书面报告

学号：21214913　　　　姓名：林国梁

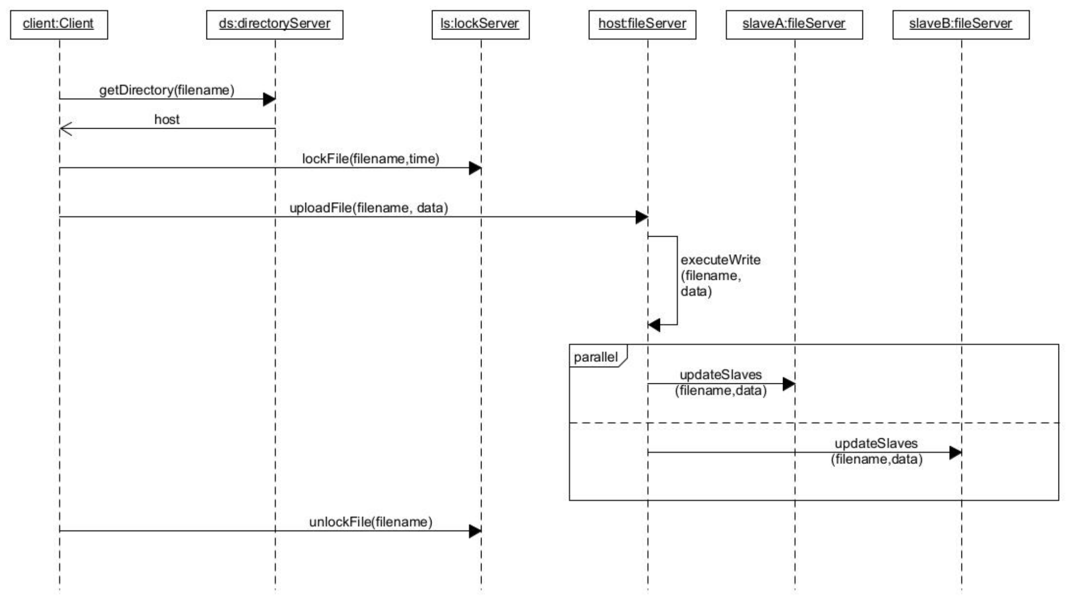
本次书面报告分析的代码来自于https://github.com/PinPinIre/CS4032-Distributed-File-System。

一、系统总览

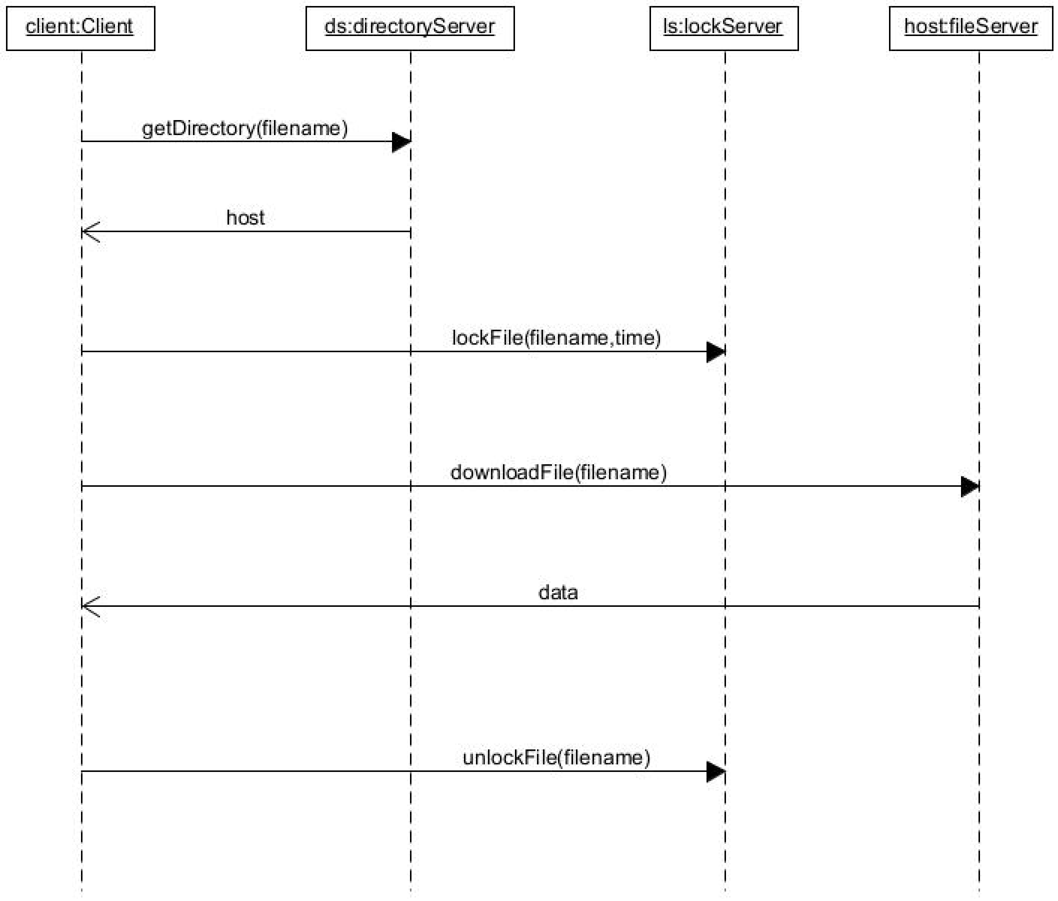
本系统可分为四个组件：客户端、目录服务器、文件服务器、锁服务器。它们的主要功能如下表所示：

