Servicii stabile, bazate pe comunicația de grup, pentru accesul rapid la volume mari de date din sistemele distribuite

Teză de doctorat

Rezumat

ing. Adrian Colesa

Coordonator științific Prof. dr. ing. Iosif Ignat

Catedra de Calculatoare

2010

1 Introducere

Această lucrare prezintă câteva metode prin care se asigură disponibilitatea permanentă a datelor aflate la distanță, date la care accesul este oferit de către un sistem software specializat. Acest sistem este constituit din mai multe servicii distincte, ce se ocupă fiecare în parte de către anumite problemele legate de accesul rapid la date, dar care în același tip colaborează pentru atingerea aceluiași obiectiv. Posibilitatea accesului rapid și permanent la date implică necesitatea stabilității serviciilor ce asigură accesul la date și a eficienței metodelor folosite de acestea.

1.1 Contextul și motivația

Contextul în care este abordată problema accesului la date este cel al gestionării unor *volume mari de date*, aparținând unui număr mare de utilizatori. În mod firesc, utilizatorii își doresc ca datele să fie disponibile și furnizate imediat, oricând e nevoie de ele.

Problema gestionării volumelor mari de date și a oferirii *accesului permanent* și *rapid* la ele este una de stringentă actualitate. Ea se datorează atât acumulării permanente a datelor, chiar și la nivelul utilizatorilor obișnuiți, cât și datorită faptului că există o tendință evidentă de plasare a datelor și a serviciilor de acces la ele la nivelul centrelor de date specializate, ceea ce induce distanța dintre utilizator și date. Astfel, atât datele private ale utilizatorilor, cât și multe dintre aplicațiile acestora sunt mutate de la nivel local la distanță, la nivelul centrelor de date aflate în proprietatea unor companii specializate, care oferă servicii de stocare, gestionare, prelucrare și acces la date. Câteva dintre motivele pentru care se întâmplă acest lucru sunt:

- eliberarea utilizatorului obișnuit, nespecialist, de complexitatea și costurile necesare instalării, întreținerii și actualizării serviciilor locale;
- posibilitatea accesului la date și aplicații din orice loc de pe Internet;
- facilitatea partajării eficiente a datelor între utilizatori;

- simplificarea procesului de eliminare a erorilor și a problemelor de securitate din cadrul aplicațiilor (serviciilor).

Prin urmare, accesul la date se face prin intermediul unor aplicații, ce oferă servicii de gestionare și acces la date și situate de cele mai multe ori la distanță. Condiția de disponibilitate permanentă a datelor se răsfrânge la nivelul sistemului (aplicațiilor) ce asigură accesul la date în condiția ca serviciile componente ale acestuia să fie cât mai *stabile* și mai *eficiente* în funcționalitatea lor, chiar și în situațiile în care mediul în care se execută serviciile respective nu favorizează acest lucru. Stabilitatea unui serviciu înseamnă capacitatea acestuia de a-și onora permanent obligațiile și în orice condiții. În contextul amintit mai sus, al companiilor ce gestionează date și oferă acces la ele, cerința ca un serviciu să fie stabil este esențială atât pentru clienții serviciului, cât și pentru proprietarii acestuia, nerealizarea acestui deziderat costându-i pe ambii.

Stabilitatea unui serviciu e dată de o serie de alte caracteristici ale serviciului, precum toleranța la erori, fiabilitatea, scalabilitatea, disponibilitatea etc. În general, asigurarea acestor caracteristici, presupune folosirea în implementarea serviciului a unor componente redundante. Tehnica cea mai folosită în acest sens este cea de replicare, atât a datelor, cât și a componentelor software ale unui serviciu. Acest lucru implică o implementare și o funcționare *distribuită* a serviciului. Pentru obținerea unui raport performanță cost cât mai bun, se optează, în general, pentru utilizarea unui hardware obișnuit pe care să fie replicate componentele software. Din păcate însă astfel de resurse nu prezintă, la nivel individual, un grad ridicat de stabilitate. Acest lucru este valabil, atât în ceea ce privește calculatoarele folosite, cât și rețeaua ce le interconectează. Toate aceste aspecte fac ca asigurarea stabilității unui serviciu să fie o problemă reală și în același timp dificil de realizat.

1.2 Prezentarea generală

Tehnicile de folosire a componentelor redundante pentru asigurarea stabilității serviciilor de date și, implicit, pentru disponibilitatea datelor constituie așadar

obiectul cercetărilor noastre. Cercetări în această direcție există de mult timp și există și numeroase soluții. Am identificat însă o serie de probleme nerezolvate sau rezolvate parțial, care au justificat abordarea particulară a temei în cadrul acestei lucrări.

Este vorba în primul rând de faptul că, deși există multe soluțiile eficiente pentru rezolvarea a diferite probleme particulare ale stabilității unui serviciu, există puține abordări globale ale problemei. Prin urmare, unul din obiectivele principale ale lucrării a fost abordarea globală a problemei stabilității unui serviciu de date și dezvoltarea unui sistem care să permită integrarea și combinarea diferitelor tehnicilor de asigurare a stabilității serviciului. Lucrarea propune în acest sens arhitectura unui sistem de acces la date, care permite integrarea multora dintre tehnicile și soluțiile particulare existente. Sistemul este compus dintr-un set de mai multe servicii, fiecare având responsabilități ce vizează rezolvarea unui anumit aspect al stabilității. El este organizat pe două niveluri, unul *logic*, ce rezolvă problema stabilității din perspectiva ușurinței de găsire și selectare a fișierelor de interes în spații mari de date și unul *fizic*, ce rezolvă problema stabilității din perspectiva furnizării rapide a conținutului fișierelor vizate și a toleranței la erori. Organizarea sistemului de acces la date pe cele două niveluri, asigură utilizatorilor sistemului transparența față de poziția fizică a fișierelor și de modul de stocare distribuită al acestora. Serviciile de pe cele două niveluri ale sistemului sunt integrate astfel încât ele colaborează strâns pentru atingerea obiectivului comun, cel al stabilității de ansamblu a sistemului și implicit a disponibilității permanente a datelor gestionate. În acest sens, de exemplu, serviciile de replicare plasate la nivelul fizic, considerat nivelul de stocare distribuită a datelor, se folosesc de serviciile de descriere și organizare a datelor prin metadate (plasate la nivelul logic) pentru interpretarea șabloanelor de replicare asociate de administratorul sistemului nodurilor de stocare. Pe de altă parte, serviciile nivelului logic, jucând și rolul de servicii ce permit și asigură clienților accesul la datele sistemului, se folosesc de serviciile de replicare pentru transferul eficient al fișierelor de pe nodurile de replicare pe cele ale clienților. În felul acesta, am dezvoltat o modalitate de tratare uniformă, ca replicare de date, a oricărui tip de transfer de date, atât

în cazul distribuirii datelor la nivelul nodurilor de stocare pentru realizarea copiilor redondante, cât și în cazul furnizării datelor clienților.

La nivelul serviciilor de replicare am propus integrarea a două tehnici diferite, folosite în contexte diferite, și anume, cea de trimitere ("împingere") a datelor, folosită în general în replicarea datelor pe nodurile de stocare, respectiv cea de aducere ("tragere") a datelor simultan din mai multe surse. Am propus combinarea cele două tehnici în ambele situații, pentru a se putea obține algoritmi de transfer care să se adapteze și să folosească eficient toate legăturile existente între nodurile de stocare.

