PA1 report
Group 5
The structure of the project

Yuhan Li A20431466
Ji Tin Justin Li A20423037
Subowen Yan A20430537

For the server, we create 5 ports. One port is for listening the peers to register the filename, the other four ports are used to listen the peers to search.
We also create an object to save each filename information.

```
class PeerInfo
{
    String filename;
    int peerid;
    String ipAddress;
}
```

In search function, we use hashmap to save the data, so that we can have a better performance in searching execution. If the server find the data in hashmap, it will return the hostname. Otherwise, it will show which file it can not find and return “X” to peer client.
Everystep has a clear instruction on screen.

Evaluation and Measurement

First of all, to test whether our program works on a small scale or not. We pass into several binary files and texts file. It’s working. The following screenshots show that we are doing simple experiments.

```
========== Central Index Server start ==========
Awaiting peer connect to register and search...
The sever is connecting from  127.0.0.1  for Registration.
Sever start to register...
The peerID and filenames are:  1234 abcd.txt
Peer register on sever successfully !!!!!
-----------------------------------------------
The sever is connecting from  127.0.0.1 for Search
Sever start to search...
Central Index Server finds the file ( abcd.txt ) in peer:
1
The IP address is :
127.0.0.1
Search end.
-----------------------------------------------
```

Figure 1: The server interface

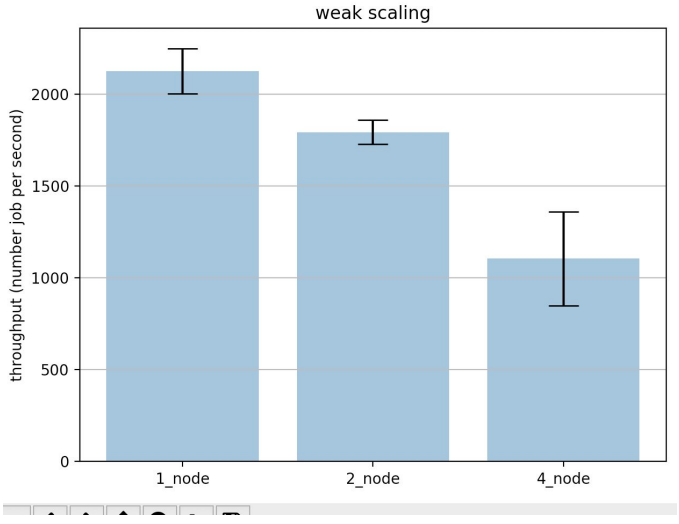
```
Enter The Option :
==================
1. Registering the File

2. Searching On CentralIndxServer

3. Downloading From Peer Server

4. Exit

1
Enter the String in Format: 4Digit id and File Names separated by Space
1234 abcd.txt

Connected to Register on CentralIndxServer on port 2001

Registered Successfully!!
```

Figure 2 : The client interface (1)

```
Enter The Option :
==================
1. Registering the File

2. Searching On CentralIndxServer

3. Downloading From Peer Server

4. Exit

2
Enter the File Name to Search
abcd.txt

Connected to Search on CentralIndxServer on port 2002

File:'abcd.txt' found at peers:127.0.0.1
```

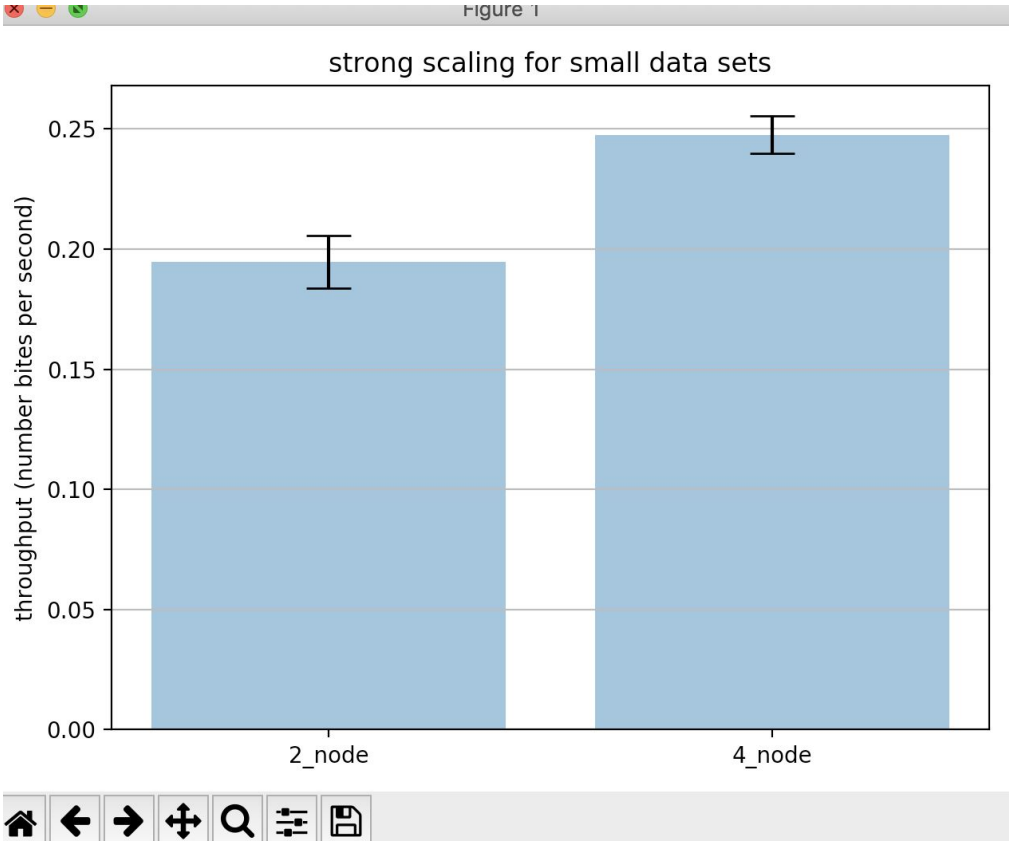
Figure 3: The client interface (2)

