CS 6210 Fall 2010 Final

Name: Kishore GT Number: _____

Wednesday Decembet 13, 2010 (4:00 PM to 6:00 PM - UTC+9 Seoul)

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. Where appropriate **use figures** to convey your points (a figure is worth a thousand words!)
6. **Illegible answers are wrong answers. WE MEAN IT!!**
7. Please look through the whole test before starting so that you can manage your time better.

Good luck!

Question number	Points earned	Running total
1 (Max: 1 pts)		
2 (Max: 10 pts)		
3 (Max: 10 pts)		
4 (Max: 10 pts)		
5 (Max: 10 pts)		
6 (Max: 10 pts)		
7 (Max: 9 pts)		
8 (Max: 10 pts)		
9 (Max: 5 pts)		
10 (Max: 5pts)		
11 (Max: 5 pts)		
12 (Max: 15 pts)		
Total(Max: 100 pts)		

1. (1 min, 1 point)
கிஷோர் is the way you write Kishore in which language?

a) Korean
b) Japanese
c) Russian
d) Hindi
e) Tamil
f) Chinese

For 1 point extra credit, write Kishore in at least 3 of the above choices other than the one already given to you (give the name of the language as well).

korean: 키쇼르 Hindi: किशोर

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

2. (10 min, 10 points) (Global Memory System (GMS))
   a) (5 points) Consider the following scenario:
      - Two nodes in the cluster R and Q
      - Node R:
        - Age of oldest local page = 40
        - Age of oldest global page = 30 → Let this be page X
      - Node Q:
        - Age of oldest local page = 45 → let this be page y
        - Age of oldest global page = 35
      - Node R has a page fault. The faulting page is on the disk.

   Explain what will happen during the service of this page fault as per the GMS replacement algorithm

- R sends x (oldest global page on R) to Q
- R gets faulting page from disk
- R's L count up by 1; G count down by 1
- Q sends y (oldest page in Q) to disk if dirty else discards it
- Q places x in its global cache; Q's L ↓1; Q's G ↑1

   b) (5 points) Recall that MinAge is the minimum age of the M oldest pages that are expected to be evicted from the system in the next epoch. There are 3 nodes Ni, Nj, Nk.
   Consider the following scenario:
   MinAge = 23
   M = 10 (number of pages that will be evicted during this epoch)
   Wi = 0.2 (weight of node Ni)
   Wj = 0.5 (weight of node Nj)
   Wk = 0.3 (weight of node Nk)

      - Ni has a page fault and has to evict a page P.
      - The age of this page P is 51.

   What will Ni do to the page P and **why**?

- P's age > MinAge
  ⇒ P is expected to be discarded in this epoch
- Ni will write P to disk if dirty; else simply discard P

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

3. (10 min, 10 points) (Distributed Shared Memory - TreadMarks)
   a) (5 points) Consider the following execution happening in Treadmarks. Assume initially that both processors P1 and P2 have copies of the pages X, Y, and Z.

```
P1:
      Lock (L)
            Step 1:  Updates to page X;
            Step 2:  Updates to page Y;
            Step 3:  Updates to page Z;
      Unlock (L)

Subsequently, processor P2 executes the following code:
P2:
      Lock (L)
            Step 4: Read page X;
      Unlock (L)
```

