Semestrální projekt MI-PPR.2 2014/2015
Paralelní algoritmus pro řešení problému

Karel Fiala
Michal Kučera

České vysoké učení technické v Praze
Fakulta informačních technologií
Thákurova 9, 160 00 Praha 6
Česká republika

2. prosince 2014

Obsah

1 Definice problému a popis sekvenčního algoritmu

1.1 Úloha PEK: Permutace číselných koleček

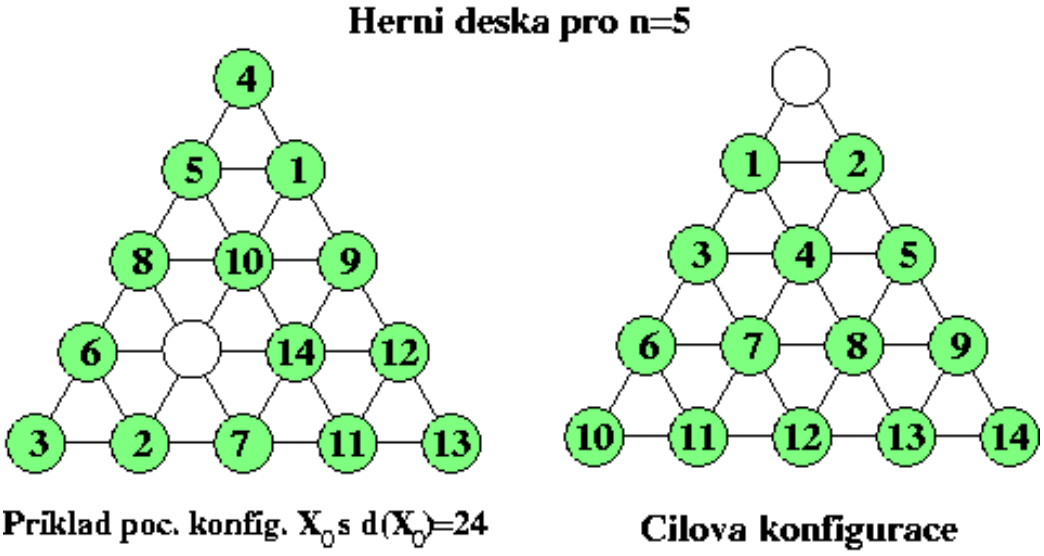
1.1.1 Vstupní data

n = délka rovnostranného trojúhelníka, $n >= 5$
q = přirozené číslo, $n^2 > q$
X_0 = počáteční konfigurace zkonstruovaná zpětným provedením q náhodných tahu z cílové konfigurace. Platí $q >= d(X_0)$.

```
./balls.out <n> <q> <data file>
```

1.1.2 Pravidla a cíl hry

Herní deska má tvar rovnostranného trojúhelníka o délce strany n, kde v i-tem řádku je i políček, ležících na průsečících úseček, rovnoběžných se stranami trojúhelníka. V těchto políčkách jsou podle určité permutace rozmístěna kolečka s čísly $1, \ldots, M-1$, kde $M = n(n+1)/2$. Jedno políčko zůstává volné, viz příklad na obrázku 1.



Obrázek 1: Hrací plocha

Tomuto rozmístění koleček budeme říkat počáteční konfigurace X_0. Jeden tah je přesun kolečka na sousední volné políčko ve směru některé úsečky.

Cílem hry je použitím minimálního počtu tahů převést počáteční konfiguraci X_0 do cílové konfigurace C, ve které jsou kolečka seřazena vzestupně po řádcích tak, že políčko na horním vrcholu trojúhelníkové desky je volné, viz obrázek vpravo. Úloha má vždy řešení.

1.1.3 Definice

Je-li X konfigurace rozmístění všech koleček na herní desce, pak t_X je počet doposud provedených tahů, kterými jsme převedli počáteční konfiguraci X_0 do konfigurace X.

d_X je spodní mez počtu tahů, kterými se lze dostat z konfigurace X do cílové konfigurace C. Tato spodní mez je rovna součtu vzdáleností koleček od jejich cílových políček. Vzdálenost 2 políček v této síti se počítá takto: Jsou-li obě políčka na úsečce rovnoběžné se stranou trojúhelníka, pak je vzdálenost rovna jejich lineární vzdálenosti po této úsečce. V opačném případě tvoří políčka vrcholy kosodélníka a vzdálenost se rovná součtu délek jeho dvou stran. Spodní mez počtu tahů nejlepšího možného řešení je tedy d_{X_0}.

