

CAPITOLUL 7

FIABILITATEA ECHIPAMENTELOR DE CALCUL

7.1. Introducere

În acest capitol se prezintă arhitectura echipamentelor de calcul dintr-un singur punct de vedere, și anume cel al fiabilității. Prezentarea nu va fi exhaustivă și nici matematizată; se vor folosi mai ales exemple pentru a ilustra cum felurite considerații despre fiabilitate influențează structura echipamentelor de calcul.

Principalul subiect al teoriei fiabilității constă în construirea de echipamente cât mai fiabile din componente nefiabile. Dacă un echipament ar funcționa numai atunci când toate componentele sale ar fi funcționale, ar fi virtual imposibil de construit un echipament complex, pentru că fiabilitatea ar descrește exponențial cu numărul de componente din structura sa.

Principală unealtă folosită în construirea echipamentelor complexe este **abstracția**. Un echipament este construit pe nivele: nivelul B este alcătuit din componente de nivel A. La rândul lor, componente de nivel B sunt folosite ca și cum ar fi atomice, indivizibile, pentru a construi nivelul C, și așa mai departe. Acest proces este inspirat din matematică, unde lemele și teoremele sunt folosite drept componente elementare în demonstrațiile altor leme și teoreme. În acest capitol se prezintă alcătuirea unor nivele din arhitectura echipamentelor de calcul din punctul de vedere al fiabilității pe care o oferă nivelelor superioare. Astfel putem distinge:

- Nivele care **măresc fiabilitatea**, construind un echipament mai fiabil din componente mai puțin fiabile. Acest lucru este obținut folosind **redundanță** în stocarea sau calculul informației. Acest tip de nivel este cel mai adesea folosit în construcția calculatoarelor contemporane.
- Nivele care **expun lipsa de fiabilitate** nivelelor superioare, lăsându-le pe acestea să rezolve imperfecțiunile. Nivelele superioare au adesea informații suplimentare despre cerințele reale de fiabilitate ale echipamentului și ca atare pot construi fiabilitatea pe măsura necesităților.

- Anumite nivele **partiționează** resursele în părți oarecum independente, izolate una de alta. Partiționarea are drept efect **izolarea defectelor** (fault isolation), astfel încât o defecțiune într-o parte să nu afecteze celelalte părți. În calculatoare această tehnică este folosită în sistemele de operare și clustere-le de calculatoare.

La ora actuală circuitele integrate pe scară largă (Very Large Scale Integrated circuits, VLSI) au ajuns la nivele incredibile de fiabilitate. Ca atare arhitecții calculatoarelor privesc în general nivelul hardware ca fiind "perfect" și folosesc această abstracție foarte convenabilă în proiectarea nivelelor superioare. Anumite clase de aplicații au nevoie însă de o fiabilitate foarte ridicată (de exemplu, controlul de trafic aerian, supervizarea centralelor nucleare sau a echipamentelor militare). În astfel de misiuni critice proiectanții echipamentelor de calcul iau în considerare și posibilitatea defectelor hardware, pe care le tratează în software.

Miniaturizarea continuă a circuitelor integrate va conduce la schimbări în această stare de fapt, astfel încât trebuie să ne așteptăm ca în viitor circuitele să conțină din ce în ce mai multe defecțiuni și să fie din ce în ce mai sensibile la fluctuații termodinamice și particule de înaltă energie din radiația cosmică sau chiar din degradarea radioactivă a circuitului integrat respectiv. Astfel de schimbări vor necesita o reproiectare completă a arhitecturii echipamentelor de calcul.

Ingredientul cel mai folosit pentru a construi echipamente fiabile este *redundanța*. În cazul echipamentelor de calcul putem distinge două genuri de redundanță, spațială și temporală:

- **Redundanța spațială** folosește mai multe componente decât strictul necesar pentru a implementa un anumit echipament de calcul. Resursele adiționale fac calcule suplimentare și rezultatele sunt comparate între ele. În general, cu cât redundanța unui echipament de calcul este mai mare, cu atât poate detecta sau tolera mai multe erori.
- **Redundanța temporală** constă în folosirea aceluiași dispozitiv pentru a calcula același lucru în mod repetat, după care rezultatele sunt comparate între ele.

Defectele ce afectează componenta soft a unui echipament de calcul se pot clasifica în două mari categorii:

- **Erori tranzitorii**, care se manifestă printr-o funcționare temporară eronată a unei componente, dar nu prin defectarea ei definitivă. În echipamentele de calcul contemporane, cea mai mare parte a erorilor sunt tranzitorii.

- **Erori permanente** care se produc la un moment dat și persistă până când echipamentul este reparat. În această categorie includem și defectele din faza de proiectare sau din fabricație.

Rezultă că redundanța temporală poate fi folosită numai pentru a tolera defectele tranzitorii. Pentru a tolera efecte permanente trebuie să avem o formă de redundanță spațială.

7.2. Proiectarea echipamentelor de calcul

Când proiectăm un echipament complex este foarte important să ”echilibrăm” fiabilitatea părților. De exemplu, dacă memoria unui echipament de calcul are o fiabilitate mult mai mare decât procesorul, atunci echipamentul se va defecta cel mai adesea cu probleme de procesor. Faptul că memoria este de foarte bună calitate nu ne ajută cu nimic, dimpotrivă probabil că am plătit un preț mai mare pentru memorie decât ar fi fost strict necesar. În general, o componentă este ”destul de bună” dacă nu are cea mai mare probabilitate de defectare.

În cazul analizei fiabilității unui echipament trebuie să socotim nu numai costul componentelor fiabile, ci și costul întreținerii echipamentului în timpul misiunii sale (vezi figura 1.2). Dacă utilizăm componente foarte fiabile plătim prea mult pentru construcția echipamentului, iar dacă utilizăm componente cu fiabilitate prea redusă, ne va costa prea mult întreținerea echipamentului. Numai contextul poate dicta cât de fiabil trebuie să fie un echipament: de exemplu, în aplicațiile critice descrise mai sus, costul ne-funcționării echipamentului este uriaș, așa încât are sens să investim în componente extrem de fiabile.

Evitarea defectelor echipamentelor de calcul este o metodologie idealizată, care presupune că toate componentele sunt perfecte. Pentru că hardware-ul actual are o calitate excepțională, nivelul software în calculatoarele obișnuite adoptă o astfel de viziune idealizată. Programatorii presupun că echipamentul pe care se rulează programele lor este lipsit de defecțiuni.

Fiabilitatea excelentă a dispozitivelor hardware este obținută printr-o combinație de tehnici, cum ar fi felurite forme de redundanță, proiectare și fabricație cu precizie foarte ridicată, și o fază agresivă de testare și ”ardere” (burn-in).

Empiric s-a observat că echipamentele de calcul tind să aibă o mortalitate care urmărește o curbă în formă de cadă de baie, echivalentă ratei

mortalității din studiile demografice. Echipamentele foarte tinere și cele foarte uzate se strică mult mai des decât echipamentele ”mature”. ”Burn-in” este o fază de testare care folosește componentele până devin mature și în acest fel, componentele cu mortalitate infantilă ridicată sunt eliminate în această etapă.

În plus fabricanții proiectează și testează echipamentele de calcul în condiții mai nefavorabile decât cele specificate. De exemplu, pe acest fapt se bazează cei care fac ”overclocking”: specificațiile unui procesor indică frecvența de ceas la care acesta poate opera, însă în mod frecvent un procesor cu specificație de ceas de 1Ghz poate opera la 1.2Ghz, datorită marginilor de toleranță din fabricație.

Metoda evitării defectelor este cu adevărat extremă. Întrucât s-a constatat că oricare dintre componente se poate defecta, se utilizează proceduri de menținere a echipamentului în funcționare prin implementarea ”toleranței la defectare” (fault-tolerance).

7.2.1. Structuri tolerante la defectări

O metodă foarte simplă, dar scumpă, de a tolera erori este de a multiplica fiecare componentă. De exemplu, dacă duplicăm întreg echipamentul de calcul, apariția unui defect poate fi detectată comparând rezultatele celor două echipamente cu structură identică.

O altă modalitate de a tolera defectele constă în utilizarea structurilor TMR (§ 4.3.1.1). În această structură trei module fac aceeași operație și un modul de decizie (”voter”) alege rezultatul majoritar. Structura va avea performanțe net superioare unui singur modul, însă cu un preț considerabil mai ridicat. Întrucât voterul devine elementul ”slab” al structurii, există și scheme în care echipamentul de votare este multiplicat, pentru ca decizia să nu depindă de o singură componentă.

Un astfel de echipament de votare este folosit în calculatoarele care controlează navele spațiale: echipamentul este compus din cinci calculatoare, din care patru fac aceleași calcule și al cincilea este folosit pentru operațiuni ne-critice. Rezultatele celor patru calculatoare se duc până la echipamentele controlate (*exemplu*: motoare de propulsie), care calculează local rezultatul votului. În plus, fiecare calculator compară rezultatele cu celelalte trei, iar atunci când unul dintre ele dă rezultate diferite este scos din funcțiune.

