ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Nerijus Barauskas

Informatikos magistro specialybės II kurso dieninio skyriaus studentas

Debesų kompiuterijos paslaugos pritaikymas internetinei matematinio programavimo ir modeliavimo sistemai

The Application of Cloud Computing service for Online Mathematical Programming and Simulation System

Magistro darbas

Darbo vadovas:

Doc. dr. V. Giedrimas

Recenzentė:

Doc. dr. S. Turskienė

Šiauliai, 2013

Turinys

Įvadas

Šiuolaikinę kompiuteriją bandoma transformuoti į modelį, kuris susideda iš paslaugų. Jos yra apdorojamos ir perduodamos vartotojams, panašiai kaip kitos komunalinės paslaugos: vanduo, elektra, dujos ar telefoninis ryšys. Tokiame modelyje vartotojai prieina prie paslaugų, priklausomai nuo jų keliamų reikalavimų, neatsižvelgiant į tai kur jos patalpintos ir kaip jos pristatomos galutiniam vartotojui. Į kelias kompiuterijos paradigmas – spietinę kompiuteriją, grid kompiuteriją ir naująją debesų kompiuteriją – buvo bandoma pritaikyti tiekti skaičiavimo paslaugas. Kiek vėliau terminas debesų kompiuterija įgijo ir kitokią prasmę – debesies infrastruktūra leidžia vartotojams, prieiti prie programinės įrangos paketų iš bet kurios pasaulio vietos. Tai reiškia, kad kompiuterijos pasaulis keičiasi ir didesnė programinės įrangos dalis bus kuriama kaip paslauga, o ne kaip pavienė programinė įranga individualiam kompiuteriui.[15]

Kaip dar 2008 m. prognozavo [1] šuolis nuo lokaliai įdiegiamos programinės įrangos į debesų kompiuteriją igyja didelį pagreitį. Tačiau lokaliai diegiamų programų kūrimas neturėtų visiškai nutrūkti, bet didesnis dėmesys bus skirtas debesų kompiuterijai. Dalis kompiuterijos veiklos bus perkelta iš individualių kompiuterių į paslaugas ir tai turės įtakos visai kompiuterių ekosistemai nuo galutinio vartotojo iki programinės įrangos kūrėjų, informacinių technologijų vadybininkų ir netgi techninės įrangos kūrėjų.[1]

Darbo tikslas – parengti ir realizuoti debesų kompiuterijos paslaugų integravimo į internetinę matematinio programavimo ir modeliavimo sistemą modelį.

Tam, kad įgyvendinti šį tikslą suformuluoti šie uždaviniai:

1. Išanalizuoti siūlomas debesų kompiuterijos paslaugas, platformas ir įrankius skirtus šioms paslaugoms kurti.
2. Analizės pagrindu parinkti tinkamą debesų kompiuterijos platformą, bei tinkamus įrankius.
3. Šiuo metu internetinėje matematinio programavimo ir modeliavimo sistemoje naudojamą užduoties kompiliavimo ir vykdymo modulį transformuoti į debesų kompiuterijos paslaugą.
4. Atlikti sistemos testavimą ir empiriniu būdu įvertinti jos funkcionalumą.

1. Teorinė dalis

1.1.Debesų kompiuterija

1.1.1. Paslaugomis orientuota architektūra

Paslaugomis orientuota architektūra (Service Oriented Architecture – SOA) nėra nauja koncepcija, nors pastaraisiais metais iš naujo susilaukė dėmesio. Pavyzdžiui, pirmosios tinklinėmis paslaugomis orientuotos architektūros buvo nuotolinis procedūrų kvietimas (remote procedure call – RPC), išskirstytų komponentinių objektų modelis (Distributed Component Object model – DCOM) ir objektų atsakymų brokeriai (Object Request Brokers – ORBs), kurie remiasi CORBA specifikacija. Naujesnis tokios architektūros pavyzdys būtų - grid tinklo architektūros ir sprendimai. [9]

SOA aplinkoje galutinis vartotojas kreipiasi į norimą funkcionalumą, kokybę palaikančią paslaugą arba į integruotą paslaugų paketą, ir priima jį realiu laiku arba tada, kai to reikalauja paslaugos specifikacija. Paslaugų atradimas, tarpininkavimas ir patikimumas yra labai svarbu ir paprastai jie kuriami taip, kad būtų sudėtiniai (sudaryti iš kitų paslaugų). Tikimąsi, kad per pastaruosius dešimt metų paslaugomis pagrįsti sprendimai bus pagrindiniai informacijos perdavimo būdai, bei atliks kitas pagalbines informacinių technologijų užduotis tiek individualiame tiek organizaciniame lygmenyje. [9]

1.1.2. Debesis

„Debesis - tai paralelinės ir paskirstytos sistemos tipas, susidedantis iš tarpusavyje sujungtų ir virtualizuotų kompiuterių, kurie dinamiškai atideda ir pateikia vieną ar daugiau vieningų resursų, paremtų paslaugų lygių susitarimu (Service Level Agreement – SLA) sudarytų derybų metu tarp vartotojo ir paslaugos tiekėjo.“ [15]

Debesų kompiuterija (Cloud Computing) – tai sekantis informacinių technologijų žingsnis, paslaugų kaip galutinių produktų srityje. Didžioji dalis debesų kompiuterijos turėtų būti paremta virtualizuotų resursų pagrindu. Debesų kompiuterijos pirmtakai atsirado maždaug 2008 metais, tačiau pats terminas tapo naudojamas nuo tų pačių metų spalio 23 dienos, kai *IBM* ir *Google* paskelbė bendradarbiavimo pradžią šioje srityje. Šio bendradarbiavimo priežastis buvo sutelkti bendras pajėgas į *IBM* „Blue Cloud“ (Mėlyno debesies) projekto vystymą. [9]

Raktas į sėkmingas informacines technologijas, tai galimybė įgyvendinti vertingą ir ekonomiškai naudingą kibernetinę infrastruktūrą. Debesų kompiuterija apima kibernetines infrastruktūras, remiasi dešimtmečio tyrimais virtualizacijos srityje, paskirstyto skaičiavimo, grid tinklo technologijomis. Taip pat apima ir naująsias tinklines, internetines ir programines paslaugas. Tai

reiškia, jog paslaugomis orientuotose architektūrose sumažinami informacinių technologijų valdymo resursų kaštai, sumažinama bendra nuosavybės teisės kaina galutiniam vartotojui. Taip pat gaunamas didesnis lankstumas bei padidinamas užsakomųjų paslaugų skaičius. [9]

Taigi, naujausia paradigma yra debesų kompiuterija, kuri žada patikimas paslaugas teikiamas per naujos kartos duomenų centrus, kurie patalpinti virtualiose skaičiavimo ir duomenų saugyklų technologijose. Vartotojai gali pasiekti programinę įrangą arba duomenis, esančius debesyje iš bet kurios pasaulio vietos pagal pareikalavimą. Vartotojai yra užtikrinti, kad debesies infrastruktūra yra robustiška ir bus prieinama bet kuriuo metu. Paslaugos turi būti patikimos, derinamos, be to turi palaikyti laisvą prieigą. Jos turi būti dinamiškai aptinkamos ir apjungiamos. Praktiškai, vartotojai nurodo reikiamą paslaugos lygį tiekėjui, pasinaudoję paslaugos kokybės (Quality of Service – QoS) parametrais, kurie yra aprašyti paslaugų lygių susitarime. Iš visų naujausių paradigmų, debesų kompiuterija pasirodo daugiausiai žadanti ir paremta kitų paradigmų pagrindu. [15]

Giliau paanalizavus, galima pastebėti jog debesų kompiuterija yra grid ir spietinės kompiuterijos derinys. Debesys yra užtikrinti naujos kartos duomenų centrai su virtualizuotais mazgais, kurie dinamiškai atideda asmeninius resursų paketus pagal užklausas. Taip užtikrinami specifiniai paslaugų lygio susitarimai, kurie nustatomi derybų metu ir priėjimą prie sudėtinių paslaugų per internetinius protokolus, tokius kaip SOAP ir REST. [15]

1.1.3. Debesies charakteristikos

Charakteristikos	Sistemos		
	Tinklinis spiečius (cluster)	Grid	Debesis
Populiacija	Prekiniai kompiuteriai	Aukštos klasės kompiuteriai (Serveriai, tinkliniai spiečiai)	Prekiniai kompiuteriai ir aukštos klasės kompiuteriai, ir tinklinės duomenų saugyklos
Dydis/Mastelis	Šimtai	Tūkstančiai	Nuo šimtų iki tūkstančių
Operacinės sistemos mazgai	Viena iš standartinių operacinių sistemų (Linux, Windows)	Bet kuri operacinė sistema (dominuoja Linux)	Virtuali mašina, kurioje vienu metu veikia kelios operacinės sistemos
Nuosavybės teisė	Pavienė	Sudėtinė	Pavienė
Sujungimas į tinklą/greitis	Aukštos kokybės su trumpu atsako laiku ir didele sparta	Dažniausiai internetas su trumpu atsako laiku ir didele sparta	Aukštos kokybės su trumpu atsako laiku ir didele sparta
Sauga/privatumas	Tradicinis prisijungimas. Vidutinis privatumo lygis, priklauso nuo	Vieša/privačia raktų pora paremtas autentifikavimas. Ribotas privatumo palaikymas	Kiekvienas vartotojas/programa yra priklausoma nuo virtualios mašinos. Aukštos kokybės saugumas ir privatumas

	vartotojo privilegijų		užtikrintas
Aptinkamumas	Draugiški servisai	Centralizuotas indeksavimas ir decentralizuota servisų informacija	Draugiški servisai
Servisų derybos	Ribotos	Paremtos SLA	Paremtos SLA
Vartotojų administravimas	Centralizuotas	Decentralizuotas ir taip pat paremtas virtualia organizacija	Centralizuotas arba gali būti perduotas trečiai šaliai
Resursų administravimas	Centralizuotas	Paskirstytas	Centralizuotas/paskirstytas
Paskirstymas/planavimas	Centralizuotas	Decentralizuotas	Centralizuotas ir decentralizuotas
Standartai/sąveikos	Paremtas virtualios sąsajos architektūra	Keletas atvirų Grid forumo standartų	Paremtas internetinių servisų (SOAP ir REST) pagrindu
Vienos sistemos atvaizdas	Yra	Nėra	Yra, bet pasirinktinis
Talpa	Stabili ir užtikrinta	Svyruoja, bet didelė	Atidedama nuo paklausos
Klaidų administravimas	Ribotas (dažnai neįvykdo užduočių, programos pasileidžia iš naujo)	Ribotas (dažnai neįvykdo užduočių, programos pasileidžia iš naujo)	Stiprus automatikos palaikymas. Virtualios mašinos lengvai perkeliamos iš vieno mazgo į kitą
Panaudojimas	Mokslui, verslui, duomenų centrams	Mokslo bendradarbiavimo ir didelio našumo skaičiavimo programoms	Internetiniams servisams bei duomenų perdavimui
Potencialas kuriant trečios šalies ar pridėtinės vertės sprendimus	Ribotas dėl standžios architektūros	Ribotas dėl stiprios orientacijos į mokslinę kompiuteriją	Didelis potencialas – gali kurti naujus servisus dinamiškai atidedant duomenų talpyklas, programinius servisus ir pasiūlo vartotojui naujus izoliuotus ar sudėtinius debesų servisus

Lentelė 1. Debesų kompiuterijos charakteristikos. [15]

Rinkinys charakteristikų, padedančių atskirti spietinę, grid ir debesų kompiuterijos sistemas, pateiktas pirmoje lentelėje (žr. Lentelė 1). Resursai spiečiuose yra patalpinti pavieniame administruojamame domene ir administruojami vieno subjekto. Grid sistemose, resursai yra išsidėstę geografiškai, išskirstyti daugelyje administruojamų domenų, turinčių savo valdymo teises ir tikslus. Dar vienas pagrindinis skirtumas tarp spietinių ir Grid sistemų, kyla iš programų atlikimo planavimo. Planavimo metu, spietinėje sistemoje, daug dėmesio skiriama bendros sistemos efektyvumui ir naudingumui, nes jis atsako už visos sistemos darbą. Iš kitos pusės planavimas grid

sistemose vadinamas resursų brokeriu, daug dėmesio skiriama specifinių programų efektyvumui pagerinti. Tokiu būdu galutiniai vartotojai susiduria su paslaugos kokybės reikalvaimais. [15]

Debesų kompiuterijos platformos apima abi, spietinę ir grid, sistemas su savo specialiais atributais ir galimybėmis tokiomis kaip virtualizacijos palaikymas, dinamiškai apjungiamomis paslaugomis su internetine jų sąsaja. Bet to debesis tiekia paslaugas vartotojams nepaisant infrastruktūros kurioje jos patalpintos.

1.1.4. Paslaugos

Debesų kompiuterija yra modelis, leidžiantis patogiai prieiti per tinklą prie bendrų konfigūruojamų resursų, pavyzdžiui, paties tinklo, serverių, duomenų, programų, paslaugų, kurie gali būti greitai atidėti ir paleisti, įdedant minimalų kiekį pastangų. Šis debesies modelis padidina galimybes penkiais esminiais bruožais: paskirstytos paslaugos, plati prieiga prie tinklo, resursų sutelkimas, greitas prisitaikymas ir matuojamos paslaugos. Taip pat, šis modelis siūlo tris paslaugų modelius: IaaS (Infrastructure as a Service – debesies infrastruktūra kaip paslauga), PaaS (Platform as a Service – debesies platforma kaip paslauga) ir SaaS (Software as a Service – debesies programinė įranga kaip paslauga). Be to, debesies modelis siūlo keturis darbo modelius: privatus debesis, bendruomenės debesis, viešas debesis, hibridinis debesis. [4]

Žiūrint iš techninės perspektyvos, debesies sistemos elementai susideda iš procesinių, tinklo ir duomenų saugojimo elementų. Debesies architektūra sudaryta iš trijų abstrakčių lygių: infrastruktūros, platformos ir programinio lygmens. Infrastruktūros lygmuo yra pats žemiausias ir siūlo apdorojimo, tinklo, duomenų saugojimo ir kitus fundamentalius kompiuterinius resursus, kaip standartizuotas paslaugas prieinamas internetu. Serveriai, duomenų saugojimo sistemos, komutatoriai, maršrutizatoriai ir kitos sistemos valdančios specifinius darbo krūvius iš paketinio apdorojimo serverio. Debesų klientai gali išdėstyti ir paleisti operacines sistemas ar programas savo turimose infrastruktūrose. Vidurinis lygmuo suteikia sudėtingesnes paslaugas programų kūrimui, testavimui, išskleidimui, patalpinimui ir palaikymui. Tuo pačiu siūlo integruotą kūrimo aplinką. Šis sluoksnis suteikia aplinką ir terpę patalpinti programas naudojant programavimo kalbas ir įrankius, siūlomus kaip paslaugas. Programinis lygmuo pats aukščiausias ir kaip paslauga yra siūloma pilna programa. [4]

1.1.4.1. IaaS

Infrastruktūra kaip paslauga (Infrastructure as a Service – IaaS) tai vienas iš trejų debesų kompiuterijos priėjimo būdų, paremtas paslaugomis. Organizacijos nuomojasi kompiuterių resursus skaičiavimams ir kietojo disko atmintį, kad būtų galima prie jų prieiti tiesiogiai iš asmeninio kompiuterio vietiniame tinkle ar internetu.

