验证码/方式破解统一服务系统（CCS）

架构设计

**一、业务需求描述**

为公司互联网抓取系统提供统一的验证码识别服务入口，以便于抓取程序的统一调用。公司验证码识别（破解）需求包括本地验证码识别服务的调用（工商验证码识别HTTP服务实际已废弃，此处是泛指未来可能有的新的服务）和第三方在线验证码识别服务。

**二、需求分析**

抓取系统在访问特定网站页面时，会出现验证页面，要求访问用户证明自己是自然人。

不同的网站会有不同的验证方式，所以验证破解也会有不同的实现。不同网站的访问速度、频率的要求都不一样，所以对验证破解程序的资源要求也不一样，对验证破解服务的可靠性要求也不一样。

抓取系统的业务场景决定了允许验证破解过程存在一定的技术性失败，比如验证码破解结果错误、验证码破解程序或服务无法访问或访问超时等。这意味着架构设计时不用考虑失败重做机制，也不必太强调负载均衡和高可用性。

考虑到验证方式的多样性，架构设计时要对支持的验证方式进行限定。

考虑到未来会引入新的第三方在线服务或者本地程序（含本地库），所以本地服务不能限定死开发语言和运行环境，但是需要符合特定的开发规范。

考虑到公司现有滑动验证码服务（由孙亚星负责）的设计本意是模拟浏览服务，在工商采集过程中该服务对上下文页面访问存在依赖，且大大超出单纯验证码破解服务的需求范围，故暂不纳入本架构考虑范畴。待和其他同事进一步沟通清楚后再做处理。

公司的抓取系统除了给自己用外，还会部署到企业客户环境中。但客户环境一般应无特别要求，故不做特殊设计。

**三、设计边界和前置约定**

1、本架构仅限于支持纯线上交互验证方式（如：常规字符图片识别、图片文字点击验证、单纯滑块拖动验证、极验验证码、算术题、常识问答题、常识填空题等）的工作场景，需要进行线下交互验证方式（如：短信验证码、语音验证码、手势验证码、信令交互认证等）不在架构考虑范围之内。

2、本架构仅提供上述验证方式破解的服务统一封装、请求分发、压力控制、状态监控等功能，不包括各验证方式破解的具体实现，同样也不对破解是否成功做出保证。

3、基于对公司未来两年内（数据抓取需求中）验证码服务调用需求的粗略估算，本架构理论设计能力应达到0.5~2万次/分钟的请求频率（相当于720~2880万次/天）。如果服务调用需求大大超过估算值，则本架构不一定能达到足够的稳定性和可用性等。且最终性能和硬件配置、服务请求类型的分布、破解程序的执行效率等外部因素都有很大相关性，故此处给出的值只是设计参考值，不作为验收和考察标准。

