if(kakao)2021

Druid@Kakao

Druid 도입 사례 및 Multi-Tenant 클러스터 소개

황보동규 Evan.Hwangbo
카카오

Druid에 대한 기대

Use Case — 1

Use Case — 2

Use Case — 3

Use Case — 4

Multi-Tenant 클러스터 소개

Druid에 대한 기대

Use Case — 1

Use Case — 2

Use Case — 3

Use Case — 4

Multi-Tenant 클러스터 소개

Druid's core architecture combines ideas from data warehouses, timeseries databases, and logsearch systems. Some of Druid's key features are:

1. **Columnar storage format.** Druid uses column-oriented storage, meaning it only needs to load the exact columns needed for a particular query. This gives a huge speed boost to queries that only hit a few columns. In addition, each column is stored optimized for its particular data type, which supports fast scans and aggregations.
2. **Scalable distributed system.** Druid is typically deployed in clusters of tens to hundreds of servers, and can offer ingest rates of millions of records/sec, retention of trillions of records, and query latencies of sub-second to a few seconds.
3. **Massively parallel processing.** Druid can process a query in parallel across the entire cluster.
4. **Realtime or batch ingestion.** Druid can ingest data either real-time (ingested data is immediately available for querying) or in batches.
5. **Self-healing, self-balancing, easy to operate.** As an operator, to scale the cluster out or in, simply add or remove servers and the cluster will rebalance itself automatically, in the background, without any downtime. If any Druid servers fail, the system will automatically route around the damage until those servers can be replaced. Druid is designed to run 24/7 with no need for planned downtimes for any reason, including configuration changes and software updates.
6. **Cloud-native, fault-tolerant architecture that won't lose data.** Once Druid has ingested your data, a copy is stored safely in deep storage (typically cloud storage, HDFS, or a shared filesystem). Your data can be recovered from deep storage even if every single Druid server fails. For more limited failures affecting just a few Druid servers, replication ensures that queries are still possible while the system recovers.
7. **Indexes for quick filtering.** Druid uses Roaring or CONCISE compressed bitmap indexes to create indexes that power fast filtering and searching across multiple columns.
8. **Time-based partitioning.** Druid first partitions data by time, and can additionally partition based on other fields. This means time-based queries will only access the partitions that match the time range of the query. This leads to significant performance improvements for time-based data.
9. **Approximate algorithms.** Druid includes algorithms for approximate count-distinct, approximate ranking, and computation of approximate histograms and quantiles. These algorithms offer bounded memory usage and are often substantially faster than exact computations. For situations where accuracy is more important than speed, Druid also offers exact count-distinct and exact ranking.
10. **Automatic summarization at ingest time.** Druid optionally supports data summarization at ingestion time. This summarization partially pre-aggregates your data, and can lead to big costs savings and performance boosts.

https://druid.apache.org/docs/latest/design/

Druid's core architecture combines ideas from data warehouses, timeseries databases, and logsearch systems. Some of Druid's key features are:

1. **Columnar storage format.** Druid uses column-oriented storage, meaning it only needs to load the exact columns needed for a particular query. This gives a huge speed boost to queries that only hit a few columns. In addition, each column is stored optimized for its particular data type, which supports fast scans and aggregations.
2. **Scalable distributed system.** Druid is typically deployed in clusters of tens to hundreds of servers, and can offer ingest rates of millions of records/sec, retention of trillions of records, and query latencies of sub-second to a few seconds.
3. **Massively parallel processing.** Druid can process a query in parallel across the entire cluster.
4. **Realtime or batch ingestion.** Druid can ingest data either real-time (ingested data is immediately available for querying) or in batches.
5. **Self-healing, self-balancing, easy to operate.** As an operator, to scale the cluster out or in, simply add or remove servers and the cluster will rebalance itself automatically, in the background, without any downtime. If any Druid servers fail, the system will automatically route around the damage until those servers can be replaced. Druid is designed to operate 24/7 without need for planned downtimes for any reason, including configuration changes and software updates.
6. **Cloud-native, fault-tolerant architecture that won't lose data.** Once Druid has ingested your data, a copy is stored safely in deep storage (typically cloud storage, HDFS, or a shared filesystem). Your data can be recovered from deep storage even if every single Druid server fails. For more limited failures affecting just a few Druid servers, replication ensures that queries are still possible while the system recovers.
7. **Indexes for quick filtering.** Druid uses Roaring or CONCISE compressed bitmap indexes to create indexes that power fast filtering and searching across multiple columns.
8. **Time-based partitioning.** Druid first partitions data by time, and can additionally partition based on other fields. This means time-based queries will only access the partitions that match the time range of the query. This leads to significant performance improvements for time-based data.
9. **Approximate algorithms.** Druid includes algorithms for approximate count-distinct, approximate ranking, and computation of approximate histograms and quantiles. These algorithms offer bounded memory usage and are often substantially faster than exact computations. For situations where accuracy is more important than speed, Druid also offers exact count-distinct and exact ranking.
10. **Automatic summarization at ingest time.** Druid optionally supports data summarization at ingestion time. This summarization partially pre-aggregates your data, and can lead to big costs savings and performance boosts.

