Visionary Art

人工智能绘画分享站：项目开发里程碑 3 汇报

2051857 曾诗容——项目主管 + 前端开发人员，
2052636 陈骁——ECS 云服务器技术顾问 + 全栈开发人员，
2050250 李其桐——技术支持 + 全栈开发人员，
2054080 林奕如——需求分析师 + 前端开发人员，
2053865 刘昱彤——产品经理 + 前端开发人员，
1751118 吴达鹏——运维 + 全栈开发人员，
2053868 于采篱——项目主管 + 前端开发人员

Computer Science and Technology Department, College of Electronic and Information Engineering(CEIE), Tongji University.
同济大学 电子与信息工程学院 计算机科学与技术系

2023 年 5 月 23 日

项目内容简介

项目开发背景

- 随着人工智能技术的飞速发展，AI 绘画技术也日趋成熟。近些日子来，无数精美的 AI 绘画作品都让我们眼前一新。对于没有接触过 AI 绘画领域的小白，他们可能也想体验 AI 绘画的奇妙，但不知从何下手。对于钻研 AI 绘画领域的技术人员，他们可能想分享自己的训练成果，同时和其他从业人员沟通交流，但是缺乏相关的平台。本软件的创建便是为了解决以上问题，为小白和技术人员提供一个在线生成图片，上传分享训练参数并和他人沟通的平台。

项目核心功能需求

1. 用户能够在线欣赏、生成图片，下载相关模型参数
2. 用户能够上传分享自己训练的模型参数
3. 用户能够使用软件系统提供的相关社交功能，与其他用户进行交流

项目发布测试反馈与 issue 提取

相关问题

本小组基于迭代 2 项目发布系统测试的反馈，结合答辩过程中与甲方沟通的相关情况，经过迭代 3 会议讨论，提取了以下问题：

- 在高并发情景下，绘图请求切换模型过多可能导致 Stable Diffusion 服务缓存池溢出，导致服务崩溃
- 模型查询、模型下载、模型上传等功能模块的后端接口由于数据库事务设计不当，存在性能瓶颈，导致用户体验不佳
- SD 服务生成图片存在合法性问题，部分图片中出现了不合理的色块、扭曲的图像结构和我国法律法规禁止在互联网平台上传播的敏感内容
- 前端交互逻辑中存在不合理点，例如搜索界面的搜索按钮应该支持通过键盘按键触发，而不是需要用户手动点击；某些按钮功能提示缺失等
- 配置的 ECS 服务器由于硬件条件限制存在性能瓶颈，在高并发情景下的请求响应与处理可能由于硬件瓶颈失败

项目发布测试反馈与 issue 提取

解决方案

本小组针对以上问题，在迭代 3 开发会议上进行讨论，最终提出了以下解决方案：

- 优化 SD 服务缓存池策略，基于 LRU 合理设计缓存换出机制，保证缓存命中率的情况下尽可能减少缓存池溢出的可能性
- 在服务端 DAO 操作和数据库事务之间，添加 Redis 中间件缓存访问机制，采用旁路缓存策略，将数据库访问压力转移到 Redis 上，从而在涉及到大表 join 查询或者高并发事务情景下减少数据库访问压力
- 由小组的相关前端开发人员，在单元测试和集成测试中，重新从用户角度出发对于人机交互逻辑进行省察，从而以用户友好为核心导向，进一步优化前端界面交互逻辑的相关功能需求实现
- 由小组的相关 SD 服务开发与维护人员，重新精读 High-Resolution Image Synthesis with Latent Diffusion Models 一文 [1]，分别从 prompt 文本预处理方法、网络模型结构和权重文件检测的三个方向，对于 <x>2image 操作的生成图片合法性进行检测和优化

本次里程碑任务

- 根据上述 issue 抽取工作中的相关内容项，对于项目开发中的技术债进行偿还，进一步提升系统的运行性能和可用性
- 进一步部署并且完善项目的自动化测试和 CI 流程，通过 Apifox 管理开发工作流，部署整套的项目自动化集成与测试解决方案，并且撰写测试报告与系统文档，为项目交付上线做好准备
- 按照小组开发章程，定时开展组会进行小组内部沟通，拉通对齐组员开发进度，稳步推进开发工作，保证在项目开发后期能够按照计划完成项目开发任务，交付能够让甲方客户满意的软件系统产品

