Compare and contrast NFSv2 and AFS.   Which features/properties of NFS seem superior and which features/properties of AFS?

Answer:

The most important difference is that AFS has stateful servers, whereas NFS has stateless servers. Another difference between the two file systems is that AFS provides location independence(the physical storage location of the file can be changed, without having to change the path of the file, etc.) as well as location transparency(the file name does not hint at its physical storage location), but NFS has only location transparency. In NFS, clients use delayed write, but in AFS, clients use write-on-close. In a word, NFS’s goal is fast crash recovery and simple, but AFS’s goal is scale.

NFS: fast crash recovery; stateless server

AFS: read and write faster; better in scale; performance better with frequent read and writes operation

<https://docstore.mik.ua/orelly/networking_2ndEd/nfs/ch07_02.htm>

* Most NFS requests are *idempotent*, which means that an NFS client may send the same request one or more times without any harmful side effects. The net result of these duplicate requests is the same. For example, reading a specific block from a file is idempotent: the same data is returned from each operation.

Obviously, some operations are not idempotent: removing a file can't be repeated without side effects, because a second attempt to remove the file will fail if the first one succeeded. Most NFS servers make all requests idempotent by recording recently performed operations. A duplicate request that matches one of the recently performed requests is thrown away by the NFS server.[[11]](https://docstore.mik.ua/orelly/networking_2ndEd/nfs/ch07_02.htm" \l "FOOTNOTE-11)

[11]Not all implementations of NFS have this duplicate request cache. Current releases of Solaris, Compaq's Tru64 Unix, and other current operating systems implement the cache to improve the performance and "correctness" of NFS. A few, older implementations of NFS do not reject nonidempotent, duplicate requests. This produces some strange and often incorrect results when requests are retransmitted. An NFS client that sends the same *remove* operation to such a server may find that the designated file was removed, but the RPC call returns the "No such file or directory" error.

The primary motivation for choosing a stateless protocol was to minimize the burden of crash recovery. Unlike a database system, which must verify transaction logs and look for incomplete operations, NFS has no explicit crash recovery mechanism. Because no state is maintained, the server may reboot and begin accepting client NFS requests again as if nothing had happened. Similarly, when clients reboot, the server does not need to know anything about them. Each NFS request contains enough information to be completed without any reference to state on the client or server.