## 第二次迭代

#### 选择元素

第二次迭代选择的元素是ESB模块。

#### 选择ASR

第二次迭代选择的ASR是性能、安全性。

对于该系统架构，ESB模块是前后端连接的最重要渠道，前端来的所有流量都要经过ESB才能调用到后端的其他服务，所以性能对于ESB模块极为重要，因为低性能的ESB可能成为整个系统的瓶颈部分。此外，由于此处是前端和后端交互的地方，所以还需要防护从外部来的攻击，保证信息安全和保证系统被攻击后能正常运行。综上，我们选择性能和安全性作为ASR。

#### 候选策略表和决策

#### 性能

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | 优点 | 缺点 | 决策 |
| 增加资源（硬件能力） | 对其他的质量属性没有太多的影响 | 增加硬件成本（租用购买成本及维护成本） | 采用。比起其他质量属性的降低，经济成本的投入更值得。首先应该保证ESB不成为整个系统的瓶颈部分。 |
| 引入并行 | 并行处理，提高吞吐量，减少平均阻塞等待时间 | 需要对架构作较大的改动，增加实现难度 | 采用。部署多个ESB并注册好，在客户端绑定服务时进行负载均衡 |
| 限制请求执行时间 | 控制一项请求的操作时间，保证处理资源的合理分配 | 丢失请求，降低用户的体验。另一方面计算请求执行时间也会带来额外的负担。 | 不采用。缺点比较明显，可以通过其他的策略提高ESB模块的性能。 |
| 提高处理效率（更高效的库，优化算法等） | 能提高单个请求的处理速度；对其他的外部质量属性没有太多的影响。 | 实现难度大，投入不一定有产出 | 采用。缺点基本可以忽略，但是成功后带来的优点很大，能够整体上根本上提高ESB的性能。 |
| 缓存请求结果 | 减少了对其他服务的访问次数，服务的访问往往是性能的瓶颈。而且还能减少服务提供者的负担。 | 信息的实时性有一定程度的损失。 | 采用。对服务进行访问本身就有一定程度的延时，且系统的实时性需求不是很高。 |

#### 安全性

|  |  |  |  |
| --- | --- | --- | --- |
| 策略 | 优点 | 缺点 | 决策 |
| 主动攻击检测 | 及时发现攻击方，避免更大的损失 | 带来一定的性能负担 | 采用。对攻击的主动识别是有必要的，否则ESB模块作为前后端的通道，一旦被攻陷会导致整个系统无法正常运行。 |
| 数据验证 | 防止数据被恶意修改 | 带来一定的性能负担 | 采用。系统牵涉个人身份信息，银行信息等敏感信息，需要保证信息的integrity |
| 身份验证 | 可以防止资源被恶意方访问 | 带来一定的性能负担 | 采用。系统需要保证个人身份信息，银行信息等敏感信息不被其他人访问 |
| 数据加密 | 可以防止信息传输过程中被窃听，从而泄露信息 | 带来一定的性能负担，且需要其他的组件的配合，减弱了互操作性 | 采用。系统需要保证个人身份信息，银行信息等敏感信息不被其他人访问 |
| 备份恢复 | 可以从被攻击的状态较快恢复 | 提高了硬件成本。 | 采用。可以作为应急措施，使得ESB模块的沦陷不会带来全局的瘫痪。 |

#### 第二次迭代结果

新增元素如下：

缓存模块 (Cache): 缓存请求调用后端服务后得到的结果，在碰到相同请求的时候将缓存的结果返回，在一定时间后将过时的数据删除。

冗余备份模块 (Redundancy): 冗余存储请求队列中的信息，在主服务失去响应后进行恢复。

过滤模块 (Filter): 对请求进行身份验证，攻击检测，丢弃不合法的请求。同时在收到的时候解密，在发出信息时加密。

数据验证模块 (Data Verifier): 验证数据是否正常，是否被修改过。

监测重启模块 (Monitor): 监测主服务的状态，若无响应，则从冗余备份模块中进行恢复。