(i) Explain what happens on P1 at Step 1

- create twin for X; call it X'
- write updates to X

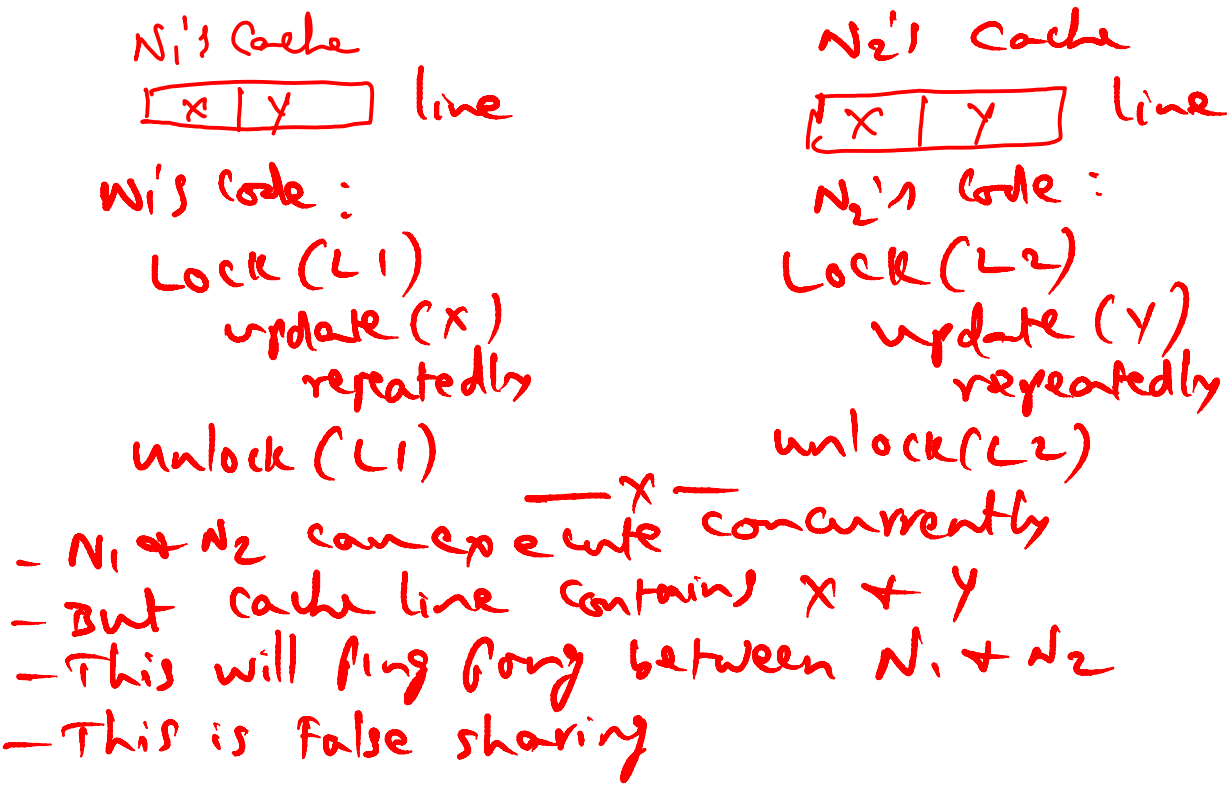
(ii) Explain what happens on P2 at Step 4

- P2 at the point of lock acquisition invalidates X, Y, Z pages
- At step 4, P2 goes to P1 and obtains diff(X) : created by P1 at the point of unlock(L)
- P2 applies diff(X) to its copy of X and makes X valid & completes read operation

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

b) (5 points) Explain false sharing with a code example.



4. (10 mins, 10 points) (xFS) All the questions pertain to xFS

a) (2 points)Choose **ONE TRUE statement from the following**

I. The log file is striped across **all** the storage servers in the entire distributed system
II. The log file is stored on exactly **one** of the storage servers in the distributed system
III. The log file is striped across a **subset** of the storage servers in the entire distributed system.

b) (2 points) Choose **ONE FALSE statement from the following**

I. The in-memory cache for a file is always at the manager node.
II. Small file write problem is solved in xFS.
III. The manager node for a file for meta data management is chosen dynamically.

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

c) (2 points) Choose **ONE TRUE statement from the following**

I. A file read request from a client **always** involves **at least one** network access.
II. A client directly goes to a peer to get a copy of a file from the peer's in-memory cache.
III. If a client needs to write to a file it has to first get write ownership by contacting the manager for the file.

d) (2 points) Choose **ONE FALSE statement from the following**

I. If a client currently has write permission for a file block then no other client can read that block unless the first client gives up the write permission.
II. Multiple clients could have write ownership to different blocks of the same file at the same time.
III. Writes to a file are buffered in memory locally by a client and eventually committed to the storage servers.

e) (2 points) Explain what is meant by a log structured file system.

– updates to independent files written as time ordered log records sequentially to a "log file"

log file [f1, block#, newContent][f2, block#, newContent]....

