PaxosStore 源码分析「七、其他细节」

2020.04.26　SF-Zhou

作为本系列的最后一篇博文，本篇会争取把之前挖的坑一一填上，包括读取、CatchUp 和 Recovery 的流程，以及 WaitingMsg 的使用。

1. 读取流程

首先可以看下 example/ServiceImpl.cpp 中对读取的处理：

```
int clsServiceImpl::SelectCard(grpc::ServerContext &oContext, const example::CardRequest
                               example::CardResponse &oResponse) {
  int iRet = BatchFunc(example::OperCode::eSelectCard, oRequest, oResponse);
  if (iRet != 0)
    return iRet;

  clsDBImpl *poDBEngine =
      dynamic_cast<clsDBImpl *>(Certain::clsCertainWrapper::GetInstance()->GetDBEngine())
  dbtype::DB *poDB = poDBEngine->GetDB();
  clsTemporaryTable oTable(poDB);

  std::string strKey;
  EncodeInfoKey(strKey, oRequest.entity_id(), oRequest.card_id());
  std::string strValue;
  iRet = oTable.Get(strKey, strValue);
  if (iRet == Certain::eRetCodeNotFound) {
    return example::StatusCode::eCardNotExist;
  }

  if (iRet == 0 && !oResponse.mutable_card_info()->ParseFromString(strValue)) {
    return Certain::eRetCodeParseProtoErr;
  }

  return iRet;
}
```

