VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Šilumos laidumo uždavinio lygiagretinimas MIF klasteryje

Parallelization of the steady-state heat problem on the MIF computing cluster

Kursinis darbas

Atliko: Mantas Petrikas (parašas)

Darbo vadovas: dr. Rokas Astrauskas (parašas)

Vilnius – 2022

TURINYS

Įvadas

Kompiuterinės simuliacijos yra svarbi modernių tyrimų dalis. Norint elektyviai atlikti sudėtingas simuliacijas, dažnai neužtenka vieno kompiuterio branduolio resursų, ir reikia paskirtyti darbą kelioms procesoriaus gijoms. Tai galima padaryti išnaudojant centriniam procesoriuje esančius branduolius ir gijas, naudojant grafinius procesorių, ar pasiteliant superkompiuteriuose esančias sujungtus mazgus, taip išnaudojant daugiau nei vieną centrinį ar grafinį procesorių.

Viena tipinė lygiagretino užduotis yra šilumos lygties sprendimas. Šilumos lygtis yra vienas iš Puasono lygties (ang. Poisson's equation) pritaikymo galimybių. Šios dalinės diferencialinės lygtys plačiai naudojamos fizikoje, sprendžiant elektrostatikos [Hou08], gravitacijos, magnetizmo [Bla96], pastovios būsenos temperatūrų [BE01] ir hidrodinamikos [Kad85] problemas. Ši dalinė diferencialinė lygtis turi kelis sprendimų būdus, kurių vienas yra naudojant baigtinių skirtumų metodą (ang. finite difference method) [YM15]. Šis metodas leidžia apskaičiuoti šilumos pasiskirtymą tam tikrame apribotame plote, žinant ploto kraštinių taškų temperatūrą. Norint efekyviau apskaičiuoti temperatūros pasiskirtymą plote, galima pradinį plotą suskirtyti į dalis, ir vykdyti temperatūros skaičiavimus atskirtuose plotuose lygiagrečiai. Šis darbas nagrinėja šilumos lygties sprendimo galimybes, naudojant centrinius procesorius (ang. CPU) lygiagrečiam lygities sprendimui apribotame plote, ir palygina lygiagretaus algortimo pagreitėjimą ir efektyvumą.

1. Šilumos laidumo uždavinys

Šio darbo tyrimo objektas - šilumos laidumo lygtis [BF11]. Šilumos pasiskitymą erdvėje aprašo Puasono lygtis dvimatei erdvei:

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y)$$

Kadangi šiame darbe bus nagrinėjamas erdvė be papildomų vidinių šilumos šaltinių, todėl $f(x,y) = 0$, o temperatūros pasisiskirtymą lemia kraštinių taškų temperatūra.

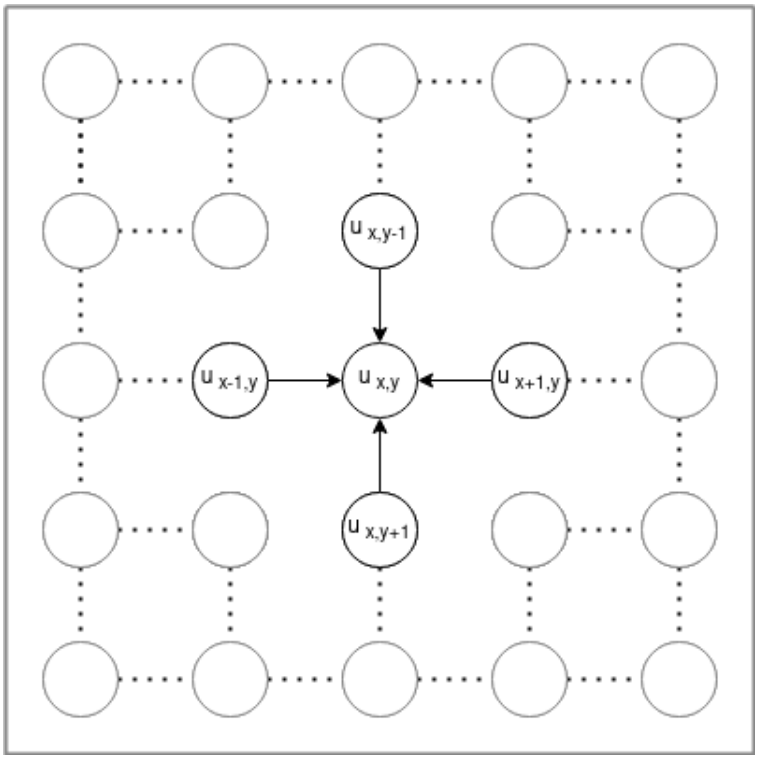
$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0$$

Ši lygtis dar yra žinoma kaip Laplaso lygtis (ang. Laplace's equation). Norint šia lygtį pritaikyti praktikoje, naudodami baigtinių skirtingų metodą apibrėžiame erdvę - matricą, kurioje bus skaičiuojama temperatūra, pradinę sistemos temperatūra šoniniuose ir vidiniuose taškuose.

