

Greenplum 数据库架构分析及5.x 新功能分享

杨瑜

Pivotal中国研发中心

日程

- Greenplum 数据库 (GPDB) 简介
- Greenplum 数据库 (GPDB) 架构
- Greenplum 数据库 (GPDB) 组件
- Greenplum 数据库 (GPDB) 执行流程
- Greenplum 数据库 (GPDB) 5.x

Greenplum 简介

GPDB: 为大数据存储、计算、挖掘而设计

- 标准 SQL 数据库: ANSI SQL 2008 标准, OLAP, JDBC/ODBC
- 支持ACID、分布式事务
- 分布式数据库: 线性扩展, 支持上百物理节点
- 企业级数据库: 全球大客户超过 1000+ 安装集群
- 百万行源代码, 超过10年的全球研发投入
- 开源数据库(greenplum.org), 良性生态系统

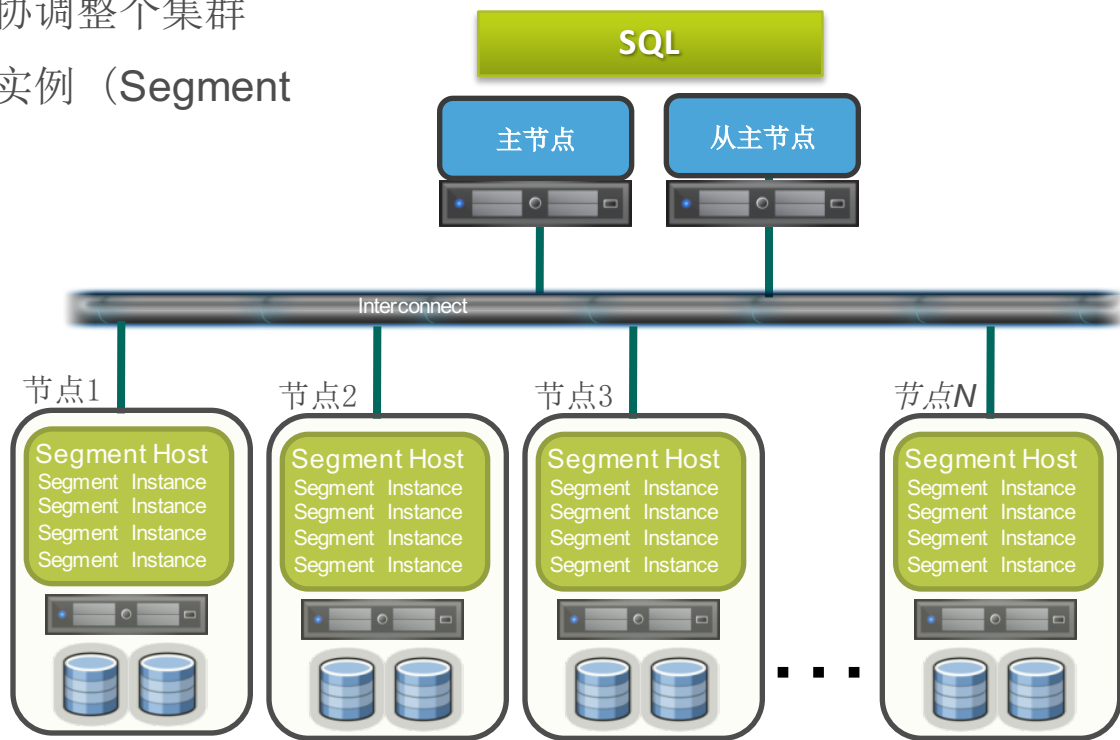
Greenplum 架构

平台概况

客户端访问和工具	客户端访问 ODBC, JDBC, OLEDB, etc.	第三方工具 BI 工具, ETL 工具 文本分析, 数据挖掘等	管理工具 GP Command Center GP Workload Manager
产品特性	加载 & 数据联邦 高速数据加载 近实时数据加载 任意系统数据访问	存储 & 数据访问 混合存储引擎 (行存&列存) 多种压缩, 多级分区表 索引 (B树, 位图, GiST) 安全性	语言支持 标准SQL支持, SQL 2003 OLAP扩展 支持 MapReduce 扩展编程语言 (Python, R, Java, Perl, C/C++)
服务	多级容错机制	在线系统扩展	任务管理
核心MPP架构	无共享大规模并行处理 先进的查询优化器 多态存储系统		并行数据流引擎 高速软数据交换机制 MPP Scatter/Gather 流处理

MPP (大规模并行处理) 无共享体系架构

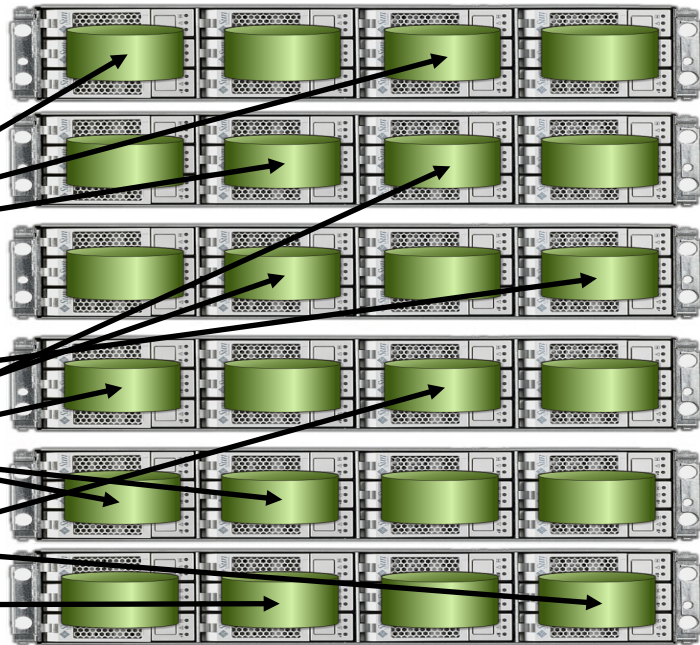
- 主节点和从主节点，主节点负责协调整个集群
- 一个数据节点可以配置多个节点实例 (Segment Instances)
- 节点实例并行处理查询 (SQL)
- 数据节点有自己的CPU、磁盘和内存 (Share nothing)
- 高速Interconnect处理持续数据流 (Pipelining)



数据分布：并行化的根基

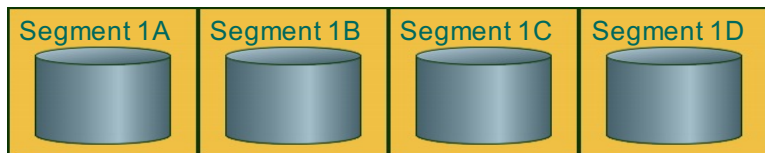
最重要的策略和目标是均匀分布数据到各个数据节点。

Order		
Order #	Order Date	Customer ID
43	Oct 20 2005	12
64	Oct 20 2005	111
45	Oct 20 2005	42
46	Oct 20 2005	64
77	Oct 20 2005	32
48	Oct 20 2005	12
50	Oct 20 2005	34
56	Oct 20 2005	213
63	Oct 20 2005	15
44	Oct 20 2005	102
53	Oct 20 2005	82
55	Oct 20 2005	55



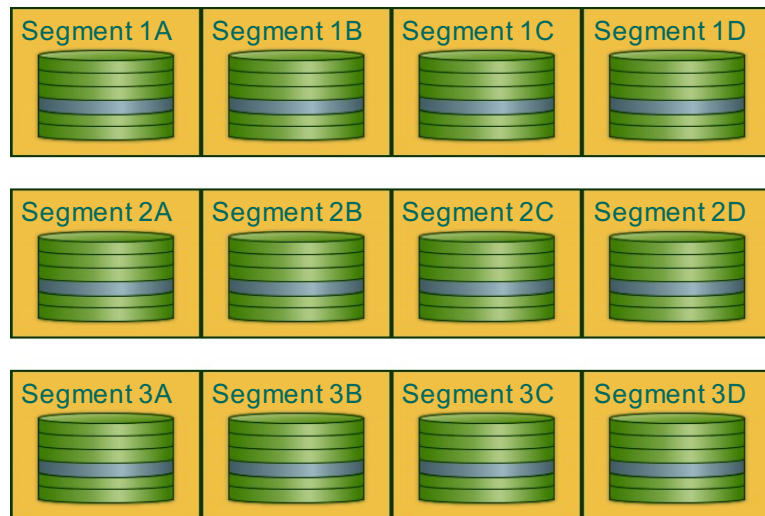
分布和分区

```
SELECT COUNT(*)  
FROM orders  
WHERE order_date >= 'Oct 20 2007'  
AND order_date < 'Oct 27 2007'
```



