1. Intel and AMD are the major manufacturers of general purpose processors today (2019). What is the most number of cores that a high-end server processor from AMD or Intel will have?
A) 72 cores

2. What properties does a distributed system have?
A) Connected by a network, Independent Computers, Multiple Computers, complex structure

a. Resource sharing b. Openness c. Concurrency d. Scalability e. Fault tolerance f. Transparency

3. **Which of these are examples of distributed systems?**
A) Facebook,Youtube, Google Drive, etc..

4. **What is the difference between a 6-core processor and a 28-core processor from the Intel Scalable Processor lineup?**
A) There is no difference (They have blocked the other cores)

5. **Why is it cheaper to build a distributed system than a centralized system for a given quantity of computing capacity, memory, and storage?**
A) Manufacturing costs to build large quantity of small/medium system is better than building small quantity of large systems.

6.**What is the difference between a process and a thread?**
A) A process is a “program” with its own address space. – A process has at least one thread!

A thread of execution is an independent sequential computational task with its own control flow, stack, registers, etc. – There can be many threads in the same process sharing the same address space

• A UNIX Process is – a running program with – a bundle of resources (file descriptor table, address space)

• A thread has its own – stack – program counter (PC) – All the other resources are shared by all threads of that process. These include: u open files u virtual address space u child processes

-**(alex ans**)A process is an abstraction representing a program in execution. It encapsulates all of the required information for that program to run: Registers, Memory, I/O. A thread is the most basic unit of computation that typically runs as part of a process and each thread has its own Registers and Stack, sharing the Heap and I/O information with other threads.

7. **What is the difference between a software thread and a hardware thread?c**
A) A hardware thread is a physical CPU or core. One hardware thread can run many software threads through time slicing
However, each physical core can offer more than one hardware thread, also known as a logical core or logical processor.
Windows Operating System recognizes each hardware thread as a schedulable logical processor. Each logical processor can run code for a software thread.

Software threads are **threads** of execution managed by the operating system. **Hardware threads** are a feature of some processors that allow better utilisation of the processor under some circumstances.

8. **Why do we need synchronization locks (e.g. mutex) in a language like C/C++?c**
A) it is to prevent 2 or more threads from accessing a shared data variable concurrently
https://web.mit.edu/6.005/www/fa15/classes/23-locks/#deadlock

A2) atomicity and critical section.

A3) The **mutex** class is a synchronization primitive that can be used to protect shared data from being simultaneously accessed by multiple threads. **mutex** offers exclusive, non-recursive ownership semantics: A calling thread owns a **mutex** from the time that it successfully calls either **lock** or try_lock until it calls unlock
https://en.cppreference.com/w/cpp/thread/mutex

9. What reasons are there to use a virtual machine for?
A) - Try New Operating Systems
- Run a Web Server or Other Software
- Create Backup Snapshots
- Run Old or Incompatible Software
- Deliberately Execute Malware
	i) Due to the sandboxed nature of a virtual machine, you can be reckless with security and do things that you normally should avoid. For example, you should never open unsolicited email attachments because they could be malware in disguise or harmful in other ways.
	ii) Having the Ability To Test Out New Ideas Or Install Potentially Buggy Software Without Screwing Up Your Base System
- Develop Software/Build a Library for Other Platforms
- Having The Ability To Easily Delete And Recreate New Virtual Machines
- Having The Ability To Test Out Software In The Environment(s) They Were Made For
- Cost effective
	i) You could test out what your app or website would look like in different versions of Android on a Google Nexus phone. Or you could test what the app or website would look like in a Samsung Galaxy S6 and many other mobile phones, tablets, and devices which support Android.

To experiment with system level files of the OS, a VM would be a suitable test medium.(c)

10. **What reasons are there to NOT use a virtual machine for?**
A) Virtual machines are less efficient than real machines because they access the hardware indirectly. Running software on top of the host operating system means that it will have to request access to the hardware from the host. That will slow the usability.

11. **What is the primary use of the program "ssh"?**
A) To have a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfers, deployments. It uses RSA public prvate encryption.

12. **If you use the "chmod" command to set the permissions on a file to 777, what permissions are you setting on the file?**
A) Everyone can read, write, and execute(Everyone comprises of the owner, the member of the group and others)

13. **Define what "bash" is in Linux.**A) On linux, the interpreter that can understand the user commands and setting and pass them on to the computer for further processing is called a **shell**. A bourne again shell (Bash) is responsible for letting users configure a set of user preferences or settings.
A) Bash is default user shell in linux. It allows user to interact effectively with the cmd line. It includes features such as korn shell & C shell.

14. **What mechanism is used to search unstructured overlays networks?**
A) Flooding search
Broadcast is not allowed here

15. **What is the worst-case scenario for locating the object in an unstructured overlay network in terms of number of nodes visited, where n is the number of nodes?**
A) O(n)
O(n)-Unstructured network
O(logn)- Structured network
O(1) – Centralized system

16. **Can unstructured overlay networks have network partitions?**
A) Yes

17. **What is one advantage of a non-blocking form of Send? What about one advantage of blocking communication?**
A) Non-blocking can hide communication latency while blocking can be easier to implement.

18. **What type of applications should use an unbuffered communication approach?**
A) SSH server

19. **Is super-linear speedup is possible in practice? Explain your answer.**
A) C. True: More cores means more cache is accessible allowing memory latency operations to run faster

20. **T/F: Multi/many-core is hard for architecture people, but pretty easy for software people.**
A) False: Parallel processors put the burden of concurrency largely on the software side

21. **T/F: Multi/many-core made it possible for Google to search the web.**
A) False: Web search and other Google problems have huge amounts of data. The performance bottleneck becomes RAM amounts and speeds! (CPU-RAM gap)

22. **T/F: Email uses persistent communication**
A) True

23. **T/F: Network routers use transient communication**
A. True Ex: Phone call

24. **T/F: Asynchronous communication allows for more scalable systems**
A. True

25. **T/F: Synchronous communication allows for simpler implementation**
A. True

26. **T/F: Asynchronous communication can be implemented using synchronous communication**
A. False

