Quiz 4

Due Nov 23 at 11:59pm **Points** 50 **Questions** 5 **Available** Nov 23 at 12:01am - Nov 23 at 11:59pm about 24 hours **Time Limit** 30 Minutes

Instructions

You have to work on the quiz independently. **You cannot discuss with any other people.** Any form of discussion is considered cheating.

This is open book and open note quiz. But **no search on the internet**. You can access the reading list and the paper pdf files listed in the reading list on Canvas.

This quiz was locked Nov 23 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	29 minutes	50 out of 50

ⓘ Correct answers are hidden.

Score for this quiz: 50 out of 50
Submitted Nov 23 at 7:17pm
This attempt took 29 minutes.

Question 1

10 / 10 pts

Paper: CELL
Question: Is CELL a true virtual machine like Xen? Why or Why not?

Your Answer:

It is not really a true virtual machine. CELL mimic the OS interface, means it cannot run many types of GuestOS like Xen. The kernel of CELL is fixed. Also, the VPs are only bunch of processes and isolated by setup resource control. I would compared CELL to a docker daemon on linux in this case.

No, CELL is not a true virtual machine like Xen. It provides an process-level isolation between virtual phones (VPs), but not a virtual machine isolation. Additionally, it doesn't allow you to run iOS on top it. For true virtual machine like Xen or VmWare, you can run different OSes.

Question 2

10 / 10 pts

Paper: Bigtable: A Distributed Storage System for Structured Data
Paper: The Google File System
Question: What capabilities do GFS and Bigtable have to allow it to scale to thousands of machines?

Your Answer:

They have a single master server where store all metadata only and redirect traffic to each of the trunk server, where each of these server may have replicas that can handle the client request by using the server id and offsets. The metadata server stores all locations in memory to speed up the look up process thus be able to handle as many machines as possible.

Tolerate faults since with thousands of machines, failures happen more often.

Question 3

10 / 10 pts

Paper: Scheduler Activations: Effective Kernel Support for the User-level Management of Parallelism
Question. For a uni-processor machine, what do you need to do to achieve a reasonably good parallelism if you are using a user-level thread package (no scheduler activation)?

Your Answer:

It has to make the user-level thread package aware the kernel events, especially blocking/preempts events, or the user process will be frozen anyway when such events has happened, the scheduler will be frozen as well to wait until the kernel has finished the operation, which basically makes the parallelism invalid.

Make sure each user level thread not being blocked by using asynchronous I/0 calls, and also make sure to avoid page faults if possible.

Question 4

10 / 10 pts

Paper: The Google File System
Question: Name **two** specializations that the Google File System has made over a traditional distributed file system?

Your Answer:

1. It does not guarantee only one append action will be done to data when performing the action. The application has to make keep track of data duplication. This specialization leverage the consistency complexity away the file system perspective.
2. GFS trunk are by default 65MB, which has poor performance is large amount of small files in the system. This stay the system goal toward serving large file and providing high throughput.

Support appending operation more efficiently

Large block size (chunk)

No POSIX file API, i.e. specialized API

Question 5

10 / 10 pts

Paper: Scheduler Activations: Effective Kernel Support for the User-level Management of Parallelism
Question: This paper does not talk much about the thread scheduler policy. Does use level thread management have advantage in terms of scheduling policy over kernel-level thread approach? Why?

Your Answer:

The application may have be better understanding the workload characteristics and make better decision on scheduling. E.g. database application may need to have fast response to sudden small query while temporary borrow resource from on going complex query. Kernel level thread schduling may not have such knowledge while the user-level library allows programmer to specify such policy.

User level thread management can implement application-specific scheduling, for example lottery scheduling, even though the OS itself doesn't support lottery scheduling at all.

Quiz Score: 50 out of 50