链滴

pbft 总结

作者：deshanxiao

原文链接：https://ld246.com/article/1512004435976

来源网站：链滴

pbft的启动及配置

当节点是vp时，server方法 -> GetEngine设置engine.consenter为obcBatch，obcBatch有一个成externalEventReceiver实现了RecvMsg

方法。

id:由getValidatorID可得 （vpX..id就为X）

当收到消息时（和noops消息类型一样），调用externalEventReceiver.RecvMsg:

```
// RecvMsg is called by the stack when a new message is received
func (eer *externalEventReceiver) RecvMsg(ocMsg *pb.Message, senderHandle *pb.PeerID) er
or {
    eer.manager.Queue() <- batchMessageEvent{
        msg:    ocMsg,
        sender: senderHandle,
    }
    return nil
}
```

由于externalEventReceiver设置的Receiver是obcbatch,所以外面所有事件交由obcbatch.ProcessEvnt处理，obcbatch.ProcessEvent转

发给obcbatch.processMessage，然后调用op.submitToLeader(req)

submitToLeader存下该请求（storeOutstanding(req)），并广播（防止自己处于错误的view），后如果自己是主节点的话，执行leaderProcReq，它的策略主要是

在请求数达到op.batchSize的时候才返回events.Event，否则返回空。当返回不为空时，processMessage会返回一个非空事件，然后传递到ProcessEvent

进入default分支，交给pbft-core的ProcessEvent模块处理(RequestBatch事件)

pbft协议过程

主节点收到RequestBatch过程

1. 存入<hash(req), req>到reqBatchStore，outstandingReqBatches,并持久化(instance.consumerStoreState(k,v))

2. 打开一个与hash(req)相关的计时器

3. 判断当前节点是不是主节点，如果是sendPrePrepare

主节点发送PrePrepare过程

1. 先得到序列号n(instance.seqNo + 1)

2. 如果发现已经收到其他摘要相同view相同而序列号n不同的，则返回

3. 当n不在watermarks间，或n > h + L/2 返回

4. 当n大于viewChangeSeqNo,因为将要view-change主节点，返回

```
func (instance *pbftCore) updateViewChangeSeqNo() {
```

```
    if instance.viewChangePeriod <= 0 {
        return
    }
    // Ensure the view change always occurs at a checkpoint boundary
    instance.viewChangeSeqNo = instance.seqNo + instance.viewChangePeriod*instance.K - in
stance.seqNo%instance.K
    logger.Debugf("Replica %d updating view change sequence number to %d", instance.id, ins
ance.viewChangeSeqNo)
}
```

如果viewchangeperiod配置为0则不会改变viewChangeSeqNo

5. 以上条件都不满足。构造pre-prepare消息

```
    preprep := &PrePrepare{
        View:           instance.view,     // 当前view
        SequenceNumber: n,                 // 消息号
        BatchDigest:    digest,
        RequestBatch:   reqBatch,
        ReplicaId:      instance.id,       //主节点id
    }
```

将 prePrepare, digest存入cert里，并持久化qset(?)

6. 广播pre-Prepare消息

7. maybeSendCommit （目前不是特别明白）

recvPrePrepare处理流程

1. 如果正在view-change,忽略这个Pre-Prepare消息

2. 如果收到的Pre-Prepare发送id不是当前view的主节点，忽略这个Pre-Prepare消息（view一定，节点一定 pbftCore.primary函数）

3. 当收到的view与pbft-core的view不一致，或n不在watermarks间的话，丢弃这个消息

4. 当n大于viewChangeSeqNo,发送viewChange消息，返回

5. 当收到相同序列号的消息时，如果消息体不同，进行view-change

否则，将收到的prePrepare, digest存入cert里

6. 当reqBatchStore不存在当前PrePrepare消息的摘要时，首先计算摘要，如果摘要与收到的摘要一，将摘要记录

到reqBatchStore与outstandingReqBatches中，并持久化该reqBatch

7.当前节点不是主节点并且这个Pre-Prepare消息之前没有发送Prepare消息的话，构造prepare消息：

```
prep := &Prepare{
    View:           preprep.View,
    SequenceNumber: preprep.SequenceNumber,
    BatchDigest:    preprep.BatchDigest,
    ReplicaId:      instance.id,
}
```

自己调用recvPrepare（相当于自己收到了Prepare消息）

然后持久化qset(?)
调用instance.innerBroadcast(&Message{Payload: &Message_Prepare{Prepare: prep}})广播消息

节点收到prepare消息

1. 如果收到的是主节点的消息，丢弃（prepare不可能由主节点发送）
2. 当收到的view与pbft-core的view不一致，或n不在watermarks间的话，丢弃这个消息
3. 获取cert,如果收到同一节点同一view的同一序号消息，则退出
4. 将prepare放入cert,并持久化pset
5. maybeSendCommit(目前不是特别明白)

节点收到commit消息 recvCommit

1. 当收到的view与pbft-core的view不一致，或n不在watermarks间的话，丢弃这个消息
2. 获取cert,如果收到同一节点同一view的同一序号存在该commit消息，丢弃这个消息
3. 添加commit到cert里
4. 看该msg是否满足commited函数，若满足，将先前放入的outstandingReqBatches删除
执行未完成的batch,执行完后，如果序列号等于viewChangeSeqNo，进行view-change

执行未完成的batch executeOutstanding()

大概思路是每来一个commit，执行所有没执行完的，当中任意一个执行成功则返回，否则继续执行

1. 如果instance.currentExec当前正在执行，退出
2. 执行所有没执行完的,执行成功一次就返回

```
for idx := range instance.certStore {
    if instance.executeOne(idx) {
        break
    }
}
```

执行某一个commit executeOne(n)

1. 如果n不等于lastExec+1，返回false(为了保证按序执行)
这里的lastExec初始化由getLastSeqNo获得
2. 如果设置了skipInProgress，则返回false
skipInProgress： 在发现了一个落后的情况到我们选择一个新的起点间设置
3. 如果该req不满足commited函数，则返回false
4. 如果是空请求instance.execDoneSync()
否则instance.consumer.execute(idx.n, reqBatch)进入执行状态

obcbatch execute

1. 把即将执行的交易放入txs
2. 存入最近执行交易的节点和时间戳
3. 调用op.stack.Execute(meta, txs)　（后台执行，执行完会收到executedEvent）

广播消息：

消息	函数
Message_PrePrepare	sendPrePrepare
Message_Prepare	recvPrePrepare
Message_Commit	maybeSendCommit
Message_Checkpoint	Checkpoint
Message_FetchRequestBatches	fetchRequestBat
Message_ViewChange	sendViewChange
Message_NewView	sendNewView
Message_Prepare	processNewView2