VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Šilumos laidumo uždavinio lygiagretinimo tyrimas naudojant centrinius ir grafinius procesorius

Steady-state heat equasion parallization analysis on CPU and GPU

Bakalauro baigiamasis darbas

Atliko:	Mantas Petrikas	(parašas)
Darbo vadovas:	dr. Rokas Astrauskas	(parašas)
Darbo recenzentas:	lekt. Irus Grinis	(parašas)

Vilnius – 2022

Santrauka

Raktiniai žodžiai: raktinis žodis 1, raktinis žodis 2, raktinis žodis 3, raktinis žodis 4, raktinis žodis 5

Summary

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

TURINYS

ĮVADAS 4

1. LYGIAGRETUS SKAIČIAVIMAMS NAUDOJAMA APRATINĖ ĮRANGA 6
   1.1. Centriniai procesoriai 6
   1.2. Grafiniai procesoriai 6
   1.3. Vartotojų programuojamos loginių elementų matricos 6
   1.4. Programos specifiniai grandynų grandinės 6

2. ŠILUMOS LAIDUMO UŽDAVINYS 7

3. ŠILUMOS UŽDAVINIO SPRENDIMAS 11
   3.1. Programos testavimo aplinka 11

4. NUOSEKLUSIS ALGORITMAS 12

5. LYGIAGRETUSIS ALGORITMAS, NAUDOJANTIS CENTRINIUS PROCESORIUS 13
   5.1. Duomenų dalinimas eilutėmis 15
   5.2. Lygiagretaus algoritmo, kai duomenys padalinimo eilutėmis, analyzė 17

6. LYGIAGRETUSIS ALGORITMAS, NAUDOJANTIS GRAFINIUS PROCESORIUS 21

REZULTATAI 22

IŠVADOS 23

Įvadas

Kompiuterinės simuliacijos yra svarbi modernių tyrimų dalis. Norint elektyviai atlikti sudėtingas simuliacijas, dažnai neužtenka vieno kompiuterio branduolio resursų, ir reikia paskirstyti užduoties sprendimo darbą kelioms procesoriaus gijoms, kurios sugebėtų vienu metu, lygiagrečiai atlikti skaičiavimus. Tai galima padaryti išnaudojant centriniam procesoriuje esančius branduolius ir gijas, naudojant grafinius procesorių, ar pasitekiant 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 pasiskirstymą tam tikrame apribotame plote, žinant ploto kraštinių taškų temperatūrą. Norint efektyviau apskaičiuoti temperatūros pasiskirstymą plote, galima pradinį plotą suskirstyti į dalis, ir vykdyti temperatūros skaičiavimus atskirtuose plotuose lygiagrečiai. Šis darbas nagrinėja šilumos lygties sprendimo galimybes, naudojant centrinius procesorius (ang. CPU) ir grafinius (ang. GPU) lygiagrečiam lygties sprędimui apribotame plote, ir palygina lygiagrečiųjų algoritmų pagreitėjimą ir efektyvumą.

Darbo tikslas - įvertinti ir palyginti šilumos laidumo uždavinio lygiagretinimo algoritmo efektyvumą naudojant centrinius ir grafinius procesorius, siekiant įvertinti grafinių procesorių efektyvumą sprendžiant dalinių diferencialinių lygčių uždavinius.

Darbo uždaviniai:

- implementuoti nuoseklųjį algoritmą, sprendžianti šilumos pasiskirstymo uždavinį
- implementuoti nuseklu ir įvertinti šilumos laidumo uždavinio algoritmo pagreitėjimą naudojant centrinius procesorius
- suprojektuoti ir implementuoti šilumos laidumo uždavinio sprendimo algorimtą, naudojantį grafinių procesorių resursus
- įvertinti grafinius procesorius naudojančio algoritmo našumą ir praktiškumą lyginant su centrinius procesorius naudojančiu algoritmu
- palyginti gautus rezultatus rezultatus su kitais panašiais problemas nagrinėjančių mokslinių darbų rezultatais

Darbo rezultatai:

- Nuoskelioji šilumos laidumo uždavinio sprendimo implementacija
- Lygiagrečioji šilumos laidumo uždavinio algoritmo sprendimo naudojanti centrinius procesorius
- Lygiagrečioji laidumo uždavinio sprendimo implementacija naudojanti grafinius procesorius
- Algoritmų teorinių ir praktinių pagreitėjimų analyzė
- Grafinius ir centrinius procesorius naudojančių algoritmų pagreitėjimų palyginimas

