PREFAȚĂ

Rezultatele cercetării științifice prezentate în cadrul acestei cărți reprezintă o continuare, dezvoltare și aprofundare a celor expuse în cadrul cărții [Pi13], axându-se pe dezvoltarea de soluții informatice pentru procesarea paralelă a datelor pe cele mai noi unități de procesare grafică ce înglobează arhitectura Compute Unified Device Arhitecture (CUDA), ***Kepler (2012)*** și ***Maxwell (2014)***. De un interes deosebit a fost cercetarea posibilităților de dezvoltare a soluțiilor informatice prin prisma noilor facilități oferite de capabilitatea computațională 5.2 a arhitecturii ***Maxwell GM204*** comparativ cu capabilitatea computațională 3.0 a arhitecturii ***Kepler GK104***.

Cercetarea desfășurată în cadrul acestei cărți vizează identificarea și dezvoltarea de soluții în vederea optimizării performanțelor software ale unor funcții algoritmice de bază (identificate, prezentate și analizate în [Pi13]). Aceste funcții sunt deseori apelate în etapele de procesare ale unei multitudini de algoritmi și aplicații utile în procesarea datelor. În cadrul cercetărilor din [Pi13] funcțiile algoritmice de bază au fost dezvoltate în mediul de programare CUDA, fiind destinate pentru trei arhitecturi de procesoare grafice din diferite generații (cele mai recente ale acelui moment) ce implementează CUDA: GT200 Tesla, GF100 Fermi și GK104 Kepler.

În cadrul acestei cărți a fost studiată posibilitatea implementării funcțiilor algoritmice de bază în arhitectura de procesare paralelă CUDA, pe cele mai recente arhitecturi de procesoare grafice din generații diferite, GTX 680 (din arhitectura ***Kepler 2012***) și GTX 980 (din arhitectura ***Maxwell 2014***).

Deși procesorul grafic din arhitectura GK104 a fost utilizat și în cercetarea anterioară din cadrul [Pi13], o parte dintre cercetările acelui moment au fost reluate întrucât posibilitățile de optimizare oferite la ora actuală (2015) de mediul de dezvoltare CUDA Toolkit 6.5, sunt net superioare față de cele ale mediului

CUDA Toolkit 4.1, disponibil la momentul cercetării anterioare (2013). De asemenea, noi posibilități de optimizare pentru performanțele acestui procesor au apărut în anul 2015, odată cu lansarea versiunii de driver destinată dezvoltatorilor de aplicații Nvidia, utilă pentru a programa și accesa unitatea de procesare grafică. Astfel, versiunea utilizată în 2015 este 347.62, iar versiunea utilizată în cercetarea anterioară, în anul 2013, este 301.10.

În urma cercetărilor, au fost dezvoltate soluții de optimizare a performanței software a funcțiilor algoritmice în arhitectura Compute Unified Device Arhitecture, pe arhitecturile ***Kepler (2012)*** și ***Maxwell (2014)***. Astfel, a fost obținut un cadru de dezvoltare, ***BFAB-KM*** (Biblioteca de Funcții Algoritmice de Bază în cadrul arhitecturilor CUDA ***Kepler*** și ***Maxwell***), ce poate fi folosit în optimizarea aplicațiilor de procesare a datelor și a algoritmilor specializați, având un spectru larg de aplicabilitate în domeniul informaticii economice. Soluția ***BFAB-KM*** dezvoltată are în componență funcțiile algoritmice optimizate pentru arhitecturile ***Kepler (2012)*** și ***Maxwell (2014)*** oferind premisele necesare îmbunătățirii performanței software și dezvoltării de soluții informatice pentru o gamă variată de probleme din diverse domenii.

Efortul Nvidia de a oferi consumatorilor un produs atât pentru calcule de uz general cât și pentru redare grafică nu s-a limitat la producerea de hardware care incorporează arhitectura CUDA. Indiferent de cât de multe caracteristici a adăugat Nvidia procesoarelor sale grafice pentru a facilita calculele, aceste funcții nu puteau fi accesate fără a folosi OpenGL sau DirectX. Acest lucru ar fi impus dezvoltatorilor să își formuleze problemele computaționale ca probleme grafice, iar pe de altă parte ar fi fost necesar ca aceştia să cunoască şi să utilizeze limbaje de programare orientate către aplicațiile grafice precum GLSL, OpenGL sau HLSL Microsoft.

