Dokumentácia projektu do predmentu Paralelné a distribuované algoritmy: Algoritmus výpočtu úrovne vrcholov

Juraj Ondrej Dúbrava (xdubra03)

25. apríla 2019

1 Úvod

Cieľom projektu bolo implementovať v jazyku C/C++ paralelný algortimus na určenie úrovne vrcholov v binárnom strome a riadiaci skript na riadenie výpočtu.

2 Rozbor a analýza algoritmu

Algoritmus výpočtu úrovne vrcholov sa skladá z niekoľkých častí, z ktorých medzi najdôležitejšie patria výpočet tzv. Eulerovej cesty a paralelná suma sufixov.

Eulerova cesta je tvorená hranami v orientovanom grafe, kde sa nachádzajú dopredné a spätné hrany. Pre každú hranu je v Eulerovej ceste definovaná nasledujúca hrana v ceste. Každá hrana sa nachádza v ceste práve raz. Algoritmus nepoužíva koreň, ak sa zavedie, dostávame prechod depth-first search. Vstupom algoritmu je tzv. pole susednosi, ktorým je graf reprezentovaný a výstupom algoritmu je pole o veľkosti $2 * n - 2$, kde n je počet vrcholov v grafe, pre každú hranu je jedna položka.

Paralelná suma sufixov je algoritmus na výpočet súčtu sumy sufixov v zozname. Tento algoritmus uvažuje používanie zdieľanej pamäte. Vstupom je pole hodnôt a binárny asociatívny operátor. Postupne sa prechádza od aktuálneho prvku až na koniec zoznamu, v tejto fáze sa postupne vypočítava výsledná suma. Keď nejaký prvok dosiahol koniec, iné prvky koniec dosiahnuť ešte nemuseli, tým pádom by sa k výsledku niekoľkokrát pričítala hodnota posledného prvku. Preto sa posledný prvok nastavuje na hodnotu neutrálneho prvku pre binárnu operáciu. Asymptotická časová zložitosť tohto algoritmu je $O(log\, n)$ a cena je $O(n.log\, n)$.

Samotný algoritmus výpočtu úrovne vrcholov pozostáva z niekoľkých krokov vrátane vyššie uvedených. Algoritmus uvažuje použitie zdieľanej pamäte. Prvým krokom algoritmu je nastavenie váh pre doprené a spätné hrany, dopredné majú váhu -1 a spätné 1. Následne nasleduje fáza výpočtu sumy sufixov v poli váh s použitím Eulerovej cesty na určenie následníka. Posledným krokom algoritmu je úprava, po ktorej dostaneme výslednú úroveň. Pre všetky dopredné hrany $e = (u, v)$ sa vykoná úprava taká, že úroveň vrcholu v sa rovná váhe danej hrany plus 1.

Čo sa týka asymptotickej časovej zložitosti, tá je $O(log\, n)$, pričom táto zložitosť je výsledkom zložitostí jednotlivých krokov pri algoritme, konktrétne zložitosti výpočtu Eulerovej cesty, ktorá je $O(c)$, inicializácie váh so zložitosťou $O(c)$, výpočte sumy sufixov so zložitosťou $O(log\, n)$ a korekcie výsledku so zložitosťou $O(c)$. Cena algoritmu závisí na implementácii sumy sufixov [1].

3 Implementácia algoritmu

Algoritmus bol implementovaný v jazyku C++ pomocou knižnice na paralelné výpočty *openMPI*. Pred začiatkom výpočtu sa v skripte *test* vypočíta potrebný počet procesorov podľa vzorca $p = 2 * n - 2$, kde n je počet vrcholov. Dostávame teda 1 procesor pre každú hranu, pre dopredné aj spätné.

