NUMERIKUS MÓDSZEREK PÁRHUZAMOSÍTÁSA PÁRHUZAMOS PROCESSZOR ARCHITEKTÚRÁKON

László Endre

Doktori disszertáció tézisei

Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar
Roska Tamás Műszaki és Természettudományi
Doktori Iskola

Témavezető:
Prof. Dr. Szolgay Péter D.Sc.

Budapest, 2015

1. Bevezetés

Ma minden tudományos, műszaki és pénzügyi terület fejlődését - ami numerikus számítási módszereken alapul -, nagymértékben behatárolja a számítógépek számítási kapacitásának stagnálása. Ez a VLSI (Very Large Scale Integration) technológia fizikai korlátainak következménye, ami miatt a CPU-k (Central Processing Unit) egy processzor magra jutó számítási teljesítménye a hatalmas hőleadás miatt nem növelhető tovább. Ezen fizikai korlát miatt 2004 körül szükségessé vált az egy szilícium hordozón implementált, több magos processzorok megalkotása. Ennek a több-magos párhuzamosításnak a következményeként jött létre a numerikus módszerek és algoritmusok párhuzamosításának problémája.

A doktori munkám elsődleges célja numerikus módszerek és algoritmusok futásidejének csökkentése új algoritmusok és megoldások kidolgozásával különböző, modern, párhuzamos processzor architektúrák használatával.

Fizikai korlátok Gordon Moore az 1960-as években CMOS (Complementary Metal-Oxide-Semiconductor) technológiával gyártott integrált áramkörök (IC - Integrated Circuit) gyártásának fejlődését tanulmányozta. 1965-ben mindössze öt darab legyártott IC-re alapozva azt a következtetést vonta le, hogy az egységnyi felületre jutó tranzisztorok száma évente fog duplázódni. Ez a megfigyelés még fél évszázad távlatában is lényegében érvényes, azzal a változással, hogy a tranzisztorok száma már csak két évente duplázódik.

E törvény különösen érdekes, mivel 2004 környékén a gyártási technológia elérte a hődisszipáció fizikai határát, ami a digitális áramkörök órajelének növelését korlátozza. A néhány négyzetcentiméter felületű szilícium hordozó felületéről disszipált hő ma 100 Watt nagyságrendjében van. Ez az abszolút felső korlát (TDP - Thermal Design Power), amire egy processzort tervezni lehet. A folyamatosan csökkenő VLSI (Very Large Scale Integration) áramköri elem méretek miatt: 1) a vezetők ellenállása (és impedanciája) növekszik; 2) a parazita kapacitások növekednek; 3) a CMOS tranzisztorok gate oxidjain folyó szivárgási áram növekszik. Az ellenálláshoz és kapacitáshoz kapcsolódó, növekvő paraméterek korlátozzák az áramkörök órajelének növelését. Az órajel növelése ilyen paraméterek mellett csak az áram növelésével lehetséges, ami további hőtermelést eredményez. Ugyanez a fizikai korlát akadályozza az adatátviteli sebesség további növelését is.

Egy digitális áramkör tervezett hődisszipációja a következő: $P_{TDP} =$

$P_{DYN} + P_{SC} + P_{LEAK}$, ahol P_{DYN} a dinamikus kapcsolásból eredő teljesítmény veszteség, P_{SC} a pillanatnyi rövidzárból eredő hőveszteség (a digitális jelek éleinek nem nulla felfutási és lefutási sebessége miatt) és P_{LEAK} a szivárgási áramból eredő veszteség. A legdominánsabb teljesítmény disszipációs tényező a dinamikus kapcsolásból eredő $P_{DYN} \propto CV^2 f$ tag, ahol $\propto$ az arányosságot jelöli, f az órajel, C az áramkörön terhelt kapacitás és V a kapacitásra kapcsolt feszültség.

Párhuzamosítás Korábban a számítógépek folytonos fejlődésének indikátora a tranzisztorok számának és a processzor órajelének a növelése volt. Ma a fókusz szigorúan a számítási kapacitás növelésére korlátozódik, aminek kulcsa az új megoldásokban rejlik és kevésbé az órajel növelésében. Az egyik megoldás a processzorok több szintű párhuzamosítása, ami a szilícium hordozóra integrált processzor magok és tranzisztorok számának növelését jelenti. Az eszközök párhuzamosítása és specializálása növeli a hardware, algoritmus és software komplexitását. Ennek következtében 2004 környékén új processzor architektúrák jelentek meg, amin egy szilícium hordozóra több processzor magot integráltak. Ezzel egyidőben elindult az architektúrák párhuzamosítása a legalacsonyabb szinteken is. Mivel a számítógép rendszerekben a párhuzamosításnak számos szintje jelent meg, hatékony párhuzamos algoritmusok kidolgozásánál elengedhetetlen ezeknek a architektúrális paradigmáknak a mély ismerete.

Amdahl törvénye Gene Amdahl egy 1967-es konferencián publikálta megfigyelését a több processzoros rendszereken - az egy processzor teljesítményéhez képest -, elérhető gyorsulással kapcsolatban [1]. Ezt az eredményt az (1) egyenlet foglalja össze, amit ma Amdahl törvényként ismerünk. Amdahl törvénye kimondja, hogy egy feladat megoldása $S(N)$-szeresére gyorsul, ha a feladat P hányadát N identikus processzorra osztjuk szét.

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}} \tag{1}$$

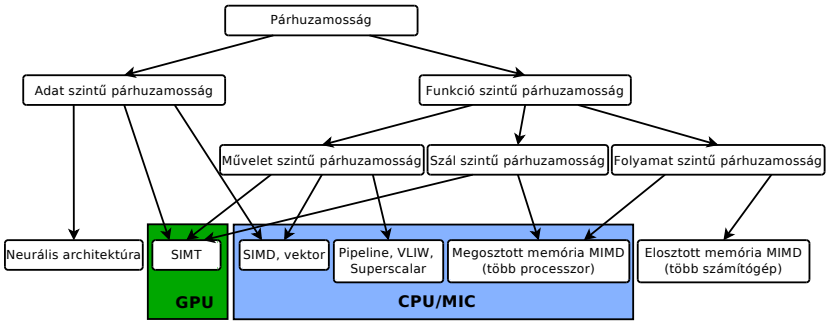
P hányad a feladat azon része, ami párhuzamosítható egy algoritmussal (és implementációval). Minél nagyobb ez a hányad, annál jobb a feladat párhuzamos skálázódása. Ezt hívják erős skálázódásnak. Ez egy különösen fontos koncepció, aminek a következményei impliciten megjelennek a párhuzamosítással kapcsolatos érvelésekben.

Gustafson törvénye John Gustafson 1988-ban újragondolta Amdahl törvényét [2] és azt állapította meg, hogy gyenge skálázódás esetén a gyorsulást a (2) egyenlet adja meg. A gyenge skálázódás esetén a futási idő mérését N processzoron a probléma méretének N-szeresére növelésével mérik. A (2) egyenlet alapján a futási idő ($S_{latency}(s)$) csökken, amikor a probléma P részének megoldásából eredő késleltetés - ami párhuzamosítható -, s-ed részére csökken.

$$S_{latency}(s) = 1 - P + sP \tag{2}$$

2. Párhuzamosság osztályozása

A 1. ábra a párhuzamosítás elméleti osztályozásának minden szintjét mutatja. A blokkvázlat alsó sora azokat a hardware és software komponenseket mutatja, amik a különböző processzor architektúrákon meg vannak valósítva. A GPU (Graphics Processing Unit) valamint CPU/MIC-ként (MIC - Many Integrated Core) feltüntetett processzor és software tulajdonságok azok a komponensek, amik ebben a munkában fel lettek használva.