Dacă două calculatoare se defectează, echipamentul intră într-un mod de funcționare în care rezultatele sunt comparate și recalulate atunci când diferă. Al cincilea calculator conține un echipament de control complet separat, dezvoltat de altă companie, care intră în funcțiune când un bug identic este detectat în celelalte patru programe.

În continuare vor fi prezentate o serie de structuri de echipamente de calcul tolerante la defectări:

A) Procesorul IBM G5 din echipamentul S/390. În cazul acestui procesor se folosește un tip de redundanță hibridă, care utilizează redundanța spațială pentru a detecta erori tranzitorii și redundanța temporală pentru a le remedia. Acest echipament este asemănător cu modul de funcționare cu două defecțiuni folosit de naveta spațială, descris mai înainte.

Microprocesorul G5 conține două benzi de execuție identice, care sunt controlate de același ceas. Toate instrucțiunile sunt executate în mod sincron de ambele benzi, iar la sfârșitul execuției rezultatele sunt comparate. Dacă rezultatele sunt identice, rezultatul instrucțiunii este scris în registrul destinație sau în memorie. Dacă nu, se generează o excepție software, care de obicei se soldează cu re-execuția instrucțiunii-problemă. Erorile tranzitorii sunt astfel reparate în mod transparent. Această schemă este funcțională pentru că probabilitatea ca o eroare tranzientă să afecteze ambele benzi în același mod este una foarte mică.

B) Procesor superscalar tolerant la erori tranzitorii. O schemă foarte originală care folosește doar redundanță temporală pentru a tolera erori tranzitorii a fost propusă în anul 2001 la conferința de microarhitectură MICRO 2001 de un grup de cercetători de la universitatea Carnegie Mellon. În această schemă unui procesor superscalar obișnuit i se fac câteva modificări simple, astfel încât fiecare instrucțiune citită să fie lansată în execuție în mod repetat. Metodele de redenumire a regiștrilor folosite în procesorul superscalar fac din executarea unor instrucțiuni suplimentare, care nu afectează echipamentul, un lucru foarte simplu. La sfârșitul benzii de asamblare rezultatele copiilor lansate în execuție sunt comparate între ele. Robustețea depinde de gradul de redundanță: dacă fiecare instrucțiune este executată de două ori, o eroare se manifestă prin rezultate diferite și instrucțiunea trebuie re-executată; dacă o instrucțiune este executată de mai mult de două ori, se poate folosi o schemă de votare cu majoritate.

Un astfel de procesor poate fi proiectat să lucreze fie în mod normal, fie în mod cu fiabilitate crescută, depinzând de tipul de program executat. Performanța în modul cu fiabilitate ridicată este invers proporțională cu

gradul de redundanță; de exemplu, dacă fiecare instrucțiune este executată de două ori, ar putea rezulta o scădere a vitezei de calcul la 50%. În realitate, penalizarea este ceva mai mică, din cauză că un program nu folosește toate resursele computaționale. De exemplu, dacă un program folosește 80% din resurse, când executăm programul duplicând fiecare instrucțiune avem nevoie de 160% resurse, ceea ce se traduce într-o degradare a performanței cu 37,5% ($100/160 = 62,5 = 100 - 37,5$).

C) Echipament de calcul tolerant la defectări având arhitectură cu verificare dinamică. Această structură, propusă în anul 1999 de către cercetătorii de la universitatea Michigan este cunoscută sub numele de **DIVA**, de la **D**ynamic **I**mplementation **V**erification **A**rchitecture (arhitectură cu verificare dinamică).

Spre deosebire de schemele anterioare, structura DIVA e proiectată pentru a tolera atât erori tranzitorii, cât și permanente (cele din urmă doar în anumite părți ale echipamentului). Observația centrală pe care se bazează DIVA rezidă din faptul că este că e mai ușor de verificat dacă rezultatul unui calcul e corect decât este de efectuat calculul însuși. Ca atare, arhitectura DIVA este compusă din două procesoare diferite, fiind prezentată în figura 7.1.

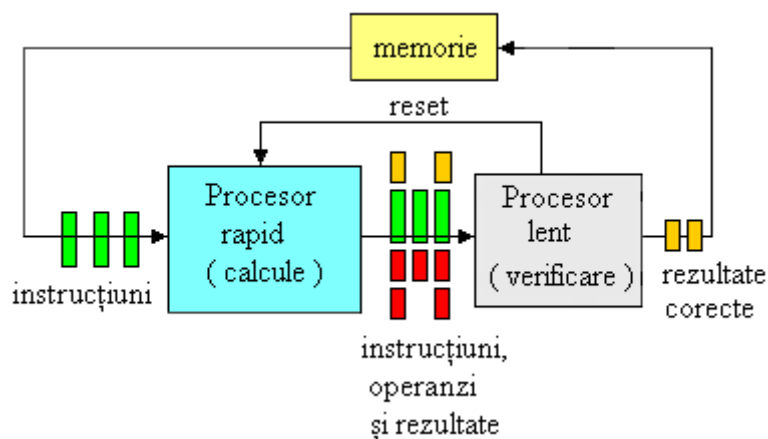


Fig.7.1. Structura echipamentului de calcul DIVA

Această structură conține:

- Un procesor complex, superscalar, foarte optimizat, care face calculele în mod normal.
- Un procesor extrem de simplu, mai lent, dar foarte fiabil, care execută instrucțiunile în ordine, și este construit folosind tehnici de evitare a defectelor.

Echipamentul de calcul DIVA funcționează astfel:

- Procesorul complex execută toate instrucțiunile și calculează rezultatele lor. Rezultatele însă nu sunt scrise, ci sunt transmise procesorului simplu (lent și fiabil).
- Procesorul simplu merge ceva mai încet, și verifică în paralel toate detaliile rezultatelor primite. Deși acest procesor este mai simplu, are o treabă mai ușoară, și ca atare poate atinge aceeași performanță ca cel rapid (exprimată în instrucțiuni procesate pe secundă). Când verificarea descoperă o eroare, procesorul simplu calculează rezultatul corect și re-pornește procesorul complex de la instrucțiunea următoare.

Foarte interesant este faptul că o arhitectură DIVA poate tolera chiar erori de proiectare în procesorul foarte complicat, pentru că acestea sunt detectate și corectate de procesorul lent și simplu. Poate fi chiar avantajos ca procesorul rapid să fie proiectat ”incorect”, dar extrem de rapid, în cazul în care nu produce rezultate eronate prea frecvent. De exemplu, într-un procesor normal foarte multe circuite suplimentare sunt introduse pentru a trata corect cazul programelor care se auto-modifică. În realitate, practic nici un program modern nu folosește această tehnică în mod curent; procesorul rapid fără aceste circuite poate fi făcut mult mai eficient, iar corectitudinea calculelor va fi asigurată de către procesorul lent.

7.2.2. Coduri detectoare și corectoare de erori

S-au prezentat deja mai multe exemple de folosire a redundanței spațiale pentru detectarea și corectarea erorilor. Costul schemelor prezentate mai înainte este substanțial: ele cer o multiplicare identică a unui întreg echipament. De exemplu, redundanța modulară triplă are o eficiență de 33%, pentru că hardware-ul este multiplicat de trei ori.

E interesant de explorat dacă nu putem obține aceleași beneficii cheltuind mai puține resurse suplimentare. Deschizători de drumuri au fost în această privință Claude Shannon și Richard Hamming, spre sfârșitul anilor '40. În continuare se prezintă metodele propuse de ei pentru a stoca informație utilizând structuri tolerante la defectări.

Să presupunem că dorim să stocăm niște informații codificate în baza doi într-un mod fiabil. Putem atunci de pildă face două copii ale informației. Dar o defecțiune a unui singur bit va face informația de nerecuperat, pentru că acel bit va fi diferit în cele două copii, și nu putem deduce care este

valoarea originală. Slăbiciunea acestei metode constă în faptul că biții stocați nu sunt ”robuști”: fiecare bit din mesaj este reprezentat în doar doi biți din cod.

Pentru a obține toleranță la erori trebuie să adăugăm redundanță în cod; astfel, vom codifica n biți de informație folosind $m > n$ biți de cod. Cu cât m e mai mare ca n , cu atât mai robust va fi codul nostru. Cuvintele de m biți care reprezintă coduri corecte se numesc ”cuvinte de cod” (code words). Se observă că nu toate cuvintele de m biți sunt cuvinte de cod, ci numai 2^n dintre ele.