IaaS buvo pradėtas naudoti 1990 metų viduryje, kaip viena iš skaičiavimo technikų. Bet ji nebuvo pradėta plačiai naudoti iki to laiko kol, virtualizacija bei greitas ir patikimas interneto ryšys suteikė galimybę tai daryti. Įmonės besinuomojančios IaaS paslaugas turi turėti IT įgūdžių, nes paslauga yra ganėtinai paprasta infrastruktūra. Klientai pasirenka programų serverius kaip savo debesies dalį ir įkelia ten savo bibliotekas, aplikacijas ir kitus duomenis. Visas konfigūracijas atlieka pats klientas. Virtualizacija leidžia IaaS tiekėjams pasiūlyti beveik beribį kiekį serverių už palyginti nedidelius techninės įrangos priežiūros kaštus. [5]

IaaS leidžia organizacijoms greičiau konstruoti naujas programų ar aplinkų versijas, nes galima išvengti užsakinėti naują techninę įrangą, išvengiamo laukimo kol ji bus pristatyta bei tinkamai sukonfigūruota. Dar vienas populiarus IaaS panaudojimo būdas – tai talpinti organizacijų tinklaraščius. Tai padeda laikyti tinklaraštį neišnaudojant vidinių resursų, kurių pagrindinis tikslas yra tarnauti verslui. Tokiu atveju IaaS tiekėjas pilnai atsako už duomenų srauto priežiūrą bei tinklaraščio prieinamumą. [5]

Kaip ir dauguma debesų kompiuterijos paslaugų, IaaS paslaugų apmokėjimas paremtas išankstinio apmokėjimo modeliu. Dažniausiai yra nustatoma kaina už serverių skaičių, atsižvelgiant į tai ar programinė platforma yra *Windows* ar *Unix*. Taip pat mokesčiai apima ir saugomų duomenų kiekį bei įeinančių ir išeinančių duomenų kiekį. Tačiau kainų modeliai tarp tiekėjų yra labai skirtingi, tai apsunkina tiekėjo pasirinkimo procesą, bet leidžia prisitaikyti lankstesnį apmokėjimo planą.

Dažniausiai sutinkamas argumentas, kodėl reikėtų pasirinkti debesų kompiuteriją vietoj vidinės infrastruktūros, yra sąlyginai maži išlaikymo kaštai bei atsikratymas IT priežiūros. Su IaaS nėra viskas taip paprasta. IT personalas vis tiek bus reikalingas, kurti bei prižiūrėti programinę įrangą bei aplikacijas, kurios bus įdiegtos išnuomotoje infrastruktūroje. [5]

1.1.4.2. PaaS

Platforma kaip paslauga (Platform as a Service – PaaS) tai debesų kompiuterijos forma, kuri laikoma potencialia pagalba programuotojams. Padeda greičiau parašyti ir ištestuoti kliento principu veikiančias internetines aplikacijas, arba produktą, kurio paklausa planuojama labai didelė. Tokių internetinių programavimo aplinkų tiekėjų yra nemažai įskaitant salesforce.com (force.com), *Microsoft* (Azure) ir paleidimui skirta *WaveMaker*.

Šios platformos yra koncentruotos į vieną programavimo kalbą ar metodologiją. Ganėtinai patogu įmonių programuotojams, kurie neieško tiesiog platformos, bet ieško tos kuri atitiktų jų prioritetus įrankiuose ar programavimo kalbose, tokiose kaip .Net, Java ar Ruby on Rails. Platforma suteikia daug didesnį efektyvumą koduojant automatizuotas užduotis, tokias kaip nustatyti ką tik

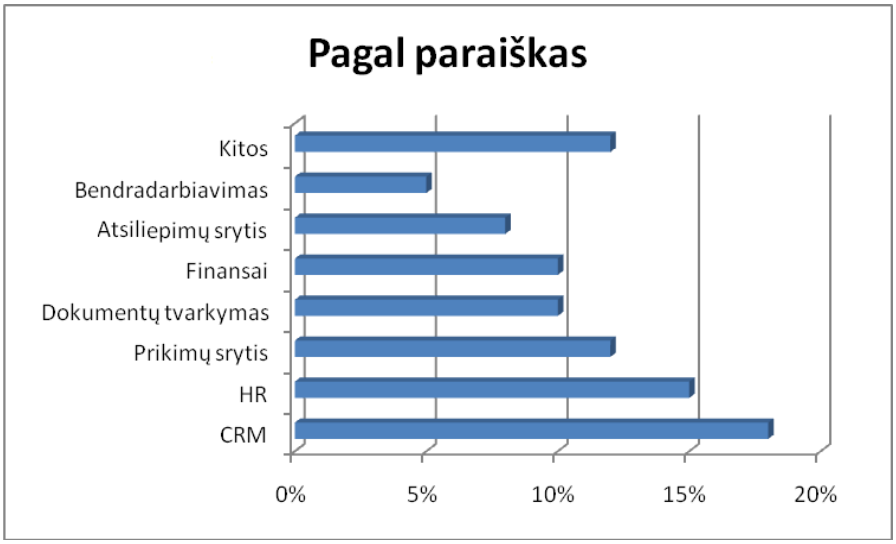
sukurtą aplikaciją kaip internetinę paslaugą. Dažniausiai siūloma debesų infrastruktūros paslauga tam, kad kūrėjas galėtų paleisti tai ką sukūrė debesies infrastruktūroje bei galėtų išsikviesti savo sukurtą paslaugą naujoje aplikacijoje.

Tokias atvejais, PaaS metodikos patrauklumas yra tas, kad aplikaciją galima kurti naudojantis tais pačiais standartais ir technologijomis, kurios bus naudojamos produkcinėje aplinkoje. Toks būdas supaprastina aplikacijos perkėlimą, pavyzdžiui, iš *Windows* kūrimo aplinkos į tikslinę produkcijos aplinką. Klaidų paieška PaaS aplinkoje yra tokia pat kaip ir tikslinės produkcijos aplinkos, tai padeda tiksliau šalinti klaidas, suteikia galimybę greitesniam aplikacijų kūrimui. Kūrėjai besinaudojantys *force.com* kuria internetines aplikacijas 4,9 karto greičiau negu tie kurie naudojasi paprasta Java ar .Net metodika, *Nucleus Research* tai patvirtino peržiūrėję septyniolika *force.com* projektų 2008 metų gegužės mėnesį. [14]

1.1.4.3. SaaS

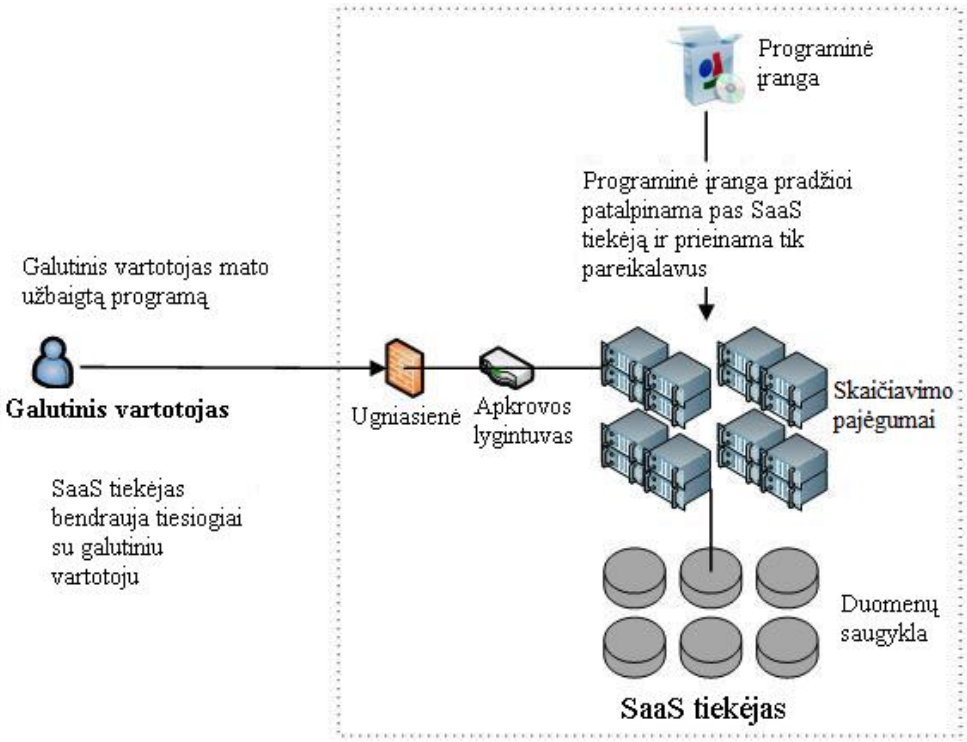
Programinė įranga kaip paslauga (Service as a Service – SaaS) yra vienas populiariausių ir produktyviausių debesų kompiuterijos tipų dėl savo lankstumo ir mastelio, didelio efektyvumo su geresniu prieinamumu, didžiulėmis paslaugomis ir mažesniais išlaikymo kaštais. *Yahoo mail*, *Google docs*, ERP, BPM ir CRM yra SaaS tipo programos. Norint pasinaudoti SaaS CRM vartotojui tereikia užregistruoti abonentą, prisijungti prie centrinės sistemos ir pasirinki jam reikalingas CRM paslaugas. Taip pat, vartotojas gali pasirinkti papildomas duomenų bazes ar programas, įterpti savo duomenis. Galinis vartotojas gali laisvai naudotis visomis tiekėjo siūlomomis paslaugomis bei duomenų talpinimu. SaaS vis labiau įgyja pripažinimą įmonėse, dėl žymiai pigesnių išlaikymo kaštų, todėl kad siūlo programinę įrangą skirtą verslo administravimui mokant mėnesinį mokestį, kuris yra kur kas mažesnis už licencijų įsigijimą. SaaS projektai dažnai kuriami pačių tiekėjų, panaudojant daugiavartotojišką aplinką, palaikančią API ir SLA standartus susietus su plačiu profesionalių paslaugų paketų. [16]

SaaS modelis siūlo aukšto lygio išskirstytos duomenų architektūros aprašymus, kurie reikalingi duomenų perdavimo programinės įrangos kūrimui, talpinimui ir naudojimui. SaaS architektūroje tiekėjo licencija pagrįsta paslaugos teikimo prenumeratos modeliu. Tai reikalauja jog klientas turėtų kompiuterį ar serverį, turintį interneto ryšį, kad galėtų atsisiųsti ir naudoti programinę įrangą, kas leidžia klientui atsisakyti įsigyti brangią techninę ar programinę įrangą. Taip pat tokia licencija gali būti skirta naudoti vienam asmeniui ar žmonių grupei. [16]



Pav. 1. Aplikacijos teikiamos naudojant SaaS modelį procentais.

Programinės įrangos kaip paslaugos architektūra tapo labai svarbi versle, atliekant tokias užduotis kaip ERP, CRM, kompiuterizuotai HR, sąskaitų tvarkymas, paslaugų valdymas, pirkimų administravimas, darbo eigos sistemų valdymas, dokumentų valdymas ir panašiai. Aplikacijas pristatomas SaaS modeliu padalinus į grupes, labiausiai išsiskiria CRM, HR ir pirkimų sritis (žr. Pav. 1). SaaS pilnai užbaigta programinė įranga nepriklauso klientui, tačiau ji gali būti prieinama prireikus. Programa egzistuojanti debesyje yra internetinė programa, todėl gali būti prieinama iš bet kurio kompiuterio naršyklės.



Pav. 2. Programinė įranga kaip paslauga. [16]

Kaip matome antrame paveikslėlyje, SaaS tiekėjas atsakingas netik už duomenų centro paslaugos, reikalingos programinės įrangos vykdymą, tiekimą, bet ir siūlo visą programinę įrangą galutiniam vartotojui. SaaS tiekėjai taip pat gali būti PaaS ir IaaS paslaugų tiekėjais, taip sukurdami virtualią infrastruktūrą [16].

1.1.4.4. XaaS

Ankščiau minėtos trys debesų kompiuterijos paslaugos yra ne vienintelės. Sparčiai augant debesų kompiuterijos panaudojimo kiekiui, paslaugų rūšių atsirado kur kas daugiau. Tačiau IaaS, PaaS ir SaaS yra pagrindas, kuriuo remiasi kitos paslaugos.

Viena iš XaaS paslaugų yra geležis kaip paslauga (Metal as a Service – MaaS) – tiltas tarp debesų kompiuterijos ir tradicinio programų diegimo. Ši paslauga pasirodė kartu su *Ubuntu 12.04 LTS* operacine sistema 2012 metų balandžio 26 dieną. MaaS leidžia sistemų administratoriams aprūpinti ir įdiegti didelius kiekius fizinių serverių. Įtraukia debesies tipo semantiką, į pajėgumų pagal pareikalavimą atidėjimą, fizinių serverių registravimo procese. Ši paslauga yra suprojektuota horizontaliai išdėstytai aplinkai, pavyzdžiui, didelių duomenų, darbo krūviui ir vidinių debesų kūrimui, bet veikia lygiai taip pat kaip ir bet kokia kita debesų kompiuterijos paslauga. [8]

MaaS prižiūri fizinių serverių grupes, kaip debesies tipo resursus, kad būtų paskirstytos pagal pareikalavimą. Didelio kiekio daugkartinio panaudojimo debesų paslaugos gali būti įdiegtos į MaaS prižiūrimą įrangą ir išskirstyti arba sutraukti pagal pareikalavimą. Mazgai gali būti statomi į eilę, kad būtų dinamiškai paskirstyti, arba galima tiesiog įrašyti *Ubuntu* ir sukonfigūruoti rankiniu būdu.

Serveriai lengvai priskiriami MaaS automatiniu ar rankiniu būdu, po to visą atsakomybę už mazgų priežiūrą, tokią kaip, programinės įrangos atnaujinimas, veiklos vertinimo ir duomenų pašalinimo iš mašinos jei atsijungiant nuo MaaS atlieka pati sistema. [8]

1.2. Platformos

Darbo tikslams pasiekti bus naudojamas PaaS modelis, todėl tikslinga apžvelgti egzistuojančias platformas.

