Viditelnost

Peter Grofčík
xgrofc00

April 19, 2020

1 Princíp úlohy

Samotné riešenie úlohy spočíva v štyroch častiach:

1. Každý z procesorov obdrží tri hodnoty zo vstupnej postupnosti (dve korešpondujúce pre procesor a hodnotu pozorovateľa). Pre obe svoje hodnoty vyráta pozorovací úhol oproti pozorovateľovi, s ktorými pracuje vo zvyšku úlohy.

2. UpSweep algoritmus, ktorý simuluje prechod po úrovniach hĺbky stromu(od listov k rootovi). Hĺbka stromu je vždy logaritmus o základe dva z počtu vstupných čísel. Pre výpočet maxima na každom uzle úrovne sú potrebné dve hodnoty, ktorých maximum sa vždy zapíše na miesto pravého z dvojice procesorov. Toto je zariadené poslaním hodnoty z ľavého procesora pravému. Každou úrovňou tak klesá počet využitých procesorov o polovicu, keďže v ďalšej úrovni pracujú iba pravé procesory z predchádzajúcej úrovne opäť vo dvojiciach ľavá pravá. Index ľavého a pravého procesora závysí od kroku (resp. úrovne stromu), kde krok odpovedá hodnote $2^{úroveň+1}$. Index ľavého a pravého procesora potom odpovedá hodnete $2^{úroveň}$ a $2^{úroveň+1}$, avšak ku každému indexu je potrebné pridať hodnotu, ktorá odpovedá násobkom kroku úrovne od nuly po celkový počet vstupných čísel, čím je zariadené, že na každej úrovni dôjde k výpočtu medzi všetkými dvojicami aktívnych procesorov. Keďže každá úroveň obsahuje ľavé procesoy, ktoré len odošlú svoju hodnotu pre výpočet maxima na pravom procesore, tak odosielanie medzi procesormy začína až od úrovne 1, keďže na úrovni 0 (listová) si každý procesor vyráta lokálne maximum z dvoch obdržaných čísel z kroku 1.

3. DownSweep algoritmus, ktorý simuluje prechod po úrovniach hĺbky stromu(od roota k listom). Prechod je simulovaný nad rovnakým stromom ako pri UpSweep algoritme, avšak opačným smerom (zhora dole). Root (resp. posledný z procesorov) zmení hodnotu svojho druhého čísla (resp. posledného čísla v postupnosti) číslo 0. Opať pomocou kroku aktuálnej úrovne dochádza k výberu dvojíc procesorov, ktoré si však na danej úrovni pošlú hodnoty. Ľavým potomkom nasledujúcej úrovne sa stáva procesor s id na hodnote $2^{(úroveň-1)} - 1$ (samozrejme opäť s pridanou hodnotou násobku kroku úrovne pre pokrytie všetkých ľavých potomkov) a pravým je samotný root. Ľavý potomok pramo prepíše obdržanú hodnotu z roota za svoju, ale pred tým pošle svoju aktuálnu hodnotu späť rootovi. Ten vytvorí maximum zo svojej a obdržanej hodnoty a zapíše ju na miesto svojho čísla, keďže sám predstavuje pravého potomka v strome. Pri tomto prechode sa veľkosť kroku naopak zmenšuje a teda špeciálnym prípadom je posledná úroveň kedy každý procesor zapíše hodnotu pravého prvku do ľavého a zároveň vytvorí maximum z ľavého a pravého prvku na indexe pravého. Obe hodnoty však na sebe závysia od aktuálnej, preto je potrebné si pamätať hodnotu indexu 0 v pomocnej premennej.

4. Posledná časť spočíva v paralelnom porovnaní kedy každý procesor porovná hodnotu uhla paprsku voči pozorovateľovi z prvého kroku voči vypočítanému maximu pre každý index (každý procesor pre obe svoje hodnoty).

2 Analýza úlohy

1. Master procesor rozpošle každému procesoru dve čísla, čo sa dá vykonať v konštantnom čase, a teda zložitosť je možné zapísať ako *O(1)*.