首先会跑一遍 BatchFunc，然后正常地读取一遍本地数据库。换句话说如果 BatchFunc 成功返回了，那么本地数据也是最新的。如果仔细看 BatchFunc 的实现，会发现纯读的请求对应的 write_batch 是空的，其他的和正常写入没有区别。来看下对于纯读的请求 PaxosStore 是如何处理的：

```
// src/CertainWrapper.cpp
int clsCertainWrapper::RunPaxos(uint64_t iEntityID, uint64_t iEntry, uint16_t hSubCmdID,
                                const vector<uint64_t> &vecWBUUID, const string &strWrite
  ...
  // WriteBatch 为空时为 ReadOnly 模式
  poWB->SetReadOnly(strWriteBatch.size() == 0);
  ...
}

// src/EntryState.cpp
void clsEntryStateMachine::SetCheckedEmpty(uint32_t iAcceptorID) {
  // 设定 CheckedEmpty 状态
  m_atRecord[iAcceptorID].bCheckedEmpty = true;
}
bool clsEntryStateMachine::IsReadOK() {
  // 检查是否多数派为空
  uint32_t iCount = 0;
  for (uint32_t i = 0; i < s_iAcceptorNum; ++i) {
    if (m_atRecord[i].bCheckedEmpty && m_atRecord[i].iPromisedNum == 0) {
      iCount++;
    }
  }
  CertainLogDebug("iCount %u", iCount);
  return iCount >= s_iMajorityNum;
}

// src/Command.cpp
int clsPaxosCmd::ParseFromArray(const char *pcBuffer, uint32_t iLen) {
  CertainPB::PaxosCmd oPaxosCmd;
  if (!oPaxosCmd.ParseFromArray(pcBuffer, iLen)) {
    CertainLogError("ParseFromArray fail");
    return -1;
  }

  SetFromHeader(oPaxosCmd.mutable_header());
  m_iSrcAcceptorID = oPaxosCmd.src_acceptor_id();
  m_iDestAcceptorID = oPaxosCmd.dest_acceptor_id();
  ConvertFromPB(m_tSrcRecord, &oPaxosCmd.src_record());
  ConvertFromPB(m_tDestRecord, &oPaxosCmd.dest_record());

  // 如果是 Check Empty 的 Cmd
  if (oPaxosCmd.check_empty()) {
    Assert(IsEntryRecordEmpty(m_tSrcRecord));
    Assert(IsEntryRecordEmpty(m_tDestRecord));
    // 目标 PromisedNum 设为 -1，-1 < 0
    m_tDestRecord.iPromisedNum = INVALID_PROPOSAL_NUM;
  }

  m_bQuickRsp = oPaxosCmd.quick_rsp();
  m_iMaxChosenEntry = oPaxosCmd.max_chosen_entry();
  return 0;
}

// src/EntityWorker.cpp
int clsEntityWorker::DoWithClientCmd(clsClientCmd *poCmd) {
  ...
  if (poCmd->IsReadOnly()) {
    if (poMachine->IsLocalEmpty()) {
      poMachine->ResetAllCheckedEmpty();
      poMachine->SetCheckedEmpty(iLocalAcceptorID);
      BroadcastToRemote(ptInfo, NULL, poCmd);  // 此时的 Record 为初始化状态
      m_poEntryMng->AddTimeout(ptInfo, m_poConf->GetCmdTimeoutMS());
      OSS::ReportCheckEmpty();
      return eRetCodePtrReuse;
    } else {
      OSS::ReportPaxosForRead();
    }
  } else {
    OSS::ReportPaxosForWrite();
  }
  ...
}
```

```
// 接收回复的 PaxosCmd 并更新 Record
int clsEntityWorker::UpdateRecord(clsPaxosCmd *poPaxosCmd) {
  ...
  // 判断是否存在远端 Record 更新，上面的 -1 会让这里变成 true
  bool bRemoteUpdated = IsEntryRecordUpdated(tDestRecord, tNewRecord);
  // 判断是否存在本地 Record 更新，都是初始化的状态，仍然 false
  bool bLocalUpdated = IsEntryRecordUpdated(tOldRecord, tNewRecord);

  if (bLocalUpdated) {
    ...
  } else {
    // 通知 DB 落盘
    CheckIfNeedNotifyDB(ptEntityInfo);
    clsAutoDelete<clsPaxosCmd> oAuto(po);

    if (ptEntityInfo->poClientCmd != NULL && ptEntityInfo->poClientCmd->IsReadOnly()) {
      // 如果发起方的命令是 ReadOnly 的
      if (ptEntityInfo->poClientCmd->GetUUID() == poPaxosCmd->GetUUID() &&
          poMachine->IsLocalEmpty()) {
        // 标记回复节点 Checked 成功
        poMachine->SetCheckedEmpty(poPaxosCmd->GetSrcAcceptorID());
      }

      // 多数派为空，返回成功
      if (poMachine->IsReadOK()) {
        InvalidClientCmd(ptEntityInfo, eRetCodeOK);
        return 0;
      } else if (!poMachine->IsLocalEmpty()) {
        InvalidClientCmd(ptEntityInfo, eRetCodeReadFailed);
      }
    }

    // 同步状态
    ptInfo->bRemoteUpdated = bRemoteUpdated;
    SyncEntryRecord(ptInfo, po->GetDestAcceptorID(), po->GetUUID());
  }

  return 0;
}
```

逻辑非常隐含，初始化 Record 后，标记 CheckedEmpty 直接发送出去；接收到的节点反序列化时将 CheckedEmpty 的 Cmd 的 src.PromisedNum 设为 -1，使得 bRemoteUpdated 始终成立，始终回包；bLocalUpdated 为 false 所以不会落盘；最后发起请求的节点获得多数派为空的回复后，确定本地和全局进度一致。对应的失败处理留给读者自己分析。

严格来说这样写代码并不好，逻辑隐藏地太深了。当然以此实现的不落盘读取还是很精妙的。

2. CatchUp 流程

CatchUp 包括 EntityCatchUp 和 CheckForCatchUp，前者负责将 CommittedEntry 追赶到 MaxChosenEntry，后者负责将 MaxContChosenEntry 追赶到 MaxChosenEntry。在之前的「五、实现细节」有对应的函数细节分析，这里看一下实际运行时的状态。考虑以下情况：

