Heterogén feldolgozás: egy lehetséges stratégia Moore törvényének kiterjesztésére

Ebben a cikkben egy olyan kezdeményezést és a lehetséges fejlődési utakat kívánjuk bemutatni, amelyek lehetőséget teremthetnek a számítástechnikában közismert Moore törvény érvényességének meghosszabbítására.

Az alkalmazások teljesítményének folyamatos növelése olyasmi, amit mindenki akar. A nagy teljesítményű számításokkal foglalkozók körében (***High Performance Computing; HPC***) ráadásul ez nem is egyszerűen csak vágy, hanem egyenesen elvárás.
Az igazság az, hogy talán egy kicsit el is vagyunk kényeztetve ezzel az üggyel kapcsolatban, hiszen az elmúlt négy évtizedben ***Moore*** törvényének köszönhetően valóban folyamatos fejlődésnek lehettünk tanúi.
És bár ez a törvény immár negyvenedik évét tapossa, még mindig érvényesnek tűnik, vagyis a processzorok tranzisztorainak sűrűsége minden 18 hónapban megduplázódik. Persze a figyelmesebb szemlélők már észrevették, hogy gyülekeznek a viharfelhők. A tranzisztorok számának növekedése egy ideje már nem produkál hasonló eredményt a teljesítmény növekedésében. Ennek oka pedig szintén közismert: a több tranzisztor összekapcsolásához több vezetékre van szükség, ez pedig egyben több késleltetést is jelent a rendszerben. Hasonló problémát jelent a memória tartalmának gyors elérése. Ha újabb trükköket vetünk be az egymagos processzorok tervezése során, azzal elkerülhetetlenül növeljük azok összetettségét, és az általuk termelt hőt. Végezetül a skalár processzorok eleve magukban hordozzák a fejlődés gátját, hiszen működési alapelvük a gépi utasítások egymás után történő végrehajtása, ami a bonyolultság egy bizonyos fokán túl rendkívül megnehezíti az utasításszintű párhuzamosítások (***Instruction Level Parallelism; ILP***) felfedezését és kihasználását.
A fent felsorolt problémák pedig immár nem csak a felhasználók azon szűk körét érintik, akik a legnagyobb teljesítményt is képesek azonnal kihasználni, sőt, ez a bizonyos kör talán nem is volt soha szűk, épp csak a helyzet maga akadályozta a növekedését. Manapság egyre világosabb, hogy a számítási teljesítmény növelése gyakorlatilag valamennyi tudományterületre jótékony hatással lenne.
Az ***Elnöki Információtechnológiai Tanácsadó Testület (Presidents Information Technology Advisory Committee)*** éppen ezért máris megkeresett számos, a ***HPC*** területen dolgozó kutatót azzal az ötlettel, hogy 2010-re valós alkalmazások számára is elérhetővé kellene tenni a petaflop-os teljesítményt. A bizottság szerint ez elkerülhetetlen ahhoz, hogy jobb eredményeket érhessenek el az időjárás-előrejelzéssel, a gyártási folyamatok tervezésével, a gyógyszerkutatással és más, a nemzetgazdaság számára stratégiai fontossággal bíró területekkel kapcsolatban. Ugyanezt erősíti meg az a tapasztalat is, hogy az olyan szakmai konferenciákon, mint amilyen a ***Petaflops II*** az iparvállalatok fejlesztői hosszú listákat képesek összeállítani azoknak az alkalmazásoknak a neveiből, amelyeknek meglátásuk szerint kifejezetten jót tenne, ha nagyobb számítási teljesítmény állna rendelkezésre. Ezek között akadnak töréstesztek kivitelezésre alkalmas szoftverek, repülőgépek és űrjárművek tervezésére használt szimulációs eszközök, de vannak gazdasági, járványterjedési, vagy a bioterrorizmus hatásait vizsgáló modellek is.
Mindezekre az igényekre a ***HPC*** közösség olyan fejlesztési stratégiák kidolgozásával válaszol, amelyek segítségével ***Moore*** törvényének érvényessége nem csak meghosszabbítható, hanem át is hidalhatók azok az elvi akadályok, amelyek a ma használatos rendszerek korlátaiból és felépítéséből erednek.
Az alkalmazható stratégiák a következőkben foglalhatók össze:

- Olyan többmagos rendszerek építése, amelyek egy chipen több, többé- kevésbé önálló feldolgozóegységet tartalmaznak, így biztosítva a szükséges többletteljesítményt.
- Speciális processzorok alkalmazása, amelyek kiemelkedően jó teljesítményt nyújtanak az olyan különleges területeken, ahol a közönséges feldolgozóegységek rosszul teljesítenek.
- Olyan heterogén számítógépek tervezése, amelyekben a konvencionális és a specializált processzorok képesek egymással együttműködni.