1. ábra. Párhuzamosság osztályozása valamint a párhuzamos számítógép és processzor architektúrák, amik megvalósítják ezeket.

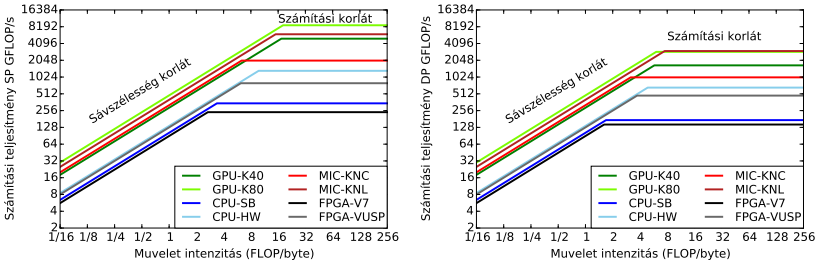
Architektúra függő teljesítmény korlátok A tervezési céloknak megfelelően a különböző párhuzamos processzor architektúrák számítási teljesítménye a konkrét megoldandó problémától függ. A teljesítményt a feladat megoldása szempontjából kritikus processzor erőforrás hiánya

korlátozhatja, mint például több lebegőpontos számítási egység, nagyobb memória sávszélesség stb. Egy implementációt ezért nevezhetünk: 1) *számítás korlátozottnak*, ha a futási teljesítmény csak a számítási kapacitás növelésével lehetséges vagy 2) *memória sávszélesség korlátozottnak*, ha a futási teljesítmény csak a memória sávszélesség növelésével lehetséges. A „roofline" modell [3] segít annak a meghatározásában, hogy az adott algoritmus vagy implementáció futási teljesítménye az adott processzoron a számítási egység vagy a memória vezérlő erőforrásai korlátozzák. Ez a modell úgyszintén segít az architektúra kihasználtságának meghatározásában. A 2. ábra összehasonlít számos, a disszertációban használt valamint azóta megjelent vagy megjelenés előtt álló párhuzamos processzor architektúrát a roofline modell alapján.

A 2. ábrán a szakaszok az adott architektúra maximális, elméleti számítási kapacitás (GFLOP/s) és adat sávszélesség (GB/s) metrikák alapján lettek meghatározva. FPGA esetén a megvalósítható szorzó áramkörök száma valamint a megvalósítható memória interfészek száma és órajele adta a számítás alapját. A feltüntetett processzor architektúrák pontos típusai: 1) GPU-K40: NVIDIA Tesla K40m kártya Kepler GK110B mikroarchitektúrával, „Boost" órajel beállítással; 2) GPU-K80: NVIDIA Tesla K80 kártya Kepler GK210 mikroarchitektúrával, „Boost" órajel beállítással; 3) CPU-SB: Intel Xeon E5-2680 CPU Sandy Bridge mikroarchitektúrával; 4) CPU-HW: Intel Xeon E5-2699v3 CPU Haswell mikroarchitektúrával; 5) MIC-KNC: Intel Xeon Phi 5110P koprocesszor kártya Knights Corner mikroarchitektúrával; 6) MIC-KNL: Intel Xeon Phi koprocesszor Knights Landing mikroarchitektúrával - a konkrét terméktípus még nem lett bejelentve; 7) FPGA-V7: Xilinx Virtex-7 XC7VX690T; 8) FPGA-VUSP: Xilinx Virtex UltraScale+ VU13P.

Mivel a doktori disszertációm célja a futási idő csökkentése - különböző numerikus számítási problémák esetén -, a megoldók futásidejének optimalitását - ahol lehetséges -, a roofline modell alapján vizsgáltam.

Párhuzamos processzor architektúrák és nyelvek A HPC-ben (High Performance Computing) a tudományos problémák megoldásához különböző típusú párhuzamos architektúrákat használnak és kísérleteznek velük. A többmagos (multi-core) Intel Xeon szerver processzorok a legelterjedtebbek napjainkban a HPC területén. A korábban csak grafikai alkalmazásokra használt GPU-kat mára széles körben használják bizonyos tudományterületeken. Az elmúlt években mutatta be az Intel a MIC (Knights Corner mikroarchitektúra) architektúráját, ami a Xeon Phi koprocesszor kártyákon érhetőek el. Az IBM a POWER és a Fujitsu a



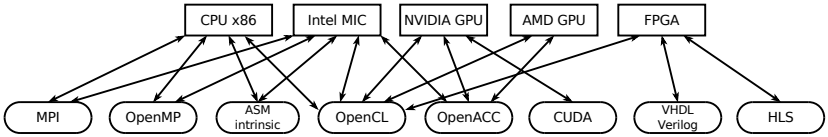
2. ábra. Roofline modell a párhuzamos processzor architektúrák összehasonlításához. Az ábrán az SP (Single Precision) a egyszeres lebegőpontos pontosságot jelenti, a DP (Double Precision) a kétszeres lebegőpontos pontosságot jelenti.

SPARC architektúrájú processzoraikkal a HPC-re fókuszálnak. A Xilinx és Altera által gyártott FPGA (Field Programmable Gate Array) néhány különleges probléma megoldásában használhatók nagy hatékonysággal. Az ARM, mint IP-t (Intellectual Property) előállító vállalat kísérleti célú, x86 alapú tervekkel látja el a HPC közösséget. A CPU-k teljesítmény hatékonysága és a gyorsító processzorok programozhatóságának nehézségei szükségessé tette a kutatást új, heterogén rendszerek kidolgozásának irányába.

A FORTRAN, C/C++ és Python programozási nyelvek már nem elegendők a multi- és many-core architektúrák párhuzamosságának produktív módon történő kiaknázásához. Ebből fakadóan az elmúlt években új programozási nyelvek, nyelvi kiterjesztések, keretrendszerek és DSL (Domain Specific Language) nyelvek jelentek meg. A teljesség igénye nélkül, a legfontosabbak: 1) CUDA (Compute Unified Device Architecture) a C99 programozási nyelv kiterjesztett változata NVIDIA GPU-k programozására; 2) OpenMP (Open Multi Processing) direktíva alapú nyelvi kiterjesztés multi-core CPU és many-core MIC architektúrák többszálúsításához; 3) OpenCL (Open Compute Language) nagyfokú párhuzamosítást és forráskód hordozhatóságot támogató nyelv; 4) AVX (Advanced Vector eXtension) és IMCI (Initial Many Core Instruction) vektorizált, SIMD (Single Instruction Multiple Data) ISA (Instruction Set Architecture) műveletkészlet és „intrinsic” műveletek a megnövelt ILP (Instruction Level Parallelism) párhuzamossága számára; 5) OpenACC (Open Accelerators) direktíva alapú nyelvi kiterjesztés gyorsító processzor architektúrák programozásához; 6) HLS (High Level Synthe-

sis) a Xilinx innovációja az FPGA általános programozási célú fejlesztési idejének csökkentése érdekében.

