GREENPLUM DATABASE®

一个用于分析，机器学习和AI的开源大规模并行数据平台

杨峻峰
Greenplum内核研发工程师

2021年5月

GREENPLUM DATABASE®

Agenda

- Greenplum的历史
- Greenplum的基本架构
- 重要扩展组件
- 最近的工作
- 开源社区
- Q & A

GREENPLUM DATABASE®

Greenplum历史

GREENPLUM DATABASE®

Greenplum的基本架构

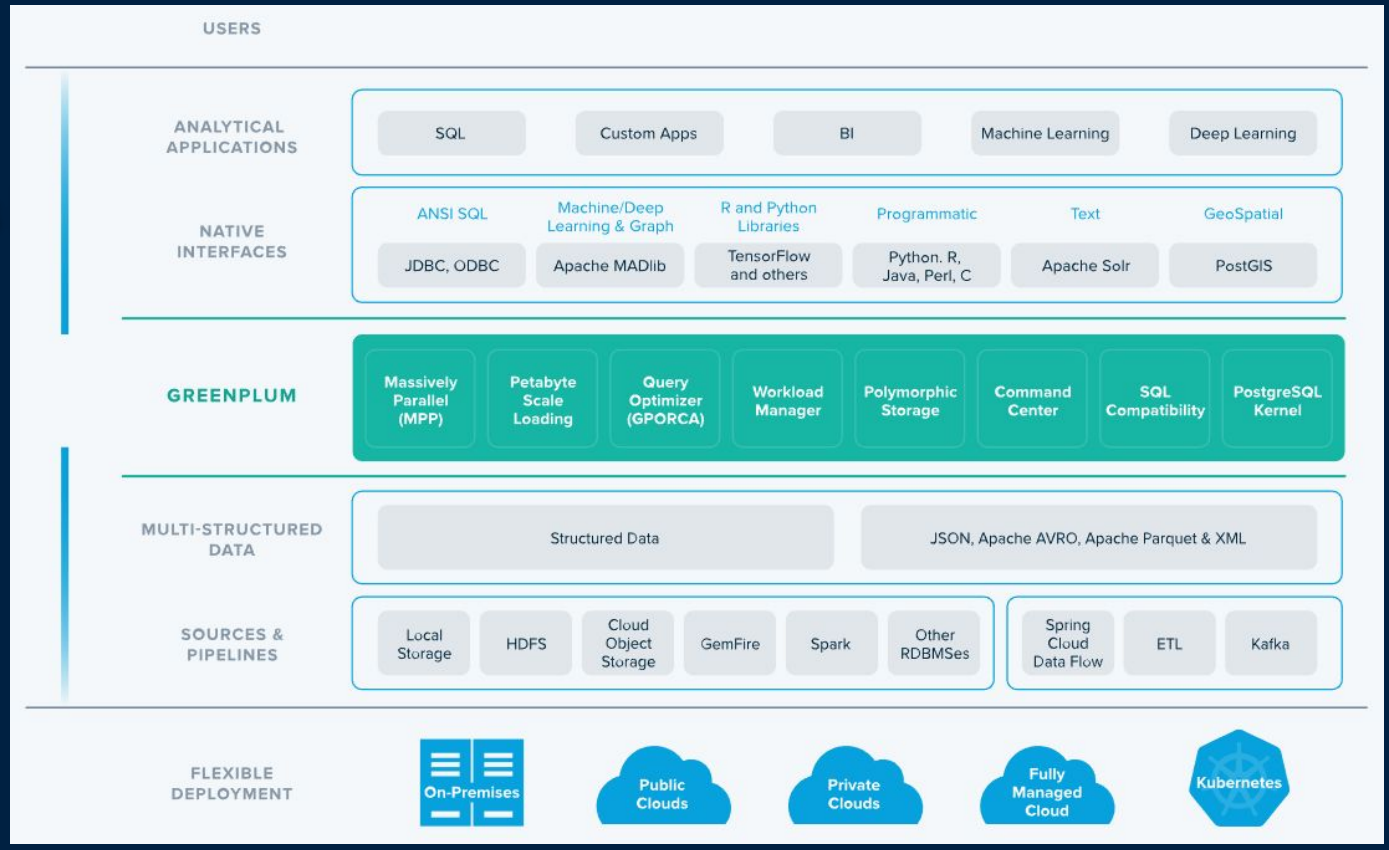
Greenplum是什么

PostgreSQL

+ Massively Parallel Process

+ Extra features

+ And addons



Greenplum 是一个开源项目

Greenplum 是一个开源项目

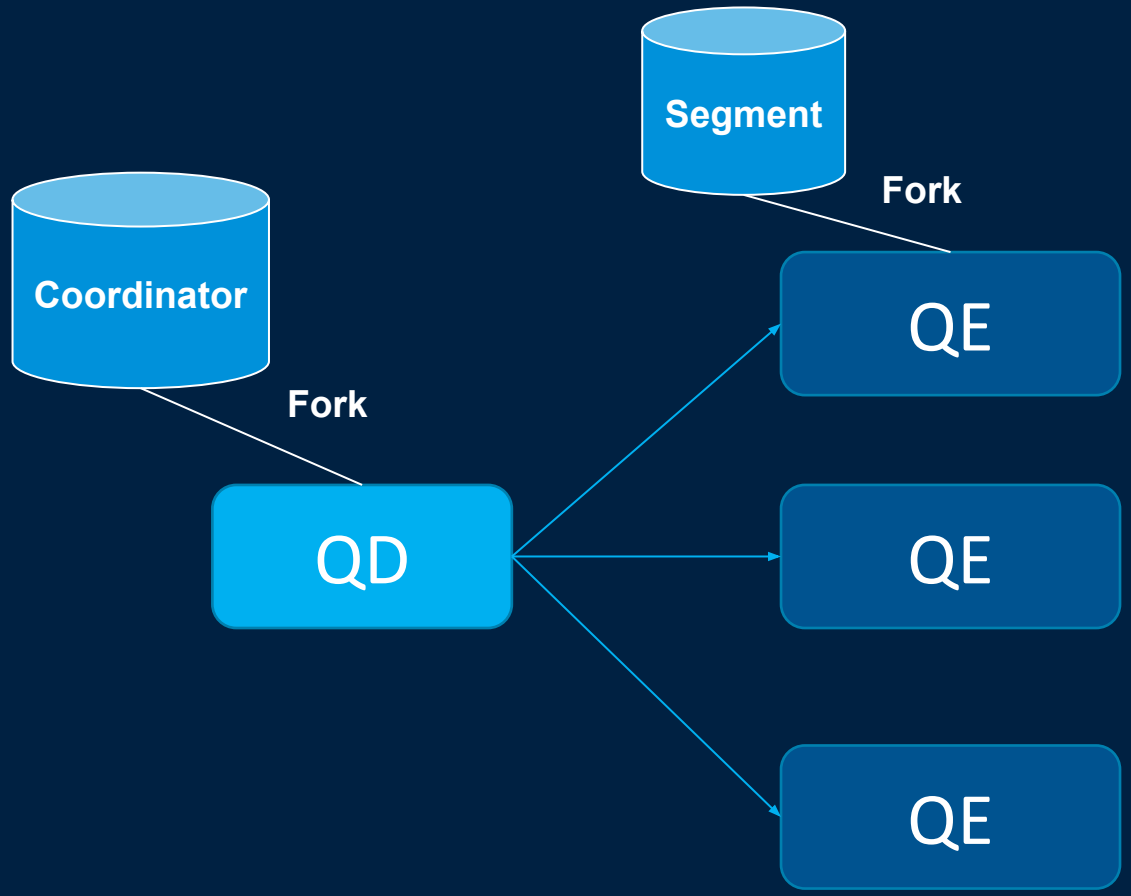
部署

- 一个Coordinator，数个Segment 实例（根据用户需求）
- 一台主机上可以部署多个Segment实例
  - 取决于主机的相关配置
- Mirroring 实现高可用
- Gb级别的交换机
- 一般部署在防火墙之后