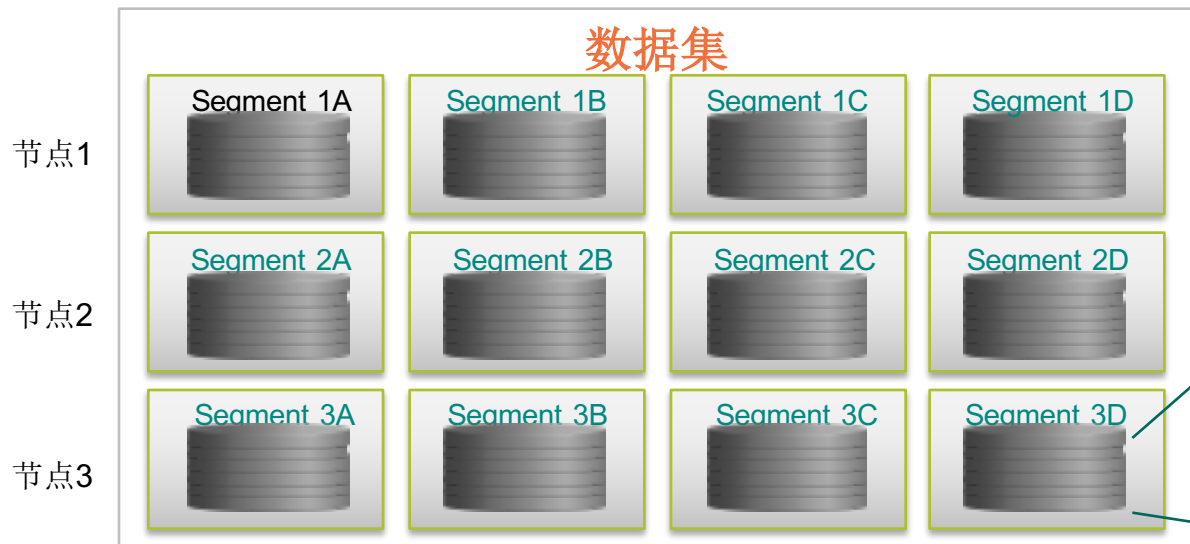
orders 表数据均匀分布于各个节点

&



仅仅扫描 orders 表相关的分区

多级分区存储

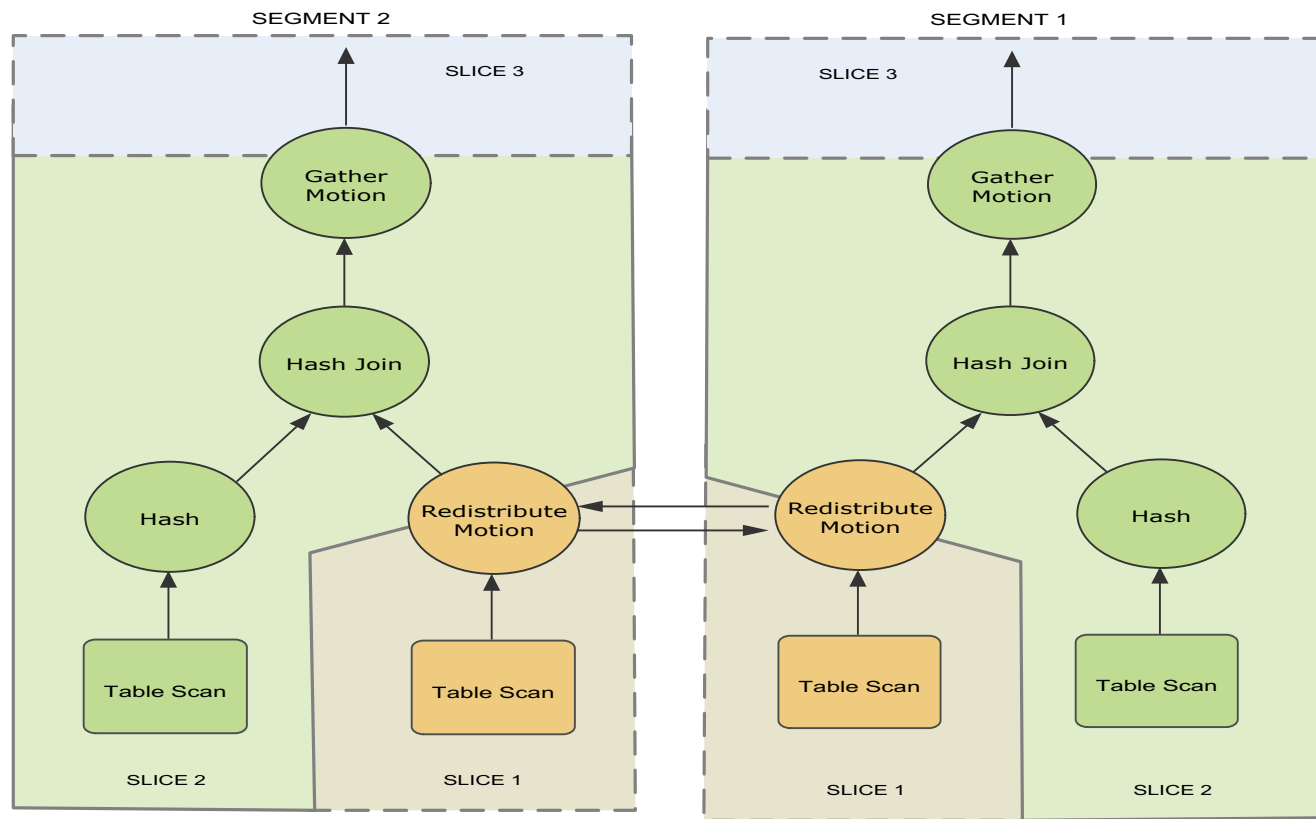


- 哈希Distribution: 数据均匀的分布到各个数据节点
- 范围分区: 数据节点内部, 根据多种规则分区, 降低扫描量

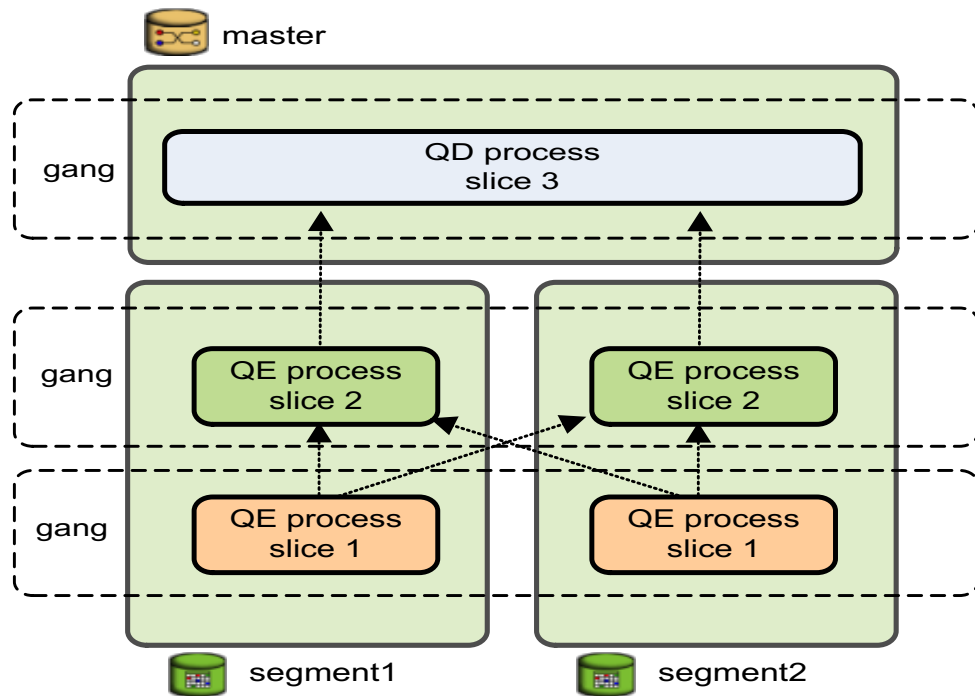


生成并行查询计划

```
SELECT customer,  
  
amount  
FROM sales  
JOIN customer  
USING (cust_id)  
WHERE date=2008;
```



执行并行计划



多态存储

用户自定义数据存储格式



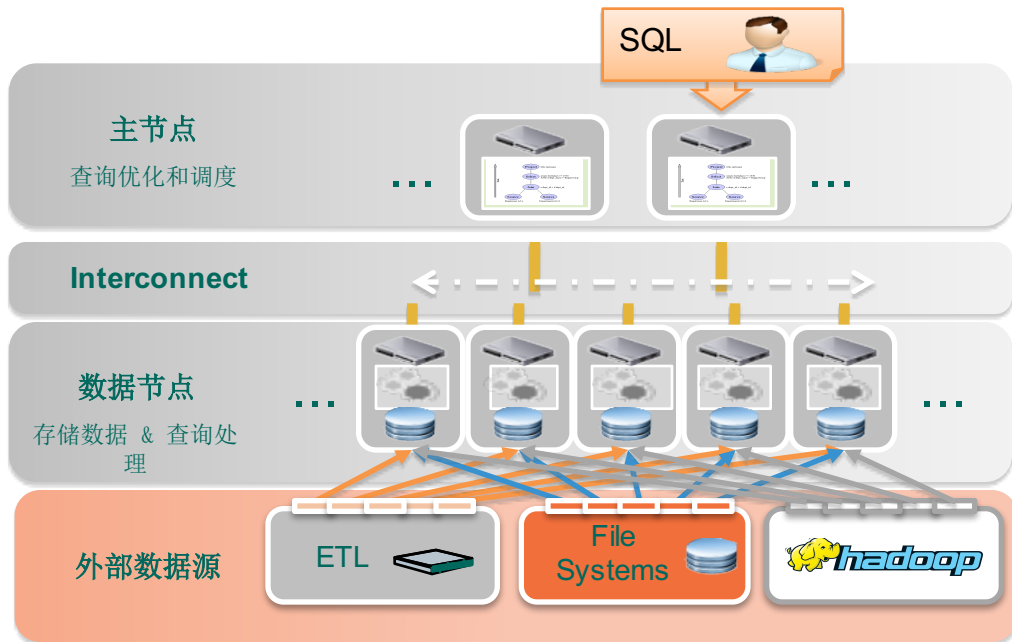
- 访问多列时速度快
- 支持高效更新和删除
- AO 主要为插入而优化

- 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

大规模并行数据加载

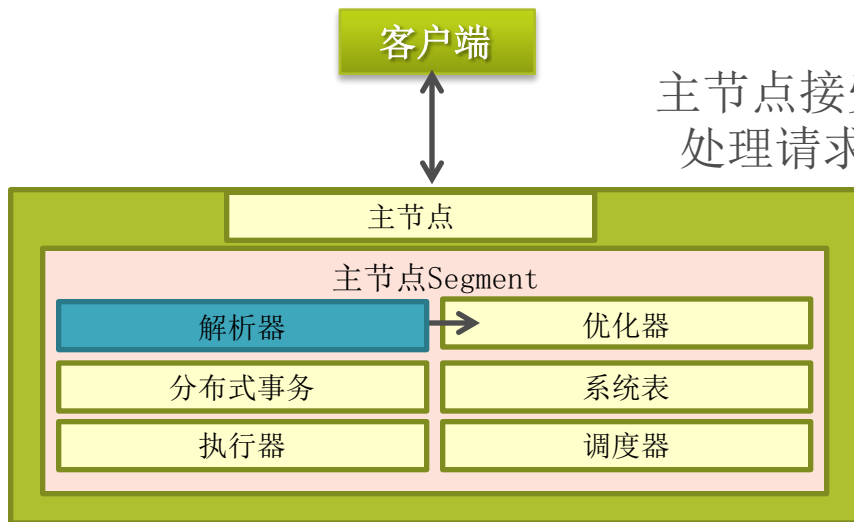
- 高速数据导入和导出
 - 主节点不是瓶颈
 - 10+ TB/小时/Rack
 - 线性扩展
- 低延迟
 - 加载后立刻可用
 - 不需要中间存储
 - 不需要额外数据处理
- 导入/导出 到&从：
 - 文件系统
 - 任意 ETL 产品
 - Hadoop 发行版



