1 Data Ingestion - Stats

Kafka Stats

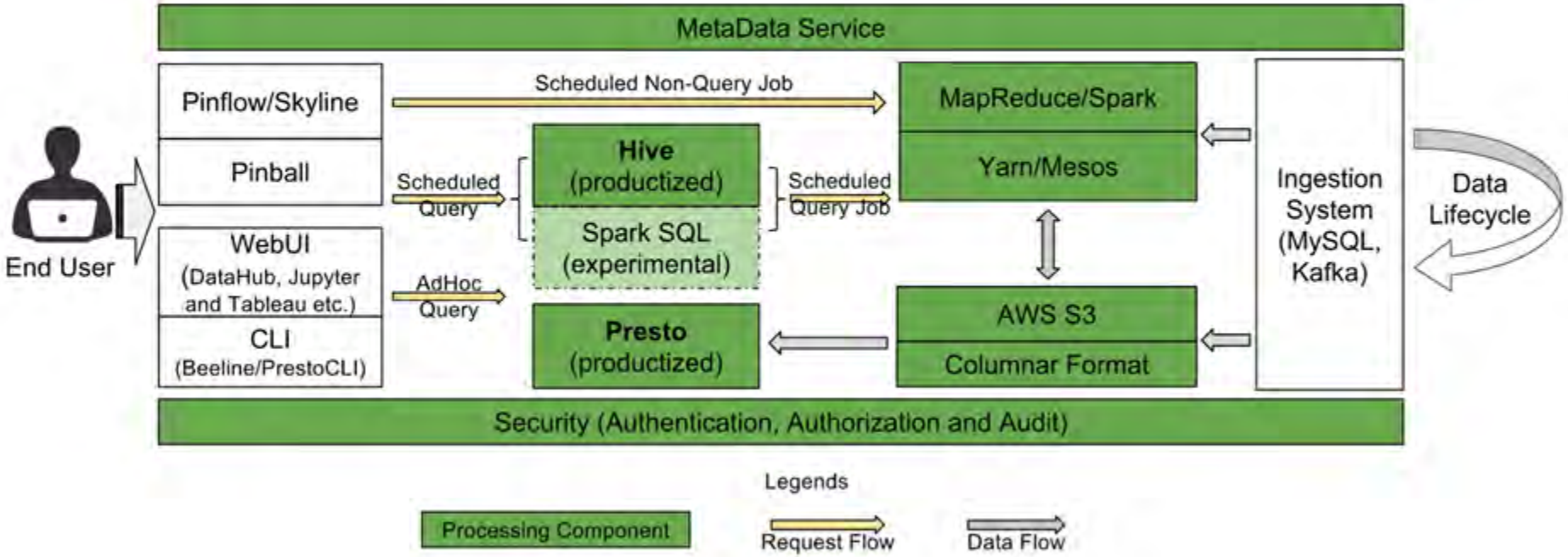
- ~150B Messages/Day
- ~400T Bytes/Day
- ~1000 Kafka Topics
- ~1400 Kafka Brokers

