喜马拉雅

Application of 

at Ximalaya.com

alex.huang@ximalaya.com

Agenda

- Overview of our ClickHouse applications
- Integrating ClickHouse
- Evaluating ClickHouse
- Monitoring ClickHouse
- Optimizing funnel analysis
- Wish list for ClickHouse

Why OLAP in Ximalaya.com ?

- About Ximalaya.com
  - Top audio sharing platform in China
- Huge volume of
  - User Behavior Tracking log: page visit, album play, advertisement click…
  - System logs
  - Order database
  - …
- Increase business ROI by analyzing above data
  - How many users pass the page visit funnel to finally buy an album?
  - Application/infrastructure monitoring
- Start with ClickHouse in May 2018

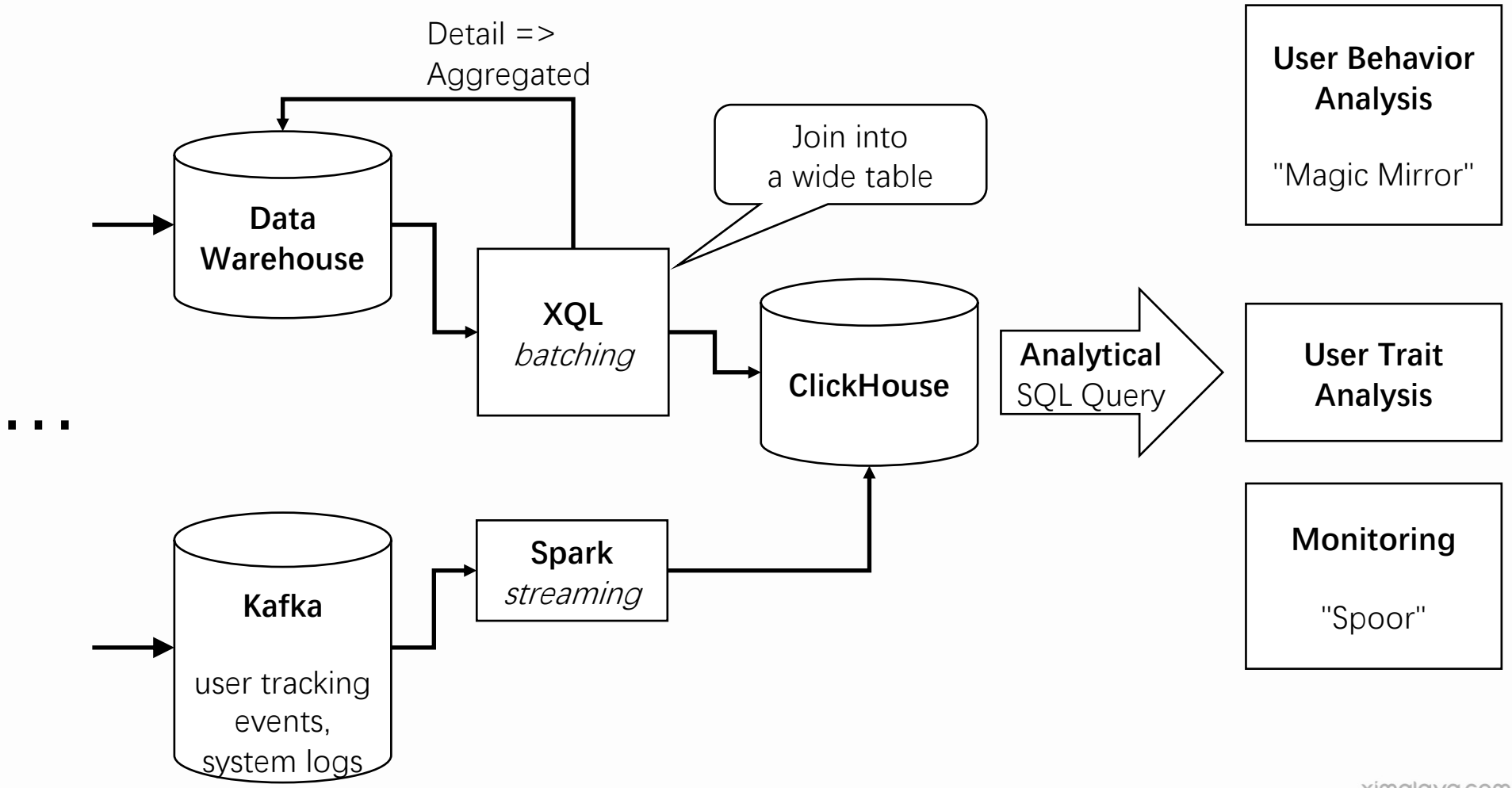
Why we choose ClickHouse as the OLAP Solution?

