Elosztott rendszerek: Alapelvek és paradigmák
Distributed Systems: Principles and Paradigms

Maarten van Steen[1] Kitlei Róbert [2]

[1]VU Amsterdam, Dept. Computer Science
[2]ELTE Informatikai Kar

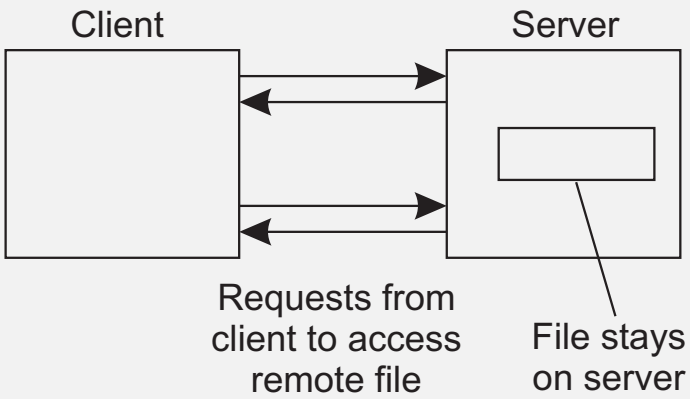
11. rész: Elosztott fájlrendszerek

2015. május 24.

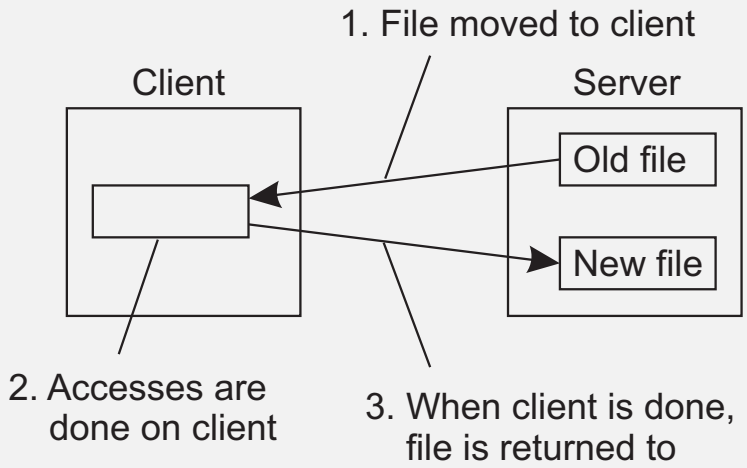
Tartalomjegyzék

Elosztott fájlrendszerek

Cél: a fájlrendszer átlátszó elérését biztosítani távoli kliensek számára.