Industriniai analitikai darė įvairias spėliones, kokią įtaką turės debesų kompiuterija visai kompiuterijos industrijai. Atsižvelgiant į Merrill Lynch atliktą tyrimą, pastebėta kad debesų kompiuterija turėtų siekti šimto šešiasdešimties milijardų JAV dolerių pelną adresuojamomis rinkos galimybėmis, devyniasdešimt penkių milijardų pelną verslo ir produktyvumo programomis ir šešiasdešimt penkis milijardus internetinėje reklamoje. Kitas tyrimas atliktas Morgan Stanley taip pat nustatė, kad debesų kompiuterija viena iš ryškesnių technologinių tendencijų. Kompiuterijos industrija eina link platformos kaip paslaugos ir programinės įrangos kaip paslaugos tiekimo vartotojams ar įmonėms, nepriklausomai nuo laiko ir vietos. Neseniai keletas akademinių ir

industrinių organizacijų pradėjo investuoti ir kurti technologijas ir infrastruktūras debesų kompiuterijai. Akademiniams bandymas priklauso *Virtual Workspaces, OpenNebula*, ir *Reservoir*. O industriniams bandymams *Amazon, Google App Engine, Microsoft Azure, Sun Network.com* ir *Aneka*. [15]

1.2.1. Amazon EC2.

Amazon Elastic Compute Cloud (elastinis skaičiavimų debesis – EC2) suteikia virtualią kompiuterinę aplinką, kuri leidžia vartotojams paleisti *Linux* operacine sistema pagrįstas programas. Vartotojas gali sukurti naują *Amazon* mašinos atvaizdą (Amazon Machine Image – AMI) su reikiamomis programomis, bibliotekomis, duomenimis ir globaliai prieinamais *Amazon* mašinų atvaizdais. Vartotojui norint įkelti sukurtą ar pasirinktą AMI į *Amazon* paprastą saugojimo paslaugą (Simple Storage Service – S3), prieš tai reikia paleisti, sustabdyti, ir sekti visas keliamas AMI rinkmenas. *Amazon EC2* kraunasi tuo metu, kai vartotojai naudojasi AMI. O, *Amazon S3* kraunasi bet kuriuo duomenų perdavimo metu, tiek išsiunčiant tiek priimant duomenis. [16]

2008 metų pabaigoje *Amazon* buvo populiariausia debesies platforma, skirta bendram naudojimui. Joje buvo registruoti keturi šimtai tūkstančių programinės įrangos kūrėjų, tam kad galėtų naudotis *Amazon* siūlomomis paslaugomis. Ši platforma tapo populiari todėl, kad galėjo pasiūlyti galingą infrastruktūrą, kuri palaiko mažmeninės prekybos operacijas internetu ir suteikia galimybę savo vartotojams naudotis duomenų saugojimo, kompiuterinių resursų, žinučių mechanizmo, turinio valdymo ir sąskaitų tvarkymo paslaugomis. [10]

1.2.2. Google App Engine

Google App Engine (*Google* programų variklis) suteikia galimybę vartotojams paleisti internetines programas, parašytas naudojant *Python* standartinę biblioteką, patį programų variklį, duomenų saugyklą, *Google* sąskaitas, URL eilutę, atvaizdų manipuliatorius ir elektroninio pašto servisus. *Google App Engine* taip pat tiekia internetu pagrystą administravimo prieigą vartotojams, kad būtų galima lengviau valdyti jo veikiančias internetines programas. Dabar *Google App Engine* leidžia nemokamai naudoti penkis šimtus megabaitų atminties ir maždaug penkis milijonus puslapio peržiūrų. [15]

1.2.3. Microsoft Azure

Microsoft Azure siekia suteikti integruotą kūrimo aplinką, talpinimą ir debesų kompiuterijos aplinkos valdymą, kad programų kūrėjai galėtų lengvai kurti, talpinti, administruoti, paskirstyti tiek internetines, tiek paprastas programas po *Microsoft* duomenų centrus. Kad įgyvendintų šiuos siekius *Microsoft Azure* palaiko visapusišką rinkinį nuosavų kūrimo įrankių ir protokolų, tokių kaip *Live Search, Microsoft, .Net services, Microsoft SQL services, Microsoft Share Point Services* ir

Microsoft Dynamic CRM Services. *Microsoft Azure* taip pat palaiko internetinius protokolus tokius kaip SOAP ir REST, kad suteiktų programinės įrangos kūrėjams sąsaja tarp *Microsoft* ir ne *Microsoft* įrankių ir technologijų. [15]

1.2.4. Sun Network.com

Sun Network.com (*Sun Grid*) leidžia vartotojams paleisti *Solaris* operacinę sistemą, palaiko *Java*, *C*, *C++* ir *Fortran* pagrindu sukurtas programas. Pirma vartotojas turi surinkti ir patikrinti savo kurtas programas ir scenarijus vietinėje kūrimo aplinkoje, kuri sukonfigūruota panašiai kaip ir *Sun Grid*. Po to vartotojui, reikia sukurti sugrupuotą *zip* archyvą, kuriame būtų visi susiję scenarijai, bibliotekos, vykdomos rinkmenos ir įvesties duomenys, ir jį įkelti į *Sun Grid*. Tada vartotojas gali vykdyti ir prižiūrėti programą naudodamas *Sun Grid* internetinį portalą arba API. Po programos užbaigimo, vartotojas turi parsisiųsti vykdymo rezultatus į lokalią kūrimo aplinką peržiūrai. [15]

1.2.5. Aneka

Aneka buvo sukomercinta kompanijos *Manjrasoft*. Tai *.Net* pagrindu, paslaugomis orientuota, resursų valdymo sistema. Ji suprojektuota taip, kad palaikytų kelis programų modelius bei privačius ir saugius sprendimus, komunikacinius protokolus. Tam, kad sukurti *Aneka* debesį, serviso tiekėjui reikia paleisti sukonfigūruotą *Aneka* konteinerį, palaikantį reikiamus servisus kiekviename parinktame kompiuteryje. Konteinerio tikslas – sukurti paslaugas, jis dirba kaip atskiras taškas bendraujantis su likusiu debesiu. Ji taip pat palaiko SLA, todėl vartotojai gali nurodyti QoS reikalavimus, pavyzdžiui, galinis terminas, biudžetas. Vartotojai gali prieiti prie *Aneka* debesies nuotoliniu būdu per *Gridbus* brokerius. *Gridbus* brokeriai taip pat leidžia vartotojams derėtis dėl QoS reikalavimų servisų tiekimui. [15]

1.2.6. OpenNebula

Atviro kodo *OpenNebula* yra virtualus infrastruktūros variklis turintis didelį funkcionalumą, kuris reikalingas norint atskirti, prižiūrėti ir valdyti virtualias mašinas atskiruose fiziniuose resursuose. *OpenNebula* architektūra yra lanksti ir modulinė, tai leidžia integruoti ją į skirtingai valdomas ir skirtingai sukonfigūruotas infrastruktūras. [2]

OpenNebula sudaryta iš trijų pagrindinių dalių. Jos branduolys yra centralizuotas komponentas, kuris valdo visą virtualios mašinos gyvavimo ciklą ir atlieka pagrindines virtualios mašinos operacijas (išskirstymą, priežiūrą, migraciją ir sunaikinimą). Branduolys taip pat atsako už pagrindinį valdymą ir stebėjimo sąsają fiziniams šeimininkams. Talpos valdymas gali būti atjungiamas. Jis išplečia *OpenNebula* branduolio funkcionalumą. Talpos valdymas susijęs su virtualios mašinos vieta, kuri priklauso nuo rinkinio iš anksto sudarytų taisyklių. Talpos planavimas

pagal nutylėjimą leidžia paprastą porų sudarymo mechanizmą ir palaiko vartotojų apribojimus. Tam, kad butų gaunama pagrindinės virtualizacijos sluoksnio abstrakcija, *OpenNebula* naudoja prijungiamas virtualizuojamas priėjimo tvarkykles kurios atskleidžia pagrindinį administratoriaus funkcionalumą (pvz. išskirstymą, priežiūra ir virtualios mašinos išjungimą). Nors *OpenNebula* nėra priskiriama nei vienai specifinei aplinkai, tiekiančiai valdymo sluoksnį neatsižvelgiant į tai, kokia virtualizacijos technologija yra panaudojama. [2]

Taigi, *OpenNebula* forma yra fizinė infrastruktūra palaikanti duotų paslaugų vykdymą. Be to, ji gali dinamiškai išplėsti savo infrastruktūrą, prisijungiant išorinio debesies sąsajos *Amazon EC2* virtualizacijos tvarkykles, kurios įtrauktos į *OpenNebula* tvarkykles ir naudojamos sukurti sąsajas su kito tipo debesimis. Ši betarpiška išorinių debesų integracija leidžia efektyvų priėjimą prie išorinių kompiuterinių resursų turint paprastus namų kompiuterius. [2]

1.2.7. OpenStack

OpenStack yra atvira ir skaidoma debesų kompiuterijos platforma, skirta konstruoti tiek privačius tiek atvirus debesis. Tai bendradarbiavimu grįstas projektas, skirtas sukurti laisvai prieinamą kodą, reikalingus standartus ir tinkamą pagrindą siekiant naudos tiek debesų paslaugos tiekėjams tiek vartotojams. *OpenStack* šiuo metu tiekia dviejų tipų paslaugas: skaičiavimo (OpenStack Compute) ir duomenų saugojimo (OpenStack Object Storage). [13]

Skaičiavimo paslauga suprojektuota taip, kad galėtų valdyti didelius virtualių mašinų tinklus ir sukurti kintamo dydžio debesų kompiuterijos platformą. Taip pat, ji suteikia programinę įrangą ir valdymo skydus reikalingus palaikyti debesies aplinką, įskaitant veikiančias mašinas, tinklo administravimą ir galimybę kontroliuoti vartotojų prieigą prie projektų. Taip pat yra galimybė naudotis *Amazon EC2* resursais.

Saugojimo paslauga leidžia saugoti kintamo dydžio objektus standartizuotų serverių spiečiuose. Atmintis gali būti plečiama tiek, kad galėtų saugoti pentabaitų dydžio failus, padarant paslaugą prieinamą virtualių mašinų saugojimui, pašto saugojimui, arba atsarginių kopijų archyvavimui už nedidelę kainą, replikuojant ir paskirstant duomenis dideliame kiekyje standžiųjų diskų. [13]

1.2.8. Platformų palyginimas

	Sistema							
Pavadinimas	*Amazon EC2*	*Google App engine*	*Microsoft Azure*	*Sun network.com (Sun Grid)*	*Aneka*	*OpenNebula*	*Grids Lab Aneka*	*OpenStack*
Dėmesys	*Infrastruktūra*	*Platforma*	*Platforma*	*Infrastruktūra*	*Platforma*	*Infrastruktūra*	*Programinės įrangos platforma įmonių debesims*	*Platforma*
Serviso tipas	*Skaičiavimams, duomenų saugojimui (Amazon S3)*	*Internetinės programos*	*Internetinės ir neinternetinės programos*	*Skaičiavimai*	*Internetinės ir neinternetinės programos*	*Internetinės ir neinternetinės programos*	*Skaičiavimams*	*Skaičiavimams, duomenų saugojimui*
Virtualizacija	*Operacinės sistemos lygmenyje ant Xen administratoriaus*	*Programų konteineris*	*Operacinės sistemos lygmenyje per gamyklinius kontrolerius*	*Darbo valdymo sistema (Sun Grid Engine)*	*Resursų valdymas ir planavimas*	*Resursų valdymas ir planavimas*	*Resursų valdymas ir planavimas*	*Resursų valdymas ir planavimas*
Dinaminės derybos dėl Qos parametrų	*Nėra*	*Nėra*	*Nėra*	*Nėra*	*Nėra*	*Nėra*	*SLA pagrysta resursų rezervacija iš Aneka pusės*	*Nėra*
Vartotojo priėjimo sąsaja	*Amazon EC2 komandinės eilutės įrankiai*	*Internetinė administravimo konsolė*	*MS Azure portalas*	*Darbų pateikimo scenarijai, Su Grid internetinis portalas.*	*Internetinė administravimo konsolė*	*OpenSuse, Debian, Ubuntu*	*Internetinis portalas*	*Ubuntu, CentOS*
Internetinės programos	*Taip*	*Taip*	*Taip*	*Taip*	*Taip*	*Taip*	*Taip*	*Taip*
Pridėtinės vertės servisų tiekimas	*Taip*	*Ne*	*Taip*	*Taip*	*Taip*	*Taip*	*Ne*	*Taip*
Programavimo karkasas	*Pritaikomi Linux pagrindu AMI*	*Python*	*Microsoft .Net*	*Solaris OS, Java, C, C++, Fortran*	*Microsoft .Net*	*Pritaikomi Linux pagrindu AMI*	*Programavimo modelis C# ir kitos .Net palaikančios programavimo kalbos*	*Pritaikomi Linux pagrindu AMI*

Lentelė 2. Debesų platformų palyginimo lentelė.

Šioje lentelėje pateiktas debesų platformų palyginimas pagal tam tikras ypatybes. Pagal lentelėje pateiktus duomenis galima pastebėti, kad kiekviena platforma siūlo vis kitokias debesų paslaugas. Internetinėms programoms labiausiai tinka Google App Engine, nes jos visos ypatybės skirtos tik joms. Tai pat internetinėms programos palaikyti tinka Windows Azure platforma, tačiau joje gali puikiai veikti ir ne internetiniai servisai. Kita vertus internetines programas gali palaikyti ir kitos platformos, tačiau jų efektyvumas būtų mažesnis. Skaičiavimui labiausiai tiktų Sun Network.com ir Aneka. Tačiau OpenStack panašus į Amazon EC2 ir OpenNebula junginį, kas leistų sukurti Amazon EC2 tipo debesį privačiam naudojimui, bei esant poreikiui panaudoti Amazon EC2 resursus.

2. Projektinė dalis

2.1. Įrankių ir priemonių pasirinkimo analizė

Norint sukurti debesų kompiuterijos paslaugą, pirmiausia reikėtų turėti prieigą prie debesies infrastruktūros. Atsižvelgiant į kuriamą paslaugą, uždaviniui įgyvendinti reikalingas platformos kaip paslaugos lygmuo. Platformos gali būti teikiamos (*Windows Azure, Amazon EC2*) arba kuriamos (*Windows Server 2012 System Center, OpenStack, OpenNebula*). Kadangi kuriama paslauga yra specifinė ir vienas šio darbo uždavinių išsiaiškinti ar debesų kompiuterija tinkama jo realizacijai, buvo nutarta naudoti kuriamą platformą. Be to kuriamoji platforma naudoja privatų debesį, o teikiamoji – viešąjį, todėl ekonominiu požiūriu, šio darbo ribose, naudingesnė kuriamoji. Taigi bus reikalingi įrankiai debesies kūrimui ir konfigūravimui.

