中期报告

# What

## 项目概述

本项目基于2023年My-Glow小组项目进行优化，主要应用Nginx将项目的中央服务器变成分布式，并进行反向代理。除此之外，尝试对文件的打标技术用大模型进行优化，从而简化打标模型，实现多模态。

## 核心概念相关介绍：

**什么是图数据库：**

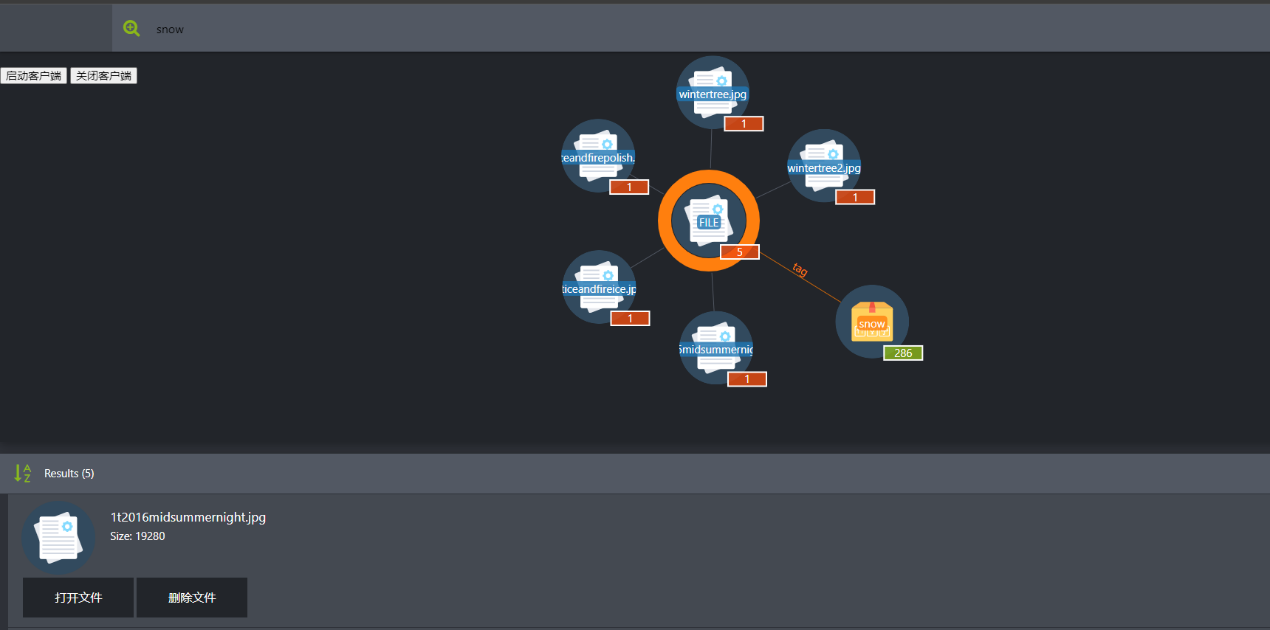
图数据库（graph database，GDB）是一种使用图结构进行语义查询的数据库。它使用节点、边和属性来表示和存储数据。，它直接将存储中的数据项，与数据节点和节点间表示关系的边的集合相关联。这些关系允许直接将存储区中的数据链接在一起，并且在许多情况下，可以通过一个操作进行检索。