Se poate defini **distanța Hamming** între două șiruri de biți ca fiind numărul de diferențe între cele două șiruri. De exemplu, distanța Hamming dintre 1111 și 1010 este 2, pentru că cele două șiruri diferă în pozițiile a doua și a patra. Cea mai mică distanță Hamming dintre cuvintele unui cod este o măsură foarte bună a robusteții codului. De exemplu, dacă distanță Hamming între oricare două cuvinte este mai mare decât 3, atunci o schimbare de 1 bit poate fi întotdeauna corectată: cel mai apropiat cuvânt de cod este cel care a fost modificat de eroare, pentru că toate celelalte cuvinte de cod se vor afla la o distanță mai mare de 2 de cuvântul eronat. Astfel, un cod cu distanță Hamming 3 poate corecta orice eroare de 1 bit, și poate detecta orice eroare de doi biți. Un astfel de cod va detecta și alte erori, de exemplu va detecta unele erori de trei biți, dar nu orice eroare de trei biți. Există efectiv zeci de coduri diferite, fiecare potrivit în alte circumstanțe.

Codurile detectoare și corectoare de erori sunt folosite pe larg în rețelele de calculatoare. Depinzând de caracteristicile canalului de comunicații (distanță, cost de transmisiune, viteza semnalului, zgomot) se pot folosi coduri mai mult sau mai puțin robuste. În anumite cazuri e preferabil ca erorile să fie detectate și datele incorecte să fie retransmise, în alte cazuri costul retransmisiei este prea mare, și ca atare se folosesc coduri corectoare. Folosirea unui cod corector în transmisiunea de date se mai numește și codare preventivă (Forward Error Correction).

Pentru comunicația cu sondele spațiale se folosesc coduri corectoare de erori extrem de robuste, pentru că la astfel de distanțe semnalul electromagnetic are nevoie de multe minute pentru a se propaga. În 1993 un grup de cercetători francezi a inventat o clasă de coduri extrem de robuste numite Turbo-coduri care, folosind o redundanță relativ redusă de 200%, obțin o reziliență excepțională la zgomot, fiind foarte aproape de limitele maxime teoretice.

Turbo-codurile ilustrează un nou tip de compromis pe care proiectantul îl poate face în relația robustețe/cost: costul cel mare al unui turbo-cod nu este în cantitatea mare de informație suplimentară, ci în algoritmul de decodificare, care este foarte complicat și necesită multe iterații. După cum am văzut și în cazul memoriilor, cu aceeași redundanță putem obține garanții diferite de fiabilitate, în funcție de algoritmul de codificare folosit. În cazul comunicației interplanetare costul transmisiunii face costul decodificării insignifiant, deci turbo-codurile sunt potrivite.

7.2.3. Memorii tolerante la defectări

În funcție de modalitățile utilizate în scopul protecției împotriva erorilor, memoriile în structura unui echipament de calcul se pot clasifica astfel:

Memoriile neprotejate. Aceste memorii stochează fiecare bit de date în mod separat și nu oferă nici o protecție împotriva erorilor. Ca atare sunt cele mai ieftine. Cum însă dimensiunea memoriilor a crescut foarte repede, la ora actuală această soluție este riscantă, căci probabilitatea ca nici un bit să nu se defecteze este foarte redusă.

Memoriile cu paritate. Aceste memorii folosesc o metodă foarte simplă pentru a detecta erori de un bit în fiecare octet (și, în general, erori care schimbă un număr impar de biți). Pentru fiecare 8 biți de date aceste memorii stochează un al nouălea bit de paritate, a cărui valoare este calculată astfel încât oricare cuvânt de nouă biți are un număr par de biți "1" (de aici și numele schemei).

Când hardware-ul accesează memoria, automat verifică și paritatea. Dacă paritatea nu este corectă se declanșează o excepție și echipamentul de operare decide cum trebuie să acționeze. O soluție este de a opri programul care folosea acea memorie și de a marca memoria ca fiind defectă, astfel încât alte programe să nu o poată refolosi. Verificarea parității este o operație foarte rapidă, care se poate face foarte simplu folosind structuri hardware, în paralel cu transferul informației.

Memoriile ECC. Acestea sunt protejate cu un cod sofisticat de corecție a erorilor (Error Correcting Code). Acest cod poate corecta automat orice eroare de 1 bit care apare într-un cuvânt de 64 de biți. Pentru acest scop memoria stochează fiecare cuvânt de 64 de biți folosind cuvinte de cod de 72 de biți. Se observă că "risipa" (overhead) acestei scheme este aceeași cu cea a parității ($9/8 = 72/64$). Această schemă oferă corecție cu o robustețe

mai mică, pentru că poate corecta o eroare la 64 de biți, spre deosebire de cealaltă schemă care poate detecta o eroare la 8 biți.

La fiecare acces la memorie hardware-ul verifică dacă cuvântul de cod este corect, iar dacă nu, calculează automat cel mai apropiat cuvânt de cod pe care apoi îl decodifică. Aceste operații sunt destul de complicate, astfel încât un echipament cu memorii ECC funcționează cu aproximativ 5% mai lent decât unul cu memorii ce utilizează principiul parității.

Discuri. Cel mai comun suport permanent de informație este discul, în multiplele lui implementări: hard-disc, dischetă, disc optic, compact-disc, flash-disk etc. Informațiile din această secțiune sunt valabile pentru multe dintre aceste tipuri de discuri.

Discurile folosesc simultan două metode diferite de redundanță spațială; o protecție sporită este necesară din cauză că discurile funcționează într-un mediu mult mai aspru decât memoriile: unele discuri au părți mecanice în mișcare, care se uzează și se pot strica mai ușor.

Informația este stocată pe discurile clasice în **sectoare**. Un sector este relativ mare (comparat cu un cuvânt de memorie), fiind de ordinul a jumătate de kilooctet (512 octeți). Discurile folosesc sectoare mari pentru că la viteza lor de rotație (peste 5000 de rotații pe minut) capetele de citire/scriere nu se pot plasa foarte precis pe suprafață. Astfel, unitatea elementară în care se scrie pe un disc este sectorul: chiar dacă vrem să modificăm un singur bit, trebuie să rescriem tot sectorul.

În figura 7.2 este prezentat formatul unui sector de disc. Informația servo este folosită pentru controlul mișcării capului, identificatorul indică numărul sectorului curent, iar informația de sincronizare este folosită pentru a sincroniza poziția capului cu începutul sectorului.

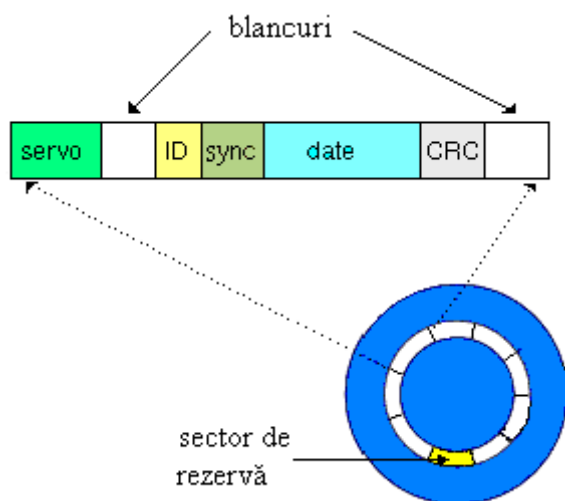


Fig. 7.2. Formatul unui sector de disc

Datele sunt stocate într-un șir compact și codificate folosind un cod detector de erori. Între două sectoare consecutive este un blank (spațiu liber), care-i dă capului ceva libertate când rescrie sectorul (niciodată nu va rescrie începând chiar din același loc).

Codurile folosite pentru discuri se numesc CRC, de la Cyclic Redundancy Check: *coduri ciclice*. Un cuvânt de cod constă din chiar cuvântul de date urmat de informații de control. Decodificarea codurilor CRC este foarte simplă: se extrage direct cuvântul de date, după care codul de control verifică dacă vreunul dintre biții stocați e incorect. Un cod ciclic are proprietatea că orice permutare a datelor este protejată de același cuvânt de control.

Când *codul de control* indică defectarea unui sector, discurile folosesc în mod automat a doua formă de redundanță spațială: sectoare de rezervă. Pe disc sunt ascunse sectoare invizibile, care sunt folosite atunci când sectoarele de date încep să dea rateuri. În mod transparent software-ul alocă un sector de rezervă în locul unuia defect. Identificatorul de sector este folosit pentru a indica cine pe cine înlocuiește.

Discurile stochează o **hartă de defecte** care indică sectoarele înlocuite, acest lucru permițând o funcționare corectă și după ce apar defecțiuni, și permite, de asemenea, un proces de fabricație mai imperfect și mai ieftin.