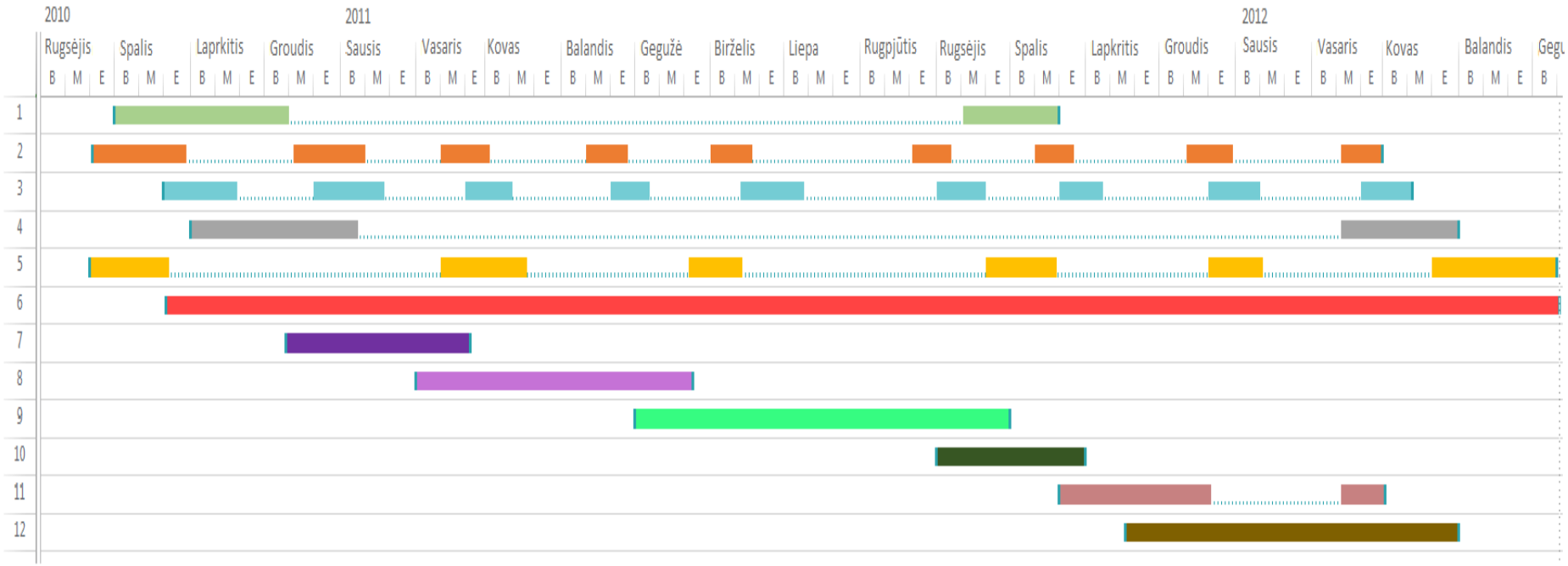
Sekantis žingsnis išsirinkti įrankius debesies kūrimui, tai gali buti *Windows Server 2012 System Center, OpenNebula, OpenStack* bei *Aneka*. OpenNebula ir OpenStack yra atviro kodo projektai, todėl yra geresnės galimybės jas naudoti eksperimentiniais tikslais, keičiant parametrus. Iš pradžių didesnis dėmesys buvo skiriamas *OpenNebula* platformai, ją galimą įdiegti naudojant *Ubuntu* operacinę sistemą. Tačiau 2012 metų balandžio mėnesį pasirodžiusi nauja *Ubuntu* operacinė sistemos versija turėjo integruotą įrankį *Essex* ir visą kitą ko reikia norint įdiegti *OpenStack* platformą. *OpenNebula* kaip ir *OpenStack* virtualizacija orientuota į resursų valdymą ir planavimą, tai reiškia jog mazgai dažniausiai būna virtualios mašinos, turinčios vienodus parametrus. Tačiau *Ubuntu 12.04LTS* turi MaaS paslaugą, kuri leidžia kur kas paprasčiau įtraukti ir fizines mašinas. Norint prijungti naują mašiną tereikia prisijungi prie pagrindinio serverio naudojant internetinį portalą, įrašyti naujojo mazgo MAC (Media Access Control – įrenginio priėjimo kontrolės) adresą, ir mazge įgalinti aktyvavimo internetu galimybę. Taigi atsižvelgiant į tai, jog galutiniam projekto realizavimui bus naudojami keli įvairių pajėgumų fiziniai serveriai ir *Ubuntu* kartu su *OpenStack*

leis išvengti serverių virtualizavimo. Taigi, tolesniam etapui bus naudojama *OpenStack* platforma. [17]

Kaip minėta ankščiau paslaugos aprašymo įrankius dažniausiai apsprendžia pasirinkta debesies platforma. *Ubuntu* šiam tikslui naudoja *JUJU Orchestration,* šis įrankis leidžia patalpinti, pašalinti, atnaujinti, publikuoti ir apjungti paslaugas esančias debesyje. Paslaugų patalpinimui naudojamas paslaugos aprašymas, kuris susideda iš dviejų segmentų – tai pats paslaugos aprašymas ir scenarijai skirti paslaugos įdiegimui, paleidimui, stabdymui. Paslaugos gali būti paprastos arba sudėtinės (tokios, kurioms teikti reikalingos dar kitos paslaugos). Paslaugų aprašymas sudaromas *yaml* failuose, nurodant paslaugos pavadinimą, aprašą, kokios papildomos paslaugos reikalingos bei ryšį tarp jų. Jeigu paslauga sudėtinė tai gali būti pridėti scenarijai, aprašantys paslaugų tarpusavio ryšių sudarymą, nutraukimą ar pašalinimą. [3] [17]

Atsižvelgiant į tai, jog debesų kompiuterijos paslauga patalpinta ne vienoje fizinėje ar virtualioje mašinoje ir kuriama dažniausiai kitoje aplinkoje, buvo reikalingas įrankis paslaugos versijavimui. *Ubuntu* sistemoje dažniausia sutinkama *Apache Subversion* (SVN) versijavimo sistema. Tačiau GIT (*Global Information Tracker*) sistema kur kas efektyviau leidžia išnaudoti projekto šakas ir taip vienu metu vystyti kelias jo versijas, pakeitimus išsaugant pagrindiniame serveryje nekeičiant pagrindinės šakos išeities kodų. Tai leidžia ateityje tobulinti paslaugas ir jas atnaujinti tik įsitikinus, kad naujoji versija veikia sklandžiai išlaikant reikiamą funkcionalumą [7]. Šios abi sistemos reikalauja atskiro serverio, kuriame būtų talpinami ir saugomi projektai, be to reikia kad debesies mazgai turėtų galimybę parsisiųsti ir įsidiegti paslaugas. Čia vėlgi, puikiai pasitarnauja GIT versijavimo sistema. Esant dideliam GIT populiarumui 2008 buvo įkurtas portalas *GitHub,* kuriame visi norintys gali susikurti paskyrą ir naudotis standartiniu paketu, kuris leidžia laikyti neribotą kiekį atvirų projektų. *GitHub* orientuotas į atvira kodą, todėl atviri projektai gali būti laisvai kopijuojami kitų vartotojų. Tai kur kas palengvina projekto vystymą ir kitų vartotojų bendradarbiavimą.

2.2. Projekto vykdymo planas



Pav. 3. Darbų vykdymo planas

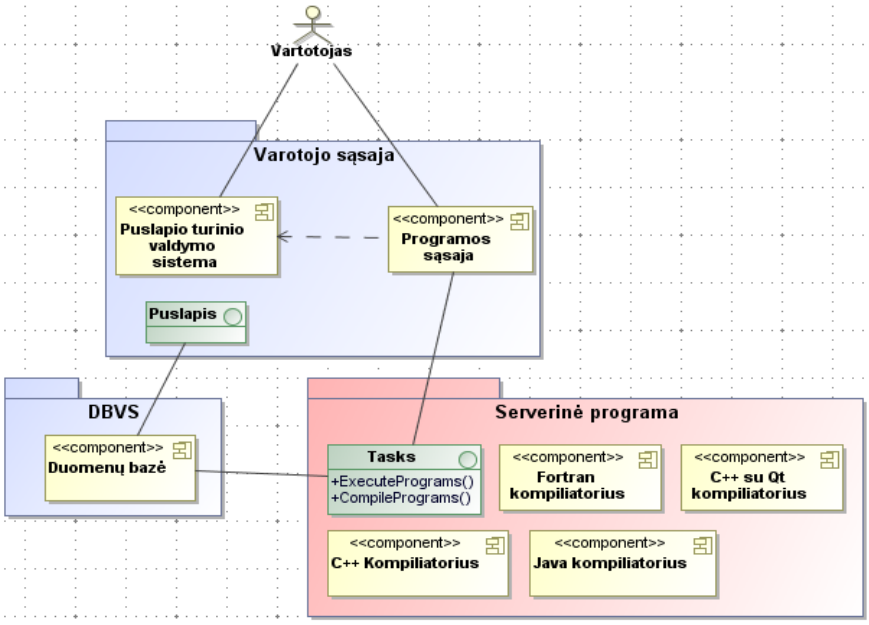
Darbų paaiškinimai:

1. Debesų kompiuterijos paslaugos realizavimui skirtų reikalavimų formulavimas.
2. Informacijos paieška.
3. Informacijos šaltinių analizė.
4. Debesų kompiuterijos paslaugos projektui reikalingų *UML* diagramų braižymas.
5. Konsultacijos su darbo vadovu.
6. Darbo aprašymo sudarymas.
7. Techninės įrangos konfigūravimas.
8. Debesies platformos diegimas.
9. Debesies paslaugos aprašymas.
10. Debesies paslaugos publikavimas.
11. Internetinės matematinio programavimo ir modeliavimo sistemos ir sukurtos paslaugos derinimas.
12. Sukurtos paslaugos testavimas.

Įvykdyti visus darbe numatytus uždavinius planuojama pagal pateiktų darbų vykdymo planą (žr. Pav. 3). Darbai 1, 2, 3 yra skirti pirmo ir antro uždavinių įgyvendinimui. Darbai 7, 8, 9, 10 ir 11 yra skirti terčio uždavinio realizavimui. Paskutinis darbas skirtas ketvirto uždavinio įgyvendinimui.

2.3. Pradinis projekto aprašymas

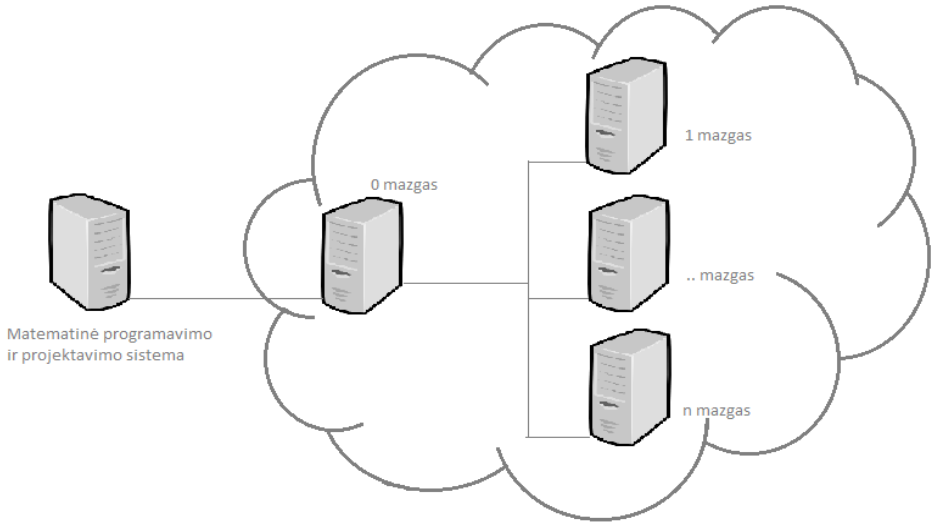
Šis darbas yra dalis Šiaulių Universiteto, Matematikos ir Informatikos fakulteto, Informatikos katedroje kuriamos ir tobulinamos internetinės matematinio programavimo ir modeliavimo sistemos. Ji buvo pristatyta autorių Remigijaus Valčiuko ir Mariaus Neimanto du tūkstančiai dvyliktų metų birželį, ginant baigiamuosius magistro darbus. Kuriamos sistemos paskirtis suteikti aplinką, kurioje akademinės visuomenės atstovai bei kiti suinteresuoti asmenys galėtų aprašyti, redaguoti ir realizuoti (suprogramuoti) matematinius modelius, algoritmus, atlikti jų paiešką ir vykdyti realizuotas programas. Dabartinėje sistemos versijoje galima programuoti pasinaudojant *C/C++*, *Java* ir *Fortran 90* programavimo kalbomis bei *QT* ir *Netlib Repository LAPACK* bibliotekomis. Sistema buvo ištestuota į ją patalpinus Monte Karlo metodą realizuojančių programų kodus [11]. Sistemos funkcionalumas anot autorių yra didesnis nei *Scilab* ir daug prilygsta *Mathematica* funkcionalumui. [6]



Pav.4. Internetinės matematinio programavimo ir modeliavimo sistemos schema. [6]

