COMP90015 – Distributed Systems – Project 1

Group 30: Mingyang Liu, Jiahao Chen

Question 1:

Calculating the average number of blocks for every peer's operation is applied to help selecting the peers in the case that there was more than one such peer. For example, if there are 2 peers, 5 blocks are needed to download, first 3 blocks will be downloaded from the first peer and remaining 2 blocks will download from the second peer. Each download operation will create a Thread to handle that download, in the case of 2 peers, 5 blocks, 2 threads will be created and each of the threads will download from each peer. Inside one thread it will download in natural order. File will be written depending on which thread receives data first.

If downloading in natural order, blocks will have to stay in the queue until all the previous ones are downloaded. In this case, the system must wait until the previous block is downloaded and written into file so that the system will have a lower efficiency than downloading in an arbitrary order. Download from multiple peers is usually in an arbitrary order, system will not know which block will arrive socket buffer first, therefore, first come first serve strategy is useful and can increase system efficiency when writing blocks into file. However, in this case, file lock must be used to ensure that multiple threads will not write the same file at same time.

Question 2:

Currently, whenever the peer wants to make a request, a socket is required to be established between the peer and the index server. A persistent connection can be established instead so that if a peer needs to perform many requests over a period of time, it only needs one socket. The peer needs to notify the server to close the connection. A timeout is also required so that if a peer is inactive for a long time, the idle socket can be closed to save resources. Since less sockets are established, the system can have less overheads.

Jsonserializable may not be efficient since it is much more complicated than json. Serialisation is slow and serialised data has a large volume, which may lead to slower transmission. Besides, it can only be applied in Java so that it is difficult to plug systems using other languages into it. In addition, Serializable has poor compatibility since the serialVersionID must be the same. Therefore, json can be applied to transmit data because it is simpler and has a much better compatibility.

Although Base 64 encoding and decoding is relatively fast, it increases the file size by over 25% on average, which increases the bandwidth load as well. Directly using bytes for transmission can also reduce the overheads.

A flexible timeout socket can be applied in the protocol, downloading large files usually will have larger time needed than processing requests from peers. Therefore, setting a small timeout when dealing with search, lookup, drop, share requests can save resources when peer is not response, and cause servers to have more ability to deal with new requests.

Question 3:

The socket time parameter is set to limit the maximum allowed time for the socket connection. The socket will not stop attempting the connection if this parameter is not set.

Possible failures:

- Peer is crashed (no available network)
- Remote server is crashed
- The file is too big so that the download time exceeds the timeout parameter
- Network speed is too slow
- Too many requests

The system can handle the first two failures. However, it still cannot perfectly handle the rest 3 failures.

A small timeout value will highly increase the possibility of timeout error. Large files may not be successfully transmitted within the allowed time. But it can improve the system's efficiency because bad connections will be dropped quickly, and the server can process more requests.

A large timeout has higher fault tolerance. For example, if the server manages to restart quickly after crashing and the timeout value is still not exceeded, the socket will still be established. However, the system will have larger overheads if there are a lot of connection failures.

Selection of timeout value can depend on the type of the connection. For instance, if it is a download request, it is better to increase the timeout to allow transmission for large files.
In addition, if the server has limited resources and cannot process many requests at the same time, it may be better to select a smaller timeout value to avoid waste of resources.

Question 4:

A server running on the NeCTAR cloud can handle an infinite number of peers theoretically. Since the server is not using a persistent connection, it creates one connection per request. Therefore, if peers do nothing, users can keep creating new peers. However, that is not possible, peers will always send and receive data from the server, therefore, the number of peers the server can handle will depend on how many requests the server will receive in a small amount of time. To measure server performance, we could consider following aspects:

1. Type of request

Requests like SearchRequest will cost more overhead when there is a large amount of data stored in the server. Server will take a longer time to respond back to peers. Also requests like ShareRequest, server have to store each Share inside memory.

2. Number of files shared

Since the server in this case stores all shared file records in memory, if there is a large number of shares, the server might run out of memory and cause the server to crash.

3. CPU performance

Higher performance of the CPU means the server will handle search requests faster.

To sum up, estimate how big this file sharing system can grow. We can estimate how many requests the server can handle in a given time and timeout. If a server can respond to all peers in time, it means this server can handle at least this number of peers.

After testing by using the code in StressTest branch, the server can easily handle 20 search requests simultaneously, with a given timeout 1000ms, which means, in 1000 millisecond this server can only handle 20 requests. Therefore, we conclude that a single server can handle at most 20 peers if timeout is set to be 1000. Because server have to be run without any problem, if peers is more than 20, and when they download or search simultaneously, server will not respond anything to late arriving peers. If we increase timeout time, server can grow larger with higher timeout time, but this will cause bad user experience.