Paralelní a distribuované algoritmy
Výpočet úrovně vrcholu

Bc. David Kolečkář, xkolec07
xkolec07@stud.fit.vutbr.cz

22. dubna 2019

1 Úvod

V této dokumentaci je popsán algoritmus pro výpočet úrovně vrcholu stromu, který byl vytvořen v rámci projektu Paralelní a distribuované algoritmy. Dokumentace obsahuje analýzu algoritmu včetně odvození složitosti, popis implementace, rozbor experimentu a vizualizaci komunikačního protokolu.

2 Rozbor a analýza algoritmu

Algoritmus pro výpočet úrovně vrcholu stromu vyžaduje $2n-2$ procesorů, kde n je počet uzlů stromu. To znamená že pro každou hranu existuje jeden procesor. Prvním krokem algoritmu je výpočet Eulerovy cesty. Obecně pro orientovaný graf prochází Eulerova cesta každou hranu právě jednou. Pro orientovaný graf ho lze získat nahrazením každé hrany $< u, v >$ dvěma hranami $< u, v >$ a $< v, u >$. Při tvorbě Eulerovy cesty se vychází ze seznamu sousednosti. Dalším krokem algoritmu je určení váhy každé hrany, v případě dopředné hrany je hodnota -1 jinak 1. Předposledním krokem je spočítání sumy sufixů, neboli přiřazení sumy vah následníků až po konec pole. Nakonec probíhá korekce, kde se pro každou dopřednou hranu zvýší suma sufixů o jedničku a pro kořen se nastaví úroveň 0.

Asymptotická časová složitost tohoto algoritmu je $t(n) = \mathcal{O}(logn)$, kde n je počet uzlů stromu. Časová složitost vychází ze 4 částí:

- Výpočet Eulerovy cesty $\mathcal{O}(c)$
- Inicializace váhy hrany $\mathcal{O}(c)$
- Výpočet sumy sufixů $\mathcal{O}(logn)$
- Korekce výsledku $\mathcal{O}(c)$

Počet procesorů je $2 * n - 2 \rightarrow p(n) = \mathcal{O}(n)$. Celkově tedy cena algoritmu je $c(n) = t(n) * p(n) \rightarrow \mathcal{O}(n * log(n))$, což je optimální.