Elvileg mindhárom említett megközelítés jelentős többletteljesítményt eredményezhet, ha a megfelelő területen alkalmazzák. A ***Cray*** fejlesztői mindhárom fejlődési utat lehetségesnek

tartják, így mindhárommal foglalkoznak. Ugyanakkor ami a hosszú távú lehetőségeket illeti, a kutatók úgy gondolják, hogy a heterogén feldolgozásban óriási tartalékok rejlenek. Ezzel a módszerrel nem csak egyszerűen meghosszabbítható a *Moore* törvény érvényességi ideje, hanem olyan teljesítmény érhető el, amely messze meghaladja a törvény által jósolt mértéket. A heterogén feldolgozás tehát olyasmi, ami ledönti mindazokat a korlátokat, amelyeket a hagyományos architektúrák kezdettől fogva magukban hordoznak. A *Cray* mint a *DARPA Nagy Teljesítményű Számítási Rendszerek Programjának (DARPA High Productivity Computing Systems Program)* egyik résztvevője úgy gondolja, hogy a heterogén feldolgozásnak alapvető jelentőségű szerep jut majd a következő néhány év műszaki történelmében.

A közvetlen megoldás: többmagos rendszerek

Ha egy gyártó a legegyszerűbb és leggyorsabb módszert keresi arra, miként feleljen meg termékeivel a *Moore* törvénynek, valószínűleg a többmagos rendszerek mellett fog dönteni. Kiváló példa erre az *AMD* kétmagos *Opteron* processzora.
A *Cray* természetesen szintén alkalmazza ezt a modellt, hiszen már ma is szállít kétmagos rendszereket, sőt fel van készülve az ilyen irányú továbbfejlődésre is. Ez a stratégia egyrészt azonnali növekedést okoz a felhasználó rendelkezésére álló teljesítményben, másrészt némi lehetőséget teremt arra is, hogy a hőtermelést és a fogyasztást korlátozni lehessen.
Számos alkalmazás számára, különösképpen pedig a rengeteg lebegőpontos számítást igénylők számára a többmagos processzorok azt az elsődleges „menekülési útvonalat” jelentik, amelyen keresztül Moore törvényének érvényessége fenntartható. Ugyanakkor létezik számos olyan terület is, ahol a Moore törvény egyszerű betartása már nem is elég. Ilyenek az olyan sok bitszintű műveletet, rendezést vagy jelfeldolgozást igénylő területek mint az adatbázisok kezelése, kép-, mozgókép- vagy hangfeldolgozás, illetve a titkosítási eljárások használata. Ezeken az alkalmazási területeken a jövő kihívásaival csak úgy boldogulhatunk, ha a jelenleg elérhetőnél nagyságrendekkel nagyobb számítási teljesítmény áll rendelkezésünkre. Pontosan ezzel magyarázható, hogy a *HPC* területen dolgozó szakemberek máris alternatív megoldásokon dolgoznak.

Újszerű feldolgozóelemek

Az elmúlt években a klaszter-alapú megoldások szép lassan kiszorították a *HPC* piacról a nagy méretű, erősen specializált rendszereket. Az ok egyszerű: a klaszterek számos alkalmazási területen olcsóbban képesek stabil és komolynak mondható teljesítményt szolgáltatni. Ugyanakkor egyre több olyan felhasználó jelentkezik, aki munkája során beleütközött a skalár processzorok eredendő, belső korlátaiba, ami arra késztette a *Cray*-t, hogy a fenti tendenciát kicsit megfordítsa. Ez a részleges visszafordulás a következőket takarja:

- *Vektorszámítógépek alkalmazása:* A vektorprocesszorok működésnek alapgondolata az, hogy a nagy adatsorokon elvégzendő azonos típusú számítási műveletek párhuzamosan is elvégezhetők, s így a hagyományos processzorok teljesítményénél jóval nagyobb sebesség érhető el.

- *Többszálú processzorok:* A *HPC* egyik érdekes ellentmondása, hogy a memóriamodulok sebessége sokkal kisebb ütemben nőtt, mint a processzoroké. Ez aztán értelemszerűen szűk keresztmetszet kialakulását eredményezte, hiszen a soros feldolgozást végző processzorok idejük jelentős részét azzal töltik, hogy az adatok megérkezésére várakoznak. A többszálú processzorokat használó rendszerekben (ilyen az *IBM Simultaneous Multi-Threading* processzora, vagy az *Intel Hyper-Threading* technológiája) ezt a problémát úgy oldják meg, hogy a processzor egyszerre több kisebb műveletsort hajt végre, miközben a szálak között megosztja a memóriát és a hozzá vezető sávszélességet. A *Cray* még egy kicsivel továbblépett ezen az úton, hiszen szálak tucatjainak párhuzamos futását, és ezzel a memória-sávszélesség teljes kihasználását teszi lehetővé.

