From ElasticSearch To ClickHouse, Log Analysis Practice in bilibili

Jiaqi Shu(束家麒)
bilibili INFRA

Tao Ling (凌涛)
bilibili OLAP

Agenda

- Log Analysis System based on ClickHouse
  - Challenge & problems
  - Why ClickHouse?
  - How we build
- Optimization & Customization of ClickHouse for Log Analysis
  - Challenges for ClickHouse
  - Overall Design
  - Implicit Map

Log Analysis System based on ClickHouse

Challenge & problems

- PB level data, Based on Elasticsearch, 1k+. nodes
- Write performance bottleneck
- Heavy cost on index
- Operation and maintenance costs

Why ClickHouse?

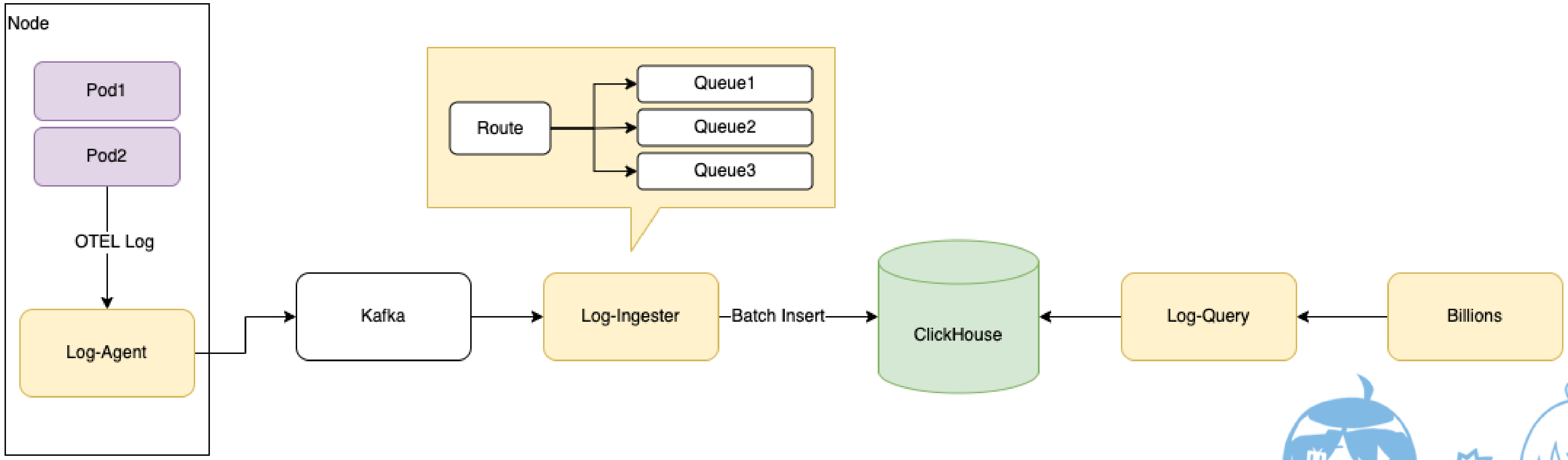
Requirements

- Low cost
- High performance
- Support semi-structure

ClickHouse

- Cost saving over 60%
- Avg Query latency < 1s

How we build


Node
Pod1
Pod2
OTEL Log
Log-Agent
Kafka
Route
Queue1
Queue2
Queue3
Log-Ingester
Batch Insert
ClickHouse
Log-Query
Billions