실시간 빠른응답 경제성

Druid에 대한 기대

Use Case — 1

Use Case — 2

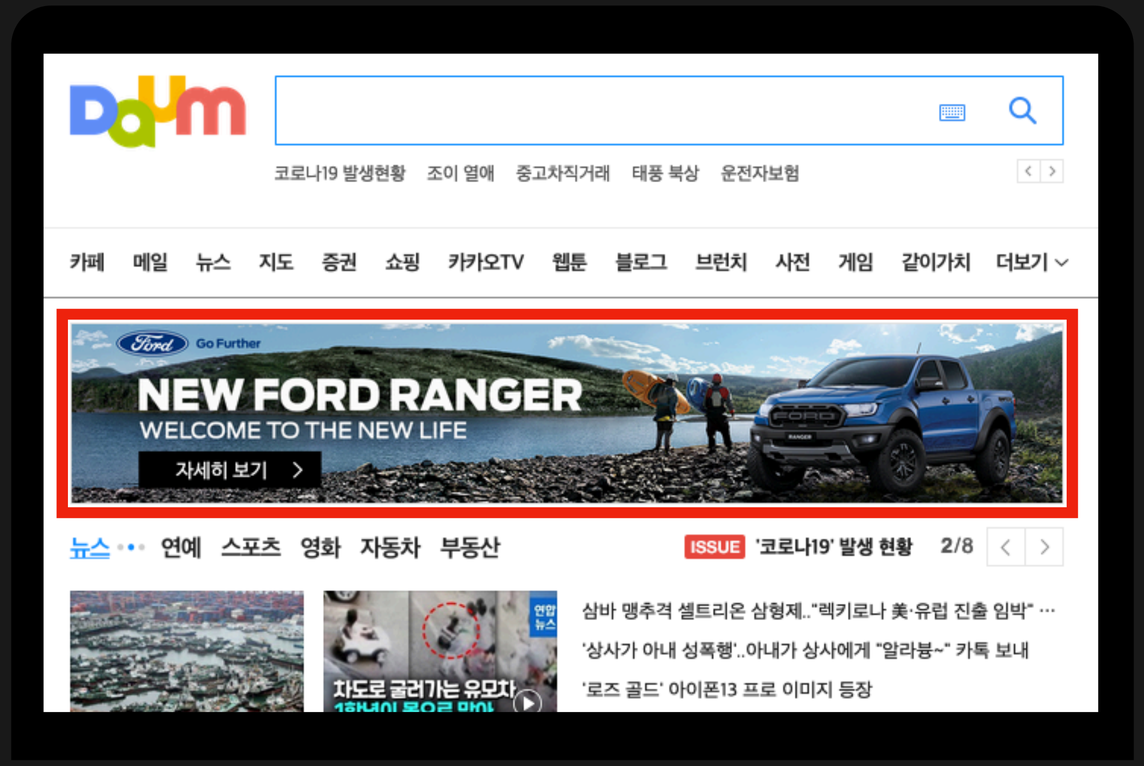
Use Case — 3

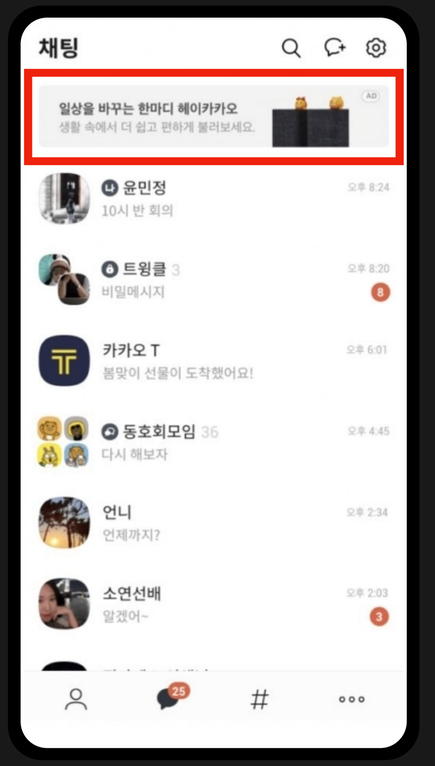
Use Case — 4

Multi-Tenant 클러스터 소개