Pentru a face CUDA accesibilă unui număr cât mai mare de dezvoltatori, Nvidia a preluat limbajul C standard şi i-a adăugat o serie de cuvinte cheie, în scopul de a valorifica caracteristicile specifice ale arhitecturii CUDA, dezvoltând astfel limbajul CUDA C. Astfel, CUDA C a devenit primul limbaj dezvoltat în mod special de un producător de unități de procesare grafică (GPUs) în vederea facilitării calculelor computaționale de uz general, folosind GPU-uri programabile. În plus, pe lângă crearea unui limbaj pentru a scrie cod destinat unității de procesare grafică, Nvidia a oferit un driver specific pentru a exploata puterea masivă de calcul a arhitecturii CUDA. În consecință, nu a mai fost necesar

ca dezvoltatorii să aibă cunoștințe legate de OpenGL, de interfețe de programare DirectX, sau să îşi formuleze problemele sub forma unor sarcini de grafică computerizată.

Cercetarea abordată în cadrul acestei cărți este actuală și importantă, reflectând evoluția metodelor de procesare a volumelor de date prin tehnici și mijloace inovative. Soluțiile de optimizare ale funcțiilor algoritmice pentru arhitecturile ***Kepler (2012)*** și ***Maxwell (2014)*** rezolvă multe dintre deficiențele existente cu privire la obținerea unor resurse computaționale de performanță ridicată, la costuri accesibile.

Volumul mare de date acumulat într-o multitudine de domenii necesită dezvoltarea unor soluții eficiente de procesare, fiind una dintre temele principale de interes pentru cercetările din domeniu, reprezentând o provocare în continuă creștere. Dezvoltarea spectaculoasă a tehnicii de calcul, a sistemelor de gestiune a bazelor de date și a depozitelor de date, a conceptului de Big Data, a tehnologiilor de stocare Cloud, au avut drept efect o adevărată explozie informațională a volumului de date referitor la aspecte determinante ale vieții, societății și activităților din diverse sectoare economice. În consecință, este primordială dezvoltarea unor soluții cu un grad ridicat de performanță care să asigure procesarea eficientă, rapidă și la costuri accesibile a cantității imense de date acumulate, problemă ce îi preocupă pe oamenii de știință, pe cercetători și deopotrivă pe dezvoltatorii de aplicații. Una dintre cele mai promițătoare și avantajoase soluții pentru această problemă constă în utilizarea arhitecturilor de procesare paralelă.

Dezvoltarea de soluții eficiente de optimizare a procesării datelor este de o importanță majoră într-o multitudine de sectoare de activitate: medical, industrial, de afaceri, de cercetare, etc. În cadrul companiilor, pentru a obține rezultate în conformitate cu strategia economică stabilită, pentru a obține poziția de lider pe piața concurențială și pentru a asigura o dezvoltare susținută, este vitală implementarea de metode adecvate pentru procesarea și administrarea eficientă a datelor disponibile.

Astfel procesarea eficientă a datelor este extrem de importantă în dezvoltarea rețelelor neurale de predicție, în comerțul electronic, în monitorizarea și gestionarea traficului de date uriaș generat de rețelele de socializare, în optimizarea procesului de redare a conținutului multimedia la cerere (servicii video streaming, on demand, etc.), în analiza tranzacțiilor, în cercetarea unor probleme din matematică, fizică, anatomie, biologie, medicină, chimie, arhitectură și multe altele. O soluție eficientă de optimizare a procesării datelor trebuie să

folosească pe deplin resursele hardware și software disponibile pentru a rezolva sarcinile computaționale necesare în vederea efectuării unor analize eficiente de previzionare a evoluției economice a pieței.

Analiza situației actuale privind volumul de date acumulat în diverse domenii, necesitatea imperioasă a procesării eficiente a acestuia, au avut drept finalitate identificarea funcțiilor algoritmice de bază, cercetarea celor mai noi arhitecturi hardware și software care să faciliteze dezvoltarea și implementarea de soluții performante pentru optimizarea acestor funcții pe arhitecturile grafice de procesare paralelă ***Kepler (2012)*** și ***Maxwell (2014)*** care să faciliteze dezvoltarea de soluții performante și eficiente, precum și aplicarea acestora în dezvoltarea de aplicații privind procesarea datelor. Toate soluțiile obținute au avut drept efect înregistrarea unor rezultate spectaculoase, atât din punct de vedere al performanței obținute, cât și al costurilor implicate, comparativ cu rezultatele înregistrate prin utilizarea arhitecturilor ce folosesc unitățile de procesare centrală (CPUs).

Întrucât s-a decis proiectarea și rularea funcțiilor algoritmice de bază pe unități de procesare grafică ce facilitează execuția în paralel a sarcinilor implementând tehnologia CUDA, în dezvoltarea algoritmilor și a soluțiilor de optimizare a performanței software a acestora pe arhitecturile ***Kepler (2012)*** și ***Maxwell (2014)***, au fost implementate tehnici de paralelizare a execuției anumitor etape algoritmice.

