Test 3

Implement a storage array simulator as a web service or application. It provides SOAP or REST API to users. The following functions are supported:

1. Create one or multiple LUN’s.

2. Resize a LUN.

3. Remove a LUN.

4. Retrieve the information (size) of a LUN.

5. Concurrent requests.

6. Data persistence.

1. 需求分析
2. Array模拟器

模拟阵列，首先考虑阵列的特性和可支持的操作。阵列有自身大小、剩余可用空间、阵列上已创建的LUN信息等；其次阵列常见操作有为LUN分配空间，在LUN删除后释放存储空间。

1. 阵列的属性：
2. 阵列大小：整数，阵列包含基本存储单元个数
3. 可以空间：剩余可用存储单元个数
4. 阵列上已创建的LUN信息
5. 阵列上存储空间占用情况：以可用存储空间和阵列地址记录
6. 阵列支持的操作
7. 给LUN分配存储空间
8. LUN删除后释放存储空间
9. 对外提供的web service分析
10. 创建1个或多个LUN

根据请求的LUN的数目和大小,判读存储空间是否足够,如果足够,则创建LUN,返回LUN地址；并更新array上的LUN信息；如果存储空间不够,返回”存储空间不够,创建失败提示!”.

1. 修改LUN的大小

根据LUN的ID，找到该LUN对应的实际存储空间的地址，然后先释放当前分配空间，然后重新分配对应大小的存储空间，并更新阵列上LUN信息，如果存储空间不够， 返回”存储空间不够,修改大小失败提示!”.

1. 删除LUN

根据LUN的ID，找到该LUN的实际存储空间并释放，然后更新阵列上LUN的信息

1. 返回LUN的大小

根据LUN的ID，从阵列存储的LUN信息中查看LUN的大小信息

1. 并发请求

对api的并发访问，由web开发框架支持，当前主要针对同一个LUN可能同时存在多个的操作的情况;设置LUN的flag属性记录状态信息,每个操作进行时,都需要对此状态变量进行锁定,操作完成后,解除锁定.当同时发生两个请求时,只有一个请求能获得马上响应,另一个请求需要等待.

1. 数据持久化

将阵列上LUN列表以及LUN相关信息（每个LUN的起始地址和长度）写入文件中。

1. 详细设计

根据需求分析，在该service中涉及两个对象，array和LUN。Array在实现时用一个list来模拟，list的长度就是array的大小

Class array():

Def \_\_init\_\_()

Self.array\_size =size #阵列大小，基本存储单元数量，int

Self.array = [0,0,0,0,0,0,] #list模拟array

Self.array\_available\_size = #阵列可用空间大小, 基本存储单元数量， int

Self.array\_available\_space #可用存储块{space\_lenth: start\_index}

Self.array\_lun\_info #阵列上已创建LUN的起始地址和长度{lunID: (start\_index, lun\_size)}

Self.array\_lun #阵列上创建的LUN实例的列表，{lunID: lun\_instance}

Def assign\_space(size) #按需要的size大小分配空间

Def free\_space(lunID) #释放给定LUN占用的存储空间

其中assign\_space函数实现的基本思路是：Self.array\_available\_space中记录了可用的空间块和起始index，每次分配空间时，先将可用空间块按大小排序，遍历查找可满足需求空间的最小可用空间块进行分配， 暂不考虑所需空间无法找到整块连续空间分配的情况（此情况按空间不足返回）。

Free\_space()函数实现释放指定LUN空间的功能，由lunID从Self.array\_lun\_info中找到该LUN占用的存储空间块，包括空间大小和起始index，然和在Self.array\_available\_space中查找是否有空间和即将释放的空间相连，若相连修改Self.array\_available\_space中相连元素的长度和起始index，若查找完毕不相连，则向Self.array\_available\_space中添加该可用存储块。

Class lun()

Def \_\_init\_\_()

Self.id #LUN的标识，毫秒时间戳+4位随机数组成字符串

Self.size #LUN的大小

Self.array #LUN所在的阵列

Self.flag #标识LUN当前是否可用，‘0’可用，‘1’不可用，防止对同一个LUN进行多个操作。

LUN支持的操作如下：

Assign\_lun\_space: 调用阵列的assign\_space方法，为lun分配空间；

Resize\_lun：调整LUN的大小，调用阵列free\_space方法，先释放当前空间，然后再按需求空间assign\_space

Get\_size：获取LUN的size信息

Set\_flag: 修改LUN的状态，使其可用或者不可用

Get\_flag：查看LUN的状态

Free\_lun\_space: 释放LUN占用的空间

根据需求，需要对外提供5个接口，包括如下功能：

1. Create\_lun(size, number=1), 可一次创建一个或者多个相同大小的LUN，创建一个LUN就是实例化一个lun，并为lun分配size大小的空间，更新array\_lun，返回创建好的LUN ID和size，格式{lunID1:size, lunID2:size2}
2. Resize\_lun(lunID): 修改lun的大小，根据lunID从array\_lun中找到该lun实例，修改前查看该lun当前可用，调用lun的resize方法完成修改，然后设置lun的flag
3. Remove\_lun(lunID): 删除lun，根据lunID从array\_lun中找到该lun实例，lun的flag为可用状态下，释放该lun的空间，然后更新array的array\_lun信息。
4. retrieve\_lun\_size(lunID): 查看lun的大小，由lunID找到lun实例，调用lun的get\_size方法。
5. persistence(): 将array上LUN的信息写入文件保存到磁盘

使用bottlepy 框架提供上述5个接口

1. 接口使用说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口 | 访问方法 | 参数 | 返回值 | 返回码说明 |
| /create\_lun | POST | Size:必选；number可选,默认值1 | JSON格式，创建成功返回[{lunID:size},{ret\_code:msg}] | 0:操作成功  -1:参数错误  -2：阵列空间不足  -4：分配空间失败 |
| /resize\_lun | POST | lunID:必选 | JSON格式，[{lunID:size},{ret\_code:msg}] | 0：操作成功  -1：参数错误，阵列上不存在该lunID  -3：lun不可用  -5：修改阵列大小失败 |
| /remove\_lun | POST | lunID:必选 | {ret\_code:msg} | 0：操作成功  -1：参数错误，阵列上不存在该lunID  -3：lun不可用  -5：删除失败 |
| /retrieve\_lun\_size | GET | lunID:必选 | [{size:ret\_size},{ret\_code:msg}] | 0: 操作成功  -1：参数错误 |
| /persistece | GET |  | {ret\_code:msg} | 0: 操作成功  -1：操作失败 |