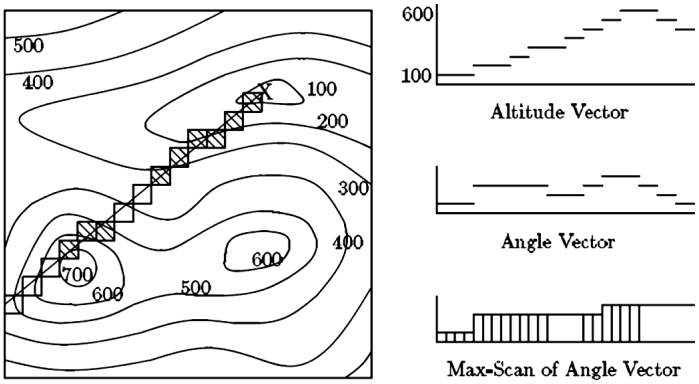
Paralelní a distribuované algoritmy - Projekt č.3

Buchal Petr, xbucha02

19. dubna 2020

1. Rozbor a analýza problému

Následující projekt řeší problém viditelnosti prezentovaný v přednášce předmětu PRL. Zjednodušeně se jedná o to, jestli pozorovatel z místa X o určité nadmořské výšce vidí bod Y v jiné nadmořské výšce, který může před sebou mít objekty o různých nadmořských výškách. Vizuální ukázka problému je na obrázku 1.



Obrázek 1: Vlevo je k dispozici výšková mapa na které je zobrazen vstupní vektor. Vpravo je vizualizace vektorů, pomocí kterých algoritmus počítá viditelnost. Převzato z shorturl.at/dkGW5.

Má paralelní implementace obsahuje počet procesů rovný počtu dotazovaných výšek. Na začátku algoritmu nultý proces odešle všem ostatním procesům dvě hodnoty. První hodnota je výška, ve které stojí pozorovatel, druhá hodnota pak výška náležící indexu procesu. Každý proces si poté vypočítá úhel (angle vector na obrázku 1). V následujícím kroku algoritmu postupně procesy posílají nejvyšší předchozí úhel následujícímu procesu. Z úhlu a nejvyššího předcházejícího úhlu (max-scan of angle vector na obrázku 1) pak každý proces spočítá, jestli je jeho výška viditelná nebo ne. Pseudokód algoritmu je na obrázku 2.

```
procedure LINE_OF_SIGHT
for each index i do in parallel
       angle[i] = arctan ((alt[i] - alt[0])/i)
max_prev_angle = max_prescan(angle)
for each index i do in parallel
       if (angle[i] > max_prev_angle[i]) then result[i] = visible
              else result[i] = unvisible endif
endfor
```

Obrázek 2: Pseudokód algoritmu pro výpočet viditelnosti. Převzato z shorturl.at/dkGW5.

2. Implementace

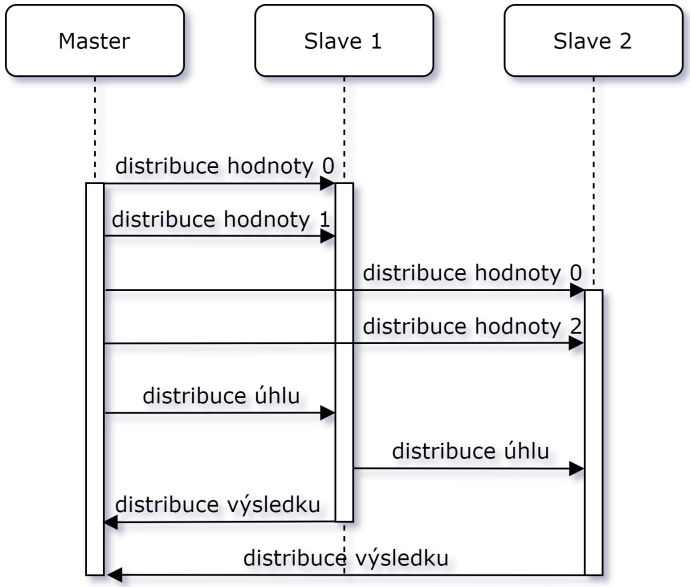
Algoritmus jsem implementoval v jazyce c++ s využitím knihovny Open MPI. Zdrojový kód se nachází ve dvou souborech. V souboru *vid.cpp* se nachází implementace algoritmu. Soubor *test* slouží

ke spouštění algoritmu.

