KunlunBase

HTAP 场景下的内核建设实践

吴夏 summerxwu

泽拓科技（深圳）有限责任公司

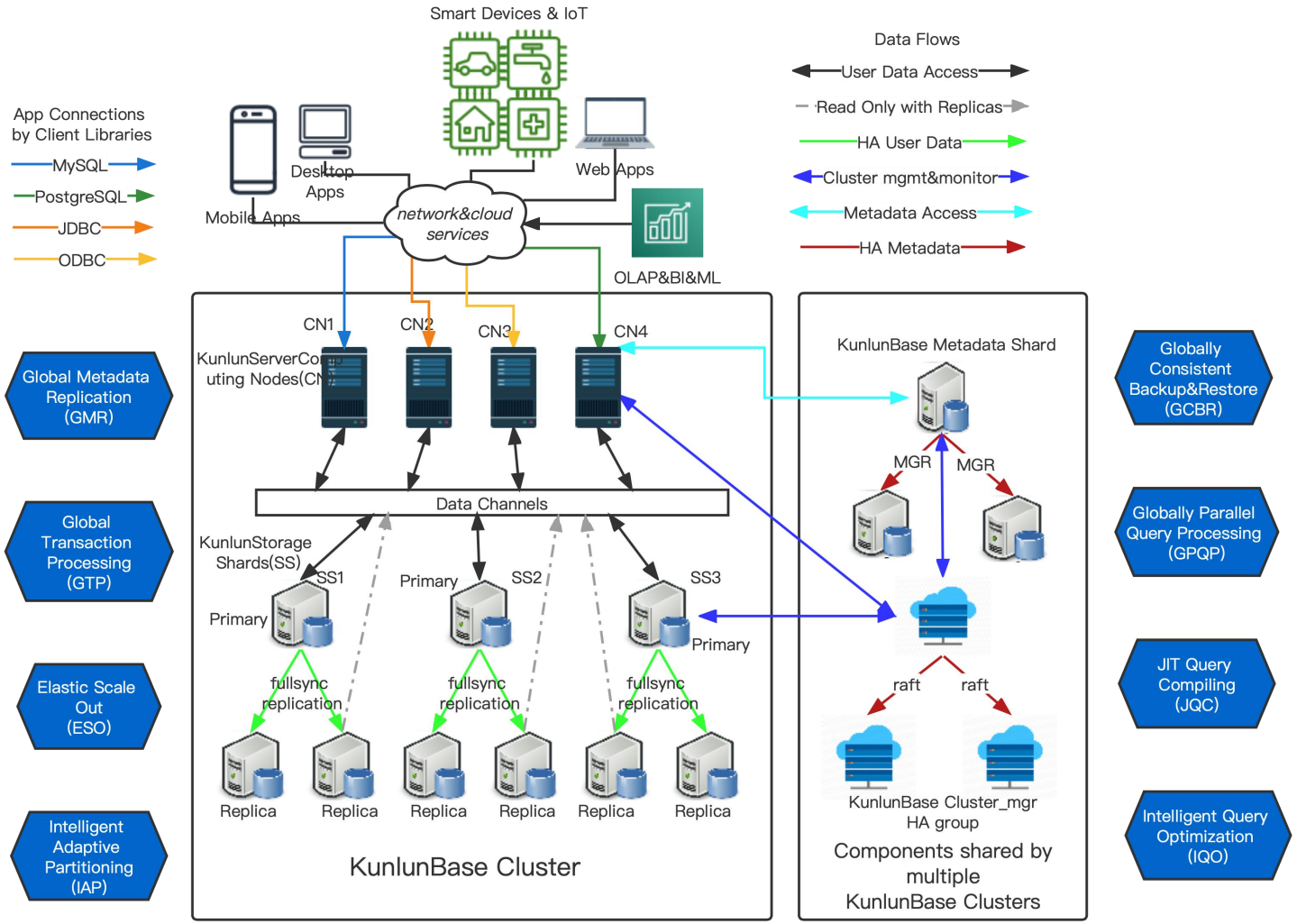
Klustron 架构介绍

KunlunBase

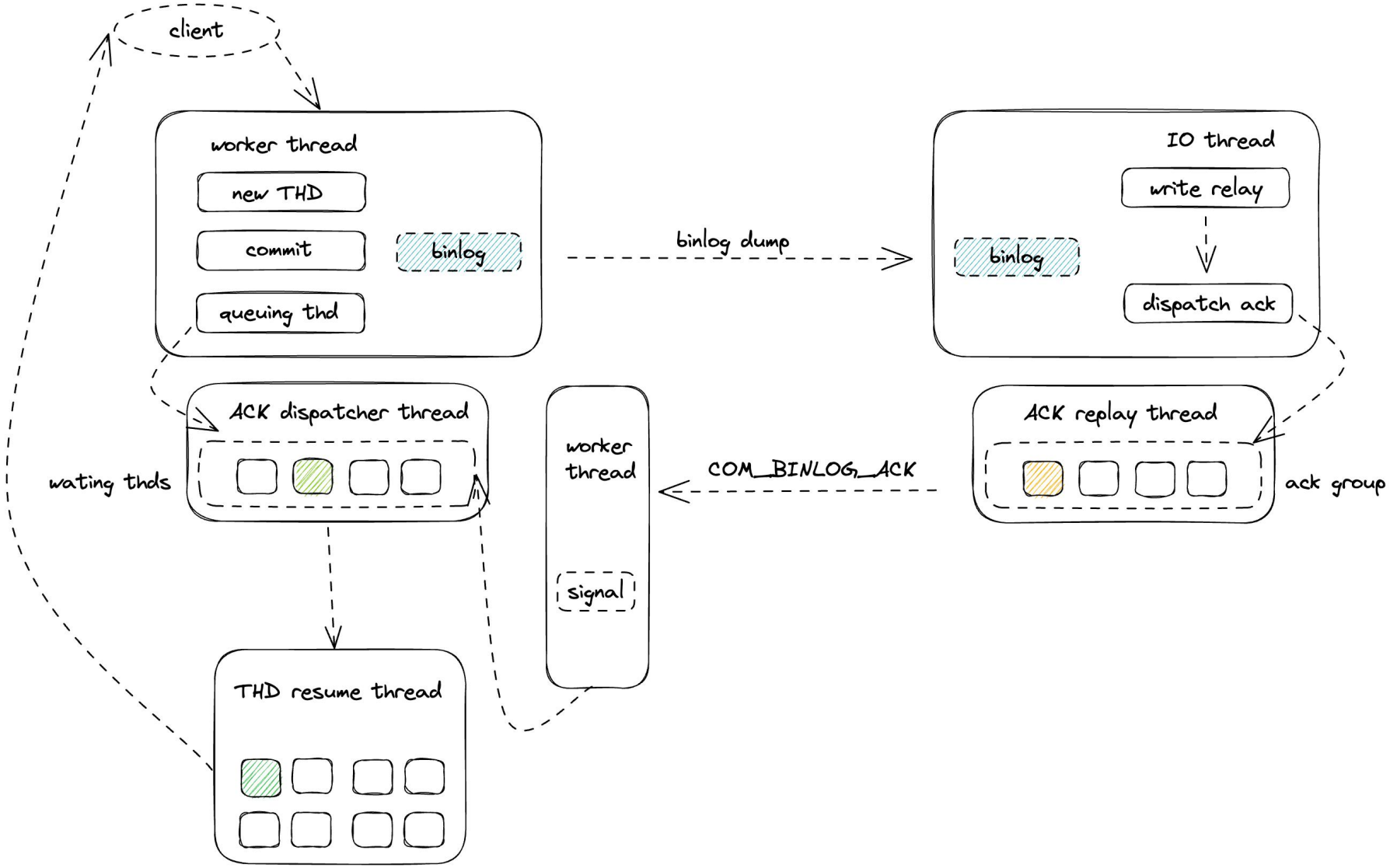
弹性伸缩&高可用

OLTP&OLAP并行查询

MySQL&PostgreSQL 双协议兼容



存储层高可用建设--fullsync

