VILNIAUS UNIVERSITETAS

Aleksandr Igumenov

LYGIAGRETIEJI IR PASKIRSTYTIEJI ELEKTROS ENERGIJĄ TAUSOJANTYS SKAIČIAVIMAI

Daktaro disertacija

Technologijos mokslai, informatikos inžinerija (07T)

Vilnius, 2012

Disertacija rengta 2007–2012 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinis vadovas

prof. dr. Julius ŽILINSKAS (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07T).

Padėka

Nuoširdžiai dėkoju darbo vadovui prof. dr. J. Žilinskui už atkaklų ir nuoseklų vadovavimą, vertingas mokslines konsultacijas, visokeriopą pagalbą ir kantrybę man studijuojant doktorantūroje bei rengiant disertaciją. Dėkoju prof. habil. dr. G. Dzemydai ir doc. dr. T. Petkui už patarimus tęsti magistratūroje pradėtas studijas, pasiūlymus ir bendradarbiavimą.

Esu dėkingas disertacijos recenzentams prof. dr. D. Navakauskui ir dr. V. Medvedevui atidžiai perskaičiusiems disertaciją ir pateikusiems vertingų pastabų bei patarimų, padėjusiems pagerinti šio darbo kokybę.

Noriu ypač padėkoti dr. V. Medvedevui už jo pastangas, kantrybę ir konsultacijas ne tik disertacijos rengimo bet ir visų doktorantūros studijų metu.

Ačiū Vilniaus universiteto Matematikos ir informatikos instituto Sistemų analizės skyriaus kolegoms už kritiką ir draugišką pagalbą rengiant disertaciją.

Nuoširdžiai dėkoju savo artimiesiems ir draugams už jų paramą, moralinį palaikymą, kantrybę ir supratingumą.

Dėkoju Lietuvos valstybiniam studijų bei Lietuvos valstybiniam mokslo ir studijų fondams už suteiktą finansinę paramą disertacijos rengimo metu.

Dėkoju visiems, kurie tiesiogiai ar netiesiogiai prisidėjo prie šio darbo.

Aleksandr Igumenov

Reziumė

Pastaruoju metu pasaulis susiduria su problema – elektros energijos suvartojimo augimas. Tai yra ne vienintelė problema. Žmonija labai neatsargiai naudojo ir naudoja žemės neatsistatančius resursus. Kaip pavyzdį galima paminėti elektros energijos gamybą. Yra keli būdai skirti elektros energijos gamybai: šiluminės elektrinės, atominės elektrinės, hidroelektrinės, vėjo jėgainės ir saulės baterijos. Iš visų paminėtų elektros energijos gavybos būdų atsistatančiuosius resursus naudoja tik vėjo jėgainės, hidroelektrinės ir saulės baterijos. Visi kiti elektros energijos gamybos būdai yra ne tik neatsistatančius resursus eikvojantys, bet ir pavojingi bei teršiantys aplinką. Dėl to reikia mažinti įrenginių elektros sąnaudas tam, kad mažėtų elektros energijos gamybos mastai ir kad elektros energijos gamybos būdai mažiau įtakotų žmonių aplinką.

Elektros energijos suvartojimas liečia ir lygiagrečiuosius bei paskirstytuosius skaičiavimus. Pasaulyje kiekvienais metais didėja superkompiuterių, klasterių, gridų bei debesų kompiuterių skaičius. Visi šie resursai reikalauja daug elektros energijos. Pats galingiausias pasaulyje kompiuteris, „K computer“, veikiantis pilnu pajėgumu, reikalauja 12 659,89 kW elektros galios. Tokios elektros energijos galios užtektų išlaikyti vidutiniškai apie 60 000 įprastų namų ūkių.

Siekiant atkreipti dėmesį į kompiuterių suvartojamą elektros energiją atsirado – „žalioji kompiuterija“. Žalioji kompiuterija – mokslas bei praktiniai patarimai apie tai, kaip reikia projektuoti, gaminti, naudoti, utilizuoti kompiuterius, serverius ir su jais susietus įrenginius, kad jie veiktų kuo efektyviau, minimaliai įtakojant arba visiškai neįtakojant aplinkos (Murugesan 2008).

Disertacijoje pristatoma žaliosios kompiuterijos tema, kuriai Lietuvoje skiriamas nepakankamas dėmesys. Pristatomi lygiagretieji skaičiavimai,

siūlomas kombinatorinio optimizavimo šakų ir rėžių algoritmo taikymas, skirtas strypinių konstrukcijų optimizavimui, apžvelgiamos gridų ir klasterių žaliosios kompiuterijos problemos. Disertacijoje siūlomas būdas, skirtas paskirstytųjų skaičiavimų vykdymui gride, kai nėra galimybės naudoti MPI bibliotekas. Be to, siūlomas ir eksperimentiškai ištirtas vienas iš būdų, skirtų lygiagrečiųjų kompiuterių elektros energijos suvartojimui mažinti.

Abstract

Recently, the world has faced the problem – growing of electric energy consumption. It is not the only one problem. The human race used and uses non-regenerating land resources very carelessly. A very god example is the production of electricity. There are several methods of the production of electricity: thermal power plants, nuclear power plants, hydropower plants, wind power plants, and solar batteries. And only three of them use regenerating resources, namely only wind power, hydropower plants, and solar batteries. All the other electric power generation methods use not only non-regenerating resources, but also polluting and dangerous to the environment. It is necessary to reduce the power consumption of equipment. To this end, we need to reduce the scale of power generation and make electric power generation less influential on people and environment.

Electric power consumption is also applied to parallel and distributed computing. The number of supercomputers, clusters, grid and cloud computers is growing up in the world every year. All of these resources require a lot of electricity. Same applies to the most powerful computer in the world, "K computer". It requires 12 659.89 kW of electricity when running at full power. That much power would be sufficient to maintain about 60 000 usual households. As a result, the term green computing has emerged. San Murugesan defines the field of green computing as “the study and practice of designing, manufacturing, using, and disposing of computers, servers and associated subsystems – such as monitors, printers, storage devices and networking and communications systems – efficiently and effectively with minimal or no impact on the environment” (Murugesan 2008).

The dissertation presents the green computing theme, which has not got much attention in Lithuania. Here parallel computing is presented, a combinatorial optimization branch and bound algorithm is proposed for the optimization of truss structures, and grid computing and clusters are

overviewed. A method for parallel execution of tasks on grids, where there is no access to the MPI, is proposed in the dissertation. As well in addition, a way to reduce power consumption for parallel computers is proposed and experimentally investigated.

Žymėjimai

Simboliai

A_i	Strypo skerspjūvio plotas
D_i	Leistinųjų sprendinių aibės poaibis
L_i	Strypo ilgis konstrukcijoje
D^*	Optimalių sprendinių aibė
x_i	Kintamasis
ρ_i	Strypo medžiagos tankis
$h(X)$	Apribojimų funkcija, įvertinanti įtempimus mazge
D	Leistinųjų sprendinių aibė
X	Kintamųjų vektorius
$f(X)$	Tikslo funkcija
$g(X)$	Apribojimų funkcija, įvertinanti konstrukcijos pusiausviros sąlygas
m	Mazgų skaičius
n	Visų galimų strypų skaičius konstrukcijoje

Santrumpos ir pavadinimai

80 PLUS	Kompiuterių maitinimo blokų elektros energijos suvartojimo vertinimo standartas
API	Aplikacijų programavimo sąsaja, angl. – *Application Programming Interface*
Bash	Linux OS komandinės eilutes interpretatorius
BIOS	Bazinė įvesties/išvesties sistema, angl. – *Basic Input/Output System* – sisteminė programa, kuri saugoma pastovios atminties mikroschemoje, skirta testavimui, derinimui, bei kompiuterio įkrovos proceso kontrolei
CA	Grid tinklo resursų ir naudotojų identifikavimo sertifikatų tarnyba, angl. – *Certificate Authority service*
CPU	Centrinis procesorius, angl. – *Central Processing Unit*
CRON	Reguliarius veiksmus vykdanti Linux OS tarnyba
VĮ, CU	Valdymo įtaisas, ang. – *Control Unit*
CUDA	Technologija bei programavimo standartas, skirti lygiagretiesiems skaičiavimams atlikti su GPU, angl. – *Compute Unified Device Architecture*
DC	Daugiamatės skalės
DirectX	API rinkinys, skirtas multimedijos užduočių tvarkymui
DKDD	Daugelio komandų daugelio duomenų srautų kompiuteris
DKVD	Daugelio komandų vieno Duomenų srauto kompiuteris
Energy Star	Elektros energijos suvartojimo vertinimo standartas
GPU	Grafikos apdorojimo įrenginys, angl. – *Graphics Processing Unit*
Green500	Ekologiškiausių ir greičiausių superkompiuterių sąrašas
ICT	Informacinės komunikacinės technologijos, angl. – *Information Comunication Technologies*
IT	Informacinės technologijos, angl. – *Information Technologies*
LCD	Skystųjų kristalų vaizduoklis, angl. – *Liquid crystal display*
LED	Šviesos diodas, angl. – *Light-Emitting Diode*

LB	Apatinis rėžis, angl. – *Lover Bound*
MAC	Fizinis kompiuterio tinklo plokštės adresas
MII	Matematikos ir informatikos institutas
OS	Operacinė sistema, angl. – *Operating Sistem*
SSD	Puslaidininkinis informacijos įrašymo ir nuskaitymo įrenginys (diskas), angl. – *Solid State Drive*
TFT	Plonas plėvelinis tranzistorius, angl. – *Thin Film Transistor*
UB	Viršutinis rėžis, angl. – *Upper Bound*
Top500	Greičiausių superkompiuterių sąrašas
VKDD	Vienos komandos daugelio duomenų srautų kompiuteris
VKVD	Vienos komandos vieno duomenų srauto kompiuteris
VO	Grid tinklo virtualioji organizacija – viename projekte dirbančių asmenų grupė
VU	Vilniaus universitetas
Web 2.0	Antrosios kartos saitynas

Turinys

1

Įvadas

1.1. Tyrimų sritis ir problemos aktualumas

Elektros energijos suvartojimo didėjimas kelia problemas, susijusias su elektros energijos gamyba ir importu (kai neužtenka vidinių šalies resursų). Dažniausiai naujos technologijos reikalauja vis didesnių elektros energijos kiekių. Tai galima pastebėti visur, bet ypač ši problema išryškėja kalbant apie šiuolaikines informacines technologijas. Išleidžiami vis naujesni įrenginiai, kurie neimlūs elektros energijai, tokių įrenginių skaičius didėja, o tai didina elektros energijos suvartojimą. Suvartojimo didėjimas liečia ir lygiagrečiuosius bei paskirstytuosius skaičiavimus. Pasaulyje kiekvienais metais didėja superkompiuterių, klasterių, gridų bei debesų kompiuterių skaičius. Visi šie resursai reikalauja daug elektros energijos. Pats galingiausiais pasaulyje kompiuteris, „K computer“, veikiantis pilnu pajėgumu, reikalauja 12 659,89 kW elektros galios. Tokios elektros galios užtektų išlaikyti vidutiniškai apie 60 000 įprastų namų ūkių.

Lygiagretieji skaičiavimai per pastaruosius 30 metų labai išsivystė. Atsirado įvairios technologijos įgyvendinančios lygiagrečiuosius skaičiavimus. Visos technologijos yra skirtingos ir reikalauja iš šiuolaikinio mokslininko atitinkamų žinių. Mokslininkai gali naudoti skirtingą techniką: superkompiuterius, klasterines sistemas, gridus (skaičiuojamuosius tinklus), debesų kompiuterius ir kitus resursus. Priklausomai nuo techninės skaičiavimo resursų realizacijos gali skirtis ir su jais teikiamos programinės technologijos. Taip atsiranda naujos bibliotekos skirtos atlikti lygiagrečiuosius skaičiavimus, taip pat nauji programavimo standartai. Tarp tokių galima paminėti MPI, OpenMP, gLite, XMPP-MPI, CUDA, OpenCL ir kiti. O tai reiškia, kad šiuolaikinis mokslininkas turi mokėti teisingai realizuoti savo algoritmus su viena ar keliomis technologijomis.

Kalbant apie lygiagrečiąsias technologijas, dažniausiai turima omeny daug kompiuterių, sujungtų kompiuteriniu tinklu į vieną skaičiavimo resursą. Jei toks skaičiavimo resursas nedidelis, tai ir jo pajėgumai nedideli. Jei auga pajėgumai, auga ir kompiuterių skaičius, iš kurių susideda skaičiavimų resursas. Tai įtakoja ir elektros energijos suvartojimą. Šiuo metu tai yra didelė problema, reikalaujanti dėmesio iš mokslininkų bei kompiuterinės įrangos gamintojų. Taip 2005 metais atsirado terminas, įvardinęs šią problemą – „green computing“ (liet. – *žalioji kompiuterija*). Pirmas terminą įvedė San Murugesan, pateikęs tokį apibrėžimą: žalioji kompiuterija yra mokslas bei praktiniai patarimai apie tai, kaip reikia projektuoti, gaminti, naudoti ir utilizuoti kompiuterius ir serverius, kad jie veiktų kuo efektyviau, minimaliai įtakojant arba visiškai neįtakojant aplinkos (Murugesan 2008). Dabartiniu metu įvairių autorių straipsniuose galima sutikti ir kitokius šios problemos įvardinimus: žalios informacinės technologijos (žaliosios IT), aplinką tausojantys skaičiavimai ir IT. Šią problemą akcentuoja ir sąrašas Green500.com (egzistuoja nuo 2005 metų). Optimizavimo metodai gali padėti sprendžiant elektros energijos suvartojimo mažinimo uždavinius. Be to, įvairių uždavinių

sprendimui nebūtina naudoti visų prieinamų skaičiavimo resursų, galima panaudoti tik jų dalį, siekiant mažinti jų elektros energijos suvartojimą. Šį faktą pabrėžia efektyvios žaliosios kompiuterijos posakis – „kuo didesnis kompiuterių skaičiavimo pajėgumas su kuo mažesne įtaka aplinkai“.

1.2. Tyrimo objektas

Disertacijos tyrimo objektas yra technologijos, mažinančios kompiuterių elektros energijos sąnaudas, sprendžiant optimizavimo uždavinius, pasitelkiant lygiagrečiuosius ir paskirstytuosius skaičiavimus.

1.3. Darbo tikslas ir uždaviniai

Darbo tikslas – pasiūlyti elektros energijos suvartojimą mažinančius būdus, skirtus lygiagretiesiems ir paskirstytiesiems skaičiavimams.

Siekiant iškelto tikslo buvo sprendžiami šie uždaviniai:

- analitiškai apžvelgti žaliosios kompiuterijos technologijas ir elektros energijos suvartojimo problemą;
- ištirti esamas lygiagrečiųjų ir paskirstytųjų skaičiavimų technologijas;
- pasiūlyti nuoseklųjį, lygiagretųjį ir paskirstytųjų strypinių konstrukcijų tikslaus optimizavimo algoritmą;
- ištirti lygiagrečiųjų ir paskirstytųjų skaičiavimų sistemų elektros energijos suvartojimą, sprendžiant lygiagrečiųjų skaičiavimų ir optimizavimo uždavinius;
- pasiūlyti ir sukurti programinę realizaciją, skirtą mažinti lygiagrečiųjų ir paskirstytųjų skaičiavimų sistemų elektros energijos suvartojimą.

1.4. Tyrimo metodika

Analizuojant mokslinius ir eksperimentinius pasiekimus lygiagrečiųjų skaičiavimų srityje naudoti informacijos paieškos, sisteminimo, analizės, lyginamosios analizės ir apibendrinimo metodai. Remiantis eksperimentinio tyrimo metodu, atlikta statistinė duomenų ir tyrimų rezultatų analizė, kurios rezultatams įvertinti naudotas apibendrinimo metodas.

1.5. Darbo mokslinis naujumas

1. Ištirtas šakų ir rėžių algoritmo taikymas strypinėms konstrukcijoms optimizuoti.
2. Eksperimentiškai ištirtas lygiagrečiųjų ir paskirstytųjų skaičiavimo sistemų elektros energijos suvartojimas sprendžiant optimizavimo uždavinius.
3. Pasiūlyta ir eksperimentiškai ištirta lygiagrečiųjų ir paskirstytųjų skaičiavimo sistemų darbo strategija, skirta elektros energijos suvartojimui mažinti.

1.6. Darbo rezultatų praktinė reikšmė

Tyrimai atlikti moksliniuose projektuose:

- Lietuvos Respublikos švietimo ir mokslo ministerijos programa „Lygiagrečių ir paskirstytų skaičiavimų ir e-paslaugų tinklas (LitGrid)“ 2007–2009 (Bagdonavičius ir Lapienis 2005);
- Lietuvos valstybinio mokslo ir studijų fondo Aukštųjų technologijų plėtros programos projektas B-03/2007 „Globalus sudėtingų sistemų optimizavimas naudojant didelio našumo skaičiavimus ir grid technologijas“ 2007–2009 (Čiegis *ir kt.* 2007-2009);

- Lietuvos mokslo tarybos finansuojamas mokslininkų iniciatyva vykdomas mokslinis tyrimas „Globaliojo optimizavimo algoritmai su simpleksiniais posričiais“ 2010–2011;
- COST veikla IC0805 “Open European Network for High Performance Computing on Complex Environments” nuo 2008 iki dabar.

Praktinė darbo reikšmė:

- Pasiūlyta lygiagrečiųjų ir paskirstytųjų skaičiavimo sistemų darbo strategija buvo ištirta ir pritaikyta HPC.MII.VU.LT klasteryje.
- Bendradarbiaujant su Lenkijos „*Poznan Supercomputing and Networking Center*“ (PSNC) mokslininkais buvo testuojamas ir šiuo metu tobulinamas naujas lygiagrečiųjų ir paskirstytųjų skaičiavimų standartas XMPP-MPI.

1.7. Ginamieji teiginiai

- Sukurta strategija leidžia sumažinti klasterių ir gridų elektros energijos suvartojimą iki 90 %.
- Šakų ir rėžių kombinatorinis algoritmas leidžia gauti optimalią strypinę konstrukciją iki 9 mazgų topologijos optimizavimo uždaviniams.

1.8. Darbo rezultatų aprobavimas

Tyrimų rezultatai publikuoti 5 moksliniuose straipsniuose: 4 periodiniuose recenzuojamuose mokslo žurnaluose, 1 konferencijos pranešimų medžiagoje. Tyrimų rezultatai buvo pristatyti ir aptarti 9 nacionalinėse ir tarptautinėse konferencijose Lietuvoje ir užsienyje:

1. **A. Igumenov**, J. Žilinskas. Parallel Combinatorial Optimization for Optimal Design of Truss Structures. *International Networking for*

Young Scientists on High Performance Scientific Computing INYS-2008, February 5-8, 2008, Druskininkai, Lithuania.

2. **A. Igumenov**, J. Žilinskas. Combinatorial Algorithms for Topology Optimization of Truss Structure. *Information Technologies 2009 (IT 2009) 15th International Conference on Information and Software Technologies*, April 23-24, 2009, Kaunas, Lithuania.
3. **A. Igumenov**, J. Žilinskas. Combinatorial Topology Optimization of Truss Structures using Distributed Grid Computing. *International Young Scientists Conference and Erasmus Staff Mobility Event 2010*, April 29-30, 2010, Šiauliai, Lithuania.
4. **A. Igumenov**, J. Žilinskas. Power Consumption in Parallel Computations Optimizing Topology of Truss Structures using Branch and Bound. *15th International Conference Mathematical Modelling and Analysis (MMA2010)*, May 26-29, 2010, Druskininkai, Lithuania.
5. **A. Igumenov**, J. Žilinskas. Optimization of Topology of Truss Structures using Grid Computing. *IEEE International Conference on Cluster Computing 2010 (IEEE Cluster 2010)*, September 20-24, 2010, Crete, Greece.
6. **A. Igumenov**, J. Žilinskas. Optimization of Power Consumption in Parallel Branch and Bound for Optimization of Truss Structures. *Operacijų tyrimai verslui ir socialinėms procesams, LOTD – 2010*, 2010 m. spalio 1 d., Vilnius, Lietuva.
7. **A. Igumenov**, J. Žilinskas. Energijos suvartojimas skaičiavimo klasteriuose ir XMPP kaip MPI alternatyva. *Fizinių ir technologijos mokslų tarpdalykiniai tyrimai (Pirmoji jaunųjų mokslininkų konferencija)*, LMA, 2011 m. vasario 8 d., Vilnius, Lietuva.

8. **A. Igumenov**, J. Žilinskas. Power Consumption in Parallel Computations Optimizing Topology of Truss Structures using Branch and Bound. *ComplexHPC Spring School 2011*, May 9-13, 2011, Amsterdam, Netherlands (Stendinis pranešimas).
9. **A. Igumenov**, J. Žilinskas. Elektros sąnaudų optimizavimas lygiagrečiuose skaičiavimuose. *Operacijų tyrimai versle, inžinerijoje ir informacinėse technologijose, LOTD – 2011*, 2011 m. rugsėjo 30 d., Vilnius, Lietuva.

Mokslinė stažuotė:

- Stažuotės tema: Implementation of parallel branch-and-bound algorithm using XMPP-MPI tool.
- Stažuotės vieta: Poznan Supercomputing and Networking Center, Poznanė, Lenkija.
- Stažuotės trukmė: nuo 2011-01-23 iki 2011-02-05.

1.9. Disertacijos struktūra

Disertaciją sudaro 5 skyriai ir literatūros sąrašas. Disertacijos skyriai: Įvadas, Elektros energijos suvartojimas bei lygiagretieji ir paskirstytieji skaičiavimai, Elektros energijos suvartojimo mažinimo būdai lygiagrečiųjų skaičiavimų klasteriams, lygiagretieji šakų ir rėžių algoritmai, Eksperimentiniai tyrimai, Bendrosios išvados. Papildomai disertacijoje pateikti naudotų žymėjimų ir santrumpų sąrašai. Disertacijos apimtis 114 puslapiai, kuriuose pateikti 22 paveikslai ir 14 lentelių. Disertacijoje remtasi 125 literatūros šaltiniais.

2

Elektros energijos suvartojimas bei lygiagretieji ir paskirstytieji skaičiavimai

Šiame skyriuje apžvelgiama dabartiniu metu pasaulyje aktuali problema – elektros energijos suvartojimas lygiagrečiuose skaičiavimuose. Pastaruoju metu pasaulyje vis dažniau projektuojami, kuriami ir surenkami kompiuteriai, skirti našiesiems skaičiavimams atlikti. Tokių kompiuterių ir iš jų sudarytų klasterių skaičius kiekvienais metais vis didėja, didėja ir jų elektros energijos suvartojimas. Šiuo metu greičiausias pasaulyje superkompiuteris pagal Top500 sąrašą – „K computer“, Green500 sąraše užima tik 32 vietą pagal santykį MFlop/s per W, o bendra jo elektros galia, kai jis veikia pilnu pajėgumu, yra 12 659,89 kW.

Šio skyriaus pradžioje yra supažindinama su kompiuterių architektūra, kuo skiriasi nuoseklieji kompiuteriai nuo lygiagrečiųjų, yra pristatyta lygiagrečiųjų kompiuterių Flyno klasifikacija, lygiagrečiųjų kompiuterių atminties klasifikavimas, taip pat pristatyti pasaulyje plačiausiai naudojamų lygiagrečiųjų kompiuterių tipai. Kadangi kompiuteriai negali veikti be

programinės įrangos šiame skyriuje yra aprašomi plačiai paplitę lygiagrečiojo programavimo standartai ir bibliotekos.

Be to, dauguma šiuolaikinių lygiagrečiųjų skaičiavimų sistemų yra kuriamos iš asmeninių kompiuterių ir į jų elektros energijos suvartojimo problemą galima žiūrėti kaip į vieno asmeninio kompiuterio elektros energijos suvartojimo problemą. Tokios problemos sprendimo būdus galima taikyti visiems kompiuteriams, iš kurių susideda klasteriai ir gridai. Dėl to, skyriuje dar aprašomi įvairių autorių siūlomi elektros energijos suvartojimo mažinimo būdai.

Skyriuje yra aprašomas ir kombinatorinis optimizavimas, nes jo algoritmai vykdomi tiriant elektros energijos suvartojimą.

2.1. Elektros energijos suvartojimo problema

Pastaruoju metu elektros energijos suvartojimo optimizavimas tampa labai svarbiu uždaviniu. Moksliniuose tyrimuose naudojami skirtingi kompiuteriniai ištekliai: superkompiuteriai, klasteriai ir klasteriniai tinklai bei debesų kompiuterija. Šie resursai vartoja daug elektros energijos, todėl pastaruoju metu labai aktualus uždavinys – sumažinti elektros energijos sąnaudas nemažinant skaičiavimų pajėgumų.

Atsirado ir terminas, charakterizuojantis šią problemą – žalioji kompiuterija (angl. *green computing*). Terminas gana naujas, egzistuoja nuo 2005 metų. Pirmas terminą įvedė San Murugesan, pateikęs tokį apibrėžimą: žalioji kompiuterija yra mokslas bei praktiniai patarimai apie tai, kaip reikia projektuoti, gaminti, naudoti ir utilizuoti kompiuterius, serverius, kad jie veiktų kuo efektyviau, minimaliai įtakojant arba visiškai neįtakojant aplinkos (Murugesan 2008). Dabartiniu metu įvairių autorių straipsniuose galima sutikti ir kitokių šios problemos įvardinimų: draugiškos žaliosios informacinės (Friendly IT), informacinės komunikacines technologijos (ICT) (Chai-

Arayalert ir Nakata 2011; Ruth 2009), aplinką tausojantys skaičiavimai ir IT (Witkowski *ir kt.* 2010) ir kiti.

Green500 (www.green500.org) – sąrašas analogas Top500 (galingiausių pasaulio superkompiuterių sąrašas), tik jame vertinami klasteriai ir superkompiuteriai energijos suvartojimo atžvilgiu. Sąrašas atsirado 2005 metais po dr. Wu-chun Feng pristatymo ir straipsnio konferencijoje "IEEE IPDPS Workshop on High-Performance, Power-Aware Computing" (Feng 2006; Sharma *ir kt.* 2006).

Po vėlesnio dr. Wu-Chun Feng pranešimo, kuris buvo pristatytas konferencijoje "Clusters and Computational Grids for Scientific Computing 2006", mokslininkai pradėjo išsamiau nagrinėti elektros energijos suvartojimo problemą. Ši problema paminėta ir kitų autorių (Scogland *ir kt.* 2012; Murugesan 2008). 2010 metais buvo organizuojama pirmoji tarptautinė žaliosios kompiuterijos konferencija: First International Green Computing Conference (IGCC'10). Žaliosios kompiuterijos aktualumą parodo šios konferencijos tolimesnis egzistavimas. 2012 metais organizuojama jau trečioji konferencija „Third International Green Computing Conference (IGCC'12)" (Morse ir Shirazi 2010). Pastaruoju metu internete atsirado puslapiai, kurie renka informaciją apie visas žaliosios kompiuterijos konferencijas (Berral-García *ir kt.* 2012). Taip 2012 metais buvo suplanuotos 53 konferencijos, susietos su žaliosios kompiuterijos tematika.

Šiuo metu nėra viešai prieinamų elektros energijos suvartojimo optimizavimo sprendimų, skirtų klasterinėms sistemoms. Dažniausiai siūlomi kompleksiniai įrankiai, skirti klasterių priežiūrai ir administravimui, be to, jie yra mokami. Lietuvoje kol kas tam skiriamas per mažas dėmesys, nors klasterių ir superkompiuterių skaičius kiekvienais metais vis didėja.

Kadangi dauguma šiuolaikinių lygiagrečiųjų skaičiavimų klasterių sudaryti iš asmeninių kompiuterių, tai energijos suvartojimo mažinimo problemą galima nagrinėti vieno tokio kompiuterio atžvilgiu. Sprendimus reikia taikyti

globaliai visiems kompiuteriams, iš kurių sudarytas klasteris. Toliau aprašomi būdai, sumažinantys energijos suvartojimą, skirti vienam asmeniniam kompiuteriui.

