PRL project – Binary tree preorder Michael Halinár – xhalin01

Rozbor

Tento paralelný algoritmus potrebuje pri danom binárnom strome 2*n-2 procesorov, pričom n je počet uzlov binárneho stromu. Postup pre vytváranie prechodu je nasledovný. Ako prvé sa vytvoria zoznamy hrán podľa daného vstupu. Následne zoznam susednosti. V tomto zozname má koreň 2 susedov, uzoľ má 3 susedov a listy majú iba jedného suseda. Po vytvorení zoznamu susednosti nasleduje vytvorenie eulerovej cesty, kde každý procesor zistí svojho nasledníka. Potom prichádza na rad vytvorenie váh hrán, ak hrana ide z uzla s vyšším indexom do uzla s nižším indexom ide o hranu doprednú a dostáva váhu 1, inak váhu 0. Teraz uzly majú všetko potrebné pre vytvorenie sumy suffixov. Začíname od poslednej hrany a zistujeme poče dopredných hrán ktorá má daná hrana pred sebou. Po vypočítaní sumy suffixov už vypočítame hodnoty jednotlivých uzlov podla vzorca: **2*n-2-suffix+1**. Teraz máme všetko pre to aby sme mohli priradiť poradie preorder jednotlivým vrcholom. Uzol s najmenšou hodnotou je teda štartovací uzol, a za ním postupne nasledujú uzly s väčšími hodnotami.

Analýza: pole etour O(c) , priradeni etour pre hranu O(1) , list ranking O(log n), pozicia uzla(1)

Časová náročnost: O(log n)

Celková cena: c(n) = O(log n)*p(n)