- *Digitális jelfeldolgozó egységek (Digital Signal Processing; DSP)*: Ezek olyan, specializált processzorok, amelyek rendkívül hatékonyan tudnak folytonos jeleket, például audió, videó vagy radar adatfolyamokat feldolgozni. Mivel mindemellett általában kicsi a fogyasztásuk is, kiválóan alkalmazhatók plazmatévékben, mobiltelefonokban és számos más beágyazott rendszerben.

- *Specializált társprocesszorok:* Az olyan lebegőpontos számítások gyors elvégzésére alkalmas matematikai társprocesszorok, mint amilyeneket például a *Clearspeed Technology* gyárt, vagy amilyen a *GRAPE* n-test problémák megoldásának gyorsítására szolgáló eszköze általában egyedi tervezésű mátrixprocesszorokat használnak. Ezekkel óriási mennyiségű lebegőpontos művelet végezhető egyszerre, egyetlen chipen belül, hiszen működésük lényege, hogy rengeteg szorzó és összeadó egységet tartalmaznak. Ennek megfelelően ezek az eszközök jelentős teljesítménytöbbletet tudnak felmutatni az olyan matematikailag intenzív számításokkal kapcsolatban, mint amilyen a mátrixok invertálása vagy az n-test problémák megoldása.

A specializált feldolgozóegységek egyes területeken érzékelhetően nagyobb teljesítményt szolgáltatnak, mint általános célú társaik. A vektor- és többszálú processzorok ráadásul a késleltetésekre sem különösebben érzékenyek, hiszen képesek úgy is folytatni a folyamatban levő számításokat, hogy közben nagy mennyiségű memóriahivatkozás vár kiszolgálásra. Összefoglalva tehát ez előnyöket ezek a fejlesztések úgy juttatják jelentős teljesítménytöbblethez a speciális igényekkel rendelkező felhasználót, hogy közben csökken a cache-ek közötti kommunikáció és a hagyományos gyorstárazási stratégiák megvalósításával kapcsolatos „digitális építészetet” sem kell a tervezőknek túlzásba vinniük.
Ugyanakkor – bár a specializált processzorokat régóta sikerrel alkalmazzák – ezeknek az eszközöknek is megvannak a maguk korlátai. Először is igaz ugyan, hogy a speciális számításokat

nagyon gyorsan tudják végrehajtani, a hagyományos skaláris kódot viszont lassabban, mint a közönséges processzorok. Márpedig a legtöbb a való életben használt szoftver kódjának legalább egy része ilyen „egyszerű” algoritmust tartalmaz. Ezt a problémát általában úgy próbálják áthidalni, hogy a speciális feldolgozást végző egységet nem önállóan használják, hanem egy hagyományos rendszerhez csatlakoztatják *PCI* buszon keresztül, gyakorlatilag egyfajta perifériaként. Ezzel csak az a gond, hogy így a kommunikáció sávszélessége a két világ között meglehetősen kicsi lesz, ami értelemszerűen akadályozza köztük az együttműködést, és így gátat szab az elérhető gyorsulásnak is. (Ami azt illeti, a kiszámított adatoknak a hagyományos rendszerbe való visszaolvasása gyakran több időt vesz igénybe, mint maga a számítás.)
Van ezen kívül még egy nagy gond a célprocesszorokkal, nevezetesen a gyártásukkal kapcsolatos gazdasági megfontolások. Ha egy processzortípusnak nincs egy viszonylag nagy méretű és jól kialakult piaca, amely képes eltartani az eszközzel kapcsolatos fejlesztéseket és magát a gyártást, akkor egy vállalat kétszer is meggondolja, hogy elő merjen-e hozakodni egy új fejlesztéssel. Minden speciális eszköznek kell hogy legyen egy olyan felvevőpiaca, amely gazdaságossá teheti a gyártását. A *DSP*-knél például a fogyasztási elektronika jelenti ezt az életteret.
Mindezek a problémák arra vezették a *Cray*-t és számos más gyártót is, hogy alternatív megoldásokat keressen.

