Computer Lab 2, Part I

This notebook consists of instructions, exercises and questions that form the practical part of Lab II, Part I. In this assignment, you will learn the basics of the OpenStack Python APIs that can be used to interact directly with the IaaS services Nova (compute) and Swift (Object Store). Please prepare your solution and answers to questions directly in this notebook, and export it to PDF. Upload that PDF as to the student portal to complete Part I of the Lab.

In [1]:

```
import os
import swiftclient.client
```

To establish a client connection, we will need to pass a dictionary with information about the tenant, user, credentials and the API Identity endpoint. Here, I have sourced the "openrc.sh file" obtained from the Horizon dashboard in the underlying shell prior to starting the notebook. Hence, in order to actually run the code below, you would need to do the same with your own credentials.

In [2]:

```
config = {'user':os.environ['OS_USERNAME'],
          'key':os.environ['OS_PASSWORD'],
          'tenant_name':os.environ['OS_TENANT_NAME'],
          'authurl':os.environ['OS_AUTH_URL']}
```

First, we obtain a client connection to Swift (we are usign the v2 APIs)

In [3]:

```
conn = swiftclient.client.Connection(auth_version=2, **config)
```

In [4]:

```
# Create a container, use a UUID to make sure it has a globally unique name.
import uuid
bucket_name = "DoktorFalafel".format(str(uuid.uuid4()))
conn.put_container(bucket_name)
```

Question 1:

What does it mean that the object store has a global, flat namespace? What is the practical consequence for you when using it?

In [29]:

```
#See attached questions document for answer.
```

In [14]:

```
# List containers
(response, bucket_list) = conn.get_account()
for bucket in bucket_list:
    print bucket['name']
```