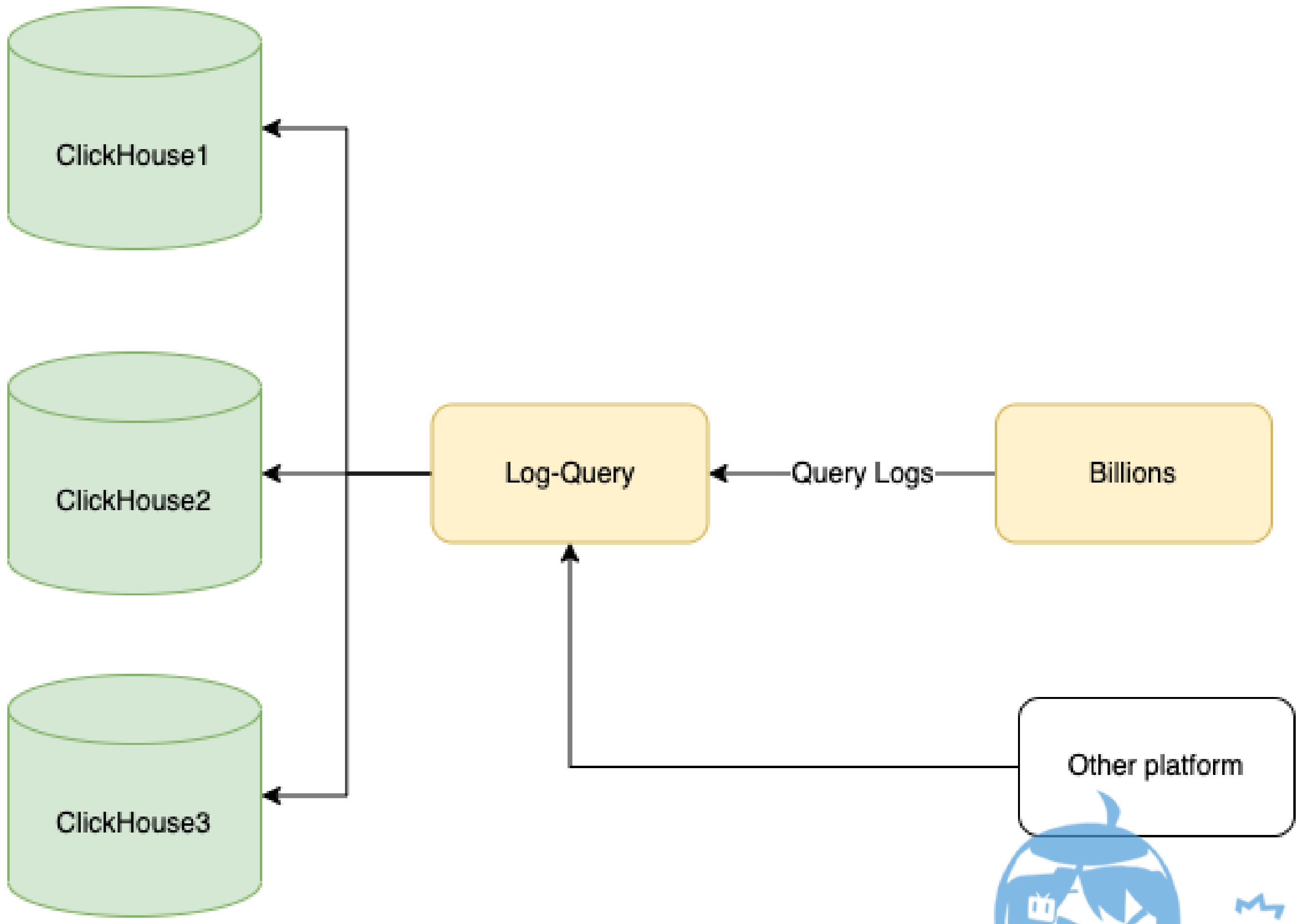
How we build

```
Create Table <log_app_name>
{
    timestamp Datetime64,
    hostname  String CODC(ZSTD(1)),
    zone      String CODC(ZSTD(1)),
    log       String CODC(ZSTD(1)),
    ....
    string_map  MapV2(String, Nullable (String))
                              CODEC(ZSTD(1))
    number_map  MapV2(String, Nullable (Float64))
                              CODEC(ZSTD(1))
    bool_map  MapV2(String, Nullable (UInt8))
}
....
```

