姓名： 李青坪

学号： MF1733034

日期： 2017.12.9

论文信息

Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: Wait-free Coordination for Internet-scale Systems[C]//USENIX annual technical conference. 2010, 8: 9.

1 简介

ZooKeeper 是一个分布式应用协同进程服务，目标是通过提供一个简单的、高性能的内核，在客户端构建更加复杂的协同原语。ZooKeeper 将来自群组消息、共享注册和分布式锁服务的元素合并到一个复制的、集中的服务中。ZooKeeper 的接口具有共享注册的无等待的特点，以及与分布式文件系统的高速缓存失效类似的事件驱动机制，来提供简单但强大的系统服务。

ZooKeeper 可以看作是一个分布式文件系统，它的数据节点 (ZNode) 类似于文件和目录，可以存储一些元数据、节点的 IP 和端口等。但每个节点可以存储的数据大小默认是限制在 1MB 以内的，但也可以根据实际情况进行修改。而且因为 ZooKeeper 的数据是保存在内存中的，所以访问更快，为了可恢复，对数据的更新将会被记录到磁盘中。

在设计协同服务的时候，ZooKeeper 不是考虑在服务端实现特定的、约束的原语，而是考虑公开接口来使用户定义自己的原语。故 ZooKeeper 实现了协同内核，允许新的原语而不会对服务内核作出改变。在设计 ZooKeeper API 的时候，阻塞原语 (比如锁) 被移除了。因为在协同服务中阻塞原语容易使操作慢的、出错的客户端会影响操作快的客户端的性能，如果采用阻塞原语，前者会使系统的吞吐量严重降低。故 ZooKeeper 设计了一个像文件系统一样有层次结构的无等待的数据对象——ZNode，ZNode 映射客户端应用的抽象，特别是那些相关的用于协同目的的元数据。客户端可以创建永久节点和临时节点两种 ZNode，前者是需要明确的创建和删除，且可以拥有子节点；后者可以由系统自动删除掉，且不能拥有子节点。

由于 ZooKeeper 里面大多数操作是读操作，因此要扩展读操作的吞吐量。在 ZooKeeper 中使用 Zab 协议来保持一致性，但对读操作，服务器都是在本地读取数据，为了提升读取的吞吐量，没有使用 Zab 完全对读操作进行排序，但这并不影响读取的准取性。同时，采用客户端缓存的机制也有助于提升读取性能，ZooKeeper 采用 watch 机制开获取数据对象的更改，相比于直接操作缓存数据而使更新被阻塞，watch 机制更能保证读取的性能。

2 ZooKeeper 的顺序保证

客户端通过 ZooKeeper API 向服务器发送请求，ZooKeeper 需要保证请求执行的顺序。采用两种基本的顺序保证：**线性化写入**和**先进先出的客户端顺序**，保证所有客户端的写操作序列化执行并确保优先权；保证给定客户端的请求按照客户端发送请求的顺序执行。要理解这两种保证机制是如何交互的，考虑一个新的 leader 指挥 ZooKeeper 的其他进程的情景。我们对整个系统有两个要求：

- 当 leader 开始对配置进行更改时，其他进程不能使用正在被更改的配置
- 如果 leader 在配置被完全修改好之前崩溃了，其他进程不能使用这个被部分更改的配置

ZooKeeper 的实现机制是：首先，leader 会在节点树中新建一个叫 ready 的节点，只要这个节点存在，其他服务器进程会向该节点读取配置信息。当新的 leader 被选举出来的时候，会删除掉之前的 ready 节点，并对配置节点进行修改，等到修改完成后，创建新的 ready 节点。这个操作会在很短的时间内完成，既满足了以上两个要求，又通过多个服务器对客户端请求的异步调用提升了系统性能。如果在新的 leader 开始修改之前，客户端发现了 ready 节点，客户端会在它看到更改发生后系统将要达到的新状态之前收到通知。

3 原语的实例

配置管理：初始时，配置被存储在一个叫 z_c 的 ZNode 中，进程读取 z_c 里的配置并启动，一旦 z_c 里的配置被更改了，进程将被通知被读取新的配置。

集中点：客户端创建一个集中点 ZNode，z_r，z_r 中包含了该客户端 master 进程的一些重要的信息，该客户端上的 worker 进程读取 z_r 中的信息并连接到 master 进程。

组成员关系：创建一个 ZNode，z_g 代表组，当组成员中的某个进程开始的时候，将在 z_g 下生成一个临时节点。通过列出 z_g 的孩子结点即可获得组相关信息。

