Greenplum资源管理器

姚珂男/Pivotal
kyao@pivotal.io

2017 年象行中国（杭州 站）第一期

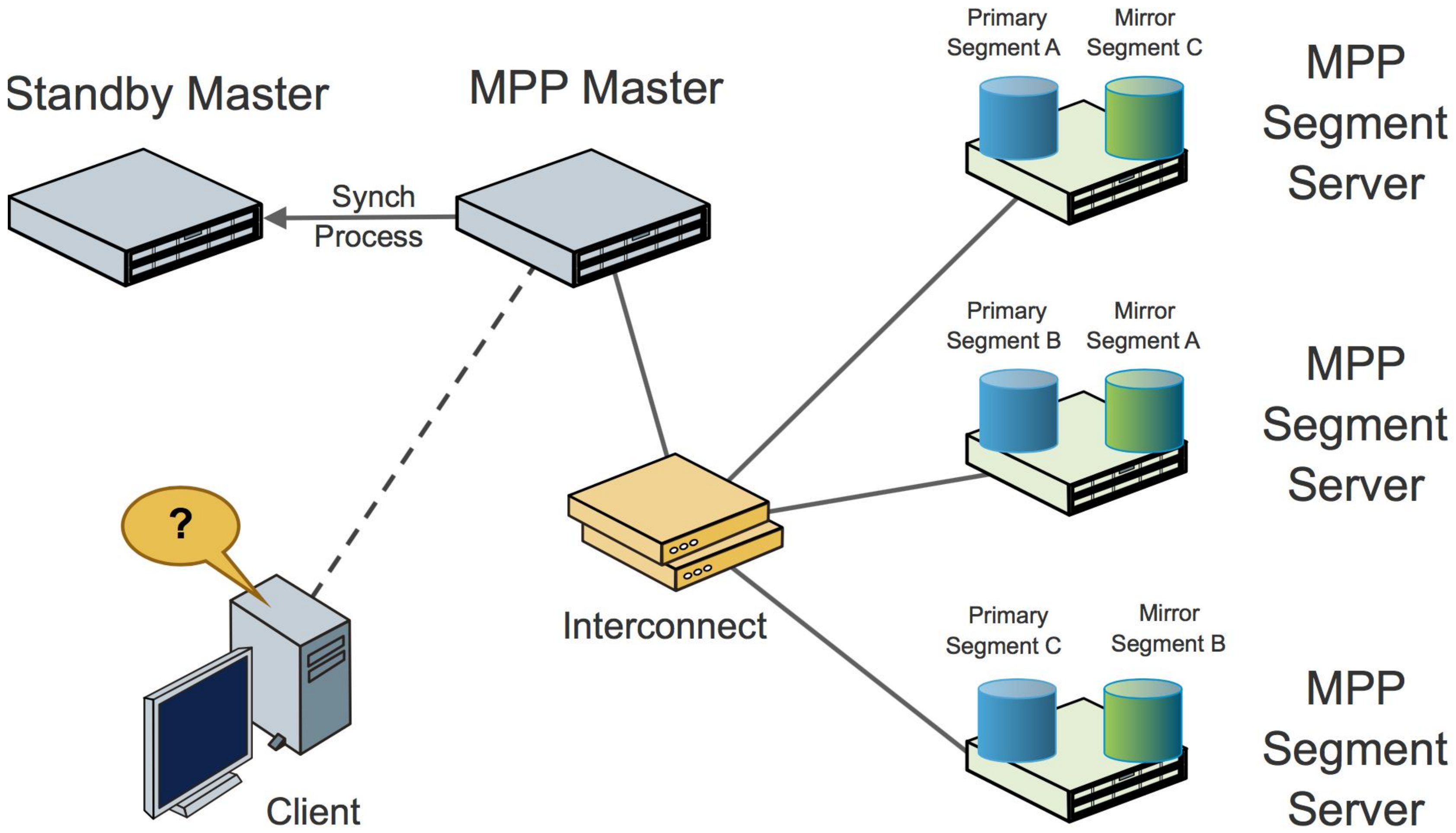
Agenda

- Greenplum数据库
- Resource Queue
- Resource Group

Greenplum数据库

- 基于PostgreSQL
- 分布式
- OLAP
- MPP(Massively Parallel Processing)