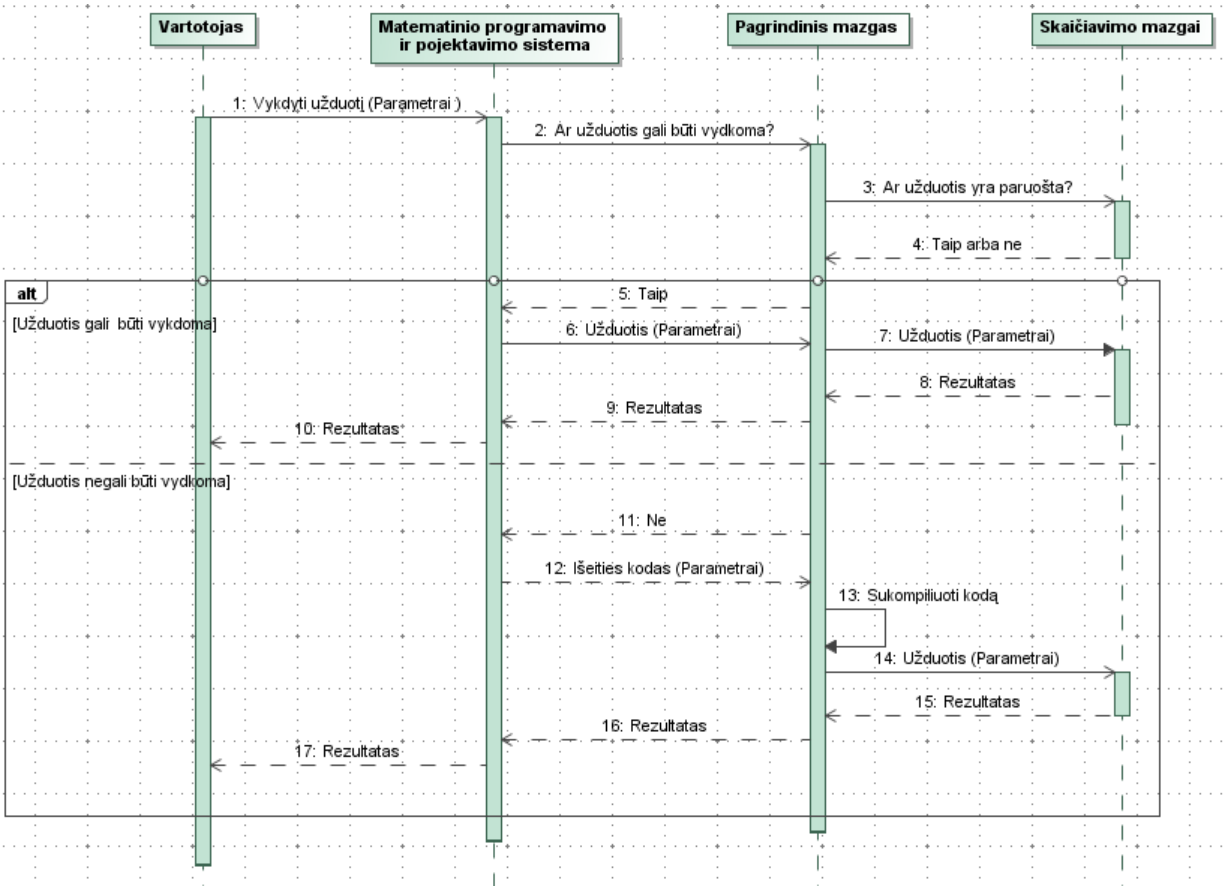
Sistema susideda trijų pagrindinių dalių: vartotojo sąsajos (Graphic User Interface – GUI), duomenų bazių valdymo sistemos (Database Management System – DBMS (DBVS)) bei serverinės programos (Server Application) (žr. Pav. 4.). Serverinė programa apjungia *Fortran*, *C++*, *Java* ir *C++* su *Qt* biblioteka kompiliatorius. Taip pat ji turi sąsają su grafine vartotojo sąsaja, kuri leidžia iškviesti vykdymo (ExecutePrograms()) ir kompiliavimo (CompilePrograms()) funkcijas. [6]

Pradiniame projekte numatyta sukurti debesų kompiuterijos paslaugą, kuri galėtų pakeisti internetinės matematinio programavimo ir projektavimo sistemos serverinę programą (žr. Pav. 4.). Paslauga turėtų turėti kiek įmanoma mažesnę sankiba su pačia sistema. Tam, kad būtų įmanoma tai pasiekti, reikėtų vykdyti netik skaičiavimo bet ir kompiliavimo procesus. Todėl tiekiama paslauga turėtų turėti reikiamus kompiliatorius (žr. Pav. 5.).



Pav. 5. Pradinio projekto debesies vaizdas

Matematinio programavimo ir modeliavimo sistemai norint vykdyti tam tikrą užduotį, pirmiausia jai reikėtų kreiptis į paslaugą su užklausa: ar užduotis gali būti vykdoma? Tai yra reikėtų patikrinti ar jau yra debesyje sukompiliuota užduotis, kurią būtų galima vykdyti. Jei sukompiliuotos užduoties nėra, tada sistema turėtų vėl kreiptis į paslaugą su išeities kodu, bei programavimo kalbos pavadinimu (*c++*, *qt, fortran*), ir pradinėmis reikšmėmis. Debesis turėtų sukompiliuoti pageidaujamą užduotį ir nedelsiant ją įvykdyti grąžinant rezultatą, taip sumažinant papildomų užklausų siuntimą. Jei debesyje užduotis jau sukompiliuota, tada viskas kiek paprasčiau, sistema siunčia tik reikiamus parametrus, ir paslauga grąžina rezultatą (žr. Pav. 6).



Pav. 6. Paslaugos sekos diagrama.

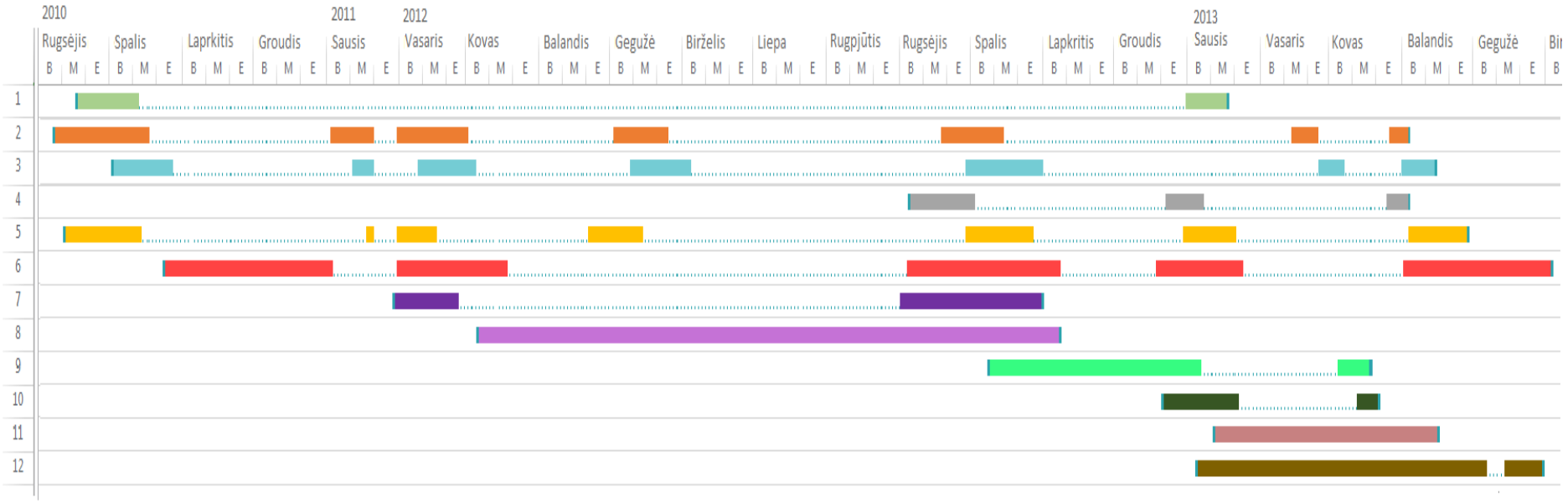
Debesies viduje turėtų būti centrinis mazgas. Šis mazgas aptarnauja likusius mazgus, kurie atliks pačius skaičiavimo darbus. Centrinis mazgas gavęs užklausą iš matematinės sistemos peržiūri visus mazgus, ieškodamas norimos užduoties. Jeigu užduoties rasti nepavyko tada, prašomas išeities kodas ir kompiliavimas vyksta centriniame mazge. Sukompiliuota užduotis įkeliama į tą mazgą, kurio apkrova tuo metu yra mažiausia ir užduotis iškarto įvykdoma. Rezultatas grąžinamas sistemai, jei užduotis rasta bent viename iš mazgų, tada gaunami reikalingi parametrai ir užduotis įvykdoma.

Skaičiavimo mazgų sankiba tampa labai maža, jei atjungsime vieną iš mazgų debesis neras užduoties, ji bus perkompiliuojama ir patalpinama kitame mazge (žr. Pav. 6).

Skaičiavimo mazgų funkcija tokiu atveju tampa labai paprasta, saugoti užduotis ir esant reikalui jas vykdyti. Pagrindinis vaidmuo tenka valdančiajam mazgui, kuris turėtų prižiūrėti užduočių prieinamumą, ir taip pat atsakyti už bendravimą su sistema. Be to, dar turėtų turėti galimybę peržiūrėti mazgų apkrovas, skirtas naujos užduoties talpinimui ir vykdymui.

3. Realizacinė dalis

3.1.Darbų eigos grafas



Pav. 7. Darbų eigos grafas

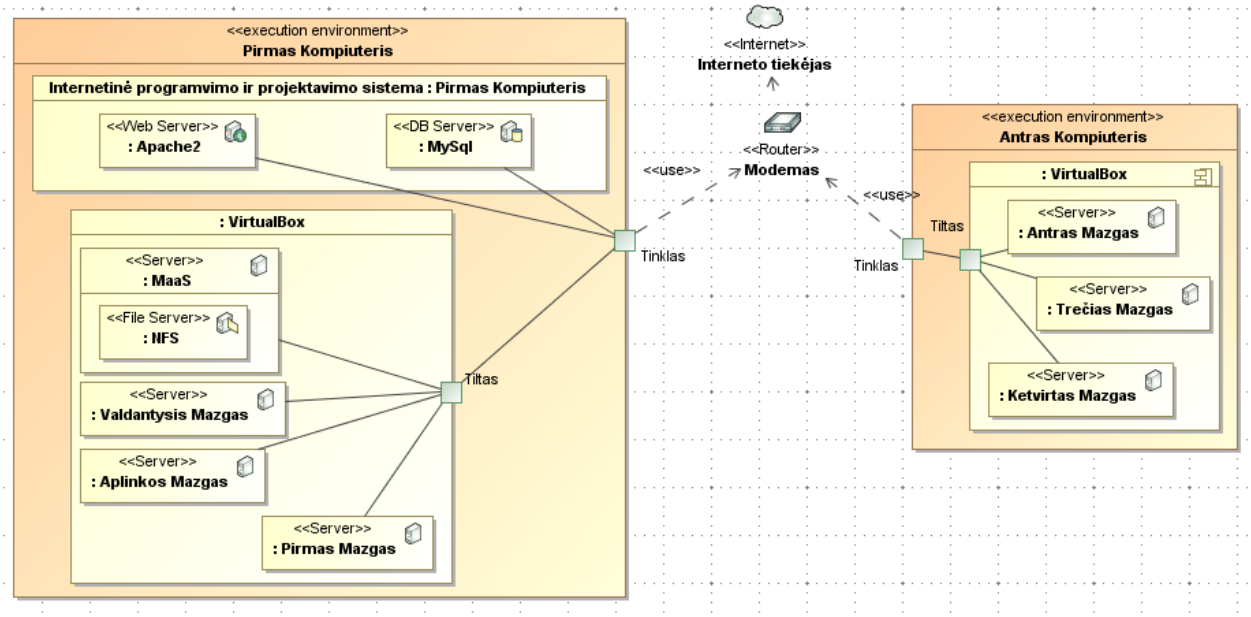
Galutiniame darbo eigos grafe (žr. Pav. 7) galima pastebėti, kaip pasikeitė darbų vykdymo laikas lyginant su darbų vykdymo planu.

Darbų paaiškinimai:

1. Debesų kompiuterijos paslaugos realizavimui skirtų reikalavimų formulavimas.
2. Informacijos paieška.
3. Informacijos šaltinių analizė.
4. Debesų kompiuterijos paslaugos projektui reikalingų *UML* diagramų braižymas.
5. Konsultacijos su darbo vadovu.
6. Darbo aprašymo sudarymas.
7. Techninės įrangos konfigūravimas.
8. Debesies platformos diegimas.
9. Debesies paslaugos aprašymas.
10. Debesies paslaugos publikavimas.
11. Internetinės matematinio programavimo ir modeliavimo sistemos ir sukurtos paslaugos derinimas.
12. Sukurtos paslaugos testavimas.

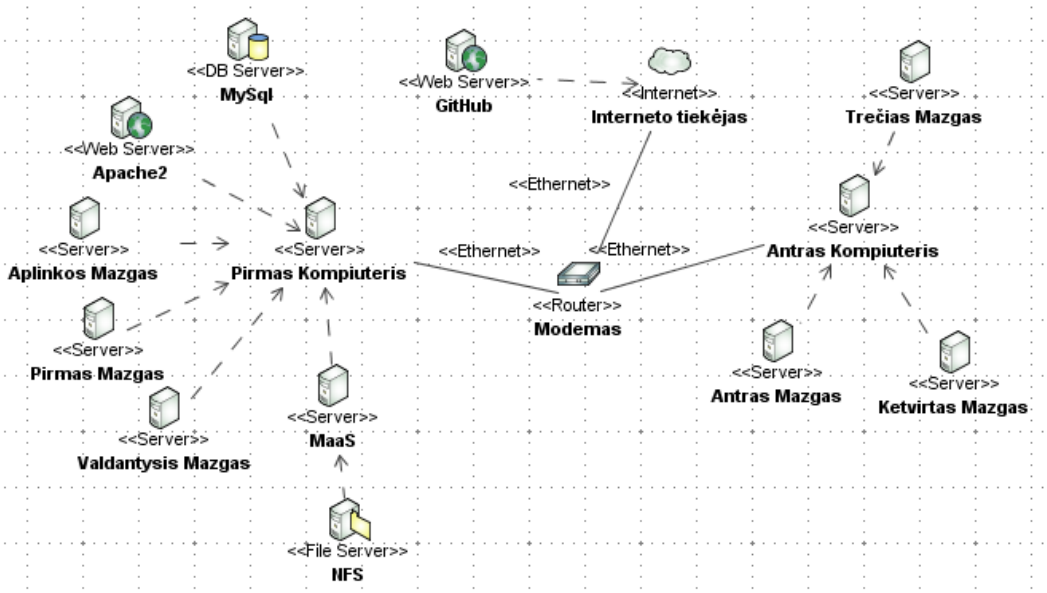
3.2.Galutinis projekto aprašymas

Suprojektuota ir realizuota testavimo aplinka, kurioje buvo įdiegta debesies platformos paslauga. Sistemą sudaro du kompiuteriai, kuriuose įdiegtos virtualios mašinos. Pirmame kompiuteryje buvo patalpinta matematinio programavimo ir projektavimo sistema bei įdiegta *VirtualBox* programinė įranga, kurioje buvo paleistos trys virtualios mašinos: debesies valdantysis (MaaS), valdantysis, aplinkos ir pirmasis mazgai. Antrajame kompiuteryje buvo įdiegta virtuali mašina *VirtualBox*, kurioje buvo paleisti dar trys darbiniai mazgai. Taigi, pati debesies platforma susideda iš MaaS ir aplinkos mazgų, o kuriama paslauga išnaudos likusius keturis darbinius mazgus bei valdantįjį mazgą (žr. Pav. 8).



Pav. 8. Tinklo komponentų diagrama.

