4 laboratorinis darbas

Maksim Jaroslavcevas

April 2025

1 Problema

Sukurti lygiagretu Insertion sort“ rūšiavimo algoritma naudojant C programavimo kalba ir MPI instrumentus.

2 Lygiagretusis algoritmas

Dėl igyvendinimo specifikos as naudoju 'pipeline' buda. Kur visi processai yra isdestiti i viena 'pipelina'. Ir duomenis keliauja is pradinio iki paskutinio processo.

Pirmiausia inicializuojama procesu struktūra ('pipeline'), paimamas pirmasis skaičius ir siunčiamas pirmajam procesui. Tada paimamas antrasis skaičius ir taip pat nusiunčiamas pirmajam procesui, pirmasis procesas palygina abu skaičius ir, jei naujasis skaičius yra mažesnis už senaji, antrajam procesui nusiunčiamas senasis skaičius.Taip viskas kartojama, kol masyvas bus surūšiuotas.

3 Vykdymo aplinka

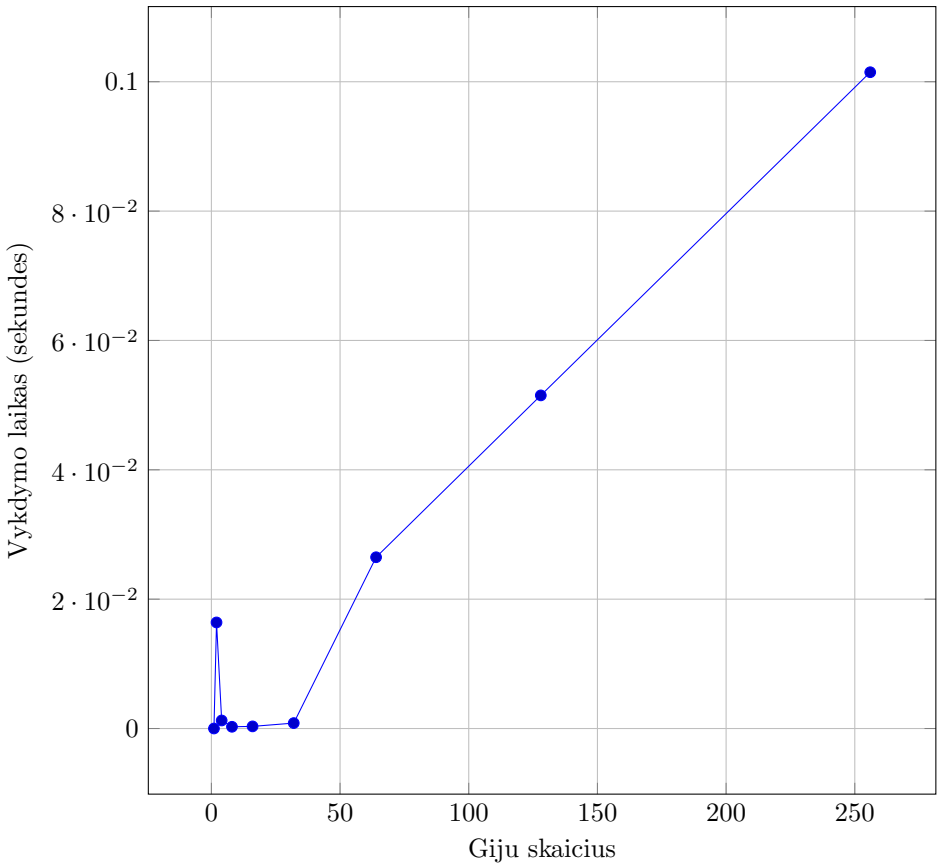
3.1 HPC klasteris

- (1, 2, 4, 8, 16, 32, 64, 128, 256) branduoliu
- Komandinis failas paleidimui MIF klasteryje žr. Priedas

4 Eksperimentinio tyrimo rezultatai

Šiam eksperimentui nusprendžiau iš karto sudaryti dvi diagramas, nes rezultatai nėra itin vienareikšmiai, apie tai rašoma išvadu skyriuje.

