CS744 : Design and Engineering of Computing Systems

Asynchronous Server Architecture

Akanksha Dadhich (23M0830)
Chaitra Gurjar (23M0831)

Version 1

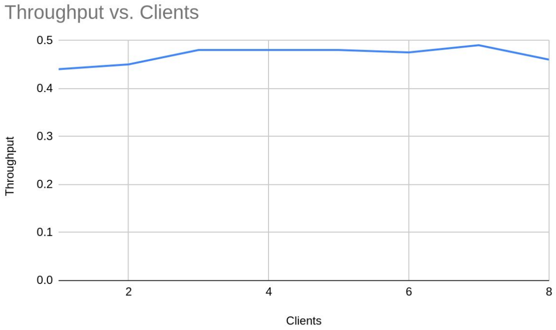
Single Threaded Server

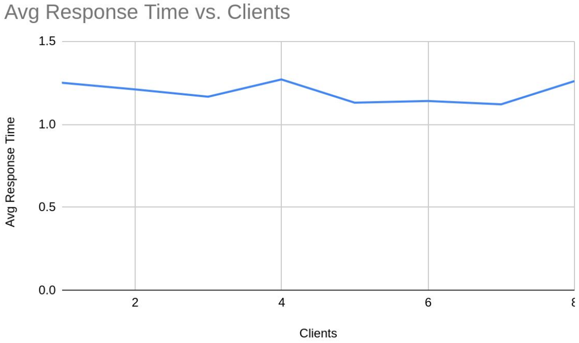
Explanation

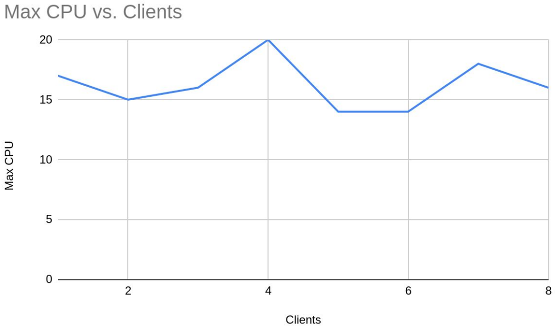
1) We use C language to create a single threaded server.

2) It can process only one client at once.

3) Theoretically, the throughput and response time remains same.







Version 2

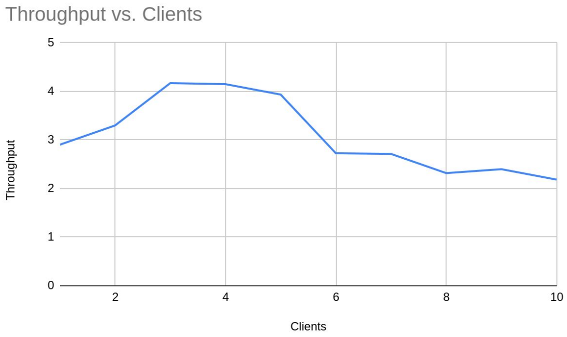
Multi Threaded Server

Explanation

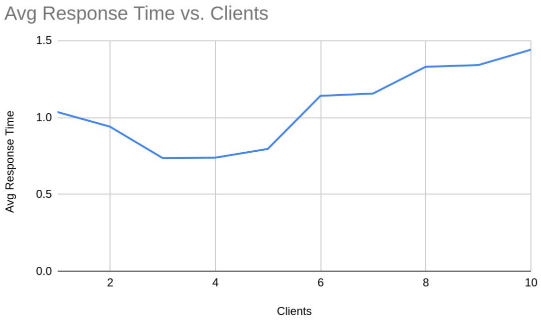
1) We use the pthread library to create a multi threaded server.

2) A new thread is created and assigned to each request.

