BLOC IV. Servicii distribuite.

– Serviciu de fișiere distribuite. Studiu de caz. –

Sisteme și servicii distribuite

Inginerie Telematică de gradul III

Cuprins

Introducere

- Sistemul de fișiere distribuit (SAD) permite accesul la informațiile stocate într-un fișier de pe orice

  computer din rețea.
- Acces cu aceeași (sau mai bună) performanță și fiabilitate decât stocarea locală pe disc.
- Caracteristicile unui SAD: transparență, concurență, replicare fișier, eterogenitate, consistență etc.
- Vom studia: service și server de fișiere, protocoale de implementare, replicare și actualizare.
- Studiu de caz: NFS

Cuprins

Ce este un sistem de fișiere?

- Sistemul informatic trebuie să stocheze și să recupereze informații
  => mediu de stocare NON-VOLATIL
- Mulțime de medii de stocare: disc, bandă, CD,
  Flash
  - Diferite particularități: viteza de acces, fiabilitate etc.
- Sistemul de operare extrage proprietățile fizice ale suportului
  stocare prin definirea unei unități de stocare logică: fișierul
  - Magazin de informații persistente accesibil pe nume, care ascunde realitatea fizică a stocării și servește la organizarea informațiilor
  - Un fișier este o secvență de octeți plus unele atribute care descrieți (nume, tip de fișier, locație, dimensiune, permisiuni de acces, data creării/modificării, proprietar)

Ce este un sistem de fișiere?

- Sistemul de fișiere este partea sistemului de operare responsabilă de organizarea, stocarea, preluarea, denumirea, preluarea și protecția fișierelor.
- Aspectele legate de nume sunt organizate prin intermediul directoarelor.
- Director: tip special de fișiere al căror conținut este o mapare a numelor de fișiere la identificatorii de fișiere interni
  - Poate include numele altor directoare
  - Trebuie să fie, de asemenea, nevolatil

Ce este un sistem de fișiere?

- Operații principale care pot fi efectuate pe fișiere (UNIX)
  - filedes =open(name,mode) –
  - filedes=creat(name,mode) –
  - status=close(filedes) – count
  - =read(filedes,buffer,n) – count
  - =write(filedes,buffer,n) – pos=
  - lseek(filedes,offset,whence) –
  - status=unlink(nume) –
  - status=link(name1,name2) –
  - status=stat(nume,buffer)
- Aceste funcţii sunt implementate în nucleu. Este accesat prin proceduri din bibliotecile C.
- Sistemul de fișiere este responsabil pentru controlul accesului la fișiere (permisiuni)
- Diferite moduri de alocare a spațiului fișierelor: alocare contiguă, legată sau indexată
  - Utilizare eficientă
  - Viteza de acces

atribuire contiguă

- Fiecare fișier ocupă un set de blocuri învecinate pe disc (de ex. bloc=512 octeți)
- Pentru fiecare fișier trebuie să salvați adresa blocului inițial și lungimea (numărul de blocuri) zonei alocate fișierului.
- Avantaje
  - Simplu de implementat
  - Accesul secvenţial sau aleator la fişiere este rapid
- Dezavantaje:
  - Fișierul nu poate crește
  - Algoritm de căutare necesar pentru spațiul pe disc
  - Fragmentare externă: spațiu pe disc suficient, dar nu contiguu
    - Soluție: compactare => proces lent
    - Atribuire adiacentă schimbată = link către extensii
  - Fragmentare internă: dacă dimensiunea fișierului nu se potrivește cu un număr întreg de blocuri, o parte din spațiu este irosită.

Temă legată (lista legată)

• Fiecare fișier este o listă legată de blocuri de disc. • Directorul

conține pentru fiecare fișier primul și ultimul bloc • Fiecare bloc

conține, pe lângă date, un pointer către următorul
bloc •

Avantaje

- Simplu de implementat
- Fără fragmentare externă
- Nu este necesar să declarați dimensiunea fișierului și acesta poate crește

• Dezavantaje:

- Accesul secvenţial nu este la fel de eficient
- Spațiul necesar pentru indicatoare este irosit.
- Fiabilitate: dacă un pointer este deteriorat, restul fișierului este imposibil de urmărit

Alocare legată (FAT)

