Pipeline merge sort
Dokumentácia k projektu č. 2

Peter Lacko
xlacko06@stud.fit.vutbr.cz

1. dubna 2015

Cieľom tohoto projektu bolo implementovať paralelnú verziu radiaceho algoritmu Merge sort: Pipeline merge sort. Algoritmus bol implementovaný v jazyku C/C++ s využitím knižnice *OpenMPI*[1]. Program dokáže radiť 8 bitové neznamienkové čísla uložené v súbore a jeho výstupom je zoznam týchto čísel na jednom riadku nasledovaný už vzostupne zoradenou postupnosťou – každé číslo na samostatnom riadku.

1 Popis a analýza algoritmu Pipeline merge sort

Algoritmus Pipeline merge sort pracuje nasledovne: Nech n je veľkosť vstupných dát a $r = (\log n) + 1$ počet použitých procesorov. Všetky procesory bežia paralelne a v jednom cykle dokážu prečítať číslo zo vstupu, porovnať dve čísla a poslať číslo na výstup. Procesor P_1 má jednu vstupnú frontu, označme ju q_1 a dve výstupné fronty, P_{r+1} má dve vstupné a jednu výstupnú frontu, $q_{2(r+1)}$. Procesory P_i a P_{i+1} spolu komunikujú pomocou dvoch front, q_{2i} a q_{2i+1}. Procesor P_1 číta čísla zo súboru a posiela ich procesoru P_2, ktorý ich ďalej posiela P_3 atd, až posledný procesor v sérii produkuje zoradenú postupnosť. Sekvenčný diagram ich komunikácie je znázornený na obrázku 1. Procesor P_i, pre $2 \leq i \leq r + 1$, má teda dve fronty dĺžky 2^{i-2} ktoré spája do postupnosti dĺžky 2^{i-1} a posiela ich procesoru P_{i+1}. Pri prijímaní postupností P_i alternuje medzi svojimi dvoma frontami, tj. po naplnení q_{2i} ukladá vstupné hodnoty do q_{2i+1} a naopak. P_i začína radiť (pošle vačšiu z hodnôt na začiatku svojich front), ak q_{2i} obsahuje 2^{i-2} prvkov a q_{2i+1} jeden prvok.

To znamená $2^{i-2} + 1$ cyklov po tom čo začal P_{i-1}. Pokiaľ teda P_1 začína radiť v prvom cykle, P_i začne o

$$1 + \sum_{j=0}^{i-2} 2^j + 1 = 2^{i-1} + i - 1 \tag{1}$$

cyklov neskôr a zastaví sa po ďalších $n - 1$ spracovaných prvkoch, teda v cykle $(n - 1) + 2^{i-1} + i - 1$. Posledný procesor P_{r+1} teda zastaví v cykle $n + 2^r + r - 1 = 2n + \log n - 1$, čo je i výsledná doba trvania radenia. Časová zložitosť algoritmu je potom

$$t(n) = \mathrm{O}(n) \tag{2}$$

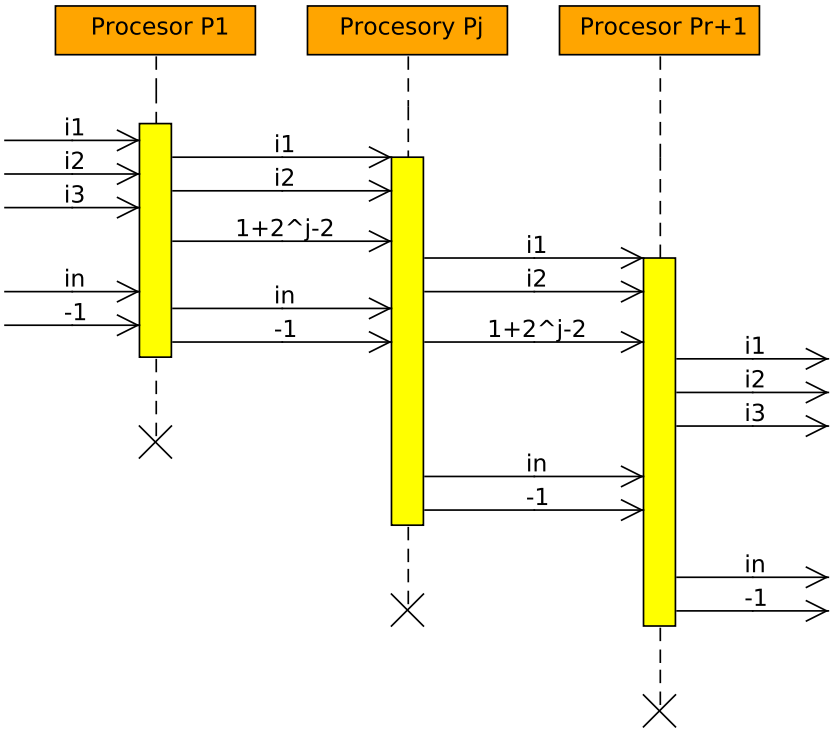
a jeho cena je

$$c(n) = t(n) \times p(n) = \mathrm{O}(n) \times (\log n + 1) = \mathrm{O}(n \log n), \tag{3}$$

čo je zároveň aj časová zložitosť optimálneho sekvenčného algoritmu. Algoritmus Pipeline merge sort teda je optimálny.

