（一）分布式文件系统项目

(可以借助大模型如 ChatGPT 完成作业)

1. 题目：

设计一个分布式文件系统。该文件系统可以是 client-server 架构，也可以是 P2P 非集中式架构。 要求文件系统具有基本的访问、打开、删除、缓存等功能，同时具有一致性、支持多用户特点。 在设计过程中能够体现在分布式课程中学习到的一些机制或者思想, 例如 Paxos 共识、缓存更新机制、访问控制机制、并行扩展等。 实现语言不限， 要求提交代码和实验报告，实验报告模板稍后在课程网站下载，提交时间为考试之后一周内。

2. 题目要求：

基本要求：

(1)、编程语言不限，选择自己熟悉的语言，但是推荐用 Python 或者 Java 语言实现；

(2)、文件系统中不同节点之间的通信方式采用 RPC 模式，可选择 Python 版本的 RPC、gRPC 等；

(3)、文件系统具备基本的文件操作模型包括：创建、删除、访问等功能；

(4)、作为文件系统的客户端要求具有缓存功能即文件信息首先在本地存储搜索，作为缓存的介质可以是内存也可以是磁盘文件；

(5)、为了保证数据的可用性和文件系统性能，数据需要创建多

个副本，且在正常情况下，多个副本不在同一物理机器，多个副本之间能够保持一致性（可选择最终一致性即延迟一致性也可以选择瞬时一致性即同时写）；

（6）、支持多用户即多个客户端，文件可以并行读写（即包含文件锁）；

（7）、对于上述基本功能，可以在本地测试，利用多个进程模拟不同的节点，需要有相应的测试命令或者测试用例，并有截屏或者 video 支持；

(8)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

（9）、实验报告长度不超过 20 页；

加分项：

（1）、加入其它高级功能如缓存更新算法；

（2）、Paxos 共识方法或者主副本选择算法等；

（3）、访问权限控制；

（4）、其他高级功能；

3. 参考实现:

(1). https://github.com/PinPinIre/CS4032-Distributed-File-System;
(2). https://github.com/topics/distributed-file-system;
(3). https://github.com/chrislusf/seaweedfs;
(4). https://github.com/vvanirudh/Distributed-File-System;
(5). https://github.com/mattdonnelly/CS4032-Distributed-File-System;
(6). https://github.com/Hasil-Sharma/distributed-file-system;
(7). https://github.com/mazumdarparijat/simple-distributed-file-system;
(8). http://www.scs.stanford.edu/06wi-cs240d/lab/project.html;
(9). https://github.com/Hasil-Sharma/distributed-file-system;

（二）分布式健值存储系统

1. 题目

设计并实现一个分布式键值（key-value）存储系统，可以是基于磁盘的存储系统，也可以是基于内存的存储系统，可以是主从结构的集中式分布式系统，也可以是 P2P 式的非集中式分布式系统。能够完成基本的读、写、删除等功能，支持缓存、多用户和数据一致性保证，提交时间为考试之后一周内。

2. 要求

1)、必须是分布式的键值存储系统，至少在两个节点或者两个进程中测试；

2)、可以是集中式的也可以是非集中式；

3)、能够完成基本的操作如：PUT、GET、DEL 等；

4)、支持多用户同时操作；

5)、至少实现一种面向客户的一致性如单调写；

6)、需要完整的功能测试用例；

7)、涉及到节点通信时须采用 RPC 机制；

8)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

加分项：

1)、具备性能优化措施如 cache 等；

2)、具备失效容错方法如：Paxos、Raft 等；

3)、具备安全防护功能；

4)、其他高级功能；

3. **参考实现**

1)、http://anishjain89.github.io/15418/；

2)、https://accumulo.apache.org/；

3)、https://www.mongodb.com/

4)、https://github.com/yuantiku/YTKKeyValueStore

5)、https://github.com/boltdb/bolt

6)、https://github.com/dgraph-io/badger

7)、https://github.com/google/leveldb

8)、https://github.com/apple/foundationdb

9).https://github.com/etcd-io/etcd/tree/1f8764be3b43448ccfd60706c42dab09b0bc6ed3

（三）非集中式的 DNS 系统

1. 题目

传统的 DNS 系统大都是集中式的，在性能和安全性等方面存在一定的缺陷，因此本项目设计一个集中式的 DNS 系统。该 DNS 系统分布式在互联网中的多个节点上，客户端能够通过该 DNS 系统进行域名查询、增加和删除等操作。