Mindezek az architektúrális és programozási újítások új lehetőségeket nyújtanak a már meglévő párhuzamosítási problémák megoldására, de egyikük sem biztosítja a magas fejlesztési produktivitást, forráskód és teljesítmény hordozhatóságot egy megoldásként. A HPC közösség számára ezek a kérdések állnak a folyó kutatási témák fókuszában.



3. ábra. Processzor architektúrák, programozási nyelvek és nyelvi kiterjesztések közötti kapcsolatok.

Tudományos problémák párhuzamosíthatóság szerinti osztályozása A kutatásra kiválasztott problémák az "A view of the parallel computing landscape" [4] tanulmány 13 „törpéjén" („dwarves") alapul. Az eredményeim a következő lista kiemelt problémaköreibe tartoznak:

1. Sűrű mátrixok
2. **Ritka mátrixok**
3. Spektrális metódusok
4. N-test metódusok
5. **Strukturált térhálók**
6. **Nem-strukturált térhálók**
7. MapReduce
8. Kombinációs logika
9. Gráf bejárás
10. Dinamikus programozás
11. Backtrack és Branch-and-Bound
12. Grafikai modellek
13. Véges állapotú gépek

3. Kiválasztott és párhuzamosított numerikus problémák

A kiválasztott numerikus algoritmusok lefedik a numerikus matematika azon területeit, amik a PDE-k (Parciális Differenciál Egyenlet) megoldásával foglalkoznak különböző mérnöki területeken, mint a CFD

(Computational Fluid Dynamics), pénzügyi számítások, elektromágneses hullámterjedés szimulációk vagy képfeldolgozás.

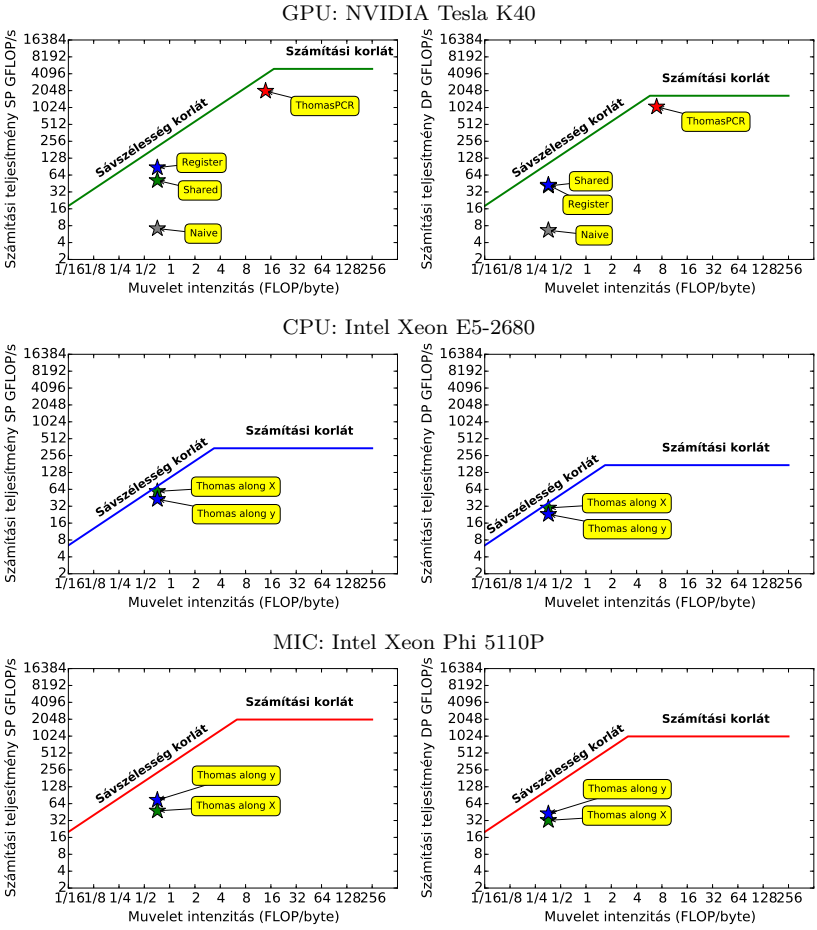
3.1. Tridiagonális egyenletrendszerek

Mérnöki, tudományos és pénzügyi alkalmazások sokszor igénylik nagy számú, független, változó paraméterű tridiagonális egyenletrendszer egyidejű megoldását [5, 6, 7, 8, 9, 10]. Tridiagonális egyenletrendszerek megoldása emellett felmerül „multi-grid" megoldók részeként a „line-implicit smoother" megoldókban [11], magas rendű kompakt differenciálásban [12, 13]. Mivel az egyenletrendszerek száma kellően nagy ahhoz, hogy elegendő párhuzamosításra adjon lehetőséget many-core architektúrák esetén, a különböző tridiagonális egyenletrendszer megoldó algoritmusok - mint például a Thomas, CR (Cyclic Reduction) vagy PCR (Parallel Cyclic Reduction) -, választását újra meg kell vizsgálni. A disszertációmban olyan, közel optimális megoldókat dolgoztam ki és implementáltam, amik a skaláris és blokk tridiagonális egyenletrendszereket hatékonyan oldják meg CPU, Intel MIC, és NVIDIA GPU architektúrákon. A cél eléréséhez a memória adatátviteli sebességének maximalizálása mellett a kommunikáció minimalizálása új algoritmusokkal történik, lokális, „register blocking" metódussal. Ez egyben azt is jelenti, hogy az elért számítási kapacitás is maximalizálva van (lásd. 2 szakasz és 4. ábra), mivel a tridiagonális egyenletrendszerek megoldása memória sávszélesség korlátozott a kis számú lebegőpontos művelet miatt.

A 4. ábrán a számítási kapacitás felső korláta az adott processzor elméleti maximális számítási kapacitása (GFLOP/s) és adat sávszélesség (GB/s) metrikák alapján lett meghatározva. A művelet intenzitás a memória buszon keresztül betöltött adatok mennyiségéből és a rajtuk végzett lebegőpontos műveletek számából van meghatározva. Az elvégzett lebegő pontos műveletek száma az algoritmus alapján van meghatározva, ami a futásidőre vetítve megadja a GFLOP/s számítási teljesítményt. A művelet intenzitás és a GFLOP/s számítási teljesítmény metrikák megoldónként van kiszámolva és csillaggal jelölve az ábrán.

A legtöbb esetben a tridiagonális egyenletrendszereket olyan numerikus problémákra alkalmazzák, ahol a rács pontjai egy ismeretlent tartalmaznak. Folyadékdinamikai számításoknál előfordulnak olyan problémák, ahol a rács pontjai kevesebb, mint 8 ismeretlent tartalmaznak [7]. Ilyen numerikus problémák esetén alakulnak ki blokk tridiagonális egyenletrendszerek. A CFD szimulációkban használt blokk tridiagonális megoldók ezért úgyszintén a disszertáció részét képezik egy új munka-

megosztáson és register blocking megoldáson alapuló GPU megoldóval együtt.



