Apache Flink empowered large-scale near real-time (NRT) data analytics platform

Ying Xu, Streaming Platform, Lyft Inc

Kailash HD, Streaming Platform, Lyft Inc

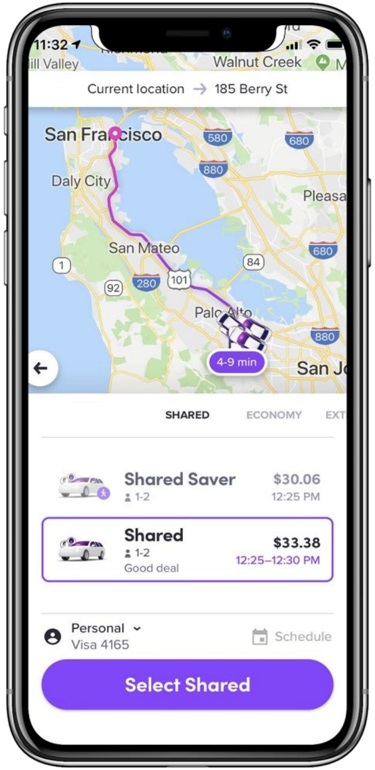
Agenda

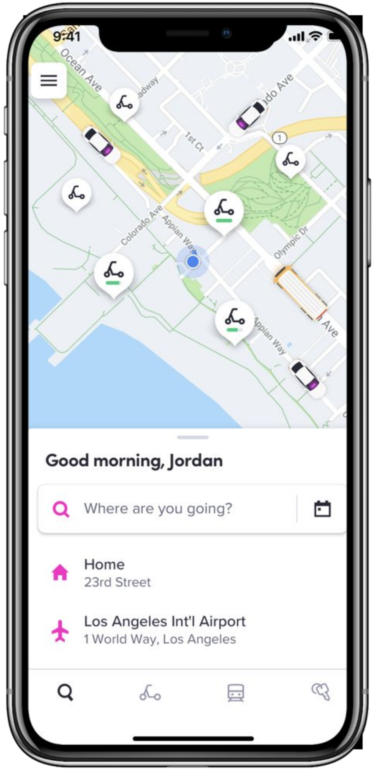
- Streaming data scenarios at Lyft
- Architecture of near real-time data analytics platform
- Deep dive on platform design and fault tolerance
- Summarization and future directions

