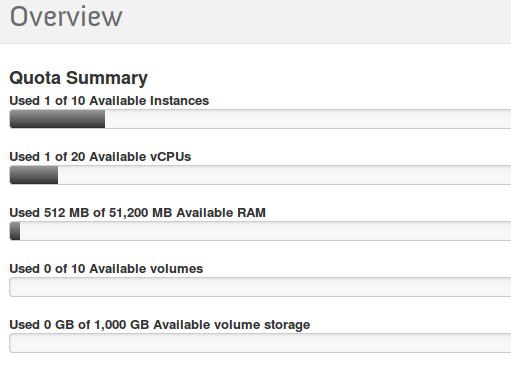
正文：

一、什么是**配额**(quota)**管理**

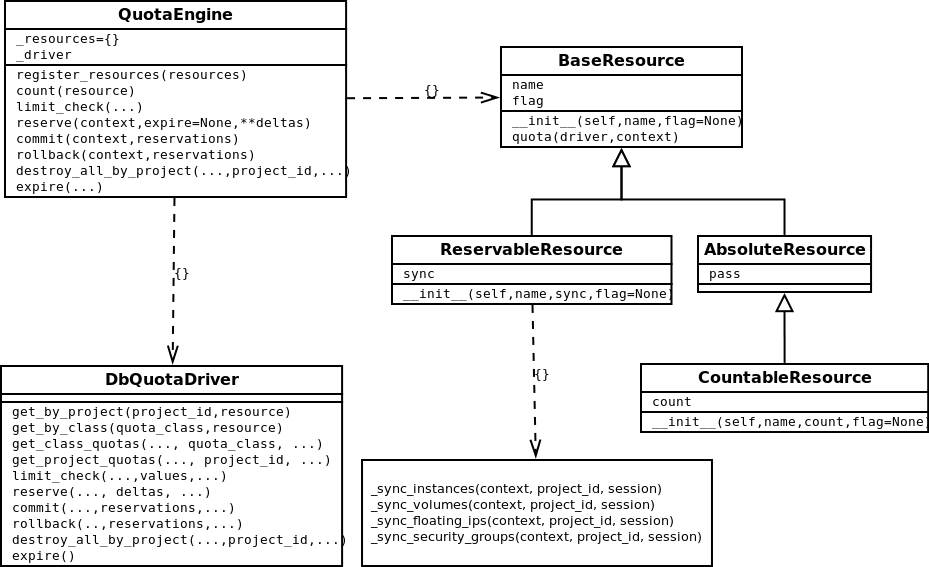
简单的讲就是控制用户资源的数量。在openstack里，**管理**员为每一个工程(project)分配的资源都是有一定限制的，这些资源包括实例（instance）、cpu、内存、存储空间等等，不能让一个工程无限制的使用资源，所以**配额管理**针对的单位是工程(project)。先来个感性的认识，看一下dashboard里的一个工程的overview：



**管理**员给这个工程的资源**配额**是最多创建10个实例，最多使用20个vcpu，最多使用5G的内存==，只要达到某一个资源的使用上限，就会出现异常，这就是**配额管理**。

二、如何实现

nova里**配额管理**是在~/nova/quota.py中实现的，首先来看一下这个模块的静态类图：



使用时是直接调用QuotaEngine类中的方法，而真正实现功能的是DbQuotaDriver这个类，在DbQuotaDriver类中比较重要的是reserve(), commit(), rollback()这三个方法，这三个方法共同构成了**配额管理**一个很重要的特性：事务性，这里所说的事务性，就是当资源分配失败或者已达到**配额**上限等这些异常情况发生时，对已经修改过的数据库，回滚到分配之前的状态，如果不设置成这种特性，那**配额管理**就乱套了。

资源在**配额管理**中封装成了一个简单的类体系，资源被分为三类：ReservableResource, AbsoulteResource, CountableResource，至于这三种分类的区别，我现在还不是很清楚，在nova中，是这样来分类的：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. resources = [
2. ReservableResource('instances', \_sync\_instances, 'quota\_instances'),
3. ReservableResource('cores', \_sync\_instances, 'quota\_cores'),
4. ReservableResource('ram', \_sync\_instances, 'quota\_ram'),
5. ReservableResource('volumes', \_sync\_volumes, 'quota\_volumes'),
6. ReservableResource('gigabytes', \_sync\_volumes, 'quota\_gigabytes'),
7. ReservableResource('floating\_ips', \_sync\_floating\_ips,
8. 'quota\_floating\_ips'),
9. AbsoluteResource('metadata\_items', 'quota\_metadata\_items'),
10. AbsoluteResource('injected\_files', 'quota\_injected\_files'),
11. AbsoluteResource('injected\_file\_content\_bytes',
12. 'quota\_injected\_file\_content\_bytes'),
13. AbsoluteResource('injected\_file\_path\_bytes',
14. 'quota\_injected\_file\_path\_bytes'),
15. ReservableResource('security\_groups', \_sync\_security\_groups,
16. 'quota\_security\_groups'),
17. CountableResource('security\_group\_rules',
18. db.security\_group\_rule\_count\_by\_group,
19. 'quota\_security\_group\_rules'),
20. CountableResource('key\_pairs', db.key\_pair\_count\_by\_user,
21. 'quota\_key\_pairs'),
22. ]

