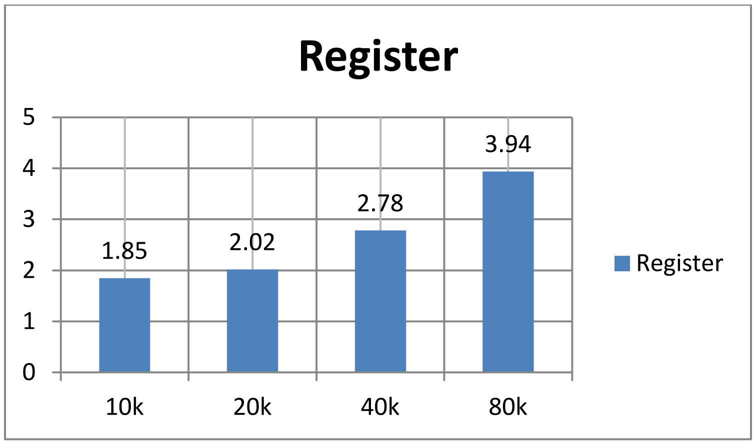
Performance Evaluation

Part-I

Register Operations(Without Replication of Meta-Data)

This is using a Optimized Code .

Sr. no	Total Servers	Total Clients(running concurrently)	Avg. Total Time(Secs)	Avg .Time per Operation(msec)
1	8	1(10K)	1.85	0.185
2	8	2(20K)	2.02	0.202
3	8	4(40K)	2.78	0.278
4	8	8(80K)	3.94	0.394



Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

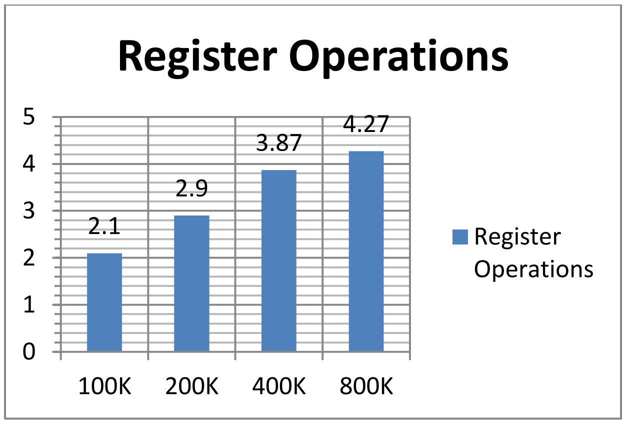
Reasons : The time appears to hardly changing even when the total number of clients have been increased because:

- There is a separate code of the De-centralized Server and a separate Peer,
- **Optimized Code**:The File List is first created depending on the hash value and the sent to the decentralized server.So for every operation the files are grouped first depending on the hash value and then sent to the De-centralized Server.(similary to Batch Processing)

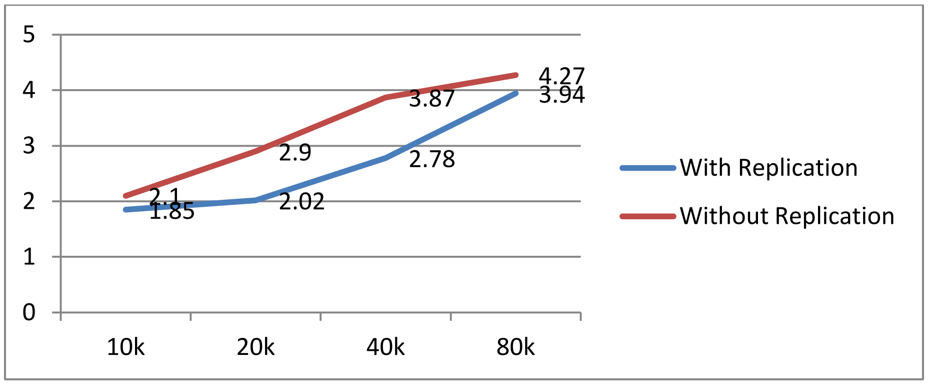
Register Operations(With Replication of Meta-Data)

This is using a Optimized Code .