Streaming data scenarios at Lyft

About Lyft

MISSION: Improve people's life with the world's best transportation





Streaming data scenarios at Lyft

Streaming Events Enrichment

seconds

Real-time Adaptive Pricing

minute

Fraud and Anomaly Detection

minute

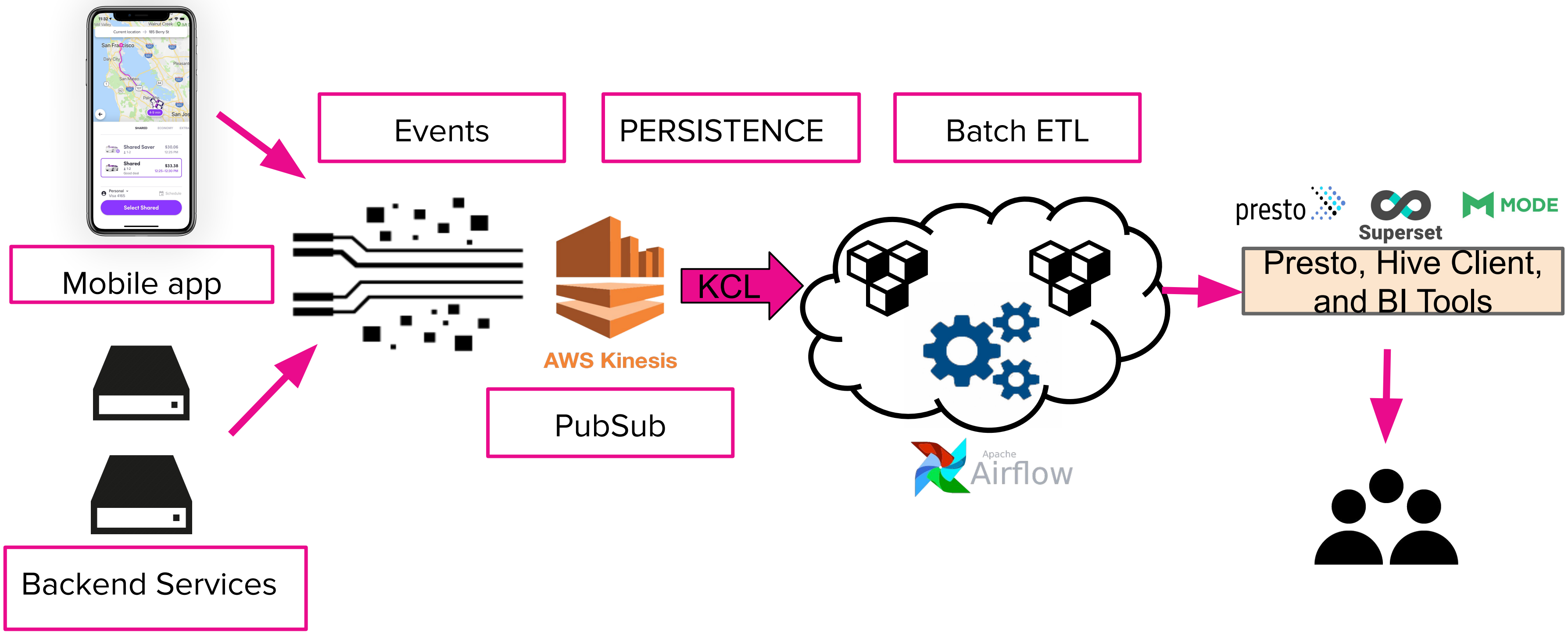
ML Feature Engineering

minute

Near Real-time Interactive Query

< 5 minutes

Lyft’s data analytics platform architecture

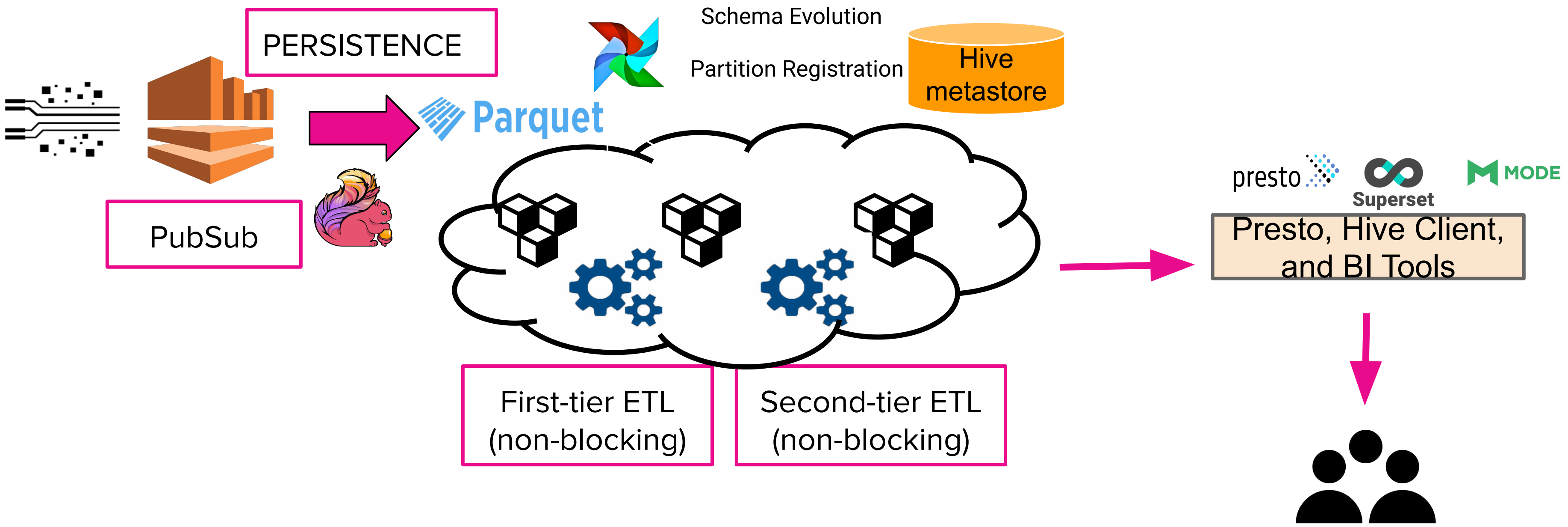


Issues of the legacy platform

- Persisted data cannot be ready for query in near real-time
- Streaming persistence using KCL exhibit limited performance
- Presence of too many small files limits performance of S3 operations
- Most ETL were scheduled on a daily basis and have multi-day latency
- Legacy platform offers limited support for nested data