De asemenea, am întreprins cercetări și la nivelul unor probleme particulare legate de stabilitatea serviciilor de date, probleme a căror rezolvare nu a fost încă făcută satisfăcător. Una dintre problemele abordate în acest sens a fost aceea a găsirii unei metode transparente și generale de asigurare a stabilității unui serviciu. Ne-a interesat la modul particular, servicii cu o implementare centralizată, care nu putea fi modificată. În acest sens transparența am interpretat-o nu doar ca transparență a clientului față de mecanismele interne ale serviciului, ci și ca transparență a serviciului însuși față de aspectele de natură distribuită prin care i se asigură stabilitatea. Am dezvoltat în acest sens o metodă de replicare independentă de serviciul tratat și care maschează în mod transparent față de clienți căderile unora dintre componentele serviciului. Metoda se bazează pe rularea serviciului într-o mașină virtuală plasată pe un nod fizic principal și replicarea mașinii virtuale respective pe alte noduri fizice secundare. În momentul în care mașina virtuală principală nu mai funcționează (cade), oricare dintre cele de pe nodurile secundare poate fi pornită pentru a asigura funcționarea continuă a serviciului.

Una dintre tehnicile importante integrate în sistemul propus de noi este cea de *comunicație de grup*. E vorba de fapt de două servicii diferite și anume, cel de gestionare a componenței unui grup de procese și respectiv, cel de trimitere a mesajelor de multicast în cadrul unui grup de procese. Tehnicile respective sunt utile în cadrul grupurilor de procese ce colaborează pentru expunerea spre exterior a unei funcționalități unitare. Ele sunt, în cazul sistemului nostru, fie replici ale aceluiași serviciu, fie gestionează datele

replicate. Principala responsabilitate a unui proces din cadrul unui astfel de grup este aceea de a asigura consistența replicii gestionate local cu celelalte replici de la distanță. În acest sens este esențială în primul rând cunoașterea de către toți membri grupului a componenței acestuia, pentru a putea lua decizii coerente și consistente la nivelul grupului. Este exact ceea ce oferă serviciul de gestionare a componenței grupului. De asemenea, comunicarea între procese trebuie efectuată în mod consistent, fiabil și tolerant la erori la nivelul întregului grup, lucru de care este responsabil serviciul de multicast. Cele două aspecte sunt însă dificil de realizat în cazul sistemelor distribuite a căror resurse hardware manifestă caracteristici de instabilitate. Așa cum am precizat și mai sus, lucrarea de față se plasează în contextul unui astfel de sistem. Modelul de descriere folosit în acest caz este cel de sistem distribuit asincron. În sistemele distribuite asincrone nu se pot face presupuneri legate de existența unor limite de timp, nici în ceea ce privește execuția unor operații, nici în ceea ce privește transmiterea unor mesaje. În acest sens, în cadrul comunicării a două procese aflate pe noduri diferite, atunci când unul dintre ele nu primește răspunsul așteptat de la celălalt, nu se poate face distincție între următoarele cazuri: 1. procesul partener nu mai funcționează; 2. procesul partener funcționează, dar foarte lent; 3. legătura dintre cele două procese este întreruptă; 4. legătura dintre procese funcționează, dar foarte lent.

În astfel de cazuri, procese diferite din grup pot ajunge la concluzii și stări diferite, ceea ce per ansamblu se poate manifesta printr-o funcționare inconsistentă a grupului de procese sau chiar la indisponibilitatea serviciului oferit de grup. Comunicația de grup rezolvă exact astfel de situații, eliberând pe dezvoltatorii de aplicații de complexitatea problemelor ce nu țin strict de funcționalitatea aplicației și furnizând o metodă uniformă de dezvoltare a aplicațiilor distribuite. Am folosit comunicația de grup în cadrul grupurilor de procese ce asigură în mod colaborativ stabilitatea unui serviciu, în sensul toleranței la erori, adică a mascării căderilor unora dintre membrii grupului.

Un alt aspect al stabilității abordat în cadrul lucrării a fost acela al integrării, sub forma unui sistem de fișiere, unor metode eficiente de organizare și regăsire a datelor în spații mari de date. Problema respectivă a fost mai

fost abordată parțial la nivelul local al sistemelor de fișiere, dar nu a fost integrată la nivelul unui serviciu de date. Am încercat să facem noi, deci acest lucru. Tehnica rezultată constă în asocierea de informații de descriere, așanumite metadate, fișierelor gestionate de sistemul propus de noi. Metadate sunt folosite la stabilirea de relații între fișiere. Pe baza relațiilor respective pot fi apoi regăsite rapid diferite seturi de fișiere, plasate în diferite locuri în sistemul de fișiere. Practic, am aplicat modelul bazelor de date relaționale la nivelul sistemului de fișiere, dezvoltând pentru aceasta o serie de concepte și tehnici prin care modelul respectiv a putut fi integrat în sistemul de operare și făcut accesibil aplicațiilor și utilizatorilor printr-o interfață specifică sistemelor de fișiere.

1.3 Contribuțiile tezei

În contextul descris mai sus, principalele subiecte abordate în cadrul acestei lucrări și rezultatele obținute sunt:

- o analiză sistematică și completă a problemelor ce apar în încercarea de a asigura stabilitatea unui serviciu de acces la date;
- realizarea unei arhitecturi detaliate a unui sistem de gestionare a datelor, care să asigure clienților săi un acces permanent la date;
- dezvoltarea unei modalități de organizare a nodurilor de stocare bazate pe șabloane de replicare, care să asigure o scalabilitate mărită procesului de replicare a datelor și faptul că datele sunt replicate pe nodurile unde trebuie să ajungă;
- implementarea și testarea în Grid a sistemului propus, folosind serviciile Globus Toolkit 4;
- dezvoltarea unei tehnici de execuție distribuită replicată a unui serviciu cu implementare centralizată, care să asigure în mod transparent atât pentru serviciu în sine, cât și pentru clienții săi stabilitatea serviciului; strategia e bazată pe rularea serviciului într-o mașină virtuală și replicarea întregii mașini virtuale;

- dezvoltarea unui protocol de replicare asincron adaptiv, care să reducă întârzierile introduse de protocolul de asigurare a stabilității serviciului la nivelul timpilor de răspuns la cererile clienților; protocolul se adaptează comportamentului curent al serviciului protejat și stării curente a rețelei ce leagă nodurile de replicare;

- implementarea peste sistemul de virtualizare Xen a unui sistem de asigurarea transparentă a stabilității unui serviciu centralizat, folosind tehnica de virtualizare și replicare dezvoltată;

- dezvoltarea și implementarea unei metode de actualizare dinamică a modulelor Linux în timpul rulării sistemului și folosirii lor;

- dezvoltarea unui sistem de fișiere bazat pe modelul relațional, care permite organizări flexibile a fișierelor și posibilități multiple de regăsire a lor, tehnici utile în spații mari de date; integrarea sistemului propus în sistemele de operare se poate face păstrând compatibilitatea cu aplicațiile existente;

- implementarea în Linux a sistemului de fișiere dezvoltat;