Šiame rašto darbe nagrinėjamas temperatūros pasisiskirtymą kvadratinėje erdvėję, kurią susikirtysime į vienodo dydžio kvadratinius taškus, turinčius savo temperatūrą. Tada, taikydami baigtinių skirtumų metodą, gauname, kad vidinių taško temperatūra yra lygį jį supančių 4 taškų temperatūrų vidurkiui (1 paveikslėlis). Matematiškai tai galima išreikšti kaip:

$$u_{x,y} = \frac{u_{x,y-1} + u_{x-1,y} + u_{x+1,y} + u_{x,y+1}}{4};$$

kur $u_{x,y}$ - taško temperatūra taške koordinatėse x,y.



1 pav. Temperatūros pasiskirtymas vidiniuose matricos taškuose

Norėdami išspręsti šias lygčiu sistemas, naudosime Jakobi iteracinį metodą [BF11], kiekvienos iteracijos metu formulės kintamųjų reiškmės keisdami praėjusios iteracijos reikšmėmis. Laikysime, kad sistema pasiekė galutinę savo būsena, kai taško temperatūrų skirtumas skirtingų iteracijų metu bus mažesnis už norima paklaida.

Kraštinių taškų temperatūros pasiskirtymas nulemia galutinį temperatūros pasiskirtymą erdvėje. Vidinių taškų pradinis temperatūros pasiskirtymas įtakos galinam temperatūros pasiskirtymui neturi, tačiau gali turėti įtakos konvergacijos greičiui. Kraštinių taškų temperatūra šito tyrimo būdu aprašomą kaip:

$$f(x,0) = f(x,N-1) = |sin(x/N * 2\pi * 5/2)| = |sin(\frac{5\pi * x}{N})|$$

,

kur N yra matricos kraštinės taškų skaičius. Ši formulė pirmajai ir paskutinei eilutei priskiria 5 sin bangų teigiamas reikšmes. Likusiems taškams priskiriama 0 temperatūra. Šio darbo metu taškams prikiramos temperatūros reikšmės nuo 0 iki 1 imtinai, taip leidžiant modeliuoti į realia objekto temperatūra taikant paprasta proporcija. Vizualiai pradinis temperatūrų pasiskirtymas matomas 2 paveikslėlyje. Juodi taškai atitinka 1 reikšmę, balti - 0. Uždavinio tikslas - nustatyti galutinį temperatūros pasiskirtymą plokštelėje, kai temperatūra plokštelėje nebekinta. Leistina paklaida (temperatūros skirtumas kuri pasiekus laikysime kad programa pasiekė galutinę savo būseną)

šio tyrimo eksperimentų metu buvo 0.001. Vizualiai galutinis temperatūros pasiskirtymas matomas 3 paveikslėlyje. Laikas per kurį temperatūra pasiskirto erdvėje, erdvė sudarančios medžiagų šiluminis laidumas šiame darbe nėra nagrinėjami.

2 pav. Pradinis temperatūros pasiskirtymas

3 pav. Galutinis temperatūros pasiskirtymas plokštelėje

2. Šilumos uždavinio sprendimas

Šiame skyriuje nagrinėjamas programinis šilumos uždavinio sprendimas.

2.1. Programos testavimo aplinka

Siekiant išlaikyti vienodas sąlygas ir ištestuoti algoritmą turint didelį centinių procesorių kiekį, visi šiame darbe aprašyti eksperimentai buvo vykdomi Vilniaus Universiteto Matememathikos ir Informatikos fakulteto Skaitmeninių tyrimų ir skaičiavimų centro paskirstytų skaičiavimų tinkle. Šio darbo rašymo metu buvo naudojamas „beta“ telkinys, kurį sudaro 56 (testavimo metu praktiškai buvo pasiekiami tik 42) mazgai turinys po 2 Intel Xeon X5650 procesius, kurių kiekvienas turi po 6 branduolius. Kiekvienojame mazge yra 24 GB operatyviosios atminties (ang. RAM) ir jie turi prieigą prie 20Gbit/s infiniband tinklo. Skaičiavimų tinklę buvo instaliuota Debian operacinė sistemą, lygiagrečiam algoritmui buvo naudojamas OpenMP bibliotekos 6.3.0 versija. Programoms paleisti buvo naudojama sistemoje esanti „Slurm“ užduočių valdymo sistema ir „short“ užduočių eilė, turinti 2 valandų ir 2 GB vienam naudojamam branduoliui limitą.