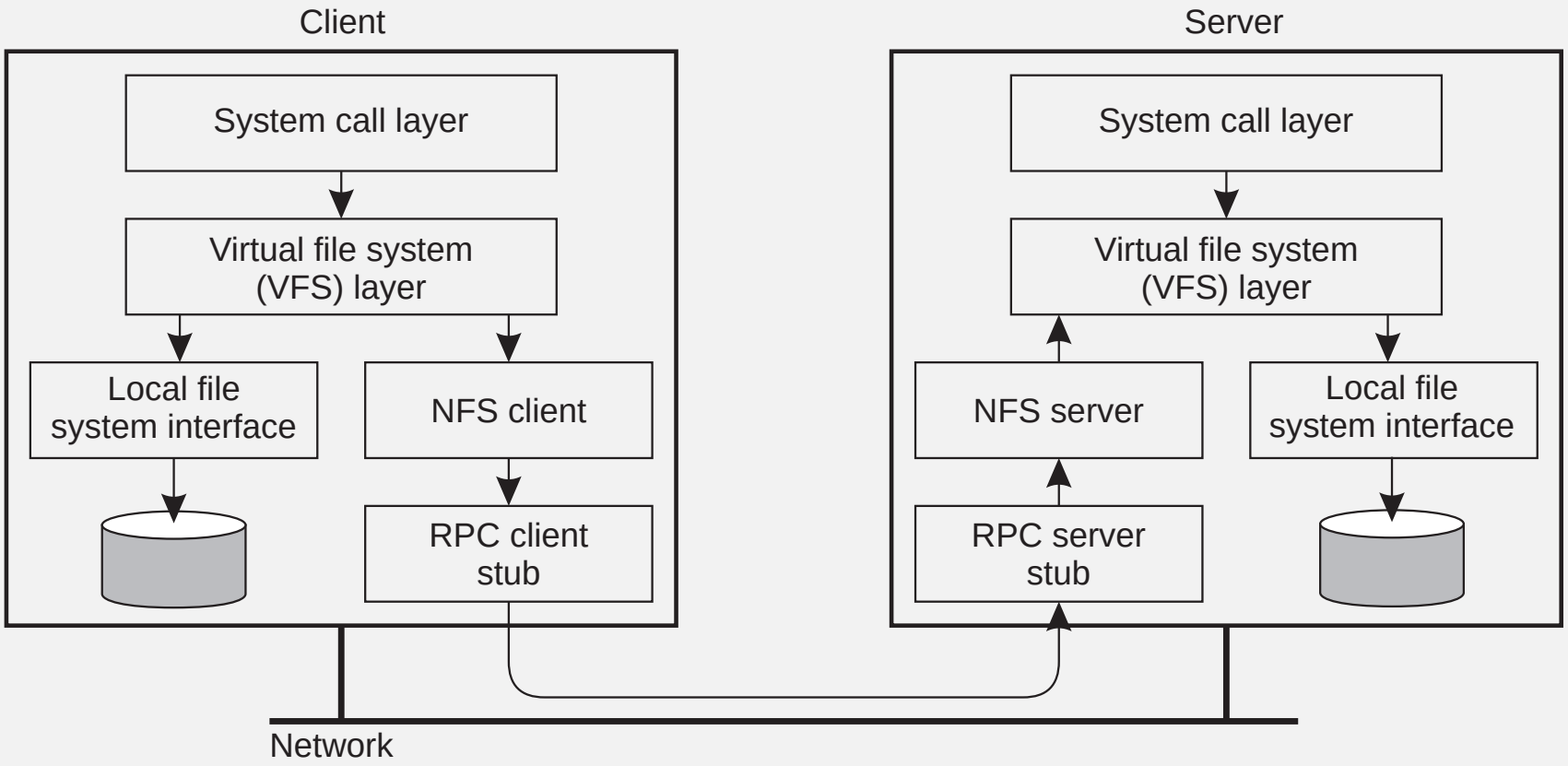
Távoli hozzáférési modell



Feltöltés/letöltés modell

Példa: NFS architektúra

Az NFS (Network File System) elosztott fájlok távoli elérését teszi lehetővé. Az alkalmazások a helyi VFS (Virtual File System) réteget érik el, ez biztosítja a távoli elérés átlátszóságát.

