黑客派

Zookeeper 应用场景之分布式队列

作者：guobingwei

原文链接：https://hacpai.com/article/1553921806541

来源网站：黑客派

业界存在不少消息中间件产品，实现各不相同。这篇文章主要介绍基于 ZooKeeper 实现的分布队列。分布式队列，简单来说分为两大类，一种是常规的先入先出队列，另一种是要等到所有元素集之后才能统一安排执行的 Barrier 模型。

<h2 id="1--FIFO-先入先出">1. FIFO：先入先出</h2>
FIFO 队列是一种非常典型的按序执行的队列模型：先进入队列的请求操作完成后，才会开始处后面的请求。
使用 zk 实现 FIFO 队列，和 zk 对于共享锁的实现类似。FIFO 队列就类似于一个全局的共享锁型。所有客户端都会到一个节点（比如 <code>queue_fifo</code>）下创建一个临时顺序节点。<b> <img src="https://static.hacpai.com/images/img-loading.svg" alt="jiedian" data-src="http:/p0.meituan.net/scarlett/ecfa19da7facb612b1f5993f8e3f1f5825005.png">
 创建完节点，据如下四个步骤确定执行顺序。
<blockquote>

1.通过调用 getChildren() 接口来获取 /queue_fifo 节点下的所有子节点，即获取队列中的所有素。
2.确定自己的节点序号在所有子节点中的顺序
3.如果自己不是序号最小的子节点，那么就需要进入等待，同时向比自己序号小的最后一个节注册 Watcher 监听
4.接收到 Watcher 通知后，重复步骤 1

</blockquote>
整个通过流程如下图所示：
<img src="https://static.hacpai.com/images/img-loading.svg" alt="工作流程示意图" data-rc="http://p1.meituan.net/scarlett/cf489b0f4f8ebb79a729faa3eda94c5690801.png">
<h2 id="2--Barrier-分布式屏障">2. Barrier：分布式屏障</h2>
Barrier 在分布式系统中特指系统之间的一个协调条件，规定了一个队里的元素必须都集聚之后能统一进行安排，否则一直等到。往往出现在一些大规模分布式并行计算的应用场景上：最终的合并算需要基于很多并行计算的结果来进行。这些队列其实是在 FIFO 队列的基础上进行了增强。大致的计思路如下：
 开始时，<code>/queue_barrier</code> 节点是一个已经存在的默认节点，并将节点的数据内容赋值为数字 n 表示 Barrier 值，表示当 <code>/queue_barrier</code> 节点下子节点个数达到 n 个后，才会打开 Barrier。之后所有的客户端都会在 <code>/queue_barrier</cod> 节点下创建一个临时节点，如下图所示：
<img src="https://static.hacpai.com/images/img-loading.svg" alt="示意图" data-src="htt://p0.meituan.net/scarlett/c40fc3a7fb5b65abe6d5846e1bf3600e20404.png">
创建完节点后，接下来的流程为：
<blockquote>

1.通过调用 getData() 接口获取 <code>/queue_barrier</code> 节点的数据内容
2.通过调用 getChildren() 接口获取 <code>/queue_barrier</code> 节点下的所有子节点，获取队列中的所有元素，同时注册对子节点列表变更的 Watcher 监听。
3.统计子节点个数
4.如果子节点个数不足 n 个，就需要等待
5.接收到 Watcher 通知后，重复步骤 2

</blockquote>

整个 Barrier 队列的工作流程如下：
<blockquote>
来自 <code>《从paxos到zookeeper 分布式一致性原理与实战》</code>，稍微进行了归纳结。
</blockquote>