



وبکست یلدایی نیکآموز

دوشنبه | ۱۴۰۰ آذرماه | ۱۷:۰۰ ساعت

وبکست آشنایی عملی با
دیتابیس تحلیلی آپاچی دروید

استاد مجتبی بنائی

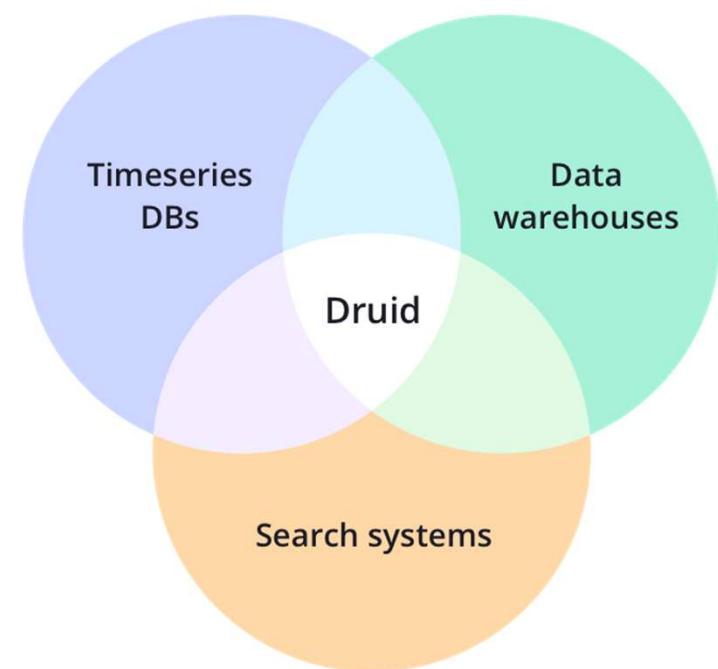
مجتبی بنائی



- مدیر واحد دیتای شرکت خدمات مدیریت صباتامین
- مدرس دوره‌های مهندسی داده موسسه نیکآموز
- موسس و سردبیر وب سایت مهندسی داده (BigData.ir)
- دانشجوی دکترای نرم افزار دانشگاه تهران (گروه تحقیقاتی پایگاه داده دانشگاه)
- مشاوره و تدریس اصول طراحی سامانه‌های مقیاس پذیر
- ...

دروید - تعریف رسمی سایت آپاچی

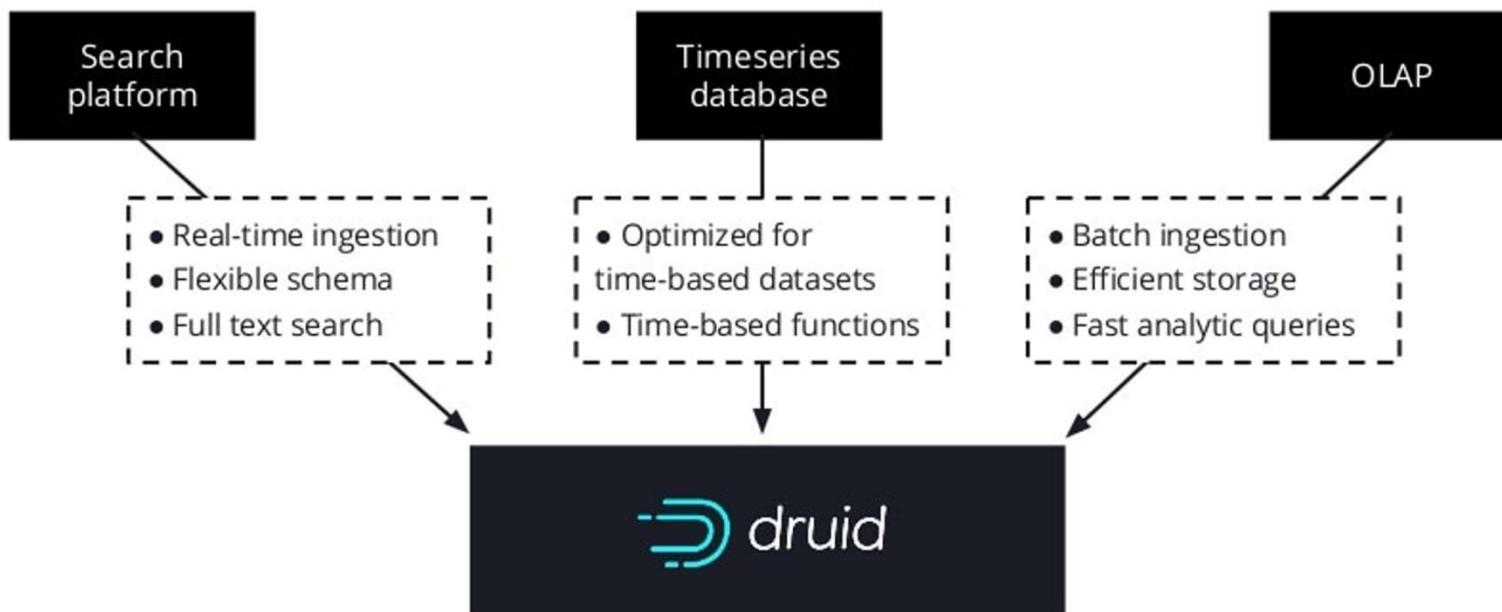
- Apache Druid is an open source distributed data store.
- Druid's core design combines ideas from [data warehouses](#), [timeseries databases](#), and [search systems](#) to create a high performance real-time analytics database for a broad range of [use cases](#).
- Druid merges key characteristics of each of the 3 systems into [its ingestion layer](#), [storage format](#), [querying layer](#), and [core architecture](#).



آپاچی دروید یک دیتابیس تحلیلی توزیع شده برای ذخیره و پردازش **بلادرنگ** حجم عظیم جریان‌های ورودی داده است.

دروید - در یک نگاه

Druid combines ideas to power a new type of analytics application.



قابلیت‌های کلیدی



Column-oriented storage

Druid stores and compresses each column individually, and only needs to read the ones needed for a particular query, which supports fast scans, rankings, and groupBys.



Native search indexes

Druid creates inverted indexes for string values for fast search and filter.



Streaming and batch ingest

Out-of-the-box connectors for Apache Kafka, HDFS, AWS S3, stream processors, and more.



Flexible schemas

Druid gracefully handles evolving schemas and [nested data](#).

قابلیت‌های کلیدی



Time-optimized partitioning

Druid intelligently partitions data based on time and time-based queries are significantly faster than traditional databases.



SQL support

In addition to its native [JSON based language](#), Druid speaks [SQL](#) over either HTTP or JDBC.



Horizontal scalability

Druid has been [used in production](#) to ingest millions of events/sec, retain years of data, and provide sub-second queries.



Easy operation

Scale up or down by just adding or removing servers, and Druid automatically rebalances. Fault-tolerant architecture routes around server failures.

