创新点：

PALF 是作为 OceanBase 的先写日志复制系统而提出的。它具有高可用性、出色的性能和类似文件的接口，适用于分布式数据库。

我们将特定于数据库的需求抽象为 PALF 基元，如显式复制结果和变更序列号，这对 OceanBase 数据库大有裨益

我们提出了一种新方法，可将日志从一个 Paxos 组同步到其他组，从而为物理备用数据库等功能提供动力

我们在第 6 节中介绍了构建高性能共识协议的设计，并在第 7 节中讨论了 PALF 的设计考虑因素。在闭环客户端和数据库工作负载下进行的评估均显示出卓越的性能

面对挑战：

Leader Election：日志系统leader的选举传统上取决于共识协议，而不考虑数据库的要求。

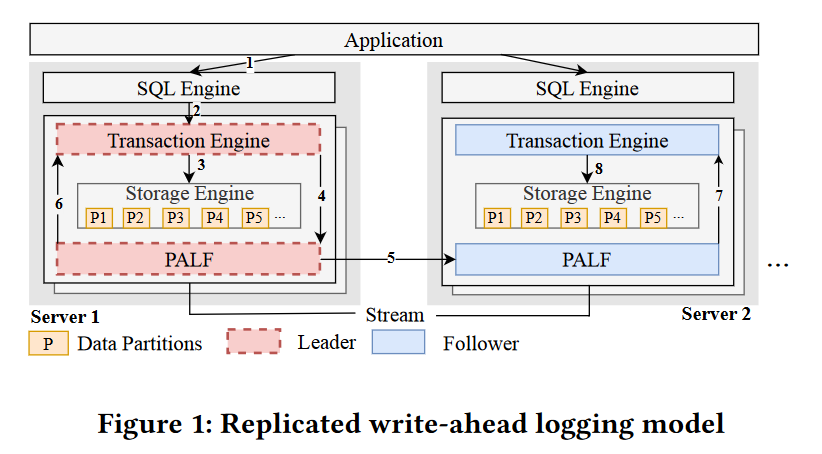
Uncertain Replication Results：事务是否提交取决于日志是否持久化，如果日志系统因异常而向事务模型返回了错误结果，数据就会变得不一致

Data Change Synchronization：分布式数据库将日志从主数据库中的一个复制组同步到备用数据库中的下游组方面存在挑战，而且这些复制组应该是独立可用的。

Performance：单个复制组的吞吐量是有限的，当通过增加复制组来提高吞吐量时，由于复制组和数据分区是对应的，数据分区的数量也会增加，导致数据分区变小，出现更多的分布式事务，降低性能。

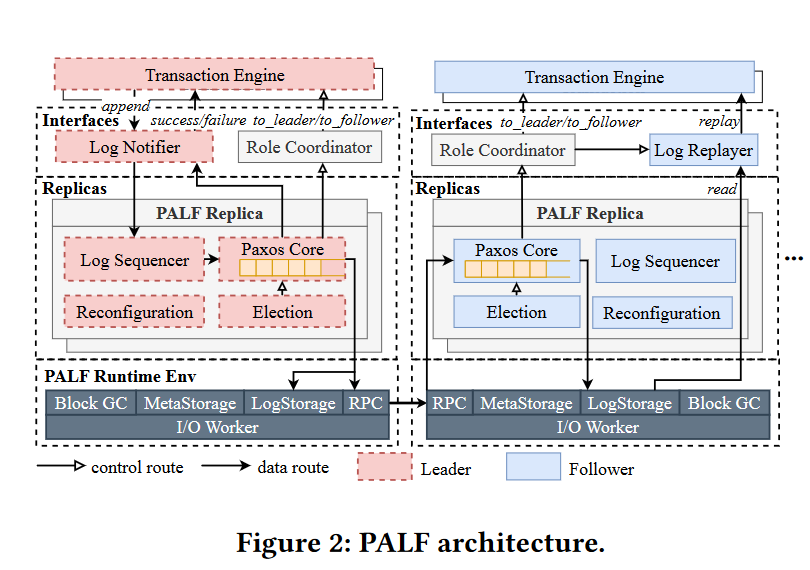
1. PALA设计

1.1复制WAL模型



应用执行事务时，直接修改内存存储引擎中的数据（2-3）。然后生成日志并追加到palf中（4）。Leader将palf视为本地日志文件系统，只关心日志是否已被刷新。Palf将leader中执行的修改复制到fallower（5），如果日志被palf提交会通知事务引擎结果(6)，并在fallower重放leader所做的修改（7-8）。