Sr. no	Total Servers	Total Clients(running concurrently)	Avg. Total Time(Secs)	Avg. Time per Operation(msec)
1	8	1(10K)	2.10	0.210
2	8	2(20K)	2.9	0.290
3	8	4(40K)	3.87	0.387
4	8	8(80K)	4.27	0.427



Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

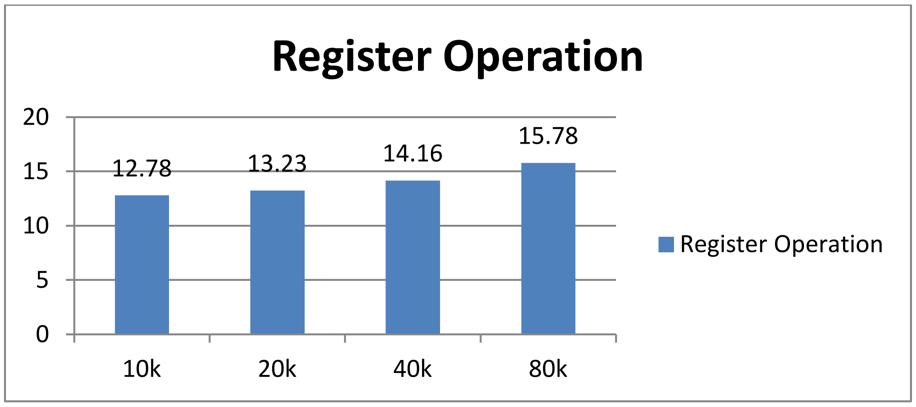


Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

Register Operation(10k) Without Optimization, Without Replication

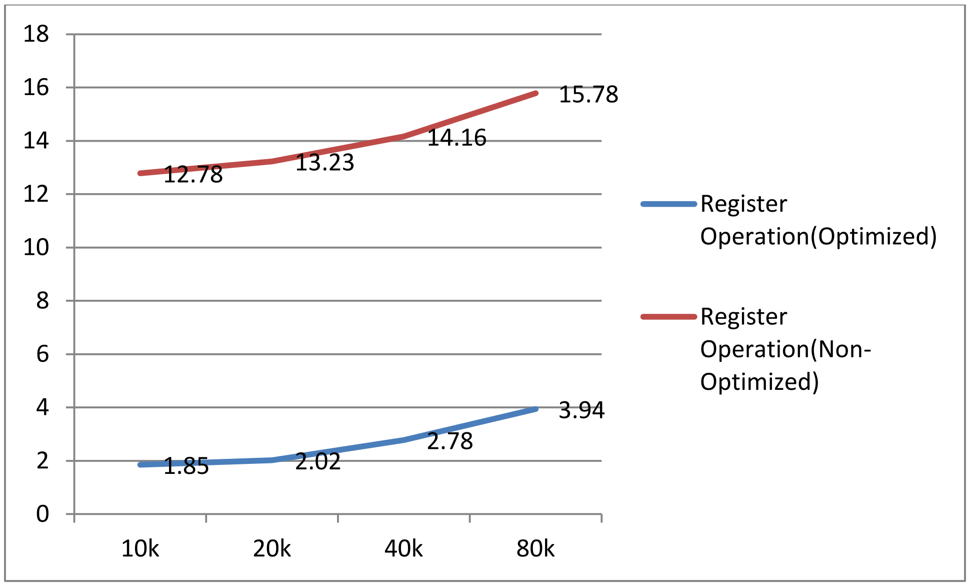
In this case the file are sent one by one to the DHT no grouping is done before sending the files

Sr. no	Total Servers	Total Clients(running concurrently)	Avg. Total Time(Secs)	Avg. Time per Operation(msec)
1	8	1(10K)	12.78	1.278
2	8	2(20K)	13.23	1.323
3	8	4(40K)	14.16	1.416
4	8	8(80K)	15.78	1.578



Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

In this scenario the file names are read from the folder and sent one-by-one to the DHT of the specific hash value. So for N files N messages are used. Whereas in case of an optimized code for N files having K Servers only K messages are used and sent.

