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。

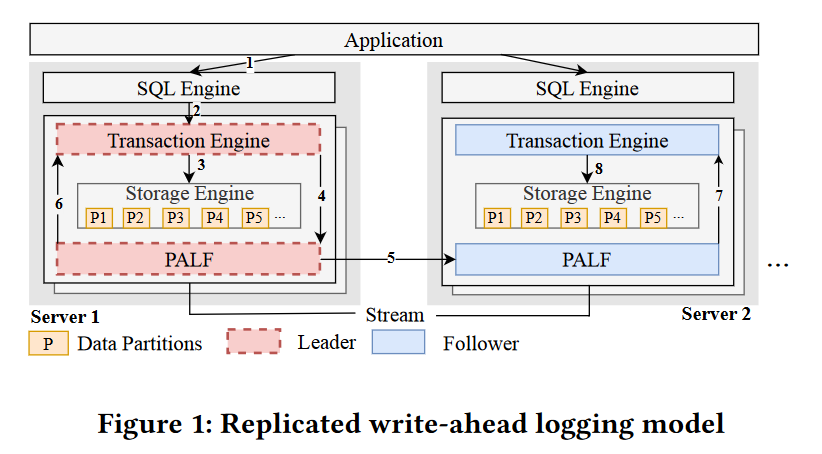
Uncertain Replication Results：事务是否提交取决于日志是否持久化，如果日志系统因异常而向事务模型返回了错误结果，数据就会变得不一致

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。

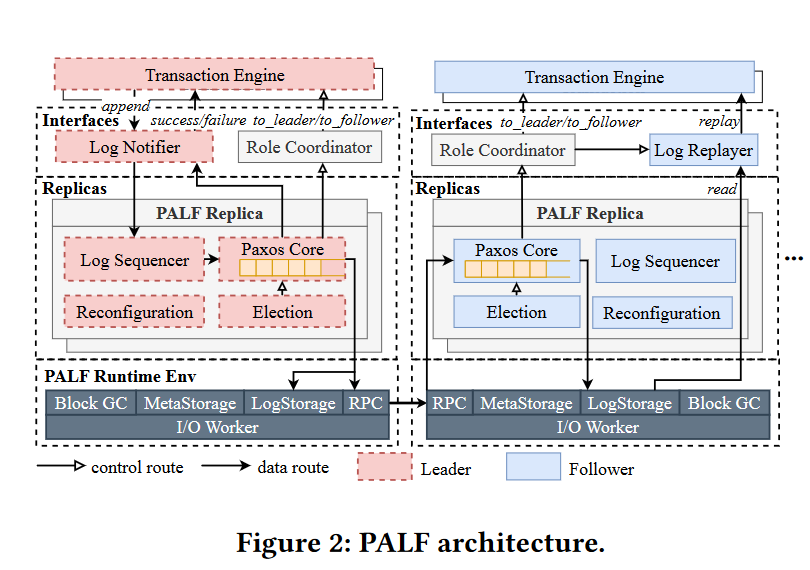
1. PALA设计

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。

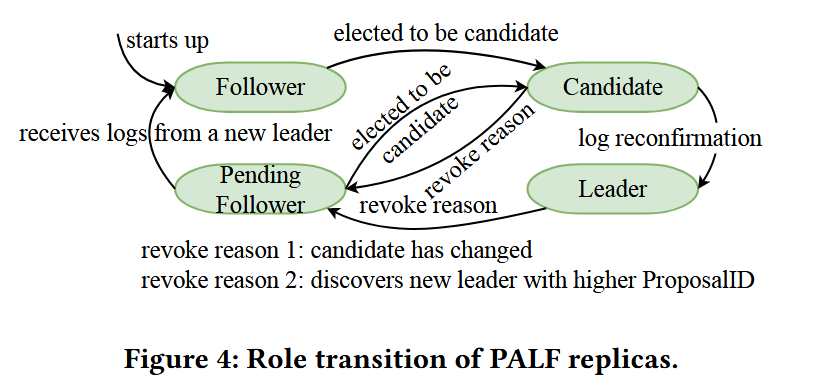
RoleCb 函数用于协调 PALF 复制和事务引擎的角色。当 PALF 复制的角色切换时，将调用 to\_follower (to\_leader) 方法。