ACCA
AJ_48c95173-4b77-467d-aa64-c934f3efeb5a
AJ_5cda2de6-f80c-40db-9ab6-d189dd1946fb
AJ_bc9904ef-9f6d-4c87-b2d8-ed8bbe9ac8e4
A_a8b211e2-d7e0-4016-bf30-2075ad07158e
A_b989361a-b156-4eb7-b0c2-f747aa13ae31
CarlssonBucket
DoktorFalafel
FarezCon
Harry_lab2_65536b3d-4965-4f8d-a513-5db2f82c5dc7
Harry_lab2_7119c0a9-21dd-4b55-a543-e0a5d0f1e4a9
JustAnotherContainer
Lab1
Labb 3, needed files
Lundetainer
MarcusBucket
Marcus_bucket
MatLab2_023f0b1b-fb91-4f66-bdcf-72f3fc81bcba
MatLab2_2d53a9e7-cc12-45b3-8c02-6c7fe7386127
MatLab2_5d1555a5-df16-40f1-a10c-a6aa2b3a0cf0
MatLab2_63276e91-a470-4414-bb92-bb50c7fa82de
MatLab2_7d11a714-57c3-4d11-bde2-145ff3c0b79d
MatLab2_8ba15c79-32c3-4ccb-8fdb-923fb4b532af
MatLab2_a0ba73f0-4014-4f7d-a563-26845f3740df
MatLab2_c1c2d1fe-4dfc-4a53-bacc-5cc956c93815
MatLab2_d537e697-26f7-4bc1-b35b-8c3d17e7d277
MavaLab2_73c7d646-335f-4b2e-b409-f5913691f84e
MavaLab2_88ee1622-7de2-4c0e-8eb0-732a13f10cae
MavaLab2_a70b7a68-6ba0-4cfe-ac66-1b1475f36080
Nope_container
PrivateContainer
RuulTainer
Sarolab2_a64887d0-5d67-41c2-9cf0-f48a34182c53
Sarolab2_f5578519-b479-42bc-b271-f5fd0085f8f4
Svenssotainer
Vask_Bucket
Why? Because I can
acc_nerisa
anaz
anga6525
asdContainer
haad2548_container
hej
joni2138
lab2CC_67f75ae8-c3ce-4f9b-8fb9-b7f9911a0ec2
lab2CC_90018735-23e6-47dd-a682-e97c7fbcee86
lab2CC_c3ab3ed9-272e-4181-86b3-930567b57ffe
lab2ML_45686723-e3ab-46e4-97f7-7bfad4c8fe7b
lab2ML_6d0f0d8f-5255-49f5-a3ac-89bf6190a415
lab2_008485ad-c159-4204-be94-f7ca5c4bcc9e
lab2_00931654-0a83-42d5-afe4-cad664b499b8
lab2_00f8c232-d78e-47f8-8c74-aa5ff845d200
lab2_02372d69-9e6d-487f-852e-2b4c6a6f9fe9
lab2_02a73c14-a7b6-42a2-ac0c-7fc1299ef613
lab2_02e077bc-606d-4005-8caf-645d23d11a49
lab2_0529f95c-03cf-4b52-a395-c59dccb3c4b1
lab2_0a69b6cc-37e1-4111-aa5a-4ecd6a13b354
lab2_0bb63fbb-7e77-4843-9936-d07b87b69cd5
lab2_0cf9de56-4eb3-4aaf-b770-68f7005ada37
lab2_0d636fbb-71ea-4eca-9999-15b0893b329e
lab2_0dcec9c6-3271-4fec-a129-81d1ef7b816a
lab2_131bb1ae-8322-4141-b42d-a4292d6bfb72
lab2_13449a49-a9d8-4c0b-85bc-62eb18466816
lab2_1368cac2-f82a-4f05-b14a-8ff2caeff71b
lab2_13b66b5f-8dd6-4eb7-b802-1758f047db2a
lab2_14016c03-0998-48c3-a0f7-be3389bf4e5e
lab2_14714d5f-3549-4e33-8303-20a67d214ca8
lab2_14788dcd-53c5-48c6-91fb-9b6546cf3c5d_A
lab2_14e77f32-8498-4e64-944e-e18904e76fad
lab2_16c180e0-730b-426d-b1f4-ba4683b5ead3
lab2_1cf61fd4-5c5b-433c-b425-d90067ecbf66
lab2_1cfbaa23-7c35-4ce2-b0c8-05163d5538a1
lab2_1d734207-9267-460a-8365-9877d5d63fbf
lab2_20627fd3-ceb1-4b14-ad37-e0e757565062
lab2_206f8731-4ff0-4103-84d6-b186e3363469
lab2_22b9adba-1ced-4a71-8e6b-7929c486bbab
lab2_2b4890b3-97e2-4e64-8f45-ad94759eebd9
lab2_324df79d-4d29-48cf-80fa-ae8481d63c6a
lab2_343268c3-f0c4-4f64-abaa-c0beb191ac14