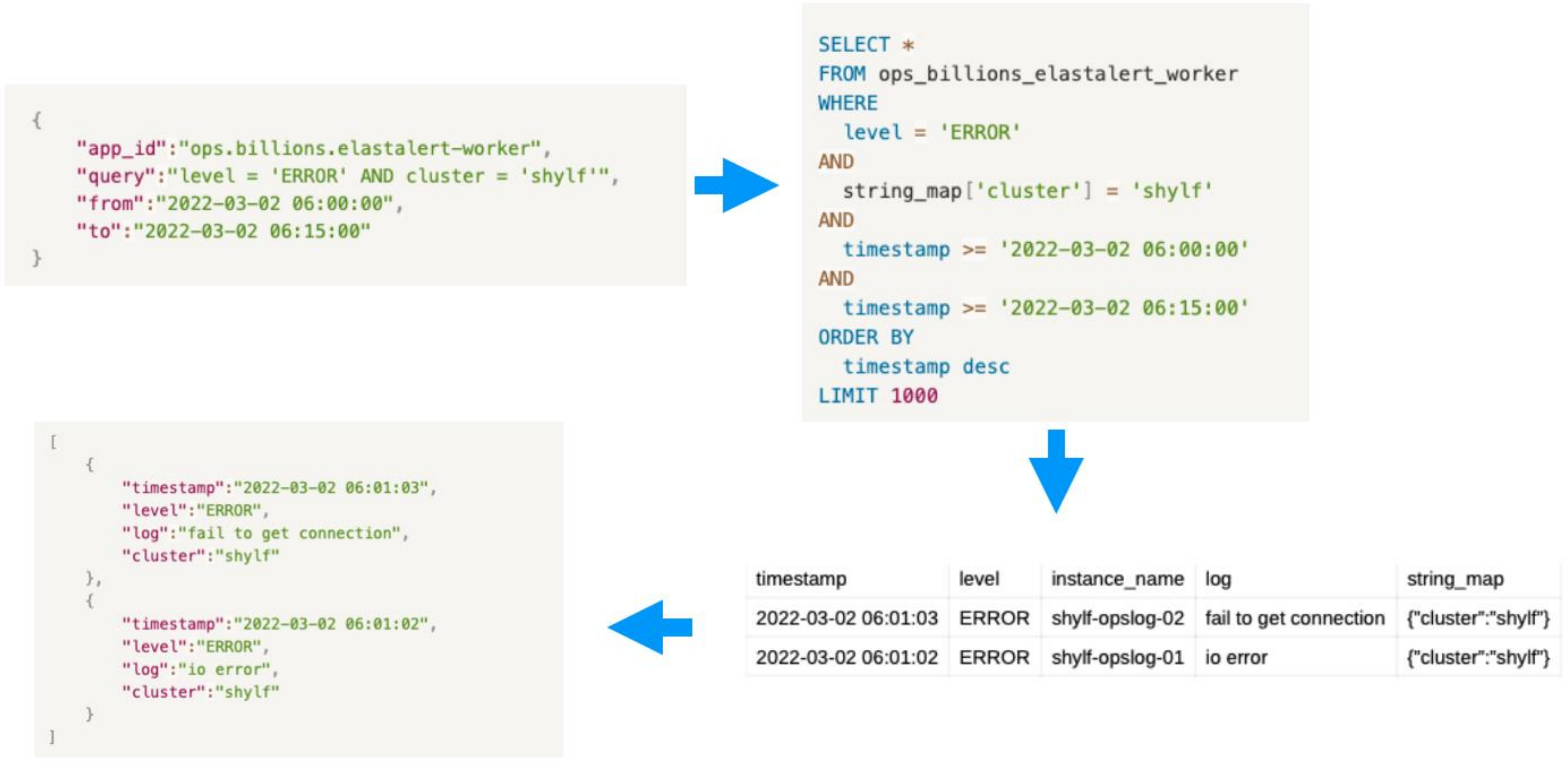
How we build

Log Query

- High level query
- Query routing
- Rate Limit, Access log…



How we build



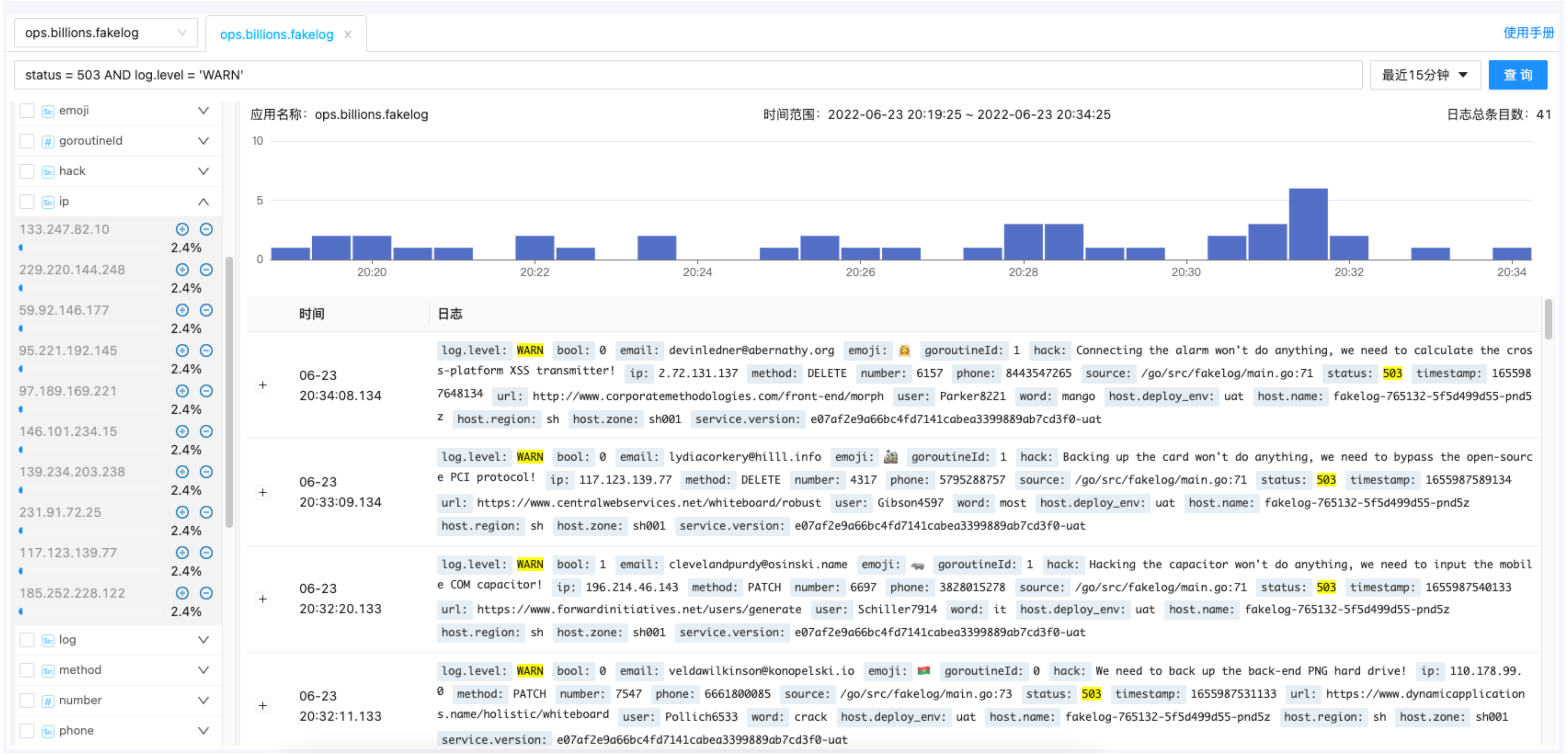
```
{
    "app_id":"ops.billions.elastalert-worker",
    "query":"level = 'ERROR' AND cluster = 'shylf'",
    "from":"2022-03-02 06:00:00",
    "to":"2022-03-02 06:15:00"
}
```

```
SELECT *
FROM ops_billions_elastalert_worker
WHERE
  level = 'ERROR'
AND
  string_map['cluster'] = 'shylf'
AND
  timestamp >= '2022-03-02 06:00:00'
AND
  timestamp >= '2022-03-02 06:15:00'
ORDER BY
  timestamp desc
LIMIT 1000
```