DB Dump Stats

- ~80 Table/Day
- ~100T Bytes/Day

Doctor Kafka https://github.com/pinterest/doctorkafka

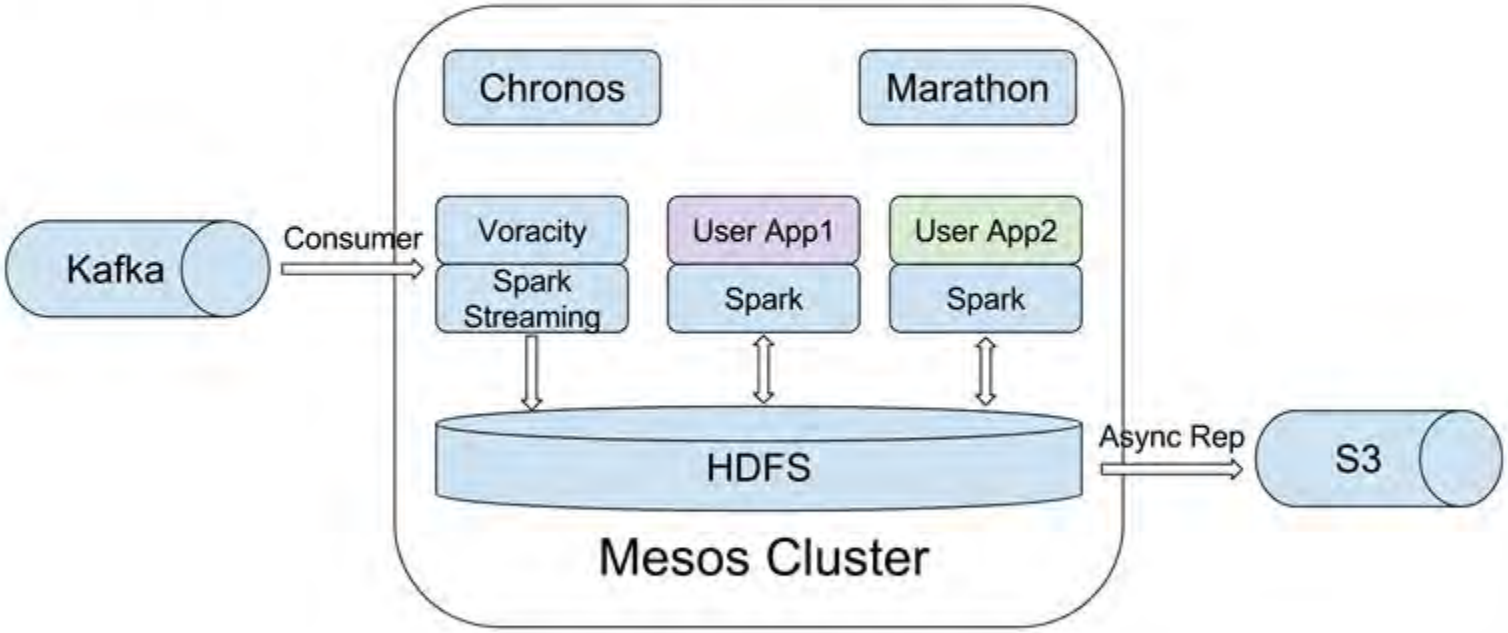
Data Processing Overview



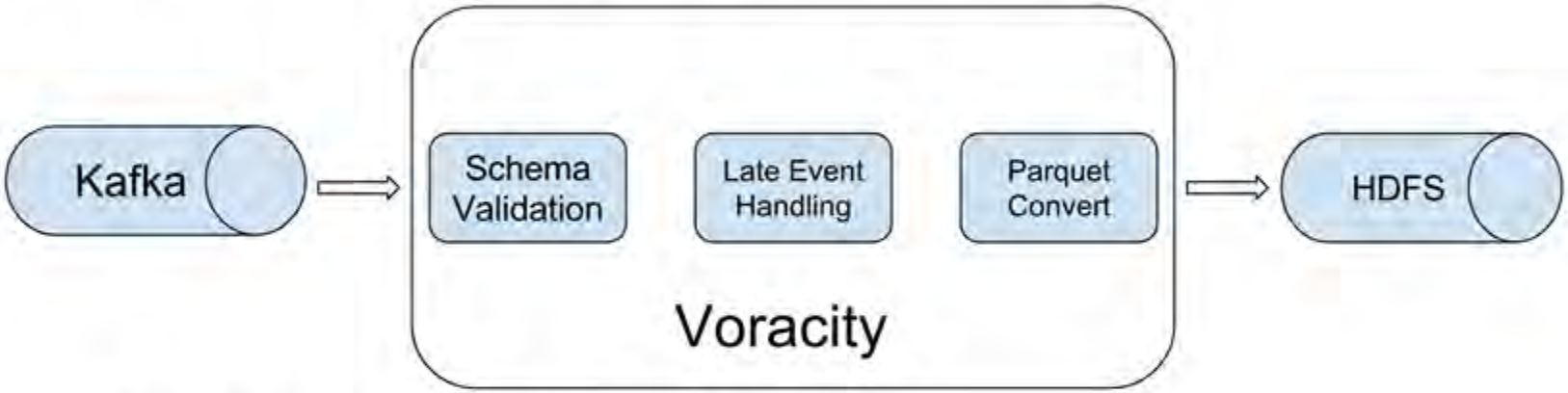
Agenda

1. Data Ingestion
2. **Streaming**
3. Hive
4. Presto
5. Workflow

2 Streaming - Overview



2 Streaming - Voracity

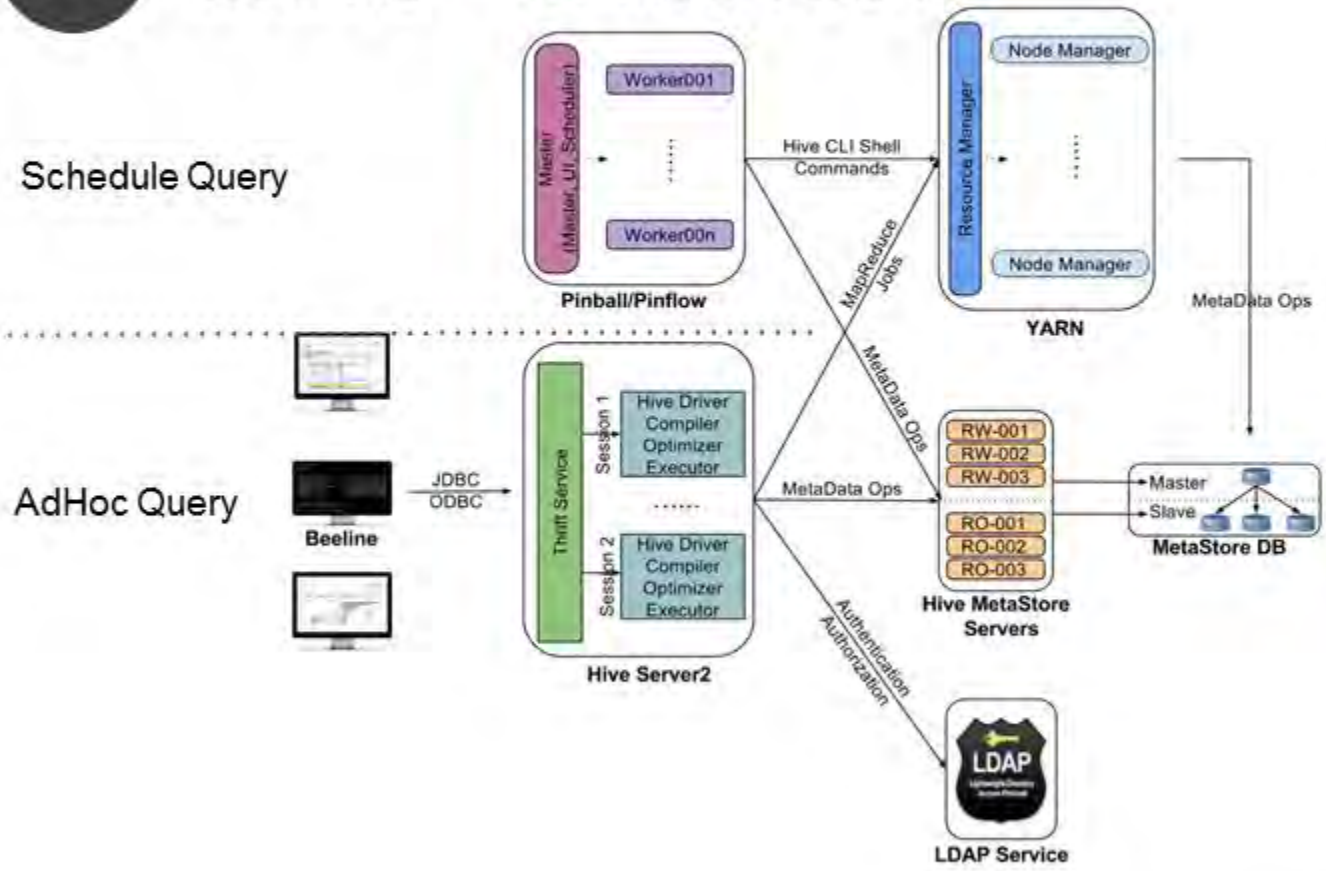


- Schema Validation
  - Filter out malformed records
- Late Arrival Event Handling
  - Move late event to proper bucket
- Parquet Conversion
  - Json/Thrift -> Parquet

Agenda

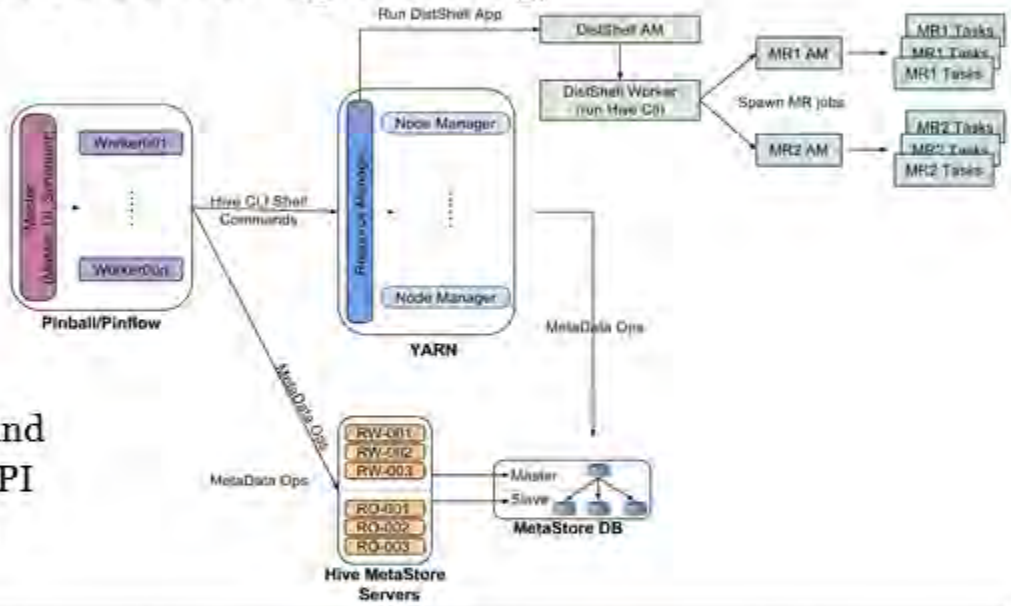
1. Data Ingestion
2. Streaming
3. **Hive**
4. Presto
5. Workflow

3 Hive - Overview



3 Hive - Scheduled Query

- Query is submitted by System
- Based on Hive CLI
- Talking to MetaStore (MySQL)
- YARN REST API
  - DistShell: AppMaster + Worker
    - Hive CLI as pure shell command
  - Get stdout/stderr from NM REST API
  - Rewrite select query to insert to s3
    - Container log is not stable



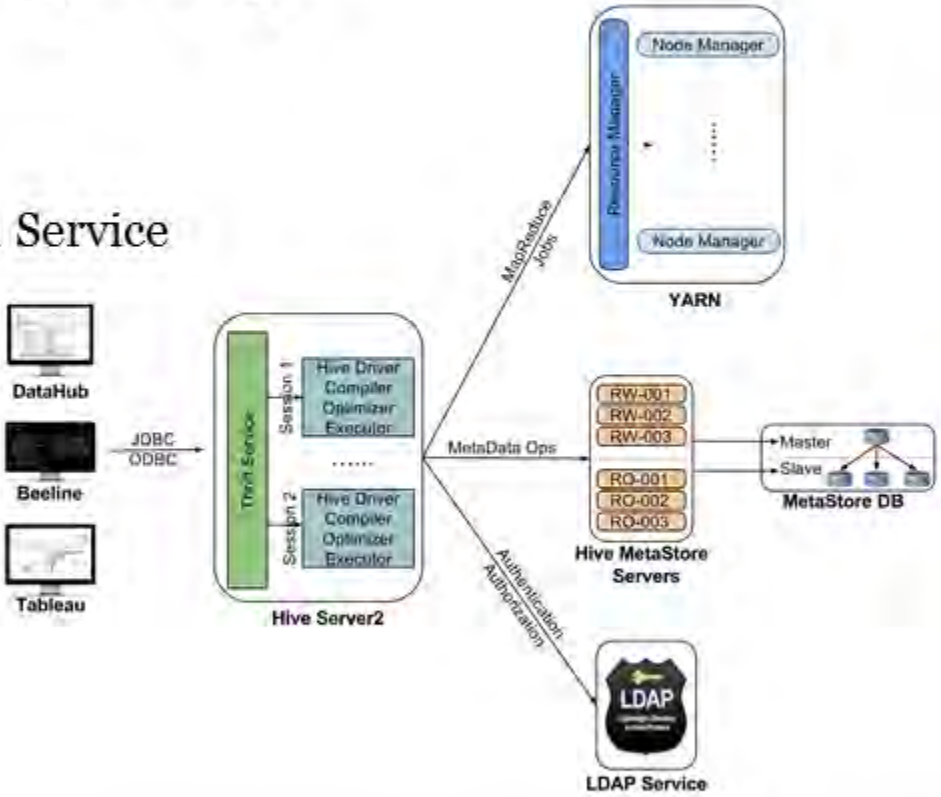

```
SELECT country, count(*)
FROM db_users
GROUP BY country
```

Rewrite

```
INSERT OVERWRITE DIRECTORY 's3://hive_query_output/cluster_name/20171019/application_1508263025827_2301/0'
 ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '\001'
   COLLECTION ITEMS TERMINATED BY '\002'
   MAP KEYS TERMINATED BY '\003'
   LINES TERMINATED BY '\n'
 STORED AS TEXTFILE
SELECT country, count(*)
FROM db_users
GROUP BY country
```

3 Hive - AdHoc Query

- Query is Submitted by Human
  - Through beeline/tabeau etc
- HiveServer2 is the Query Submission Service
  - via JDBC/ODBC
  - Zookeeper based HA
- Talking to Hive MetaStore Server
  - HiveServer2 is a shared resource
- LDAP for Authentication
  - Impersonation for 3rd party tool
- Hive hook to Audit all queries



3 Hive - Cloud Integration

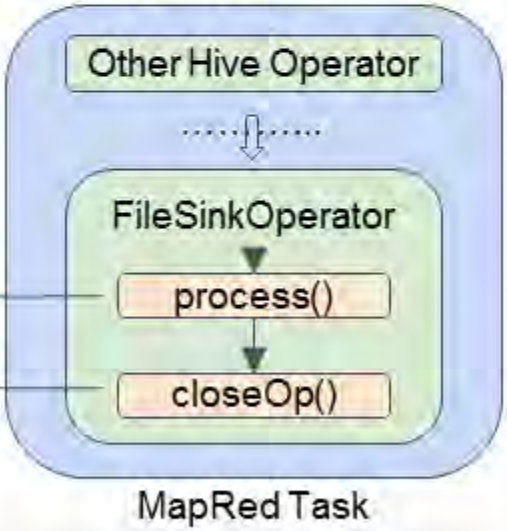
FileSinkOperator - Where S3 doesn't fit

- Kind of "2-phase commit"
- Phase 1 - each task attempt will write to a unique tmp output path

hdfs://${staging_dir}_${**query_id**}/

_task_tmp.-ext-${**stage_id**}/_tmp.${**task_id**}_${**attpt_id**}

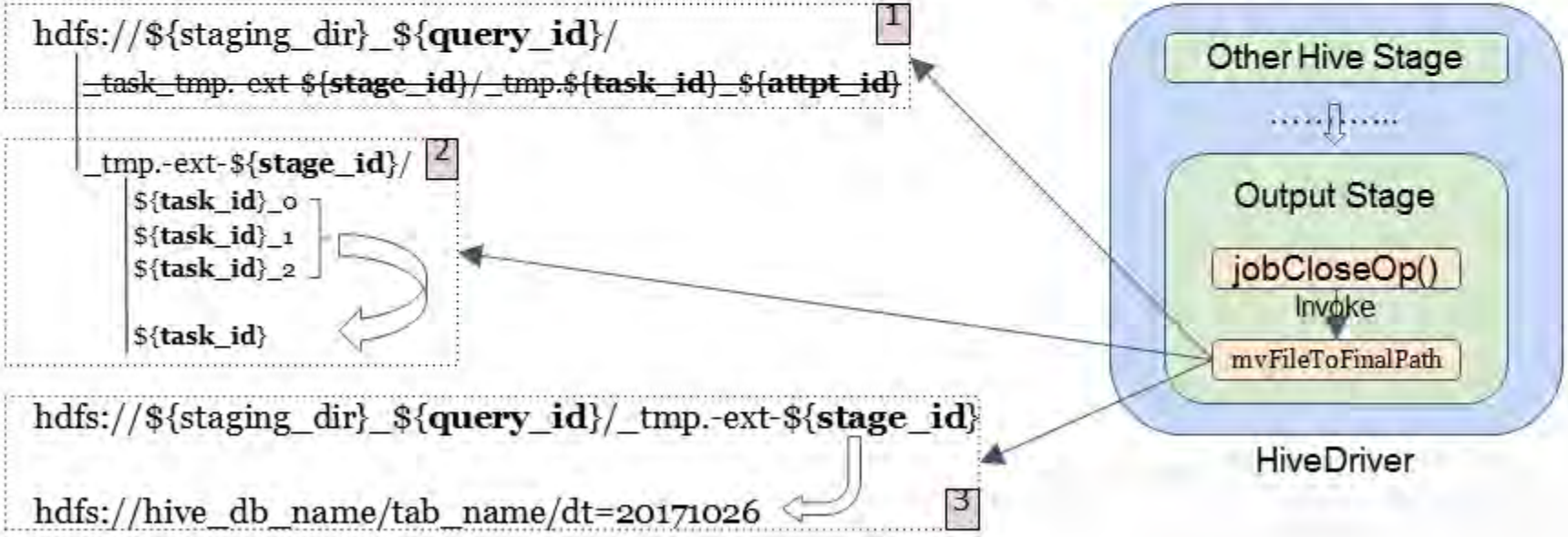
_tmp.-ext-${**stage_id**}/${**task_id**}_${**attempt_id**}