Architecture of near real-time data analytics platform

Near real-time data analytics platform architecture



Platform Design

- Highly efficient streaming persistence: Flink with StreamingFileSink
- Data persisted in Parquet supporting interactive query
- Multi-stage (hourly) ETL for enhanced performance and data quality
- Performance, fault tolerance and evolvability built into the design

Platform Characteristics and use cases

Hundreds of billions of events per day

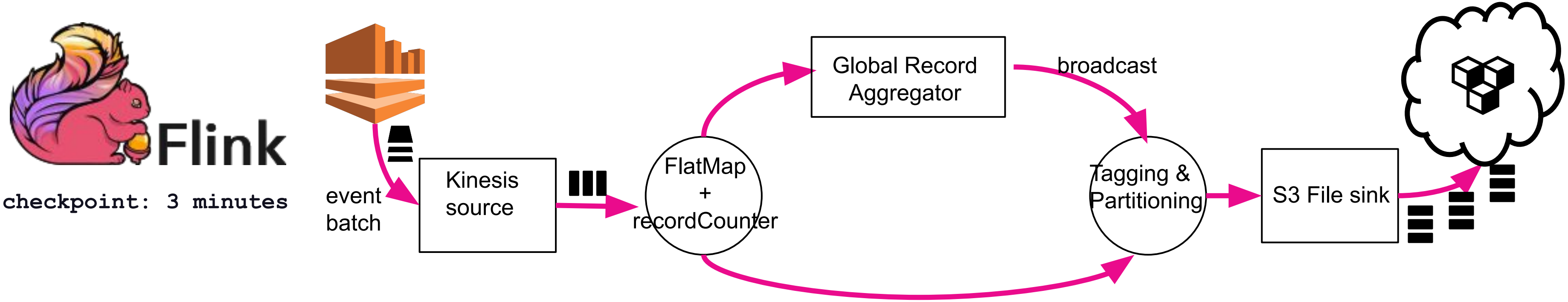
Data Freshness
(< 5 minutes)

Data Completeness
(at least once)

Data Uniqueness
(ETL dedup)

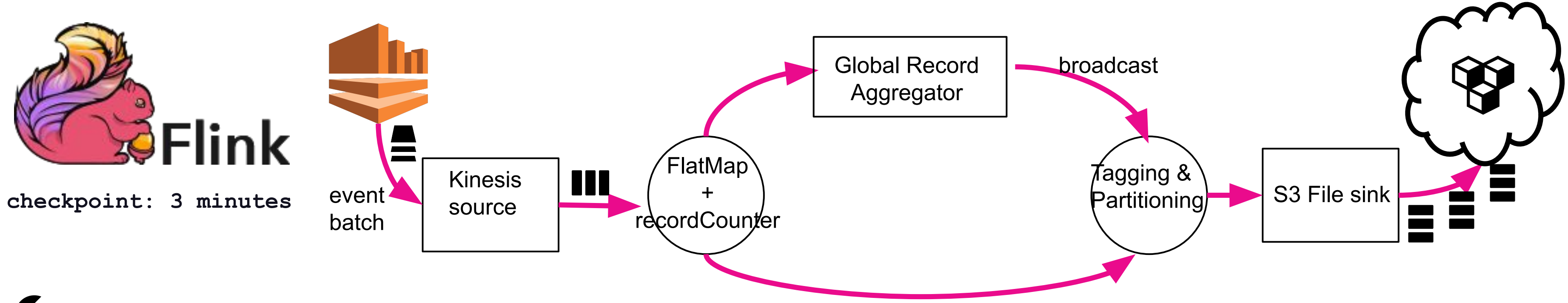
- Ad-hoc interactive queries
- Real-time dashboards for marketplace health monitoring
- Real-time alerting on ML model accuracy

Flink empowered near real-time data ingestion



- Flink kinesis source connector (watermark and source sync)
- StreamingFileSink unlocks writing bulk-encoded data in Parquet
- Checkpoint interval: 3 minutes

