servicenow

# Realtime User Analytics with ClickHouse

Amir Vaza

# Overview

- Motivation
- Appsee’s pre-ClickHouse Data Model
- Requirements
- ClickHouse
- Future Work

servicenow

# Motivation

Why would we replace something that works?

# Motivation

The motivation to replace our existing reliable, battle tested model comes from the product growth, allowing richer analytics features such as:

- Realtime dynamic slicing and dicing the data
- Realtime Funnels and Cohorts

servicenow

# Pre-ClickHouse Data Model

# Appsee's Pre-ClickHouse Data Model

Aggregation is performed upon incoming data processing to allow an interactive dashboard experience.

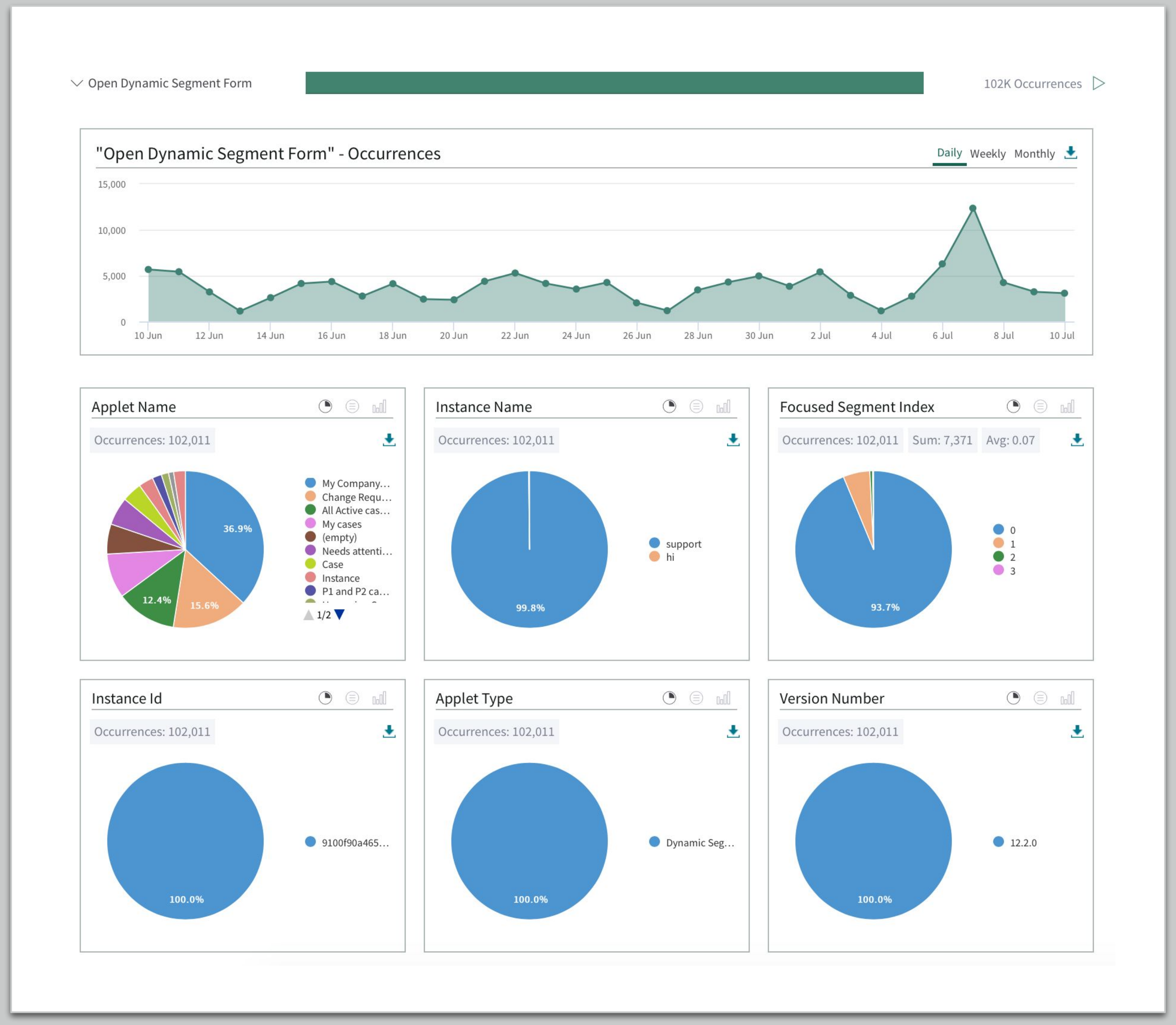
Pre-aggregated segmentation is based on a set of fixed dimensions:

- Application
- App Version
- Platform

Appsee holds ~30 different pre-aggregated reports, each segmented by the above dimensions + any other dimension relevant to the specific needs.