至于那几个\_sync\_\*()函数，是作为ReservableResource类的成员变量存在的，是用来同步数据库的(下面会讲到如何同步)，他们实现的功能是实时的从数据库中查询出正在使用的资源的数量。

**配额管理**在数据库中主要涉及到4个表：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. Quotas: Represents a single quota override **for** a project.
2. mysql> show columns **from** quotas;
3. +------------+--------------+------+-----+---------+----------------+
4. | Field      | Type         | Null | **Key** | **Default** | Extra          |
5. +------------+--------------+------+-----+---------+----------------+
6. | id         | **int**(11)      | **NO**   | PRI | NULL    | auto\_increment |
7. | created\_at | datetime     | YES  |     | NULL    |                |
8. | updated\_at | datetime     | YES  |     | NULL    |                |
9. | deleted\_at | datetime     | YES  |     | NULL    |                |
10. | deleted    | tinyint(1)   | YES  |     | NULL    |                |
11. | project\_id | **varchar**(255) | YES  |     | NULL    |                |
12. | resource   | **varchar**(255) | **NO**   |     | NULL    |                |
13. | hard\_limit | **int**(11)      | YES  |     | NULL    |                |
14. +------------+--------------+------+-----+---------+----------------+

17. QuotaClass: Represents a single quota override **for** a quota class.
18. mysql> show columns **from** quota\_classes;
19. +------------+--------------+------+-----+---------+----------------+
20. | Field      | Type         | Null | **Key** | **Default** | Extra          |
21. +------------+--------------+------+-----+---------+----------------+
22. | created\_at | datetime     | YES  |     | NULL    |                |
23. | updated\_at | datetime     | YES  |     | NULL    |                |
24. | deleted\_at | datetime     | YES  |     | NULL    |                |
25. | deleted    | tinyint(1)   | YES  |     | NULL    |                |
26. | id         | **int**(11)      | **NO**   | PRI | NULL    | auto\_increment |
27. | class\_name | **varchar**(255) | YES  | MUL | NULL    |                |
28. | resource   | **varchar**(255) | YES  |     | NULL    |                |
29. | hard\_limit | **int**(11)      | YES  |     | NULL    |                |
30. +------------+--------------+------+-----+---------+----------------+
32. QuotaUsage: Represents the **current** usage **for** a given resource.
33. mysql> show columns **from** quota\_usages;
34. +---------------+--------------+------+-----+---------+----------------+
35. | Field         | Type         | Null | **Key** | **Default** | Extra          |
36. +---------------+--------------+------+-----+---------+----------------+
37. | created\_at    | datetime     | YES  |     | NULL    |                |
38. | updated\_at    | datetime     | YES  |     | NULL    |                |
39. | deleted\_at    | datetime     | YES  |     | NULL    |                |
40. | deleted       | tinyint(1)   | YES  |     | NULL    |                |
41. | id            | **int**(11)      | **NO**   | PRI | NULL    | auto\_increment |
42. | project\_id    | **varchar**(255) | YES  | MUL | NULL    |                |
43. | resource      | **varchar**(255) | YES  |     | NULL    |                |
44. | in\_use        | **int**(11)      | **NO**   |     | NULL    |                |
45. | reserved      | **int**(11)      | **NO**   |     | NULL    |                |
46. | until\_refresh | **int**(11)      | YES  |     | NULL    |                |
47. +---------------+--------------+------+-----+---------+----------------+
49. Reservation: Represents a resource reservation **for** quotas.
50. usage\_id **is** the foreign\_key **of** quota\_usages.
51. mysql> show columns **from** reservations;
52. +------------+--------------+------+-----+---------+----------------+
53. | Field      | Type         | Null | **Key** | **Default** | Extra          |
54. +------------+--------------+------+-----+---------+----------------+
55. | created\_at | datetime     | YES  |     | NULL    |                |
56. | updated\_at | datetime     | YES  |     | NULL    |                |
57. | deleted\_at | datetime     | YES  |     | NULL    |                |
58. | deleted    | tinyint(1)   | YES  |     | NULL    |                |
59. | id         | **int**(11)      | **NO**   | PRI | NULL    | auto\_increment |
60. | uuid       | **varchar**(36)  | **NO**   |     | NULL    |                |
61. | usage\_id   | **int**(11)      | **NO**   | MUL | NULL    |                |
62. | project\_id | **varchar**(255) | YES  | MUL | NULL    |                |
63. | resource   | **varchar**(255) | YES  |     | NULL    |                |
64. | delta      | **int**(11)      | **NO**   |     | NULL    |                |
65. | expire     | datetime     | YES  |     | NULL    |                |
66. +------------+--------------+------+-----+---------+----------------+