lab2_344aa7ea-f447-44f4-9885-edc6f76fa61c
lab2_3581912a-9d6c-43ce-8e0e-b706e8793daa
lab2_3622d5a9-63f9-4fc6-b3eb-c13bc5adbbfa
lab2_36374f9c-6727-4b96-b86b-22383d60d697
lab2_386f557e-6249-4cb2-8122-c8d3426fbc92
lab2_39fecf5c-b4f7-476f-bfad-fa749cdce5bf
lab2_3a284bb5-a89e-4cf4-93eb-a42f78531194
lab2_3b66d18b-20bf-4c41-8384-3a90919490a5
lab2_3bc61e66-7493-43b7-8668-6a0f2ed5c12b
lab2_3d256a86-f4a3-4a1e-867e-e3dfc1e65c8d
lab2_3da655b1-5fb7-452e-a30f-3e6c180272b7
lab2_3dde5963-2903-49f7-ab0f-555425946df0
lab2_3df085da-cc90-42a3-bc61-83bd19696b66
lab2_3f23e889-510e-41f7-8b09-91a6978b90da
lab2_3f77c1d7-59e7-4292-9af9-af22c40e0723
lab2_3fe6f8ba-4673-4778-a0ab-773decb3f4eb
lab2_4444ef81-5f32-4e98-aa84-9e8b1afbf431
lab2_44cf7db8-203a-4028-8485-8e9c6359af63
lab2_4531d587-f985-4ccf-bfdd-6c69d2811e37
lab2_46899b66-b33d-46f5-80dd-8514f3c4039c
lab2_4731c88c-766f-42c7-820b-23313bea676e
lab2_48bb021c-4cdf-435b-8ddf-7361d6b602bc
lab2_494918fa-267d-4ce2-88b8-3e77ea9f320c
lab2_4952515c-6780-4ec9-a8b5-c01d4ed976eb
lab2_4aa1aad5-e045-404e-83e6-8ac3dc6e59a7
lab2_4c1f8999-2e69-4fc7-af58-3de83aedce42
lab2_4c49147a-c005-4537-a136-213c82cf88db
lab2_4c5620ce-a1d3-411e-90f3-f2ff1f1b76e3
lab2_4cd69e9b-938e-4064-a9e2-8ff632bd2e89
lab2_4ce4c6d5-18aa-4db1-b0bd-b318baeb62a6
lab2_4de7e356-3c72-4da8-912e-6ad72197059d
lab2_4deb0295-8486-41db-9f05-cbf7ebf0d836
lab2_4e030ab9-1c2a-4306-bd72-4111e0684641
lab2_4eba3317-16c3-4912-914e-dd720fc5c4dd
lab2_5486c61a-a7a0-4e13-9eda-adafad36e1a4
lab2_54bb6487-917b-4fd6-bb5f-d01ddd6c13b7
lab2_54cb7a58-202b-41a3-8d22-a0dac6e93eb6
lab2_54dddbfc-104e-4912-a462-0fbecb92a367
lab2_5585561b-9845-4eb3-87ae-c8c7ba746ab6
lab2_55ca2d98-924d-4822-afc3-ac2a7c654f6d
lab2_5693d0e4-dff3-4794-b0bd-5add49a68f74
lab2_59274559-22a8-4693-b740-c42c4214c56a
lab2_59906132-ccfc-44a7-933d-986c2725073e_A
lab2_5a657400-8a1f-4c7d-9f00-8f7758bcdc2a
lab2_611c0435-2df1-4f90-867a-526792f6e38d
lab2_614062b9-df06-45b9-80cb-51db059c22d4
lab2_648d8d4e-2b7a-46d7-819e-3e6893f7777c
lab2_68e4a4fd-6055-458b-b3b3-aaffd81e4d7a
lab2_69bcdd0c-0bae-4fca-891e-cb406b985beb
lab2_6b4c6497-fb4b-4cda-92cc-a10a73f7c9cc
lab2_6def1a03-a878-4ac1-91b4-6aff9f556019
lab2_6e2fc91d-0a90-49af-82fc-ff267b4a6bdc
lab2_72c98f6e-0f0f-4791-8051-8f4a5791104a
lab2_79b8feb6-bdae-4741-83d3-cfd60204fa21
lab2_7a604995-18ad-4505-b850-2ef42ec57104
lab2_7b14a9c7-0095-4956-8341-a34e57eb5e38
lab2_7c351f20-fb9e-438b-af76-868d09eadc8a
lab2_7d3e46b9-d4d1-4832-baaa-63579aa7f6ab
lab2_7e1ad882-787f-4c3f-badb-46f5d33133d2
lab2_7eac835e-7553-4aca-a299-043d61cdece5
lab2_8044cf6a-459e-4e62-9eba-7343a73aeeb9
lab2_80825366-3409-410f-89ab-f2dfec807a80
lab2_82540008-a332-41a0-9a7c-2dfb68f82bb2
lab2_84e9e011-caab-467d-9629-28a0a825a081
lab2_871f423f-ce26-4ce3-be06-6f85c313b57f_A
lab2_88c9d4d0-0479-4d5d-9923-5c137a9096c1
lab2_8abd9b9b-8986-4d6c-8594-130ebb9a3773
lab2_8b07d6bc-055f-47d4-831e-7feedf0a09f3
lab2_8ba4567b-f223-4775-92a8-7fdfe680fd4e
lab2_8bac4ec3-c65a-4704-a6bf-2cbe79625abf
lab2_8c6b428f-5246-43d9-a862-b72ad5836bc2
lab2_8d239771-2d95-4e18-bd90-d3fdc6de73bf
lab2_8d6ab63a-22ac-49dc-b521-175da027745e
lab2_8f804759-017f-4c9a-854a-8ac3153c3d5b
lab2_8fbe91fc-ad8b-4bae-be81-f6cf5d8c1342
lab2_9188da34-fb45-43e2-98bd-82adafdd7193
lab2_9236a513-c934-40a8-8a27-7cd7e8c64a0f
lab2_958a6a57-c40f-494a-8dff-043921257eb5
lab2_97d04ff2-f3f2-438c-bdcf-13a80a441cf8
lab2_99ea414d-21ba-4ea8-82be-0cf1c4e6f3e7