# Custom Data Points

- Data points sent by customers (SN app teams / SN customers)
- Each data point is in the form {EventName, Properties<K,V>}
- Appsee provides information about the occurrences count of each event and a GROUP BY breakdown of each property value's occurrences



Open Dynamic Segment Form — 102K Occurrences

"Open Dynamic Segment Form" - Occurrences

Applet Name — Occurrences: 102,011

Instance Name — Occurrences: 102,011

Focused Segment Index — Occurrences: 102,011 Sum: 7,371 Avg: 0.07

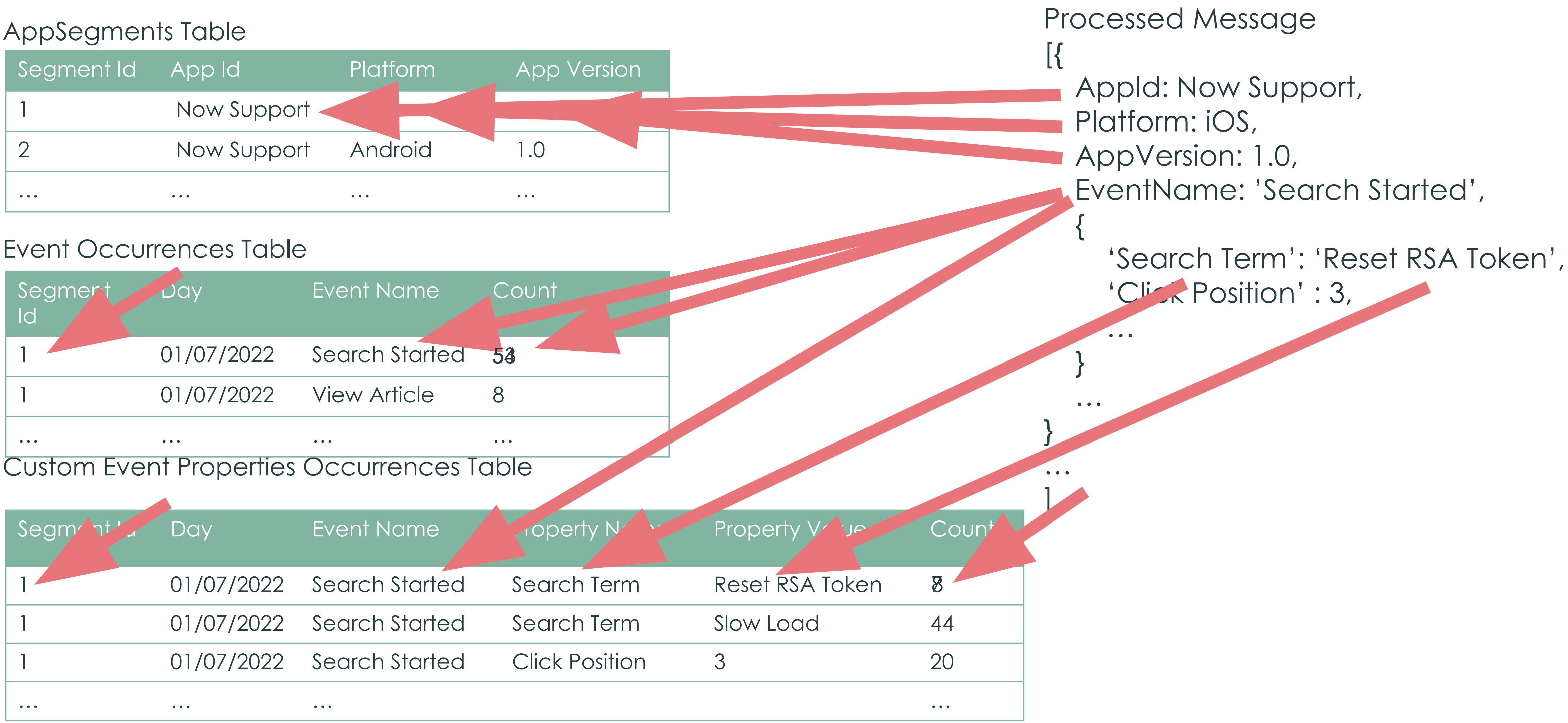
Instance Id — Occurrences: 102,011

Applet Type — Occurrences: 102,011

Version Number — Occurrences: 102,011

now.

# Pre Aggregated Data Model  - Example



- AppSegments Table

| Segment Id | App Id | Platform | App Version |
|---|---|---|---|
| 1 | Now Support | | |
| 2 | Now Support | Android | 1.0 |
| … | … | … | … |

- Event Occurrences Table

| Segment Id | Day | Event Name | Count |