4. ábra. A skalár tridiagonális megoldókra alkalmazott roofline modell eredményei GPU, CPU és MIC processzor architektúra esetén. Megjegyzés: SP (Single Precision) - egyszeres pontosság, DP (Double Precision) - kétszeres pontosság.

3.2. Alternating Directions Implicit metódus

Többdimenziós PDE numerikus megoldása strukturált térrácson sokszor igényli nagy számú tridiagonális egyenletrendszer megoldását. Mérnöki alkalmazásokban és pénzügyi számítások esetén ezt a megoldót, az ADI metódus részeként használják, ami egy népszerű diszkretizálási módszer [10]. Az ADI metódus szükségessé teszi ezeknek a megoldóknak a használatát többdimenziós értelmezési tartomány minden egyes dimenziója mentén [14, 9, 15, 16].

3.3. Celluláris Neurális Hálózat

A CNN (Cellular Neural Network) [17] egy hatékony képfeldolgozó architektúra, aminek hardware implementációja különösen nagy sebességű feldolgozást tesz lehetővé [18, 19]. Egy ilyen eszköz hiánya azonban egy fejlesztési folyamatban helyettesíthető egy hatékony szimulátor implementációval. Egy CNN szimulátor NVIDIA Fermi architektúrájú GPU implementációja képes ezt a feladatot ellátni. A általam kifejlesztett CNN szimulátor különböző implementációs megközelítések figyelembe vételével lett megalkotva. Az eredményeket összehasonlítottam egy párhuzamosított CPU és egy korábbi GPU implementációval.

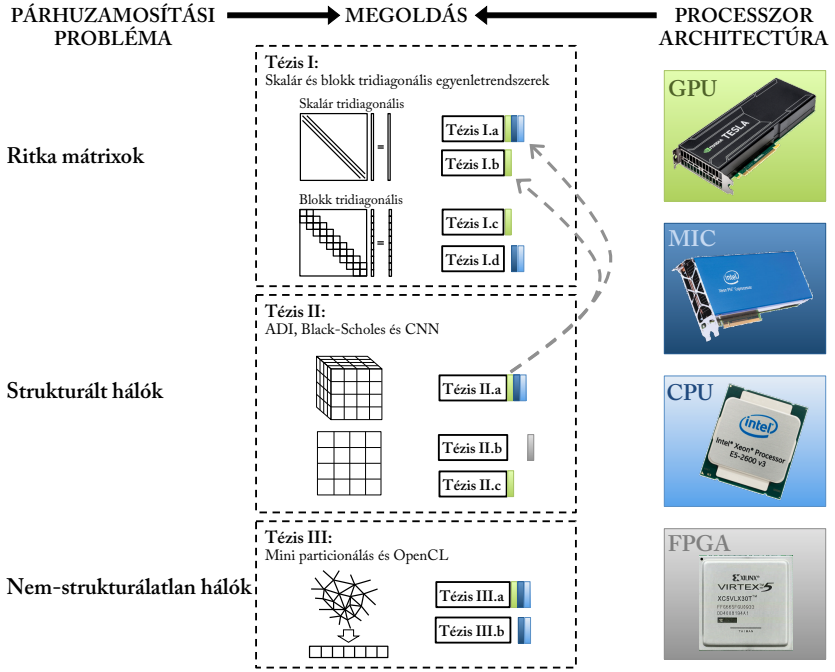
3.4. Numerikus folyadékdinamikai számítások

Optimális számítási teljesítmény elérése a legkorábbi multi- és manycore architektúrákon egyre inkább a vektor műveletvégző egységek hatékonyabb kihasználásán múlik. A napjainkban elérhető fordítók automatikus vektorizálási lehetőséget biztosítanak a vektor műveletek explicit használata nélkül. Azonban ezek számos esetben csak néhány jól definiált probléma esetén működnek hatékonyan, amikor a memória elérési és a számítási mintázat jól strukturált. Más probléma osztályok számára a CUDA és OpenCL által kínált párhuzamos programozási modell megoldás lehet GPU-k esetén. CPU és MIC esetén viszont nem egyértelmű, hogy milyen programozási paradigma a célravezető a vektor műveletvégző egységek hatékony kihasználásához. Kérdés ezért, hogy ezeken az alapvetően SIMD (Single Instruction Multiple Data) architektúrákon a SIMT (Single Instruction Multiple Threads) programozási modell hogyan teljesít. Számos problémaosztály esetén az OpenCL SIMT model nem képezhető le hatékonyan a CPU vektor műveleteire. Disszertációmban megoldásokat adok vektorizálás kivitelezésére és szinkronizálási

pontok elkerülésére numerikus folyadékdinamikai szimuláció gyorsításához az OP2 keretrendszeren belül.

4. Új tudományos eredmények

Az új tudományos eredmények a 13 „dwarf”-nak megfelelően három tézis csoportba vannak sorolva. Az első tézis csoport ritka mátrixok kategóriájának megfelelően a tridiagonális egyenletrendszerek megoldásával kapcsolatos eredményeket foglalja össze. A második tézis csoport strukturált térhálók témakörébe tartozó képfeldolgozással, az ADI metódussal kapcsolatos PDE megoldásokat foglalja magában. A harmadik tézis csoport a nem-strukturált térhálók témakörébe tartozó numerikus folyadékdinamikai számításokhoz kapcsolódó eredményeket tartalmazza. Az 5. ábra a tézis csoportok felépítését szemlélteti.



5. ábra. Tézis csoportok: a tézis csoportok osztályozása és a párhuzamos processzor architektúrák közötti kapcsolatok. A tézisek számozásától jobbra a színes, függőleges oszlopok a tézisben felhasznált processzor architektúrákat jelölik. A szaggatott szürke nyíl a tézisek egymáshoz viszonyított kapcsolatát jelöli.

Az új tudományos eredmények nemzetközi folyóiratokban („[J]”-vel jelölve), konferencia kiadványokban („[C]”-vel jelölve) és konferencia beszédekben („[CT]”-vel jelölve) lettek publikálva. A publikációk listája lentebb, a kapcsolódó tézis csoportoknál vannak jelölve.

Tézis I. Ritka mátrixokon értelemzett lineáris algebrai feladatok hatékony algoritmusainak kifejlesztése és megvalósítása

A tudományos, mérnöki és pénzügyi területen használt PDE-k numerikus megoldása sokszor veszi igénybe a ritka mátrixok eszközeit, amit hatékonyan kell párhuzamosítani a jelenlegi és jövőbeli párhuzamos processzor architektúrák számára. Sok esetben a parabolikus, diffúzió típusú PDE-k numerikus megoldása implicit diszkretizálással, tridiagonális egyenletrendszerek megoldására bontható le. Az ezt leíró tridiagonális mátrix elemei vagy skalár értékűek vagy $M \times M$ méretű blokkok, ahol $M \in [2, 8]$ nagyságrendileg. A tudományos, mérnöki és pénzügyi kutatási területek fejlődése számára ezért szükségesek olyan új, párhuzamos algoritmusok, amik ezeknek a tridiagonális egyenletrendszereknek a megoldását gyorsítják. E csoportba sorolt tézisek ezeknek a metódusoknak a párhuzamosításához és gyorsításához járulnak hozzá.