3 Hive - Cloud Integration

FileSinkOperator - Where S3 doesn't fit

- Kind of "2-phase commit" protocol
- Phase 2 - stage controller (HiveDriver) rename query temp to final path



3 Hive - Cloud Integration

FileSinkOperator - Where S3 doesn't fit

- Fast & Atomic file/dir **rename** operation is the key
- Unfortunately, rename on s3 is copy & delete

3 Hive - Direct Committer @ S3

Direct Output Committer

- Each task attempt writes to final output directly
  - {target folder}/{task_id}

- Consequences
  - Query Level Failure -> partial output in target folder
    - Cleanup target folder in the very beginning of each query
    - Disable INSERT INTO (using INSERT OVERWRITE Q1 UNION ALL Q2)
  - Task Level Failure -> FileAlreadyExistsException
    - File open mode CREATE -> OVERWRITE (ORC & Parquet)
  - Speculative Execution -> corrupted task output
    - Cause: outWriter.close() in killed task attempt
      - S3A write to local disk first and then upload to s3 in close()
    - Solution: Disable speculative execution

3 Hive - Speculative Exec @ S3

Direct Committer with Speculative Execution

- S3A Staging Committer (originated from Netflix)
  - Based on S3 multipart uploading[1]
  - Decouple upload & commit, simulate write to tmp & rename to final
  - Being actively worked in Apache: HADOOP-13786

- S3A Hive Committer (originated from Pinterest, on-going)
  - Ignore the MapReduce commit protocol
  - Allow different task attempts to commit at the same time
  - Stage controller (HiveDriver) dedup multiple attempts for the same task

3 Hive - Dynamic Partition @ S3

New Challenges

- Target folder path is only known at runtime
- Race condition among multiple FinkSinkOperators
- Discover new dynamic partitions to update Metastore

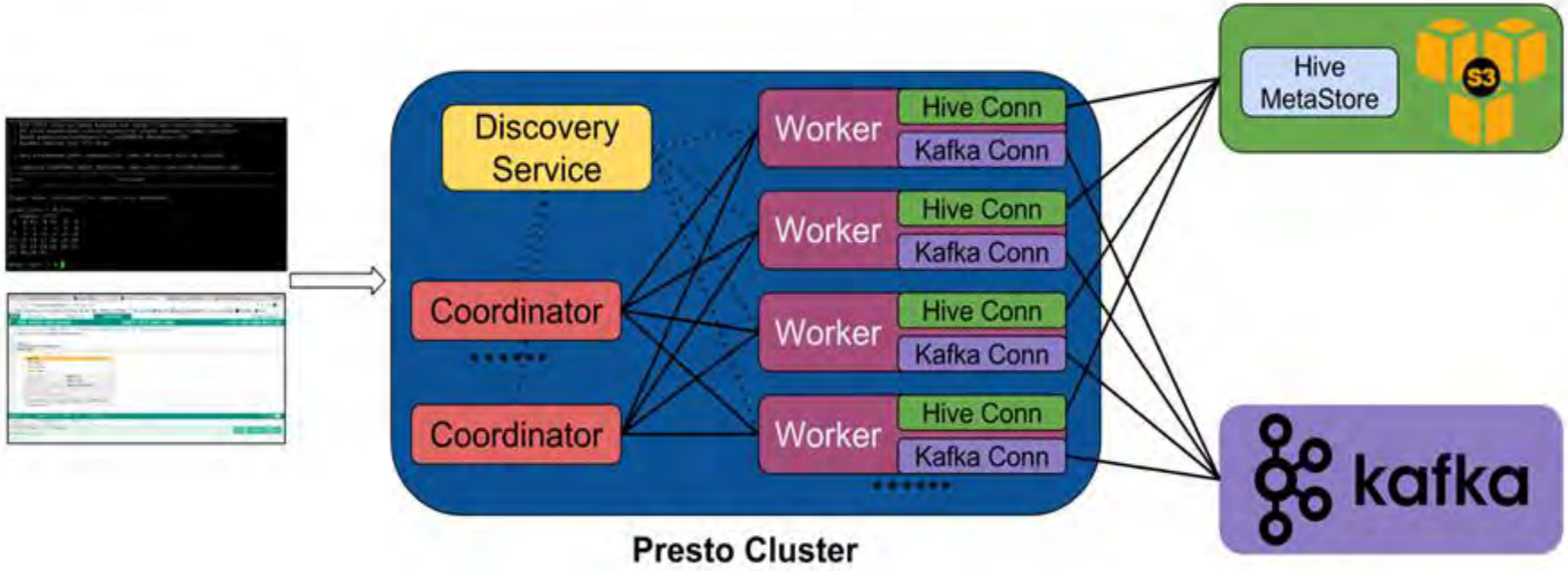
Solutions

- Encode the ts into output file name: {target_part_folder}/{dpCtx.queryStartTs}_{task_id}
- New dynamic partition: folders that
  - Modification time > query start time
  - Contains files starting with query_start_ts

Agenda

1. Data Ingestion
2. Streaming
3. Hive
4. **Presto**
5. Workflow

4 Presto - Overview



4 Presto - Cluster & Stats

Cluster Management

- Instance type: r3.8xl
- Computing only, no local data
- Using AWS ASG
- Managed by Terraform

Presto Stats

- ~800 users
- ~10k queries daily
- 5X ~ 50X faster than Hive
- AdHoc (250 node): P90: 4min, P99: 30min
- App (150 node): P90: 2min, P99: 20min

4 Presto - Feature Challenges

Concurrency Bug in ObjectInspector library

- Hive runtime is for single thread env
- Presto is highly concurrent and shared

Thrift Table Support

- Presto Hive MetaStore client doesn't support "dynamic" table
- Deeply nested table will hang in planning phase

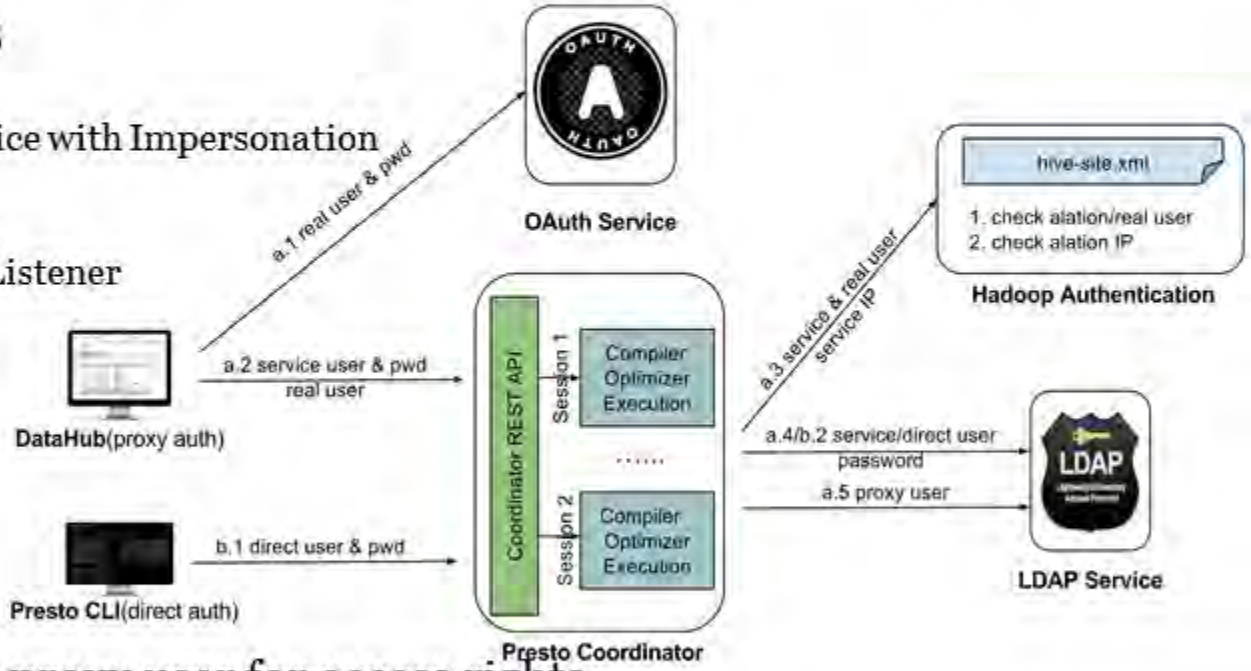
Schema Evolution Support

- Historical partition schema mismatch
- Small int -> big int
- Parquet table column rename

4 Presto - Security Enhancement

Security Features

- Authentication
  - Integrated middle service with Impersonation
- Audit
  - Every Query is Logged
  - Plugin based on EventListener



Access Control

- Check LDAP group for proxy user for access rights
- Data Level Access is controlled by IAM role

4 Presto - Cluster Operations

Separate AdHoc & Application Cluster

- Adhoc cluster is generally less stable
- App cluster only run well known queries

Bad Query Monitor

- Detected by bot based on runtime and splits #
- Kill the query and send slack message to end user

Bad Node Monitor

- Detected by bot based on query failure due to node issue stats
  - Bad node - 3 query failed on it in last 5 min
- Specially important when the cluster is scaled up & down on a daily basis