|---|---|---|---|
| 1 | 01/07/2022 | Search Started | 5~~3~~4 |
| 1 | 01/07/2022 | View Article | 8 |
| … | … | … | … |

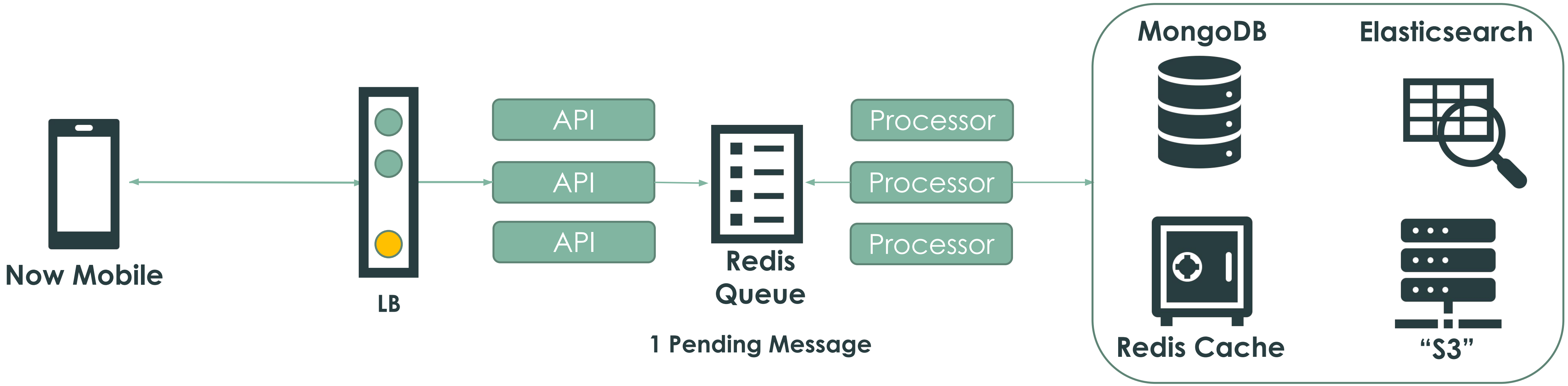
- Custom Event Properties Occurrences Table

| Segment Id | Day | Event Name | Property Name | Property Value | Count |
|---|---|---|---|---|---|
| 1 | 01/07/2022 | Search Started | Search Term | Reset RSA Token | ~~7~~8 |
| 1 | 01/07/2022 | Search Started | Search Term | Slow Load | 44 |
| 1 | 01/07/2022 | Search Started | Click Position | 3 | 20 |
| … | … | … | | | … |

Processed Message

```
[{
  AppId: Now Support,
  Platform: iOS,
  AppVersion: 1.0,
  EventName: 'Search Started',
  {
    'Search Term': 'Reset RSA Token',
    'Click Position' : 3,
    …
  }
  …
}
…
]
```

# Appsee's High Level Pipeline



```
Metadata Upload Call
{
   Session Data Points
   User Data Points
   …
}
```

```
Upload Result
{
   Accepted (Y / N)
   Skip upon next error
   …
}
```