Fronty procesoru P_i má dohromady dĺžku nanajvýš $2^{i-2} + 1$ prvkov, čo nám pri počte procesorov $\log n$ dáva lineárnu priestorovú zložitosť $\mathrm{O}(n)$.

[1] OpenMPI implementuje štandard/špecifikáciu MPI (Message Passing Interface), popisujúci komunikáciu procesorov v paralelnom prostredí pomocou zasielania správ medzi nimi.



Obrázek 1: Sekvenčný diagram komunikácie procesorov.

2 Implementácia algoritmu

Implementácia algoritmu je priamočiara, samotný cyklus je rozdelený do troch hlavných častí: Prvou je načítanie hodnôt prvým procesorom a ich posunutie ďalšiemu procesoru. V druhej časti každý nasledujúci procesor skontroluje svoje fronty, a ak začal radiť, pošle na výstup väčšiu z hodnôt uložených na ich začiatkoch, resp. na `stdout` ak sa jedná o posledný procesor. V tretej časti naopak každý procesor okrem prvého čaká na správu od svojho predchodcu a obsah uloží do jednej zo svojich front. Po spracovaní všetkých hodnôt každý procesor odošle svojmu nasedovníkovi číslo -1, slúžiace ako zarážka a následne ukončí svoju činnosť. Posledný procesor v rade napokon vypíše výslednú zoradený postupnosť na stdout.

Fronty sú implementované ako vektor hodnôt typu `uint8_t` z knižnice `std` a to nasledujúcim spôsobom: každý procesor pracuje s dvoma dvojicami front, pričom manipulácia s nimi prebieha pomocou ukazovateľov. Jedna dvojica odkazuje na vstupné fronty – procesor do nich ukladá prijaté hodnoty, a truhá dvojica na výstupné – vyberá z nich pri radení. Po naplnení „vstupných front“ sa ukazovateľom na vstupné fronty priradí druhá dvojica front, obdobne tak pri ich vyprázdnení.

3 Spôsob testovania a dosiahnuté výsledky

Testovanie rýchlosti algoritmu prebiehalo na servere merlin pri jeho minimálnom vyťažení a to pre počet vstupných hodnôt 2^8 až 2^{20}. Výsledný čas je aritmetickým priemerom z desiatich behov, výsledky sú zobrazené v tabuľke 1. Čas bol meraný pomocou funkcie MPI_Wtime(), vracajúcej skutočný uplynutý čas na procesore. Meranie je vykonávané posledným procesorom a vyjadruje dobu od vstupu do prvého cyklu po ukončenie jeho činnosti (ale pred započatím výpisu hodnôt).

Z výsledkov je patrné, že doba behu rastie lineárne s veľkosťou vstupu (približne sa zdvojnásobuje), pokiaľ počet procesorov nedosiahne hodnotu 17, odkedy čas rastie približne švornásobne s dĺžkou vstupu. To je možné vysvetliť počtom procesorov na servere merlin – 12, kedy výpočty už neprebiehajú plne paralelne, ale čiastočne sekvenčne.

Počet vstupov	Nameraný čas [s]
256	0.002 756 309
512	0.004 427 506
1024	0.009 436 905
2048	0.020 318 68
4096	0.046 272 75
8192	1.091 675
16 384	0.994 661 7
32 768	1.452 801
65 536	1.911 85
131 072	4.172 162
262 144	13.863 61
524 288	53.087 88
1 048 576	211.9356

Tabulka 1: Nameraný čas predstavuje priemer z 10 behov programu pri danom veľkosti vstupu.

4 Zhodnotenie výsledkov

Výstupom projetu je program v jazyku C/C++ implementujúci algoritmus Pipeline Merge sort schopný zoradiť až 2^{20} hodnôt o veľkosti 1 byte. Radenie je vykonávané v čase $O(n)$ a to po hranicu, kedy sa paralelné výpočty začínajú počítať sekvenčne. Rád časovej zložitosti sa od určitej hranice potrebných procesorov zvyšuje, priestorová zložitosť je ale vždy lineárna. Pipeline Merge sort je z časového hľadiska optimálny algoritmus, čo ho robí mimoriadne vhodným na radenie vysokého počtu hodnôt v paralelnom prostredí.

Reference

[1] Selim G. Akl. *Parallel Sorting Algorithms.* Academic Press, Inc., 1990.