Pentru stocarea de înaltă fiabilitate a datelor se utilizează Flash Disk-uri. Aceste dispozitive au apărut din necesitatea ca în aplicațiile industriale sau de comunicații portabile, să fie nevoie de o capacitate de memorie de numai 80 MB sau mai puțin, fiind necesară și o foarte bună rezistență la șocuri și vibrații. În astfel de cazuri, soluția folosirii unor unități de hard-disk convenționale este improprie, fie și numai pentru motivul că astăzi nu se mai fabrică HDD-uri cu capacități așa de mici. Până acum singura soluție era utilizarea unui hard-disk cu o capacitate minimă de 3-4 GB. În pofida excedentului de memorie, se putea întâmpla ca acesta să se defecteze în mai puțin de o lună de funcționare efectivă, datorită mediului de lucru sever în care se desfășura misiunea. În mod normal, hard-diskurile au o fiabilitate ridicată în mediul de birou sau laborator, MTBF fiind de ordinul a multe zeci de mii de ore (echivalent a 8 ani de funcționare continuă sau 30 ani de utilizare normală). Însă acestea nu suportă lovituri sau șocuri puternice, care le pot scoate definitiv din funcțiune, iar dacă mediul de lucru prezintă vibrații, capetele se vor poziționa eronat pe pistă, generând erori soft (de citire) sau crescând considerabil timpul de acces (durează până când se repoziționează pe pista corectă).

O soluție parțială ar fi HDD-rile pentru notebook-uri, deoarece prezintă o rezistență mecanică considerabil sporită, fiind compacte și special proiectate pentru o utilizare în medii cu șocuri și vibrații. Însă capacitatea este tot excedentară, iar prețul este foarte ridicat.

Odată cu apariția Flash Disk-urilor, problema pare a fi soluționată. Ideea înlocuirii mediilor magnetice cu memorii semiconductoare nu este nouă, dar la început prețul era prohibitiv, iar capacitatea redusă. Au apărut apoi memoriile reprogramabile Flash. Inițial, acestea erau doar un fel de EEPROM-uri mai performante, nu puteau fi șterse decât de un număr limitat de ori (de circa 10.000). Desigur, această valoare este excelentă pentru o memorie programabilă, dar cu totul insuficientă pentru a înlocui un disk magnetic. Ulterior au fost perfecționate, iar acum rezistă la un milion de cicluri de scriere/citire. De asemenea, dacă la început erau disponibile doar sub forma de cartele de memorie, acum emulează perfect o unitate de disk IDE, astfel că oferă o soluție de stocare a datelor cu o mare fiabilitate hardware și software.

Absența totală a componentelor în mișcare face modulele de memorie Flash mai rapide și mai robuste decât mediile magnetice rotative, respectiv unitățile de hard-disk. Principalul dezavantaj îl reprezintă prețul mai ridicat, dar pentru capacități de stocare mici, ele reprezintă o alternativă convenabilă la HDD. Termenul ”mici” este ceva relativ: dacă la începutul anului 1993 însemna 1-2MB, acum se referă la 100-400MB. În momentul de față sunt disponibile și Flash Disk-uri de peste 500MB, dar odată cu sporirea capacității, prețul crește considerabil.

În continuare se prezintă o comparație între Flash Disk-uri și unitățile de hard-disk miniatură (de 1,3 sau 2,5 inch). La capacități mici, până la 100MB, modulele Flash sunt categoric competitive. De fapt, la aceste valori mici, principalul concurent al Flash Disk-urilor îl reprezintă memoria RAM statică, nu hard-diskurile. Prețul memoriei Flash este proporțional cu capacitatea, deoarece capacitatea de stocare este direct proporțională cu numărul de circuite integrate Flash folosite. Unitățile de disk magnetic au un preț de bază ce include prețul componentelor mecanice și al controlerului. Prețul hard-diskului nu poate scăde sub această valoare de pornire, dar, mărinind suprafața mediului magnetic sau numărul de fețe, se poate mări capacitatea cu o creștere de preț foarte mică.

În privința rezistenței la șocuri fizice, (cum ar fi simpla cădere a aparatului), modulele Flash sunt clar superioare hard-diskurilor. Gama de

temperaturi de lucru este similară la cele două categorii. Intervine însă, din nou, conceptul de fiabilitate software: în cazul unor variații rapide de temperatură, HDD-urile clasice nu sunt disponibile imediat. Când este adus de la o temperatură scăzută într-o încăpere încălzită, unitatea de hard-disk trebuie lăsată să se aclimatizeze, pentru a nu apare probleme cu condensarea umidității. Dacă temperatura ambiantă se modifică rapid și frecvent, sistemul de urmărire a pistelor pierde un timp considerabil cu operațiile de recalibrare. În aceste cazuri, disponibilitatea unității de disk este redusă. Flash Disk-urile nu prezintă asemenea probleme.

O comparație directă a vitezelor este greu de realizat. Scrierea efectivă este mai lentă la Flash, dar hard-diskul necesită un timp pentru atingerea vitezei de rotație de regim și prezintă întârzieri de rotație și în trecerea de la o pistă la alta.

Într-o aplicație tipică, memoria Flash este de 2-10 ori mai rapidă decât hard-diskul. "Aplicație tipică" este cea în care datele sunt citite de 4-5 ori mai des decât sunt scrise, iar transferurile de date tind să fie scurte și nu foarte frecvente. Debitul de date în cazul unui transfer continuu este mai redus: 0,7-1 Moctet pe secundă la citire și 2-300 Kocteți pe secundă la scriere, față de valoarea de 5 Mocteti pe secundă în mod rafală (burst).

Alte avantaje ale FlashDisk-urilor sunt nivelul sonor foarte redus în timpul funcționării (unele hard-diskuri pot fi destul de zgomotoase) și posibilitatea lucrului la altitudini foarte ridicate. Hard-diskurile au nevoie de o presiune atmosferică minimă pentru a realiza perna de aer pe care plutesc capetele.

Caracteristicile generale ale Flash Disk-urilor sunt prezentate în continuare:

- au dimensiunile clasice ale HDD-urilor obișnuite;
- sunt Disk-uri cu interfața IDE, bazate pe o tehnologie Flash;
- au capacități de stocare extrem de flexibile: între 4MB și 512 MB;
- sunt mult mai stabile decât HDD-urile convenționale. Ele pot rezista la șocuri și vibrații fiind astfel foarte potrivite pentru aplicații ce se desfășoară în medii cu cerințe severe;
- au un consum redus de energie;
- sunt 100% compatibile IDE, nefiind necesară instalarea vreunui driver;
- folosesc algoritmul de detecție și corecție al erorilor Reed Solomon pe 16 biti;
- cu o interfață de 3,3V sau 5V, Flash Disk-urile suportă gestiunea automată a consumului de energie (APM) precum și comenzi ATA de întrerupere a alimentării și mod "sleep" (ațipire).

Cel mai important lucru este că oferă o stabilitate mult mai bună a aplicațiilor, în medii de lucru dure, cu șocuri și vibrații. În astfel de medii, unitățile de hard-disk clasice chiar dacă nu se defectează, prezintă erori de căutare, astfel că răspunsul devine lent, iar comportarea - aleatoare. Consumul redus de energie contribuie la creșterea duratei de viață a întregului sistem.

Capacitatea de stocare flexibilă înseamnă că se poate utiliza un model cu capacitate foarte apropiată de cea necesară pentru aplicație, pentru a nu crește costurile aferente cu memoria suplimentară ce nu va fi niciodată folosită. Comenzile interfeței IDE sunt conforme cu standardului industrial ATA-4.

Flash-Disk-urile au fost verificate cu succes sub următoarele sisteme de operare: MS-DOS, Windows 95/98/NT, Windows CE, OS2 Warp, Linux, QNX, Unix Ware. Media timpului de bună funcționare (MTBF) este de peste 1.000.000 de ore. Fiabilitatea datelor memorate este asigurată prin funcții încorporate de detectare și corecție a erorilor.

Flash-Disk-urile au performanțe foarte bune în ceea ce privește anduranța: pentru operații de scriere/citire sunt garantate pentru circa un milion de cicluri, iar pentru operații de citire se prevede un număr nelimitat de astfel de operații.

În ceea ce privește viteza de transfer pe magistrală, aceasta este de peste 700 KB/sec în caz de citire susținută și de peste 250 KB/sec la operații de scriere susținută. Flash-Disk-urile se remarcă, de asemenea, prin gama temperaturilor de lucru (de la -25°C la +85°C) și prin altitudinea maximă la care pot funcționa corect (16.000m).

Valorile prezentate mai înainte sunt valabile pentru FlashDisk-urile tipice. Realizările de vârf, "state of the art", au performanțe net superioare, însă și costurile de producție sunt mai ridicate. Astfel, firma BiT Microsystems produce seria E-Disk cu performanțe deosebite și cu o capacitate de stocare foarte ridicată. Se urmărește înlocuirea unităților de hard-disk cu module semiconductoare cu viteza de lucru mare, fiabilitate, longevitate și scalabilitate superioare. Rata de transfer este de 40 MB/sec în mod și de 34 MB/sec în mod susținut, iar capacitatea poate ajunge până la valoarea de 13.312 MB. Aceste dispozitive nu conțin componente în mișcare, care reprezintă principala cauza a defectărilor și întârzierilor electromecanice la HDD-urile clasice.

