ESB模块

Performance，Security

**步骤一：**

该迭代不需要此步骤

**步骤二：**

该迭代针对ESB模块进行分解

**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3：应对系统错误 | 高 | 高 |
| 2 | 场景5：正常的客户端操作 | 高 | 低 |
| 3 | 场景6：高并发的客户流量 | 高 | 高 |
| 4 | 场景7：硬件升级 | 中 | 低 |

**步骤四：**

选择满足架构驱动因素的架构模式。

1. **针对性能（Performance）**
2. **确定设计关注点**

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 单次请求的延时 | 单任务的性能 |
| 大量请求到来时的吞吐量 | 分布式处理 |
| 请求的管理 |

1. **备选方案以及选择方案**
2. **单任务的性能**

·用于区分的参数：

- 成本

- 提高程度

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 成本 | 提高程度 |
| 1 | 改用 C / C++ 实现 | 高 | 中 |
| 2 | 更换更好的硬件 | 中 | 中 |

* 选择方案：更换更好的硬件
* 选择原因：改用C／C++ 实现的硬件成本虽然低，但是维护代码的成本和后续开发成本却会比一次性更新硬件要高，同时比起Java等往往也只能提高数倍的性能。

1. **分布式处理**

·用于区分的参数：

- 管理难度

- 资源利用率

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 管理难度 | 资源利用率 |
| 1 | 在Registry注册多个ESB | 低 | 高 |
| 2 | 本地分发 | 高 | 低 |

* 选择方案：在Registry注册多个ESB
* 选择原因：由于ESB本身没有像数据库一样的数据一致性需求，可以很容易地进行分布式架构，而比起请求都发到一个ESB再进行分发给本地的ESB，将多个ESB注册到Registry组件，直接在服务绑定的时候负载均衡，能更有效减少请求的层数有利于提高性能。而且交由Registry统一管理能降低管理难度，且不会导致single point of failure。

1. **请求的管理**

·用于区分的参数：

- 高并发能力

- 用户友好程度

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 高并发处理能力 | 用户友好程度 |
| 1 | 持久化未处理请求 | 中 | 中 |
| 2 | 优先队列，丢弃低优先请求 | 高 | 低 |

* 选择方案：优先队列，丢弃低优先请求
* 选择原因：持久化处理可能导致延时非常高，此时用户可能也已经放弃了该次请求，优先队列虽然会丢弃一些请求，但是可以保证系统运转的流畅，而且不会对后续的系统组件增加负担。

1. **针对安全性（Security）**
2. **确定设计关注点**

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 攻击检测 | 攻击模式识别 |
| 攻击防护 | 保护信息不被窃取 |
| 保护信息不被修改 |

1. **备选方案以及选择方案**
2. **攻击模式识别的方式**

·用于区分的参数：

- 安全性

- 性能影响

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 安全性 | 性能影响 |
| 1 | 精确检验 | 高 | 高 |
| 2 | 粗略检验 | 中 | 低 |
| 3 | 分场景使用上两者 | 高 | 中 |

* 选择方案：分场景使用上两者
* 选择原因：由于该组件在安全性之外的另一个重要的质量需求是性能，故不全部采取精确检验的方式，只在和用户私密信息有关的请求中采取精确检验。而像查询余票这种不泄密的请求中粗略判断即可。

1. **保护信息不被窃取**

·用于区分的参数：

- 安全性

- 性能影响

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 安全性 | 性能影响 |
| 1 | 信息加密传输 | 高 | 中 |
| 2 | 授权管理 | 中 | 低 |
| 3 | 结合两者 | 高 | 中 |

* 选择方案：结合两者
* 选择原因：两个缺一不可，缺少任何一个都不能保证信息能不被窃取。

1. **保护信息不被修改**

·用于区分的参数：

- 安全性

- 成本

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名 | 安全性 | 成本 |
| 1 | 授权管理 | 中 | 低 |
| 2 | 备份冗余 | 低 | 高 |

* 选择方案：授权管理
* 选择原因：冗余备份只能在数据在被修改了之后恢复，但是不能从根本上阻止发生，而且冗余备份需要更多的硬件成本，且降低了性能。