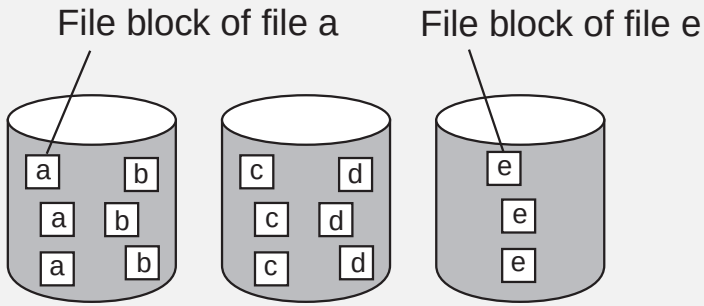


NFS fájlműveletek

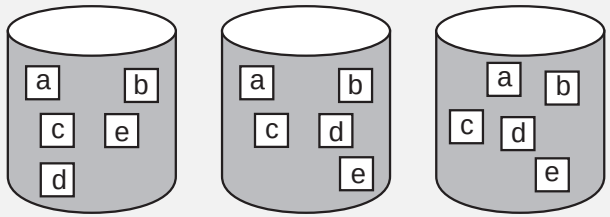
Művelet	Verzió	Leírás
Create	v3	Hagyományos fájl létrehozása
Create	v4	Nem-hagyományos fájl létrehozása (hagyományos: Open)
Link	v3 és v4	Valódi csatolás (hard link) létrehozása fájlra
Symlink	v3	Szimbolikus csatolás (soft link, symlink) létrehozása fájlra
Mkdir	v3	Alkönyvtár létrehozása
Mknod	v3	Különleges fájl létrehozása
Rename	v3 és v4	Fájl átnevezése
Remove	v3 és v4	Fájl eltávolítása a fájlrendszerből
Rmdir	v3	Üres könyvtár eltávolítása
Open	v4	Fájl megnyitása
Close	v4	Fájl bezárása
Lookup	v3 és v4	Fájl megkeresése név szerint
Readdir	v3 és v4	Könyvtár tartalmának lekérése
Readlink	v3 és v4	Szimbolikus csatolás elérési útvonalának olvasása
Getattr	v3 és v4	Fájl tulajdonságainak lekérdezése
Setattr	v3 és v4	Egy vagy több tulajdonság írása
Read	v3 és v4	Fájl tartalmának olvasása
Write	v3 és v4	Fájl adatainak írása

Fürt (cluster) alapú fájlrendszerek

Nagy fájlrendszerek esetén a kliens-szerver alapú megközelítés nem elég jó $\Rightarrow$ a fájlokat csíkokra bontjuk (striping), így a részeiket párhuzamosan érjük el. Ennek célja a rendszer gyorsítása és biztonságosabbá tétele.

