Drill Introduction

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

Key features of Apache Drill are:

- 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)
- BI/SQL tool integration using standard JDBC/ODBC drivers

Architecture

Apache Drill is a low latency distributed query engine for large-scale datasets, including structured and semi-structured/nested data. Inspired by Google's Dremel, Drill is designed to scale to several thousands of nodes and query petabytes of data at interactive speeds that BI/Analytics environments require.

High-Level Architecture

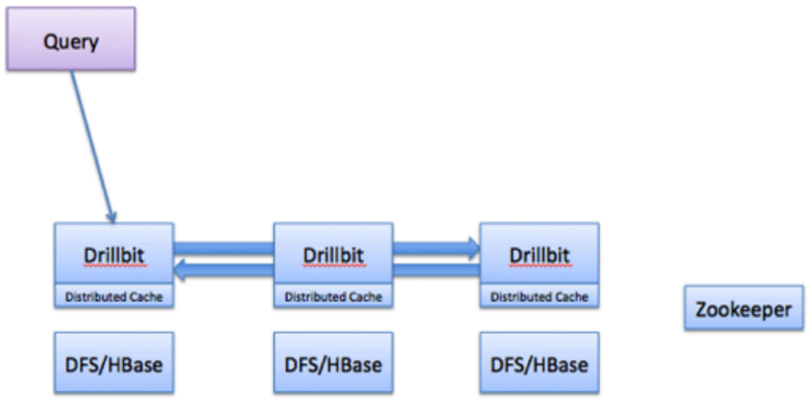
Drill includes a distributed execution environment, purpose built for large-scale data processing. At the core of Apache Drill is the 'Drillbit' service, which is responsible for accepting requests from the client, processing the queries, and returning results to the client.

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.

Note that 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.

Query Flow in Drill

The following image represents the flow of a Drill query:

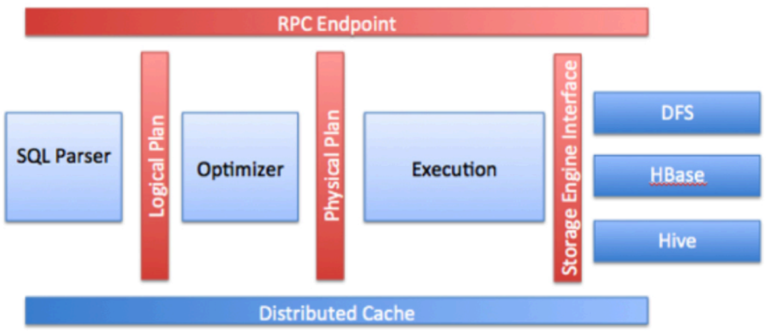


The flow of a Drill query typically involves the following steps:

1. The Drill client issues a query. A Drill client is a JDBC, ODBC, command line interface or a REST API. Any Drillbit in the cluster can accept queries from the clients. There is no master-slave concept.
2. The Drillbit then parses the query, optimizes it, and generates a distributed query plan that is optimized for fast and efficient execution.
3. The Drillbit that accepts the query becomes the driving Drillbit node for the request. It gets a list of available Drillbit nodes in the cluster from ZooKeeper. The driving Drillbit determines the appropriate nodes to execute various query plan fragments to maximize data locality.
4. The Drillbit schedules the execution of query fragments on individual nodes according to the execution plan.
5. The individual nodes finish their execution and return data to the driving Drillbit.
6. The driving Drillbit streams results back to the client.

Core Modules within a Drillbit

The following image represents Drillbit components:



• RPC end point: Drill exposes a low overhead protobuf-based RPC protocol to communicate with the clients. Additionally, a C++ and Java API layers are also available for the client applications to interact with Drill. Clients can communicate to 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 framework, to parse incoming queries. The output of the parser component is a language agnostic, computer-friendly logical plan that represents the query.

• Optimizer: Drill uses various standard database optimizations such as rule based/cost based, as well as data locality and other optimization rules exposed by the storage engine to re-write and split the query. The output of the optimizer is a distributed physical query plan that represents the most efficient and fastest way to execute the query across different nodes in the cluster.

• Execution engine: Drill provides a MPP execution engine built to perform distributed query processing across the various nodes in the cluster.

• Storage plugin interfaces: 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:

• Metadata available in the source
• Interfaces for Drill to read from and write to data sources
• Location of data and a set of optimization rules to help with efficient and faster execution of Drill queries on a specific data source

In the context of Hadoop, Drill provides storage plugins for files and HBase/M7. Drill also integrates with Hive as a storage plugin since Hive provides a metadata abstraction layer on top of files, HBase/M7, and provides libraries to read data and operate on these sources (SerDes and UDFs).