27. T/F: Remote procedure call (RPC), remote method invocation (RMI), and web services (WS) are all abstractions to allow interprocess communication across a network to access remote resources.
A True

28. **T/F: Network sockets using TCP/IP are preferred over RPC/RMI/WS due to ease of use.**
A) False, RPC/RMI is easier to implement as compared to TCP/IP

29. **What are the key motivations for process migration?**-(same as 74)
A) • Performance –The overall system performance can be improved if processes are moved from heavily-loaded to lightly-loaded machines –Exploit parallelism, e.g., searching for information in the web through the development of mobile agent, that moves from site to site –fault tolerance, e.g., moving from failure prone to failure-free machines

• Flexibility –Dynamic configuration of distributed system –Clients don't need preinstalled software – download on demand

30. What is a disadvantage of process migration across cores of a processor?
A. I/O redirection: if a process does I/O to files or devices that are bound to a certain machine, there must be a way to redirect access to these resources even after the process migrated. This involves redirection of the I/O data stream over the network and has disadvantages concerning security, performance and reliability.

31. **What are problems with centralized scheduling**?
A) Fault Tolerance, Scalability

32. **What are problems with distributed scheduling**?
A) Synchronization / Coordination; load balance, replication

33**. What is work stealing? What parameters influence the scalability of work stealing most (we discussed this in detail in the lecture)?**
A) Work stealing is a scheduling algorithm which achieves an efficient dynamic load balancing.

It has many good assets:-1. It is a scalable distributed algorithm, the execution time is theoretically bounded & the number of steal attempts is theoretically bounded.
Parameters that influence scalability most are as follows:
1.Number of tasks to steal
2.Number of neighbour of a scheduler
Static Neighbours Dynamic neighbours
3.Polling Interval

34. **How is the namespace of a POSIX file system organized?**
A) in a hierarchy
POSIX - Portable Operating System Interface

35. **What is the best data structure to implement a POSIX file system metadata management that would minimize the memory/storage footprint and maximize performance?**
A) Hash Tables

36. **What does the domain name space service do?**
A) Maps names to IP addresses

37. **What is LDAP used for?**
A) LDAP stands for Light Weight Directory Access Protocol. It is an application protocol used over IP network to manage and access the distributed directory information service. The records are usually stored in hierarchical structure.

38. **Why would someone choose NFS(network file system) over GPFS(general parallel file system)?**
A) Simple to install, manage, and configure

39. **T/F: Clock synchronization is hard in distributed systems.**
A) True Each node in the distributed system has its own clock.

40. What is today's approach to clock synchronization in distributed systems?
- Lamport Logical Clocks
- PAXOS
- Network Time Protocol ———— correct
- System Time Protocol
- Domain Name Service
- POSIX

41. What are the advantages to logical clocks vs. real clocks?

- Cheaper to implement as a logical clock has fewer transistors in its implementation than a real clock
- Easier to implement as absolute time synchronization is relaxed to event ordering ———— correct
- More scalable since logical clocks have dedicated and reliable communication networks
- There are no advantages to logical clocks compared to real clocks

42. What is the primary difference between structured and unstructured overlay networks?
A) structured overlays impose constraints on the network topology for efficient object discovery, while unstructured overlays organize nodes in a random graph topology that is arguably more resilient to peer population transiency

A2) Generally an unstructured network has peers randomly connecting to some subset of all peers. This is nice and simple, but doesn't scale particularly well. The main problem being that you have to search the entire network to find something.   There's no way to get directly closer to your answer/destination, that would require structure. The result is more latency and overhead to find a particular pieces of data. So if 25% of all peers have what you are looking for, no problem. But if only 1 of a million peers has what you are looking for it might take awhile.

Structured networks on the other hand are organized to allow for searching efficiently. Ignoring churn, finding a piece of info among a million peers is straight forward. A DHT is a common example of a structured network. Each hop through the DHT gets you an order of magnitude closer to what you are searching for. As a result DHTs scale well into the millions. Churn does have to be delt with, usually by replicating information among many nodes.

43. **What mechanism is used to search unstructured overlays? What is the worst case scenario for locating the object in question in terms of number of nodes visited, where n is the number of nodes?**
A. Flooding and O(n)

44. **Can unstructured overlay networks have network partitions?**
A) Yes

45. **What are advantages of a non-blocking form of Send?**
A) 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.
A2: Non-blocking can hide communication latency while blocking can be easier to implement.

46**. Name some advantages of distributed systems over centralized systems.**
A) 1. Incremental growth - computing power can be added in small increments
2. Reliability - If one system crashes, the system as a whole can still survive
3. Speed - A distributed system have more total computing power than mainframe system
4. Open System - As an open system it is always ready to communicate with other systems
5. Economy - Microprocessors offer better price/performance than mainframes

47**. In many layered protocols, each layer has its own header. Surely it would be more efficient to have a single header at the front of each message with all of the control in it than to have so many separate headers. Why is this not done?**
A) Each layer has its own header since it must be independent for the layer, though this would be efficient to have one single header at the front of each message but this could destroy the transparency and make changes in the protocol of one layer visible to other layers. This is unwanted.
A2) Sathyaveer Karmarkar

48. In this problem you are to compare reading a file using a single-threaded file server with a multi-threaded file server. It takes 16 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming the data are in the block cache. If a disk operation is needed (assume a spinning disk drive with 1 head), as is the case one-fourth of the time, an additional 32 msec is required. What is the throughput (requests/sec) if a single threaded server is required, rounded to the nearest whole number? What is the throughput (requests/sec) if a multi-threaded server is required with 4-cores and 4-threads, rounded to the nearest whole number?
A) in the case of single thread, calculate the average probability of requests per millisecond
Average probability = (probability of no disk operation needed) * 16 + (probability of disk operation needed) * (16+32) = (¾) * 16 + (¼) * (16+32) = 24 requests / ms . This amounts to 41.67 requests / second.

incase of multi threaded for 4 threads, and each request takes 0.016 s to complete, then we have 250 requests / second on a multi threaded system. - Sathyaveer Karmarkar check it - Alex

49. **About how many cores/threads are expected to be in future commodity processors in the next 5 years?**
A) Future commodity processor will have 100+ cores provided by Tilera and currently the highest number of cores in a processor is 72 with 144 threads in Intel Xeon Phi.