Greenplum 组件

解析器

解析器执行词法分析、语法分析并生成解析树

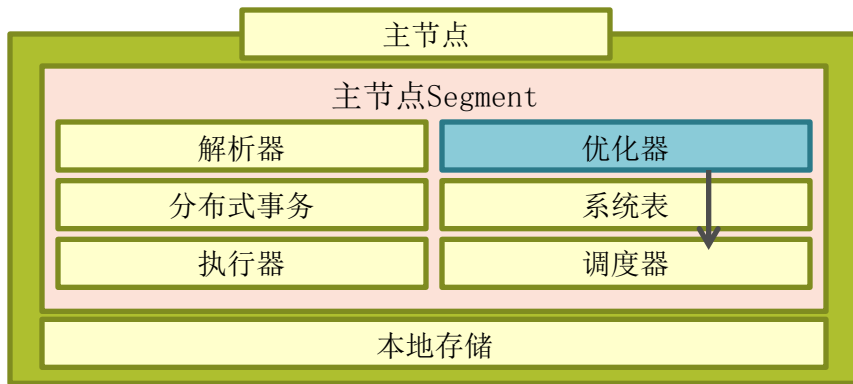


主节点接受客户连接，
处理请求，执行认证

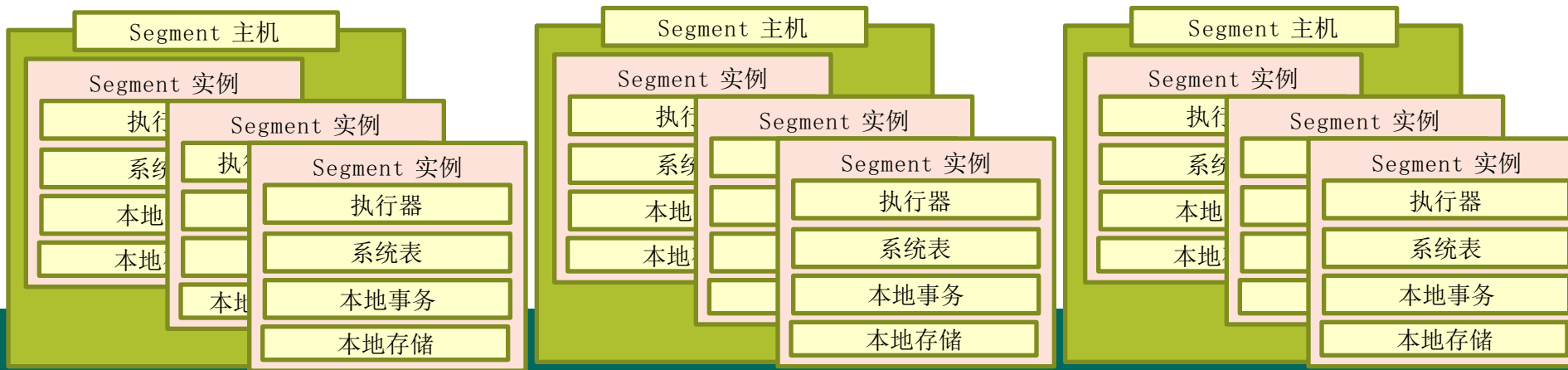
优化器

处理解析树，生成
查询计划

查询计划描述了如
何执行查询



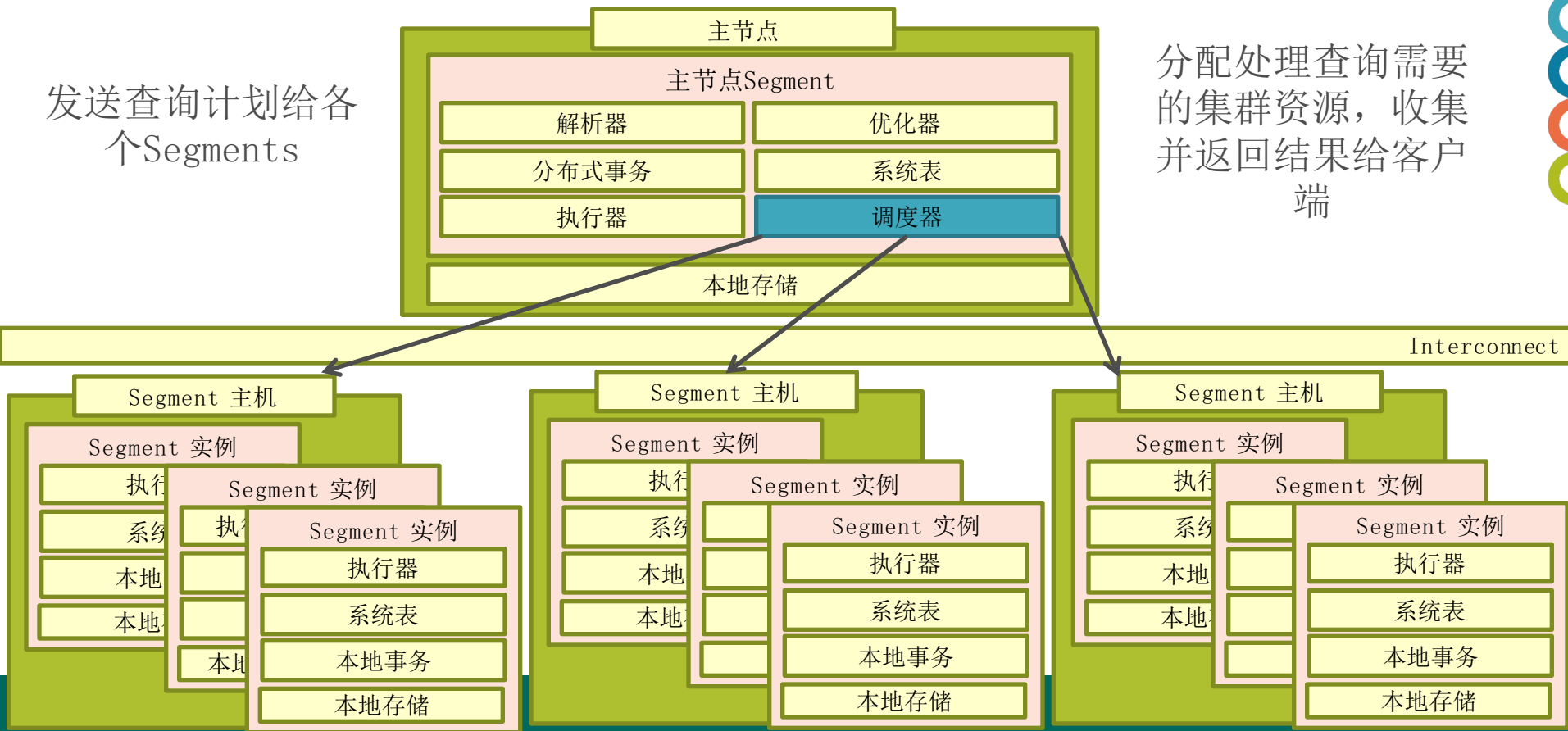
Interconnect



调度器

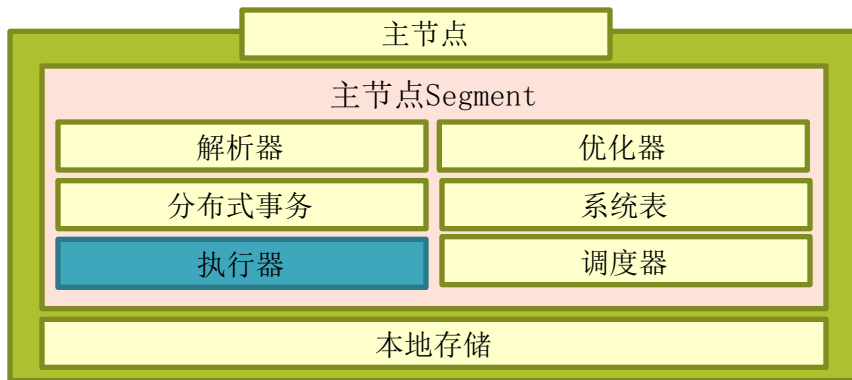
发送查询计划给各
个Segments

分配处理查询需要的
集群资源，收集
并返回结果给客户
端

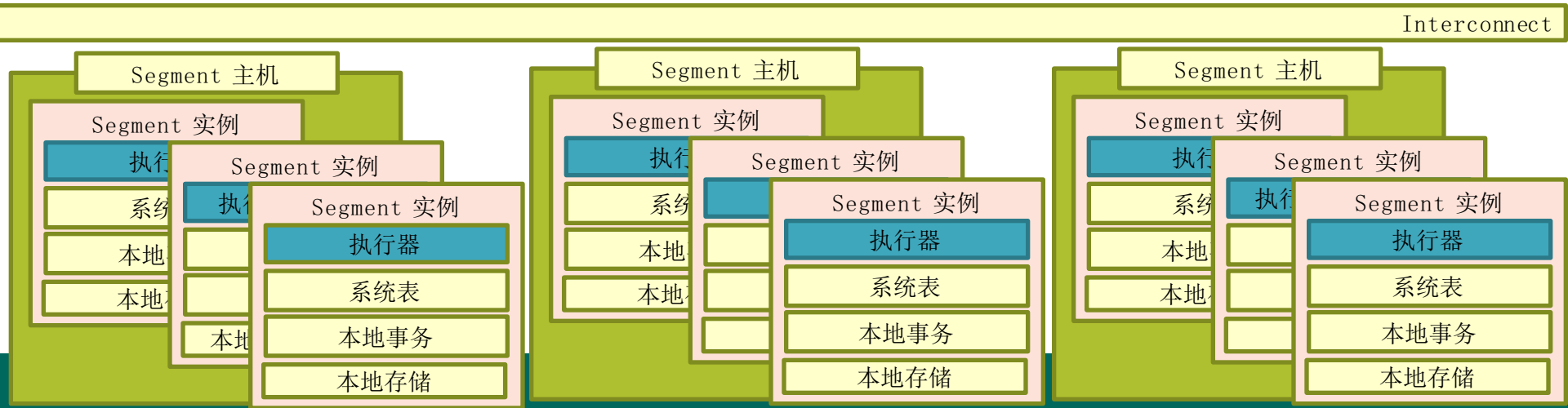


执行器

发送查询计划给各个Segments



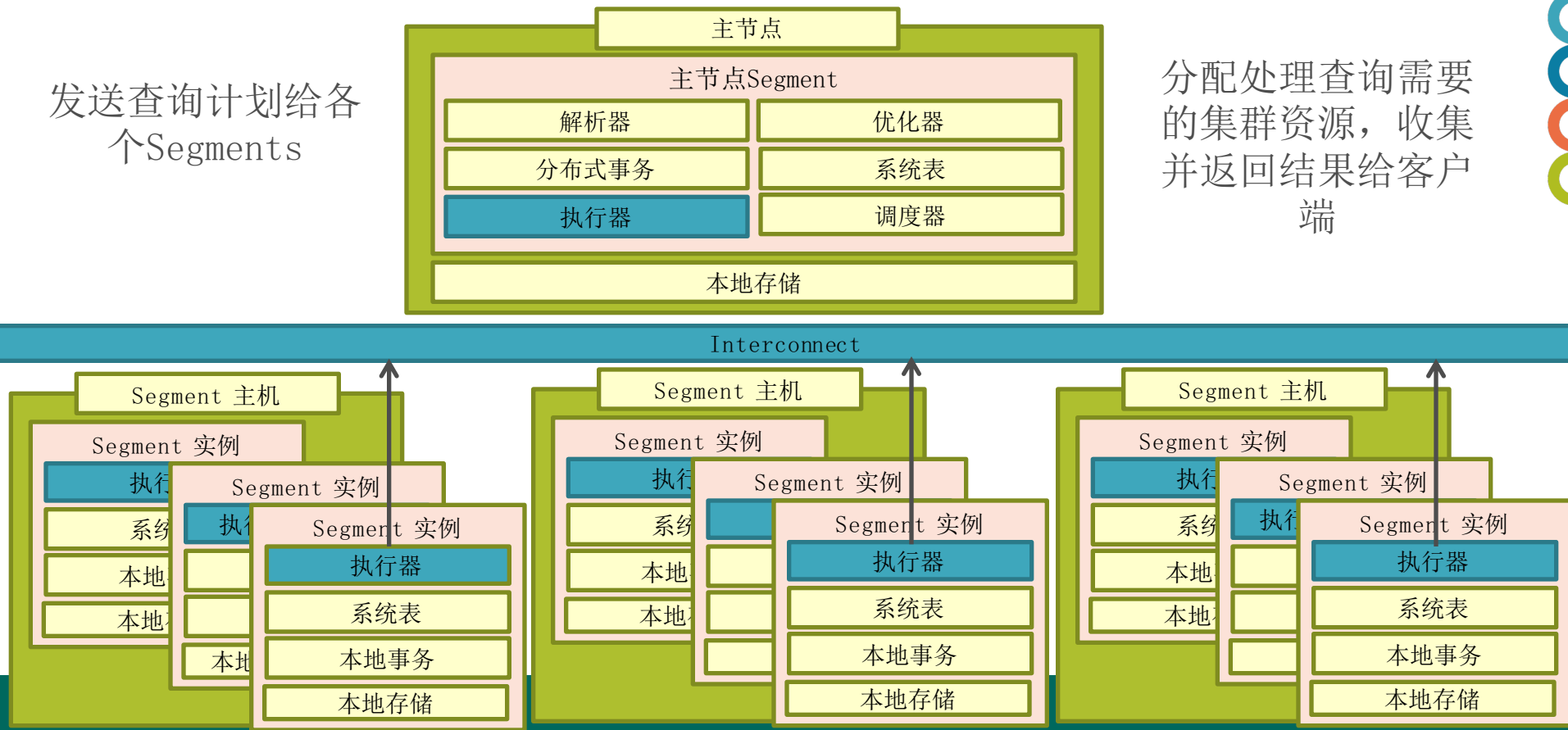
分配处理查询需要的集群资源，收集并返回结果给客户
端