Coordinator/Segment

- Coordinator
  - 管理元数据
  - 接受客户端请求
- Segment
  - 存储数据
  - 执行计算任务

Query Dispatching



QD = Query Dispatcher

QE = Query Executor

QD和QE之前的通信协议

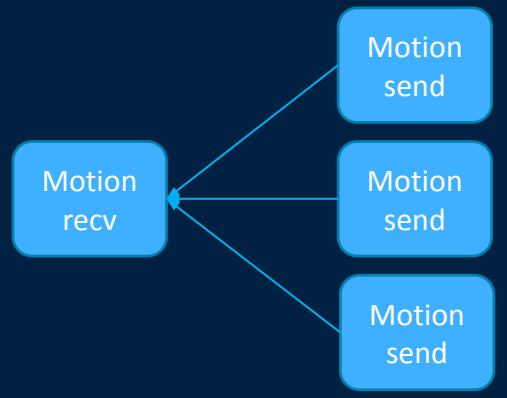
- 通信链接控制
  - libpq

- 数据传输 Interconnect
  - TCP
  - 自定义 UDP

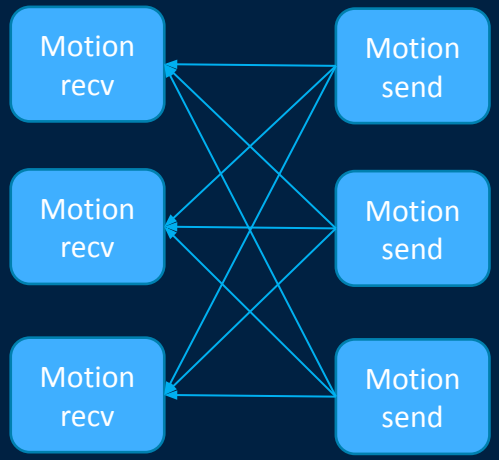
Motion节点

- Gather motion (N->1)
- Broadcast motion(N->N)
- Redistribute motion(N->N)

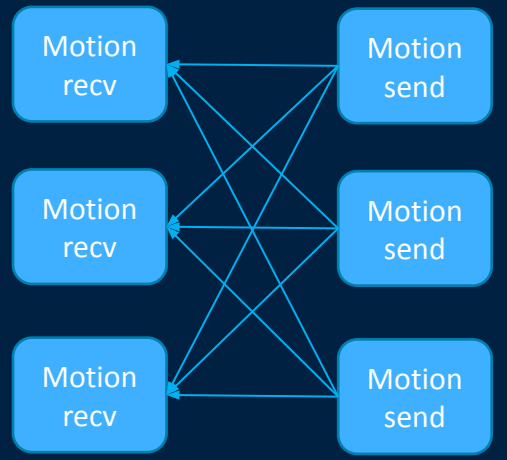
Gather motion



Redistribute motion



Broadcast motion



分布式查询计划

Postgres 查询计划：

```
postgres=# explain select * from foo;
        QUERY PLAN
-----------------------------------
Seq Scan on foo
```

Greenplum 查询计划

```
postgres=# explain select * from foo;
        QUERY PLAN
-----------------------------------
Gather Motion 3:1 (slice1; segments: 3)
        -> Seq Scan on foo
```

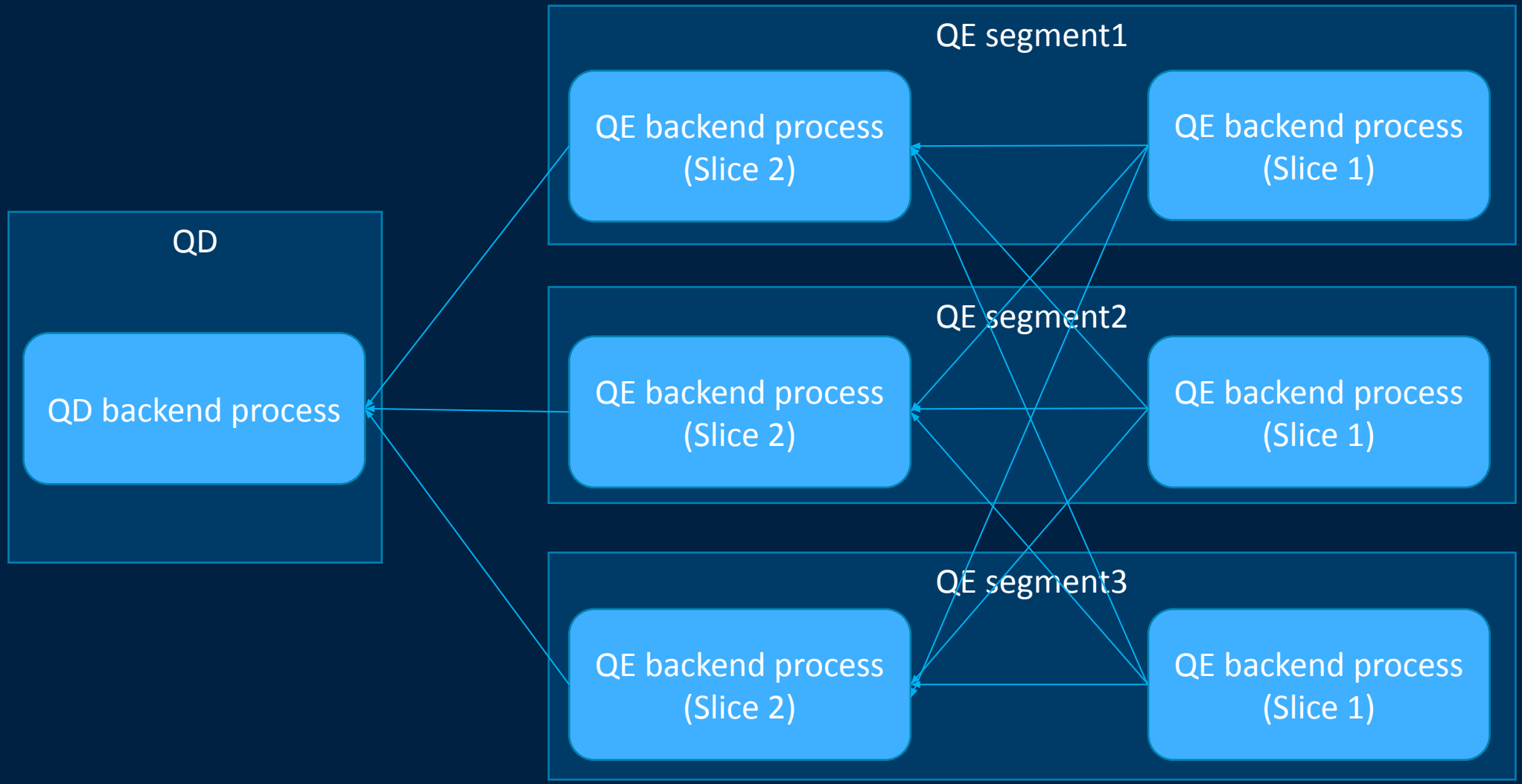
- 查询计划从QD分发给每个QE节点
- 查询结果通过Motion从QE节点返回给QD

一个更复杂的查询

```
create table t1(a int, c int) distributed by (a);
create table t2(a int, c int) distributed by (a);

demo=# explain select * from t1, t2 where t1.c = t2.c;
                                QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice2; segments: 3)
   ->  Hash Join
         Hash Cond: t1.c = t2.c
         ->  Seq Scan on t1
         ->  Hash
               ->  Broadcast Motion 3:3  (slice1; segments: 3)
                     ->  Seq Scan on t2
```