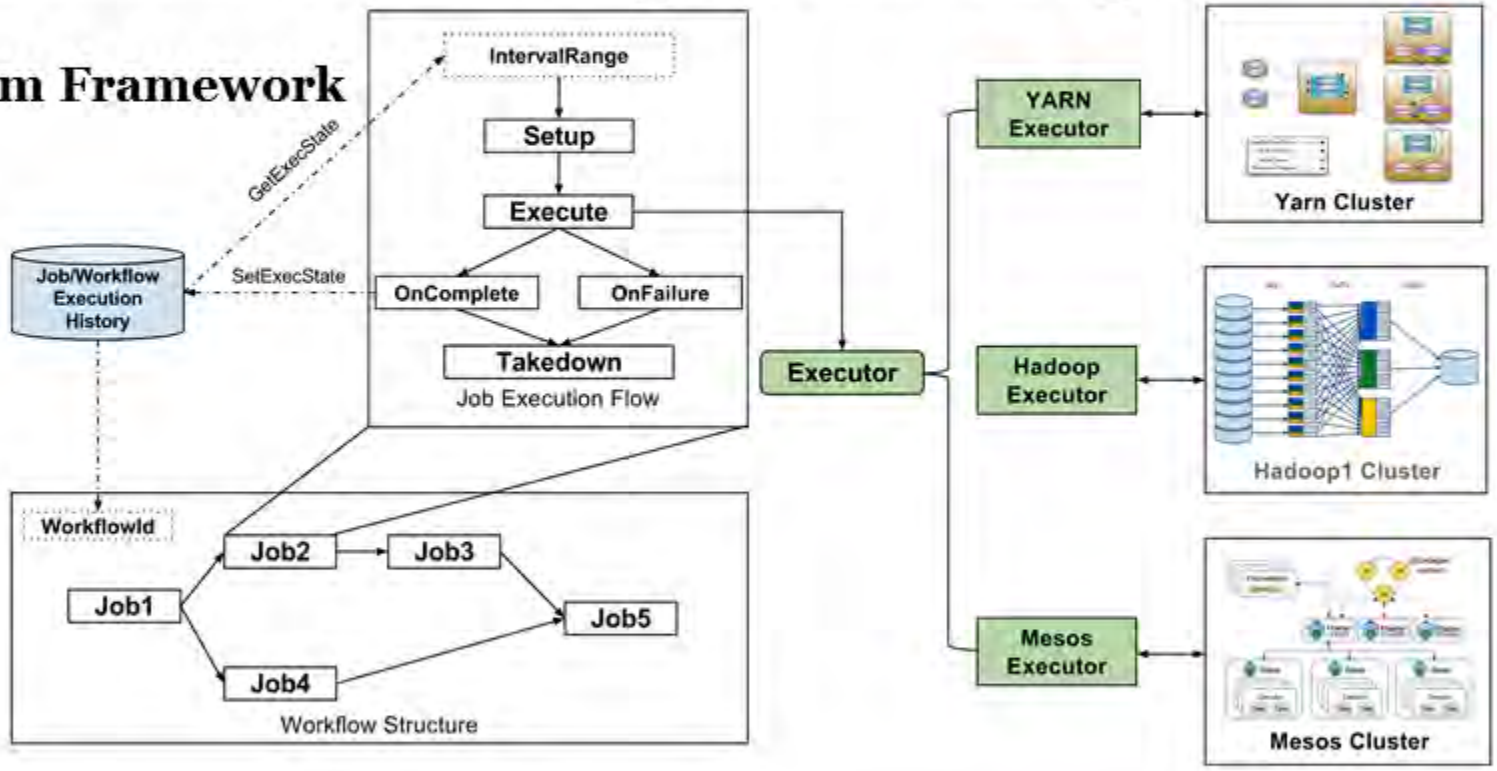
Agenda

1. Data Ingestion
2. Streaming
3. Hive
4. Presto
5. Workflow

5 Workflow - Programming

Workflow Prgm Framework

- Skyline
- DataJob
- Pinflow



5 Data Architecture - Manager

Workflow Manager

- Pinball
- https://github.com/pinterest/pinball

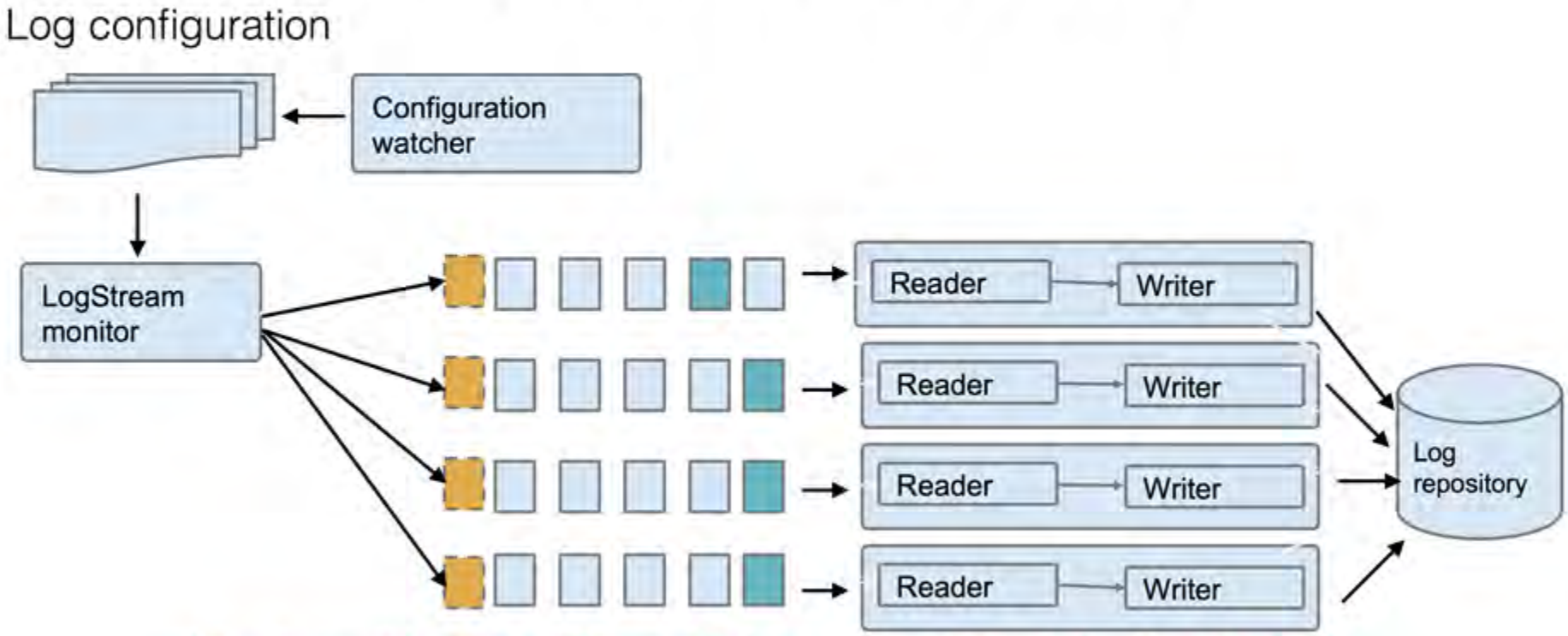
Workflow Stats

- 2000+ Workflows
- 60k+ Jobs
- ~80% Hive
- ~15% Cascading/Spark
- ~5% Python

The World's Catalog of Ideas

Discover Ideas

Singer Architecture

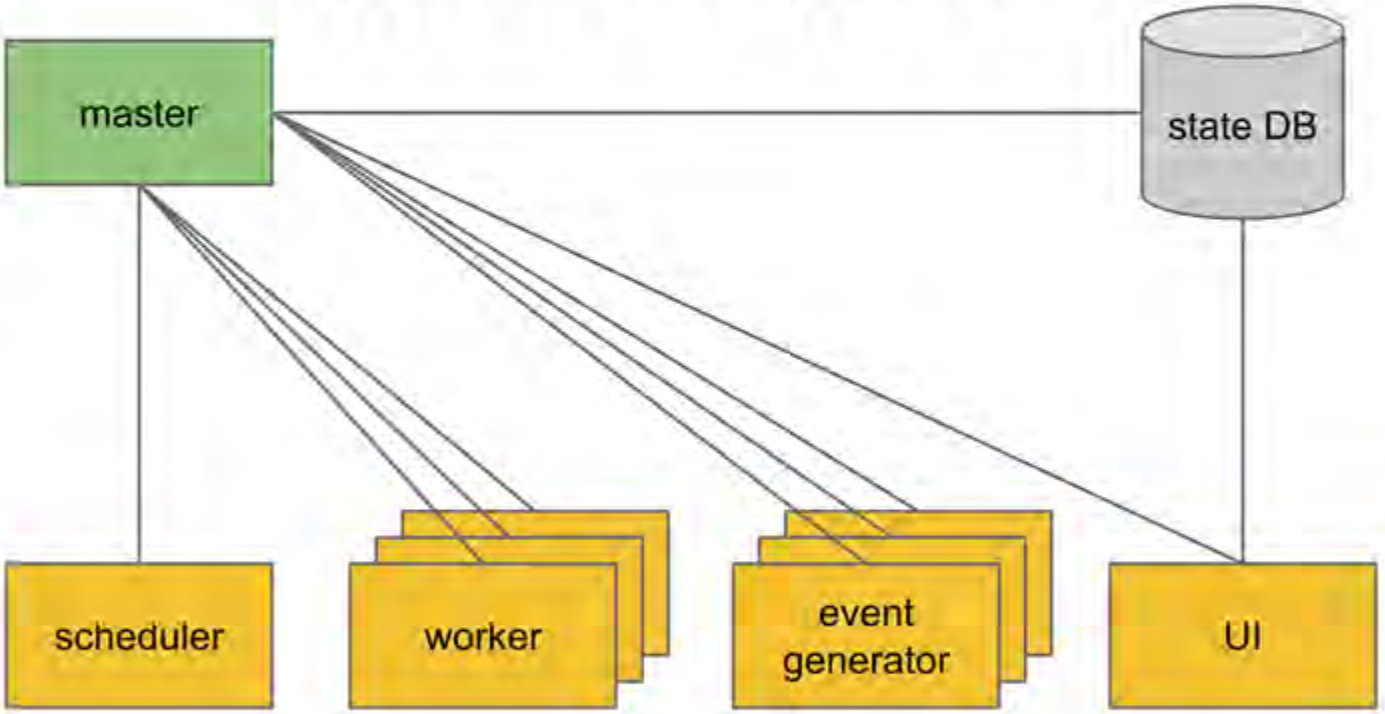


https://www.slideshare.net/DiscoverPinterest/singer-pinterests-logging-infrastructure