2. UpSweep algoritmus je vykonávaný iteratívne s počtom iterácií $log_2(n)$ pričom v každej iterácií dochádza k paralelnému výpočtu na všetkých aktívnych procesoroch, avšak na najnižšej úrovni môže z dôsledku nedostatku procesorov dôjsť sekvenčnému riešeniu úlohy, čo znamená časovú zložitosť $O(n/N + log\ N)$.

3. DownSweep algoritmus je rovnako vykonávaný iteratívne celou výškou stromu, čo opäť znamená časovú zložitosť rovnú $O(n/N + log\ N)$.

4. V poslednej fáze každý procesor porovná veľkosť uhla s vypočítaným maximom a odošle ho masterovi, ktorý ich v cykle ukladá, a teda zložitosť je možné zapísať ako *O(n)*.

Zistená časová zložitosť je teda $O(n/N + log\ N)$. Z toho môžme odvodiť cenu paralelného riešenia $c(n) = p(n) * t(n) = O(n + N * log\ N)$, čo v prípade že hodnota $log\ N$ je menšia ako hodnota n/N znamená optimálnu cenu $c(n) = O(n)$, avšak za cenu lineárnej časovej zložitosti $O(n/N)$. Pri implementácii som tak zvolil variantu kedy k sekvenčnej časti nedochádza a časová zložitosť je tak logaritmická $O(log\ n)$, pričom potrebný počet procesorov je $p(n) = n/2$, čo však znamená neoptimálnu cenu riešenia $c(n) = O(n * log\ n)$.

3 Implementácia

Algoritmus je implementovaný v jazyku **C++** za pomoci knižnice **Open MPI**. Počet procesorov potrebných pre výpočet je vypočítaný z počtu vstupných čísel tak aby aby jeho hodnota odpovedala $2^{(výška\ stromu)}/2$, kde výška stromu je rovná $log_2(n)$ zaokrúhlenému na celé číslo nahor, čo zabezpečí potrebný počet procesorov pre vyvážený strom o výške $log_2(n)$.

Na začiatku dôjde k inicializácii MPI prostredia pomocou funkcie MPI_Init. Nasleduje volanie funkcií *init* (iba Master) a *initRec* (zvyšné procesory). Tieto volania tak isto nespadajú od meranú časť, keďže v nich dochádza k distribúcii jednotlivých hodnôt pre procesory. Každý z procesorov obdrží tri hodnoty (hodnota pozorovateľa a dve prislúchajúce hodnoty zo vstupného reťazca).

V momente kedy už každý z procesorov obdržal svoje hodnoty a nachádza sa na mieste za volaním prislúchajúcej funkcie pre inicializáciu (docielené pomocou funkcie **MPI_Barrier**) dôjde k zapísaniu počiatočného času a začína výpočet samotného algoritmu zavolaním funkcie *initAngleCompute*, v ktorej každý z procesorov paralelne vypočíta uhol oboch prislúchajúcich hodnôt v závyslosti na hodnote pozorovateľa.

Nasleduje volanie funkcie *sweepU* každým procesorom zvlášť. Vo funkcii každý z procesorov vykoná hlavný cyklus po celej výške simulovaného stromu, pričom na každej úrovni procesor pomocou cyklu cez indexy jednotlivých hodnôt a porovnaním svojho id voči aktívnym id na danej úrovni overí, či sa z neho pre danú úroveň stáva aktívny procesor (resp. či daný procesor vôbec bude posielať alebo príjmať dáta na danej úrovni). Touto časťou je simulovaný *SweepUP* algoritmus, ktorý je súčasťou metódy *max_prescan*.

Ďalšia časť je volanie funkcie *sweepDown* opäť každým procesorom. Táto časť opäť prechádza simulovaným stromom (tento krát však zhora dole). V tejto časti vždy ľavý potomok (resp. ľavý z rootov nasledujúcej úrovne) obdrží hodnotu od roota aktuálnej úrovne a zapíše si ju namiesto svojej vlastnej. Ešte predtým ako prepíše svoju aktuálnu hodnotu tak ju pošle rootovi, ktorý vypočíta maximum zo svojej a obdržanej hodnoty. Toto maximum následne zapíše za svoju hodnotu, keďže v nasledujúcej simulovanej úrovni predstavuje zároveň pravého potomka.

