1 Viditelnost

Viditelnost je v zásadě velmi jednoduchý algoritmus založený na výpočtu vektoru úhlů, provedení sumy prefixů[1] (s binární asociativní operací maximum) a následné porovnání úhlů s dosavadním maximem pro zjištění viditelnosti [1].

Složitost a cena: Vytvoření pole nadmořských výšek lze provést v konstantním čase. Pole úhlu je z něj také vytvořeno v konstantním čase, suma prefixů má logaritmickou časovou složitost (strom) a výsledné porovnání lze opět provést v konstantním čase. Výsledná časová složitost je tedy rovna složitosti výpočtu sumy prefixů: $O(\frac{n}{N} + log(N))$, kde n je velikost vstupu a N počet procesorů. První část $\frac{n}{N}$ značí sekvenční *prescan* detailně popsaný v sekci 1.1.1.

Časová složitost má jako parametr N, kterým lze nastavit parametry tohoto algoritmu. Pro velmi malé N převládá první část (tedy výsledná složitost bude lineární, stejná jako u optimálního sekvenčního algoritmu), nebo pro velké N převládá druhá část (tedy algoritmus bude nejrychlejší možný, bude mít logaritmickou složitost, ale cena nebude optimální). Nebo můžeme N zvolit tak, aby platilo $log(N) < \frac{n}{N}$, poté je časová složitost $O(\frac{n}{N})$, což říká: Kolikrát se zvýší počet procesorů, tolikrát se algoritmus zrychlí. Cena je pak optimální $O(n)$.

1.1 Max-prescan

Algoritmus se skládá ze tří částí a funguje na principu vytvoření vyváženého binárního stromu procesorů, a proto je pro výpočet potřeba druhá mocnina počtu procesorů, t.j. $\{1, 2, 4, 8, 16, ...\}$.

1.1.1 První fáze – operace *UpSweep*

Tato fáze je obdobná algoritmu *reduce*, pouze si každý uzel pamatuje mezisoučet. Cílem je pro každý uzel ve stromě vypočítat maximum všech jeho potomků. Pokud je počet procesorů N menší, než je velikost vstupu n, musí každý procesor provést sekvenční *reduce* (pro svou část posloupnosti délky $\frac{n}{N}$) s lineární časovou složitostí. Časová složitost samotného algoritmu *reduce* počítaného paralelně ve formě vyváženého binárního stromu je $O(log(N))$. Výsledná časová složitost se tedy skládá ze dvou komponent, první je optimální sekvenční algoritmus (lineární sl.) a druhá je stromový algoritmus (logaritmická sl.) a tedy: $O(\frac{n}{N} + log(N))$.

1.1.2 Druhá fáze – operace *Clearing*

Kořenu přiřadíme neutrální prvek (v tomto případě pro maximum to bude minimální hodnota daného datového typu). Časová složitost přiřazení je konstantní.

1.1.3 Třetí fáze – operace *DownSweep*

Fáze obdobná fázi *UpSweep*, pouze se nezačíná výpočet od listů nahoru, nýbrž od kořenu stromu směrem dolů. Uzel dá svému pravému potomkovi maximum hodnoty své a hodnoty levého potomka, a levému potomkovi dá přímo svou hodnotu. Cílem je do každého nelistového uzlu uložit dosavadní maximální hodnotu. Složitost je stejná jako u operace *UpSweep*, protože se opět prochází stromem a v každém uzlu se provádí činnost s konstantní časovou složitostí.

2 Implementace

Popis implementace je rozdělen na dvě části, první testovací skript v jazyce `Bash` a druhá část implementace algoritmu viditelnost v jazyce `C++` za pomoci knihovny `OpenMPI` [2].

[1] Dále jen *prescan*.

2.1 Testovací skript

Počet procesorů je zvolen tak, aby byla použita co největší mocnina čísla dva. To znamená, že pokud je počet vrcholů ve vstupních datech roven mocnině dva, je použita daná mocnina, jinak největší možná mocnina pod daným počtem vrcholů (viz. obrázek 2). Tedy pro počet vrcholů 8 bude použito 8 procesorů. Pro počet vrcholů 10 bude taktéž použito 8 procesorů, až pro počet vrcholů 16 bude použito 16 procesorů. Poté se provede kompilace a spuštění s daným počtem procesorů a předáním vrcholů na vstup programu `vid`.

