

20 | 基于Raft的分布式KV系统开发实战（二）：如何实现代码？

2020-03-30 韩健

分布式协议与算法实战

[进入课程 >](#)



讲述：于航

时长 13:34 大小 12.44M



你好，我是韩健。

学完 [上一讲](#) 后，相信你已经了解了分布式 KV 系统的架构设计，同时应该也很好奇，架构背后的细节代码是怎么实现的呢？

别着急，今天这节课，我会带你弄明白这个问题。我会具体讲解 [分布式 KV 系统](#) 核心功能点的实现细节。比如，如何实现读操作对应的 3 种一致性模型。而我希望你能在课下反复运行程序，多阅读源码，掌握所有的细节实现。



话不多说，我们开始今天的学习。

在上一讲中，咱们将系统划分为三大功能块（接入协议、KV 操作、分布式集群），那么今天我会按顺序具体说一说每块功能的实现，帮助你掌握架构背后的细节代码。首先，先来了解一下，如何实现接入协议。

如何实现接入协议？

在 19 讲提到，我们选择了 HTTP 协议作为通讯协议，并设计了"/key"和"/join"2 个 HTTP RESTful API，分别用于支持 KV 操作和增加节点的操作，那么，它们是如何实现的呢？

接入协议的核心实现，就是下面的样子。

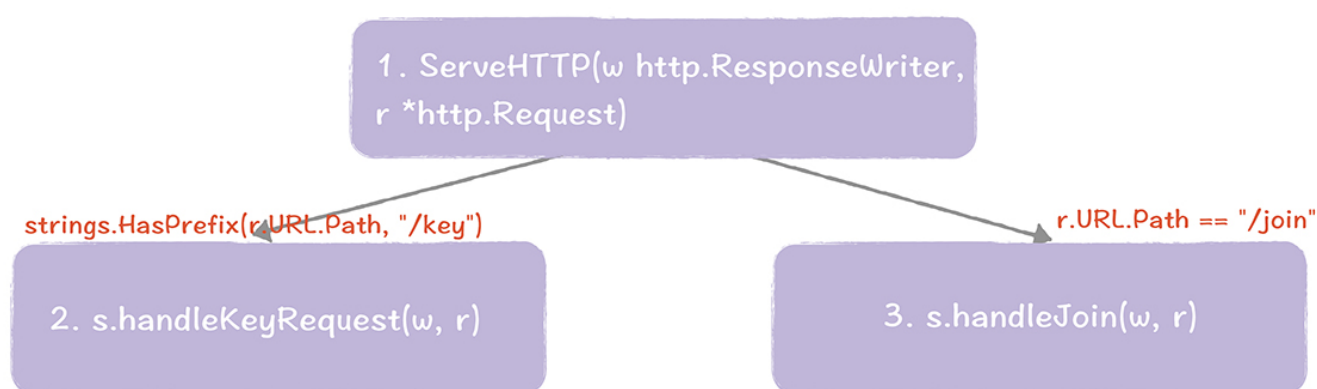



图1

我带你走一遍这三个步骤，便于你加深印象。

1. 在 ServeHTTP() 中，会根据 URL 路径设置相关的路由信息。比如，会在 handlerKeyRequest() 中处理 URL 路径前缀为"/key"的请求，会在 handleJoin() 中处理 URL 路径为"/join"的请求。
2. 在 handleKeyRequest() 中，处理来自客户端的 KV 操作请求，也就是基于 HTTP POST 请求的赋值操作、基于 HTTP GET 请求的查询操作、基于 HTTP DELETE 请求的删除操作。
3. 在 handleJoin() 中，处理增加节点的请求，最终调用 raft.AddVoter() 函数，将新节点加入到集群中。

在这里，需要你注意的是，在根据 URL 设置相关路由信息时，你需要考虑是路径前缀匹配（比如 strings.HasPrefix(r.URL.Path, "/key"）），还是完整匹配（比如 r.URL.Path == "/join"），避免在实际运行时，路径匹配出错。比如，如果对"/key"做完整匹配（比如