Ca dovadă suplimentară a performanțelor ridicate, E-Disk sunt prevăzute nu cu obișnuita interfață EIDE, ci cu variante avansate de SCSI: SCA Ultra Wide SCSI cu tensiune diferențială redusă.

Media timpului de bună funcționare ajunge la 1,9 milioane de ore, se folosesc coduri detectoare și corectoare de erori foarte performante, astfel că rata erorilor nedetectate este foarte scăzută. Astfel de dispozitive de stocare de înaltă fiabilitate sunt ideale pentru telecomunicații, transporturi, aplicații industriale, aerospațiale sau militare, unde sistemele sunt supuse unor condiții de temperaturi extreme, câmpuri magnetice puternice, vibrații, praf, mizerie și vapori corosivi. Sunt de asemenea potrivite pentru aplicațiile unde este esențial un coeficient foarte ridicat de disponibilitate și o viteză mare: depozite de date, multimedia, financiare, furnizori de servicii Internet, comerț electronic, servere proxy etc.

În continuare se prezintă un alt tip de echipament fiabil redundant, care, spre deosebire de alte soluții prezentate, are o performanță mai bună decât echipamentul de bază. În plus, acest echipament adaugă o dimensiune nouă în spațiul opțiunilor fiabilității, și anume capacitatea de a fi reparat în timp ce funcționează (maintainability).

Structura este cunoscută sub numele de **RAID**, care este o prescurtare de la Redundant Array of Inexpensive Disks, sau set redundant de discuri ieftine. Ideea a fost introdusă în 1987 de cercetători de la universitatea Berkeley din California.

Ideea centrală a structurii RAID este de a stoca informație pe mai multe discuri simultan. Informația este codificată redundant, astfel încât să poată fi recuperată dacă oricare dintre discuri se defectează. Această proprietate este foarte utilă pentru echipamentele care trebuie să funcționeze continuu.

Există mai multe tipuri de echipamente RAID, însă în continuare se prezintă unul singur, în care informația este scrisă pe 5 discuri, din care 4 conțin date și unul paritate. Un astfel de echipament RAID se poate afla într-unul dintre trei moduri de funcționare:

Funcționare normală: operațiile de citire extrag date de pe cele patru discuri cu date. O operație de scriere însă strânge patru blocuri de informație și calculează un al cincilea bloc de paritate; fiecare bloc este stocat pe alt disc. Acest mod de scriere se numește "striping", adică "feliere", pentru că datele sunt scrise în paralel, câte o felie pe fiecare disc.

Funcționarea degradată: este începută când un disc se defectează. Atunci citirile și scrierile de pe discul stricat trebuie să acceseze celelalte patru discuri și să calculeze informația lipsă. Avantajul parității este că oricare din biții lipsă poate fi recalculat ca paritate a celorlalți patru biți.

Reconstrucția: este începută când un disc defect este înlocuit. Un proces secundar recalculează informația lipsă și o scrie pe noul disc.

În continuare se prezintă un echipament de calcul dezvoltat de cercetători de la Hewlett-Packard, care demonstrează o metodologie extremă în tratamentul fiabilității echipamentelor. Acest echipament, cunoscut sub numele de Teramac, este construit din componente defecte: mai mult de 70% dintre circuitele sale componente au o malfuncție oarecare. Cu toate acestea, echipamentul funcționează corect și poate efectua calcule extrem de performante.

Arhitectura echipamentului Teramac tolerează numai defecțiuni permanente, care sunt prezente încă de la fabricație. Componentele Teramac sunt circuite hardware de un tip anume, numit hardware reconfigurabil. Înainte de a prezenta echipamentul Teramac se va face o prezentare succintă a structurii hardware-ului reconfigurabil, arătând cum un echipament fiabil poate fi construit din componente reconfigurabile nefiabile.

Într-o primă aproximație, circuitele digitale obișnuite sunt compuse din elemente computaționale simple, numite porți logice, conectate între ele prin sârme. Porțile logice sunt construite din tranzistori. Fiecare poartă logică face calcule pe mai multe valori de 1 bit. Porțile logice sunt universale, în sensul că orice proces calcul poate fi exprimat în termeni de operații ale porților logice.

În hardware-ul reconfigurabil porțile logice nu au o funcționalitate fixată, iar sârmele formează o grilă. Fiecare poartă este configurabilă, adică poate fi forțată să efectueze orice operație logică. La fiecare intersecție de sârme se află un mic comutator configurabil, care poate fi, de asemenea, programat să conecteze sârmele. Configurarea porților și a sârmelor se face prin semnale electrice. Fiecare poartă și fiecare comutator are o mică memorie asociată, în care-și stochează configurarea. Pentru că schimbând conținutul acestor memorii putem schimba funcționalitatea hardware-ului, circuitele acestea se numesc "reconfigurabile" (figura 7.3).

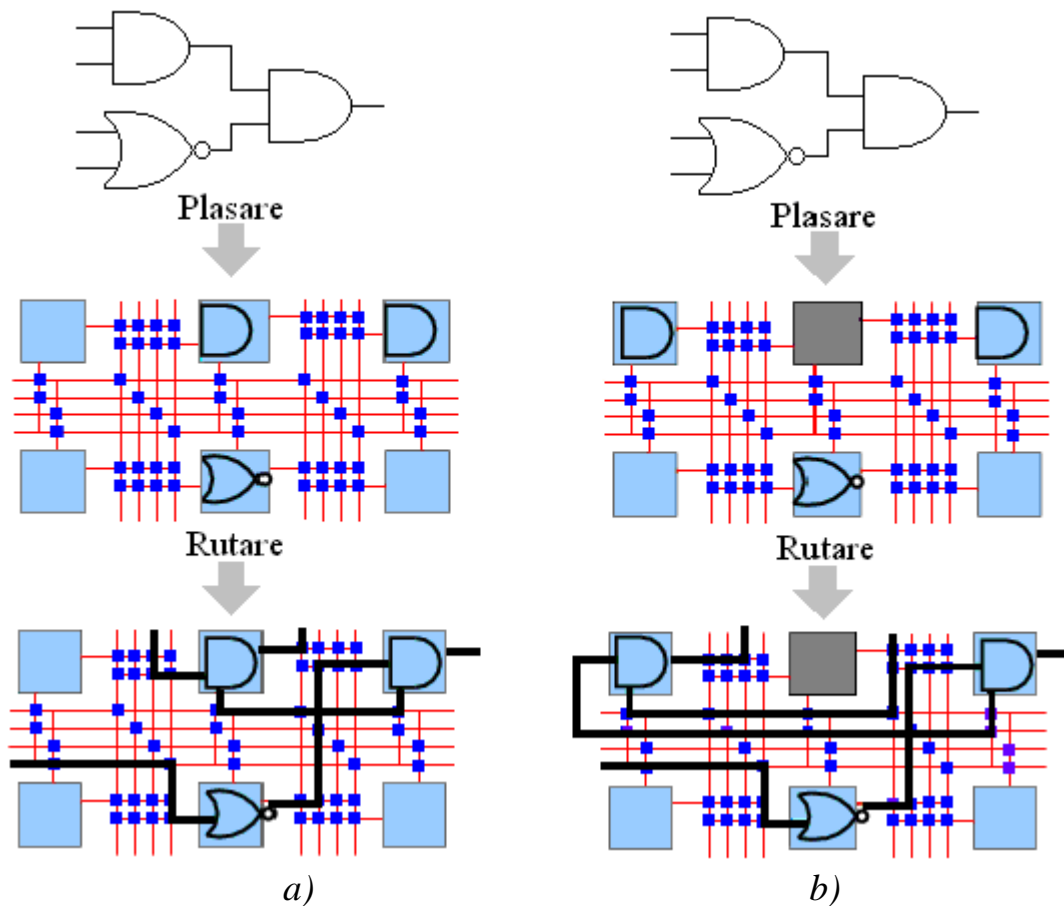


Figura 7.3. Structura unui circuit reconfigurabil

Hardware-ul reconfigurabil este echivalent cu cel obișnuit, în sensul că orice circuit poate fi implementat folosind ambele tehnologii. Hardware-ul reconfigurabil tinde însă să fie inefficient: memoriile și porțile configurabile ocupă mult mai mult loc decât porțile obișnuite. Pe de altă parte, semnalele electrice care traversează doar sârme într-un circuit obișnuit, trebuie să treacă printr-o serie de comutatoare în hardware-ul reconfigurabil, ceea ce face circuitele mai lente. Un factor de 10, diferență în viteză și densitate, este de așteptat între un hardware obișnuit și cel configurabil de aceeași generație.

Pentru a programa un circuit reconfigurabil cu funcțiunea unui circuit obișnuit, trebuie să asociem fiecare poartă din circuit cu o poartă configurabilă; acest proces se numește "plasare"; de asemenea, fiecare sârmă trebuie asociată cu succesiuni de segmente legate prin comutatoare, în procesul de "rutare".