timestamp	level	instance_name	log	string_map
2022-03-02 06:01:03	ERROR	shylf-opslog-02	fail to get connection	{"cluster":"shylf"}
2022-03-02 06:01:02	ERROR	shylf-opslog-01	io error	{"cluster":"shylf"}

```
[
    {
        "timestamp":"2022-03-02 06:01:03",
        "level":"ERROR",
        "log":"fail to get connection",
        "cluster":"shylf"
    },
    {
        "timestamp":"2022-03-02 06:01:02",
        "level":"ERROR",
        "log":"io error",
        "cluster":"shylf"
    }
]
```

How we build



```
status = 503 AND log.level = 'WARN'
```

应用名称：ops.billions.fakelog　　时间范围：2022-06-23 20:19:25 ~ 2022-06-23 20:34:25　　日志总条目数：41

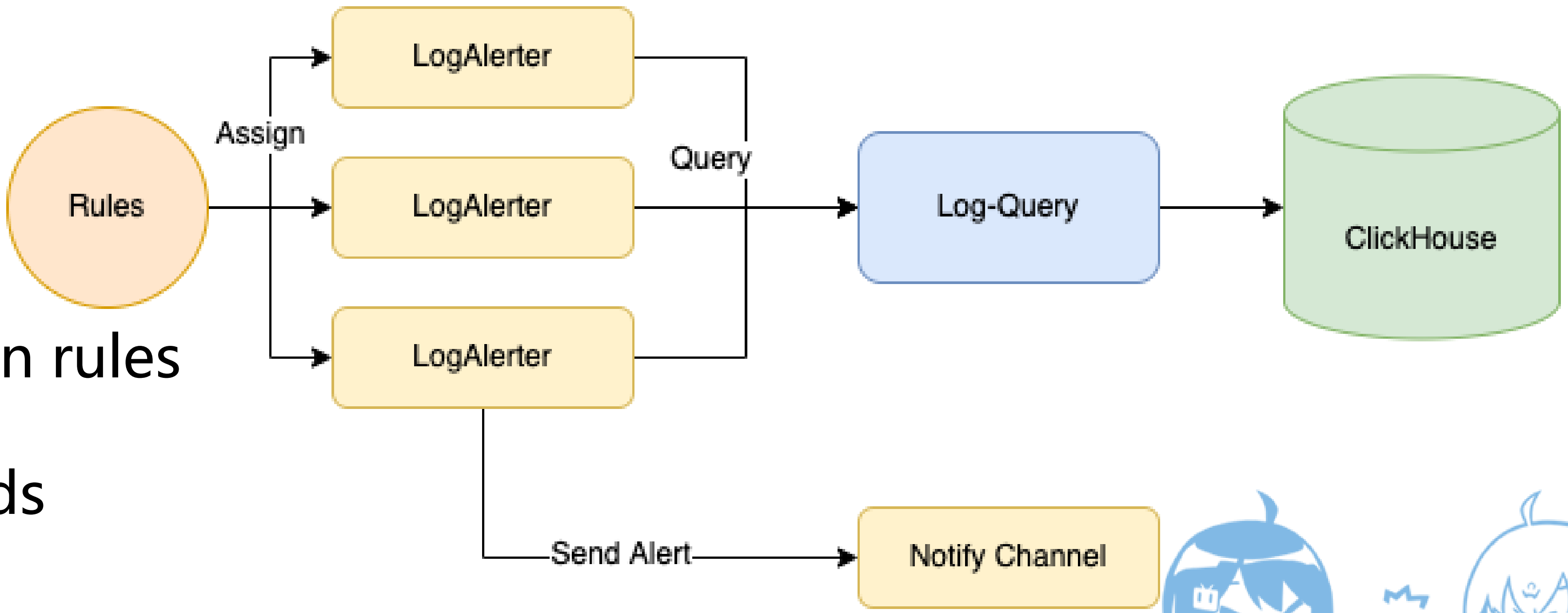
ip	
133.247.82.10	2.4%
229.220.144.248	2.4%
59.92.146.177	2.4%
95.221.192.145	2.4%
97.189.169.221	2.4%
146.101.234.15	2.4%
139.234.203.238	2.4%
231.91.72.25	2.4%
117.123.139.77	2.4%
185.252.228.122	2.4%

```
06-23 20:34:08.134
log.level: WARN  bool: 0  email: devinledner@abernathy.org  emoji: 🙆  goroutineId: 1  hack: Connecting the alarm won't do anything, we need to calculate the cross-platform XSS transmitter!  ip: 2.72.131.137  method: DELETE  number: 6157  phone: 8443547265  source: /go/src/fakelog/main.go:71  status: 503  timestamp: 1655987648134  url: http://www.corporatemethodologies.com/front-end/morph  user: Parker8221  word: mango  host.deploy_env: uat  host.name: fakelog-765132-5f5d499d55-pnd5z  host.region: sh  host.zone: sh001  service.version: e07af2e9a66bc4fd7141cabea3399889ab7cd3f0-uat

06-23 20:33:09.134
log.level: WARN  bool: 0  email: lydiacorkery@hilll.info  emoji: 🚵  goroutineId: 1  hack: Backing up the card won't do anything, we need to bypass the open-source PCI protocol!  ip: 117.123.139.77  method: DELETE  number: 4317  phone: 5795288757  source: /go/src/fakelog/main.go:71  status: 503  timestamp: 1655987589134  url: https://www.centralwebservices.net/whiteboard/robust  user: Gibson4597  word: most  host.deploy_env: uat  host.name: fakelog-765132-5f5d499d55-pnd5z  host.region: sh  host.zone: sh001  service.version: e07af2e9a66bc4fd7141cabea3399889ab7cd3f0-uat

06-23 20:32:20.133
log.level: WARN  bool: 1  email: clevelandpurdy@osinski.name  emoji: 🦏  goroutineId: 1  hack: Hacking the capacitor won't do anything, we need to input the mobile COM capacitor!  ip: 196.214.46.143  method: PATCH  number: 6697  phone: 3828015278  source: /go/src/fakelog/main.go:71  status: 503  timestamp: 1655987540133  url: https://www.forwardinitiatives.net/users/generate  user: Schiller7914  word: it  host.deploy_env: uat  host.name: fakelog-765132-5f5d499d55-pnd5z  host.region: sh  host.zone: sh001  service.version: e07af2e9a66bc4fd7141cabea3399889ab7cd3f0-uat

06-23 20:32:11.133
log.level: WARN  bool: 0  email: veldawilkinson@konopelski.io  emoji: 🇧🇫  goroutineId: 0  hack: We need to back up the back-end PNG hard drive!  ip: 110.178.99.0  method: PATCH  number: 7547  phone: 6661800085  source: /go/src/fakelog/main.go:73  status: 503  timestamp: 1655987531133  url: https://www.dynamicapplications.name/holistic/whiteboard  user: Pollich6533  word: crack  host.deploy_env: uat  host.name: fakelog-765132-5f5d499d55-pnd5z  host.region: sh  host.zone: sh001  service.version: e07af2e9a66bc4fd7141cabea3399889ab7cd3f0-uat
```

