Test 3 Sample questions

File Systems & UNIX Paper

1. What is a File system

2. In a file-system, (a) What is meta-data? (b) Where is meta-data stored? (c) Why is it important for a file system to maintain the meta-data information? (d) List some of the typical information that is part of the meta-data.

3. In the "UNIX/Ritchie" paper, consider three major system components: files, I/O devices, and memory. UNIX treats I/O devices as special files in its file system. What other mappings are possible among the above three components? (In other words, which component can be treated as another component)? What would be the use for each possible new mapping?

4. Why did the authors of the "UNIX" paper consider the UNIX file-system to be their most important innovation?

5. (a) Suppose you collect a trace of I/O operations above the file system layer (in applications or in system calls). Do you expect to see more write I/O operations or read I/O operations? (b) Now suppose you collect a similar trace of I/O operations below the block device layer (in the disk or device driver). Do you expect to see more write I/O operations or read I/O operations? Explain why?

6. If you increase or decrease the disk block size in a file system, how (and why) will it affect ***(a)*** the size of the inode, and ***(b)*** the maximum size of a file accessible only through direct block addresses?

7. How does the inode structure in UNIX-based file-systems (such as Unix V7) support fast access to small files and at the same time support large file sizes.

8. What does the file system cache do and how does it work? Explain with focus on the data structures used by the file system cache.

9. Explain the role of *file system cache* during (a) read I/O operations and (b) write I/O operations.

10. How does a log-structured file system work? Why is its performance (typically) better than conventional file systems?

11. In a file-system, explain how two different directories can contain a common (shared) file. In other words, how do hard links work?

12. How does the inode structure in UNIX-based file-systems (such as Unix V7) support ***fast access to small files*** and at the same time ***support large file sizes***.

13. Explain the structure of a UNIX i-node. Why is it better than having just a single array that maps logical block addresses in a file to physical block addresses on disk?

14. What's wrong with storing file metadata as content within each directory "file"? In other words, why do we need a separate i-node to store metadata for each file?

15. In a UNIX-like i-node, suppose you need to store a file of size 32 Terabytes (32 * 240 bytes). Approximately how large is the i-node (in bytes)? Assume 8096 bytes (8KB) block size, 8 bytes for each block pointer (entry in the inode)., and that i-node can have more than three levels of indirection. For simplicity, you can ignore any space occupied by file attributes (owner, permissions etc) and also focus on the dominant contributors to the i-node size.

16. In a UNIX-based filesystems, approximately how big (in bytes) will be ***an inode*** for a ***400 Terabyte (400 * 2^{40} bytes) file***? Assume 4096 bytes (4KB) block size and 8 bytes for each entry in the inode that references one data block. For simplicity, you can ignore intermediate levels of indirections in the inode data structure and any space occupied by other file attributes (owner, permissions etc).

17. Assume that the size of each disk block is 4KB. Address of each block is 4 bytes long. What is the size of the ***largest*** file that can be supported by a UNIX inode? What is the size of the ***largest "small"*** file that can be addressed only using direct block addresses? Explain how you derived your answer.

18. Assume all disk blocks are of size 8KB. Top level of a UNIX inode is also stored in a disk block of size 8KB. All file attributes, except data block locations, take up 256 bytes of the top-level of inode. Each direct block address takes up 8 bytes of space and gives the address of a disk block of size 8KB. Last three entries of the first level of the inode point to single, double, and triple indirect blocks respectively. Calculate **(a)** the largest size of a file that can be accessed through the direct block entries of the inode. **(b)** The largest size of a file that can be accessed using the entire inode.

Virtualization

1. Explain briefly with examples (1) Process virtual machine, (2) System virtual machine, (3) Emulator, (4) Binary optimizer, (5) High-level Language Virtual Machine.

2. Which interface does a Process VM virtualize? Which interface does a System VM virtualize?

3. (a) How do Interpreters differ from Dynamic Binary Translators? (b) How do Binary Optimizers differ from Emulators?

4. What are the advantages and disadvantages of Classical System VMs compared to Para-virtualized VMs?

5. For system virtual machines, explain how virtual memory addresses are translated to physical addresses when (a) hardware supports EPT/NPT (extended/nested page tables) and (b) hardware only supports traditional (non-nested) page tables.

6. How would you define the notion of *isolation* for any software or system component? What are the two (or more) extremes?

7. How does Intel VTx extending the traditional CPU execution privilege levels to support system virtual machines?

8. Compare different approaches for virtualizing I/O devices for virtual machines.

9. Give at least three mechanism(s) by which the highest privileged software, such as an operating system or a hypervisor, retains control over the CPU?

