

PROJECTIONS

ClickHouse 最新特性原理解析



快手贡献

郑天祺

数据架构研发工程师

自我介绍

郑天祺 快手科技数据平台研发工程师

- ▶ 中科院计算所系统结构博士
- ▶ ClickHouse Community Member
- ▶ 四年 ClickHouse 研发经验，已贡献合并 300+ PR



- 1 ClickHouse 背景介绍
- 2 Projection 简介与用例
- 3 Projection 原理与实现
- 4 特性对比与生产应用效果



1

ClickHouse 背景介绍

2

Projection 简介与用例

3

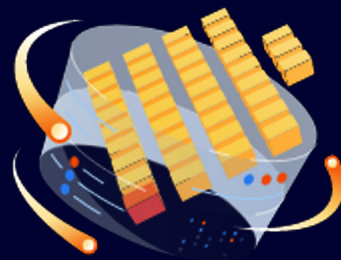
Projection 原理与实现

4

特性对比与生产应用效果

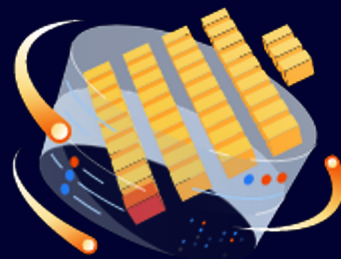
起源

- 诞生于 Yandex.Metrica 团队 (俄罗斯版百度统计)
- 旨在提升网页点击日志分析的性能
- 替换原先的 MySQL MyISAM 引擎



起源

- 诞生于 Yandex.Metrica 团队 (俄罗斯版百度统计)
- 旨在提升网页点击日志分析的性能
- 替换原先的 MySQL MyISAM 引擎

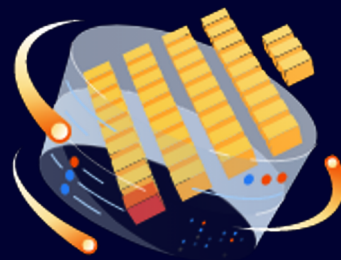


开源

- 2016 年公开代码，性能远超同期开源竞品
- 国内多个互联网企业引入用作 OLAP 引擎
- Roadmap 逐渐转向通用分析型数据库

主要特性（用户导向）

- 高效的数据读写性能（类 LSM 树，列式存储与压缩）
- 高效的数据处理性能（向量化计算，列式组织，指令优化）
- 灵活的计算扩展能力（P2P 分布式架构）



主要特性（用户导向）

- 高效的数据读写性能（类 LSM 树，列式存储与压缩）
- 高效的数据处理性能（向量化计算，列式组织，指令优化）
- 灵活的计算扩展能力（P2P 分布式架构）

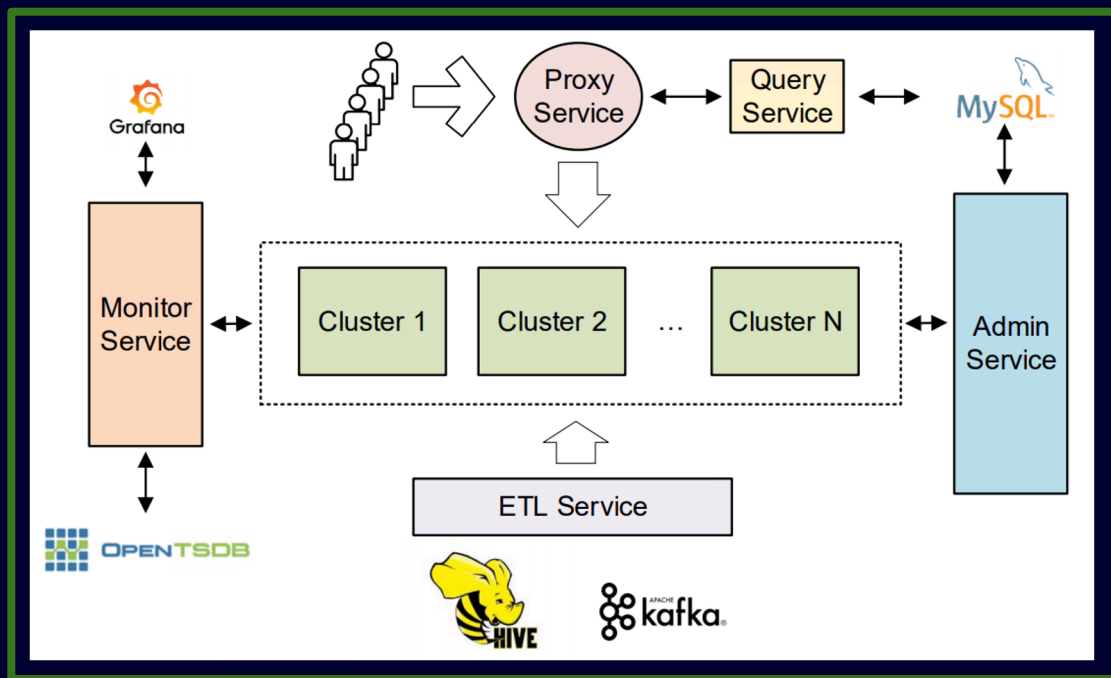


设计理念（研发导向）

- 追求高效的实时写入与交互式查询
- 追求系统设计纯度：计算 SQL 化，存储 Block 化
- “白盒”计算模式，“手动档”应用模式

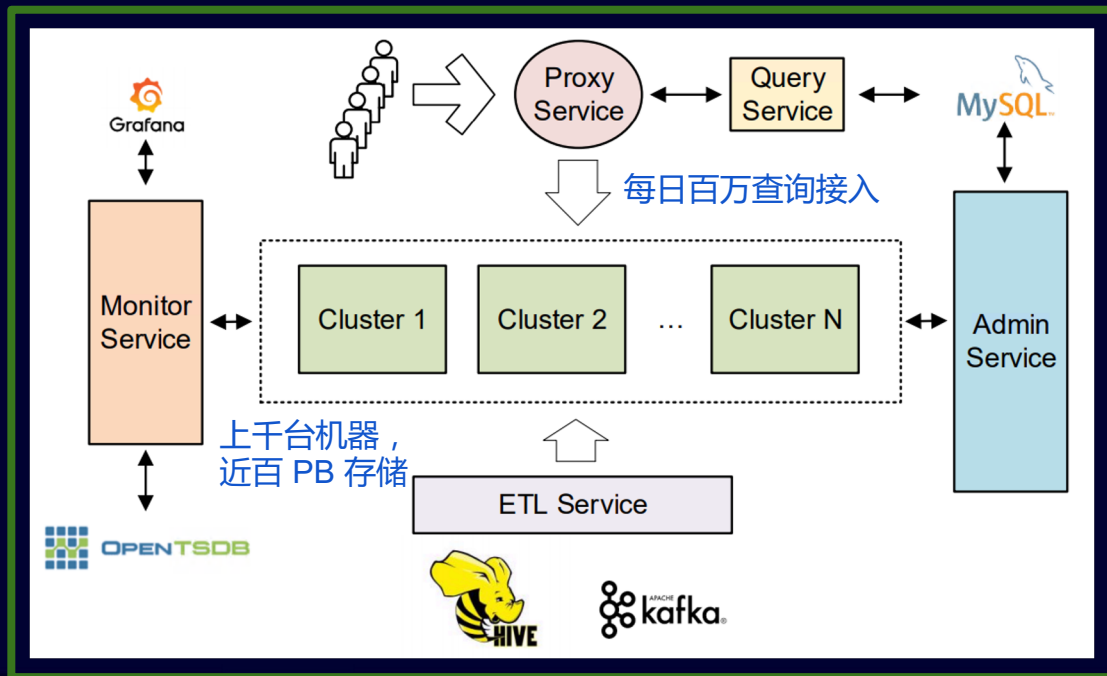
CH 在快手 OLAP

多集群架构，统一的查询与数据接入，完备的监控服务体系



CH 在快手 OLAP

支撑留存计算，AB 实验，音视频分析，风控预警等多个业务



开源竞品对比

强

中

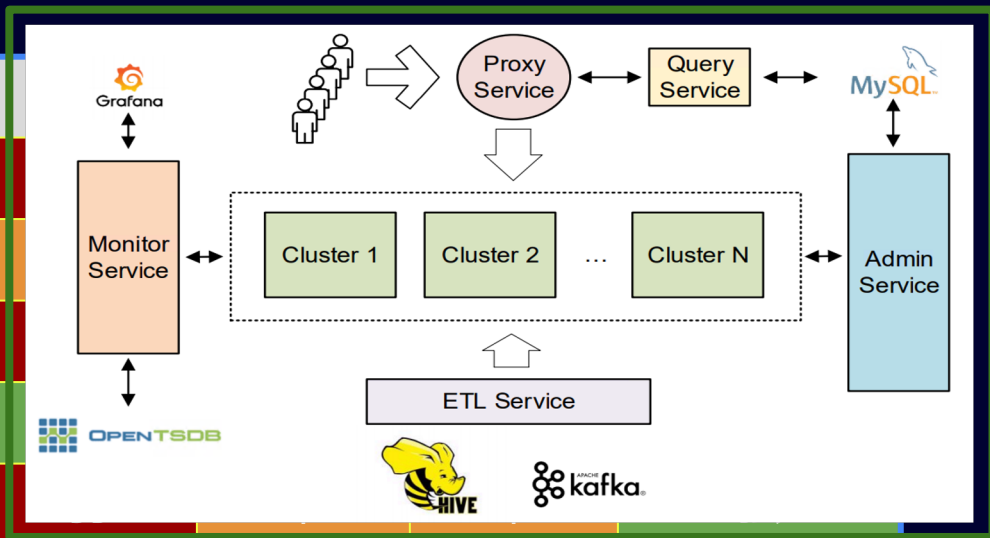
弱

特点	ClickHouse	Druid	Kylin	Doris	Greenplum
明细查询	强	弱	弱	中	中
实时写入	强	中	弱	弱	中
分析功能	强	弱	弱	中	强
物化存储	弱	强	强	中	中
事务特性	弱	弱	中	中	强
管控能力	弱	强	强	中	中

开源竞品对比

通过完善的架构设计与外围设施，取得了不俗的管控能力

特点	ClickHouse				
明细查询	强				
实时写入	强				
分析功能	强				
物化存储	弱				
事务特性	弱				