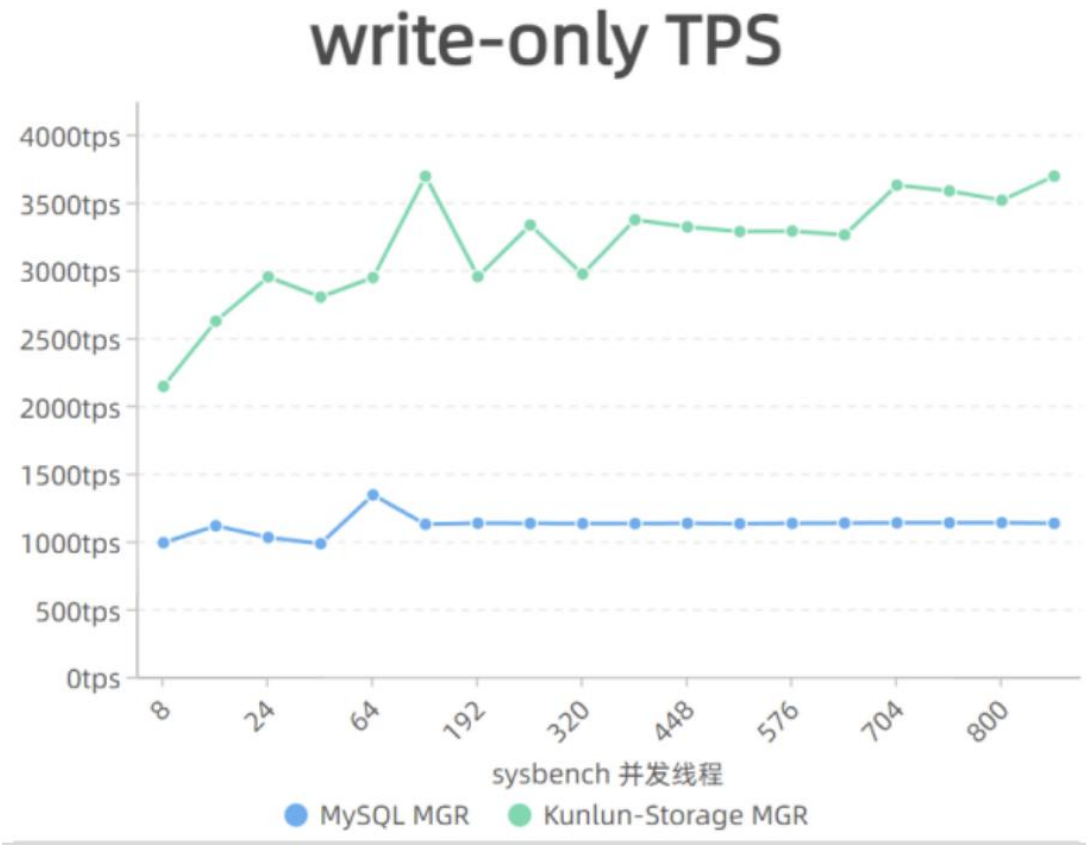


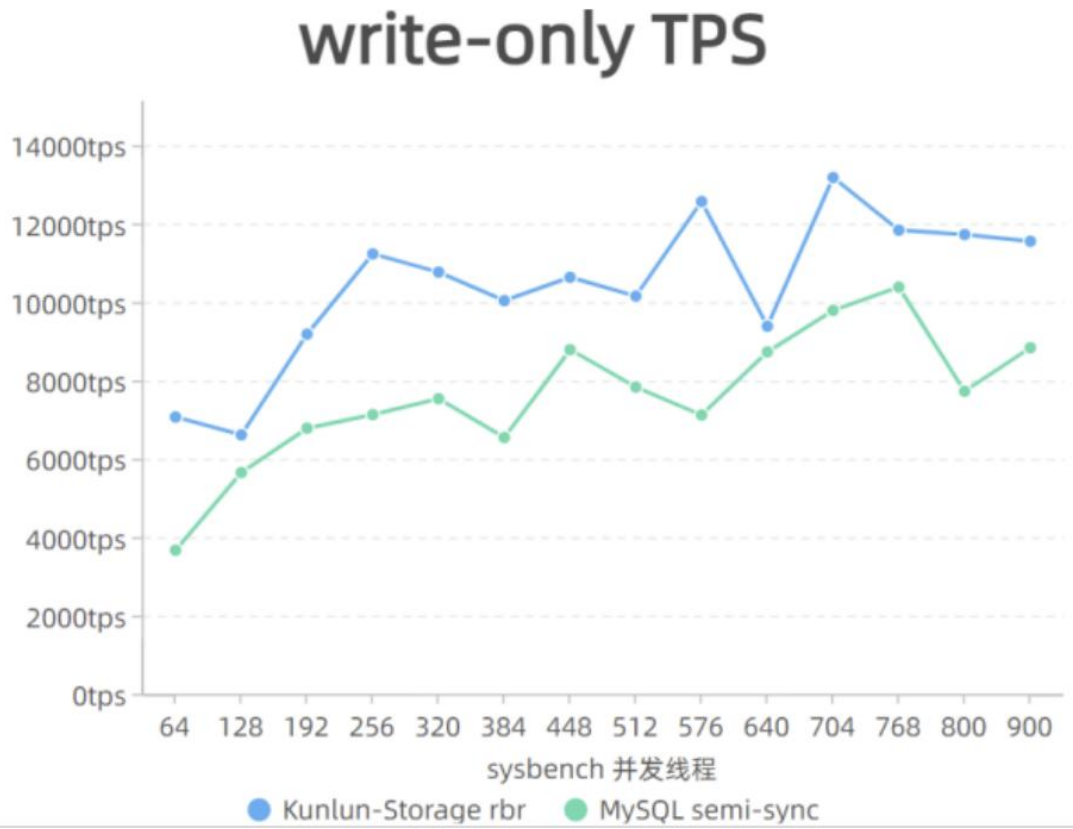
存储层高可用建设--相关参数

KunlunBase

- fullsync_consistency_level
- disable_fullsync_on_slave_ack_timeout
- fullsync_replica_ack_upto_file & fullsync_replica_ack_upto_offset