Flink empowered near real-time data ingestion

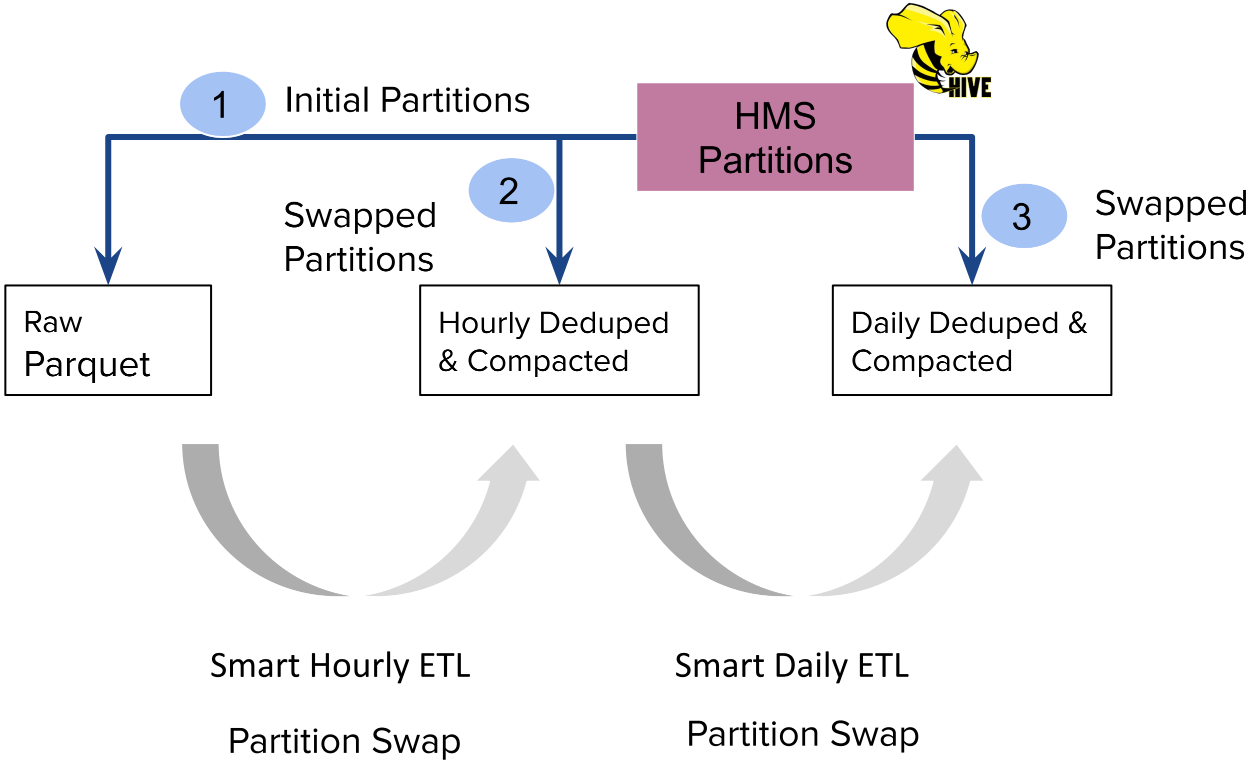


Too-many-small-files problem

- Subtask records event counts and broadcast periodically
- Global record aggregator: windowing function collecting event weights
- Tagging function: (event name) -> (subtask)

```
(# of subtasks) = event_weight * sink_parallelism
```

ETL Multi-tier compaction and deduplication



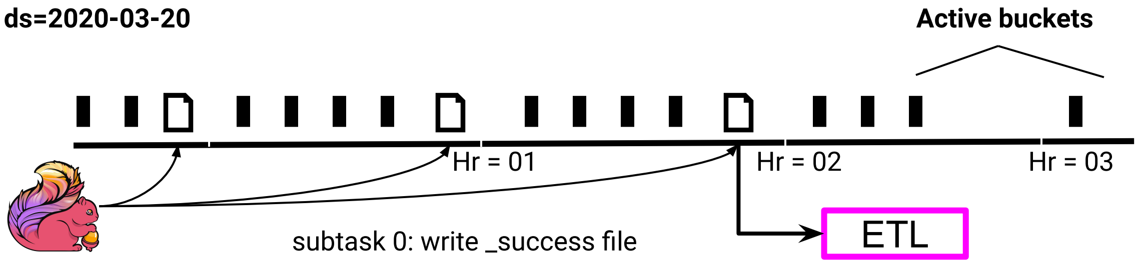
Deep dive on platform design and fault tolerance

Event-time driven partition sensing - Flink

- AWS S3 partition scheme

```
s3://rawevents/$entropy/<event_name>/ds=yyyy-MM-dd/hr=HH
```

