小组成员

杨雁飞 141250169

邢程 141250153

徐璠斐 141250157

陶子涵 141250125

张瑞 141250189

宋益明 141250111

徐珠峰 141250164

联系方式 [141250169@smail.nju.edu.cn](mailto:141250169@smail.nju.edu.cn)

2017年3月10日星期五

七只松鼠项目文档

目录

[1. 系统功能需求功能 2](#_Toc477271730)

[2. 应用ADD过程 2](#_Toc477271731)

[2.1. 分布式微服务架构 2](#_Toc477271732)

[2.1.1. 迭代一 2](#_Toc477271733)

[2.1.2. 迭代二 7](#_Toc477271734)

[2.1.3. 迭代三 13](#_Toc477271735)

[2.1.4. 迭代四 19](#_Toc477271742)

[2.1.5. 迭代五 26](#_Toc477271746)

[2.1.6. 迭代六 31](#_Toc477271747)

[2.1.7. 最终架构视图 38](#_Toc477271748)

[2.2. MVC 38](#_Toc477271749)

[2.2.1. 迭代一 38](#_Toc477271750)

[2.2.2. 迭代二 39](#_Toc477271751)

[2.2.3. 迭代三 43](#_Toc477271752)

[2.2.4. 迭代四 48](#_Toc477271753)

[2.2.5. 迭代五 52](#_Toc477271754)

[2.1.6. 迭代六 54](#_Toc477271755)

[2.2.7. 最终架构视图 54](#_Toc477271756)

[3. 架构对比 55](#_Toc477271757)

[3.1. 综述 55](#_Toc477271758)

[3.2. 性能对比 55](#_Toc477271759)

[3.3. 可用性对比 55](#_Toc477271760)

[3.4. 安全性对比 55](#_Toc477271761)

[3.5. 成本对比 55](#_Toc477271762)

[4. 类图 56](#_Toc477271763)

[4.1. 类图 56](#_Toc477271764)

[4.2. 映射关系 56](#_Toc477271765)

[5. 挑战与经验 57](#_Toc477271766)

[6. 个人总结 57](#_Toc477271767)

# 系统功能需求功能

1. 零食购买：用户可选择购买的零食数量，下订单购买零食，并扫码完成支付。
2. 零食查看：用户可查看零食的描述，包括商品基本信息（零食名、单价、库存量、生产日期、产地、成分）和高清大图，全方位、多角度地了解零食。
3. 购物车管理：用户可以将心仪零食加入购物车；将购物车中的零食删除；增减购物车中的零食数量。
4. 帐户管理：用户可设置个人信息，包括用户昵称、手机号、收货地址；可修改密码。
5. 订单管理：用户可以查看全部的订单信息，每一笔订单都包括订单编号、下单时间、零食种类及数量、总金额、订单状态(待付款、待发货、待签收、退款)、物流信息。
6. 报表查看：管理员可以查看系统在一段时间内的零食销售情况。
7. 零食管理：管理员可对系统内的零食进行管理

# 应用ADD过程

## 分布式微服务架构

### 迭代一

#### 需求信息

1. **功能性需求：**
2. 零食购买：用户可选择购买的零食数量，下订单购买零食，并扫码完成支付。
3. 零食查看：用户可查看零食的描述，包括商品基本信息（零食名、单价、库存量、生产日期、产地、成分）和高清大图，全方位、多角度地了解零食。
4. 购物车管理：用户可以将心仪零食加入购物车；将购物车中的零食删除；增减购物车中的零食数量。
5. 帐户管理：用户可设置个人信息，包括用户昵称、手机号、收货地址；可修改密码。
6. 订单管理：用户可以查看全部的订单信息，每一笔订单都包括订单编号、下单时间、零食种类及数量、总金额、订单状态(待付款、待发货、待签收、退款)、物流信息。
7. 报表查看：管理员可以查看系统在一段时间内的零食销售情况。
8. 零食管理：管理员可对系统内的零食品种（包括零食名、价格、库存量、商品基本信息、高清大图）进行增删改查。
9. **场景**

**场景1 ： >1000用户同时购买同一件商品（可靠性、性能）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户 |
| 刺激 | >1000用户同时购买一件商品 |
| 制品 | 系统负载均衡模块、计算业务模块、数据库系统 |
| 环境 | 系统正常运行 |
| 响应 | 系统能正常完成用户的购买操作  服务器对大规模的计算请求进行节点分发  数据库中的数据进行正常变化 |
| 响应度量 | >99%的用户请求能正常完成  >85%的用户的请求能在1s内完成 |

**场景2 ：系统需要加入新服务（可扩展性、可维护性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 系统开发人员 |
| 刺激 | 有新功能需要加入系统 |
| 制品 | 系统计算业务模块 |
| 环境 | 系统已经发布，在正常工作 |
| 响应 | 系统发布新的功能  系统功能发布过程应该是简洁、方便的 |
| 响应度量 | 系统发布不影响99%以上用户的使用  功能的发布应该在1小时内完成 |

**场景3 ：系统计算能力需要增减（可伸缩性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 系统开发人员 |
| 刺激 | 系统需要增减业务的计算能力 |
| 制品 | 系统业务计算模块 |
| 环境 | 系统已发布，正常工作 |
| 响应 | 系统增减对应业务功能的计算能力 |
| 响应度量 | 计算能力的改变能在1小时内完成  计算能力的改变不影响99%以上用户的使用 |

**场景4 ：用户进行正常操作（易用性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 系统用户 |
| 刺激 | 用户执行正常系统操作 |
| 制品 | 系统 |
| 环境 | 系统正常运行 |
| 响应 | 正确引导用户操作  操作简单易学 |
| 响应度量 | 完成简单、中等、复杂任务的时间分别不超过20s、1min、2min  >99%的用户能正确使用本系统完成任务  >90以上的用户能在1小时内熟练使用本系统 |

**场景5 ：未登陆用户进行支付操作（安全性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 未经授权的个人或其它系统用户 |
| 刺激 | 进行支付操作（或其他安全性操作） |
| 制品 | 系统安全模块 |
| 环境 | 系统正常运行，用户未授权 |
| 响应 | 系统拒绝用户请求  系统记录相应日志 |
| 响应度量 | 系统能在2s内拒绝用户请求  系统应在拒绝请求之后立即记录相应日志 |

**场景6 ：用户进行了错误操作（鲁棒性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户 |
| 刺激 | 进行了错误操作 |
| 制品 | 错误处理模块 |
| 环境 | 系统正常运行 |
| 响应 | 系统提示错误信息，  系统给用户提供可能的解决办法 |
| 响应度量 | 系统能在2s内确认错误操作并给出反馈 |

**场景7 ：****服务宕机（可用性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 服务器死机 |
| 制品 | 服务集群 |
| 环境 | 服务器正常运行时、开发时、维护时 |
| 响应 | 查明服务器错误  重启服务器 |
| 响应度量 | 服务器重新恢复正常工作<=10min |

**场景8 ：****服务器或客户端移植到其他环境（可移植性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员、维护人员 |
| 刺激 | 服务器或客户端移植到其他环境 |
| 制品 | 服务器、客户端 |
| 环境 | 开发时、维护时、配置时 |
| 响应 | 完成移植所需的修改  测试修改部分  部署系统到新的环境中 |
| 响应度量 | 完成移植所花费的代价不得高于2人/月  移植成本低于重新开发成本的20% |

**场景9 ：系统界面发生变化（可修改性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 开发人员想要修改系统界面 |
| 制品 | 系统 |
| 环境 | 开发时 |
| 响应 | 完成界面变化所需要的修改  测试修改部分的代码  部署到系统的界面层 |
| 响应度量 | 完成界面修改所花费的代价不得高于2人/月  界面修改成本低于重新开发成本的20% |

**场景10 ：数据库崩溃（可用性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 数据库崩溃 |
| 制品 | 数据库系统、错误处理模块 |
| 环境 | 系统正常运行时、测试时 |
| 响应 | 查明崩溃原因  重新恢复数据库 |
| 响应度量 | 重新恢复数据库的时间<=6h  检查崩溃原因并恢复正常使用的成本低于2人/月 |

**场景11 ：服务找不到（可靠性、可用性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 服务找不到 |
| 制品 | 服务器，通信模块、错误处理模块 |
| 环境 | 系统正常运行时、测试时 |
| 响应 | 检查线路故障，若有，排除故障；  检查服务器故障，若有，排除故障 |
| 响应度量 | 检查故障并恢复正常使用的时间<=6h  检查故障并恢复正常使用的成本低于2人/月 |

**场景12 ：****服务分发出现问题（可靠性、可用性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 服务分发出现问题 |
| 制品 | 通信模块、服务集群模块、错误处理模块 |
| 环境 | 系统正常运行时、测试时 |
| 响应 | 分析出现问题的原因  解决问题并恢复正常的分发机制 |
| 响应度量 | 检查故障并恢复正常使用的时间<=6h  检查故障并恢复正常使用的成本低于2人/月 |

**场景13 ：用户进行正常操作（性能）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户 |
| 刺激 | 用户进行正常操作 |
| 制品 | 系统 |
| 环境 | 系统正常运行时 |
| 响应 | 操作流程流畅不卡顿  没有过多的等待时间  多个用户可以同时操作 |
| 响应度量 | <1000人可以同时流畅操作  操作的反馈不超过1s  操作的等待时间不超过2s |

**场景14 ：发生网络闪断（可靠性、鲁棒性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 用户、开发人员 |
| 刺激 | 发生网络闪断 |
| 制品 | 通信模块、错误处理模块 |
| 环境 | 系统正常运行时、测试时 |
| 响应 | 检查网络连接问题  检查通信模块是否有故障  修复故障并恢复网络 |
| 响应度量 | 检查故障并恢复网络的时间<=6h  检查故障并恢复网络的成本低于2人/月 |

**场景15 ：查看服务集群状态信息（可维护性）**

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 查看服务集群状态信息 |
| 制品 | 服务集群状态信息日志 |
| 环境 | 系统正常运行时 |
| 响应 | 系统生成服务集群状态信息日志 |
| 响应度量 | 系统可以生成任意时间段的服务集群状态信息日志  服务集群状态信息日志的正确率达到99.9% |

#### 分解的系统组件

此次迭代进行整体架构设计

#### E:\downloads\C (1).png架构视图

### 迭代二

#### 需求信息

同迭代一

#### 分解的系统组件

选择的是前端服务器模块

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1 | H | H |
| 2 | 场景6 | H | H |
| 3 | 场景7 | M | M |
| 4 | 场景12 | M | M |
| 6 | 场景14 | L | L |
| 7 | 场景15 | M | M |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 关注的质量属性 | 泛化的组件的机制或功能（如错误处理、传输质量、） | 具体机制或功能（如请求重发） |
|  |
| 可用性 | 服务监控 | 和注册中心保持连接 |
| 容错处理 | 服务转发机制 |
| 服务地址失效处理 |
| 性能 | 提高响应速度 | 缓存方式 |
| 缓存更新机制 |
| 系统对任何操作有相应反馈 | 响应时间监测 |
| 高并发 | 建立集群，多点处理 |
| 健壮性 | 从故障中复原 | 备份 |
| 添加硬件资源 |
| 安全性 | 快速检测攻击 | 根据攻击特征识别攻击 |
| 有效抵御攻击 | 设置访问权限，身份验证，访问控制 |
| 系统被攻击后迅速做出有效反馈 | 拒绝访问 |

1. **关注点的候选模式**

（对每个子关注点，列出实现此关注点的可选模式/设计；识别出量化评价这些关注点实现程度好坏的变量，如传输时间、传输成功率，可以用量化指标如具体数字，也可用高中低的程度表示）如

1. **和注册中心保持连接**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成功概率 |
| 1 | 心跳连接 | 高 | 中 | 高 |
| 2 | 在请求到来时ping连接测试 | 低 | 低 | 低 |

1. 选择的模式及理由

选择心跳连接，虽然连接时的时间开销和资源开销不少，不过可以保证和注册中心连接通畅，能够及时地接收到注册中心进行负载均衡判断后的服务器地址信息。方案二无法实时了解与注册中心的连接状态，只有当请求到来时才进行ping连接监测，一旦出现连接问题，还要启动错误处理机制，导致响应时间过长，降低可用性。

1. **服务转发机制**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 准确度 |
| 1 | 直接转发注册中心发回的服务地址 | 低 | 低 | 低 |
| 2 | 对发回的服务地址再确认 | 高 | 高 | 高 |

1. 选择的模式及理由

选择对发回的服务地址再确认，能够在服务器端确认注册中心发回的服务器地址是否有效，如果无效，可以立即进行错误处理，容错性高。而直接转发发回的地址没有考虑异常情况，容易出现可用性的问题。

1. **服务地址失效处理**
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 延时性 | 缺点 |
| 1 | 重新向注册中心发送请求 | 高 | 服务器长时间等待，可能出现响应超时 |
| 2 | 直接返回错误码 | 低 | 影响用户体验 |
| 3 | 如果重复发送3次请求失效，则直接返回错误码 | 中 | 有一定延时 |

1. 选择的模式及理由

选择模式3，是对模式1和模式2的中和，一方面进行部分容错处理，重新请求服务器获得正确的地址，提升用户的体验；另一方面避免服务器多次发送请求仍然出错的异常情况，造成响应超时，让系统在规定的时间内给出合适的反馈。

1. **备份**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 延时 | 更新频率 |
| 1 | 备份部分关键信息  （不包含缓存等） | 中 | 中 | 低 |
| 2 | 备份系统的全部信息 | 高 | 低 | 高 |

1. 选择的模式及理由

选择备份局部信息，因为缓存具有时效性，当负载发生变化后，路由表也会发生很大变化。如要要维护路由表，还需要消耗大量资源去更新，因此备份关键信息即可。

1. **增加硬件资源**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 技术要求 | 提升效果 |
| 1 | 增加服务器数量 | 高 | 高 | 大幅增强响应能力 |
| 2 | 增加服务器内存 | 低 | 低 | 小幅提升重复的数据请求速度 |
| 3 | 增加网络带宽 | 极高 | 低 | 直接提升网络响应速度与能力 |
| 4 | 更换更好的多核的CPU | 适中 | 高 | 中 |

1. 选择的模式及理由

选择增加服务器数量、增加服务内存和增加网络带宽。对项目来说，服务器数量、大内存、高速的处理器是处理高并发需求必须的。更换更好的多核的CPU虽然成本适中，但所要求的多核编程技术极其复杂，提升效果并不显著，所以不选择这种模式。

1. **在客户机设置缓存**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成功概率 |
| 1 | 主动刷新缓存 | 中 | 高 | 高 |
| 2 | 被动刷新缓存 | 高 | 高 | 高 |

1. 选择的模式及理由

选择主动刷新缓存，主要进行过程：当获得请求时，先在缓存表里面查找，若找到对应服务地址，则进行可用性检查后返回客户机；若未找到，则向注册中心发送请求，获得更新后的服务地址，进行可用性检查后存入缓存表，并返回给客户机。缓存表按照一定算法进行维护：删去最近最少使用的服务地址。如果选用被动方式，即一旦注册中心的服务地址发生改变，就通知更新缓存，则缓存表需要判断是否为最近经常使用，来决定要不要保存，会花费较多时间和资源，降低响应速度。