Žaliosios kompiuterijos atsiradimo motyvacija gana aiški ir jos principų taikymas ekonominiais sumetimais pagirtinas ir gana efektyvus. Patys paprasčiausi būdai mažinti elektros energijos suvartojimą: vaizduoklio naudojimo minimizavimas, kompiuterių užmigdymas arba išjungimas, kai jie nėra naudojami, ir kt. (Przybyla ir Pegah 2007).

Yra keli būdai, kaip sumažinti elektros energijos suvartojimą šiuolaikiniuose kompiuteriuose (Talebi ir Way 2009):

1. Išjungti įrenginius, kai jie nėra naudojami. Įrenginių išjungimas yra vienas veiksmingiausių energijos suvartojimą mažinančių būdų. Tai vienas iš veiksmingiausių būdų minimizuoti energijos suvartojimą įmonėse ir ne tik. Bet tai ne visada padeda, nes daugelis įrenginių ir toliau vartoja elektrą, bet jau kur kas mažesniais kiekiais – „phantom load“ (Nordman *ir kt.* 1998; Talebi ir Way 2009).
2. Kompiuterių elektros energijos suvartojimą mažinančių įrankių naudojimas. Dauguma šiuolaikinių kompiuterių ir operacinių sistemų turi integruotas priemones, skirtas mažinti energijos suvartojimą. Problema tame, kad tos priemonės išlieka nepanaudotos vartotojų dėl įvairių priežasčių. Be to, tokių priemonių administravimas reikalauja papildomų žinių, nes, blogai atlikus nustatymus, kompiuteris po užmigimo gali ir neatsibusti (Nordman *ir kt.* 1998; Talebi ir Way 2009; Talebi 2008). Tokių priemonių yra daug:
    a. Vaizduoklio apsauga. Vienas iš paprasčiausių ir visiems vartotojams prieinamų elektros energijos suvartojimą mažinančių būdų.

b. Vaizduoklio užmigdymas. Kaip ir vaizduoklio apsauga, labai efektyvi ir paprastam vartotojui prieinama priemonė. Kuo anksčiau vaizduoklis pereis į miego režimą (persijungs į laukimo režimą), tuo greičiau sumažės elektros energijos suvartojimas.

c. Kietojo disko miego režimas. Kompiuteris gali sumažinti kietojo disko energijos suvartojimą išjungdamas jį, kai kietas diskas yra nenaudojamas.

d. Kompiuterio laukimo režimas. Kai kompiuteris nenaudojamas ilgą laiką, operacinėse sistemose numatytas kompiuterio pervedimas į laukimo režimą. T. y., visi kompiuterio komponentai išjungiami, išskyrus pagrindinę plokštę. Pagrindinė plokštė minimaliai vartoja elektros energiją tik operatyviosios atminties palaikymui – duomenims saugoti. Tokiu būdu pavyksta žymiai sumažinti elektros energijos suvartojimą. Pavyzdžiui, jei kompiuterio darbo metu elektros galia yra apie 100 W, tai po perėjimo į laukimo režimą ji bus apie 10 W.

e. Kompiuterio išjungimas su operatyviosios atminties išsaugojimu kietajame diske. Toks energijos taupymo režimas panašus į kompiuterio laukimo režimą tik su vienu dideliu skirtumu – kompiuteris pilnai išjungiamas. Prieš išsijungimą, kompiuterio operatyviosios atminties būsena įrašomą į kietąjį diską. Dėl šios priežasties išjungimas trunka ilgiau negu be įrašymo. Privalumas – lyginant su kompiuterio laukimo režimu energijos suvartojimas sumažėja apie 2 kartus. Įjungimo metu kompiuterio operatyviosios atminties būsena yra talpinama į operatyviąją atmintį ir vartotojas gali sėkmingai dirbti ten, kur baigė darbą.

3. „Phantom load“ panaikinimas. „Phantom load“ – terminas, apibrėžiantis kompiuterinės technikos elektros energijos suvartojimą,

kai įrenginys pilnai išjungtas. Daugelis įrenginių tiesiog neturi fizinio elektros energijos išjungimo mygtuko, tik elektros laidą, kurį galima ištraukti iš rozetės. Kitų įrenginių atvejais tai būna dėl įvairių techninių priežasčių. Dalį tokių priežasčių galima panaikinti. Pavyzdžiui – vaizduoklio laukimo režimas. Tam, kad energija nebūtų vartojama, vaizduoklį galima visiškai išjungti mygtuko pagalba. Be to, „phantom load“ energijos suvartojimą apsprendžia ir pagrindinės plokštės parametrų nustatymai. Vienas iš tokių parametrų yra „WAKE ON LINE“, jį išjungus galima sumažinti energijos suvartojimą (Talebi ir Way 2009).

4. Kompiuterio atnaujinimas naudojant mažai elektros energijos resursus eikvojančius komponentus. Darant kompiuterio komponenčių atnaujinimą reikia atkreipti dėmesį į tokias naujas komponentes, kurios efektyviai naudoja resursus. Pavyzdžiui, TFT LCD vaizduoklius reikia keisti į LED LCD vaizduoklius. Tokiu būdu galima sumažinti elektros energijos vartojimą 2 ar 3 kartus (HP S2231a 21.5″ TFT LCD maksimali galia – 50 W, HP 2211x 21.5″ LED LCD maksimali galia – 29 W) (Simpson 2003; HP company 2010). Kitas pavyzdys – naudoti naujos kartos daugiabranduolinius procesorius vietoje senų, tokiu būdu galima sutaupyti apie 38 % procesoriaus ir iki 50 % bendro kompiuterio elektros energijos suvartojimo (Tseng ir Figueira 2010).

5. Kompiuterio atnaujinimas pratęsiant jo gyvavimo trukmę. Prie tokių naujinimų galima priskirti operatyviosios atminties praplėtimą. Nauji atminties moduliai sunaudoja tiek pat elektros energijos, bet kompiuterio sparta pakyla. Be to, praplečiant atmintį, nebūtina keisti kitų komponenčių (Horvath ir Masanet 2006; Simpson 2003).

6. Perkant naujus įrenginius reikia pirkti efektyvius ir mažai elektros energijos vartojančius. Perkant naujus įrenginius reikia atkreipti dėmesį į jų energijos vartojimą, ar jie atitinka standartus, tokius, kaip Energy Star, 80 PLUS (Bronze, Silver, Gold, Platinum, Titanium) (Nordman *ir kt.* 1998). Be to, nauji įrenginiai turi būti pagaminti iš ekologiškai švarių medžiagų (Grossman 2006; Talebi ir Way 2009).
7. Reikia ne tik pirkti efektyviai veikiančius įrenginius, bet ir vystyti technologijas, susietas su jais (Talebi ir Way 2009).

Visi išvardinti patarimai gerai atspindi žaliosios kompiuterijos sąvoką. Superkompiuterio atveju juos taikyti reikia ne vieno kompiuterio atžvilgiu, o visiems kompiuteriniams resursams, iš kurių sudarytas klasteris arba superkompiuteris. O tą padaryti nėra taip paprasta. Plačiau lygiagrečiųjų sistemų trūkumai žaliosios kompiuterijos atžvilgiu aptariami 2.4 skyriuje.

2.2. Kompiuteriniai skaičiavimai ir jų rūšys

Poskyryje yra aprašomi šiuolaikinių kompiuterių procesoriaus architektūra, nuoseklieji ir lygiagretieji skaičiavimai.

2.2.1. Kompiuterių architektūra

Kalbant apie kompiuterius reikia žinoti iš ko susideda šiuolaikiniai procesoriai.

Procesorius – loginis įtaisas, apdorojantis duomenų srautą. Procesoriaus sąvoka yra bendrinė, reiškianti gana abstrakčią informacinių sistemų rūšį, atliekančią manipuliacijas su duomenimis, tačiau dažniau naudojama kalbant apie skaičiavimams skirtus procesorius, realizuotus, kaip aparatiniai įrenginiai (dažniausiai – mikroprocesoriai). Šiuolaikinis procesorius dažniausiai būna daugiabranduolinis – vienoje silicio pagrindo plokštelėje integruoti 2 arba daugiau mikroprocesorių atliekančių tas pačias arba skirtingas funkcijas

(centrinio procesoriaus branduolių, vaizdo informacijos apdorojimo ir pateikimo bei kitas).

Procesas – programos vykdomų veiksmų seka. Procesams skirstomi kompiuterio ištekliai. Procesai gali vienas su kitu keistis duomenimis, vienas procesas gali inicijuoti kitą (Dagienė *ir kt.* 2008).

Disertacijoje procesu laikomas programos vykdymas viename iš procesoriaus branduolių.

2.2.2. Nuoseklieji skaičiavimai

Nuoseklieji skaičiavimai – toks skaičiavimo būdas, kai kompiuteriniai skaičiavimai vykdomi viename kompiuteryje (viename skaičiavimo įrenginyje CPU – angl. *central processing unit*), visa užduotis yra padalinama į žingsnius (instrukcijas), kurie atliekami vienas po kito nuosekliai (2.1 pav.). Toks skaičiavimo būdas vyravo kol nebuvo sukurti superkompiuteriai, kompiuterių klasteriai ir daugiabranduoliniai procesoriai.

2.1 pav. *Nuosekliųjų skaičiavimų schema*

2.2.3. Lygiagretieji skaičiavimai

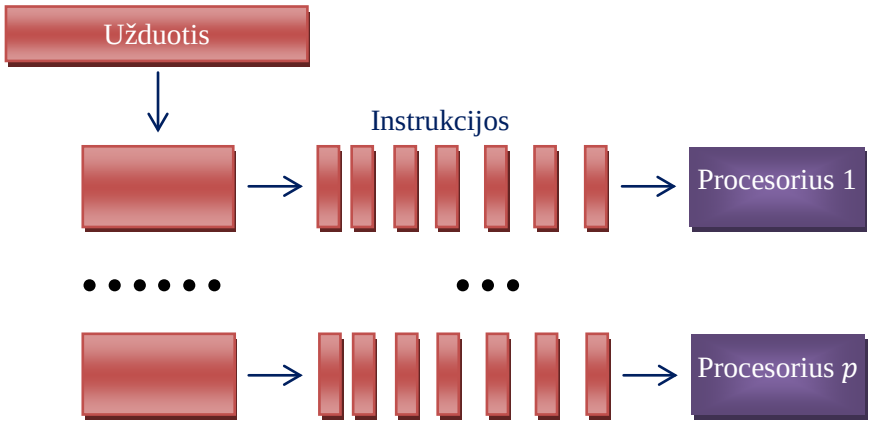
Lygiagretieji skaičiavimai – tai kompiuterinių skaičiavimų organizavimo būdas, kai programos kuriamos kaip vienu metu (lygiagrečiai) dirbančių ir sąveikaujančių procesų visuma. Terminas apima ir lygiagretumą programinėse realizacijose, ir efektyviai veikiančios techninės dalies kūrimą. Lygiagrečiųjų skaičiavimų teorija sudaro algoritmų teorijos taikomąją dalį.

Aprašant procesą galima apibrėžti tokius žingsnius (2.2 pav.):

- užduotis yra padalijama į p dalių (p dažnai atitinka procesorių arba

branduolių skaičių);

- kiekvienas iš p procesorių gauna savo užduoties dalį;
- užduoties dalis, atitekusi procesoriui, yra sudalijama į porcijas (instrukcijas), kurias ir vykdo procesorius.



2.2 pav. *Lygiagrečiųjų skaičiavimų schema*

Egzistuoja keli lygiagretinimo būdai. Vienas iš jų – lygiagretusis duomenų apdorojimas, kitas – procesų lygiagretumas, trečias – mišrus. Šie būdai skiriasi programine ir technine skaičiavimų realizacija (Čiegis 2005).

Lygiagretusis duomenų apdorojimas turi dvi atmainas: lygiagretųjį ir konvejerinį variantą.

- Konvejerinis apdorojimas. Konvejerinio apdorojimo būdu visos bendros operacijos yra išskaidomos į kelis vienodos trukmės etapus. Kiekvienas etapas perduoda savo rezultatą kitam etapui ir tuo pat laiku priima naujus duomenis. Jeigu konvejerinis įrenginys turi l etapų, o kiekvienas etapas atliekamas per vieną laiko vienetą, tai n nepriklausomų operacijų apdorojimas tokiu įrenginiu sudarys $(l + n - 1)$ laiko vienetų. Lyginant su apdorojimu, nekonvejerizuojant $(l \times n)$ apdorojimo laikas sumažėja $(l + n - 1 \leq l \times n)$.
- Lygiagretusis apdorojimas. Jeigu įrenginys, konkrečiu atveju – procesorius, atlieka vieną operaciją per laiko vienetą, tai 1 000

operacijų jis atliks per 1 000 laiko vienetų. Padarius prielaidą, kad turime penkis tokius pačius nepriklausomus įrenginius, galinčius vienu metu ir nepriklausomai vienas nuo kito veikti, tai 1 000 operacijų penkių įrenginių sistema gali atlikti jau ne per 1 000, o per 200 laiko vienetų. Analogiškai p įrenginių sistema tą patį darbą idealiu atveju atliks per $1\,000/p$ laiko vienetų.

2.3. Lygiagrečiųjų kompiuterių architektūra

Poskyryje pristatomas Flyno lygiagrečiųjų kompiuterių ir lygiagrečiųjų kompiuterių atminties klasifikavimas, kad būtų lengviau suvokiama kuo skiriasi lygiagretieji kompiuteriai nuo nuosekliųjų ir kitų.

2.3.1. Flyno taksonomija

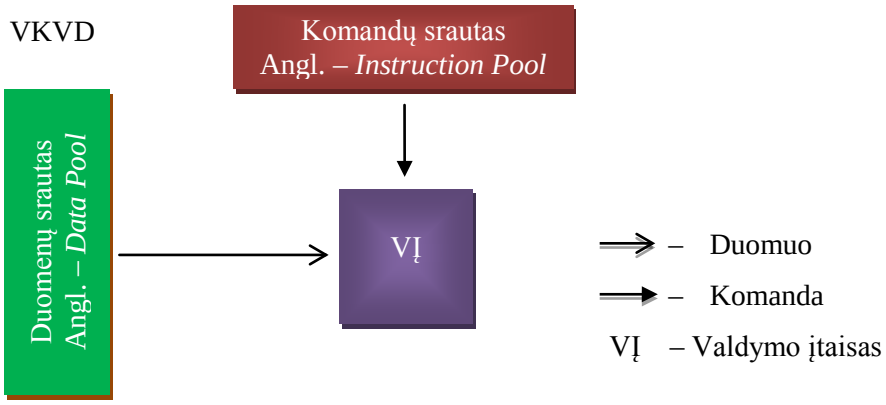
M. Flynas 1966 m. (Flynn 1972) sukūrė vieną iš pirmųjų klasifikavimo sistemų, skirtų lygiagrečiųjų (ir nuosekliųjų) kompiuterių bei programų klasifikavimui. Dabar ji žinoma kaip Flyno taksonomija (angl. *Flynn's taxonomy*). Flynas klasifikavo programas ir kompiuterius pagal tai, ar jie naudojo vieną ar kelis komandų srautus ir pagal tai, ar tos komandos naudojo vieną ar kelis duomenų srautus (Ralph 1990). Flyno taksonomija pateikta 2.1 lentelėje.

2.1 lentelė. *Flyno taksonomija*

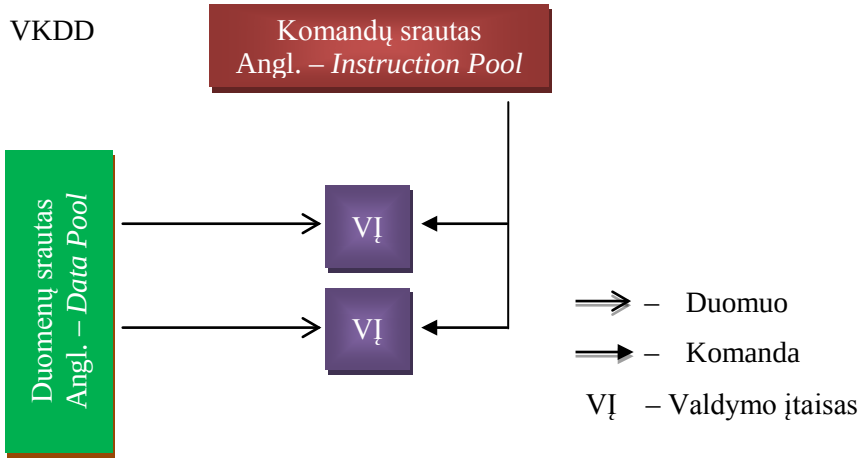
	Vienas komandų srautas (angl. *Single Instruction*)	Daug komandų srautų (angl. *Multiple Instruction*)
Vienas duomenų srautas (angl. *Single Data*)	VKVD (angl. *SISD*)	DKVD (angl. *MISD*)
Daug duomenų srautų (angl. *Multiple Data*)	VKDD (angl. *SIMD*)	DKDD (angl. *MIMD*)

VKVD – tai vienas komandų srautas ir vienas duomenų srautas (2.3 pav.), tradiciniai vieno branduolio personaliniai kompiuteriai yra tokios struktūros.

VKDD – vienas komandų srautas ir daug duomenų srautų, tokia yra vektorinių procesorių (angl. *Vector Processor*) struktūra (2.4 pav.). Tokios sistemos buvo populiarios 1980-aisiais.



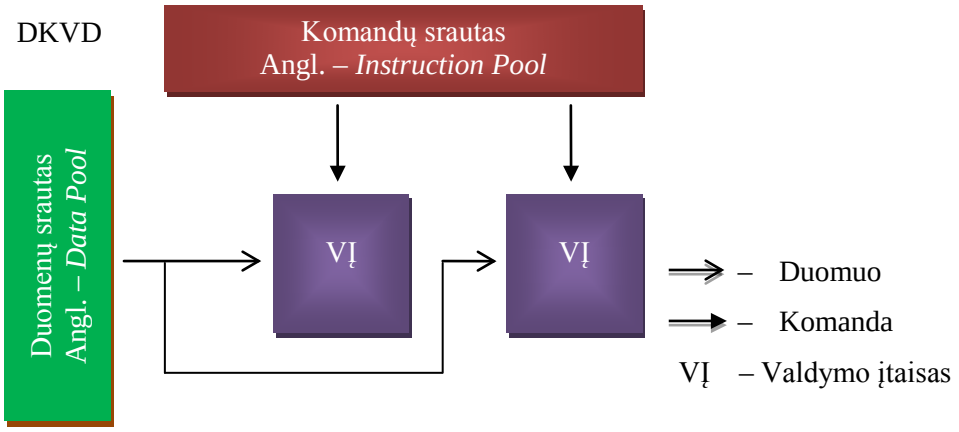
2.3 pav. *Vienos Komandos Vieno Duomenų srauto kompiuterio schema pagal Flyno taksonomiją*



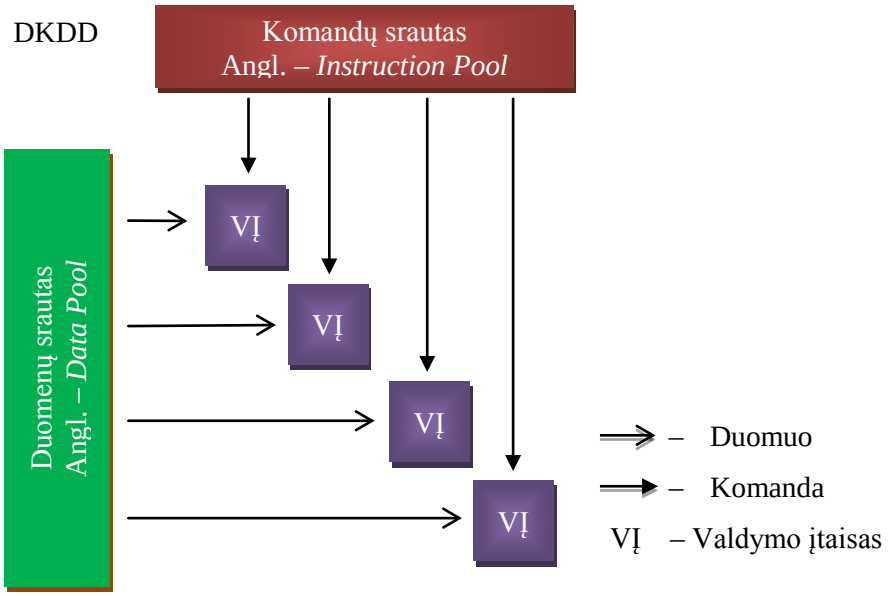
2.4 pav. *Vienos Komandos Daug Duomenų srautų kompiuterio schema pagal Flyno taksonomiją*

DKVD (MISD) – daug komandų srautų ir vienas duomenų srautas (2.5 pav.).

DKDD – daug komandų srautų ir daug duomenų srautų. Dauguma šiuolaikinių daugiaprocesorinių kompiuterių turi tokią struktūrą (2.6 pav.).



2.5 pav. *Daug Komandų Vieno Duomenų srauto kompiuterio schema pagal Flyno taksonomiją*



2.6 pav. *Daug Komandų Daug Duomenų srautų kompiuterio schema pagal Flyno taksonomiją*

Egzistuoja ne tik Flyno lygiagrečiųjų kompiuterių klasifikacija. Savo klasifikaciją siūlo ir kiti mokslininkai: R. Hockney (Flyno DKDD taksonomijos praplėtimas), T. Feng (skirsto pagal bitų skaičių mašininiame žodyje ir pagal vienu metu apdorojamų žodžių skaičių), W. Handler (pagal lygiagretųjį ir konvejerinį duomenų apdorojimą), L. Snyder (pagal duomenų, komandų srautuose išrinkimą ir apdorojimą), D. Skillicorn (praplėtė Flyno

taksonomiją bei pabandė suklasifikuoti sunkiai klasifikuojamas sistemas) (Воеводин ir Воеводин 2002).

2.3.2. Lygiagrečiųjų kompiuterių atmintis

DKDD tipo kompiuteriai dažnai klasifikuojami pagal naudojamos atminties tipą – bendrosios ir paskirstytosios atminties lygiagretieji kompiuteriai (Воеводин ir Воеводин 2002).

Bendrosios atminties lygiagretieji kompiuteriai turi vieną bendrąją atminties bloką ir visi procesoriai gali tiesiogiai pasiekti visas atminties vietas (Žilinskas J. 2002). Procesoriai bendrauja tarpusavyje per bendrąsias duomenų struktūras, esančias bendrojoje atmintyje.

Paskirstytosios atminties lygiagrečiajame kompiuteryje kiekvienas procesorius gali tiesiogiai perskaityti ir įrašyti tik jo lokaliojoje atmintyje esančius duomenis (Paulavičius 2010). Procesoriai bendrauja tarpusavyje siųsdami vienas kitam pranešimus.

2.4. Lygiagretieji kompiuteriai

Egzistuoja daugelis lygiagrečiųjų kompiuterinių sistemų tipų. Galima išvardinti kelias iš jų:

- superkompiuteriai;
- klasteriai;
- vaizdo plokštės, skirtos skaičiavimams (GPU);
- gridai;
- debesų kompiuteriai;
- kiti.

Šie kompiuteriai skiriasi tarpusavyje. Skiriasi jų techninė realizacija ir programinė įranga. Toliau trumpai pristatomi visi paminėti lygiagretieji kompiuteriai.

2.4.1. Superkompiuteriai

Superkompiuterių eros pradžia galima laikyti 1976 metus. Tais metais buvo sukurta pirma vektorinė sistema Cray 1 (vektorinė sistema savo pavadinimą gavo už duomenų struktūros saugojimą sistemos atmintyje, vektoriaus pavidalu). Rezultatai, gauti su programų rinkiniu sukurtu tai sistemai, buvo žymiai geresni už bet kokio kito tuo metu egzistuojančio kompiuterio rezultatus, ir sistema gavo superkompiuterio vardą.

Kompiuterio architektūros ir programinės įrangos vystymas įtakojo naujų sistemų, kurios kardinaliai skiriasi viena nuo kitos atsiradimą. Dėl to „superkompiuterio“ sąvoka tapo daugiaprasmė ir verta diskusijų.

Top500 (www.top500.org) – oficialus pačių greičiausių superkompiuterių pasaulyje sąrašas (Meuer ir Dongarra 2011). Sąrašas skelbiamas nuo 1993 metų ir atnaujinamas du kartus per metus – birželį ir lapkritį. Kompiuterių realus našumas įvertinamas naudojant Linpack testą (Linpack 2012). Teorinį viršutinį kompiuterio našumo rėžį (Rpeak) galima įvertinti sudauginus procesorių ar branduolių skaičių, taktinį dažnį ir atliekamų slankaus kablelio operacijų skaičių per vieną procesoriaus taktą (Žilinskas J. 2011). Paprastai rezultatai viešinami lentelėje kurioje yra nurodoma ne tik Top500 superkompiuterio pozicija sąraše, bet ir gamintojas, kompiuterių tipai, įstaiga savininkė, šalis, įdiegimo metai, taikymo sritis, procesorių (branduolių) skaičius, pasiektas maksimalus našumas sprendžiant „Linpack“ testą (Rmax), teorinis viršutinis kompiuterio našumo rėžis (Rpeak), uždavinio dydžiai maksimaliam ir pusei maksimalaus našumui pasiekti, elektrinė galia.

Lietuvoje taip pat yra keli superkompiuteriai. Vienas iš jų ne taip seniai pradėjo savo veiklą – Vilniaus universiteto superkompiuteris, veikiantis nuo

2011 birželio mėnesio, o pristatytas plačiai visuomenei 2012 m. kovo 13 dieną. Kai kurie šiuo metu patys greičiausi pasaulyje superkompiuteriai iš 2011 metų lapkričio sudaryto Top500 sąrašo pateikti priede.

Bet superkompiuteriai turi vieną didžiulį trūkumą – didelis elektros energijos suvartojimas. Greičiausias pasaulyje superkompiuteris pagal Top500 sąrašą (pagal 2011 lapkričio mėnesio sąrašą) – „K computer“, sąraše Green500, užima tik 32 vietą pagal santykį MFlop/s per W, o bendras jo vartojimas, kai jis veikia pilnu pajėgumu, yra 12 659,89 kW. Be to, šiam kompiuteriui aušinti reikalingos labai galingos aušinimo sistemos. Daugelis mokslininkų ir kompiuterinės įrangos kūrėjų bando išspręsti šią problemą. Taip mokslininkai ieško būdų paleidinėti darbus taip, kad mažiau veiktų kondicionavimo sistemos (Banerjee *ir kt.* 2010). Taip pat siūlo atsisakyti brangių superkompiuterių ir naudoti skaičiavimams mažus, bet galingus, klasterius (Feng *ir kt.* 2007). Be to kai kurie mokslininkai formuluoja naujas rekomendacijas kaip kurti naujus žaliuosius superkompiuterius ir išnaudoti virtualizacijos technologijas juose (naudoti greitus ir sparčius SSD diskus, jei yra galimybė visai atsisakyti kietųjų diskų, nenaudoti superkompiuteriuose GPU technologijų, daugiau išnaudoti skaičiavimuose virtualizacijos technologijas ir kitos rekomendacijos) (Oliker 2009; Feng *ir kt.* 2008; Williams ir Curtis 2008).

2.4.2. Klasteriai

Klasteris – tai įprastų asmeninių kompiuterių, sujungtų į lokalų kompiuterių tinklą grupė, kurios visi kompiuteriai gali dirbti kaip vienas skaičiavimo resursas. Manoma, kad klasteriai yra efektyvesni negu asmeniniai kompiuteriai ir daug pigesni už kitas lygiagrečiąsias skaičiavimo sistemas (sistemas, kuriose naudojamos netipinės kompiuterinės technikos komponentės ir specializuota programinė įranga).

Per šiuos metus įvyko kokybinis skaičiavimo technikos pasikeitimas. Buvo sukurti ir paplito pasaulyje asmeniniai kompiuteriai, susiformavo daugybė lygiagrečiojo programavimo teorijų ir metodų. Tai paspartino lygiagrečiųjų skaičiavimų teorijas realizuoti praktiškai. Tais laikais realizacija buvo brangi, nes reikalavo greitų specifinių skaičiavimo komponenčių. Be to, visą laiką gerėjo rodiklis „kaina / našumas“. Dėl tokių rodiklių klasterių atsiradimas tapo neišvengiamas, o tokių sistemų privalumai buvo neabejotini. 2011 metų lapkričio mėnesio Top500 pačių greičiausių skaičiavimo sistemų sąraše klasterinės sistemos sudaro didžiąją dalį – 410 klasterių iš 500 kompiuterių.