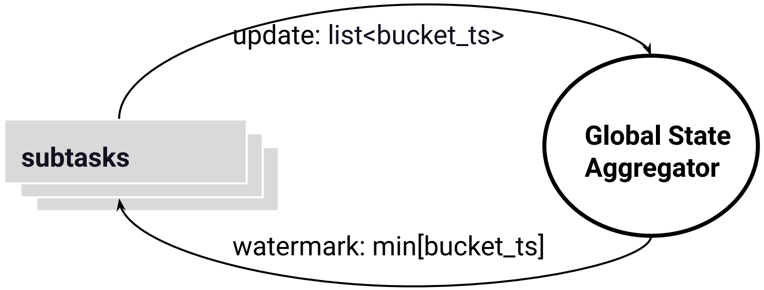
- Event-time driven partition sensing



- Success-file driven partition sensing

Event-time driven partition sensing - Flink

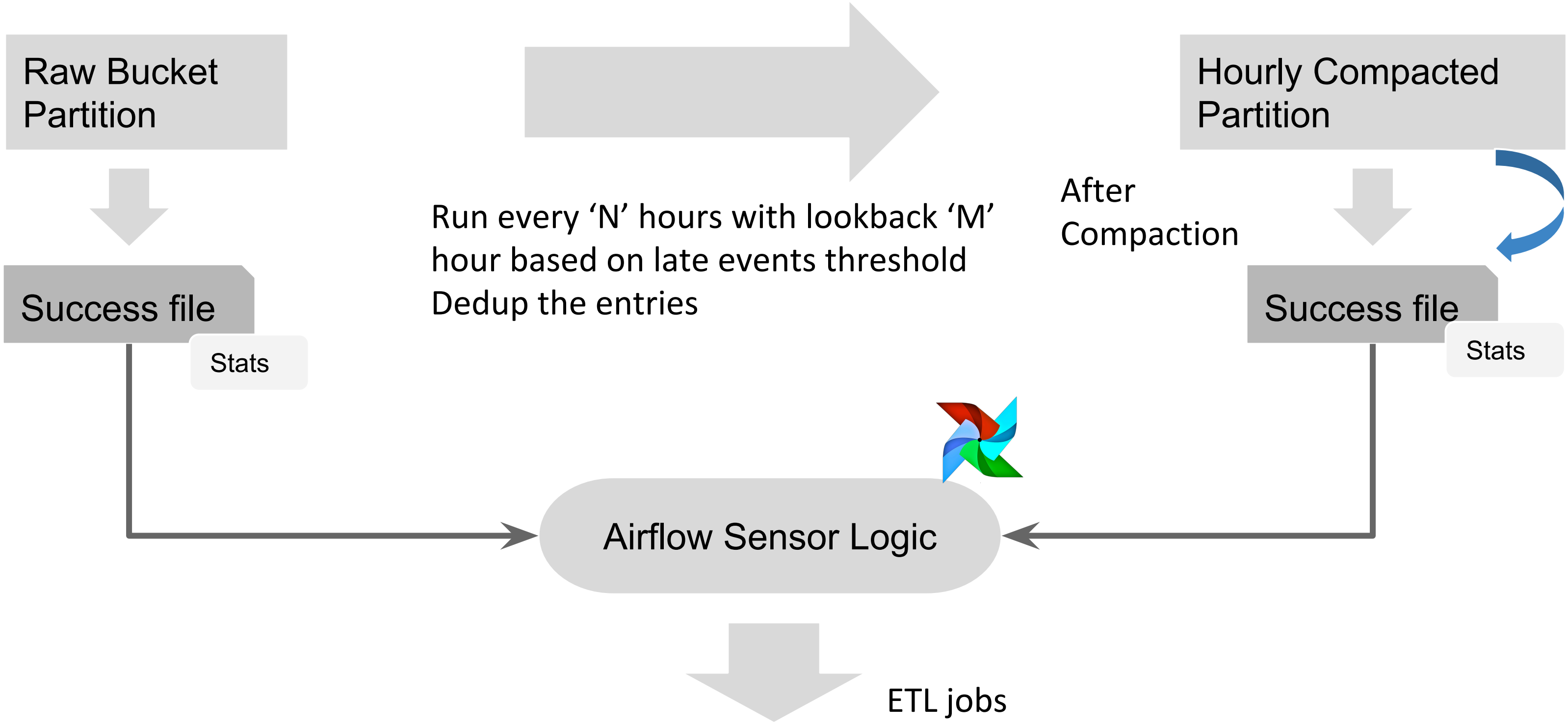
- Constructing bucket watermark



- Bucket watermark stored as job state

```
private transient ListState<byte[]> successFileState;
```

