Paralelné a distribuované algoritmy
Dokumentácia k projektu č. 2

Michal Gabonay (xgabon@stud.fit.vutbr.cz) 5.4.2018
Vysoké Učení Technické v Brne

1. Úvod

Tento dokument slúži ako dokumentácia k projektu do predmet Paralelné a distribuované algoritmy. Úlohou tohto projektu bolo implementovať radiaci algoritmus Merge-splitting sort v jazyku C++ za pomoci knihovne Open MPI. V dokumente je rozobraná teoretická stránka algoritmu ako napríklad odvodenie zložitosti, ďalej je tu popis implementácie a taktiež popis experimentov.

2. Algoritmus Merge-splitting sort

Merge-splitting sort funguje nad lineárnym polom procesorov p(n)<n, čiže počet procesorov je menší ako počet radených prvkov. Každý procesor sa stará o viac prvkov, približne n/p.

2.1 Princíp

1. Každý procesor dostane n/p prvkov.
2. Tieto prvky si každý procesor zotriedi optimálnym sekvenčným algoritmom.
3. Striedajú sa procesory na párnej a nepárnej pozícii. Po p/2 krokoch by mala byť postupnosť zotriedená.
    3.1 Všetky procesory na párnej pozícii pošlú svojmu pravému susedovi svoju postupnosť čísel. Nepárne procesory príjmu hodnoty od svojho ľavého suseda, spoja tieto hodnoty a svoje hodnoty do zotriedeného poľa. Prvú polovicu tohto poľa pošlú naspäť svojmu ľavému susedovi a druhú polovicu si nechajú.
    3.2 Rovnaký postup ale s procesormi na nepárnej pozícii.

2.2 Analýza

Teoretická časová zložitosť a celková cena

a) Rozposlanie hodnôt všetkým procesorom – lineárna architektúra - *O(n)*

b) Každý procesor zoradí svoju časť sekvenčne - *O((n/p)*log (n/p))*

c) Poslanie hodnôt susedovi - *O(n/p)*

d) Spojenie hodnôt procesora s hodnotami ktoré dostal od suseda optimálnym alg. - *2*n/p*

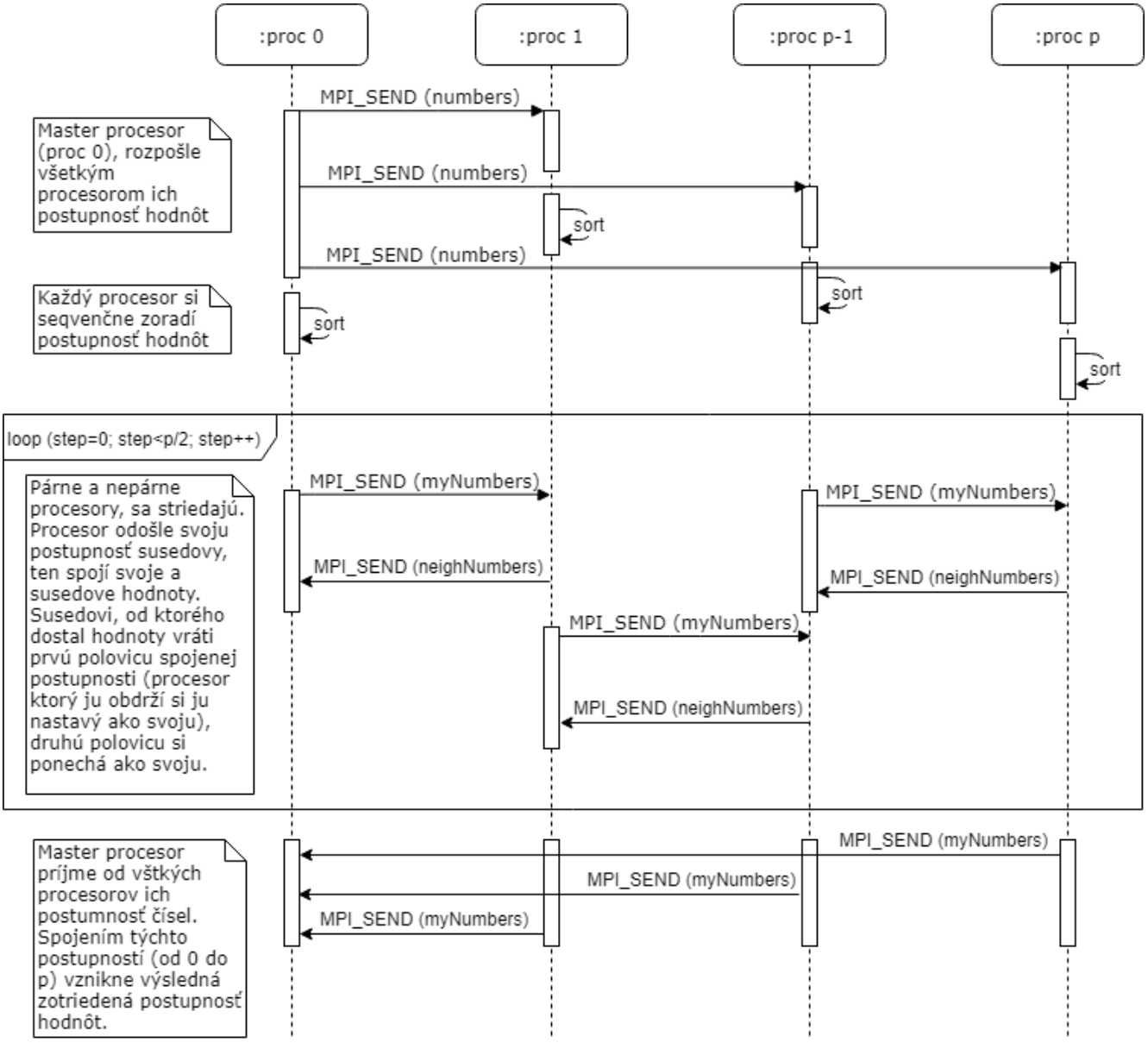
e) Odoslanie hodnôt naspäť susedovi - *O(n/p)*