قابلیت‌های کلیدی

Druid Key Features

Automatic Summarization at Ingestion

Approximate Algorithms

Time-based Partition

Indexing for Quick Filtering

Cloud-Native

Self-Healing, Self-Balancing

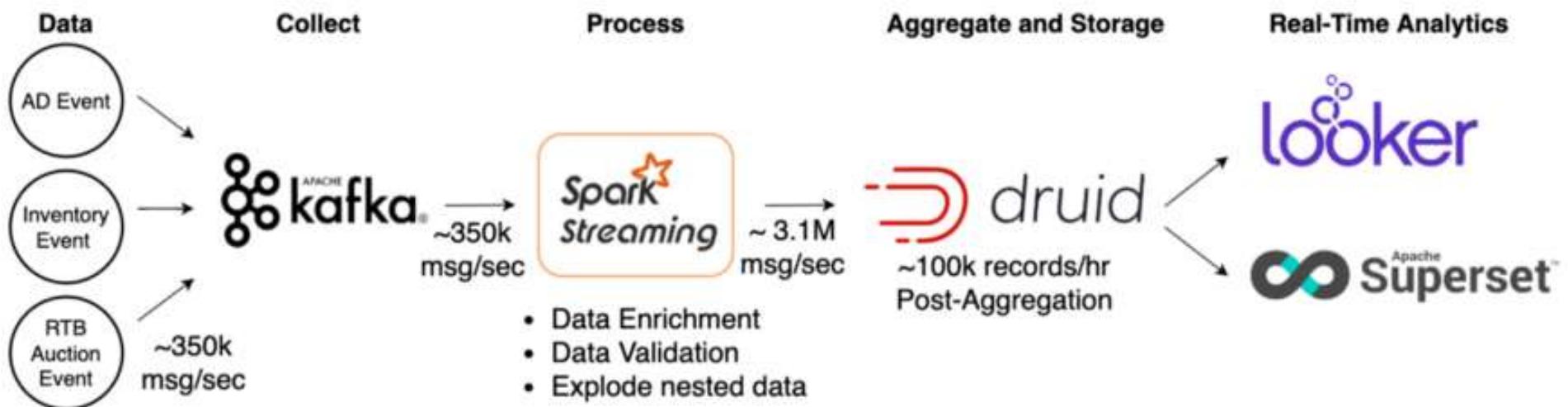
Reatime / Batch Ingestion

Column Based Storage

Scalable Distributed System

Massively Parallel Systems

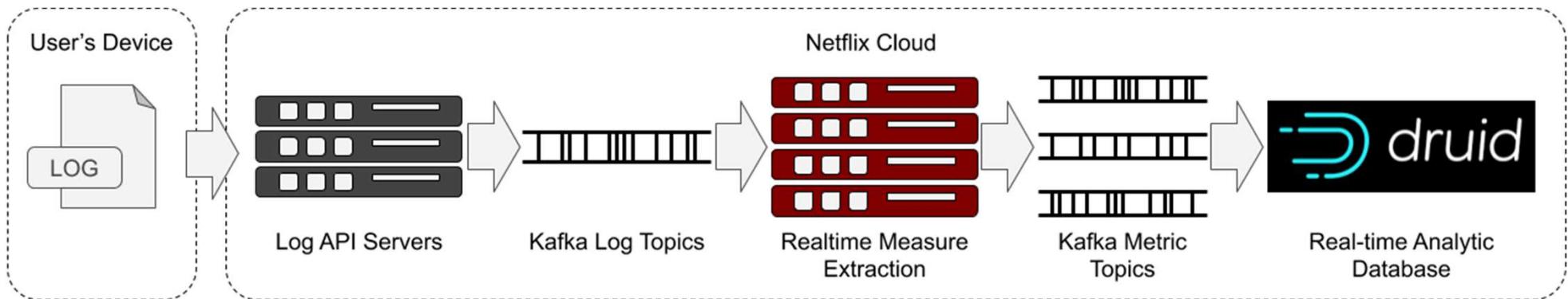
دروید در عمل - GumGum



Optimized Real-time Analytics using Spark
Streaming and Apache Druid Jun 2, 2020

gumgum

دروید در عمل - Netflix



How Netflix uses Druid for Real-time Insights to
Ensure a High-Quality Experience - Mar 3, 2020

NETFLIX

تولد دروید - 2011

metamarkets.com/2011/druid-part-i-real-time-analytics-at-a-billion-rows-per-second/



PRODUCTS SOLUTIONS RESOURCES COMPANY

LOG IN

Technology Corporate Druid

Introducing Druid: Real-Time Analytics at a Billion Rows Per Second

APRIL 30TH, 2011 ERIC TSCHETTER



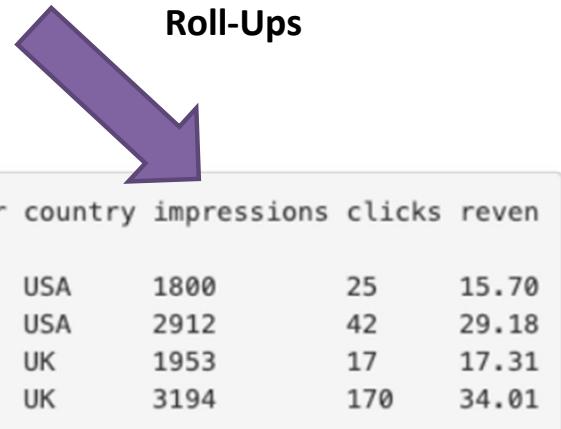
Initially, Apache Druid became an open-source software in 2012 under the GPL license, thereafter in 2015 changed to Apache 2 license and in 2018 joined the Apache Software Foundation as an incubating project.

تولد دروید - 2011

Here at Metamarkets we have developed a web-based analytics console that supports drill-downs and roll-ups of high dimensional data sets – comprising billions of events – in real-time. This is the first of two blog posts introducing Druid, the data store that powers our console. Over the last twelve months, we tried and failed to achieve scale and speed with relational databases (Greenplum, InfoBright, MySQL) and NoSQL offerings (HBase). So instead we did something crazy: we rolled our own database. Druid is the distributed, in-memory OLAP data store that resulted.

The Challenge: Fast Roll-Ups Over Big Data

To frame our discussion, let's begin with an illustration of what our raw impression event logs look like, containing many dimensions and two metrics (click and price).



Raw Event Log Data:

timestamp	publisher	advertiser	gender	country	dimensions	click	price
2011-01-01T01:01:35Z	bieberfever.com	google.com	Male	USA		0	0.65
2011-01-01T01:03:63Z	bieberfever.com	google.com	Male	USA		0	0.62
2011-01-01T01:04:51Z							
...							
2011-01-01T01:00:00Z							
2011-01-01T02:00:00Z							
2011-01-01T02:00:00Z							
...							

Roll-Ups:

timestamp	publisher	advertiser	gender	country	impressions	clicks	revenue
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Male	USA	1800	25	15.70
2011-01-01T01:00:00Z	bieberfever.com	google.com	Male	USA	2912	42	29.18
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Male	UK	1953	17	17.31
2011-01-01T02:00:00Z	bieberfever.com	google.com	Male	UK	3194	170	34.01

ساختار داده‌ها در دروید

Timestamp		Dimensions		Metrics	
Timestamp		Page	Username	Characters Added	Characters Removed
2011-01-01T01:00:00Z		Justin Bieber	Boxer	1800	25
2011-01-01T01:00:00Z		Justin Bieber	Reach	2912	42
2011-01-01T02:00:00Z		Ke\$ha	Helz	1953	17
2011-01-01T02:00:00Z		Ke\$ha	Xeno	3194	170
:		:	:	:	:

- Every dataset is stored in **datasources** (similar to tables in relational DB)
- **Timestamp** serves as primary partition column
- **Dimensions** are used to filter, query/groupBy
- **Metrics** are values that can be aggregated and must be numeric

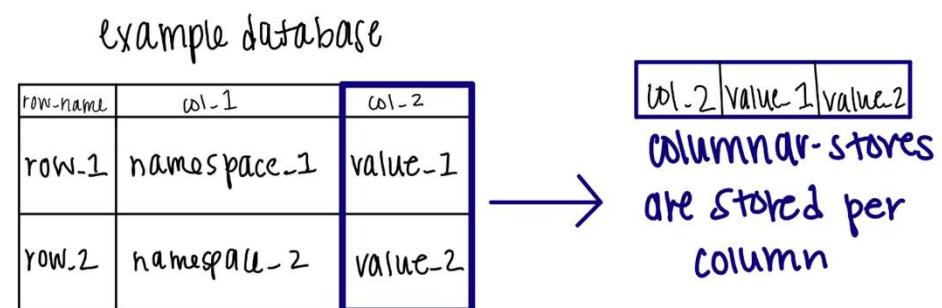
چرا دروید اینقدر سریع است؟

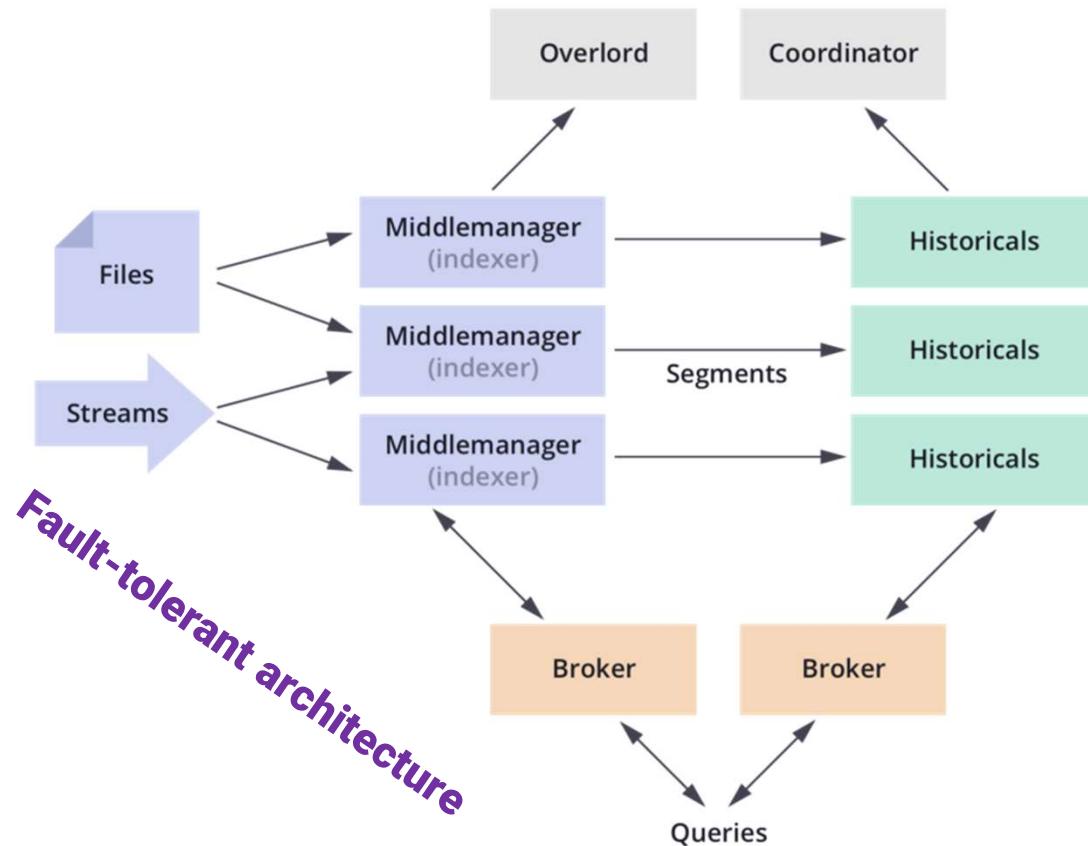
For storage efficiency and fast filtering over dimensions of data, Druid **stores data in a column-oriented compressed format**. Therefore, it can deal with data redundancy and at the same time it uses an efficient format for performing queries over multi-dimensional aggregations and groupings.

