Evaluation and Measurement file

Weilun Zhao; A20329942

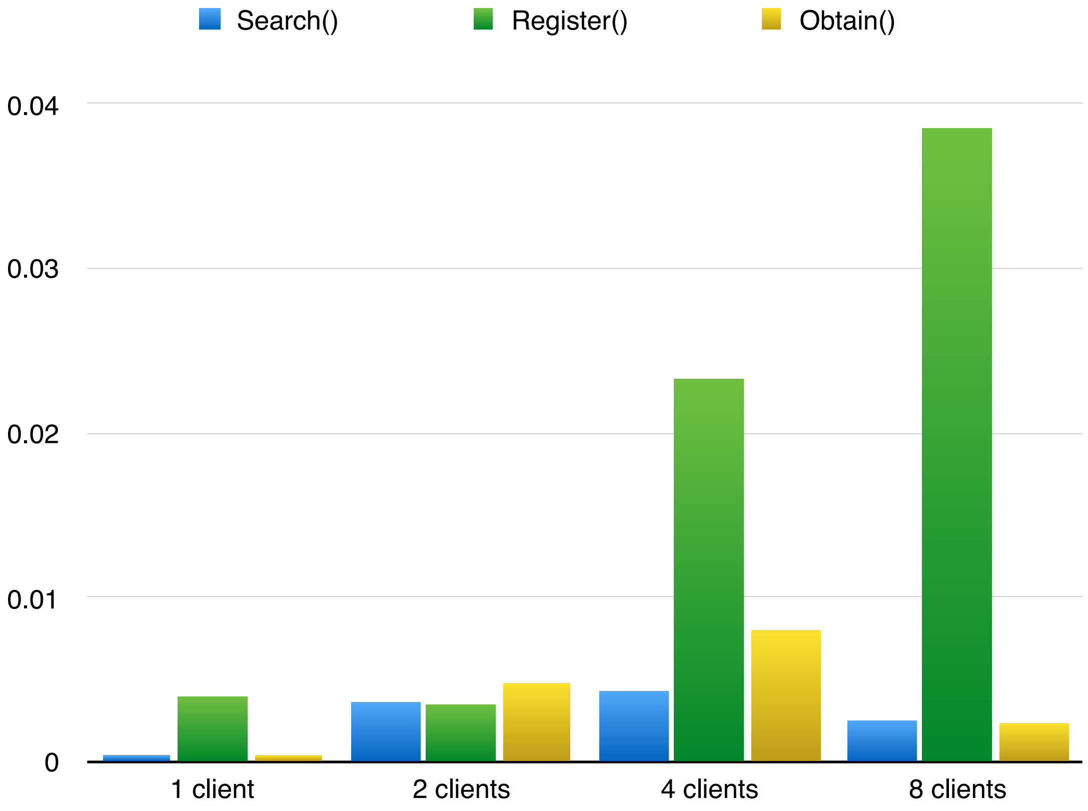
Performance evaluation:

1. Test: (the test is based on running 8 servers at the same time.

    a) register local files to DHT (passed)
    b) searching by file name in the DHT (passed)
    c) obtain verity size of files, for example 1K, 10K, 100K, 1MB, 10MB, 100MB (passed)
    d) obtain big file 4G (passed)
    e) input invalid or non-existed file name (passed)
    f) register no file (passed)

experiment 1:

In the experiment, there is a list about doing client(), search() and obtain() based on 1 client, two clients, and 8 clients executing 10K times. And the obtain() experiment, client just request one small files in the DHT. (base on the test code in the "test" folder)



1 client:
Start search file test, and execute : 10000 times
the average time of 10000 search : 3.553E-4

Start register testing, and execute : 10000s
fileName: A2.txt ;fileSize: 28
the average time of 10000 register : 0.003931

```
Test obtain :10000 times
the average time of 10000 obtain : 4.534E-4
```

2 Clients:

```
Thread 0: the average time of 10000  search : 0.0037051
Thread 1: the average time of 10000  search : 0.003705
Thread 1: the average time of 10000 register : 0.0035277
Thread 0: the average time of 10000 register : 0.0035481
Thread 1: the average time of 10000 obtain : 0.0047573
Thread 0: the average time of 10000 obtain : 0.0047387
```

