MProxy项目日志

|  |  |
| --- | --- |
| 文档状态 | 撰写中 |
| 负责人 | mrbcy |
| 生效日期 |  |
| 文档概述 | 本文档用于记录Mproxy项目的开发过程。包括系统设计、开发过程中的决策、做出决策的依据。为今后的维护、扩展提供参考。 |

|  |  |  |
| --- | --- | --- |
| **文档变更记录** | | |
| 日期 | 变更人 | 变更内容 |
| 2017-02-25 | mrbcy | 添加了第1章和第2章中系统设计的部分。 |
| 2017-03-03 | mrbcy | 添加了2.2.1、2.2.2、2.3.1、3.1.1部分 |
| 2017-03-20 | mrbcy | 修改了2.1.4，添加了2.1.5、2.2.3、2.2.4、3.2、3.3 |

# 第1章 概述

这几天没什么时间用电脑，所以有很多时间自己想事。突然想做一个提供代理服务器的服务。客户通过简单的API就可以拿到一些代理服务器用于网站测试等用途。

当然，代理服务器地址是要从网上来收集的。然后经过系统的测试、整理存放在数据库中。最好还要将过程中的信息比如说代理服务器所属的地区、爬取的时间、最后一次验证可用的时间等数据用日志记录下来，方便今后进行统计分析。

所以，第一期的任务就是开发一套代理服务器的收集系统，包括爬虫、验证器、数据交互、任务调度、数据存储部分。至于所用的技术，爬虫部分当然是用Python。验证器用Python实现也很方便。数据交互部分准备是用Kafka，也支持Python操作。任务调度部分要用ZooKeeper，也支持Python。数据存储准备使用MySQL，也支持Python。Python上也有强大的日志模块，能够完成日志记录的需求。因此这套系统准备用Python来开发。

# 第2章 系统设计及技术验证

## 2.1 系统设计

关于Kafka和ZooKeeper的简单介绍可以参看[这篇博客](http://www.cnblogs.com/likehua/p/3999538.html)。接下来给出系统的顶层设计并作一些简要的说明。