- To answer a query, it **only loads the exact columns that are required**.
- Each column **is stored optimized for its particular data type**.
- To provide **fast filtering and searching across multiple columns**, Druid uses state-of-the-art compressed bitmap indexes (for details see [CONCISE](#) and [Roaring](#)).

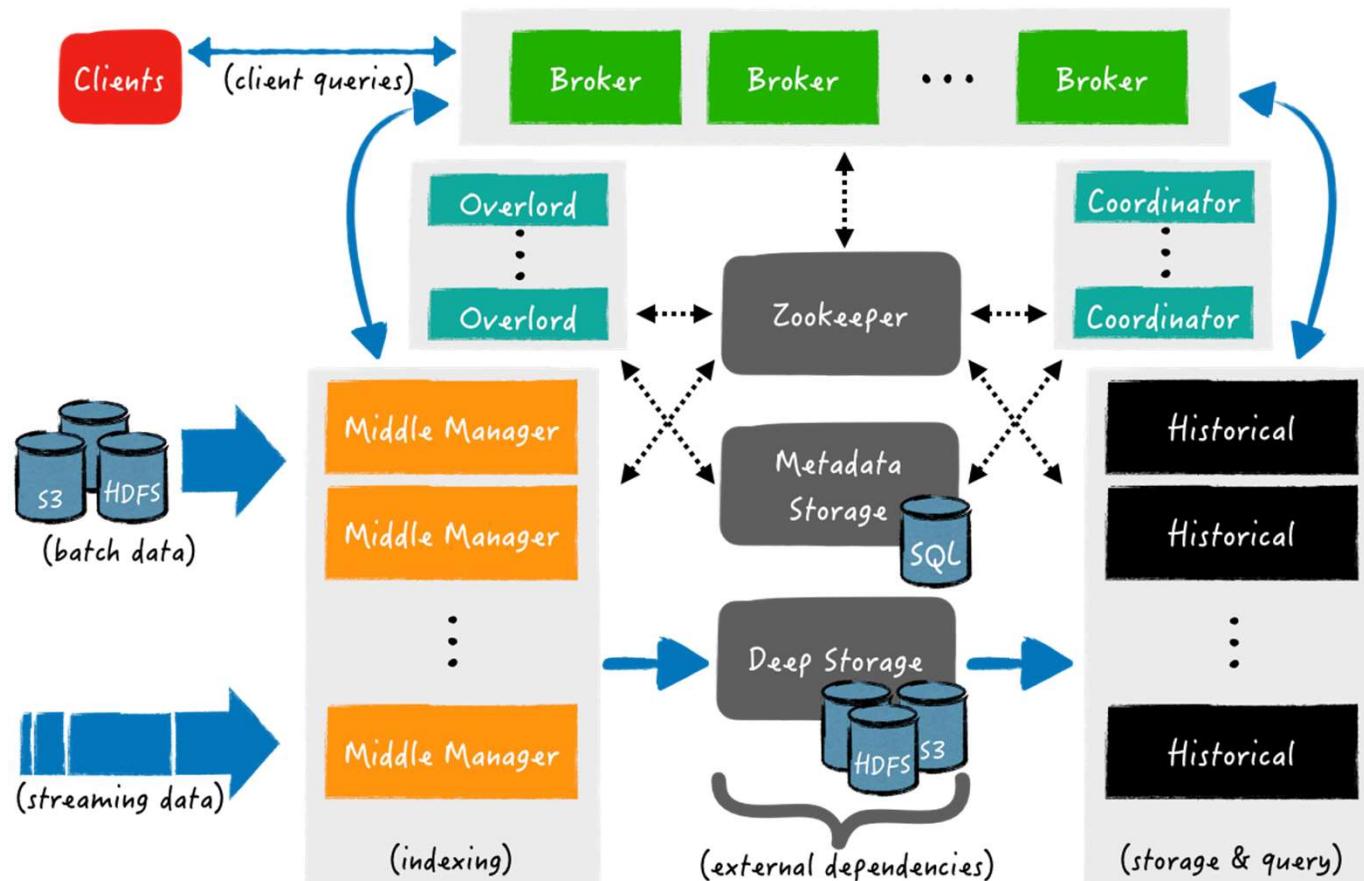
The data are **being partitioned based on time**, thus timeseries queries are significantly faster than traditional databases.

Druid provides out-of-the-box **algorithms for approximate count-distinct, approximate ranking, and computation of approximate histograms and quantiles**.





معماری دروید



**معماری دروید
نگاهی دقیق‌تر**

معماری دروید - وابستگی‌های خارجی

- **Deep Storage:** it can be **any distributed file system or object storage**, like Amazon S3, Azure Blob Storage, Apache HDFS (or any HDFS compatible system), or a network mounted file system. The purpose of the deep storage is to persist all data ingested by Druid, as a backup solution and at the same time to be available to all Druid components and clusters, when it is needed.
- **Metadata Storage:** **is backed by a traditional relational database system**, e.g., PostgreSQL or MySQL. All metadata are available to any Druid component. There are various types of metadata in Druid, some are related to the persisted segments in deep storage, for example paths of segment files, their timestamp, their version, etc., other may relate to external systems like the ingested partition offsets from a Apache Kafka topic and the rest are related to metadata of various internal processes (e.g., in which segments are being created now).
- **ZooKeeper:** is used for **internal service discovery, coordination, and leader election**.

معماری دروید - مولفه‌های اصلی

- **Coordinator:** Manages data availability on the clusters
- **Overlord:** For controlling the assignment of data ingestion workloads
- **Broker:** This is for handling queries from external clients
- **Router:** Used to route requests to brokers, coordinators and overlords
- **Historical:** For storing the data that can be queried
- **Middle manager:** This is responsible for ingesting data There are three types of servers.

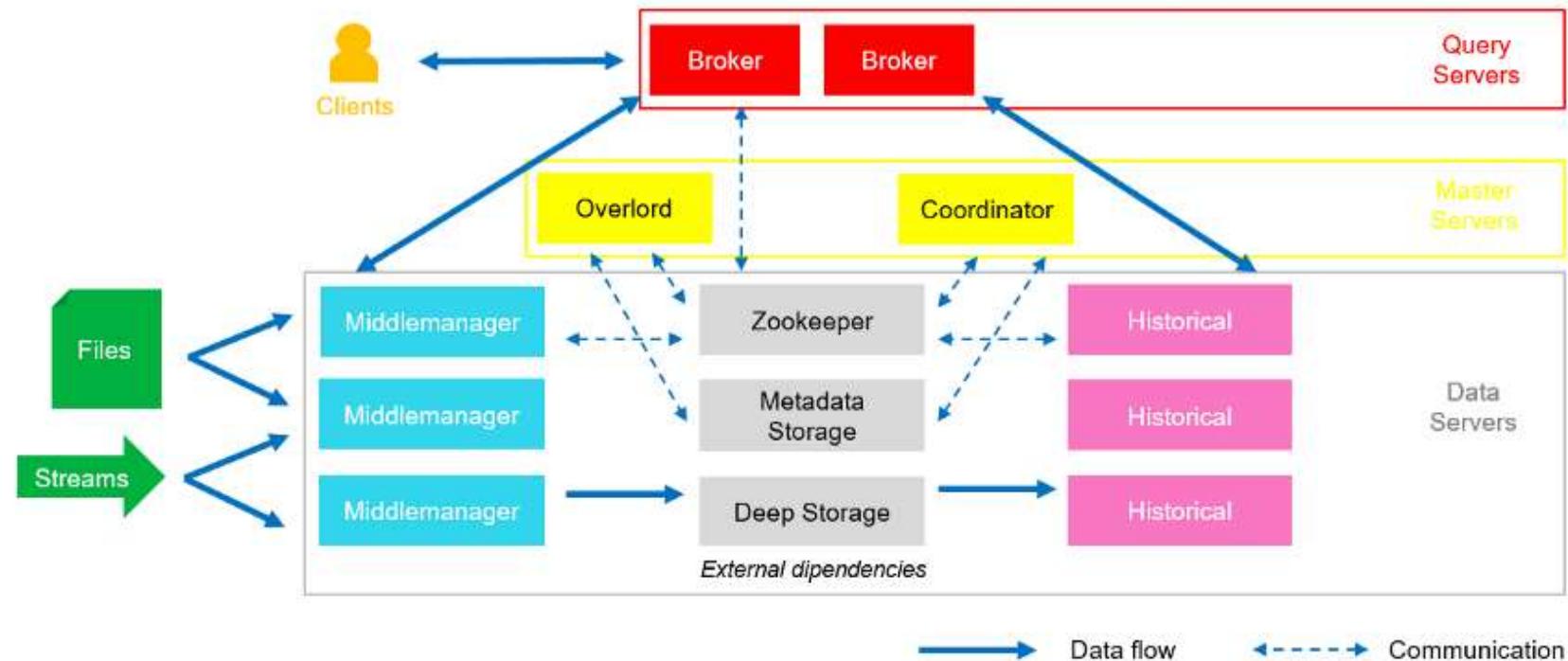
Master: This is for running the **coordinator** and **overlord** processes. It also manages data availability and ingestion.

Query: This runs **broker** and **router** processes (optional). It is used for handling queries from external clients.

Data: This runs the **historical** and **middle manager** processes. It is used to execute ingestion workloads.