În figura 7.3-a procesul de plasare asociază fiecare poartă logică din circuitul de implementat cu o poartă universală. Procesul de rutare conectează porțile universale folosind segmente de sârmă legate cu comutatoare. În cazul defectării unora dintre porțile universale plasarea și rutarea le pot ocoli, sintetizând un circuit perfect funcțional, situație ilustrată în figura 7.3-b.

Calitatea circuitelor reconfigurabile exploatare de Teramac pentru a obține fiabilitate este faptul că porțile logice configurabile sunt esențialmente interschimbabile. Cercetătorii proiectului Teramac au dezvoltat un program de plasare care folosește o hartă de defecte ale circuitelor reconfigurabile. Această program ocolește porțiunile inutilizabile și rutează conexiunile în jurul defectelor, exploatând doar porțiunile funcționale ale fiecărui circuit (figura 7.3-b). Cercetătorii au creat și o serie de programe, care descoperă și cataloghează defectele. Programele acestea folosesc chiar programabilitatea circuitelor pentru a le configura ca dispozitive care se auto-testează. Fiecare porțiune din fiecare circuit este programată să efectueze calcule simple și să verifice corectitudinea rezultatelor. Micile programe de test sunt ”plimbate” pe suprafața circuitului, acoperind toate porțile logice. Proiectarea unor programe de auto-testare este o sarcină mai complicată decât ar putea părea la prima vedere. Programele trebuie să descopere o mulțime de defecte posibile și trebuie să nu poată fi păcălite de defecțiuni (de exemplu, dacă chiar partea care compară rezultatele cu cele corecte este defectă). Programele de testare aplică în mod repetat calcule care amestecă toți biții: astfel, apariția unei singure erori se va propaga rapid la toți biții din rezultat, fiind ușor de depistat.

Proiectul Teramac a avut un succes enorm, principala sa contribuție a constat în a demonstra că defectele din hardware pot fi expuse nivelelor superioare, și pot fi tratate în întregime în software, fără ca costul plătit în performanță să fie prohibitiv. Această metodologie este o schimbare completă de paradigmă în arhitectura calculatoarelor, care probabil va avea din ce în ce mai multe aplicații în viitor.

7.3. Fiabilitatea programelor

Ne-am putea aștepta ca spre deosebire de hardware, software-ul să nu aibă nici un fel de probleme de fiabilitate deoarece programele nu se uzează, și sunt executate într-un mediu foarte specializat; în plus, programele sunt obiecte deterministe, deci ar trebui să se comporte de fiecare dată în același fel când procesează aceleași date de intrare. Cu toate acestea, de fapt fiabilitatea programelor este mult mai scăzută decât a echipamentelor hardware; este potrivit să modelăm deci programele ca echipamente cu fiabilitate imperfectă. În această secțiune se prezintă în mod superficial unele dintre motivele lipsei de fiabilitate a programelor și se menționează unele tehnici care pot fi folosite pentru a realiza programe fiabile.

Cea mai importantă cauză a defectelor programelor sunt bug-urile, adică implementări incorecte. Chiar și programatori foarte pricepuți produc programe cu defecte. Complexitatea componentelor software este pur și simplu prea mare, în momentul de față, pentru a putea fi stăpânită de către oameni. Cu tot progresul în tehnici de programare, cum ar fi descompunerea programelor în module mici, folosirea unor limbaje de programare evaluate și a unor scule complexe pentru dezvoltarea, testarea și analiza programelor, rezultatele sunt încă foarte departe de perfecțiune, iar productivitatea programatorilor nu a crescut substanțial în ultimele două decenii.

Cel mai adesea problemele rezolvate în software sunt atât de complicate încât nici nu pot fi specificate în mod precis. În consecință programatorii întâlnesc tot felul de incertitudini când încearcă să implementeze soluțiile. O cauză fundamentală a lipsei de fiabilitate a programelor este deci specificația de proiectare incompletă și imprecisă.

Cele mai imprevizibile defecțiuni software se manifestă numai cu ocazia unor anumite combinații de valori pentru datele de intrare sau pentru anumite succesiuni de evenimente externe, care nu au fost prevăzute de programator. Asemenea combinații apar cu probabilitate foarte mică în timpul procedurilor normale de testare, deci adesea supraviețuiesc până în faza operațională.

A vedea programele software ca pe o entitate monolitică este o aproximare grosolană a realității: un program trece prin nenumărate revizii și îmbunătățiri. Versiunile noi sunt construite pe scheletul celor vechi, reparând defecțiunile descoperite și adăugând noi funcționalități. Cu toate acestea, procesul reparării defecțiunilor introduce adesea noi defecțiuni, pentru că efectele unei reparații au uneori consecințe imprevizibile.

Creșterea continuă a performanțelor hardware-ului este o motivație constantă pentru reînnoirea echipamentelor software. Pe măsură ce dispozitivele hardware devin mai ieftine și mai compacte, ele pot fi integrate în dispozitive electronice mai "deștepte". Toate aceste noi dispozitive au nevoie de un nou software, care să le manipuleze. Pe măsură ce costul dispozitivelor de stocare a informației scade, din ce în ce mai complexe și mai bogate tipuri de informație pot fi stocate și prelucrate. De exemplu, imagini și muzică sunt tipuri curenți manipulați de PC-urile contemporane, iar capacitatea lor de prelucrare a devenit de curând suficient de puternică pentru a manipula în mod interactiv chiar filme.

Un fenomen legat de acest ciclu permanent de înnoiri este cel cunoscut sub denumirea "putrezirea biților" (bit rot). Acest fenomen se manifestă pe două planuri: datele stocate cu mult timp în urmă nu mai pot fi folosite în noile echipamente de calcul, pentru că dispozitivele periferice învechite nu mai sunt suportate de fabricanți, și programe vechi, care mergeau foarte bine, încep să manifeste erori. "Boala" programelor este legată de mediul în care programele se execută, și care este în continuă schimbare. De exemplu, multe programe vechi făceau anumite presupuneri despre cât de mari vor fi seturile de date pe care le vor prelucra. Cea mai faimoasă astfel de presupunere este cea care a cauzat bug-ul Y2K: programatorii din anii '60 au presupus că programele lor nu vor manipula niciodată date calendaristice al căror an nu va începe cu cifrele 19. Chiar dacă Y2K a făcut mai mult zgomot decât pagube, astfel de presupuneri se întâlnesc la tot pasul în programele de astăzi. De exemplu, pot apare dificultăți în a transporta programe de la procesoare pe 32 de biți la procesoare pe 64 de biți. Din moment ce orice valoare pe 32 de biți se poate reprezenta exact atunci când folosim 64 de biți, teoretic nu ar trebui să fie nici o problemă, și vechile programe ar trebui să funcționeze corect. În realitate multe programe depind în feluri subtile de precizia datelor pe care le manipulează. Când un astfel de program este mutat pe o platformă nouă toate aceste dependențe se transformă în bug-uri.

7.3.1. Creșterea fiabilității produselor software

Domeniul ingineriei programelor (software engineering) se ocupă de metode prin care se poate cuantifica și îmbunătăți calitatea programelor. Una dintre soluțiile studiate este foarte înrudită cu tehnicile de votare folosite pentru toleranța erorilor hardware. Numele acestei soluții în lumea software este "programare cu N versiuni". Votarea folosește redundanță spațială: dispozitivul de calcul este replicat de N ori și rezultatul final este obținut prin votul majoritar al rezultatelor individuale.

Bug-urile software sunt persistente: aflat în aceleași condiții programul se va comporta în același fel. Tehnicile de votare sunt neputincioase dacă toate componentele fac aceeași eroare în același timp. Votarea este utilă pentru tratamentul erorilor tranzitorii. Programarea cu N versiuni se face deci prin executarea în paralel a N programe diferite, scrise de echipe diferite de programatori, dacă e posibil, folosind scule și tehnologii diferite. Toate cele N programe rezolvă aceeași problemă, dar în moduri diferite. Numai folosind o astfel de strategie tehnica votării poate funcționa în cazul programelor.

Specificații imprecise ale problemei pot fi detectate cu ușurință de această tehnică, pentru că implementările diferite pot lua decizii diferite pentru cazurile nespecificate clar. Din nefericire, programarea cu N versiuni este o metodologie foarte scumpă, folosită numai pentru aplicații critice, unde siguranța este fundamentală.

Diferența fundamentală între hardware și software este aceea că un program poate avea o stare internă arbitrar de complicată. În general, dispozitivele hardware pot fi approximate ca fiind automate finite (adică spațiul stărilor în care se pot afla, chiar dacă este foarte mare, este totuși finit). Chiar și cele mai simple programe au un spațiu de stări infinit, mai exact, nu putem pune nici o limită arbitrară dimensiunii spațiului lor.