- Amazing fast
  - SIMD instructions not yet generated by production JVM
  - Linearly scalable
- Storage of each single raw record; query for trouble shooting
- SQL as the query language.
  - Built-in, official SQL support => robust.
- Isolation of workloads by *logical* clusters
  - One workload, one dedicated (small) cluster
  - Configurable at runtime, no need to reboot
- Free and open source

3 Major OLAP Business using ClickHouse

- User behavior analysis “Magic Mirror”
  - Funnel / Path / Retention analysis …
- User trait analysis
  - Find “age distribution of all users in Shanghai, not like music, and retired and … ”
  - (WHERE condition + GROUP BY) involving 50+ user attributes
    - Free to filter / group by any combination of columns
- Monitoring
  - Application / JMX / JVM metrics
  - Dump HDFS audit log / directory strcture into ClickHouse
    - Find most frequently accessed HDFS dirs

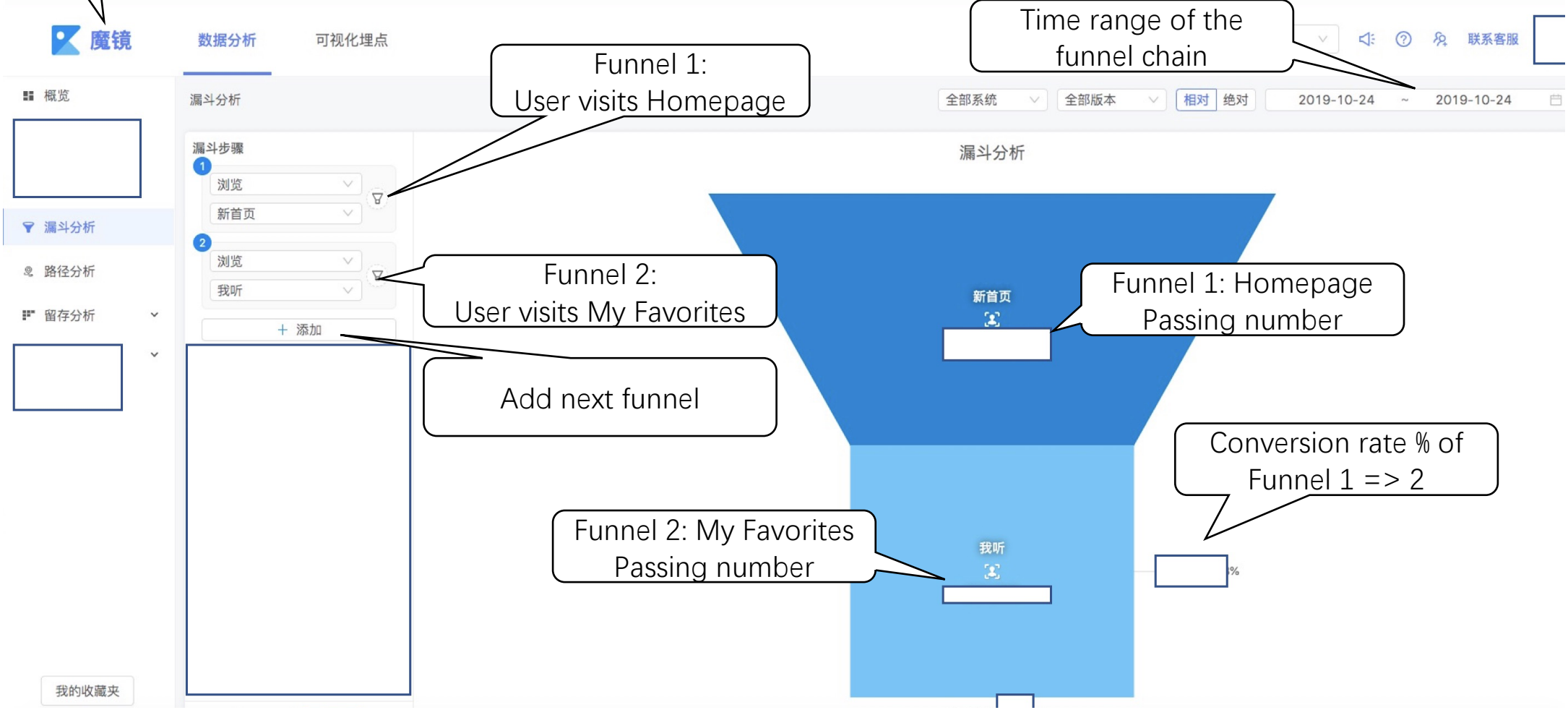
Architecture



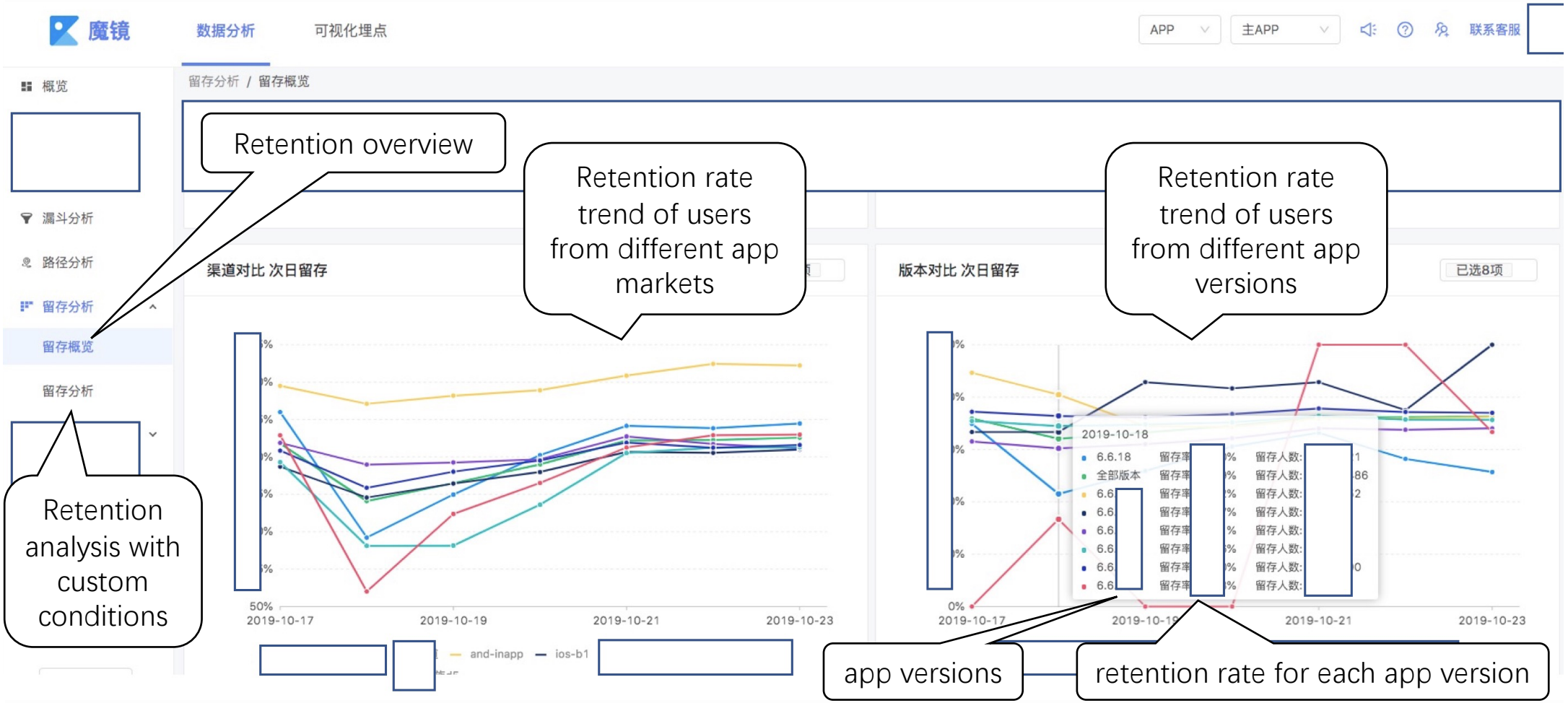
喜马拉雅

Magic Mirror

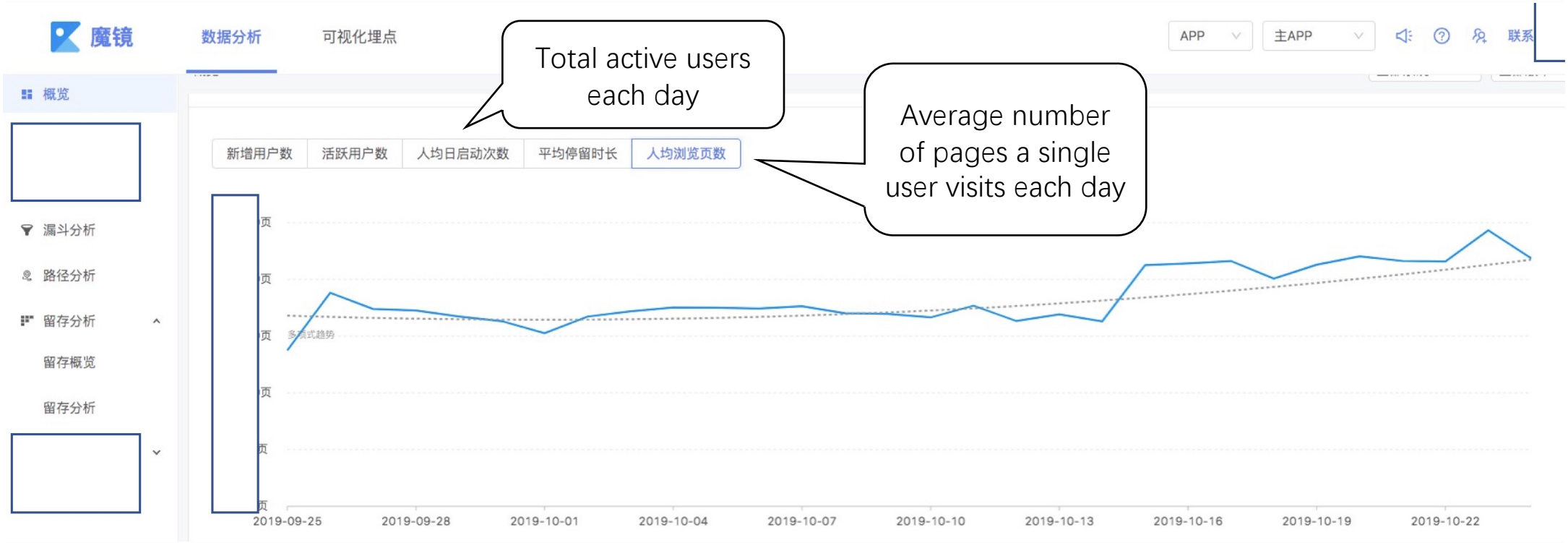
Funnel Analysis by "Magic Mirror" (魔镜)