A tézishez tartozó publikációk: [J1], [C1], [CT1], [C2], [C3]

Tézis I.a *Kifejlesztettem egy új, „register blocking” alapú, lokális adattranszponáláson alapuló algoritmust multi- és many-core párhuzamos processzor architektúrákra, hogy javítsam a Thomas algoritmus memória elérési mintázatát olyan tridiagonális egyenletrendszerek megoldása esetén, amiknek az együtthatói a memóriában egymás utáni sorrendben vannak tárolva. A teljes sebességnövekedés: 1) akár* 4.3*-szeres a GPU-n, a NVIDIA cuSPARSE függvény könyvtárához viszonyítva; 2) akár* 1.5*-szörös a CPU-n, az Intel MKL függvény könyvtárához viszonyítva; 3) akár* 1.9*-szeres a MIC-en az Intel MKL függvény könyvtárához viszonyítva.*

Egy tridiagonális egyenletrendszer az **a**, **b** és **c** együttható vektorokból, az **u** ismeretlen vektorból és a **d** jobb oldali együttható vektorból áll. Mindezek a vektorok az $\mathbb{R}^N$ halmaz elemei és ugyanúgy vannak leképezve a memóriába. Egy tridiagonális egyenletrendszer együtthatói a memóriában vagy egymást követő vagy N távolságra lévő memóriacímeken vannak, attól függően, hogy milyen numerikus módszer része és

milyen térbeli rács esetén lett alkalmazva. Az egymást követő leképezésnél az **a** vektor a_k és $a_{(k+1)}$ eleme egymást követő k és $(k+1)$ memória címen helyezkednek el. Az N-el csúsztatott leképezésnél a két egymást követő a_k és $a_{(k+1)}$ együttható a $k \times N$ és a $(k+1) \times N$ memória címen helyezkedik el. Sok esetben az adat egymást követő leképezéssel van tárolva, ami alacsony cache-line kihasználáshoz vezet és ezért szükséges a memória elérési mintázat optimalizálása. Kifejlesztettem két algoritmust GPU architektúrákra, amik egyszerre több érték betöltési (és tárolási) műveletét végzik el, majd ezen adaton végrehajtják a szükséges numerikus műveleteket. CPU és MIC architektúrákra is kifejlesztettem hasonló, transzponáláson alapuló tridiagonális egyenletrendszer megoldót. A megoldásom lehetővé teszi, hogy ez a lokális adat transzponálás a műveletek elvégzésével együtt regiszterekben történjen meg.

Tézis I.b *Kidolgoztam egy új, a Thomas és PCR algoritmusok hibridjének egy hatékony implementációját tridiagonális egyenletrendszerek megoldására, amelyeknél az együtthatók egyenletrendszerenként változnak és a globális memóriában vannak tárolva. Ez a GPU alapú megoldás kihasználja az architektúra nagy méretű regiszter memóriáját úgy, hogy az adatokat a globális memóriából, regiszterekben történő lokális adattranszponálással olvassa. Az eredmény egy olyan megoldó, ami akár* 9*-szer gyorsabb, mint az NVIDIA cuSPARSE függvénykönyvtárában elérhető megoldó, és akár* 2.1*-szer gyorsabb, mint a legoptimálisabb, transzponálás alapú GPU megoldó, amit az I.a tézisben mutattam be.*

A Thomas és PCR algoritmusok keresztezésével létrehozott új algoritmus hatékony implementációját készítettem el, ami GPU architektúrákon közel optimális futásteljesítményt eredményez és kihasználja az architektúra maximális memória sávszélességét, vagy számítási kapacitását a lebegőpontos aritmetika pontosságának függvényében. Egymás utáni vagy N távolságokon lévő memória címeken elhelyezett együtthatókat egyaránt képes a megoldásom hatékonyan kezelni. Amikor szükséges, az adatok a regiszterekben, a „shared" memória használata nélkül, lokálisan transzponálhatók. Ez a hibrid algoritmus a számítás köztes eredményeit - kellően kisméretű rendszerek esetén -, nem tárolja az off-chip globális vagy lokális memóriában, mint ahogy azt a konvencionális, Thomas algoritmus teszi. A lebegőpontos aritmetikai műveletek és betöltött byteok aránya a problémát számításkorlátozottá teszi, ami hatékony futási teljesítményt eredményez GPU-k esetén.

Tézis I.c *Kifejlesztettem egy új, szál kooperációs megoldást és „register blocking" alapú algoritmust blokk tridiagonális egyenletrendszerek GPU-n történő megoldására, ahol a blokkok mérete $M \times M$, ahol $M \in [2, 8]$. Ennek a GPU specifikus szál munkamegosztáson alapuló megoldásnak a számítási teljesítménye felülmúlja más algoritmusok és implementációk teljesítményét. Kísérletileg igazoltam, hogy ez az implementáció akár* 27*-szer gyorsabb, mint az Intel MKL sávmátrixos megoldója CPU-n, és akár* 9.8*-szor gyorsabb, mint egy GPU-ra optimalizált PCR megoldó [20].*

Blokk tridiagonális egyenletrendszereket bizonyos numerikus folyadékdinamikai számításokban használnak [8, 7], ahol az alkalmazott, több változós PDE határozza meg a blokk méretét. Adtam egy, a szálak közötti munkamegosztáson alapuló párhuzamosított algoritmust a blokk Thomas algoritmus megvalósítására. Azt is megmutattam, hogy ez a megközelítés számítás szempontjából hatékonyabb, mint a CR vagy PCR algoritmusok és hogy ezzel a megoldással kellő mértékű párhuzamosság kiaknázható, ami futásidőben felülmúlja a CR és PCR implementációkat $M \in [2, 8]$ esetén.

Tézis I.d *Kidolgoztam a blokk tridiagonális egyenletrendszerek megoldására egy vektorizált implementációt, ami multi-core CPU és many-core MIC esetén felülmúlja az Intel MKL függvény könyvtár sávmátrix megoldó számítási teljesítményét. CPU esetén* 6.5*-szörös, MIC architektúra esetén* 5*-szörös sebességnövekedést értem el ezzel a megoldással.*

C++ template-ek és kódtranszformációs technikákkal úgy strukturáltam át a standard blokk Thomas algoritmust, hogy jobb adatlokalitást érjek el és a fordító automatikus vektorizálási funkcióját segítsem a hatékony vektorizálás elérésben. Az eredmény a blokk Thomas algoritmus nagy futásteljesítményű, SIMD alapú implementációja CPU és MIC architektúrákon.

Tézis II. Strukturált térhálókon értelmezett számítások hatékony algoritmusainak kidolgozása és implementálása

Parabolikus, diffúzió típusú PDE megoldása strukturált térhálókon számos esetben hatékonyan megoldhatók az ADI metódus használatával, ami magasabb dimenziós értelmezési tartományon több, a különböző dimenziók mentén értelmezett tridiagonális egyenletrendszer megoldására

egyszerűsödik. A különböző dimenziók mentén történő egyenletrendszerek megoldása standard numerikus függvénykönyvtárak használatával sok esetben csak jelentős futásteljesítmény csökkenéssel végezhető el az eltérő memória elérési mintázatok miatt. Ez teszi szükségessé hatékony algoritmusok kifejlesztését. Úgyszintén új algoritmusokra van szükség, amikor a pénzügyi számításokban a Black-Scholes PDE numerikus megoldását vagy a CNN alapú képfeldolgozáshoz kapcsolódó PDE megoldását kell kiszámolni és hatékonyan párhuzamosítani FPGA és GPU architektúrákon.

