PRL – Odd-even transposition sort

Alena Tesařová (xtesar36), březen 2020

1 Úvod

Zadáním projektu bylo naimplementovat a zdokumentovat řadící algoritmus *Odd-even transposition*, který je založen na sekvenčním bubble sortu a byl podrobně rozebrán na přednášce z předmětu PRL[1]. V sekci 2 bude podrobněji zanalyzována časová složitost algoritmu. Popis implementace se nachází v sekci 3. Pro lepší názornost obsahuje zpráva i komunikační prokol (sekce 5) a průběh časové složitosti pro různě velké vstupy (sekce 4).

2 Rozbor a analýza algoritmu

Algoritmus je založen na porovnání sousedních hodnot a hlavní část je popsána pseudokódem na Algoritmu 1. Každý z kroků (1) a (2) provádí jedno porovnání a dva přenosy, což vyžaduje konstantní čas. Tyto dva kroky jsou spuštěny $n/2$ krát. Proto časová náročnost algoritmu je lineární tedy $t(n) = O(n)$.

Pro provedení algoritmu je vyžadováno n procesorů pro každé číslo, tedy prostorová složitost je $p(n) = n$, což pak ovlivňuje celkovou cenu. Cena se vypočítá jako $c(n) = p(n) * t(n) = O(n^2)$, což není optimální (optimální by bylo $n * \log(n)$). Z tohoto pohledu se nezdá být algoritmus atraktivní, nedosahuje tak velkého zrychlení (sekvenční *Quicksort* dosahuje až O($log\ n$)) a navíc používá počet procesorů, který se rovná délce vstupu a to není optimální.

```
for j = 1to[n/2] do
    (1) for i = 1, 3, ...., 2[n/2] − 1 do in parallel do
        if x_i > x_{i+1} then
            x_i ⇔ x_{i+1}
        end
    end
    (2) for i = 2, 4, ...., 2[(n − 1)/2] − 1 do in parallel do
        if x_i > x_{i+1} then
            x_i ⇔ x_{i+1}
        end
    end
end
```

Algorithm 1: Hlavní část algoritmu Odd-even transposition

3 Implementace

Implementace algoritmu byla částečně převzata ze vzorového příkladu *odd-even.cpp*[2] na privátní straně předmětu PRL.

V první části provádí Master proces načtení čísel ze souboru *numbers* a rozesílá čísla ostatním procesům. Jakmile proces přijme číslo, může pokračovat k vykovávání dalších kroků. Pro zaslání zprávy byla použita funkce MPI_Send[3] a pro příjem funkce MPI_Recv[4]. Obě z knihovny *mpi.h*.

V druhé části dochází přehazování čísel – nejprve na sudých pozicích a následně na lichých. Toto prohození se opakuje maximálně $n/2$, kde n je počet procesů. Prohazování čísel je realizováno opět zasíláním zpráv, jak lze vidět v komunikačním diagramu na obrázku 2. Nejprve všechny sudé procesy pošlou svému lichému sousedovi jejich číslo a ten pak provede porovnání a vrátí menší z hodnot (jelikož

[1] https://www.fit.vutbr.cz/study/courses/PDA/private/www/h003.pdf
[2] http://www.fit.vutbr.cz/ ikalmar/PRL/odd-even-trans/odd-even.cpp
[3] https://www.mpich.org/static/docs/v3.1.x/www3/MPI_Send.html
[4] https://www.mpich.org/static/docs/v3.1.x/www3/MPI_Recv.html

