Questions for AFS/NFS:

1. *Briefly explain how caching is used to improve the performance of NFS (or AFS).*

   NFS caches data blocks to improve performance. Before using a data block, it will check whether this data block is still up to date. NFS uses time-out mechanism and attribute-block checking to decide whether to use the cached data blocks.

   AFS caches the whole file. After the first open, later file accesses will get local-disk access throughput. In later versions of AFS, call-back mechanism is used to determine whether a file needs to be discarded and fetched again from the server.

Questions for GFS:

1. *What did GFS do to make sure the master node does not become a performance bottle-neck?*

   1) Data transfer during read and write requests does not go through the master node. Instead, the client only contacts the master node to fetch the chunk-server information. After that, the client directly contacts the chunkservers to conduct read and write.

   2) The client will cache some chunkserver information, and does not always ask master for the chunk-server location.

Questions for RAID:

Compare the performance and fault tolerance capability between RAID 3 and RAID 5.

RAID3 uses bit/byte level parity (each data block spreads across all data disks)

RAID5 uses rotational parity

Performance:

RAID3 and RAID5 have the same amount of sequential read/write throughput.

RAID5 has better random read throughput. Reason 1: RAID5 only needs to read one disk to obtain the target data block. In RAID3, all data disks need to be read in order to obtain one data block. Reason 2: RAID5 does not have a dedicated parity disk. Therefore, all disks in the system can serve random read requests.

RAID5 has better random write throughput, Reason 1: RAID3 uses a dedicated parity disk. Every random write will require the system to read and write this parity disk. As a result, no parallelism can be achieved within the disk array. RAID5 can conduct G/2 random writes at the same time, because there is no dedicated parity disk in the system (G is the total number of disks in RAID5). 2. Since RAID3 spreads the data of each block across disks, there is an extra `synchronization' overhead for RAID3.

Reliability: Both RAID3 and RAID5 can survive one disk failure, no more.

Questions for MapReduce:

Discuss how Map-Reduce systems handle node failures:

A crash of a completed mapper task will cause the mapper task to be re-executed and all the reducers will be notified; a crash of a completed reducer task will not cause any reexecution; a crash of an on-going mapper or reduce task will cause the task to be executed from beginning on a notehr working node.

A crash of a master will cause the whole mapreduce job to reexecute.

Questions for Bugs:

1. *Briefly explain how to inject and execute arbitrary code through stack overflow bugs.*

   Briefly speaking, the attacker will (1) put the attacking code into the problematic stack buffer (the code is usually an *execv* call to start running /bin/shell); (2) rewrite the stack word where the current function's return-address was stored with the starting address of the attacking code. Ideally, when the program exits the current function, control flow will transfer to the attacking code.

   There are several tricky issues involved in the above process: (1) the attacking code sequence cannot contain `\0', a character denoting the end of the string; (2) how to inject and locate a constant string needs some effort; (3) NOP instructions need to be padded before the real attacking code to improve the chance of `safe' control-flow change from function return to the attacking code. …

1. *Name one weakness of the `deviant' based bug detection.*

   (1) It requires a large sample set to achieve statistical confidence. (2) It is based on the assumption that `the majority is correct', which could be wrong.

Questions for FFS:

1. *What are the pros and cons of using large blocks.*

   + .More data in each file can be represented by direct data pointers in the i-node;

   + More sequential read, less seek/rotational delay, higher throughput;

   - waste disk space, especially when most files are small; .

2. *Why does LFS have better writer performance than FFS?*

   LFS does not update a block in its original location in disk; instead, all updates are organized and buffered in a segment, and then written to disk one segment at a time. Since LFS has turned random writes to sequential writes, its write-access performance is much better than FFS.

Questions for Xen

X86 does not trap all priviledged instructions when they are executed in non-kernel mode. Xen solves this problem by rewriting guest OS to avoid issuing such priviledged instructions.