`r.URL.Path == "/key")` , 那么下面的查询操作会因为路径匹配出错, 无法找到路由信息, 而执行失败。

 复制代码


```
1 curl -XGET raft-cluster-host01:8091/key/foo
```

另外, 还需要你注意的是, 只有领导者节点才能执行 `raft.AddVoter()` 函数, 也就是说, `handleJoin()` 函数, 只能在领导者节点上执行。

说完接入协议后, 接下来咱们来分析一下第二块功能的实现, 也就是, 如何实现 KV 操作。

如何实现 KV 操作?

上一节课, 我提到这个分布式 KV 系统会实现赋值、查询、删除 3 类操作, 那具体怎么实现呢? 你应该知道, 赋值操作是基于 HTTP POST 请求来实现的, 就像下面的样子。

 复制代码

```
1 curl -XPOST http://raft-cluster-host01:8091/key -d '{"foo": "bar"}'
```

也就是说, 我们是通过 HTTP POST 请求, 实现了赋值操作。

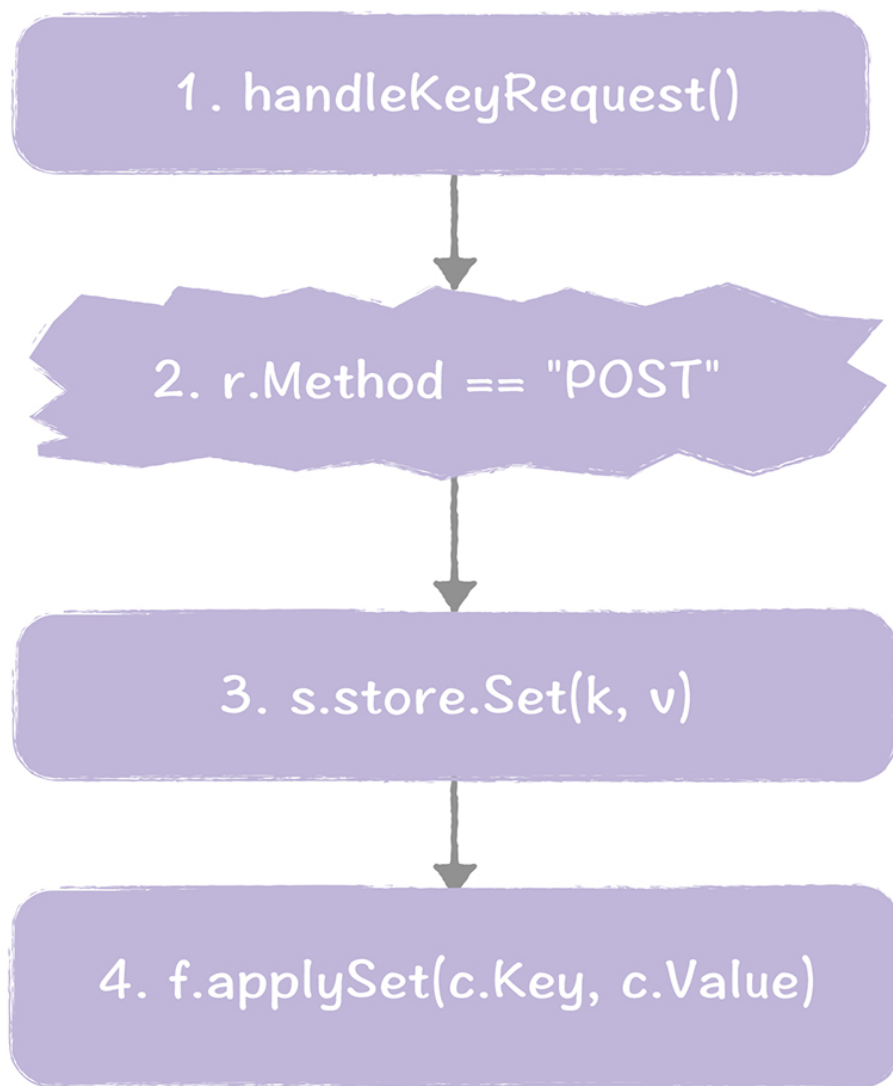


图2

同样的，我们走一遍这个过程，加深一下印象。

1. 当接收到 KV 操作的请求时，系统将调用 `handleKeyRequest()` 进行处理。
2. 在 `handleKeyRequest()` 函数中，检测到 HTTP 请求类型为 POST 请求时，确认了这是一个赋值操作，将执行 `store.Set()` 函数。
3. 在 `Set()` 函数中，将创建指令，并通过 `raft.Apply()` 函数将指令提交给 Raft。最终指令将被应用到状态机。
4. 当 Raft 将指令应用到状态机后，最终将执行 `applySet()` 函数，创建相应的 key 和值到内存中。

在这里，我想补充一下，FSM 结构复用了 Store 结构体，并实现了 `fsm.Apply()`、`fsm.Snapshot()`、`fsm.Restore()` 3 个函数。最终应用到状态机的数据，以

map[string]string 的形式，存放在 Store.m 中。

那查询操作是怎么实现的呢？它是基于 HTTP GET 请求来实现的。

```
1 curl -XGET http://raft-cluster-host01:8091/key/foo
```

复制代码

也就是说，我们是通过 HTTP GET 请求实现了查询操作。在这里我想强调一下，相比需要将指令应用到状态机的赋值操作，查询操作要简单多了，因为系统只需要查询内存中的数据就可以了，不涉及状态机。具体的代码流程如图所示。