Merced Architecture

https://github.com/pinterest/secor

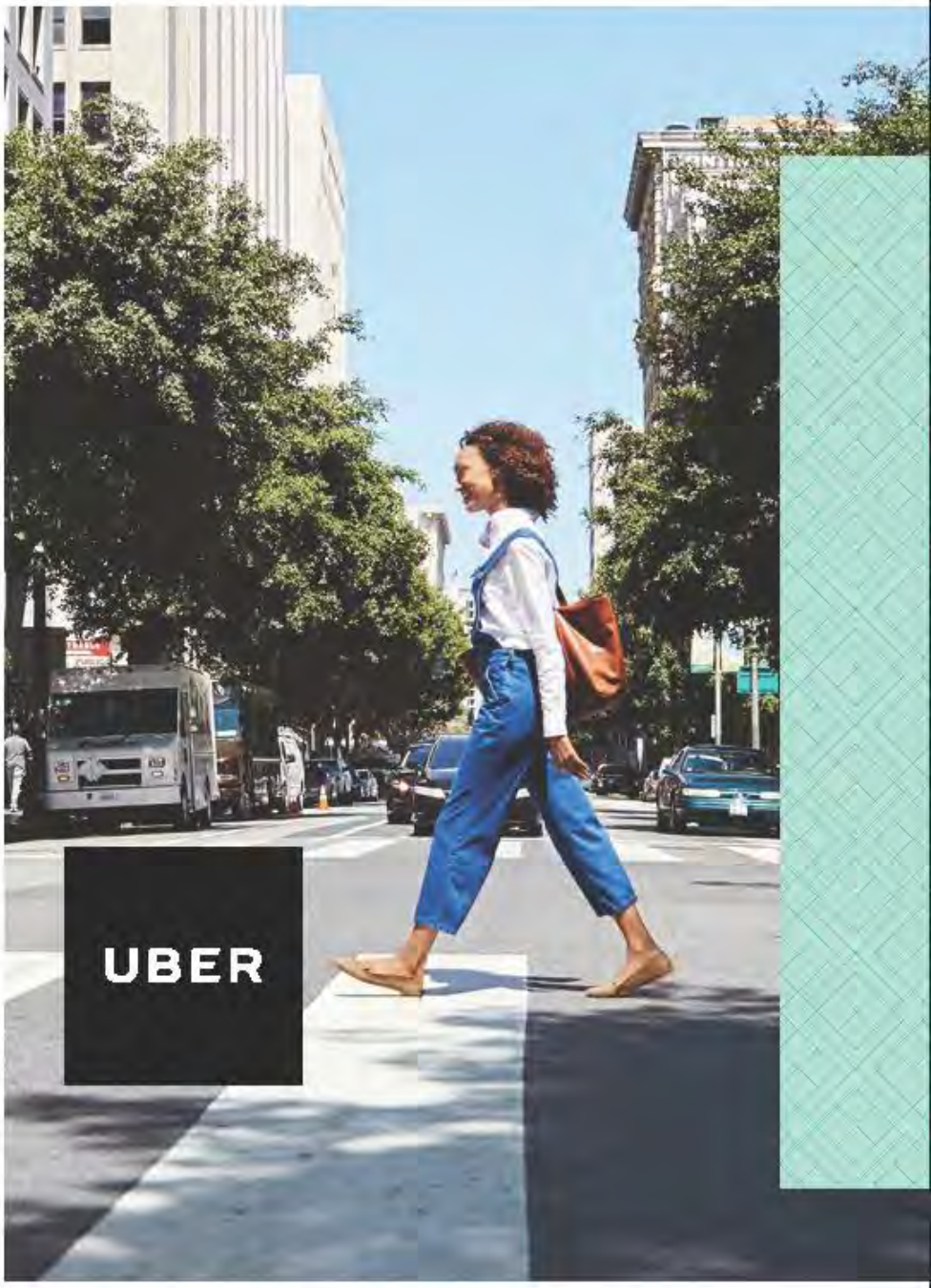
Pinball Architecture



https://github.com/pinterest/pinball

Uber实时处理系统的演化

袁泳@Uber

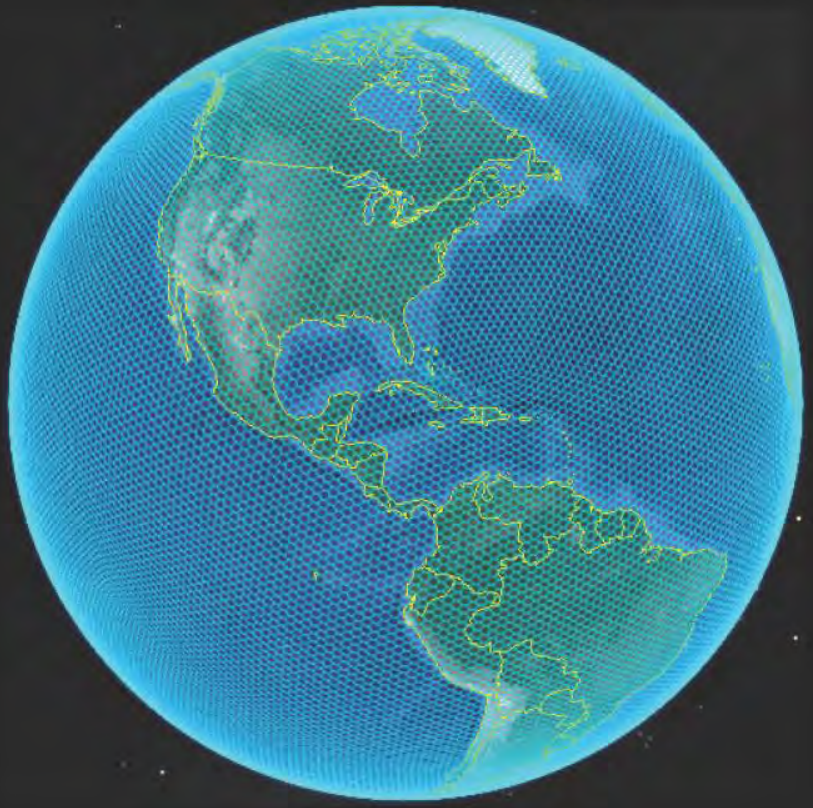


Four Kinds of Analytics

- On demand aggregation and pattern detection
- Clustering
- Forecasting
- Pattern detection on geo-temporal data

Two Ingredients

Geo/Spatial



Time

Real-time aggregation and pattern matching

Complex Event Processing

Examples

How many cars enter and exit a user defined area in past 5 minutes

CEP with full historical context

Notify me if a partner completed her 100th trip in a given area just now?

Patterns in the future

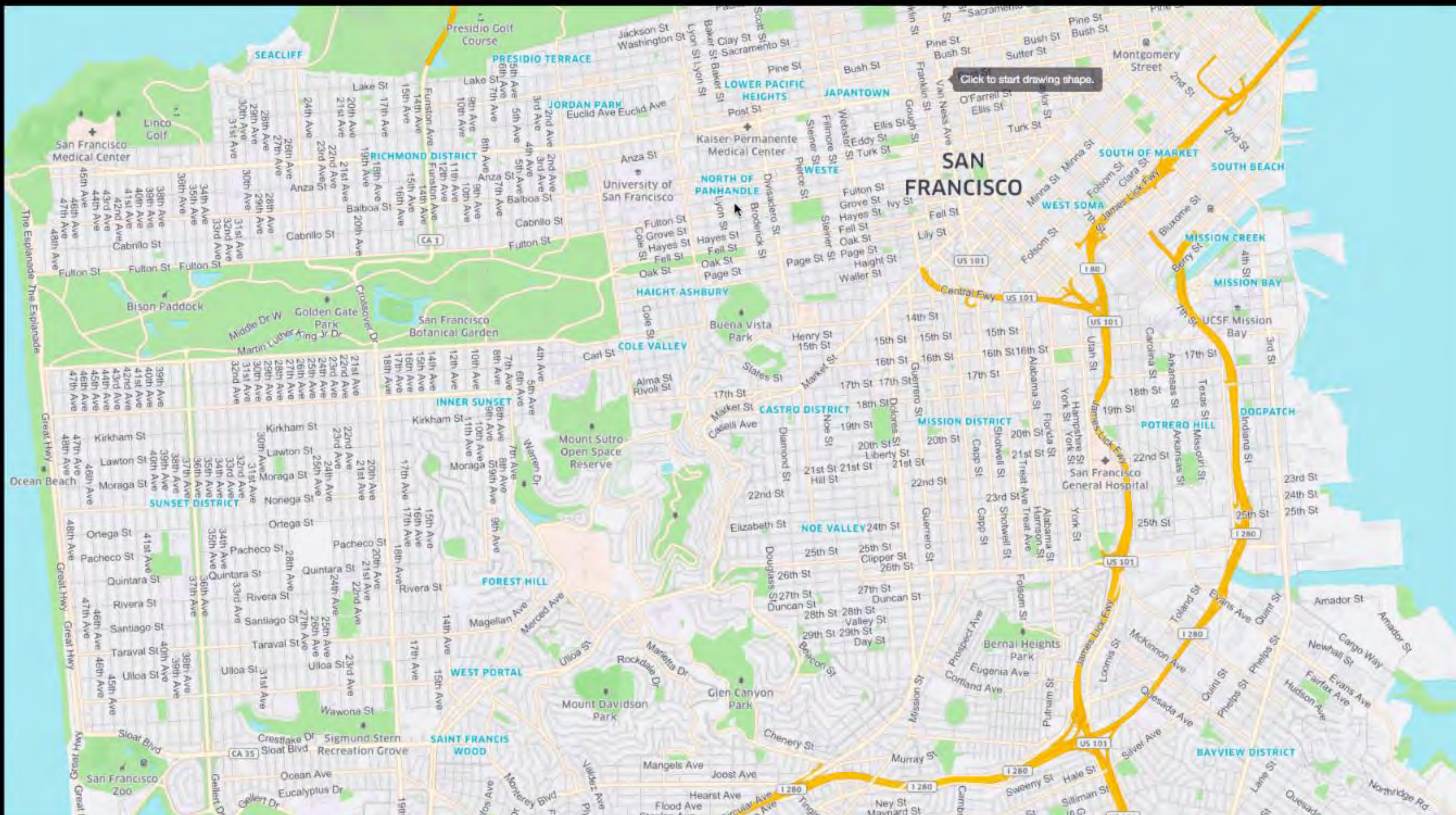
How many first-time riders will be dropped off in a given area in the next 5 minutes?

Patterns in the future

How many first-time riders will be dropped off in a given area in the next 5 minutes?