Soluțiile informatice de optimizare a procesării datelor pe unități de procesare grafică sunt programate astfel încât să aloce automat resursele computaționale (numărul firelor de execuție per bloc, numărul blocurilor de fire de execuție, dimensiunea grilelor de blocuri), în urma analizei automate a volumului de date ce trebuie procesat și a caracteristicilor tehnice specifice arhitecturilor de procesare grafică de ultimă generație ***Kepler (2012)*** și ***Maxwell (2014)***. În vederea obținerii unui studiu cât mai relevant, corect și profund al performanțelor și eficienței înregistrate, s-a folosit un volum variabil de date de intrare (pornind de la dimensiuni mici până la dimensiuni foarte mari), o gamă variata de arhitecturi de procesare (atât unități de procesare grafică cât și unități de procesare centrală), un număr foarte mare de iterații de teste (câte 10.000 pentru fiecare caz analizat), o varietate de situații analizate (diverse tipuri de date, diverse tipuri de operatori binari asociativi utilizați).

În rularea testelor experimentale a fost utilizată următoarea configurație hardware și software:

- unitatea de procesare centrală: Intel i7-4770K la 3,5 GHz
- memoria cu acces aleatoriu (RAM): 16 GB (2x8GB) de memorie, de tip DDR3, cu frecvența 1333 MHz, rulând în modul dual channel
- unitățile de procesare grafică (GPUs): GeForce GTX 680 din arhitectura Kepler (GK104) și GTX 980 din arhitectura Maxwell (GM204)
- programarea și accesul la unitatea de procesare grafică s-au realizat prin CUDA Toolkit 6.5 și versiunea de driver Nvidia 347.62
- sistemul de operare: Windows 8.1 pe 64 biți
- procesele legate de interfața grafică a utilizatorului au fost dezactivate pentru a minimiza traficul extern al unității de procesare grafică
- întrucât funcțiile algoritmice sunt proiectate pentru a fi utilizate în dezvoltarea unui număr mare de aplicații ce rulează pe procesoare grafice, iar timpul aferent CPU și GPU pentru a efectua transferuri de date variază în funcție de complexitatea și specificul fiecărei aplicații, s-a decis ca măsurătorile efectuate să nu includă acești timpi
- soluția dezvoltată pentru a calcula timpul mediu necesar unității de procesare grafică pentru rularea aplicațiilor dezvoltate folosește evenimentele de înregistrare a ștampilelor de timp din cadrul interfeței de programare a aplicației (API).

În cadrul cărții sunt detaliate într-un mod consistent și riguros rezultatele obținute în urma rulării suitei de teste experimentale prin includerea tabelelor ce conțin valorile detaliate înregistrate ale timpilor de execuție, ale lățimilor de bandă, ale consumului de energie, prin includerea graficelor ce reprezintă într-un mod clar variația performantei în urma aplicării soluțiilor de optimizare dezvoltate. Astfel este realizată o analiză în profunzime a performanței și eficienței software înregistrate de funcțiile algoritmice optimizate pentru arhitecturile ***Kepler (2012)*** și ***Maxwell (2014)*****.**

Unitățile de procesare grafică (GPU) oferă resurse computaționale semnificative și un potențial imens în dezvoltarea de soluții informatice pentru optimizarea procesării datelor luând în considerare numărul mare de nuclee de procesare și fire de execuție disponibile. Gestionarea firelor de execuție la nivel hardware și nucleele de procesare virtualizate ale unităților de procesare grafică oferă oportunități considerabile pentru îmbunătățirea portabilității și scalabilității aplicațiilor de procesare de date dezvoltate.

Dezvoltatorul de aplicații are la dispoziție mai multe metode în vederea

îmbunătățirii performanței software a rulării unei funcții, performanță ce se reflectă prin timpul de execuție mai mic, lățimea de bandă mai mare, costuri de procesare semnificativ reduse. În particular, în cazul funcțiilor algoritmice de bază, se poate obține o îmbunătățire considerabilă a performanței prin creșterea capacității computaționale folosind mai multe unități de procesare centrală în cadrul unei platforme multi-procesor (platformă ce utilizează mai multe unități de procesare centrală în vederea procesării datelor) fie prin utilizarea unei soluții hibride ce folosește unități de procesare specializate ce conlucrează cu echipamentul gazdă de procesare (unitatea de procesare centrală).

Soluția de optimizare dezvoltată și prezentată în cadrul acestei cărți constă în folosirea capacității de procesare paralelă a unei unități de procesare grafică ce înglobează arhitectura CUDA alături de puterea de procesare secvențială a unității de procesare centrală. Această soluție s-a dovedit a fi cea mai avantajoasă atât din raționamente economice cât și prin prisma nivelului de performanță înregistrat.