4 Clients:

```
Thread 1: the average time of 10000  search : 0.0042546
Thread 3: the average time of 10000  search : 0.0042648
Thread 0: the average time of 10000  search : 0.0042833
Thread 2: the average time of 10000  search : 0.0042931
Thread 2: the average time of 10000 register : 0.0232689
Thread 0: the average time of 10000 register : 0.0234324
Thread 1: the average time of 10000 obtain : 0.0080521
Thread 3: the average time of 10000 obtain : 0.0080533
Thread 0: the average time of 10000 obtain : 5.049E-4
Thread 2: the average time of 10000 obtain : 0.0040793
```

8 Clients:(Total time: 45 seconds)

```
    [java] Thread 7: the average time of 1000  search : 0.002306
    [java] Thread 2: the average time of 1000  search : 0.002343
    [java] Thread 6: the average time of 1000  search : 0.002519
    [java] Thread 4: the average time of 1000  search : 0.002521
    [java] Thread 1: the average time of 1000  search : 0.002746
    [java] Thread 0: the average time of 1000  search : 0.002751
    [java] Thread 5: the average time of 1000  search : 0.002751
    [java] Thread 3: the average time of 1000  search : 0.002755
    [java] Thread 7: the average time of 1000 register : 0.03811
    [java] Thread 2: the average time of 1000 register : 0.038512
    [java] Thread 6: the average time of 1000 register : 0.039327
    [java] Thread 4: the average time of 1000 register : 0.039326
    [java] Thread 1: the average time of 1000 register : 0.039821
    [java] Thread 0: the average time of 1000 register : 0.039858
    [java] Thread 5: the average time of 1000 register : 0.039861
    [java] Thread 3: the average time of 1000 register : 0.03986
    [java] Thread 7: the average time of 1000 obtain : 0.002868
    [java] Thread 2: the average time of 1000 obtain : 0.00277
    [java] Thread 4: the average time of 1000 obtain : 0.002371
    [java] Thread 6: the average time of 1000 obtain : 0.002376
    [java] Thread 1: the average time of 1000 obtain : 0.001898
    [java] Thread 3: the average time of 1000 obtain : 0.001862
    [java] Thread 0: the average time of 1000 obtain : 0.00187
    [java] Thread 5: the average time of 1000 obtain : 0.001867
```

The part, test the throughput(RM/S) based on deploying 8 clients
run:(1 time)(test code in the “tester3” folder)

```
    [java] Thread 6: the average time of 1 obtain : 0.065
    [java] tN: 6 ; throughput: 15.384615384615383
    [java] Thread 3: the average time of 1 obtain : 0.075
    [java] tN: 3 ; throughput: 13.333333333333334
    [java] Thread 5: the average time of 1 obtain : 0.075
    [java] tN: 5 ; throughput: 13.333333333333334
    [java] Thread 0: the average time of 1 obtain : 0.079
    [java] tN: 0 ; throughput: 12.658227848101266
    [java] Thread 7: the average time of 1 obtain : 0.08
    [java] tN: 7 ; throughput: 12.5
    [java] Thread 2: the average time of 1 obtain : 0.084
    [java] tN: 2 ; throughput: 11.904761904761903
    [java] Thread 4: the average time of 1 obtain : 0.084
    [java] tN: 4 ; throughput: 11.904761904761903
    [java] Thread 1: the average time of 1 obtain : 0.105
    [java] tN: 1 ; throughput: 9.523809523809524
    [java] total throughput is 100.54284323271663
```

run:(0.1K)