Event-time driven partition sensing: ETL

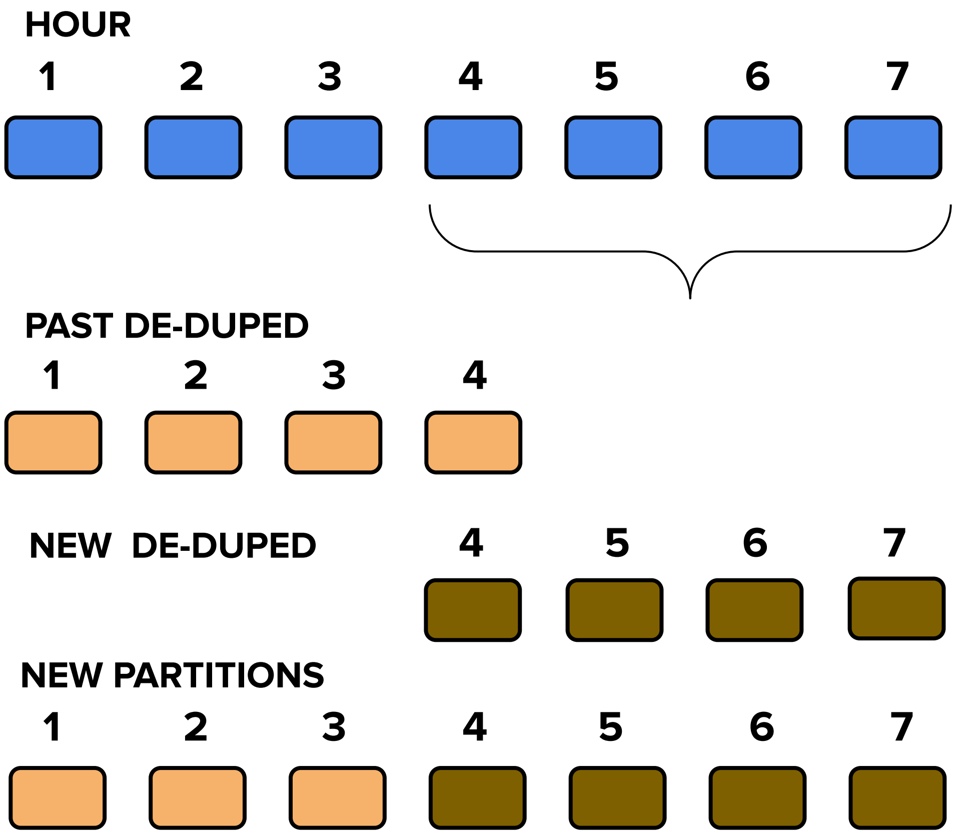


Deduplicate Records

- Provide guarantees that there are no duplicates.
  - UUID which is part of all events to unique identify an event.

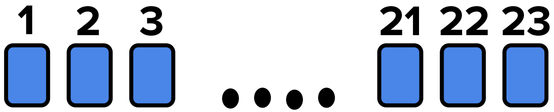
Challenges

- Not all events have the same duplicates rate -
  - Events from mobile clients are likely to have more
- Duplicates events (late arriving) can occur few hours later
  - Need to run dedup at different frequencies
- Dedup logic is different based on event format

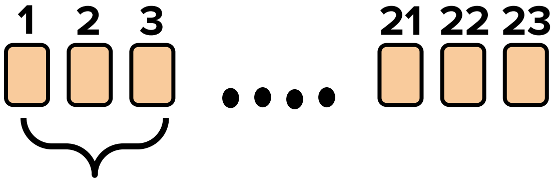


S3 Layout and Partition Management

Raw bucket layout (hourly)
Single S3 bucket with prefix containing event_name, ds, hour

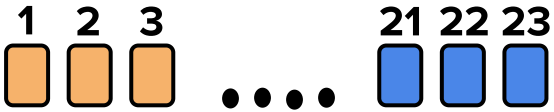


Compacted bucket layout (hourly)
Single S3 bucket with prefix containing event_name, entropy, ds, hour



Every 3 hours different prefix (Entropy)

HMS entries: Starts with all raw and as compacted, atomic swap to compacted

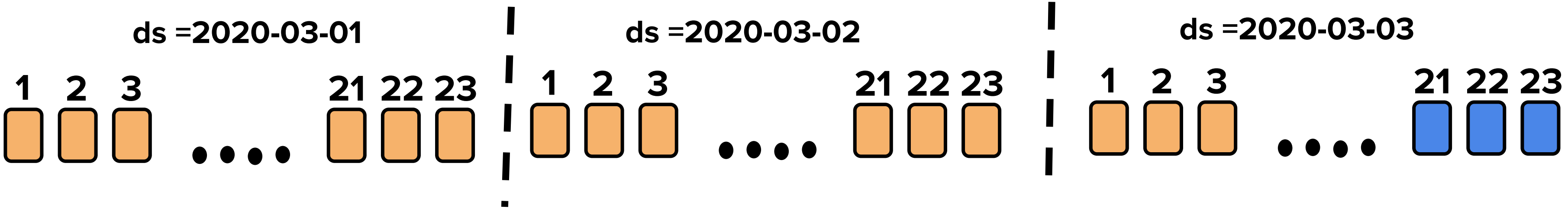


Each partition locations like s3://<prefix>/<event_name>/ds=<date>/hr=<hour> has a SUCCESS file indicating data completeness