1. **响应时间监测**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成功概率 |
| 1 | 错误码提示用户刷新连接 | 中 | 中 | 高 |
| 2 | 在一定时间内自动刷新重连 | 中 | 高 | 中 |
| 3 | 组合模式 | 中 | 高 | 高 |

1. 选择的模式及理由

选择组合模式，即：在预定的响应时间内没有响应的情况下，系统返回正在加载的提示，后台尝试自动刷新重连。这种方式对易用性的支持也比较友好，模式一给用户一种系统崩溃的体验，模式二没有给用户任何提示，两者都没有很好地满足易用性。

1. **建立集群**
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 基于业务划分 | 业务区分度较高，服务器多台处理效率高 | 同一种业务的请求也可能出现高并发 |
| 2 | 基于客户端服务器地理位置划分 | 连接时间较少，增强了用户体验和易用性，很好地避免了单点高并发 | 解析地址可能花费较多资源 |

1. 选择的模式及理由

选择地理位置划分模式，1将短时内集中收到的用户请求按照业务划分，通过负载均衡分摊到多个请求解析转发器，2按照客户端请求来自的地理位置，寻找最近的相对空闲的请求解析转发器，这种模式出现单点高并发的几率较小，很好地解决了高并发问题，并且提高了响应速度，增强了易用性

1. **根据攻击特征识别攻击**
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 根据已有的攻击特征识别出新的攻击 | 对于被记录的攻击十分有效 | 要时刻更新攻击库，对新的攻击类型需要较长的检测时间 |
| 2 | 根据历史被攻击的 记录识别新的攻击 | 对历史被攻击的攻 击类型的识别十分 有效 | 对新的攻击类型需 要较长的检测时间 |

1. 选择的模式及理由

选择根据已有的攻击特征识别出新的攻击，因为相对于根据历史攻击来识别，可以拥有更大的攻击类别库，识别攻击的能力强，同时相对其他方法也比较方便实现。

1. **拒绝访问**
2. 候选模式

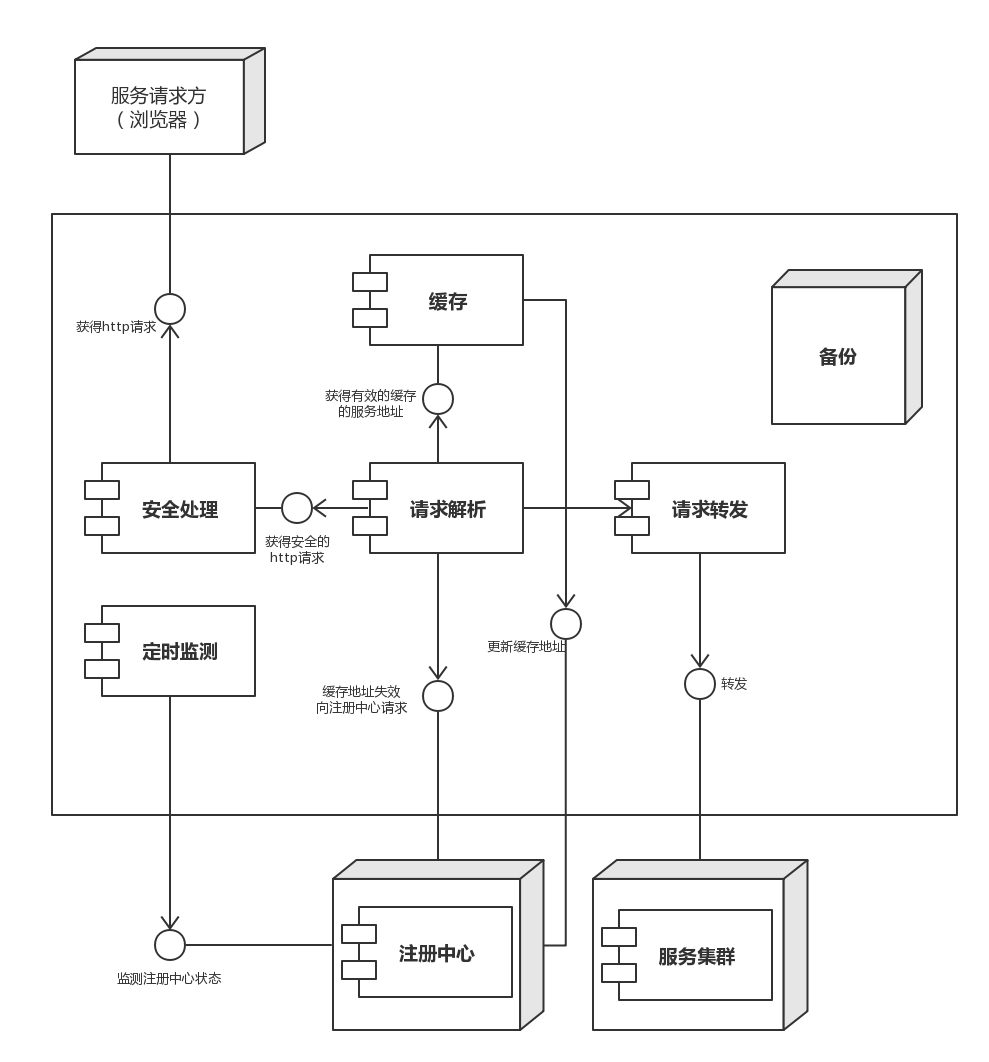
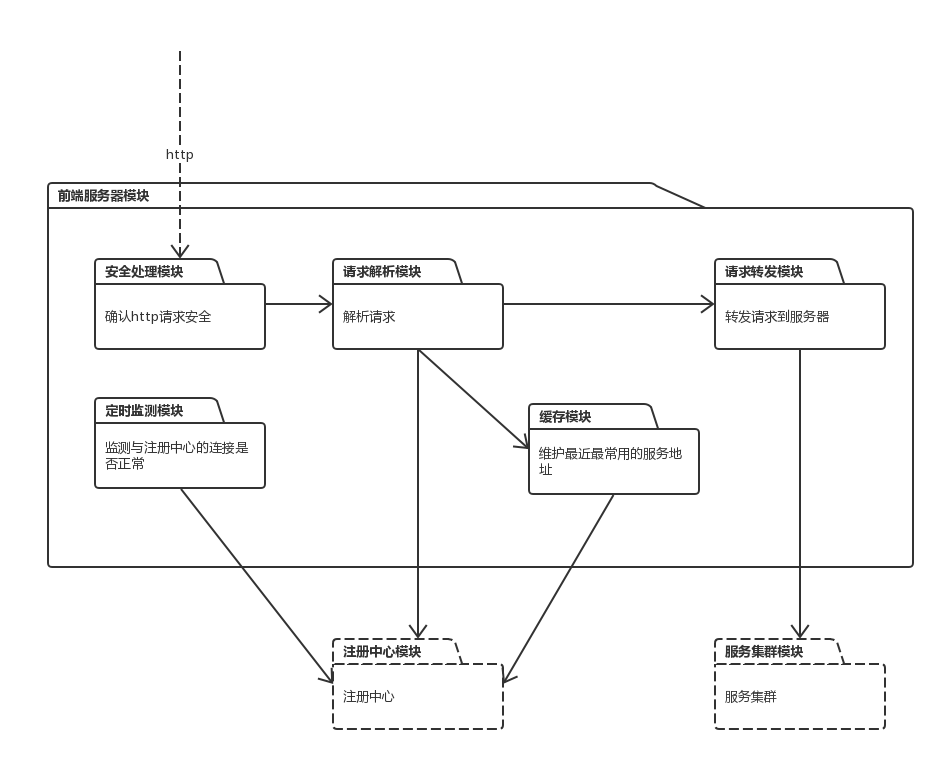
|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 断开连接 |  | 只能抵挡一次攻击 |
| 2 | 黑名单 | 长时间保护系统 | 可能出现失误 |

1. 选择的模式及理由

选择黑名单模式，系统用户群体较大，不能容忍单个 ip 的不停攻击，断开连接会导致正常用户无法使用系统，但考虑到误操作的可能性，可以设置黑名单有效时间。

1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 和注册中心保持连接 | 心跳连接 | 场景1 |
| 2 | 服务转发机制 | 对发回的服务地址再确认 | 场景1，场景12 |
| 3 | 服务地址失效处理 | 向注册中心再次发送请求，如果3次请求失败，则直接返回错误码 | 场景6 |
| 4 | 备份 | 备份部分关键信息 | 场景8 |
| 5 | 增加硬件资源 | 选择增加服务器数量、增加服务内存和增加网络带宽 | 场景15 |
| 6 | 客户机设置缓存 | 主动刷新缓存 | 场景1 |
| 7 | 响应时间监测 | 组合模式 | 场景6 |
| 8 | 建立集群 | 基于地理位置分发请求 | 场景1，场景15 |
| 9 | 根据攻击特征识别攻击 | 根据已有的攻击特征识别出新的攻击 | 场景7 |
| 10 | 拒绝访问 | 黑名单 | 场景7 |

1. **架构视图**
2. **C&C视图**
3. **Module视图**
4. **评估**

此次设计没有冲突

#### 分配职责

1. 安全处理模块：用于确认客户机来的http请求是安全的，发送给请求解析模块。
2. 请求解析模块：用于解析请求的服务地址，并返回相应的服务给客户端。
3. 缓存模块：用于在本地保存并维护最近最常用的服务IP地址，提高响应速度。
4. 定时监测模块：用于检测与注册中心的连接是否正常。
5. 请求转发模块：用于转发请求到对应的服务器，并返回服务。

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 请求解析模块 | 缓存模块 | LookInCache | 服务标识符 |
| 缓存模块 | 注册中心模块 | Update | 服务标识符，新IP地址 |
| 请求解析模块 | 注册中心模块 | Re-request | 服务标识符 |
| 请求解析模块 | 请求转发模块 | Deliver | 服务IP地址 |
| 请求转发模块 | 服务集群模块 | Server | 服务IP地址 |

### 迭代三

#### 需求信息

同迭代一

#### 分解的系统组件

此次迭代分解了业务集群模块

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1  >1000用户同时购买一件商品 | H | H |
| 2 | 场景5  未登录用户进行支付操作 | H | M |
| 3 | 场景7  服务宕机 | M | M |
| 4 | 场景8  服务器或客户端移植到其他环境 | L | L |
| 5 | 场景13  用户进行正常操作 | H | H |
| 6 | 场景15  查看服务集群状态信息 | M | H |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 性能、可靠性 | 业务处理能力 | 数据读取能力 |
| 数据计算能力 |
| 请求速率控制 |
| 安全性 | 阻止未授权用户进行高权限操作 | 未授权识别 |
| 权限拒绝处理 |
| 可用性 | 服务器宕机容错 | 宕机监控 |
| 宕机恢复 |
| 可移植性 | 系统可移植 | 服务器可移植 |
| 可维护性 | 服务监控 | 业务计算量监控 |
| 服务器工作状态监控 |

1. **关注点的候选模式**
2. 数据读取能力
3. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 读取性能 | 资源消耗 | 开发成本 | 并发性能 |
| 1 | 增加数据库性能 | 高 | 大 | 高 | 中 |
| 2 | 增加专用信息检索服务节点 | 高 | 大 | 中 | 中 |

1. 选择的模式及理由

由于数据库容易成为数据读取的瓶颈，且增加数据库的性能会大大增加开发成本（设备、数据库系统），而增加专用的信息检索节点，可以更灵活、自由、快速的获得所需要的数据。所以这里选择模式2.

1. 数据计算能力
   1. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 可伸缩 | 单个请求计算速度 | 大量请求计算速度 |
| 1 | 增加单机设备性能 | 高 | 低 | 高 | 低 |
| 2 | 增加服务节点 | 中 | 高 | 中 | 中 |
| 3 | 增加单机设备性能并增加服务节点 | 高 | 高 | 高 | 高 |

* 1. 选择的模式及理由

在电商场景下，业务请求不是每时每刻都是大量的，模式1在请求量较低时，资源浪费太大，在应对大量的计算请求时也没有良好的计算性能；模式3在目前的条件下过于浪费资源；这里选择模式2。

1. 请求速率控制
   1. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 并发量 | 成本 | 用户任务完成度 | 请求完成速度 |
| 1 | 增加消息队列控制消息发送速率 | 高 | 高 | 高并发和低并发时均很高 | 低并发时高，高并发时可能低 |
| 2 | 当单位时间内的请求到一定量时拒绝请求 | 中 | 中 | 低并发时高，高并发时低 | 低并发时高，高并发时部分用户请求不能得到完成 |

* 1. 选择的模式及理由

在电商场景下，必须要考虑到在某些情况下的大量用户同时操作，于是系统必须要有很高的并发性能，即便是要付出一定的成本，于是这里选择模式1。

1. 未授权识别
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 可执行性 | 成本 | 安全性 |
| 1 | 使用签名算法 | 高 | 低 | 中 |
| 2 | 发布证书 | 中 | 高 | 高 |

* 1. 选择的模式及理由

以上两种模式在电商环境中均有各自优势，于是这里把两种模式均列为候选模式

1. 未授权拒绝处理
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 安全性 | 可执行性 |
| 1 | 拒绝处理、返回错误码 | 低 | 中 | 高 |
| 2 | 拒绝处理、记录日志 | 低 | 中 | 高 |
| 3 | 拒绝处理、记录IP、IP黑名单 | 中 | 高 | 中 |

* 1. 选择的模式及理由

由于电商环境涉及到具体的支付业务，需要的安全性较高，可以在牺牲一定的成本下，换取较高的安全性，于是这里选择模式3。

1. 宕机监控
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时延 | 资源消耗 | 监控力度 |
| 1 | 心跳 | 低 | 中 | 高 |
| 2 | Ping/echo | 中 | 低 | 中 |

* 1. 选择的模式及理由

在分布式环境下，服务计算节点较多，相对来说宕机一台或多台服务器对用户的使用没有较大的影响，但是由于监控中心需要对大量的服务器进行监控，所以必须选择资源消耗较低的模式来保证监控中心的稳定运行。这里选择模式2。

1. 宕机恢复
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成功率 | 宕机时间 | 故障排查率 |
| 1 | 记录错误、服务自动重启 | 低 | 中 | 低 |
| 2 | 报告错误、服务手动重启 | 高 | 高 | 高 |

* 1. 选择的模式及理由

在分布式环境下，一台服务器的暂时宕机不影响整个系统的正常工作，但是服务器宕机的原因却必须要被维护人员所了解，以便此类问题再次发生，于是这里选择模式2。

1. 服务器可移植
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 移植时间 | 可完成功能 |
| 1 | 使用跨平台语言 | 低 | 低 | 部分受限 |
| 2 | 设计专用的运行支撑环境 | 高 | 中 | 不受限 |

* 1. 选择的模式及理由

由于服务器的可移植性不具有太高的优先级，且移植场景发生的频率较低，所以尽可能选择成本较低的模式，于是我们采取模式1。

1. 业务计算量监控
   1. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 资源消耗 | 信息即时性 |
| 1 | 设定每隔单位时间向监控服务器发送消息 | 中 | 中 |
| 2 | 设定每到一定处理数量向监控服务器发送消息 | 中 | 中 |
| 3 | 每处理一条服务向监控服务器发送消息 | 高 | 高 |