A tézishez tartozó publikációk: [J1], [CT1], [C3], [C6]

Tézis II.a *Kidolgoztam és implementáltam új, párhuzamos algoritmusokat, melyek a magas dimenziójú értelmezési tartományon definiált PDE-k numerikus megoldását gyorsítják az ADI metódus hatékony kiszámításával CPU, MIC és GPU architektúrákon. A párhuzamosított ADI megoldó új tridiagonális egyenletrendszer megoldókat használ a memóriában stride-1 és stride-N memória elérési mintázat esetén egyaránt.*

Parabolikus, diffúzió típusú N dimenziós PDE megoldása az ADI metódus használatával szétcsatolható N különálló egy dimenziós probléma megoldására. Kifejlesztettem új megoldásokat, hogy az egydimenziós problémák megoldását különböző dimenziók mentén, a változó memória elérési mintázatnak megfelelően hatékonyan lehessen megoldani. Ez az eredmény az I.a és I.b tézisek reprezentatív, felhasználási példája.

Tézis II.b *HLS (High Level Synthesis) használatával kifejlesztettem egy teljesítmény-hatékony, párhuzamos, FPGA alapú PDE megoldót, ami az egy faktoros Black-Scholes PDE megoldását gyorsítja jelentős hődisszipáció csökkentése mellett. A kifejlesztett FPGA alapú megoldó számítási teljesítményben a CPU-val megegyező képességű, azonban összehasonlítva a számítás hatékonyságokat (GFLOPS/W), az FPGA* 4*-szer hatékonyabb az explicit megoldó és* 1.3*-szor hatékonyabb a Thomas algoritmus alapján készült implicit megoldó esetén.*

Az egy faktoros Black-Scholes PDE-t olyan derivatív pénzügyi termékének árazására használják, ahol a származtatott termék egy másik termék árfolyam mozgását követi. Az egy faktoros Black-Scholes PDE megoldás történhet explicit vagy implicit megoldóval. FPGA-ra kifejlesztettem egy új, stencil alapú megoldót, ami az explicit, előrelépő számítást hatékonyan gyorsítja. Kidolgoztam továbbá egy Thomas algoritmus alapú, teljesítmény-hatékony implicit megoldót.

Tézis II.c *Kísérletileg igazoltam, hogy a CNN állapotegyenlet megoldása textúra memória és a dupla pufferes (double buffer) módszerek alkalmazásával hatékony megoldást biztosít GPU architektúrákon. A textúra egységek a két dimenziós, térbeli adat-lokalitásra specializált cache memóriával és a beépített, egész számokon végzett, index számolási műveletekre specializált egységeivel erőforrásokat takarít meg.*

A CNN állapotegyenlet megoldása egy diffúziós képfeldolgozási műveletként egy hőterjedési PDE egyenlet megoldását végzi el. Egy ilyen művelet kiszámítása memória sávszélesség korlátozott. A „tiling" és „cache blocking" típusú optimalizálások csökkentik az adat-átviteli igényt, így javítva a futási időt. Megmutattam, hogy ez a teljesítmény is felülmúlható a GPU beépített textúra cache megfelelő használatával. Ezzel a megoldással a két dimenziós CNN állapotegyenlet megoldása jelentős cache találati rátát tud elérni az adatok betöltésekor. Ugyanakkor, jelentős mennyiségű, egész számokon végzett művelet takarítható meg a textúra egység koordináta számoló egységeivel.

Tézis III. Nem-strukturált térhálókon értelemezett számítások hatékony algoritmusainak kidolgozása és megvalósítása

Az OP2 keretrendszerrel nem-strukturált térhálókon végzett párhuzamos számítások hatékonyságának a növelése a tudományos közösség számára nagy értékkel bír. Változók indirekt elérésen keresztüli értékeinek inkrementális változtatásának jól skálázódó párhuzamosítása egy számítógépen a teljesítményt jelentősen befolyásoló tényező. Egyik komponense a skálázhatóságnak a mini-partíciók létrehozása. A jelenlegi, mini-partíciók létrehozásának egyszerű algoritmusának hatékonysága és a mini-partíciók színezése nagyban javítható mátrix sávszélesség minimalizáló módszerek használatával. Az OpenCL segítségével bevezetett, új párhuzamosítási módszerek jelentősen javítják az OP2 keretrendszer teljesítményét és a vele készült forráskód hordozhatóságát.

A tézishez tartozó publikációk: [C4], [J2], [C5]

Tézis III.a *Kísérletileg igazoltam, hogy valós geometriákon futtatott folyadékdinamikai számítások esetén javítható a változók értéknövelése indirekt memóriaelérés esetén az OP2 mini-particionálásának javításával a GPS (Gibbs-Poole-Stockmeyer) mátrix sávszélesség minimalizáló algoritmus használatával. Ez az eljárás nagy mértékben csökkenti a blokk*

színek számát a párhuzamos, indirekt elérésű műveleteknél, ami növeli a párhuzamosan feldolgozható elemek számát egy blokk színen belül, így javítva a párhuzamosítást és a teljesítményt. Valós geometrián futtatott szimulációk esetén legfeljebb 37.5*-szeres blokk szín csökkenést és ezzel együtt* 7.4*-szeres futásteljesítmény javulást értem el.*

Az indirekt memóriaelérésű műveletek hatékony párhuzamosíthatóságának érdekében a nem-strukturált térhálós számításokra kifejlesztett OP2 keretrendszer kétszintű színezési eljárással azonosítja azokat az elemeket és blokkokat, amik indirekt memória elérésen keresztül írják a memóriát. A blokk színek száma meghatározza a sorosan végrehajtandó műveleteket az indirekt műveletvégzések esetén. A valós geometriákon futtatott folyadékdinamikai számítások esetén a blokk színek száma magas lehet, mivel a blokkok sok szomszédos kapcsolatot tartalmaznak. A GPS mátrix sávszélesség minimalizáló algoritmus alkalmazását javaslom a térhálók újrasorszámozásához, hogy a mini partíciók elkészítését végző naiv algoritmus alacsonyabb számú kapcsolatot alakítson ki a szomszéd blokkokkal és a színező algoritmus alacsonyabb számú színt tudjon felhasználni a blokkok színezéséhez.

Tézis III.b *Megvizsgáltam és heurisztikus forráskód transzformációs technikákat ajánlottam OpenCL vektorizálás javítására CPU és MIC architektúrákon. Az általam ajánlott megoldásnak köszönhetően az OP2 keretrendszer OpenCL-ben implementált, valós geometrián futó szimuláció kernel függvényeit nagyobb hatékonysággal vektorizálja a fordító. CPU esetén az elért futásteljesítmény megegyezik az OpenMP és MPI implementációk futási teljesítményével. MIC esetén* 1.5*-szörös teljesítmény javulást értem el az automatikusan vektorizált MPI+OpenMP implementációhoz képest.*

Kiaknáztam az Intel OpenCL implementációjának képességeit, hogy növeljem a CPU és MIC futási teljesítményét egy OpenCL alapú OP2 támogatás megvalósításával. Ennek a hatékonyságnövelésnek a kulcsa a SIMT típusú OpenCL kernel absztrakció megfelelő leképezése a CPU-n és MIC-en elérhető többszálú futtatás és SIMD vektorműveletek használatában van.

