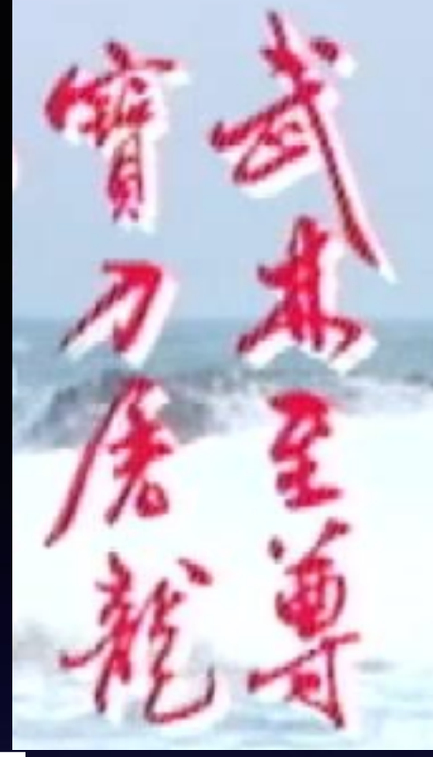


ClickHouse
大数据分析的屠龙刀

尚书杰

目录

- 快到什么程度（Benchmark）
- 为什么快
  - 存储
  - 计算
- 怎么才能快/最佳实践

About me

- 2010 - 2014 云平台/云存储
  - EBS / S3
  - CDN / MQ
  - Cloudfoudry
  - hadoop / hbase / cassandra
- 2014 - 至今 大数据
  - Kafka
  - spark(streaming, sql) / Hadoop / hive
  - Greenplum / HAWQ / Postgres
  - Druid
  - ETL
- Now, I use CLICKHOUSE

How fast - Benchmark 1

- 1.1 billion taxi rides benchmark
  - http://tech.marksblogg.com/benchmarks.html

How fast - benchmark 2

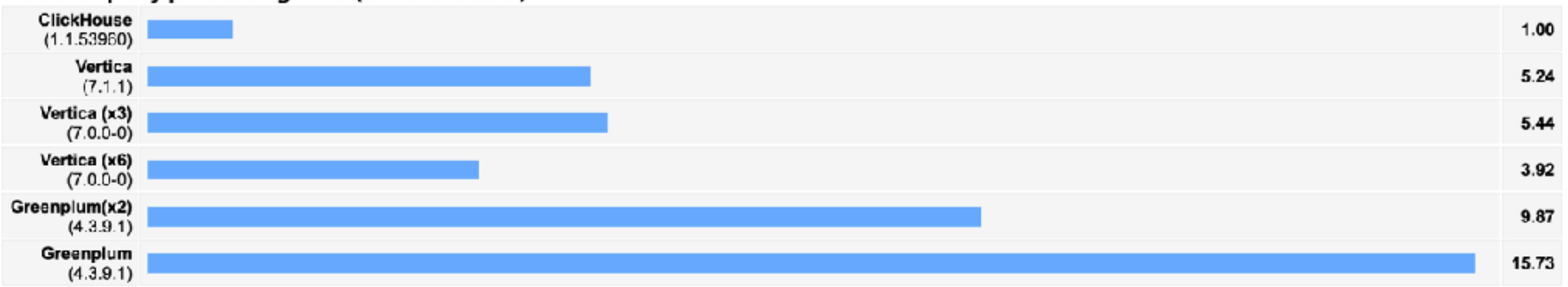
- https://clickhouse.yandex/benchmark.html
- Select AggregateFunction, fields from Table group by xxx having xxx
- 性能甚至好于成熟商业数据库(vertica, greenplum, redshift, …)

Compare: ClickHouse | ClickHouse, new | Vertica | Vertica (x3) | Vertica (x6) | InfiniDB | MonetDB | Infobright | Hive | MySQL | MemSQL | Greenplum(x2) | Greenplum

Dataset size: 10 mln. | 100 mln. | 1 bn.

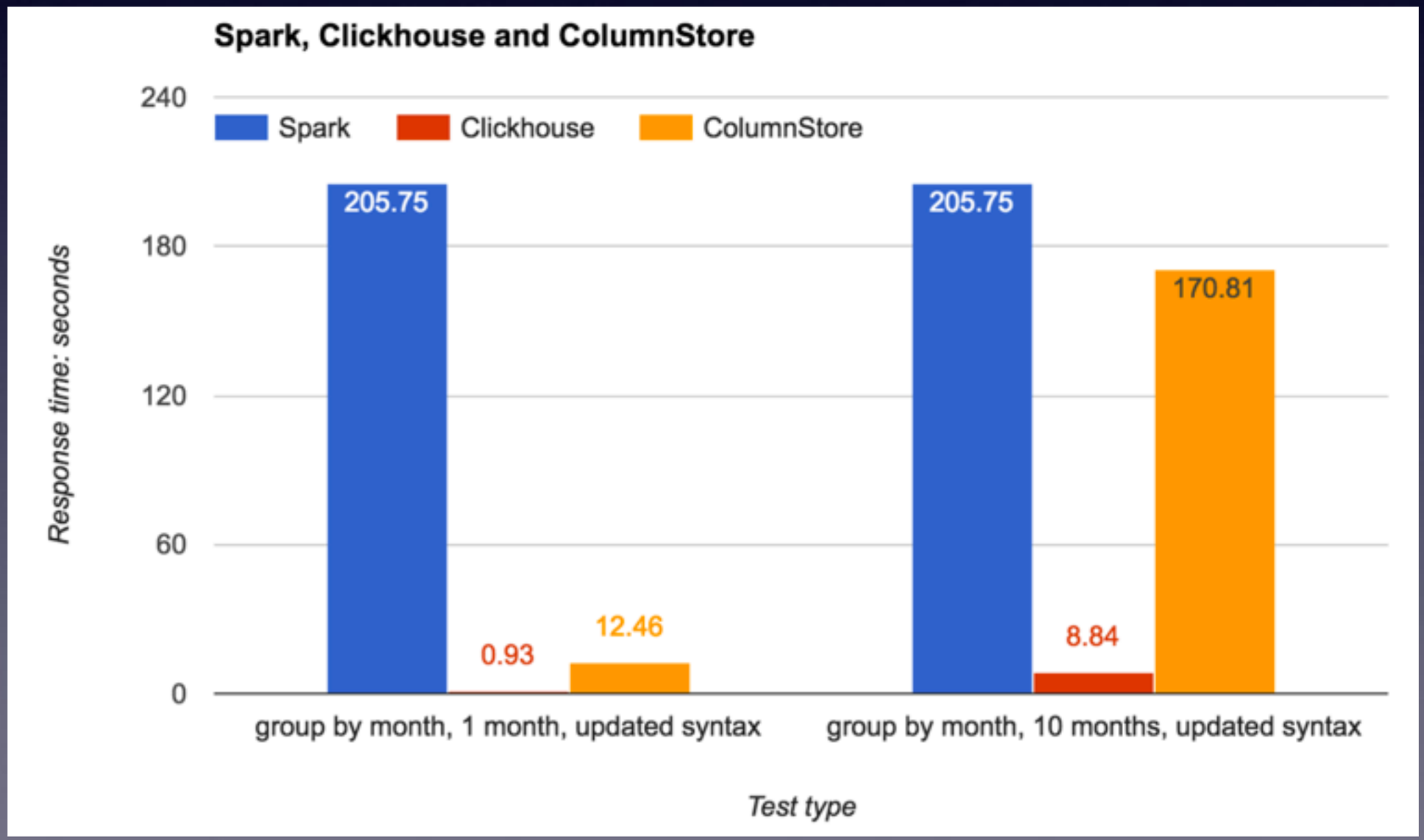
Run number: first (cold cache) | second | third