1. A 机网络中断;
2. 客户端向 A 机发出请求，超时，转而请求 B;
3. B / C 达成了一些共识;
4. A 机网络恢复;
5. 客户端向 A 机发出读取请求，A 机首先执行 EntityCatchUp 提交 PLog 到 DB，而后发起协议，由于 B / C 进度更新，返回 RemoteNewer;
6. A 机执行 CheckForCatchUp，向 B / C 机请求缺失的 PLog。

```
int clsEntityWorker::DoWithPaxosCmd(clsPaxosCmd *poPaxosCmd) {
  ...
  // B / C 检查发现 A 机发起协议的 Entry 落后，返回当前 MaxChosenEntry 和 RemoteNewer
  if (ptEntityInfo->iMaxChosenEntry >= iEntry && poPaxosCmd->IsQuickRsp()) {
    clsPaxosCmd *po = new clsPaxosCmd(iLocalAcceptorID, iEntityID, iEntry, NULL, NULL);
    po->SetDestAcceptorID(iAcceptorID);
    po->SetResult(eRetCodeRemoteNewer);
    po->SetUUID(poPaxosCmd->GetUUID());
    po->SetMaxChosenEntry(uint64_t(ptEntityInfo->iMaxChosenEntry));

    OSS::ReportFastFail();
    m_poIOWorkerRouter->GoAndDeleteIfFailed(po);

    CheckForCatchUp(ptEntityInfo, iAcceptorID, 0);

    CertainLogError("E(%lu %lu) QuickRsp iMaxChosenEntry %lu", iEntityID, iEntry,
                    ptEntityInfo->iMaxChosenEntry);
    return 0;
  }
  // A 机接收到 B / C 的回复，首先通知 ClientCmd 失败，而后执行 CheckForCatchUp
  if (poPaxosCmd->GetResult() == eRetCodeRemoteNewer) {
    if (ptEntityInfo->poClientCmd != NULL) {
      if (ptEntityInfo->poClientCmd->GetUUID() == poPaxosCmd->GetUUID() &&
          ptEntityInfo->poClientCmd->GetEntry() == poPaxosCmd->GetEntry()) {
        assert(ptEntityInfo->poClientCmd->IsReadOnly());
        InvalidClientCmd(ptEntityInfo, eRetCodeRemoteNewer);
      }
    }

    if (ptInfo != NULL && !ptInfo->bUncertain) {
      if (ptEntityInfo->poClientCmd != NULL && ptEntityInfo->poClientCmd->GetEntry() == i
        InvalidClientCmd(ptEntityInfo);
      }
      m_poEntryMng->RemoveTimeout(ptInfo);
      CertainLogError("E(%lu, %lu) Remove For CatchUp", iEntityID, iEntry);
    }
    CheckForCatchUp(ptEntityInfo, iAcceptorID, poPaxosCmd->GetMaxChosenEntry());
    return 0;
  }
  ...
}
```

由于本地不存在对应的 PLog 记录， CheckForCatchUp 最终会调用 ActivateEntry 向 Active 节点通过 RPC 同步记录。这里有一个隐含的拦截逻辑，在 src/IOWorker.cpp 中：

```
// 调用 Go 并 Delete
int clsIOWorkerRouter::GoAndDeleteIfFailed(clsCmdBase *poCmd) {
```

```
int iRet;

if (poCmd->GetCmdID() == kPaxosCmd) {
  clsPaxosCmd *poPaxosCmd = dynamic_cast<clsPaxosCmd *>(poCmd);

  // 当前 Cmd 的 Entry 小于 MaxChosenEntry
  // 或者当前 Cmd 的 Entry 等于 MaxChosenEntry 并且 SrcRecord 还没有被 Chosen
  if (poPaxosCmd->GetEntry() < poPaxosCmd->GetMaxChosenEntry() ||
      (poPaxosCmd->GetEntry() == poPaxosCmd->GetMaxChosenEntry() &&
       !poPaxosCmd->GetSrcRecord().bChosen)) {
    // 表明自己对全局仍然是需要追赶的状态，通过 CatchUpWorker 缓速 Cmd
    iRet = m_poCatchUpWorker->PushCatchUpCmdByMultiThread(poPaxosCmd);
    if (iRet != 0) {
      CertainLogError("PushCatchUpCmdByMultiThread ret %d", iRet);
      delete poCmd, poCmd = NULL;
      return -1;
    }

    return 0;
  }
}

iRet = Go(poCmd);
if (iRet != 0) {
  CertainLogError("Go E(%lu, %lu) ret %d", poCmd->GetEntityID(), poCmd->GetEntry(), iRe
  delete poCmd, poCmd = NULL;
  return -2;
}
```

```
  return 0;
}
```

