**Napster-Style Peer-to-Peer (P2P) File Sharing System**

Vaishnavi Manjunath

## INTRODUCTION:

Peer-to-Peer(P2P) Technologies are used for sharing the data between the servers and the peers. One of the most implemented technology these days is the Napster-Style Peer-to-Peer File Sharing System. The concept is file distributed throughout the nodes.

The older versions of the systems used to have a central server which stores the files in its directory that are received from the peers. All transfers would happen only between the central server and the peers.The major drawback of these systems was that if a new file has been created in one of the peers, it must be transferred to the server before another peer can access it, which delays the process of transfer from one peer to another. This can be conquered using the Napster system which allows the peer to peer file transfer.

## SYSTEM REQUIREMENTS:

* JDK and Java to be installed
* Server to execute the program on multiple systems

## DESIGN:

Entire project is designed using Java where we have used the concepts of Socket Programming and Multi-threading. For establishing the connections between the Server and the Peers, we have used TCP/IP protocol using the sockets.

## Major Components of the Project:

* Server and
* Peer

## Server (Central Index Server):

This server indexes the content of all the peers that register with it. It also provides a search facility to peers.

## Server Functionalities:

* Registry and
* Search

## Peer:

As a peer, the user specifies a file name with the indexing server using "lookup". The indexing server returns a list of all other peers that hold the file. The user can pick one such peer and the peer then connects to this peer and downloads the file.

## Major function of the peer:

* Download
* Multiple request search

As a server, the peer waits for requests from other peers and sends the requested file when receiving a request. The peers here, act as both the peer and the server. This server is different from the central index server which only indexes the files. But, the server functionality of the peer can be used to download the files from its directory. The peer downloads the files from other peers into its directory.

## The peers provide the following interface to the users:

1. Register – registers the file into the server
2. Search – searches the server for a file and returns the list of peers
3. Download – downloads the file from another peer

## TRADE OFFS:

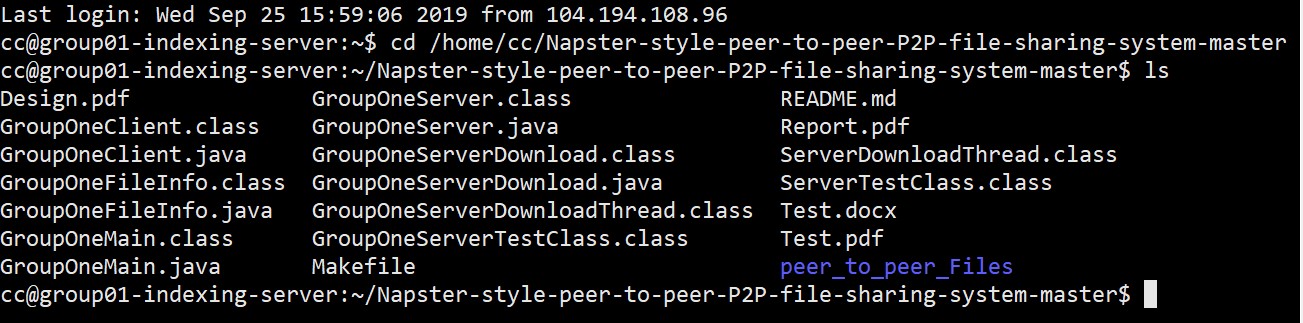
Instead of using the Array List for indexing, we can make use of the Data Structures. Even though Array List works fine in our case, but in case of randomly searching the files, hashing techniques serves well.

## POSSIBLE IMPROVEMENTS:

* We can improve the performance using the Data Structures.
* Could develop a User Interface.
* Port numbers can be eliminated.

