OMSCS 6210 Spring 2023 Test 3 (120 min Canvas Quiz)

Instructions

1. **The test is closed everything except intellect.**
2. **You can use only one scratch paper which should be visible when you perform the room scan on Honorlock.**
3. **For True/False questions, no credit will be given without justification.**
4. **If you face any issue, you can contact Honorlock support from Canvas while taking the exam.**
5. **Clarifications will be posted on ed sometime after Noon on April 29th.**
6. **We suggest you wait till we post clarifications before giving the exam.**

Good luck!

Internet-scale computing

1. (11 minutes, 11 points) (Giant-scale services)

Assume you are the developer responsible for deploying ChatGPT. Assume that the underlying ML model used by ChatGPT needs a total of 800 GB memory for storing the model parameters. The model can be decomposed into 8 GB slices that can be run in a pipelined manner. The data center you are deploying ChatGPT offers machines with a fixed 8 GB of memory, and you have profiled each slice to take 1 ms of execution time. Assume zero communication cost to convey the results from one slice of model execution to the next.

1. You now want to deploy this model to support 1 million requests per sec. How many minimum machines will you need? [2]

- Machines for 1 instance : 200 000 000 000 * 4 bytes = 800 000 000 000 bytes = 800 GB = 100 machines.
- 1 pipelined instance will create a throughput of 1000 requests per sec.
- We need 1000 instances for 1 M requests / sec.
- Thus we need 100*1000 = 100,000 machines!

2. What is the latency incurred by the user for each query? Answer with justification if you can reduce the latency by employing more machines per query. [2]

- Latency for 1 call: 100 machines * 1 ms = 100 ms.
- No the latency doesn't change if we scale the instances as the model is still sequential.

3. Calculate the throughput if one node in the whole fleet fails. [2]

- If one node fails, one entire pipeline will be bottlenecked by the failing node.
- We thus will have a throughput of 999 000 requests per s.

4. Calculate the throughput if two nodes fail. [2]

- If the two failing nodes are serving the same pipeline then the throughput will fall to 999,000.
- If the two failing nodes are serving different pipelines then the throughput will fall to 998,000.
-
- 5. The model gets updated. The updated model has to be deployed in the servers. Discuss pros and cons of each of the strategies (fast, rolling, big-flip) for doing the update. [3]
-
- Fast:
- Pro: No inconsistent results since all machines updated at the same time
- Con: Downtime when the service is unavailable for a time equal to that needed for updating 1 server
-
- Rolling-update:
- Pro: No downtime but just decreased yield until all the servers are updated
- Con: Different results for different queries until all the servers are updated.
-
- Big-flip:
- Pro: No downtime, but yield down to 50% for a duration equal to updating half the servers in the entire fleet.

- Con: Inconsistent results for 50% of the queries during the upgrade.
-

2. (x minutes, 14 points) (Map-reduce)

a) [10 points] A map-reduce application consists of:

- **30 shards of data to be processed**
- **10 distinct outputs to be produced**
- **CPU time to execute a map function: T_m**
- **CPU time to execute a reduce function: T_r**
- **I/O time to write the intermediate result by a mapper: T_i**
- **RPC time to fetch an intermediate result by a reducer: T_{rpc}**
- **I/O time to write out the result by a reducer: T_f**

The Map-reduce infrastructure includes asynchronous RPC, allowing a reducer to fetch the intermediate results in parallel from the mappers. The map-reduce infrastructure uses 10 threads for the map function and 5 threads for the reduce function. Ignoring scheduling overheads by the infrastructure, compute the total execution time for the above application. (Show your work for partial credit)

Ans:

- 10 threads work in parallel each taking: Tm+Ti to finish (+2)

- Since there are 30 map instances total time for mapping: 3*(Tm+Ti) (+2)

- Each reducer has to do 30 RPCs to get the intermediate results. However, the asynchronous RPCs go on in parallel to fetch the map results from all the reducers.

Trpc (+2)

- 5 reducers works in parallel to produce the final result each taking:

Trpc + Tr + Tf (+2)

- Since there are 10 reduce instances total time to complete the reduce

work: 2*(Trpc+Tr+Tf) (+1)

- Total time = 3*(Tm+Ti) + 2*(Trpc+Tr+Tf) (+1)

b) [+2 points] T/F with justification. In the MapReduce framework, the Domain expert is responsible for sharding the data.

Ans: False, the Domain Expert is only responsible for writing the Map and Reduce functions.

c){+2 points] T/F with justification. When there are M mappers and R reducers, M+R intermediate files are passed from mappers to reducers.

False, M*R intermediate files are passed from the mappers to the reducers.

3. (x minutes, 16 points) (CDN – Coral)

a) [3 points] Why is Coral called an overlay network? Give another example of an overlay network.