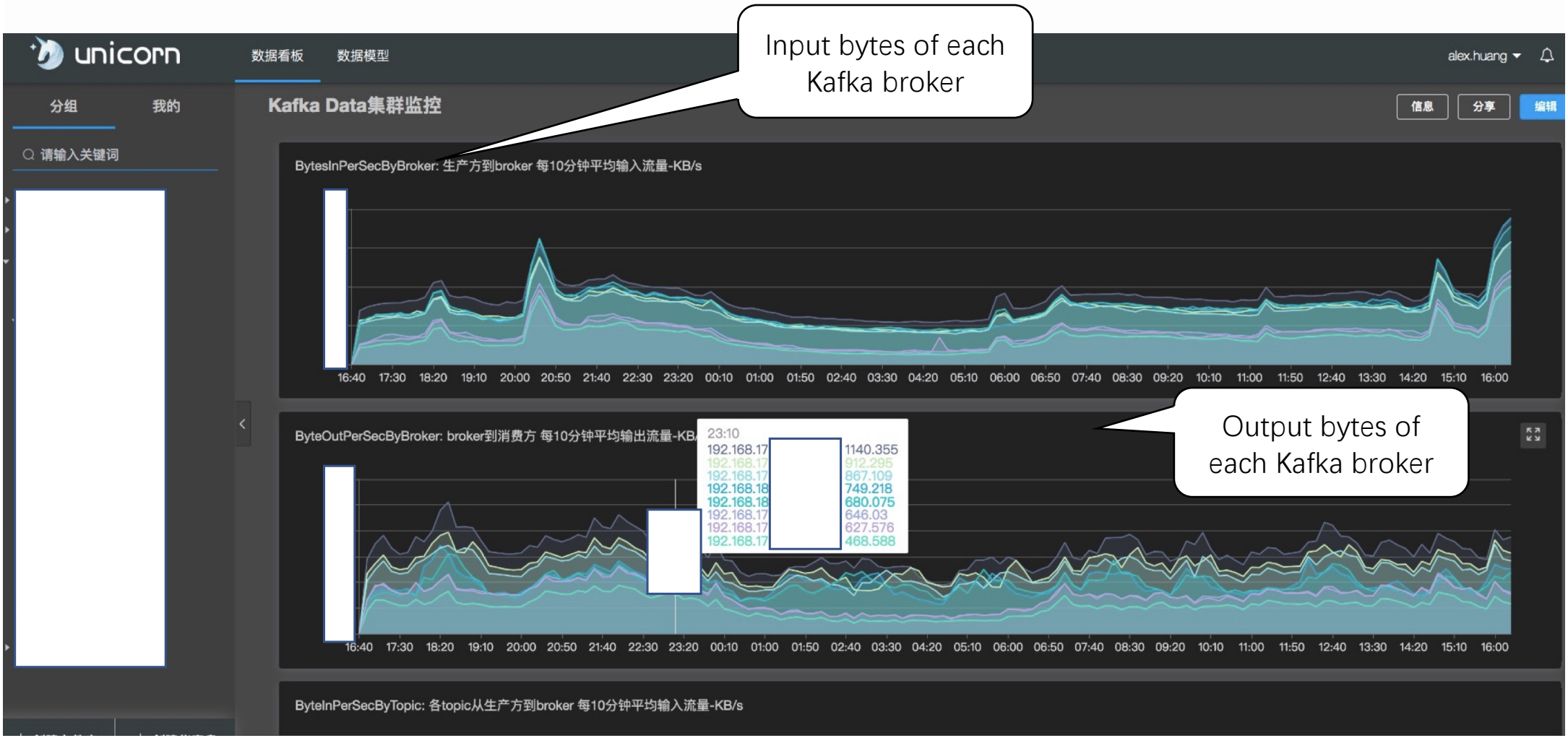
Retention Analysis by "Magic Mirror" (魔镜)



Page visit statistics by "Magic Mirror" (魔镜)



