Paralelné a distribuované algoritmy
Dokumentácia k projektu č. 3

Michal Gabonay (xgabon@stud.fit.vutbr.cz) 17.4.2018
Vysoké Učení Technické v Brne

1. Úvod

Tento dokument slúži ako dokumentácia k projektu do predmet Paralelné a distribuované algoritmy. Úlohou tohto projektu bolo implementovať paralelné priradenie poradia preorder vrcholom.

2. Algoritmus

Algoritmus na riešenie problému preorder prechodu stromu sa skladá z 3 časí, vytvorenie Eulerovej cesty, výpočet Suffix Sum a úprava výsledkov do požadovanej podoby, čiže preorder poradie vrcholov. Jeden procesor sa stará o jednu hranu, čiže $p(n) = 2n-2$.

2.1 Princíp

1. Vytvorenie Eulerovaj cesty (v mojom prípade aj pole predchodcov hrán).
2. Každej hrane sa priradí váha, ak je dopredná, nastaví sa na 1, v opačnom prípade na 0.
3. Vypočíta sa Suffix sum pre každú hranu na základe ich váh a Eulerovaj cesty
4. Vziať len dopredné hrany a ich suffix sum upraviť na poradie v preorder prechode

2.2 Analýza

Teoretická časová zložitosť a celková cena

a) Vytvorenie Eulerovej cesty - každý procesor samostatne paralelne - $O(1)$
b) Na základe Eulerovej cesty vyrátať a rozposlať predchodcu každej hrany $O(1)$
c) Vyrátanie hodnôt hrán pomocou alg. suffix sum – log(2n-2) potrebných zopakovaní - $O(\log n)$
d) Úpravu hodnôt hrán na poradie v preorder prechode – každý procesor samostatne – $O(1)$

Výsledná časová zložitosť je : $O(\log n)$
Celková cena algoritmu je $c(n) = t(n)*p(n) = O(\log n)*p = O(n.\log n)$. Optimálny sekvenčný algoritmus má zložitosť $O(n)$. Z toho vyplýva, že algoritmus nie je optimálny.