lab2_9b0abb45-56e4-4a9b-a087-9f50a1dcf638
lab2_9d859a5a-16e7-4732-838f-a92577319469
lab2_9f714ac2-464d-4bfe-b771-4c0b978efeb8
lab2_a24b1468-dcde-42ad-ae25-b5910140c9d9
lab2_a3bb5ba0-0666-451e-9638-2567c4eb0423
lab2_a66664eb-d66f-46b0-9385-92c3bd10b2ce
lab2_a6a75903-7211-4d64-a6d1-cff8c336febc
lab2_a6d3e6ab-d047-4cb8-ae47-60690b43fb4a
lab2_a72f9724-a7e5-47b4-af68-9c45b40a5229
lab2_a957df19-5b03-4790-b88c-f877793e89cc
lab2_aaa595db-a51c-40f8-bf56-afd1e354bc9d
lab2_ab1f2225-db78-46c3-b218-a779e425a4c9
lab2_ad21d6a5-3f20-4cb8-a68c-795a3bd810d4
lab2_adf3256d-1842-4d65-81ec-aafc736edda8
lab2_af37f158-b18c-469f-9751-833c6c5da7f4
lab2_b08a695b-3363-4544-a21a-d2e40a866497
lab2_b478b629-7b95-4748-8424-171bf97ee576
lab2_b4d45e6e-f291-49de-8189-d95e6be9b262
lab2_b5086519-259d-4594-af49-26ecbd73056a
lab2_b5d87232-429e-47b3-b82b-c5b6b0944450
lab2_b6e70489-f1b1-4db0-95e4-50ff942e5404_A
lab2_b773a787-e20f-46a6-996d-79ccea7a9ed1
lab2_b85b9305-5e43-4ce6-9f7a-f7b40114f6a7
lab2_b8f8383c-8d79-4f8e-a8f9-739469f9ffe9
lab2_bac6460b-1318-440d-905d-14532d16b252
lab2_bc12a331-2c8e-4dc3-8538-cdbf2dd15350
lab2_be4f6d9d-ce36-4433-aa76-27e4b7f9b18a
lab2_bf9589b5-379e-40e5-b509-1f57794bd0c8
lab2_c325123b-140e-4b82-9567-ead7632b9694
lab2_c6aa6cd8-7d68-41e6-b5bf-5204eb7428c7
lab2_c836c490-c75a-4512-9232-d45b01125965
lab2_c85ab379-166c-447c-bb20-1aba951642b9
lab2_c8c944e7-af7b-4f0c-be39-abf190e98112
lab2_c9304df7-139d-4c9d-abf9-384b5c4ce065
lab2_ca39750f-31b4-43a2-a0ad-63c88633b2ac
lab2_caf947fa-63be-4d2a-ac6f-616a6cb302f7
lab2_cc6022fc-4197-4d4b-afdb-abe68115e578
lab2_ce241e3d-831e-4f23-8175-7b7013f45483
lab2_ce66a3ec-fb47-4bfa-9f24-6e19030a505e
lab2_ce9e058c-d2b1-4951-8c2c-f4bdb1959069
lab2_ceae12a2-fad5-4f56-9b5e-213de8a12272
lab2_cf56b649-a916-4c6f-99c3-14c46f8fcdbe
lab2_d04107ae-0a21-4658-abdd-95fc438f7943
lab2_d27c17c5-7e75-4e4b-9cca-fa5273bb5f19