2.2. Nuoseklusis algoritmas

Nuoseklusis algoritmas įgyvendintas C++ kalba. Algoritmo pradžioje alokuojama atmintis dviems N*N dydžio masyvams, kur N yra matricos kraštinės ilgis: viename saugoma dabartinė matricos būsena, kitas yra pildomas naujomis reikšmėmis iteracijos metu. Naudojami viemačiai masyvai, nes didžioji dalis OpenMPI biblietekos funkcijų, kuri buvo naudotos lygiagretinant algoritmą, tikisi vienačius masyvų duomenų perdavimui, o nuoseklaus algoritmo veikimui vienmačių ir dvimačių masyvų naudojimas nedaro didelės įtakos algoritmo veikimo laikui. Pirmasis masyvas užpildomas pradine matricos būsena, tada pradedamas iteracinis procesas, kai kiekvienos įteracijos metu, kiekvienam vidiniam matricos taškui yra priskiriama reikšmė, lygi jį supančių 4 taškų vidurkiui, o kraštinės matricos reikšmės nekinta. Taip pat iteracijos metu fiksuojamas maksimalus temperatūros pasikeitimas, skirtas nustatyti ar matrica pasiekė galutinė savo būseną. Ši reikšmė kiekvienos iteracijos palyginama su nustatyta paklaidos reikšme ir jei temperatūros pasikeitimas yra mažesnis už leistina paklaida, laikome kad matrica pasiekė pasavo galutinį temperatūrų pasiskirtymą.

Testuojant nuoseklųjį algoritmą su skirtingais matricos kraštinės ilgio reikšmėmis (1 lentelė), pastebėta kad algoritmo vykdymo laikas logoritmiškai priklauso matricos kraštinės ilgio. Padidinus kraštinės ilgį beveik 2 kartus, vykdymo laikas padidėjėja beveik 4 kartus, nes algoritmas nag-

Matricos kraštinė	Vykdymo laikas (s)			
	Bandymas 1	Bandymas 2	Bandymas 3	Vidurkis
258	7.71	7.712	7.815	7.746
250	7.710	7.712	7.815	7.746
514	35.243	34.921	34.721	34.962
1026	140.821	141.342	140.621	140.928
2050	577.458	572.498	582.138	577.365
4098	2393.624	2421.341	2362.241	2392.402

1 lentelė. Nuoseklaus algortimo vykdymo laikas

rinėja šilumos pasiskirtymą dvimatinėjė erdvėje, todėl padidinus matricos kraštinės ilgį 2 kartus, duomenų kiekis padidėja 4 kartus. Matricos kraštinės ilgiai pasirinkti dvejeto laipsniai + 2, kad lygiagretinant algoritmą vidinės matricos reikšmes būtų padalinti skirtingams procesams vienodais kiekiais. Pridedamos dvi eilutės šiame skaičiavime atitinka dvi matricos kraštines - eilutes ir stulpelius kurių temperatūrų reikšmės nekinta. Testuojant nuoseklųjį algoritmą, kai matricos kraštinę sudaro 8194 taškai, buvo pasiektas sistemos užduočių vykdymo laiko limitas (2 valandos).

2.3. Lygiagretieji algoritmai

Pradžiai apribrėžsime kriterijus pagal kurios bus vertinami lygiagretus algoritmas. Vienas iš pagrindinių vertinimo kriterijų yra algortimo pagreitėjimas, nusakantis kiek kartų greičiau lygiagretus algortimas, naudojantis p branduolių, įvykdo užduotį už nuoseklųjį algoritmą [EZL89]. Algoritmo pagreitėjimas vertinamas kaip:

$$S(p) = \frac{t_s}{t_p},$$

kur p yra procesoriu skaičius, t_s - geriausias nuoseklaus algoritmo vykdymo laikas, t_p - lygiagretaus algoritmo, naudojančio p procesorių, vykdymo laikas.

Maksimalus įmanomas teorinis pagreitėjimas gali būti tiesinis. Tai galėtų būti pasiekta duomenys yra padalinami į vienodas dalis, o komunikacijos tarp procesų nevyksta. Tuomet pagreitėjimas yra lygus procesorių skaičiui:

$$S_{max}(p) = \frac{t_s}{t_s/p} = p.$$

Jei lygiagretaus algoritmo pagreitėjimo reikšmė yra didesnė nei procesorių skaičių - $S(p) > p$ - pasiekiamas supertiesininis (ang. superlinear) pagreitėjimas. Tai gali atsitikti dėl neoptimaliai įgyvendinto nuoseklaus algoritmo, arba netiksliai įgyvendinto lygiagretaus algoritmo, kai yra praleidžiami žingsiniai.

