**2. SOA架构设计与评估**

**2.1 ADD过程**

**2.1.1 第一次迭代**

* 步骤一

获得样本输入，筛选掉不必要的需求，列出ASR：

互操作性：场景1：需要与支付系统进行交互

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 希望现有系统能与手机支付APP端进行交互 |
| 制品 | 客户端 |
| 环境 | 系统运行时或构建时 |
| 响应 | 与用户所选的支付APP进行数据传输与响应 |
| 响应度量 | 交互成功率达到100% |

互操作性：场景2：需要与GPS全球定位系统进行交互

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 希望现有系统能与全球定位系统进行交互，获取用户当前位置 |
| 制品 | 服务端 |
| 环境 | 系统运行时或构建时 |
| 响应 | 与GPS全球定位系统进行交互 |
| 响应度量 | 误差在20米之内的交互成功率达到99% |

可扩展性：场景3：需要增加运动统计模块

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 需要增加运动统计功能 |
| 制品 | 新的功能模块 |
| 环境 | 系统在构建中或系统已经上线运行 |
| 响应 | 编写运动统计模块功能代码  对所编写代码进行测试  发布更新 |
| 响应度量 | 更新大小不超过原APP大小的10%  所影响到的源代码量不能超过2% |

可扩展性：场景4：需要增加社交圈子模块

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 开发人员 |
| 刺激 | 需要增加社交圈子模块 |
| 制品 | 新的功能模块 |
| 环境 | 系统在构建中或系统已经上线运行 |
| 响应 | 编写社交圈子模块功能代码  对所编写代码进行测试  发布更新 |
| 响应度量 | 更新大小不超过原APP大小的15%  所影响到的源代码量不能超过5% |

可恢复性：场景5：数据库丢失数据或程序

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 由于服务器宕机，损坏，或遭受攻击等原因造成的数据或程序丢失 |
| 制品 | 数据库被还原的数据，日志 |
| 环境 | 系统暂停运行或运行时 |
| 响应 | 查明数据丢失原因  将数据库快速恢复到尽可能新的时间点 |
| 响应度量 | 查明数据丢失原因时间 < 10分钟  数据库恢复时间 < 10分钟  丢失所影响的数据量 < 1% |

可恢复性：场景6：系统故障或局部故障

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维护人员 |
| 刺激 | 某服务失效，或系统崩溃，宕机。 |
| 制品 | 重启系统或重启单个服务 |
| 环境 | 系统运行时 |
| 响应 | 查明故障原因  重启故障部分  恢复到故障前的状态 |
| 响应度量 | 查明数据丢失原因时间 < 10分钟  重启故障部分 < 1分钟  恢复到故障前的状态 < 1分钟  服务24小时均可用，可自动恢复状态 |

性能：场景7：用户正常使用客户端

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 用户正常使用客户端进行借车/还车等服务 |
| 制品 | 客户端 |
| 环境 | 系统正常运行 |
| 响应 | 用户操作得到快速地处理 |
| 响应度量 | 客户端中用户操作的响应时间应小于1s |

性能：场景8：用户客户端进行网络访问：如举报，用车/还车，支付等

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 用户正常使用客户端进行涉及网络访问的服务 |
| 制品 | 客户端中涉及网络访问的模块，ESB，服务的网络层 |
| 环境 | 系统正常运行 |
| 响应 | 客户访问服务速度快，响应良好 |
| 响应度量 | 若不涉及外部服务，客户端向服务器发出请求后响应时间应少于2s。  若涉及外部服务，服务器中转的响应时间应少于2s。  系统能够允许30000客户端同时访问网络服务 |

可伸缩性：场景9：用户数量增加

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 手机端用户 |
| 刺激 | 同时使用查询附近车辆、借车功能的用户数量增加 |
| 制品 | 系统 |
| 环境 | 系统已上线运行 |
| 响应 | 正常进行车辆位置查询、正常借车 |
| 响应度量 | 可支持的并发用户数量不低于10000  请求查询附近车辆成功的用户不低于95%  请求借车成功的用户不低于95%  用户请求查询车辆，获得返回的信息的平均时间< 5秒  用户借车的平均请求时间不超过5秒 |

可伸缩性：场景10：单车数量增加

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 单车端 |
| 刺激 | 同时报告自身位置的单车数量增加 |
| 制品 | 系统 |
| 环境 | 系统已上线运行 |
| 响应 | 单车可以成功报告自己所在位置 |
| 响应度量 | 可支持的并发单车数量不低于10000  成功报告自己位置的单车不低于95% |