When users query files and HBase/M7 with Drill, they can do it directly or go through Hive if they have metadata defined there. Drill integration with Hive is only for metadata. Drill does not invoke the Hive execution engine for any requests.

• Distributed cache: Drill uses a distributed cache to manage metadata (not the data) and configuration information across various nodes. Sample metadata information that is stored in the cache includes query plan fragments, intermediate state of the query execution, and statistics. Drill uses Infinispan as its cache technology.

Architectural Highlights

The goal for Drill is to bring the SQL ecosystem and performance of the relational systems to Hadoop scale data WITHOUT compromising on the flexibility of Hadoop/NoSQL systems. There are several core architectural elements in Apache Drill that make it a highly flexible and efficient query engine.

Flexibility

- **Dynamic schema discovery**: Drill does not require schema or type specification for the data in order to start the query execution process. Instead, Drill starts processing the data in units called record-batches and discovers the schema on the fly during processing. Self-describing data formats such as Parquet, JSON, AVRO and NoSQL databases have schema specified as part of the data itself, which will be leveraged by Drill dynamically at the query time. Schema can change over the course of a Drill query, so all of the Drill operators are designed to reconfigure themselves when such schema changing events occur.

- **Flexible data model**: Drill is purpose-built from ground up for complex/multi-structured data commonly seen in Hadoop/NoSQL applications such as social/mobile, clickstream, logs and sensor equipped IOT. From a user point of view, Drill allows access to nested data attributes, just like SQL columns, and provides intuitive extensions to easily operate on them. From an architecture point of view, Drill provides a flexible hierarchical columnar data model that can represent complex, highly dynamic and evolving data models and allows efficient processing of it without need to flatten or materialize it either design time or at execution time. Relational data in Drill is treated as a special or simplified case of complex/multi-structured data.

- **Decentralized metadata**: Unlike other SQL-on-Hadoop technologies or any traditional relational database, Drill does not have a centralized metadata requirement. In order to query data through Drill, users do not need to create and manage tables/views in a metadata repository, or rely on a database administrator group for such a function.

  Drill metadata is derived from the storage plugins that correspond to data sources. Drill supports a varied set of storage plugins that provide a spectrum of metadata ranging from full metadata such as for Hive, partial metadata such as for HBase, or no central metadata such as for files.

  De-centralized metadata also means that Drill is NOT tied to a single Hive repository either. Users can query from multiple Hive repositories in a single query and then combine data with information from HBase tables or a file in the distributed file system.

  Users also have the ability to create metadata (tables/views/databases) within Drill using the SQL DDL syntax. De-centralized metadata is applicable during metadata creation. Drill allows persisting metadata in one of the underlying data sources.

  From a client access perspective, Drill metadata is organized just like a traditional DB (Databases->Tables/Views->Columns). The metadata is accessible through the ANSI standard INFORMATION_SCHEMA database.

- **Extensibility**: Drill provides an extensible architecture at all layers, including storage plugin layer, query layer, query optimization/execution, and client APIs. You can customize any layer for the specific needs of an organization or you can extend the layer to a broader array of use cases.

  Drill provides a built in classpath scanning and plugin concept to add additional storage plugins, functions, and operators with minimal configuration.

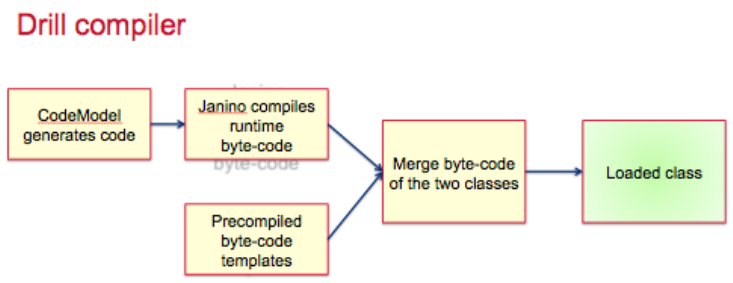
  The following list provides a few examples of Drill's extensible architectural capabilities:

  - A high performance Java API to implement custom UDFs/UDAFs
  - Ability to go beyond Hadoop by implementing custom storage plugins to other data sources such as Oracle/MySQL or NoSQL stores, such as Mongo or Cassandra
  - An API to implement custom operators
  - Support for direct execution of strongly specified JSON based logical and physical plans to help with the simplification of testing, and to enable integration of alternative query languages other than SQL.