Relative query processing time (lower is better):



How fast

- https://www.percona.com/blog/2017/03/17/column-store-database-benchmarks-mariadb-columnstore-vs-clickhouse-vs-apache-spark/



Who use it

- 成熟度高
  - 数百家公司使用
  - 中国许多小集群, 遍地开花
  - 国外许多集群, 重度使用
    - Yandex ~ 500 nodes
    - 每天百亿非常常见
      - https://blog.cloudflare.com/http-analytics-for-6m-requests-per-second-using-clickhouse/ (2018.5)
        - It is blazing fast, linearly scalable, hardware efficient, fault tolerant, feature rich, highly reliable, simple and handy
  - Yandex已经上云,对外服务
- 英文
  - 代码，注释，文档
- 适用于
  - 多维分析
  - 事件分析
  - 宽表分析

- Yandex: 500+ servers, 25B rec/day
- LifeStreet: 60 servers, 75B rec/day
- CloudFlare: 36 servers, 200B rec/day
- Bloomberg: 102 servers, 1000B rec/day

Who use it

2017

Cloudflare vs Bloomberg

36	x3	12M+	50Gbit+	4PB+
Nodes	Replication factor	Row Insertion/s	Insertion Throughput/s	Raid-0 Spinning Disks

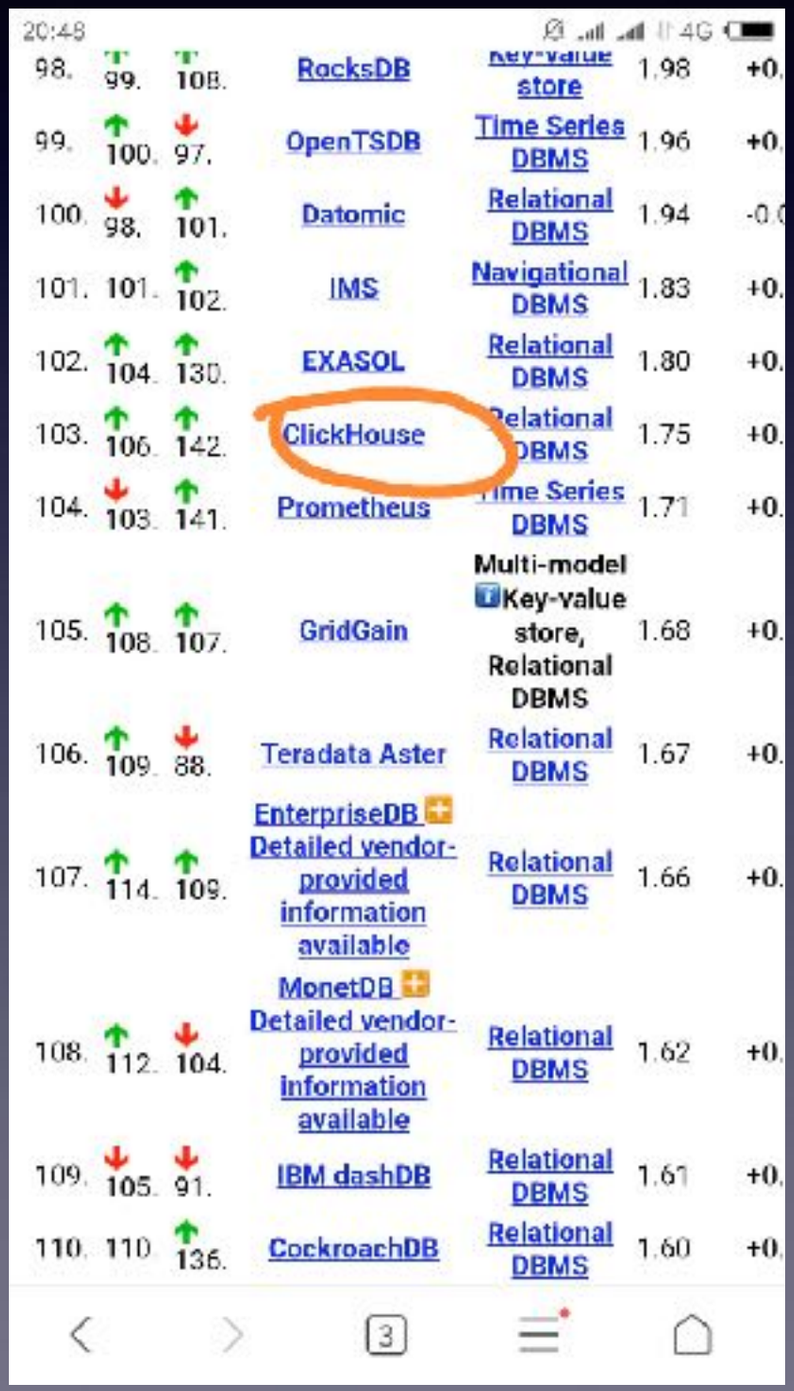
Each node:

- → **CPU** - 40 logical cores E5-2630 v3 @ 2.40 GHz
- → **RAM** - 256 GB RAM
- → **Disks** - 12 x 10 TB Seagate ST10000NM0016-1TT101
- → **Network** - 2 x 25G Mellanox ConnectX-4

102	x3	60M+	80Gbit+	1PB+
Nodes	Replication factor	Row Insertion/s	Insertion Throughput/s 42 fields of netflow data	NVMe SSDs

Who use it

- 2017.10, 142 -> 2018.10, 103

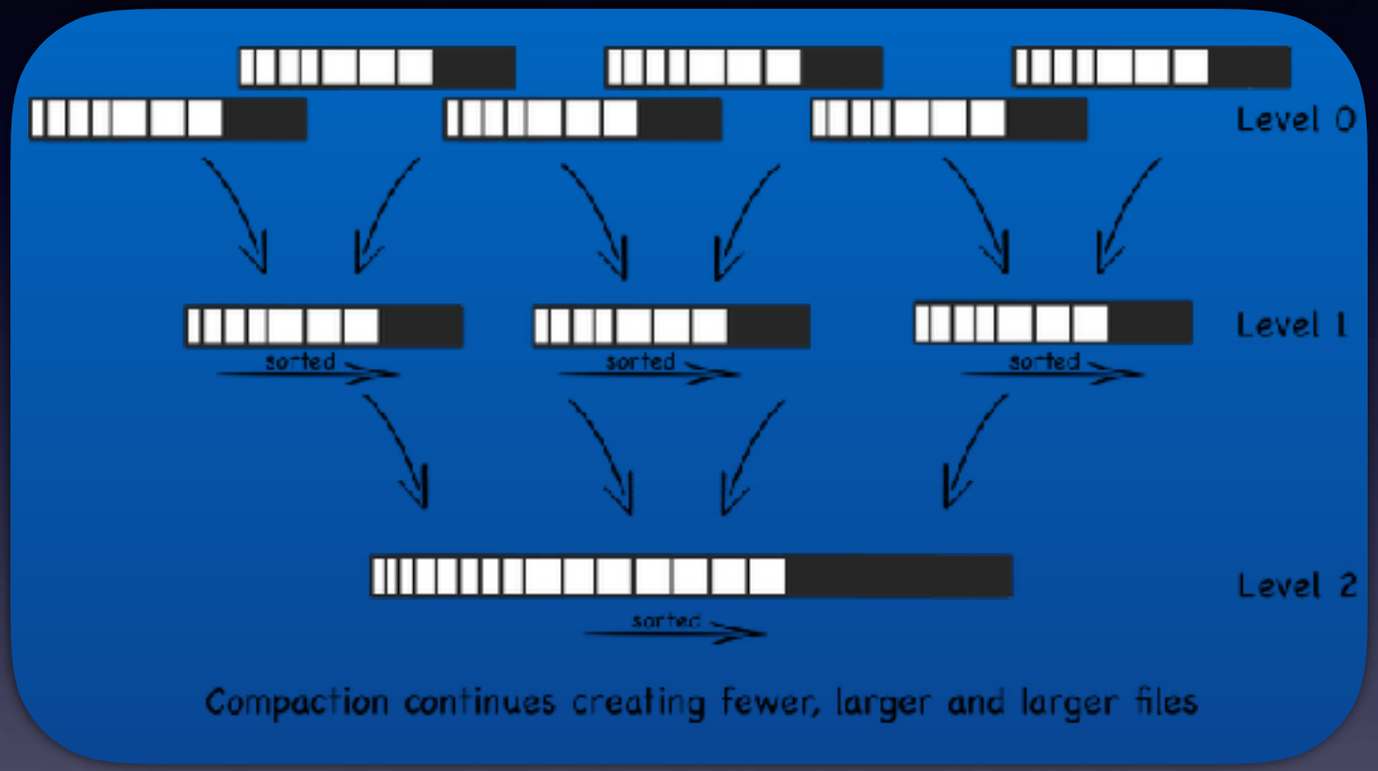


Why fast

- StorageEngine
    - Pure column based storage
    - LSM-Tree Family
        - Max-min index
    - Partition
        - Sharding / Distributed table
- ExecuteEngine
    - Multi-threads / Multi-nodes
    - Vector engine
        - Processed by block, not line
    - SIMD enhancement
    - LLVM enhancement
- C++