1.2 PALF架构



PALF由三层组成：

interface layer（接口层）：一组与数据相关的应用程序接口，省略了用于系统管理的接口。隔离了数据库特性对 PALF 的影响，提高了 PALF 的通用性。Leader中的log notifier会通知事务引擎日志是否提交。follower中的log replayer向事务引擎重放日志中记录的变化。role coordinator负责在role发生变化时接收角色转换信号并转发信号来转换事务引擎的角色。

PALF replicas layer（PALF 复制层）：log sequencer负责将为每条记录分配一个单调递增的日志序列号 (LSN)，该序列号唯一标识 PALF 组内的日志条目。记录将封装为日志条目，并通过 Paxos 协议按照 LSN 的顺序复制到fallower并由其持久化。当大多数 PALF 复制已持久化日志条目时，提交该日志条目。并且PALF的选举是独立的。Reconfiguration模块负责管理 PALF 小组的成员资格。

PALF runtime environment（PALF 运行环境）：在该层，为 PALF 复制提供了RPC接口并管理磁盘资源。PALF 复制中的所有日志条目都以多个恒定大小的块的形式存储在 LogStorage 的一个唯一目录中。MetaStorage 存储元信息，如所有 PALF 复制的成员信息。BlockGC 负责在不再需要日志块时对其进行裁剪。PALF 复制发出的所有 I/O 请求都由 PALFEnv 中的统一 I/O Worker 池处理。

1.3 系统接口

文本, 信件

描述已自动生成

Palf提供了两种写日志的方法append和mirror

append：将记录r提交给palf组的leader，返回一个LSN标识的日志条目。该方法是异步的，事务引擎只需要保证日志条目被分配了唯一的LSN并提交到leader的缓冲区即可。事务引擎通过回调函数Appendcb来获悉日志条目是否提交，每个日志条目的回调函数最多被调用一次。

mirror：是为了palf组镜像设计的，只接受另一个palf组提交的日志条目。

读取接口read可根据lsn随机访问日志条目

locate接口能够实现将csn映射到lsn。

monitor\_tail 方法注册了一个回调函数TailCb来监控PALF 组的尾部。当提交新日志时，tail函数就会被调用，通知日志消费当前日志的尾部。

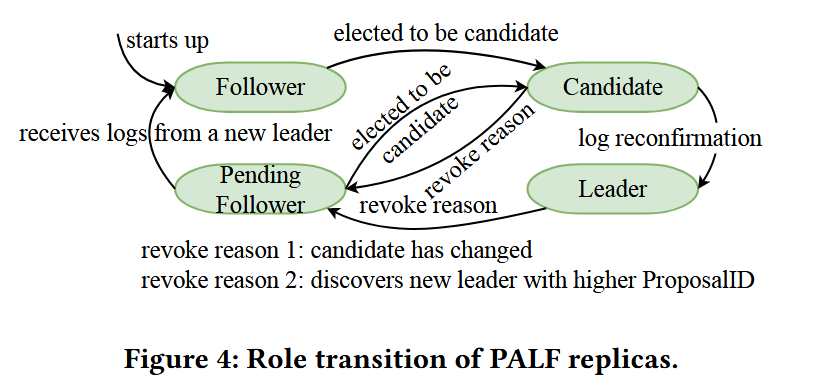
RoleCb 函数用于协调 PALF 复制和事务引擎的角色。当 PALF 复制的角色切换时，将调用 to\_follower (to\_leader) 方法。

trim 方法负责指出给定 LSN 之前的无用日志条目。BlockGC 负责回收这些日志。

* 1. 共识实现

**领导选举的要求**：PALF 将Leader选举与共识协议分离开来。用户可以灵活指定当选Leader的副本的优先级。如果前一个Leader崩溃，优先级第二高的副本将优先被选为新的Leader，而无需任何外部操作。如果前一个Leader从故障中恢复，且其优先级仍高于当前Leader的优先级，则领导权可自动转到已恢复的副本。