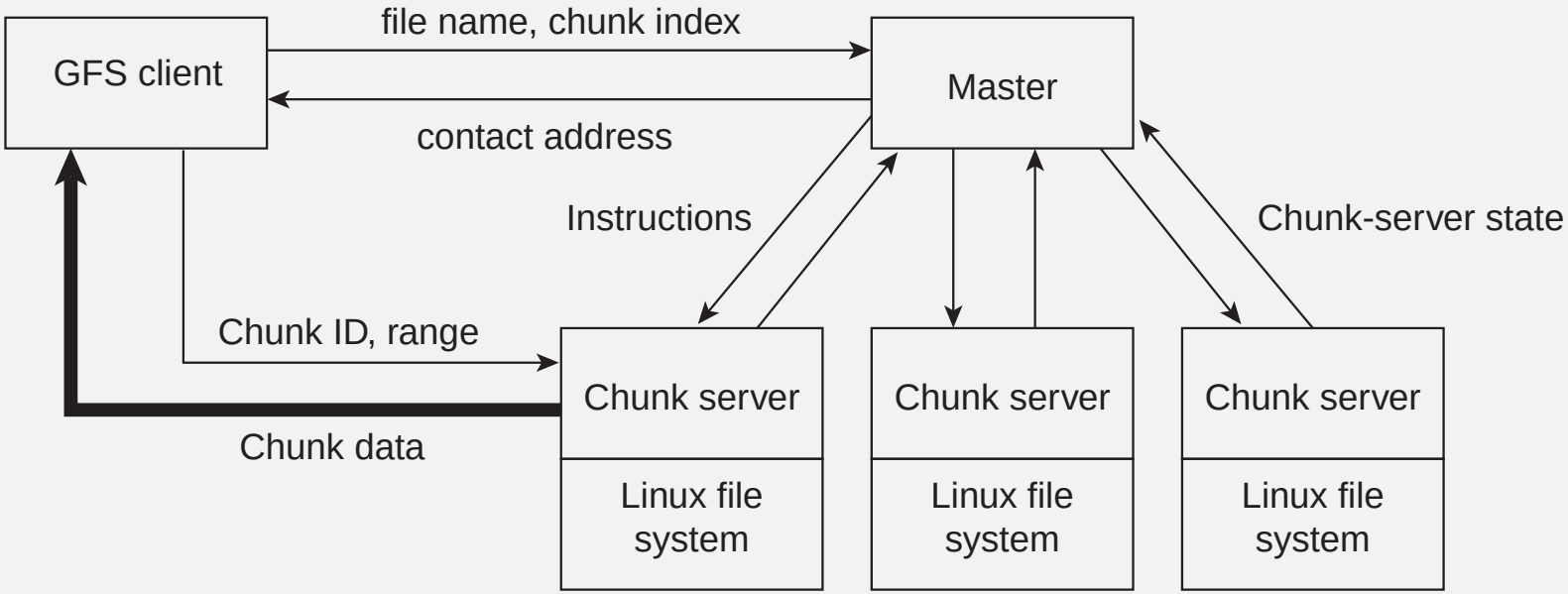


Whole-file distribution



File-striped system

Példa: Google File System

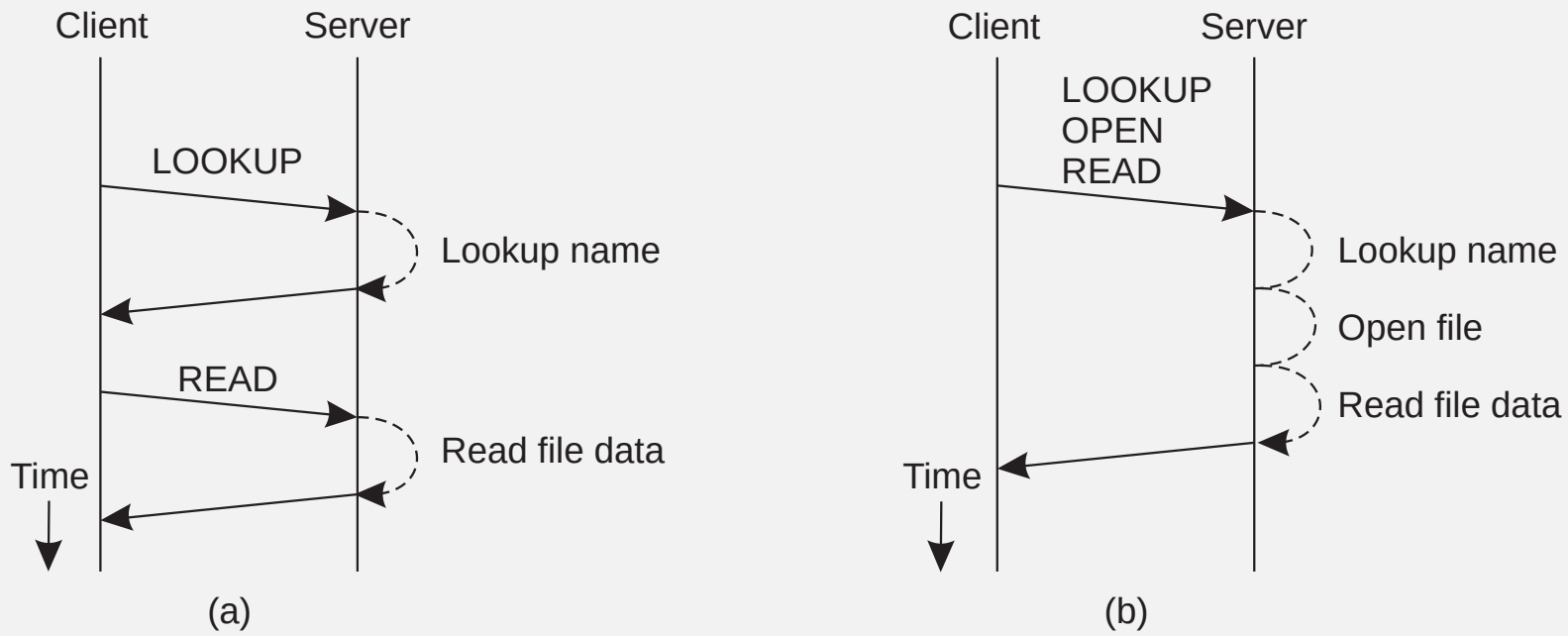


A fájlt 64 MB méretű részekre (chunk) bontjuk, és több szerveren elosztva, replikálva tároljuk.