Program v souboru *vid.cpp* nejprve rozdělí řetězec obsahující výšky, zadaný jako argument, na integery. Následně pomocí funkce *MPI_Send* rozešle všem procesům konkrétní přidělenou hodnotu. Ty je přijmou pomocí funkce *MPI_Recv*. Poté program přejde k vykonávání samotného algoritmu.

2.1 Komunikační protokol

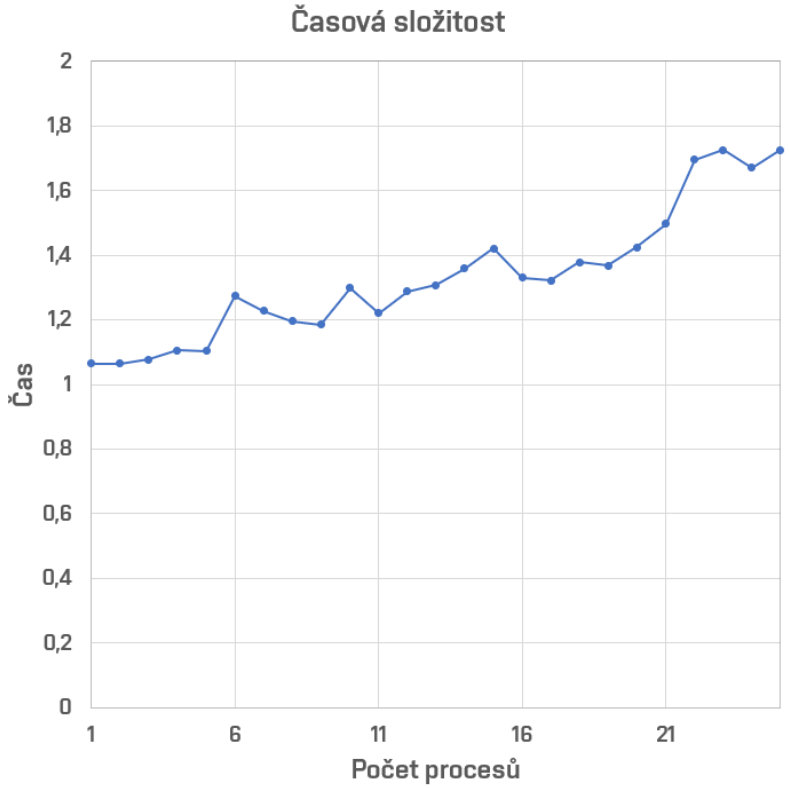
Vedoucí proces (tzv. master, viz obrázek 3) na začátku programu odešle všem procesům (včetně sebe) 2 zprávy. První obsahuje nultou zadanou hodnotu a druhá n-tou zadanou hodnotu, kde n je index procesu. První zpráva má TAG 0, druhá zpráva má TAG 1. Ostatní procesy čekají na příjem těchto dvou zpráv od nultého procesu. Po obdržení si každý proces vypočítá úhel a následně od nultého procesu probíhá zasálání zpráv procesům s vyšším indexem obsahující maximální předchozí úhel. Následně každý proces spočítá to, zdali je jeho výška viditelná a pošle zprávu obsahující tuto informaci nultému procesu. Ten vytiskne požadovaný řetězec. Vyjma druhé zprávy odesílané z master procesu se během komunikace mění jen obsah zprávy a políčko zdroj/cíl. Argument TAG zůstává s touto jednou výjimkou nastaven na hodnotu 0. Sekvenční diagram použitého protokolu je na obrázku 3.



Obrázek 3: Sekvenční diagram komunikace procesů pro 3 zadané hodnoty.

3. Experimenty

Pro počet hodnot od 1 do 25 bylo pro každou variantu provedeno 10 testů a jejich čas byl zprůměrován. Na obrázku 4 jsou vidět výsledky experimentu, které podporují logaritmickou časovou složitost algoritmu.



Obrázek 4: Graf zobrazující časovou složitost algoritmu vzhledem k počtu zadaných výšek.

4. Závěr

Během programování algoritmu pro viditelnost jsem si osvěžil programovací postupy knihovny pro paralelní programování Open MPI. Výsledky experimentu podporují logaritmickou složitost algoritmu.