一个更复杂的查询

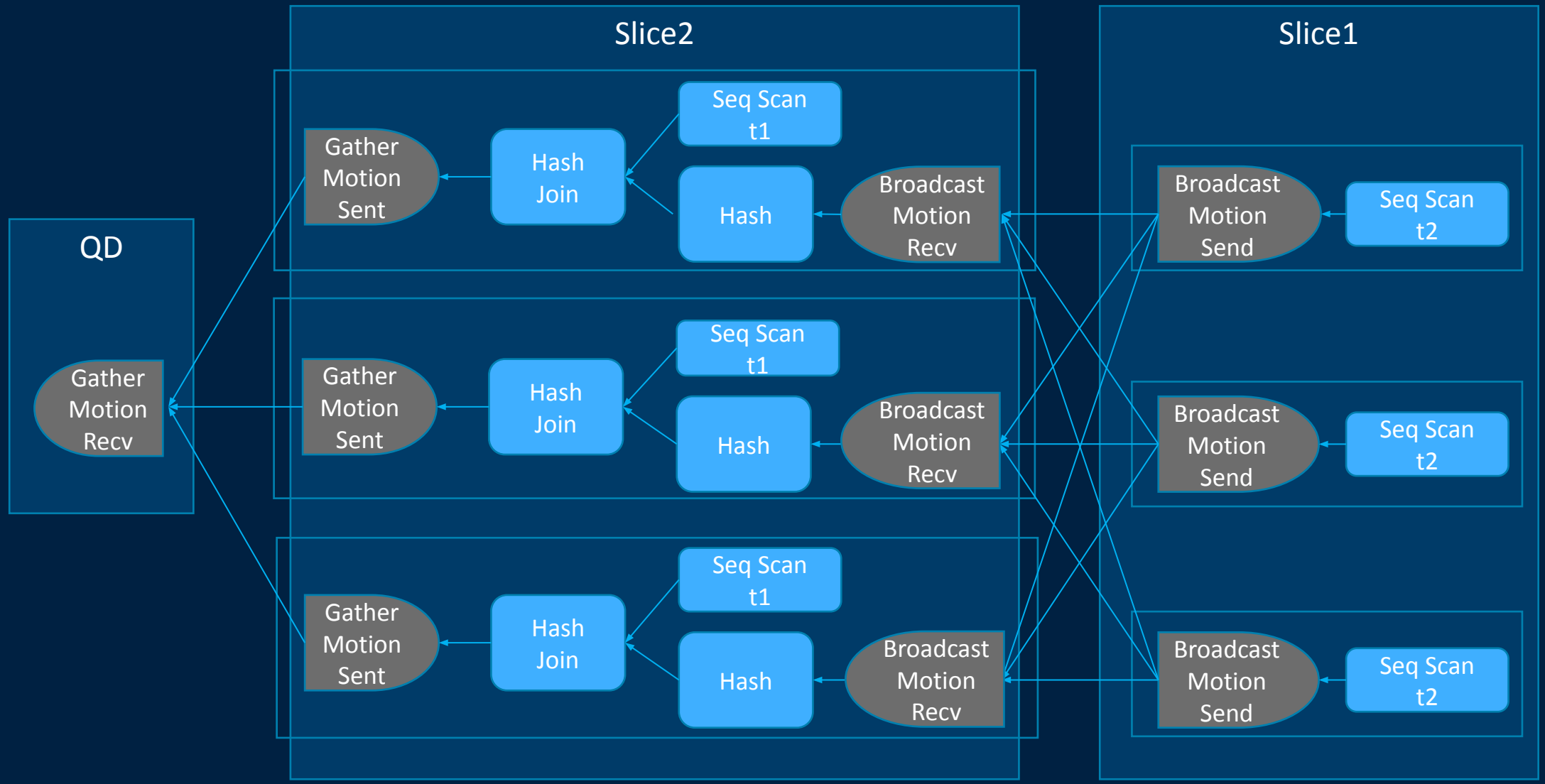

QD
QD backend process
QE segment1
QE backend process (Slice 2)
QE backend process (Slice 1)
QE segment2
QE backend process (Slice 2)
QE backend process (Slice 1)
QE segment3
QE backend process (Slice 2)
QE backend process (Slice 1)

一个更复杂的查询

```
create table t1(a int, c int) distributed by (a);
create table t2(a int, c int) distributed by (a);

demo=# explain select * from t1, t2 where t1.c = t2.c;
                                QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice2; segments: 3)
   ->  Hash Join
         Hash Cond: t1.c = t2.c
         ->  Seq Scan on t1
         ->  Hash
               ->  Broadcast Motion 3:3  (slice1; segments: 3)
                     ->  Seq Scan on t2
```

一个更复杂的查询

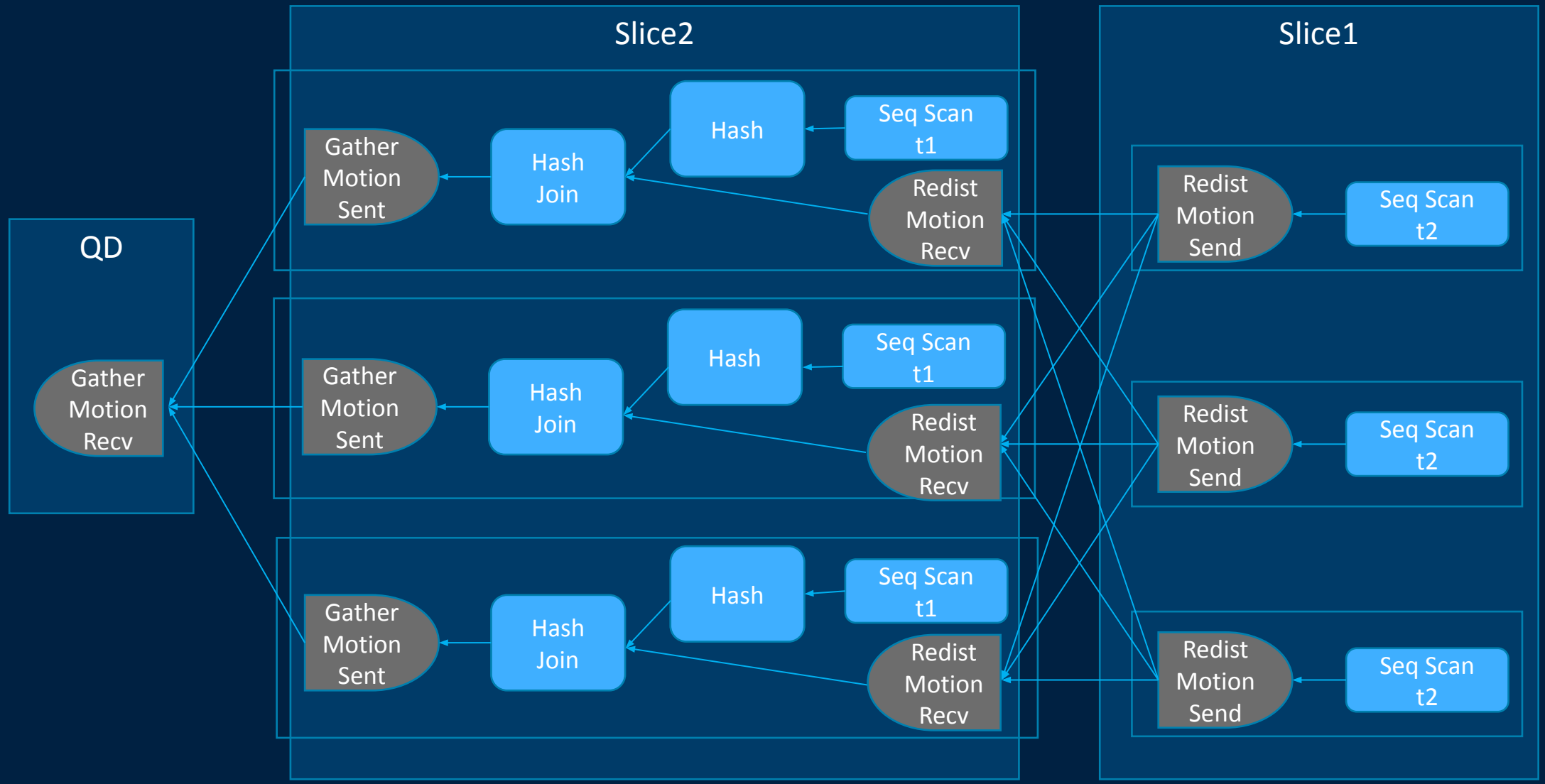

Slice2
Slice1
QD
Gather Motion Recv
Gather Motion Sent
Hash Join
Seq Scan t1
Hash
Broadcast Motion Recv
Broadcast Motion Send
Seq Scan t2