CatchUpWorker 中有流量和次数限制，避免短时间内发出大量追赶请求而影响正常服务的响应：

```
// CatchUp Loop，从 CatchUp 队列中取出 Cmd，通过流量和次数控速，再发送
void clsCatchUpWorker::Run() {
  int iRet;
  SetThreadTitle("catchup_%u", m_iLocalServerID);
  CertainLogInfo("catchup_%u run", m_iLocalServerID);

  clsSmartSleepCtrl oSleepCtrl(200, 1000);

  while (1) {
    if (CheckIfExiting(1000)) {
      printf("catchup_%u exit\n", m_iLocalServerID);
      CertainLogInfo("catchup_%u exit", m_iLocalServerID);
      break;
    }

    m_poCatchUpCtrl->UpdateCatchUpSpeed(m_poConf->GetMaxCatchUpSpeedKB());
    m_poCatchUpCtrl->UpdateCatchUpCnt(m_poConf->GetMaxCatchUpCnt());

    clsPaxosCmd *poCmd = NULL;
    iRet = m_poCatchUpQueue->TakeByOneThread(&poCmd);
    if (iRet != 0) {
      oSleepCtrl.Sleep();
      PrintStat();
      continue;
    } else {
      oSleepCtrl.Reset();
    }

    uint64_t iEntityID = poCmd->GetEntityID();
    uint32_t iDestAcceptorID = poCmd->GetDestAcceptorID();
    uint64_t iByteSize = EstimateSize(poCmd);

    // 流量控速
    while (1) {
      uint64_t iWaitTimeMS = m_poCatchUpCtrl->UseByteSize(iByteSize);
      if (iWaitTimeMS == 0) {
        break;
      }

      CertainLogImpt("catchup iByteSize %lu iWaitTimeMS %lu", iByteSize, iWaitTimeMS);
      usleep(iWaitTimeMS * 1000);
    }

    // 次数控速
    while (1) {
      uint64_t iWaitTimeMS = m_poCatchUpCtrl->UseCount();
      if (iWaitTimeMS == 0) {
        break;
      }

      CertainLogImpt("catchup iWaitTimeMS %lu by count", iWaitTimeMS);
      usleep(iWaitTimeMS * 1000);
    }

    // 发送 CMD 并更新统计
    iRet = m_poIOWorkerRouter->Go(poCmd);
    if (iRet != 0) {
      CertainLogError("Go ret %d cmd: %s", iRet, poCmd->GetTextCmd().c_str());
      delete poCmd, poCmd = NULL;
    } else {
      DoStat(iEntityID, iDestAcceptorID, iByteSize);
    }
  }
}
```

B / C 机收到这些 Entry 落后的请求后，将自身的 PLog 记录回包，A 机收到后正常更新即可追赶到全局最新的进度。

3. Recover 流程

当 Entry Record 中的 Value 成功提交到 DB 之后，其对应的 PLog 就可以“安全”删除了。PLog 删除的代码在开源版中并没有提供，询问了 PaxosStore 的开发者之后，确认 PLog 定期扫描删除已经提交的 Record 以减轻存储压力。考虑以下情况：

1. A 机挂掉；
2. B / C 机达成了一些共识；
3. B / C 机删除了对应的 PLog；
4. A 机重启成功；

5. 客户端向 A 机发出读取请求，A 机发起协议，由于 B / C 进度更新，返回 RemoteNewer；
6. A 机 CheckForCatchUp，向 B / C 机请求缺失的 PLog，返回 NotFound。