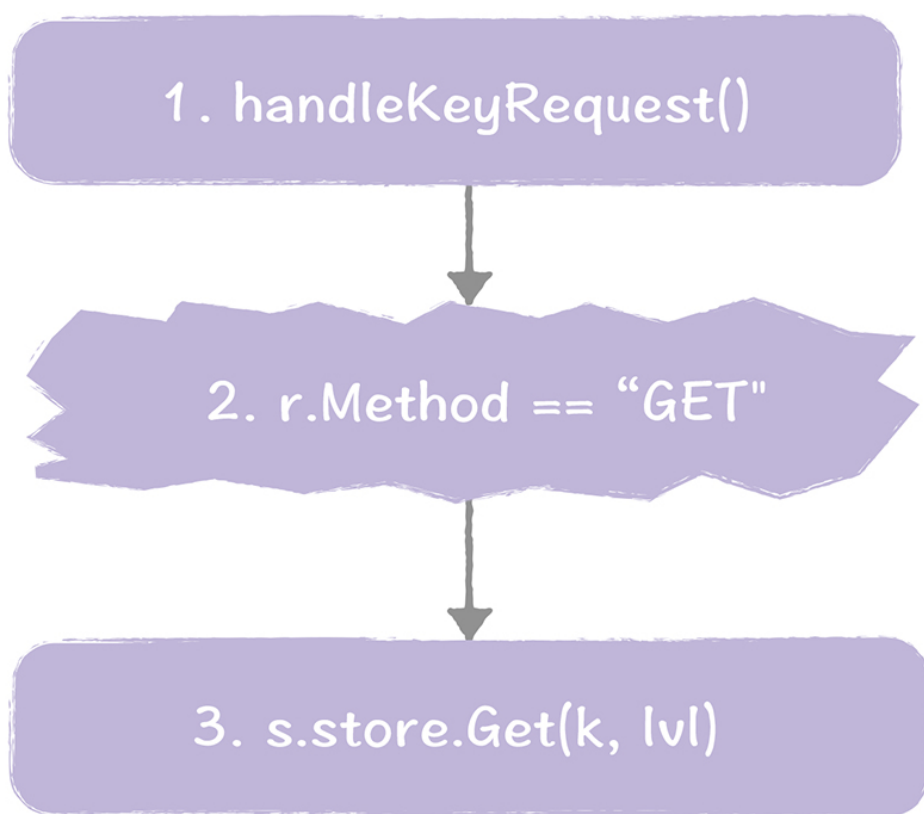



图3

我们走一遍这个过程，加深一下印象。

1. 当接收到 KV 操作的请求时，系统将调用 handleKeyRequest() 进行处理。
2. 在 handleKeyRequest() 函数中，检测到 HTTP 请求类型为 GET 请求时，确认了这是一个赋值操作，将执行 store.Get() 函数。
3. Get() 函数在内存中查询指定 key 对应的值。

而最后一个删除操作，是基于 HTTP DELETE 请求来实现的。

```
1 curl -XDELETE http://raft-cluster-host01:8091/key/foo
```