- A központ (master) csak azt tárolja, melyik szerver melyik részek felelőse $\Rightarrow$ I/O terhelése alacsony
- A szerverek elsődleges másolaton alapuló replikációs protokollt használnak; a központot ez nem terheli

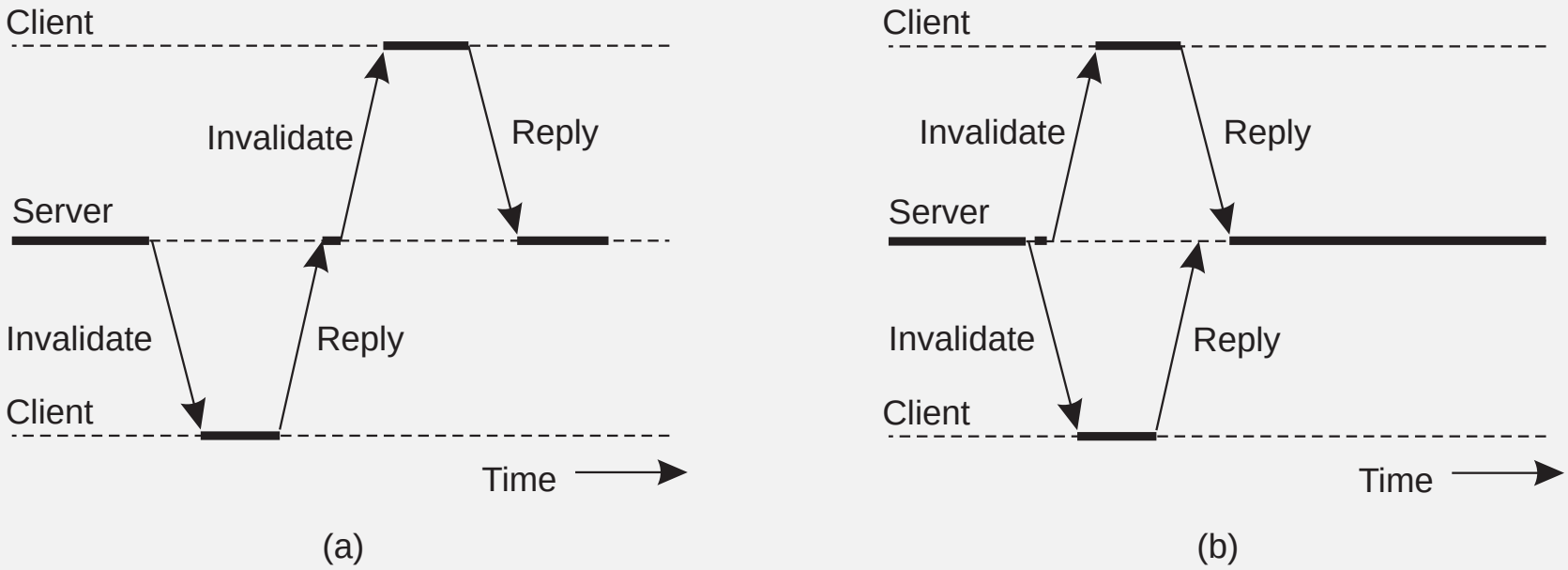
RPC fájlrendszerekben

Egy lehetőség távoli fájlrendszerek megvalósítására, ha távoli eljáráshívások segítségével végezzük a fájlműveleteket.
Ha a távoli gép elérése költséges, alternatív megoldásokra van szükség, pl. az NFSv4 támogatja több művelet összekombinálását egy hívásba.



