Tổng quan và đánh giá khả năng áp dụng Apache Hudi cho hệ thống CPM

LuanVT & MinhNX2

Nội Dung

I. Tổng quan về Hudi
II. VPS Data Big Picture
III. Hudi trong CPM
IV. Next Step

I. Tổng Quan Về Hudi

I. Tổng Quan Về Hudi

1. Hudi là gì?
2. Tính năng của Hudi
3. Design & Concepts
4. Hudi Writing
5. Hudi Reading
6. Table Services

I. Tổng Quan Về Hudi

1. Hudi là gì?

Hình 1.1: Hudi Overview

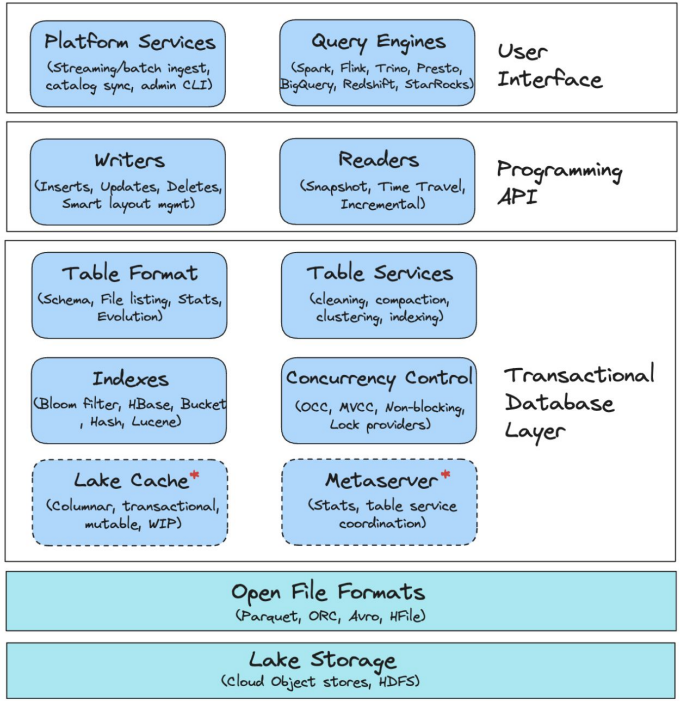
I. Tổng Quan Về Hudi

2. Tính năng của Hudi

- ✓ Mutability support for all data lake workloads
- ✓ Improved efficiency by incrementally processing new data
- ✓ ACID Transactional guarantees to data lake
- ✓ Unlock historical data with time travel
- ✓ Interoperable multi-cloud ecosystem support
- ✓ Comprehensive table services for high-performance analytics
- ✓ A rich platform to build lakehouse faster
- ✓ Query acceleration through multi-modal indexes
- ✓ Resilient Pipelines with schema evolution & enforcement