Výsledná časová zložitosť: *O[(n/p)* log (n/p)]+O(n) = O((n* log n)/p) + O(n)*
Celková cena algoritmu je *c(n) = t(n)*p(n) = (O((n* log n)/p) + O(n))*p = O(n.log n) + O(n.p).* Optimálny sekvenčný algoritmus má zložitosť *O(n*log n).* Z toho vyplýva, že algoritmus Merge-spliting sort je optimálny pri počte procesorov p ≤ log n .

Teoretická priestorová zložitosť
Každý procesor potrebuje *p/n* pamäte pre svoju postupnosť hodnôt, ďalej *p/n* pamäte pre hodnoty svojho suseda a ešte *2*p/n* pamäte pre spojenie svojich a susedových hodnôt, čiže dokopy *4*p/n* pamäte. To znamená, že celková asymptotická zložitosť algoritmu je *O(n)*.

3. Implementácia

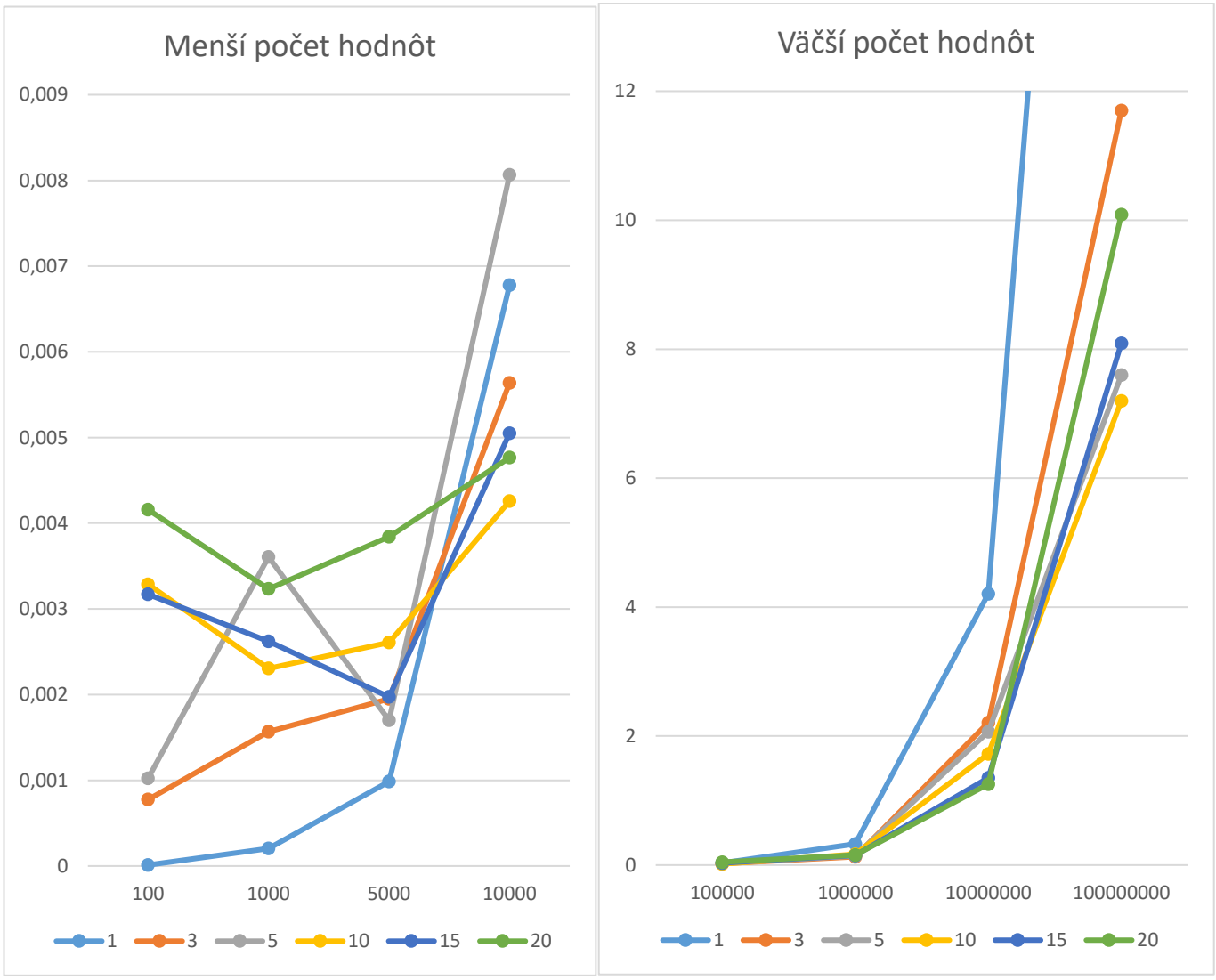
Aplikácie ja implementovaná v jazyku C++ s využitím knižnice Open MPI, ktorá umožňuje paralelné riešenie problémov. Ako podklad pre aplikáciu som využil ukážkovú súbor odd-even.cpp stiahnutý zo študijných materiálov do predmetu PRL. Využívam najmä funkcie *MPI_Send* a *MPI_Recv*, ktoré sú blokujúce a umožňujú komunikácie medzi jednotlivými procesormi. Ďalej využívam MPI_Wtime pre meranie času. Problém, ktorý nastáva, keď počet prvkov nie je deliteľným počtom procesorov riešim tak, že toľko procesorov aký je zvyšok po delení (prvky modulo procesory), si vezmú o jeden prvok naviac. Master procesor rozpošle všetkým procesorom hodnoty, sebe si len skopíruje (pri väčšom počte prvkov z nejakého dôvodu blokoval poslanie prvkov samému sebe). Každý procesor si zoradí svoju postupnosť hodnou funkciou *std::sort()*. Ďalej je cyklus, kde sa striedajú párne a nepárne procesory. Procesor odošle svoje hodnoty susedovi, ten volá funkciu *mergeAndSplit()*, ktorá spojí susedove a svoje hodnoty do jednej zoradenej postupnosti tak, že vždy porovnáva vrcholy polí a väčší dá do výsledného poľa. To následne rozdelí na 2 polia o veľkosti akej do funkcie vstúpili. Prvú časť procesor