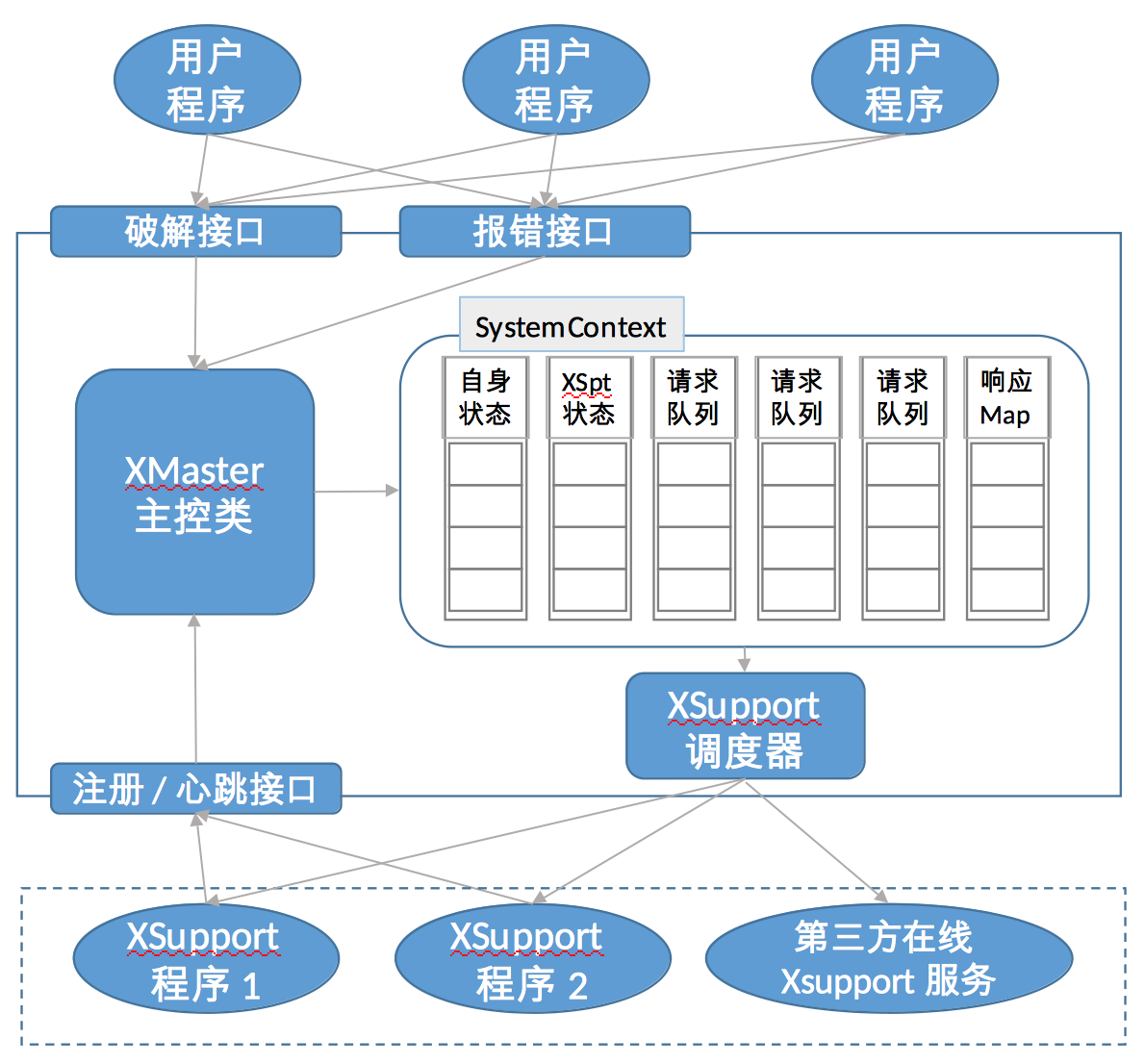
4、为了保证高优先级用户端的服务质量，出于验证码识别允许技术性失败（如服务超时、无可用处理程序实例等）的业务实际，本架构不保证低优先级请求的服务可用性，即会在高并发压力情况下适当丢弃部分低优先级请求。

5、本架构不对验证码破解程序的开发语言做出要求，但是该程序需要符合后续的开发规范的要求。

6、由于本架构对应程序实例部署在局域网的可信任环境中，且非核心业务服务，不存在数据泄密的风险，为了降低系统复杂性，故暂不考虑添加安全机制。

**四、架构概述**

1、逻辑架构图



注：为了便于理解，架构图中的箭头只标注为单向箭头，意指请求数据处理的方向。实际运行过程中，响应数据是按反方向进行流转的。

2、主要对象及概念

用户：使用本服务的各个程序单元。如果一个程序中有n处验证码服务的调用，则应理解为是n个不用的用户。

用户等级：为了区分用户的重要程度，引入了用户服务等级概念。目前用户只分为1、5、9三个服务等级（0~9的其他数字只是预留空间），9为高服务等级的不可抛弃用户，1为低服务等级的可抛弃用户（高压力情况下会优先被抛弃），5为中等服务等级的可抛弃用户。为了简化设计，用户等级由开发人员自行酌情给定（大部分用户等级应该为1和5，不建议设置太多9），且在调用过程中直接传入。