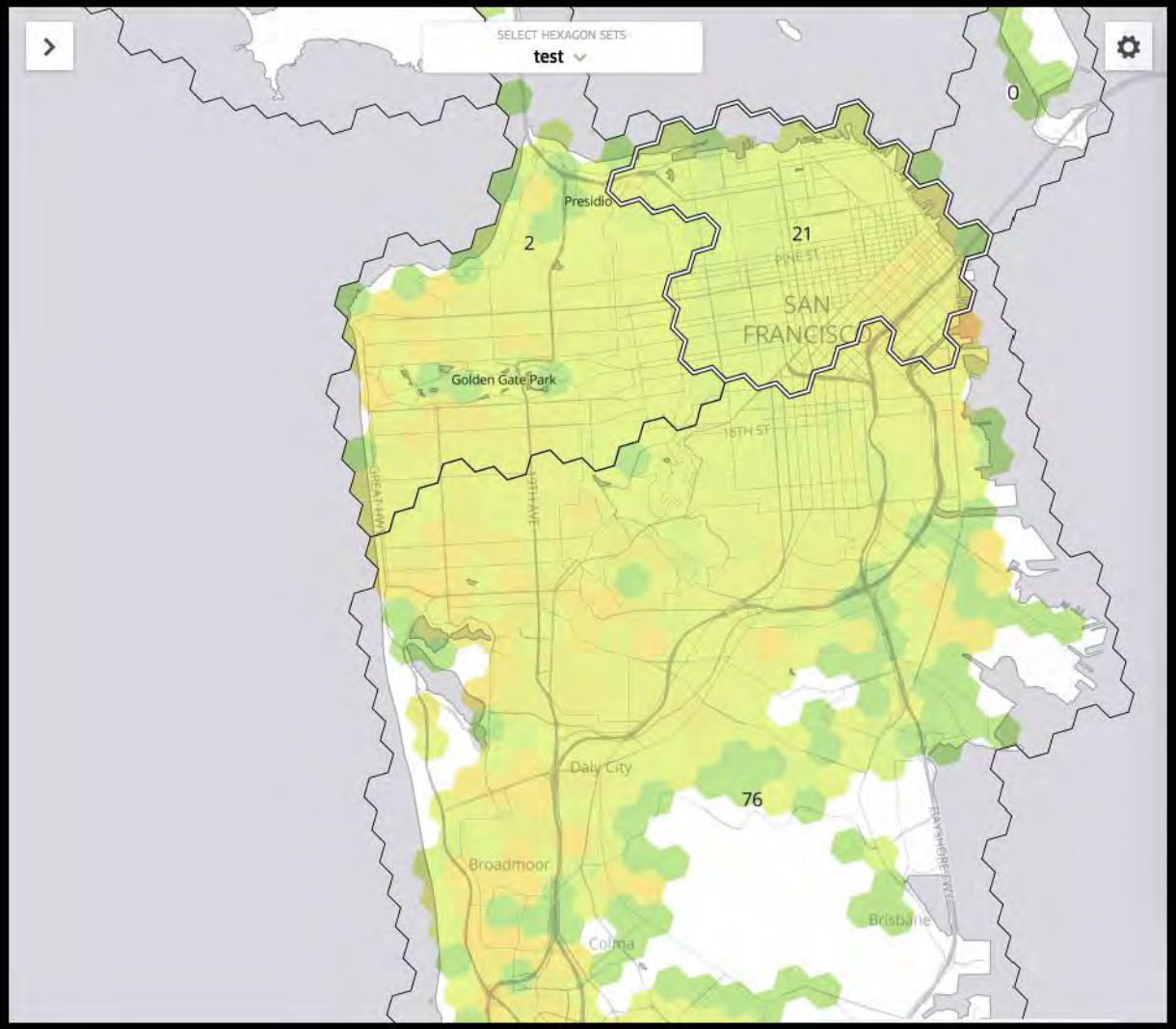
Geo: user flexibility is important

Geo: user flexibility is important

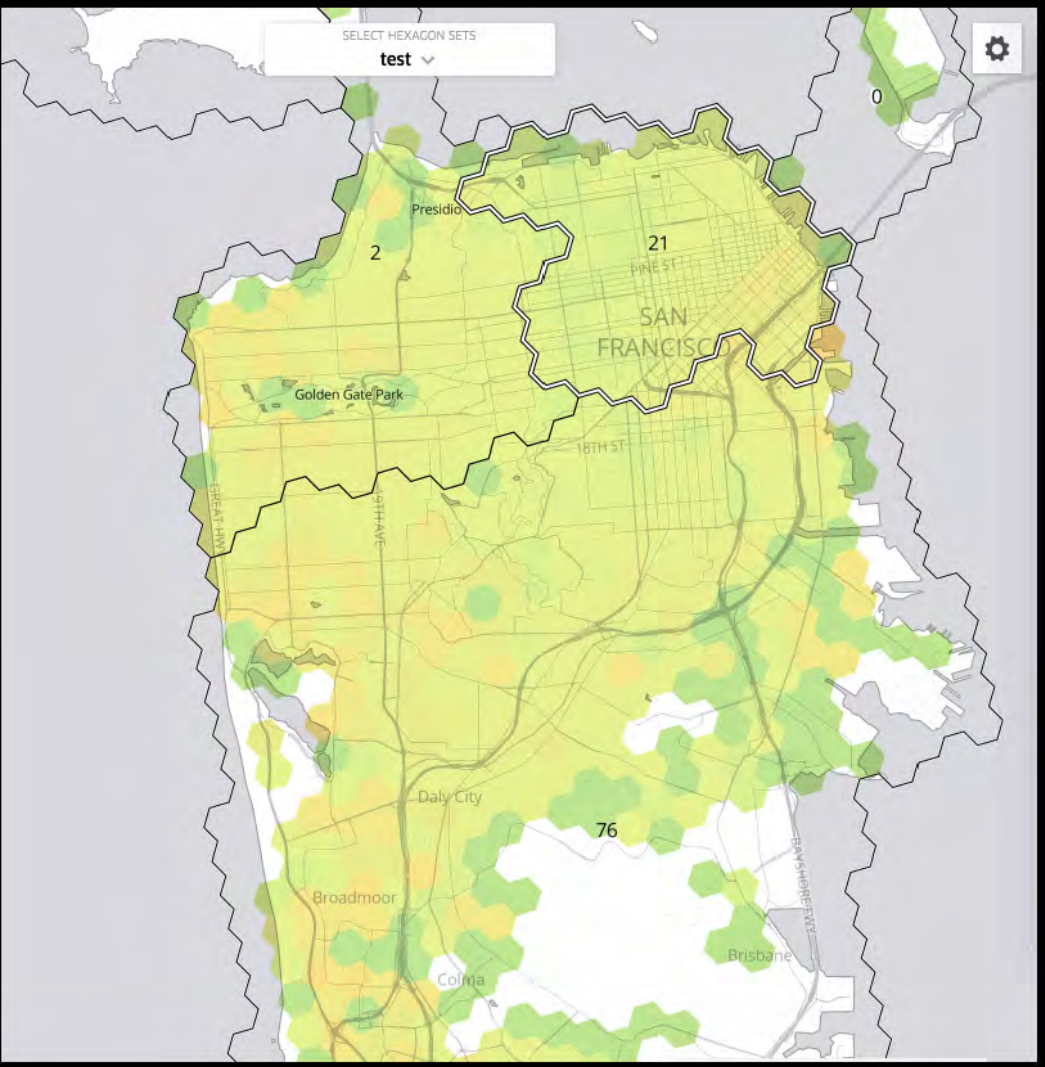


It needs to be scalable

It needs to be scalable



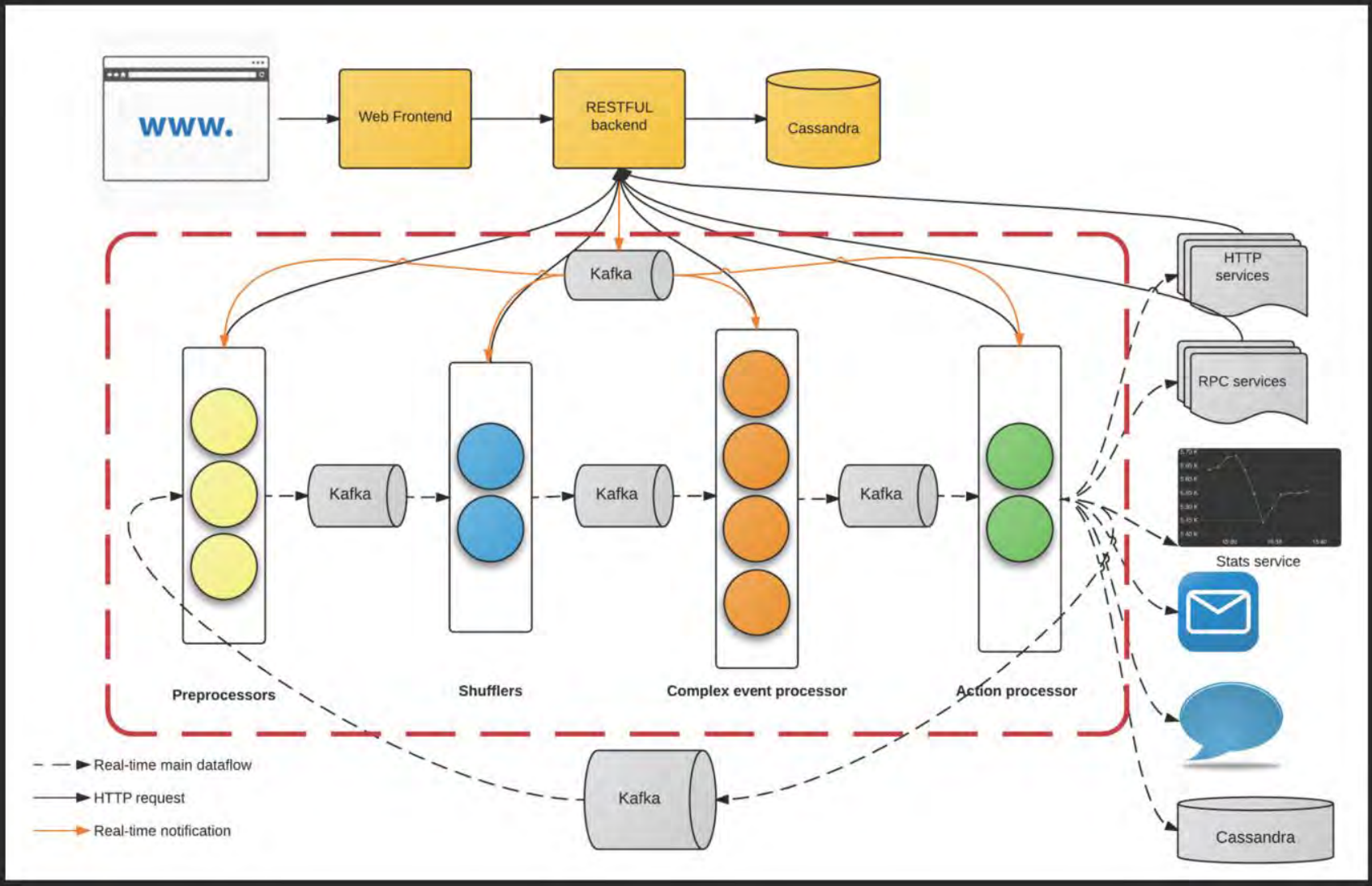
It needs to be scalable

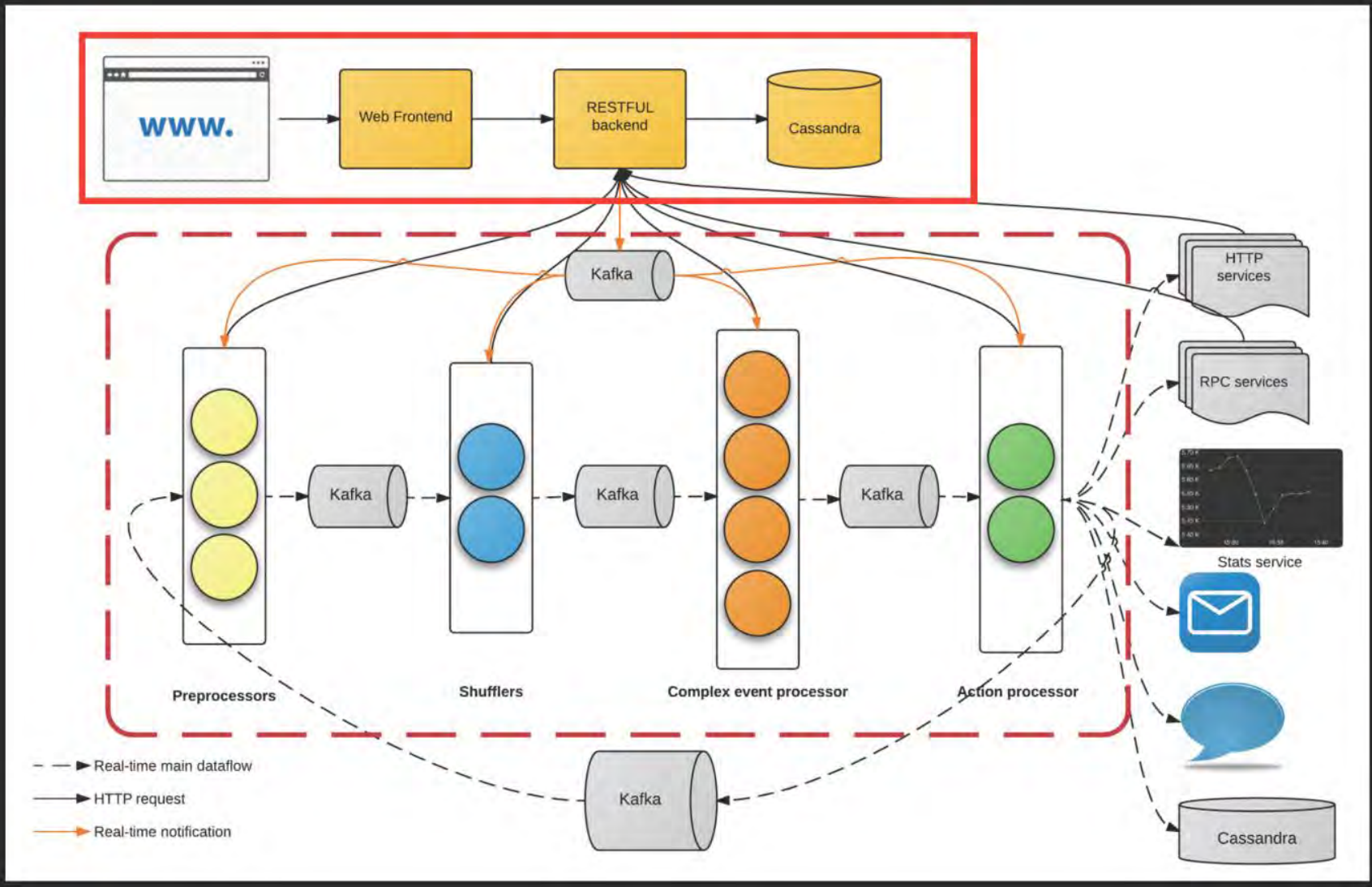


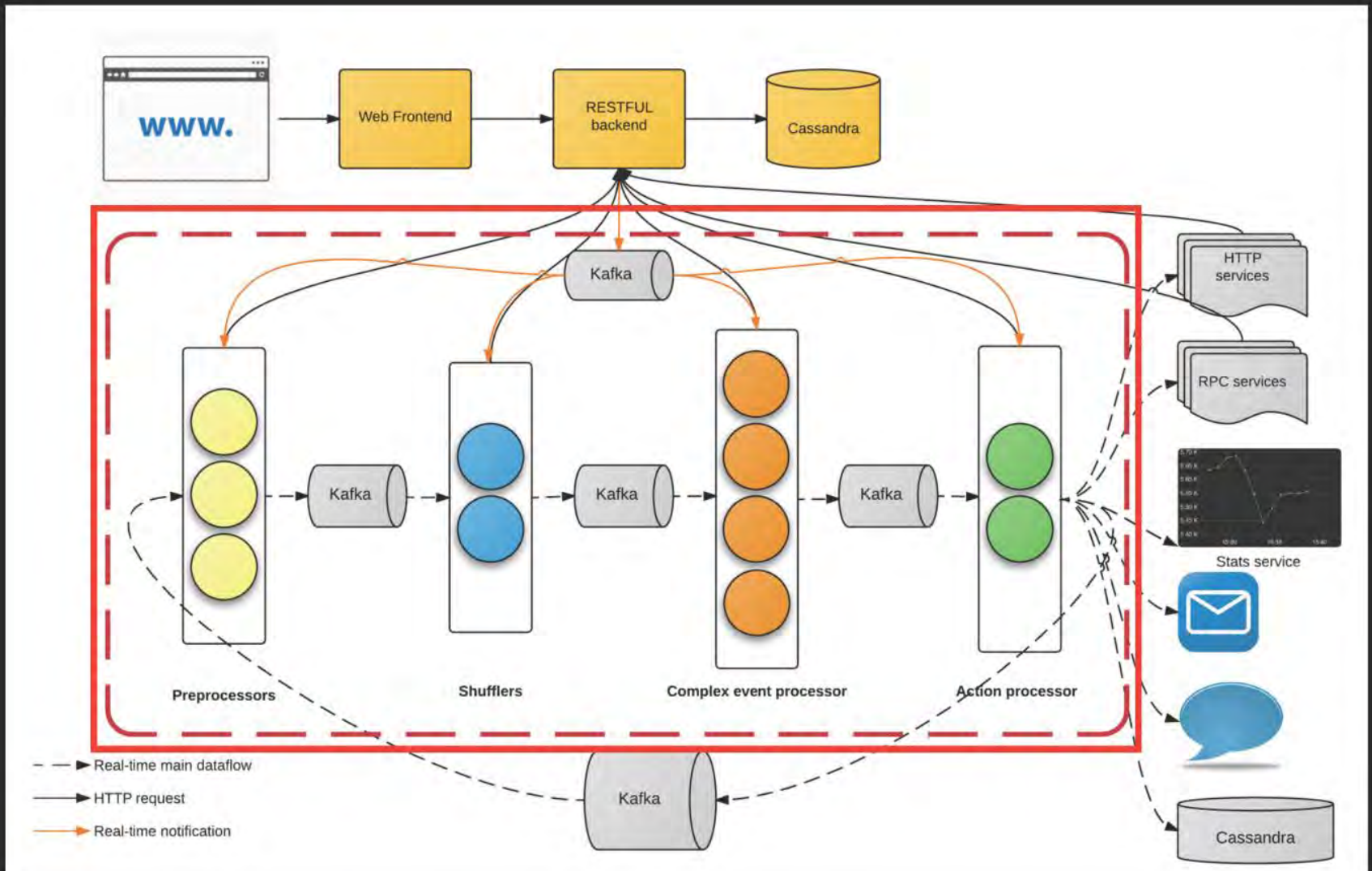
- Every hexagon
- Every driver/rider

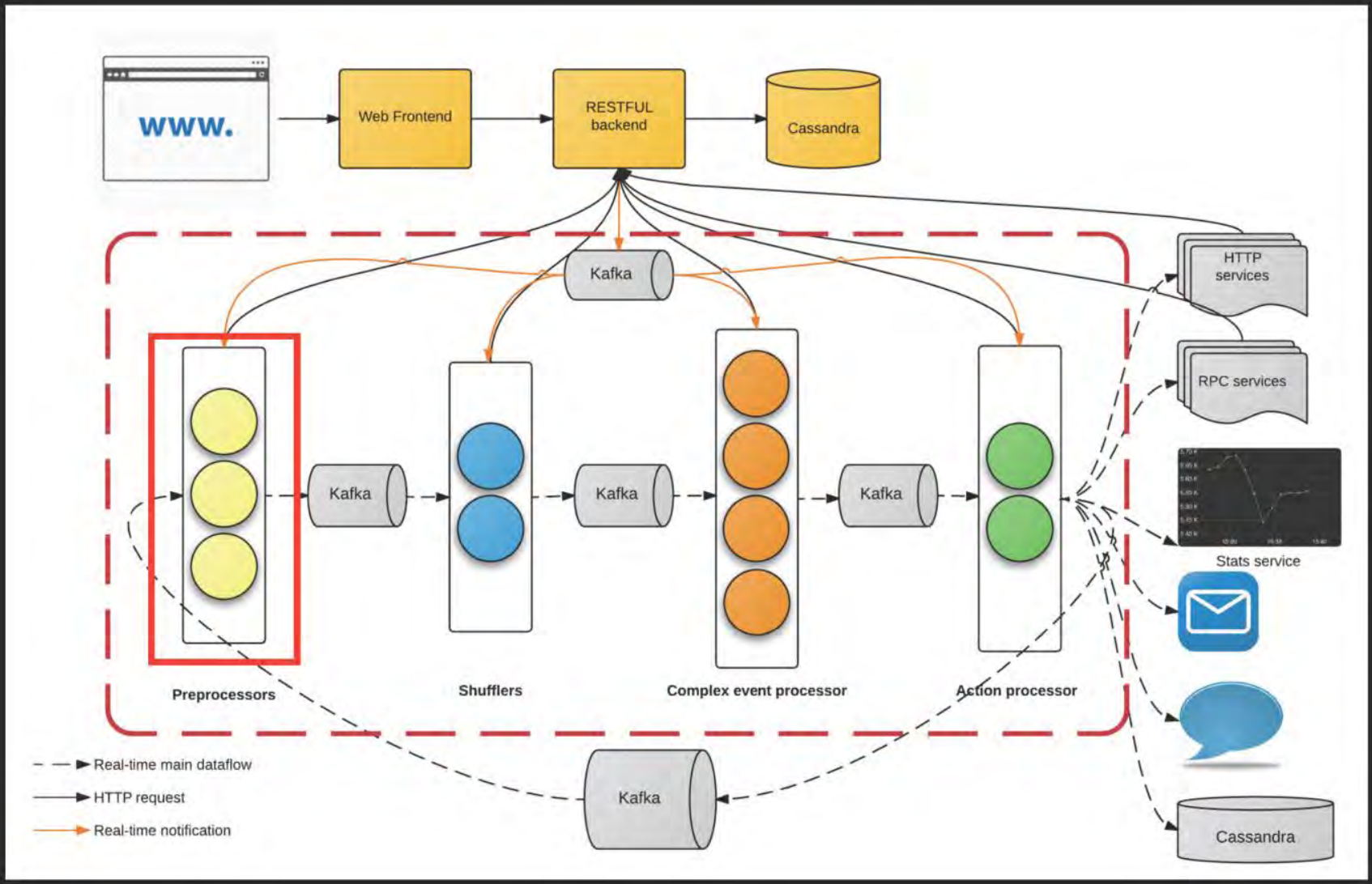
CEP Pipeline Built on Samza

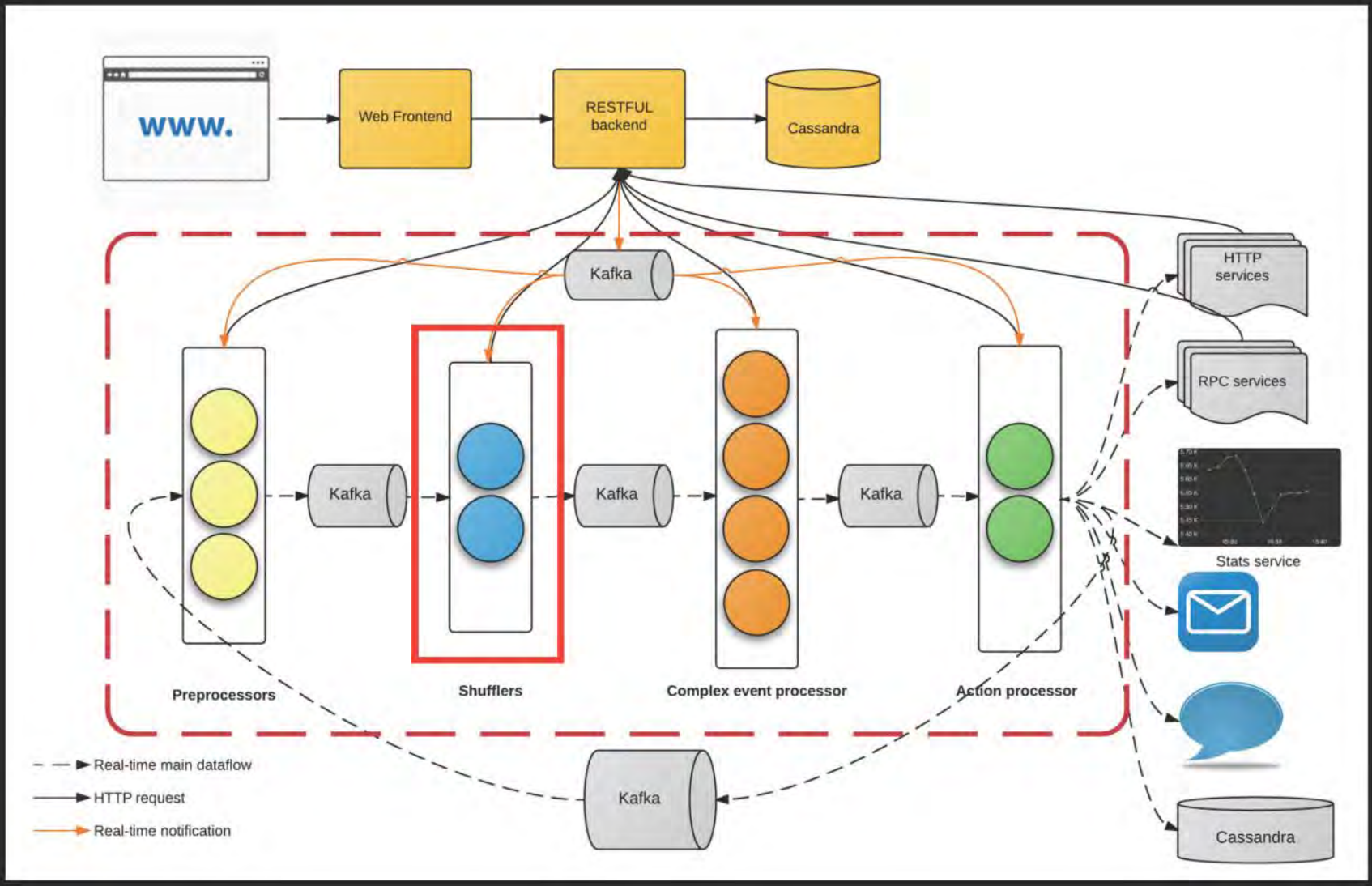
- No hard-coded CEP rules
