PRL PROJEKT 1

Odd-Even Merge Sort

Implementace Odd-Even Merge Sort algoritmu pomocí procesorů.

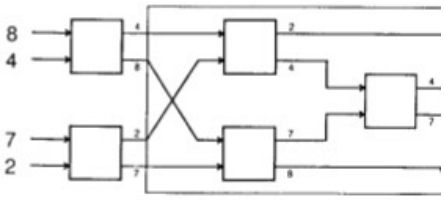
Autor: Monika Rosinská, xrosin00

Datum: 9. 4. 2022

Princip algoritmu

Odd-even merge sort využívá k řazení síť procesorů, z nichž každý má 2 vstupy a jeden výstup, a každý umí porovnat dvě hodnoty. Procesor vezme dvě hodnoty na vstupu, a nižší hodnotu dá na první (vrchní) výstup a vyšší hodnotu na druhý (spodní) výstup. Tyto procesory tvoří kaskádu, jejíž komplexita se odvíjí od počtu řazených prvků.

Lepší představu lze získat při analýze sítě, která řadí 4 prvky:



První dva procesory (nejvíce vlevo) seřadí své prvky, a dají je ve správném pořadí na výstup. Další dva procesory následně porovnají dva větší prvky (horní procesor), a dva menší prvky (dolní procesor). Prvek, který vyjde z horního procesoru prvním výstupem je určitě největší – porovnali jsme jej s původním prvkem z dvojice, a poté s větším prvkem z druhé dvojice. Obdobně to platí i pro nejmenší prvek, který vyjde druhým výstupem u druhéheho procesoru. Poslední krok je porovnání prostředních prvků – na to využijeme poslední procesor. Výsledkem je seřazená posloupnost 4 čísel.

Při rozšíření sítě pro 8 prvků získáme následující síť:

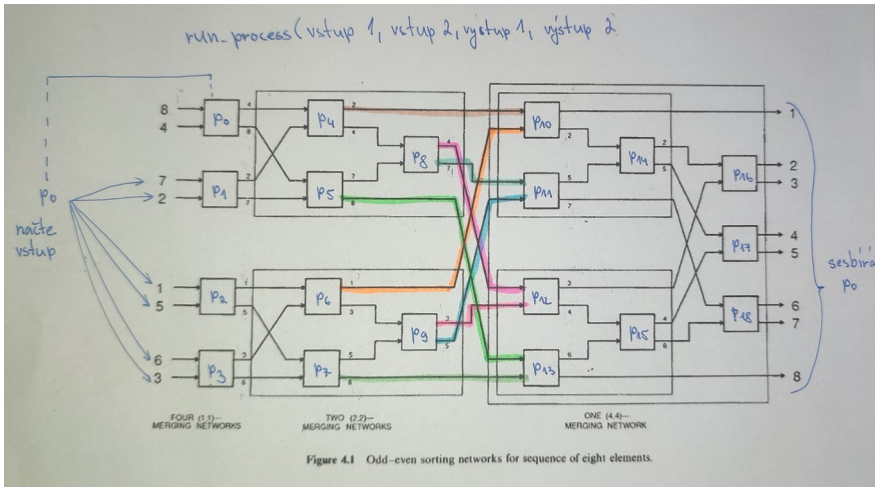


Figure 4.1 Odd–even sorting networks for sequence of eight elements.

Dvě části vlevo jsou totožné s prvem pro řazení 4 prvků, získáme tak dvě seřazené posloupnosti o délce 4 prvků. U řazení 8 prvků je potřeba poté porovnat nejvyšší hodnoty z těchto dvou částí, a dvě nejnižší hodnoty z nich. Obdobně je poté potřeba porovnat i prostřední hodnoty. Na schématu je vidět, že i kdyby nejvyšší hodnota ze spodní části byla menší než všechny hodnoty z vrchní části, tak tato hodnota postupně “propadne” zpět dolů přes procesory 10, 14 a 17.

Komplexita a propojení sítě se mění s počtem řazených prvků.