2. 要求

1)、实现的 DNS 是非集中式系统；

2)、采用 DHT 作为数据存储；

3)、能够完成基本的增删查改的操作；

4)、具有缓存功能；

5)、在至少 2 个节点或者进程上测试；

6)、需进行性能测试；

7)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

加分项：

1)、具备安全加密特性；

2)、支持多用户；

3. 参考实现

1)、https://github.com/Mononofu/P2P-DNS

2)、https://github.com/HarryR/ffff-dnsp2p

3)、https://github.com/torrentkino/torrentkino

4)、https://github.com/mwarning/KadNode
5)、https://github.com/BrendanBenshoof/P2PDNS
6)、https://github.com/samuelmaddock/swarm-peer-server
7)、https://github.com/BradNeuberg/p2psockets

（四）共享文档编辑系统

1、 题目

设计并实现可同时支持多人进行文档编辑的系统。允许每个人进行读写操作，并能够保障系统的一致性，此外还应具备一定的容错能力。

2、 要求

1)、支持多人同时在线编辑文档；

2)、通信方式选择 RPC；

3)、具备分布式系统互斥协议；

4)、支持至少一种系统一致性；

5)、具备一定的容错能力；

6)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、 参考实现

1)、https://github.com/star7th/showdoc

2)、https://github.com/Kinto/kinto

（五）去中心化的聊天系统

1、 题目

传统的聊天系统如微信等是一种中心化系统设计，数据集中存放。本项目的是设计一种去中心化的聊天系统，将聊天数据分散存储在各个客户端上。

2、 要求

1)、支持一对一聊天；

2)、支持聊天室群聊；

3)、能够满足实时性要求如响应时间控制在 10ms 以内；

4)、通信方式采用 RPC；

5)、支持分布式系统一致性；

6)、具备一定的失效容错措施；

7)、需进行性能测试；

8)、聊天数据分散存储；

9)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、 参考实现

1)、https://github.com/mgax/zechat
2)、https://github.com/RocHack/meshchat
3)、https://github.com/web3infra/dchat
4)、https://github.com/wgaylord/DecentralizedPythonChat
5)、https://github.com/ninthcrow/distributed-chat

6)、https://github.com/AlanWilms/Decentralized-Chat
7)、https://github.com/PortalNetwork/dchat

（六）基于 MapReduce 的软件 Bug 分类

1、 题目

在 Github 代码仓库中，存在大量已分类（即加上标签）的软件 bug。但是，现在的分类标签大都是基于人工添加的，效率比较低。本项目通过爬取大量具有分类标签的 Bug，利用 MapReduce 分布式编程模型，实现分类算法，自动给 Bug 加上标签。

2、 要求

1)、爬取至少 1000 个具有分类标签的 bug;

2)、采用 MapReduce 实现分类算法;

3)、测试验证算法的准确度;

4)、分析结果并得出结论;

5)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、 参考实现

参考 Hadoop 或者 Spark 中的相关算法案例

(七) 分布式强化学习系统

1、 题目

强化学习作为一种典型的无模型在线学习方法在各个领域都有应用，比如人机对弈系统等。但是对于大规模复杂的问题场景，单节点的强化学习已经无法满足性能和模型容量的要求，需要一种分布式强化学习系统。请根据所学到的分布式系统的相关知识如分布式架构、分布式通信、分布式深度学习等实现分布式强化学习系统，并通过简单场景，如图像识别、文本识别等验证算法的有效性。

2、 主要要求

1)、编程语言不限，但是推荐 Python;