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。

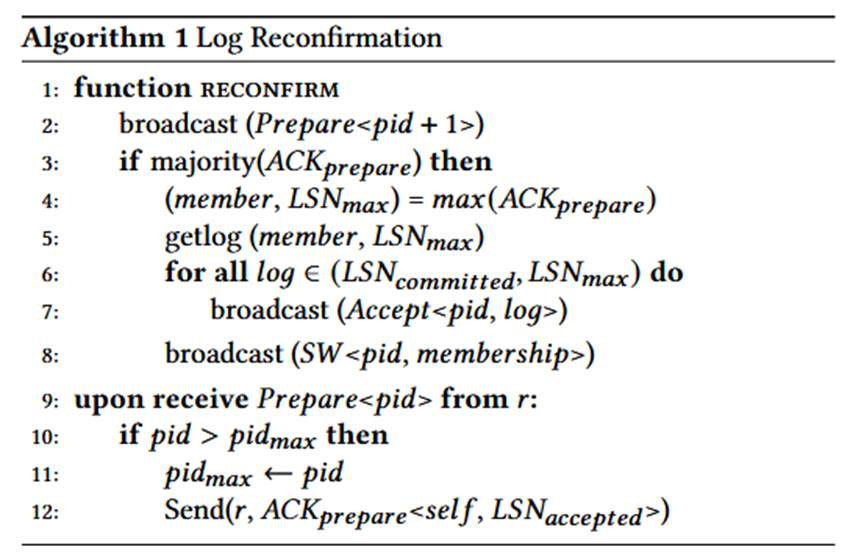
角色转变：



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）

由于数据分区通常分布在多个数据流中，不同的的事务修改了不同数据流中的分区，lsn是本地排序的，所以这些日志的lsn不能进行比较来跟踪事务的顺序。

PALF 提供另一种日志条目标识符，即更改序列号 (CSN)，它与 LSN 保持一致，并反映多个 PALF 组中的事务顺序。CSN 是一个 64 位整数，存储在每个日志条目的header中。PALF 不会推断 CSN 的含义，这意味着维护和识别 CSN 的开销非常小。当调用append方法时，Leader中的日志排序器会为日志分配 LSN 和 CSN。 PALF 维护 CSN 的不变性：在 PALF 组内单调递增。在每个 PALF 组内，对于任意两个日志 α 和 β，如果 α 的 LSN 大于或等于 β 的 LSN，那么 α 的 CSN 必须大于或等于 β 的 CSN。Formally,

LSNα ≥ LSNβ → CSNα ≥ CSNβ, ∀α, β.

该不变式保证了 CSN 在 PALF 组内单调增长，从而与 LSN 的顺序保持一致

PALF 提供了另一个关键保证：对于append方法，其输入参数refcsn和输出参数 csn始终遵循 csn >= refcsn。refcsn作为参考；该保证表示日志条目必须在 refcsn 发生事件后添加。因此，如果 refcsn是一个具有全局意义的值，日志的 csn就能反映整个系统的顺序。

OceanBase 数据库使用 CSN 为事务提供全局意义的提交版本。当一个事务要提交时，事务引擎会从全局时间戳中获取一个时间戳，并将带有时间戳的提交记录追加为refcsn。append方法返回的 CSN 会跟踪 refcsn所指示的顺序，并作为事务的提交版本。(注，CSN 不是由全局时间戳oracle生成的，全局时间戳oracle的值可能大于当前的全局时间戳。因此，未来从全局时间戳oracle中获取较小可读版本的读取请求可能无法看到该事务。为避免这种情况，事务引擎在全局时间戳大于 CSN 时才会响应客户端。通过事务引擎和 PALF 之间的合作，CSN 成功地跟踪了 PALF 组之间的事务顺序。

另一项受益于 CSN 的数据库功能是fallower读取。fallower读使fallower副本能够为只读请求提供最终一致性保证，从而减少延迟。T 的读取请求若能读取fallower中的完整数据，则要求 CSN 小于 T 的所有日志都已在fallower中重放，且任何未来写入的提交版本都应大于 T。CSN 的单调递增特性很自然地提供了这种保证。日志按照 LSN 的顺序复制和重放给fallower，这与 CSN 的顺序一致。 如果 CSN 为 T 的日志已在fallower日志中重放，则fallower日志的 CSN 必须大于 T。

3，数据更改同步

除了为事务提供服务，分布式数据库还充当数据流的源头。下游应用程序可通过同步物理日志中记录的数据变化，提供各种服务。有两种典型的应用场景：physical standby databases 和database restore.。

