

Semestrální projekt PAR 2009/2010:

Paralelní algoritmus pro řešení problému bipartitního podgrafu

Petr Smejkal
David Vavrousek

5. ročník, obor počítače, K336 FEL CVUT, Karlovo nám. 13, 121
35 Praha 2

May 11, 2010

1 Definice problému a popis sekvencního algoritmu

1.1 Definice problému BPG: bipartitní podgraf

1.1.1 Vstupní data

n = přirozené číslo představující počet uzlů grafu, $n \geq 5$

k = přirozené číslo řádu jednotek představující maximální stupeň uzlů grafu

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

Doporučení pro algoritmus generování G :

Použijte generátor grafu Generator1 s typem grafu "-t 1", který vygeneruje souvislý neorientovaný neohodnocený graf.

1.1.2 Definice

Graf $G(V,E)$ je bipartitní, jestliže můžeme rozdělit množinu uzlů na disjunktní podmnožiny U a W tak, že každá hrana v G spojuje uzel z U s uzlem z W . Bipartitní graf je možné uzly obarvit 2 barvami.

1.1.3 Úkol

Graf $G(V,E)$ je definován množinou uzlů V a množinou hran E . Úkolem je nalézt maximální podmnožinu hran F takovou, že graf $G(V,F)$ je bipartitní (viz obrázek).

1.2 Popis sekvencního algoritmu

Základním algoritmem bylo procházení stavového prostoru do hloubky. Hloubka stavového stromu je omezena na $|E|$. Pro reprezentaci stavu jsme si vytvořili třídu State, která obsahuje matici sousednosti grafu. Pripustný mezistav je podmnožina hran F , která tvoří bipartitní podgraf $G(V,F)$. Cena, kterou maximalizujeme, je počet hran v F . Pokud je graf G bipartitní, pak triviálně $F=E$,

namerena data	
instance (minut)	1
InfiniBand	
5	278,51
10	610,11
15	870,93
Ethernet	
5	277,60
10	615,88
15	883,11

Figure 1: namerena data pro jeden uzel

jinak F je podmnožinou E . Vstupní graf nejprve otestujeme, zda je bipartitní (lineární algoritmus) a pokud ne, použijeme algoritmus prohledávání do hloubky. Základní myšlenkou algoritmu je postupně odebírání hran. Na zásobník se vloží počáteční stav, reprezentující zadaný graf. Dale se postupuje podle algoritmu prohledávání do hloubky, při expanzi se na zásobník vloží 2 nové stavy, jeden s hranou n a jeden bez ní. Celkový počet hran n určuje maximální hloubku stromu reprezentujícího graf stavového prostoru. Pro ořezávání stavového prostoru využíváme techniky branch and bounds, celý algoritmus je tedy DFS-BB. Jedná se o úplné prohledávání stavového prostoru do hloubky $|E|-|V|$. Prohledávání se může navracet v mezistavech s $|F|=|V|-1$ (strom je triviálně bipartitní, proto musí existovat řešení s $|F|=|V|-1$). Triviální dolní mez je $|F|=|V|-1$. Tesná horní mez není známa. Triviální horní mez je $|F|=|E|$.

2 Popis paralelního algoritmu a jeho implementace v MPI

Základem paralelního algoritmu je sekvencní algoritmus. S tím rozdílem, že stavový prostor se disjunktne rozdělí mezi více procesorů, které ho samostatně zpracovávají. Kvůli omezujícím podmínkám algoritmu DFS-BB si procesory mezi sebou posílají nalezená nová nejlepší řešení. Jednotlivé procesory mají svůj vlastní lokální zásobník a používá se dynamické vyvazování výpočetní zátěže (load balancing). Vyvazování zátěže je implementováno pomocí 2 algoritmu: algoritmu pro dělení zásobníku (ADZ) a algoritmu pro hledání darce (AHD), kterými nečinný procesor s prázdným lokálním zásobníkem (stav $I=idle$) získává práci od vhodného darce, což je aktivní procesor s neprázdným zásobníkem (stav $A=active$), jehož zásobník odpovídá dostatečně velkému stavovému podprostoru. Procesor si o práci zadá procesor s číslem o jedno větší, pokud ten práci