1.4 Structura lucrării

Teza de doctorat este structurată sub forma a cinci capitole, similar cu organizarea acestui rezumat. Primul capitol, cel de introducere, prezintă tema tezei, motivația abordării temei respective, contextul și particularitățile abordării propuse. Cel de-al doilea capitol reunește logic temele abordate în celelalte capitole din cadrul tezei, descriind cadrul în care teme aparent diferite, converg. În capitolul respectiv propunem un sistem complex de gestionare și acces la date care are ca scop principal asigurarea disponibilității datelor, ideal disponibilitatea permanentă. Al treilea capitol este dedicat problemei stabilității unui serviciu, mai precis a posibilităților de asigurare a stabilității unui serviciu proiectat și implementat în manieră centralizată. Al patrulea capitol prezintă o posibilă implementare, la nivelul și sub forma

unui sistem de fișiere, a unuia dintre serviciile sistemului propus în cel de-al doilea capitol. E vorba de serviciul de gestionare a metadatelor atașate fișierelor. Sistemul propus se bazează pe modelul de date specific bazelor de date relaționale. Motivul integrării lui ca sistem de fișiere este cel al transparenței și compatibilității cu aplicațiile existente. Noutatea propunerii constă așadar nu în modelul de date, ci în modalitatea integrării lui la nivelul sistemului de fișiere și în avantajele pe care le aduce în contextul unor volume mari de date, concretizate în cazul nostru sub forma unor spații mari de fișiere. Al cincilea capitol sintetizează concluziile și principalele realizări și direcții de dezvoltare ale acestei lucrări.

2 DIPED: sistem de asigurare a disponibilității permanente a datelor

În cadrul acestui capitol este descrisă arhitectura unui sistem software distribuit, care are ca scop asigurarea disponibilității permanente a unui set de date, motiv pentru care l-am numit numit DIPED (DIsponibilitate PErmanentă a Datelor). Datele sunt plasate ca fișiere pe diferite noduri de stocare în cadrul unui rețele extinse și se dorește a fi accesibile oricând și de pe oricare nod al rețelei. Sistemul propus oferă posibilitatea accesului transparent, rapid și fiabil la datele pe care le gestionează.

DIPED se înscrie în categoria sistemelor de fișiere distribuite, însă nu este un sistem complet din această perspectivă. Principale obiective vizate sunt nu atât caracteristicile funcționale ale unui sistem de fișiere distribuit, cât cele non-funcționale, precum eficiența, fiabilitatea, toleranța la erori, generalitatea și adaptabilitatea la context a metodelor adoptate. Aspectele asupra cărora ne-am concentrat atenția sunt cele legate de distribuirea, organizarea și accesul rapid la date.

Principala tehnică de care ne-am folosit este replicarea. Ea a fost aplicată atât la nivelul datelor, cât și a serviciilor ce asigură gestionarea și accesul la date. DIPED nu oferă suport pentru modificarea replicilor gestionate, ci doar acces pentru citire, datorită faptului că am urmărit cu precădere dezvoltarea unor metode eficiente de distribuire și acces la date.

Capitolul este organizat sub forma a 7 secțiuni. Prima secțiune este cea de introducere, care prezintă motivația, contextul în care am abordat tema propusă și obiectivele propuse. A doua secțiune prezintă comparativ sistemul DIPED și metodele propuse de noi cu metode și proiecte similare din literatura de specialitate. Ultima secțiune este cea de concluzii, în care sunt rezumate rezultatele obținute, contribuțiile personale și posibilitățile de dezvoltare ulterioară. Celelalte patru secțiuni constituie partea centrală și consistentă a capitolului și pot fi văzute ca formând două parți, și anume: o parte teoretică și respectiv, una practică, de implementare și testare a unora dintre ideile și metodele propuse în prima parte. Vom analiza mai jos pe scurt cele patru secțiuni.

Secțiunea a treia a capitolului constituie partea de analiză a problemelor specifice asigurării disponibilității continue a datelor situate la distanță. Ea poate fi privită de asemenea ca secțiunea de definire a unor termeni și tehnici folosite în restul lucrării. Analiza pornește de la identificare replicării, ca metodă de bază necesară atingerii obiectivelor propuse și continua apoi cu identificarea tuturor celorlalte probleme ce trebuie tratate: *toleranța la erori*, *fiabilitatea*, *scalabilitatea*, *eficiența*. Unul dintre obiectivele importante propus a fi atins este cel de *transparență*, aceasta fiind înțeleasă ca ascunderea față de client a detaliilor de implementare a sistemului de gestionare a datelor și neimplicarea lui în mecanismele de asigurare a disponibilității datelor. În urma analizei efectuate, sunt identificate anumite funcționalități ce ar trebuie asigurate de către sistemul DIPED și sunt, de asemenea, propuse serviciile componentele care ar putea să ofere funcționalitatea respectivă.

Secțiunea a patra constituie partea centrală și cea mai consistentă a capitolului. Ea aparține atât părții teoretice a capitolului, prin analiza pe care o face serviciilor componente ale sistemului DIPED propus, cât și părții practice, prin prezentarea detaliată a arhitecturii sistemului și a componentelor sale. Secțiunea nu prezintă însă soluții de implementare a serviciilor respective, ci doar rolul fiecăruia și modalitatea de interacțiune și colaborarea între servicii pentru atingerea obiectivelor globale ale sistemului. Soluții de implementare a unora dintre serviciile respective vor fi descrise în celelalte capitolele ale tezei.

Secțiunea începe cu propunerea unei arhitecturii a sistemului DIPED, în care acesta este format din mai multe servicii, ce asigură funcționalitatea de ansamblu a sistemului. Serviciile sunt organizate pe două niveluri: *nivelul logic*, de organizare a datelor și respectiv, *nivelul fizic*, de stocare a datelor. Datele sunt considerate stocate în fișiere. Nivelul logic permite asocierea de metadate (proprietăți de descriere) fișierelor și posibilitatea organizării și perceperii spațiului de fișiere în forme multiple și dinamice, pe baza stabilirii de relații între fișiere prin intermediul metadatelor lor. Nivelul logic, asigură, de asemenea, pentru clienții sistemului, transparența de locație și de faptul că fișierele au replici. Nivelul fizic, asigură stocarea distribuită prin replicare a fișierelor, astfel încât să se asigure proprietățile de toleranță la indisponibilitatea unor noduri de stocare și eficiența de acces la date. Astfel, dacă unele dintre nodurile de stocare devin indisponibile, fișierele de pe acele noduri vor putea fi accesate prin replicile lor existente pe nodurile de stocare rămase funcționale. De asemenea, accesul la fișierele de dimensiuni mari va putea fi făcut prin transferul simultan spre client a fragmente diferite ale fișierelor de pe noduri de stocare diferite, care dispun de replici ale fișierelor respective.

Serviciile propuse să facă parte din componența sistemului DIPED sunt:

- serviciu de gestionare a metadatelor (SGM)
- serviciu de acces la date (SAD)
- serviciu de replicare a datelor (SRD)
- serviciu de gestionare a replicilor (SGR)
- serviciul de transfer al datelor (STD)
- serviciu de comunicație de grup (SCG)

Secțiunea a patra prezintă, de asemenea, modalitățile de interacțiune cu serviciile sistemului DIPED, implementate în general ca grupuri de procese, pentru a li se asigura stabilitatea.

