PRL – Paralelní a distribuované algoritmy
Merge-splitting sort

Jméno a příjmení: Klára Nečasová
Login: xnecas24
Akademický rok: 2017/2018

Úvod

Cílem projektu bylo pomocí knihovny Open MPI a v jazyce C/C++ implementovat paralelní řadící algoritmus Merge-splitting sort.

Rozbor a analýza algoritmu

Paralelní řadící algoritmus Merge-splitting sort používá k řazení p procesorů, kterých je méně než řazených čísel n. Řazení čísel probíhá v iteracích, a to tak, že v každé iteraci nejdříve všechny sudé procesory (počítáno od nuly) seřadí svoji posloupnost a posloupnost svého pravého souseda. Totéž provede každý lichý procesor. Posloupnost čísel je seřazena po $\lceil \frac{p}{2} \rceil + 1$ iteracích, protože je třeba uvažovat i případ, kdy některé procesory mají o jeden prvek méně než ostatní procesory.

Algortimus pracuje následovně:

1. Procesor s $ID = 0$ nejdříve načte posloupnost čísel a rozdělí ji na přibližně stejně velké části, které pošle jednotlivým procesorům.
2. Všechny procesory seřadí paralelně obdrženou posloupnost optimálním sekvenčním algoritmem.
3. Všechny procesory provádí paralelně následující (celkem $\lceil \frac{p}{2} \rceil + 1$ krát):
   (a) Každý sudý procesor načte posloupnost od svého pravého souseda (pokud nějakého má), setřídí obě posloupnosti (svou i posloupnost souseda), následně ji rozdělí na dvě části a horní část odešle pravému sousedovi.
   (b) Každý lichý procesor provádí totéž co sudý procesor (pokud má nějakého pravého souseda) (viz 3a).
4. Každý procesor odešle svou posloupnost procesoru s $ID = 0$, který vypíše výslednou seřazenou posloupnost čísel.

Analýza složitosti algoritmu

Počet procesorů je dán vztahem: $p(n) = p$ a platí $p < n$.

1. V kroku 1 se načítá vstupní posloupnost n čísel, která se rozdělí a rozešle jednotlivým procesorům: $\mathcal{O}(p \cdot \frac{n}{p})$, což je $\mathcal{O}(n)$.
2. V kroku 2 probíhá seřazení posloupnosti optimálním sekvenčním algoritmem, který má složitost $\mathcal{O}(n \cdot \log n)$. Každý procesor musí seřadit $\frac{n}{p}$ prvků, časová složitost je tedy $\mathcal{O}((\frac{n}{p}) \log(\frac{n}{p}))$.
3. Krok 3 se provádí celkem $\lceil \frac{p}{2} \rceil + 1$ krát (liché a sudé procesory):
   - odeslání i příjem $\frac{n}{p}$ prvků má časovou složitost $\mathcal{O}(\frac{n}{p})$,
   - setřídění dvou seřazených posloupností do jedné seřazené posloupnosti má časovou složitost $\mathcal{O}(\frac{2n}{p}) = \mathcal{O}(\frac{n}{p})$.

   Krok 3 má tedy celkovou časovou složitost: $(\lceil \frac{p}{2} \rceil + 1) \cdot \mathcal{O}(\frac{n}{p}) \cdot 3 = \mathcal{O}(n)$.
4. V kroku 4 zasílají všechny procesory seřazené posloupnosti procesoru s $ID = 0$, tedy: $\mathcal{O}(p \cdot \frac{n}{p}) = \mathcal{O}(n)$.

Celkovou časovou složitost algoritmu můžeme vyjádřit vztahem:

$$t(n) = \mathcal{O}((\tfrac{n}{p})\log(\tfrac{n}{p})) + \mathcal{O}(n) = \mathcal{O}(\tfrac{n \cdot \log n}{p}) + \mathcal{O}(n).$$

Celková cena algoritmu:

$$c(n) = t(n) \cdot p(n) = (\mathcal{O}(\tfrac{n \cdot \log n}{p}) + \mathcal{O}(n)) \cdot p(n) = \mathcal{O}(n \cdot \log n) + \mathcal{O}(n \cdot p).$$

Pokud platí $p \leq \log n$, potom je algoritmus optimální.

Implementace

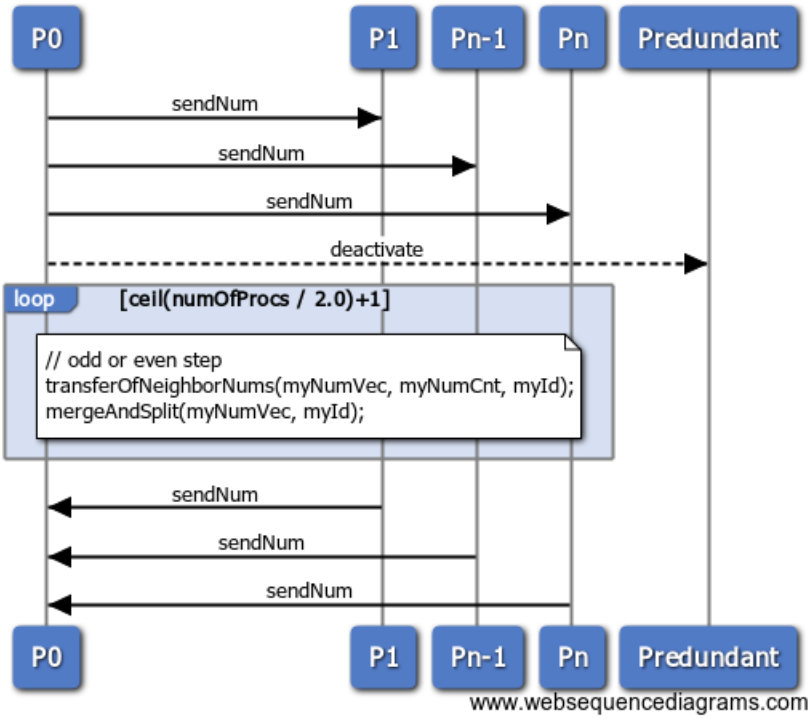
Pro implementaci byl zvolen jazyk C++ a knihovna Open MPI. Implementace se téměř shoduje s algoritmem uvedeným na přednáškách předmětu PRL. Dále tedy zdůrazníme pouze odlišnosti.