- It is a virtual network that sits on top of the physical network of nodes that enables nodes to share content with each other
- Coral CDN has user level routing tables with routing information that may differ from physical network topology (+2)
- The internet, VPNs, CDNs are examples of overlay networks (+1)

b) [2 points] [True or False with justification] Coral is prone to tree saturation.

False.

Coral uses sloppy DHT key-based routing to prevent tree saturation by ensuring the get/put calls get serviced by possibly many intermediate nodes instead of just one node with node ID N ≅ key

+0 Incorrect
+1 False without valid justification
+2 False with valid justification

c) [5 points] Key-based routing in Coral
Given below is a routing table for Coral, where the source node 9 (src) is trying to reach the destination node 1 (dst). Entries in the second row show the XOR distance from the source (src) to a node that is currently reachable (i.e., src has a valid IP-address for that node). Entries in the third row show how the routing table evolved after the first iteration of key-based routing.

Nodes	**1 (dst)**	0	3	2	5	4		6	**9 (src)**
XOR dist				3	4			7	8
XOR dist	0	1	2	3	4			7	8

(i) [2 points] Which node did "src" make an RPC call to in the first iteration to get the new entries in the third row? Why?

Ans: Node 5, since it is half the distance (i.e., 4 from dst)

(ii) [2 points] Which node will "src" make the next RPC call? Why?

Ans: Node 3, since it quarter the distance (i.e., 2 from dst)

(iii) [1 point] How many hops does it take "src" to reach "dst"? List the nodes visited by "src" in getting to "dst".

Ans: 4 hops; {5, 3, 0, 1}

d) [6 points] In the following problem

- Put(x, y) denotes putting the key-value pair (key = x and value = y)
- Get(x) denotes getting the value corresponding to the key = x
- Assume a coral system in which the "l" value for a node is set to 2; "β" is 1

- Node-id 200 currently has 1 entry for the key 200 with value 6

  (i) [1 point] Node-id 100 does: Put(200,100). Coral's key-based routing mechanism routes this request through nodes 100,150,200. Where will the key-value pair be placed?

  Answer: Key-value pair (200,100) will be placed on node-id 200 (+1)

  (ii) [1 point] Next, node-id does: Put(200,250). Coral's key-based routing mechanism routes this request through nodes 250,209,200.
  Where will the key-value pair be placed?

  Ans: Key-value pair (200,250) will be placed on node-id 209 since node-id 200 is full (+1)

  (iii) [2 points] Next, node-id 15 does: Get(200). This request is routed through node-ids 150,200.
  At the same time, node-id 300 does: Get(200). This request is routed through node-ids 209,200.
  What values do node-ids 15 and 300 get back?
  Node-id 15 gets back values {6, 100}.(+1)
  Node-id 300 gets back value 250.(+1)

  (iv) [2 points] Next, node-id 60 does: Get(200) and its request gets routed through the node-ids 80,200. Will this "Get" request be successful? Explain why or why not.

  Answer depends.
  Yes, if node 200 is always up
  No, if node 200 is down

Real-time and Multimedia

4. (x minutes, 10 points) (TS-Linux)

[2 points] What is the priority inversion problem and how can it be handled in a Linux-like OS that uses priority-based scheduling?
Priority inversion is a situation where a higher priority task is indirectly superseded by a lower priority task.
The scheduler uses the HLP protocol: when a task acquires a resource, it automatically gets the highest priority of any task that can acquire this resource

[2 points] Proportional period scheduling in TS-Linux allocates to a requesting task a desired proportion (Q) of the CPU in each period (T - a scheduling parameter). What problem is this aiming to solve?

- Provides guarantee towards meeting the application's requirement by performing admission control based on proportion Q of quantum T required by an application

[6 points] A video game running on top of TS-Linux is using the one-shot timer. It has it programmed to go off every 300 microseconds to update some internal state of the game. It uses an overshoot parameter of 30 microseconds. At 290 microseconds since the last firing of the one-shot timer there is an external interrupt (lower in priority compared to timer events) into TS- Linux. List the steps taken by TS-Linux upon getting this interrupt (concise bullets please).

- Upon receiving the external interrupt, TS-Linux inspects its timer queue
- It recognizes the one-shot timer is within the "overshoot" window
- It reprograms the one-shot timer and sticks it into the timer queue at the appropriate spot
- It services the one-shot timer interrupt as though it happened now thus avoiding the one-shot timer
  interrupt which would have gone off in another 10 microseconds
- It services the external interrupt as appropriate commensurate with its priority.

