**CS-580K/452K** **Name:** ____________________

**Fall 2020**

**Midterm Exam 1**

**10/12/2020**

**Time Limit: 90 Minutes** B-Number: ____________________

**This exam contains 11 pages (including this cover page) and 4 questions. Total of points is 100. Please notice that *less is more* — more isn't always better, especially when it comes to answer. Please keep the answers short and to the point. If there are any factual errors and/or the answers are totally irrelevant to the questions, marks will be deducted.**

Grade Table (for teacher use only)

| Question | Points | Score |
|---|---|---|
| 1 | 18 | |
| 2 | 32 | |
| 3 | 42 | |
| 4 | 8 | |
| Total: | 100 | |

1. (18 points) **Question 1 – Machine-level Virtualization!**
   A machine has at least two modes (a) user mode and (b) system mode. Typically, applications' code runs in the user mode and the operating system code runs in the system mode. In the system mode, the code/program can see and manipulate the machine without restrictions. In the user mode, the code/program has some limitations in what it can do (e.g. it can't access all of the machine's memory without acquiring permission).

   Instructions are either (a) privileged or (b) non-privileged. Privileged instructions are trapped when executed in the user mode. Trapping means that the machine is forced into the system mode whereby it executes some code of the operating system to deal with the situation. In a sense, they alert the operating system when being executed. Non-priviledged instructions can run without trapping in the user mode.

   Instructions can also be either (a) sensitive or (b) non-sensitive. Sensitive instructions modify part of the machine's resources, or, exhibit different behaviors depending on if they are executed in the user mode or system mode.

   When virtualizing a CPU architecture (i.e., ISA), it is important that the virtual machine monitor (VMM) can detect and handle smoothly, any attempt of the program or guest operating system to modify the machine's resources. **It must be able to see when sensitive instructions are being executed.**

   (1) Please describe the high-level design ideas to virtualize CPU, given that all of the sensitive instructions are the privilege instructions? (6 points)

(2) Please describe the high-level design ideas to virtualize CPU, given that only part of the sensitive instructions are the privilege instructions? (6 points)

(3) Without hardware MMU (i.e., memory management unit) like Intel EPT, how do you use a software manner to realize memory management in virtualization — mapping the address space of a process running in a VM to its machine's physical memory? Please describe the high-level design ideas. (6 points)

2. (32 points) **Question 2 – Virtualization is Costly!**
   (1) Please state the main sources that negatively impact I/O performance in KVM or Xen based virtualization solutions in comparison with the native (i.e., no virtualization). (6 points)

   (2) To achieve high I/O performance under Xen/KVM virtualization, what are the possible optimization solutions, and why do you think these solutions will work? (name three) (6 points)

(3) Under a round-robin vCPU scheduler, suppose the time-slice (i.e., maximum time that a vCPU can run) is 30 ms. Four VMs (VM1 to VM4, each with 1 vCPU) share a single CPU. When a client sends an I/O request to VM1, how long does the client expect to receive the response from VM1 in the *worst* case? (4 points)

What are the results if the time-slice is 10 ms and 1 ms, separately? (4 points)

What are the gains and losses using a small time-slice (e.g, 1 ms) in comparison with a large one? (4 points)

(4) We do want to schedule I/O-intensive VMs with small time-slice and CPU-intensive VMs with large time-slice. Please design a vCPU scheduler that can approximately achieve this goal. Please make reasonable assumptions and justifications. If you want to distinguish types of VMs (e.g., I/O-intensive or CPU-intensive), please describe how. (8 points)

3. (42 points) **Question 3 – OS-level Virtualization!**
(1) What are the two systems-level techniques that underpin containerization? (6 points)

(2) Why is I/O performance under containerization much better than KVM or Xen based virtualization? (If you don't think so, please also justify.) (6 points)

(3) Then would the I/O performance of containers be the same as the native case (i.e., running processes directly on an OS), and why? (6 points)