A heterogén modell

A heterogén feldolgozás alapfilozófiája az, hogy a teljes rendszer többféle feldolgozóelemből áll, és munka közben mindegyik azt a feladatot végzi, ami a képességeinek a legjobban megfelel. Ez a modell egyrészt akár százszoros gyorsulást is hozhat a fent felsorolt speciális processzorok alkalmazásával, miközben kiszélesíti a hagyományos mikroprocesszoros architektúrák alkalmazási területeit. Mivel a *HPC* alkalmazások rendszerint olyan kódrészleteket is tartalmaznak, amelyek képesek kihasználni a gyorsításra tervezett céleszközöket, és olyanokat is, amelyek legjobban a közönséges, soros feldolgozást alkalmazó processzorokon futtathatók, ezért bátran kijelenthetjük, hogy nincs „legjobb” processzor. Nincs olyan eszköz, ami valamennyi számítástípusra egyaránt jó lenne. A heterogén feldolgozásnak éppen az a lényege, hogy a felhasználónak és az általa futtatott alkalmazásnak többféle processzor áll a rendelkezésére, és mindegyik részművelethez azt használhatja, amelyek a célnak a legjobban megfelel.
Hagyományosan két nagy akadálya volt a heterogén számítógépes rendszerek széles körben való elterjedésének. Az egyik az ilyen rendszerek programozásának összetettsége, ami abból fakad, hogy a felhasználónak (fejlesztőnek) magának kell gondoskodnia az egyes részfeladatok ésszerű kiosztásáról. A másik probléma a hagyományos és a célprocesszorok eltérő architektúrájából fakad, ami szintén bonyolítja az ilyen rendszerek programozását. Ezek értelemszerűen jelentős gátló tényezők lehetnek, amelyeket mindenképpen figyelembe kell venni, mielőtt az ember belevág egy ilyen vállalkozásba. A jó döntéshez látnunk kell előre, hogy menyit nyerünk az egyik oldalon és mennyit vesztünk a másikon. Ugyanakkor az is igaz, hogy a többmagos rendszerek megjelenésével a HPC területen dolgozó fejlesztők általános hozzáállása is változóban van. Itt gyakorlatilag egy technológiai ugrás következett be, így a programozók és az alkalmazások tervezői egyre inkább hajlandóak elsajátítani új architektúrák használatát, vagy legalább fontolóra venni azt.
És ebbe azok a heterogén rendszerek is belefférnek, amelyek egyre-másra kezdenek megjelenni a piacon.
A *Cray X1E* szuperszámítógép például vektor és skalár processzorokat egyaránt tartalmaz, sőt van hozzá egy olyan speciális fordítóprogram is, amely automatikusan szétosztja a terhelést azok között. De akad hasonló példa más cégek háza táján is. Az új *Cell* processzor-architektúra, amit az *IBM*, a *Sony* és a *Toshiba* közösen fejlesztettek az új *Playstation 3* rendszeren futó játékok támogatására, szintén úgy működik, hogy egy hagyományos processzor szükség estén részfeladatokat oszt ki olyan specializált feldolgozóegységeknek, amelyeknek hozzá hasonlóan közvetlen hozzáférése van a memóriához. A heterogén rendszerek manapság legizgalmasabb területe azonban az úgynevezett térprogramozható kapumátrix processzorok (*Field Programmable Gate Array; FPGA*) alkalmazása.

Az FPGA társprocesszor modell

Az *FPGA*-k olyan hardveresen átkonfigurálható célprocesszorok, amelyek belső logikáját a programozó újra és újra átírhatja az éppen megoldandó problémának megfelelően. *FPGA*-kat tulajdonképpen már több mint egy évtizede használnak programozható logikai eszközként, társprocesszorként való felhasználhatóságuk azonban csak újabban merült fel. Az *Egyesül Államokban* és néhány más országban a közelmúltban több a témával kapcsolatos konferenciát is tartottak, az *Ohio Supercomputing Center* pedig létrehozta az *OpenFPGA* kezdeményezést (➲ *www.openfpga.org*) azzal a céllal, hogy felgyorsítsa az *FPGA* eszközöknek a *HPC* területén való meghonosodását.
Ennek az általános lelkesedésnek természetesen jó oka van: az *FPGA*-k bizonyos típusú problémákkal kapcsolatban több nagyságrenddel növelhetik a feldolgozás sebességét. *FPGA*-k segítségével a tervezők olyan céleszközöket hozhatnak létre minden egyes megoldandó problémához, amelyekben elemi feldolgozóegységek százait vagy ezreit vezényelhetik arra, hogy egymással párhuzamosan működve hajtsanak végre egy műveletet. Ez különösen azokon a területeken jár óriási előnnyel, ahol sok bitszintű műveletet, összeadást, szorzást, összehasonlítást, konvolúciót vagy transzformációt kell végezni. Az *FPGA*-k megfelelően felprogramozva ilyen műveletből egyszerre rengeteget tudnak végrehajtani úgy, hogy a műveletet sem az operációs rendszernek, sem a futó folyamatnak nem kell kívülről felügyelnie. Az már csak a hab a tortán, hogy a kapcsolódó energiafogyasztás is jóval kisebb, mint a hagyományos processzorok esetében.
Az *FPGA*-k széles körben való elterjedésének – amint mondani szokták – történeti okai vannak. Először is ezeket a processzorokat általában hagyományos rendszerek részeként használták úgy, hogy azokhoz *PCI* buszon keresztül, perifériaként kapcsolódtak. Ebből ugyanazok az átviteli sávszélességgel