物理备用数据库最重要的特点之一是数据库级数据保护和灾难恢复，如果原主数据库不可用，物理备用数据库可以通过故障转移操作切换为主数据库。

数据库还原是高可靠性功能的核心组成部分。如果数据因存储介质损坏或人为错误而丢失，可使用离线存储（如 NFS 或云对象存储）中存储的归档日志来恢复相同的数据库。实现这些功能的挑战之一是将日志从一个PALF组（或外部存储）同步到另一个PALF组。此外，这些 PALF 组应该是独立可用的。

一个简单的解决方案是从主数据库的 PALF 组中读取日志条目，并将日志作为记录追加到备用数据库的相应 PALF 组中。但是，共识协议会在追加记录上追加日志头以进行复制，这就导致主数据库和备用数据库的日志格式不一致。

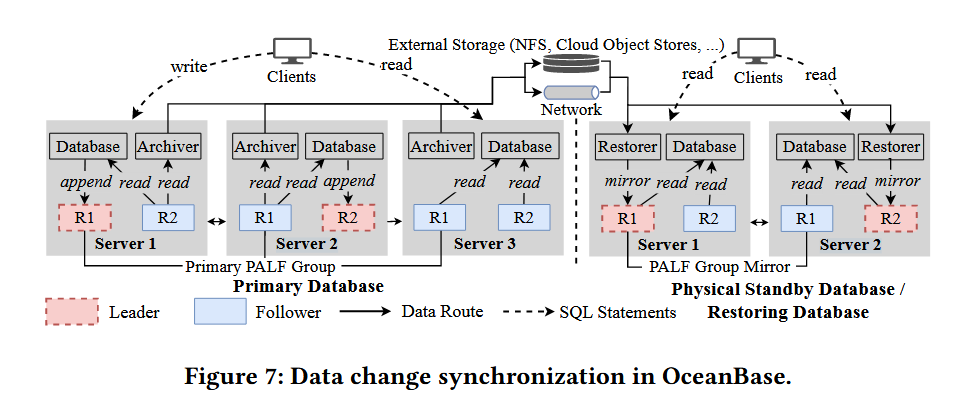
3.1 PALF 组镜像

我们将跨 PALF 组同步数据变化的要求抽象为一个基本要素：PALF 组镜像，它是一个独立的 PALF 组，执行相同的共识协议。它维护着数据变更前缀的镜像，数据变更前缀存储在主 PALF 组或外部存储中。PALF 组镜像可独立重新配置，并根据需要切换为主组。

主 PALF 组和 PALF 组镜像之间最重要的区别之一是写日志记录的模式。在主 PALF 组中，日志记录被追加到 PALF 组，并附有日志头，然后通过共识协议复制到副本。至于 PALF 组镜像，它只接受主 PALF 组提交的日志。当提交日志被镜像到Leader时，日志头的某些字段（如 ProposalID）将被替换为Leader自己的值。Leader会重复使用原始日志条目的 LSN 和 CSN，将日志存储到 LSN，并复制日志给fallower。

为了区分主 PALF 组和镜像组，我们提出了两种访问模式 "主 "和 "镜像"。PALF 组的访问模式可通过故障转移或切换操作进行切换。问题是如何以原子方式向所有副本广播新的访问模式。显然，访问模式的原子广播等同于共识问题。因此，我们采用了基本的 Paxos 来切换 PALF 组的访问模式，并将每个副本的访问模式存储到 MetaStorage 中。

如图 7 所示，physical standby databases中的所有 PALF 组都是主数据库中 PALF 组的镜像，database restore中的所有 PALF 组都是存储在外部存储中的数据变化的镜像。主 PALF 组提交事务日志后，log archivers会从每个 PALF 组读取日志，然后将其存储到外部存储中或传输到备用数据库。在physical standby databases中，restorers从archiver接收日志，并将日志映射mirror到 PALF 组镜像的Leader。日志由Leader提交后，将重放给所有副本（包括Leader）中的事务引擎。因此，在主数据库中执行的事务将同步到物理备用数据库；数据库还原执行类似的程序。



值得注意的是，database restore中事务引擎与 PALF 组之间的交互与主数据库不同。在database restore中，事务引擎执行标准的 RSM 模型，所有副本只需从 PALF 副本读取日志记录，并将数据变化重放至数据分区。因此，尽管 PALF 复制的角色可能是Leader，但事务引擎的角色始终是fallower。