管控能力	强*	强	强	中	中



开源竞品对比

事务往往伴随性能开销，在互联网分析场景中非高优

特点	ClickHouse	Druid	Kylin	Doris	Greenplum
明细查询	强	弱	弱	中	中
实时写入	强	中	弱	弱	中
分析功能	强	弱	弱	中	强
物化存储	弱	强	强	中	中
事务特性	弱	弱	中	中	强
管控能力	强*	强	强	中	中

开源竞品对比

物化是 OLAP 分析强需求，亟需解决

特点	ClickHouse	Druid	Kylin	Doris	Greenplum
明细查询	强	弱	弱	中	中
实时写入	强	中	弱	弱	中
分析功能	强	弱	弱	中	强
物化存储	弱	强	强	中	中
事务特性	弱	弱	中	中	强
管控能力	强*	强	强	中	中

ClickHouse 按照类 LSM 树的结构存储数据 (MergeTree Family)

ClickHouse 按照类 LSM 树的结构存储数据 (MergeTree Family)

- 仅支持一种列排序方式
 - *ORDER BY (author_id, photo_id)* 无法优化基于 *photo_id* 的查询
 - Z-Curve 索引支持近邻查询，但索引效率整体下降 (仍在开发中)
 - Skip-index 在召回率高的数据分布中表现很差

ClickHouse 按照类 LSM 树的结构存储数据 (MergeTree Family)

- 仅支持一种列排序方式
 - `ORDER BY (author_id, photo_id)` 无法优化基于 `photo_id` 的查询
 - Z-Curve 索引支持近邻查询，但索引效率整体下降（仍在开发中）
 - Skip-index 在召回率高的数据分布中表现很差
- OLAP 预聚合模型需要手动参与设计（“手动档”）
 - 预聚合存储 AggregatingMergeTree 仅支持一种预聚合方式
 - 查询需要改写方可使用预聚合数据，用户体验差
 - 明细数据不复存在，无法解决实时明细混合分析需求

ClickHouse 按照类 LSM 树的结构存储数据 (MergeTree Family)

- 仅支持一种列排序方式
 - `ORDER BY (author_id, photo_id)` 无法优化基于 `photo_id` 的查询
 - Z-Curve 索引支持近邻查询，但索引效率整体下降（仍在开发中）
 - Skip-index 在召回率高的数据分布中表现很差
- OLAP 预聚合模型需要手动参与设计（“手动档”）
 - 预聚合存储 AggregatingMergeTree 仅支持一种预聚合方式
 - 查询需要改写方可使用预聚合数据，用户体验差
 - 明细数据不复存在，无法解决实时明细混合分析需求
- ClickHouse 物化视图无一致性保证

PROJECTIONS

Projection 使用通用的机制完备地解决了前述三大问题：

1. Projection 可按照不同的列进行数据重排
2. Projection 可使用聚合查询直接定义预聚合模型
3. Projection 查询分析能自动选择最优 Projection 进行查询优化，无需改写查询
4. Projection 在任一时刻针对任一数据变换操作均提供一致性保证

Feel Brave ?