lab2_d2fc3a94-080d-477d-9fcc-78bd22d0b598
lab2_d509e90b-2a1b-4c6f-aa0f-ee8f63544512
lab2_d9b2b6ae-7407-4c27-87f5-7ad8cd96b48a
lab2_da6a09c6-b72b-475b-b28d-5521be35fde8
lab2_da6ae4e9-6cf1-4ec5-b281-6f93d0c9eecf
lab2_db241c6d-f6f1-41eb-838b-2eb4878abb04
lab2_dbfe24df-4e41-4882-a72e-e9d2c38c5fb5
lab2_de3d0216-7af8-4633-b93c-c1d61628dbfc
lab2_e0532a43-a314-48a3-9807-f34a68737479
lab2_e07def4d-fe8f-493f-86d1-3ccff88e1732
lab2_e199a905-bde2-4904-8df8-0bf859dc05a9
lab2_e3385a11-19f8-46da-983a-c1563776d9b5
lab2_e6e46375-f4c4-457a-9f4a-a105775f5102
lab2_e732a2e4-9491-4c6f-8c39-c3f62bb987be
lab2_e79615cb-f99e-4832-907a-f250b339c6a1
lab2_e93a0588-7b16-4246-8b8b-71038b90e9d4
lab2_e9f0d343-088c-4e97-834e-5c7ab681280c
lab2_ea457775-ffc5-4f87-a18f-9bc358c9f5fb
lab2_ea6ebf31-10d1-4705-aaf3-32fceadd433a
lab2_ea8c0312-32b5-44e1-875c-1dd8d5f21817
lab2_eb353d7f-fde8-400b-b89e-2ae8414a4f0e
lab2_ebf4931a-c6ad-4139-84c2-74c85e3837fb
lab2_ee6fec6f-5b70-43e0-a71b-3ea437433491
lab2_f052ef54-81ba-487b-96df-69361b11e167
lab2_f30555be-d8fa-4503-bf0c-c038a1564df8
lab2_f36b0b4d-5c44-45a9-b5de-fb82c443ff37
lab2_f5ccae8b-d070-4beb-b525-76cc49149518
lab2_f5e8a316-28e8-4f6a-81a6-cc6ab7348424
lab2_f61acf89-61f0-4595-9c10-c51057448291
lab2_f63cf641-370e-4353-9eee-a81cbdb87de6
lab2_f79a7c4b-30ac-41ff-b83c-7a47ea9d5a72
lab2_f84761ff-8fff-4b34-892e-7508c11b5f21
lab2_f99ce126-7cc9-40f2-a6f4-7dc0514c9e63
lab2_fb65d4a2-1531-4c0f-a079-6d37d64a7cd1
lab2_fe9ede3a-3a28-4729-a90e-3bd95e361c6f
lab2_mm_743e283c-49f1-484c-b4ef-87866ff8d7f8

```
lab2_mm_c47bf1be-10fb-4072-979d-9f7c9eb9a0bb
lab2_mm_d138c2b1-99d1-4335-a356-5d913e0b80d8
lab2_mm_ee98f366-8a4c-4628-8423-986891d3e609
lab2_puan
ljoni2138
lufr
lufr2071
noaabukk
ruul_bucket_ah
ryman
testContainer
testContainer_Saim
testContainerrrr
test_bucket
testcontainer2
testcontainerAndrew
testcontainerKalle
testcontainerS
testcontainerr
tweets
```