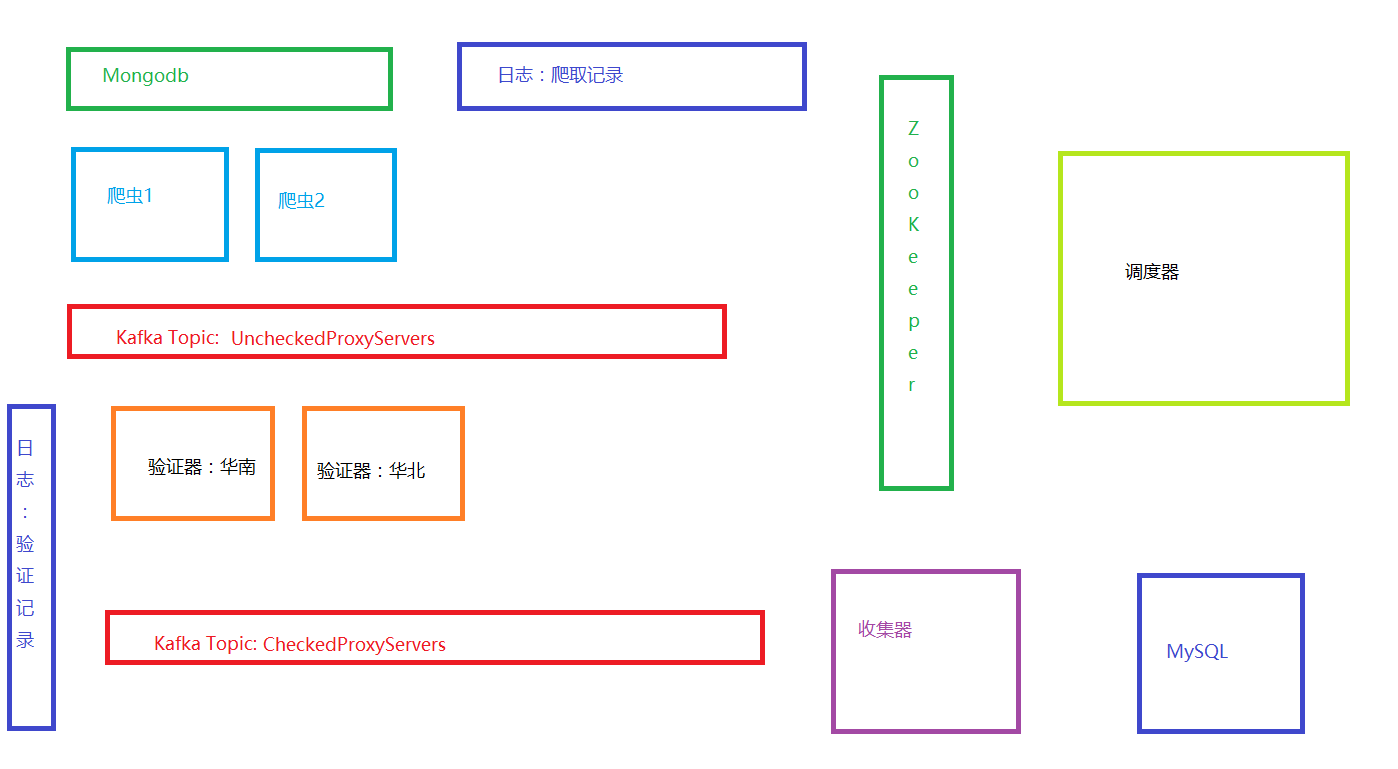


图2. 1系统结构图

整个系统我们需要实现的部分包括爬虫、验证器、收集器和调度器四部分，下面分别阐述一下它们的工作流程，来说明一下上面的系统结构图。

### 2.1.1爬虫

爬虫的任务是从网页上提取出代理服务器列表，并把提取出来的代理服务器提交到Kafka上，同时记录日志信息。

爬虫的工作流程如下：

1. 爬虫启动后，从指定的首页开始，爬取详情页地址列表。
2. 对每一个详情页地址，先去MongoDB里面查询一下是否已经爬取过该地址，如果没有爬取过，就把这个地址加入任务列表。
3. 从任务列表中取出一个任务，开始爬取详情页中的代理服务器地址，将提取出的地址信息发送到Kafka集群中，同时记录爬取日志。然后将任务地址保存到MongoDB标注为已爬取状态。

**关于使用MongoDB的原因：**因为查询是否爬取一个地址是一个高频操作。如果使用MySQL等传统数据库的话，一条条的查询效率会很低而采用批量查询加缓存的方式又会占用很多内存。而MongoDB对查询的优化是非常好的，查询效率非常高，因此选用MongoDB。

**关于Kafka中Topic过期时间的选择。**有时候不能保证爬取的同时验证器也在工作，所以应该把Kafka中的爬取记录Topic的过期时间选得长一些。初步打算是选择3个月，等以后积累了大量的数据，能得出代理服务器的平均可用时间，然后再进行优化。

### 2.1.2 验证器

验证器主要负责测试爬虫爬到的代理服务器是否可用，并将可用的代理服务器提交到Kafka集群中，并记录验证日志。

验证器的工作流程如下：

1. 启动后先向ZooKeeper注册自己的信息。
2. 启动后先从ZooKeeper集群中获取调度器发布的工作状态，并添加监听。如果工作状态不存在或指定不工作，那么进行等待。如果指定工作，那么开始进行验证工作。
3. 从Kafka集群中取出一条未验证的代理服务器地址，通过这个代理服务器访问指定网页，然后提取特征判断是否访问成功。
4. 将验证结果提交到Kafka集群。同时记录验证日志。

**关于验证器的数量。**在项目的设计中，验证器用于确保爬取到的代理服务器可用。因此至少有两个验证器，并且需要分布在不同的地区（图中的华南和华北）。

**关于工作状态。**上面关于验证器的数量的讨论决定了每个代理服务器必须通过全部验证器的验证才能被收录到数据库中，因此如果有验证器没有启动，其他的验证器是没必要进行工作的。调度器会监听ZooKeeper集群上验证器的状态信息，只有当所有的验证器都在线的时候才会将工作状态设定为工作。