Procesory jsou číslovány od 0 nikoliv od 1, první procesor má tedy index 0. Ten provádí načtení vstupního souboru s čísly, jejich distribuci ostatním procesorům a příjem seřazených posloupností od ostatních procesorů.

Pokud je délka vstupní neseřazené posloupnosti n dělitelná počtem procesorů p, potom dostane každý procesor přiděleno $\frac{n}{p}$ čísel. Jinak získá prvních $(n \mod p)$ procesorů $\frac{n}{p}$ hodnot a zbývajících $(p - (n \mod p))$ procesorů $\frac{n}{p} - 1$ hodnot.

Komunikační protokol

Komunikační protokol zachycuje sekvenční diagram 1. Procesor $P0$ nejdříve rozešle příslušné posloupnosti čísel ostatním procesorům, redundantní procesory jsou deaktivovány. Následuje samotný algoritmus Merge-splitting sort. Nejdříve všechny sudé procesory setřídí svoji posloupnost a posloupnost pravého souseda, poté totéž provedou liché procesory. Na závěr odesílá každý procesor svou seřazenou posloupnost hlavnímu procesoru $P0$.



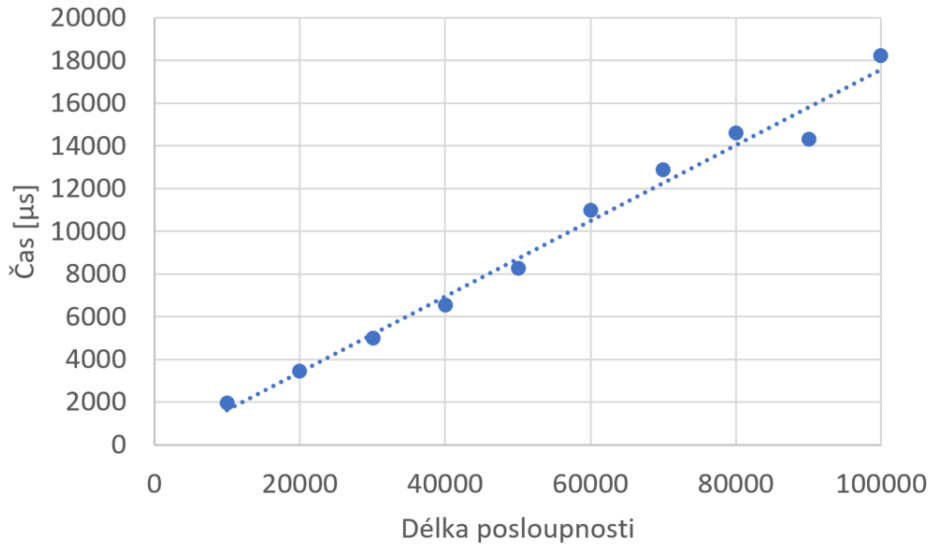
Obrázek 1: Grafické znázornění komunikačního protokolu

Experimenty

Pro experimentální ověření časové složitosti algoritmu byla vytvořena sada s deseti testovacími vstupy. Délky neseřazených posloupností byly zvoleny v intervalu od 10 tisíc do 100 tisíc. Pro

každou délku vstupní posloupnosti byl program spuštěn celkem desetkrát a poté byl vypočítán průměrný čas. Časová náročnost výpočtu byla měřena pomocí knihovny `chrono`. Zdrojový kód byl pro tyto účely modifikován, na dvě místa byla vložena časová razítka. První bylo vloženo za výpis vstupní neseřazené posloupnosti a druhé před výpis výsledku. Celková doba výpočtu byla získána odečtením těchto hodnot od sebe.

Graf na obrázku 2 znázorňuje závislost doby běhu algoritmu na délce vstupní posloupnosti. V předchozí sekci jsme odvodili časovou složitost $t(n) = \mathcal{O}(\frac{n \cdot \log n}{p}) + \mathcal{O}(n)$. Výsledná křivka odpovídá teoretické časové složitosti algoritmu, má téměř lineární průběh.



Obrázek 2: Závislost doby běhu algoritmu na délce vstupní posloupnosti

Závěr

Cílem projektu bylo implementovat paralelní algoritmus Merge-splitting sort. Nejdříve byl proveden rozbor a analýza celého algoritmu, včetně odvození jeho teoretické časové složitosti. Další část dokumentu se věnovala samotné implementaci a komunikačnímu protokolu, který byl znázorněn pomocí sekvenčního diagramu. V poslední části byly provedeny experimenty, které ověřily časovou složitost implementovaného algoritmu.