1. Lygiagretus skaičiavimams naudojama apratinė įranga

Šiame skyriuje apžvelgsime įvairius skaičiavimams naudojamus įrenginius,

1.1. Centriniai procesoriai

Centriniai procesoriai (ang. central processing unit -CPU) - yra bendri procesorius, skirtas įvairioms užduotims atlikti. Jie leidžia naudoti

1.2. Grafiniai procesoriai

Grafiniai procesoriai (ang. graphic processing unit - GPU) yra specializuotas įrenginys su patobulintomis matematinio skaičiavimo galimybėmis. Lyginant su centriniais procesoriais, šie grafiniai procesoriai turi didesnis slankaus kablelio skaičiavimo galimybes, ir tai leidžia vienu metu vykdyti skaičiavimus didesniams duomenų kiekiui. Dažnas grafinių procesorių naudojimo atvejis kompiuterinės grafikos, matematiniai skaičiavimai ir mašininio mokymosi užduotys. TODO: minusiai

1.3. Vartotojų programuojamos loginių elementų matricos

Vartotojų programuojamos loginių elementų matricos (ang. Field Programmable Gate Array - FPGA). The Field Programmable Gate Array (FPGA) is also a silicon based semiconductor, but it is based on a matrix of configurable logic blocks (CLB) that are connected by programmable interconnects.

1.4. Programos specifiniai grandynų grandinės

Programos specifiniai grandynų grandinės - (ang. Application-Specific Integrated Circuit - ASIC) yra procesoriai skirtos atlikti vienai specifiniai užduočiai. Po procesoriaus pagaminimo, jo nebegalima perprogramuoti, todėl jų projektavimui ir testavimui dažniausiai naudojami kitokia aparatinė įranga, dažnai dėl panašumo - FPGA. ASIC tipo lustai pasižymi geriausia greitaveika ir mažiausiais energijos sanaudomis, bet turi didelius proketavimo kaštus, ir gaminimo kaštus, todėl dažniausiai naudojami industrinėse sferose.

2. Šilumos laidumo uždavinys

Šio darbo tyrimo objektas - šilumos laidumo lygtis [BF11] sprendimo būdai.

Šilumos pasiskirstymą erdvėje aprašo Puasono lygtis dvimatei erdvei:

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \tag{1}$$

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

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

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 pasiskirstymą kvadratinėje erdvėje, kurią suskirstysime į vienodo dydžio kvadratinius taškus, turinčius savo temperatūrą.

Naudodami baigtinių skirtumų metodą galime apibrėžti antros eilės dalinės diferencialinės lygties apytikslią reikšmes:

$$\frac{\partial^2 u}{\partial x^2}(x,y) \approx \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} \tag{3}$$

$$\frac{\partial^2 u}{\partial y^2}(x,y) \approx \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} \tag{4}$$

kur h yra atstumas tarp taškų, o $u_{x,y}$ - taško temperatūra taške, x,y. Įstatydami šias reikėmes į Laplaso lygtį gauname:

$$\frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} + \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} = 0 \tag{5}$$

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} = 0 \tag{6}$$

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h)}{h^2} = \frac{4u(x,y)}{h^2} \tag{7}$$

Kadangi atstumas šiame darbe nagrinėjame uždavinio taškų nėra lygus 0, galime lygties abi puses padauginti iš h^2:

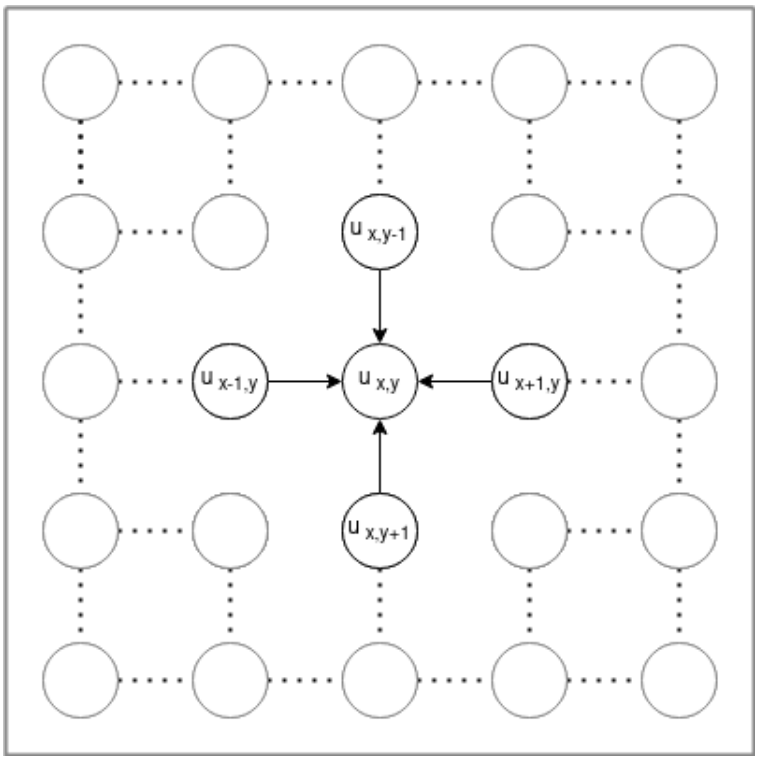
$$u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) = 4u(x,y) \tag{8}$$

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h)}{4} = u(x,y) \tag{9}$$

Šia formulė galime pritaikyti praktiškai, dvimatę erdvę projektuodami kaip matrica, padalinę ją į atstumo taškus:

$$u_{x,y} = \frac{u_{x+1,y} + u_{x-1,y} + u_{x,y+1} + u_{x,y-1}}{4} \tag{10}$$

Gauname, kad vidinių taško temperatūra yra lygį jį supančių 4 taškų temperatūrų vidurkiui (1 paveikslėlis).



1 pav. Temperatūros pasiskirstymas vidiniuose matricos taškuose

Norėdami išspręsti šias lygčių sistemas, naudosime Jakobi iteracinį metodą [BF11], kiekvienos iteracijos metu formulės kintamųjų reiškmes 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 pasiskirstymas nulemia galutinį temperatūros pasiskirstymą erdvėje. Vidinių taškų pradinis temperatūros pasiskirstymas įtakos galiniam temperatūros pasiskirstymui neturi, tačiau gali turėti įtakos konvergencijos greičiui. Kraštinių taškų temperatūra šito tyrimo metu aprašomą kaip:

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

,

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 priskiriamos temperatūros reikšmės nuo 0 iki 1 imtinai, taip leidžiant modeliuoti į realia objekto temperatūra taikant paprasta proporcija. Vizualiai pradinis temperatūrų pasiskirstymas matomas 2 paveikslėlyje. Juodi taškai atitinka 1 reikšmę, balti - 0. Uždavinio tikslas - nustatyti galutinį temperatūros pasiskirstymą 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 pasiskirstymas matomas 3 paveikslėlyje. Laikas per kurį temperatūra pasiskirsto erdvėje, erdvė sudarančios medžiagų šiluminis laidumas šiame darbe nėra nagrinėjami.

2 pav. Pradinis temperatūros pasiskirstymas

3 pav. Galutinis temperatūros pasiskirstymas plokštelėje

3. Šilumos uždavinio sprendimas

Šiame skyriuje nagrinėjamas programinis šilumos uždavinio sprendimas.

3.1. Programos testavimo aplinka

Siekiant išlaikyti vienodas sąlygas ir ištestuoti algoritmą turint didelį centinių procesorių kiekį, visi šiame darbe aprašyti eksperimentai buvo vykdomi Vilniaus Universiteto Matematikos 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 procesorius, kurių kiekvienas turi po 6 branduolius. Kiekviename mazge yra 24 GB operatyviosios atminties (ang. RAM) ir jie turi prieigą prie 20Gbit/s infiniband tinklo. Skaičiavimų tinkle buvo instaliuota Debian operacinė sistemą, lygiagrečiam algoritmui buvo naudojamas OpenMPI bibliotekos 2.1.0 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.

Matricos kraštinė	Vykdymo laikas (s)	Iteracijų skaičius	10000 iteracijų vykdymo laikas
128	3.747	17000	2.204
256	42.596	49000	8.693
512	432.985	122000	35.491
1024	2155.811	156000	138.193
2048	> 7200	155000	543.031

1 lentelė. Nuoseklaus algoritmo vykdymo laikas

4. 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 vienmačiai masyvai, nes didžioji dalis OpenMPI bibliotekos funkcijų, kuri buvo naudotos lygiagretinant algoritmą, tikisi vienmačių 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 iteracijos 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 pasiskirstymą.