How we build

Log Alerter

- Distributed
- Various calculation rules
- Alert in 30 seconds



Optimization & Customization of ClickHouse for Log Analysis

Challenges for ClickHouse

Data writing:

- Large number of applications
- Low write latency
- Dynamic schema for private attribute
- Specify codec to save more disk

Data querying:

- Query based on application use different conditions
- Clickhouse native map type is underperforming

Overall Design

- Each application has a separate table

  We can specify different settings based on application at table level.
  1. Skipping indexes
  2. Max implicit columns size

- Storage policy for hot and cold data

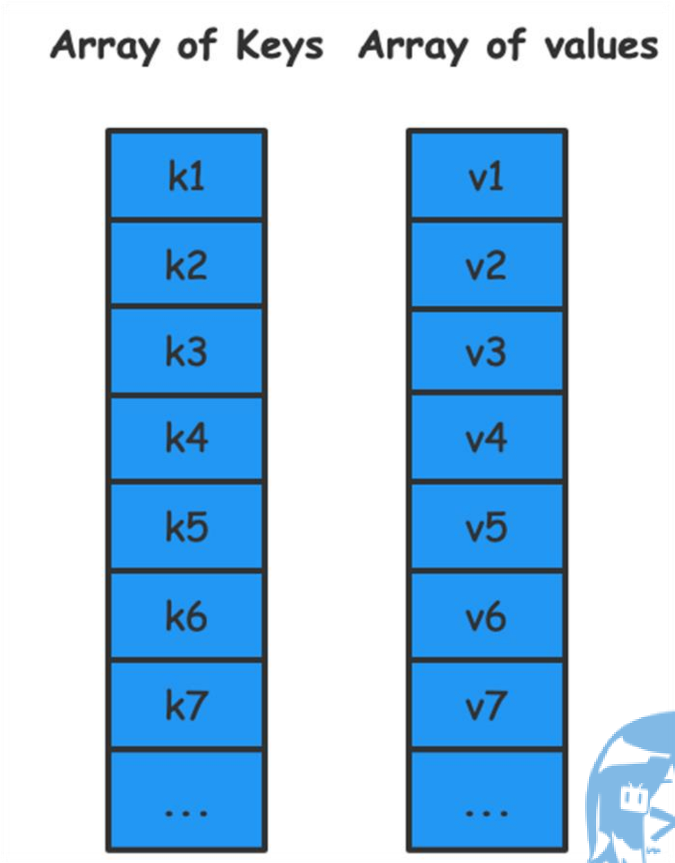
  Specify storage policy in table DDL
  1. NVME for hot data
  2. SATA for cold data
  3. ZSTD codec better than default codec （ZSTD can save up to 60% more space in some scenarios)