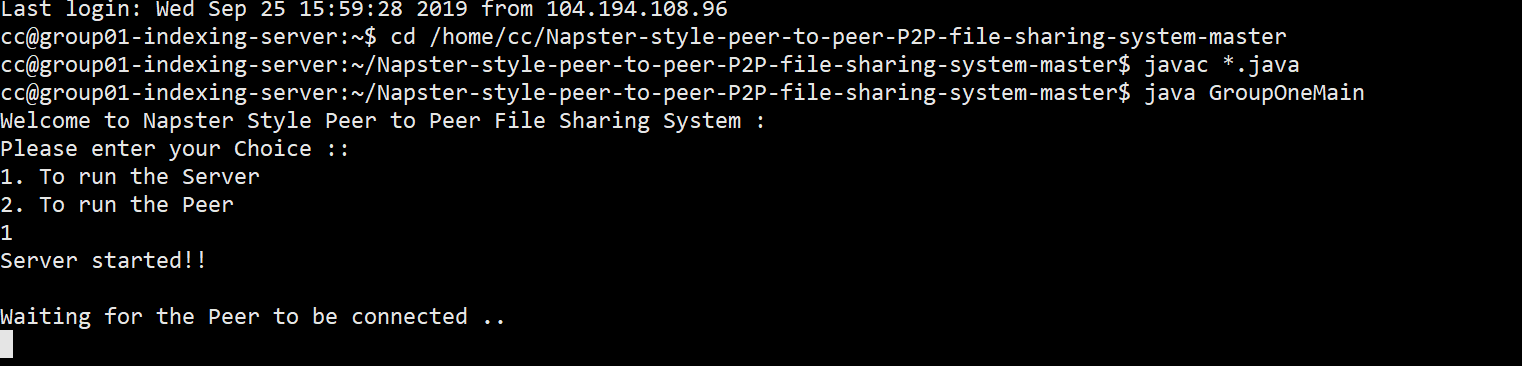
**EXECUTION OF THE PROJECT:**

Below is the screenshot of the available java files present in the directory along with the Makefile



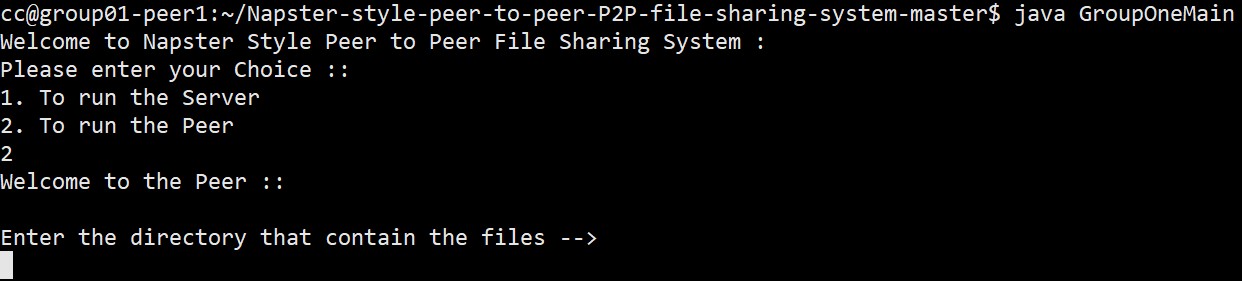
Server:

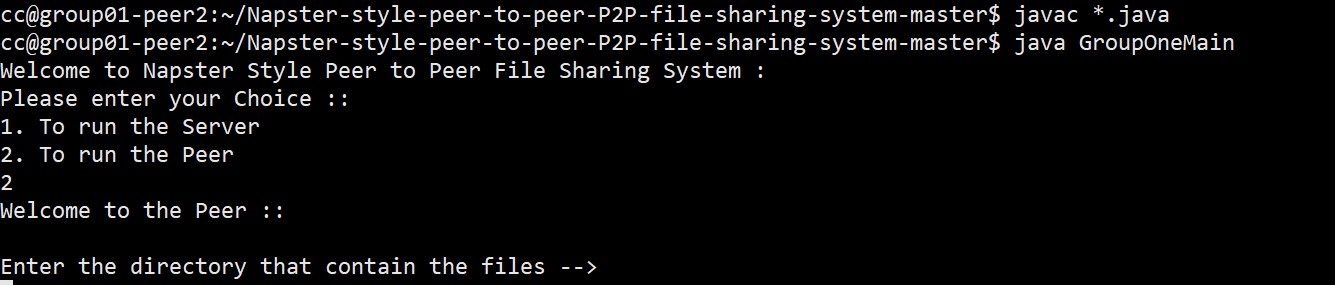
We Start our execution by executing the Server –