În urma analizei sistemelor de procesare paralelă, se remarcă faptul că ultimele arhitecturi de procesare paralelă CUDA, ***Kepler 2012*** și ***Maxwell 2014*** oferă un potențial imens de optimizare a funcțiilor algoritmice identificate. Una dintre cele mai importante facilități oferite programatorilor de arhitectura Compute Unified Device Arhitecture constă în posibilitatea de a dezvolta aplicații performante, folosind funcții optimizate dezvoltate în CUDA, rulate pe unitățile de procesare grafică, grupate în cadrul unor biblioteci specializate. Apelând aceste funcții, se obține nu numai îmbunătățirea semnificativă a performanței aplicațiilor în care acestea sunt utilizate, cât și a tuturor etapelor procesării datelor.

Acesta a reprezentat punctul de pornire al cercetărilor abordate în cadrul acestei cărți: construirea unei biblioteci de funcții algoritmice de bază (***BFAB-KM***) în CUDA, destinată arhitecturilor ***Kepler2012*** și ***Maxwell 2014***, în vederea dezvoltării de soluții de optimizare a performanței funcțiilor algoritmice. Prin apelarea ***BFAB-KM*** în cadrul unei game variate de aplicații, programatorii au posibilitatea de a îmbunătăți performanța aplicațiilor proprii într-un mod rapid, eficient, avantajos din punct de vedere economic. În cadrul acestei cărți, după ce a fost argumentată necesitatea și importanța dezvoltării soluției ***BFAB-KM*** pentru programatori, este prezentată arhitectura soluției și apoi funcțiile algoritmice de bază incluse în ***BFAB-KM,*** dezvoltate pe arhitecturile ***Kepler2012*** și ***Maxwell 2014***: însumarea paralelă prefixată, însumarea paralelă segmentată, reducerea paralelă, sortarea paralelă radix și compactarea fluxurilor de date.

Structura cărții cuprinde **10 capitole** în care sunt prezentate elemente teoretice, dezvoltări ale soluțiilor informatice de optimizare a procesării datelor pe unități de procesare grafică, rezultate experimentale, analiza și interpretarea acestora, un capitol de concluzii, bibliografie și anexe. Cercetarea prezentată în carte are o succesiune evolutivă, a cărei structură este prezentată mai jos.

În **Capitolul 1**, "Sisteme de procesare paralelă a datelor", sunt analizate și prezentate elemente legate de procesarea paralelă a datelor, un scurt istoric și evoluția sistemelor de procesare paralelă, taxonomia lui Flynn, terminologia specifică sistemelor de procesare paralelă, gestionarea memoriei în sistemele paralele, paralelismul prin utilizarea firelor de execuție și paralelizarea programelor.

În cadrul **Capitolului 2,** "Arhitectura CUDA (Compute Unified Device Architecture) de optimizare a procesării paralele a datelor" sunt prezentate evoluția de la unități de procesare centrală la unități de procesare grafică, arhitectura "Compute Unified Device Architecture" (CUDA) și limbajul de programare CUDA C, concepte specifice ale modelului de programare "Compute Unified Device Architecture" (CUDA) și componentele arhitecturii CUDA (ierarhia firelor de execuție, ierarhia memoriei, echipamentul hardware gazdă și cel specializat, componentele arhitecturii CUDA), o comparație a celor mai importante arhitecturi de procesoare grafice ce implementează CUDA, o serie de avantaje şi limitări ale soluției CUDA în procesarea datelor precum și oportunitatea dezvoltării de soluții în CUDA pentru optimizarea procesării datelor.

În **Capitolul 3**, "Soluții de îmbunătățire a performanței funcțiilor algoritmice de însumare prefixată și însumare segmentată în cadrul arhitecturilor ***Kepler*** și ***Maxwell***", sunt prezentate pentru început însumarea paralelă prefixată și însumarea paralelă segmentată, apoi proiectarea unei funcții algoritmice eficiente de însumare prefixată în CUDA, proiectarea unei funcții algoritmice eficiente de însumare paralelă segmentată în CUDA, tehnici de îmbunătățire a performanței funcțiilor algoritmice de însumare prefixată și însumare segmentată în CUDA, rezultate experimentale și analiza performanței funcțiilor algoritmice de însumare prefixată și însumare segmentată, concluzii privind funcția algoritmică de însumare prefixată și însumare segmentată în CUDA.

În cadrul **Capitolului 4,** "Soluții de îmbunătățire a performanței funcției algoritmice de reducere paralelă în cadrul arhitecturilor ***Kepler*** și ***Maxwell***", sunt prezentate elemente referitoare la reducerea paralelă, la proiectarea funcției algoritmice de reducere paralelă în CUDA, apoi tehnici de îmbunătățire a