The following graph shows the relations between throughput (number of jobs per second) and the number of nodes on the weak scaling graph.



As we can see, adding more nodes will reduce the throughput for the program. Mean for weak test with one node will be 2124.72 number of jobs per second. Standard deviation for weak test with one node will be 123.14. Mean for weak test with two node will be 1793.30 number of jobs per second. Standard deviation for weak test with two node will be 66.33. Mean for weak test with four node will be 1104.95 number of jobs per second. Standard deviation for weak test with four node will be 255.96.

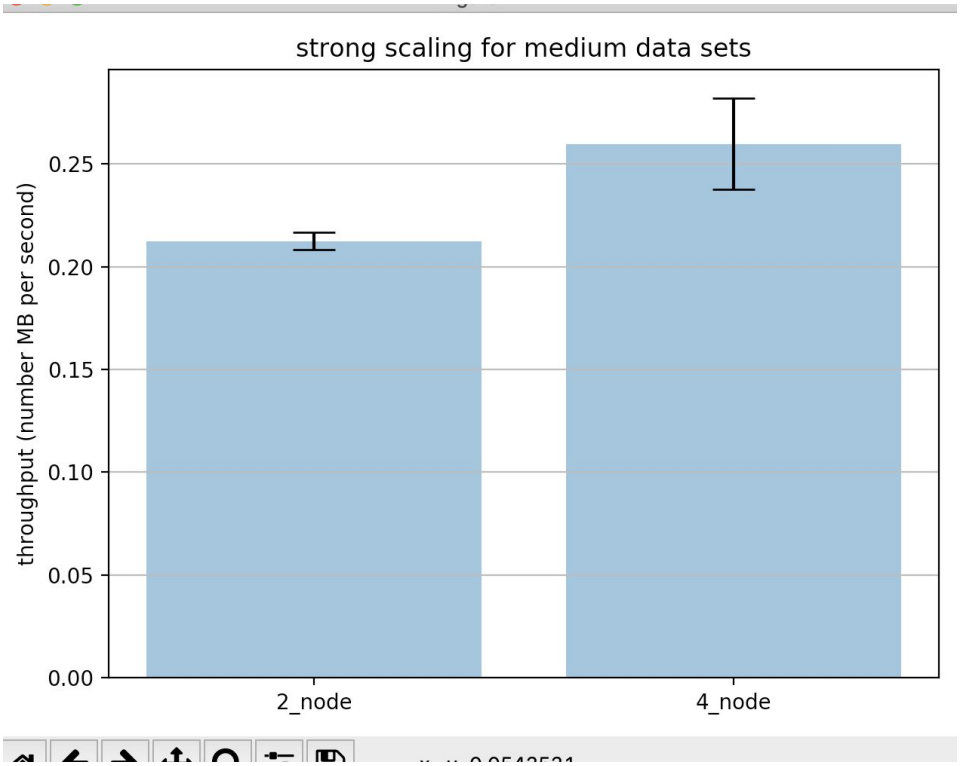
The following graph shows the relations between throughput (number of MB per second) and the number of nodes on the strong scaling graph with 10k files. We only plot two nodes and four nodes for doing strong scaling. The reason why we do not do one node experiment is that doing one node experiment for strong scaling is equaling to searching files without transferring it. Without transferring, we cannot get the right throughput.



Therefore mean for strong test with a small size of files and two node will be 0.1946 MB per second. Standard deviation for strong test with a small size of files and two node will be 0.0108 MB. Mean for strong test with a small size of files and four node will be 0.2476 MB per second. Standard deviation for strong test with a small size of files and four node will be 0.0078 MB.

As we can see, adding more nodes will increase the performance for the program.

The following graph shows the relations between throughput (number of MB per second) and the number of nodes on the strong scaling graph with 1k files.



mean for strong test with a medium size of files and two node will be 0.2125MB per second.
Standard deviation for strong test with a medium size of files and two node will be 0.0043 MB.
Mean for strong test with a medium size of files and four node will be 0.2597 MB per second.
Standard deviation for strong test with a medium size of files and four node will be 0.0222 MB.

As we can see, adding more nodes will increase the performance for the program, as well.

We can deduce that our P2P centralized system is scalable up to 4 nodes. It scales well for all file size and all types of files (mainly using binary files). Based on the data we have; it will continually decrease for weak scaling test no matter what files we have if we had 1k peers. However, if we had 1 billion peers, the graph will be decrease at beginning, after a while, it will be flat. For strong scaling, based on the we have so far, if I had 1k peers with small, medium, large files, the throughput will be increasing. However, if had 1billion peers with small, medium, and large files, the throughput will be increasing first for a while, and starting to decrease.