 复制代码

也就是说，我们是通过 HTTP DELETE 请求，实现了删除操作。

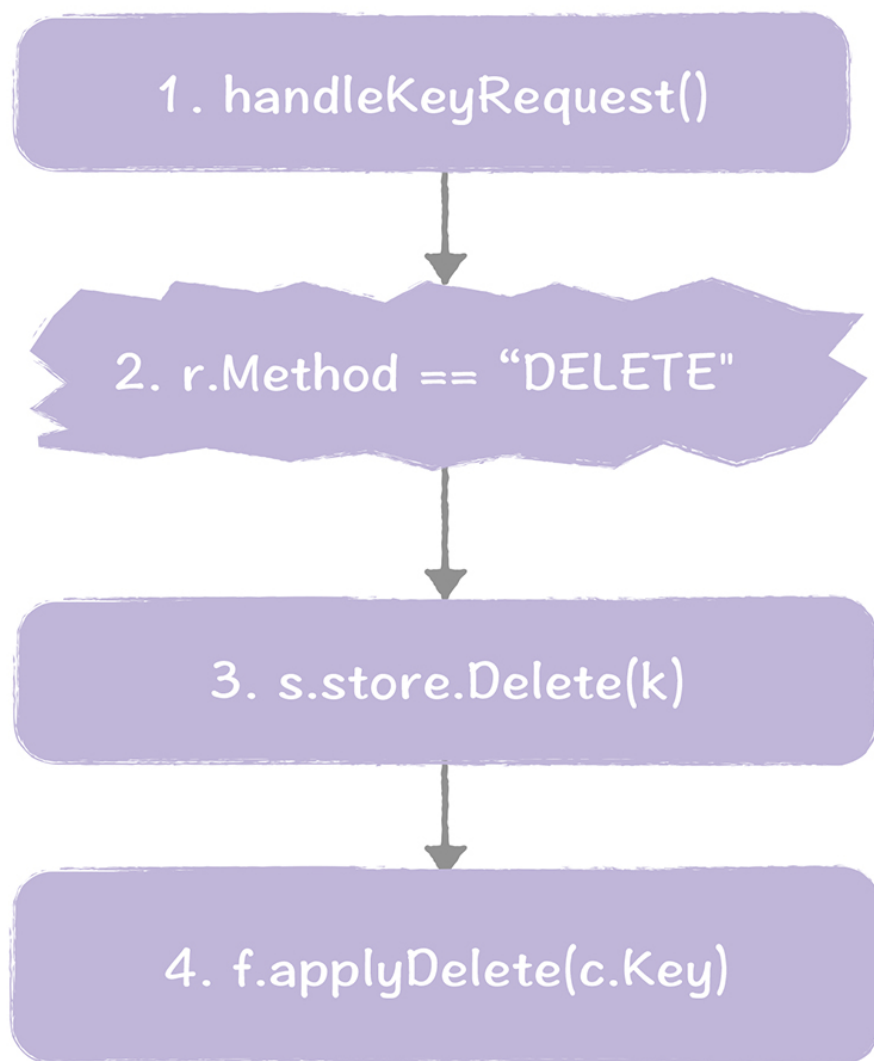


图4

同样的，我们走一遍这个过程。

1. 当接收到 KV 操作的请求时，系统将调用 `handleKeyRequest()` 进行处理。
2. 在 `handleKeyRequest()` 函数中，检测到 HTTP 请求类型为 DELETE 请求时，确认了这是一个删除操作，将执行 `store.Delete()` 函数。

3. 在 Delete() 函数中，将创建指令，并通过 raft.Apply() 函数，将指令提交给 Raft。最终指令将被应用到状态机。
4. 当前 Raft 将指令应用到状态机后，最终执行 applyDelete() 函数，删除 key 和值。

学习这部分内容的时候，有一些同学可能会遇到，不知道如何判断指定的操作是否需要在领导者节点上执行的问题，我给的建议是这样的。

需要向 Raft 状态机中提交指令的操作，是必须要在领导者节点上执行的，也就是所谓的写请求，比如赋值操作和删除操作。

需要读取最新数据的查询操作（比如客户端设置查询操作的读一致性级别为 consistent），是必须在领导者节点上执行的。

说完了如何实现 KV 操作后，来看一下最后一块功能，如何实现分布式集群。

如何实现分布式集群？

创建集群

实现一个 Raft 集群，首先我们要做的就是创建集群，创建 Raft 集群，主要分为两步。首先，第一个节点通过 Bootstrap 的方式启动，并作为领导者节点。启动命令就像下面的样子。

 复制代码

```