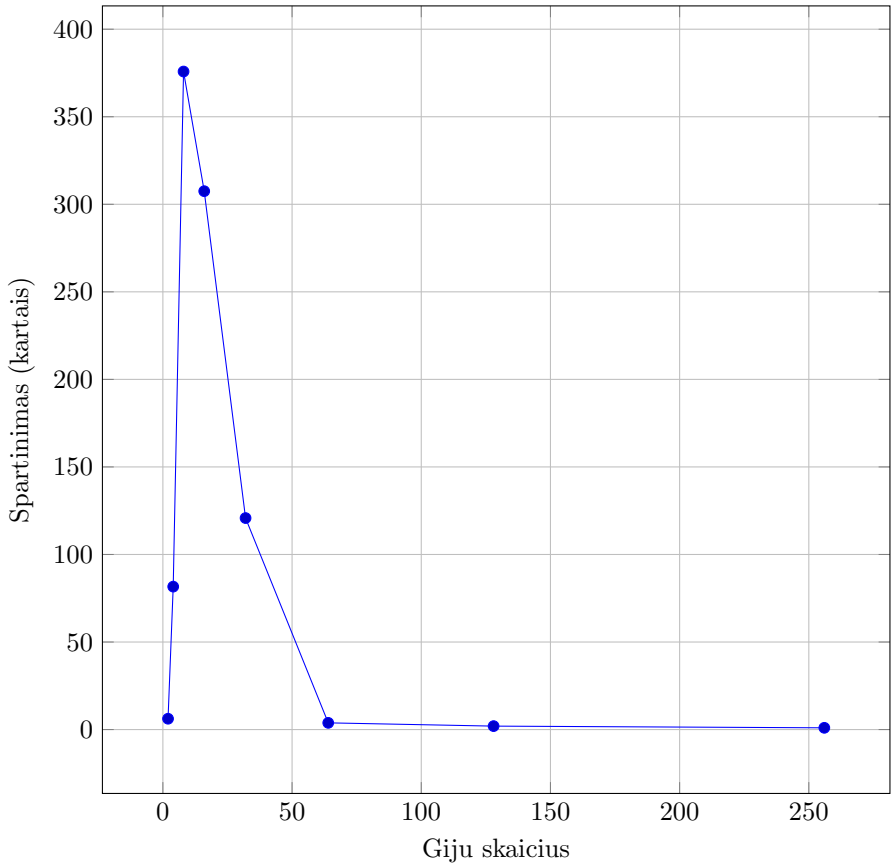
4.1 Vykdymo laiko nuo giju skaiciaus



Abiejuose bandymuose naudotas vienodas procesu skaičius (1, 2, 4, 8, 16, 32, 64, 128, 256), tačiau dėl rūšiavimo algoritmo igyvendinimo specifikos masyvo dydis taip pat turėtu būti lygus procesu skaičiui, todėl kiekviename bandyme naudotas skirtingo dydžio masyvas.

Dėl to sunku padaryti išvada apie našumo padidėjima, nes iš šio grafiko matome, kad rūšiavimo laikas tik augo, manau, kad tai gali būti susije su duomenu siuntimo tarp procesu mechanizmu, nes vienu metu 1 procesas atlieka tik viena palyginima, kuris yra labai greita operacija, o duomenu siuntimas tarp procesu užima daugiau laiko, nes reikia papildomos sinchronizacijos.

4.2 Spartinimo priklausomybe nuo giju skaiciaus



Šioje diagramoje parodytas skirtingo skaičiaus procesu produktyvumo skirtumas, tačiau, kaip ir pirmojoje diagramoje, labai sunku daryti išvadas apie šio metodo veiksminguma.

5 Išvados

Apibendrinant ši laboratorini darba, manau, galima teigti kad šis metodas yra labai dviprasmiškas, bent jau su šia salyga ir igyvendinimu. Nors šiuolaikiniai kompiuteriai gali atlikti milijardus instrukciju per sekunde, tačiau vis tiek pagrindinė kliūtis yra duomenu perdavimas tarp skaičiavimo mazgu, mūsu atveju - procesu.Taip pat manau, kad šis metodas turi labai rimta trūkuma, jis priklauso nuo sistemos skaiciavimo mazgu skaičiaus. Apskritai manau, kad tai puikus pipeline“ panaudojimas, tačiau manau, kad šiame pavyzdyje turėtume patobulinti algoritma didesniems masyvams ir labiau paskirstytiems skaičiavimams.

6 Priedas

```
#!/bin/bash
#SBATCH -p main
#SBATCH -n256
module load openmpi
mpicc -o connectivity connectivity.c
mpirun -np 1 ./connectivity
mpirun -np 2 ./connectivity
mpirun -np 4 ./connectivity
mpirun -np 8 ./connectivity
mpirun -np 16 ./connectivity
mpirun -np 32 ./connectivity
mpirun -np 64 ./connectivity
mpirun -np 128 ./connectivity
mpirun -np 256 ./connectivity
```