2)、 至少包含一种经典的强化学习方法如 Q-Learning、DQN 等;

3)、 节点之间通信采用 RPC 或者既有的通信模型如 MPI 通信接口;

4)、 具有容错措施如冗余、多个 Master 等;

5)、 可以借用深度学习中参数服务的实现形式以及源码;

6)、结合具体场景给出效果和性能测试结果;

7)、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、 参考实现

参考开源系统 Ray (https://github.com/ray-project/ray)、TensorFlow、Pytorch 等;

（八） 分布式共享内存系统

1、题目

创建一个分布式共享内存（DSM）系统，以便在不同机器上运行的进程可以共享一个地址空间。您需要一个允许缓存但保持一致性的计划。 还需要找到至少一个可以充分利用 DSM 的样例程序，以帮助评估所实现的系统。

2、主要要求

1、编程语言不限，推荐使用 C 语言;

2、节点之间的远程通信要求使用 RPC;

3、包含至少一种缓存替换算法;

4、包含至少一种缓存一致性保障方法;

5、有一定的容错能力如数据丢失或者损坏可以恢复;

6、利用至少 1 个样例程度对实现的 DSM 进行性能和可扩展性（节点数量增多）的评估;

7、可以利用一台机器上的多个进程的形式模拟分布式系统;

8、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、参考实现

可以参考 Github 上的一些 DSM 的开源实现

（九） 分布式爬虫系统

1、题目

网络爬虫是 90 年代的话题。但是，一个高效且可扩展的版本是一个复杂的分布式系统，包含许多有趣的部分。实现分布在多个数据中心的工作爬虫、以分布式方式维护 web 视图、分布式页面排名计算和关键字搜索功能。

2、主要要求

1、编程语言不限；

2、包含分布式爬取网页能力；

3、能够进行分布式页面排名；

4、实现分布式关键字搜索；

5、节点通信方式采用 RPC；

6、提交源码和报告，压缩后命名方式为：学号_姓名_班级

3、参考

请参考：https://www.cs.ubc.ca/~bestchai/teaching/cs416_2018w1/project2/index.html

（十） 分布式系统概念可视化系统

1、题目

分布式系统中涉及纷繁复杂的概念和原理，有些概念和原理比较抽象难理解，对学生的能力要求比较高，难以完全掌握。为了帮助学生理解，实现可以直观可视化的例子或者过程演示阐述概念和原理，辅助教学。

2、主要要求

包含分布式系统中的关键概念和原理如逻辑时钟、向量时钟、两阶段提交协议、Paxos/RAFT/PBFT 共识协议、至少一种一致性协议以及至少一种容错方法，有可视化界面，鼓励提供一些与使用者的交互能力。

3、参考

无

11 大模型应用

1、 题目

大模型驱动的多智能体协作系统

2、 主要要求

选择可以使用大模型智能体的场景比如学校、医院、工厂等；

形成至少由 4 个以上角色构成的多智能体系统；

智能体之间通过利用 RPC 进行通信；

不需要人为介入，系统自主运行；

能够运行并演示系统；

3、 参考实现

— https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/；

— https://github.com/THUDM/AgentBench；

— https://microsoft.github.io/autogen/；

— https://aitutor.liduos.com/07-agents/07-2.html

— https://lucas-soares.medium.com/building-a-simple-llm-powered-github-agent-with-langchain-f255f156c0ba；

— https://github.com/langroid/langroid；

— https://github.com/OpenBMB/AgentVerse

— https://medium.com/@AIWorldBlog/conversant-building-advanced-llm-apps-with-multi-agent-interactions-aac72a41bd35;

— https://arxiv.org/pdf/2309.07870.pdf%EF%BC%8C;

— https://promptengineering.org/what-are-large-language-model-llm-agents/

— https://lilianweng.github.io/posts/2023-06-23-agent/

— https://github.com/WooooDyy/LLM-Agent-Paper-List

— https://github.com/geekan/M etaGPT

—