HW4 Scalable Bank Server

SHENGXIN QIAN (sq16)

April 11, 2017

1 Server Design

In this assignment, the bank server is constructed by using "Pre-Create Threads" parallelism and "Producer-Consumer" model. The overall structure is shown as Figure 1:

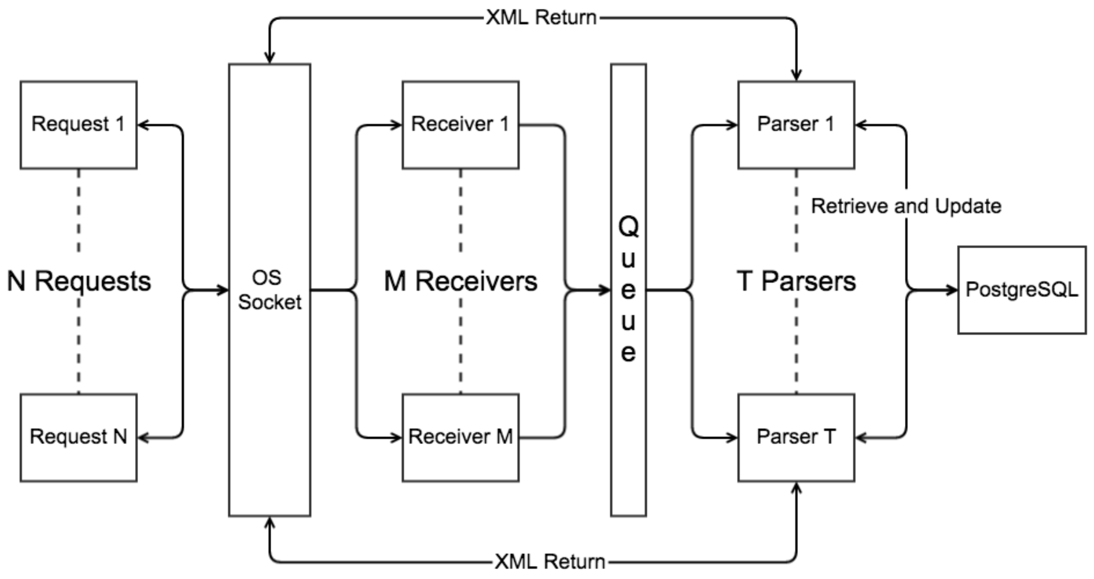

Figure 1: Structure of Scalable Bank Server

In Figure 1, "OS Socket" represent the class "xmlReceiver" which is used to set up the socket of the bank server. This socket is shared by multiple threads. "Receiver" represents the function which is used to accept the requests from shared socket and store the "connection" object (identify the connection with clients), received XML file in the queue. Each "Receiver" is a thread. "Parser" represents the class "xmlParser" which is used to parse the received XML file, update or check the database and return the generated XML file to the clients. Class "xmlParser" will use the error checking functions in module "xmlCheck" to validate the received XML. Each "Parser" is also a thread. The advantage of this model is it can balancing the load of the server by adjusting the number of Receivers or the number of Parsers. In some network critical scenarios, we may need more receivers to handle the connections. In some XML parsing or database critical scenarios, we may need more parsers to balance the load.