- Use Map type for dynamic attribute

Implicit Map

Native Map Type

1. Tuple of Arrays
2. Queries read redundant data
3. Skipping indexes don't support Map type

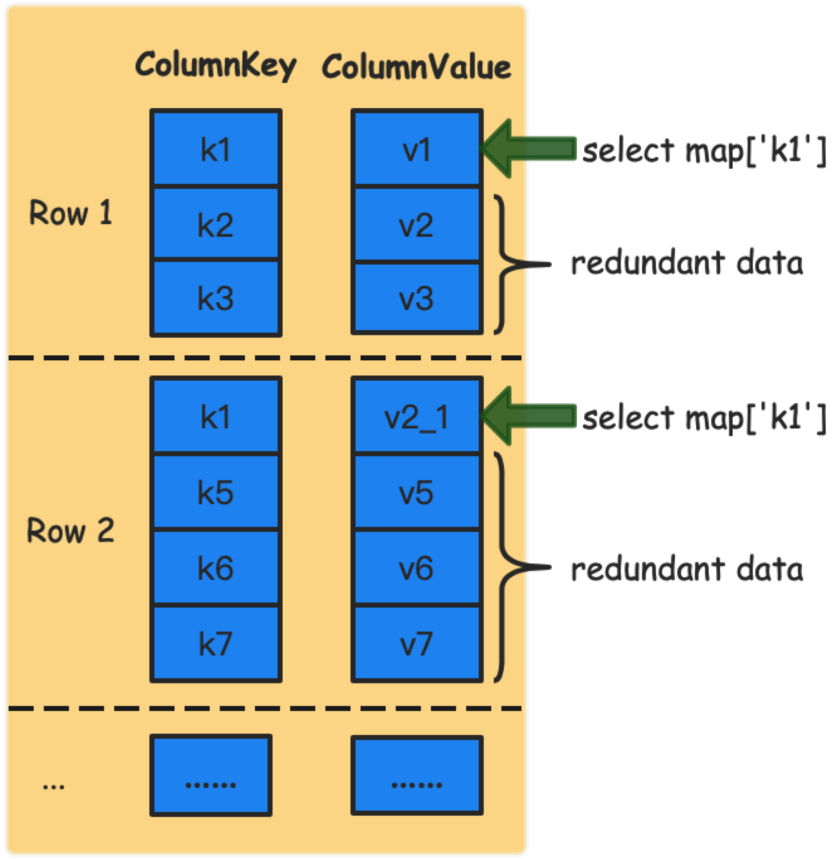


Implicit Map

Redundant data read

SELECT map['k1'] FROM table

- 2 arrays will be read
- All elements of map will be read
- Every rows of 2 arrays will be read