Sunt analizate apoi o serie de scenarii de funcționare a sistemului DIPED, ilustrându-se intercomunicarea și colaborarea serviciilor sistemului. Este

vorba despre scenariile de: introducere a unui nou nod de replicare (stocare) în sistem, replicare a datelor nou introduse în sistem, actualizarea catalogului de replici și respectiv, de acces la date al clienților. Analiza scenariilor respective a contribuit la definitivarea completă a funcționalității și interfeței serviciilor sistemului. Ele au contribuit, de asemenea, la dezvoltarea unor tehnici speciale de gestionare a replicării datelor și a accesului la ele.

Merită menționat în acest sens, faptul că am propus două categorii de clienți ai sistemului, și anume: *implicați* și *neimplicați*. În cazul clienților neimplicați, sistemul va asigura o transparență totală, relativ la structura lui internă. În cazul clienților implicați sistemul va asigura o eficiență mărită de furnizare a conținutului fișierelor și, de asemenea, permite și poate gestiona participarea clienților respectivi la furnizarea datelor către alți clienți. În acest fel, nodurile clienților implicați devin parte a sistemului, ca noduri de stocare. Tratarea uniformă a nodurilor de stocare, fie cele dedicate, ale sistemului DIPED, cât și cele ale clienților implicați este făcută prin intermediul unor *șabloane de replicare*. Fiecare nod de stocare are asociat un șablon de replicare, care indică ce anume este permis să se stocheze pe acel nod sau, dintr-o altă perspectivă, ce anume se dorește a fi stocat (replicat) pe acel nod. Utilizarea șabloanelor de replicare permite sistemului să realizeze o replicare automată pentru menținerea unui anumit grad de reziliență a datelor, în cazul în care unele dintre nodurile ce stocau deja replici ale datelor respective devin indisponibile. Șabloanele de replicare sunt exprimate în termeni specifici nivelului logic, folosind metadate, dar sunt folosite la nivelul fizic. Acest lucru a impus o comunicare între cele două servicii, comunicare făcută prin intermediul serviciului de gestionare a replicilor.

O altă tehnică particulară introdusă la nivelul serviciilor sistemului DIPED a fost aceea de tratare uniformă a oricărui tip de transfer de date (fie de replicare pe nodurile de stocare, fie de furnizare spre client) ca replicare de date și implicit a oricărui nod din sistem (fie de dedicat pentru stocare, fie al unui client) ca nod de replicare. Responsabilitatea transferurilor de date de orice tip revine serviciului de replicare, care realizează acest lucru prin aplicarea combinată a două tehnici: trimiterea datelor (distribuire) de la un nod sursă la mai multe destinații, respectiv aducerea datelor simultan de la mai

multe noduri pe un singur nod destinație. Inițiativa operațiilor de transfer (replicare) aparține însă serviciilor de acces la date, în cazul cererii datelor de către clienți, respectiv serviciului de gestionare a metadatelor, în cazul introducerii unor noi date în sistem, ce trebuie stocate la nivelul fizic.

Secțiunea a patra prezintă în final câteva modalități de implementare efectivă a serviciilor propuse.

Secțiunea a cincea a capitolului prezintă implementarea în Grid a unui prototip al sistemului DIPED propus. Implementarea se bazează pe serviciile infrastructurii software Globus Toolkit 4 (GT4) de gestionare a sistemelor Grid, pe care le utilizează și le extinde. Este vorba de serviciile GridFTP, RFT, RLS și DRS, care sunt mapate pe funcționalitatea unora dintre serviciile propuse în DIPED. Funcționalitatea acestora a fost extinsă prin introducerea unor servicii noi ce asigură, pe de o parte, toleranța la erori și pentru cazuri neprevăzute de serviciile existente în GT4, iar pe de altă parte automatizează replicarea datelor, atunci când e necesară. În cadrul aceleiași secțiuni, propunem, de asemenea, și demonstrăm funcționarea corectă a unei strategii de organizare a nodurilor de replicare sub forma unei ierarhii de grupuri de replicare construite pe baza șabloanelor de replicare. Structura propusă asigură scalabilitatea serviciul de replicare și automatizarea operației de replicare.

În secțiunea a șasea capitolului prezentăm o serie de teste efectuate pe prototipul implementat în Grid și rezultatele obținute.

Idei și părți din acest capitol au fost publicate în [1], [2], [3], [4], [5] și [6]. De asemenea, activitatea de cercetare desfășurată a fost motivată și susținută parțial prin grantul de cercetare 19CEEX-I03 (MedioGRID) finanțat de Ministerul Educației și Cercetării din România.

3 Tehnici de asigurare transparentă a stabilității unui serviciu centralizat

În cadrul acestui capitol prezentăm câteva tehnici pe care le propunem pentru a asigura un grad ridicat de stabilitate implementării centralizate existente a unui serviciu care deservește clienți aflați la distanță. Prin serviciu stabil am

înțeles un serviciu ce este tolerant la erori, fiabil și scalabil. Serviciul căruia i se asigură stabilitatea prin tehnicile propuse de noi l-am numit *serviciu protejat*, iar sistemul ce aplică tehnicile respective prin termenul *sistem de protecție*.

Există două lucruri ce particularizează abordarea noastră. În primul rând, ne-am propunem ca tehnicile dezvoltate să fie cât mai *generale*, astfel încât să poată fi utilizate pentru orice serviciu, în mod independent de funcționalitatea acestuia și de sistemul de operare pe care rulează. În al doilea rând, am urmărit ca modalitatea prin care se acționează asupra serviciului ce trebuie protejat să fie absolut *transparentă* pentru serviciul protejat, aceasta însemnând că nu este nevoie de modificarea implementării existente a serviciului. Înseamnă deci că serviciul protejat nu colaborează cu sistemul de protecție pentru asigurarea propriei stabilități și, prin urmare, nu va fi implicat în tehnicile aplicate de acesta, ceea ce va face mai dificilă misiunea lui. Prin caracteristica de transparență față de serviciu a tehnicilor propuse de noi am vizat în special serviciile a căror modificare (reproiectare și reimplementare) este fie imposibilă (de exemplu, codul nu este disponibil), fie foarte costisitoare în termeni de timp și/sau resurse.

Capitolul este organizat sub forma a 7 secțiuni. Prima secțiune este cea de introducere și prezintă motivația abordării temei și obiectivele propuse. A doua secțiune prezintă comparativ metoda propusă de noi cu metode și proiecte similare din literatura de specialitate. Ultima secțiune, cea de concluzii, rezumă rezultatele obținute, contribuțiile personale și posibilitățile de dezvoltare ulterioară. Celelalte patru secțiuni constituie partea centrală și consistentă a capitolului și pot fi văzute ca formând două părți, și anume: o parte teoretică și respectiv, una practică, de implementare și testare a unora dintre ideile și metodele propuse în prima parte. Vom analiza mai jos pe scurt cele patru secțiuni.