一个更复杂的查询

```
create table t1(a int, c int) distributed by (a);
create table t2(a int, c int) distributed by (a);

demo=# explain select * from t1, t2 where t1.a = t2.c;
                                QUERY PLAN
-------------------------------------------------------------------------------------------
 Gather Motion 3:1  (slice2; segments: 3)
   ->  Hash Join
         Hash Cond: t2.c = t1.a
         ->  Redistribute Motion 3:3  (slice1; segments: 3)
               Hash Key: t2.c
               ->  Seq Scan on t2
         ->  Hash
               ->  Seq Scan on t1
```

一个更复杂的查询

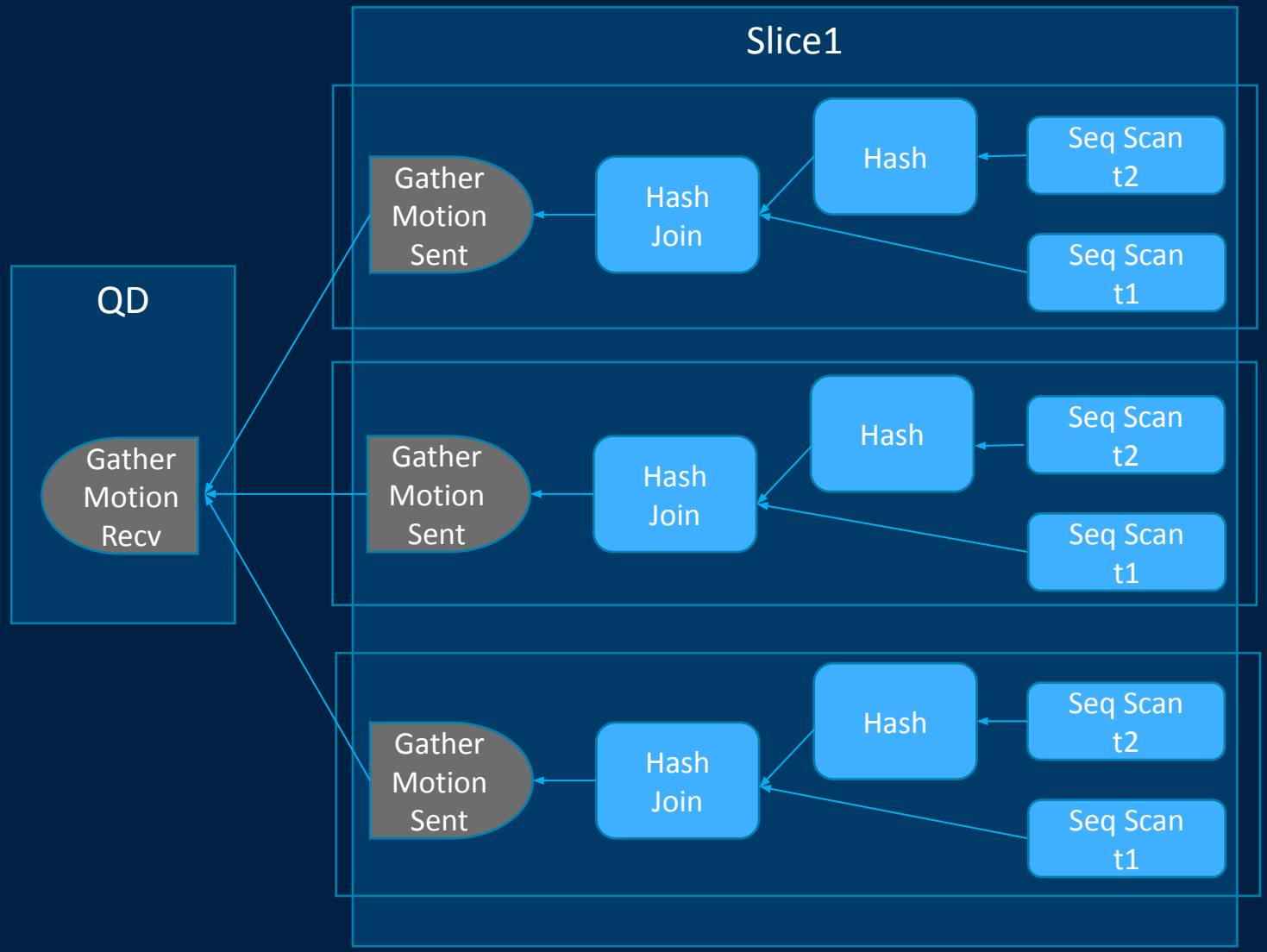

Slice2
Slice1
QD
Gather Motion Recv
Gather Motion Sent
Hash Join
Hash
Seq Scan t1
Redist Motion Recv
Redist Motion Send
Seq Scan t2

一个更复杂的查询

```
create table t1(a int, c int) distributed by (a);
create table t2(a int, c int) distributed by (a);

demo=# explain select * from t1, t2 where t1.a = t2.a;
                              QUERY PLAN
-----------------------------------------------------------------------------------
 Gather Motion 3:1  (slice1; segments: 3)
   ->  Hash Join
         Hash Cond: t1.a = t2.a
         ->  Seq Scan on t1
         ->  Hash
               ->  Seq Scan on t2
```

一个更复杂的查询


Slice1
QD
Gather Motion Recv
Gather Motion Sent
Hash Join
Hash
Seq Scan t2
Seq Scan t1
Gather Motion Sent
Hash Join
Hash
Seq Scan t2
Seq Scan t1
Gather Motion Sent
Hash Join
Hash
Seq Scan t2
Seq Scan t1

