Semestrálna práca MI-PAR 2013/2014:

Paralelný algoritmus pre rozklad obdĺžnikov

Jakub Melezínek

Martin Klepáč

November 26, 2013

1 Definícia problému

Našou úlohou bolo vytvoriť program, ktorý implementuje usporiadanie obdĺžnikov v 2D mriežke so zachovaním minimálneho celkového obvodu týchto obdĺžnikov.

Obsah obdĺžnika je daný hodnotou uloženou v 2D mriežke, pričom tento element musí byť súčasťou obdĺžnika s daným obsahom.

Jednotlivé obdĺžniky sú vzájomne disjunktné až na spoločné vrcholy a hrany, pričom zjednotenie všetkých obdĺžnikov pokrýva pôvodnú mriežku.

Riešení, ako rozdeliť mriežku na jednotlivé obdĺžniky, môže byť viac – v takom prípade hľadáme riešenie s minimálnym obvodom – zároveň ale riešenie nemusí existovať.

2 Formát vstupu, výstupu

Formálne, vstup vyjadríme pomocou

- a, b = prirodzené čísla predstavujúce rozmery mriežky
- $H[1..a][1..b]$ = mriežka
- n = prirodzené číslo predstavujúce počet obdĺžnikov vo vnútri mriežky

Výstupom algoritmu je okrem celkového obvodu dielčích obdĺžnikov vyfarbená mriežka, t.j. mriežka, v ktorej každému elementu je priradené písmeno abecedy, ktoré jednoznačne identifikuje obdĺžnik, ktorého je bod súčasťou.

3 Implementácia sekvenčného riešenia

Primárny cieľ sekvenčného riešenia, na ktorom ďalej staviame v paralelnej implementácii, spočíva v nájdení a následnom prehľadaní celého stavového priestoru množiny potenciálnych riešení.

Veľkosť mriežky	T(n) [s]
15x15	29
20x20	473
21x21	571
23x23	983

Table 1: Trvanie sekvenčného výpočtu

Stavový priestor v našom prípade rozumieme množinu mriežok, v ktorých postupne spracúvame obdĺžniky s dvojicou parametrov

1. *shape* - veľkosť obdĺžnika (dĺžka x šírka)
2. *position* - pozícia v mriežke

Očividne, *shape* obdĺžnika je určený jeho plochou, zatiaľ čo *position* je vlastnosťou mriežky a ostatných obdĺžnikov v nej existujúcich - napr. obdĺžnik nedokážem vložiť do mriežky, pokiaľ jeden z jeho rozmerov presahuje veľkosť mriežky alebo je okolie obdĺžnika posiate inými, už zafixovanými obdĺžnikmi.

Popis sekvečného riešenia v skratke: procesor zo zásobníka mriežok vezme obdĺžnik, zistí, či má *shape* (v prípade negatívnej odpovede vygeneruje všetky dvojice *a*, *b* tak, aby ich súčin bol rovný očakávanej ploche). V ďalšom kroku procesor zistí, či daný obdĺžnik má stanovenú *position* v mriežke - ak nie, na zásobník uloží všetky prípustné možnosti s ohľadom na veľkosť mriežky a susedné obdĺžniky. Procesor pokračuje do spracovania posledného obdĺžnika alebo do nájdenia prvého obdĺžnika, ktorý nemožno umiestniť do mriežky. V prípade, ak sa všetky obdĺžniky podarilo umiestniť do mriežky, procesor porovná výsledok s doterajším minimom a zo zásobníka vezme ďalšiu mriežku a opakuje takto popísanú akciu.

V nami implementovanom triviálnom sekvenčnom riešení neuplatňujeme orezávanie neperspektívnych ciest - každá mriežka dobehne do konca v prípade existencie rozdelenia obdĺžnikov a až následne sa porovná výsledný obvod s doterajším minimom.

Trvanie sekvenčného výpočtu na výpočtovom klastri star.fit.cvut.cz je znázornené v tabuľke 1. Výsledný sekvenčný čas je určený ako aritmetický priemer trojice meraní. Rovnakú metodiku sme použili aj pri popisovaní výsledkov paralelného behu.

4 Príklad zadania a výsledku

Program spúšťame s parametrom -f, ktorý udáva cestu k súboru obsahujúce vstupné dáta. Pre triviálny vstup o veľkosti mriežky 5x5 dostávame výstup zobrazený na obrázku 1.

