A. Arpaci-Dusseau
CS739: Distributed Systems

Department of Computer Science
University of Wisconsin, Madison

Distributed State in NFS from v2 to v3 to v4

1 The Role of Distributed State

- **Introduction:** What is *state* in a computer system? What is *distributed state*? What are the three *benefits* Ousterhout gives of distributed state?

- What are the four reasons given for why distributed state can be bad?

- In NFSv2, servers are stateless; what does it mean to be *stateless*? Is it okay to keep anything in memory? What must be included in requests to the server given that it is stateless?

- Most NFSv2 operations are idempotent; what does it mean for an operation to be *idempotent*? Are all NFSv2 operatons idempotent?

- What is the main advantage of having stateless NFS servers? What does a client need to do when a server crashes? What does a server need to do when a client crashes?

- Why does the stateless NFSv2 server cause *performance* problems for client write requests? How do some NFS servers fix this problem?

- To obtain respectable performance, NFS clients may cache data. Why does the stateless server cause problems with data *consistency*? How does a client find out whether data it is caching is stale? Why does this approach cause performance problems too? When does a client write back modified data to the server?

- Can a stateless NFS server provide locks? Why or why not? What problems do non-idempotent operations, such as mkdir, cause?

2 NFS Version 3: Design and Implementation

- What were the three cited problems with NFSv2? We'll focus on problems having to do with distributed state...

- Why was keeping the server stateless still a goal? Combining stateless servers and non-idempotent requests is difficult; how did previous NFSv2 implementations deal with non-idempotent operations? With this technique, what happens if the reply to a non-idempotent operation is lost? What happens if the server crashes before it sends the reply? NFSv3 will continue to encourage this same implementation technique.

- How does NFSv3 improve performance of writes to the server? How does this optimization complicate the client if the server crashes? How does a client now know that the server has crashed?

  (Detail: Is it possible to have worse consistency semantics with this optimization?)

- NFSv2 suffered from the problem of too many calls to `GETATTR` being sent to the server. How does NFSv3 improve performance by reducing the number of calls to get attributes?

- NFSv3 attempts to improve the consistency model somewhat with *weak cache consistency*. In NFSv2, under what circumstances are `GETATTR` requests useless? How is NFSv3 changed to help this case? Given this protocol change, are clients now guaranteed to see the most recent writes made by other clients?

3 The NFS Version 4 Protocol

- What is the significance of the quote "Old Marley was as dead as a door-nail"?

- Why does NFSv4 introduce the `COMPOUND` procedure? What are its semantics? Does it introduce any complexities?

- Why does NFSv4 introduce `OPEN` and `CLOSE` operations? What does an exchange between Client A wishing to open file X for reading and writing look like with the server? What operations can Client A now keep local? What happens when Client B wishes to open the file for reading? How much state does the server now track?

- Adding state to the server complicates crash recovery. What happens now if client A crashes while it has the delegation for the open? How can the server give the delegation to another client? What are some of the problems with this solution?

- What happens now if the server crashes; that is, how can the server avoid simultaneously giving client B the delegation?

- When two clients have file open for writing, what consistency is guaranteed?

- Why are synchronization operations like `lock`/`unlock` needed for NFS? How does the lock protocol work?

  Leases are used for locks as well; what is different about leases for locks compared to delagations? What happens when client A holding a lock reboots?

  What if the server crashes while client A is holding a lock? What happens when client A tries to refresh its lease?