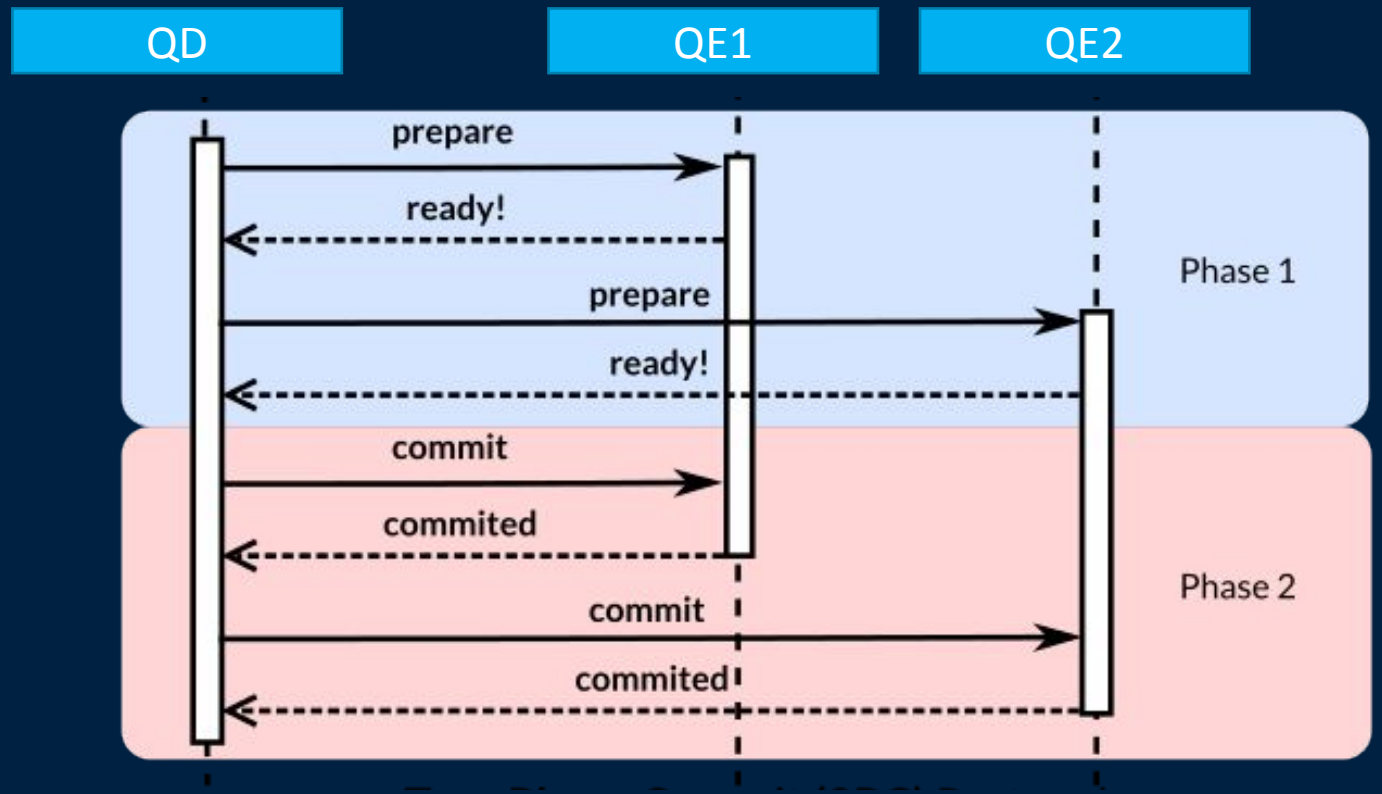
分布式查询计划

- 生成一个单机查询计划
- 跟踪每一步的distribution keys
- 在需要重分布的位置加入motion nodes
- 在聚合函数上实现多阶段聚合

分布式事务

- 两阶段提交
- QD作为两阶段提交的协调者
- 分布式快照保证一致性

两阶段提交


QD
QE1
QE2
prepare
ready!
prepare
ready!
Phase 1
commit
commited
commit
commited
Phase 2

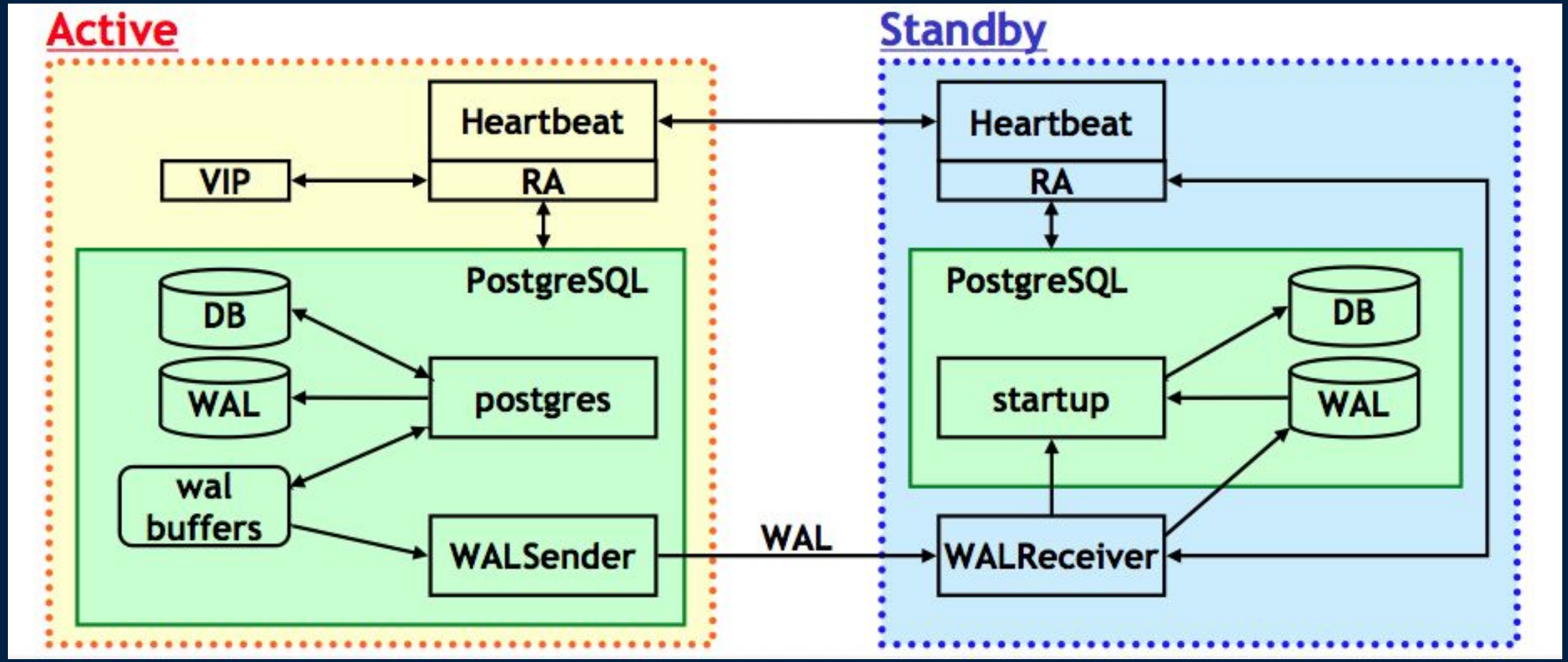
分布式快照

distribuTransactionTimeStamep	The unique times for this start of DTM
distribSnapshotId	Snap shot id
xmin	XID < xmin are visible to me
xmax	XID >= xmax are invisible to me
count	Lenth of xid array
inProgressXidArray	Array of distributed transactions in progress

Mirroring

- 每个segment有一个mirror segment作为备份
- Coordinator有一个standby 作为备份
- Wal replication

Wal replication



Partitioning

```
CREATE TABLE partition_test
(
  col1 int,
  col2 decimal,
  col3 text
)
distributed by (col1) partition by list(col2) (
  partition part1 VALUES(1,2,3,4,5,6,7,8,9,10),
  partition part2 VALUES(11,12,13,14,15,16,17,18,19,20),
  partition part3 VALUES(21,22,23,24,25,26,27,28,29,30),
  partition part4 VALUES(31,32,33,34,35,36,37,38,39,40),
  default partition def
);
```

自定义存储格式


多态存储
用户自定义数据存储格式
表'SALES'
11月
10月
9月
8月
7月
6月
5月
一年前
二年前
行存储
列存储
外部表
访问多列时速度快
支持高效更新和删除
AO 主要为插入而优化
列存储更适合压缩
查询列子集时速度快
不同列可以使用不同压缩方式：gzip (1-9)，quicklz，delta，RLE
历史数据和不常访问的数据存储在 HDFS 或者其他外部系统中
无缝查询所有数据
Text，CSV，Binary，Avro，Parquet 格式

GREENPLUM DATABASE®

Greenplum的重要扩展组件

- GPORCA
- GPSS
- Apache MADlib
- GPText
- ...

GPORCA

- 一种新式的查询优化器
- 使用C++实现
- 可以同时适配greenplum和HAWQ
- https://github.com/greenplum-db/gporca
- 对于复杂查询，需要更长的计算时间，但是通常能够给出更好的计划

Greenplum Streaming Server

- GPSS是一种ETL（提取，转换，加载）工具。

- GPSS服务器的实例使用Greenplum数据库可读的外部表从一个或多个客户端获取流数据，以将数据转换并将其插入到目标Greenplum表中。

- 数据源和数据格式特定于客户端。

MADlib

- 一个基于SQL的数据库内置的可扩展机的器学习库
- 强大的分析能力
- 将机器学习逻辑与数据库特定的实现细节分开
- 充分利用MPP架构处理海量数据
- Apache ASF上的顶级开源项目

https://github.com/apache/madlib

GPText

- 海量文本数据处理
- 高速并行全文本检索
- 支持半结构化和结构化数据（社交媒体，文档等）
- 易用的SQL接口
- 复杂文本分析（实体识别，情感分析等）
- 内置自然语言处理工具

示例 MADlib + GPText

KMeans example:

```
SELECT * FROM madlib.kmeans_plusplus(
    ‘<tfidf_table>’,
    ‘<tfidf_column’>,
    ‘<document_id’>,
    … MADLib Kmeans parameters …
);
```