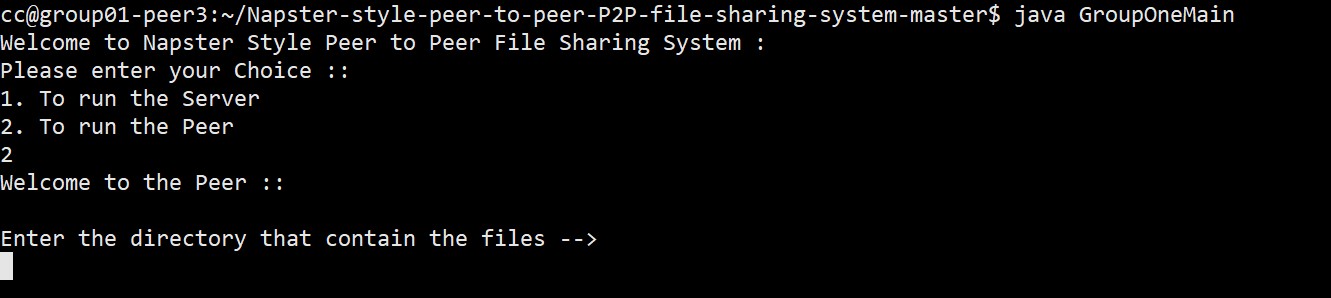


Now Server will open its Socket and waits for the Peers to get connected Peer:

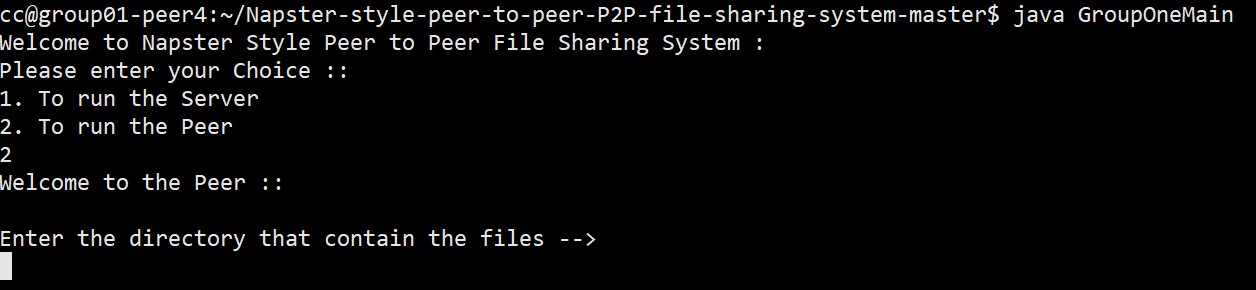
We will register 4 Peers (Client) on to the Server as per the project requirement For Peer1 -