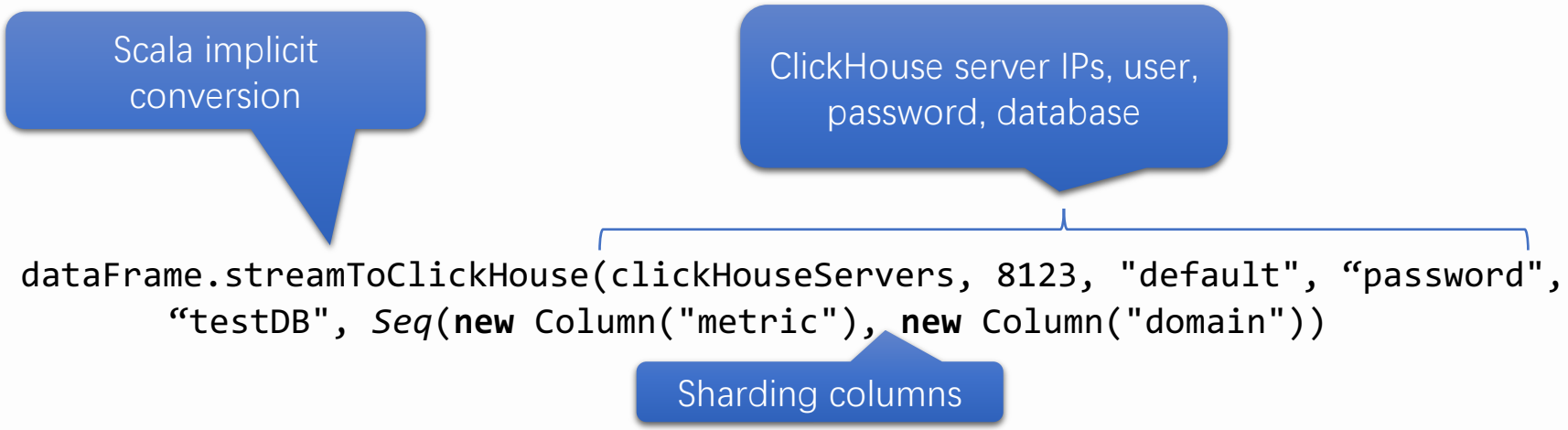
Monitoring: ClickHouse as a time series DB



Monitoring: HDFS audit log statistics

Integrating ClickHouse

- Seamless integration with Spark DataFrame



- one Spark partition 1:1 one ClickHouse server
  - Crossing does not help insertion performance
  - **`dataFrame.repartition(clickHouseServers.size)`**
  - Implement a Spark foreach writer to insert into ClickHouse

Integrating ClickHouse

- Explicit validation of Spark row schema v.s. ClickHouse table schema
  for the 1st row

  Friendly error message with a *line context* to ease trouble shooting
  "String [abc] cannot be converted to UInt32. Exception··· The whole line is ··· at offset 12631··· "

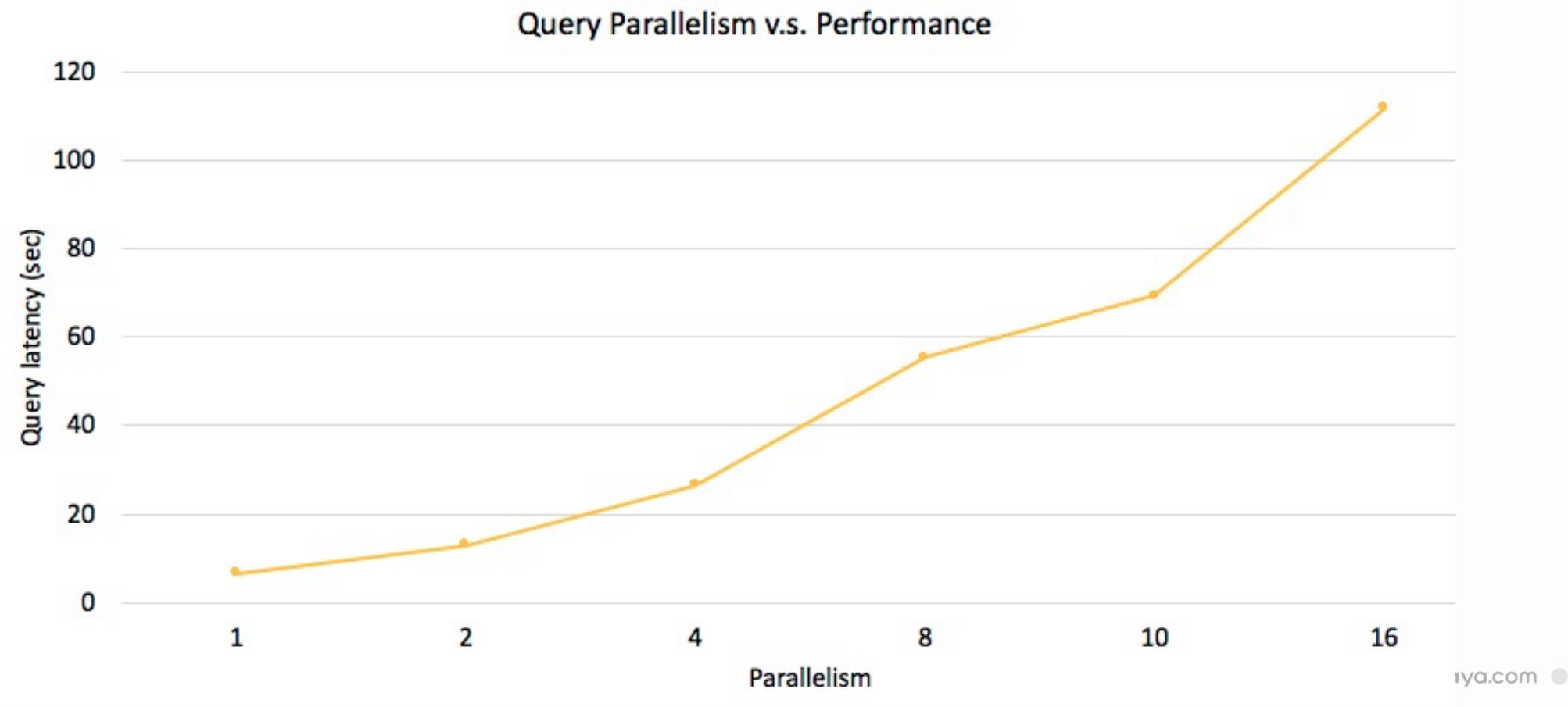
- Provide more implicit type conversions
  - String yyyy-MM-dd HH:mm:ss.*SSS* => DateTime
  - String "true" => UInt8 value 1