分布式图数据库在图数据库的基础上增加了分布式部署的功能

**什么是图文件系统：**

DisGraphFS是一个分布式图文件系统，他将图结构的思想应用到分布式文件系统上面，不同于一般的文件系统，该系统用图的结构来描述文件之间的关系，

这是其效果图，可以看到这个图结构更加更符合人类思维。

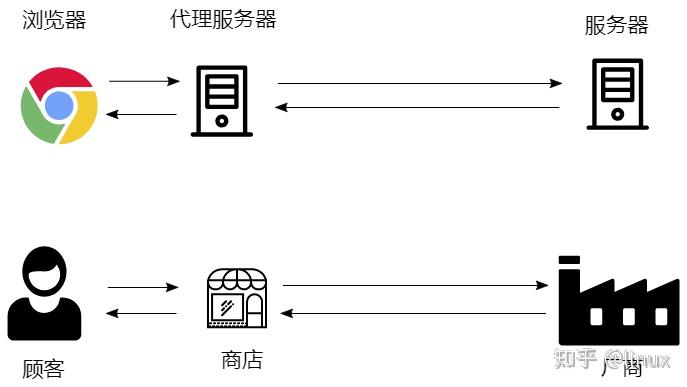


在实现上，该文件系统采用JuiceFS用于分布式存储，Ray用于分布式计算。

首先先介绍代理，

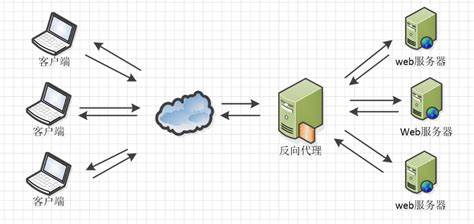
正向代理：

正向代理：就像我们科学上网的过程就是给自己配置了一个正向代理服务器。我们主动配置一个服务器，它把请求转到最终的网页服务器，再把后者的结果转发给我们，在该过程中，我们需要告诉浏览器我们要请求的代理服务器是什么。



反向代理：

先看一个例子   
例子：  
例如在租房过程中，如果我们通过房东来租房，有时候这个房东不一定是本人，可能是二房东，但是我们并不知道，还以为是真的房东，这种情况下我们就被反向代理了。  
在反向代理过程中，客户端并不知道目标服务器是谁的，就像是我们不知道房东到底是谁。   
于是我们可以利用反向代理实现mainserver的分布式，在客户端访问mainserver的过程中，我们表面上访问了同一个mainserver，但是可以通过反向代理来把请求转给不同的服务器进行响应，从而实现分布式。