安全性：场景11：未经授权的访问

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 未经授权的个人或其他系统 |
| 刺激 | 试图显示数据或者改变／删除数据，试图改变系统服务 |
| 制品 | 系统服务，系统服务器中的数据 |
| 环境 | 系统联网状态下运行 |
| 响应 | 识别请求源是否被授权  阻止对服务或数据的访问  以一种不可读的方式存储数据  数据或者服务遭到破坏后可恢复 |
| 响应度量 | 成功识别攻击的概率不低于99%  在拒绝服务的情况下，攻击者仍能获得服务的概率低于1%  恢复数据和服务可在3人日内完成 |

安全性：场景12：经过授权的访问

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 经授权的个人或其他系统 |
| 刺激 | 试图访问不在其权限范围内的数据或者服务 |
| 制品 | 系统服务，系统服务器中的数据 |
| 环境 | 系统联网状态下运行 |
| 响应 | 对用户身份进行验证  允许访问用户权限范围内的服务和数据  拒绝访问不在用户权限范围内的服务和数据，并通知用户 |
| 响应度量 | 成功识别用户身份的概率不低于99%  拒绝不合法访问的概率高于99%  在拒绝服务的情况下，用户仍然可以获得服务的概率低于0.1% |

易用性：场景13：用户解锁单车

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 终端用户解锁单车 |
| 制品 | 系统 |
| 环境 | 系统正常运行，单车电量充足 |
| 响应 | 用户打开客户端时，显示周边单车  在用户解锁单车时，指导用户如何解锁  用户正在解锁时，提示用户单车正在被解锁 |
| 响应度量 | 用户能在1分钟内解锁单车  解锁一辆单车，用户需要点击的次数不超过4次  解锁成功率高于99.9% |

易用性：场景14：用户结束骑行

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 终端用户结束骑行 |
| 制品 | 系统 |
| 环境 | 系统正常运行，单车电量充足 |
| 响应 | 客户终端指导用户结束骑行  客户上锁时，提醒客户上锁成功、骑行结束 |
| 响应度量 | 从客户上锁到系统提示客户上锁成功的时间<1分钟  客户完成结束骑行的操作时间<2分钟  客户操作成功率>99.9% |

易用性：场景15：用户注册账号

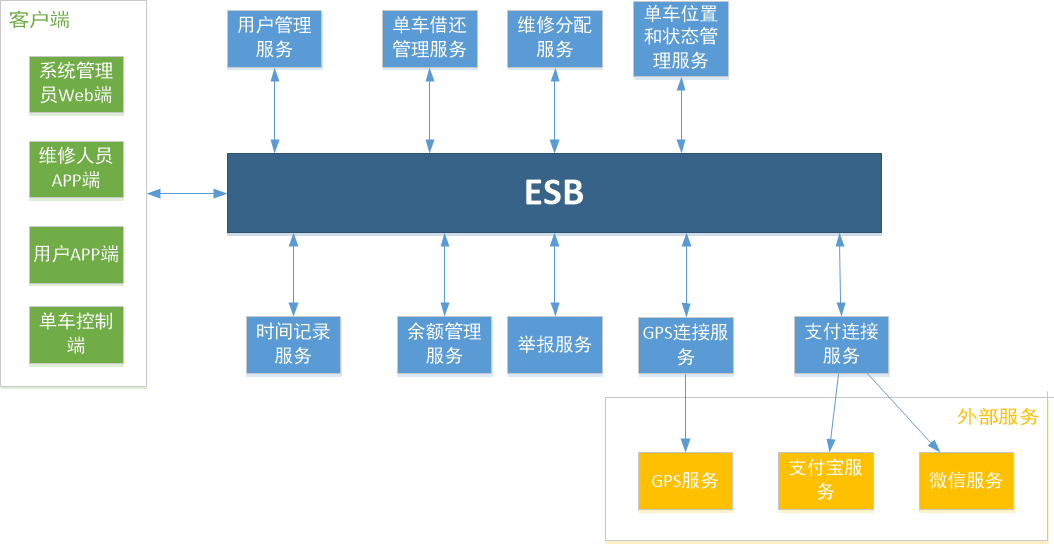
|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 终端用户 |
| 刺激 | 终端用户想要注册一个账户 |
| 制品 | 系统 |
| 环境 | 系统正常运行 |
| 响应 | 指导用户填写账号名、密码、证件号时，并提示其格式  用户交押金时，系统应提供用户支付手段用于选择 |
| 响应度量 | 用户注册账号的时间<3分钟  用户注册成功所发生的错误应<2次 |