În cadrul celei de-a treia secțiuni a capitolului facem o analiză a modalităților prin care se poate asigura stabilitatea unui serviciu centralizat și identificăm problemele ce apar în încercarea de a face acest lucru în mod transparent. Stabilim, de asemenea, posibile soluții la aceste probleme, soluții pe care le detaliem în celelalte secțiuni. Tehnica principală identificată ca necesară pen-

tru asigurarea stabilității unui serviciu este cea de replicare a serverului unic ce implementează serviciul la nivelul mai multor noduri fizice, astfel încât să se formeze un grup de servere. Replicarea și asigurarea consistenței replicilor poate asigura toleranța la erori, dar pentru stabilitatea serviciului trebuie folosită și tehnica de echilibrare a solicitării replicilor. Aceste probleme sunt dificil de tratat în contextul abordat de noi, în care serverele nu colaborează nici între ele și nici cu sistemul de protecție. În urma analizei făcute s-a ajuns la concluzia că singura soluție posibilă pentru atingerea obiectelor propuse o reprezintă *virtualizarea* serviciului, adică rularea serverului serviciului într-o mașină virtuală, și *replicarea* întregii *mașini virtuale*. Tehnica de replicare folosită trebuie să fie ce a de replicare pasivă, cu un nod principal și celelalte noduri, noduri secundare. Pe de o parte, această tehnică asigură transparență atât pentru clienții serviciului protejat, cât și pentru serviciul însuși, însă, pe de altă parte, posibilitatea echilibrării încărcării serverelor replicare ale serviciului poate fi făcută doar pentru anumite tipuri de servicii. Am propus în acest sens o posibilă configurație de echilibrare, în care sunt rulate simultan mai multe grupuri (tuple) independente de serverele ale serviciului protejat, fiecare în configurația server principal – servere secundare, iar cererile clienților sunt distribuite echilibrat între grupurile de servere. Această configurație de echilibrare funcționează doar pentru serviciile în care cererile clienților diferiți pot fi tratate independent unele de altele și restricționează, de asemenea, tratarea tuturor cererilor unui client de către același grup de servere.

În secțiunea a patra a capitolului descriem detaliat posibilitatea și problematica realizării metodei de virtualizare și replicare, arătând modul în care un serviciu centralizat este executat ca unul distribuit și cum se asigură astfel stabilitatea lui. Tehnica de replicare pasivă a fost aplicată în două variante, numite de noi *sincronă* și respectiv, *asincronă*. Replicarea este aplicată în mod continuu în faze succesive, metoda sincronă efectuând operațiile replicării secvențial, pe când ce asincronă introducând un paralelism al acestor operații între faze diferite.

După descrierea celor două variante, am dezvoltat o metodă de evaluare a eficienței lor, din perspectiva întârzierilor pe care le introduc la nivelul

timpului de răspuns la cererile clienților, atât în cazul funcționării normale a serviciului, cât și în cazul apariției unor erori, mai precis căderea serverului principal. Am dedus în acest sens formulele de calcul a întârzierii maxime posibil a fi introduse în fiecare din cazurile considerate. Pe baza analizei făcute am propus apoi un protocol de replicare adaptiv, care încearcă să diminueze cât mai mult întârzierile introduse, pentru a le menține în cadrul unor limite predefinite, specifice serviciului. El face acest lucru luând în calcul atât anumiți parametri ce caracterizează comportamentul curent al serviciului — cum ar fi, de exemplu, numărul de pagini de memorie modificate, gradul de focalizare a modificărilor memoriei, numărul de pachete de ieșire generate ca răspuns la cererile clienților, cât și capacitatea curentă de transfer a rețelei ce leagă nodul principal de cele secundare. Ultima parte a secțiunii a patra prezintă o implementare parțială a metodelor propuse în sistemul de virtualizare Xen, implementare care se folosește de operația de migrarea a unei mașini virtuale oferite de Xen, operație pe care o modifică pentru a o transforma într-un transfer de replicare continuu al mașinii virtuale de pe serverul principal spre cel secundar.

În secțiunea a cincea descriem o tehnică de asigurare a stabilității unui serviciu prin posibilitatea actualizării dinamice, în timpul funcționării, a sistemului de operare peste care rulează serviciul. Sistemul de actualizare propus este construit pe mecanismul de gestionare a modulelor din Linux, mecanism pe care îl extinde pentru a permite actualizarea sau înlocuirea modulelor încărcate, în tipul funcționării sistemului și a utilizării lor. Sistemul de actualizare dinamică se bazează pe tehnica de actualizare indirectă, un modulul actualizabil fiind împărțit în două componente, cea de interfață și cea de implementare. Componenta actualizabilă este doar cea de implementare, cea de interfață funcționând pe post de nivel de redirectare a cererilor adresate modulului spre componenta de implementare actuală.

Secțiunea a șasea a capitolului conține rezultatele de test făcute pe ambele sisteme dezvoltate.

În cazul sistemului de protecție prin virtualizare și replicare din Xen am efectuat teste pe diferite tipuri de servicii, cu diferite caracteristici de modificare a memoriei și comunicare prin rețea (*compilarea nucleului Linux* și

testul *nbench*), analizând comparativ rezultatele obținute de metoda de replicare sincronă și asincronă. Am efectuat, de asemenea, o serie de teste, pe care ne-am bazat în dezvoltarea algoritmului de replicare adaptiv, ca de exemplu testele de focalizare a modificărilor de memorie. Ca o concluzie a testelor efectuate, am observat că factorul hotărâtor în ceea ce privește performanța protocolului de replicare din sistemul de protecție propus este raportul dintre numărul de pagini ce trebuie transferate și capacitatea legăturilor dintre nodul principal și cele secundare. Cu cât acest raport este mai mare, fie datorită modificărilor intense ale memoriei efectuate de către serviciul protejat, fie datorită degradării capacității de transfer a legăturilor, cu atât întârzierile percepute de către clienții serviciului protejat vor fi mai mari. Prin urmare, în anumite condiții date privind caracteristicile rețelei, protocolul de protecție propus nu este utilizabil pentru orice tip de serviciu, în special pentru cele ai căror clienți pot fi afectați de întârzierile maxime introduse de protocol în condițiile date. Ca regulă generală însă, protocolul nostru de protecție poate fi aplicat cu succes serviciilor care solicită puțin memoria. Probabil că în realitate serviciile se situează între cele tipuri simulate de noi. De asemenea, e foarte posibil ca întârzierile acceptate la nivelul clienților să depindă de tipul de servicii. Astfel, de exemplu, este foarte posibil ca clienții unui serviciu de tipul celui simulat prin compilarea nucleului Linux să accepte întârzieri de ordinul secundelor, știind că cererile lor declanșează operații computaționale de durată, lucru care se potrivește ordinului întârzierilor introduse de protocolul de replicare, putându-se în acest caz mări considerabil și timpul de rulare a mașinii într-o fază a replicării, pe de o parte pentru a reduce întârzierile, iar pe de altă parte pentru a face mai eficientă utilizarea procesorului. În ceea ce privește comparația dintre strategia asincronă și sincronă, prima se dovedește eficientă în condițiile în care transferul este operația dominantă, adică exact în cazul creșterii raportului pagini modificate, capacitate de transfer a rețelei. În condiții optime de funcționare, câștigul de performanță este nesemnificativ.

În cazul sistemului de actualizare dinamică a modulelor Linux am testat încărcarea suplimentară introdusă de sistemul propus în cazul modulelor de gestionare a sistemelor de fișiere *msdos* (*FAT16*) și *vfat* (FAT32). Întârzierea

introduse la actualizarea modulelor testate au fost în medie de aproximativ 3.30%, ceea ce reprezintă un rezultat rezonabil, având în vedere că actualizarea unui modul este o operație relativ rară.

Idei și părți din acest capitol au fost publicate în [7], [6] și [8].