Greenplum数据库



Resource Queue

- SQL语句并发控制
- 基于cost的并发控制
- 基于priority的CPU控制
- 内存控制

Running Example

- CREATE RESOURCE QUEUE rq WITH

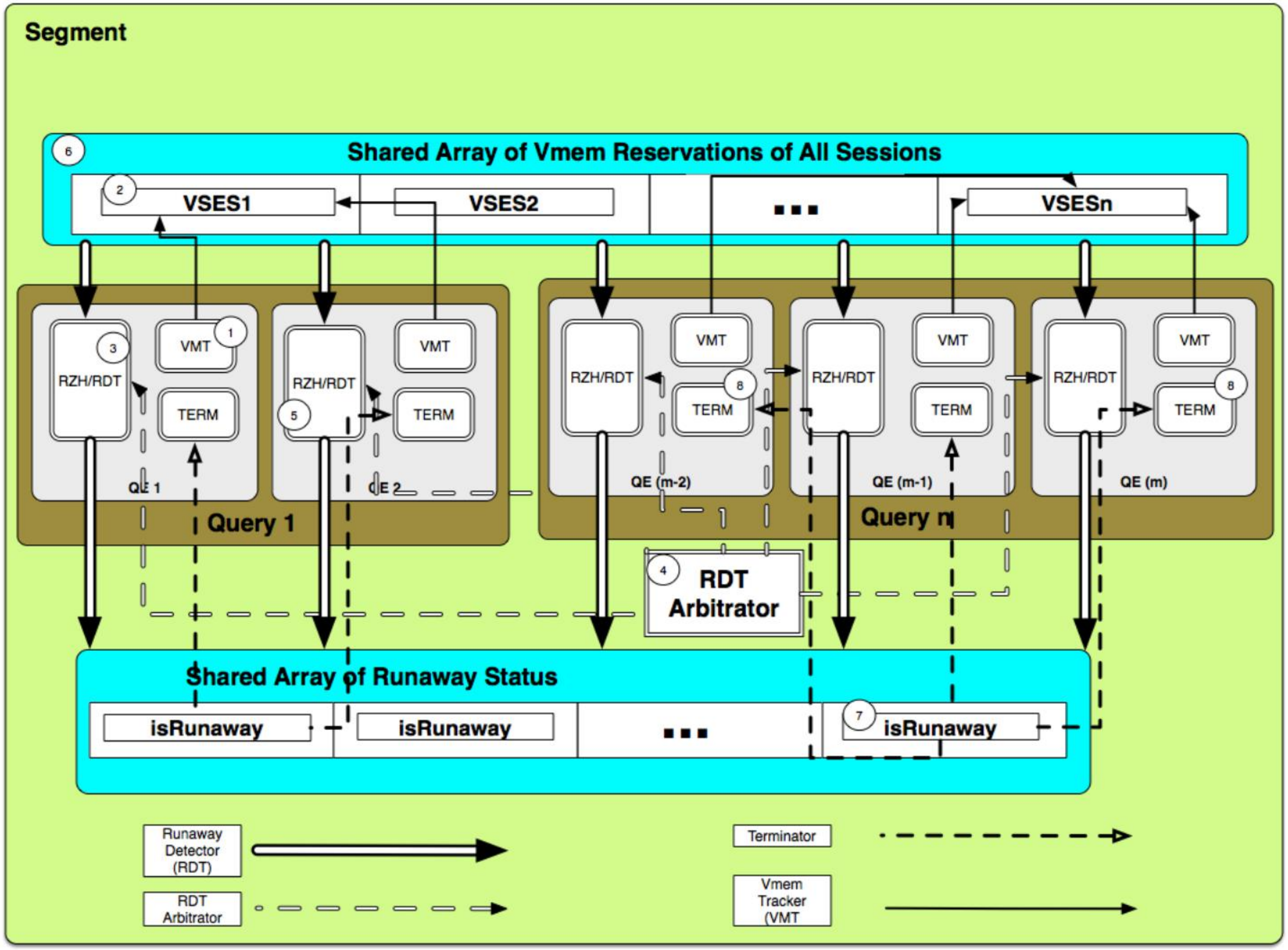
  ```
  (
       active_statements = 6,
       max_cost = 5e+06,
       cost_overcommit = true,
       min_cost = 50000,
       priority = high,
       memory_limit = ‘1024MB'
  );
  ```