已合并进 master 分支，预计 21.6 release，是 ClickHouse 截止目前最高赞 Pull Request
已上生产三月有余，欢迎尝试 <https://github.com/ClickHouse/ClickHouse/pull/20202>

The screenshot shows a GitHub Pull Request (PR) #20202 titled "Projections" in the ClickHouse repository. The PR is merged and has 34 commits. The search bar at the top is set to "is:pr sort:reactions+1-desc". The PR list on the left highlights the "Projections" PR. The right panel shows the PR details, including the CLA agreement and changelog entry.

Filters **Clear current search query, filters, and sorts**

207 Open ✓ 15,996 Closed Author Label

Projections doc-alert force tests pr-feature
#20202 by amosbird was merged 6 days ago 2 of 4

Fixed warnings found by PVS-Studio (demo scan). doc-alert
#1204 by alexey-milovidov was merged on Sep 6, 2017

Initial explain doc-alert pr-feature
#11873 by KochetovNicolai was merged on Jul 8, 2020 Approved

S3 zero copy replication doc-alert pr-feature
#16240 by ianton-ru was merged on Mar 14 Approved

Add new DataType Map(key,value) can be tested doc-alert force tests pr-feature
#15806 by hexiaoting was merged on Dec 16, 2020

Polymorphic parts (compact format). doc-alert pr-feature
#8290 by CurtizJ was merged on Feb 23, 2020 Approved

Polymorphic parts (in-memory format) doc-alert pr-feature st-waiting-for-fix
#10697 by CurtizJ was merged on Jul 8, 2020 Approved

Projections #20202

Merged KochetovNicolai merged 34 commits into ClickHouse:master from amosbird:projection 6 days ago

Conversation 9 Commits 34 Checks 1 Files changed 148

amosbird commented on Feb 8 • edited Member

I hereby agree to the terms of the CLA available at: <https://yandex.ru/legal/cla/?lang=en>

Changelog category (leave one):

- New Feature

Changelog entry (a user-readable short description of the changes that goes to CHANGELOG.md):

Add projection support for MergeTree* tables.

Detailed description / Documentation draft:

The documentation is almost listed in #14730

Possible TODOs.

- ☒ Complete query analysis for projections.
- ☒ Normal projections with different order by clause
- ☐ ProjectionMergeTree-to-store-projection-data-only can be done via CollapsingMergeTree with default sign = 1
- ☐ Secondary indices via projection

27 👍 9 ❤️



- 1 ClickHouse 背景介绍
- 2 Projection 简介与用例
- 3 Projection 原理与实现
- 4 特性对比与生产应用效果

Projection 概念

引自 C-Store 文章，一作 Mike Stonebraker，
数据库系统图灵奖获得者，列存数据库 Vertica 之父

The logo for Vertica, consisting of the word "VERTICA" in a bold, white, sans-serif font, set against a black rectangular background.

*Hence, C-Store physically stores a collection of columns, each sorted on some attribute(s). **Groups of columns sorted on the same attribute are referred to as "projections"**; the same column may exist in multiple projections, possibly sorted on a different attribute in each.*

Projection 概念

- Projection 概念由 C-Store 提出，并在 Vertica 数据库中落地发展 (不同于 SQL 中的 Projection 运算)
 - Projections 是一组列的集合，使用建表语句定义
 - Projections 按照不同的顺序存储数据，用以优化多样的查询
 - Vertica 扩展 Projection 支持使用部分聚合函数进行数据上卷优化

Projection 概念

- Projection 概念由 C-Store 提出，并在 Vertica 数据库中落地发展 (不同于 SQL 中的 Projection 运算)
 - Projections 是一组列的集合，使用建表语句定义
 - Projections 按照不同的顺序存储数据，用以优化多样的查询
 - Vertica 扩展 Projection 支持使用部分聚合函数进行数据上卷优化
- ClickHouse Projection 沿袭该设计并进行优化
 - 支持使用任意函数存储并自由组合参与数据上卷运算
 - 支持 Projection 与原始明细表的联合查询

Projection 用例介绍

Projection 用例

使用简化的视频分析日志表进行演示

```
CREATE TABLE video_log
(
  `datetime` DateTime, -- 20,000 records per second
  `user_id` UInt64, -- Cardinality == 100,000,000
  `device_id` UInt64, -- Cardinality == 200,000,000
  `domain` LowCardinality(String), -- Cardinality == 100
  `bytes` UInt64, -- Ranging from 128 to 1152
  `duration` UInt64 -- Ranging from 100 to 400
)
ENGINE = MergeTree
PARTITION BY toDate(datetime) -- Daily partitioning
ORDER BY (user_id, device_id); -- Can only favor one column here
```

例 1: 根据给定的 *user_id* 分析当日视频流特点

```
SELECT
    toStartOfHour(datetime) AS hour,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE (toDate(hour) = today()) AND (user_id = 100)
GROUP BY hour;
```

19 rows in set. Elapsed: 0.017 sec. Processed 32.77 thousand rows,

Log: 4/210940 marks by primary key, 4 marks to read from 4 ranges

例 2: 根据给定的 *device_id* 分析当日视频流特点

```
SELECT
    toStartOfHour(datetime) AS hour,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE (toDate(hour) = today()) AND (device_id = '100')
GROUP BY hour;
```

7 rows in set. Elapsed: 8.434 sec. Processed 1.73 billion rows,

Log: 210940/210940 marks by primary key, 210940 marks to read from 4 ranges

Projection 用例

例 2: 根据给定的 *device_id* 分析当日视频流特点

添加一个 normal projection : *p_norm* 来加速 *device_id* 查询

```
ALTER TABLE video_log ADD PROJECTION p_norm
(
    SELECT
        datetime,
        device_id,
        bytes,
        duration
    ORDER BY device_id
);
```

选择所需的列并按照 *device_id* 排序

```
ALTER TABLE video_log MATERIALIZE PROJECTION p_norm;
```

例 2: 根据给定的 *device_id* 分析当日视频流特点

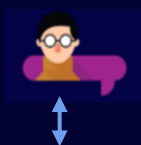
重新尝试该查询

```
SELECT
    toStartOfHour(datetime) AS hour,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE (toDate(hour) = today()) AND (device_id = '100')
GROUP BY hour;
```

7 rows in set. Elapsed: 0.055 sec. Processed 24.58 thousand rows,
153x faster!!

