PostgreSQL

第12届PostgreSQL中国技术大会

安全可靠 × 突破 × 进化

云原生数据库PieCloudDB 性能优化之路

郭峰 拓数派

OpenPie 拓数派

打造立足于国内 基础数据计算领域的世界级高科技创新驱动机构

杭州拓数派科技发展有限公司（又称“OpenPie”），以“Data Computing for New Discoveries”「数据计算，只为新发现」为使命，成立后的短短10个月时间内，完成了包括头部产业基金、东吴证券、元禾重元和政府科创平台在内的**连续三轮战略融资**。

旗下云原生分析型数据库 PieCloudDB，以云计算架构为设计基础，**首创全新 eMPP 分布式技术**，帮助企业建立竞争壁垒的同时，实现数据价值最大化，并在新基建中承担可靠和可控的世界级云数据库底座。

OpenPie

2022 PostgreSQLChina Conference
第12届PostgreSQL中国技术大会

目录 Contents

- PostgreSQL优化器简介
- PieCloudDB优化器之分布式特性简介
- PieCloudDB优化器之云原生特性简介
- Q/A

01 PostgreSQL优化器简介

查询优化的几个阶段

- 预处理阶段
  - 通过逻辑上的等价变换，把查询树转换为更加简单高效的等式
  - 分发约束条件，收集外连接信息等
- 扫描/连接优化阶段
  - 主要处理扫描和连接操作
- 扫描/连接之外的优化阶段
  - 主要处理除扫描和连接之外的其他操作，例如聚集、排序等
- 后处理阶段
  - 主要把前面的处理结果转换成执行器期望的形式

预处理阶段

- 简化表达式
- 简化连接树
  - 把 IN,EXISTS 等类型的子查询转换为半连接
  - 提升子查询
  - 把外连接转换为内连接
  - 把外连接转换为反连接
- 分发约束条件
- 构建等价类
- 收集外连接信息
- 消除无用连接
- ...

把 IN,EXISTS 等类型的子查询转换为半连接

SELECT … FROM foo WHERE EXISTS (SELECT 1 FROM bar WHERE foo.a = bar.c);

= >

SELECT … FROM foo *SEMI JOIN* bar ON foo.a = bar.c;

提升子查询


```
SELECT * FROM foo JOIN (SELECT bar.c FROM bar JOIN baz ON TRUE) AS sub ON foo.a
= sub.c;

=>

SELECT * FROM foo JOIN (bar JOIN baz ON TRUE) ON foo.a = bar.c;
```

把外连接转换为内连接

SELECT ... FROM foo LEFT JOIN bar ON (...) WHERE bar.d = 42;

=>

SELECT ... FROM foo INNER JOIN bar ON (...) WHERE bar.d = 42;

外连接的上层有严格的约束条件，且该约束条件限定了来自 nullable side 的某一变量为非 NULL 值

把外连接转换为反连接

SELECT * FROM foo LEFT JOIN bar ON foo.a = bar.c WHERE bar.c IS NULL;

= >

SELECT * FROM foo *ANTI JOIN* bar on foo.a = bar.c;

外连接本身有严格的连接条件，且该连接条件引用了来自 nullable side 的某一变量，且该变量被上层的约束条件限定为 NULL 值

分发约束条件

- 在只有内连接的情况下，约束条件可以直接下推到它的自然语义位置
- 如果有外连接存在，那么约束条件的下推可能会受到限制

分发约束条件

- 对于外连接自己的连接条件，如果它引用了nonnullable side的表，那么它就无法被下推到该外连接之下

```
# EXPLAIN (COSTS OFF) SELECT * FROM foo LEFT JOIN bar ON foo.a = 42;
        QUERY PLAN
----------------------------
 Nested Loop Left Join
   Join Filter: (foo.a = 42)
   ->  Seq Scan on foo
   ->  Materialize
         ->  Seq Scan on bar
(5 rows)
```