Generování počátečního stavu: X_0 vygenerujeme nejprve q náhodně provedenými zpětnými tahy z cílové konfigurace C.

1.1.4 Výstup algoritmu

Výpis nejkratší posloupnosti tahů vedoucí z počáteční konfigurace do cílové konfigurace.

Odchylka od zadání: Výpis nejmenšího počtu tahů vedoucí z počáteční konfigurace do cílové konfigurace.

1.1.5 Sekvenční algoritmus

Sekvenční algoritmus je typu BB-DFS s neomezenou hloubkou stromu konfiguraci. Přípustný stav je cesta z počáteční do cílové konfigurace C. Cena, která se minimalizuje, je počet tahů takové cesty.

Horní mez počtu tahů je q.
Dolní mez je d_{X_0}.

1.1.6 Paralelní algoritmus

Paralelní algoritmus je typu L-PBB-DFS-D.

Odchylka od zadání: Implementovali jsme algoritmus typu G-PBB-DFS-D. Důvodem bylo zrychlení hledání řešení, pomocí dynamického zkracování prohledávaného prostoru.

2 Popis paralelního algoritmu a jeho implementace v MPI

2.1 Odlišnost od sekvenčního algoritmu

Paralelní algoritmus se výpočetně neliší od sekvenčního algoritmu. O celou komunikaci v MPI se stará jedna funkce, která je volána ze „sekvenčního algoritmu“ v případě, že

- nemá proces práci (zásobník je prázdný)
- proces nalezl nové řešení
- proces vykonal XYZ kroků

2.2 Směr komunikace

Celá komunikace probíhá jedním směrem v kruhu. Procesy žádají o prácí proces číslo $MyRank - 1$ a naopak práci posílají procesu s číslem $MyRank + 1$.

2.3 Práce se zásobníkem

Zásobník se spravedlivě půlí a posílá se vždy spodní část zásobníku. Tím nedochází k situace, že by nejlehčí konfigurace byla zpracována až jako poslední. Díky rozdělení výpočetně naročnějších konfigurací ze dna zásobníku také klesá potřeba komunikace a přerozdělování práce, což se pozitivně odráží na lineárním zrychlení paralelního algoritmu.

2.4 Ukončení výpočtu

Ukončení výpočtu je realizováno pomocí *token* ve formě čítače. Každý proces, který od doby inicializace žádosti o ukončení dostane práci, najde výsledek nebo „uslyší“ o novém výsledku, tento čítač resetuje na počáteční hodnotu a nedojde tak k předčasnému ukončení výpočtu. O inicializaci žádosti o

ukončení a případně o ukončení výpočtu vždy rozhoduje proces s číslem 0.

2.5 Parametry algoritmu

Zapne dynamického zkracování prohledávaného prostoru na základě již známých výsledků. Tento parametr však „maskuje“ linearitu zrychlení a proto byl pro potřeby měření vypnut.

```
#define SMART_SEARCH 1
```

Parametr, který určuje frekvenci komunikace. Jedná se o počet konfigurací, které budou vyřešeny před zavolám komunikačního okénka.

```
#define COMWIN_REQ_STEPS 100
```

2.6 Spouštění

Program spouštíme

```
./balls.out <n> <q> <data file>
```

3 Naměřené výsledky a vyhodnocení

3.1 Ethernet vs. Infiniband

Na námi navrženém algoritmu se rozdíl mezi komunikační sítí **Infiniband** $(41,4s)$ a **Ethernet** $(39,8s)$ neprojevil. Všechny rozdíly měření byly menší než odchylka měření. Vysvětlujeme si to tím, že náš algoritmus přenáší pouze nezbytně nutné a výpočetně výhodné informace a také heterogenní prostředí clusteru STAR s sebou přináší velkou odchylku měření (až 8%).

3.2 Konfigurace měření

3.2.1 Spouštění

Všechny uvedené hodnoty byly naměřeny s komunikační sítí **Infiniband** a spuštěny pomocí:

```
qrun.sh 24c #CPU long <skript s konfigurací>
```

3.2.2 Konfigurace dimenze a horní mez

```
./balls <dimenze> <horní mez> <data>
./balls n  q  data_n

./balls 4 19 triangle4
./balls 5 17 triangle5
./balls 6 15 triangle6
```

3.2.3 Datové soubory

-1 reprezentuje prázdné políčko.

