

1 Úvod

Cieľom projektu je pomocou knižnice Open MPI implementovať v jazyku C/C++ algoritmus Merge-Splitting Sort, ako bol definovaný na prednáškach predmetu PRL.

1.1 Vstup

Vstupom je postupnosť čísel veľkosti 1B uložená v súbore.

1.2 Výstup

Výstup sa skladá z 2 častí:

1. jednotlivé načítané hodnoty v jednom riadku oddelené medzerou
2. jednotlivé zoradené hodnoty oddelené novým riadkom, zoradené od najmensej po najväčšiu

2 Rozbor a analýza algoritmu

Algoritmus sa používa na zoradenie poľa o veľkosti n pomocou $p(n)$ procesorov, pričom $p(n) < n$, teda počet procesorov je menší ako počet prvkov v zoradovanom poli. Je variantou algoritmu Odd Even Merge sort, ale každý procesor obsahuje pole prvkov. Každý procesor sa teda stará o viac prvkov. Zoradovanie prebieha v dvoch krokoch, v prvom kroku si každý procesor zoradí pole prvkov, ktoré obsahuje. Na zoradenie je použitý nejaký optimálny sekvenčný algoritmus. Tento krok prebieha paralelne na všetkých procesoroch naraz. V druhom kroku prebieha výmena prvkov medzi susednými procesormi. Najprv každý sudý procesor sa spojí so svojim pravým sudým susedom, spoja svoje dve postupnosti do zoradenej postupnosti. A túto postupnosť si rozdelia tak že ten prvý si vezme nižšiu polovicu a druhý vyššiu. V druhej fáze sa rovnaká akcia opakuje pre liché procesory. Tento proces sa opakuje presne $p/2$ krát. Podľa toho či sa iteruje cez sudé alebo liché procesory, tak rozlišujeme názov krok 1 alebo krok 2:

2.1 Analýza zložitosti

1. predspracovanie optimálnym sekvenčným algoritmom: $O((n/p) \log (n/p))$
2. prenos S_{i+1} do vedľajšieho procesora: $O(n/p)$
3. spojenie S_i a S_{i+1} do sekvencie: $2 \cdot n/p$
4. spätný prenos S_{i+1} do vedľajšieho procesora: $O(n/p)$
5. krok 1 alebo 2: $O(n/p)$

Výsledná časová zložitosť:

$$t(n) = O[(n/p) \log (n/p)] + O(n) = O((n \log n)/p) + O(n)$$

Cena:

$$c(n) = t(n) \cdot p = O(n \cdot \log n) + O(n \cdot p)$$

2.2 Závislosť na počte procesorov

Keďže je zložitosť závislá na počte procesorov, a ten si môžeme v Merge-Splitting Sorte zvoliť, môžeme túto zložitosť ďalej rozobrať. Ak by sme si zvolili extrém, že $p=1$, výsledná zložitosť bude $O(n \log(n))$, čo je vlastne optimálny sekvenčný algoritmus. Druhý extrém bude, že počet procesorov sa rovná počtu radených prvkov. Výsledná zložitosť bude $O(n)$. Z toho vyplýva že pomocou nastavenia počtu procesorov môžeme meniť zložitosť výpočtu. Algoritmus je optimálny pre $p \leq \log n$.

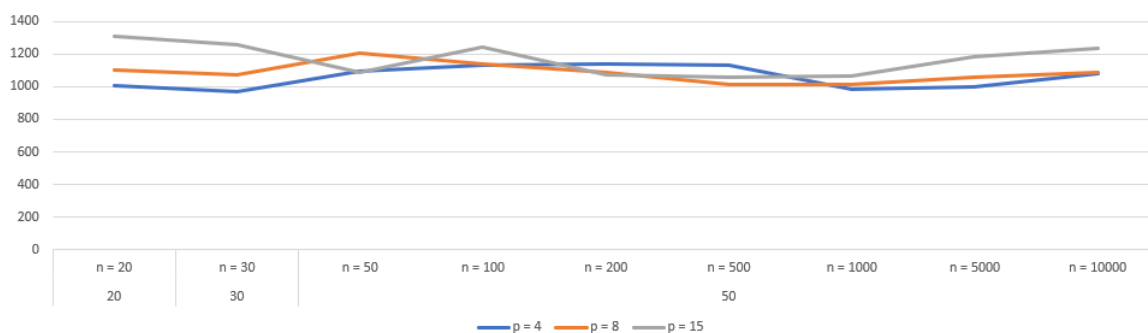
3 Implementácia

Algoritmus bol implementovaný v jazyku C++ za pomoci knižnice Open MPI. Implementácia sa skladá z 3 väčších celkov.