quotas表记录了分配给每个工程的各种资源的最大限额，hard\_limit值就是保存最大限额的。

quota\_classes表现在还不太明白有什么用，在程序里是先查quotas表，找不到相关资源的话，再去quota\_classes表中查，如果还找不到的话，就使用默认值。

quota\_usages表记录了每个工程当前使用的各种资源的数量，in\_use值就是保存正在使用的资源数量的，程序中就是通过更新这个in\_use值来实现**配额管理**的，至于reserv的作用，现在还不太明白。

reservations表记录的是每次分配的各种资源的变化值，即delta保存的值，它和quota\_usages通过usage\_id外键关联。这个表中的记录是不更新的，每次分配都会相应的增加记录。

好了，下面结合这4个表，来详细说一下构成事务性特性的那三个方法：

1. reserv()

这个函数的作用就是判断一下如果分配给请求的资源的数量给这个工程，是否会超出限额，如果超出就报异常，如果没有超出，就更新一下数据库中quota\_usages表的in\_use值。跟踪源码，最终实现功能的是下面的源代码：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. @require\_context
2. **def** quota\_reserve(context, resources, quotas, deltas, expire,
3. until\_refresh, max\_age):
4. elevated = context.elevated()
5. session = get\_session()
6. with session.begin():
7. # Get the current usages
8. usages = \_get\_quota\_usages(context, session)#从quota\_usages表中获得当前工程的各种资源的使用情况
10. # Handle usage refresh
11. work = set(deltas.keys())
12. **while** work:
13. resource = work.pop()
15. # Do we need to refresh the usage?
16. refresh = False
18. #如果当前的resource不在当前工程所使用的资源列表中，那么就把该资源添加进去，并且在数据库中增加一条相应的记录。
19. #并且in\_use和reserv都置为0.
20. **if** resource **not** **in** usages:
21. usages[resource] = quota\_usage\_create(elevated,
22. context.project\_id,
23. resource,
24. 0, 0,
25. until\_refresh **or** None,
26. session=session)
27. refresh = True
29. #如果当前的resource在当前工程的使用列表中，并且该资源的in\_use小于0，说明不同步，则refresh
30. **elif** usages[resource].in\_use < 0:
31. # Negative in\_use count indicates a desync, so try to
32. # heal from that...
33. refresh = True
35. #如果当前resource的until\_refresh不为空，那么将其减1，若减1之后，小于0，则refresh
36. **elif** usages[resource].until\_refresh **is** **not** None:
37. usages[resource].until\_refresh -= 1
38. **if** usages[resource].until\_refresh <= 0:
39. refresh = True
41. #如果max\_age不为空，并且该资源更新的时间减去当前的时间大于max\_age，那么就执行刷新
42. #max\_age==0, 使用的是FLAGS中的值
43. **elif** max\_age **and** (usages[resource].updated\_at -
44. timeutils.utcnow()).seconds >= max\_age:
45. refresh = True
47. # OK, refresh the usage
48. **if** refresh:
49. # Grab the sync routine
50. sync = resources[resource].sync #获得同步函数：\_sync\_\*()，这些函数定义在quota模块中，不同的资源有不同的同步函数
52. updates = sync(elevated, context.project\_id, session) #查询出当前正在使用的资源的一些情况，是实时的情况
53. **for** res, in\_use **in** updates.items():
54. # Make sure we have a destination for the usage!
55. **if** res **not** **in** usages:#如果实时的使用的资源没有在usages中，那么把它添加进去
56. usages[res] = quota\_usage\_create(elevated,
57. context.project\_id,
58. res,
59. 0, 0,
60. until\_refresh **or** None,
61. session=session)
63. # Update the usage
64. usages[res].in\_use = in\_use #更新usages中该resource的in\_use
65. usages[res].until\_refresh = until\_refresh **or** None
67. work.discard(res)
69. # Check for deltas that would go negative
70. #>检查in\_use加上delta之后，可能小于0的情况
71. unders = [resource **for** resource, delta **in** deltas.items()
72. **if** delta < 0 **and**
73. delta + usages[resource].in\_use < 0]
75. #>检查这个resource的hard\_limit是否小于in\_use+reserved+delta之和
76. overs = [resource **for** resource, delta **in** deltas.items()
77. **if** quotas[resource] >= 0 **and** delta >= 0 **and**
78. quotas[resource] < delta + usages[resource].total]
80. # Create the reservations
81. #>如果没有超过的话，更新reservations表，再次更新usages
82. **if** **not** overs:
83. reservations = []
84. **for** resource, delta **in** deltas.items():
85. reservation = reservation\_create(elevated,
86. str(utils.gen\_uuid()),#uuid，唯一
87. usages[resource],#当前资源在quota\_usages表中的使用情况（更新过的）
88. context.project\_id,
89. resource, delta, expire, #资源，变化值，超期值
90. session=session)
91. reservations.append(reservation.uuid)
93. **if** delta > 0:
94. usages[resource].reserved += delta  #更新usages中的reserved值，加上变化值
96. # Apply updates to the usages table
97. # 更新的是quota\_usages表
98. **for** usage\_ref **in** usages.values():
99. usage\_ref.save(session=session)
101. **if** unders:
102. LOG.warning(\_("Change will make usage less than 0 for the following "
103. "resources: %(unders)s") % locals())
104. **if** overs:
105. usages = dict((k, dict(in\_use=v['in\_use'], reserved=v['reserved']))
106. **for** k, v **in** usages.items())
107. **raise** exception.OverQuota(overs=sorted(overs), quotas=quotas,
108. usages=usages)
110. **return** reservations