Na začiatku programu sa podľa argumentu, ktorý tvorí reťazec znakov reprezentujúci vrcholy binárneho stromu, vytvorí pole znakov *nodes_array*, kde sú uložené mená vrcholov. Pred algoritmom výpočtu úrovne vrcholov je potrebné vytvoriť pre potreby algoritmu pole susednosti a následne ho použiť pri tvorbe Eulerovej cesty. Pole susednosti

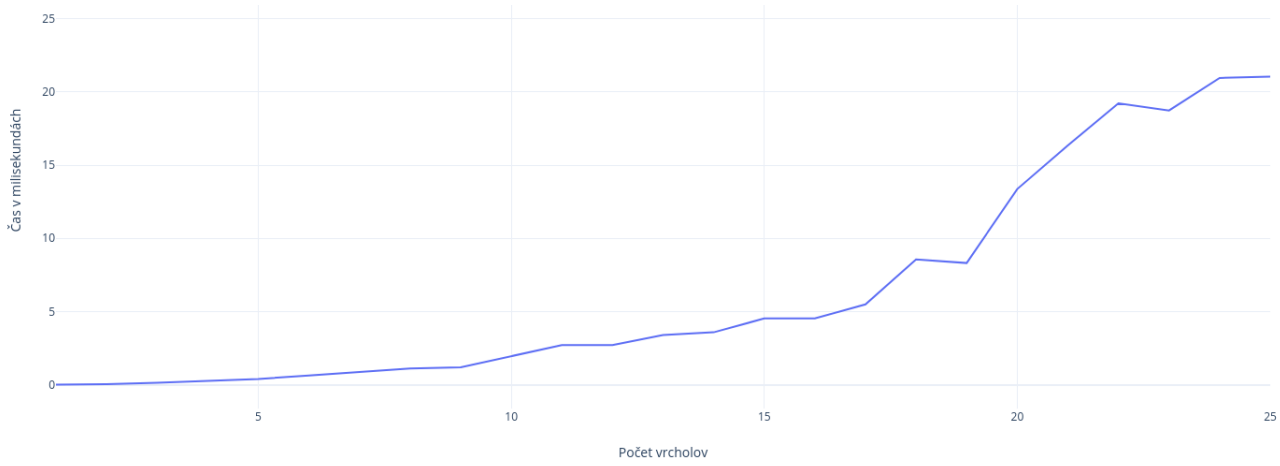
nodes_neighbours je realizované ako 2D pole, kde sú pre každý uzol susedné uzly. Naplnenie poľa pre každý procesor sa realizuje vo funkcii *get_neighbours*. Následne pokračuje fáza tvorby Eulerovej cesty. Aby bolo možné rozlíšiť dopredné a spätné hrany, tak bolo zvolené, že prvá polovica procesorov sa stará o dopredné hrany a druhá polovica o spätné hrany, hrany sú číslované od 0. Potom si každý procesor zistí číslo počiatočného a koncového uzlu, ktoré spája daná hrana. Pomocou funkcie *get_position* sa zistí, či v poli susedov existuje pre cieľový uzol následník. Ak áno, na pozícií za cieľovým uzlom je číslo následníka a hodnota v poli *etour_array* sa nastaví na hodnotu následníka. Ak je na pozícii za uzlom -1, následník neexistuje a hodnota v poli *etour_array* sa získa ako hodnota cieľového uzlu plus počet hrán. Po tejto fáze má každý procesor v poli *etour_array* na indexe svojho ranku hodnotu nasledujúcej hrany v Eulerovej ceste. Cestu je potrebné zakončiť, čo sa dosahuje tým, že posledná hrana má za následníka samú seba.

