配置并使用 Raft 排序服务

受众： *Raft 排序节点管理员*

概述

有关排序概念的整体概述以及排序服务（包括 Raft）是如何工作的，请查看我们关于排序服务的概念文档。

要了解设置排序节点的过程（包括创建本地 MSP 和创建初始区块），请查看我们关于设置排序节点的文档。

配置

虽然必须将每个 Raft 节点添加到系统通道，但不需要将节点添加到每个应用程序通道。此外，您可以动态地从通道中删除和添加节点，而不会影响其他节点，下面的重新配置部分中描述了该过程。

Raft 节点使用 TLS pinning 技术互相识别，因此要假冒 Raft 节点，攻击者需要获取其 TLS 证书的私钥。所以如果没有有效的 TLS 配置，就无法运行 Raft 节点。

Raft 集群要在两个部分进行配置：

- **本地配置**：主要用于配置节点控制方面，如 TLS 通信、复制行为和文件存储。
- **通道配置**：定义相应通道的 Raft 集群成员，以及协议指定的参数，如心跳频率、领导节点超时等。

回想一下，每个通道都有自己的 Raft 协议实例运行。因此，必须在其所属的每个通道的配置中引用 Raft 节点，方法是将其服务器和客户端 TLS 证书（以 `PEM` 格式）添加到通道配置中。这确保了当其他节点接收到该节点的消息时，它们可以安全地确认发送消息的节点的身份。

下边是 `configtx.yaml` 中的一部分内容，展示了通道中三个 Raft 节点（也就是所谓的“共识者”）的配置：

```
        Consenters:
             - Host: raft0.example.com
               Port: 7050
               ClientTLSCert: path/to/ClientTLSCert0
               ServerTLSCert: path/to/ServerTLSCert0
             - Host: raft1.example.com
               Port: 7050
               ClientTLSCert: path/to/ClientTLSCert1
               ServerTLSCert: path/to/ServerTLSCert1
             - Host: raft2.example.com
               Port: 7050
               ClientTLSCert: path/to/ClientTLSCert2
               ServerTLSCert: path/to/ServerTLSCert2
```

注意：排序节点将被列为系统通道以及他们加入的任何应用程序通道的共识者。

当创建通道配置区块时，configtxgen 工具读取到 TLS 证书的路径，并将路径替换为相应的证书。

本地配置

`orderer.yaml` 中有两个部分和 Raft 排序节点的配置有关：

Cluster，决定了 TLS 通信的配置；和 **consensus**，决定了预写式日志和快照的存储位置。

Cluster 参数：

默认情况下，Raft 服务与面向客户端的服务器（用于发送交易或拉取区块）运行在同一个 gRPC 服务器上，但它可以配置为单独的具有独立端口的 gRPC 服务器。

这对于希望组织 CA 颁发的 TLS 证书仅用于群集节点的相互通信，而公共 TLS CA 颁发的 TLS 证书用于面向客户端的 API 的情况是非常有用的。

- `ClientCertificate`，`ClientPrivateKey`：客户端 TLS 证书和相关私钥的文件路径。
- `ListenPort`：集群监听的端口。如果为空，该端口就和排序节点通用端口（general.listenPort）一致 。
- `ListenAddress`：集群服务监听的地址。
- `ServerCertificate`，`ServerPrivateKey`：TLS 服务器证书密钥对，当集群服务运行在单独的 gRPC 服务器（不同端口）上时使用。
- `SendBufferSize`：调节出口缓冲区中的消息数。

注意：`ListenPort`、`ListenAddress`、`ServerCertificate` 和 `ServerPrivateKey` 必须同时设置或都不设置，如果没有设置它们，它们就会从 general TLS 部分继承，例如 `general.tls.{privateKey, certificate}`。

`general.cluster` 还有隐藏的配置参数，可用于进一步微调集群通信或复制机制：

- `DialTimeout`，`RPCTimeout`：指定创建连接和建立流的超时时间。
- `ReplicationBufferSize`：可以为从其他群集节点进行块复制的每个内存缓冲区分配的最大字节数。每个通道都有自己的内存缓冲区。默认为 `20971520`，即 `20 MB`。
- `PullTimeout`：排序节点等待接收区块的最长时间。默认为五秒。
- `ReplicationBackgroundRefreshInterval`：节点连续两次尝试复制其加入的通道，或无法复制通道的时间间隔。默认为五分钟。
- `TLSHandshakeTimeShift`：如果排序节点的 TLS 证书已过期且没有被及时替换（参见下文的 TLS 证书替换），那么节点之间的通信就无法建立，也不可能向排序服务传输新的交易。要从这种场景中恢复，可以让排序节点间的 TLS 握手认为时间向后移动了给定的时长，这个时长配置为 `TLSHandshakeTimeShift`。这只对为集群内部通信使用了独立 gRPC 服务器的排序节点有效（通过 `general.cluster.ListenPort` 和 `general.cluster.ListenAddress`）。
- `TLSHandshakeTimeShift`:如果ordering节点的TLS证书过期且没有按时更换（参见下面的TLS证书轮换），它们之间就不能建立交流，也不能向ordering服务发送新的交易。 为了从这种情况下恢复，可以考虑在ordering节点间进行TLS握手并使用`TLSHandshakeTimeShift`来配置一个回溯的时间。 为保证尽可能的不受外部影响，这个设置只在使用独立gRPC服务的ordering节点集群内有效。 如果您