(4) Containers are much light-weight in comparison with machine-level virtualization solutions like VMs. However, in practice we do not completely replace VMs with containers. What are the restrictions of using containers? Or in what scenarios, it's better **not** to use containers; instead, we should use KVM or Xen based virtualization? (You can list the main drawbacks of containers, and figure out the scenarios unsuitable for using containers) (6 points)

(5) gVisor (Google's container solution) provides another isolation mechanism for container, which intercepts application system calls and acts as the guest kernel. Instead of passing system calls to the native kernel, gVisor implements a substantial portion of the Linux system surface. Thus each container may have its own user-level kernel. This design supposes to have **better security** properties than traditional ones (e.g., Docker containers). How does gVisor achieve this? (6 points)

Actually, gVisor trades performance for such better security properties. Which types of applications do you think suffer *less* in terms of performance drop when we use gVisor containers, and why? Do you have some ideas to mitigate performance overhead of gVisor? (6 points) gVisor just mitigates but does not completely eliminate the security vulnerability – e.g., it is possible that if any one of the gVisor containers breaks out, it can allow unauthorized access across containers running on the same host. Please describe your ideas to further protect gVisor containers. (6 points)

4. (8 points) **Question 4 – Server Consolidation**

One typical usage scenario for containers and/or virtual machines (i.e., server virtualization techniques) is server consolidation. Server consolidation is an approach to place multiple virtualized servers (either in containers or virtual machines) on the same physical machine in order to reduce the total number of servers or server locations that an organisation requires. The practice originally developed in response to the problem of server sprawl, a situation in which multiple, under-utilised servers take up more space and consume more resources than can be justified by their workload. Later, the concept of server consolidation has been well materialized in cloud platforms where a single physical machine is shared by multiple cloud tenants.

Figure 1 shows the total performance (summed from all VMs running on the same physical host) changes as we consolidate more VMs on the same physical host.

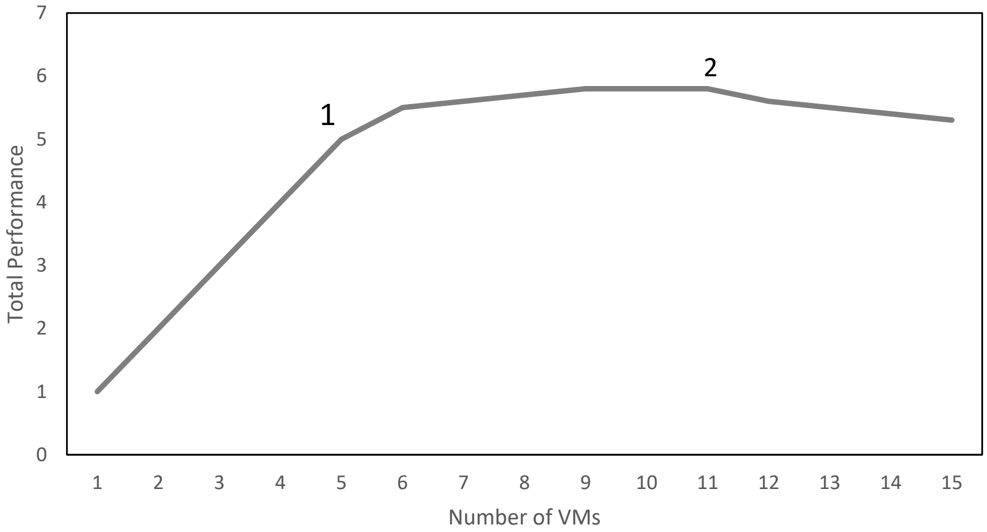

Figure 1: Total performance changes as more VMs consolidate on a single physical host.

(1) As you can observe, after point 1 in Figure 1, the total performance stop increasing. Can you guess why? (4 points)
(2) After point 2 in Figure 1, the total performance start dropping with more VMs, and why? (4 points)