1 $GOPATH/bin/raftdb -id node01 -haddr raft-cluster-host01:8091 -raddr raft-clu:
```

这时将在 Store.Open() 函数中，调用 BootstrapCluster() 函数将节点启动起来。


接着，其他节点会通过 -join 参数指定领导者节点的地址信息，并向领导者节点发送，包含当前节点配置信息的增加节点请求。启动命令就像下面的样子。

 复制代码

```
1 $GOPATH/bin/raftdb -id node02 -haddr raft-cluster-host02:8091 -raddr raft-clus:
```

当领导者节点接收到来自其他节点的增加节点请求后，将调用 `handleJoin()` 函数进行处理，并最终调用 `raft.AddVoter()` 函数，将新节点加入到集群中。

在这里，需要你注意的是，只有在向集群中添加新节点时，才需要使用 `-join` 参数。当节点加入集群后，就可以像下面这样，正常启动进程就可以了。

 复制代码

```
1 $GOPATH/bin/raftdb -id node02 -haddr raft-cluster-host02:8091 -raddr raft-clus
```

集群运行起来后，因为领导者是可能会变的，那么如何实现写操作，来保证写请求都在领导者节点上执行呢？

写操作

在 19 讲中，我们选择了方法 2 来实现写操作。也就是，当跟随者接收到写请求后，将拒绝处理该请求，并将领导者的地址信息转发给客户端。后续客户端就可以直接访问领导者（为了演示方便，我们以赋值操作为例）。

