CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

Note:

1. **Write your name and GT number on each page.**
2. The test is **CLOSED BOOK** and **NOTES.**
3. Please provide the answers in the space provided. You can use scratch paper (provided by us) to figure things out (if needed) but you get credit **only** for what you put down in the space provided for each answer.
4. For conceptual questions, **concise bullets** (**not wordy sentences**) are preferred.
5. While it is NOT REQUIRED, where appropriate use figures to convey your points (a figure is worth a thousand words!)
6. **Illegible answers are wrong answers.**
7. **DON'T GET STUCK ON ANY SINGLE QUESTION…FIRST PASS: ANSWER QUESTIONS YOU CAN WITHOUT MUCH THINK TIME; SECOND PASS: DO THE REST.**

Good luck!

Question number	Points earned	Running total
0 (0 minutes) (Max: 2 pts)		
1 (10 minutes) (Max: 10 pts)		
2 (5 minutes) (Max: 5 pts)		
3 (5 minutes) (Max: 5 pts)		
4 (5 minutes) (Max: 5 pts)		
5 (5 minutes) (Max: 5 pts)		
6 (10 minutes) (Max: 10 pts)		
7 (10 minutes) (Max: 10 pts)		
8 (10 minutes) (Max: 10 pts)		
Total (60 minutes) (Max: 62 pts)		

0. **(0 min, 2 points)** (you get 2 points regardless of your answer)

Pre-exam quiz using Gradescope:
(a) I took it
(b) I procrastinated taking it (Kishore has said this is a good practice!)

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

Lesson 5: Distributed Systems

1. **(10 min, 10 points) (Lamport's M.E. Algorithm)**

In a distributed system with N nodes, the communication among the nodes is constrained to use a ring topology. That is, node P_i sends messages ONLY to node $P_{(i+1) \bmod N}$. Assume that there are no loss of messages and the messages go in-order between any two nodes. The nodes use Lamport's logical clock to order the communication events.

Your co-worker is implementing Lamport's M.E algorithm in this system. Her goal is to minimize the amount of communication. She uses three types of messages in her implementation: LOCK, ACK, UNLOCK.

Answer the following questions:

(a) What should be the action at a node upon receiving a LOCK message?

Ans:

1. Enqueue the LOCK Message in the queue according to the timestamp
2. If the received LOCK's message's timestamp is later than the node's lock request timestamp, defer Ack, else, send Ack to the next node in the ring. This Ack will have the same timestamp as of the incoming LOCK Request.

+1 for each point

(b) What should be the action at a node upon a receiving an ACK message?

Ans:
If the node has a pending LOCK Request:
Case 1: The Ack received is in the response to the LOCK Request sent by this node – The node will have mutual exclusion and will enter the critical section
Case 2: The Ack has a timestamp earlier than the node's LOCK request timestamp – Just forward this Ack message
Case 3: The Ack has a timestamp which is later than the node's LOCK request – Enqueue this Ack according to the timestamp

If the node does not have a pending LOCK Request, then, simply forward the ACK message received

+1.5 - Handling the case where the node has a pending LOCK request
+0.5 - Handling the case where the node does not have a pending LOCK Request

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

(c) What should be the action at a node upon receiving an UNLOCK message?

Ans:

1. If the node has a LOCK Request pending, remove the Unlock message and wait for the Ack
2. If the node does not have a LOCK Request pending, forward the unlock message to the next node.
3. If the node generating the Unlock message received the same, it will just remove the Unlock message.

+1 – Point 1
+0.5 – Point 2
+0.5 - Point 3

(d) When does a node know that it has successfully acquired the LOCK?

Ans: When a node receives an ACK message in response to its LOCK request (the ACK is marked with the timestamp of its LOCK request), it knows it has successfully acquired the lock and can enter the critical section.

+2 – The node (that generated the LOCK request) receiving the ACK message

(e) How many ACKS will a node receive before it knows that it has successfully acquired the LOCK? (No credit without justification)

Ans: A node must receive a single ACK with its timestamp to know it has acquired the lock. Due to the ring topology, wherein only information regarding the earliest timestamped request is propagated, a node waiting on a LOCK will receive an ACK with its timestamp only if its request had the earliest timestamp. Otherwise, its request will be buffered by a node with an earlier request and will only be considered by the system once all earlier requests have been ACKed.

Kishore's comment: Please note that all the above answers assume that they have an algorithm similar to what I gave. What if somebody gives an algorithm that IS EXACTLY the same as Lamport's ME algorithm from the paper? I.e., they completely ignore the ring topology? I suggest in that case give a grace score of 3 points for the entire question and let them know their mistake. The question explicitly asks for minimizing the communication.

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

2. **(5 min, 5 points) (Latency Reduction in RPC)**
Given the following:

- It takes 10 ms to perform a context switch at a node (i.e., switch from one process to another one)
- It takes 20 ms to communicate between the client and the server (includes all the OS protocol processing overheads at both ends and the network communication)
- It takes anywhere from 1 ms to 30 ms for the server procedure to execute

Your buddy is designing an RPC package. His goal is two-fold: (a) reduce the end-to-end latency for the RPC, (b) ensure that the processing resource at both the client and the server are fully utilized. What tips would you give him to accomplish both these objectives?