50. **What is an advantage to blocking communication?**
A)Blocking communication is done using MPI_Send() and MPI_Recv(). These functions do not return (i.e., they block) until the communication is finished. Simplifying somewhat, this means that the buffer passed to MPI_Send() can be reused, either because MPI saved it somewhere, or because it has been received by the destination. Similarly, MPI_Recv() returns when the receive buffer has been filled with valid data.

In contrast, non-blocking communication is done using MPI_Isend() and MPI_Irecv(). These function return immediately (i.e., they do not block) even if the communication is not finished yet. You must call MPI_Wait() or MPI_Test() to see whether the communication has finished.

Blocking communication is used when it is sufficient, since it is somewhat easier to use. Non-blocking communication is used when necessary, for example, you may call MPI_Isend(), do some computations, then do MPI_Wait(). This allows computations and communication to overlap, which generally leads to improved performance.

Note that collective communication (e.g., all-reduce) is only available in its blocking version up to MPIv2. IIRC, MPIv3 introduces non-blocking collective communication.

non-blocking:
Advantages:
-- allows the separation between the initialization of the
communication and the completion.
-- can avoid deadlock
-- can reduce latency by posting receive calls early
• Disadvantages:
-- complex to develop, maintain and debug code

Blocking disadvantage:
: The major disadvantage if that if the kernel is single threaded, a system call of one thread will block the whole process and CPU may be idle during the blocking period.

51. In what situation would non-blocking communication improve performance most?
A) One can improve performance on many systems by overlapping communication and computation. This is especially true on systems where communication can be executed autonomously by an intelligent communication controller.

Non-blocking sends and receives
    – MPI_Isend & MPI_Irecv
    – returns immediately and sends/receives in background
Enables some computing concurrently with
communication
Avoids many common dead-lock situations
Also non-blocking collective operations in MPI 3.0

52. **What type of application should use an unbuffered communication approach?**
A) SSH Server

53. **What overheads are you limiting when using buffered communication?**
A. with communication using a buffer of predefined size, you don't have to worry about resizing the buffer.

54. **Name several unique features of the TCP communication protocol.**
A.
1) Multi-Vendor Support. TCP/IP is implemented by many hardware and software vendors. It is an industry standard and not limited to any specific vendor.

2) Interoperability. Today we can work in a heterogeneous network because of TCP/IP. A user who is sitting on a Windows box can download files from a Linux machine, because both Operating Systems support TCP/IP. TCP/IP eliminates the cross-platform boundaries.

3) Logical Addressing. Every network adapter has a globally unique and permanent physical address, which is known as MAC address (or hardware address). The physical address is burnt into the card while manufacturing. Low-lying hardware-conscious protocols on a LAN deliver data packets using the adapter's physical address. The network adapter of each computer listens to every transmission on the local network to determine whether a message is addressed to its own physical address.

For a small LAN, this will work well. But when your computer is connected to a big network like internet, it may need to listen to millions of transmissions per second. This may cause your network connection to stop functioning.

To avoid this, network administrators often segment (divide) big networks into smaller networks using devices such as routers to reduce network traffic, so that the unwanted data traffic from one network may not create problem in another network. A network can be again subdivided into smaller subnets so that a message can travel efficiently from its source to the destination. TCP/IP has a robust subnetting capability achieved using logical addressing. A logical address is an address configured through the network software. The logical addressing system used in TCP/IP protocol suit is known as IP address.

4) Routability. A router is a network infrastructure device which can read logical addressing information and direct data across the network to its destination.TCP/IP is a routable protocol, which means the TCP/IP data packets can be moved from one network segment to another.

5) Name Resolution. IP addresses are designed for the computers and it is difficult for humans to remember many IP addresses. TCP/IP allows us to use human-friendly names, which are very easy to remember (Ex. www.omnisecu.com). Name Resolutions servers (DNS Servers) are used to resolve a human readable name (also known as Fully Qualified Domain Names (FQDN)) to an IP address and vice versa.

6) Error Control and Flow Control.The TCP/IP protocol has features that ensure the reliable delivery of data from source computer to the destination computer. TCP (Transmisssion Control Protocol) defines many of these error-checking, flow-control, and acknowledgement functions.

7) Multiplexing/De-multiplexing. Multiplexing means accepting data from different applications and directing that data to different applications listening on different receiving computers. On the receiving side the data need to be directed to the correct application, for that data was meant for. This is called De-multiplexing. We can run many network applications on the same computer. By using logical channels called ports, TCP/IP provides means for delivering packets to the correct application. In TCP/IP, ports are identified by using TCP or UDP port numbers.
8) Three way Handshake

55.**Name a technique that can be used to handle out of order packets efficiently?**
A) **By using sliding window protocol.**
TCP uses Sliding Window Protocol (SWP) for segments flow control and segment level error control. It is a connection oriented communication. SWP is a theoretical model practically implemented as stop and wait, go back n(GBN), selective repeat protocol. Depends on technique it depends. If TCP connection uses SR protocol then duplicate packets are simply dropped and out of order packets are stored in window after that it sends ack to other machine saying that selectively send packets. Coming to Stop and Wait and GBN duplicate packets and out of order packets both are dropped sends ack for the next packet to be received.

TCP functionality called Flow Control.
- For out of order segments, each TCP segment has an identifier for the sequence of that segments amongst others in its group. This identifier called a sequence number can be used to reorder packets

56. **Identify the function that cannot be used to implement a concurrent server?**
A) Please review
A concurrent server handles multiple clients at the same time. The simplest technique for a concurrent server is to call the fork function, creating one child process for each client. An alternative technique is to use threads instead

For example, a chat server may be serving a specific client for hours—it cannot wait till it stops serving a client before it serves the next one.