这个函数执行的过程主要分为以下几步：

（1）同步

为什么要同步呢？这里同步的是什么呢？因为在为工程分配资源时，可能有各种特殊情况导致quota\_usages表中记录的in\_use不准确，需要得到当前实际使用的资源的情况，更新一下in\_use，得到真实的资源使用情况。这里的特殊情况有一下4个：

          1）当前申请的资源没有在quota\_usages表中记录

          2）当前申请的资源在quota\_usages表中的in\_use值小于0

          3）当前申请的资源的until\_refresh值不为空，减1之后小于0

          4）当前申请的资源在quota\_usages表中更新的时间减去现在的时间大于max\_age

如果符合这四种情况之一，就执行同步。同步时，是调用当前资源的\_sync\_\*()函数，去相关的表中查询实时的使用情况，比如\_sync\_instances()就是去instances表中查询出当前工程的instances数量，vcpu之和，ram之和，然后以字典的方式返回。然后根据这些实时的数据，更新in\_use值。

（2）检查

根据各种资源的**配额**，和变化的情况(delta)，来检查两种极端的情况:under和over。under是检查delta为负数的情况，即执行了删除等操作，使delta为负，in\_use减少，导致in\_use值可能小于0。over是检查delta为正数时的情况，in\_use+delta就有可能大于最大的限额了。这里只对over的情况进行处理，即它只关心上限，不关心下限。如果没有over的话，就有下面的第(3)步了。如果over的话，就直接进行第(4)步。

（3）向reservations表中增加记录，记录申请资源的delta值。

（4）把in\_use值写入到quota\_usages表中保存。（不论under还是over都执行）

（5）如果over，即超出最大限额，则报出OverQuota异常。

2. rollback()

reserv()中如果没报异常的话，也不是万事大吉的，还可能有别的原因导致不能分配请求的资源，比如说当前申请的内存大小在**配额**的限额之下，但是它超出了镜像的内存，这种情况也是不能分配资源的，所以这个函数是在reserv()执行正常的情况下执行的，它的作用是回滚在reserv()函数中更改的数据。此处主要改变的数据是reserved值。因为在reserv()中，如果没有over的话，reserv值会执行reserved+=delta，而在这里，它又执行了reserved-=delta，使它回归到以前的值。主要功能代码如下：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. @require\_context
2. **def** reservation\_rollback(context, reservations):
3. session = get\_session()
4. with session.begin():
5. usages = \_get\_quota\_usages(context, session)
7. **for** reservation **in** \_quota\_reservations(session, context, reservations):
8. usage = usages[reservation.resource]
9. **if** reservation.delta >= 0:
10. usage.reserved -= reservation.delta
12. reservation.delete(session=session)
14. **for** usage **in** usages.values():
15. usage.save(session=session)

3. commit()

这个函数是在reserv()函数执行完之后，没有报OverQuota异常的情况下执行的，即本次请求的资源加上当前工程使用资源的数量没有超过限额，它的作用是将in\_use加上变化值(delta)，然后更新到quota\_usages表中。主要功能源码如下：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. @require\_context
2. **def** reservation\_commit(context, reservations):
3. session = get\_session()
4. with session.begin():
5. usages = \_get\_quota\_usages(context, session)# 查询quota\_usages表，获得当前工程中记录的资源的使用情况
7. **for** reservation **in** \_quota\_reservations(session, context, reservations):
8. usage = usages[reservation.resource]#某个资源当前记录的使用情况
9. **if** reservation.delta >= 0:
10. usage.reserved -= reservation.delta
11. usage.in\_use += reservation.delta
13. reservation.delete(session=session)#从reservations表中删除这个reservation的记录
15. **for** usage **in** usages.values():#再次更新quota\_usages表
16. usage.save(session=session)