5. (x minutes, 10 points) (PTS)

a. [2 points] PTS runtime system provides time synchronization for different producers and consumers of a distributed application.(Answer True/False with justification)

False, PTS system relies on each node in the system having their clocks aligned with one another using a standard protocol such as NTP.
+2 False with correct justification
+1 False with incorrect justification

b. (6 points) Your friend is developing a multi-modal live streaming application that is represented using a pipelined graph of tasks. Give three reasons why you would advise your friend to choose PTS rather than using Unix sockets and processes.

Ans:

1. PTS gives the developer to associate timestamp with the items being generated by the streaming application.
2. PTS gives the developer the ability to correlate items from different channels (corresponding to different modalities) that have the same timestamps.
3. PTS gives the developer to propagate temporal causality for processed items in the application graph.

c. [2 points] Assume a PTS Channel ch1 has items with timestamps 25, 50, 75, 100.

Consider the following PTS code sequence by a thread T1:
<item, ts> = Get (ch1, "now"); // returns latest item from channel 1

Digest = Process(item);

<some code> // code to process item just gotten

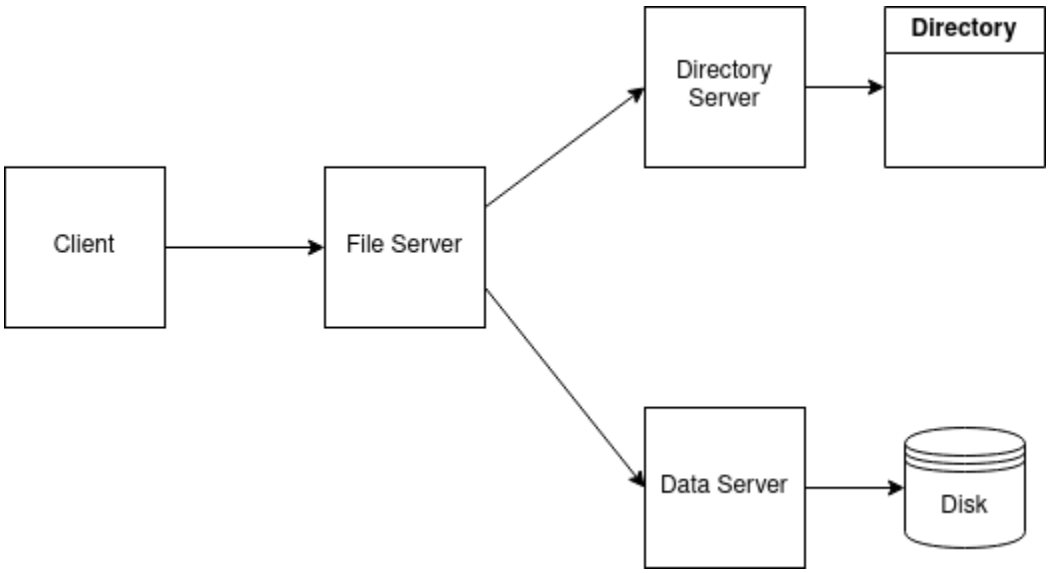
Put (ch2, Digest, ts+25); //put digest with timestamp ts+25

What is the timestamp associated with the above put operation?
Timestamp associated is 125

Failures and Recovery

6. (12 minutes, 10 points) (Quicksilver)

In Quicksilver, consider a client is contacting a file server for opening and writing to a file. For this, the file server contacts a directory server for creating the file (which maps filenames to "pointers" to file's data), and a data server for writing to a disk block (which allocates data blocks on disk and writes bytes to them).



(a) (2 points) Describe at least one scenario in which this client's request fails. What are the breadcrumbs that could be left behind due to such a failure?

Suppose the data server allocates a block on the disk, writes data to it, but the directory server fails. Now, there is no *pointer* pointing to the block, and the block has effectively "leaked", and cannot be easily reclaimed. (2 points)

(b) In the above scenario, assume that the client closes the file after writing to it. The coordinator for the shadow transaction tree that represents this client-server interaction is the client node.

(i) (2 points) In the absence of any failures, what will happen upon the "close" call by the client?

Ans: The coordinator will issue a 2-phase commit down the transaction tree. Once the 2-phase commit completes, the shadow transaction tree will be taken down by Quicksilver.

(ii) (2 points) Suppose a 1-phase commit is used instead of 2-phase commit. Describe a failure scenario that may leave the system in an inconsistent state.

Ans: Suppose there is a one-phase end commit message going from the coordinator to the disk server and the file server, following

which one of the servers fails. The coordinator will declare the transaction a success, but the system will be left in an inconsistent state.

(iii) (2 points) Explain why a 2-phase commit will NOT leave the system in an inconsistent state for the failure scenario you described above.

Ans: Suppose a failure happens in case of a two-phase commit, the server that fails will either not vote "yes" for the commit message, or will fail to respond. The coordinator will abort the commit, and the system will return to a consistent state.