We use below functions:
socket()
connect()
accept()
fork() — Important
bind(listenfd, ...);
listen(listenfd, ...);
server(child)
server(parent)
close()

57. ~~What does it mean for a system to be scalable?~~ (same as 94)

58. **Identify which system has a client-pull architecture.**
A. http , https.
A web browser requests a web page

59. Identify which system has a server-push architecture.
A. video streaming
An email server transmits an email to an email client, notification

60. **Which is not an OS state for a thread?**
A. The four states of a thread are ready, running and waiting.

61. **Why is threading useful on a single-core processor?**
A) Gives us multi tasking capability and ability to efficiently use system resources by pipelining tasks rather than having to wait for a whole task to complete before assigning another task

62. Identify what a thread has of its own (not shared with other threads):
A) private stack
B) own set of registers
A thread is a lightweight process and forms the basic unit of CPU utilization. A process can perform more than one task at the same time by including multiple threads.

- A thread has its own program counter, register set, and stack
- A thread shares resources with other threads of the same process the code section, the data section, files and signals.

63. **What is the advantage of OpenMP over PThreads?**
A)
Pthreads and OpenMP represent two totally different multiprocessing paradigms.

Pthreads(POSIX Threads) is a very low-level API for working with threads. Thus, you have extremely fine-grained control over thread management (create/join/etc), mutexes, and so on. It's fairly bare-bones.

On the other hand, OpenMP has,
Its cross platform, and simpler for some operations
OpenMP is much higher level, is more portable and doesn't limit you to using C.
It's also much more easily scaled than pthreads.

One specific example of this is OpenMP's work-sharing constructs, which let you divide work across multiple threads with relative ease. (See also Wikipedia's pros and cons list.)

Finally,
If your library has thread tools (almost always built on some flavor of PThread), USE THOSE. If you are a library developer, spend the time (if possible) to build them. It is worth it- you can put together much more fine-grained, advanced threading than OpenMP will give you.

Conversely, if you are pressed for time or just developing apps or something off of 3rd party tools, use OpenMP. You can wrap it in a few macros and get the basic parallelism you need.

64. **Do more threads always mean better performance?**
A) no, threads require memory to setup their own private stack, too many threads initialized may cause a performance degrade. In addition, there will be more communication among threads which results in overhead.

65**. What was the flaw in Amdahl’s Law that made it inaccurate?**
A) They calculated overall speedup assuming the size of the data set remains constant. Where as, if you increase the number of cores , it is likely you will also increase your dataset.

66. **How does RPC/RMI/WS make improvements over sockets/threads? Professor**
A) Whenever you implement RMI/RPC/WS programs they have api which implements sockets and threads. You explicitly do not have to write code for the same. Also whenever server decides to connect to a client, a new thread is assigned to their connection and in this way multiple connections are possible at the same time.

67. **What is the purpose of parameter marshaling?**
A) The stub resides on the client machine, not on the server. It knows how to contact the server over the network. The stub packages the parameters used in the remote method into a block of bytes. **The process of encoding the parameters is called parameter marshalling. The purpose of parameter marshalling is to convert the parameters into a format suitable for transport from one virtual machine to another**. In the RMI protocol, objects are encoded with the serialization mechanism described in Chapter 1. In the SOAP protocol, objects are encoded as XML.

68. **What is an advantage to a connectionless protocol compared to a connection-oriented protocol?**
A) in applications of broadcast or real time communication, connectionless protocol provides a closer experience to what we know as “real-time”. No acks are sent back and forth and so it is faster, any lost packets are simply retransmitted.

69. **How are web services (WS) similar to RMI/RPC?**
A. Remote procedure call (RPC), remote method invocation (RMI), and web services (WS) are all abstractions to allow interprocess communication across a network to access remote resources.

70. **What form of communication is email considered?**
A. persistent asynchronous communication

71. **What form of communication is UDP considered?**
A) transient asynchronous communication.

72. **What form of a data stream is needed to handle archival video streaming?**
A) udp

73. **What form of a data stream is needed to handle live 2-way audio streaming?**
A) udp data stream

74. **What is the purpose of code migration?**
A) moving processes from a heavily utilized server to a more vacant server to balance the loads among a cluster. This comes useful in regards to energy consumption overall and server performance as a whole
Shipment of server code to client – filling forms (reduce communication, no need to pre-link stubs with client) – Ship parts of client application to server instead of data from server to client (e.g., databases) – Improve parallelism – agent-based web searches

75**. What is the purpose of the scheduler in a distributed system?**
A) to assign work to nodes connected in the distributed system

76. You have a cluster with 1000 compute nodes. You have a centralized scheduler that can schedule 10 tasks per second. What is the smallest granularity of task lengths that your scheduler can support in order to achieve high system utilization?
A) will be 100s before everyone gets 1 piece of work. If your work granularity is 100s then your system utilization will be high, because after the work is done being processed, another piece of work is assigned to a node.

77. You have a cluster with 1000 compute nodes. You have a distributed scheduler that has 1000 schedulers, and each scheduler can process 1 task per second. What is the smallest granularity of task lengths that your scheduler can support in order to achieve high system utilization?
A) 1 second

78. In distributed scheduling, why are dynamic neighbors better than static ones?
A) Number of Dynamic Neighbours are square root of all schedulers and Number of Static neighbours are eighth of all schedulers.

79. A user is in front of a browser and types in www.google.com, and hits the enter key. Think of all the protocols that are used in retrieving and rendering the Google logo and the empty search box. Select all that could apply (partial credit will be given).
A. Http protocol ,DNS,

80. Identify which algorithm is suitable for generating a unique name across a distributed system (hint: these operations would be done on each node, and do not involve any network communication)?
Ans. GUID

81. **Why is time synchronization hard?**
A) once you synchronize 2 devices, there is no guarantee that they will remain synced, because of different quality crystals vibrating at different rates, synchronization typically happens every so often and the time of the fastest crystal propagates to the devices running slower clocks.