**打标问题：**

# 打标就是从不同格式，如文字、图片、视频等文件中提取出其内容关键词，可以借助深度学习，这是以往小组的做法。

# Why

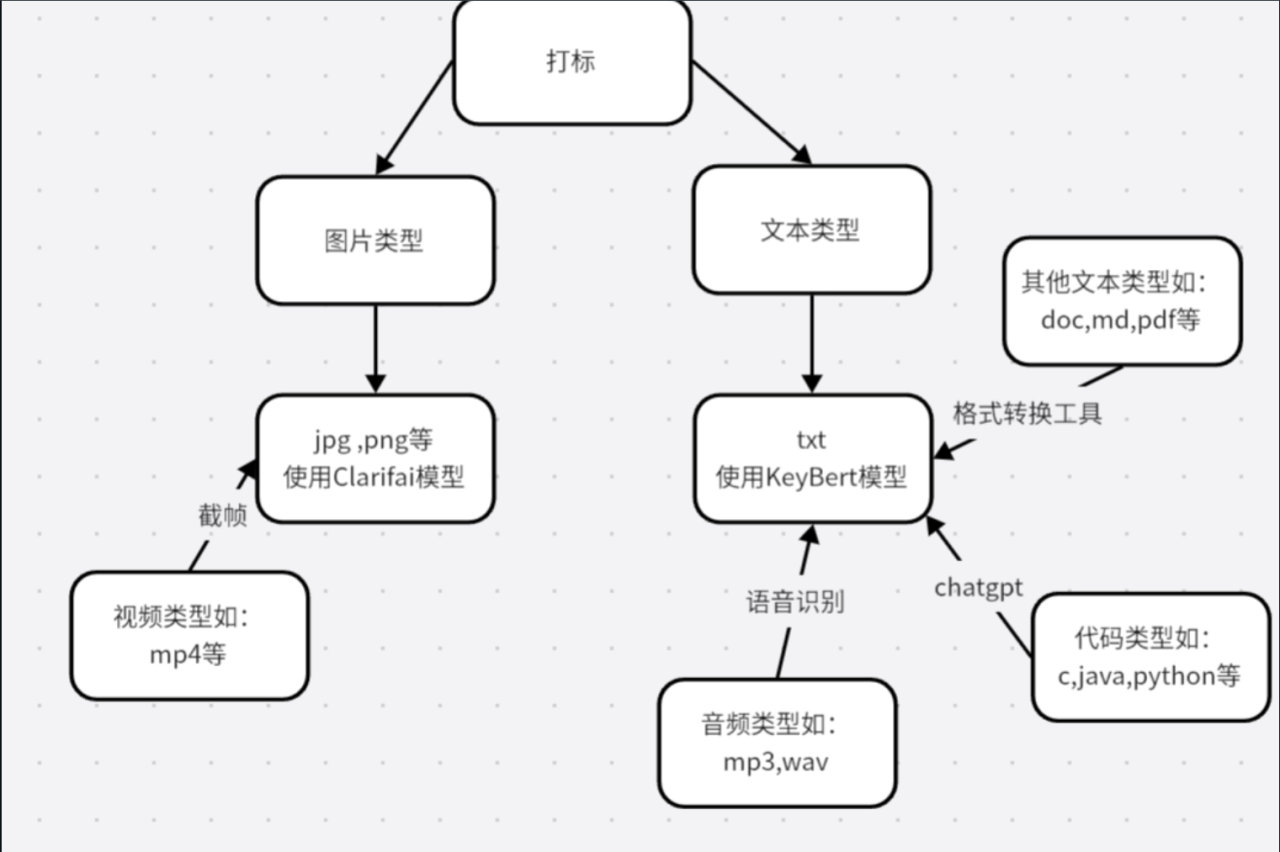
### 单一MainServer问题：

我们先看传统分布式文件系统，其通常采用集中式的架构，其中一个或多个主服务器负责管理文件的存储和访问。这种架构存在一些局限性，例如单点故障、性能瓶颈和可扩展性差等。。

我们小组提出使用Nginx作为反向代理，将主服务器内容部署到分布式集群中，从而实现主服务器的分布式部署。这种优化方案不仅可以提高系统的性能和可用性，还可以增强系统的安全性，减少单点故障的风险。

