二

用户故事 学而时习之，不亦乐乎

你好，我是大熊。

目前是一名 Java 后端开发工程师，坐标广州，主要从事人事系统的研发工作。

我算是《中间件核心技术与实战》这个专栏最早的一批读者了，专栏上线伊始我就被这个“高大上”的专栏名称给吸引住了。试看了前两节，感觉干货满满，就开始“路转粉”。专栏上线至今也有一段时间了，我也有了不少收获，今天就在这里跟你分享一下。希望也能给你提供一些新的思考和启发。

为什么要学习中间件?

作为一名Javaer，只要公司在使用的技术栈不是太老，工作中就难免与各式各样的中间件打交道。有了使用的需求，就有了了解的必要。不知道你是不是跟我一样，在使用各种中间件组件的时候，都是把它们当成一个“黑盒子”。从官方的示例教程中简单学了一下搭建方法、使用 API 以及大致原理之后，就在项目里用起来，也不去深挖中间件背后的实现原理。

这种使用方式不能说不好，但是有个很致命的问题。那就是，一旦项目中使用的中间件组件出现问题，我们往往就会手足无措。幸运的话，我们可以“面向谷歌编程”，在网上搜索报错信息，找到别人的解决方案。

但是，网上的信息稂莠不齐，而且别人的场景也不见得跟你的一样，把别人的方案生搬过来也不一定能起什么作用。此时，我们就不得不求助于中间件的源码了。正如 Linux 创始人 Linus Torvalds 所言："Talk is cheap,show me the code."阅读中间件的源码是了解中间件设计最直接也是收获最大的途径。

我也分享一下我最近的一段经历。当时，我们项目用到了 RabbitMQ。为了提高系统可用性，我要在我们公司的 Kubernetes 集群上部署一个三个节点的 RabbitMQ 集群。我按照官网的 YAML 文件示例，在 Kubernetes 集群新建了各种资源，但是在最后的启动阶段，我发现三个节点的容器都无法正常启动。检查 RabbitMQ 的启动日志之后，我发现了一段错误日志，这段日志的大致意思是，“有一个名叫 Host 的字段没有找到”。

我在网上搜了一圈，也没找到这方面的回答。这就让我很抓狂。后来没办法，我只能硬着头皮去 GitHub 翻出 RabbitMQ 集群发现组件的源码来看（这里不得不吐槽一句，Erlang 是真的难懂）。

我定位到日志输出的那段代码之后，仔细阅读了一遍，发现它的执行逻辑是这样的：容器在启动的时候，发现组件会向 Kubernetes 的 API Server 请求处于同一个 StatefulSet 下的所有副本信息，然后从副本信息中提取 Host 字段，让它们彼此进行注册。

那我们的系统为啥会找不到 Host 字段呢？

为了搞明白这个问题，我自己去尝试请求了一下那个 API，发现在这一步返回的信息中就没有 Host 字段。最后，确定是我们公司的 Kubernetes 集群部署存在问题。将问题反馈给维护方之后，问题就成功解决了。

这个专栏给我带来了什么帮助？

从这个例子就可以看得出，了解中间件的设计原理和底层实现确实十分重要，而阅读源码就是达到这个目标一个很好的方法。但是，任何一个大型的中间件项目，它的源代码行数都是上万级别的，我们漫无目的地去看项目源码显然效率不高，而且也容易抓不住重点，导致最后迷失在函数调用中。而这个专栏正好在代码的汪洋大海中给我们指明了一个方向。

在专栏的基础篇，丁威老师介绍了中间件编程必知必会的数据结构，多线程编程以及 IO 编程等几个领域的核心知识。这些知识都是实现高性能中间件的基石，为我们了解中间件的底层实现扫清了障碍。

我个人尤其喜欢第六讲《锁：如何理解锁的同步阻塞队列与条件队列？》，这一讲从源码的角度剖析了 JUC 里一些基于 AQS 常用锁的实现。带着我们一步一个脚印，先掌握执行流程，然后又到代码中去求证。做到了知其然，也知其所以然。

在实战篇里，丁老师分享了自己在实践中对 Dubbo 和消息中间件的一些巧妙应用。有些使用方式不禁让我感叹：“原来还能这么用！”这也极大地扩宽了我的视野。

值得一提的是，丁老师还分享了在自己的项目中用 Dubbo 和 RocketMQ 落地蓝绿发布的方案。恰好这段时间，我自己所在的项目组也计划搭建租户隔离环境和灰度发布环境，专栏内容给了我不少启发。

最开始我们用的是 RabbitMQ 的虚拟主机机制，想通过它对不同租户的业务消息进行隔离。但是这种做法太过绝对了，导致一个多租户公用的服务在想要监听所有租户的相关消息时，实现成本太高。看了丁老师对“基于消息主题的隔离机制”的分析之后，我们放弃了每个租户配置一个虚拟主机的方案，改为每个租户都有属于自己的主题队列，并以租户标识作为

后缀。这样一来，多个租户公用的服务就可以使用通配符模式消费了。

我是如何学习本专栏的?

最后，我想再聊聊我学习这个专栏的方法。

丁老师的这个专栏干货满满，有时候看了一遍却感觉没有消化多少。遇到这样的内容，我就会隔一段时间再复习一遍。第一遍看不懂的东西，过几天来看或许会豁然开朗。这大抵是“书读百遍，其意自现”的道理吧。

极客时间的专栏大都配有音频，很方便我们在上班通勤这段时间学习。不过，因为专栏内容很多都附有代码样例，这都是音频呈现不出来的，而且手机上也无法编译运行样例，所以我还是倾向于专门规划一段时间来学习，这个专栏的知识密度更应该如此。每次的学习时间不需要太长，半个小时到一个小时左右即可。

另外，我觉得评论区也对学习有非常不错的辅助作用。学完一节课之后，可以扫一遍评论区的内容，看看同学们提的问题你有没有想到。如果你也想到了，那你的回答跟老师的有什么差别？这些都是检验自己学习成果很好的手段。

最后我想说，好记性不如烂笔头。我随时都会拿个笔记本把自己感兴趣的，或者没听过的新概念、新技术记录下来。如果时间允许，我还会在纸上画出知识点之间的关联，时常翻阅温习一下。

我们每个人的实践经历不同，学习同一个专栏也一定会有不同的理解与收获。也期待的你的留言，我们共同进步。

上一页 下一页