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 padidėja beveik 4 kartus, nes algoritmas nagrinėja šilumos pasiskirstymą dvimatinėje erdvėje, todėl padidinus matricos kraštinės ilgį 2 kartus, duomenų kiekis padidėja 4 kartus. Matricos kraštinės ilgiai pasirinkti dvejeto laipsniai, kad lygiagretinant algoritmą vidinės matricos reikšmes būtų padalinti skirtingiems procesams vienodais kiekiais. Į ši skaičių neieina papildomos dvi eilutės, kurios šiame skaičiavime atitinka 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).

5. Lygiagretusis algoritmas, naudojantis centrinius procesorius

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

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

kur p yra procesorių 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 žingsniai.

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 lygiagretinamas 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 1048 taškų, nuoseklaus algoritmo konfiguracijos nuskaitymas vidutiniškai truko 0,002 sekundes (žymėsime t_{np}), kai tuo

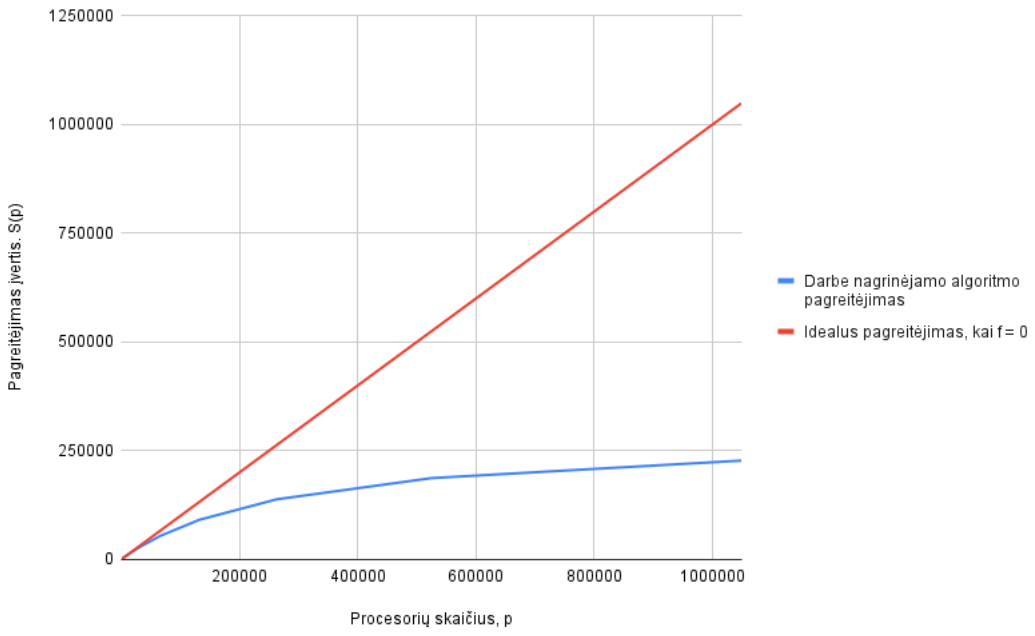
metu likusieji skaičiavimai vidutiniškai užtruko 577,363 sekundes. Toks laiko santykis reikia kad nelygiagretinama algoritmo dalis yra:

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

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

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

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



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

Maksimalaus pagreitėjimo reiškmes naudojant skirtus procesorių kiekius matomos 4 grafike. Šio darbo ribose algoritmas 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 algoritmams 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:

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

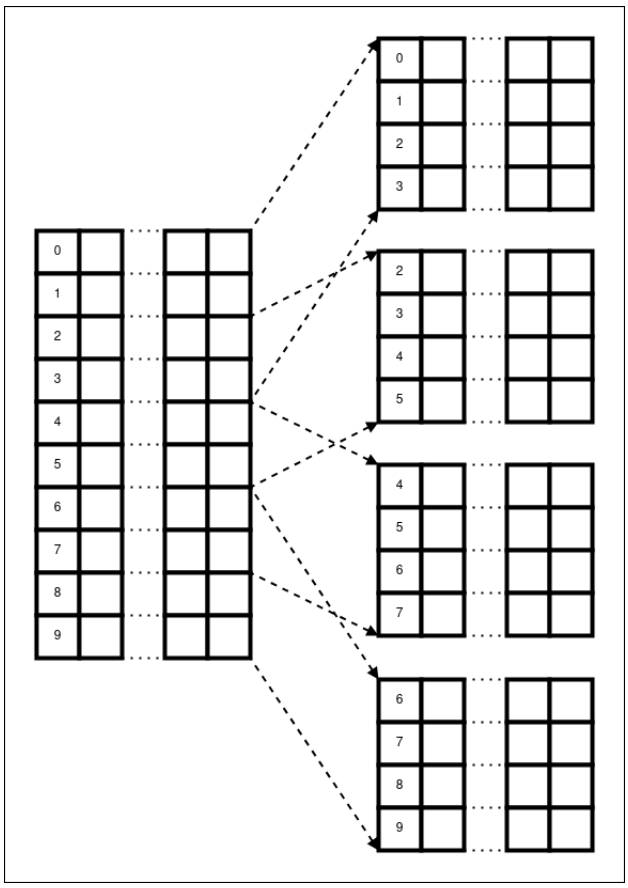
Taip pat efektyvumą galime išreikšti per algoritmo pagreitėjimą:

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

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

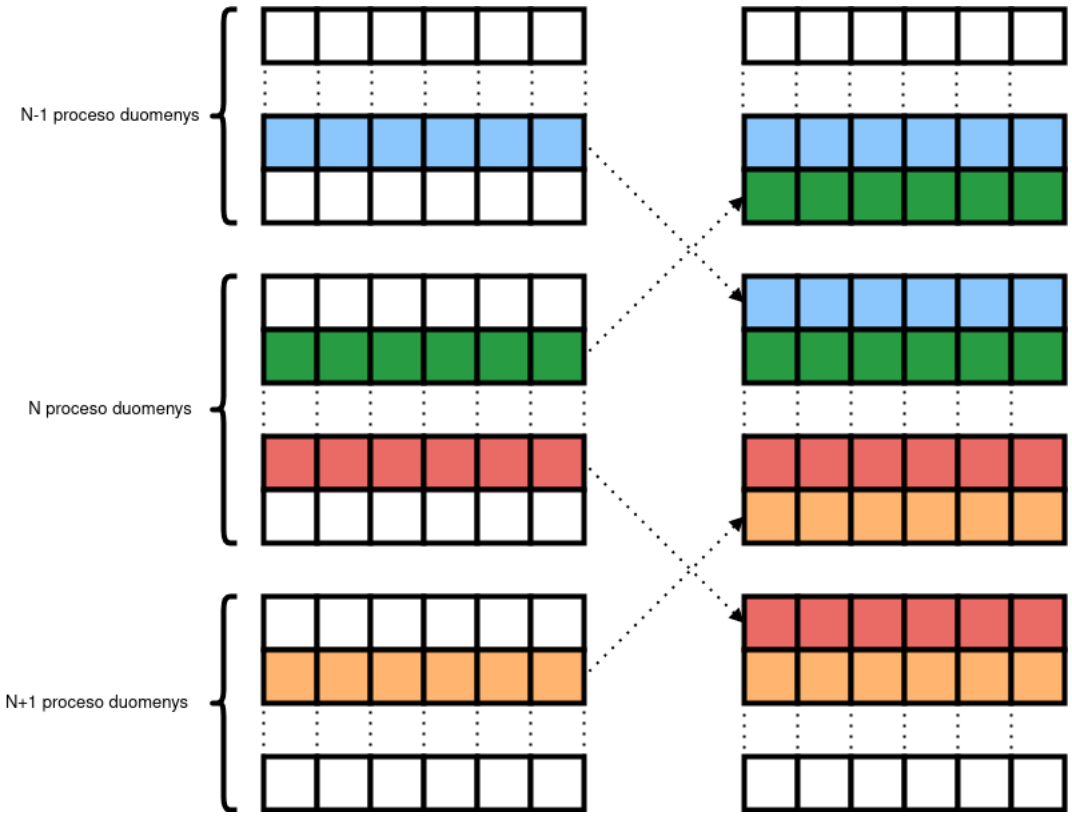
5.1. Duomenų dalinimas eilutėmis

Viena iš duomenų paskirstymo lygiagrečiajam algoritmui strategijų yra duomenis paskirstyti 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 eilučių 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 reikšmės sinchronizuoja tik su vienu iš procesu, nes viena iš jas sudarančių matricos eilučių atitinka pradinės matricos kraštines eilutes, turinčias pastovias reikšmes.