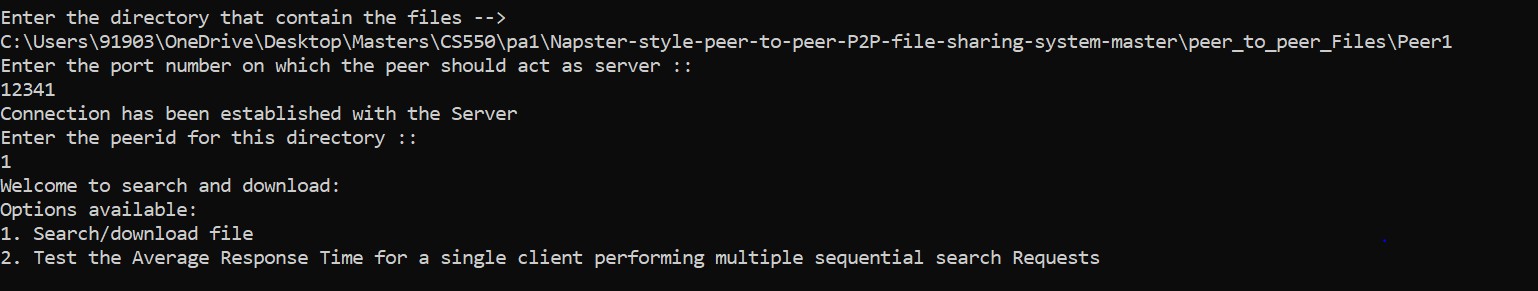
For Peer2 -

For Peer3 -

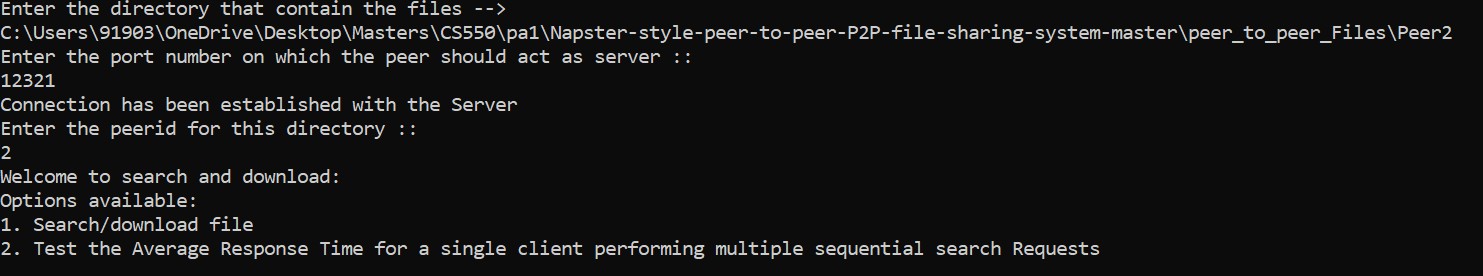
For Peer4 –



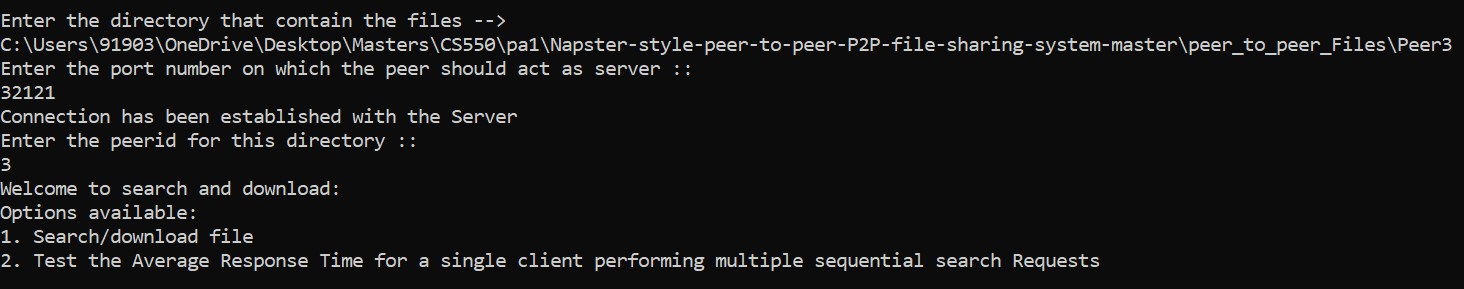
Now, we will enter the directory of the files for each Peer along with the port number and the peerid For Peer1 –