4 Organizarea flexibilă și regăsirea rapidă a datelor în sistemele de fișiere folosind relațiile între fișiere

În acest capitol prezentăm o posibilă implementare ca sistem de fișiere integrat în cadrul sistemului de operare, a unuia dintre serviciile sistemului DIPED propus în Capitolul 2. E vorba de serviciul de gestionare a metadatelor atașate fișierelor și de modul în care acesta permite utilizatorilor organizarea datelor și găsirea datelor dorite.

Arhitectura și implementarea pe care le-am propus în acest capitol pentru serviciul de gestionare a metadatelor se bazează pe concepte și tehnici prin care să se permită un acces cât mai rapid la date, în contextul unor spații de fișiere foarte mari. Reținem faptul că nu obținerea efectivă a datelor din fișier face obiectul acestui capitol, ci găsirea setului numelor fișierelor în care sunt stocate datele vizate. Prin urmare, sistemul urmărește să faciliteze clienților săi găsirea cât mai rapidă a fișierelor necesare.

Soluția propusă are la bază stabilirea de diverse relații între fișiere sau chiar seturi de fișiere. Acest lucru se va face prin atașarea de proprietăți (metadate) sau "etichete" fișierelor, lucru ce poate fi făcut atât manual de către utilizator, cât și automat pentru anumite tipuri de fișiere de programe specializate. Sistemul permite organizare simultană a fișierelor pe diferite criterii, ceea ce oferă posibilități multiple de regăsire a unui fișier. În principiu sistemul se folosește atât de tehnici specifice organizării și accesului la bazele de date relaționale, cât și de cele specifice sistemelor de fișiere clasice, mai precis de organizarea ierarhică. Noutatea propunerii constă nu în modelul de date, ci în modalitatea integrării lui la nivelul sistemului de fișiere și în avantajele pe care le aduce în contextul unor volume mari de date, concretizate în cazul nostru sub forma unor spații mari de fișiere. Motivul integrării soluției propuse la nivelul sistemului de fișiere este cel al transparenței și

compatibilității cu aplicațiile existente.

Capitolul este organizat sub forma a 8 secțiuni. Prima secțiune este cea de introducere și prezintă motivația abordării temei și obiectivele propuse. A doua secțiune prezintă comparativ metoda propusă de noi cu metode și proiecte similare din literatura de specialitate. Ultima secțiune, cea de concluzii, rezumă rezultatele obținute, contribuțiile personale și posibilitățile de dezvoltare ulterioară. Celelalte cinci secțiuni constituie partea centrală și consistentă a capitolului și le vom analiza mai jos pe scurt pe fiecare.

În secțiunea a treia a capitolului propunem și definim o serie de concepte necesare noului model de date folosit de sistemul nostru de fișiere. Principalul concept este cel de *proprietate* asociată unui fișier. Cu ajutorul proprietăților se pot stabili diferite relații între fișiere, precum: apartenența la aceeași categorie, apartenența la aceeași clasă, dependența de un alt fișier etc. Tot aici prezentăm, de asemenea, și noua formă de interacțiune a utilizatorului cu sistemul, și anume cea de *interogare*, prin care căutarea unor fișiere sau seturi de fișiere se face prin precizarea unor proprietății ale acestora sau a relațiilor dintre ele. Spre deosebire însă de modalitatea de interacțiune prin interogare cu bazele de date sau cu motoarele de căutare, sistemul nostru propune și folosirea tehnicii tradiționale de *navigare*, specifică sistemelor de fișiere ierarhice. În acest sens, rezultatul unei interogări este plasat în cadrul unui director virtual, în cadrul căruia este posibilă rafinarea interogării inițiale prin navigarea în subdirectoare ce corespund unor criterii de căutare rămase nespecificate în cadrul expresiei ce descrie interogarea inițială.

În cea de-a patra secțiune a capitolului, descriem arhitectura sistemului propus, integrat ca sistem de fișiere în sistemul de operare. Sistemul este compus din trei componente principale: *modulul de stocare*, *modulul de gestionare a bazei de date* și *modulul de prezentare*. Structura lui este una specifică unui sistem de fișiere, componenta adițională fiind cea de gestionare a bazei de metadate asociate fișierelor.

Ceea ce aduce nou sistemul este însă în primul rând modalitatea de implementare a interfeței tradiționale de apeluri sistem de interacțiune a aplicațiilor utilizator cu sistemul de fișiere, nivel la care introduce o reinterpretare a unora dintre conceptele sistemelor de fișiere tradiționale, în special

cel de director și cel de navigare. Este exact ceea ce descrie secțiunea a cincea ca și modalitate de integrare transparentă în sistemul de operare și de păstrare a compatibilității cu aplicațiile existente. Această parte a capitolului, conține de altfel una dintre particularitățile sistemului nostru prin comparație cu alte soluții similare, și anume posibilitatea de navigare prin intermediul interfeței tradiționale de apeluri sistem, și implicit prin aplicațiile existente de explorare a spațiului de fișiere, pe căi multiple spre același fișier. Modalitatea în care am realizat acest lucru a fost prin generarea, la nivelul fiecărui director a unui subdirector virtual, numit *Classification*, care oferă aplicațiilor existente acces spre zona de explorare specială oferită de sistemul nostru, bazată pe clasificarea fișierelor și pe interacțiunea prin interogare. O altă particularitate a sistemului propus este aceea că tratează în mod similar atât fișierele individuale, cât și categoriile de fișiere, ceea ce face posibilă navigarea prin întregul spațiu de fișiere, organizat pe un schelet fix arborescent.

În secțiunea a șasea a capitolului prezentăm apoi o implementare în Linux a sistemului propus realizată în spațiul utilizator. Motivul implementării în spațiul utilizator a fost unul practic, și anume ușurința dezvoltării și depanării sistemului, ceea ce ne-a dat posibilitatea să ne concentrăm cu precădere asupra chestiunilor nou introduse de sistem și nu neapărat asupra detaliilor de implementare. Implementarea se folosește de utilitarul FUSE pentru redirectarea apelurilor sistem din sistemul de operare în spațiul utilizator și de librăria SQLite pentru gestionarea bazei de metadate asociate fișierelor. Tot în această secțiune descriem, de asemenea, sintaxa limbajului de interogare pus la dispoziție aplicațiilor utilizator, expresiile ce descriu interogările fiind transmise sistemului nostru prin intermediul parametrului apelului sistem *chdir*. Trebuie remarcat faptul că prin interogare, este modul nativ de interacțiune cu sistemul nostru, acest lucru fiind valabil atât în cazul căutărilor fișierelor, cât și în cazul navigării prin arborele corespunzător unui set de fișiere, caz în care o interogare curentă este extinsă cu constrângeri corespunzătoare subdirectorului în care se coboară. Spre deosebire însă de simpla navigare, care permite găsirea doar a fișierelor accesibile pe căile vizibile în cadrul arborelui de fișiere, interogarea explicită (și având o formă mai complexă) permite selecția unor fișiere ce nu pot fi localizate prin navigare

într-un singur loc. Exemplul de mai jos ilustrează cele două tipuri de interogare ('&' corespunde operației logice ȘI, '|' operației SAU, iar '!' operației NOT).

```
> cd Books/Classification
> cd 'Author=Author1&Year=2007'

> cd Books/Classification
> cd 'Author=Author1&Year=2007|Year=2008'

> cd Books/Classification
> cd 'Author=Author1!Year=2007'
```