Analýza složitosti

Analýza složitosti tohoto algoritmu je poměrně komplikovaná, a vyžaduje hlubší analýzu. Nejdříve zanalyzujme, kolik úrovní (fází) je pro seřazení *n* prvků potřeba:

- víme, že počet seřazených prvků se s každou další úrovní zdvojnásobí

- jednoduchou úvahou dojdeme k závěru, že pro seřazení *n* prvků je potřeba *log n* úrovní

Počet potřebných procesorů

Při řazení 2 prvků je potřeba 1 procesor, pro řazení 4 prvků je potřeba 5 procesorů, pro řazení 8 prvků je potřeba 19 procesorů, a tak dále. Bohužel počet procesorů se při tomto pohledu nejeví jako logické číslo. Je třeba použít kreativnější způsob nalezní počtu procesorů:

- předpokládejme, že pro seřazení dvou seřazených posloupností, které mají 2^{i-1} prvků, potřebujeme $q(2^i)$ komparátorů

- víme, že pro seřazení 2 posloupností o 1 prvku potřebujeme 1 porovnání → získáme rovnici:

  1. $q(2) = 1$

- podívejme se na zbylé možnosti: pro seřazení seřazených posloupností o 2 prvcích potřebujeme 3 komparátory, pro seřazení posloupností o 4 prvcích potřebujeme 9 komparátorů, ...

- díky této úvaze získáme druhou rovnici:

  2. $q(2^i) = (i - 1) * 2^{(i-1)} + 1$

- počet potřebných procesorů poté jednoduše určíme pomocí vzorce $p(n) = \sum_{i=1}^{\log n} 2^{(\log n)-i} q(2^i)$ [1]$= O(n * log^2 n)$

Čas potřebný pro výpočet

- víme, že čas potřebný pro seřazení 2 čísel je konstantní; můžeme si to představit také jako čas, který zabere jednomu “sloupci” procesorů, než vstup přesune na výstup

- víme, že pro seřazení *n* čísel je potřeba *log n* úrovní, zbývá zjistit, kolik sloupců procesorů je v každé úrovni

- podíváme-li se na nákresy řadících sítí, můžeme si všimnout, že počet sloupců procesorů s každou úrovní (tj. s každým zvyšujícím se n) vzrůstá o 1; tento trend se udržuje i u řazení většího počtu prvků

- získáváme tedy řadu: p(s) = 1 + 2 + 3 + ... + log n; řada je od 1 do log n, jelikož máme log n úrovní

- součet této řady je $log^2 n$, tedy celkově máme $\log^2$ n sloupců procesorů

- jelikož sloupci procesorů trvá seřazení svých vstupů a jejich odeslání na výstup, získáváme časovou složitost $t(n) = O(c) * O(log^2 n) = O(log^2 n)$

Cena řešení

Cena řešení $c(n) = p(n) * t(n)$, kde $p(n)$ je počet potřebných procesorů a $t(n)$ je čas potřebný pro seřazení n prvků.

Tedy: $c(n) = O(n * log^2 n) * O(log^2 n) = O(n * log^4 n)$, což není optimální cena (optimální cena je $O(n * log n)$).

Zhodnocení analýzy

Ačkoli algoritmus není optimální, je velmi rychlý – jeho časová složitost je jen $O(log^2 n)$. Důvod, proč není optimální, je velmi vysoký počet procesorů, který je pro seřazení čísel potřeba, který je u rozmného počtu řazených čísel nereálný.

---

1 suma je do *log n* z důvodu, že potřebujeme přesně tolik úrovní pro seřazení *n* čísel, jak bylo popsáno výše; ostatní symboly mají stejný význam, jako bylo popsáno v předchozím odstavci

Implementace

Algoritmus byl implementován pomocí openMPI knihovny v jazyce C++. Pro seřazení posloupnosti 8 čísel bylo využito 19 procesorů, které spolu komunikují výhradně pomocí funkcí *MPI_Send* a *MPI_Recv*, a svých ID, které mají hodnotu 0 – 18.

Za hlavní, takzvaný řídící procesor, je považován procesor s ID 0. Tento procesor načte čísla ze souboru *numbers*, seřadí první dvě hodnoty, a poté rozešle hodnoty všem ostatním procesorům. Následně procesor čeká na seřazená čísla od ostatních procesorů (konkrétně od procesorů 10, 13, 16, 17 a 18). Tato čísla čte v pořadí od nejmenšího po největší (procesor 0 nepřečte hodnotu, dokud nepřečetl všechny předcházející, tedy menší, hodnoty). Postupně tyto obdržené hodnoty tiskne na výstup.