- Caz special de atribuire legată
- File Allocation Table: tabel de alocare a fișierelor. O secțiune este rezervată de pe disc până la începutul partiției pentru a salva tabelul.
- Toate indicatoarele către toate fișierele sunt salvate în FAT. FAT conține o intrare pe bloc de disc, indicând următorul bloc al unui fișier sau NULL.
- Când discul este accesat pentru prima dată, FAT-ul este citit și copiat în RAM
- De fiecare dată când se actualizează FAT RAM, discul este de asemenea actualizat
- Diverse tipuri: FAT12, FAT16, FAT32
- Folosește little endian
- Are 4 regiuni: 0: rezervat, 1: regiune FAT, 2: regiune director rădăcină (nu în FAT32) și 3: regiune de directoare și fișiere
- Avantaje
  - Eficient pe partiții mici
  - Accesul direct este eficient

alocare indexată

- Pointerii către fiecare fișier sunt păstrați împreună într-o structură (tabel) asociată cu acel fișier: i-node
- i-node salvează, de asemenea, informații despre atributul fișierului.
- Directorul va conține un pointer către blocul care are tabelul (i-node) al fiecărui fișier din director
- Dacă fișierul ocupă mai multe blocuri decât încadează pointerii în i-node, sunt folosite blocuri de index indirect (i-nodes).
- Folosit pe UNIX
- Avantaje
  - – Acces direct mai eficient
- Dezavantaje:
  - - Mai complex

Cuprins

Caracteristicile unui SAD

- Definiția SAD: sistem de fișiere în care fișiere clienții, serverele și dispozitivele de stocare sunt dispersate într-o rețea, dar distribuția este transparentă pentru client

- Caracteristicile de dorit ale unui SAD:
  - Transparență: a accesului și locației
    - Acces: programele client nu au nevoie să cunoască locația fizică a fișierelor, aceleași operațiuni sunt utilizate pentru accesul local și la distanță fără a fi nevoie de modificarea programelor.
    - Localizare: clienții văd un spațiu de nume uniform: fișierele pot fi mutate fără a le schimba calea
  - Controlul concurenței: modificările efectuate de un client nu trebuie să interfereze cu operațiunile simultane ale altor clienți
    - Tehnici scumpe. UNIX folosește blocarea fișierelor bine sfătuit, bun de obligație
  - Consistență: semantica actualizării unei copii => The conținutul unui fișier pe care îl văd clienții este ceea ce ar vedea dacă ar exista o singură copie a fișierului

Caracteristicile unui SAD

- Caracteristicile de dorit ale unui SAD:
  - Replicare: Copiile unui fișier sunt păstrate în locuri diferite.
    - Avantaje: echilibrare a sarcinii și toleranță la erori
    - Puține servicii SAD oferă această caracteristică
  - Eterogeneitate: interfețele de servicii trebuie definite astfel încât să poată fi implementate de diferite sisteme de operare și cu diferite platforme
  - Securitate: mecanisme de control al accesului, cerere de autentificare și protecție a conținutului mesajului
  - Eficiență: un SAD trebuie să ofere facilități cu aceleași generalitate și importanță decât cele utilizate în sistemele locale și atinge un nivel comparabil de performanță.
  - Toleranță la erori: este esențial ca SAD-ul să continue să funcționeze în cazul unor defecțiuni la client sau la server

Cuprins

Server de fișiere și serviciu

- Serviciu de fișiere: specificarea serviciilor (funcționalitatea) pe care SAD le oferă clienților săi
  - Primitive disponibile, parametrii de intrare și de ieșire și acțiunile care se realizează.
  - Nu definește modul de implementare
- File server: proces care rulează pe a mașină și ajută la implementarea serviciului de fișiere
  - Pot exista unul sau mai multe servere de fișiere.
  - Clienții nu trebuie să cunoască existența sau numărul serverelor de fișiere, poziția sau funcția acestora.