In [5]:

```
# Put an object in the container
object_id = conn.put_object(bucket_name, "Test", "Hej Swift" )
```

Excercise 1:

Try to measure the speed with which you can put and get objects to and from Swift usign the API. Conduct your experiment several times to gather statistic and plot a) A estimated distribution of the time taken (in wall clock) to put and read an object of size 10MB in your swift container and b) vary the size of the object from 10kB to 100MB and plot the put and get throughput (in MB/s) times as a function of object size (for the smaller data sizes, you might need to repeat the experiment many times and obtain a statistical average). Include the resulting graphs and a description of your experiment in the report.

In [16]:

```
import time
%pylab inline
time_list = []
filepath = '/home/felix/10MB'
for a in range(0, 3):
   with open(filepath) as file:
        start = time.time()
        object_id = conn.put_object(bucket_name, filepath, contents = file.read(), content_type = 'text/plain')
        #get_object = conn.get_object(bucket_name, '10MB')
        end = time.time()
        time_taken = end - start
        time_list.append(time_taken)
mean_time = [sum(time_list)/3]
print 'Estimated time per 10 mb package: %s' % mean_time
# will  make matplotlib/pylab available and plots will be displayed directly in the notebook, for example
#plt.plot(time_list)
#plt.ylabel('Time')
#plt.xlabel('Object')
#plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
Estimated time per 10 mb package: [12.297785758972168]
```

In [7]:

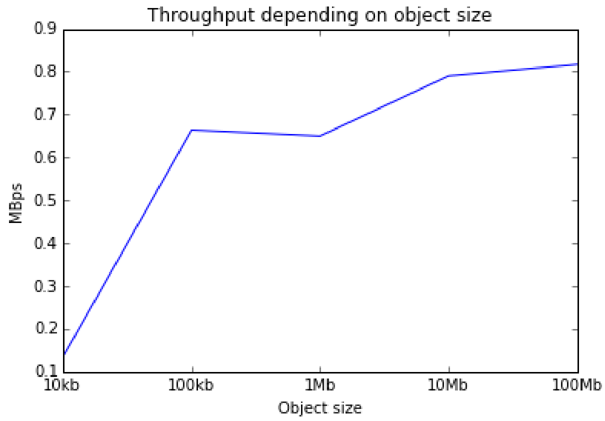
```
# Implement you solution here.
import time
%pylab inline
#10 kb object is put in container 50 times.
ten_kb_time_list = []
filepath10kb = '/home/felix//10kb.dat'
for a in range(0, 50):
    with open(filepath10kb) as file:
        start = time.time()
        object_id = conn.put_object(bucket_name, filepath10kb, contents = file.read(), content_type = 'text/plai
n')
    end = time.time()
    time_taken = end - start
    ten_kb_time_list.append(time_taken)
    mean_ten_kb = [sum(ten_kb_time_list)/50]
ten_kb_time_list_s =  [0.01/x for x in mean_ten_kb]
print 'MBps for 10kb: %s' % ten_kb_time_list_s

#100kb object is put in container 20 times.
hundred_kb_time_list = []
filepath100kb = '/home/felix//100kb.dat'
for a in range(0, 20):
    with open(filepath100kb) as file:
        start = time.time()
        object_id = conn.put_object(bucket_name, filepath100kb, contents = file.read(), content_type = 'text/pla
in')
    end = time.time()
    time_taken = end - start
    hundred_kb_time_list.append(time_taken)
mean_hundred_kb = [sum(hundred_kb_time_list)/20]
hundred_kb_time_list_s =  [0.1/x for x in mean_hundred_kb]
print 'MBps for 100kb: %s' % hundred_kb_time_list_s

#1 mb object is put in container 1 time.
one_mb_time_list = []
filepath1mb = '/home/felix//1mb.dat'
with open(filepath1mb) as file:
    start = time.time()
    object_id = conn.put_object(bucket_name, filepath1mb, contents = file.read(), content_type = 'text/plain')
    end = time.time()
    time_taken = end - start
one_mb_time_list = [time_taken]
one_mb_time_list_s =  [1/x for x in one_mb_time_list]
print 'MBps for 1Mb: %s' % one_mb_time_list_s
ten_mb_time_list = []

#10 mb object is put in container 1 time.
filepath10mb = '/home/felix//10mb.dat'
with open(filepath10mb) as file:
    start = time.time()
    object_id = conn.put_object(bucket_name, filepath10mb, contents = file.read(), content_type = 'text/plain')
    end = time.time()
    time_taken = end - start
ten_mb_time_list = [time_taken]
ten_mb_time_list_s =  [10/x for x in ten_mb_time_list]
print 'MBps for 10Mb: %s' % ten_mb_time_list_s

#10 mb object is put in container 1 time.
hundred_mb_time_list = []
filepath100mb = '/home/felix//100mb.dat'
with open(filepath100mb) as file:
    start = time.time()
    object_id = conn.put_object(bucket_name, filepath100mb, contents = file.read(), content_type = 'text/plain')
    end = time.time()
    time_taken = end - start
hundred_mb_time_list = [time_taken]
hundred_mb_time_list_s =  [100/x for x in hundred_mb_time_list]
print 'MBps for 100Mb: %s' % hundred_mb_time_list_s

#All times of above objects is plotted to show differences in throughput.
list = [ten_kb_time_list_s, hundred_kb_time_list_s, one_mb_time_list_s, ten_mb_time_list_s, hundred_mb_time_list
_s]
object_axis = [0,1,2,3,4]
plt.ylabel('MBps')
plt.xlabel('Object size')
plt.title('Throughput depending on object size')
```

```
my_xticks = ['10kb', '100kb', '1Mb', '10Mb', '100Mb']
plt.xticks(object_axis, my_xticks)
plt.plot(list)
```

```
Populating the interactive namespace from numpy and matplotlib
MBps for 10kb: [0.13444220561294312]
MBps for 100kb: [0.66377285421184717]
MBps for 1Mb: [0.6498135368627808]
MBps for 10Mb: [0.7901376269497503]
MBps for 100Mb: [0.8166958446489628]
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0xb0ca48cc>]
```