Maksimalus praktinis algoritmo pagreitėjimas priklauso nuo algoritmo dalies, kuri gali būti išlygiagretinamas. Jei algoritmo dalį, kuri negali būti išlygiagretinama, pažymėsime f ($0 \leq f \leq 1$), trumpiausias lygiagrečios programos vykdymo laikas bus lygus $f * t_s + (1 - f) * t_s/p$, o maksimalus pagreitėjimas apibrėžiamas kaip Amdahl'o dėsnis [Amd67]:

$$S(p) = \frac{t_s}{f * t_s + (1 - f) * t_s/p} = \frac{p}{1 + (p - 1)f}.$$

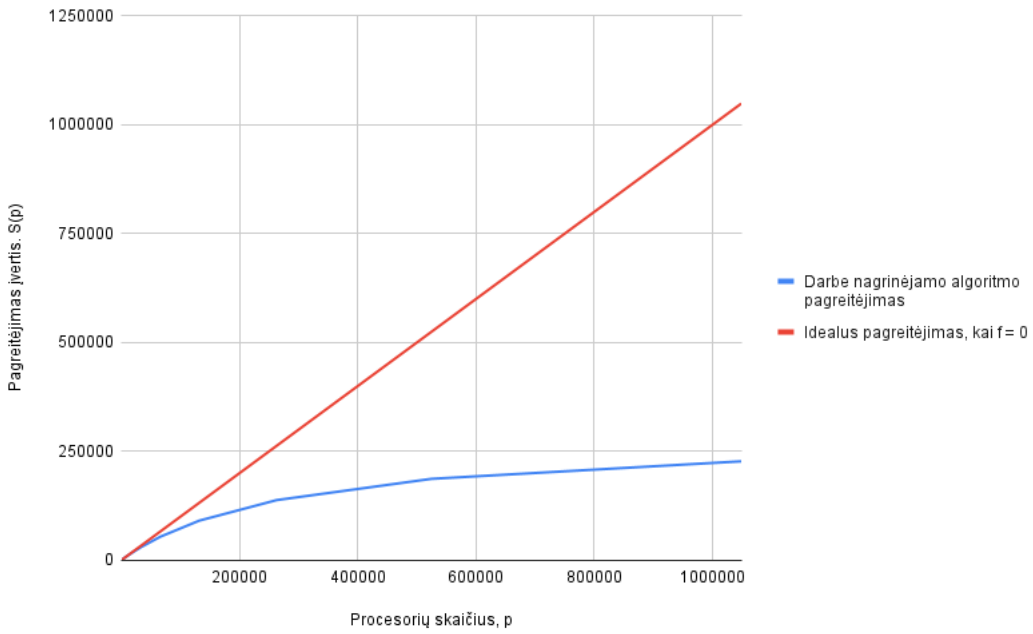
Šiame darbe nagrinėjamo lygiagretaus algoritmo nuosekliojoje dalyje (t.y. dalyje kuri nėra išlygretinama) vyksta tik konfiguracinių parametrų nuskaitymas. Taip pat nėra lygiagretinams paketų užkrovimas ir kitos prieš pagrindinės (ang. main) programos funkcija iškvietimą vykstančios operacijos, tačiau tai nedaro įtakos algoritmo tyrimui, nes laiko matavimas pradedamas tik iškvietus pagrindinę programos funkciją. Matricos kraštines suskaidžius į 2050 taškų, nuoseklaus algortimo konfiguracijos nuskaitymas vidutiniškai truko 0,002 sekundes (žymesime t_{np}), kai tuo metu likusieji skaičiavimai vidutiniškai užtruko 577,363 sekundes. Toks laiko santykis reikia kad nelygiatinama algoritmo dalis yra:

$$f = \frac{t_{np}}{t_p} = \frac{0{,}002}{0{,}002 + 577{,}363} \approx 0.00000346.$$

Šis įvertis reiškia, kad maksimalus algortimo pagreitėjimas naudojant begalinį skaičių procesorių bus lygus:

$$\underset{p \to \text{inf}}{S(p)} = \frac{1}{f} = \frac{1}{0.00000346} \approx 289017;$$

o tai reiškia neišlygetinama algoritmo dalis turės įtakos lygiagretaus vykdymo laikui tik naudojant labai didelį procesų kiekį.



4 pav. Maksimalus lygiagretaus algortimo pagreitėjimo įverčio priklausomybė nuo procesorių skaičiaus

Maksimalaus pagreitėjimo reiškmes naudojant skirtus procesorių kiekius matomos 4 grafike. Šio darbo ribose algortimas nebus tiriamas naudojant labai didelį procesorių kiekį, todėl net ir naudojant maksimalų procesorių kiekį (256), maksimalus pagreitėjimo reikšme bus artima procesorių skaičiui ($S(256) \approx 255.774$), todėl neišlygegretinta algoritmo dalis neribos algoritmo pagreitėjimo.

