**分布式文件系统项目**

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。

1. **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>;

**分布式的健值存储系统**

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、些、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。

1. **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；

1. **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

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。

1. 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；**

1. 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>

8）、<https://github.com/mwarning/masala>

**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。

1. **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

1. **参考实现**

1）、<https://github.com/star7th/showdoc>

2）、<https://github.com/Kinto/kinto>

**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。

1. **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

1. **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>

**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。

1. **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

1. **参考实现**

无