Viditeľnosť
Dokumentácia k 3. projektu

Peter Lukáč - xlukac11

Apríl 2020

1 Analýza úlohy "Viditeľnosť"

Úloha viditeľnosti spočíva vo výpočte viditeľnosti bodu určitej výšky vo vektore výšok podľa lúča z počiatočného bodu(pozorovateľa). Vstupom je vektor viditeľnosti, kde prvý bod je pozorovateľ. Algoritmus pracuje v nasledujúich krokoch:

1. Výpočet vektorov výšiek pozdĺž paprsku.
2. Vektorov výšiek sa prepočitá na vektorov uhlov.
3. Pomocou `max_prescan` sa vypočíta vektor maximálnych uhlov
4. Bod je viditeľný ak je jeho uhol vačší ako maximálny uhol.

1.1 Teoretická zložitosť

Vstupom je vektor výšok dĺžky n. N procesorov si rovomerne rozdelí vstup a budú paralelne vykonávať uvedené kroky na EREW PRAM:

1. Výpočet vektorov výšiek: $O(n/N)$
2. Výpočet vektorov uhlov: $O(n/N)$
3. `max_prescan` pozostáva z UpSweep a DownSweep. UpSweep je redukcia kde $n > N$. Teda každý procesor najpr sekvenčne vypočíta svoj úsek($O(n/N)$) a potom sa použije strom procesorov($O(\log N)$): $O(n/N + \log N)$. DownSweep najprv použije strom procesorov($O(\log N)$) a potom každý procesor sekvenčne dopočíta svoj úsek($O(n/N)$): $O(n/N + \log N)$. Celkom: $O(n/N + \log N) + O(n/N + \log N) = O(n/N + \log N)$.
4. Výpočet či je bod vyditeľný: $O(n/N)$.

Celkovo časová zložitosť: $t(n) = O(n/N) + O(n/N) + O(n/N + \log N) + O(n/N) = O(n/N + \log N)$. `max_prescan` pridáva celkovej zložitosi parameter $\log N$. Avšak pre $\log N < n/N$ je čas sekvenčnej časťi vačší ako časť so stromom procesorov a časová zložitosť je $O(n/N)$. Potom aj úloha viditeľnosti bude mať zložitosť $O(n/N)$ ak $\log N < n/N$.

Sekvenčná verzia algoritmu má lineárnu časovú zložitosť: $O(n)$. Cena paralelného algoritmu je $O(n/N + \log N) * N = O(n + N \log N)$ čo je viac ako sekvenčná verzia takže paralelný algoritmus nieje oprimálny. Avšak ak $n/N > \log N$ tak časová zložitosť je $O(n/N)$ a celková cena je $O(n/N) * N = O(n)$ čo už je optimálne.

Priestorová zložitosť je lineárna: $O(n)$.

2 Implementácia

Algoritmus je implementovaný v jazyku C++ a používa knižnicu `Open MPI` pre komunikáciu procesov.