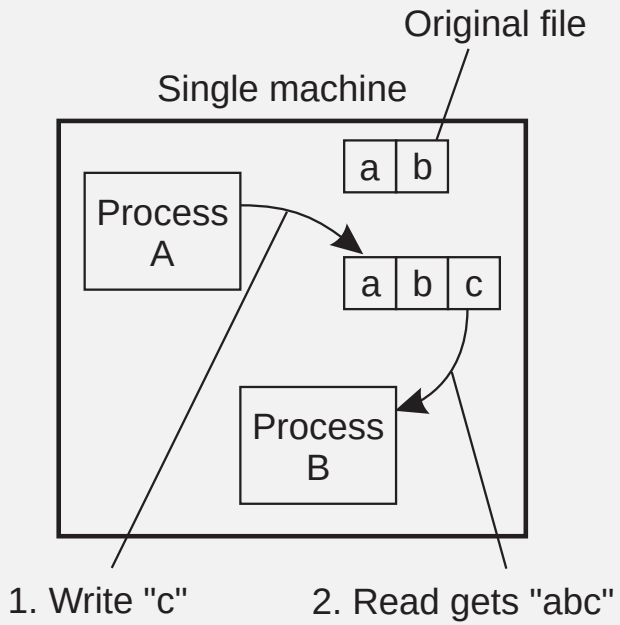
Példa: RPC a Coda fájlrendszerben

Ha replikált fájlokkal dolgozunk (pl. a Coda kliensei cache-elhetik a fájlokat), akkor a kérések sorrendjének kikényszerítése (`(a)` ábra) túlságosan költséges lehet, mert ha összeomlik egy kliens, akkor csak hosszú timeout után jöhet a következő.

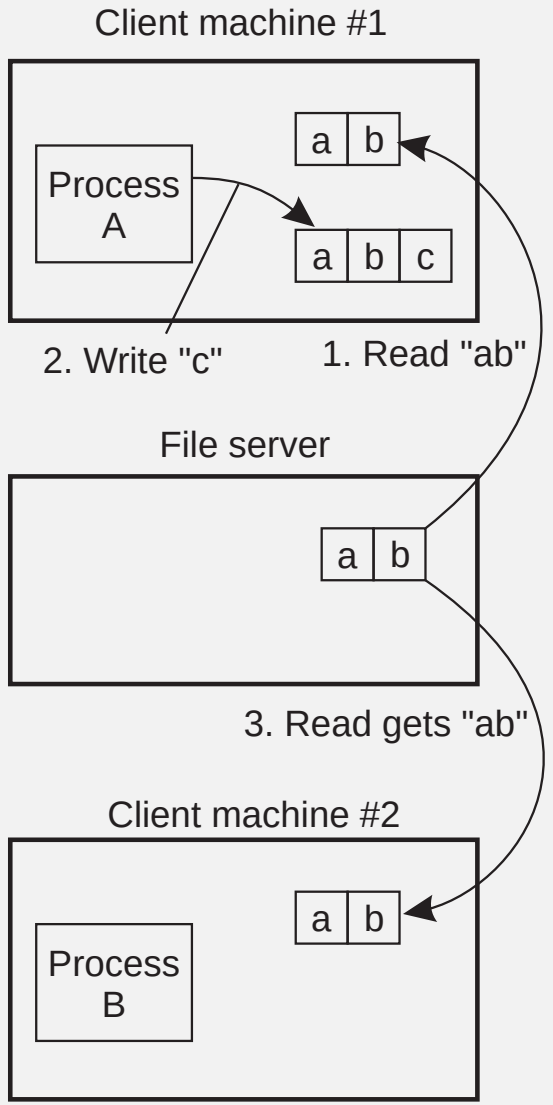


A fájlmegosztás szemantikája

Ha egyszerre több kliens is hozzáférhet egy fájlhoz, a konkurens írási és olvasási műveletek lehetséges végrehajtási sorrendjeit és a kijöhető eredményeket (összefoglalva: a rendszer szemantikáját) rögzíteni kell.



(a)



(b)