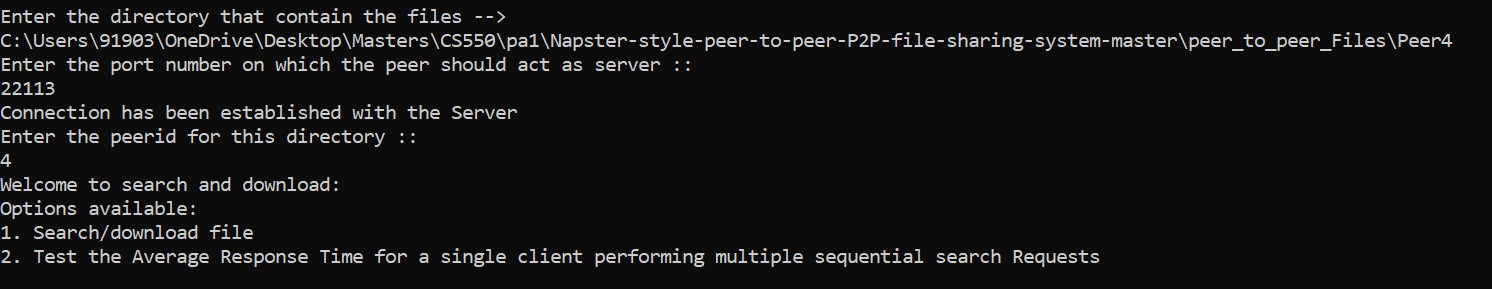
For Peer2 –



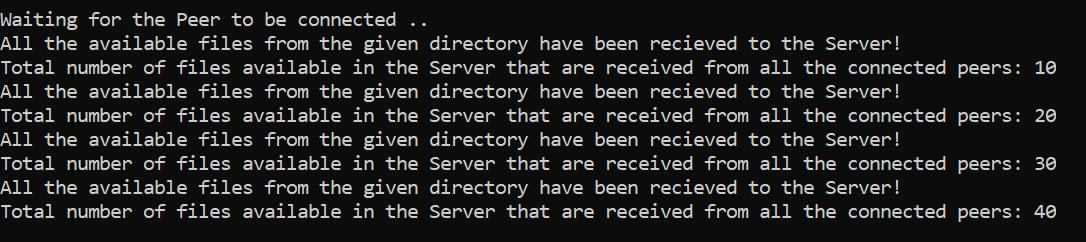
For Peer3 –



For Peer4 –



Once all the Peers have registered on to the server, the total available filenames at each of the Peers will be registered in the Server –