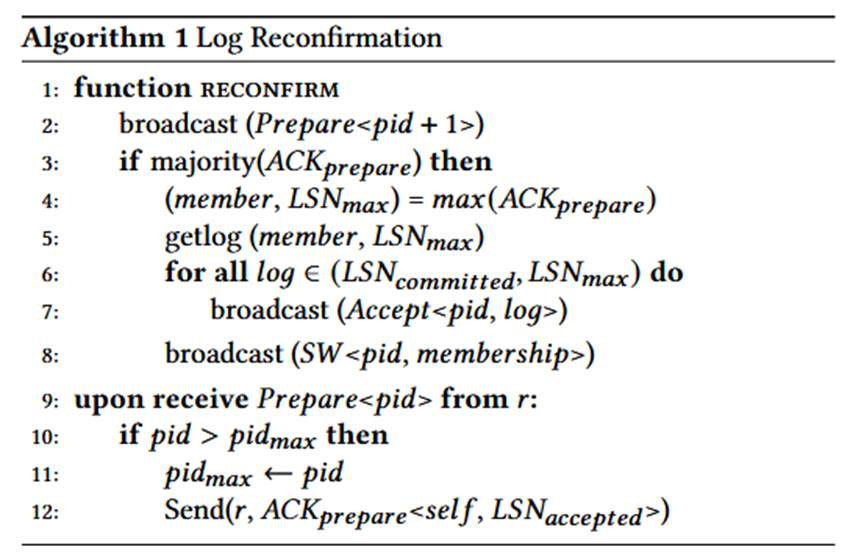
角色转变：



在任何时候，每个副本都处于四种角色之一：Leader、fallower、candidate, pending follower.。启动后首先默认为fallower，并且不定期的向选举算法询问自己是否是candidate，如果是则进行切换，然后通过log reconfirmation算法后成为leader。当leader发现candidate已经改变或者接受到新leader的信息，会转变为pending fallower，用来保证在领导权转移之前追加的日志结果的确定，当确定之后会变为fallower。

领导选举：paxos算法的变体，每个副本在提出新一轮选举之前，都会尝试向所有副本广播自己的优先级；副本将响应在一定时间内收到的所有请求中优先级最高的请求。此外，选举算法还在每个副本中使用单调计时，以保证在一定时间内，如果有任何多数副本存活，候选人就会当选。

log reconfirmation：



由于leader的选举根据优先级灵活选择，会出现leader的日志数目小于其他副本，因此需要重新确认来保证日志数目不少于其他副本。

具体来说，候选者会通过 Paxos 准备信息（第 2 行）向所有副本广播高级提案 ID（Leader任期的标识符）pid + 1。每个收到广播的副本会比较其见过的最大的pid，只有当大于时才会回应候选者，并且回执消息中只包含日志的lsn。

当候选者接受到prepare消息后会选择日志最长的副本，从该副本中获取日志，并将这些日志复制到所有副本。最后会向所有副本广播StartWorking 日志，用来回滚上一任leader可能未承诺的成员资格。只要该日志达到多数，就会成为leader。

日志复制：一旦Leader成功接管，它就会承担起复制和提交日志的责任。简而言之，日志条目按照 LSN 的顺序依次被添加到Leader、被Leader复制、被fallower确认以及被Leader提交。当记录添加到leader时，log sequence将为每个日志条目分配一个LSN。

2，与事务引擎的交互

2.1 显示复制结果

当leader转移时，前任leader可能不清楚追加日志是否提交成功，这些日志被称为pending logs。会增加事务引擎的负担。PALF 保证，除非Leader崩溃或网络永久中断，否则事务引擎将明确收到复制结果的通知。当leader转移时，leader会变为pending follower，当其接收到leader的消息时，pending logs的结果就会变得明确，已提交日志的复制结果将通过调用 success 函数通知，反之通过调用 failure 函数通知复制失败。这就是为什么先前的Leader必须切换为待定fallower，并在成为fallower之前等待新Leader的日志。

2.2 更改序列号（csn）

3，数据更改同步