X86 uses hardware to handle TLB misses (i.e., hardware will walk through the page table to resolve a TLB miss). Xen makes guest OS be aware of the fact that it does not own the whole virtual address space. Consequently, the page table maintained by OS actually stores virtual to machine address translation, different from that in Disco.

Questions for Disco

Physical address is what guest OS considers the real address pointing to physical memory (which is just an illusion of the guest OS); machine address is the real address pointing to the physical memory. TLB stores virtual to machine address translation.

Questions for Resource Container and Lottery Scheduling:

2. *Briefly explain the difference among `protection domain', `resource principal', and `execution/scheduling unit'.*

Process is the protection domain. We need to avoid activities from different protection domains to overwrite/see each other's state.

Resource principle is about resource allocation. Entities that belong to the same resource principle will share the same resource allocation and scheduling priority.

Execution/scheduling unit is thread in OSes that support threads. It is the unit that will occupy CPU at any moment.

2. *Suppose we need to schedule applications with the same priority but different numbers of threads. What is the resource-container's solution to achieve fairness among these applications? What is the solution under lottery scheduling?*

   Resource container: create two resource containers, one for each application, and give them equal priority. Whenever a thread is created, bind it with its corresponding application's resource container.

   Lottery scheduling: create two currencies, one for each application. Assign the same amount of base tickets to these two currencies. Suppose each thread with tickets issued by corresponding application currency.

Questions for Scheduler Activation: *Compare user-level thread and kernel-level thread: name two advantages of user-level thread over kernel-level thread and two advantages of kernel-level thread over user-level thread.*

Advantages of user-level threads:

- It has less overhead on operations like synchronizations, thread creation, etc.
- It has more flexibility regarding scheduling algorithm and others.

Disadvantage of user-level threads:

- Kernel is not aware of the existence of multiple threads. One user-thread's I/O could block the whole group of user threads.
- In M:N (M user threads sharing N kernel threads) mode, Kernel cannot tell the priority difference among the M user threads. Bad kernel scheduling decision is inevitable.

Questions for Monitors: *Name one difference between the design of Hoare-Monitor and Mesa-Monitor.*

The biggest difference is on their designs of condition variables.

In Hoare-Monitor, the signaling process is blocked; one waiting process immediately wakes up and executes. The invariant associated with the condition variable is GUARANTEED to be satisfied when the waiting process wakes up.

In Mesa-Monitor, the signaling process continues its execution after the signal. The waiting process might wake up and finds out the condition it is waiting for is still unsatisfied.

Questions for Unix:

1. *Give two examples of UNIX design that help the perfromance of disk file access.*

   The open-file table caches the translation from file path/name to inode.
   Buffer cache in physical memory significantly decreases the number of disk accesses.

2. *What is the difference between UNIX and Multics virtual memory management?*

   UNIX does not use segments as much as Multics.

Questions for Multics:

1. *Multics Virtual Memory uses segmentation upon paging. Why?*

   Each segment is a logical unit in the system. It serves as a sharing unit, as well as protection unit. It is easy for developers to specify in their program.

   Page is a better unit for physical memory management. A segment could be very long, and segmentation could lead to external fragmentation. Therefore, Multics uses segments and use pages to support each segment.

2. *Why does Multics need lp register?*

   lp points to the current linkage section. Multics needs lp to access data and code outside the current code segment.

   Specifically, shared code segment needs to be `pure'. As a result, if we want to refer to another segment, we cannot directly use the other segment's segment id for reference (different processes could have different segID for the same segment). The solution is to use linkage section and lp register.

Questions for THE and Nucleus: *What is the biggest difference between the design concepts of THE and Nucleus? What are the pros and cons?*

THE: monolithic kernel, hierarchical design.
Nucleus: micro-kernel. Many operating system functionalities are implemented as processes running on the micro-kernel.

Comparison:

Nucleus is more reliable (the `kernel' is smaller, some resource-management component's crash will not affect other part of the system, message passing naturally puts more isolation among processes than shared memory schemes)
Nucleus can provide more flexibility in system design
Nucleus has worse performance than monolithic kernel