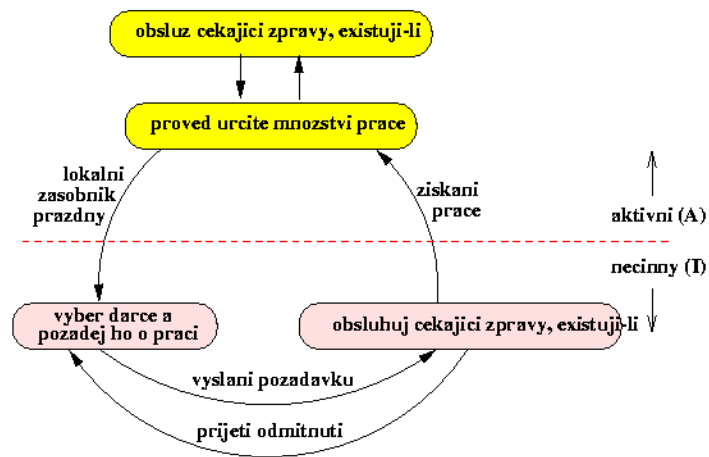


Figure 2: sekvenční

nema, zvysí se citac o jednicku, takto cyklicky zada procesory dokud neobdrží práci nebo požadavek na ukončení. Na počátku procesor P1 zna počet procesoru p, na kterých se úloha bude řešit. Provede dostatečný počet expanzí počátečního stavu, rozdelí svůj zásobník na p částí a rozesle jednotlivé části ostatním procesorům. Všechny procesory začnou prohledávat svůj přidělený podprostor a přitom realizují programovou smyčku naznačenou na tomto obrázku.

Aktivní procesor, který vycerpa svůj přidělený díl práce, se stane nečinný a pomocí algoritmu pro hledání darce AHD vybere darce a pošle mu žádost o práci. Pokud mu darce práci pošle, přepne se zpět do stavu aktivní. Pokud se mu vrátí odmítnutí, vše se opakuje. Procesor ve stavu nečinný musí také periodicky kontrolovat, zda nema ve frontě žádosti o práci, na které odpovídá odmítnutím, nebo zprávy o nalezení řešení jiným procesorem. V případě, že se jedna o optimální řešení, jsou to v podstatě žádosti o ukončení výpočtu. Aktivní procesor provádí fixní objem práce nad lokálním zásobníkem (expanduje určitý daný počet stavu na lokálním zásobníku) a pak kontroluje frontu zpráv. V té mohou být žádosti o práci, informace o nalezení řešení jiným procesorem a žádosti o ukončení výpočtu. Aktivní procesor zpracovává žádosti o práci následujícím způsobem: Pokud je velikost lokálního zásobníku větší než nastavená minimální mez, tak se zásobník rozdelí na polovinu a jedna část se zasle zadateli o práci. Pokud není, tak procesor požadavek odmítne.

3 Analýza složitosti paralelního řešení

3.1 Analýza sekvencního řešení

Složitost sekvencního řešení lze rozdělit na dvě části. Nejprve na to, kolik stavů se projde, cili jak je velký stavový prostor a pro každý stav se pak ještě provádí algoritmus BFS pro testování bipartitnosti. Orezávání algoritmem Branch and Bound se nedá odhadnout, proto ho neuvažujeme. Vysoká stromová stavová prostorová odpovídá počtu hran. V každém kroku jednu hranu buď odeberu, nebo nechám.

Počet prohledávaných stavů je tedy:

$$\sum_{i=0}^{|E|} 2^i = 2^{|E|} + 2^{|E|-1} + \dots + 2^0 = O(2^{|E|})$$

Test bipartitnosti se provádí procházením do šířky. Složitost tohoto algoritmu je :

$$O(|E| + |V|)$$

Celková složitost je tedy:

$$SU(|E|, |V|) = O(|E| + |V|) \times O(2^{|E|}) = O(2^{|E|} \times (|E| + |V|))$$

3.2 Analýza paralelního řešení

Při paralelním algoritmu se nejprve expanduje prvních p stavů, které se rozřeší, potom si p procesorů rozdělí stejnou práci, jakou měl sekvencní algoritmus. Toto tvoří výpočetní složku složitosti. Vedle té ještě musíme započítat složku komunikace, která vyjadřuje komunikaci mezi procesory a předávání práce. Tato složka je už zcela závislá na rovnoměrném rozdělení práce a poloze hledaného řešení ve stavovém prostoru. Konkrétně tuto složku tedy vyjádřit nemůžeme, nicméně rozhodně není zanedbatelná, proto ji ve vzorci uvádíme symbolicky.

