

分布式系统作业5

18340013 陈琮昊

1.简述下列软件采用的选举算法: Zookeeper, Redis, MongoDB, Cassandra.

Zookeeper:

当 zookeeper 集群中的一台服务器出现以下两种情况之一时, 需要进入Leader selection:

- 1.服务器初始化启动。
- 2.服务器运行期间无法和Leader保持连接。

对于第一种情况, 选举过程如下:

每个Server发出一个投票。每次投票会包含所推举的服务器的 myid 和 zxid, 使用(myid, zxid)来表示。各自将这个投票发给集群中其他机器。

接受来自各个服务器的投票。集群的每个服务器收到投票后, 判断该票是否有效。

处理投票。针对每一个投票, 比较规则如下:

- 1.优先检查ZXID。zxid 比较大的服务器优先作为Leader。
- 2.如果ZXID相同, 那么就比较myid。myid 较大的服务器作为Leader服务器。

统计投票。每次投票后, 服务器都会统计投票信息, 判断是否已经有过半机器接受到相同的投票信息, 如果已经过半, 则选出了Leader。

改变服务器状态。一旦确定了Leader, 每个服务器就会更新自己的状态, 如果是Follower, 那么就变更为FOLLOWING, 如果是Leader, 就变更为LEADING。

对于第二种情况, 选举过程如下:

不是Leader的服务器宕机或者有新成员加入, 此时依旧正常运行。但是一旦Leader服务器挂了, 那么整个集群将暂停对外服务, 进入新一轮Leader选举。首先剩下的服务器将改变服务器状态为LOOKING, 后续选举过程同上。

Redis:

采用哨兵机制进行选举, 其中每个集群有 master 和 slave, 为上下级的关系。

当 slave 发现自己的 master 变为FAIL状态时, 便尝试发起选举, 以期成为新的 master。由于挂掉的 master 可能会有多个 slave, 从而存在多个 slave 竞争成为 master 节点的过程, 其过程如下:

1. slave 发现自己的 master 变为FAIL
- 2.将自己记录的集群 currentEpoch (选举轮次标记) 加1, 并广播信息给集群中其他节点
- 3.其他节点收到该信息, 只有 master 响应, 判断请求者的合法性, 并发送结果
- 4.尝试选举的 slave 收集 master 返回的结果, 收到超过半数 master 的统一后变成新的 master
- 5.广播消息通知其他集群节点。

MongoDB :

MongoDB 节点之间维护**心跳检查**，主节点选举由心跳触发。

心跳检查:

MongoDB 复制集成员会向自己之外的所有成员发送心跳并处理响应信息，因此每个节点都维护着从该节点看其他所有节点的状态信息。节点根据自己的集群状态信息判断是否需要更换新的Primary。

在实现的时候主要由两个异步的过程分别处理心跳响应和超时，抛开复杂的条件检查，核心逻辑主要包括：

- 1.Secondary节点权重比Primary节点高时，发起替换选举；
- 2.Secondary节点发现集群中没有Primary时，发起选举；
- 3.Primary节点不能访问到大部分(Majority)成员时主动降级；

降级操作会断开链接，终止用户请求等。

选举发起:

发起选举的节点需要首先做一些条件判断，比如节点位于备选节点列表中。然后将自己标记为“选举过程中”，并发起投票请求。

投票:

投票发起者向集群成员发起Elect请求，成员在收到请求后经过一系列检查，如果通过检查则为发起者投一票。一轮选举中每个成员最多投一票。在 PV0 中采用30秒的“选举锁”避免为其他发起者重复投票，当然这导致了如果新选举的Primary挂掉，可能30秒内不会有新的Primary选举产生；在 PV1 中通过引入一个单调递增的变量Term来解决重复投票的问题。

如果投票发起者获得超过半数的投票，则选举通过成为Primary节点，否则重新发起投票。

Cassandra :

关于 Cassandra 分布式数据库系统如何进行选举并没有找到太多资料，能找到的资料里只是说使用 Zookeeper 来进行选举Leader。

2.请从以下软件中选择一种，编译运行，观察是否可以实现可靠多播，并撰写报告：

<https://github.com/baessler/pmul>

<https://github.com/glycerine/nack-oriented-reliable-multicast>

<https://github.com/GcherkosH/Reliable-and-ordered-multicast-protocol>

<https://github.com/daeyun/reliable-multicast-chat>





我选择的是最后一个：

<https://github.com/daeyun/reliable-multicast-chat>


其文件目录结构如下：

📁 .idea	2020/12/19 20:50	文件夹
📁 bin	2020/12/19 19:59	文件夹
📁 reliable_multicast_chat	2020/12/19 20:44	文件夹

