Scenario分析

互操作性：场景：和第三方系统进行交互

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 系统 |
| 刺激 | 系统请求和第三方系统交互 |
| 制品 | 第三方系统 |
| 环境 | 系统运行时或构建时 |
| 响应 | 系统成功拿到第三方系统的服务 |
| 响应度量 | 交互请求成功率大于99.9% |

易用性：场景：正常操作

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 想要使用系统完成操作 |
| 制品 | 系统 |
| 环境 | 系统运行时 |
| 响应 | 能够正确引导用户完成操作  提供撤销或取消功能，在用户操作过程中识别并纠正用户错误  错误发生时的恢复 |
| 响应度量 | 用户完成单项操作的时间小于等于2min  用户操作成功率应该大于等于99%  错误恢复时间小于5min |

User-broker模块迭代

## 第二次迭代

## 选择元素

选择的元素是user-broker模块，负责用户账户、订单和购票请求的转发分派。

## 选择ASR

第二次迭代选择的ASR是性能、可用性。

对于该系统，由于该系统的用户基数较大且流量高峰时间比较集中，就要求系统在正常情况下和高并发模式下能做出正确而快速的响应，另外由于broker在系统中的枢纽地位，一旦出错当机，对系统的影响是难以估量的，因此系统的可用性和性能显得尤为重要。

## 候选策略表和决策

|  |  |  |  |
| --- | --- | --- | --- |
|  | 优点 | 缺点 | 是否采用 |
| 主动冗余 | 通过对重要broker的备份，可以在主broker出现问题时切换到备份的broker。增强了系统的可用性和安全性；另外由于broker架构的灵活性，可以比较方便地切换主从broker，回复速度较快 | 冗余的broker增加了系统的负担，当系统切换broker时不可避免地会增加系统的负载压力和复杂度，耗费资源 | 不采用，耗费资源太多 |
| 被动冗余 | 通过从broker中的某些信息来恢复主broker。增强了系统的可用性，两个broker各自有不同的功能，功能不完全冗余，系统的负担较小 | 主broker和从broker之间的信息同步可能对系统造成一定的压力，而且错误恢复的速度比较慢 | 采用，可以对于较重要的或者有前后顺序逻辑的broker使用被动冗余来加强可用性 |
| 移除可疑broker | 错误恢复后将发生错误的broker移除，可以防范broker再次出错 | 某些broker的功能是无法替代的，移除后可能造成系统瘫痪 | 不采用 |
| 模块细化 | User-broker可以将自己的功能分解成几个较独立的功能模块，分派给二级broker。User-broker负载压力减小，能够更好更快地处理高并发问题 | Broker之间的通信更加复杂，增加了系统的复杂度 | 采用 |
| 优先级调度 | User-broker通过识别优先级对优先级更高的请求进行优先转发，可以保证系统对于重要功能的快速响应 | 请求优先级设置问题，难以决定哪些请求更重要。另外优先级调度也会增加系统的复杂度 | 不采用，对于本系统而言用户的请求并没有很明显的优先级 |
| 建立缓存模块 | 通过把请求结果缓存到缓存模块中，对于重复请求可以直接响应而不用向server进行请求，解决一部分并发问题，可以减少等待时间，提高系统性能 | 需要对用户请求结果进行缓存，可能造成一定的负载压力，另外某些实时变化的信息并不适合缓存 | 不采用，因为不会有多个用户对user-broker中的信息同时进行访问（账户信息、订单信息、票务信息），没有缓存的必要 |

## 第二次迭代结果

如图所示，新增元素说明如下

account-broker模块：负责user-broker中的账户请求服务的二级转发

order-broker模块：负责user-broker中的订单请求服务的二级转发

ticket-broker模块：负责user-broker中的票务操作请求服务的二级转发

schedule-cache模块：负责对列车调度表进行缓存

