1 Rozbor a analýza algoritmu Odd-even merge sort

Algoritmus Odd-even merge sort na svoje radenie využíva špeciálnu sieť procesorov. Každý procesor, ktorý sa nazýva CE (comparison element) má dva vstupné a dva výstupné kanály a jeho funkciou je komparácia dvoch čísel. Procesor porovná dve čísla zo vstupu, ktoré boli výstupom predošlých procesorov a na svoj výstup uloží hodnoty L (minimum) a H (maximum). Takéto siete môžu byť typicky až rozmerov $k \times k$ a ich veľkosť závisí od vstupnej nezoradenej sekvencie n čísel, kde $n = 2^m$, kde $m > 0$. Na zoradenie n čísel je potrebné v prvej úrovni vytvoriť sieť CE o rozmeroch 1×1, ktorej veľkosť je $\frac{n}{2}$ a vytvorí $\frac{n}{2}$ zoradených sekvencií čísel o dĺžke 2. V druhej úrovni sú páry týchto postupností spojené do zoradenej postupnosti o dĺžke 4 s využitím siete o rozmeroch 2×2. V tretej fáze sú znovu tieto páry sekvencií o dĺžke 4 spojené pomocou siete o rozmeroch 4×4 do sekvencií o dĺžke 8. Takýmto spôsobom by sme pokračovali, dokým dve sekvencie čísel o dĺžke $\frac{n}{2}$ nie sú spojené sieťou o rozmeroch $\frac{n}{2} \times \frac{n}{2}$, čím sa nakoniec vytvorí jedna konečná zoradená sekvencia čísel o dĺžke n [1].

1.1 Časová zložitosť algoritmu

Označme $s(2^i)$ ako čas, potrebný v *i-tej* úrovni siete na spojenie dvoch zoradených sekvencií čísel, kde každá sekvencia obsahuje 2^{i-1} čísel. Následne môžeme povedať, že platia nasledujúce rovnice:

$$\begin{aligned} s(2) &= 1, && \text{kde } i = 1, \\ s(2^i) &= s(2^{i-1}) + 1, && \text{kde } i > 1, \end{aligned}$$

potom platí, že $s(2^i) = i$. Z toho vyplýva, že čas potrebný na zoradenie sekvencie čísel o dĺžke n v danej sieti je

$$\begin{aligned} t(n) &= \sum_{i=1}^{\log n} s(2^i) \\ &= \mathcal{O}(\log^2 n) \end{aligned}$$

1.2 Priestorová zložitosť algoritmu

Označme $q(2^i)$ počet potrebných CE v *i-tej* úrovni siete na spojenie dvoch zoradených sekvencií čísel, kde každá sekvencia obsahuje 2^{i-1} čísel. Následne môžeme povedať, že platia nasledujúce rovnice:

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

potom platí, že $q(2^i) = (i-1)2^{i-1} + 1$. Z toho vyplýva, že počet komparatívnych procesorov v sieti na zoradenie sekvencie čísel o dĺžke n je

$$\begin{aligned} p(n) &= \sum_{i=1}^{\log n} 2^{(\log n)-i} q(2^i) \\ &= \mathcal{O}(n \log^2 n) \end{aligned}$$

1.3 Celková cena algoritmu

Keďže poznáme časovú aj priestorovú zložitosť, ľahko môžeme odvodiť aj celkovú cenu Odd-even merge algoritmu, kde celkový počet porovnaní v tejto sieti je

$$\begin{aligned} c(n) &= p(n) \times t(n) \\ &= \mathcal{O}(n \log^4 n). \end{aligned}$$

Z toho vyplýva, že cena tohto algoritmu nie je optimálna, keďže vykonáva viac ako $\mathcal{O}(n \log n)$ operácií, ktoré sú kritériom pre optimálnosť.

2 Implementácia

Algoritmus bol implementovaný v jazyku C++, kde bola využitá knižnica *OpenMPI*. V ostatných prípadoch boli použité základné dátové štruktúry, ktoré C++ poskytuje, najmä *std::vector*.

