Paralelní a distribuované algoritmy
2. projekt

Odd-even transposition sort

Peter Horňák
xhorna14@stud.fit.vutbr.cz

1. Analýza algoritmu

Algoritmus prebieha v dvoch krokoch. V prvom kroku sa každý nepárny proces porovná so svojim susedom a na základe toho si vymenia svoje hodnoty. V ďalšom kroku spraví rovnaký úkon každý párny proces. Po maximálne n krokoch sú všetky hodnoty zoradené.

- Časová zložitosť algoritmu môžeme vyjadriť ako $t(n) = \mathcal{O}(n)$, keďže sa pri každom kroku vykonáva porovnanie a dve výmeny z čoho vyplýva, že čas je konštantný.

- Pamäťová zložitosť algoritmu je $p(n) = \mathcal{O}(n)$, keďže pre každé číslo na vstupe algoritmu je priradený jeden proces.

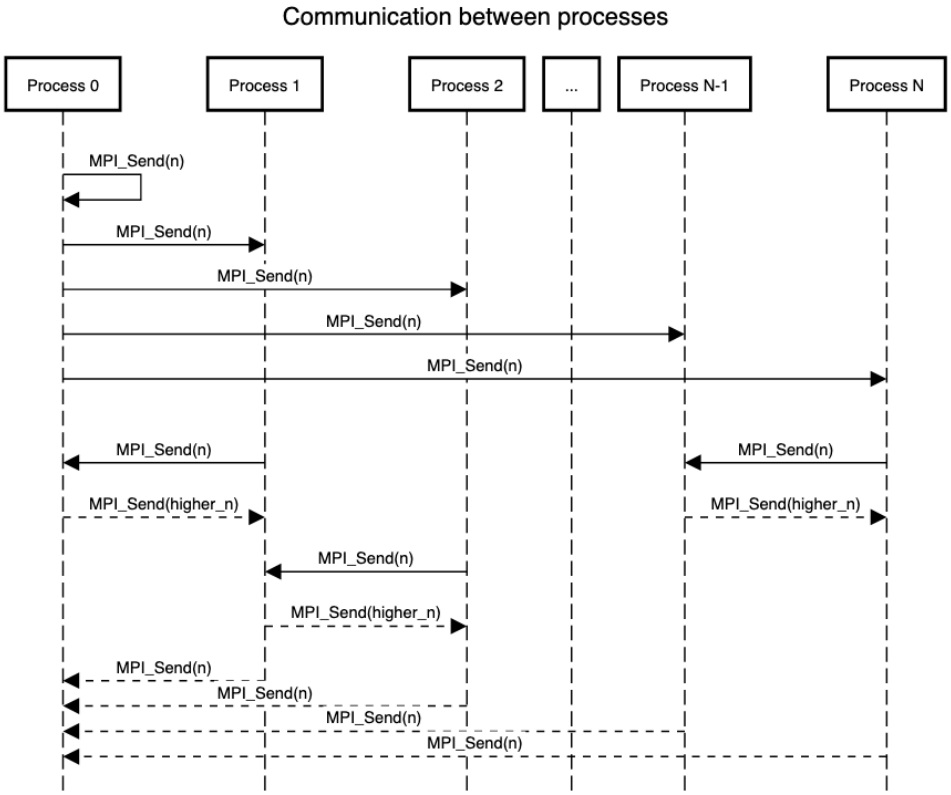
- Z toho je možné odvodiť celkovú cenu algoritmu, ktorá je nasledovná:
  $c(n) = t(n) * p(n) = \mathcal{O}(n * n) = \mathcal{O}(n^2)$ čo nie je optimálna cena.

