KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
KOMPIUTERINIŲ TINKLŲ KATEDRA

Linas Daugirdas

Grafikos generavimo debesų kompiuterijoje informacinė sistema

Magistro darbas

Darbo vadovas: doc. Gytis Vilutis

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
KOMPIUTERINIŲ TINKLŲ KATEDRA

Linas Daugirdas

Grafikos generavimo debesų kompiuterijoje informacinė sistema

Magistro darbas

Recenzentas: lekt. dr. Algirdas Noreika

2011-05-24

Darbo vadovas: doc. dr. Gytis Vilutis

2011-05-24

Atliko: IFM-9/4 gr. stud. Linas Daugirdas

2011-05-24

KAUNAS, 2011

Summary

Information System for Graphics Rendering in Cloud Computing

Graphics rendering is a complex task which needs a great number of computation resources. Therefore rendering systems (farms) are being developed and used by designers and animators. Because of non-consistent and dynamically changing workload it's hard to maintain rendering systems fully loaded. So the downtime of such systems is an issue which leads to loss of profit.

In this work using of cloud computing services is analysed as a solution to handle resources utilisation problem in graphics rendering systems. A complex of task was implemented to do this research. Firstly analysis of existent workload balancer models and graphics rendering systems which uses cloud computing was made. Secondly the information system for graphics rendering was created and resource distribution as a hybrid IaaS cloud was implemented. Algorithm of load balancing between private and public clouds was presented and experimentally tested in this system. Also different job scheduling strategies with ability to predict number of incoming jobs and reserve resources for them were suggested. To determine virtualisation influence to rendering speed an experiment of different resources distribution scenarios in network was made. As a result showed using of this complex solution can raise private resources utilisation level and lower usage of public cloud.

Terminų ir santrumpų žodynėlis

Amazon EC2 (angl. *Amazon Elastic Compute Cloud*) - pagrindinė Amazon debesies infrastruktūros dalis ir ją valdanti sąsaja.

Amazon S3 (angl. *Amazon Simple Storage Service*) - Amazon debesų kompiuterijos duomenų saugykla.

AMI (angl. *Amazon Machine Image*) - programinė įranga skirta kurti ir valdyti virtualias mašinas „Amazon Elastic Compute Cloud“ aplinkoje.

API (angl. *Application Programing Interface*) - aplikacijų programavimo sąsaja, kuri skirta palengvinti programų tarpusavio komunikavimą.

CC (angl. *Cluster Controller*) - Eucalyptus programinės įrangos dalis atsakinga už kompiuterinių blokų valdymą.

CL (angl. *Cloud Controller*) - Eucalyptus programinės įrangos dalis atsakinga už debesies valdymą.

CLI (angl. *Command-line interface*) - tekstinėmis komandomis grindžiama kompiuterio valdymo sąsaja.

Debesų kompiuterija - terminas apibrėžiantis kompiuterinių resursų, kasdieninių paslaugų ir taikomųjų programų visumą internete.

Front-end - Eucalyptus sistemoje suformuotas privatų debesį valdantis serveris su Eucalyptus CC ir CL programine įranga.

GNU GPL (angl. *GNU General Public License*) - linux skirtai programinei įrangai atviro kodo licencija. Šios licencijos programinę įrangą galima laisvai koreguoti ir taikyti savo reikmėms.

IaaS (angl. *Infrastructure as a service*) – procesoriaus, operatyvinės atminties, kietojo disko ir kompiuterinio tinklo resursų tiekimas vartotojui kaip paslauga

KVM - programinė įranga skirta virtualių mašinų diegimui ir eksploatavimui fiziniuose serveriuose.

NC - (angl. *Node Controller*) - Eucalyptus sistemos dalis, kuri suteikia galimybę paleisti norimą kiekį virtualių mašinų skaičiavimo serveryje.

OCCI (angl. *Open Cloud Computing Interface*) - atviras užduočių valdymo protokolas skirtas IaaS, Paas ir SaaS debesies kompiuterijai.

OGF (angl. *Open Grid Forum*) - bendruomenė, kurios tikslas tirti, kurti ir naudoti taikomąsias paskirstyto skaičiavimo šakas.

PaaS (angl. *Platform as a service*) – paslauga, kuri leidžia vartotojui naudotis programinės įrangos kūrimo įrankiais programoms kurti, bei jas patalpinti nutolusioje aplinkoje.

REST (angl. *Representational State Transfer*) - programinės įrangos architektūros stilius skirtas paskirstytai informacijai. Architektūra grindžiama kliento ir serverio veikimu.

SaaS (angl. *Software as a service*) – programinė įranga vartotojui tiekiama, kaip paslauga.

SOAP (angl. *Simple Object Access Protocol*) - protokolas skirtas struktūrizuotos informacijos apsikeitimui kompiuteriniuose tinkluose ir internetiniuose servisuose.

Virtualizacija - galimybė fiziniuose resursuose suformuoti ir paleisti veikti virtualius skaičiavimo vienetus - virtualias mašinas.

Virtuali mašina - iš fizinių resursų sudarytas kompiuterinis resursas, kuris turi suformuotą virtualią atmintį, procesoriaus ir operatyviosios atminties dydžius. Vartotojas dirbdamas su tokiu resursu nejaučia realaus skirtumo nuo fizinių resursų.

XEN - programinė įranga skirta virtualių mašinų diegimui ir eksploatavimui fiziniuose serveriuose.

Turinys

1. Įvadas

Kompiuteriu generuojamas (angl. *Rendering*) trimatis ir dvimatis vaizdas tapo vienu iš pagrindinių multimedijos uždavinių (pvz.: animacija kine ir žaidimuose, projektų vizualizacijos) [1,2]. Šis procesas skirstomas į iš anksto sugeneruojamą (angl. *Pre-rendered*) ir realiu laiku generuojamą (angl. *Real-time rendering*) vaizdą. Abiejų uždavinių įgyvendinimo laikas tiesiogiai priklauso nuo kompiuterio techninės įrangos spartos, nes reikalauja sudėtingų skaičiavimų ir didelių duomenų srautų. Realiu laiku generuojamas vaizdas daugumoje atvejų realizuojamas nuosekliai viename kompiuteryje. O iš anksto generuojamos grafikos uždaviniai gali būti išskaidyti keliems kompiuteriams ir skaičiuojami lygiagrečiai. Šių lygiagrečiai atliekamų uždavinių rezultatų failai apjungiami į galutinį produktą. Tačiau pastarasis atvejis reikalauja didelio kiekio techninės įrangos - tiek skaičiavimo kompiuterių, tiek ir tinklo resursų. Problema kyla, kai dėl nepastovaus uždavinių srauto atsiranda dideli laiko intervalai tarp generuojamų vaizdo projektų. Esant tokioms prastovoms ši įranga yra neišnaudojama ir į ją investuoti finansiniai kaštai neatsiperka. Šiai problemai spręsti kuriamos vaizdų generavimo sistemos, kurios leidžia vartotojams gauti tokią paslaugą nenaudojant savų resursų ir mokėti tik už tai kiek ja naudojamasi. Tokios sistemos patrauklios vartotojams, kurie vaizdo generavimu užsiima retai ir jiems nenaudinga turėti savos įrangos. Tačiau kompiuterinio vaizdo generavimo sistemų nuolat kintančio apkrovimo ir nenašaus resursų išnaudojimo problemos persikelia iš individualių vartotojų į tokias paslaugas teikiančias įmones.

Pastaruoju metu programiniuose ir internetiniuose sprendimuose sparčiai plinta debesies kompiuterijos technologijos (angl. *cloud computing*). Viena iš šios, kompiuterinių resursų panaudojimą, jų buvimo vietą ir paslaugų teikimo būdą nusakančios, paradigmos galimybė - kompiuterinių resursų nuoma pagal pageidavimą ir apmokėjimas tik už naudotą laiką. Egzistuoja trys pagrindiniai debesų kompiuterijos tipai [3]:

- Privatus debesis (ang. *Private cloud*) - resursų panaudojimo scenarijus, kai turimi įmonės kompiuteriniai ištekliai apjungiami į vieningą, valdomą infrastruktūrą, kurioje vietoj fizinių kompiuterinių vienetų naudojamos virtualios mašinos.
- Viešas debesis (ang. *Public cloud*) - kompiuteriniai resursai sujungti į vieningą infrastruktūrą paslaugos tiekėjo, kuris leidžia nuomotis reikiamą kiekį ir tipą virtualių mašinų. Ši paslauga naudinga esant nenuolatiniam, bet dideliam resursų poreikiui.
- Hibridinis debesis (ang. *Hybrid cloud*) - kompiuterinių resursų išdėstymas, kai dalis jų yra apjungti į privatų debesį, o likusi dalis pagal poreikį nuomojama iš viešo debesies tiekėjo. Toks scenarijus pasirenkamas, kai turimas nuolatinis naudotojų srautas, tačiau tam tikrais laiko momentais jis gali padidėti.

Dėl greito reikalingų resursų kiekio keitimo bei mokėjimo tik už naudojimosi laiką debesies technologijos yra vienas iš galimų, dinaminio resursų poreikio problemos, sprendimo būdų. Tačiau susiduriama su problema - kada, kiek ir kaip resursų nuomoti iš viešo debesies tiekėjo. Šiai problemai spręsti kuriami apkrovimo balansavimo modeliai [4,5], tačiau jie nėra universalūs ir priklauso nuo naudojamų resursų tipo, taikomų uždavinių pobūdžio ir vartotojų kiekio. Todėl kiekvienu atveju reikalingas savitas sprendimas.

Vaizdų generavimo sistemos resursų išnaudojimo ir apkrovimo problemai spręsti darbe projektuojama ir kuriama prototipinė kompiuterinių vaizdų generavimo paslaugą teikianti informacinė sistema. Ši sistema naudoja skaičiavimo serverius apjungtus į privatų debesį. Tačiau esant dideliam apkrovimui, panaudojant IaaS (angl. Infrastructure as a service) debesų kompiuterijos principą, gali skaičiavimams prijungti nuomojamus resursus iš viešo debesies paslaugų tiekėjo (pvz.: „Amazon“). Taigi pati sistema tuo pačiu metu yra ir tiekėjas ir vartotojas, realiu laiku dinamiškai ir automatizuotai pagal poreikį naudojantis tiekėjų paslaugas.

Darbe analizuojamas ir kompiuterinio vaizdo generavimo informacinėje sistemoje taikomas uždavinių paskirstymo algoritmas. Jis leidžia sumažinti viešame debesyje naudojamų virtualių mašinų kiekį ir veikimo laiką. Apkrovimo balansavimo modelis buvo pristatytas „Matematika ir matematikos dėstymas - 2011“ konferencijoje. Pranešimo pavadinimas - „Cloud sistemose teikiamų resursų krūvio valdymo modelis“ (autoriai: Gytis Vilutis, Kristina Šutienė, Linas Daugirdas, Martynas Vaidelys). Šio pranešimo straipsnis pateikiamas priede nr. 1.

Darbo struktūra:

- antrajame skyriuje atlikta debesies kompiuterijos apžvalga, esamų sprendimų internetinėje erdvėje ir literatūros šaltiniuose analizė. Taip pat parinkta tinkamiausia debesies kompiuterijos architektūra.
- trečiajame skyriuje sudaryti kompiuterinės grafikos generavimo informacinės sistemos reikalavimai.
- ketvirtajame skyriuje pateiktas kompiuterinės grafikos generavimo informacinės sistemos projektas, apibrėžiantis sistemos esybes, duomenų bazę, diegimo modelį ir sistemos veiklos taisykles.
- penktajame skyriuje aprašyta kompiuterinės grafikos generavimo informacinės sistemos ir debesies kompiuterijos realizacija.
- šeštajame skyriuje atlikti debesų kompiuterijos įtakos vaizdų generavimui ir sistemoje generuojamų uždavinių srauto valdymo eksperimentai.
- septintajame skyriuje pateikiamos išvados.

2. Debesų kompiuterijos taikymo grafikos generavimo IS analizė

Siekiant sukurti novatorišką, pažangiai veikiančią grafikos generavimo informacinę sistemą atlikta esamų sprendimų, literatūros šaltinių, vartotojų ir įgyvendinimo priemonių analizė. Prieš tai, pagal analizuojamus objektų tipus, pasirinkta analizės metodika.

2.1.Analizės tikslas

Išsiaiškinti debesų kompiuterijos taikymo galimybes kuriant prototipinę vaizdo generavimo sistemą. Surasti ir palyginti šiuo metu realizuotų grafikos generavimo sistemų, naudojančių debesies paslaugas, privalumus ir trūkumus. Palyginti literatūros šaltiniuose siūlomus apkrovimo balansavimo metodus. Parinkti tinkamiausią debesies programinę įrangą šios sistemos kūrimui.

Analizei pasirinktas lyginamosios analizės principas. Analizės procesas detalizuojamas tokiais žingsniais:

1. Apibrėžiamas palyginimo kontekstas;
2. Atrenkami lygintini sprendimai ir jie analizuojami;
3. Nustatomos esminės lyginamų sprendimų savybės.

2.2.Tyrimo sritis, objektas ir problema

Tyrimo sritis – debesų kompiuterijos technologijų panaudojimo galimybės IS kūrime, kuri skirta vaizdų generavimo paslaugoms teikti.

Tyrimo objektas – debesų kompiuterijos IaaS ir SaaS (angl. *Software as a service*) principais veikianti sistema skirta generuoti vaizdus Blender programinės įrangos aplinkoje.

Problema kyla dideliame vaizdo generavimo resursų poreikyje [1,2]. Pavyzdžiui, sugeneruoti DreamWorks kino studijos kurtam animaciniam filmui „ShrektheThird“ reikėjo 20 milijonų kompiuterio darbo valandų [6]. Šis procesas animacinius vaizdus kuriančiose įmonėse dažnai skaidomas į lygiagrečius skaičiavimus keletui ar net keliems šimtams skaičiavimo serverių. Tokiu būdu išsprendžiamas generavimo laiko sumažinimo klausimas, nes jo sutrumpėjimo kartų kiekis lygus prijungiamų resursų kiekiui, tai galime matyti iš „Shandong“ universitete atliktų eksperimentų [7] bei [1] šaltinio. Tačiau pastarasis sprendimo būdas neefektyvus, jeigu vaizdo generavimo procesai yra reti arba naudojami nedidelėje įmonėje mažiems projektams. Dėl to kuriamos IS, teikiančios vaizdų generavimo paslaugą su galimybe vartotojui mokėti tiktai už naudojamus resursus arba laiką. Tokiose sistemose yra sunku numatyti kiek resursų tikslinga turėti, nes norint aptarnauti visus naudotojus reikalingi dideli

skaičiavimo mašinų kiekiai. Šiuo metu egzistuoja grafikos generavimo sistemos, kurios resursų išnaudojimo klausimą sprendžia naudojant debesų kompiuterijos paslaugas. Tačiau čia ir kyla **pagrindinė problema** - reikia nuspręsti kada ir kiek resursų nuomotis iš viešo debesies tiekėjo. Kaip valdyti uždavinių srautą ir naudojamų resursų kiekį, kad privačių resursų užimtumas būtų maksimalus, o viešo debesies naudojimas nebūtų nuostolingas.

2.3.Vartotojų analizė

Norint sudaryti sistemos funkcinius reikalavimus reikia iš anksto ištirti potencialių vartotojų aibę. Taip pat, siekiant realizuoti populiarią ir funkcionalią sistemą, reikia iš anksto numatyti vartotojų tikslus ir problemas.

2.3.1. Vartotojų aibė ir tipai

Kompiuterinio vaizdo generavimo sistemos vartotojai – Blender kompiuterinės grafikos generavimo programos vartotojai. Jie gali būti kelių tipų :

1. Įmonės apsiimančios retais kompiuterinės animacijos ar vizualizacijų kūrimo projektais, kurioms neapsimoka dėl to investuoti didelių kaštų į kompiuterinę techniką, kuri nebus pilnai išnaudojama ir taip sumažins projektų atsiperkamumą.
2. Įmonės, kurios turi pakankamai skaičiavimo resursu nuolatiniams darbams, tačiau susiklosčius nepalankioms aplinkybėms gali nespėti sugeneruoti grafiką iki terminų pabaigos.
3. Pavieniai asmenys - animatoriai, kuriantys animaciją tačiau neturintys pakankamai galingų skaičiavimo resursų.
4. Universiteto dėstytojai ir studentai, kuriems ši paslauga reikalinga sistemingai visiems vienu metu, sesijos eigoje prieš atsiskaitymų datas. Tokiu metu gali neužtekti turimų resursų visiem projektams sugeneruoti laiku.

2.3.2. Vartotojų tikslai ir problemos

Pagrindinis vartotojų tikslas – greitas galutinis kompiuterinės animacijos projekto vaizdas. Antrinis tikslas – noras turėti generavimo paslaugą, nesirūpinant technine įranga ir jos palaikymo užtikrinimu. Taip pat vartotojui svarbu pačiam pasirinkti priimtiniausią apmokėjimo planą.

Vartotojų problemos pateikiamos 2.1 lentelėje.

2.1 lentelė. Vartotojų problemos

Problema	Kaip viskas vyksta dabar
Kompiuterinės grafikos generavimo skaičiavimai užima labai daug laiko.	Kompiuterinė grafika generuojama naudojant vieną kompiuterį.
Techniniai resursai reikalingi paskirstytam kompiuterinės grafikos generavimui yra brangūs.	Įmonės perka savus kompiuterius ar darbines stotis, kuriuose realizuoja paskirstytą skaičiavimą, tačiau ši technika didelį laiko tarpsnį būna neišnaudojama.
Techninės ir programinės įrangos gedimai gali sutrikdyti vykdomus projektus pabaigti laiku.	Įmonė ar fiziniai asmenys kompiuterinio vaizdo generavimui naudoja savo nuosavą kompiuterinę ir programinę įrangą, kuri pastoviai reikalauja priežiūros, atnaujinimų ir palaikymo.
Sunku palaikyti reikalingą skaičiavimo resursų kiekį, todėl jie pilnai neišnaudojami.	Skaičiavimo serveriai esant mažam sistemos apkrovimui lieka neišnaudoti ir neneša finansinės naudos.

2.4.Tyrimo objekto analizė

Tiriant debesies kompiuterijos panaudojimo galimybes buvo apžvelgti debesies teikiamų paslaugų tipai. Taip pat atlikta šiuo metu esančių, alternatyvias paslaugas teikiančių, sistemų analizė. Pagal surinktus duomenis pasirinkta labiausiai skaičiavimo uždavinių paskirstymui kompiuteriniams resursams tinkanti debesų kompiuterijos architektūra.

2.4.1. Debesų kompiuterijos apžvalga

Debesų kompiuterija – paskutiniu metu išpopuliarėjęs terminas apibrėžiantis kompiuteriu sprendžiamų uždavinių perkėlimą iš vartotojo kompiuterio į nutolusią skaičiavimo mašinų sistemą. Vartotojui nebereikia rūpintis duomenų saugumu, techninės įrangos nestabilumu ar jos kiekiu. Visi atliekami veiksmai vartotojui pateikiami kaip paslauga. Šiuo metu skiriamos trys pagrindinės debesų kompiuterijos paslaugos:

- „Infrastructure as a service“ – pirmoji debesų kompiuterijos teikiama paslauga. Šis terminas apibrėžia techninės įrangos – procesoriaus, operatyvinės atminties, kietojo disko ir kompiuterinio tinklo resursų tiekimą vartotojui kaip paslaugą [8,9]. Šie resursai gali būti tiek virtualios mašinos, tiek ir dedikuotos tarnybinės stotys. Tačiau išlaikoma pagrindinė idėja – mokama panaudos principu tik už naudojimosi laiką, duomenų srautą ir nuomotų resursų apimtį. Toks debesies modelis gali būti taikomas sistemoms, kurios reikalauja kintančio resursų skaičiaus.

- „Platform as a service“ – antroji debesų kompiuterijos teikiama paslauga, kuri leidžia vartotojui naudotis programinės įrangos kūrimo įrankiais nesudėtingoms programoms kurti, bei jas patalpinti nutolusioje aplinkoje. Šios sukurtos programos vartotojams yra pasiekiamos per internetinę sąsają. Paslaugos tiekėjai užtikrina nenutrūkstamą sukurtos programinės įrangos veiklą ir jos išplečiamumą pagal kintantį vartotojų poreikį. Tačiau šiuo metu nėra nusistovėjusių bendrų standartų tarp tiekėjų, todėl prisirišama prie to paties tiekėjo.
- „Software as a service“ – trečioji debesų kompiuterijos teikiama paslauga. Šios paradigmos pagrindinis principas – programinė įranga vartotojui tiekiama kaip paslauga, tačiau visi procesai atliekami nutolusioje erdvėje esančiuose serveriuose. Šiuo atveju skaičiavimai ir duomenų apdorojimas paskirstytas IaaS modelyje. Tokiu būdu vartotojui nebereikia turėti įdiegtos programinės įrangos savame kompiuteryje [10,3], jis gali pasinaudoti ir mokėti tik už tai, kiek laiko naudojosi. IaaS paslaugos atveju visada užtikrinama duomenų apsauga, daromos atsarginės kopijos ir paslaugos teikiamos 24/7 principu [11].

Apžvelgus debesų kompiuterijos teikiamas paslaugas ir įvertinus, jog sistemos funkcionavimui reikės kompiuterinių resursų, nustatyta, kad kuriamos IS architektūrai yra tinkama IaaS architektūra. Toliau analizuojami literatūros šaltiniai, kurie sprendžia resursų paskirstymo debesyse problemą, bei esamos sistemos, kurios naudoja šią debesies technologiją kompiuteriniam vaizdui generuoti. Taip pat atliekama galimų įgyvendinimo priemonių analizė pagal kurią pasirinkta tinkamiausia nemokama IaaS debesies programinė įranga.

2.4.2. Esamų sistemų analizė

Debesies terminas sparčiai plinta sistemose, kurios naudoja didelius skaičiavimo resursų kiekius, ne išimtis ir grafikos generavimas. Šiuo metu rinkoje jau egzistuoja sistemos teikiančios kompiuterinės grafikos generavimo paslaugas, kurios skiriasi savo įgyvendinimo būdais ir paslaugų teikimo principais. Apžvalgai atrinktos populiariausios kompiuterinės grafikos generavimo debesies kompiuterijos principais besiremiančios BevyRender, BlenderCloud, CloudFuzion, Felix, RenderCore ir RenderRocket sistemos, kurios analizuojamos pagal sekančius kriterijus:

- Ar sistema naudoja privatų debesį,
- Ar sistema naudoja viešą debesį,
- Ar esant poreikiui yra galimybė išplėsti turimus skaičiavimo resursus viešame debesyje,
- Ar naudojama eilės arba grafikos generavimo optimizacija,

- Ar atsižvelgiama į vartotojų srauto istoriją ir pagal tai atliekamas prognozavimas,
- Kaip ši paslauga tiekiama vartotojui.

Sistemų tarpusavio palyginimo duomenys, kurie paimti ir susisteminti iš oficialių sistemų internetinių svetainių[12,13,14,15,16,17], pateikiami 2.2 lentelėje. Privataus debesies naudojimas sudaro galimybę vykdomiems uždaviniams dinamiškai priskirti tiek virtualius, tiek ir fizinius resursus. CloudFuzion programinė įranga leidžia savo turimus resursus valdyti kaip privatų grafikos generavimo debesį. RenderCore, Felix ir RenderRocket taip pat naudoja privačius resursus grafikos generavimui, tačiau šios sistemos, skirtingai negu CloudFuzion, tai teikia kaip paslaugą. Šią galimybę, bet su apribojimais, turi ir BevyRender sistema. Šiuo metu joje siūlomas nemokamas iki 50 kadrų projektų generavimas, kuris atliekamas privačiame Eucalyptus debesyje.

Viešo debesies naudojimas leidžia dinamiškai praplėsti ir sumažinti skolinamų resursų skaičių, tokiu būdų sudaroma galimybė mokėti tik už tai kiek naudotasi ir tik tada kada reikia. Viešame Amazon EC2 (angl. *Amazon Elastic Compute Cloud*) debesyje skaičiavimus atlikti turi galimybę BevyRender, CloudFuzion ir BlenderCloud sistemos. RenderRocket, savo ruožtu, išorinio debesies tiekėjo neatskleidžia, tačiau esant poreikiui irgi gali išplėsti virtualių resursų skaičių. Reiktų pabrėžti, kad BevyRender leidžia įsigyti jų sukurtą Amazon EC2 virtualios mašinos atvaizdą su paruošta grafikos generavimo įranga. Ko pasekoje galima grafika generuoti ne per BevyRender internetinę sistemą, o išsinuomoti iš Amazon kompanijos reikiamą resursų kiekį ir skaičiavimus atlikti tiesiogiai juose, taip mokant tiktai Amazon kompanijai.

Automatinio resursų išplėtimo ir sumažinimo debesyje pagal sistemos vartotojų skaičių galimybę turi CloudFuzion ir RenderRocket sistemos. Tokia sistemos galimybė yra privaloma kokybiškam kintamo vartotojų skaičiaus aptarnavimui. Apie savo automatizuotą veikimą Felix sistema informacijos nepateikia.

Grafikos projektų generavimui naudojama įvairi optimizacija. BevyRender kiekvieną kadrą daliną į mažesnes dalis. Kiekviena kadro dalis generuojama skirtingame kompiuteryje, tokiu būdu naudojamas mažesnis buferis ir kadro dydis optimizuoja generavimą. Taip pat šioje sistemoje naudojama prioritetinė eilė. BlenderCloud eilės apdorojimui ir skaičiavimų paskirstymui naudoja DrQueue programinę įrangą. CloudFuzion skelbia jog naudoja algoritmus leidžiančius išnaudoti 100% procesorių pajėgumo, o generavimas vykdomas pagal prioritetinę eilę. Eilės prioritetus taip pat naudoja RenderCore ir RenderRocket sistemos, tačiau pastaroji geba automatiškai parinkti reikalingą papildomą resursų kiekį.

Per Web sąsają paslaugas teikiančių sistemų pranašumas yra galimybė teikti paslaugas nereikalaujant papildomos įrangos vartotojo kompiuteryje, taip pat nereikalingas nuolatinis

vartotojo resursų veikimas. CloudFuzion ir Felix sistemos veikia kaip programinė įranga vartotojo kompiuteryje ir jos skirtos privačiam naudojimui, o ne paslaugų tiekimui vartotojams. Likusios sistemos tiekia vaizdo generavimo paslaugą nuotoliniu būdu, kaip Web servisą. RenderCore ir RenderRocket sistemose yra galimybė pateikti ir parsisiųsti duomenis per FTP aplinką.

Ateinančio srauto prognozavimas yra svarbi funkcija sistemoms, kurios kartu naudoja ir privatų, ir viešą debesį uždavinių skaičiavimui, o paslaugas teikia Web serviso pagrindu. Pavyzdžiui, kai sistemoje skaičiuojami skirtingų prioritetų uždaviniai, iš kurių vieni turi būti pradėti generuoti iš karto, sistema, esant pikui, tokių uždavinių iš karto aptarnauti nesugeba, nes išorinio debesies prijungimas užima tam tikrą laiko tarpą. Todėl sistema turėtų atsižvelgti į užduočių pateikimo srauto istoriją ir numatyti bei rezervuoti privačius resursus skubiems uždaviniams. Šiuo atveju tik RenderRocket sistema veikia automatinio išsiplėtimo hibridiniame debesyje principu ir turi du skirtingus generavimo planus, kurių vienas įpareigoja sistemą pradėti generuoti iš karto. Tačiau ši sistema neturi prognozavimo galimybės.

2.2 lentelė. Esamų sistemų palyginimas

Pavadinimas	**Naudoja privatų debesį**	**Naudoja viešą debesį**	**Išsiplėtimo galimybė**	**Naudojama optimizacija**	**Vartotojo sąsaja**	**Prognozė**
BevyRender	Taip, bet apribotai	Taip, Amazon EC2	Ne	Kadrai skaldomi į mažesnius ir naudojama eilė	Web arba Amazon Ec2	Ne
BlenderCloud	Ne	Taip, Amazon EC2	Ne	DrQueue paskirstytam skaičiavimui ir eilei apdoroti.	Web	Ne
CloudFuzion	Taip	Taip, Amazon EC2	Taip, kintamas debesies resursų dydis	Eilės ir CPU optimizacija	Veikia kaip PĮ.	Ne
Felix	Taip	Ne	Nežinoma	Nežinoma	Veikia kaip PĮ.	Ne
RenderCore	Taip	Ne	Ne	Prioritetai eilėje	Web ir FTP	Ne
RenderRocket	Taip	Taip	Taip, prijungiamas viešas debesis	Prioritetai eilėje, automatinis resursų kiekio parinkimas	Web ir FTP	Ne

Apžvelgus šias sistemas pastebėta, jog yra tik viena sistema teikianti grafikos generavimo paslaugas per internetinę sąsają privačiame debesyje ir gebanti automatiškai prasiplėsti viešame. Tačiau ši sistema neturi galimybės pagal skaičiavimų istoriją prognozuoti būsimą vartotojų srautą, ir taip, iš anksto rezervuoti resursus skubiems uždaviniams.

2.4.3. Problemos sprendimo būdų literatūros šaltiniuose analizė

Prieš kuriant sistemą reikia išnagrinėti uždavinių valdymo ir paskirstymo tarp privataus ir viešo debesies metodus. [4] Darbe sprendžiamas uždavinių su konkrečia pabaiga paskirstymas hibridiniame debesyje siekiant gauti optimalią mokėjimo kainą. Čia modeliuojamos kelios situacijos. Pirmiausiai automatinis parinkimas debesies su optimaliausia kainodara turimiems uždaviniams. Antrame eksperimente pradžioje savaites pateikiama 50 skirtingo ilgio uždavinių, kuriem suskaičiuoti iki ilgiausio pabaigos laiko neužtenka savų resursų. Algoritmas iškart kreipiasi i viešą debesį ir jame suskaičiuoja perteklinius uždavinius. Tačiau nuo vidurio savaitės privataus debesies apkrovimas smarkiai krinta. Toks metodas skirtas iš anksto žinomiem uždaviniams, todėl nebūtų tinkamas grafikos generavimo paslaugų teikimui, kur sistemos vartotojai nuolat pateikinėja užduotis ir jų vykdymas turi būti perskaičiuojamas dažnai.

[18] eksperimente hibridiniame debesyje skaičiuojami vidurkinimo filtrų pritaikymo paveikslėliams uždaviniai. Kiekvienas paveikslėlis turi savo matricos dydį bei ilgiausią vykdymo laiką. Čia lyginami skaičiavimai atliekami viešame, privačiame ir hibridiniame debesyje prieš tai nuotraukas išskaidžius į mažesnes dalis. Algoritmas, kaip ir prieš tai apžvelgtame šaltinyje, vykdo skaičiavimus viešame debesyje, kol visų likusių uždavinių įgyvendinimo laikas tampa pakankamas jų įvykdymui privačiam debesyje. Tačiau čia irgi pasigendama naujų uždavinių pateikimo įvertinimo ir prognozavimo.

[19] ir [20] eksperimentai leidžia spręsti internetinių puslapių ir internetinių programų apkrovimo paskirstymo tarp savų ir nuomotų resursų problemą. Momentais. kai yra daug HTTP užklausų ir privatūs serveriai tampa perkrauti, įjungiami debesyje nuomojami serveriai. Toks sprendimas tinkamas tik sistemoms, kurios teikia Web paslaugas pvz.: elektroninio pašto ar vaizdo transliacijų.

[21] šaltinyje pateikiamas statistika besiremiantis balansavimo modelis, skirtas intensyvius duomenų srautus turinčioms programoms. Šis sprendimas skirtas Hadoop, MapReduce karkasams (angl. Framework), kurie leidžia išlygiagretinti internetinių programų ir puslapių vykdomus skaičiavimus. Modelis taip pat apima užduočių vykdymo prognozavimą, tačiau vėlgi jis nėra universalus ir būtų sunkiai taikomas grafikos generavimo paslaugoms teikti.

Pastebėta, kad balansavimo tarp privataus ir viešo debesies naudojimo metodai priklauso nuo uždavinių tipų ir pobūdžio. Todėl nėra vieno bendro, visur tinkančio metodo ir kuriant informacinę sistemą, skirtą generuoti Blender grafikos projektus hibridiniame debesyje reikia savito balansavimo metodo.

2.4.4. Architektūros ir galimų įgyvendinimo priemonių variantų analizė

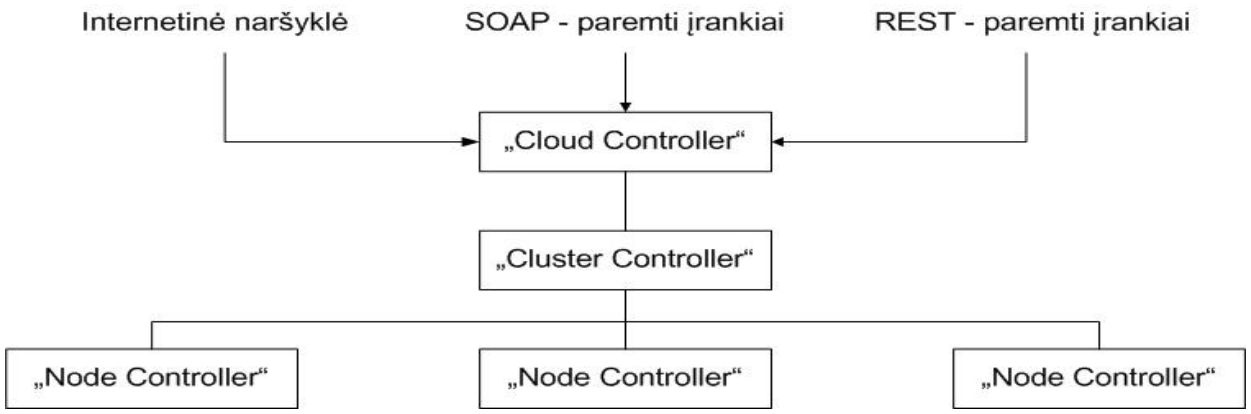
Viena iš populiariausių IaaS paslaugos tiekėjų rinkoje – Amazon. Amazon EC2 vartotojui leidžia susikurti virtualų kompiuterį iš reikiamo resursų kiekio ir mokėti tik už jo naudojimo laiką bei parsiųstą duomenų kiekį [22]. Vartotojui leidžiama pasirinkti tarp Linux ir Windows operacinės sistemos pagrindu veikiančio kompiuterio. Toks įmonės kompiuterinės technikos architektūros sprendimas organizacijai leidžia susikoncentruoti į jų vykdomą veiklą, vietoj papildomo rūpinimosi technikos valdymu, priežiūra, veikimo užtikrinimu ar operacinės sistemos atnaujinimais. Žinoma, IaaS paslaugos pasiekimui reikalinga minimali techninės įrangos dalis ir organizacijos viduje.

Antrasis būdas naudotis IaaS privalumais, tai galimybė panaudoti organizacijoje esančius kompiuterinės įrangos resursus, apjungiant juos į vidinę privataus debesies infrastruktūros aplinką. Šiuo metu populiariausi ir labiausiai paplitę Eucalyptus, OpenNebula, Globus Nimbus atviro kodo projektai [23,24,25]. Jų veikimo principas sąlyginai panašaus.

Eucalyptus (angl. *Elastic Utility Computing Architecture Linking Your Programs To Useful Systems*) – tai atviro kodo programinės įrangos infrastruktūra, skirta iš esamų skaičiavimo kompiuterių blokų (angl. *cluster*) ar paprastų kompiuterių įgyvendinti debesies sistemą. Eucalyptus sukurtas paplitusiais Linux operacinės sistemos įrankiais ir pagrindinėmis web technologijomis, tokiomis kaip – Axis2, Mule, SOAP, REST. Šiuo metu yra išleista 2 versija, kurios pagrindiniai privalumai būtų:

- Sąsajos suderinamumas su Amazon EC2 ir S3, šios sąsajos realizuotos tiek web servisų, tiek ir Query/Rest tekstinių užklausų rėžimais.
- Galimybė išskaidyti pagrindinius Eucalyptus komponentus (atminties, valdymo ir sąsajos) realizuoti atskirose skaičiavimų mašinose.
- Palaikoma didžioji dalis paplitusių Linux distribucijų.
- Vidinei duomenų srautų apsaugai naudojamas SOAP su WS-security protokolu.
- Sistemos valdymui skirti administratoriaus įrankiai patogiai realizuoti web sąsajos pagalba.
- Galimybė lengvai konfigūruoti ir prijungti kitus skaičiavimo mašinų blokus.