注意上面reserv()中的in\_use值，它只是分配资源之前，保存的是当前工程的资源使用情况，用来检查是否超过限额的，它本身并没有加上或减去delta，即并没有执行in\_use+=delta，执行的是in\_use+delta，因为它只是检查一下是否超额，如果超额了，in\_use就不该变化的，所以in\_use值只是分配本次请求资源之前的当前工程的资源使用情况，而in\_use+=delta才是分配本次请求资源之后的情况。所以这个commit()函数必须是在reserv()执行之后，没报异常的情况下才执行的。

通过分析，可以知道这三个函数的执行顺序是先执行reserv()，再执行rollback()，再执行commit()，如果没有按这个顺序来的话，就有可能出现问题。

三、如何使用

通过上面的分析，怎么使用应该很清楚了，这里就举一个创建实例的过程中使用到**配额管理**的例子。创建实例之前，首先要检查申请的资源是否超出了该工程的限额，所以这部分功能是在向rabbitmq发出rpc call/cast请求之前来执行的。创建一个实例需要很多资源，这里创建资源使用的是flavor是m1.tiny, 即申请的资源是1个instance, 1个vcpu, 512M 内存。来看一下主要的代码：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. **def** \_create\_instance(self, context, instance\_type,
2. image\_href, kernel\_id, ramdisk\_id,
3. min\_count, max\_count,
4. display\_name, display\_description,
5. key\_name, key\_data, security\_group,
6. availability\_zone, user\_data, metadata,
7. injected\_files, admin\_password,
8. access\_ip\_v4, access\_ip\_v6,
9. requested\_networks, config\_drive,
10. block\_device\_mapping, auto\_disk\_config,
11. reservation\_id=None, scheduler\_hints=None):
13. ………
15. **if** **not** min\_count:
16. min\_count = 1
17. **if** **not** max\_count:
18. max\_count = min\_count
19. ………
21. # 主要就是检查instances,ram,cores是否超过<strong>配额</strong>。如果没有超过，则返回instances的数目和他们三个的reservation的uuid
22. num\_instances, quota\_reservations = self.\_check\_num\_instances\_quota(
23. context, instance\_type, min\_count, max\_count)
25. ………
27. **if** instance\_type['memory\_mb'] < int(image.get('min\_ram') **or** 0):
28. QUOTAS.rollback(context, quota\_reservations)
29. **raise** exception.InstanceTypeMemoryTooSmall()
30. **if** instance\_type['root\_gb'] < int(image.get('min\_disk') **or** 0):
31. QUOTAS.rollback(context, quota\_reservations)
32. **raise** exception.InstanceTypeDiskTooSmall()
34. ………
36. instances = []
37. instance\_uuids = []
38. **try**:
39. **for** i **in** xrange(num\_instances):
40. options = base\_options.copy()
41. instance = self.create\_db\_entry\_for\_new\_instance(
42. context, instance\_type, image, options,
43. security\_group, block\_device\_mapping)
44. instances.append(instance)
45. instance\_uuids.append(instance['uuid'])
46. **except** Exception:
47. # Clean up as best we can.
48. with excutils.save\_and\_reraise\_exception():
49. **try**:
50. **for** instance\_uuid **in** instance\_uuids:
51. self.db.instance\_destroy(context,
52. instance\_uuid)
53. **finally**:
54. QUOTAS.rollback(context, quota\_reservations)
56. QUOTAS.commit(context, quota\_reservations)
58. …………
60. **return** (instances, reservation\_id)

63. **def** \_check\_num\_instances\_quota(self, context, instance\_type, min\_count,
64. max\_count):
66. req\_cores = max\_count \* instance\_type['vcpus'] #req\_cores=1
67. req\_ram = max\_count \* instance\_type['memory\_mb'] #req\_ram=512
69. **try**:
70. reservations = QUOTAS.reserve(context, instances=max\_count,
71. cores=req\_cores, ram=req\_ram)
72. **except** exception.OverQuota as exc:
73. …………
74. **raise** exception.TooManyInstances(overs=overs,
75. req=requested[resource],
76. used=used, allowed=total\_allowed,
77. resource=resource)
79. **return** max\_count, reservations

主要是\_check\_num\_instances\_quota()函数中的QUOTAS.reserv(context, instances=max\_count, cores=req\_cores, ram=req\_ram)，其中参数instances=1, cores=1, ram=512。在这里检查申请的资源是否超过限额。如果没有超出限额的话，看其他情况是否正常，如果不正常的话，则rollback()，如果一切正常的话，commit()提交。这里就可以验证刚才在第3步中的分析，先执行reserv()，再执行rollback()，再执行commit()，使整个**配额管理**在一个事务中。