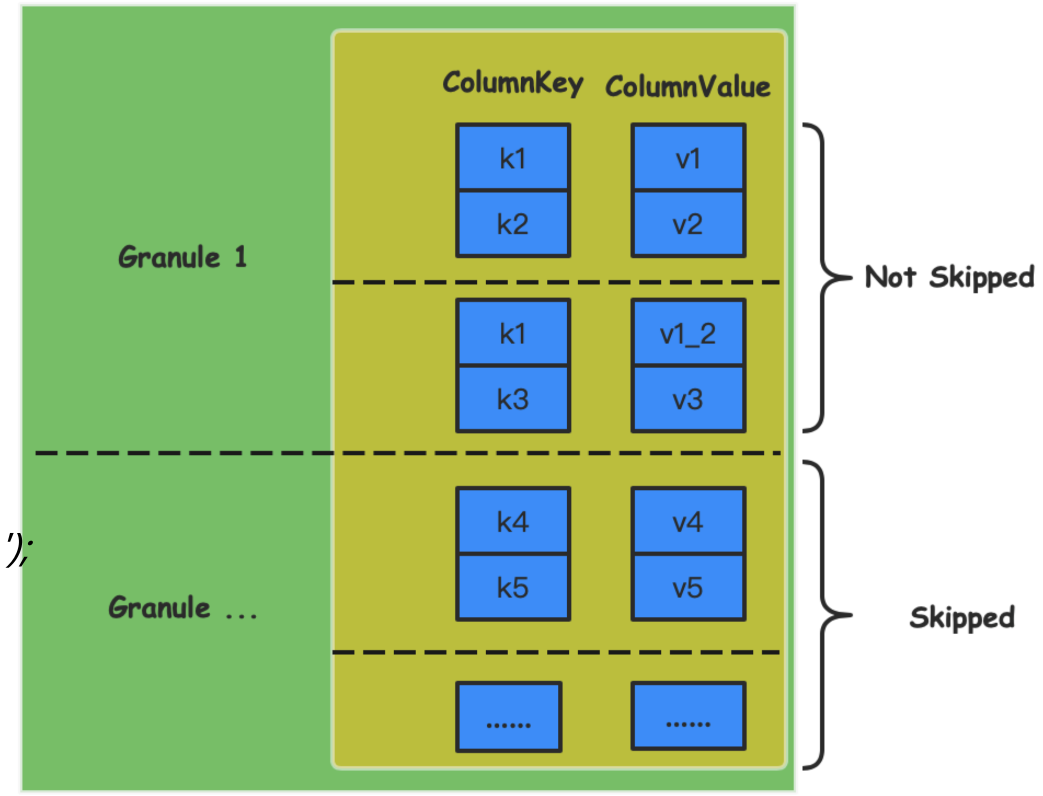
Skip Indices for Map

Skip Indices for Map

```
CREATE TABLE bloom_filter_map
(
    `id` UInt32,
    `map` Map(String, String),
    INDEX map_index map TYPE tokenbf_v1(128, 3, 0) GRANULARITY 1
)
ENGINE = MergeTree
ORDER BY id
SETTINGS index_granularity = 2


SELECT map['k1'] FROM bloom_filter_map WHERE mapContains(map, 'k1');
```

Avoid reading unnecessary granules

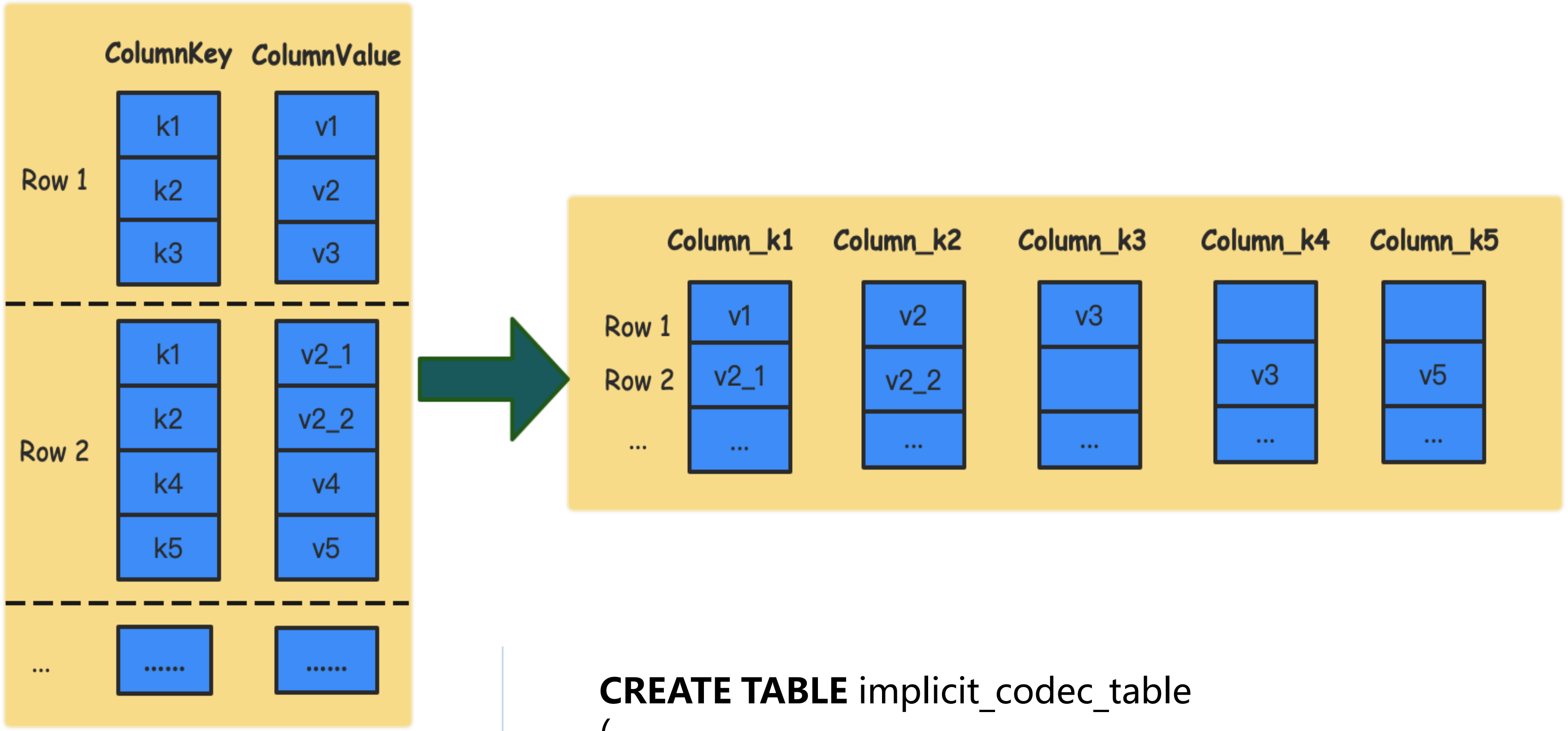
Still read all elements in map



Implicit Map

What is Implicit Map?

