQL queries
- Dynamic queries on self-describing data in files (such as JSON, Parquet, text) and HBase tables, without requiring metadata definitions in the Hive metastore.
- ANSI SQL
- Nested data support
- Integration with Apache Hive (queries on Hive tables and views, support for all Hive file formats and Hive UDFs)

What is NoSQL?

- NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

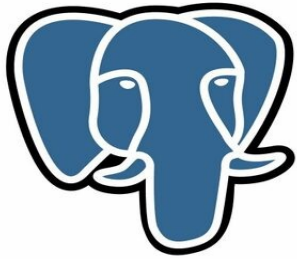
Student Table in SQL Server

teachoo

TABLE STUDENT

RollNo	Name	Class	DOB	Gender	City	Marks
1	Nanda	X	1995-06-06	M	Agra	551
2	Saurabh	XII	1993-05-07	M	Mumbai	462
3	Sonal	XI	1994-05-06	F	Delhi	400
4	Trisla	XII	1995-08-08	F	Mumbai	450
5	Store	XII	1995-10-08	M	Delhi	369
6	Marisla	XI	1994-12-12	F	Dubai	250
7	Neha	X	1995-12-08	F	Moscow	377
8	Nishant	X	1995-06-12	M	Moscow	489

Student Table in SQL Server



Microsoft®
SQL Server®



ORACLE

JSON format in NoSQL

```
1  {  
2    "_id": 1,  
3    "first_name": "Leslie",  
4    "last_name": "Yepp",  
5    "cell": "8125552344",  
6    "city": "Pawnee",  
7    "hobbies": ["scrapbooking", "eating waffles", "working"]  
8  }
```


What is wide-column in NoSQL?

Row A	Column 1	Column 2	Column 3
	Value	Value	Value
Row B	Column 1	Column 2	Column 3
	Value	Value	Value

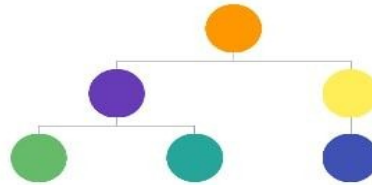
What is graph in NoSQL?

SQL Database
(Relational)

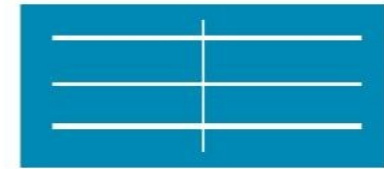


NoSQL Databases (Non-Relational)

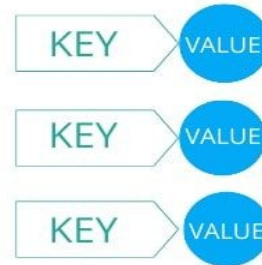
Document-oriented



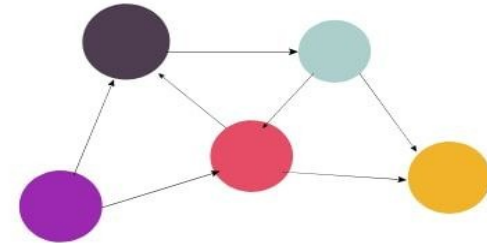
Column-oriented










Key-Value



Graph



NoSQL

Type	Example	
Key-Value Store	 redis	 riak
Wide Column Store	 HBASE	 <i>cassandra</i>
Document Store	 mongoDB	 CouchDB <i>relax</i>
Graph Store	 Neo4j	InfiniteGraph The Distributed Graph Database

Why NoSQL?

Flexible data model and schema

NoSQL databases store many different types of data and offer flexible schemas, ideal for semi-structured and unstructured data. You can easily adapt them to new types of data and evolve the schema to meet any changing data requirements.

Massive data storage

NoSQL is designed to handle large, complex datasets, allowing organizations to adopt and scale big data, real-time analytics, and IoT use cases.

Agile development

The flexibility of NoSQL complements agile app development. NoSQL databases can store many types of data in its native format and allows the data model to be defined and adapted as you go, so developers can get going fast, spend less time on data transformation, and iterate quickly.

