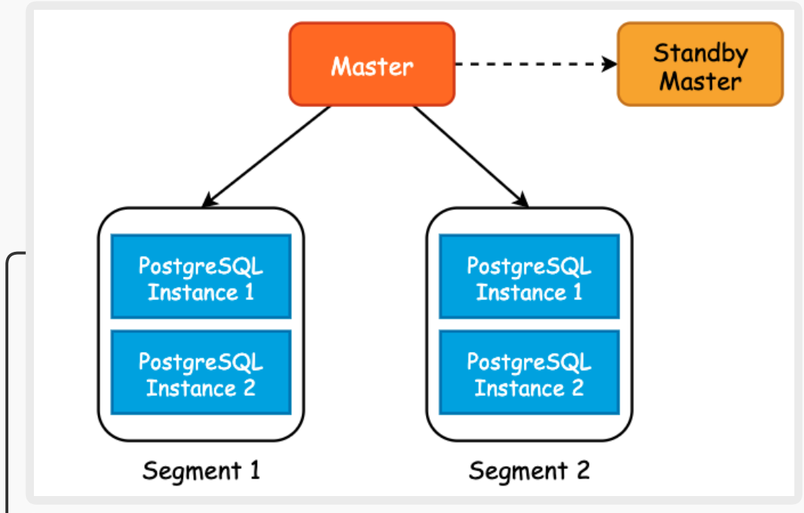
Greenplum 架构概览

基本拓扑结构



如上图，我们可以认为 Greenplum（后简称 GP） 就是很多个 PostgreSQL 实例所组成的集群。GP 对外提供统一的数据接口，并帮助用户自动完成数据分片、并行查询与聚合等诸多分布式数据库功能

GP 是一种典型的 Master-Segment 架构，一个 GP 集群通常由一个 Master 节点、一个 Standby Master 节点以及多个 Segment 节点所组成

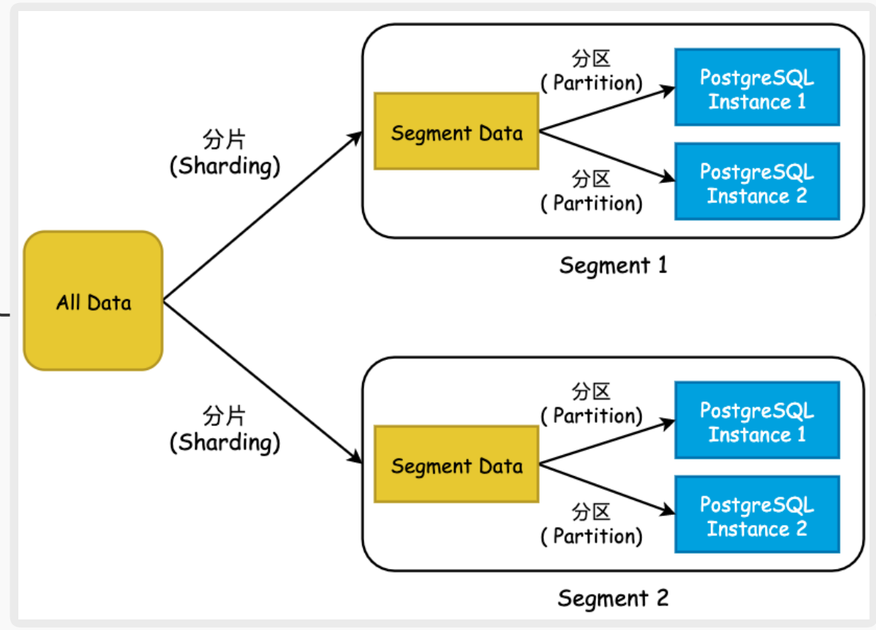
Master 节点通常不存储数据，只作为数据库的入口对 Segment 进行管理；Standby Master 节点则为 Master 提供高可用支持；而 Segment 节点就是真正的工作节点，数据存储在此处，并且一个 Segment 节点上通常会有多个 PostgreSQL 实例

Master-Segment 和 Master-Slave 有何区别?

- 在 Master-Slave 模型下，Master 和 Slave 拥有相同的数据，并且 Master 是存储和处理数据的唯一入口，Slave 仅复制 Master 的数据。比如 MySQL 的主从模型、Redis 的主从模型
- 在 Master-Segment 模型下，首先 Master 节点不存储数据，其次就是数据将会以分片的方式存储在多个 Segment 节点中。这里可以类比 Redis Cluster，只不过 Redis Cluster 是去中心化的。在 Master-Segment 模型中通常也会包含 Master-Slave 模型，也就是增加数据副本，以实现高可用
- **简单地来说，Master-Slave 主要进行数据复制（冗余），而 Master-Segment 则会同时进行数据分区（水平扩展）和复制（冗余）**

分区与分片

在项目初期，我们使用一张表 T 存储数据。随着业务的增多，单表出现性能瓶颈，因而将 T 水平拆分成多个表进行存储，这个过程通常称为分区。紧接着，单一的数据库实例出现瓶颈，因此需要使用多个节点创建多个数据库实例，再按照某种规则将数据尽可能均匀地分布到各个节点上 ，这个过程通常称之为分片