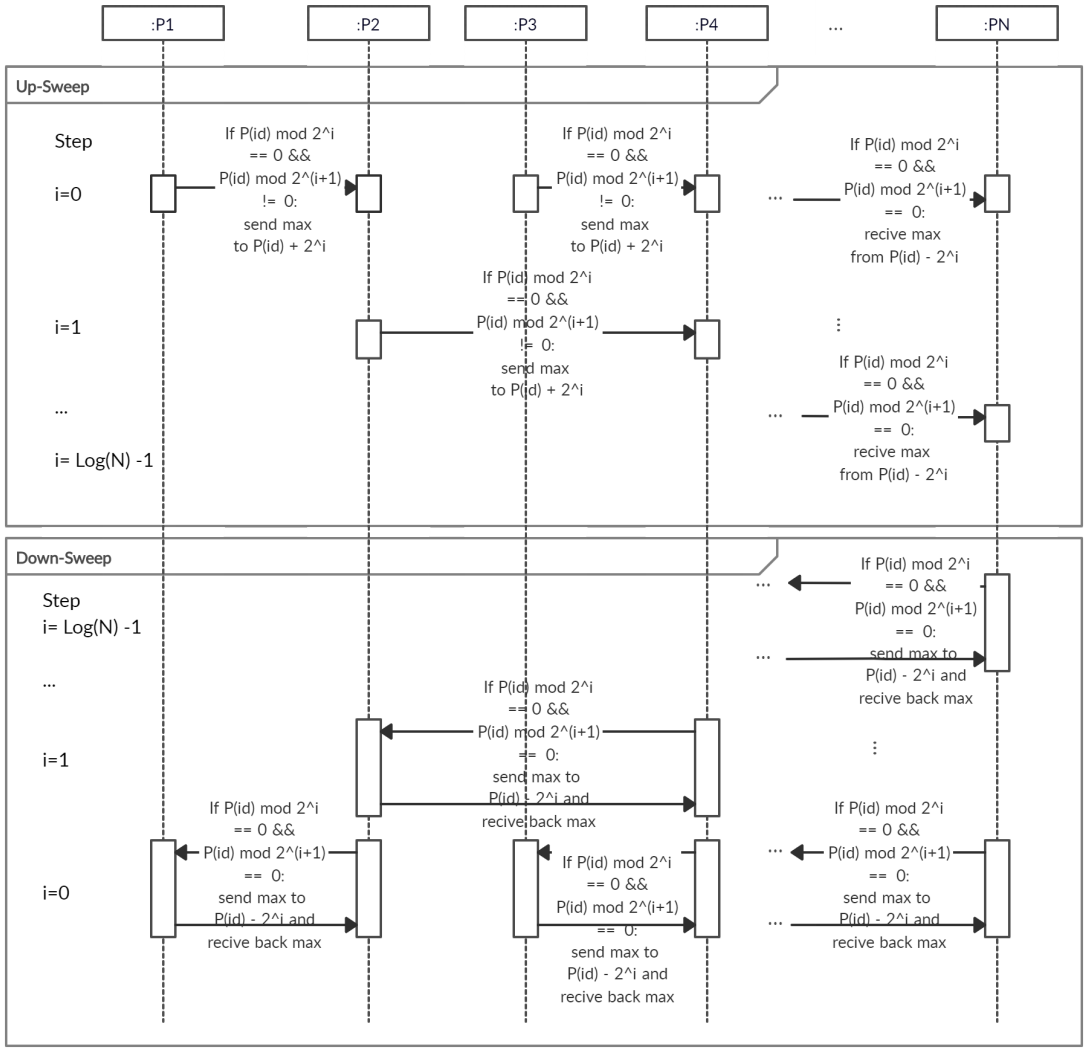
Podľa popisu, algoritmus používa N procesorov. Pre praktickú implementáciu toto napodobňujeme požitím systémových procesov.

Každý proces pracuje na svojom pridelej skupine vstupných hodnôt veľkosti n/N. Procesy komunikujú interprocesovou komunikáciou cez rozhranie `Open MPI`.

Významná časť algoritmu je časť `max_prescan`, ktorá vypočitá prefix maxím. Je použitá verzia pre počet procesorov vačší ako dĺžka vstupu, takže každý procesor musí sekvenčne vypočiať svoju časť n/N pred tým ako sa použije strom procesov. Každý uzol stromu predstavuje jeden proces a každý proces si drží svoju hodnotu. Zasielanie a prijímanie hodnôt je realizované interprocesovou komunikáciou. Pre optimálne využitie binárneho stromu procesov používame $N = \{2^n | n \in \mathbb{N}\}$ procesov. Testovací skript `test.sh` spúšťa program s maximálnym počtom procesov z týchto hodnôt, pričom berie ohľad aby bolo $\log N < n/N$.

3 Komunikačný protokol

Komunkačný protokol zobecnený pre n-procesov pre funkciu `max_prescan`.

