二

33 结束语 分布技术发展与 ZooKeeper 应用前景

到目前为止，本专栏的所有课程已经结束了。在这个专栏中，我们主要介绍了分布式系统架构最核心的问题：如何解决分布式环境一致性。围绕这个问题，我为你引入了分布式一致性问题的工业解决方案——开源框架 ZooKeeper。

通过本专栏的学习，可以帮助你掌握实现分布式一致性的各种算法，包括：二阶段提交 、三阶段提交、 ZooKeeper 的 ZAB 协议算法以及 Paxos 算法。掌握了这些算法的理论知识后，我们又进一步分析了，代码层面的底层架构设计思想和实现过程。实现一个分布式系统会面临很多的挑战，在明确问题的原因后，更应该灵活运用学到的知识，找到问题的最优解。

在结束语中，我打算**添加一个小彩蛋**，向你介绍除了已经掌握的 ZooKeeper 框架的 ZAB 协议算法之外，解决分布式一致性问题的另一种方法：Raft 算法。

彩蛋：Raft 算法

相比之前学到的 ZAB 协议算法和 Paxos 算法，Raft 算法的实现逻辑则更为简单。Raft 算法将分布式一致性问题的解决，分解为一个个更加细小简单的问题。

实现过程

它的实现过程与 Paoxs 等一致性协议算法有相似的地方，但也有其自身的特点。其中强领导者、领导选举、成员关系调整是其特有的概念。

强领导者

在 ZAB 协议算法中，集群中会有一个 Leader 服务器，同样，Raft 算法也需要在集群中创建一个领导者服务。与 Leader 服务器相比，Raft 算法中的领导者服务器具有更强的功能，比如在数据的同步方式上，它只通过领导者服务发送给集群中的其他服务器。

领导选举

与我们之前介绍的一致性协议不同，在领导者服务的选举方式上，Raft 算法只采用随机技术器的方式。在该随机计数器产生的超时时间内，集群中的服务器各自向网络中广播投票信息，当某一台服务器收到超过集群中一半服务器的响应后，该台服务器就被选举为新的领导者。

成员关系

在处理事务性的回来请求时，Raft 算法中的领导者服务器会执行该条会话操作，但并不提交，只是将该操作写入到日志中，再发送给集群中的其他服务器。当接收到超过一半的服务能够正常操作的反馈信息后，领导者服务器才最终提交本次会话请求操作，并向集群中的其他服务器发送提交请求。

总结

只要采用分布式系统架构，都避免不了一致性的问题。本专栏我们主要学习了 Paxos 算法和 ZAB 协议算法。本节课又介绍了一种一致性协议算法——Raft。Paoxs 算法是解决一致性问题的最完善的算法，但对其底层实现的细节并没有给出更详细的解释。整个算法的实现难度较大。ZAB 协议和 Raft 算法都可以说是在 Paxos 算法的基础上，做了各自的优化和改进，给我们提供了一致性协议的工业化解决方案。你可以深入研究 Paxos 算法的理论，并结合 ZooKeeper 框架的设计思路，设计开发自己的一致性协议框架。

上一页