Raed A. Hemeed, #SID A20539655
CS 550
FALL23, WA2
Due: Oct 12, 2023

1. **A single-threaded server** is a server that uses only one thread of execution to handle incoming requests. This means that it can handle only one request at a time, and all incoming requests are processed sequentially.
   A multi-threaded server uses multiple threads of execution to handle incoming requests. This allows the server to handle multiple requests simultaneously, with each thread processing a different request concurrently.

- Number of requests handled by a single threaded server:
  Time taken to process a request =15 msec
  Time taken for disk operation =75 msec

  Number of disk operation occurs one-third of the time and therefore, for every 3-process request, there will be 1 disk operation.
  Time taken by a request if there is no disk operation = 15 msec
  Time taken by a request if disk operation is required = 15+75= 90 msec

  - Weighted average of time taken in case of disk operation and no disk operation
  = $\frac{2}{3} \times 15 + \frac{1}{3} \times 90$
  = 10 + 30
  = 40 msec

  - Number of requests per second handled by a single-threaded server
  = $\frac{1\ request}{40\ msec}$
  = $\frac{1000}{40}$
  = 25 request/sec

- Number of requests handled by a multi- threaded server:
  Time taken by a request if there is no disk operation = 15 msec

  - Number of requests per second handled by a multi-threaded server
  = $\frac{1\ request}{15\ msec}$
  = $\frac{1000}{15}$
  = 66.66 request/sec

2. Yes, for two reasons. First, threads require memory for setting up their own private stack. Second, having many threads may consume too much memory for the server to work properly.

3.

Advantages of Spawning Processes:

i. Isolation: Processes are more isolated from each other since they have their own memory space. This can lead to better fault tolerance, as one process's failure is less likely to affect others.

ii. Parallelism: Processes can take full advantage of multiple processor cores as they run in separate memory spaces. This can result in better performance on multi-core systems.

Disadvantages of Spawning Processes:

i. Communication: Inter-process communication (IPC) is generally more complex and slower than thread communication. This can lead to synchronization and data sharing challenges.

ii. Platform Dependent: Process management and creation can be platform-dependent, making the code less portable.

Advantages of Multithreaded Servers:

i. Communication: Threads can easily communicate with each other through shared memory, which simplifies data sharing and synchronization.

ii. Platform Independence: Thread management and creation are usually more consistent across platforms, leading to greater portability of code.

Disadvantages of Multithreaded Servers:

i. Isolation: Threads share the same memory space, which means a bug or error in one thread can potentially affect other threads

ii. Debugging Complexity: Debugging multithreaded applications can be more challenging due to the non-deterministic nature of thread execution.

4. Assuming the server maintains no other information on that client, one could just ably argue that the server is stateless. The issue is that not the server, but the transport layer at the server maintains state on the client. What the local operating systems keep track of is, in principle, of no concern to the server.

5. Both the client and the server create a socket, but only the server binds the socket to a local endpoint. The server can then subsequently do a blocking read call in which it waits for incoming data from any client. Likewise, after creating the socket, the client simply does a blocking call to write data to the server. There is no need to close a connection.

6. Yes, but only on a hop-to-hop basis in which a process managing a queue passes a message to a next queue manager by means of an RPC. Effectively, the service offered by a queue manager to another is the storage of a message. The calling queue manager is offered a proxy implementation of the interface to the remote queue, possibly receiving a status indicating the success or failure of each operation. In this way, even queue managers see only queues and no further communication.

7. Having the user specify the size makes its implementation easier. The system creates a buffer of the specified size and is done. Buffer management becomes easy. However, if the buffer fills up, messages may be lost. The alternative is to have the communication system manage buffer size, starting with some default size, but then growing (or shrinking) buffers as need be. This method reduces the chance of having to discard messages for lack of room but requires much more work of the system.

8. The idea is very simple: if, during gossiping, nodes exchange membership information, every node will eventually get to know about all other nodes in the system. Each time it discovers a new node, it can be evaluated with respect to its semantic proximity, for example, by counting the number of files in common. The semantically nearest nodes are then selected for submitting a search query.

9. A process is an executing program. One or more threads run in the context of the process. A thread is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread.

10. The advantage of using the non-blocking send occurs when the system buffer is full. In this case, a blocking send would have to wait until the receiving task pulled some message data out of the buffer. If a non-blocking call is used, computation can be done during this interval.

11. Use a buffer if the stream is going to have lots of small access. However, the only time you should use unbuffered I/O is when the delay and aggregation imposed by buffering is inappropriate to your application. In other words, If the channel is unbuffered, the sender blocks until the receiver has received the value.

12. Asynchronous messaging is a communication pattern for applications to exchange messages without requiring immediate responses. In asynchronous messaging, the sender and receiver operate independently of each other in terms of timing. The sender can send a message or request and continue its work without waiting for a response. The receiver, on the other hand, processes the message or request whenever it becomes available or when it is ready to do so. This decoupling of sender and receiver allows for greater flexibility and scalability in distributed systems.

13. **RPC** stands for Remote Procedure Call which supports procedural programming. It's almost like an IPC mechanism wherever the software permits the processes to manage shared information Associated with an environment wherever completely different processes area unit death penalty on separate systems and essentially need message-based communication.
    RMI stands for Remote Method Invocation, which is a like RPC, but it supports object-oriented programming which is java's feature. A thread is allowable to decision the strategy on a foreign object. In RMI, objects are passed as a parameter rather than ordinary data.
    WS ?

14. Moore's Law states that "the number of transistors on a microchip doubles every two years." It's no secret that Moore's Law has had a huge effect on computing power. That's because the transistors used in integrated circuits have gotten faster and faster. Transistors are made up of molecules that made up of carbon and silicon that move electricity around the circuit. The quicker the transistors move electricity, the faster the computer will run.
    I think that the high temperatures of the transistors would make the generation of smaller circuits impossible. This is because cooling the transistors consumes more energy than is already flowing through them. Without cooling, transistors won't be able to operate many processes at the same time, which means we will reach a certain fast circuit, and then stop that development.

Reference:

Q4: http://www.leonardomostarda.net/ds/5-CS-AnatomyForProf.pdf
Q5: http://www.leonardomostarda.net/ds/7-communicationforProf.pdf
Q6 & Q8: http://www.dl.edi-info.ir/Distributed%20systems%20principles%20and%20paradigms.pdf
Q9: https://learn.microsoft.com/en-us/windows/win32/procthread/processes-and-threads
Q10: https://www.cs.mtsu.edu/~rbutler/courses/pp6330/www.navo.hpc.mil/pet/Video/Courses/MPI/Mod_2/Slides/more.html#:~:text=The%20advantage%20of%20using%20the,be%20done%20during%20this%20interval.
Q11: https://stackoverflow.com/questions/1088602/buffered-vs-non-buffered-which-one-to-use
Q12: https://memphis.dev/blog/the-power-of-asynchronous-messages-in-distributed-systems/#:~:text=Asynchronous%20Messaging%20and%20why%20it%20is%20important,-Asynchronous%20messaging%20is&text=The%20receiver%2C%20on%20the%20other,and%20scalability%20in%20distributed%20systems.