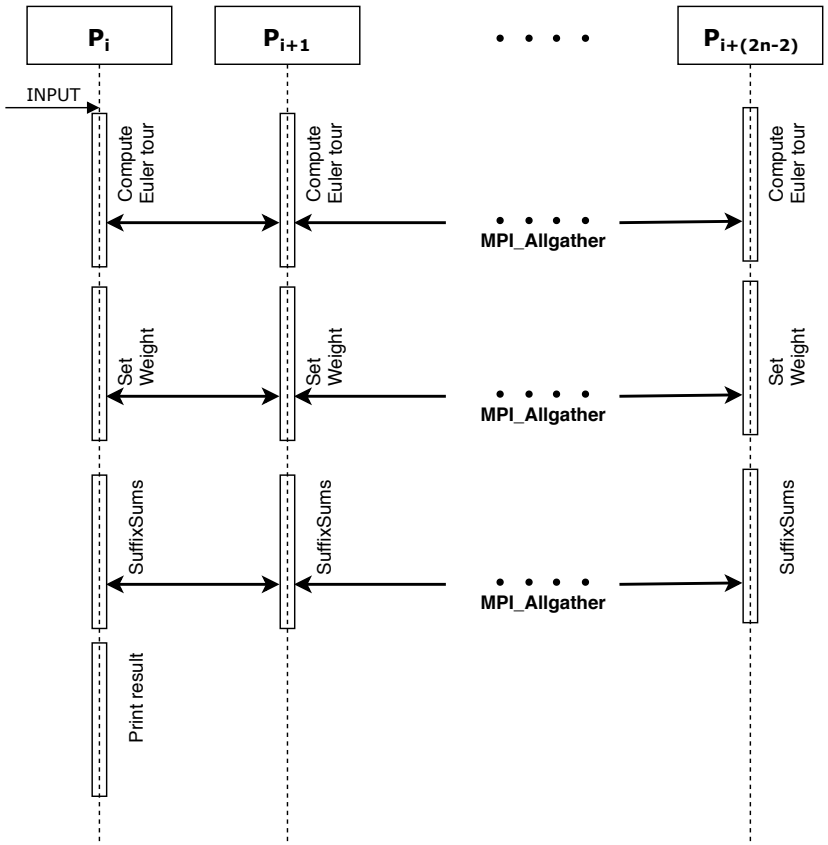
3 Implementace algoritmu

Pro spuštění aplikace byl vytvořen skript *test.sh*, který aplikaci přeloží a spočítá počet procesorů nutných pro spuštění aplikace. Parametrem skriptu je řetězec, který reprezentuje hodnoty uzlů binárního stromu, které se indexují od jedničky. Na základě délky vstupního řetězce je dopočítán počet procesorů $(2 * n - 2)$. Ve skriptu také probíhá kontrola parametrů. Pokud vstupní parametr není zadán nebo je jich více něž jeden skript vyhodí chybu.

Program byl napsán v jazyce C++ s využitím knihovny Open MPI. Jak již bylo uvedeno, vstupem programu je řetězec, který reprezentuje hodnoty uzlů binárního stromu. Znaky v řetězci jsou indexovány od jedničky (kořen má hodnotu 1). Pak levý potomek uzlu i má hodnotu $2*i$ a pravý potomek $2 * i + 1$. Dopředná hrana směřující do uzlu i má hodnotu $i - 1$ a zpětná hrana směřující z uzlu i má index $i + n - 2$. Na základě této informace byl vytvořen seznam sousednosti (struktura seznamu je: index hrany, index reversní hrany, uzel z kterého hrana vychází a informace o tom zda uzel má další hranu). Na základě struktury seznamu každá hrana (procesor) najde svého následníka a všechny procesory ho zapíší do pole. V tomto případě se nepodařilo naimplementovat výpočet Eulerovy cesty s konstantní složitostí. Dále se na základě informace o dopředných hranách určí váha každé hrany (-1 pro dopřednou hranu a 1 pro zpětnou hranu) a zapíše se do pole. V dalším kroku každý procesor spočítá sumu sufixů, respektive udělá sumu jednotlivých vah od svojí hrany až po hranu směřující do kořene stromu. Výstupem aplikace je řetězec, zapsaný ve formě hodnota uzlu:úroveň uzlu, kde jednotlivé uzly jsou odděleny čárkou. Tento výpis provádí jeden procesor. V případě že na vstupu je řetězec o délce jedna, je vypsán jedním procesorem hodnota uzlu s úrovní nula.

4 Komunikační protokol

Na obrázku 1 je znázorněna komunikace mezi procesory, jedná se o obecný model (proměnná n, uvedená v diagramu, znamená počet uzlů binárního stromu). Komunikace simuluje funkci MPI All-Gather.



Obrázek 1: Komunikační protokol

5 Experiment

Algoritmus byl testován na různě velké vstupy (respektive pro n o velikosti 6, 8, 10 a 12, kde n udává počet uzlů binárního stromu) pro ověření časové složitosti. K tomuto účelu byl vytvořen jednoduchý skript, který pro každé n spustil 50 iterací algoritmu a změřené časy zapisoval do souboru. Následně byl udělán z těchto hodnot průměr, který lze vidět v grafu 2. Měření v algoritmu bylo spuštěno před určením váhy každé hrany a skončilo po výpočtě sumy sufixů. Tvorba seznamu sousednosti (adjacency listu), inicializace knihovny openMPI a samotný výpis hodnot nejsou ve výsledném čase zahrnuty. V čase také není zahrnut výpočet eulerovy cesty, z důvodu neoptimální implementace této funkce. Pro měření času byla využita funkce *MPI Wtime()*.

Obrázek 2: Výsledky měření zanesené do grafu

6 Závěr

Teoretická časová složitost algoritmu uvedená v kapitole 2, odpovídá grafu získaného pomocí experimentu v kaptiole 4. Algoritmus byl úspěšně otestován na školním serveru merlin (CentOS) a na privátním počítači s operačním systémem Ubuntu 18.