Cuprins

Design SAD

- Un SAD are în mod normal 2 componente:

  1. Servire de fișiere: operațiuni pe fișiere individuale 2. Servire de directoare: crearea și gestionarea directoarelor

- Identificator unic de fișier (UFID): secvență de biți, astfel încât fiecare nume de fișier să aibă un UFID unic printre toate fișierele din sistem
  - Sunt folosite pentru a efectua operațiuni cu fișiere
  - Când serviciul de fișiere primește o solicitare de creare dintr-un fișier generează un nou UFID
  - Serviciul de directoare mapează numele fișierelor și UFID-urile
  - Permite implementarea transparenței locației

Interfața serviciului de fișiere

- Interfața serviciului de fișiere este definită de următoarele operații:

  – Citiți (Id fișier, i, n) -> Date
  - Scriere (ID fișier, i, date)
  – Create()->FileId
  - Șterge (FileId)
  – GetAttributes (ID fișier) -> Attr
  - SetAttributes (ID fișier, Attr)

- Aceste operațiuni nu sunt utilizate direct la nivelul de

  Nume de utilizator

- Cu excepția Create(), toți lansează o excepție dacă File ID nu este un UFID valid sau permisiunile necesare nu sunt disponibile.
- Nu există operațiuni de deschidere și închidere, pe care le puteți accesa direct într-un fișier folosind UFID-ul său
  – Acestea permit implementarea SAD care nu trebuie să salveze starea. Mai eficient și tolerant la erori, în teorie. Dar face dificil controlul concurenței

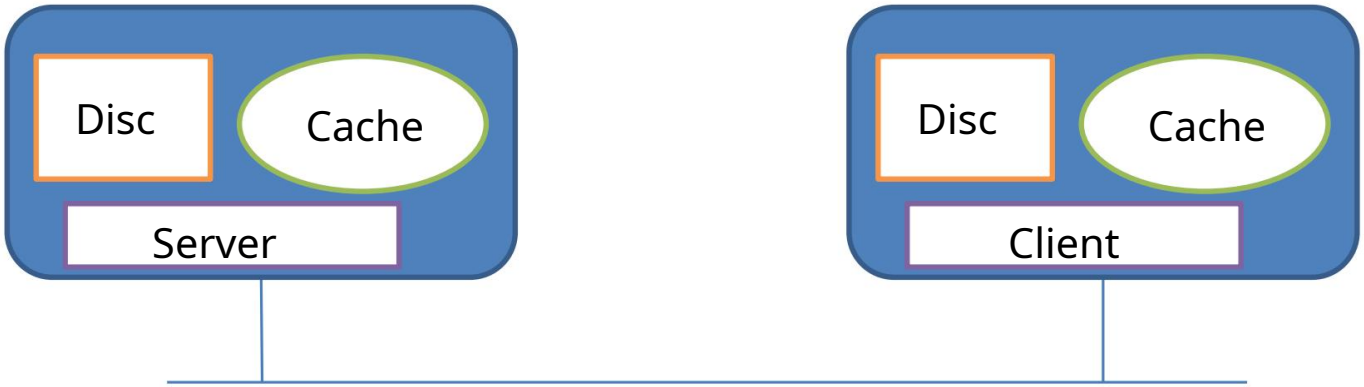
Interfața serviciului de directoare

- Interfața serviciului de director este definită de următoarele operații:
  - Căutare (Dir, nume) -> FileId
  - Adăugare nume (Dir, Nume, ID fișier)
  - UnName(Dir,Nume)
  - GetNames(Dir, Pattern) ->NameSeq
- Obiectivul său de bază este de a oferi un serviciu de traducere a numelor în UFID.
- Fiecare director este stocat cu un nume și un UFID, astfel încât serviciul de director este el însuși un client al serviciului de fișiere.
- Este posibil să aveți referințe la directoare în cadrul unui director, permițând sisteme de fișiere ierarhice

Ascundere

• Ascundere = stocare în cache, în acest context • Cel mai bun loc pentru stocarea fișierelor: disc server
  - • Accesibil pentru toți clienții
  - • Ineficient: necesită acces la disc, stocare principală și transmisie în rețea • Mai eficient pentru a stoca fișierele cele mai recente utilizate în memoria principală a serverului • Algoritmi pentru a determina ce părți să memoreze în cache • Ușor și transparent pentru client • Există încă acces la retea