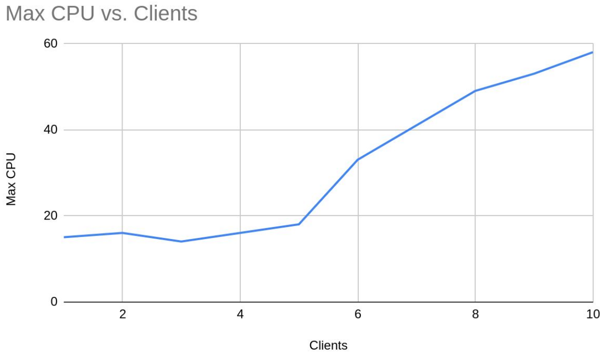
3) Theoretically, the throughput increases till some number of clients and then stabilizes.



Throughput vs. Clients



Avg Response Time vs. Clients



Max CPU vs. Clients

Version 3

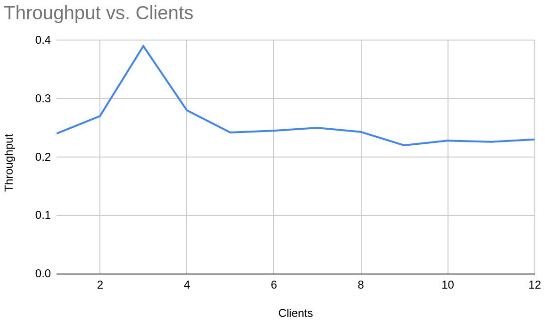
Server with Thread Pools

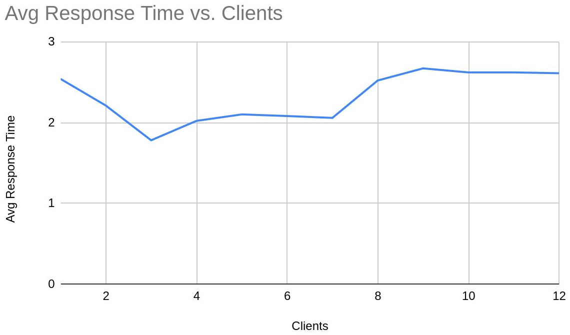
Serving the Request

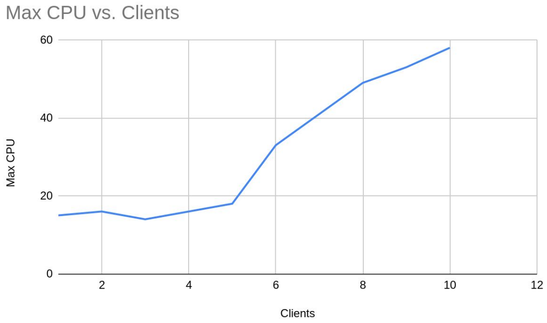
1) We created a thread with ‘n’ threads.

2) Each request waits in a queue before it is assigned a thread.

3) Theoretically, the throughput increases until ‘n’ clients are reached, and then stabilizes.


Throughput vs. Clients


Avg Response Time vs. Clients


Max CPU vs. Clients

Version 4

Asynchronous Server Architecture

Generation of Request ID

1) Now the ‘submit’ program takes one more argument: the string ‘new’ or the string ‘status’.

2) If the argument is ‘new’ the client is sending a new request and the 3rd argument should be the filename.

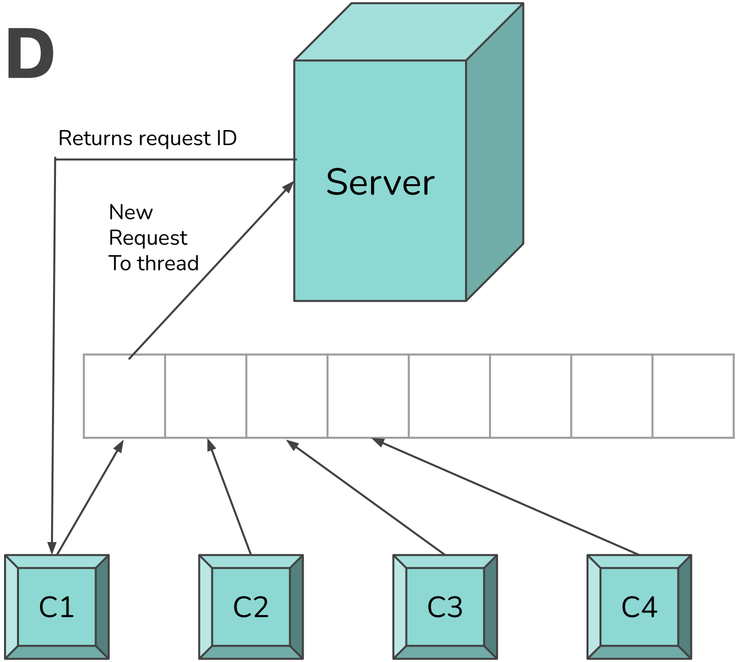
3) If old, the client is checking the status of an old request and the 3rd argument should be the requestID.