- The service time on the server can be as small as 1ms.
- Even though the communication takes 20ms, the context switches (client process -> new process -> back to current client) may take at least 20ms.
    - This could result in increasing the RPC latency.

Thus, the advice I would give him to strike a happy medium between using the resources on the client side and reducing RPC latency:

- Spin on the client side for a short time (say 10 ms – the time it takes for one context switch)
- If the response does not come within this time, then context switch the client

+3 if they simply context switch on the client side for conserving resources or SPIN continuously on the client side for reducing the RPC latency.
+5 if they spin and then context switch

3. **(5 min, 5 points) (Active Networks)**
You are the designer of the "capsule" mechanism in Active Network. You make the following design decision:

- upon receiving a capsule which the node has not seen before, the node requests the code for processing the capsule from the "prev" node contained in the capsule

What should be the action at a node if the "prev" node says it does not have the code? Justify your answer.

The node should simply drop the capsule. We rely on higher level protocols to handle end-to-end acknowledgements.

(All or nothing)

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

Lesson 6: Distributed Objects and Middleware

4. **(5 min, 5 points) (Spring Kernel)**
You are the implementor of the "subcontract" subsystem in the Spring kernel. On the client side, you have the following API calls available to the Client-side stub: Invoke; Marshall: Unmarshall

You want to optimize the "Marshall" call to exploit the location of the server. If the server is on the same machine as the client, how would you optimize marshalling the arguments of the call?

If the server is on the same machine as the clients, the arguments can be marshalled into a shared memory space instead of transmitting the marshalled object to the server and vice versa.

(All or nothing)

5. **(5 min, 5 points) (EJB)**
You have a startup to implement a portal for airline reservations. The clients come to you over an insecure wide-area network. These are the objectives which are your "secret sauce" for the startup:

- You want to exploit parallelism across independent client requests
- You want to exploit parallelism within each client request
- You want to protect your business logic from being exposed to the wide-area Internet

You are planning to use EJB for meeting these objectives. Your N-tier solution has a Web container, an EJB container, and a Database server. To meet the design objectives:
(a) What functionalities would you put into the Web container (that interfaces with the client browsers)?

Functionalities to put in the Web container:
i. Servlet which corresponds to a session of a client
ii. Presentation logic for each servlet

(b) What functionalities would you put into the EJB container?

i. A session façade that corresponds to each servelet in the web container
ii. The business logic layer associated with each session façade
iii. Entity beans which are the data access objects serving as an interface for the business logic layer to access the database

Name:_____GT Number:

With this design, the business logic is not exposed and also parallelism across independent client requests and within the client requests can be exploited through the entity beans.

- 2 if business logic exposed to the WAN
- 2 if no parallelism within client
- 1 if no parallelism across clients

Lesson 7: Distributed Subsystems

6. **(10 min, 10 points) (GMS)**
You are implementing the idea of paging to peer memories from GMS in a datacenter environment. There are 100,000 nodes in the datacenter. The datacenter has a scalable global file system which is available to all the nodes to use as a paging device. The implementation objectives you set for yourself are as follows:

- You want page fault service to take as little time as possible
- You want to be space efficient in the data structures to support your GMS implementation
- You want to be communication efficient in your GMS implementation
- You want to be resilient to failures of peer nodes

(a) One co-worker suggests using a replicated data structure at each node (functionally equivalent to a page table for managing virtual memory) that informs the faulting node which peer node has the page that it is looking for. What is the downside of this design choice with respect to the above implementation objectives?

- Replicating a page table with potentially 100000 * (page table size) entries would not be in keeping with the space efficiency requirement.
- Maintaining consistency of this data structure across 100,000 nodes would be inefficient- each time a page fault occurs it requires at least 100,000 messages on the network, which is not in keeping with the communication efficiency requirement.

-1 if space inefficiency not mentioned
-1 if communication overhead for consistency maintenance is not mentioned

(b) Another co-worker suggests broadcasting the faulting page details to all the peer nodes to quickly discover which node is currently hosting that page. What is the downside of this design choice with respect to the above implementation objectives?

Name:_____GT Number:

- In a large cluster like the above setup, broadcasting faulting page details would result in significant communication overhead since the page is at best going to be in EXACTLY one peer node (or on the disk). This is not in keeping with the efficient communication requirement.

-2 if communication overhead not mentioned

c) Disk writes are several orders of magnitude more expensive compared to writing the page to peer memory. A co-worker suggests skipping writes to the disk altogether at the time of page eviction from a node since the page is anyway being stored in a peer memory. What is the downside of this design choice with respect to the above implementation objectives?

If the evicted page is dirty, and the peer node that is chosen to store the evicted page crashes, then the page is unrecoverable. This design choice would violate the "resilient to failures of nodes" objective.

+2 if answer accounts for peer node failure.

(d) For page eviction using LRU, each node must know the "age" of each page in its local DRAM. A co-worker suggests intercepting each memory access from the CPU to collect this age information. What is the downside of this design choice with respect to the above implementation objectives?