MergeTree： LSM-Tree

- MergeTree

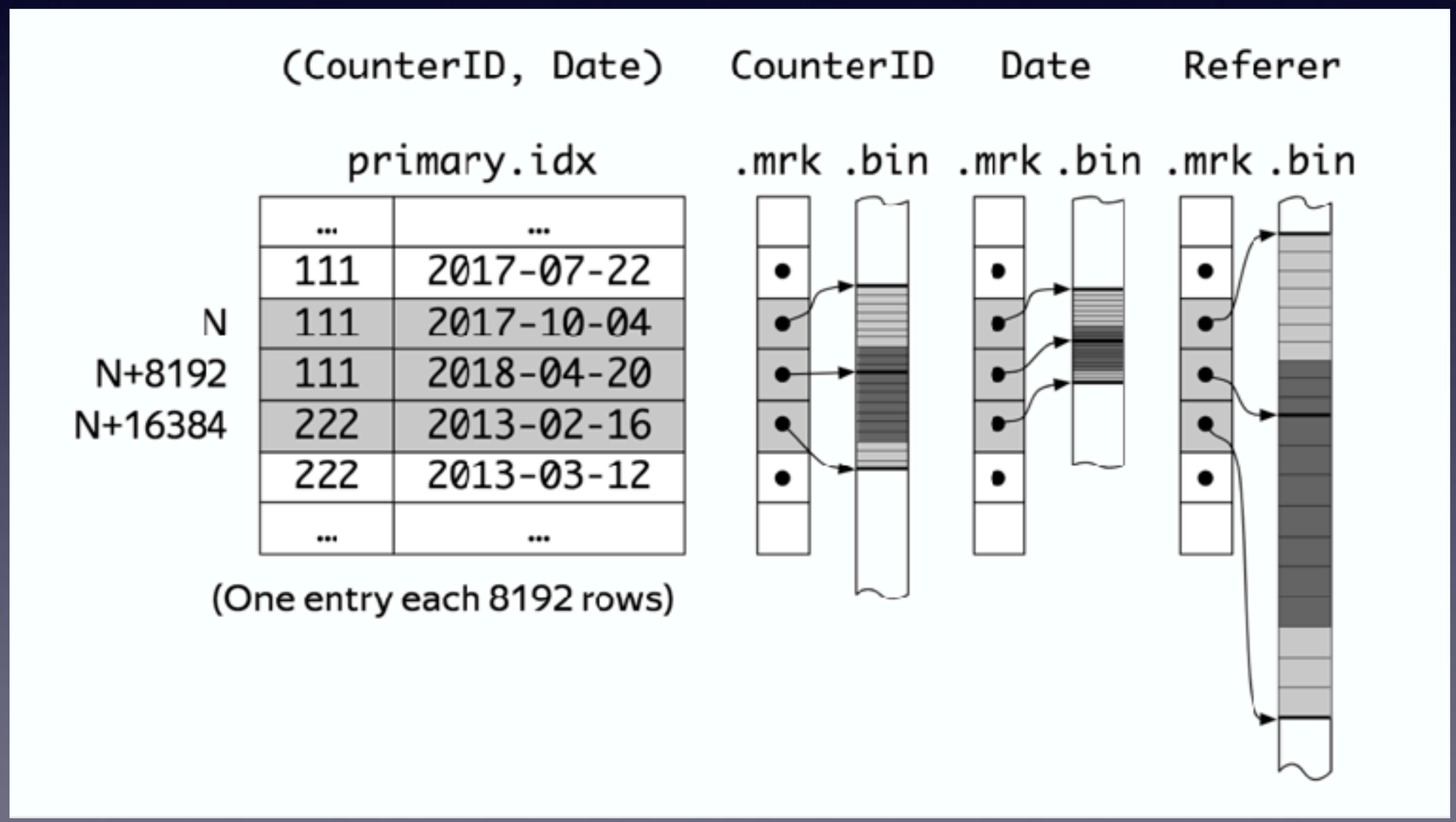


- Each folder is a part
  - Partition_StartBlock_EndBlock_MergeTreeLevel
- Part Tree (MergeTree)
  - 2 partitions, 17802, 17803

```
17802_2_350_3
17802_351_362_1
17802_363_373_1
17802_375_385_1
17802_387_387_0
17802_390_390_0
17802_391_391_0
17802_393_393_0
17802_395_395_0
17803_1_354_3
17803_356_366_1
17803_367_378_1
17803_380_389_1
17803_392_392_0
17803_394_394_0
17803_396_396_0
```

How to: MergeTree

- Data In file



Why fast: MergeTree

- LSM-Tree
  - Buffer + MergeTree
  - Primary key block index
- Chosen primary key
  - 此主键非mysql主键, 不唯一，用于排序
  - 经常查询的key作为主键
  - 多主键联合索引
  - 主键不要太碎, no timestamp, eg. toHour

Why fast: Vector Enging

- Vector Engine
  - data processed by each block, not each line
- SIMD加速
- LLVM加速
  - set compile_expressions=1

Why fast:极致代码

- 方法一： ‘B’ = (‘b’ - ‘a’) + ‘A’
- 方法二：’B’ = ‘b’ ^ (‘A’ ^ ‘a’)

```
const auto flip_case_mask = 'A' ^ 'a';
for (; src < src_end; ++src, ++dst)
    if (*src >= not_case_lower_bound && *src <= not_case_upper_bound)
        *dst = *src ^ flip_case_mask;
    else
        *dst = *src;
```

- 方法三：SIMD

```
/// load 16 sequential 8-bit characters
const auto chars = _mm_loadu_si128(reinterpret_cast<const __m128i *>(src));

/// find which 8-bit sequences belong to range [case_lower_bound, case_upper_bound]
const auto is_not_case
    = _mm_and_si128(_mm_cmpgt_epi8(chars, v_not_case_lower_bound), _mm_cmplt_epi8(chars, v_not_case_upper_bound));

/// keep `flip_case_mask` only where necessary, zero out elsewhere
const auto xor_mask = _mm_and_si128(v_flip_case_mask, is_not_case);

/// flip case by applying calculated mask
const auto cased_chars = _mm_xor_si128(chars, xor_mask);

/// store result back to destination
_mm_storeu_si128(reinterpret_cast<__m128i *>(dst), cased_chars);
```

