

# CS 6210 Spring 2015 Final

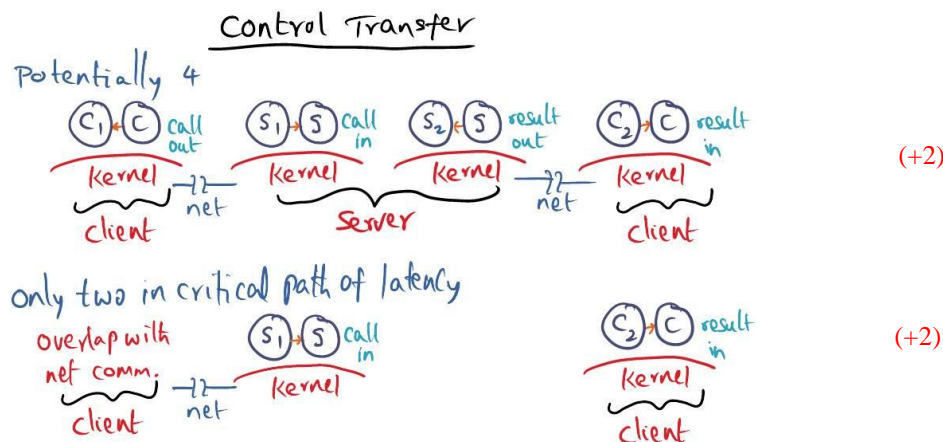
Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

2. (10 mins, 16 points) (**Distributed Systems - Latency limits, Active Networks, Ensemble/NuPr1**) (Concise bullets please)

(a) Thekkath and Levy suggest using a shared descriptor between the client stub and the kernel to avoid the data copying overhead during marshaling the arguments of an RPC call. Explain how this works.

- Each element of the shared descriptor contains:
  - <start address, number of contiguous bytes> (+1)
- Client stub fills up the descriptor at point of call: (+1)
  - <&arg1, length>, <&arg2, length>, ...,
- Kernel assembles the arguments into the kernel buffer without an extra copy by the client stub (+2)

(b) There are potentially four control transfers during the execution of an RPC. Enumerate them and identify the ones that are in the critical path of the RPC.



(c) List the pros and cons of the "Active Network" vision. (4 bullet points)

Pros:

- Flexibility
- Potential for reducing network traffic (e.g., using multicast/broadcast)

Cons:

- Protection threats
- Resource management threats

(+1 for any valid bullet pro or con)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

(d) Why does a modular approach work for designing large-scale VLSI circuits but breaks down for designing large-scale software systems? (be concise)

- Modularity in software implies layering, this could potentially lead to inefficiencies depending on the interfaces between the layers
- Building modular VLSI circuits is akin to assembling lego blocks together; building blocks are electrically connected in a VLSI circuit and current passes freely among the components; no interfacing overhead as in software components

(+2 if each of the above bullets are verbalized in some form)

3. (10 mins, 10 points) (**Distributed Objects - Java, Spring Kernel**) (one sentence answer for each part)

(a) What is the purpose of the Interface Definition Language (IDL) in software construction?

Allows expressing interfaces in a language independent manner facilitating third party software development.

(+2 for any reasonable statement along the above lines)

(b) What is the purpose of the "subcontract" mechanism in Spring?

To simplify marshaling/unmarshaling by the client/server and allow dynamic location/relocation of servers in a distributed setting.

(+2 for any reasonable statement along the above lines)

(c) What is the Spring kernel approach to "extensibility"?

Use of a microkernel based design, extensibility via servers above the microkernel, and subcontract mechanism for location transparency.

(+2 for mentioning microkernel-based design and servers above the microkernel)

(d) How does parameter passing differ between Java RMI and local object invocation?

Object references in parameters are passed by value.

(+2 for any reasonable statement along the above lines)

(e) Name one important virtue of object-oriented programming.

Re-use via inheritance, Modularity, containment of bugs, ease of evolution

(+2 for any reasonable statement along the above lines)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

4. (15 min, 12 points) (GMS, DSM)

(a) Assume 4 nodes (N1 through N4) each with 16 physical page frames. Initially none of the nodes have any valid pages in their physical memory. N1 and N2 each make a sequence of page accesses to 32 distinct pages all from the disk. N3 and N4 are idle.