5 Implementácia paralelného riešenia

Paralelní algoritmus začíná tím, že Master načte vstupní data a začne řešit úlohu. Ve chvíli, kdy má dostatek práce na odeslání práce alespoň jednomu procesoru, rozdělí zá-

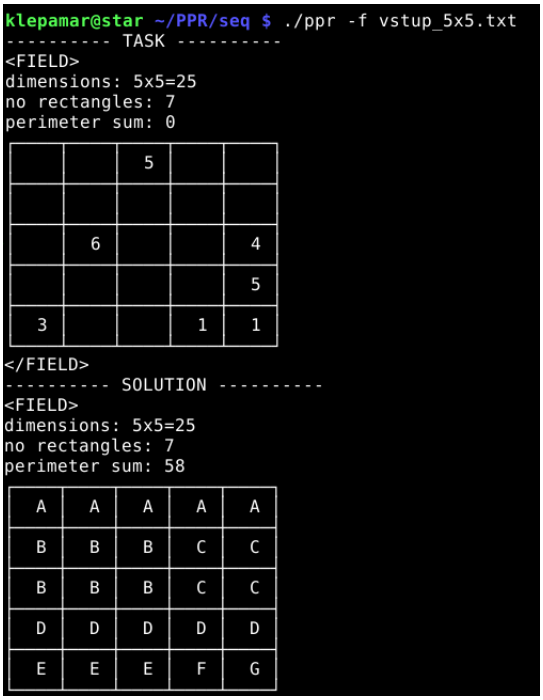

Figure 1: Príklad výstupu pre sekvenčnú úlohu

sobník podélně, přičemž se z každé úrovně stromu vezme jen jeden prvek. Oddělenou část zásobníku Master neblokujícím způsobem odesílá nenastartovanému procesoru. Jednotlivé procesory jsou tedy startovány postupně a obdrží dostatek práce. Procesory bez prvotní práce (neaktivní/čekající) čekají na zprávu s prací od Mastera a mezitím obsluhují i ostatní zprávy.

Nastartovaný procesor řeší úloho pomocí algoritmu ze sekvenčního řešení. Navíc je implementováno ořezávání stavového prostoru podle nejlepšího známého lokálního řešení. Procesy toto lokální nejlepší řešení posílají zároveň s odesílanou prací a je tedy zajištěna určitá propagace globálního nejlepšího řešení.

Pokud procesoru dojde práce, odešle token pokud ho drží (procesor si přijatý token drží do chvíle než má prázdný zásobník, aby nedocházelo ke zbytečnému posílání zpráv). Poté zažádá náhodného dárce o práci a dokud neobdrží zprávu s prací, obsluhuje přijaté zprávy a případně žádá nového dárce. Dárce se pokusí rozdělit zásobník u dna. Pokud se mu to povede, odesílá práci (a své lokální nejlepší řešení). Pokud nemá dostatek práce, aby mohl být zásobník rozdělen, odesílá zprávu bez práce a žádající hledá nového dárce.

Aktivní procesor každých 50 vyřešených DFS (vnější cyklus algoritmu - umístění jednoho obdélníku) obsluhuje přijaté žádosti o práci. Ostatní zprávy obsluhuje jen v případě prázdného zásobníku (ve volné chvíli kdy čeká na zprávu od dárce). Neaktivní/čekající procesor obsluhuje zprávy ve smyčce tak, že jsou obslouženy všechny

Veľkosť mriežky	Sieť	T(n,2)	T(n,4)	T(n,8)	T(n,16)	T(n,24)	T(n,32)
15x15	E	27.67	11.97	16.15	15.33	25.60	27.23
	I	45.03	20.60	21.73	21.07	30.93	25.23
20x20	E	126.07	68.83	42.43	47.37	70.87	52.17
	I	201.23	103.57	88.47	85.70	107.57	101.80
21x21	E	420.13	200.17	68.63	58.00	67.97	92.83
	I	633.60	318.57	118.40	88.57	93.10	113.63
23x23	E	78.67	13.30	25.73	40.63	44.17	50.47
	I	129.93	46.73	48.90	46.60	54.67	72.80

Table 2: Trvanie paralelného výpočtu

zprávy ve frontě a poté je procesor uspán na 50ms a to opakuje do chvíle než přijme nějakou práci. Tím se přepne se do stavu aktivní a počítá řešení.

Tento cyklus přijímání zpráv může být také ukončen přijmutím zprávy o ukončení výpočtu. V takovém případě procesor ihned ukončí obsluhou zpráv, ukončí algoritmus řešící úlohu a odesílá své nejlepší řešení Masterovi. Master tyto řešení sekvenčně přijme od každého procesoru a porovnáním zjistí konečné řešení.