Why fast: parallel processing

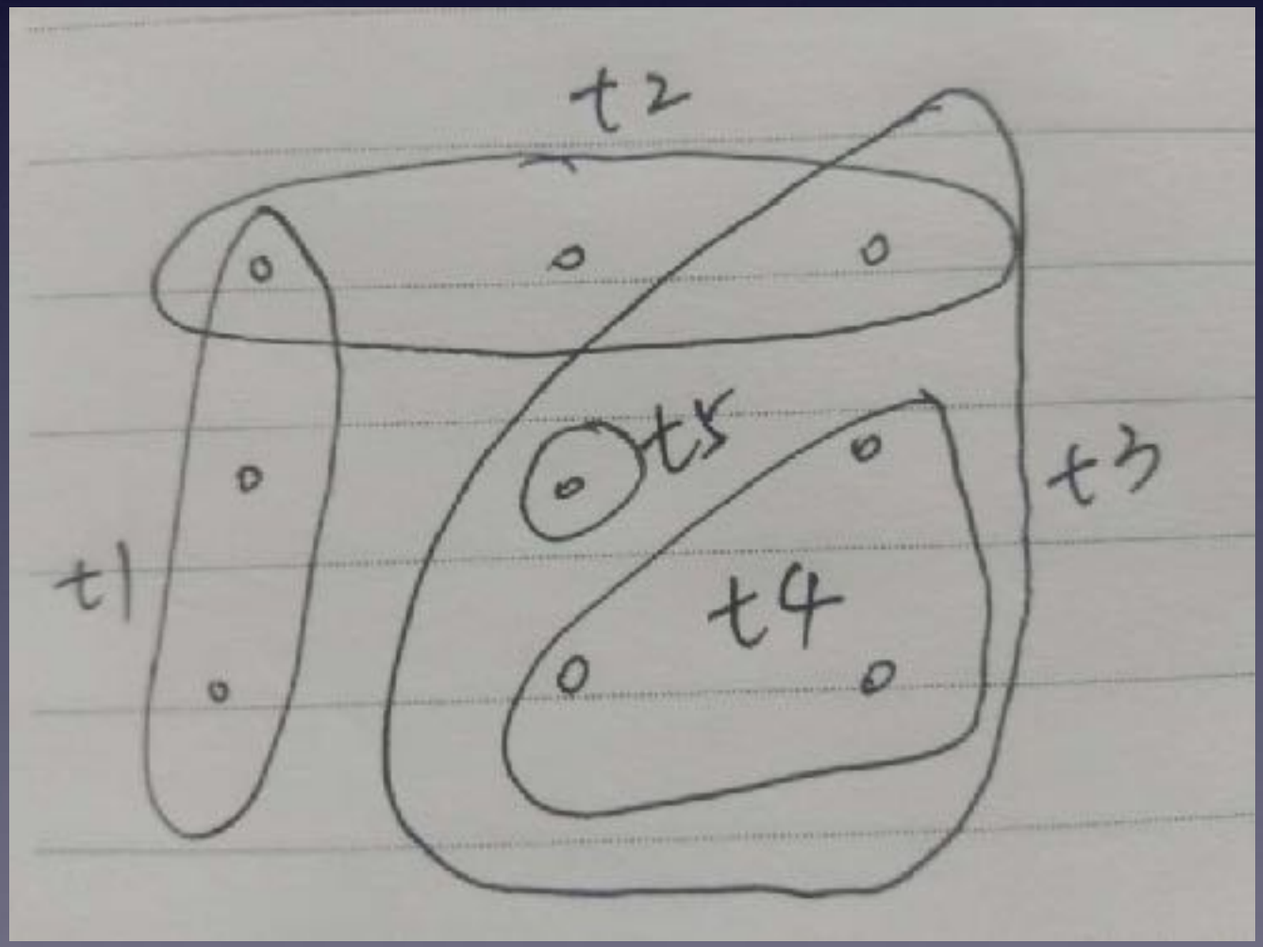
- Multi-threads
  - Cpu 100%
    - set max-threads
- Multi-nodes
  - Liner scalability
    - shard by user_id for event_processing

How to: Select

```
SELECT [DISTINCT] expr_list
    [FROM [db.]table | (subquery) | table_function] [FINAL]
    [SAMPLE sample_coeff]
    [ARRAY JOIN ...]
    [GLOBAL] ANY|ALL INNER|LEFT JOIN (subquery)|table USING
columns_list
    [PREWHERE expr]
    [WHERE expr]
    [GROUP BY expr_list] [WITH TOTALS/ROLLUP/CUBE]
    [HAVING expr]
    [ORDER BY expr_list]
    [LIMIT [n, ]m]
    [UNION ALL ...]
    [INTO OUTFILE filename]
    [FORMAT format]
    [LIMIT n BY columns]
```

Cluster

- Cluster is just a route map, defined in config file
- Local table / Distributed table(cluster)



Local table/ Distributed table

- Cluster
  - Like router table
- Distributed table
  - Radom distributed
  - Hash, hash-key, only work when inserting
- On cluster 语句
  - 方便运维

How to: replica

- 主主复制
- Zookeeper
- Re-insert
  - Just retry

Sharding

- Hash sharding
- Fully copy sharing
- Co-locate join
- …

How to: 配置

- https://clickhouse.yandex/docs/en/single/#usage-recommendations
- Raid
    - 加速io
    - 提升容量
- SSD
- 大内存
- 多核

How to: 并行导入

- CreateTable
  - String => int, StringWithDictionary
- 多个命令行导入
  - Hadoop fs -cat xxxx | clickhouse-client --host=xxx --query="INSERT INTO criteo_log FORMAT TabSeparated”（CSV, JSONEachRow)
    - input_format_allow_errors_num / input_format_allow_errors_ratio
  - json ~ 30w/s per node, 约100字段
  - Csv ~ 50w/s per node
  - Default value, visitParamExtractString
  - TmpTable/Attach Partition / Detach Partition
  - ReplicateTable: just retry
- 压缩率高
  - 与原始txt文件相比, 30~50倍以上very easy
  - 优于hadoop文本文件存储，snappy
  - 优于parquet