存储层高可用建设--fullsync





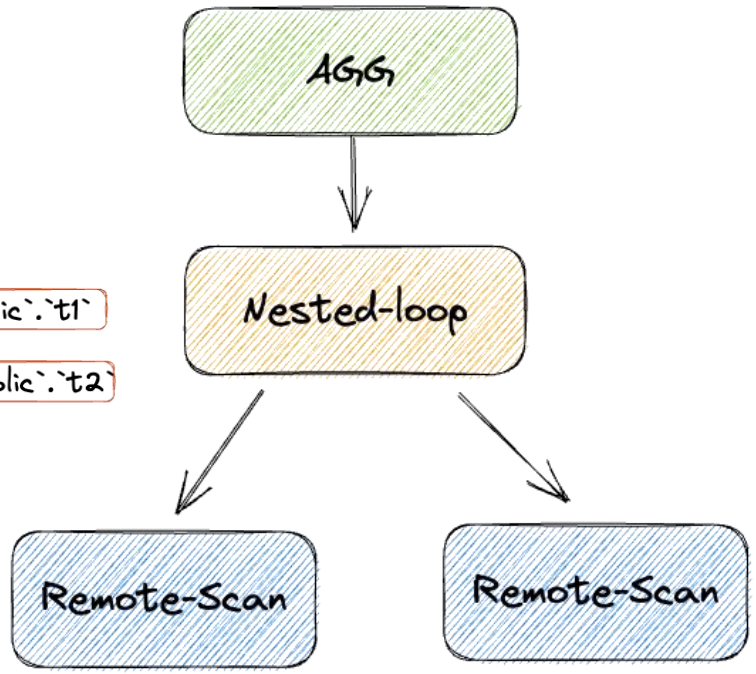
并行查询 -- remote scan

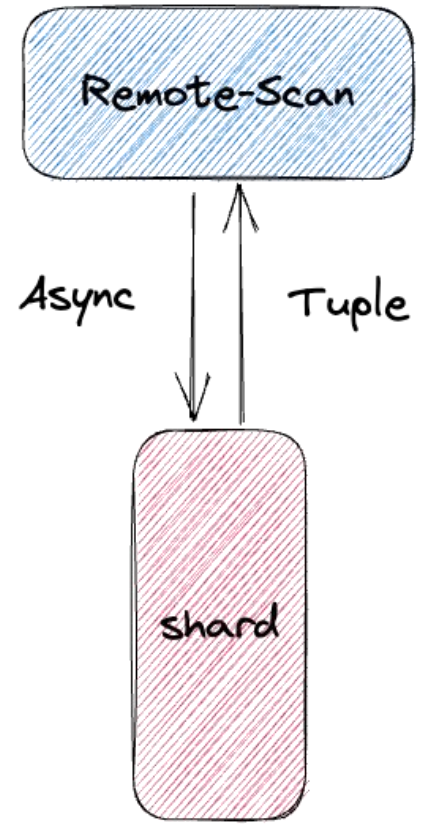
KunlunBase

```
postgres=# select count(*) from t1 inner join t2 on t1.id = t2.id ;
```

```
postgres=# explain select count(*) from t1 inner join t2 on t1.id = t2.id ;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=44.05..44.06 rows=1 width=8)
   ->  Nested Loop  (cost=44.04..44.05 rows=1 width=0)
         Join Filter: (t1.id = t2.id)
         ->  RemotePlan  (cost=22.02..22.02 rows=1 width=4)
               Shard: 6  Remote SQL: SELECT t1.id FROM  `postgres_$$_public`.`t1`
         ->  RemotePlan  (cost=22.02..22.02 rows=1 width=4)
               Shard: 5  Remote SQL: SELECT t2.id FROM  `postgres_$$_public`.`t2`
(7 rows)
```

OPEN() => NEXT() => CLOSE()