Taip pat lygiagretiems algortimams vertinti naudojama efektyvumo metrika [EZL89]. Darant prielaidą, kad nuoseklusis algoritmas visada efektyviai išnaudoja jam priskirto procesoriaus resursą, efektyvumo metrika parodo kokia laiko dalį nuoseklusis sprendimas išnaudoja jam priskirtus procesorius. Efektyvumas, dažniausiai žymimas reide E, matematiškai išreiškiamas kaip nuoseklaus algoritmo ir lygiagretaus algoritmo padauginto iš naudotų branduolių skaičiaus vykdymo laiko santykis:

$$E = \frac{t_s}{t_p * p}$$

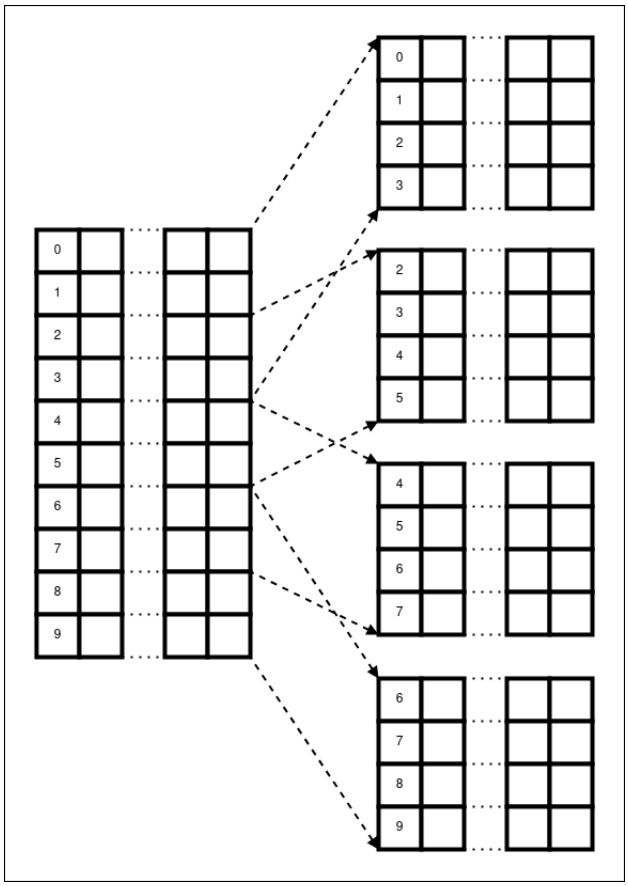
Taip pat efektyvumą galime išreikšti per algoritmo pagreitėjimą:

$$E = \frac{S(p)}{p}$$

Įdealiu atveju, esant maksimaliam algoritmo pagreitėjimui $S(p) =$" algoritmo efektyvumas bus lygus 1. Tai pasiekiama kai visi procesoriai visa laiką yra išnaudojami skaičiavimo užduotims.