Klasteris Beowulf. Teigiama, kad pirmasis klasteris buvo sukurtas mokslo centre NASA – Goddard Space Flight Center 1994 m. vasarą, vadovaujant Tomui Sterlingui ir Donui Bakerui. Šis klasteris buvo sudarytas iš 16 kompiuterių su tokia specifikacija: procesorius – 486DX4 100 MHz, operatyvioji atmintis – 16 MB, tinklo plokštė – 10 Mbit/s, operacinė sistema – Linux su kompiliatorium GNU, jame veikė programos su MPI sąsaja (MPI standartas, kaip lygiagrečiųjų skaičiavimų vykdymo įrankis buvo pristatytas plačiai visuomenei 1993 m.).

Dažniausiai mokslinių skaičiavimų klasteriams naudojama OS – Rocks Cluster Linux‘o versija (naujausia versija – 6.0) (University of California 2011).

Lietuvoje taip pat yra nemažai klasterių, juos turi dauguma didžiųjų universitetų: Vilniaus universiteto klasteriai – CLUSTER.MII.VU.LT ir HPC.MII.VU.LT, Vilniaus Gedimino technikos universiteto klasteris „Vilkas“, Kauno Technologijų universiteto klasteriai ir kiti. Greičiausių pasaulyje klasterių sąrašas iš Top500 sąrašo pateiktas prieduose, 2 lentelė.

Kaip ir superkompiuteriai, klasteriai turi trūkumą. Elektros energijos suvartojimas auga kartu su klasterio dydžiu. Dėl to dauguma tokių klasterių administratorių taip pat susiduria su šia problema. Dažnai galima taikyti metodus, skirtus superkompiuteriams, bet yra ir skirtumų. Taip vieni

mokslininkai siūlo naudoti klasterius, kuriuose kompiuteriuose-darbininkuose nėra kietųjų diskų (Salah *ir kt.* 2011), kiti siūlo išnaudoti virtualizacijos technologijas (Hu *ir kt.* 2008), be to formuluojami ir bendri patarimai, kaip ir iš kokių komponenčių turi būti surenkami nauji arba tobulinami seni klasteriai (Li *ir kt.* 2010; Schott ir Emmen 2010; Baliga *ir kt.* 2011).

2.4.3. Gridai

Paskirstytasis skaičiavimas – tai kompiuterinio apdorojimo metodas, skirtas didelio skaičiaus užduočių sprendimui. Tam tikslui naudojamas „virtualus superkompiuteris“, sujungtas į vieną skaičiavimų resursą kompiuterinių tinklų pagalba iš vienu metu dirbančių kompiuterių. Tokios technologijos naudojamos spręsti mokslinius, matematiškai sudėtingus skaičiavimus, reikalaujančius didelių skaičiavimo resursų. Be to, griduose paskirstytieji skaičiavimai naudojami ir komercinėse organizacijose, tokių sudėtingų uždavinių sprendimui kaip: ekonominis prognozavimas, seismoanalizė, naujų vaistų kūrimas ir testavimas ir t. t.

Gridas kaip tinklo organizacija – tai laisvai prieinama suderinta ir standartizuota aplinka, kuri užtikrina lankstų, saugų, koordinuotą skaičiavimų ir informacijos saugojimo resursų paskirstymą. Tokie resursai priklauso tai pačiai aplinkai vienos virtualiosios organizacijos ribose. Be to, gridai – tai geografiškai paskirstyta infrastruktūra, jungianti daug skirtingo tipo resursų (procesoriai, ilgalaikė ir trumpalaikė atmintys, saugyklos ir duomenų bazės, kt.), prieigą prie kurių vartotojas gali gauti iš bet kurios vietos, nepriklausomai nuo tų resursų išdėstymo.

Gridą sudaro daug įvairių kompiuterių, naudojančių skirtingos architektūros procesorius, turinčių įvairią programinę įrangą ir sujungtų kompiuteriniais tinklais. Gridai gali jungti ir labai nutolusius kompiuterius (Čiegis 2005).

Dabartiniu metu egzistuoja keletas grido sistemų tipų (Yang ir Li 2005):

- Savanoriški gridai – gridai, kurių naudojimo pagrindą sudaro savanoriškai teikiami kompiuteriniai resursai, pavyzdžiui, BOINC (Anderson 2004).
- Moksliniai gridai – gridai, kuriuose gerai lygiagretinamos programos specialiu būdu (naudojant gLite, NorduGrid, arba Globus Toolkit programinę aplinką) (Muntean ir Joldos 2011).
- Gridai, kurių veikimą sudaro resursų ir telkinių išskyrimas pagal reikalavimą (dažniausiai tai komerciniai gridai – angl. *Enterprise grid*). Komercinės programos leidžiamos viename virtualiame kompiuteryje, kuris būna sudarytas iš kelių fizinių kompiuterių, sujungtų į vieną, grido technologijų pagalba (Amazon, Google, Yahoo).

2005 metais Lietuvoje, Latvijoje, Estijoje, Lenkijoje, Švedijoje, Šveicarijoje ir Baltarusijoje buvo sukurtas klasterių junginys (Baltic Grid), skirtas bendram paskirstytųjų skaičiavimo resursų panaudojimui. Kiekviena iš šių šalių turi savo gridus: Eesti Grid, Latvy Grid, LitGrid. Kiekvienos šalies gridas sudarytas iš klasterių, kurie sujungiami į vieną didelį skaičiavimo resursą specialios programinės ir techninės įrangos pagalba. Lietuvoje LitGrid tinklui priklauso tokios organizacijos: Vilniaus universitetas, Vilniaus universiteto Matematikos ir informatikos institutas, Vilniaus universiteto Teorines fizikos ir astronomijos institutas, Kauno technologijos universitetas, Vilniaus Gedimino technikos universitetas, Klaipėdos universitetas, Vytauto Didžiojo universitetas, Šiaulių universitetas, LSMU Psichofiziologijos ir reabilitacijos institutas, FTMC Fizikos institutas, Lietuvos energetikos institutas, Alytaus kolegija, Marijampolės kolegija, Panevėžio kolegija [B 1] (Bagdonavičius ir Lapienis 2005). Visos organizacijos turi savo skaičiavimų klasterius ir dedikuoja jų dalį darbui LitGrid ir Baltic Grid tinkle. VU MII klasterio jungiamoji schema pateikiama 4.2 skyriuje.

Taip pat Baltic Grid turi savo tinklo resursų ir naudotojų identifikavimo sertifikatų pasirašymo servisą (angl. *Certificate Authority* – sutrumpintai CA). Daugiau informacijos apie reikalavimus sertifikatui yra pateikta Baltic Grid CA puslapyje (BalticGrid II 2012). Sertifikato turėjimas tik identifikuoja asmenį. Dėl prieigos prie skaičiavimo resursų reikia kreiptis į juos valdančias organizacijas.

Visi resursai Baltic Grid tinkle yra padalijami pagal interesų grupes (Baltic Grid virtualiosios organizacijos). Resursų valdytojai paprastai suteikia prieigos teises virtualiosioms organizacijoms (VO). Virtualioji organizacija – viename projekte dirbančių asmenų grupė. Baltic Grid yra šios virtualiosios organizacijos: Baltic Grid VO, Gamess VO, LitGrid VO. Į Baltic Grid VO įtraukiami visi, gavę Baltic Grid CA pasirašytą sertifikatą. Resursų turėtojai neprivalo leisti jungtis jos nariams.

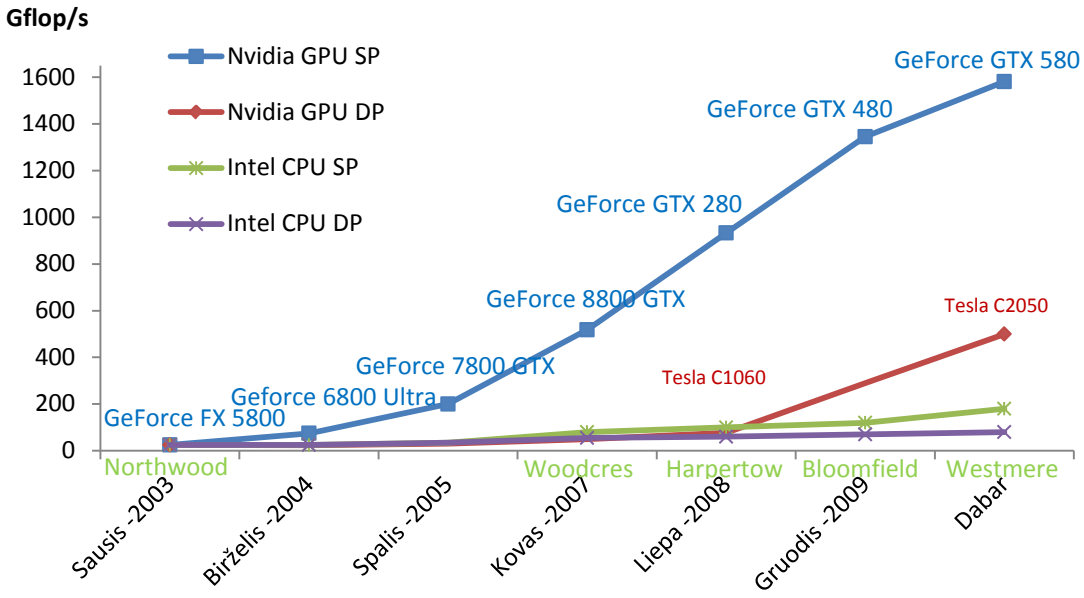
Kadangi dauguma iš gridų sudaryti iš paprastų klasterių, tai visi jie turi tas pačias elektros energijos suvartojimo problemas kaip ir klasteriai, bet yra ir skirtumų. Pavyzdžiui, patariama naudoti daugiabranduolinius procesorius, vengti kondicionierių naudojimo, naudoti savaiminį serverinių kondicionavimą, efektyviau realizuoti programas, tam, kad jie mažiau kaitintų procesorius (Schott ir Emmen 2010; Zhang *ir kt.* 2010 b).

2.4.4. Grafikos apdorojimo įrenginiai

Per paskutinius kelerius metus grafikos apdorojimo įrenginiai (angl. *GPU – graphic processing unit*) smarkiai patobulėjo ir tapo galingais skaičiavimo įrenginiais, ką galima matyti 2.7 pav. Paveiksle labai gerai matosi, kad dabartiniu metu GPU įrenginiai, pagal skaičiavimų spartą net kelis kartus lenkia klasikinius procesorius.

Šiandien egzistuojantys grafikos procesoriai su keliais branduoliais ir labai dideliu atminties pralaidumu turi neįtikėtiną galią grafiniuose ir negrafiniuose skaičiavimuose. Dabartiniu metu juos plačiai naudoja mokslininkai savo

eksperimentiniuose tyrimuose bei tyrimų metu gautiems duomenims apdoroti. TOP 500 superkompiuterių sąraše klasterių su grafikos apdorojimo įrenginiais skaičius didėja. Kelios institucijos, kurių klasteriuose yra nVidia Tesla GPU su CUDA (plačiau apie CUDA parašyta poskyriuje 2.5.4): Ames Lab Iowa State, Argonne National Labs, BNP-Paribas, British Aerospace, Cambridge, Chinese Academy of Sciences, CSIRO Australian National Supercomputing Center, Daresbury Labs, Fermi Research Labs, Harvard, Hess, HLRS, Max Planck Institute, National Taiwan University, NCSA (National Center for Supercomputing Applications), Oak Ridge National Laboratory, Pacific Northwest National Laboratory, Petrobras, Riken, TACC, Tokyo Institute of Technology, TOTAL, University of Michigan (NVIDIA corporation 2011).



2.7 pav. *Procesorių ir grafikos skaičiavimų įrenginių sparta, flop/s*

Tačiau klasteriai ar superkompiuteriai, sudaryti iš komponenčių su GPU, turi didelį trūkumą lyginant su klasikinėmis sistemomis. Jų elektros energijos suvartojimas yra ypač didelis. Nes paprastai GPU darbo metu labai kaista, o tai reiškia, kad jie turi būti aušinami. Kai veikia vienas kompiuteris su GPU tai silpnai pastebima, bet, kai jų daug, reikia naudoti galingas kondicionavimo

sistemas. Tokių kondicionavimo sistemų elektros energijos vartojimas yra irgi nemažas. Kol kas nėra lengvų šios problemos sprendimo būdų, bet dauguma autorių siūlo būdus, siejamus su efektyviu uždavinių paskirstymu tarp kompiuterių su GPU (Chen *ir kt.* 2010; Wang *ir kt.* 2010; Wakatani 2012).

2.5. Lygiagrečiosios programinės technologijos

Lygiagrečiųjų skaičiavimų realizavimas priklauso nuo techninės įrangos išvystymo bei programinės įrangos evoliucionavimo. Egzistuoja įvairios lygiagrečiųjų skaičiavimų realizavimo priemonės. Vieni įrankiai skirti realizuoti skaičiavimus bendrosios atminties kompiuteriuose, kiti – paskirstytosios atminties kompiuteriuose. Toliau yra aprašomi tokie lygiagrečiųjų skaičiavimų standartai ir realizacijos kaip OpenMP, MPI, JPPF, Cuda, OpenCL, XMPP-MPI.

2.5.1. OpenMP – programavimo standartas

OpenMP (angl. *Open MultiProcessing* – atviras daugiaprocesiškumas) realizuoja lygiagrečiuosius skaičiavimus naudojant gijų (programos dalys vykdomos lygiagrečiai) ir bendrosios atminties technologijas (Chandra *ir kt.* 2000). Pagrindinė gija skaičiavimų metu sukuria kelias papildomas gijas, tarp kurių paskirstomos užduoties dalys. Gijos yra vykdomos kompiuteryje su bendrąja atmintimi.

Lygiagrečiai vykdomos užduotys ir jų vykdymui reikalingi duomenys yra aprašomi specialiomis direktyvomis. Gijų skaičių galima keisti programos vykdymo metu.

Esminiai OpenMP elementai:

- konstrukcija, skirta gijų kūrimui (direktyva „parallel“);
- konstrukcija, skirta paskirstyti darbus tarp gijų (direktyvos „do“, „for“ ir „section“);

- konstrukcija, skirta darbui su duomenimis (direktyvos „shared“ ir „private“);
- konstrukcija, skirta gijų sinchronizavimui arba atminties blokavimui (direktyvos „critical“, „atomic“ ir „barrier“);
- bibliotekų procedūros, skirtos apskaičiuoti vykdymo laiką („omp_get_wtime“, „omp_get_wtick“);
- aplinkos kintamieji („omp_num_threads“, „omp_get_thred_num“).

Dabartiniu metu naujausia versija yra OpenMP 3.1 (OpenMP 2011). OpenMP yra realizuota įvairiuose kompiliatoriuose, pavyzdžiui: GCC, Sun Studio compilers, Visual C++ (nuo 2005 versijos), Intel C++ Compiler, IBM XL compiler, PGI ir kiti (Chandra *ir kt.* 2000).

2.5.2. MPI – pranešimo perdavimo sąsaja

Kuriant MPI (angl. Message Passing Interface – pranešimo perdavimo sąsaja) standartą dalyvavo ne tik mokslininkai, bet ir superkompiuterių gamintojai, kurių pagrindinis tikslas buvo efektyviai panaudoti esamus ir būsimus kompiuterių resursus (Gropp *ir kt.* 1995; Čiegis 2005; Jakušev 2007). Kuriant MPI standartą, buvo atsižvelgta į daugelio kitų tuo metu egzistuojančių standartų panaudojimo patirtį. MPI naudojanti programa yra vykdoma naudojant specialų servisą – paleidėją. Paleidėjas paleidžia programą (vykdomąjį failą) kelis kartus (priklauso nuo tuo, kiek branduolių yra procesoriuje ir kiek procesų skirta vienam kompiuteriui) skirtinguose kompiuteriuose. Taikant specialias MPI funkcijas, kiekviena programa gali sužinoti, kiek procesų yra paleista, koks yra proceso numeris, taip pat ir keistis duomenimis su kitais procesais. MPI yra standartas, kuriame realizuotas išreikštinis duomenų siuntimo modelis (duomenys perkeliami iš vieno proceso adresų srities į kitų procesorių adresų sritis tinklo bei bendradarbiavimo operacijų pagalbą) (Gropp *ir kt.* 1995; Čiegis 2005).

MPI turi tokias savybes:

- Fiksuotas procesų skaičius – procesų skaičius apibrėžiamas paleidimo metu.
- Komunikatoriai ir topologijos – nurodant komunikatorių galima išskirti procesų grupę, kurioje siunčiami pranešimai nebus supainioti su kitais pranešimais, siunčiamais tarp tų pačių procesų. Tai ypač svarbu kuriant naujas bibliotekas MPI pagrindu. Taip pat, kuriant naujas grupes, galima nurodyti norimą procesų topologiją ir tikėtis, kad MPI realizacija suskirstys procesus į grupes optimalia tvarka.
- Platus apsikeitimo duomenimis funkcionalumas – nors visas apsikeitimas duomenimis gali būti realizuotas taikant 6 pagrindines MPI funkcijas (Čiegis 2005; Jakušev 2007), MPI 1.1 standartas apibrėžia daugiau kaip 120 funkcijų. Jas taikant galima patogiai programuoti įvairius apsikeitimus duomenimis ir tikėtis, kad jie bus efektyviai realizuoti. MPI standarto 1.1 versija buvo priimta 1995 metais ir netrukus realizuota daugelyje sistemų. 1997 metais buvo išleista 2.0 versija. Tačiau antroji versija yra žymiai sudėtingesnė už pirmąją, joje realizuota daugiau funkcijų, todėl jos pilna realizacija nėra plačiai paplitusi. Egzistuoja labai daug MPI standarto realizacijų, čia galima paminėti laisvai platinamas LAM/MPI ir MPICH.

Kalbant apie MPI ir C++ suderinamumą, MPI 1.1 standarte buvo apibrėžtos tik C ir FORTRAN kalbų sąsajos, tačiau netrukus atsirado ir C++ sąsajos (pavyzdžiui, OOMPI). MPI 2.0 standarte jau apibrėžta ir C++ sąsaja. Minimalus funkcijų sąrašas, skirtas realizuoti lygiagrečiuosius skaičiavimus su MPI parodytas 2.2 lentelėje.

2.5.3. JPPF – JAVA lygiagrečiųjų skaičiavimų standartas

JPPF (angl. *Java Parallel Processing Framework* – JAVA lygiagrečiųjų

skaičiavimų aplinka) – lygiagrečiųjų skaičiavimų standartas, skirtas realizuoti skaičiavimus kompiuterių tinkluose naudojant JAVA programavimo kalbą ir nereikalaujantis iš vartotojo išsamių lygiagrečiųjų skaičiavimų žinių. Be to, standartas leidžia vykdyti tarptinklinius skaičiavimus įvairiose klasterinėse sistemose, taip pat ir heterogeninėse (Lyndon *ir kt.* 1994).

2.2lentelė. *Tipinės MPI funkcijos*

Funkcija	Aprašas
MPI_Init	Sukuriamas virtualus lygiagretusis kompiuteris
MPI_Comm_size	Nustatomas procesų skaičius
MPI_Comm_rank	Nustatomas proceso numeris
MPI_Send	Siųsti pranešimą
MPI_Recv	Gauti pranešimą
MPI_Finalize	Visi procesai informuojami apie lygiagrečiojo darbo pabaigą

Privalumas – lengva naudoti. Paketas turi užduočių monitoringo ir pačių paketų tvarkymų įrankius, tokius kaip automatinis darbininkų perkonfigūravimas įvykus klaidai ir kiti.

Standartas realizuotas šeimininko–darbininkų principu. Vienas procesas skaičiavimų metu visada turi būti šeimininku, kiti procesai gali būti tik darbininkais. Skaičiavimai vykdomi viena iš dviejų strategijų: duomenų lygiagretusis paskirstymas ir apdorojimas ir funkcijų lygiagretinimas.

JPPF standartas yra atviras ir platinamas pagal Apache 2.0 licenciją. Aplinką gali keisti ir pritaikyti savo poreikiams bet koks JPPF vartotojas, be to licencija leidžia egzistuoti ir mokamoms JPPF aplinkos realizacijoms (Xiong *ir kt.* 2010).

2.5.4. CUDA – NVIDIA GPU lygiagrečiųjų skaičiavimų standartas

CUDA (angl. *Compute Unified Device Architecture*) – bendrosios paskirties lygiagrečiųjų skaičiavimų architektūra, kuri panaudoja Nvidia grafikos apdorojimo įrenginius (GPU) skaičiavimams su slankaus kablelio skaičiais (LI

2009). Palaikoma įranga GeForce 8 serija, Quadro FX 5600/4600 ir Tesla įrenginiai. Operacinės sistemos daugiaprogramino apdorojimo (multitasking) sistema yra atsakinga už prieigą prie GPU, kelių CUDA ir grafikos programų vienu metu. CUDA programinė įranga yra sudaryta iš kelių sluoksnių: techninės įrangos tvarkyklės, aplikacijų programavimo sąsajos (API) ir dviejų, bendrosios paskirties, aukšto lygio matematinių bibliotekų „CUFFT“ ir „CUBLAS“ (Боресков ir Харламов 2010).

Rašant programą su CUDA reikia atkreipti dėmesį į tai, kad programoje reikia numatyti veiksmus, kuriuos vykdo procesorius ir veiksmus, kuriuos vykdys GPU. Be šios sąlygos CUDA programos realizuoti nepavyks (Боресков ir Харламов 2010).

2.5.5. OpenCL – atviras GPU lygiagrečiųjų skaičiavimų standartas

OpenCL (angl. *Open Computing Language*) – atviras standartas, skirtas skaičiuoti lygiagrečiai heterogeninėse skaičiavimų sistemose. Tai viena pirmųjų atvirojo kodo tarp platforminio skaičiavimo aplinka, veikianti ne tik standartinės komplektacijos kompiuteriuose, serveriuose, bet ir nešiojamuose įrenginiuose. OpenCL platina nekomercinis konsorciumas Khronos group (Apple, AMD, Intel, nVidia, ARM, Sun Microsystems, Sony Computer ir kiti). Standartas kartu su pirma realizacija jau egzistuoja nuo 2009-04-20 ir pasirodė kartu su MacOS X 10.6. Dabartiniu metu egzistuojanti versija yra 1.2.

Kalba realizuota C99 bei API (angl. *Application Programming Interface* – taikomojo programavimo aplinka) standartų pagrindu. Bet turi savo savybes: nėra rodyklių į funkcijas, nėra realizuotos rekursijos, egzistuoja atminties valdymo konstrukcijos (_global, _local, _constant, _private), be to, yra realizuotos funkcijos, skirtos atlikti lygiagrečiuosius skaičiavimus (Tsuchiyama *ir kt.* 2010).

2.5.6. XMPP-MPI – atvira pranešimų perdavimo sąsaja

XMPP-MPI (angl. *Extensible Messaging and Presence Protocol* – išplėstas pranešimų siuntimo ir egzistavimo protokolas) – XMPP protokolo pagalba realizuotos pranešimų siuntimo funkcijos, naudojančios viešos prieigos serverius. Skaičiavimams realizuoti naudojama serverio-kliento programinė realizacija, o tai reiškia, kad reikalingos dvi programos.

Programa-serveris susijungia su viešos prieigos serveriu ir laukia programų-darbininkų. Sulaukus, paskirsto užduotis tarp darbininkų ir renka rezultatus.

Programa-darbininkas prisijungia prie viešos prieigos serverio ir tokiu būdu susijungia su programa-šeimininku. Programa-darbininkas gauna užduotis ir, išsprendus juos, siunčia rezultatus šeimininkui.

XMPP-MPI privalumas yra tai, kad realizacija yra daugiaterpė, nemokama, be to į skaičiavimų resursą galima sujungti didelį skaičių kompiuterių nepriklausomai nuo jų architektūros. Skaičiavimams realizuoti galima naudoti viešos prieigos serverius. Paskutinė versija palaiko ir CUDA skaičiavimus.

2.6. Lygiagretieji algoritmai

Dabartiniu metu mokslininkai sprendžia daugelį sudėtingų uždavinių. Kad būtų patikrinamos visos galimybės, reikia atlikti daug įvairių skaičiavimų. Ne visada tokie skaičiavimai trunka kelias valandas, pasitaiko uždavinių, kurių sprendimas trunka keliolika mėnesių arba net metų. Būtent dėl šios priežasties ir atsirado poreikis skaičiuoti lygiagrečiai, kad būtų sumažinamas skaičiavimų laikas. Lietuvoje jau yra apginta keletas daktaro disertacijų, kuriose tiriami lygiagretieji skaičiavimai, A. Baravykaitė kūrė šabloną, skirtą lygiagrečiųjų algoritmų sprendimui (Baravykaitė 2006), S. Ivanikovas taikė lygiagrečiuosius skaičiavimus daugiamačių duomenų vizualizavimui (Ivanikovas 2009), A. Jakušev kūrė lygiagrečiuosius algoritmus, skirtus diferencialinių lygčių ir jų

sistemų sprendimui (Jakušev 2007), R. Paulavičius taikė lygiagrečiuosius algoritmus globaliajam optimizavimui su simpleksiniais posričiais (Paulavičius 2010), J. Žilinskas lygiagretino padengimo metodus, skirtus juodosios dėžės globaliajam optimizavimui (Žilinskas J. 2002), T. Petkus ir po jo E. Filatovas tyrė lygiagretinimo metodus skirtus, daugiakriterinių uždavinių optimizavimui (Petkus 2001; Filatovas 2012) ir kiti (Medvedev 2007; Starikovičius 2002; Šablinskas 1999; Šilko 2003).

2.6.1. Užduočių paskirstymas lygiagrečiuosiuose skaičiavimuose

Užduočių paskirstymas procesams gali būti atliekamas dviem būdais:

- Pirmasis būdas – statinis paskirstymas skaičiavimo pradžioje, naudojantis informaciją apie užduočių sudėtingumą ir procesorių spartą.
- Antrasis būdas – dinaminis darbo paskirstymas skaičiavimo eigoje. Naudojamas tada, kai užduočių sudėtingumas yra kintamas ar neprognozuojamas ir procesorių greitaveika gali keistis skaičiavimo metu. Tokiuose algoritmuose realizuojamas darbo perskirstymo mechanizmas.

Pagal sprendžiamus uždavinius lygiagretieji algoritmai skirstomi į kelias klases:

- Duomenų lygiagretumo tipo algoritmai naudojami tada, kai yra daug apdorojamų duomenų, tie duomenys porcijomis paskirstomi tarp visų galimų kompiuterių ir apdorojami vienu metu.
- Funkcinio lygiagretumo algoritmai naudojami tada, kai apdorojant duomenis reikia atlikti daug viena nuo kitos nepriklausomų ir neįtakojančių vieną kitos operacijų. Tokių operacijų vykdymo trukmė apytiksliai turi būti vienoda. Tada jas galima išdalinti tarp visų skaičiavimo mazgų.

- Šeimininko-darbininkų tipo algoritmai gali būti taikomi visiems uždaviniams, kuriuose galima sudaryti vienas nuo kito nepriklausomų darbų sąrašą.

Darbo paskirstymo tarp procesų uždavinys, arba balansavimo uždavinys, yra vienas svarbiausių konstruojant efektyvius lygiagrečiuosius algoritmus (Ortega 1988).

Aptariant išsamiau šeimininko-darbininkų algoritmą reikia pabrėžti, kad algoritmas yra „vienas su daugeliu“ komunikacijos modelį naudojantis algoritmas, t. y. vienas procesas, vadinamas šeimininku, komunikuoja su daugeliu kitų procesų, vadinamų darbininkais. Darbininkai tarp savęs informacija nesikeičia. Šis apribojimas atmeta didelę uždavinių klasę, kurioje lygiagrečiai sprendžiamos užduotys reikalauja informacijos iš kaimyninių užduočių.