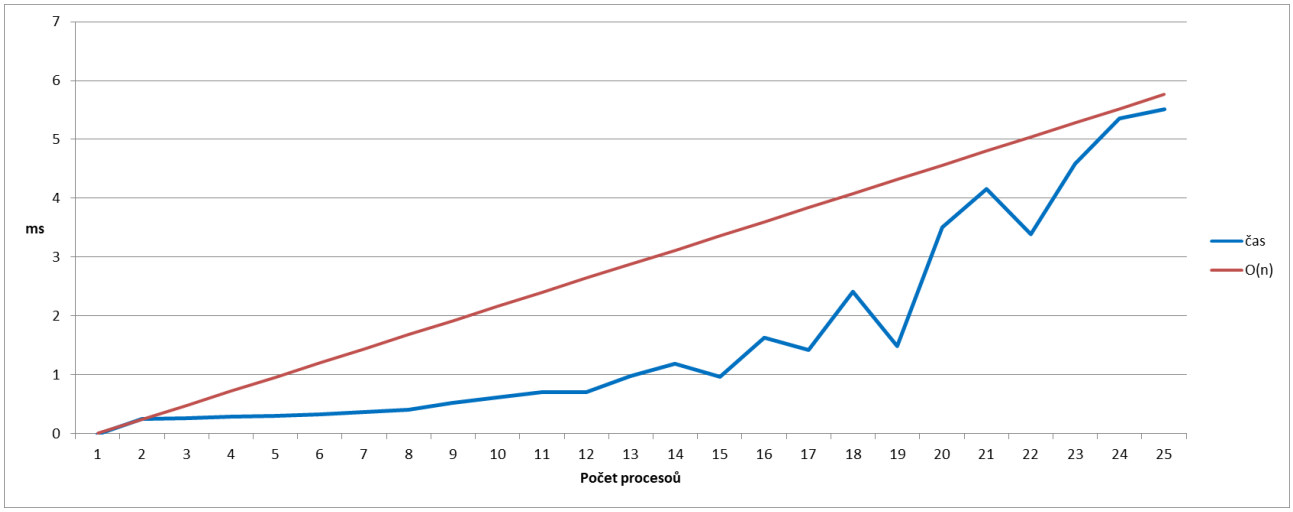
je sudá na nižší pozici). Stejný proces se děje v druhé podmínce s rozdílem, že liché procesy posílají své číslo sudým procesům, které dělají porovnání. Tímto jednoduchým způsobem docílíme vzestupné seřazení čísel, tak, že procesor s nejmenším *pid* bude obsahovat nejmenší číslo.

Poslední část implementace zajišťuje správné vypsání čísel na výstup. Realizace je opět provedena na základě posílání zpráv. Procesy od 1 až n posílají svá čísla hlavnímu Master procesu ($pid = 0$) a ten si je pak uloží do pole na správný index (podle čísla procesu, od kterého přijímá). V posledním kroku pak Master všechna čísla správně vypíše na výstup.

4 Experimenty

Experimenty byly naměřeny pro 1 až 25 procesů. Výpočet pro jednotlivé procesy se prováděl 100× a z každého měření se vzal nejpomalejší proces. Následně se ze 100 nejpomalejších procesů vyhodnotil medián a výsledek lze vidět v grafu 1. Na grafu lze vidět, že křivka roste lineárně a tedy jsme ověřili, že časová složitost je skutečně lineární $O(n)$.

Pro měření času byla použita funkce *MPI_ Wtime*[5], která poskytuje vysoké rozlišení při měření času. Funkce byla volána před a po fází řazení a vypočítal se rozdíl těchto dvou hodnot.