– writes are efficient: random writes converted to sequential writes
– reads require "constructing" the file from log records

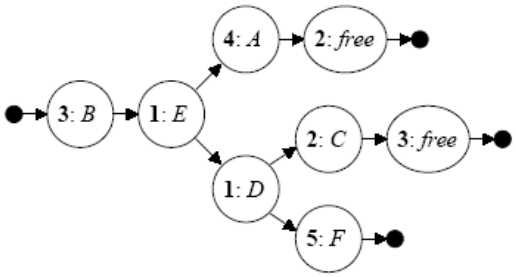
CS 6210 Fall 2010 Final

Name:_____GT Number: _____

5. (10 min, 10 points) (Rialto CPU scheduler)

(a) (5 points) Consider a scheduling graph with a base period of 10 ms as shown below. Assume that the original request by A was

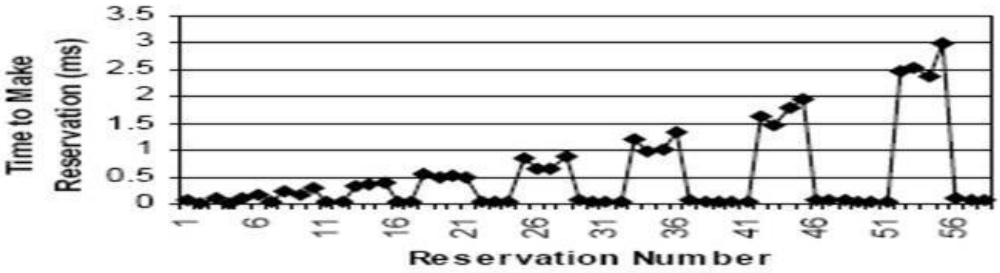
A: 6 ms every 30 ms



Does the schedule shown in this Figure really satisfy A's request? You have to justify your answer (**just saying yes or no gets no credit**)

- A needs 6ms every 30ms
- In every 20 ms it should get $6 \times \frac{20}{30} = 4$ ms
- This is what it gets ⇒ Schedule is good

(b) (5 points)The graph below shows the scheduling overhead by plotting the time to make a reservation (y-axis), against the specific reservation request (x-axis). Why do some reservations (e.g., between request numbers 41 to 46) take more time than others (e.g., 46 to 51)?



- reservation request results in the reconstruction of the scheduling graph

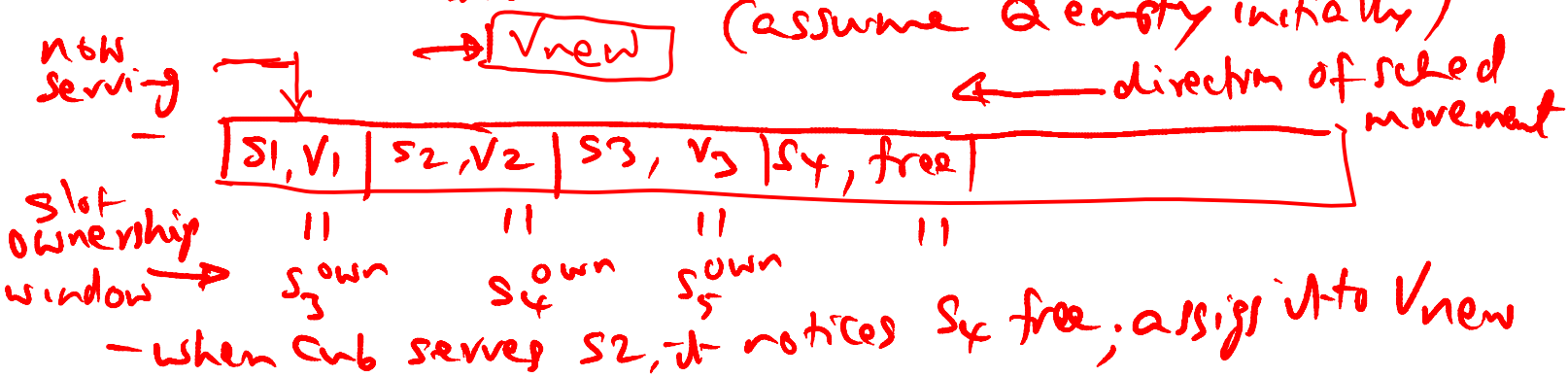
CS 6210 Fall 2010 Final

Name:_____GT Number: _____

6. (10 min, 10 points) (Tiger Disk scheduling)

a) (5 points) The controller receives a new request from a viewer for a movie. Write down the steps (**do not write an essay**) by which this viewer gets inserted into the global schedule.