Po výpočte Eulerovej cesty nasleduje samotný algoritmus výpočtu úrovne vrcholov. Jeho prvým krokom je nastavenie váh pre dopredné a spätné hrany. Váhy sa nachádzajú v poli *weights_array*. Dopredným hranám je daná váha -1, spätným 1. Keďže tento algoritmus predpokladá použitie zdieľanej pamäti a aby každý procesor vedel aj o váhach ostatných uzlov bez použitia zdieľanej pamäti, pomocou funkcie *MPI_Allgather* je zdieľaná pamäť simulovaná. Každý procesor tým to spôsobom odošle svoju hodnotu ostatným procesorom. Nasleduje krok výpočtu sumy sufixov. Tento algoritmus opäť predpokladá použitie zdieľanej pamäte, takže na jeho realizáciu bola použitá funkcia *MPI_Allgather*. Vstupom je pole hodnôt *val_array* (kópia poľa váh) a pole reprezentujúce Eulerovu cestu. Na indexe, kde v poli Eulerovej cesty bola hodnota rovná indexu, sa v poli *val_array* zapíše hodnota 0. Nastavená 0 je hodnotou pre poslednú hranu v Eulerovej ceste, keďže ukazuje sama na seba. Výpočet prebieha v cykle od 1 do hodnoty logaritmu počtu hrán. Počíta sa nová hodnota v Eulerovej ceste a v poli *val_array*. Po vypočítaní nových hodnôt každý procesor pomocou *MPI_Allgather* rozošle nové hodnoty ostatným procesorom na simuláciu zdieľanej pamäte. Po výpočte je potrebné uskutočniť úpravu, pretože posledná hodnota mohla byť niekoľkokrát pričítaná k sume. Podľa algoritmu je potrebné overiť, či sa hodnota pre poslednú hranu v pôvodnom poli váh rovnala hodnote 0, ak nie, táto hodnota je pričítaná k výsledku. Posledný krokom pri výpočte úrovne vrcholov je nastavenie hodnoty úrovne pre každý vrchol. Úroveň počítajú len procesory starajúce sa o dopredné hrany. Hodnota úrovne je vypočítaná ako hodnota v poli *val_array*, kde sú hodnoty pre jednotlivé sumy sufixov, zväčšená o 1. Táto hodnota je opäť distribuovaná všetkým procesorom, úrovne sa ukladajú do poľa *final_nodes_levels*. Ako posledné procesor s rankom 0 prejde pole vrcholov a ku každému vypíše hodnotu z poľa *final_nodes_levels*.