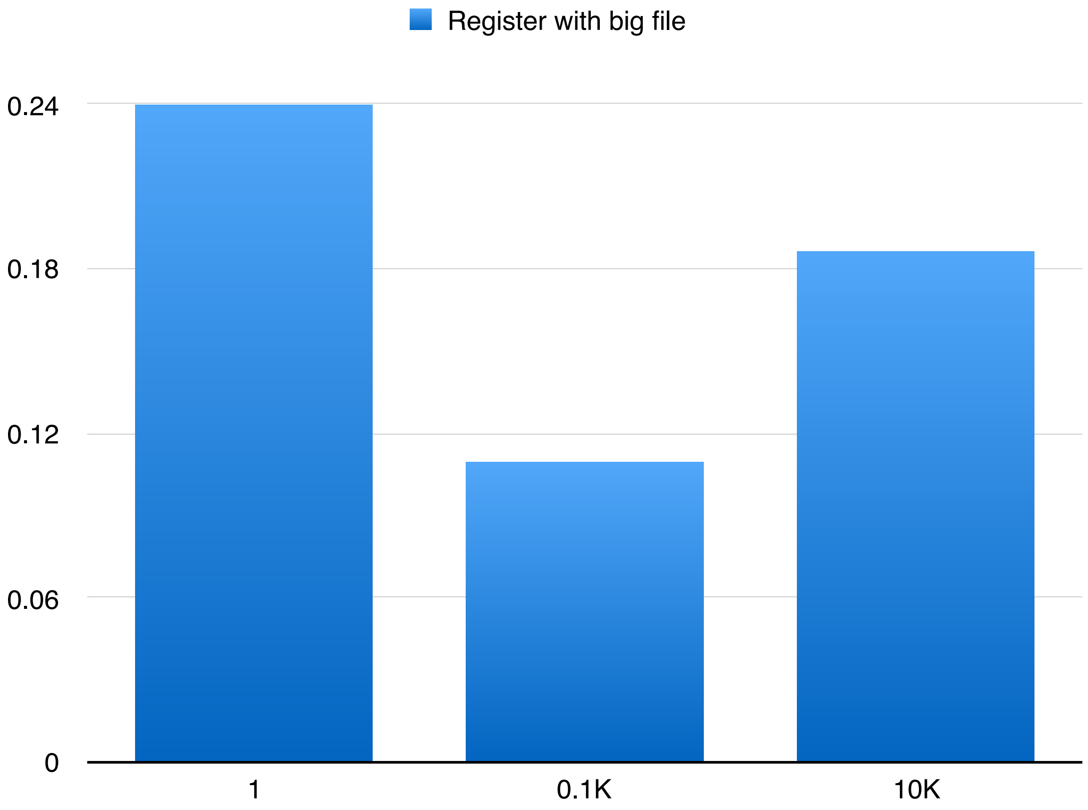
```
    [java] Thread 7: the average time of 100 obtain : 0.0458
    [java] tN: 7 ; throughput: 21.83406113537118
    [java] Thread 6: the average time of 100 obtain : 0.04595
    [java] tN: 6 ; throughput: 21.762785636561482
    [java] Thread 0: the average time of 100 obtain : 0.04596
    [java] tN: 0 ; throughput: 21.75805047867711
    [java] Thread 4: the average time of 100 obtain : 0.04654
    [java] tN: 4 ; throughput: 21.486892995272886
    [java] Thread 2: the average time of 100 obtain : 0.04659
    [java] tN: 2 ; throughput: 21.4638334406525
    [java] Thread 3: the average time of 100 obtain : 0.04662
    [java] tN: 3 ; throughput: 21.45002145002145
    [java] Thread 1: the average time of 100 obtain : 0.04691
    [java] tN: 1 ; throughput: 21.317416329140908
    [java] Thread 5: the average time of 100 obtain : 0.04696
    [java] tN: 5 ; throughput: 21.29471890971039
    [java] total average throughput is 172.36778037540788
```

run:(10K)
(the throughput is the last to execute obtain() and return the throughput result)
(Total time: 7 minutes 24 seconds)

```
    [java] Thread 5: the average time of 10000 obtain : 0.0442289
    [java] tN: 5 ; throughput: 22.60965115569232
    [java] Thread 1: the average time of 10000 obtain : 0.0443116
    [java] tN: 1 ; throughput: 22.567454120365774
    [java] Thread 0: the average time of 10000 obtain : 0.0443136
    [java] tN: 0 ; throughput: 22.56643558636626
    [java] Thread 3: the average time of 10000 obtain : 0.0443419
    [java] tN: 3 ; throughput: 22.552033178551213
    [java] Thread 2: the average time of 10000 obtain : 0.0443456
    [java] tN: 2 ; throughput: 22.55015153701833
    [java] Thread 6: the average time of 10000 obtain : 0.0443564
    [java] tN: 6 ; throughput: 22.544660973388282
    [java] Thread 4: the average time of 10000 obtain : 0.0443654
    [java] tN: 4 ; throughput: 22.54008754570003
    [java] Thread 7: the average time of 10000 obtain : 0.0443794
    [java] tN: 7 ; throughput: 22.532977011856854
    [java] total average throughput is 180.46345110893907
```

experiment 2:

In the experiment, tester client have files and register them to the DHT based on deploying 8 clients :
(test code in the “tester2” folder)
1). there are total 48 files. And each clients have 6 files register info:
8*(1KB, 10KB, 100KB, 1MB, 10MB, 100MB)



run:(once)

```
    [java] Thread 6: the average time of 1 register : 0.236
    [java] Thread 4: the average time of 1 register : 0.246
    [java] Thread 7: the average time of 1 register : 0.235
    [java] Thread 3: the average time of 1 register : 0.242
    [java] Thread 0: the average time of 1 register : 0.25
    [java] Thread 5: the average time of 1 register : 0.248
    [java] Thread 2: the average time of 1 register : 0.25
    [java] Thread 1: the average time of 1 register : 0.251
```

```
run:(0.1K)
    [java] Thread 1: the average time of 100 register : 0.08461
    [java] Thread 3: the average time of 100 register : 0.08463
    [java] Thread 0: the average time of 100 register : 0.11451
    [java] Thread 4: the average time of 100 register : 0.11513
    [java] Thread 2: the average time of 100 register : 0.11528
    [java] Thread 5: the average time of 100 register : 0.11529
    [java] Thread 6: the average time of 100 register : 0.12397
    [java] Thread 7: the average time of 100 register : 0.49834
```

```
run: (In the testing, the 4G big file be registered into the DHT)
    [java] Thread 5: the average time of 10000 register : 0.1863036
    [java] Thread 1: the average time of 10000 register : 0.1863077
    [java] Thread 6: the average time of 10000 register : 0.1863115
    [java] Thread 3: the average time of 10000 register : 0.1863146
    [java] Thread 7: the average time of 10000 register : 0.1863163
    [java] Thread 2: the average time of 10000 register : 0.1863169
    [java] Thread 0: the average time of 10000 register : 0.1863179
    [java] Thread 4: the average time of 10000 register : 0.1863205
```

The Comparison With Assignment 1

In my assignment 1, there are 10 client threads executing execute 1000 times search(), and the average time of search is 0.001861s after deploy 3 clients and central server.

In the experiment 1, I deployed 8 clients to get the average time of search(), register() and obtain() time. And the average time of 1000 times search() is :

```
    [java] Thread 7: the average time of 1000  search : 0.002306
    [java] Thread 2: the average time of 1000  search : 0.002343
    [java] Thread 6: the average time of 1000  search : 0.002519
    [java] Thread 4: the average time of 1000  search : 0.002521
    [java] Thread 1: the average time of 1000  search : 0.002746
    [java] Thread 0: the average time of 1000  search : 0.002751
    [java] Thread 5: the average time of 1000  search : 0.002751
    [java] Thread 3: the average time of 1000  search : 0.002755
```

The 8 client average of search is 0.0025865s which is slower than assignment 1.

I believe the average time of operation will increase with the increasing of clients number or operation times, so the distribute server will take more time to handle request. I suppose that clients sending and handling requests concurrently in the DHT can make each operation costs more time. The throughput, however, will increase with more distribute servers, thus the load and execution ability will be increased in this situation. As a result, DHT design try to optimization the balance between the execution time and the throughput.

Optional: Do the evaluation on the EC2 (test code is similar with Experiment 1 test code)

```
Thread 3: the average time of 1000  search : 0.002627
Thread 1: the average time of 1000  search : 0.002624
Thread 7: the average time of 1000  search : 0.002624
Thread 2: the average time of 1000  search : 0.002644
Thread 4: the average time of 1000  search : 0.002642
Thread 6: the average time of 1000  search : 0.00264
Thread 5: the average time of 1000  search : 0.002662
Thread 0: the average time of 1000  search : 0.003206
Thread 7: the average time of 1000 register : 0.004754
Thread 6: the average time of 1000 register : 0.00477
Thread 5: the average time of 1000 register : 0.004777
Thread 3: the average time of 1000 register : 0.004851
Thread 1: the average time of 1000 register : 0.004856
Thread 4: the average time of 1000 register : 0.004864
Thread 2: the average time of 1000 register : 0.004888
Thread 0: the average time of 1000 register : 0.004455
Thread 7: the average time of 1000 obtain : 0.001491
Thread 6: the average time of 1000 obtain : 0.001506
Thread 5: the average time of 1000 obtain : 0.001497
Thread 3: the average time of 1000 obtain : 0.001491
Thread 1: the average time of 1000 obtain : 0.001473
Thread 4: the average time of 1000 obtain : 0.001469
Thread 2: the average time of 1000 obtain : 0.001456
Thread 0: the average time of 1000 obtain : 0.001709
```