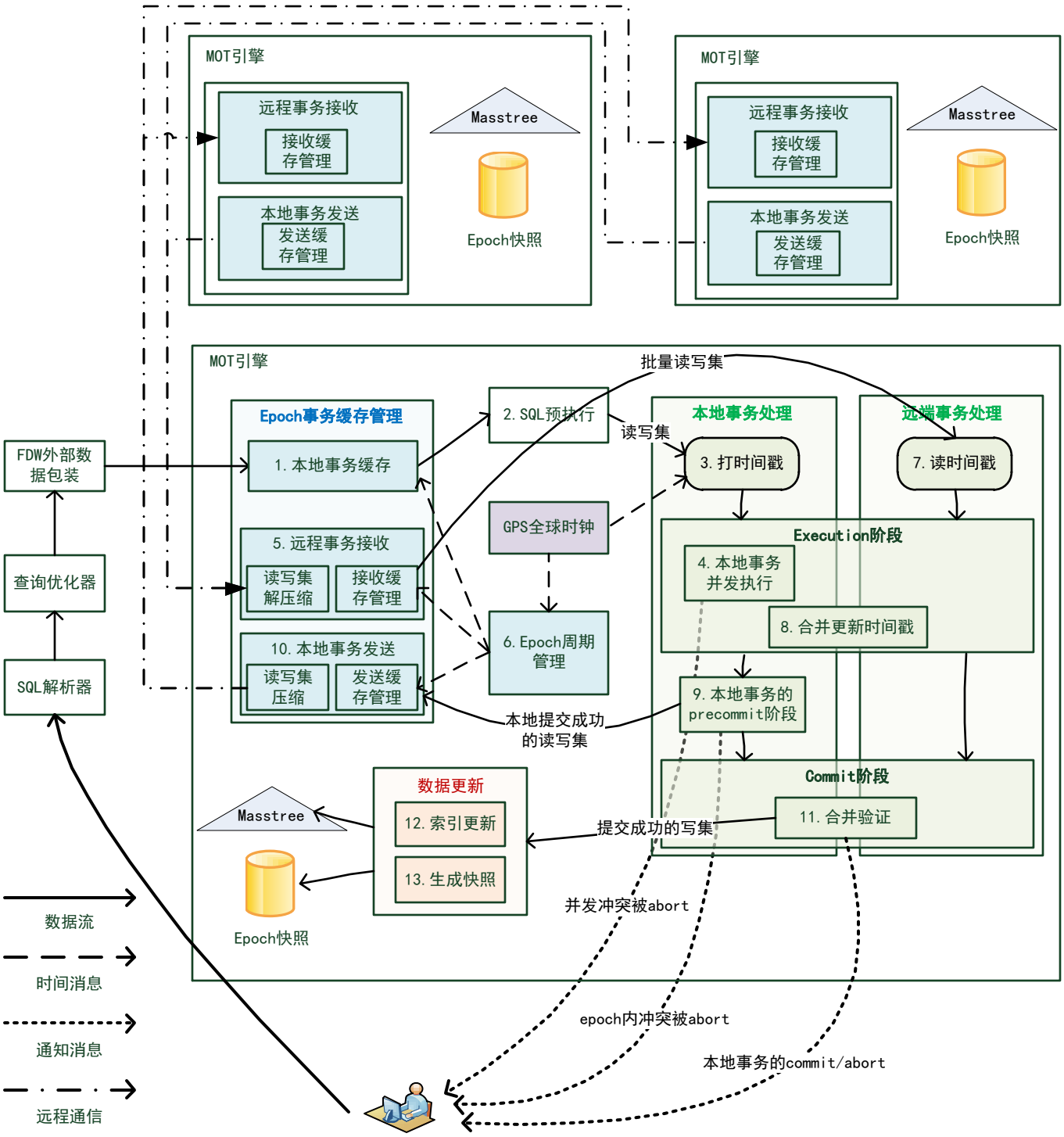

基于时间戳的Coordination-Free分布式事务处理系统：

高可用：基于复制数据库技术和Geo-replicated架构，提供跨区域高可用性质。每个副本为本区域用户服务，因为本区域用户大多对相关的部分数据进行读写操作，所以写写冲突大多发生在区域内，可以立即返回结果。而部分写操作可能会在和远端写操作合并后返回结果。

一致性：结合Epoch快照隔离技术和无协调副本合并技术，利用华为提供的GPS全球时钟模块实现。Epoch周期内都读取上一个周期生成的保证一致性的快照，保证读一致性。写写冲突通过Coordination-Free的多副本合并技术解决。

隔离性：支持读已提交（总是读最新的epoch快照）、可重复读（读事务开始时的epoch快照）

高性能：基于openGauss MOT引擎实现，复用高性能并发控制实现，提高事务处理吞吐率和降低延迟（epoch控制，tail latency保证）。

系统包括三大模块，即Epoch事务缓存管理（蓝色）、事务处理（绿色）、数据更新（红色）。具体工作流程如下：

0. 区域本地内用户提交事务处理请求，经过openGauss的SQL解析器、优化器、FDW等组件进入MOT引擎
1. 本地事务缓存（多线程并行执行），在epoch切换的时候，由于要等待新快照的生成（读操作要基于新快照），所以会涉及一个缓存。如果不是在epoch切换时候，那就可以来一条处理一条，就是不缓存。
2. SQL预执行生成读写集（多线程并行执行）
3. 通过请求GPS全球时钟组件，获取全局时间戳，并给该事务的starttime和committime时间戳（多线程并行执行）
4. 本地多线程并行处理用户提交的事务，在本地内存表上根据“后写胜利”规则（即max单调操作），执行写操作覆盖，并更新写操作数据项的最大时间戳，所有读操作读取上一个epoch快照数据。并发执行时候，发现已有更大时间戳的事务，被abort，发送abort消息返回给本地用户
5. 在0-4执行过程中，远程事务接收线程一直接收远程事务读写集，并缓存起来形成批量读写集，这里设置多个接收线程，每个远端peer节点对应一个接收线程
6. epoch周期管理根据全局时钟，在约定好的epoch结束时候，确认收到全部peer节点的读写集之后，开始处理远程事务读写集
7. 不同于本地事务要自己打时间戳，远程读写集的时间戳已经打好，这里只需读取
8. 并行地合并更新时间戳，因为之前本地事务会更新一部分数据的写时间戳，这里根据远程读写集还会更新一批（多线程并行执行）
9. 被本地的一个epoch内的后续写操作覆盖的本地写事务会在precommit阶段确定被abort，并通知用户
10. 在epoch end时候，接受完该epoch内所有的本地事务之后，确定了本地事务可以提交成功的事务读写集（注意，与远程读写集合并后仍然可能会有被abort的本地事务，这里只检查本地事务的冲突，然后尽快发送出去），交给本地事务发送模块，发送给其他所有peer节点
11. 本地事务读写集与远程事务读写集进行合并OCC验证（多线程并行执行）。通过对比事务读写集时间戳和数据最新的时间戳对比，确定是否被覆盖，确定最终提交成功和被abort的事务，然后每个节点只负责通知区域本地用户提交成功还是失败
12. 提交成功的写集发送给数据更新模块，进行masstree索引更新，这里涉及内存资源分配管理
13. 另外根据最新的数据生成该epoch快照，下一个epoch的读操作都基于该快照