- Applying CEP rules per individual entity: topic, driver, rider, cohorts, and etc
- Flexible checkpointing and statement management

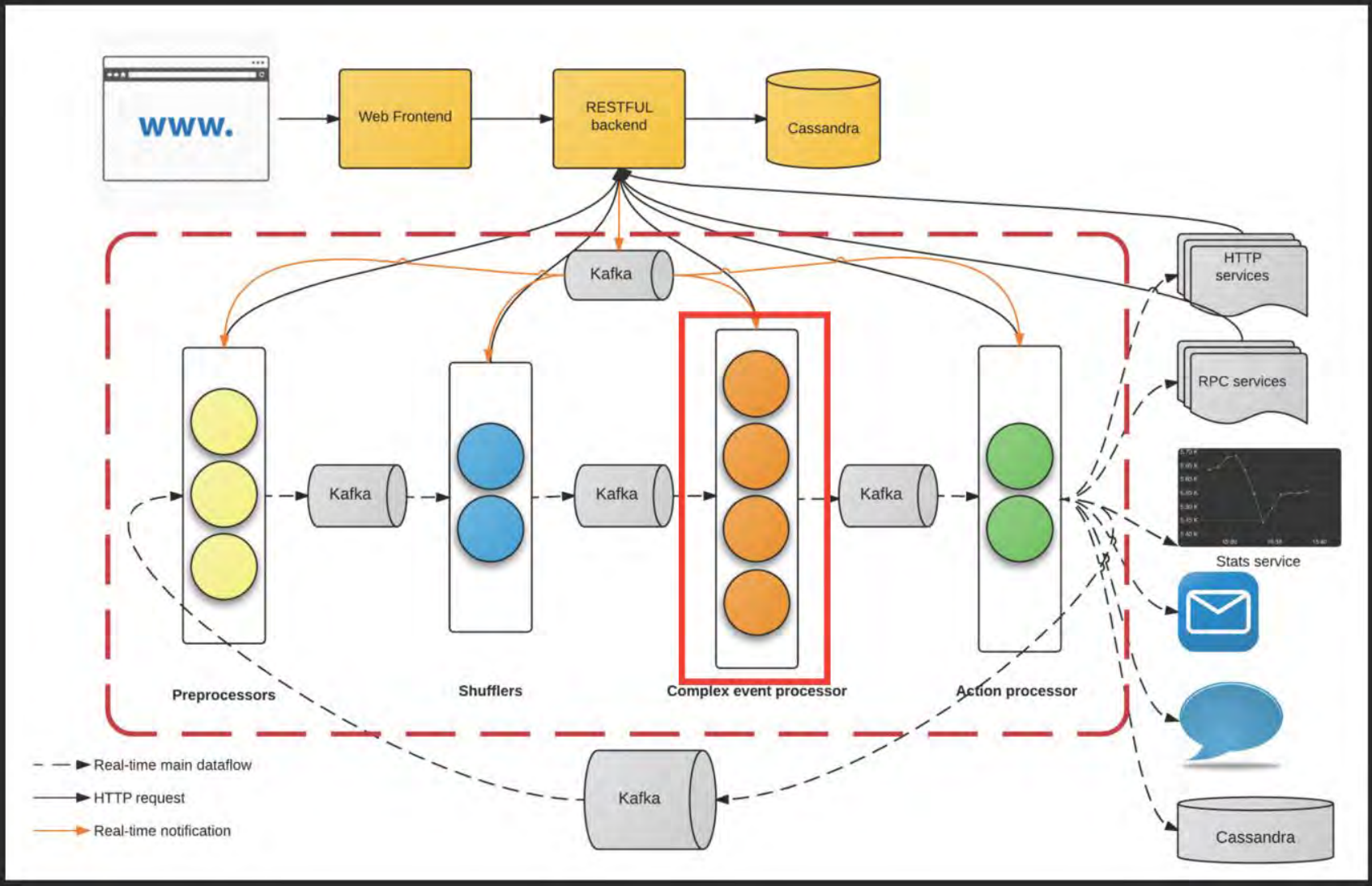

www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Stats service
Kafka
Kafka
Kafka
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification


www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Stats service
Kafka
Kafka
Kafka
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification


www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Stats service
Kafka
Kafka
Kafka
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification


www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Kafka
Kafka
Kafka
Stats service
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification


www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Stats service
Kafka
Kafka
Kafka
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification


www.
Web Frontend
RESTFUL backend
Cassandra
Kafka
HTTP services
RPC services
Stats service
Kafka
Kafka
Kafka
Preprocessors
Shufflers
Complex event processor
Action processor
Kafka
Cassandra
Real-time main dataflow
HTTP request
Real-time notification