2 Performance Evaluation

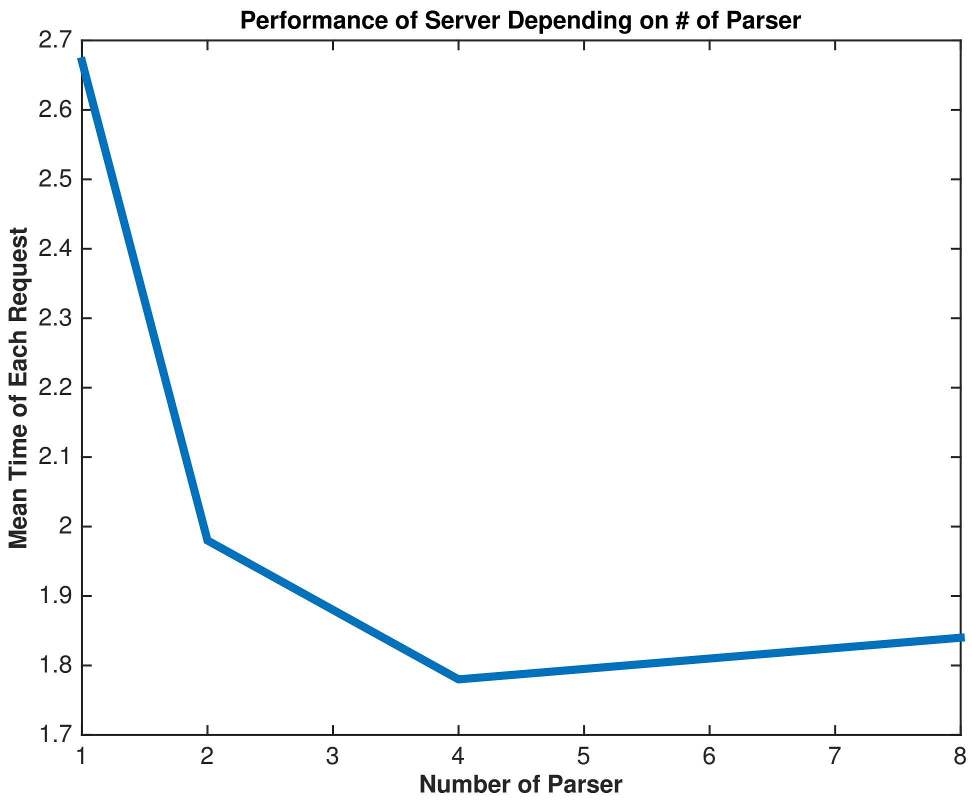

Figure 2: Performance of Server Depending on # of Parser

Because in most of my test, clients and servers are connected by localhost, the network part will not be a critical section. So, after several tests, increasing the number of ”Receiver” will not gain much performance boosting. On the contrary, paring and database communication part is critical to the server performance. In Figure 2, the tests were performed on Intel i5 two-core machine. The number of ”Receiver” is 1 and the number of concurrent requests is 500. As we can see in Figure 2, after we keep increasing the number of ”Parser” threads, the average waiting time of each request is dropping at first and then increasing. The performance boosting at first proves server’s scalability. The performance dropping when there are 8 ”Parsers” is because of the limits of physical cores.

As we can see in Figure 3, with the increasing of the number of concurrent requests, the average waiting time of each request keep increasing. This is because the synchronization between different threads (queue, database lock) will stall the processing of the incomming requests.

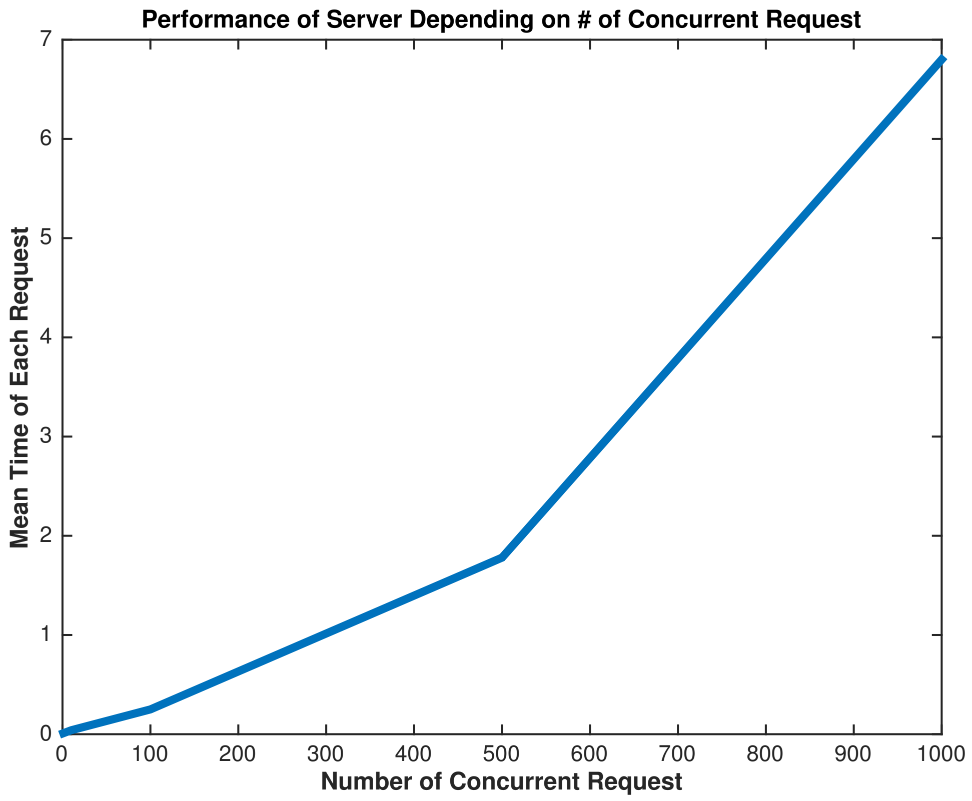

Figure 3: Performance of Server Depending on # of Concurrent Request