2.1 Analýza pre počet procesorov

Na základe rozboru z kapitoly 1 môžme uvážiť, že veľkosť spojovaných sekvencií čísel sa bude po každej úrovni v sieti zdvojnásobovať. Platí, že $n = 2^m$, kde $n = 8$, z čoho vyplýva že počet úrovní v sieti je $m = \log_2 8 = 3$. Jednotlivé úrovne v sieti potrebujú následovné počty CE:

- Prvá úroveň: 2^{m-1} CE $= 4$
- Druhá úroveň: 2^{m-2} 2×2 spojovacích sietí, kde každá potrebuje 3 CE $= 6$
- Tretia úroveň: 2^{m-3} 4×4 spojovacích sietí, kde každá potrebuje 9 CE $= 9$

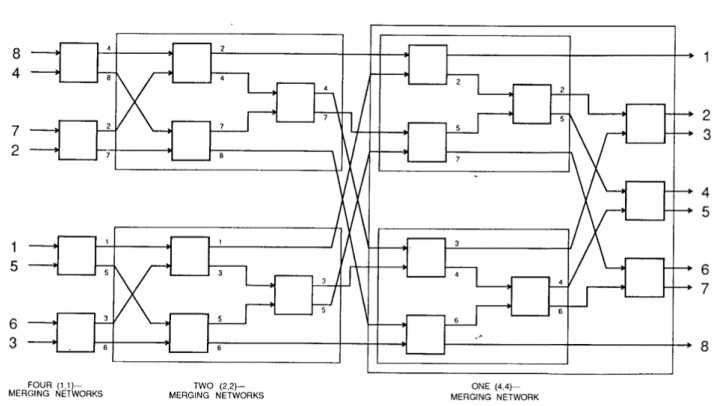
V prípade tohto projektu bude teda zoradená vždy fixná postupnosť 8 čísel a výsledná sieť bude zložená z troch úrovní, kde v prvej úrovni bude sieť o rozmeroch 1×1 zložená zo 4 procesorov, v druhej úrovni o rozmeroch 2×2 bude 6 procesorov a v poslednej tretej úrovni o rozmeroch 4×4 bude 9 procesorov. Dohromady bude potrebných 19 komparatívnych elementov v sieti. K tomu je však potrebné ešte pripočítať jeden riadiaci procesor, ktorého funkcionalita je popísaná v sekcií 2.2. Výsledný program bude manipulovať s celkovým počtom 20 procesorov.

2.2 Prepojenie procesorov a ich synchronizácia

V programe sú jednotlivé procesory prepojené pomocou predom vytvorenej matice, ktorá obsahuje pre každý procesor P pridelenie *id* iného procesoru, ktorý mu bude zasielať svoju zoradenú postupnosť, a teda buď hodnotu z výstupu L alebo H, kde poradie vstupov nezohráva žiadnu úlohu. Súčasne má tento procesor priradené dve id hodnoty následujúcich procesorov, kde je potrebné dbať na určenie správnej hodnoty L a H pre svojich príjemcov. Matica 2.2 o rozmeroch 20×4 pre ilustratívnosť zobrazuje bližšie zakódovanie architektúry procesorov v sieti, kde nultý riadok spôsobuje redundantnosť no je použitý pre lepšiu orientáciu v indexácií samotných procesorov v kóde. Stĺpce v matici určujú postupne dvoch odosielateľov a dvoch príjemcov.

$$RT = \begin{bmatrix} \text{Rec1\_id0} & \text{Rec2\_id0} & \text{Send\_min\_id0} & \text{Send\_max\_id0} \\ \vdots & & & \\ \text{Rec1\_id19} & \text{Rec2\_id19} & \text{Send\_min\_id19} & \text{Send\_max\_id19} \end{bmatrix}$$

Obrázok 2.1 znázorňuje logické prepojenie procesorov, ktoré tvoria výslednú sieť na zoradenie 8 vstupných hodnôt a slúži ako ilustratívne zobrazenie vnútornej matice, ktorá bola vyhotovená podľa tejto predlohy a samotné číslovanie procesorov sa v skutočnosti trochu odlišuje.



Obr. 2.1: Radiaca sieť 19 procesorov, ktorej prepojenie je zakódované v matici [1].

Procesory sú synchronizované pomocou príkazov z knižnice OpenMPI, kde operácie *send()* a *receive()* sú vždy blokujúce operácie a pokiaľ nebudú prijaté alebo zaslané obe hodnoty z výstupov L a H, tak procesor musí čakať.