使用gptext.terms得到terms vector，值为tfidf。

结合MADLib就可以去做更多的machine learning处理

GREENPLUM DATABASE®

最近的工作

- **[SIGMOD paper]** Greenplum: A Hybrid Database for Transactional and Analytical Workloads
- PG 12 merge
  - Partition logic (optimizer improvement)
  - BRIN index on AO
  - ...
- GP Auto Failover
- Auto Vacuum/Analyze

GREENPLUM DATABASE®

Greenplum开源社区

源码地址及代码结构

- https://github.com/greenplum-db/gpdb

gpMgmt /
包含用于管理集群的特定于Greenplum的命令行工具。像gpinit，gpstart，gpstop之类的脚本都在这里。它们主要是用Python编写的。

gpAux /
包含特定于Greenplum的版本管理脚本和依赖。还包含一些子模块。

gpcontrib /
与PostgreSQL contrib /目录非常相似，该目录包含gpfdist，PXF和gpmapreduce之类的扩展，它们是Greenplum特有的。

doc /
在PostgreSQL中，用户手册位于此处。在Greenplum中，用户手册是单独维护的，此处仅用于构建手册页的参考页。

源码地址及代码结构

gpdb-doc /
包含DITA XML格式的Greenplum文档。有关如何构建和使用文档的信息，请参考gpdb-doc / README.md。

ci /
包含GPDB持续集成系统的配置文件。

src /backend/ cdb /
Greenplum特定的后端模块。例如，segment之间的通信，将计划转变为可并行化计划，mirroring，分布式事务和快照管理等。cdb代表集群数据库-它是早期使用的工作名称。该名称不再使用，但保留cdb前缀。

src /backend/ gpopt /
包含所谓的翻译器库，用于将GPORCA优化器与Greenplum一起使用。转换器库是用C++代码编写的，并且包含用于在GPORCA使用的DXL格式与PostgreSQL内部表示之间转换计划和查询的结合代码。

源码地址及代码结构

src /backend/ gporca /
包含GPORCA优化器代码和测试。这是用C ++编写的。有关更多信息以及如何对GPORCA进行单元测试，请参见README.md。

src /backend/ fts /
FTS是在Coordinator节点中运行的进程，并定期轮询以维护每个segment节点的状态。

src/backend/*
其他内核代码，可根据文件夹名大致理解其功能模块，例如optimizer主要包含查询优化器相关代码。executor主要包含执行器相关代码。

丰富的社区活动


GREENPLUM DATABASE
《Greenplum: 从大数据战略到实现》读者技术沙龙
Pivotal+阿里云+蓝灯科技，与大咖面对面
陈淼 Pivotal Greenplum售后技术顾问
缪长风 阿里云 数据团队高级专家
吴籍 Pivotal Greenplum资深产品经理
周强 蓝灯数据 董事长兼CEO
Greenplum 的人工智能解决方案
Greenplum 如何成功版本开发
Greenplum +阿里云如何擦出花火
Greenplum 的金融行业应用案例
直播+线下 6月28日13:00
直播原价：10元
限量免费
仅限 100 名


GREENPLUM DATABASE
你的青梅称号
趣味学习，答题巩固
参加完直播活动，想测一测你的学习成果？
参加Greenplum小测试获取你的"青梅称号"吧！
分享你获得的称号图片至朋友圈，还有机会获得
Greenplum定制精致礼品！
青梅萌新
青梅闲客
青梅游侠
青梅雅士
青梅宗师
扫码加入活动