(c) (2 mins, 2 points) Suppose the file server is built to serve numerous clients at the same time, and talks to multiple data servers (each managing a single disk, for example). What would be the advantage of the file server making non-blocking IPC calls to data servers, as opposed to blocking IPC calls?

By not blocking on the completion of a write by a particular data server, the file server will be able to have multiple write calls to data servers in-flight at the same time, and allow them to progress in parallel. This will help it handle multiple clients better than if it were doing blocking calls. (2 points)

7. (9 minutes, 8 points) (LRVM)

(a) (3 min, 3 points) How does the "no restore" mode in begin-transaction help in improving the performance of a server written on top of LRVM?

Ans: + 3 No need to create an "undo" record for the "set-range" call within this transaction.

+ 1 Partial points awarded. Student does not explain 'undo' records and how they are not needed here.

+0 Student mentioned 'redo' instead of 'undo' record.

(b) (3 min, 3 points) During crash recovery, the redo log is applied to the data segments to bring the server to a consistent state prior to the crash. LRVM chooses to apply the log to the affected data segments starting from the tail of the log rather than the head. Why?

Answer: It is conceivable the same records of the data segment will written in multiple transactions T1, T2, T3, ... in that order. T3 is the most recent write to that data record. So the writes by transactions T1 and T2 are redundant and can be eliminated thus reducing the crash recovery time.

(c) (3 min, 2 points)True/ False with justification.
No-flush during end-transaction aids in enhanced performance but increases the vulnerability window.

Answer: True, No-flush helps in lazy persistence which improves performance (since the server is not blocked until the "redo log" I/O is complete), but a crash before the log is written will leave the application in an inconsistent state.

8. (x minutes, 10 points) (RioVista)

a. [3 points] Your friend John argues that a battery backed file cache such as Rio provides the exact same functionality and benefits as the end transaction in LRVM with the no-flush mode. Would you agree with his argument? State why or why not.
Ans - No we would not agree with his argument
This is because the end transaction with the no-flush mode introduces a window of vulnerability - from the time of the transaction ending to the time when the "redo log" I/O is complete. On the other hand, power failures would not affect the contents of the Rio file cache and hence it is not vulnerable to power failures.

b. [2 points] Answer True/False with justification
Rio file cache helps in improving performance when applications running on top of it create and delete temporary files frequently.
Ans - True. Since these files need to be housed only in the file cache and do not need to be written back to disk, we end up saving a lot of I/O cycles.

c. [3 points] In RioVista, consider a transaction that completes successfully. List the number of copies of the persistent data structures that happen in the following table:

What is copied?	Where is it copied?	Lifetime of the copy

Ans:
first entry in the table : Undo log; Rio file cache; Duration of the transaction
Second entry: Persistent data; Rio file cache; Until cache replacement
Third entry: Persistent data; Disk; Lifetime of the data segment

d. [2 points] Answer True/False with justification
The "mmap" system call maps a file on the disk into the application's virtual address space. With Rio file cache, an
"msync" system call is required to ensure that writes to the mapped file safely persist on stable storage.
Ans - False. Newly written data is automatically persistent because the Rio file cache is battery backed and survives power failure.

Security

9. (x minutes, 10 points) (Security principles, AFS)

Throw back to the 80's. You are one of the designers of AFS. You choose to implement AFS using ONLY a public-key encryption system. Note: Symmetric key (i.e., private-key) encryption should NOT be used for any of the client-server interactions.

Answer the following questions:

(a) (2 points) A new user joins the system. What all needs to happen in the system to give the new user the same rights and privileges as existing users?

Ans:
- create login (user name, password, home directory on server)
- create a pair of keys {public, private} for the new user
- populate the login information and the keys to all the workstation kiosks on campus

(b) (2 points) With your implementation, when a user logs in to "virtue", what should happen?

Ans:
- user logs in with their credentials
- "virtue" retrieves {public, private} key pair for this user from its database

(c) (2 points) In your implementation of the system, when a request comes from a client, how will the server know the identity of the client to enable decryption of the message?

Ans:
- Every message from "virtue" will be
    - {public-key of the currently logged in user, message encrypted with the public key}
- server uses the public-key as the identify of the user to retrieve the associated private-key for decrypting the message.

(d) (2 points)
In your implementation of the system, when a reply comes from the server, how will "virtue" know how to decrypt the message?

Ans:
- Every message from the server will be
    - {public-key of the client, message encrypted with the public key}
- "virtue" will retrieve the private-key associated with this public-key for decrypting the message.

(e) (2 points) A student graduates and his privileges to AFS have to be revoked. What all needs to happen in the system to ensure that the student has no access to the system?

Ans:
- Login credentials (user name, home directory) need to be deleted from the server
- On all the workstation kiosks {user-name, associated (public-key, private-key) pair} needs to be purged.