NFS & AFS

NFS

→ remote file implementation unde UNIX that is easily portable to other OS and machine architectures.
→ Goal : make sharing of filesystem resources in a network of heterogeneous machines easier.
→ Main Design Goals : Machine and OS independence, Crash Recovers, Transparent Access, Unix semantics maintained on clients, "reasonable performance"
→ 3 pieces : protocol, server-side, client-side
→ NFS Protocol:
    Uses RPC
        Why? Because RPC helps simplify definition,organization and
        Implementation of remote services. RPCs are synchronous, and hence
        Easy to use because behave like local procedure call.
    Stateless ⇒ avoids complex crash recovery
    Transport independent : new transport protocols can be plugged in to the RPC implementation without affecting the higher level protocol code(uses UDP and IP)
    NFC protocol and RPC built on top of XDR ⇒ makes protocols machine and
    Language independent and easy to define
→ Server-Side:
    When servicing a NFS request it must commit(flush to disk) any modified data
    to a stable storage before returning results.
    Addition of a generation number (why? Because the server may hand out an
    fhandle with an inode of a file that is later removed and the inode re-used,
    when the new file comes back with the same inode , server has to know it is a different file) in the
    inode and a filesystem id in superblock. Servers to use the inode number, inode generation number and
        filesystem id together as fhandle for file

AFS

→ Goal : build a performant and scalable system that would be easy for a small operational staff to run and monitor with minimal inconvenience to users.
→ Design Challenges: large scale degrades performance, complicates administration as well as day to day operation
→ Caching: Two caches, one for status and the other for data.

- Status cache: kept in virtual memory and contains information such as the size of a file and its modification timestamp
- Data cache: kept on local disk and caches the actual data within a file.

→ State: AFS synchronizes state of local cached files and the server (Vice process) on file open/close.
→ Performance boosts:

- When caching in AFS, an assumption is made that your local cache will always be up to date as the server promises to send updates as the file is changed on the server. This lowers the number of cache validation requests received by the server.
- On the server, names are identified by a unique fixed-length id which all map to directories and pathnames. Locally, the Venus process maps from these fixed ids to pathnames which is logically equivalent to a *namei* operation which each kernel would need to perform. As Venus did this name resolution efficiently, this improved performance.

→ Trust model: Clients are not trusted in AFS and validation happens on the server by handing out expiring tickets to clients.

Some points that can go in the group report.

In AFS, the concept of a volume increases the speed of data management. With a volume, files can be moved to any other place inside that volume without physically moving the data. AFS's file synchronization scheme is another big performance feature. Read and write operation are performed on the cached copy of files, and are only synchronized on the server side when the file is closed. This greatly reduces the number of writes made to a file, thereby freeing up valuable processing time on servers, and client machines alike. Carrying on with caching in AFS, it was assumed that the cached copies could not be trusted, and therefore every cache copy was verified by comparing the timestamp of the cached copy with that on the home server of the file.

In NFS, its stateless model reduces the need for certain messages to be sent. Particularly error handling can be cut down on. This is because a client can just resend its requests until it received a response. In this manner, a server will not need to remember what the request was across server failures. Reducing the number of messages sent between servers is very important because this becomes a particular problem at scale.