• Ascundere în memoria clientului: introduce inconsecvență în sistem



Ascundere

	Avantaje	Dezavantaje
Disc server	• Mai mult spatiu • Accesibil de către toți clienții • O singură copie: nu există incoerență	• Performanta scazuta • Aglomerație (trafic intens)
Memoria serverului	• Accesibil de către toți clienții • O singură copie: nu există incoerență	• Aglomerație (trafic intens)
Memoria clientului	• Nu există aglomerație • Performanta ridicata	• Incoerență

Soluții la inconsecvența de ascundere

- Scrieți prin cache (scrieți prin cache)
  - Când scrieți în cache, este trimis imediat la server
  - Problemă: Nu remediază neconcordanța cu versiunile anterioare citite de alt client
  - Sun: verificați întotdeauna cu serverul dacă copia este actualizată => mai mult trafic
- Scriere întârziată
  - Trimite actualizări la fiecare X secunde.
- Act la închidere
  - Se scrie pe server doar când aplicația închide fișierul
- Control centralizat
  - Serverul ține evidența fișierelor deschise, a proprietarilor acestora și a modului (citește, scrie sau ambele)
  - Nu este robust și nu este foarte scalabil

Replica

- Replicarea fișierelor ca serviciu pentru clienți
  - Copierea unor fișiere pe servere separate/diferite • Avantaje:
  - Flexibilitate
  - toleranta la erori
  - Echilibrarea sarcinii de lucru • Replicare explicită
  - Controlat de programator • Replicare întârziată
  - O copie pe server
  - Serverul se replică automat după • Replicarea grupului
  - Toate apelurile sunt redirecționate automat către grup
  - Aveți nevoie de comunicare în grup (multicast)

Replicare: protocoale de actualizare (I)

Cum se actualizează fișierele care sunt replicate?

- Replicarea copiei primare (soluție centralizată):
  - Serverul primar controlează cele secundare
  - Problemă: defecțiune primară
- Algoritm de vot:
  - Clienții solicită mai multor servere permisiunea înainte de a citi sau scrie
  - Pentru a actualiza clientul trebuie să aibă acordul (consensul) unui număr de servere (de exemplu, majoritatea = jumătate plus unul dintre servere)
  - Pentru fiecare modificare, serverele care aparțin grupului de consens actualizează numărul versiunii (sau data ultimei modificări).
  - Pentru citire, serverele grupului de consens trimit numărul versiunii
  - Se asigură că clientul are cea mai recentă versiune

Aftershock: Actualizare protocoale (II)