3. Design & Concepts

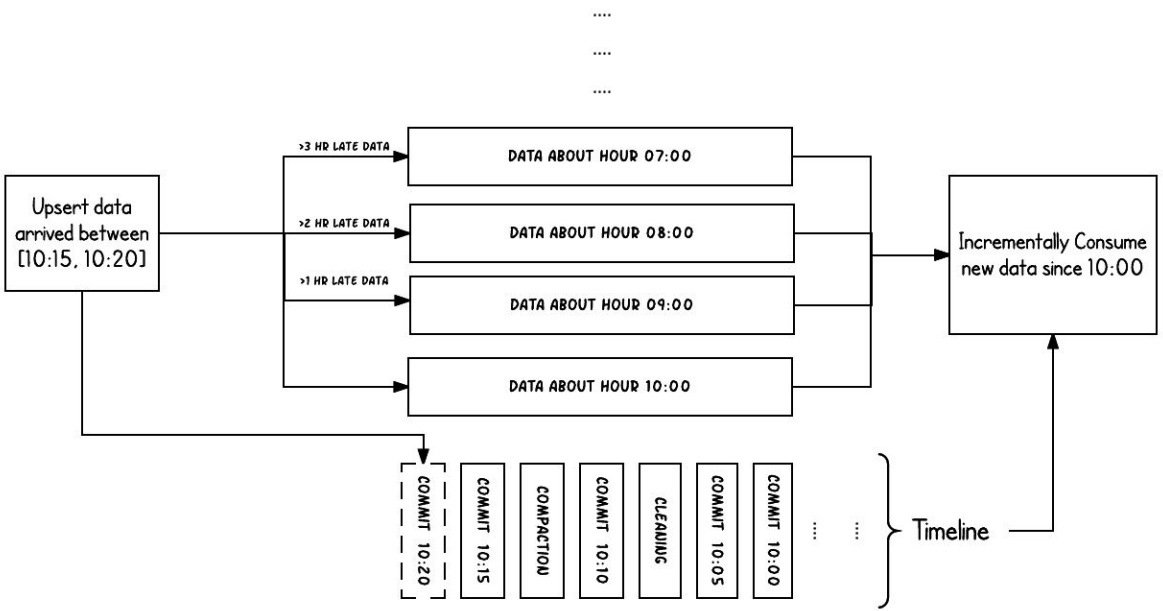
a. Hudi Architecture



Hình 1.2: Hudi Architecture

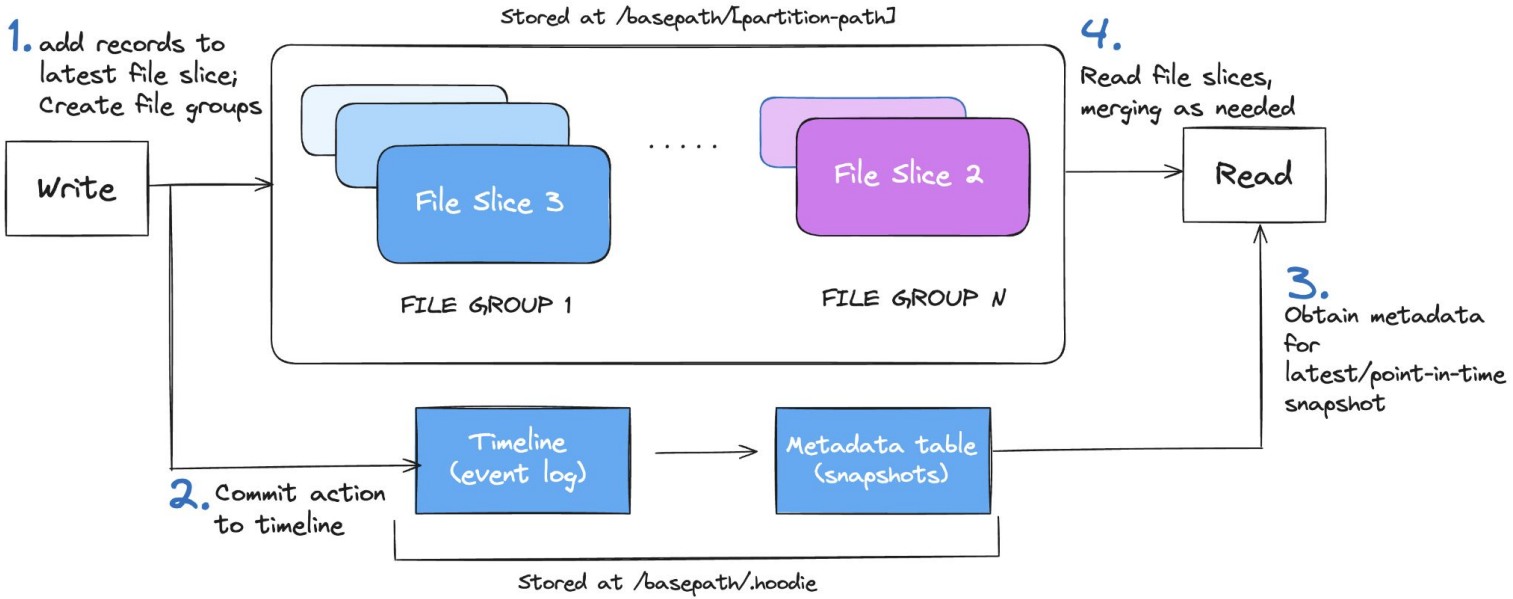
3. Design & Concepts

b. TimeLine



3. Design & Concepts

c. File layouts



3. Design & Concepts

d. Table Types (1)

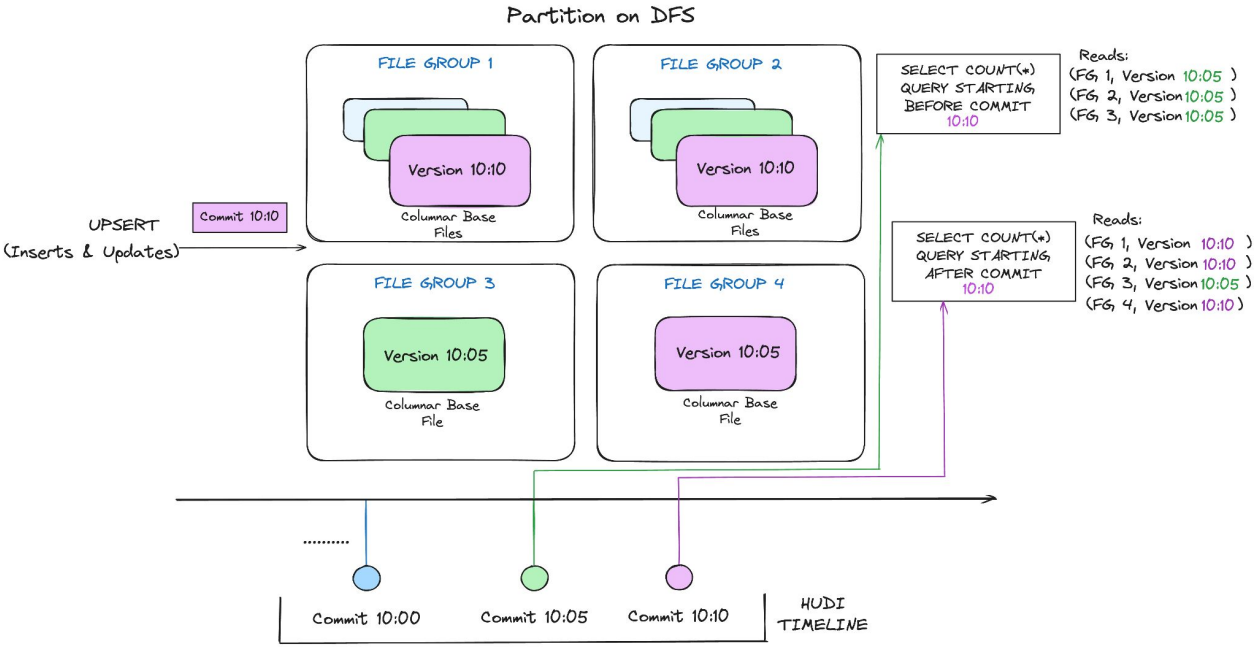
Hudi supports two table types:

- Copy On Write : Stores data using exclusively columnar file formats (e.g parquet). Updates simply version & rewrite the files by performing a synchronous merge during write.
- Merge On Read : Stores data using a combination of columnar (e.g parquet) + row based (e.g avro) file formats. Updates are logged to delta files & later compacted to produce new versions of columnar files synchronously or asynchronously.

