Semestralní projekt MI-PPR 2015/2016:

Paralelní algoritmus pro řešení problému ZBS

Valentin Banshchikov
Michal Kápar
magisterské studijum, FIT ČVUT, Kolejní 550/2, 160 00 Praha 6

December 22, 2015

1 Definice problému

1.1 Vstupní data

a – přirozené číslo,

n – přirozené číslo představující počet uzlů grafu G, $5 \leq n$,

m – přirozené číslo představující počet hran grafu G, $n \leq m$,

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

$G(V, E)$ – jednoduchý souvislý neorientovaný neohodnocený graf o n uzlech a m hranách.

1.2 Úkol

Naleznout rozdělení množiny n uzlů grafu G do dvou disjunktních podmnožin X a Y tak, že podmnožina X obsahuje a uzlů, podmnožina Y obsahuje $n - a$ uzlů a počet všech hran $\{u, v\}$ takových, že u je z X a v je z Y, je minimální.

1.3 Výstup algoritmu

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

1.4 Popis vstupu

Algoritmus dostává na vstup matici přechodů s číslem a. Matice je generována z aplikace `generator`, ta vytvoří matici na základě zadané velikosti a stupně grafu. Následně je na výsledek uplatněna aplikace `souvislost`, ta má za cíl z nesouvislého grafu vygenerovat souvislý graf.

2 Popis sekvenčního algoritmu a jeho implementace

Sekvenční algoritmus je typu `BB-DFS` (Branch-and-bound Depth-first search). Při hledání stavového prostoru hledáme nejmenší cenu řešení. Není možné žádné stavy vynechat. Počáteční cena je nastavena na nekonečno (přesněji `UINT_MAX`).

Implementace je popsána v následujícím pseudokódu:

```
1. MIN_HRAN = nekonečno
2. ALL_HRAN = (1, 2, 3, ..., n)
3. Vytvoř 1. kombinace v lexikografickem pořadí a zapíš
   do proměny COMB
4. Vytvoř doplněk a zapíš do proměny COMP:
   COMP = ALL_HRAN \ COMB
5. Spočti počet hran mezi množinou COMB a COMP
6. Pokud je počet hran menší, než MIN_HRAN:
     MIN_HRAN = nový počet hran
7. Zapíš do COMB následující kombinace v lexikografickem pořadí:
     Pokud taková existuje:
       GOTO 4
     Pokud taková neexistuje:
       Výsledkem je MIN_HRAN
```

Na základě objemu zvolených dat vypočteme asymptotickou složitost:

$$O(n) = \binom{n}{a} \cdot ((n-a) \cdot a + 2 \cdot (n+a) - 1)$$

$\binom{n}{a}$ je počet kombinací v podmnožině X, pro které se hledá minimum vazeb mezi podmnožinami X a Y. Násobení $(n-a)\cdot a$ představuje hledání hrany mezi podmnožinami X a Y. $2\cdot(n+a)-1$ je složitost výpočtu doplňku.

3 Popis paralelního algoritmu a jeho implementace v MPI

Paralelní algoritmus je typu `PBB-DFS-V`. To znamená, že každý proces ví, jaká je horní mez (při startu uložena v dolní mezi) a lokálně udržuje informaci o svém dosud nejlepším řešení. Pomocí paralelní redukce se ze všech nejlepších lokálních řešení vybere globálně nejlepší.

Implementace je popsána v následujícím pseudokódu:

```
1. LOCAL_MIN_HRAN = nekonečno
2. ALL_HRAN = (1, 2, 3, ..., n)
3. Každý proces vypočte podle svého čísla první a poslední
   kombinace a zapíše je do proměn LOCAL_COMB a LOCAL_STOP
4. Vytvoř doplněk a zapíš do proměny LOCAL_COMP:
   LOCAL_COMP = ALL_HRAN \ LOCAL_COMB
5. Spočti počet hran mezi množinou LOCAL_COMB a LOCAL_COMP
6. Pokud je počet hran menší, než LOCAL_MIN_HRAN:
     LOCAL_MIN_HRAN = nový počet hran
7. Pokud LOCAL_COMB se rovna LOCAL_STOP:
     Čekej na ostatní procesy
   Pokud ne:
     Zapíš do LOCAL_COMB následující kombinace v lexikografickem pořadí:
     Pokud taková existuje:
       GOTO 4
     Pokud taková neexistuje:
       Čekej na ostatní procesy
8. Pošle LOCAL_MIN_HRAN procesu 0
9. Pokud je proces 0
     Spočte globální minima a vypíše výsledek
```

Ideální čas výpočtu je následující:

$T(n,p)_{vyp} = \frac{\binom{n}{a}}{p} \cdot ((n-a) \cdot a + 2 \cdot (n+a) - 1)$ – část samotného výpočtu

$T(n,p)_{br} = \alpha log_2 p$– binární redukce výsledku

Celkově tedy $T(n,p) = \frac{\binom{n}{a}}{p} \cdot ((n-a) \cdot a + 2 \cdot (n+a) - 1) + \alpha log_2 p$

4 Naměřené výsledky a vyhodnocení

Pro měření byl vybrán graf o velikosti 35 vrcholů. Pro výpočet není důležité, abych byl graf řídký nebo hustý. Pro každou podmnožinu, kterých je $\binom{n}{a}$ se kontrolují všechny hrany. V následující tabulce jsou vyneseny rychlosti výpočtu v sekundách pro zvolenou podmnožinu grafu velikosti a.