2. Implementácia

Projekt je implementovaný v jazyku C++. Pre jeho správne spustenie je k zdrojovému kódu priložený skript *test.sh*, ktorý prijíma jeden parameter určujúci počet čísiel. Tento skript vygeneruje náhodné čísla a vloží ich do súboru *numbers*.
Následne sa spustí program s rovnakým s takým počtom procesov, že každému číslu bude priradený jeden vlastný proces. Pridelenie čísiel procesom vykonáva proces s id 0 a to tak, že postupne číta čísla zo súboru *numbers* a prideľuje ich ostatným procesom pomocou funkcie *MPI_Send()*.
Po prijatí čísla pomocou funkcie *MPI_Recv()*, začína fáza radenia. V cykle sa volá funkcia *do_phase*(), do ktorej nasledujúce vstupujú parametre:

- *phase* – Číslo, ktoré určuje či párne alebo nepárne procesy budú meniť svoje hodnoty.
- *proc_num* – Celkový počet procesov
- *my_id* – Id procesu
- *current_number* – Číslo, ktoré daný proces obsluhuje.

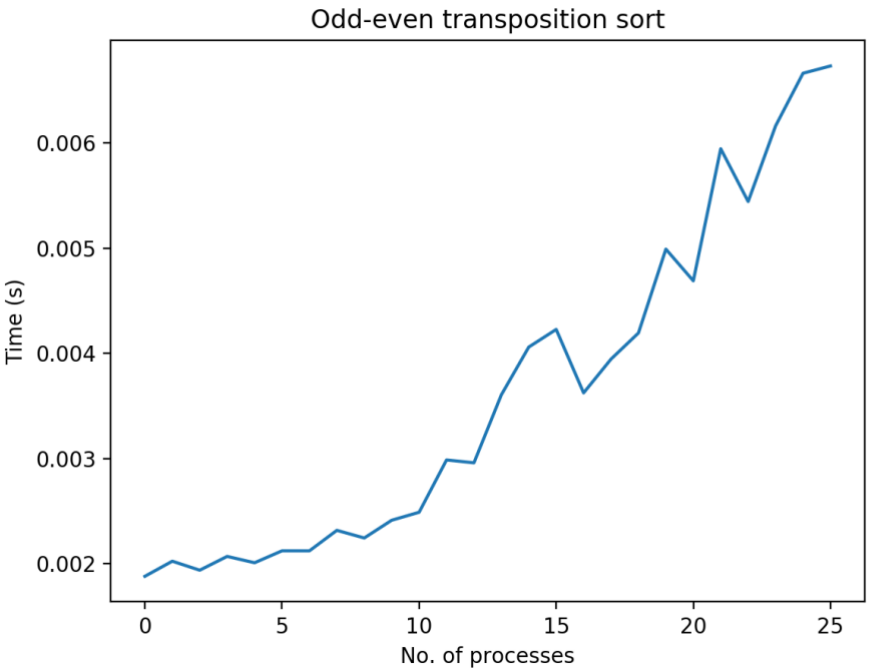
Proces, ktorého id po operácií modulo 2 sa nerovná parametru *phase*, s výnimkou procesu s id 0, pošle svoje číslo procesu s id o jedno menším a následne čaká na jeho odpoveď. Ostatné procesy, ktorých id je menšie ako parameter *proc_num – 1*, čakajú na prijatie čísla od procesu s id o jedno väčším. Následne porovná svoje číslo a číslo, ktoré prijal, menšie z týchto čísiel si ponechá a to druhé pošle naspäť.
Nakoniec všetky procesy pošlú, ich čísla procesu s id 0 a ten ich postupne prijíma a ukladá si ich do vektoru, ktorý potom vypíše na *stdout*.



Obr. 1: Sekvenčný diagram komunikácie

3. Experimenty

Experimenty boli vykonávané na školskom servery Merlin. Kvôli limitom serveru bolo možné testovať algoritmus na obmedzenom množstve vstupov a to 26. Na meranie bola použitá štandardná knižnica C++ *std::chrono*, konkrétne funkcia *std::chrono::high_resolution_clock::now()*.



Obr. 2: Výsledky experimentov.

4. Záver

Keďže sme sa pohybovali v rozmedzí malých množstiev vstupných čísiel a to z dôvodu limitu serveru, je možné vidieť malé odchýlky vo výsledkoch aj po viacerých meraniach.