可用性：场景16：单车发生故障，不能解锁

|  |  |
| --- | --- |
| 场景组成部分 | 可能的值 |
| 源 | 维修人员 |
| 刺激 | 单车发生故障，维修人员进行维修 |
| 制品 | 单车端嵌入式系统 |
| 环境 | 单车管理服务正常运行 |
| 响应 | 查明故障原因  更换损坏部件  重启单车端嵌入式系统 |
| 响应度量 | 修复一辆单车的时间<10分钟  单车端嵌入式系统无故障运行时间>1个月 |

* 步骤二

对整个系统进行分解，分为用户管理服务、单车借还管理服务、维修分配服务、单车位置和状态管理服务、时间记录服务、余额管理服务、举报服务、GPS连接服务、支付连接服务和客户端。



**2.1.2 第二次迭代**

* 步骤一

第二次迭代可以不使用此步骤

* 步骤二

在此次迭代中针对时间记录服务进行分解

* 步骤三

确定架构驱动因素，如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动因素 | 重要性 | 难易度 |
| 1 | 场景9：用户数量增加 | 高 | 高 |
| 2 | 场景5：数据库丢失数据或程序 | 中 | 中 |
| 3 | 场景6：系统故障或局部故障 | 中 | 高 |

* 步骤四

选择满足架构因素的架构模式

1)针对可恢复性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 数据丢失 | 数据库数据丢失 |
| 服务故障 | 服务失效 |
| 系统崩溃 |

ii. 各子关注点的候选架构模式

1. 数据库数据丢失

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 开发时间开销 | 成本开销 | 恢复成功率 |
| 1 | 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | 高 | 高 | 高 |
| 2 | 采用中央数据库保存多次副本 | 低 | 低 | 低 |

* 模式选择：选择采用分布式数据库，虽然成本和开发时间消耗较高，但是恢复成功率远高于采用本地多副本保存而可能所有数据库都遭到数据波动影响的本地中央数据库，本地多副本中央数据库也容易因为数据量的增大而面临存储空间不足的问题。

1. 服务失效

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 开发时间开销 | 成本开销 | 服务失效所带来影响 |
| 1 | 注册备用服务进行转接 | 低 | 低 | 中 |
| 2 | 本地提前开发相应应急服务 | 高 | 高 | 低 |

* 模式选择：选择采用本地提前开发的应急服务，服务失效时除了检查ESB是否出现问题外，还要检查服务提供商原服务是否出现问题，而且另外注册的服务也很有可能同时出现问题，因此开发本地应急服务以保证系统正常运行。

1. 系统崩溃

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 开发时间开销 | 成本开销 | 恢复时间 |
| 1 | 本地购买备用主机 | 低 | 高 | 低 |
| 2 | 依赖区域远程主机 | 低 | 中 | 中 |
| 3 | 购买质量较好的主机 | 低 | 低 | 高 |

* 模式选择：选择采用本地购买备用主机的方案。依赖远程主机需要进行服务转接，也同时瞬间增加了远程主机的负载，远程主机带来的延迟也可能对性能因素带来影响。购买较好主机不能避免问题的发生，而且恢复时间最长，因此不予采用。

iii. 候选模式综合评价

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 数据库数据丢失 | | 服务失效 | | 系统崩溃 | |
| 优点 | 缺点 | 优点 | 缺点 | 优点 | 缺点 |
| 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | 保证了数据丢失时恢复的完整性和争取性 | 恢复时间较长，数据丢失点不清楚，需要完全通过网络拷贝 | 无显著影响 | 维护副本 | 系统崩溃时远程主机可临时充当代理主机 | 维持庞大数据库可能会加重服务器负担 |
| 本地提前开发相应应急服务 | 无显著影响 | 无显著影响 | 极大减少失效影响持续时间 | 本地服务可能与原服务存在实现差异或不同地方，可能造成错误 | 无显著影响 | 可能会加重服务器负担 |
| 本地购买备用主机 | 可以作为临时服务器，并对丢失数据库主机进行数据恢复 | 无显著影响 | 无显著影响 | 无显著影响 | 保证了服务器崩溃时能够及时转接 | 极大增加了成本开销 |