What is the state of the cluster (i.e. local and global caches at each node) at the end of the above set of accesses?

**Note: Each of N1 and N2 make 32 accesses (I misstated this to two students when they came to clarify their doubt: Emily and Thomas - their answer assumes cumulatively 32 accesses)**

All of N1 and N2 memories contain only LOCAL pages; (+3)

All of N3 and N4 memories contain only GLOBAL pages (+3)

(b) Consider the following scenario: N3 acquires a lock L; at the point of acquire, N3 receives a notice to invalidate page P; upon access to P inside the critical section, N3 gets the pristine copy of P, and obtains diffs for the page  $P_{d1}$ , and  $P_{d2}$  for the page P from previous lock holders N1 and N2, respectively. How does N3 create a current version of the page P?


- N3 gets a pristine copy of P
- It applies the diffs ( $P_{d1}$ , and  $P_{d2}$ ) IN THE ORDER OF PRIOR LOCK acquisitions to P to create an up to date copy. (-4 if the phrase "IN ORDER OF PRIOR LOCK acquisitions" missing)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

5. (15 mins, 12 points) (**Distributed File System - xFS**) (Concise bullets please)

(a) What is a "small file write problem" with the software RAID approach? How is this solved?

- 
- A small file is striped across multiple disks => high overhead for reads/writes to small file
  - Log-structured file system avoids this problem by recording changes to files in a single "log" data structure owned by the file system and persisted by striping to the RAID.

(+1.5 for each correct answer along the above lines)

(b) What are the advantages of decoupling the location of data and meta-data associated with files in a distributed file system?

- Meta data management for "hot files" distributed
- Caching and serving the files to clients of "hot files" distributed

(+1.5 for each correct answer along the above lines)

(c) Let us assume file f1 is a popular file accessed by several clients. How is the node that hosts the file in its disk prevented from being overloaded by requests for this file in xFS?

- The meta data manager has a record of all the clients that has a copy of the file f1
- The request is routed to one of the client nodes that has a copy of the file f1 thus reducing the host that has the disk copy of f1

(+1.5 for each correct answer along the above lines)

(d) What happens when a client wishes to write to a file that is currently actively shared for reading by several peer clients in xFS?

- The meta-data manager for the file invalidates the read-only copies (if any) of the file from the client caches
- Upon receiving all the acks from the clients that have read copies, the manager grants permission to the requesting node for writing to the file.
- The granularity for read/write is at block level (not entire file)

(+1.5 for each correct answer along the lines of the first two bullets)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

6. (15 min, 15 points) (**Failures, LRVM, Rio Vista, Quicksilver**)

(a) Does the use of "transaction" in LRVM ensure that changes made to persistent data structures survive machine failures and/or software crashes? Explain why or why not.

Q6

- It does not (+1)
- The intent of LRVM is to provide a consistent point for server recovery upon crashes. (+2)
- Persistence for data structures modified between "begin-end" transaction is guaranteed only if the transaction has successfully flushed the in-memory re-do logs to the disk and written a commit record. (+2)
- The consistent point for resuming the server is derived by looking at the last commit record that exists on the disk.

(the last bullet point is more for understanding and not expected for getting full credit)

(b) Rio Vista implements the LRVM semantics. Yet it does not have "redo" logs as in the original LRVM system. Explain why.

- Redo logs are necessary if the transaction has committed and the data segments have NOT yet been updated with the changes present in the redo logs. (+1)
- Rio Vista is built on top of Rio which is a battery backed file cache. (+2)
- The "data segments" of LRVM are persistent by construction since they are in the file cache. (+2)
- Therefore, there are no redo logs.
- Rio Vista writes an undo log which is also persistent by construction.
- If the system crashes and recovers and if there are undo logs present in the file cache, they are applied to the data segments to restore the state to "before" the transaction.

(c) Both Quicksilver and LRVM are using the idea of transaction to provide recoverability of persistent data. Enumerate two meaningful differences between the approaches used in the two systems.