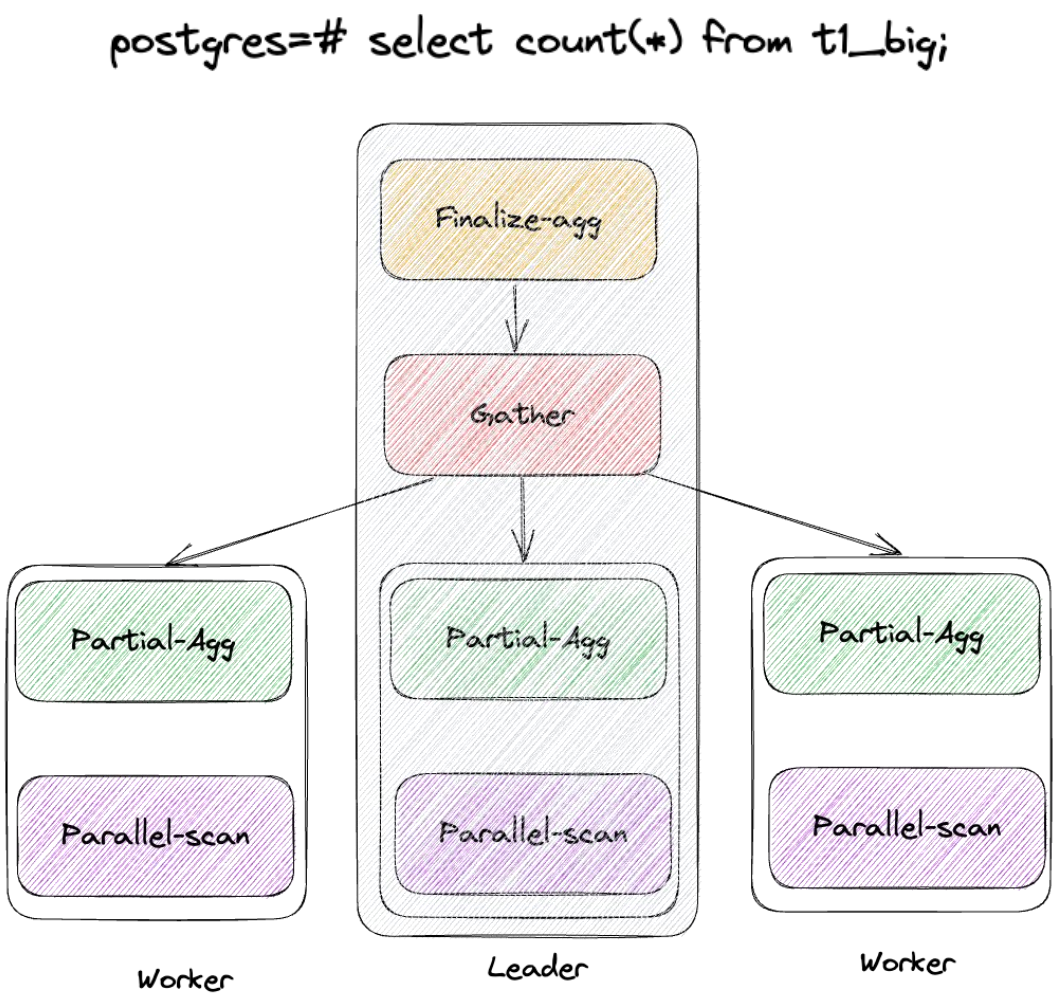




并行查询

KunlunBase

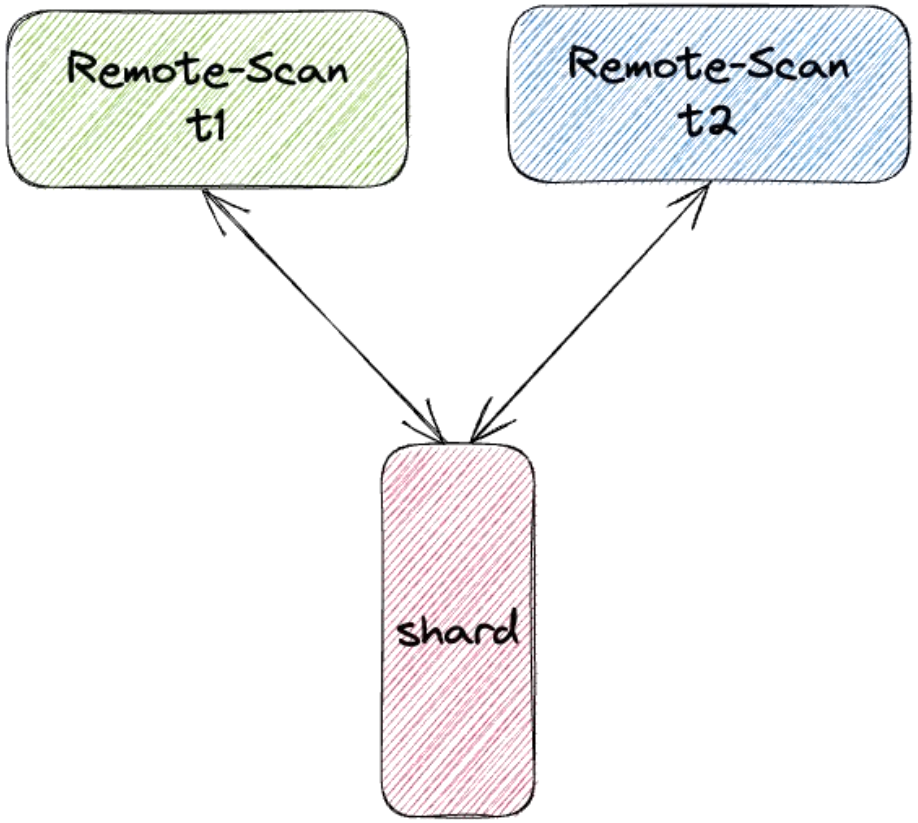
Leader-Worker & Gather



并行查询优化--ReadView 共享

KunlunBase

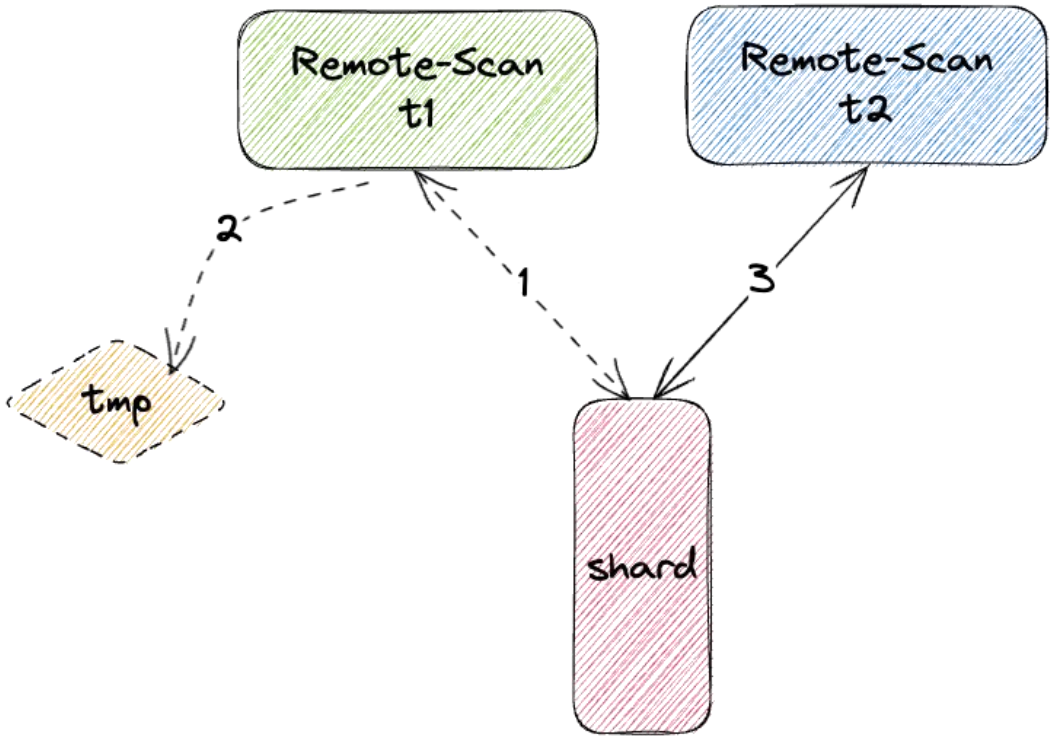
一个连接 VS 多个连接



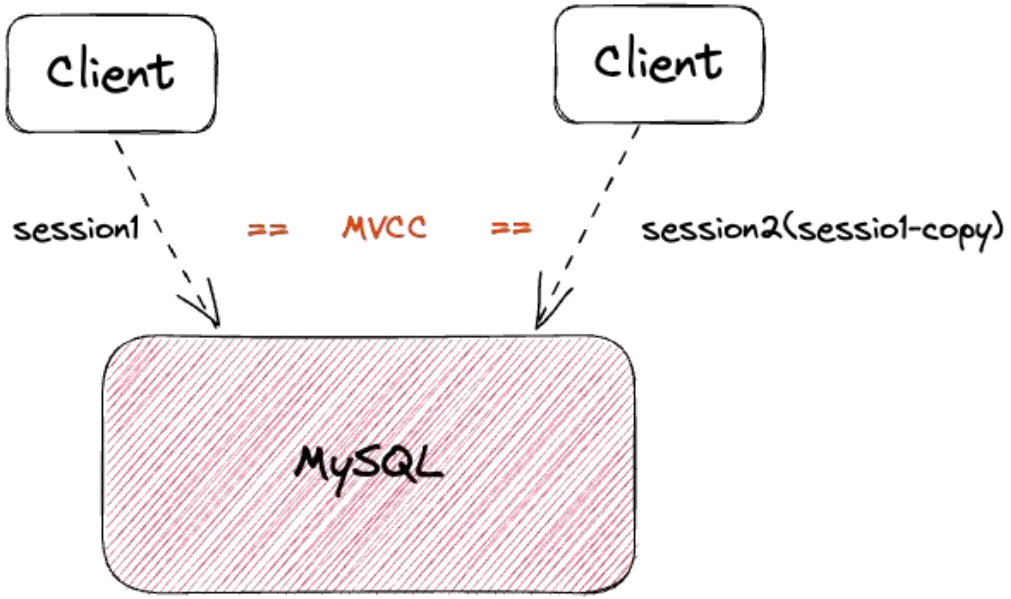
并行查询优化--ReadView 共享

物化结果集

让出连接供其他算子

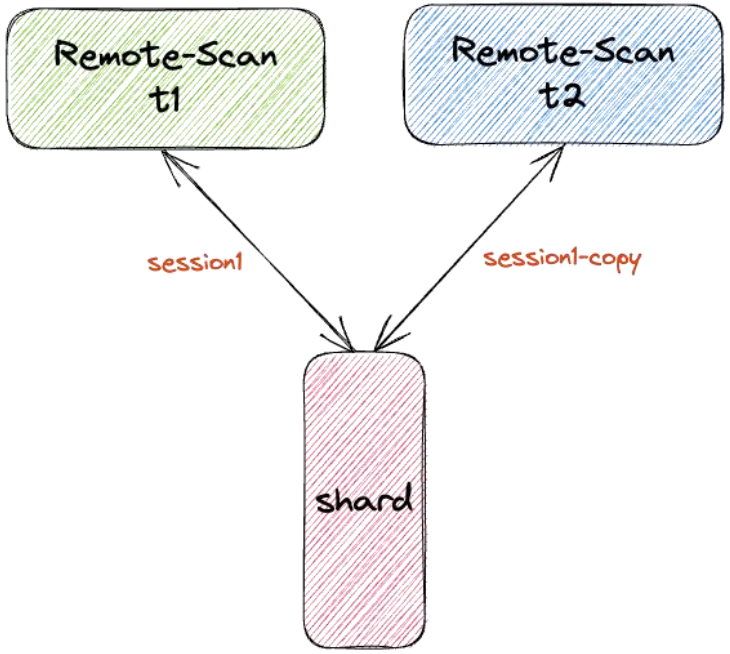