### 2.1.3 收集器

收集器主要负责把验证器提交到Kafka集群的数据提取出来，并将通过所有验证器验证的代理服务器地址写入到MySQL中。

收集器的工作流程如下：

1. 启动后从配置文件中获得所有的验证器的列表。
2. 从Kafka集群中取得已通过验证器验证的代理服务器地址。
3. 以任务号为单位，将同一任务号的验证结果放在一起。
4. 将任务结果写入MySQL。对于失败的任务，更新其状态信息为“暂时不可用”

**关于配置文件中的验证器列表。**从配置文件中读取验证器的列表使得收集器可以独立于验证器的工作状态进行工作。即使验证器没有全部启动，还是可以进行收集工作，我认为这是比较好的。

**关于任务的超时时间。**在收集器收到一个任务的第一条记录时记录一个时间戳。然后后台线程扫描整个任务列表，将超过12小时还没有收到所有验证器验证成功记录的任务判定为失败，清除掉这个记录。

### 2.1.4 调度器

调度器主要负责对验证器的工作状态进行控制，同时负责定时将需要重新验证的代理服务器地址放入Kafka集群中。

调度器的工作流程如下：

1. 启动后获取ZooKeeper中的验证器列表并创建监听。如果所有的验证器都在线，更新验证器工作状态为工作。否则，更新为不工作。
2. 启动扫描线程，定时将上次验证时间超过24小时的代理服务器地址提交到Kafka集群中，等待验证器验证。

### 2.1.5 运行监测器

运行监控器是在开发过程中分化出来的，我把调度器的功能分了一部分出来，交给运行监测器负责。运行监测器的主要任务是监测两个核心的系统状态是否正常，第1个是当前可用的代理服务器数量，另一个是所有代理服务器中最近的验证时间。

只要这两个值不异常，即使系统宕机也没关系。如果异常，就通知管理员来处理。

针对这两个指标，提出了如下的监测方案。

1. 如果发现可用的代理服务器数量少于4000，就运行爬虫进行爬取。这个检查2小时执行1次。

2. 如果last\_validate\_time中的最晚时间距今超过24小时，发送短信提醒。这个检查6小时执行1次。

3. 如果发现可用的代理服务器数量少于2000，发送短信提醒。这个检查2小时执行1次。

4. 如果在执行上述查询的过程中抛出异常，也发送短信提醒。

## 2.2 技术验证

经过上面的说明，系统的整体概况已经比较清楚了。但因为我不太熟悉Python，因此还有大量的技术验证工作要进行。

### 2.2.1 Scrapy的使用及Python与MySQL的交互

Scrapy是非常流行的Python爬虫框架。我们准备使用这个框架进行代理服务器信息的收集。

在开始之前，我们先要找几个目标网站。当然开始我们不会用这几个网站上手。先试一下简单的。

<http://www.kuaidaili.com/free/inha/>

<http://www.xicidaili.com/nn/>

<http://www.youdaili.net/Daili/http/>

关于Scrapy的使用，我专门做了一个例子：[..\技术验证文档\使用Scrapy爬取笑话并存储到文件和MySQL.docx](../技术验证文档/使用Scrapy爬取笑话并存储到文件和MySQL.docx)，可以作为参考。里面也包含了Python与MySQL的交互

### 2.2.2 Python与Kafka集群的交互

Kafka是非常流行的分布式消息系统，我们的系统将依赖它来交换包括爬取到的代理服务器地址、代理服务器的验证结果在内的消息。

关于Python与Kafka集群的交互，我也专门做了一个例子：[..\技术验证文档\Python与Kafka集群的交互.docx](../技术验证文档/Python与Kafka集群的交互.docx)。同时也写了专门的技术验证代码。