Performance

Drill is designed from the ground up for high performance on large datasets. Few core elements of Drill processing that help Drill achieve its performance include:

- **Distributed engine**: Drill provides a powerful distributed execution engine for processing queries. Users can submit requests to any node in the cluster. You can simply add new nodes to the cluster to scale for larger volumes of data, support more users or to 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 an analytic query. Drill also provides an execution layer that 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/Analytic type 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. Record Batches are 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 something impossible to achieve in traditional database engines, primarily due to code complexity.
- **Runtime compilation**: Runtime compilation is faster compared to the interpreted execution. Drill generates highly efficient custom code for every single query for every single operator. Here is a quick overview of the Drill compilation/code generation process at a glance.


Drill compiler
CodeModel generates code
Janino compiles runtime byte-code
Precompiled byte-code templates
Merge byte-code of the two classes
Loaded class

- **Optimistic and pipelined query execution**: Drill adopts an optimistic execution model to process queries. Drill assumes that failures are infrequent within the short span of a query and therefore does not spend time creating boundaries or checkpoints to minimize recovery time. Failures at node level are handled gracefully. In the instance of a single query failure, the query is rerun. Drill execution uses a pipeline model where all tasks are scheduled at once. The query execution happens in-memory as much as possible to move data through task pipelines, persisting to disk only if there is memory overflow.

Frequently Asked Questions

Overview

Why Drill?

The 40-year monopoly of the RDBMS is over. With the exponential growth of data in recent years, and the shift towards rapid application development, new data is increasingly being stored in non-relational datastores including Hadoop, NoSQL and cloud storage. Apache Drill enables analysts, business users, data scientists and developers to explore and analyze this data without sacrificing the flexibility and agility offered by these datastores. Drill processes the data in-situ without requiring users to define schemas or transform data.

What are some of Drill's key features?

Drill is an innovative distributed SQL engine designed to enable data exploration and analytics on non-relational datastores. Users can query the data using standard SQL and BI tools without having to create and manage schemas. Some of the key features are:

- Schema-free JSON document model similar to MongoDB and Elasticsearch
- Industry-standard APIs: ANSI SQL, ODBC/JDBC, RESTful APIs
- Extremely user and developer friendly
- Pluggable architecture enables connectivity to multiple datastores

How does Drill achieve performance?

Drill is built from the ground up to achieve high throughput and low latency. The following capabilities help accomplish that:

- **Distributed query optimization and execution**: Drill is designed to scale from a single node (your laptop) to large clusters with thousands of servers.
- **Columnar execution**: Drill is the world's only columnar execution engine that supports complex data and schema-free data. It uses a shredded, in-memory, columnar data representation.
- **Runtime compilation and code generation**: Drill is the world's only query engine that compiles and re-compiles queries at runtime. This allows Drill to achieve high performance without knowing the structure of the data in advance. Drill leverages multiple compilers as well as ASM-based bytecode rewriting to optimize the code.
- **Vectorization**: Drill takes advantage of the latest SIMD instructions available in modern processors.
- **Optimistic/pipelined execution**: Drill is able to stream data in memory between operators. Drill minimizes the use of disks unless needed to complete the query.

What datastores does Drill support?

Drill is primarily focused on non-relational datastores, including Hadoop, NoSQL and cloud storage. The following datastores are currently supported:

- **Hadoop**: All Hadoop distributions (HDFS API 2.3+), including Apache Hadoop, MapR, CDH and Amazon EMR
- **NoSQL**: MongoDB, HBase
- **Cloud storage**: Amazon S3, Google Cloud Storage, Azure Blog Storage, Swift

A new datastore can be added by developing a storage plugin. Drill's unique schema-free JSON data model enables it to query non-relational datastores in-situ (many of these systems store complex or schema-free data).

What clients are supported?

- **BI tools** via the ODBC and JDBC drivers (eg, Tableau, Excel, MicroStrategy, Spotfire, QlikView, Business Objects)
- **Custom applications** via the REST API
- **Java and C applications** via the dedicated Java and C libraries

Comparisons

Drill supports a variety of non-relational datastores in addition to Hadoop. Drill takes a different approach compared to traditional SQL-on-Hadoop technologies like Hive and Impala. For example, users can directly query self-describing data (eg, JSON, Parquet) without having to create and manage schemas.