În secțiunea a șaptea a capitolului prezentăm rezultatele unor teste de performanță efectuate asupra sistemului implementat. Am comparat performanța sistemului propus cu cea a sistemului tradițional *Ext2* din Linux, în ceea ce privește operațiile de creare a noi fișiere și directoare, de citire a conținutului unui director și, respectiv, de găsire a unui set de fișiere. Rezultatele testelor efectuate au evidențiat încărcarea suplimentară introdusă de sistemul nostru, comparativ cu un sistem de fișiere tradițional, încărcare datorată asocierii de metadate unui element nou introdus în sistem. Totuși această încărcarea suplimentară nu este foarte mare în cazul introducerii unui număr mic de fișiere (în medie de aproximativ 25–30%). Testele au indicat de asemenea, posibile probleme de scalabilitate ale modulului de gestionare a bazei de date, lucru ce ar trebui rezolvat în implementările viitoare. Pe de altă, parte în cazul căutării de fișiere, sistemul nostru se dovedește net superior unui sistemului tradițional (cu aproape 100% mai performant doar în cazul unui set mic de proprietăți ale fișierelor) prin multitudinea de vederi pe care le oferă asupra spațiului de fișiere și a modurilor multiple prin care fișierele pot fi găsite, lucru pe care sistemul tradițional nu-l poate oferi deloc sau cel mult într-o formă limitată, dar la un cost ce-l face total ineficient.

Idei și părți din acest capitol au fost publicate în [9], [5], [10]. De asemenea, activitatea de cercetare desfășurată a fost motivată și susținută parțial prin grantul de cercetare CNCSIS tip A nr. 24/2007–2008, finanțat de Ministerul Educației și Cercetării din România.

5 Concluzii

Principalul obiectiv al lucrării a fost acela de găsire a unor tehnici care să asigure disponibilitatea permanentă a datelor și implicit a serviciilor ce oferă accesul la datele respective.

În acest sens, lucrarea oferă în primul rând o viziune de ansamblu asupra acestei probleme, în special prin intermediul analizei făcute în Capitolul 2. În cadrul aceluiași capitol prezintă, de asemenea, o arhitectură a unui sistem de gestionare și acces la date, numit DIPED, în cadrul căruia pot fi integrate mai multe tehnici ce contribuie fiecare în parte la atingerea obiectivului final. În cadrul celorlalte două capitole au fost abordate cazuri punctuale ale unor probleme legate de disponibilitatea datelor.

În Capitolul 3 a fost abordată problema asigurării stabilității unui serviciu cu implementare centralizată, într-un mod independent de serviciu și de asemenea transparent atât pentru serviciu, cât și pentru clienții săi. Tehnica folosită a fost cea de virtualizare a serviciului, adică de rulare a lui în cadrul unei mașini virtuale executate la rândul ei pe un nod principal și replicarea mașinii respective la nivelul mai multor noduri secundare. Au fost descrise și analizate două strategii de replicare: sincronă și asincronă. S-a dezvoltat, de asemenea, un protocol de replicare adaptiv, care să încerce să reducă întârzierile introduse de protocolul de replicare la nivelul timpului de răspuns la cererile clienților. În acest scop, protocolul ține cont atât de comportamentul curent al mașinii replicate (indirect al serviciului rulat în acea mașină) și de capacitatea curentă de transfer a legăturilor dintre nodul principal și cele secundare.

Capitolul 4 propune și descrie detaliat o posibilă soluție de realizare a serviciului de gestionare a metadatelor introdus de sistemul DIPED. Soluția integrează modelul de date relațional la nivelul unui sistem de fișiere, definind noi concepte de organizare a fișierelor și de căutare și selecție a lor. Conceptele introduse se bazează în principiu pe atașarea de proprietăți fișierelor, proprietăți prin care se stabilesc relații între fișiere. Interacțiunea cu noul sistem de fișiere se face prin intermediul interogărilor, dar în același timp, rezultatele interogărilor pot fi filtrare succesiv prin navigarea în adâncime în

arborele ce corespunde rezultatului interogării.

Lucrarea prezintă, în cadrul fiecărui capitol, implementări ale tehnicilor și sistemelor propuse. Astfel, am realizat o implementare în Grid a sistemului DIPED, o implementare peste Xen a sistemului de asigurare a stabilității unui serviciu prin tehnica de virtualizare și replicare asincronă, o implementare în Linux a sistemului de actualizare dinamică a modulelor din nucleu și o implementare în Linux a sistemului de fișiere bazat pe relații între fișiere.

5.1 Contribuții

Descriem mai jos, detaliat lista contribuțiile aduse în cadrul acestei lucrări, corespunzător fiecărui capitol.

Principalele contribuții aduse în Capitolul 2 sunt:

1. sintetizarea problemelor ce apar în dezvoltarea sistemelor de acces la date ce trebuie să asigure o disponibilitate permanentă a datelor și a posibilelor soluții la aceste probleme;

2. arhitectura detaliată a sistemului DIPED, arhitectură ce oferă posibilitatea integrării a diferite metode de gestionare și transfer eficient al datelor, metode de care s-a ținut cont și care au fost schițate, cum ar fi: organizarea fișierelor sub forma unei baze de date relaționale, regăsirea datelor prin combinarea metodelor de căutare și navigare, replicarea datelor prin combinarea metodelor de trimitere și aducere a datelor; transfer adaptiv la starea actuală a nodurilor și a legăturilor dintre noduri; unele dintre aceste metode vor fi detaliate în celelalte capitole ale lucrării;

3. identificarea și clarificarea posibilităților de interacțiune dintre și cu servicii implementate ca grupuri de procese;

4. mecanismul de tratare uniformă a transferurilor de date la nivelul întregului sistem, atât pentru replicarea propriu-zisă a datelor, cât și pentru furnizarea lor clienților, ambele cazuri fiind gestionate de serviciile de replicare, lucru care face posibile tehnici precum: (1) controlul

distribuirii echilibrate a solicitărilor adresate sistemului și adaptarea la condițiile actuale din sistem, (2) obținerea datelor atât de pe nodurile de replicare dedicate ale sistemului, cât și de pe cele ale clienților etc.;

5. propunerea de clienți neimplicați și implicați și detalierea arhitecturii lor;

6. structura de replicare scalabilă bazată pe șabloane de replicare și organizarea nodurilor de replicare într-o ierarhie de grupuri de replicare; demonstrarea faptului că structura respectivă asigură distribuirea datelor la toate nodurile la care ele trebuie să ajungă;

7. implementarea și testarea unui prototip de test al sistemului DIPED în Grid, folosindu-ne de serviciile de date oferite de infrastructura Globus Toolkit 4, dar extinzând și îmbunătățind funcționalitatea lor;

Principalele contribuții aduse în Capitolul 3 sunt:

1. Realizarea unei analize sistematice a problemei abordate și identificarea singurei metode prin care se pot atinge obiectivele propuse. Metoda constă în rularea serviciului într-o mașină virtuală și replicarea mașinii virtuale. Metoda este bazată pe replicarea pasivă, cu un server principal activ și restul secundare inactive, este generală, deoarece funcționează pentru orice serviciu și sistem de operare, funcționează și în cazul serviciilor cu comportament non-determinist și este, de asemenea, absolut transparent față de serviciul protejat.

2. Am propus, de asemenea, o tehnică de distribuire echilibrată a cererilor aplicabilă în cazul sistemului de protecție bazat pe virtualizare și replicare. Deși aceasta funcționează doar pentru o categorie particulară de servicii, am identificat-o ca fiind sigura posibilitate de a asigura echilibrarea serverelor replicate ale serviciului protejat.

