CS 537: Introduction to Operating Systems
Fall 2015: Midterm Exam #4
Tuesday, December 15th 11:00 – 12:15

Advanced Topics: Distributed File Systems

SOLUTIONS

This exam is closed book, closed notes.

All cell phones must be turned off.

No calculators may be used.

You have 1 hour and 15 minutes to complete this exam. The exam has a total of 64 questions.

Write all of your answers on the accu-scan form with a #2 pencil.

These exam questions must be returned at the end of the exam, but we will not grade anything in this booklet.

Please fill in the accu-scan form with your Last Name, First Name and Student Identification Number; remember to fill in the corresponding bubbles as well.

This exam has multiple versions. To make sure you are graded with the correct answer key, you must identify this exam version with a Special Code in Column A on your accu-tron sheet. Be sure to fill in the corresponding bubble as well. Your special code is shown on the next page.

Good luck!

This exam has multiple versions. To make sure you are graded with the correct answer key, you must identify this exam version with a Special Code in Column A on your accu-tron sheet. Be sure to fill in the corresponding bubble as well. Your special code is: 1.

Part 1: Which Communication Protocol is this? [3 points each]
QUESTIONS 24-29 on exam #2
For each of the following questions, designate if the statement is True for
(a) only the UDP protocol
(b) only the TCP protocol
(c) both UDP and TCP
(d) neither UDP nor TCP.

1. If using this protocol, a message may be **lost** between the sending process and the receiving process and as a result the receiving process may never receive that message.

Answer: a. The UDP protocol does not do anything to resend lost messages, but TCP will resend until an ACK is received.

2. If using this protocol, a message may be **corrupted** between the sending process and the receiving process and as a result the receiving process may receive a corrupted message.

Answer: d. Neither protocol will allow a corrupted message to be delivered to a receiving process; both protocols use checksums to ensure that corrupted messages are not delivered to the receiving process.

3. With this protocol, if a sender does not receive an **acknowledgement** for a message, it means the receiver did not receive the intended message.

Answer: d. UDP does not use acknowledgements; although TCP does use acknowledgements, not receiving one could mean that the ACK was dropped and the receiver did still receive the intended message.

4. With this protocol, if a sender does not receive an acknowledgement for a message, the sender will **resend** that message.

Answer: b. TCP resends messages if an ack is not received.

5. With this protocol, the receiver is able to discard **duplicate** messages by tracking the checksum for each previously received message.

Question discarded. The question meant to ask whether or not this what either protocol does (answer: neither do), but the words "is able" in the question made it sound like the question was asking what could instead be possible.

6. **RPC** can be implemented on top of this protocol.

Answer: c. RPC can be built on top of either protocol. If it is built on UDP, the RPC protocol must do the extra work to ensure it discards duplicates and resends lost messages. If it is built on TCP,

there is some inefficiency because TCP sends ACKs back when RPC is going to be sending back a reply very soon. Both approaches have their pros and cons.

Part 2: Which Distributed File System is this? [3 points each]
Questions 30 – 38 on exam #2.
For each of the following questions, designate if the statement is True for
(a) only NFS
(b) only AFS
(c) both NFS and AFS
(d) neither NFS nor AFS.
Assume NFS refers to NFS version 2 (or NFSv2) which is the NFS protocol we discussed in lecture.

7. With this file system, clients can access files either in their **local file system** or on a remote server.

Answer: c. Both file systems work with the mount protocol so that clients can transparently access files in their directory namespace that might be located in different file systems, whether a local file system (e.g., ext3) or a distributed file system using a remote server (e.g., NFS or AFS).

8. Files and directories may be easily **moved** from one server to another.

Answer: b. AFS allows volumes (subtrees of the directory space) to be moved to a new server (e.g., for better load balancing across servers).

9. Servers in this file system are **stateless**.

Answer: a. NFS servers do not remember any state about particular clients that are interacting with it or on-going operations. AFS does track state: callbacks for clients that currently have each file open.