5. Az eredmények alkalmazhatósága

A kiválasztott, megoldott problémák a tudományos, mérnöki és pénzügyi számítások területéről származnak, így ezek a gyakorlatban direkt módon hasznosíthatók. A skaláris és blokk tridiagonális egyenletrendszer megoldókkal kapcsolatos eredményeket a mai napig a legnagyobb, GPUGPU témakörében szervezett eseményen, a kaliforniai San Jose-ban rendezett GPU Technology Conference 2014 konferencián mutattam be. Emellett az eredményeim más konferencia és folyóirat publikációkban jelentek meg.

A skalár tridiagonális egyenletrendszer megoldó integrálását atmoszférikus szimulációkba Eike Mueller és kollégái kezdték el tanulmányozni és használni az Egyesült Királyságban a Bathi Egyetem Matematika Tudományok Karán. Az általuk alkalmazott atomszférikus szimulációk strukturált térhálót illesztenek a vizsgált légtérre. A térháló cellák oldalainak mérete nagyságrendileg eltér a vertikális és horizontális dimenziók mentén: a cellák lapos téglatest alakúak. Ez a torzítottság numerikus stabilitási problémákat vet fel, amit jelentős anizotrópiának neveznek. E probléma kiküszöbölésére úgynevezett „line-smoothert" használnak a multigrid prekondícionáló részeként, aminek egy alap metódusa a tridiagonális egyenletrendszer megoldó. A Thomas-PCR alapú megoldónak ebben az alkalmazásban jelentős szerepe lehet a futásidő csökkentésében.

Az eredményeimet felhasználták a Turbostream nevű Navier-Stokes CFD megoldó részeként. A Turbostream szimulációs kódot Dr. Tobias Brandvik és Dr. Graham Pullan fejlesztik az Egyesült Királyságban a Cambridge-i Egyetem Whittle laboratóriumában. Ezt a szimulációs kódot turbinák és sugárhajtóművek CFD szimulációjához használják, ahol a Navier-Stokes egyenletek megoldását „line-implicit" vagy „semi-implicit" Runge-Kutta módszerrel oldják meg olyan strukturálatlan térhálókon, ahol a harmadik dimenziót a két dimenziós rács kihúzásával alakítják ki. Ezen kíhúzott vonalak mentén a blokk tridiagonális megoldón alapuló implicit megoldást kell kiszámolni, ami a feltétlen numerikus stabilitás kritériuma a jelentősen anizotróp rács cellák miatt.

Dr. Serge Guillas az Egyesült Királyság University College London kutatója és kollégái a Reguly Istvánés általam készített VolnaOP2 tsunami szimulációs kódot használta a [21] és [22] publikációk elkészítéséhez. Ez az implementáció lehetővé teszi nagy skálájú tsunami szimulációk futtatását, amikből statisztikai szempontból releváns adatokat kapnak.

A szerző publikációi

[J1] **Endre Laszló**, Mike Giles, and Jeremy Appleyard. “Manycore algorithms for batch scalar and block tridiagonal solvers (Accepted)”. In: *ACM Transactions on Mathematical Software (TOMS)* (2015).

[C1] **Endre Laszló**, Zoltán Nagy, Mike Giles, István Reguly, Jeremy Appleyard, and Péter Szolgay. “Analisys of Parallel Processor Architectures for the Solution of Tridiagonal System of Equations”. In: *International Symposium on Circuits and Systems*. Lisbon, Protugal: IEEE Press, 24-27 May 2015.

[CT1] **Endre László** and Mike Giles. “Efficient Solution of Multiple Scalar and Block-Tridiagonal Equations”. In: *GPU Technology Conference*. San Jose, CA: NVIDIA, 2014.

[C2] Mike Giles, **Endre László**, István Reguly, Jeremy Appleyard, and Julien Demouth. “GPU Implementation of Finite Difference Solvers”. In: *Proceedings of the 7th Workshop on High Performance Computational Finance*. WHPCF ’14. New Orleans, Louisiana: IEEE Press, 2014, pp. 1–8. ISBN: 978-1-4799-7027-8. DOI: 10.1109/WHPCF.2014.10. URL: http://dx.doi.org/10.1109/WHPCF.2014.10.

[C3] **Endre László**, Michael B Giles, Jeremy Appleyard, and Péter Szolgay. “Methods to utilize SIMT and SIMD instruction level parallelism in tridiagonal solvers”. In: *Cellular Nanoscale Networks and their Applications (CNNA), 2014 14th International Workshop on*. IEEE. 2014, pp. 1–2.

[C6] **Endre László**, Péter Szolgay, and Zoltán Nagy. “Analysis of a GPU based CNN implementation”. In: *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*. IEEE. 2012, pp. 1–5.

[C4] István Z Reguly, **Endre László**, Gihan R Mudalige, and Mike B Giles. “Vectorizing Unstructured Mesh Computations for Manycore Architectures”. In: *Proceedings of Programming Models and Applications on Multicores and Manycores*. ACM. 2014, p. 39.

[J2] I Z. Reguly, **Endre László**, Gihan R. Mudalige, and Mike B. Giles. “Vectorizing unstructured mesh computations for many-core architectures”. In: *Concurrency and Computation: Practice and Experience* (2015). ISSN: 1532-0634. DOI: 10.1002/cpe.3621. URL: http://dx.doi.org/10.1002/cpe.3621.

[C5] Mike B Giles, Gihan R Mudalige, Carlo Bertolli, Paul HJ Kelly, **Endre László**, and I Reguly. “An analytical study of loop tiling for a large-scale unstructured mesh application”. In: *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:* IEEE. 2012, pp. 477–482.

A szerző egyéb publikációi

A doktori tézishez nem szigorúan kapcsolódó publikációk

[J3] Béla Szentpáli, Gábor Matyi, Péter Fürjes, **Endre László**, Gábor Battistig, István Bársony, Gergely Károlyi, and Tibor Berceli. “Thermopile-based THz antenna”. In: *Microsystem technologies* 18.7-8 (2012), pp. 849–856.

[C7] Béla Szentpáli, Gábor Matyi, Péter Fürjes, **Endre László**, Gábor Battistig, István Bársony, Gergely Károlyi, and Tibor Berceli. “THz detection by modified thermopile”. In: *SPIE Microtechnologies* (2011).

[C8] **Endre László**, Kálmán Tornai, Gergely Treplán, and János Levendovszky. “Novel load balancing scheduling algorithms for wireless sensor networks”. In: *CTRQ 2011, The Fourth International Conference on Communication Theory, Reliability, and Quality of Service*. 2011, pp. 54–59.

[C9] Janos Levendovszky, **Endre László**, Kalman Tornai, and Gergely Treplan. “Optimal pricing based resource management”. In: *Proceedings of the International Conference on Operations Research* (2010), p. 169.