Išvardinti užduočių paskirstymo metodai geri, bet ne visada padeda išvengti papildomų išlaidų. Taip žaliosios kompiuterijos tematikoje atsirado efektyvaus užduočių pasiskirstymo tarp skaičiavimo mazgų (Mair *ir kt.* 2010; Zhang ir Qi 2006; Wang ir Zhang 2011; Zhang *ir kt.* 2010 a), efektyvaus kompiuterių procesorių apkrovimo sąvokos (Chakraborty *ir kt.* 2009; Rathore ir Chana 2011) ir k. t.

2.6.2. Lygiagrečiųjų algoritmų vertinimo kriterijai

Pirmasis kriterijus, su kuriuo galima palyginti kelis lygiagrečiuosius algoritmus, yra *lygiagrečiojo algoritmo spartinimo koeficientas* sprendžiant uždavinį su p procesorių (Čiegis 2005):

$$S_p = \frac{T_1^*}{T_p}, \tag{2.1}$$

čia T_1^* yra laikas, per kurį uždavinys išsprendžiamas geriausiu žinomu nuosekliuoju algoritmu, o T_p– laikas, kurį sugaišta lygiagretusis algoritmas, spręsdamas uždavinį su p procesorių.

Pažymėsime, kad T_1^* ir T_1 dažniausiai skiriasi, kadangi ne visi nuoseklieji algoritmai gali būti visiškai išlygiagretinami, tada greičiausias nuoseklusis algoritmas skiriasi nuo lygiagrečiojo. Be to, net tais atvejais, kai lygiagretusis algoritmas yra sudarytas modifikuojant nuoseklųjį, atsiranda papildomos programinės konstrukcijos, lėtinančios algoritmą, todėl visuomet yra teisinga nelygybė $T_1^* \leq T_1$. Taip pat galima pastebėti, jog skirtingi lygiagretieji algoritmai, sprendžiantys tą patį uždavinį skirtingais būdais, gali būti pranašesni arba prastesni keičiantis procesorių skaičiui, be to tam gali daryti įtaką lygiagretusis kompiuteris.

Kitas kriterijus, kuriuo galima vertinti lygiagretųjį algoritmą, yra *lygiagrečiojo algoritmo efektyvumo koeficientas*:

$$E_p = \frac{S_p}{p}. \tag{2.2}$$

Efektyvumas parodo, kaip naudojami procesoriai. Sprendžiant kai kuriuos uždavinius, didelis efektyvumas gali būti pasiektas tik esant fiksuotam procesorių kiekiui, tačiau toliau jam didėjant, efektyvumas mažėja. Efektyvumas priklauso ne tik nuo uždavinio, bet ir nuo lygiagrečiojo algoritmo bei lygiagrečiųjų kompiuterių parametrų: granuliacijos (duomenų grupavimas), procesorių skaičiavimo greičio, komunikacijos greičių ir pan. To paties algoritmo efektyvumas skirtinguose kompiuteriuose gali labai skirtis. Paprastai kuo didesnis procesorių komunikacijos greitis ir kuo lėtesni procesoriai, tuo geresnis efektyvumas yra pasiekiamas (Čiegis 2005).

Dar vienas kriterijus, kurį galima naudoti lygiagrečiųjų algoritmų vertinimui yra *procesorių panaudojimo efektyvumas* (*utilizacija*) (Thiebaut 1995):

$$U = \frac{1}{p}\sum_{i=1}^{p}\frac{t_i}{T_p}, \tag{2.3}$$

čia p – bendras skaičiavimams panaudotas procesų skaičius, t_i – i-tojo procesoriaus skaičiavimų (neapimant pranešimų siuntimo ir laukimo) laikas, T_p

– visas užduoties vykdymo laikas.Utilizacija parodo kaip gerai skaičiavimai apkrauna procesorius, tai geras rodiklis parodantis lygiagrečiojo algoritmo efektyvumą. Jei algoritmas realizuotas labai gerai, tai procesorių panaudojimas skaičiavimams bus arti vieneto.

2.6.3. Tipiniai lygiagretieji algoritmai

Dauguma realių uždavinių (orų prognozės ir klimato pokyčių modeliavimas, optimizavimo uždaviniai) yra sudėtingi ir dažniausiai kompleksiniai, jie susideda iš daugelio paprastesnių uždavinių. Patys paprasčiausieji ir pakankamai lengvai išlygiagretinami iš jų: dalinių sumų skaičiavimo uždaviniai, matricų sandauga(atskiras atvejis – su vektoriumi), sudėtis, tiesinių lygčių sistemos sprendimas (Bronshtein *ir kt.* 2004).

Tiesinių lygčių sistemų sprendimas

Tiesinės lygties bendras pavidalas:

$$a_1x_1 + a_2x_2 + \cdots a_nx_n = b. \tag{2.4}$$

Tiesinių lygčių sistemos (TLS) bendras pavidalas:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n-1}x_{n-1} + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n-1}x_{n-1} + a_{2,n}x_n = b_2 \\ \vdots \quad\quad \cdots \quad\quad \vdots \\ a_{n-1,1}x_1 + a_{n-1,2}x_2 + \cdots + a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{n,1}x_1 + a_{n-1,2}x_2 + \cdots + a_{n,n-1}x_{n-1} + a_{n,n}x_n = b_n \end{cases}. \tag{2.5}$$

Vienas iš Gauso metodo etapų vadinamas tiesiogine algoritmo eiga, jo metu apskaičiuojamas elementas x_n.

Po to prasideda atbulinė algoritmo eiga. Rezultatas gaunamas elementariųjų pertvarkymų būdu. Taip gaunama pertvarkyta tiesinių lygčių sistema (2.6). Tokiu būdu apskaičiuojami visi x_i elementai, kai i = 1, 2, 3, …, n.

$$\begin{cases} a'_{0,0}x_0 + 0 & + \cdots + 0 & + 0 & = b'_0 \\ 0 & + a'_{1,1}x_1 + \cdots + 0 & + 0 & = b'_1 \\ & \vdots & \vdots & \\ 0 & + 0 & + \cdots + a'_{n-2,n-2}x_{n-2} + 0 & = b'_{n-2} \\ 0 & + 0 & + \cdots + 0 & + a'_{n-1,n-1}x_{n-1} = b'_{n-1} \end{cases}. \quad (2.6)$$

Matricos ir vektoriaus sandauga

Kaip matome, vektorius yra taip pat matrica, tik vieno stulpelio. Matricos ir vektoriaus sandauga apskaičiuojama taip:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} a_{1,i} \times b_i \\ \sum_{i=1}^{n} a_{2,i} \times b_i \\ \sum_{i=1}^{n} a_{3,i} \times b_i \\ \vdots \\ \sum_{i=1}^{n} a_{m,i} \times b_i \end{pmatrix}. \quad (2.7)$$

Lygiagrečiųjų skaičiavimų teorijoje matricos ir vektoriaus sandaugą apskaičiuoti galima keliais būdais: perduodant darbininkui matricos A eilutę ir vektorių b, perduodant darbininkui matricos A stulpelį ir vektorių b ar perduodant darbininkui matricos A ir vektoriaus b fragmentus.

Dviejų matricų sandauga

Dviejų matricų sandauga išreiškiama taip (Golub ir Van Loan 1996):

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,n} \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \cdots & c_{n,n} \end{pmatrix} \quad (2.8)$$

Kiekvienas matricos C elementas apskaičiuojamas taip

$$c_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + a_{i,3}b_{3,j} + \cdots + a_{i,n}b_{n,j}. \quad (2.9)$$

Galimos įvairios lygiagrečiosios matricų sandaugos realizacijos:

- juostinis duomenų paskirstymas;

- duomenų paskirstymas blokais:
  - Fox′o algoritmas;
  - Cannon′o algoritmas.

*Fox′*o ir *Cannon′*o algoritmas skiriasi tik duomenų pasiskirstymu tarp procesų. Duomenų pasiskirstymas blokais kai procesų skaičius yra $p = s \times q$. Be to, matricos eilučių skaičius turi būti s kartotinis, o stulpelių skaičius – q, reiškia $m = k * s$ ir $n = l * q$:

$$\begin{pmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n} \end{pmatrix} \times \begin{pmatrix} B_{1,1} & \cdots & B_{1,n} \\ \vdots & \ddots & \vdots \\ B_{n,1} & \cdots & B_{n,n} \end{pmatrix} = \begin{pmatrix} C_{1,1} & \cdots & C_{1,n} \\ \vdots & \ddots & \vdots \\ C_{n,1} & \cdots & C_{n,n} \end{pmatrix}. \quad (2.10)$$

Kiekvienas matricos C blokas, $C_{i\ j}$, gaunamas taip:

$$C_{i,j} = \sum_{s=1}^{q} A_{i,s} \times B_{s,j}. \quad (2.11)$$

Blokų skaičius yra vienodas tiek vertikaliai, tiek horizontaliai.

2.7. Kombinatorinis optimizavimas

Spręsdami optimizavimo uždavinius, turime tikslą rasti geriausią sprendinį (Diwekar 2008; Floudas ir Pardalos 2009). Matematiškai optimizavimo uždavinys yra formuluojamas taip:

$$f^* = \min_{X \in D} f(X), \quad (2.12)$$

čia f – tikslo funkcija, f^* – jos minimumas, X – kintamieji, D – leistinoji aibė/sritis.

Kombinatorinio optimizavimo uždavinių atveju operuojama su diskretaus tipo kintamaisiais arba kombinatorinėmis struktūromis (pvz., sveikaisiais skaičiais, sveikųjų skaičių rinkiniais, perstatymais, poaibiais, grafais, kėliniais, deriniais, gretiniais ir pan.) (Nocedal ir Wright 2000). Kintamųjų reikšmių aibė šiuo atveju yra baigtinė ar bent jau suskaičiuojama. Tokių uždavinių

sprendimui taikomi kombinatorinio optimizavimo metodai. Kombinatorinis optimizavimas naudojamas daugelyje taikomųjų sričių, tokių kaip: dirbtinis intelektas (Poole *ir kt.* 1998; Russell ir Norvig 2003), mašininis mokymas (Witten ir Frank 2011), taikomoji matematika (Brimkov *ir kt.* 2012), lažybų teorija (Papadimitriou ir Roughgarden 2008), programų inžinerija (Piff 1991) ir kt. Tikslo funkcijos pobūdis ir išraiška priklauso nuo konkretaus sprendžiamo uždavinio. Labai dažnai kombinatorinio optimizavimo uždavinių tikslo funkcijos yra netiesinės, neiškilos, nediferencijuojamos ir daugiaekstremės. Iš (2.12) formulės seka, kad optimizavimo metu yra ieškomas tikslo funkcijos minimumas. Tokiu būdu sprendžiant optimizavimo uždavinį ieškoma tokio sprendinio X^*, kuris atitiktų sąlygą:

$$X^* \in D^* \subseteq D, \tag{2.13}$$

čia sprendinys X^* vadinamas uždavinio optimaliu sprendiniu, o D^* – optimalių sprendinių aibė, kur $D^* = \{x: f(x) = f^*\}$.

Kombinatoriniams optimizavimo uždaviniams spręsti yra naudojami kombinatoriniai sprendimo metodai (Kreher ir Stinson 1998):

- Generavimo metodai sukonstruoja aibę galimų sprendinių (priklauso nuo uždavinio specifikos). Tokiu būdu gaunama optimalių sprendinių aibė (D^*), kuri yra leistinosios sprendinių aibės (D) poaibis.
- Perrinkimo metodai perrenka visus galimus sprendinius, tokiu būdu gauname visą leistinąją galimų sprendinių aibę D. Generavimo metodai tuo pačiu priklauso ir perrinkimo metodų klasei, nes kiekvieną perrinkimo metodu gautą sprendinį galima gauti ir kuriuo nors generavimo metodu. Bet atvirkštinis teiginys negalioja.
- Paieškos metodai randa bent vieną sprendinį, atitinkantį sąlygas. Generavimo metodus galima taikyti sprendinių paieškai, bet dažnai tokie metodai yra neefektyvūs. Praktikoje kur kas lengviau rasti ieškomą sprendinį paieškos metodais negu perrinkti ar sugeneruoti visų

galimų sprendinių aibę.

Paieškos metodai gali būti adaptyvūs ir neadaptyvūs. Neadaptyvūs metodai vykdomi dažniausiai neatsižvelgiant į prieš tai gautus rezultatus. Šiuos metodus tyrė daugelis mokslininkų. Jie labai paprasti, bet ir nepakankamai efektyvūs. Bet didelis privalumas yra tai, kad jie paprastai lygiagretinami. Kiekvienas iš procesorių gauna savo užduoties dalį (kurios dydis yra vienodas visiems procesoriams) ir vykdo algoritmą nepriklausomai nuo kitų. Kadangi procesoriai neapsikeitinėja tarpusavyje informaciniais pranešimais, tai algoritmo spartinimas paprastai yra lygus procesorių skaičiui, o efektyvumas lygus vienetui.

Adaptyvūs atsitiktinės paieškos metodai atsižvelgia į ankstesnių bandymų metu gautus rezultatus. Geriausios surastos funkcijos reikšmės nurodo daugiau žadančias sritis. Adaptyviųjų atsitiktinių metodų klasei priklauso genetiniai algoritmai, evoliucinės strategijos, atkaitinimo modeliavimo metodai. Adaptyviųjų paieškos metodų efektyvumas tradiciškai priklauso nuo jų parametrų reikšmių parinkimo, todėl geriausiai veikia naudojami jų autorių (Paulavičius 2010).

Kombinatoriniams optimizavimo metodams priklauso šakų ir rėžių algoritmai, kurie priskiriami prie perrinkimo metodų. Šakų ir rėžių algoritmai suranda sritis, kuriose nėra globaliojo minimumo ir pašalina tokias sritis iš tolimesnės paieškos. Problema – reikia griežtai įvertinti rėžius. Rėžių įvertinimui, pagal kuriuos nustatomos šalintinos sritys, dažniausiai reikalinga papildoma informacija. Tačiau tokio tipo informacija ne visada iš anksto žinoma sprendžiant praktinius uždavinius.

Keli kombinatorinio optimizavimo uždavinių pavyzdžiai:

- Transporto trumpiausio maršruto paieškos uždavinys (Cherkassky *ir kt.* 1994);
- Keliaujančiojo prekeivio uždavinys (Woeginger 2003);

- Minimalaus jungimo medžio uždavinys (Graham ir Pavol 1985);
- Sveikaskaitinio programavimo uždaviniai (Llewellyn 1993; Klamroth *ir kt.* 2004);
- Kuprinės pakavimo uždaviniai (Gallo *ir kt.* 1980).

2.8. Antrojo skyriaus apibendrinimas ir išvados

Šiame skyriuje buvo pristatyta žaliosios kompiuterijos sąvoka. Taip pat apžvelgta klasterių ir superkompiuterių elektros energijos suvartojimo mažinimo problema. Dauguma šios problemos tyrinėtojų teigia, jog elektros sąnaudų mažinimo metodai, skirti asmeniniams kompiuteriams, gali būti pritaikyti ir klasteriams bei superkompiuteriams. Taigi siūloma atsisakyti nebūtinų, papildomai prijungtų, techninių įrenginių, pavyzdžiui, vaizduoklių, išorinių įrenginių, kurie reikalauja papildomo maitinimo, ir t. t. Be to, yra siūloma naudotis kompiuterinės technikos gamintojų numatytais elektros energijos sąnaudų mažinimo būdais, būtent, kietųjų diskų ir kompiuterių laikinas išjungimas. Vis dėlto pats veiksmingiausias būdas – išjunginėti kompiuterius, kai jie nėra reikalingi.

Be to, šiame skyriuje yra atlikta lygiagrečiųjų skaičiavimų technologijų analizė: aprašyti lygiagrečiųjų skaičiavimų kompiuteriai ir programinės technologijos, skirtos skaičiuoti lygiagrečiai. Standartų ir realizacijų yra gana daug, o dauguma iš jų kuriama kaip C++ kalbos praplėtimas. Todėl buvo nuspręsta, kad tolimesniuose tyrimuose visos programos bus realizuotos C++ programavimo kalba su OpenMP ir MPI bibliotekomis, skirtomis lygiagrečiųjų programų kūrimui. Disertacijos kūrimo metu autorius dalyvavo projekte LitGrid ir administravo Vilniaus universiteto Matematikos ir informatikos instituto klasterį. Dėl to buvo galimybė vykdyti eksperimentus dviejų tipų lygiagrečiuose kompiuteriuose: gride ir klasteryje.

Klasterių ir superkompiuterių elektros energijos sąnaudoms tirti yra reikalingos užduotys, kurios apkrautųjų skaičiavimo procesorius. Tam tikslui bus naudojamos optimizavimo algoritmus realizuojančios programos. Todėl šiame skyriuje buvo nagrinėjamas ir kombinatorinis optimizavimas. Vėliau bus taikomas vienas iš kombinatorinio optimizavimo metodų – šakų ir rėžių optimizavimo algoritmas. Sprendžiant kombinatorinio optimizavimo uždavinius yra reikalingos didelės laiko sąnaudos ir dideli skaičiavimo resursai. Dėl šių priežasčių, jiems spręsti ir yra naudojami superkompiuteriai, klasteriai, gridai ir kiti lygiagretieji kompiuteriai, pasitelkiant uždavinių sprendimui lygiagrečiųjų skaičiavimų žinias.

3

Elektros energijos suvartojimo tyrimo ir mažinimo būdai lygiagrečiųjų skaičiavimų klasteriams, lygiagretieji šakų ir rėžių algoritmai

Pasaulyje kompiuterių elektros energijos suvartojimo problema yra ypač akcentuojama nuo 2005 metų, kai įvesta žaliosios kompiuterijos sąvoka. Tuo tarpu Lietuvoje šiai problemai skiriamas nepakankamas dėmesys. Įvairūs autoriai siūlo įvairus elektros energijos suvartojimo mažinimo būdus, bet ne visada jie yra viešai prieinami (būdai yra paminėti ir aprašyti 2 skyriuje). Klasteriuose dažniausiai įdiegti įrankiai, kurie turi taupyti elektrą, veikia tik kompiuteryje-šeimininke arba jų nėra. Dėl to autorius siūlo metodiką, kaip atlikinėti tokius tyrimus. Disertacijoje taip pat siūloma naudoti elektros energiją tausojančius būdus skirtus lygiagretiesiems ir paskirstytiesiems kompiuteriams. Vienas iš tokių būdų pasiūlytas šiame skyriuje.

Kaip jau buvo minėta 2 skyriuje, klasterių ir gridų elektros energijos suvartojimui tirti reikalingi atitinkami uždaviniai, kurie apkrauna kompiuterių procesorius. Tokiems uždaviniams priklauso optimizavimo metodų

realizacijos. Dėl didelio skaičiavimų kiekio optimizavimo uždaviniams spręsti naudojami lygiagretieji ir paskirstytieji skaičiavimai. Šiame skyriuje aprašomas pasirinktas taikomasis uždavinys (strypinių konstrukcijų optimizavimas), kurį sprendžiant bus tiriami nuoseklieji ir lygiagretieji algoritmai, lygiagretieji ir paskirstytieji kompiuteriai. Šio skyriaus rezultatai publikuoti straipsniuose (A 2, A 3, A 4, B 1).

Pastaruoju metu lygiagrečiųjų skaičiavimų tematika yra labai aktuali. Lietuvoje per pastaruosius metus yra apginta nemažai daktaro disertacijų lygiagrečiųjų skaičiavimų tematikoje (Baravykaitė 2006; Filatovas 2012; Ivanikovas 2009; Medvedev 2007; Paulavičius 2010; Petkus 2001; Starikovičius 2002; Šablinskas 1999; Šilko 2003; Žilinskas J. 2002). Tačiau minėtuose disertacijose nebuvo tiriama žaliosios kompiuterijos tematika.

3.1. Siūloma elektros energijos suvartojimo tyrimo metodika

Pastaruoju metu elektros energijos suvartojimo optimizavimas tampa labai svarbus. Moksliniuose tyrimuose naudojami skirtingi kompiuteriniai ištekliai: superkompiuteriai, klasteriai ir klasteriniai tinklai, debesų kompiuterija. Visi resursai vartoja daug elektros energijos ir pastaruoju metu labai aktualus uždavinys – sumažinti elektros energijos suvartojimą nemažinant skaičiavimų pajėgumų.

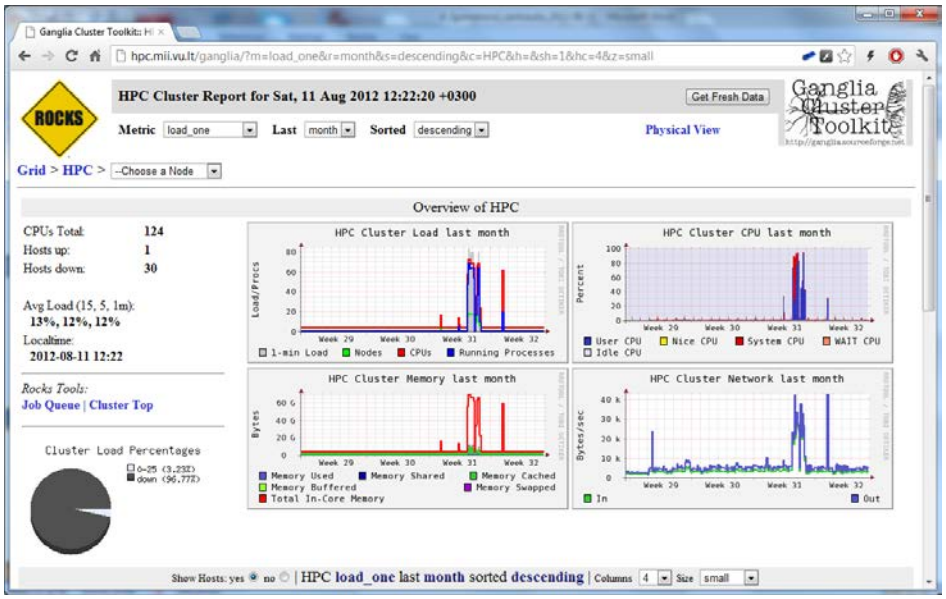
Kaip parodė žodinė LitGrid klasterių administratorių apklausa, elektros energijos suvartojimo problemai Lietuvoje skiriamas nepakankamas dėmesys arba ji net visai netiriama. Klasteriuose dažniausiai įdiegti įrankiai, kurie turi taupyti elektrą, veikia tik kompiuteryje-šeimininke arba jų nėra.

Tyrimai, parodantys grid sistemų ir klasterių energijos suvartojimo neefektyvumą, plačiau aprašyti 4 skyriuje.

Tyrimams atlikti buvo pasiūlyta metodika: vykdomos lygiagrečiosios užduotys, fiksuojami elektros galia ir bendras elektros energijos suvartojimas atliekant skaičiavimus, be to klasterio apkrovos stebėjimų sistemoje buvo fiksuojamas bendras klasterio apkrovimas. Elektros galia parodo kiek matavimų metų elektros galios reikalauja klasteris. Bendras elektros energijos suvartojimas parodo kiek elektros energijos klasteris suvartojo per tam tikrą laiką, pagal matuoklio rodmenis.

Tiems tikslams realizuoti naudojami: klasterių užduočių paleidimo aplinka, elektros matavimo prietaisas „UPM MP300 Energy Meter“ (plačiau apie jį 4.6 skyriuje), nuotolinio stebėjimo sistema susidedanti iš kameros „ACME PC Camera CA09 1 300 pikselių“ (3.1 paveiksle parodytas kameros stebimas matuoklis) ir sistemos apkrovimo tinklalapio GANGLIA (3.2 paveikslas) (University of California ir Berkeley MillenniumProject 2010). Sukurti scenarijų failai vaizdo transliacijai sukurti ir atkurti.

3.1 pav. *GSTlaunch-0.10 – UPM MP300 Energy Meter rodmenų stebėjimo sistema*



3.2 pav. *GANGLIA klasterio stebėjimų sistema*

3.2. Elektros energijos suvartojimo mažinimas klasteriams

Tyrimų metu atliekant eksperimentus gride buvo pastebėta, kad grido klasteriai ir jų kompiuteriai ne visada pilnai apkrauti. Darbo metu užduotys, vykdomos grido infrastruktūroje, neoptimaliai pasiskirsto tarp kompiuterių, tokiu būdu ne pilnai juos apkrauna. T. y. užduotys gali būti vykdomos ne su vienu daugiabranduoliniu kompiuteriu, o su keliais (nors užtektų ir vieno). Taip pasitvirtino kitų mokslininkų tyrimai apie neefektyvų užduočių pasiskirstymą gride (plačiau pristatyta 2 skyriuje).

Klasterinės sistemos gana efektyvios, nes nenaudoja vaizduoklių. Viso klasterio administravimui ir priežiūrai užtenka tik interneto ryšio. Vaizduoklis reikalingas atliekant instaliavimo ar derinimo darbus, o tai atsitinka labai retai. Dauguma klasterio vartotojų jungiasi prie kompiuterio-šeimininko per internetą ir registruoja savo užduotis. Bet buvo pastebėta, kad klasterio darbo metu kai jis mažai apkraunamas užduotimis, klasterio kompiuteriai-darbininkai ilgą laiką stovi apkrauti tik sisteminiais procesais (angl. *Idle State* – liet. tuščia

eiga), t. y. nevykdant jokių naudingų skaičiavimų. Tokiu būdu klasterio kompiuteriai-darbininkai beprasmiškai vartoja elektros energiją.

Pastebėjus tokį klasterių veikimą yra siūloma nauja klasterių darbo strategija. Atsižvelgiant į patarimus, aprašytus 2.1 skyriuje, siūloma automatiškai išjungti kompiuterius-darbininkus, kai jie nereikalingi skaičiavimams. Tokiu būdu galima priversti klasterines sistemas vartoti kur kas mažiau elektros ne skaičiavimų, o klasterio prastovos (kai kompiuteriai-darbininkai laukia iš tinklo užduoties) metu. Vienintelis reikalavimas klasterio kompiuteriuose, BIOS sistemoje turi būti nurodyta, kad tinklo plokštė visą laiką yra įjungta. Tai daroma dėl to, kad tinklo plokštė turi laukti kompiuterio žadinimo komandos iš kompiuterio-šeimininko.

Šiems tikslams pasiekti buvo sukurti ir įdiegti į kompiuterius scenarijų failai 3.1 ir 3.2. Failai sukurti naudojant Linux Bash komandas. Dėl to jie yra gana universalūs ir gali būti naudojami visuose klasteriuose su Linux tipo OS. Tam, kad darbo scenarijus veiktų mazge-šeimininke, jame turi būti įdiegtas papildomas paketas WAKEONLINE (Oliveira 2005). Paketas WAKEONLINE leidžia pažadinti bet kokį tinklo kompiuterį, žinant tik jo tinklo plokštės fizinį adresą (MAC adresą). Be to, scenarijų paleidimui mazguose-darbininkuose ir mazge-šeimininke turi veikti CRON (The IEEE ir The Open Group 2008) darbų paleidimo tarnyba. Taip pat, kaip buvo minėta, mazguose-darbininkuose BIOS sistemoje turi būti nurodyta, kad tinklo plokštė visą laiką yra įjungta. Tai būtinos sąlygos, kad klasteryje veiktų 3.1 scenarijus.

Abu scenarijai yra įdiegti į CRON tarnybą ir reguliariai paleidžiami, 3.1 scenarijus paleidžiamas kas 5 minutes, kad klasterio vartotojams nereikėtų ilgai laukti užduočių paleidimo. 3.2 scenarijus paleidžiamas karta per vieną valandą. Tai daroma siekiant pratęsti kietojo disko tarnavimo trukmę nes blogiausiu atveju kompiuteriai, o su jais ir kietieji diskai, bus išjungiami kas valandą.