Log: 3/210940 marks by primary key, 3 marks to read from 3 ranges

用户行为表



timestamp	user_id	phone_num	device_id	ip_addr	...
1621409711	amos	133XXXX4567	xxxx-123	10.1.1.2	
1624098005	bird	188XXXX9985	yyyy-456	10.2.2.3	

多种数据查询口径

稀疏索引？IO 比重大

二级索引？大量随机 IO

实际应用案例

明细统计分析提速

用户行为表



timestamp	user_id	phone_num	device_id	ip_addr	...
1621409711	amos	133XXXX4567	xxxx-123	10.1.1.2	
1624098005	bird	188XXXX9985	yyyy-456	10.2.2.3	

多种数据查询口径

稀疏索引？IO 比重大

二级索引？大量随机 IO

Base

timestamp
user_id
phone_num
device_id
ip_addr
...

Projection1

user_id
timestamp

Projection2

phone_num
timestamp

Projection3

device_id
timestamp

Projection4

ip_addr
timestamp



异构多序存储

例 3: 根据域名聚合分析当日视频流特点

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

2400 rows in set. Elapsed: 11.493 sec. Processed 1.73 billion rows,

Log: 210940/210940 marks by primary key, 210940 marks to read from 4 ranges

Aggregate 1,728,000,000 rows to 2400 rows

Projection 用例

例 3: 根据域名聚合分析当日视频流特点

添加一个 aggregate projection : p_agg 来加速聚合查询

```
ALTER TABLE video_log ADD PROJECTION p_agg  
(  
    SELECT  
        toStartOfHour(datetime) AS hour,  
        domain,  
        sum(bytes),  
        avg(duration)  
    GROUP BY  
        hour,  
        domain  
);
```

聚合查询可直接定义 Projection

```
ALTER TABLE video_log MATERIALIZE PROJECTION p_agg;
```

例 3: 根据域名聚合分析当日视频流特点

重新尝试该查询

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

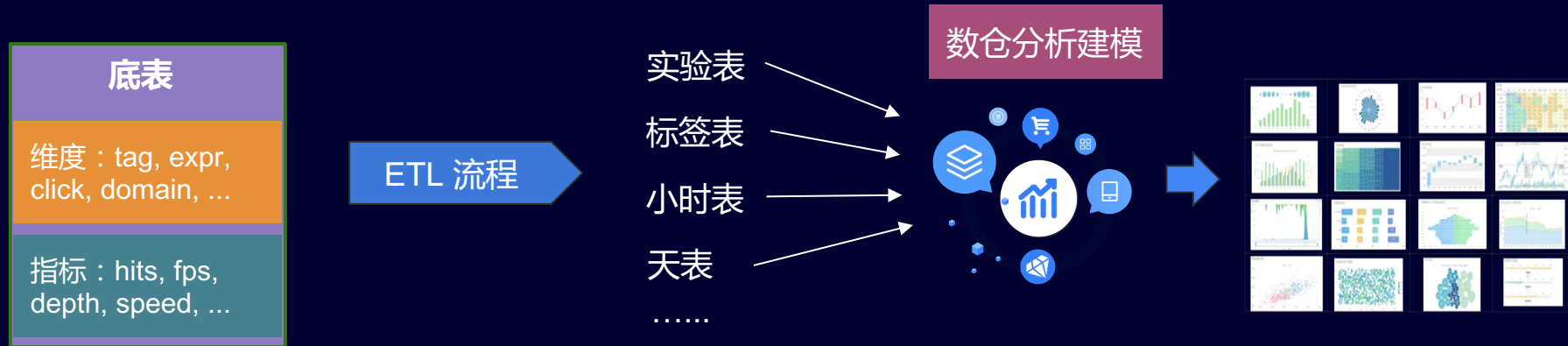
2400 rows in set. Elapsed: 0.029 sec. Processed 5.20 thousand rows,
396x faster!!

Log: 4/4 marks by primary key, 4 marks to read from 4 ranges

Aggregate 5200 rows to 2400 rows

实际应用案例

聚合统计分析提速



实际应用案例

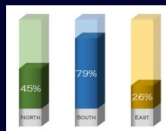
聚合统计分析提速



ETL 流程

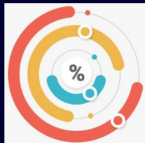
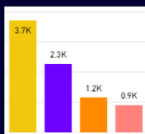
实验表
标签表
小时表
天表
.....

数仓分析建模



底表

Projection1
Projection2
Projection3
.....



~~数仓分析建模~~

~~ETL 流程~~

看板智能提速

数据一致性保障

Projection 用例

Projection 优化本质是用空间换时间，存储开销可查询 `system.projection_parts`

```
Normal projection: p_norm
SELECT
    name,
    parent_name,
    formatReadableSize(bytes_on_disk) AS bytes,
    formatReadableSize(parent_bytes_on_disk) AS parent_bytes,
    bytes_on_disk / parent_bytes_on_disk AS ratio
FROM system.projection_parts
WHERE (name = 'p_norm') AND (table = 'video_log')
```

name	parent_name	bytes	parent_bytes	ratio
p_norm	20210506_1_740_4_1651	8.77 GiB	23.94 GiB	0.36642
p_norm	20210506_741_1480_4_1651	8.81 GiB	24.01 GiB	0.36681
p_norm	20210506_1481_1647_3_1651	2.09 GiB	5.67 GiB	0.36895
p_norm	20210506_1648_1648_0_1651	14.99 MiB	38.38 MiB	0.39063

Projection 用例

Projection 优化本质是用空间换时间，存储开销可查询 `system.projection_parts`

```
SELECT                                Aggregate projection: p_agg
  name,
  formatReadableSize(bytes_on_disk) AS bytes,
  formatReadableSize(parent_bytes_on_disk) AS parent_bytes,
  rows,
  parent_rows,
  rows / parent_rows AS ratio
FROM system.projection_parts
WHERE (name = 'p_agg') AND (table = 'video_log')
```

name	bytes	parent_bytes	rows	parent_rows	ratio
p_agg	14.80 KiB	23.94 GiB	1100	775923300	0.0000014
p_agg	16.09 KiB	24.01 GiB	1200	775923300	0.0000015
p_agg	4.78 KiB	5.67 GiB	300	175107015	0.0000017
p_agg	26.80 KiB	38.38 MiB	2600	1046385	0.0024847

用例小结

- ClickHouse Projections 包含两大类：*normal* 与 *aggregate*
- 使用查询定义 Projection，新建的 Projection 仅影响后续的写入数据
- 对历史数据构建 Projection 需要进行 Materialize 操作
- 查询无需任何改动即可使用 Projection 优化
- 可对单表增加多个 Projection，查询将择优使用