* 1. 选择的模式及理由

由于分布式服务节点数量较多，监控中心应当能监控大量服务节点的处理信息，于是应当选择使监控中心资源消耗较低的模式；在系统中对信息的即时性要求不是特别高，所以这里选择模式1或2

1. 服务器工作状态监控
   1. 候选模式

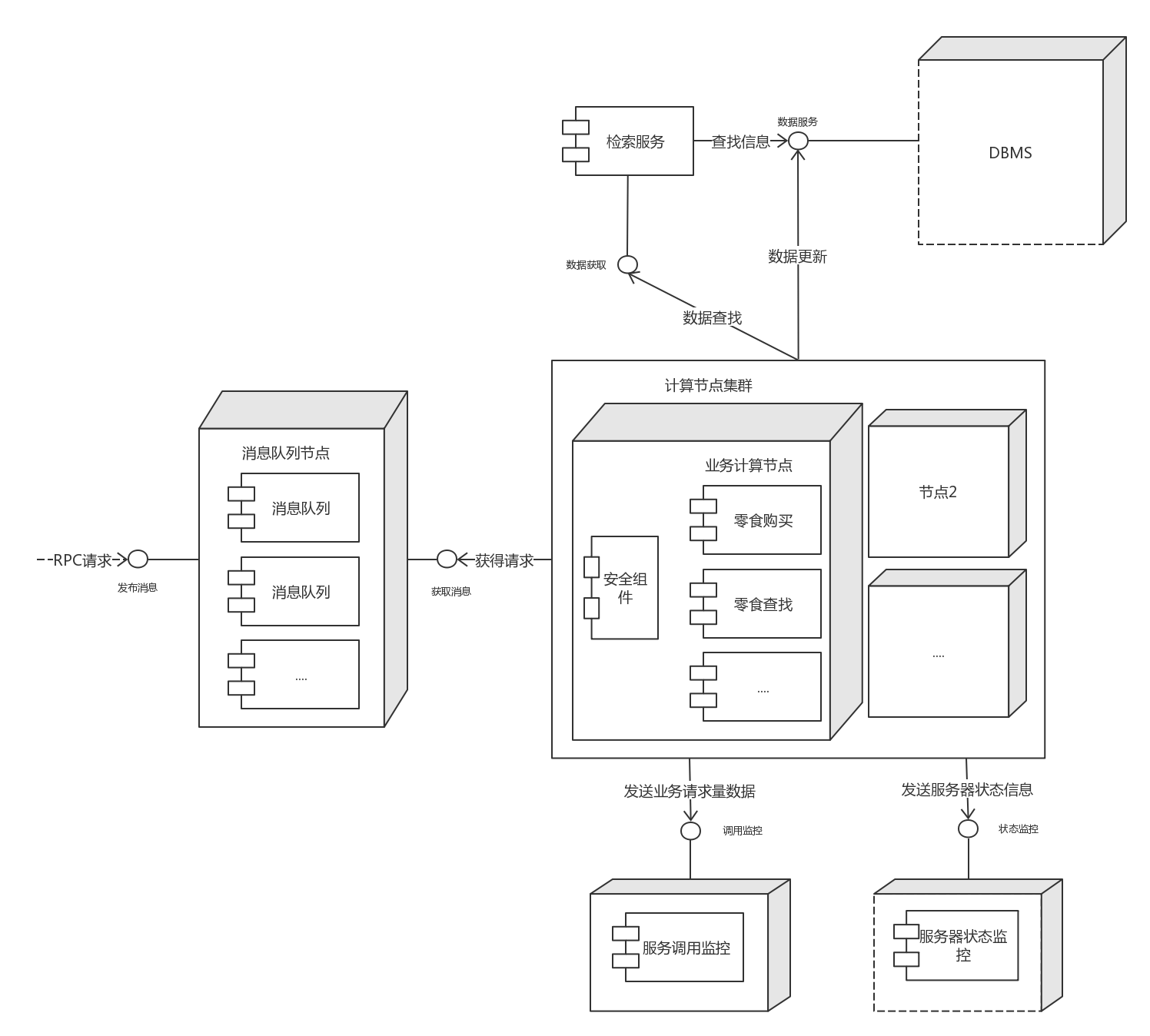
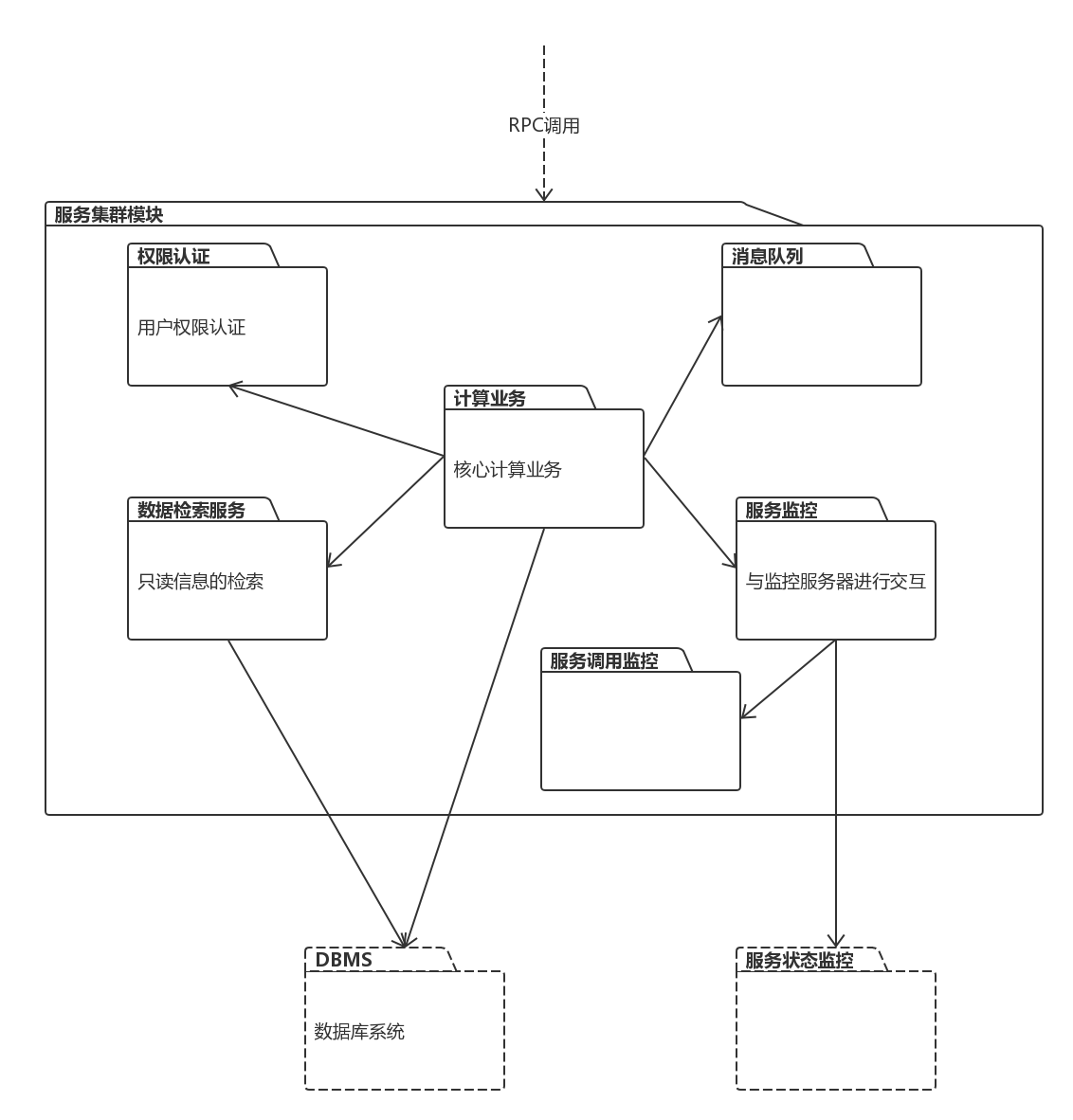
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 服务器资源消耗 | 监控中心资源消耗 | 最大监控服务器集群规模 | 信息更新速率 |
| 1 | 服务器定时向监控服务器发起自身状态信息 | 高 | 中 | 高 | 中 |
| 2 | 监控服务器定时询问服务器状态信息 | 中 | 高 | 中 | 高 |

* 1. 选择的模式及理由

由于服务计算节点是特别巨大的，因此监控中心必须能对大规模的服务集群进行监控，于是这里选择的是模式1。

1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 数据读取能力 | 增加专用信息检索服务节点 | SC13 |
| 2 | 数据计算能力 | 增加服务节点 | SC1、SC3 |
| 3 | 请求速率控制 | 增加消息队列控制消息发送速率 | SC1 |
| 4 | 未授权识别 | 签名算法/证书 | SC5 |
| 5 | 未授权拒绝处理 | 拒绝请求、记录IP、IP黑名单 | SC5 |
| 6 | 宕机监控 | Ping/Echo | SC7 |
| 7 | 宕机恢复 | 报告错误、服务手动重启 | SC7 |
| 8 | 服务可移植 | 使用跨平台语言 | SC8 |
| 9 | 业务计算量监控 | 每到单位时间/一定处理数量服务器向监控中心发送消息 | FC6、SC15 |
| 10 | 服务器工作状态监控 | 服务器定时向监控服务器发起自身状态信息 | SC15 |

1. **架构视图**
2. C&C视图
3. Module视图
4. **评估**

此次迭代没有发现矛盾

#### 分配职责

1. **消息队列：**用于存放发送过来的请求，为了应对当请求发送过多、业务计算节点无法负载、导致计算几点有宕机的风险的情况。将请求暂存于消息队列里，根据业务计算节点的处理速度进行请求的获取，可以提高系统在高并发的情况下的可用性。
2. **安全组件：**用于验证用户是否处于认证状态，保证用户信息、财产的安全性。
3. **业务计算节点：**核心的业务处理节点。整个集群有大量的业务计算节点，每个节点也有一种或多种具体的业务计算。用户发起的请求最终均是在此处进行处理，多节点、节点多服务的设计可以让系统一直处于高可用的状态。
4. **服务调用监控：**用于监控整个系统的请求流量（如单位时间处理的购买请求等），可用于对系统的服务计算能力进行调整、系统资源分配等决策上。
5. **服务状态监控：**用于监控服务器状态（工作状态、负载等）。
6. **数据检索服务：**对只读性数据进行查找（如零食信息），在正常使用场景下，大量的操作是浏览商品信息等单一的信息获取，数据检索服务用其专门的数据获取操作保证数据获取能更快速、更高效。

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 服务前端 | 消息队列 | 消息发布 | 对应服务需要的参数信息 |
| 计算节点 | 消息队列 | 消息获取 | 计算节点自身的服务类型 |
| 计算节点 | 服务调用监控 | 调用监控 | 当前时间、服务调用数 |
| 计算节点 | 服务状态监控 | 状态监控 | 当前时间、服务状态信息 |
| 计算节点 | 检索服务 | 数据查找 | 查找条件 |
| 计算节点 | DBMS | 数据更新 | 更新的数据信息 |
| 检索服务 | DBMS | 数据查找 | 查找条件 |



### 迭代四

#### 需求信息

同迭代一

#### 分解的系统组件

针对注册中心模块进行分解、分析与设计，这个模块主要负责服务的分发，需要保证服务的快速、稳定、高并发以及整个系统的负载均衡，对系统的运行至关重要。

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1  >1000用户同时购买一件商品 | H | H |
| 2 | 场景2  需要新服务加入系统 | M | H |
| 3 | 场景3  需要增减服务计算能力 | M | H |
| 4 | 场景7  服务宕机 | H | H |
| 5 | 场景11  服务找不到 | M | M |
| 6 | 场景13  用户进行正常操作 | H | H |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 可用性 | 服务分发能力 | 服务接收请求的负载均衡 |
| 维护服务分发的负载均衡的能力 |
| 服务分发的负载均衡 |
| 可靠性 | 服务更新能力 | 注册节点同步能力 |
| 可扩展性，可伸缩性 | 添加或减少服务节点或注册节点对系统的影响 | 服务注册节点的重新分布能力 |
| 服务注册表的重新构建能力 |

1. **关注点的候选模式**
2. **服务接收请求的负载均衡**
3. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 读取性能 | 资源消耗 | 开发成本 | 并发性能 |
| 1 | 单点服务器处理请求分发 | 低 | 低 | 低 | 低 |
| 2 | F5硬件处理请求分发 | 中 | 低 | 低 | 中 |
| 3 | 利用Zab协议建立注册集群 | 高 | 高 | 高 | 高 |

1. 选择的模式及理由

无论是模式1还是模式2，在数据量极具集中访问的情况下都会有单点的瓶颈问题。Zab协议利用服务器集群的动态选主和同步流程，让服务器动态的分配接收请求的服务器，让请求平均分布；所以这里选择模式3

1. **服务分发的负载均衡**
   1. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 业务处理性能 | 资源消耗 | 开发成本 | 并发性能 |
| 1 | 平均分配服务节点 | 低 | 低 | 低 | 低 |
| 2 | 维护注册表记录负载信息 | 高 | 高 | 中 | 高 |

* 1. 选择的模式及理由

模式1平均分配的服务节点，实现方式非常简单，但是由于每种服务的消耗不一样可能导致某些服务需要额外的资源（时间、空间），从而造成不同服务器上的负载不一。利用注册表动态的更新服务器的负载信息可以实现动态的负载均衡信息。此处选择模式2.