Use Case — 1

- 개인화 광고 타게팅



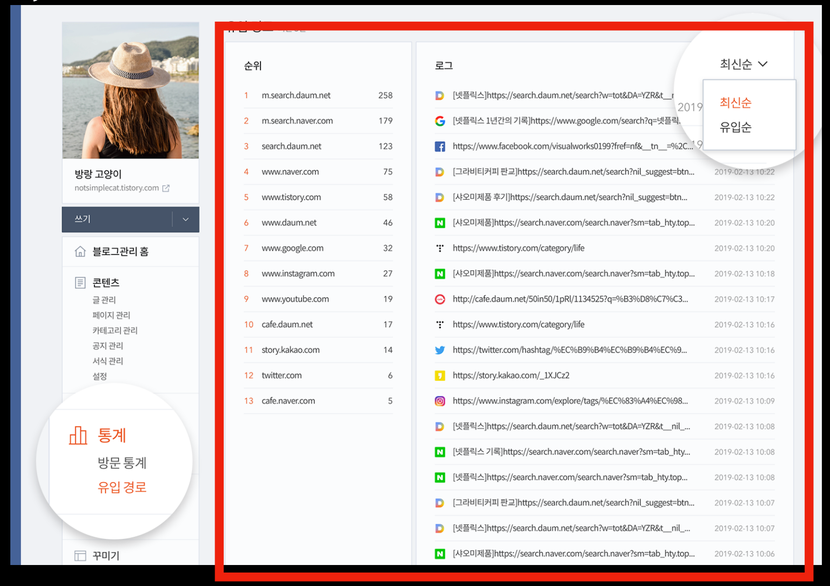


Use Case — 2

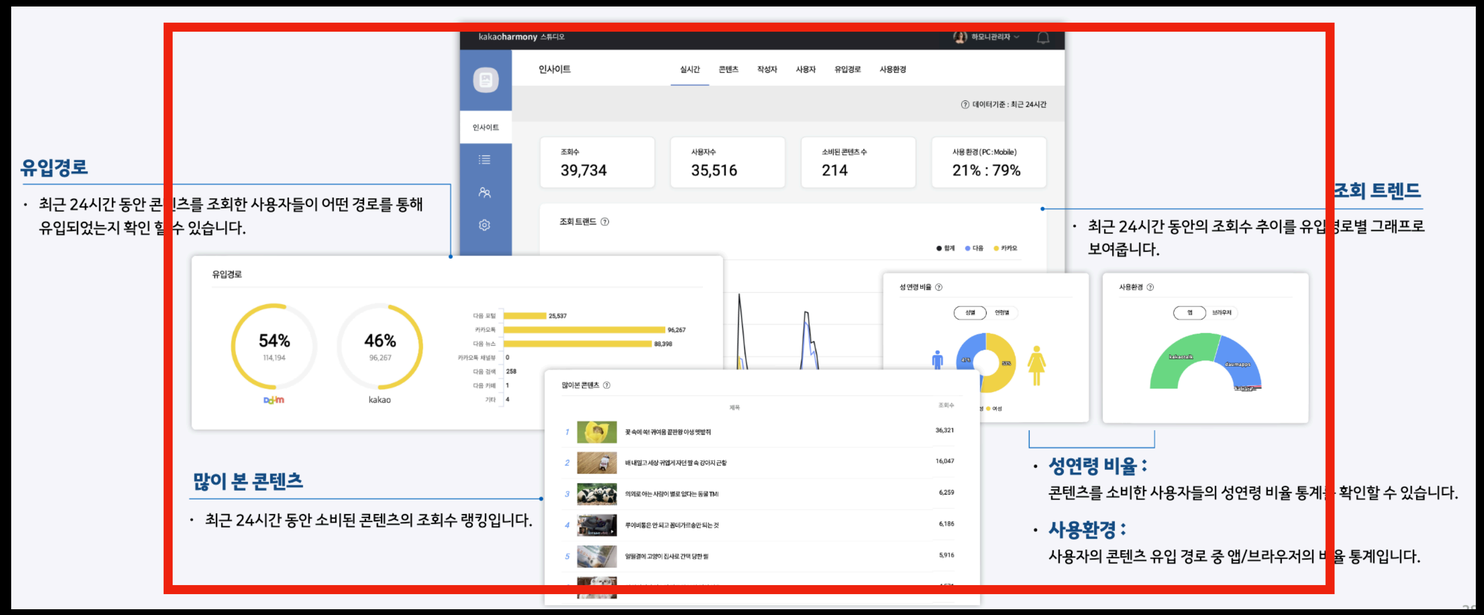
- 콘텐츠 관리자 통계 페이지 제공

1)

2)