Excercise 2:

In the cell below, we obtain a client connection to the Nova endpoint. It can be used to for example start, stop and terminate instances.

In [5]:

```
import novaclient.client
config = {'username':os.environ['OS_USERNAME'],
          'api_key':os.environ['OS_PASSWORD'],
          'project_id':os.environ['OS_TENANT_NAME'],
          'auth_url':os.environ['OS_AUTH_URL'],
           }
from novaclient.client import Client
nc = Client('2',**config)
```

Excercise 3:

Boot a new instance (hint, look client.server in the API docs) with flavor 'm1.medium' (remember to provide an ssh-key so that you can access it later). In booting the instance, use the mechanism of 'user_data' (learn about this in the openstack and 'cloud-init' documentations) to provide a startup-script to 1. Update the instance, 2. install 'git', 'cowsay' and 'flask'.

In [6]:

```
# Use paramiko to access your instance and, using ssh, start the cowsay service on your instance,
# usinf the same command as in Task 4, lab 1.

import time
#Wanted flavors, images, keys etc is set in when instance is created.
userdatafp = open('/home/felix/Cloud/userdata.yml', 'r')
image = nc.images.find(name='Ubuntu Server 14.04 LTS (Trusty Tahr)')
flavor = nc.flavors.find(name='m1.medium')
key_pair = nc.keypairs.find(name='felixkey')
network = nc.networks.find(label='ACC-Course-net')
server = nc.servers.create(name ='Doktorns_instans', flavor = flavor.id, image = image.id, key_name = key_pair.n
ame, userdata = userdatafp, network = network.id)
nc.servers.list()

#Floating IP must be assigned when instance is actually booted, so if it isnt we wait.
status = server.status
while status == 'BUILD':
 time.sleep(5)
 server = nc.servers.get(server.id)
 status = server.status
floating_ip = nc.floating_ips.create(nc.floating_ip_pools.list()[0].name)
server.add_floating_ip(floating_ip)
my_floating_ip = floating_ip.ip
instance_info = next((x for x in nc.floating_ips.list() if x.ip == my_floating_ip), None)
my_fixed_ip=instance_info.ip
#Necessary pre-existing security group added for access to instance
server.add_security_group('safety_first')
```