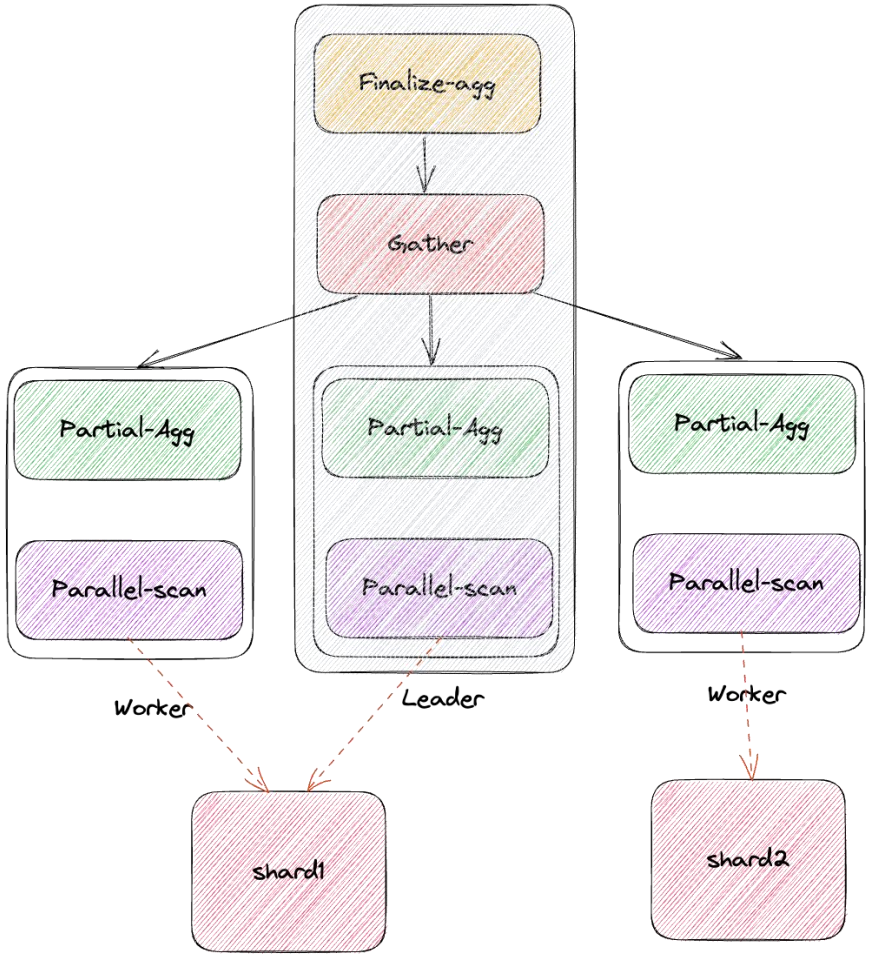
并行查询优化--ReadView 共享



start transaction read only from session 1

并行查询优化--ReadView 共享



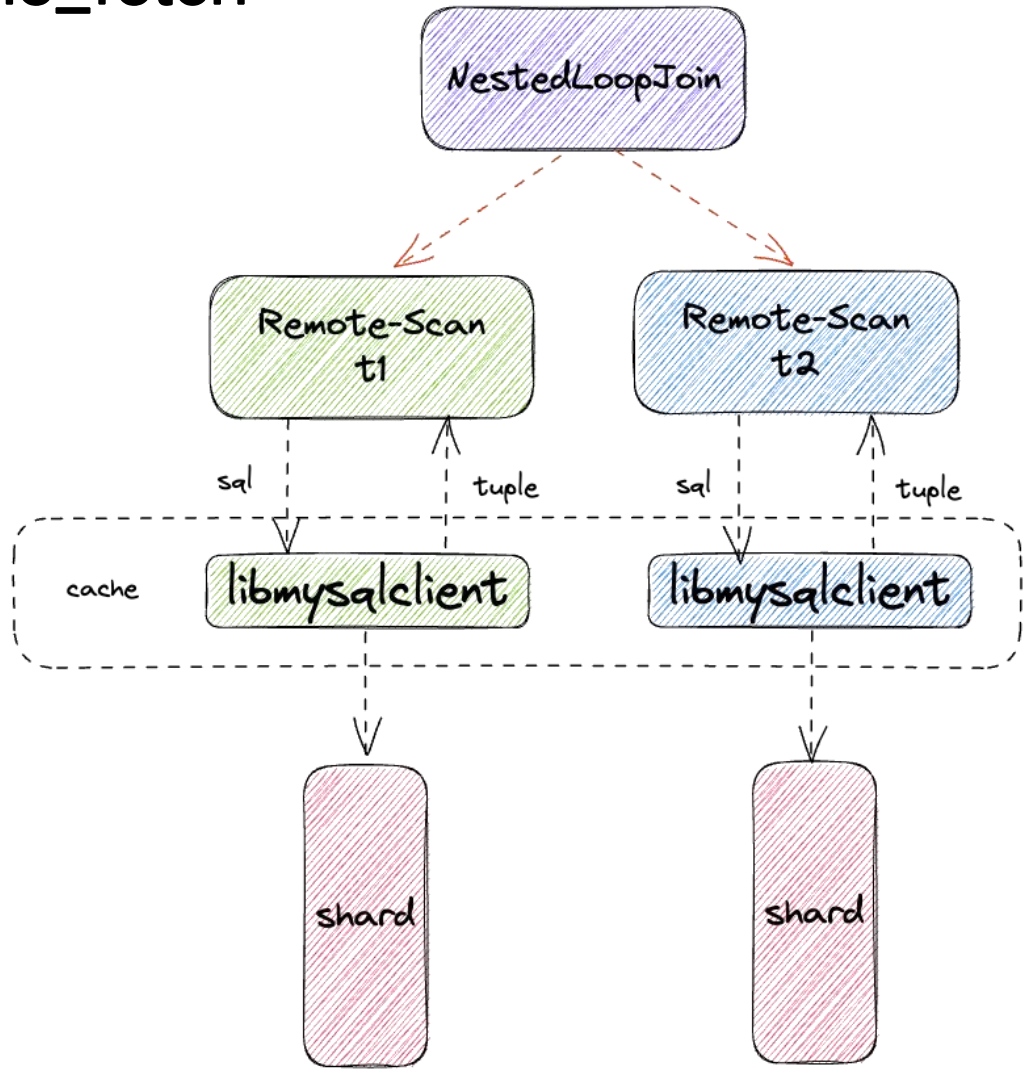


并行查询优化--tuple_fetch

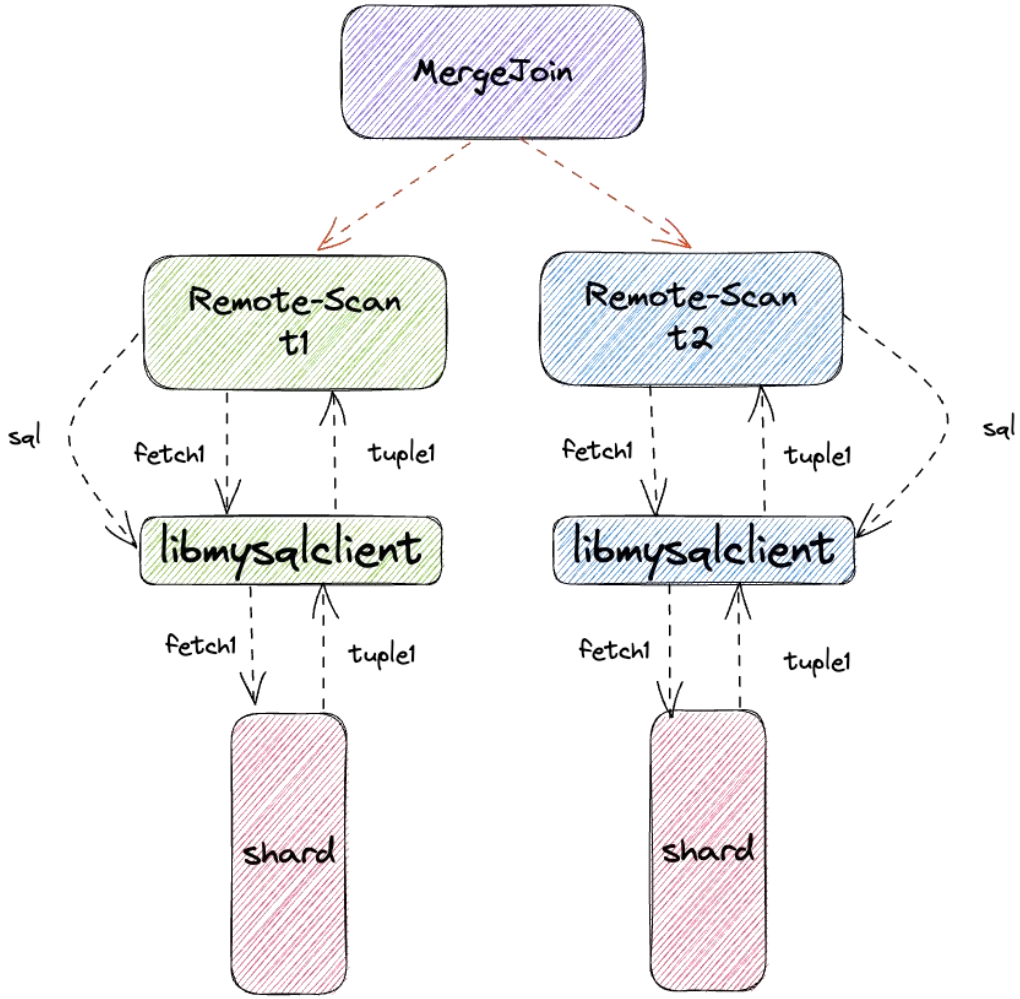
KunlunBase

```
postgres=# explain select * from t1 ,t3 limit 1;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Limit  (cost=44.04..44.05 rows=1 width=8)
   ->  Nested Loop  (cost=44.04..44.05 rows=1 width=8)
         ->  RemotePlan  (cost=22.02..22.02 rows=1 width=4)
               Shard: 4  Remote SQL: SELECT t1.id FROM  `postgres_$$_public`.`t1`
         ->  RemotePlan  (cost=22.02..22.02 rows=1 width=4)
               Shard: 5  Remote SQL: SELECT t3.id FROM  `postgres_$$_public`.`t3`
```