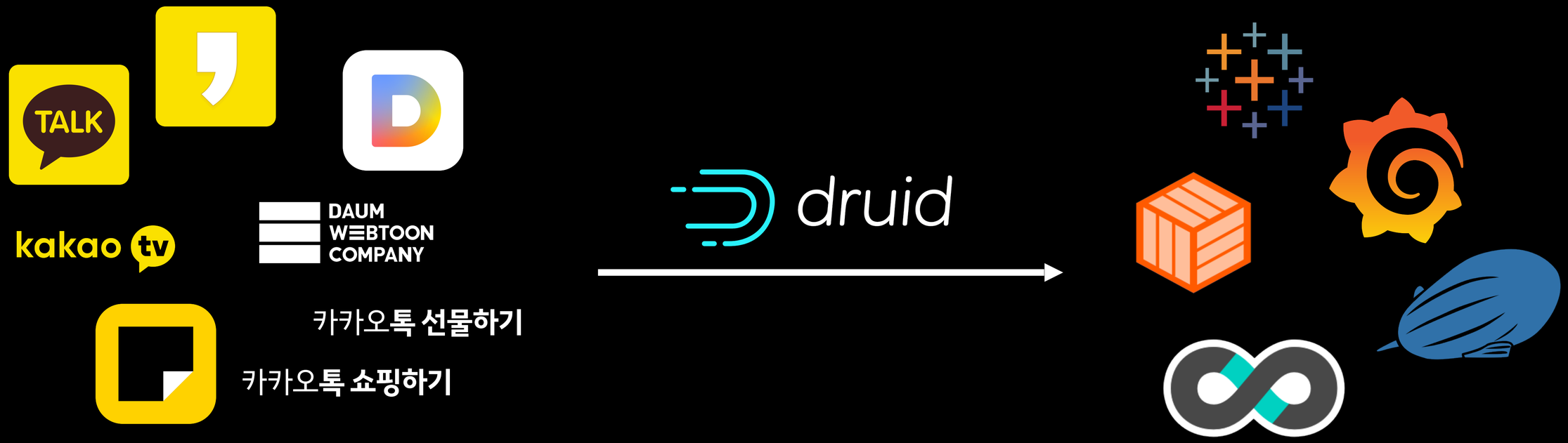
3)



1) https://notice.tistory.com/2466 | 2) https://notice.tistory.com/2473 | 3) https://t1.daumcdn.net/harmony_static/harmony/guide/카카오하모니 이용가이드_0806.pdf