3.1 scenarijus. *Kompiuterių-darbininkų žadinimo scenarijus*

```
#!/bin/bash // nurodoma darbo aplinka ir jos parametrai
#export PATH=/bin:/bin:/opt/gridengine/bin/lx26-x86 //nurodomas kelias iki papildomų aplinkos failų
. /etc/profile.d/sge-binaries.sh //taip pat bash scenarijaus failas su papildomais aplinkos failais
if [ `qstat -u '*' | grep -c 'laukimo_eile'` != '0' ]; // bash aplinkos funkcija if ieškanti, kiek yra klasteryje užduočių laukimo eilėje, t.y. užduotys, kurioms paleisti neužtenka dabartiniu metu veikiančių resursų
then
        echo "Paleidžiu kompiuterius" >> /tmp/log_pal.log; // scenarijaus veikimo būklės išvedimas į failą
        wakeonlan XX:XX:XX:XX:XX:XX; // paleidžiamų kompiuterių sąrašas
else
        echo "Papildomų kompiuterių nereikia" >> /tmp/log_pal.log; // scenarijaus veikimo būklės išvedimas į failą;
fi
```

3.2 scenarijus. *Kompiuterių-darbininkų išjungimo scenarijus*

```
#!/bin/sh
if [ -z "$(ps ax | grep -v grep | grep 'programa'" ]; // funkcija if, tikrinanti ar procesas su vardu „programa“ yra paleistas ir, jei ne, kompiuteris išjungiamas; parametras -v grep | grep 'programa' leidžią iš rezultatų išmesti eilutę su komandą grep.
then
        shutdown -h now ; // kompiuterio išjungimo komanda
fi
```

Kietųjų diskų gamintojų duomenimis (www.seagate.com, www.samsung.com), kieteji diskai pasižymi tokia savybe, kaip įjungimo / išjungimo ciklų skaičius iki blogųjų sektorių atsiradimo po grubaus ir normalaus kompiuterio išjungimo. Kietasis diskas paprastai ir taisyklingai išjungiant kompiuterį išlaiko iki 200 000 tokių išjungimų. Kai kompiuteris

išjungiamas netaisyklingai – tas rodiklis krenta iki 50 000. Jei kieteji diskai išjungiami taisyklingai kartą per valandą, tai tokiu darbo režimu jų darbo trukmė bus:

200 000 kartų / 24 valandų / 365 dienų = 22,8 darbo metų.

Iš to galima padaryti išvadą, kad siūloma darbo strategija mažai kenkia kietiesiems diskams.

Siūloma klasterių darbo strategija pasiteisina tik tada, kai klasteriai mažai apkrauti, t.y. turi mažai vartotojų bei mažai sprendžia taikomąsias užduotis. Klasteriams, kurie yra stipriai apkrauti, tokios strategijos taikymas ne visada gali padėti sumažinti elektros energijos suvartojimą.

3.3. Strypinių konstrukcijų optimizavimo uždavinys

Strypinės konstrukcijos – tai objektai sudaryti iš nustatytų koordinačių mazgų, sujungtų tarpusavyje įvairių parametrų strypais. Konstrukcijos gali būti įvairios, gali turėti judamų ir nejudamų atramų, gali turėti vieną ar kelis jėgos veikimo taškus. Tokios konstrukcijos pavyzdžiai yra tiltai, statybiniai kranai, stogų perdengimo ir palaikymo sistemos, reklaminiai skydai su šoniniu tvirtinimu.

Tokių sistemų optimizavimui galima naudoti įvairius optimizavimo algoritmus: genetinius algoritmus(Šešok 2006; Šešok 2008 a; Šešok 2008 b; Šešok ir Belevičius 2007 a; Šešok ir Belevičius 2007 b; Šešok ir Belevičius 2008; Šešok ir Ragauskas 2007), atkaitinimo modeliavimo (angl. *Simulated Annealing*), šakų ir rėžių (angl. *Branch and Bound*).

Matematiškai strypinių konstrukcijų optimizavimo uždavinys formuluojamas taip:

$$\min f(X) = \sum_{i=1}^{n} x_i L_i \rho_i A_i , \tag{3.1}$$

$$g(X) = 0,$$

$$h(X) \leq 0,$$

$$x_i \in \{0,1\},$$

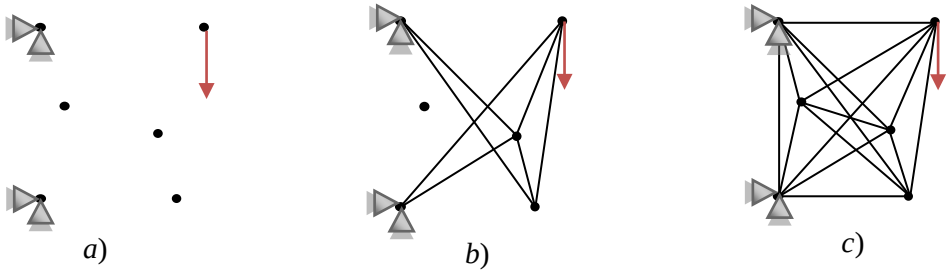
čia tikslo funkcija $f(X)$ yra konstrukcijos masė, kintamasis x_i koduoja i-ojo strypo buvimą konstrukcijoje, L_i – i-ojo strypo ilgis konstrukcijoje, ρ_i – strypo medžiagos tankis, A_i – atitinkamo strypo skerspjūvio plotas, n – visų galimų strypų skaičius konstrukcijoje, $n = m(m-1)/2$, kur m – mazgų skaičius. Pusiausviros lygybė išreiškiama per apribojimų funkciją $g(X)$ su sąlyga, kad visų jėgų veikiančių mazgą suma turi būti lygi 0. Tada konstrukcija yra lokaliai pusiausvyroje. Apribojimų funkcija $h(X)$ įvertina įtempimus mazge ir, kad konstrukcija išliktų lokaliai stabili, įtempimai neturi viršyti leistinų reikšmių. Apribojimų funkcijos $g(X)$ ir $h(X)$ yra modeliuojamos baigtinių elementų metodo pagalba.

Tikrinant kiekvieną konstrukciją, reikia spręsti tiesinių lygčių sistemą bei atlikti matricų daugybą. Tiesinių lygčių sistemos ir matricų daugybos skaičiavimai taip pat gali būti realizuojami lygiagrečiųjų technologijų pagalba.

Ieškant optimalios konstrukcijos perrinkimo būdu, reikia patikrinti visas galimas konstrukcijas. Galimų konstrukcijų skaičius auga eksponentiškai nuo mazgų (m) ir galimų strypų (n) skaičiaus ... $= 2^{\mathrm{n}} = 2^{\mathrm{m}\times(\mathrm{m}-1)/2}$. Perrinkimui pagreitinti gali būti naudojami lygiagretieji skaičiavimai.

Strypinių konstrukcijų pavyzdžiai pateikti 3.3 pav. Paveikslėlis 3.3 (a) vaizduoja konstrukciją be jokių strypų, yra tik 6 mazgai. Iš jų tik du mazgai yra įtvirtinti (atraminiai mazgai, įtvirtinimą schemoje vaizduoja trikampiai), viename iš mazgų pridėta veikimo jėga (rodyklė žemyn). Kiti mazgai parodo galimų strypų jungimo taškus (strypų sujungimo mazgai). Paveikslėlyje 3.3 (b) pavaizduota optimali konstrukcija su minimalia mase, konstrukcija yra iš 7

strypų. Paveikslėlis 3.3 (c) parodo konstrukciją su visais galimais strypais, bet tokia konstrukcija neoptimali.



3.3 pav. *Strypinių konstrukcijų pavyzdžiai: a) konstrukcija be strypų, b) optimali konstrukcija su 7 strypais ir c) konstrukcija su visais galimais strypais*

3.3.1. Pilnojo perrinkimo algoritmas, skirtas strypinių konstrukcijų optimizavimui

Paprasčiausias algoritmas, kurio pagalba galima spręsti kombinatorinius optimizavimo uždavinius, yra visų galimų sprendinių perrinkimas. Strypinių konstrukcijų atveju, konstrukcija koduojama binariniu pavidalu, tai kombinatoriniai algoritmai tinka tokių konstrukcijų optimizavimui. Lengviausia konstrukcija, tenkinanti apribojimus, yra optimali.

1 algoritmas realizuotas pilnojo perrinkimo metodo pagalba ir skirtas strypinių konstrukcijų optimizavimui. Sprendimo eigoje yra skaičiuojami apribojimai. Jei apribojimai nėra pažeisti ir tikslo funkcijos reikšmė yra mažesnė negu iki tol žinoma, tai ta reikšmė yra įsimenama, be to yra įsimenama ir binarinio vektoriaus X reikšmė. Baigus vykdyti algoritmą, minimali tikslo funkcijos reikšmė yra f^*, ją atitinkanti konstrukcija yra užkoduota vektoriuje X^*.

Vienas iš būdų paspartinti 1 algoritmo sprendimą – algoritmo lygiagretinimas. Ciklas **„for“** gali būti išlygiagretinamas, kai įvairių konstrukcijų tikrinimas atliekamas skirtingais nepriklausomais procesoriais (arba daugiaprocesorinių procesorių branduoliais) tuo pačiu metu. Dažniausiai pilnojo perrinkimo algoritmas gerai lygiagretinamas (Žilinskas J. 2009).

1 algoritmas. *Nuoseklusis pilnojo perrinkimo algoritmas*

$f^* = \infty$

for $\forall X \in \{(0,\dots,0,0),(0,\dots,0,1),\dots,(1,\dots,1,1)\}$ **do**

  if $f(X) < f^*$ **and** $g(X) = 0$ **and** $h(X) \leq 0$ **then**

    $f^* = f(X)$

    $X^* = X$

  end if

end for

2 algoritmas. *Lygiagretusis pilnojo perrinkimo algoritmas su OpenMP*

$f^* = \infty$

Include: *omp.h*

#pragma omp parallel for

for $\forall X \in \{(0,\dots,0,0),(0,\dots,0,1),\dots,(1,\dots,1,1)\}$ **do**

  if $g(X) = 0$ **and** $h(X) \leq 0$ **then**

    #pragma omp critical

      if $f(X) < f^*$ **then**

        $f^* = f(X)$

        $X^* = X$

      end if

  end if

end for

2 algoritme realizuota pilnojo perrinkimo algoritmo lygiagrečioji versija. OpenMP pragma „#pragma omp parallel for“ naudojama lygiagrečiojo ciklo identifikavimui, „#pragma omp critical“ naudojama apsaugoti kintamojo f^* tikrinimą bei pakeitimą algoritmo vykdymo eigoje. Efektyvesnės algoritmo versijos naudoja lokaliuosius kintamuosius f^* ir X^* ciklo eigoje, kad nebūtų galimas duomenų įrašymo konfliktas. Tokiu atveju rezultatai surenkami tik algoritmo pabaigoje. Pavyzdžiui MPI versijoje, pateiktoje 3 algoritme, visi kintamieji yra lokalieji, o skaičiavimų pabaigoje rezultatai surenkami ir apibendrinami 0 procesoriuje.

3 algoritmas. *Lygiagretusis pilnojo perrinkimo algoritmas su MPI*

$f^* = \infty$

Input: *rank*, *size*

Include: *mpi.h*

for $\forall X \in \{(0,\dots,0,0),(0,\dots,0,1),\dots,(1,\dots,1,1)\}$ **do**

if ((*int*) X % *size* = *rank*) **and** $f(X) < f^*$ **and** $g(X) = 0$ **and** $h(X) \leq 0$ **then**

$f^* = f(X)$

$X^* = X$

end if

end for

if ($rank <> 0$) **then MPI_SEND** (f^*, X^* **to** 0)

else

for 1 **to** *size*-1 **do MPI_RECV** ($f^{*\prime}$, X' **from** ANY)

if $f^{*\prime} < f^*$ **then**

$f^* = f^{*\prime}$

$X^* = X'$

end if

end for

end if

3.3.2. Šakų ir rėžių algoritmas skirtas strypinių konstrukcijų optimizavimui

Pirmą kartą šakų ir rėžių metodas buvo pasiūlytas 1960 metais (Land ir Doig 1960). Metodas buvo skirtas sveikaskaitinio tiesinio programavimo uždavinio sprendimui. Vėliau metodą pradėjo naudoti savo darbuose Little, Murty, Sweeney ir Karel keliaujančio pirklio uždaviniui spręsti (Little *ir kt.* 1963). Nuo to laiko mokslinių darbų, susietų su šakų ir rėžių metodu, kiekis padidėjo. Tai įvyko dėl to, kad minėti autoriai pirmieji pastebėjo metodo taikymo platumą ir akcentavo jo pritaikomumą sprendžiamų uždavinių specifikai.

Vėliau šakų ir rėžių metodą siūlė Horst (Horst *ir kt.* 1995), Horst ir Tuy (Horst ir Tuy 1987; Horst ir Tuy 1996) savo darbuose. Jų šakų ir rėžių metodas

naudojamas realizuojant globaliojo optimizavimo padengimo ir kombinatorinio optimizavimo metodus. Šakų ir rėžių algoritmai gali būti naudojami kandidatų atrinkimui ir jų šalinimui bei dalijimui valdyti. Šakų ir rėžių algoritmus sudaro inicializavimo, išrinkimo, dalijimo bei rėžių skaičiavimo taisyklės. Pagrindiniai šakų ir rėžių algoritmo žingsniai pristatyti R. Čiegio knygoje (Čiegis 2007).

Šakų ir rėžių algoritmą sudaro:

1. Inicializavimo etapas.

   Leistinųjų sprendinių aibę skaidome į baigtinį poaibių skaičių:

$$D = \bigcup_{i=1}^{m} D_i. \tag{3.2}$$

   Funkcijos $f(X)$ minimumo paieškos algoritmą patogu vaizduoti paieškos medžiu. Medžio šaknys apibrėžią visų leistinųjų sprendinių aibę D. Šaknies vaikai apibrėžia poaibius D_i.

2. Iš kandidatų aibės išrenkamas ir padalijamas poaibis; apskaičiuojami funkcijos minimumo apatinis (LB) ir viršutinis (UB) rėžiai naujai gautuose poaibiuose. Algoritmas apibrėžia, kaip efektyviai apskaičiuoti funkcijos $f(X)$ minimumo srityje D_i apatinį (LB) ir viršutinį (UB) rėžius:

$$LB(D_i) \leq \min_{X \in D_i} f(X) \leq UB(D_i). \tag{3.3}$$

   Tada, žinodami $UB(D_i)$, galime apskaičiuoti $f(X)$ minimalios reikšmės viršutinį rėžį $UB(D)$ visoje aibėje D:

$$UB(D) = \min_{1 \leq i \leq m} UB(D_i). \tag{3.4}$$

3. Žinodami šiuos rėžius, dažnai galime gerokai sumažinti skaičiavimų kiekį. Jeigu srityje D_i apatinis $f(X)$ reikšmių rėžis $LB(D_i)$ yra didesnis už $UB(D)$:

$$LB(D_i) > UB(D), \tag{3.5}$$

tai tokios srities toliau tirti nereikia, nes joje globaliojo minimumo negali būti.

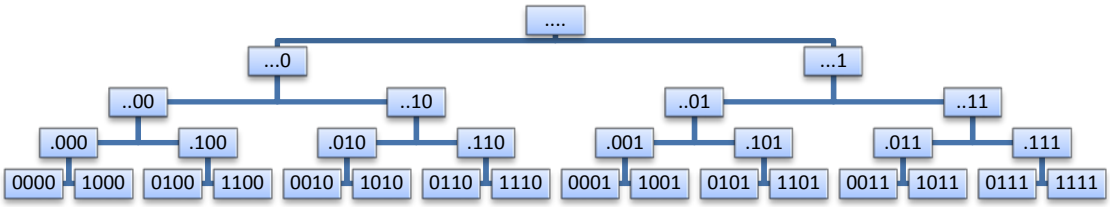
Algoritmas siekia, kad kandidatų aibė greitai mažėtų ir konverguotų į sprendinį.

Galimos išrinkimo taisyklės:

- pirmas geriausias – išrenkamas elementas su geriausiu įverčiu (su minimaliu apatiniu tikslo funkcijos rėžiu);
- pirmas gilyn – išrenkamas jauniausias kandidatas;
- pirmas platyn – išrenkamas vyriausias kandidatas.

Rėžių skaičiavimo taisyklės nusako, kaip yra įvertinami funkcijos minimumo rėžiai. Šakų ir rėžių algoritmų efektyvumas priklauso nuo rėžių tikslumo (Žilinskas ir Žilinskas 2006).

Taikomajam uždaviniui spręsti naudojama išrinkimo taisyklė – pirmas gilyn. Strypinių konstrukcijų optimizavimo binarinio medžio pavyzdys parodytas 3.4 pav.



3.4 pav. *Galimų konstrukcijų medis*

Medžio viršūnėje neturime konstrukcijų, pažymėtos tik vietos, kurios reiškia strypo vietą konstrukcijoje. Pradedant konstruoti binarinį medį, pridedame prie konstrukcijos strypus palaipsniui pradedant nuo pirmos vietos, tokiu būdu gauname dvi medžio šakas su 0 ir 1, t.y. šaką, kur pridėjome vieną strypą (1), ir kitą, kurioje to strypo nėra (0). Nagrinėjant šakas palaipsniui

pridedame naujus strypus arba paliekam vietoje jų 0. Tokiu būdu išvengiama pasikartojimo ir gaunamas visas įmanomas galimų konstrukcijų medis.

Šakų ir rėžių algoritmo vykdymo metu šakų konstravimą įtakoja geriausia žinoma tikslo funkcijos reikšmė. Medžio konstravimo metu konstrukcijos, esančios medžio šakoje, gali būti arba tokios pačios, arba turės papildomų strypų. Konstrukcijos, turinčios papildomus strypus bus sunkesnės, nes papildomas strypas konstrukcijoje padidina ir visos konstrukcijos masę. Dėl to radus naują minimalią reikšmę šakų galima toliau nenagrinėti. Tokios šakos yra nukerpamos.

4 algoritme realizuotas šakų ir rėžių algoritmas strypinėms konstrukcijoms optimizuoti. Konstrukcija koduojama kaip vektorius X, kuris priklauso nuo kintamojo p. Kiekvienas papildomas strypas konstrukcijoje gali tik padidinti konstrukcijos masę, dėl to galima išskirti algoritmo šaką su sprendiniu, atsižvelgiant į tikslo funkciją, po kurios nebus šakų geresnių už jau žinomą optimumą. Jeigu tas sprendinys yra didesnis už žinomą f^* (f^* – yra lokalus minimumas), tai tokios šakos tolimesni sprendiniai neturi būti nagrinėjami.

4 algoritmas. *Nuoseklusis šakų ir rėžių algoritmas strypinėms konstrukcijoms*

$f^* = \infty$; $x_i = 0, i = 1, 2, \dots, n$; $p = 1$;
while $p > 0$ **do**
    if $f(X) < f^*$ **and** $g(X) = 0$ **and** $h(X) \leq 0$ **then**
        $f^* = f(X)$
        $X^* = X$
    end if
    if $f(X) < f^*$ **then** $p = n$
    while $X_p = 1$ **and** $p \geq 1$ **do**
        $x_p = 0$
        $p = p - 1$
    end while
    if $p > 0$ **then** $x_p = 1$
end while

Šiuo metu Lietuvos paskirstytųjų skaičiavimų grido tinklas turi techninių trūkumų. Ne visuose klasteriuose veikia MPI biblioteka, be to neįmanomas MPI pranešimų siuntimas tarp klasterių. Kai užduotis siunčiama į klasterį, vienas iš kompiuterių tampa užduoties vykdymo šeimininku. Toks kompiuteris-šeimininkas mato kompiuterius-darbininkus klasterio viduje. Jis gali jungtis su išorėje esančiais kompiuteriais, turinčiais išorinį IP-adresą, bet negali patekti į kitų klasterių vidinius kompiuterius. O tai padaro neįmanomus didelius skaičiavimus, nes klasteriai turi ribotus skaičiavimo resursus. Gali neužtekti kompiuterių-darbininkų atlikti užduotį.

Naudojantis tipinėmis grido valdymo komandomis sukurtas užduoties valdymo scenarijus. Pagal jį, užduotys paskirstomos tarp reikiamo kompiuterių skaičiaus, nepriklausomai nuo jų kilmės (gali būti ne tik iš vieno klasterio, bet ir iš skirtingų). Vienintelė sąlyga – kompiuteriuose turi veikti *gcc C++* kalbos kompiliatorius. Pradinė užduočių sritis yra padalinama tarp kompiuterių-darbininkų automatiškai užduoties valdymo scenarijaus pagalba. Po užduoties įvykdymo, rezultatai perduodami per grido valdymo sąsają vartotojui. Rezultatai gaunami iš kiekvieno kompiuterio-darbininko, kiek tokių kompiuterių – tiek rezultatų failų.

Tokį scenarijų galima taikyti šakų ir rėžių bei kitiems algoritmams. Scenarijus pateikiamas kaip 3.3 scenarijus.

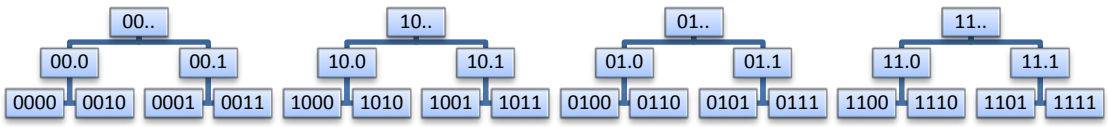
Sukurtas šakų ir rėžių algoritmas strypinėms konstrukcijoms optimizuoti atliekant paskirstytuosius skaičiavimus pateiktas kaip 5 algoritmas. Algoritmo vykdymo eigoje skaičiavimus atlieka keli kompiuteriai nepriklausomai vienas nuo kito. Priklausomai nuo kompiuterio unikalaus numerio yra keičiamas vektorius X (3.6 pav.). Pirmos $x_i,\ i = 1, \dots, k,$ kuoduoja kompiuterio numerį rank, tokių pozicijų skaičius lygus visų kompiuterių skaičiaus binarinės išraiškos ilgiui $k = \mathbf{log}(size)\ /\mathbf{log}\,(2)$. Po jų esančios $x_i,\ i = k+1, \dots, n,$

komponentės laisvai kaitaliojamos algoritmo vykdymo metu, taip 3.5 pav. pavaizduoti konstrukcijų medžiai, padalinti tarp 4 skaičiavimų procesų.

3.3 scenarijus. *Grid tinklo darbo scenarijus*

```
JobType = "Parametric";                    //Darbo tipas – parametrinis
Executable = "Paleidziamasis_Linux_Shell_Scenarijus.sh";          //paleidžiamasis failas
Parameters = 64;                           // ciklas iki 64, bet gali būti bet koks 2 kartotinis, 2^n;
ParameterStart = 0;                        //ciklas prasideda nuo 0;
ParameterStep = 1;                         //ciklo kintamasis didinamas 1;
Arguments = "_PARAM_";                     //Vietoje _PARAM_ vėliau automatiškai įrašomas ciklo kintamasis;
StdOutput = "_PARAM_.out";                 //standartinis pranešimų failas
StdError = "_PARAM_.err";                  //standartinis klaidų pranešimų failas
InputSandbox = {"Paleidziamasis_Linux_Shell_Scenarijus.sh", "C++_Kodo_failas.cpp", "Programos_Vykdymo_Duomenų_Failas.txt"};   // failai perduodami gridui
OutputSandbox = {"_PARAM_.out","_PARAM_.err", "Programos_Vykdymo_Rezultatų_Failas_PARAM_.txt"};   //failai gražinami po vykdymo
```

3.4 ir 3.5 paveiksluose galima pastebėti, kad medžio, kurį gauna procesas, dydis sumažėjo, nors visų galimų konstrukcijų skaičius išliko tas pats. Tokiu būdu kiekvienas kompiuteris atlieka tik jam skirtus skaičiavimus, o visa užduotis pasidalinama tarp visų ją sprendžiančių kompiuterių. O iš to išplaukia, kad kuo didesnis kompiuterių skaičius nagrinės galimas konstrukcijas, tuo sparčiau vyks skaičiavimai.



3.5 pav. *Galimų konstrukcijų medis, padalintas tarp 4 procesų*

Vektoriaus dalys atliekant paskirstytuosius skaičiavimus pavaizduotos 3.6 pav. Pirmosios vektoriaus X pozicijos koduoja procesoriaus, kuriam atiteko

konstrukcijų medžio šaka, numerį ir nesikeičia viso skaičiavimo proceso metu. Kitos vektoriaus X pozicijos gali keistis viso skaičiavimo proceso metu.

5 algoritmas. *Paskirstytųjų skaičiavimų šakų ir rėžių algoritmas strypinėms konstrukcijoms*

Input: *rank, size*

$k = \mathbf{log}(size)\ /\mathbf{log}\ (2)$

$(x_1, x_2, \dots, x_k) = \mathbf{binary\_code}(rank)$

$x_i = 0, i = k+1, \dots, n$

$f^* = \infty; p = k\ + 1;$

while $p > k$ **do**

  if $f(X) < f^*$ **and** $g(X) = 0$ **and** $h(X) \leq 0$ **then**

    $f^* = f(X)$

    $X^* = X$

  end if

  if $f(X) < f_{min}$ **then** $p = n$

  while $X_p = 1$ **and** $p \geq k + 1$ **do**

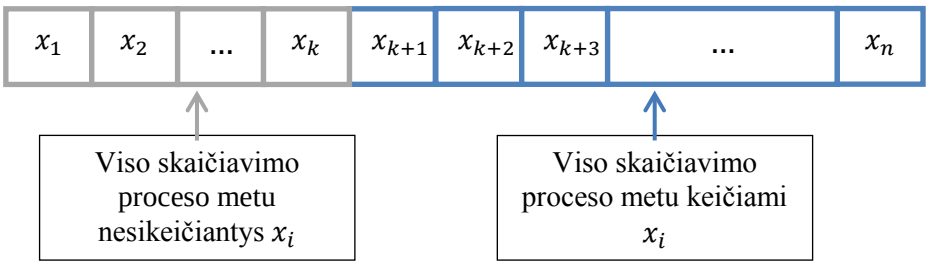
    $x_p = 0$

    $p = p - 1$

  end while

  if $p > k$ **then** $x_p = 1$

end while



3.6 pav. *Vektoriaus **X** sandara, atliekant skaičiavimus paskirstytuoju būdu*

3.4. Lygiagretusis šakų ir rėžių algoritmas daugiamatėms skalėms

Daugiamatės skalės (DS) naudojamos daugiamačių duomenų struktūros analizei atvaizduojant juos dvimatėje arba trimatėje erdvėje. DS metodas sprendžia, kaip n skirtingumu apibrėžti objektai gali būti patikimai atvaizduoti taškais mažo matavimo erdvėje. i-ojo ir j-ojo objektų skirtingumas yra apibrėžtas realiu skaičiumi $\delta_{i,j}$, $i,j = 1,\dots,n$. Ieškoma taškų m-matėje erdvėje $X_i \in \mathbb{R}^{\mathrm{m}}$, $i = 1,\dots,n$, tarp kurių atstumai atitiktų duotus skirtingumus. Atvaizdavimo kokybė yra matuojama įtempimo funkcija, kuri lygina objektų skirtingumą ir atstumą tarp juos atvaizduojančių taškų. Objektų atvaizdžiai gali būti randami minimizuojant šią funkciją: turi būti rastos tokios n taškų koordinatės m-matėje erdvėje, kad įtempimo funkcija būtų minimali. Dažniausiai naudojama mažiausių kvadratų įtempimo funkcija. Plačiau šis metodas aprašytas J. Žilinsko darbuose (Žilinskas ir Žilinskas 2009; Žilinskas J. 2009; Žilinskas J. 2012).

3.5. Trečiojo skyriaus apibendrinimas ir išvados

Šiame skyriuje pasiūlyta klasterių ir gridų elektros energijos suvartojimo mažinimo strategija. Nauja strategija gali būti įdiegta į bet kokį klasterį arba gridą, kur veikia Linux OS. Pasiūlyti scenarijai lengvai realizuojami ir gali būti įdiegti lygiagrečiuose kompiuteriuose. Be to, skyriuje pasiūlyta klasterio elektros energijos suvartojimo tyrimo metodika.

Taip pat pasiūlytas lygiagrečiųjų skaičiavimų darbų paleidimo scenarijus skirtas gridui. Scenarijaus privalumas – tai, kad skaičiavimams atlikti klasteriuose reikalingas tik gcc C++ kalbos kompiliatorius be papildomų bibliotekų. Nereikia, kad grido klasteriuose (telkiniuose) būtų įdegtos ir suderintos MPI bibliotekos. Be to, dideliu privalumu yra tai, kad užduotims spręsti galima išnaudoti didelius grido resursus, apeinant MPI ir lokalių klasterių dydžio apribojimus.