(MongoDB) Update the session entry with all collected MD

(MongoDB) Update the user entry

(MongoDB) Calculate all analytics metrics and update state (Funnels, Cohorts, Retention…)

(MongoDB) Update all user-facing reports

(Elasticsearch) Perform Elasticsearch Indexing

# Requirements

# Non Functional Requirements

- **Scalability**
  - Horizontally scalable (CAP: prefer availability over consistency)
  - Native sharding + replication support
  - Write optimized
  - Prefer products with simple topologies, No Single Point Of Failure (SPOF)

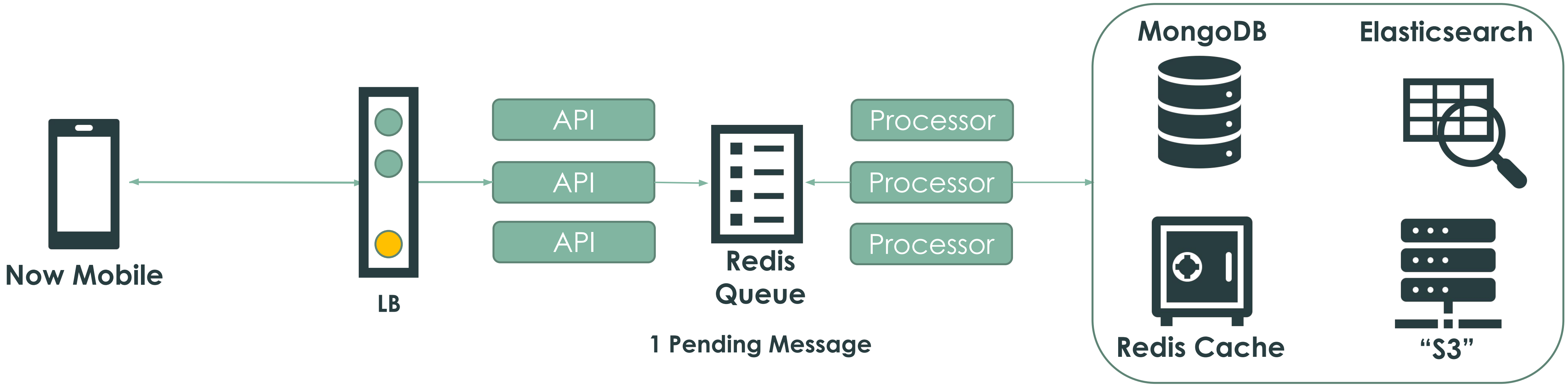
- **Maturity & Licensing**
  - Rich documentation + Community
  - Proper drivers (Python)
  - Fully open source, commercial support availability

# Functional Requirements

- Support all of Appsee's Existing dashboard aggregations:
  - Sub-second query time during insertion(1k data points/s per shard)
  - Same HW
- **Dynamic Segmentation**
  - **Allow segmenting by any session property**
  - **Allow segmenting by any event property value / combination of multiple properties**
  - **Same session "Join" - segment by multiple events (w/ prop values)**
- **Bonus: Data points ⇄ Users Join Aggregations**
  **(Users are mutable)**