10. Requires that clients fetch the **entire file** when they open a file (unless they already have that file cached locally).

Answer: b. NFS does not mandate any reading or prefetching when a file is opened; AFS does.

11. Requires that clients push their **write operations** immediately to the server.

Answer: d. Neither file system requires clients to push their write operations immediately; both require writes to be pushed out when the file is closed.

12. Writes to one file from one client can be **intermingled** with writes to that same file from another client.

Answer: a. If two clients A and B both write to two blocks 0 and 1 from the same file, with NFS it is possible for client A's block 0 and client B's block 1 to end up being the final version of that file; AFS guarantees that the server will store either A's two blocks or B's two blocks.

13. Clients are guaranteed to read and write to the **same version** of a file for the entire time between when that client opens and closes that file.

Answer: b. NFS will re-read blocks of a file after that file has been opened if the file is changed on the server (and the attribute cache has expired); AFS does not re-read blocks of a file after that file is opened.

14. Increasing the time between getattr (or STAT) calls to the server increases the **scalability** of the system.

Answer: a. NFS uses getattr calls; if the time between getattr's is too short, the server is flooded with these getattr requests and becomes a bottleneck for the entire distributed system (limiting the number of clients that can use one server); by increasing the time between requests, the server can handle more clients.

15. If the server does not contain sufficient memory to track the **callbacks** for a particular file, the server can simply discard (i.e., forget) those callbacks.

Answer: d. Although AFS does use callbacks, the server cannot just throw callbacks away; if it needs to discard a callback for any reason, the server needs to break that callback and notify the client so the client will not assume it can trust its cached contents later.

Part 3: True/False for MapReduce and GFS [3 points each]
Questions 1-23 on exam #2.
Designate if the statement is True (a) or False (b).
For the GFS file system, assume a replication level of 3 is always used.

16. An **idempotent** operation has the same result whether it is executed zero, one, or more times.

Answer: false. Idempotent operations give the same result when executed one or more times.

17. With the MapReduce programming framework, each of the M mappers is responsible for reading sequentially through **1/M-th** of the amount of input data.

Answer: true. Each mapper is assigned a sequential contiguous portion of the input data (hopefully a chunk in GFS).

18. With MapReduce, each of the M mappers uses RPC to directly send **partitioned** data to each of the R reducers.

Answer: false; each of the R reducers READs their portion of the data from all of the mappers that generated the data (the intermediate generated data may be stored on the mapper's local disk).

19. With MapReduce, each of the R reducers is responsible for writing sequentially to **1/R-th** of the amount of output data.

Answer: false; different reducers may write much more or much less than 1/Rth of the output data; for example, one reducer may be responsible for records matching WI and another for records matching MN, and there may be many more WI than MN records.

20. One worker machine may be responsible for running **multiple mappers** within a single MapReduce job.

Answer: true. In most cases, there will be many more mappers than worker machines; when a worker finishes one mapper, it moves on to executing a different mapper; this improves load balancing (since some mappers may take longer to run than others).

21. GFS is used to store **input, intermediate, and output files** for MapReduce jobs.

Answer: false. GFS is only used for input and output files; intermediate files are just stored on the local disk.

22. Each mapper function must perform **network I/O** in order to read its input data.

Answer: false. Ideally, a mapper is run on a machine storing one of the replicas of its input data (stored in GFS).

23. Each reduce function must perform **network I/O** in order to read its input data.

Answer: true; each reducer must contact every machine that ran a mapper to fetch the appropriate data for this reducer.

24. Each reduce function must perform **network I/O** in order to write its output data.

Answer: true; with 3 replicas in GFS, at least two of the replicas will be on other machines.

25. Increasing the value of M increases the amount of **parallelism** in the MapReduce job.

Answer: true; as M increases, there are more mappers, which can each be run in parallel.

26. If a **reducer fails**, that reducer is run again, as are the mappers responsible for generating the input for that reducer.