破解(CC)请求：CAPTCHA Crack请求，请求参数包括：服务等级、XSupportClass、params、可接受的超时时间。不同类型的XSupport需要不同的params，比如普通字符验证码图片只需要传入图片内容和验证码分类类型，而九宫格验证方式需要传入两张图片的内容和验证码分类类型。

破解(CC)响应：CAPTCHA Crack请求 对应的响应对象，包括处理状态、描述信息、处理结果等信息。不同的处理流程（包括异常情况）会返回不同的处理状态和描述信息。不同的XSupport会返回不同的处理结果，比如普通字符验证码图片返回的是字符串，而九宫格验证方式返回的是鼠标点击坐标序列。

破解服务接口：CC请求以HTTP Post方式提交，拆解并构造为CC请求对象后，提交给XMaster主控类进行处理。

报错(ER)服务接口：ER请求以HTTP Post方式提交，拆解并构造为ER请求对象后，提交给XMaster主控类进行处理。

注册/心跳服务接口：请求以HTTP Post方式提交，拆解并构造为注册请求对象后，提交给XMaster主控类进行处理。如果该XSupport实例是之前注册过的，则该请求只会更新SystemContext的XSupport列表中的最后响应时间，等效于心跳操作。

XMaster主控类：主控类对非法CC请求（比如无效的XSupportClass）、无效CC请求（比如无可用XSupport实例）、超时CC请求直接返回报错信息；对达到SQA条件的请求进行特殊处理（参见后续“异常情况下的系统运行状况”中的说明）；然后将正常的请求加入CC请求队列中。主控类还会轮询CC响应Map，获取到CC响应后，将CC响应重新封装为HTTP Response然后返回给用户。

主控类还对ER请求和心跳请求做即时处理，ER请求会记录到本地或反馈给第三方服务平台；心跳请求会更新到SystemContext中。

CC请求队列：用户请求先被放置到请求队列中进行缓存，请求队列按服务等级分为1、5、9三条队列。队列可以是本地队列（推荐方式），也可以是异地队列服务。该队列不应过长，过长的无法及时处理的队列在验证码场景中是没有意义的。同样道理，队列的持久化也没有意义。

CC响应Map：破解响应被放置到Map中等待处理。因为在请求过程中已经进行了流量控制，所以响应是数量是有限的，为了加快查找速度，所以采用了Map数据结构而不是队列。

ER请求队列：用户报错请求先被放置到ER请求队列中进行缓存，调度者会自行读取该请求并进行分发。

XSupport调度器：对请求队列中的请求进行分发处理，然后将破解响应存放到CC响应Map中。高等级的队列的调度间隔会更短。

SystemContext：包括多个请求队列、CC响应Map、可用破解程序实例状态汇总、用户访问情况汇总、报错信息汇总等。

破解程序（XSupport）：破解程序不限定语言类型和运行环境，既可以是Python程序，也可以是Java程序或.net程序，也可以是封装后的第三方在线服务等。正因为如此，所以不提供XSupport的任何抽象实现，只要求符合下述调用规范即可。