High availability

NoSQL data architectures are distributed by design and have no single point of failure. They also provide easy replication, making them more resistant to unplanned outages and disruptions.

Scalability

Unlike relational databases, NoSQL databases make it easy to increase capacity as data and traffic grows—in most cases, with zero downtime. Cloud-based databases are even easier to scale based on demand, offering autoscaling features and flexible pricing models.

Faster queries

Unlike relational databases, which are normalized to reduce data duplication, NoSQL is optimized for fast querying. It typically does not require complex joins, meaning that database queries return results more quickly.

Install Drill Introduction

2 ways

1: embedded mode

2: distributed mode

Install Drill Introduction on embedded mode

- You can install Drill for use in either embedded mode or distributed mode. Choose embedded mode to use Drill only on a single node. Installing Drill for use in embedded mode does not require installation of ZooKeeper. Using Drill in embedded mode requires no configuration.

Install Drill Introduction on distributed mode

- Choose distributed mode to use Drill in a clustered Hadoop environment. A clustered (multi-server) installation of ZooKeeper is one of the prerequisites. You also need to configure Drill for use in distributed mode. After you complete these tasks, connect Drill to your Hive, HBase, or distributed file system data sources, and run queries on them.

Embedded Mode Prerequisites

Before you install Drill, ensure that the machine meets the following prerequisites:

- Linux, Mac OS X, and Windows: Oracle or OpenJDK 8
- Windows only:
 - A `JAVA_HOME` environment variable that points to the JDK installation
 - A `PATH` environment variable that includes a pointer to the JDK installation
- A utility for unzipping a tar.gz file

Running Drill on Docker

- You can start and run a Docker container in detached mode or foreground mode. Detached mode runs the container in the background. Foreground is the default mode. Foreground mode runs the Drill process in the container and attaches the console to Drill's standard input, output, and standard error.

Running the Drill Docker Container in Foreground Mode

- `docker run -it --name drill \`
 `-p 8047:8047 \ # web and REST`
 `-p 31010:31010 \ # JDBC`
 `apache/drill`

Running the Drill Docker Container in Detached Mode

- `$ docker run --name drill \`
 `-p 8047:8047 \ # web and REST`
 `-p 31010:31010 \ # JDBC`
 `--detach`
 `apache/drill`

`$ docker exec -it drill /bin/bash`

`$ $DRILL_HOME/bin/drill-embedded`

Confirm Install Drill

```
Apache Drill 1.19.0  
"json ain't no thang"  
apache drill>
```

```
SELECT version FROM sys.version;
```

Confirm Install Drill

```
~ : docker — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
amirhbana@debianhome:~$ docker run -it --name drill -p 8047:8047 -p 31010:31010 apache/drill
Apache Drill 1.21.1
"Everything is easier with Drill."
apache drill> SELECT version FROM sys.version;
+-----+
| version |
+-----+
| 1.21.1   |
+-----+
1 row selected (2.677 seconds)
apache drill> 
```

SELECT first_name, last_name FROM cp.`employee.json` LIMIT 1;

```
~ : docker — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View
+-----+
| Sheri   | Nowmer   |
+-----+
1 row selected (0.546 seconds)
apache drill> cle
2..semicolon>
3..semicolon> ;
Error: PARSE ERROR: Non-query expression encountered in illegal context

SQL Query: cle
      ^

[Error Id: 84c23a0b-9a5b-42ef-8482-fd928044c70e ] (state=,code=0)
apache drill> SELECT first_name, last_name FROM cp.`employee.json` LIMIT 1;
+-----+
| first_name | last_name |
+-----+
| Sheri      | Nowmer    |
+-----+
1 row selected (0.252 seconds)
apache drill> 
```

Architecture Introduction

Apache Drill is a low latency distributed query engine for large-scale datasets, including structured and semi-structured/nested data.