Skyriuje siūloma nuoseklioji bei lygiagrečioji šakų ir rėžių algoritmo versijos strypiniems konstrukcijoms optimizuoti. Skyriuje paminėtas ir lygiagretusis šakų ir rėžių algoritmas daugiamatėms skalėms optimizuoti. Siūlomas bei paminėtas algoritmai naudojami klasterių ir grido elektros energijos suvartojimui tirti (4 skyrius).

4

Eksperimentiniai tyrimai

Šiame skyriuje pateikiami eksperimentų rezultatai skirti parodyti siūlomo elektros energijos taupymo būdo privalumą. Buvo atlikti eksperimentiniai tyrimai naudojant gridus bei kompiuterių klasterius. Elektros energijos suvartojimui tirti naudojama 3 skyriuje aprašyta tyrimo metodika. Taipogi skyriuje aprašomi ir kiti eksperimentų metu gauti rezultatai: kompiuterinio tinklo pralaidumo įtaka lygiagrečiųjų skaičiavimų rezultatams, ištirtas strypinių konstrukcijų optimizavimas griduose ir kompiuterių klasteriuose.

Skyriuje aprašomas Lietuvos paskirstytųjų skaičiavimų tinklas LitGrid ir problemos su kuriomis gali susidurti šio tinklo naudotojai. Rezultatai, gauti eksperimentų metu, publikuoti autoriaus darbuose (A 1, A 2, A 3, A 4, B 1).

4.1. Programinė ir techninė įranga

Eksperimentams atlikti naudojama techninė ir programinė įranga:

- Kompiuteriai su 4-ių branduolių procesoriais;
- Kompiuteriai su 2-jų branduolių procesoriais;

- Kompiuteriai su procesoriais, turinčiais du virtualiuosius branduolius.

LitGrid projekto resursai:

- MII-LCG2 (grid.mii.lt);
- VU-MIF-LCG2 (grid.mif.vu.lt);
- Kiti LitGrid (Bagdonavičius ir Lapienis 2005) projekto klasteriai.

Klasteriai:

- Lietuvos edukologijos universiteto klasteris;
- Vilniaus universiteto Matematikos ir informatikos instituto klasteriai cluster.mii.lt ir hpc.mii.vu.lt.

Operacinės sistemos:

- Windows 7 Professional x86;
- Scientific Linux 4.7 x86;
- Rocks Cluster 5.2 x86.

Algoritmų ir skriptų programinė realizacija:

- MS Visual Studio 2005 C++;
- Gcc C++;
- OpenMP ir MPI bibliotekos;
- Linux Bash;
- LCG gLite.

4.2. LitGrid klasterių apžvalga

Atliekant tyrimus grid tinkle buvo pastebėta, kad įvairūs klasteriai tą patį uždavinį išsprendžia per skirtingą laiką – net tame pačiame klasteryje sprendimo laikai kartais ganėtinai skirtingi. Grid tinkle kiekviename iš

klasterių 10 kartų buvo vykdyta strypinių konstrukcijų optimizavimo šakų ir rėžių algoritmo nuoseklioji versija skirtingo sunkumo uždaviniams (7 ir 8 mazgų strypinės konstrukcijos).

Minimalūs ir maksimalūs vykdymo laikai pateikti 4.1 lentelėje. Šioje lentelėje taip pat pateiktas ir kiekvieno klasterio kompiuterių skaičius.

4.1 lentelė. *LitGrid klasterių skaičiavimo greičių tyrimo rezultatai*

LitGrid klasteris	Prieinamas klasteryje procesų skaičius	MPI	GCC	Strypinių konstrukcijų optimizavimo uždavinio sprendimo laikas, sek			
				7 mazgų užduotis		8 mazgų užduotis	
				Min, sek.	Max, sek.	Min, sek.	Max, sek
ce.grid.lei.lt (LEI)	10	-	yra	4,59	4,73	628,47	632,42
dane.ku.lt (KU)	18	-	Yra	5,68	5,78	765,82	785,07
grid.akolegija.lt (AKOLEGIJA)	5	yra	Yra	5,78	5,81	770,80	772,26
grid.marko.lt (MARKO)	5	yra	Yra	4,50	5,18	600,16	603,32
grid.mii.lt (VU MII)	68	yra	Yra	4,75	4,76	632,19	633,34
grid.pri.kmu.lt (KMU)	12	-	Yra	9,20	12,48	1 258,78	1 709,95
grid9.mif.vu.lt (VU MIF)	64	yra	Yra	8,68	8,75	1 265,43	1 266,90
pupa.elen.ktu.lt (KTU)	12	yra	Yra	9,59	11,95	1 615,86	1 658,50
vdu.pdc.vdu.lt (VDU)	10	-	yra	9,35	13,32	1 274,35	2 660,00

Kaip parodė tyrimo rezultatai, visame LitGrid tinkle skiriasi skaičiavimų greitis dėl didelio kompiuterinių resursų nehomogeniškumo. Kai kuriuose klasteriuose galima pastebėti vidinį klasterio nehomogeniškumą, nes minimalūs ir maksimalūs tos pačios užduoties vykdymo laikai žymiai skiriasi.

Labiausiai skiriasi laikai VDU klasteryje, taigi jis labiausiai heterogeninis. Mažiausiai skiriasi minimalūs ir maksimalūs laikai: VU MII, KTU, VU MIF, MARKO, AKOLEGIJA. Šių klasterių kompiuteriai homogeniškiausi. Tarp

visų klasterių mažiausi sprendimo laikai yra VU MII, MARKO, LEI klasteriuose, šių klasterių kompiuteriai yra greičiausi. Tik MARKO ir LEI klasteriai yra labai maži, tik 5 ir 10 skaičiavimo mazgų. Patys stambiausi klasteriai yra VU MII ir VU MIF, jie turi daugiausiai skaičiavimo mazgų. Šie klasteriai yra sudaryti iš vienodų ar panašių kompiuterių.

Atlikus klasterių apžvalginį tyrimą galima teigti, kad LitGrid tinkle sudėtingiems ir tiksliems lygiagretiesiems skaičiavimams galima naudoti tik kelis klasterius, nes tik keli klasteriai, VU MII ir VU MIF, turi pakankamai didelį skaičiavimo mazgų skaičių ir tik keli iš jų yra homogeniški.

4.2.1. LitGrid MII-LCG2 klasteris

Disertacijos rašymo metu buvo administruojamas klasteris, priklausantis LitGrid paskirstytųjų skaičiavimų tinklui – MII-LCG2 (VU MII – grid.mii.lt). Klasterio jungimo schema pavaizduota 4.1 pav. Kaip matosi iš schemos, visi klasterio kompiuteriai sujungti 1 Gbps tinklu.

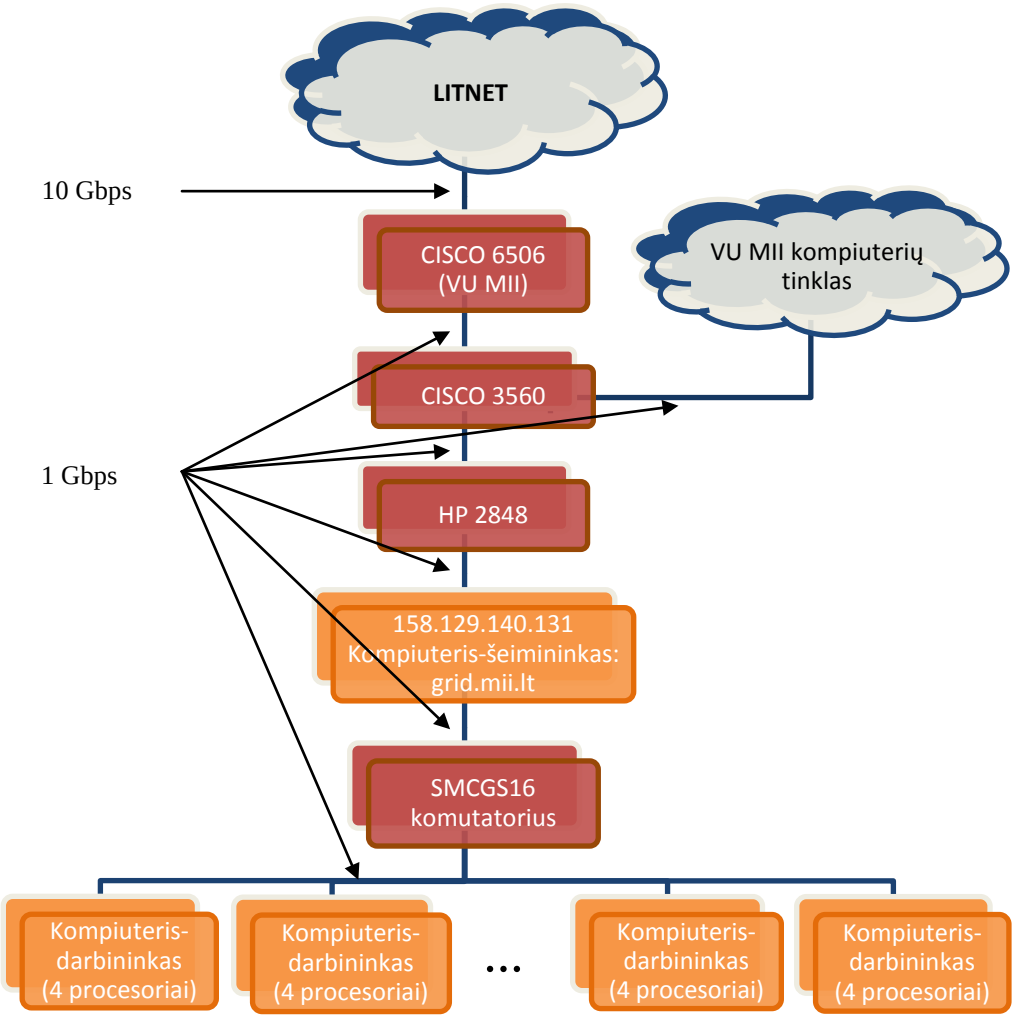
Jungimuisi prie klasterio ir užduočių paleidimui per gLite grido aplinką buvo sukurtas virtualiosios mašinos elektroninis atvaizdis (nemokamos *VmWare player* programinės įrangos pagrindu). Visi LitGrid naudotojai galėjo jį laisvai atsisiųsti ir įdiegti savo kompiuteriuose.

Klasteryje be gLite programinės įrangos yra įdiegtos ir kitos tarnybos: httpd – web prieigos servisas, Ganglia – klasterio darbo stebėjimo servisas, MySQL – žiniatinklio duomenų saugojimo duomenų bazės servisas, ssh – nuotolinės prieigos prie kompiuterio servisas ir kiti.

4.3. Kompiuterių tinklo įtaka lygiagrečiųjų skaičiavimų rezultatams

Apžvelgiant lygiagrečiuosius skaičiavimus buvo nuspręsta išanalizuoti esamų kompiuterinių tinklų efektyvumą. Eksperimentams atlikti buvo naudoti Lietuvos edukologijos universiteto resursai.

Eksperimentams atlikti, buvo panaudoti tipiniai lygiagrečiųjų skaičiavimų algoritmai (matricų sandauga, tiesinių lygčių sistemos (TLS) sprendimas Gauso metodu).



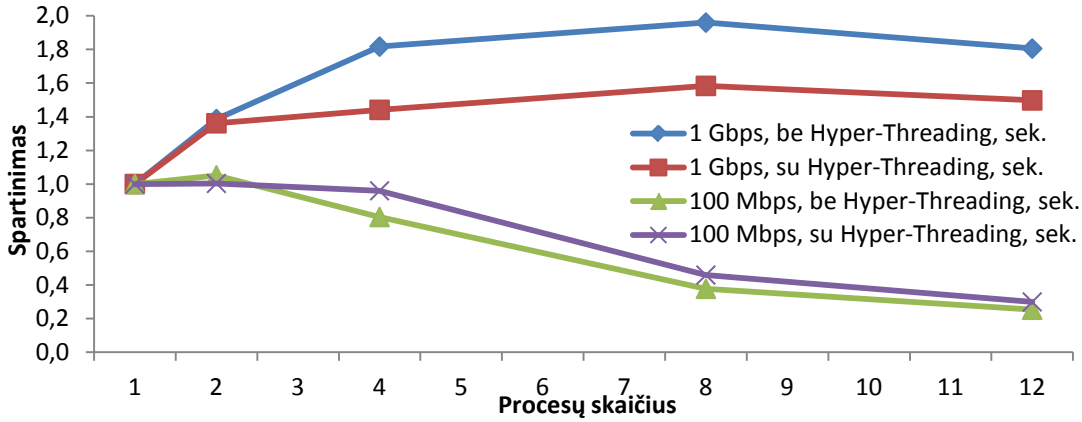
4.1 pav. *LitGrid VU MII klasterio prijungimo schema*

Eksperimentams atlikti naudojami: TLS iš 1 000 lygčių, matricos iš 100 x 100, 500 x 500, 1 000 x 1 000 elementų ir vektoriai iš 1 000 elementų. Double tipo skaičių generavimui naudojamos funkcijos – srand (unsigned (clock ())) ir rand () / double (1000). Eksperimentai atlikti su 100 Mbps ir su 1 Gbps kompiuterių tinklais. Jų metu kompiuteriuose buvo paleidžiamas vienas

arba du procesai. Eksperimentai buvo kartojami 100 kartų ir jų rezultatai suvesti prie bendro vidurkio.

4.3.1. Lygiagretieji TLS Gauso metodu skaičiavimai

Eksperimentų pradžioje sugeneruojama tiesinių lygčių sistemą. Po TLS elementų generavimo vykdomi elementarieji TLS pertvarkymai Gauso metodu. Rezultatai pateikti 4.2 paveikslėlyje.



4.2 pav. *TLS sprendimų spartą, priklausomai nuo tinklo pralaidumo ir Hyper-Threading technologijos naudojimo*

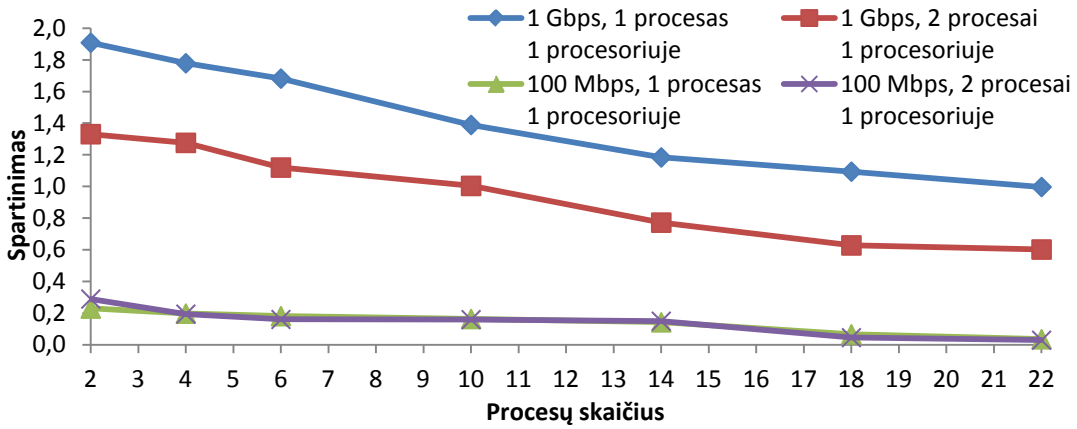
4.2 lentelė. *TLS sprendimo laikai skirtingo pralaidumo tinkluose*

Procesų skaičius	1	2	4	8	12
1 Gbps, be Hyper-Threading, sek.	4,017	2,898	2,211	2,050	2,225
100 Mbps, be Hyper-Threading, sek.	4,017	4,003	4,185	8,752	13,422
1 Gbps, su Hyper-Threading, sek.	4,017	2,951	2,788	2,537	2,483
100 Mbps, su Hyper-Threading, sek.	4,017	3,823	4,995	10,647	15,872

Rezultatai gauti atliekant skaičiavimus 1 Gbps tinkle kur kas geresni negu 100 Mb tinkle. Tai įvyksta dėl labai intensyvaus duomenų apsikeitimo tarp procesų. Taip pat matosi akivaizdus Hyper-Threading technologijos nenaudingumas šio uždavinio atžvilgiu 1 Gbps tinkle. Rezultatai pasiekti su 100 Mbps tinklu parodė, kad Hyper-Threading technologija pranašesnė su 100 Mbps tinklu. Tai labai gerai matosi ir iš 4.2 lentelės duomenų.

4.3.2. Matricos ir vektoriaus sandauga

Eksperimentų pradžioje sugeneruojami matrica ir vektorius. Rezultatai pateikti 4.3 paveikslėlyje.



4.3 pav. *Matricos ir vektoriaus sandaugos spartinimas priklausomai nuo procesų skaičiaus*

4.3 paveikslėlyje matosi kad 100 Mbps tinklo naudojimas sulėtina algoritmo vykdymą. 1 Gbps tinklo nauda matosi kai naudojami 2 kompiuteriai. Po to, didėjant kompiuterių skaičiui, spartinimas mažėja. Tai įvyksta dėl per didelio žinučių apsikeitimo tarp kompiuterių-darbininkų. Hyper-Threading technologija su šiuo algoritmu gerų rezultatų neparodo. Tyrimų metu atliktų skaičiavimų laikai pateikti lentelėje 4.3.

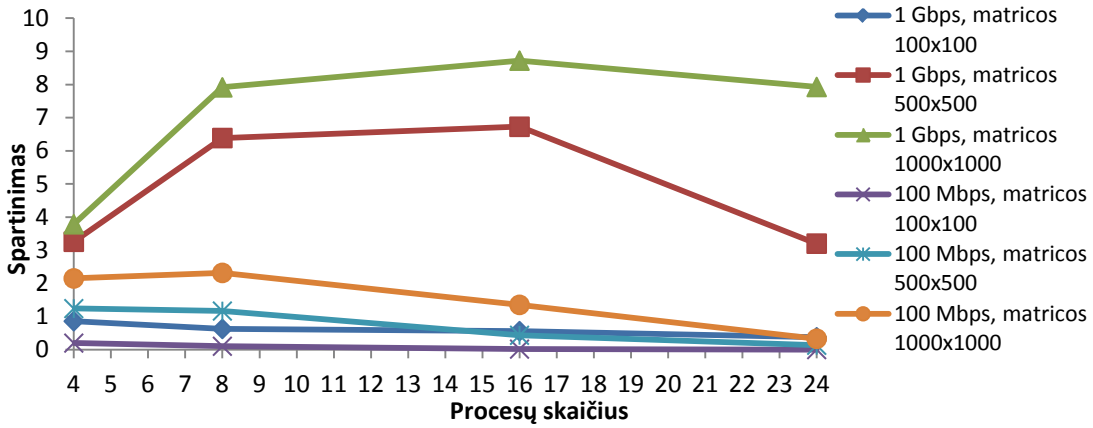
4.3 lentelė. *Matricos su vektoriumi sandaugos laikai skirtingo pralaidumo tinkluose*

Procesų skaičius	1	2	4	8	12	16	20
1 Gbps, be Hyper-Threading, sek.	0,160	0,084	0,090	0,107	0,129	0,140	0,152
100 Mbps, be Hyper-Threading, sek.	0,160	0,535	0,811	0,933	1,054	1,783	3,519
1 Gbps, su Hyper-Threading, sek.	0,160	1,103	0,120	0,125	0,143	0,150	0,159
100 Mbps, su Hyper-Threading, sek.	0,160	0,427	0,553	0,823	1,002	1,006	1,009

4.3.3. Matricų sandauga

Eksperimentai atlikti su įvairių dydžių kvadratinėmis matricomis. Po matricų generavimo vykdoma jų daugyba. Rezultatai pateikti 4.4 paveikslėlyje.

Rezultatai parodo 1 Gbps tinklo pranašumą. Matricų daugybos rezultatai (dydis yra 100 x 100), 1 Gbps tinkle prastesni dėl mažo užduoties dydžio. Su didesnėmis matricomis rezultatai kur kas geresni, spartinimas išauga iki 8,5. Tą galima pastebėti ir lentelėje 4.4, kur pateikiami įvykdytų skaičiavimų suvesti prie bendro vidurkio vykdymų laikai.



4.4 pav. *Įvairaus dydžio matricų sandaugos skirtingo pralaidumo tinkluose spartinimas priklausomai nuo procesų skaičiaus*

4.4 lentelė. *Įvairių didžių matricų sandaugos laikai skirtingo pralaidumo tinkluose*

Procesų skaičius	1	4	8	16	24
1 Gbps, matricos 100x100, sek.	0,014	0,016	0,022	0,021	0,037
1 Gbps, matricos 500x500, sek.	2,423	0,740	0,377	0,358	0,753
1 Gbps, matricos 1000x1000, sek.	22,223	5,546	2,653	2,407	2,649
100 Mbps, matricos 100x100, sek.	0,014	0,069	0,133	0,707	3,105
100 Mbps, matricos 500x500, sek.	2,423	1,949	2,072	5,540	17,859
100 Mbps, matricos 1000x1000, sek.	22,223	10,329	9,584	16,401	66,835

4.4. Strypinių konstrukcijų optimizavimas

Strypinių konstrukcijų optimizavimo uždavinio nagrinėjimo metu buvo sukurtos pilnojo perrinkimo bei šakų ir rėžių algoritmų realizacijos. Pirmieji bandymai parodė, kad šakų ir rėžių algoritmas yra spartesnis už pilnojo perrinkimo algoritmą. Tai labai aiškiai matosi iš 4.2 lentelės. Rezultatai gauti vykdant nuosekliąsias pilnojo perrinkimo bei šakų ir rėžių algoritmo versijas,

eksperimentai buvo atliekami 10 kartų ir jų rezultatai vidurkinami.

4.5 lentelė. *Nuosekliųjų algoritmų vykdymo laikai optimizuojant strypines konstrukcijas*

Mazgų skaičius konstrukcijoje	6 mazgai	7 mazgai	8 mazgai
Pilnojo perrinkimo algoritmo vykdymo laikai, sek.	0,14	14,39	2 824,88
Šakų ir rėžių algoritmo vykdymo laikai, sek.	0,11	6,05	792,19

Toliau buvo nuspręsta išbandyti algoritmų veikimą paskirstytųjų skaičiavimų tinkle, 4.3 lentelė. Skaičiavimams atlikti buvo pasirinktas grid9.mif.vu.lt klasteris, kadangi tuo metu tai buvo pats didžiausias klasteris LitGrid tinkle. Rezultatai parodė, kad vykdant skaičiavimus tinkle gaunami panašūs vykdymo laikų santykiai. Tik išaiškėjo, kad pradedant vykdyti užduotį su keliais kompiuteriais, šakų ir rėžių algoritmas sulėtėja. Tai įvyksta dėl to, kad didėjant procesų skaičiui, didėja bendras tikrinimų skaičius dėl informacijos neapsikeitimo tarp procesų.

Po klasterio grid.mii.lt išplėtimo ir ištyrus šakų ir rėžių algoritmo veikimą LitGrid tinkle buvo nuspręsta tolimesniems tyrimams naudoti klasterį grid.mii.lt. Taip buvo nuspręsta dėl klasterio resursų homogeniškumo ir didelės greitaveikos (4.2 poskyris). Rezultatai parodė, kad algoritmo veikimas paspartėjo, 4.4 lentelė.

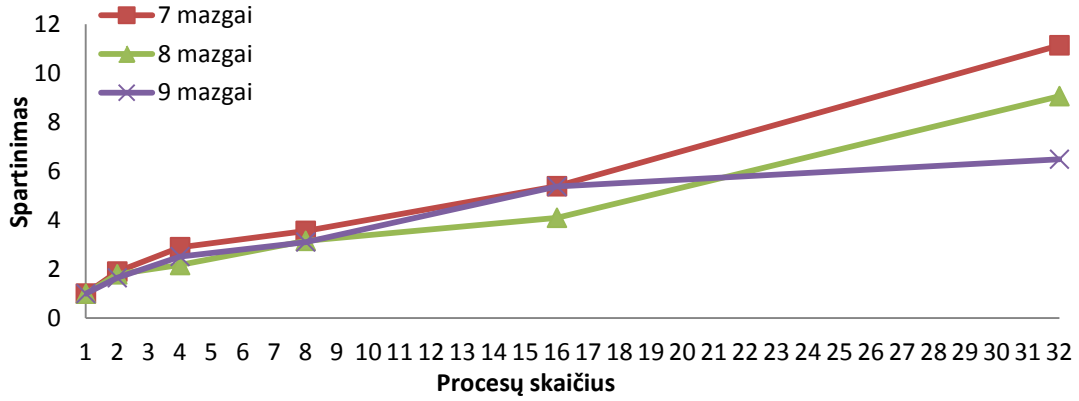
4.6 lentelė. *Pilnojo perrinkimo ir šakų ir rėžių algoritmų vykdymas gride (grid.mif.vu.lt)*

Kompiuterių sprendžiančių uždavinį skaičius, vnt.	1 procesas	2 procesai	4 procesai	8 procesai	16 procesų
Pilnojo perrinkimo algoritmo vykdymo laikai, 6 mazgai, sek.	0,18	0,10	0,06	0,03	0,02
Šakų ir rėžių algoritmo vykdymo laikai, 6 mazgai, sek.	0,13	0,07	0,04	0,03	0,02
Pilnojo perrinkimo algoritmo vykdymo laikai, 7 mazgai, sek.	19,78	11,19	6,06	3,22	1,71
Šakų ir rėžių algoritmo vykdymo laikai, 7 mazgai, sek.	8,66	4,53	2,88	2,29	1,30
Pilnojo perrinkimo algoritmo vykdymo laikai, 8 mazgai, sek.	4 088,82	2 217,81	1 167,12	606,54	316,39
Šakų ir rėžių algoritmo vykdymo laikai, 8 mazgai, sek.	1 266,55	703,60	472,73	335,42	220,07

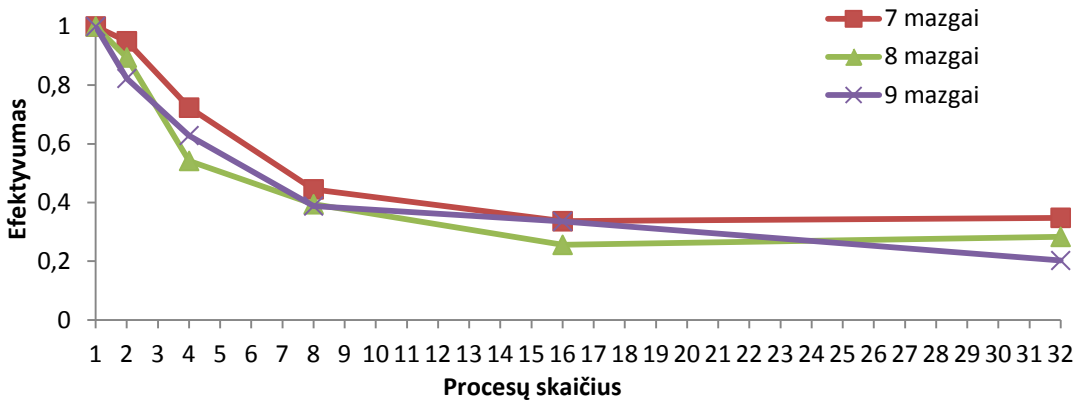
4.7 lentelė. *Šakų ir rėžių algoritmo vykdymo laikai gride (grid.mii.lt)*

Procesų skaičius	1 procesas	2 procesai	4 procesai	8 procesai	16 procesų
7 mazgai, sek.	4,75	2,50	1,64	1,33	0,88
8 mazgai, sek.	633,34	354,10	292,20	200,77	154,65
9 mazgai, sek.	142 067,00	98 458,00	64 466,00	52 186,00	49 376,00

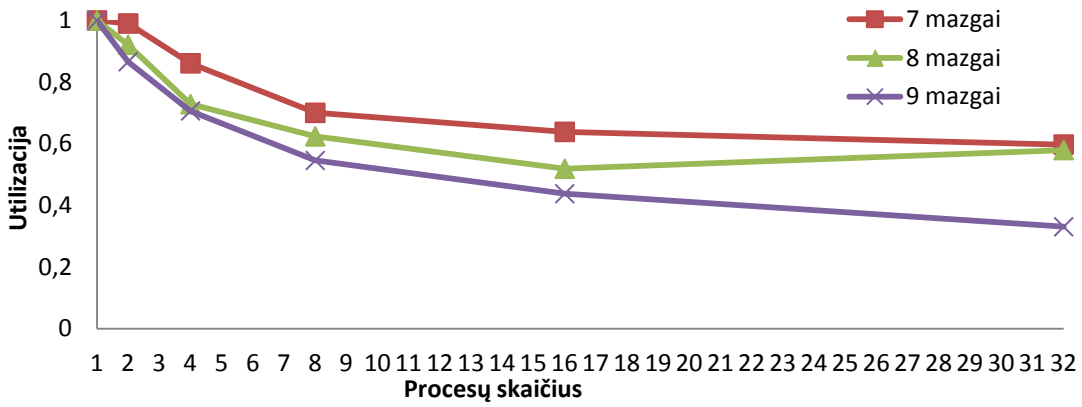
Atlikus matavimus, buvo apskaičiuoti ir tipiniai lygiagrečiųjų algoritmų vertinimo rodikliai: spartinimas, efektyvumas ir utilizacija (4.5–4.7 paveikslėliai.). Spartinimas auga didinant procesų-darbininkų skaičių. Užduočiai sunkėjant, padidėjo konstrukcijos dydis, spartinimo augimo tendencija mažesnė lyginant su lengvesnių uždavinių spartinimu. Bet visai kiti rezultatai matomi iš efektyvumo. Efektyvumas krenta didinant procesų-darbininkų skaičių (4.6 paveikslėlis). Tai vyksta dėl to, kad algoritmų realizacijai siūlomas paskirstytųjų skaičiavimų metodas.