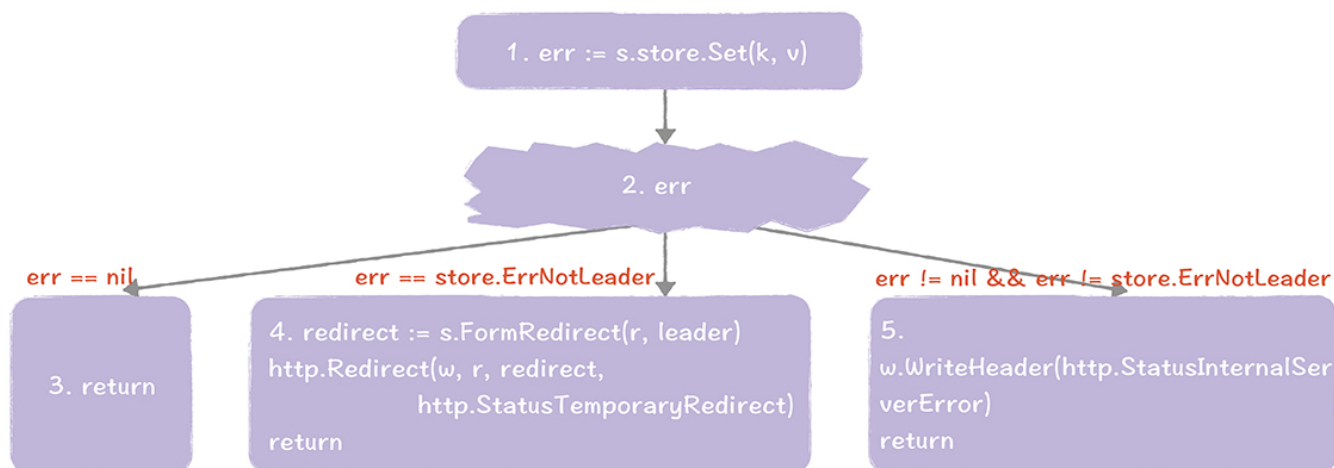


图5

我们来看一下具体的内容。

1. 调用 `Set()` 函数执行赋值操作。
2. 如果执行 `Set()` 函数成功，将执行步骤 3；如果执行 `Set()` 函数出错，且提示出错的原因是当前节点不是领导者，那这就说明了当前节点不是领导者，不能执行写操作，将执行

- 步骤 4；如果执行 Set() 函数出错，且提示出错的原因不是因为当前节点不是领导者，将执行步骤 5。
- 赋值操作执行成功，正常返回。
 - 节点将构造包含领导者地址信息的重定向响应，并返回给客户端。然后客户端直接访问领导者节点执行赋值操作。
 - 系统运行出错，返回错误信息给客户端。

需要你注意的是，赋值操作和删除操作属于写操作，必须在领导者节点上执行。而查询操作，只是查询内存中的数据，不涉及指令提交，可以在任何节点上执行。

而为了更好的利用 curl 客户端的 HTTP 重定向功能，我实现了 HTTP 307 重定向，这样，你在执行赋值操作时，就不需要关心访问节点是否是领导者节点了。比如，你可以使用下面的命令，访问节点 2（也就是 raft-cluster-host02，192.168.0.20）执行赋值操作。

复制代码

```
1 curl -XPOST raft-cluster-host02:8091/key -d '{"foo": "bar"}' -L
```

如果当前节点（也就是节点 2）不是领导者，它将返回包含领导者地址信息的 HTTP 307 重定向响应给 curl。这时，curl 根据响应信息，重新发起赋值操作请求，并直接访问领导者节点（也就是节点 1，192.168.0.10）。具体的过程，就像下面的 Wireshark 截图。

Time	Source	Destination	Protocol	Length	Info
1466 12.506462	192.168.0.30	192.168.0.20	HTTP	241	POST /key HTTP/1.1 (application/x-www-form-urlencoded)
1468 12.507209	192.168.0.20	192.168.0.30	HTTP	204	HTTP/1.1 307 Temporary Redirect
1476 12.512549	192.168.0.30	192.168.0.10	HTTP	241	POST /key HTTP/1.1 (application/x-www-form-urlencoded)
1483 12.516255	192.168.0.10	192.168.0.30	HTTP	141	HTTP/1.1 200 OK

图6

相比写请求必须在领导者节点上执行，虽然查询操作属于读操作，可以在任何节点上执行，但是如何实现却更加复杂，因为读操作的实现关乎着一致性的实现。那么，具体怎么实现呢？