Extract every uniques key as a base column. This process is transparent to the users, so we called it as implicit columns.



```
CREATE TABLE implicit_codec_table
(
    `id` UInt32,
    `map` MapV2(String, Nullable(String)) CODEC(ZSTD(1)),
    `map2` MapV2(String, Nullable(String))
)
ENGINE = MergeTree
ORDER BY id
SETTINGS index_granularity = 8192
```

Implicit Map

Performance comparison

The data of two table is totally same

- Bytes on disk

Table Name	Rows	Bytes on disk
normal_map_table	10,000,000	3.19 GiB
implicit_map_table	10,000,000	3.11 GiB

- Query specified key

Table Name	Elasped	Processed	RPS
normal_map_table	10.460 sec	5.07 GB	956.04 thousand rows/s 484.87 MiB/s
implicit_map_table	1.450 sec	551.34 MB	16.89 million rows/s 380.14 MiB/s.

Table DDL:

```
CREATE TABLE test.implicit_map_table
(
    `id` UInt64,
    `map` MapV2(String, String)
)
ENGINE = MergeTree
ORDER BY id
```

Query:

```
SELECT uniqExact(op)
FROM (
        SELECT map['OnePiece'] AS op
        FROM
        normal_map_table
)
```

Implicit Map

select all for Implicit Map

We have tried:

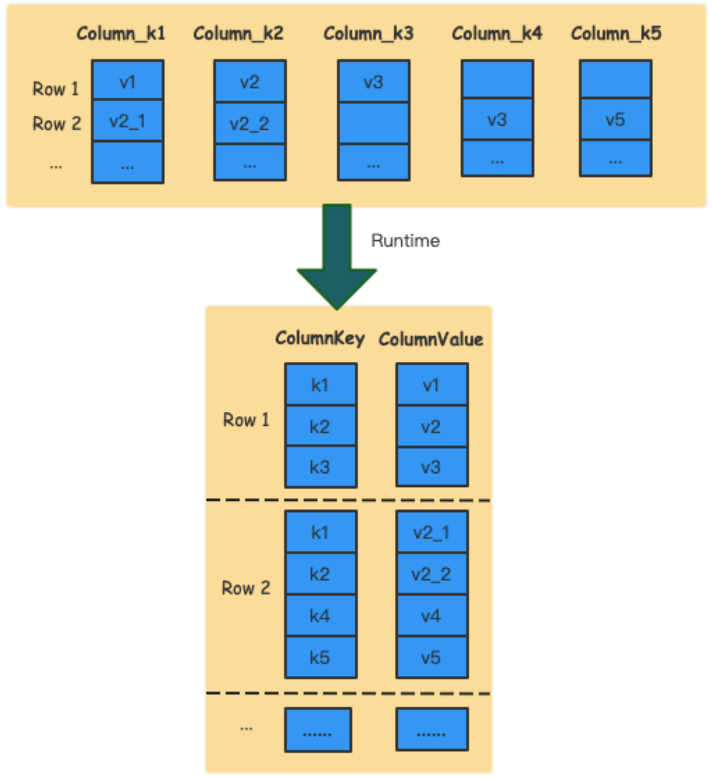
- Construct Map data from all implicit columns during query process
- We create implicit columns and also store native map

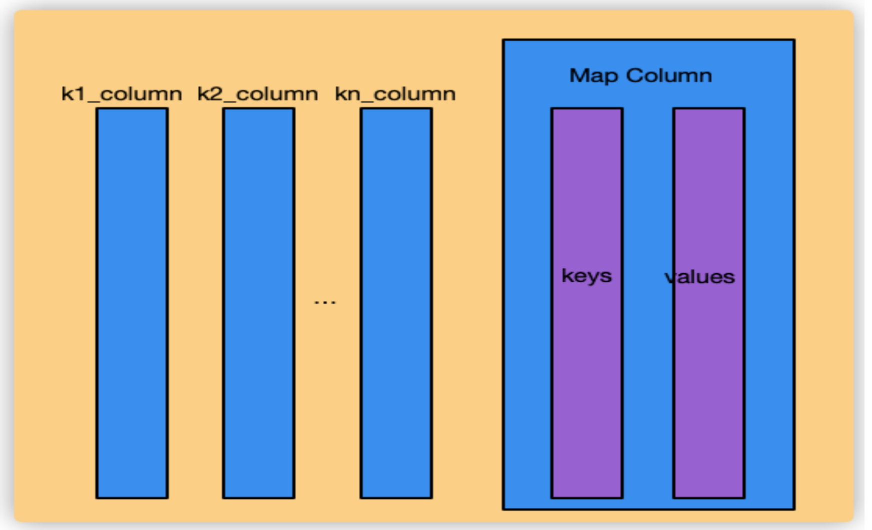
  Query with specified key will read implicit column.
  If users want get whole map, it will read native map column as usual.
- Specify every key

  SELECT map FROM map_table

  SELECT map['k1'], map['k2'] ... map['kn'] FROM map_table

  We can get all implicit column name from system.parts.





Thanks