- Support both protocols: JDBC (TabSeparated) or CSV format, configurable
  - Similar insertion performance between them

Evaluating ClickHouse

- A test cluster of 4 nodes.
- Each node:
  - CPU: Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz. 6 core
  - Memory: 64GB
- Scenario
  - 1 billion page visit logs join 0.12 billion user trait records

Evaluating ClickHouse
Query concurrency v.s. query latency



Evaluating ClickHouse
`max_threads` boosts query *with a limit*

max_threads	Query Latency (seconds)
1	33.15
2	16.85
4	8.60
8	6.15
16	6.22
32	6.44

Evaluating ClickHouse

- When inserting, sharding similar rows onto the same machine
  - Saves disk space by 2X+
  - Because similar rows, if put together, have a higher compression ratio

e.g.: A distributed table of 10 billion user behavior logs:

Sharding policy	Disk space consumption on a single node
Random sharding	1.7 GB
Shard by user id	0.5 GB

Monitoring ClickHouse

- A monitoring service polling meta tables
  - system.parts, system.query_log, system.merges …
- Display web UI of:
  - Currently running SQLs, on-going merges
  - Historical big queries scanning most records
  - Line charts of data growth trend, per table * day
  - …
- Define partition expiry period per table
  - Drop expired partitions periodically

Funnel Analysis - Background

- Page visit logs

Timestamp	User	Page
2018-07-13 10:00:01	user1	Home Page
2018-07-13 10:00:02	user2	Home Page
2018-07-13 10:00:03	user3	Home Page
2018-07-13 10:00:04	user4	Home Page
2018-07-13 10:00:05	user5	Home Page
2018-07-13 10:00:06	user1	Detail
2018-07-13 10:00:07	user2	Detail
2018-07-13 10:00:08	user3	Detail
2018-07-13 10:00:09	user3	Order
2018-07-13 10:00:10	user4	Order

Funnel Analysis – the Problem

- For funnel chain: Home Page => Detail => Order:
  - How many people passed one, two or all three funnels respectively ?