四、测试

下面以创建实例为例，来进行一下测试，观看一下数据库中数据的变化。测试过程如下：

首先创建一个m1.tiny类型的实例，然后再创建一个m1.tiny类型的实例，然后再删除其中一个实例，注意看数据库的变化。然后再创建实例达到第11个，因为admin工程的默认实例限额为10个，所以创建第11个实例时，会报出异常。

1. 创建第一个实例及数据库数据的变化

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. $ nova boot --flavor 1 --image 6a5a1117-b7cb-4f14-9ab1-0fa9500542a5 --key\_name pubkey-01 instance-01
2. $ nova list
3. +--------------------------------------+-------------+--------+------------------+
4. | ID                                   | **Name**        | Status | Networks         |
5. +--------------------------------------+-------------+--------+------------------+
6. | 46013b5c-d246-4544-9be1-6c33398357b8 | instance-01 | ACTIVE | private=10.0.0.2 |
7. +--------------------------------------+-------------+--------+------------------+
9. mysql> **select** id,uuid,project\_id,hostname,vcpus,memory\_mb **from** instances;
10. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
11. | id | uuid                                 | project\_id                       | hostname    | vcpus | memory\_mb |
12. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
13. |  1 | 46013b5c-d246-4544-9be1-6c33398357b8 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instance-01 |     1 |       512 |
14. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
16. mysql> **select** id,created\_at,deleted,project\_id,resource,hard\_limit **from** quotas;
17. Empty **set** (0.00 sec)
19. mysql> **select** id,created\_at,deleted,class\_name,resource,hard\_limit **from** quota\_classes;
20. Empty **set** (0.00 sec)
22. mysql> **select** id,deleted,project\_id,resource,in\_use,reserved,until\_refresh **from** quota\_usages;
23. +----+---------+----------------------------------+-----------+--------+----------+---------------+
24. | id | deleted | project\_id                       | resource  | in\_use | reserved | until\_refresh |
25. +----+---------+----------------------------------+-----------+--------+----------+---------------+
26. |  1 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |      1 |        0 |          NULL |
27. |  2 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |    512 |        0 |          NULL |
28. |  3 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |      1 |        0 |          NULL |
29. +----+---------+----------------------------------+-----------+--------+----------+---------------+
31. mysql> **select** id,deleted,usage\_id,project\_id,resource,delta,expire **from** reservations;
32. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
33. | id | deleted | usage\_id | project\_id                       | resource  | delta | expire              |
34. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
35. |  1 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |     1 | 2012-11-26 08:55:26 |
36. |  2 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   512 | 2012-11-26 08:55:26 |
37. |  3 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |     1 | 2012-11-26 08:55:26 |
38. +----+---------+----------+----------------------------------+-----------+-------+---------------------+

可以看到quota\_usages表中的in\_use值分别为1, 512, 1,  reservations表中的delta值分别为1, 512, 1

2.  创建第二个实例及数据库的变化

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. $ nova boot --flavor 1 --image 6a5a1117-b7cb-4f14-9ab1-0fa9500542a5 --key\_name pubkey-01 instance-02
2. $ nova list
3. +--------------------------------------+-------------+--------+------------------+
4. | ID                                   | **Name**        | Status | Networks         |
5. +--------------------------------------+-------------+--------+------------------+
6. | 46013b5c-d246-4544-9be1-6c33398357b8 | instance-01 | ACTIVE | private=10.0.0.2 |
7. | 2a0fb8ca-9f5d-4751-b95b-43d6fe6c66d2 | instance-02 | ACTIVE | private=10.0.0.3 |
8. +--------------------------------------+-------------+--------+------------------+
10. mysql> **select** id,uuid,project\_id,hostname,vcpus,memory\_mb **from** instances;
11. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
12. | id | uuid                                 | project\_id                       | hostname    | vcpus | memory\_mb |
13. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
14. |  1 | 46013b5c-d246-4544-9be1-6c33398357b8 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instance-01 |     1 |       512 |
15. |  2 | 2a0fb8ca-9f5d-4751-b95b-43d6fe6c66d2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instance-02 |     1 |       512 |
16. +----+--------------------------------------+----------------------------------+-------------+-------+-----------+
18. mysql> **select** id,created\_at,deleted,project\_id,resource,hard\_limit **from** quotas;
19. Empty **set** (0.00 sec)
21. mysql> **select** id,created\_at,deleted,class\_name,resource,hard\_limit **from** quota\_classes;
22. Empty **set** (0.00 sec)
24. mysql> **select** id,deleted,project\_id,resource,in\_use,reserved,until\_refresh **from** quota\_usages;
25. +----+---------+----------------------------------+-----------+--------+----------+---------------+
26. | id | deleted | project\_id                       | resource  | in\_use | reserved | until\_refresh |
27. +----+---------+----------------------------------+-----------+--------+----------+---------------+
28. |  1 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |      2 |        0 |          NULL |
29. |  2 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   1024 |        0 |          NULL |
30. |  3 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |      2 |        0 |          NULL |
31. +----+---------+----------------------------------+-----------+--------+----------+---------------+
33. mysql> **select** id,deleted,usage\_id,project\_id,resource,delta,expire **from** reservations;
34. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
35. | id | deleted | usage\_id | project\_id                       | resource  | delta | expire              |
36. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
37. |  1 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |     1 | 2012-11-26 08:55:26 |
38. |  2 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   512 | 2012-11-26 08:55:26 |
39. |  3 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |     1 | 2012-11-26 08:55:26 |
40. |  4 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |     1 | 2012-11-26 09:09:44 |
41. |  5 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   512 | 2012-11-26 09:09:44 |
42. |  6 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |     1 | 2012-11-26 09:09:44 |
43. +----+---------+----------+----------------------------------+-----------+-------+---------------------+