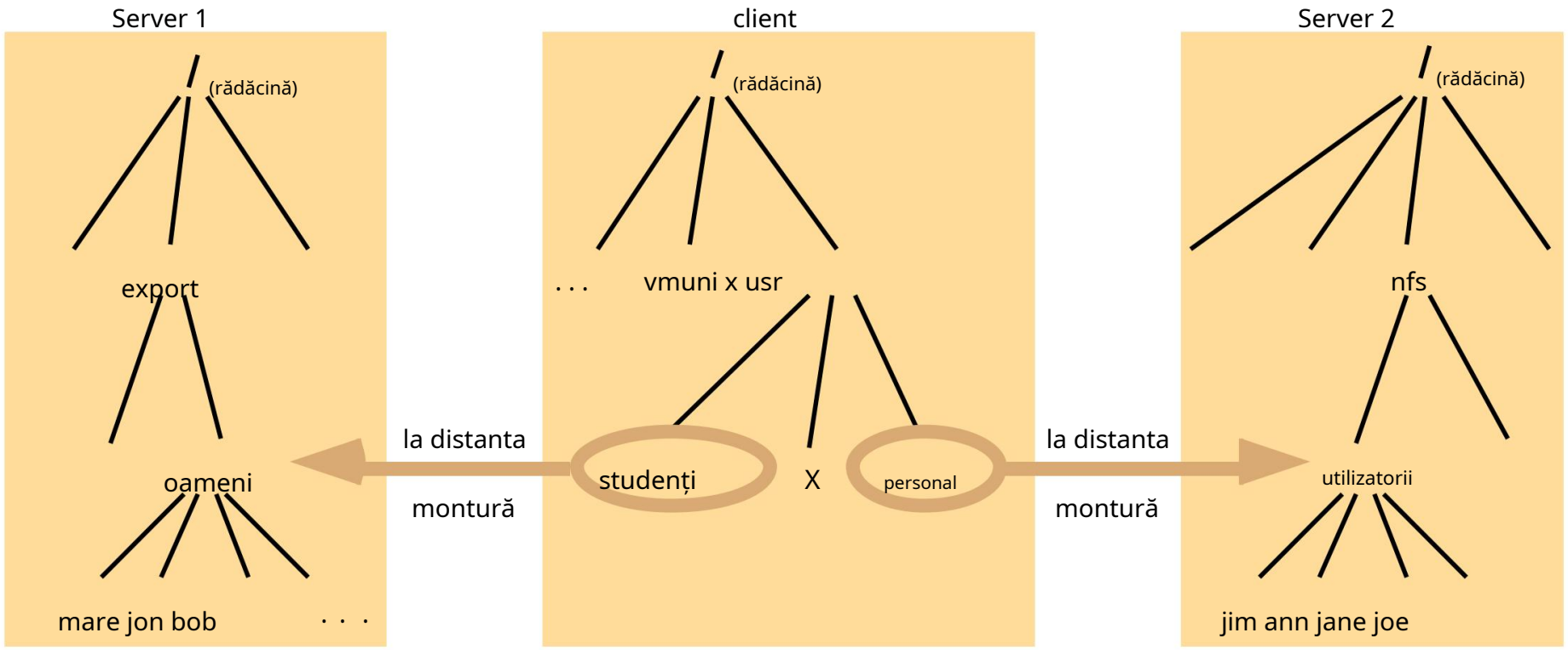
- Schema Gifford:
  - Există N replici ale unui fișier
  - Cvorum de citire: contactați serverele Nr pentru a putea citi
  - Cvorum de scriere: contactați serverele Nw pentru a putea a scrie
  - Nr+Nw > N
- Votarea cu fantomă:
  - Operațiile de citire sunt mai frecvente: Nr și Nw mici aproape de N, dar dacă vreun server eșuează, este ușor să pierdeți cvorumul de scriere
  - Server fantomă fără spațiu pentru fiecare server real care a eșuat și fantoma NU este permisă în cvorumul de citire (DA în cvorumul de scriere)
  - La pornirea serverului eșuat, acesta obține cvorum de citire și copiază cea mai recentă versiune înainte de a relua funcționarea normală

