Scalability

For the file sharing system, Scalability is mainly from two aspects: horizontal and vertical. Our system has problems in both two aspects.

**scale vertically**

for mainly considering one node(server) in the system, we identified resource storage as an issue that constrained the scalability of the system. Currently, we used a HashMap to store all the resources in memory, which limited the amount of resources that could be maintained by a server. And once the server was closed, the resources saved by this server were lost.

Thus, we chose NoSQL database as a solution to this problem. For one thing, all the resources were in form of JSON, which was a perfect fit for NoSQL database like MongoDB and CouchDB, where one JSON could be saved as one record. It would be much harder for us to query or remove a resource if we applied relational database because according to the normalization principles, we would have to separate the data for one resource into several tables. For another, NoSQL database would be easier to apply for partitions and distributed designs, while this would be harder for relational databases.

**scale horizontally**

There are also some scalability issues in our system to be concerned when considering horizontally. For example, if a server successfully accepts exchange command for adding the new server(s), the server will contact to the new server(s) and synchronize all of its data with new server(s). However, with growing resources stored in the system, the time for synchronization as well as the frequency of synchronization will increase. Thus, it’s difficult to make sure each client always to see the same result each time, because in that case, all servers cannot provide any service to clients until it finishes the synchronization processes. If some servers are located far away from others, the time for synchronization can take longer and clients will not able to get services from servers for a long time. This situation adversely affects the performance of the system, and therefore the system is not scalable.

Revisions: Since the basic goal of the system is to ensure servers can deliver services all the time, availability is an essential factor to be taken into consideration. Therefore, when considering the design of the system, choosing of an eventually model, which sacrifices some consistency but make sure availability, will be good choice to maintain scalability. When adding more servers in the system, they can provide services to clients even during the synchronization process. But Under this situation, the server should make some strategies for dealing with the order of synchronized data and newly added data in order to maintain scalability.