Ostatní procesory čekají, dokud nedostanou na vstupy hodnoty, poté je seřadí, a pošlou je na správné vstupy dalších procesorů. Který procesor komunikuje se kterým je vidět na obrázku na konci str. 1 (např. procesor s ID 8 přijímá data od procesorů 4 a 5 a posílá je procesorům 12 a 11). Procesory získávají informaci o tom, komu poslat kterou hodnotu, pomocí pole *task*, které obsahuje 18 záznamů, které v uvedeném pořadí obsahují následující hodnoty:

i. ID procesoru, který mi zašle první vstup

ii. ID procesoru, který mi zašle druhý vstup

iii. ID procesoru, kterému mám zaslat nižší hodnotu (vrchní výstup)

iv. ID procesoru, kterému mám zaslat vyšší hodnotu (spodní výstup)

K tomuto poli poté procesory přistupují pomocí svého ID, kdy jejich údaje jsou na řádku ID-1.

Čekání procesorů na všechny vstupy, a čekání hlavního procesoru na seřazené hodnoty, je synchronizováno výhradně pomocí výše zmíněných MPI funkcí. Ty samy implementují aktivní čekání na vstup, a procesor nemůže vykonávat další práci, dokud tuto zprávu neobdrží/úspěšně neodešle.

Procesory jsou vzájemně propojeny tak, jak je uvedeno na schématu:

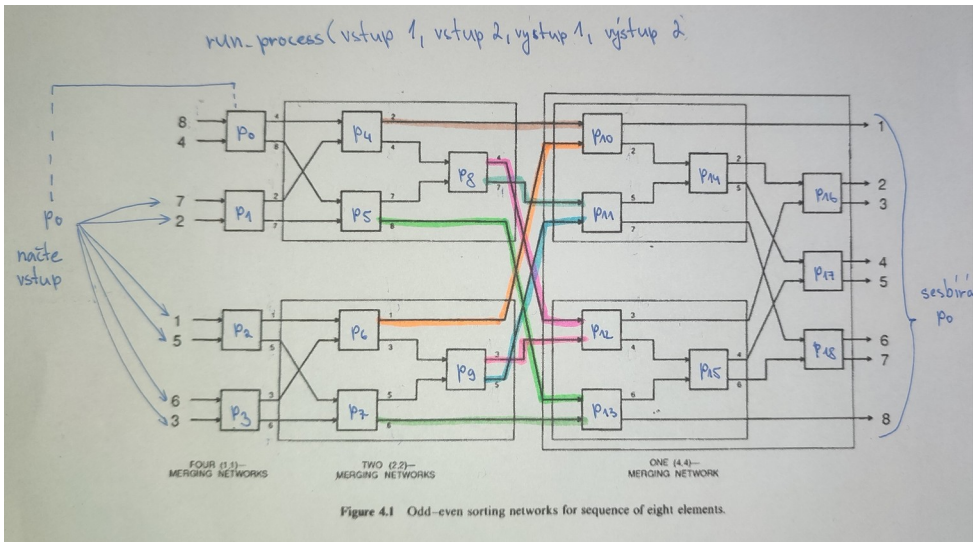


Figure 4.1 Odd–even sorting networks for sequence of eight elements.

Závěr a zhodnocení

Tento projekt úspěšně implementuje Odd-Even Merge Sort pomocí 19 procesorů a pomocí knihovny MPI. Současně byla provedena analýza prostorové a časové složitosti, a tedy i cena tohoto algoritmu, která je bohužel ale velmi vysoká, a proto neumožňuje praktické využití tohoto algoritmu – řešením ceny je snížit prostorovou složitost, což by bylo možné se "znovupoužitím" procesorů, protože při současné implementaci vždy nějaké procesory "stojí" a nevykonávají žádnou práci (vždy pracuje jen jedna úroveň procesorů). To by ovšem vneslo do implementace komplikovanější řídící logiku, což by algoritmus mírně zpomalovalo – ovšem pokles na ceně by byl výrazný, a proto se toto řešení jeví dobrým.