Midterm
Fall 2000 15-712

Please write your name on the following line:

____

This midterm is open book and open notes.

You should write brief, precise, and legible answers. Rambling brain-dump answers may be punished twice — you run out of time writing them and we run out of patience reading them. :–)

There are eight questions but we will only grade five. List the five questions we should grade:

____

Name: ______________________________

1. Soft Updates involves a significant amount of mechanism to propogate metadata changes to disk in a specific order when employing write-back caching. Explain why soft updates is not needed in a log-structured file system.

   In a log-structure file system, all of the recent disk activity occurs at the end of the log, making it easy to check the consistency of the last few operations. Log-structured file systems have a notion of a "checkpoint", which is a position in the log at which all of the file system structures are consistent and complete. During crash recovery, the file system examines operations that occur after a checkpoint in the log.

   A traditional UNIX file system, in contrast, has its metadata scattered across the disk, making consistency checking hard. Soft updates has a significant amount of mechanism to ensure that on-disk metadata is self-consistent. Without soft updates, upon crash all of the metadata would need to be scanned for consistency.

Name: ______________________________

2. Callbacks are server-maintained state. Describe how consistency may be compromised if the server reboots.

   A callback is a promise that server will inform a client when the client's cache is out-of-date. Imagine a client has some cached data and the server holds a callback for the client. When the server reboots, the server loses the callback for that client. Now another client modifies the data that the first client has cached. The first client will continue to use stale data in its cache because the server didn't know to contact it and inform it that its data has become invalid.

Name: ____________________________________

3. NASD and xFS both focus on scalability. Explain why xFS is probably more scalable and why NASD is probably more cost effective (in some circumstances).

   xFS is probably more scalable than NASD because there is no one centralized component for any file system service. All of data service, meta-data service (name lookup, etc.), and caching (cooperative caching) is distributed. However, in NASD, there is a centralized file manager that provides capabilities on file lookup that is then later used by clients to do bulk data movement with NASD devices. The file manager in NASD may become a bottleneck.

   NASD is probably more cost-effective than NASD because no additional system components are needed above and beyond those that are part of traditional file servers. The usual file server is one computer controlling many disks. In NASD that file server becomes the file manager, and the disks become NASD devices. This holds if you believe the NASD paper's argument that NASD devices will not cost significantly more than traditional disks.

Name: ______________________________

4. A server can use RPCs to contact clients and inform them that their callbacks are broken. Explain why doing this within a single server thread can be inefficient. What might you change to fix this problem?

   Using a single server thread to make RPCs can be inefficient because there is no parallelism. Most of the time the server thread will probably be blocked waiting on network latency and the remote clients, which may be slow.

   A solution would be to make the server multithreaded. By doing so, the system components will be less idle, which is the same as saying that they are being more efficiently used.

Name: ________________________________

5. River and xFS both try to balance load across collections of servers. Explain how their foci differ and the resulting effects on their designs.

   xFS is a general-purpose file system that needs to have a well-known interface and very specific semantics, such as a hierarchical namespace, metadata consistency, etc. xFS was designed to distribute its functionality across nodes in a way that the authors thought would best suit a scalable file system.

   Instead of implementing one service, River is a programming model and run-time system for a class of applications. Specifically, they target data-parallel applications where producer and consumer code is replicated across many machines. They notice that a big hurdle to achieving high-performance in such systems are imbalances in node performance. They overcome this with specific run-time mechanisms: distributed queues and graduated declustering.

Name: ____________________________________

6. In the Emerald system, programmers must decide what node each object executes on. However, Emerald does have a few ways to automate some migration decisions; explain two such migration mechanisms.

   The two fundamental mechanisms that Emerald uses are compile-time specializations, and run-time adaptations.

   When making a remote procedure call, Emerald uses a combination of both to determine whether object arguments should migrate to the node with the remote procedure. To make the decision, Emerald needs to know: if there are current invocations on the object; how many invocations will be made on the object by the remote procedure, the object size, and the relative costs of mobility and local and remote invocation.

   At compile-time, Emerald knows the object size and may have information on whether there will be many invocations on the object by the remote procedure.

   At run-time, Emerald knows if there are current invocations on the object, and Emerald may have more information concerning how many invocations will be made on the object by the remote procedure. Further, only at run-time does Emerald know the relative costs of mobility and local and remote invocation.

Name: ______________________________

7. The Abacus function placement approach works well for data-intensive applications structured as a stack of objects. Emerald supports a more general communication architecture, where communication patterns could be any arbitrary graph. What problems would Abacus face if it tried to support less regular, graph-like applications?

   The more general the communication patterns are, the harder it is for the Abacus run-time to model the system and divine appropriate object locations. A stack of objects on two machines (client and server) is relatively simple. In this scenario, there are few potential ways to partition the objects. Further, the run-time can assume that all the objects except the two at the network boundary have zero communication costs. However, if you introduce more nodes and more complex communication patterns between the objects, then the problem reduces to graph-cutting, which is hard.

Name: ______________________________

8. The UCLA stackable FS infrastructure allows untrusted file system layers to be run in separate processes. Describe two approaches that avoid context switch overheads. Mention the consequence of each of your proposed approaches.

   In class we've discussed interpretation and type-safe languages.

   Imagine a running the file system layers inside of an interpreter. The interpreter may or may not be a general purpose language (constrast a simple packet filter with Java). Safety is provided by design, because the interpreter doesn't have constructs that would allow one layer to attack another. Drawbacks: interpretation can be slow; and if the interpreter isn't a general purpose language, users may find it constricting to work in.

   Type-safe languages (like Modula-3 used in SPIN) provide safety because one layer cannot obtain a reference (analogous to address pointer) to objects in another layer. Although a point of debate, some believe that because these are high-level languages, they impose a big performance cost.

   Two other approaches are: (1) sandbox the code using software fault-isolation; and (2) supply the code with an unforgeable proof of its correctness as in the proof-carrying code work.