4.5 pav. *Strypinių konstrukcijų optimizavimo ŠR algoritmo vykdymo spartinimas*



4.6 pav. *Strypinių konstrukcijų optimizavimo ŠR lgoritmo vykdymo efektyvumas*



4.7 pav. *Strypinių konstrukcijų optimizavimo ŠR algoritmo vykdymo utilizacija*

Paskirstytųjų skaičiavimų metodo minusas – procesai-darbininkai nesikeičia informacija apie geriausius sprendimus. Utilizacija parodo tokio duomenų ir užduočių paskirstymo trūkumą (4.7 paveikslėlis). Bendras kompiuterių išnaudojimo rodiklis krenta didinant procesų-darbininkų skaičių, o tai reiškia, kad atsiranda procesų, kurie labai greitai išsprendžia savo užduoties dalį ir po to nieko neveikia. Tai yra didžiausias paskirstytųjų skaičiavimų sprendimo būdo trūkumas.

4.5. Elektros energijos suvartojimo tyrimas

Disertacijoje tiriamas klasterio energijos suvartojimas atliekant įvairius lygiagrečiuosius skaičiavimus. Vienas iš algoritmų – lygiagretusis šakų ir rėžių algoritmas daugiamatėms skaliems su miesto kvartalo atstumais (Žilinskas J. 2012; Žilinskas ir Žilinskas 2009), kitas – lygiagretusis šakų ir rėžių strypinių konstrukcijų optimizavimo algoritmas. Abu optimizavimo uždaviniai buvo taikomi klasterio energijos suvartojimo tyrimuose. Be to, buvo ištirta vieno kompiuterio elektros galia, priklausomai nuo jo apkrovos (4.8 lentelė).

Matavimai atlikti naudojant elektros energijos suvartojimo matavimo prietaisą UPM MP300 Energy Meter (4.8 paveikslėlis). Matuoklis turi kelis matavimo režimus, tyrimų metu iš jų buvo naudojamas tik elektros galią (W) ir elektros energijos suvartojimą (kWh) matuojantys režimai.

Papildomai į klasterio mazgą-šeimininką monitoringui buvo įdiegtas paketas „gst-launch-0.10“ (GStreamer Community 2010), tam, kad būtų galima interaktyviai stebėti matuoklio parodymus per internetą (3.1 poskyrius, 3.1 paveikslėlis). Monitoringui buvo sukurti dar du darbo scenarijai 4.1 ir 4.2. Scenarijus 4.1 yra įdiegtas CRON tarnyboje mazge-šeimininke ir reguliariai paleidžiamas. Scenarijus 4.2 gali būti leidžiamas bet kuriame kompiuteryje su Linux OS su įdiegtu moduliu „gst-launch-0.10“. Taip pat matavimams stebėti naudojama GANGLIA klasterio darbų stebėjimo sistema (3.1 poskyrius, 3.2 paveikslėlis).

4.8 pav. *Elektros energijos matuoklis*

4.1 scenarijus. *Serverio WEB transliacijos įjungimo scenarijus*

```
#!/bin/sh
if [ -z "$( ps ax | grep -v grep | grep 'gst-launch')" ]        // funkcija IF tikrina ar WEB transliacija yra įjungta ir jei atsakymas neigiamas – įjungia
then
        gst-launch v4l2src ! video/x-raw-yuv, width=640, height=480,
        framerate=\(fraction\)5/1 ! ffmpegcolorspace ! jpegenc ! multipartmux ! tcpserversink host=localhost port=XXXXX;      // port – serveryje atidaryta vaizdo transliavimui prieiga
fi
```

4.2 scenarijus. *Linux‘ OS prieigos prie WEB transliacijos scenarijus*

```
#!/bin/sh
gst-launch tcpclientsrc host=SERVERIO_ADRESAS port=XXXXX ! multipartdemux ! jpegdec ! autovideosink // Serverio_adresas – serverio prie kurio yra jungiamasi realus IP adresas arba jo raidinis atitikmuo, port – turi atitikti prievado numerį kaip iš scenarijaus 4.1
```

Kaip parodė tyrimas net ir nevykdant skaičiavimų (tuščia eiga), t. y. kai kompiuteris laukia užduočių, kompiuterio elektros galia yra nemaža (vidutiniškai 50 W, 4.8 lentelė). Kai kompiuterio procesorius gauna užduotis, elektros galia palaipsniui auga, priklausomai nuo procesoriaus apkrovimo, bet nežymiai lyginant su tuščia eiga. Be to, elektros suvartojimui daro įtaką ir operacinė sistema, veikianti kompiuteriuose-darbininkuose. Grid.mii.lt ir hpc.mii.vu.lt skaičiavimams naudojo tuos pačius procesorius – Intel Quad Core Q9400, bet jų elektros suvartojimas skiriasi. Grid.mii.vu.lt klasteryje buvo įdiegta Scientific Linux 4.7 operacinė sistema, kuri įdiegiama kartu su grafine aplinka. Klasteryje hpc.mii.vu.lt įdiegta Rocks cluster 5.4 operacinė sistema, kuri įdiegia į kompiuterius-darbininkus OS be grafinės aplinkos. Tai sumažino elektros suvartojimą klasteryje. 4.8 lentelėje pateikti kompiuterių galios matavimai. Apkrovimas rodo kompiuterio darbo režimą ir kiek iš jo branduolių yra apkrauti, 25 % – 1 branduolys, 50 % – 2 branduoliai (grid.mii.lt, hpc.mii.vu.lt, hpc.mii.vu.lt) arba vienas (cluster.mii.lt), 75 % – 3 branduoliai ir 100 % – 4 branduoliai (grid.mii.lt, hpc.mii.vu.lt, hpc.mii.vu.lt) arba du (cluster.mii.lt). Be to, parodyta tuščios eigos galia ir galia su apkrova. Stulpelis – be tuščios eigos, rodo elektros galią atmetus tuščiąją eigą. Vienam branduoliui – elektros galia atitenkanti vidutiniškai kiekvienam iš veikiančių branduolių be tuščios eigos, papildomam branduoliui – kiek padidėjo galia, apkrovus kompiuterį papildomu darbu.

Taip pat išaiškėjo, kad 2 branduolių AMD Athlon 64 x2 Dual Core 5000+ procesoriai suvartoja daugiau elektros energijos ir atliekant skaičiavimus, ir

tuščios eigos režime. Be to, jie turi didžiausią elektros suvartojimo lygį, atitenkantį vienam branduoliui, kai procesorius pilnai apkrautas. Intel Quad Core Q9400 su Rocks cluster 5.4 operacine sistema suvartoja mažiau elektros energijos lyginant su Intel I5-760 procesoriais, taip pat ir Intel Quad Core Q9400 su Scientific Linux 4.7 operacine sistema.

4.8 lentelė. *Daugiabranduolinio kompiuterio elektros galingumas priklausomai nuo apkrovos*

Apkrovimas	Kompiuteriui	Galia, W		
		Be tuščiuos eigos	Vienam branduoliui	Papildomam branduoliui
grid.mii.lt		Intel Quad Core Q9400		
Tuščia eiga (Idle)	50 ± 5			
25 %	80 ± 5	30	30	30
50 %	93 ± 5	43	22	13
75 %	110 ± 5	60	20	17
100 %	120 ± 5	70	18	10
cluster.mii.lt		AMD Athlon 64 x2 Dual Core 5000+		
Tuščia eiga (Idle)	59 ± 5			
50 %	91 ± 5	32	32	32
100 %	115 ± 5	56	28	24
hpc.mii.vu.lt		Intel Quad Core Q9400		
Tuščia eiga (Idle)	46 ± 5			
25 %	67 ± 5	21	21	21
50 %	81 ± 5	34	17	13
75 %	89 ± 5	43	14	9
100 %	97 ± 5	51	13	7
hpc.mii.vu.lt		Intel I5-760		
Tuščia eiga (Idle)	50 ± 5			
25 %	78 ± 5	28	28	28
50 %	90 ± 5	40	20	12
75 %	105 ± 5	55	18	15
100 %	122 ± 5	72	18	7

Atsižvelgiant į rezultatus buvo padaryta išvada, kad norint taupiai vartoti elektros energiją (kad būtų mažesnis elektros energijos suvartojimas) kompiuterių procesorius reikia kiek galima pilnai apkrauti, o ne išskirstyti užduotis keliems kompiuteriams (jų pilnai neapkraunant). Kaip paaiškėjo skaičiavimų vykdymo metu, grido infrastruktūra to nedaro (lentele 4.9), užduotimis apkrauna klasterio kompiuterius nepilnai, dėl to klasteris suvartoja

kur kas daugiau elektros energijos. Kaip matosi iš 4.9 lentelės duomenų, klasterio suvartojimas tai pačiai užduočiai skiriasi ir gali išaugti iki 2,5 karto lyginant su rankiniu užduočių skirstymu. Tai atsitinka dėl reto duomenų apsikeitimo tarp klasterio kompiuterių-darbininkų, kompiuterio-šeimininko ir kompiuterio, kuris skirsto visas užduotis grid tinkle. Eksperimentui atlikti užduotis buvo leidžiami 10 kartu ir fiksuojamas minimalus ir maksimalus elektros energijos suvartojimas. Suvartojimas apskaičiuotas sudauginus atitinkamų kompiuterių užduoties vykdymo laikus su tu metu veikiančia apkrova. Tokiu būdu gaunant pilnąjį elektros energijos suvartojimą, neatmetant Idle.

4.9 lentelė. *Grido kompiuterių elektros energijos suvartojimas priklausomai nuo užduočių skirstymo ir neatmetant tuščiosios eigos elektros energijos suvartojimą*

Procesų skaičius	Elektros energijos suvartojimas, kWh					
	8 mazgų užduotys			9 mazgų užduotys		
	Rankinis užduočių skirstymas	gLite užduočių skirstymas	Suvartomas padidėjimas su gLite	Rankinis užduočių skirstymas	gLite užduočių skirstymas	Suvartomas padidėjimas su gLite
1	0,0141	0,0141	1,00	3,157	3,157	1,00
2	0,0090	0,0090-0,0145	1,00-1,61	2,499	2,499-3,689	1,00-1,48
4	0,0088	0,0088-0,0173	1,00-1,97	1,797	1,798-4,053	1,00-2,26
8	0,0080	0,0100-0,0120	1,25-1,50	2,580	2,902-4,195	1,12-1,62
16	0,0106	0,0120-0,0160	1,13-1,51	3,037	3,147-3,405	1,04-1,12

Rocks cluster 5.4 operacinė sistema paskirsto užduotis maksimaliai apkraunant klasterio kompiuterių procesorius. 4.10 lentelėje parodyti apskaičiuoti ir išmatuoti klasterio elektros suvartojimo lygiai. Kaip matosi iš lentelės duomenų energijos suvartojimas priklauso ir nuo algoritmo veikimo. Šakų ir rėžių algoritmui daugiamatiems skalėms su miesto kvartalo atstumais elektros energijos suvartojimas apskaičiuotas ir išmatuotas labai panašūs, nes labai gerai apkrauna klasterių kompiuterius. Strypinių konstrukcijų optimizavimo uždavinio apskaičiuotas energijos suvartojimas skiriasi nuo realaus dėl to, kad ne visi procesai skaičiuoja vienu metu, dalis jų gali baigti darbą anksčiau už kitus.

4.10 lentelė. *Klasterio hpc.mii.vu.lt energijos suvartojimas vykdant lygiagrečiuosius šakų ir rėžių algoritmus*

Procesų skaičius	Energijos suvartojimas, Wh Matavimai	Apskaičiuota	Laikas, sek.	Spartinimas	Efektyvumas
Šakų ir rėžių algoritmas daugiamatėms skalėms su miesto kvartalo atstumais					
1	49,5	48,7	8351	1,00	1,00
2	39,7	40,3	4239	1,97	0,99
3	37,9	34,7	3031	2,70	0,92
4	33,0	30,8	2235	3,74	0,93
8	35,4	32,8	1283	6,51	0,81
16	37,0	35,7	707	11,81	0,74
32	42,0	39,8	398	21,01	0,66
Strypinių konstrukcijų optimizavimo šakų ir rėžių algoritmas					
1	3,7	6,0	1027	1,00	1,00
2	3,6	5,2	577	1,78	0,89
4	3,9	4,8	397	2,59	0,65
8	7,9	6,2	304	3,38	0,42
16	7,7	6,6	190	5,41	0,34
32	7,7	8,1	111	9,29	0,29

Be to iš lentelės matyti, kad didinant procesorių skaičių mažėja ir užduočių optimizavimo laikai. Dėl to algoritmai gana efektyvūs ir spartūs.

4.6. Elektros energijos taupymo strategijos tyrimas

Disertacijoje siūloma nauja klasterių darbo strategija (3.2 poskyrius). Jos privalumas – kompiuteriai išjungiami, kai nėra naudojami. Atlikus matavimus paaiškėjo, jog siūlomas elektros energijos taupymo būdas yra naudingas taikant jį klasteriams bei gridams su Linux operacinėmis sistemomis. 4.11 lentelėje pateiktas klasterio iš 16 kompiuterių su Intel I5-760 procesoriais elektros energijos suvartojimas priklausomai nuo naujos taupymo strategijos taikymo.

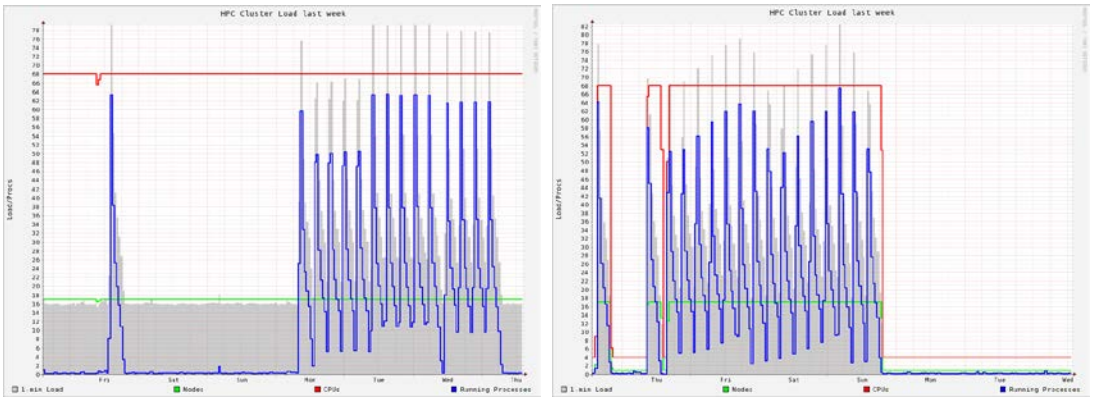
Iš 4.11 lentelės matyti, kad per tą patį laiką klasteris, dirbantis pagal siūlomą darbo būdą, suvartojo 30 % mažiau elektros energijos negu klasteris dirbantis įprastu būdu. Tokiu būdu galima apskaičiuoti apytikslųjį elektros energijos suvartojimo mažinimą per metus: (1 106 – 777) * 24 * 365 = 2 882 kWh per metus. Klasterio veikimo iliustracija, paimta iš GANGLIA klasterio darbo stebėjimų sistemos, atvaizduota 4.9a paveikslėlis (be taupymo būdo) ir

4.9b paveikslėlis (su taupymo būdu). Paveikslėliuose matyti, kad klasterio kompiuteriai dirbant siūlomu būdu buvo išjungiami ir įjungiami pagal poreikį: žalia linija rodo, kiek kompiuterių įjungta, raudona – kiek branduolių yra pasiekiama, mėlyna – kiek procesų (užduočių) dabartiniu metu yra paleista. Sistemoje visada rodoma vienu kompiuteriu daugiau negu yra kompiuterių-darbininkų, tai – kompiuteris-šeimininkas. Visa sistema veikia automatiniu būdu, kadangi klasterio kompiuterių išjungimo ir įjungimo scenarijai (3.2 poskyrius) reguliariai paleidžiami CRON tarnybos pagalba.

4.11 lentelė. *Naujas elektros energijos taupymo būdas, lyginant su įprastu klasterio darbu*

	Laikas, val.	Bendras elektros energijos suvartojimas, kWh	Suvartojimas / 1 val., Wh	Sutaupymas, %
Įprastas klasterio darbas	171	189,07	1 106	–
Klasterio darbas su nauju elektros energijos taupymo būdu	171	132,12	777	30

Po to buvo išmatuoti klasterio „fantome load“ bei klasterio kompiuterių tuščia eiga (Idle) (4.12 lentelė). Iš lentelės duomenų matosi, kad siūlomas elektros energijos taupymo būdas labai efektyvus, jei klasteris mažai naudojimas – 16 įjungtų kompiuterių elektros energijos suvartojimas yra 835 ± 10 Wh, tame tarpe tokių kompiuterių išjungimas kai jie nėra naudojami priverčia juos dirbti „Phantome load“ režime ir 16 kompiuterių suvartojimas yra tik 80 ± 5 Wh. Tokiu būdu siūlomo būdo teorinis naudingumas yra iki 90 %. Kai klasteris apkraunamas darbais, tas rodiklis krenta, tai ir parodyta 4.11 lentelėje. Tokiu darbo režimu, kaip klasteris dirbo 4.9b paveikslėlyje, siūlomas būdas sutaupo 30 % elektros energijos. Dėl to galima daryti išvadą, kad siūlomas elektros energijos taupymo būdas gali padėti sumažinti klasteriuose iki 90 % elektros energijos sąnaudas, bet tik tada kai klasteris nepilnai apkrautas.



a) *b)*

4.9 pav. *Klasterio darbas pagal Ganglia: a) įprastas, b) naudojant siūlomą elektros energijos taupymo būdą*

4.12 lentelė. *Kompiuterių su Intel I5 740 procesoriais elektros energijos suvartojimas „Idle“ ir „Phantome Load“ būsenoje*

Kompiuterių skaičius, vnt.	Energijos suvartojimas	
	Tuščia eiga (Idle), Wh	Šešėlinis apkrovimas (Phantome load), Wh
1	50 ± 5	5 ± 2
2	110 ± 5	10 ± 2
3	155 ± 5	15 ± 2
4	200 ± 8	20 ± 2
5	255 ± 8	25 ± 2
6	305 ± 8	30 ± 3
7	360 ± 8	35 ± 3
8	410 ± 12	40 ± 3
9	460 ± 12	45 ± 3
10	510 ± 12	50 ± 3
11	565 ± 15	55 ± 4
12	620± 15	60 ± 4
13	670 ± 15	65 ± 4
14	725 ± 20	70 ± 5
15	780 ± 20	75 ± 5
16	835 ± 20	80 ± 5

4.7. Ketvirtojo skyriaus rezultatai ir išvados

Šiame skyriuje pateikti eksperimentinių tyrimų rezultatai. Pateikti kompiuterinių tinklų, strypinių konstrukcijų optimizavimo bei siūlomo elektros energijos suvartojimo mažinimo būdo eksperimentiniai tyrimai.

Vykdant skaičiavimus neverta naudoti Hyper-Threading technologijos, nes ne visada gaunamas teigiamas rezultatas ją panaudojus. Gigabitinis kompiuterių tinklas yra naudingas vykdant lygiagrečiuosius skaičiavimus.

Tyrimų metu nustatyta, kad LitGrid tinklas Lietuvoje veikia ne taip gerai kaip norėtųsi. Klasteriai yra įvairūs dydžiu ir kompiuterių pajėgumu. Klasteriuose ne visada veikia ir MPI biblioteka, dėl to 3.2 poskyryje buvo siūlomas lygiagrečiųjų uždavinių vykdymo scenarijus 3.3, skirtas grid tinklui ir nereikalaujantis MPI bibliotekos. Tokiu būdu buvo išspręsta lygiagrečiųjų uždavinių vykdymo gride problema.

Išanalizavus strypinių konstrukcijų optimizavimo užduoties vykdymą buvo nustatyta, kad šakų ir rėžių algoritmas kur kas spartesnis už pilnojo perrinkimo algoritmą. Užduoties sprendimo sparta auga didėjant kompiuterių skaičiui. Tai labai aiškiai matosi, kai atliekamas sudėtingos konstrukcijos optimizavimas. Kai skaičiavimus atlieka 1 kompiuteris šakų ir rėžių algoritmas 3,6 kartų spartesnis už pilnojo perrinkimo, 16 kompiuterių – tas rodiklis krenta iki 1,4 karto. Efektyvumas ne visada aukštas kaip ir utilizacija – tai vyksta dėl užduoties vykdymo specifikos. Procesai-darbininkai neapsikeičia esama informacija tarpusavyje, dėl to auga uždavinio sprendimo laikas. Tačiau pasiūlytas kombinatorinis šakų ir rėžių algoritmo taikymas visada pateikia uždavinio globalųjį optimumą.

Vykdant elektros energijos suvartojimo tyrimus pastebėta, kad gridas ne visada apkrauna grido klasterio kompiuterius, t. y. išskaido uždavinius tarp kompiuterių pilnai jų neapkraunant (nors gali apkrauti tik kelius iš jų). Tai didina klasterio elektros energijos suvartojimą (ši problema pastebėta ir kitų mokslininkų (2 skyrius)). Klasteriai su Rocks OS tokios problemos neturi. Jie visada maksimaliai efektyviai apkrauna klasterio mazgus-darbininkus. Bet žaliosios kompiuterijos atžvilgiu ir klasteriai turi trūkumų. Klasterio prastovos metu kiekvienas iš kompiuterių suvartoja apie 50 Wh, o tai beveik pusė to, kiek vidutiniškai suvartoja pilnai apkrautas kompiuteris (apie 100-120 Wh).

Jeigu klasteris didžiąja laiko dalį laukia naujų užduočių, elektros energija yra suvartojama netikslingai.

Siūlomas elektros energijos taupymo būdas leidžia klasteriuose ir griduose sutaupyti iki 90 % elektros energijos suvartojimo. Tokių rezultatų galima pasiekti jei klasteriai mažai naudojami. Didinant klasterių naudojimo intensyvumą, šis rodiklis krenta. Siūlomas būdas įdiegtas klasteryje hpc.mii.vu.lt ir veikia, tuo galima įsitikinti klasterio darbų stebėjimų sistemoje.

5

Bendrosios išvados

Disertacijoje pristatyta žaliosios kompiuterijos sritis. Pasaulyje tai labai aktuali problema, bet Lietuvoje jai skiriamas per mažas dėmesys. Elektros energijos poreikis kiekvienais metais auga, nes vystomos technologijos, didėja klasterių ir superkompiuterių skaičius.

Disertacijoje pristatyti lygiagretieji kompiuteriai ir parodyta, kad žaliosios kompiuterijos atžvilgiu jie yra netobuli: neoptimalus užduočių paskirstymas tarp skaičiuojamųjų mazgų, pernelyg didelis elektros energijos suvartojimas. Reikia mažinti lygiagrečiųjų kompiuterių įtaką aplinkai ir spręsti kitas gridų, debesų kompiuterių, superkompiuterių bei klasterių problemas.

Siūlomas lygiagrečiųjų ir paskirstytųjų sistemų elektros suvartojimo mažinimo būdas pasižymi gerais rezultatais. Eksperimentinių tyrimų rezultatai leido daryti šias išvadas:

1. Siūlomas šakų ir rėžių algoritmas strypinių konstrukcijų optimizavimui veikia sparčiau (nuo 2,37 kartų 7 mazgams iki 3,6 kartų 8 mazgams) už pilnojo perrinkimo algoritmą. Pasiūlytu

algoritmu per priimtiną laiką galima išspręsti iki 8 mazgų strypinių konstrukcijų optimizavimo uždavinius.

2. Pasiūlytas strypinių konstrukcijų optimizavimo gride šakų ir rėžių algoritmu būdas leidžia vykdyti paskirstytuosius skaičiavimus ir be įdiegtų pranešimų siuntimo bibliotekų, o taip pat ir neapsiribojant vienu klasteriu. Siūlomo būdo trūkumas – procesai-darbininkai neapsikeitinėja informacija tarpusavyje, dėl to gali išaugti reikalingų atlikti skaičiavimų kiekis. Paskirstytuoju algoritmu per priimtiną laiką galima išspręsti iki 9 mazgų strypinių konstrukcijų optimizavimo uždavinius.
3. Eksperimentų metu buvo nustatyta, kad grido paskirstytųjų skaičiavimų sistemos užduočių skirstymas nėra optimalus, dažnai klasteriai apkraunami neefektyviai skirstant uždavinius tarp mazgų-darbininkų. Dėl to skaičiavimams atlikti suvartojama kur kas daugiau elektros energijos (iki 2,5 karto) negu reikėtų skirstant užduotis optimaliai.
4. Lygiagrečiųjų skaičiavimų klasteriai bei grido klasteriai vartoja elektros energiją net ir nevykdydami skaičiavimų – 50 Wh per valandą vienam kompiuteriui.
5. Disertacijoje pasiūlytas elektros energijos taupymo būdas – automatiškai išjunginėti mazgus-darbininkus kol nėra skaičiavimo užduočių. Siūlomas būdas leidžia sumažinti klasterio elektros energijos suvartojimą iki 90 %, kai kompiuteriai laukia užduočių. Kai 16 kompiuterių klasteris su Intel Core i5 CPU 760 @ 2.80GHz procesoriais yra apkrautas apie 50 %, sutaupoma apie 2882 kWh elektros energijos per metus.

Literatūra ir šaltiniai

Anderson, D. 2004. BOINC: A System for Public-Resource Computing and Storage. *In Proccedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Washington, DC, USA: IEEE Computer Society. 4–10.

Bagdonavičius, A.; Lapienis, S. 2005. *LitGrid*. Paimta 2012 m. 02 19 d. iš LitGrid: www.litgrid.lt

Baliga, J.; Ayre, R.; Hinton, K.; Tucker, R. 2011. Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport. Proceedings of the IEEE, Vol. 99, No. 1, 149–167.

BalticGrid II, a. 2012. *Balticgrid CA*. Paimta 2012 m. 02 19 d. iš Balticgrid CA: http://ca.balticgrid.org/

Banerjee, A.; Mukherjee, T.; Varsamopoulos, G.; Gupta, S. K. 2010. Cooling-Aware and Thermal-Aware Workload Placement for Green HPC Data Centers. *In Proccedings of the 2010 International Green Computing Conference*, Washington, DC, USA: IEEE Computer Society. 245–256.

Baravykaitė, M. M. 2006. *Lygiagrečiųjų algoritmų šablonų tyrimas ir kūrimas. Daktaro disertacija*. Vilnius: Technika, VGTU.