可以看到quota\_usages表中的in\_use更新了，在原来的**基础**上，又加上了创建的第二个实例的资源数量，变为2, 1024, 2，而reservations表中的数据是不更新的，只是又增加了三条记录。

3.  删除第二个实例及数据库数据的变化

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. $ nova **delete** 2a0fb8ca-9f5d-4751-b95b-43d6fe6c66d2
2. $ nova list
3. +--------------------------------------+-------------+--------+------------------+
4. | ID                                   | **Name**        | Status | Networks         |
5. +--------------------------------------+-------------+--------+------------------+
6. | 46013b5c-d246-4544-9be1-6c33398357b8 | instance-01 | ACTIVE | private=10.0.0.2 |
7. +--------------------------------------+-------------+--------+------------------+
9. mysql> **select** id,deleted,uuid,project\_id,hostname,vcpus,memory\_mb **from** instances;
10. +----+---------+--------------------------------------+----------------------------------+-------------+-------+-----------+
11. | id | deleted | uuid                                 | project\_id                       | hostname    | vcpus | memory\_mb |
12. +----+---------+--------------------------------------+----------------------------------+-------------+-------+-----------+
13. |  1 |       0 | 46013b5c-d246-4544-9be1-6c33398357b8 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instance-01 |     1 |       512 |
14. |  2 |       1 | 2a0fb8ca-9f5d-4751-b95b-43d6fe6c66d2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instance-02 |     1 |       512 |
15. +----+---------+--------------------------------------+----------------------------------+-------------+-------+-----------+
17. mysql> **select** id,created\_at,deleted,project\_id,resource,hard\_limit **from** quotas;
18. Empty **set** (0.00 sec)
20. mysql> **select** id,created\_at,deleted,class\_name,resource,hard\_limit **from** quota\_classes;
21. Empty **set** (0.00 sec)
23. mysql> **select** id,deleted,project\_id,resource,in\_use,reserved,until\_refresh **from** quota\_usages;
24. +----+---------+----------------------------------+-----------+--------+----------+---------------+
25. | id | deleted | project\_id                       | resource  | in\_use | reserved | until\_refresh |
26. +----+---------+----------------------------------+-----------+--------+----------+---------------+
27. |  1 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |      1 |        0 |          NULL |
28. |  2 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |    512 |        0 |          NULL |
29. |  3 |       0 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |      1 |        0 |          NULL |
30. +----+---------+----------------------------------+-----------+--------+----------+---------------+
32. mysql> **select** id,deleted,usage\_id,project\_id,resource,delta,expire **from** reservations;
33. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
34. | id | deleted | usage\_id | project\_id                       | resource  | delta | expire              |
35. +----+---------+----------+----------------------------------+-----------+-------+---------------------+
36. |  1 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |     1 | 2012-11-26 08:55:26 |
37. |  2 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   512 | 2012-11-26 08:55:26 |
38. |  3 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |     1 | 2012-11-26 08:55:26 |
39. |  4 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |     1 | 2012-11-26 09:09:44 |
40. |  5 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |   512 | 2012-11-26 09:09:44 |
41. |  6 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |     1 | 2012-11-26 09:09:44 |
42. |  7 |       1 |        1 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | instances |    -1 | 2012-11-26 09:17:30 |
43. |  8 |       1 |        2 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | ram       |  -512 | 2012-11-26 09:17:30 |
44. |  9 |       1 |        3 | aefecee7a7ca4cb3bcfac8e3c9fd27d0 | cores     |    -1 | 2012-11-26 09:17:30 |
45. +----+---------+----------+----------------------------------+-----------+-------+---------------------+

可以看到quota\_usages表中的in\_use值又变化了，分别减去了1, 512, 1 ，变成了1, 512, 1，而reservations表中又增加了三条记录，而delta值为-1, -512, -1，因为是删除实例嘛，所以delta值为负。