Drill is also useful for short, interactive ad-hoc queries on large-scale data sets. Drill is capable of querying nested data in formats like JSON and Parquet and performing dynamic schema discovery. Drill does not require a centralized metadata repository.

High-Level Architecture

Drill includes a distributed execution environment, purpose built for large- scale data processing. At the core of Apache Drill is the “Drillbit” service, which is responsible for accepting requests from the client, processing the queries, and returning results to the client.

High-Level Architecture

A Drillbit service can be installed and run on all of the required nodes in a Hadoop cluster to form a distributed cluster environment. When a Drillbit runs on each data node in the cluster, Drill can maximize data locality during query execution without moving data over the network or between nodes. Drill uses ZooKeeper to maintain cluster membership and health-check information.

High-Level Architecture

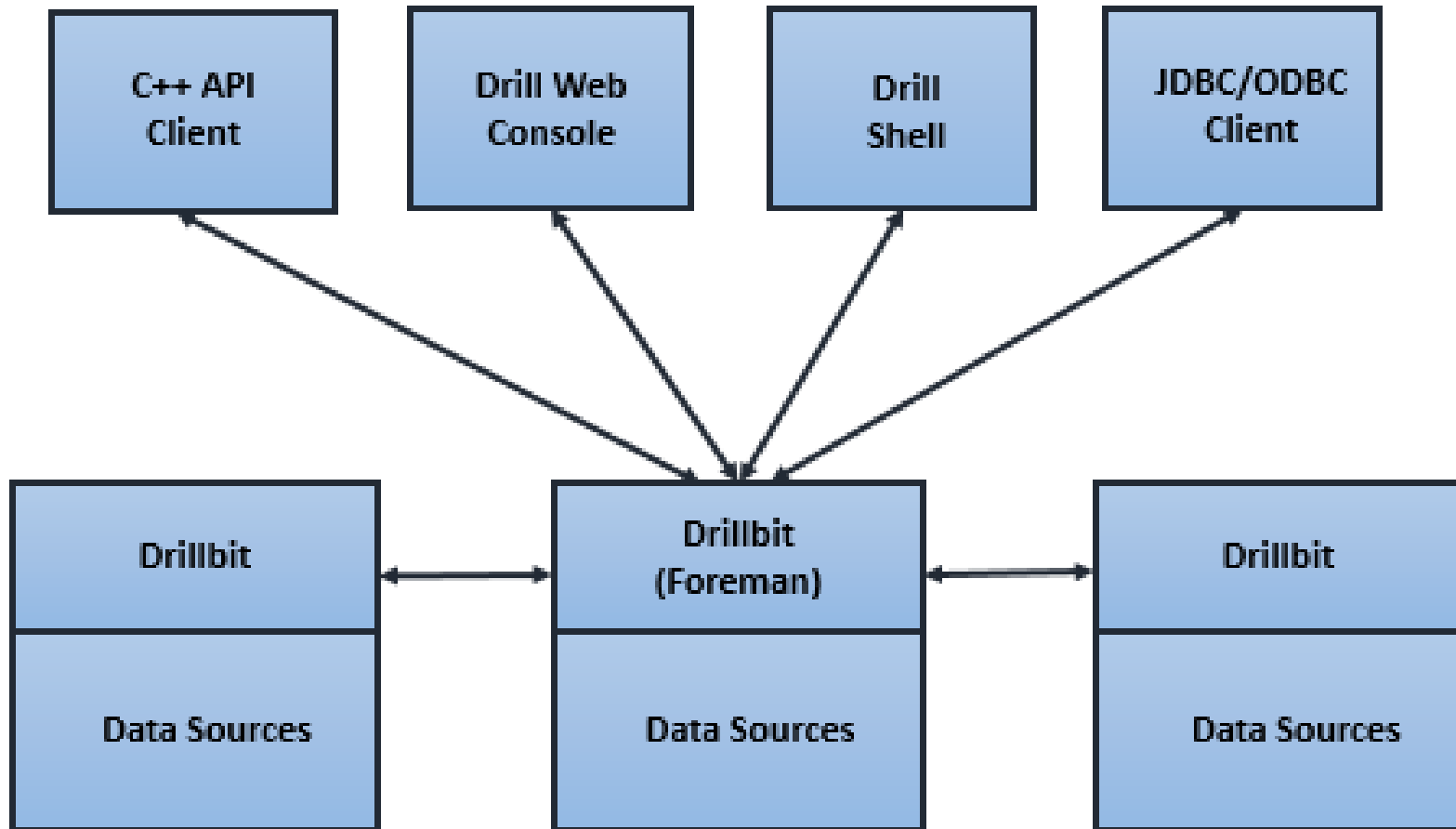
Though Drill works in a Hadoop cluster environment, Drill is not tied to Hadoop and can run in any distributed cluster environment. The only pre-requisite for Drill is ZooKeeper.