S3 File Sensor - Downstream ETL

Provides guarantee that data is complete before triggering ETLs

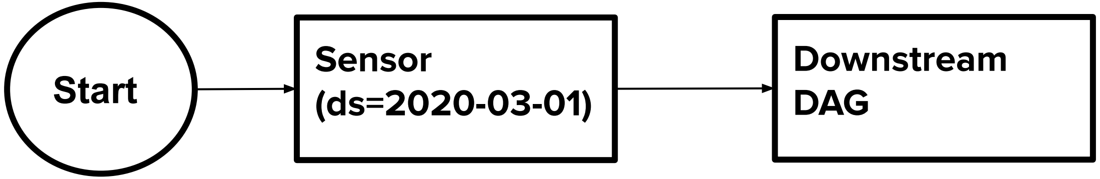
HMS entries



Each partition locations contains a SUCCESS file

- **Sensor - Take date and time (Optional) and determines if SUCCESS files exists**
- **Provides option to check for compacted partitions**
- **Uses S3 APIs**

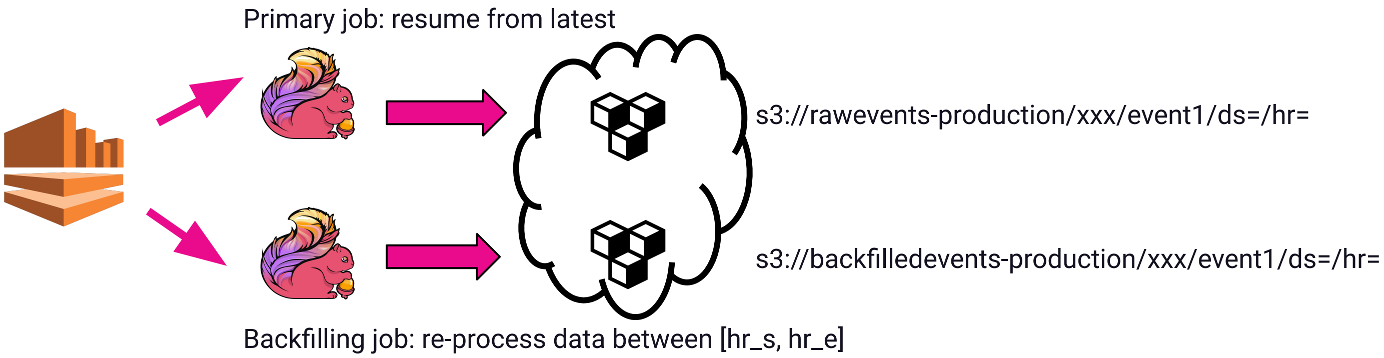
Example



Compacted Partitions **Flink Partitions (without de-dup)**

Data backfilling: coping with failures and outages: Flink

Flink can recover from short-term system failures. What about long outages ?



- Primary and backfilling Flink jobs running in parallel
- Event-time driven partitioning: backfilling process idempotent with stream processing