- CREATE ROLE r1 RESOUCE QUEUE rq;
- SELECT * FROM gp_toolkit.gp_resqueue_status;

内存控制

- virtual memory note keeping (gp_malloc)
- statement_mem
- gp_resqueue_memory_policy
- work_mem & spill files
- gp_vmem_limit_per_query
- gp_vmem_protect_limit
- redzone & runaway detector

Redzone & Runaway Detector



Resource Queue

- Deadlock
  - active_statements => ‘等待’
  - SQL级并发控制 => 可能持有锁
  - 拿着锁等待 => 环状等待
  - Tx1: LOCK tbl; -- AccessExclusiveLock
    Tx2: INSERT INTO tbl; -- RowExclusiveLock, hang
    Tx1: SELECT * FROM tbl; -- hang

Resource Queue

- Self-deadlock
  - 每条SQL语句占用一个slot
  - extended query
    - prepare/bind/execute libpq protocol
    - cursor
    - named portal
  - SQL结束不一定释放slot
  - 一个事务用光所有slot

Resource Queue

- System PANIC
  - 需要睡眠/唤醒机制
  - Count + LWLock + Lock
    - Count：记录并发数
    - LWLock：保护count
    - Lock：睡眠/唤醒，死锁检测，状态报告
  - 维护Lock在共享内存的状态
  - bug => lock table corruption => PANIC

Resource Queue

- Cost is tricky
  - 没有明确的定义
  - 不同优化器不一致
  - 优化器不能纳入资源管理器

Resource Queue

- Priority is rough
  - 不能精确控制CPU
  - CHECK_FOR_INTERRUPTS
  - BackoffBackendTick
  - sweeper process (backoff.c)

Resource Queue

- Memory
  - Chaotic
  - 没有严格资源隔离
  - 第三方库的malloc

Resource Group

- SQL语句并发控制 => 事务并发控制
- ~~基于cost的并发控制~~
- 基于优先级的CPU控制 => 精确CPU比例
- 内存控制 => 严格资源隔离

Running Example

-
  ```
  CREATE RESOURCE Group rg WITH
   (
          concurrency=1,
          cpu_rate_limit=.5,
          memory_limit=.6,
          memory_redzone_limit=.7
   );
  ```

-
  ```
  CREATE ROLE r1 RESOUCE GROUP rg;
  ```

-
  ```
  SELECT * FROM gp_toolkit.gp_resgroup_status;
  ```

Resource Group

- Deadlock
  - 事务开始时拿slot
- Self-deadlock
  - 事务结束时一定会释放slot
- PANIC
  - Count + LWLock + Latch
  - Decouple with lock manager

Resource Group

- CGroups控制CPU
  - 目录：cpu/gpdb/rg1/, cpu/gpdb/rg2/ …
  - 设置cpu/gpdb/cpu.cfs_quota_us
  - cpu/gpdb/cpu.shares足够大
  - rg1和rg2的cpu.shares按比例配置
  - 空闲group配额会被抢占
  - 精确控制

Resource Group

- Memory
  - Not using CGroups
  - 重构resource queue内存管理
  - 严格资源隔离
  - statement_mem控制spill
  - 每个group内做redzone和runaway detection

Resource Group

- What’s more?
  - ALTER RESOURCE GROUP
    - 延迟生效
    - merge proposed value to real value
  - 动态迁移事务到其他group
    - 一致性
    - 死锁
  - Disk IO control?
    - buffered write?
  - Network IO control?
  - ...

Thanks!

Q & A