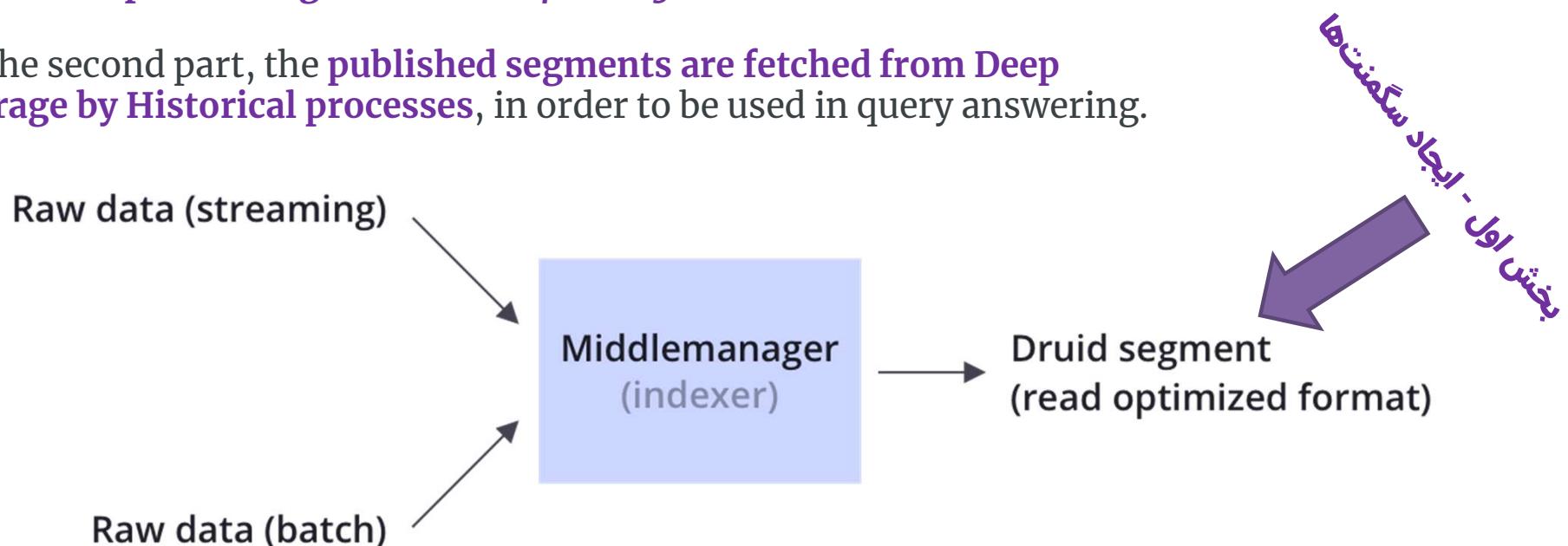
معماری دروید - مولفه‌های اصلی



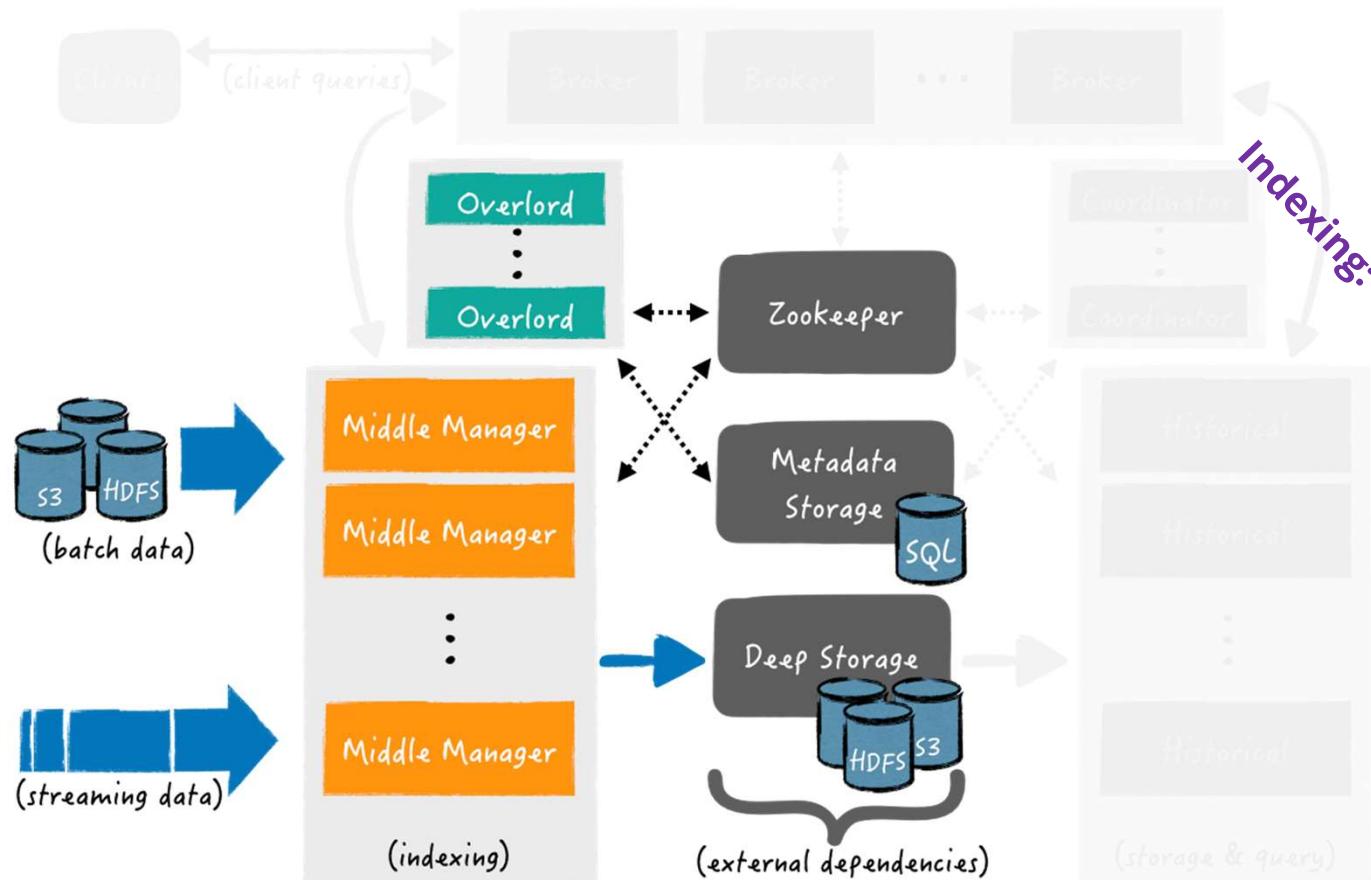
Data ingestion flow

Data ingestion flow is **a two part process**.

- In the first part, *Middle Managers* are running indexing tasks that **create and publish segments to Deep storage**.
- In the second part, the **published segments are fetched from Deep storage by Historical processes**, in order to be used in query answering.



Data ingestion flow - part 1



Indexing: Segments Creation and Publishing

Data ingestion flow - part 1

At the beginning of an indexing task, a new segment is created.

The input data of a segment may originate by a real-time stream (e.g., a Kafka topic) or by a batch of files (e.g., CSV files in HDFS)

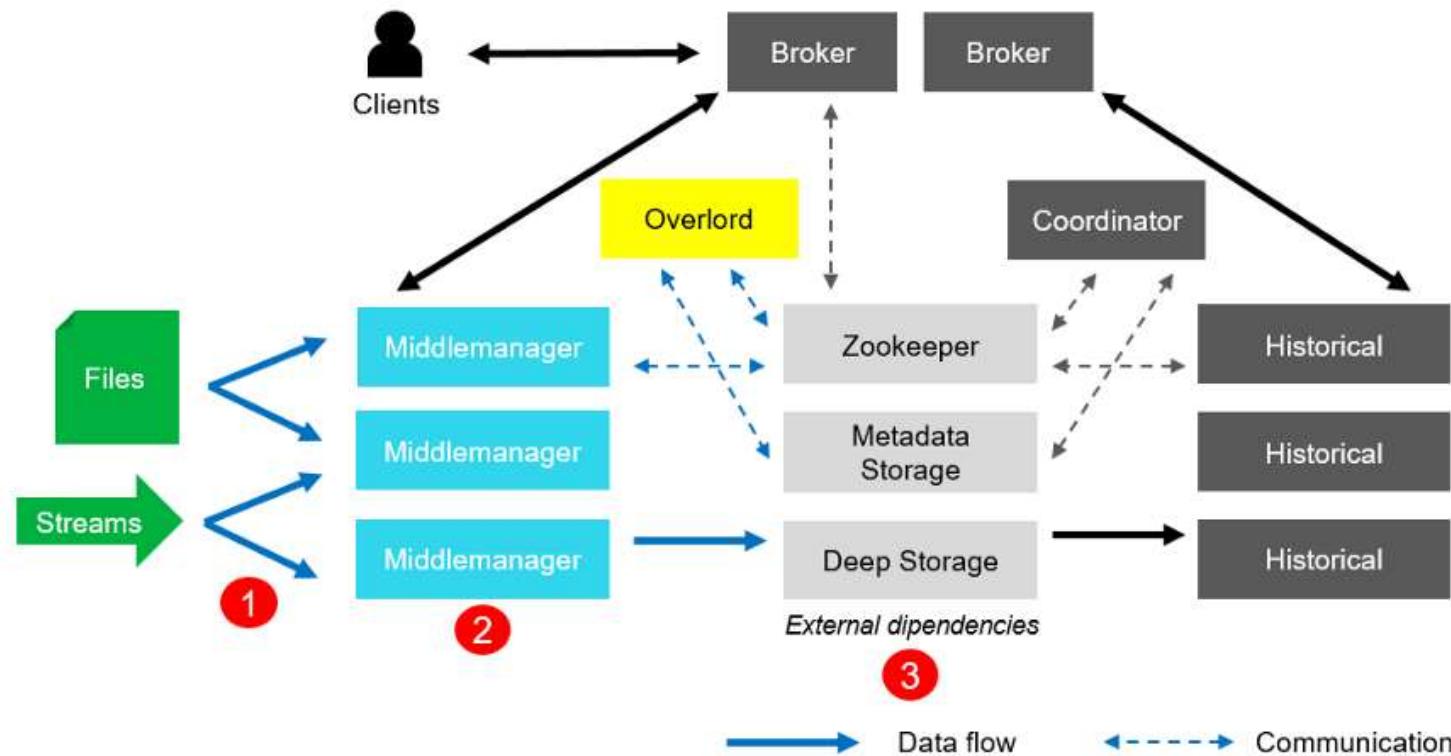
the currently creating segments from tasks that are real-time are immediately queryable

A task is completed either when it has indexed the maximum number of records per segment (e.g., five million records) or reached the desired rollup interval of time (e.g., hourly aggregates).