Cuprins

NFS

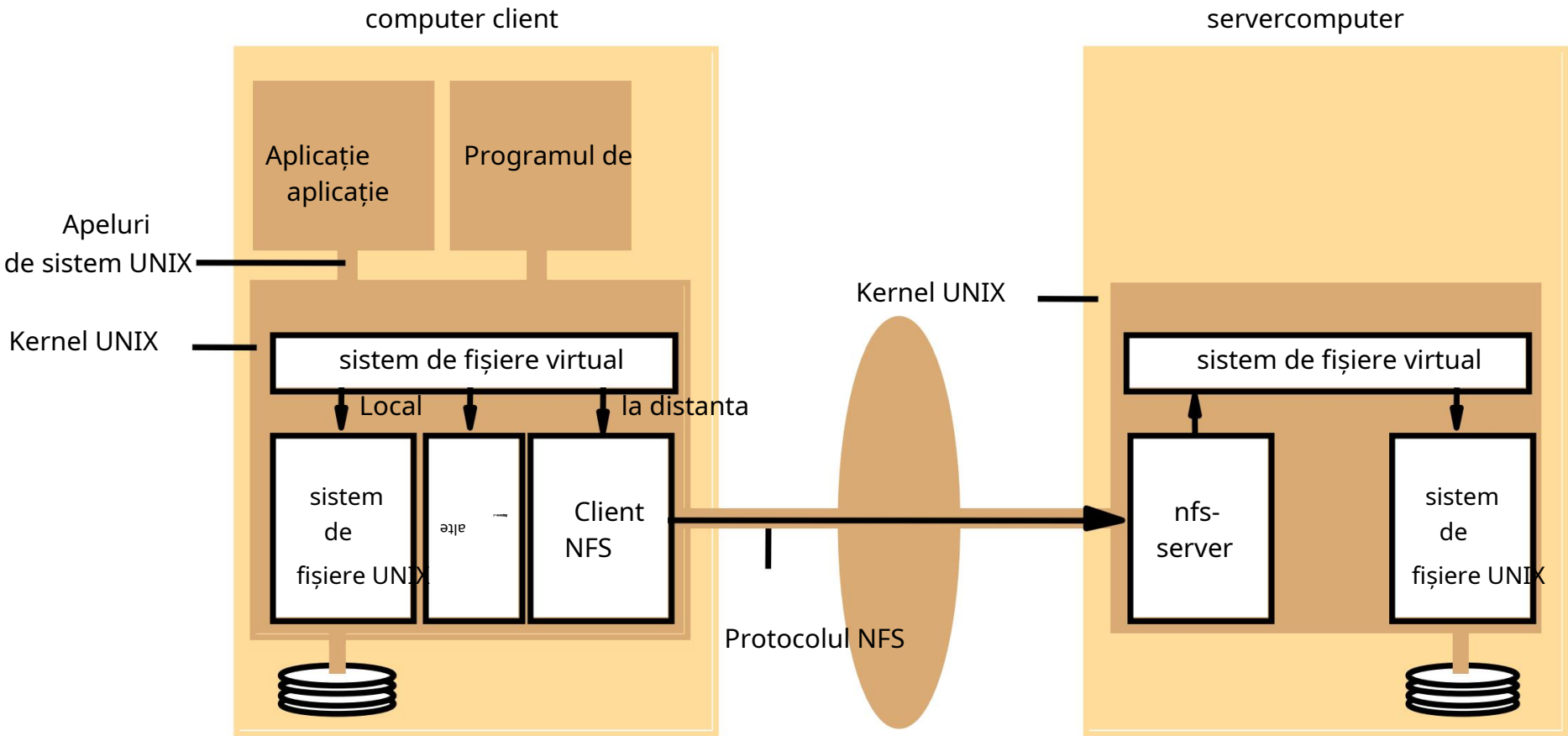
- NFS, Network File System
- 1985, proiectat pentru sistemul de operare UNIX
  - Suportă în prezent sisteme eterogene (MSDOS, Windows etc.)
- Scop: O colecție arbitrară de clienți și servere partajează un sistem de fișiere comun.
- Relația client/server este simetrică
  - Exporturi de server (/etc/exports)
  - Se montează client
- NFS pentru rețele locale (LAN) sau rețele de zonă largă (WAN)

NFS



NFS

- Arhitectura Sun NFS



NFS

- Toate implementările folosesc două protocoale:
  - protocolul nfs
  - protocol de montare
- protocol nfs: set de apeluri de procedură la distanță care oferă clienților mijloacele de a efectua operațiuni pe un sistem de fișiere la distanță
  - Server NFS module = server NFS
- Sun RPC a fost conceput pentru a fi utilizat cu NFS
  - Sun RPC acceptă TCP și UDP => NFS acceptă TCP și UDP
  - Interfața RPC către server este deschisă: orice proces poate executa invocări RPC pe server (dar acestea vor fi servite doar dacă sunt disponibile permisiunile corespunzătoare)

NFS

- Acces transparent => Sistem de fișiere virtual
  (Sistem de fișiere virtual, VFS)
  - Distingeți fișierele locale și cele de la distanță
  - Traduce mânerele de fișier NFS în mânere de fișiere
    fișier local (UFID)
  - Cunoaște sistemele de fișiere disponibile (locale sau la distanță)
- File handler în NFS => file handle

id Sistemul de fișiere	Numărul nodului-i	Număr Generaţie i-nod

NFS

- VFS au o structură pentru fiecare sistem de fișiere montat și un v-node pentru fiecare fișier deschis
  - Structura leagă sistemul de fișiere la distanță de directorul local unde este montat
  - Node-v indică dacă fișierul este local sau la distanță
- Operațiuni pe NFS ...