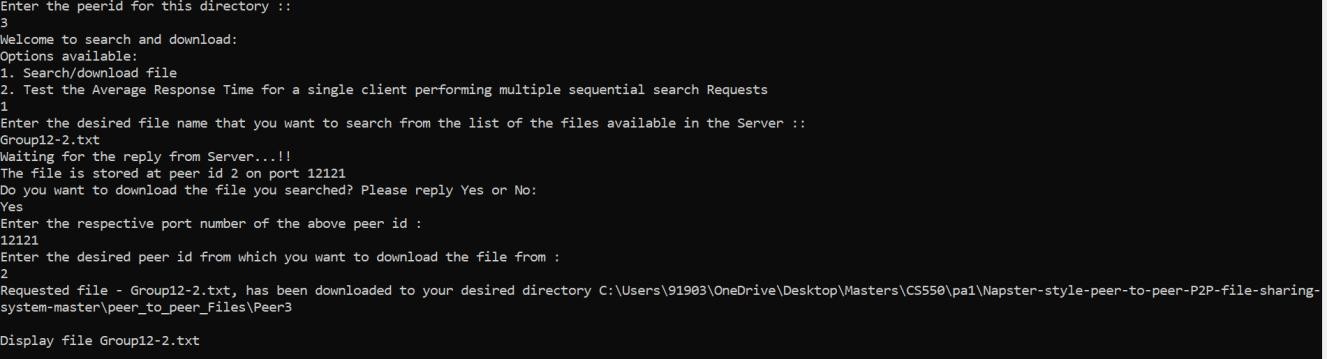


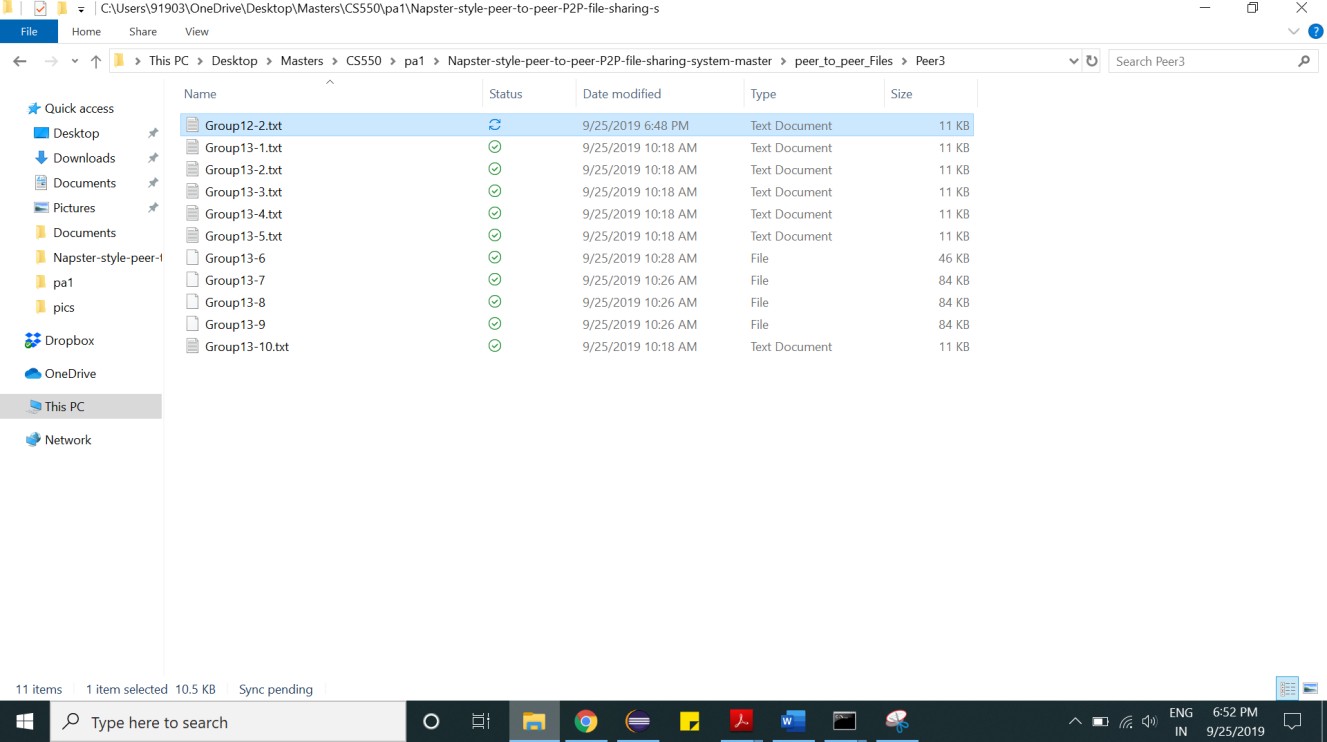
1. **For Searching the Files and Downloading it to the desired folder:**

Go to the desired Peer (Say suppose, Peer 3) to where you want to download the file to.