- Aggregated page visit path:

User	Page Visit Path
user1	Home Page => Detail
user2	Home Page => Detail
user3	Home Page => Detail => Order
user4	Home Page => Order
user5	Home Page

Funnel Analysis – Expected Result

User	Page Visit Path
user1	Home Page => Detail
user2	Home Page => Detail
user3	Home Page => Detail => Order
user4	Home Page => Order
user5	Home Page

- Expected output
  - for the given funnel chain: Home Page => Detail => Order
  - Num of users passing Home Page only: 2 (user4, user5)
  - Num of users passing Home Page => Detail only: 2 (user1, user2)
  - Num of users passing entire chain: 1 (user3)

Funnel Analysis - Preparation

- Generate page visit path for each user

```
SELECT
        user,
        groupArray(page) as pages,
        groupArray(timestamp) as timestamps,
        arrayEnumerate(pages) as index
FROM (SELECT * FROM client_log_all ORDER BY timestamp) GROUP BY user
```

- For each user,
  - create a page path, a timestamp sequence, an index sequence
  - all of them are arrays sorted by timestamp

Funnel Analysis – Solution 1

```
SELECT
  user,
  groupArray(page) as pages,
  groupArray(timestamp) as timestamps,
  arrayEnumerate(pages) as index,
  arrayFilter((i, p) -> (pages[i] = 'HomePage' AND pages[i+1]!= 'Detail'), index, pages) as
       level_1,

  // In pages array, find a subarray of [HomePage, Detail, Anything except Order]
  arrayFilter((i, p) -> (pages[i] = 'HomePage' AND pages[i+1]= 'Detail' AND
       pages[i+2]!='Order'), index, pages) as level_2,

  // In pages array, find a subarray of [HomePage, Detail, Order]
  arrayFilter((i, p) -> (pages[i] = 'HomePage' AND pages[i+1]= 'Detail' AND
       pages[i+2]='Order'), index, pages) as level_3

FROM (SELECT * FROM client_log_all ORDER BY timestamp) GROUP BY user
```

- If level_2 is not null, the current user passes exactly 2 funnels only.

Funnel Analysis – Solution 2

```
SELECT
 user,
 groupArray(timestamp) as unsortedTimestamps,
 arraySort((p,t)->t, groupArray(page), unsortedTimestamps) as sortedPages,

 // Left-shift sorted pages by one element and add a trailing zero
 arrayPushBack(arraySlice(sortedPages,2),0) as nextSortedPages,

 // Left-shift sorted pages by two elements and add two trailing zeros
 arrayPushBack(arraySlice(nextSortedPage,2),0) as nextNextSortedPages,

 arrayFirst((p1, p2) -> ( p1 = 'HomePage' AND p2!= 'Detail'), sortedPages, nextSortedPages)
       as level_1,
 …
 arrayFirst((p1, p2, p3) -> (p1 = 'HomePage' AND p2= 'Detail' AND p3!= 'Order' ),
       sortedPages, nextSortedPages, nextNextSortedPages) as level_2
 …
FROM client_log_all GROUP BY user
```

- In real world, add time constraints similarly: adjacent events cannot span over X minutes

Funnel Analysis

Solution 2 is over 7X faster than Solution 1

Solution 1: Find subarray starting at i, such that

```
pages[i] = 'Home Page' AND pages[i+1] = 'Detail' AND pages[i+2] = 'Order'
```

pages	…	…	…	Home Page	Detail	Order	…

Solution 2: Find subarrays starting at i, such that

```
pages[i] = 'Home Page' AND nextPages[i] = 'Detail' AND nextNextpages[i] = 'Order'
```

pages	…	…	…	Home Page	Detail	Order	…
nextPages	…	…	Home Page	Detail	Order	…	0
nextNextPages	…	Home Page	Detail	Order	…	0	0

Padding zeros at the tail so that arrays have the same length, required by arrayFirst()

Wish List

- Make join faster for MergeTree tables
  - table A join table B on id
    - If table B is a MergeTree table, and id is its primary key,
    - can we skip unmatching blocks of table B quickly, using MergeTree index of table B ?
- Explain query
- One-stop, insight monitoring solution
  - Detailed internal metrics
  - Also OS environment metrics
  - Visualization of metrics;

Summary

- Amazing fast; workload well isolated
- Nice to keep raw records for trouble-shooting.
  - Aggregation manually verifiable, explainable to users.
- Easy to develop applications with SQL
- Wish for even faster join
- Continue to be widely used for OLAP and montoring in the Firm
- We provided easy integration, performance benchmark, best practices & optimizations

Thank you

Q & A

We are hiring: