KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Vytautas Vaškevičius

Matematinių išraiškų pertvarkymo programinė įranga. Lygiagretaus funkcionavimo valdymas

Magistro darbas

Darbo vadovas: doc. dr. Romas Marcinkevičius

KAUNAS, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Matematinių išraiškų pertvarkymo programinė įranga. Lygiagretaus funkcionavimo valdymas

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė

Darbo vadovas
Programų inžinerijos katedros docentas
dr. R. Marcinkevičius

Recenzentas
Kompiuterių katedros docentas
dr. S. Maciulevičius

Atliko
IFM-8/2 grupės studentas
V. Vaškevičius

KAUNAS, 2004

TURINYS

Santrauka

Šis projektas – tai matematinių išraiškų pertvarkymo programinės įrangos gyvavimo ciklas. Galutinis produktas – tai sistema, kuri funkcionuoja integruodamasi su Maple ir MPI sistemomis. Maple – tai universalus matematinis programinis produktas [2], [3]. Ši sistema reikalinga patogiai įvesti ir išvesti duomenis, panaudoti standartines funkcijas. MPI – tai paskirstytosios procesorių sistemos valdymo servisas [6], [7]. Šis servisas reikalingas reikalavimuose apibrėžtiems pertvarkymams lygiagrečiai vykdyti naudojant kelis vieno procesoriaus kompiuterius.

Projekto tikslas – suprojektuoti, realizuoti ir atiduoti užsakovui naudoti sukurtą produktą vietoj dabartinių lėtesnių sprendimų. Būtent greitis yra pagrindinis šio produkto nefunkcinis reikalavimas. Dabartiniai sprendimai egzistuoja, tačiau jų laikiniai parametrai netenkina užsakovų. Dauguma tokių sprendimų yra universali matematinė programinė įranga, kuri naudoja vartotojui nesunkiai prieinamą vieną kompiuterio procesorių. Daugiaprocesorinės sistemos yra labai brangios. Mūsų sukurtas produktas yra specializuotas, dėl to jame yra greitesnis pats pertvarkymo algoritmas. Be to, mūsų produktas naudoja už daugiaprocesorinę sistemą daug pigesnį variantą – vieno procesoriaus kompiuterių, sujungtų į vietinį tinklą, procesorius. Lygiagrečiai (naudojant visus procesorius) atliekamas matematinės išraiškos pertvarkymas ypač sumažina bendrą uždavinio sprendimo laiką.

Kadangi resursai buvo labai riboti, sistemos realizacija užtuko 16 mėnesių. Per šį laiką buvo atlikta poreikio analizė, surinkti ir specifikuoti reikalavimai, suprojektuota sistemos architektūra, detaliai suprojektuoti sistemos komponentai, sistema realizuota ir ištestuota, įdiegta pas užsakovą. Projektas buvo pilnai dokumentuotas ir turi tokius dokumentus kaip projekto paraiška, projekto planas, reikalavimų specifikacija, architektūros specifikacija, detalios architektūros specifikacija, testavimo planas, testavimo ataskaita, diegimo ataskaita. Produktas taip pat dokumentuotas ir turi vartotojo dokumentaciją, reklaminę medžiagą, interneto svetainę.

Šiame dokumente detaliai nagrinėjama taikomoji sritis, analizuojami esami ir pasirinkti sprendimai bei algoritmai, pateikiami pagrindiniai produkto projektavimo ir realizavimo etapų aspektai. Pagrindinę dokumento dalį sudaro atliktas tyrimas. Tyrimo metu mūsų sistemos matematinių išraiškų pertvarkymams sugaištas laikas buvo lyginamas su kitomis sistemomis. Taip pat buvo analizuojama kaip pertvarkymų laiką įtakoja sistemos charakteristikos, pvz. matematinės išraiškos ilgis ar naudojamų procesorių kiekis. Tyrimų rezultatai pateikti lentelėmis ir grafikais, pagal tai tendencingai padarytos išvados.

Summary

This project is a life cycle of the software for reformation of the mathematical expressions. A final product is a system, which operate integrated with Maple and MPI systems. Maple is universal mathematical software product [2], [3]. This system is required to handily input and output data, to have ability using standard functions. MPI is a service for controlling the distributed system [6], [7]. This service is required for parallel reformations of the mathematical expressions using several single-processor computers.

The project goal is to design, implement and deliver to customer for usage developed product instead of current slower solutions. Exactly speed is the key non-functional requirement for the product. Current solutions exist, but the speed is unacceptable for the customer. Most solutions are universal mathematical software, which uses one single-processor computer, which is obtainable for most users. Multi-processor systems are very expensive. The product we developed is specialized, so it has high-speed reformation algorithm. Besides, our product uses cheaper solution then multi-processor system – it uses processors of intranet single-processor computers. Parallel (using all processors) performing the reformation of mathematical expressions especially reduces the total time of decision of the problem.

We have been developing the system for 16 months, because the resources were limited. We have done demand analysis, collected and specified requirements, designed system architecture and components, implemented the system, tested it and installed it for the customer while this period. The all project was documented by writing such documents as project order, project schedule, requirement specification, architecture specification, detail architecture specification, test schedule, test report, install report. The product also was documented and has user guide, publicity and Internet site.

We have deeply researched environment, analyzed present and selected solutions and algorithms, reported key aspects of product design and implementation stages in this document. The main part of the document consists of accomplished research. The time taken for reformation of mathematical expressions in our system was compared with the taken time in other systems during the research. It was also analyzed how the system characteristics, like length of the mathematical expression or number of processors, influence the reformation time. Research results are presented in tables and diagrams, accordingly figured tendentious conclusions.

1. ĮVADAS

Šiame skyriuje trumpai apžvelgiamas visas magistro baigiamasis darbas: apibrėžiama taikymo sritis, nustatomi darbo tikslai, trumpai aprašomas temos aktualumas, nurodoma taikoma metodika, apibrėžiamas tiriamasis objektas, pristatoma teorinė literatūra, nustatoma analizės tvarka, nurodoma dokumento struktūra.

1.1. Taikymo sritis

Taikomoji sritis – labai didelių (ilgų) matematinių išraiškų pertvarkymas. Tai labai konkreti ir sudėtinga programinės įrangos taikymo sritis. Labai didelėms matematinėms išraiškoms pertvarkyti naudojami daugiaprocesoriniai kompiuteriai arba vieno procesoriaus kompiuterių tinklai. Tokiose sistemose skaičiavimai yra suskaidomi į atskirus procesus ir vykdomi lygiagrečiai. Didelės projektavimo ir programavimo išlaidos bei santykinai maža paklausa lemia ypatingai dideles tokių sistemų kainas. Dėl to šioje srityje neįmanoma suderinti pigumo ir efektyvumo, todėl paprastesniems skaičiavimams efektyvesne programine įranga galime laikyti universalias matematines programas, kurios pasižymi ypač plačiu funkcionalumu, vidutiniu greičiu ir vis dar aukšta kaina. Dažniausiai naudojami šie universalūs matematiniai produktai: Mathcad, Maple, MatLab.

Tokioje taikomojoje srityje dirba matematikos, fizikos, chemijos ir kt. mokslininkų grupės. Taigi ir produktas yra orientuotas į šių mokslinių sričių specialistus. Savo produktą Lietuvoje galėtume siūlyti aukštųjų mokyklų mokslininkams, institutų ir laboratorijų specialistams. Jį galėtume publikuoti ir internete, kad apie produktą sužinotų ir užsienio mokslininkai bei specialistai.

Praktikoje didelės matematinės išraiškos yra gaunamos sprendžiant didesnes diferencialinių lygčių sistemas, skaičiuojant aukštos eilės diferencialą.

1.2. Projekto tikslai

1.2.1. Projekto tikslas ir adresatas

Projektas tikslas – sukurti numatytą produktą ir platinti jį tarp vartotojų, dirbančių taikomojoje srityje. Šiuo atveju, taikomoji sritis – tai matematinių išraiškų pertvarkymas. Šioje srityje dirba matematikos, fizikos, chemijos ir kt. mokslininkų grupės. Todėl ir produktas yra skirtas šių mokslo sričių specialistams.

1.2.2. Produkto užsakovas

Produkto užsakovas – Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros docentas dr. Romas Marcinkevičius. Užsakovas dažnai naudoja Maple sistemą ir yra suinteresuotas naudoti naująją specializuotą sistemą, kuri sukurta siekiant padidinti pertvarkymų greitį. Projekto eigoje visos autorinės teisės priklausė sistemos kūrėjams, o galutinio produkto autorinės teisės buvo perleistos ir priklauso užsakovui.

Kai produktas jau realizuotas, galimi ir kiti sistemos vartotojai, numatyti 1.2.1. skyrelyje.

1.3. Temos aktualumas

1.3.1. Problemos sprendimas pasaulyje

Matematinių išraiškų pertvarkymas – labai konkreti ir sudėtinga programinės įrangos taikymo sritis. Didelės projektavimo ir programavimo išlaidos bei santykinai maža paklausa lemia ypatingai dideles tokių sistemų kainas. Šioje srityje neįmanoma suderinti pigumo ir efektyvumo.

Išsiskiria tokios produktų klasės:

- universali matematinė programinė įranga: pritaikyta įvairioms platformoms, turi platų funkcionalumą, integralumą, patrauklią vartotojo sąsają;
- specializuota matematinė programinė įranga: pritaikyta konkrečiai taikymo sričiai, turi ribotą funkcionalumą, dažnai sudėtingą vartotojo sąsają.

Pasaulyje geriausiai žinomi matematinių produktų kūrėjai yra MathSoft, MathWorks, Waterloo Maple kompanijos. Jų produktai gerai žinomi visame pasaulyje. Tokių kompanijų produktai yra labai universalūs, turi patogias grafines sąsajas, puikias integracines priemones. Žinomiausi pasaulyje produktai yra Mathcad, MatLab, Mathematica, Maple, IDL, Gauss, LabView [1].

Nagrinėjamoje probleminėje srityje paskirstytos sistemos naudojamos gana retai, kadangi kiekvienas matematinio uždavinio tipas reikalauja vis kitokio procesorių valdymo mechanizmo. Kompanija Risc-Linz 1998 m. pradėjo kurti tokių mechanizmų aibę ir pavadino ją Distributed Maple [2].

1.3.2. Problemos sprendimas Lietuvoje

Peržiūrėjus prieinamus literatūros šaltinius, mums nepavyko rasti tokio Lietuvos programinės įrangos gamintojo, kuris būtų sukūręs ar kurtų universalią matematinę programinę įrangą. Tačiau Lietuvoje yra kuriami specializuotos paskirties užsakomieji projektai, kurie dažniausiai vis tiek susiveda į komplikuotos, daug resursų reikalaujančios programinės įrangos kūrimą.

1.4. Metodologija

1.4.1. Projekto vykdymo metodika

Visi projekto etapai buvo vykdomi pagal Programų inžinerijos standartus ir dėstytojų rekomendacijas. Projekto eigoje buvo išklausyti tokie naudingi moduliai kaip Programų inžinerijos procesai, Programų inžinerijos valdymas, Reikalavimų specifikavimas, Programų projektavimas, Duomenų projektavimas, Programų sistemos architektūros analizė, Programų testavimo metodai, Programinės įrangos įdiegimo tyrimas, Programinės įrangos kokybė, Programinės įrangos įrankiai, Informacinės technologijos projektavimo vadyboje. Vykdant projektą buvo griežtai atsižvelgta į šių modulių teorinę medžiagą, atsakingų dėstytojų rekomendacijas.

1.4.2. Produkto kūrimo technologijos

Projektavimo etapuose buvo naudojama UML projektavimo technologija. Realizavimo etape buvo naudojamas objektinis programavimas, C++ programavimo kalba ir šios kalbos kompiliatorius Linux operacinei sistemai. Testavimo vykdymui buvo naudojami specialiai pritaikyti automatinio testavimo įrankiai.

1.5. Analizės tvarka

Pirmiausia reikia detaliai aprašyti inžinerinę problemą. Problemai spręsti būtina išnagrinėti susijusią literatūrą. Tokiu būdu įsigilinus į problemą, galime išsikelti sau detalius darbo tikslus, parinkti ir pagrįsti tuos tikslus įgyvendinančius sprendimus. Sprendimams realizuoti reikia suprojektuoti ir realizuoti tinkamos architektūros bei greitų ir patikimų algoritmų visumos sistemą.

1.6. Tyrimo objektas ir tikslas

Teoriniuose ir eksperimentiniuose tyrimuose bus tiriamas sukurtas produktas. Tyrimo tikslas – ištirti matematinių išraiškų pertvarkymo greitį. Bus atlikti tyrimai, kokią įtaką matematinių išraiškų pertvarkymo greičiui daro įvairios sistemos ir duomenų charakteristikos, pvz. sistemos procesorių kiekis, lygiagrečių procesų kiekis, matematinės išraiškos ilgis ir sudėtingumas ir pan. Tuo pačiu bus atlikti pertvarkymų greičio palyginimai su kitais rinkoje esančiais produktais. Pagal tyrimų rezultatus bus padarytos pagrindinės išvados, nustatyta vizija sistemos plėtrai, pateiktos rekomendacijos naudotojams.

1.7. Dokumento struktūra

Magistro baigiamasis darbas susideda iš 6 esminių dalių: Įvadas, Analizė, Projektavimas, Tyrimas, Išvados ir priedai. Įvade bendrais bruožais apibendrinamas visas magistro darbas ir šio darbo dokumentas. Antroje dalyje analizuojama problema, nagrinėjama susijusi literatūra ir esmai sprendimai, nustatomi detalūs darbo tikslai, parenkami ir pagrindžiami tikslus įgyvendinantys sprendimai ir algoritmai. Projektavimo dalyje išdėstomi visų produkto projektavimo etapų dokumentai: pirminis produkto apibūdinimas, reikalavimų specifikacija, architektūros specifikacija ir detalios architektūros specifikacija. Taip pat šioje dalyje detalizuojama produkto programinė realizacija. Tyrimo dalyje apibrėžiamos tyrimo aplinkybės, išdėstomi atlikti eksperimentiniai tyrimai, įvertinami rezultatai. Penktoje dalyje pagal tyrimų rezultatus nustatomos ir išdėstomos darbo išvados, apibrėžiamos galimos produkto plėtros kryptys, pateikiamos rekomendacijos naudotojams. Šeštoji dalis susideda iš kelių skyrių: literatūros, santrumpų ir terminų žodynų bei dokumento priedų.

Šis dokumentas parengtas pagal bendruosius nurodymus baigiamiesiems darbams, dokumento struktūra sudaryta pagal Programų inžinerijos katedros rekomendacijas.

2. ANALIZĖ

Šiame skyriuje aprašoma sprendžiama problema, detaliai ji analizuojama, nagrinėjant susijusią literatūrą ir esamus sprendimus, iškeliami detalūs darbo tikslai, parenkami ir pagrindžiami tikslus įgyvendinantys sprendimai bei algoritmai.

2.1. Problemos aprašymas

Užsakovas iki šiol matematinėms išraiškoms pertvarkyti naudojo įvairius programinius paketus, tačiau sprendimų laikiniai parametrai užsakovo netenkino. Dauguma tokių sprendimų yra universali matematinė programinė įranga, kuri naudoja vartotojui nesunkiai prieinamą vieną kompiuterio procesorių. Daugiaprocesorinės sistemos yra labai brangios. Turėjo būti sukurta specializuota sistema, dėl to joje turėjo būti greitesnis pats pertvarkymo algoritmas. Be to, sistema turėjo naudoti už daugiaprocesorinę sistemą daug pigesnį variantą: vieno procesoriaus kompiuterių, sujungtų į vietinį tinklą, procesorius. Lygiagrečiai atliekamas matematinės išraiškos pertvarkymas taip pat turėjo sumažinti bendrą uždavinio sprendimo laiką.

Reikia tokios sistemos, kuri pakeistų šiuo metu užsakovo naudojamus programinius paketus ir būtų greitesnė. Greitis yra pagrindinis šios sistemos nefunkcinis reikalavimas. Pagrindinis funkcinis reikalavimas – matematinių išraiškų pertvarkymas.Taigi, apibendrinant, mūsų galutinis tikslas buvo sukurti greitą ir patikimą matematinių išraiškų pertvarkymo sistemą. Todėl turėjome tokias pagrindines problemas: diferencijavimo laiko minimizavimas bei patikimumo užtikrinimas.

2.1.1. Nagrinėjama problema

Kuriamai sistemai reikėjo sukurti matematinių išraiškų pertvarkymo išlygiagretinimo valdymo modulį, kuris greitai ir kuo optimaliau mokėtų valdyti matematinės išraiškos diferencijavimą naudojant kelis procesus. Sistema turi naudotis matematinių išraiškų pertvarkymo modulio paslaugomis išraiškoms diferencijuoti.

Reikia detaliai išnagrinėti lygiagrečių skaičiavimų metodikas, esamus sprendimus ir priimti tinkamus sprendimus kuriamai sistemai.

2.2. Problemos analizė

Problemai spręsti pirmiausia reikia detaliai išanalizuoti susijusią literatūrą bei esamus sprendimus. Išsiaiškinsime kas tai yra lygiagretūs skaičiavimas, kokia jiems yra reikalinga aparatūra, kaip organizuojamas ryšys tarp atskirų procesorių, kaip vertinamas lygiagrečių sistemų našumas. Tai svarbiausios analizės dalys, kurios nulems kuriamos sistemos sprendimo pasirinkimą.

2.2.1. Lygiagretūs skaičiavimai

M. J. Quinn apibrėžia svarbiausius lygiagrečių skaičiavimų aspektus parodydamas jų analogiją su realiu pasauliu [14].

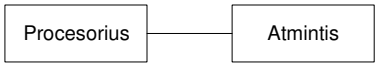
Įsivaizuokite biblioteką, kurios lentynose reikia pertvarkyti knygas. Vienas darbuotojas bando sudėlioti visas knygas į reikiamas lentynas. Mes galime paspartinti šį procesą įtraukiant kitus darbuotojus, ir padalijant jiems po lygiai knygų. Bibliotekoje lentynos yra sugruopuotos į sekcijas ir į eiles, todėl toks knygų padalijimas bus neefektyvus, jeigu kiekvienas darbuotojas vaikščios po visas bibliotekos eiles. Alternatyvus būdas yra priskirti darbuotojams eiles. Kaip ir prieš tai, darbuotojams galima padalinti po lygiai knygų, bet jeigu jam tenka padėti knygą į ne savo eilės lentyną, jis ją atideda į atitinkamos eilės darbuotojo knygų eilę. Toks būdas yra gerokai efektyvesnis už pirmąjį.

Pavyzdys demonstruoja kaip užduotis gali būti įvykdyta greičiau padalijant ją į subužduotis atskiriems darbuotojams. Darbuotojai bendradarbiauja perduodami vieni kitiems ne jų lentynų knygas. Lygiagretūs skaičiavimai vyksta tais pačiais principais. Užduočių padalijimas darbuotojams priskiriant jiems knygų aibę vadinamas užduočių padalijimu, o knygų perdavimas kitiems darbuotojams yra vadinamas bendradarbiavimu tarp subužduočių.

Problemų sprendimo išlygiagretinimas gali būti skirtingų lygių. Kai kurių problemų sprendimas lygiagrečiai gali būti net neefektyvesnis už vykdymą nuosekliai. Tai ypač būdinga problemoms, kurių sprendimas apima tik vieną nedalijamą užduotį. Tokiu atveju papildomų darbuotojų priskyrimas nepagreitina problemos sprendimo. Viena problema ne lygi kitai, todėl ir problemų sprendimo išlygiagretinimas gali turėti įvairias išlygiagretinimo technikas, kurios parenkamos priklausomai nuo įvairių kriterijų.

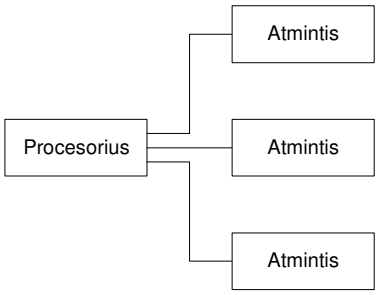
2.2.2. Aparatūra lygiagretiems skaičiavimams

Tradicinį nuoseklių kompiuterių modelį pirmas pristatė John von Neumann [15]. Modelį sudaro centrinis procesorius (*angl.* CPU – central processing unit) ir atmintis (*angl.* Memory).



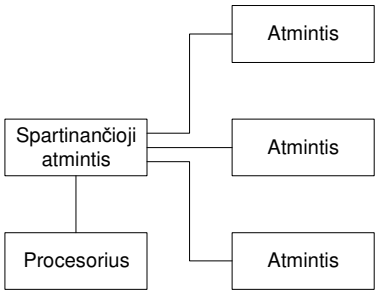
pav. 1. Paprastas nuoseklus kompiuteris

Šis skaičiavimo modelis vykdo vieną komandų seką ir naudojasi viena duomenų seka. Tokio modelio kompiuteriai dažnai vadinamas vieno komandų srauto ir vieno duomenų srauto (*angl.* SISD – single instruction stream, single data stream) kompiuteriais. SISD kompiuterių greitį apriboja du faktoriai: komandų vykdymo greitis ir duomenų mainų tarp procesoriaus ir atminties greitis. Pastarasis gali būti padidintas padidinant kanalų, kuriais vienu metu gali būti vykdomi mainai, kiekį. Tai padaroma padalijant atmintį.



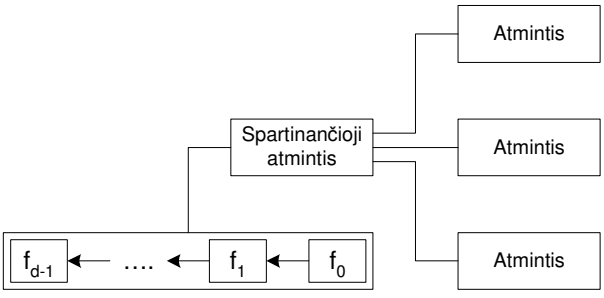
pav. 2. Nuoseklus kompiuteris su atminties padalijimu

Kita būdas duomenų mainams tarp procesoriaus ir atminties greičiui padidinti yra spartinančiosios atminties (*angl*. Cache memory) naudojimas. Santykinai maža ir labai greita atmintis naudojama kaip buferis santykinai lėtai ir didelei pagrindinei atminčiai.



pav. 3. Nuoseklus kompiuteris su atminties padalijimu ir spartinančiąja atmintimi

Komandų vykdymo greitis taip pat gali būti padidintas perdengiant komandos vykdymą su sekančios komandos paruošimu vykdymui. Kol procesorius yra užimtas aktyvios komandos vykdymu, sekanti komanda yra perskaitoma iš atminties ir pastatoma į komandų seką. Tokia technika vadinama komandų ruošimu (*angl*. Instruction pipelining). Panašioje, komandos sudalijimu (angl. Execution pipelining) vadinamoje technikoje, kelios komandos gali būti įvairiose vykdymo stadijose (pakopose).



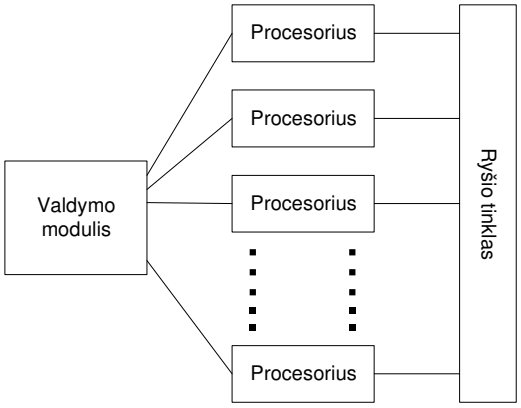
pav. 4. d pakopų turintis pakopinis procesorius

Atminties padalijimo, spartinančios atminties ir komandos sudalijimo technikos yra kartu naudojamos didelio našumo SISD kompiuteriuose, tačiau visi jie turi apribojimus. Atminties padalijimas ir komandos sudalijimas yra naudingas tik kai nedaug operacijų yra atliekama su

dideliais duomenų kiekiais. Spartinančioji atmintis pagreitina duomenų mainus, bet greitį vis tiek apriboja aparatūros technologija.

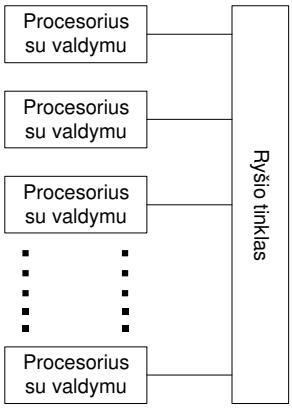
Alternatyvus būdas komandų vykdymo greičio padidinimui yra kelių procesorių naudojimas. Yra daug būdų kaip lygiagretūs kompiuteriai (*angl*. Parallel computers) gali būti sujungti. Visi būdai susiveda į 4 dimensijas: valdymo mechanizmo tipai, atminties tipai, ryšio tipai ir procesorių tipai.

Vykdymo moduliai lygiagrečiuose kompiuteriuose operuoja valdomi centralizuoto valdymo arba operuoja nepriklausomai. Vieno komandų srauto ir kelių duomenų srautų (*angl*. SIMD – single instruction stream, multiple data stream) architektūroje vienas valdantysis modulis siuntinėja komandas kiekvienam vykdančiajam moduliui.



pav. 5. Tipinė SIMD architektūra

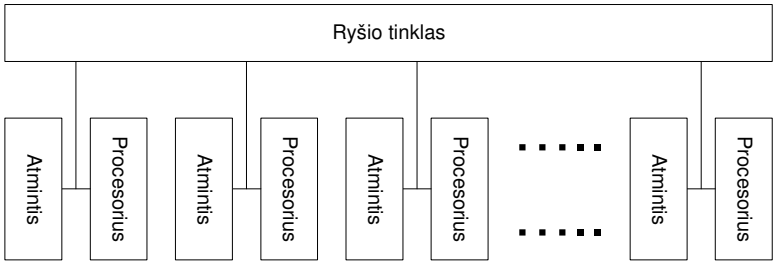
Tokiame kompiuteryje ta pati komanda yra sinchroniškai vykdoma kiekvieno procesoriaus. Vykdymo moduliai tokioje architektūroje gali būti išjungiami ir įjungiami skaičiavimų metu. Tarp lygiagrečių SIMD kompiuterių galime išskirti Illiac IV, MPP, DAP, CM-2, MasPar MP-1 ir MasPar MP-2. Kompiuteriai, kur kiekvienas procesorius gali vykdyti skirtingą programą (komandų seką) yra vadinami kelių komandų srautų ir kelių duomenų srautų (*angl*. MIMD – multiple instruction stream, multiple data stream) kompiuteriais.



pav. 6. Tipinė MIMD architektūra

Tarp lygiagrečių MIMD kompiuterių galime išskirti Cosmic Cube, nCUBE 2, iPSC, Symmetry, FX-8, FX-2800, TC-2000, CM-5, KSR-1 ir Paragon XP/S. SIMD kompiuteriai reikalauja mažiau aparatūros nei MIMD kompiuteriai, nes jie turi tik vieną valdymo modulį. Be to, SIMD kompiuteriai reikalauja mažiau atminties, kadangi tik viena programos kopija turi būti laikoma atmintyje. Priešingai SIMD, MIMD kompiuteriai laiko programą ir operacinę sistemą kiekviename procesoriuje. Tokiu atveju, SIMD kompiuteriai natūraliai pritaikomi duomenų lygiagretumo uždaviniams spręsti, kur tas pats kodas turi apdoroti didelius duomenų kiekius. Tuo tarpu MIMD kompiuteriai pritaikomi daug plačiau.

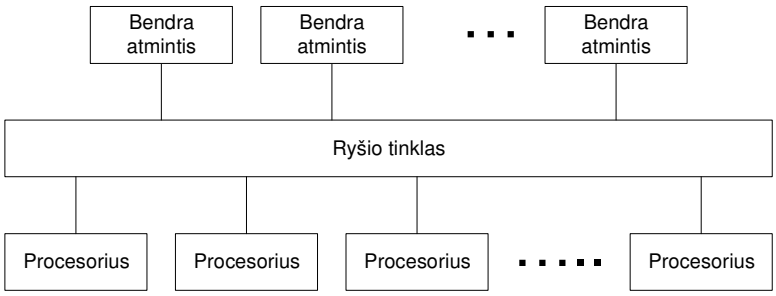
Kai skaičiavimai vykdomi skirtinguose procesoriuose, iškyla duomenų mainų problemos. Yra du pagrindiniai duomenų mainų tipai: pranešimų perdavimas ir bendra atmintis. Pranešimų perdavimo (*angl.* Message-passing) architektūroje procesoriai sujungti ryšio tinklu, kuriuo yra perduodami pranešimai. Kiekvienas procesorius turi savo atmintį (*angl*. Local/private memory), kuri yra pasiekiama tik to procesoriaus. Procesoriai gali sąveikauti tik perdavinėdami pranešimus. Tokia architektūra dar dažnai vadinama paskirstytos atminties (*angl*. Distributed memory) arba privačios atminties (*angl*. Private memory) architektūra.



pav. 7. Tipinė pranešimų perdavimo architektūra

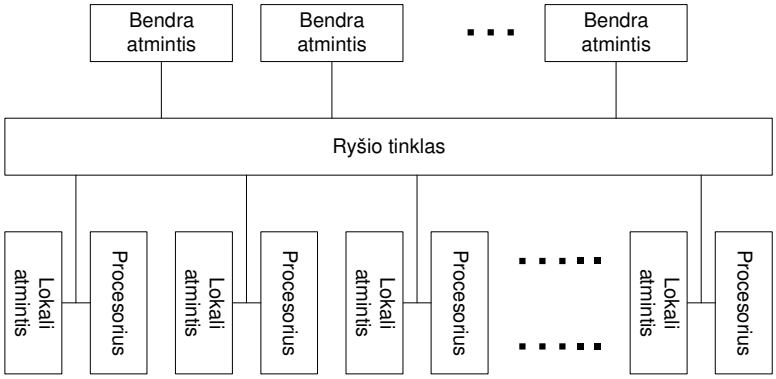
MIMD ir pranešimų perdavimo architektūros kompiuteriai bendrai vadinami multikompiuteriais, tarp kurių galima išskirti Cosmic Cube, Paragon XP/S, iPSC, CM-5 ir nCUBE 2. Bendros atminties (*angl*. Shared address space) architektūra suteikia aparatūrinę

skaitymo iš bendros atminties ir rašymo į bendrą atmintį galimybę iš kiekvieno procesoriaus. Procesoriai sąveikauja keisdami bendrus duomenis bendroje atmintyje. MIMD ir bendros atminties architektūros kompiuteriai dažnai vadinami multiprocesoriais. Ankstyvieji bendros atminties kompiuteriai turėjo bendrą atmintį, kurią pasiekdavo per ryšio tinklą.



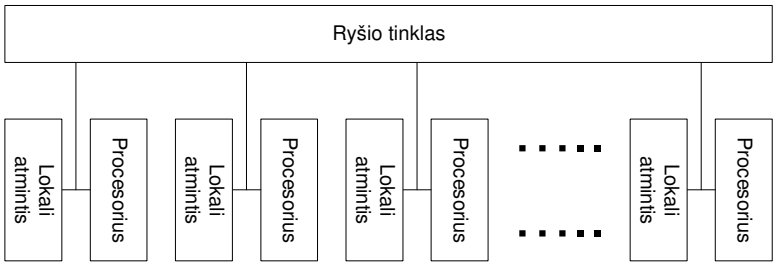
pav. 8. Pastovaus priėjimo bendros atminties architektūra

Tokios architektūros yra vadinamos bendros atminties lygiagretūs kompiuteriai, tarp kurių galima išskirti C.mmp ir NYU Untracomputer. Lėčiausia tokios sistemos vieta – ryšio tinklas. Tuo pačiu tai ir didžiausias tokios architektūros trūkumas. Vienas būdas trūkumams sumažinti yra papildyti kiekvieną modulį lokalia atmintimi, kur gali būti laikoma programa ir duomenys, kurių nereikia dalintis.



pav. 9. Nepastovaus priėjimo bendros atminties architektūra su lokalia ir globalia atmintimi

Lokalios atminties koncepcija gali būti išplėsta eliminuojant bendrą atmintį.



pav. 10. Nepastovaus priėjimo bendros atminties architektūra tik su lokalia atmintimi

Atminties dalijimasis tokiu atveju yra aparatūriškai organizuojamas per ryšio tinklą.

Bendros atminties kompiuteriai ir pranešimų perdavimo kompiuteriai gali būti sukonstruoti sujungiant procesorius ir atminties blokus naudojant įvairius ryšio tinklus. Ryšio tinklai klasifikuojami į statinius ir dinaminius. Statiniai tinklai susideda iš taškas-į-tašką (*angl.* Point-to-point) komunikacinius ryšius tarp procesorių ir dar kartais vadinami tiesioginiais tinklais (*angl.* Direct networks). Statiniai tinklai tipiškai naudojami konstruoti pranešimų perdavimo architektūra paremtiems kompiuteriams. Dinaminiai tinklai yra sudaryti naudojant komutatorius ir komunikacijos ryšius. Ryšys tarp dviejų procesorių ar atminties blokų yra užtikrinamas dinamiškai komutuojant atitinkamus komunikacijos ryšius. Dinaminiai tinklai priskiriami netiesioginių tinklų (*angl.* Indirect networks) klasei ir dažniausiai naudojami bendros atminties architektūra paremtiems kompiuteriams.

Lygiagretūs kompiuteriai gali būti sudaryti iš nedaug galingų procesorių arba daugybės paprastų procesorių. Pirmieji kompiuteriai yra vadinami stambiagrūdžiais (*angl.* Coarse-grain), o antrieji – smulkiagrūdžiais (*angl.* Fine-grain) kompiuteriais. Pavyzdžiui Cray Y-MP yra stambiagrūdis kompiuteris, kuris turi nuo 8 iki 16 procesorių, kurių kiekvienas yra kelių Gflops (1 Gflops = 10^9 floating-point operations per second) galingumo. O pavyzdžiui CM-2, MasPar MP-1 ar MasPar MP-2 yra smulkiagrūdžiai kompiuteriai, kurie turi didelį kiekį santykinai lėtų procesorių (CM-2 turi 65.536 vieno bito procesorius, o MasPar MP-1 turi 16.384 keturių bitų procesorius). Aišku, tarp šių kraštutinumų yra ir daug sistemų su tarpiniu "grūdų smulkumu", pavyzdžiui CM-5, nCUBE 2 ar Paragon XP/S, turintys iki kelių tūkstančių procesorių, kurių kiekvienas yra darbo vietos kompiuterio klasės pajėgumo. Skirtingiems "grūdų smulkumams" pritaikomi ir sprendžiami uždaviniai. Smulkiagrūdžiai kompiuteriai tinkamiausi aukštą išlygiagretinimo faktorių turintiems uždaviniams. Stambiagrūdžiai kompiuteriai tinkamiausi uždaviniams su apibrėžtu lygiagretumo lygiu. Be to, pasirenkant "grudų smulkumą", visada reikia priimti kompromisą tarp kainos ir efektyvumo.

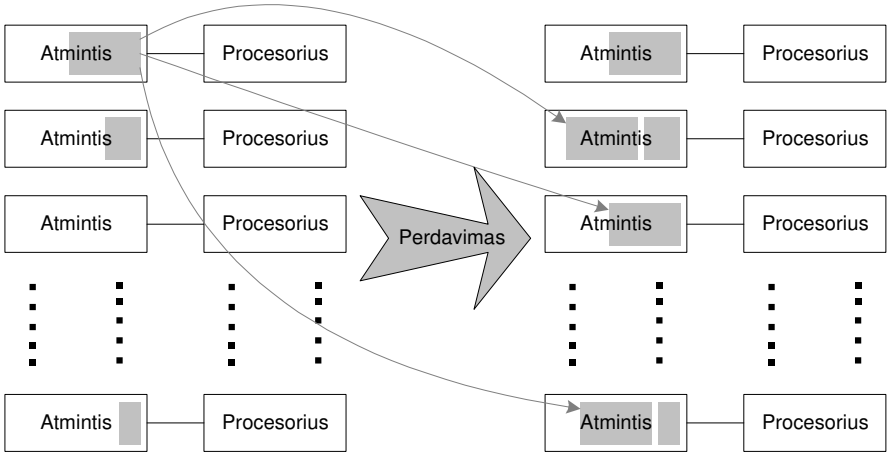
2.2.3. Komunikacijos operacijos

Daugumoje lygiagrečių skaičiavimų procesoriai turi dalintis duomenimis [19]. Tokie duomenų mainai akivaizdžiai paveikia lygiagrečių skaičiavimų efektyvumą įnešant komunikacijos užlaikymus. Yra du įprasti komunikacijos tarp procesorių modeliai, kurie naudojami kaip atspirties taškai lygiagrečių algoritmų projektavimui. Teisinga komunikacijos realizacija algoritmuose yra esminis efektyvumo užtikrinimo patvirtinimas.

Pats paprasčiausias komunikacijos tipas yra siųsti pranešimą iš vieno procesoriaus į kitą. Vieno pranešimo, kuris sudarytas iš m žodžių, siuntimas užtrunka $t_s+t_w ml$ laiko, kur t_s yra komunikacijos inicijavimo laikas, t_w – laikas vienam žodžiui perduoti, l – mazgų pranešimui perduoti kiekis, kuris pagrinde priklauso nuo ryšio tinklo architektūros, kuri atitinkamai gali būti

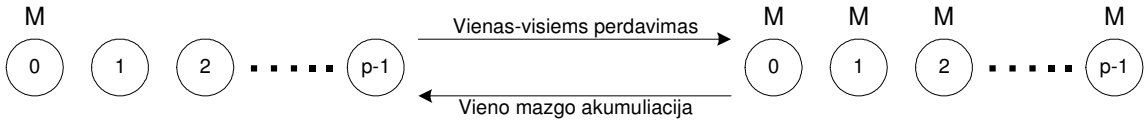
žiedinė, kilpinė arba kubinė. Žinodami ryšio tinklo architektūrą, galime nustatyti maksimalią l reikšmę. t_s priklauso nuo procesorių našumo charakteristikų, t_w – nuo ryšio tinklo našumo charakteristikų. Modeliuojant lygiagrečius algoritmus ir skaičiuojant komunikacijų trukmės apimtis belieka įstatyti visas reikšmes į formulę ir laikas apskaičiuotas.

Lygiagretūs algoritmai dažnai reikalauja, kad vienas procesorius pasiųstų tą pačią porciją duomenų keliems ar visiems procesoriams. Tokia operacija vadinama vienas-visiems perdavimas (*angl*. One-to-all broadcast or Single-node broadcast). Pradžioje tik inicijuojantis procesorius turi m dydžio duomenis, kuriuos turi perduoti. Po perdavimo operacijos jau kiekvienas procesorius turi tuos pačius duomenis (kiekvienas savo kopiją).



pav. 11. Vienas-visiems perdavimas

Lygiagretus algoritmas gali reikalauti, kad vienas procesorius sunktų informaciją iš visų kitų procesorių. Tokia operacija vadinama vieno mazgo akumuliacija (*angl*. Single-node accumulation) ir atvirkščias vienas-visiems perdavimui. Pradžioje kiekvienas procesorius turi m dydžio duomenis. Duomenys iš visų procesorių yra apjungiami jungiamuoju operatoriumi ir surenkami į rezultato procesorių. Operacija naudojama sumavimui, maksimumo ar minimumo radimui, loginėms bitų operacijoms atlikti.



pav. 12. Vienas-visiems perdavimas ir vieno mazgo akumuliacija

Visi-visiems perdavimas (*angl*. All-to-all broadcast or Multi-node broadcast) yra vienas-visiems perdavimo apibendrinimas, kai kiekvienas procesorius tuo pat metu inicijuoja perdavimą visiems kitiems procesoriams. Vienas procesorius siunčia m dydžio duomenis kiekvienam kitam procesoriui, bet kiti procesoriai gali siųsti visai kitus duomenis.

pav. 13. Visi-visiems perdavimas ir visų mazgų akumuliacija

Visi-visiems perdavimas yra naudojamas matricų ir vektorių daugyboje. Atrodytų, kad norint atlikti visi-visiems perdavimą, galima tiesiog atlikti *p* vienas-visiems peravimų, tačiau kai kuriose architektūrose tai truks būtent *p* kartų ilgiau.

2.2.4. Lygiagrečių sistemų našumas

Nuoseklių algoritmų našumas dažniausiai yra vertinamas pagal vykdymo laiką, išreikštą funkcija, kuri priklauso nuo įėjimų kiekio. Lygiagrečių algoritmų vykdymo laikas priklauso ne tik nuo įėjimų kiekio, bet ir nuo lygiagretaus kompiuterio architektūros bei procesorių kiekio [10]. Vadinasi, lygiagretus algoritmas negali būti vertinamas nežinant lygiagretumo architektūros. Lygiagreti sistema – tai lygiagrečios architektūros ir lygiagretaus algoritmo kombinacija. Kaip galima įvertinti tokios sistemos našumą? Kokie turi būti atlikti matavimai? Kokie matavimo vienetai? Šiuos klausimus detaliai nagrinėja Minesotos universiteto mokslininkai [8].

Yra keletas matavimų lygiagrečios sistemos našumui matuoti. Nuoseklios sistemos vykdymo laikas – tai laikas nuo programos pradžios iki pabaigos. Lygiagrečios sistemos vykdymo laikas – tai laikas nuo programos pradžios iki momento, kai paskutinis procesorius baigia darbą. T_S žymime nuoseklios, o T_P – lygiagrečios sistemos vykdymo laiką. Vertinant lygiagrečią sistemą, dažnai domimės, kiek našumo laimėta išlygiagretinant skaičiavimus lyginant su nuoseklia realizacija. Pagreitėjimas – tai matavimas, kuris rodo santykinę lygiagretumo naudojimo naudą. Pagreitėjimą žymime *S*, o apskaičiuojame kaip lygiagrečios sistemos vykdymo laiko ir nuoseklios sistemos vykdymo laiko santykį. Tik ideali lygiagreti sistema, turinti *p* procesorių gali turėti *p* dydžio pagreitėjimą. Praktikoje tai nėra įmanoma, nes gaištamas papildomas laikas duomenų mainams. Efektyvumas – tai matavimas, kuris parodo vidutinį laiką, kurį vienas procesorius yra užimtas. Efektyvumas apskaičiuojamas pagreitėjimą *S* padalinus iš procesorių kiekio *p*. Idealiu atveju, pagreitėjimas yra lygus *p*, todėl efektyvumas siekia 1. Praktikoje pagreitėjimas *S* yra mažesnis nei *p*, todėl efektyvumas, kurį žymime *E* yra tarp 0 ir 1. Apibrėžiame dar vieną matavimą – problemos lygiagretaus sprendimo kainą, kuri priklauso nuo vykdymo greičio T_P ir procesorių kiekio *p*. Kaina atspindi visų procesorių vykdymo laikų sumą. Tokiu būdu,

efektyvumas taip pat gali būti išreikštas greičiausio žinomo nuoseklaus algoritmo vykdymo laikų santykiui su lygiagrečios sistemos kaina.

Modeliuojant algoritmą ir pasirenkant architektūrą, jau reikia skaičiuoti lygiagrečios sistemos kainą ir bandyti ją optimizuoti. Darome prielaidą, kad reikia naudoti daug procesorių. Bent tiek, kiek yra įėjimų. Tačiau, kad duomenų mainų trukmė mažėtų, reikia duomenis skirstyti į kuo didesnius blokus. Tai atitinka grūdėtumo didinimą arba visų procesorių neišnaudojimą. Paprasčiausias būdas grūdėtumo didinimui yra modeliavimas tokio algoritmo, kur vienas įėjimas būtų skirtas vienam procesoriui, o naudoti vis tiek mažiau procesorių. Pavyzdžiui jeigu yra n įėjimų ir tik p procesorių ($p < n$), tai galime naudoti algoritmą, sukurtą n procesoriams, laikant juos virtualiais procesoriais. Tokiu būdu, kiekvienas fizinis procesorius simuliuoja n/p virtualaus procesoriaus. Kadangi procesorių kiekis sumažėja n/p faktoriumi, skaičiavimai kiekviename procesoriuje padidėja n/p faktoriumi. Tiek pat, bet ne daugiau, gali padidėti ir duomenų mainų laikas. Visas lygiagrečios sistemos vykdymo laikas padidėja ne daugiau, kaip n/p. Taigi, jeigu lygiagrečios sistemos su n procesorių kaina yra optimali, tai naudojant p procesorių n procesorių simuliacijai sistemos kainos optimalumo nekeičia. Šio naivaus metodo trūkumas – pradinė prielaida, kad pradinės lygiagrečios sistemos kaina yra optimali. Tokiu būdu, atlikus grūdėtumo didinimą, kaina nebūtinai taps optimalia.

Efektyviai naudojant šiuolaikinius galingus lygiagrečius kompiuterius, didesnės problemos gali būti išspręstos padidinus procesorių kiekį. Tačiau, kai problemos dydis yra fiksuotas, tikslas yra pasiekti kompromisą tarp efektyvumo ir vykdymo laiko.

2.2.5. Išvados

Daugybė knygų skirtingu detalumu nagrinėja įvairiausius lygiagrečių skaičiavimų aspektus. Mes iš jų susipažinome su lygiagrečių skaičiavimų kilme ir nauda. Detaliai išanalizavome naudotinus aparatūrų tipus, konstrukcijas, atminties, ryšių ir procesorių įtaką skaičiavimams. Supratome kokią įtaką lygiagretiems skaičiavimams turi duomenų mainų operacijos. Tokiu būdu detaliai išanalizavome duomenų mainų operacijų tipus, jų pritaikymą. Galop nagrinėjome lygiagrečių sistemų našumo veiksnius ir matavimo metodikas.

Analizės metu visada stengiamės apibrėžti vienos ar kitos metodikos naudojimo nišą ir tikslingumą. Taip susidarėme savo nuomonę apie mums tinkamiausią aparatūros kombinaciją, duomenų mainų preliminarią koncepciją, būsimų algoritmų našumo skaičiavimo techniką.

2.3. Darbo tikslai

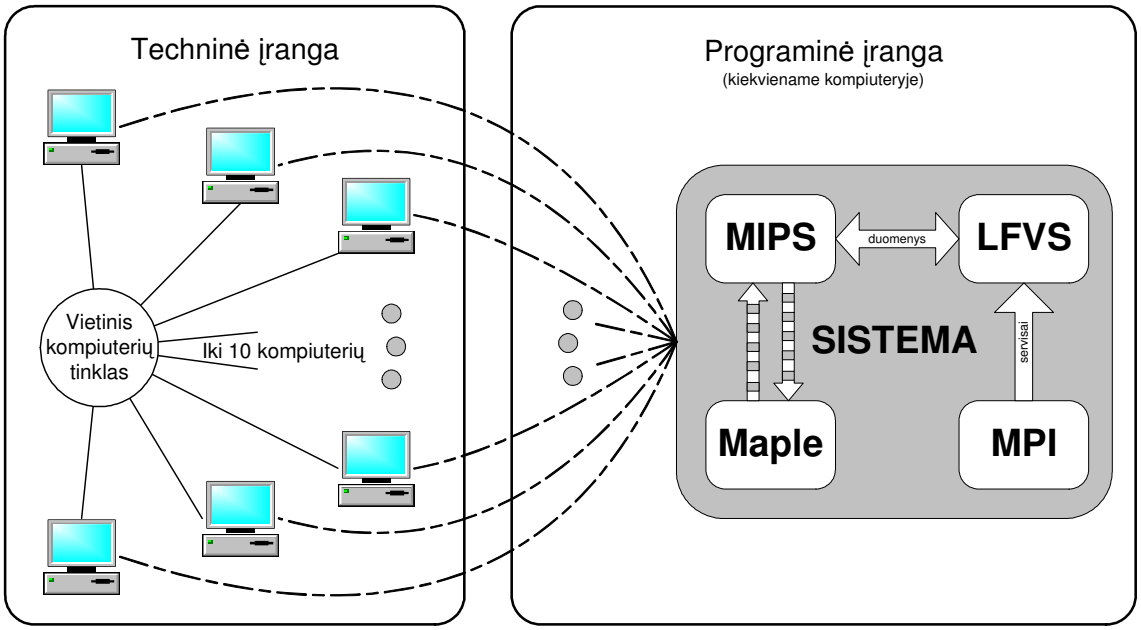
Identifikavus problemą, ją kruopščiai išanalizavus ir įsisavinus esamus sprendimus, sistemos kūrėjų komanda išsikėlė sau darbo tikslus, kurių dalis, susijusi su nagrinėjama problema, yra išvardijama žemiau:

- suprojektuoti detalią sistemos architektūrą išskiriant reikiamas valdymo ir objektų klases;
- suprojektuoti ir realizuoti matematinių išraiškų pertvarkymo išlygiagretinimo valdymo algoritmą;
- ištestuoti matematinių išraiškų pertvarkymo išlygiagretinimo valdymo algoritmo patikimumą;
- integruoti matematinių išraiškų pertvarkymo išlygiagretinimo valdymo algoritmą į sistemą;
- atlikti matematinių išraiškų pertvarkymo sistemos tyrimus, palyginti su kitomis sistemomis;
- įvertinti tyrimų rezultatų, įžvelgti tendencijas, padaryti išvadas, pateikti rekomendacijas.

2.4. Priimti sprendimai

Darbo tikslams pasiekti sistemos kūrėjų komanda pirmiausia turėjo priimti įvairius strateginius sprendimus, kurie numatytų sistemos veikimo principą, sistemos aplinką, programų inžinerijos metodikas, technologijas, programavimo kalbas ir pan.

2.4.1. Sistemos veikimo principas



pav. 14. Sistemos veikimo principas

Aptarsime aukščiau pavaizduotos sistemos veikimo principą. Pirmiausia, sistema turėjo mokėti tiesiog diferencijuoti matematinę išraišką. Tokios funkcijos yra paskirtos matematinių išraiškų pertvarkymo sistemai MIPS. Dar didesniam diferencijavimo greičiui užtikrinti buvo pasirinkta lygiagrečių skaičiavimų logika. Tokia logika realizuojama lygiagretaus funkcionavimo

valdymo sistemoje LFVS, kuri atsakinga už matematinės išraiškos suskaldymą į smulkesnes ir paprastesnes išraiškas, jų perdavimą atskiriems MIPS skaičiavimų procesams ir, galiausiai, apskaičiuotų išraiškų atgalinį sujungimą. Procesams lygiagretinti yra naudojamas MPI servisas. Efektyvumui užtikrinti reikalingi keli procesoriai. Visa minėta programinė įranga įdiegiama į kiekvieną naudojamą kompiuterį. Vartotojas bet kuriame kompiuteryje inicijuoja matematinės išraiškos diferencijavimą. LFVS suskaido išraišką į smulkesnes ir per MPI servisą inicijuoja atskirus procesus kituose kompiuteriuose. Procesuose veikianti MIPS pagal reikalavimus pertvarko matematinę išraišką ir grąžina iniciatoriui. Grąžinti rezultatai yra sujungiami ir išvedami vartotojui.

2.4.2. Lygiagretaus funkcionavimo valdymo sistema

Lygiagretaus funkcionavimo valdymo sistema priklausomai nuo proceso tipo gali veikti dvejopai. Procesai skirstomi į vieną pagrindinį, kuriame buvo inicijuotas matematinės išraiškos pertvarkymas, ir kitus.

Pagrindinis LFVS procesas veikia tokiais etapais:

- parametrų išplatinimas;
- dalijimas į subišraiškas;
- subišraiškų perdavimas atskiriems procesams;
- subišraiškų priėmimas;
- subišraiškų sujungimas;
- procesų sustabdymas.

Kiti LFVS procesai veikia tokiais etapais:

- parametrų įsisavinimas;
- išraiškų priėmimas;
- išraiškų diferencijavimas naudojant MIPS;
- rezultatinių išraiškų perdavimas pagrindiniam procesui.

Lygiagretaus funkcionavimo valdymo sistema užtikrina išlygiagretintą matematinės išraiškos diferencijavimą. Pagrindinis procesas yra laikomas valdančiu procesu, kuris atlieka tokius bendrus veiksmus, kaip parametrų nustatymas ir išplatinimas visiems procesams, matematinės išraiškos sudalijimas į subišraiškas, tų subišraiškų perdavimas atskiriems procesams, vėliau rezultatų surinkimas iš tų procesų ir apjungimas į vieną rezultatinę matematinę išraišką. Kiti procesai yra skirti tik matematinėms išraiškoms diferencijuoti. Kiekvienas iš jų nuolatos gauna matematines išraiškas, MIPS pagalba jas išdiferencijuoja ir siunčia atgal pagrindiniam procesui.

2.4.3. Naudojami produktai

Projekto metu dažnai buvo naudojamas Maple programinis paketas. Jis buvo pasirinktas dėl to, kad jį naudoja užsakovas. Prie šio produkto yra priderintos matematinės išraiškos vaizdavimo taisyklės. Be to, Maple paketas yra patikimas, greitas, lankstus, pritaikytas papildomų funkcijų integracijai, veikiantis Linux operacinėje sistemoje. Produkto gamintojas – Waterloo Maple, Inc.. Buvo naudojama Maple 7 versija.

Produktas efektyviai veikia naudodamasis MPI serviso paslaugomis. Konkrečiai lygiagretaus funkcionavimo valdymo sistema naudoja MPI servisą, kuris organizuoja procesų paskirstymą atskiriems procesoriams. Šis sprendimas pirmiausia buvo pasirinktas dėl to, kad jis pats tapo pranešimų perdavimo tarp kompiuterių standartu. Be to, paketas pritraukia savo efektyvumu ir paprastumu, yra pritaikytas Linux operacinei sistemai. Yra naudojama Local Area Multicomputer/Message Passing Interface (LAM/MPI) 7.0.3 versija.

3. PROJEKTAVIMAS

Šiame skyriuje išdėstomi visų produkto projektavimo etapų dokumentai: pirminis produkto apibūdinimas (projekto paraiška), reikalavimų specifikacija, architektūros specifikacija ir detalios architektūros specifikacija. Čia yra įkelti tik su magistro darbu susiję minėtų dokumentų skyriai. Taip pat šioje dalyje detalizuojama produkto programinė realizacija.

3.1. Produkto apibūdinimas

3.1.1. Programų sistemos funkcijos

Produkto esmė – matematinių išraiškų simbolinis diferencijavimas. Diferencijavimo rezultatas – nauja matematinė išraiška. Diferencijavimas yra riboto funkcionalumo, t. y. sistema turi atpažinti ir diferencijuoti tokias funkcijas:

- $f_1(x) + f_2(x)$
- $f_1(x) – f_2(x)$
- $f_1(x) \times f_2(x)$
- $(f(x))^n$
- ir jų junginius,

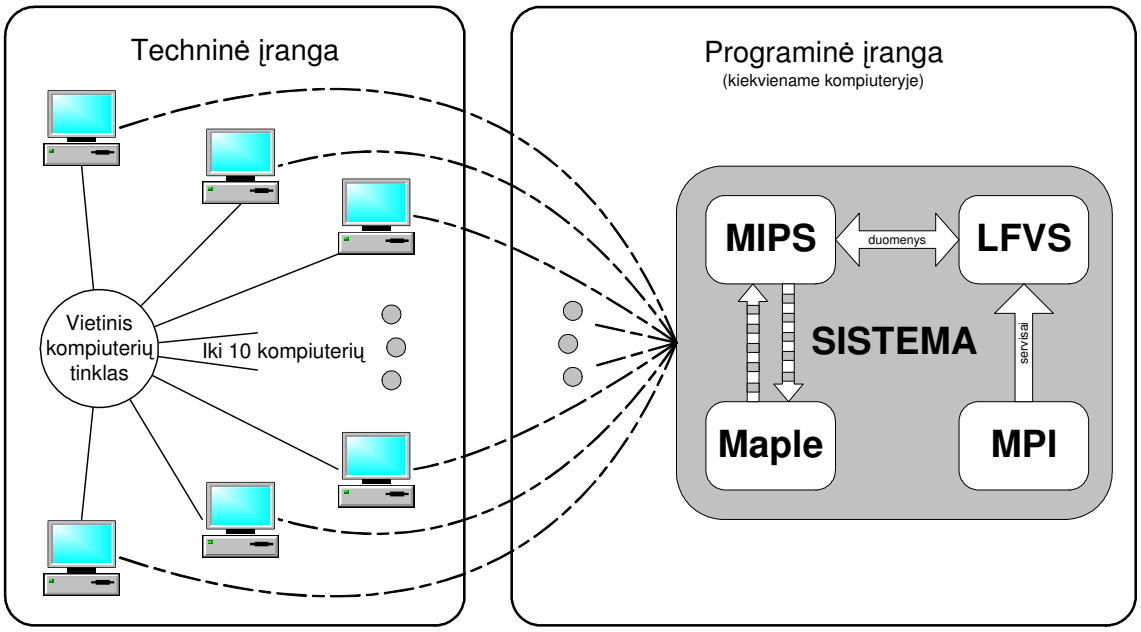
  kai f(x), $f_1(x)$, $f_2(x)$ gali būti:
- $a \cdot x^n + b$
- sin x
- cos x
- ir jų junginiai.

Kitos diferencijavimo formulės paliekamos kitoms produkto versijoms. Taip pat ir kitokie funkcionalumai (pvz., integravimas) paliekami kitoms produkto versijoms. Taigi sistema turi būti atvira funkcionalumų papildymams.

Atliekant matematinės išraiškos diferencijavimą, išraiška turi būti kuo optimaliau skaidoma į nepriklausomas išraiškas, kurias galėtų lygiagrečiai apdoroti atskiri procesoriai. Reikia įvertinti tokios paskirstytos sistemos apribojimus ir ypatybes:

- sistemos procesoriai neturi bendros atminties;
- duomenims perduoti gaištamas papildomas laikas.

Apibendrintas sistemos veikimo principas pavaizduotas pav. 15.



pav. 15. Sistemos veikimo principas

Sistema sudaryta iš dviejų posistemių:

- MIPS – matematinių išraiškų pertvarkymo sistema;
- LFVS – lygiagretaus funkcionavimo valdymo sistema.

Matematinių išraiškų pertvarkymo sistema atsakinga už matematinių išraiškų reikalavimuose nurodytų pertvarkymų atlikimą. Ši posistemė gali naudotis Maple paslaugomis. Lygiagretų matematinių išraiškų pertvarkymo vykdymą organizuoja lygiagretaus funkcionavimo valdymo sistema. Ši posistemė, naudodamasi paskirstytosios sistemos valdymo terpės MPI servisais, manipuliuoja vietiniame tinkle sujungtų kompiuterių procesoriais, t. y. naudoja juos lygiagretiems skaičiavimams.

Produktas turi turėti vartotojo sąsają, kuri:

- būtų patogi;
- būtų lengvai suprantama;
- nereikalautų papildomų programavimo žinių;
- mandagiai praneštų apie galinčius kilti nesklandumus.

3.1.2. Sistemos kontekstas

Projekte realizuojamas produktas yra integruojamas kartu su Maple ir MPI sistemomis. Produktas naudosis Maple sistema duomenims įvesti ir išvesti (lentelės pavidalu, grafiku) bei kai kurioms funkcijoms (pvz. matematinės išraiškos supaprastinimui) vykdyti. Todėl kuriamas produktas turi atitikti Maple sistemoje taikomus duomenų struktūrų apibrėžimus. Produktas naudos MPI sistemos servisus, lygiagrečiam matematinių išraiškų pertvarkymui valdyti. Todėl kuriamas produktas turi atitikti MPI sistemos teikiamų servisų reikalavimus.

3.1.3. Produkto vartotojas

Potencialūs produkto vartotojai – dabartiniai Maple sistemos vartotojai. Todėl produkto vartotojas gali neblogai žinoti Maple sistemą, bet gali nieko nežinoti apie MPI. Todėl sistema turi būti suprantama ir nežinantiems MPI.

3.1.4. Vartotojo problemos

Dabartinė vartotojo problema: nepatenkinamas didelių matematinių išraiškų diferencijavimo greitis. Sukurtame produkte turi būti pasiektas vartotoją tenkinantis matematinių išraiškų diferencijavimo greitis.

3.1.5. Vartotojo tikslai

Vartojimo požiūriu, kuriamas produktas neturi būti sudėtingesnis nei Maple sistema. Vartotojas neturi papildomai mokytis, kad galėtų inicijuoti matematinių išraiškų diferencijavimą.

Vartotojas naudodamas sukurtą produktą nori greičiau nei su Maple sistema diferencijuoti matematines išraiškas. Vartotojas negali sau leisti panaudoti efektyvias bet brangias daugiaprocesorines sistemas, tačiau gali sau leisti įsigyti kelis kompiuterius, sujungti juos į vietinį tinklą ir diferencijuoti matematines išraiškas naudodamas kelis paskirstytus procesorius.

3.1.6. Bendri apribojimai

Produktas turi tokius nefunkcinius reikalavimus:

- C++ programavimo kalba;
- Linux operacinė sistema;
- atvira funkcionalumų papildymams.

Projektas turi tokius organizacinius apribojimus:

- resursų trūkumas (techniniai resursai testavimui prieinami tik 2 val. per savaitę, kūrėjų grupės vadovas projektui gali skirti tik 2 val. per savaitę, kūrėjų grupės nariai – po 6 val. per savaitę);
- personalo patirties stoka.

3.2. Reikalavimų specifikacija

Reikalavimų specifikacija skirta sistemos reikalavimams tarp sistemos užsakovo ir sistemą realizuojančios komandos suderinti. Dokumentas parengtas pagal programų inžinerijos standartus ir pagal dėstytojų rekomendacijas.

3.2.1. Diegimo aplinka

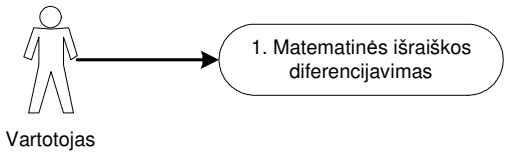
Sistema turi būti sukurta funkcionuoti PC tipo kompiuteriuose Linux operacinėje sistemoje. Sistemos greičiui užtikrinti yra reikalaujamas C++ kompiliatorius. Pilnaverčiam sistemos funkcionavimui reikalingas vietinis kompiuterinis tinklas (iki 10 kompiuterių), su įdiegtomis lygiagretaus valdymo MPI ir matematinio programinio paketo Maple sistemomis.

3.2.2. Bendradarbiaujančios sistemos

Sistema turi bendradarbiauti su matematiniu programiniu paketu Maple. Tiksli sąsajos specifikacija bus rengiama sistemos realizacijos etape. Šis paketas turi būti naudojamas kai kuriems standartiniams matematinių išraiškų pertvarkymams.

Sistema taip pat turi naudotis lygiagretaus funkcionavimo valdymo sistema MPI.

3.2.3. Veiklos kontekstas



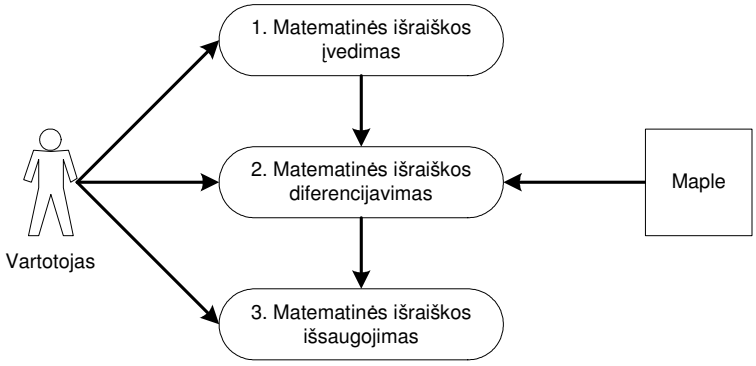
pav. 16. Veiklos konteksto diagrama

3.2.4. Veiklos padalijimas

lentelė 1. Veiklos padalijimas

Įvykio Nr.	1
Įvykis	**Matematinės išraiškos diferencijavimas**
Įeinantys informacijos srautai	Pradinė matematinė išraiška
Išeinantys informacijos srautai	Išdiferencijuota matematinė išraiška Diferencijavimui sugaištas laikas

3.2.5. Sistemos ribos



pav. 17. Panaudojimo atvejų diagrama

3.2.6. Panaudojimo atvejų sąrašas

lentelė 2. Matematinės išraiškos diferencijavimo panaudojimo atvejis

Panaudojimo atvejo Nr.	2
Panaudojimo atvejis	**Matematinės išraiškos diferencijavimas**
Tikslas	Išdiferencijuoti vartotojo įvestą matematinę išraišką
Aktoriai	Vartotojas, Maple
Ryšiai su kitais PA	1, 3
Nefunkciniai reikalavimai	Diferencijavimas turi būti greitesnis nei užsakovo anksčiau naudoti produktai
Pradinės sąlygos	Sistema yra aktyvi Matematinė išraiška yra įvesta Maple sistema yra pasiekiama

Sužadinimo sąlygos	Vartotojas sužadina matematinės išraiškos diferencijavimo funkciją
Pabaigos sąlygos	Ekrane parodoma išdiferencijuota matematinė išraiška arba jos dalis (kai išraiška yra labai ilga)
Pagrindinis scenarijus	Sistema integruota su Maple sistema išdiferencijuoja įvestą matematinę išraišką Sistema ekrane atvaizduoja išdiferencijuotą išraišką arba jos dalį (kai išraiška yra labai ilga) ir diferencijavimui sugaištą laiką
Alternatyvūs scenarijai	Matematinė išraiška neįvesta Nekorektiška matematinė išraiška Maple sistema yra nepasiekiama Kitais atvejais, sistema praneša "Nenumatyta klaida"

3.2.7. Funkciniai reikalavimai

lentelė 3. Funkcinis reikalavimas Nr. 1

Reikalavimo Nr.	1
Reikalavimo tipas	Funkcinis reikalavimas
Panaudojimo atvejis	1. Matematinės išraiškos diferencijavimas
Aprašymas	Sistema turi teisingai išdiferencijuoti vartotojo įvestą matematinę išraišką
Pagrindimas	Vartotojui reikalingas tik teisingas matematinės išraiškos išdiferencijavimas
Šaltinis	Vartotojas
Tikimo kriterijus	Įvesta matematinė išraiška yra teisingai išdiferencijuota
Užsakovo tenkinimas	3
Užsakovo netenkinimas	5
Priklausomybės	2, 3, 4, 5
Konfliktai	–
Papildoma medžiaga	–
Istorija	Reikalavimas užregistruotas 2003 03 14

3.2.8. Reikalavimai duomenims

Matematinė išraiška – tai simbolinė išraiška, galinti susidėti iš lentelėje Nr. nurodytų simbolių.

lentelė 4. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau pateiktoje lentelėje nurodytos funkcijos.

lentelė 5. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x ^ 2, x / 5

Matematinės išraiškos ilgis nėra apribotas. Ekrane yra vaizduojama visa matematinė išraiška jei jos ilgis neviršija 1024 simbolių. Priešingu atveju yra vaizduojami pirmi 1024 simboliai.

3.2.9. Reikalavimai vykdymo charakteristikoms

Diferencijavimo greitis – pagrindinis sistemos reikalavimas. Skaitinės šio reikalavimo vertės nėra numatytos, nes sistema kuriama tiriamojo darbo pagrindu. Yra numatyta, kad diferencijavimo laikas turi būti mažesnis nei iki šiol užsakovo naudojamų matematinių programiniu paketų.

3.3. Architektūros specifikacija

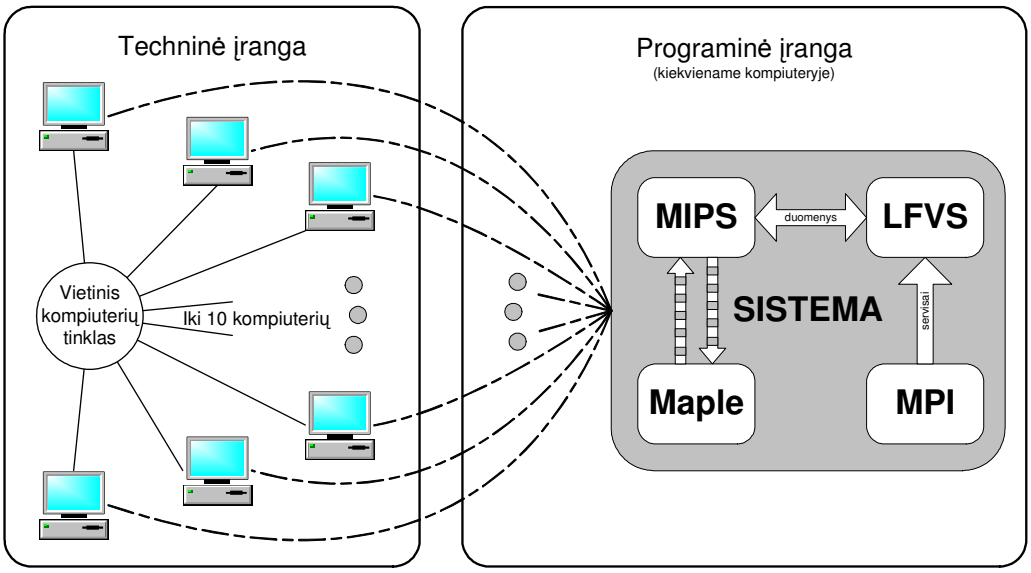
Architektūros specifikacija yra skirta sistemos architektūrai specifikuoti. Dokumentas yra parengtas Rational Unified Process (RUP) pagrindu naudojant Unified Modeling Language (UML).

Dokumente pateikiamas apibendrintas architektūros modelis, sistemos aplinka ir sistemos panaudojimo scenarijai. Vėliau apžvelgiami sistemos projektavimo kriterijai, detalizuojama procesų architektūra, veikimo aplinka, komponentų architektūra, pagrindinių sistemos nefunkcinių reikalavimų įvykdymo priemonės ir sprendimai.

Kuriamos programų sistemos architektūros parinkimas – svarbus projektinis sprendimas – atliekamas vadovaujantis analizės etape pateikta reikalavimų specifikacija.

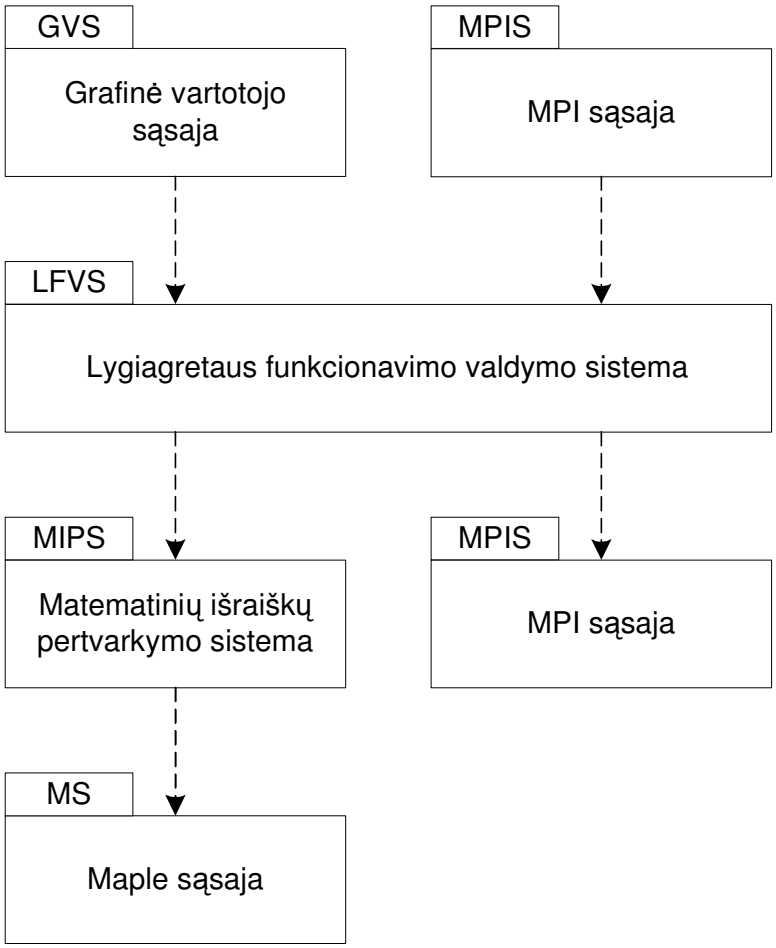
3.3.1. Apibendrintas architektūros modelis

Žemiau pateiktas apibendrintas sistemos veikimo principas.



pav. 18. Sistemos veikimo principas

Toks sistemos veikimo principas buvo numatytas dar projekto paraiškoje. Šis architektūros modelis yra tinkamas ir bus plėtojamas.

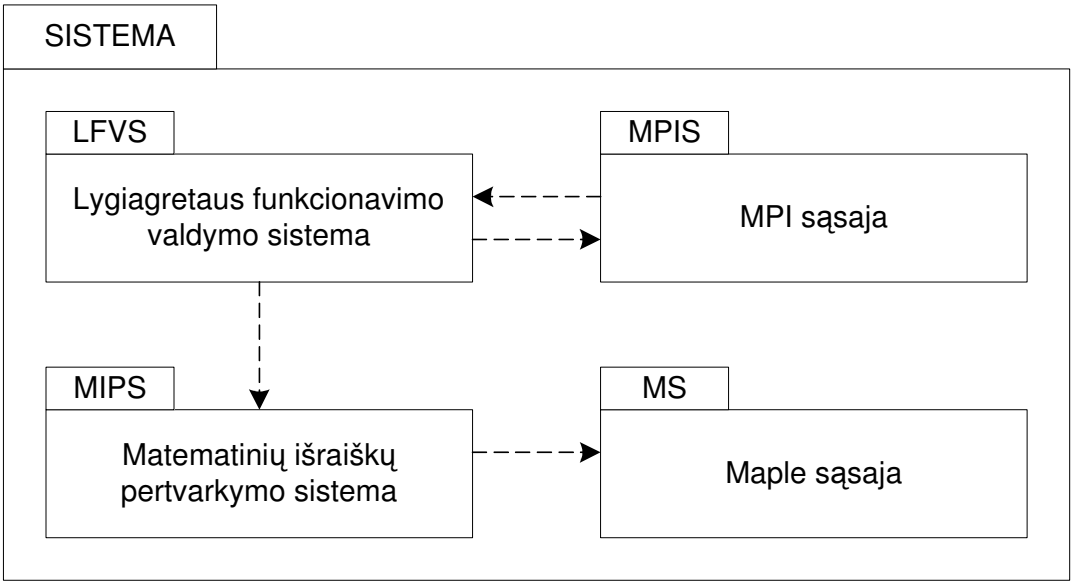


pav. 19. Architektūros modelis

Tokios architektūros sistema turi būti įdiegta kiekviename skaičiavimams naudojamame vietinio tinklo kompiuteryje. Vartotojo sąsaja yra naudojama pagrindiniame kompiuteryje, t. y. tame kompiuteryje, kuriame vartotojas inicijuoja skaičiavimus. Kituose kompiuteriuose skaičiavimus inicijuoja MPI servisas. Lygiagretaus funkcionavimo valdymo sistema yra atsakinga už matematinių išraiškų pertvarkymo veiksmų išlygiagretinimą į kelis procesus (jei tai yra naudinga). MPI sąsaja naudojama išlygiagretintiems procesams inicijuoti naudojant MPI servisą. Matematinių išraiškų pertvarkymo sistema geba pagal reikalavimus pertvarkyti reikalavimus atitinkančią matematinę išraišką. MIPS naudoja Maple sąsają, kad galėtų įvykdyti kai kurias standartines Maple funkcijas (pvz. reiškinio supaprastinimas).

3.3.2. Sistemos struktūra

Kuriama sistema yra paskirstyta ir yra projektuojama veikti vietiniame (lokaliame) kompiuterių tinkle, kai sistema yra atskirai įdiegiama kiekviename kompiuteryje.



pav. 20. Sistemos modulių dekompozicija

Kiekviena modulis aptariamas žemiau.

3.3.3. Lygiagretaus funkcionavimo valdymo sistema

Lygiagretaus funkcionavimo valdymo sistema (LFVS) yra atsakinga už matematinių išraiškų pertvarkymo veiksmų išlygiagretinimą į kelis procesus.

Modulio įėjimo duomenimis yra reikalavimus atitinkanti ir korektiška matematinė išraiška. Gautą išraišką LFVS analizuoja ir jei įmanoma ir jei naudinga suskaldo į nepriklausomas išraiškas. Jei išraiška nebuvo suskaldyta, tai ji tiesiog perduodama matematinių išraiškų pertvarkymo sistemai (MIPS), kuri, savo ruoštu, išdiferencijuoja išraišką bei grąžina rezultatą. Jei pradinė išraiška vis dėlto buvo suskaldyta į tam tikrą aibę išraiškų, tai kiekviena išraiška iš aibės

yra perduodamos atskiram MPI sąsajos egzemplioriui, kuris, savo ruoštu, kiekvienai išraiškai inicijuoja atskirą matematinės išraiškos diferencijavimo procesą. Kiekvienas MPI sąsajos egzempliorius geba pranešti apie baigtus pertvarkymus. Taigi, LFVS geba nustatyti visų išraiškų pertvarkymo pabaigą. Tada LFVS pagal tas pačias taisykles, pagal kurias suskaldė pradinę išraišką, apjungia rezultatus į vieną išraišką. Galutinė išraiška yra laikoma modulio išėjimo duomenimis (rezultatais).

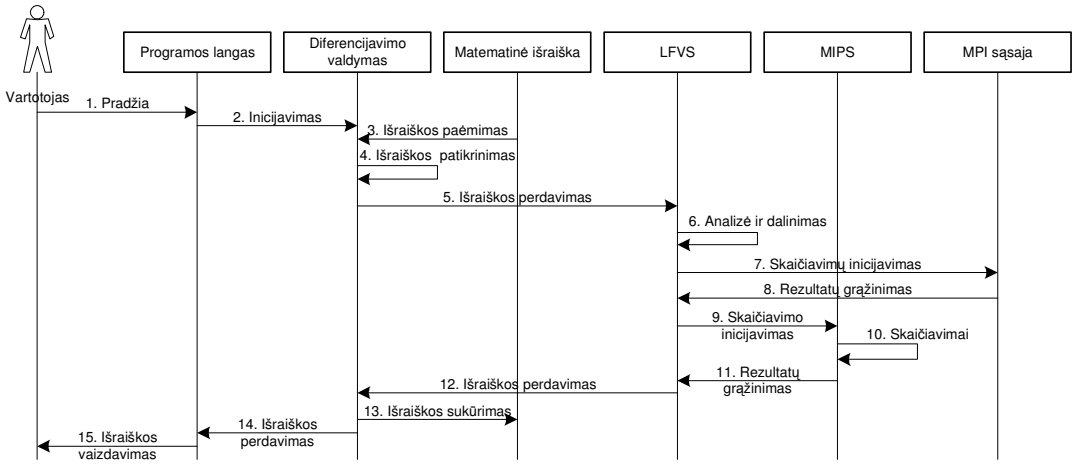
3.3.4. MPI sąsaja

MPI sąsaja (MPIS) yra atsakinga už matematinių išraiškų pertvarkymo inicijavimą kituose procesuose naudojant MPI servisą.

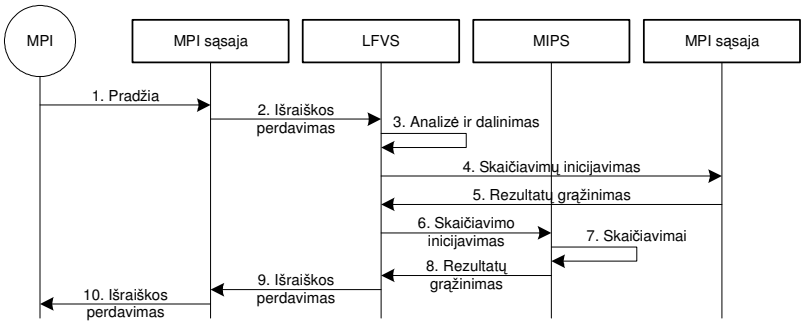
Modulio įėjimo duomenimis yra reikalavimus atitinkanti ir korektiška matematinė išraiška. Modulis perduoda matematinę išraišką MPI servisui ir inicijuoja jame naują procesą matematinės išraiškos pertvarkymui. MPI sąsaja lieka aktyvi ir sureaguoja tada, kai išskirtas procesas grąžina rezultatus, t. y. pertvarkytą matematinę išraišką. Ši išraiška yra laikoma modulio išėjimo duomenimis (rezultatais).

3.3.5. Panaudojimo atvejų sekų diagramos

Sekų diagramos naudojamos atvaizduoti sistemos objektų sąveikai, ryšiams tarp objektų ir pranešimams, kuriais keičiasi objektai.



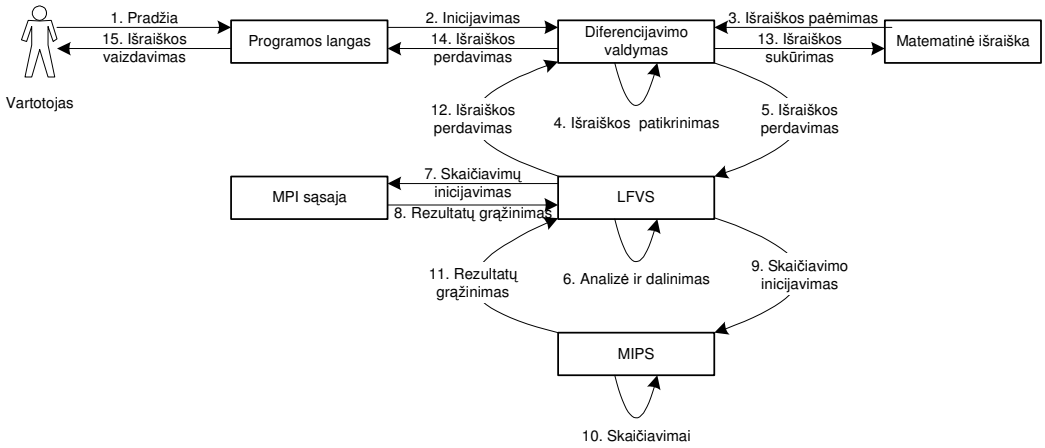
pav. 21. Matematinės išraiškos diferencijavimo (kai inicijuoja vartotojas) sekų diagrama



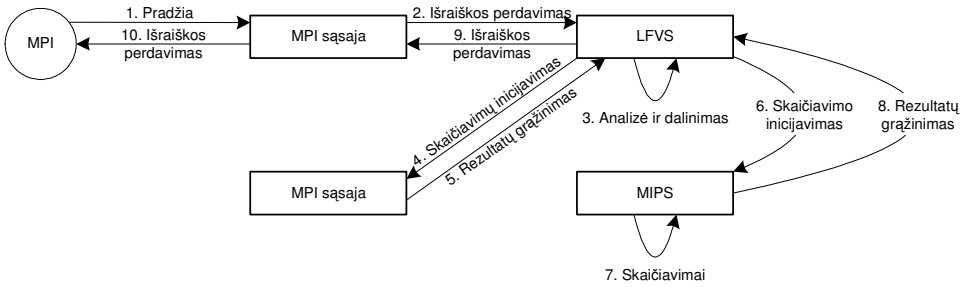
pav. 22. Matematinės išraiškos diferencijavimo (kai inicijuoja sistema) sekų diagrama

3.3.6. Objektų bendradarbiavimo diagramos

Objektų bendradarbiavimo diagramos vaizduoja didelį, bet detalų objektų sąveikos paveikslą, nes jos pateikia tiek objektų ryšius, tiek pranešimus.



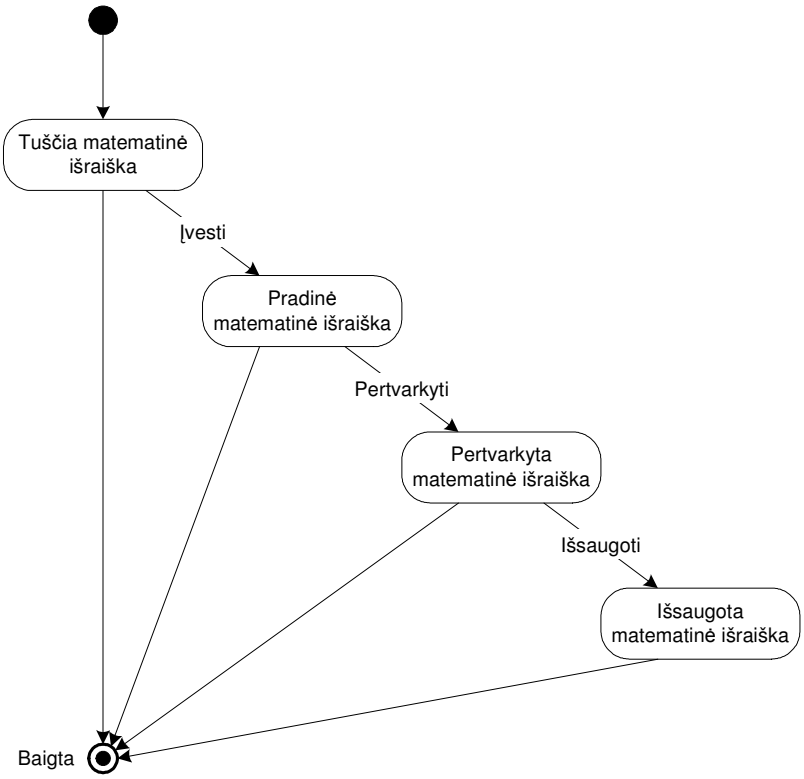
pav. 23. Matematinės išraiškos diferencijavimo (kai inicijuoja vartotojas) bendradarbiavimo diagrama



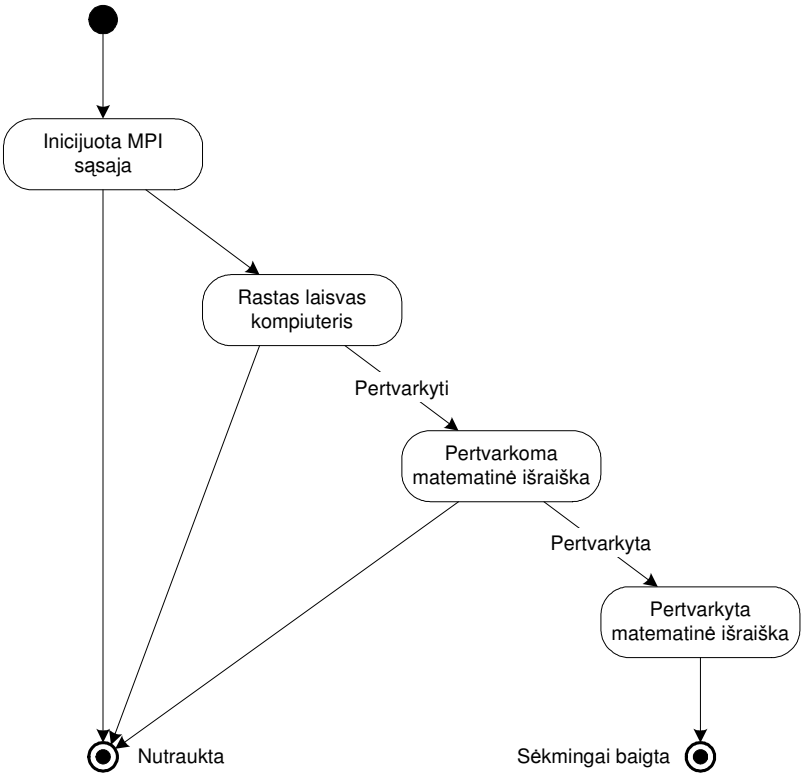
pav. 24. Matematinės išraiškos diferencijavimo (kai inicijuoja sistema) bendradarbiavimo diagrama

3.3.7. Objektų būsenų kaitos diagramos

Būsenų diagramos leidžia aprašyti modeliuojamų objektų elgesį. Paprastai jos naudojamos klasių elgesiui aprašyti.



pav. 25. Matematinės išraiškos būsenų kaitos diagrama

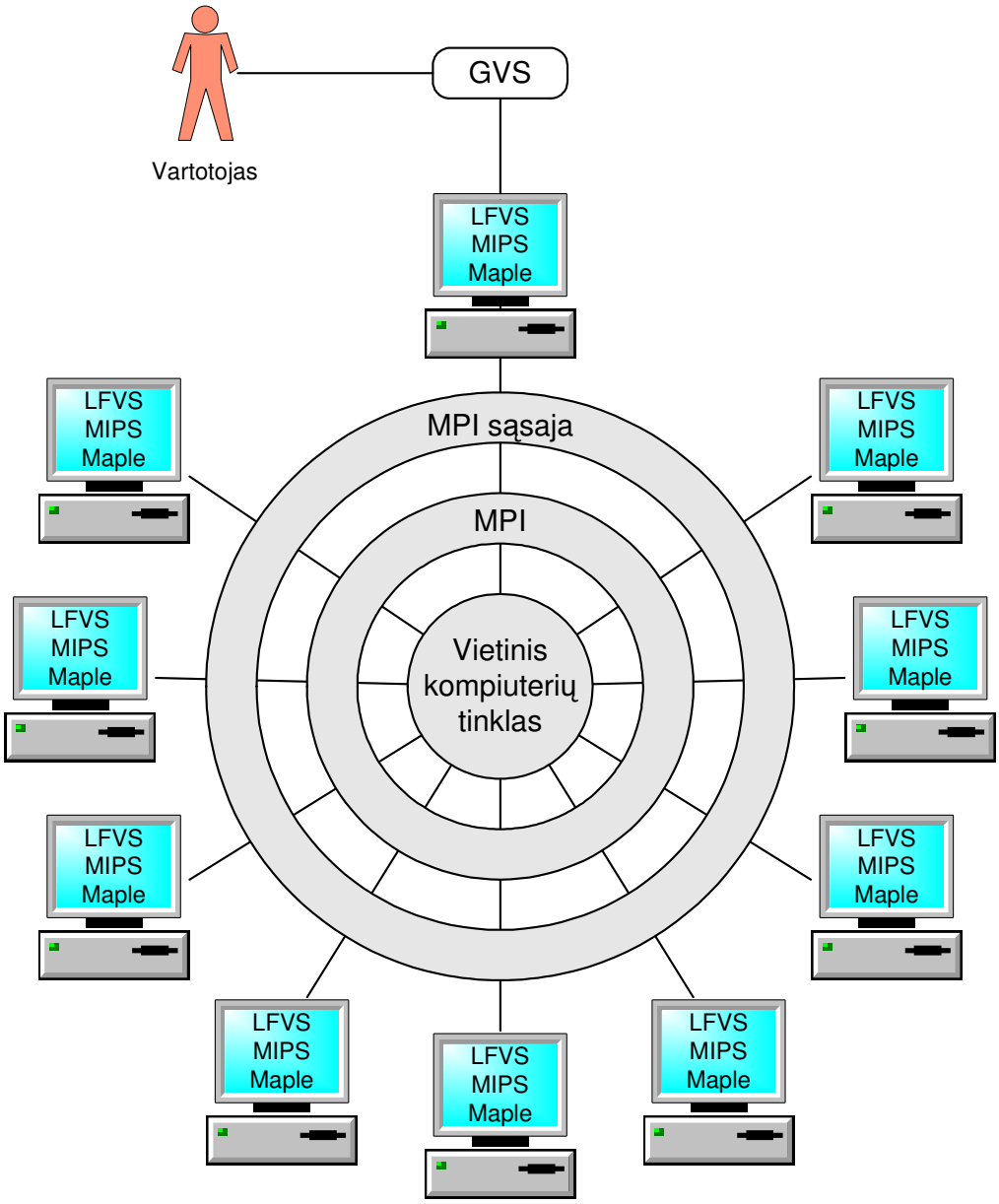


pav. 26. MPI sąsajos būsenų kaitos diagrama

3.3.8. Veikimo aplinka

Sistema gali funkcionuoti ir atskiroje darbo vietoje. Pilnam sistemos išnaudojimui reikalingi iki 10-ies kompiuterių, sujungtų į vietinį kompiuterių tinklą. Kiekviename kompiuteryje turi būti įdiegta Linux operacinė sistema, MPI servisas, Maple programinis paketas bei kuriama sistema. Tokios sistemos schema buvo pateikta produkto apibūdinimo skyrelyje.

Žemiau esančioje schemoje pavaizduotas architektūrinių komponentų išsidėstymas minėtoje veikimo aplinkoje.

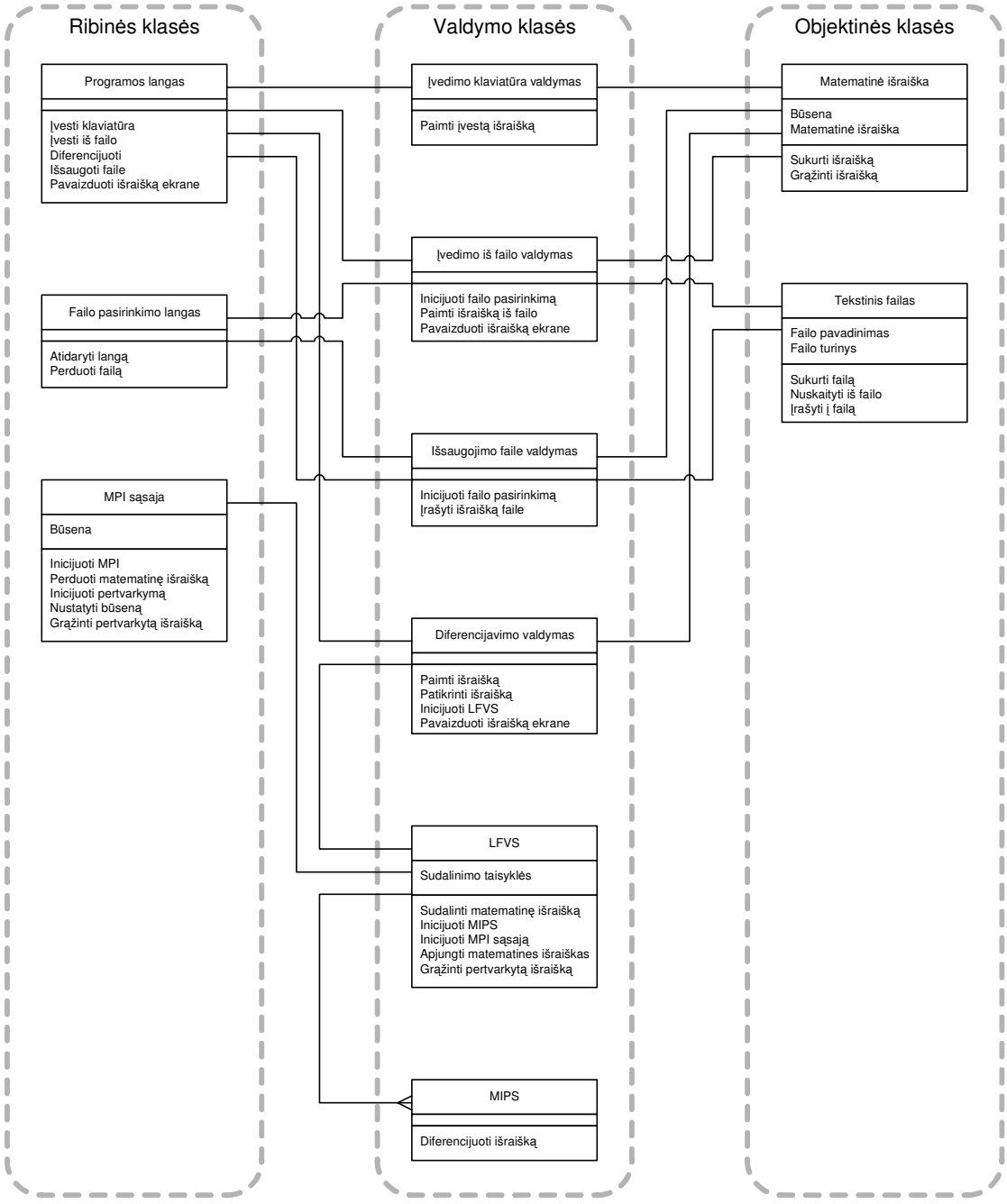


pav. 27. Architektūrinių komponentų išsidėstymas

3.3.9. Klasių diagrama

Komponentų architektūra – tai sistemos struktūra sudaryta iš susietų komponentų. Sistemos komponentų projektavimas apima komponentų identifikavimą ir jų sąryšių pavaizdavimą. Čia atspindimi statiniai sistemos aspektai.

Kaip pavaizduota pav. 20., sistema sudaryta iš keturių modulių: LFVS, MIPS, MPI sąsaja ir Maple sąsaja. Žemiau pateikiama klasių diagrama, kurioje matyti kiekvienos klasės pavadinimas, atributai ir operacijos bei ryšiai tarp klasių.



pav. 28. Klasių diagrama

3.3.10. Vykdymo charakteristikos

Pagrindinis sistemos reikalavimas – simbolinio diferencijavimo greitis. Dėl to turi būti priimti ir atitinkami architektūriniai sprendimai, kuriems esant būtų užtikrintas minimalus sprendimų laikas.

Jau analizės stadijoje buvo parinkta tinkamiausia veikimo aplinka (Linux operacinė sistema) ir tinkamiausias kompiliatorius (funkcionali bei greita C++ programavimo kalba). Greitis taip pat turi būti pagrindinis rodiklis modeliuojant ir realizuojant sąsajos, valdymo ir, svarbiausia, skaičiavimų algoritmus. Algoritmų modeliavimas atliekamas vėlyvosiose projektavimo ir ankstyvosiose realizacijos stadijose, dėl to šiame dokumente greičio uždavinio sprendimai dar neapibrėžti.

3.3.11. Kokybė

Sistemos architektūrai specifikuoti ir tolimesniems projektavimo darbams buvo nustatyti sistemos kokybės kriterijai ir jų prioritetai. Projektuojant ir parenkant sprendimus, bus pirmiausia atsižvelgiama į aukščiausio prioriteto kriterijų patenkinimą. Kriterijų ir prioritetų lentelė yra nustatyta pagal reikalavimų specifikacijoje nurodytus nefunkcinius reikalavimus.

lentelė 6. Sistemos kokybės kriterijai

Prioritetas / Kriterijus	l. aukštas	aukštas	vidutinis	žemas	l. žemas
Panaudojamumas				●	
Saugumas					●
Efektyvumas	●				
Teisingumas	●				
Patikimumas	●				
Palaikomumas		●			
Testuojamumas		●			
Lankstumas				●	
Suprantamumas			●		
Pakartotinis panaudojimas		●			
Pernešamumas				●	
Įsiliejimas			●		

Sistema skirta greitam konkrečių uždavinių sprendimui. Dėl to yra reikalaujamas didelis greitis, teisingumas, patikimumas, tačiau nereikalaujama aukšto panaudojamumo, lankstumo, suprantamumo. Dėl labai aukštų efektyvumo, teisingumo ir patikimumo kriterijų prioritetų, sistemai papildomai reikalingas aukšto prioriteto testuojamumo kriterijus, kad sistema būtų sėkmingai ištestuota. Reikalavimų specifikacijoje apibrėžta nedidelė sistemos pernešimo tikimybė, tačiau aukšta sistemos plėtojimo tikimybė, kas lemia aukštus palaikomumo ir pakartotinio panaudojimo kriterijų prioritetus.

3.4. Detalios architektūros specifikacija

Detalios architektūros specifikacija yra skirta sistemos komponentų detaliai architektūrai specifikuoti. Dokumentas parengtas pagal programų inžinerijos standartus ir pagal dėstytojų rekomendacijas. Komponentų projektavimas – vėlyva sistemos projektavimo proceso stadija. Dokumentas skirtas sistemos kūrėjams, kurie naudos dokumentą visose sistemos realizacijos proceso stadijose.

3.4.1. MPI sąsaja

- Klasifikacija

Pavadinimas	MPI sąsaja
Tipas	Ribinė klasė
Posistemė	MPI sąsaja
Versija	1.0, 2003.05.18

- Apibrėžimas

  Komponentas skirtas komunikavimo tarp dviejų procesų valdymui.

- Atsakomybės

  Komponentas geba dirbti dviem režimais: iniciatoriaus ir vykdytojo. Dirbant iniciatoriaus režimu, komponentas geba inicijuoti ir paleisti naują diferencijavimo procesą nurodytai matematinei išraiškai, taip pat nustatyti paleisto proceso būseną, o procesui pasibaigus, susigrąžinti rezultatus. Dirbant vykdytojo režimu, komponentas geba inicijuoti lygiagretaus funkcionavimo valdymo sistemą perduodant jai iniciatoriaus nurodytą matematinę išraišką, o pasibaigus diferencijavimui, atitinkamai pakeisti iniciatoriaus būseną.

- Apribojimai

  Komunikuoti tarp dviejų procesų yra naudojamas MPI servisas. Kad kiekvienas simbolinio diferencijavimo procesas būtų įvykdytas kuo greičiau, nenaudinga kiekvieno proceso pradžioje tikrinti matematinės išraiškos korektiškumo. Dėl to, bendram sistemos greičiui tarp procesų užtikrinti reikalinga perdavinėti tik korektiškas matematines išraiškas.

- Struktūra

  Į komponento sudėtį įeina du matematinės išraiškos egzemplioriai (vienas pradinei, kitas rezultatinei išraiškai), darbo režimo indikatorius bei būsenos žymeklis, kuris rodo proceso būseną (inicijuotas, pradėtas, baigtas).

- Sąveikavimas

  Komponentas yra naudojamas lygiagretaus funkcionavimo valdymo sistemoje, o norint inicijuoti ir paleisti naują diferencijavimo procesą kuriamas komponento egzempliorius darbui iniciatoriaus režime.

Taip pat komponentas ir pats naudoja lygiagretaus funkcionavimo valdymo sistemą, kai dirba vykdytojo režime.

- Resursai

  MPI servisas leidžia naudotis vietiniame tinkle sujungtų kompiuterių procesoriais. Manipuliavimu procesoriais rūpinasi pats MPI servisas.

- Skaičiavimai

  Komponento konstruktorius skirtas darbo režimui parinkti, MPI servisui inicijuoti. Dirbant iniciatoriaus režime, paleidimo metodas skirtas naujam diferencijavimo procesui paleisti, o dirbant vykdytojo režime, paleidimo metodas skirtas perduotiems duomenims nuskaityti ir diferencijavimui paleisti. Komponento destruktorius naudojamas tik vykdytojo režime ir yra skirtas rezultatams perduoti iniciatoriui.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kurio parametras yra darbo režimas. Dirbant iniciatoriaus režime, paleidimo metodui kaip parametrą reikia perduoti matematinės išraiškos egzempliorių, o dirbant vykdytojo režime, paleidimo metodas parametrų nereikalauja. Taip pat parametrų nereikalauja ir komponento destruktorius.

3.4.2. Diferencijavimo valdymas

- Klasifikacija

Pavadinimas	Diferencijavimo valdymas
Tipas	Valdymo klasė
Posistemė	Grafinė vartotojo sąsaja
Versija	1.0, 2003.05.17

- Apibrėžimas

  Komponentas skirtas matematinės išraiškos simboliniam diferencijavimui valdyti.

- Atsakomybės

  Komponentas atsakingas už matematinės išraiškos simbolinio diferencijavimo valdymą, kas apima matematinės išraiškos korektiškumo patikrinimo inicijavimą, lygiagretaus funkcionavimo valdymo sistemos sužadinimą bei rezultatinės matematinės išraiškos pavaizdavimą pagrindiniame programos lange.

- Apribojimai

  Apribojimų nėra.

- Struktūra

  Komponentas neturi viešų subkomponentų.

- Sąveikavimas

Komponentas yra naudojamas programos lango komponento egzemplioriaus, kuris atitinkamai sužadina šią valdymo klasę.

Komponentas savo ruožtu naudoja du matematinės išraiškos egzempliorius bei lygiagretaus funkcionavimo valdymo sistemos egzempliorių. Vienas matematinės išraiškos egzempliorius saugo pradinę matematinę išraišką, o kitas – rezultatinę matematinę išraišką. Lygiagretaus funkcionavimo valdymo sistemos komponento egzempliorius naudojamas lygiagrečiam simboliniam diferencijavimui.

- Resursai

  Komponentas naudojasi operacinės sistemos funkcijomis grafinei vartotojo sąsajai organizuoti.

- Skaičiavimai

  Komponentas turi vienintelį valdantį metodą, kuris inicijuoja matematinės išraiškos korektiškumo patikrinimą, paleidžia lygiagretaus funkcionavimo valdymo sistemą bei pavaizduoja rezultatinę matematinę išraišką pagrindiniame programos lange.

- Sąsaja/eksportas

  Valdantis metodas parametrų neturi.

3.4.3. Lygiagretaus funkcionavimo valdymo sistema

- Klasifikacija

Pavadinimas	Lygiagretaus funkcionavimo valdymo sistema
Tipas	Valdymo klasė
Posistemė	Lygiagretaus funkcionavimo valdymo sistema
Versija	1.0, 2003.05.17

- Apibrėžimas

  Komponentas skirtas lygiagretaus matematinės išraiškos simbolinio diferencijavimo funkcionavimui valdyti.

- Atsakomybės

  Komponentas atsakingas už perduotos matematinės išraiškos suskaidymą į smulkesnes ir paprastesnes matematines išraiškas, jų simbolinio diferencijavimo inicijavimą atskiruose procesuose bei atgalinį išraiškų apjungimą į vieną rezultatinę matematinę išraišką.

- Apribojimai

  Matematinė išraiškos nereikia skaidyti į smulkesnes ir paprastesnes, jei tam nėra pagrindo, t. y. jei matematinė išraiška ir taip nėra didelė ir/ar sudėtinga.

- Struktūra

  Komponentas neturi viešų subkomponentų.

- Sąveikavimas

  Kai matematinė išraiška dėl jos trivialumo nėra skaidoma, komponentas naudoja du matematinės išraiškos egzempliorius ir vieną matematinės išraiškos pertvarkymo sistemos egzempliorių. Vienas matematinės išraiškos egzempliorius skirtas pradinei matematinei išraiškai saugoti, o kitas – rezultatinei matematinei išraiškai saugoti. Matematinės išraiškos pertvarkymo sistemos egzemplioriaus pagalba matematinė išraiška yra simboliškai išdiferencijuojama.

  Vykdant matematinės išraiškos suskaidymą į smulkesnes ir paprastesnes matematines išraiškas yra naudojami net keli matematinės išraiškos egzemplioriai: vienas – pradinei matematinei išraiškai saugoti, visi kiti – gautoms smulkesnėms matematinėms išraiškoms saugoti. Tokiu atveju komponentas kiekvienai suskaidytai matematinei išraiškai taip pat naudoja po vieną MPI sąsajos egzempliorių, kurio pagalba yra inicijuojami ir paleidžiami atskiri diferencijavimo procesai. Kiekvienas MPI sąsajos egzempliorius, nustatęs kad procesas baigtas, grąžina dar po vieną matematinės išraiškos egzempliorių, kuriame saugoma atitinkamo proceso rezultatinė matematinė išraiška. Šioms išraiškoms apjungti atgaline tvarka yra naudojamas dar vienas, jau paskutinis, matematinės išraiškos egzempliorius.

  Pats komponentas yra naudojamas diferencijavimo valdymo komponente.

- Resursai

  Komponentui svarbiausias resursas yra procesorius.

- Skaičiavimai

  Komponento konstruktorius skirtas komponentui inicijuoti. Valdantis metodas skirtas matematinei išraiškai suskaidyti į smulkesnes ir paprastesnes matematines išraiškas, jų simboliniam diferencijavimui inicijuoti atskiruose procesuose bei gautoms išraiškoms sujungti į vieną rezultatinę matematinę išraišką.

  Lygiagretaus funkcionavimo valdymo algoritmas dar nėra numatytas. Dėl savo sudėtingumo algoritmas bus realizuojamas evoliuciniu principu.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kurio parametras yra matematinė išraiška. Valdantis metodas parametrų neturi, tačiau grąžina rezultatinę matematinę išraišką.

3.4.4. Matematinė išraiška

- Klasifikacija

Pavadinimas	Matematinė išraiška
Tipas	Objektinė klasė
Posistemė	-
Versija	1.0, 2003.05.15

- Apibrėžimas

  Komponentas skirtas matematinei išraiškai saugoti, korektiškumui tikrinti. Realizacijos procese klasė gali būti papildoma kitais funkcionalumais, pvz. nedideliems pertvarkymams vykdyti, supaprastinti ir pan.

- Atsakomybės

  Komponentas atsakingas už vienos matematinės išraiškos saugojimą. Taip pat komponentas geba pasitvarkyti išraišką, t. y. pašalinti ir taip ignoruojamus simbolius, bei patikrinti išraiškos korektiškumą.

- Apribojimai

  Veiksmai su komponentu yra neleidžiami, kol matematinei išraiškai nėra patikrintas korektiškumas. Jei matematinė išraiška nekorektiška, veiksmai ir toliau draudžiami.

  Matematinės išraiškos formatas buvo numatytas dar reikalavimų specifikacijoje. Išraiškoje gali būti sakančioje lentelėje nurodyti simboliai.

lentelė 7. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau esančioje lentelėje nurodytos funkcijos.

lentelė 8. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x^2, x / 5

Matematinės išraiškos ilgis nėra apribotas.

- Struktūra

Pagrindinė komponento dalis – matematinė išraiška. Kad išraiškos dydis nebūtų labai apribotas, jai saugoti išskiriama vieta yra dinaminėje atminties dalyje. Taip pat komponentas turi išraiškos korektiškumo požymį.

- Sąveikavimas

  Komponento egzempliorius yra naudojamas visoje sistemoje. Komponento egzemplioriaus naudojimas yra prasmingas tik esant korektiškai matematinei išraiškai.

- Resursai

  Matematinė išraiška yra saugoma dinaminėje operatyvinės atminties dalyje. Ši atmintis, priklausomai nuo naudojamo kompiuterio, yra ribota. Tačiau, pagal reikalavimų specifikaciją, matematinės išraiškos ilgis nėra apribotas, todėl naudojant komponentą, būtina, kad operatyvinėje atmintyje būtų pakankamai laisvos vietos.

- Skaičiavimai

  Komponento konstruktorius skirtas komponentui inicijuoti. Papildymo metodas geba papildyti matematinę išraišką atitinkamai išskirdamas papildomos atminties. Korektiškumo patikrinimo metodas geba patikrinti matematinės išraiškos korektiškumą. Komponento destruktorius skirtas išskirtai atminčiai atlaisvinti.

  Matematinės išraiškos korektiškumo tikrinimas vykdomas rekurentiškai tikrinant tarp skliaustų esančią matematinę išraišką. Matematinė išraiška, kurioje nėra skliaustų, tikrinama kaip išraiškos elementų (kintamųjų, konstantų, operacijų) seka, žinant kokios elementų poros gali būti viena paskui kitą ir kokios negali.

  Komponentas turi savo išimčių gaudyklę, t. y. įvykus klaidai, geba tvarkingai nutraukti tolimesnius veiksmus ir apie tai informuoti vartotoją. Specialiai yra apdorojamos darbo su operatyvine atmintimi klaidos.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kuris neturi parametrų. Papildymo metodas kaip parametro reikalauja simbolių eilutės, kuria yra papildoma matematinė išraiška. Korektiškumo patikrinimo metodas bei destruktorius parametrų neturi.

3.4.5. Tekstinis failas

- Klasifikacija

Pavadinimas	Tekstinis failas
Tipas	Objektinė klasė
Posistemė	Grafinė vartotojo sąsaja
Versija	1.0, 2003.05.15

- Apibrėžimas

Komponentas skirtas matematinių išraiškų skaitymui iš tekstinio failo ir matematinių išraiškų rašymui į tekstinį failą organizuoti.

- Atsakomybės

  Komponentas gali nuskaityti iš tekstinio failo jame esančią matematinę išraišką bei įrašyti į tekstinį failą nurodytą matematinę išraišką.

- Apribojimai

  Komponentas gali skaityti tik iš egzistuojančio tekstinio failo. Atliekant įrašymą turi būti pakankamai atminties naudojamame pastovios atminties kaupiklyje.

  Nuskaitymas yra įmanomas tik tokiu atveju jei faile įrašyta matematinė išraiška yra korektiška, t. y. atitinka Maple sistemos duomenų formatą. Būtent tokį formatą naudoja ir įrašymo į tekstinį failą metodas. Reikalavimai matematinės išraiškos duomenų formatui yra aprašyti matematinės išraiškos komponento apribojimų skiltyje.

- Struktūra

  Komponentą sudaro tekstinio failo pavadinimas ir rodyklė į failą. Pavadinimas skirtas failo identifikacijai, o rodyklę į failą naudoja sisteminės failų apdorojimo procedūros.

- Sąveikavimas

  Komponentas naudoja matematinės išraiškos komponentą. Šio komponento egzemplioriui yra priskiriama nuskaityta matematinė išraiška arba atvirkščiai, t. y. šio komponento egzemplioriuje esanti matematinė išraiška yra įrašoma į tekstinį failą.

  Komponentas yra naudojamas grafinės vartotojo sąsajos posistemėje. Komponento egzempliorius yra kuriamas matematinės išraiškos įvedimo iš tekstinio failo valdymo klasės komponente ir matematinės išraiškos išsaugojimo tekstiniame faile valdymo klasės komponente.

- Resursai

  Komponentas glaudžiai susijęs su naudojamu pastovios atminties kaupikliu. Naudojant įrašymo metodą, būtina, kad būtų pakankamai laisvos atminties. Kai atminties nėra pakankamai įrašymas neįvyksta, o metodas atitinkamai praneša apie klaidą. Be to, tekstinis failas, į kurį yra įrašinėjama matematinė išraiška, negali būti užrakintas operacinės sistemos, t. y. turi būti laisvai prieinamas įrašymui. Naudojant skaitymo iš tekstinio failo metodą, naudojamas tekstinis failas turi egzistuoti naudojamame atminties kaupiklyje.

- Skaičiavimai

  Komponento konstruktorius geba susirasti ir susirišti su nurodytame pastovios atminties kaupiklyje esančiu nurodytu tekstiniu failu. Nuskaitymo metodas geba nuskaityti iš tekstinio

failo matematinę išraišką, o įrašymo metodas geba įrašyti matematinę išraišką į tekstinį failą. Komponento destruktorius geba išsaugoti failą (jei vyko įrašymas) ir jį uždaryti.

Komponentas turi savo išimčių gaudyklę, t. y. įvykus klaidai, geba tvarkingai nutraukti tolimesnius veiksmus ir apie tai informuoti vartotoją. Specialiai yra apdorojamos darbo su failais klaidos.

- Sąsaja/eksportas

  Komponentui inicijuoti reikalinga iškviesti konstruktorių, kurio parametras yra tekstinio failo vardas ir matematinė išraiška. Nuskaitymo ir įrašymo metodai parametrų neturi. Taip pat ir destruktorius neturi parametrų.

3.5. Programinė realizacija

Analizės dalyje aprašytiems sprendimams įgyvendinti reikėjo detaliai suprojektuoti ir realizuoti atitinkamus algoritmus bei tinkamas duomenų struktūras. Projektavimo etape nebuvo atliekami algoritmų projektavimai, nes dar projekto plane buvo numatyta, kad algoritmai bus vystomi evoliuciniu būdu programavimo stadijoje. Šiame skyriuje išdėstomi ir pagrindžiami suprojektuoti algoritmai, detalizuojamos naudojamos duomenų struktūros.

3.5.1. Matematinė išraiška

Pagrindinis sistemos objektas – matematinė išraiška – tai simbolinė eilutė, matematikoje reiškianti reiškinį, kuris gali būti sudarytas iš sveikųjų ir realiųjų, teigiamų ir neigiamų skaičių, konstantų, kintamųjų, aritmetinių operandų (neigimas, suma, atimtis, daugyba, dalyba, kėlimas laipsniu, šaknies traukimas) bei įvairių funkcijų (trigonometrinių, logaritminių ir pan.). Matematinės išraiškos ilgis nėra apribotas. Matematinė išraiška gali susidėti iš žemiau esančioje lentelėje nurodytų simbolių.

lentelė 9. Leistini matematinės išraiškos simboliai

Simbolis	Apibūdinimas	Naudojimas	Pavyzdžiai
0-9	Visi skaitmenys	Skaičiai, indeksai	805, t2
,	Kablelis	Trupmenoms	2,05
.	Taškas	Trupmenoms	0,92
A-Z	Didžiosios lotyniškos raidės	Funkcijos, nežinomieji	PI, X, Y
a-z	Mažosios lotyniškos raidės	Funkcijos, nežinomieji	sin, a, h
+	Pliusas	Sudėtis	a + 5, t + t1
–	Minusas	Atimtis, neigimas	h – 1, -3
*	Žvaigždutė	Sandauga	7 * h, 2 * PI
/	Pavirtęs brūkšniukas	Dalyba	1 / 2, PI / 2
( )	Skliausteliai	Operacijų tvarka	2 * (b + 3)
^	Stogelis	Laipsnis	c^2, x^e

Matematinėse išraiškose gali būti naudojamos žemiau pateiktoje lentelėje nurodytos funkcijos.

lentelė 10. Matematinėse išraiškose leistinos funkcijos

Junginys	Apibūdinimas	Pavyzdžiai
sin, SIN	Trigonometrinė funkcija sinusas	sin(x), sin(pi)
cos, COS	Trigonometrinė funkcija kosinusas	cos(a – b), cos(1)
x, X	Nežinomasis x	x ^ 2, x / 5

Žemiau pateikti keli matematinių išraiškų pavyzdžiai:

- `3 * x ^ 4 – 5 * x ^ 3 + 7 * x ^ 2 – 9 * x + 11`
- `(5 * x – 9,2) ^ 3 * (7 * x + 4) ^ 2`
- `2 * sin(x) – cos(2 * x)`

- `(6 * x * sin(5 * x ^ 2 - x)) ^ 3,5`
- `(sin(3 * x)) ^ 2 * (-cos(x ^ 1/2)) * (4 * x ^ 1/3) ^ 4`

Produktas pritaikytas labai didelių (ilgų) matematinių išraiškų pertvarkymams. Labai didelės matematinės išraiškos gali būti sudarytos pavyzdžiui iš 5 milijonų simbolių.

Matematinėms išraiškoms saugoti yra naudojama simbolinės eilutės duomenų struktūra, atitinkanti matematinį reiškinį, kurių pavyzdžiai pateikti aukščiau.

3.5.2. Lygiagretaus funkcionavimo valdymas

Lygiagretaus funkcionavimo valdymo sistema (LFVS) yra atsakinga už lygiagrečių procesų valdymą ir duomenų sinchronizavimą. LFVS turi du atskirus modulius skirtingiems procesams valdyti: valdantysis ir vykdantysis.

Pagrindinis LFVS modulis yra valdantysis. Šis modulis visada veikia tik pagrindiniame procese (*angl*. Master process), nepriklausomai nuo to, kiek iš viso procesų sistemoje. Pagrindinis procesas nusistato pagal tai, iš kur naudotojas inicijuoja matematinių išraiškų pertvarkymo sistemą. Valdantysis LFVS modulis veikia trimis pagrindiniais etapais:

- Sistemos inicijavimas. Suskaičiuojami procesai, paskirstomi prioritetai, nuskaitomi sistemos parametrai bei pertvarkymui skirta matematinė išraiška, sistemos parametrai persiunčiami visiems kitiems procesams.
- Matematinės išraiškos pertvarkymo valdymas. Matematinė išraiška dalijama į subišraiškas, šios yra perduodamos atskiriems procesams, laukiami rezultatai, rezultatai apjungiami į vieną.
- Sistemos sustabdymas. Rezultatų įrašymas, procesų sustabdymas, atminties atlaisvinimas, išėjimas iš sistemos.

Ties pirmuoju ir trečiuoju etapu neapsistosime. Šiek tiek žemiau, nagrinėdami LFVS veikimą, apsistosime ties pagrindiniu – matematinių išraiškų pertvarkymo valdymo – etapu.

Kituose lygiagretaus funkcionavimo valdymo sistemos procesuose (*angl*. Slave process) veikia vykdantysis LFVS modulis, kurį inicijuoja jau ne vartotojas, o valdantysis LFVS modulis, veikiantis pagrindiniame procese. Sistemai startavus, vykdantysis modulis ima laukti darbo. Pirmiausia modulis laukia perduodamų sistemos parametrų. Toliau vykdantysis lygiagretaus funkcionavimo valdymo sistemos modulis veikia periodiškai. Vienas modulio periodas susideda iš trijų etapų:

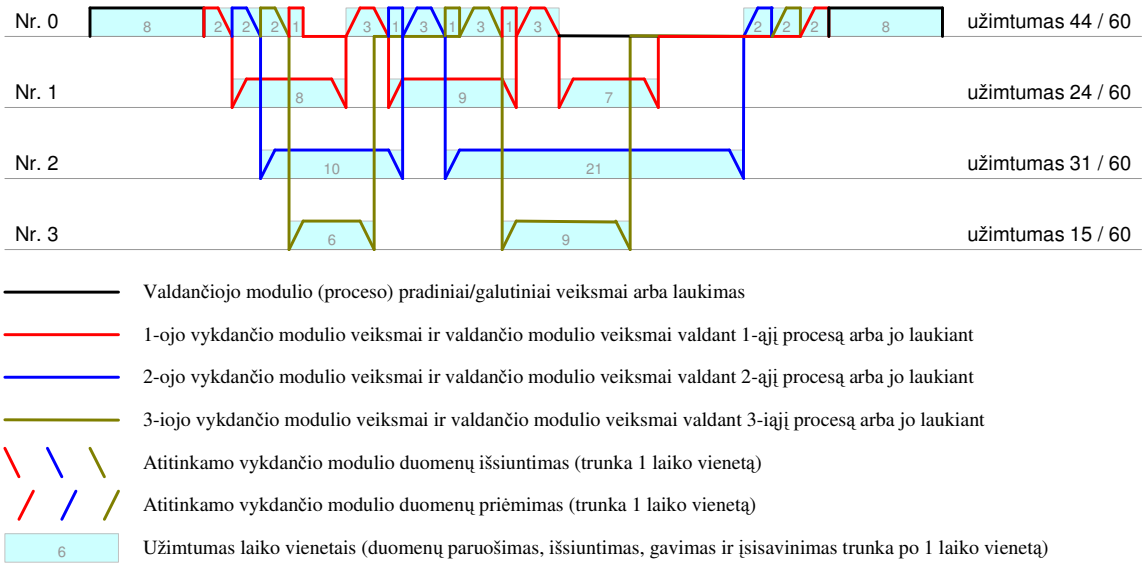
- Matematinės išraiškos priėmimas. Modulis laukia, kol valdantysis procesas persiunčia jam kokią nors matematinę išraišką.
- Matematinės išraiškos diferencijavimas naudojant MIPS. Išraiška yra perduodama matematinių išraiškų pertvarkymo sistemai.
- Rezultatinės matematinės išraiškos perdavimas valdančiajam procesui. MIPS pertvarkyta išraiška yra persiunčiama valdančiam procesui.

Lygiagretaus funkcionavimo valdymo sistema užtikrina išlygiagretintą matematinės išraiškos diferencijavimą. Pasiaiškinkime kaip tai galėtų veikti ir kaip tai veikia.

Kokį matematinių išraiškų dalinimo į subišraiškas metodą bepasirinksim, vykdantieji procesai vis tiek visada veiktų taip pat, o skirtųsi tik valdantysis procesas. Produkto realizavimo stadijoje buvo sugeneruota ne viena idėja matematinių išraiškų pertvarkymo išlygiagretinimui, tačiau lygiagrečių procesų valdymo ir sinchronizavimo sudėtingumas, privertė rinktis paprastesnius metodus. Visus sugalvotus metodus galima suskirstyti į 6 klases, kurias atitiktų dviejų parametrų kombinacijos:

- vykdančiųjų procesų naudojimas (rekursinis, prioritetinis arba nuoseklus);
- išraiškų dalijimas (bet kur arba tik per sumas ir atimtis).

Pasirinkome vieną iš paprastesnių metodų – nuoseklaus vykdančiųjų procesų naudojimą ir išraiškų dalijimą tik per sumas ir atimtis ir tik pirmame lygyje (ne skliausteliuose). Žemiau pateikiame tokio išlygiagretinimo pavyzdį ir paaiškinimus.



pav. 29. Lygiagretaus funkcionavimo valdymas

Viršuje pavaizduotas procesų užimtumas laiko ašyje. Pavyzdyje naudojami 4 procesai: 1 valdantysis ir 3 vykdantieji. Pradžioje užimtas tik valdantysis procesas, kadangi vyksta sistemos inicijavimo etapas (inicijavimo etape nepavaizduotas nedaug įtakos turintis sistemos parametrų perdavimas vykdantiesiems procesams). Toliau pradedamas matematinės išraiškos dalinimas. Čia matematinė išraiška nepavaizduota, bet iš schemos matyti, kad ją sudaro 7 subišraiškos, atskirtos viena nuo kitos pliusu arba minusu. Taigi, vieną laiko vienetą užima pirmos subišraiškos atskyrimas, dar vieną laiko vienetą – išsiuntimas sekančiam vykdančiajam procesui Nr. 1. Tada valdantysis procesas imasi antros subišraiškos atskyrimo ir išsiuntimo dar sekančiam vykdančiajam procesui Nr. 2. Ir dar kartą paskutiniam procesui Nr. 3. Toliau valdantysis procesas

ir vėl atskiria ketvirtą subišraišką, bet, patikrinęs, kad sekantis vykdantysis procesas Nr. 1 vis dar užimtas, neišsiunčia jos. Dabar jau valdantysis procesas, nepriklausomai nuo to, ar kiti vykdantieji procesai yra užimti ar laisvi, laukia kol atsilaisvins vykdantysis procesas Nr. 1. Kai galų gale šis baigia perduotos matematinės subišraiškos diferencijavimą ir atsiunčia ją atgal valdančiajam procesui, šis vieną laiko vienetą priiminėja rezultatus ir dar vieną laiko vienetą prijunginėja subišraišką prie galutinės rezultatinės matematinės išraiškos. Sekanti subišraiška jau buvo paruošta, dėl to iš karto galima jos išsiuntimas vykdančiajam procesui Nr. 1. Kai valdantysis procesas išsiunčia subišraišką pertvarkymams, svarbu iš karto pasiruošti sekančią subišraišką. Pavyzdžio atveju toliau yra ruošiama subišraiška vykdančiajam procesui Nr. 2. Šį kartą laukti nereikia – priimami rezultatai, rezultatinė subišraiška prijungiama prie bendros matematinės išraiškos, pertvarkymams išsiunčiama paruošta subišraiška. Iš karto paruošiama dar sekanti subišraiška. Laukti ir vėl nereikia – iš vykdančiojo proceso Nr. 3 priimami rezultatai, rezultatinė subišraiška prijungiama prie bendros išraiškos, pertvarkymams išsiunčiama paruošta subišraiška. Ir vėl tas pats su procesu Nr. 1. Subišraiškos baigėsi, belieka laukti rezultatų ir juos prijungti. Čia viskas tokia pačia tvarka – nuosekliai.

Kaip vyksta vykdantysis procesas lengva suprasti iš schemos – atsiųstos subišraiškos pasiėmimas, pertvarkymas ir išsiuntimas. Toliau laukimas, kol bus atsiųsta kita subišraiška.

4. TYRIMAS

Šiame skyriuje apibrėžiamos tyrimo aplinkybės, įvairiais pjūviais lentelėmis ir grafikais išdėstomi atlikti eksperimentiniai tyrimai, įvertinami rezultatai. Šio skyriaus pagrindu yra daromos pagrindinės darbo išvados.

4.1. Tyrimo aplinkybės

4.1.1. Tyrimo objektas ir tikslas

Teoriniuose ir eksperimentiniuose tyrimuose tiriamas sukurtas produktas matematinėms išraiškoms pertvarkyti.

Tyrimo tikslas – ištirti matematinių išraiškų pertvarkymo, o konkrečiai diferencijavimo, greitį, palyginti jį su kitų sistemų greičiu, nustatyti sistemos ir duomenų charakteristikų kitimo įtaką skaičiavimų greičiui.

4.1.2. Tyrimo kryptys

Tyrimo metu buvo lyginamas sistemos greitis esant besikeičiančioms sąlygoms. Sąlygomis čia vadiname įvairių sistemos ir duomenų charakteristikų rinkinius. Išskyrėme tokias charakteristikas, kurių kitimo pagrindu galėtume atlikti tyrimą:

- sistemoje naudojamų procesorių kiekis;
- lygiagrečių procesų kiekis;
- išlygiagretinimo algoritmas;
- matematinės išraiškos ilgis;
- matematinės išraiškos sudėtingumas;
- kompiuterinės technikos našumas;
- papildomi sistemos apkrovimai.

Pagrindinėmis tyrimų kryptimis pasirinkome sistemoje naudojamų procesorių kiekio ir matematinės išraiškos ilgio įtakos diferencijavimo greičiui tyrimus.

Tyrime įtraukėme ir matematinių išraiškų pertvarkymo greičio palyginimą su kitais produktais.

4.1.3. Tyrimo įrankiai

Tyrimams atlikti buvo naudojama testavimo etape sukurto automatinio testavimo įrankio atskira modifikacija. Įrankio pagalba galima vykdyti matematinių išraiškų pertvarkymą ir greta rezultatų gauti pertvarkymams sugaištą laiką sekundėmis.

Taip pat buvo naudojami atskirai sukurti testinių atvejų generavimo ir testavimo rezultatų apibendrintos suvestinės sudarymo įrankiai.

4.1.4. Tyrimo duomenys

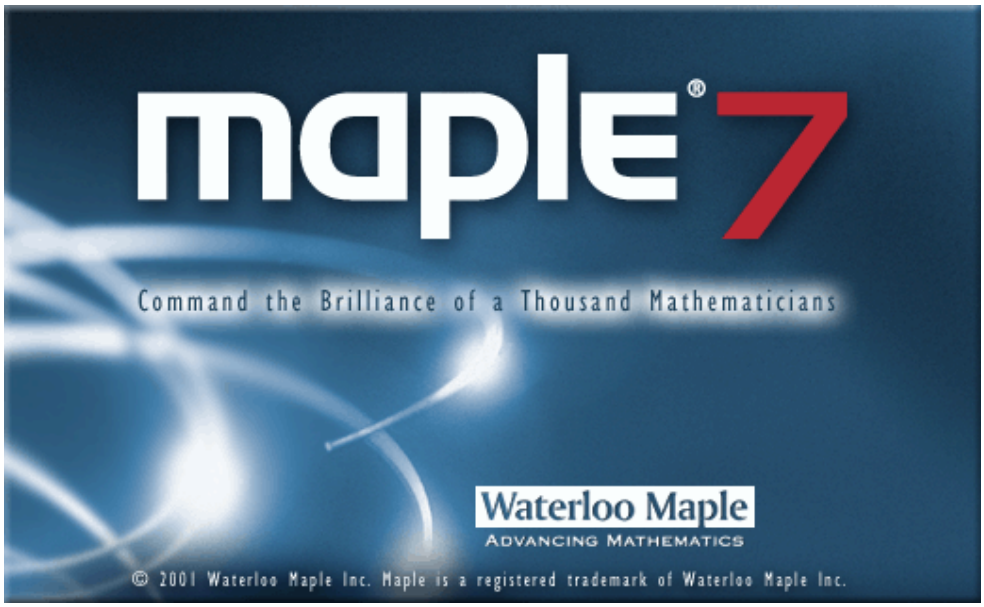
Tyrimams atlikti buvo naudojama testavimo etape sukauptų testinių atvejų biblioteka. Be šiai bibliotekai priklausančių matematinių išraiškų, papildomai buvo sugeneruotos dar kelios labai ilgos išraiškos.

Visose tyrimuose naudotose matematinėse išraiškose figūruoja įvairūs reikalavimuose numatyti operandai: neigimas, suma, atimtis, daugyba, dalyba, kėlimas laipsniu, sinusas ir kosinusas. Taip pat intensyviai naudojami ir skliausteliai. Tyrimo metu duomenims naudotos matematinės išraiškos buvo nuo 28.567 iki 28.851.913 simbolių ilgio, o pertvarkytos matematinės išraiškos buvo nuo 61.432 iki 62.372.342 simbolių ilgio.

4.1.5. Programinė įranga

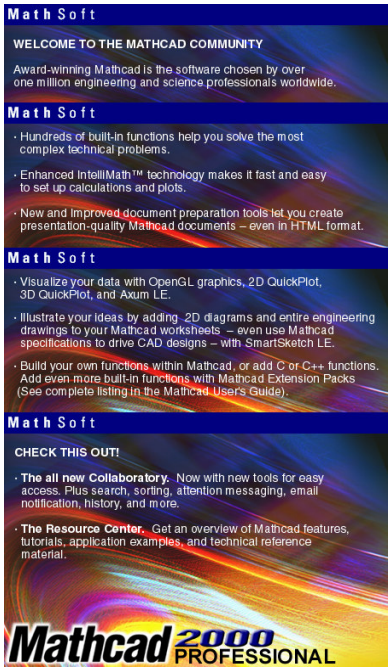
Sukurto produkto matematinių išraiškų pertvarkymo greitis gali būti lyginamas su įvairiais universaliais matematiniais produktais: Maple, Mathcad, MatLab, Mathematica ir pan.

Pirmiausia pasirinkome Maple, nes jį naudoja užsakovas. Mūsų sukurtoje sistemoje prie šio produkto yra priderintos matematinės išraiškos vaizdavimo taisyklės. Maple sukūrė kompanija Waterloo Maple, Inc. Buvo pasirinkta Maple 7 versija (naujausia versija – Maple 9.5).



pav. 30. Universalus matematinis produktas Maple 7

Taip pat keli testai buvo vykdomi ir su Mathcad. Tai populiariausias universalus matematinis produktas pasaulyje. Mathcad kitais tikslais sistemos kūrėjai yra naudoję ir anksčiau. Mathcad sukūrė kompanija MathSoft. Tyrimams buvo pasirinkta Mathcad 2000 Professional versija (naujausia versija – Mathcad 11.2).



pav. 31. Universalus matematinis produktas Mathcad 2000 Professional

Tačiau Mathcad priemonėmis nepavyko sukurti automatizuoto tyrimo įrankių, dėl to tyrimas šiuo produktu buvo nutrauktas. Tai neturėtų labai pakenkti tyrimų patikimumui, nes daugelis šaltinių teigia, kad tyrimams naudojama Maple sistema yra pati profesionaliausia.

4.1.6. Kompiuterinė technika

Tyrimai buvo atliekami naudojant Kauno Technologijos Universitetui priklausančio Informacinių technologijų plėtros instituto 320 auditorijos kompiuterių sistemą. Čia yra sukonfigūruotas 8 kompiuterių, kurių charakteristikos pateiktos žemiau esančioje lentelėje, klasteris.

lentelė 11. Klasterio kompiuterių charakteristikos

Eil. Nr.	Kompiuterio vardas	Archi-tektūra	Operacinė sistema	Procesorių kiekis	Procesorių dažnis	Atminties kiekis
1.	tulpe.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
2.	kopustas.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
3.	alyva.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
4.	zibute.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
5.	roze.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.276 MB
6.	kietas.elen.ktu.lt	sun4u	SunOS 5.9	1 / 1	167 MHz	498.160 MB
7.	geras.elen.ktu.lt	sun4u	SunOS 5.9	1 / 1	270 MHz	498.184 MB
8.	astra.elen.ktu.lt	x86	Linux 2.4.21	1 / 2	2.992 MHz	1.032.472 MB

Sukurtos sistemos tyrimų metu nebuvo naudojami 1-as, 6-as ir 7-as kompiuteriai. 1-asis buvo sugedęs, o 6-as ir 7-as nebuvo naudojami dėl jų našumo trūkumo bei operacinių sistemų nesuderinamumo.

4.2. Eksperimentai

4.2.1. Naudojamų procesorių kiekio įtaka

Naudojamų procesorių kiekio įtaka pertvarkymų greičiui buvo tiriama naudojant nuo 173.741 iki 28.851.913 simbolių ilgio matematines išraiškas. Buvo naudojami nuo 2 iki 5 procesorių. Žemiau pateikti tyrimų rezultatai lentelės ir grafikų formomis.

lentelė 12. Naudojamų procesorių kiekio įtaka

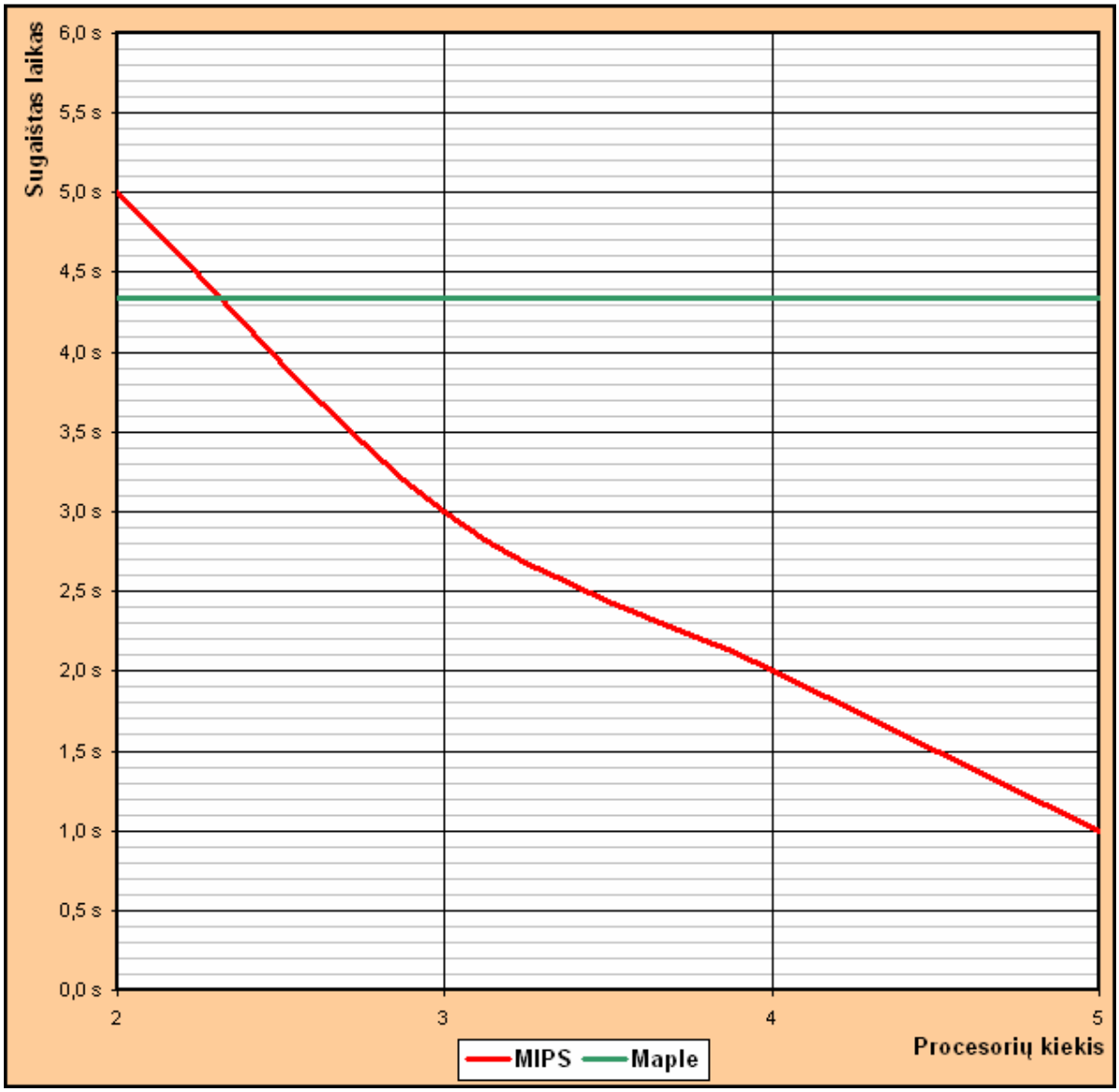
Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9203.txt	173.741	expr9203.txt	377.280	5 s	4 s
3	3	expr9203.txt	173.741	expr9203.txt	377.280	3 s	4 s
4	4	expr9203.txt	173.741	expr9203.txt	377.280	2 s	4 s
5	5	expr9203.txt	173.741	expr9203.txt	377.280	1 s	4 s
2	2	expr9304.txt	287.155	expr9304.txt	622.657	10 s	8 s
3	3	expr9304.txt	287.155	expr9304.txt	622.657	5 s	8 s
4	4	expr9304.txt	287.155	expr9304.txt	622.657	4 s	8 s
5	5	expr9304.txt	287.155	expr9304.txt	622.657	3 s	8 s
2	2	expr9405.txt	863.846	expr9405.txt	1.868.551	28 s	22 s
3	3	expr9405.txt	863.846	expr9405.txt	1.868.551	14 s	22 s
4	4	expr9405.txt	863.846	expr9405.txt	1.868.551	10 s	22 s
5	5	expr9405.txt	863.846	expr9405.txt	1.868.551	7 s	22 s
2	2	expr9506.txt	1.725.384	expr9506.txt	3.734.976	56 s	45 s
3	3	expr9506.txt	1.725.384	expr9506.txt	3.734.976	29 s	45 s
4	4	expr9506.txt	1.725.384	expr9506.txt	3.734.976	21 s	45 s
5	5	expr9506.txt	1.725.384	expr9506.txt	3.734.976	15 s	45 s
2	2	expr9607.txt	2.875.741	expr9607.txt	6.223.370	92 s	69 s
3	3	expr9607.txt	2.875.741	expr9607.txt	6.223.370	49 s	69 s
4	4	expr9607.txt	2.875.741	expr9607.txt	6.223.370	34 s	69 s
5	5	expr9607.txt	2.875.741	expr9607.txt	6.223.370	26 s	69 s
2	2	expr9708.txt	8.640.976	expr9708.txt	18.729.545	277 s	146 s
3	3	expr9708.txt	8.640.976	expr9708.txt	18.729.545	146 s	146 s
4	4	expr9708.txt	8.640.976	expr9708.txt	18.729.545	100 s	146 s
5	5	expr9708.txt	8.640.976	expr9708.txt	18.729.545	75 s	146 s
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
3	3	expr9800.txt	17.305.880	expr9800.txt	37.409.525	294 s	249 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
5	5	expr9800.txt	17.305.880	expr9800.txt	37.409.525	156 s	249 s
2	2	expr9901.txt	28.851.913	expr9901.txt	62.372.342	926 s	604 s
3	3	expr9901.txt	28.851.913	expr9901.txt	62.372.342	491 s	604 s
4	4	expr9901.txt	28.851.913	expr9901.txt	62.372.342	337 s	604 s
5	5	expr9901.txt	28.851.913	expr9901.txt	62.372.342	260 s	604 s

Sekančiuose skyreliuose naudojamų procesorių įtaka matematinių išraiškų pertvarkymų greičiui yra detalizuojama ir palyginama su Maple.

4.2.2. Naudojamų procesorių kiekio įtaka (173.741 simbolio išraiška)

lentelė 13. Naudojamų procesorių kiekio įtaka (173.741 simbolio išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9203.txt	173.741	expr9203.txt	377.280	5 s	4 s
3	3	expr9203.txt	173.741	expr9203.txt	377.280	3 s	4 s
4	4	expr9203.txt	173.741	expr9203.txt	377.280	2 s	4 s
5	5	expr9203.txt	173.741	expr9203.txt	377.280	1 s	4 s

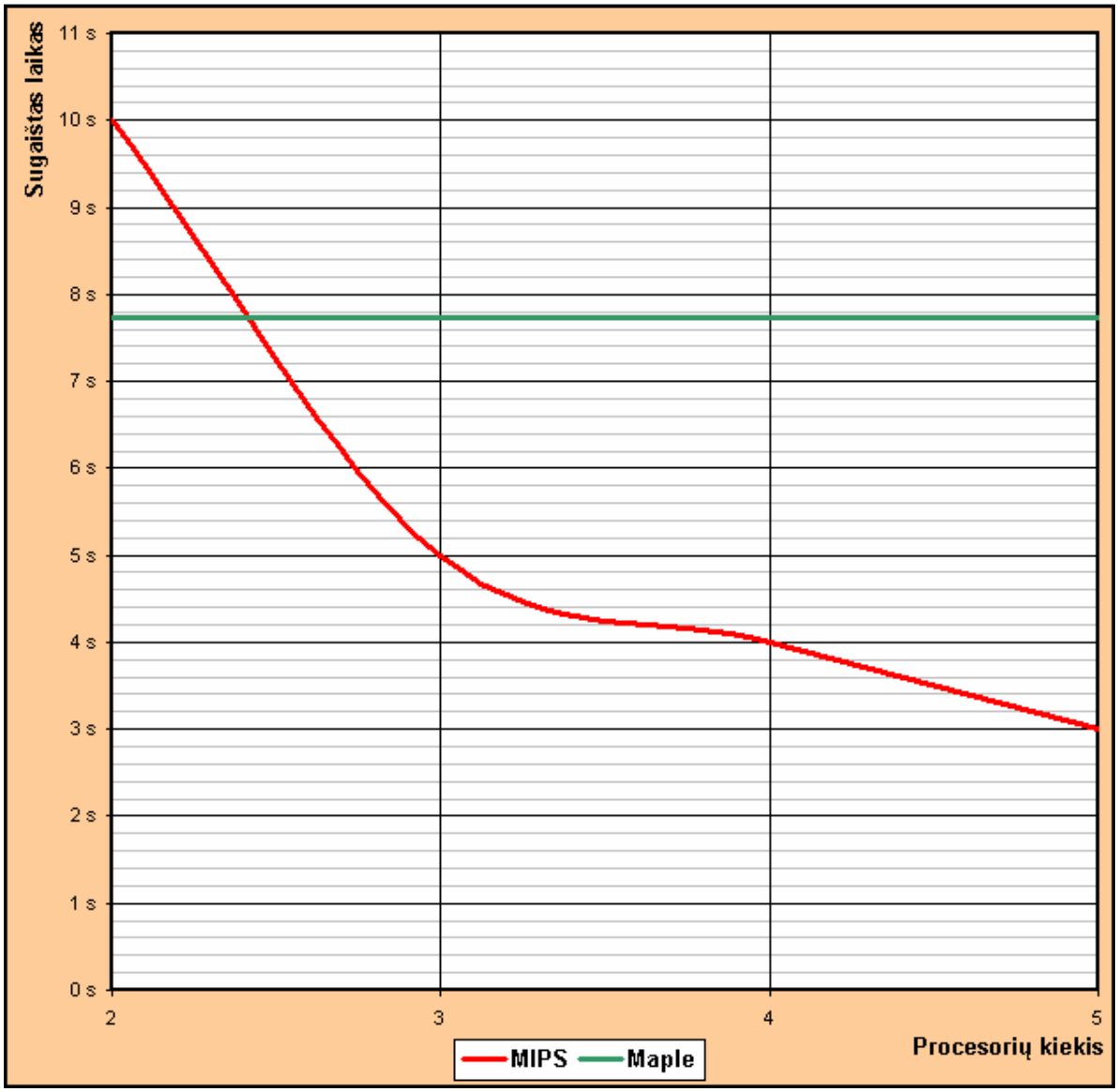


pav. 32. Naudojamų procesorių kiekio įtaka (173.741 simbolio išraiška)

4.2.3. Naudojamų procesorių kiekio įtaka (287.155 simbolių išraiška)

lentelė 14. Naudojamų procesorių kiekio įtaka (287.155 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9304.txt	287.155	expr9304.txt	622.657	10 s	8 s
3	3	expr9304.txt	287.155	expr9304.txt	622.657	5 s	8 s
4	4	expr9304.txt	287.155	expr9304.txt	622.657	4 s	8 s
5	5	expr9304.txt	287.155	expr9304.txt	622.657	3 s	8 s

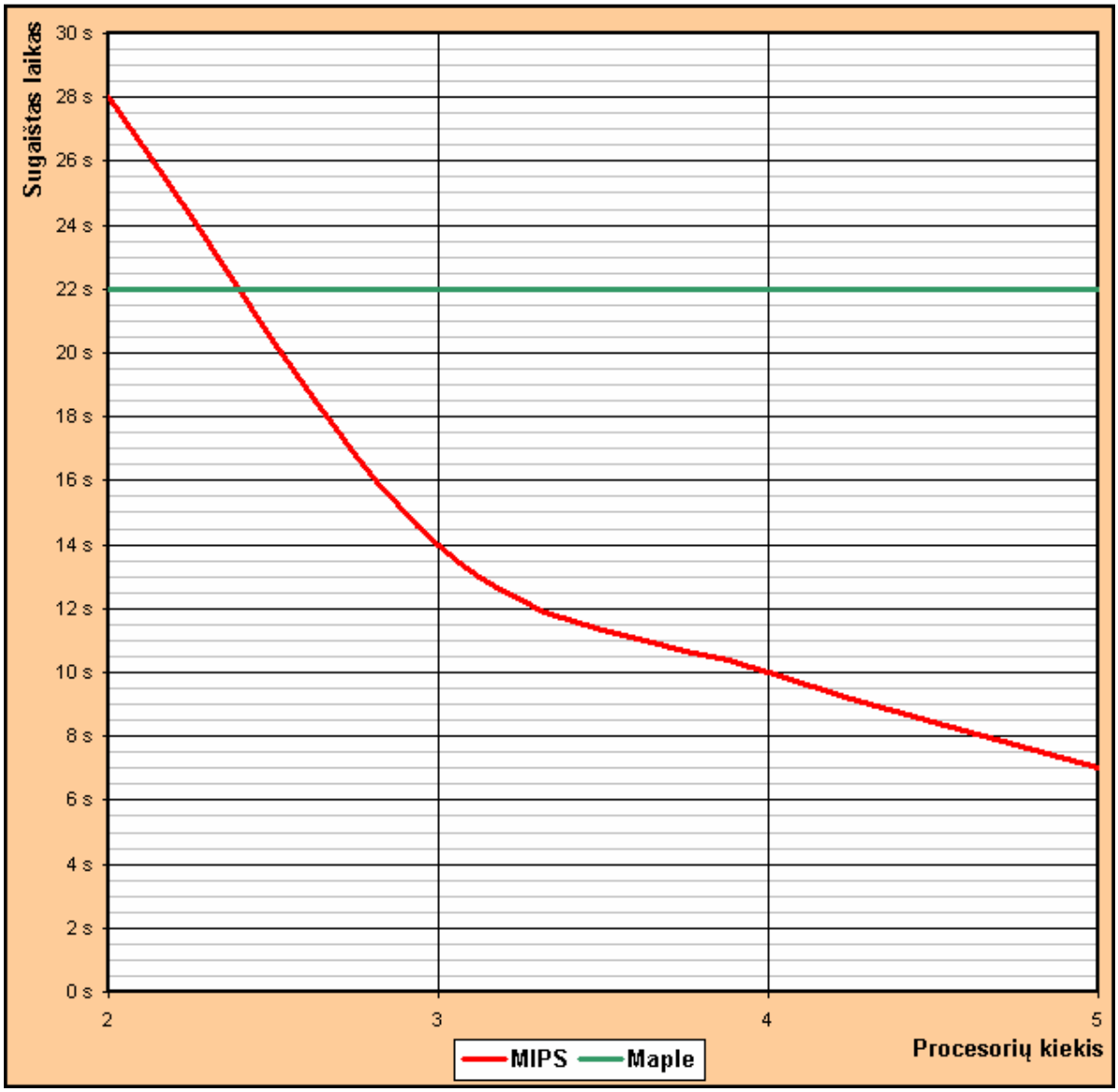


pav. 33. Naudojamų procesorių kiekio įtaka (287.155 simbolių išraiška)

4.2.4. Naudojamų procesorių kiekio įtaka (863.846 simbolių išraiška)

lentelė 15. Naudojamų procesorių kiekio įtaka (863.846 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9405.txt	863.846	expr9405.txt	1.868.551	28 s	22 s
3	3	expr9405.txt	863.846	expr9405.txt	1.868.551	14 s	22 s
4	4	expr9405.txt	863.846	expr9405.txt	1.868.551	10 s	22 s
5	5	expr9405.txt	863.846	expr9405.txt	1.868.551	7 s	22 s

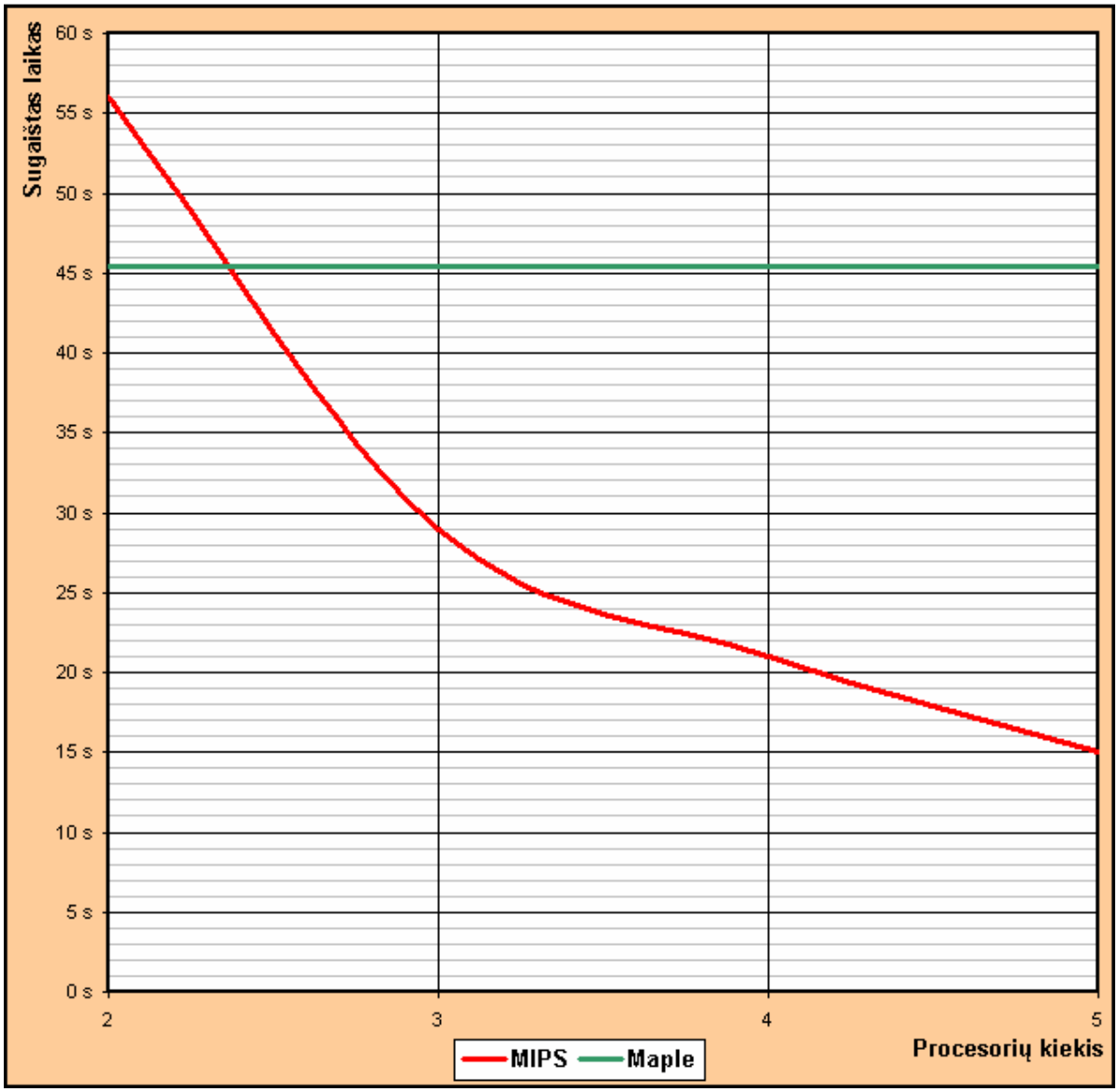


pav. 34. Naudojamų procesorių kiekio įtaka (863.846 simbolių išraiška)

4.2.5. Naudojamų procesorių kiekio įtaka (1.725.384 simbolių išraiška)

lentelė 16. Naudojamų procesorių kiekio įtaka (1.725.384 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9506.txt	1.725.384	expr9506.txt	3.734.976	56 s	45 s
3	3	expr9506.txt	1.725.384	expr9506.txt	3.734.976	29 s	45 s
4	4	expr9506.txt	1.725.384	expr9506.txt	3.734.976	21 s	45 s
5	5	expr9506.txt	1.725.384	expr9506.txt	3.734.976	15 s	45 s

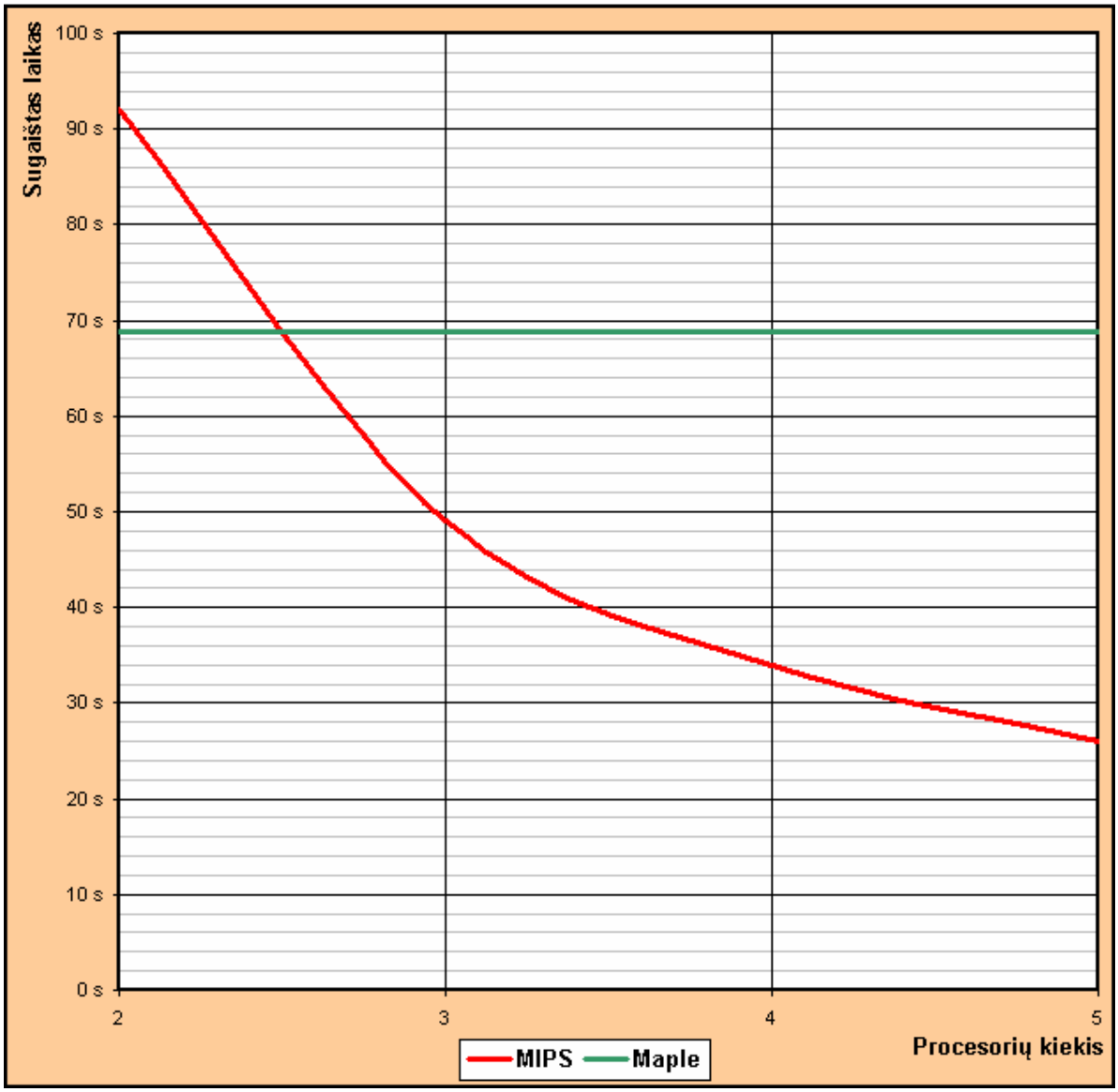


pav. 35. Naudojamų procesorių kiekio įtaka (1.725.384 simbolių išraiška)

4.2.6. Naudojamų procesorių kiekio įtaka (2.875.741 simbolio išraiška)

lentelė 17. Naudojamų procesorių kiekio įtaka (2.875.741 simbolio išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9607.txt	2.875.741	expr9607.txt	6.223.370	92 s	69 s
3	3	expr9607.txt	2.875.741	expr9607.txt	6.223.370	49 s	69 s
4	4	expr9607.txt	2.875.741	expr9607.txt	6.223.370	34 s	69 s
5	5	expr9607.txt	2.875.741	expr9607.txt	6.223.370	26 s	69 s

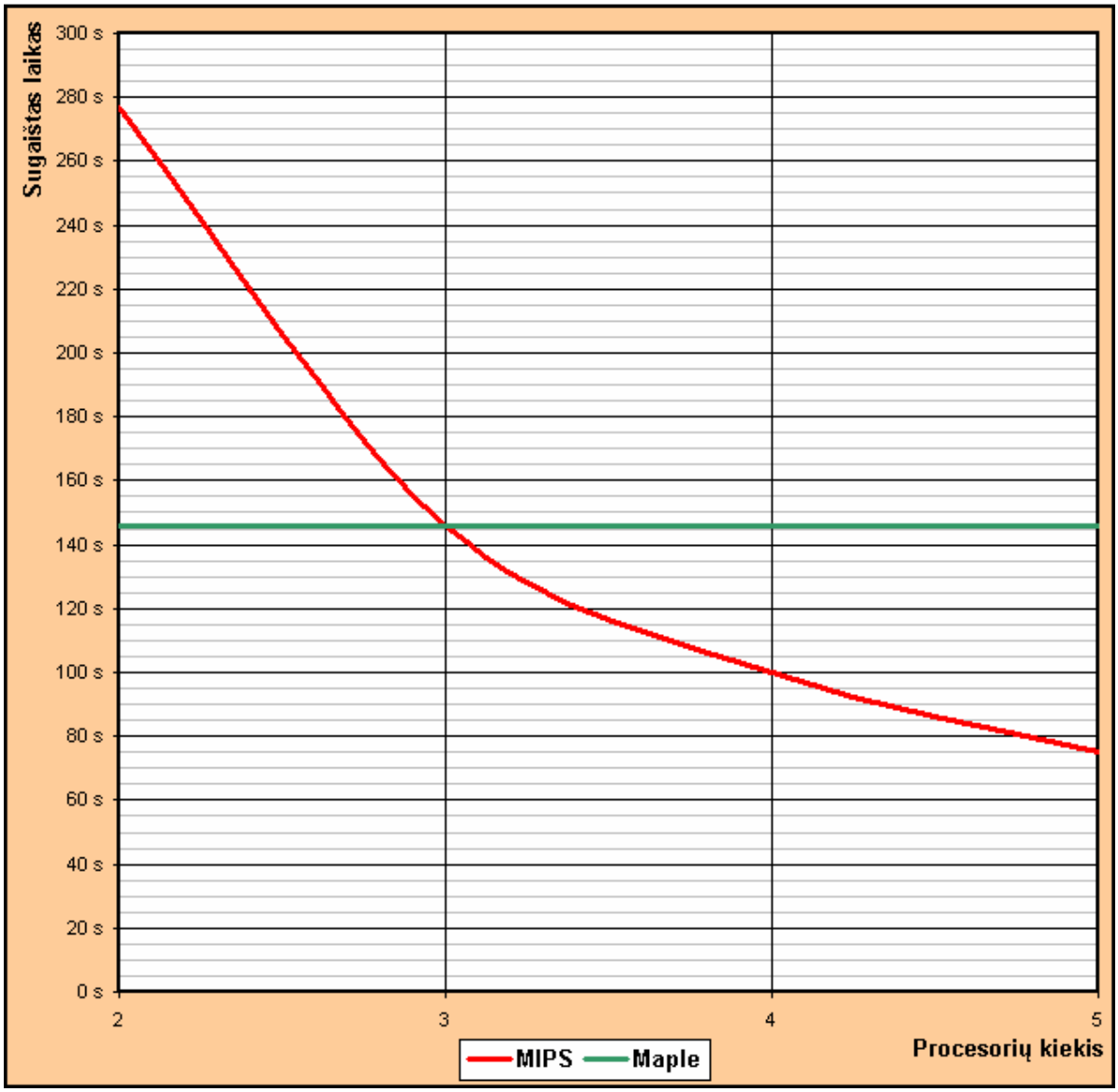


pav. 36. Naudojamų procesorių kiekio įtaka (2.875.741 simbolio išraiška)

4.2.7. Naudojamų procesorių kiekio įtaka (8.640.976 simbolių išraiška)

lentelė 18. Naudojamų procesorių kiekio įtaka (8.640.976 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9708.txt	8.640.976	expr9708.txt	18.729.545	277 s	146 s
3	3	expr9708.txt	8.640.976	expr9708.txt	18.729.545	146 s	146 s
4	4	expr9708.txt	8.640.976	expr9708.txt	18.729.545	100 s	146 s
5	5	expr9708.txt	8.640.976	expr9708.txt	18.729.545	75 s	146 s

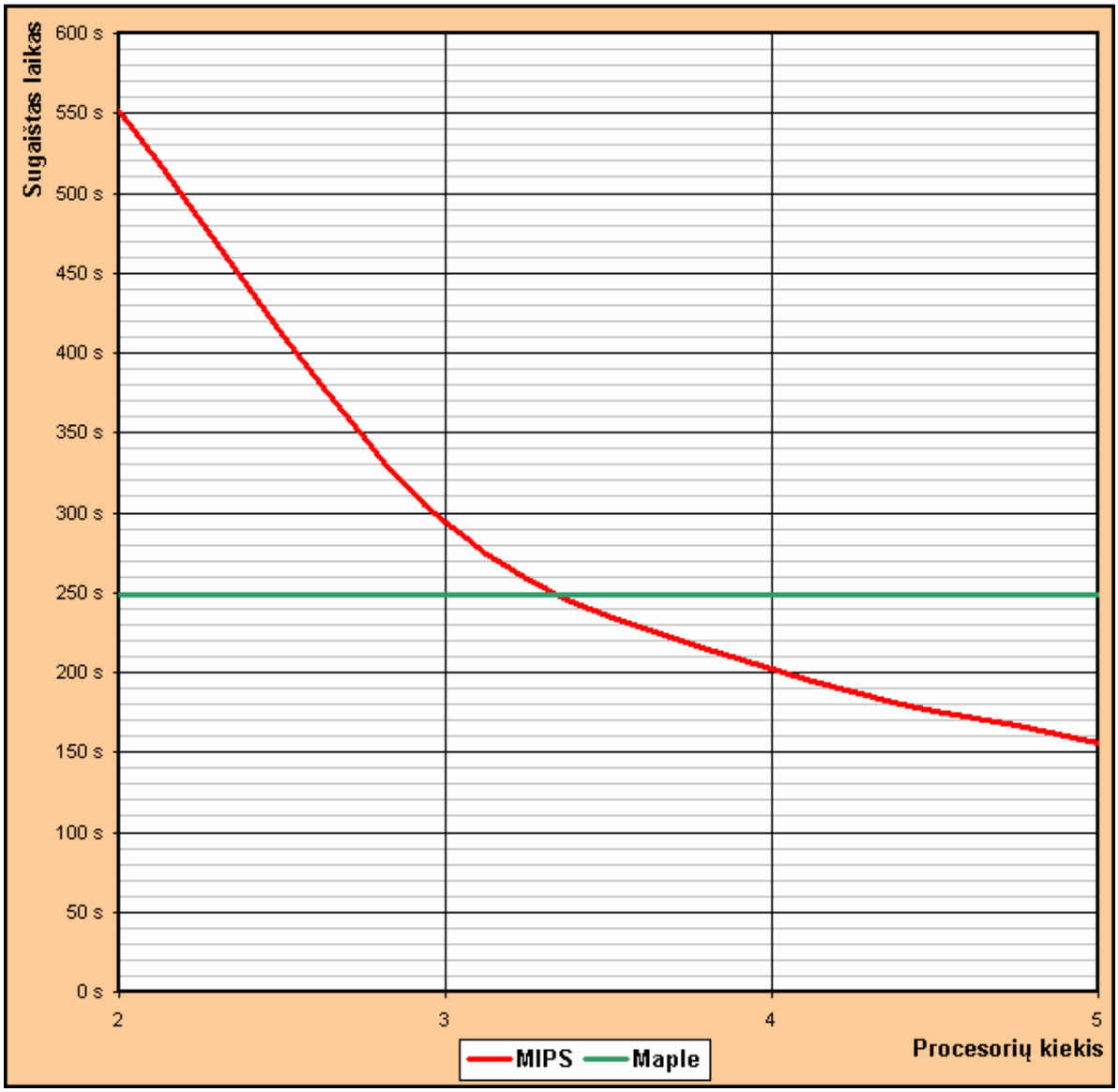


pav. 37. Naudojamų procesorių kiekio įtaka (8.640.976 simbolių išraiška)

4.2.8. Naudojamų procesorių kiekio įtaka (17.305.880 simbolių išraiška)

lentelė 19. Naudojamų procesorių kiekio įtaka (17.305.880 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
3	3	expr9800.txt	17.305.880	expr9800.txt	37.409.525	294 s	249 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
5	5	expr9800.txt	17.305.880	expr9800.txt	37.409.525	156 s	249 s

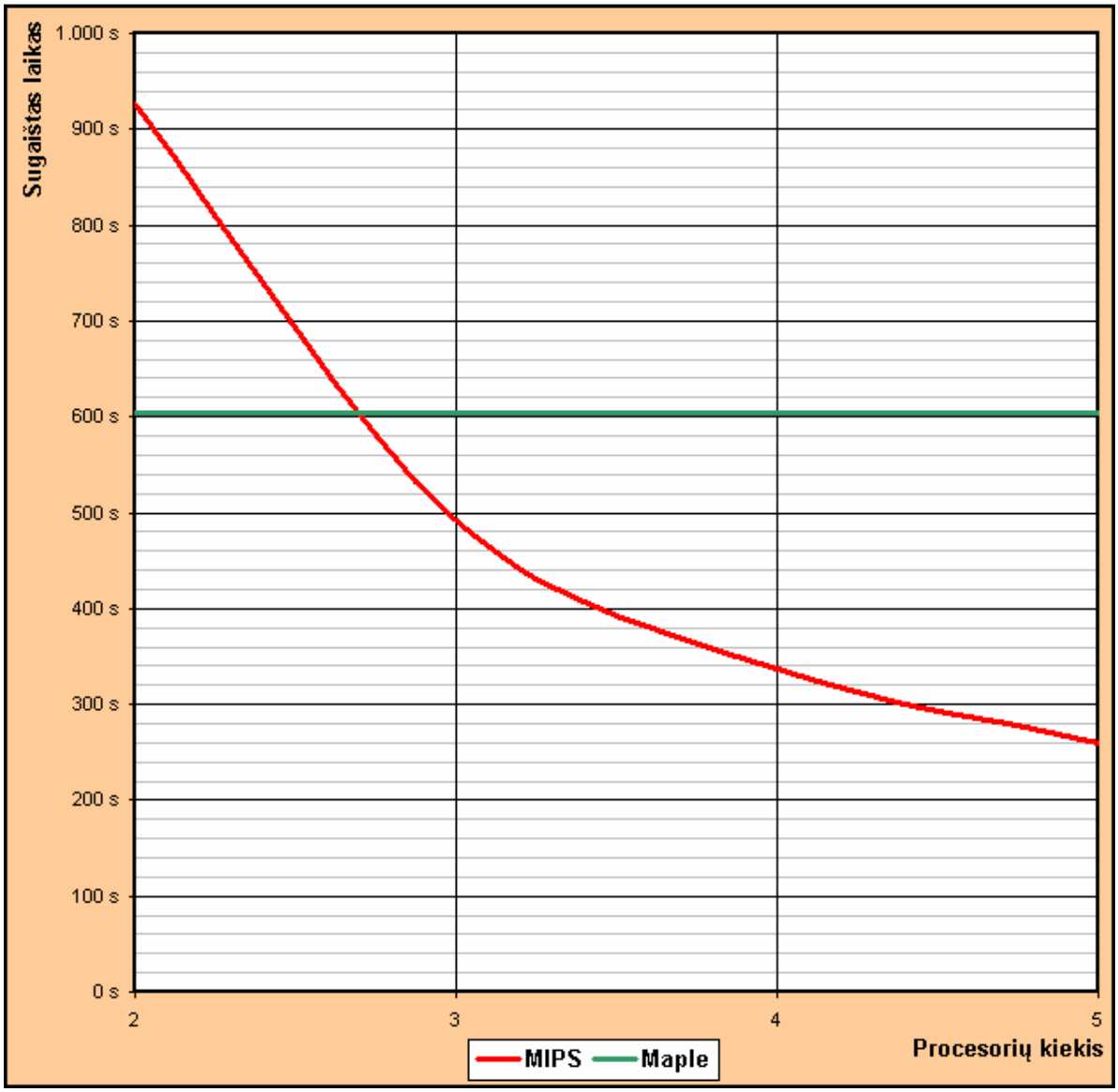


pav. 38. Naudojamų procesorių kiekio įtaka (17.305.880 simbolių išraiška)

4.2.9. Naudojamų procesorių kiekio įtaka (28.851.913 simbolių išraiška)

lentelė 20. Naudojamų procesorių kiekio įtaka (28.851.913 simbolių išraiška)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9901.txt	28.851.913	expr9901.txt	62.372.342	926 s	604 s
3	3	expr9901.txt	28.851.913	expr9901.txt	62.372.342	491 s	604 s
4	4	expr9901.txt	28.851.913	expr9901.txt	62.372.342	337 s	604 s
5	5	expr9901.txt	28.851.913	expr9901.txt	62.372.342	260 s	604 s



pav. 39. Naudojamų procesorių kiekio įtaka (28.851.913 simbolių išraiška)

4.2.10. Matematinių išraiškų ilgio įtaka

Matematinių išraiškų ilgio įtaka pertvarkymų greičiui buvo tiriama sistemoje naudojant nuo 2 iki 5 procesorių ir tiek pat lygiagrečių procesų. Buvo pasirinktos nuo 28.694 iki 8.621.540 simbolių ilgio matematinės išraiškos.

lentelė 21. Matematinių išraiškų ilgio įtaka

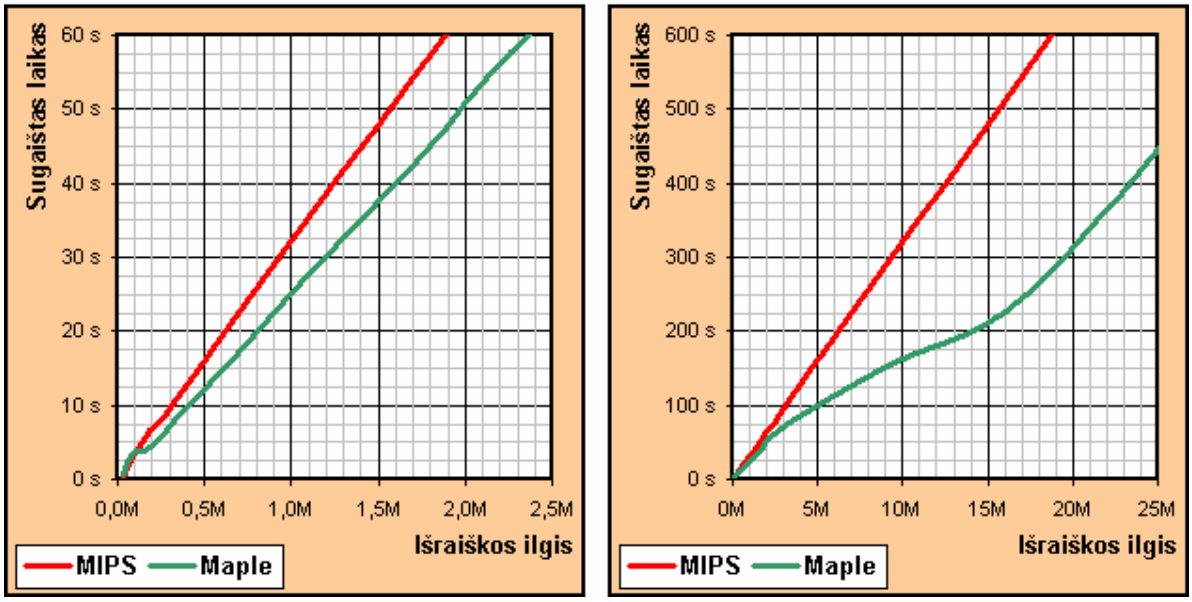
Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
2	2	expr9100.txt	86.160	expr9100.txt	185.720	3 s	4 s
2	2	expr9200.txt	171.825	expr9200.txt	372.238	6 s	4 s
2	2	expr9300.txt	285.527	expr9300.txt	619.136	9 s	7 s
2	2	expr9400.txt	865.470	expr9400.txt	1.876.867	28 s	22 s
2	2	expr9500.txt	1.721.990	expr9500.txt	3.726.800	55 s	43 s
2	2	expr9600.txt	2.878.752	expr9600.txt	6.240.910	92 s	69 s
2	2	expr9700.txt	8.621.540	expr9700.txt	18.678.390	278 s	146 s
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
2	2	expr9900.txt	28.845.371	expr9900.txt	62.350.917	926 s	550 s
3	3	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
3	3	expr9100.txt	86.160	expr9100.txt	185.720	2 s	4 s
3	3	expr9200.txt	171.825	expr9200.txt	372.238	3 s	4 s
3	3	expr9300.txt	285.527	expr9300.txt	619.136	4 s	7 s
3	3	expr9400.txt	865.470	expr9400.txt	1.876.867	14 s	22 s
3	3	expr9500.txt	1.721.990	expr9500.txt	3.726.800	29 s	43 s
3	3	expr9600.txt	2.878.752	expr9600.txt	6.240.910	49 s	69 s
3	3	expr9700.txt	8.621.540	expr9700.txt	18.678.390	146 s	146 s
3	3	expr9800.txt	17.305.880	expr9800.txt	37.409.525	294 s	249 s
3	3	expr9900.txt	28.845.371	expr9900.txt	62.350.917	489 s	550 s
4	4	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
4	4	expr9100.txt	86.160	expr9100.txt	185.720	1 s	4 s
4	4	expr9200.txt	171.825	expr9200.txt	372.238	2 s	4 s
4	4	expr9300.txt	285.527	expr9300.txt	619.136	3 s	7 s
4	4	expr9400.txt	865.470	expr9400.txt	1.876.867	10 s	22 s
4	4	expr9500.txt	1.721.990	expr9500.txt	3.726.800	20 s	43 s
4	4	expr9600.txt	2.878.752	expr9600.txt	6.240.910	34 s	69 s
4	4	expr9700.txt	8.621.540	expr9700.txt	18.678.390	100 s	146 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
4	4	expr9900.txt	28.845.371	expr9900.txt	62.350.917	335 s	550 s
5	5	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
5	5	expr9100.txt	86.160	expr9100.txt	185.720	1 s	4 s
5	5	expr9200.txt	171.825	expr9200.txt	372.238	2 s	4 s
5	5	expr9300.txt	285.527	expr9300.txt	619.136	2 s	7 s
5	5	expr9400.txt	865.470	expr9400.txt	1.876.867	8 s	22 s
5	5	expr9500.txt	1.721.990	expr9500.txt	3.726.800	16 s	43 s
5	5	expr9600.txt	2.878.752	expr9600.txt	6.240.910	26 s	69 s
5	5	expr9700.txt	8.621.540	expr9700.txt	18.678.390	74 s	146 s
5	5	expr9800.txt	17.305.880	expr9800.txt	37.409.525	156 s	249 s
5	5	expr9900.txt	28.845.371	expr9900.txt	62.350.917	262 s	550 s

Sekančiuose skyreliuose matematinių išraiškų ilgio įtaka pertvarkymų greičiui yra šiek tiek detalizuojama.

4.2.11. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

lentelė 22. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
2	2	expr9000.txt	28.695	expr9000.txt	61.844	0 s	0 s
2	2	expr9100.txt	86.160	expr9100.txt	185.720	3 s	4 s
2	2	expr9200.txt	171.825	expr9200.txt	372.238	6 s	4 s
2	2	expr9300.txt	285.527	expr9300.txt	619.136	9 s	7 s
2	2	expr9400.txt	865.470	expr9400.txt	1.876.867	28 s	22 s
2	2	expr9500.txt	1.721.990	expr9500.txt	3.726.800	55 s	43 s
2	2	expr9600.txt	2.878.752	expr9600.txt	6.240.910	92 s	69 s
2	2	expr9700.txt	8.621.540	expr9700.txt	18.678.390	278 s	146 s
2	2	expr9800.txt	17.305.880	expr9800.txt	37.409.525	551 s	249 s
2	2	expr9900.txt	28.845.371	expr9900.txt	62.350.917	926 s	550 s

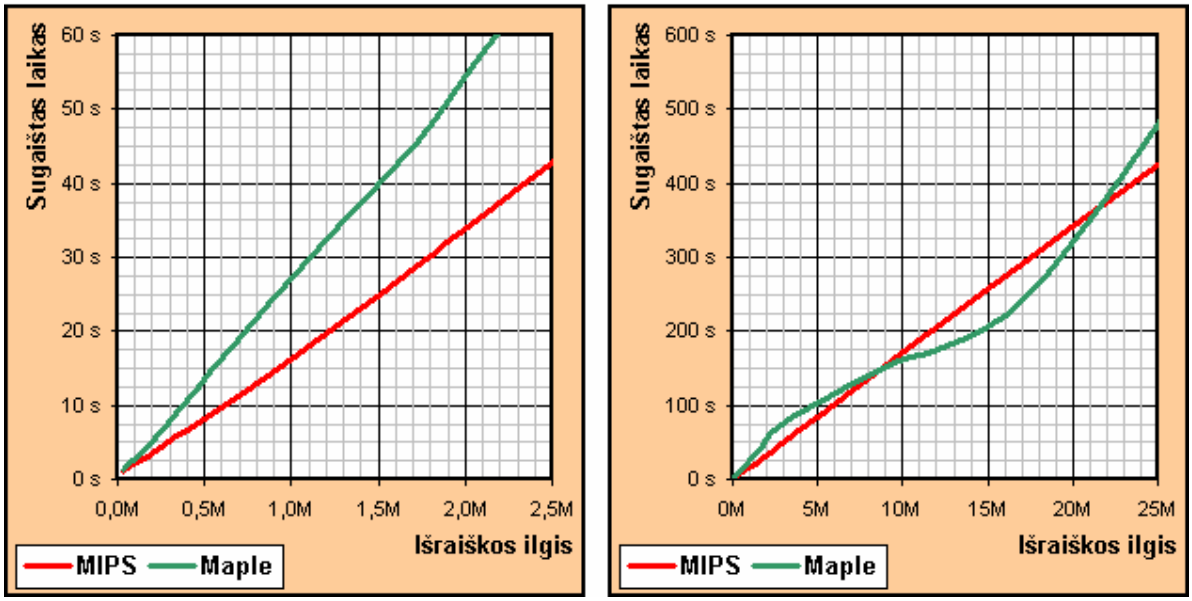


pav. 40. Matematinių išraiškų ilgio įtaka (2 naudojami procesoriai)

4.2.12. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

lentelė 23. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
3	3	expr9001.txt	28.922	expr9001.txt	63.297	1 s	1 s
3	3	expr9101.txt	86.215	expr9101.txt	186.497	2 s	3 s
3	3	expr9201.txt	173.777	expr9201.txt	378.010	3 s	4 s
3	3	expr9301.txt	287.920	expr9301.txt	623.400	5 s	8 s
3	3	expr9401.txt	866.648	expr9401.txt	1.883.594	14 s	24 s
3	3	expr9501.txt	1.727.643	expr9501.txt	3.742.278	29 s	46 s
3	3	expr9601.txt	2.872.551	expr9601.txt	6.222.326	49 s	74 s
3	3	expr9701.txt	8.647.693	expr9701.txt	18.753.983	147 s	146 s
3	3	expr9801.txt	17.308.035	expr9801.txt	37.413.622	294 s	247 s
3	3	expr9901.txt	28.851.913	expr9901.txt	62.372.342	491 s	604 s

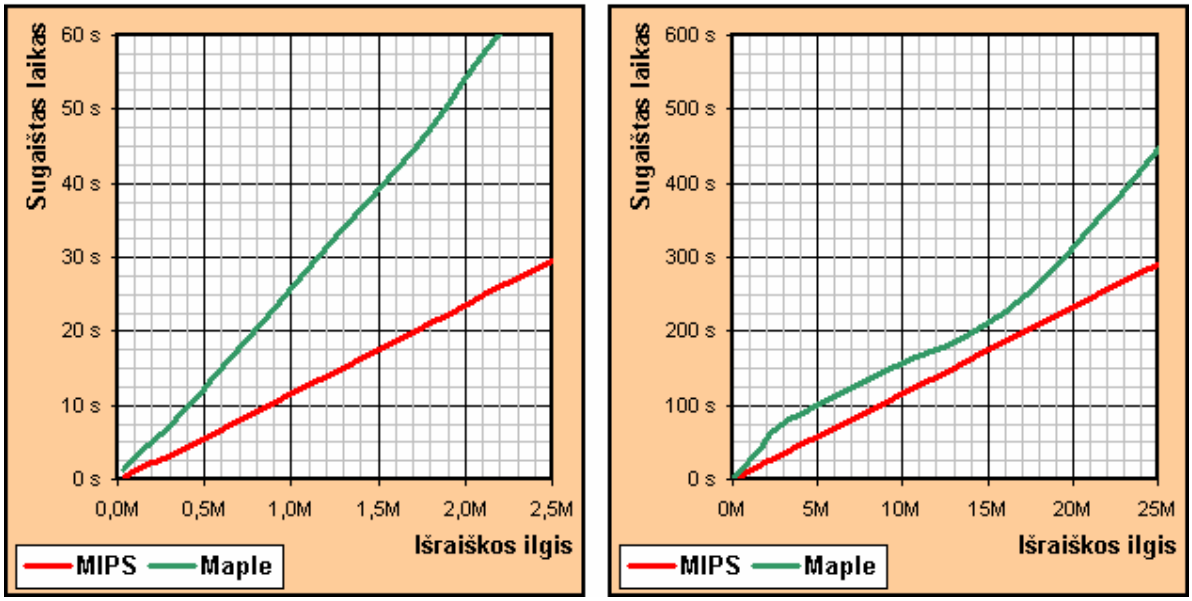


pav. 41. Matematinių išraiškų ilgio įtaka (3 naudojami procesoriai)

4.2.13. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

lentelė 24. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
4	4	expr9002.txt	28.751	expr9002.txt	62.039	0 s	1 s
4	4	expr9102.txt	85.419	expr9102.txt	183.488	1 s	3 s
4	4	expr9202.txt	172.909	expr9202.txt	374.853	2 s	5 s
4	4	expr9302.txt	286.894	expr9302.txt	620.521	3 s	7 s
4	4	expr9402.txt	864.761	expr9402.txt	1.876.381	10 s	22 s
4	4	expr9502.txt	1.725.880	expr9502.txt	3.737.053	20 s	45 s
4	4	expr9602.txt	2.878.669	expr9602.txt	6.237.786	34 s	74 s
4	4	expr9702.txt	8.626.377	expr9702.txt	18.692.958	100 s	143 s
4	4	expr9800.txt	17.305.880	expr9800.txt	37.409.525	202 s	249 s
4	4	expr9900.txt	28.845.371	expr9900.txt	62.350.917	335 s	550 s

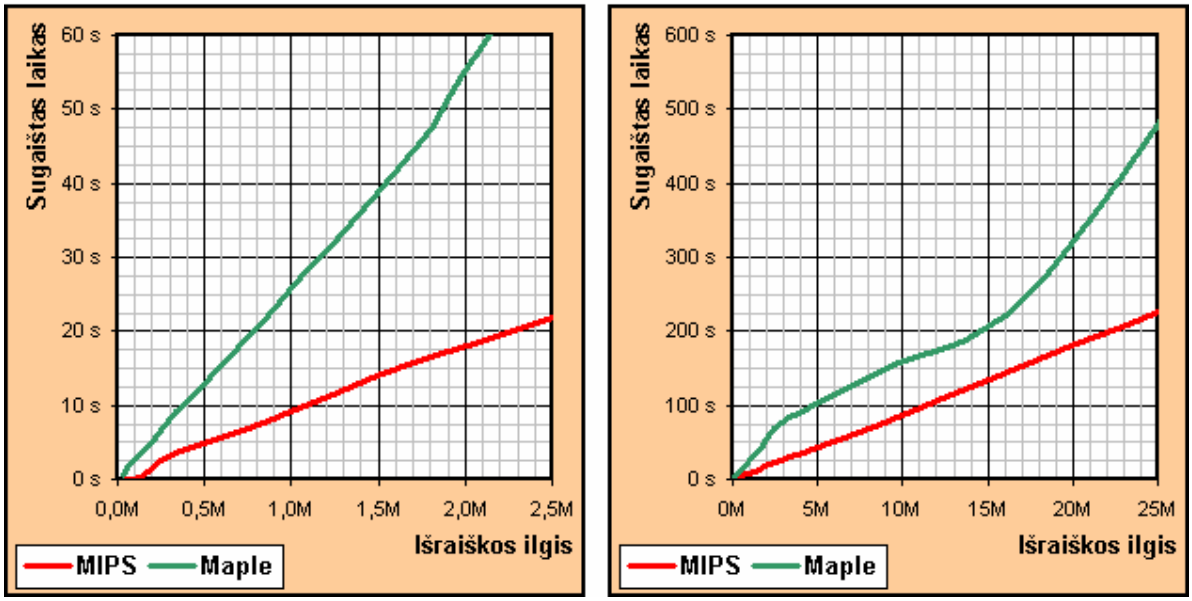


pav. 42. Matematinių išraiškų ilgio įtaka (4 naudojami procesoriai)

4.2.14. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

lentelė 25. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

Charakteristikos		Duomenys		Rezultatai		Sugaištas laikas	
Procesorių kiekis	Procesų kiekis	Duomenų failas	Išraiškos ilgis	Rezultatų failas	Išraiškos ilgis	MIPS laikas	Maple laikas
5	5	expr9003.txt	28.859	expr9003.txt	62.834	0 s	0 s
5	5	expr9103.txt	86.147	expr9103.txt	185.661	0 s	3 s
5	5	expr9203.txt	173.741	expr9203.txt	377.280	1 s	4 s
5	5	expr9303.txt	288.259	expr9303.txt	625.574	3 s	8 s
5	5	expr9403.txt	860.550	expr9403.txt	1.866.003	8 s	22 s
5	5	expr9503.txt	1.727.841	expr9503.txt	3.744.006	16 s	45 s
5	5	expr9603.txt	2.876.643	expr9603.txt	6.230.294	25 s	77 s
5	5	expr9703.txt	8.631.403	expr9703.txt	18.705.667	75 s	144 s
5	5	expr9801.txt	17.308.035	expr9801.txt	37.413.622	156 s	247 s
5	5	expr9901.txt	28.851.913	expr9901.txt	62.372.342	260 s	604 s

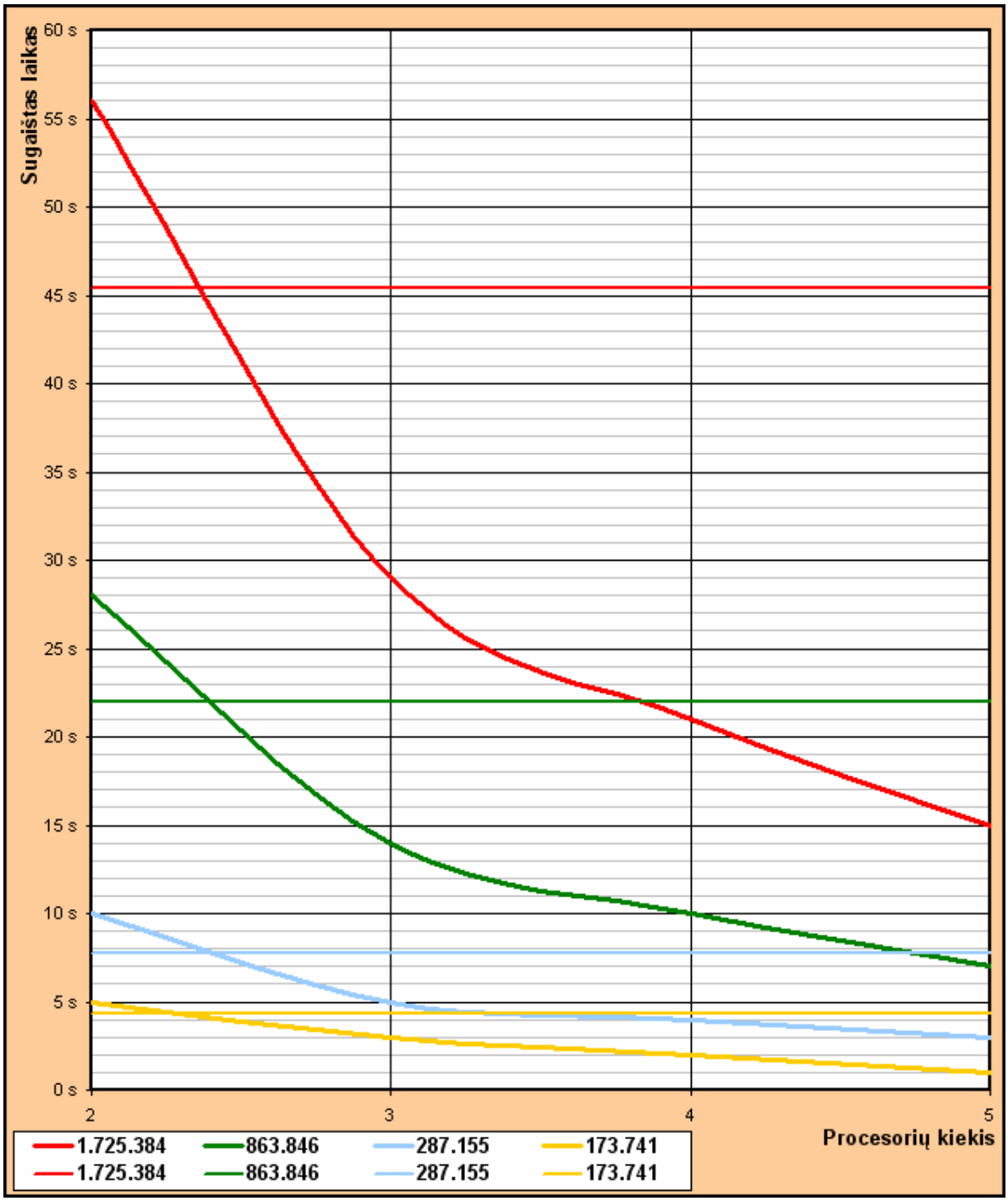


pav. 43. Matematinių išraiškų ilgio įtaka (5 naudojami procesoriai)

4.3. Rezultatų įvertinimas

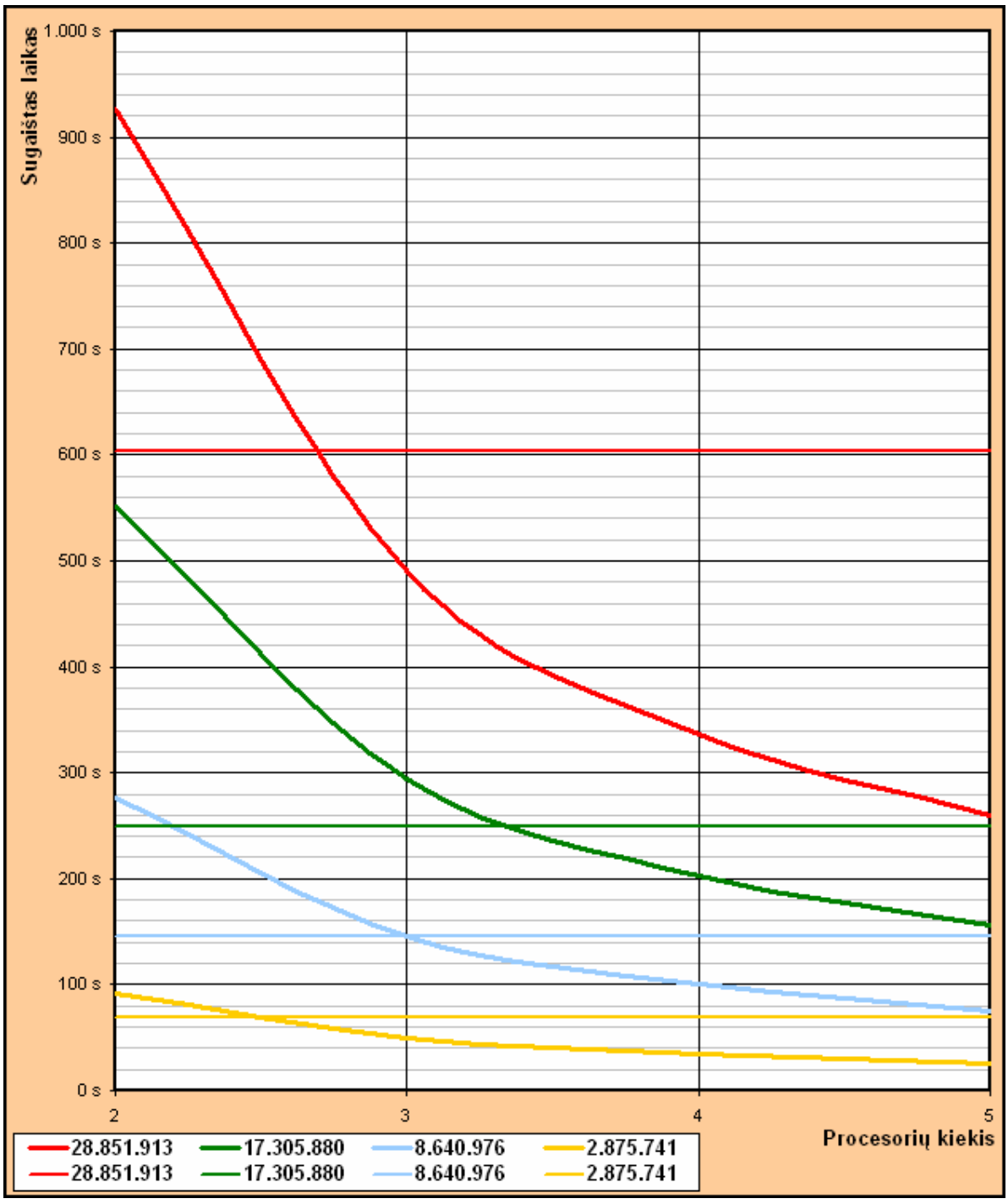
4.3.1. Naudojamų procesorių kiekio įtaka

Žemiau pateikiame apibendrintą matematinių išraiškų pertvarkymo greičio priklausomybę nuo sistemoje naudojamų procesorių kiekio. Pirmasis grafikas rodo priklausomybę, kai matematinių išraiškų ilgis yra iki 2.000.000, antrasis grafikas – iki 28.000.000 simbolių.



pav. 44. Naudojamų procesorių kiekio įtaka (iki 2 mln. simbolių)

Atskiros grafikų kreivės rodo priklausomybę esant skirtingiems matematinių išraiškų ilgiams. Horizontalios plonesnės tos pačios spalvos linijos rodo Maple sistemos greitį pertvarkant tą pačią matematinę išraišką naudojant vieną procesorių.



pav. 45. Naudojamų procesorių kiekio įtaka (iki 28 mln. simbolių)

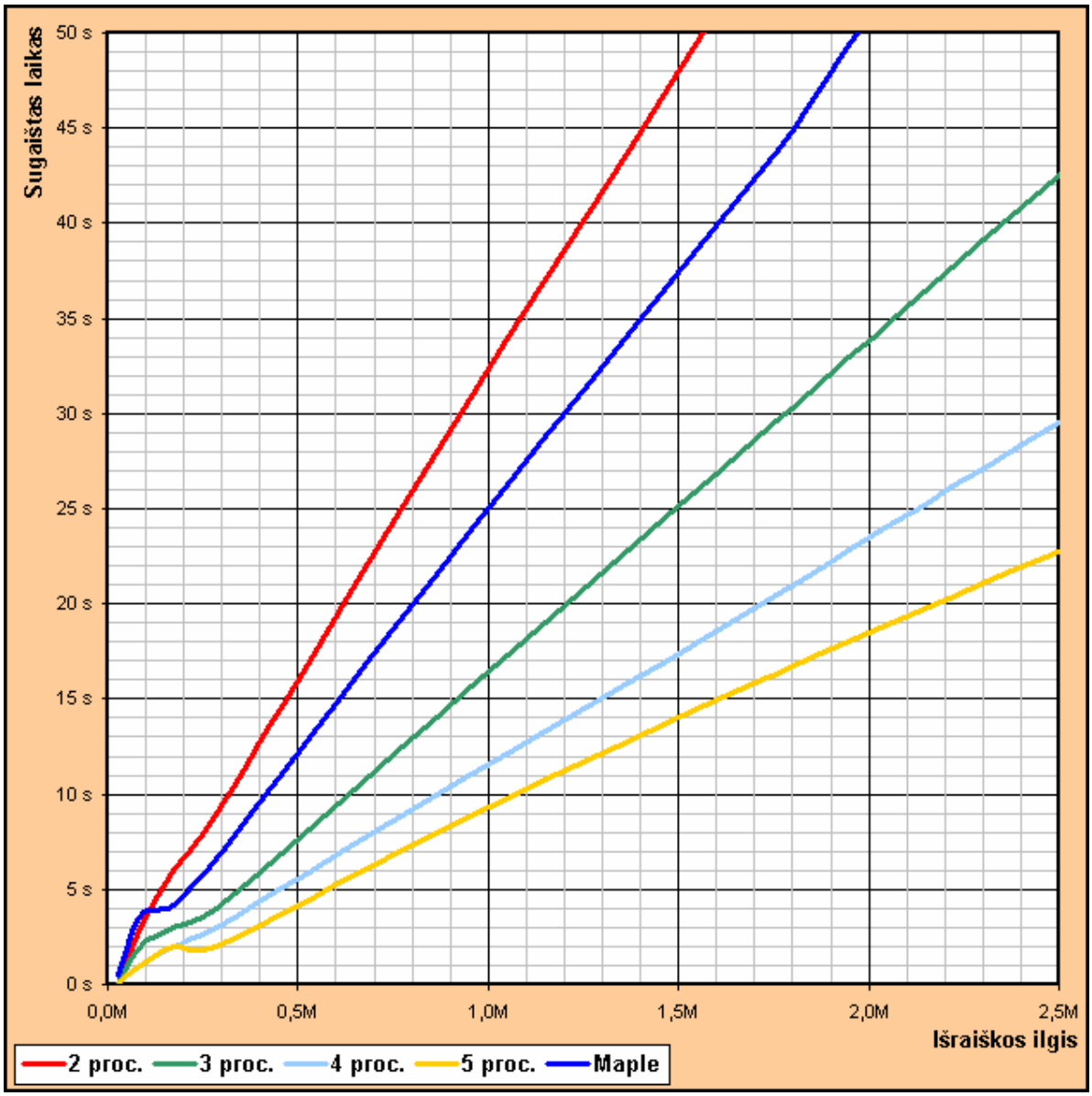
Iš grafikų matosi tokios tendencijos:

- matematinėms išraiškos nuo maždaug 8.500.000 iki maždaug 23.000.000 simbolių sistemoje reikia naudoti 4 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- visoms kitoms matematinėms išraiškos sistemoje užtenka naudoti 3 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema.

Pagal tendencijas matosi, kad Maple sistemą galime maždaug sulyginti su sukurta sistema, kai joje naudojami 3 procesoriai. Šią tendenciją nesunkiai patvirtina ir išlygiagretinimo algoritmo architektūra. Kadangi pagrindinis procesas (veikiantis 1-ajame procesoriuje) yra valdantysis, t. y. pats neatlieka pertvarkymo, tai sistemoje matematinės išraiškos pertvarkymu visada užsiima vienu procosoriumi mažiau. Kai naudojami 2 procesoriai, tai pertvarkymu užsiima tik 1 procesorius, kai 3 – tik 2 procesoriai, kai 4 – tik 3 ir t. t. Taigi, sistemoje naudojant 2 procesorius, matematinių išraiškų pertvarkymu užsiimantis vienintelis procesorius yra papildomai apkraunamas ir duomenų priėmimo bei išsiuntimo darbais.

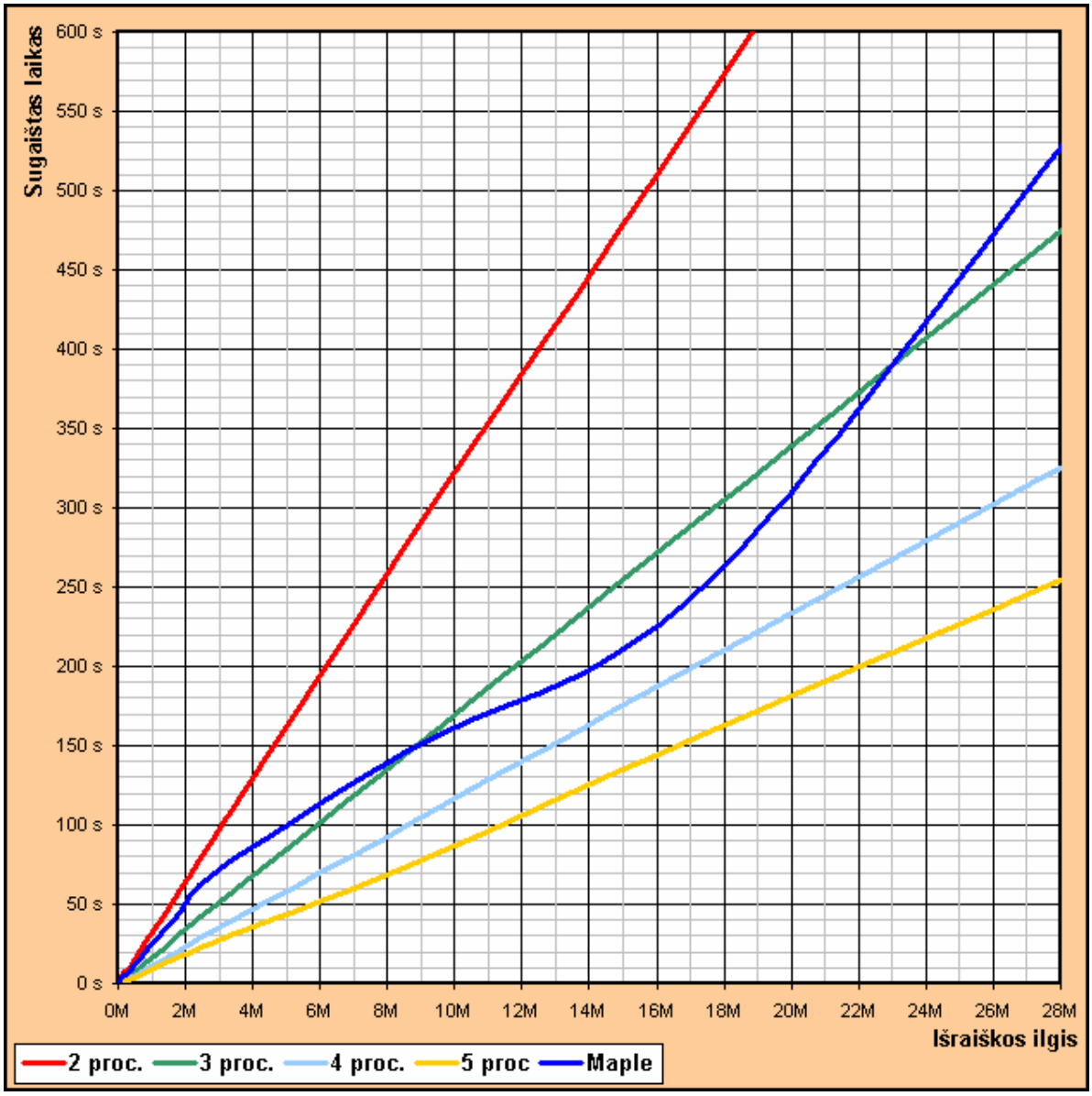
4.3.2. Matematinių išraiškų ilgio įtaka

Žemiau pateikiame apibendrintą matematinių išraiškų pertvarkymo greičio priklausomybę nuo matematinių išraiškų ilgio. Pirmasis grafikas rodo priklausomybę, kai matematinių išraiškų ilgis yra iki 2.500.000, antrasis grafikas – iki 28.000.000 simbolių.



pav. 46. Matematinių išraiškų ilgio (iki 2,5 mln. simbolių) įtaka

Atskiros grafikų kreivės rodo priklausomybę naudojant skirtingus kiekis procesorių. Mėlynoji kreivė rodo Maple sistemos greitį pertvarkant tas pačias matematines išraiškas naudojant vieną procesorių.



pav. 47. Matematinių išraiškų ilgio (iki 28 mln. simbolių) įtaka

Iš abiejų grafikų matosi tokios tendencijos:

- matematinėms išraiškoms iki 100.000 simbolių sistemoje užtenka naudoti 2 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 100.000 iki 8.500.000 simbolių sistemoje jau reikia naudoti 3 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 8.500.000 iki 23.000.000 simbolių sistemoje jau reikia naudoti 4 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema;
- matematinėms išraiškos nuo 23.000.000 simbolių sistemoje užtenka naudoti 3 procesorius, kad skaičiavimai truktų trumpiau nei su Maple sistema.

Pagal tendencijas matosi, kad Maple sistemą galime maždaug sulyginti su sukurta sistema, kai joje naudojami 3 procesoriai.

4.3.3. Rezultatų įvertinimas

Abiejų tyrimo krypčių rezultatai atspindi labai panašias tendencijas. Matosi, kad sukurtą sistemą visais atvejais naudinga naudoti, kai sistemoje yra bent 4 procesoriai, o daugeliu atveju užtenka ir 3 procesorių. Naudotojai, turintys panašią kompiuterinę sistemą, jau gali efektyviau spręsti taikymo srities uždavinius.

Tyrimų rezultatai išryškina ir sistemos trūkumus. Pagrindinis trūkumas yra tai, kad vis dėl to nepavyko sukurti greitesnio matematinių išraiškų pertvarkymo algoritmo nei jis yra profesionalioje sistemoje Maple.

4.3.4. Rezultatų gerinimas

Žinant, kad buvo siekiama padaryti specializuotą matematinių išraiškų pertvarkymo algoritmą, dar galima tikėtis jo pagreitėjimo. Apie išlygiagretinimo algoritmą negalime nieko spręsti, nes kol kas neturime jo su kuo jį lyginti. Išlygiagretinimo greitinimui reikia projektuoti naujus algoritmus arba modifikuoti esamą algoritmą. Modifikavimo ar kūrimo metu būtina kartoti atliktus tyrimus, kad galima būti objektyviai vertinti modifikacijų įtaką algoritmo greičiui. Teisingos išlygiagretinimo algoritmo idėjos pasirinkimui gerai būtų įvykdyti atskirų algoritmo dalių tyrimą, t. y. kiek trunka subišraiškų atskyrimas, kiek trunka duomenų mainai, kiek trunka subišraiškų apjungimas.

5. IŠVADOS

Šiame skyriuje užfiksuojama sukurta sistema, pakartojami pagrindiniai tyrimų rezultatai, jų pagrindu padaromos ir išdėstomos darbo išvados. Skyriaus gale apibrėžiamos galimos produkto plėtros kryptys, pateikiamos rekomendacijos naudotojams.

5.1. Sukurta sistema

Per 16 mėnesių buvo sukurta ir pilnai dokumentuota matematinių išraiškų pertvarkymo programinė įranga. Sistema skirta matematinių išraiškų pertvarkymui lygiagrečiai naudojant kelis vieno procesoriaus kompiuterius, sujungtus į vietinį tinklą. Šiuo metu sistema pritaikyta matematinių išraiškų diferencijavimui.

5.2. Pagrindiniai tyrimų rezultatai

Tyrimų metu nustatėme, kad sukurta sistema yra itin efektyvi, kai sistemoje naudojami bent 3 procesoriai. Sistemos efektyvumas yra mažesnis esant 2 procesoriams, nes naudojams tik vienas vykdantysis procesas antrajame procesoriuje. Matematinių išraiškų ilgis pertvarkymų greitį įtakoja beveik tiesiškai proporcingai.

5.3. Plėtros kryptys

Darbo tiksluose buvo numatytas susiaurintas funkcionalumas. Dėl to lieka nemenka erdvė funkcionalumui plėsti, t. y. diferencijavimo funkcijų papildymas ir kitokie pertvarkymai, pvz. integravimas. Kadangi sistema yra komplikuota ir sudėtinga, liko erdvės algoritmams optimizuoti, greičiui bei patikimumui didinti.

Produktas neturi patogios grafinės vartotojo sąsajos, todėl sekančiose versijose galima būtų ją sukurti pagal sistemos naudotojų rekomendacijas.

5.4. Algoritmų tobulinimas

Kalbant apie matematinių išraiškų pertvarkymo išlygiagretinimo algoritmo tobulinimą, sistemos kūrėjams reikia detaliai ištirti atskiras algoritmo dalis, kad nustatyti kiek trunka subišraiškų atskyrimas, duomenų perdavimas ir priėmimas, subišraiškų apjungimas. To pagrindu reikia suprojektuoti teisingą algoritmo karkasą.

Būtų verta išbandyti ir kitas veiksmų išlygiagretinimo technikas: rekursinę, prioritetinę ir pan., kitas matematinių išraiškų dalijimo metodikas. Taip pat būtų verta pabandyti kitokią matematinių išraiškų duomenų struktūrą: MIPS modulyje naudotą operacijų medį.

5.5. Rekomendacijos

Žemiau lentelės forma pateiktos rekomendacijos sistemos naudotojams kada kurią sistemą naudoti.

lentelė 26. Kada kurią sistemą naudoti

Matematinės išraiškos ilgis		Naudojamų procesorių kiekis				
nuo	**iki**	**1**	**2**	**3**	**4**	**5**
0	100.000	Maple	MIPS	MIPS	MIPS	MIPS
100.000	8.500.000	Maple	Maple	MIPS	MIPS	MIPS
8.500.000	23.000.000	Maple	Maple	Maple	MIPS	MIPS
23.000.000	∞	Maple	Maple	MIPS	MIPS	MIPS

6. LITERATŪRA

Šiame skyriuje pateikiamas literatūros sąrašas, nurodomi papildomai naudoti dokumentai.

6.1. Literatūra

[1] Problem Solving Environments [interaktyvus]. – [žiūrėta 2002 10 24]. Prieiga per internetą: <www-cgi.cs.purdue.edu/cgi-bin/acc/pses.cgi>

[2] Distributed Maple [interaktyvus]. – [žiūrėta 2002-10-17]. Prieiga per internetą: <http://www.risc.uni-linz.ac.at/software/distmaple>

[3] Maple [interaktyvus]. – [žiūrėta 2002-10-10]. Prieiga per internetą: <http://www.maplesoft.com/products/maple7>

[4] The Maple Reporter. 2001.

[5] Ross S. S. Maple 6: The Analytical Engine. 2000.

[6] Message Passing Interface Forum: Document for a Standard Message-Passing Paradigm. – 1993.

[7] MPI forum [interaktyvus]. – [žiūrėta 2002-10-10]. Prieiga per internetą: <http://www.mpi-forum.org>

[8] Introduction to Parallel Computing: Design and Analysis of Algorithms / V. Kumar, A. Grama, A. Gupta, G. Karypis. – Redwood City, CA, USA: Benjamin/Cummings, 1994.

[9] Solving Problems on Concurrent Procesors: Software for Concurrent Processors. – Volume II. I. Angus, G. C. Fox, J. Kim, D. Walker – Englewood Cliffs, NJ, USA: Prentice-Hall, 1990.

[10] Akl S. G. The Design and analysis of Parallel Algorithms. – Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[11] Andrews G. R. Concurrent Programing: Principles and Practice. – Redwood City, CA, USA: Benjamin/Cummings, 1991.

[12] Carey G. F. Parallel Supercomputing: Methods, Algorithms and Applications. – New York, NY, USA: Wiley, 1989.

[13] Messina P., Murli A. Practical Parallel Computing: Status and Prospects. – West Sussex, UK: Wiley, 1991.

[14] Quinn M. J. Parallel Computing: Theory and Practice. – New York, NY, USA: McGraw-Hill, 1987.

[15] Dally W. J. The message-driven processor. – IEEE Micro, 1992.

[16] Bal H. E., Steiner J. G., Tanenbaum A. S. Programming languages for distributed computing systems. – ACM Computing Surveys, 1989.

[17] A user's guide to PVM / J. J. Dongarra, G. A. Geist, R. Manchek, V. S. Sunderam. – Technical Report 37831-6367. – Oak Ridge, TN, USA: Oak Ridge National Laboratory, 1991.

[18] Gehani N. H., Roome W. D. The Concurrent C Programming Language. – Summit, NJ, USA: Silicon Press, 1988.

[19] Hoare C. A. R. Communciating Sequential Processes. – Cambrige, MA, USA: Prentice-Hall, 1985.

[20] Polycronopoulos C. D. Parallel Programming and Compilers. – Boston, MA, USA: Kluwer Academic Publishers, 1988.

6.2. Naudoti dokumentai

lentelė 27. Naudoti dokumentai

Dokumentas	Data	Dokumento autorius
Projekto paraiška	2002.11.27	A. Vaškevičienė, V. Vaškevičius
Reikalavimų specifikacija	2003.03.17	A. Vaškevičienė, V. Vaškevičius
Architektūros specifikacija	2003.04.15	A. Vaškevičienė, V. Vaškevičius
Detalios architektūros specifikacija	2003.05.23	A. Vaškevičienė, V. Vaškevičius
Testavimo planas	2003.09.28	A. Vaškevičienė, V. Vaškevičius
Vartotojo dokumentacija	2003.12.11	A. Vaškevičienė, V. Vaškevičius
Dokumentacija	2004.01.05	A. Vaškevičienė, V. Vaškevičius

7. SANTRUMPŲ IR TERMINŲ ŽODYNAI

Šiame skyriuje atskirai pateikiami santrumpų ir terminų žodynai.

7.1. Santrumpos

lentelė 28. Santrumpos

Santrumpa	Apibūdinimas
MIPS	Matematinių išraiškų pertvarkymo sistema
LFVS	Lygiagretaus funkcionavimo valdymo sistema
MPI	Message Passing Interface – pranešimų perdavimo sąsaja
Maple	Universalus simbolinės matematikos produktas (gamintojas – Waterloo Maple, Inc). Naudojama Maple 7 versija.
Mathcad	Universalus simbolinės matematikos produktas (gamintojas – MathSoft). Naudojama Mathcad 2000 Professional.

7.2. Terminai

lentelė 29. Terminai

Terminas	Terminas anglų kalba (English)	Apibūdinimas
Matematinė išraiška	Mathematical expression	Matematinis reiškinys sudarytas iš skaičių, kintamųjų, aritmetinių, trigonometrinių, logaritminių funkcijų
Diferencijavimas	Differentiation	Simbolinio diferencialo (išvestinės) nustatymas
Nuoseklieji algoritmas	Sequential algorithm	Standartinis algoritmas, vykdomas viename procese
Lygiagretieji algoritmas	Parallel algorithm	Algoritmas, kurio atskiros dalys lygiagrečiai vienu metu vykdomos atskiruose procesuose
Lygiagretusis funkcionavimas	Parallel operation	Kelių procesų veikimas vienu metu

PRIEDAI

Priedas A. Magistro darbo dokumentas

Dokumento paskirtis

Dokumentas skirtas magistro darbui ir tyrimui aprašyti. Taip pat šis dokumentas yra Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros modulio "Magistro baigiamasis darbas" atsiskaitymo ataskaita.

Dokumento autoriai

Šį magistro darbo dokumentą ruošė Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros IFM-8/2 grupės magistrantas Vytautas Vaškevičius ir docentas dr. Romas Marcinkevičius.

Dokumentą maketavo Vytautas Vaškevičius.

Dokumento istorija

Magistro darbas pradėtas rengti 2004 m. kovo 1 d.

lentelė 30. Dokumento istorija

Data	Versija	Aprašymas	Redaktorius
2004.04.18	1.0	Pirminės dokumento struktūros sudarymas	V. Vaškevičius
2004.05.05	1.1	Pradinis dokumento parengimas	V. Vaškevičius
2004.05.07	1.2	Lietuvių kalbos konsultantės redakcija	J. Mikelionienė
2004.05.10	1.3	Įvairūs pataisymai ir papildymai	V. Vaškevičius
2004.05.12	1.4	Darbo vadovo redakcija	R. Marcinkevičius
2004.05.13	1.5	Įvairūs pataisymai ir papildymai	V. Vaškevičius
2004.05.16	1.6	Tyrimų skyrius	V. Vaškevičius
2004.05.19	1.7	Analizės skyrius	V. Vaškevičius
2004.05.20	1.8	Lietuvių kalbos konsultantės redakcija	J. Mikelionienė
2004.05.21	1.9	Darbo vadovo redakcija	R. Marcinkevičius
2004.05.24	2.0	Galutinė redakcija	V. Vaškevičius