# ClickHouse

# Appsee’s High Level Pipeline



(MongoDB) Update the session entry with all collected MD

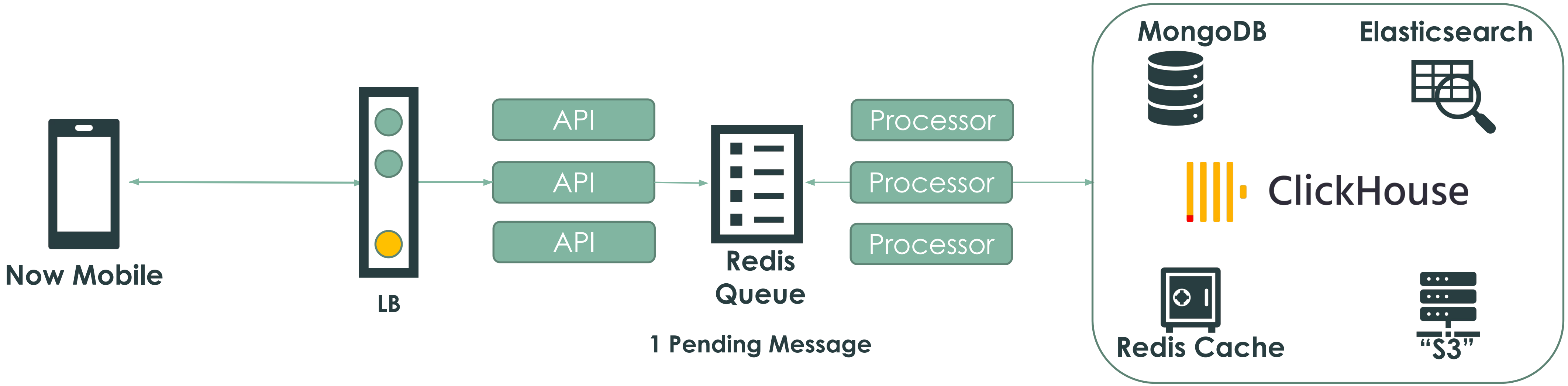
(MongoDB) Update the user entry

➡ (MongoDB) Calculate all analytics metrics and update state (Funnels, Cohorts, Retention…)