82. **What makes time synchronization inaccurate?**
A) 2 devices may show the same time accurate to the minute, but their seconds could very much be out of sync. The vibration of devices also vary depending on the quality of the crystal inside the clock which introduces clock skew.
A2) Every system will have its own clock, whenever we try to sync 2 clocks there will atleast be a delay of few millisecs

83. Name a protocol used to synchronize time.
A) Christian’s Algorithm
Or Berkeley Algorithm

84. **When synchronizing clocks, why is it a bad idea to change clocks infrequently a large amount?**
A. The process of synchronizing clocks has a few ms error bcus of the latency btn the time server and the requesting system.

85. **What makes logical clocks more scalable than time synchronization?**
A) When you have a network broadcasting a virtual heartbeat to all its connected nodes, you don't need to care if their actual time is correct as long as all the nodes receive the same logical time to carry out time sensitive tasks.

A2. If you carry out time synchronization there has to be a network which will broadcast the time to its connected nodes, whereas when it comes to logical clocks , there is no need of synchronization as the process are carried out upon understandibility of logical time.

86. **What are the advantages of centralized locks?**
A) Centralized lock manager maintains information on which process have requested to enter the critical section, which process have been granted access. Centralized locks assures mutual exclusion (i.e., only one process is granted access at a time and no other process can enter its critical section until the first process finishes and sends a release message.

87. **What makes distributed locks hard to implement?**
A. Paxos is the gold standard in consensus algorithms. It is a distributed consensus protocol (or a family of protocols if you include all its derivatives) designed to reach an agreement across a family of unreliable distributed processes. The reason this is hard is because the participating processes can be unreliable due to failures (e.g., server failure, datacenter failure or network partitions) or malicious intent

88. **How is replication different than caching?**
A) Replication usually involves storing data in memory across many locations. Caching is a temporary space where frequently requested data is provided. In caching memory is volatile and is wiped if the server goes offline or if it is overwritten.

89. **Why do we need replication/caching?**
A) to improve the accessibility of our data on a global scale, as well as to have a recovery medium for disaster recovery.
Caching is used to retrieve the data rapidly, so you don’t have to access primary memory.

90. **What guarantees does eventual consistency give?**
A) That eventually all copies of the data will be synchronized from every geographical node you may access it from.

91. **Which operation(s) are most difficult in eventual consistency?**

A) Update/Write operation
This weak form of consistency does not restrict the ordering of operations on different keys in any way, thus forcing programmers to reason about all possible orderings and exposing many inconsistencies to users

92. **Why did processors from the 1980s not need cache-coherent processors?**
A) In the 1980's, multiprocessors were designed with two major architectural approaches. For small numbers of processors (typically less than 16 or 32), the dominant architecture was a single shared memory with multiple processors, interconnected with a bus. These machines were called bus-based multiprocessors or symmetric multiprocessors (SMP's), since all processors have an equal relationship with the centralized main memory.
To scale to larger numbers of processors, designers distributed the memory throughout the machine and used a scalable interconnect to enable processor–memory pairs to communicate.
The use of multiple memory modules and a scalable network allows the machine to scale to larger numbers of processors. In addition, because the memory is distributed with the processors, local memory accesses do not consume global bandwidth and can achieve lower access times. The shared memory architectures supported the traditional programming model, which saw memory as a single, shared address space. The shared memory machines also had lower communication costs since the processors communicated through shared memory rather than through a software layer. As a reason these processors did not need cache-coherent processors.

93**. Why is it sometimes so hard to hide the occurrence and recovery from failures in a distributed system?**
A) based on the size of the system at hand, it is hard to conclude whether a system is down or is simply facing a bottleneck. At times, the system may be unresponsive due to network traffic or low memory.

94. **Describe precisely what is meant by a scalable system.**
A) A system is scalable with respect to its number of internal components , or number of nodes on a geographical scale. It will be scalable if it can grow without a sizable loss in performance metrics

95. Consider again an unstructured overlay network in which every node randomly chooses c neighbors. To search for a file, a node floods a request to its neighbors and requests those to flood the request once more. How many nodes will be reached?
A.) An easy upper bound can be computed as c × (c −1), but in that case we ignore the fact that neighbors of node P can be each other's neighbor as well. The probability q that a neighbor of P will flood a message only to non neighbors of P is 1 minus the probability of sending it to at least one neighbor of P: q = 1 − c−1 k=1 Σ   (c −1 k)   (c/ N −1)^k (1 − c/N −1)^ c−1−k In that case, this flooding strategy will reach c × q (c −1) nodes. For example, with c = 20 and N = 10, 000,aquery will be flooded to 365.817 nodes

96. Suppose that you could make use of only transient asynchronous communication primitives, including only an asynchronous receive primitive. How would you implement primitives for transient synchronous communication?
http://www.leonardomostarda.net/ds/7-communicationforProf.pdf
A) Consider a synchronous send primitive. A simple implementation is to send a message to the server using asynchronous communication, and subsequently let the caller continuously poll for an incoming acknowledgment or response from the server. If we assume that the local operating system stores incoming messages into a local buffer, then an alternative implementation is to block the caller until it receives a signal from the operating system that a message has arrived, after which the caller does an asynchronous receive.

97. **How could you guarantee a minimum end-to-end delay when a collection of computers are organized in a (logical or physical) ring?**
A) Strangely enough, this is much harder than guaranteeing a maximum delay. The problem is that the receiving computer should, in principle, not receive data before some elapsed time. The only solution is to buffer packets as long as necessary. Buffering can take place either at the sender, the receiver, or somewhere in between, for example, at intermediate stations. The
best place to temporarily buffer data is at the receiver, because at that point there are no more unforeseen obstacles that may delay data delivery. The receiver need merely remove data from its buffer and pass it to the application using a simple timing mechanism. The drawback is that enough buffering capacity needs to be provided.

A2) https://www.ppgia.pucpr.br/~alcides/Teaching/SistemasDistribuidos/Tanenbaum/DSTanenbaumsolutions.pdf - 24th question in that PDF
We let a token circulate the ring. Each computer is permitted to send data across the ring (in the same direction as the token) only when holding the token. Moreover, no computer is allowed to hold the token for more than T

seconds. Effectively, if we assume that communication between two adjacent computers is bounded, then the token will have a maximum circulation time, which corresponds to a maximum end-to-end delay for each packet sent.