Soubor: triangle4

```
1
3 2
8 4 5
6 7 -1 9
```

Soubor: triangle5

```
4
5 1
8 3 9
6 -1 2 7
10 11 12 13 14
```

Soubor: triangle6

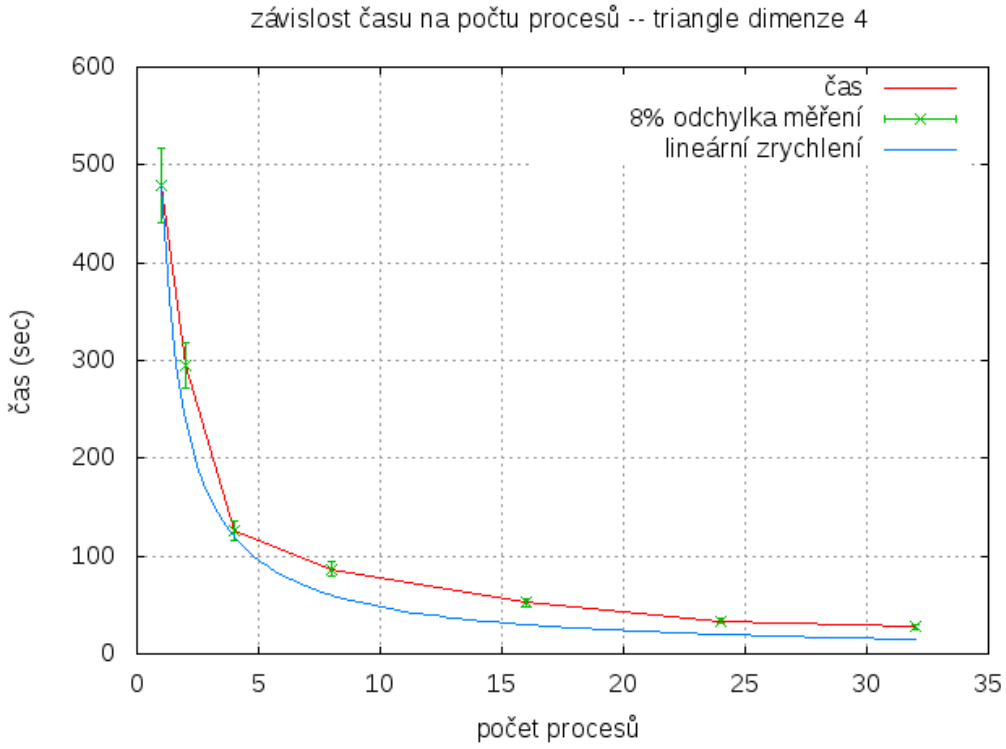
```
4
5 1
2 3 7
6 -1 8 9
10 11 12 13 14
15 16 17 18 19 20
```

3.3 Tabulka s výsledky

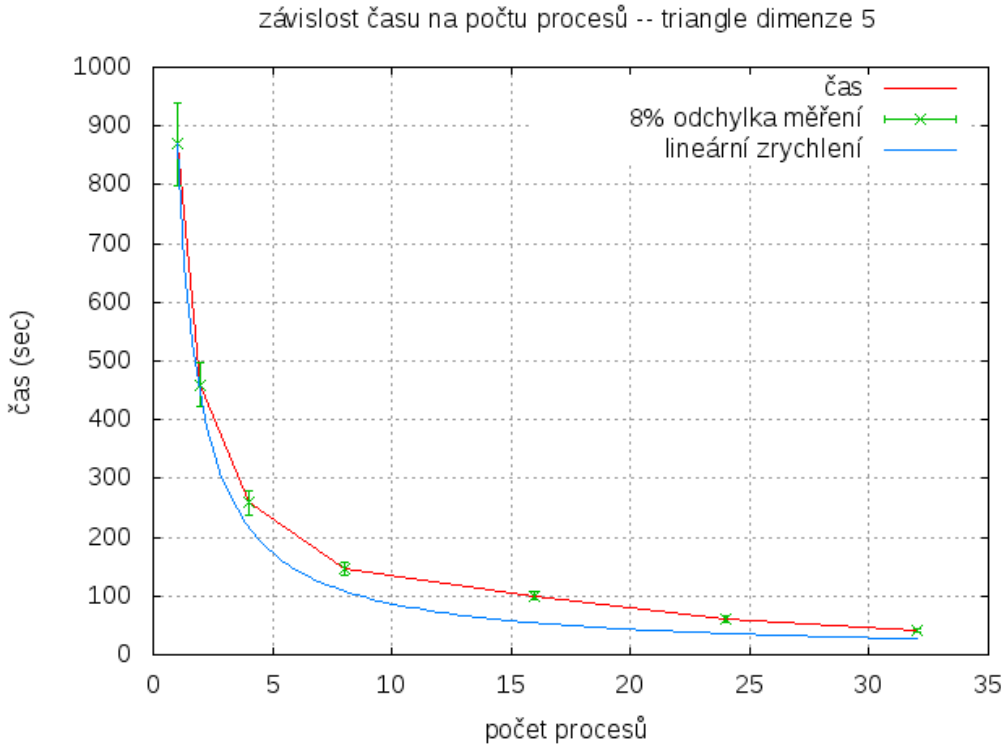
#CPU	triangle4	triangle5	triangle6
1	478.645257	868.954290	287.384048
2	294.697964	459.874666	177.874139
4	126.608475	259.122906	97.633753
8	86.888339	146.344138	44.997678
16	52.924784	100.802686	32.854664
24	33.590004	60.888389	18.830592
32	27.893136	42.098020	14.369734

