Úvod

Vývoj výpočtovej techniky patrí od samotného počiatku k rýchlo napredujúcim oblastiam. Jedným z trendov určujúcich faktorov počas asi posledných troch dekád 20. storočia bol empirický zákon – Mooreov zákon [1]. Podľa neho sa zložitosť integrovaných obvodov (počet tranzistorov) zdvojnásobí približne každé dva roky. Vďaka tomuto faktu nebolo potrebné pre zvýšenie rýchlosti aplikáciu nijak významne upravovať, jednoducho stačilo počkať rok na ďalšiu generáciu procesorov s približne o 50 % vyšším výkonom.

Platnosť tohto zákona však naráža na isté limity, čo sa odzrkadlilo na prelome tisícročí, kedy nastáva dramatický pokles nárastu výkonu jednojadrových procesorov na asi 20 % ročne. Zatiaľ čo predtým sa výkon procesora počas jednej dekády zvýšil približne 60-krát, teraz by to predstavovalo iba približne 10-násobné zvýšenie výkonu počas desiatich rokov. Hlavná podstata zvyšovania výkonu procesora spočívala najmä v jeho vyššej integrácií, tj. zvyšovania počtu tranzistorov v integrovanom obvode. Zmenšujúca sa veľkosť tranzistorov umožnila zrýchlenie ich funkcie avšak za cenu väčšej spotreby energie a následne väčšieho množstva uvoľňovanej nežiadúcej energie v podobe tepla.

Toto je hlavným dôvodom, prečo nie je možné naďalej zvyšovať rýchlosť procesora pri použití bežne používaných súčasných technológií. Preto do roku 2005 väčšina veľkých výrobcov procesorov pristúpila k zásadnej zmene návrhu procesorov. Namiesto toho, aby sa snažili zvyšovať výkon samotného procesora, pristúpili k integrovaniu viacerých procesorov do jedného integrovaného obvodu, čo viedlo k masívnemu rozšíreniu viacjadrových procesorov (tzv. Multicore CPU) [2].

Zavedené zmeny v dizajne procesorov však neviedli k významnému zvýšeniu rýchlosti behu väčšiny existujúcich sériových programov, ktoré aj tak dokázali využívať iba jedno jadro procesora. Tu si môžeme položiť otázku, či je vôbec potrebné zvyšovať výkon procesora a ako prispôsobiť program tomuto novému dizajnu multiprocesora.

K neustálej požiadavke po vyššom výkone procesorov prispievajú dve skutočnosti:

- snaha o skrátenie času výpočtu,
- zvýšenie množstva spracovaných údajov.

Ľudstvo, ako také, je od nepamäti zvedavé, čo vedie k posúvaniu hraníc poznania. Obzvlášť to platí pre vedeckú komunitu, ktorej práve rastúce výpočtové kapacity poskytujú priestor pre komplexnejšie a presnejšie skúmanie modelovaných javov z reálneho sveta. Charakteristickou črtou súčasnej *Štvrtej priemyselnej revolúcie*, Inteligentných riešení a Internetu vecí je veľké množstvo generovaných a ukladaných údajov, ktoré je potrebné spracovať. Významné výsledky boli dosiahnuté v rôznych vedných oblastiach, ako sú: aeronautika, astrofyzika, bioinformatika, chémia, ekonomika a obchod, energetika, geológia a

geofyzika, materiálové vedy, klimatológia, medicína, meteorológia, nanotechnológia, obrana a mnohé ďalšie [3–5]. Pre lepšiu predstavu uvedieme niekoľko príkladov:

- *Aeronautika.* Simulácie javov súvisiacich s letmi raketoplánov NASA. Na základe simulácie tlaku vzduchu počas letu boli navrhnuté niektoré časti raketoplánu [6]. Ďalšou úlohou bola simulácia toku kvapalného paliva do hlavných motorov raketoplánu [7]. Taktiež návrh rotora helikoptéry UH-60 Blackhawk bol zrealizovaný za pomoci veľmi presnej simulácie počas letu vpred pri vysokej rýchlosti [8, 9].
- *Klimatológia.* Pre lepšie pochopenie klimatických zmien na Zemi potrebujeme použiť čo najpresnejšie modely, ktoré by zahŕňali interakcie medzi atmosférou, oceánmi a súšou, ako aj zmeny v polárnych ľadovcoch. Výpočty môžu napomôcť k zodpovedaniu otázok: či bude viesť zvyšovanie teploty ku klimatickej katastrofe, alebo či je globálne otepľovanie prirodzený jav, alebo je zapríčinené ľudskou činnosťou a skleníkovým efektom a produkciou oxidu uhličitého [10].
- *Meteorológia.* Ďalšou z domén paralelného počítania je predpovedanie počasia [11, 12], tiež predpovedanie prírodných katastrof [13], ako sú zemetrasenia [14], sopečné erupcie a vĺn tsunami [15], hurikánov, tornád a tropických cyklónov [16–18].
- *Medicína.* V oblasti zdravotnej starostlivosti je význam rastúceho výpočtového výkonu taktiež nezanedbateľný. Využitie počítačových modelov a simulácií pri vývoji nových liečiv si našlo svoje uplatnenie [19]. Taktiež analýza ľudského genómu môže napomôcť k pochopeniu a liečeniu rôznych typov špecifických ochorení alebo aplikácie špeciálne navrhnutých liečiv [20–22].
- *Analýza dát.* Súčasným trendom je produkcia a zaznamenávanie veľkého množstva údajov z rôznych senzorov, ako aj údaje generované používateľmi rôznych aplikácií, ktoré umožňujú získavať rôzne druhy informácií. Podľa niektorých odhadov sa každé dva roky zdvojnásobí množstvo uložených údajov [23]. Tieto však nadobúdajú svoj význam až po ich analýze. Ďalšími príkladmi sú dáta získané z urýchľovača v CERNe [24, 25], či údaje z bežne používaných webových vyhľadávačov [26].

Posledná otázka, ktorú by sme si mali položiť je, či existuje jednoduchý spôsob, ako prispôsobiť sériovo napísaný program tak, aby bol schopný efektívne využiť potenciál, ktorý mu ponúkajú viacjadrové procesory. Prvý najjednoduchší spôsob predstavuje niekoľkonásobné spustenie inštancie toho istého programu, čo umožní tento potenciál využiť. Avšak toto riešenie nie je vždy postačujúce. Predstavme si, že máme obľúbenú počítačovú hru a súčasne spustíme jej dve inštancie, ktoré budú bežať na tom istom počítači. Je zrejmé, že toto riešenie nám neprinieslo žiadnu výhodu, naopak, uvítali by sme využitie výkonu procesora na prepracovanejšie detaily grafiky a plynulejší beh hry. Toho je možné dosiahnuť tak, že pôvodný sériový program prerobíme na paralelný program, ktorý dokáže plne využiť výkon multiprocesora. Jednou z možností je napísať prekladač, ktorý dokáže túto zmenu programu urobiť automaticky. Táto možnosť však častokrát nevedie k efektívnemu paralelnému programu a efektívnemu využitiu výpočto-

vých prostriedkov. Takýto prekladač dokáže rozpoznať niektoré bežne používané konštrukcie v programe, ktoré je možné efektívne paralelizovať, ale nedokáže posúdiť program ako celok a nájsť efektívnejšie paralelné riešenie.

Druhá možnosť predstavuje celkové predizajnovanie a prepísanie programu do nového paralelného programu, pričom je potrebné zohľadniť všetky špecifiká novej architektúry. Ako príklad si môžme uviesť program 1 pre výpočet hodnoty faktoriálu prirodzeného čísla n podľa vzťahu 1.

$$n! = \prod_{i=1}^{n} i \tag{1}$$

Zdrojový kód 1: Sériový kód pre výpočet faktoriálu prirodzeného čísla

```
faktorial = 1;
for(i = 1; i <= n; i++) {
    faktorial *= i;
}
```

Takýto seriový program by vedel využiť iba jedno jadro procesora, ktoré by vykonalo všetkých n násobení. Za predpokladu, že máme k dispozícii p jadier procesora a $p << n$, tak môžme rozdeliť tento výpočet čo najrovnomernejšie medzi jednotlivé jadrá tak, že každé z nich bude musieť vykonať $\frac{n}{p}$ násobení, tak ako je uvedené v zdrojovom kóde 2.

Úloha: Odvoďte vzťah pre výpočet hodnôt pre premenné moj_začiatok a moj_koniec.

Zdrojový kód 2: Paralelný kód pre výpočet faktoriálu prirodzeného čísla

```
moj_faktorial = 1;
moj_začiatok = ... ;
moj_koniec = ... ;
for(i = moj_zaciatok; i <= moj_koniec; i++) {
    moj_faktorial *= i;
}
```

Predpona `moj_` predstavuje súkromnú premennú pre každé z jadier, ktoré vykonávajú svoju časť kódu nezávisle od ostatných jadier. Predpokladajme, že máme k dispozícii 8-jadrový procesor, teda $p = 8$ a chceme vypočítať faktoriál čísla $n = 16$. Jednotlivé výpočty sa rozdelia medzi jednotlivé jadrá podľa tabuľky 1. Predpokladáme, že každé z jadier bude vykonávať približne rovnaké množstvo práce, a teda každé z jadier sa dopracuje k svojmu čiastkovému výsledku v približne tom istom čase.