* 评估结果：三个模式相互之间没有显著矛盾，可以作为最终方案

**2.1.3 第三次迭代**

* 步骤一

第三次迭代可以不用此步骤

* 步骤二

我们选择单车位置和状态管理服务模块进行分解

* 步骤三

确定架构驱动因素，如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动因素 | 重要性 | 难易度 |
| 1 | 场景10：单车数量增加 | 高 | 中 |
| 2 | 场景9：用户数量增加 | 中 | 中 |
| 3 | 场景5：数据库丢失数据或程序 | 高 | 中 |
| 4 | 场景6：系统故障或局部故障 | 中 | 高 |

* 步骤四

选择满足架构驱动因素的架构模式

1. 针对可伸缩性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 可伸缩性 | 单车数量增加导致数据库容量扩大 |
| 用户数量增加导致对数据库访问增多 |

ii. 各子关注点的候选架构模式

1. 单车数量增加导致数据库容量扩大

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 按照需求将数据库水平分区（可以按城市分区或按单车编号索引分区） | 中 | 低 | 高 |
| 2 | 将数据库垂直分区（由于位置和状态两个不同的单车属性同时应用的场景不多，将其二者分开存储） | 中 | 中 | 高 |
| 3 | 将垂直分区与水平分区共同使用 | 高 | 高 | 低 |

* 选择理由：选择1.水平分区。垂直分区：由于写操作较多，因此会同时写入两个数据库，造成写操作翻了一倍。共同使用：虽然能更大地将容量分布到不同的服务器，但是造成了不必要的成本，稳定性和系统复杂度也随之降低。水平分区：简单，表的属性少因此稳定性高。

1. 用户数量增加导致对数据库访问增多

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 换用更快的存储，提高磁盘性能 | 高 | 低 | 高 |
| 2 | 使用cache缓存经常被访问的数据。 | 低 | 低 | 高 |
| 3 | 将数据库读写分离，设置一个主数据库接受写操作，多个子数据库接受读操作。且主数据库的数据以较长的时间周期写入子数据库中。 | 高 | 中 | 高 |

* 选择理由：选择读写分离。换用更快的存储：只能暂时解决用户增加的问题，无法应对用户持续增加，成本过高。Cache存储：由于共享单车数据显然不存在访问的局部性（我们可以理解为共享单车的数据在被访问后的一段时间内，它被再次访问的概率并不会变大），因此使用cache缓存并不合理。而共享单车位置和状态写入周期是较长且固定的，而且对其的读操作更为频繁且复杂，因此使用读写分离的方案是最好的。

iii. 候选架构模式综合评估

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 单车数量增加导致数据库容量扩大 | | 用户数量增加导致对数据库访问增多 | |
| 优点 | 缺点 | 优点 | 缺点 |
| 水平分区 | 易实现  简单  稳定性高 | 表中的不同列可能被访问地不均匀，未更充分地分离 | 能够将用户的访问分散到不同数据库中，提高了吞吐量 | 无显著缺点 |
| 读写分离 | 无明显影响 | | 稳定性高，且能够充分地考虑并利用到数据库读大于写的特征 | 复杂度偏高  实现难度稍大，成本较高 |

* 选择理由：二者无冲突，且水平分区对解决数据库访问增多有明显的促进作用，因此我们选择这两种方案。

1. 针对可恢复性相关的架构驱动因素

请参照“第二次迭代”中针对“时间记录服务”分解所选择的针对可恢复性相关的架构模式

**2.1.4 第四次迭代**

* 步骤一

第四次迭代可以不用此步骤

* 步骤二

我们选择余额管理服务模块进行分解

* 步骤三

确定架构驱动因素，如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动因素 | 重要性 | 难易度 |
| 1 | 场景11：未经授权的访问 | 高 | 高 |
|  | 场景12：经过授权的访问 | 高 | 高 |
| 2 | 场景5：数据库丢失数据或程序 | 高 | 高 |

* 步骤四

选择满足架构驱动因素的架构模式

1. 针对安全性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 安全性 | 用户账户、登陆密码、支付密码、余额信息不能泄漏 |
| 只有用户本人可以对自己帐号余额，进行充值、支付操作 |

ii. 各子关注点的候选架构模式

1. 用户账户、登陆密码、支付密码、余额、不能泄漏

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 数据库存储加密数据 | 中 | 高 | 高 |
| 2 | Tokenisation | 中 | 高 | 高 |
| 3 | 检测延迟的信息传输，发现异常延迟则判断为攻击截断信息 | 高 | 中 | 高 |