Paralelní čas tedy je:

$$T(|E|, |V|, p) = T_v(|E|, |V|, p) + T_k(|E|, |V|, p) = O\left(p + \frac{2^{|E|} \times (|E| + |V|)}{p}\right) + T_k(|E|, |V|, p)$$

Teoretická paralelní cena:

$$C(|E|, |V|, p) = p \cdot T(|E|, |V|, p) = O(p^2 + (2^{|E|} \times (|E| + |V|)) + T_k(|E|, |V|, p) \times p)$$

Efektivnost:

$$E(|E|, |V|, p) = \frac{SU(|E|, |V|)}{C(|E|, |V|, p)} = \frac{O(2^{|E|} \times (|E| + |V|))}{O(p^2 + (2^{|E|} \times (|E| + |V|))) + T_k(|E|, |V|, p) \times p}$$

A zrychleni:

$$S(|E|, |V|, p) = \frac{SU(|E|, |V|)}{T(|E|, |V|, p)} = \frac{O(2^{|E|} \times (|E| + |V|))}{O(p + \frac{2^{|E|} \times (|E| + |V|)}{p}) + T_k(|E|, |V|, p)} = O(p + \frac{2^{|E|} \times (|E| + |V|)}{p}) + T_k(|E|, |V|, p)$$

4 Namerene vysledky a vyhodnoceni

Z namerenych dat (Figure 3 a 4) je videt, ze namerene casy pro site InfiniBand a Ethernet jsou temer stejne, lisi se v radu par sekund. Vyssi casy jsou prevazne u site Ethernet. S ohlednutim na mozne spozdeni, kter nemusi vzniknout primo algoritmem, lze rici, ze namerene casy jsou pro obe site stejne. To znamena, ze se neprenasi velke mnozstvi zprav, ktere by algoritmus zpomalovaly.

Z grafu, znazornujici zrychleni (Figure 5 a 6) je videt, ze je na zaklade namerenych dat podobne pro obe site. Nejlepsi rovnomerne zrychleni pro zvolene instance se zda byt pro 12 vypocetnich uzlu.

4.1 Vyhodnoceni

V teto uloze jsem nejdrive implementovali sekvenčni algoritmus pro hledani bipartitního podgrafu. Nasledne jsme implementovali paralelni reseni. Toto reseni jsme otestovali na datech ziskanych ze sekvenčního mereni, kde instance bezeli 5, 10 a 15 minut. Tyto instance jsme otestovali na paralelním algoritmu. Z vyslednych grafu lze rici, ze se algoritmus chova spravne a s poctem vypocetnich uzlu klesa vypocetni cas.

namerena data	pocet procesoru					
instance (minut)	1	2	4	8	12	16
InfiniBand						
5	278,51	182,30	45,12	27,94	21,39	16,73
10	610,11	311,41	173,57	111,56	43,72	36,11
15	870,93	455,01	267,05	175,11	63,38	50,12
Ethernet						
5	277,60	182,13	45,20	28,44	19,06	16,81
10	615,88	314,21	175,39	112,22	45,78	36,81
15	883,11	453,81	269,91	179,62	63,08	52,19

