Evaluation and Measurement for Programming Assignment #3

Test 1

We are making 10K request from each peer to server for search and download operation for multiple files having file size of 1KB

Evaluation done on Single System

Central Index Server	Search	Download
Peer 1	7s 136ms	17s 739ms

Decentralised Index Server	Search	Download
Peer 1	6s 930ms	16s 865ms

Central Index Server	Search	Download
Peer 1	8s 211ms	16s 858ms
Peer 2	8s 639ms	14s 167ms
Average	8s 425ms	15s 512ms

Decentralised Index Server	Search	Download
Peer 1	8s 13ms	17s 430ms
Peer 2	11s 811ms	16s 187ms
Average	9s 912ms	16s 809ms

Central Index Server	Search	Download
Peer 1	12s 483ms	16s 845ms
Peer 2	13s 309ms	18s 318ms
Peer 3	13s 409ms	16s 346ms
Peer 4	10s 469ms	16s 691ms
Average	12s 418ms	17s 50ms

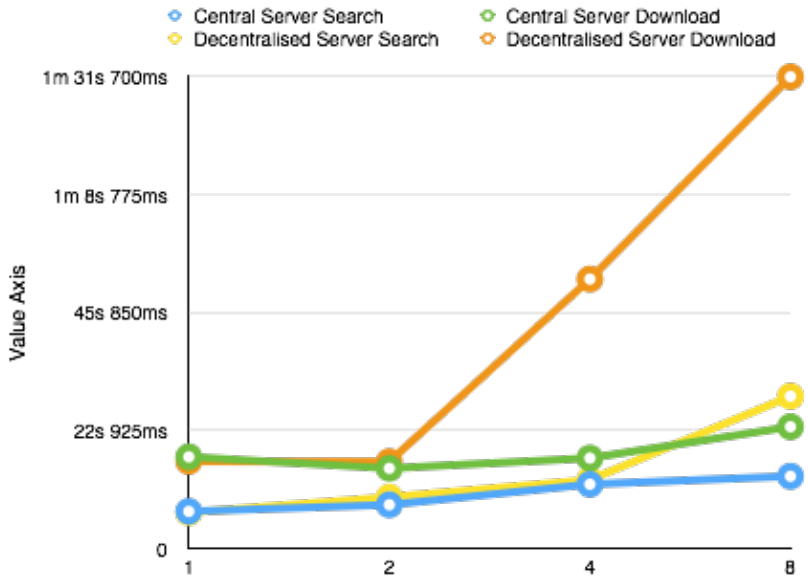
Decentralised Index Server	Search	Download
Peer 1	16s 106ms	48s 271ms
Peer 2	14s 83ms	51s 513ms
Peer 3	7s 834ms	55s 899ms
Peer 4	15s 58ms	53s 585ms
Average	13s 270ms	52s 317ms

Central Index Server	Search	Download
Peer 1	17s 816ms	18s 794ms
Peer 2	10s 914ms	22s 855ms
Peer 3	9s 413ms	19s 586ms
Peer 4	12s 178ms	28s 121ms
Peer 5	14s 915ms	24s 561ms
Peer 6	18s 380ms	25s 303ms
Peer 7	9s 954ms	24s 455ms
Peer 8	18s 195ms	25s 89ms
Average	13s 971ms	23s 596ms

Decentralised Index Server	Search	Download
Peer 1	21s 450ms	1m 29s 525ms
Peer 2	22s 840ms	1m 28s 987ms
Peer 3	26s 855ms	1m 28s 349ms
Peer 4	36s 76ms	1m 48s 550ms
Peer 5	37s 93ms	1m 30s 602ms
Peer 6	31s 582ms	1m 27s 874ms
Peer 7	30s 972ms	1m 32s 772ms
Peer 8	29s 392ms	1m 26s 555ms
Average	29s 533ms	1m 31s 652ms

Average response time plot

No Of peers	Central Server Search	Central Server Download	Decentralised Server Search	Decentralised Server Download
1	7s 136ms	17s 739ms	6s 930ms	16s 865ms
2	8s 425ms	15s 512ms	9s 912ms	16s 809ms
4	12s 418ms	17s 500ms	13s 270ms	52s 317ms
8	13s 971ms	23s 596ms	29s 533ms	1m 31s 652ms



Evaluation on 16 instances of Amazon EC2

Let's have look at the result when we run the same above set of experiment on different machine or different instances of Amazon Ec2 micro Amazon Linux instances. Average of ping request of 64 bytes of data takes `0.686 ms.` We have made 10K request from each peer and response time from each peer is given in sec.