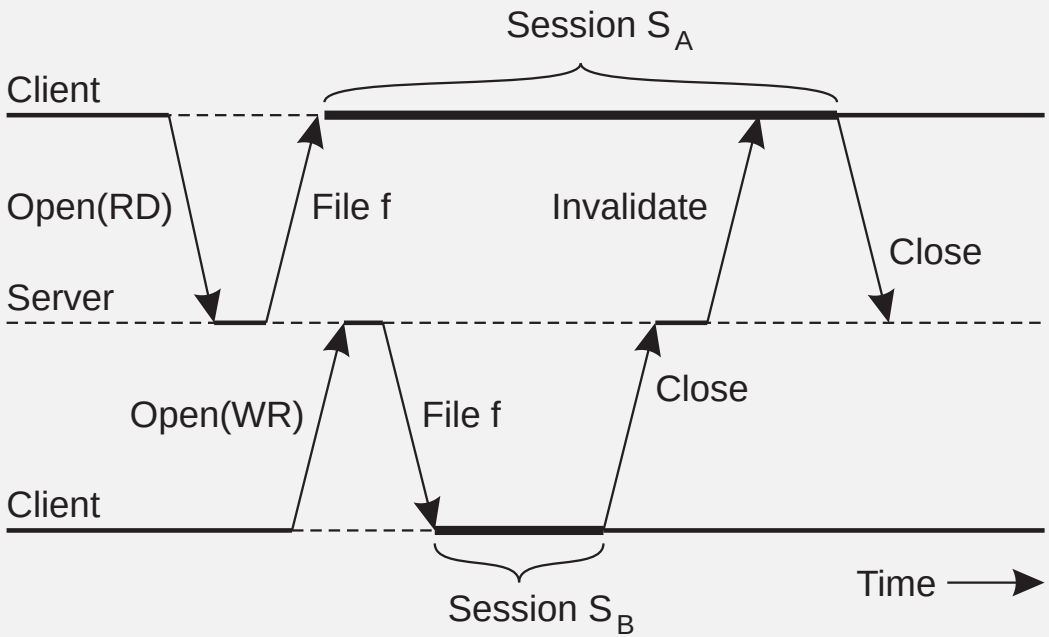
A fájlmegosztás szemantikája

Sokfajta szemantika jöhet szóba.

- Megváltoztathatatlan fájlok: a fájlok tartalmát nem lehet módosítani létrehozás után; ez a modell csak ritkán használható
- UNIX szemantika: az olvasási műveletek mindig a legutolsó írási művelet eredményét adják $\Rightarrow$ a fájlból csak egy példányunk lehet az elosztott rendszerben (az előző `(a)` ábra)
- Tranzakciós szemantika: a rendszer minden fájlra külön biztosít tranzakciókat
- Munkamenet szemantika: onnantól, hogy a kliens megnyitja a fájlt, odáig, amíg vissza nem írja, az írási és olvasási műveletei csak saját maga számára látszanak (gyakori választás a rendszerekben; az előző `(b)` ábra)

Példa: fájlmegosztás Coda rendszerben

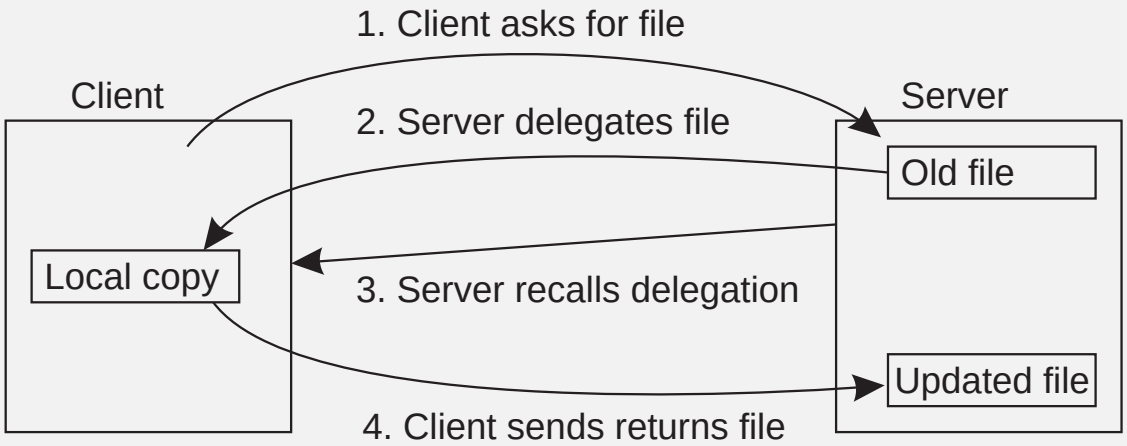
A Coda fájlrendszer munkamenetei tranzakciós szemantikát valósítanak meg. A megnyitott fájl tartalma átmásolódik a kliensre; ha más módosítja a fájlt, arról a kliens értesítést kap.
Ha a kliens csak olvas (pl. az ábrán S_A), akkor folytathatja a működését – úgy tekintjük, hogy a tranzakció, amelyet végez, már korábban lezárult.



