PRL – Projekt 1 Odd-Even Merge Sort

Algoritmus Odd-Even Merge Sort je paralelní řadicí algoritmus fungující na principu slučování seřazených posloupností.

Algoritmus

Algoritmus k řazení používá síť procesorů, jejíž základní jednotkou je síť 1×1 reprezentovaná jedním procesorem, která má dva vstupy a dva výstupy. Na každém vstupu má jedno číslo (tedy seřazenou posloupnost o délce 1). Výstupem sítě je seřazená posloupnost, kdy na jeden výstup **L** (**low**) je vrácena menší hodnota a na druhý **H** (**high**) hodnota vyšší. Každý procesor tedy slouží k řazení dvou čísel.

Sítě $n \times n$ o delších vstupech se vytvářejí ze základních sítí 1×1, viz diagram 1. Síť má na vstupu dvě seřazené posloupnosti o délce `n` a je složena ze dvou úrovní. Na první úrovni jsou dvě sítě $\frac{n}{2} \times \frac{n}{2}$, kde první má na vstupu postupně čísla na lichých pozicích obou vstupních posloupností a druhá síť má na vstupu čísla ze sudých pozic. První a poslední výstup těchto dvou sítí jde přímo na výstup, výstupy mezi nimi jdou ve druhé úrovni na vstup sítí 1×1, výstupy z první sítě jdou na vstupy **L** a výstupy druhé sítě jdou na vstupy **H**.

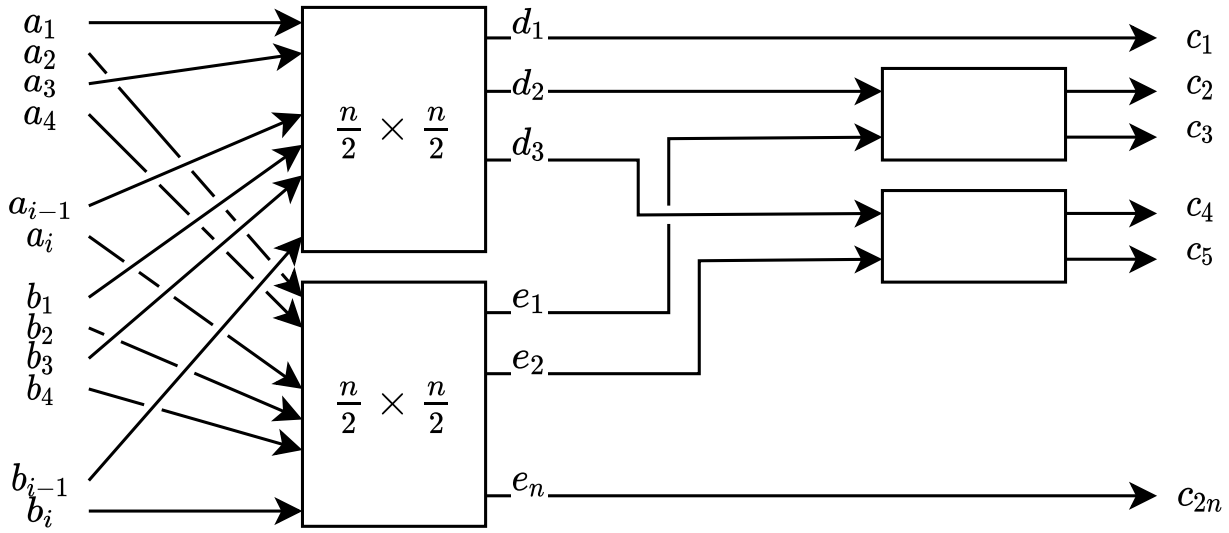

Diagram 1: Síť $n \times n$

Řazení je zajištěno kaskádou sítí, kde první úroveň je tvořená sítěmi 1×1, poté následují sítě dvojnásobné velikosti až po síť, která odpovídá polovině velikosti vstupu $\frac{n}{2}$.

Analýza složitosti

Každá úroveň řadicí sítě vytvoří posloupnost seřazených sekvencí dvojnásobné délky až po poslední úroveň, která obsahuje síť, která na vstup přijímá sekvenci odpovídající délce vstupu. Počet úrovní řadicí sítě je proto $\log_2 n$.

Vytvoření seřazených posloupností v první úrovni používá sítě 1×1 tvořené jediným procesorem, časová složitost pro vstup o délce 2 je tedy 1. Pro delší posloupnosti při každém zdvojnásobení vstupu přibude 1 úroveň procesorů. Z čehož vyplývá, že pro síť s n prvky na vstupu, kde $n = 2^i$, je časová složitost $t(2^i) = t(2^{i-1}) + 1$. Podle [1] potom můžeme napsat:

$$s(2) \quad = \quad 1$$

$$s(2^i) \quad = \quad s(2^{i-1}) + 1$$

z čehož vyplývá, že složitost pro jeden blok řadicí sítě (slučovací síť) odpovídá $s(2^i) = i$. Složitost celé řadicí sítě je potom následující:

$$t(n) = \sum_{i=1}^{\log n} s(2^i) = O(\log^2 n)$$

Počet procesorů v síti se vstupem o délce n odpovídá počtu procesorů ve dvou polovičních sítích, ke kterému se přičtou procesory ve druhé vrstvě sítí 1×1. Dle [1] tedy:

$$\begin{aligned} q(2) &= 1 \\ q(2^i) &= 2q(2^{i-1}) + 2^{i-1} - 1 \end{aligned}$$

potřebný počet procesorů pro jednu slučovací síť je potom $q(2^i) = (i-1)2^{i-1} + 1$. Potřebný počet procesorů pro celou řadicí síť je:

$$p(n) = \sum_{i=1}^{\log n} 2^{(\log n)-1} q(2^i) = O(n \log^2 n)$$

Na základě časové složitosti a potřebného počtu procesorů můžeme odvodit cenu algoritmu.

$$c(n) = t(n)p(n) = O(n \log^4 n)$$

Algoritmus tedy není optimální, ale je výrazně rychlejší než sekvenční řadicí algoritmy.