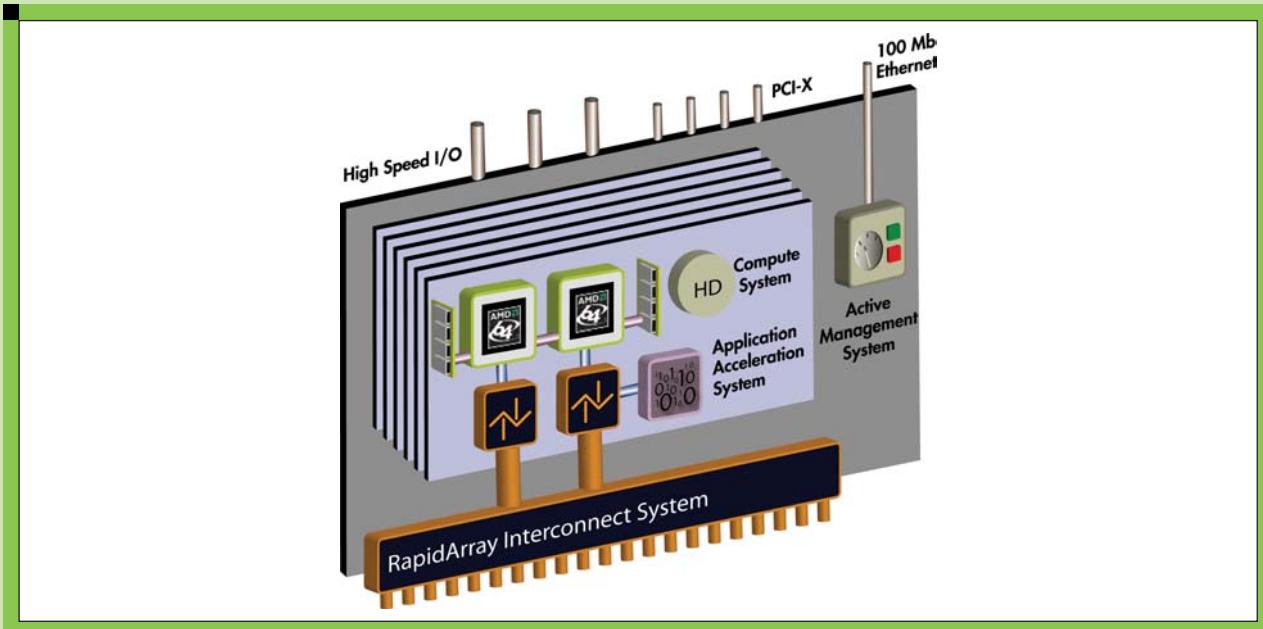
kapcsolatos problémák adódtak, mint amelyeket korábban a specializált processzorokkal kapcsolatban említettem. Az igazán nagy problémát ugyanakkor nem is ez jelentette, hanem hogy a hagyományos alkalmazásokat át kellett volna írni úgy, hogy képesek legyenek együttműködni az *FPGA*-val. Ez pedig kifejezetten nehéznek bizonyult, mivel az *FPGA*-kat *HDL (Hardware Design Language)* nyelven kellet volna programozni. És bár ezt a nyelvet az elektronikával foglalkozó villamosmérnökök naponta használták, a *HPC* rendszerek tervezői, programozó és különösen felhasználói számára tökéletesen ismeretlen volt. Igazából még manapság is épp csak fejlődőben vannak azok az eszközök, amelyekkel a „földi halandók" is képesek kihasználni az *FPGA*-k nyújtotta lehetőségeket. Ami pedig a meglevő alkalmazások átírását illeti, a felhasználók egyelőre csak várnak azokra az eszközökre, amelyekkel ezt viszonylag könnyen megtehetik. A *Cray*-nél éppen ezért számos szakember dolgozik azok, hogy elhárítsa ezeket az akadályokat a fejlődés útjából.
A *Cray XD1* szuperszámítógép például az egyik első olyan kereskedelmi forgalomban kapható *HPC* rendszer, amely *FPGA*-kat használ felhasználó

Smith-Waterman: Gyakorlati példa az FPGA társprocesszorok és a heterogén számítógépek felhasználhatóságára