The following table provides a more detailed comparison between Drill and traditional SQL-on-Hadoop technologies:

	Drill	SQL-on-Hadoop (Hive, Impala, etc.)
Use case	Self-service, in-situ, SQL-based analytics	Data warehouse offload
Data sources	Hadoop, NoSQL, cloud storage (including multiple instances)	A single Hadoop cluster
Data model	Schema-free JSON (like MongoDB)	Relational
User experience	Point-and-query	Ingest data → define schemas → query
Deployment model	Standalone service or co-located with Hadoop or NoSQL	Co-located with Hadoop
Data management	Self-service	IT-driven
SQL	ANSI SQL	SQL-like
1.0 availability	Q2 2015	Q2 2013 or earlier

Is Spark SQL similar to Drill?

No. Spark SQL is primarily designed to enable developers to incorporate SQL statements in Spark programs. Drill does not depend on Spark, and is targeted at business users, analysts, data scientists and developers.

Does Drill replace Hive?

Hive is a batch processing framework most suitable for long-running jobs. For data exploration and BI, Drill provides a much better experience than Hive.

In addition, Drill is not limited to Hadoop. For example, it can query NoSQL databases (eg, MongoDB, HBase) and cloud storage (eg, Amazon S3, Google Cloud Storage, Azure Blob Storage, Swift).

Metadata

How does Drill support queries on self-describing data?

Drill's flexible JSON data model and on-the-fly schema discovery enable it to query self-describing data.

- **JSON data model**: Traditional query engines have a relational data model, which is limited to flat records with a fixed structure. Drill is built from the ground up to support modern complex/semi-structured data commonly seen in non-relational datastores such as Hadoop, NoSQL and cloud storage. Drill's internal in-memory data representation is hierarchical and columnar, allowing it to perform efficient SQL processing on complex data without flattening into rows.
- **On-the-fly schema discovery (or late binding)**: Traditional query engines (eg, relational databases, Hive, Impala, Spark SQL) need to know the structure of the data before query execution. Drill, on the other hand, features a fundamentally different architecture, which enables execution to begin without knowing the structure of the data. The query is automatically compiled and re-compiled during the execution phase, based on the actual data flowing through the system. As a result, Drill can handle data with evolving schema or even no schema at all (eg, JSON files, MongoDB collections, HBase tables).

But I already have schemas defined in Hive Metastore? Can I use that with Drill?

Absolutely. Drill has a storage plugin for Hive tables, so you can simply point Drill to the Hive Metastore and start performing low-latency queries on Hive tables. In fact, a single Drill cluster can query data from multiple Hive Metastores, and even perform joins across these datasets.

Is Drill "anti-schema" or "anti-DBA"?

Not at all. Drill actually takes advantage of schemas when available. For example, Drill leverages the schema information in Hive when querying Hive tables. However, when querying schema-free datastores like MongoDB, or raw files on S3 or Hadoop, schemas are not available, and Drill is still able to query that data.

challenges. For example:

- Complex data (eg, JSON) is hard to map to relational tables
- Centralized schemas are hard to keep in sync when the data structure is changing rapidly
- Non-repetitive/ad-hoc queries and data exploration needs may not justify modeling costs

Drill is all about flexibility. The flexible schema management capabilities in Drill allow users to explore raw data and then create models/structure with `CREATE TABLE` or `CREATE VIEW` statements, or with Hive Metastore.

What does a Drill query look like?

Drill uses a decentralized metadata model and relies on its storage plugins to provide metadata. There is a storage plugin associated with each data source that is supported by Drill.

The name of the table in a query tells Drill where to get the data:

```
SELECT * FROM dfs1.root.`/my/log/files/`;
SELECT * FROM dfs2.root.`/home/john/log.json`;
SELECT * FROM mongodb1.website.users;
SELECT * FROM hive1.logs.frontend;
SELECT * FROM hbase1.events.clicks;
```

What SQL functionality does Drill support?

Drill supports standard SQL (aka ANSI SQL). In addition, it features several extensions that help with complex data, such as the `KVGEN` and `FLATTEN` functions. For more details, refer to the SQL Reference.

Do I need to load data into Drill to start querying it?

No. Drill can query data 'in-situ'.

Getting Started

What is the best way to get started with Drill?

The best way to get started is to try it out. It only takes a few minutes and all you need is a laptop (Mac, Windows or Linux). We've compiled several tutorials to help you get started.

How can I ask questions and provide feedback?

Please post your questions and feedback to user@drill.apache.org. We are happy to help!

How can I contribute to Drill?

The documentation has information on how to contribute.