

18 | Hashicorp Raft (二) : 如何以“集群节点”为中心使用 API?

2020-03-23 韩健

分布式协议与算法实战

[进入课程 >](#)



讲述：于航

时长 12:20 大小 11.30M



你好，我是韩健。

上一讲结束后，相信有的同学已经跃跃欲试，想把 Hashicorp Raft 使用起来了。不过，也有一些同学跟我反馈，说自己看到 Hashicorp Raft 的 [Godoc](#)，阅读完接口文档后，感觉有些不知所措，无从下手，Hashicorp Raft 支持了那么多的函数，自己却不知道如何将这些函数使用起来。




这似乎是一个共性的问题，在我看来，之所以出现这个问题，是因为文档里虽然提到了 API 的功能，但并没有提如何在实际场景中使用这些 API，每个 API 都是孤立的点，缺乏一些场景化的线将它们串联起来。

所以，为了帮你更好地理解 Hashicorp Raft 的 API 接口，在实践中将它们用起来，我以“集群节点”为核心，通过创建、增加、移除集群节点，查看集群节点状态这 4 个典型的场景，具体聊一聊在 Hashicorp Raft 中，通过哪些 API 接口能创建、增加、移除集群节点，查看集群节点状态。这样一来，我们会一步一步，循序渐进地彻底吃透 Hashicorp Raft 的 API 接口用法。

我们知道，开发实现一个 Raft 集群的时候，首先要做的第一个事情就是创建 Raft 节点，那么在 Hashicorp Raft 中如何创建节点呢？

如何创建 Raft 节点

在 Hashicorp Raft 中，你可以通过 `NewRaft()` 函数，来创建 Raft 节点。我强调一下，`NewRaft()` 是非常核心的函数，是 Raft 节点的抽象实现，`NewRaft()` 函数的原型是这样的：

 复制代码

```
1 func NewRaft(  
2     conf *Config,  
3     fsm FSM,  
4     logs LogStore,  
5     stable StableStore,  
6     snaps SnapshotStore,  
7     trans Transport) (*Raft, error)
```

你可以从这段代码中看到，`NewRaft()` 函数有这么几种类型的参数，它们分别是：

`Config`（节点的配置信息）；

`FSM`（有限状态机）；

`LogStore`（用来存储 Raft 的日志）；

`StableStore`（稳定存储，用来存储 Raft 集群的节点信息等）；

`SnapshotStore`（快照存储，用来存储节点的快照信息）；

`Transport`（Raft 节点间的通信通道）。

这 6 种类型的参数决定了 Raft 节点的配置、通讯、存储、状态机操作等核心信息，所以我带你详细了解一下，在这个过程中，你要注意是如何创建这些参数信息的。

Config 是节点的配置信息，可通过函数 DefaultConfig() 来创建默认配置信息，然后按需修改对应的配置项。一般情况下，使用默认配置项就可以了。不过，有时你可能还是需要根
据实际场景，调整配置项的，比如：

如果在生产环境中部署的时候，你可以将 LogLevel 从 DEBUG 调整为 WARM 或
ERROR；

如果部署环境中网络拥堵，你可以适当地调大 HeartbeatTimeout 的值，比如，从 1s
调整为 1.5s，避免频繁的领导者选举；

那么 FSM 又是什么呢？它是一个 interface 类型的数据结构，借助 Golang Interface 的泛
型编程能力，应用程序可以实现自己的 Apply(*Log)、Snapshot()、
Restore(io.ReadCloser) 3 个函数，分别实现将日志应用到本地状态机、生成快照和根据
快照恢复数据的功能。FSM 是日志处理的核心实现，原理比较复杂，不过不是咱们本节课
的重点，现在你只需要知道这 3 个函数就可以了。在 20 讲，我会结合实际代码具体讲解
的。


第三个参数 LogStore 存储的是 Raft 日志，你可以用 [raft-boltdb](#) 来实现底层存储，持久
化存储数据。在这里我想说的是，raft-boltdb 是 Hashicorp 团队专门为 Hashicorp Raft
持久化存储，而开发设计的，使用广泛，打磨充分。具体用法是这样的：

 复制代码

```
1 logStore, err := raftboltdb.NewBoltStore(filepath.Join(raftDir, "raft-log.db"))
```

NewBoltStore() 函数只支持一个参数，也就是文件路径。

第四个参数 StableStore 存储的是节点的关键状态信息，比如，当前任期编号、最新投票
时的任期编号等，同样，你也可以采用 raft-boltdb 来实现底层存储，持久化存储数据。