Use Case — 3

- 사내 분석 용도
- 사용자 데이터의 실시간 다차원 분석



Use Case — 4

- 전사 서버 지표 실시간 수집 / 모니터링

Druid에 대한 기대

Use Case — 1

Use Case — 2

Use Case — 3

Use Case — 4

Multi-Tenant 클러스터 소개

50+ Servers

900+ CPU

6TB+ Memory

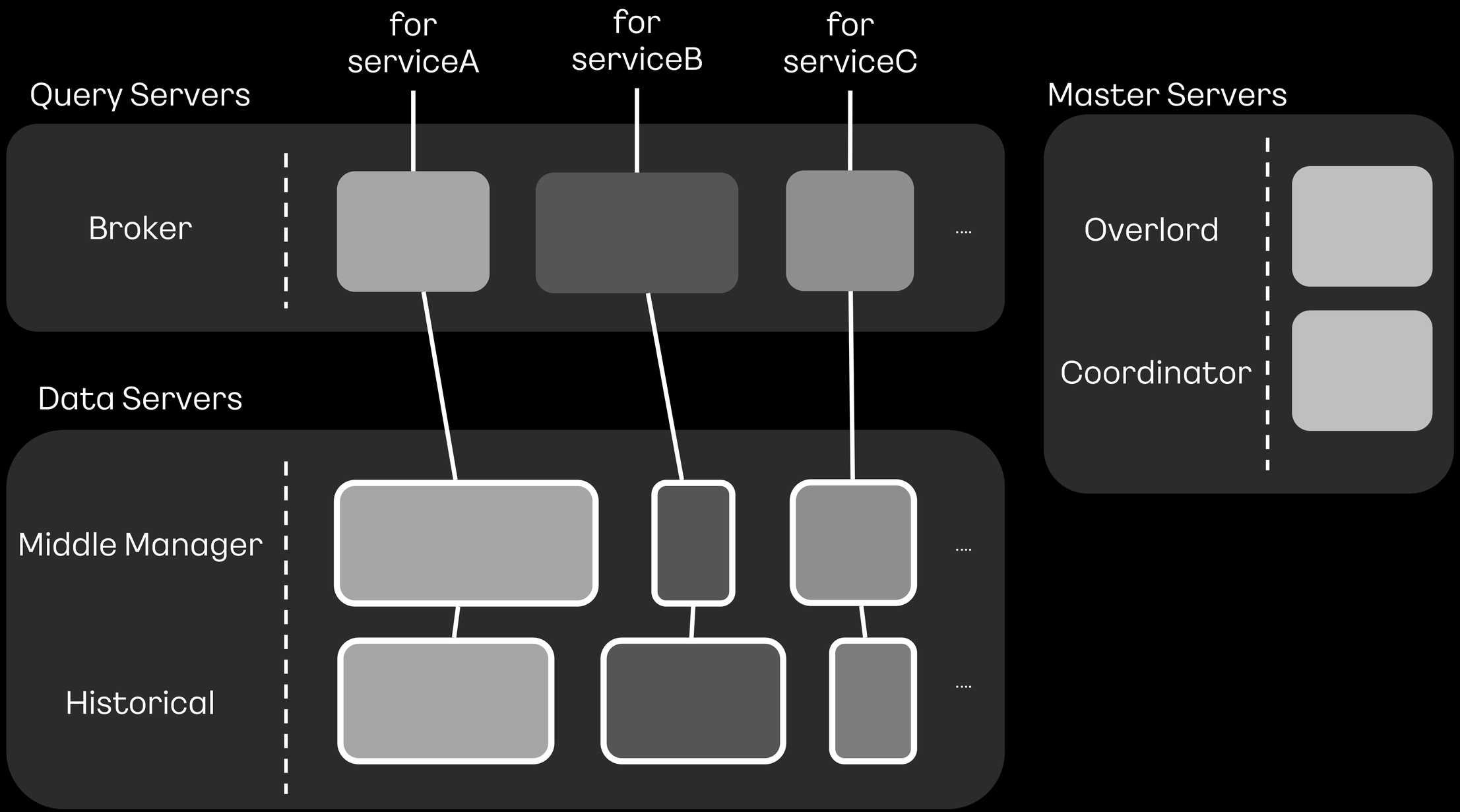
180TB+ Disk

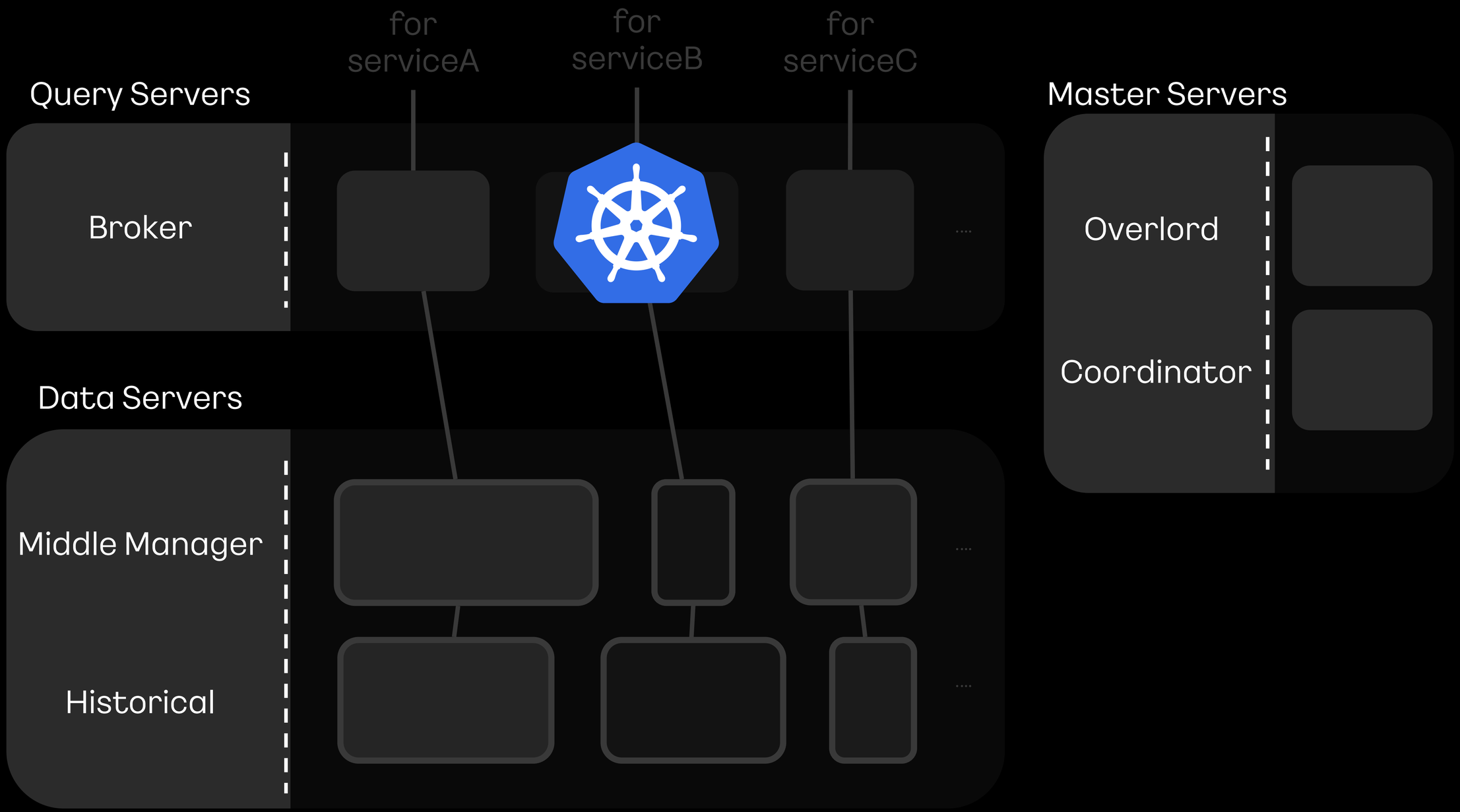
0.10.1 / 0.21.0 Druid Version

120+ Datasources

160bil+ Daily Write

15mil+ Daily Request


for
serviceA
for
serviceB
for
serviceC
Query Servers
Broker
....
Data Servers
Middle Manager
....
Historical
....
Master Servers
Overlord
Coordinator


for serviceA
for serviceB
for serviceC
Query Servers
Broker
Master Servers
Overlord
Coordinator
Data Servers
Middle Manager
Historical