* 选择理由：每次信息传输都监测传输时间成本高，判断是否为攻击实现不是很困难，这种方案防御攻击能力强，比较稳定，因此选择此策略；加密数据不会造成额外开销，但是加密算法解密算法较难设计和实现，不过只要有好的加密解密算法，数据的安全性会得到非常大的提高，即使数据库外泄，也不会有损失。Tokeniastion策略中信息传递时检测到攻击就会撤回信息，安全性高，但是实现困难。综合来看，三种策略各有优势和劣势。

1. 只有用户本人可以对自己帐号余额，进行充值、支付操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 用户－服务器身份认证 | 中 | 低 | 高 |
| 2 | 检测异常拒绝服务 | 高 | 高 | 中 |

* 选择理由：支付密码与指纹验证实现容易，占用存储空间可以接受，且稳定性高；监测余额异常操作较难定义什么样的操作是异常的，容易发生误判。因此选择策略1，即支付密码与指纹验证。

iii. 候选架构模式综合评估

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 用户账户、登陆密码、支付密码、余额信息不能泄漏 | | 只有用户本人可以对自己帐号余额，进行充值、支付操作 | |
| 优点 | 缺点 | 优点 | 缺点 |
| 数据对称加密 | 没有造成明显的时间开销和存储开销  稳定性高 | 加密解密算法有一定开销 | 保障了用户账户的安全，有效预防盗号、盗刷的行为 | 无显著缺点 |
| 传输全程延迟监测 | 稳定性高  防御效果好 | 检测时间成本开销大  有概率发生攻击误判拒绝用户正常访问 | 稳定性高  保障用户账户的信息安全  保证盗刷现象几乎不会出现 | 发生误判，会造成性能、可用性降低 |
| Tokenisation | 安全性高 | 发生误判，会造成性能降低 | 安全性高 | 发生误判，会造成性能、可用性降低 |
| 用户－服务器身份认证 | 无显著影响 | | 易于实现  稳定性高  存储开销可接受 | 无显著缺点 |

* 选择理由：四者无冲突，可作为最终可选方案。

1. 针对可恢复性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 可恢复性 | 余额管理数据库崩溃后，用户余额数据可以恢复到最近一次的变化 |

ii. 各子关注点的候选架构模式

余额管理数据库崩溃后，用户余额数据可以恢复到最近一次的变化

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 当有数据更新时就对数据库备份 | 高 | 低 | 中 |
| 2 | 纪录余额数据库每次变化的日志，根据日志恢复数据 | 低 | 高 | 高 |

* 选择理由：虽然当每次数据库变化时就对整个数据库进行备份，可以容易地实现数据库崩溃后的数据恢复，但是这样的存储成本过高，而且并不能保证备份数据库的稳定性，如果备份数据库也发生数据崩溃，就无法实现数据恢复了；使用日志记录数据库变化并根据日志恢复数据，存储成本较低，实现难度较高，但是可以使日志精确到数据库的每一次读写，数据恢复效果稳定。因此，选择策略2，即纪录余额数据库每次变化的日志，根据日志恢复数据

**2.1.5 第五次迭代**

* 步骤一

第五次迭代可以不用此步骤

* 步骤二

我们选择GPS连接服务模块进行分解

* 步骤三

确定架构驱动因素，如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
| # | 架构驱动因素 | 重要性 | 难易度 |
| 1 | 场景9：用户数量增加 | 高 | 中 |
| 2 | 场景10：单车数量增加 | 高 | 中 |
| 3 | 场景14：用户结束骑行 | 高 | 中 |
| 4 | 场景2：与GPS全球定位系统进行交互 | 高 | 低 |

* 步骤四

选择满足架构驱动因素的架构模式

1. 针对可用性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 设计关注点 | 子关注点 |
| 正确性 | 精确地定位用户的位置 |
| 精确地定位用户周边单车的位置 |
| 解锁单车时发生故障的概率 |

ii. 各子关注点的候选架构模式

1. 精确地定位用户的位置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 多次定位取均值 | 低 | 低 | 低 |
| 2 | 使用高精度的GPS定位服务 | 高 | 低 | 高 |