### 原有小组的打标框架问题:

在打标过程中，需要对文件类型进行分类，对于图片类型和文本类型需要用不同的模型框架来处理，还需要对不同文件类型进行格式转化，打标结构复杂，这在一定程度上加大了图文件系统的负担，可维护性变差。

所以我们想借助大模型拉简化打标程序，最好能建立统一索引，从而可以更好地支持不同格式的文件数据，尽量避免格式转换带来的数据丢失问题。

# How

总体的技术路线：

1. 复现往年项目my Glow并大致搞清楚结构
2. 使用nginx做反向代理以及负载均衡将主服务器转换为分布式
3. 尝试使用更好的打标工具对数据进行达标

### 复现往年项目

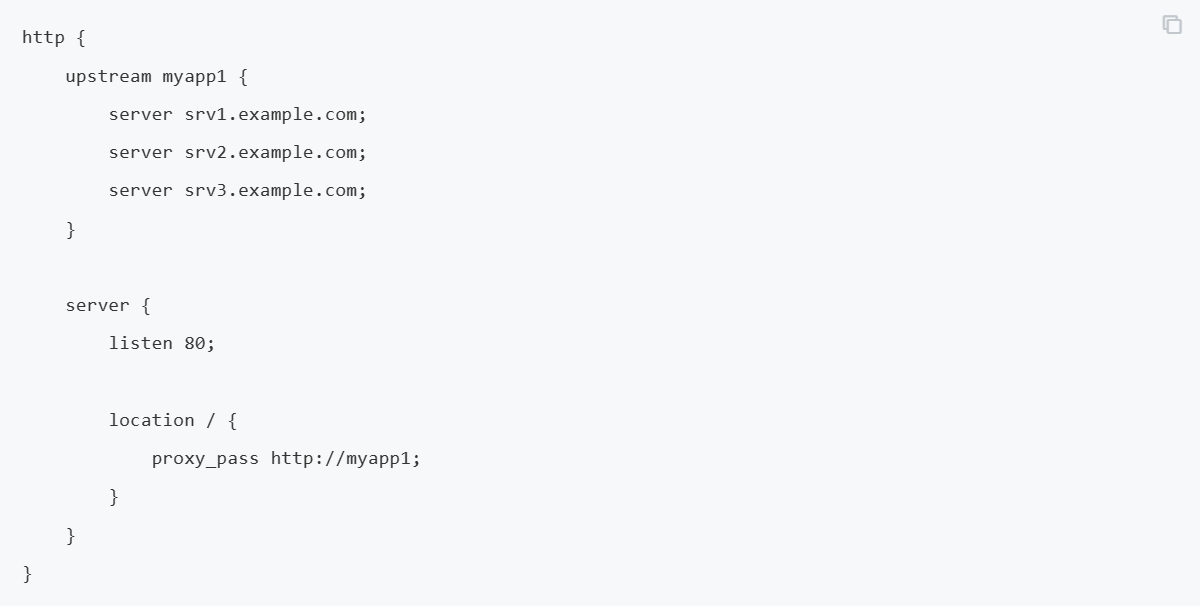
正在复现MyGlow小组的项目中，目前已进行了相关软件的下载和环境的部署，但还没有完全实现其功能。