98. **Would it make sense to limit the number of threads in a server process?**
A.) It makes sense to limit the threads in a server process. Below are the few points to justify for limiting the threads:
1. The threads have their own stack and hence they require memory.
Having multiple threads will consume a lot of memory and will certainly affect other threads and the viability of the process itself.
2. Having too many threads can be chaotic. It will generally delay the processing of requests as you need to wait for the thread to become available. The data will have to be exchanged between threads and they have overheads like context switches.
3. In a virtual memory system, it may be difficult to build a relatively stable working set, and hence there will be page faults. This might lead to performance degradation and hence page thrashing

A2) Yes, for two reasons. First, threads require memory for setting up their ownprivate stack. Consequently, having many threads may consume too muchmemory for the server to work properly. Another, more serious reason, is that,to an operating system, independent threads tend to operate in a chaotic man-ner. In a virtual memory system it may be difficult to build a relatively stableworking set, resulting in many page faults and thus I/O. Having many threadsmay thus lead to a performance degradation resulting from page thrashing.Even in those cases where everything fits into memory, we may easily see thatmemory is accessed following a chaotic pattern rendering caches useless.Again, performance may degrade in comparison to the single-threaded case

A3) Yes, threads require memory to setup their own private stack. So having many threads may consume too much memory of the server for its other tasks which may ultimately result in performance degradation.

99. **Constructing a concurrent server by spawning a process has some advantages and disadvantages compared to multithreaded servers. Please explain**.
A) Separate processes by nature do not share data or can be said to be isolated from each other, this is an advantage as we don’t need to worry about 2 programs modifying the same data. Don’t need to worry about implementing locks. Process spawning however is more costly compared to assigning threads and having processes communicate with each other may also be costly.

A2) The server can be iterative i.e., it repeats through each client and serves one request at a time. Alternatively, a server can handle numerous clients at the same time in analogous, and this type of a server is called concurrent server.
Having separate processes has its advantages. They are protected against each other which may prove to be necessary like in the case of a supersaver handling completely independent services. The process spawning is a costly operation which can be saved by using multithreaded servers instead. If the processes need to communicate, then using threads will be cheaper and in many cases and we can avoid having the kernel implement the communication.

A3) http://www.cs.wichita.edu/~chang/lecture/cs843/homework/hwk3-sol.txt

An important advantage is that separate processes are protected against each other, which may prove to be necessary as in the case of a superserver handling completely independent services.

On the other hand, process spawning is a relatively costly operation that can be saved when using multithreaded servers. Also, if processes do need to communicate, then using threads is much cheaper as in many cases we can avoid having the kernel implement the communication.

100.**Explain the difference between a hard link and a soft link in UNIX systems. Are there things that can be done with a hard link that cannot be done with a soft link or vice versa?**
A) hard link is a named entry in the directory pointing to a file possibly in another directory. Both files will contain identical contents. There is a two way pointer mechanism implemented with hard links. With soft links, you can link to different disk partitions or even to a different machine, but if the file the link is pointing to gets deleted, the link does not automatically get updated and may return an error message when trying to access.

101.Consider a distributed file system that uses per-user name spaces. In other words, each user has his own, private name space. Can names from such name spaces be used to share resources between two different users?
A) If the names in the per-user name spaces can be resolved to names in a shared global namespace, then Yes it is possible. 2 identical names in different name spaces should be independent and may refer to different entities.

102.Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what is the maximum clock skew that will occur?

A) 1000 ticks / ms = 1,000,000 ticks /s
990 ticks / ms = 990,000 ticks /s. Approx 10,000 ticks slower.
10k ticks is the work done in approx 10ms. Meaning per second, we have lag of 10ms which equates to approx 600ms skew time difference per minute.

103.One of the modern devices that have (silently) crept into distributed systems are GPS receivers. Give examples of distributed applications that can make use of GPS information.
A) weather forecast systems, car navigation equipment. Tracking aircrafts in transit

104.When a node synchronizes its clock to that of another node, it is generally a good idea to take previous measurements into account as well. Why? Also, give an example of how such past readings could be taken into account.
A) there may be an error in the current reading. One possibility is to take the last N values and compute a median or average. If the measured value falls outside the current interval, it is disregarded.
A2) https://www.distributed-systems.net/my-data/ds2/ds-solutions.pdf
The obvious reason is that there may be an error in the current reading. Assuming that clocks need only be gradually adjusted, one possibility is to consider the last N values and compute a median or average. If the measured value falls outside a current interval, it is not taken into account (but is added to the list). Likewise, a new value can be computed by taking a weighted average, or an aging algorithm

105.Explain in your own words what the main reason is for considering relaxed consistency models.
A) works well on a large scale. Better scalability than strong consistency models. Don't need a lot of coordination between nodes.

106.What kind of consistency would you use to implement an electronic stock market? Explain your answer.
A) Causal consistency should suffice for our need. Because reaction to change in stock will only take place after the actual change occurs.

107.Consider a personal mailbox for a mobile user, implemented as part of a wide-area distributed database. What kind of client-centric consistency would be most appropriate?
A) As long as the owner sees the same mailbox, it doesn't matter which protocol is implemented. The simplest to implement however would be a monotonic write, where the primary copy is always located on the user's device.
A2) no matter whether he is reading or updating it. In fact, the simplest implementation for such a mailbox may well be that of a primary-based local-write protocol, where the primary is always located on the user's mobile computer.

108.**Name a few advantages and disadvantages of using centralized servers for key management**.
A) Advantage could be simplicity of key management, efficient storage and maintenance at a single site. Disadvantage could be a bottleneck in terms of availability, and if the server gets compromised, then new keys will have to be issued.