- The log maintenance in LRVM is per process, while the log maintenance in Quicksilver is for all the processes running in any particular node. (+2.5)
- The recovery management in Quicksilver is for a transaction that spans multiple nodes of the distributed system while it is only for a given node in LRVM. works across the entire distributed system NOT just for a single node as in LRVM. (+2.5)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

7. (15 min, 12 points) (**Internet Scale Computing**)

(a) List the tradeoffs between replication and partitioning in architecting the data repositories of giant scale services. (Concise bullets please)

- 
- With replication we can have complete harvest despite failures, while with partitioning we cannot have complete harvest in the presence of failures. (+1.5)
  - Since replication gives more control on harvest it gives more choice for the system administrator (give full harvest for some types of queries or partial harvest for some types of queries) when DQ goes down due to failures, replication is preferred beyond a point for building scalable servers. (+1.5)

(b) How does the MapReduce runtime deal with redundant Map and Reduce workers that could potentially be working on the same "split" of the data set for the map and reduce operations, respectively?

- The master node knows the mapper nodes; if some are assigned duplicate shards, the early finishing mapper's result is used; the rest ignored. (+1.5)
- The master node knows the reducer nodes; the master is responsible for "renaming" the local file produced by a reducer to the final "named" output file; once again master node does the renaming when the first redundant reducer finishes, and then ignores the rest of the redundant reducers. (+1.5)

(c) Distinguish between web proxies and content distribution networks.

- Web proxies "pull" content and store it locally to serve the clients. (+1.5)
- With CDNs, the content is "pushed" from the origin servers to the CDN mirrors. (+1.5)

(d) "Coral CDN avoids tree saturation." Explain this statement with concise bullets.

- The intent in Coral as in traditional CDNs, is to store the key K in node N, where  $K \approx N$ . (+1.5)
- However, if the node is loaded or full then an earlier node in the path from the source to the destination that is NOT loaded or full is chosen for keeping the meta data associated with a particular content. (+1.5)

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

8. (10 min, 10 points) (**Real time computing - TS Linux, PTS**)

(a) What are the goals of the "firm timer" design in TS Linux? (Concise bullets please)

- Provide accurate timer for applications that need a finer precision (e.g., multimedia applications) by using one-shot timers. **(+2.5)**
- Do not incur the performance penalty of one-shot timer interrupts where possible by providing an overshoot window that would allow executing the handler for the one-shot timer (without an interrupt) at a close enough preceding periodic timer interrupt or a system call. **(+2.5)**

(b) How does the PTS computational model (with PTS threads and channels) differ from a Unix distributed program written with sockets and processes? (Concise bullets please)

- Many to many connections allowed in PTS **(+1)**
- Time is a first class entity managed by PTS **(+2)**
- Data in channels can be persisted on demand by the application in PTS **(+2)**

# CS 6210 Spring 2015 Final

Name: \_\_\_\_\_ Kishore \_\_\_\_\_ GT Number: \_\_\_\_\_

9. (10 min, 12 points) (**Security and Andrew File System**) (Concise bullets please)

(a) "AFS does not address confinement of resource usage by the clients." Explain this statement.

Q9 A client can flood the LAN with requests to the servers and thus making the system vulnerable to DOS attacks. (+3)

(b) How does AFS avoid replay attacks?

- Upon being contacted by a client, the server challenges the client by sending a random encrypted with a key known only to the client. (+1.5)
- A replay attacker will not be able to successfully decrypt the message to retrieve the random number. (+1.5)

(c) In AFS, "clientident" has to be always sent in plaintext. Why?

The security infrastructure of AFS uses private key encryption. (+1)

In this system, the server has to be able to find the right key to use to decrypt a client message. (+1)

ClientIdent helps the server to find the right key. Hence plaintext for the ClientIdent. (+1)

(d) In AFS, how is over-exposure of client identity and passwords avoided?

- Username and password used only once per login session. (+1)
- Secrettoken is used as a ClientIdent only for the duration of one login session. (+1)
- The handshake key extracted from the ClearToken is used only once per establishment of an RPC session. (+0.5)
- Each new RPC session uses a new session key (SK) for encryption. (+0.5)