1. **维护服务分发的负载均衡的能力**
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 可靠性 | 耦合度 |
| 1 | 业务服务器更新注册中心注册表 | 中 | 高 | 低 |
| 2 | 注册中心定时检测业务服务器负载 | 中 | 中 | 中 |
| 3 | 注册中心根据接收请求数维护注册表 | 低 | 低 | 低 |

* 1. 选择的模式及理由

注册中心需要时刻保证将合适的服务节点返回给前端服务器，模式2使得注册中心的耦合度增高，模式3虽然保证解耦性，但是单方面有请求数量来维护注册表并不能得知服务结束的时刻，导致注册表的不可靠。模式1中业务服务器要实现注册中心规定的更新接口，提高了一定的成本，但是主动更新负载可以保证高可靠性而且保证了注册中心的低耦合，所以此处选择模式1。

1. **注册节点同步能力**
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 速度 | 成本 | 高可用 |
| 1 | CP原则模式 | 慢 | 低 | 低 |
| 2 | AP原则模式 | 快 | 低 | 高 |
| 3 | Zab协议栈模式 | 快 | 中 | 中 |

* 1. 选择的模式及理由

CAP原则，Consistency（一致性），Availability（可用性），Partition tolerance（分区容错性）中放弃高可用追求一致性，例如模式1，可以达到高可靠的要求但是不能保证高可用。放弃一致性，保证高可用，例如模式2，可以达到高可用，但是不能保证高可靠。Zab协议栈模式达到了一种中间的妥协，但是也不能完全保证时刻可用、可靠。这里的选择需要根据实际的业务要求来进行选择，如果对一致性要求高就选择CP模式，需要高可用就选择AP模式，Zab协议栈模式也可以作为一种选择。

1. **服务注册节点的重新分布能力**
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 速度 | 成本 | 高可用 |
| 1 | 暂停注册中心重新分配 | 慢 | 低 | 低 |
| 2 | Zab协议栈模式 | 快 | 低 | 高 |

* 1. 选择的模式及理由

暂停注册中心会带来暂时的不可用，而Zab协议栈利用恢复模式和广播模式保证服务的高可用，新注册节点服务器加入后自动进入learner群组，在进行下一轮恢复模式（选主）前暂时不可用，删除注册节点服务器后，进入恢复模式，利用fast paxos算法进行重新选主。此处选择模式2。

1. **服务注册表的重新构建能力**
   1. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 速度 | 成本 | 高可用 |
| 1 | 业务服务器更新注册中心注册表 | 快 | 高 | 高 |
| 2 | 注册中心定时检测业务服务器状态 | 慢 | 中 | 高 |

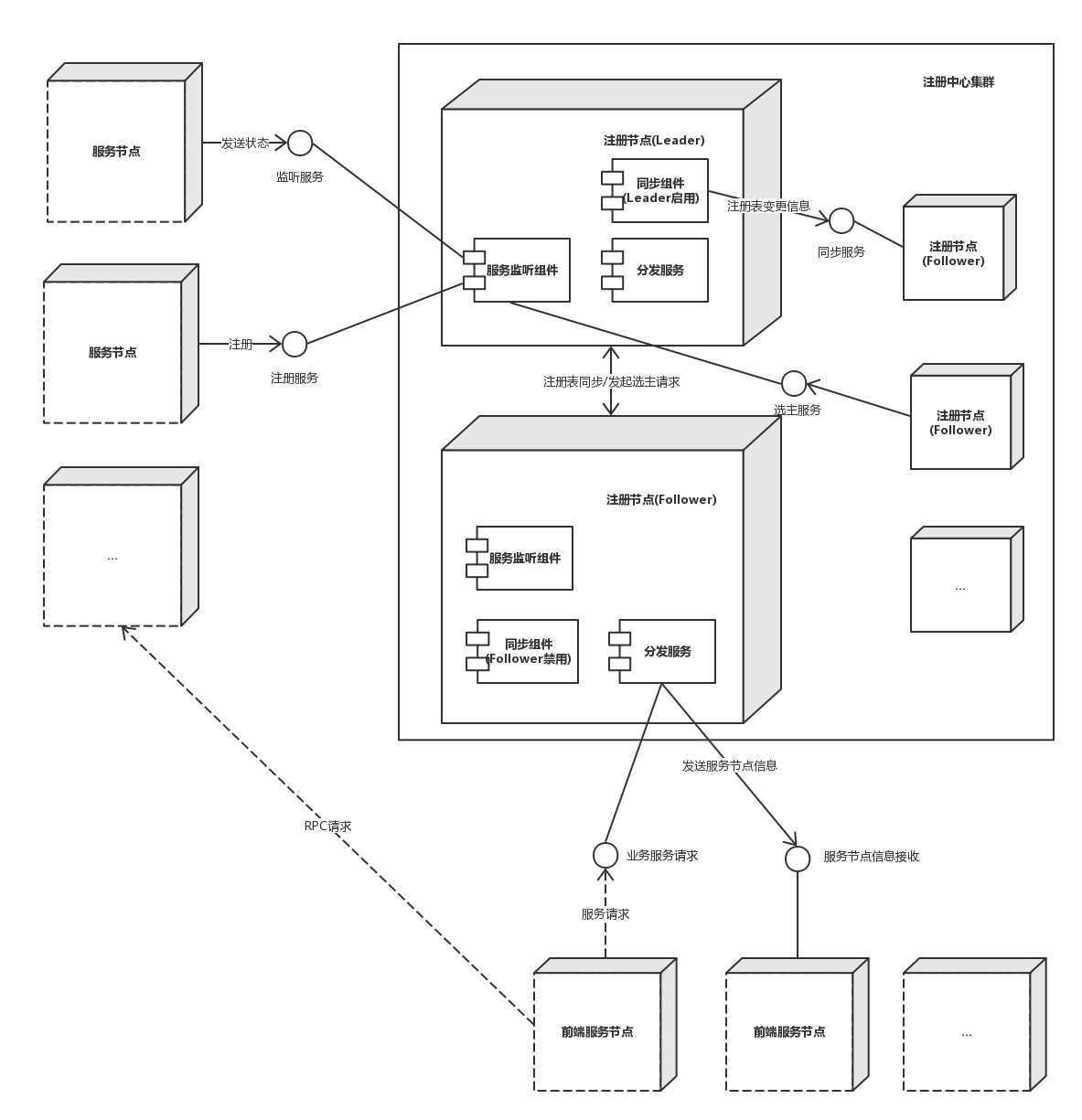
* 1. 选择的模式及理由

模式1中业务服务器需要实现注册中心的接口，并且在初始化或在离开时向注册中心发送信息来保证注册表的更新，这样对业务服务器而言，成本稍高，但新加入的服务器能够在第一时间得到使用。模式2中业务服务器不需要实现注册中的接口，但是注册中心需要定时检测业务服务器集群，这样做成本比较低但是新加入的服务器可能不能立刻进行业务协作。这两种方案各有利弊，在真实开发场景中，如果业务服务器的添加/删除频率比较低，就选择模式1，如果业务服务器的添加/删除频率比较高，就选择模式2。

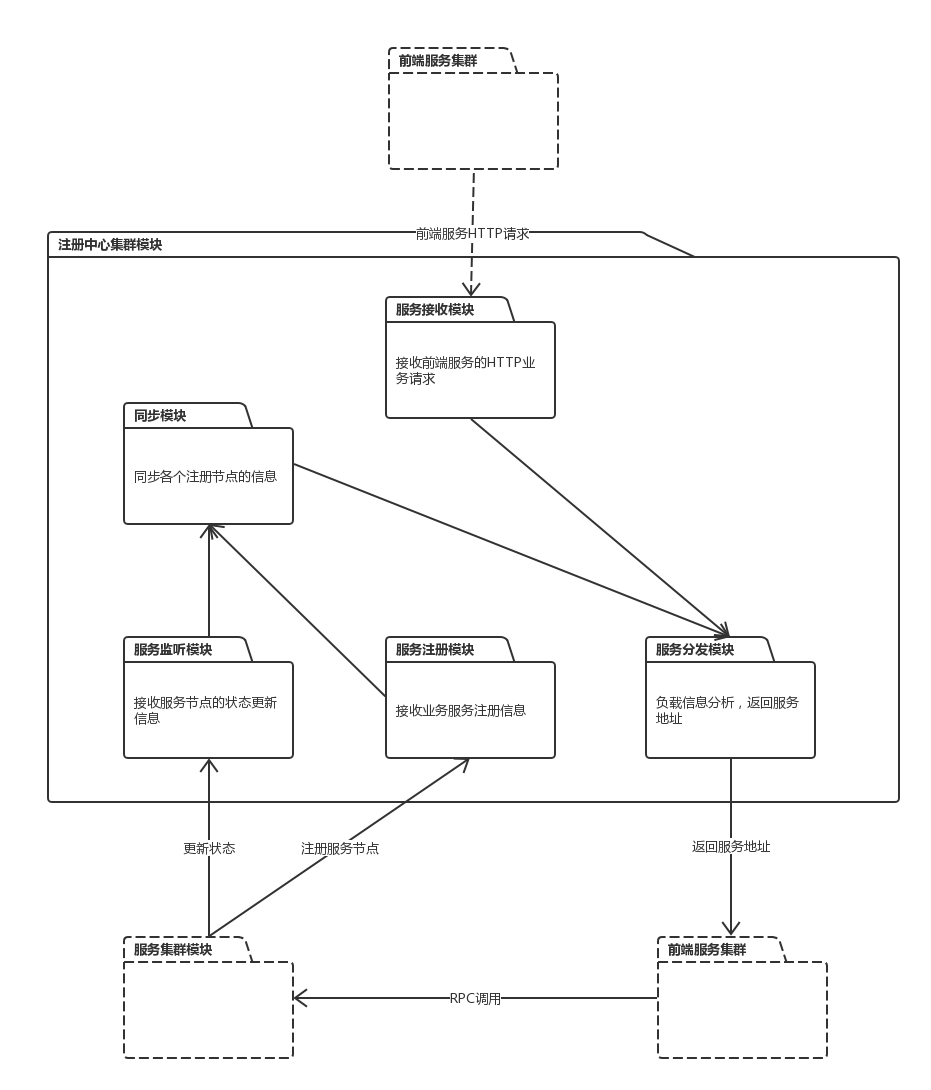
1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 服务接收请求的负载均衡 | 利用Zab协议建立注册集群 | SC13 |
| 2 | 服务分发的负载均衡 | 维护注册表记录负载信息 | SC 1、SC 3 |
| 3 | 维护服务分发的负载均衡的能力 | 业务服务器更新注册中心注册表 | SC 1 |
| 4 | 注册节点同步能力 | 根据不同业务需求选择：  CP原则模式/AP原则模式/Zab协议栈模式 | SC2、SC7、SC11 |
| 5 | 服务注册节点的重新分布能力 | Zab协议栈模式 | SC 5 |
| 6 | 服务注册表的重新构建能力 | 业务服务器更新注册中心注册表 / 注册中心定时检测业务服务器状态 | SC 5 |

1. **架构视图**
   1. **C&C视图**



* 1. **Module视图**



1. **评估**

本次迭代没有发现矛盾

#### 分配职责

1. **服务接收模块：**用于接收前端服务节点发送的服务请求信息，为了应对单点问题，服务接收模块有多个注册中心节点，在真实环境中不同地区的不同请求会发往不同注册中心节点的服务接收模块，提高系统在高并发情况下的可用性。
2. **服务分发模块：**用于负债信息分析并且返回服务地址，为了实现负载均衡，该模块会接受同步模块的注册节点更新信息并维护不同服务节点的负载表，在服务接收模块完成工作后该模块进行负载的查询和选择合适服务节点返回，提高了系统的可用性，平衡了服务节点的压力。
3. **同步模块：**用于同步注册中心集群中各个注册节点的服务负载表信息等，让服务发布模块能够选择负载合适的服务节点信息进行返回。
4. **服务监听模块：**用于监听来自服务节点的状态更新信息，服务节点主动进行负载信息等的发送，监听模块记录并通知同步模块对负载注册表进行更新，是负载均衡解决方案中重要的环节。
5. **服务注册模块：**用于接收业务服务节点的初始化注册信息，并向同步模块发送注册信息，是负载均衡解决方案中重要的环节。

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| **源组件** | **目标组件** | **接口名** | **参数** |
| 前端服务节点组件 | 分发服务组件 | 业务服务请求 | 需要的服务信息 |
| 分发服务组件 | 服务前端 | 服务节点信息接收 | 服务节点信息 |
| 服务节点组件 | 服务监听组件 | 监听服务 | 服务节点状态（负载信息等） |
| 服务节点组件 | 服务监听组件 | 注册服务 | 服务节点信息 |
| 同步组件 | 服务监听组件 | 同步服务 | 注册表变更信息 |
| 服务监听组件 | 同步组件 | 选主服务 | 选主请求 |



### 迭代五

#### 需求信息

同迭代一

#### 分解的系统组件

选择系统的数据存储模块进行分解，数据存储模块主要承担职责如下：

* 存储用户信息、商品信息、交易数据等
* 存储商品图片、大段商品介绍等静态资源
* 对外提供统一访问接口
* 内部监控各存储单元（主机）的工作情况

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1：（>1000）用户同时购买同一件商品 | H | M |
| 2 | 场景3：需要增减服务计算能力 | M | H |
| 3 | 场景9：服务器移植到其他环境 | L | M |
| 4 | 场景11：数据库崩溃 | H | L |

#### 为ASR进行设计

1. **设计关注点**

静态资源：商品图片及大段文字介绍（包括商家详情、商品详情等）

存储单元：存储数据的主机

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 可靠性 | 数据备份 | 数据库备份 |
| 文件系统备份 |
| 性能 | 静态资源存储 | 图片存储 |
| 大段文字存储 |
| 数据库存储 | 数据存取方式 |
| 可伸缩性 | 负载均衡 | 负载均衡策略 |
| 可移植性 | 对外接口实现 | 数据库操作实现 |
| 可用性 | 错误处理 | 存储单元宕机监控及处理 |

1. **关注点的候选模式**
2. **数据库备份**
3. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成功概率 |
| 1 | 实时备份 | 高 | 高 | 高 |
| 2 | 固定时间间隔备份 | 中 | 中 | 中 |

1. 选择的模式及理由

选择实时备份。

为了保证系统的可靠性，需要对数据库中的业务数据进行备份，以防数据库崩溃造成无法挽回的损失。由于业务数据直接关系到用户的财产安全，所以选择实时备份模式，虽然相对会消耗更多资源，但是用户的安全始终是第一位的。

1. **文件系统备份**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成功概率 |
| 1 | 实时备份 | 高 | 高 | 高 |
| 2 | 固定时间间隔备份 | 中 | 中 | 中 |

1. 选择的模式及理由

选择固定时间间隔备份。

相对业务数据而言，访问静态资源所需的系统开销更大，但重要性较低。若采用实时备份，将会造成大量的不必要的系统开销。另一方面，即使文件系统崩溃，大多数情况下也可以通过磁盘恢复得到原有数据。

1. **图片存储**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成本 |
| 1 | 文件系统存储 | 中 | 低 | 低 |
| 2 | 图形数据库存储 | 低 | 中 | 高 |

1. 选择的模式及理由

选择文件系统存储。

相较于极其稳定和成熟的文件系统存储技术，图形数据库技术尚不成熟，无法保证高可用和稳定性。另一方面，文件系统存储所需的磁盘空间更少，可以存储更多图片。

1. **大段文本存储**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 成本 |
| 1 | 文件系统存储 | 中 | 低 | 低 |
| 2 | 数据库存储 | 低 | 中 | 高 |

1. 选择的模式及理由

选择数据库存储。

大段文本若存储在数据库中，因为各字段长短不一，会造成极大的资源浪费，同时会降低数据库访问性能，而使用文件系统存储则不会出现类似问题。

1. **数据存储方式**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 写入开销 | 读取开销 | 是否需要额外备份 |
| 1 | 分库分表存储 | 低 | 高 | 是 |
| 2 | 读写分离 | 高 | 低 | 否 |

1. 选择的模式及理由

选择读写分离。

采用分库分表策略，数据写入开销低，但读取开销高，尤其当需要多表查询时，会极大地影响性能；同时，为了保证系统的可用性，需要对各数据库进行备份，所需要的资源开销和成本都非常巨大。

使用读写分离策略，虽然数据写入时会触发数据同步操作，但读取开销低，同时不需要对数据进行额外备份，可以大幅节约资源和成本。

1. **负载均衡策略**
2. 候选模式

主机列表：负载均衡服务器维护注册主机列表，表中存储主机地址及当前负载量，每次有新的请求时，寻找负载最低的主机提供服务。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 动态添加/减少主机 |
| 1 | 固定主机列表 | 低 | 低 | 不能实现 |
| 2 | 动态更新主机列表 | 低 | 中 | 可以实现 |