Această diferență este foarte importantă și din punct de vedere teoretic: foarte multe proprietăți interesante ale automatelor finite se pot decide, adică se pot scrie algoritmi care atunci când primesc descrierea unui automat finit, pot răspunde în mod exact la întrebări legate de orice evoluție viitoare a automatului. Din păcate, aceleași întrebări pentru un echipament cu stare infinită adesea nu pot fi decise. Într-adevăr, matematicienii au arătat în anii '30 că foarte multe dintre proprietățile unui echipament software, în general, nu pot fi calculate de un alt echipament software.

O consecință practică a dimensiunii infinite a spațiului de stări ale programelor este că, pe măsură ce un program se execută mai mult timp, cu atât mai complicată poate deveni starea sa internă. Dacă un program nu își întrerupe execuția, chiar dacă va primi aceleași date la intrare, ar putea calcula un răspuns diferit. Un bug în program poate corupe starea internă, dar efectele acestei defectări pot deveni vizibile mult mai târziu în execuția programului, când programul ia o decizie bazată pe elementele de stare incorecte. Un tip faimos de problemă, în mod normal benignă, asociată cu programele care se execută un timp îndelungat, este scurgerea de memorie (memory leak).

Adesea programele alocă spațiu temporar de memorie, pe care îl eliberează după ce au terminat calculele care aveau nevoie de el. Dacă programatorul uită să elibereze această memorie se spune că memoria se scurge (leak). Aceasta este o eroare frecvent întâlnită în programare, relativ greu de descoperit. În mod normal o astfel de eroare nu afectează corectitudinea programului: rezultatele produse la final sunt corecte. Când programul își termină execuția, echipamentul de operare recuperează automat memoria scursă. În cazul programelor care se execută timp îndelungat, cum ar fi echipamentele de operare sau serverele de web, dacă o scurgere se

întâlnește în interiorul unei bucle, cu timpul memoria pierdută va crește până când toată memoria echipamentului este pierdută. În astfel de cazuri de obicei echipamentul își încetează execuția, sau funcționarea sa devine extrem de lentă din cauză că resursele rămase sunt insuficiente.

Utilizatorii echipamentului de operare Windows de la Microsoft au descoperit și o soluție pentru această problemă: *reboot-area calculatorului*. Numele științific pentru această soluție este "reîntinerirea programelor" (software rejuvenation). Reîntinerirea este cauzată de repornirea periodică a programelor. Repornirea cauzează inițializarea stării interne la o aceeași valoare inițială. Tehnica aceasta este aplicabilă numai dacă starea internă a programului nu este importantă și poate fi pierdută; altfel, întinerirea trebuie să fie combinată cu "checkpoint"-uri. Un checkpoint salvează informația importantă pe un mediu de memorie persistent, și o restaurează după ce programul este repornit.

Reîntinerirea se aplică cu precădere programelor de tip server, care execută tot timpul o buclă, acceptând cereri de la clienți și răspunzându-le. Multe servere sunt lipsite de stare (stateless), adică nu păstrează nici un fel de informații despre o tranzacție cu un client după ce tranzacția s-a consumat. Reîntinerirea este eficientă dacă costul repornirilor periodice este mai redus decât costul repornirii după o cădere catastrofică, care poate să implice o procedură sofisticată de recuperare a datelor pierdute. Reîntinerirea este de asemenea folosită cu succes când serverele care oferă serviciul au rezerve, astfel încât serverele de rezervă pot răspunde clienților în timp ce altele se reinițializează.

O altă modalitate de creștere a fiabilității programelor de calcul constă în utilizarea unei tehnici numită *verificare formală*. Acesta este un nume generic pentru o serie întreagă de tehnici sofisticate care certifică corectitudinea, mai ales a echipamentelor hardware, dar în ultima vreme și a unor echipamente software. Verificarea formală se ocupă de descoperirea și eliminarea bug-urilor, și în acest sens este o tehnică de creștere a fiabilității programelor.

Cheia metodelor de verificare formală este specificarea foarte precisă a comportării componentelor echipamentului de analizat (folosind formule matematice) și verificarea automată a proprietăților echipamentului în întregime. Dacă știm cum este construit echipamentul, și dacă știm comportarea fiecăreia dintre componente, putem raționa despre comportarea ansamblului. Raționamentele pot fi făcute foarte precise folosind diferite variante de logici matematice. Fiecare raționament este o serie de derivări, în care din fapte știute ca fiind adevărate deducem alte

adevăruri. Verificarea formală studiază aceste derivări, și verifică faptul că sunt corecte.

Două aspecte fac din verificarea formală o tehnică foarte puternică:

- 1) calculele minuțioase sunt efectuate de către calculatoare, a căror atenție nu obosește niciodată;
- 2) certitudinea nu vine din faptul că demonstrăm ceva, ci din faptul că putem verifica dacă demonstrația este corectă.

Când s-a descris echipamentul DIVA s-a făcut mențiunea că a verifica corectitudinea unui rezultat este mult mai simplu decât a demonstra rezultatul însuși. Acest fapt este extrem de folositor în contextul verificării formale, în care programul care face demonstrațiile este extrem de complicat, și ca atare poate conține erori (ca orice alt program complex) și deci poate genera demonstrații eronate. Un program care verifică dacă o demonstrație este corectă însă este mult mai simplu, și ca atare ne oferă mai multă încredere.

7.4. Rețeaua internet

Una dintre cele mai uimitoare tehnologii ale secolului douăzeci este cu siguranță Internetul. Acesta este o rețea de calculatoare, proiectată inițial pentru a conecta rețele militare de calculatoare și de a le permite să opereze chiar și în condițiile distrugerii unui mare număr de echipamente din rețea, de exemplu în cazul unei conflagrații nucleare. Internetul a evoluat astăzi într-o rețea comercială care acoperă toate continentele, cu mai mult de 125 de milioane de calculatoare și peste 1 miliard de utilizatori.

Internetul nu este prima rețea de dimensiune globală; cu mai mult de un secol înainte de crearea Internetului a apărut telefonul; rețelele telefonice au cu siguranță întâietatea în acoperirea planetei. Ne-am aștepta ca proiectanții Internetului să fi folosit multe din tehnologiile folosite în construcția rețelelor telefonice, despre care există o cantitate mare de informații și o experiență substanțială. În realitate, nimic nu poate fi mai departe de adevăr: arhitectura Internetului pare a fi în mod deliberat opusă rețelei de telefonie. Nicăieri nu se vede mai bine diferența dintre cele două rețele decât în felul în care tratează fiabilitatea.

Rețeaua telefonică a fost proiectată de la început pentru o fiabilitate excepțională. O centrală telefonică trebuie să însumeze mai puțin de trei minute de indisponibilitate în fiecare an. Numai în circumstanțe absolut excepționale o conversație inițiată poate fi întreruptă datorită unor

probleme din rețea. Rețeaua telefonică va permite stabilirea unei legături numai după ce a rezervat toate resursele necesare pentru transmisiunea promptă a semnalelor vocale pe întregul traseu dintre cele două puncte care comunică. Standarde stricte dictează cât de mult timp poate dura faza de construcție a legăturii; dacă nu pot fi obținute toate resursele utilizatorul primește un ton de ocupat. Capacitatea rețelei este planificată atent pe baza unor statistici detaliate despre comportarea vorbitorilor, astfel încât în condiții normale, șansa obținerii unui ton de ocupat din cauza resurselor insuficiente din rețea să fie extrem de redusă.

Un factor crucial care garantează calitatea conexiunilor telefonice este prealocarea tuturor resurselor necesare înainte ca legătura să fie stabilită. Pornind de la numărul format, prima centrală telefonică calculează o secvență de centrale prin care semnalul trebuie să treacă pentru a lega apelantul cu apelatul; acest calcul se bazează pe tabele de rutare precalculate cu mare grijă și stabilite de către proiectanții rețelei. Fiecare centrală negociază apoi cu cea succesivă folosind un protocol sofisticat de semnalizare, și alocă capacitate pentru transportul datelor și pentru comutarea acestora (care în centrală leagă circuitul de intrare cu cel de ieșire). Când toate conexiunile punct-la-punct între centrale sunt stabilite se generează un ton de "apel". Când conversația a fost inițiată, semnalul vocal este eșantionat și digitizat în prima centrală. Pentru fiecare bit din acest semnal s-a prealocat deja o cuantă periodică de timp pe fiecare dintre circuitele pe care le va traversa. Biții sunt transmiși unul câte unul și traversează toate trunchiurile în aceeași ordine în care au fost generați, sosind la destinație la timp pentru a fi reasamblați și convertiți la loc într-un semnal auditiv. Din cauza prealocării, de îndată ce un bit intră în rețea, cu o probabilitate extrem de ridicată el va ajunge la celălalt capăt exact când trebuie. Când unul dintre vorbitori închide telefonul, protocolul de semnalizare intră din nou într-o fază complicată, prin care eliberează toate resursele alocate la momentul apelului.