At this point, the segment is published by persisting its data to the Deep storage and its metadata to the Metadata store

Indexing: Segments Creation and Publishing

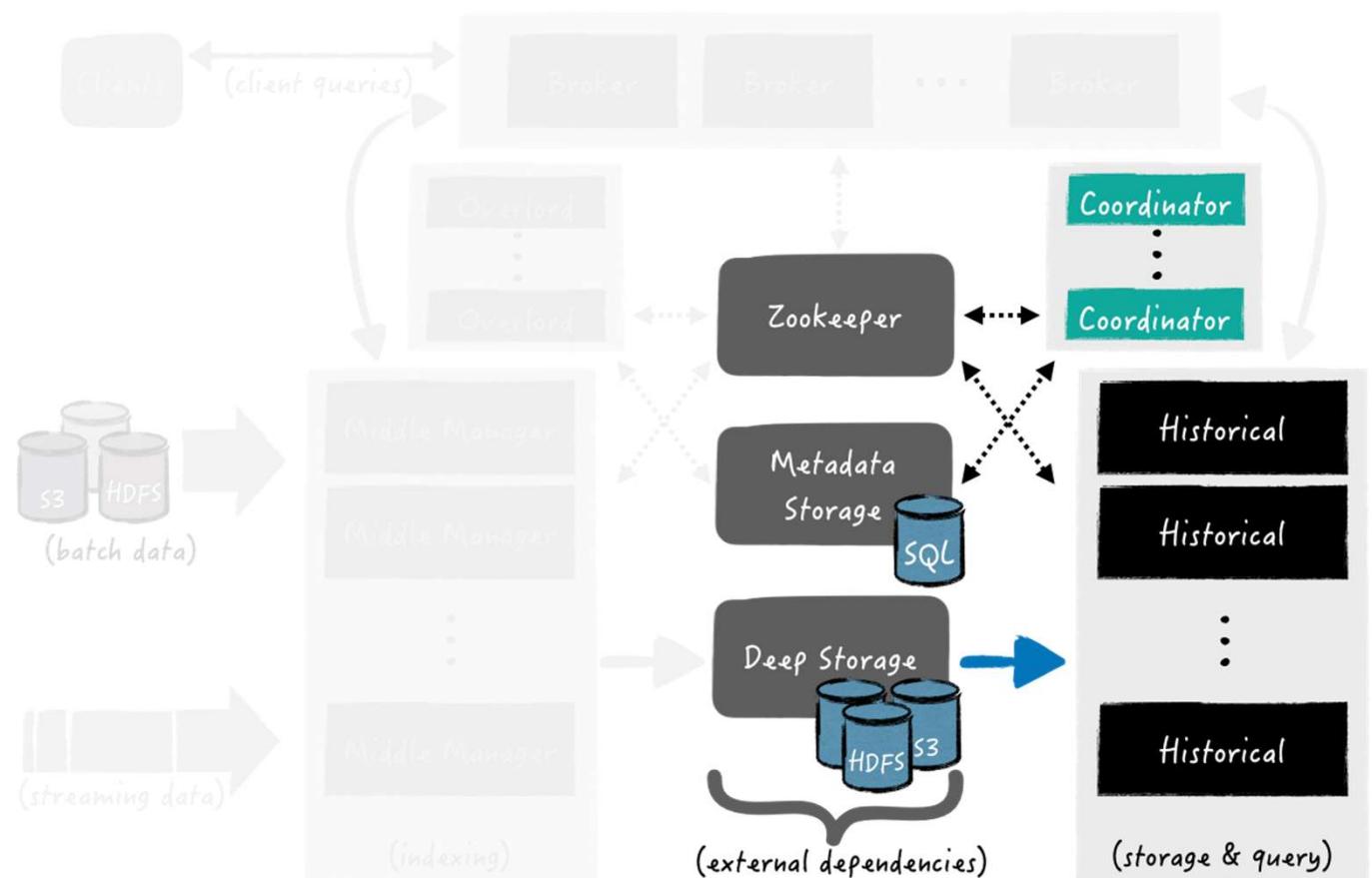
Data ingestion flow - part 1



In the indexing phase, at first data are gathered from the sources (1), then they are indexed and pre-aggregated (2) in order to create new segments and published (3) into the Deep Storage.

Data ingestion flow - part 2

Handoff: Published Segments Fetching for Query Answering



Data ingestion flow - part 2

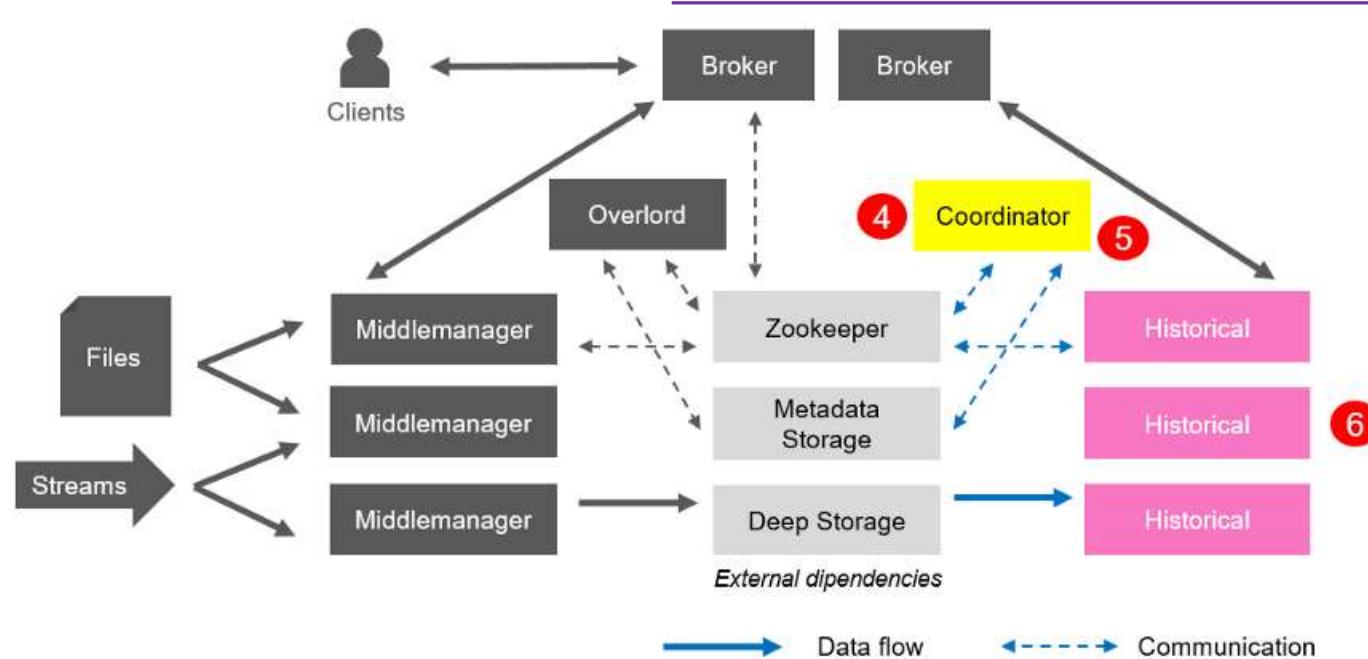
The **Coordinator** processes are periodically polling the Metadata store in order to find any newly published segments from data ingestion tasks.