Interconnect

发送查询计划给各个Segments

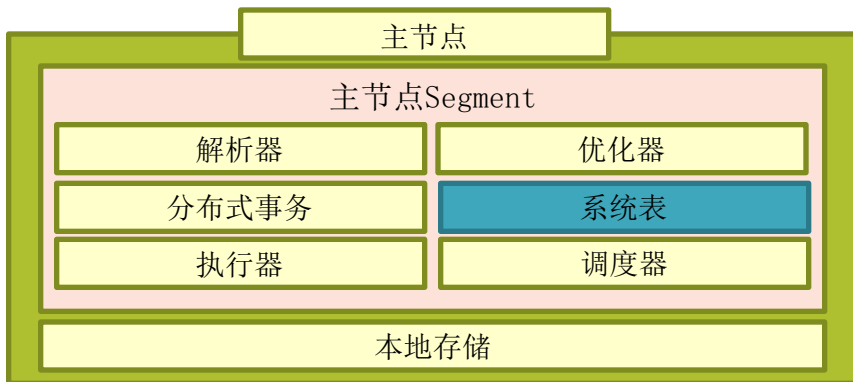
分配处理查询需要的集群资源，收集并返回结果给客户端



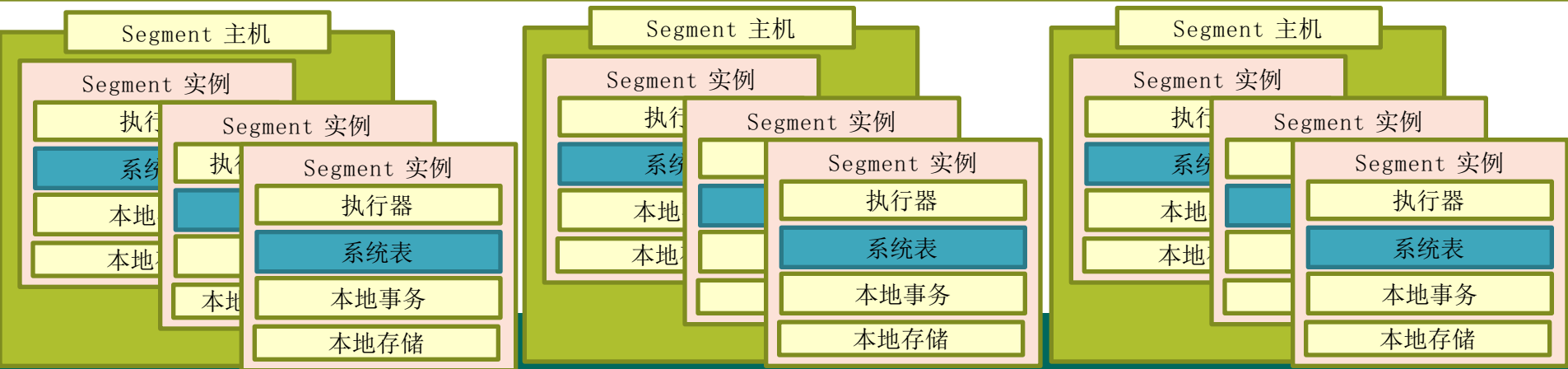
系统表

存储和管理数据库、
表、字段的元数据

每个节点保存一个
拷贝

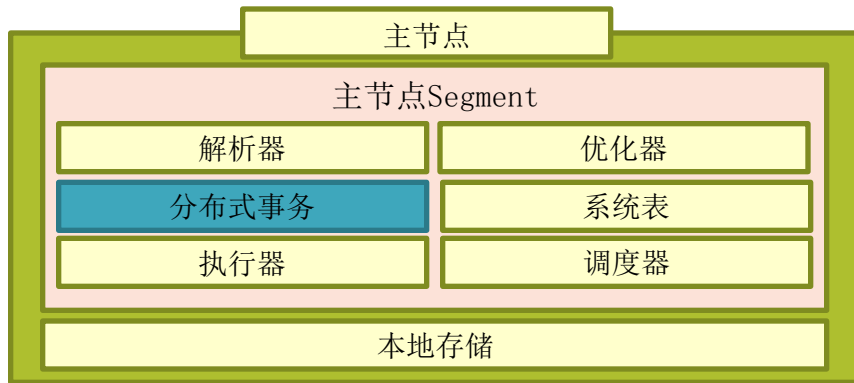


Interconnect

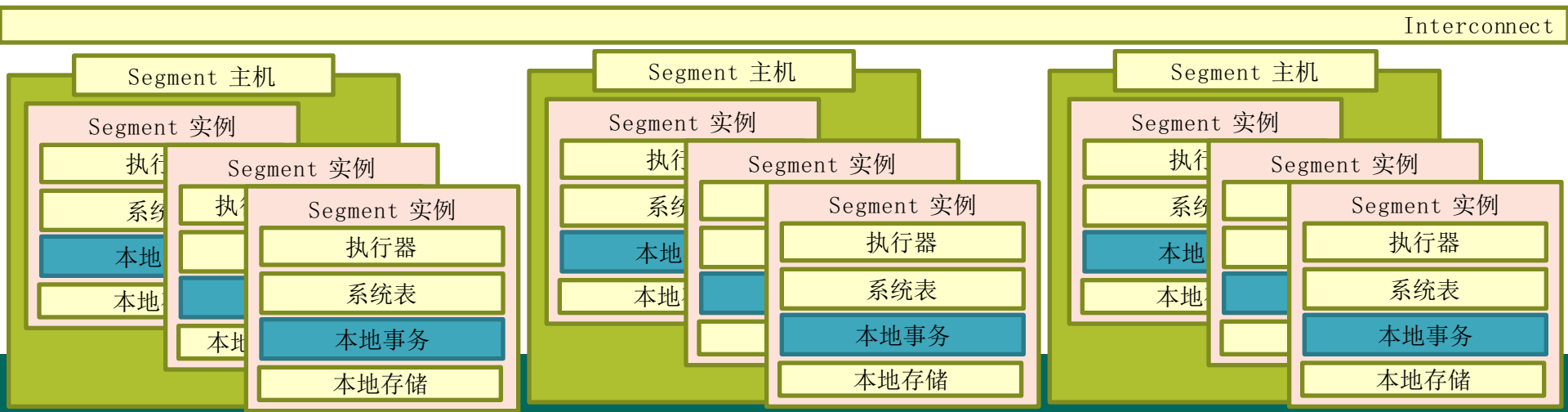


分布式事务

主节点上的分布式
事务管理器协调
Segment上的提交和
回滚操作

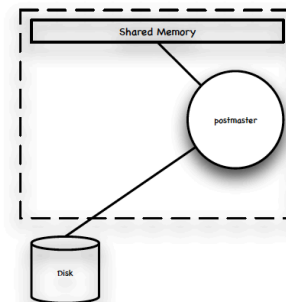
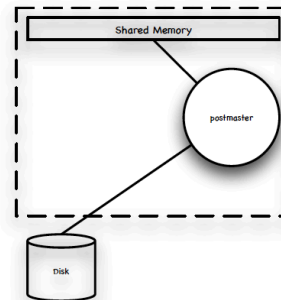
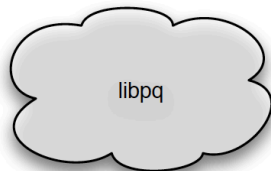
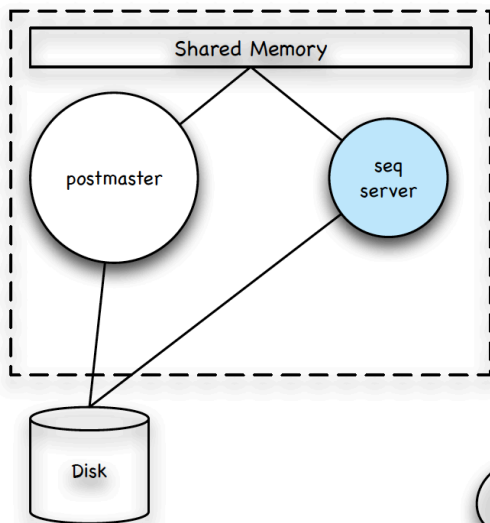


