PRL 2016/2017: Projekt 2
Enumeration Sort

Dávid Mikuš
xmikus15@stud.fit.vutbr.cz

7. apríla 2017

1 Enumeration Sort

Enumeration sort je paralelný radiaci algoritmus. Obsahuje lineárne pole procesorov so spoločnou zbernicou. Procesory su prepojené linearným spojením, ktoré umožnuje susediacim procesorom spolu komunikovať a priamo si vymienať medzi sebou hodnoty. Každý procesor obsahuje 4 rôzne registre: `X`, `Y`, `Z` a `C` ktorý slúži pre uchovanie relativného poradia v postupnosti.

1.1 Princíp

1. Všetky registre C sa nastavia na hodnotu 1
2. Nasledújce činosti sa opakŭju $2.n$ krát pre $1 \leq k \leq 2n$
   - Pokiaľ vstup není vyčerpaný, vstupný prvok x_i sa vloží do X_i (zbernici) a do Y_1 (lineárnym spojením) a obsah všetkých regsitrov Y sa posunie doľava.
   - Každý procesor s neprázdnými registrami `X` a `Y` ich porovná, ak je $X > Y$ tak inkrementuje `C`
   - Ak je $k > n$ (po vyčerpaní vstupov) procesor P_{Ck-n} pošle zbernici obsah svojho registru `X` procesoru P_{Ck-n}, ktorý ho uloži do svojho registru `Z`.
3. V následujúcich `n` cyklov procesory posúvaju obsah svojich registrov `Z` doprava a procesor P_n produkuje zoradenú postupnosť

1.2 Teoretická zložitosť

Algoritmus sa skladá z 3 krokov

1. $\theta(1)$ - inicializácia registrov `C`
2. $\theta(2.n)$ - distribŭcia hodnôt a porvnávanie

3. $\theta(n)$ - distribúcia výsledkov

Kde n značí počet hodnôt. Asymptotická zložitosť algoritmu je teda lineárna.

$$\theta(1) + \theta(2.n) + \theta(n) = \theta(n) \tag{1}$$

Pre algoritmus je ale potrebný n procesorov: $p(n) = n$. Celková zložitošt algoritmu je teda $\theta(n^2)$

2 Implementácia

Algoritmus bol implementovaný v jazyku `C++` spolu s knižnicou `OpenMPI`[1] ktorý bol využitý pre paralelizáciu výpočtu. Ako podklad pre implementáciu bol použitý formálny algoritmus z `https://www.fit.vutbr.cz/study/courses/PDA/private/www/h003.pdf` str. 24.

Algoritmus bol mierne zmenený čo sa týka zápisu kódu ale inak na jeho princípe to nič nemení.

Komunikáciu procesorov simuluje knižnica `OpenMPI` pomocou funkcii `MPI_Send` a `MPI_Recv` ktoré slúžia pre zasielanie správ medzi procesormi.

Bola zavedená konštanta `EMPTY` ktorá je mimo rozsahu 0-255 a značí že daný register nemá pridelenú žiadnu hodnotu.

2.1 Duplikatné hodnoty

Základna implementácia sa nevie vysporiadať s duplikatnými hodnotami. Tento problém[2] bol vyriešený tak že prvky ktoré su pred v porovnaní s `rank` aktuálneho procesoru su porovnávane ako $x \geq y$ inak ako ostrá nerovnosť $x > y$

```
if (x != EMPTY && y != EMPTY)
{
    if((i < id && x >= y) || x > y)
        ++c;
}
```