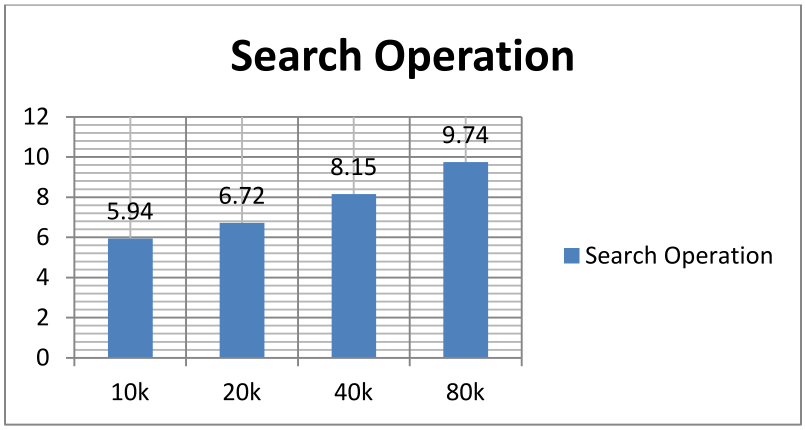


Vertical Axis: Avg Total Time in seconds **horizontal Axis:** Total Operations(Concurrent)

So from the graph it can be concluded that the Registration performance improves when the files are NOT send individually but are grouped and send. So hence the code uses the Optimized Performance .The Non-Optimized Performance is used a reference for comparing its pros and cons to the Optimized Version.

Search Operations(10K)

Sr. no	Total Servers	Total Clients(running concurrently)	Avg. Total Time(Secs)	Avg. Time per Operation(msec)
1	8	1(10K)	5.94	0.594
2	8	2(20K)	6.72	0.672
3	8	4(40K)	8.15	0.815
4	8	8(80K)	9.74	0.974



Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

Result Verification(Ping Experiment):In this case a simple search request for 100 files was made to 4 DHT Servers by the 1 client in the same test environment the time taken was 0.16secs (total).

Every Peer searches for a specific file using the de-centralized hash table of the server

File Download (10K)

Sr. no	Total Servers	Total Clients(running concurrently)	Total Size of 10K files in each folder	Avg. Total Time(Secs)	Avg. Time per Operation(msec)	Time to transfer 1MB(secs)
1	8	1(10K)	40MB	312.23	31.23	7.8
2	8	2(20K)	40MB	318.11	31.81	7.9
3	8	4(40K)	40MB	323.28	32.23	8.08
4	8	8(80K)	40MB	334.41	33.44	8.32

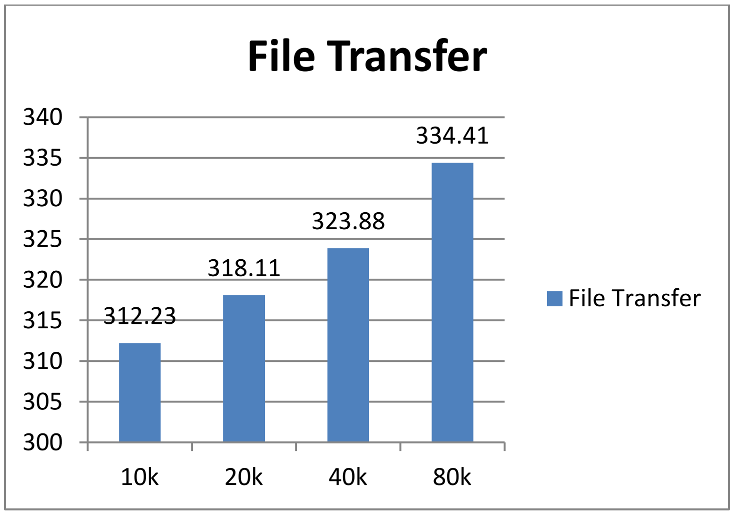
File Size from 1KB-5MB in the folder.

