Paralelní a distribuované algoritmy
Implementace algoritmu Merge-splitting sort

Nikola Valešová
xvales02@stud.fit.vutbr.cz

31. března 2018

1 Úvod

Cílem projektu je implementace algoritmu *Merge-splitting sort* v jazyce C/C++ s využitím knihovny Open MPI a následná analýza časové a prostorové složitosti daného algoritmu.

2 Vstup programu

Vstupem programu je posloupnost náhodných čísel uložená v binárním souboru „numbers“. Ten obsahuje předem daný počet bajtů, z nichž každý reprezentuje náhodnou hodnotu v rozsahu 0 – 255. Program nejprve tuto vstupní posloupnost načte, a poté se ji snaží seřadit s využitím zadaného algoritmu.

3 Výstup programu

Program neseřazenou vstupní posloupnost ihned po načtení vypíše na výstup, přičemž jednotlivé hodnoty jsou odděleny mezerou. Po skončení řadicího algoritmu je vzestupně vypsána seřazená posloupnost a každé číslo se nachází na novém řádku.

4 Popis algoritmu

Merge-splitting sort je řadicí algoritmus, který pro seřazení vstupní posloupnosti využívá lineární pole p procesorů. Nejprve je každému procesoru zaslána jemu přiřazená část vstupní posloupnosti, tu každý procesor načte a seřadí optimálním sekvenčním algoritmem. Poté probíhá samotné řazení, které sestává z $\lceil \frac{p}{2} \rceil$ iterací:

1. každý procesor se sudým indexem příjme posloupnost svého pravého souseda, setřídí jeho posloupnost se svou a pravému sousedovi vrátí horní polovinu seřazených čísel, levou polovinu si uloží jako svou novou posloupnost
2. každý lichý procesor příjme posloupnost svého pravého souseda, setřídí obě posloupnosti do jedné a odešle vyšší polovinu zpět sousedovi.

V poslední části algoritmu probíhá zpětné zaslání hodnot z procesorů a vytvoření cílové seřazené sekvence.

5 Analýza složitosti algoritmu

Počet procesorů lze vyjádřit vztahem:

$$p(n) = p, \text{ kde } p < n.$$

Analýza časové složitosti:

1. distribuce řazených prvků, procesor 0 odesílá $\frac{n}{p}$ hodnot postupně každému z p procesorů – složitost $O\big(p \cdot \frac{n}{p}\big) = O(n)$,

2. sekvenční seřazení prvků – optimální sekvenční algoritmus má složitost $O(n \cdot log\, n)$, každý procesor řadí $\frac{n}{p}$ prvků – celková složitost této části je tedy $O\left(\frac{n}{p} \cdot log\, \frac{n}{p}\right)$,
3. $2 \cdot \lceil \frac{p}{2} \rceil$ provedení následujících kroků:
   - odeslání, resp. příjem $\frac{n}{p}$ prvků – složitost $O\left(\frac{n}{p}\right)$,
   - setřídění dvou seřazených posloupností do jedné – složitost $O\left(\frac{n}{p}\right)$,
   - příjem, resp. odeslání $\frac{n}{p}$ prvků – složitost $O\left(\frac{n}{p}\right)$.

   Celková složitost kroku 3 je tedy $2 \cdot \lceil \frac{p}{2} \rceil \cdot O\left(\frac{n}{p}\right) \cdot 3 = O(n)$.
4. zpětné zaslání seřazených prvků, procesor 0 načítá $\frac{n}{p}$ hodnot postupně od p procesorů – složitost $O\left(p \cdot \frac{n}{p}\right) = O(n)$.

Celková teoretická časová složitost lze popsat vztahem:

$$t(n) = O\left(\frac{n}{p} \cdot log\, \frac{n}{p}\right) + O(n) = O\left(\frac{n \cdot log\, n}{p}\right) + O(n).$$

Z počtu procesorů a časové složitosti jsme schopni odvodit celkovou cenu algoritmu:

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

Algoritmus je tedy optimální, jestliže platí $p \leq log\, n$.