Eucalyptus sistemą sudarantys pagrindiniai komponentai atvaizduoti 2.1 paveiksle. Aukščiausiame lygmenyje veikia CL (angl. *Cloud Controler*) ir Warlus duomenų saugykla. CC - tai Java programa teikianti Web, SOAP ir REST įrankius. Apart užklausų, gaunamų iš aplinkos, apdorojimo CL taip pat atlieka resursų paskirstymo ir valdymo veiksmus. CC (angl. *Cluster Controler*) apjungia ir valdo vidiniame tinkle sujungtų kompiuterių ar darbinių stočių grupę. Kiekvienas iš apjungtų kompiuterių yra Eucalyptus sistemos resursas NC (angl. *Node Controler*), kuriame kuriamos virtualios mašinos [26,27].



2.1 pav. Eucalyptus sistemos komponentai [26]

Open Nebula – atviras ir lankstus įrankis, suderinamas su egzistuojančiom duomenų centrų aplinkom, skirtas kurti ir valdyti virtualią infrastruktūrą duomenų centre ar skaičiavimo kompiuterių bloke. Open Nebula taip pat palaiko galimybę kurti ne tik privatų debesį organizacijos infrastruktūroje, bet ir apjungti jį su išorine, viešu debesimi paremta kompiuterinių išteklių infrastruktūra. Stabili ir labiausiai paplitusi Open Nebula 1.2 versija pasižymi privalumais [28]:

- Palaikomos dvi virtualizacijos platformos Xen ir KVM.
- Turi įskiepius skirtus apkrovos metu kreiptis į Amazon EC2 išorinę infrastruktūrą papildomų resursų.
- Palaiko trečiųjų šalių programinės įrangos (pvz.: užduočių ar virtualių mašinų valdymo PĮ (Programinė Įranga) integravimą į Open Nebula aplinką
- Infrastruktūros valdymui ir stebėjimui pateikiamos tiek API (angl. *Application Programming Interface*), tiek ir CLI (angl. *Command-Line Interface*) sąsajos.
- Realizuotos dvi CLI sąsajos – EC2 Query API ir OGF OCCI API.

Nimbus – atviro kodo įrankis, skirtas paversti turimą skaičiavimo kompiuterių bloką į IaaS infrastruktūrą. Nimbus TP 2.2 Versija susideda iš dviejų dalių. Pirmoji tai SN (angl. *Service Node*) – valdančioji programinės įrangos dalis bei antroji VMM (angl. *Virtual Machine Monitor*)

– programinės įrangos dalis skirta kompiuterinių resursų valdymui ir virtualių mašinų kūrimui bei stebėjimui juose. Nimbus infrastruktūros privalumai [29]:

- Palaikomos dvi sąsajos bendravimui su sistema- WSRF ir EC2;
- Lengvas vartotojų administravimas;
- Plačios virtualių mašinų valdymo galimybės;
- Palengvinta integracija su kitokio tipo debesų infrastruktūromis.

Šios debesų kompiuterijos programinės įrangos lyginamos pagal penkis Informacinės sistemos įgyvendinimui svarbius kriterijus:

1. Vartotojo sąsaja. Norint patogiai ir paprastai valdyti debesies infrastruktūrą būtina atsižvelgti į vartotojo sąsajos įvairiapusiškumą ir paprastumą.
2. Galimybė prijungti viešo debesies resursus. Pagal atliktą esamų sprendimų analizę žinome, kad efektyvesniam resursų panaudojimui turi būti galimybė naudotis ne tik turima privačia infrastruktūra, bet ir tam tikrais momentais prijungti išorinių tiekėjų infrastruktūrų resursus. Todėl svarbu atsižvelgti į kito tiekėjo debesies prijungimo galimybę.
3. CLI sąsajos standartas. Kadangi IS veikimo modelyje numatytas resursų praplėtimas iš viešo debesies tiekėjo, tai vartotojo sąsajos bendro standarto buvimas tampa dar vienu svarbiu vertinimo kriterijumi.
4. Programinės įrangos palaikymas. Kol šie projektai yra atviro kodo ir nemokami bei realizuojami atviro kodo operacinėse sistemose, esamo kūrėjų palaikymo klausimas tampa vienas iš svarbiausių, kuris tiesiogiai siejasi su sistemos įgyvendinimo sparta ir patikimumu. Šis kriterijus vertinamas balais nuo 1 iki 5, skiriant 5 balus geriausią palaikymą turinčiai sistemai ir laikant tai etalonu vertinant likusias. Atsižvelgiama į tiesioginį kontaktavimą su kūrėjais, vartotojų bendruomenės aktyvumą, kylančių problemų sprendimo laiką.
5. Programinės įrangos populiarumas ir paplitimas. Plačiai naudojamos sistemos pasirinkimas leidžia sumažinti kilsiančių problemų kiekį dėl suderinamumo bei sutrumpinti jų sprendimo laiką. Šis kriterijus vertinamas balais nuo 1 iki 5. 5 balai skiriami populiariausiai ir labiausiai paplitusiai sistemai, kuri laikoma etalonu vertinant likusias. Atsižvelgiama į šių infrastruktūrų pagrindu realizuotų sistemų praktiką, stebimas paplitimas ir integracija į operacines sistemas bei mokslinių publikacijų kiekį.

Sistemų palyginimui naudoti [23, 30] šaltiniai bei oficialiose svetainėse pateikti duomenys. Rezultatų palyginimui sudaryta 2.3 lentelė.

2.3 lentelė. Nemokamų debesies architektūrų palyginimas

Cloud PĮ	**Vartotojo sąsaja**	**Viešo debesies prijungimas**	**CLI sąsajos standartas**	**Palaikymas**	**Populiarumas**
Eucalyptus	Web, CLI	Taip	EC2 ir Euca	4	5
Nimbus	CLI	Taip	EC2 ir Nimbus	1	2
OpenNebula	CLI	Taip	VEE ir EC2	5	3

2.3 lentelėje pateikta informacija rodo, kad šių sistemų esminės funkcijos yra lygiavertės, tačiau Eucalyptus gerokai lenkia savo populiarumu likusias sistemas tiek mokslinių publikacijų skaičiumi, tiek realizuotų sistemų atžvilgių, o taip pat ši sistema vienintelė integruota į serveriams skirtas Linux distribucijas. Open Nebula savo ruožtu turi geresnį palaikymą, dėl galimo kontaktavimo su kūrėjais ir deklaruojamos vienos darbo dienos atsakymo termino trukmės.

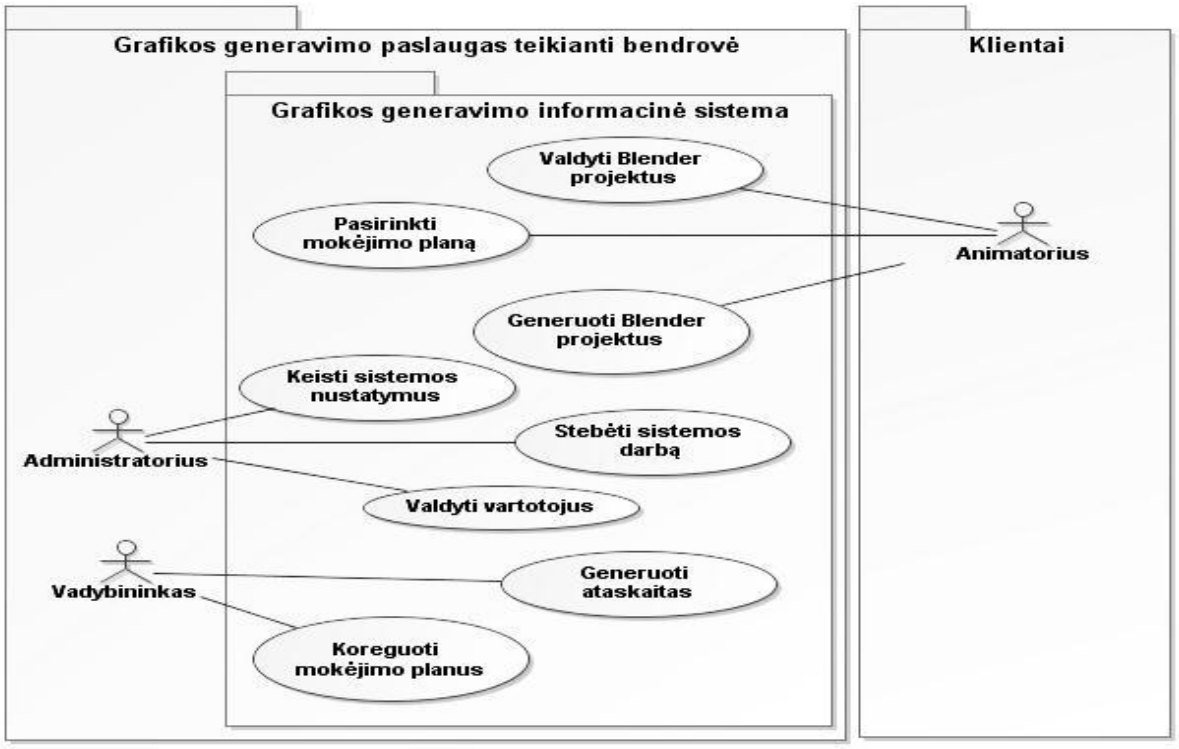
2.5.Siekiamos sistemos apibrėžimas, funkcijos, darbo tikslas ir uždaviniai

Siekiant pagerinti vartotojų galimybes generuoti grafikos projektus, turi būti sukurta kompiuterinės animacijos generavimo informacinė sistema, kuri užtikrintų:

- Spartų kompiuterinės grafikos generavimą,
- Visų pateiktų grafinių projektų generavimą laiku nepaisant jų kiekio,
- Naudojamų resursų apkrovimo stebėjimą,
- Projektų generavimo statistikos stebėjimą ir analizę.

Vartotojų atliekamų veiksmų sistemoje konceptą galima apibendrinti panaudojimo atvejų diagrama, kuri atvaizduojama 2.2 paveiksle. Jame išskiriami pagrindiniai veiksmai :

- Pasirinkti mokėjimo planą (animatorius),
- Valdyti Blender projektus (animatorius),
- Generuoti Blender projektus (animatorius),
- Valdyti vartotojus (administratorius),
- Stebėti sistemos darbą (administratorius),
- Keisti sistemos nustatymus (administratorius),
- Generuoti ataskaitas (vadybininkas),
- Koreguoti mokėjimo planus (vadybininkas).



2.2 pav. Apibendrinta vartotojų panaudojimo atvejų diagrama

2.2 paveikslo diagramoje taip pat apibrėžiamas kuriamos sistemos kontekstas. Administratorius ir vadybininkas yra sistemos vidiniai vartotojai, o animatorius išorinis klientas.

Pagrindinis šio darbo tikslas - panaudojant debesų kompiuterijos technologijas sukurti teorinį ir praktinį kompiuterinės grafikos generavimo sistemos modelį, kuris našiau išnaudotų skaičiavimo resursus negu dabar esančios sistemos bei atsižvelgtų į vartotojų srautų istoriją.

Pagrindiniai uždaviniai:

- Atlikti esamų grafikos generavimo sistemų, kurios naudoja debesies technologijas, analizę,
- Apžvelgti taikomus balansavimo tarp viešo debesies ir privačių resursų metodus,
- Parinkti privataus debesies kūrimui tinkamiausią nemokamą programinę įrangą ir realizuoti privačią debesies infrastruktūrą,
- Sukurti kompiuterinės animacijos generavimo informacinės sistemos projektą ir savitą, šios sistemos paslaugoms teikti tinkamą, debesies apkrovimo balansavimo modelį,
- Realizuoti sistemos prototipą,
- Ištirti debesies technologijose taikomos resursų virtualizacijos įtaką uždavinių skaičiavimo laikui,
- Ištirti apkrovimo balansavimo modelio naudą prototipinėje sistemoje.

2.6.Rizikos faktoriai ir kokybės kriterijai

Kuriamos sistemos efektyvus veikimas tiesiogiai priklauso nuo trečių šalių kompiuterinių resursų ir ryšio su jais, todėl bet kokie sutrikimais šioje vietoje gali būti sistemos stabilumo problema. Rizikos faktorius ir numatomus jų sprendimo būdus patogu apžvelgti lentele (2.4 lentelė).

2.4 lentelė. Rizikos faktoriai

Problema	Sprendimas
Gali dingti interneto ryšys su IaaS paslaugos tiekėju.	Jei tuo metu vėluos klientų aptarnavimas – vėluojantiems paslaugas teikti neapmokestintas.
Gali atsirasti nesuderinamumų sistemai komunikuojant su trečių šalių sistemomis.	Kuriant sistema turi būti naudojamas bendras debesies aplinkos valdymo standartas (pvz.: Amazon EC2).
Programinė įranga naudojama trečių šalių sistemose gali būti pakeista arba atnaujinta.	Kuriama sistema turi būti lanksti tokiems pakeitimams.
Gali atsirasti atnaujinta grafikos generavimo programinė įranga, todėl sistema nesugebėtų generuoti naujo tipo projektų.	Turi būti nuolatinis grafikos generavimo programinės įrangos galimų atnaujinimų stebėjimas. O esant planuojamoms išleisti naujoms versijoms reikia atnaujinti ir ištestuoti sistemoje veikiančią programinę įrangą ir su ja susijusius komponentus.

Sistemos veikimą galima laikyti kokybišku tik užtikrinant, jog visi besinaudojantys vartotojai gaus savo paslaugą nepriklausomai nuo vartotojų ir generuojamų projektų kiekio. Taip pat svarbu užtikrinti paslaugos gavimą laiku, t.y. kiekvienas į sistemą generuoti įkeltas projektas turi būti pradėtas generuoti ne vėliau negu nurodyta to projekto nustatymuose. Pastarajam kriterijui įgyvendinti svarbu laiku nuomoti papildomus resursus iš viešo debesies paslaugų tiekėjo.

2.7.Analizės išvados

Atlikus debesies kompiuterijos teikiamų paslaugų apžvalgą, sprendimų literatūros šaltiniuose ir esamų sistemų analizę bei apžvelgus galimas architektūros įgyvendinimo priemones prieitos išvados:

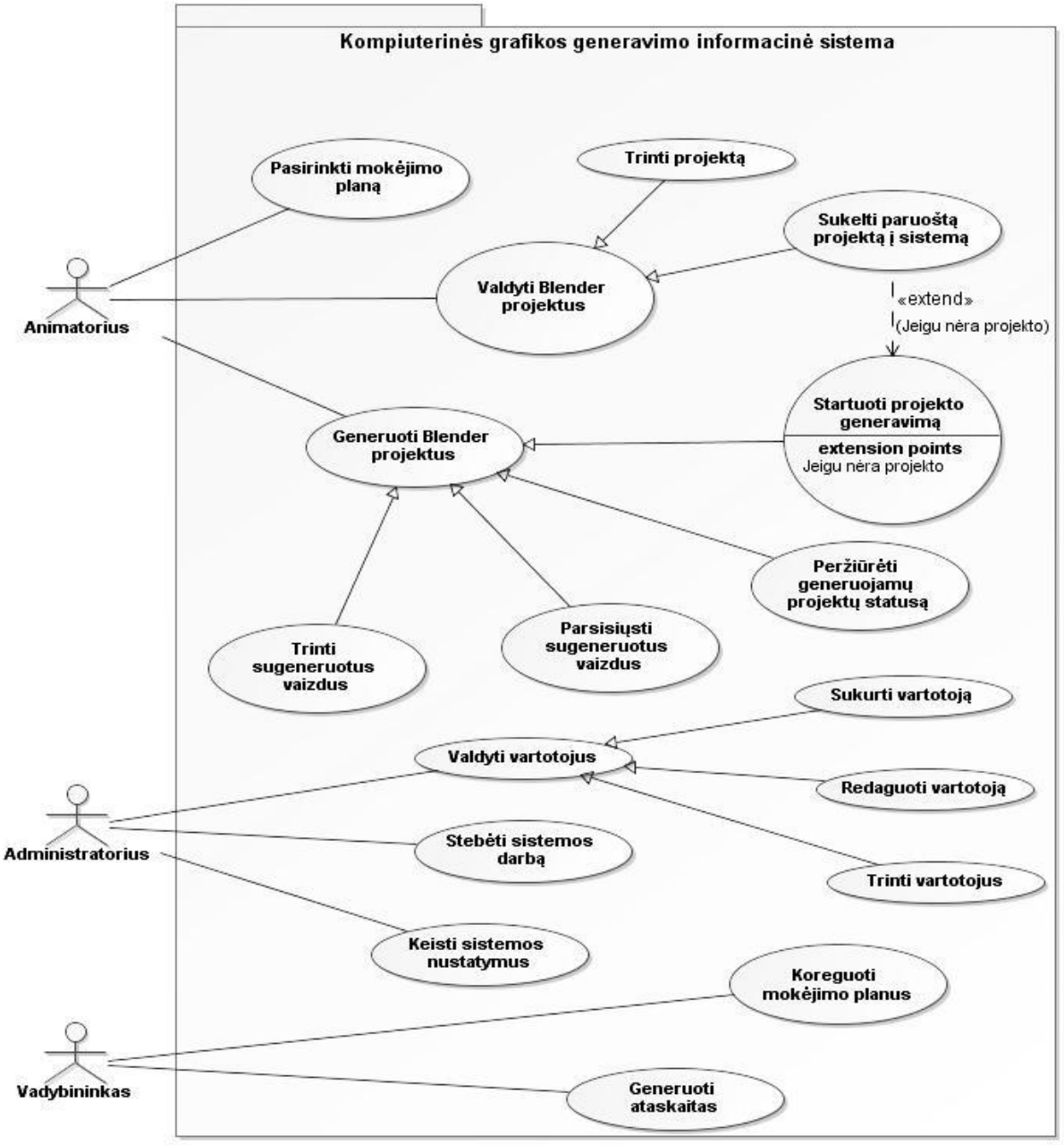
- Kuriamai IS realizuoti labiausiai tinka IaaS debesies infrastruktūra, nes uždavinių generavimui reikalingi kompiuteriniai resursai, kuriuose būtų diegiama specializuota programinė įranga.
- Šiuo metu esančios grafikos generavimo sistemos jau naudoja debesų kompiuterijos paslaugas. Tačiau šiose sistemose nėra galimybės atsižvelgiant į vartotojų srautus prognozuoti resursų poreikį ateityje. Todėl yra poreikis tirti ir realizuoti grafikos generavimo sistemą su galimybe prognozuoti ir rezervuoti resursus numatomiems uždaviniams.
- Atlikus apkrovimo balansavimo sprendimų literatūros šaltiniuose analizę nustatyta, kad balansavimo tarp privataus ir viešo debesies naudojimo metodai priklauso nuo uždavinių tipų ir jų pobūdžio. Todėl nėra vieno bendro, visur tinkančio metodo ir kuriamai kompiuterinės grafikos generavimo sistemai, kuri veikia hibridiniame debesyje, reikia savito balansavimo metodo.
- Privataus debesies infrastruktūrai dėl savo didelio paplitimo, populiarumo, palaikymo ir integracijos į operacines sistemas buvo pasirinkta Eucalyptus sistema. Taip sumažinamas sistemos kūrimo sudėtingumas ir laikas bei užtikrinamas didesnis stabilumas negu naudojantis panašias galimybes teikiančia mažiau paplitusia programine įranga.

3. Grafikos generavimo IS reikalavimai

Sistemos reikalavimų sudarymas esminis korektiško jos sukūrimo garantas. Tam reikalinga numatyti visas sistemos funkcijas, kurias jos vartotojas naudos. Taip pat ir apibrėžti nefunkcinius reikalavimus sistemos gyvavimui bei suderinimui su kontekstu, kuriame ji bus integruojama.

3.1.Reikalavimų specifikacija

Aprašant funkcinius reikalavimus sudaroma detali panaudojimo atvejų diagrama, kuri apima visus galimus vartotojų veiksmus sistemoje. Diagrama pateikta 3.1 paveiksle.



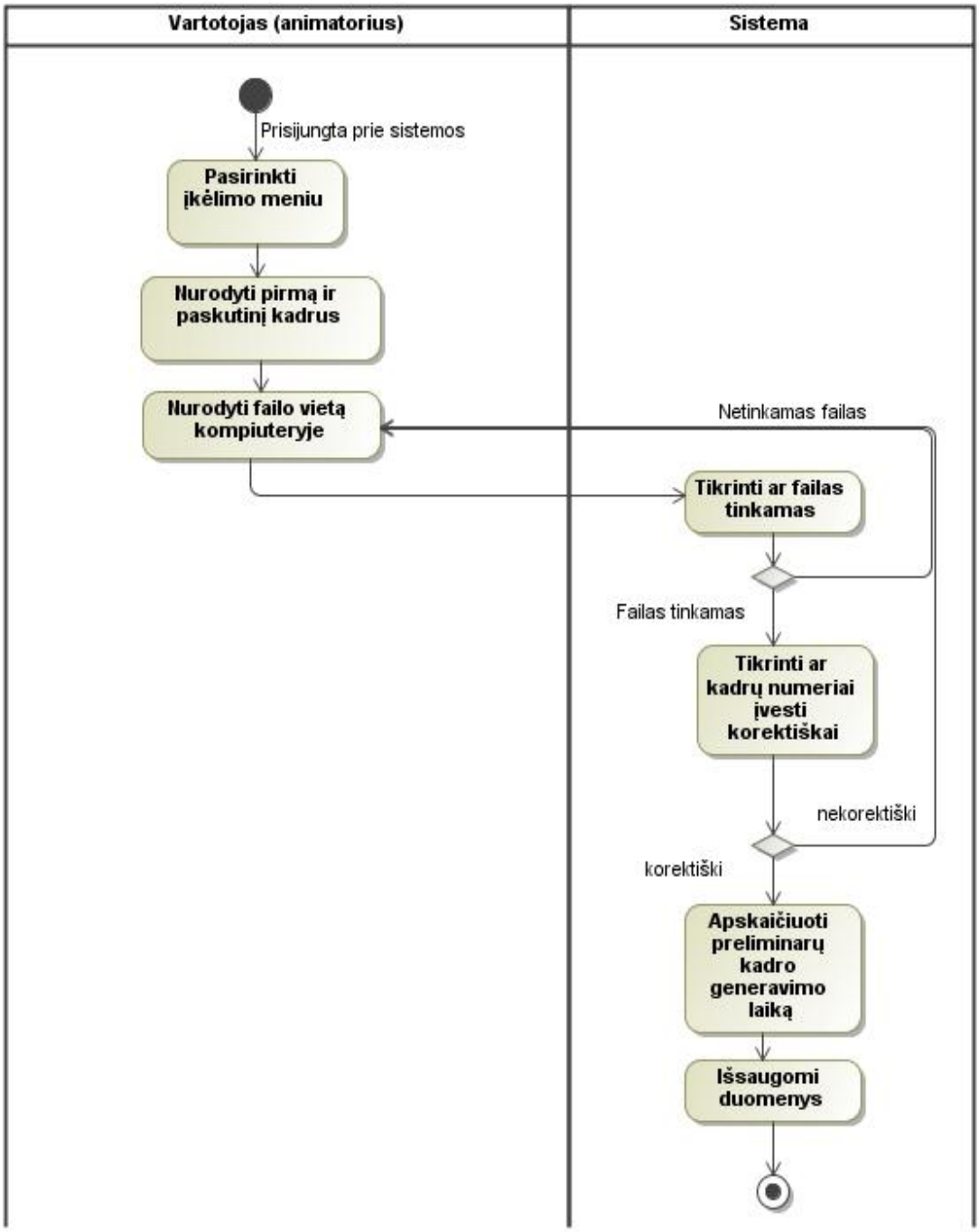
3.1 pav. Detaliausio lygio vartotojų panaudojimo atvejų diagrama.

Toliau lentelėmis aprašomi funkciniai reikalavimai pagal panaudojimo atvejus, pateikiamos kiekvieno jų prieš ir po sąlygos. Sudėtingesni panaudojimo atvejai aprašomi ir veiklos diagramomis:

1. Panaudojimo atvejis – „Sukelti paruoštą projektą į sistemą“. Animatorius sukelia paruoštus generuoti projektus į sistemą. Įkeliant projektą privalo nurodyti jo pirmą ir paskutinį kadrus. Sistema sugeneruoja 5 iteracinius projekto kadrus ir išveda vieno kadro generavimo laiko vidurkį. Pagal šį vidurkį bus numanomas preliminarus projekto generavimo laikas sistemoje. PA aprašymas pateikiamas 3.1 lentelėje, o diagrama 3.2 paveiksle.

3.1 lentelė. PA „Sukelti paruoštą projektą į sistemą“ specifikacija

PA „Sukelti paruoštą projektą į sistemą“	
Tikslas. Sukelti sukurtą Blender animacinės grafikos projektą į sistemą.	
Aprašymas. Šis panaudojimo atvejis yra PA „Valdyti Blender projektus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje
Aktorius	Animatorius
Sužadinimo sąlyga	Vartotojas nori nusiųsti projektą į sistemą (iš PA „Valdyti Blender projektus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas nurodo pirmą ir paskutinį kadrus.	
2. Vartotojas nurodo Blender projekto failo vietą savo kompiuteryje.	2.1. Sistema tikrina ar tai tinkamas failas.
	2.2. Sistema išsaugo Blender projekto duomenis
	2.3. Sistema apskaičiuoja vidutinį kadro generavimo laiką.
	2.4 DB ir serveryje išsaugomi duomenys.
***Po* sąlyga**	Serveryje išsaugotas Blender projektas, o duomenų bazėje išsaugotas kelias iki jo bei direktorija ir duomenys su jį įkėlusio vartotojo identifikatoriumi
Alternatyvūs scenarijai	
1a. Netinkamas failas.	Sistema pateikia pranešimą apie netinkamą failą.
Pastabos	
1. Vartotojas gali bet kada baigti PA, tačiau projektas bus nenusiųstas.	



3.2 pav. „Sukelti paruoštą projektą į sistemą“ veiklos diagrama

2. Panaudojimo atvejis – „Trinti projektą“. Vartotojui paliekama teisė trinti nebenaudojamus ar nereikalingus projektus, kuriuos jis pats įkėlė. PA aprašymas pateikiamas 3.2 lentelėje.

3.2 lentelė. PA „Trinti projektą“ specifikacija

PA „Trinti projektą“	
Tikslas. Trinti Blender animacinės grafikos projektą iš sistemos.	
Aprašymas. Šis panaudojimo atvejis yra PA „Valdyti Blender projektus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje
Aktorius	Animatorius
Sužadinimo sąlyga	Vartotojas nori ištrinti projektą (iš PA „Valdyti Blender projektus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka Blender projektus.	1.1. Sistema pateikia vartotojo projektų sąrašą.
2. Vartotojas pasirenka projektą ištrynimui.	
3. Patvirtina	3.1. Sistema ištrina projektą iš serverio ir duomenų bazės.
4. Vartotojas baigia PA	
***Po* sąlyga**	Projekto failas ištrintas iš serverio, o taip pat duomenų bazėje visi su juo susiję įrašai.
Alternatyvūs scenarijai	
Pastabos	
1. Vartotojas gali trinti tik savo įkeltus projektus.	

3. Panaudojimo atvejis – „Peržiūrėti generuojamų projektų statusą“. Vartotojas turi matyti generuojamų projektų statusą „laukia - generuojamas - baigtas“ lygmenyje bei matyti prognozuojamą vykdymo laiką. PA aprašymas pateikiamas 3.3 lentelėje.

3.3 lentelė. PA „Peržiūrėti generuojamų projektų statusą“ specifikacija

PA „Peržiūrėti generuojamų projektų statusą“		
Tikslas. Stebėti projektų generavimo eigą.		
Aprašymas. Šis panaudojimo atvejis yra PA „Generuoti Blender projektus“ dalis		
***Prieš* sąlyga**		Vartotojas yra registruotas sistemoje. Blender projektas yra įkeltas sistemoje, jam priskirti generavimo nustatymai bei yra startavęs.
Aktorius		Animatorius
Sužadinimo sąlyga		Vartotojas nori stebėti projekto vykdymą (iš PA „Generuoti Blender projektus“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka projektus.		1.1. Sistema pateikia projektų sąrašą su jų statusu (laukia, generuojamas arba baigtas) ir prognozuojamą vykdymo trukmę.
4. Vartotojas baigia PA		
***Po* sąlyga**		Sustabdytas Blender projekto vykdymas skaičiavimo mašinose ir pašalinti skaičiavimų duomenys arba sistemos būsena nepakitus.
Alternatyvūs scenarijai		
4a. Vartotojas sustabdo Blender projekto generavimo vykdymą.		Sistema ištrina duomenis iš skaičiavimo mašinų apie vykdytą projektą.
Pastabos		
1. Sustabdyto projekto tęsti nebegalima.		

4. Panaudojimo atvejis – „Startuoti projekto generavimą“. Vartotojas sistemoje, prieš generuodamas Blender projektą, turi nurodyti pirmą ir paskutinį kadrus atkarpos, kurią nori generuoti. Suvedus nustatymus projektas patalpinamas į generavimo eilę. PA aprašymas pateikiamas 3.4 lentelėje, o veiklos diagrama 3.4 paveiksle.

3.4 lentelė. PA „Startuoti projekto generavimą“ specifikacija

PA „Startuoti projekto generavimą“		
Tikslas. Pasirinkti Blender animacinės grafikos projekto generavimo formatą bei startuoti jo generavimą.		
Aprašymas. Šis panaudojimo atvejis yra PA „Generuoti grafikos projektus“ dalis		
***Prieš* sąlyga**		Vartotojas yra registruotas sistemoje. Blender projektas yra įkeltas sistemoje.
Aktorius		Animatorius
Sužadinimo sąlyga		Vartotojas nori generuoti projektą (iš PA „Generuoti Blender projektus“)
Veiklos taisyklės		
Susiję panaudojimo atvejai	**Išplečia PA**	Sukelti paruoštą projektą į sistemą
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka projektų sąrašą.		1.1 Sistema pateikia projektų sąrašą.
2. Vartotojas pasirenka norimą generuoti projektą		2.1 Sistema suformuoja nustatymų meniu.
3. Vartotojas nurodo pradžios ir pabaigos kadrus.		3.1. Sistema tikrina ar įvesti kadrai korektiški.
4. Vartotojas pasirenka norimą garso kokybę.		3.2. Sistema išsaugo projekto duomenis skirtus generavimui ir patalpina projektą į eilę.
7. Vartotojas baigia PA		
***Po* sąlyga**		Duomenų bazėje pagal Blender projekto identifikatorių išsaugoti generavimo duomenys ir projektas įrašomas į laukimo eilę
Alternatyvūs scenarijai		
3a. Nenurodomi kadrų parametrai.		Sistema pateikia pranešimą apie nenurodytus parametrus.
Pastabos		
1. Vartotojas gali vienu metu pradėti kelių Blender projektų generavimą, tačiau priklausomai nuo pasirinkto plano jie gali būtį generuojami ne iš karto.		

5. Panaudojimo atvejis – „Parsisiųsti sugeneruotus vaizdus“. Vartotojas pasirinktinai gali parsisiųsti sugeneruotą projekto vaizdą avi video arba zip archyvo formatais. PA aprašymas pateikiamas 3.5 lentelėje.

3.5 lentelė. PA „Parsisiųsti sugeneruotus vaizdus“ specifikacija

PA „Parsisiųsti sugeneruotus vaizdus“		
Tikslas. Parsisiųsti sugeneruota kompiuterinę animaciją į savo kompiuterį.		
Aprašymas. Šis panaudojimo atvejis yra PA „Generuoti Blender projektus“ dalis		
***Prieš* sąlyga**		Vartotojas yra registruotas sistemoje. Blender projektas yra jau baigtas generuoti.
Aktorius		Animatorius
Sužadinimo sąlyga		Vartotojas nori parsisiųsti projektą (iš PA „Generuoti Blender projektus“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka animacijų failus.		1.1. Sistema pateikia sugeneruotų failų sąrašą.
2. Vartotojas pasirenka norimą failą parsisiuntimui.		
3. Patvirtina		3.1. Sistema sugeneruoja parsisiuntimo nuorodą į animacijos arba zip failo vietą.
4. Vartotojas baigia PA		
***Po* sąlyga**		Vartotojui aktyvuotas failo parsisiuntimas.
Alternatyvūs scenarijai		
1a. Sugeneruotų animacijų nėra.		Sistema praneša, jog nėra sugeneruotų animacijų.
Pastabos		
1. Vartotojas gali vienu metu pradėti kelių Blender projektų parsisiuntimą.		

6. Panaudojimo atvejis - „Pasirinkti mokėjimo planą“. Vartotojui sistemoje turi būti sudaroma galimybė pasirinkti apmokėjimo už paslaugas planą, nuo kurio priklausytų Blender grafinių projektų generavimo trukmė bei įkainis. Vienu metu vartotojas apmokestinamas vienu planu. PA aprašymas pateikiamas 3.6 lentelėje.

3.6 lentelė. PA „Pasirinkti mokėjimo planą“ specifikacija

PA „Pasirinkti mokėjimo planą“	
Tikslas. Pasirinkti įkainius už paslaugas.	
Aprašymas. Šis panaudojimo atvejis yra atskira sistemos funkcija.	
***Prieš* sąlyga**	Animatorius yra registruotas sistemoje.
Aktorius	Animatorius
Sužadinimo sąlyga	Animatorius nori pasirinkti mokėjimo planą.
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka mokėjimo planus.	1.1. Sistema pateikia mokėjimo planų sąrašą.
2. Vartotojas išsirenka norimą mokėjimo planą.	
3. Patvirtina	3.1. Sistema pateikia informaciją apie įsigaliojantį mokėjimo planą.
4. Vartotojas baigia PA	
***Po* sąlyga**	Duomenų bazėje išsaugotas mokėjimo plano numeris ir susietas su vartotojo identifikatoriumi.
Alternatyvūs scenarijai	
2a. Vartotojas išsirenka jau naudojamą planą.	Sistema praneša, jog šis planas jau yra įjungtas.
Pastabos	
1. Vienu metu galima naudoti vieną mokėjimo planą, kuris taikomas visiems tuo metu generuojamiems projektams. 2. Pakeitus planą esant nebaigtiems generuoti projektams, jie baigiami generuoti pagal senus įkainius.	

7. Panaudojimo atvejis – „Trinti sugeneruotus vaizdus“. Vartotojas turi galimybę trinti iš sistemos nebereikalingus savo paties sugeneruotus vaizdus. PA aprašymas pateikiamas 3.6 lentelėje.

3.7 lentelė. PA „Trinti sugeneruotus vaizdus“ specifikacija

PA „Trinti sugeneruotus vaizdus“	
Tikslas. Trinti Blender animacinės grafikos sugeneruotus vaizdus iš sistemos.	
Aprašymas. Šis panaudojimo atvejis yra PA „Generuoti Blender projektus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir yra jo projektų sugeneruotas vaizdas.
Aktorius	Animatorius
Sužadinimo sąlyga	Vartotojas nori ištrinti sugeneruotą vaizdą (iš PA „Generuoti Blender projektus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka vaizdų sąrašą.	1.1. Sistema pateikia vartotojo sugeneruotų vaizdų sąrašą.
2. Vartotojas pasirenka vaizdą ištrynimui.	
3. Patvirtina	3.1. Sistema ištrina vaizdą.
4. Vartotojas baigia PA	
***Po* sąlyga**	Vaizdo failas ištrintas iš serverio, o taip pat duomenų bazėje visi su juo susiję įrašai.
Alternatyvūs scenarijai	
Pastabos	
1. Vartotojas gali trinti tik savo sugeneruotus vaizdus.	