1. 选择的模式及理由

选择动态更新主机列表。

固定主机列表响应速度快，消耗资源少，但可伸缩性较差，且无法动态添加/减少主机，灵活性较差。动态更新主机列表模式可极大地提高系统的可伸缩性和灵活性。

1. **数据库操作实现**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 启动时间开销 |
| 1 | 静态sql语句 | 低 | 低 | 低 |
| 2 | 根据数据源动态生成sql语句 | 低 | 中 | 高 |

1. 选择的模式及理由

选择根据数据源动态生成sql语句。

采用静态sql语句实现的可移植性差，比如数据库从Mysql 移植到 Oracle后，所有数据处理逻辑都要重写。而根据数据源动态生成sql语句的实现仅会在服务器启动时慢于静态sql语句，此外性能相差无几，但无需在迁移数据库时重写代码。

1. **存储单元宕机监控及处理**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 | 监控即时性 |
| 1 | 固定时间间隔监控 | 中 | 中 | 高 |
| 2 | 请求无响应时报错 | 低 | 低 | 低 |

1. 选择的模式及理由

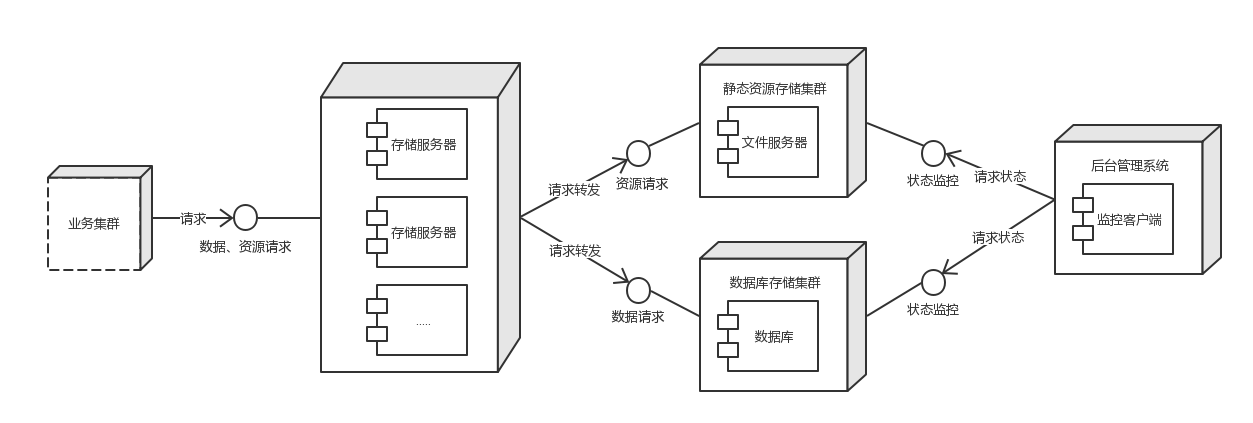
选择固定时间间隔监控各存储单元，虽然会产生额外的时间开销和资源开销，但所产生的开销由一台单独的服务器承担，各存储单元只需消耗非常少的资源，不会对系统产生大的性能影响。若采用请求无响应时报错，则无法及时发现宕机的存储单元，对系统可用性造成影响。

1. **候选模式与对应ASR**

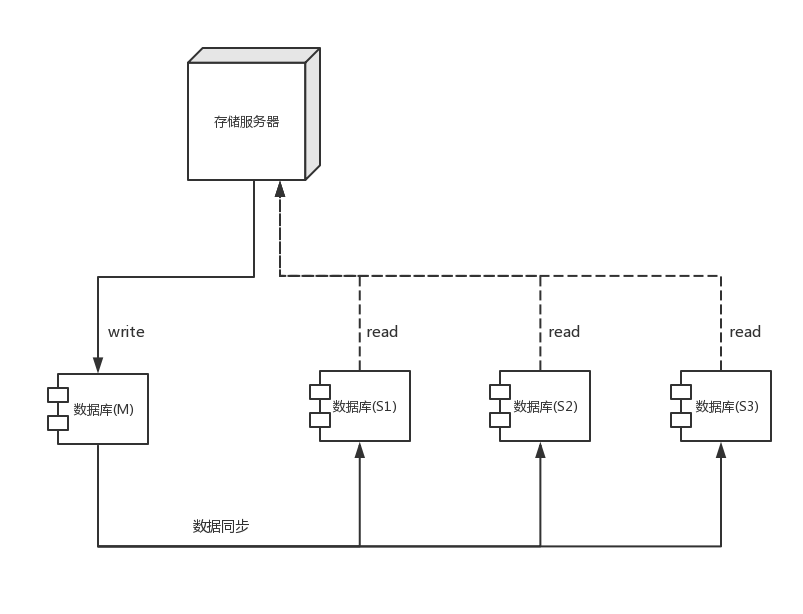
|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 数据库备份 | 实时备份 | 场景11：数据库崩溃  可靠性 |
| 2 | 文件系统备份 | 固定时间间隔备份 | 场景11：数据库崩溃  可靠性 |
| 3 | 图片存储 | 文件系统存储 | 场景1：（>1000）用户同时购买同一件商品  性能 |
| 4 | 大段文字存储 | 文件系统存储 | 场景1：（>1000）用户同时购买同一件商品  性能 |
| 5 | 数据存取方式 | 读写分离 | 场景1：（>1000）用户同时购买同一件商品  性能 |
| 6 | 负载均衡策略 | 动态更新主机列表 | 场景3：需要增减服务计算能力  可伸缩性 |
| 7 | 数据库操作实现 | 根据数据源动态生成sql语句 | 场景9：服务器移植到其他环境  可移植性 |
| 8 | 存储单元宕机监控及处理 | 固定时间间隔监控 | 场景11：数据库崩溃  可用性 |

1. **架构视图**

**1）C&C视图**



**2）业务数据库示意图**



1. **评估**

本次设计使用ADD方法对数据存储模块进行分解、细化，在考虑各质量属性的同时，兼顾了系统的实际成本（数据库存储），并在系统性能、可伸缩性、可移植性、可靠性及可用性之间做了权衡，较好的兼顾了上述各质量属性。

#### 职责

1. **存储服务器**：用于接收服务集群的请求，并将请求分发到静态资源存储集群及业务数据库集群。
2. **静态资源存储集群：**存储图片、大段文本等不宜使用数据库存储的资源，接收存储服务器分发的请求，对请求做出相应。
3. **业务数据库集群：**存储业务数据，为保证性能及可用性，采用读写分离的模式。
4. **后台管理系统：**定时监控各存储主机的工作情况，发现宕机的主机，并及时发出警告。
5. **监控客户端：**显示各存储主机的工作情况，接收管理系统发出的警告消息。

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 服务集群 | 存储服务器 | 数据、资源请求 | 对应数据请求的参数 |
| 存储服务器 | 静态资源存储集群 | 资源请求 | 对应资源名称 |
| 存储服务器 | 业务数据库集群 | 数据请求 | 对应数据请求的参数 |
| 后台管理系统 | 静态资源存储集群 | 状态监控 | 当前服务状态信息 |
| 后台管理系统 | 静态资源存储集群 | 状态监控 | 当前服务状态信息 |

### 迭代六

#### 需求信息

同迭代一

#### 分解的系统组件

针对通信模块进行分解、分析与设计，这个模块主要负责整个系统客户端与服务端之间网络的传输，需要保证服务的快速、稳定、高并发，对系统的运行至关重要。

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1：(>1000)用户同时购买同一件商品 | 高 | 高 |
| 2 | 场景2：新服务加入系统 | 中 | 高 |
| 3 | 场景3：需要增减服务计算能力 | 中 | 高 |
| 4 | 场景5：未登录用户进行支付操作 | 高 | 中 |
| 5 | 场景6：用户进行正常操作，快速反馈 | 高 | 中 |
| 6 | 场景8：服务宕机 | 高 | 高 |
| 7 | 场景9：服务器或客户端移植到其它环境 | 低 | 中 |
| 8 | 场景14：发生网络闪断 | 高 | 中 |
| 9 | 约束1：必须在连接互联网的情况下可访问 | 高 | 低 |
| 10 | 约束3：服务器使用云服务器 | 中 | 中 |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 性能 | 管理资源 | I/O |
| 控制资源请求 | 减少开销 |
| 设置任务优先级 |
| 可用性 | 错误预防 | 连接方式 |
| 半包处理 |
| 错误检测 | 链路有效性检测 |
| 传输检验 |
| 错误恢复 | 断连重连 |
| 消息重发 |
| 可扩展性 | 序列化/反序列化 | 语言支持 |
| 数据结构升级 |
| 传输协议 | 协议支持 |
| 协议类型 |
| 安全性 | 抵制攻击 | 安全认证 |

1. **关注点的候选模式**
2. **I/O**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 复杂度 | 并发访问量 | 开销 |
| 1 | 同步阻塞I/O | 低 | 低 | 高 |
| 2 | 非阻塞I/O | 高 | 高 | 低 |

ii) 选择的模式及理由

选择非阻塞I/O。因为在同步阻塞I/O里，当客户端并发访问量增加后，服务端的线程个数和客户端并发访问数呈1：1的正比关系。当线程数膨胀之后，系统的性能急剧下降，随着并发访问量的继续增大，系统会发生线程堆栈溢出、创建新线程失败等问题，并最终导致进程宕机或僵死，不能对外提供服务。

而I/O多路复用技术通过把多个I/O的阻塞服用到同一个select的阻塞上，单线程可以同时处理多个客户端请求。它的最大优势是系统开销小，体统不需要创建新的额外进程或者线程，也不需要维护这些进程和线程的运行，降低了系统的维护工作量，节省了系统资源。

1. **减少开销**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | #服务器 | #服务器资源 | 通信负担 |
| 1 | 限制连接数量 | 高 | 低 | 低 |
| 2 | 对数据进行压缩 | 低 | 高 | 低 |

ii) 选择的模式及理由

选择限制连接数量。限制连接数量可以有效保证服务器运行的稳定与效率，通过使用更多的服务器来缓解通信负担是可以做到的，而对数据压缩虽然也可以通过减少通信的数据量减轻服务器通信负担，但是会占用更多的服务器资源进行压缩与解码。

1. **设置任务优先级**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 服务器效率 | 及时性 | 数据丢失 |
| 1 | 队列处理请求 | 高 | 低 | 否 |
| 2 | 优先处理消息 | 高 | 中 | 否 |
| 3 | 忽略非重要请求 | 高 | 高 | 是 |

ii) 选择的模式及理由

选择优先处理消息。将重要的信息优先处理，可以保证服务器正常运行，放弃不重要的信息，让用户再次请求。队列处理消息模式虽然减少了并行访问中资源共享的问题，但是难以对访问权限进行控制，不能保证消息一定被处理，无法保证及时性，忽略非重要请求的模式虽然可以让服务器集中处理关键数据，维持高效运行，但存在数据的丢失。相比较之下，优先处理消息模式虽然对于优先级低的请求有很大延迟，但也是重要信息优先处理不可避免的，是合理的选择。

1. **连接方式**

i) 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 资源消耗 | 远程调用时延 |
| 1 | 长连接 | 低 | 低 |
| 2 | 短连接 | 高 | 高 |
| 3 | 长连接和短连接结合 | 高 | 高 |

ii) 选择的模式及理由

选择长连接和短连接结合。长连接更节省资源，它只在首次创建时或者链路断连重连时才创建链路，链路创建成功之后服务提供者和消费者通过业务消息和心跳维系链路。以实现多消息复用同一个链路节省资源。同时，长连接可以减少链路的重建工作，这在远程通信中可以大大减少耗时，从而达到减少网络时延的目的。

但是在RPC过程中采用短连接更加合适。因为服务集群中的服务地址可能会改变，采用短连接可以及时释放资源。

1. **半包处理**
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 复杂程度 | 解码难易 |
| 1 | 固定长度，不足的前面补位 | 低 | 易 |
| 2 | 通过回车换行符区分消息 | 低 | 中 |
| 3 | 通过特定的分隔符区分整包消息 | 高 | 较高 |
| 4 | 通过在协议头/消息头中设置长度字段来标识整包消息 | 中 | 中 |

1. 选择的模式及理由

通过在协议头/消息头中设置长度字段来标识整包消息。通过在包头设置数据段的长度，可以保证读取的数据是完整的。不会造成空间的浪费，效率比较高。在服务端接收的时候，先接收该包头数据，然后再接收指定长度的数据体即可。

如果采用通过特定的分隔符区分整包消息，或者通过回车换行符区分消息，那么需要不停的判断每个发送过来的数据是不是结束符或者转义后的字符，增大开销。如果采用固定长度，不足的前面补位，这样会限制数据类型，如果一个包数据远少于这个长度，会造成空间的浪费。

1. **链路有效性检测**
2. 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | 链路负担 |
| 1 | 心跳检测 | 中 |
| 2 | Ping/Echo | 高 |
| 3 | 客户端检测错误 | 低(无) |

1. 选择的模式及理由

选择心跳检测。心跳是周期性的，检查速度快，更加全面，而Ping/Echo不仅是按需检查，实现更加复杂，而且需要的带宽是心跳检测的两倍。客户端检测错误虽然对链路无额外负担，但是服务端发送信息后，客户端不保证回复，所以是无法完成检测的。因此选择心跳检测。

1. **传输检验**
   1. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 准确率 | 成本 |
| 1 | 奇偶校验 | 中 | 低 |
| 2 | CRC校验 | 高 | 高 |

* 1. 选择的模式及理由

选择奇偶校验。可以对网络传输的数据进行基本的校验，虽然不能检测出偶数个错误，有损准确率，但是CRC校验的高准确率是以更多校验位为代价的，增加了网络资源的消耗。相较之下，节省资源、降低成本，已能满足数据可靠性的要求的奇偶校验是更好的选择。

1. **断连重连**

i) 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | #占用资源 |
| 1 | 客户端立即发起重连，反复发起重连直到重连成功 | 高 |
| 2 | 客户端等待INTERVAL时间，再发起重连操作，如果重连失败，间隔周期INTERVAL后再次发起重连 | 低 |

ii) 选择的模式及理由

选择客户端等待INTERVAL时间后再发起重连操作。因为这样可以保证客户端能够有充足的时间来释放句柄资源。

1. **消息重发**

i) 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | #资源 |
| 1 | 重连成功则重发：将缓存队列中的消息重新发送给通信对端 | 高 |
| 2 | 因链路中断导致的服务提供者不可用的消息，不再重发 | 低 |

ii) 选择的模式及理由

选择仅重发除中断导致的服务提供者不可用的消息。因为这类消息已经通过特定机制切换到另一个服务提供者处理，即使链路中断，也没有必要重发。

1. **语言支持**

i) 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | #编解码方式 |
| 1 | 不支持跨语言 | 单一 |
| 2 | 支持多种常用的序列化/反序列化方式，文本类例如XML/JSON等，二进制的如PB(Protocol Buffer)/thrift等 | 多 |

ii) 选择的模式及理由

选择提供跨语言支持的模式。这是衡量序列化框架通用性的一个重要指标。分布式服务框架根据业务特点不同，可能会采用不同的语言实现，使用同一种服务框架的不同语言开发的服务之间要能够互通，序列化与反序列化首先要能够支持互通，这就是支持跨语言的重要所在。

