- Test script : **test.sh**
- Test log files are placed in **test_log** folder in the source code
- **time** command of Linux is used to calculate the performance
- Buffer size is set to a maximum of **4096** bytes for every read / write
- Every access is repeated **200** times
- For multiple clients case, we ran tests with **2 or 3** clients accessing simultaneously.

Test Case 1 : Relay Test Case

compute-0-0 <-----> compute-0-1 <-----> compute-0-2 <------> compute-0-3

Here we try to search a file from compute-0-0 and the file is served by compute-0-3. This is to test that the request are relayed properly.

Relay			
File Size	Requestor	Provider	Time
1k	compute-0-0	compute-0-3	34.53
2k	compute-0-0	compute-0-1, 0-2	0.91
4k	compute-0-0	compute-0-1	9.03
8k	compute-0-0	compute-0-3	17.44
Multiple requests			
8k	Compute-0-0, 0-1, 0-2	compute-0-3	17.438 / 17.458

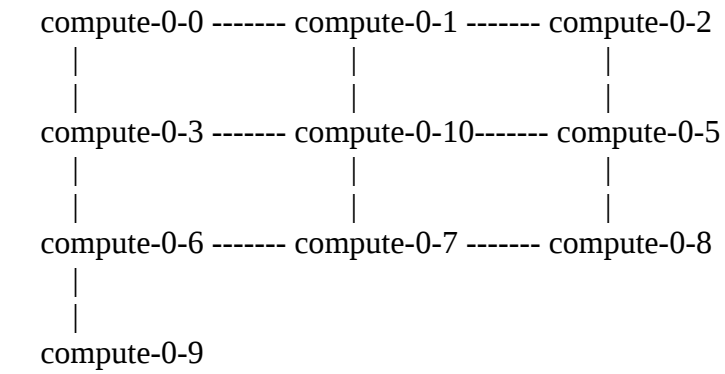
Test Case 2 : Star Test Case with 4 nodes

compute-0-0 - Peers: compute-0-1, compute-0-2, compute-0-3
compute-0-1 - Peers: compute-0-0, compute-0-2, compute-0-3
compute-0-2 - Peers: compute-0-0, compute-0-1, compute-0-3
compute-0-3 - Peers: compute-0-0, compute-0-1, compute-0-2

Star-4			
File Size	Requestor	Provider	Time
1k	compute-0-0	compute-0-3	34.28
2k	compute-0-0	compute-0-1, 0-2	1.32
4k	compute-0-0	compute-0-1	9.81
8k	compute-0-0	compute-0-3	17.63
Multiple requests			
8k	Compute-0-0, 0-1, 0-2	compute-0-3	17.711 / 17.685

Test Case 3 : Mesh Test Case with 10 nodes

10 Nodes : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9

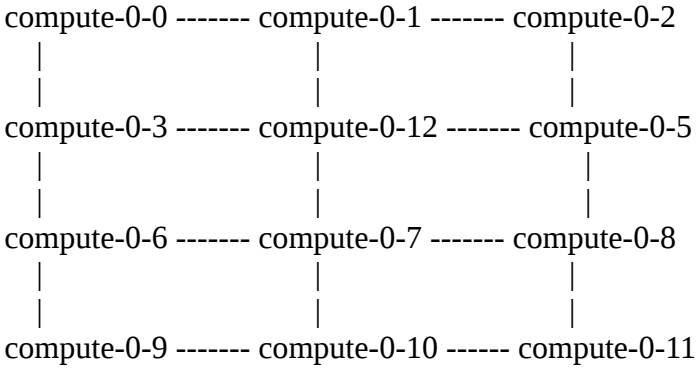


compute-0-0 - Peers : compute-0-1, compute-0-3
compute-0-1 - Peers : compute-0-0, compute-0-10, compute-0-2
compute-0-2 - Peers : compute-0-1, compute-0-5
compute-0-3 - Peers : compute-0-0, compute-0-10, compute-0-6
compute-0-10 - Peers : compute-0-1, compute-0-5, compute-0-7, compute-0-3
compute-0-5 - Peers : compute-0-2, compute-0-10, compute-0-8
compute-0-6 - Peers : compute-0-3, compute-0-7, compute-0-9
compute-0-7 - Peers : compute-0-6, compute-0-10, compute-0-8
compute-0-8 - Peers : compute-0-5, compute-0-7
compute-0-9 - Peers : compute-0-6

Mesh-10			
File Size	Requestor	Provider	Time
1k	compute-0-1	compute-0-3	34.08
2k	compute-0-3	compute-0-10	1.76
4k	compute-0-7	compute-0-1	9.56
8k	compute-0-0	compute-0-9	17.35
Multiple requests			
8k	compute-0-0, 0-1	compute-0-9	17.612 / 17.975