Drill Query Execution

When you submit a Drill query, a client or an application sends the query in the form of an SQL statement to a Drillbit in the Drill cluster.

A Drillbit is the process running on each active Drill node that coordinates, plans, and executes queries, as well as distributes query work across the cluster to maximize data locality.

The following image represents the communication between clients, applications, and Drillbits

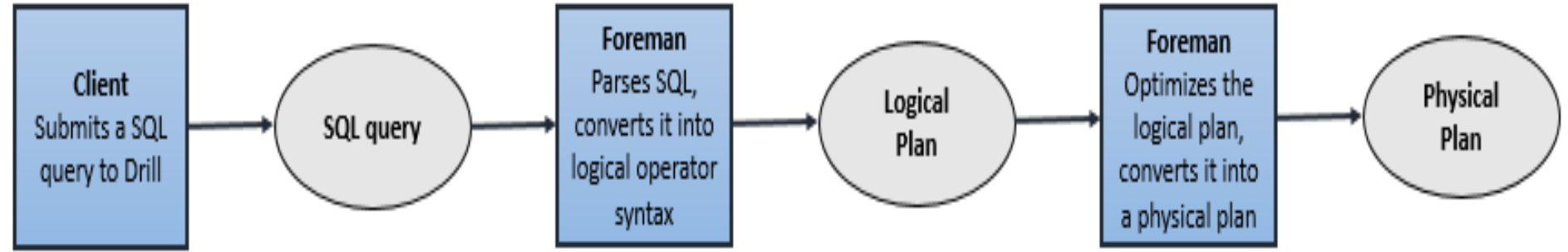


Drill Query Execution

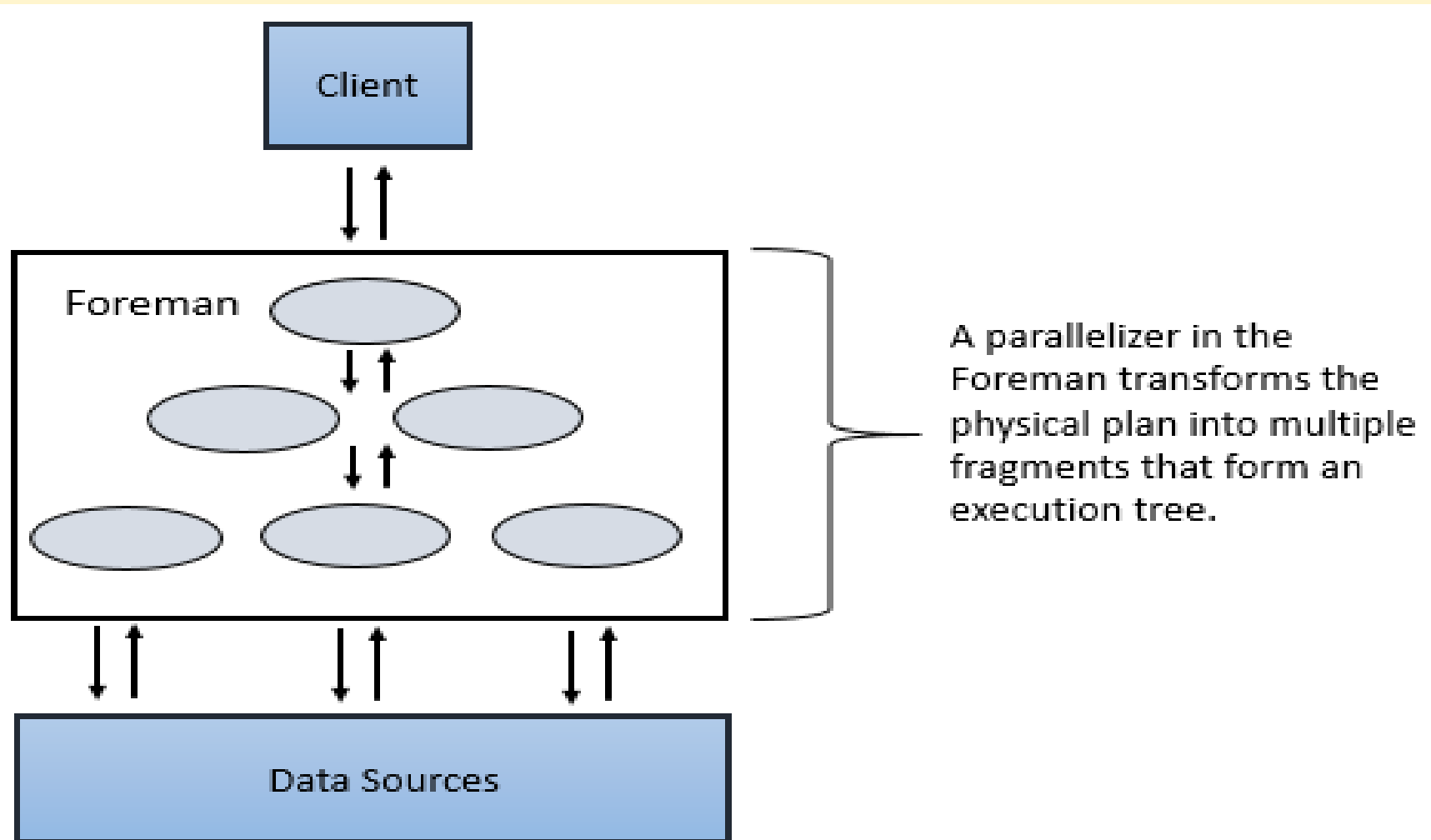
The Drillbit that receives the query from a client or application becomes the Foreman for the query and drives the entire query. A parser in the Foreman parses the SQL, applying custom rules to convert specific SQL operators into a specific logical operator syntax that Drill understands. This collection of logical operators forms a logical plan. The logical plan describes the work required to generate the query results and defines which data sources and operations to apply.

Drill Query Execution

The Foreman sends the logical plan into a cost-based optimizer to optimize the order of SQL operators in a statement and read the logical plan. The optimizer applies various types of rules to rearrange operators and functions into an optimal plan. The optimizer converts the logical plan into a physical plan that describes how to execute the query.