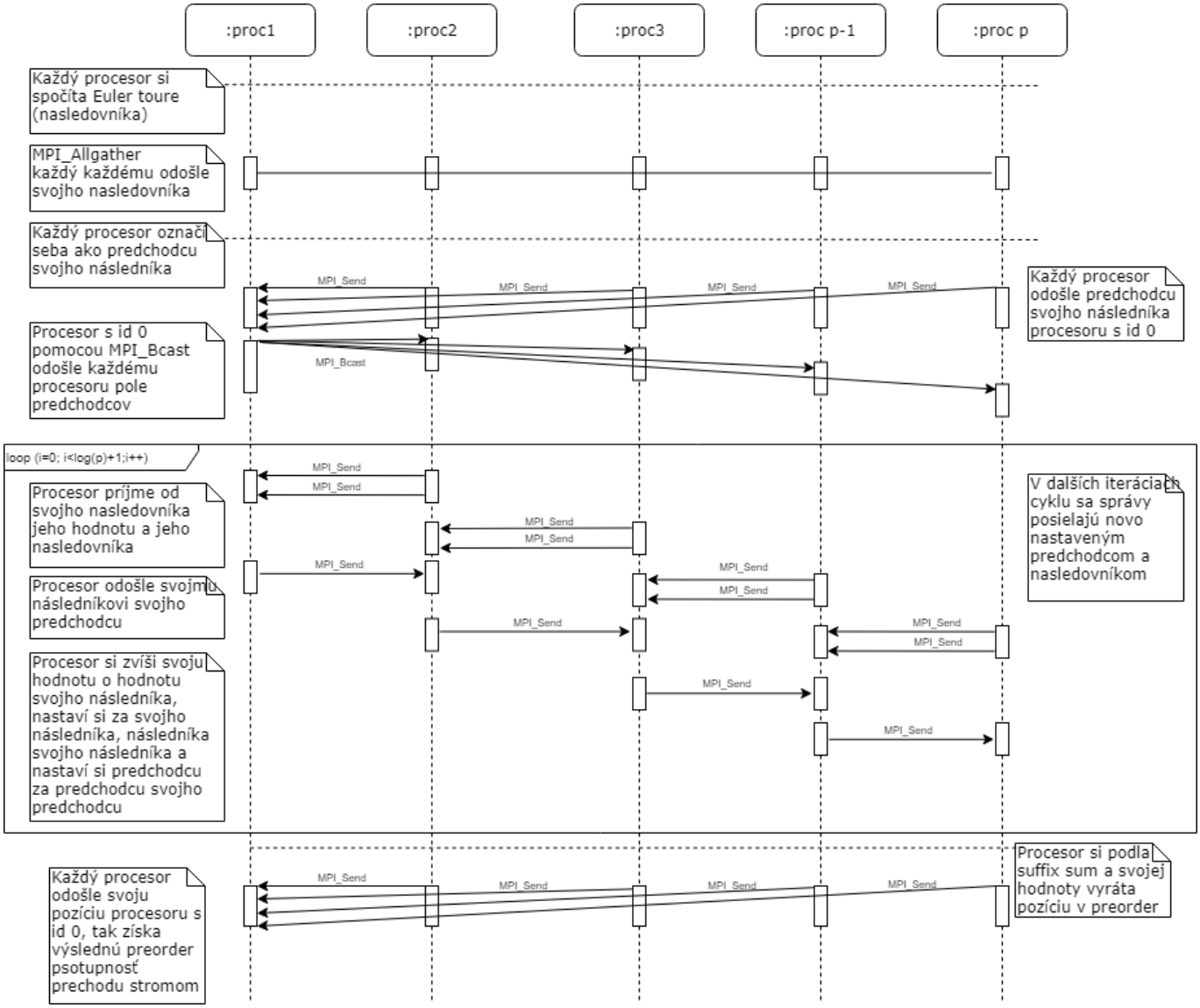
3. Implementácia

Aplikácie ja implementovaná v jazyku C++ s využitím knižnice Open MPI, ktorá umožňuje paralelné riešenie problémov. Ešte pred samotným algoritmu je potrebné si vytvoriť zoznam susedov (**adjacency list**) zo zadaných uzlov. Tento zoznam reprezentuje pomocná štruktúra pre popis zadaného binárneho stromu. Lineárnym prechodom cez všetky uzly si ukladám každú hranu do pomocnej štruktúry (dopredné aj spätné), ktorej si ukladám všetky potrebné informácie o danej hrane.

Každý procesor z adjacency listu spočíta svojho nasledovníka pomocou **Eulerovej cesty**. Hrana ktorá smeruje do koreňa, čiže posledná hrana si nastaví ako predchodcu -1 (reprezentujúca, že žiadneho následníka nemá). Výsledok Eulerovej cesty si procesory navzájom vymenia pomocou MPI_Allgather a uložia do poľa nasledovníkov. Následne sa z poľa nasledovníkom paralelne spočíta pole predchodcov tak, že všetky procesory odošlú svojho následníka procesoru s id 0. Ten uloží do poľa predchodcov s kľúčom následníka, id od ktorého túto informáciu získal. Sám sebe nastaví ako predchodcu -1, čo znamená, že žiadneho nemá.

Následne pomocou algoritmu **suffix sum** spočítame jednotlivé hodnoty hrán. Váha každej hrany je buď 1 (dopredná hrana) alebo 0 (spätná hrana). Obmedzenie na tento algoritmus je, že mohli byť použité len funkcie MPI_Send a MPI_Recv. Práve kvôli tomuto obmedzeniu bolo potrebné dopočítať pole predchodcov. V cykle od 0<k<log(2*n-2)+1 procesory pracujú v 4 štádiách. Ak nemá predchodcu, ani následníka (nastavené oba na -1), neposiela ani neprijíma nič. Ak nemá predchodcu, príjme od následníka jeho hodnotu a následníka jeho následníka a odošle mu naspäť svojho predchodcu (čiže informáciu, že žiadneho nemá), svoju hodnotu zvýši o hodnotu svojho následníka a nastaví si nového následníka. Ak procesor nemá následníka, tak len odošle svoju hodnotu a informáciu o tom, že nemá následníka a príjme informáciu od svojho predchodcu s predchodcom predchodcu a tohto predchodcu si nastaví za svojho. Ak procesor má následníka aj predchodcu tak odošle svojmu predchodcovi svoju hodnotu a svojho následníka, tieto rovnaké informácie príjme tiež od svojho následníka, naviac príjme od svojho predchodcu jeho predchodcu a odošle svojmu následníkovi informácie o svojom predchodcovi a všetky informácie si aktualizuje. Z tohto popisu to môže znieť dosť matúco, preto je táto komunikácie je lepšie znázornená na sekvenčnom diagrame na obrázku číslo 1.