Segments 有自己的
事务日志，确定合
适提交或回滚自己
的事务

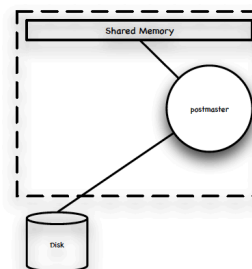
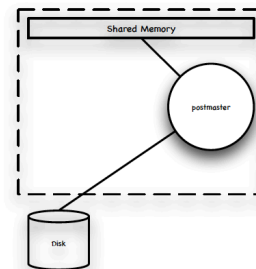
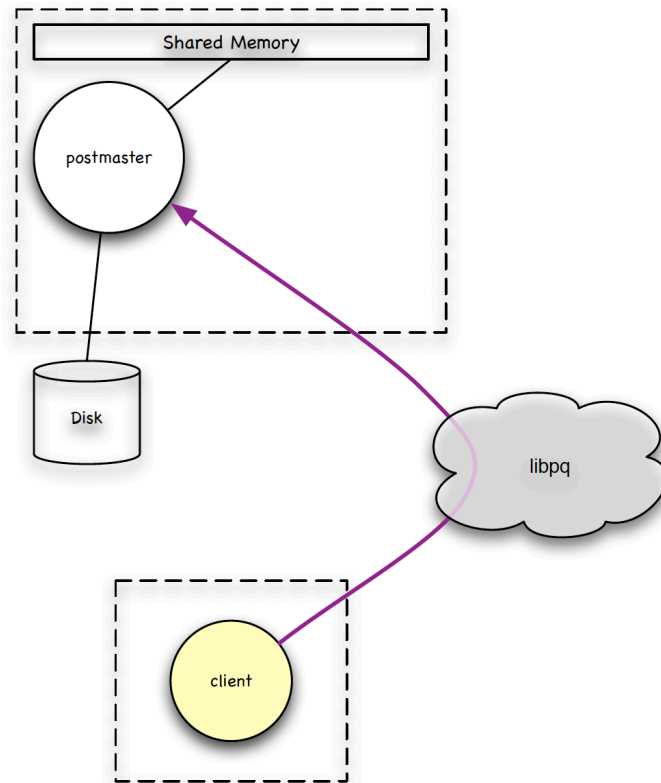


Greenplum 执行流程

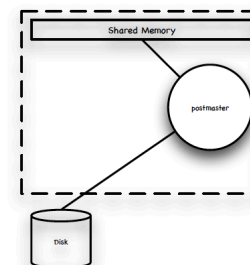
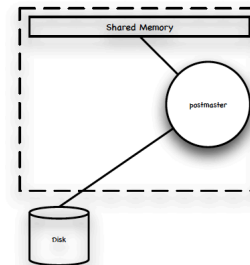
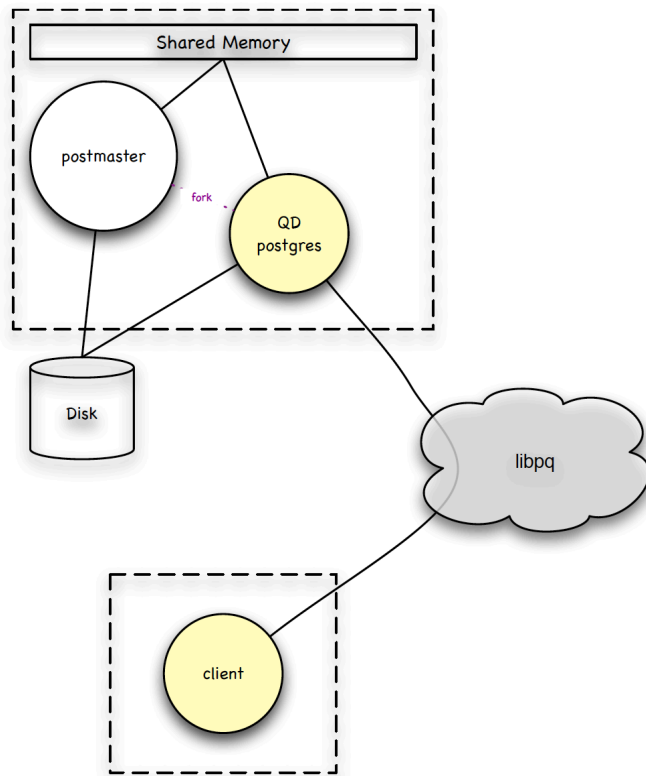
0. The system at rest.



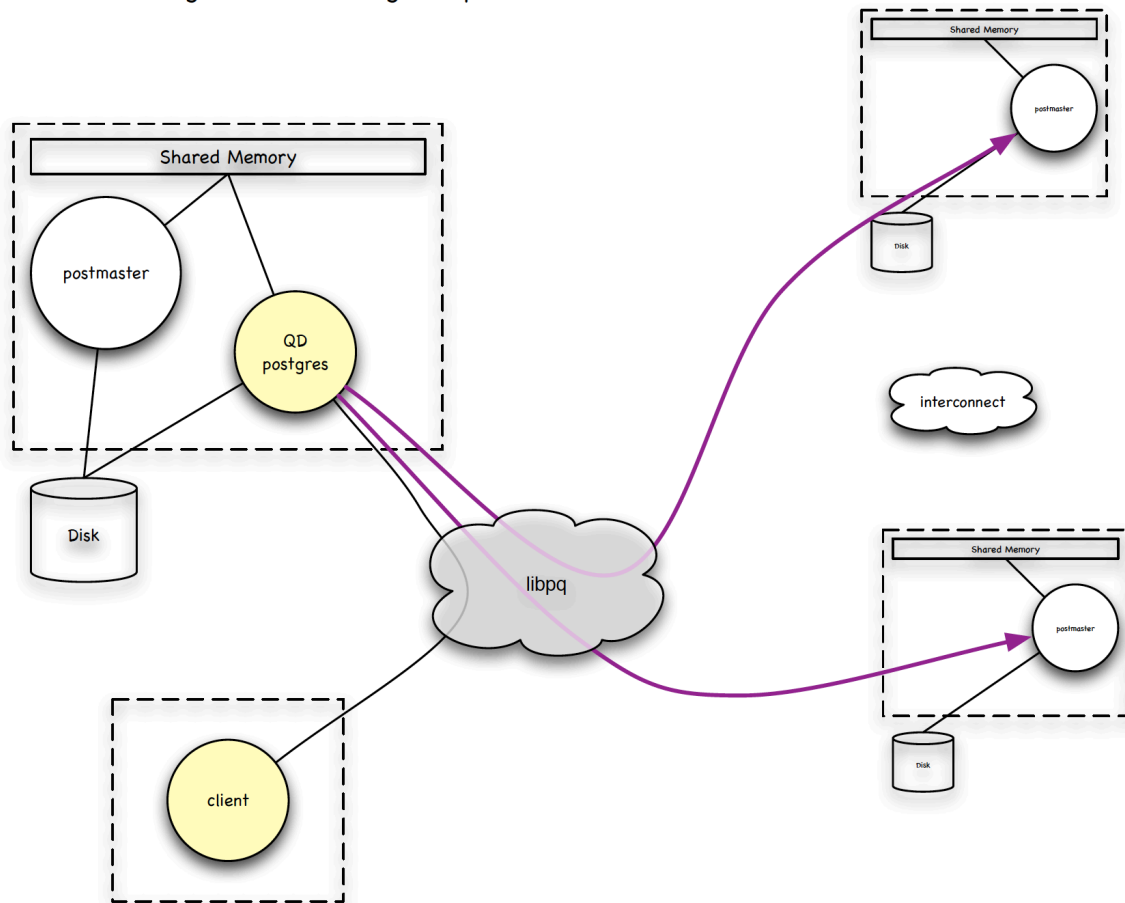
1. Client connects via the entry postmaster.



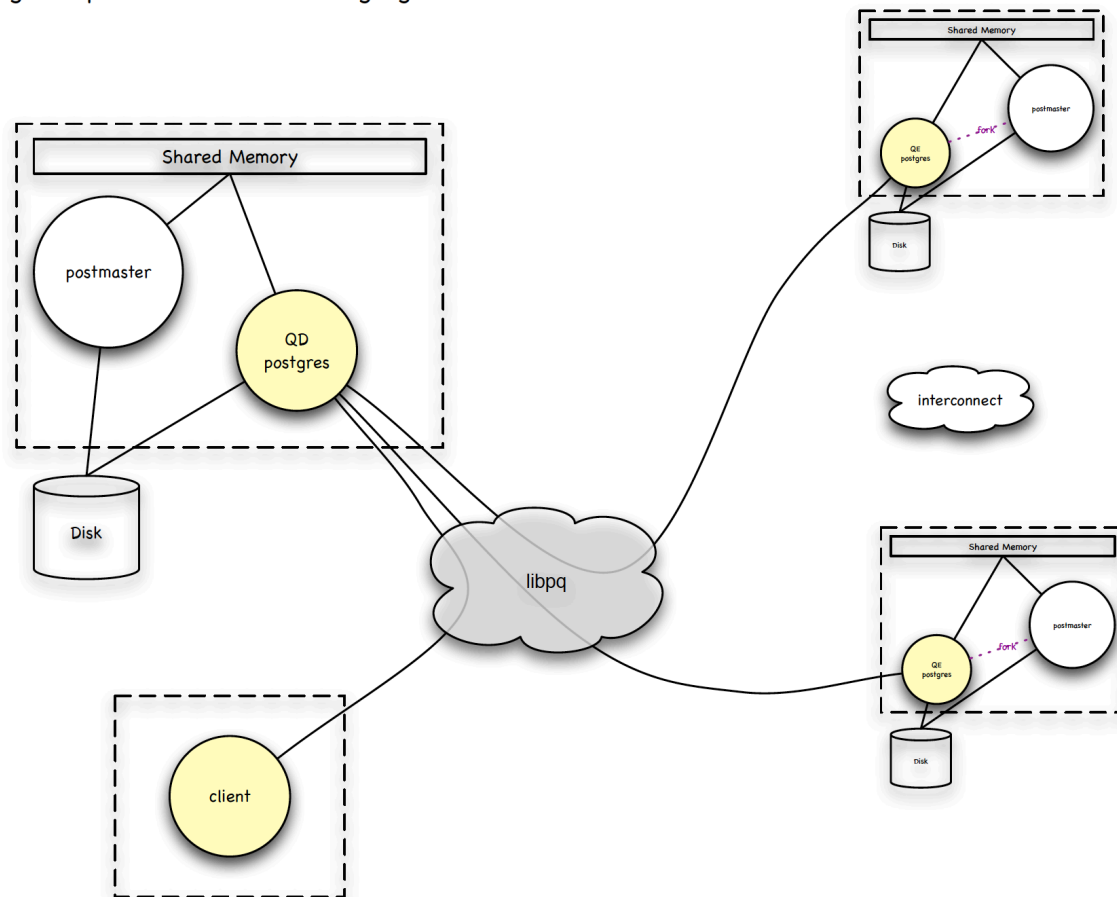
2. Entry postmaster forks a new backend -- the QD.