- Controller sends viewer request to the cub that has first block of movie & Vnew
- Cub puts this in its new viewer request Q (assume Q empty initially)



- when Cub serves S2, it notices S_4 free, assigns it to Vnew

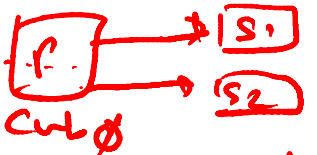
b) (5 points) Explain the scheme used in Tiger to

(i) ensure that the load is uniformly distributed to all the cubs independent of the number of viewers

- All movies striped across all disks
- Every cub participates in serving every movie for every viewer

(ii) overcome failure of a cub

- Every primary block split into two secondary blocks & stored on two other cubs



- Cub 0 fails; cub 1 & cub 2 picking slack
- Every cub serving a viewer notifies next two successor cubs to detect failure

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

7. (10 min, 9 points) (LRVM)

(a) (5 points) A subsystem needs persistence for a data structure called "inode"

map (region, &inode); // map inode data structure to region on the disk

The application wants to do the following:

update (inode)
if (some_condition_satisfied) store it in persistent memory
else restore original value in inode;

Write down the pseudo code for the above action using LRVM. You do not have to worry about the exact syntax. It is sufficient if you show in some reasonable manner how persistence is achieved, if needed; and how the original contents are restored, if needed.

```
begin_xact(tid, restore);
set_range(tid, &inode, sizeof(inode));
update(inode)
if <predicate>
    end_xact(tid, flush);
else abort_xact(tid);
```

(b) (4 points)LRVM lives above the OS; thus the VM subsystem and LRVM subsystem are independent of each other. Give two ways this separation can hurt the performance of LRVM.

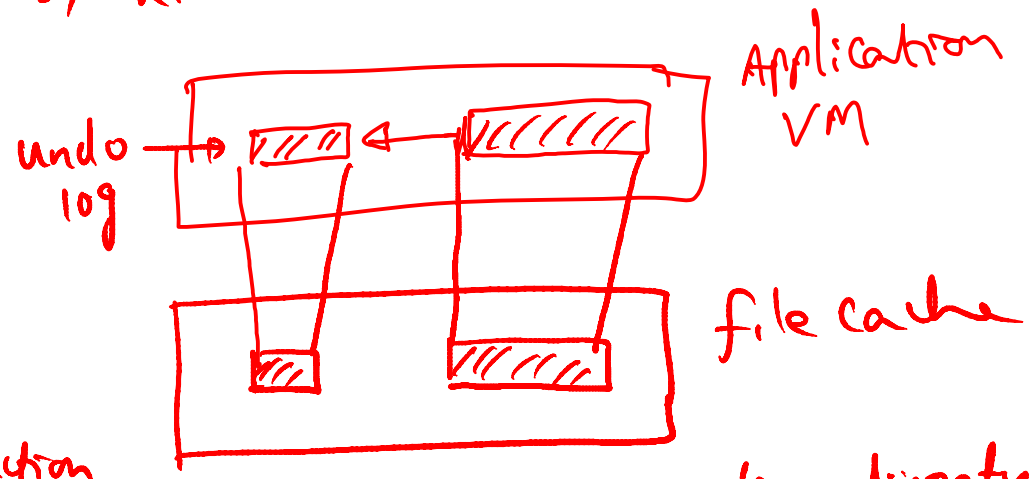
- starting latency for copying segment into VM
- dirty page of LRVM swapped out by VM due to memory pressure

CS 6210 Fall 2010 Final

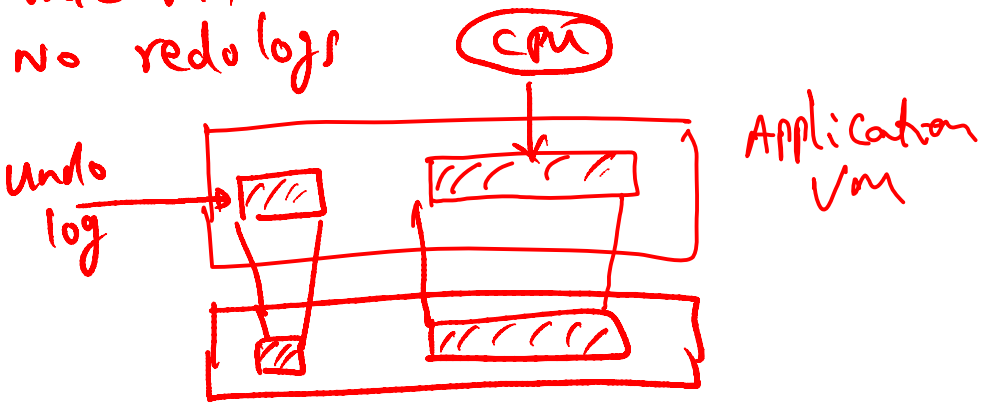
Name:_____GT Number: _____

8. (10 min, 10 points) (RioVista)
With some simple pictures show how Vista recoverable memory works on top of a battery-backed Rio file cache.