1. **数据结构升级**
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 升级时间 | 成本 |
| 1 | 前向兼容 | 快 | 低 |
| 2 | 代码重构 | 慢 | 高 |

ii) 选择的模式及理由

选择前向兼容的数据结构。这样服务升级时不必破坏已部署的、依靠“老”数据格式的程序就可以对数据结构进行升级。如果没有进行前向兼容，就需要进行大规模的代码重构或者迁移，所造成的成本是极高的，所以选择前向兼容。

1. **协议支持**

i) 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | #协议种类 | 功能 |
| 1 | 支持多协议 | 多 | 强大 |
| 2 | 不支持多协议 | 少 | 较弱 |

ii) 选择的模式及理由

选择不支持多协议。尽管从功能上看分布式服务框架默认支持的协议种类越多，功能越强大，但是分布式服务架构不需要负责异构框架的对接，所以所有使用分布式服务框架服务化的业务都需要遵循服务化规范，包括通信协议、服务配置等。

1. **协议类型**

i) 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 吞吐率 | 时延 |
| 1 | 公有协议 | 低 | 长 |
| 2 | 私有协议 | 高 | 短 |

ii) 选择的模式及理由

选择私有协议。因为分布式服务框架中吞吐率、时延等性能指标是关键。而在性能方面，私有协议往往可以根据业务的具体需求进行针对性优化，性能更优。而公有协议，以Web Service公有协议为例，因为SOAP消息使用XML进行序列化，又有HTTP协议承载，不支持双向全双工通信，这样都导致了较差的性能。所以选择性能更高、扩展性更好的私有协议进行通信。

1. **安全认证**

i) 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | 严格程度 |
| 1 | 基于IP地址的安全认证机制 | 低 |
| 2 | 基于密钥和AES加密的用户名+密码认证机制 | 高 |
| 3 | SSL/TSL安全传输 | 高 |

1. 选择的模式及理由

选择基于IP地址的安全认证机制。因为采用内网，不需要将服务开放给第三方非信任域的消费者，因此不需要特别严格的安全认证机制。内部长连接采用基于IP地址的安全认证机制。服务端对握手请求的IP地址记性合法性校验；如果在白名单之内，则校验通过；否则拒绝连接。

1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 0 | I/O | 非阻塞I/O | SC1,SC3,SC6 |
| 1 | 减少开销 | 限制每台服务器连接数量 | SC1,SC3,SC6 |
| 2 | 设置任务优先级 | 优先处理消息 | SC6 |
| 3 | 连接方式 | 长连接和短连接结合 | SC8,SC14 |
| 4 | 半包处理 | 通过在协议头/消息头中设置长度字段来标识整包消息 | SC6 |
| 5 | 链路有效性检测 | 心跳检测 | SC8,SC14,DC3 |
| 6 | 传输检验 | 奇偶检验 | SC6 |
| 7 | 断连重连 | 客户端等待INTERVAL时间后再发起重连操作 | SC8,SC14 |
| 8 | 消息重发 | 链路中断导致的服务提供者不可用的消息不重发 | SC8,SC14 |
| 9 | 语言支持 | 提供跨语言支持 | DC1 |
| 10 | 数据结构升级 | 前向兼容 | SC9,DC1 |
| 11 | 协议支持 | 不支持多协议 | DC1 |
| 12 | 协议类型 | 私有协议 | SC9,DC1 |
| 13 | 安全认证 | 基于IP地址的安全认证机制 | SC5,DC1 |

1. **架构视图**
2. **modules图（uses view）**



1. **C&C 图（SOA view）**



1. **评估**

此次设计没有发现矛盾。

#### 分配职责

1. **传输模块：**以 tcp/ip 协议为标准，进行数据的接收与消息的发送。接收到的 ping 消息将送给通信检测模块，接受到的普通消息将送给校验模块进行校验。
2. **校验模块：**包括传输检验，半包处理，接受消息，进行奇偶校验，验证消息的完整性与安全性。
3. **任务队列模块：**包括设置任务优先级，根据消息的优先级进行排列，等待被处理。
4. **通信检测模块：**包括链路有效性检测，负责验证并记录与客户端的连接情况。
5. **序列化/反序列化模块：**包括语言支持、数据结构升级，对信息进行序列化和反序列化的处理

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 传输模块 | 通信检测模块 | PingEcho | 0.5s |
| 传输模块 | 校验模块 | Check | 0.1s |
| 校验模块 | 序列化模块 | Serial | 0.1s |
| 序列化模块 | 任务队列模块 | RequestQuery | 0.1s |
| 任务队列模块 | 反序列化模块 | Deserial | 0.1s |

### 最终架构视图

参见附件：分布式微服务C&C.png 分布式微服务Module.png

## MVC

### 迭代一

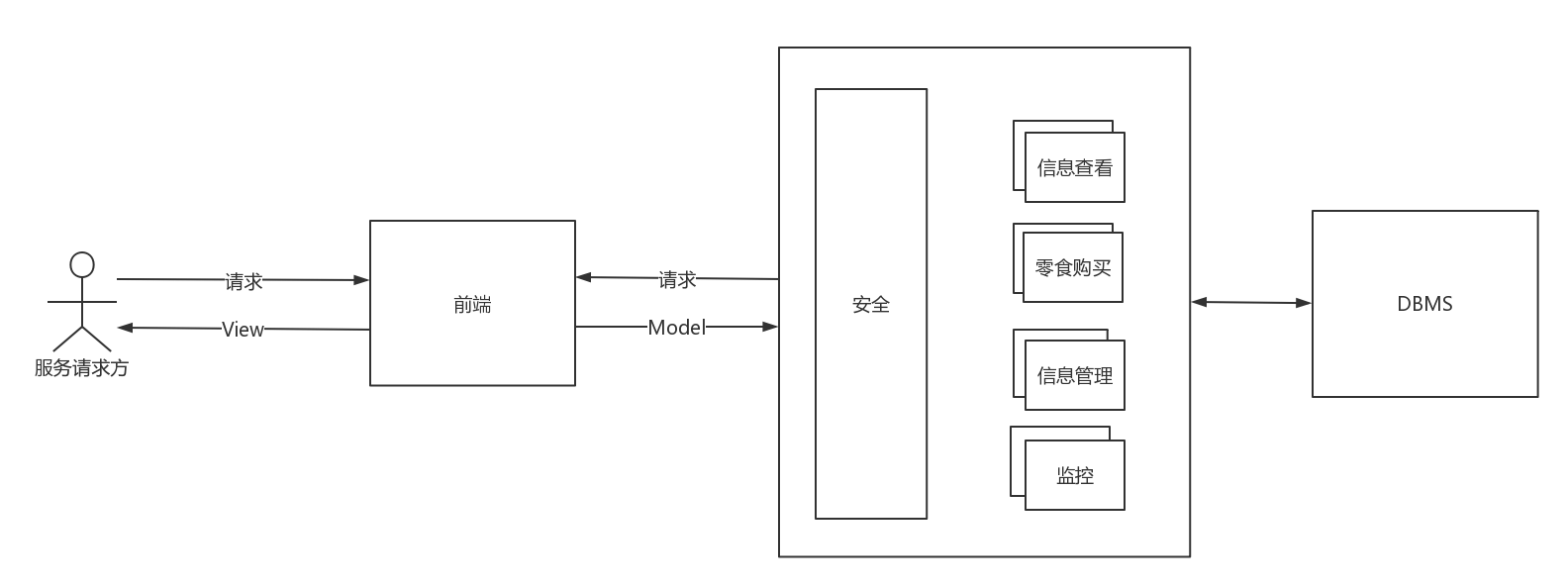
#### 需求信息

同分布式微服务迭代一

#### 分解的系统组件

本次迭代对系统进行整体设计

#### 架构视图



### 迭代二

#### 需求信息

同分布式微服务迭代一

#### 分解的系统组件

本次迭代选择零食购买模块进行分解

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 功能需求1  零食购买 | H | H |
| 2 | 场景1  >1000用户同时购买同一件商品 | H | H |
| 3 | 场景2  系统计算能力需要增减 | M | H |
| 4 | 场景3  用户进行正常操作 | H | H |
| 5 | 场景14  发生网络闪断 | M | M |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 性能 | 请求处理速度 | 购买操作完成速度 |
| 可伸缩性 | 增减计算能力 | 增减服务计算能力 |
| 可靠性 | 并发操作 | 并发计算 |
| 网络闪断 | 闪断检测 |
| 闪断处理 |

1. **关注点的候选模式**
2. 购买操作完成速度
3. 候选模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 单个请求时间开销 | 吞吐量 | 开发难度 | 成本 | 可扩展性 |
| 1 | 将购买操作分为流水线，分流水完成操作 | 高 | 高 | 高 | 中 | 高 |
| 2 | 提高硬件机器性能 | 中 | 中 | 中 | 高 | 中 |

1. 选择的模式及理由

由于需要应对大规模的请求操作，单单增加机器的性能会有较高的成本和较低的性价比，而在打规模请求的环境下，高吞吐量比单个请求完成的时间开销要更重要，于是这里选择模式1。

1. 并发计算
2. 候选模式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 可扩展性 | 最大并发量 | 维护难度 |
| 1 | 增加服务器数量，负载均衡 | 高 | 高 | 高 | 高 |
| 2 | 提高硬件机器性能 | 高 | 中 | 中 | 低 |

1. 选择的模式及理由

在电商环境下，必须有应对大量购买操作同时进行的场景，于是系统必须要有极高的并发量。仅仅提高单机的性能容易遇到并发量的瓶颈，达不到极高的并发要求。虽然增加服务器数量、使用负载均衡会使得服务器难以管理，但是为了应对大量操作的场景，还是必须的。所以这里选择模式1。

1. 增减服务计算能力
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 难度 | 成本 | 对当前服务的影响 |
| 1 | 增减硬件机器性能 | 低 | 高 | 有影响，更新机器性能会使得当前服务不可用 |
| 2 | 在多服务器的情况下增减服务器数量 | 低 | 中 | 无影响 |

1. 选择的模式及理由

由于之前选择了多服务器的模式，于是这里选择模式2。

1. 闪断检测
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 时间开销 | 资源开销 |
| 1 | 间隔一定时间未收到回复 | 高 | 中 |
| 2 | 间隔一定时间未收到回复确认 | 高 | 中 |
| 3 | 维持长连接链路，心跳检测 | 中 | 高 |

1. 选择的模式及理由

由于系统需要一定的健壮性来应对一些微小误差，需要对网络闪断有良好的检测能力。但是由于系统有大量的购买等请求，在资源开销上要选择较低的开销，这里选择模式1和2。

1. 闪断处理
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 服务器资源开销 | 客户端请求开销 |
| 1 | 请求重发 | 低 | 高 |
| 2 | 回复重发 | 高 | 低 |

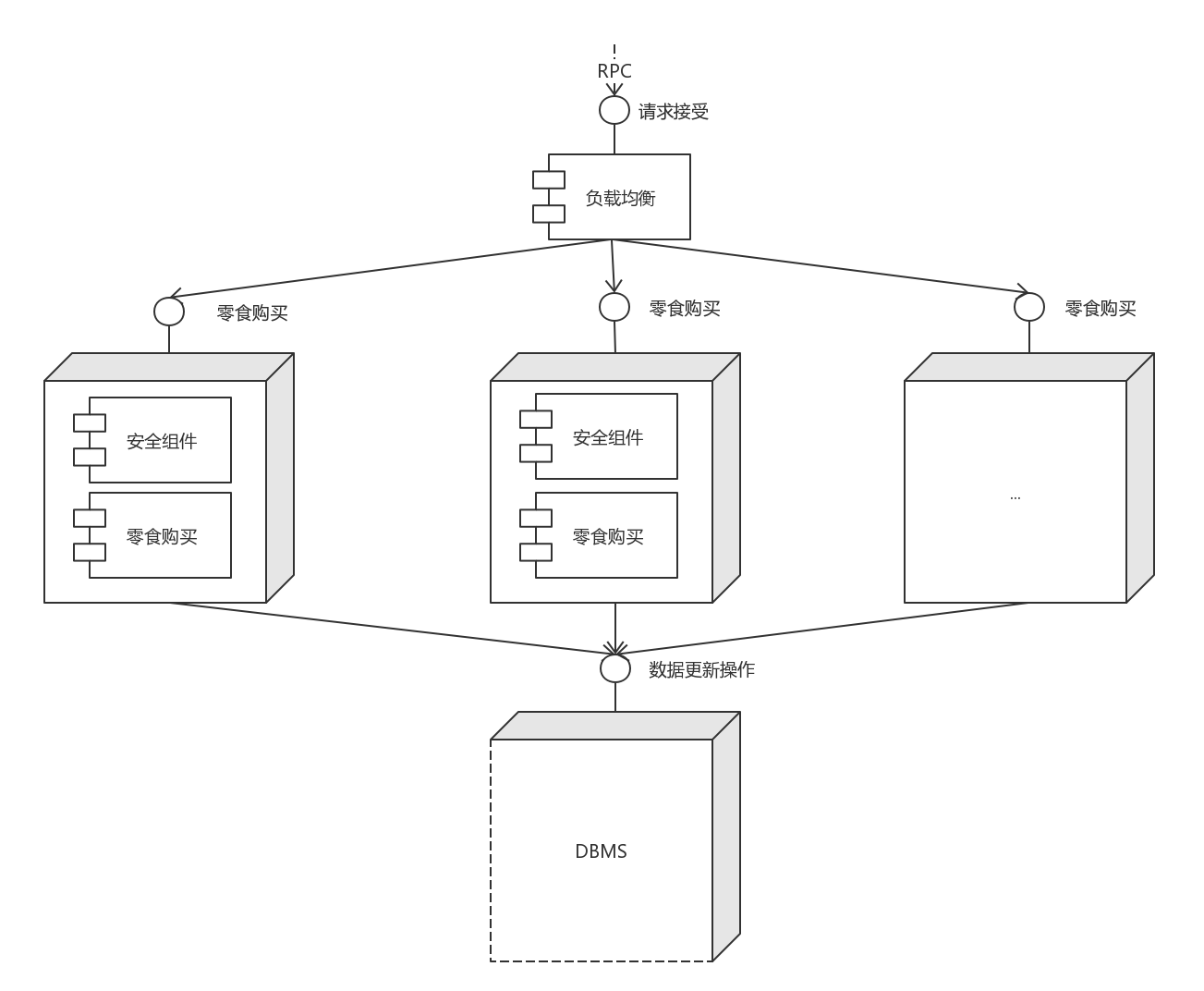
1. 选择的模式及理由

请求重发是指客户端在一定时间内没有接受到请求后再次向服务器发起请求，回复重发是指服务器没有收到客户端的回复确认时再次发起回复，由于服务器的压力较大，在此选择服务器资源开销较小的模式1。

