Semestrální projekt MI-PDP 2016/2017:

Paralelní algoritmus pro řešení problému ZBS

Jaroslav Hrách
magisterské studijum, FIT ČVUT, Thakurova 9, 160 00 Praha 6
10. května 2017

1 Definice problému a popis sekvenčního algoritmu

1.1 Vstupní data

n – přirozené číslo představující počet uzlů grafu G, $n \geq 10$,

k – přirozené číslo řádu jednotek představující průměrný stupeň uzlu grafu G, $n \geq k \geq 3$,

$G(V, E)$ – jednoduchý souvislý neorientovaný neohodnocený graf o n uzlech a průměrném stupni k

a – přirozené číslo, $5 \leq a \leq \frac{n}{2}$,

1.2 Úkol

Nalezněte rozdělení množiny uzlů V do dvou disjunktních podmnožin X a Y tak, že množina X obsahuje a uzlů, množina Y obsahuje $n - a$ uzlů a počet všech hran $\{u, v\}$ takových, že u je z X a v je z Y (čili velikost hranového řezu mezi X a Y), je minimální.

1.3 Výstup algoritmu

Výpis disjuktních podmnožin uzlů X a Y a počet hran tyto množiny spojující.

1.4 Popis vstupu

Algoritmus dostane na vstupu matici přechodů s číslem a. Z matice přechodů se vypočítá počet všech hran, které uzly propojují, což slouží jako maximální hodnota, kterou se algoritmus snaží snížit na minimální možnou hodnotu.

2 Popis sekvenčního algoritmu

Sekvenční algoritmus je implementován prohledáváním stavového prostoru, konkrétně prohledáváním do hloubky. Algoritmus obsahuje rekurzivní funkci, která postupně generuje všechny možné varianty uzlů a v případě odpovídající velikosti množiny podle a zkontroluje, kolik je mezi jednotlivými uzly hran.

```
if počet prvků < a then
    while generuj všechny varianty do
        if počet uzlů varianty == a then
            spočítej cenu(počet hran);
            if nová cena < nejnižší cena then
                ulož variantu a nejnižší cenu
            end
        end
    end
end
```

3 Popis paralelního algoritmu a jeho implementace v OpenMP

3.1 OpenMP - verze s task paralelismem

Jedná se v podstatě o stejný algoritmus jako v případě sekvenčního, jen je doplněn o OpenMP direktivy pro task paralelismus. Zde je potřeba řešit kritickou sekci při hledání nejlepšího řešení. Dále se na začátku běhu vypočítá práh, do kterého se úlohy paralelizují. V našem případě to je $threshold = a - floor(\frac{a}{3})$.

3.2 OpenMP - verze s datovým paralelismem

Datový paralelismus kombinuje prohledávání do šířky do hloubky. Algoritmus nejprve nageneruje množinu řešení a uloží je do fronty. Následně ve *while* cyklu vybere každé možné řešení a to zpracuje paralelní pomocí OpenMP direktivy for a řeší je klasickým sekvenčním algoritmem. Po řadě testování byla použita klauzule $schedule(dynamic)$, poněvadž vracela správné řešení v nejlepším možném čase.

4 Popis paralelního algoritmu a jeho implementace v MPI

Implementace paralelního algoritmu v MPI vychází z datového paralelismu a je použit moder master-slave. Master proces na začátku vygeneruje první "patro", které rozděluje mezi všechny slave procesy. Slave procesy komunikují přes značky a když dokončí práci, dají vědět master procesu, že jim může poslat další práci, pokud ještě nějaká zbývá. Slave proces pak funguje stejně jako v datovém paralelismu s OpenMP. Ve frontě si nageneruje několik možných řešení a ty zpracovává přes direktivu for.

5 Naměřené výsledky a vyhodnocení

Měření probíhalo na výpočetním klasteru Star. Pro výpočet byly vybrány tři instance, jejichž sekvenční zpracování se časově pohybovalo v jednotkách minut. Pro výpočet času u variant s OpenMP byl použit běžný shellovský příkaz *time*. V případě MPI byla pro výpočet běhu

programu použita funkce *MPI_Wtime*. Měření bylo provedeno pro sekvenční řešení, následně pro jednu instanci pro obě OpenMP varianty s různým počtem vláken a samozřejmě také pro MPI s různým počtem uzlů.

5.1 Sekvenční řešení

Nalezení vhodných instancí, které by běžely v rozumném čase, bylo poněkud komplikované. Nakonec byla zvolena jedna instance s dlouhých časem a dvě s výrazně kratším časem.