8. Panaudojimo atvejis – „Sukurti vartotoją“. Sistemos administratorius turi galimybę sukurti naujus vartotojus priskiriant jiems vardą, pavardę, vartotojų grupę, įmonę, telefoną, adresą. PA aprašymas pateikiamas 3.8 lentelėje

3.8 lentelė. PA „Valdyti vartotojus“ specifikacija

PA „Sukurti vartotoją“	
Tikslas. Sukurti naujus sistemos vartotojus.	
Aprašymas. Šis panaudojimo atvejis yra PA „Valdyti vartotojus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi administratoriaus teises.
Aktorius	Administratorius
Sužadinimo sąlyga	Administratorius nori sukurti naują vartotoją (iš PA „Valdyti vartotojus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Administratorius išsirenka vartotojų kūrimo langą.	1.1. Sistema suformuoja vartotojo kūrimo meniu.
2. Administratorius įveda vartotojo duomenis.	
3. Patvirtina	3.1. Sistema išsaugo naują vartotoją.
4. Administratorius baigia PA	
***Po* sąlyga**	Duomenų bazėje sukurtas naujas vartotojas su savo identifikatoriumi ir priskirtas vartotojų grupei.
Alternatyvūs scenarijai	

9. Panaudojimo atvejis - „Redaguoti vartotoją“. Administratorius gali redaguoti esamų vartotojų duomenis. PA aprašymas pateikiamas 3.9 lentelėje.

3.9lentelė. PA „Redaguoti vartotojus“ specifikacija

PA „Redaguoti vartotoją“	
Tikslas. Redaguoti esamus sistemos vartotojus.	
Aprašymas. Šis panaudojimo atvejis yra PA „Valdyti vartotojus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi administratoriaus teises.
Aktorius	Administratorius
Sužadinimo sąlyga	Administratorius nori redaguoti vartotoją (iš PA „Valdyti vartotojus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Administratorius išsirenka vartotojų sąrašą.	1.1. Sistema suformuoja vartotojų sąrašą.
2. Administratorius redaguoja vartotojo duomenis.	
3. Patvirtina	3.1. Sistema išsaugo naujus vartotojo duomenis.
4. Administratorius baigia PA	
***Po* sąlyga**	Duomenų bazėje įterpti nauji vartotojo duomenys vietoje buvusių.
Alternatyvūs scenarijai	
Pastabos	

10. Panaudojimo atvejis – „Trinti vartotoją“. Administratorius gali trinti vartotoją, tuo pačiu pašalinami su tuo vartotoju susiję duomenys. PA aprašymas pateikiamas 3.10 lentelėje.

3.10 lentelė. PA „Trinti vartotojus“ specifikacija

PA „Trinti vartotoją“	
Tikslas. Trinti esamus sistemos vartotojus.	
Aprašymas. Šis panaudojimo atvejis yra PA „Valdyti vartotojus“ dalis	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi administratoriaus teises.
Aktorius	Administratorius
Sužadinimo sąlyga	Administratorius nori trinti vartotoją (iš PA „Valdyti vartotojus“)
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Administratorius išsirenka vartotojų sąrašą.	1.1. Sistema suformuoja vartotojų sąrašą.
2. Administratorius pasirenka vartotoją.	
3. Patvirtina	3.1. Sistema ištrina vartotoją.
4. Administratorius baigia PA	
***Po* sąlyga**	Duomenų bazėje ištrinti vartotojo duomenys, o serveriuose jo valdyti projektų failai ir sugeneruoti vaizdai.
Alternatyvūs scenarijai	
Pastabos	

11. Panaudojimo atvejis – „Stebėti sistemos darbą“. Administratorius turi matyti sistemoje veikiančių resursų apkrovimus ir būklę. PA aprašymas pateikiamas 3.11 lentelėje.

3.11 lentelė. PA „Stebėti sistemos darbą“ specifikacija

PA „Stebėti sistemos darbą“	
Tikslas. Stebėti skaičiavimo resursų apkrovimą.	
Aprašymas. Šis panaudojimo atvejis yra atskira sistemos funkcija.	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi administratoriaus teises.
Aktorius	Administratorius
Sužadinimo sąlyga	Vartotojas nori stebėti sistemos apkrovimus
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka meniu.	1.1. Sistema pateikia šiuo metu naudojamų resursų duomenis ir jų apkrovimo lygį.
2. Vartotojas baigia PA	
***Po* sąlyga**	Iš duomenų bazės įrašų ir sistemos užklausų suformuota ataskaita apie resursus.
Alternatyvūs scenarijai	
Pastabos	

12. Panaudojimo atvejis – „Keisti sistemos nustatymus“. Administratorius turi galimybę keisti sistemoje naudojamą algoritmą ir maksimalų vieno vartotojo generuojamų projektų skaičių. PA aprašymas pateikiamas 3.12 lentelėje.

3.12 lentelė. PA „Keisti sistemos nustatymus“ specifikacija

PA „Keisti sistemos nustatymus“	
Tikslas. Pakeisti veiksnius įtakojančius sistemos elgseną.	
Aprašymas. Šis panaudojimo atvejis yra atskira sistemos funkcija.	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi administratoriaus teises.
Aktorius	Administratorius
Sužadinimo sąlyga	Vartotojas nori pakeisti sistemos elgseną
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vartotojas išsirenka meniu.	1.1. Sistema pateikia šiuo metu naudojamus ir galimus nustatymus.
2. Vartotojas pasirenka norimus nustatymus.	2. Sistema išsaugo naujus nustatymus duomenų bazėje.
***Po* sąlyga**	Pasikeitus nustatymams kinta sistemos elgsena
Alternatyvūs scenarijai	
2.a Vartotojas neišsaugo naujų nustatymų.	2.a Sistemos veikla išlieka nepakitusi.
Pastabos	

13. Panaudojimo atvejis – „Generuoti ataskaitas“. Vadybininkas gali generuoti ataskaitas apie vartotojų veiksmus sistemoje pagal tam tikrus kriterijus – laikas, kaina, kompiuteriniai ištekliai. PA aprašymas pateikiamas 3.13 lentelėje.

3.13 lentelė. PA „Generuoti ataskaitas“ specifikacija

PA „Generuoti ataskaitas“	
Tikslas. Generuoti ataskaitas apie vartotoju veiksmus.	
Aprašymas. Šis panaudojimo atvejis yra atskira sistemos funkcija.	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi vadybininko teises.
Aktorius	Vadybininkas
Sužadinimo sąlyga	Vadybininkas nori analizuoti veiklą.
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vadybininkas pasirenka ataskaitą.	1.1. Sistema suformuoja ataskaitos formą.
2. Vadybininkas pasirenka ataskaitų duomenų rūšiavimo kriterijus.	
3. Patvirtina	3.1. Sistema suformuoja ataskaitą.
4. Vadybininkas baigia PA	
***Po* sąlyga**	Iš duomenų bazės įrašų suformuota ataskaita pagal pasirinktus kriterijus.
Alternatyvūs scenarijai	
Pastabos	

14. Panaudojimo atvejis – „Koreguoti mokėjimo planus“. Vadybininkas gali koreguoti mokėjimo planų aprašymus ir įkainius. PA aprašymas pateikiamas 3.14 lentelėje.

3.14 lentelė. PA „Koreguoti mokėjimo planus“ specifikacija

PA „Koreguoti mokėjimo planus“	
Tikslas. Pakeisti mokėjimo planų įkainius arba aprašymus.	
Aprašymas. Šis panaudojimo atvejis yra atskira sistemos funkcija.	
***Prieš* sąlyga**	Vartotojas yra registruotas sistemoje ir turi vadybininko teises.
Aktorius	Vadybininkas
Sužadinimo sąlyga	Vadybininkas nori koreguoti paslaugų apmokestinimą.
Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1. Vadybininkas pasirenka planą.	1.1. Sistema suformuoja plano nustatymus.
2. Vadybininkas koreguoja aprašymą ir įkainį.	
3. Patvirtina	3.1. Sistema išsaugo duomenis.
4. Vadybininkas baigia PA	
***Po* sąlyga**	Planams pradeda galioti nauji įkainiai.
Alternatyvūs scenarijai	
Pastabos	

Toliau suformuojami nefunkciniai sistemos reikalavimai:

REIKALAVIMAI SISTEMOS IŠVAIZDAI

Reikalavimas #:	**1**
Aprašymas:	Vartotojo sąsaja turi būti lengvai suprantama.
Pagrindimas:	Vartotojui turi būti paprasta naudotis sistema
Tinkamumo kriterijus:	Vartotojo sąsajos meniu punktų, laukų pavadinimai yra lengvai suprantami, naudojamos aiškios piktogramos, informacijos įvedimo laukai patogiai išdėstyti
Papildoma medžiaga:	
Istorija:	

Reikalavimas #:	**2**
Aprašymas:	Lengvai skaitoma ir valdoma sąsaja bei paprastas nesudėtingas panaudojimas.
Pagrindimas:	Siekiama, kad vartotojai kuo greičiau ir efektyviau įsisavintų naują sistemą.
Tinkamumo kriterijus:	Lengvai skaitoma sąsaja bei paprastas ir nesudėtingas panaudojimas.
Papildoma medžiaga:	
Istorija:	

Reikalavimas #:	**3**
Aprašymas:	Vartotojo sąsaja realizuota lietuvių kalba.
Pagrindimas:	Sistema kuriama Lietuvos vartotojams.
Tinkamumo kriterijus:	Tinkamai realizuota sąsaja.
Papildoma medžiaga:	**Administratoriui pateikiama informacija apie serverių būklę turi būti serverių operacinės sistemos kalba.**
Istorija:	

REIKALAVIMAI PANAUDOJAMUMUI

Reikalavimas #:	**4**
Aprašymas:	Privalomas teisingas UTF-8 koduotės simbolių apdorojimas.
Pagrindimas:	Vardai ir pavardės gali būti rašomos ne tik anglų abėcėlės raidėmis todėl turi būti teisingai apdorojamos.
Tinkamumo kriterijus:	Simbolinės informacijos įvedimo laukuose galima įvesti įvairius simbolius, kurie bus atvaizduoti neiškraipant.
Papildoma medžiaga:	

Reikalavimas #:	**5**
Aprašymas:	Galimybė riktis iš dalinai užpildytų laukų ir formų įvedant informaciją.
Pagrindimas:	Sistemos valdymas ir duomenų pildymas turi būti patogus vartotojui.
Tinkamumo kriterijus:	Galimybė pildyti duomenis pelės pagalba.
Papildoma medžiaga:	
Istorija:	

REIKALAVIMAI SAUGUMUI

Reikalavimas #:	**6**
Aprašymas:	Skirtingų grupių vartotojai mato tik jiems skirtą informaciją.
Pagrindimas:	Tam tikra su vartotoju susijusi informacija gali būti konfidenciali.
Tinkamumo kriterijus:	Vartotojai su skirtingais prisijungimo vardais ir slaptažodžiais mato tik savo duomenis.
Papildoma medžiaga:	
Istorija:	

Reikalavimas #:	**7**
Aprašymas:	Prisijungimų duomenys turi būti koduojami MD5 algoritmu.
Pagrindimas:	Sistemos saugumas.
Tinkamumo kriterijus:	Slaptažodžių atitikmenys lyginami užkoduoti.
Papildoma medžiaga:	
Istorija:	

REIKALAVIMAI SISTEMOS REALIZACIJAI

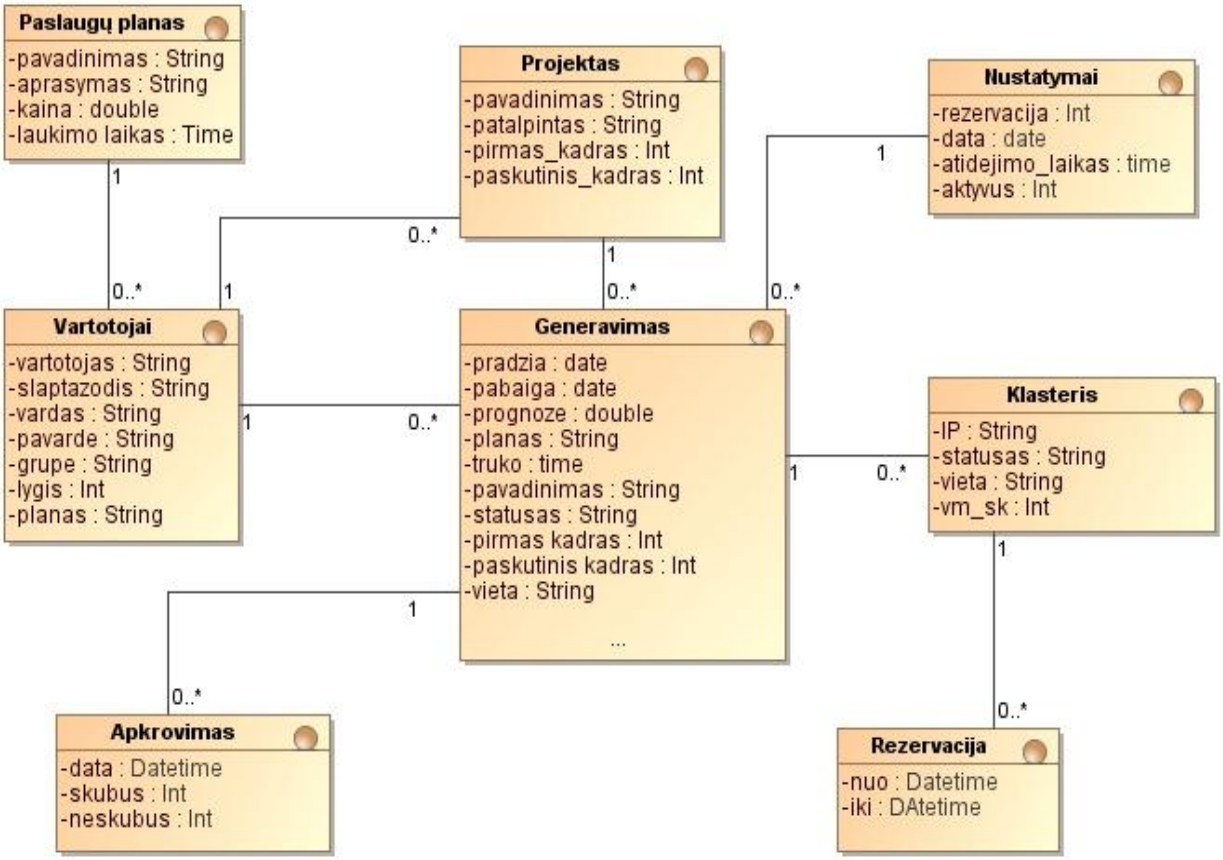
Reikalavimas #:	**8**
Aprašymas:	Sistema turi būti realizuota nemokamais kūrimo įrankiais (PHP, MySQL, JavaScript, JQuery ir kita atviro kodo įranga)
Pagrindimas:	Sistemos kūrimo kaštai turi būti minimalūs
Tinkamumo kriterijus:	Sistema kuriama atviro kodo įrankiais.
Papildoma medžiaga:	
Istorija:	

Reikalavimas #:	**9**
Aprašymas:	Sistema turi valdyti ir kompiuterinę grafiką generuoti ant analizėje pasirinktos programinės įrangos sukurtos debesies infrastruktūros.
Pagrindimas:	Sistemos darbui užtikrinti reikalingi dideli ir dinaminiai kompiuterinių resursų kiekiai
Tinkamumo kriterijus:	Sistema dinamiškai plečia naudojamą resursų kiekį.
Papildoma medžiaga: **Istorija:**	

Reikalavimas #:	**10**
Aprašymas:	Sistema turi pagal poreikį prijungti ir naudoti išoriniame debesyje nuomojamus kompiuterinius resursus.
Pagrindimas:	Sistemos darbui užtikrinti reikalingi dideli ir dinaminiai kompiuterinių resursų kiekiai
Tinkamumo kriterijus:	Sistema dinamiškai plečia naudojamą resursų kiekį.
Papildoma medžiaga: **Istorija:**	

3.2.Dalykinės srities modelis

Dalykinės srities modelis perteiktas esybių klasių diagrama (3.2pav.), kurioje aprašomos pagrindinės kuriamos sistemos esybės bei sąveika tarp jų. „Vartotojai“ saugo sistemos kliento informaciją bei nurodo jo galimų atlikti veiksmų lygi sistemoje. „Projektas“ esybėje talpinama informacija kur ir kokiu pavadinimu patalpintas projektas. „Paslaugų planas“ sistemoje saugo paslaugų kainodaros informaciją. „Nustatymai“ skirta sistemos elgseną įtakojantiems nustatymams saugoti. „Generavimai“ apjungia informaciją apie generuotus ir generuojamus projektus – pradžios ir pabaigos laikus, sugeneruoto failo pavadinimą ir dydi, bei naudojamos kainodaros planą ir generavimo statusą. „Klasteris“ skirta duomenims apie virtualių mašinų užimtumą realiu laiku ir siejantis su „Generavimai“ nurodo kokie projektai kuriuose klasteriuose generavosi. „Apkrovimas“ esybė skirta duomenų kaupimui apie aukšto ir žemo prioritetų uždavinių srautus, pagal juos pateikiamos ataskaitos. „Rezervacija“ savo ruožtu žymi kuriais laiko momentais kurie klasteriai yra rezervuoti.



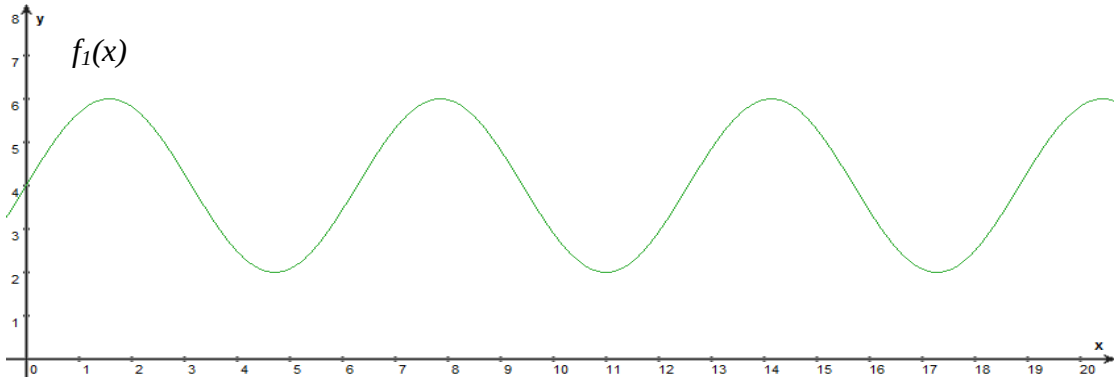
3.2 pav. UML esybių klasių diagrama

4. Grafikos generavimo IS projektas

Pagal panaudojimo atvejams sudarytus reikalavimus ir veiklų diagramas, toliau projektuojama informacinės sistemos dalis atsakinga už vartotojų administravimą, paslaugų planų sukūrimą, projektų valdymą bei generavimo eilių sudarymą ir paskirstymą. Pirmiausiai suformuojamas matematinis resursų panaudojimo optimizavimo modelis. Detalizuojama vartotojo sąsaja, jos navigavimas bei sąryšis su sistemos valdikliais. Taip pat sudaromas informacinės sistemos duomenų bazės modelis. Projekte taip pat pateikiami algoritmai vykstantys skaičiavimo serveriuose, kurie užtikrina prieš tai sudaryto matematinio modelio realizaciją.

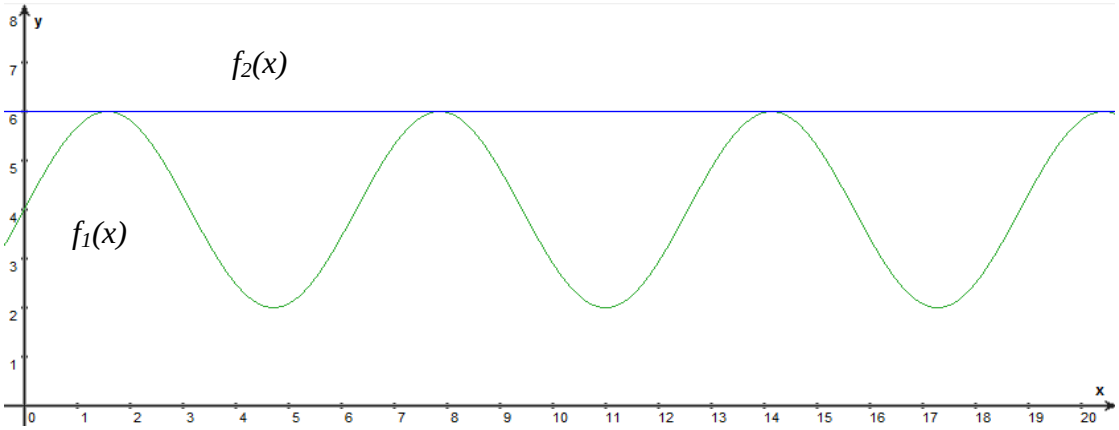
4.1.Sistemos pagrindimas ir esmės išdėstymas

Užtikrinti visų sistemos vartotojų poreikius, kai jie reikalauja daug resursų, o vartotojų srautas gali būti kintamas, yra sudėtingas uždavinys. Šiam uždaviniui spręsti galima įsigyti didelį kiekį skaičiavimo serverių. Bet toks sprendimo būdas nėra finansiškai efektyvus, nes atsiranda prastovos. Kompiuterinių resursų poreikį laike apibrėšime funkcija $f_1(x)$, kuri imituos resursų panaudojimo poreikio kitimą laike (4.1 pav.), kai X ašis nusako laiką valandomis, o Y ašis skaičiavimo mašinų kiekį.



4.1 pav. Skaičiavimo mašinų poreikis laike

Tokių poreikių įgyvendinimui reikalingų kompiuterinių resursų kiekį atvaizduosime prijungdami dar vieną funkciją $f_2(x)$ (4.2 pav.) , kai X ašis nusako laiką valandomis, o Y ašis skaičiavimo mašinų kiekį.



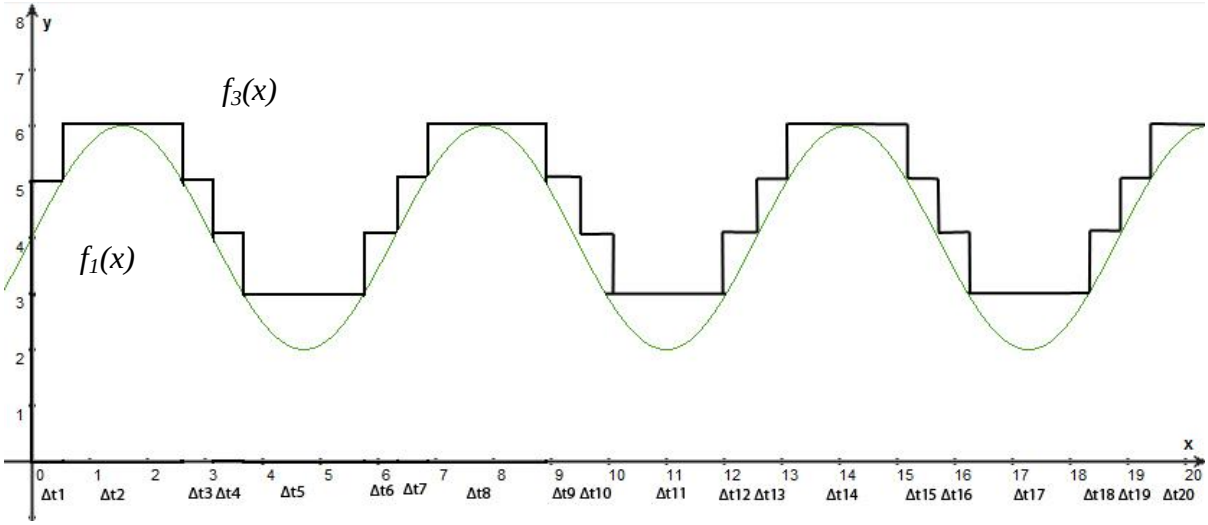
4.2 pav. Skaičiavimo mašinų turimas ir reikiamas kiekis laike

Šių funkcijų apibrėžtinių integralų skirtumas laike, gali parodyti nepanaudojamų resursų kiekį laiko periode (2).

$$R = \int_{x1}^{x2} f_2(x)dx - \int_{x1}^{x2} f_1(x)dx \quad ; \tag{2}$$

kur R – nepanaudotų resursų kiekis laike. Kuriant informacinę sistemą siekiama, kad $R \to 0$, tik tokiu būdų sistema bus efektyvi. Pilnai efektyvia sistema laikomas scenarijus, kai aptarnaujant visą vartotojų srautą privatūs resursai išnaudojami 100%.

Sumažinti grafikos generavimo sistemos prastovas leidžia debesų kompiuterijos paslaugos. Sistemos apkrovimo piko momentais galima iš IaaS viešo debesies paslaugos tiekėjo išsinuomoti papildomą kiekį virtualių resursų. Tokiu būdu suformuojamas hibridinis debesis, kurio jungtinis resursų kiekis yra užtektinas visiem uždaviniam, o prastovos sumažinamos iki minimumo. Išplečiant ir sumažinant viešų resursų kiekį atsiranda diskretizuoti laiko intervalai Δt1 ... Δtn , kurių momentu visada bus sveikas resursų skaičius, kuris visada lygus arba šiek tiek didesnis už tuo metu reikiamą kiekį. Tokį resursų kiekį apibrėžianti $f_3(x)$ funkcija atvaizduojama 4.3 paveiksle, kur taip pat matomas reikiamas funkcijos $f_2(x)$ apibrėžiamas resursų kiekis.



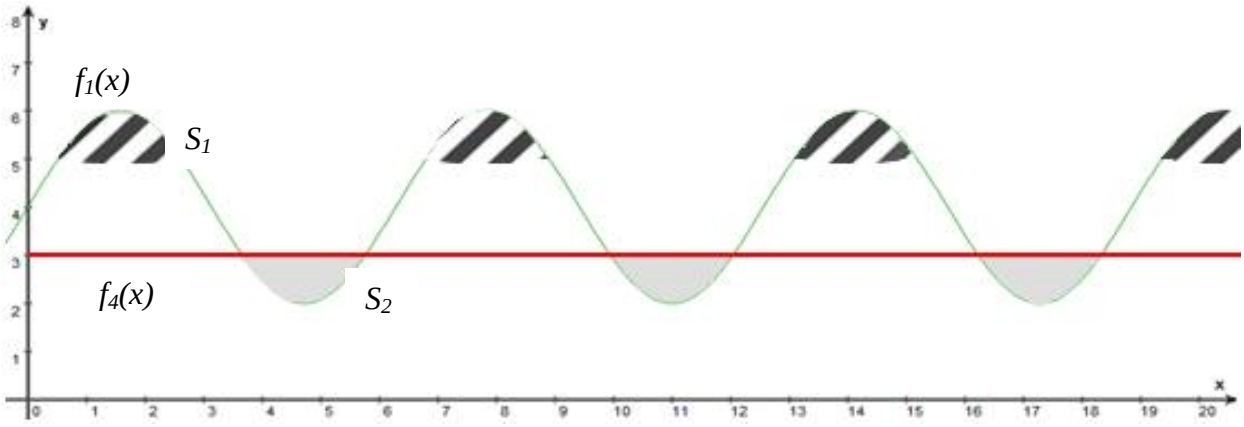
4.3 pav. Skaičiavimo mašinų jungtinis kiekis ir reikiamas kiekis laike

Tokio principo efektyvumo patikrinimui galime palyginti 4.2 paveiksle nubrėžtos funkcijos $f_2(x)$ (turint tik nuosavą įrangą) su 4.3 paveiksle gautos $f_3(x)$ (hibridinio debesies) funkcijos apibrėžtinius integralus. Šiuo atveju mažesnis integralas reiškia mažiau sunaudotų resursų tam pačiam skaičiavimų kiekiui (3).

$$\int_{x1}^{x2} f_3(x)dx < \int_{x1}^{x2} f_2(x)dx \ ; \tag{3}$$

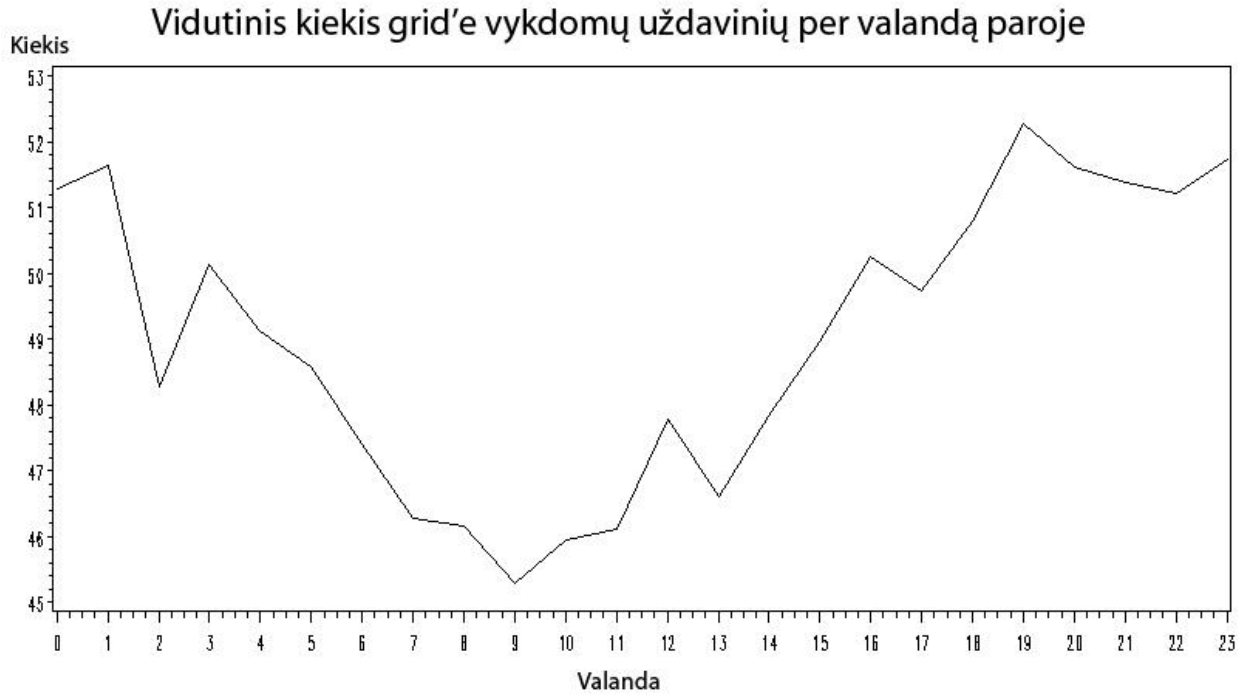
kadangi 4.3 paveiksle $f_3(x)$ funkcijos integralas yra mažesnis už funkcijos $f_2(x)$ 4.2 paveiksle, tai taikomas metodas našiau išnaudoja kompiuterinius resursus.

Kuriamos kompiuterinės animacijos generavimo informacinės sistemos privačių resursų panaudojimo optimizavimui numatyta realizuoti du vartotojų planus. Pirmas planas suteiktų aukštą prioritetą generuojamiems projektams ir juos sistema generuotų iš karto. Antras planas leistų atidėti projekto generavimą pasirinktą laiko tarpą, jeigu tuo metu sistemoje privačiame debesyje nėra laisvų resursų. Tai leidžia sumažinti uždavinių siunčiamų į viešą debesį kiekį, juos atidedant laiko tarpui, kai privačiuose resursuose nevykdomi skaičiavimai. 4.4 paveiksle atvaizduota $f_1(x)$ funkcija žymi resursų poreikį uždaviniams, $f_4(x)$ funkcija - turimą resursų kiekį, S_1 dryžuotas plotas - galimus atidėti uždavinius, o S_2 pilkas plotas - laisvus laiko tarpus privačiuose resursuose į kuriuos galima atidėti uždavinius.



4.4 pav. Neskubių uždavinių atidėjimas į žemo apkrovimo laiko intervalus

Priede pateikto straipsnio darbe, kartu su bendraautoriais buvo atliktas teorinis tokios sistemos veikimo tyrimas. Buvo susisteminti pusės metų laikotarpio Grid sistemos uždavinių vykdymo laike duomenys. Šių duomenų analizė parodė, kad sistemos apkrovimas svyruoja priklausomai nuo paros meto (4.5 paveikslas) ir dienos savaitėje (4.6 paveikslas).



4.5 pav. Vidutinis uždavinių kiekis per valandą paroje

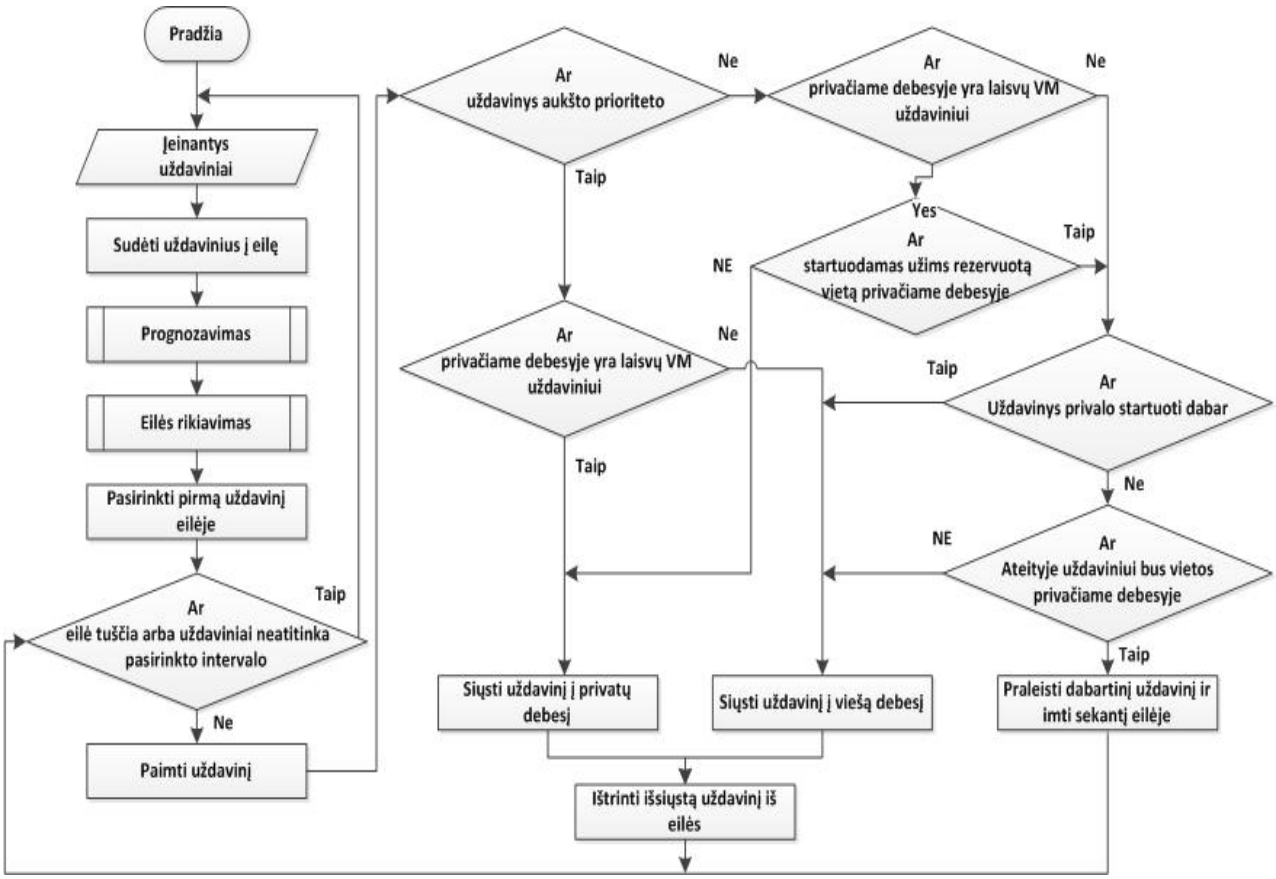
4.6 pav. Vidutinis uždavinių kiekis per valandą savaitėje

Matomas uždavinių kiekio periodinis ir sistemingas padidėjimas tuo pačiu paros metu kiekvieną savaitės dieną. Kadangi Grid sistema, skirta uždavinių skaičiavimui, veikia panašiais principais į debesų kompiuterijos IaaS sistemas, galima traktuoti, kad ir debesies technologijom veikianti sistema bus apkraunama sistemingai. Todėl sukurtas ateinančių uždavinių pagal statistinius duomenis prognozavimo modelis.