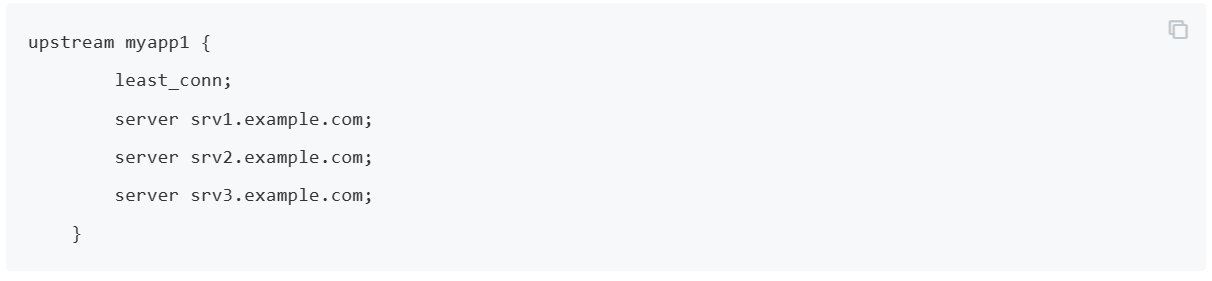
### Nginx

Nginx是一款轻量级的HTTP服务器，采用事件驱动的异步非阻塞处理方式框架，这让其具有极好的IO性能，时常用于服务端的反向代理和负载均衡。

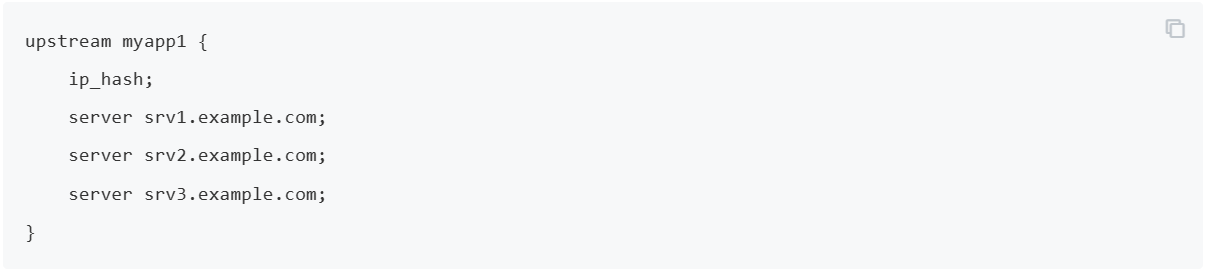
用Nginx做反向代理以及负载均衡的大致过程如下：



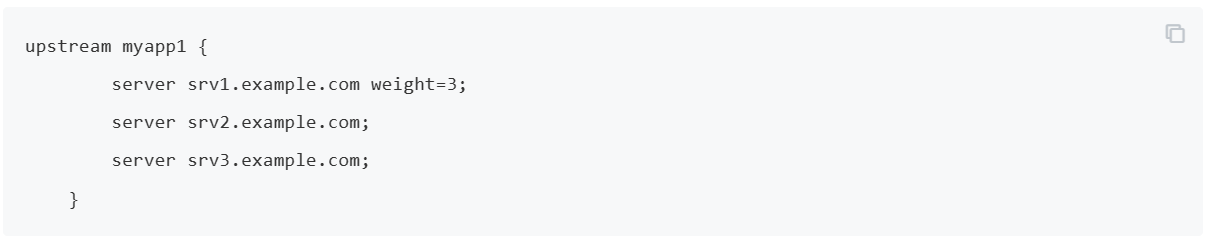
1. 首先介绍一个简单的Nginx配置，将server中的root替换成proxy\_pass，root说明是静态资源，可以由Nginx返回，而proxy\_pass说明是动态请求，需要进行转发，比如代理到Tomcat上。反向代理的过程就是request -> Nginx -> Tomcat，那么对于Tomcat而言，请求的IP地址就是Nginx的地址，而非真实的request地址。
2. 如何实现负载均衡。指定多台Tomcat来达到负载均衡，通过upstream来定义一组Tomcat，并指定负载策略：
3. 未特别配置负载平衡方法，则默认为循环。所有请求均为[proxied](https://www.docs4dev.com/docs/zh/nginx/current/reference/http-ngx_http_proxy_module.html" \l "proxy_pass)到服务器组 myapp1，并且 nginx 应用 HTTP 负载平衡来分发请求。
4. 另一个负载平衡原则是连接最少的。这种原则会将新的请求分配给不太繁忙的服务器。当least\_conn指令用作服务器组配置的一部分时，将激活 nginx 中的最少连接负载均衡：



1. 如果需要将客户端绑定到特定的应用程序服务器则可以使用 ip-hash 负载平衡机制，以客户端的 IP 地址用作哈希密钥，以确定应为客户端的请求选择服务器组中的哪个服务器。要配置 ip-hash 负载平衡，只需将ip\_hash指令添加到服务器(上游)组配置中：



1. 还可以通过使用服务器权重来进一步影响 nginx 负载平衡算法。这样就会按照权重来分配请求在下面的示例中，当为服务器指定weight参数时，权重将作为负载平衡决策的一部分。

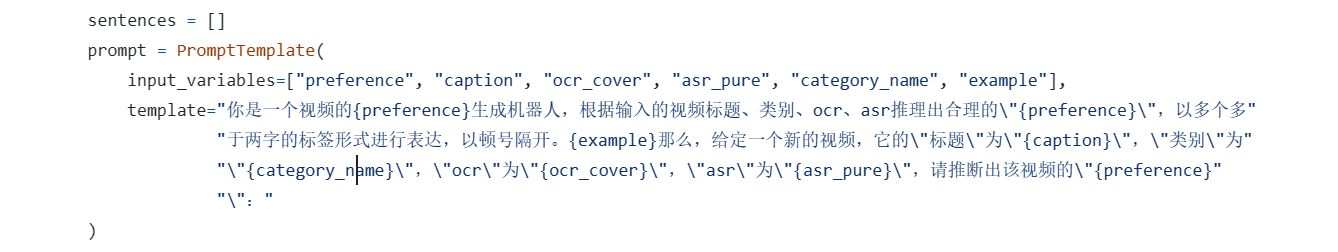


1. 具体使用哪一种负载策略还需要进一步确定
2. 再将proxy\_pass替换为upstream指定的值即可。

### 打标工具

TagGPT

首先是我们一直在探讨的TagGPT，其可以根据一个视频的标题，类别，ocr，asr生成一个template，就是一句类似这样的话



然后TagGPT会用这句话去询问chatgpt，得到一个回答，这个回答就是这个视频的兴趣标签。

根据论文，TagGPT是一种大型语言模型，具有零-shot多模式标记功能。它可以同时处理文本和图像信息，并能够在未经过任何训练的情况下为图像进行标记。所以在原本的计划当中，我们准备将文件系统的文件传给TagGPT，让TagGPT做打标的工作。

