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

Advanced Topics: Distributed File Systems

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]

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.
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.
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.
4. With this protocol, if a sender does not receive an acknowledgement for a message, the sender will **resend** that message.
5. With this protocol, the receiver is able to discard **duplicate** messages by tracking the checksum for each previously received message.
6. **RPC** can be implemented on top of this protocol.

Part 2: Which Distributed File System is this? [3 points each]
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.
8. Files and directories may be easily **moved** from one server to another.
9. Servers in this file system are **stateless**.
10. Requires that clients fetch the **entire file** when they open a file (unless they already have that file cached locally).
11. Requires that clients push their **write operations** immediately to the server.
12. Writes to one file from one client can be **intermingled** with writes to that same file from another client.
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.
14. Increasing the time between getattr (or STAT) calls to the server increases the **scalability** of the system.
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.

Part 3: True/False for MapReduce and GFS [3 points each]
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.
17. With the MapReduce programming framework, each of the M mappers is responsible for reading sequentially through **1/M-th** of the input data.
18. With MapReduce, each of the M mappers uses RPC to directly send **partitioned** data to each of the R reducers.
19. With MapReduce, each of the R reducers is responsible for writing sequentially to **1/R-th** of the output data.
20. One worker machine may be responsible for running **multiple mappers** within a single MapReduce job.
21. GFS is used to store **input, intermediate, and output files** for MapReduce jobs.
22. Each mapper function must perform **network I/O** in order to read its input data.
23. Each reduce function must perform **network I/O** in order to read its input data.
24. Each reduce function must perform **network I/O** in order to write its output data.
25. Increasing the value of M increases the amount of **parallelism** in the MapReduce job.
26. If a **reducer fails**, that reducer is run again, as are the mappers responsible for generating the input for that reducer.
27. If a **mapper runs slowly**, the master can start a duplicate mapper and still obtain correct results.
28. If a **reducer runs slowly**, the master can start a duplicate reducer and still obtain correct results.
29. MapReduce and other GFS-specific workloads often **overwrite** previously written data.
30. MapReduce workloads require high I/O **throughput**.
31. In GFS, the **scalability** of the system is likely to increase as the size of chunk is increased.
32. In GFS, clients must communicate with the **master** every time they read or write to a file.
33. In GFS, clients may only **read** from the primary replica.
34. In GFS, the **master** is responsible for mapping file names to logical chunk numbers and their chunk servers.
35. In GFS, the master persists updates to the **file name space** to the disk on other backup masters.
36. In GFS, the master persists updates to the **chunk map** to the disk on other backup masters.
37. In GFS, the master is responsible for determining the **order** in which different clients update replicas.
38. In GFS, it is possible that only a subset of a chunk's replicas successfully performs an **append** operation.

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]
                                                                            open:a [fd:0]
                        open:a [fd:0]
                                                                            read:0 -> value?(Q39)
                                                                            write:0 -> 1

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

                        read:0 -> value?(Q41)
                        write:0 -> 3
file:a contains:?(Q42)
                                                                            close:0
file:a contains:?(Q43)
                                                    close:0
                                                                            open:a [fd:1]
                                                                            read:1 -> value?(Q44)
                                                                            write:1 -> 4

                        close:0

                                                    open:a [fd:1]
                        open:a [fd:1]
                                                                            close:1
file:a contains:?(Q45) (Note: Switch to different multiple choice options)

                        read:1 -> value?(Q46)

                                                                            open:a [fd:2]

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

                        open:a [fd:2]

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

                                                    close:1

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

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

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.