- V prvom kroku prebieha výpočet, koľko prvkov bude priradených do každého procesoru. Potom procesor s najnižším poradovým číslom postupne číta hodnoty zo vstupného súboru a rozosiela ich jednotlivým procesorom.
- V druhom kroku prebieha samotné zorad'ovanie postupnosti v jednotlivých procesoroch. Procesory si medzi sebou posielajú a vymieňajú hodnoty a postupne sa postupnosť zorad'uje do výslednej podoby.
- V treťom kroku si procesor s najnižším poradovým číslom vyžiada hodnoty zo všetkých procesorov a vypíše ich na výstup, už ako zoradenú postupnosť.

4 Experimenty a namerané hodnoty

Experimenty prebiehali na školskom serveri **merlin**. Testy prebiehali na počte procesorov 4, 8 a 15. Experimenty prebiehali opakovane, a do nasledujúceho grafu boli zanesené priemerné hodnoty. V grafe je čas výpočtu v milisekundách.



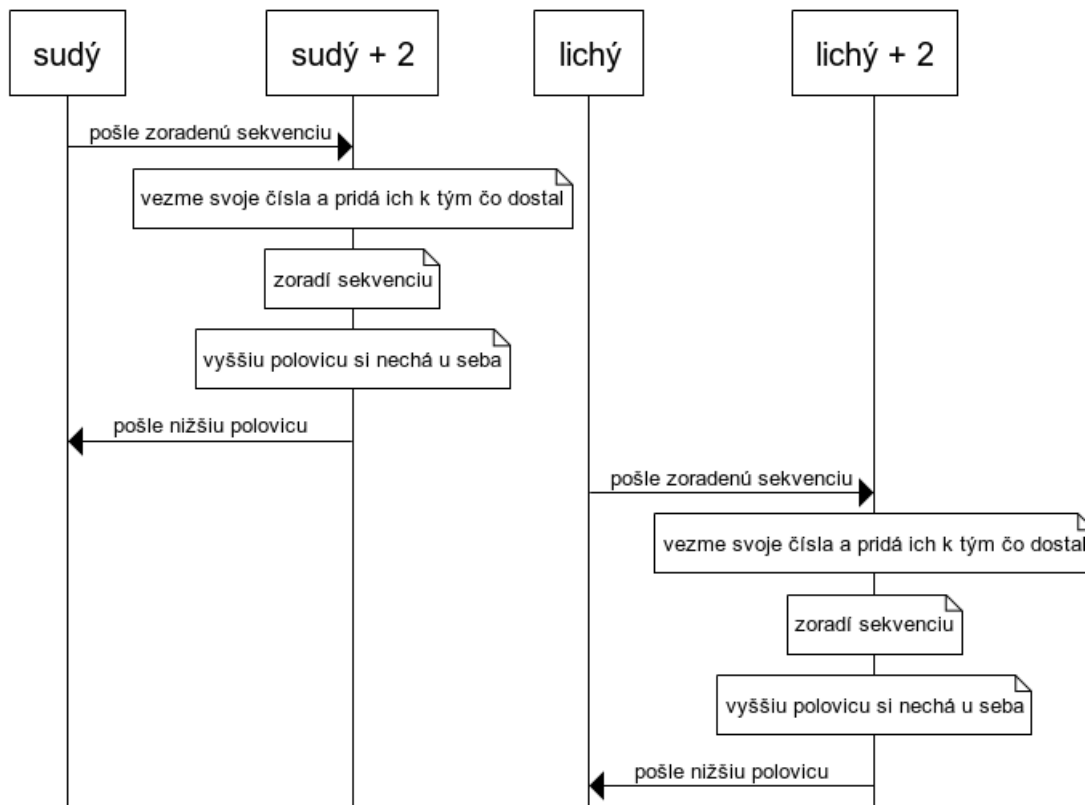
Obr. 1: Graf časovej zložitosti

Z grafu a nameraných hodnôt viacmenej vyplýva že pre daný počet procesorov sa čas výpočtu veľmi nelíši pre rôzne počty čísel. Väčšie rozdiely by sme videli pri vyššom počte procesorov a čísel, ale to server merlin nedovoľuje. Maximálny dostupný počet procesorov, ktorý bolo možné otestovať bol okolo 15. Z grafu nevyplývajú žiadne výraznejšie odchýlky. Menšie odchýlky sú spôsobené menším počtom meraní a nepresnosťou výpočtu, vyššiu presnosť by bolo možné dosiahnuť vyšším počtom meraní a presnejším meracím nástrojom.

5 Komunikačný protokol

Nasledujúci komunikačný protokol zhruba popisuje, ako si procesory posielajú medzi sebou správy. Diagram popisuje komunikáciu medzi sudými a lichými procesormi v jednom kroku iterácie.

Merge Splitting Sort



Obr. 2: Komunikačný protokol

6 Záver

V projekte bol úspešne implementovaný algoritmus Merge-Splitting Sort pomocou knižnice Open MPI. Následne bola overená časová zložitosť algoritmu. Z experimentov vyplynulo, že časová zložitosť implementovaného algoritmu približne odpovedá teoreticky spočítanej zložitosti.