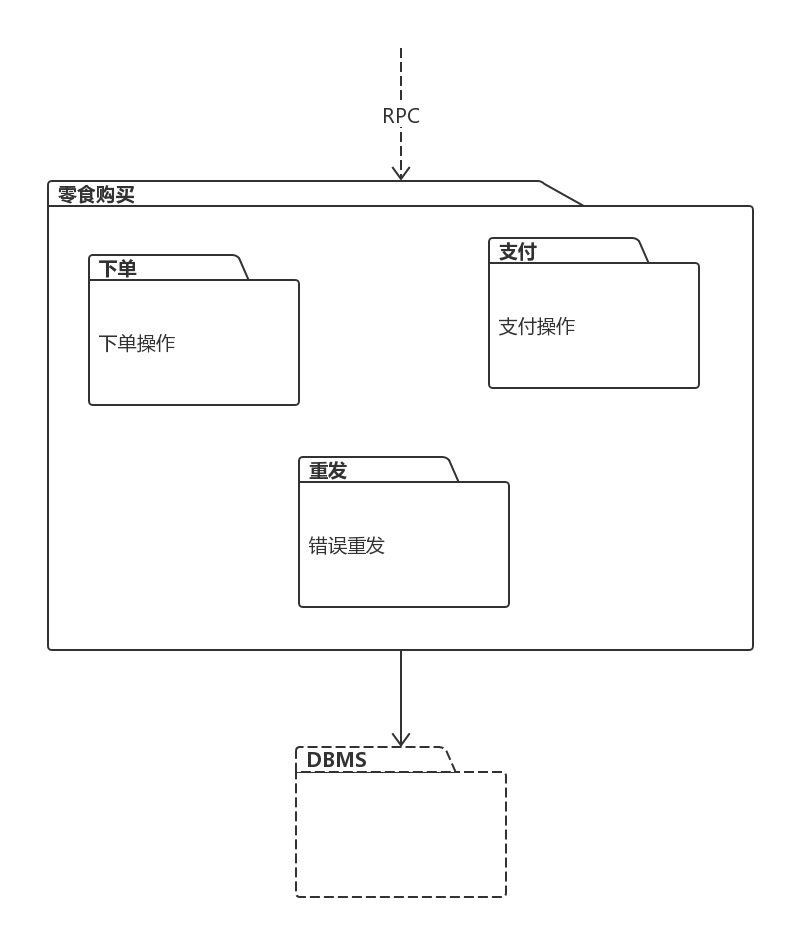
1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 购买操作完成速度 | 将购买操作分为流水线，分流水完成操作 | FC1、SC13 |
| 2 | 并发计算 | 增加服务器数量，负载均衡 | SC1、SC13 |
| 3 | 增减计算能力 | 在多服务器的情况下增减服务器数量 | SC3 |
| 4 | 闪断检测 | 间隔一定时间未收到回复 | SC14 |
| 5 | 闪断恢复 | 请求重发 | SC14 |

1. **架构视图**
2. **C&C视图**

****

1. **Module**

****

1. **评估**

此次迭代没有发现矛盾之处

#### 分配职责

1. **负载均衡：**对请求进行对应的负载均衡，使得每个计算节点上的负载情况大致相同。
2. **零食购买计算节点：**负载处理主要的业务逻辑，这里主要是对用户支付环境、传输安全等进行一系列检测，然后进行对应的零食购买操作。

#### 组件接口设计

（组件之间交互调用的服务，源组件调用目标组件）

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 零食购买服务节点 | 数据库系统 | 数据库更新操作 | 对应购买信息 |
| 负载均衡 | 零食购买服务节点 | 零食购买服务 | 对应购买信息 |
| 远程客户端 | 负载均衡 | 请求接受 | 需要的请求参数 |

### 迭代三

#### 需求信息

同分布式微服务迭代一

#### 分解的系统组件

选择进行分解的是信息查看模块

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 功能需求2  零食查看 | H | H |
| 2 | 功能需求6  报表查看 | H | M |
| 3 | 场景4  用户正常操作简单快捷 | H | M |
| 4 | 场景6  用户正常操作 | H | H |

#### 为ASR进行设计

1. **设计关注点**

|  |  |
| --- | --- |
| 质量属性 | 子关注点 |
| 易用性 | 使用简单，展示清晰 |
| 性能 | 限制信息并发展示数量 |
| 可移植性 | 适应不同的浏览器模式 |
| 安全性 | 不同身份的用户有不同权限 |

1. **子关注点的候选模式**
2. 使用简单，展示清晰
3. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 简单的界面和提示模式 | 能够增强系统的易用性，上手简单 | 设计负担较重 |

1. 选择的模式及理由

选择这个模式，因为系统本身要求简单明快，操作简单，才能够产生价值，因此对交互有较高的要求。

1. 限制信息并发展示数量
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 设置固定的最大信息展示值 | 能够使信息展示的数量在一定范围内，提高速度 | 信息展示不完整，可能导致用户操作变复杂 |
| 2 | 根据网络环境设置最大信息展示值 | 能够应对不同的网络环境情况 | 增加了系统的复杂度 |

1. 选择的模式及理由

选择设置固定最大值的模式，最大值的数值设定好之后，可以通过良好的界面和人机交互设计，在提高速度的同时有良好的用户体验。第二种模式，增加了系统设计的复杂度。

1. 适应不同的浏览器模式
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 使用响应式网页设计 | 能够适应不同页面大小来展示信息 |  |
| 2 | 使用浏览器兼容设计 | 能够在不同的浏览器中使用系统 |  |
| 3 | 组合模式 | 能够在适应不同浏览器的同时，适应不同界面大小的要求 |  |

1. 选择的模式及理由

选择组合模式，能够很好地应对各种不同的展示和使用要求，有良好的用户体验，更好地支持易用性。

1. 不同身份的用户有不同权限
2. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 优点 | 缺点 |
| 1 | 身份验证，权限设置 | 能够根据用户身份确定是否有操作权限 |  |

1. 选择的模式及理由

选择身份验证，权限设置模式，因为涉及核心信息，只有管理员才有权限查看报表，这种模式可以很好地避免非法用户的非法操作，提高了信息安全性。

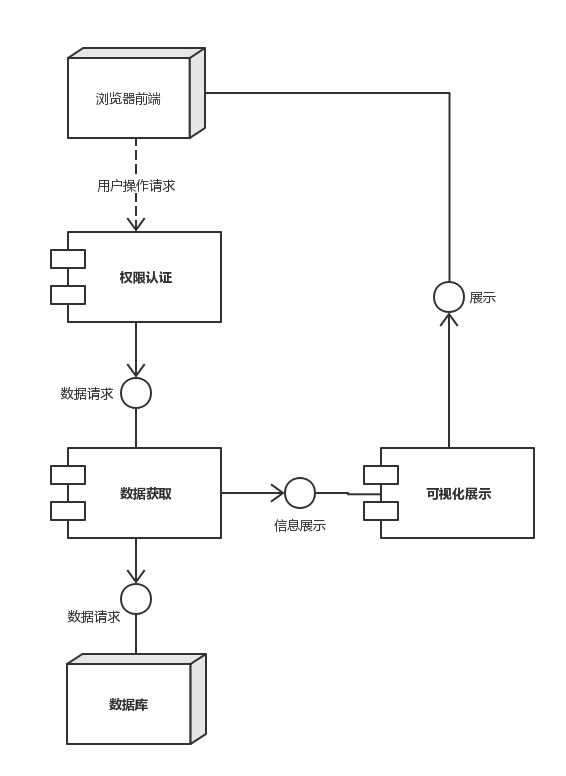
1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 使用简单，展示清晰 | 简单的界面和提示模式 | 功能需求2，场景4 |
| 2 | 限制信息并发展示数量 | 设置固定最大展示值 | 场景6 |
| 3 | 适应不同的浏览器模式 | 响应式和兼容设计组合模式 | 场景4 |
| 4 | 不同身份的用户有不同权限 | 身份验证，权限设置 | 功能需求6 |

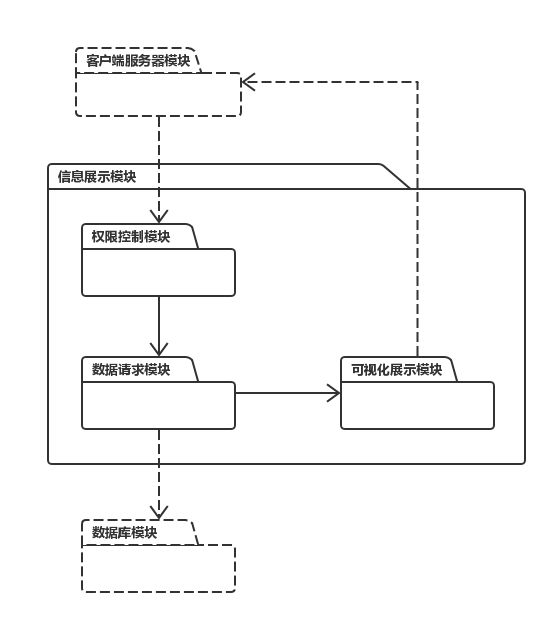
1. **架构视图**

画出此次迭代后的架构视图

* + 1. **C&C视图**



* + 1. **Module视图**



1. **评估**

此次设计完成良好，没有冲突

#### 分配职责

1. **权限控制模块：**验证用户身份是否具有操作权限
2. **数据请求模块：**向数据库请求数据
3. **可视化展示模块：**将服务请求结果返回到客户端

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 客户机服务器模块 | 权限控制模块 | Config | 用户ID |
| 权限控制模块 | 数据请求模块 | DataRequest | 用户身份，服务ID |
| 数据请求模块 | 数据库模块 | DataRequestDB | 用户身份，服务ID |
| 数据请求模块 | 可视化展示模块 | Display | 用户身份，服务结果 |
| 可视化展示模块 | 客户机服务器模块 | FinalView | 用户身份，服务结果 |

### 迭代四

#### 需求信息

同分布式微服务迭代一

#### 分解的系统组件

选择进行分解的是信息管理模块，负责零食、订单、报表、购物车等信息的增删改操作。

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 功能需求3：购物车管理 | H | H |
| 2 | 功能需求4：账户管理 | H | M |
| 3 | 功能需求5：订单管理 | H | H |
| 4 | 功能需求7：零食管理 | H | H |
| 5 | 场景4：用户正常操作简单快捷 | H | M |
| 6 | 场景7：用户进行了错误操作 | M | H |

#### 为ASR进行设计

1. **设计关注点**

|  |  |
| --- | --- |
| 质量属性 | 子关注点 |
| 易用性 | 使用简单，展示清晰 |
| 容错性、健壮性 | 操作可撤销 |
| 安全性 | API监控 |

1. **子关注点的候选模式**
2. **使用简单，展示清晰**
3. 候选模式

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式名称 | 易用性 | 成本 |
| 1 | 为用户和商家分别提供可视化界面 | 高 | 中 |

1. 选择的模式及理由

本模块由于既要为用户提供信息管理如购物车、账户管理等，又要为商家提供信息管理如零食管理、订单管理等，所以要为用户和商家分别提供可视化界面，这样提高信息管理模块的易用性。

1. **操作可撤销**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 时间成本 | 空间成本 | 开发成本 |
| 1 | Git仓库模式 | 低 | 高 | 低 |
| 2 | Commander回滚模式 | 高 | 低 | 高 |

1. 选择的模式及理由

Git仓库模式保存各个操作时间节点的数据更改信息，以达到保存每个时间点状态的效果，在进行误操作时可以撤销到任意一个时间点的状态；但是这种模式要求大量的空间来存储历史数据，在大型电商系统中对存储空间的需求更是巨大。Commander回滚模式利用Commander设计模式思想，要求为每个操作实例实现Do和Undo接口，利用Invoker的操作栈模式储存每个操作的信息，在进行错误操作后只需调用操作栈中各个操作的undo方法即可实现撤销，由于只记录操作和时间点，所需空间要比Git仓库小很多，但是由于业务操作的undo方法往往复杂度较高，加大了开发成本，而且进行undo时可能需求的时间很长。综合考虑，在硬件成本越来越低的现状下，此处选择Git仓库模式来满足操作可撤销的关注点。

1. **API监控**
2. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 效率 | 成功率 |
| 1 | Etag缓存 | 低 | 中 | 低 |
| 2 | 监控IP地址访问 | 中 | 低 | 低 |
| 3 | OAuth2.0认证 | 高 | 高 | 高 |

1. 选择的模式及理由

模式1利用Http协议缓存Etag可以让同样的请求结果不重复返回实体，而是以304Not Modified的方式返回给前端，这样可以过滤一部分重复请求；但是对于修改类型的操作而言是不可以使用Etag返回的，所以Etag在管理模块中实现的成功率比较低。

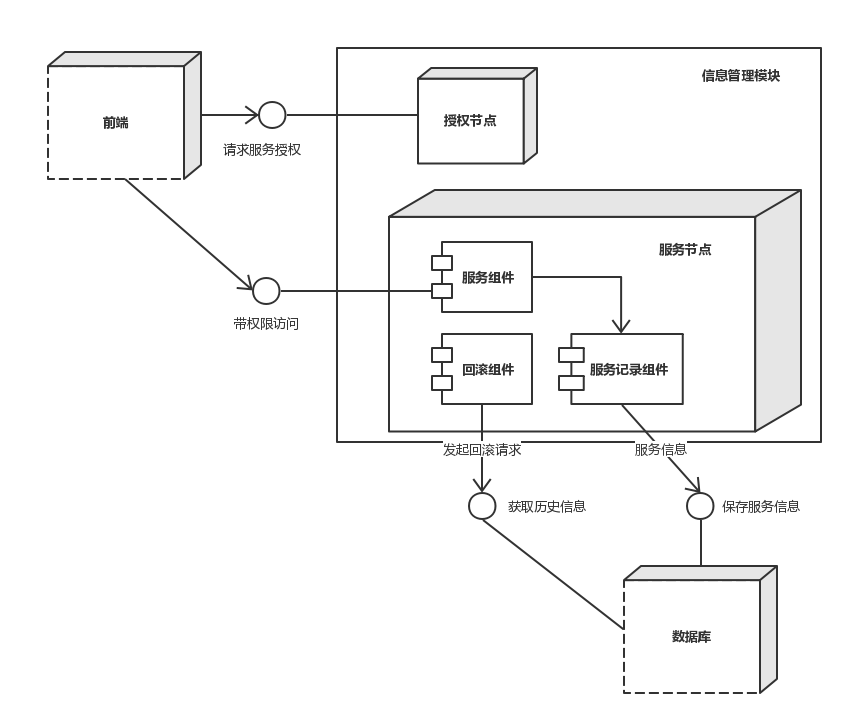
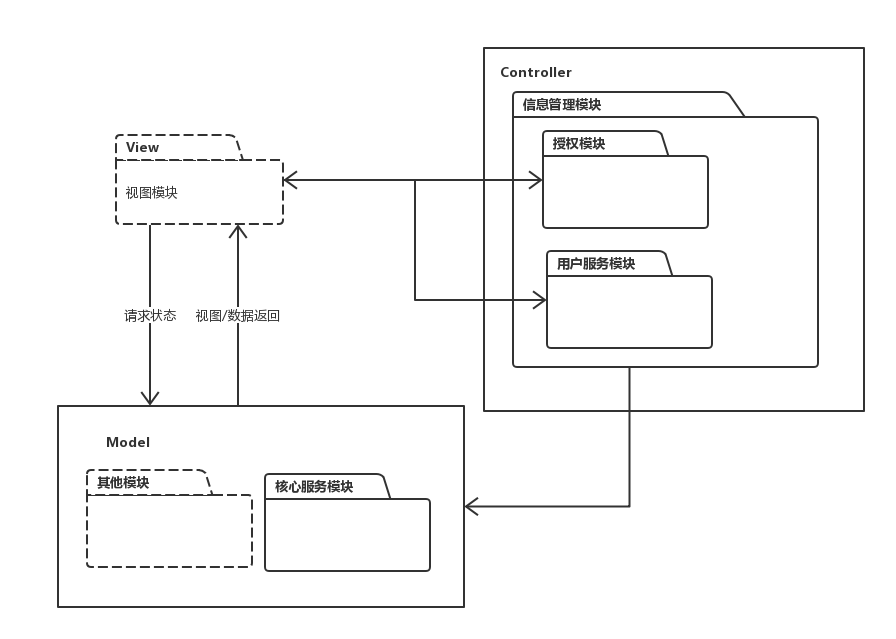
模式2利用IP控制，利用中间件或是高性能内存数据库临时存储各个IP地址的访问频率，从而达到控制的效果；但是这样做会带来额外的性能开销，开发成本也比较高。