Central Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	800	1734
Peer 2	780	1834
Average	790	1784

Decentralised Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	810	1926
Peer 2	806	1834
Average	808	1880

Central Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	845	1856
Peer 2	831	1790
Peer 3	901	1901
Peer 4	823	1843
Average	850	1847.5

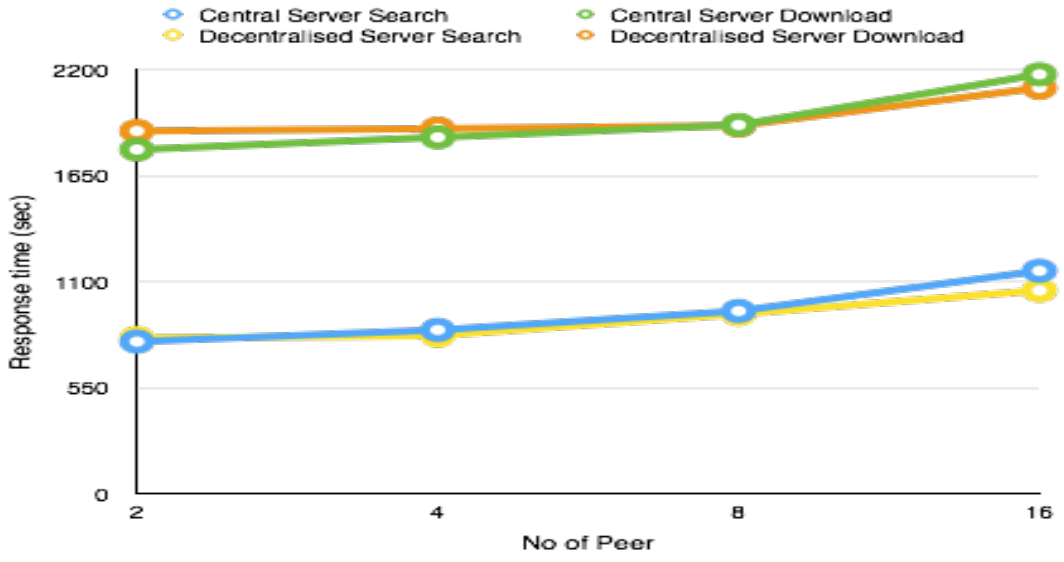
Decentralised Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	820	1854
Peer 2	826	1942
Peer 3	819	1936
Peer 4	819	1837
Average	821	1892.25