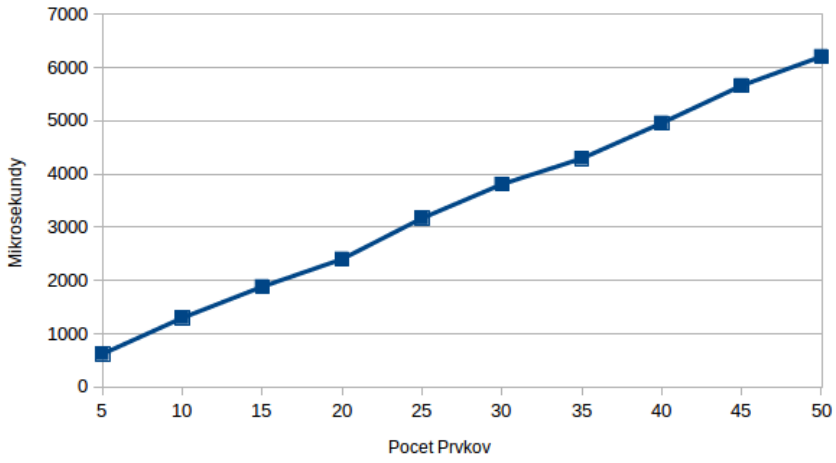
3 Experimenty

Výkonosť algoritmu sa merala v kode po inicializacii MPI kniznice, po zaslaní prveho prvku `X` a po prijatí posledného prvku.

Testy boli vykonavane pre rozny pocet prvkov. Pre kazdu sadu sa vykonalo niekolko merani. Vstup bola opacne zoradena postupnost.

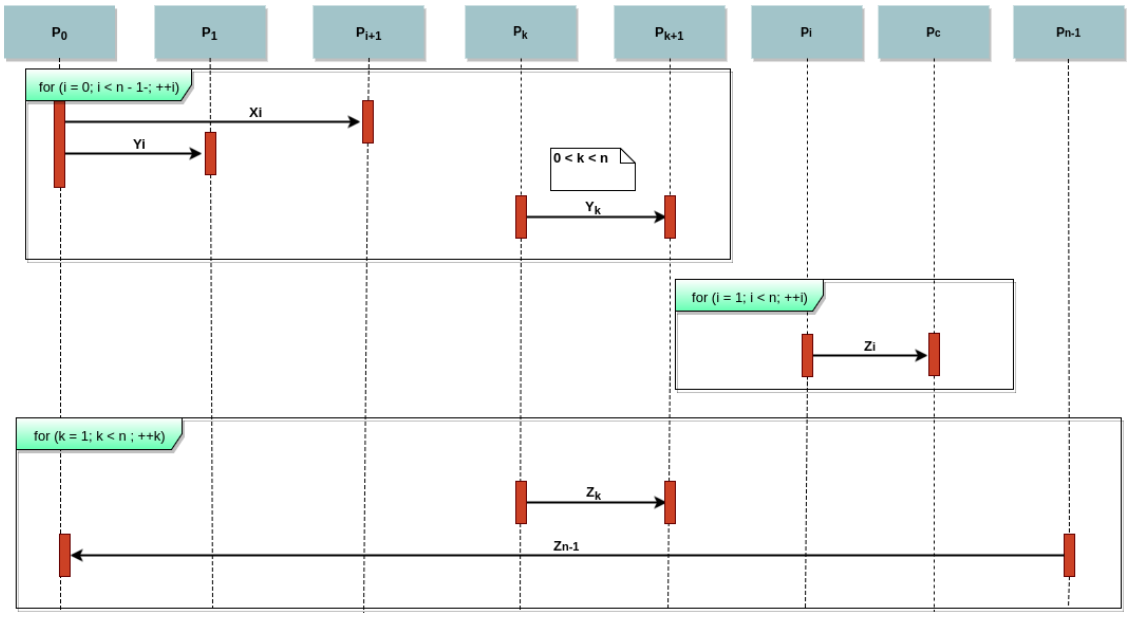
[1] `https://www.open-mpi.org/`

[2] `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.4976&rep=rep1&type=pdf`



Obr. 1: Graf meraní

4 Diagram



Obr. 2: Sekvenčný diagram zasielania správ cez MPI Send

5 Záver

Experimenty v kapitole 3 potvrdzujú lineárnu asymptotickú zložitosť, napriek tomu že implementácia bola mierne pozmeneá voči pseudokódu pre lahšiu prácu s knižnicou `OpenMPI` ale princíp algoritmu bol zachovaný.

Kód by sa dal zoptimalizovať ale pre zachovania logiky algoritmy nebola optimlizácia vykonaná.