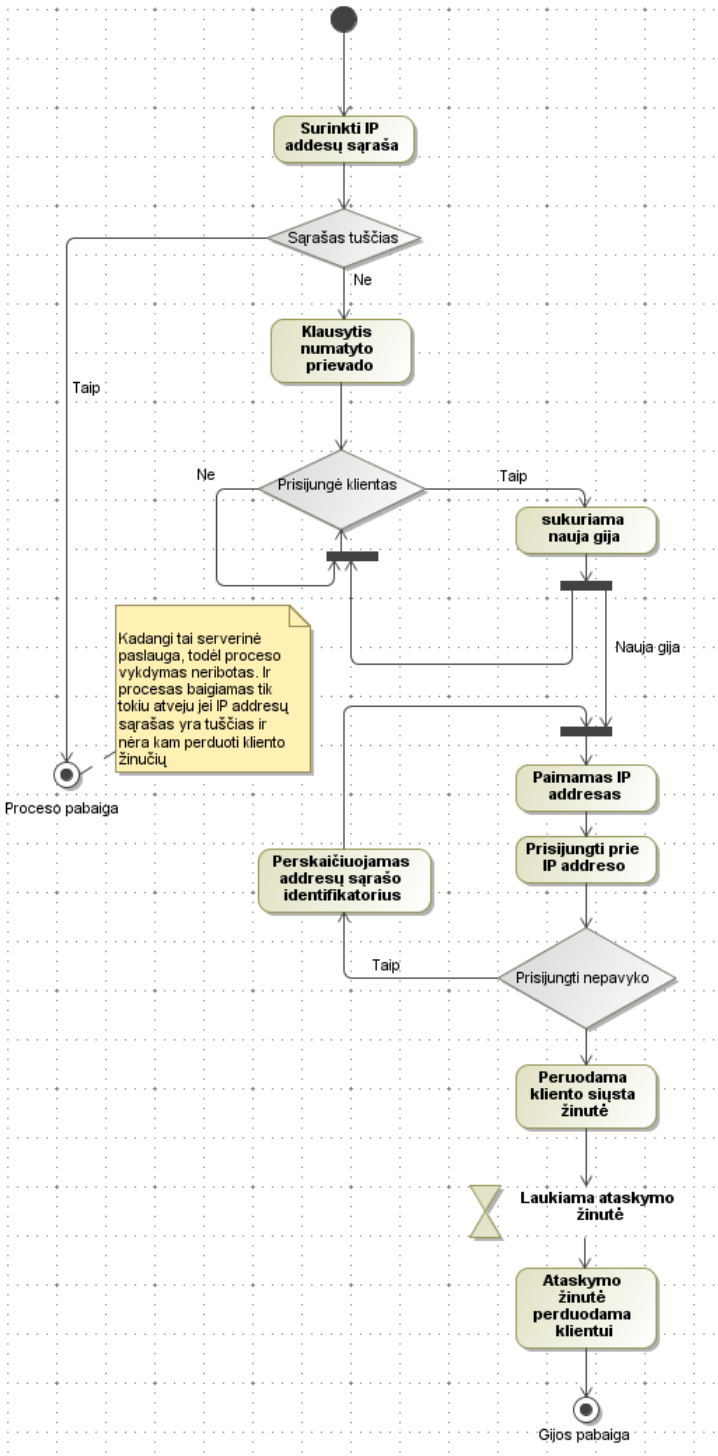
Tinklo požiūriu testinę aplinką sudaro modemas ir du kompiuteriai. O visi mazgai sudarantys privatų debesį yra patalpinti virtualiose mašinose ir į modemą kreipiasi tik per pagrindinius kompiuterius naudojant tilto (bridge) sąsaja (žr. Pav. 9). Tilto sąsaja leidžia virtualias mašinas sujungti į bendrą tinklą, tokiu būdu virtualios mašinos esančios pirmame kompiuteryje matomos viename tinkle su virtualiomis mašinomis iš antrojo kompiuterio. Taip pat trečiojoje ir antrojoje diagramose matome papildomus serverius *MySql*, *Apache2* ir *NFS*. Pirmieji du papildomi serveriai reikalingi matematinės programavimo ir projektavimo sistemos realizavimui, o *NFS* serveris reikalingas realizuoti debesų kompiuterijos paslaugai (žr. 3.3 Skyrelį).



Pav. 9. Fizinė tinklo schema.

Suprojektuota ir realizuota sudėtinė debesų kompiuterijos paslauga skirta internetiniai matematinio programavimo ir modeliavimo sistemai (OMPSS - Online Mathematical Programming and Simulation System). Suprojektuotą sistemą sudaro dvi paslaugos: pirmoji – OMPSS sistemos serverinė paslauga (OMPSS-Server) ir antroji paslauga – internetinio matematinio programavimo ir modeliavimo sistemos serverinių paslaugų kontroliavimo paslauga (OMPSS-Controller).

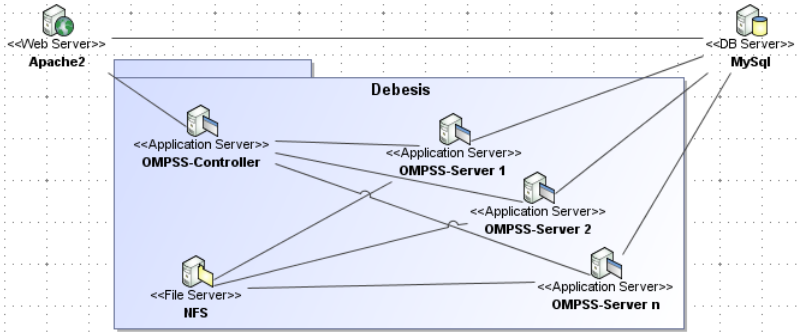
OMPSS-Server paslaugos branduolys buvo realizuotas, kitame baigiamajame darbe (Server Application) [6], tačiau jį teko pakoreguoti norint pritaikyti *OpenStack* debesies platformai. Visų pirma, ankstesnė OMPSS-Server versija buvo pritaikyta *Microsoft Windows* operacinei sistemai, tačiau naudojama debesies platforma paremta *Ubuntu*. Be to, ankstesnė versija taip pat turėjo ir grafinę vartotojo sąsaja, kuri netinkama paslaugos eksploatavimui debesies aplinkoje. Buvo peržiūrėtas ir pakeistas serverinės programos išeities kodas taip, kad ją butu galima vykdyti *Ubuntu* aplinkoje be grafinės vartotojo sąsajos. Taip pat ankstesnioji OMPSS-Server versija buvo naudojama ir realizuota kaip programa, o tai neatitinka paslaugos kriterijų, todėl buvo pridėta papildomų scenarijų, kurie leidžia panaudoti minėtą programą kaip paslaugą *Ubuntu* aplinkoje.



Pav. 10. Ryšiai tarp paslaugų

Realizuota OMPSS-Controller paslauga (žr. Pav. 10.), kuri leidžia sudaryti neribotą OMPSS ir OMPSS-Server jungčių kiekį. Pagal OMPSS standartus vienu metu sistema gali palaikyti tik vieną ryšį su OMPSS-Server. Tačiau debesyje OMPSS-Server paslaugų turėtų būti kur kas daugiau. Todėl neišvengiamai atsirado poreikis tokios paslaugos, kuri palaikytų ryšį vienas-prie-vieno (one-to-one) su OMPSS ir tuo pat metu galėtų bendrauti su visomis OMPSS-Server paslaugomis

esančiomis debesyje. Taigi OMPSS-Controller kiekvieną naują užklausą persiunčia vis kitam OMPSS-Server paslaugą turinčiam mazgui (žr. Pav. 11). Mazgo parinkimo algoritmas yra ganėtinai paprastas taupant serverio atsako laiką. Yra sudarytas mazgų, turinčių OMPSS-Server paslauga, IP (Internet Protocol – internetinio protokolo) adresų sąrašas, programa manipuliuoja identifikatoriumi, kuris yra perskaičiuojamas kiekvieną kartą kai OMPSS kreipiasi į OMPSS-Controller paslaugą. Jeigu serveris neatsako per tam tikrą laiko tarpą, tada daroma prielaida, kad serveris labai užimtas arba tiesiog išjungtas. Identifikatorius perskaičiuojamas ir vėl persiunčiama užklausa tačiau jau į kitą serverį.



Pav. 11. Ryšiai tarp paslaugų

Suprojektuoti ir realizuoti OMPSS-Service ir OMPSSC-Service paslaugų aprašymai skirti paslaugų diegimui į debesį, naudojant *JUJU Orchestration* įrankį. OMPSS-Service skirtas OMPSS-Server paslaugos aprašymui. Svarbiausia paslaugos aprašymo paketo dalis – įdiegimo scenarijus, jame aprašytas nuoseklus algoritmas kaip turėtų būti įdiegiama OPMSS-Server paslauga, t.y. kokios bibliotekos turi būti įdiegiamos, kokios nuorodos sukurtos ir kokie kompiliatoriai bus įdiegiami bei kokia seka viskas turi būti įgyvendinta. OMPSSC-Service skirtas OMPSS-Controller paslaugos aprašymui. Šis paslaugos aprašymo paketas turi papildomus scenarijus, kurie iškviečiami paslaugos tarpusavio ryšio sudarymo, nutrakimo bei atjungimo metu. Šie scenarijai papildo arba sumažina OMPSS-Controller IP adresų sąrašą priklausomai nuo to ar ryšys tarp mazgų yra sudaromas ar nutraukiamas. Ryšys tarp paslaugų gali būti nutraukiamas dviem būdais – sunaikinant egzistuojantį ryšį tarp paslaugų debesies aplinkoje arba išjungiant vieną iš mazgų, kuris tuo metu paliko ryšį. Taigi paslaugų diegimas ir valdymas tampa pilnai automatizuotas, reikalaujantis labai nedidelio vartotojo įsikišimo: nesudėtingų konfigūracijų bei komandų.

3.3. Problemos ir jų sprendimo būdai.

Pirmoji problema, kuri iškilo norint pasiekti šio darbo tikslą buvo **tinklo konfigūravimas**. Reikėjo sujungti i bendrą tinklą virtualias mašinas, bei fizines mašinas taip, kad jie nebūtų padalinti į atskirus potinklius. Standartinės virtualių mašinų konfigūracijos sujungia jas į atskirus virtualius

potinklius, tokie nustatymai netinka nes pirmojo kompiuterio virtualios mašinos neturi galimybės pasiekti antrojo kompiuterio virtualias mašinas. Šiai problemai išspręsti, virtualių mašinų tinklo nustatymai buvo pakeisti taip jog virtualios tinklo plokštės naudotų tilto sąsaja, taip visos virtualios ir fizinės mašinos pateko į viena tinklą.

Antroji problema sekė iškart po pirmosios, **valdantysis mazgas vis tiek negalėjo prisijungti** prie darbinių mazgų. Reikėjo sugeneruoti dvi *RSA* raktų poras pasinaudojant komanda *ssh-keygen* tiek pagrindinio vartotojo tiek administratoriaus teises turinčio vartotojo ir juos patalpinti MaaS administravimo aplinkoje. Tokiu būdu raktai automatiškai įdiegiami prijungiant naujus darbinius mazgus ir tuomet be didesnių keblumų galima prisijungti prie darbinių mazgų ir atlikti reikiamus pakeitimus. Raktų poros turi būti patalpintos MaaS administravimo aplinkoje iki mazgų prijungimo.

Kur **kas didesnė problema** iškilo projektuojant sudėtinę debesies paslaugą. OMPSS sąsaja su serveriu yra ganėtinai paprasta, OMPSS kreipiasi į serverį tik tada, kai reikia kompiliuoti programą arba ją vykdyti. **Serveris** niekada **nesikreipia į OMPSS**, jis tik grąžina identifikatorius pagal įvykdytas užklausas ir atsakymus pateikia tiesiai į duomenų bazę. Taigi, čia iškyla sukompiliuotų programų tarp mazgų sinchronizacijos problema. Sakykime, jeigu programa buvo sukompiliuota pirmajame mazge, o ją vykdyti bus užduota sekančiam mazgui, jis neturės galimybės įvykdyti programą. OMPSS sistemoje nėra realizuota galimybė pakartotinai paprašyti išeities kodo, kad serveris galėtų persikompiliuoti programos vykdomąjį failą. Sprendimo variantai yra keli:

- Realizuoti papildomą OMPSS funkcionalumą, serveriui pateikus tam tikras užklausas atliktų pageidaujamus veiksmus. Pavyzdžiui, atsiųsti kompiliavimo kodą, išsaugoti įvykdytos programos rezultatą ir panašiai.
- Valdančiajame mazge saugoti visus išeities kodus ir jeigu kompiliavimo-vykdymo mazgas neturi reikiamos programos, jis kreiptųsi į valdantįjį mazgą, šis grąžintų išeities kodą ir tokiu būdu kompiliavimo-vykdymo mazgai galėtų susikompiliuoti trūkstamas programas.
- Sukurti bendro naudojimo tinkle aplanką, kuris būtu prieinamas visiem mazgams ir tokiu būdų visos programos sukompiliuotos bet kuriame kompiliavimo-vykdymo mazge būtų prieinamos visiem debesyje esantiems mazgams.