3. Design & Concepts

d. Table Types (2)

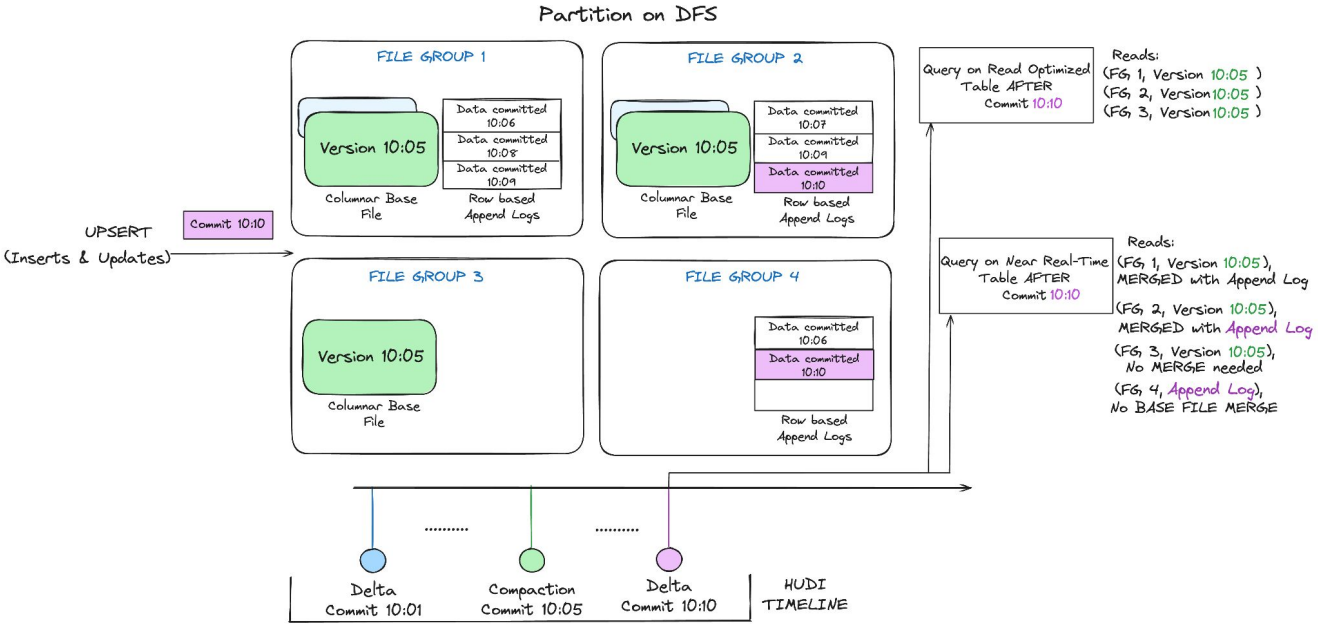
➢ Copy On Write Table



3. Design & Concepts

d. Table Types (3)

➢ Merge On Read Table



3. Design & Concepts

d. Table Types (4)

➢ Trade-off between two table types

Trade-off	**Copy On Write**	**Merge On Read**
Data Latency	Higher	Lower
Query Latency	Lower	Higher
Update cost (I/O)	Higher (rewrite entire parquet)	Lower (append to delta log)
Parquet File Size	Smaller (high update(I/0) cost)	Larger (low update cost)
Write Amplification	Higher	Lower (depending on compaction strategy)