NFS

căutare(dirfh, nume) -> fh, attr	Returnează mânerul fișierului și atributele pentru numele fișierului în directorul dirfh.
create(dirfh, name, attr) -> newfh, attr	Creează un nou nume de fișier în directorul dirfh cu atributele attr și returnează noul handle de fișier și atributele.
eliminare (dirfh, nume) stare	Elimină numele fișierului din directorul dirfh.
getattr(fh) -> attr	Returnează atributele fișierului fh. (Asemănător apelului de sistem UNIX stat .)
setattr(fh, attr) -> attr	Setează atributele (mod, ID utilizator, ID grup, dimensiune, timp de acces și modificarea timpului unui fișier). Setarea dimensiunii la 0 trunchiază fișierul.
read(fh, offset, count) -> attr, data	Returnează până la un număr de octeți de date dintr-un fișier începând cu offset. Returnează, de asemenea, cele mai recente atribute ale fișierului.
write(fh, offset, count, data) -> attr	Scrie numărul de octeți de date într-un fișier începând cu offset. returnează atributele fișierului după ce a avut loc scrierea.
redenumește(dirfh, nume, todirfh, toname) -> stare	Schimbă numele numelui fișierului din directorul dirfh în toname în directorul către todirfh .
link(newdirfh, newname, dirfh, nume) -> stare	Creează o intrare newname în directorul newdirfh la care se referă nume de fișier în directorul dirfh.

NFS

link simbol(newdirfh, newname, șir) -> stare	Creează o intrare newname în directorul newdirfh de tip link simbolic cu șirul de valoare. Serverul nu interpretează șirul , ci creează un fișier de legătură simbolic pentru a-l păstra.
readlink(fh) -> șir	Returnează șirul care este asociat cu fișierul link simbolic identificat prin fh.
mkdir(dirfh, nume, attr) -> newfh, attr	Creează un nou nume de director cu atributele attr și returnează noul handle de fișier și atributele.
rmdir(dirfh, nume) -> stare	Îndepărtează numele directorului gol din directorul părinte dirfh. Eșuează dacă directorul nu este gol.
readdir(dirfh, cookie, count) -> intrări	Returnează până la un număr de octeți de intrări de director din directorul dirfh. Fiecare intrare conține un nume de fișier, un handle de fișier și un indicator opac către următoarea intrare de director, numită cookie. Cookie -ul este utilizat în apelurile ulterioare readdir pentru a începe citirea de la următoarea intrare. Dacă valoarea cookie -ului este 0, se citește de la prima intrare din director.
statfs(fh) -> fsstats	Returnează informații despre sistemul de fișiere (cum ar fi dimensiunea blocului, numărul de blocuri libere și așa mai departe) pentru sistemul de fișiere care conține un fișier fh.

NFS

- protocol de montare: Mijloace de comunicare între comanda mount și procesul pentru serviciul mountd remote mount .
  - Având în vedere calea unui director, returnează mânerul fișierului corespondent
  - Mount: efectuează acțiunile necesare pentru ca un dispozitiv să poată fi utilizat de sistemul de operare. Un dispozitiv și sistemul său de fișiere asociat sunt atașate la sistemul de fișiere general la un moment dat.
    - Rigidă
    - Flexibil
- Rezolvarea unui nume care face referire la un fișier de la distanță cu operații separate de căutare () .
  - O operație per director, mânerul de fișier returnat este folosit la următoarea căutare
  - Este implementat în acest fel deoarece poate fi nevoie să accesați o altă mașină dacă există un alt punct de montare între ele

NFS

- Automounter, montează dinamic directoarele de la distanță ori de câte ori se face referire la un punct de montare gol

- Serverul NFS este apatrid: nu păstrează informații despre fișierele deschise (nu există operațiuni Open și Open fișier). Închide)
  - Mai multă toleranță la erori
  - Informații de autentificare în fiecare apel RPC
  - Blocarea fișierelor deschise nu poate fi utilizată
    - Network Lock Manager și Network Status Monitor

NFS

- Ascunderea (utilizarea cache-urilor)
  - Pe server
    - Lectură, fără probleme de inconsecvență
    - Scrie, problemă dacă serverul se defectează
      - Scrieți prin cache
      - Scrieți pentru a închide (operația Versiunea 3 Commit())
  - În client, ei sunt responsabili să verifice dacă păstrează cea mai recentă copie
    - Citire, temporizator asociat blocurilor cache (3 sec -> date, 30 sec -> att). Când fișierul este deschis, se verifică ora ultimei modificări
    - Metodă de scriere, scriere întârziată (30 sec.)