组件	主要功能
客户端	发送文件的上传和下载请求
目录服务器	提供文件服务器的地址
文件服务器	实现文件的同步更新和下载
锁服务器	实现文件的上/解锁

在上传文件时的时序图如下图所示：

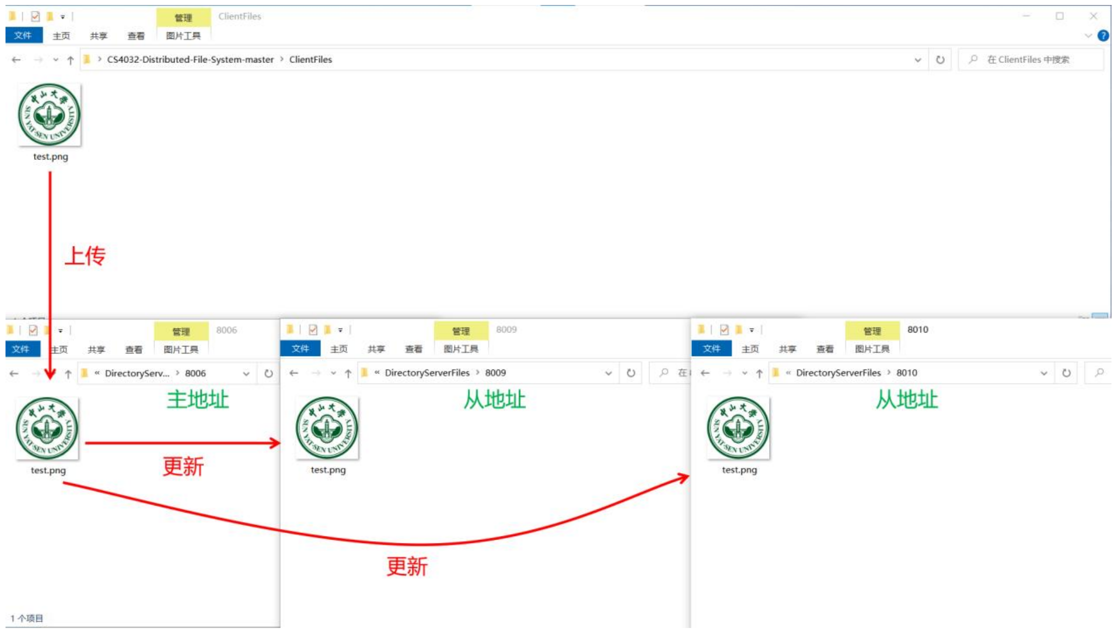


在下载文件时的时序图如下图所示：

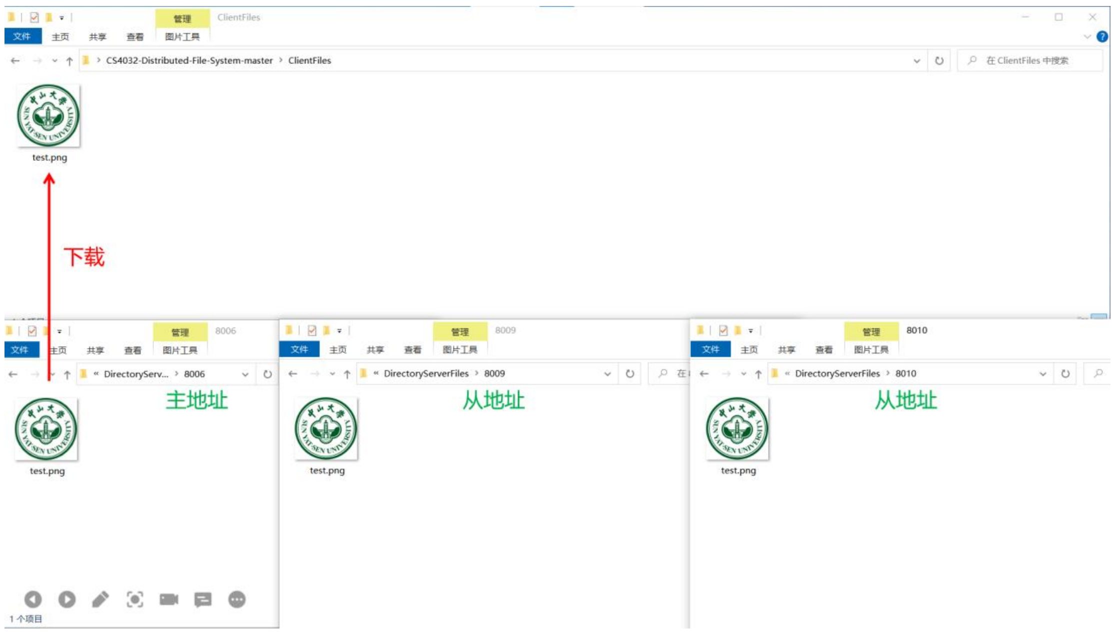


二、运行实例

在启动服务器后，在客户端输入”upload test.png”命令后，文件会被上传主文件服务器中，然后主文件服务器会更新各从文件服务器，即如下图所示：


ClientFiles
CS4032-Distributed-File-System-master › ClientFiles
test.png
上传
DirectoryServ... › 8006
主地址
更新
DirectoryServerFiles › 8009
从地址
DirectoryServerFiles › 8010
从地址
更新
test.png
test.png
test.png

在客户端输入”download test.png”命令后，文件会从主文件服务器下载到客户端目录，即如下图所示：


ClientFiles
CS4032-Distributed-File-System-master › ClientFiles
test.png
下载
DirectoryServ... › 8006
主地址
DirectoryServerFiles › 8009
从地址
DirectoryServerFiles › 8010
从地址
1 个项目

三、源码分析

1.客户端（client.py）

1.1.文件上传

文件的上传含三个步骤：open、write、close：

```
        elif re.match(UPLOAD_REGEX, user_input.lower()):
            request = user_input.lower()
            file_name = request.split()[1]
            con.open(file_name, if_upload= True)
            con.write(file_name, data)
            con.close(file_name)
```

在 open 步骤中，首先从目录服务器获得文件服务器主地址，然后对该文件上锁：

```
    def open(self, filename, if_upload = False):
        """Function opens a file by downloading from a remote server"""
        file_downloaded = False
        if filename not in self.open_files.keys():
            # Get the info of the server hosting the file
            request = self.__get_directory(filename)
            if re.match(self.SERVER_RESPONSE, request):
                params = request.splitlines()
                server = params[0].split()[1]
                port = int(params[1].split()[1])
                open_file = params[2].split()[1]
                # Get lock on file before downloading
                self.__lock_file(filename, 10)
                if not if_upload:
                    file_downloaded = self.__download_file(server, port, open_file)
                if file_downloaded or if_upload:
                        self.open_files[filename] = open_file
        return file_downloaded
```

在 write 步骤中，将数据写到客户端本地文件：

```
    def write(self, filename, data):
        """Function that writes to an open file"""
        success = False
        if filename in self.open_files.keys():
            local_name = self.open_files[filename]
            path = os.path.join(self.BUCKET_LOCATION, local_name)
            file_handle = open(path, "wb+")
            file_handle.write(data)
            success = True
        return success
```