➡ (MongoDB) Update all user-facing reports

(Elasticsearch) Perform Elasticsearch Indexing

# Appsee’s High Level Pipeline



(MongoDB) Update the session entry with all collected MD

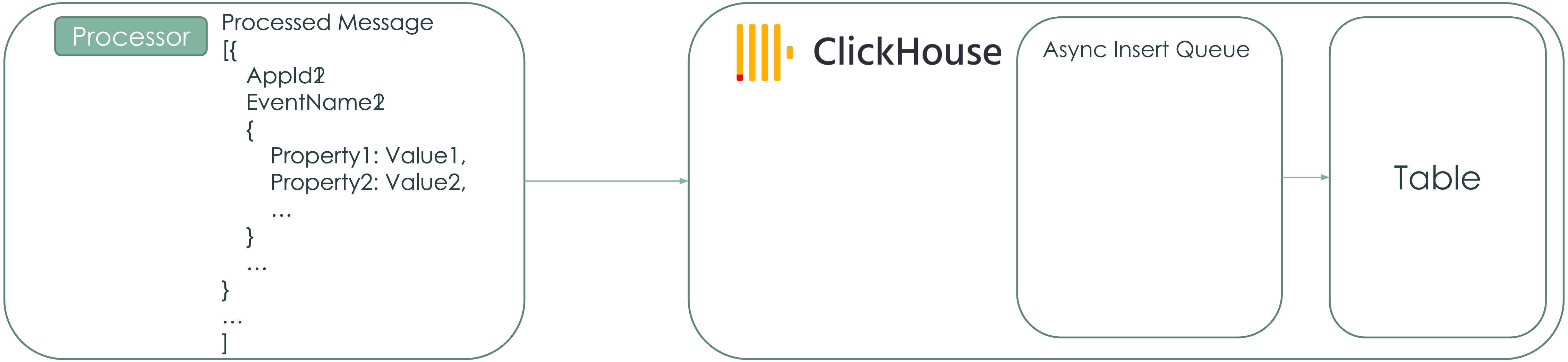
(MongoDB) Update the user entry

Break the MD file into data points

(ClickHouse) Insert all data points

(Elasticsearch) Perform Elasticsearch Indexing

# Insertion to ClickHouse

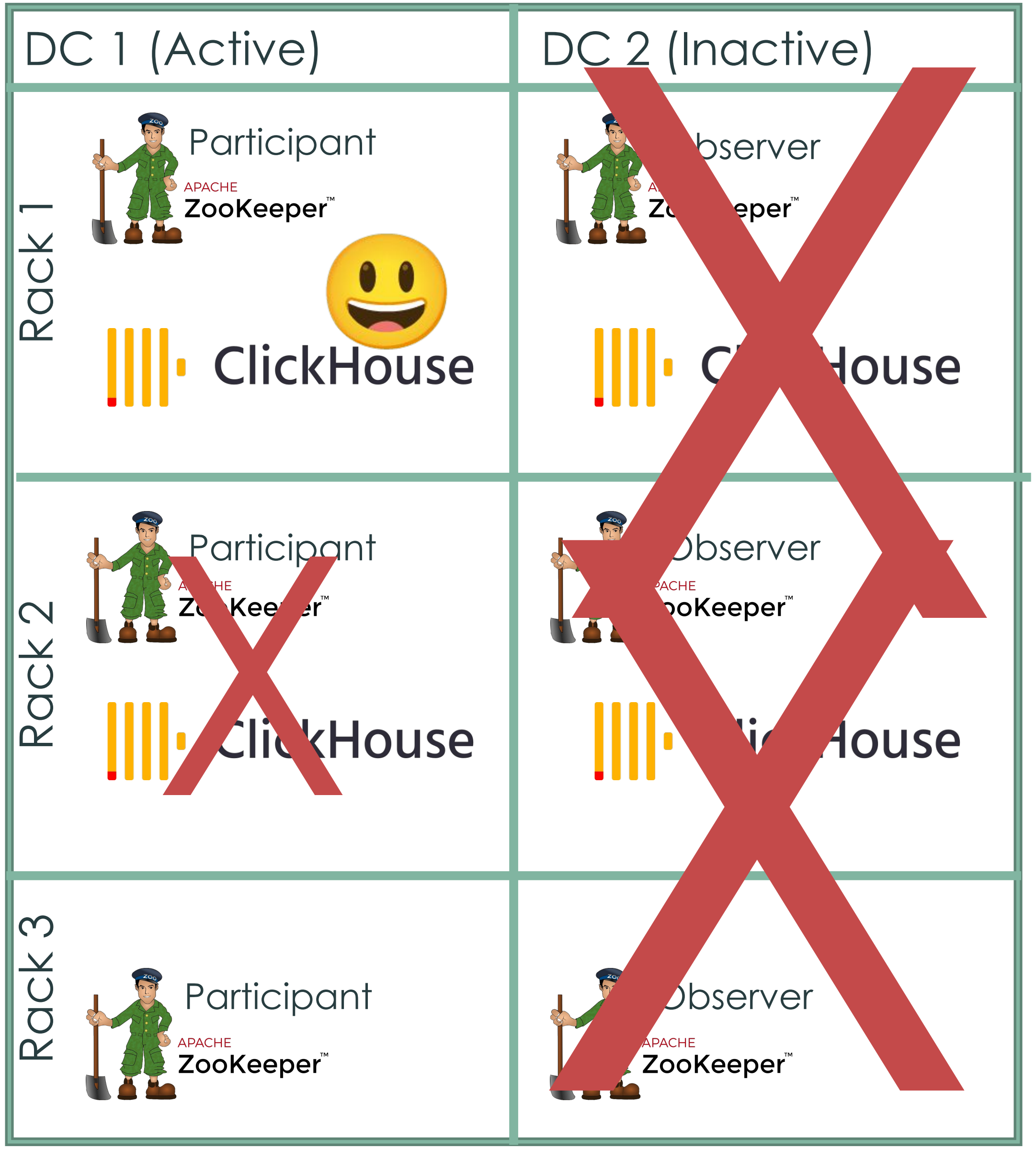


```
Processed Message
[{
   AppId2
   EventName2
   {
      Property1: Value1,
      Property2: Value2,
      …
   }
   …
}
…
]
```

# ClickHouse – Deployment

- 2 DCs (Active / Inactive)
- 3 Racks each DC
- A Zookeeper in each Rack
- Each CH shard has 2 replicas per DC (4 in total)
- 3 participant Zookeepers in the active DC
- 3 observer Zookeepers in the inctive DC

* Allowing DC + Rack failure



| | DC 1 (Active) | DC 2 (Inactive) |
|---|---|---|
| Rack 1 | Participant<br>APACHE ZooKeeper™<br>ClickHouse | Observer<br>APACHE ZooKeeper™<br>ClickHouse |
| Rack 2 | Participant<br>APACHE ZooKeeper™<br>ClickHouse | Observer<br>APACHE ZooKeeper™<br>ClickHouse |
| Rack 3 | Participant<br>APACHE ZooKeeper™ | Observer<br>APACHE ZooKeeper™ |