读写锁：

```
Write Lock
1 n = create(l + "/write-", EPHEMERAL|SEQUENTIAL)
2 C = getChildren(l, false)
3 if n is lowest znode in C, exit
4 p = znode in C ordered just before n
5 if exists(p, true) wait for event
6 goto 2
Read Lock
1 n = create(l + "/read-", EPHEMERAL|SEQUENTIAL)
2 C = getChildren(l, false)
3 if no write znodes lower than n in C, exit
4 p = write znode in C ordered just before n
5 if exists(p, true) wait for event
6 goto 3
```

注意 Read Lock 的第 3 行，如果 C 中没有比 n 更小的 Write ZNode 了，则将 Read Lock 赋给 n。保证比 n 小的 Write ZNode 都在 n 之前被执行。

双重屏障：屏障中设置了阈值，只有进入屏障的进程数达到阈值，里面的进程才能开始执行；如果屏障里面没有子节点了，也就是里面所有的进程都执行完了，进程才能离开屏障。

4 ZooKeeper 的实现

在收到请求后，服务器通过请求处理器来准备执行该请求。当受到的是写请求时，请求处理器会计算写操作过后系统可能达到的状态并把写操作转化为捕获新状态的一个事务。当服务器准备好执行请求时，如果该请求需要服务器之间的协同，(比如写操作) 则这个请求将被发送到 leader 节点，leader 节点执行该请求并使用到一致性协议——Zab 协议来满足服务器之间数据的一致性。Zab 协议是一个原子广播协议，它通过集群中大多数的节点来决定某一个方案的可行性。如果集群中有 2f+1 个服务器，Zab 可以容忍 f 个服务器出错。Zab 保证 leader 做的更改按照更改请求发送过来的顺序被广播到各个服务器端。对于读请求，服务器只需要读取本地的数据库状态并生成回应。

为了能够使服务器恢复，我们需要将更新操作记录到磁盘上，即在磁盘上保留一个被提交的操作的重现日志，并生成内存数据库的定期快照，该快照在原子广播的时候起作用：当服务器恢复的时候，Zab 协议会将该服务器记录的最新的一次快照开始后传递的所有消息重新传递到该服务器上。

客户端和服务器交互时，使用 zxid 定义读请求和写请求之间的偏序，每个读操作被 zxid 标识，如果读操作的 zxid 与服务器端不匹配，表明服务器端发生了新的写操作 (事务)，则读操作请求被拒绝，这样读操作的性能就能有很大提升。但这种快速读操作不保证读操作的优先权顺序，由此可能导致读取到旧版本的数据，如果要保证每次读取都能读到最新的数据，可以使用 sync 操作。将 sync 操作放在读操作的前面，先进先出的顺序保证策略会保证读操作会映射出 sync 操作执行之前 ZooKeeper 里面发生的任何更改操作。

同时，为了保持客户端和服务端的连接，如果客户端不太活跃，需要没 s/3 毫秒向服务器发送一次心跳，s 表示会话超时时间。

5 主要贡献

ZooKeeper 是被设计为提供分布式系统的无等待的协同服务的框架，是 Google 的 Chubby 的一个开源的实现，是 Hadoop 和 Hbase 的重要组件。ZooKeeper 设计的时候保证了最终一致性，即无论客户端向哪个服务器发送请求，客户端最终将得到相同的视图，这是 ZooKeeper 最重要的性能。ZooKeeper 将进程间的协同问题转移到分布式应用中来，设计时用到协同服务、容错系统、分布式算法和文件系统的概念。ZooKeeper 提供了一套完整的客户端 API 供开发者自定义原语，开发者可以根据自身需要实现相应的功能。读操作就在服务器本地进行，使得读操作的性能很高；写操作交由 leader 完成，并在完成后通过 Zab 协议将更新后的数据广播到其他 follower 服务器上。这种实现机制使得 ZooKeeper 的横向扩展能力较强。

ZooKeeper 没有假设服务器会出现拜占庭错误，但实现了一些机制来解决一些非恶意的拜占庭错误，这可能是 ZooKeeper 存在的一些小的瑕疵，不过在实际生产环境下，还没有遇到需要使用完全的拜占庭容错协议的。在测试屏障的性能时，应该将数据以折线图的形式呈现，这样在提到执行完所有的屏障的用时随屏障的数量呈线性增长这一观测结果的时候更直观。

参考文献

[1] http://holynull.leanote.com/post/Zookeeper

[2] http://www.jianshu.com/p/cfcacc87d74a

[3] https://www.ibm.com/developerworks/cn/opensource/os-cn-zookeeper/