Once a newly created segment is being discovered by a Coordinator process, it chooses which Historical process should fetch the segment from Deep storage

When the segment has been successfully loaded, the *Historical process* is ready to serve it for querying

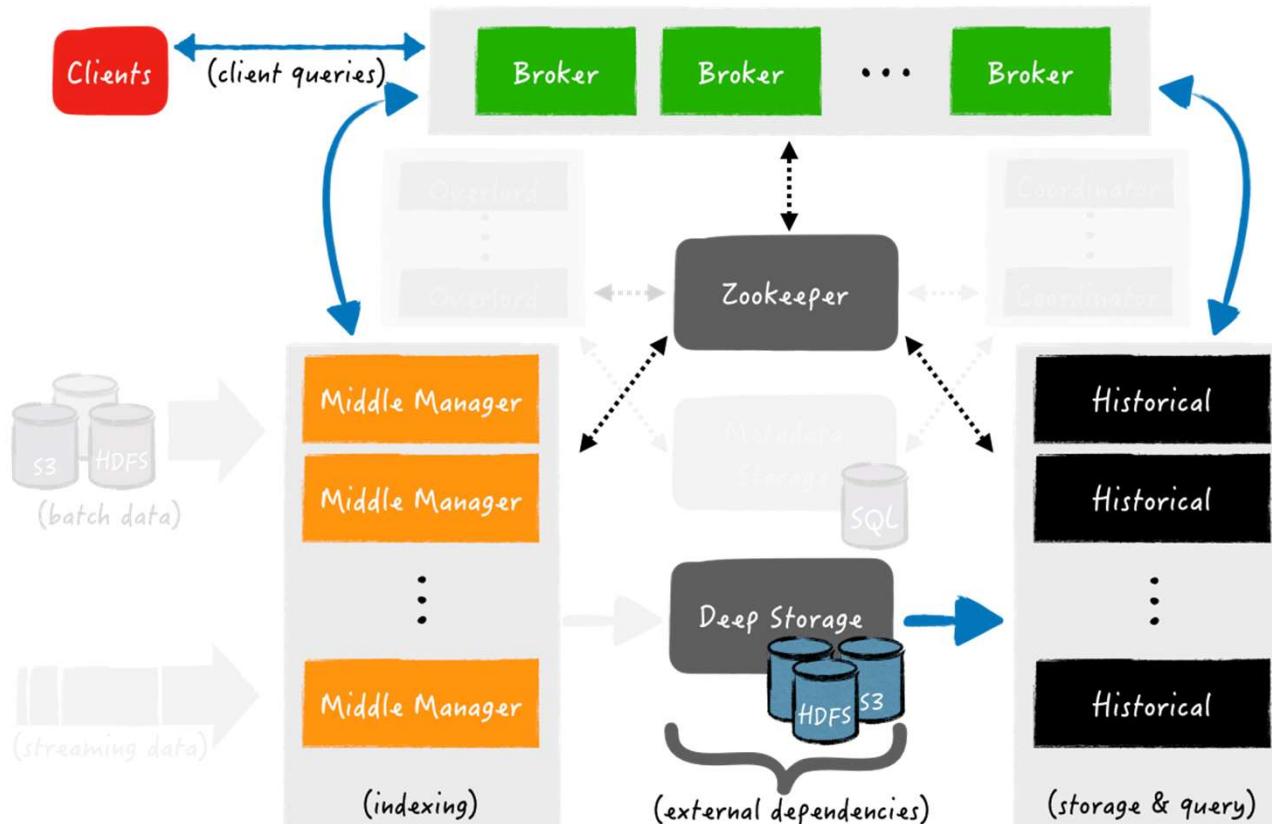
Handoff: Published Segments
Fetching for Query Answering

Data ingestion flow - part 2

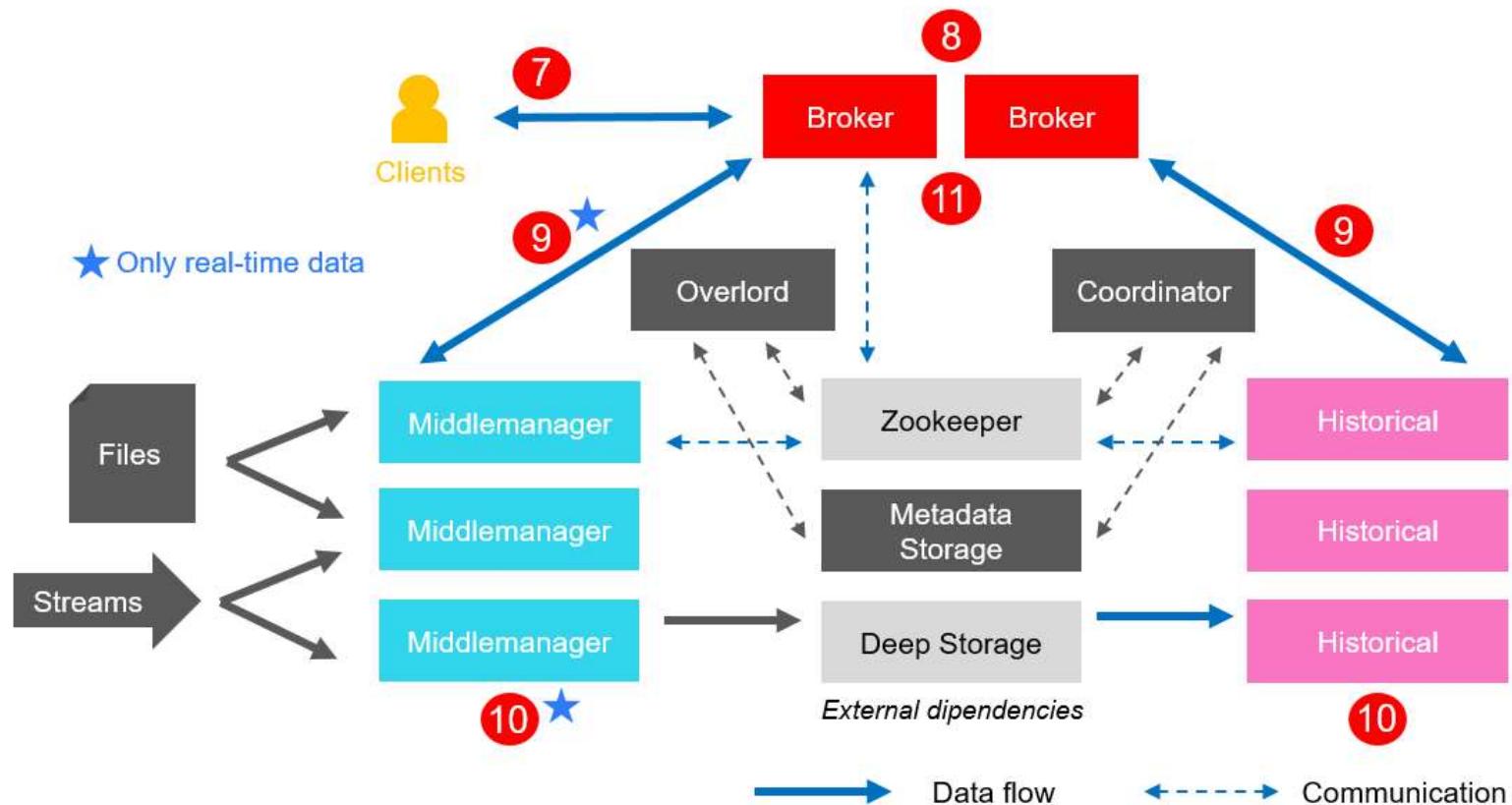


In the fetching phase, since Coordinators are listening to Metadata Storage periodically for newly published segments (4), when one of them finds a segment that is published and used but unavailable, it chooses a Historical process to load that segment in memory and instructs that Historical to do so (5). At the end, the Historical performs its task and begins serving the data(6).

Query Answering



Query Answering

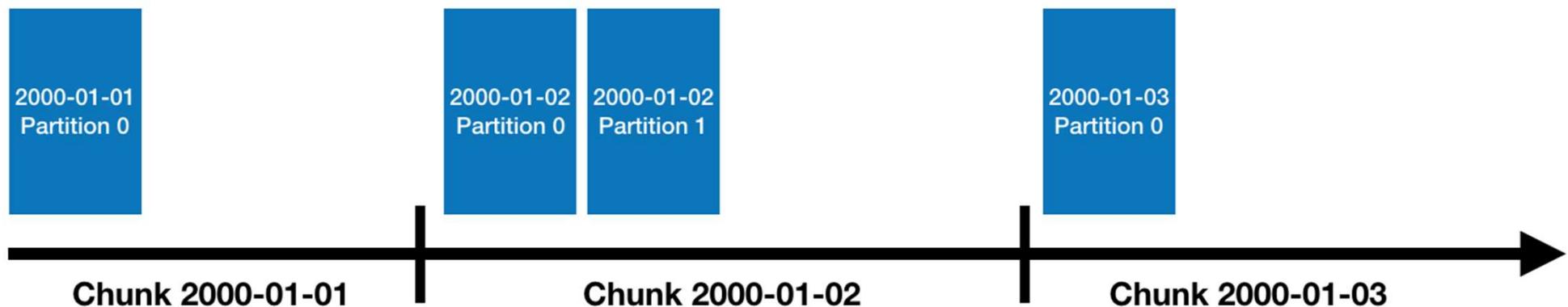


<https://medium.com/data-reply-it-datatech/what-is-apache-druid-ad67bb8cd7f7>

نگاهی دقیق‌تر به ساختار داخلی داده‌ها

- هنگام تعریف یک دیتاسورس، نحوه **پارتیشنینگ داده‌ها** را تعیین می‌کنیم.
- پارتیشن **بر اساس زمان** انجام می‌شود. (بازه روزانه در مثال زیر)
- بسته به ریزدانگی پارتیشن، بازه‌های زمانی به **مجموعه‌ای از چانک‌ها** تقسیم می‌شوند
- هر چانک، به **مجموعه‌ای از سگمنت‌ها** تقسیم می‌شود

