Sprite FS

- The entire namespace is a tree hierarchy, but is spread over disjoint domains. A server has one or more domains. Each client keeps a prefix table, which maps the topmost directory in a domain(prefix) to the domain and the server it is on.
  - Prefx tables are created dynamically using broadcast, and entries treated as hints.
- To lookup a name, the client finds the longest matching prefix in the table, and sends the remaining path along with the domain token to the corresponding server. This server returns a file token(handle) which is used subsequently.
  - CWD's prefix entry is a part of the state, to make relative filename access faster.
  - What if name sent to remote server again crosses boundary, e.g. on .. or symlinks.

- Caching is done on both client and server side.
- Client Cache stores recently accessed blocks indexed by file token
  - Don’t need to map to disk block, less communication with server.
  - Cache block size is 4K
  - Directories are not cached to avoid inconsistency
- Server Cache also stores file blocks.
- When client requests data, client cache is first checked. Otherwise, request passed to local FS or remote server. Remote Server checks server cache or passes request onto its local FS. No prefetching.
- Writes are delayed. Data unchaged for 30 secs or on cache miss (LRU replacement) is written
  - 20-30% data deleted within 30 secs, 75% files open for < 0.5 sec, 90% for < 10 seconds.
  - No write on close.
  - Cache misses 40% read req, 1% write req. Mostly for large files.

- Cache Consistency is server initiated.
- Concurrent write sharing is avoided
  - When a concurrent write request is received, server instructs current writer to flush data, commits it, and then declares the file uncachable to *all* clients.
  - All requests now flow through the server, which will serialize them.
  - About 1% traffic overhead
- Sequential Write Sharing is overcome by using versions, version number is incremented upon write . Server also keeps track of last client doing write, and requests that client to flush data when it gets the next file open from some other client.

- Virtual Memory and FS cache compete for memory!
  - Cache size is changed in response to memory usage.
  - Cache and VM keep separate pool of free pages, and track time of last access for each block. This is used in negotiation between the systems. Conversion from VM to cache is made difficult by “holding down”.
    - Double caching of executable programs!
- Backing store for VM is the regular filesystem.
  - Process migration is helped
  - Double Caching – avoid by not using client cache for such files.