Datový soubor	Čas [s]
d30-10.dat	88.0705
d50-7.dat	202.135
d30-15.dat	1020.40

Tabulka 1: Doba výpočtu sekvenčního řešení

5.2 OpenMP - task a datový paralelismus

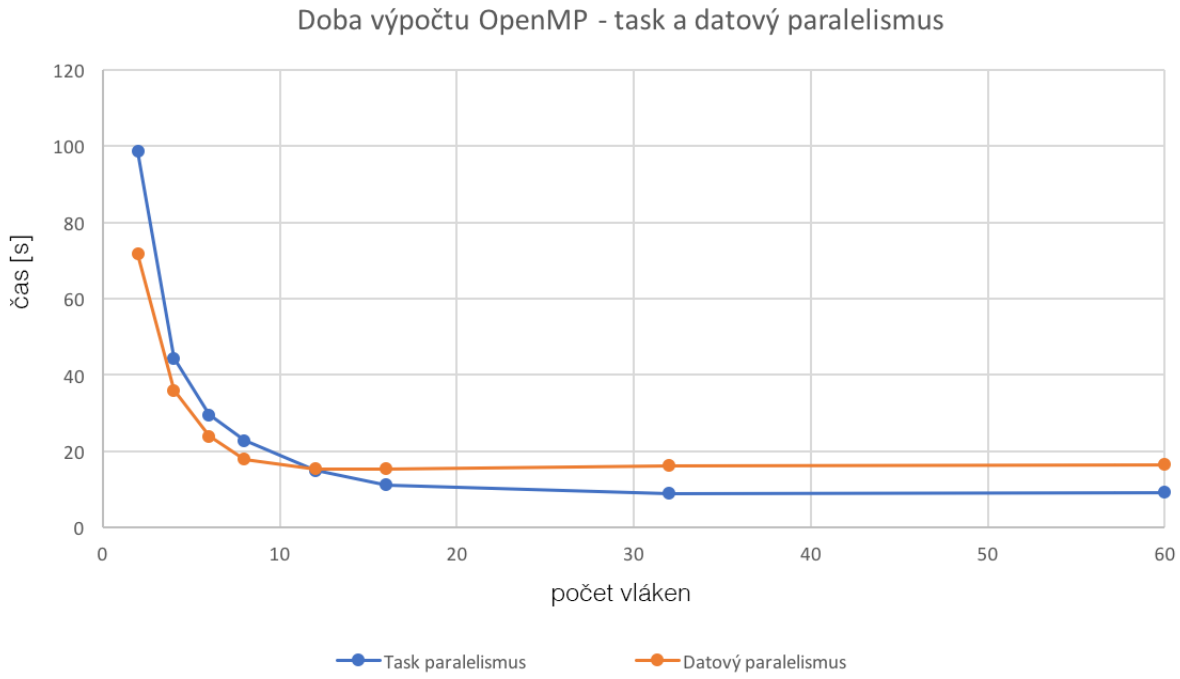
Pro měření výpočtu pod OpenMP byla vybrána instance **d50-7.dat**, jejíž sekvenční výpočet trvá přes tři minuty. Jak je vidět z níže přiložené tabulky a grafu, podařilo se výpočet zrychlit celkem výrazně. Task paralelismus se s počtem vláken neustále zrychluje, nicméně datový paralelismus běží rychleji. U datového paralelismu se však ukázalo, že se rychlost výpočtu s větším počtem vláken dříve ustálí.

Počet vláken	Task paralelismus - čas [s]	Datový paralelismus - čas [s]
2	98,516	71,748
4	44,43	35,953
6	29,608	23,988
8	22,828	17,982
12	14,981	15,373
16	11,292	15,388
32	9,052	16,297
60	9,314	16,587

Tabulka 2: Doba výpočtu OpenMP - task a datový paralelismus

5.3 MPI

Měření probíhalo na třech instancích pro různý počet uzlů a vláken. U některých instancí byla změna rychlosti výpočtu výrazná, u některých se však žádné velké změny nestaly.



Obrázek 1: Grafické znázornění výpočtu OpenMP - task a datový paralelismus

Počet vláken	d30-10.dat	d50-7.dat	d30-15.dat
2	89,14	431,97	1276,59
4	89,18	143,03	276,59
8	89,15	143,29	276,59
16	7,51	11,99	143,29
24	6,79	8,01	131,36
32	6,78	6,21	131,22
48	6,50	6,21	111,65
60	6,41	3,69	100,50

Tabulka 3: Doba výpočtu MPI

6 Závěr

V semesetrální práci jsme si na složité úloze ukázali, jak ji ze sekvenčního řešení lze paralelizovat hned několika způsoby. Nejprve jsme vyzkoušeli knihovnu OpenMP a dvě její varianty - task paralelismus a datový paralelismus. Task paralelismus byl na implementaci naprosto triviální. U datového paralelismu bylo potřeba program trochu pozměnit, nicméně výsledné zrychlení bylo ještě lepší než u task paralelismu. Poslední částí bylo programování pomocí knihovny MPI zkombinované s OpenMP. K řešení úlohy byl využit vypočetní cluster STAR.

V semestrální úloze jsme se naučili navrhnout paralelní algoritmus, naprogramovat ho, otestovat ho, následně vyzkoušet jeho běh s využitím více procesorů a v samém závěru ho analyzovat a provést měření.

7 Literatura

Zobecněná bisekční šířka - implementace. Github [online]. [cit. 2017-05-10]. Dostupné z: https://github.com/izmy/PDP-zobecnena-bisekcni-sirka