Darbe taip pat Matlab programine įranga buvo sukurtas sistemos, kuri skaičiuoja uždavinius privačiame debesyje, o esant resursų trūkumui prijungia viešą, veikimo simuliatorius. Tokios sistemos modelį sudarė trys dalys - ateinančių uždavinių į sistemą generavimas, uždavinių eilės skaičiavimo ir uždavinių skaičiavimo simuliavimo. Šio modelio veikimas buvo apribotas sekančiomis sąlygomis:

- Sistemoje egzistuoja aukšto ir žemo prioriteto ateinančių užduočių srautai. Žemo prioriteto uždavinių skaičiavimas gali būti atidėtas 2 valandas, o aukšto prioriteto uždaviniai skaičiuojami iš karto,
- Generuojamas uždavinių kiekis laike yra priklausomas nuo statistinių duomenų,
- Uždaviniai į sistemą įvedami kas valandą,
- Uždaviniai generuojami taip, kad jų skaičiavimo laikas kistų intervale nuo 1 iki 7 valandų,
- Kiekvienas uždavinys padalinamas į keturias lygias dalis ir siunčiamas į simuliavimo modelį, kuriame kiekviena dalis priskiriama vienai virtualiai mašinai. Laikoma, kad visos dalys yra lygios ir jų skaičiavimas užtrunka tą patį laiko intervalą,
- Į simuliuojamą privatų debesį uždaviniai siunčiami priklausomai nuo jų prioriteto,
- Aukšto prioriteto uždaviniai siunčiami į viešą debesį, jeigu tuo momentu nėra laisvų resursų privačiame,
- Žemo prioriteto uždaviniai siunčiami į viešą debesį, jeigu po 2 valandų atidėjimo jiems vietos neatsirado,
- Žemo prioriteto uždaviniai siunčiami į viešą debesį iš karto, jeigu prognozuojama, kad per 2 valandų atidėjimo laikotarpį neatsiras vietos privačiuose resursuose,
- Kadangi aukšto prioriteto uždaviniai turi būti vykdomi iš karto, jiems rezervuojami resursai privačiame debesyje pagal statistinius duomenis.

Pagal šias sąlygas sistemos veikimą simuliuojantis Matlab aplinkoje algoritmas pateikiamas 4.7 paveiksle.



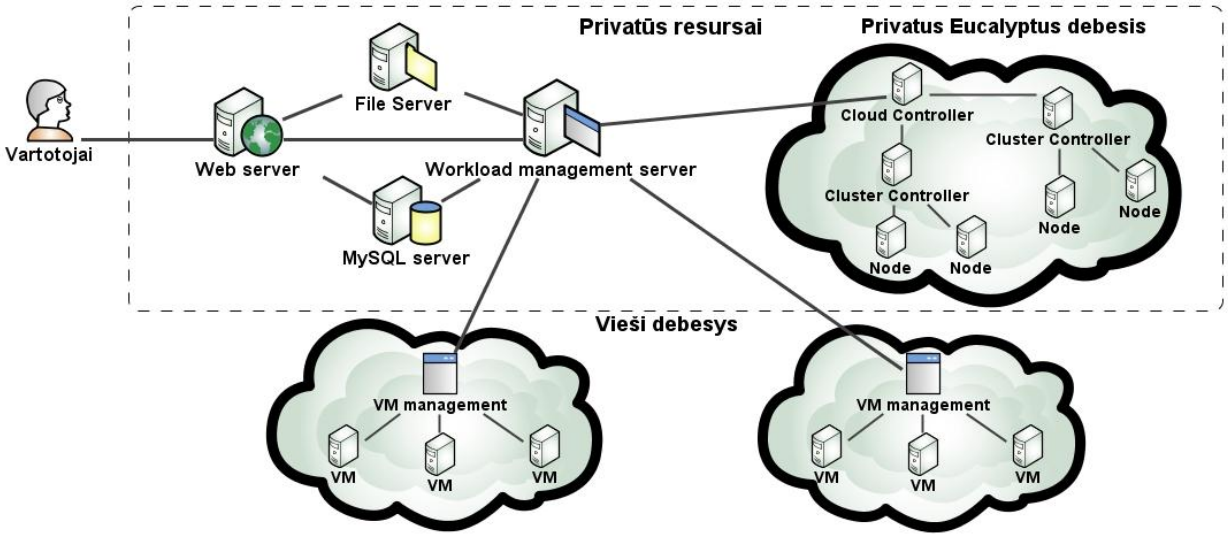
4.7 pav. Simuliuojamos sistemos veikimo algoritmas

Panaudojus šį apkrovimo balansavimo algoritmą bendrame naudotų resursų kiekyje viešo debesies resursų dalis sumažėjo nuo 35% iki 17% . Taip pat nuo 0% iki 28% padidėjo hibridinio debesies subalansavimas - buvo išnaudota 28% prastovų laiko atidėtų projektų generavimui.

4.2.Sistemos architektūra

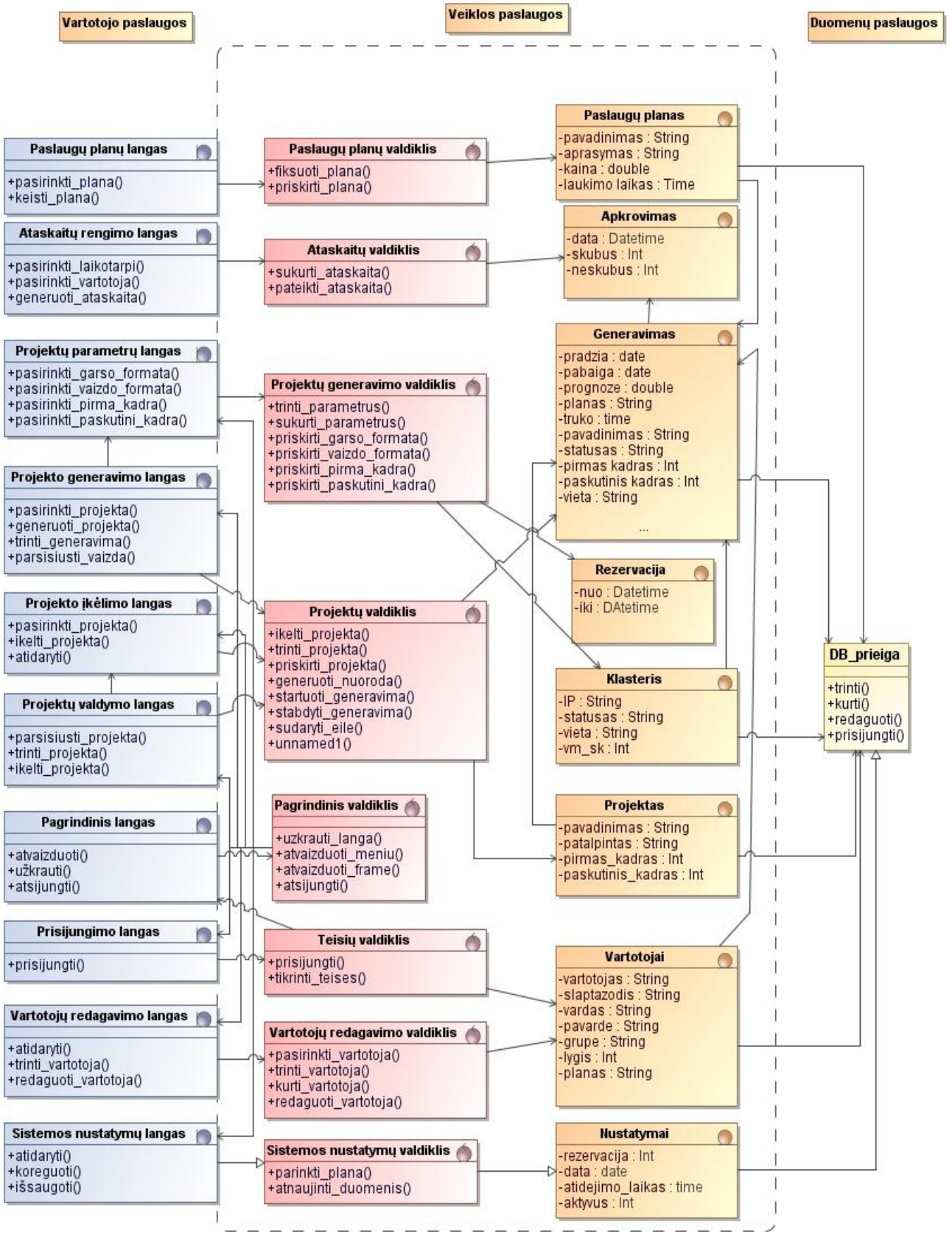
Sistemos realizacija kai apjungiami privačiame ir viešuose debesyse esantys resursai dar vadinami hibridiniais debesimis. Toks sistemos modelis leidžia dinamiškai išplėsti bei sumažinti papildomų resursų skaičių. Detali sistemos architektūra pavaizduota 4.8 paveiksle. Šiame sistemos modelyje apibrėžti privatūs resursai, kuriuos sudaro:

- Skaičiavimo serveriai apjungti naudojant Eucalyptus programinę įrangą,
- Web taikomųjų programų serveris,
- Duomenų serveris,
- Duomenų bazės serveris,
- Apkrovimo valdymo serveris.



4.8 pav. Sistemos architektūra

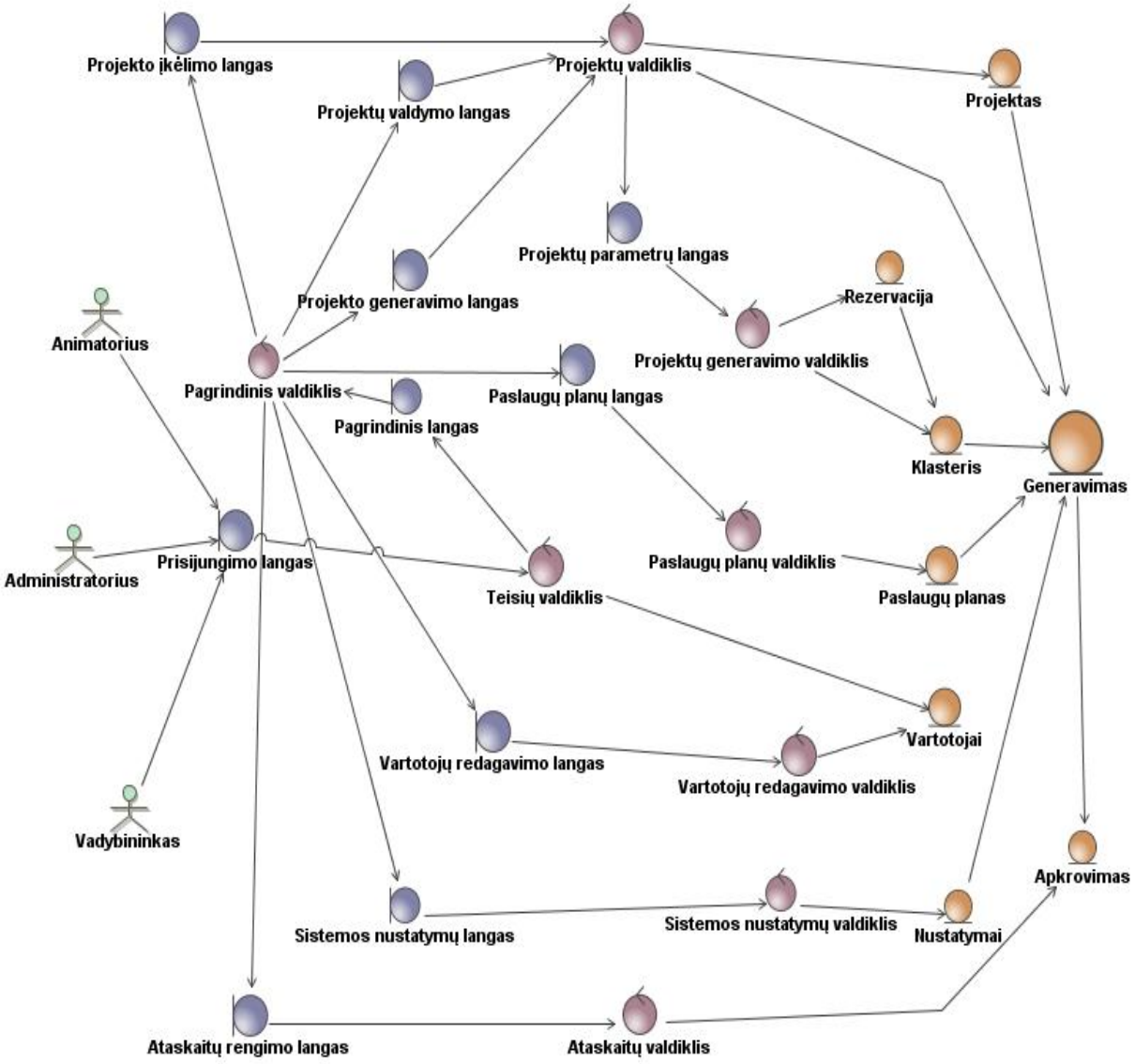
Vidinę informacinės sistemos architektūrą patogu vaizduoti klasių diagrama, kuri apima visus posistemius bei lygius. Šiuo atveju, trijų lygių UML klasių diagramoje atvaizduojami ryšiai tarp sistemos grafinės sąsajos langų ir valdiklių, taip pat valdiklių sąveika su esybėmis. Esybės, savo ruožtu, jungiasi su duomenų baze. Schema pavaizduota 4.9 paveiksle. Klasės suskirstytos pagal paslaugų rūšį - vartotojo, veiklos ir duomenų.



4.9 pav. UML trijų lygių klasių diagrama

4.2.1. Reikalavimų analizė

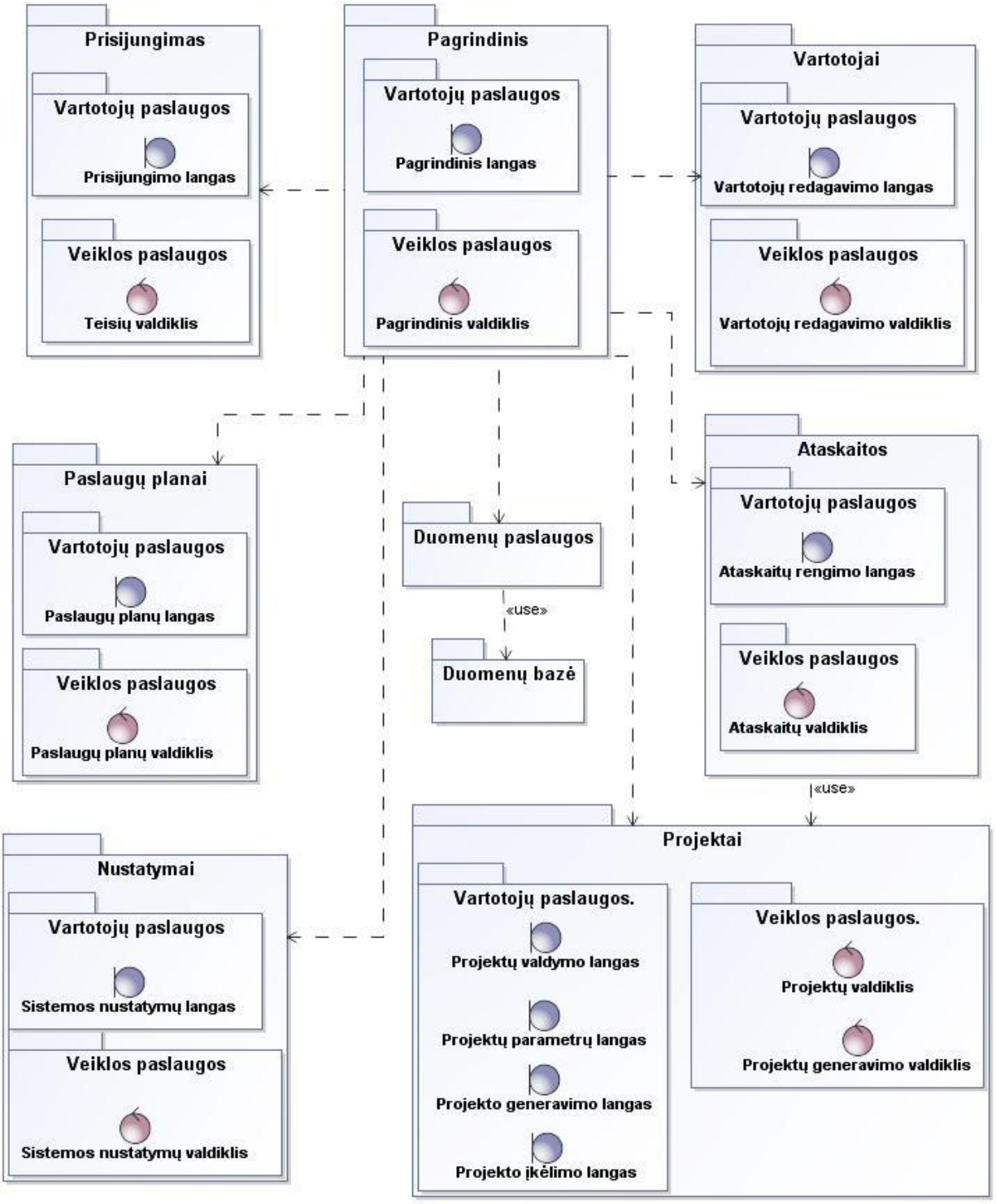
Pagal specifikuotus reikalavimus, reikalingus sistemai komponentus ir sąveikas tarp jų galime atvaizduoti robastiškumo diagrama (4.10 paveikslas). Čia nurodoma vartotojų, sistemos langų, valdiklių ir esybių sąveiką. Šiuo atveju vartotojai su sistema bendrauja per „Projekto įkėlimo“, „Projektų valdymo“, „Projektų generavimo“, „Prisijungimo“, „Paslaugų planų“, „Vartotojų redagavimo“ ir „Ataskaitų rengimo“ langus. Šis sistemos interfeisas ir gauti duomenys per jį apdorojami „Projektų“, „Projektų generavimo“ „Sistemos nustatymų“, „Paslaugų planų“, „Teisių“, „Vartotojų redagavimo“ ir „Ataskaitų generavimo“ valdiklių pagalba. Pastarieji valdikliai manipuliuoja „Projekto“, „Generavimų“, „Klasterio“, „Paslaugų planų“, „Nustatymų“, „Apkrovimo“ ir „Vartotojų“ esybėmis.



4.10 pav. Robastiškumo diagrama

4.2.2. Loginė visos sistemos architektūra

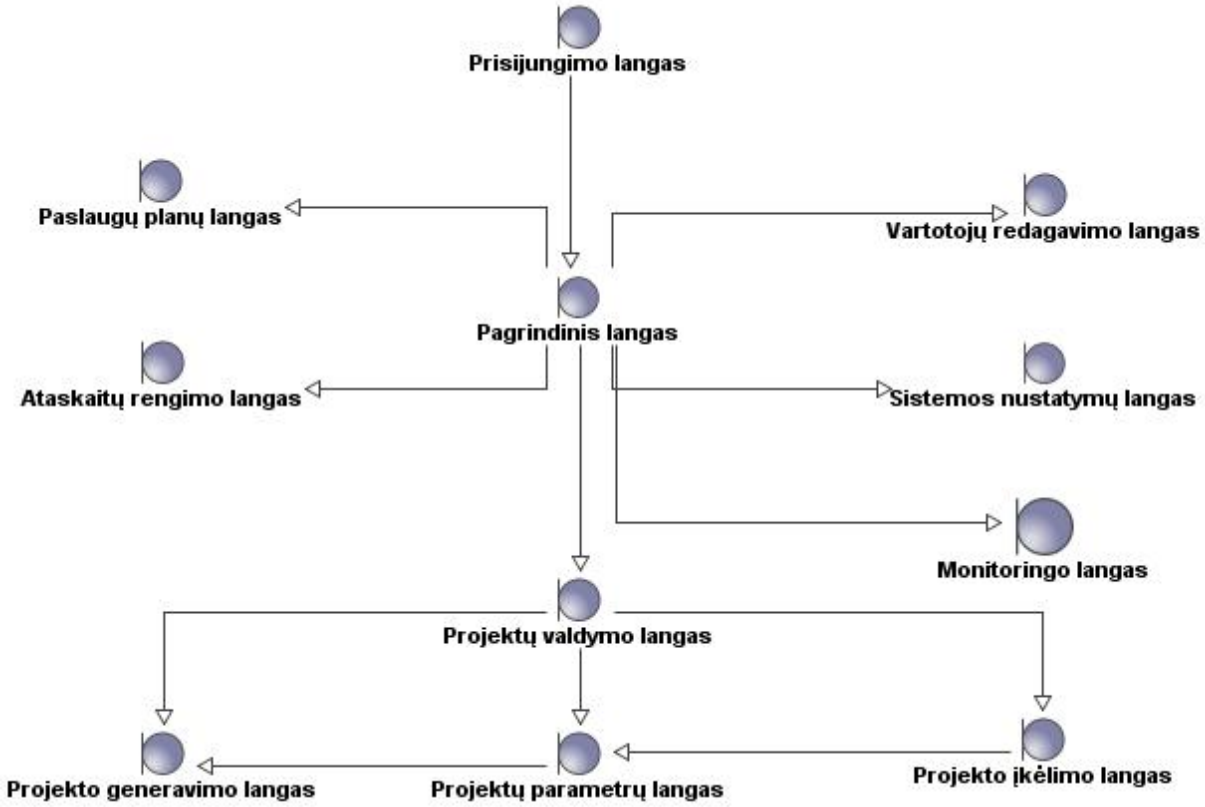
IS loginę sistemos architektūrą galima atvaizduoti detalizuotos sistemos architektūros diagrama, kurioje matosi posistemių tarpusavio ir išorinių paslaugų panaudojimas. Diagrama pateikta 4.11 paveiksle.



4.11 pav. detalizuota sistemos architektūra

4.2.3. Vartotojo paslaugos

Bendras sistemos sąsajos navigavimo planas atvaizduotas 4.12 paveiksle. Čia matome vartotojo kelio po sistemą langų hierarchiją. Pirmiausiai visi sistemos vartotojai patenka į Prisijungimo langą. Po autorizacijos priklausomai nuo vartotojo lygmens galima patekti į „Paslaugų planų“, „Ataskaitų generavimo“, „Vartotojų redagavimo“, „Monitoringo“, „Sistemos nustatymų“, „Projektų valdymo“ langus. Patogumo dėlei, vartotojui suteikiama galimybė tiesiogiai pereiti iš „Projekto įkėlimo lango“ į „Projektų parametrų langą“ ir atitinkamai į „Projekto generavimo langą“.



4.12 pav. Sistemos navigavimo planas

4.2.4. Veiklos paslaugos

Veiklos paslaugų modelyje aprašomos valdiklių ir esybių klasės bei sąveika tarp jų. Grafiškai atvaizduoti ryšiai yra pateikti, anksčiau darbe naudotoje, projekto klasių modelio diagramoje 4.9 paveiksle.

4.2.5. Duomenų paslaugos

Duomenų paslaugų modelyje aprašoma duomenų bazės klasė per kurią esybės sąveikauja su duomenų baze. Grafiškai atvaizduoti ryšiai yra pateikti, anksčiau darbe naudotoje, projekto klasių modelio diagramoje 4.9 paveiksle.

4.3.Detalus projektas

Pagal 4.1 dalyje aprašytą veikimo principą ir algoritmą projektuojamas realios kompiuterinės grafikos uždavinius generuojančios sistemos veikimo algoritmas. Pirmiausiai tam apibrėžiamos sistemos ribinės veikimo sąlygos, kuriomis remiantis algoritmas kurtas:

- Sistemoje galima pateikti dviejų skirtingų prioritetų uždavinius,
- Aukšto prioriteto skubūs grafikos uždaviniai turi būti generuojami iš karto,
- Žemo prioriteto uždaviniai nėra skubūs ir gali laukti tam tikrą, iš anksto nustatytą, laiko tarpą (priklauso nuo sistemos nustatymų),
- Atidėtų žemo prioriteto uždavinių pirmumas tarpusavyje priklauso nuo arčiausio laiko likusio iki privalomos jo generavimo pradžios,
- Žemo prioriteto uždavinys, nors ir pasibaigus maksimaliam atidėjimo laikui, vis tiek yra žemesnio prioriteto negu aukšto,
- Uždaviniai gali būti pateikiami bet kuriuo laiko momentu,
- Sistemoje gali būti įjungta arba išjungta, tikrinimo, ar uždavinys vėliau tilps privačiuose resursuose, funkcija,
- Priklausomai nuo sistemos nustatymų, gali būti privačiame debesyje iš anksto rezervuota vieta aukšto prioriteto uždaviniams,
- Žemo prioriteto uždaviniai niekada negali užimti rezervuotos vietos,
- Animatorius gali ištrinti projektą iš eilės, kol jis nepradėjo generuotis.

Algoritmas atvaizduojamas veiklos diagrama 4.13 paveiksle. Pirmiausiai vykdomas tikrinimas, ar yra projektų eilėje. Jeigu yra, jie išrikiuojami pagal artimiausią laiką iki vėliausios pradžios, o jei šis laikas sutampa - pagal prioritetą. Toliau imamas pirmas projektas iš eilės ir tikrinamas jo prioritetas.

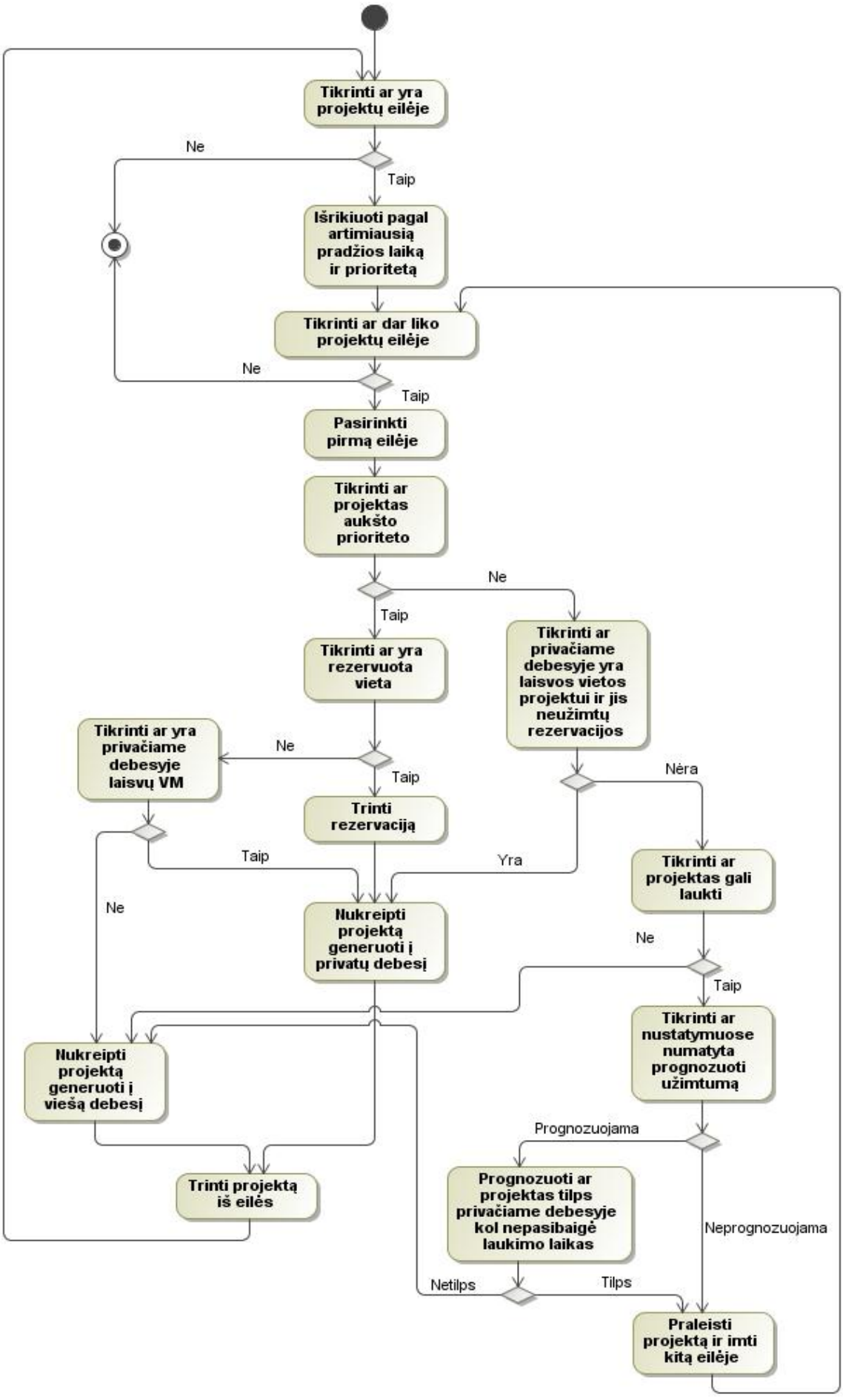
- Jeigu prioritetas aukštas, sistema tikrina, ar buvo jam iš anksto rezervuota vieta privačiame debesyje. Jeigu vieta rezervuota, tai jis siunčiamas generuotis į privatų debesį, priešingu atveju tikrinama, ar yra šiuo metu privačiame debesyje laisvų resursų. Teigiamu atveju projektas siunčiamas generuoti į privatų debesį, neigiamu, kadangi

aukšto prioriteto darbai negali laukti, - viešą debesį. Projektas pašalinamas iš eilės ir imamas sekantis.

- Jeigu prioritetas žemas, sistema tikrina, ar yra šiuo metu laisvų resursų privačiame debesyje ir ar tokiu atveju šis projektas neužims ateityje rezervuotiems projektams vietos. Teigiamu atveju projektas siunčiamas generuotis į privačius resursus, neigiamu - tikrinama, ar projektas gali būti atidėtas. Jei gali būti atidėtas, galimi du scenarijai:
    1. Sistemoje numatyta tikrinti ar projektas, pagal turimus duomenis, ateityje tilps į privačius resursus ir jeigu netilps, jį iš karto siųsti į viešą debesį.
    2. Sistemoje netikrinama ar projektas tilps ateityje pagal turimus duomenis, nes projektai gali būti pašalinami ir pridedami į eilę bet kuriuo metu. Tokiu atveju projektas atidedamas kaskart nerandant jam vietos ir tik esant maksimaliam atidėjimo laikui siunčiamas į viešą debesį.

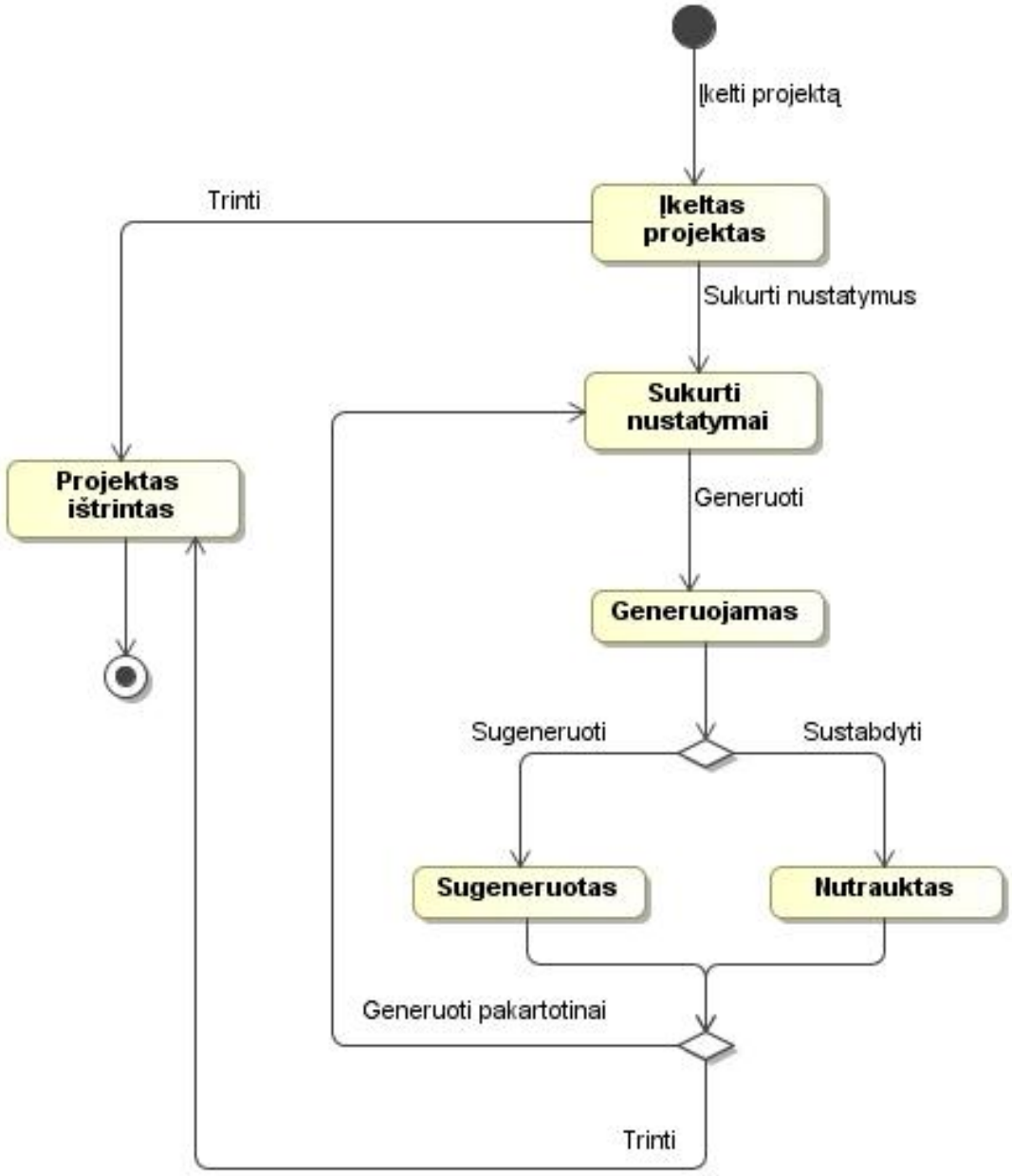
Atidėjus arba išsiuntus projektą generavimui imamas sekantis projektas.

Algoritmas sistemoje vykdomas kiekvieną minutę, todėl visi naujai atėję uždaviniai apdorojami iš karto.



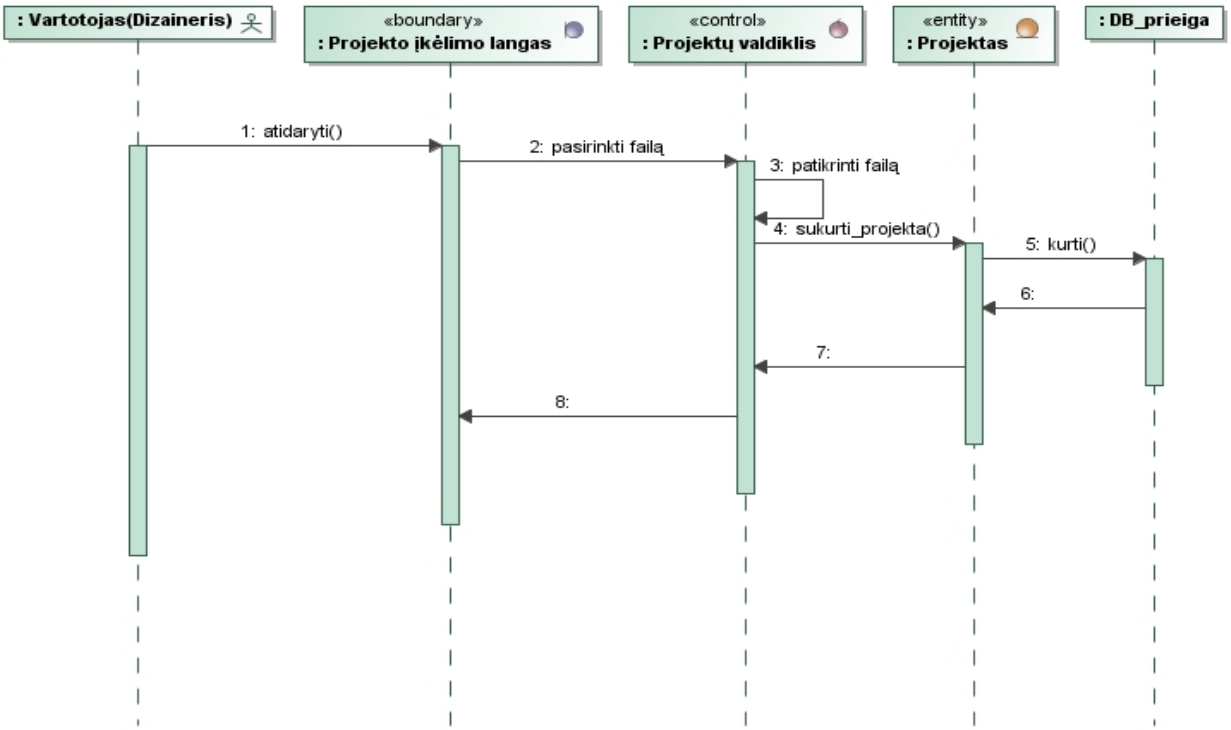
4.13 pav. Grafikos projektų generavimo sistemoje algoritmas

Projekto gyvavimo ciklui sistemoje aprašyti patogu naudoti būsenų diagramą. Pirmiausiai projektas įkeliamas į sistemą. Po šio veiksmo jis gali būti šalinamas iš jos arba generuojamas sistemoje. Prieš generuojant reikia nurodyti pirmą ir paskutinį kadrus. Projektas atiduodamas vykdymui ir gali būti pabaigiamas arba vartotojo ar kitų pašalinių veiksnių nutrauktas. Šioje stadijoje projektas gali būti generuojamas pakartotinai arba tapęs nereikalingu pašalinamas iš sistemos. Projekto gyvavimo ciklo būsenų diagrama atvaizduota 4.14 paveiksle.



