**分布式使用说明**

game

lobby

PHP

client

Dispatch

框架图

lobby: 大厅服务器：保持客户端的连接

game：游戏服务器：具体的玩牌逻辑

Dispatch: 中心服务器：存储所有服务器的状态，动态增减大厅服务器，全服消息转发

**多开大厅，同一个游戏多开进程**

①多开大厅

php从dispatch获取所有的大厅服务器信息，选择一个返回给客户端，客户端再 向大厅发起连接。

lua配置文件中masetersvrid为机器人服务器，机器人服务器只是lobbyServer可执 行文件的拷贝。机器人服务器只用于加载机器人，向Gameserver登陆机器人。不 会让玩家登陆，php给客户端返回的大厅服务器信息不 会有机器人服务器的信息。

②同一个游戏多开进程

返回给客户端的游戏服务器信息中，添加每个进程上的玩家人数量，机器人数量， 客户根据筛选条件进入一个gameid指定的游戏。

**Dispatch 中心服务器**

①lobby ，gameserver在启动的时候，都会注册到Dispatch。

②在新增加一个大厅的时候，Dispatch会通知子游戏，有新的大厅进程启动，子游戏进 程发送注册信息到新大厅。

③退休子游戏，Dispatch发消息到相应的子游戏进程，及所有的大厅进程

退休大厅，Dispatch发送消息到相应的大厅进程

**动态增减进程的操作流程**

① 添加的大厅进程

在serverinfo添加要新增的进程配置信息，启动进程

② 退休大厅进程

在后台操作退休需要退休的大厅进程

③添加子游戏进程

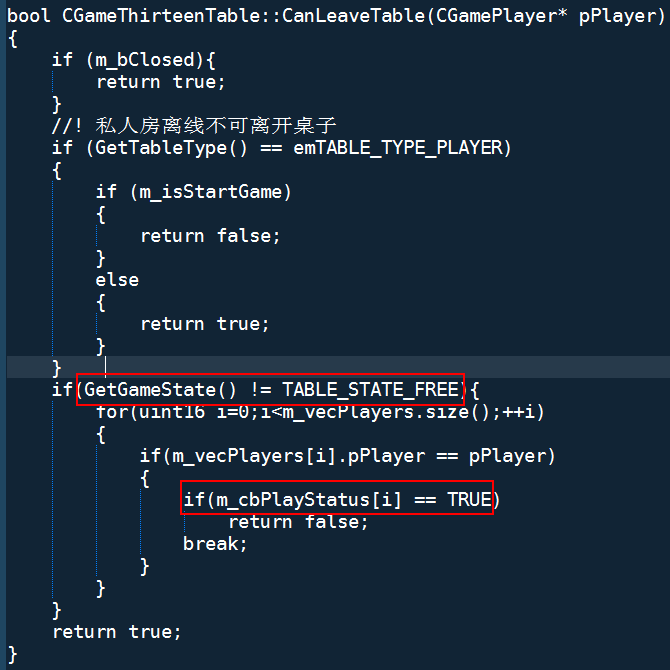
在serverinfo中添加新增子游戏进程的配置信息，启动进程

④退休子游戏进程

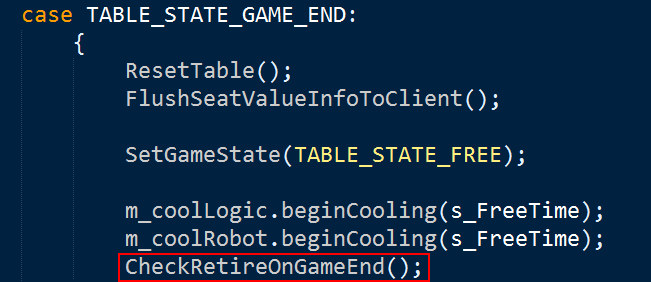
dispatch发消息到要退休的子游戏进程，退休子游戏，并通知大厅svr退休的子游戏进程的信息，在退休期间，玩家不能进入该子游戏进程。

**子游戏退休逻辑：①金币场** 子游戏进程收到退休命令后，正在玩牌的玩家，该局结束后会被踢回到大厅，当该子游戏进程上没有玩家或退休时间超时后，该进程会自杀掉。 由于每个子游戏中，一局结束后，玩家玩家可以退回到大厅的判断条件不同，所以无法统一修改。 所以在**CGameTable**基类中添加virtual void CheckRetireOnGameEnd()函数，牌局结束后，调用该函数将玩家踢回到大厅。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*例如在十三水中添加该函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



只有当桌子状态为TABLE\_STATE\_FREE，m\_cbPlayStatus[i]为false时，玩家才能退出十三水。



此时调用该函数，如果该进程已经收到退休命令，桌子上的玩家就会被踢到大厅。 这个函数需要根据具体的子游戏的逻辑进行调用。