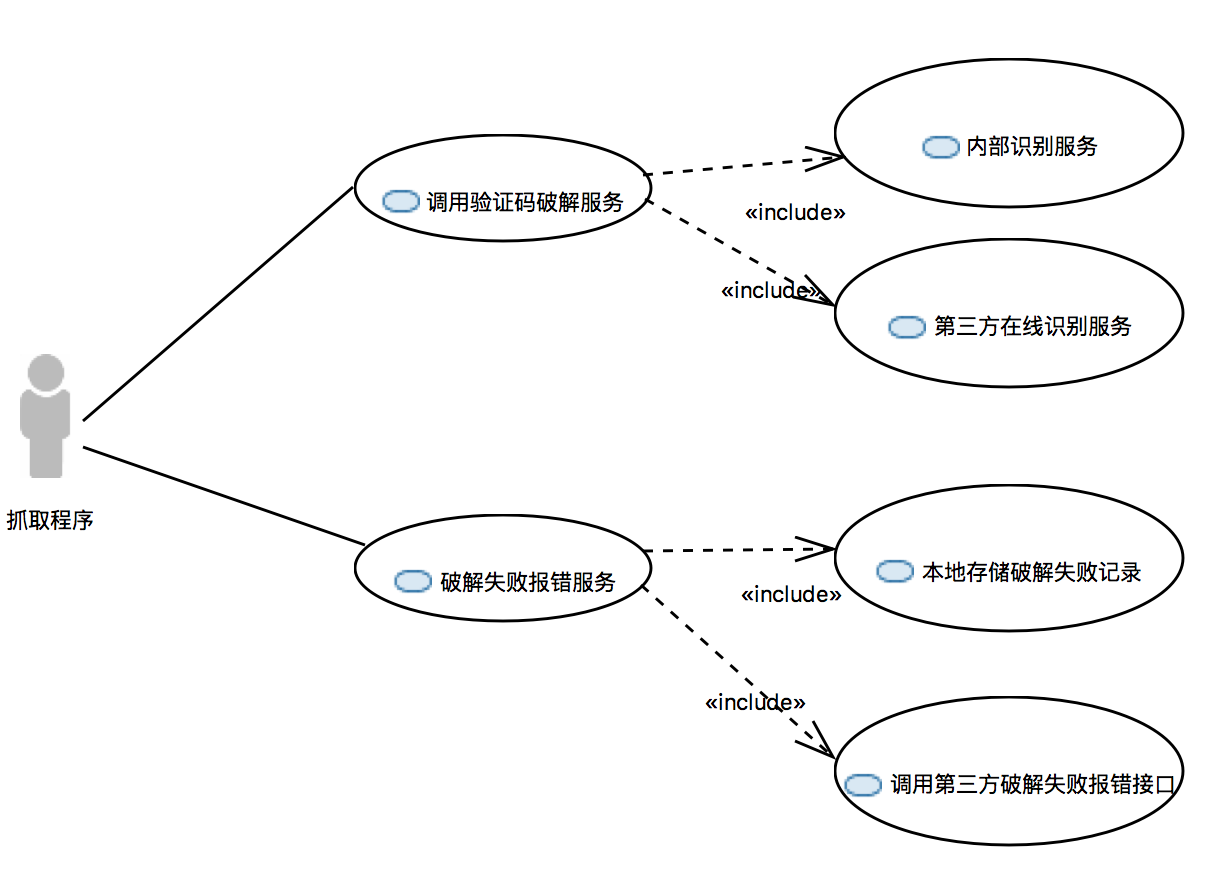
规范包括：

A）为了简化调用方式，XSupport统一通过HTTP接口提供破解调用。第三方在线服务的封装XSupport实例还需要提供报错接口。

B）为了知道有哪些XSupport程序是存在或者存活的，需要引入注册和心跳机制，并且为了简化接口实现，将两个机制合并为一个接口调用。对于第三方在线服务，采用静态注册机制，在系统启动时从配置文件中读取该注册信息，同时意味着不用进行心跳；其他局域网内的XSupport采用主动注册机制，根据配置文件中的服务器地址和端口主动向服务器注册自己，同时在后续心跳过程中访问同一接口。

3、基本业务流程

基本的调用破解服务和报错服务的用例如下：



用户程序通过HTTP接口调用CC服务接口，请求会被拆解封装为CC请求对象传递给XMaster主控类，主控类会判断CCS系统状态、XSupport实例状态，如果整体状况正常，主控类会将CC请求放入CC请求队列中。XSupport调度者随后从CC请求队列中获取该对象，并分发给特定的XSupport实例，XSupport计算完成并返回CC响应后，调度者会将CC响应放入CC响应Map中。主控类轮询到CC响应，会将CC响应重新封装为HTTP Response并返回给用户。