4. 创建实例达到11个，观看异常：

$ nova boot --flavor 1 --image 6a5a1117-b7cb-4f14-9ab1-0fa9500542a5 --key\_name pubkey-01 instance-11

ERROR: Quota exceeded for instances: Requested 1, but already used 10 of 10 instances (HTTP 413) (Request-ID: req-2337fb00-d109-46e1-b381-1f4d63df0fd5)

这个异常就是超出限额，\_check\_num\_instances\_quota()函数中报的TooManyInstance异常：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. **class** TooManyInstances(QuotaError):
2. message = \_("Quota exceeded for %(overs)s: Requested %(req)s,"
3. " but already used %(used)d of %(allowed)d %(resource)s")

其实创建实例是可以达到最多10个，但是创建成功的就不一定有10个了，比如我的数据：

[查看文本打印?](http://www.verydemo.com/demo_c152_i3355.html)

1. $ nova list
2. +--------------------------------------+-------------+--------+------------------+
3. | ID                                   | **Name**        | Status | Networks         |
4. +--------------------------------------+-------------+--------+------------------+
5. | 46013b5c-d246-4544-9be1-6c33398357b8 | instance-01 | ACTIVE | private=10.0.0.2 |
6. | 18b4a216-dad3-4e30-8f63-e4fd04ed5cd2 | instance-02 | ACTIVE | private=10.0.0.3 |
7. | 76e545b5-fa43-4ff1-b7f1-f3f1e0ee4ff2 | instance-03 | ACTIVE | private=10.0.0.4 |
8. | a279f645-f733-4f7d-b95a-42c186582b39 | instance-04 | ACTIVE | private=10.0.0.5 |
9. | 3e2c491f-3f3b-48d2-8fbf-c4459956639b | instance-05 | ERROR  |                  |
10. | 8ff24877-b4d1-4b3d-be20-f7fae03113e7 | instance-06 | ERROR  |                  |
11. | 3177a8ba-4c01-46a4-a1c1-904aac71fcd2 | instance-07 | ERROR  |                  |
12. | d4adb08b-c199-48fd-a2f9-024e19ba8bbd | instance-08 | ERROR  |                  |
13. | ada1c610-2a93-4ee5-bd06-f58f18be04dd | instance-09 | ERROR  |                  |
14. | fdd9c770-3bae-48cb-987c-00425e917574 | instance-10 | ERROR  |                  |
15. +--------------------------------------+-------------+--------+------------------+

只成功创建了4个实例，因为我的2G内存已经不够了：

$ free -m

total used free shared buffers cached

Mem: 1974 1824 150 0 20 404

-/+ buffers/cache: 1400 574

Swap: 2382 355 2027

这也从另外一个方面说明了**配额管理**只是管**配额**的而已。

五、问题

写blog之前本来有很多疑问的，但是在写的过程中思考，竟然把一些疑问想明白了，但是还是有一些：

1. quotas和quota\_classes表为空，我想这可能是因为没有配置的原因吧，因为程序中如果没有从这两个表中得到**配额**(hard\_limit)，那么就使用默认的值，所以我的测试里这两个表为空，但是程序正常执行，是因为使用了默认的值。

2. 关于那几个资源分类的，为什么要那样分，他们之间有什么区别，我还不明白。

3. dashboard界面的问题，它的quota界面的限额值是写死了的，还是会根据配置的值来变化，这个我还没有验证过。

4. 关于expire的，我看程序中参数的传递，expire始终为空，但是不知道怎么回事，到数据库中就变成有值的了，而且这个值很奇怪，即使删除了实例，它还是有，为什么呢？

5. 关于quota\_usages表中的reserved值，它究竟是干什么用的呢？它的值，在一个事务之外，始终是为0，不为0的情况只在一个事务当中，这种情况只有调试才会看到，难道它就是只起一个回滚的作用？

6. quota\_classes表不知道是用来做什么的，难道用一个quotas表还不够吗？难道说这个表保存的是每个资源的整体情况？

7. 还有就是一些资源不知道是什么资源，比如injected\_file

六、想法

1. 实习第一天，经理跟我说是要吃苦的，有1/4的人都没坚持下去，走了，唉，当时压力也还挺大的，甚至怀疑起自己的能力，我到底行不行啊？但是每每当我看起代码时，就会有一种莫名的自信，代码也是人写的，我能把握住它。至于吃苦，我不认为学习是一种吃苦，学习就好像吃馒头一样，不动嘴嚼两下，怎么知道它的甜呢？

2. 公司的牛人真的很多，我要好好向他们学习。

3. 云计算，这是我选择的路，这也是我的梦想，也是我实现自己价值的地方，我会坚持下去的。

七、我的不足

1. 首要的不足，效率太低，一个**知识**点，要弄很久才能搞清楚，要提高效率啊。

2. 还不会openstack的调试，不会调试，不能深刻理解。