Paralelní a distribuované algoritmy
Projekt 1

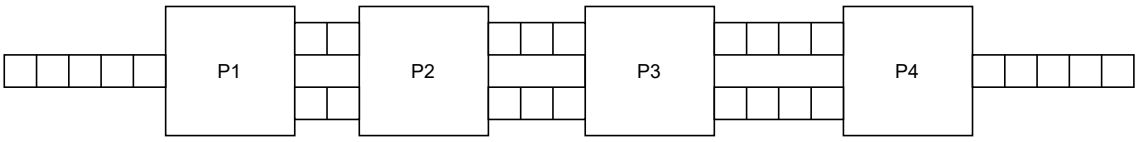
Branislav Mateáš (xmatea00)
xmatea00@stud.fit.vutbr.cz

1 Pipeline Merge Sort

Pipeline Merge Sort je paralelný algoritmus, ktorý je založený na zreťazení procesorov (obr. 1). Dáta postupne prechádadzaju od procesoru s nižším poradovým číslom k procesoru s vyšším poradovým číslom. Každý procesor s výnimkou prvého a posledného je spojený dvoma linkami, do ktorých ukladajú zoradené postupnosti. Prvý procesor obsahuje jednú vstupnú linku, ktorá obsahuje nezoradenú postupnosť a posledný procesor obsahuje jednu výstupnú linku, ktorá vo výsledku bude obsahovať výslednú zoradenú postupnosť.

Každý procesor (okrem prvého) vytvára zoradenú postupnosť dĺžky 2^i z dvoch prichádzajúcich postupností dĺžky 2^{i-1}. Takže obe vstupné linky procesora majú veľkosť 2^{i-1} a pre algoritmus platí, že pre vstup o veľkosť N vyžaduje $log_2(N)+1$ procesorov. Prvý procesor len načíta vstup a posiela ho striedavo na vstupné linky ďalšieho procesoru.

Výhodou toho, že s každým procesorom v poradí narastá aj veľkosť postupností, ktoré radí, je, že procesor sa stane aktívnym až, keď má na vstupe dostatočnú veľkosť postupností a teda nie všetky procesory sú zaneprázdnené vždy.



Obr. 1: Zreťazenie procesorov

1.1 Zložitosť algoritmu

Procesor P_i začína radiť ako náhle je na jednej linke postupnosť dĺžky 2^{i-1} a na druhej linke postupnosť dĺžky 1, to je $2^{i-1}+1$ cyklov po tom, čo začal P_{i-1}. Takže ak P_0 začal počas cyklu 1, P_i začal radiť v cykle

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

Po spracovaní všetkých zvyšných (n-1) prvkov, P_i zastaví v cykle $(n-1)+2^{i-1}+i-1$. Keďže P_{r+1} je posledný procesor, celkové zoradenie skončí v cykle

$$n + 2^r + r - 1 = 2n + log_2(n) - 1$$

teda O(n). Keďže $p(n) = log_2(n)+1$ tak cena algoritmu je:

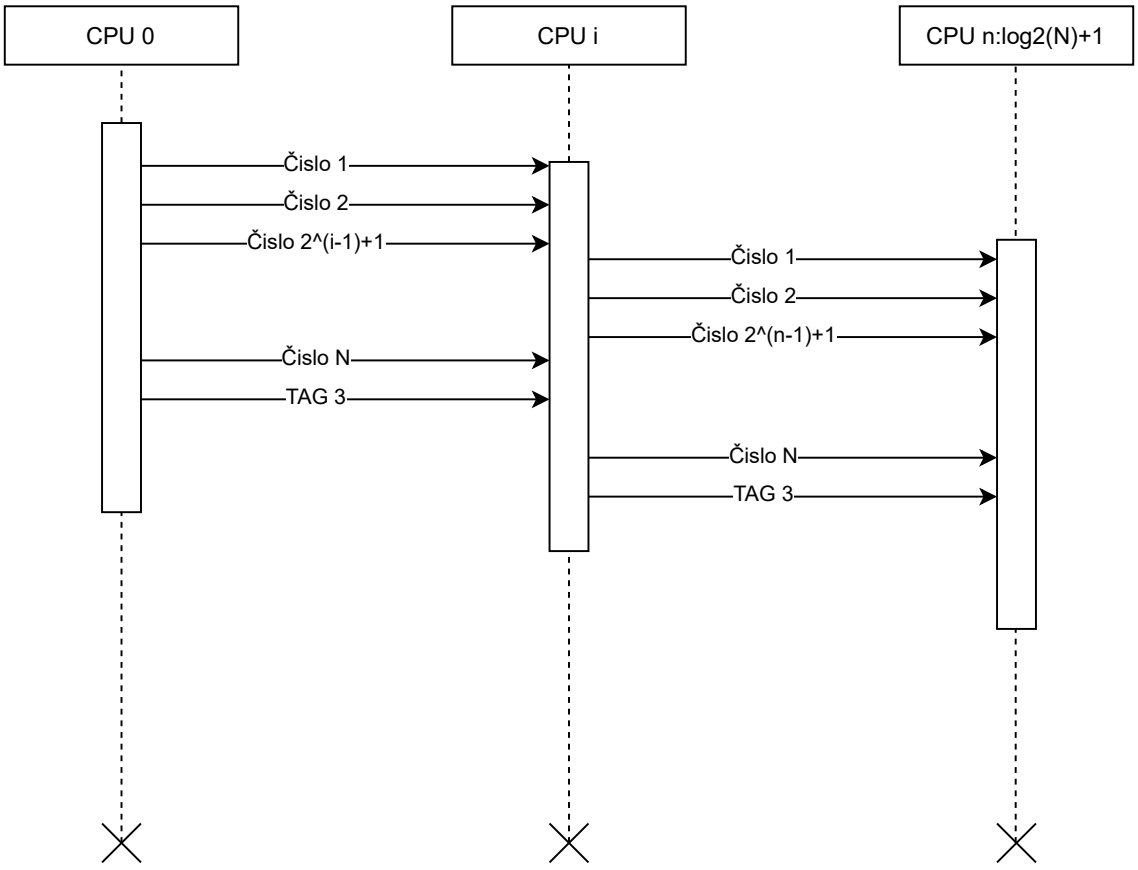
$$c(n) = t(n) * p(n) = O(n) * (log_2(n)+1) = O(n * log_2(n))$$