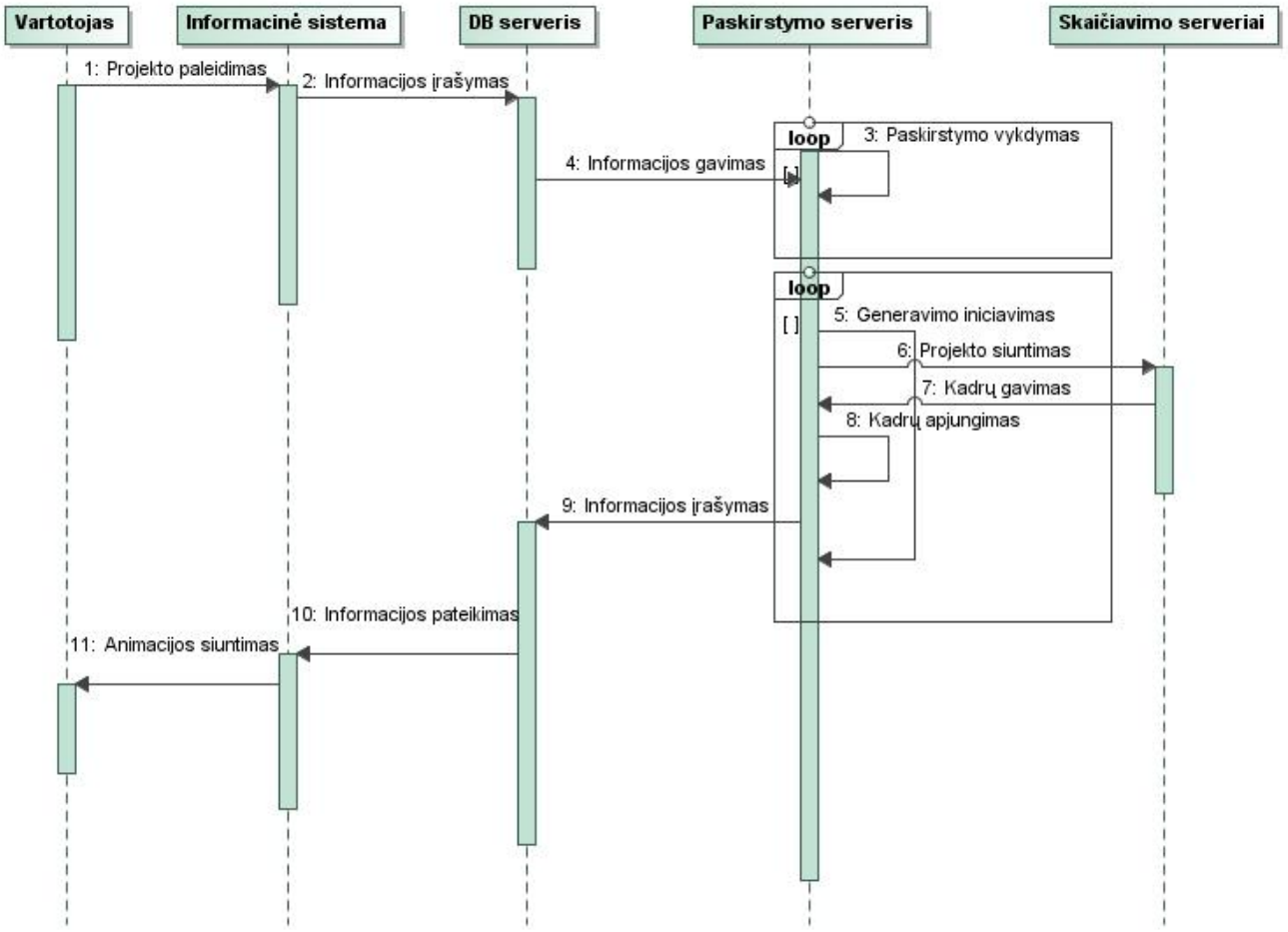
4.14 pav. Projekto gyvavimo būsenų diagrama

Projekto įkėlimą į sistemą galima perteikti sekų diagrama. Šis veiksmas inicijuojamas vartotojo per „Projekto įkėlimo langą“ ir yra apdorojamas „Projektų valdiklio“, kuris patikrina, ar pasirinktas įkėlimo failas yra tinkamo formato ir įkelia jį į duomenų bazę. Sekų diagrama atvaizduota 4.15 paveiksle.



4.15 pav. Projekto įkėlimo sekų diagrama

Taip pat sekų diagrama galime atvaizduoti veiksmų seką nuo projekto pateikimo generavimui iki jo galutinio rezultato grąžinimo vartotojui. Šį veiksmą inicijuoja vartotojas per „Projekto generavimo“ langą informacinėje sistemoj, apie tai atliekamas įrašas duomenų bazėje. Paskirstymo serveryje vykstantis nuolatinis apkrovimo valdymo ciklas gauna informaciją iš duomenų bazės apie generuojamus projektus ir juos paskirsto resursams. Paskirstyti projektai apdorojami nuolat cikle veikiančio paleidimo algoritmo, kuris inicijuoja projekto generavimą skaičiavimo serveriuose ir laukia grįžtančių sugeneruotų kadrų. Sugeneravus visus kadrus paskirstymo serveris apjungia juos į vientisą failą ir patalpina informaciją duomenų bazės serveryje apie atliktą generavimą. Informacinė sistema vartotojui pateikia atnaujintus duomenis apie projekto generavimą, o vartotojas, savo ruožtu, parsisiunčia sugeneruotą animaciją. Sekų diagrama atvaizduota 4.16 paveiksle.

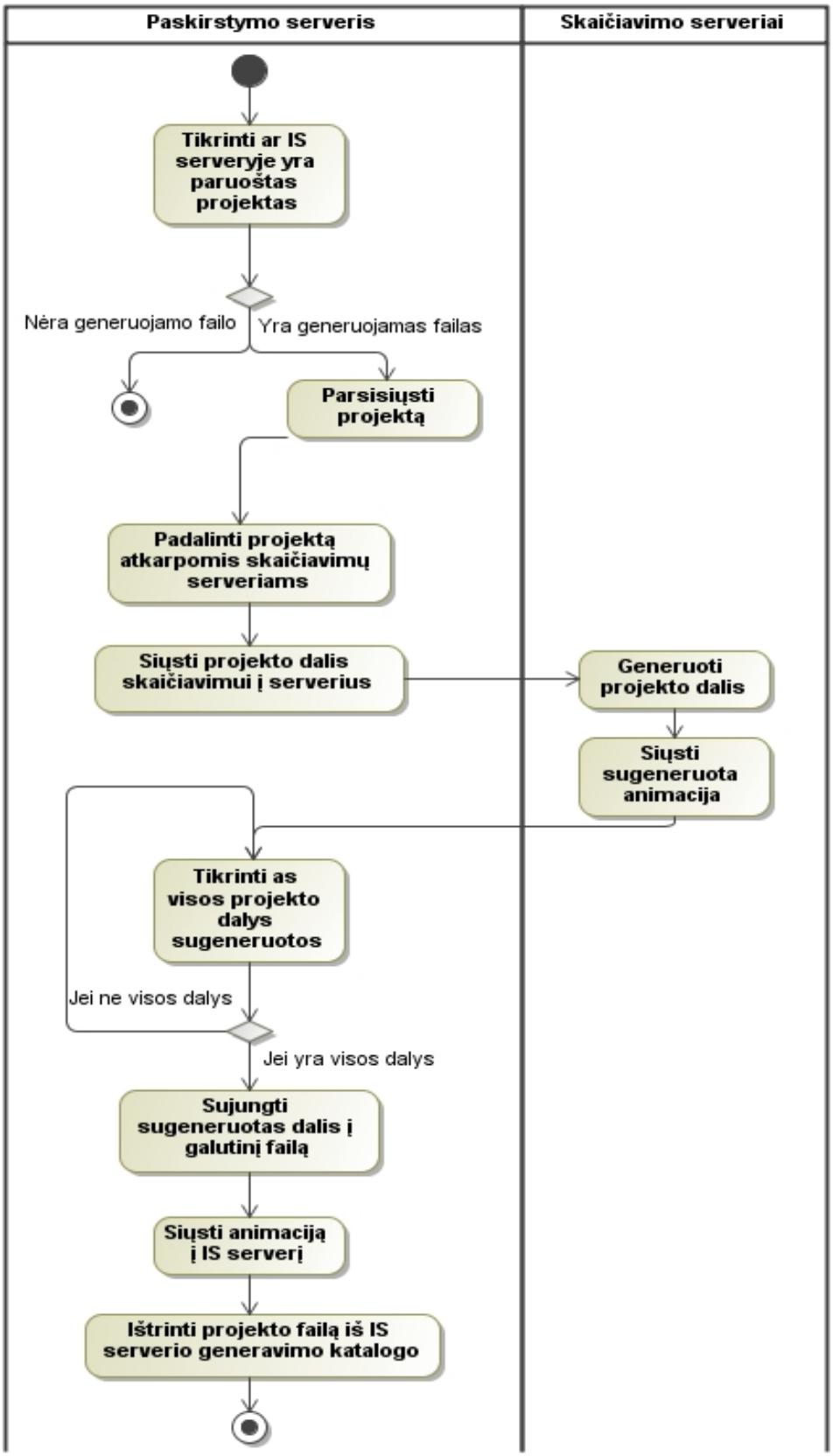


4.16 pav. Projekto generavimo sekų diagrama

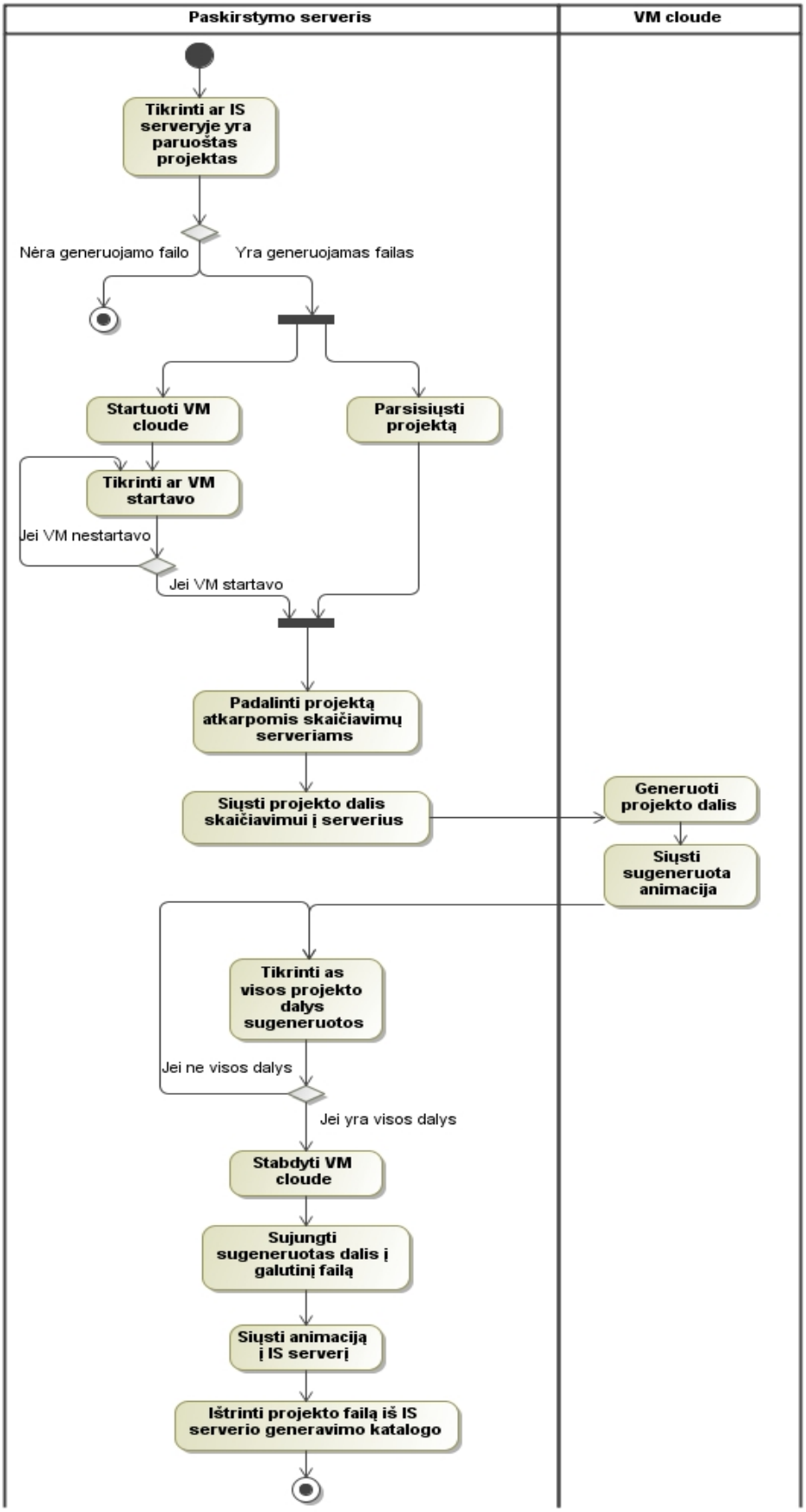
Toliau pateikiama Blender projekto generavimo algoritmo veiklos diagrama, kai skaičiavimai atliekami savuose serveriuose (4.17 paveikslas). Šiuos veiksmus atlieka atskiras paskirstymo serveris (grandis tarp informacinės sistemos ir skaičiavimo resursų). Pirmiausiai tikrinama ar yra paruoštas projektas informacinės sistemos serveryje. Jeigu yra – toliau parsisiunčiamas ir padalinamas atkarpomis animacijos projekto failas. Šios atkarpos lygiagrečiai siunčiamos į skaičiavimų serverius, kur pradedamas jų generavimas. Sugeneruotos dalys parsiunčiamos atgal į paskirstymo serverį, kur patikrinama ar jau sugeneruotos visos dalys. Esant visoms animacijos dalims, jos apjungiamos į vientisą galutinį failą. Pastarasis siunčiamas į informacinėje sistemoje iš anksto paruoštą katalogą. Baigus replikaciją ištrinamas projektas iš generavimo eilės.

Taip pat (4.18) paveiksle pateikiama alternatyvi Blender projekto generavimo algoritmo veiklos diagrama, kai skaičiavimai atliekami nuomotose serveriuose iš IaaS paslaugos tiekėjo. Veiklos scenarijus alternatyvus prieš tai analizuotam, tačiau atsiranda keli papildomi veiksmai. Kol parsisiunčiamas Blender projekto failas iš IS serverio, paskirstymo serveris siunčia komandą startuoti virtualias mašinas nuomojamuose resursuose. Tik startavus virtualioms mašinoms ir parsiuntus projekto failą vykdomi tolimesni veiksmai. Lyginant su prieš tai aprašytu scenarijumi

įsiterpia dar vienas paskirstymo serverio atliekamas veiksmas. Gavus visas sugeneruotas animacijos dalis į paskirstymo serverį siunčiama komanda stabdyti virtualias mašinas. Toliau seka identiški veiksmai kaip ir vykdant algoritmą savuose serveriuose.



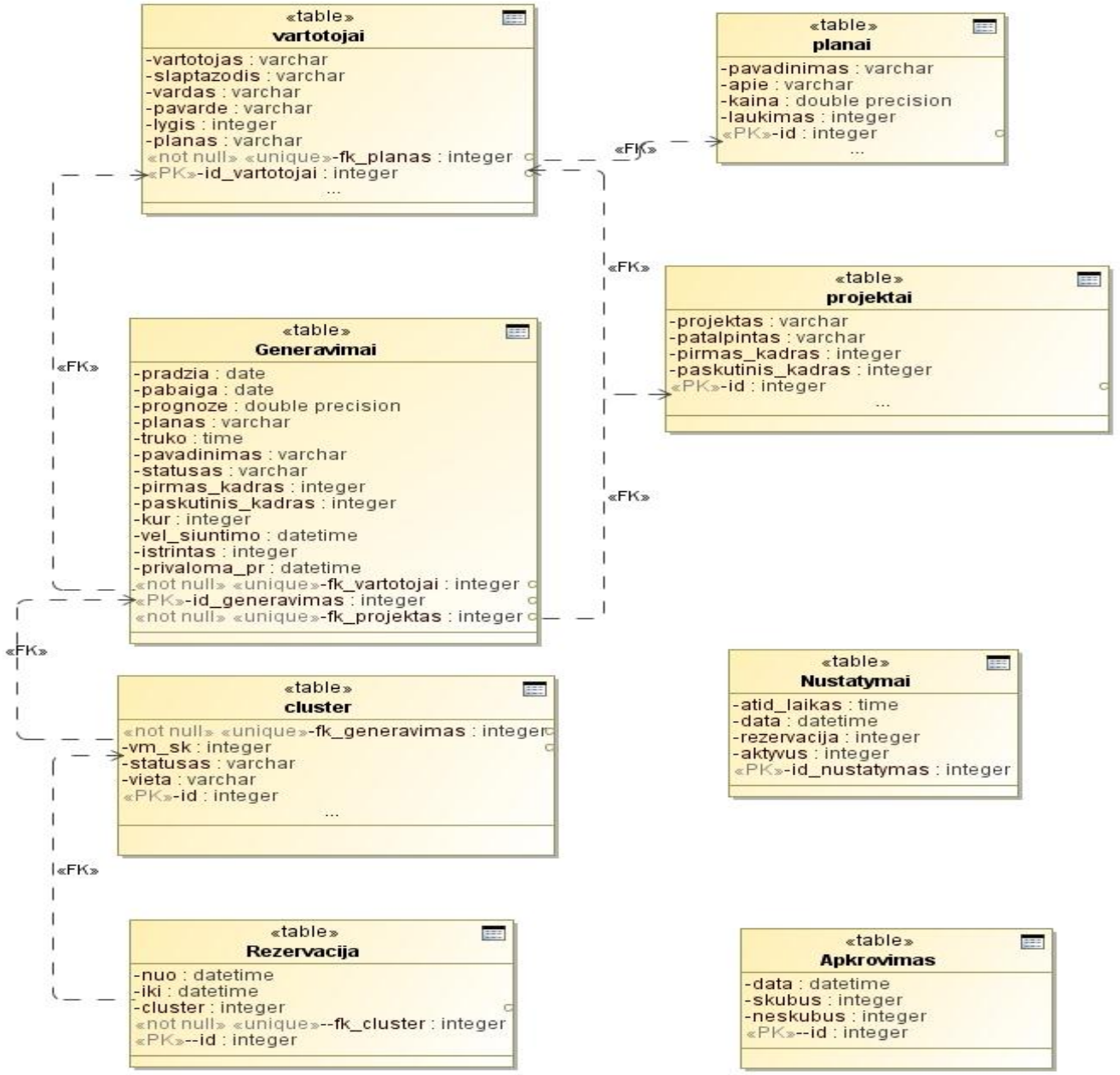
4.17 pav. Projekto generavimo savuose serveriuose veiklos diagrama



4.18 pav. Projekto generavimo nuomojamuose serveriuose veiklos diagrama.

4.4.Duomenų bazės schema

Šios sistemos duomenų bazei realizuoti (4.19 paveikslas), kaip ir esybių diagramoje, naudojamos 8 lentelės iš kurių 6 su saugomais išoriniais raktais, pagal kuriuos jos susaistomos. 2 lentelės saugo informaciją ir siejasi tik per sistemą. Visi ryšiai tarp lentelių yra „vienas su daug“. „Vartotojai“ saugo sistemos vartotojų duomenis. Lentelėje „Projektas“ talpinama informacija apie Blender projekto failą ir kurioje vietoje serveryje jis yra patalpintas. „Paslaugų planas“ saugo informaciją apie sistemoje esančius planus vartotojui. „Sistemos nustatymai“ leidžia pasirinkti sistemos elgseną. Lentelėje „Generavimai“ apjungiama informacija – koks vartotojas, naudodamas atitinkamą paslaugų planą generuoja projektą. „Cluster“ lentelėje saugoma informacija apie virtualių mašinų apkrovimą. „Rezervacija“ naudojama VM rezervacijai, o Apkrovimas sistemos sprendžiamų uždavinių kiekiui stebėti ir ataskaitoms ruošti.



4.19 pav. Projekto duomenų bazės schema.

4.5.Realizacijos modelis

Pagal parengtą informacinės sistemos teikiančios grafinių projektų generavimo paslaugą projektą, toliau sudaromas realizacijos modelis, kuriame aprašomas projekto įgyvendinimas įrankiais ir komponentais.

4.5.1. Komponentų modelis

Sistemos funkcijos realizuojamos skirtingais komponentais (php programiniais failais), kurie apjungiami sistemos. Komponentai skirstomi į matomus, kurie atvaizduoja informaciją per grafinę sąsają ir nematomus – šie kontroliuoja duomenų mainus, valdymą bei skaičiavimus sistemoje. Komponentus patogu atvaizduoti lentele (4.1 lentelė).

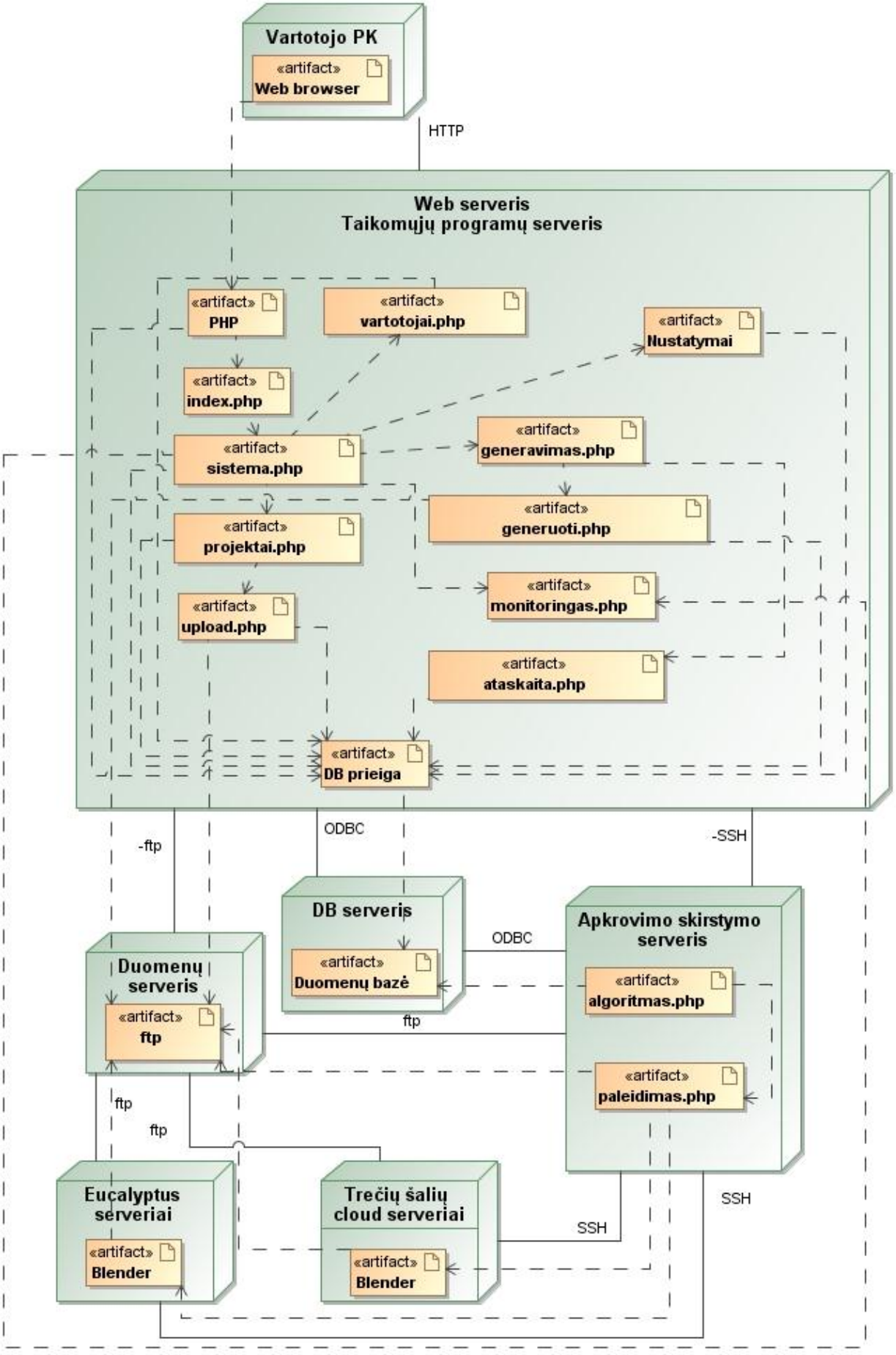
4.1 lentelė. „IS komponentai“

Komponentas	Funkcija
Sistema (matomas)	Pagrindinis sistemos komponentas, kuris yra kaip karkasas, skirtas integruoti ir atvaizduoti likusius sistemos komponentus vienoje integralioje sąsajoje.
Projektai (matomas)	Atvaizduoja duomenų bazėje esančius projektus, kurie įkelti konkretaus vartotojo, bei užkraunamos www serveryje esančios šio projekto nuotraukos. Šiame komponente galima įkelti naują, trinti esamą arba siųsti generuoti pasirinktą projektą.
Generavimas (matomas)	Atvaizduoja duomenų bazėje esančius įrašus apie sugeneruotus, generuojamus ir laukiančius projektus. Čia prognozuojama projekto generavimo trukmė ir rodomas tikrasis generavimo laikas bei vykdymo statusas. Taip pat galima ištrinti eilėje esančius projektus.
Nustatymai (matomas)	Atvaizduoja ir leidžia pakeisti sistemos nustatymus.
Ataskaita (matomas)	Atvaizduoja ataskaitų pasirinkimo langą ir sugeneruoja pasirinktos dienos ataskaitą.
Vartotojai (matomas)	Komponentas skirtas redaguoti, kurti ir trinti vartotojus.
Upload (nematomas)	Skirtas patikrinti įkėlimo failo ir duomenų korektiškumą bei įkelti failą į sistemą ir duomenų bazę. Taip pat šis komponentas automatiškai sugeneruoja 5 iteracinius projekto kadrus, kurie naudojami projektų generavimo laiko prognozavimui ir piešinėlių atvaizdavimui „Projektai“ komponente.
Generuoti (nematomas)	Skirtas surinkti duomenis apie generuoti siunčiamą projektą ir įrašyti į duomenų bazę.
Login (nematomas)	Skirtas patikrinti prisijungimo duomenų korektiškumą.
Trintig (nematomas)	Skirtas trinti generavimus iš sistemos.
Trintip (nematomas)	Skirtas trinti projektus iš sistemos.

Komponentas	Funkcija
DB (nematomas)	Skirtas sistemos komponentų prisijungimui prie duomenų bazės.
Monitoringas (matomas)	Komponentas skirtas debesies veikimo stebėjimui. Jis seka debesyje veikiančių virtualių mašinų apkrovimą. Sukuria ssh tunelį į kiekvieną virtualią mašiną ir įvykdo užklausą apie procesoriaus darbą. Nuskaito gautą informaciją iš tekstinės aplinkos ir priskiria juos kintamiesiems, kurie atvaizduojami informacinės sistemos sąsajoje. Šis komponentas vykdomas jį iššaukus.
Algoritmas (nematomas)	Šis komponentas skirtas tikrinti ar yra naujų nesugeneruotų projektų duomenų bazėje, jeigu yra, jo tikslas nuspręsti kokia eilės tvarka jie bus generuojami. Pasirinktas projektas generavimui yra nukopijuojamas iš katalogo į darbinį generavimo katalogą. Jo pagrindinis tikslas optimizuoti projektų generavimo eilę, kad optimaliai išnaudotų resursus.
Paleidimas (nematomas)	Komponentas, kuris pastoviai tikrina ar kitas vykdomas „Algoritmas“ komponentas neįkėlė naujo projekto į generavimo darbinį katalogą. Naujo generuojamo projekto atveju šis komponentas inicijuoja generavimo debesies sistemoje paleidimą. Kai projektas sugeneruojamas, jis yra konvertuojamas į video avi ir archyvo zip formatų failus. Šie rezultatų failai perkopijuojami į įprastą projekto katalogą. Informacija apie generavimą ir trukmę įrašoma duomenų bazėje ir išvalomas darbinis generavimų katalogas.

4.5.2. Diegimo modelis

Diegimo modelyje (4.16 paveiksle) atvaizduojama vartotojo sąveika per HTTP sąsają su sistema. Taikomųjų programų serveryje realizuotos php klasės, kurios apdoroja duomenis ir realizuoja informacinę sistemą. Informacinė sistema su skaičiavimų serveriais bendrauja per paskirstymo serverį, kuris, savo ruožtu, iš failų serveryje esančių katalogų parsisiunčia paruoštus generuoti projektus bei juose saugo rezultatus. Tiek Eucalyptus privačius, tiek trečių šalių viešus serverius paskirstymo serveris pasiekia ssh tunelio pagalba.



4.16 pav. Diegimo modelis

5. Sistemos realizacija

Kompiuterinės animacijos generavimo sistemą sudaro dvi dalys:

1. Informacinė sistema ir jos valdoma programinė įranga serveriuose,
2. Debesies infrastruktūros realizacija,

Šios dalys realizuojamos atskirais etapais bei įrankiais, todėl toliau aprašomos atskirai.

5.1. IS realizuoti naudota programinė įranga

IS realizuojama panaudojant šiuolaikines programavimo tendencijas ir įrankius:

1. PHP- serverio pusės programavimo kalba,
2. JavaScript – naršyklėje vykdomi skriptai,
3. JQuery – JavaScript biblioteka skirta dinaminiam turinio atvaizdavimui,
4. MySQL – duomenų bazė,
5. Ajax – dinaminio turinio atvaizdavimo iš duomenų bazės technologija,
6. Lightbox – įskiepis naudojantis JQuery skirtas dinaminiam nuotraukų atvaizdavimui,
7. Multiblend – atviro kodo, pagal GNU GPL licencija platinamas paskirstyto Blender PĮ skaičiavimo skriptas.

Jais siekiama sukurti neapkrautą, lengvai suprantamą ir dinamišką aplinką, kaip numatyta sistemai keliamuose nefunkciniuose reikalavimuose.

5.1.1. Duomenų saugos užtikrinimas

Sukurta sistema bus pasiekiama vartotojams internete, todėl padidėja duomenų nutekėjimo rizika. Sistemoje naudojama daugiapakopė apsauga nuo galimo informacijos nutekėjimo.

1. Prisijungimas prie sistemos koduojamas MD5 algoritmu, o informacijos patikrinimui MySQL duomenų bazėje naudojama „htmlspecialchars“ funkcija, kuri apsaugo nuo nesankcionuotų prisijungimų panaudojant SQL kalbos simbolius(pvz.: „ ‘ OR ‘a’='a“),
2. Skirtingi vartotojų lygiai ir aplinkos. Visi vartotojai išskyrus sistemos administratorių gali pasiekti ir modifikuoti tiktai savo sukurtus ar įkeltus duomenis,
3. Sukuriamas sesijos kintamasis ir kiekvieno php failo pradžioje autentifikuojamas vartotojas. Taip apsaugoma nuo nesankcionuoto sistemos naudojimo.

5.1.2. Vartotojo sąsaja

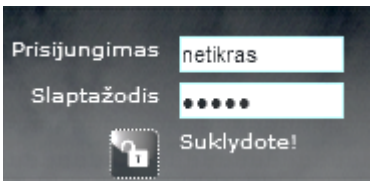
Pagrindiniame lange (5.1 paveikslas) - visada užtikrinamas vieningos sąsajos integralumas ir atvaizdavimas, todėl tiek neprisijungęs, tiek ir prisijungę autorizuoti vartotojai pateikiamą informaciją mato ta pačia struktūra.

5.1 pav. Pagrindinis(prisijungimo) langas

Grafinę sąsają sudaro 3 dinaminiai komponentai:

1. Prisijungimo/vartotojo informacijos komponentas,
2. Meniu juosta,
3. Turinio atvaizdavimo komponentas.

Prisijungimo meniu viršutiniame kairiame kampe dinamiškai patikrina suvestą vartotojo informaciją ir esant neteisingiems duomenims nevykdo naujo puslapio atidarymo komandos, o apie tai informuoja vartotoją (5.2 paveikslas). Prisijungus prie sistemos šis komponentas atvaizduoja prisijungusio vartotojo informaciją (5.3 paveikslas).



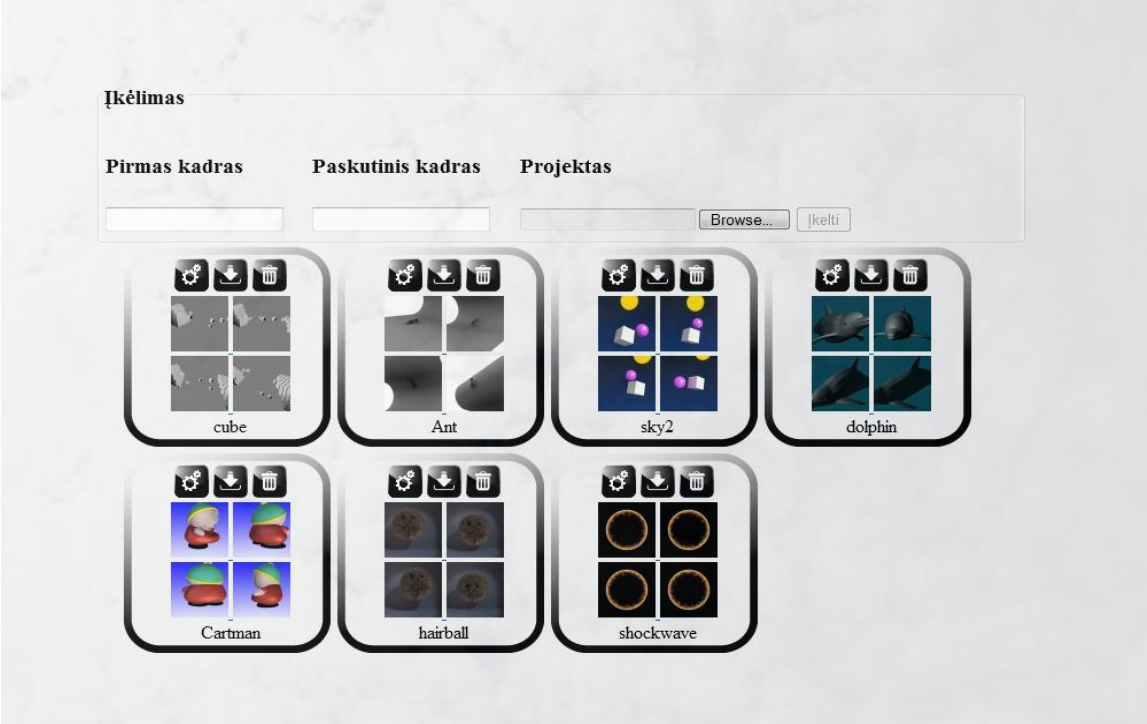
5.2 pav. Prisijungimo komponentas

5.3 pav. Prisijungimo komponentas

Šoninė meniu juosta dinamiškai pasipildo skiltimis, kurios yra skirtos prisijungusiam vartotojui. Ji sudaryta iš intuityvių paveikslėlių, šis meniu taip pat pateikiamas tekstine išraiška apatinėje užduočių juostoje (5.4 pav.).