4) For generating this new requestID we are using a global variable requestID and updating it each time a new request arrives.

5) We are having a lock on it so to avoid race around problem. We are sending this to client each time a new request comes.

6) For storing the status of the request we are using HashMap data structure where the key would be requestID and the value of the key would be any of the following status “In queue” , “in process” , “Completed”.

7) As the request arrives we are updating the requestId and sending the response to the client with the requestID.



Serving the Request

1) Simultaneously we are adding the request in queue for the threads to take and adding pair (requestId, status) into hashmap

2) Now once a thread picks up this request we are updating the hashmap value of this requestId key to “In progress”

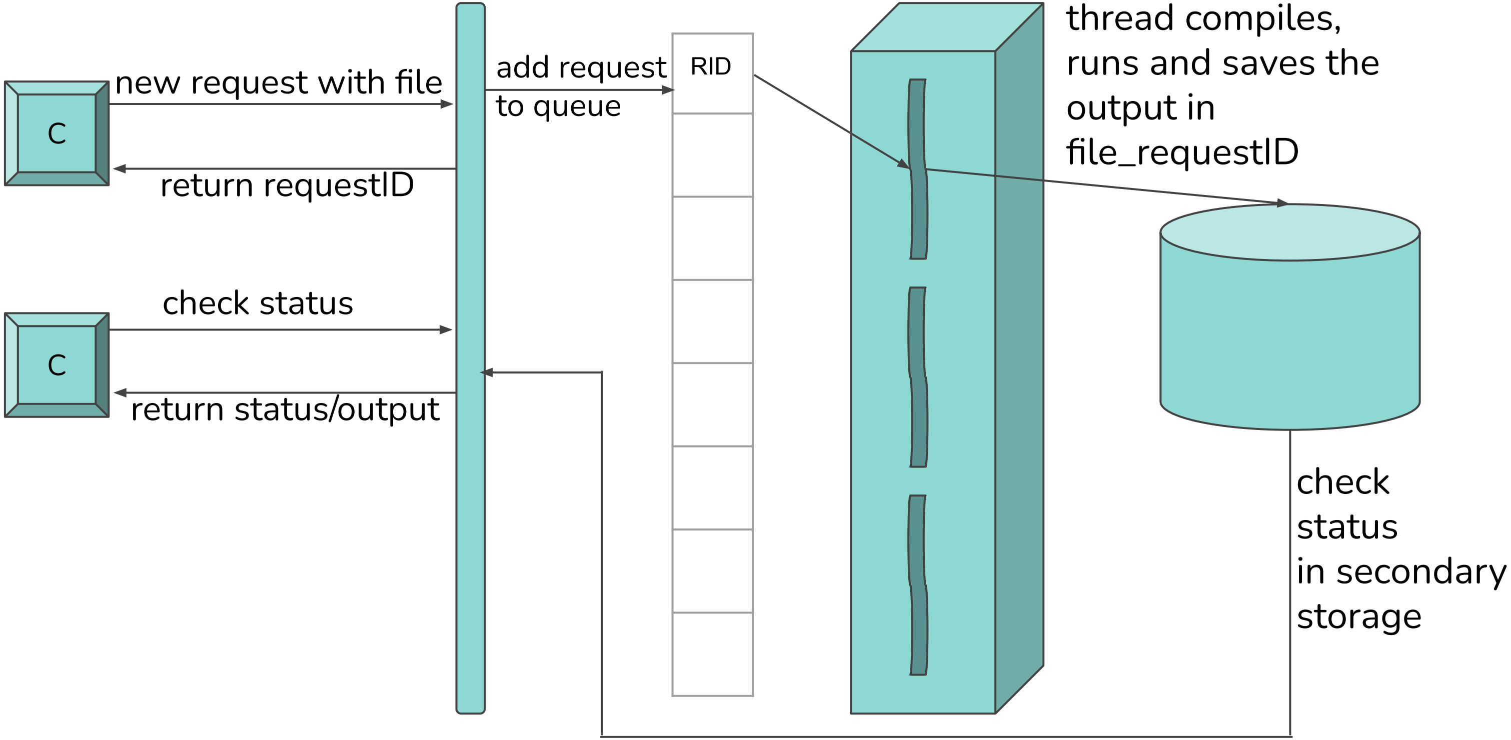
3) Once the task is complete we are storing the file output into secondary storage with a unique name as “file_requestId” for server restart issues and now we are also updating the hashmap requestId key value to “Complete”.