10. What is a co-designed virtual machine? Briefly describe and give an example.

11. What type of virtual machine (VM) is each of the following and why? Be as specific as possible. (a) Java Virtual Machine (JVM) (b) VMWare (c) Xen (d) Digital FX!32 (e) VirtualPC (f) (e) Transmeta Crusoe (Code Morphing)

12. What is the difference between a Type-1 hypervisor and a Type-2 hypervisor? Explain with examples

13. What is the difference between the concepts of full-virtualization and para-virtualization? Explain with examples.

14. When you have to design a system that does emulation, under what circumstances would you opt for Interpretation and when would you opt for Binary Translation? Justify your answer.

15. (a) What is a shadow page table? How is it constructed and/or updated by the hypervisor? (b) What type of hardware virtualization support is needed to avoid constructing shadow page-tables in full-virtualization?

Containers

1. What are “containers”? What type of virtualization do they provide? Explain.
2. What is OS-level virtualization? OS-level as opposed to what other levels?
3. How do containers improve upon the isolation model for a (a) single process (b) chroot jail, (c) sandboxes?
4. What are jails or sandboxes?
5. What are some key system resources that are virtualized by containers?
6. Linux containers contain consist of two main components: Namespaces and CGroups. Explain the role of each.

Security

1. Explain the three key principles of computer security?
2. Explain the basic security mechanisms supported by (a) the CPU execution hardware, (b) Memory management hardware and software, (c) File system. Assume that the machine uses x86 ISA.
3. What is the difference between security and privacy? Are they entirely the same or entirely different or neither? Explain.
4. What is a threat model? What factors should you consider when defining threat model?
5. What is authentication?
6. Describe different techniques to authenticate users.
7. What are some ways in which by which authentication mechanisms can be subverted?
8. What is sandboxing? List two sandboxing mechanisms.
9. What hardware mechanism does x86 ISA provide to ensure that Operating System's code and data are protected from user-level processes?
10. Explain Discretionary, Mandatory, and Role-based access control mechanisms.
11. Explain (a) trusted computing base (TCB) including why is it called "Trusted", (b) Reference Monitor, and (c) relationship between TCB and reference monitor.
12. Explain the two key data access principles of multi-level security (MLS) systems (also called Mandatory Access Control).
13. Give an example of a scenario where the software doesn't trust the OS, hypervisor, and/or the hardware platform on which it runs? What can the software possibly do to "secure" itself in this situation?
14. Considering memory protection, explain how the operating system ensures that user-level processes don't access kernel-level memory?

Live Migration of VMs

1. What constitutes “liveness” during live migration of virtual machines?
2. What are the key states transferred during live VM migration?
3. What are the metrics of performance in live migration of VMs?
4. How do different VM migration techniques work? How do they differ?
    1. Stop-and-copy
    2. Pure demand paging
    3. Post-copy
    4. Pre-copy
    5. Hybrid pre/post copy
5. What are the tradeoffs between different performance metrics when using each migration technique?
6. What are some optimization techniques you can use to speed up pre-copy? Post-copy? Both?
7. Under what situation would you use each migration technique?
8. What is dirty-page tracking? Why is it needed? How does it work? What are its overheads?

RAID

1. Distinction between logical and physical I/O address spaces.
2. What was the original & current motivation for RAID?
3. Why is a multiple-disk system less reliable than a single disk?
4. How does Mean Time to Failure (MTTF) change as number of components in a system increases?
5. What are the different levels of RAID and how do each of them work?
6. What are the relative benefits/drawbacks of each RAID level?
7. How is data distributed in each RAID level?
8. How is parity calculated and stored in each RAID level?
9. What is the extent of read and write parallelism in each level?
10. How is the parity calculation bottleneck in RAID 4 solved?
11. In RAID-5, explain how can you perform a single logical write operation in no more than one physical read and two physical writes?
12. In RAID 5, describe how you can complete a write I/O operation using just 2 disk reads and 2 disk writes.
13. (a) Explain (with formula), how does parity computation differ between RAID 3 and RAID 4? (b) How does parity placement on the disk (not parity computation) differ between RAID 4 and RAID 5? Explain with example.
14. In order to save power, disks are usually spun down (placed in sleep or low-power mode). This works well if there is only one disk in the system, if all data resides on the single disk, and if performance is not a major concern. Consider a RAID-5 system consisting of N+1 disks. Explain how you can redesign RAID-5 so that all the following requirements are satisfied: (1) fault-tolerance of original RAID-5 is maintained under all conditions, (2) energy consumption is minimized by spinning down one or more disks whenever possible, and (3) performance (read/write throughput) of the system is maximized to the extent possible. Again, while there is no single correct answer, you must explain all salient aspects of your design, justify any assumptions you make, and examine any design tradeoffs (e.g. energy savings to performance).