At Begin transaction:
- make copy of "before" image of range of virtual addresses to be modified in a VM buffer backed by Rio File Cache



During xaction
- perform writes during transaction directly into VM backed by file cache
- No redo logs



Commit
- throw away undo log

Abort - restore from undo log

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

9. (5 min, 5 points) (Quicksilver)
(5 points) Explain the rationale for using transaction in OS design and implementation as is done in Quicksilver.

- Subsystems may have recoverable state (file system)
- subsystems may have to do clean up (orphan windows of window manager)
- Subsystems may need to report errors in a reliable + consistent manner (communication library)
- xaction unifies + meets all these needs

10. (10 min, 5 points) (Map/Reduce)Write a Map/Reduce program that counts the URL access frequency. The input to the map/reduce system are web logs of page requests. The final output should be <URL, total-count>

Variety of solutions accepted

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

11. (5 min, 5 points) (Giant Scale Service)
DQ Principle: If D is data per query and Q is the number of queries per second, then the product D*Q tends towards a constant for a fully utilized system. How can the DQ principle be used to come up with policies for graceful degradation of the system under excess load?

Q relates to yield => number of successful queries processed

D relates to harvest => fraction of complete data served per query

Graceful degradation policy:
- DQ is a constant
- admission control to reduce Q but keep D same
- fidelity control to reduce D to allow increase in Q

12. (10 min, 15 points) (Coral)
Consider an 8-node system with node-ids: 0, 1, 2, 3, 4, 8, 10, 14.
Node 14 wants to do a "get_key" operation for some key which matches with node-id 2. Initially, assume the following about the routing tables at the different nodes:

- Node 14 has IP addresses for node 4 and 8.
- Node 10 has IP addresses for 8 and 14
- Node 8 has IP addresses for nodes 2 and 3
- Node 4 has IP addresses for nodes 0 and 1
- Node 3 has IP addresses for nodes 2 and 4
- Node 2 has IP addresses for nodes 0, 1, and 3
- Node 1 has IP addresses for 0 and 3
- Node 0 has IP addresses for 2 and 4

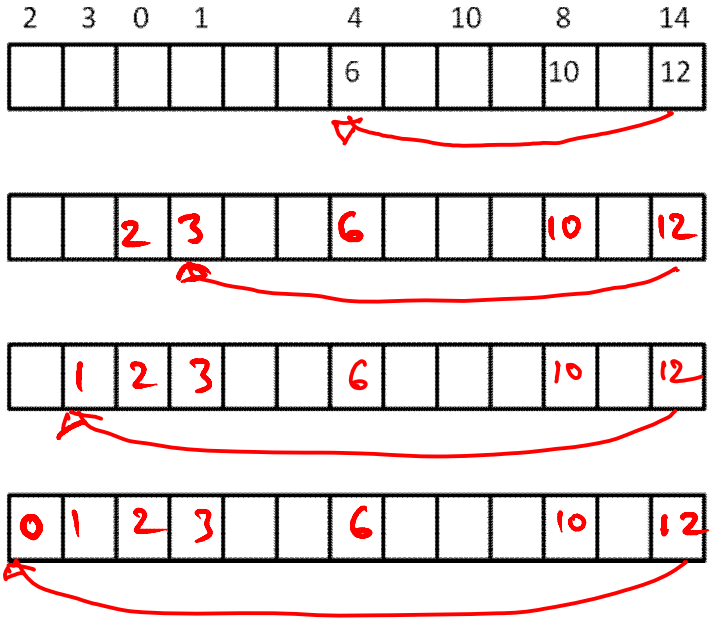
Show pictorially the evolution of the routing table at Node 14 in implementing the key-based routing in Coral. To get you started, the **first row** in the figure below shows the routing table at Node 14 at the start. The contents of the table are the XOR distances of that node from the destination (i.e., node 2). For e.g., the distance between nodes 4 and 2 is 6. Recall that the "distance" between any two nodes is the XOR distance of the node-ids.

CS 6210 Fall 2010 Final

Name:_____GT Number: _____

For each row, as a progression of the key-based routing

- Show what RPC call is made
- Show what is returned as a result of the RPC
- Show the changes that result in the routing table because of what is returned



RPC call: nodes $\frac{6}{2}$-distant
response: nodes 0, 1

RPC call: nodes $\frac{3}{2}$ distant
response: nodes 0, 3

RPC call: node 0 distant
response: node 2

RPC call: get_key
response: value