6 pav. Duomenų sinchronizacija 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 pagrindiniam procesoriui, kuris gavęs visas reikšmes, įvertiną ar bent vienas procesoriaus lokalus skirtumas viršija norima paklaidą, ir nusprendžia ar tęsti iteravimą. Norint išvengti visuotinio sinchronizavimo tarp visų branduolių ir pagrindinio proceso kiekvienos iteracijos metu, maksimalaus pokyčio reikšmių siuntimas ir palyginimas vykdomas tik kas 100 iteracijų.

5.2. Lygiagretaus algoritmo, kai duomenys padalinimo eilutėmis, analyzė

Lygiagretaus algoritmo veikimo laiko įverčiai, kai matricos kraštinę sudaro 1048 taškų pateikiami 2 lentelėje. Prieš testuojant lygiagretųjį 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 būsenoje (ang. Hibernation) ir būtų pasiruošę vykdyti algoritmą.

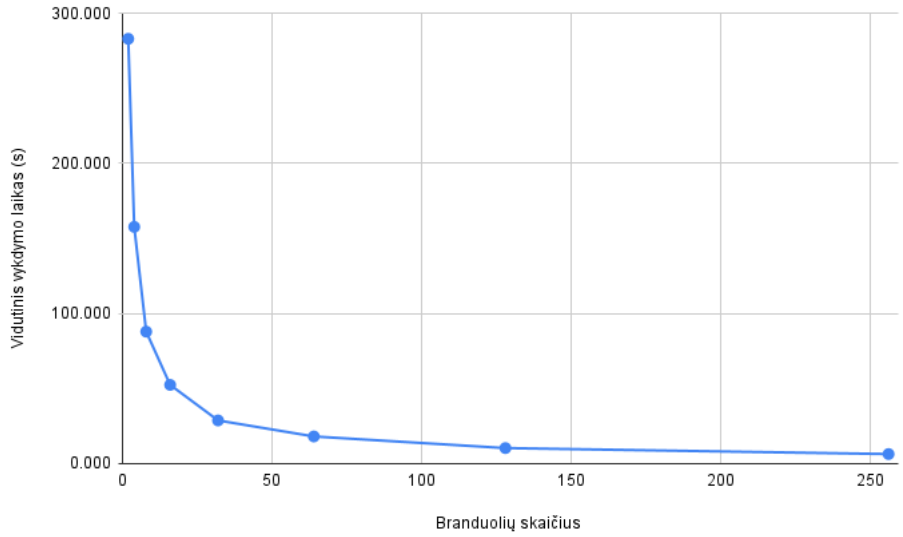
Kaip matoma 7 grafike, vykdymo laikas atvirkščiai priklauso nuo naudojamų branduolių skaičiaus.

Branduolių skaičius	Vykdymo laikas (s)
128	4027.416
256	2236.912

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

Branduolių skaičius	Vykdymo laikas (s)
2	1104.232
4	571.615
8	318.093
16	189.300
32	106.901
64	60.894
128	36.370
256	21.911

2 lentelė. Lygiagretaus algoritmo vykdymo laikas, kai matricos kraštinę sudaro 1048 taškų