并行查询优化--tuple_fetch

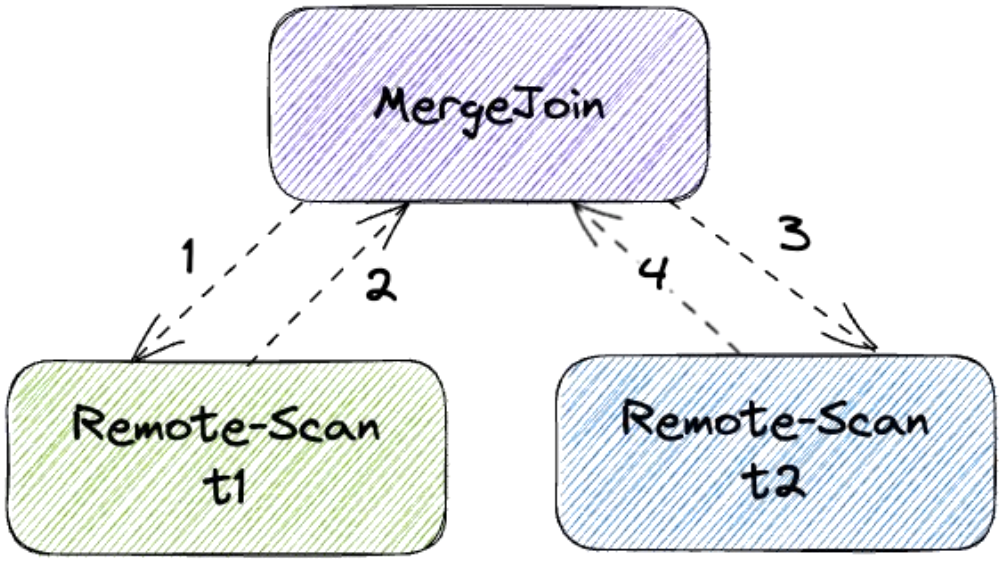


并行查询优化--tuple_fetch



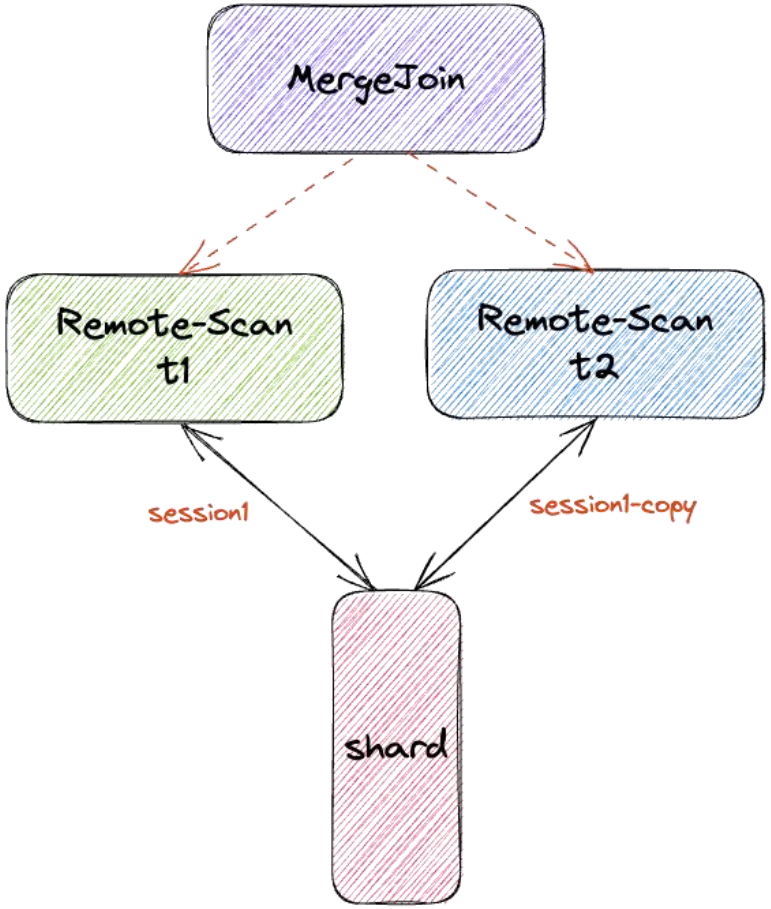
并行查询优化-- 异步查询实践

KunlunBase




```
next-mergeJoin(){
    tuple1 = next-t1();
    tuple2 = next-t2();
    result = tuple1 join tuple2;
    return result;
}
```

并行查询优化-- 异步查询实践

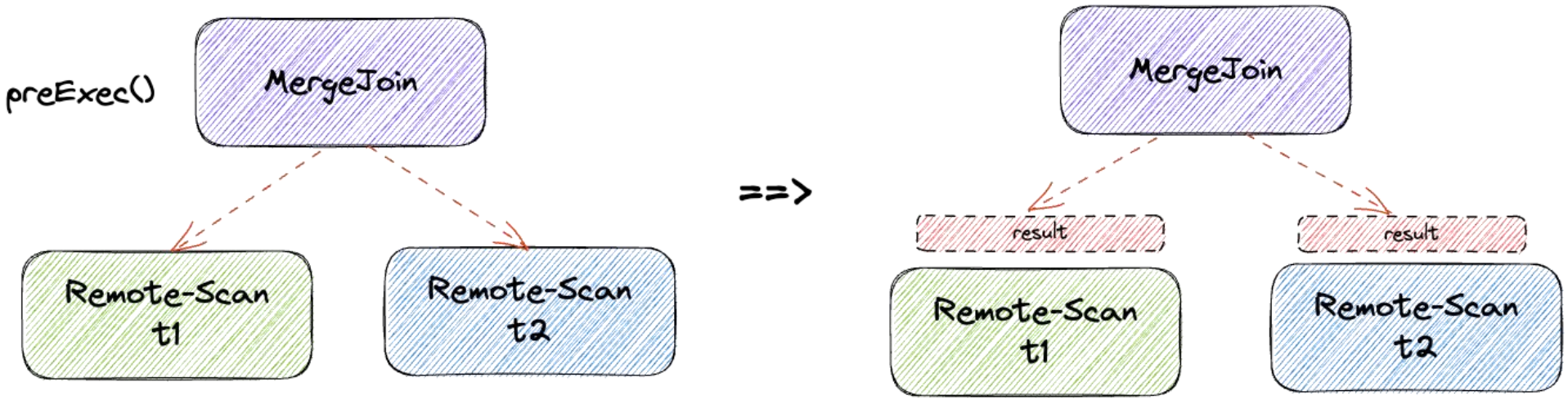



```
Open()
{
    preExecute(remoteNodeList);
}
```

```
PreExecute(remoteNode)
{
    send_sql_async(SQL);
}
```