V oboch spomenutých častiach nastáva špeciálny prípad (na 0. úrovni simulovaného stromu), kedy každý z procesorov nekomunikuje s ostatnými ale vykoná rovnaké úkony medzi svojimi dvoma hodnotami (index 0 a 1 resp. ľavá a pravá hodnota). Jedná sa o najnižšiu listovú úroveň, kde tieto hodnoty predstavujú ľavého a pravého potomka simulovaného stromu, ale keďže na výpočet je vždy potrebný práve jeden procesor a druhý by tak len odoslal a prijal hodnotu tak nie je nutné použiť rovnaký počet procesorov ako je počet prvkov, ale stačí použiť počet procesorov odpovedajúci počtu uzlov nad listami.

V poslednej paralelnej časti už len každý z procesorov porovná svoj vypočítaný uhol voči maximálnemu vypočítanému uhlu pre obe svoje prislúchajúce hodnoty. V momente kedy každý z procesorov ukončil svoju prácu po toto miesto (opäť **MPI_Barrier**) dôjde k zapísaniu konca času simulácie a následný zber hodnôt Masterom tak už nepodlieha meraniu času vykonávanej úlohy.

4 Experimenty

Cieľom experimentov bolo overiť časovú zložitosť algoritmu. Pre merania som volil navyšovanie vstupných prvkov vždy o dva a na každý vstup som previedol sto meraní, ktoré som následne zpriemeroval. Merania je možné spustiť zámenou vstupného čísla za argument **"EXP"**. Na grafe sú vyobrazené výsledné hodnoty meraní po násobkoch dvojky, keďže hodnoty medzi násobkami sú merané na rovnakej výške stromu a vytvárali mierne nepresné výsledky.