6 Implementace

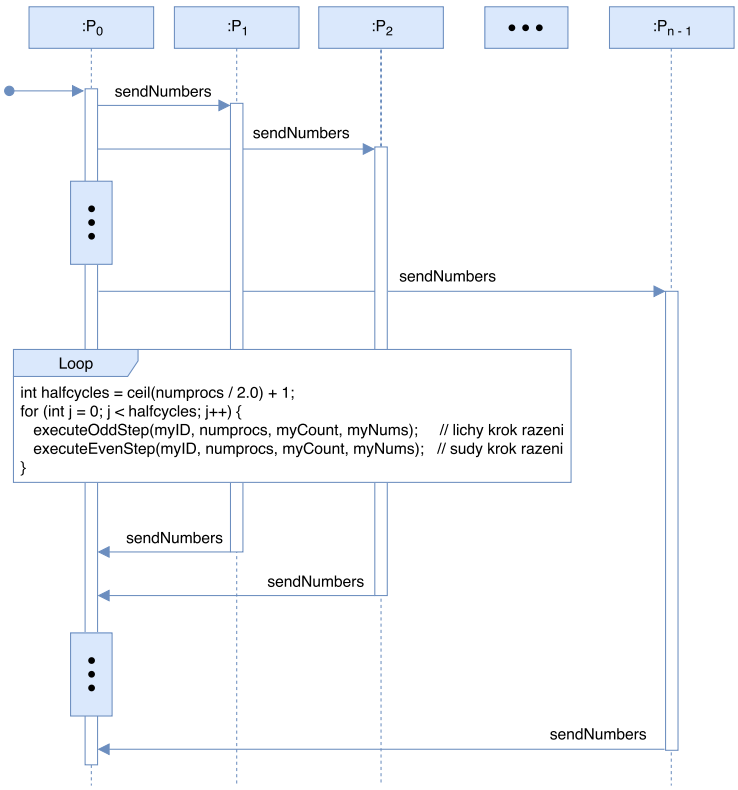
Pro implementaci byl zvolen jazyk C/C++ a knihovna Open MPI.

Implementace algoritmu je shodná s jeho popisem představeným na přednáškách předmětu PRL. Rozdíl je v číslování procesorů – namísto indexování od čísla 1 je v projektu první procesor označen indexem 0. Tomu odpovídá i implementované řazení, nejprve se spojí všechny sudé procesory se svými pravými sousedy, ve druhém kroku naopak řadí všechny liché procesory. Druhou změnou oproti původnímu algoritmu je počet iterací při řazení sekvence, původní počet $\lceil \frac{p}{2} \rceil$ je inkrementován o 1, jelikož při rozdílném počtu hodnot na procesor nemusí vždy původní počet iterací stačit na seřazení posloupnosti.

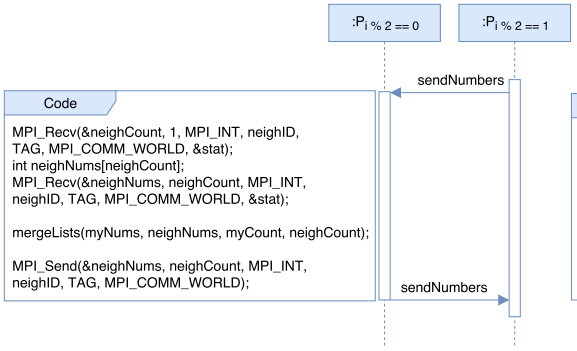
Načtení vstupního souboru, distribuci hodnot jednotlivým procesorům a finální příjem seřazených sekvencí provádí procesor s indexem 0.

Pokud je délka neseřazené vstupní posloupnosti n dělitelná počtem procesorů p, je každému procesoru přiděleno $\frac{n}{p}$ čísel. V opačném případě získá $n \bmod p$ prvních procesorů $\lceil \frac{n}{p} \rceil$ hodnot ze vstupní posloupnosti a zbylým procesorům je přirazeno hodnot o jedna méně, tedy $\lfloor \frac{n}{p} \rfloor$.