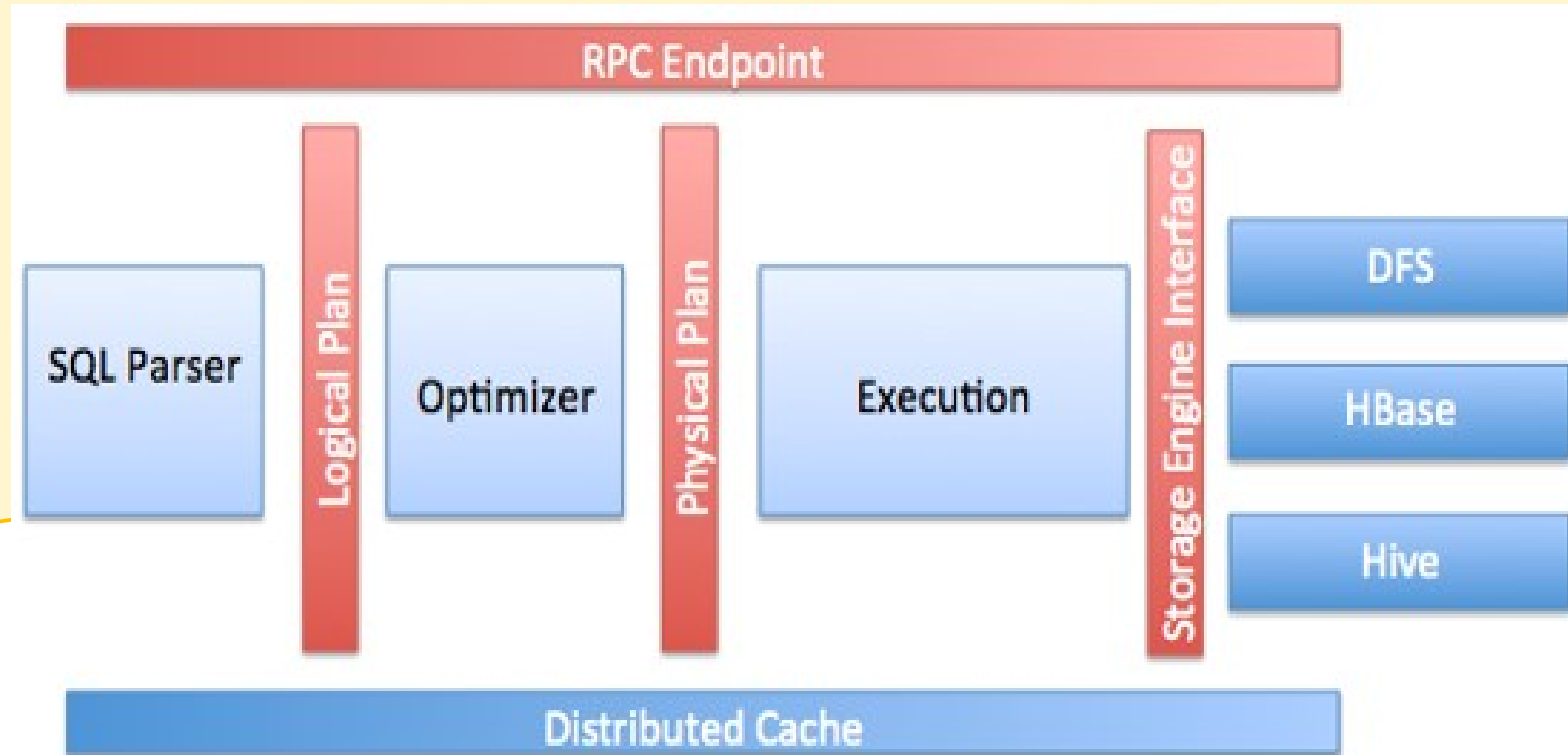


A parallelizer in the Foreman transforms the physical plan into multiple phases, called major and minor fragments. These fragments create a multi-level execution tree that rewrites the query and executes it in parallel against the configured data sources, sending the results back to the client or application.



Core Modules

The following image represents components within each Drillbit:



RPC endpoint

Drill exposes a low overhead protobuf-based RPC protocol to communicate with the clients.

Additionally, C++ and Java API layers are also available for client applications to interact with Drill.

Clients can communicate with a specific Drillbit directly or go through a ZooKeeper quorum to discover the available Drillbits before submitting queries.

It is recommended that the clients always go through ZooKeeper to shield clients from the intricacies of cluster management, such as the addition or removal of nodes.

SQL parser

Drill uses Calcite, the open source SQL parser framework, to parse incoming queries.

The output of the parser component is a language agnostic, computer-friendly logical plan that represents the query.