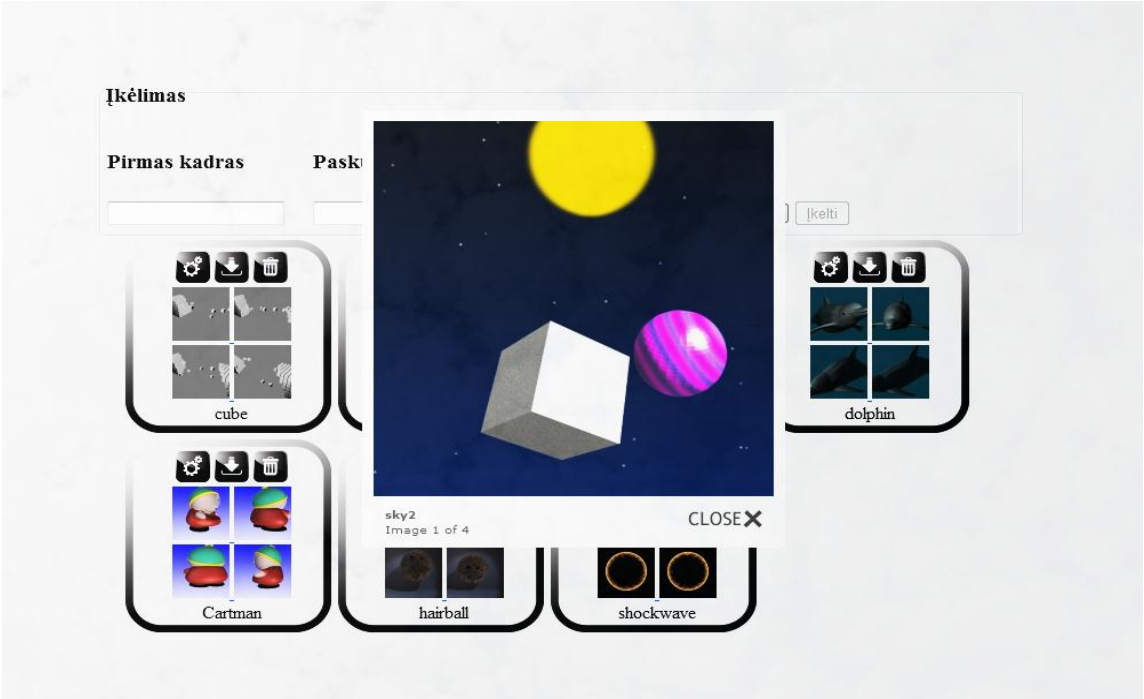
5.4 pav. Apatinis meniu

Pagrindinis turinio atvaizdavimo komponentas skirtas išvedamai informacijai, kuri gali būti tiek tekstinė tiek ir vaizdinė(5.5 paveikslas).



5.5 pav. Informacijos pateikimo komponentas

Patogiam piešinėlių atvaizdavimui yra naudojamas „lightbox“ įskiepis, kuris iššaukiamas ir atvaizduojamas turinio atvaizdavimo komponento viduje (5.6 paveikslas).



5.6 pav. „lightbox“ įskiepis.

5.1.3. Sistemos stebėjimas (monitoringas)

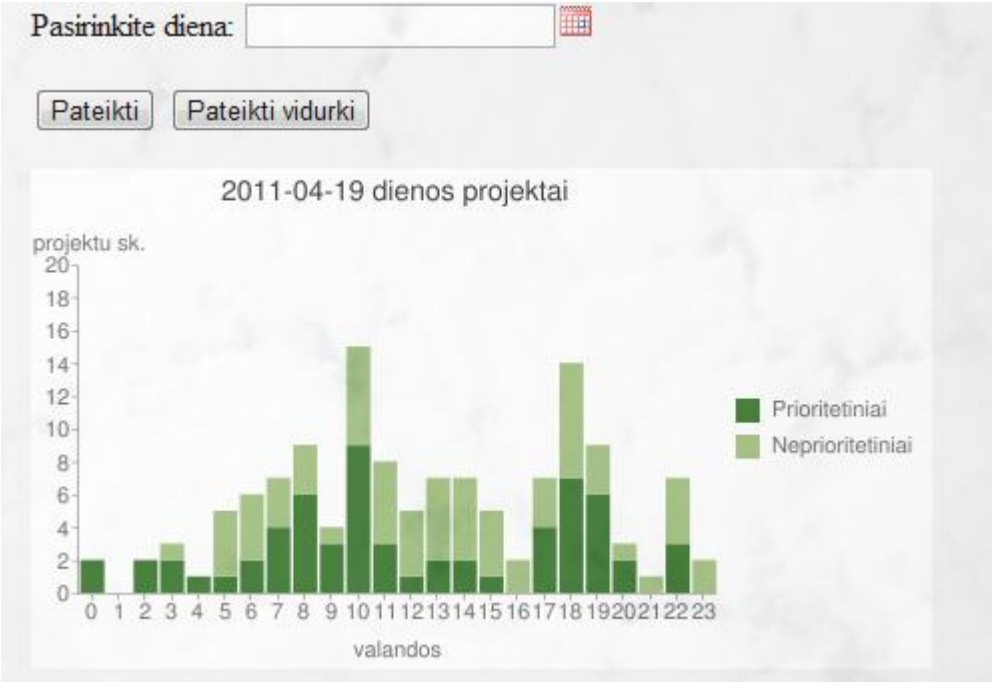
Informacinė sistema valdo skaičiavimo serverius, kuriuose vykdomas generavimas. Todėl realizuotas komponentas, kuris realiu laiku rodo sistemos veikimą. Esant sutrikimams apie tai perspėja administratorių. Administratoriui pateikiama informacija apie valdančio serverio apkrovimą, veikimo laiką, kurį jis veikia be perkrovimo, ir kieto disko talpą. Pagal šiuos duomenis galima spręsti kada tikslinga perkrauti serverį arba atlaisvinti jame daugiau laisvos vietos.

Taip pat pateikiama informacija apie virtualių mašinų veikiančių debesyje procesoriaus apkrovimą. Pagal tai galima stebėti generavimo proceso išlygiagretinimą. Šis komponentas dar atvaizduoja išorinių debesies paslaugos tiekėjų būklę.

Esant sutrikimams ar neatsakant virtualiai mašinai rodomas perspėjimas . Šis komponentas atvaizduojamas 5.7 paveiksle.

5.7 pav. Monitoringo komponentas.

Ataskaitos sistemoje pateikiamos automatiškai sugeneruojant grafikus. Šio komponento pavyzdys pateikiamas 5.8 paveiksle. Sistemos apkrovimo ataskaitos pateikiamos diagramų pavidalu, kuriose išskiriamas aukšto prioriteto (prioritetiniai) ir žemo prioriteto (neprioritetiniai) uždavinių pasiskirstymas paroje.



5.8 pav. Ataskaitų komponentas.

5.2.Debesies infrastruktūros realizacija

Debesies infrastruktūrai realizuoti reikalinga techninė ir programinė įranga aprašomos atskirai. Taip pat buvo sukurtas virtualios mašinos atvaizdas skirtas Blender grafinių projektų generacijai.

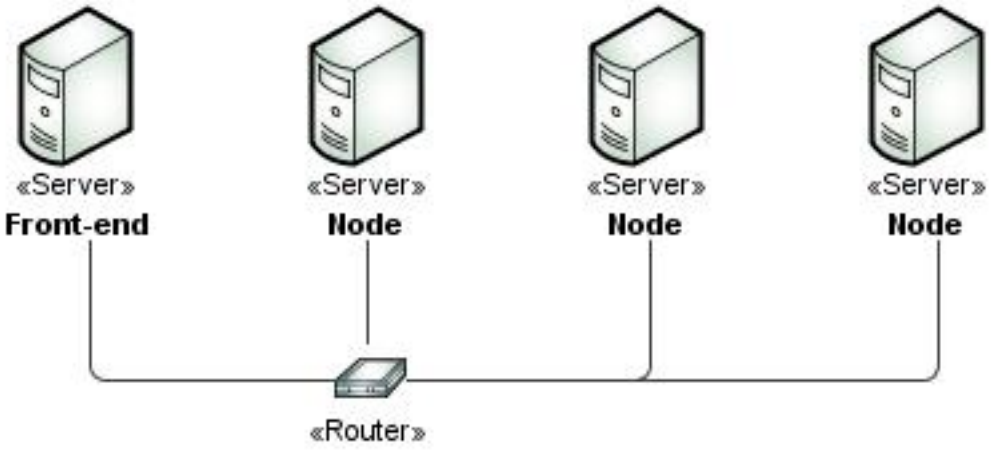
5.2.1. Debesies infrastruktūros diegimui naudota techninė įranga

Debesies infrastruktūra buvo įdiegta Kauno technologijos universiteto Informacinių technologijų plėtros institute. Tam panaudoti 4 identiški kompiuteriai apjungti tame pačiame potinklyje. Jų duomenys – Intel Pentium IV 2,66 GHz centriniai procesoriai, 512Mb RAM bei 40 Gb kietieji diskai.

5.2.2. Debesies infrastruktūrai naudota programinė įranga

Debesies aplinkai kurti, pagal analizėje padarytas išvadas, buvo pasirinkta Eucalyptus programinė įranga. Naudota Eucalyptus 1.5.2 versija. Ji įdiegta į standartinės konfigūracijos 4

Linux CentOS 5.4 operacinę sistemą naudojančius serverius. Eucalyptus 1.5.2 susideda iš trijų pagrindinių komponentų – eucalyptus-CL, eucalyptus-CC ir eucalyptus-NC, kurie gali būti suinstaliuoti atskiruose serveriuose. Tačiau šioje sistemoje dėl kompiuterinių resursų stokos eucalyptus-CL ir eucalyptus-CC dalys buvo realizuotos viename serveryje ir sudaro „Front-end“ dalį. Eucalyptus-NC dalys įdiegtos trijuose serveriuose kaip pavaizduota 5.9 paveiksle.



5.9 pav. Eucalyptus diegimas

Šioje sistemoje turime 3 Node serverius, kuriuose veikia virtualios mašinos. Debesies valdymui ir klientui naudojami Amazon EC2 debesies įrankiai, kurie leis praplėsti esamą sistemą nuomotais resursais, kurie yra valdomi Amazon AMI įrankiais. Pasirinkto versijos :

- ec2-ami-tools-1.3-26357;
- ec2-api-tools-1.3-30349.

Debesies programinės įrangos instaliavimo ir konfigūravimo vartotojo vadovas pateikiamas antrame priede

5.2.3. Virtualios mašinos atvaizdo kūrimas

Dėl savo suderinamumo su Python 2.6 versija, kuri reikalinga Blender 2.49 veikimui, debesies virtualios mašinos operacine sistema buvo pasirinkta Ubuntu 9.04. Programinė įranga į operacinę sistemą integruota naudojant EC2 API ir AMI įrankius. Virtualiose mašinose pagal nutylėjimą įjungtas ssh prisijungimas, kuris naudojamas projektų siuntimui ir rezultatų grąžinimui į Front-end dalyje veikiantį generavimo valdiklį.

5.3.Sistemos testavimas

Tik ištestuota ir korektiškai veikianti sistema gali užtikrinti stabilų darbą. Tam parengtas testavimo planas į kurį įeina:

- IS panaudojimo atvejų testavimas,
- Sistemoje realizuoto uždavinių paskirstymo veikimo testavimas.

5.3.1. IS panaudojimo atvejų testavimas

Testuojami visi 14 panaudojimo atvejų. 5.1 lentelėje pateikiamas panaudojimo atvejo numeris, juo atliekamas veiksmas, laukiamas rezultatas, gautas rezultatas bei rezultatų atitikimas arba veiksmai neatitikimams taisyti.

5.1 lentelė. „IS testavimas“

Nr.	Panaudojimo atveju atliekamas veiksmas	Laukiamas rezultatas	Gautas rezultatas	Rezultatų atitikimas, korekcija ir pastabos
1	Įkeliamas generavimo projektas į sistemą.	Projektas bus įkeltas į failų serverį, DB atsiras įrašas, bus sugeneruoti iteraciniai projekto kadrai ir jis bus matomas projektų sąraše.	Projektas įkeltas į serverį, DB atsirado įrašas, buvo sugeneruoti iteraciniai projekto kadrai ir jis yra matomas projektų sąraše.	Rezultatai atitiko.
2	Projektas trinamas.	Projektas nebebus rodomas vartotojui, DB bus pažymėtas kaip ištrintas bei bus išvalyta su juo susijusi informacija failų serveryje.	Projektas neberodomas vartotojui, DB pažymėtas kaip ištrintas, failų serveryje liko projekto generuotos animacijos.	Rezultatai atitiko dalinai. Buvo pataisytas trynimo skriptas tam, kad rezultatai sutaptų.
3	Peržiūrimas generuojamų projektų statusas.	Sistema pateikia sąrašą generuojamų projektų ir rodo jų dabartinį statusą.	Atvaizduotas projektų sąrašas su jų statusais.	Rezultatai atitiko dalinai. Esant dideliam kiekiui projektų sąrašas yra per ilgas patogiam žiūrėjimui, todėl įvedamas puslapiavimas.
4	Startuojamas projekto generavimas.	Nurodoma projekto atkarpa kuri turi būti sugeneruota, duomenų bazės „generavimas“	Duomenų bazėje atsirado įrašas apie generuotiną projektą su visa reikiama informacija. Projektas	Rezultatai atitiko.

Nr.	Panaudojimo atveju atliekamas veiksmas	Laukiamas rezultatas	Gautas rezultatas	Rezultatų atitikimas, korekcija ir pastabos
		lentelėje atsiranda įrašas. Šiame įraše taip pat turi būti informacija apie įkėlimo laiką, galiojantį planą, vartotoją, generavimo prognozę. Taip pat šis projektas turi atsirasti generuojamų projektų sąraše.	vartotojui matomas generavimų lange.	
5	Parsiunčiamas video arba kadrų archyvo failas.	Vartotojui pasirinkus projektą, sugeneruojama nuoroda į parsisiuntimo failą.	Nuoroda sugeneruota teisingai ir failas pradėjo siųstis.	Rezultatai atitiko.
6	Pasirenkamas mokėjimo planas.	Vartotojas pasirenka mokėjimo planą iš tuo metu sistemoje galimų. Apie tai įrašomas įrašas DB „vartotojai“ lentelėje.	Pasirinktas planas įrašytas DB „vartotojai“ lentelėje.	Rezultatai atitiko.
7	Trinami sugeneruoti vaizdai.	Failų serveryje ištrinami animacijos ir kadrų archyvo failai, o DB šis generavimas pažymimas kaip ištrintas.	Failų serveryje ištrinti animacijos ir kadrų archyvo failai, o DB šis generavimas pažymėtas kaip ištrintas.	Rezultatai atitiko.
8	Sukuriamas vartotojas.	Užpildžius vartotojo formą duomenų DB lentelėje „vartotojai“ atsiranda naujas įrašas. Naujas vartotojas gali pradėti naudotis sistema.	Sukurtas naujas vartotojas su informacija DB. Jis gali naudotis sistema.	Rezultatai atitiko.
9	Redaguojamas vartotojas.	Sistema pateikia vartotojo duomenis, kurie yra keičiami ir išsaugomi DB.	Sistema pateikė vartotojo duomenis, kurie sėkmingai išsaugoti DB.	Rezultatas atitiko dalinai. Aptikta galimybė pakeisti vartotojo identifikacijos lauką, kas vėliau gali sukelti nesuderinamumą. Šio lauko keitimo galimybė apribota.

Nr.	Panaudojimo atveju atliekamas veiksmas	Laukiamas rezultatas	Gautas rezultatas	Rezultatų atitikimas, korekcija ir pastabos
10	Trinamas vartotojas.	Sistema pateikia sąrašą vartotojų. Pasirinktas vartotojas ištrinamas. Iš sistemos ištrinamas įrašas apie jį, o iš failų serverio jo projektų ir generavimų katalogas.	Sistemoje ištrinta informacija apie vartotoją, o failų serveryje nebeliko jo duomenų.	Rezultatai atitiko.
11	Stebėti sistemos darbą.	Sistema pateikia informaciją apie valdantį serverį ir VM veikiančias debesyje.	Informacija apie valdantį serverį ir VM apkrovimą atvaizduota. Pastaba: atvaizdavimas užtruko apie 10 sekundžių.	Rezultatai atitiko. Užklausa trunka ilgai, nes tikrinamas kiekvieno serverio aktyvumas nuotoliniu būdų. Turimuose resursuose šio proceso paspartinti neįmanoma.
12	Keisti sistemos nustatymus.	Sistema pateikia dabartinių ir prieš tai naudotų nustatymų šablonus, pasirinkus arba sukūrus naują jis išsaugomas DB ir aktyvuojamas.	Pasirinkus anksčiau taikytą, o taip pat ir sukūrus naują nustatymų rinkinį šis tampa aktyvus ir išsisaugo DB.	Rezultatai atitiko.
13	Generuojama ataskaita.	Sistema pagal pasirinktą datą pateikia ataskaitą apie tą dieną generuotus projektus.	Buvo suformuota ataskaita pagal pasirinktą dieną.	Rezultatai atitiko.
14	Koreguojamas mokėjimo planas.	Pakeista mokėjimo plano informacija išsaugoma DB ir tampa matoma visiems vartotojams.	Informacija apie pakeistą mokėjimo planą išsaugota DB ir matoma visiems vartotojams.	Rezultatai atitiko.

Atliekant testavimą pagal PA buvo ištaisyti neatitikimai ir šiek tiek patobulintas iš pradžių nenumatytas funkcionalumas.

5.3.2. Sistemoje realizuoto uždavinių paskirstymo veikimo testavimas

Grafikos generavimo sistemos uždavinių apkrovimo balansavimo algoritmas turi tenkinti 4.3 dalyje iškeltas sąlygas, todėl sudaromi testai šių sąlygų korektiškumui tikrinti:

1. Aukšto prioriteto skubūs grafikos uždaviniai turi būti generuojami iš karto, patikrinimui į sistemą įkeliama 10 aukšto prioriteto uždavinių seka ir stebimas jų generavimas;
2. Žemo prioriteto uždaviniai nėra skubūs ir gali laukti tam tikrą, iš anksto nustatytą, laiko tarpą. Patikrinimui į sistemą įkeliama 5 aukšto ir 5 žemo prioriteto uždavinių seka ir stebimas jų generavimas;
3. Atidėtų žemo prioriteto uždavinių pirmumas tarpusavyje priklauso nuo arčiausio laiko likusio iki privalomos jo generavimo pradžios. Patikrinimui į sistemą įkeliami 5 žemo prioriteto uždaviniai ir stebima jų generavimo tvarka;
4. Žemo prioriteto uždavinys nors ir pasibaigus maksimaliam atidėjimo laikui vis tiek yra žemesnio prioriteto negu aukšto lygio uždavinys. Patikrinimui į sistema įkeliama 5 žemo prioriteto uždavinių seka su numatytu 30 minučių laukimu. Po 30 minučių įkeliama 5 aukšto prioriteto uždavinių seka ir stebimas jų generavimo eiliškumas.
5. Priklausomai nuo sistemos nustatymų, gali būti privačiame debesyje iš anksto rezervuota vieta aukšto prioriteto uždaviniams. Šios funkcijos tikrinimui sistemoje pagal 2 mėnesių apkrovimo duomenis sugeneruojama aušto prioriteto uždavinių rezervacija. Sistemoje vykdomi visos paros uždaviniai ir stebima ar aukšto prioriteto uždaviniai panaudoja rezervacijas.
6. Žemo prioriteto uždaviniai niekada negali užimti rezervuotos vietos, Šios funkcijos tikrinimui sistemoje pagal 2 mėnesių apkrovimo duomenis sugeneruojama aušto prioriteto uždavinių rezervacija. Sistemoje vykdomi visos paros uždaviniai ir stebima ar žemo prioriteto uždaviniai neužima rezervuotų vietų.

Rezultatų atvaizdavimui sudaroma 5.2 lentelė.

5.2 lentelė. „Paskirstymo testavimas“

Testo nr.	Laukiamas rezultatas	Gautas rezultatas	Rezultatų atitikimas, korekcija ir pastabos
1	Uždaviniai privačiame debesyje bus generuojami iškart, o jeigu nėra laisvų resursų bus siunčiami į viešą debesį.	Uždaviniai privačiuose resursuose generuoti iškart. Kai baigėsi privatūs resursai, toliau buvo naudojami vieši.	Rezultatai atitiko.
2	Žemo prioriteto uždaviniai lauks eilėje, kol atsilaisvins resursai. Tačiau lauks ne ilgiau negu numatytas 30 minučių.	Žemo prioriteto uždaviniai laukė laisvų resursų, tačiau buvo peržengta ilgiausio laukimo riba.	Rezultatas neatitiko. Buvo pakoreguota algoritmo dalis atsakinga už laukimo laiko įvertinimą.
3	Uždaviniai bus generuojami ta pačia tvarka, kaip buvo įkelti į sistemą.	Uždaviniai sugeneruoti ta pačia tvarka kaip įkelti į sistemą.	Rezultatas atitiko.
4	Kai aukšto ir žemo projektų laukimo laikas sutampa, t.y. jis lygus 0, pirmiau generuojami aukšto prioriteto uždaviniai.	Visais atvejais pirmiau generuojami aukšto prioriteto uždaviniai	Rezultatas atitiko.
5	Į sistemą atėjus naujam aukšto prioriteto uždaviniui tuo metu esanti rezervacija ištrinama.	Esamos rezervacijos buvo sėkmingai ištrintos jų metu atėjus naujiems aukšto prioriteto uždaviniams.	Rezultatas atitiko.
6	Į sistemą atėjus žemo prioriteto uždaviniui rezervuotos vietos išlieka ir yra nenaudojamos.	Rezervuotos vietos buvo nepanaudotos, kai į sistemą įkeliami žemo prioriteto uždaviniai.	Rezultatas atitiko.

Atlikus realizuotos sistemos testavimą pašalintos pastebėtos klaidos, todėl sukurta sistema pilnai atitinka iškeltus reikalavimus ir veikia korektiškai. Tačiau buvo pastebėtas veikimo sutrikimas nesusijęs su sukurta programine įranga. Dėl per mažos serveriuose esančios operatyvinės atminties generuojant sudėtingus projektus pastebėti sistemos veikimo sutrikimai.

6. Eksperimentas

6.1.Paskirstyto skaičiavimo bei virtualizacijos įtakos Blender projektų generavimui eksperimentas

Kompiuterinės grafikos generavimas yra tiesiogiai priklausomas nuo turimų skaičiavimo resursų kiekio ir pajėgumo. Šiuo tyrimu siekiama palyginti įvairius generavimo atvejus, nustatyti optimalų atvejį bei palyginti laiko nuostolius tarp fizinių ir virtualiu serverių. Tokiu būdu bus išaiškintas savo turimų resursų virtualizacijos naudingumas.

Eksperimente naudojama įranga ir metodika:

Šiam tyrimui naudojami trys Pentium IV 2.66 GHz 512 Mb RAM specifikacijos serveriai. Du iš jų naudojami generavimui tiek fiziniuose resursuose, tiek ir juose veikiančiose virtualiose mašinose. Vienas serveris naudojamas generavimo proceso valdymui.

Tyrimo metodika:

Blender programinės įrangos projektų generavimas buvo atliekamas tiek viename serveryje, tiek išlygiagretinus skaičiavimą dviejuose. Taip pat buvo bandoma generuoti fiziniuose ir virtualiuose resursuose. Kiekvieno generavimo metu fiksuota uždavinių skaičiavimo trukmė, kuri vėliau konvertuota į santykį ir lyginama tarpusavyje.

Šiam tyrimui naudoti trys Blender animacijos failai:

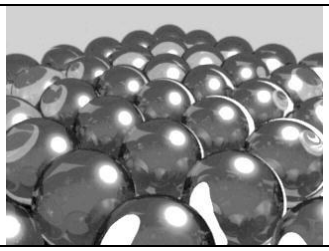
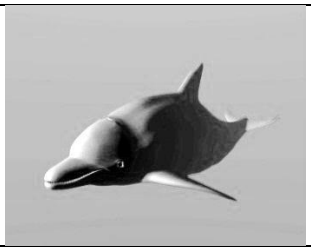
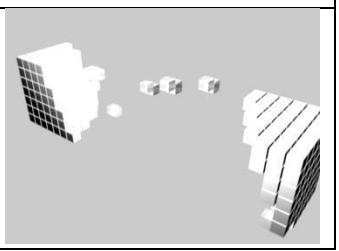
1. „raytrace.blend“ - gautas iš Multimedijos inžinerijos katedros dėstytojo Algirdo Noreikos,
2. „dolphin.blend“ - pavyzdinis projektas paimtas iš www.blender.org tinklalapio,
3. „cube.blend“ - pavyzdinis projektas paimtas iš nemokamos mokomosios medžiagos internetinėje erdvėje.

Jais siekiama patikrinti, kaip skaičiavimo resursų išnaudojimą įtakoja trijų tipų uždaviniai. Tam parinkti skirtingi Blender animacijų nustatymai, kurie apkrautų procesorių skirtingais lygmenimis:

1. Silpnai – procesoriaus apkrovimas generuojant failą apie 30%;
2. Vidutiniškai – procesoriaus apkrovimas generuojant failą apie 60%;
3. Stipriai – procesorius apkrautas 100%.

Šių projektų nustatymai aprašomi 6.1 lentelėje.

6.1 lentelė. testiniai failai

Failo pavadinimas:	**raytrace.blend**	**dolphin.blend**	**cube.blend**
Bendras sudėtingumas:	Sudėtingas (apkrauna procesorių 100%)	Vidutinis (apkrauna procesorių 60%)	Nesudėtingas (apkrauna procesorių 30%)
Kadro dydis:	320x240	720x576	640x480
Kadrų skaičius:	4	222	620
Iliustracija:			

Kiekvienas failas yra generuojamas šešiais skirtingais atvejais (6.2 lentelė). Atvejo pavadinime pirmiausiai nurodoma kiek virtualių mašinų naudojama skaičiavimams. Multiblend – skaičiavimai inicijuojami iš valdančio serverio. Skliausteliuose pateikiamas realus fizinių serverių skaičius.

6.2 lentelė. Generavimo atvejai

Atvejo pavadinimas	**Atvejo paaiškinimas**
1 fizinis serveris (etalonas)	Generavimas atliekamas viename fiziniame serveryje, o gautasis rezultatas laikomas etalonu lyginant su kitais duomenimis.
1 VM (1 fizinis)	Generavimas vykdomas 1 virtualioje mašinoje. Virtuali mašina veikia viename fiziniame serveryje.
1 VM multiblend(1 fizinis)	Generavimas vykdomas 1 virtualioje mašinoje nuotoliniu būdu. Virtuali mašina veikia viename fiziniame serveryje.
2 VM multiblend(1 fizinis)	Generavimas vykdomas 2 virtualiose mašinose nuotoliniu būdu. Virtualios mašinos veikia viename fiziniame serveryje.
2 VM multiblend(2 fiziniai)	Generavimas vykdomas 2 virtualiose mašinose nuotoliniu būdu. Virtualios mašinos veikia dviejuose fiziniuose serveriuose.
4 VM multiblend(2 fiziniai)	Generavimas vykdomas 4 virtualiose mašinose nuotoliniu būdu. Virtualios mašinos veikia dviejuose fiziniuose serveriuose.

Laukiami rezultatai:

1. Dėl virtualizacijai naudojamų kompiuterio resursų, skaičiavimai turėtų sulėtėti juos atliekant virtualiose mašinose.
2. Viename serveryje naudojant dvi virtualias mašinas rezultatai laukiami šiek tiek prastesni negu naudojant vieną virtualia mašiną. Tai lemia didesnis resursų kiekis naudojamas virtualizacijai.
3. Tikimasi, kad 2 fiziniai serveriai skaičiuos 2 kartus greičiau negu vienas.
4. Iš aprašytų atvejų optimaliausias rezultatas laukiamas naudojant “ 2 VM multiblend(2 fiziniai)“ variantą

5. Pavertus generavimo laiką į santykį su etalonu tikimasi, kad visų failų rezultatai bus skirtingi dėl skirtingo jų sudėtingumo.

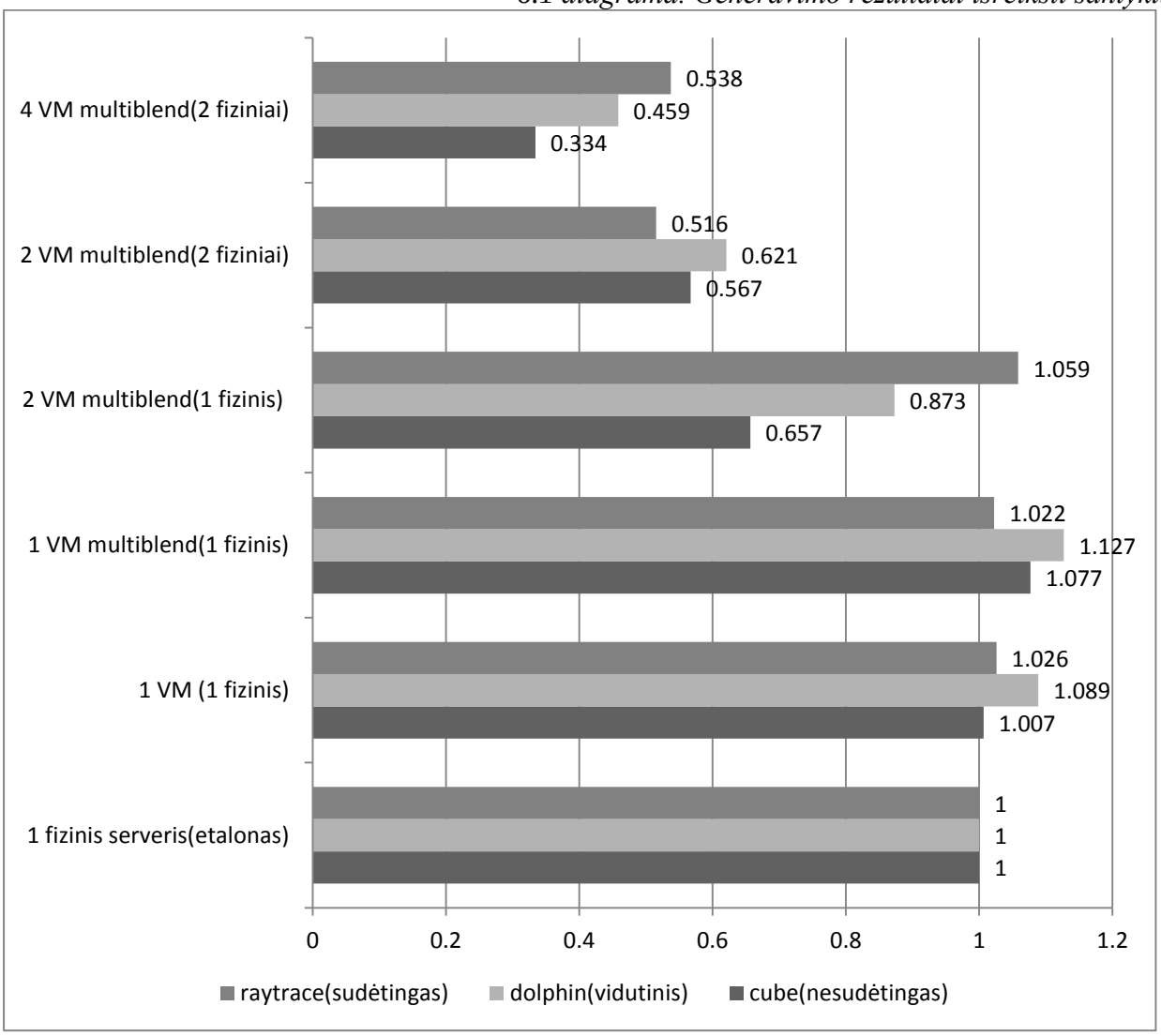
Gauti rezultatai:

Generavimo rezultatai sekundėmis pateikiami 6.5 lentelėje. Norint objektyviai palyginti kaip uždavinių sudėtingumas įtakoja resursų išnaudojimą, rezultatai konvertuojami į santykį su etaloniniu fizinio serverio rezultatu ir pateikiami 6.1 diagramoje. Rezultatai virš 1 turi mažesnį efektyvumą negu etalonas, o mažesni negu 1 – didesnį.

6.5 lentelė. Generavimo rezultatai

Atvejis	cube.blend	dolphin.blend	raytrace.blend
1 fizinis serveris(etalonas)	1197	630	3088
1 VM (1 fizinis)	1205	686	3168
1 VM multiblend(1 fizinis)	1289	710	3157
2 VM multiblend(1 fizinis)	786	550	3269
2 VM multiblend(2 fiziniai)	679	391	1592
4 VM multiblend(2 fiziniai)	400	289	1660

6.1 diagrama. Generavimo rezultatai išreikšti santykiu



Gautų rezultatų apibendrinimas

1. Kaip ir tikėtasi fiziniame kompiuteryje veikiančioje virtualioje mašinoje generavimas trunka trumpiau negu būtų atliekamas fiziniuose resursuose.
2. Generuojant nesudėtingus kadrus kompiuterio procesorius yra neišnaudojamas 100%, todėl dvi virtualios mašinos veikiančios viename kompiuteryje procesorių išnaudoja kur kas optimaliau, todėl generavimo laikas yra geresnis negu fiziniuose resursuose.
3. Generuojant sudėtingus kadrus, kurie išnaudoja 100% procesoriaus galios, generavimo laikas fiziniuose serveriuose buvo greitesnis negu juose veikiančiose virtualiose mašinose. Tai įtakoja virtualizacijos procesas, kuris naudoja procesoriaus resursą.
4. Išlygiagretinus generavimo procesą dviejuose serveriuose, kaip ir tikėtasi generavimo trukmė sumažėjo dvigubai, taigi skaičiavimo mašinų skaičius yra tiesiogiai proporcingas generavimo laikui.
5. Iš aprašytų atvejų optimaliausias variantas tapo „4 VM multiblend(2 fiziniai)“. Nors generuojant sudėtingus failus gaunamas šiek tiek prastesnis rezultatas, tačiau jeigu animacija yra nesudėtinga – generavimo sparta gali padidėti net iki trijų kartų.

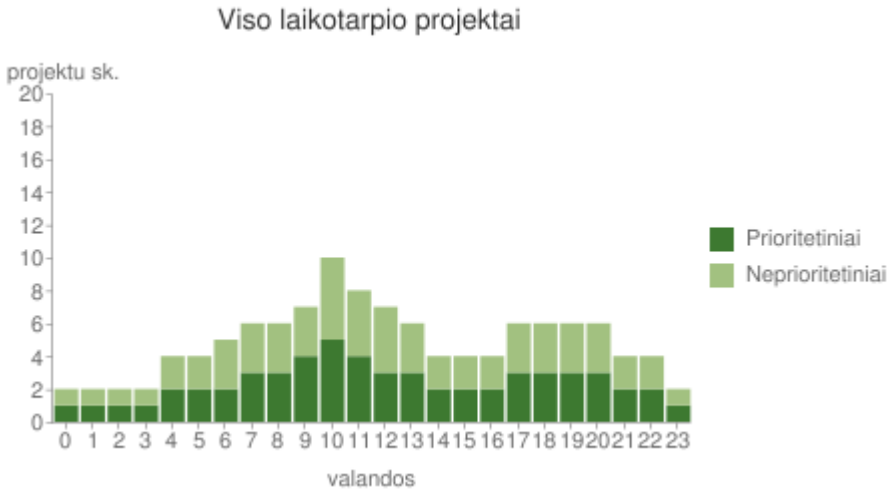
6.2.Grafikos generavimo uždavinių apkrovimo balansavimo algoritmo eksperimentas

Šio eksperimento tikslas ištirti, kaip realizuotas uždavinių srauto paskirstymo algoritmas įtakoja uždavinių generavimą, viešo debesies naudojimą ir privataus debesies išnaudojimą. Grafikos generavimo uždavinių apkrovimo balansavimo algoritmas tiriamas ir vertinamas trimis aspektais:

- tiriama dviejų prioritetų uždavinių srautų įtaka generavimui,
- analizuojamas realizacijos koeficiento kitimo poveikis privataus debesies išnaudojimui,
- tiriama srauto dydžio kitimo įtaka sistemos darbui.