2 Implementácia

K implementácii ako to vyžaduje zadanie bol použitý programovací jazyk C++ a knižnica openMPI. Samotná implementácia radiaceho algoritmu je rozdelená do dvoch vetiev.

Prvú vetvu vykonáva prvý procesor, ktorého úlohou je načítať vstup, skontrolovať počet vstupných čísel a posielať načítané čísla ďalšiemu procesoru funkciou MPI_Send(). Procesor načíta poomocou fstream čísla zo vstupného súboru a posiela čísla ďalšiemu procesoru. Posielať začína hneď po načítaní prvého čísla. Linku, na ktorú to ma poslať číslo rozoznáva podľa hodnoty TAG(tagy 0 a 1) vo funkcii MPI_Send(). Po skončení posielania čísiel posiela TAG s hodnotou 3, ktorým dáva ďalšiemu procesoru najavo, že skončíl s posielaním čísiel.

Druhá vetva obsahuje implementáciu pre zvýšne procesory. Každý procesor obsahuje dve linky, ktoré sú implementované ako fronty dátovým typom std::queue. Hlavným telom tejto vetvy je cyklus, v ktorom prebieha prijímanie čísiel a ukladanie do jednotlívých front podľa hodnoty prijatého TAG funkciou MPI_Recv(). Ak príjme TAG s hodnotou 3 končí príjamnie čísel. Samotné zoraďovanie začína ak je vo fronte q0 2^{i-1} čísel a vo fronte q1 jedno číslo. Zvyšné čísla začnú postupne pribúdať. Procesor porovná čísla na začiatku oboch front, vyberie menšie z nich a pošle ho ďalšiemu procesoru. Výnimkou je posledný procesor, ktorý neposiela čísla ďalej, ale vypisuje ich na štandardný výstup.



Obr. 2: Sekvenčný diagram správ

3 Experimenty

Experimentovanie prebehlo nad náhodne zoradenou postupnosťou, postupnosťou zoradenou od najmenšieho po najväčšie číslo a opačne zoradenou postupnosťou . Pre meranie časových údajov bola použitá C++ knižnica Chrono, meranie času bolo zastavené v dobe, kedy prišla procesoru P_0 správa od procesoru P_n, keď skončil so spracovaním posledného čísla. V tabuľke 3 sú uvedené výsledky merania nad postupnosťami.

náhodne zoradená postupnosť	333ms
od najmenšieho po najväčší	334ms
od najväčšieho po najmenší	334ms

Tabuľka 1: Namerané hodnoty

4 Záver

V rámci prjektu bol implementovaný algoritmus Pipeline Merge Sort v jazyku C++ s použitím knižnice openMPI. V sekcii 1 je popísaný samotný algoritmus a jeho odvodená časová a pamäťová zložitosť. V sekcii 2 je popísaná samotná implementácia algoritmu, popísaný spôsob komunikácie procesov použitím sekvenčného diagramu. Vzhľadom na pevne daný počet vstupných čisel pre zoradenie bolo vykonané experimentovanie s rôzným zoradením vstupnej postupnosti, ako je popísané v sekcii 3.Výsledky experimentov ukazujú, že nech je vstupná postupnosť akokoľvek zoradená, nemá to vplyv na dobu behu algoritmu.