的集群使用同一个gRPC服务来响应客户端和peer,您首先需要重新配置您的orderer，添加`general.cluster.ListenPort`，`general.cluster.ListenAddress`，`ServerCertificate`和`ServerPrivateKey`，然后重启orderer以让新配置生效。

Consensus 参数：

- `WALDir`：指定`etcd/raft`预写式日志的存储位置。每个通道都会有以通道 ID 为名的子目录。
- SnapDir`：指定`etcd/raft` 快照的存储位置。每个通道都会有以通道 ID 为名的子目录。

还有一个隐藏配置参数可以加入到`orderer.yaml`的共识部分：

- `EvictionSuspicion`：通道驱逐嫌疑节点的时间，这会触发节点从其他节点拉取区块并查看它是否被通道驱逐以此来确认它的嫌疑。如果确认了它的嫌疑（被检查的区块没有包含节点 TLS 证书），该节点将终止对通道的操作。当一个节点不知道任何被选举的领导节点或者不能被选举为该通道中的领导节点时，它就有可能是被通道逐出了。默认十分钟。

通道配置

除（已经讨论过的）共识者之外，Raft 通道配置中还有一个`Options`部分和协议有关。目前，在节点运行的时候不能动态地修改这些值。只能重新修改配置并重启节点。

只有`SnapshotIntervalSize`是例外，它可以在节点运行的时候修改。

注意：建议不要修改下面的这些配置，因为错误的配置可能会导致领导节点选举失败（比如，`TickInterval`和`ElectionTick`设置得过小）。不能选举领导节点的情况是无法解决的，因为领导节点是做变更所必需的。由于这些危险，我们建议一般不要调整这些参数。

- `TickInterval`：两次`Node.Tick`之间的时间间隔。
- `ElectionTick`：两次选举之间必须度过的`Node.Tick`次数。就是说，如果一个跟随节点在经过`ElectionTick`次的时间后仍没有从当前一轮的领导节点接收到任何消息，它就会变为候选节点，并且开始新一轮选举。`ElectionTick`必须大于`HeartbeatTick`。
- `ElectionTick`必须大于`HeartbeatTick`。
- `HeartbeatTick`：两次心跳之间必须度过的`Node.Tick`次数。就是说，领导节点每经过`HeartbeatTick`次的时间就会发送心跳消息来维持其领导地位。
- `MaxInflightBlocks`：限制乐观复制阶段在途新增区块的最大数量。
- `SnapshotIntervalSize`：定义每次创建的快照字节数。

重新配置

Raft 排序节点支持动态地（意思是，当通道正在使用时）添加和移除节点，只是一次只能添加或移除一个节点。在你尝试重新配置之前，请注意你的集群必须可以维护，并且能够获得共识。举个例子，如果你有 3 个节点，然后 2 个节点宕机了，你就不能重新配置你的集群来移除那些节点。同样地，如果你在一个有着三个节点的通道内有一个宕机的节点，那么不应该尝试替换证书，因为这会造成二次错误。作为一个准则，除非所有共识者都在线且健康，你永远都不应该尝试对 Raft 共识者做配置变更，如添加或删除共识者，或替换共识者的证书等。

如果你决定修改这些参数，我们建议只在维护周期内进行尝试。修改配置的问题绝大多数都发生在只有少量节点的集群中且有一个节点宕机之时。比如，如果你有三个节点的共识者，其中有一个宕机，这意味着你只有两个节点存活。如果你在这个状态下将集群扩展到 4 个节点，你仍然只有 2 个节点存活，这无法达到法定人数。第四个节点不能上线，因为节点只能加入到运作中的集群（除非集群总大小是 1 或 2）。