How to: 大宽表模型

- Less join
  - Depend on ETL
- Wide table + dictionary
- In/global In
- join/global join

抽样大法好

- 自带抽样，Sample语句
- 近似计算
  - HyperLogLog
  - Uniq
  - Quantiles

物化视图好

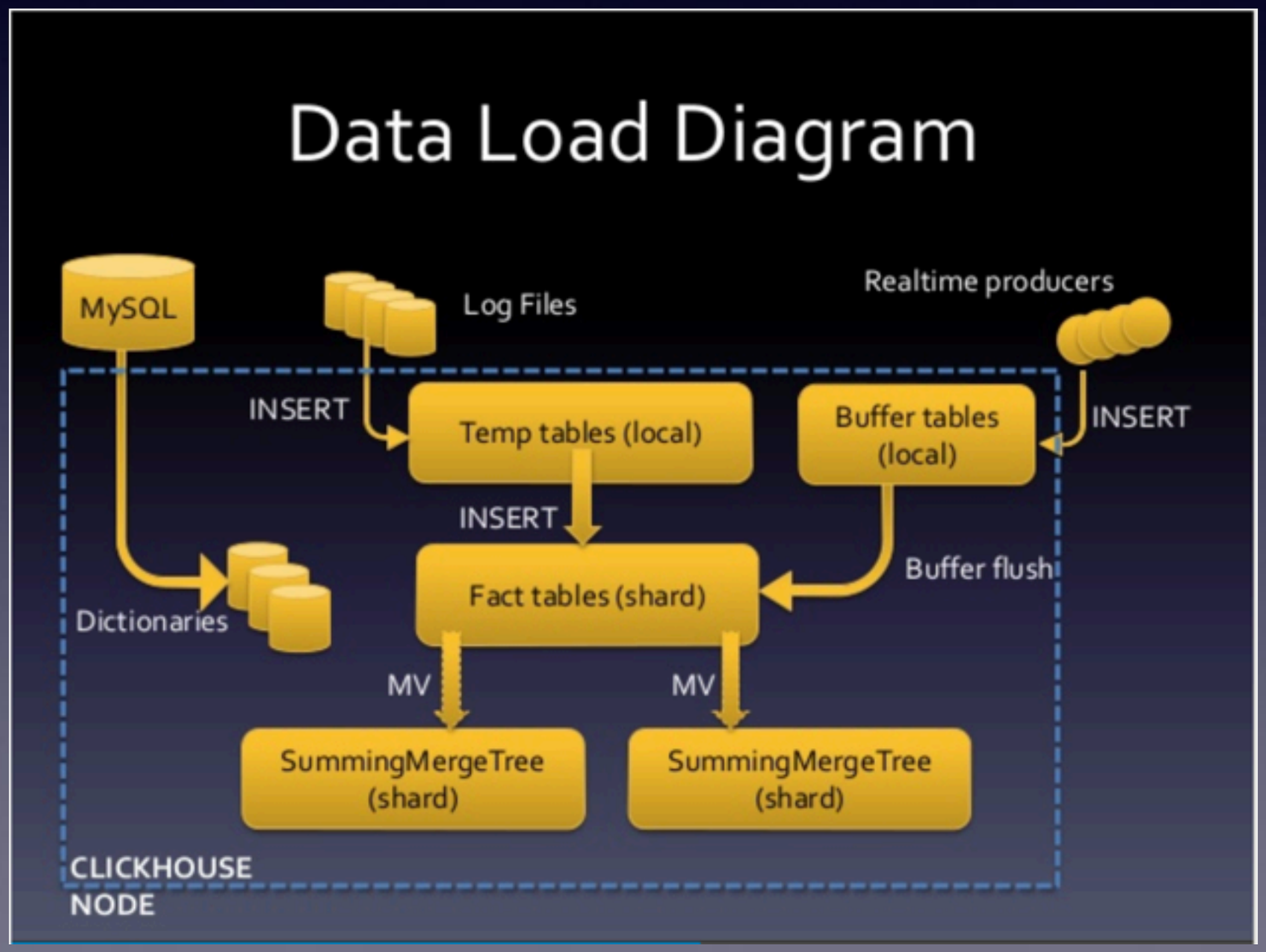
- 级联计算
- Rollup
- Cube

Working with Others

- Kafka Engine
- Mysql
- ODBC
  - Oracle, sqlserver, postgres, …
- MongoDB

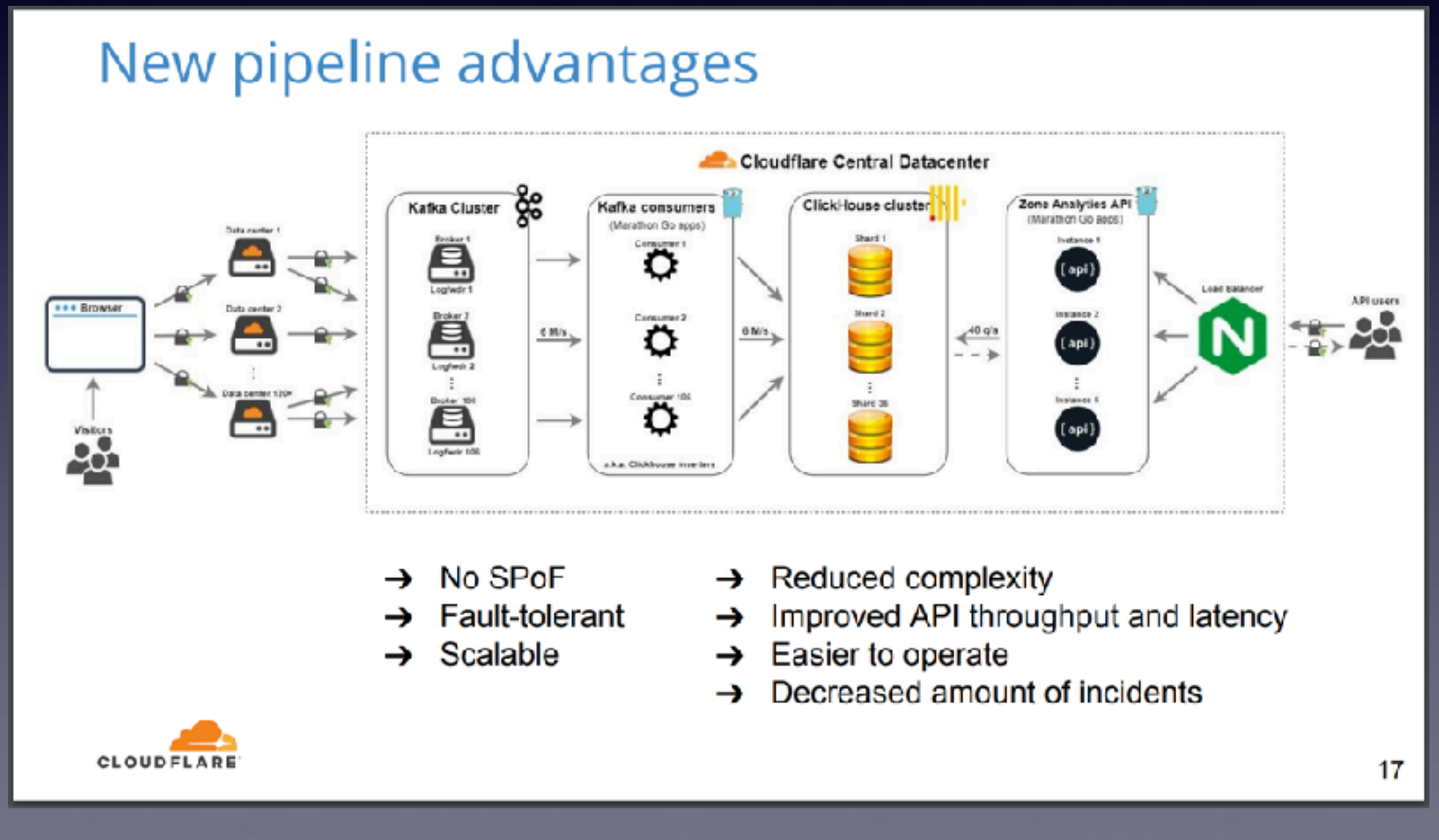
DataFlow Example 1

- Fact table + MV



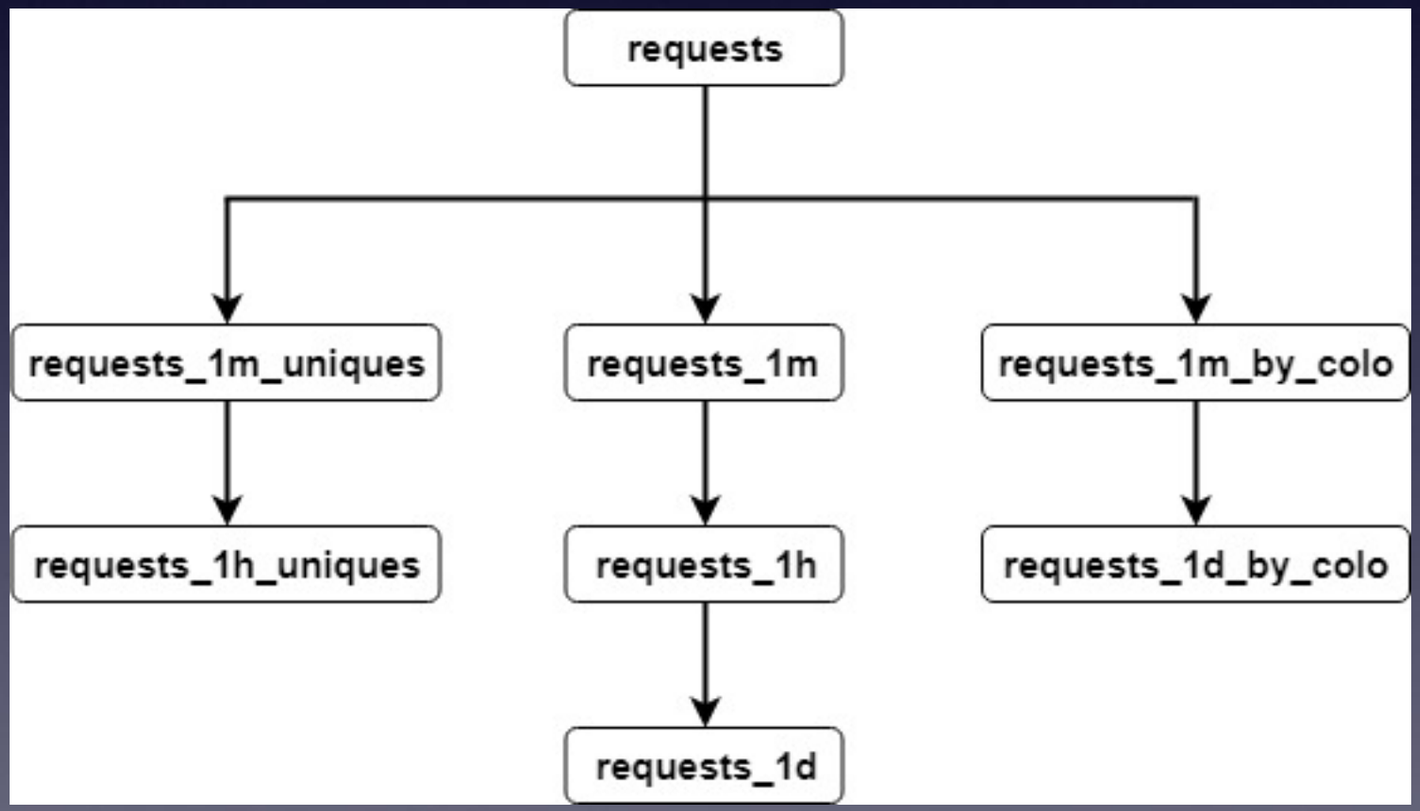
DataFlow Example 2

- KAFKA + CH



DataFlow Example 2

- Table structure



修改数据

- Update / Delete
  - Rewrite whole block
- CollapsingMergeTree

Lambda function

```
select
countIf(has(days, 0)),
countIf(has(days, 0) AND has(days, 1))
from (
    SELECT
    groupUniqArray(date - toDate(‘2018-08-01’)) as days
    from Table1
    where ((date = '2018-08-01') and (gender = 'F')) or
(date = ‘2018-08-02')
    group by user_id
)
```