A genomika legújabb módszereivel egyszerre több millió *DNS* szakaszról nyerhetünk információt, ennek a nyers adattömegnek az értelmes eredményekké való átalakítása azonban nem triviális feladat. A géneket nukleotidok sorrendjeként írhatjuk le. (Ehhez teljesen hasonló a fehérjék leírása, de ott az aminosavak sorrendjét kell megadni.)
A kutatók ezeknek a sorrendeknek a statisztikai elemzésével számos érdekes kérdésre tudnak választ adni. Összehasonlítva például két faj génkészletét érdekes hasonlóságok, átfedések fedezhetők fel, amelyek a genetikai rokonságra utalnak. Ehhez persze kell egy olyan módszer, amivel pontosan meg lehet határozni, hogy két jelsorozat miben és milyen mértékig hasonlít egymásra.
Ez a módszer egy ideje már létezik is, hiszen a megfelelő algoritmus leírását *Smith* és *Waterman* 1981-ben közölte le (*Temple F. Smith and Michael S. Waterman, "Identification of Common Molecular Subsequences", J. Mol. Biol.*, 147:195—197, 1981). Van azonban a dologgal egy apró probléma: az algoritmus olyan bonyolult, hogy hagyományos processzorokkal csak igen lassan hajtható végre. Az *FPGA* társprocesszorokat tartalmazó *Cray XD1* heterogén szuperszámítógép jelentős áttörést hozott ezen a területen, hiszen megfelelően programozva a *Smith-Waterman* algoritmust körülbelül 40-szer gyorsabban képes végigszámolni, mint elődei.

A Smith-Waterman algoritmus

A *Smith-Waterman* algoritmus *DNS* vagy fehérjeszekvenciákat hasonlít össze már létező adatbázisokkal. Mivel mind a minta, mind maga az adatbázis tartalmazhat hibákat hiányzó betűk, vagy véletlenül többletként bekerült jelek formájában, és mivel ezek az apró eltérések adott esetben óriási biológiai különbségnek felelhetnek meg, a feldolgozás során nagyon pontos megfeleltetést kell megvalósítani.
Egy génszekvencia tulajdonképpen nem egyéb, mint négy betű (a négy nukleotid: *G*, *C*, A és *T*) látszólag véletlenszerű sorozata. Egy fehérje ezzel szemben húsz elemből (aminosavak) épül fel, de matematikai szempontból ugyanilyen alaptulajdonságokkal rendelkezik. Mivel a génszekvenciák rendezett jelsorozatok, az összehasonlítás során azt kell megállapítani, hogy az adatbázis tartalmaz-e olyan betűket/jeleket, mint a minta, illetve ha igen, akkor a kettő mennyire hozható fedésbe egymással. Magyarra lefordítva például a STOP és a POTS ugyanazokat a jeleket tartalmazza, de a két jelsorozat nem hozható fedésbe egymással. Ugyanakkor a POTS és a POINTS hiába tartalmaz eltérő betűket is, a genetika játékszabályai szerint mégis fedésbe hozhatók, ha a második sorozatban egy lyukat képzelünk az O és T közé. Ezek a szekvenciák tehát potenciális rokonok. A *Smith-Waterman* algoritmus úgynevezett „dinamikus programozást" használ az optimális illesztések megtalálásához, aminek a végrehajtása rengeteg bitszintű műveletet és egyszerű de párhuzamosan végezhető számításokat tartalmaz. És sajnos éppen ez az a két terület, ahol a hagyományos processzorok rendkívül rosszul teljesítenek.
Ha a *Smith-Waterman* tesztet egy hagyományos processzoron futtatjuk, akkor minden egyes adatnak a mintával való összehasonlítása több ezer egyedi lépés végrehajtását jelenti. A tényleges összehasonlítás elvégzésére fordított elemi lépések száma csupán töredéke azok számának, amelyek ahhoz szükségesek, hogy a következő összehasonlítási pontot kiszámítsuk, illetve hogy az egész rendszer logikáját kezeljük. Ha statisztikailag közelítjük meg a dolgot, akkor azt mondhatjuk, hogy egy skalár processzor 100 elemi lépésből mindössze egyet fordít a tényleges összehasonlítás elvégzésére, vagyis – bizonyos értelemben – 1 százalékos hatásfokkal működik.
Egy *FPGA* processzorokat használó *HPC* rendszer számos ponton ennél sokkal hatékonyabban képes működni. Először is szemben a hagyományos processzorokkal, amelyeket úgy terveznek meg, hogy számos különböző típusú kód futtatható legyen rajtuk, az *FPGA* processzorok utasításkészletét teljesen az adott problémához lehet igazítani. Az *FPGA*-k felépítése emellett eleve olyan, hogy rengeteg lehetőséget adnak a párhuzamosításra. Lehetőségünk van például arra, hogy párhuzamosan kapcsoljunk nagy számú elemi összehasonlító egységet, és összehasonlítások ezreit végeztessük el az *FPGA*-val egyetlen órajel alatt.



1. ábra A Cray XD1 rendszer felépítése

Hasonlóan nagy hatékonyságot érhetünk el a bitszintű műveleteknél is, mivel az *FPGA*-k felépítésükből fakadóan ezeket is sokkal hatékonyabban képesek végrehajtani.