Výnimku v chovaní procesorov predstavuje takzvaný *master* proces, ktorého úlohou je na začiatku vypísať nezoradenú postupnosť čísel na štandardný výstup a poslať do predom stanovených procesorov 8 hodnôt zo vstupu a následne čakať, až dokým neobdrží výslednú zoradenú postupnosť 8 hodnôt zo siete procesorov. Na záver tento proces vypíše zoradenú postupnosť na štandardný výstup a algoritmus sa skončí. Tento procesor má v programe zakódovanú vlastnú maticu, ktorá určuje do akých procesorov má poslať nezoradenú postupnosť, a od akých procesorov má získať výslednú postupnosť. Veľkosť matice nultého riadiaceho procesoru je o rozmeroch 2×8, kde riadky riadky určujú prijímajúce a odosielajúce procesory postupne a stĺpce sú určené pre jednotlivé id procesorov. Samotné zakódovanie matice je veľmi podobné matici 2.2.

2.3 Fungovanie systému pre osem hodnôt na vstupe

Algoritmus na zoradenie náhodne vygenerovaných 8 hodnôt z rozsahu 0-255 najskôr spracuje túto postupnosť čísel z binárneho súboru a následne ich uloží do vektoru celých čísiel. Následne sú tieto

čísla postupne zasielané jednotlivým vstupným procesorom v radiacej sieti podľa indexácie vnútorne reprezentovanej matice procesorov. Tieto kroky vykoná na začiatku nultý riadiaci procesor podľa algoritmu 1, ktorý musí následne čakať na výsledne zoradené hodnoty zo siete procesorov, pričom má v cykle spustenú blokujúcu operáciu na prjímanie výsledne zoradenej postupnosti čísel. Identifikátory odosielajúcich procesorov musí byť pre master proces nastavený v správnom poradí aby bola vo výsledku na výstupe korektne zoradená postupnosť čísel, čo je znovu zabezpečené procesorovou maticou. Počas toho sa spustí práca procesorov 1..19, ktorých úlohou je vykonávať komparáciu dvoch hodnôt, na ktoré vždy musí počkať do obdržania. Následne porovná 2 hodnoty, z ktorých určí minimum a maximum a tieto zoradené hodnoty ďalej posiela určeným procesorom, podľa matice 2.2. Činnosť komparatívnych procesorov CE je v jednoduchosti popísaná algoritmom 2.

Algorithm 1 Činnosť procesu 0

```
while je číslo na vstupe do
    načítaj a vypíš načítané číslo;
    pošli načítané číslo ďalšiemu procesoru;
end while
while neprijal všetky čísla do
    prijmi zoradené číslo od procesoru;
    ulož číslo do bufferu;
end while
while je číslo v bufferi do
    vypíš zoradené číslo;
end while
```

Algorithm 2 Činnosť procesov 1..19

```
while neprišli dve čísla do
    čakaj na prijatie 2 čísel od procesorov;
end while
vytvor minimum a maximum pre prijaté čísla;
pošli minimum a maximum pre určené procesory z matice;
```

2.4 Zhodnotenie algoritmu

Tento algoritmus sa stáva nepraktický na zoradenie veľkých vstupných sekvencií čísel. Napriek tomu je veľmi rýchly pre zoradenie veľkej sekvencie čísiel, kde pre veľkosť 2^{20} je schopný danú sekvenciu zoradiť do 20^2 časových jednotiek. Pre porovnanie by napríklad algoritmus *quicksort* takto dlhú postupnosť zoradil až do približne 20 miliónov časových jednotiek. Problémom však aj naďalej zostáva jeho procesorová zložitosť, kde by pre 2^{20} dlhú sekvenciu čísel potreboval viac ako 400 miliónov CE. To vo výsledku vedie na to, že ak by nás zaujímala iba časová efektivita algoritmu, tak je Odd-even merge sort optimálnym riešením no jeho celková cena zostáva stále za hranicou optimálnosti.

Bibliografia

[1] Selim G. Akl. *Parallel sorting algorithms*. Notes and reports in computer science and applied mathematics 12. Orlando: Academic Press, 1985. ISBN: 9780120476800.