此时已经没有可读取的 PLog 供 A 机恢复，A 机只能向 B / C 请求全量的 DB 数据以恢复，也就是这里说的 Recover 流程。

```
int clsEntityWorker::DoWithPaxosCmd(clsPaxosCmd *poPaxosCmd) {
  ...
  if (poPaxosCmd->GetResult() == eRetCodeNotFound) {
    CertainLogError(
        "E(%lu %lu) not found, need get all, "
        "bGetAllPending %d MaxPLogEntry %lu MaxContChosenEntry %lu",
        iEntityID, iEntry, ptEntityInfo->bGetAllPending, ptEntityInfo->iMaxPLogEntry,
        ptEntityInfo->iMaxContChosenEntry);

    if (ptEntityInfo->bGetAllPending) {
      return eRetCodeGetAllPending;
    }

    if (ptEntityInfo->iMaxContChosenEntry >= iEntry) {
      return 0;
    }

    ptEntityInfo->bGetAllPending = true;
    InvalidClientCmd(ptEntityInfo, eRetCodeGetAllPending);

    iRet = clsGetAllWorker::EnterReqQueue(poPaxosCmd);
    if (iRet == 0) {
      return eRetCodePtrReuse;
    }

    ptEntityInfo->bGetAllPending = false;
    return eRetCodeQueueFailed;
  }
  ...
}

int clsGetAllWorker::HandleInQueue(clsPaxosCmd *poCmd) {
  OSS::ReportGetAllReq();

  clsDBBase *pDataDB = clsCertainWrapper::GetInstance()->GetDBEngine();
  uint64_t iCommitedEntry = 0;
  int iRet = pDataDB->GetAllAndSet(poCmd->GetEntityID(), poCmd->GetSrcAcceptorID(), iComm
  if (iRet != 0) {
    CertainLogError("EntityID %lu GetAllAndSet iRet %d", poCmd->GetEntityID(), iRet);
    OSS::ReportGetAllFail();
  }

  poCmd->SetResult(iRet);

  poCmd->SetEntry(iCommitedEntry);
  uint64_t iEntityID = poCmd->GetEntityID();
  uint32_t iSrcAcceptorID = poCmd->GetSrcAcceptorID();

  while (1) {
    iRet = clsEntityWorker::EnterGetAllRspQueue(poCmd);
    if (iRet != 0) {
      CertainLogError("EntityID %lu EnterGetAllRspQueue iRet %d", poCmd->GetEntityID(), i
      poll(NULL, 0, 10);
    } else {
      break;
    }
  }

  CertainLogImpt("EntityID %lu srcAcceptorid %u iCommitedEntry %lu iRet %d", iEntityID,
                 iSrcAcceptorID, iCommitedEntry, iRet);

  return 0;
}
```