Test Case 4 : Mesh Test Case with 12 nodes

10 Nodes : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9, compute-0-11, compute-0-12



compute-0-0 - Peers : compute-0-1, compute-0-3
compute-0-1 - Peers : compute-0-0, compute-0-12, compute-0-2
compute-0-2 - Peers : compute-0-1, compute-0-5
compute-0-3 - Peers : compute-0-0, compute-0-12, compute-0-6
compute-0-12 - Peers : compute-0-1, compute-0-5, compute-0-7, compute-0-3
compute-0-5 - Peers : compute-0-2, compute-0-12, compute-0-8
compute-0-6 - Peers : compute-0-3, compute-0-7, compute-0-9
compute-0-7 - Peers : compute-0-6, compute-0-12, compute-0-8, compute-0-10
compute-0-8 - Peers : compute-0-5, compute-0-7, compute-0-11
compute-0-9 - Peers : compute-0-6, compute-0-10
compute-0-10 - Peers : compute-0-9, compute-0-7, compute-0-11
compute-0-11 - Peers : compute-0-8, compute-0-10

Mesh-12			
File Size	Requestor	Provider	Time
1k	compute-0-1	compute-0-3	34.14
2k	compute-0-3	compute-0-10	7.44
4k	compute-0-0	compute-0-1	9.75
8k	compute-0-0	compute-0-9	18.03
Multiple requests			
8k	compute-0-0, 0-1	compute-0-3	17.752 / 17.891

Test Case 5 : Star Test Case with 10 nodes

10 Nodes : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9

compute-0-0 - Peers : compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-1 - Peers : compute-0-0, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-2 - Peers : compute-0-0, compute-0-1, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-3 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-10 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-5, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-5 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-6, compute-0-7, compute-0-8, compute-0-9
compute-0-6 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-7, compute-0-8, compute-0-9
compute-0-7 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-8, compute-0-9
compute-0-8 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-9
compute-0-9 - Peers : compute-0-0, compute-0-1, compute-0-2, compute-0-3, compute-0-10, compute-0-5, compute-0-6, compute-0-7, compute-0-8

Star-10			
File Size	Requestor	Provider	Time
1k	compute-0-0	compute-0-3	36.28
2k	compute-0-0	compute-0-10	3.66
4k	compute-0-0	compute-0-1	11.86
8k	compute-0-0	compute-0-9	19.87
Multiple requests			
8k	compute-0-0, 0-1	compute-0-3	19.911 / 19.885

Analysis :

Analyzing the number of calls made from the logs we found that in PA2 number of RPC calls made to search for a file is very high compared to number of calls made in PA1. (Unfortunately could not use tcpdump or wireshark utilities on the Linux machines as we don't have root privileges on them).

Number of RPC calls made to fetch the file remains the same in PA1 and PA2. This is dependent on the size of the file. In each call, 4096 bytes of data is read and transferred.

In PA1, one RPC call was made to the Index server to get the location of the file. All subsequent calls were related to actual file transfer. In PA2, the number of RPC calls to get the file location is comparatively higher than PA1. On that note, time taken to find the file location is relatively higher than PA1.

In spite of having a MAXTTL of 4 and a broadcast to all once the file is found, number of search calls exchanged is quite high .

However we do not depend on one node to give us the file location. Index server responsibility is now scattered to all peers and hence is no more a bottle neck.

In our implementation we are caching the results of the previous queries in each peers local cache. And for future requests we send these results. With our caching feature that we have in our project, time taken to search a file should get reduced as time progresses.

Possible improvement :

Today if a peer's name is not listed against the requested file we relay request to it again. We do this because, we want to make sure that if the file was added later, we should be able to detect it. However, if the file was not dynamically added (which is the case with us today) this call is a waste.

A possible improvement would be for the peers to send a negative hitqiery (concept similar to negative dentry cache) and we could cache it. So, next time we get a query we don't blindly relay the query to peers. Instead if there is already a negative entry against that peer, we don't bother relaying the query to it. This would reduce the network traffic generated. However, with this there should be periodic timing out of negative entries or clearing of negative entries when the peer starts serving a file which it was not in the past.

Multiple client query (possible improvement) :

We found that when multiple clients were trying to access the same file the network traffic was high as two queries were generated for the same file.

A possible improvement would be : the peer server which initiates the query (in our case the peer server which is processing the search_1_svc) can check if there is already a pending request for the same file. If yes, just wait for that query to complete as it would fetch the needed results anyway. For this the comparison would have to be filename and not sequence number (like we do today).