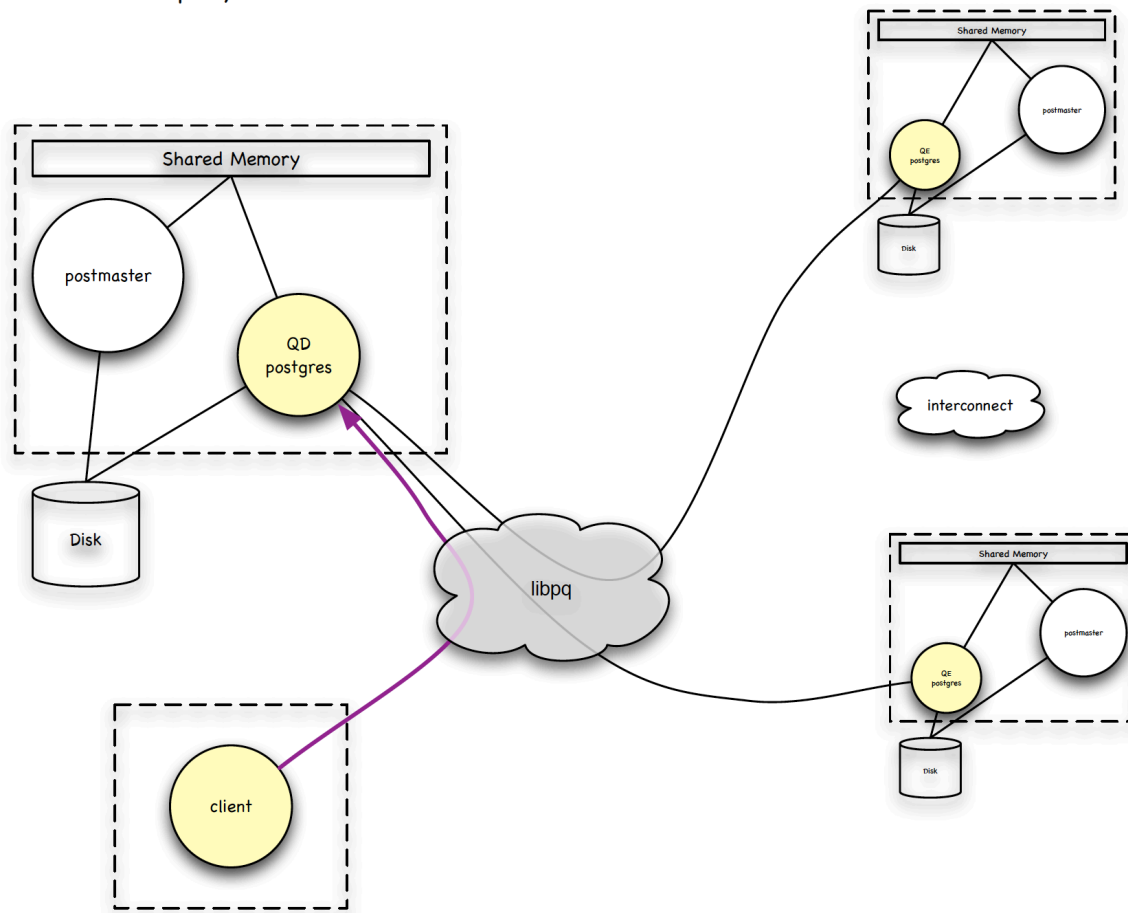
3. QD connects to segments via the segment postmasters.



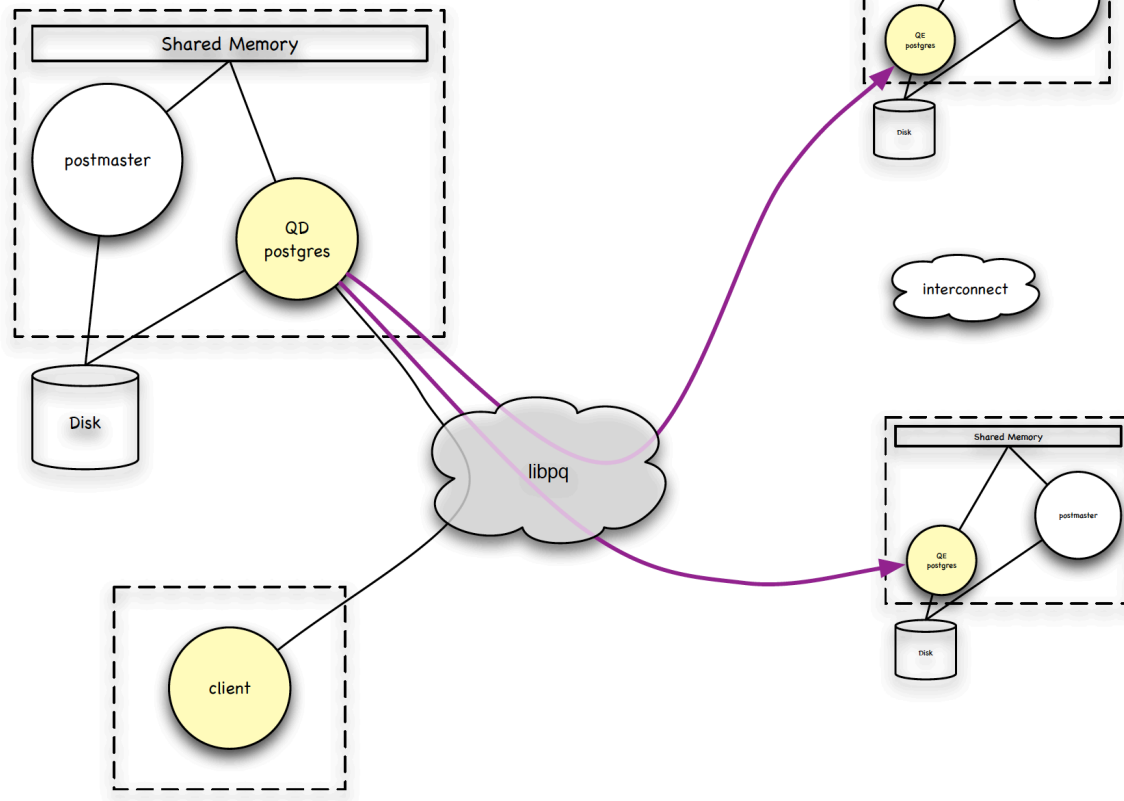
4. Segment postmasters fork initial gang of QEs.



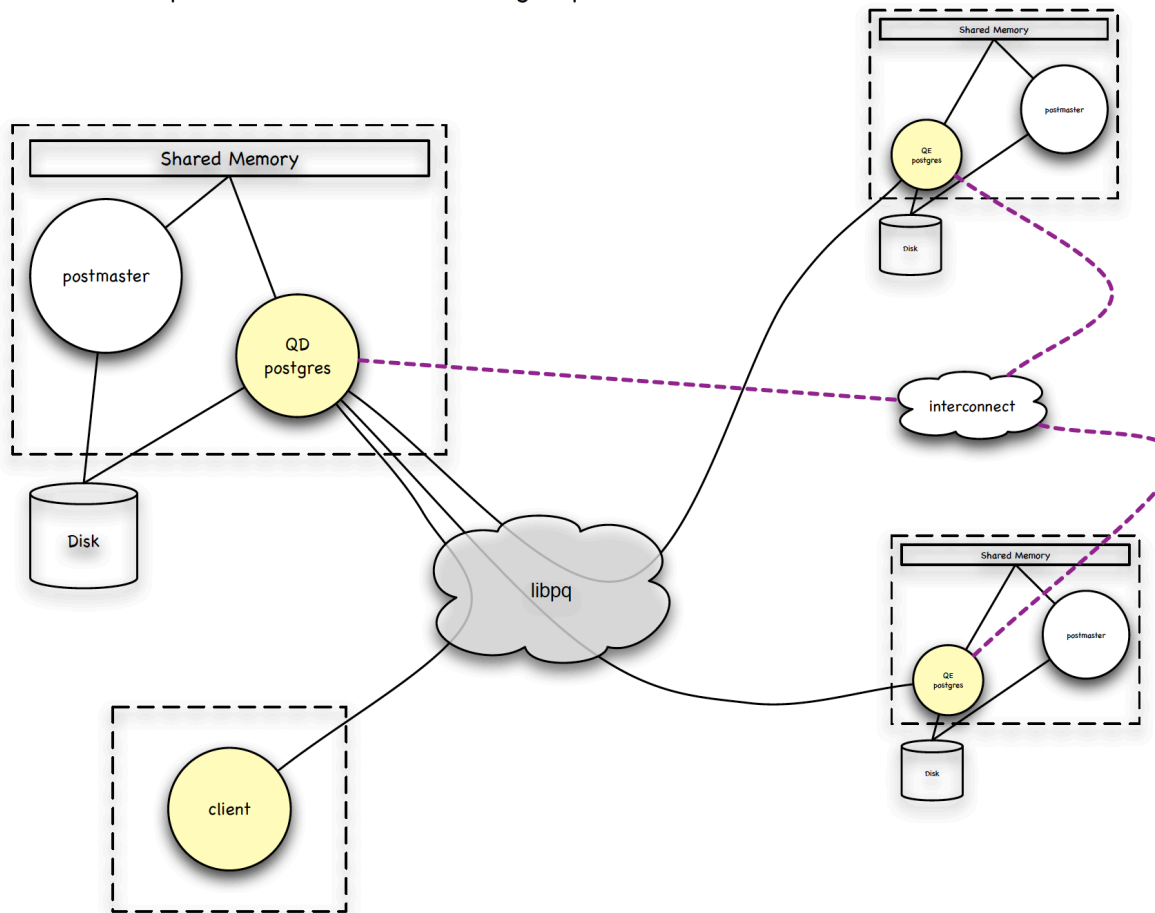
5. Client submits a query to the QD.



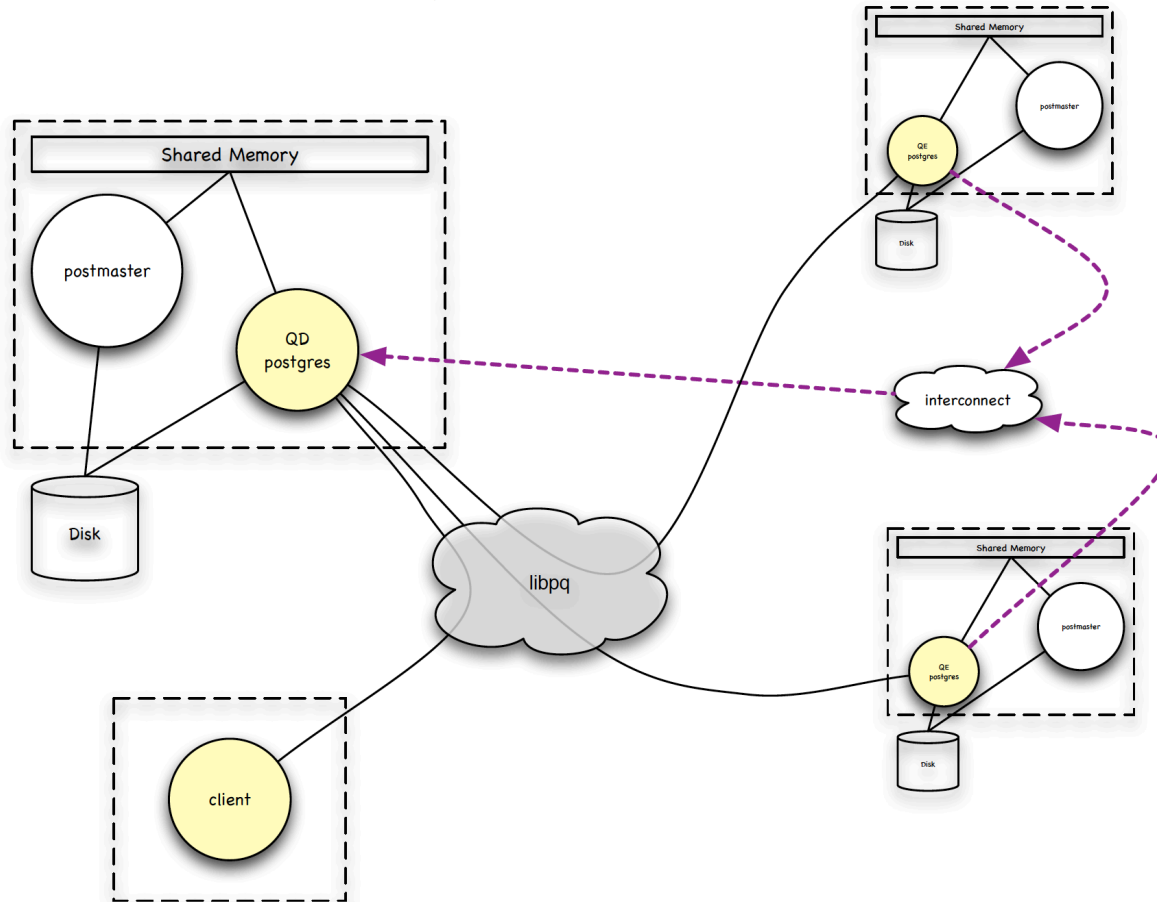
6. QD plans query and submits plans to QEs.



