Distributed System: File System

Alex Chi

Update: April 23, 2020

Contents

1 Background

- Network - Node(s) - Disk(s)

2 DFS Structure

- service: software providing fs
- server: where service runs
- client: invoke service
- client interface: primitive file operations
- client interface of DFS should be transparent (as if local files)

2.1 Unix file system operations

- refer to book and slides
- generally these are syscalls

2.2 Naming and Transparency

- naming: mapping between logical filename and physical block (inode)
- multi-level mapping: hides detail of how and where is stores
  - first level: file name
  - second level: inode
  - more levels if there are replicas
- transparent DFS hides where file actually is stored
- location transparency: file name not reveal physical storage
- location independence: file name is not bounded to physical location
- location independence is stronger than transparency
  - mount remote file system folder on NFS client

2.3 Remote File Access

- remote service mechanism (use RPC)
- reduce network traffic with cache
- cache consistency issue

2.3.1 Caching

- cache location: disk / memory
- update policy: write-through / delayed-write (aka. write back)

    - write through: one entry changed, flush all to disk
    - write back: modification written to cache later
        - flush on regular interval
        - write-on-close
    - very similar to main memory cache design in CPU
- why server and workstation use different cache policy?
    - server need to be stable (write-through)
    - client should be fast (write-back)
- consistency
    - client-initiated approach
        - client check validity
        - server check local data consistent with master copy
        - trade-off between access performance and reading latest data
    - server-initiated approach
        - servers record what files a client cache
        - server notifies client on inconsistency

2.3.2 Cache / Remote Service

- many pros and cons. refer to slides.

2.4 Stateful vs Stateless

- stateful server remembers client information
    - Unix file API is stateful
- stateless means that each request is new to server
    - HTTP is a stateless protocol
- difference: refer to slides

2.5 Replication

- fault tolerance
- refer to slides

3 Andrew File System

- clients are presented local name space and shared name space
- dedicated servers: Vice
- protocol: Virtue
- client: Venus
- cluster: in LAN, a cluster server, several workstations, a router
- whole file caching