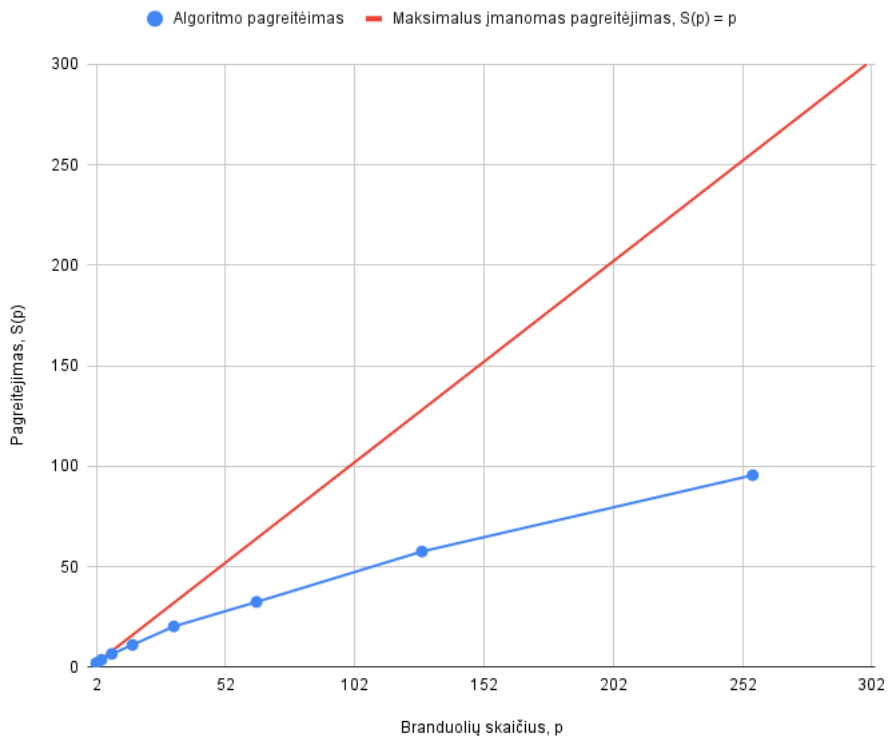


7 pav. Lygiagretaus algoritmo vykdymo laikas priklausomybė nuo naudojamų branduolių skaičiaus, kai matricos kraštinę sudaro 1048 taškų

Taip pat lygiagretus algoritmas 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	Pagreitėjimas, $S(p)$	Efektyvumas, $E(p) = \frac{S(p)}{p}$
1	2155.811	1	1
2	1104.232	1.952	0.98
4	571.615	3.771	0.94
8	318.093	6.777	0.85
16	189.300	11.388	0.71
32	106.901	20.166	0.63
64	60.894	35.402	0.55
128	36.370	59.275	0.46
256	21.911	98.390	0.38

4 lentelė. Lygiagretaus algoritmo pagreitėjimas ir efektyvumas.



8 pav. Lygiagretaus algoritmo pagreitėjimo priklausomybė nuo naudojamų branduolių skaičiaus, kai matricos kraštinę sudaro 1048 taškų

8 grafike matomas lygiagretaus algoritmo logaritminis 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 neatsižvelgia į komunikacijos tarp skirtingų procesorių kaštus. Tikslų programos skaičiavimų ir komunikacijos santykį įvertinti yra sunku, nes komunikacijos laikas gali skirtis laikas priklausomai 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 algoritmo efektyvumas naudojant 2 ir 4 branduo-

lius 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 atskirų procesų. Taip pat, 4 lentelėje matomas algoritmo 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 algoritmo efektyvumas mažėja, lygiagretaus algoritmo pagreitėjimo metrika didinant naudojamų branduolių nenustoja didėti, iš to galima daryti prielaidą šių testų metu nebuvo rastas branduolių skaičius, kurį naudojant komunikacijos kaštai nusvertų gautą algoritmo pagreitėjimą, kai užduotis yra padalinama į mažesnes dalis.

6. Lygiagretusis algoritmas, naudojantis grafinius procesorius

Tesla V100-SXM2 grafinis procesorius, turintis 32GB atminties

Rezultatai

Šio darbo rašymo metu buvo implementuoti nuoseklusis ir lygiagretusis algoritmai, spendžiantys šilumos pasiskirstymo užduotį Jacobi iteraciniu metodu naudojant baigtinių skirtumų schemą. Abu algoritmai buvo ištestuoti naudojant prieinamus MIF superkompiuterio resursus, lygiagretusis algoritmas buvo testuojamas naudojant iki 256 branduolių. Naudojant lygiagretųjį algoritmą, buvo išspręstas uždavinys, kurio dėl sistemų resursų apribojimų nebuvo įmanoma išspręsti nuosekliuoju algoritmu. Buvo pasiektas lygiagrečiojo algoritmo pagreitėjimas iki 95 kartų, o atlikta pagreitėjimo ir efektyvumo analizė parodė kad naudojant didesnį branduolių kiekį būtų galima dar labiau sutrumpinti skaičiavimo laiką. Sukurta programa, įgyvendinanti algoritmo sprendimą, yra konfigūruojama, leidžianti lengvai keisti pradines sąlygas ir ją panaudoti kitų praktinių uždavinių sprendimams.

Ateityje, norint gauti geresnį lygiagretaus algoritmo pagreitėjimo ir efektyvumo rezultatus, būtų verta pabandyti kitokius duomenų dalinimo būdus, (pvz kvadratais), pabandyti lygiagretizuoti sprendimą naudojant grafinius procesorius.

Išvados

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.