因此，在扩展一个（只有两个节点存活的）三节点集群为四节点时，你完全会被卡住，直到原先离线的节点恢复。

添加一个新节点到 Raft 集群需要通过以下步骤完成：

1. 通过一个通道配置更新交易**将新节点的 TLS 证书添加到通道中**。注意：新节点在加入一个或更多应用通道前，必须先加入到系统通道。
2. 从一个排序节点中**获取最新的系统通道配置区块**，这是系统通道的一部分。
3. 通过验证配置区块是否包含（即将）加入的节点证书来**确保此节点是系统通道的一部分**。
4. 使用在 `General.BootstrapFile` 配置参数中指定的配置区块路径**启动新的 Raft 节点**。
5. **等待 Raft 节点**从已有节点**复制**其证书所加入的通道中的**区块**。在这一步完成后，节点开始服务于通道。
6. 将新增的 Raft 节点端点**加入**所有通道的配置。

可以将已经运行（且已经加入某些通道）的节点在运行时加入到通道中。要做到这点，只需添加该节点的证书到通道的通道配置中。节点会自动检测其加入到新的通道（默认值是 5 分钟，但如果你想让节点更快检测新通道，可以重启节点），然后从通道中的 orderer 拉取通道区块，最后为该链启动 Raft 实例。

在成功完成以上步骤后，就能更新通道配置以纳入新 Raft orderer 的端点。

从一个 Raft 集群中移除一个节点需要通过以下步骤完成：

1. 从所有通道的通道配置中移除其端点，包括由 orderer 管理员控制的系统通道。
2. 从所有通道的通道配置中移除该节点（由证书识别）的记录。再次强调，这也包括系统通道。
3. 关闭节点。

从一个指定的通道中移除节点，但仍维持其服务于其他通道需要通过以下步骤完成：

1. 从该通道的通道配置中移除其端点。
2. 从该通道的配置中移除该节点（由证书识别）的记录。
3. 第二个阶段会导致：
   - 通道中剩余的 orderer 节点在被移除的通道上下文中停止与被移除的 orderer 节点通信。它们仍会在其他通道上通信。
   - 从通道中移出的节点会立即或在度过 `EvictionSuspicion` 的时间（默认 10 分钟）之后自动检测它的移除，然后停止其 Raft 实例。

Orderer 节点的 TLS 证书替换

所有 TLS 证书都有一个由颁发者决定的过期日期。这些过期日期从颁发日期算起，长至 10 年，短则数月，所以请与颁发者确认。在过期日期到来前，你需要更换节点的证书以及节点所加入的每一个通道，包括系统通道。

对于每一个节点加入的通道：

1. 使用新证书更新通道配置。
2. 替换节点文件系统上的证书。
3. 重启节点。

因为一个节点只能有一对 TLS 证书私钥，因此在更新过程中，节点不能在新证书加入前服务通道，从而降低了容错的能力。由于这个原因，**证书替换的过程一旦开始，就应该尽快完成**。

如果因为某些原因，TLS 证书的替换已经开始但不能在所有通道上完成，建议回退到原先的 TLS 证书，并在以后再尝试替换。

证书过期的相关身份验证

如果一个客户端的标识含有过期日期（如基于X509证书的标识）,当它发送给orderer一笔交易时，orderer会首先检查标识是否过期，如果是，拒绝这笔交易的提交。

然而，你可以通过打开 `General.Authentication.NoExpirationChecks` 配置来让orderer忽略过期日期，相关配置在 `orderer.yaml` 中。

这仅限于在极端情况下使用，当一个管理员的证书过期，就会导致没有办法发送新的配置文件来更新管理员证书，因为配置文件的签名由当前管理员证书进行，但它因为已经过期而会被拒绝。 更新通道之后建议将配置文件改回默认配置，即强制标识进行过期检测。

Metrics

关于维护服务的描述和如何配置启动，参见我们关于维护服务的文档。

关于维护服务收集的指标列表，参考我们的指标参考文档。

当你为特定的用例为指标排列优先级而做大量配置工作的时候，有两个特别的指标可能是你想要监控的：

- `consensus_etcdraft_is_leader`：识别集群中的哪个节点是当前的领导者。如果没有设置任何节点，你就是丢失了法定人数。
- `consensus_etcdraft_data_persist_duration`：指示写入 Raft 集群持久化预写式日志所持续的时间。从协议安全性考虑，消息必须在被共享给共识者集合之前适当地调用 `fsync` 来持续持久化。如果这个值开始攀升，这个节点可能无法参与到共识中（这会导致此节点或者网络的服务中断）。

排错

- 你越是给节点更大的压力，就越可能需要修改相关参数。对于任何系统来说，电脑或机器，压力会导致性能的拖累。正如我们在概念文档中所述，Raft 中的领导者选举会发生在跟随者节点在一段时间内未接收到来自领导者的“心跳”消息或带有数据的“附加”消息时。因为 Raft 节点在通道间共享相同的通信层（这不代表他们共享数据——他们不共享！），如果一个 Raft 节点在许多通道中都属于共识者集合的部分，你会想增加它触发选举所需的时间长度来避免无意的领导者选举。