其中 reliable_multicast_chat 文件夹内是Python的代码实现，包含以下几个文件：

 <code>_init_.py</code>	2014/4/4 10:35	JetBrains PyCharm	0 KB
 <code>chat_process.py</code>	2020/12/19 19:55	JetBrains PyCharm	9 KB
 <code>config.py</code>	2020/12/19 20:44	JetBrains PyCharm	1 KB
 <code>main.py</code>	2014/4/4 10:35	JetBrains PyCharm	1 KB

`bin` 文件夹内是可执行文件：

 <code>reliable_multicast_chat</code>	2014/4/4 10:35	文件	1 KB
--	----------------	----	------

其中 `config.py` 为配置文件，可以配置IP地址、端口号，还有多播种类：

```
ordering can be either 'casual' or 'total'
"""

config = {
    'hosts': [
        ('localhost', 49664),
        ('localhost', 49665),
        ('localhost', 49666),
        ('localhost', 49667),
        ('localhost', 49671),
        ('localhost', 49673),
    ],
    'ordering': 'casual',
}
```

可以看到，顺序有两种选择，一种是 `casual`，另一种是 `total`。在 `casual` 的情况下则是因果有序多播；而在 `total` 的情况下则是全序多播。此时由PID=0的进程来管理：当PID=0的进程接收到一条消息时，它增加一个内部计数器，并将该编号指定为消息的订单号，然后将该编号多路传输给所有其他进程。当一个进程多播一条消息时，其他进程将把该消息保存在一个缓冲区中，直到它们从PID=0的进程接收到指示该消息顺序的编号时才释放。接下来通过实验来观察这一现象：

Case1: casual

在终端下执行如下指令：

```
./reliable_multicast_chat [process ID] [delay time (in seconds)] [drop rate
(0<=P<1)]
```

指令带有三个参数，分别是PID、`delay time`和`drop rate`。

打开三个终端，分别执行如下指令：

```
./reliable_multicast_chat 1 0.02 0.9
./reliable_multicast_chat 2 0.02 0.9
./reliable_multicast_chat 3 0.02 0.9
```

然后分别输入消息，可以看到消息的确以多播的形式发送给了各个进程：

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master$ cd bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 1 0.02 0.9
2 says:  ack
ack
2 says:  666
666
3 says:  123
3 says:  456
2 says:  245
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master$ cd bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 2 0.02 0.9
ack
1 says:  ack
666
1 says:  666
3 says:  123
3 says:  456
245
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 3 0.02 0.9
123
2 says:  ack
1 says:  ack
2 says:  666
1 says:  666
456
2 says:  245
```

Case2: total

接下来修改顺序为 `total`：

```
config = {
    'hosts': [
        ('localhost', 49664),
        ('localhost', 49665),
        ('localhost', 49666),
        ('localhost', 49667),
        ('localhost', 49671),
        ('localhost', 49673),
    ],
    'ordering': 'total',
}
```

同上进行多播，效果如下：

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 1 0.01 0.8
dgcsvjh
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 2 0.01 0.8
dvghjbg
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 3 0.01 0.8
cgfvhb
```

可以看到3个进程都没有收到任何消息，即此时消息在缓冲区中；此时加入PID=0的进程：

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 0 0.01 0.8
2 says:  dvghjbgb
1 says:  dgcsvjh
3 says:  cgfvhb
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 1 0.01 0.8
dgcsvjh
2 says:  dvghjbgb
1 says:  dgcsvjh
3 says:  cgfvhb
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 2 0.01 0.8
dvghjbgb
2 says:  dvghjbgb
1 says:  dgcsvjh
3 says:  cgfvhb
```

```
chench@LAPTOP-TOEITVUA:~$ cd reliable-multicast-chat-master/bin
chench@LAPTOP-TOEITVUA:~/reliable-multicast-chat-master/bin$ ./reliable_multicast_chat 3 0.01 0.8
cgfvhb
2 says:  dvghjbgb
1 says:  dgcsvjh
3 says:  cgfvhb
```

可以看到加入PID=0的进程后，消息被允许从缓冲区内释放出来，开始在进程间多播，这符合我们的预期。