在 close 步骤中，文件会被上传到主文件服务器。在文件上传完成后，对文件解锁。

```
    def close(self, filename):
        """Function closes a file by uploading it"""
        file_uploaded = False
        if filename in self.open_files.keys():
            request = self.__get_directory(filename)
            if re.match(self.SERVER_RESPONSE, request):
                params = request.splitlines()
                server = params[0].split()[1]
                port = int(params[1].split()[1])
                open_file = params[2].split()[1]
                # Upload the file and then delete from local host
                file_uploaded = self.__upload_file(server, port, open_file)
                # Remove lock from file
                self.__unlock_file(filename)
                if file_uploaded:
                    del self.open_files[filename]
        return file_uploaded
```

1.2.文件下载

文件的下载含两个步骤：open、close：

```
        elif re.match(DOWNLOAD_REGEX, user_input.lower()):
            request = user_input.lower()
            file_name = request.split()[1]
            con.open(file_name)
            con.close(file_name)
```

在 open 步骤中，首先从目录服务器获得文件服务器主地址，然后对该文件上锁。此处与上传文件操作不同的是，在 open 步骤中会下载文件到本地。

```
    def open(self, filename, if_upload = False):
        """Function opens a file by downloading from a remote server"""
```

```python
        file_downloaded = False
        if filename not in self.open_files.keys():
            # Get the info of the server hosting the file
            request = self.__get_directory(filename)
            if re.match(self.SERVER_RESPONSE, request):
                params = request.splitlines()
                server = params[0].split()[1]
                port = int(params[1].split()[1])
                open_file = params[2].split()[1]
                # Get lock on file before downloading
                self.__lock_file(filename, 10)
                if not if_upload:
                    file_downloaded = self.__download_file(server, port, open_file)
                if file_downloaded or if_upload:
                    self.open_files[filename] = open_file
        return file_downloaded
```

下载文件的 close 步骤与上传文件的 close 步骤相同。

2.目录服务器（directoryServer.py）

对于一般的文件目录查询请求，目录服务器会调用 get_server 函数来返回主服务器地址及从服务器地址：

```python
    def handler(self, message, con, addr):
        if re.match(self.GET_REGEX, message):
            self.get_server(con, addr, message)
```

get_server 函数首先向数据库的表“Directories”查询文件是否有主地址，若有则返回该地址；否则，随机选取一个文件服务器的地址作为主地址并返回该地址。一同返回的还有从文件服务器的地址。

```python
    def get_server(self, con, addr, text):
        # Handler for file upload requests
        request = text.splitlines()
        full_path = request[1].split()[1]
        path, file = os.path.split(full_path)
        name, ext = os.path.splitext(file)
        #filename = hashlib.sha256(full_path).hexdigest() + ext
        filename = full_path
        host, port = self.find_host(path)
        if not host:
            # The Directory doesn't exist and must be added to the db
            server_id = self.pick_random_host()
            self.create_dir(path, server_id)
            host, port = self.find_host(path)
        # Get the list of slaves that have a copy of the file
        slave_string = self.get_slave_string(host, port)
        return_string = self.GET_RESPONSE % (host, port, filename, slave_string)
        print return_string
        con.sendall(return_string)
        return
```

3.锁服务器（lockServer.py）

3.1.文件上锁

对于文件的上锁请求，锁服务器会调用 get_lock 函数来对文件进行上锁：

```
        if re.match(self.LOCK_REGEX, message):
            self.get_lock(con, addr, message)
```

get_lock 函数则会调用 lock_file 函数来对文件上锁：

```
    def get_lock(self, con, addr, text):
        # Handler for file locking requests
        request = text.splitlines()
        full_path = request[0].split()[1]
        duration = int(request[1].split()[1])
        lock_time = self.lock_file(full_path, duration)
        if lock_time:
            return_string = self.LOCK_RESPONSE % (full_path, lock_time)
        else:
            return_string = self.FAIL_RESPONSE % (0, str(duration))
        con.sendall(return_string)
        return
```