	a=11	**a=12**	**a=13**
1	251.944	529.863	967.434
2	148.523	299.84	533.992
4	72.56	152.613	269.537
8	35.674	76.163	136.131
12	24.248	51.47	90.065
16	18.373	37.966	70.219
20	15.129	30.452	56.116
24	12.267	25.1	46.577

Table 1: Tabulka naměřených hodnot v sekundách pro graf s počtem vrcholů 35 a rozdílné velikosti podmnožiny a

Pro porovnání, že je výpočet dostatečně náročný pro zvolená a, slouží následující graf, ve kterém jsou vyneseny všechny časy výpočtů.

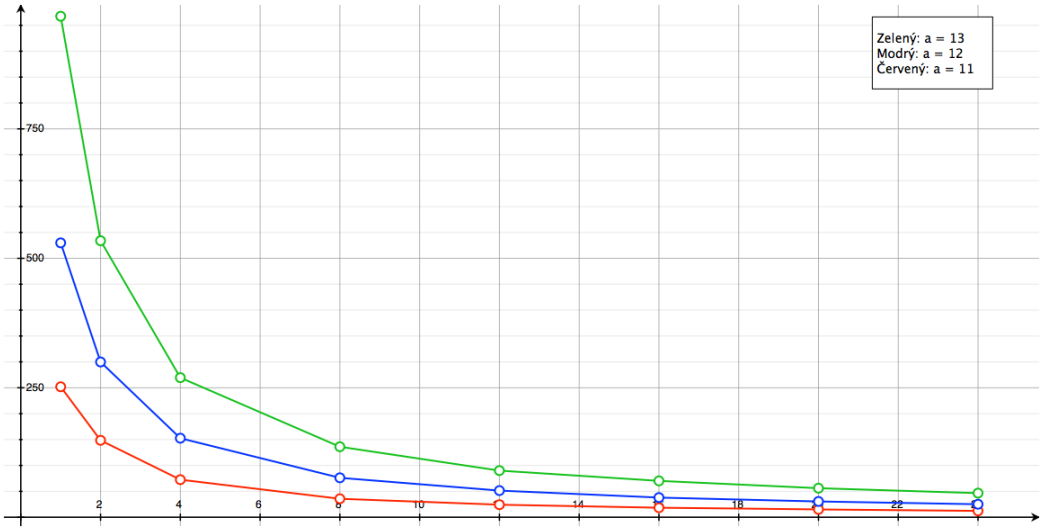

Figure 1: Graf všech výpočtů

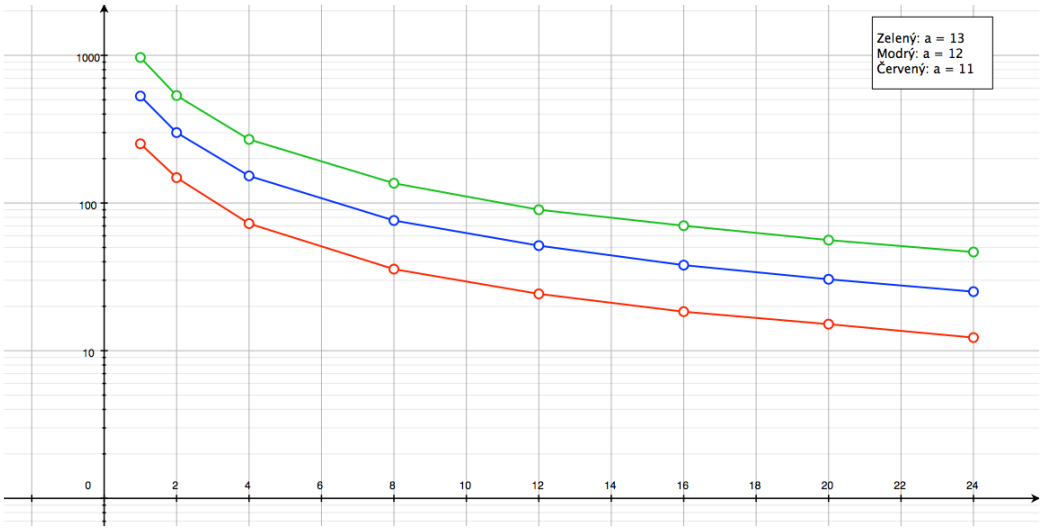

Figure 2: Graf všech výpočtů kde osa y je logaritmická

5 Závěr

V semestrální práci jsme si vyzkoušeli programování složitější úlohy, která lze paralelizovat k tomu jsme využili knihovnu MPI. K řešení úlohy jsme využívali výpočetní cluster STAR, propojený přes Ethernet s maximem 24 procesorů. Při zvyšování procesorů jsme se dostali k linearnímu zrychlení, které jsme podle typu úlohy a implementace našeho algoritmu čekali.

V semestru jsme si vyzkoušeli analyzovat a navrhovat jak rešit úlohy s využitím více procesorů, přes počáteční přerozdělení problému, následnou komunikaci a výpočet na jednotlivých procesorech až po zavěrečný sběr výsledků.