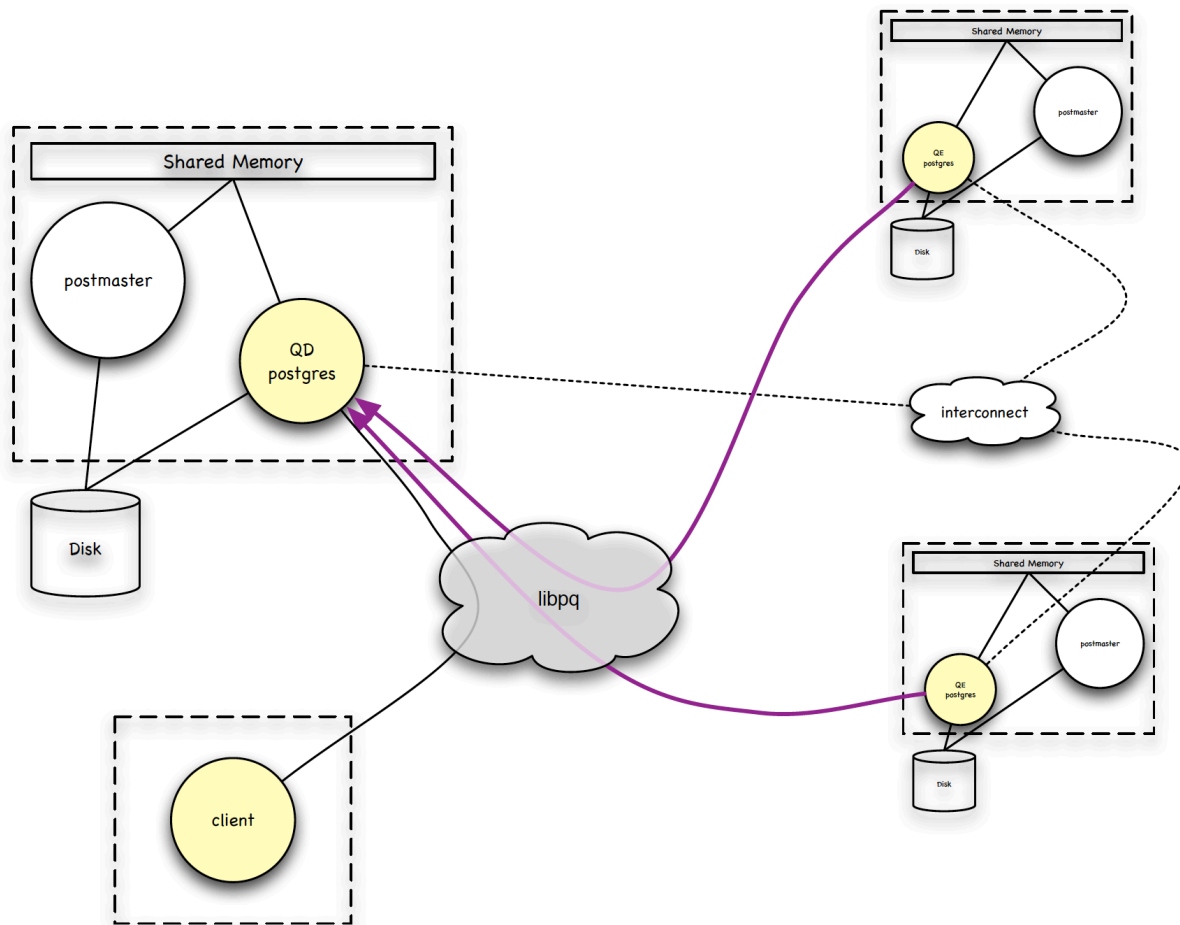
7. QD and QEs setup interconnect routes according to plan.



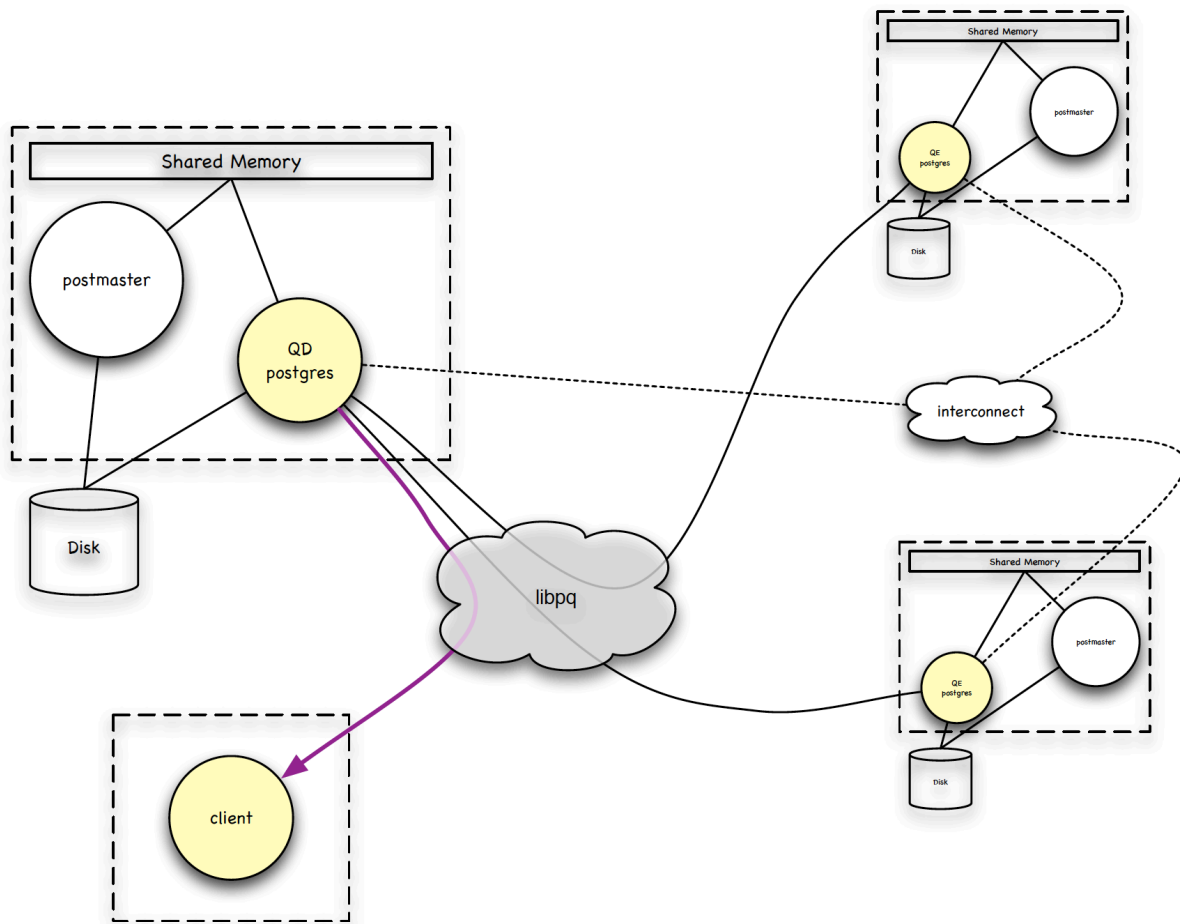
8. QD and QEs execute their slices sending tuples up the slice tree.



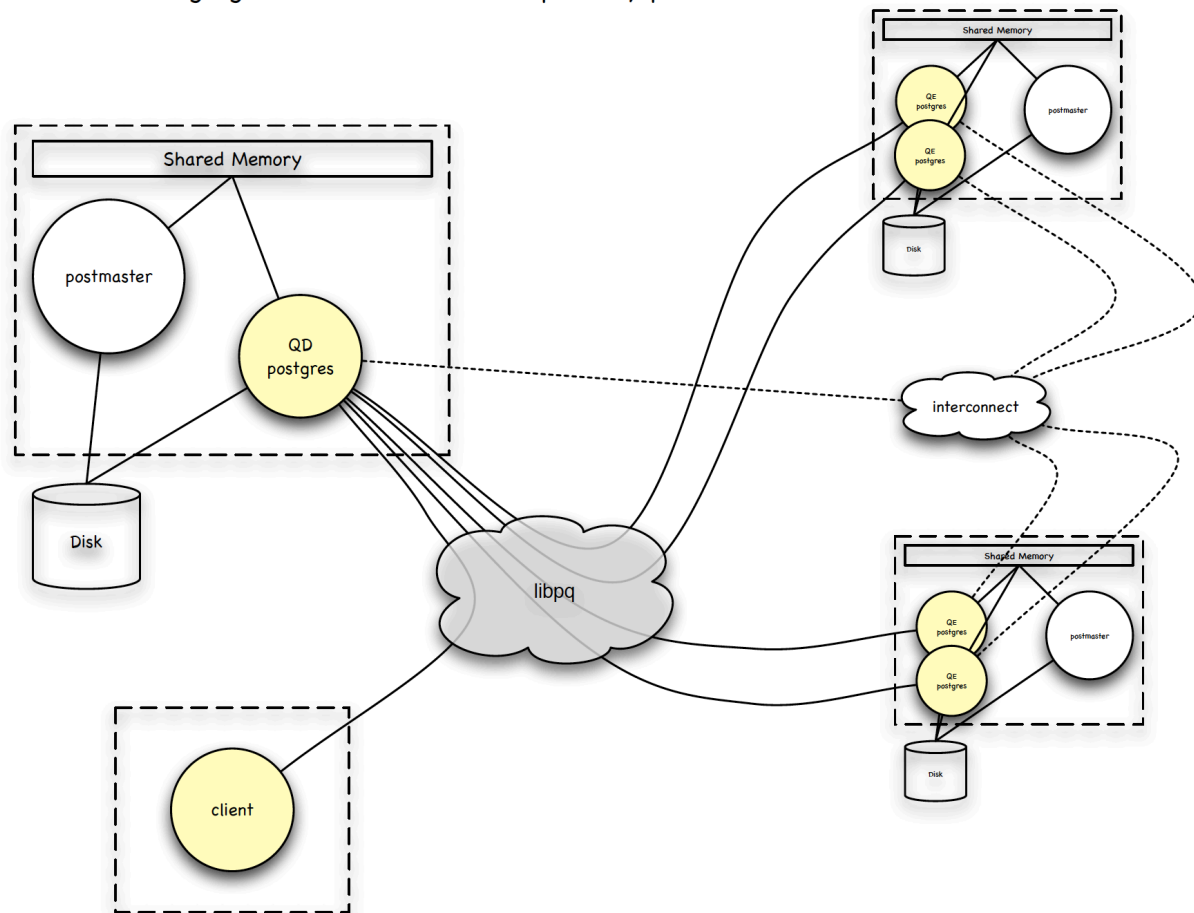
9. QEs return status to QD.