lock_file 函数首先向数据库的表“Locks”查询对应文件是否有结束时间比当前时间更大的锁，若无则将文件上锁，即设定“结束时间←当前时间+锁持续时间”；否则，返回上锁失败的信息。

```
    def lock_file(self, path, lock_period):
        # Function that attempts to lock a file
        return_time = -1
        con = db.connect(self.DATABASE)
        # Exclusive r/w access to the db
        con.isolation_level = 'EXCLUSIVE'
        con.execute('BEGIN EXCLUSIVE')
        current_time = int(time.time())
        end_time = current_time + lock_period

        cur = con.cursor()
        cur.execute("SELECT count(*) FROM Locks WHERE Path = ? AND
Time > ?", (path, current_time))
        count = cur.fetchone()[0]
        if count is 0:
            cur.execute("INSERT INTO Locks (Path, Time) VALUES (?, ?)", (path,
end_time))
            return_time = end_time
        else:
            return_time = False
        # End Exclusive access to the db
        con.commit()
        con.close()
        return return_time
```

3.2.文件解锁

对于文件的解锁请求，锁服务器会调用 get_unlock 函数来对文件进行解锁：

```
        elif re.match(self.UNLOCK_REGEX, message):
            self.get_unlock(con, addr, message)
```

get_unlock 函数则会调用 unlock_file 函数来对文件上锁：

```
    def get_unlock(self, con, addr, text):
        # Handler for file unlocking requests
        request = text.splitlines()
        full_path = request[0].split()[1]
        lock_time = self.unlock_file(full_path)
        return_string = self.LOCK_RESPONSE % (full_path, lock_time)
        con.sendall(return_string)
        return
```

unlock_file 函数首先向数据库的表“Locks”查询对应文件是否有结束时间比当前时间更大的锁，若有，则将更大的锁的结束时间设定为当前时间。

```
    def unlock_file(self, path):
        # Function that attempts to unlock a file
        return_time = -1
        con = db.connect(self.DATABASE)
        # Exclusive r/w access to the db
        con.isolation_level = 'EXCLUSIVE'
        con.execute('BEGIN EXCLUSIVE')
        current_time = int(time.time())
        cur = con.cursor()
        cur.execute("SELECT count(*) FROM Locks WHERE Path = ? AND
Time > ?", (path, current_time))
        count = cur.fetchone()[0]
        if count is not 0:
            cur.execute("UPDATE Locks SET Time=? WHERE Path = ? AND
Time > ?", (current_time, path, current_time))
        # End Exclusive access to the db
        con.commit()
        con.close()
        return current_time
```

4.文件服务器（fileServer.py）

4.1.处理文件上传请求

对于文件上传请求，文件服务器调用 upload 函数进行处理：

```
        if re.match(self.UPLOAD_REGEX, message):
            self.upload(con, addr, message)
```

upload 函数首先在服务器本地进行写操作，再通知从服务器进行更新：

```
    def upload(self, con, addr, text):
        # Handler for file upload requests
        filename, data = self.execute_write(text)
        return_string = self.UPLOAD_RESPONSE
        con.sendall(return_string)
        self.update_slaves(filename, data)
        return
```

4.2.处理文件下载请求

对于文件下载请求，文件服务器调用 download 函数进行处理：

```
        elif re.match(self.DOWNLOAD_REGEX, message):
            self.download(con, addr, message)
```

download 函数以二进制方式打开文件，并使用 base64 将二进制数据转化为文本，然后将文本传输到客户端。

```
    def download(self, con, addr, text):
        # Handler for file download requests
        request = text.splitlines()
        filename = request[0].split()[1]
        path = os.path.join(self.BUCKET_LOCATION, filename)
        file_handle = open(path, "rb")
        data = file_handle.read()
        return_string = self.DOWNLOAD_RESPONSE % (base64.b64encode(data))
        con.sendall(return_string)
        return
```

四、代码勘误

代码主要 BUG 如下：

文件	fileServer.py	fileServer.py	lockServer.py	client.py
函数	execute_write	download	unlock_file	__download_file
修改前	`file_handle = open(path, "w+")`	`file_handle = open(path, "w+")`	`if count is 0:`	`data = request_data.split()[0]`
修改后	`file_handle = open(path, "wb+")`	`file_handle = open(path, "rb")`	`if count is not 0:`	`data = request_data.split()[1]`
备注	若使用”w+”打开文件，则会以文本形	若使用”w+”打开文件，则文件内容会	此处应该是将那些未到结束时间的	数据在第 1 个子字符串而非在第 0 个子字符串

	式进行写操作	被清空	锁全部解锁	

其他修改详见 https://github.com/Stories-z/simpleDFS。