行列转换

- Array join
- Group array

No complex planner

- Join on
- max_bytes_before_external_group_by
- max_bytes_before_external_sort_by

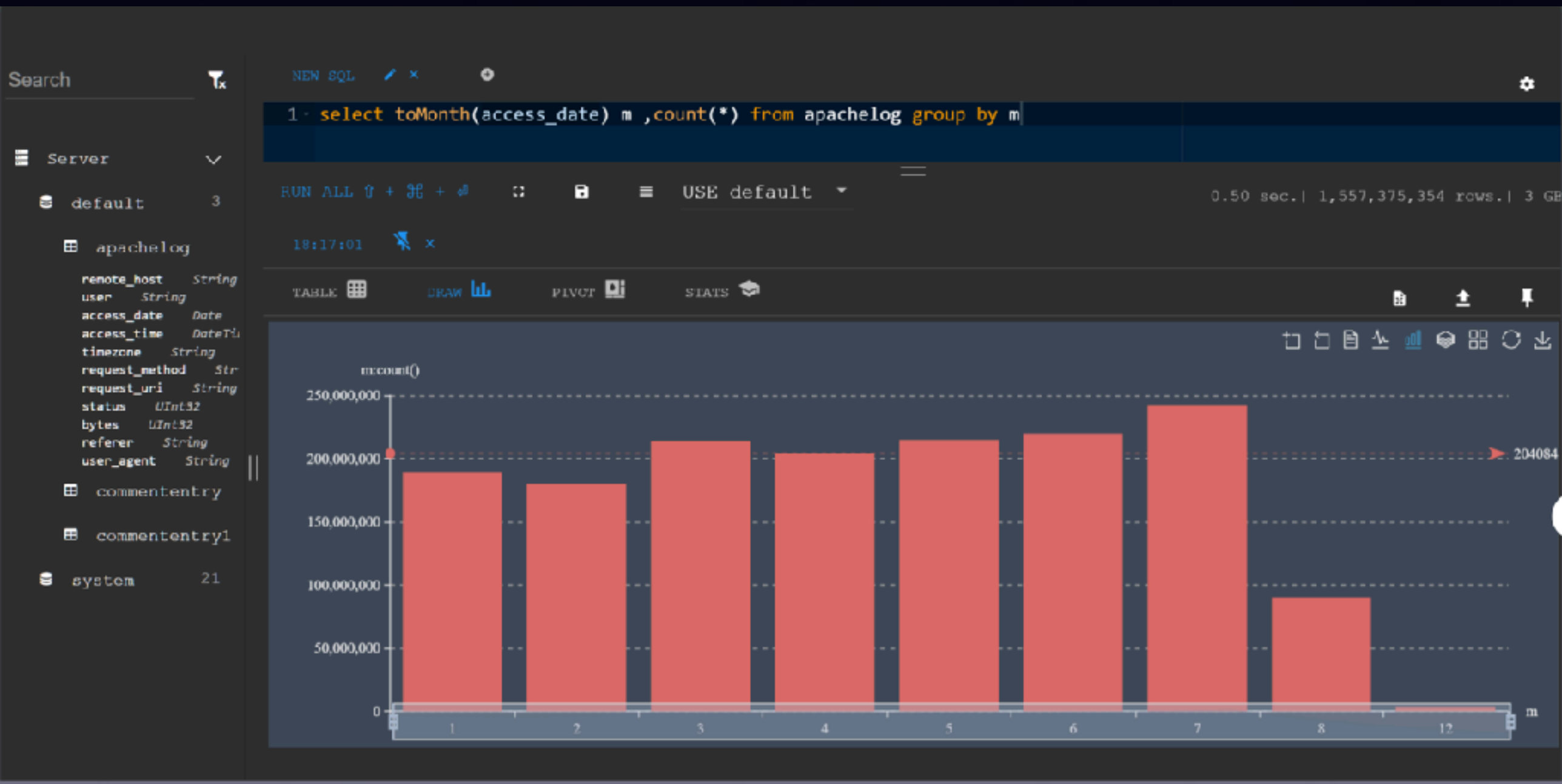
udf/udaf

- windowFunnel
- retention
- sumMap
- topK - 按频率
- uniq
- 地理函数

Ui & 监控: 都是现成的!

- Tabix
  - Config.xml , http_server_default_response
- SuperSet
- Graphite + Grafana, 配置打开
- Zabix

Tabix


Search
NEW SQL
1 select toMonth(access_date) m ,count(*) from apachelog group by m
Server
default 3
apachelog
remote_host String
user String
access_date Date
access_time DateTi
timezone String
request_method Str
request_uri String
status UInt32
bytes UInt32
referer String
user_agent String
commententry
commententry1
system 21
RUN ALL ⇧ + ⌘ + ⏎
USE default
0.50 sec.| 1,557,375,354 rows.| 3 GB
18:17:01
TABLE
DRAW
PIVOT
STATS
mcount()
250,000,000
200,000,000
150,000,000
100,000,000
50,000,000
0
204084
1
2
3
4
5
6
7
8
12
m

升级建议

- 版本帝
  - 永无止境
    - 仍有很大提升空间
- 测好了再升
- 双集群跑一段时间

Q & A