[LN1] Zoltán Nagy, Péter Szolgay, András Kiss, and **Endre László**. “GPU architektúrák”. In: *Párhuzamos számítógép architektúrák, processzortömbök*. Pázmány Egyetem eKiadó, 2015. Chap. 3, pp. 34–59.

Hivatkozásjegyzék

[1] Gene M. Amdahl. “Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities”. In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS ’67 (Spring). Atlantic City, New Jersey: ACM, 1967, pp. 483–485. DOI: 10.1145/1465482.1465560. URL: http://doi.acm.org/10.1145/1465482.1465560.

[2] John L. Gustafson. “Reevaluating Amdahl’s Law”. In: *Communications of the ACM* 31 (1988), pp. 532–533.

[3] Samuel Williams, Andrew Waterman, and David Patterson. “Roofline: An Insightful Visual Performance Model for Multicore Architectures”. In: *Commun. ACM* 52.4 (Apr. 2009), pp. 65–76. ISSN: 0001-0782. DOI: 10.1145/1498765.1498785. URL: http://doi.acm.org/10.1145/1498765.1498785.

[4] Krste Asanovic et al. “A View of the Parallel Computing Landscape”. In: *Commun. ACM* 52.10 (Oct. 2009), pp. 56–67. ISSN: 0001-0782. DOI: 10.1145/1562764.1562783. URL: http://doi.acm.org/10.1145/1562764.1562783.

[5] Yushan Wang, Marc Baboulin, Jack Dongarra, Joël Falcou, Yann Fraigneau, and Olivier Le Maître. “A Parallel Solver for Incompressible Fluid Flows”. In: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, pp. 439–448. ISSN: 1877-0509. DOI: http://dx.doi.org/10.1016/j.procs.2013.05.207. URL: http://www.sciencedirect.com/science/article/pii/S1877050913003505.

[6] Tobias Brandvik and Graham Pullan. “An Accelerated 3D Navier–Stokes Solver for Flows in Turbomachines”. In: *Journal of Turbomachinery* 133.2 (2011), pp. 021025+. DOI: 10.1115/1.4001192. URL: http://dx.doi.org/10.1115/1.4001192.

[7] Thomas H. Pulliam. “Implicit solution methods in computational fluid dynamics”. In: *Applied Numerical Mathematics* 2.6 (1986), pp. 441–474. ISSN: 0168-9274. DOI: http://dx.doi.org/10.1016/0168-9274(86)90002-4. URL: http://www.sciencedirect.com/science/article/pii/0168927486900024.

[8] Thomas H Pulliam. “Solution methods in computational fluid dynamics”. In: *Notes for the von Kármán Institute For Fluid Dynamics Lecture Series* (1986).

[9] I.J.D. Craig and A.D. Sneyd. “An alternating-direction implicit scheme for parabolic equations with mixed derivatives”. In: *Computers and Mathematics with Applications* 16.4 (1988), pp. 341–350. ISSN: 0898-1221. DOI: http://dx.doi.org/10.1016/0898-1221(88)90150-2. URL: http://www.sciencedirect.com/science/article/pii/0898122188901502.

[10] Duy M. Dang, Christina Christara, and Kenneth R. Jackson. “Parallel Implementation on GPUs of ADI Finite Difference Methods for Parabolic PDEs with Applications in Finance”. In: *Social Science Research Network Working Paper Series* (Apr. 3, 2010). URL: http://ssrn.com/abstract=1580057.

[11] Craig C. Douglas, Sachit Malhotra, and Martin H. Schultz. *Parallel Multigrid with ADI-like Smoothers in Two Dimensions*. 1998.

[12] B. Düring, M. Fournié, and A. Rigal. “High-Order ADI Schemes for Convection-Diffusion Equations with Mixed Derivative Terms”. English. In: *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM 2012*. Ed. by Mejdi Azaiez, Henda El Fekih, and Jan S. Hesthaven. Vol. 95. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2014, pp. 217–226. ISBN: 978-3-319-01600-9. DOI: 10.1007/978-3-319-01601-6_17. URL: http://dx.doi.org/10.1007/978-3-319-01601-6_17.

[13] Samir Karaa and Jun Zhang. “High order ADI method for solving unsteady convection–diffusion problems”. In: *Journal of Computational Physics* 198.1 (2004), pp. 1–9. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2004.01.002. URL: http://www.sciencedirect.com/science/article/pii/S002199910400018X.

[14] D. W. Peaceman and Jr. Rachford H. H. “The Numerical Solution of Parabolic and Elliptic Differential Equations”. English. In: *Journal of the Society for Industrial and Applied Mathematics* 3.1 (1955), ISSN: 03684245. URL: http://www.jstor.org/stable/2098834.

[15] J. Douglas and H. H. Rachford. “On the numerical solution of heat conduction problems in two and three space variables”. In: *Transaction of the American Mathematical Society* 82 (1956), pp. 421–489.

[16] Jr. Douglas Jim and JamesE. Gunn. “A general formulation of alternating direction methods”. English. In: *Numerische Mathematik* 6.1 (1964), pp. 428–453. ISSN: 0029-599X. DOI: 10.1007/BF01386093. URL: http://dx.doi.org/10.1007/BF01386093.

[17] T. Roska and L.O. Chua. “The CNN universal machine: an analogic array computer”. In: *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on* 40.3 (Mar. 1993), pp. 163–173. ISSN: 1057-7130. DOI: 10.1109/82.222815.

[18] Balázs Gergely Soós, Ádám Rák, József Veres, and György Cserey. “GPU Boosted CNN Simulator Library for Graphical Flow-based Programmability”. In: *EURASIP J. Adv. Signal Process* 2009 (Jan. 2009), 8:1–8:11. ISSN: 1110-8657. DOI: 10.1155/2009/930619. URL: http://dx.doi.org/10.1155/2009/930619.

[19] Zsolt Vörösházi, András Kiss, Zoltán Nagy, and Péter Szolgay. “Implementation of embedded emulated-digital CNN-UM global analogic programming unit on FPGA and its application”. In: *International Journal of Circuit Theory and Applications* 36.5-6 (2008), pp. 589–603. ISSN: 1097-007X. DOI: 10.1002/cta.507. URL: http://dx.doi.org/10.1002/cta.507.

[20] Christopher P Stone, Earl PN Duque, Yao Zhang, David Car, John D Owens, and Roger L Davis. “GPGPU parallel algorithms for structured-grid CFD codes”. In: *Proceedings of the 20th AIAA Computational Fluid Dynamics Conference*. Vol. 3221. 2011.

[21] Joakim Beck and Serge Guillas. “Sequential design with Mutual Information for Computer Experiments (MICE): emulation of a tsunami model”. In: *arXiv preprint arXiv:1410.0215* (2014).

[22] Dimitra Makrina Salmanidou, Aggeliki Georgiopoulou, Serge Guillas, and Frederic Dias. “Numerical Modelling of Mass Failure Processes and Tsunamigenesis on the Rockall Trough, NE Atlantic Ocean”. In: *Proceedings of the Twenty-fifth (2015) International Ocean and Polar Engineering Conference* (2015).