这里需要 DB 提供 GetAllAndSet 方法用以向 iAcceptorID 对应的机器获取 iEntityID 相关的全量数据。一般通过 RPC 请求数据，可以参见 example/DBImpl.cpp：

```
// 使用 RPC 向 iAcceptorID 取数据
int clsDBImpl::GetAllAndSet(uint64_t iEntityID, uint32_t iAcceptorID, uint64_t &iMaxCommi
  clsAutoDisableHook oAuto;
  CertainLogInfo("Start GetAllAndSet()");

  int iRet = 0;

  // Step 1: Sets flags for deleting.
  {
    Certain::clsAutoEntityLock oEntityLock(iEntityID);
    iRet = SetEntityMeta(iEntityID, -1, 1);
    if (iRet != 0) {
      CertainLogError("SetFlag() iEntityID %lu iRet %d", iEntityID, iRet);
      return Certain::eRetCodeSetFlagErr;
    }
  }

  // Step 2: Deletes all kvs in db related to iEntityID.
  std::string strNextKey;
  EncodeInfoKey(strNextKey, iEntityID, 0);
  do {
    iRet = Clear(iEntityID, strNextKey);
    if (iRet != 0) {
      CertainLogError("Clear() iEntityID %lu iRet %d", iEntityID, iRet);
      return Certain::eRetCodeClearDBErr;
    }

    if (!strNextKey.empty())
      poll(NULL, 0, 1);
  } while (!strNextKey.empty());

  // Step 3: Gets local machine ID.
  clsCertainUserImpl *poCertainUser = dynamic_cast<clsCertainUserImpl *>(
      Certain::clsCertainWrapper::GetInstance()->GetCertainUser());
  uint32_t iLocalAcceptorID = 0;
  iRet = poCertainUser->GetLocalAcceptorID(iEntityID, iLocalAcceptorID);
  if (iRet != 0) {
    CertainLogError("GetLocalAcceptorID() iEntityID %lu iRet %d", iEntityID, iRet);
    return Certain::eRetCodeGetLocalMachineIDErr;
  }

  grpc_init();
```

```
grpc::ChannelArguments oArgs;
oArgs.SetInt(GRPC_ARG_MAX_CONCURRENT_STREAMS, 20);

iRet = -1;
// iAcceptorID is the ID of peer machine.
uint32_t iAcceptorNum = Certain::clsCertainWrapper::GetInstance()->GetConf()->GetAccept
for (iAcceptorID = (iLocalAcceptorID + 1) % iAcceptorNum;
     iRet != 0 && iAcceptorID != iLocalAcceptorID;
     iAcceptorID = (iAcceptorID + 1) % iAcceptorNum) {
  std::string strAddr;
  iRet = poCertainUser->GetServiceAddr(iEntityID, iAcceptorID, strAddr);
  if (iRet != 0) {
    CertainLogError("GetSvrAddr() iEntityID %lu iRet %d", iEntityID, iRet);
    iRet = Certain::eRetCodeGetPeerSvrAddrErr;
    continue;
  }

  // Step 4: Gets commited entry and sets local value.
  example::GetRequest oRequest;
  oRequest.set_entity_id(iEntityID);
  example::GetResponse oResponse;

  clsClient oClient(strAddr);

  static const int kRetry = 3;
  for (int i = 0; i < kRetry; ++i) {
    grpc::Status oRet = oClient.Call(oRequest.entity_id(), example::OperCode::eGetDBEnt
                                     &oRequest, &oResponse, strAddr);
    iRet = oRet.error_code();
    if (iRet == 0)
```

```
        break;
    }

    if (iRet != 0) {
      CertainLogError("GetEntityMeta() iEntityID %lu iRet %d", iEntityID, iRet);
      iRet = Certain::eRetCodeGetPeerCommitedEntryErr;
      continue;
    }

    iMaxCommitedEntry = oResponse.max_commited_entry();
    uint64_t iSequenceNumber = oResponse.sequence_number();

    {
      Certain::clsAutoEntityLock oEntityLock(iEntityID);
      iRet = SetEntityMeta(iEntityID, iMaxCommitedEntry, -1);
      if (iRet != 0) {
        CertainLogError("SetEntityMeta() iEntityID %lu iRet %d", iEntityID, iRet);
        iRet = Certain::eRetCodeSetDBEntityMetaErr;
        continue;
      }
    }

    // Step 5: Gets data from peer endpoint.
    std::string strNextKey;
    EncodeInfoKey(strNextKey, iEntityID, 0);
    do {
      std::string strWriteBatch;
      oRequest.set_max_size(1024 * 1024);
      oRequest.set_next_key(strNextKey);
      oRequest.set_sequence_number(iSequenceNumber);
```

```
oResponse.Clear();
for (int i = 0; i < kRetry; ++i) {
  grpc::Status oRet = oClient.Call(oRequest.entity_id(), example::OperCode::eGetAll
                                   &oRequest, &oResponse, strAddr);
  iRet = oRet.error_code();
  if (iRet == 0) {
    strNextKey = oResponse.next_key();
    strWriteBatch = oResponse.value();
    break;
  }
}

if (iRet != 0) {
  CertainLogError("GetAllForCertain() iEntityID %lu iRet %d", iEntityID, iRet);
  iRet = Certain::eRetCodeGetDataFromPeerErr;
  break;
}

if (!strWriteBatch.empty()) {
  dbtype::WriteBatch oWriteBatch(strWriteBatch);
  if (strWriteBatch.empty())
    oWriteBatch.Clear();
  iRet = MultiPut(&oWriteBatch);
  if (iRet != 0) {
    CertainLogError("WriteBatch::Write() iEntityID %lu iRet %d", iEntityID, iRet);
    iRet = Certain::eRetCodeCommitLocalDBErr;
    break;
  }
}
```

```
    if (!strNextKey.empty())
      poll(NULL, 0, 1);
  } while (!strNextKey.empty());

  if (iRet != 0) {
    // Step 6: Re-deletes all kvs in db related to iEntityID.
    std::string strNextKey;
    EncodeInfoKey(strNextKey, iEntityID, 0);
    do {
      int iDelRet = Clear(iEntityID, strNextKey);
      if (iDelRet != 0) {
        CertainLogError("Re-clear() iEntityID %lu iRet %d", iEntityID, iRet);
        iRet = Certain::eRetCodeReClearDBErr;
      }

      if (!strNextKey.empty())
        poll(NULL, 0, 1);
    } while (!strNextKey.empty());
  }
}

if (iRet != 0) {
  CertainLogError("Abort GetAllAndSet() iEntityID %lu iRet %d", iEntityID, iRet);
  return iRet;
}

// Step 7: Clear flag.
{
  Certain::clsAutoEntityLock oEntityLock(iEntityID);
  iRet = SetEntityMeta(iEntityID, -1, 0);
```

```
    if (iRet != 0) {
      CertainLogError("SetFlag() iEntityID %lu iRet %d", iEntityID, iRet);
      return Certain::eRetCodeClearFlagErr;
    }
  }

  CertainLogInfo("Finish GetAllAndSet()");

  return Certain::eRetCodeOK;
}
```