然而，实际运行时并好像并没有发现它有这样的能力，

所以我们不得不考虑更换模型，在老师的建议下我们查阅了一些llama index相关的信息

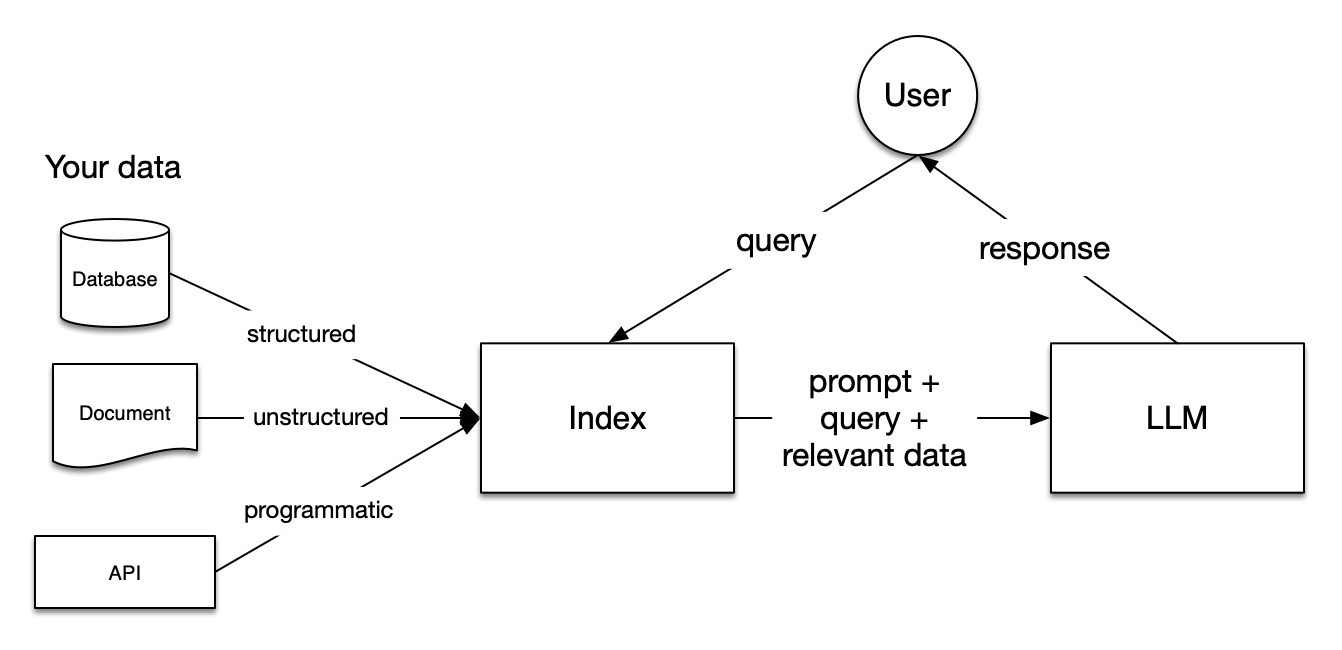
### Llama index

LlamaIndex 是一个将大语言模型（Large Language Models, LLMs，后简称大模型）和外部数据连接在一起的工具。

LlamaIndex的任务是通过查询、检索的方式挖掘外部数据的信息，并将其传递给大模型，因此其主要由3部分组成：

1. 数据连接
2. 索引构建
3. 查询接口

而我们要做的简单来说

1. 将文件传入Llama index
2. 由llama index解析文件为Node
3. 再从Node构建index，再存储文件至图数据库中。
4. 使用自然语言存储文件。比如调取某个文件，用langchain、anythingLLM等现成的工具如何与分布式文件系统更好地来交互

具体操作还需要进一步学习。

谢谢大家，我们今天的汇报结束了，接下来是提问及讨论环节，欢迎各位同学提出问题或者讨论想法