Give the name of the desired file that you want to download from the list of the files available in the Server along with corresponding port number and the peerid

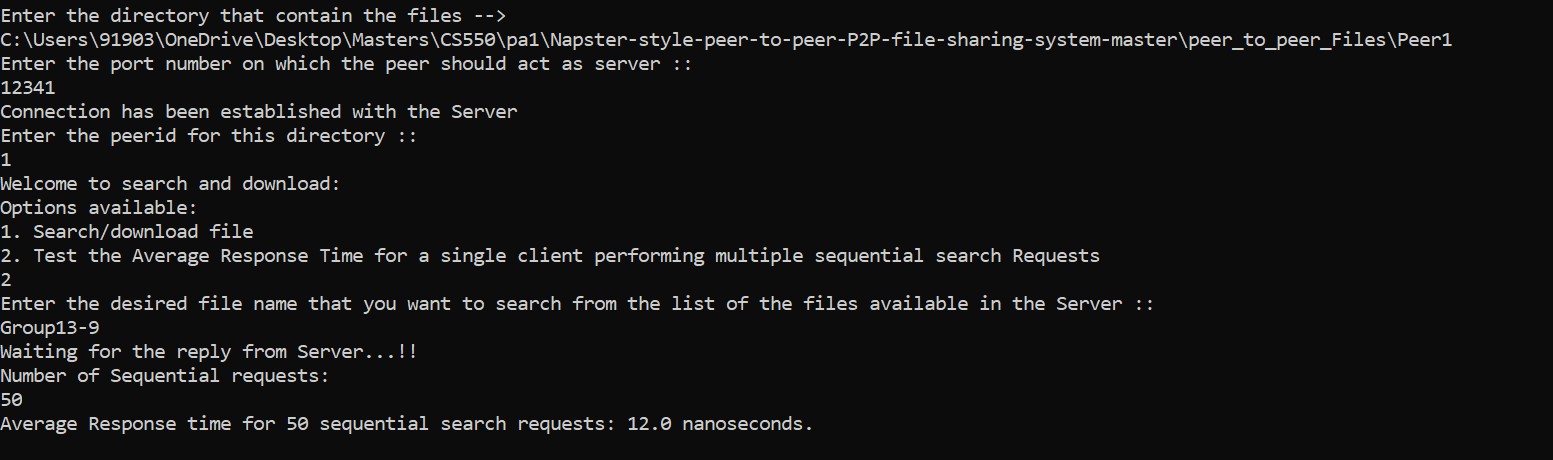
First file will be searched and then for downloading a message will be popped if given Yes file will be downloaded in the respective file folder and it displays File Name





1. **For multiple sequential requests:**

File from Peer-3 was requested to be searched for 50 times



PERFORMANCE EVALUATION:

Weak Scaling:

Weak Scaling Evaluation

2.5

2.0071

1.9476

1.9047

2

1.63031

1.8625

1.58294

1.5

1.8802

1.7491

1

0.5

0

Number of Nodes

File - 10kb

File - 100kb

Files/Nanoseconds

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Node-1 | Node-2 | Node-3 | Node-4 |
| File - 10kb | 1.8802 | 1.7491 | 1.63031 | 1.58294 |
| File - 100kb | 2.0071 | 1.9476 | 1.9047 | 1.8625 |

Standard Deviation

Average

Node-4

Node-3

Node-2

Node-1

0

0.197678772

0.19402303

0.140360696

0.089731851

0.5

1

1.5

1.7227

1.7675

1.8483

2

1.9436

2.5

Weak Scaling Evaluation

Strong Scaling:



Chart Title

2.5

2.1769

2.0813

2

1.8209

1.8779

2.0656

1.4599

1.8375

1.7869

1.5

1.5552

1.5891

1.4847

1.3411

1

0.5

0

Number of Nodes

Files/nanoseconds

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Node-1 | Node-2 | Node-3 | Node-4 |
| File - 100kb | 1.3411 | 1.8375 | 1.8779 | 2.0656 |
| File - 1MB | 2.1769 | 1.8209 | 1.5891 | 1.4847 |
| File - 100MB Binary File | 1.4599 | 1.5552 | 1.7869 | 2.0813 |

Strong Scaling

2

1.8

1.8772

1.6

1.4

1.2

1

0.8

0.6

0.4

0.2

0

1.6593

1.7378

1.7513

0.452173285

0.158411563

0.147654597

0.340005603

Node-1 Node-2 Node-3 Node-4

Average

Standard Deviation