 复制代码

```
1 stableStore, err := raftboltdb.NewBoltStore(filepath.Join(raftDir, "raft-stabl
```

第五个参数 SnapshotStore 存储的是快照信息，也就是压缩后的日志数据。在 Hashicorp
Raft 中提供了 3 种快照存储方式，它们分别是：

DiscardSnapshotStore（不存储，忽略快照，相当于 /dev/null，一般来说用于测试）；

FileSnapshotStore（文件持久化存储）；

InmemSnapshotStore（内存存储，不持久化，重启程序后，数据会丢失）。

这 3 种方式，在生产环境中，建议你采用 FileSnapshotStore 实现快照，使用文件持久化存储，避免因程序重启，导致快照数据丢失。具体代码实现如下：

 复制代码

```
1 snapshots, err := raft.NewFileSnapshotStore(raftDir, retainSnapshotCount, os.S
```

NewFileSnapshotStore() 函数支持 3 个参数。也就是说，除了指定存储路径（raftDir），还要指定需要保留的快照副本的数量（retainSnapshotCount），以及日志输出的方式。**一般而言，将日志输出到标准错误 IO 就可以了。**

最后一个 Transport 指的是 Raft 集群内部节点之间的通信机制，节点之间需要通过这个通道来进行日志同步、领导者选举等等。Hashicorp Raft 支持两种方式：

一种是基于 TCP 协议的 TCPTransport，可以跨机器跨网络通信的；

另一种是基于内存的 InmemTransport，不走网络，在内存里面通过 Channel 来通信。

在生产环境中，我建议你使用 TCPTransport，使用 TCP 进行网络通讯，突破单机限制，提升集群的健壮性和容灾能力。具体代码实现如下：

 复制代码

```
1 addr, err := net.ResolveTCPAddr("tcp", raftBind)
2 transport, err := raft.NewTCPTransport(raftBind, addr, maxPool, timeout, os.Sti
```

NewTCPTransport() 函数支持 5 个参数，也就是，指定创建连接需要的信息。比如，要绑定的地址信息（raftBind、addr）、连接池的大小（maxPool）、超时时间（timeout），以及日志输出的方式，一般而言，将日志输出到标准错误 IO 就可以了。

以上就是这 6 个参数的详细内容了，既然我们已经了解了这些基础信息，那么如何使用 NewRaft() 函数呢？其实，你可以在代码中直接调用 NewRaft() 函数，创建 Raft 节点对象，就像下面的样子：

 复制代码

```
1 raft, err := raft.NewRaft(config, (*storeFSM)(s), logStore, stableStore, snapsl
```

接口清晰，使用方便，你可以亲手试一试。

现在，我们已经创建了 Raft 节点，打好了基础，但是我们要实现的是一个多节点的集群，所以，创建一个节点是不够的，另外，创建了节点后，你还需要让节点启动，当一个节点启动后，你还需要创建新的节点，并将它加入到集群中，那么具体怎么操作呢？

如何增加集群节点

集群最开始的时候，只有一个节点，我们让第一个节点通过 bootstrap 的方式启动，它启动后成为领导者：

 复制代码

```
1 raftNode.BootstrapCluster(configuration)
```

BootstrapCluster() 函数只支持一个参数，也就是 Raft 集群的配置信息，因为此时只有一个节点，所以配置信息为这个节点的地址信息。

后续的节点在启动的时候，可以通过向第一个节点发送加入集群的请求，然后加入到集群中。具体来说，先启动的节点（也就是第一个节点）收到请求后，获取对方的地址（指 Raft 集群内部通信的 TCP 地址），然后调用 AddVoter() 把新节点加入到集群就可以了。具体代码如下：

 复制代码

```
1 raftNode.AddVoter(id,  
2                 addr, prevIndex, timeout)
```


AddVoter() 函数支持 4 个参数，使用时，一般只需要设置服务器 ID 信息和地址信息，其他参数使用默认值 0，就可以了：

id（服务器 ID 信息）；

addr（地址信息）；

prevIndex（前一个集群配置的索引值，一般设置为 0，使用默认值）；

timeout（在完成集群配置的日志项添加前，最长等待多久，一般设置为 0，使用默认值）。

当然了，也可以通过 AddNonvoter()，将一个节点加入到集群中，但不赋予它投票权，让它只接受日志记录，这个函数平时用不到，你只需知道有这么函数，就可以了。

在这里，我想补充下，早期版本中的用于增加集群节点的函数，AddPeer() 函数，已废弃，不再推荐使用。

你看，在创建集群或者扩容时，我们尝试着增加了集群节点，但一旦出现不可恢复性的机器故障或机器裁撤时，我们就需要移除节点，进行节点替换，那么具体怎么做呢？

如何移除集群节点

我们可以通过 RemoveServer() 函数来移除节点，具体代码如下：

```
1 raftNode.RemoveServer(id, prevIndex, timeout)
```