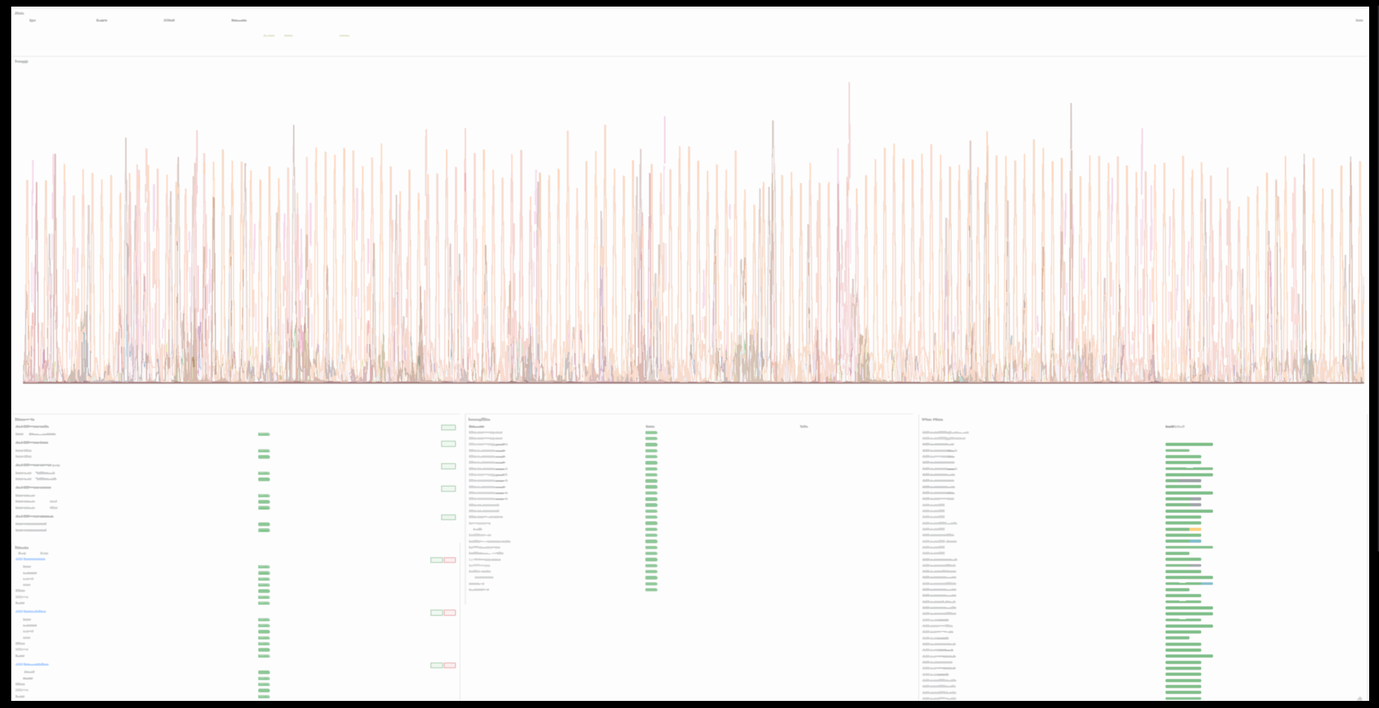

hadoop

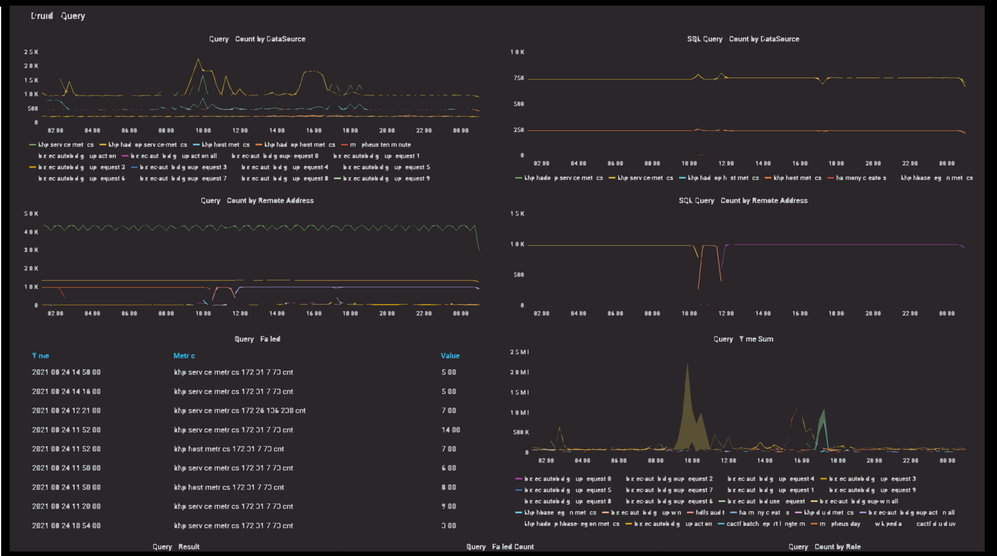

druid


druid


kafka


hadoop




Druid Query
Query Count by DataSource
SQL Query Count by DataSource
Query Count by Remote Address
SQL Query Count by Remote Address
Query Failed
Query Time Sum
Query Result
Query Failed Count
Query Count by Role

E.O.D