3. Am dezvoltat o tehnică de replicare asincronă a mașinii virtuale și am analizat detaliat performanțele ei comparativ cu a celei sincrone. Am stabilit în acest sens formulele de calcul a întârzierii introduse de

protocol în cazul de funcționare normală și în cel de cădere a serverului principal.

4. Am dezvoltat un algoritm de replicare adaptivă a mașinii virtuale, având ca obiectiv reducerea cât mai mult posibil a întârzierilor introduse în timpul de răspuns la cererile clienților.

5. Am dezvoltat o tehnică de modificare sau actualizare a implementării modulelor Linux în timpul funcționării sistemului de operare. Tehnica se bazează pe separarea unui modul în interfață și implementare și prin indirectarea apelurilor adresate interfeței către implementarea curentă. Ea introduce o încărcare suplimentară a mecanismului de acces la module, dar aceasta este acceptabilă în majoritatea cazurilor.

Principalele contribuții aduse în Capitolul 4 sunt:

1. Adaptarea modelului relațional la sistemele de fișiere;

2. Introducerea conceptelor de *categorie* și *vederi multiple*, care permit o organizare mult mai flexibilă a spațiului de fișiere, comparativ cu sistemele tradiționale și posibilitatea căutării și navigării pentru rafinarea rezultatului.

3. Tratarea uniformă a categoriilor și fișierelor, ambele putând fi clasificate pe baza proprietăților asociate. Se obține astfel un schelet fix de organizare, cu ramuri având forme flexibile și multiple de organizare.

4. Integrarea transparentă a sistemului propus în sistemul de operare Linux și păstrarea compatibilității cu aplicațiile existente.

5. Dezvoltarea unei metode de utilizare a unora dintre facilitățile sistemului de către aplicațiile existente.

5.2 Direcții de dezvoltare

În primul rând, direcții noi de dezvoltare se deschid relativ la sistemul DIPED. Acesta a și fost, de altfel gândit și proiectat astfel încât să permită integrarea

a diferite tehnici. Dintre acestea doar unele au fost dezvoltate și integrate efectiv în sistemul nostru. Avem, în acest sens în stadiu de dezvoltare protocolul de replicare a datelor schițat în Capitolul 2 și care va folosi combinat tehnicile de trimitere a datelor, respectiv de aducere a lor simultan de la mai multe surse.

La nivelul sistemului de realizare a stabilității unui serviciu prin replicare și virtualizare, posibile dezvoltări ulterioare ar fi:

- implementarea protocolului adaptiv și a tehnicilor de optimizare propuse, în special a celei de reducere a timpului de salvare prin tehnica "copy-on-write" aplicată asupra paginilor mașinii virtuale;
- dezvoltarea unui protocol de replicare a dispozitivelor de stocare ale mașinii virtuale și integrarea lui cu protocolul de replicare al mașinii virtuale;
- plasarea componentelor replicate ale serviciului în cadrul unei rețele extinse și folosirea unui protocolului de replicare menționat mai sus;

În privința sistemului de actualizare dinamică a modulelor Linux, există următoarele posibile dezvoltări:

- optimizarea tehnicii de actualizare a modulelor;
- dezvoltarea unui mediu care să automatizeze pe cât posibil procesul de translatare a unui modul obișnuit într-unul actualizabil și să ofere sprijin dezvoltatorului în acest sens; avem deja în dezvoltare un astfel de mediu;
- aplicarea tehnicii de actualizare în cazul unui modul kernel de gestionare a bazelor de date, modul care să poată fi folosit de sistemul de fișiere propus în Capitolul 4; scopul ar fi acela de modificare dinamică a modulului respectiv pentru a putea testa comparativ funcționalitatea sistemului cu diferite implementări ale modulului respectiv;

Sistemul de fișiere introdus în Capitolul 4 suportă, de asemenea, o serie de dezvoltări ulterioare, dintre care menționăm:

- introducerea suportului pentru crearea de traductoare și utilizarea lor pentru extragerea automată a proprietăților fișierelor de anumite tipuri;
- implementarea tuturor tipurilor de proprietăți propuse și a relațiilor dintre fișiere;
- suport pentru definirea unor categorii tip, cu seturi de proprietăți predefinite, care să ușureze munca utilizatorului;
- proiectarea și implementarea unei interfețe cu apeluri sistem specializate, care să facă mai eficientă interacțiunea aplicațiilor utilizator cu sistemul;
- proiectarea și implementarea unei aplicații specializate de explorare a spațiului de fișiere; introducerea de noi concepte, care să faciliteze acest lucru;
- integrarea în nucleul sistemului de operare;

Lista de publicații

[1] A. Coleșa, "Probleme de fiabilitate, scalabilitate şi disponibilitate a serviciilor în sistemele grid," in *Volumul 1 al Atelierului de lucru MedioGRID*, Cluj-Napoca, Romania, Decembrie 2005, pp. 169–183. 12

[2] A. Coleșa and I. Ignat, "Fiabilitatea şi scalabilitatea serviciilor de replicare a datelor în sistemele grid," in *Volumul 2 al Atelierului de lucru MedioGRID*, Cluj-Napoca, Romania, February 2006, pp. 160–180. 12

[3] A. Colesa, I. Ignat, and R. Opris, "Providing high data availability in mediogrid," in *Proceedings of The 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '06)*. IEEE Computer Society, 2006, pp. 296–302. 12

[4] A. Colesa, T. Pop, I. Ignat, and C. Ardelean, "Automatic and reliable distribution of data in grids over globus toolkit," in *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for*

Scientific Computing (SYNASC07). IEEE Computer Society, 2007, pp. 310–316. 12

[5] A. Coleșa, V. Cionca, A. Tața, and I. Ignat, "A meta-data enhanced file system," in *Proceedings of the 3rd IEEE International Conference on Intelligent Computer Communication and Processing (ICCP07)*. IEEE, September 2007, pp. 267–270. 12, 20

[6] A. Colesa, B. Marincas, I. Ignat, and C. Ardelean, "Strategies to transparently make a centralized service highly-available," in *Proceedings of the 5rd IEEE International Conference on Intelligent Computer Communication and Processing (ICCP09)*. IEEE Computer Society, 2009, pp. 296–302. 12, 17

[7] Y. Balde, A. Coleșa, and I. Ignat, "An indirect hotswapping system for linux kernel modules," in *Proceedings of the 3rd IEEE International Conference on Intelligent Computer Communication and Processing (ICCP07)*. IEEE, September 2007, pp. 270–276. 17

[8] A. Coleșa, I. Stan, and I. Ignat, "Transparent fault-tolerance based on asynchronous virtual machine replication," in *Proceedings of The 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '10)*. IEEE Computer Society, 2010, to appear. 17

[9] A. Coleșa, I. Ignat, Z. Majo, and V. Cionca, "A meta-data enhanced file system using the classical interface," in *Proceedings of The 10th International Conference an Applied Mathematics and Computer Science*, May 2006, pp. 100–106. 20

[10] A. Coldea, A. Coleșa, and I. Ignat, "Orcfs: Organized relationships between components of the file system for efficient file retrival," in *Proceedings of The 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '10)*. IEEE Computer Society, 2010, to appear. 20