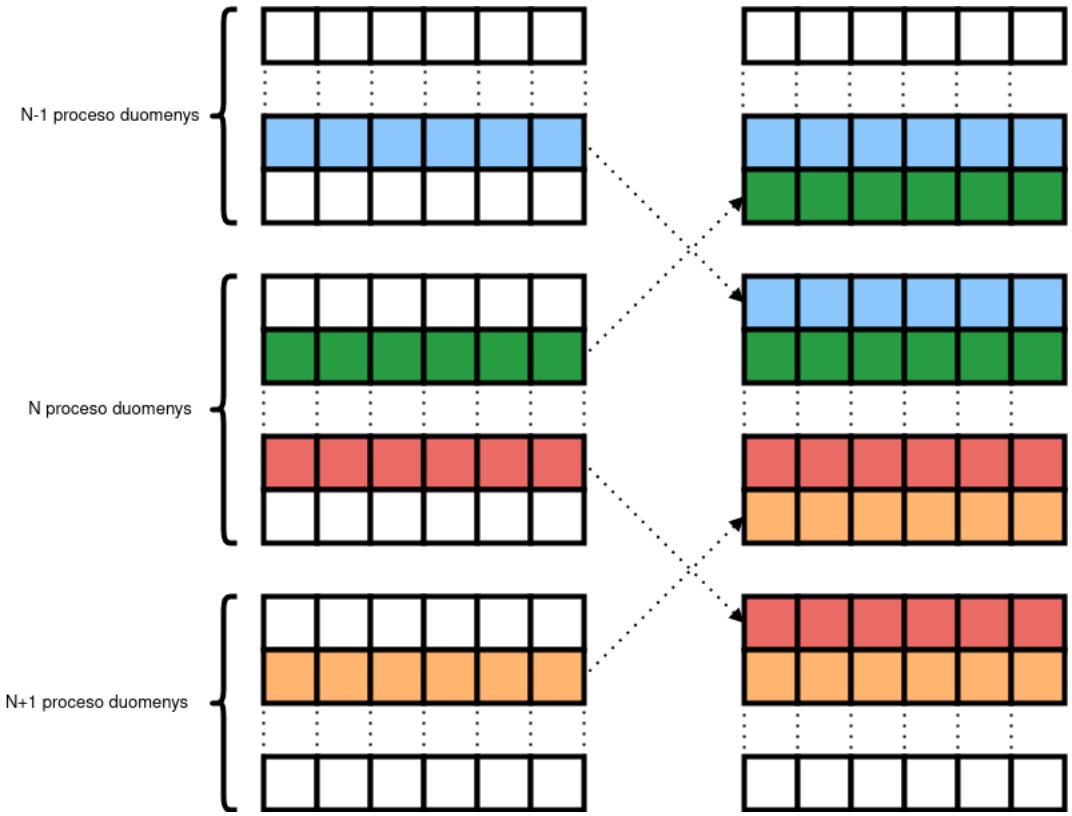
2.4. Duomenų dalinimas eilutėmis

Viena iš duomenų paskirtimo lygiagrečiajam algoritmui strategijų yra duomenis paskirtinyti eilutėmis (arba stulpeliais). Norint $N * N$ dydžio matricos duomenis padalint duomenis p procesams, kiekvienam procesui programos vykdymo pradžioje priskiriama po $(\frac{N-2}{p} + 2) * N$ taškų. $N - 2$ atitinka vidinių matricos stulpelių kiekį, kuris kinta laikui bėgant (2 atitinka išorinius matricos stulpelius turinčius pastovią temperatūrą). Ši reikšmė padalinama iš procesų skaičiaus, ir kiekvienas procesui priskiriamos po 2 papildomas eilutes kurios yra naudojamos vidinių reikšmių skaičiavimui (5 paveikslėlis).



5 pav. Duomenų padalinimas. 10 eilučiu padalinamos 4 procesams.

Naudojant Jakobi iteracinį metodą, kiekvienos iteracijos metu, pirmos ir paskutinės padalintų eilučių reikšmės turi būti sinchronizuojamos tarp gretimas matricos dalis apdorojančių procesų 6. Pirmasis ir paskutinis procesas šonines reišmės sinchornizuoja tik su vienu iš procesu, nes viena iš jas sudarančių matricos eilučių atitinka pradinės matricos kraštines eilutes, turinčias pastovias reikšmes.



6 pav. Duomenų synchronizacija tarp skirtingų procesų

Norint nustatyti, kada didžiausias temperatūrų iteracijų skirtumas pasiekė norimą reikšmę ir reikia nustoti vykdyti programą, kiekvienoje proceso iteracijos pabaigoje suskaičiuoja didžiausią lokalų temperatūros pokytį ir jį siunčia pagrindinam procesoriui, kuris gavęs visas reišmes, įvertiną ar bent vienas procesoriaus lokalus skirtumas viršyja norima paklaidą, ir nusprendžia ar tęsti iteravimą. Norint išvengti visuotinio sinchronizavimo tarp visų branduolių ir pagrininio proceso kiekvienos iteracijos metu, maksimlaus pokyčio reikšmių siuntimas ir palyginimas vykdomas tik kas 100 iteracijų.

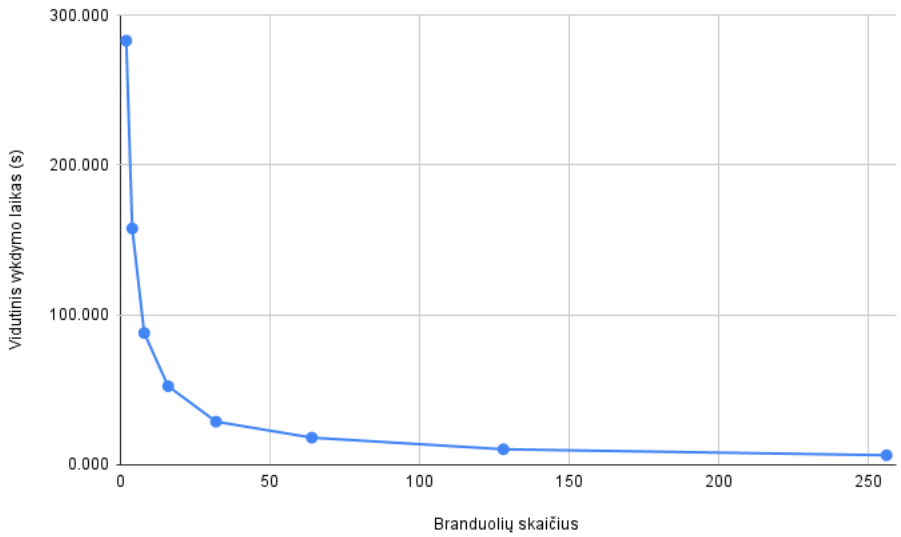
2.5. Lygiagretaus algortimo, kai duomenys padalinimo eilutėmis, analyzė

Lygiagretaus algoritmo veikimo laiko įverčiai, kai matricos kraštinę sudoro 2050 taškų pateikiami 2 lentelėje. Prieš testuojant lygiagretūjį algoritmą pakeitus branduolių skaičių, buvo atliekamas testinis paleidimas, kurio vykdymo laikas nėra įtrauktas į šią lentelė. Testinio paleidimo yra skirtas užtikrinti kad visi užduočiai reikalingi mazgai nebūtų miego busenoje (ang. Hibernation) ir būtų pasiruošę vykdyti algoritmą.

