性能管理 Performance Management

1 引言

1.1 编写目的

该组件为软件复用课程的课程项目，撰写此文档的目的在于使得用户能够更好的复用该组件。

2 组件简介

2.1 功能

这一组件用来接收应用程序的性能指标，每分钟自动生成性能报告，对每个指标求和，性能报告输出到单独文件（报告文件名为：state.json）

2.2 输入

应用程序的性能指标

2.3 输出

每分钟输出性能报告，内容是性能指标之和，输出到 state.json

2.4 参数

ThrottleManager()：建立一个单例模式的节流管理器，该管理器可以动态地将连接到服务器的客户端与请求添加到延迟队列中

ThrottleStatesListener()：建立一个单例模式的节流监听器，该监听器内部含有一个计时器（dumpTimer）以及一个消息类型和消息数量的映射（map）

run()：为计时器的日程，每分钟将服务器的状态打印到文件中

2.5 使用方法

```
private ListeningExecutorService service = MoreExecutors.listeningDecorator(
            Executors.newFixedThreadPool(4)
);

public void sendMessage(Message message, String username){
      ListenableFuture future = service.submit(Executors.callable(new
UnicastResponseTask(message, username)));
      future.addListener(new ThrottleStatsListener(), service);
}

public void sendMessage(Message message){
      ListenableFuture future = service.submit(Executors.callable(new
MulticastResponseTask(message)));
      future.addListener(new ThrottleStatsListener(), service);
}
```

本段代码为发送消息，上方为单播模式，下方为广播模式。在发送消息的过程中要建立一个监听器。

```
private Map<Long, Integer> loginCountMap = new ConcurrentHashMap<Long, Integer>();

public void registerLogin(long clientID){
      assert Validators.validateClientID(clientID);
      if(loginCountMap.containsKey(clientID)){
            Integer currentCount = loginCountMap.get(clientID);
            currentCount = currentCount + 1;
            loginCountMap.put(clientID, currentCount);
      }else{
            loginCountMap.put(clientID, 1);
      }
      /** Produce count down task **/
      DelayedNotify notify = new DelayedNotify(clientID, Constants.MILLIS_PER_MINUTE
                  * Constants.NANOS_PER_MILLI);
      ThrottleManager.getInstance().notity(notify);
}
```

本段代码为将一个用户注册到服务器中，并且通知节流管理器。