A Cray XD1 által alkalmazott megközelítés

Ahhoz, hogy átláthassuk, miért is olyan hatékony a *Cray XD1* a *Smith-Waterman* algoritmussal kapcsolatban, ismernünk kell a rendszer *FPGA* társprocesszorokkal kapcsolatos architektúráját (lásd az *1. ábrát*), illetve magának az alkalmazásnak a működési logikáját.

A *Smith-Waterman* algoritmus működésének alapja egy ponttáblázat (*scoring matrix*), amelynek felső sorában és oldalsó oszlopában az összehasonlítandó sorozatok vannak, a metszéspontokban levő cellák pedig az egyezések számát tartalmazzák. Utóbbiakat természetesen a programnak kell kitöltenie. Amint a mátrix elkészült, az algoritmus veszi a legnagyobb találatokat tartalmazó mezőket, visszaköveti, hogy miből keletkeztek a találatok, és ebből határozza meg a végső illesztést (*2.* és *3. ábra*).

Ennek a folyamatnak a felgyorsítása végett a *Cray XD1* felosztja az elvégzendő műveleteket az *FPGA* és az *Opteron* processzorok között. A pontozási táblázatot az *FPGA*-n futó kód tölti ki (mivel ez jól párhuzamosítható), a mátrix frissítését (szekvenciális kód) pedig az *Opteronok* végzik az *FPGA*-k által visszaküldött adatok alapján.

Mindez a gyakorlatban úgy valósul meg, hogy a szuperszámítógép *HPC*-re optimalizált *Linux* operációs rendszere kizárólag a *Smith-Waterman* alkalmazás magjának végrehajtásakor hívja meg az *FPGA* processzorokat. Ugyanakkor mivel az *FPGA*-k nagy mennyiségű ilyen speciális számítást tudnak párhuzamosan végrehajtani, a végeredmény körülbelül 24-40-szeres gyorsulás lesz. Az alábbi kódrészlet azt mutatja be, miként használhatja ki egy alkalmazás az *FPGA* képességeit:

Scoring Matrix

	0	A	C	G	T	A	T	G	C
0	0	0	0	0	0	0	0	0	0
A	0	2	0	0	0	2	0	0	0
C	0	0	4	2	1	0	1	0	2
G	0	0	2	6	4	3	2	3	1
A	0	2	1	4	5	6	4	3	2
A	0	2	1	3	3	7	5	4	3
C	0	2	4	2	2	5	6	4	6
C	0	0	2	3	1	4	4	5	6
C	0	0	2	1	2	3	3	3	7
T	0	0	0	1	3	2	5	3	5
T	0	0	0	0	3	2	4	4	4
G	0	0	0	2	1	2	2	6	4
C	0	0	2	0	1	0	1	4	8

2. ábra A pontozási táblázat

Final Alignment

A	C	G	A	A	C	C	C	T	T	G	C
|	|	|		|					|	|	|
A	C	G	T	A	-	-	-	-	T	G	C

3. ábra A végső illesztés meghatározása a Smith-Waterman formula alapján

```
/* Tömbök elforgatása az FPGA segítségével */
static void tilt (int fp_id, u_64
➥*trans_matrix, int row_len)
{
  int i = 0;
  u_64 status = 0;
```

```
  /* Inicializáljuk az FPGA-t, és felkészítjük
  ➥a tömbök egy új sorozatának fogadására. */
  fpga_wrt_appif_val (fp_id, TILT_START,
  ➥TILT_APP_CFG, TYPE_VAL, &e);
  /* Átmásoljuk a mátrixot az FPGA
  ➥memóriaterületére. */
  memcpy((char *) fpga_ptr, (char *)
  ➥trans_matrix,
row_len*sizeof(u_64));
  /* Folyamatosan figyeljük, hogy az FPGA
  ➥elkészült-e a végrehajtással. */
  while (1) {
    fpga_rd_appif_val (fp_id, &status,
    ➥TILT_APP_STAT, &e);
    if (status & TILT_DONE) break;
  }
  /* Amikor az FPGA elkészült, valamennyi
  ➥transzponált adatot kiírja */
  /* a DRAM azon területére, ami az adatcserére
  ➥szolgál. */
  /* Adatok visszamásolása az átviteli
  ➥területről az eredti tömbbe. */
  //  for(i=0;i<row_len;i++) {
  //  trans_matrix[i] = dram_ptr[i];
  //}
  return;
}
```