3. Design & Concepts

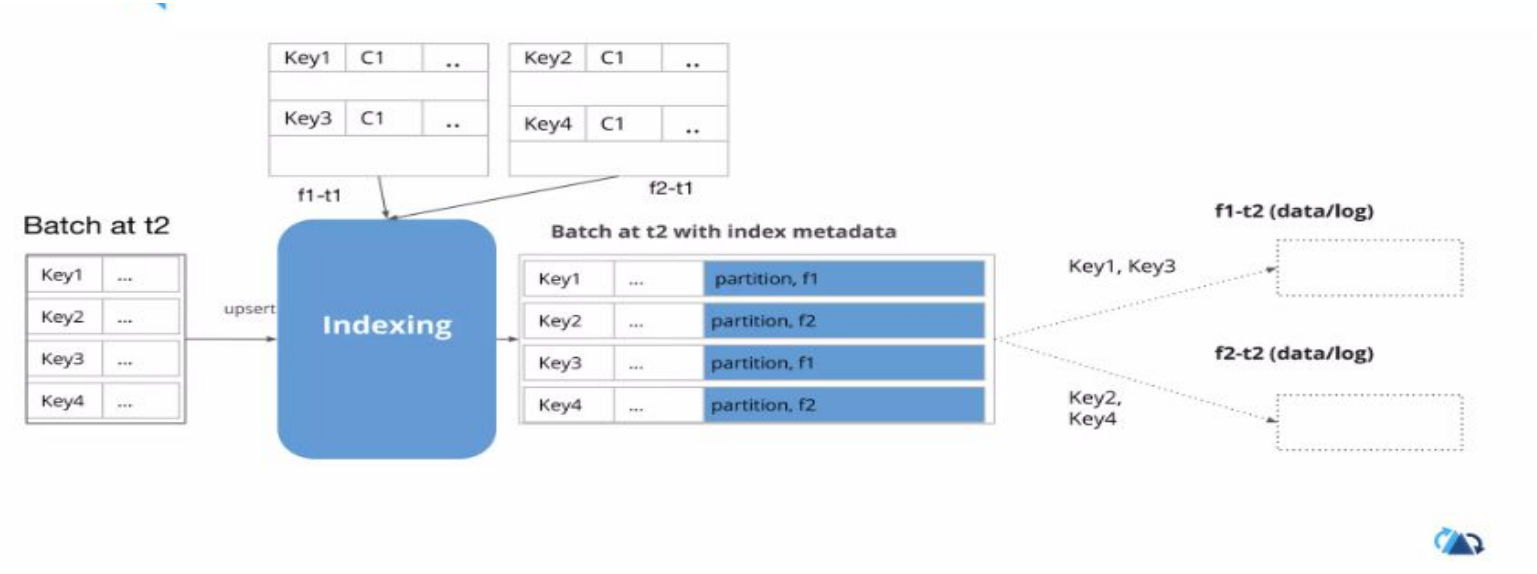
e. Query Types

- Snapshot Query
- Time Travel Query
- Change Data Capture Query
- Incremental Query

Table Type	Supported Query types
Copy On Write	• Snapshot Queries • Incremental Queries • Incremental Queries (CDC) • Time Travel
Merge On Read	• Snapshot Queries • Incremental Queries • Time Travel

3. Design & Concepts

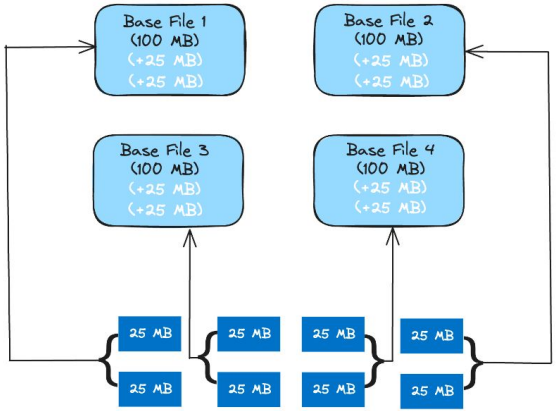
f. Indexing

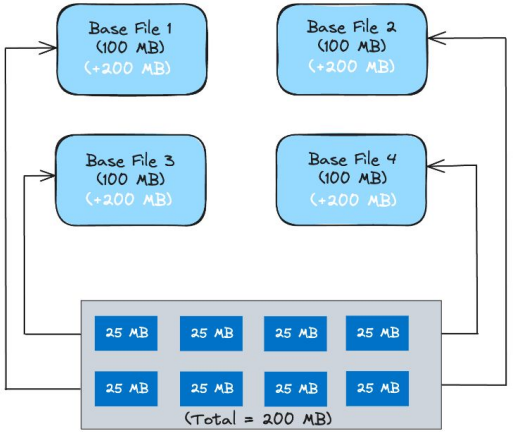


3. Design & Concepts

f. Indexing

Comparison of merge cost for updates (dark blue blocks) against base files (light blue blocks)





3. Design & Concepts

g. Write Operations

- Insert
- Bulk Insert
- Insert Overwrite
- Insert Overwrite Table
- Upsert
- Delete
- Rollback/Restore