4 Experimenty

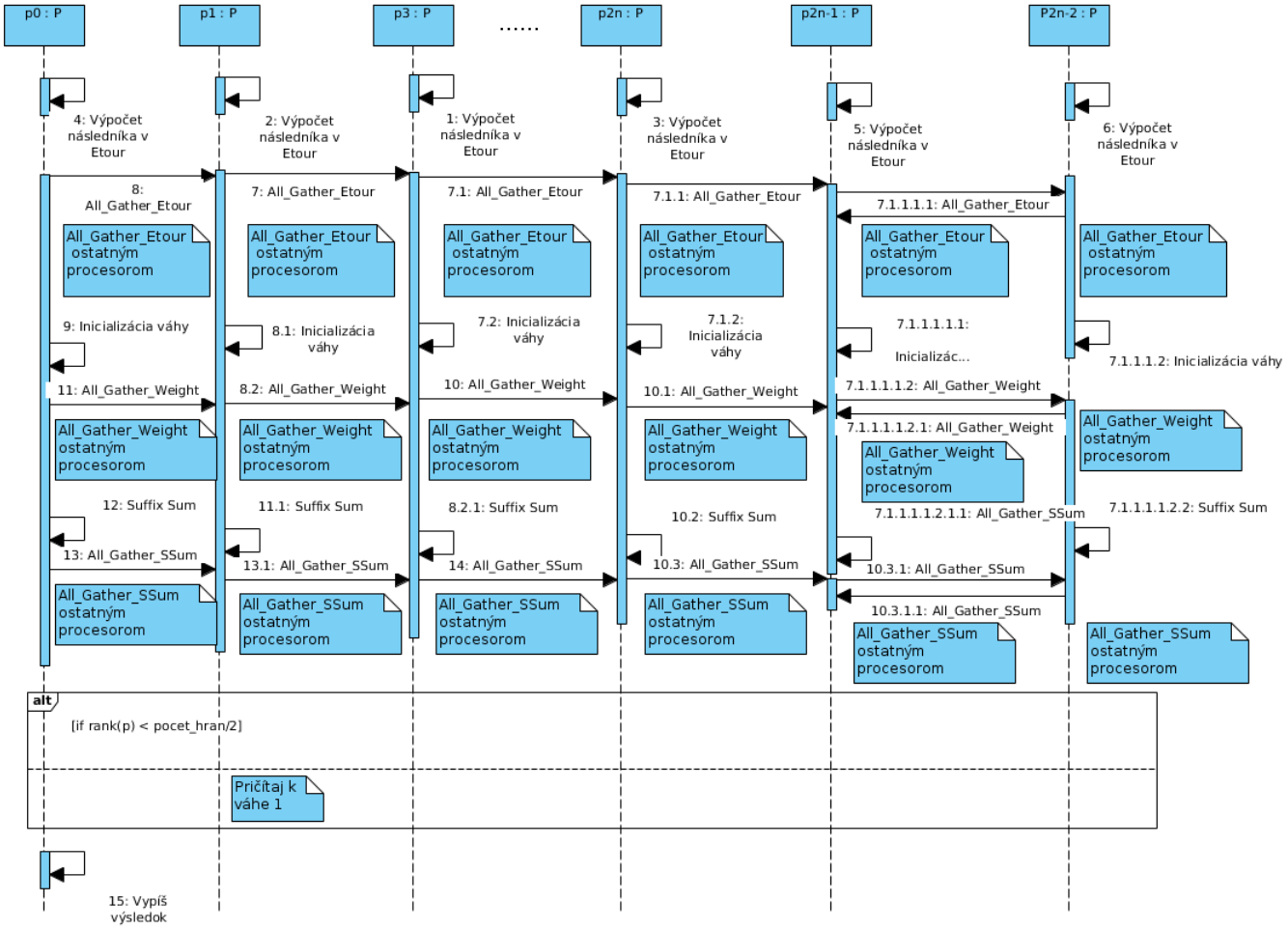
Algoritmus bol testovaný na overenie časovej zložitosti na vstupoch o veľkosti od 1 do 25 vrcholov s krokom 1. Pre každú hodnotu počtu vrcholov bolo vykonaných 25 meraní, z ktorých bol následne vybraný medián ako výsledná hodnota. Meranie času behu algoritmu bolo realizované funkciou *MPI_Wtime()*, meranie začalo pred výpočtom Eulerovej cesty a skončilo pred výpisom.

Získané hodnoty merania sú zobrazené v grafe na obrázku 2.



Obr. 1: Graf merania pre jednotlivé počty vstupných prvkov.

5 Komunikačný protokol



Obr. 2: Sekvenčný diagram reprezentujúci vzájomnú komunikáciu medzi procesmi.

6 Záver

Algoritmus výpočtu úrovne vrcholov je paralelný algoritmus navrhnutý na archituktúru PRAM. Jeho teoretická časová zložitosť je $O(log\, n)$. Z grafu 2 je vidieť, že k tejto časovej zložitosti sa približujeme až od určitého počtu vrcholov. Logaritmická časová zložitosť nie je úplne dosiahnutá, veľkú úlohu určite zohráva simuláciu zdieľanej pamäti, nakoľko na jej používanie bol algoritmus navrhnutý. Ďalším faktorom ovplyvňujúcim výsledky je vyťaženie serveru Merlin, na ktorom bolo testovanie uskutočnené.

Referencie

[1] HANÁČEK, Petr: List,Tree, Contraction,List coloring, Ruling set [cit. 2019-04-18]. Dostupné z `http://www.fit.vutbr.cz/study/courses/PDA/private/www/h007.pdf`