Segment



مشاهده آماره‌های مرتبط با سگمنت‌ها در واسط کاربری دروید

Datasource name	Availability	Segment load/drop queues	Total data size	Segment size (MB) min / avg / max	Total rows	Avg. row size (bytes)	Replicated size	Compaction	Retention	Actions
druid-user	• Fully available (...)	No segments to l...	38.14 MB	0.1 0.7 2.0	35,394	1,077	78.43 MB	Target: Default (5,...)	loadByInterval(20...	
kinesis_index_test_5f2...	• Fully available (...)	No segments to l...	4.79 KB	0.0 0.0 0.0	60	79	9.59 KB	Not enabled	Cluster default: lo...	
wikiticker_transform	• Fully available (...)	No segments to l...	47.96 GB	0.0 1.8 19.1	210,596,500	227	96.42 GB	Not enabled	Cluster default: lo...	
broadcast_table_test1	• Fully available (...)	No segments to l...	13.86 MB	13.2 13.2 13.2	100,000	138	83.14 MB	Not enabled	broadcastForever(...)	
parking-citations	• Fully available (...)	No segments to l...	636.58 MB	0.0 0.3 0.6	9,372,178	67	1.29 GB	Not enabled	Cluster default: lo...	
ipv4_cardinality100K_r...	• Fully available (...)	No segments to l...	18.28 MB	17.4 17.4 17.4	1,000,000	18	36.57 MB	Not enabled	Cluster default: lo...	
wiki-par1	• Fully available (...)	No segments to l...	24.97 KB	0.0 0.0 0.0	5	4,994	49.95 KB	Not enabled	Cluster default: lo...	
twitter_v3	• Fully available (...)	No segments to l...	18.85 MB	1.8 9.0 16.2	99,985	188	37.70 MB	Not enabled	Cluster default: lo...	
trips	• Fully available (...)	No segments to l...	294.37 GB	0.0 598.6 1,105.7	1,464,785,771	200	596.31 GB	Not enabled	Cluster default: lo...	
ipv4_cardinality100K_r...	• Fully available (...)	No segments to l...	159.04 MB	43.0 50.6 54.3	10,000,000	15	318.07 MB	Not enabled	Cluster default: lo...	
wikipedia-test	• Fully available (...)	No segments to l...	5.45 MB	5.2 5.2 5.2	24,433	223	10.90 MB	Not enabled	Cluster default: lo...	
druid_survey_results	• Fully available (...)	No segments to l...	58.39 KB	0.0 0.0 0.0	69	846	116.78 KB	Not enabled	Cluster default: lo...	

نامگذاری سگمنت‌ها

Segments all have a four-part identifier with the following components (_ is the separator):

- **Datasource name**
- **Time interval** (for the time chunk containing the segment; this corresponds to the segmentGranularity specified at ingestion time).
- **Version number** (generally an ISO8601 timestamp corresponding to when the segment set was first started).
- **Partition number** (an integer, unique within a datasource+interval+version; may not necessarily be contiguous).

```
clarity-cloud0_2018-05-21T16:00:00.000Z_2018-05-21T17:00:00.000Z_2018-05-21T15:56:09.909Z_1
```

Segments with partition number ◦ (the first partition in a chunk) omit the partition number

```
clarity-cloud0_2018-05-21T16:00:00.000Z_2018-05-21T17:00:00.000Z_2018-05-21T15:56:09.909Z
```

داده‌های موجود در هر سگمنت

- Within a Segment:

- Timestamp Column Group.
- Dimensions Column Group.
- Metrics Column Group.
- Indexes that facilitate fast lookup and aggregation.

Timestamp	Dimensions				Metrics	
Timestamp	Page	Username	Gender	City	Characters Added	Characters Removed
2011-01-01T01:00:00Z	Justin Bieber	Boxer	Male	San Francisco	1800	25
2011-01-01T01:00:00Z	Justin Bieber	Reach	Male	Waterloo	2912	42
2011-01-01T02:00:00Z	Ke\$ha	Helz	Male	Calgary	1953	17
2011-01-01T02:00:00Z	Ke\$ha	Xeno	Male	Taiyuan	3194	170

انتخاب نوع ستون‌ها

String column

- ✓ indexed
- ✗ fast aggregation
- ✓ fast grouping

Numeric column

- ✗ indexed
- ✓ fast aggregation
- ✓ fast grouping

مروری بر نحوه ایندکسینگ و فشرده‌سازی داده‌ها

COLUMN COMPRESSION - DICTIONARIES

timestamp	publisher	advertiser	gender	country	...	click	price
2011-01-01T00:01:35Z	bieberfever.com	google.com	Male	USA		0	0.65
2011-01-01T00:03:63Z	bieberfever.com	google.com	Male	USA		0	0.62
2011-01-01T00:04:51Z	bieberfever.com	google.com	Male	USA		1	0.45
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Female	UK		0	0.87
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK		0	0.99
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK		1	1.53
...							

- Create ids
 - bieberfever.com -> 0, ultratrimfast.com -> 1
- Store
 - publisher -> [0, 0, 0, 1, 1, 1]
 - advertiser -> [0, 0, 0, 0, 0, 0]

مروری بر نحوه ایندکسینگ و فشرده سازی داده ها

BITMAP INDEXES

timestamp	publisher	advertiser	gender	country	...	click	price
2011-01-01T00:01:35Z	bieberfever.com	google.com	Male	USA		0	0.65
2011-01-01T00:03:63Z	bieberfever.com	google.com	Male	USA		0	0.62
2011-01-01T00:04:51Z	bieberfever.com	google.com	Male	USA		1	0.45
2011-01-01T01:00:00Z	ultratrimfast.com	google.com	Female	UK		0	0.87
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK		0	0.99
2011-01-01T02:00:00Z	ultratrimfast.com	google.com	Female	UK		1	1.53
...							

- bieberfever.com -> [0, 1, 2] -> [111000]
- ultratrimfast.com -> [3, 4, 5] -> [000111]
- Compress using Concise or Roaring (take advantage of dimension sparsity)

مروری بر نحوه ایندکسینگ و فشرده‌سازی داده‌ها

FAST AND FLEXIBLE QUERIES

Rows	POETS
0	JUSTIN BIEBER
1	JUSTIN BIEBER
2	KE\$HA
3	KE\$HA

JUSTIN BIEBER

[1, 1, 0, 0]

KE\$HA

[0, 0, 1, 1]

JUSTIN BIEBER

OR

KE\$HA

[1, 1, 1, 1]

Queries that solely aggregate metrics based on filters do not need to touch the list of dimension values !

لوكاپ‌ها : راهی برای جلوگیری از جوین‌های غیرضروری

Think back to lookups.

timestamp	product_id	country	city	gender	age	revenue
2030-01-01	212	US	New York	F	34	180.00
2030-01-01	998	FR	Paris	M	28	24.95

↓
Lookup: "products"

id	name
212	Office chair
998	Coffee mug, 2-pack

SELECT

LOOKUP(id, 'products') AS product_name

SUM(sales.revenue)

FROM sales

GROUP BY product_name

تجمعیع داده‌ها هنگام ورود - Rollup

timestamp	page	city	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	SF	10	5
2011-01-01T00:03:45Z	Justin Bieber	SF	25	37
2011-01-01T00:05:62Z	Justin Bieber	SF	15	19
2011-01-01T00:06:33Z	Ke\$ha	LA	30	45
2011-01-01T00:08:51Z	Ke\$ha	LA	16	8
2011-01-01T00:09:17Z	Miley Cyrus	DC	75	10
2011-01-01T00:11:25Z	Miley Cyrus	DC	11	25
2011-01-01T00:23:30Z	Miley Cyrus	DC	22	12
2011-01-01T00:49:33Z	Miley Cyrus	DC	90	41



timestamp	page	city	count	sum_added	sum_deleted
2011-01-01T00:00:00Z	Justin Bieber	SF	3	50	61
2011-01-01T00:00:00Z	Ke\$ha	LA	2	46	53
2011-01-01T00:00:00Z	Miley Cyrus	DC	4	198	88

الگوریتم‌های تقریبی - بنیاد آپاچی

C datasketches.apache.org/docs/Background/TheChallenge.html



DOCUMENTATION

DOWNLOAD

RESEARCH



APACHE

API Snapshots: Java Core, Memory, Pig, Hive,

Background ▾

The Challenge

- Sketch Origins
- Sketch Elements
- Presentations
- Overview Slide Deck

Architecture And Design ▾

Sketch Families ▾

System Integrations ▾

Community ▾

Research ▾

The Challenge: Fast, Approximate Analysis of Big Data¹

Suppose you have a new Internet company that sells Mobile Apps and Music. Your internal reporting system collects log data from two main sources: your web servers and your Financial Transactions System that records purchases and handles credit cards and authentication. The data logs from these two systems might look something like this:

Web Site Logs					Financial Transactions System Log				
Time	User ID	Site	Time Spent Sec	Items Viewed	Time	User ID	Site	Purchased	Revenue
9:00 AM	U1	Apps	59	5	9:00 AM	U1	Apps	FaceTune	\$3.99
9:30 AM	U2	Apps	179	15	9:30 AM	U2	Apps	Minecraft	\$6.99
10:00 AM	U3	Music	29	3	10:00 AM	U3	Music	Purple Rain	\$1.29
1:00 PM	U1	Music	89	10	Billions of rows ...				
Billions of rows ...									

The web server logs contain information such as a time-stamp, a user identifier (obfuscated, of course), the site visited, a time-spent metric, and a number of items viewed metric. The financial logs contain information such as a time-stamp, a user identifier, the site visited, the purchased item and revenue received for the item.