3.4 Grafy

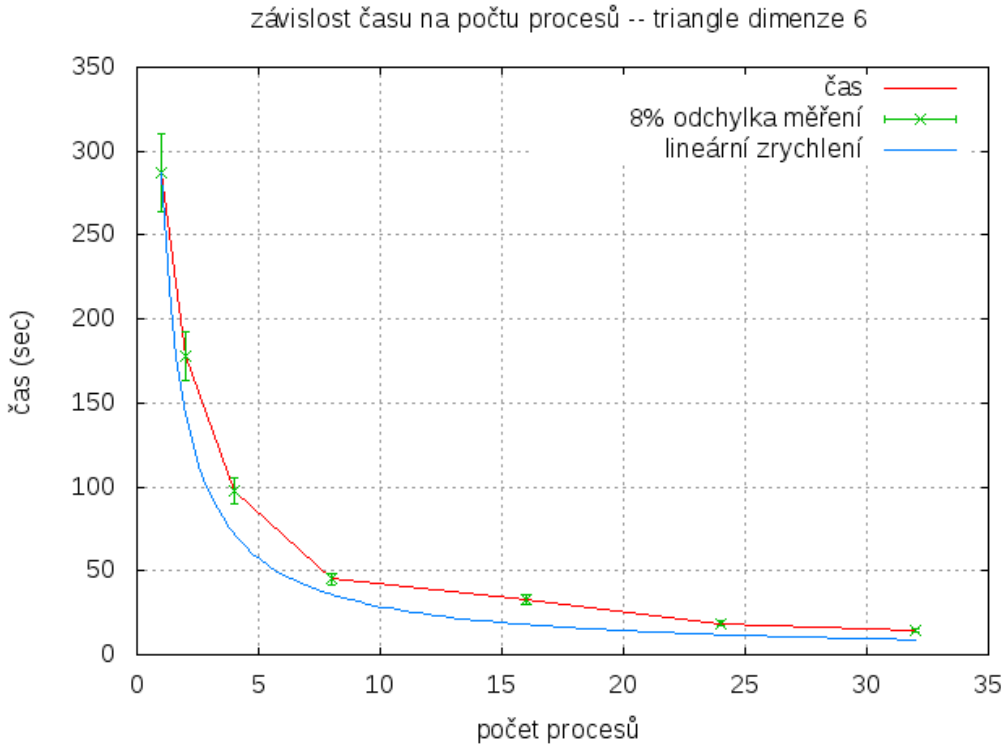
Na grafech lineárního zrychlení $S(n, p)$ je zobrazen naměřený čas, přibližná odchylka měření a průběh lineárního zrychlení, který vychází z naměřené hodnoty sekvenčního řešení.



Obrázek 2: Měření pro trojúhelník dimenze 4



Obrázek 3: Měření pro trojúhelník dimenze 5



Obrázek 4: Měření pro trojúhelník dimenze 6

3.5 Vyhodnocení

Zrychlení je lineární.

Díky prohledávání stavového prostoru do hloubky je algoritmus paměťově nenáročný a nevzniká tak superlineární zrychlení.

4 Závěr

Zkušenost s MPI je velkým přínosem, stejně tak i oživení programování v $C/C++$.

Překvapilo nás relativně snadné použití knihovny MPI.