In File Transfer the 10K files are transferred from one Peer to another.

Results from the Ping Experiment: In the Ping experiment a simple 1MB file is transferred over the network, but in the same test environment but between 2 Clients.Here only File Transfer was done,no additional messages, parameters were passed or computed.

Avg Time for the Operation: 1MB file the time is 4.73 secs

Vertical Axis: Time in seconds horizontal Axis: Total Operations(Concurrent)



Timing Difference of Search and Register(optimized Code)

General Scenario:

Time of Search < Time to Register(Non-Optimized Code)

Actual Scenario:

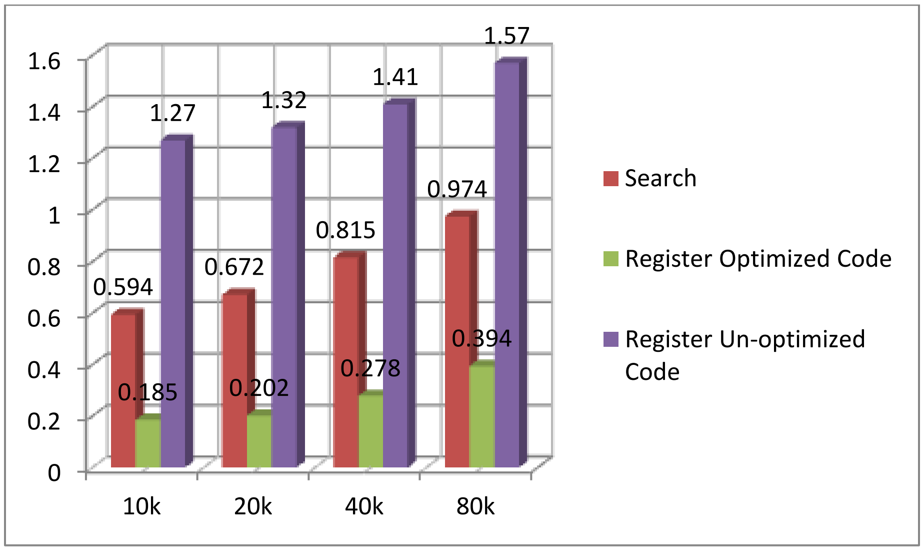
Time of Search > Time to Register(Optimized Code)

Reason:

- When the files register from a folder the files are sorted and grouped upon the hash value and then depending on the hash value the entire group of files is sent to server as a specific formatted string.(for n files 1 message between client and server) **using Concept like Batch Processing**

- In case of search a specific file is only sent to server for searching in the de-centralized hash table.(for n files n messages between client and server)

Vertical Axis: Time per operation in msecs   horizontal Axis: Total Operations(Concurrent)



Part-II

Seq	File Size	Total Files	Total Data Transferred by each peer	Total Peers	Avg Total Time in Secs	Avg Time per Single File	Transfer Speed	Bytes/sec
F	1GB	1	1GB	8	372.12	372.12 secs	2.68mbps	2680000
E	100MB	5	500MB	8	133.34	26.68 secs	3.74 mbps	3750000
D	10MB	100	1000MB	8	248.67	2.487 secs	4.09mbps	4090000
C	100KB	1K	10MB	8	47.67	0.04767 secs	210.12Kbps	210120
B	10KB	7K	70MB	8	197.86	0.028 secs	354.60 kbps	354600
A	1KB	10K	10MB	8	121.02	0.0137 secs	82.50kbps	82500

What does the Avg total File transfer time include?