naspäť vráti susedovi. Tento cyklus sa opakuje *p/2* krát, potom už sú všetky hodnoty zoradené. Ostáva už len to, že každý procesor vráti svoju postupnosť master procesoru, ktorý ich všetky vloží do jedného poľa (postupne od seba až po posledný procesor) a vypíše výsledok. Tento postup je zjednodušene znázornený v sekvenčnom diagrame na obrázku 1.



Obrázok 1 - Sekvenčný diagram

4. Experimenty

S algoritmom som vykonal experimenty, ktoré sú zhrnuté do grafov na Obrázku 2. Vertikálna osa je čas v sekundách, horizontálna je počet radených prvkov a jednotlivé krivky reprezentujú počet procesorov. Pre každú kombináciu počtu prvkov a počtu procesorov som aplikáciu spustil 3-5 krát a z výsledných časov spočítal priemer (malo by byť viac, ale pri veľkých hodnotách spustenie trvá veľmi dlho z dôvodov nezávislých na algoritme, ako napríklad generovanie čísel a výpis na štandardný výstup). Testovanie prebehlo na školskom servery merlin, ktorý má 12 procesorov, čo sa odzrkadľuje na výsledkoch pre 15 a 20 procesorov, kde narástla réžia prepínania kontextu. Taktiež treba brať v úvahu, že na testovanom servery nie som jediný, čiže sa o výpočtovú silu delím s ostatnými študentami a zamestnancami.
Z výsledkov je vidno, že pri malých hodnotách je sekvenčný výpočet (1 procesor) rýchlejší, ktorý však pri veľkom počte hodnôt ďaleko prevyšuje paralelné spracovanie. Čas som nemeral od spustenia aplikácie po koniec aplikácie, ale od začiatku algoritmu, ako je v prednáške, čiže až keď už každý procesor mal svoju postupnosť čísel a meranie končil pred výpisom hodnôt.



Obrázok 2 – Grafy z experimentu

5. Záver

Algoritmus sa mi podarilo úspešne implementovať. Výsledky experimentovania boli zdokumentované a zhrnuté do tohto dokumentu. Som si vedomí, že by sa dal implementovať aj lepšie, optimálnejšie, ale na študijné účely, ako projekt je plne postačujúci.