Pirmasis problemos sprendimo būdas būtų universaliausias, nes tada būtų galima realizuoti ir daugelį papildomų funkcijų ateityje, tačiau reikėtų papildomai išanalizuoti ir pakeisti nemažą dalį OMPSS. Antrasis būdas papildomai apkrauna tiek valdantįjį mazgą, tiek ir kompiliavimo-vykdymo mazgus. Kadangi pirmieji du variantai pasirodė reikalaujantys nemažai papildomų resursų, todėl šio darbo ribose buvo pasirinkta realizuoti trečiąjį variantą. MaaS mazge buvo sukurtas aplankas ir naudojant *NFS* (Network File System – tinklinė failų sistema) buvo nustatytos atviros prieigos

teisės. Ir papildytas OMPSS-Service įdiegimo scenarijus taip, kad jei įgalinta *NFS* paslauga tada prisijungti prie bendro naudojimo tinkle aplanko. Išsprendus šią problemą gautas papildomas funkcionalumas. Į bendrojo naudojimo aplanką patalpinus konfigūracinį failą, galima naudoti bendrus nustatymus visiems kompiliavimo-vykdymo mazgams. [12]

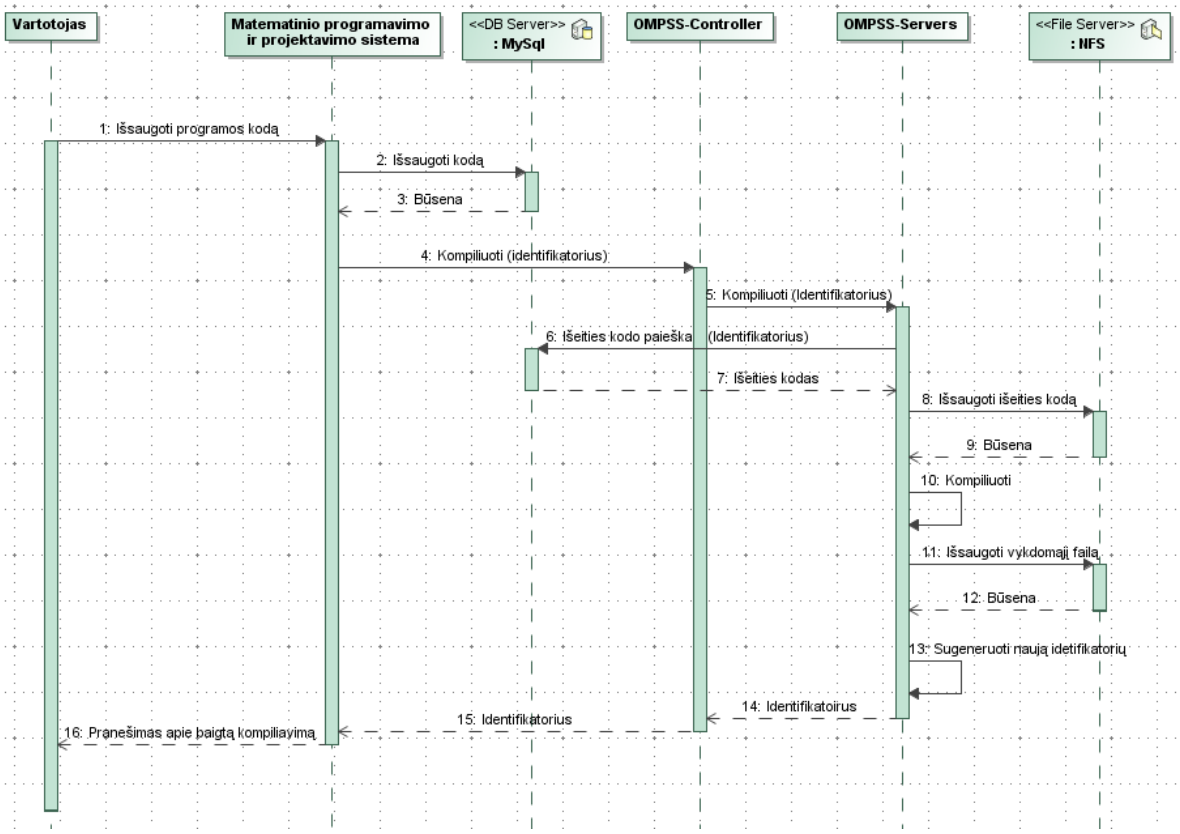
Realizuojant OMPSSC-Service ryšio tarp paslaugų sudarymo ir nutraukimo scenarijus paaiškėjo jog **komanda *relation-get ip* neveiksni**, todėl teko ieškoti alternatyvaus būdo kaip gauti kito mazgo IP adresą. Panaudojus komandą *relation-get private-address* gaunamas domenas. Tačiau domenas vėlgi nėra tinkamas IP adreso paieškai, nes jis modifikuotas debesies aplinkos. Modifikacija ganėtinai paprasta, prie egzistuojančio domeno prijungta *.localdomain* eilutė. Tačiau sistema neranda tokio domeno adreso, nes jis nėra registruotas domeno vardų serveryje (DNS – domain name server). Scenarijų teko pakoreguoti taip, kad jis gautą domeną padalintų į atkarpas ties tašku ir pirmą dalį naudotų kaip domeną. Tačiau diegiant sistemą į naują aplinką, reikėtų patikrinti kokie domeno vardai yra generuojami ir jei modifikacija kitokia, reikia pakoreguoti ryšio tarp paslaugų sudarymo ir nutraukimo scenarijus.

3.4. Darbo rezultatų analizė

Patalpinus sukurtas paslaugas į debesį ir jas sujungus tarpusavyje gaunamas OMPSS-Server paslaugų masyvas apjungtas OMPSS-Controller paslauga. Pradiniame projekte buvo numatyta jog kompiliavimo ir vykdymo mazgai neturės tiesioginio ryšio su internetinės matematinio programavimo ir projektavimo sistemos dalimis. Tačiau realizuojant projektą paaiškėjo jog ankstesnė OMPSS-Server versija turėjo tiesioginį ryšį su duomenų baze, norint išvengti OMPSS sistemos koregavimo šis ryšys išliko. Taip pat, pradiniame projekte buvo numatyta, jog sukompiliuotų programų išeities kodai bus saugomi valdančiajame mazge ir prireikus pasiųsti darbiniams mazgams, pakartotinam programos kompiliavimui. Bet realizuojant projektą paaiškėjo, jog pakartotinas programų kompiliavimas užima papildomą darbinių mazgų laiką bei papildomai apkrauna OMPSS-Controller paslaugą (žr. 3.3. Problemos ir jų sprendimo būdai).

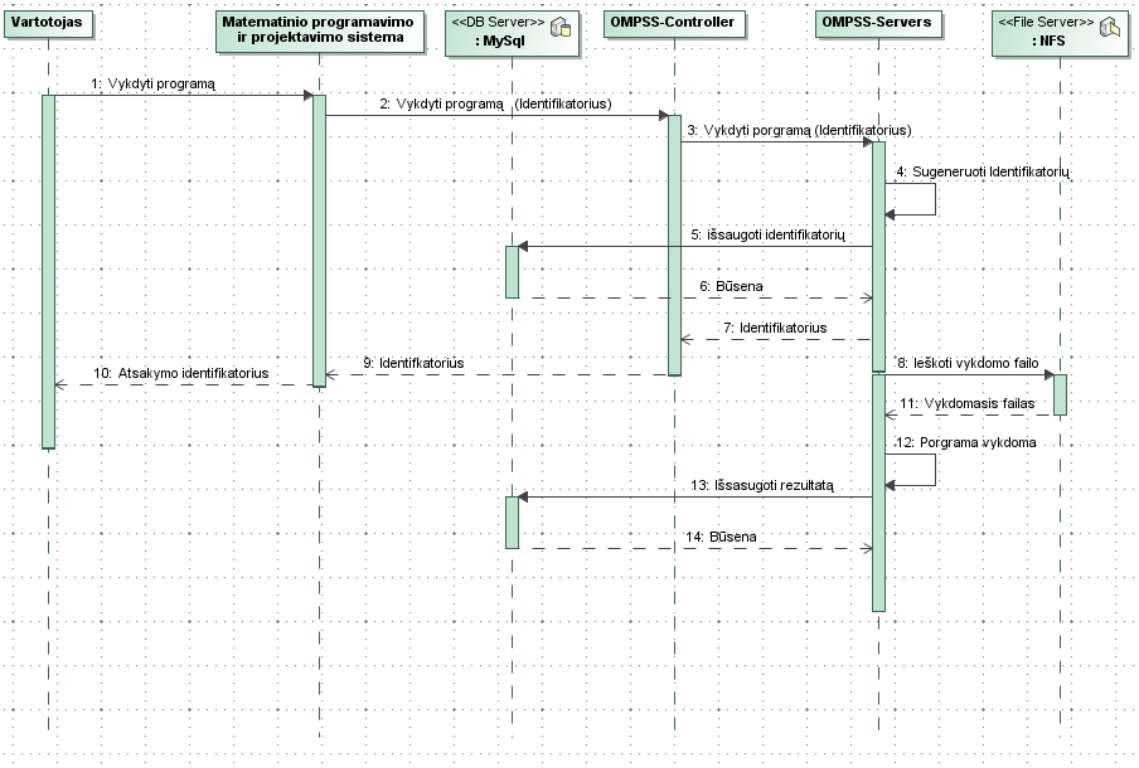
Kintant projektui realizacijos metu pakito kompiliavimo ir vykdymo seka. Pradiniame projekte buvo numatyta jog kompiliavimas tūrėtų būti įtrauktas į vykdymo seką. Tačiau realizavimo metu kompiliavimas buvo atskirtas nuo vykdymo ir gautos atskiros dvi atskiros sekos. Tarkime, kad OMPSS vartotojas aprašė naujos programos kodą arba atnaujino jau egzistuojantį ir paspaudė išsaugoti. Tada OMPSS sistema kreipiasi į debesį persiųsdama programos identifikatorių, ten užklausą prima OMPSS-Controller paslauga ir kreipiasi į mazgą turintį OMPSS-Server paslaugą. Pagal gauta identifikatorių OMPSS-Server kreipiasi į duomenų bazę prašydama išeities kodo. Jį

gavus paslauga sukompiliuoja programą ir grąžina atsakymą OMPSS-Controller, kuris atsakymą grąžina atgal į OMPSS (žr. Pav. 12).



Pav. 12. Kompiliavimo seka

Vykdymo metu OMPSS kreipiasi į debesį, kitaip tariant OMPSS-Controller su parametrų sąrašu bei programos identifikatoriumi. Valdantysis mazgas vėl persiunčia užklausą vienam iš kompiliavimo-vykdymo mazgų. OMPSS-Server gavęs užklausą sugeneruoja specialų paieškos raktą ir grąžina jį OMPSS-Controller paslaugai, kuri persiunčia jį atgal į OMPSS. Tada pagal gautą identifikatorių kompiliavimo ir vykdymo mazgas suranda ir iškviečia programą naudodamas pateiktus parametrus. Ir gautą rezultatą pateikia į duomenų bazę pagal paieškos raktą (žr. Pav. 13).



Pav. 13. Vykdymo seka

3.4.1. Debesų kompiuterijos paslaugos testavimas

Debesies testavimas buvo atliktas naudojant dvi užduotis. Pirmoji apskaičiuoti duotojo skaičiaus faktorialas, testavimo metu buvo skaičiuojamas dešimties faktorialas. Programos kodas nedidelis, užduotis greitai įvykdoma. Jos tikslas patikrinti koks paslaugos efektyvumas vykdant mažai resursų reikalaujančias užduotis. Antroji užduotis kur kas sudėtingesnė, tai lygiagreti programa realizuojanti Monte Kralo metodą [6][11]. Programos vykdymo laikas priklauso nuo devynių parametrų (skliausteliuose pateiktos reikšmės naudotos testavimo metu):

- Iteracijų skaičius kurie skaičiuojami lygiagrečiai (turis = 100);
- 1 etapo kintamųjų skaičius (mm = 20);
- 2 etapo kintamųjų skaičius (nm = 30);
- 1 etapo ribojimų skaičius (nt = 10);
- Didžiausias iteracijų skaičius (iter = 1000);
- Pradinis iteracijų skaičius (nn = 200);
- Mažiausias imties tūris (Lmin = 200);
- Didžiausias imties tūris (Lmax = 1000000);
- Norimo tikslumo skaičius (eps = 2).

Antroji užduotis leido patikrinti kaip paslauga vykdo sudėtingas užduotis. Testavimas buvo atliekamas trimis etapais. Pirmuoju etapu vienu metu buvo užduodama dešimt užduočių, antruoju šimtas užduočių, o trečiuoju šimtas penkiasdešimt. Skirtingas užduočių kiekis leidžia įvertinti kaip paslauga pajėgia susidoroti su skirtingu vartotojų kiekiu. Abi užduotys buvo realizuotos pasinaudojant C++ kalba su Qt biblioteka.

	10 vartotojų	100 vartotojų	150 vartotojų
10 faktorialas	1s	3s	16s
Monte Karlo metodas	8m 39s	1h 9m 47s	2h 21m 32s

Lentelė 3. Realizuotos debesų kompiuterijos paslaugos testavimo rezultatai.

Iš testavimo rezultatų galima pastebėti, jog vykdant mažai resursų reikalaujančią programą, padidinus vartotojų skaičių dešimt kartų (nuo dešimties iki šimto) vartotojų aptarnavimo laikas išauga tik tris kartus. O vartotojų skaičių padidinus penkiolika kartų (nuo dešimties iki šimto penkiasdešimt) aptarnavimo laikas pailgėja iki šešiolikos kartų. Aptarnaujant šimtą vartotojų paslauga buvo dvidešimt aštuoniais procentais efektyvesnė nei aptarnaujant šimtą penkiasdešimt. Vykdant Monte Karlo metodą padidinus vartotojų skaičių dešimt kartų aptarnavimo laikas pailgėjo aštuonis kartus, o padidinus penkiolika kartų aptarnavimo laikas pailgėjo šešiolika kartų, kaip ir skaičiuojant faktorialą. Įdomu tai, kad skaičiuojant faktorialą ir padidinus vartotojų skaičių dešimt kartų aptarnavimo laikas išauga tik trimis kartais, kai vykdant lygiagrečią Monte Karlo metodą realizuojančią programą vykdymo laikas išauga iki aštuonių kartų. O padidinus vartotojų skaičių po penkiolika kartų vykdymo laikas išauga vienodai. Tai leidžia manyti jog efektyvumo riba yra peržengiama kažkur tarp šimto ir šimto penkiasdešimties vartotojų ir paslauga išnaudojama nebe efektyviai, todėl sugaištas laikas padidėja daugiau kartų negu yra padidinamas vartotojų skaičius.

3.4.2. Debesų kompiuterijos paslaugos palyginimas

Debesų kompiuterijos paslauga buvo palyginta su šiuo metu internetinėje matematinio programavimo ir modeliavimo sistemoje naudojamu užduočių kompiliavimo ir vykdymo moduliu bei tuo pačiu moduliu transformuotu į *Ubuntu* operacinę sistemą. Kaip ir testuojant debesų kompiuterijos paslaugą (žr. 3.4.1. Skyrelis) testuojant šiuos modulius buvo atliekamos tos pačios užduotys.