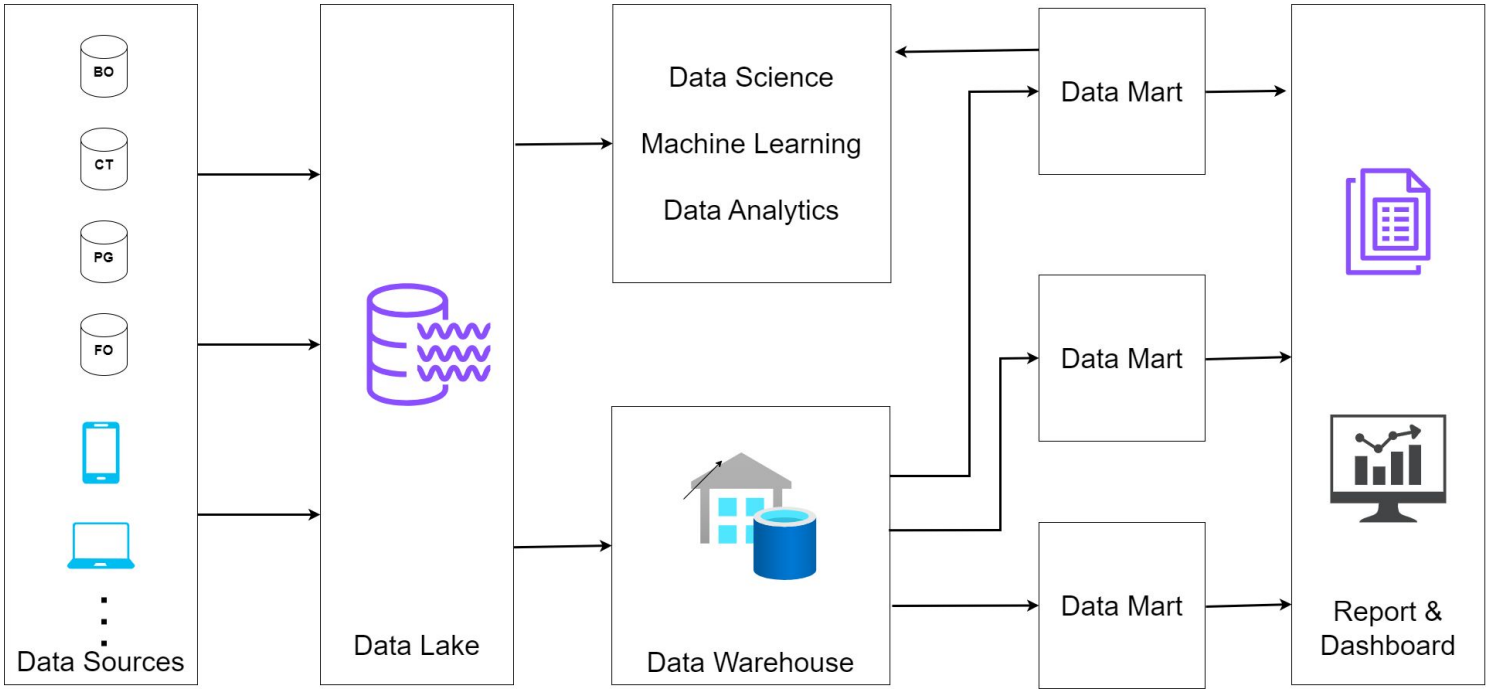
Table services

- Clustering
- Compaction
- Clean
- Rollback
- File Sizing

II. VPS Data Big Picture

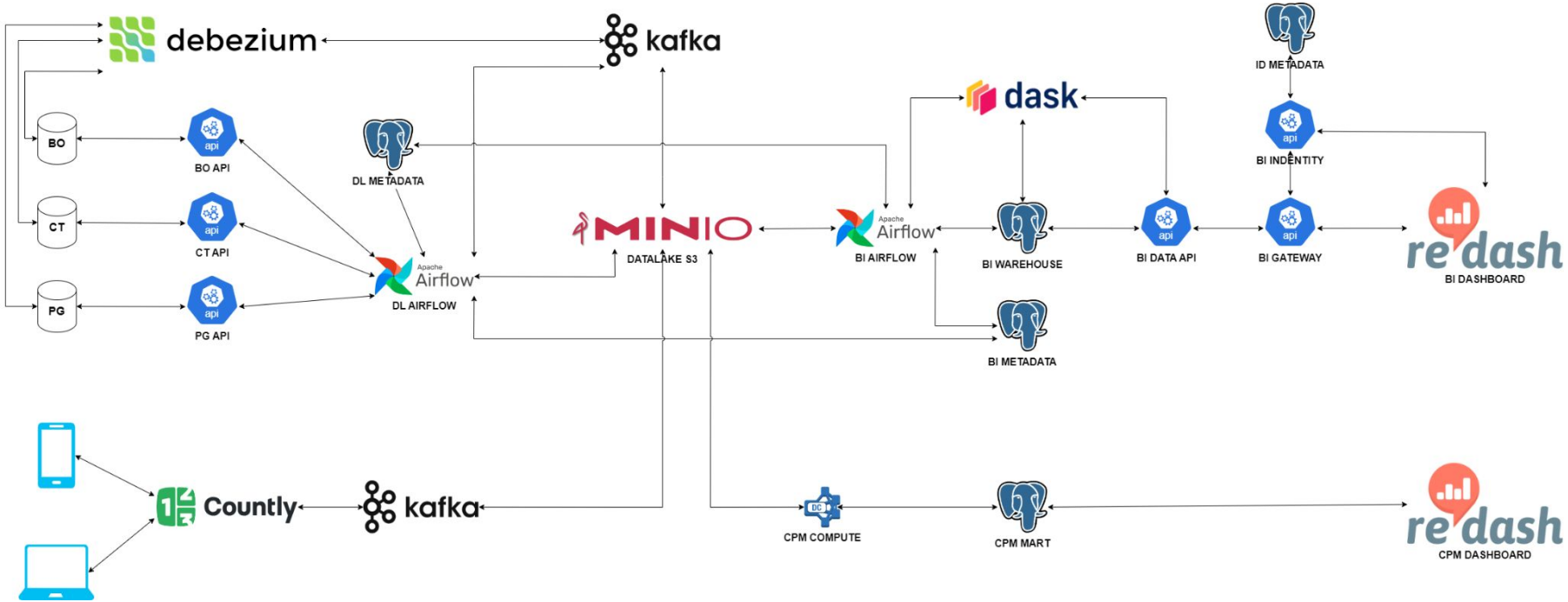
II. VPS Data Big Picture

1. Data Big Picture at VPS



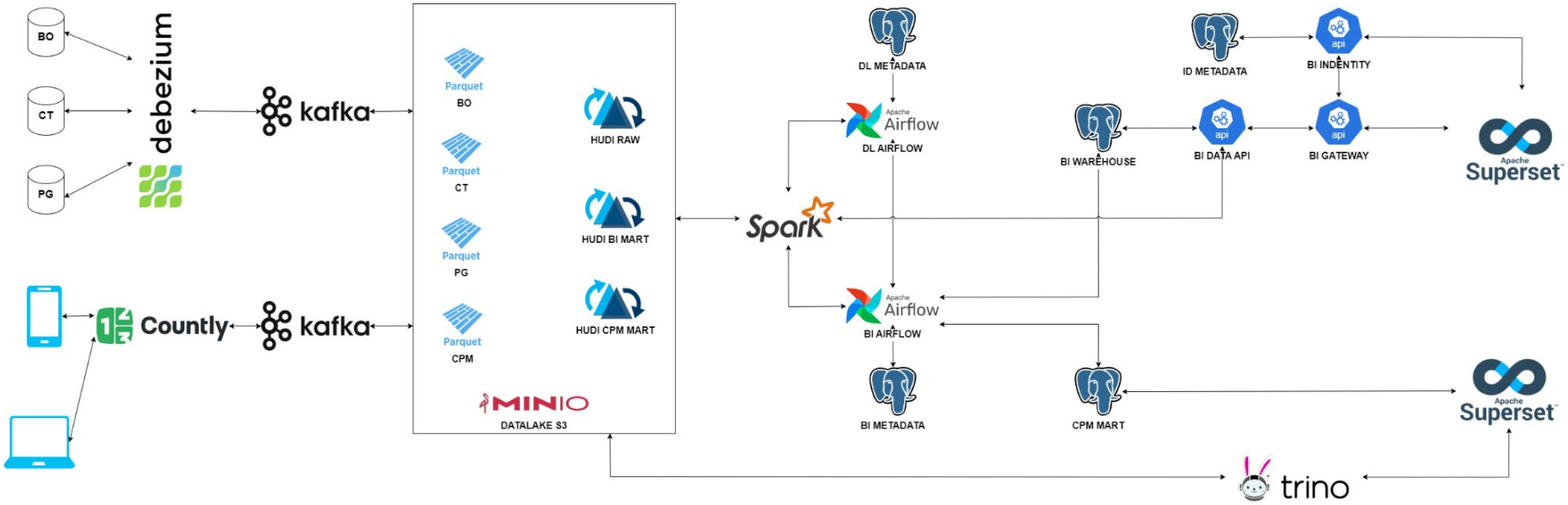
II. VPS Data Big Picture

2. Hệ thống hiện tại



II. VPS Data Big Picture