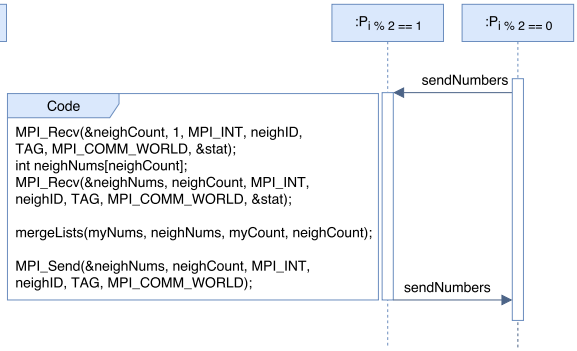
Komunikační protokol popisující způsob a následnost zasílání zpráv mezi jednotlivými procesory je znázorněn v obrázku 1.



(a) Celkové schema komunikace



(b) Lichý krok řadicího algoritmu



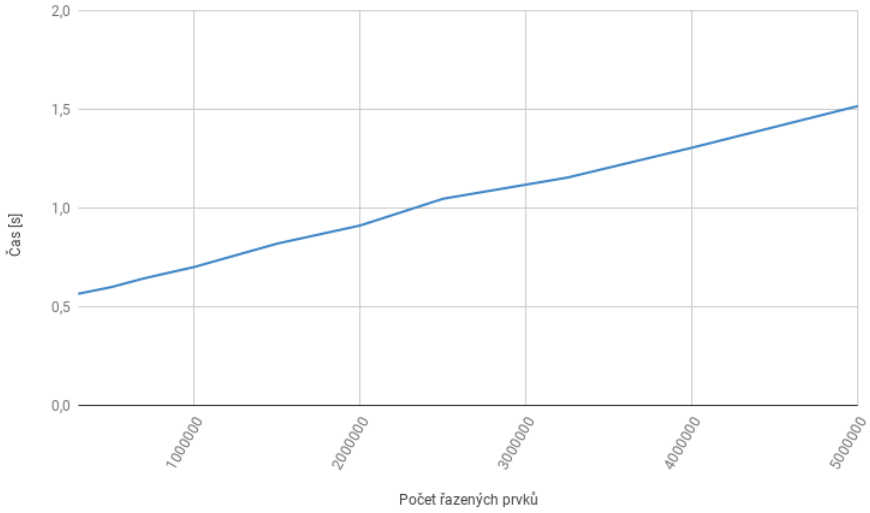
(c) Sudý krok řadicího algoritmu

Obrázek 1: Komunikační protokol zasílání zpráv

7 Experimenty

Pro praktické ověření časové složitosti algoritmu byla vytvořena sada deseti testovacích vstupů. Počet řazených prvků se v nich pohyboval v rozmezí od 300 tisíc do 5 milionů a pokaždé byly použity čtyři procesory. Pro každou délku vstupní posloupnosti bylo měření spuštěno celkem desetkrát a do grafu byl poté zanesen jejich průměr, aby se do statistiky výrazněji nepromítaly extrémní hodnoty. Pro získání doby běhu byl využit reálný čas běhu programu získaný pomocí příkazu `time`.

Na obrázku 2 je zobrazena křivka představující experimentálně získanou závislost doby běhu programu na počtu řazených prvků. Teoretická časová složitost algoritmu je $t(n) = O(\frac{n \cdot log\, n}{p}) + O(n)$. Získaná křivka má téměř lineární průběh, což odpovídá očekávané časové složitosti. Pokud bychom provedli více experimentů z ještě většího rozsahu počtu řazených hodnot, byl by patrný linearitmický průběh křivky. Celá křivka nemůže mít čistě lineární průběh, jelikož by bylo porušeno pravidlo, že cena paralelního algoritmu je vždy stejná nebo vyšší než cena optimálního sekvenčního algoritmu.



Obrázek 2: Závislost doby běhu programu na počtu řazených prvků

8 Závěr

V rámci projektu byl naimplementován paralelní řadicí algoritmus *Merge-splitting sort* a pomocí experimentů byla ověřena jeho časová složitost. Experimentálně získaná časová složitost se jevila lineární, avšak v takovém případě by bylo porušeno pravidlo, že cena paralelního algoritmu nikdy neklesne pod cenu optimálního sekvenčního algoritmu. Celkový průběh časové složitosti tedy musí být linearitmický a podařilo se experimenty ověřit teoretickou časovou složitost.