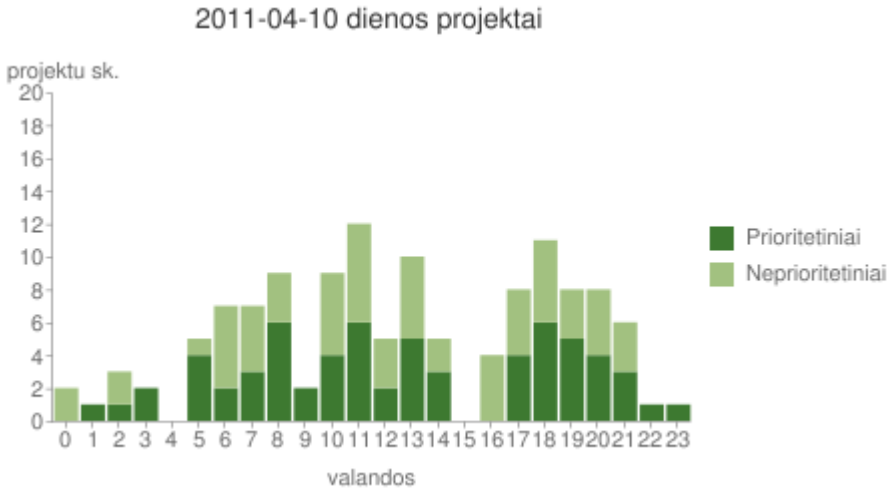
Eksperimentui vykdyti buvo sukurti papildomi modeliai leisiantys tiksliai ir vienodai apkrauti sistemą kaskart atliekant matavimus:

- Apkrovimo generavimo komponentas - skirtas imituoti sistemingą ir periodišką vartotojų pateikiamų užduočių srautą. Naudojant šį modelį buvo sugeneruoti 2 mėnesių laikotarpio grafikos generavimo sistemos apkrovimo duomenys. Šie duomenys bus naudojami ateinančių projektų prognozavimui ir laisvos vietos jiems rezervavimui. Dviejų mėnesių uždavinių apkrovimo vidurkis pateikiamas 6.1 paveiksle.
- Uždavinių komponentas - skirtas iš anksto sugeneruotu ir pastoviu grafiku į sistemą įkelti aukšto ir žemo prioriteto uždavinius.



6.1 pav. Suvidurkintas 2 mėnesių sistemos apkrovimas

Dviejų srautų uždavinių įtakai tirti buvo simuliuojamas vartotojų naudojimasis sistema. Į sistemą 126 kartus dienoje pateikiamas vienas uždavinys, kurio generavimas viename kompiuteriniame vienete trunka apytiksliai 30 minučių. Šių uždavinių pateikimo periodiškumas ir intensyvumas pateikiamas 6.2 paveiksle.



6.2 pav. Sistemos bandymuose naudojamas dienos apkrovimas

Pasirinkti trys bandymo scenarijai:

1. Visi į sistemą įkelti projektai turi būti generuojami iš karto,
2. Į sistemą pateikiami dviejų srautų uždaviniai (6.2 paveikslas). Aukšto prioriteto uždaviniai turi būti generuojami iš karto ir turi pirmenybę prieš žemo prioriteto uždavinius. Žemo prioriteto uždaviniai, atvejais kai nėra laisvų resursų, gali laukti iki 2 h,
3. Į sistemą keliami dviejų srautų uždaviniai kaip ir antrame atvejyje, tačiau įvedama aukšto prioriteto uždavinių prognozė ateityje. Ji atliekama pagal statistinius dviejų mėnesių duomenis pateiktus 6.1 paveiksle. Tokiu būdu į priekį rezervuojami resursai numatomiem aukšto prioriteto uždaviniams. Žemo prioriteto uždavinys negali užimti rezervuotos vietos.

Laukti rezultatai:

- Pirmasis atvejis bus laikomas etalonu. Atlikus šį bandymą bus matomas uždavinių kiekio pasiskirstymas tarp viešo ir privataus debesies.
- Antruoju atveju tikimasi, jog žemo prioriteto uždaviniai nusikels į laikotarpius kai sistemoje yra prastovos. Todėl uždavinių išsiuntimo į viešą debesį kiekis sumažės.
- Trečiuoju atveju įvedama rezervacija aukšto prioriteto projektams. Tokiu būdų sistema apsisaugo nuo galimo resursų apkrovimo žemo prioriteto uždaviniais, kai netrukus gali ateiti aukšto prioriteto. Toks scenarijus turėtų dar labiau sumažinti išsiunčiamų projektų į viešą debesį kiekį.

Gauti rezultatai:

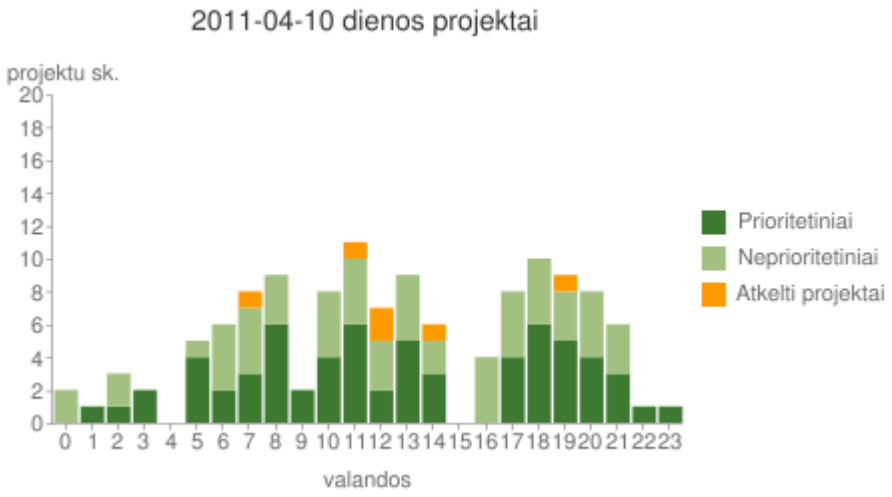
Gauti rezultatai aprašomi lentele:

Bandymo nr.	Uždavinių kiekis privačiame debesyje	Uždavinių kiekis viešame debesyje	Viešo debesies naudojimo sumažėjimas procentais
1	85	41	-
2	95	31	15%
3	99	27	34%

Tarpinių rezultatų analizė:

Pirmuoju bandymu buvo gauti etaloniniai rezultatai ir uždavinių generavimo pasiskirstymas tarp viešo ir privataus debesies standartiniu atveju.

Antrajame bandyme panaudotas dviejų srautų modelis leido sumažinti viešo debesies naudojimą 15%, nes žemo prioriteto uždaviniai buvo atidėti vėlesniam vykdymui, kai privatūs resursai buvo užimti. Šiuo bandymu atidėtus projektus galima pavaizduoti 6.3 grafike.



6.3 pav. Uždavinių atidėjimas naudojant du srautus

Atidėjimo laikai buvo pakankamai maži, todėl užduočių vykdymas nepersikėlė į sekančią valandą. Taip pat pastebėta, jog žemo prioriteto uždaviniai dažnai užimdavo resursus ir nauji į sistemą įkelti aukšto prioriteto uždaviniai buvo išsiunčiami į viešą debesį.

Trečiame bandyme buvo įvestas rezervacijos faktorius. Tai funkcija, kuri prognozuodama pagal 6.1 paveiksle pateiktus statistinius duomenis vykdo resursų rezervaciją aukšto prioriteto uždaviniams. Tokiu būdų buvo išspręsta antrajame bandyme pastebėta problema, kai žemo prioriteto uždaviniai užimdavo resursus ir aukšto lygio nauji uždaviniai buvo siunčiami į viešą debesį. Kas lėmė net 34% sumažėjusį viešo debesies naudojimą lyginant su etalonu. Taip pat smarkiai išaugo žemo prioriteto uždavinių laukimo laikas.

Antrojoje eksperimento dalyje vertinama rezervacijos įtaka. Grafikos generavimo sistemos nustatymuose yra numatytas procentinis rezervuojamų aukšto prioriteto uždavinių kiekio koeficientas. Šioje dalyje buvo atlikti ir palyginti su jau gautaisiais dar 3 bandymai:

4. bandymo metu rezervacijos koeficientas buvo - 25%,
5. bandymo metu rezervacijos koeficientas buvo - 50%,
6. bandymo metu rezervacijos koeficientas buvo -75%.

Laukti rezultatai:

Daroma prielaida, kad mažinant rezervacijos koeficientą - mažės ir privataus debesies utilizavimas. Kiekvienu atveju vis daugiau projektų bus siunčiama į viešą debesį, nes aukšto lygio projektų vieta bus užimta.

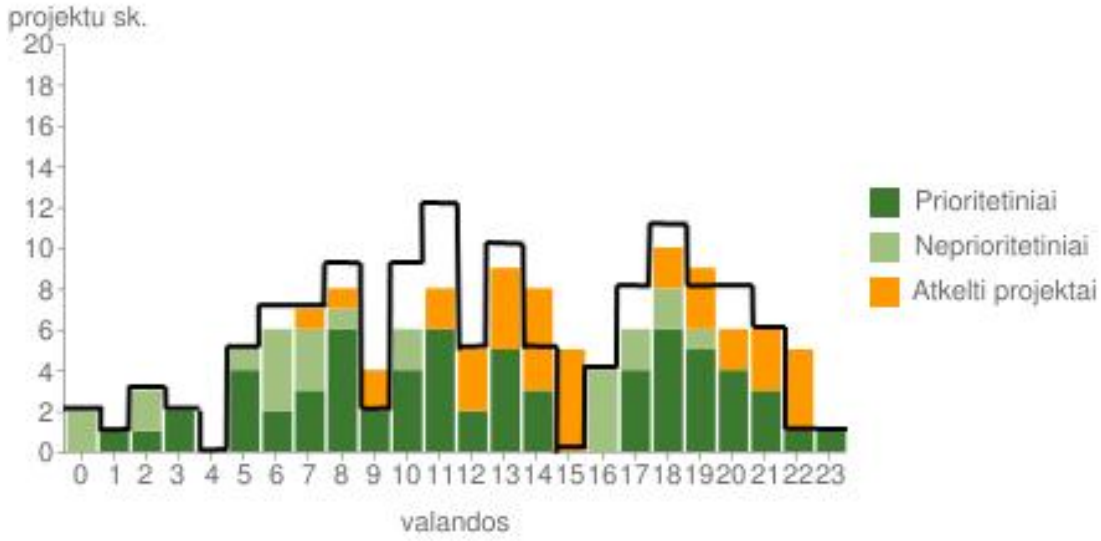
Gauti rezultatai:

Gauti rezultatai lyginami su prieš tai gautais ir aprašomi lentele:

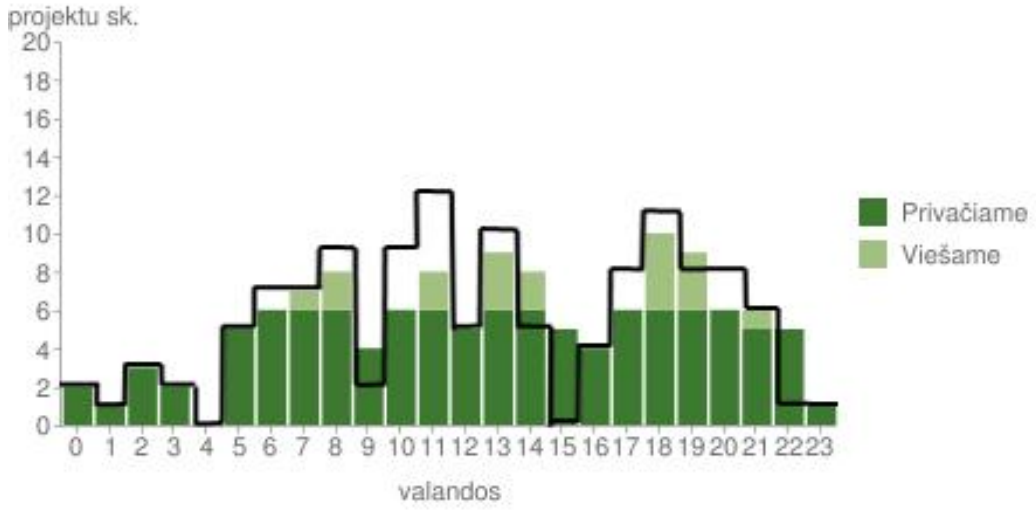
Bandymo nr.	Rezervacija procentais	Uždavinių kiekis privačiame debesyje	Uždavinių kiekis viešame debesyje	Viešo debesies naudojimo sumažėjimas procentais
1	-	85	41	-
4	25%	97	29	29%
5	50%	108	18	56%
6	75%	104	22	46%
3	100%	101	27	39%

Rezultatų analizė:

Pastebėta, jog esant mažesniam rezervacijos koeficientui gaunami geresni rezultatai. Taip nutinka, nes pateikiamas uždavinių srautas nėra identiškas vidutiniam srautui. Analizuojant duomenis pastebėta, kad ne visos rezervacijos aukšto prioriteto uždaviniams panaudojamos. Todėl sistemoje atsiranda prastovos. Su 50% rezervacijos koeficientu buvo gauti geriausi rezultatai. Uždavinių atidėjimo ir pasiskirstymo grafikas pateikiamas 6.4 paveiksle, kuriame juodas kontūras žymi uždavinių srautą, o legendoje nurodytos spalvos prioritetinius uždavinius, neprioritetinius uždavinius ir neprioritetinius atidėtus uždavinius. Išsiųstų į viešą ir generuotų privačiame debesyse uždavinių pasiskirstymas pateikiamas 6.5 paveiksle, čia juodas kontūras žymi uždavinių srautą.



6.4 pav. Uždavinių atidėjimas naudojant 50% rezervaciją



6.5 pav. Uždavinių generavimo pasiskirstymas naudojant 50% rezervaciją

Iš 6.4 paveiksle pateikto grafiko matome, kaip realizuotas algoritmas atideda neprioritetinius uždavinius į prastovų laikotarpius. 6.5 paveikslo diagrama patvirtina algoritmo veikimo korektiškumą. Šiame bandyme didžiausias galimas sugeneruoti privačiuose resursuose projektų skaičius buvo - 6. Naudojant siūlomą algoritmą privatūs resursai išnaudojami maksimaliai ir tik tada kreipiamasi į viešą debesį.

Panaudojus šį apkrovimo balansavimo algoritmą bendrame naudotų resursų kiekyje viešo debesies resursų dalis sumažėjo nuo 32.5% iki 14% . Gautieji rezultatai yra panašūs į 4.1 dalyje aprašyto simuliuoto tokio algoritmo veikimo rezultatus. Tačiau šiuo atveju sistemą veikė realūs veiksniai, kurie simuliuojant veikimą buvo nevertinami - užduočių siuntimo ir gavimo iš serverių laikas, tinklo pralaidumas ir apkrautumas, serverių procesoriaus apkrautumas, virtualizuotų resursų įtaka. Todėl sistemos veikimas ištirtas nuodugniai, o gauti rezultatai patvirtina siūlomo algoritmo korektiškumą. Siūlomas sistemos veikimo modelis efektyviai pagerina nuosavų resursų išnaudojimą ir yra tinkamas spręsti vaizdų generavimo sistemose kylančiai neišnaudojamų resursų problemai.

7. Išvados

1. Šiuo metu esančios grafikos generavimo sistemos jau naudoja debesų kompiuterijos paslaugas. Tačiau šiose sistemose nėra galimybės atsižvelgiant į vartotojų srautus prognozuoti resursų poreikį ateityje. Todėl yra poreikis tirti ir realizuoti grafikos generavimo sistemą su galimybe prognozuoti ir rezervuoti resursus numatomiems uždaviniams.
2. Literatūros šaltiniuose apžvelgus taikomus apkrovimo balansavimo tarp viešo ir privataus debesies metodus paaiškėjo, kad balansavimo tarp privataus ir viešo debesies naudojimo metodai priklauso nuo uždavinių tipų ir jų pobūdžio. Todėl nėra vieno bendro, visur tinkančio metodo ir kuriamai kompiuterinės grafikos generavimo sistemai, kuri veikia hibridiniame debesyje reikia savito balansavimo metodo.
3. Atlikus realizuotos sistemos panaudojimo atvejų ir apkrovimo balansavimo algoritmo testavimą pašalintos pastebėtos klaidos, todėl sukurta sistema pilnai atitinka iškeltus reikalavimus ir veikia korektiškai.
4. Atliktas debesų kompiuterijos ir paskirstyto skaičiavimo įtakos Blender projektų generavimo laikui eksperimentas. Gauti rezultatai parodė, kad fiziniame serveryje veikiančios dvi virtualios mašinos gali pagerinti generavimo rezultatus iki 35% lyginant su to paties fizinio serverio generavimo rezultatais. Todėl tikslinga kuriant grafikos generavimo sistemą privačius resursus apjungti į debesį ir taip virtualizuoti resursus.
5. Atlikus darbe siūlomo apkrovimo balansavimo ir ateinančių uždavinių srauto prognozavimo eksperimento efektyvumo tyrimą rezultatai parodė, kad toks modelis gali sumažini viešo debesies naudojimą su eksperimentiniais duomenimis iki 56%. Todėl siūlomas algoritmas efektyviai pagerina nuosavų resursų išnaudojimą ir yra tinkamas spręsti vaizdų generavimo sistemose kylančiai neišnaudojamų resursų problemai.

8. Literatūra

1. Gooding S. L., Arns L., Smith P., Tillotson J., Implementation of a Distributed Rendering Environment for the TeraGrid, IEEE 1-4244-0420, 2006;
2. Jing H., Gong B., The Design and Implementation of Rendering Farm Manager Based on OpenPBS, IEEE 978-1-4244-3291-2/08, 2008;
3. Sotomayor B., Montero R. S., Liorente I. M., Foster I., Virtual Infrastructure Management in Private and Hybrid Clouds, IEEE 1089-7801/08, 2009;
4. Bossche R. V., Vanmechelen K., Broeckhove J., Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads, IEEE 978-0-7695-4130-3/10, 2010;
5. Mattess M., Vecchiola C., Buyya R., Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market, 978-0-7695-4214-0/10 IEEE 2010;
6. Morley D., Parker C. S., Understanding Computers Today and Tomorrow, 12th Edition, 1-4239-2521-1, 2009;
7. Zhurong Z., Wei D., Yuhui Q., Linrui L., A Grid Based Graphics Rendering Design, IEEE 978-0-7695-3600-2/09, 2009;
8. Sotomayor B., Montero R. S., Llorente I. M., Resource Leasing and the Art of Suspending Virtual Machines, IEEE 978-0-7695-3738-2/09, 2009;
9. Keahey K., Tsugawa M., Matsunaga A., Fortes J. A. B., Sky Computing, IEEE 1089-7801/09, 2009;
10. Armbrust M., Fox A,, Griffith R., Joseph A. D., Above the Clouds: A Berkeley View of Cloud Computing, UC Berkeley Reliable Adaptive Distributed System Laboratory, 2009;
11. Turner M., Budgenm D., Brereton P., Turning Software into a Service, IEEE 0018-9162/03, 2003;
12. BevyRender, BevyRender sistemos internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą <http://bevyrender.com/>;
13. BlenderCloud, BlenderCloud sistemos internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą < http://www.blendercloud.net/>;
14. CloudFuzion, Axceleon bendrovės internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą <http://www.axceleon.com/products.html>;
15. Felix, Felix sistemos internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą <http://www.felixrender.com/>;
16. RenderCore, RenderCore sistemos internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą <http://www.rendercore.com/>;
17. RenderRocket, RenderRocket sistemos internetinis puslapis [žiūrėta 2011-03-13]. Prieiga per internetą <http://www.renderrocket.com/>

18. Bittencourt L. F., Senna C. R., Madeira E. R. M., Scheduling Service Workflows for Cost Optimization in Hybrid Clouds, IEEE 978-1-4244-8909-1, 2010;
19. Zhang H., Jiang G., Yoshihira K., Chen H., Saxena A., Intelligent Workload Factoring for A Hybrid Cloud Computing Model, IEEE 978-0-7695-3708-5/09, 2009;
20. Chieu T. C., Mohindra A., Karve A. A., Segal A., Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environmen, IEEE 978-0-7695-3842-6, 2009;
21. Ganapathi A., Chen Y., Fox A., Katz R., Patterson D., Statistics-Driven Workload Modeling for the Cloud, IEEE 978-1-4244-6523-1, 2010;
22. Dornemann T., Juhnke E., Freisleben B., On-Demand Resource Provisioning for BPEL Workflows Using Amazon‘s Elastic Compute Cloud, IEEE 978-0-7695-3622-4/09, 2009;
23. Tan T., Kiddle C., An Assessment of Eucalyptus Version 1.4, Grid Research Center, University of Calgary, 2009;
24. Baun C., Kunze M., Building a Private Cloud with Eucalyptus, IEEE 978-1-4244-5945-2/09, 2009;
25. Zhong H., Tao K., Zhang X., An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems, IEEE 978-0-7695-4106-8/10, 2010.
26. Nurmi D., Wolski R., Grzegorocyk C., Obertelli G., Soman S., Youseff L., Zagorodnov D., The Eucalyptus Open-source Cloud-computing System, IEEE 978-0-7695-3622-4/09, 2009;
27. Eucalyptus, Eucalyptus projekto internetinis puslapis [žiūrėta 2010-10-22]. Prieiga per internetą <http://open.eucalyptus.com/>;
28. Open Nebula, Open Nebula projekto internetinis puslapis [žiūrėta 2010-10-22]. Prieiga per internetą <http://www.opennebula.org/>;
29. Nimbus, Nimbus projekto internetinis puslapis [žiūrėta 2010-10-22]. Prieiga per internetą <http://workspace.globus.org/>;
30. AjayKumar S., Nachiappan C., Periyakaruppan K., Boominathan P., Enhancing Portable Environment using Cloud and Grid, IEEE 978-0-7695-3654-5, 2009;

9. Priedai

WORKLOAD MANAGEMENT MODEL FOR SERVICES PROVIDED IN CLOUD SYSTEM

Gytis Vilutis[1], Linas Daugirdas[1]
Kristina Sutiene[2], Martynas Vaidelys[2]

[1]*Kaunas University of Technology, Department of Computer Networks, Studentu 50, Kaunas, Lithuania, gytis.vilutis@ktu.lt,linas.daugirdas@stud.ktu.lt*
[2]*Kaunas University of Technology, Department of Mathematical Research in Systems, Studentu 50, Kaunas, Lithuania, kristina.sutiene@ktu.lt, martynas.vaidelys@stud.ktu.lt*

Abstract. Full usage of resources is one of the most relevant problems in maintenance of Cloud systems. The paper mainly focuses on evaluating the adequate number of servers in private Cloud system. The proposed model provides the cost efficient distribution of incoming jobs. The workload balancer is presented which is complemented by job scheduling strategies. The validity of proposed solution is shown in experiment results.

Keywords: Cloud computing, hybrid Cloud, resource scheduling, workload management, load balancing.

Introduction

Infrastructure as a service (IaaS) is one of the main Cloud computing services. This term defines the ability to deliver hardware components to customer as a service. These components can be: CPU, RAM, hard drive disk and computer network resources. In most cases these resources are virtualized so customer has plenty of combinations to get as many computation resources as he wants. Moreover this service can be started on demand, and it also keeps "pay as you go" billing model. In practice, there are three types of such Cloud computing services [5], [18], [24]: private Cloud, public Cloud, and hybrid Cloud.

This work is emphasized on hybrid Cloud, which is a combination of VM running in private and public Clouds is chosen to fulfil customer needs, i.e. the resources can be expanded if needed. However, the problem arises from development of a system which automatically expands and shrinks in the public Cloud depending on the number of users. In this case, it is difficult to determine how many resources should be deployed in private Cloud and how many of them should be rented in public Cloud to make sure that all users will receive its service and the downtime of private resources will be minimized. Cloud load balancers are developed and used to solve such problems [9], [21].

Cloud load balancer is an algorithmic model which must guarantee an optimal allocation of load distribution between the private and public Clouds. Such model minimizes expenses spent on renting VM from Clouds and determines cost optimal quantity of private resources. These models are directly dependent on the number of users, resources, tasks types, complexity, and processing duration in the system. Therefore, there is no single solution, and each case requires a distinctive solution to be found.

This paper explores the theoretical operating principles of system which provides different complexity task computation service to customers. System uses virtualized resources that optimize system work when various complexity tasks don't fully load CPU. Dependent on type of tasks, system reschedules them and rents additional resources to do computation on time. In our system model, graphics rendering service was chosen.

The remainder of this paper is organized as follows. Section 2 provides the related works in the field of Cloud computing. Section 3 presents the architecture of hybrid Cloud services. Section 4 describes the workload management model, its components, modelling assumptions, and algorithm. Numerical results are described in Section 5. Conclusions are presented in Section 6.

Related work

This section surveys a number of recent contributions in the field of Cloud computing environment. Such computing model offers the benefits including [6], [15], [19]: (a) improved peak-load handling and dynamic resource provisioning without the need for setting up a new software or hardware infrastructure in every location; (b) superior ability of resource providers to meet Service Level Agreements (SLA) compliance for clients; (c) improved service by optimizing the service location and throughput according to users' QoS needs; (d) higher reliability as providers can transparently migrate their services to other domains, thus enabling a highly resilient computing environment. However, providers of such environments must solve the challenge of configuring their system to provide maximal performance with minimum cost of resources used while fulfilling the applications' quality of service constraints.

To respond to the dynamic workload behaviour, the strategies such as job scheduling and workload balancing have been intensively studied in the literature, as well as practically applied. There are many computer resources on the Cloud, but it is very important to employ these resources in reasonable way. In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds [8], [15]. Currently, the best known Cloud service provider

Amazon EC2 offers Elastic Load Balancer [1], which automatically distributes the incoming application across EC2 instances and provides the amount of load balancing capacity needed in response to incoming traffic.

One of the main roles for workload balancer is to manage the peak load by leasing Cloud infrastructure services from market. In the references [4], [9], different strategies for extending the capacity of local clusters with commercial providers are evaluated. These strategies aim to schedule reservations for resource requests. [17] paper presents the model for predicting various runtime overheads involved in using virtual machines, allowing to efficiently support advance reservations. Both physical and simulated experimental results are presented. Prediction is also considered in [16], applying it to both a standard priority scheduler and a pre-emptive job scheduler. The paper [23] presents a segregation of base workload and trespassing workload, the two naturally different components composing the application workload. The authors propose a technology of the intelligent workload factoring service, which enables factoring incoming requests not only on volume but also on data content. [7], [14] present an initial design of a workload generator that can be used to evaluate alternative configurations without the overhead of reproducing a real workload, as well as pay attention to workload modelling.

The other group of researches formulates the problem as optimization model. [21] examines the optimization model in a multi-provider hybrid Cloud setting with deadline-constrained and workloads that are characterized by memory, CPU and data transmission requirements. This aims to decide which workloads to outsource to what Cloud provider, while fulfilling the applications' quality of service constraints. [24] proposes an optimized scheduling algorithm to achieve the optimization or sub-optimization for Cloud scheduling problems. The authors used an improved Genetic Algorithm for the automated scheduling policy. In the paper [22], a configurable model that aims to minimize a many-task computing computation's turnaround time cost with as few resources as possible is proposed. Based on this model, a self-optimizing task partitioning algorithm has been devised for scheduling tasks. The optimization helps when the automatization of processes is considered. Since the Cloud provider's infrastructure tends to be very large and complex with many parameter settings, it is difficult for system administrators to manually change these parameters at run time in order to meet customer's SLAs if the variability of the workload is intensive. The solution is to use autonomic computing techniques [10], such as control theory, machine learning, and combinatorial search methods combined with queuing network models. The Cloud computing service provider, as an autonomic controller, can include global utility function as an objective function to be maximized. This utility function may take into account the negotiated Service Level Agreements for each customer as well as their relative priority.

The payment strategies and revenue management models are also considered by researchers in the field of Cloud computing. [3] proposes a probabilistic model for the optimization of monetary costs, performance and reliability given user requirement SLA and dynamic conditions. The authors used real instance price traces and workload models. The paper [13] describes how prices should be set to achieve predefined revenue goals from Cloud computing. Two approaches that support the Cloud provider in its decision to accept or deny requests are analyzed. It is shown that predefined policies allow increasing revenue compared to widely technical models used such as first-come-first-serve. The similar problem is addressed in the paper [2], where a customized version of the concept of self-adjusting bid prices in the area of Cloud computing is introduced. It allows to determine the minimum price a consumer has to pay for requesting a service. An interesting work is published in the paper [11]. It provides a deep analysis of financial perspective of Cloud computing, taking into consideration a large number of factors from IT industry and modelling its corresponding Return on Investment.

System architecture

Since Cloud resources are virtualized there is a possibility that it slows down the private resources. In order to decide if it is worth to link resources to a Private Cloud, a simple experiment that shows an influence of virtualization was made. Blender software was chosen to render graphics, and Eucalyptus is used to make a Cloud infrastructure based on our resources. Three different complexity graphical projects were picked and rendered in three scenarios:

- Project was rendered on one physical server. This duration was defined as an etalon for further comparison;
- Project was rendered on VM running on one physical server;
- Project was distributed and rendered on two VM running on one physical server.

Rendering durations were expressed as a proportion to compare virtualization influence (Fig. 1). Results above etalon are less efficient and results below etalon are more efficient than etalon. This experiment showed equivocal results. In second scenario the virtualization makes rendering slower. It happened because the virtualization needs computation resources itself. But model using 2 VM per server is capable to make the rendering faster. In case of rendering non-complex graphics CPU is not fully load and such distribution can drastically improve utilization and rendering speed. In this case a combination of virtualized and distributed rendering gave up to 35% speed-up and in the situation of full CPU load results were only 6% slower in a comparison with physical server.

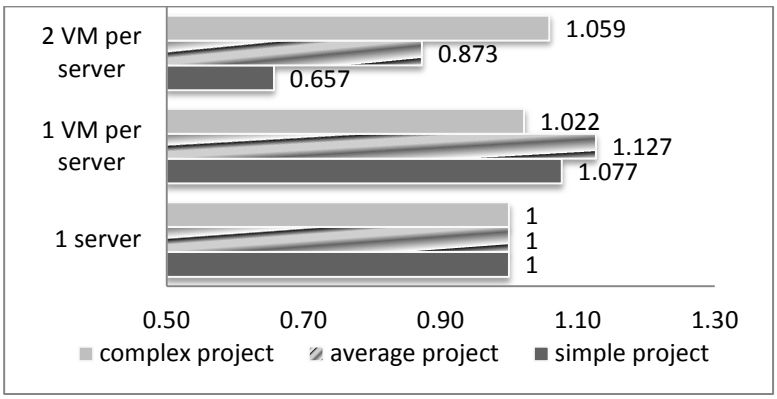


Figure 1. Virtualization influence to rendering speed

According to virtualization experiment a Hybrid Cloud type was chosen to make a system architecture model in which Eucalyptus software was preferred to deploy Private Cloud (Fig. 2). All nodes are running 2 VM and are divided into clusters. Workload management server schedules tasks and calculates how many and when additional resources from Public Cloud are needed. Web, File and MySQL servers realize an interface for submitting rendering tasks and getting rendered results.

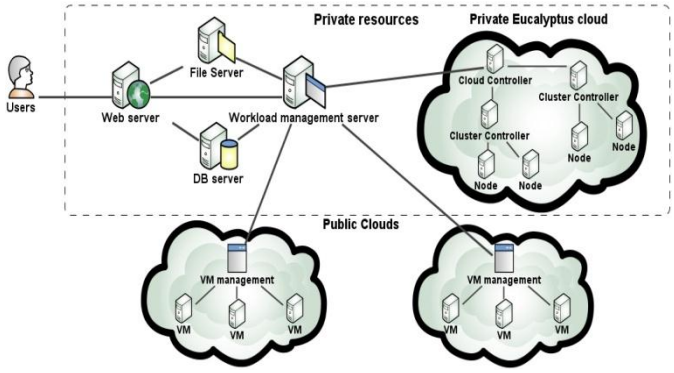


Figure 2. System architecture

This system model allows expanding resources by renting them from the Cloud but too frequent and too long usage of Cloud would be not profitable. Optimal number of private and rented servers needs to be found to ensure successful and cost efficient work of such system.

Workload management model

The purpose of modelling is to evaluate the fixed number of virtual machines (VM) in the Private Cloud that would be adequate to the incoming job flow.

Modelling assumptions

This section describes main assumption of model and its components to be developed.

The transfer of jobs to the Private Cloud or to the Public Cloud takes the same time duration, thus the data transfer time is not evaluated in the modelling. The parameters of virtual machines in the Private Cloud, as well as in the Public Cloud are identical. The modelling is based on optimization function, where penalty coefficients are used: the penalty for VM's downtime in the Private Cloud is four times less than the penalty for the usage of VM in the Public Cloud for the same period.

The model developed in this paper consists of three sub-models: generation of workload, preprocessing of jobs, and simulation of jobs processing. The first sub-model generates the arrival of jobs based on statistical data. These data are automatically captured by real Cloud monitoring system. The workload observations compose a time series, where the amount of upcoming jobs is fixed at each time moment. The analysis of given time series showed that the periodical fluctuations of workload are related to a period of one day or one week. Figure 3 demonstrates the periodical effect of one week.

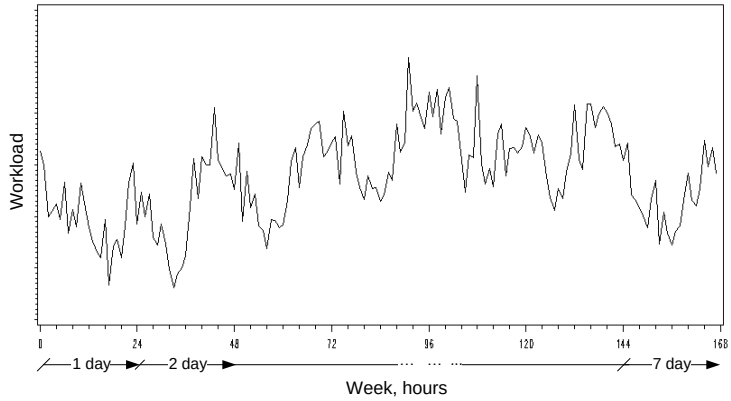


Figure 3. Weekly workload

The assumptions for sub-model of workload generation:

- Two job flows are generated with equal intensity but different priorities for job processing: low-priority flow has the postponement of two hours, high-priority flow has no postponement, i.e. jobs are processed at once they arrive to the system;
- The intensity of job flows is related with statistical time series of workload data;
- The frequency of jobs arrival into system is every hour ;
- The jobs are generated with assigned resources needed to process the task (CPU work time), which is distributed by Normal distribution taking values from 1 to 7 hours of work time for four virtual machines;
- It is assumed that each virtual machine employs the same time duration for processing the assigned part of task.

The assumptions for sub-model of jobs preprocessing:

- Each job is decomposed into four equal parts and then is sent to the simulation component;
- The decomposition of jobs allows to evaluate the resources (in hours) needed for computing. The assumptions for simulation of jobs processing;
- The decomposed task is sent to the Private Cloud if there exist vacant virtual machines. The data transfer is performed based on priority of jobs;
- High-priority job is sent to the Public Cloud if there are no vacant virtual machines in Private Cloud at the current time moment;
- Low-priority job is sent to the Public Cloud if there are no vacant virtual machines in the Private Cloud at the current time moment, and it is predicted that there will be no vacant virtual machines next two hours taking from job's arrival moment;
- Low-priority job is sent to the Public Cloud if there are no vacant virtual machines in Private Cloud at the current time moment, and the job will not be started when the postponement of two hours will terminate;
- Vacant virtual machines in the Private Cloud are predicted based on the evaluation of free virtual machines during next two hours, as well as how many new low-priority jobs will arrive during next two hours.