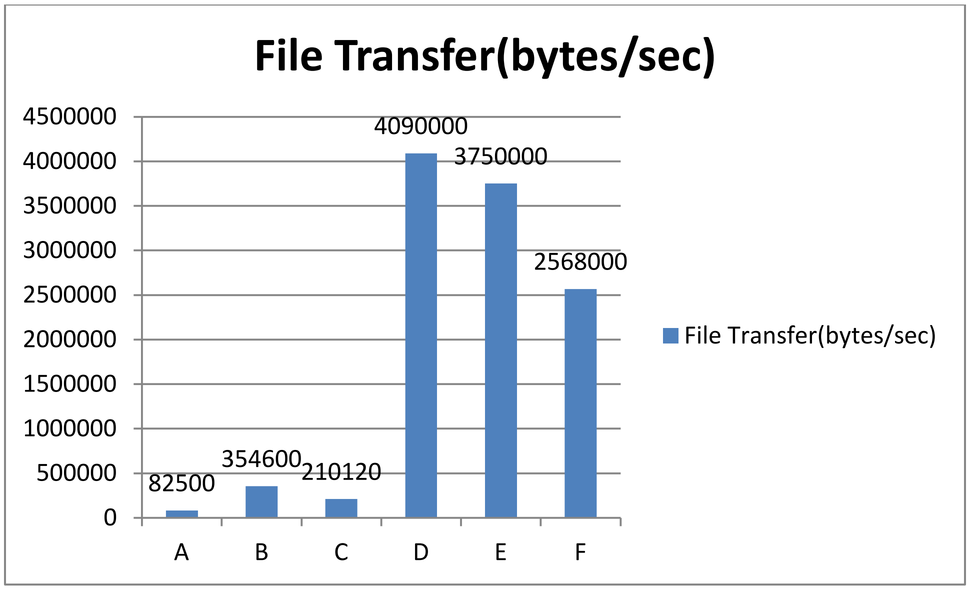
Establishing Socket Connection + Sending File Name across Network+ Transferring the File Block

Data Analysis

- It can be concluded that as the number of files increase ,with sizes the bytes/sec is independent of the total number of files it depends on the file size.An increase in file size (Data-content) leads to an higher bytes/sec file.As the data has to be continuously be read and send on the network.
- For eg:to transfer 1KB(10K) files the bytes/sec for 1 file is 82500 and for 100KB(1K) files the time went upto 4090000 and is still higher for 1GB files

  THE JVM OVER-FLOW ISSUE

  The JVM over flow issue occurs when the heap-size goes above 1GB this is the typical case for files having huge amount of data so in this case a buffer reader having 4096 bytes has been used.



Vertical Axis: Time of operation in byte/secs

horizontal Axis:A->10K files of 1KB,B->7K files of 10Kb,C->1K files of 100Kb,D->100,10MB files

E->5 ,100MB files ,F->1,1GB file

The operation takes very low time for small files but increases continuously as size increases.

Part-III

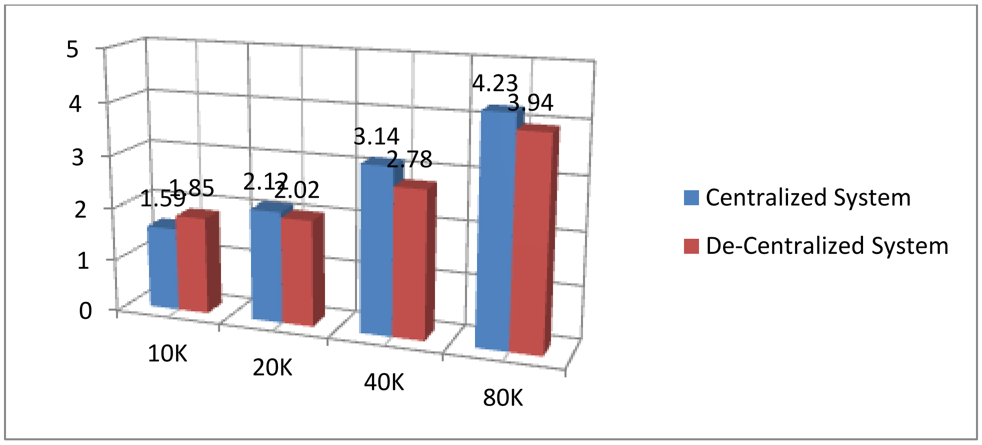
Result Comparison with Centralized Server