读操作

我想说的是，我们可以实现 3 种一致性模型（也就是 stale、default、consistent），这样，用户就可以根据场景特点，按需选择相应的一致性级别，是不是很灵活呢？

具体的读操作的代码实现，就像下面的样子。

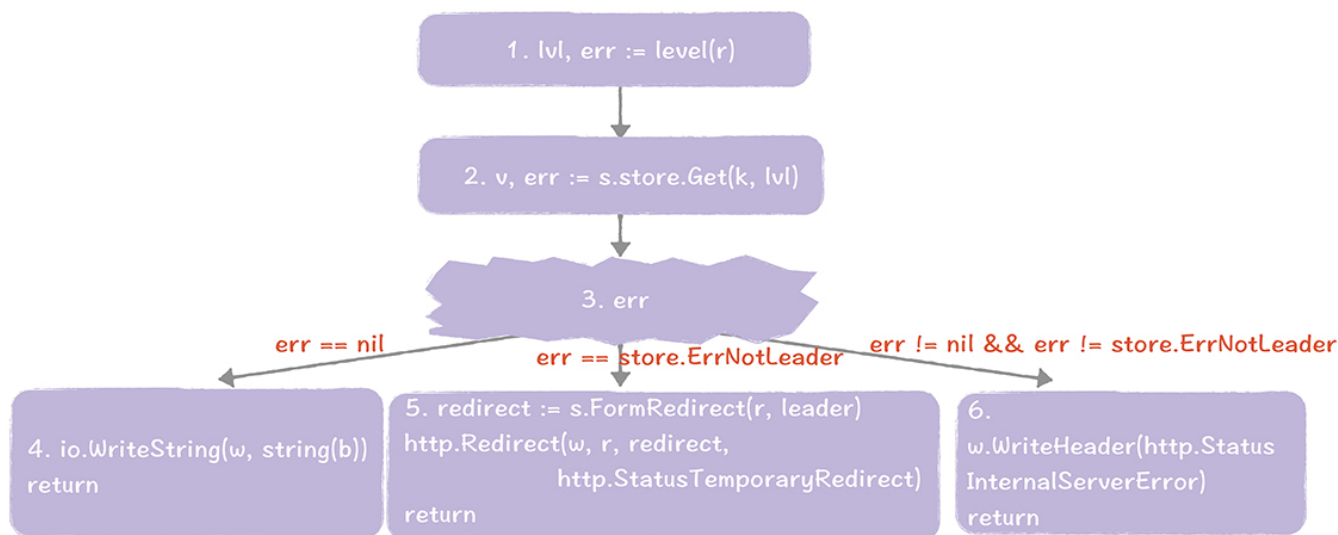


图7

我们走一遍这个过程。

1. 当接收到 HTTP GET 的查询请求时，系统会先调用 `level()` 函数，来获取当前请求的读一致性级别。
2. 调用 `Get()` 函数，查询指定 key 和读一致性级别对应的数据。
3. 如果执行 `Get()` 函数成功，将执行步骤 4；如果执行 `Get()` 函数出错，且提示出错的原因是当前节点不是领导者节点，那么这就说明了，在当前节点上执行查询操作不满足读一致性级别，必须要到领导者节点上执行查询操作，将执行步骤 5；如果执行 `Get()` 函数出错，且提示出错的原因不是因为当前节点不是领导者，将执行步骤 6。
4. 查询操作执行成功，返回查询到的值给客户端。
5. 节点将构造，包含领导者地址信息的重定向响应，并返回给客户端。然后客户端直接访问领导者节点查询数据。
6. 系统运行出错，返回错误信息给客户端。

在这里，为了更好地利用 curl 客户端的 HTTP 重定向功能，我同样实现了 HTTP 307 重定向（具体原理，前面已经介绍了，这里就不啰嗦了）。比如，你可以使用下面的命令，来实现一致性级别为 `consistent` 的查询操作，不需要关心访问节点（`raft-cluster-host02`）是否是领导者节点。