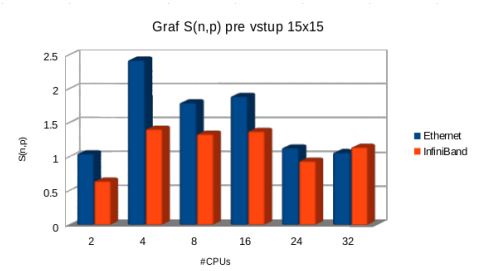

Figure 2: Paralelné zrýchlenie pre inštanciu problému 15x15

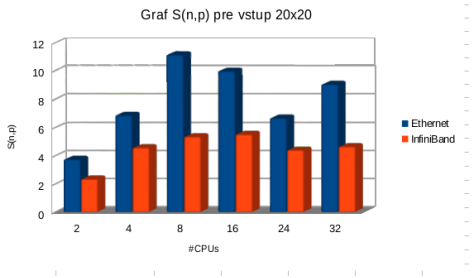

Figure 3: Paralelné zrýchlenie pre inštanciu problému 20x20

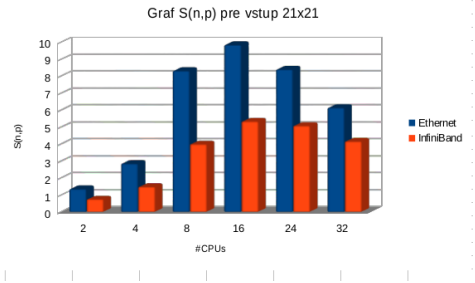

Figure 4: Paralelné zrýchlenie pre inštanciu problému 21x21

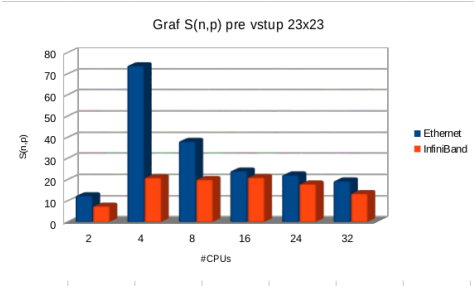

Figure 5: Paralelné zrýchlenie pre inštanciu problému 23x23

6 Vyhodnotenie

Záver tejto správy by sme vyhradili analýze dosiahnutých výsledkov. Už letmým porovnaním tabuľiek 1 a 2 je jasné, že v drvivej väčšine prípadov sme dosiahli stav, v ktorom paralelné spracovanie prebieha rýchlejšie ako sekvenčné riešenie. Zaujímavejšou otázkou preto zostáva hodnota zrýchlenia pre narastajúci počet procesorov pre všetky 4 analyzované riešenia.

Prvý vstup o veľkosti mriežky 15x15, ktorého dĺžka sekvenčného spracovania nepresiahla 0.5 minúty, dosiahla najvyššiu hodnotu zrýchlenia už pre malý počet procesorov, konkrétne 4 pre siete Ethernet a InifiniBand súčasne - viz obrázok 2.

Ďalšie vstupy o veľkosti mriežky 20x20 resp. 21x21, ktoré bežali na jednom procesore takmer 10 minút, dosiahli strop efektívnosti pri počte 8 resp. 16 procesorov. V porovnaní s menším vstupom 15x15 je tak zrejmé, že došlo k efektívnejšiemu využívaniu prostriedkov z dôvodu, že procesory pracovali nad dostatočne veľkým zásobníkom mriežok, vďaka čomu svoju prácu neukončili okamžite.

Z hľadiska efektívnosti je nemenej zaujímavý prípad vstupu 23x23, ktorý na rozdiel od predchádzajúcich vstupov nebol vybraný náhodne. Naopak, pri jeho výbere sme hľadali mriežku s 1 veľkou hodnotou plochy dielčieho obdĺžnika. Naviac sme žiadali, aby hodnota plochy bola deliteľná minimálným počtom čísel, a teda *shape* bol určený jednoznačne. Tieto predpoklady splnila mriežka obsahujúca hodnotu 121, ktorá sama zaberá približne 1/4 celkovej plochy. Vďaka nájdeniu takejto mriežky výsledný čas

sekvenčného riešenia spĺňal hornú hranicu behu - 15 minút.

Takto zvolený vstup 23x23 potom dosiahol pozoruhodné superlineárne zrýchlenie, napr. s použitím 2 procesorov došlo k zrýchleniu paralelného behu vyše 70-krát. Predpokladáme, že veľkú zásluhu na tejto hodnote má orezávanie neperspektívnych ciest na základe lokálne najlepšieho riešenia a posielanie najlepšieho známeho riešenia pri delení práce.

Zaujímavosťou nášho riešenia je fakt, že s výnimkou jedného prípadu - veľkosť mriežky 15x15, beh na 32 procesoroch - sa ako rýchlejší ukázal štandard Ethernet. Vzhľadom na to, že nemáme dostatok informácií o konkrétnej implementácii siete InfiniBand, by sme z tohto výsledku nedokázali vyvodiť žiadne závery.

7 Zdroje

1. prednášky na eduxe k predmetu MI-PAR

   `https://edux.fit.cvut.cz/courses/MI-PAR.2`

2. správa k zadaniu Othello

   `http://www.michaltrs.net/cvut_fel/36par/36PAR-othello-zprava.pdf`