* 选择理由：多次定位取均值虽然成本开销低、比较容易实现，但稳定性差，效果不好。使用高精度的GPS定位服务虽然成本开销高，但开发难度低，足够稳定。由于系统需要精确、稳定的定位服务，因此我们选择使用高精度的GPS定位服务来精确定位用户的位置。

1. 精确地定位用户周边单车的位置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | 单车端内置LocalSense定位 | 高 | 低 | 高 |
| 2 | 使用高精度的GPS定位服务 | 高 | 低 | 高 |

* 选择理由：单车端内置LocalSence定位服务的成本比使用高精确的GPS定位服务成功略高，且稳定性不如GPS定位，因此我们选择使用高精度的GPS定位服务来精确定位用户周边单车的位置。

1. 解锁单车时发生故障的概率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 可实施性 |
| 1 | 使用Observe模式，单车第一次出现故障时，发送维修请求给维修人员 | 低 | 中 | 中 |
| 2 | 使用备忘录模式，单车第一次发生故障时，单车嵌入式系统自动恢复到一个正确的状态 | 中 | 中 | 中 |
| 3 | 使用observe模式+备忘录模式 | 中 | 中 | 高 |

* 选择理由：三种解决方案的成本开销、难易程度基本一致。使用observe模式，单车发生故障后，需要维修人与手动维修，中间花的时间过长，会影响单车的使用。使用备忘录模式，单车发生的故障若是物理器件的损坏，单车不能自动修复，可行性不高。使用observe+备忘录模式，单车在第一次发生故障时，使用备忘录模式尝试单车自行恢复，若自行恢复失败，则通知维修人员，可行性高。因此我们选择使用observe+备忘录模式。

iii. 候选架构模式综合评估

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 精确地定位用户的位置 | | 精确地定位用户周边单车的位置 | | 解锁单车时发生故障的概率 | |
| 优点 | 缺点 | 优点 | 缺点 | 优点 | 缺点 |
| 使用高精度的GPS定位服务 | 易实现  稳定性高 | 成本开销高 | 易实现  稳定性高 | 成本开销高 | 可用性高 | 无显著缺点 |
| observe模式+备忘录模式 | 无明显影响 | | | | 大大降低单车发生故障的概率 | 复杂度偏高，实现难度稍大 |

* 选择理由：精确地定位用户的位置、精确地定位用户周边单车的位置可同时使用高精度的GPS定位服务，即简化了系统的复杂度，又能降低系统成本。且高精度的GPS定位服务有极高的可用性、稳定性、易于实现。因此我们采用高精度的GPS定位服务。Observe模式+备忘录模式具有较高的可行性，且成本低，因此我们采用此方案。

1. 针对可伸缩性相关的架构驱动因素

i. 设计关注点

|  |  |
| --- | --- |
| 关注点 | 子关注点 |
| 高并发的处理能力 | 适应突然暴涨的用户人数 |
| 适应突然暴涨的单车数量 |

ii. 各子关注点的候选架构模式

1. 适应突然暴涨的用户人数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | MapReduce | 高 | 中 | 低 |
| 2 | BigTable | 高 | 高 | 高 |

* 选择理由：MapReduce的实现难易程度虽然不高，但是其稳定性差，出错被认为是很寻常的事，因此我们在这里选择BigTable

1. 适应突然暴涨的单车数量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | 模式名称 | 成本开销 | 难易程度 | 稳定性 |
| 1 | MapReduce | 高 | 中 | 低 |
| 2 | BigTable | 高 | 高 | 高 |

* 选择理由：同上选择理由

iii. 候选架构模式综合评价

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 适应突然暴涨的用户人数 | | 适应突然暴涨的单车数量 | |
| 优点 | 缺点 | 优点 | 缺点 |
| MapReduce | 实现较为容易 | 成本高且不稳定 | 实现较为容易 | 成本高且不稳定 |
| BigTable | 稳定性高 | 成本高且难以实现 | 稳定性高 | 成本高且难以实现 |

* MapReduce、BigTable都适合处理高并发，都能满足可伸缩性的要求。两种架构的成本开销都较高。但由于系统需要较好的稳定性，而MapReduce又经常容易发生错误，因此，在这里我们选择BigTable架构。