Rețeaua Internet are o arhitectură fundamental diferită. Nu numai că nu există garanții despre timpul necesar pentru a ajunge de la emițător la receptor, dar nu există nici o garanție că datele nu sunt pierdute sau modificate în timpul transferului. Utilizatorii Internetului obțin un serviciu extrem de "slab", care poate fi enunțat pe scurt astfel: "tu pui date în rețea și zici unde vrei să ajungă, iar rețeaua o să încerce să livreze datele acolo".

Felul în care informația circulă în rețeaua Internet este complet diferit de rețeaua telefonică: datele sunt divizate în pachete care sunt introduse în rețea în ordinea sosirii. Fiecare pachet poate călători pe o rută complet

diferită până la destinație. Unele pachete se pot pierde, alte pot fi duplicate, și ele pot sosi la destinație în altă ordine decât au fost emise, sau chiar sparte în pachete mai mici.

Pachetele sunt plimbate prin Internet de un protocol numit IP, Internet Protocol. IP funcționează aproximativ astfel: când un calculator intermediar primește un pachet se uită întâi la adresa destinație înscrisă. Apoi el face niște calcule simple pentru a decide în ce direcție pachetul trebuie trimis, mai precis, căruia dintre vecinii săi trebuie să-i dea pachetul. Pachetul este apoi trimis vecinului. Dacă la un calculator intermediar pachetele vin mai repede decât apucă să le trimită mai departe, și dacă nici nu are unde să le stocheze pentru o vreme, are dreptul să le facă pierdute. Aceasta este principala cauză pentru care datele se pot pierde în Internet.

Spre deosebire de rețeaua telefonică, structura Internetului nu este controlată de un număr mic de companii, ci este în continuă schimbare, de la zi la zi și de la oră la oră, pe măsură ce noi calculatoare se conectează, noi utilizatori sună folosind modemuri și noi linii de transmisiune sunt instalate. Calculatoarele responsabile pentru transmiterea datelor, numite rutere, discută între ele permanent pentru a afla care este forma curentă aproximativă a rețelei. Aceste informații sunt utilizate în procesul de decizie care selectează vecinul folosit pentru transmisiunea fiecărui pachet spre destinație.

Data fiind această infrastructură, este uimitor că Internetul funcționează câtuși de puțin, și că informația ajunge câteodată neperturbată la destinație. Fiabilitatea aplicațiilor din Internet este construită pe baza acestui mediu extrem de nefiabil, folosind două ingrediente:

1) Lipsa de stare. Ruterile din Internet nu stochează nici un fel de informații despre traficul care le parcurge. Prin contrast, în rețeaua telefonică, comutatoarele știu despre fiecare bit care le traversează de unde vine, unde se duce, și când va sosi succesorul lui. Un ruter primește un pachet, calculează vecinul căruia să-i dea pachetul și livrează pachetul. Starea internă a ruterului după livrarea pachetului este aceeași cu cea de dinainte. Așa cum am văzut în cazul reîntineririi programelor, lipsa stării interne face mult mai simplă repornirea unui ruter după o defecțiune. O defecțiune nu pierde informații vitale, care nu pot fi recuperate prin alte metode. De asemenea, lipsa stării face relativ ușoară sarcina altor rutere de a prelua traficul în cazul defectării unuia.

2) Protocolul TCP. Realizarea unei transmisii fiabile prin rețeaua Internet se face utilizând un protocol numit Transmission Control Protocol (TCP),

care se execută deasupra protocolului IP. Dacă într-o rețea toate nodurile din interior execută IP, *numai sursa și destinația execută TCP*. TCP este protocolul care construiește o transmisiune fiabilă: el asigură că toate pachetele trimise ajung la destinație, fără lipsuri sau duplicate, în ordinea în care au fost trimise. TCP reușește această performanță folosind următoarele mecanisme:

- numerotează pachetele trimise;
- folosește pachete de confirmare pentru a anunța sosirea datelor la capătul celălalt;
- folosește alarme pentru a detecta pachetele care nu sunt confirmate pentru un timp îndelungat;
- folosește retransmisii pentru a retrimite pachetele care se pierd în rețea.

Din cauză că pachetele cu confirmări se pot pierde la rândul lor, unele pachete sunt injectate în mod repetat în rețea, ceea ce poate duce la livrarea unor duplicate; TCP trebuie să le elimine folosind numerele de serie ale pachetelor.

Întregul Internet este construit pe nucleul nefiabil oferit de IP: nu numai datele și confirmările sunt trimise în mod nefiabil, dar chiar și mesajele de control schimbate între rutere, prin care află despre schimbările din topologia rețelei și traficul folosit pentru monitorizarea și mentenanța rețelei folosesc aceleași mecanisme nefiabale de transmisiune.

În pofida structurii sale aparent șubrede, Internetul este un competitor formidabil al altor forme de distribuție a informației: radio, televiziune și telefonie. Costul transmisiunii vocii prin Internet este mult mai scăzut decât folosind rețelele specializate de telefonie. Multe companii importante de telefonie investesc în mod serios în echipamente care transportă voce peste protocolul IP.

Internetul mută problema fiabilității la un nivel superior, de la IP la TCP. TCP oferă o fiabilitate perfect adecvată pentru multe aplicații. TCP este executat numai de către calculatoarele terminale implicate în comunicație, și nu de către rutere. Ca atare algoritmi complicați folosiți de acest protocol nu taxează resursele rețelei, care scalează în mod natural la dimensiuni globale.

Mai mult, unele aplicații care nu au nevoie de livrarea fiabilă a datelor nu sunt obligate să folosească protocolul TCP. De exemplu, protocoalele folosite pentru posturile de radio din Internet folosesc coduri puternice de corecție a erorilor și nu au nevoie de retransmisii. Pachete pierdute sau

întârziate sunt pur și simplu ignorate. Acest lucru este acceptabil pentru că utilizatorul final, omul, tolerează semnale cu zgomot.

7.5. Concluzii

În acest capitol au fost prezentate unele considerații privind modul cum fiabilitatea influențează construcția echipamentelor de calcul. O concluzie foarte importantă care rezultă din analiza făcută este că deși fiabilitatea ridicată este dezirabilă, un proiectant trebuie întotdeauna să ia în considerație și costul plătit pentru a o obține.

Calculatoarele moderne sunt construite dintr-o serie de nivele abstracte, care oferă funcționalități din ce în ce mai puternice. Fiecare nivel are o fiabilitate diferită și folosește tehnici diferite pentru a oferi nivelurilor superioare imaginea unei fiabilități sporite. În general hardware-ul oferă lumii software aparența perfecțiunii în această privință, adică o fiabilitate excepțional de ridicată.

Tendențele tehnologiei indică însă că arhitectura calculatoarelor viitorului va fi supusă unor schimbări radicale, unul dintre motive fiind chiar schimbarea majoră a fiabilității unora dintre nivele. De exemplu, miniaturizarea continuă a componentelor electronice va fi însoțită de o degradare a fiabilității însoțită de apariția tot mai frecventă a defecțiunilor permanente și tranzitorii. Costul plătit pentru a masca aceste defecte prin tehnici tradiționale crește extrem de rapid: costul extrem de ridicat al unei fabrici de semiconductoare din ultima generație, de ordinul a câteva miliarde de dolari, este doar primul simptom al acestui fenomen.

În prezent, departamentele de cercetare ale firmelor producătoare de echipamente de calcul lucrează în mod activ pentru a defini arhitecturile structurilor de calcul al viitorului. Direcțiile urmărite sunt prezentate în continuare:

- Iluzia unui hardware perfect trebuie eliminată, imperfecțiunile din nivelul hardware trebuie să fie expuse nivelului software și rezolvate de acesta. Pentru a face acest lucru, baza echipamentelor de calcul trebuie să fie hardware-ul reconfigurabil, care este suficient de flexibil pentru a fi reprogramat după nevoi.
- Microprocesorul trebuie să fie redus la un rol secundar și înlocuit cu hardware generat specific, pentru fiecare aplicație, cu ajutorul compilatoarelor.

- Numărul de nivele abstracte trebuie să fie micșorat în mod dramatic, pentru a reduce costul suplimentar plătit, care crește exponențial.
- Trebuie folosite în mod constant tehnici care descoperă defecte și folosesc rezerve pentru a ocoli defectele de fabricație.
- Calculele trebuie să fie efectuate folosind date codificate utilizând coduri robuste, pentru a preveni efectele erorilor tranzitorii.
- Utilizarea pe scară largă a metodelor de verificare formală pentru a ne asigura că echipamentele de calcul pe care le folosim sunt corect construite.

Știința calculatoarelor este relativ tânără și, cu siguranță, viitorul ne rezervă o mulțime de surprize în ceea ce privește tehnologiile, arhitectura și algoritmi cei mai eficienți.