- 1 ClickHouse 背景介绍
- 2 Projection 简介与用例
- 3 Projection 原理与实现
- 4 特性对比与生产应用效果

Projection 的主要构件

- Projection 定义
- Projection 存储
- Projection 查询分析 ★

Projection 的主要构件

- Projection 定义
 - a. 通过用户查询直接定义
 - b. 自动推断 Projection 类型与相关属性
- Projection 存储
- Projection 查询分析 ★

Projection 的主要构件

- Projection 定义
 - a. 通过用户查询直接定义
 - b. 自动推断 Projection 类型与相关属性
- Projection 存储
 - a. 存储在原始表的 Part 目录之下
 - b. INSERT , MERGE , MUTATION , REPLICATION 均跟随原表 Part , 逻辑上保持一致
- Projection 查询分析 ★

Projection 的主要构件

- Projection 定义
 - a. 通过用户查询直接定义
 - b. 自动推断 Projection 类型与相关属性
- Projection 存储
 - a. 存储在原始表的 Part 目录之下
 - b. INSERT , MERGE , MUTATION , REPLICATION 均跟随原表 Part , 逻辑上保持一致
- Projection 查询分析 ★
 - a. 回溯分析查询计划 , 使用标准表达式名称进行匹配验证
 - b. 提前进行索引分析 , 选择最优 Projection , 并缓存分析结果
 - c. 根据所选的 Projection 进行查询计划重建

Projection 定义

- 可看作 *CREATE TABLE AS SELECT (CTAS)* 语句
 - a. *SELECT* 中的别名与表达式转换为标准命名
 - b. *ORDER BY* 语句生成 *normal projections* , 使用 *MergeTree* 存储 , 排序键即为主键
 - c. *GROUP BY* 语句生成 *aggregate projections* , 使用 *AggregatingMergeTree* 存储 , 聚合维度即为主键

Projection 定义

- 可看作 *CREATE TABLE AS SELECT (CTAS)* 语句

- a. SELECT 中的别名与表达式转换为标准命名
- b. ORDER BY 语句生成 *normal projections* , 使用 *MergeTree* 存储 , 排序键即为主键
- c. GROUP BY 语句生成 *aggregate projections* , 使用 *AggregatingMergeTree* 存储 , 聚合维度即为主键
- d. 聚合函数将生成中间状态类型: *AggregateFunction(...)* , 如前例中的

AggregateFunction(sum, UInt64)* 与 *AggregateFunction(duration, UInt64)

```
ALTER TABLE video_log ADD PROJECTION p_agg
```

```
(
```

```
  SELECT
```

```
    toStartOfHour(datetime) AS hour,  
    domain,  
    sum(bytes),  
    avg(duration)
```

```
  GROUP BY
```

```
    hour,  
    domain
```

```
);
```

```
CREATE TABLE p_agg
```

```
ENGINE AggregatingMergeTree
```

```
ORDER BY (`toStartOfHour(datetime)`, domain)
```

```
AS
```

```
SELECT
```

```
  toStartOfHour(datetime),  
  domain,  
  sum(bytes),  
  avg(duration)
```

```
FROM
```

```
  video_log
```

```
GROUP BY
```

```
  toStartOfHour(datetime), domain
```

执行至聚合前阶段

Projection 存储

- 可看作原始表 Part 的伴生 Part
 - a. 依旧是 Part，存储模型与普通 MergeTree 几乎一致
 - b. Projection Part 复用原始 Part 的分区信息，自动适配分区级 Part 剪枝
 - c. MERGE, MUTATION 与 REPLICATION 均按照递归的方式层层处理

Projection 存储

- 可看作原始表 Part 的伴生 Part
 - a. 依旧是 Part，存储模型与普通 MergeTree 几乎一致
 - b. Projection Part 复用原始 Part 的分区信息，自动适配分区级 Part 剪枝
 - c. MERGE, MUTATION 与 REPLICATION 均按照递归的方式层层处理

data/data/default/video_log/20210506_1_740_4_1651 (parent_part)

```
|-- ...
|-- minmax_datetime.idx
|-- partition.dat
|----- p_agg.proj
|-- p_norm.proj
|   |-- bytes.bin
|   |-- bytes.mrk2
|   |-- checksums.txt
|   |-- columns.txt
|   |-- count.txt
|   |-- ...
|   |-- duration.bin
|   |-- duration.mrk2
|   |-- primary.idx
|   |-- avg%28duration%29.bin
|   |-- avg%28duration%29.mrk2
|   |-- checksums.txt
|   |-- columns.txt
|   |-- count.txt
|   |-- ...
|   |-- sum%28bytes%29.bin
|   |-- sum%28bytes%29.mrk2
|   |-- toStartOfHour%28datetime%29.bin
|   |-- toStartOfHour%28datetime%29.mrk2
```

Projection part 复用分区信息

Projection 查询分析



查询分析原理

Table *video_log*:

Columns:

datetime	DateTime,
user_id	UInt64,
device_id	String,
domain	LowCardinality(String),
bytes	UInt64,
duration	UInt64,

Projection *p_agg*:

Columns:

toStartOfHour(datetime)	DateTime,
domain	LowCardinality(String),
sum(bytes)	AggregateFunction(sum, UInt64),
avg(duration)	AggregateFunction(avg, UInt64),

Primary Keys:

toStartOfHour(datetime), domain

输入聚合查询： `SELECT`

```
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

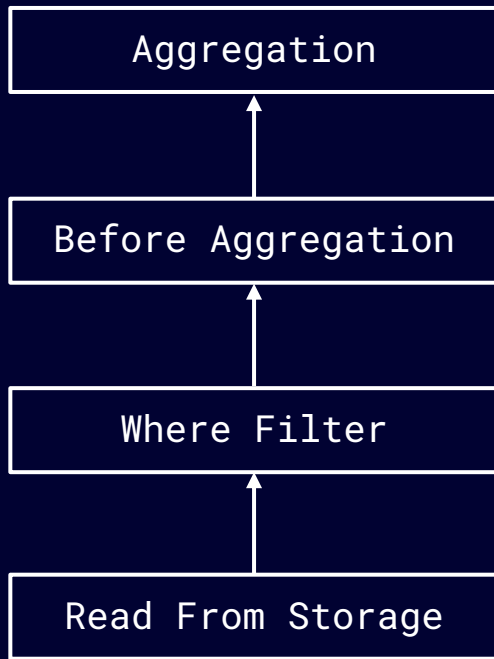
查询分析原理

输入聚合查询：

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

形成初步查询计划（聚合前）

查询计划：



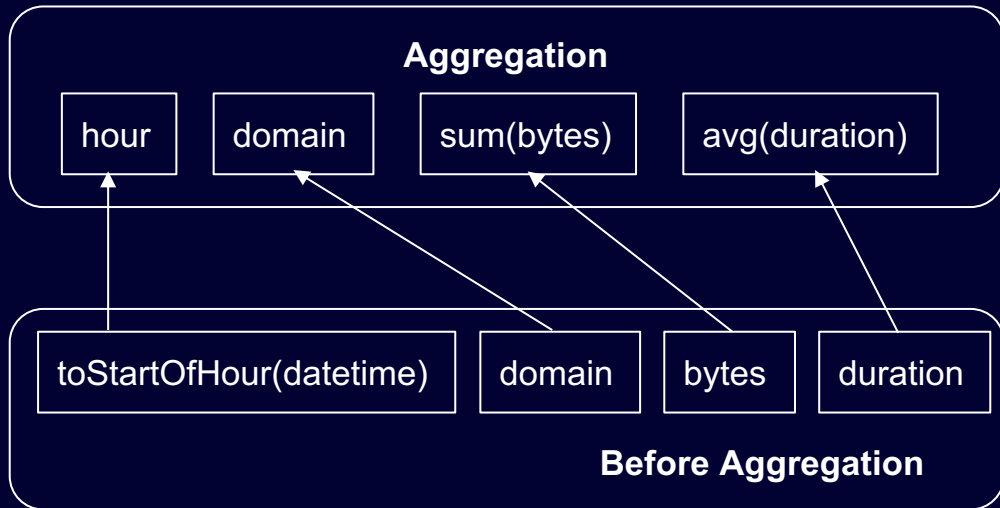
查询分析原理

输入聚合查询：

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

回溯分析各阶段算子，首先是聚合算子

聚合算子数据映射关系：



查询分析原理

Projection p_agg 元数据结构：

Columns:

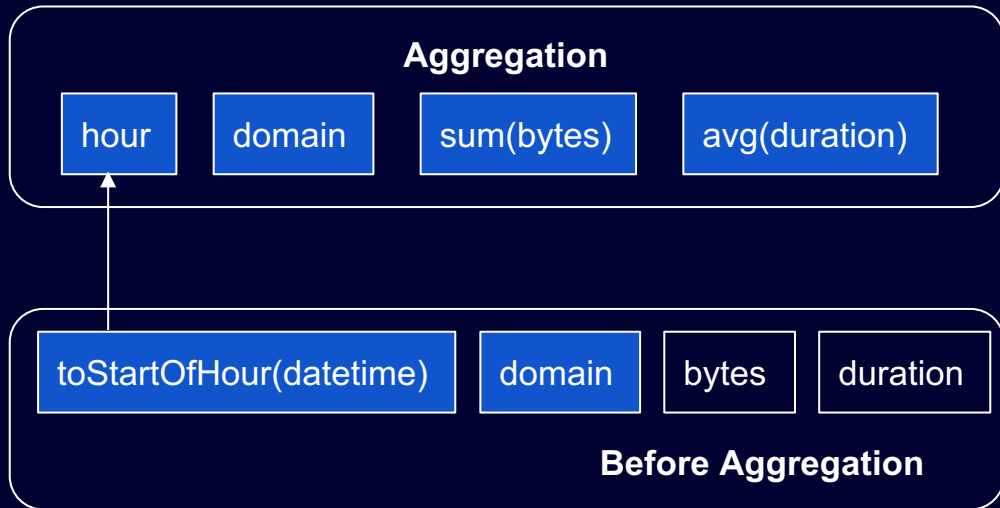
```
toStartOfHour(datetime) DateTime,  
domain LowCardinality(String),  
sum(bytes) AggregateFunction(sum, UInt64),  
avg(duration) AggregateFunction(avg, UInt64),
```

Primary Keys:

```
toStartOfHour(datetime), domain
```

回溯分析各阶段算子，首先是聚合算子

聚合算子所需的输入均可由 p_agg 提供，是潜在的匹配



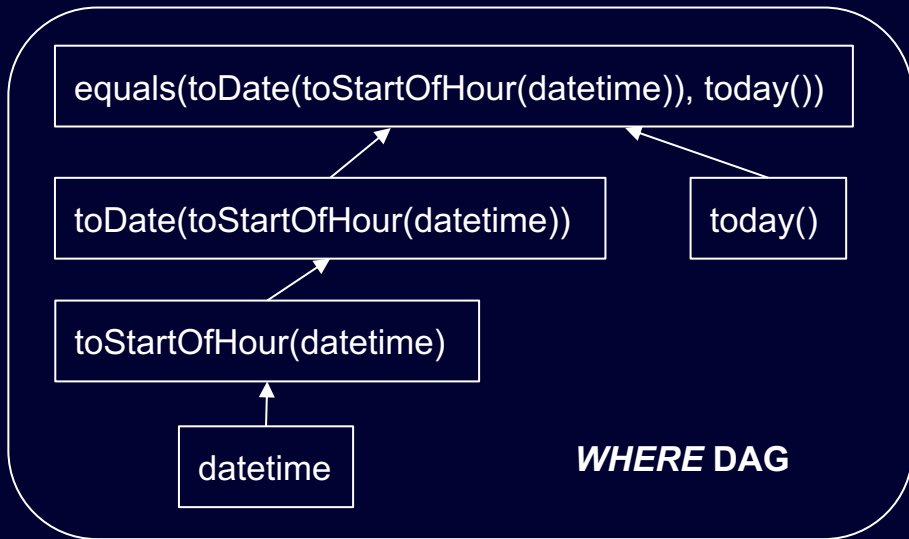
查询分析原理

输入聚合查询：

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

接着回溯 WHERE 过滤条件

WHERE 过滤算子表达式树（DAG）：



查询分析原理

Projection p_agg 元数据结构：

Columns:

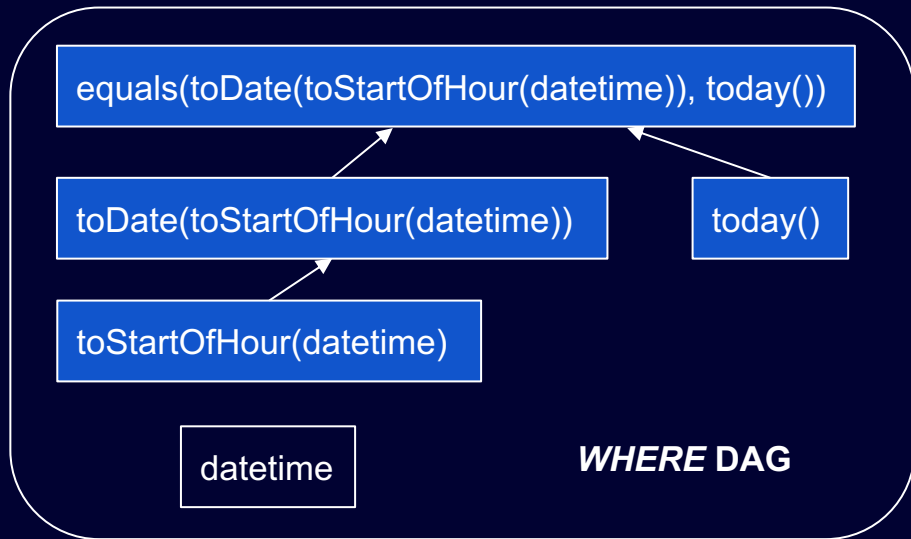
toStartOfHour(datetime) DateTime,
domain LowCardinality(String),
sum(bytes) AggregateFunction(sum, UInt64),
avg(duration) AggregateFunction(avg, UInt64),

Primary Keys:

toStartOfHour(datetime), domain

接着回溯 WHERE 过滤条件

WHERE DAG 修正后输入可由 p_agg 提供，是合法的匹配

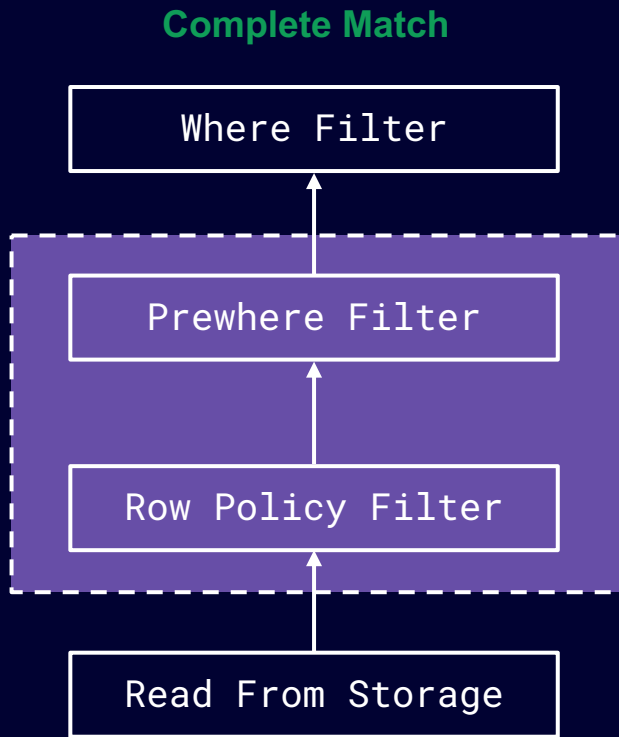


查询分析原理

输入聚合查询：

