4 laboratorinis darbas

Maksim Jaroslavcevas

April 2025

1 Problema

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

2 Lygiagretusis algoritmas

Dėl igyvendinimo specifikos, parisinkau 'pipeline' principa, kuomet visi processai yra išdėstyti viename srautem o duomenys nuosekliai keliauja nuo pat pradžios iki pabaigos.

Rūšiavimas prasideda nuo procesu struktūros (konvejerio) inicializavimo. Tuomet pradinis duomenu masyvas yra padalijimas i lygias dalis - tiek, kiek turime procesu. Pirmasis didelio masyvo poaibis siunčiamas pirmajam procesui. Netrukus antrasis pomasyvas taip pat pasiekia ta pati pirmaji procesa. Šis procesas sujungia abu masyvus i viena, atlieak rūšiavima ir antraji gauto masyvo dali siunčia kitam procesui. Šis veiksmas kartojamas tol, kol visas masyvas yra galutinai surūšiuotas.

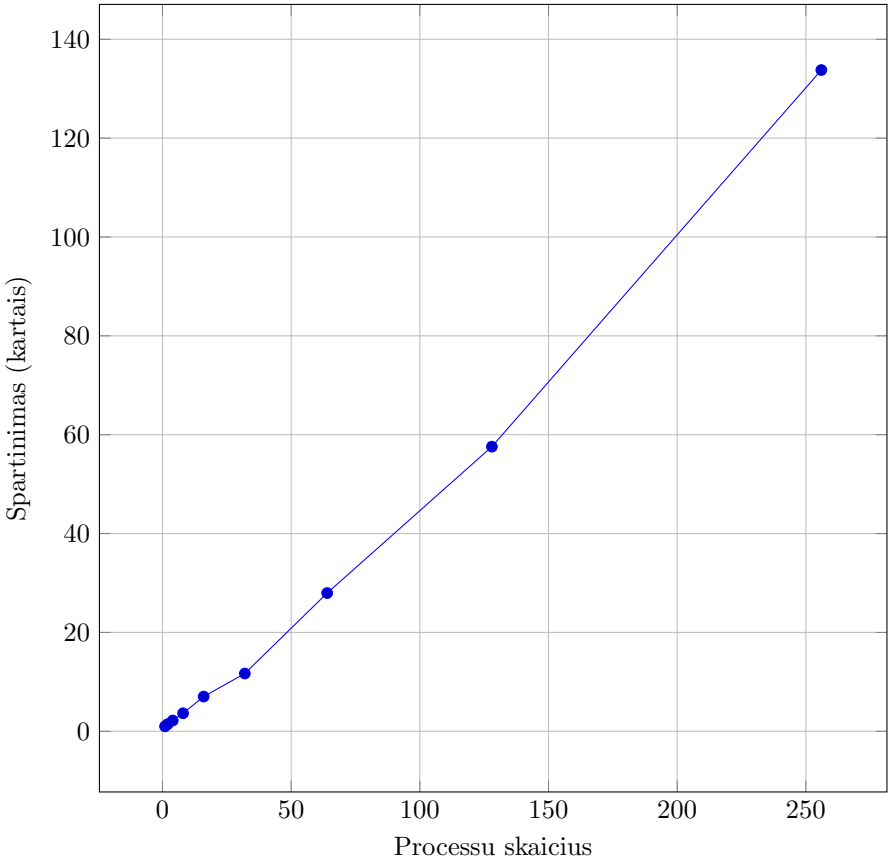
3 Vykdymo aplinka

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

4.1 Spartinimo priklausomybe nuo processu skaiciaus



Ši diagrama iliustruoja spartinimo pokyti priklausomai nuo procesu skaičiaus. Manau svarbu paminėti, kad eksperimente buvo naudojamas 10000 elementu masyvas. Rezultatai akivaizdžiai rodo, jog spartinimas puikiai didėja tiesiogiai proporcingai procesu skaičiui - padvigubinus procesu skaičiu, našumas išauga maždaug dvigubai.

5 Išvados

Apibendrinant ši laboratorini darba, galime drasiai teigti, kad procesu skaičius yra itin svarbus veiksnys, kai kalbame apie lygiagrečius algoritmus, realizuotus naudojant MPI, mūsu atveju rūšiavimo algoritmas modifikuotas Insertion Sort. Analizuodami rezultatus, nustatėme beveik tiesiogine našumo priklausomybe nuo procesu skaičiaus – našumas praktiškai padvigubėdavo kaskart padvigubinus procesu skaičiu. Manau, kad šis laboratorinis darbas puikiai parodo, kaip

naudojant keliu izoliuotu procesu arba net fiziškai atskirtu kompiuteriu (pvz., dvieju skirtingu kompiuteriu, esančiu toli vienas nuo kito) klasteri galima pasiekti labai gera našuma.

6 Priedas

```
#!/bin/bash
#SBATCH -p main
#SBATCH -n256
module load openmpi
mpic++ -o connectivity connectivity.cpp
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
```