Berral-García, J. L.; Torres, J.; and others. 2012. *Green IT Conference List – Green ICT Conference List.* Paimta 2012 m. 04 30 d. iš Green IT Conference List – Green ICT Conference List: http://www.greenit-conferences.org/

Blum, C.; Rolli, A. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys, vol. 35, 268–308.

Brimkov, V. E.; Leachb, A.; Wub, J.; Mastroianni, M. 2012. Approximation Algorithms for A Geometric Set Cover Problem. *In Proccedings of the Discrete Applied Mathematics, Vol. 160, nr. 7–8,* Amsterdam, The Netherlands: Elsevier Science Publishers B. V. 1039–1052.

Bronshtein, I. N.; Semendyayev, K. A.; Musion, G.; Muehling. 2004. *Handbook of Mathematics.* Berlin: Springer. ISBN 3-540-43491-7

Chai-Arayalert, S.; Nakata, K. 2011. The Evolution of Green ICT Practice: UK Higher Education Institutions Case Study. *In Proccedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications (GreenCom '11)*, Washington, DC, USA: IEEE Computer Society. 220–225.

Chakraborty, P.; Bhattacharyya, D.; Sattarova, N.; Bedajna, S. 2009. Green computing: Practice of Efficient and Eco-Friendly Computing Resources. International Journal of of Grid and Distributed Computing, Vol.2, No.3, 33–38.

Chandra, R.; Daguum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. 2000. *Parallel Programming in OpenMP*. San Francisco, CA, USA: Morgan Kaufmann.

Chen, D.; Wang, L.; Wang, S.; Xiong, M.; Laszewski, G.; Li, X. 2010. Enabling Energy-Efficient Analysis of Massive Neural Signals Using GPGPU. *In Proccedings of the Proceedings of the 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom), Green Computing and Communications (GreenCom)*, Washington, DC, USA: IEEE Computer Society. 147–154.

Cherkassky, B. V.; Goldberg, A. V.; Radzik, T. 1994. Shortest Paths Algorithms: Theory and Experimental Evaluation. *In Proccedings of the Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. 516–525.

Citrome, L. 2008. Creating A More Productive, Clutter-Free, Paperless Office: A Primer on Scanning, Storage and Searching of PDF Documents on Personal Computers. International Journal of Clinical Practice, Vol. 62, No. 3, 363–366.

Čiegis, R. 2005. *Lygiagretieji algoritmai ir tinklinės technologijos.* Vilnius: Technika.

Čiegis, R. 2007. *Duomenų struktūros, algoritmai ir jų analizė.* Vilnius: Technika.

Čiegis, R.; Belevičius, R.; Žilinskas, J.; ir kiti. 2007-2009. *GridGlobOpt*. Paimta 2012 m. 01 10 d. iš http://www.gridglobopt.vgtu.lt/

Čiegis, R.; Šablinskas, R. 1998. Šeimininkas darbininkas lygiagrečiojo algoritmo efektyvumo analizė. 390–392.

Dagienė, V.; Grigas, G.; Jevsikova, T. 2008. *Enciklopedinis kompiuterijos žodynas. 2-as papildytas leidimas.* Vilnius: TEV.

Daintith, J.; Edmund, T.; Wright, T. 2008. *A Dictionary of Computing*. Oxford: Oxford University Press.

Diwekar, U. 2008. *Introduction to Applied Optimization, Second Edition.* Cambridge: Springer-Verlag.

Feng, W.-C. 2006. Global Climate Warming? Yes … In the Machine Room. Clusters and Computational Grids for Scientific Computing, 1.

Feng, W.-C.; Ching, A.; Hsu, C.-H. 2007. Green Supercomputing in a Desktop Box. *In Proccedings of the IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007*, Long Beach, California, USA: IEEE International. 1–8.

Feng, W.-C.; Feng, X.; Ce, R. 2008. Green Supercomputing Comes of Age. *In Proccedings of the IT Professional, Vol. 10, No. 1*, Piscataway, NJ, USA: IEEE Educational Activities Department. 17–23.

Filatovas, E. 2012. *Daugiakriterinių optimizavimo uždavinių sprendimas interaktyvioju būdu. Daktaro disertacija.* Vilnius: Vilniaus universitetas.

Flynn, M. 1972. Some Computer Organizations and Their Effectiveness. IEEE Trans. Comput., Vol. 21, No. 9, 948–960.

Floudas, A. C.; Pardalos, P. M. 2009. *Encyclopedia of Optimization* (2). New York: Springer.

Gallo, G.; Hammer, P. L.; Simeone, B. 1980. Quadratic Knapsack Problems. Mathematical Programming, Vol. 12, 132–149.

Golub, G.; Van Loan, C. K. 1996. *Matrix Computations* (Third ed.). Baltimore, Maryland: Jon Hopkins University Press.

Graham, R. L.; Pavol, H. 1985. On the History of the Minimum Spanning Tree Problem. Annals of the History of Computing, Vol. 7, No. 1, 43–57.

Gropp, W.; Lusk, E.; Skjellum, A. 1995. *Using MPI: portable parallel programming with the message-passing interface.* London: The MIT Press.

Grossman, E. 2006. *High Tech Trash: Digital Devices, Hiddent Toxics, and Human Health.* Washington: Island Press.

GStreamer Community. 2010 m. 10 22 d.. *gst-launch-0.10.* Paimta 2011 m. 05 14 d. iš GStreamer: http://gstreamer.freedesktop.org/

Horst, R.; Pardalos, P. M.; Thoai, N. V. 1995. *Introduction to Global Optimization. Nonconvex Optimization and its Application.* Dordrecht: Kluwer Academic Publishers.

Horst, R.; Tuy, H. 1987. On the Convergence of Global Methods in Multiextremal Optimization. Journal of Optimization Theory and Applications, Vol. 54, 253–271.

Horst, R.; Tuy, H. 1996. *Global Optimization: Deterministics Approaches.* Berlin: Springer.

Horvath, A.; Masanet, E. 2006. Enterprise Strategies for Reducing the Life-Cycle Energy Use and Greenhouse Gas Emissions of Personal Computers. *In Proccedings of the Proceedings of the 2006 IEEE International Symposium on Electronics and the Environment,* Scottsdale, AZ: IEEE Xplore. 21–26.

HP company. 2010 m. 04 26 d.. *HP Web Support*. Paimta 2012 m. 03 02 d. iš Drivers and instructions: http://www8.hp.com/us/en/support-drivers.html

Hu, L.; Jin, H.; Liao, X.; Xiong, X.; Liu, H. 2008. Magnet: a Novel Scheduling Policy for Power Reduction in Cluster with Virtual Machines. *In Proccedings of the 2008 IEEE International Conference on Cluster Computing*, Tsukuba: IEEE Xplore. 13–22.

Ivanikovas, S. 2009. *Lygiagrečių skaičiavimų taikymo daugiamačiams duomenims vizualizuoti problemos. Daktaro disertacija.* Vilnius: Vytauto Didžiojo universitetas.

Yang, C.-T.; Li, K.-C. 2005. The Anatomy of a Course in Cluster and Grid Computing. Information Technology: Research and Education, 403–407.

Jakušev, A. 2007. *Diferencialinių lygčių sistemų skaitinio sprendimo algoritmų lygiagretinimo technologijos kūrimas, analizė ir taikymai. Daktaro disertacija.* Vilnius: Technika, VGTU.

Kyriazis, D.; Menychtas, A.; Kousiouris, G.; others, a. 2010. A Real-time Service Oriented Infrastructure. GSTF International Journal on Computing, Vol. 1, No. 2, 39–44.

Klamroth, K.; Tind, J.; Zust, S. 2004. Integer Programming Duality in Multiple Objective Programming. Journal of Global Optimization, Vol. 29, No. 1, 1–18.

Kreher, D. L.; Stinson, D. R. 1998. *Combinatorial Algorithms: Generation, Enumeration, and Search.* New York, NY, USA: CRC Press.

Land, A.; Doig, A. 1960. An Autmatic Method of Solving Discrete Programming Problems. Econometrica, Vol. 28, 497–520.

Li, C.-H.; Wang, S.-T.; Chang, H.-Y.; Shen, C.-Y. 2010. An Efficient Approach for Reducing Power Consumption in a Production-Run Cluster. *In Proccedings of the 2010 Third International Joint Conference on Computational Science and Optimization (CSO),* Washington, DC, USA: IEEE Computer Society. 293–299.

LI, Z. S. 2009. *GPU High-Performance Computing of CUDA.* JiangSu, JS, China: China Water Power Press Pub.

Linpack. 2012. *Linpack*. Paimta 2012 m. 05 01 d. iš Linpack: http://www.netlib.org/linpack/

Little, J.; Murty, K.; Sweeney, D.; Karel, C. 1963. An Algorithm for The Traveling Salesman Problem. Operations Research, Vol. 11, 972–989.

Livré, R. 2004. *Topology*. Paimta 2010 iš http://www.math.huji.ac.il/~erezla/TOP/top.pdf

Lyndon, C.; Glendining, I.; Hempel, R. 1994. The MPI Mesage Passing Interface Standart. Programming environments for massively parallel distributed system, 213–218.

Llewellyn, D. C. 1993. A Primal Dual Integer Programming Algorithm. Discrete Applied Mathematics, Vol. 45, No. 3, 261–275.

Mair, J.; Leung, K.; Huang, Z. 2010. Metrics and Task Scheduling Policies for Energy Saving in Multicore Computers. *In Proccedings of the 11th IEEE/ACM International Conference on Grid Computing (GRID),* Brussels: IEEE Computer Society. 266–273.

Medvedev, V. 2007. *Tiesioginio sklidimo neuroninių tinklų taikymo daugiamačiam duomenims vizualizuoti tyrimai. Daktaro disertacija.* Vilnius: Technika, VGTU.

Meuer, H.; Dongarra, J. 2011. *TOP500 Supercomputing Sites*. Paimta 2011 m. 10 15 d. iš www.top500.org

Mockus, J. B.; Mockus, L. J. 1991. Bayesian Approach to Global Optimization and Application to Multiobjective and Constrained Problems. *In Proccedings of the J. Optim. Theory Appl., Vol. 70, No. 1. 70*, 157–172. New York, NY, USA: Plenum Press.

Morse, C.; Shirazi, B. 2010. *Third International Green Computing Conference (IGCC'12): Sustainable Computing and Computing for Sustainability*. Paimta 2012 m. 04 30 d. iš Third International Green Computing Conference (IGCC'12): Sustainable Computing and Computing for Sustainability: http://www.green-conf.org/

Munkres, J. R. 1975. *Topology.* Englewood Cliffs, New Jersey: Prentice-Hall.

Muntean, I.; Joldos, M. 2011. Experiences With the Development of Applications for Heterogeneous Grids. 2011 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 519–526.

Murugesan, S. 2008. Harnessing Green IT: Principles and Practices. IEEE IT Professional, Vol. 10, No. 1, 24–33.

Nocedal, J.; Wright, S. J. 2000. *Numerical Optimization.* Berlin: Springer.

Nordman, B.; Meier, A.; Piette, M. A. 1998. PC and Monitor Night Status: Power Management Enabling and Manual Turn-Off. *In Proccedings of the Proceedings of the American Council for an Energy Efficient Economy (ACEEE) Summer Study on Energy Efficiency in Buildings,* Berkeley: Lawrence Berkeley National Laboratory. 7.89–7.100.

NVIDIA corporation. 2011. *NVIDIA CUDA: NVIDIA CUDA C Programming Guide v4.1.* Santa Clara: NVIDIA Corporation.

Oliker, L. 2009. Green Flash: Designing an Energy Efficient Climate Supercomputer. *In Proccedings of the IEEE International Symposium on Parallel & Distributed Processing,* Washington, DC, USA: IEEE Computer Society. psl. 1.

Oliveira, J. P. 2005 m. 07 26 d.. *Wakeonlan*. Paimta 2010 m. 07 27 d. iš Wakeonlan: http://gsd.di.uminho.pt/jpo/software/wakeonlan/

OpenMP, C. a. 2011 m. 07. *OpenMP 3.1 Specification*. Paimta 2012 m. 05 10 d. iš OpenMP: http://www.openmp.org/mp-documents/OpenMP3.1.pdf

Ortega, J. M. 1988. *Introduction to Parallel and Vector Solution of Linear Systems.* New York: Plenum Press.

Papadimitriou, C. H.; Roughgarden, T. 2008. Computing Correlated Equilibria in Multi-Player Games. J. ACM, Vol. 55, 1–29.

Paulavičius, R. 2010. *Globalusis optimizavimas su simpleksiniais posričiais. Daktaro disertacija.* Vilnius: Kauno technologijos universitetas.

Petkus, T. 2001. *Kompiuterinio tinklo panaudojimas interaktyviame optimizavime. Daktaro disertacija.* Vilnius: VPU.

Piff, M. 1991. *Discrete Mathematics: An Introduction for Software Engineers.* Cambridge: Cambridge University Press.

Poole, D.; Mackworth, A.; Goebel, R. 1998. *Computational Intelligence: A Logical Approach.* New York: Oxford University Press.

Przybyla, D.; Pegah, M. 2007. Dealing With The Veiled Devil: Eco-Responsible Computing Strategy. *In Proccedings of the Proceedings of the 35th annual ACM SIGUCCS Conference on User Services*, Orlando: ACM. 296–301.

Ralph, D. 1990. A Survey of Parallel Computer Architectures. IEEE Computer, Vol. 23, No. 2, 5–16.

Rathore, N.; Chana, I. 2011. A Cognitive Analysis of Load Balancing and Job Migration Technique in Grid. *In Proccedings of the World Congress on Information and Communication Technologies (WICT)*, Washington, DC, USA: IEEE Computer Society. 77–82.

Russell, S. J.; Norvig, P. 2003. *Artificial Intelligence: A Modern Approach.* New Jersey: Prentice Hall.

Ruth, S. 2009. Green IT More Than a Three Percent Solution? *In Proccedings of the IEEE Internet Computing*, Washington, DC, USA: IEEE Computer Society. 74–78.

Salah, K.; Al-Shaikh, R.; Sindi, M. 2011. Towards Green Computing Using Diskless High Performance Clusters. *In Proccedings of the Proceedings of the 7th International Conference on Network and Services Management*, Laxenburg, Austria, Austria: International Federation for Information Processing. 1–4.

Schott, B.; Emmen, A. 2010. Green Methodologies in Desktop-Grid. *In Proccedings of the Proceedings of the 2010 International Multiconference on Computer Science and Information Technology (IMCSIT), Vol. 5*, Wisla: IMCSIT. 671–676.

Scogland, T.; Subramaniam, B.; Lockhart, J.; Lin, H. 2012. *Green500.org About.* Paimta 2012 m. 03 10 d. iš Green500.org: http://www.green500.org/about.php

Sharma, S.; Hsu, C.; Feng, W. 2006. Making a Case for a Green500 List. *In Proccedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Washington, DC, USA: IEEE Computer Society. psl. 299.

Simpson, W. 2003. Energy Sustainability and the Green Campus. Planning for Higher Education, Vol. 31, No. 3, 150–158.

Starikovičius, V. 2002. *Daugiafazinio tekėjimo modelių lygiagretieji skaitiniai sprendimo algoritmai. Daktaro disertacija.* Vilnius: VGTU.

Šablinskas, R. 1999. *Lygiagrečiųjų algoritmų tyrimas paskirstytos atminties lygiagretiesiems kompiuteriams. Daktaro disertacija.* Kaunas: VDU.

Šešok, D. 2006. Strypinių sistemų optimizavimas. 9-osios Lietuvos jaunųjų mokslininkų konferencijos "Mokslas - Lietuvos ateitis" medžiaga, 151–155.

Šešok, D. 2008 a. *Santvarų topologijos optimizavimas genetiniais algoritmais. Daktaro disertacija.* Vinius: Technika, VGTU.

Šešok, D. 2008 b. Mišrus genetinis algoritmas santvarų optimizavimui. Lietuvos matematikos rinkinys, LMD darbai, 48/49, 239–244.

Šešok, D.; Belevičius, R. 2007 a. Modified Genetic Algorithm for Optimal Design of Truss Structures. Mechanika, Nr. 6(68), 53–59.

Šešok, D.; Belevičius, R. 2007 b. Use of genetic algorithms in topology optimization of truss structures. Mechanika, Nr. 2(64), 34–39.

Šešok, D.; Belevičius, R. 2008. Global Optimization of Trusses with a Modified Genetic Algorithm. Journal of Civil Engineering and Management, Vol. 14, No. 3, 147–154.

Šešok, D.; Ragauskas, P. 2007. Santvarų topologijos ir formos optimizavimas genetiniais algoritmais. Lietuvos matematikos rinkinys, 47, spec. nr. , 484–488.

Šilko, G. 2003. *Lygiagrečiųjų algoritmų tyrimas ir taikymas lazerinėse technologijose. Daktaro disertacija.* Vilnius: Technika, VGTU.

Talebi, M. 2008. *Computer Power Consumption Benchmarking For Green Computing. Master's Thesis.* Villanova: Villanova University, Department of Computing Sciences.

Talebi, M.; Way, T. 2009. Methods, Metrics and Motivation for a Green Computer Science Program. *In Proccedings of the SIGCSE Bull., Vol. 41, No. 1,* New York, NY, USA: ACM. 362–366.

The IEEE; The Open Group. 2008. *Crone*. Paimta 2012 m. 04 20 d. iš The Open Group Base Specifications Issue 7: The Open Group Base Specifications Issue 7

Thiebaut, D. 1995. *Parallel Programming in C for the Transputer*. Paimta 2012 m. 2 20 d. iš Parallel Programming in C for the Transputer: http://maven.smith.edu/~thiebaut/transputer/toc.html

Tseng, C.; Figueira, S. 2010. An Analysis of the Energy Efficiency of Multi-threading on Multi-core Machines. *In Proccedings of the International Conference on Green Computing,* Los Alamitos, CA, USA: IEEE Computer Society. 283–290.

Tsuchiyama, R.; Nakamura, T.; Iizuka, T.; Asahara, A.; Miki, S. 2010. *The OpenCL Programming Book.* Washington, DC, USA: Fixstars Corporation.

University of California. 2011. *Base Users Guide*. Paimta 2012 m. 05 13 d. iš Rocks Website: http://www.rocksclusters.org/roll-documentation/base/5.5/

University of California; Berkeley MillenniumProject. 2010 m. 03 10 d.. *Ganglia Monitoring System*. Paimta 2012 m. 05 20 d. iš Ganglia Monitoring System: http://ganglia.sourceforge.net/

Valstybinė lietuvių kalbos komisija. 2003. *Valstybinė lietuvių kalbos komisija*. Paimta 2012 m. 03 15 d. iš http://www.vlkk.lt/lit/2989: http://www.vlkk.lt/lit/2989

Wakatani, A. 2012. Implementation of Fractal Image Coding for GPGPU Systems and its Power-Aware Evaluation. *In Proccedings of the 2012 IEEE International Systems Conference (SysCon),* Washington, DC, USA: IEEE Computer Society. 1–5.

Wang, G.; Lin, Y.; Yi, W. 2010. Kernel Fusion: An Effective Method for Better Power Efficiency on Multithreaded GPU. *In Proccedings of the Proceedings of the 2010 IEEE/ACM Int'l*

Conference on Green Computing and Communications \& Int'l Conference on Cyber, Physical and Social Computing, Washington, DC, USA: IEEE Computer Society. 344–350.

Wang, Z.; Zhang, Y.-Q. 2011. Energy-Efficient Task Scheduling Algorithms with Human Intelligence Based Task Shuffling and Task Relocation. *In Proccedings of the IEEE/ACM International Conference on Green Computing and Communications (GreenCom)*, Sichuan: IEEE. 38–43.

Williams, J.; Curtis, L. 2008. Green: The New Computing Coat of Arms? *In Proccedings of the IT Professional, Vol. 10, No. 1*, Piscataway, NJ, USA: IEEE Educational Activities Department. 12–16.

Witkowski, M.; Brenner, P.; Jansen, R.; Go, D.; Ward, E. 2010. Enabling Sustainable Clouds via Environmentally Opportunistic Computing. *In Proccedings of the IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, Washington, DC, USA: IEEE Computer Society. 587–592.

Witten, I. H.; Frank, E. 2011. *Data Mining: Practical Machine Learning Tools and Techniques.* USA: Morgan Kaufmann.

Woeginger, G. 2003. Exact Algorithms for NP-Hard Problems: A Survey. *In Proccedings of the Combinatorial optimization - Eureka, you shrink!*, New York, NY, USA: Springer-Verlag New York, Inc. 185–207.

Xiong, J.; Wang, J.; Xu, J. 2010. Research of Distributed Parallel Information Retrieval Based on JPPF. *In Proccedings of the 2010 International Conference of Information Science and Management Engineering (ISME)*, Washington, DC, USA: IEEE Xplore. 109–111.

Zhang, L.; Li, K.; Zhang, Y.-Q. 2010 a. Green Task Scheduling Algorithms with Energy Reduction on Heterogeneous Computers. *In Proccedings of the IEEE International Conference on Progress in Informatics and Computing (PIC)*, Shanghai: IEEE Society. 560–563.

Zhang, L.; Li, K.; Zhang, Y.-Q. 2010 b. Green Task Scheduling Algorithms with Speeds Optimization on Heterogeneous Cloud Servers. *In Proccedings of the 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom) on Green Computing and Communications (GreenCom)*, Hangzhou: IEEE Computer Society. 76–80.

Zhang, L.; Qi, D. 2006. Energy-Efficient Task Scheduling Algorithm for Mobile Terminal. *In Proccedings of the 2006 IET International Conference on Wireless, Mobile and Multimedia Networks*, Hangzhou: IEEE Xplore. 1–4.

Žilinskas, A. 2000. *Matematinis programavimas.* Kaunas: Vytauto Didžiojo universiteto leidykla.

Žilinskas, J. 2002. *Black Box Global Optimization: Covering Methods And Their Parallelization. Doctoral Disertation.* Kaunas: Kauno technologijos universitetas.

Žilinskas, J. 2008. Branch and Bound With Simplicial Partitions for Global Optimization. Mathematical Modelling and Analysis, Vol. 13, No. 1, 145–159.

Žilinskas, J. 2009. Parallel Global Optimization in Multidimensional Scaling. Parallel Scientific Computing And Optimization, Vol. 27, Part II, 69–82.

Žilinskas, J. 2011. Belaukiant eksnašiųjų skaičiavimų. Mokslas ir technika, 12(630), 24-29.

Žilinskas, J. 2012. Prallel Branch and Bound for Multidimensional Scaling with City-block distances. Journal of Global Optimization, ISSN 0925-5001, in press. doi:10.1007/s10898-010-9624-7, –.

Žilinskas, A.; Žilinskas, J. 2006. On Efficiency of Tightening Bounds in Interval Global Optimization. Lecture Notes in Computer Science, 3732, 197–205.

Žilinskas, A.; Žilinskas, J. 2009. Branch and Bound Algorithm for Multidimensional Scaling With City-Block Metric. Journal of Global Optimization, Vol. 43, No. 2–3, 357–372.

Боресков, А.; Харламов, А. 2010. *Основы работы с технологией CUDA*. Санкт-Петербург: ДМК-Пресс.

Воеводин, В.; Воеводин, В. 2002. *Параллельные вычисления.* Санкт-Петербург: БХВ-Петербург.

Autoriaus publikacijų sąrašas disertacijos tema

Straipsniai recenzuojamuose periodiniuose mokslo žurnaluose

A 1. Igumenov A.; Petkus T.2008. Analysis of Parallel Calculations in Computer Network. *Information Technology and Control*, ISSN 1392-124X, 37(1), 57-62. [Abstracted/Indexed in *ISI Web of Science, Inspec*].

A 2. Igumenov A.; Žilinskas J. 2009. Combinatorial Algorithms for Topology Optimization of Truss Structure. In: *Information technologies 2009 (IT 2009) 15th International Conference on Information and Software Technologies*, April 23-24 2009, Kaunas, Lithuania, ISSN 2029-0063, 2029-0055. pp. 229-234. [Abstracted/Indexed in *ISI Web of Science (Conference Proceedings Citation Index)*].

A 3. Igumenov A.; Žilinskas J. 2010. Combinatorial Topology Optimization of Truss Structures Using Distributed Grid

Computing. *Jaunųjų mokslininkų darbai*, ISSN 1648-8776, 1(26); Priedas, 277-280.

A 4. Igumenov A.; Žilinskas J. 2011. Power Consumption Optimization with Parallel Computing. *Jaunųjų mokslininkų darbai*, ISSN 1648-8776, 4(33); 119-122.

Straipsniai kituose mokslo leidiniuose

B 1. Igumenov A.; Žilinskas J.; Kurowski K.; Mackowiak M. 2010. Optimization of Topology of Truss Structures Using Grid Computing. In: *2010 IEEE International Conference on Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), Heraklion, Crete, Greece, September 20-24, 2010*, ISBN: 978-1-4244-8395-2. doi:10.1109/CLUSTERWKSP.2010.5613101 [Abstracted/Indexed in *IEEE Xplore*].

Priedai

1 lentelė. Pačių greičiausių superkompiuterių sąrašas

Eil. Nr.	Kompiuterio pavadinimas	Branduolių skaičius	Rmax (Tflop/s)	Rpeak (Tflop/s)	Galia (KW)
1	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect, Fujitsu	705024	10510	11280,4	12659,9
2	Tianhe-1A - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050, NUDT	186368	2566	4701	4040
3	Jaguar – Cray XT5-HE Opteron 6-core 2.6 GHz, Cray Inc.	224162	1759	2331	6950
4	Nebulae – Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050, Dawning	120640	1271	2984,3	2580
5	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows, NEC/HP	73278	1192	2287,6	1398,6
6	Cielo – Cray XE6, Opteron 6136 8C 2.40GHz, Custom, Cray Inc.	142272	1110	1365,8	3980

7	Pleiades – SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband, SGI	111104	1088	1315,3	4102
8	Hopper – Cray XE6, Opteron 6172 12C 2.10GHz, Custom, Cray Inc.	153408	1054	1288,6	2910
9	Tera-100 - Bull bullx super-node S6010/S6030, Bull SA	138368	1050	1254,5	4590
10	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband, IBM	122400	1042	1375,8	2345

2 lentelė. Dešimt greičiausių pasaulyje klasterių

TOP 500	**Klasterio pavadinimas**	**Branduolių skaičius**	**Rmax (Tflop/s)**	**Rpeak (Tflop/s)**	**Galia (KW)**
1	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect, Fujitsu	705024	10510	11280,4	12659,9
4	Nebulae – Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050, Dawning	120640	1271	2984,3	2580
5	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows NEC/HP	73278	1192	2287,6	1398,6
9	Tera-100 - Bull bullx super-node S6010/S6030, Bull SA	138368	1050	1254,5	4590
10	Roadrunner – BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz Voltaire Infiniband, IBM	122400	1042	1375,8	2345
14	Sunway Blue Light – Sunway BlueLight MPP, ShenWei processor SW1600 975.00 MHz, Infiniband QDR, National Research Center of Parallel Computer Engineering & Technology	137200	795,9	1070,2	1074
15	Zin - Xtreme-X GreenBlade GB512X, Xeon E5 (Sandy Bridge -	46208	773,7	961,1	924,2

	EP) 8C 2.60GHz, Infiniband QDR Appro International				
18	Lomonosov - T-Platforms T-Blade2/1.1, Xeon X5570/X5670 2.93 GHz, Nvidia 2070 GPU, Infiniband QDR, T-Platforms	33072	674,1	1373,1	2800
21	Mole-8.5 - Mole-8.5 Cluster, Xeon X5520 4C 2.27 GHz, Infiniband QDR, NVIDIA 2050, IPE, Nvidia, Tyan	29440	496,5	1012,6	540
24	Red Sky – Sun Blade x6275, Xeon X55xx 2.93 Ghz, Infiniband, Oracle	42440	433,5	497,4	–

Aleksandr Igumenov

LYGIAGRETIEJI IR PASKIRSTYTIEJI ELEKTROS
ENERGIJĄ TAUSOJANTYS SKAIČIAVIMAI

Daktaro disertacija

Technologijos mokslai,
Informatikos inžinerija (07T)

Aleksandr Igumenov

ELECTRICAL ENERGY AWARE PARALLEL
AND DISTRIBUTED COMPUTING

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (07T)