Takýmto spôsobom získame p čiastkových výsledkov, ktoré je potrebné skombinovať do celkového výsledku, podľa nasledujúceho zdrojového kódu 3.

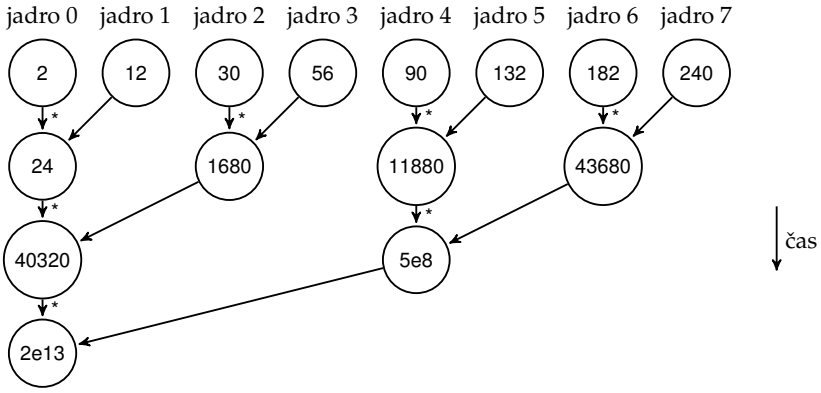
Tabuľka 1: Rozdelenie výpočtov medzi jadrá procesora

Číslo jadra	0	1	2	3	4	5	6	7
Výpočet	1*2	3*4	5*6	7*8	9*10	11*12	13*14	15*16
Výsledok	2	12	30	56	90	132	182	240

Zdrojový kód 3: Kombinovanie výsledkov od všetkých jadier

```
if(jadro == 0) {
    faktorial = moj_faktorial;
    for(j = 1; j < p; j++) {    //všetky ostatné jadrá
        prijmi hodnotu od jadra j;
        faktorial *= hodnota;
    }
} else {
    odošli moj_faktorial jadru 0;
}
```

Takto upravený program už síce bude benefitovať z využitia viacjadrového procesora, avšak po podrobnejšej analýze sa ponúka ešte efektívnejšie riešenie. Pri uvedenom riešení musí hlavné jadro vykonať $p - 1$ násobení, zatiaľ čo ostatné jadrá nemajú žiadnu prácu. Preto by bolo vhodnejšie použiť riešenie, kde svoje čiastkové výsledky každé druhé jadro pošle svojmu susednému jadru a to ho vynásobí so svojím čiastočným výsledkom. Následne sa takýto postup bude uplatňovať až po získanie finálneho výsledku tak, ako je znázornené na obrázku 1.



Obr. 1: Kombinovanie výsledkov od všetkých jadier

Úloha: Odvoďte vzťah pre počet prijatí hodnoty a počet násobení v závislosti na hodnote n a počte jadier procesora p, ktoré musí vykonať jadro 0 pre oba prípady kombinovania čiastkových výsledkov.

Táto zmena umožní vykonanie niekoľkých násobení v tom istom čase, a tým skracuje celkový čas potrebný na získanie výsledku. Predstavme si, že by sme takýmto spôsobom potrebovali skombinovať 1000 čiastkových výsledkov. V prvom prípade, keď kombinovanie výsledkov má na starosti iba jedno jadro, by toto muselo vykonať postupne 999 operácií. V druhom prípade by hlavné jadro muselo postupne vykonať týchto operácií iba 10, čo predstavuje značnú časovú úsporu.

Pri použití automatického prekladača sériového programu na paralelný je možné očakávať, že tento dokáže v sériovom kóde identifikovať isté konštrukcie, ktorých vykonávanie je schopný rozdeliť medzi

viaceré jadrá procesora alebo ich dokáže nahradiť iným efektívne paralelizovaným kódom. Avšak s rastúcou zložitosťou samotného programu je pre prekladač čoraz obtiažnejšie identifikovať takéto časti programu. Preto bude našou snahou pri vytváraní efektívnych paralelných programov zamerať sa nielen na prerobenie sériového programu, ale samotný návrh nového paralelného programu tak, aby bol schopný plne využiť potenciál dostupného hardvérového a softvérového vybavenia.

Literatúra

[1] Gordon E. Moore. "Cramming more componentsonto integrated circuits". In: *Electronics* 38.8 (1965), pp. 114–118 (cited on page 1).

[2] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. MA USA: Morgan Kaufmann, 2008 (cited on page 1).

[3] Peter S. Pacheco. *An Introduction to Parallel Programming*. MA USA: Morgan Kaufmann, 2011 (cited on page 2).