在恢复数据的过程中，需要为该 iEntityID 加锁加 Flag 以拒绝其他读写服务，样例里使用的是协程锁，并且在 Entity Meta 中设置了 kDBFlagCheckGetAll=1 的 Flag。

4. WaitingMsg

在 EntryInfo 里有一项 WaitingMsg 属性，当需要等待 PLog 读取或写入时，会将当前的 Cmd 存入该列表中暂存起来：

```
ptInfo->apWaitingMsg[iAcceptorID] = poPaxosCmd;
```

而当 PLog 读写完成时，会调用 DoWithWaitingMsg 处理这些等待的 Cmd：

```
// 将 WaitingMsg 中的 Cmd 推入 IO Req
int clsEntityWorker::DoWithWaitingMsg(clsPaxosCmd **apWaitingMsg, uint32_t iCnt) {
  uint32_t iFailCnt = 0;
  for (uint32_t i = 0; i < iCnt; ++i) {
    clsPaxosCmd *poPaxosCmd = apWaitingMsg[i];
    apWaitingMsg[i] = NULL;

    if (poPaxosCmd == NULL) {
      continue;
    }

    CertainLogInfo("cmd: %s", poPaxosCmd->GetTextCmd().c_str());

    int iRet = DoWithIOReq(dynamic_cast<clsCmdBase *>(poPaxosCmd));
    if (iRet < 0) {
      iFailCnt++;
      CertainLogError("DoWithIOReq ret %d cmd %s", iRet, poPaxosCmd->GetTextCmd().c_str()
    }
    if (iRet != eRetCodePtrReuse) {
      delete poPaxosCmd, poPaxosCmd = NULL;
    }
  }

  if (iFailCnt > 0) {
    CertainLogError("iFailCnt %u", iFailCnt);
    return -1;
  }

  return 0;
}
```

0 comments

Write Preview Aa