**2.2 ATAM评估过程**

**2.2.1 质量属性效用树**

|  |  |  |
| --- | --- | --- |
| 质量属性 | 属性求精 | 场景 |
| 可恢复性 | 数据恢复 | 数据库恢复时间不超过10分钟（H,H） |
| 系统故障恢复 | 重启、从错误中恢复的时间小于1分钟(H,H) |
| 可伸缩性 | 并发量 | 可支持的用户的并发量不低于30000(H,H) |
| 可支持单车的并发量不低于10000(H,H) |
| 并发请求响应速度 | 用户请求服务的平均响应时间不超过5秒(M,M) |
| 并发请求成功率 | 最高并发中，用户成功获得响应的概率不低于95%(M,M) |
| 能成功报告自己位置的单车不低于95％(M,M) |
| 安全性 | 机密性 | 系统验证用户身份，拒绝不在用户权限范围内的服务和数据，并通知用户(H,M) |
| 完整性 | 拒绝未授权的访问(H,H) |
| 互操作性 | 数据接口 | 与用户所选的支付APP进行数据传输，交互成功率达到100%(H,M) |
| 与GPS全球定位系统交互，误差在20米之内的交互成功率到达99%(H,M) |
| 易用性 |  | 用户完成操作时间如注册，借车操作时间不超过3分钟（M,M） |
| 用户操作成功率高于99%(M,M) |
| 可用性 | 可用时长 | 单车端嵌入式系统无故障运行时间大于1个月(M,H) |
| 故障处理 | 系统发生故障可以在1分钟内恢复，单车发生故障可以及时通知维修人员维修(M,M) |
| 可扩展性 | 扩展后大小 | 更新APP大小不超过原有的10%(M,M) |
| 扩展影响 | 所影响到的源代码量不超过2%(M,M) |
| 性能 | 服务请求响应时间 | 用户正常使用客户端，操作响应的时间小于1秒(M,M) |
| 用户使用客户端通过网络访问，在不涉及外部系统的情况下获得服务器响应时间小于2秒(M,M) |

**2.2.2 敏感点列表**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 架构决策 | 相关质量属性 | 场景编号 |
| S1 | 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | 可恢复性 | A1 |
| S2 | 记录余额数据库每次变化的日志，根据日志恢复数据 |
| S3 | 本地提前开发相应应急服务 | A2 |
| S4 | 本地购买备用主机 |
| S5 | BigTable | 可伸缩性 | A2 |
| S6 | 数据库水平分区 | A3 |
| S7 | 数据库读写分离 |
| S8 | 数据对城加密存储 | 安全性 | A5 |
| S9 | 传输全程延迟检测 | A6 |
| S10 | Tokenisation |

**2.2.3 权衡点列表**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 架构决策 | 相关质量属性 | 场景编号 |
| T1 | 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | 可恢复性、可伸缩性 | A1 |
| T2 | 数据对称加密存储 | 性能、安全性 | A3 |
| T5 | 传输延迟检测 | 性能、安全性 | A3 |
| T6 | Tokenisation | 性能、安全性 | A3 |

**2.2.4 有风险决策列表**

|  |  |  |
| --- | --- | --- |
| 编号 | 架构决策 | 风险 |
| R1 | 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | 数据量大，维护副本成本高 |
| R2 | 纪录余额数据库每次变化的日志，根据日志恢复数据 | 短期恢复数据困难大，有失败的可能 |
| R3 | 传输全程延迟检测 | 降低系统性能，时间监测成本高 |
| R4 | Tokenisation | 降低系统性能，信息传输中异常检测撤回机制较复杂，可能会影响性能 |

**2.2.5 无风险决策列表**

|  |  |
| --- | --- |
| 编号 | 描述 |
| N1 | 数据库水平分区 |
| N2 | 数据库读写分离 |
| N3 | 本地提前开发相应应急服务 |
| N4 | 本地购买备用主机 |
| N5 | 用户－服务器身份认证 |
| N6 | BigTable |
| N7 | 使用高精度的GPS定位服务 |
| N8 | observe模式+备忘录模式 |