2. Đề xuất hệ thống khi áp dụng Hudi



III. Áp dụng Hudi cho CPM

III. Áp dụng Hudi cho CPM

1. Challenges

Đầu vào:

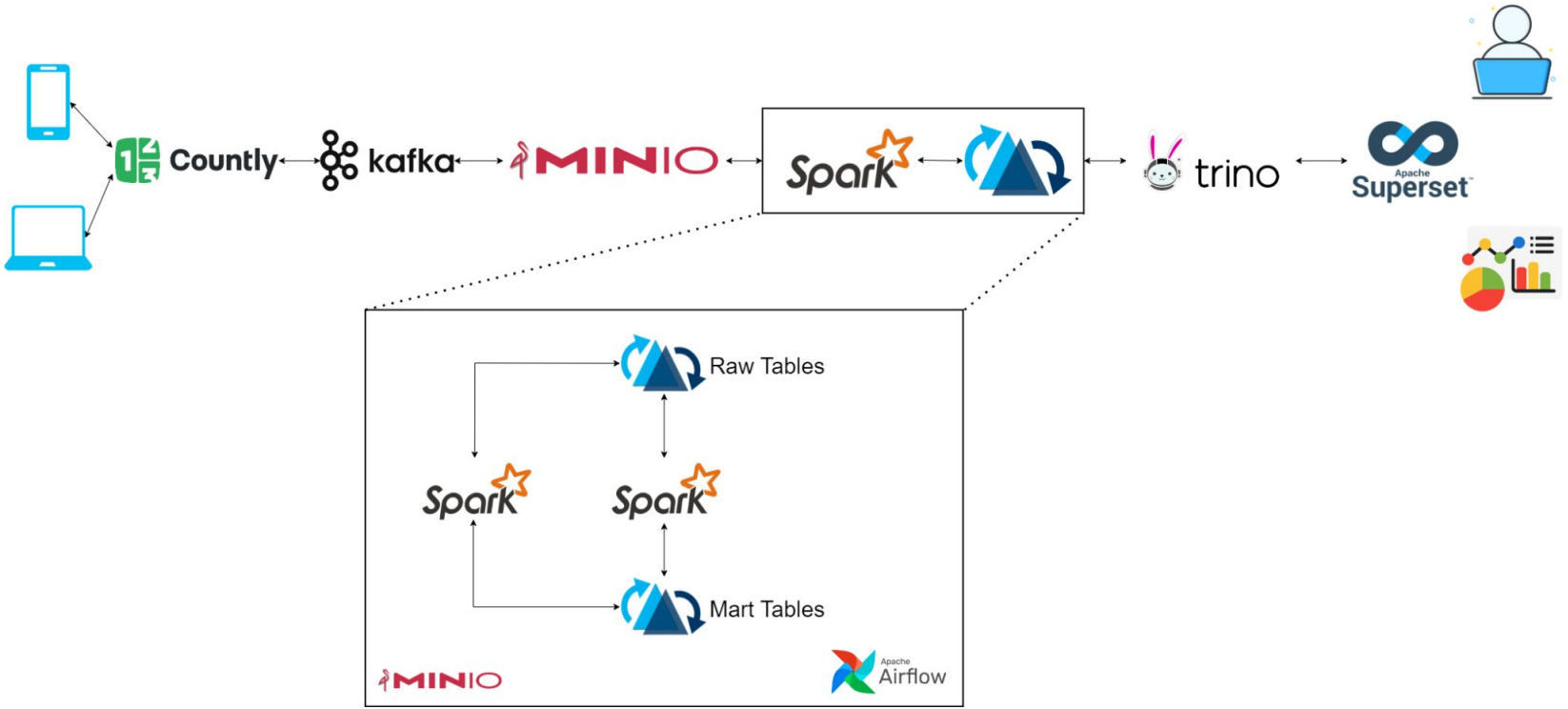
1. Event tổng hợp theo dạng batch vào 21h~24h hàng ngày
2. Event được chia thành nhiều file lưu trên Minio với giới hạn số lượng max 50k event trên 1 file, dung lượng max 1 file ~ 6.7M