A Cray XD1 heterogén architektúra előnyei

Azzal a teljesítménytöbblettel, amit a ***Cray XD1*** heterogén architektúrája nyújt, a felhasználók lehetőséget kapnak arra, hogy problémáik megoldására a legjobb algoritmust alkalmazzák ahelyett, hogy egy kevésbé pontos, de gyorsabb módszer mellett kellene dönteniük. Mivel pedig a rendszer kizárólag a ***Smith-Waterman*** módszer magjának végrehajtásához használja az ***FPGA*** társprocesszor szolgáltatásait, ezt a kódrészletet is könnyű frissíteni, ha az alkalmazás későbbi fejlesztése azt megkívánja. Szintén fontos megjegyezni, hogy szemben a különféle dedikált hardveres megoldásokkal a ***Cray XD1*** egy valódi, általános célú ***HPC*** megoldás, ami szemben az előbb említett teljesen specializált rendszerekkel többféle kódot is képes futtatni. Ennek megfelelően más, szintén bioinformatikai célokra ugyanolyan könnyen és hatékonyan használható, mint a ***Smith-Waterman*** algoritmus megvalósítására. Összefoglalva tehát a ***Cray XD1*** hatékony, megfizethető és értékálló beruházást jelent minden olyan az élettudományok területén dolgozó szervezet számára, amelyek kimagasló számítási teljesítményre van szüksége tevékenységének végzéséhez.

által programozható gyorsítóegységekként. Ennek a gépnek a tervezésekor a fejlesztők már számos, fent említett akadályt elhárítottak, amit elsősorban azzal értek el, hogy az ***FPGA***-t kezelő alrendszert szorosan összeépítették a gépen futó, ***HPC*** műveltekre optimalizált ***Linux*** operációs rendszer kódjával. Ráadásul léteznek már olyan új eszközök is, amelyekkel a felhasználó gyakorlatilag egy a C-hez hasonló magas szintű nyelven keresztül programozhatja az ***FPGA*** logikai elemeit. Ez az eszközkészlet tartalmazza például a ***Celoxica DK Design Suite*** nevű csomagot, amely egy a ***Cray XD1*** rendszerébe integrált ***C*** kódot FPGA kóddá alakító fordítóprogram. Hasonló segédeszköz az Impulse C, a Mitrition C, illetve a ***Matlab*** fejlesztői által írt ***Simulink-to-FPGA*** is, amelyek valamennyien támogatják a modell-alapú tervezési megközelítést. Összességében úgy gondoljuk, hogy amint az ***FPGA***-val kiegészített heterogén rendszerek szélesebb körben elterjednek, a ***HPC*** rendszerek felhasználói egyrészt gyorsabban oldhatnak meg olyan problémákat, amelyek megoldhatóvá válását a ***Moore*** törvény alapján is ki lehetett következtetni, másrészt viszont olyan problémák is megoldhatóvá válhatnak, amelyek a szükséges számítási kapacitás miatt eddig reménytelennek tűntek. (Ebbe az utóbbi csoportba tartozik például a keretes részben bemutatott ***Smith-Waterman*** nevű bioinformatikai eljárás, amely ***FPGA*** társprocesszoros rendszerekkel immár kezelhető.)

Előretekintés

Bár a heterogén feldolgozás terén már ma is számos érdekes kezdeményezés figyelhető meg, a dolog még egyáltalán nem tart ott, hogy dominanciát érhetne el a ***HPC*** valamennyi területén. A jelenleg is létező problémák – melyek közül a legfontosabb még mindig a nehézkes fejlesztés, illetve a meglevő alkalmazások átültetése – egyelőre túl nagyok ahhoz, hogy a módszer általánosan elterjedhessen. Ugyanakkor a ***Cray*** és a ***HPC*** piac egyéb szereplői keményen dolgoznak az út járhatóvá tételén.
Akárcsak minden új technológiánál, a heterogén feldolgozásnál is azon múlik majd a további haladás menet, hogy milyen eredményeket lehet általa elérni, és ehhez milyen anyagi és szellemi ráfordítások szükségesek. Összességében azonban úgy gondoljuk, hogy hosszú távon a módszer által nyújtott teljesítménytöbblet mindenképpen túl nagy lesz ahhoz, hogy egyszerűen figyelmen kívül lehessen hagyni.

Linux Journal 2006. 142. szám

Amar Shan

Vezető termékmenedzser a Cray Inc. nevű szuperszámítógépeket gyártó vállalatnál. Shan 2004-ben csatlakozott a céghez, amikor az felvásárolta az OctigaBay Systems Corporation nevű vállalatot. Jelenleg a Cray következő generációs termékeinek fejlesztési irányelveit dolgozza ki, illetve ő felel a Cray XD1 nagy teljesítményű szuperszámítógép fejlesztéséért. Jelenleg ez a világ egyetlen olyan Linux/Opteron rendszere, amit kifejezetten HPC célokra terveztek. Shan a University of Waterloo-n szerzett diplomát a mesterséges intelligencia gyakorlati alkalmazásából, a University of British Columbián pedig villamosmérnöki tudományból kapott BSc fokozatot.