From these two simple sets of data there are many queries that we would like to make, and among those, some very natural queries might include the following:

الگوریتم‌های تقریبی

- Data sketches are lossy data structures
- Tradeoff accuracy for reduced storage and improved performance.
- Summarize data at ingestion time using sketches
- Improves roll-up, reduce memory footprint

الگوریتم‌های تقریبی

timestamp	page	userid	city	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	user11	SF	10	5
2011-01-01T00:03:45Z	Justin Bieber	user22	SF	25	37
2011-01-01T00:05:62Z	Justin Bieber	user11	SF	15	19
2011-01-01T00:06:33Z	Ke\$ha	user33	LA	30	45
2011-01-01T00:08:51Z	Ke\$ha	user33	LA	16	8
2011-01-01T00:09:17Z	Miley Cyrus	user11	DC	75	10
2011-01-01T00:11:25Z	Miley Cyrus	user44	DC	11	25
2011-01-01T00:23:30Z	Miley Cyrus	user44	DC	22	12
2011-01-01T00:49:33Z	Miley Cyrus	user55	DC	90	41



timestamp	page	city	count	sum_added	sum_deleted	userid_sketch
2011-01-01T00:00:00Z	Justin Bieber	SF	3	50	61	sketch_obj
2011-01-01T00:00:00Z	Ke\$ha	LA	2	46	53	sketch_obj
2011-01-01T00:00:00Z	Miley Cyrus	DC	4	198	88	sketch_obj

الگوریتم‌های تقریبی

Approximate queries can provide up to 99% accuracy while greatly improving performance

- Bloom Filters
 - Self Joins
- Theta Sketches
 - Union/Intersection/Difference
- HLL Sketches
 - Count Distinct
- Quantile Sketches
 - Median, percentiles

SQL - کوئری

```
SELECT channel, SUM(added)
FROM "wikipedia"
WHERE
    "__time" >= CURRENT_TIMESTAMP - INTERVAL '1' DAY
    and commentLength >= 50
GROUP BY channel
ORDER BY SUM(added) desc
LIMIT 3
```

Native Json - کوئری

```
{  
  "queryType": "topN",  
  "dataSource": {  
    "type": "table",  
    "name": "wikipedia"  
  },  
  "virtualColumns": [],  
  "dimension": {  
    "type": "default",  
    "dimension": "channel",  
    "outputName": "d0",  
    "outputType": "STRING"  
  },  
  "metric": {  
    "type": "numeric",  
    "metric": "a0"  
  },  
  "threshold": 3,  
  "intervals": {  
    "type": "intervals",  
    "intervals": [  
      "2019-04-25T14:56:20.000Z/146140482-04-24T15:36:27.903Z"  
    ]  
  },  
}
```

```
"filter": {  
  "type": "bound",  
  "dimension": "commentLength",  
  "lower": "50",  
  "upper": null,  
  "lowerStrict": false,  
  "upperStrict": false,  
  "extractionFn": null,  
  "ordering": {  
    "type": "numeric"  
  }  
,  
  "granularity": {  
    "type": "all"  
  },  
  "aggregations": [  
    {  
      "type": "doubleSum",  
      "name": "a0",  
      "fieldName": "added",  
      "expression": null  
    }  
,  
    "postAggregations": [],  
    "context": {  
      "sqlQueryId": "b0112645-e231-47b1-95a6-6154deca8a7a",  
      "timeout": 300000  
    },  
    "descending": false  
  ]  
}
```

کوئری Native Json

```
{  
  "queryType": "topN",  
  "dataSource": {  
    "type": "table",  
    "name": "wikipedia"  
  },  
  "virtualColumns": [],  
  "dimension": {  
    "type": "default",  
    "dimension": "channel",  
    "outputName": "d0",  
    "outputType": "STRING"  
  },  
  "metric": {  
    "type": "numeric",  
    "metric": "a0"  
  },  
  "threshold": 3,  
  "intervals": {  
    "type": "intervals",  
    "intervals": [  
      "2019-04-25T14:56:20.000Z/146140482-04-24T15:36:27.903Z"  
    ]  
  },  
}
```

```
"filter": {  
  "type": "bound",  
  "dimension": "commentLength",  
  "lower": "50",  
  "upper": null,  
  "lowerStrict": false,  
  "upperStrict": false,  
  "extractionFn": null,  
  "ordering": {  
    "type": "numeric"  
  }  
},  
"granularity": {  
  "type": "all"  
},  
"aggregations": [  
  {  
    "type": "doubleSum",  
    "name": "a0",  
    "fieldName": "added",  
    "expression": null  
  }  
],  
"postAggregations": [],  
"context": {  
  "sqlQueryId": "b0112645-e231-47b1-95a6-6154deca8a7a",  
  "timeout": 300000  
},  
"descending": false  
}
```

انواع کوئری در دروید

- **TimeBoundary** - Returns min/max timestamp for given interval.
- **Timeseries** - When you don't want to group by dimension
- **TopN** - When you want to group by a single dimension
 - Approximate if > 1000 dimension values
- **GroupBy** - Least performant/most flexible
- **Scan** - For returning streaming raw data
 - Perfect ordering not preserved
- **Select** - For returning paginated raw data
- **Search** - Returns dimensions that match text search

انواع فیلترها در دروید

- Interval
 - Matches time range, can be used on `_time` or any column with millisecond timestamp
- Selector
 - Matches a single dimension to a value
- Column Comparison
 - Compares two columns. Example `ColA == ColB`
- Search
 - Filter on partial string matches
- In
 - Matches to list of values

أنواع فیلتر در دروید

- Like Filter
 - Equivalent of SQL LIKE
 - Can perform better than Search Filter for prefix only searching

A note about Extraction Functions....

Most filters also support extraction functions. Performance of functions varies greatly, and if possible using functions on ingest instead of query time can improve performance

When NOT to use Apache Druid

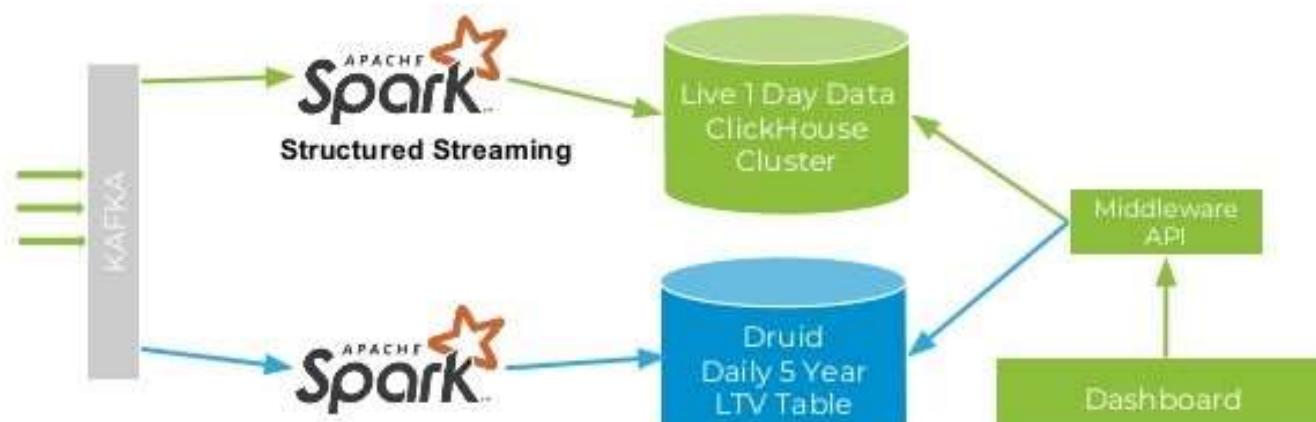
Despite Druid's ability to process large amounts of data, there are situations where you would NOT want to use it:

- **Low-latency updates of existing records** using a primary key. Druid supports streaming inserts, but not streaming updates (i.e., updates are done using background batch jobs).
- **Storing all the historical data in a raw format**, without any aggregation or transformation.
- **Building an offline reporting system** where query latency is not very important.
- **Queries that have “big” joins (joining one big fact table to another big fact table)**, where you're comfortable with these queries taking a long time to complete.

آخرین سخن



Now: ClickHouse + Druid



38 billion events daily

منابع آموزشی

druid.apache.org/docs/latest/design/index.html

Technology Use Cases Powered By Docs

druid

Getting started

- Introduction to Apache Druid
- Quickstart
- Docker
- Single server deployment
- Clustered deployment

Tutorials

Design

Ingestion

Querying

Configuration

Operations

Development

Introduction to Apache Druid

Apache Druid is a real-time analytics database designed for fast analytics ("OLAP" queries) on large data sets. Most often, Druid is used where real-time ingestion, fast query performance, and high uptime are required.

Druid is commonly used as the database backend for GUIs of dashboards or for highly-concurrent APIs that need fast aggregations. Druid is well suited for event-oriented data.

Common application areas for Druid include:

- Clickstream analytics including web and mobile analytics
- Network telemetry analytics including network performance monitoring
- Server metrics storage
- Supply chain analytics including manufacturing metrics
- Application performance metrics
- Digital marketing/advertising analytics
- Business intelligence/OLAP

imply.io/blog?p=1

Druid Nails Cost Efficiency Challenge

#benchmark #druid #clickhouse #rockset

by Eric Tschetter · in Apache Druid · November 22, 2021

To make a long story short, we were pleased to confirm that Druid can process the same amount of data with fewer hardware resources!

[Read More](#)

Unveiling Project Shapeshift Nov. 9th

#druid summit 2021 #project shapeshift

by Fangjin Yang · in Imply platform · November 3, 2021

There is a new category within data analytics emerging (not just for analysts and data scientists), but instead centered in the world of business managers.

کارگاه عملی

• دانلود مخزن کد کارگاه

https://gitlab.com/nikamooz_bigdata/webinars

- نیاز به ویپیان یا استفاده از خدمات shecan.ir خواهد داشت.