《Greenplum:
从大数据战略到实现》读者技术沙龙

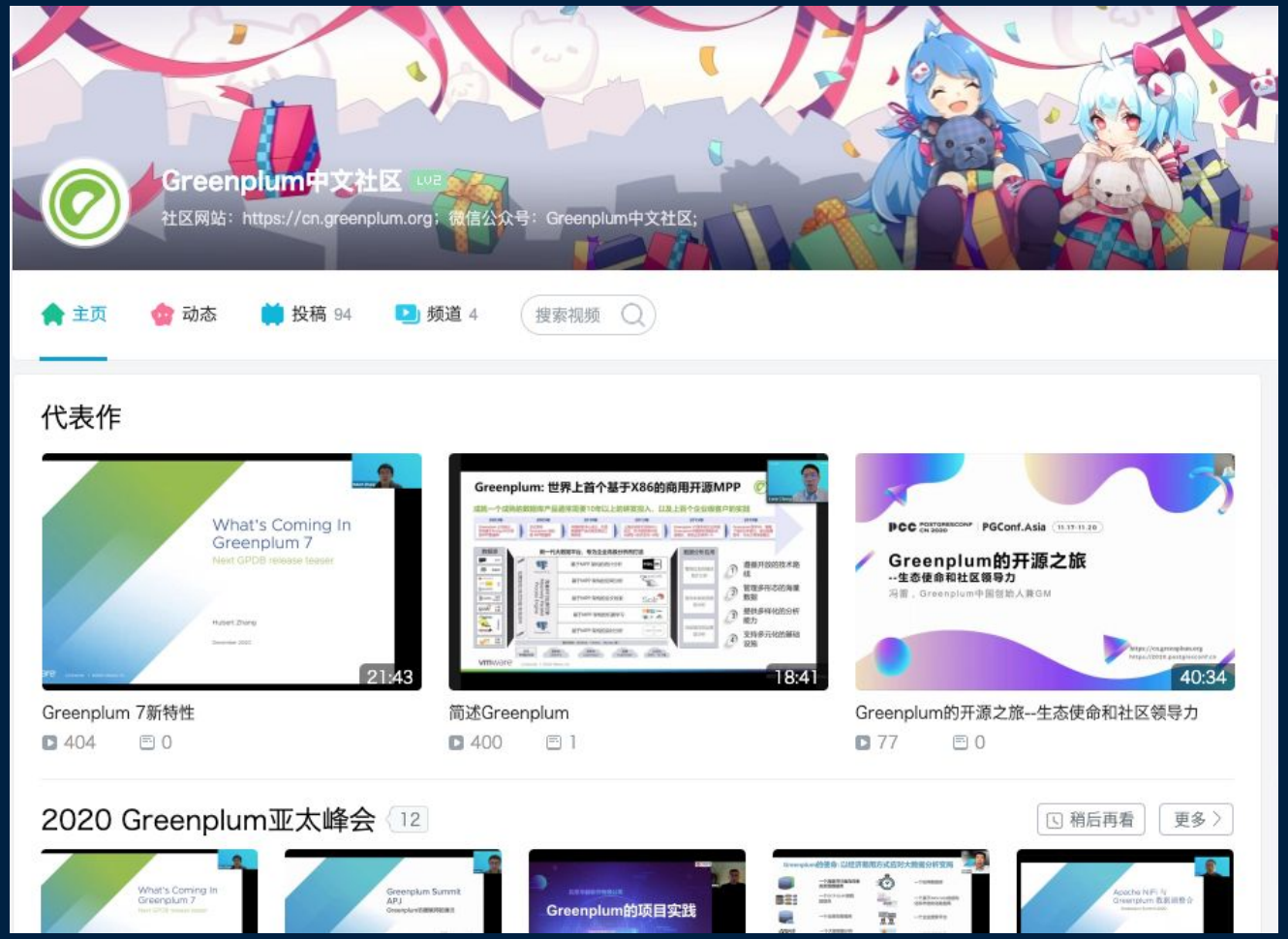

GREENPLUM DATABASE
Greenplum内核揭秘之
排序算法
钉钉直播 | 9月25日 16：00 - 17：00


GREENPLUM DATABASE
阿里云 aliyun.com
Greenplum内核揭秘之
执行引擎
钉钉直播 | 5月22日 16：00 - 17：00


阿里云 Pivotal CAICT
GREENPLUM DATABASE
走进杭州
技术交流大会
2019年8月10日 阿里巴巴西溪园区

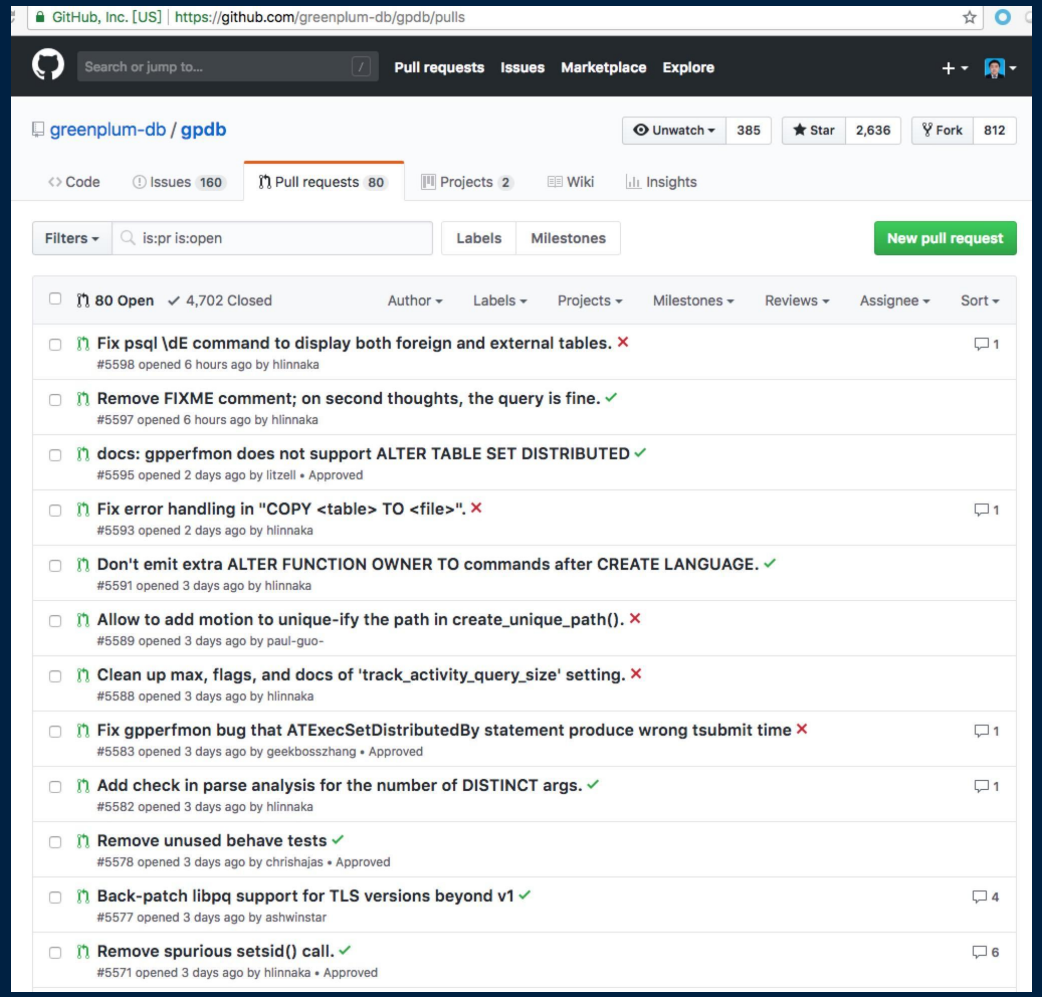
Greenplum内核课程

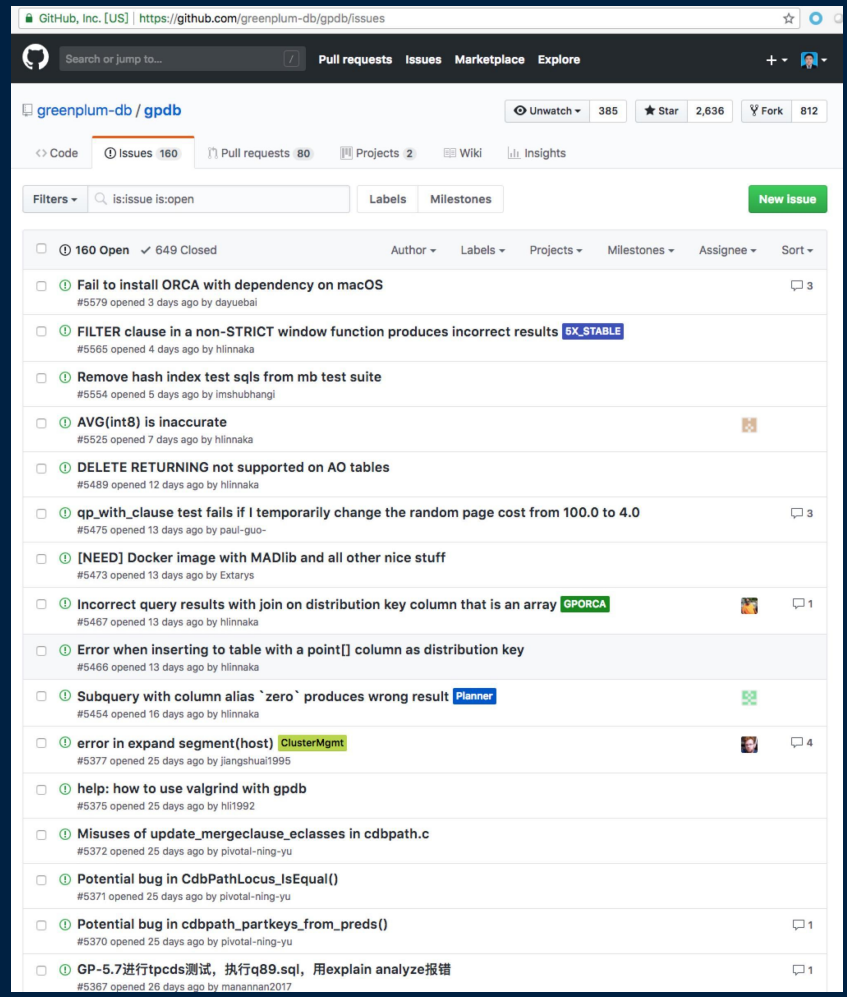
- https://space.bilibili.com/489184136



Greenplum的开发

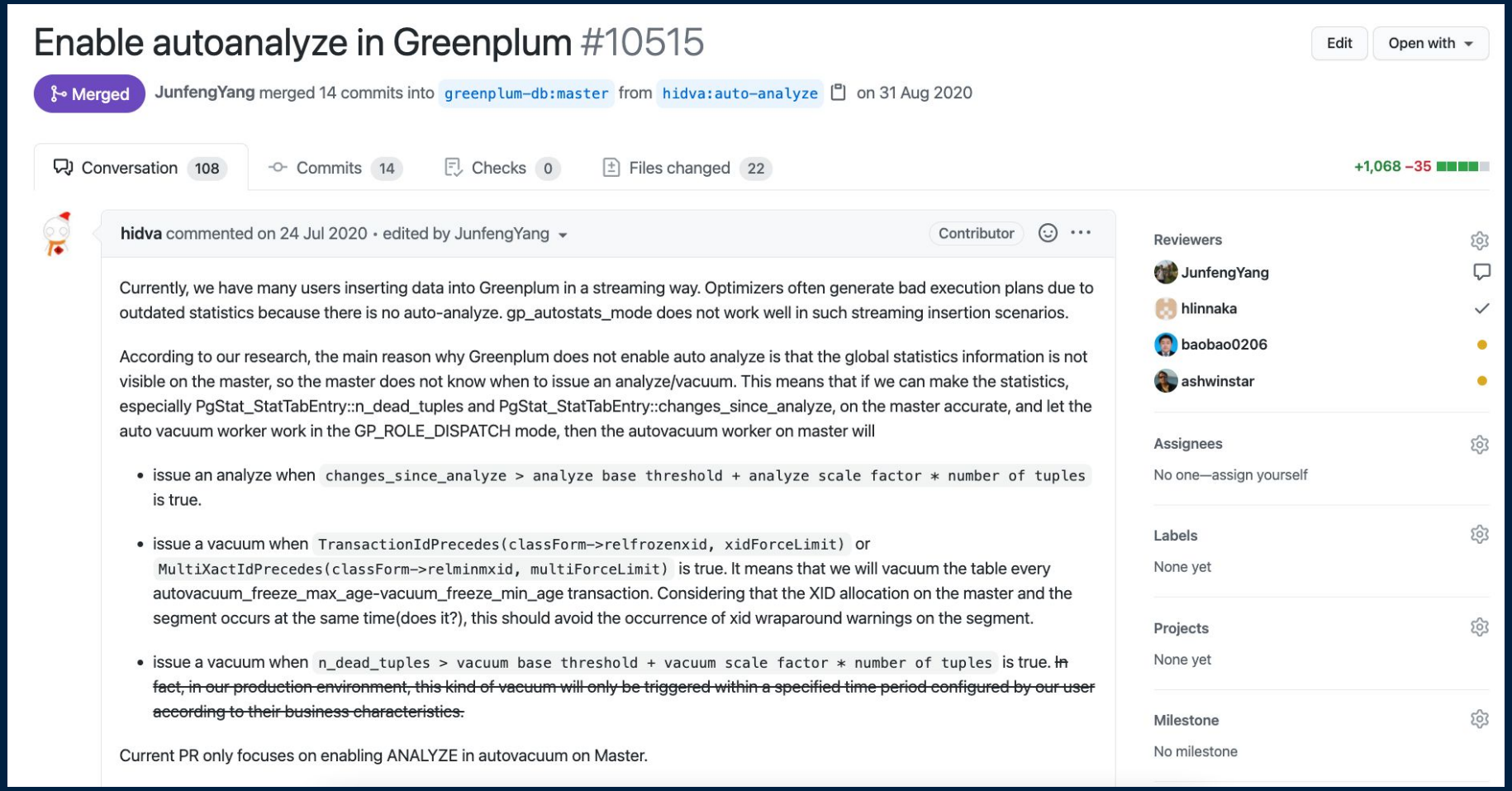
- Github Account
- Fork repository
- Code change and add tests.
- Pull requests
- Code review, validate checks and CI