- Memory access that a process does is happening in hardware on the CPU. The OS does not see each individual memory access that a program is making. Intercepting each memory access implies involving the OS on EVERY memory access, which will be an enormous overhead. It will slow down the execution of all processes just for getting accurate age information and goes against the performance objective.

-1 if the overhead of intercepting every memory access not mentioned

(e) You decide that at the time of page fault, to make room for the incoming page, you will immediately send the evicted page to some peer memory as part of the page fault processing. What is the downside of this design choice with respect to the above implementation objectives?

- Page eviction should not be in the critical path of page fault processing.
- More efficient to bunch together a set of page evictions rather than doing it individually.
- We are only projecting the future. We might need a page that we have already sent to memory of peer node. Better to delay eviction and potentially avoid a page fault.

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

-1 if page eviction should not be in the critical path of page fault servicing not

7. **(10 min, 10 points) (DSM)**
You are helping a friend implement an efficient page-based software DSM library in a local area network wherein the nodes are connected by 100 Gbps network links. The CPU architecture of the system uses a page size of 8KB. She is interested in achieving the following implementation objectives:

- Avoid false sharing
- Optimize consistency maintenance by overlapping computation with communication
- Increase the opportunity for concurrent execution of an application's threads when they are not needing mutual exclusion for shared memory access.
- Reduce the amount of data transferred between the nodes

You offer the following helpful suggestions to her:
(a) To keep the implementation simple, you suggest using sequential consistency as the memory model. What is the downside of this suggestion with respect to the above objectives?

Ans: Because sequential consistency blocks for coherence actions on every memory access, we cannot overlap computation with communication. This also increases the amount of data transfer since we are ping-ponging 8K pages.

(All or nothing)

(b) Another suggestion you make to simplify the implementation is to use the "page" as the unit of coherence maintenance. What is the downside of this suggestion with respect to the above objectives?

Ans: Potential for false sharing: within a page, there might be a lot of different data structures. If coherence maintenance is being done at the level of a page, then we are invalidating copies of the page in several nodes, in order to allow one node to make modifications to a portion of a page. Hence, data appears to be shared, even though programmatically it is not, leading to false sharing.

-1 if false sharing not mentioned

(c) You change your mind and suggest that she use "eager" release consistency memory model. What is the downside of this suggestion with respect to the above objectives?
- consistency actions are sent to all processors

- at release point a thread is blocked until all the changes have been sent to the peers

+1 for each point

(d) You change your mind yet again and suggest using "lazy" release consistency memory model. What is upside of this suggestion with respect to the above objectives?

- Consistency actions only communicated to the next acquirer of the lock
- A thread is not blocked at release point

+1 for each point

(e) You suggest creating "diffs" of the pages a node modifies in a critical section and associate this information with the specific lock that a thread uses to make these modifications. What is upside of this suggestion with respect to the above objectives?

- Avoids false sharing
- Allows concurrent access to different portions of the same page governed by distinct locks

+1 for each point

8. **(10 min, 10 points) (DFS)**
You are implementing a disk-based distributed file system for the College of Computing. The environment is a LAN interconnected by 100 Gbps network links. You are given the following marching orders in terms of design objectives:

- Implement standard NFS stateless-server semantics
- Ensure ease of system administration (i.e., allow for churn in the user community)
- Ensure that no file server becomes a hotspot
- Reduce disk accesses as much as possible
- Use the available physical memory of the clients and servers to the fullest to enhance performance
- Assume no server failures

(a) You decide to host the files of the user community into distinct and disjoint partitions in different servers. How does this implementation decision relate to the design objectives?

Name:_____GT Number:

- Storing data in disjoint partitions in different severs can lead to ease of system administration as different user group can be served by different servers.
- This scheme would lead to the creation of hotspots. If a particular set of files becomes popular, most accesses will be directed to the server(s) hosting them.

+1 for each point.

(b) You decide to implement a file cache in DRAM. How does this implementation decision relate to the design objectives?

- Reduces disk I/O

+2 if this is mentioned

(c) For simplicity, you decide to keep the meta-data for the files at the server that hosts the files on its local disk. What are the downsides of this implementation decision?

- Creates server hotspots
- Eases system administration

+1 for each point

(d) You change your mind and decouple the server location of the metadata for a file from the server that hosts the files on its local disk. How does this implementation decision relate to the design objectives?

- Decoupling the metadata location from the file location will help in alleviating "hotspot" creation.
- Since the metadata of the hot files can be spread across multiple manager nodes, the node that hosts the file no longer fully bears the brunt of limited memory resources required to maintain the metadata.

+1 for each point

(e) You decide to keep state information at the server on open files as to which client(s) have them in their respective DRAMs. How does this implementation decision relate to the design objectives?

- Uses ALL available memory (clients and servers) fully for caching files
- Reduces server hotspots
- Reduces disk I/O
- Increases system administration (dealing with client failures)

CS 6210 Fall 2020 Test2 Solution

Name:_____GT Number:

+2 if any two reasons are given