如果用户判断该破解结果无法通过验证，则调用ER服务接口。请求会被拆解封装为ER请求对象传递给XMaster主控类，主控类会存储报错信息，如果是外部第三方在线服务，主控类还会发送到ER请求队列，交由调度器进行分发。

另外，根据前文所述设计思路，CCS服务程序自身无需实现复杂的分布式架构，而由部署在不同机器上的破解程序实例实现分布式计算；较高并发的请求将缓存在本地或异地的队列中，在验证码破解的业务场景下，分布式队列没有太多意义，故不考虑分布式队列。

4、架构特性一览

|  |  |  |
| --- | --- | --- |
| 特性名称 | 特性说明 | 必要性 |
| 统一服务入口 | 支持不同语言/操作系统的验证码服务实例 | 必要 |
| 分级服务 | 对不同等级用户提供差异化服务 | 必要 |
| 分布式破解 | 可以开启多个同类XSupport实例进行服务，具体有调度器进行分发 | 必要 |
| 验证失败报错功能 |  | 必要 |
| 线性扩展 | 可以线性扩展XSupport实例 | 必要 |
| 自动注册/心跳 | 本地XSupport实例主动向CCS发起注册/心跳请 | 必要 |
| 服务质量保证 | Service Quality Assurance，是指对不可抛弃的高等级用户，要么不接受请求（超系统负荷的高压力情况下，无法处理所有的高等级请求），要么接收之后必须处理（即使返回‘没有可用的破解程序实例’，也算是处理了）。当然，我们应该尽量保证破解程序存在多份可用实例。 | 必要 |
| 状态异常告警 | 在系统出现异常的情况下，进行主动的告警 | 必要 |
| 日志 | 系统可根据设定记录CCS运行日志 | 必要 |
| 管理界面 | 可以总览系统状况，便于发现问题 | 可选 |
| 快速服务 | 高等级只提供了对服务质量的保证，虽然服务速度会快一些，但可能仍然无法保证快速处理。可以考虑再添加一种新的等级来做无延迟的快速处理。 | 可选 |

5、异常情况下系统运行状况

|  |  |  |
| --- | --- | --- |
| 分类 | 场景描述 | 系统状况 |
| 高压力场景 | 9级请求很多 | 存量1、5级请求完全废弃，增量请求不予接收；  存量9级请求正常处理，增量9级请求可暂停接收或部分接收；  如果高压力持续3mins以上 ，发出黄色警告；如果高压力持续10mins以上，发出红色警告； |
| 5级请求很多 | 保持存量和增量9级请求正常处理；  部分或全部放弃1级请求；  存量5级请求正常处理，增量5级请求可暂停接收或部分接收；  如果高压力持续3mins以上 ，发出黄色警告；如果高压力持续10mins以上，发出红色警告； |
| 1级请求很多 | 保持存量和增量5、9级请求正常处理；  存量1级请求正常处理，增量1级请求可暂停接收或部分接收；  如果高压力持续3mins以上 ，发出黄色警告；如果高压力持续10mins以上，发出红色警告； |
| 混合情况 | 优先保证高等级存量和增量请求的执行；  发出系统警告，进行人为干预； |
| 网络异常场景 | CCS服务掉线 | 用户程序无法连接到CCS服务 |
| XSupport掉线 | 该XSupport实例无法访问，CCS中会返回给用户“无可用XSupport程序实例”或“处理超时”信息。 |
| 程序更新场景 | 更新CCS服务程序 | 用户程序无法连接到CCS服务 |
| 更新XSupport | 该XSupport实例无法访问，CCS中会返回给用户“无可用XSupport程序实例”或“处理超时”信息。 |

6、系统部署