**2.2.6 架构方法分析**

| 场景号：A1 | 数据库恢复时间 | | | |
| --- | --- | --- | --- | --- |
| 属性 | 可恢复性 | | | |
| 环境 | 系统暂停运行或运行时 | | | |
| 刺激 | 由于服务器宕机、损坏或遭受攻击等原因造成的数据或者程序丢失 | | | |
| 响应 | 查明数据丢失原因  将数据库快速恢复到尽可能新的时间点 | | | |
| 构架决策 | 敏感点 | 权衡点 | 有风险决策 | 无风险决策 |
| 采用分布式数据库保存部分区域数据和全部数据的完整拷贝 | S1 | T1 | R1 |  |
| 理由 | 分布式数据库保存区域数据和全部数据的副本方式保证了数据的完整性，但是由于数据量大，维护副本的成本太高，访问数据时的网络带宽延迟也会带来服务延迟。由于涉及到关键数据，为了保证数据的正确性，需要采取这种方式 | | | |

| 场景号：A2 | 系统崩溃 | | | |
| --- | --- | --- | --- | --- |
| 属性 | 可恢复性 | | | |
| 环境 | 系统运行时 | | | |
| 刺激 | 某服务失效或者系统崩溃、宕机 | | | |
| 响应 | 查明故障原因  重启故障部分  恢复到故障前的状态 | | | |
| 构架决策 | 敏感点 | 权衡点 | 有风险决策 | 无风险决策 |
| 本地购买备用主机 | S4 |  |  | N4 |
| 理由 | 本地主机存储会极大地增加成本开销，造成不必要的资源闲置，为了保证服务器崩溃时能够及时转接服务，本地主机架构的采取很有必要性 | | | |

| 场景号：A3 | 用户并发量支持 | | | |
| --- | --- | --- | --- | --- |
| 属性 | 可伸缩性 | | | |
| 环境 | 系统已上线运行 | | | |
| 刺激 | 同时使用查询附近车辆、借车功能 | | | |
| 响应 | 正常进行车辆位置查询，正常借车 | | | |
| 构架决策 | 敏感点 | 权衡点 | 有风险决策 | 无风险决策 |
| 将数据库读写分离，设置一个主数据库接受写操作，多个子数据库接受读操作。且主数据库的数据以较长的时间周期写入子数据库中 | S7 |  |  | N2 |
| 理由 | 共享单车位置和状态写入周期是较长且固定的，而且对其的读操作更为频繁且复杂，因此使用读写分离的方案，分离读操作和写操作，写操作可以在每半个小时单车位置更新后进行，从而避免了对数据库大量的并发访问 | | | |

| 场景号：A4 | 单车并发量支持 | | | |
| --- | --- | --- | --- | --- |
| 属性 | 可伸缩性 | | | |
| 环境 | 系统已上线运行 | | | |
| 刺激 | 同时报告自身位置的单车数量增加 | | | |
| 响应 | 单车可以成功报告自己所在位置 | | | |
| 构架决策 | 敏感点 | 权衡点 | 有风险决策 | 无风险决策 |
| 按照需求将数据库水平分区（可以按城市分区或单车编号索引分区） | S6 |  |  | N1 |
| 理由 | 由于写操作较多，因此会同时写入两个数据库，造成写操作翻了一倍。共同使用：虽然能更大地将容量分布到不同的服务器，但是造成了不必要的成本，稳定性和系统复杂度也随之降低。水平分区：简单，表的属性少因此稳定性高。 | | | |

| 场景号：A5 | 未授权的访问 | | | |
| --- | --- | --- | --- | --- |
| 属性 | 安全性 | | | |
| 环境 | 系统联网状下运行 | | | |
| 刺激 | 试图显示数据或者改变/删除数据，试图改变系统服务 | | | |
| 响应 | 识别请求源是否被授权  阻止对服务或数据的访问  以一种不可读的方式存储数据  数据或者服务遭到破坏后可恢复 | | | |
| 构架决策 | 敏感点 | 构架决策 | 敏感点 | 构架决策 |
| 数据库存储加密数据 | S6 |  |  |  |
| 检测延迟的信息传输，发现异常延迟判断为攻击截断信息 | S7 | T5 | R3 |  |
| 理由 | 每次信息传输都监测传输时间成本高，判断是否为攻击实现不是很困难，这种方案防御攻击能力强，比较稳定，因此选择此策略；加密数据不会造成额外开销，但是加密算法解密算法较难设计和实现，不过只要有好的加密解密算法，数据的安全性会得到非常大的提高，即使数据库外泄，也不会有损失。Tokeniastion策略中信息传递时检测到攻击就会撤回信息，安全性高，但是实现困难。 | | | |

**一级标题 三号+粗**

**二级及以下标题 四号+粗**

**正文 小四**

**表格标题行宽为1**

**正文行宽、表格内容行宽为1.5**