Obrázek 1: Graf zachycuje trvání algoritmu pro různě velké vstupy

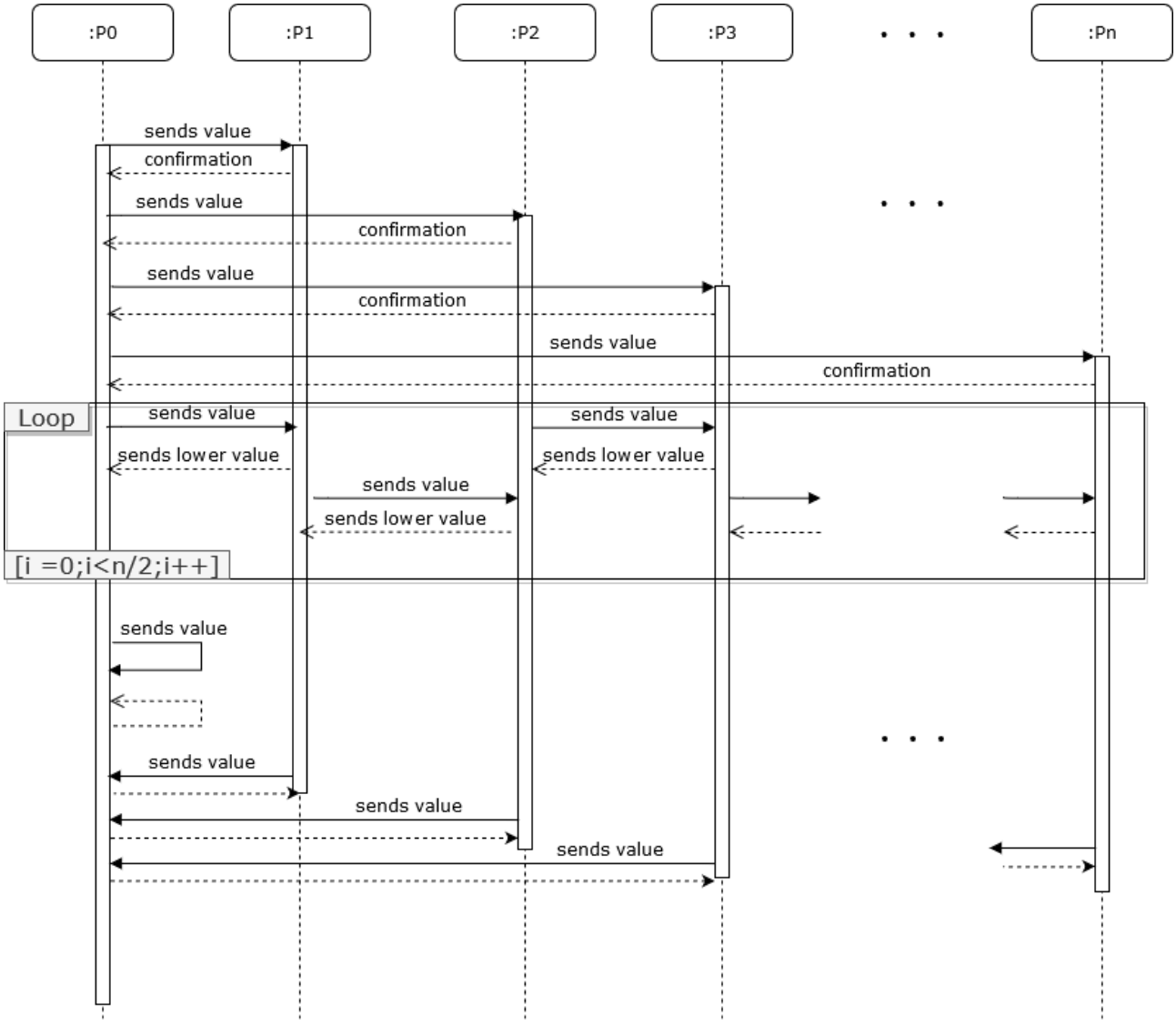
5 Komunikační protokol

Na obrázku 2 je znázorněný komunikační protokol mezi procesory. Jedná se o zjednodušený model pro n procesů.

6 Závěr

V této zprávě se podařilo podrobně popsat algoritmus odd-even transposition sort. Program funkce podle očekávání a jeho časová složitost byla ověřena na základě sady experimentů.

[5] https://www.mpich.org/static/docs/v3.2/www3/MPI_Wtime.html



Obrázek 2: Sekvenční graf zobrazuje komunikaci mezi procesy během vykonávání programu.