4) Note that we are not removing the key.

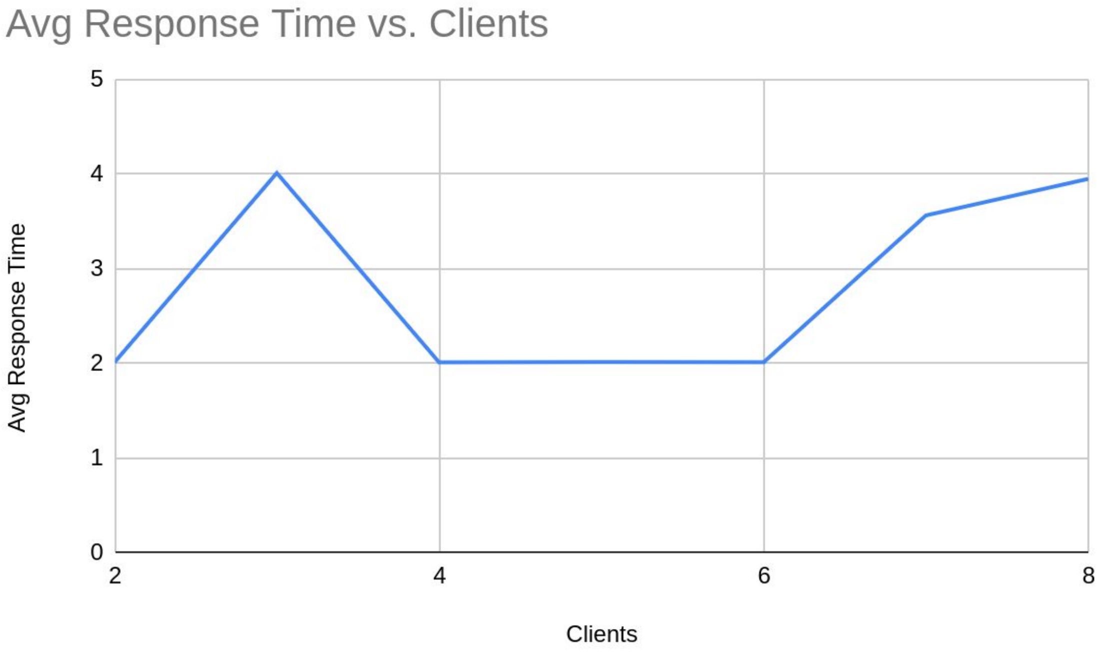
5) Now if client ask for response during this time we are searching the hashmap with the requestId and sending the value of the key requestID.

6) If the value of the response turns out to be completed we are now removing the key value pair (RequestId , “Completed”) from our hashmap

Explanation Figure



Performance



Thank You