2.2 Algoritmus viditelnost

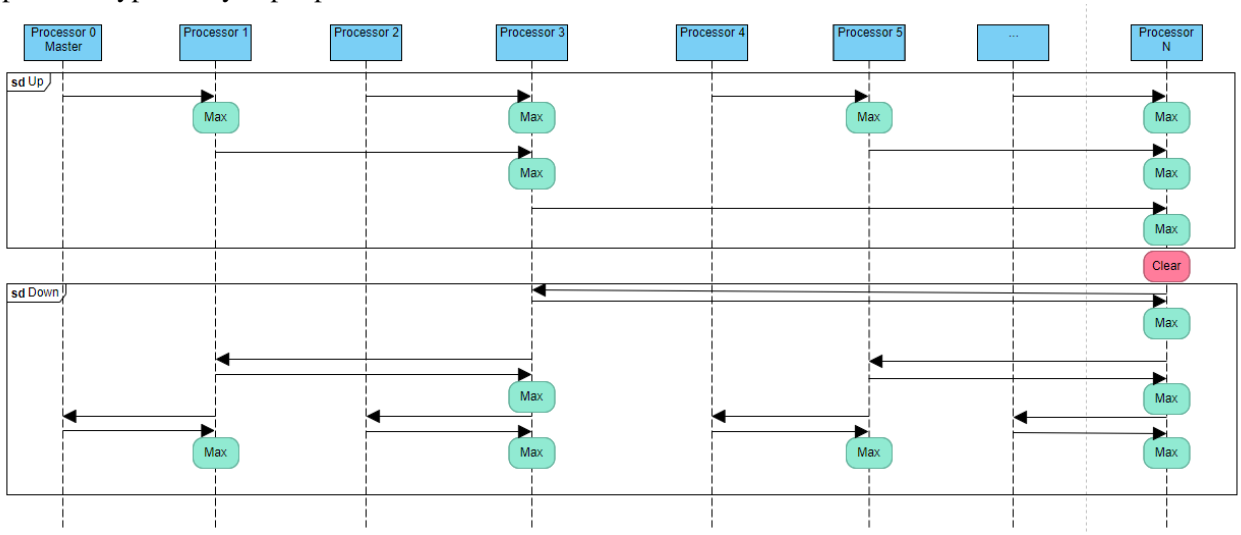
Na počátku jsou *master* (dále jen hlavním) procesorem načteny data ze vstupu a jsou distribuovány ostatním procesorům, které provedou výpočet úhlů. Vzhledem k tomu, že může být zadán i jiný počet vstupům, než je mocnina dvou, ale zároveň algoritmus (aby mohl vytvořit vyvážený binární strom) očekává mocninu dvou, je potřeba pro některé procesory provést sekvenční *max prescan* a dále pracovat jen s tímto "lokálním" maximem.

Dále se provádí fáze *UpSweep*, ve které se iteruje přes úroveň stromu (tzn. druhý logaritmus délky vstupu). V každé úrovni procesory na sudých indexech (leví potomkové) odesílají svým lichým sousedům (praví potomkové) své maximum. Tyto procesory na lichých indexech počítají maximum ze své a příchozí hodnoty a dále pracují pouze s tímto maximem. V posledním cyklu se maximum dostane až do kořenu stromu a každý nelistový uzel stromu obsahuje maximum svých potomků.

Poté se provede přiřazení neutrálního prvku, tzv. *clearing*. A to za pomoci přiřazení neutrálního prvku do kořenu stromu (pro binární asociativní operaci maximum to tedy bude minimální hodnota daného datového typu).

Ve fázi *DownSweep* se postupuje obdobně jako u *UpSweep*, pouze se postupuje v opačném směru, a to od kořenového uzlu stromu směrem k listům a procesory na lichých indexech tentokrát také odesílají svoji hodnotu svým sudým sousedům (levím potomkům), a až poté provádějí výpočet maxima.

Nakonec se provede porovnání vektoru úhlů s vektorem maximálních úhlů a zjistí se tak viditelnost jednotlivých bodů, což je následně odesláno hlavnímu procesoru, který tyto informace shromáždí od všech procesorů a provede výpis na výstup v požadovaném formátu.