Answer: false; if a reducer fails, the input to the reducer still exists (on the machines that ran the mappers) and the new reducer can simply refetch that input.

27. If a **mapper runs slowly**, the master can start a duplicate mapper and still obtain correct results.

Answer: true. The mapping function is idempotent; re-run it and it will generate the same results.

28. If a **reducer runs slowly**, the master can start a duplicate reducer and still obtain correct results.

Answer: true. The reduce function is idempotent; re-run it and it will generate the same results.

29. MapReduce and other GFS-specific workloads often **overwrite** previously written data.

Answer: false; the workloads append a lot and very rarely overwrite older data.

30. MapReduce workloads require high I/O **throughput**.

Answer: true; the workloads don't require low latency, but they do need to read and write a lot of data in each unit of time.

31. In GFS, the **scalability** of the system is likely to increase as the size of chunk is increased.

Answer: true; as the chunk size is increased, the master has fewer entries to track and thus it can handle more files.

32. In GFS, clients must communicate with the **master** every time they read or write to a file.

Answer: false; clients communicate with the master only when they open a file; at that point, the client has obtained all the meta-data for the file (i.e., the chunk ids and the chunk servers for those chunks) and the client can thereafter communicate directly with the chunkservers.

33. In GFS, clients may only **read** from the primary replica.

Answer: false; clients can read from any of the replicas.

34. In GFS, the **master** is responsible for mapping file names to logical chunk numbers and their chunk servers.

Question discarded; although the master does track all of this information, the word "responsible" in the question could imply that the master has the most-accurate knowledge of all of this; however, only the chunk servers themselves know for certain which chunks they hold.

35. In GFS, the master persists updates to the **file name space** to the disk on other backup masters.

Answer: true; the mapping between file names and chunk ids is updated persistently so that another master can take over if this one dies.

36. In GFS, the master persists updates to the **chunk map** to the disk on other backup masters.

Answer: false; this information is kept only in memory; if the master dies, this information is recreated by asking each chunk server for the valid chunk ids that it hosts.

37. In GFS, the master is responsible for determining the **order** in which different clients update replicas.

Answer: false; the primary replica determines this order (so that the master is not a bottleneck).

38. In GFS, it is possible that only a subset of a chunk's replicas successfully performs an **append** operation.

Question discarded, because we didn't cover this in lecture thoroughly enough. Answer: true; the primary tells the secondary replicas the order in which appends from different clients should be performed, but it is possible that some of those replicas will run into problems and the primary is not responsible for ensuring each replica successfully performed the append; the primary simply notifies the client of these errors and the client is responsible for retrying the append.

Part 4. Distributed File System Consistency [2 points each]
The next questions explore the cache consistency behavior of AFS and NFS**.**

The two traces on the next two pages are identical; you should use one trace for answering how AFS behaves and one trace for answering how NFS behaves.

Each trace contains three clients that each generate file opens, reads, writes, and closes on a single file 'a'. The leftmost column shows the server; the next 3 columns show the actions being taken on each of the three clients, c0, c1, and c2. The content of the file is always just a single number. Time increases downwards, with at least 5 seconds between each operation.

Opening a file returns a file descriptor, which is the first argument to each call of read, write, and close (i.e., read:fd, write:fd, and close:fd). The write call also designates the new value to be written (i.e., write:fd -> newvalue).

There are two types of questions for you to answer. Questions of the format "`read:fd -> value?`" on c0, c1, and c2 ask you to determine the value that will be read on that client. Questions of the format "`file:a contains: ?`" on the server ask you to determine the value on the server at that point in time.

You may find it useful to record the contents of the file on the server at every "interesting" point in time.

For each protocol, assume that the server and clients have sufficient memory such that no operations are performed unless required by the protocol.

