**步骤一：**

该迭代不需要此步骤

**步骤二：**

1. **选择元素**：该迭代针对购票模块进行分解
2. **选择ASR**：可靠性Reliability与互操作性Interopreability，原因如下
3. 选择可靠性：购买车票、改签以及退票是该系统最核心的功能，也是用户最需要的功能。系统应该保证其可靠性需求，在运行时期不能崩溃，不响应。
4. 选择互操作性：购票模块的实现需要频繁访问外部服务，比如外部支付平台等，所以对外需要提高系统的访问效率。同时，作为SOA架构的一部分，购票模块也是服务的提供者，为第三方平台提供服务接口。所以互操作性是系统的ASR。

**步骤三：**

确定框架驱动元素

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动元素 | 重要性 | 难易度 |
| 1 | 场景1：阻止恶意抢票 | 高 | 中 |
| 2 | 场景4：和第三方交互 | 中 | 高 |
| 3 | 场景6：高并发的客户流量 | 高 | 高 |
| 4 | 场景9：系统不可以崩溃、无响应 | 高 | 高 |
| 5 | 场景10：简单操作 | 高 | 低 |
| 6 | 场景11：改签退票操作 | 高 | 低 |

**步骤四：**

1. **针对可靠性（Reliability）**
2. **设计关注点**

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 检测可能出现的故障 | 状态检测 |
| 防止发生故障 | 数据备份 |
| 业务高并发处理 |
| 计算资源冗余 |

1. **备选方案以及选择方案**

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | 优点 | 缺点 | 是否采用 |
| 心跳模式 | 实现简单，用于监测组件是否正常工作时每次传输的数据量少。可以实现周期性的检测。 | 需要持续占用部分计算资源用于监测组件，即存在资源的等待。 | 采用。可以通过周期性的监测有效保证被监测组件正常工作。 |
| Ping/Echo模式 | 可以通过中断的方式减少计算资源的等待，监测方式更具有可控性。 | 监测组件每次传输的数据量较多。实现较为复杂，且该方式不适合周期性的组件监测。 | 不采用。Ping/echo不适用于当前情况下的周期性检查。 |
| 同步更新备份数据 | 同步更新备份数据能保证备份数据的实时性。当需要使用备份数据时，同步更新的模式能提供最新的数据。 | 数据吞吐量大时可能增加传输负担。 | 采用。为了防止数据丢失，数据的实时性更为重要。 |
| 定时更新备份数据 | 可以在高吞吐量时降低传输负担。 | 定时更新备份数据可能会因为积累了大量未更新数据而造成数据需要传输的量过大的情况，增加传输负担，影响正常数据的传输。 | 不采用。同步更新更适合保证系统的可靠性， |
| 使用缓冲 | 将调用频率高的服务比如查询某两地之间的车票数据缓存，以提高高并发的处理能力。 | 需要额外的存储空间。 | 采用。可以提高SOA中组件的高并发处理能力 |
| 增加物理资源 | 更多的处理资源、内存、更快的网络带宽会提高系统的高并发处理能力。 | 增加物理资源需要资金开支 | 采用。增加资源可以带来性能上显著的提示，由此带来的成本可以接受 |
| 消极冗余 | 在存在多个冗余的计算资源的条件下，仅有一个组件响应请求，其他冗余组件与之保持状态匹配，随时可以进行替换 | 当前组件和冗余组件之间存在频繁的信息交流 | 采用。在不占用太多计算资源的情况下，能够有效处理错误出现的情况。 |
| 积极冗余 | 在存在多个冗余的计算资源的条件下，所有组件均响应请求，最终决定一个组件的响应被采用。 | 占用的计算资源大 | 不采用。本策略增大了请求处理的任务量。 |
| 冷冗余 | 对发生错误组件进行替换，保证错误的出现不影响系统 | 系统宕机时间略长，多用于可以人工操作的组件 | 不采用。系统需要不间断的持续提供服务。 |

1. **针对互操作性（Interoperability）**
2. **设计关注点**

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 定位外部服务 | 查询外部服务 |
| 管理接口 | 管理对外接口 |

1. **备选方案以及选择方案**

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | 优点 | 缺点 | 是否采用 |
| 注册中心服务查询 | 通过向服务注册中心查询相关服务，可以灵活地调用需要的服务，在SOA架构中也符合服务调用思想。 | 每次查询服务需要消耗一定时间。 | 采用。在SOA架构中适用。 |
| 服务绑定 | 可以高效调用服务，不需要查询。 | 服务的接口固定，不利于服务扩展。 | 不采用。不利于SOA架构的实现。 |
| 定制接口 | 实现较为简单。 | 定制接口针对一个接口进行功能的增加或删除。所以定制接口会增加代码的复杂度，导致系统不利于维护。 | 不采用。不利于系统的维护。 |
| 组织协调接口调用 | 组织协调接口调用通过对不同接口进行协调排序，使接口之间分工合作完成功能。 | 需要考虑接口之间的协调，增加了设计的复杂度。 | 采用。有利于系统的扩展性。 |