Workload balancing and jobs scheduling algorithm

Jobs scheduling is done based on available postponement, then the workload balancing is enabled. The model is built based on algorithm schema that is depicted in Figure 4. The algorithm takes into account the assumptions made in previous section.

The explanations for steps in algorithm are given below.

Incoming jobs. Jobs arrive into system based on assumptions given for workload generation (Section 4.1.). Every job has necessary attributes: incoming time, priority, length, virtual machines needed to accomplish the job, and time moment job must be started. The checking of newly arrived jobs is executed every one hour. The attribute *job's length* denotes the time duration needed to process a job in the Cloud, since it is assumed that all virtual machines assigned for the job will work the same amount of time. This allows to determine the number of required virtual machines. Because of assumption that each job is decomposed into four

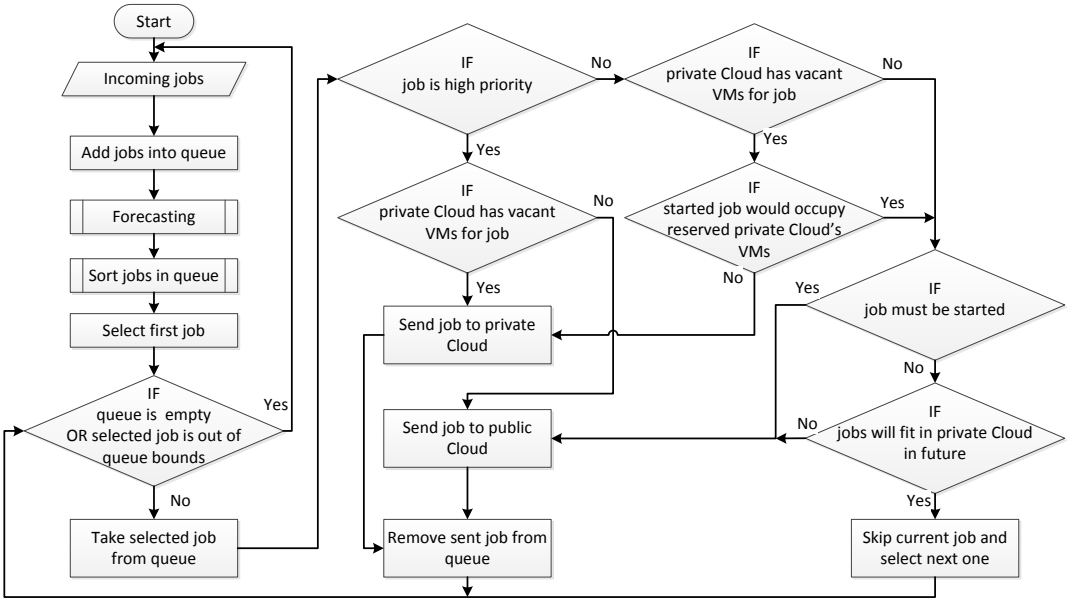


Figure 4. Workload balancing and jobs scheduling algorithm

equal parts, the number of virtual machines is multiplied by four. The attribute *time job must be started* is changing every time moment, based on formula *time job must be started = priority – (time now – job's incoming time*). It describes time left until the necessary job's execution, and is recalculated for jobs waiting in queue, as well as for newly arrived jobs. This is mostly used in sorting the job's queue.

Queue. Every incoming stream of jobs is added to the queue. The job is removed from queue if it is sent to the Cloud. Queue is always sorted in specific order. The goal is to primarily accomplish high priority jobs, then jobs that cannot wait any more (*time job must be started=0*), and finally to accomplish as more as possible long jobs in the private Cloud. Because of jobs skipping, there is a parameter that indicates the selected member in queue. It can happen that job with postponement is left in the queue and waits for the next checking.

Forecasting the number of high priority jobs. It is assumed that high priority jobs should be firstly sent to private Cloud. Thus, there is a need to reserve virtual machines for high priority jobs probable in the future. The forecasts are done for the possible number of jobs, the length of jobs, and the number of virtual machines needed for incoming high priority jobs in next eight hours. Time duration of eight hours follows from the assumed maximum job's length and maximum postponement time. In the model, forecasting refers to incoming job flow tendencies in the past and assumed probability for high priority job to arrive.

Filling the Cloud. The allocation of high priority jobs is as follows: they have the reserved virtual machines in private Cloud. So in most cases high priority jobs will easily be executed in the private Cloud. In case of peak of high priority jobs, they could be sent to public Cloud. The jobs with attribute *time job must be started=0* are sent to private Cloud if there exist vacant virtual machines. Obviously, if private Cloud is full and vacant private virtual machines are reserved, the job is sent to public Cloud. Finally, the jobs with postponement are considered. There is a need to check if it is profitable to postpone these jobs. In other words, if job after its postponement time is sent to public Cloud anyway, it must be sent right now without waiting. Because of sorting, jobs are in the front of queue whose *time job must be started* is minimal and length is maximum. The model takes a job from queue and tries to find out if it fits into private Cloud in the future based on its postponement time, length, number of needed virtual machines, and reservation in private Cloud. If not, job is sent to the public Cloud now. If yes, job is skipped; but during the next job checking, firstly all skipped jobs must find places in private Cloud.

Cost calculation function

The proposed algorithm (Section 4.2) should be tested on statistical data. But there is one unknown value: the fixed number of virtual machines in the Private Cloud that would be adequate to the incoming job flow. For this purpose, the theoretical model was investigated. The model is based on assumption that the demand of resources varies as continuous periodical function. In Figure 5, one period of this function is displayed.

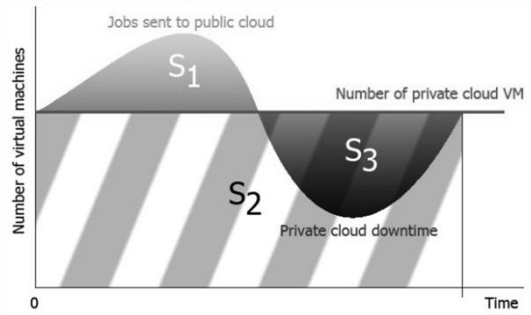


Figure 5. Visualization of workload dynamics

In the figure, S_1 denotes the surplus of workload, S_2 – the resource capacity of system to be design, and S_3 – the downtime because of low demand of resources. From this point of view, there is a possibility to evaluate own resources (number of virtual machines needed). To solve this problem, the costs calculation function was formulated:

$$C = c_{public} \cdot S_1 + c_{private} \cdot S_2 \ ; \tag{1}$$

where C – calculation's costs per T hours; T – modelled time in hours; $c_{public}, c_{private}$ – penalty coefficients for one hour of using VM in public Cloud and for one hour of using VM in private Cloud; S_1 – the number of busy VMs in public Cloud, S_2 – the number of available VMs in private Cloud.

It would be perfect to know the real costs for usage of one virtual machine. But under given assumptions it would be pointless, since real costs depend on much more factors than it is modelled in this paper. It is assumed that costs of using private Cloud are less than using public Cloud. Thus, the penalties are introduced in the function for costs evaluation. Penalty method [25] is used to tax over virtual machines used in public Cloud, and for VM's downtimes in private Cloud. Penalties proportion is given in assumptions.

To continue with modelling, Equation (1) is replaced as follows:

$$C(N) = c_{public} \cdot \int\limits_{\substack{t=0 \\ S(t)>N}}^{T} (S(t) - N) + c_{private} \cdot N \cdot T; \tag{2}$$

where N – fixed number of virtual machines in private Cloud; $C(N)$ – costs for N resources per T hours; $S(t)$ – workload in the cloud system.

As the model to be developed is discrete, the optimization model is formulated in this way:

$$C(N) = c_{public} \cdot \sum_{\substack{t=0 \\ S(t)>N}}^{T} (S(t) - N) + c_{private} \cdot N \cdot T. \tag{3}$$

Equation (3) is named as costs calculation function, which is minimized to determine the fixed number of virtual machines in the private Cloud that would be adequate to the incoming job flow. Scatter search global optimization method [20] was used to minimize costs function. Since the model is discrete, the linear interpolation [12] was applied before optimization.

Experimental results

Experiments are performed with the model presented in this paper.

The goal of experiment. To demonstrate the advantages of method presented in the paper if it is assumed that certain part of incoming jobs can be postponed for a short time. The experiment focuses on the evaluating the adequate number of virtual machines.

Firstly, the costs dependency on the VM's number in private Cloud is considered.

In Figure 6, three curves are depicted: the top one shows the costs distribution under captured statistical data; two lower curves – costs distribution if scheduling algorithm is applied for tasks with postponement time 2h and 4h. The costs are reduced because of scheduling, which cuts most 'spikes' in public Cloud and fills downtimes of virtual machines in private Cloud with postponed jobs These results are obtained under assumption that the penalty for VM's downtime in the Private Cloud is four time less than the penalty for the usage of VM in the Public Cloud for the same period. The proposed algorithm makes sense in reducing costs if the number of virtual machines

is close to adequate or less. It was assumed that two job flows of high and low priorities are generated. Under law of equal probabilities for both flows, the curve of costs dependency from the number of virtual machines would be depicted between "0 h postpone" and „2 h postpone" in Figure 6. The distribution of those flows is given in Figure 7.

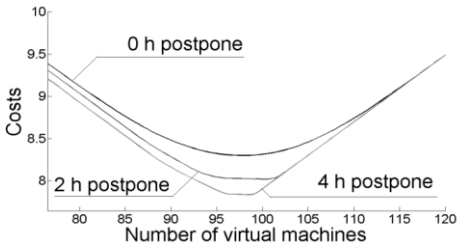


Figure 6. Costs dependency from VM's number and from postponement of tasks

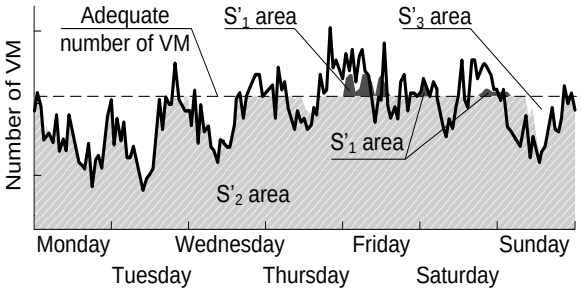


Figure 7. Workload distribution between private and public Cloud

In Figure 7, the simulated workload distribution is depicted. Black curve represents the statistical demand of virtual machines at any time, S'_2 area denotes the load in private Cloud after jobs scheduling and workload balancing algorithm was used, S'_1 area represents the load in public Cloud after jobs scheduling and workload balancing algorithm was applied. Horizontal dashed line shows the adequate number of virtual machines.

The rescheduling effect is depicted as the part of S'_2 area above black curve in Figure 7. The workload is increased in the private Cloud if rescheduling strategy (Figure 4) is applied. When the intensity of jobs flow is high for a certain time, some jobs are sent to the public Cloud (S'_1 area) based on the assumptions.

This experiment demonstrates how results are improved if the proposed algorithm of scheduling and balancing (Figure 4) is applied: (a) the utilization of public Cloud's service is reduced from 35% to 17%; (b) the full balancing of the system to be design in this paper is increased from 0% to 28%. The full balancing means the full load of VMs in private Cloud, while there is no need to use public Cloud's service.

The main experiment. The constant assumptions: two flows of tasks with different priorities; the penalty for VM's downtime in the private Cloud is four times less than the penalty for the usage of VM in the public Cloud for the same period. The variable assumptions: the improvement in costs is evaluated while the number of virtual machines and ratio of incoming jobs in the flow are variable.

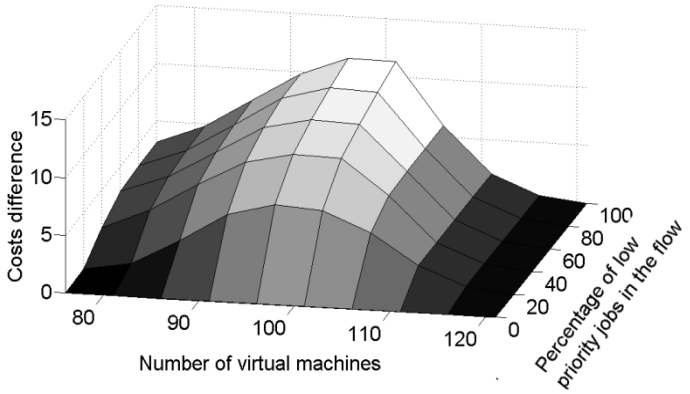


Figure 8. Costs difference dependency from VM's number and from high/low priorities

Figure 8 shows that the costs are minimized at the maximum point of surface. The form of surface is determined by three factors: the size of penalties, the high/low priority ratio in the incoming flow, the number of virtual machines in the private Cloud. The modelled system's flexibility is obtained if low priority jobs dominate in the incoming flow. VM's number deviation to the right from adequate number (surplus) determines less cost than the deviation to the left (deficit).

Conclusions

In general, the main purpose of various overviewed researches is to propose novel rescheduling techniques, workload balancing strategies, dynamic provision strategies for reliable job completion and cost efficient computing in the hybrid Cloud environment.

In this paper, the workload scheduling and balancing algorithm is proposed. Its implementation practically is worth if the certain part of tasks in the flow can be postponed for a short time period. The proposed algorithm

makes sense in reducing total costs if the number of virtual machines is close to adequate or less. The utilization of public Cloud may be significantly reduced and the workload in the private Cloud is more evenly distributed if workload management is applied, while the number of virtual machines in the private Cloud is the same.

The following improvement of the proposed model will be continued during the implementation in the real environment using open-source IaaS hybrid cloud system. It is predicted that the private Cloud will be loaded more evenly if the presented algorithm is applied.

References

[1] **Amazon Load Balancer Service.** [interactive 2010.12.07]. access via the Internet: http://aws.amazon.com/elasticloadbalancing/ .

[2] **Anandasivam A., Buschek S., Buyya R.** A Heuristic Approach for Capacity Control in Clouds. *Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria.* 2009.

[3] **Andrzejak A., Kondo D., Yi S.** Decision Model for Cloud Computing under SLA Constraints. *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 257-266.* 2010.

[4] **Assuncao M.D., Costanzo A., Buyya R.** Evaluating the cost-benefit of using Cloud computing to extend the capacity of clusters. *HPDC '09: Proceedings of the 18th ACM international symposium on High performance distributed computing, 141–150, USA.* 2008.

[5] **Baun C., Kunze M.** Building a private Cloud with Eucalyptus. *5th IEEE International Conference, E-Science Workshop, 33-38*. 2009.

[6] **Buyya R., Ranjan R.** Special section: Federated resource management in grid and Cloud computing systems. *Future Generation Computer Systems, Volume 26, Issue 8, 1189-1191*. 2010.

[7] **Ganapathi A., Chen Y., Fox A., Katz R., Patterson D.** Statistics-driven workload modeling for the Cloud. *IEEE 26th International Conference on Data Engineering Workshops, 87-92.* 2010.

[8] **Kumar Chandra P., Sahoo B.** Dynamic load distribution algorithm performance in heterogeneous distributed system for I/O- intensive task, *TENCON IEEE Region 10 Conference, 1-5.* 2008.

[9] **Mattess M., Vecchiola C., Buyya R.** Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market. *High Performance Computing and Communications (HPCC), 12th IEEE International Conference, 180-188*. 2010.

[10] **Menascé D.A., Ngo P.** Understanding Cloud Computing: Experimentation and Capacity Planning. *Proceedings of Computer Measurement Group Conference, Dallas, TX.* 2009.

[11] **Misra S.C., Mondal A.** Identification of a company's suitability for the adoption of Cloud computing and modelling its corresponding Return on Investment. *Mathematical and Computer Modelling, In Press, Corrected Proof.* 2010.

[12] **Phillips G.M.** Interpolation and Approximation by Polynomials. *Springer, 328p.* 2003.

[13] **Püschel T. J., Anandasivam A., Buschek S., Neumann D.** Making money with Clouds: revenue optimization through automated policy decisions. *17th European Conference on Information Systems (Newell S, Whitley EA, Pouloudi N, Wareham J, Mathiassen L eds.), 2303-2314, Verona, Italy*. 2009.

[14] **Quiroz A., Kim H., Parashar M., Gnanasambandam N., Sharma N.** Towards autonomic workload provisioning for enterprise Grids and Clouds. *10th IEEE/ACM International Conference on Grid Computing, 50-57.* 2009.

[15] **Ranjan R., Zhao L., Wu X., Liu A**. Peer-to-Peer Cloud Provisioning: Service Discovery and Load-Balancing. *Cloud Computing: Computer Communications and Networks, Volume 0, Part 2, 195-217.* 2010.

[16] **Sodan A.** Predictive Space- and Time-Resource Allocation for Parallel Job Scheduling in Clusters, Grids, Clouds. *Parallel Processing Workshops, 39th International Conference, 313-322.* 2010.

[17] **Sotomayor B.**, **Montero R.S., Llorente I.M., Foster I.** Resource Leasing and the Art of Suspending Virtual Machines. *High Performance Computing and Communications, 11th IEEE International Conference, 59-68.* 2009.

[18] **Sotomayor B.**, **Montero R.S., Llorente I.M., Foster I.** Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing, September/October, 14-22*. 2009.

[19] **SunGard Availability Services**. Top business and operational advantages Cloud computing can deliver to IT organizations. 2010 [interactive 2010.11.12]. access via the Internet: http://www.dssresources.com/news/3120.php .

[20] **Ugray Z., Lasdon L., Plummer K., Glover F., Kelly J., Martí R.** Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. *INFORMS Journal on Computing, Volume 19, Issue 3, 328–340.* 2007.

[21] **Van den Bossche R., Vanmechelen K., Broeckhove J.** Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads. *Cloud Computing, IEEE 3rd International Conference, 180-188*. 2010.

[22] **Yu H., Li Y., Wu X., Xiao ., Li X.** A Self-Optimizing Computation Partitioning Algorithm for Distributed Many-Task Computing. *Fifth Annual ChinaGrid Conference (ChinaGrid), 16-24.* 2010.

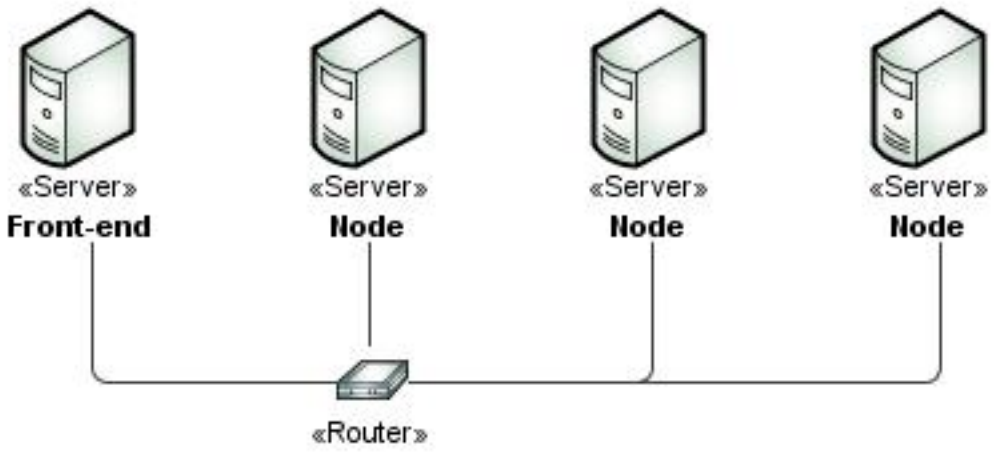
[23] **Zhang H., Jiang G., Yoshihira K., Chen H., Saxena A.** Intelligent Workload Factoring for a Hybrid Cloud Computing Model. *World Conference on Services-I, 701-708.* 2009.

[24] **Zhong H., Tao K., Zhang X.** An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems. *The Fifth Annual ChinaGrid Conference, 124-129*. 2010.

[25] **Žilinskas A.** Matematinis programavimas. *VDU leidykla, 228p.* 2000.

Eucalyptus sistemos instaliacijos vartotojo vadovas buvo parengtas pagal naudojimosi šia sistema praktiką ir informacija pateikta www.eucalyptus.com tinklalapyje.

Eucalyptus 1.5.2 versijos instaliacija į standartinės konfigūracijos Linux CentOS 5.4 serverius. Eucalyptus susideda iš trijų pagrindinių komponentų – eucalyptus-cloud, eucalyptus-cc ir eucalyptus-nc, kurie gali būti suinstaliuoti atskiruose serveriuose, tačiau šioje sistemoje cloud ir cc dalys bus realizuotos viename serveryje ir sudarys Front-end dalį, kaip pavaizduota 1 paveiksle.



1 pav. Eucalyptus realizacija.

Veiksmai prieš instaliaciją

Pirmiausiai reikia sinchronizuoti Front-end, node ir client kompiuterių laiką. Tai atliekama vykdant :

```
yum install -y ntp
ntpdate pool.ntp.org
```

Node kompiuteris turi turėti suinstaliuota Xen hypervisor.

```
yum install -y xen
sed --in-place 's/#(xend-http-server no)/(xend-http-server yes)/' /etc/xen/xend-config.sxp
sed --in-place 's/#(xend-address localhost)/(xend-address localhost)/' /etc/xen/xend-config.sxp
/etc/init.d/xend restart
```

Front-end daliai reikalinga Java, Apache ant ir DHCP serveris.

```
yum install -y java-1.6.0-openjdk ant ant-nodeps dhcp bridge-utils
```

Sėkmingai Eucalyptus komponentu tarpusavio komunikacijai reikalingi atviri prievadai:

- 8443, 8773, 8774 Front-end dalyje,
- 8775 node kompiuteryje.
- Komunikacijai iš išorinių tinklų reikalingas priėjimas prie Front-end dalies išorinio adreso 8773 prievadu.

Šiai instaliacijai išjungsime firewall ir SElinux paleidę konsolėje komandą:

```
system-config-securitylevel
```

Front-end(cloud ir cc) dalies instaliacija

Išskleidžiame parsisiųstą eucalyptus-1.5.2-centos-i386.tar.gz archyvą:

```
tar zxvf eucalyptus-1.5.2-centos-i386.tar.gz
cd eucalyptus-1.5.2-centos-i386
```

Instaliuojame trečių šalių programinę įrangą esančią parsiustame pakete:

```
rpm –Uvh aoetools-21-1.el4.i386.rpm \
euca-axis2c-1.5.0-2.i386.rpm \
euca-rampartc-1.2.0-1.i386.rpm \
vblade-14-1mdv2008.1.i586.rpm
```

Toliau instaliuojame Eucalyptus cloud ir cc reikalingus paketus:

```
rpm –Uvh eucalyptus-1.5.2-1.i386.rpm \
eucalyptus-cloud-1.5.2-1.i386.rpm \
eucalyptus-gl-1.5.2-1.i386.rpm \
eucalyptus-cc-1.5.2-1.i386.rpm
```

Startuojame CC ir cloud servisus:

```
/etc/init.d/eucalyptus-cloud start
/etc/init.d/eucalyptus-cc start
```

Node kompiuterio instaliacija

Išskleidžiame parsisiųstą eucalyptus-1.5.2-centos-i386.tar.gz archyvą:

```
tar zxvf eucalyptus-1.5.2-centos-i386.tar.gz
cd eucalyptus-1.5.2-centos-i386
```

Instaliuojame trečių šalių programinę įrangą esančią parsiustame pakete:

```
rpm –Uvh aoetools-21-1.el4.i386.rpm \
euca-axis2c-1.5.0-2.i386.rpm \
euca-rampartc-1.2.0-1.i386.rpm \
vblade-14-1mdv2008.1.i586.rpm \
euca-libvirt-1.5-1.i386.rpm
```

Toliau instaliuojame Eucalyptus nc reikalingus paketus:

```
rpm –Uvh eucalyptus-1.5.2-1.i386.rpm \
eucalyptus-gl-1.5.2-1.i386.rpm \
eucalyptus-nc-1.5.2-1.i386.rpm
```

Taip pat reikia nustatyti CentOS (2.6.18-164.9.1.el5xen) branduolį pagrindiniu:

```
nano  /boot/grub/menu.lst
defoult=1 → default=0
```

Patikriname ar eucalyptus vartotojas gali sąveikauti su Xen hypervizorium per libvirt.

```
su eucalyptus -c "virsh list"
```

Rezultate turėtų būtį grąžinamas domeno vardas, kitu atveju sąveikavimas nepavyksta. Jeigu viskas gerai, toliau startuojame eucalyptus-nc serverį:

```
/etc/init.d/eucalyptus-nc start
```

Eucalyptus konfigūracija

Startavus cc ir cloud front-end dalyje pirmiausiai reikia nurodyti EUCALYPTUS kintamajam sistemos namų direktoriją bei patikriname ar CC (Cloud Controler) vykdo Java procesą:

```
export EUCALYPTUS=/opt/eucalyptus/
ps aux | grep euca
```

Grąžintuose duomenyse turėtų matytis Java procesas.

Toliau Front-end dalyje reikia nurodyti cluster ip adresą ir vardą. Tai galima atlikti iš komandinės eilutės:

```
$EUCALYPTUS/usr/sbin/euca_conf –addcluster <clusterio vardas> <clusterio IP adresas>
```

Taip pat reikia registruoti node‘s kompiuterių ip adresus ir sinchronizuoti raktus, tai atliekama:

```
$EUCALYPTUS/usr/sbin/euca_conf –addnode “<node_1 IP adresas> … <node_N IP adresas>”
```

Likusi konfigūracijos dalis atliekama per internetinę sąsają adresu https://localhost:8443 , kur pirmą kartą prisijungus vartotojo vardas ir slaptažodis yra „admin“. Pastarąjį, per pirmą prisijungimą reikia pasikeisti bei nurodyti elektroninio pašto ir Walrus duomenų saugyklos išorinį ip adresą.

Virtualių mašinų tinklo konfigūracija

Eucalyptus sistema leidžia rinktis vieną iš trijų pagrindinių virtualių mašinų komunikavimo tinkle būdų:

1. „SYSTEM“ – pagal nutylėjimą sistema dirba šiuo rėžimu – kiekviena naujai sukuriama virtuali mašina gauna tinklo nustatymus iš tinkle esančio DHCP serverio.

2. „STATIC“ rėžimas leidžia Front-end dalyje startuojant DHCP serverį nurodyti naujai kuriamų virtualių mašinų statinius IP ir MAC adresus tinkle.

3. „MANAGED“ rėžimas skirtas virtualioms mašinoms suteikti dinaminius vidinius ir išorinius adresus bei leidžia valdyti VM ugniasienes.

Tinklo konfigūracijos keitimas iš SYSTEM į STATIC rėžimą. Front-end dalies Eucalyptus konfigūraciniame faile turime nustatyti tinklo valdymo rėžimą, interfeisą, nurodyti kelią iki DHCP direktorijos, SUBNET, NETMASK, BROADCAST, ROUTER, DNS serverius ir IP bei MAC adresus skiriamus VM. Tam geriausia panaudoti informaciją gaunamą įvykdžius **„ifconfig“** komandą Front-end dalyje, o patys nustatymai apsirašo taip:

```
nano /opt/eucalyptus/etc/eucalyptus/eucalyptus.conf


VNET_MODE="STATIC"
VNET_INTERFACE="eth0(tinkle interfeisas)"
VNET_DHCPDAEMON="/usr/sbin/dhcpd (kelias iki dhcpd)"
VNET_SUBNET="192.168.1.0"
VNET_NETMASK="255.255.255.0"
VNET_BROADCAST="192.168.1.255"
VNET_ROUTER="192.168.1.1"
VNET_DNS="192.168.1.2"
```

Nurodome būsimų virtualių mašinų fiksuotus adresus:

```
VNET_MACMAP="AA:DD:11:CE:FF:ED=192.168.1.3 MAC_adresas=ip_adresas"
```

Node dalyje užtenka pakeisti šiuos nustatymus:

```
VNET_MODE="STATIC"
VNET_BRIDGE="br0 arba xenbr0"
```

Kliento aplinka

Kliento aplinkai naudosime Amazon EC2 įrankius, kurie yra suderinami ir su Eucalyptus cloud sistema. Parsisiunčiame ec2-ami-tools-1.3-26357 ir ec2-api-tools-1.3-30349 bei juos išarchyvuojame:

```
unzip ec2-api-tools-1.3-30349.zip
unzip ec2-ami-tools-1.3-26357.zip
```

Šių įrankių veikimui reikalinga Java ir Ruby:

```
chmod +x jdk-6u17-linux-i586-rpm.bin
./jdk-6u17-linux-i586-rpm.bin
yum install ruby
```

Prieš kreipiantis į Eucalyptus reikia parsisiųsti vartotojo sertifikatą, bei jį išsiarchyvuoti. Sertifikatą galima parsisiųsti per web sąsają prisijungus savo vartotojo vardu (šiuo atveju vartotojas „admin“):

```
mkdir $HOME/.euca
unzip euca2-admin-x509.zip -d $HOME/.euca
```

Belieka išeksportuoti Java ir EC2 direktorijas bei eucarc failą padaryti „source“. Šiuos veiksmus reikia atlikti kaskart perkrovus sistemą:

```
export JAVA_HOME=/usr/java/jdk1.6.0_17
export EC2_HOME=/opt/ec2-api-tools-1.3-30349
export EC2_AMITOOL_HOME=/opt/ec2-ami-tools-1.3-26357
export PATH=$PATH:/opt/apache-ant-1.7.1/bin:$EC2_HOME/bin:$EC2_AMITOOL_HOME/bin
source /root/.euca/eucarc
```

Veikimą galima patikrinti vykdant komandą:

```
ec2-describe-availability-zones verbose
```

Bet koks rezultatas apie galimą virtualių mašinų skaičių indikuoja teisingą EC2 įrankių veikimą, tačiau esant nuliniam virtualių mašinų skaičiui galimos problemos su node kompiuterio registracija arba raktų sinchronizavimu.

Virtualios mašinos kūrimas

Norint registruoti ir patalpinti naują instance reikalingas Xen aplinkoje veikiantis linux branduolys ir linux operacinės sistemos atvaizdas. Virtualios mašinos kūrimui naudosime Ubuntu 9.4 x86 architektūrai skirtą operacinės sistemos atvaizdą ir Xen pritaikytą 2.6.18.8 linux branduolį:

```
ec2-bundle-image --image /opt/neoficialus/kerneliai/2-6-18-8/boot/vmlinuz-2.6.18.8-xenU --kernel
true
ec2-upload-bundle --bucket kernel-neof-2-6-18-8 --manifest /tmp/vmlinuz-2.6.18.8-
xenU.manifest.xml
ec2-register kernel-neof-2-6-18-8/vmlinuz-2.6.18.8-xenU.manifest.xml

ec2-bundle-image --image /opt/neoficialus/os/ubu-9-04/ubuntu.9-04.x86.img
ec2-upload-bundle --bucket image-neof-ubuntu-9-04 --manifest /tmp/ubuntu.9-
04.x86.img.manifest.xml
ec2-register image-neof-ubuntu-9-04/ubuntu.9-04.x86.img.manifest.xml
```

Prieš paleidžiant virtualią mašiną reikia sukurti raktų porą:

```
ec2-add-keypair admin_key > /root/admin_key.private
chmod 0600 /root/admin_key.private
```

Virtualios mašinos paleidimas ir jų peržiūra:

```
ec2-run-instances emi-vardas -k admin key
ec2-describe-instances
```

Sėkmingą veikimą identifikuoja priskirtas ip adresas ir statusas „running“. Esant ip adresui 0.0.0.0, o sistemos statusui „running“ galima klaida tinklo konfigūracijoje, o matant statusą „terminated“ problemos reikėtų ieškoti nc.log faile. Sėkmingai paleidę virtualią mašiną prie jos galime prisijungti:

```
ssh –i kelias_iki_rakto root@ip_adresas
```

Naujo linux sistemos atvaizdo kūrimas iš veikiančio instance

Norint sukurti naują sistemos atvaizdą modifikuojant veikiančią virtualią mašiną, joje reikia įdiegti kliento aplinką (žr. „Kliento aplinka“). Kadangi paleista virtuali mašina savaime neprijungia jai skirtos ephemeral atminties, tai padarome įvesdami:

```
mount /dev/sda2 /mnt
```

Taip pat naujo sistemos atvaizdo kūrimui reikalingi rsync ir curl paketai, juos įdiegiame komandomis:

```
yum install rsync
yum install curl
```

Atlikę norimus pakeitimus sistemoje, kuriame naują sistemos atvaizdą panaudodami ec2-bundle-vol komandą. Jos galimi nustatymai:

Nustatymas	**Apibrėžimas**	**Reikalingas**	**Pavyzdys**
-k	Nurodo kelią iki vartotojo privataus rakto	Taip	-k $HOME/euca2-admin-196608e-pk.pem
-u	EC2 vartotojo ID	Taip	-u 123456789
--ec2cert	Nurodo kelią iki eucalyptus cert rakto	Taip	--ec2cert $HOME/cloud-cert.pem
-c	Nurodo kelią iki vartotojo cert rakto	Taip	-c $HOME/euca2-admin-196608e-cert.pem
-s	Nurodo atvaizdo failo dydį MB	Taip	-s 2048
-p	Nurodo norimą atvaizdo pavadinimą	Ne	-p pavadinimas
-d	Nurodo vietą, kur bus įrašytas sukurto atvaizdo failas	Ne	-d /norimas_katalogas
-e	Neįtraukiamų į kuriamą atvaizdą direktorijų sąrašas	Ne	-e /tmp,/root/slapti_dokumentai
--no-inherit	Naudojamas, kai eucalyptus sistema veikia STATIC ir SYSTEM rėžimuose.	Ne	--no-inherit

Raktai yra išarchyvuotame sertifikate, tačiau šiuo atveju, kadangi prieš tai išeksportavome eucarc failą (žr. „Kliento aplinka“), užteks nurodyti kintamuosius, saugančius kelią iki raktų. Vykdome komandą:

```
ec2-bundle-vol -d /mnt -e /mnt,/root/.ssh -s 2048 -u 000100729354 -c $EC2_CERT -k
$EC2_PRIVATE_KEY --ec2cert $EUCALYPTUS_CERT  --no-inherit
```

Toliau siunčiame į Walrus saugyklą ir registruojame naują atvaizdą analogiškai kaip kuriant virtualią mašiną iš standartinių sistemos atvaizdų (žr. „Virtualios mašinos kūrimas“).

```
ec2-upload-bundle –bucket image-mod-centos5 –manifest /mnt/image.manifest.xml
ec2-register image-mod-centos5/mnt/image.manifest.xml
```

Sistemos atvaizdo parsisiuntimas iš eucalyptus sistemos

Norint parsisiųsti sistemos atvaizdą iš eucalyptus sistemos reikia turėti įdiegtą vartotojo aplinką (žr. „Vartotojo aplinka“). Atvaizdo parsisiuntimui naudosime ec2-download-bundle komandą. Jos galimi nustatymai:

Nustatymas	**Apibrėžimas**	**Reikalingas**	**Pavyzdys**
-b	Norimo parsisiųsti image bucket vardas Eucalyptus sistemoje	Taip	-b image-neof-centos-5-3
-m	Manifesto failo vardas	Taip	-m centos.5-3.x86.img.manifest.xml
-a	Vartotojo EC2 access raktas	Taip	-a xxxxxxxxxxxx
-s	Vartotojo EC2 secret key	Taip	-s xxxxxxxxxxxx
-k	Vartotojo privatus raktas	Taip	-k $HOME/euca2-admin-196608e-pk.pem
-d	Nurodo direktoriją į kurią bus parsiųstas atvaizdas	Ne	-d /download/pirmas/
--url	Nurodomas walrus duomenų saugyklos adresas	Taip	--url http://adresas:8773/services/Walrus

Raktai yra išarchyvuotame sertifikate, tačiau šiuo atveju, kadangi prieš tai išeksportavome eucarc failą (žr. „Kliento aplinka“), užteks nurodyti kintamuosius, saugančius kelią iki raktų ir Warlus duomenų saugyklos. Vykdome komandą:

```
ec2-download-bundle –b image-neof-centos-5-3 –m centos.5-3.x86.img.manifest.xml –a
$EC2_ACCESS_KEY –s $EC2_SECRET_KEY –k $EC2_PRIVATE_KEY –d /download/pirmas/ --url $S3_URL
```

Į direktoriją /download/pirmas (direktorija turi būti sukurta prieš vykdant komandą) parsiųstos atvaizdo dalys ir manifestas, kurie paruošti siuntimui ir registravimui Eucalyptus sistemose.