Thách thức:

1. Bulk upload to Parquet (2GB data ~14M events) của CPM event mỗi 24h
2. Xử lý và lưu trữ các data mart dựa trên 14M raw events
3. Schema change
4. Tối ưu tốc độ đọc dữ liệu từ raw để tổng hợp mart
5. Số lượng lớn I/O
6. Concurrency Writing
7. Concurrency Reading
8. Số lượng lớn Reading vào các bảng mart

III. Áp dụng Hudi cho CPM

2. CPM Data Pipeline



III. Áp dụng Hudi cho CPM

3. Cấu trúc dữ liệu Event


```
{
  "key": "derivatives_order_long",
  "count": 1,
  "timestamp": 1722823233217,
  "hour": 9,
  "dow": 0,
  "segmentation": {
    "event_feature": "derivatives_order_long",
    "acountId": "KH0009",
    "sessionid": "514E18E2BA3C888855C047339E998740",
    "version_app": "5.9.5ops-18682(2277)",
    "tracking_id": ""
  },
  "sum": 0,
  "ts": 1721630145546,
  "reqts": 1721630145376,
  "did": "4ade518f-2d5f-3b09-e9a8-0c2382fee093",
  "headers": {
    "x-forwarded-for": "10.32.79.1, 10.32.79.212",
    "x-real-ip": "10.32.79.212",
    "host": "countly_api",
    "connection": "close",
    "user-agent": "Dalvik/2.1.0 (Linux; U; Android 13; SM-A057F Build/TP1A.220624.014)",
    "accept-encoding": "gzip"
  },
  "app_id": "63b794f5fd0e4e027ae3fadf",
  "app_key": "51cbd1f7fa73f06c1c19ee8bacac373680da4d24"
}
```


Key	Ý nghĩa
key	Tên event
count	Số lượng click vào event
acountId	ID tài khoản 6 số của khách hàng
sessionid	ID của session hiện tại đăng nhập của khách hàng
version_app	App version SMO
tracking_id	Thông tin id gửi từ campaign
did	Device ID của thiết bị
timestamp	Thời gian phát sinh event trên thiết bị SMO
ts	Thời gian gửi event trên thiết bị SMO
reqts	Thời gian event nhận được trên Countly
segmentation	Thông tin chi tiết event gửi đến countly
app_id	ID của site application, web, smo, smp, ...
app_key	Key của site application, web, smo, smp, ...
headers	Thông tin header của request client đến countly

III. Áp dụng Hudi cho CPM

4. Đề bài

- Thiết kế bảng lưu trữ dữ liệu raw event cho CPM
- Thiết kế bảng mart phục vụ yêu cầu tính tổng số lượng các event theo ngày
- Thiết kế và triển khai pipeline cho bảng raw và mart
- Tối ưu tốc độ write và read cho các bảng
- Giải quyết bài toán khối lượng lớn query đồng thời vào bảng mart

III. Áp dụng Hudi cho CPM

5. Lựa chọn cấu hình

b. Raw Table

Tên cấu hình	Thông số
Table Type	Copy on Write
Write Operation	Insert Overwrite
Record key	X
Partition	data_date
File Sizing	100MB <= file size <= 120MB
Compaction	X
Clustering	Execution Strategy
Indexing	SIMPLE

III. Áp dụng Hudi cho CPM

5. Lựa chọn cấu hình

c. Mart Table

Colum	Ý nghĩa
data_date	Ngày dữ liệu
event	Tên event
total	Số lượng

Tên cấu hình	Thông số
Table Type	Copy on Write
Write Operation	Insert Overwrite
Record key	X
Partition	data_date
File Sizing	100MB <= file size <= 120MB
Compaction	X
Clustering	X
Indexing	SIMPLE

6. Demo

IV. Next Step

IV. Next Step

- Test performance
- Scale hệ thống
- Áp dụng thực tế cho CPM
- Áp dụng cho Datalake và BI