In [7]:

```
#Here we use paramiko to provide ssh connection and ability to send execute commands on our instance.
#nc.servers.networks()
import paramiko
#cloud_key_fp = '/home/user/Cloud/cloud.key'
ssh = paramiko.SSHClient()
key = paramiko.RSAKey.from_private_key_file('/home/felix/Cloud/cloud.key')
ssh.load_system_host_keys()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(my_floating_ip, username='ubuntu', pkey = key)
```

In [8]:

```
import select

# Make a request to the cowsay REST API and display the response inline in the notebook
stdin, stdout, stderr = ssh.exec_command('cd csaas/cowsay/')
time.sleep(5)
stdin, stdout, stderr = ssh.exec_command('python csaas/cowsay/app.py &')
time.sleep(5)
stdin, stdout, stderr = ssh.exec_command('curl -i http://'+my_floating_ip+':5000/cowsay/api/v1.0/saysomething')

while not stdout.channel.exit_status_ready():
    # Only print data if there is data to read in the channel
    if stdout.channel.recv_ready():
        rl, wl, xl = select.select([stdout.channel], [], [], 0.0)
        if len(rl) > 0:
            # Print data from stdout
            print stdout.channel.recv(1024)
#ssh.close()
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 175
Server: Werkzeug/0.10.4 Python/2.7.6
Date: Thu, 01 Oct 2015 17:28:54 GMT


 _______________
< Hello student >
 ---------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Question 2:

The above excersise showed a low-level way of 'contextualization' using user data 'cloud-init'. Do some reserach online and discuss alternative tools and techniques for contextualization of your VMs. Discuss the difference between instance meta-data and user-data. Some links to get you started:

http://docs.openstack.org/user-guide/cli_provide_user_data_to_instances.html http://cernvm.cern.ch/portal/contextualisation https://cloudinit.readthedocs.org/en/latest/

Aim for the equivalent of ~1/2 page of an A4 paper, 12pt font, 2cm margins.

Excersise 4:

Use the Swift and Nova APIs to terminate your instance, to delete all the objects from your bucket, and then finally to delete the container.

In [8]:

```
# Obtain a list of all the object names in your container.
object_list = []
for data in conn.get_container(bucket_name)[1]:
    print '{0}\t{1}\t{2}'.format(data['name'], data['bytes'], data['last_modified'])
    #object_list.append('{0}'.format(data['name']))
#print object_list
```

```
/home/felix//100kb.dat 102400 2015-10-01T20:27:45.000Z
/home/felix//100mb.dat 102400000 2015-10-01T20:30:01.000Z
/home/felix//10kb.dat 10240 2015-10-01T20:27:42.000Z
/home/felix//10mb.dat 10485760 2015-10-01T20:27:59.000Z
/home/felix//1mb.dat 1024000 2015-10-01T20:27:46.000Z
/home/felix/10MB 10485760 2015-10-01T13:57:27.000Z
Test 9 2015-10-01T20:27:04.000Z
```

In [9]:

```
# Clean up container in Swift
for data in conn.get_container(bucket_name)[1]:
    object_list.append('{0}'.format(data['name']))
for b in object_list:
    conn.delete_object(bucket_name,b)
for data in conn.get_container(bucket_name)[1]:
    print '{0}\t{1}\t{2}'.format(data['name'], data['bytes'], data['last_modified'])
conn.delete_container(bucket_name)
```

In [ ]:

```
# Terminate all your running instances
server.delete()
```

In [ ]:

```

```