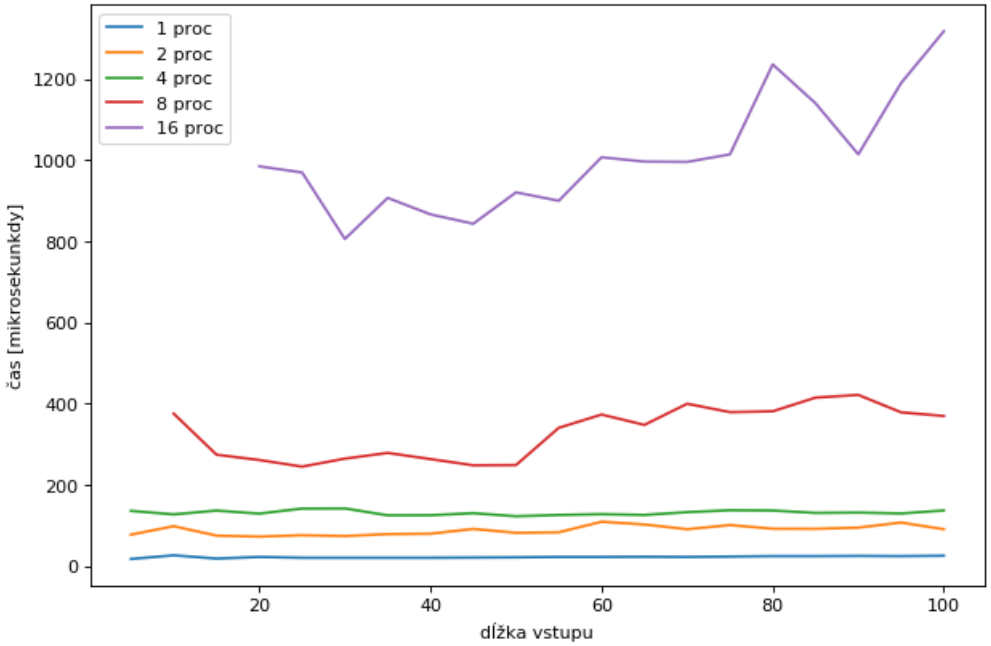


Protokol ukazuje, že `max_prescan` pracuje v cykloch, kde rozhodnutie, či a komu proces pošle alebo prijme data závisí na pozící procesu v strome a čísle iterácie. Každý proces do funkcie vstupuje len so svojou maximálnou hodnotou, svojou pozíciou, počtom procesov a funkcia mu vráti maximálny prefix.

4 Experimenty a výsledky

Program bol testovaný a meraný na servery `merlin`. Testované boli vstupy maximálnej dĺžky 100 na počte procesov 1,2,4,8,16.

Na meranie času bola použitá štandartná časová knižnica `chrono`. Časové známky sú uložené len v okolí samotného výpočtu algoritmu podľa jeho popisu. Každý proces na chybový výstup vypíše čas svojho behu. Celkový čas jedného behu sa berie ako priemer behu všetkých procesov. Pre každú dĺžku vstupu bol program spúšťaný 50-krát a dĺžka behu pre danú dĺžku vstupu sa berie ako medián všetkých meraní.



Obr. 1: Výsledky merania

5 Záver

Podľa nameraných výsledkov časová zložitosť programu neodpovedá teoretickej zložitosti. So zvyšujúcim počtom procesov výrazne stúpa čas, čo by podľa teórie malo byť naopak. Pri behu s jedným procesom vidíme, že program mal lineárnu časovú zložitosť v závislosti na dĺžke vstupu($\sim 16\mu s$ pre 4 vstupné hodnoty - $\sim 26\mu s$ pre 100 vstupných hodnôt) ako bolo očakávané. Bolo experimetované so spúšťaním s viacerími procesmi bez použitia funkcie `max_prescan`, čo malo dopad, že výsledky viditeľnosti neboli korektné. Tento experiment mal ale za účel zisiť réžiu `max_prescan`. V tomto experimente sme s vyšími počtami procesov sme dostávali lepšie výsledky. V praktickej implementácií je teda dôvodom zhoršenia časovej zložitosti réžie spojená s iterprocesovou komunikáciou.