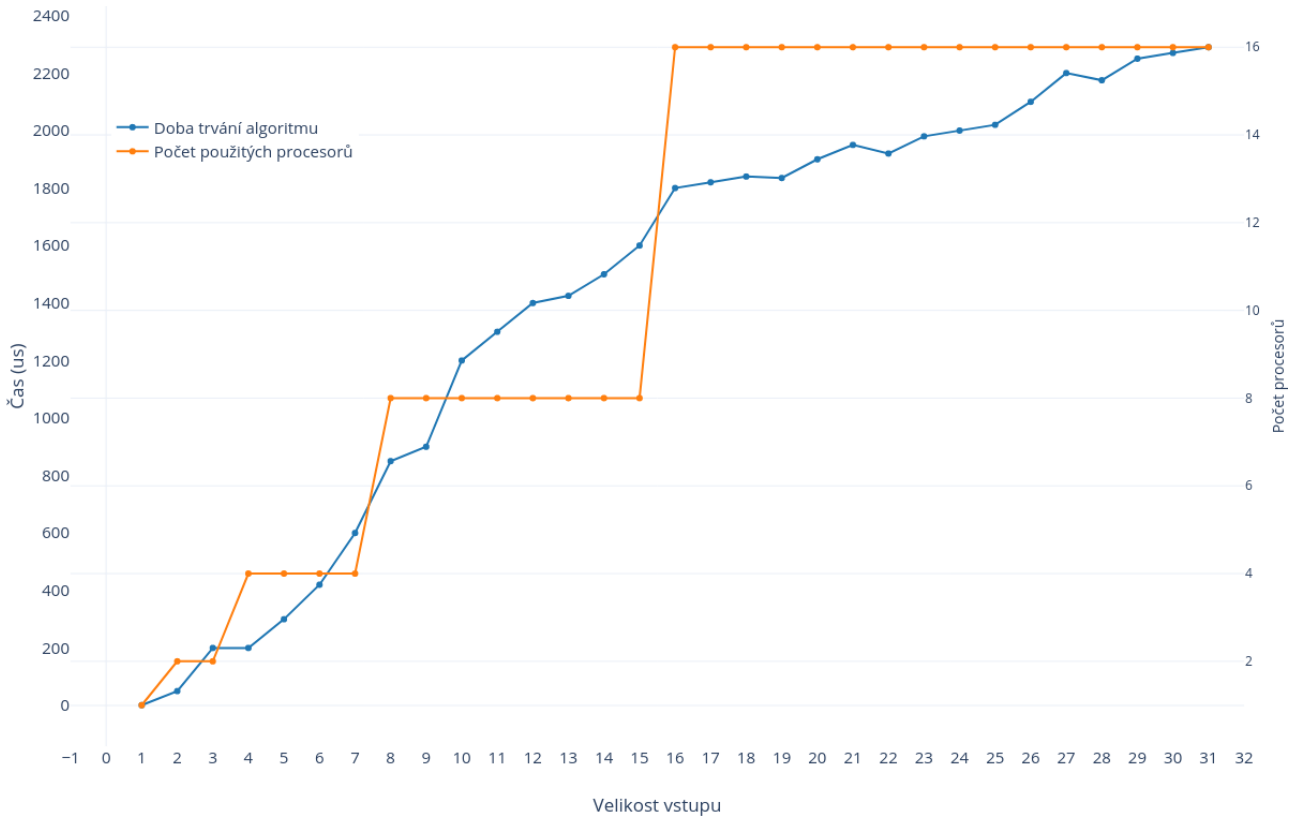


Obrázek 1: Obecný sekvenční graf zasílání zpráv při provádění algoritmu `max-prescan` pro N procesorů (za předpokladu, že N je mocnina čísla dva), bez počátečního rozeslání a koncového sesbírání hodnot do hlavního procesoru. Sekvenční graf je rozdělen na tři části, Up, Clear a Down, dle algoritmu (zachovány anglické názvy fází a operací pro zachování konzistentnosti). V grafu jsou také uvedeny operace provedené po přijetí zprávy. Jak lze vidět ze zaslaných zpráv, procesory vytvoří binární strom, kde procesor s číslem N je kořen stromu.[2]

[2]Vytvořeno pomocí `https://online.visual-paradigm.com/diagrams`.

3 Měření

Měřena byla pouze doba běhu výpočtu algoritmu sumy prefixů, nikoli počáteční a koncové rozesílání dat, výpis do terminálu, apod. Měření bylo provedeno pomocí standardní knihovny `std::chrono`. Měření není úplně směrodatné/přesné zejména proto, že byly použity příliš malé vstupy pro paralelizaci za pomoci `OpenMPI`, ale z části také proto, že server merlin má pouze 6 fyzických jader a kvůli režii při přepínání procesorů.
Měření je provedeno pouze do velikosti vstupu (počet vrcholů) rovno 31, protože se dle dané implementace použije stále jen 16 procesorů. Kdežto pro vstup velikosti 32 by se použilo již 32 procesorů, což je více než umožňuje server merlin. Výsledná hodnota je průměr sta měření. Oranžová křivka reprezentuje počet použitých procesorů (stupnice vpravo), modrá křivka reprezentuje čas potřebný pro výpočet algoritmu (stupnice vlevo).



Obrázek 2: Měření výkonnosti implementace algoritmu `max prescan` na serveru merlin.[3]

4 Závěr

Naměřený graf vykazuje známky časové složitosti algoritmu viditelnost, a tedy $O(\frac{n}{N} + log_2(N))$, ale se drobnými výkyvy. To je z části způsobeno velmi krátkou dobou běhu (tzn. snadná úloha s malým vstupem), pouze šesti fyzickými jádry na serveru merlin a režií při přepínání procesorů.
Se zvyšujícím se počtem procesorů by měla časová složitost růst víceméně logaritmicky (nebo lineárně $\frac{n}{N}$, podle množství použitých procesorů, jak je popsáno v úvodu), vzhledem k tomu, že se z procesorů "vytváří" vyvážený binární strom.
Cena algoritmu je optimální pouze v případě, že platí $log(N) < \frac{n}{N}$. Tato implementace tedy není optimální kvůli tomu, že některé procesory dělají zbytečnou práci. Vzhledem k tomu, že se používalo spíše větší N (množství procesorů), převládala spíše druhá část vzorečku $O(\frac{n}{N} + log(N))$, tedy algoritmus je nejrychlejší možný (logaritmická složitost), ale cena není optimální.

[3] Vytvořeno pomocí `https://plot.ly`.

Reference

[1] Petr Hanáček. *PRL Slides, Přednáška 7. - Model PRAM*. VUT FIT, 2010.

[2] Open MPI. Open source high performance computing.