### 2.2.3 Python与MongoDB交互

MongoDB是非常著名的NoSQL数据库，是一个内存数据库，速度非常快，特别适用于频繁的查询场景。我们的系统依赖它来存储已经爬取的代理服务器数据。

关于Python与MongoDB的交互，我也专门做了一个例子：[..\技术验证文档\Python操作MongoDB数据库.docx](../技术验证文档/Python操作MongoDB数据库.docx)。同时也写了专门的技术验证代码。

### 2.2.4 Python中使用配置文件

为了今后使用、部署的方便，我们项目中大量的数据使用配置文件信息的方式进行保存。关于Python中配置文件的使用，我专门做了一个例子：[..\技术验证文档\Python中配置文件的使用.docx](../技术验证文档/Python中配置文件的使用.docx)。

## 2.3 小结

经过上面的技术验证工作，我对几个问题又有了新的思路。记录下来。

### 2.3.1 Kafka使用的思考

先来来思考一下项目的正常数据流，看看应该如何使用系统结构图中的两个Kafka Topic。

首先是爬虫启动，然后将爬取到的代理服务器信息写入到UncheckedProxyServers这个Topic中。并为每一个代理服务器信息指定一个任务号taskId。

然后所有的验证器都作为UncheckedProxyServer的消费者，不用指定每次从头开始，就接收没有处理过的消息就可以（丢掉的代理服务器信息就让它丢掉好了）。同时必须保证在验证过程中即使发生错误，也要正常运行，不允许崩溃。验证器在对代理服务器进行验证后将验证结果（代理服务器信息，验证结果）写入到CheckedProxyServers主题中。

收集器作为CheckedProxyServers主题的消费者，将同一个taskId的验证结果放在一起进行比对。如果所有的验证器都校验通过，则更新数据库中该代理服务器的状态为“可用”，并更新最后校验时间，重置重试次数为0。如果有至少一个验证器验证失败，就更新该代理服务器的状态为“暂时不可用”，并把重试次数+1。同时收集器后台运行一个线程，定时扫描任务结果缓存区。如果一个任务在12小时后还没有收到所有验证器的验证结果就判定该任务失败，清除掉缓存中的验证结果。

调度器需要每隔一段时间就选一批代理服务器写入到UncheckedProxyServers主题中进行重新验证。同时需要运行一个后台线程来跟踪验证结果。如果一个服务器被指定重新验证，而12小时之后还没有收到验证结果，就判定验证失败，将代理服务器状态更新为“暂时不可用”。如果重试3次代理服务器仍不可用，就将其状态更新为“永久不可用”，今后不需要再次重试。

好，现在如何使用Kafka集群的结论已经很清晰了。所有的爬虫作为UncheckedProxyServers主题的producer。所有的验证器作为UncheckedProxyServers主题的consumer，同时作为CheckedProxyServers主题的producer。收集器作为CheckedProxyServers主题的consumer。

# 第3章 系统开发概述

本章只说明比较开发过程中遇到的比较重大的问题。具体的开发流程由其他文档进行说明，本章只给出文档的具体路径。

## 3.1 爬虫

### 3.1.1 Kuaidaili爬虫

这个爬虫负责爬取快代理网站的数据。这个网站使用了比较独特的反爬虫机制，需要等待js执行完毕之后才能得到真实的页面数据，从而进行爬取。为了达到这个目标，我使用了Splash来渲染js。具体参见文档：[..\开发文档\快代理爬虫开发.docx](../开发文档/快代理爬虫开发.docx)。这个文档中还包括了日志模块的使用以及与Kafka集群的交互。

## 3.2验证器

验证器负责检验爬虫收集到的代理服务器是否可用。最新的验证方式是访问我博客的页面，然后从返回的页面中查找我的昵称”mrbcy”。一个重点是验证器启动后会连接ZooKeeper集群来接收来自调度器的控制信号来确定自己是否应该开始工作，同时，也会向ZooKeeper集群注册来告知调度器自己的在线状态。

## 3.3 运行监测器

遇到的问题主要是编码。统一指定编码格式后就可以了。另外为了保证App key和Secret的安全，要记得把密码等信息放到配置文件里面。