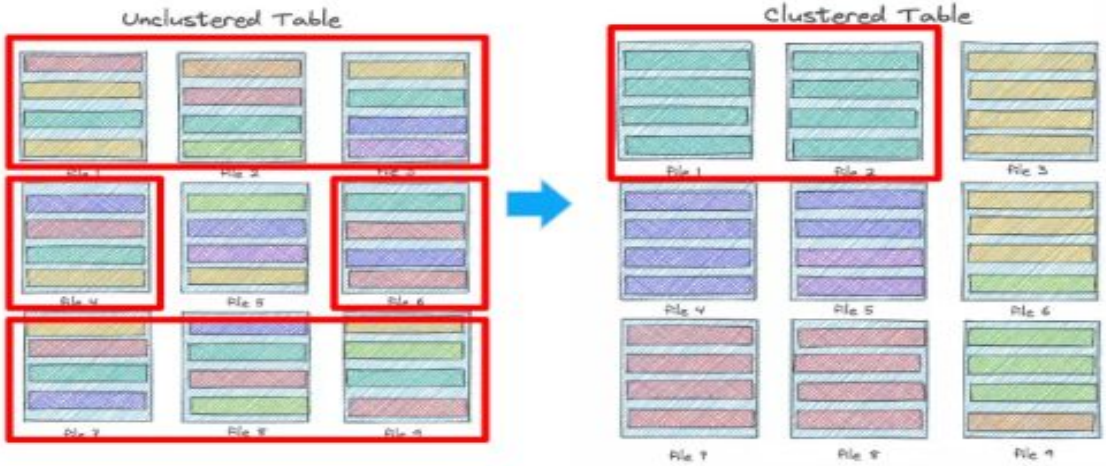
Thank You!

8. Apache Hudi vs Delta Lake Comparisons

https://www.onehouse.ai/blog/apache-hudi-vs-delta-lake-vs-apache-iceberg-lakehouse-feature-comparison

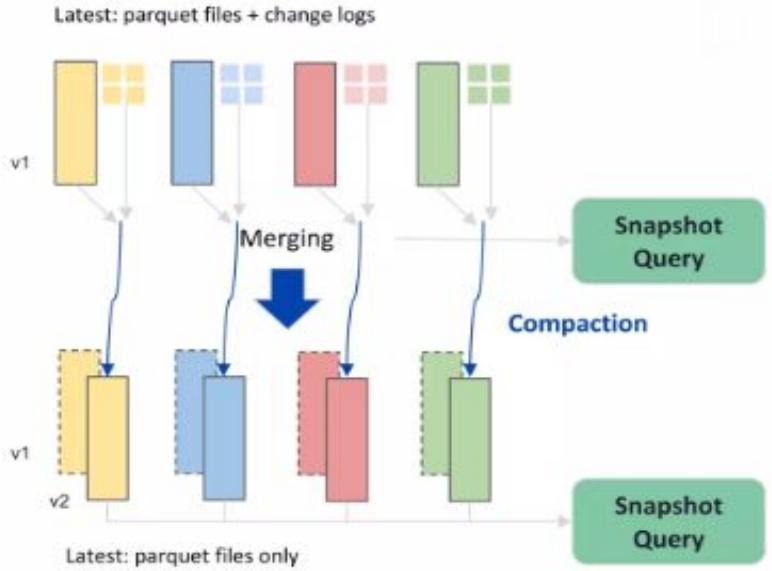
Clustering - Optimizing Data Layout

- Faster streaming ingestion -> smaller file sizes
- Data locality for query (e.g., by city) ≠ ingestion order (e.g., trips by time)
- Clustering to the rescue: auto file sizing, reorg data, no compromise on ingestion



Compaction - Balancing Read/Write Costs

- TBs of updates against PBs of data
- Delete/Update patterns often very different than query patterns
  - GDPR deletes are random
  - Analytics Queries more likely to read recent data
- Periodically and asynchronously compact log files to new base files
- Reduces write amplification
- Keep the query performance in check



Clustering Service

- Complete runtime for executing clustering in tandem with writers/compaction
- Scheduling: identify target data, generate plan in timeline
- Running: execute plan with pluggable strategy
- Reorg data with linear sorting, Z-order, Hilbert, etc.

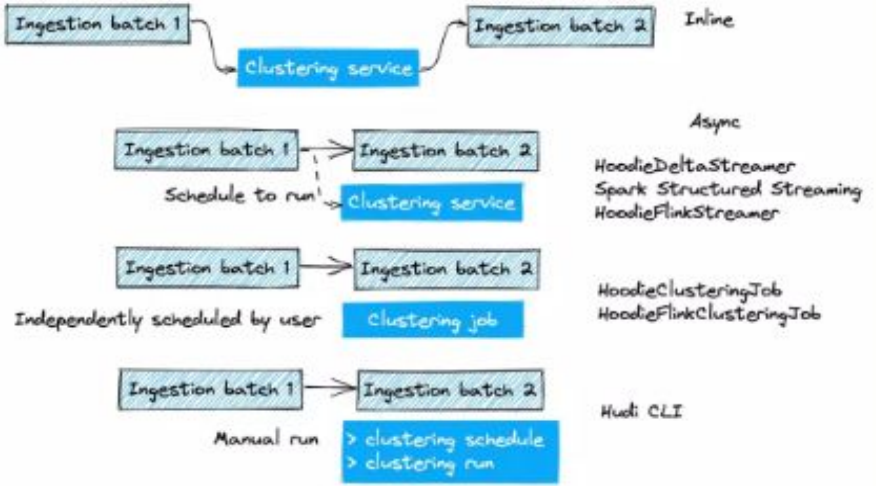


Table Services with Continuous Writers

- Self managing database runtime
  - Cleaning (committed/uncommitted), archival, clustering, compaction
  - Similar to how RocksDB daemons work
- Table services know each other
  - Avoid duplicate schedules
  - Skip compacting files being clustered
- Run continuously or scheduled, asynchronously