# ClickHouse Benchmarks - Overview

## Setup

- 2 Servers, each with 16 CPUs and 32GB RAM
- 430M Sessions (per model) over a single app == 2.1B Data points
- *Note*: Ingesting 4.5M data points took 124 seconds (~37.5K data points / second) and was bound on the loading machine and not the CH cluster

- Converted every Appsee session into a set of separate data points, each one representing an event occurrence

- Tested 6 different models with a set of 7 queries

# ClickHouse – Models Benchmark

| | Query | Result | Rows Scanned ** | Row Per Property | Nested | Map Type | Mapping | Mapping & Ordinals | Map Type & Ordinals |
|---|---|---|---|---|---|---|---|---|---|
| 1 | COUNT 'StartScreen' events | 964,553,500 | 970,536,535 | 2.29 | 0.76 | 0.77 | 0.79 | 0.13 | 0.12 |
| 2 | COUNT 'Open Tab' events WITH property 'Instance Name' = 'ABC' | 260,970,000 | 271,063,867 | 2.63 | 2.12 | 1.37 | 0.5 | 0.18 | 0.45 |
| 3 | COUNT event occurrences GROUPED BY event name | - | 2,154,496,000 | 8.46 | 2.27 | 2.26 | 2.26 | 0.59 | 0.57 |
| 4 | COUNT UNIQUE sessions WITH event 'Open Tab' * | 237,008,022 | 271,063,867 | 2.18 | 1.07 | 1.07 | 0.96 | 0.71 | 0.81 |
| 5 | COUNT UNIQUE sessions WITH event 'Open Tab' AND property 'Instance Name' = 'ABC' * | 233,070,147 | 271,063,867 | 3.17 | 2.3 | 1.71 | 0.73 | 0.35 | 0.68 |
| 6 | COUNT UNIQUE sessions WITH event 'Open Tab' AND property 'Instance Name' = 'ABC'<br>AND property 'Tab Name' = 'SafePass' * | 224,091,596 | 271,063,867 | 14.23 | 2.67 | 2.04 | 1.01 | 0.46 | 0.87 |
| 7 | COUNT UNIQUE sessions WITH 2 events and 1 property:<br>(event 'Open Tab' AND property 'Instance Name' = 'ABC')<br>AND<br>(event 'User Login' AND property 'Version Number' = '11.0.0')<br>* | 2,083,958 | 357,072,779 | 5.82 | 4.83 | 4.01 | 2.85 | 2.45 | 2.78 |
| | Storage(GB) | | | 32 | 14.02 | 13.26 | 15.02 | 15.66 | 12.8 |

* Estimation ± 1% miss

* * Irrelevant for model "Row Per Property" which has a different PK

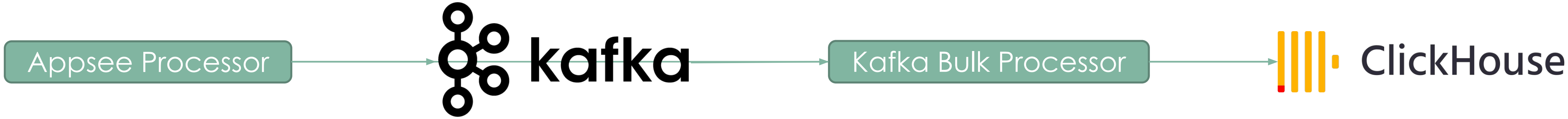
# ClickHouse – Chosen Data Model = Map Type

| App Id | Event Time | Session Id | User Id | Platform | Country | … | Event Name | Map<Event Property Name, Event Property Value> |
|---|---|---|---|---|---|---|---|---|

- Allows an arbitrary count of properties
- Sharded CH cluster using primary key: {AppId, startOfDay(EventTime), EventName, UserId}
- Ready for Ordinals migration in the future (The process of transforming String values into Integers)
- Easy to use

# Future ClickHouse Work

1. Idempotent insertion (Exactly once insert)



Disable ClickHouse Async Insert and rely on its dedup feature based on bulk hash

2. Insert User entities to allow JOINs

3. Realtime Cohorts & Funnels

4. Move to K8S

servicenow

# Realtime User Analytics with ClickHouse

Amir Vaza @ LinkedIn

Architect & Lead at ServiceNow