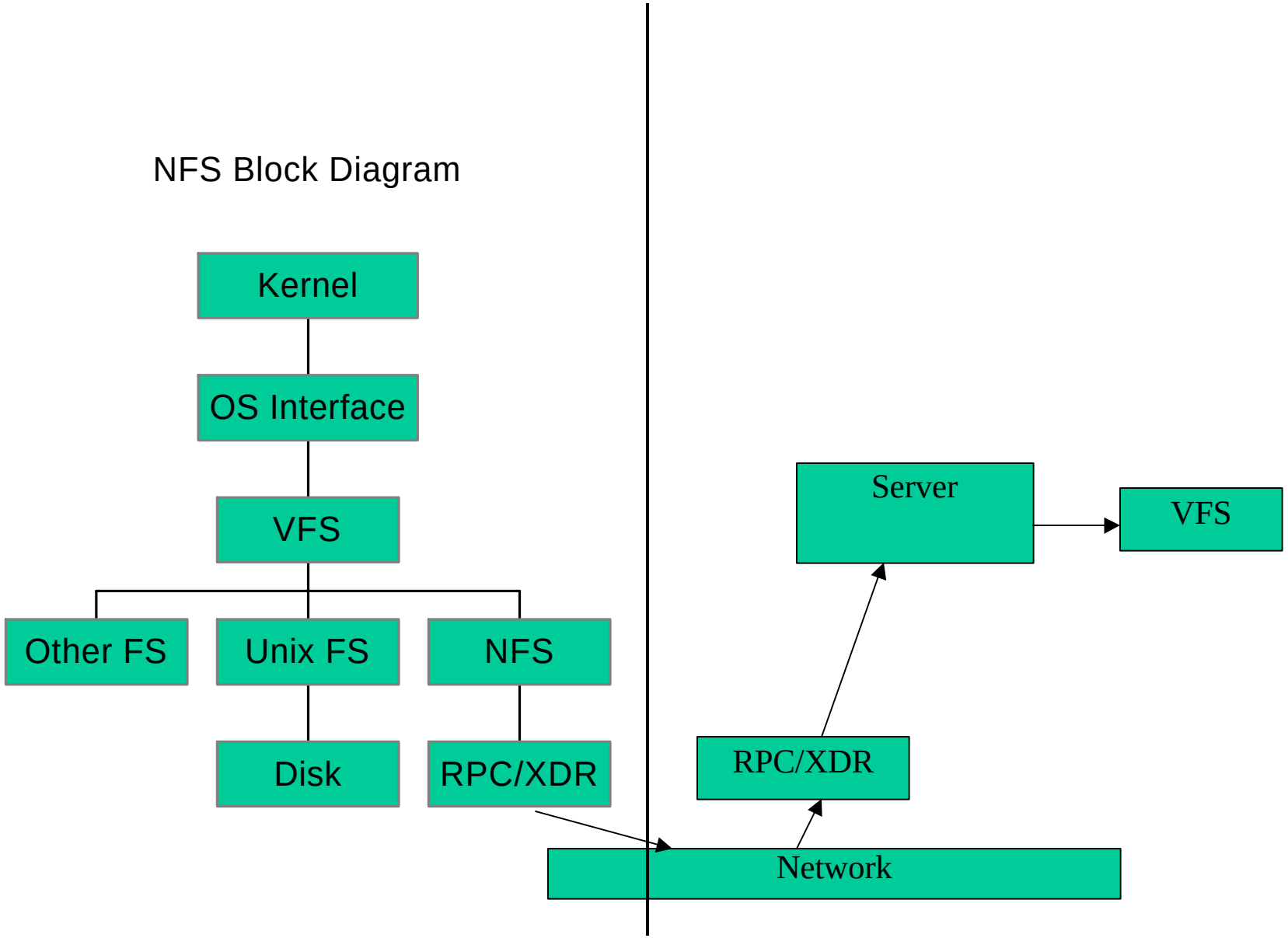
- High Availability
  - Generally via Data Replication, although the relation is not monotonically increasing.
    - Need to resolve consistency issues – strict vs. weak consistency.
  - Unit of Replication
    - Basic unit is a file. Just the needed data is replicated. However, management is harder.
    - Volume is typically a group of files (a particular user's files, all files on a disk etc.) Volume level replication can take extra storage, but makes management easier.
    - Compromise approaces, such as the one in Locus.
  - Replica Management

- Scalabity
  - How does the system scale as the number of clients and servers increase ?
    - Caching
    - Blaze and Alonso's scheme
- Semantics
  - Does the system guarantee basic unix semantics ?

Case Study -- NFS



- VFS layer is exposed to OS. Each object has a *vnode*. Each site also maintains a mount table.
- Vnode is inode like for local files. For remote files, it contains appropriate pointers. VFS will either pass the request to the local FS, or RPC it to the appropriate server. The server then translates it to an appropriate request for its local FS
- All clients are peers, and namespaces are independent across clients. In practice, some machines are dedicated as servers and namespace is made common.
- To map name into objects, vnodes are recursively followed. The server which actually has the file maps it, and returns a handle to the client.

- Each client maintains 3 caches in memory
  - File blocks, map from name to vnode, and attributes of open files/directories.
  - Blocks are cached on demand – Read ahead + Large blocks.
    - Timestamps cached with block.
    - Valid for certain duration, after which timestamp checked against server.
    - If timestamp has changed, all cached blocks for that file are invalidated, and refetched on demand.
    - Validation at file open and cache miss.
  - Delayed writing is used, combined with write on close.
  - Cached map from name to vnode helps for fast remote object lookup. Cache updated when lookup fails or when new information obtained.
  - Attributes are cached since attribute lookup constitutes a high percentage of calls. They have a short ttl in the cache (in seconds).

- NFS server is stateless. This makes crash recovery quite easy. However, this means that each request must carry "state information" needed to complete the request. This also means that a client cannot distinguish between a slow server and a crashed server in terms of resending requests ➔ requests should be idempotent!
- Cache management on server side is difficult, as is resolving conflicting accesses.