Kaip matoma 7 grafike, vykdymo laikas atvirkšiai priklauso nuo naudojamų branduolių skaičiaus.

Branduolių skaičius	Vykdymo laikas (s)			
	Bandymas 1	Bandymas 2	Bandymas 3	Vidurkis
2	304.223	302.624	302.871	303.239
4	157.587	157.362	158.112	157.687
8	87.142	87.452	88.655	87.750
16	52.371	51.578	52.713	52.221
32	29.452	28.826	30.192	29.490
64	16.897	17.044	16.454	16.798
128	10.299	9.956	9.844	10.033
256	6.027	6.078	6.028	6.044

2 lentelė. Lygiagretaus algoritmo vykdymo laikas, kai matricos kraštinę sudaro 2050 taškų



7 pav. Lygiagretaus algoritmo vykdymo laikas priklausomybė nuo naudojamų branduolių skaičiaus, kai matricos kraštinę sudaro 2050 taškų

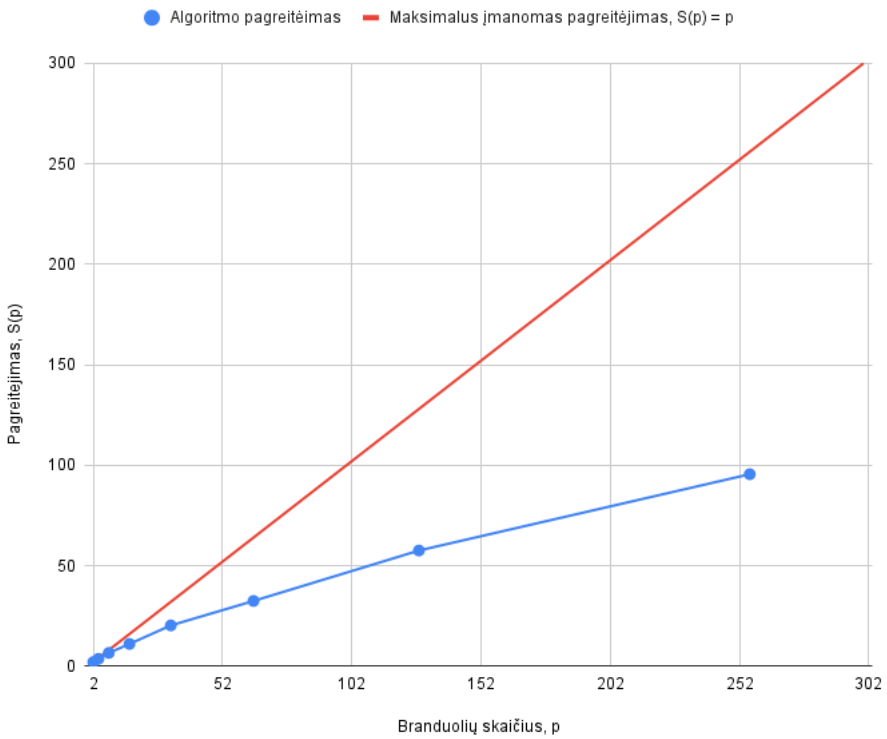
Taip pat lygiagretus algortimas buvo testuojamas kai matricos kraštinę sudarė 16386 taškai, vykdymo laikas matomas 3 lentelė. Testuojant naudojant 64 ar mažiau branduolių, programos veikimo laikas viršijo 2 valandas, ir programa buvo sustabdyta. Naudojant šią matricos kraštinės reikšmę nuosekliojo algoritmo programa viršydavo sistemos atminties limitą (2GB 1 branduoliui).

Branduolių skaičius	Vykdymo laikas (s)			
	Bandymas 1	Bandymas 2	Bandymas 3	Vidurkis
128	4419.302	4403.089	4418.429	4413.607
256	2464.156	2421.661	2468.414	2451.410

3 lentelė. Lygiagretaus algoritmo vykdymo laikas, kai matricos kraštinę sudaro 16386 taškai

Branduolių skaičius	Vykdymo laikas	Pagreitėjimas, $S(p)$	Efektyvumas, $E(p) = \frac{S(p)}{p}$
1	577.365	1	1
2	303.239	1.903991127	0.95
4	157.687	3.661462264	0.92
8	87.750	6.579683114	0.82
16	52.221	11.05625487	0.69
32	29.490	19.57833164	0.61
64	16.798	34.37037405	0.54
128	10.033	57.54659623	0.45
256	6.044	95.52170077	0.37

4 lentelė. Lygiagretaus algoritmo pagrėitėjimas ir efektyvumas.