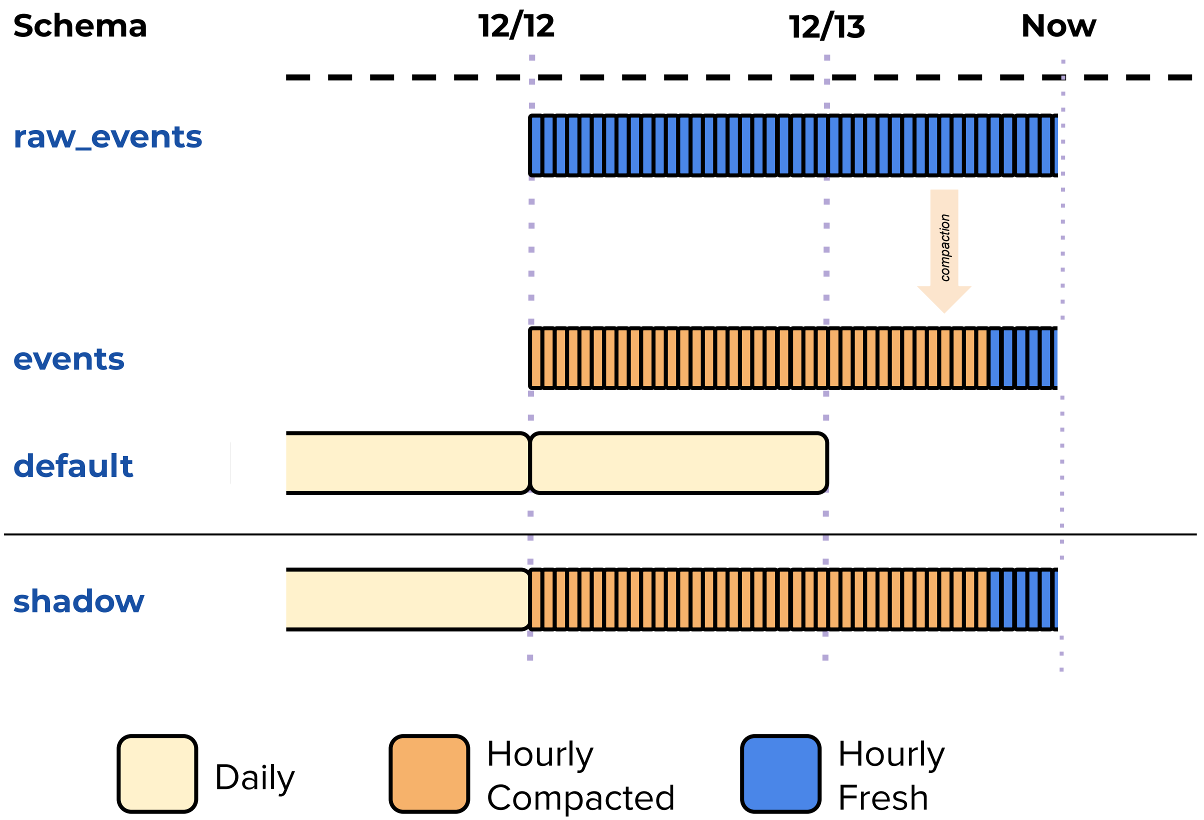
Data backfilling: coping with failures and outages - ETL

- Apache Airflow - Idempotent ETL scheduler
- Atomic compaction and HMS operations
- Automated metrics to detect partition gaps and data gaps to trigger backfills
- Schema stitching to hide the complexity of the data backfills and etl operations

Challenges for Migration

- Partition Spec changes (From daily partitions to hourly partitions)
  - **Cost to backfill historical data is high**
- Lots of existing downstream DAGs dependencies
  - Airflow as well as hive metastore
- Validation challenges
  - Data Boundaries (based of date) is going to be different

Design of Migrating Schemas



- In near real-time, events are made available as small parquet files (with potential duplicate events) in the **raw_events** schema.
- Compaction of **raw_events** runs every hour, which includes removing duplicates.
- The **events** schema reflects near real-time events, stitching compacted events with the latest raw events, and swapping out partition pointers as raw events become compacted.
- Older jobs consume JSON events and create a new partition for each day's events in the **default** schema.
- The **shadow** schema, partitioned by ds and hr, abstracts away the stitching of default, compacted, and raw events, letting data in the default schema fill in the historical data (at hr=0).

Migrating Existing workload (Run every day)

Current Pipeline

Events in JSON → Airflow DAG → Events in Parquet → Other Downstream DAG

New Pipeline

Coordination between DAGs using HMS and Airflow Task Sensor

Events in Parquet → Airflow DAG using Sensor → Other Downstream DAG

Maintaining same Airflow DAGs help in keeping contracts with downstream DAGs same

Summary and future directions

Experience and Lessons Learned

- Flink persisting Parquet in near real-time unlocks interactive query experiences
- Flink full restart or job deployment could affect SLO
- S3 file layout is critical to consistency and performance
- Backward compatible schema evolution is critical to data quality
- Migration of tables with different partition granularity requires careful design

Future Directions

- Flink job operating in k8s environment
- Event driven smart compaction
- Generalized streaming persistence framework
- Storage management improvement
- Query optimization

Teamwork and Acknowledgements

- Arup Malekar
- Chris Williams
- Dev Tagare
- Jamie Grier
- Jason Carey
- Lakshmi Gururaj Rao
- Li Gao
- Mark Grover
- Sherin Thomas
- Thomas Weise
- Valentine Lin
- Yash Kumaraswamy

We are hiring! **lyft.com/careers**

Thank you

04/22/2020