3.2 独立重新配置

PALF 的重新配置方法，每次只能添加或删除一个副本。Leader复制一个重新配置日志，记录新的成员资格，并与新成员资格确认一起提交。每个副本在收到更新的重新配置日志后，都会更新自己的成员资格。为使 PALF 组镜像能独立重新配置，PALF 将重新配置日志存储在 MetaStorage 中，与 LogStorage 分开。

独立重新配置日志独立存储可能会在成员变化（增加/替换）时，出现日志丢失或者脑裂的风险。

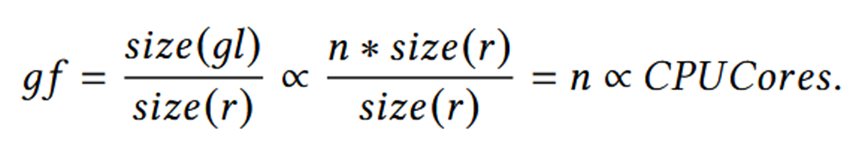
PALF 引入了对重新配置和Leader选举的限制，以解决这些异常现象。在重新配置时，Leader会向fallower发布新配置和日志屏障。 障碍是Leader发布配置时的日志尾数。每个fallower都会拒绝接受配置，直到其日志刷新到屏障之前。因此，在副本刷新到屏障之前，不会接受新配置。在选举leader时，PALF 会维护config version，以显示成员的版本；重新配置操作会递增config version。config version作为选举的主要优先级，因此进行过独立配置的副本的优先级会高，能够避免脑裂的出现。本文认为，尽管独立的元存储会给共识协议带来额外的复杂性，但它是有利的，因为它可以让日志消费者看不到元信息，并使 PALF 组镜像能够独立地重新配置集群。

4， 性能优化

**流水线复制**。为了提高吞吐量，PALF 利用现代多核处理器并发处理和复制日志。多个日志的共识相关状态被缓存在内存滑动窗口中，以避免 CPU 缓存缺失。 因此，多个日志的复制阶段可以重叠。

**自适应群组复制** 共识协议会给数据库带来额外的开销，其中包括至少两条网络信息（日志复制和确认）和一些 CPU 周期。在一个实例中批量处理多个日志条目是稀释共识产生的开销的常用方法。批处理日志的核心在于如何确定适当的批处理大小。

PALF 采用自适应组大小复制日志，以平衡延迟和吞吐量。Leader在组缓冲区内缓存已添加的日志。freeze操作会将缓存日志打包到组日志条目中，然后复制给fallower。组日志条目中日志条目的数量（组因子）取决于freeze操作的执行频率。自适应分组复制的关键思路是在低并发时定期冻结日志，在高并发时根据 I/O 工作者的反馈freeze日志，但客户端的并发程度很难直接测量。假设有 n 个客户端同时向 PALF 追加记录 r。如果所有客户端的日志都已添加，但尚未提交，则这些缓存日志应立即冻结到组日志 gl 中，而无需等待恒定的时间间隔。因此，并发程度（n）与组因子（gf，缓存日志的数量）相关，这很容易通过计算日志来衡量。这种关系可形式化为



实验表明，并发程度与 PALF 占用的 CPU 内核数成正比，因此组因子gf阈值可由硬件资源决定，而不是手动调整批量大小。如果组因子gf小于阈值，则表示并发度较低，如果 I/O 工作者处于空闲状态，PALF 将freeze日志；否则，PALF 将以固定间隔（默认为 1 ms）freeze日志。

**无锁写入路径** 高并发性非但不会提高吞吐量，反而会在发生严重争用时降低性能。因此，我们为 PALF 设计了无锁写入路径，以避免线程间的竞争。写入路径的主要组件是log sequencer 和 group buffer.。log sequencer按顺序为日志条目分配 LSN。我们采用了无锁log sequencer，以避免其成为瓶颈。 当线程将日志添加到 PALF 时，它会以原子方式将 LSN 尾部的值加载到临时变量中，更新临时值，并通过原子比较和交换操作将其存储到 LSN 尾部。如果比较和交换操作因并发追加而失败，那么线程会重新加载 LSN 尾部，并再次循环获取 LSN。

获取 LSN 后，多个线程会同时将日志条目填入group buffer。LSN 不仅是日志条目在磁盘上的地址，也是日志条目在group buffer中的偏移量，这意味着一个日志条目的预留缓冲区永远不会与另一个日志条目重叠。因此，组缓冲区不是竞争资源，多个线程可以同时将日志条目填充到不同的偏移量，而不会产生任何锁开销。