[4] Zbigniew J. Czech. *Introduction to parallel computing*. UK: Cambridge University Press, 2016 (cited on page 2).

[5] Ján Kollár. *Metódy a prostriedky pre výkonné paralelné výpočty*. Košice: elfa, 2003 (cited on page 2).

[6] Rupak Biswas et al. "Petascale Computing: Impact on Future NASA Missions". In: *Petascale Computing. Algorithms and Applications*. Ed. by David A. Bader. Boca Raton, FL: Chapman & Hall/CRC, 2008. Chap. 2, pp. 29–46 (cited on page 2).

[7] David A. Bader. *Petascale Computing. Algorithms and Applications*. Boca Raton, FL: Chapman & Hall/CRC, 2008 (cited on page 2).

[8] Neal M Chaderjian and Pieter G Buning. "High resolution Navier-Stokes simulation of rotor wakes". In: *Proceedings of the American Helicopter Society 67th Annual Forum*. 2011 (cited on page 2).

[9] Neal M Chaderjian and Jasim U Ahmad. "Detached eddy simulation of the UH-60 Rotor wake using adaptive mesh refinement". In: *Proceedings of the American Helicopter Society 68th Annual Forum*. 2012 (cited on page 2).

[10] William D Collins et al. "The community climate system model version 3 (CCSM3)". In: *Journal of Climate* 19.11 (2006), pp. 2122–2143 (cited on page 2).

[11] Takashi Shimokawabe et al. "145 TFlops performance on 3990 GPUs of TSUBAME 2.0 supercomputer for an operational weather prediction". In: *Procedia Computer Science* 4 (2011), pp. 1535–1544 (cited on page 2).

[12] Richard Loft et al. "Yellowstone: A dedicated resource for earth system science". In: *Contemporary High Performance Computing*. Chapman and Hall/CRC, 2017, pp. 185–224 (cited on page 2).

[13] Donald Hyndman and David Hyndman. *Natural hazards and disasters*. Cengage Learning, 2016 (cited on page 2).

[14] A. Nonnellan et al. *Computational Earthquake Science*. Basel: Birkhuser, 2004 (cited on page 2).

[15] Michael Bader, Alexander Breuer, and Martin Schreiber. "Parallel fully adaptive tsunami simulations". In: *Facing the Multicore-Challenge III*. Springer, 2013, pp. 137–138 (cited on page 2).

[16] Shuyi S Chen et al. "The CBLAST-Hurricane program and the next-generation fully coupled atmosphere–wave–ocean models for hurricane research and prediction". In: *Bulletin of the American Meteorological Society* 88.3 (2007), pp. 311–318 (cited on page 2).

[17] Ming Xue, Kelvin K Droegemeier, and Daniel Weber. "Numerical prediction of high-impact local weather: A driver for petascale computing". In: *Petascale Computing: Algorithms and Applications* (2007), pp. 103–124 (cited on page 2).

[18] Malcolm J Roberts et al. "Tropical cyclones in the UPSCALE ensemble of high-resolution global climate models". In: *Journal of Climate* 28.2 (2015), pp. 574–596 (cited on page 2).

[19] Chun Meng Song, Shen Jean Lim, and Joo Chuan Tong. "Recent advances in computer-aided drug design". In: *Briefings in bioinformatics* 10.5 (2009), pp. 579–591 (cited on page 2).

[20] C Venter et al. “INTERNA TIONAL HUMAN GENOME SEQUENCING CONSORTIUM”. In: *Initial Sequencing and Analysis of Human Genoma” Nature* 409 (2001), pp. 860–921 (cited on page 2).

[21] J Craig Venter et al. “The sequence of the human genome”. In: *science* 291.5507 (2001), pp. 1304–1351 (cited on page 2).

[22] Paul C Winter, G Ivor Hickey, Hugh L Fletcher, et al. *Instant notes in genetics.* BIOS Scientific Publishers Ltd, 1998 (cited on page 2).

[23] IBM. *IBM InfoSphere streams v1.2.0 supports highly complex heterogeneous data analysis*. 2010 (cited on page 2).

[24] Alessandro Motta et al. “Big data in nanoscale connectomics, and the greed for training labels”. In: *Current opinion in neurobiology* 55 (2019), pp. 180–187 (cited on page 2).

[25] Esma Mobs and Melissa Marie Rudolf. *CERN Quick Facts 2019 (English version)*. Tech. rep. 2019 (cited on page 2).

[26] Natasha Noy, Matthew Burgess, and Dan Brickley. “Google Dataset Search: Building a search engine for datasets in an open Web ecosystem”. In: *28th Web Conference (WebConf 2019)*. 2019 (cited on page 2).