Number of Centralized Server:1,Total Clients :1-8,Operations:10K-80K

Number of Decentralized Server:8 ,Total Clients: 1-8,Operations:10k-80K

Register Operations

Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)

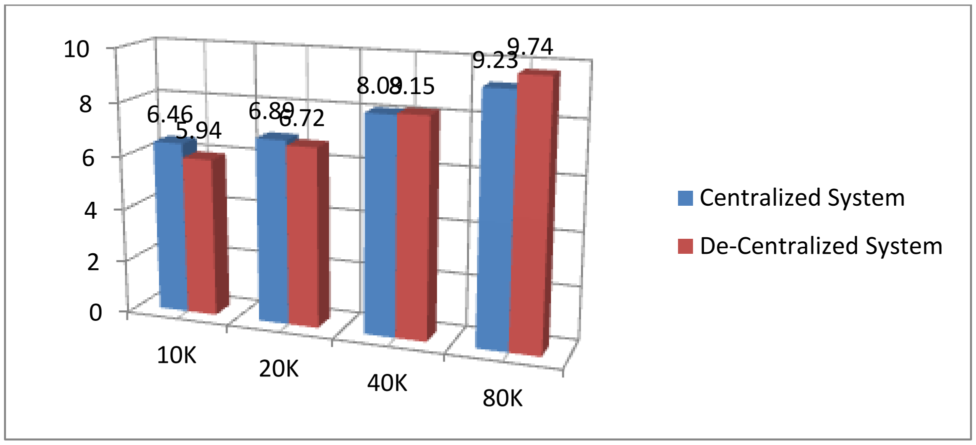


Now I have a Centralized System and a Decentralized System perform a same operation but the avg total time differences are minimal because:

- In De-centralized System my 10K or 100K files are distributed evenly across **N servers**
- In Centralized System my 10K or 100K files are broken down in N slots where each slot has 1000 files and each N of this slots is sent to a **Single Server.**
- So from this we can conclude if the no of files in increased to 1M or more and peers are increased M times by keeping the number of servers constant De-centralized System gives a better performance with a scalable systems under time T.

Search Operations:

Vertical Axis: Time in seconds   horizontal Axis: Total Operations(Concurrent)



File Transfer Operations

In any of the cases are same as they are between Peer-Peer and not Peer-Server or Peer-DHT

Performance Evaluation Method

Testing Environment

- **Windows 8.1**
- **3GB RAM**
- **Linux14.01**

- For Register, Search ,Download(10K-100K) any operation 8 servers were started up initially on different IPs and Ports
- A script was fired which generated 10K files automatically of different sizes(1MB-1GB) in different folders with different names.
- Then the peers were started a timestamp was used to start them at the same clock time and the search, register and download operations were performed.
- **Note:**
  - The time for creating file logs has been not considered as logs were not a part of performance evaluation.
  - Any background noise has been ignored.

Comparing with Centralized Server

- For Centralized Server only 1 Server was used, as in decentralized 8 were used.
- The other operations and number of peers and assumptions remain same as stated above.

Conclusion

- Centralized v De-Centralized
    - The Performance is almost the same with very minor difference.
    - The Performance of centralized will go down when Peers increased to 32 or more.
    - Though both use optimal code DeCentralized will be useful for Scalability as its has distributed servers.
- File Downloading
    - The performance depends on the size of the file higher the sizes more are the chances of the performance going down.
    - The system will support good file transfer rates upto a certain point,depending on the file size and resource availability.

- Register
    - It uses optimal functionality as explained earlier by which minimum sockets are created thereby improving the performance as less time is required to make and break connections.
- Search
    - For search the peer makes a lookup in the DHT and retrives the value. The performance increases gradually as number of operations and peers are increased simultaneously.