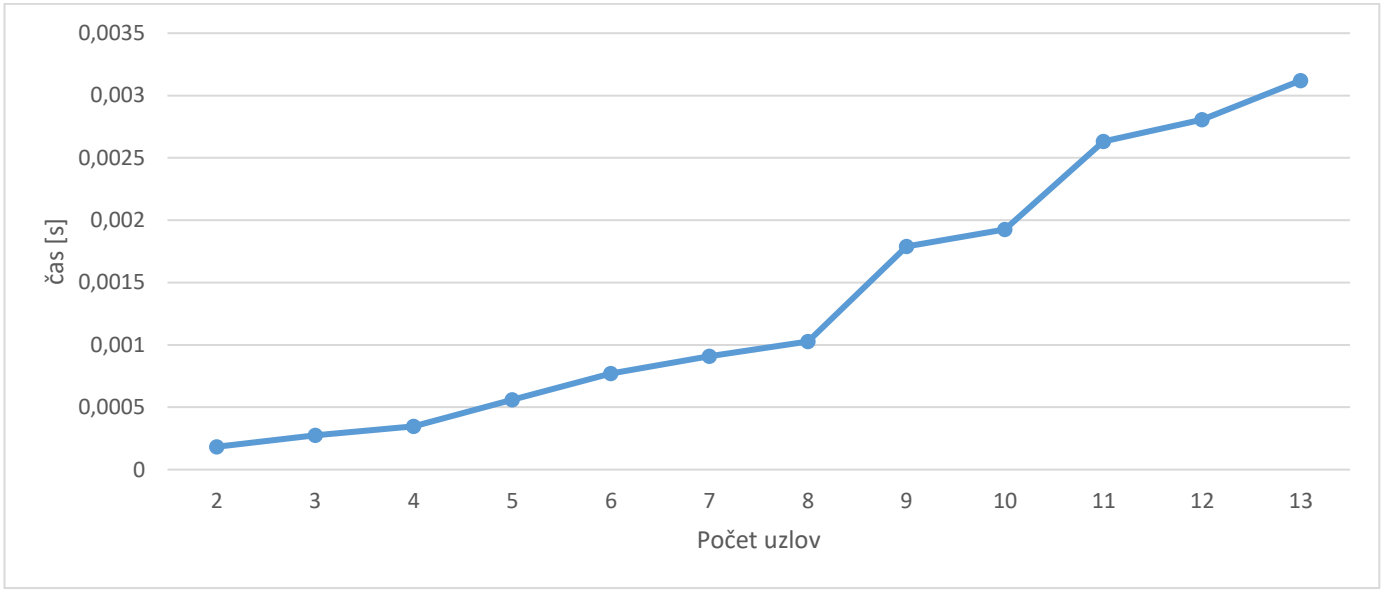
Následne každý procesor odošle svoju hodnotu získanú pomocou suffix sum procesoru s id 0. Ak je hrana, ktorú reprezentuje daný procesor dopredná, je uložená do výslednej preorder postupnosti.



Obrázok 1 - Sekvenčný diagram

4. Experimenty

S algoritmom som vykonal experimenty, ktoré sú zhrnuté do grafov na Obrázku 2. Vertikálna osa je čas v sekundách, horizontálna je počet uzlov v strome. Pre každú kombináciu počtu prvkov a počtu procesorov som aplikáciu spustil 5 krát a z výsledných časov spočítal priemer. Z experimentoch je vidno, že pri menších počtoch uzlov (do 8) je rast logaritmický, neskôr je trochu horší, čo môže byť dôvodom, že testovanie prebehlo na školskom servery merlin, ktorý má 12 procesorov a pre 8 a viac uzlov je treba väčší počet procesorov ako ich skutočne je, takže je čas spotrebovávaný aj na prepínanie kontextu medzi dostupnými procesormi. Taktiež treba brať v úvahu, že na testovanom servery nie som jediný, čiže sa o výpočtovú silu delím s ostatnými študentami a zamestnancami školy.



Obrázok 2 – Graf z experimentu

5. Záver

Algoritmus sa mi podarilo úspešne implementovať. Výsledky experimentovania boli zdokumentované a zhrnuté do tohto dokumentu. Podarila sa mi dosiahnuť požadovaná logaritmická časová zložitosť pre menšie počty vrcholov.