贡献Greenplum社区

https://github.com/greenplum-db/gpdb/pull/10515



Enable autoanalyze in Greenplum #10515

Edit | Open with

Merged **JunfengYang** merged 14 commits into `greenplum-db:master` from `hidva:auto-analyze` on 31 Aug 2020

Conversation 108 | Commits 14 | Checks 0 | Files changed 22 | +1,068 −35

hidva commented on 24 Jul 2020 • edited by JunfengYang — Contributor

Currently, we have many users inserting data into Greenplum in a streaming way. Optimizers often generate bad execution plans due to outdated statistics because there is no auto-analyze. gp_autostats_mode does not work well in such streaming insertion scenarios.

According to our research, the main reason why Greenplum does not enable auto analyze is that the global statistics information is not visible on the master, so the master does not know when to issue an analyze/vacuum. This means that if we can make the statistics, especially PgStat_StatTabEntry::n_dead_tuples and PgStat_StatTabEntry::changes_since_analyze, on the master accurate, and let the auto vacuum worker work in the GP_ROLE_DISPATCH mode, then the autovacuum worker on master will

- issue an analyze when `changes_since_analyze > analyze base threshold + analyze scale factor * number of tuples` is true.
- issue a vacuum when `TransactionIdPrecedes(classForm->relfrozenxid, xidForceLimit)` or `MultiXactIdPrecedes(classForm->relminmxid, multiForceLimit)` is true. It means that we will vacuum the table every autovacuum_freeze_max_age-vacuum_freeze_min_age transaction. Considering that the XID allocation on the master and the segment occurs at the same time(does it?), this should avoid the occurrence of xid wraparound warnings on the segment.
- issue a vacuum when `n_dead_tuples > vacuum base threshold + vacuum scale factor * number of tuples` is true. ~~In fact, in our production environment, this kind of vacuum will only be triggered within a specified time period configured by our user according to their business characteristics.~~

Current PR only focuses on enabling ANALYZE in autovacuum on Master.

Reviewers
- JunfengYang
- hlinnaka
- baobao0206
- ashwinstar

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

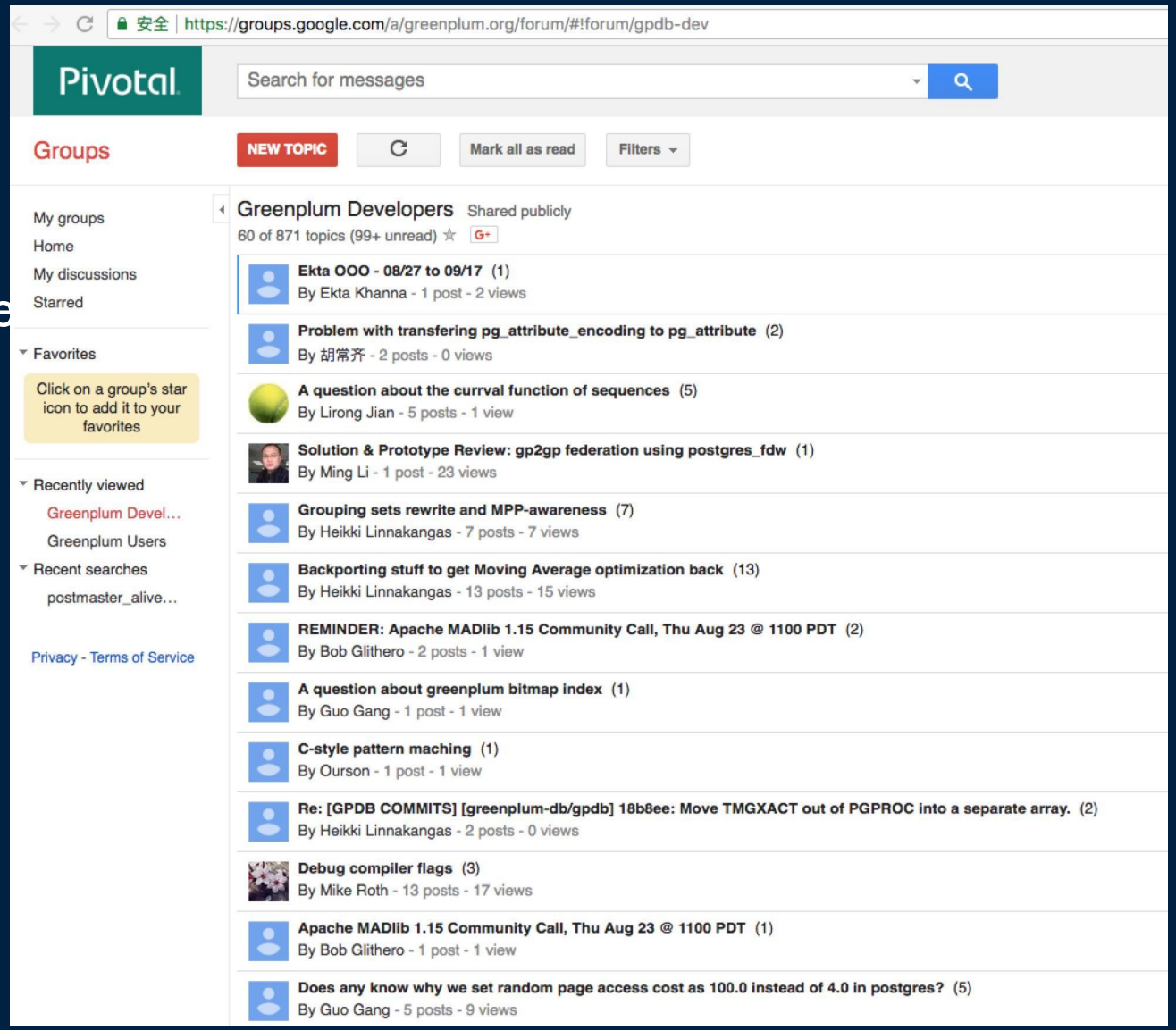
Milestone
No milestone

Greenplum参与社区

Github Issues

Public mailing lists on Google Groups:

- gpdb-dev
- gpdb-users



GREENPLUM DATABASE®

greenplum.org

- News, events
- Links to the github, project, mailing list

Q & A

Thank You