内容小结

本节课我主要带你了解了接入协议、KV 操作、分布式集群的实现，我希望你记住下面三个重点内容：

1. 我们可以借助 HTTP 请求类型，来实现相关的操作，比如，我们可以通过 HTTP GET 请求实现查询操作，通过 HTTP DELETE 请求实现删除操作。
2. 你可以通过 HTTP 307 重定向响应，来返回领导者的地址信息给客户端，需要你注意的是，curl 已支持 HTTP 307 重定向，使用起来很方便，所以推荐你优先考虑 curl，在日常中执行 KV 操作。
3. 在 Raft 中，我们可以通过 raft.VerifyLeader() 来确认当前领导者，是否仍是领导者。

在这里，我还想强调的一点，任何大系统都是由小系统和具体的技术组成的，比如能无限扩展和支撑海量服务的 QQ 后台，是由多个组件（协议接入组件、名字服务、存储组件等）组成的。而做技术最为重要的就是脚踏实地彻底吃透和掌握技术本质，小系统的关键是细节技术，大系统的关键是架构。所以，在课程结束后，我会根据你的反馈意见，再延伸性地讲解大系统（大型互联网后台）的架构设计技巧，和我之前支撑海量服务的经验。

这样一来，我希望能帮你从技术到代码、从代码到架构、从小系统到大系统，彻底掌握实战能力，跨过技术和实战的鸿沟。

虽然这个分布式 KV 系统比较简单，但它相对纯粹聚焦在技术，能帮助你很好的理解 Raft 算法、Hashicorp Raft 实现、分布式系统开发实战等。所以，我希望你不懂就问，有问题多留言，咱们一起讨论解决，不要留下盲区。

另外，我会持续维护和优化这个项目，并会针对大家共性的疑问，开发实现相关代码，从代码和理论 2 个角度，帮助你更透彻的理解技术。我希望你能在课下采用自己熟悉的编程语言，将这个系统重新实现一遍，在实战中，加深自己对技术的理解。如果条件允许，你可以将自己的分布式 KV 系统，以“配置中心”、“名字服务”等形式，在实际场景中落地和维护起来，不断加深自己对技术的理解。

课堂思考

我提到了通过 `-join` 参数，将新节点加入到集群中，那么，你不妨思考一下，如何实现代码移除一个节点呢？欢迎在留言区分享你的看法，与我一同讨论。

最后，感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

课程学习计划

关注极客时间服务号 每日学习签到

月领 25+ 极客币

【点击】保存图片，打开【微信】扫码>>>



© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 19 | 基于Raft的分布式KV系统开发实战（一）：如何设计架构？

下一篇 结束语 | 静下心来，享受技术的乐趣

精选留言 (6)

写留言



沉淀的梦想

2020-03-31

项目地址在哪里？自己实现 Raft 的话，正确性要如何测试呢？



Michael Tesla

2020-03-31

老师能不能推荐一些课外的相关资料呢？

展开 ∨



羽翼1982

2020-03-31

一路过来追老师的课，收获还是蛮多的；不过比起实现的细节，我还是想更多了解设计和架构上的知识，特别是这些理论在成熟的开源分布式系统上应用（Kafka，TiDB，ETCD，Cassandra等等），希望老师能够通过加餐的形式补充这些内容

展开 ∨



小晏子

2020-03-30

如果移除节点，那要考虑是移除主节点还是非主节点吧，如果是主节点那么需要重新发起选主流程，并将主节点数据同步到其他节点，如果是非主节点，那么要通知主节点该节点移除不需要在发送日志给它了。



wjh_all_in

2020-03-30

这里实现一致性，没有采用Quorum NWR，而是把所有读请求都转移到主节点，这在实际的生产系统会成为瓶颈吧？

展开 ∨



还有这种操作

2020-03-30

老师有没有代码示例，或者项目示例

展开 ∨