分发约束条件

- 对于外连接上层的约束条件，如果它引用了nullable side的表，那么它就无法被下推到该外连接之下

```
# EXPLAIN (COSTS OFF) SELECT * FROM foo LEFT JOIN bar ON TRUE WHERE COALESCE(bar.c, 1) = 42;
           QUERY PLAN
-----------------------------------
 Nested Loop Left Join
   Filter: (COALESCE(bar.c, 1) = 42)
   ->  Seq Scan on foo
   ->  Materialize
         ->  Seq Scan on bar
(5 rows)
```

扫描/连接优化阶段

- 主要处理查询语句中FROM和WHERE部分
- 同时也会考虑到ORDER BY的信息
- 代价驱动

扫描/连接优化阶段

- 为基表生成扫描路径，并计算扫描路径的代价和结果集大小
- 搜索整个连接顺序空间，为连接操作生成连接路径
  - O(n!)
  - 动态规划
  - 遗传算法
- 考虑外连接对连接顺序的限制

```
(A leftjoin B on (Pab)) innerjoin C on (Pbc)
!= A leftjoin (B innerjoin C on (Pbc)) on (Pab)
```

扫描/连接之外的优化阶段

- 处理GROUP BY、 聚集、窗口函数、DISTINCT
- 处理集合操作
- 处理ORDER BY
- 以上每一步操作都会产生一个或多个路径
- 为每个路径添加LockRows, Limit, ModifyTable

后处理阶段

- 把最优路径转换为查询计划
- 对最优计划进行一些调整

02

PieCloudDB优化器之分布式特性简介

分布式优化器

- PieCloudDB优化器拓展了PostgreSQL优化器，使其适用于分布式架构
  - 引入了Motion的概念，使得数据可以在不同的工作节点之间移动
  - 利用Motion产生分布式的查询计划
  - 这些分布式的查询计划会被分为更小的单元，并被分发到不同的工作节点中并行执行
  - 对于聚集操作，利用分布式的优势，在工作节点之间通过多阶段聚集来提升性能

多阶段聚集


```
# explain (costs off) select sum(b) from t group by a;
                    QUERY PLAN
-----------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)
   ->  Finalize HashAggregate
         Group Key: a
         ->  Redistribute Motion 3:3  (slice2; segments: 3)
               Hash Key: a
               ->  Partial HashAggregate
                     Group Key: a
                     ->  Seq Scan on t
```

```
# explain (costs off) select sum(distinct b) from t group by a;
                    QUERY PLAN
-----------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)
   ->  HashAggregate
         Group Key: a
         ->  HashAggregate
               Group Key: a, b
               ->  Redistribute Motion 3:3  (slice2; segments: 3)
                     Hash Key: a
                     ->  Streaming HashAggregate
                           Group Key: a, b
                           ->  Seq Scan on t
```

03

PieCloudDB优化器之云原生特性简介

云原生优化器

- PieCloudDB优化器针对云原生的特性，结合对象存储的设计，实现了更多高阶的优化
  - 聚集下推
  - Block skipping
  - 预计算
  - ...

聚集下推

- 通过把聚集操作下推到连接操作之前去执行，在有些情况下可以极大的减少连接操作需要处理的数据量

```
# EXPLAIN (COSTS OFF) SELECT t1.a, avg(t2.c) FROM t1 JOIN t2 ON t1.b = t2.b GROUP BY t1.a;
                          QUERY PLAN
------------------------------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)
   ->  Finalize GroupAggregate
       Group Key: t1.a
       ->  Sort
           Sort Key: t1.a
           ->  Redistribute Motion 3:3  (slice2; segments: 3)
               Hash Key: t1.a
               ->  Hash Join
                   Hash Cond: (t1.b = t2.b)
                   ->  Seq Scan on t1
                   ->  Hash
                       ->  Broadcast Motion 3:3  (slice3; segments: 3)
                           ->  Partial HashAggregate
                               Group Key: t2.b
                               ->  Seq Scan on t2
```

关注OpenPie公众号

获得更多资讯

加入PieCloudDB技术群

了解更多干货

感谢聆听！