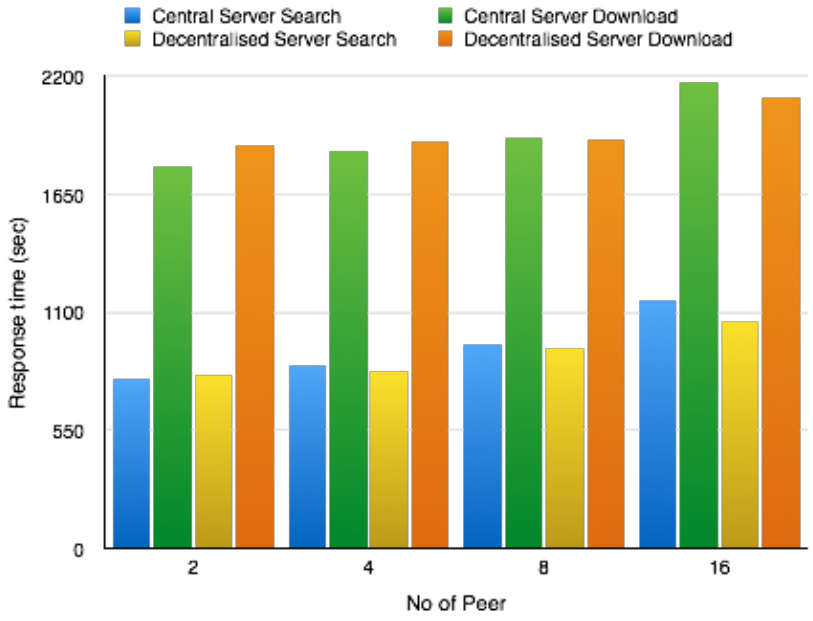
Central Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	948	1936
Peer 2	986	1944
Peer 3	1031	1851
Peer 4	945	1926
Peer 5	1042	1953
Peer 6	869	1982
Peer 7	841	1826
Peer 8	922	1871
Average	948	1911.125

Decentralised Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	923	1893
Peer 2	945	1864
Peer 3	923	1957
Peer 4	926	1958
Peer 5	938	1938
Peer 6	952	1882
Peer 7	922	1876
Peer 8	931	1894
Average	932.5	1907.75

Central Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	1053	2093
Peer 2	1021	2148
Peer 3	1019	2218
Peer 4	1194	2162
Peer 5	1205	2199
Peer 6	1191	2264
Peer 7	1257	2143
Peer 8	1293	2283
Peer 9	1104	2181
Peer10	1172	2172
Peer 11	1209	2194
Peer 12	1181	2135
Peer 13	1284	2187
Peer 14	1099	2092
Peer 15	1075	2198
Peer 16	1149	2105
Average	1156.625	2173.375

Decentralised Index Server	Search Response Time (sec)	Download Response Time (sec)
Peer 1	1001	1963
Peer 2	1032	2004
Peer 3	1082	2119
Peer 4	1043	2004
Peer 5	1093	2109
Peer 6	1057	2105
Peer 7	1001	2147
Peer 8	1093	2208
Peer 9	1051	2109
Peer10	1066	2163
Peer 11	1104	2148
Peer 12	1039	2192
Peer 13	1082	2115
Peer 14	1047	2049
Peer 15	1043	2127
Peer 16	1045	2093
Average	1054.9375	2103.4375





Average response time plot

No Of peers	Central Server Search	Central Server Download	Decentralised Server Search	Decentralised Server Download
2	790	1784	808	1880
4	850	1847.5	821	1892.25
8	948	1911.25	932.5	1907.75
16	1156.265	2173.375	1054.9375	2103.4375

Observation

While running the test on single machine we can simply see that the Centralized server outperforms the Decentralized server. Average Response time of search and Download on 1,2,4,8 nodes is less for the Centralized server than Decentralized server though Decentralized server greater flexibility and scalability. But on the other hand when the same test is done different instances of Amazon Cloud we can see that as no of peer increased. Average Response time of search and Download is less for Decentralized server than Central Index Server. So the Question is why we get different result.

We have two cases now

Case1: On single Machine: Central index server was performing better than decentralized index server because On single machine we had only one server running and 8 peer running So in total there was 9 process running. But in case of Decentralized Server experiment there were 8 server running and 8 peer running, So in total there was 16 process running on same machine. As a result of which average response time that we got on single machine is very hard to prove. But we have can say that Decentralized is scalable .

Case 2: On Different machine : Central Index server experiment and Decentralized server experiment both were performed under on same kind of machine and network. Only difference now is that we all the server and peer running on different machines. So the load factor that we were getting on single machine is now gone. As a result of which we got the result what we were looking for. As peer count increased Central index server got a lot of parallel request causing single server to perform lots of work that has cost the response time, where as on decentralized

each server was not getting lots of request compared to centralized server. So as peer get increasing the response time of Decentralized is better than Centralized.