```
SELECT
    toStartOfHour(datetime) AS hour,
    domain,
    sum(bytes),
    avg(duration)
FROM video_log
WHERE toDate(hour) = today()
GROUP BY hour, domain;
```

按需回溯剩余的环节，直到起点



查询分析原理

在所有合法的 Projection 候选中，择优选择

- 对每一个候选进行索引分析，得出其预期数据扫描量，并缓存结果

在所有合法的 Projection 候选中，择优选择

- 对每一个候选进行索引分析，得出其预期数据扫描量，并缓存结果
- 选择预期扫描数据最少的候选
 - 不用区分 projection 类型是 *normal* 或 *aggregate*，数据量少则优
 - 预期扫描量同时包含了 projection 的物化程度
 - 尽可能复用缓存结果，避免重复进行索引分析

在所有合法的 Projection 候选中，择优选择

- 对每一个候选进行索引分析，得出其预期数据扫描量，并缓存结果
- 选择预期扫描数据最少的候选
 - 不用区分 projection 类型是 *normal* 或 *aggregate*，数据量少则优
 - 预期扫描量同时包含了 projection 的物化程度
 - 尽可能复用缓存结果，避免重复进行索引分析
- 当最终选择某个 projection 后，将利用前述的回溯分析过程重建查询计划，并同时满足 projection parts 和 ordinary parts 的读取与计算

一致性保障

INSERT

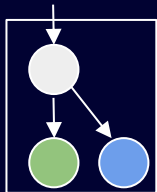
SELECT

MUTATION

一致性保障

INSERT

当数据块写入时，其作为数据源向所有定义的 Projections 提供输入，形成 Projection Parts，最终和原始数据合并构建出带有 projection 的 part 数据目录



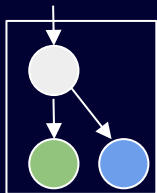
SELECT

MUTATION

一致性保障

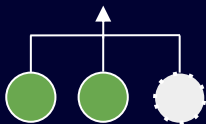
INSERT

当数据块写入时，其作为数据源向所有定义的 Projections 提供输入，形成 Projection Parts，最终和原始数据合并构建出带有 projection 的 part 数据目录



SELECT

当查询命中某一 Projection 时，形成的查询计划将确保所有数据产生符合预期的结果。针对缺失 Projection Parts 的数据，将在运行时动态构建并在不引入额外计算开销的前提下与其余数据合并

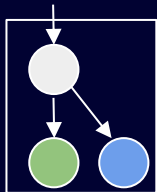


MUTATION

一致性保障

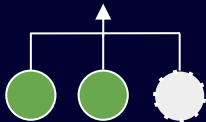
INSERT

当数据块写入时，其作为数据源向所有定义的 Projections 提供输入，形成 Projection Parts，最终和原始数据合并构建出带有 projection 的 part 数据目录



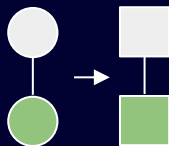
SELECT

当查询命中某一 Projection 时，形成的查询计划将确保所有数据产生符合预期的结果。针对缺失 Projection Parts 的数据，将在运行时动态构建并在不引入额外计算开销的前提下与其余数据合并



MUTATION

Projection 在定义时记录了其关联依赖的原始列信息。当对应的列发生变化时，所有相关的 Projection 将被重新物化，形成的新 Part 将包含一致的 Projection Part 进行原子提交





- 1 ClickHouse 背景介绍
- 2 Projection 简介与用例
- 3 Projection 原理与实现
- 4 特性对比与生产应用效果

Pros & Cons

Pros

1. SELECT, INSERT, UPDATE, DELETE 等操作的一致性保障
2. 查询无需任何改动，自动匹配最优 Projection 进行计算
3. 可直接通过待优化的查询进行定义，并自动泛化匹配其他查询

Pros & Cons

Pros

1. SELECT, INSERT, UPDATE, DELETE 等操作的一致性保障
2. 查询无需任何改动，自动匹配最优 Projection 进行计算
3. 可直接通过待优化的查询进行定义，并自动泛化匹配其他查询

Cons

1. 无法跨 Part 聚合
2. 无法脱离原始表存储，无法使用不同的生命周期与存储介质配置
3. 不支持 JOINS

特性对比汇总

特性	Projection	Materialized View	AggregatingMergeTree
数据一致性	Yes	No	Yes (nonintuitive)
查询分析能力	Yes	No	No
数据索引能力	Yes	No	No
明细数据存储	Yes	Yes	No
无阻塞写入	Yes*	No	No
复杂查询 (JOINS)	No	Yes	No

生产实测结果

数据集大小：每天 350 亿记录

聚合维度：group by toStartOfTenMinutes(datetime), domain

数据聚合比：0.004%



查询使用的 聚合函数	查询耗时 (1 thread)		查询耗时 (24 threads)	
	原始表	Projection GROUP BY toStartOfTenMinutes	原始表	Projection GROUP BY toStartOfTenMinutes
countIf with filter	28.75s	0.03s	1.56s	0.02s
uniqHLL12	14.18s	0.05s	1.79s	0.05s
Three simple aggregates	50.29s	0.04s	3.43s	0.02s

生产实测结果

数据集大小：每天 350 亿记录

聚合维度：group by toStartOfTenMinutes(datetime), domain

数据聚合比：0.004%

查询使用的 聚合函数	查询耗时 (1 thread)		查询耗时 (24 threads)	
	原始表	Projection GROUP BY toStartOfTenMinutes	原始表	Projection GROUP BY toStartOfTenMinutes
 countIf with filter	28.75s	0.03s ↑↑	1.56s	0.02s
 uniqHLL12	14.18s	0.05s ↑	1.79s	0.05s
Three simple aggregates	50.29s	0.04s ↑↑↑	3.43s	0.02s

生产实测结果

数据集大小：每天 350 亿记录

聚合维度：group by toStartOfTenMinutes(datetime), domain

数据聚合比：0.004%

查询使用的 聚合函数	查询耗时 (1 thread)		查询耗时 (24 threads)	
	原始表	Projection GROUP BY toStartOfTenMinutes	原始表	Projection GROUP BY toStartOfTenMinutes
countIf with filter	28.75s	0.03s	1.56s	0.02s
uniqHLL12	14.18s	0.05s	1.79s	0.05s
Three simple aggregates	50.29s	0.04s	3.43s	0.02s

生产实测结果

Parent part rows: 1118376

Projection_a rows: 9188

聚合函数	存储空间
<i>countIf(col_1 = 0)</i>	16KB
<i>count()</i>	31KB
<i>avg(col_1)</i>	51KB
<i>sum(col_1)</i>	25KB
<i>uniqHLL12(col_2)</i>	18MB
<i>uniqExact(col_2)</i>	396MB

Projection_b rows: 13314

聚合函数	存储空间
<i>max(col_3)</i>	35KB
<i>quantileTDigest(0.9)(col_4)</i>	3.9MB

生产实测结果

Parent part rows: 1118376

Projection_a rows: 9188

聚合函数	存储空间
<i>countIf(col_1 = 0)</i>	16KB
<i>count()</i>	31KB
<i>avg(col_1)</i>	51KB
<i>sum(col_1)</i>	25KB
<i>uniqHLL12(col_2)</i>	18MB
<i>uniqExact(col_2)</i>	396MB

Projection_b rows: 13314

聚合函数	存储空间
<i>max(col_3)</i>	35KB
<i>quantileTDigest(0.9)(col_4)</i>	3.9MB

生产实测结果

Parent part rows: 1118376

Projection_a rows: 9188

聚合函数	存储空间
<i>countIf(col_1 = 0)</i>	16KB
<i>count()</i>	31KB
<i>avg(col_1)</i>	51KB
<i>sum(col_1)</i>	25KB
<i>uniqHLL12(col_2)</i>	18MB
<i>uniqExact(col_2)</i>	396MB

Projection_b rows: 13314

聚合函数	存储空间
<i>max(col_3)</i>	35KB
<i>quantileTDigest(0.9)(col_4)</i>	3.9MB

生产实测结果

Parent part rows: 1118376

Projection_a rows: 9188

聚合函数	存储空间
<i>countIf(col_1 = 0)</i>	16KB
<i>count()</i>	31KB
<i>avg(col_1)</i>	51KB
<i>sum(col_1)</i>	25KB
<i>uniqHLL12(col_2)</i>	18MB
<i>uniqExact(col_2)</i>	396MB

Projection_b rows: 13314

聚合函数	存储空间
<i>max(col_3)</i>	35KB
<i>quantileTDigest(0.9)(col_4)</i>	3.9MB

生产实测结果

使用**规范化查询分析**将近期频繁的聚合查询进行抽取整合，选择合适的用于构建 Projection

1. 看板（包含 12 张图表）的渲染时间从 30 秒降至 **1 秒**。无 Projection 看板仅能成功渲染 4 张图表
2. 平均额外的存储开销小于 **20%**
3. INSERT/MERGE 未受影响，依然能保持**百万级别的每秒写入数量**



Projection 一言总结

Projection 是 ClickHouse 中更好的物化视图

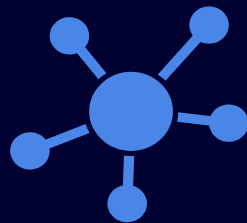
开源竞品对比

PROJECTIONS

特点	ClickHouse	Druid	Kylin	Doris	Greenplum
明细查询	强	弱	弱	中	中
实时写入	强	中	弱	弱	中
分析功能	强	弱	弱	中	强
物化存储	强*	强	强	中	中
事务特性	弱	弱	中	中	强
管控能力	强*	强	强	中	中

Future Works

Projection 设计符合 ClickHouse 计算 SQL 化，存储 Block 化的设计理念，有很大的想象空间：



- 设计实现更多类型的 Projection
 - a. 二级索引：Projection 直接存储数据指针
 - b. 位图索引：预计算过滤谓词 + 多级 PREWHERE
- 增加 Projection 的表级接口
 - a. 支持配置列编码，压缩类型，Part 格式等
 - b. 支持在 Projection 之上构建 Skip Indices 与 Projection
- 提供 Projection 与原始表分离存储的能力 ★
 - a. 支持独立的 Merge 与 TTL 策略
 - b. 支持使用不同的存储介质



THANKS