8 pav. Lygiagretaus algoritmo pagreitėjimo priklausomybė nuo naudojamų branduolių skaičiaus, kai matricos kraštinę sudaro 2050 taškų

8 grafike matomas lygiagretaus algortimo logoritminis pagreitėjimas, tačiau Amdahl'o dėsnio [Amd67] apibrėžiamas maksimalus pagreitėjimas nėra pasiekiamas - naudojant tokį branduolių kiekį pagreitėjimas turėtų būti artimas idealiam pagreitėjimui $S_{max}(p) = p$. Tai galimai nutinka

dėl to, kad Amdahl'o dėsnis neatisižvelgia į komunikacijos tarp skirtingų procesorių kaštus. Tikslų programos skaičiavimų ir komunikacijos santykį įvertinti yra sunku, nes komunikacijos laikas gali skirtis laikas prikausomai ar komunikuojantys procesai naudoja to pačio ar skirtingų centrinių procesorių branduolius, ir ar centriniai procesoriai yra toje pačioje ar skirtinguose motininėse plokštelėse. Tas iš dalies matoma 4 lentelėje, kad algortitmo efektyvumas naudojant 2 ir 4 branduolius yra artimas 1, o naudojant daugiau branduolių lygiagretaus sprendimo efektyvumas mažėja. Tai galima sieti su tuo kad testavimui naudoto paskirstytų skaičiavimų tinklo mazguose naudojami procesoriai, turinys 6 branduolius, todėl naudojant 2 ir 4 branduolius didelė tikimybė, kad bus naudojamas 1 fizinis procesorius, taip išvengiant didelių komunikacijos kaštai tarp atsikirų procesų. Taip pat, 4 lentelėje matomas algortimo efektyvumo mažėjimas didinant branduolių skaičių, iš to galime daryti prielaidą, kad pradinius duomenis skaidant į vis mažesnes dalis, vis didesnę algoritmo vykdymo laiką užima komunikacija tarp skirtingus duomenis apdorojančių procesų. Nors testuojant su vis didesnių branduolių skaičiumi algortimo efektyvumas mažėja, pagreitėjimo metrika didinant naudojamų branduolių nenustoja didėti, iš to galima daryti prielaidą šių testų metu nebuvo rastas branduolių skaičius, kurį naudojant kominikacijos kaštai nusvertų gautą algoritmo pagreitėjimą padalinus užduotį į mažesnes dalis.

Rezultatai ir išvados

Buvo implementuoti nuoseklusis ir lygiagretusis algoritmai, spendžiantys šilumos laidumo lygti Jacobi iteraciniu metodu naudojant baigtinių skirtumų schemą. Abu algortimai buvo ištestuoti naudojant prieinamus MIF superkompiuterio resursus, lygiagretusis algortimas buvo testuojamas naudojant iki 256 branduolių. Naudojant lygiagretųjį algortimą, buvo išspręstas uždavinys kurio dėl sistemų resursų apribojimų nebuvo įmanoma išspręsti nuosekliuoju algoritmu. Buvo pasiektas lygiagrečiojo algortimo pagreitėjimas iki 95 kartų, o atlikta pagreitėjimo ir efektyvumo analizė parodė kad naudojant didesnį branduolių kiekį būtų galima dar labiau sutrumpinti skaičiavimo laiką.

Norint gauti geresnį lygiagretaus algoritmo pagreitėjimo ir efektyvumo rezultatus, būtų verta pabandyti kitokius duomenų dalinimo būdus, pvz kvadratais.

Literatūra

[Amd67] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. *Proceedings of the April 18-20, 1967, spring joint computer conference*, p. 483–485, 1967.

[BE01] Fredrik Berntsson ir Lars Eldén. Numerical solution of a cauchy problem for the laplace equation. *Inverse Problems*, 17(4):839, 2001.

[BF11] Richard L. Burden ir J. Douglas Faires. *Numerical analysis*. Cengage learning, 2011. 714 psl.

[Bla96] Richard J Blakely. *Potential theory in gravity and magnetic applications*. Cambridge university press, 1996.

[EZL89] Derek L Eager, John Zahorjan ir Edward D Lazowska. Speedup versus efficiency in parallel systems. *IEEE transactions on computers*, 38(3), 1989.

[Hou08] MG House. Analytic model for electrostatic fields in surface-electrode ion traps. *Physical Review A*, 78(3):033402, 2008.

[YM15] Gangjoon Yoon ir Chohong Min. Analyses on the finite difference method by gibou et al. for poisson equation. *Journal of Computational Physics*, 280:184–194, 2015.

[Kad85] Leo P Kadanoff. Simulating hydrodynamics: a pedestrian model. *Journal of statistical physics*, 39(3):267–283, 1985.