模式3利用OAuth2.0认证，分离核心业务逻辑API和用户逻辑API，如果要调用核心业务逻辑API则需要进行OAuth2.0认证，这可以比较有效的环节API攻击，因为我们可以控制每个用户对于核心业务逻辑API请求token的获取时长、使用次数等，从而满足API监控的关注点。缺点是开发成本相较前2种模式比较高。

此处选择模式3以达到最高效率和成功率的API监控。

1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 1 | 使用简单，展示清晰 | 为用户和商家分别提供可视化界面 | 功能需求3、4、5、7，场景4 |
| 2 | 操作可撤销 | Git仓库模式 | 场景7 |
| 3 | API监控 | OAuth2.0认证 | 场景7 |

1. **架构视图**
   * 1. **C&C视图**
     2. **Module视图**
2. **评估**

此次设计完成良好，没有冲突

#### 分配职责

1. **授权模块：**验证用户身份是否具有操作权限，若有返回授权码
2. **用户服务模块：**处理授权后的服务请求
3. **核心服务模块：**信息管理相关的数据操作服务

#### 组件接口设计

|  |  |  |  |
| --- | --- | --- | --- |
| 源组件 | 目标组件 | 接口名 | 参数 |
| 前端 | 授权节点组件 | 请求服务授权 | 前端信息 |
| 前端 | 服务组件 | 带权限访问 | 授权码，服务ID |
| 服务记录组件 | 数据库模块组件 | 保存服务数据 | 服务数据 |
| 回滚组件 | 数据库模块组件 | 获取历史信息 | 回滚请求 |

### 迭代五

#### 需求信息

同分布式微服务迭代一

#### 分解的系统组件

针对安全模块进行分解、分析与设计。

#### 组件负责的ASRs

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动 | Importance | Difficulty |
| 1 | 场景1：>1000用户同时购买同一件商品 | H | H |
| 2 | 场景5：未登录用户进行支付操作 | H | M |
| 3 | 功能1：零食购买 | H | H |
| 4 | 功能2：报表查看 | M | M |
| 5 | 约束1：必须在连接互联网的情况下可访问 | H | L |

#### 为ASR进行设计

1. **设计关注点**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 设计关注点 | 子关注点 |
| 安全性 | 攻击监测 | 入侵判断 |
| 防止数据丢失 |
| 攻击抵制 | 安全认证 |
| 数据加密 |
| 系统恢复 | 恢复方式 |

1. **关注点的候选模式**
2. **入侵判断**
3. 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 技术要求 | 防御时间 |
| 1 | 检测是否被侵入 | 高 | 高 | 快 |
| 2 | 监测消息延迟 | 低 | 低 | 慢 |

1. 选择的模式及理由

选择监测消息延迟。因为可以十分方便地监测到某些模块之间的通信是否受阻，及时定位攻击点。虽然容易受网络延迟等环境因素影响，但是直接检测是否侵入需要消耗额外的人力，权衡之下，选择监测消息延迟。

1. **防止数据丢失**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | #错误时保留数据 | 备份时占用资源 | 误操作概率 |
| 1 | 定期对数据进行备份 | 部分 | 多 | 中 |
| 2 | 设置权限 | 无 | 少 | 低 |

ii) 选择的模式及理由

选择定期对数据进行备份。因为虽然设置权限能降低误操作概率，防止因此产生的数据丢失，但是一旦发生数据丢失，是无法恢复的，所以选择在出现错误时能够保留部分数据的模式一是更明智的选择。而且本系统主要存储临时数据，备份不会占用过多资源。

1. **安全认证**

i) 候选模式

|  |  |  |
| --- | --- | --- |
| # | 模式名称 | 严格程度 |
| 1 | 基于IP地址的安全认证机制 | 低 |
| 2 | 基于密钥和AES加密的用户名+密码认证机制 | 高 |
| 3 | SSL/TSL安全传输 | 高 |

ii) 选择的模式及理由

选择基于IP地址的安全认证机制。因为采用内网，不需要将服务开放给第三方非信任域的消费者，因此不需要特别严格的安全认证机制。内部长连接采用基于IP地址的安全认证机制。服务端对握手请求的IP地址进行合法性检验；如果在白名单之内，则检验通过；否则拒绝连接。

1. **数据加密**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本 | 安全性 | 处理效率 |
| 1 | https加密 | 高 | 高 | 快 |
| 2 | RSA非对称加密 | 低 | 高 | 慢 |
| 3 | AES对称加密 | 低 | 中 | 快 |
| 4 | RSA+AES | 低 | 高 | 较快 |

ii) 选择的模式及理由

选择RSA+AES相结合的模式。因为https加密方式成本高，RSA非对称加密因为使用了一对密钥，公钥与私钥，所以安全性高，但加密与解密速度慢，RSA对称加密通过加密解密使用同样的密钥解决了速度慢的问题，但安全性却降低了，尤其是在需要将密钥在网络传输的要求下，简单使用AES机密解密也是无法满足安全性的。所以选择RSA+AES模式，将对称加密的密钥使用非对称加密的公钥进行加密，然后发送出去，接收方使用私钥进行解密得到对称加密的密钥，然后双方可以使用对称加密进行沟通。这样就兼顾了性能和安全性。

1. **恢复方式**

i) 候选模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 恢复程度 | 负载 | 效率 |
| 1 | 保留系统运行记录 | 完全恢复 | 中 | 低 |
| 2 | 投票决议管理器(Voter) | 中 | 低 | 高 |
| 3 | 检查点(Checkpoint) | 便于测试 | 高 | 中 |

ii) 选择的模式及理由

选择投票决议管理器(Vector)。因为保留系统运行记录虽然可以监控全局的系统错误，及时给出修正，但是会极大地影响运行效率，设置检查点也是如此，便于测试但是会增大系统的负载。而投票决议管理器能够在不增加系统负载的情况下监控全局的状态变化，方便管理人员及时定点修正，是不错的选择。

1. **候选模式与对应ASR**

|  |  |  |  |
| --- | --- | --- | --- |
| # | 模式类型 | 选择的模式 | 架构驱动 |
| 0 | 入侵判断 | 监测消息延迟 | SC1 |
| 1 | 防止数据丢失 | 定期对数据进行备份 | SC1 |
| 2 | 安全认证 | 基于IP地址的安全认证机制 | SC1,SC5,DC1 |
| 3 | 数据加密 | RSA+AES | SC5，DC1 |
| 4 | 恢复方式 | 投票决议管理器(Vector) | SC5,DC1 |

1. **架构视图**

该模块实现机制分散在其它模块中，参见其它模块内部模块图。

1. **评估**

此次设计没有发现矛盾。

#### 分配职责

1. **安全认证模块：**包括http请求一个受保护的资源时的安全认证。
2. **数据加密/解密模块：**包括发送方发送消息的加密和接收方对消息的解密。
3. **数据备份模块：**包括定期对数据进行备份，以防丢失。
4. **监控模块：**包括对全局的状态的监控，关注系统状态的变化，方便系统被攻击时管理人员能及时定点修正。

#### 组件接口设计

因为该模块内部模块没有直接连接，所以无接口定义。

### 迭代六

此次迭代同分布式微服务迭代五，对数据模块进行拆分

### 最终架构视图

参见附件：MVC\_C&C.png MVC\_Module.png

# 架构对比

## 综述

系统分别采用MVC架构和分布式微服务架构，基于ADD方法进行系统设计。现对上述两种架构进行对比，以阐述最终选择分布式微服务架构的原因

## 性能对比

|  |  |
| --- | --- |
| 分布式微服务 | MVC |
| * 低并发情况下各模块之间交互导致速度相对较慢 * 高并发情况下访问速度快 | * 低并发、低访问量时可提供稳定的服务 * 高并发情况下单台服务器的存储能力和处理能力容易成为性能瓶颈 |

## 可用性对比

|  |  |
| --- | --- |
| 分布式微服务 | MVC |
| * 通常情况下一台服务器宕机不影响整个系统的正常工作 | * 一台服务器宕机，整个系统也随之瘫痪 |

## 安全性对比

|  |  |
| --- | --- |
| 分布式微服务 | MVC |
| * 各模块相对独立，且都具有相关安全措施，安全性高 * 一台服务器受到攻击不会影响这个系统的正常工作 | * 数据与控制流方向单一，高度集中，安全措施实现相对简单 * 一台服务器被攻击，将影响到整个系统的正常工作 |

## 成本对比

|  |  |
| --- | --- |
| 分布式微服务 | MVC |
| * 所需主机数量较多，硬件成本高 * 资源备份需要消耗大量存储空间 * 系统复杂度高，开发成本高 * 系统维护成本高 | * 所需主机数量相对较少，硬件成本低 * 系统复杂度低，开发成本低 * 系统维护成本低 |

# 类图

经过小组成员的讨论，我们最终决定对分布式微服务进行进一步设计，以下为分布式微服务架构的设计类图

## 类图

参见附件：分布式微服务类图.png

## 映射关系

|  |  |
| --- | --- |
| 组件 | 对应类 |
| **服务集群** | |
| 消息队列 | MessageQueue |
| 检索服务 | DataSearchEngine |
| 服务调用监控 | MonitorService |
| 安全组件 | AuthorityCheck |
| 零食购买等业务组件 | ConcretService |
| **服务前端** | |
| 安全处理 | SecurityService |
| 缓存 | ServiceTable, ServiceTableService, ServiceInfo |
| 请求解析 | RequestParserService |
| 请求转发 | RequestTransferService |
| 定时监测 | MonitoringService |
| **注册中心** | |
| 服务监听组件 | RegisterListener（实现类为Leader） |
| 同步组件 | SyncService（实现类为Leader） |
| 分发服务组件 | RequestService（实现类为Leader） |
| **数据库模块** | |
| 存储服务器 | User、Store、StoreManager、Snack、Order  Picture |
| 文件服务器 | PictureDaoImpl、StateServiceImpl |
| 数据库 | UserDaolmpl、StoreDaoImpl、SnackDaoImpl、OrderDaoImpl、PictureDaoImpl、StateServiceImpl |
| 后台管理系统 | StateService |
| **通讯模块** | |
| 传输 | Message |
| 校验模块 | MessageCheck |
| 通信检测 | ConnectCheck |
| 序列化/反序列化 | Serial/Deserial |
| 任务队列 | QueryOrder |

# 挑战与经验

从系统层面来讲，为了构建一个比较完整的分布式系统，我们组对整个系统的设计比较细致，但是由于对分布式系统的了解只停留在一个感性的层面上，当切切实实对系统进行细致的设计时，发现了许多意向不到的问题，比如数据库怎么进行分布式设计来满足需求、哪个部件需要进行分布式或集群处理、集群之间的通信方式等。为了解决这些问题，我们小组成员花了大量时间查阅各种资料。但是对整个系统进行设计也使得我们对分布式系统有了更深入的认识和理解。

从设计过程来讲，设计过程中我们使用了ADD方法。ADD设计方法有别于传统的功能驱动架构，它是以质量属性为核心，要求系统的“保质保量”。在开发过程中，采用迭代的方式，不断地分解系统模块，然后明确子模块中的设计要点，并给出若干备选方案，接着根据质量属性的要求从备选方案中挑出最合适的方案，推进模块设计，落实到底层接口，以保证获得当下最优选的体系结构设计。直到ASR条件全部满足，才会停止迭代，形成确定版本。ADD教会我们如何去平衡众多的质量属性，在重要程度、难易程度上做分析，在若干方案中平衡利弊，真刀真枪地理解迭代的精髓，从高层向低层、从抽象到实体，层层推进，收获颇丰。

从执行设计活动来讲，由于整个小组有7个人，人数较多，且在设计前期需要较多的时间进行讨论，于是怎么能更好的利用大家的时间也成了一个难题。于是我们从一开始就对系统进行和相对独立的拆分，将7个人分为5个小组，分别执行这些独立模块的设计，对一些需要特别交流和沟通的地方，再在小组会议上进行商讨，使得整个设计活动较为顺利。

# 个人总结

|  |  |  |
| --- | --- | --- |
| 姓名 | 学号 | 职责 |
| 杨雁飞 | 141250169 | 对系统进行整体架构设计，对各个模块进行拆分和定义；使用ADD方法对分布式微服务框架的服务集群和MVC框架零食购买模块进行设计；文档模板的书写和最终文档的汇总和评审。 |
| 邢程 | 141250153 | 阅读材料，和组员和组员一起初步讨论系统的功能需求、设计约束和质量属性以及设计的关注点; 功能需求的书写，分布式微服务框架中前端服务器模块部分的书写，MVC框架下监控模块的书写。 |
| 徐璠斐 | 141250157 | 阅读项目相关书籍，参与小组讨论确定项目功能、设计约束和场景；负责分布式微服务框架中通信模块的书写；负责MVC框架中安全模块的书写。 |
| 陶子涵 | 141250125 | 和组员一起参与讨论系统的功能需求、设计约束和质量属性以及设计的关注点; 功能需求的书写，分布式微服务框架中前端服务器模块部分的书写，MVC框架下信息查看模块的书写。 |
| 张瑞 | 141250189 | 参与项目需求、约束、非需求功能的讨论与分析，完成分布式微服务的 ADD 的通信模块的编写，完成质量属性场景的编写，阅读相关书籍。 |
| 宋益明 | 141250111 | 参与小组讨论，确定功能需求、质量属性和约束；参与了MVC架构和微服务架构的ADD过程；负责数据存储模块的设计，完成数据存储模块及架构对比的书写。 |
| 徐朱峰 | 141250164 | 参与项目需求、约束、非需求功能的讨论与分析，通过理解Apache Zookeeper的实现方案，完成分布式微服务框架中注册中心集群的设计与ADD过程和质量属性场景的编写。同时完成了MVC框架中信息管理模块的设计与ADD过程和质量属性场景的编写。 |