 复制代码

RemoveServer() 函数支持 3 个参数，使用时，一般只需要设置服务器 ID 信息，其他参数使用默认值 0，就可以了：

id（服务器 ID 信息）；

prevIndex（前一个集群配置的索引值，一般设置为 0，使用默认值）；

timeout（在完成集群配置的日志项添加前，最长等待多久，一般设置为 0，使用默认值）。

我要强调一下，`RemoveServer()` 函数必须在领导者节点上运行，否则就会报错。这一点，很多同学在实现移除节点功能时会遇到，所以需要注意一下。

最后，我想补充下，早期版本中的用于移除集群节点的函数，`RemovePeer()` 函数也已经废弃了，不再推荐使用。

关于如何移除集群节点的代码实现，也比较简单易用，通过服务器 ID 信息，就可以将对应的节点移除了。除了增加和移除集群节点，在实际场景中，我们在运营分布式系统时，有时需要查看节点的状态。那么该如何查看节点状态呢？

如何查看集群节点状态


在分布式系统中，日常调试的时候，节点的状态信息是很重要的，比如在 Raft 分布式系统中，如果我们想抓包分析写请求，那么必须知道哪个节点是领导者节点，它的地址信息是多少，因为在 Raft 集群中，只有领导者能处理写请求。

那么在 Hashicorp Raft 中，如何查看节点状态信息呢？

我们可以通过 `Raft.Leader()` 函数，查看当前领导者的地址信息，也可以通过 `Raft.State()` 函数，查看当前节点的状态，是跟随者、候选人，还是领导者。不过你要注意，`Raft.State()` 函数返回的是 `RaftState` 格式的信息，也就是 32 位无符号整数，适合在代码中使用。**如果想在日志或命令行接口中查看节点状态信息，我建议你使用 `RaftState.String()` 函数**，通过它，你可以查看字符串格式的当前节点状态。

为了便于你理解，我举个例子。比如，你可以通过下面的代码，判断当前节点是否是领导者节点：

```
1 func isLeader() bool {
2     return raft.State() == raft.Leader
3 }
```

 复制代码

了解了节点状态，你就知道了当前集群节点之间的关系，以及功能和节点的对应关系，这样一来，你在遇到问题，需要调试跟踪时，就知道登录到哪台机器，去调试分析了。

内容小结

本节课我主要以“集群节点”为核心，带你了解了 Hashicorp Raft 的常用 API 接口，我希望你明确的重点如下：

1. 除了提到的 raft-boltdb 做作为 LogStore 和 StableStore，也可以调用 NewInmemStore() 创建内存型存储，在测试时比较方便，重新执行程序进行测试时，不需要手动清理数据存储。
2. 你还可以通过 NewInmemTransport() 函数，实现内存型通讯接口，在测试时比较方便，将集群通过内存进行通讯，运行在一台机器上。
3. 你可以通过 Raft.Stats() 函数，查看集群的内部统计信息，比如节点状态、任期编号、节点数等，这在调试或确认节点运行状况的时候很有用。

我以集群节点为核心，讲解了 Hashicorp Raft 常用的 API 接口，相信现在你已经掌握这些接口的用法了，对如何开发一个分布式系统，也有了一定的感觉。既然学习是为了使用，那么我们学完这些内容，也应该用起来才是，所以，为了帮你更好地掌握 Raft 分布式系统的开发实战技巧，我会用接下来两节课的时间，以分布式 KV 系统开发实战为例，带你了解 Raft 的开发实战技巧。

课堂思考

我提到了一些常用的 API 接口，比如创建 Raft 节点、增加集群节点、移除集群节点、查看集群节点状态等，你不妨思考一下，如何创建一个支持 InmemTransport 的 Raft 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。

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 17 | Hashicorp Raft (一)：如何跨过理论和代码之间的鸿沟？

精选留言 (3)

写留言



Stony.修行僧

2020-03-23

讲讲增删改查背后的真相

展开 ∨



1



HuaMax

2020-03-23

集群最开始的时候，只有一个节点，我们让第一个节点通过 bootstrap 的方式启动，它启动后成为领导者

请问老师，节点不是通过投票成为领导者吗？是不是所有节点都是bootstrap方式启动，然后再竞争成为领导者？

展开 ∨





艾瑞克小霸王

服务器ID是在什么地方设置的呢？node的config吗？

展开 ∨