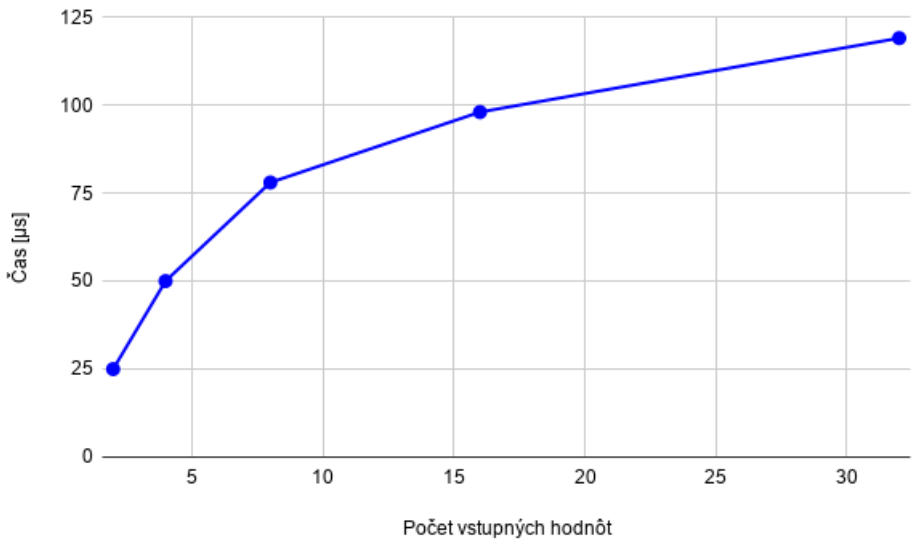

Figure 1: Meranie času experimentov

5 Komunikačný protokol

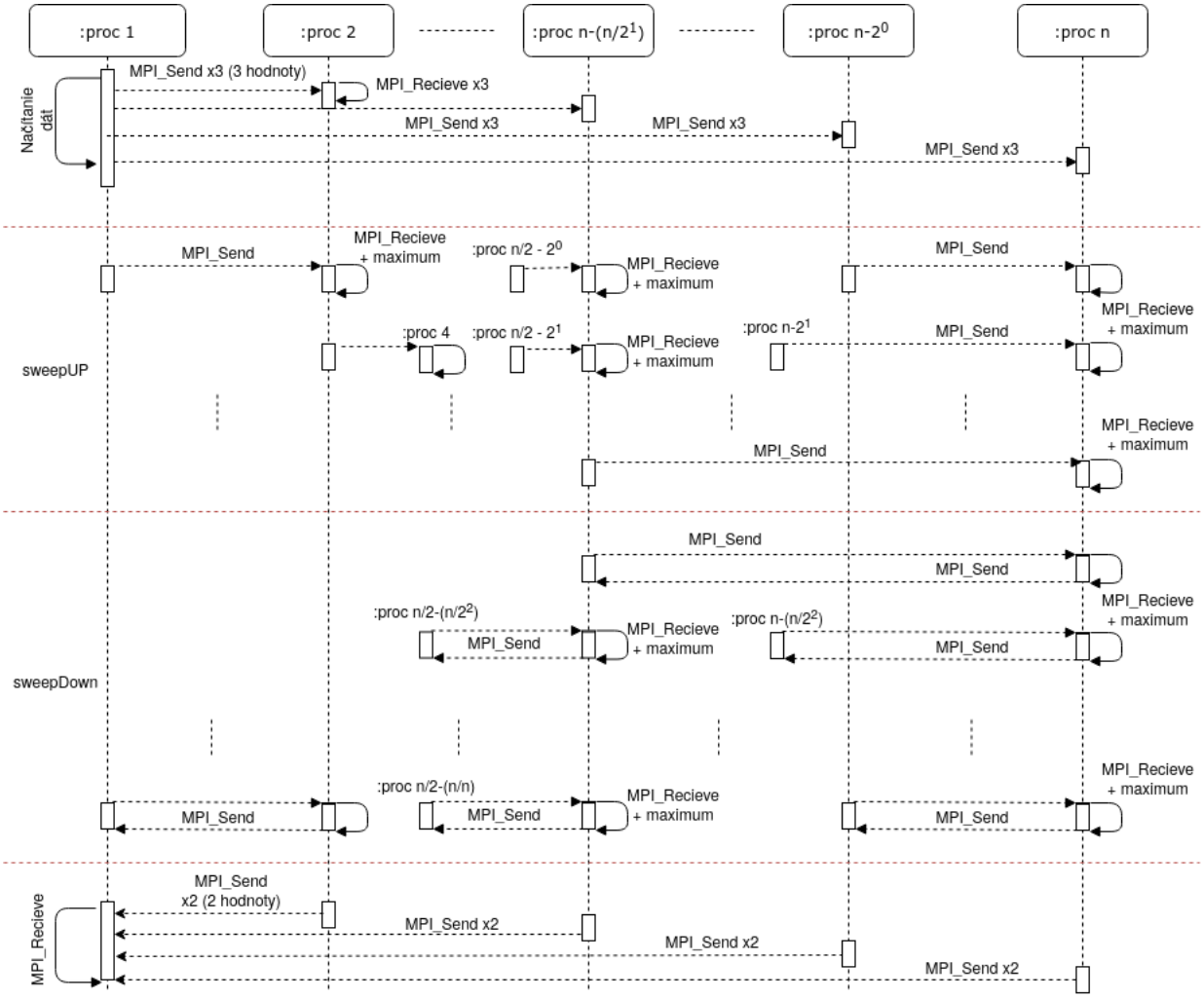

Figure 2: Sekvenčný graf komunikácie procesorov

6 Záver

Výsledky meraní vyobrazené v grafe odpovedajú tvrdeniu o logaritmickom raste času pri rastúcom počte vstupných prvkov, čo potvrdzuje zistenú logaritmickú zložitosť algoritmu. Keďže jednotlivé úrovne stromu sú vykonávané paralelne tak najrelevantnejšie výsledky merania sú výsledky pre počty prvkov rovné násobkom dvojky keďže len po každom násobku dochádza k zväčšeniu výšky stromu.