At last we can say that when the number of peer increases more than 8 Decentralized system worked much better than centralized system. Also Decentralized have added advantage of scalability.

Test 2

In this experiment every peer have multiple files of size ranging from 1KB, 10KB, 100KB, 1MB, 10MB, 100MB, and 1GB in size. Each peer perform search and download operation for the files of other peer.

File Size	No of Files	Bytes
1 KB	5000	5000000
10 KB	1000	10000000
1 MB	50	50000000
10 MB	10	100000000
100 MB	5	500000000
1 GB	1	1000000000
Total	6066	1665000000 =1.665 GB

For running this kind of test we need to increase the size of java VM. This can be done by setting property in ".bashrc" file or when running jar file through command prompt.

Java –Xmx3g –jar dist/DecentralizedIndeerver.jar

Java –Xmx3g –jar dist/Assignment3Peer.jar

My single system doesn't have that much memory. So I have conducted this experiment on multiple machines.

As search for this experiment is done very less time compare to download. So we can concentrate more download time.

Average of search operation for all clients took 21.914 sec

Output Table

Peer Id	Download Response time in sec
Peer 1	371.226
Peer 2	385.751
Peer 3	380.469
Peer 4	401.247
Peer 5	389.450
Peer 6	380.596
Peer 7	384.938
Peer 8	380.344
Average time	384.811

Download Response for centralized system is approximately same as decentralized server with few seconds of minor difference because Index server is not taking part in download operation of file except searching of file. Files are downloaded between peers for this reason the result of donwload between central index server and decentralized system are approximately same. However now we need to find the transfer rate and need to check does our result make sense.

We have transferred 1665 MB of data from each server with average response time of 384.811 sec so our transfer speed is 4.32 MB/sec

Network latency is 0.437 ms for 64 bytes

So Is 4.32 MB per sec is a good speed?

So I performed other experiment, On TCP protocol we can transfer data more than 1000MB per sec. So I have written a very simple code to test this on single machine.

```
import java.io.InputStream;
import java.net.Socket;


public class SimpleClient {
   public static void main(String[] args) throws Exception {
      Socket socket = new Socket("127.0.0.1", 6666);
      InputStream input = socket.getInputStream();
      long total = 0;
      long start = System.currentTimeMillis();

      byte[] bytes = new byte[32*1024]; // 32K
      for(int i=1;;i++) {
         int read = input.read(bytes);
         if (read < 0) break;
         total += read;
         if (i % 500000 == 0) {
            long cost = System.currentTimeMillis() - start;
            System.out.printf("Read %,d bytes, speed: %,d MB/s%n", total, total/cost/1000);
         }
      }
   }
}
```

```
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;


public class SimpleServer {
   public static void main(String[] args) throws Exception {
      ServerSocket server = new ServerSocket(6666);
      Socket socket = server.accept();
      OutputStream output = socket.getOutputStream();

      byte[] bytes = new byte[32*1024]; // 32K
      while (true) {
         output.write(bytes);
      }
   }
}
```

Response time for above code is

```
Read 12,732,530,688 bytes, speed: 1,262 MB/s
Read 25,169,002,392 bytes, speed: 1,253 MB/s
Read 37,715,050,496 bytes, speed: 1,219 MB/s
Read 50,057,543,680 bytes, speed: 1,217 MB/s
Read 62,219,517,952 bytes, speed: 1,201 MB/s
```

After carefully understanding of both scenarios I found that both cases are different. In our System Firstly we are running on different machine where as this code is run on single machine. So I checked the disk writing speed of my machine using software BlackMagic speed test result was 140MB per sec. Then how come transfer rate of simple code is so high and the reason is simple we are not persisting any thing. It is just a transferring bytes chunk that's it. On our System we are transferring file. So we need to consider factor of our network then writing speed of different machines, as we are persisting data. Another factor we also need to consider is no of files because we all know that copying a single large file takes less time than copying large number of small multiple files that will sum to size of single large file. In our system we are transferring 6066 files of different sizes.

So considering all these factor in mind, I believe we are getting correct output.