工作项总览与相关图表

详见华为云

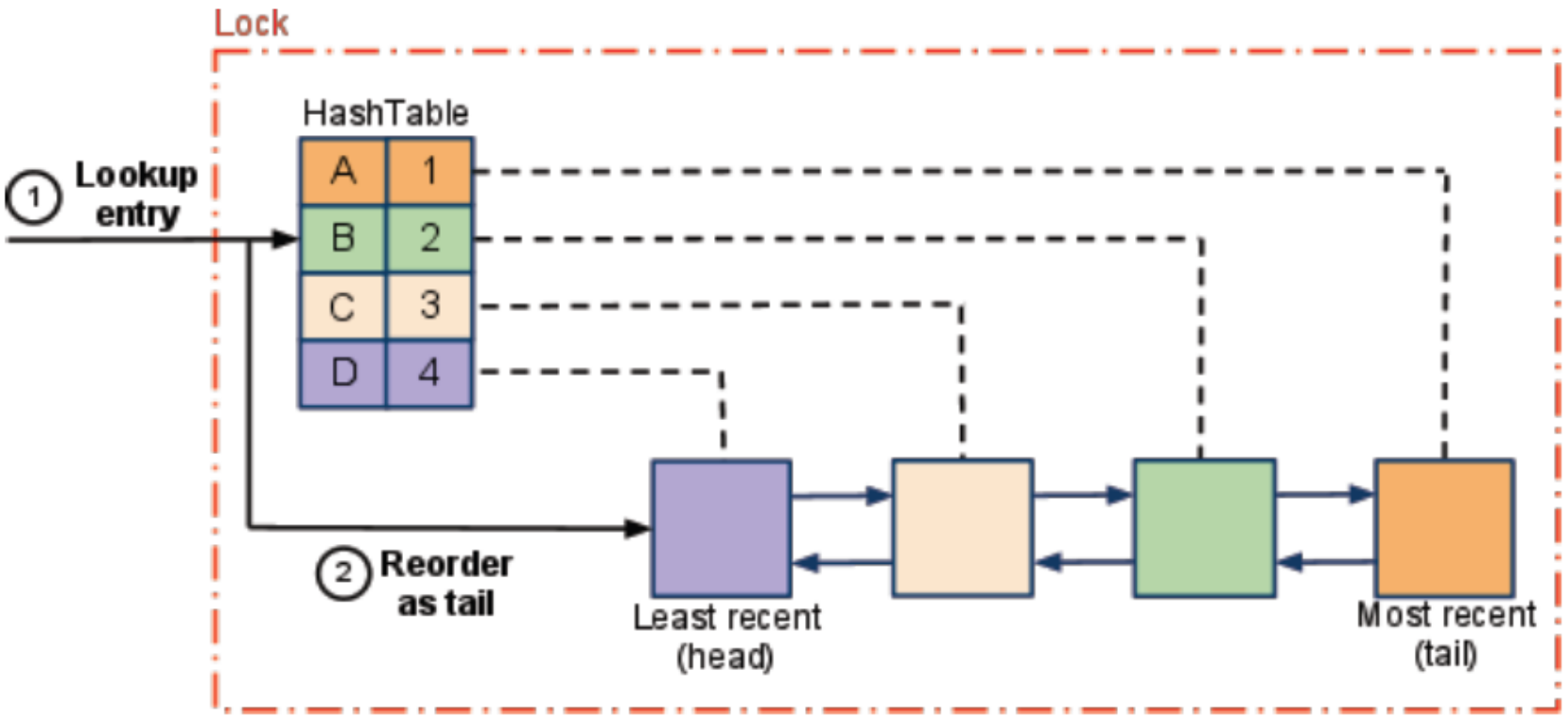
项目技术栈

- **主要开发语言**：Python
- **HTTP 服务器**：Tornado + FastAPI
- **前端技术选型**：Vue.js + Element UI/HTML + CSS + Js/Gradio
- **持久层框架**：SQLAlchemy
- **数据库服务**：MySQL + Redis
- **版本管理工具**：Git
- **远程代码托管平台**：华为云
- **接口管理与自动化测试工具**：Apifox + Mock.js

优化 SD 服务缓存池策略

- 在之前的单元测试和集成测试工作中，我们小组发现 SD 绘图的最大开销来源于从硬盘加载用户模型这一过程。因此在迭代三中，我们小组的 SD 服务维护人员重新设计了模型加载缓存池算法，采用 LRU 机制，在服务器缓存中维护若干之前用户的模型加载数据结构，并且在缓存池满且缓存未命中时，再从硬盘中将新模型加载到缓存，同时换出 LRU 队列中最近被加载频度最低的模型数据结构，从而有效提高了对于用户绘图请求的响应速度。在此基础上，我们小组的 SD 服务维护人员在经过反复的系统测试和实验后，最终将缓存池大小进行了合理设置，从而使得其既能够保证缓存池中的模型可以满足大部分的用户请求场景，同时也不会因为缓存池过大而导致内存占用过高，影响 SD 服务的运行性能。

优化 SD 服务缓存池策略



参考文献

[1] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].