Implementace

Cílem projektu bylo tento algoritmus implementovat pro seřazení vstupní sekvence čísel o fixní délce 8. Pro seřazení této posloupnosti je potřebná řadicí síť o třech úrovních, první obsahuje 4 sítě 1×1, druhá 2 sítě 2×2. Poslední úroveň 4×4 má na výstupu seřazenou posloupnost. Síť je zakreslená na diagramu 2. Pro implementaci byl použit jazyk `C++` s knihovnou `Open MPI`.

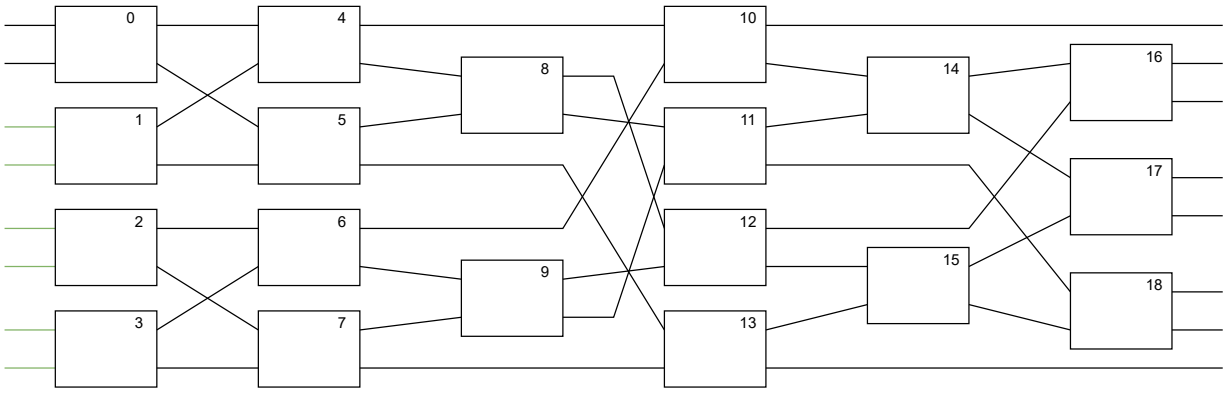

Diagram 2: Plná síť pro řazení 8 čísel

Celkem je pro provedení algoritmu potřebných 19 procesorů, pokud program nemá dostatečné množství procesorů, tak se ukončí. Jinak se po spuštění programu v procesoru s číslem **rank** 0 (**root**) načte 8 čísel o velikost 1 byte ze vstupního souboru `numbers` (konstanta). Procesor **root** čísla vypíše a poté je rozešle ostatním procesorům v první úrovni, tyto vstupy jsou v diagramu 2 zaznačeny zelenou barvou.

Následně všechny procesory zahájí přijetí dvou čísel pomocí funkce `MPI_Recv`, přijatá čísla následně porovnají, čímž získají hodnoty **low** a **high**, a ty odešlou následujícím procesorům pomocí `MPI_Send`. Vzhledem k tomu, že každý procesor pouze přijme a následně odešle čísla, tak nemůže nastat **deadlock**. Propojení mezi procesory jsou zadány fixně a každý procesor zvolí hodnoty, které odpovídají jeho číslu **rank**.

Procesor **root** v první fázi pouze seřadí dvě čísla a odešle je odpovídajícím procesorům. Následně od 5 procesorů přijme čísla, která jsou výstupem řadicí sítě, čímž získá seřazenou posloupnost, kterou nakonec vypíše.

Závěr

Vzhledem k tomu, že je zadána fixní délka vstupu, tak je řadicí síť dopředu známa. Jednotlivým procesorům je proto jejich role v rámci algoritmu předdefinována pomocí konstant, což implementaci zjednodušilo. Program ale nepodporuje vstup o žádné jiné délce.

Vzhledem k tomu, že všechny procesory, kromě procesoru **root**, pouze přijímají čísla a následně je odesílají, tak není nutné použít žádnou další synchronizaci než čekání na přijetí zprávy od procesorů, jejichž číslo **rank** je předem známo a zadáno konstantou. Procesor **root** odesláním čísel ostatním procesorům v první úrovni prakticky zahajuje výpočet a poté začne čekat na výsledek.

Algoritmus se podařilo pomocí knihovny **Open MPI** úspěšně implementovat. Vytvořený program korektně řadí vstupní posloupnost 8 čísel. Zároveň byla pro algoritmus odvozena časová složitost a potřebný počet procesorů pro vstup o obecné délce `n`.

Literatura

[1] Akl, S. G.: *The design and analysis of parallel algorithms*. New Jersey: Prentice-Hall, 1989, ISBN 0-13-200056-3.