109.How does the probability of failure of an entire distributed system (assuming all components are necessary to function properly) change as the number of independent components in the system grows?
A) Hamza

110.What components in a computer system do we know how to make resilient, and what technique is

used?
A) we know how to make data storage resilient. Techniques such as forward recovery and backwards recovery are used to implement such resilience.
HardDrives - RAID
Power supply - Redundant power supply
Processors - Extra cores
Memory - ECC

111.In the Two Army Problem and assuming an asynchronous unreliable communication channel, can the two generals ever reach consensus with 100% guarantee?
A) no safe agreement is possible, event without fault. Messages must be acknowledged with guarantee of no alteration during message transmission

112.Data resilience through forward error correcting codes is an example what type of recovery mechanism? What is the advantage of Forward Recovery in storage systems?
A) is an example of forward recovery mechanism, no time is lost in data processing and operations can resume as normal once recovery takes place.

113.Data resilience through replication is an example what type of recovery mechanism? What are some advantages of this recovery mechanism?
A) is an example of backwards recovery mechanism, it is inexpensive to implement, and it is independent of an arbitrary fault.

114.RAID (redundant array of inexpensive disks) is an example of what type of recovery mechanism?
A) forward recovery using forward error correcting

115.What is the technique called that allows applications to restart and recover from an intermediary point after the start of the application?
A) checkpoint restart recovery

AOS Cheat sheet:
Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond, however one of them ticks 1010 times per millisecond and the other ticks 990 times per millisecond. If clock synchronization updates occur once every two minutes, what is the maximum clock skew that will occur between the two machines? How frequently must these clocks be synchronized to limit clock skew to 200 milliseconds?
(1.5) Maximum clock skew = [1010-990] * 0.001 * 120 seconds = 2.4 seconds. The clocks should be synchronized every 10 seconds to limit the clock skew to a maximum of 200 milliseconds [0.02 x 10 seconds = 0.2 seconds].

Virtualization allows an application, and possibly also its complete environment including the operating system, to run concurrently with other applications, but highly independent of the underlying hardware and plat- forms, leading to a high degree of portability. Moreover, virtualization helps in isolating failures caused by errors or security problems. It is an important concept for distributed systems, and we pay attention to it in a separate section.

Code:

Server:

```
import java.io.*;
import java.util.*;
import java.util.concurrent.Executors;
import java.util.concurrent.ThreadPoolExecutor;
import java.net.*;

public class get {
	static TreeMap<String, String> sortedKey = new TreeMap<>();
	public static HashMap<String, ArrayList<String>> filemap = new HashMap<String, ArrayList<String>>();

	public static void main(String[] args) throws IOException {
		ServerSocket myServerSocket = new ServerSocket(5005);
		ThreadPoolExecutor workerpool = (ThreadPoolExecutor) Executors.newCachedThreadPool();
		while (true) {
			Socket skt = null;
			try {

				skt = myServerSocket.accept();
				System.out.println("A new client is connected : " + skt);

				ObjectInputStream objectInput = new ObjectInputStream(skt.getInputStream());
				ObjectOutputStream objectOutput = new ObjectOutputStream(skt.getOutputStream());

				System.out.println("\nAssigning new thread for this client");

				PeerHandler clientrequest = new PeerHandler(skt, objectInput, objectOutput);
				workerpool.execute(clientrequest);

			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
}

class PeerHandler implements Runnable {
	final ObjectInputStream ois1;
	final ObjectOutputStream oos1;
	final Socket s1;

	public PeerHandler(Socket s, ObjectInputStream ois, ObjectOutputStream oos) {
		this.s1 = s;
		this.ois1 = ois;
		this.oos1 = oos;
	}

	public void run() {
		try {
			ArrayList<String> titleList = new ArrayList<String>();
			Object object = ois1.readObject();
			boolean search;
			if(object instanceof String) {
				search = true;
			} else {
				search = false;
			}

			if (search == false) {
				titleList = (ArrayList<String>) object;
				for (int i = 0; i < titleList.size(); i++) {
					ArrayList<String> titles = get.filemap.get(titleList.get(i));
					if (titles != null) {
						titles.add(s1.getInetAddress().toString().substring(1));
					}
					else {
						ArrayList<String> ipaddresses = new ArrayList<String>();
```

```
                        ipaddresses.add(s1.getInetAddress().toString().substring(1));
                        get.filemap.put(titleList.get(i), ipaddresses);
                    }

                }
                System.out.print(get.filemap);

            }
            else {
                System.out.println("A new client is connected : " + s1);
                System.out.println("Searching file in get");
                String fileName = (String) object;
                ArrayList<String> ipaddresses = get.filemap.get(fileName);
                this.oos1.writeObject(ipaddresses);
                this.oos1.flush();
                System.out.println("output>>>>>>>> " + String.valueOf(ipaddresses));

            }

        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}
```

Client:

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;

public class read {
    /*1. GIVE path of files of current peer to run in normal mode.

    2. For running in Weak Scaling and Strong scalling mode, give 1 plus another path of peer 2 with space in
between.


    */

    //public final String STORE_PATH="/home/sathyaveer27/Desktop/cs550_1/data4/";

    public static void main(String[] args) throws IOException, ClassNotFoundException {

        try {

            File f = new File(args[0]);
            Socket socket = new Socket(args[2], 5005);

            ObjectOutputStream objectOutput = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream objectInput = new ObjectInputStream(socket.getInputStream());
            ArrayList<String> names = new ArrayList<String>(Arrays.asList(f.list()));
            System.out.println(names.toString());

            objectOutput.writeObject(names);
				socket.close();

			} catch (Exception e) {
				e.printStackTrace();
			}

			Thread clientSocketThread = new Thread() {
				public void run() {
					ServerSocket clientPeerSocket = null;
					try {
						clientPeerSocket = new ServerSocket(5006);
						while (true) {
							Socket socket = clientPeerSocket.accept();

							ObjectInputStream fileInput = new
ObjectInputStream(socket.getInputStream());
							String fileName = fileInput.readUTF();
							File f = new File(args[0], fileName);
							//System.out.println("Sending File " + fileName);
							long abc = f.length();
							byte[] byteslen = new byte[(int) abc];

							FileInputStream fileInputStream = new FileInputStream(f);
							BufferedInputStream bufInputStream = new
BufferedInputStream(fileInputStream);
							bufInputStream.read(byteslen, 0, byteslen.length);
							OutputStream outputStream = socket.getOutputStream();

							outputStream.write(byteslen, 0, byteslen.length);
							outputStream.flush();

							outputStream.close();
							fileInputStream.close();
							bufInputStream.close();
							socket.close();
							//clientPeerSocket.close();
							}