For questions Q39-Q44, your choices are:
a) 0
b) 1
c) 2
d) 3
e) None of the above

For questions Q45-Q51, your choices are:
a) 2
b) 3
c) 4
d) 5
e) None of the above

For questions Q52-Q57 your choices are:
a) 0
b) 1
c) 2
d) 3
e) None of the above

For questions Q58-Q64, your choices are:
a) 2
b) 3
c) 4
d) 5
e) None of the above

Questions 39-51: AFS Protocol

Remember, for questions Q39-Q44, your choices are:

a) 0
b) 1
c) 2
d) 3
e) None of the above

For questions Q45-Q51, your choices are:

a) 2
b) 3
c) 4
d) 5
e) None of the above

Determine the results if this workload is run on top of the **AFS** distributed file system.

```
Server                  c0                          c1                      c2
file:a contains:0
                                                    open:a [fd:0] 0 until close
                                                                            open:a [fd:0] 0 until
                        open:a [fd:0] 0 until close
                                                                            read:0 -> (Q39) 0 (a)
                                                                            write:0 -> 1

                                                    read:0 ->(Q40) 0 (a)
                                                    write:0 -> 2

                        read:0 -> (Q41) 0 (a)
                        write:0 -> 3
file:a contains:?(Q42) 0 (a)
                                                                            close:0 –flushes 1
file:a contains:?(Q43) 1 (b)
                                                    close:0 – flushes 2
                                                                            open:a [fd:1] 2 until
                                                                            read:1 -> (Q44) 2 c
                                                                            write:1 -> 4

                        close:0 – flushes 3

                                                    open:a [fd:1] 3 until close
                        open:a [fd:1] 3 until close
                                                                            close:1 – flushes 4
file:a contains:?(Q45) (Note: Switch to different multiple choice options) 4(c)

                        read:1 -> value?(Q46) 3 (b)

                                                                            open:a [fd:2] 4

                        close:1
                                                    read:1 -> value?(Q47) 3(b)
                                                    write:1 -> 5

                        open:a [fd:2] 4 until close

                                                                            read:2 ->(Q48) 4c
                                                                            close:2
                        read:2 -> value?(Q49) 4c

                                                    close:1 – flushes 5

                        read:2 -> value?(Q50) 4c
                        close:2
file:a contains:?(Q51) 5d
```

Questions 52-64: NFS Protocol

For questions Q52-Q57 your choices are:

a) 0
b) 1
c) 2
d) 3
e) None of the above

For questions Q58-Q64, your choices are:

a) 2
b) 3
c) 4
d) 5
e) None of the above

Determine the results if this workload is run on top of the **NFS** distributed file system.

```
Server                  c0                          c1                      c2
file:a contains:0
                                                    open:a [fd:0]
                                                                            open:a [fd:0]
                        open:a [fd:0]
                                                                            read:0 -> (Q52) 0a
                                                                            write:0 -> 1
                                                    read:0 -> (Q53) 0a
                                                    write:0 -> 2
                        read:0 -> Q54 0a
                        write:0 -> 3
file:a contains:?(Q55) 0a
                                                                            close:0 – flushes 1
file:a contains:?(Q56) 1b
                                                    close:0 – flushes 2
                                                                            open:a [fd:1]
                                                                            read:1 -> (Q57) 2c
                                                                            write:1 -> 4
                        close:0 – flushes 3
                                                    open:a [fd:1]
                        open:a [fd:1]
                                                                            close:1 – flushes 4
file:a contains:?(Q58) (Note: Switch to different multiple choice) – 4c
                        read:1 -> value?(Q59) – 4c
                                                                            open:a [fd:2]
                        close:1
                                                    read:1 -> Q60 – 4c
                                                    write:1 -> 5
                        open:a [fd:2]
                                                                            read:2 -> (Q61) -4c
                                                                            close:2
                        read:2 -> Q62 -4c
                                                    close:1 – flushes 5
                        read:2 -> value?(Q63) – 5d
                        close:2
file:a contains:?(Q64) -5d
```

Congratulations on learning a lot about Operating Systems this semester!
Please double check that specified your Special Code in Column A on your accu-tron sheet.