Ataskaita

GitHub nuoroda: https://github.com/Juliakas/parallel_2

1 Užduotis

Aprašomi sprendimai pagal naujų objektų skaičių ir nuoseklaus algoritmo laikai.

Optimalios vietos:

- **3 naujiems objektams.** 0 1 20 (510884)
  Uždavinio laikas: 5.11
- **4 naujiems objektams.** 0 1 10 20 (599737)
  Uždavinio laikas: 30.47
- **5 naujiems objektams.** 0 1 10 20 22 (678416)
  Uždavinio laikas: 137.38

2 Užduotis

Aprašomi laikai sekundėmis, kai pritaikytas sulygiagretinimo sprendimas naudojant Open MPI.

3 naujiems objektams.

- **1 procesas:** 5.2962
- **2 procesai:** 2.709
- **4 procesai:** 2.06213

4 naujiems objektams.

- **1 procesas:** 30.4672
- **2 procesai:** 16.5753
- **4 procesai:** 11.4222

5 naujiems objektams.

- **1 procesas:** 137.184
- **2 procesai:** 76.6344
- **4 procesai:** 49.3585

3 Užduotis

Programos pradžioje kviečiamas MPI_INIT, pabaigoje MPI_FINALIZE. Pasirinkta optimizuoti užduotį per evaluateSolution dalį. Apskaičiuojamos pagrindinio ciklo dalys kiekvienam procesui pagal proceso rangą. Ciklas padalijamas į lygias dalis pagal visą procesų skaičių. Lokaliai kiekvienas procesas apskaičiuoja U reikšmę, kurią visi procesai turės susumuoti į bendrą reikšmę.

U reikšmės sumavimui yra parenkamas 0-inio rango procesas. Jis per MPI_RECV priima U reikšmes iš kiekvieno likusio proceso (kurie siunčia per MPI_SEND) ir kiekvieną gautą reikšmę prideda prie esamos sumos. Komunikuojama vienetiniais double tipo masyvais.

Vėliau, kad visi procesai turėtų vienodą informaciją, 0-inio rango procesas transliuoja per MPI_BCAST susumuotą U reikšmę ir kiti procesai ją atsiima.

Visi procesai nuo sukūrimo nuolat vykdo tą pačią programos logiką – išsiskirstymas vyksta tik evaluateSolution ciklo dalyje ir rezultatus spausdina tik 0-inio rango procesas.

Išvados

Išlygiagretinus evaluateSolution dalį ir naudojant 4 procesus, galima pasiekti:

- 2.56 kartų trumpesnį laiką 3-ims naujiems objektams.
- 2.67 kartų trumpesnį laiką 4-iems naujiems objektams.
- 2.78 kartų trumpesnį laiką 5-iems naujiems objektams.