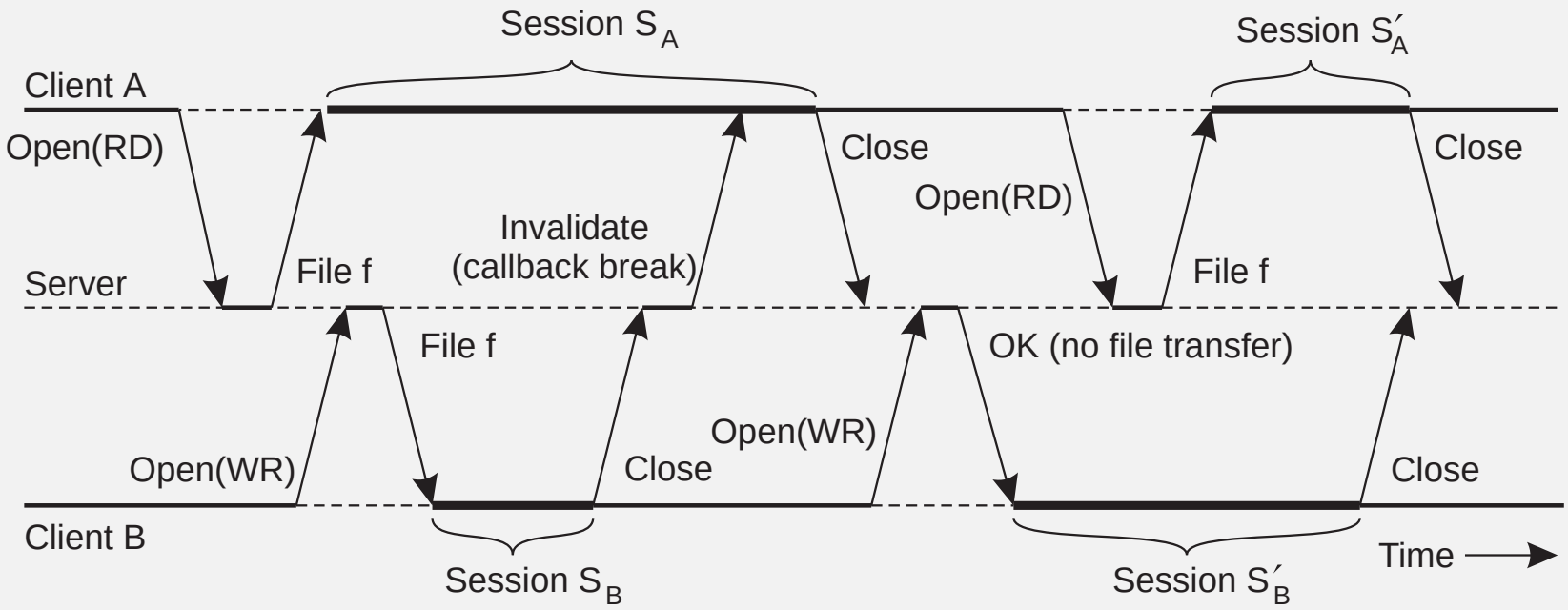
Konzisztencia és replikáció

A modern elosztott fájlrendszerekben a kliensoldali gyorsítótárazás szerepe főleg a teljesítmény növelése, a szerveroldali replikáció célja a hibatűrés biztosítása.

A kliensek tárolhatják a fájlokat (vagy részeket belőlük), és a szerver kiértesíti őket, ha ezt a jogot visszavonja tőlük $\Rightarrow$ a szerverek általában állapotteljesek.



Példa: kliensoldali gyorsítótárazás Coda rendszerben



A tranzakciós szemantika segítségével a teljesítmény tovább növelhető.

Rendelkezésre állás növelése P2P rendszerekben

Sok P2P alapú, decentralizált fájlrendszer létezik. Ezekben probléma lehet, ha túl gyorsan változik a tagság (churn), mert kiléphet akár egy fájlt tartalmazó összes csúcs. Ennek kivédéséhez replikálhatjuk a fájljainkat (arányát jelölje r_{rep}).
Másik megközelítés: erasure coding: az F fájlt bontsuk m részre, és minden szerverre tegyünk n részt, ahol $n > m$. A replikációs arány ekkor $r_{ec} = n/m$. Ez az arány általában sokkal kisebb, mint r_{rep}, ha a rendszerünk változékony.