Figure 3: tabulka namerenych vysledku

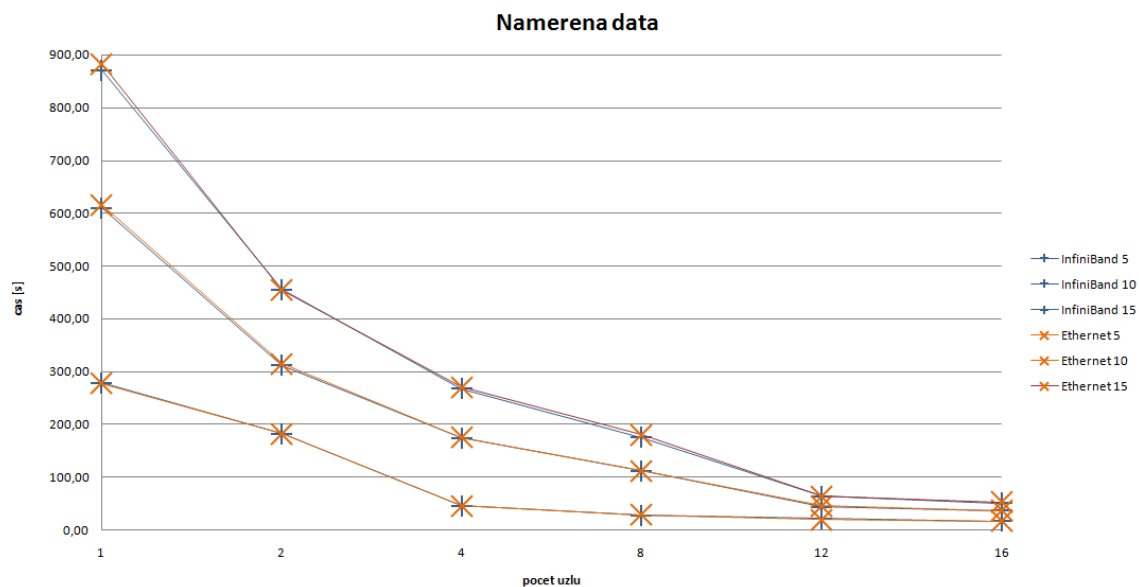


Figure 4: graf namerenych vysledku

zrychleni	pocet procesoru					
instance (minut)	1	2	4	8	12	16
InfiniBand						
5	1,00	1,53	6,17	9,97	13,02	16,64
10	1,00	1,96	3,52	5,47	13,96	16,89
15	1,00	1,91	3,26	4,97	13,74	17,38
Ethernet						
5	1,00	1,52	6,14	9,76	14,57	16,52
10	1,00	1,96	3,51	5,49	13,45	16,73
15	1,00	1,95	3,27	4,92	14,00	16,92

Figure 5: tabulka zrychleni

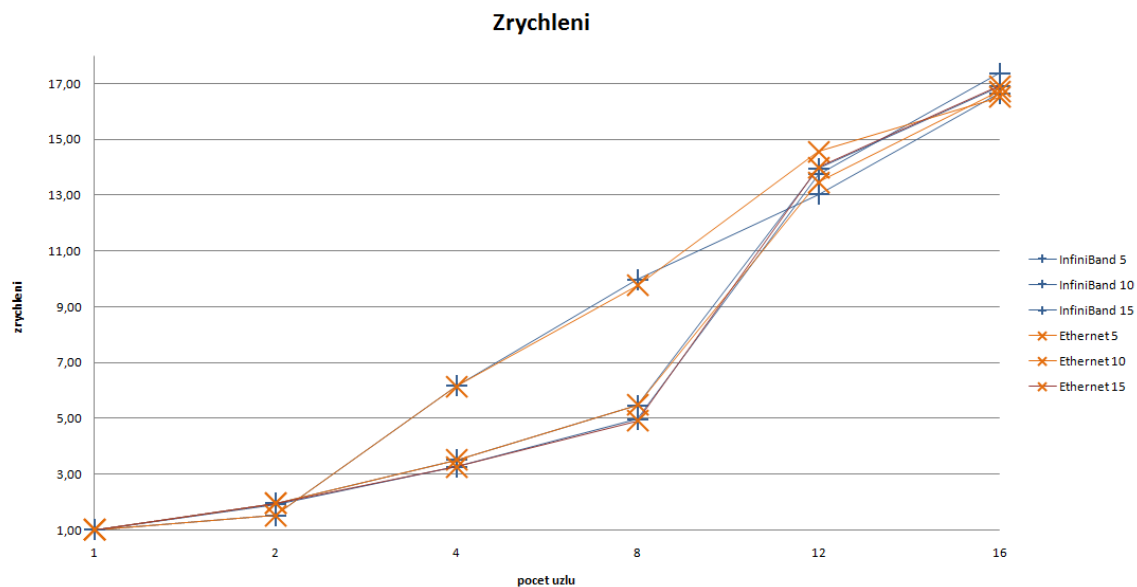


Figure 6: graf zrychleni