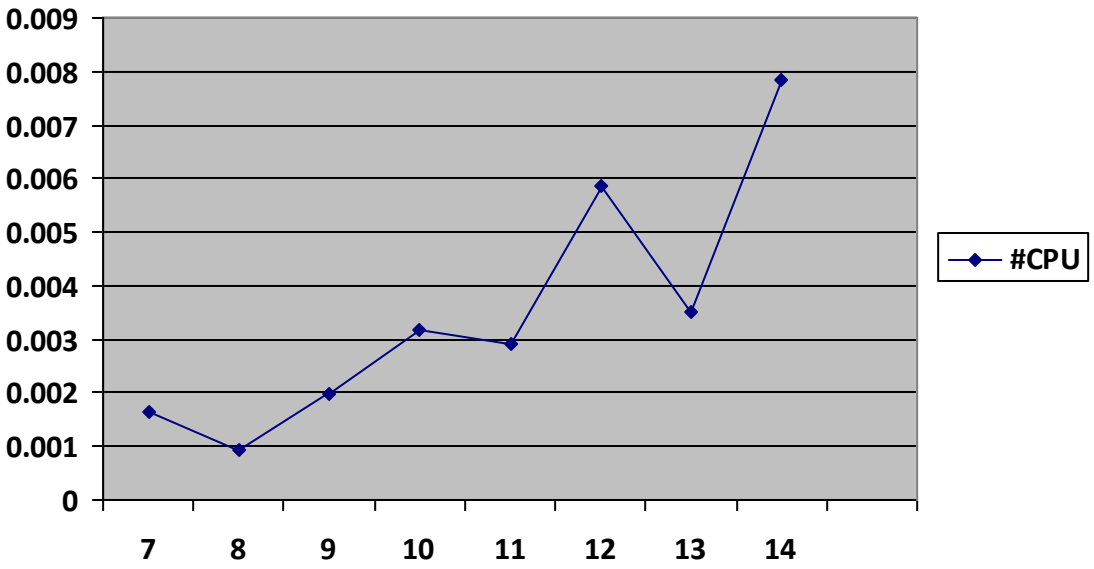
Priestorová náročnosť: p(n) = 2*n-1

Implementácia

Po rozparsovaní binárneho stromu sú hrany uložené do štruktúry ktorá ju reprezentuje uzlom z ktorého vychádza a uzlom do ktorého vchádza. Zároveň je vytvorená aj reverzná hrana. Tieto hrany sú uložené do ďalšej štuktúry ktorá tieto dve hrany zoskupuje, taktiež je v tejto štruktúre obsiahnutý ukazateľ ktorý v prípade takom že hrana má nasledníka, priradí tohoto následníka do ukazateľa *next*. Ďalšou položkou v tejto štruktúre je unikátne ID hrany. Tieto štruktúry vložíme do vektora hrán a tým sme vytvorili zoznam susednosti. Nasleduje výpočet hrany etour čiže hrany nasledníka. V cykle prechádzame zoznam susednosti ak reverzia hrany má nejakého nasledníka, uložíme ho do premennej *myEtour,* Inak sa jedná o listový uzol a ako jeho nasledníka nastavime jeho reverznú hranu. Ďalej si každá hrana zistí svoju váhu podľa indexov uzľov z ktorého vychádza a do ktorého vchádza. V tejto fáze algoritmu je potrebné rozposlanie pola váh a pola nasledníkov, čiže každá hrana pošle svoju hodnotu procesoru s číslom 0, ten vytvorí polia rozpošle ich znovu všetkým hranám. Takže každý procesor bude vedieť kto má akého nasledníka a kto má akú váhu. Nasleduje výpočet sumy suffixov, ten prebieha paralelne pre každý procesor. Výpočet je získaný z pola váh kde postupuje odzadu a každá nasledujúca hodnota je hodnotou predchádzajucou + hodnotou ktorá sa nachádza na danom indexe. Podľa sumy suffixov teraz vypočítame hodnoty uzľov z ktorých vedú dopredné hrany podľa hore uvedeného vzorca. Vypočítané hodnoty su poslané procesoru ktorý má ID 0, ten zoradí tieto hodnoty a vypíše postupnosť uzlov podľa hodnôt od najmenšej po najväčšiu. Dostali sme tak prechod preorder.

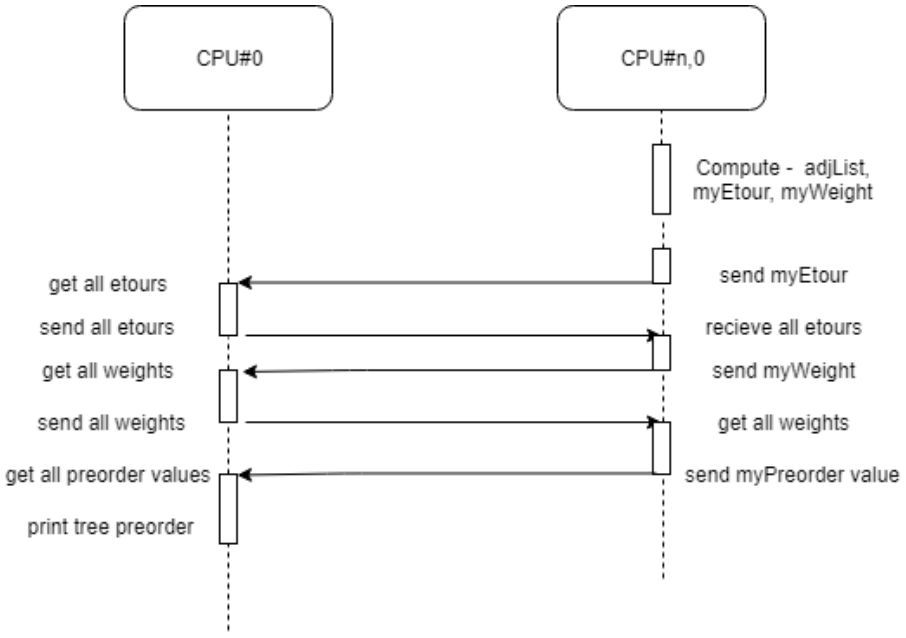
Experimenty

Pre meranie času bola použitá funkcia MPI_wtime(). Boli merané časy pre 7 až 14 uzlov. Kedže referenčný server na ktorom sa spúšťa program by viac procesorov nepovolil. Pričom pri každej veľkosti stromu bolo spustených 5 behov a z nich urobený priemer.



Komunikačný protokol

Tento komunikačný protokol popisuje komunikáciu medzi jednolivými procesmi. Všimnime si že aj procesor s číslom 0 prevádza rovnaké činnosti ako ostatné procesory, naviac ale zhromažďuje údaje od ostatných a nakonci vypíše postupnosť.



Záver

Tento algoritmus bol testovaný na školskom serveri Merlin ktorý povoluje spúštanie maximálne 26 procesorov, čo pre nás znamená binárny strom o maximálnej dĺžke 14 uzlov a teda nemôžeme využiť plný potenciáln tohoto algoritmu. Predpokladaná teoretická zložitosť nemôže byť dosiahnutá kedže v poslednej fáze procesor s rankom 0, vypisuje hodnoty sekvenčne. Dosiahnutá zožitosť sa teda blíži ku kvadratickej.