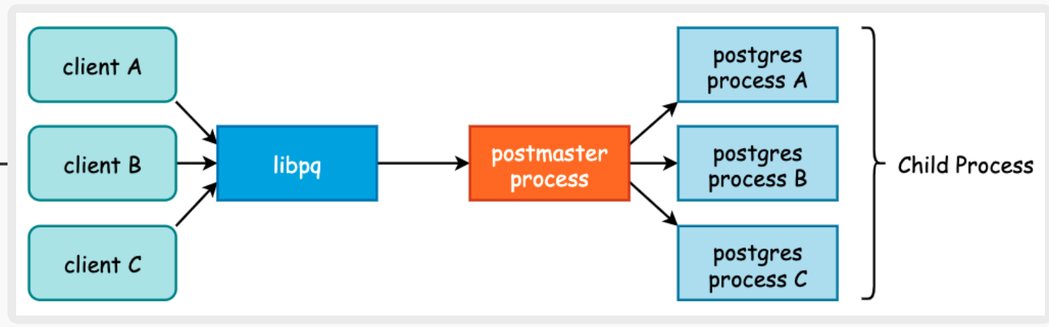


GP 同时支持数据的分片和分区，具体的分片和分区规则将会在后面的总结中详述

同时，GP 在存储上支持多态存储，也就是对于同一份数据，既可以选择基于行的存储方式，也可以选择基于列的存储方式，并且支持诸如 S3、HDFS 等外部存储

PostgreSQL 进程模型

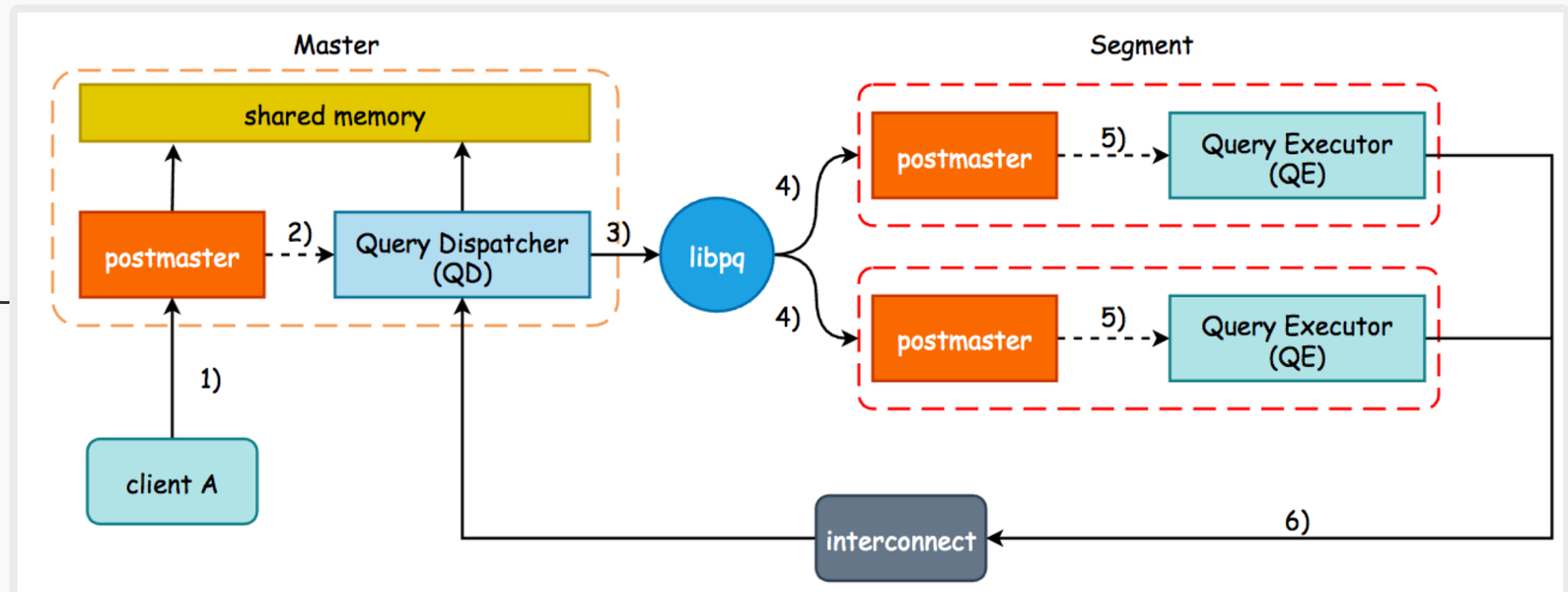
PostgreSQL（以下简称 PG）采用的是经典的 C/S 模型，即 Client-Server 模型，同时使用多进程的方式支持并发查询与写入。也就是说，每当有一个客户端连接至 PG 时，就会有一个子进程被创建出来。postmaster 进程和 postgres 进程之间采用共享内存进行通信



client 和 PG 之间的通信过程

1. client 调用 libpq 库向 PG 的 Postmaster 进程发起连接请求
2. PG fork 出一个 postgres 进程与该客户端建立连接，postmaster 进程不再处理与该客户端的相关请求
3. postgres 进程接收客户端的请求，处理并返回结果。当然，响应需要经过 libpq 库的处理

GP 基本查询流程



基本查询流程

1. Query Dispatcher
   - **当 client 向 Master 发起查询请求时，Master 节点上的 postmaster 进程将会 fork 出一个子进程，叫做 Query Dispatcher（分发器），简称为 QD 进程**
   - QD 进程会对收到的查询请求进行处理，包括解析原始查询语句、优化器优化以及生成分布式查询计划，然后将查询计划通过 libpq 库发送给其它的 Segment 节点
2. Query Executor
   - **Segment 节点上同样是 PG 进程，所以仍然由 postmaster 进程负责监听端口，由 Query Executor（QE 进程）进程处理相关查询**
   - QD 进程将查询计划发送给 QE 进程并执行，同时 QD 与 QE 进程之间、QE 与 QE 进程之间使用 interconnect 进行通信，而非 libpq

最终，QD 进程将 QEs 中得到的结果进行汇总，并通过 libpq 返回给客户端