并行查询优化-- 异步查询实践

KunlunBase



并行查询优化-- 效果对比

KunlunBase

```
postgres=#  set enable_parallel_append=off; set enable_parallel_hash=off; set  enable_parallel_remotescan=off;
SET
SET
SET
postgres=#  explain analyze verbose select * from t1 join t2 on t1.a=t2.a and t1.b=t2.b;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=21306.10..93275.70 rows=4915 width=170) (actual time=312.557..312.557 rows=0 loops=1)
   Output: t1.a, t1.b, t200.a, t200.b, t200.c, t200.d
   Inner Unique: true
   Hash Cond: ((t200.a = t1.a) AND (t200.b = t1.b))
   ->  Append  (cost=21034.10..87842.56 rows=983073 width=158) (actual time=0.508..217.279 rows=1000000 loops=1)
         ->  RemotePlan  (cost=21034.10..21034.10 rows=249470 width=158) (actual time=0.507..41.891 rows=249588 loops=1)
               Output: t200.a, t200.b, t200.c, t200.d
               Shard: 3  Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t200`  as t2
         ->  RemotePlan  (cost=20366.49..20366.49 rows=240683 width=158) (actual time=0.288..41.590 rows=250376 loops=1)
               Output: t201.a, t201.b, t201.c, t201.d
               Shard: 3  Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t201`  as t2
         ->  RemotePlan  (cost=20455.76..20455.76 rows=240092 width=158) (actual time=0.478..42.943 rows=249786 loops=1)
               Output: t202.a, t202.b, t202.c, t202.d
               Shard: 4  Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t202`  as t2
         ->  RemotePlan  (cost=21070.84..21070.84 rows=252828 width=158) (actual time=0.293..41.515 rows=250250 loops=1)
               Output: t203.a, t203.b, t203.c, t203.d
               Shard: 3  Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t203`  as t2
   ->  Hash  (cost=197.00..197.00 rows=5000 width=12) (actual time=2.318..2.318 rows=5000 loops=1)
         Output: t1.a, t1.b
         Buckets: 8192  Batches: 1  Memory Usage: 279kB
         ->  RemotePlan  (cost=197.00..197.00 rows=5000 width=12) (actual time=0.502..1.491 rows=5000 loops=1)
               Output: t1.a, t1.b
               Shard: 4  Remote SQL: SELECT t1.a,t1.b FROM  `postgres_$$_public`.`t1`
 Planning Time: 0.383 ms
 Execution Time: 312.614 ms
(25 rows)
```

并行查询优化-- 效果对比

```
postgres=#  explain analyze verbose select * from t1 join t2 on t1.a=t2.a and t1.b=t2.b;
                                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=20737.76..44846.23 rows=4915 width=170) (actual time=111.897..112.626 rows=0 loops=1)
   Output: t1.a, t1.b, t203.a, t203.b, t203.c, t203.d
   Workers Planned: 3
   Workers Launched: 3
   ->  Hash Join  (cost=20727.76..44344.73 rows=1585 width=170) (actual time=90.545..90.545 rows=0 loops=4)
         Output: t1.a, t1.b, t203.a, t203.b, t203.c, t203.d
         Inner Unique: true
         Hash Cond: ((t203.a = t1.a) AND (t203.b = t1.b))
         Worker 0: actual time=82.139..82.140 rows=0 loops=1
         Worker 1: actual time=107.644..107.644 rows=0 loops=1
         Worker 2: actual time=82.483..82.484 rows=0 loops=1
         ->  Parallel Append  (cost=20455.76..42407.85 rows=317120 width=158) (actual time=0.460..62.105 rows=250000 loops=4)
               Worker 0: actual time=0.443..55.703 rows=250250 loops=1
               Worker 1: actual time=0.412..73.952 rows=249786 loops=1
               Worker 2: actual time=0.552..55.593 rows=249588 loops=1
               ->  RemotePlan  (cost=21070.84..21070.84 rows=252828 width=158) (actual time=0.442..42.813 rows=250250 loops=1)
                     Output: t203.a, t203.b, t203.c, t203.d
                     Shard: 3     Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t203`  as t2
                     Worker 0: actual time=0.442..42.813 rows=250250 loops=1
               ->  RemotePlan  (cost=21034.10..21034.10 rows=249470 width=158) (actual time=0.552..42.976 rows=249588 loops=1)
                     Output: t200.a, t200.b, t200.c, t200.d
                     Shard: 3     Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t200`  as t2
                     Worker 2: actual time=0.552..42.976 rows=249588 loops=1
               ->  RemotePlan  (cost=20455.76..20455.76 rows=240092 width=158) (actual time=0.412..59.180 rows=249786 loops=1)
                     Output: t202.a, t202.b, t202.c, t202.d
                     Shard: 4     Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t202`  as t2
                     Worker 1: actual time=0.412..59.180 rows=249786 loops=1
               ->  RemotePlan  (cost=20366.49..20366.49 rows=240683 width=158) (actual time=0.431..50.371 rows=250376 loops=1)
                     Output: t201.a, t201.b, t201.c, t201.d
                     Shard: 3     Remote SQL: SELECT t2.a,t2.b,t2.c,t2.d FROM  `postgres_$$_public`.`t201`  as t2
         ->  Hash  (cost=197.00..197.00 rows=5000 width=12) (actual time=2.877..2.877 rows=5000 loops=4)
               Output: t1.a, t1.b
               Buckets: 8192  Batches: 1  Memory Usage: 279kB
               Worker 0: actual time=2.309..2.309 rows=5000 loops=1
               Worker 1: actual time=3.785..3.785 rows=5000 loops=1
               Worker 2: actual time=2.951..2.951 rows=5000 loops=1
               ->  RemotePlan  (cost=197.00..197.00 rows=5000 width=12) (actual time=0.512..1.872 rows=5000 loops=4)
                     Output: t1.a, t1.b
                     Shard: 4     Remote SQL: SELECT t1.a,t1.b FROM  `postgres_$$_public`.`t1`
                     Worker 0: actual time=0.379..1.417 rows=5000 loops=1
                     Worker 1: actual time=0.565..2.402 rows=5000 loops=1
                     Worker 2: actual time=0.643..2.105 rows=5000 loops=1
 Planning Time: 0.419 ms
 Execution Time: 112.690 ms
(44 rows)
```

KunlunBase

Q & A

KunlunBase

summerxwu