					} catch (IOException e) {
						e.printStackTrace();
					}
				}
			};
			clientSocketThread.start();

			Thread clientHandler = new Thread() {
				public void run() {
					String toreturn;
					String filename;

					while (true) {
						try {

							System.out.println("Press to Select \n" + "1. Search a File\n" + "2. Exit\n"
+ "3.Weak_Scalling\n" + "4.Strong Scaling");

							int received = new Scanner(System.in).nextInt();
					if (received == 1) {
						//clientSocketThread.start();
						System.out.print("Enter file name: ");
						String fileName = new Scanner(System.in).nextLine();
						Socket socket = new Socket(args[2], 5005);

						ObjectOutputStream dos = new
ObjectOutputStream(socket.getOutputStream());

						dos.writeObject(fileName);
						dos.flush();
						ObjectInputStream dis = new
ObjectInputStream(socket.getInputStream());

						ArrayList<String> list = (ArrayList<String>) dis.readObject();
						System.out.println(list);
						//to download
						for (int i = 0; i < list.size(); i++) {
						ObjectOutputStream objectOutput = null;
						FileOutputStream fos = null;
						InputStream inputStream = null;
						BufferedOutputStream bufferedOutputStream = null;
						try {
							Socket s1 = new Socket(list.get(i), 5006);
							objectOutput = new
ObjectOutputStream(s1.getOutputStream());

							objectOutput.writeUTF(fileName);
							objectOutput.flush();
							File x = new File(args[0],fileName);
							if (!(x.exists()))
							{
								x.createNewFile();
							}
							inputStream = s1.getInputStream();
							System.out.println("Reciving File" + fileName);
							fos = new FileOutputStream(args[0] + fileName);
							bufferedOutputStream = new BufferedOutputStream(fos);

							byte[] array = inputStream.readAllBytes();
							bufferedOutputStream.write(array, 0, array.length);
							bufferedOutputStream.flush();
							//fos.close();
						} catch (IOException e) {
							e.printStackTrace();
						} finally {
							try {
								objectOutput.close();
								fos.close();
								bufferedOutputStream.close();
								inputStream.close();
								//objectOutput.close();
								//s1.close();
							} catch (IOException e) {
								e.printStackTrace();
							}
						}
						}
					}else if (received == 2) {
```

```
                                        System.exit(0);




                                } else if (received == 3) {
                                        double CONVERTER = 1_000_000_000.0;
                                        File f = new File(args[1]);
                                        ArrayList<String> names1 = new
ArrayList<String>(Arrays.asList(f.list()));
                                        String fileName;
                                        long start = System.nanoTime();
                                        for (int i = 0; i < names1.size(); i++) {
                                                Socket socket = new Socket(args[2], 5005);
                                                ObjectOutputStream dos1 = new
ObjectOutputStream(socket.getOutputStream());
                                                fileName = names1.get(i);
                                                dos1.writeObject(fileName);
                                                dos1.flush();
                                                ObjectInputStream disi = new
ObjectInputStream(socket.getInputStream());
                                                ArrayList<String> list1 = (ArrayList<String>)
disi.readObject();
                                                disi.close();
                                                dos1.close();
                                                socket.close();
                                                System.out.println(list1);

                                        }
                                        long end = System.nanoTime();
                                        double Diff = (end - start) ;//  SECS_CONVERTER;
                                        Diff = Diff/CONVERTER;
                                        Diff = names1.size() / Diff;
                                        System.out.println("Total Throughput " + Diff+"/secs");

                                } else if (received == 4) {
                                        File f = new File(args[1]);
                                        ArrayList<String> names1 = new
ArrayList<String>(Arrays.asList(f.list()));
                                        String fileName;
                                        double CONVERTER = 1_000_000_000.0;
                                        long start = System.nanoTime();
                                        for (int i = 0; i < names1.size(); i++) {
                                                Socket socket = new Socket(args[2], 5005);
                                                ObjectOutputStream dos1 = new
ObjectOutputStream(socket.getOutputStream());
                                                fileName = names1.get(i);
                                                dos1.writeObject(fileName);
                                                dos1.flush();
                                                ObjectInputStream disi = new
ObjectInputStream(socket.getInputStream());
                                                ArrayList<String> list1 = (ArrayList<String>)
disi.readObject();
                                                System.out.println(list1);
                                                //TO Download
                                                for (int j = 0; j < list1.size(); j++) {

                                                        ObjectOutputStream objectOutput = null;
```

```
											FileOutputStream fos = null;
											InputStream inputStream = null;
											BufferedOutputStream bufferedOutputStream =
null;
											try {
												Socket s1 = new Socket(list1.get(j), 5006);
												objectOutput = new
ObjectOutputStream(s1.getOutputStream());
												objectOutput.writeUTF(fileName);
												objectOutput.flush();

												inputStream = s1.getInputStream();
												fos = new FileOutputStream(args[0] +
fileName);
												bufferedOutputStream = new
BufferedOutputStream(fos);
												System.out.println("Recieving File" +
fileName);
												byte[] array = inputStream.readAllBytes();
												bufferedOutputStream.write(array, 0,
array.length);
												bufferedOutputStream.flush();
												fos.close();
												dos1.close();
												disi.close();
												socket.close();
											} catch (IOException e) {
												e.printStackTrace();
											} finally {
												try {
													objectOutput.close();
													fos.close();
													bufferedOutputStream.close();
													inputStream.close();
													//objectOutput.close();
													//s1.close();
												} catch (IOException e) {
													e.printStackTrace();
												}
											}
										}

									}
									long end = System.nanoTime();
									double Diff = (end - start) ;
									Diff = Diff/CONVERTER;
									Diff = names1.size() / Diff;
									System.out.println("Total Throughput " + Diff+"/secs");

								}
							}
							catch (Exception e) {
								e.printStackTrace(); }
							}
						}

					};
			clientHandler.start();
}}
```