10. QD returns result set and status to the client.



Note that additional gangs of QEs are added as required by plans.



Greenplum 5.x

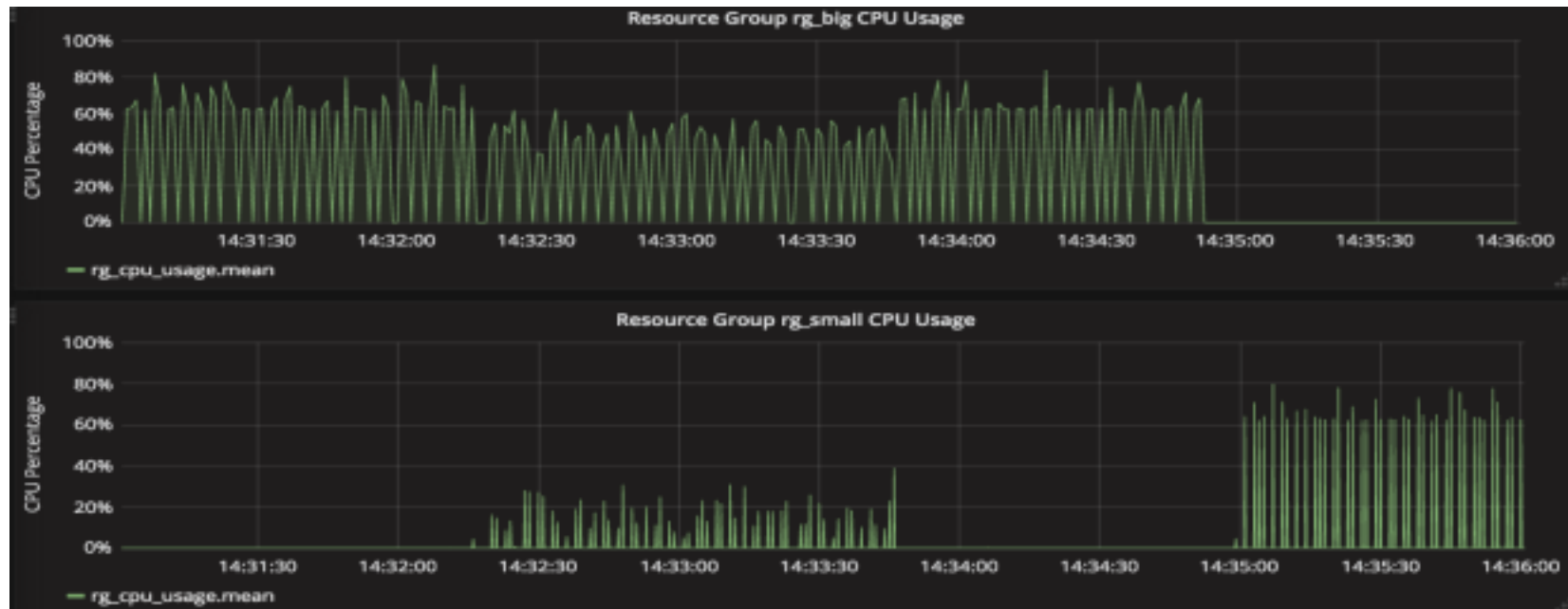
Greenplum 5.0 2017年9月已经发布！

- [PostgreSQL Core Features](#)
- [Python 2.7](#)
- [gpdbrestore Support for CASTs](#)
- [Enhanced Session State Monitoring](#)
- [Python Data Science Module Package](#)
- [R Data Science Library Package](#)
- [COPY Command ON SEGMENT Clause](#)
- ...

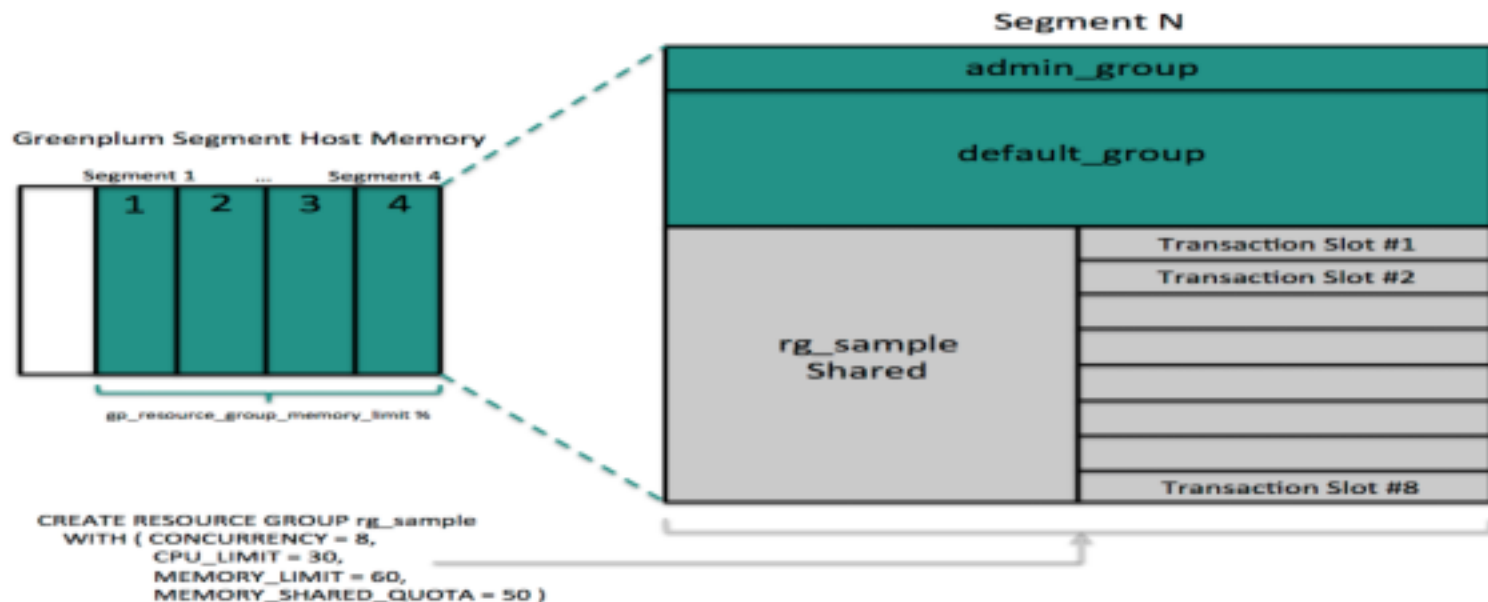
PostgreSQL 合并



CPU资源管理



内存资源管理



备份恢复和并行copy

- 重构gpbakcup/gprestore
 - 不再锁pg_class表
 - 普通表只加ACCESS SHARE锁
- 并行copy
 - COPY <table> TO <file> ON SEGMENT
 - COPY <table> FROM <file> ON SEGMENT

数据库内存储过程：TensorFlow

```
create function sfunc_train(state float[], a float)
returns float[] as
$$
    state.append(a)
    return state
$$ language plpythonu;

create aggregate agg_train(float)
(
    sfunc=sfunc_train,
    stype=float[],
    initcond='{}'
)
```

```
create function tfTrain(x_data float[], y_data float[])
returns numeric[] as
$$
    import tensorflow as tf
    import numpy as np

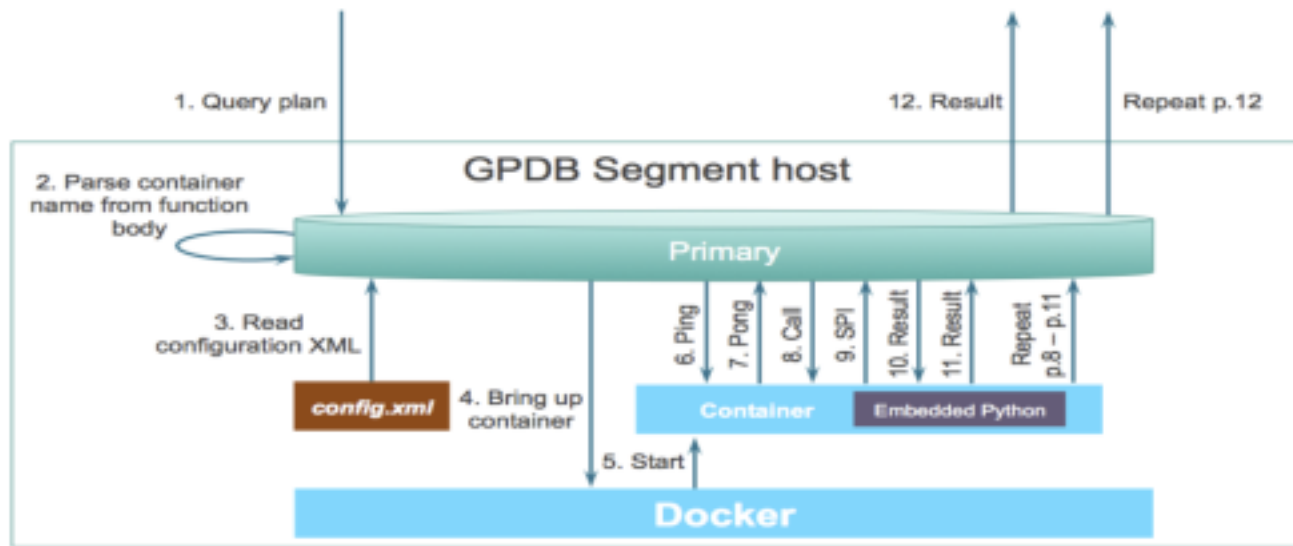
    W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
    b = tf.Variable(tf.zeros([1]))

    y = W * x_data + b

    loss = tf.reduce_mean(tf.square(y - y_data))
    optimizer = tf.train.GradientDescentOptimizer(0.5)
    train = optimizer.minimize(loss)

    init = tf.initialize_all_variables()
    sess = tf.Session()
    sess.run(init)
    for step in range(201):
        sess.run(train)
    return np.append(sess.run(W)[0], sess.run(b)[0])
$$ language plpythonu;
```

数据库内存储过程：PL/Container



Thank You