Storage plugin interface

Drill serves as a query layer on top of several data sources. Storage plugins in Drill represent the abstractions that Drill uses to interact with the data sources. Storage plugins provide Drill with the following information:

- 1) Metadata available in the source
- 2) Interfaces for Drill to read from and write to data sources
- 3) Location of data and a set of optimization rules to help with efficient and fast execution of Drill queries on a specific data source

Performance in Drill

Drill is designed from the ground up for high performance on large datasets. The following core elements of Drill processing are responsible for Drill's performance:

Distributed engine

Drill provides a powerful distributed execution engine for processing queries. Users can submit requests to any node in the cluster. You can add new nodes to the cluster to scale for larger volumes of data to support more users or improve performance.

Columnar execution

Drill optimizes for both columnar storage and execution by using an in-memory data model that is hierarchical and columnar.

When working with data stored in columnar formats such as Parquet, Drill avoids disk access for columns that are not involved in a query.

Columnar execution

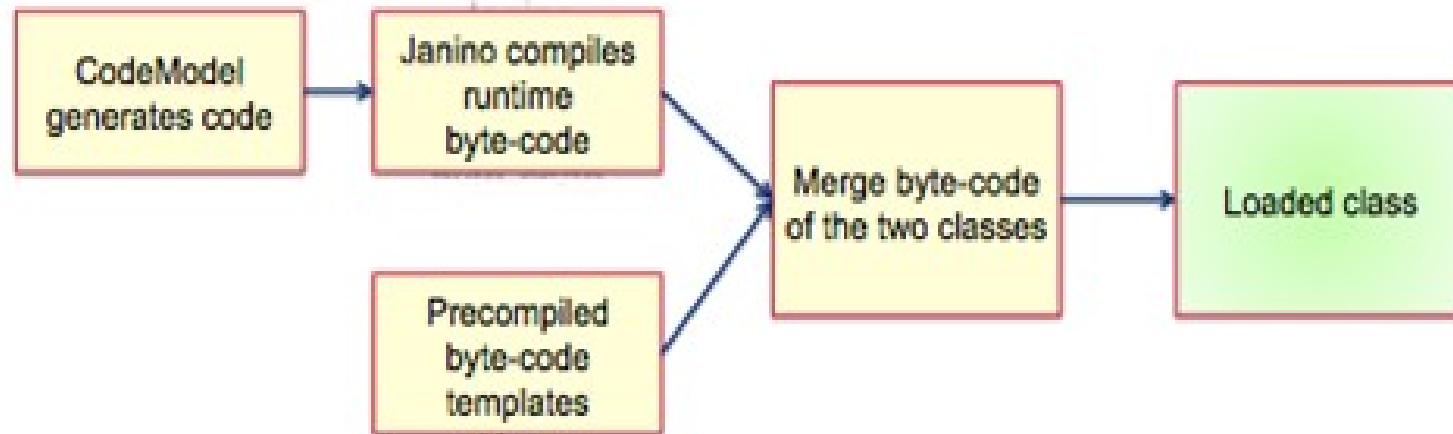
Drill's execution layer also performs SQL processing directly on columnar data without row materialization. The combination of optimizations for columnar storage and direct columnar execution significantly lowers memory footprints and provides faster execution of BI and analytic types of workloads.

Vectorization

Rather than operating on single values from a single table record at one time, vectorization in Drill allows the CPU to operate on vectors, referred to as a record batches. A record batch has arrays of values from many different records. The technical basis for efficiency of vectorized processing is modern chip technology with deep-pipelined CPU designs. Keeping all pipelines full to achieve efficiency near peak performance is impossible to achieve in traditional database engines, primarily due to code complexity.

Runtime compilation

Runtime compilation enables faster execution than interpreted execution. Drill generates highly efficient custom code for every single query. The following image shows the Drill compilation/code generation process: **Drill compiler**



REST API Introduction

The Drill REST API provides programmatic access to Drill through the Web UI.

- Using HTTP requests, you can run queries, perform storage plugin tasks, such as creating a storage plugin, obtain profiles of queries, and get current memory metrics.

AN HTTP request uses the familiar Web UI URI:

`http://<IP address or host name>:8047`