	10 vartotojų	100 vartotojų	150 vartotojų
10 faktorialas	1s	4s	6s
Monte Karlo metodas	12m 49s	2h 10m 16s	3h 15m 27s

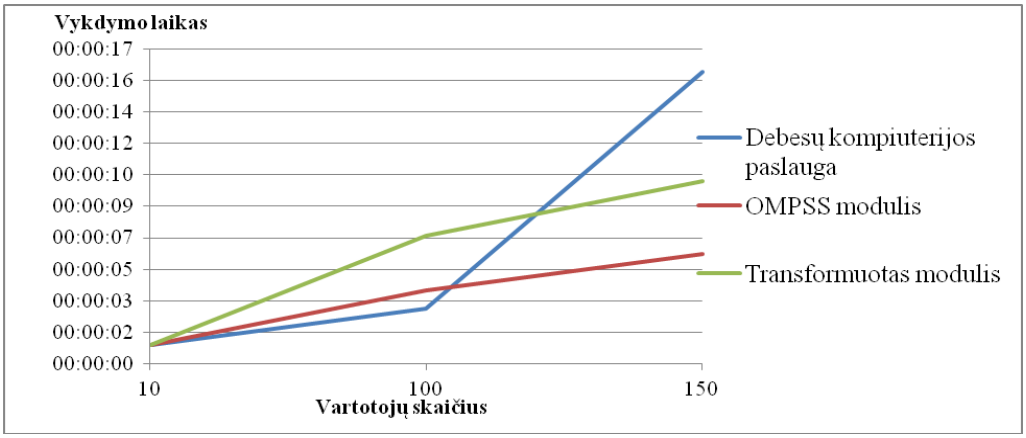
Lentelė 4. Šiuo metu OMPSS naudojamo modulio testavimo rezultatai

Ketvirtoje lentelėje pateikti rezultatai atliekant skaičiavimus su šiuo metu OMPSS naudojamu užduočių kompiliavimo ir vykdymo moduliu. Atliekant pirma užduotį ir padidinus vartotų skaičių dešimt kartų (nuo 10 iki 100) aptarnavimo laikas pailgėjo keturiais kartais, o padidinus penkiolika kartų (nuo 10 iki 150) aptarnavimo laikas pailgėjo šešiais kartais. Vykdant antrą užduotį ir padidinus vartotojų skaičių dešimčia kartų, aptarnavimo laikas prailgėjo dešimt kartų o padidinus penkiolika kartų aptarnavimo laikas prailgėjo taip pat penkiolika kartų. Tai parodo jog sudėtingos programos vykdymo laikas auga proporcingai pridėtam vartotojų skaičiui. (žr. Lentelė. 4.)

	10 vartotojų	100 vartotojų	150 vartotojų
10 faktorialas	1s	7s	10s
Monte Karlo metodas	17m 56s	2h 54m 21s	4h 24m 04s

Lentelė 5. Transformuoto kompiliavimo ir vykdymo modulio testavimo rezultatai.

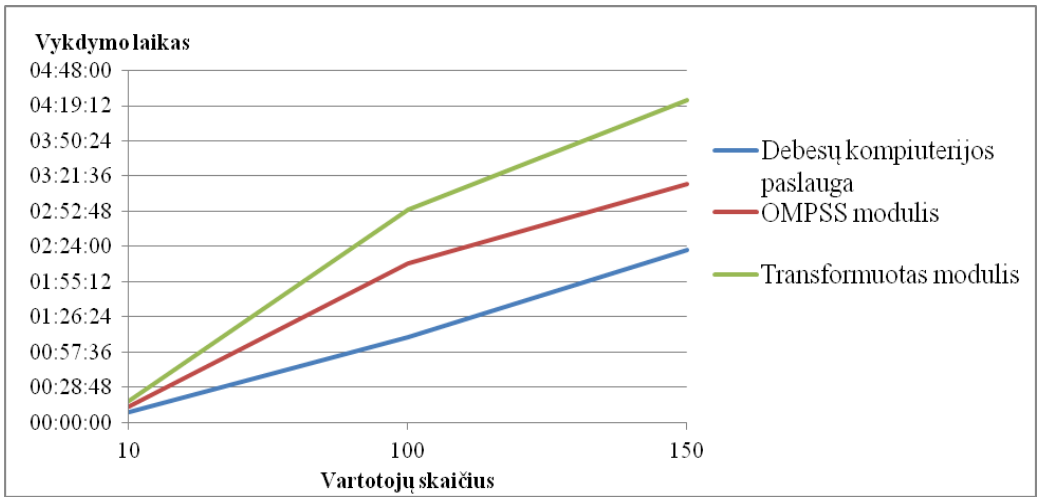
Penktojoje lentelėje pateikti rezultatai atliekant skaičiavimus su transformuotu, užduočių kompiliavimo ir vykdymo, moduliu. Atliekant pirmą užduotį ir padidinus vartotojų skaičių dešimt kartų aptarnavimo laikas padidėjo septyniais kartais, o padidinus penkiolika kartų aptarnavimo laikas padidėjo dešimčia kartų. Vykdant antrą užduotį pastebėta jog vykdymo laikas vienodai proporcingas su šiuo metu naudojamo modulio vykdymo laikais, nors transformuotas modulis vartotojus aptarnauja lėčiau.



Pav. 14. Faktorialo užduoties vartotojų aptarnavimo laikas

Atlikus bandymus pastebėta jog vykdant mažai resursų reikalaujančias užduotis, tokias kaip faktorialo skaičiavimas, paslauga efektyviau aptarnauja iki šimto vartotojų, po to jo efektyvumas

ženkliai sumažėja lyginant su moduliais (žr. Pav. 14.). Aptranaujant šimta pekiasdešimt vartotojų debesų kompiuterijos paslauga sugaišta šešiomis sekundėmis ilgiau nei transformuotas modulis ir dešimčia sekundžių iligiau negu šiuo metu internetinėje matematinio programavimo ir modeliavimo sistemoje naudojamas užduočių kompiliavimo ir vykdymo modulis.



Pav. 15. Mote Karlo metodo užduoties vartotojų aptarnavimo laikas

Monte Karlo metodo vartotojų aptarnavimo laiko grafikas kiek kitoks. Debesų kompiuterijos paslaugos vartotojų aptarnavimo laikas išlieka trumpiausias nepriklausomai nuo vartotojų skaičiaus. Taip pat aiškiai matoma jog kuo didesnis vartotojų skaičius tuo apatarnavimo laikas trumpesnis lyginiant su transformuotu ir šiuo metu nauodjamu kompiliavimo ir vykdymo moduliu (žr. Pav. 15.).

3.5. Patarimai, pastebėjimai, rekomendacijos

Ateityje būtų galima patobulinti OMPSS-Controller paslaugą taip, kad ji atrinktų kompiliavimo ir vykdymo mazgus, atsižvelgdama į daugiau parametrų. Šiuo metu paslauga tiesiog persiunčia užklausas, kitiems darbiniams mazgams, eilės tvarka ir jeigu mazgo atsakas trunka ilgiau nei numatyta, jis laikomas užimtu. Tačiau būtų kur kas efektyviau jei valdantysis mazgas pasiuntęs tam tikrą užklausą į mazgą gautų atsakymą, kuriame būtų informacija apie mazgo resursų užimtumą, vykdomų užduočių skaičių ir jų eilę. Vertindamas šią informaciją darbinis mazgas perduotų užduotis kompiliavimo ir vykdymo mazgams, tokiu atveju sumažėtų tikimybė jog naują užduotį gaus sunkiai apkrautas mazgas.

Taip pat, sprendžiant problemą susijusią su mazguose turimų programų sinchronizaciją buvo apžvelgti trys sprendimo būdai ir pasirinktas trečiasis realizuojant NFS. Bet šis sprendimo būdas galėtų būti pakeistas pirmuoju, kuriame buvo siūloma realizuoti papildomą OMPSS funkcionalumą ir taip suteikti galimybę serveriui kreiptis į OMPSS su tam tikromis užklausomis. Šis sprendimas taip pat sumažintų tinklo apkrovą. Kompiliavimo ir vykdymo mazgams nebereikėtų palaikyti ryšio

su duomenų bazės serveriu bei būtų galima atsisakyti NFS serverio paslaugų. Debesies sankiba su OMPSS sumažėtų dar labiau.

Ankstesnėje OMPSS-Server versijoje buvo galima naudoti *Netlib Repository LAPACK* biblioteką, kurios metodus galima naudoti *C++* ir *Fortran 90* programavimo kalbose. Šis funkcionalumas šiuo metu nėra realizuotas, todėl reikėtų patikrinti ar ši biblioteka prieinama *Ubuntu* operacinėje sistemoje. Jeigu biblioteka neturi galimybės būti naudojama debesies platformoje, reikėtų rasti alternatyvias bibliotekas, kurios atstotų *LAPACK* funkcionalumą.

Ateityje būtų galima patobulinti OMPSS-Server ir OMPSS-Controller taip, kad būtų galima nustatyti registravimo lygius ir registruoti vykstančius procesus. Toks patobulinimas leistų lengviau administruoti paslaugas, būtų galima peržiūrėjus registrus nustatyti kurios programos buvo sukompiliuotos kurios buvo įvykdytas, bei kiek laiko užtruko mazgas. Ankstesnė OMPSS-Server turėjo grafinę sąsają ir ši informacija buvo matoma realiu laiku, panaikinus šią sąsają tapo sudėtingiau sekti ir testuoti bendrą sistemos funkcionavimą.

Išvados

1. Atlikus siūlomas debesų kompiuterijos paslaugų rinkos analizę nustatyta kad siūlomi **dviejų tipų produktai**: paslaugoms teikti ir paslaugoms kurti. Iš aštuonių galimų kandidatų pagal iš anksto apibrėžtus kriterijus (2 Lentelė), darbo tikslui realizuoti pasirinkta *OpenStack* platforma ir *JUJU Orchestration* priemonė.
2. Projektuojant privatų debesį su ***OpenStack*** platformą, **serverių skaičių** galima paskaičiuoti pasinaudojus formule ***n+2,*** kur *n* diegiamų paslaugų skaičius.
3. Sukurtas projektas (modelis), kaip **Internetinėje matematinio programavimo ir modeliavimo sistemoje** naudojamas užduoties kompiliavimo ir vykdymo **modulis** (skirtas Windows sistemai viename kompiuteryje) gali būti **transformuotas į debesų kompiuterijos paslaugą** (veikiančią Linux OS pagrindu, beveik neribotam kompiuterių skaičiui).
4. **Modelio realizacija ir testavimas** ne tik įrodė **IMPMS sistemos pritaikymo debesų kompiuterijos** bei kitų išskirstytųjų ir lygiagrečiųjų skaičiavimų sistemoms (taip pat ir komercinėms) **galimybę**, bet ir sudarė prielaidas **padidinti šios sistemos efektyvumą,** sudarė galimybes **spręsti sudėtingesnius**, daugiau kaip vieno kompiuterio išteklių reikalaujančius **uždavinius**.

Naudota literatūra

1. B. Hayes. *Cloud Computing: As software migrates from local PCs to distant Internet servers, users and developers alike go along for the ride.* Communications of the ACM, 2008.
2. B. Sotomayor, R. Santiago Motero, I. Martin Liorente, I. Foster. *Capacity Leasing in Cloud Systems using The OpenNebula Engine.*
3. *Charms* [interaktyvus] [žiūrėta 2013m. balandžio 20d]. Prieiga per internetą: <https://juju.ubuntu.com/docs/charm.html>
4. G. Pallis. *Cloud computing. The New Frontier of Internet Computing.* IEEE Internet Computing, 2010.
5. *Infrastructure as a Service* [interaktyvus] [žiūrėta 2012m. spalio 30d.]. Prieiga per internetą: <http://www.bestpricecomputers.co.uk/glossary/infrastructure-as-a-service.htm>
6. *Internetinė matematinio programavimo ir modeliavimo sistema. Sistemos kūrimas ir testavimas* [interaktyvs] [žiūrėta 2013m. sausio 19d.]. Prieiga per internetą: <http://tools.elaba.lt/marc/fulltext.php?lib=etd01&sys=000023092>
7. J. Loeliger. *Version Control with Git.* O'Reilly, 2009
8. *"Metal as a Service" provisioning tool from Canonical in Ubuntu 12.04 LTS Beta* [interaktyvus] [žiūrėta 2012m. spalio 30d.]. Prieiga per internetą: <http://www.canonical.com/content/%E2%80%9Cmetal-service%E2%80%9D-provisioning-tool-canonical-ubuntu-1204-lts-beta>
9. M. A. Vouk. *Cloud computing – Issues, Research and Implementations*. Journal of computing and Information Technology – CIT 16, 2008.
10. M. Cusumano. *Technology Strategy and Management. Cloud Computing and SaaS as New Computing Platforms.* Communications of the ACM, 2010.
11. M. Neimantas *Internetinė matematinio programavimo ir modeliavimo sistema. Stochastinio programavimo programų kūrimas ir analizė.* Šiaulių Universitetas, 2012.
12. *Network File System (NFS)* [interaktyvus] [žiūrėta 2013m. balandžio 20d]. Prieiga per internetą: <https://help.ubuntu.com/12.04/serverguide/network-file-system.html>
13. *OpenStack® is an open and scalable cloud computing platform for building private and public clouds* [interaktyvus] [žiūrėta 2012m. spalio 30d.]. Prieiga per internetą: <http://www.rackspace.com/cloud/openstack/>
14. *Platform As A Service: What Vendors Offer* [interaktyvus] [žiūrėta 2012m. spalio 30d.]. Prieiga per internetą:

<http://www.informationweek.com/news/services/saas/showArticle.jhtml?articleID=220300686>

15. R. Buyya. *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Computer Systems 25, 2009.
16. S. Bhardwaj, L. Jain, S. Jain. *An Approach of Investigating Perspective of Cloud Software-as-a-Service (SaaS).* International journal of Computers Applications, 2010.
17. *Ubuntu Cloud Infrastructure* [interaktyvus] [žiūrėta 2013m. balandžio 20d]. Prieiga per internetą: <https://help.ubuntu.com/community/UbuntuCloudInfrastructure>

Anotacija

Barauskas N. Debesų kompiuterijos paslaugos pritaikymas internetinei matematinio programavimo ir modeliavimo sistemai. Dr. V. Giedrimas. – Šiaulių universitetas, Matematikos ir informatikos fakultetas, 2013 – 55 p.

Šio darbo pagrindinis tikslas yra suprojektuoti ir realizuoti debesų kompiuterijos paslaugą internetinei matematinio programavimo ir modeliavimo sistemai, kuri galėtų pakeisti šios sistemos kompiliavimo ir vykdymo serverinę dalį. Šiam tikslui pasiekti buvo nagrinėjama debesies kompiuterijos samprata. Atlikta debesies platformų analizė bei jų palyginimas tarpusavyje. Taip pat apžvelgti debesų kompiuterijos paslaugų tipai. Identifikuojamos ir nagrinėjamos problemos kurios iškilo projektuojant ir realizuojant debesų kompiuterijos paslaugą. Sukurta debesų kompiuterijos paslauga galinti pakeisti šio metu internetinėje matematinio programavimo ir modeliavimo sistemoje naudojama kompiliavimo ir vykdymo modulį.

Pagrindiniai žodžiai: debesis, debesų kompiuterija, paslauga, debesų kompiuterijos paslauga, atviras kodas, internetinė matematinio programavimo ir modeliavimo sistema.

Summary

Barauskas N. The Application of Cloud Computing service for Online Mathematical Programming and Simulation System. Tutor: Dr. V. Giedrimas. – Šiauliai University, Faculty of Mathematics and Informatics, 2013 – 55 p.

The purpose of this work is to develop Cloud Computing service for Online Mathematical Programming and Simulation system, which can replace currently used compilation and execution server. To achieve this goal there was analysed Cloud Computing concept. Also Cloud Computing service types were reviewed. Moreover Cloud platforms there were analysed and compared to each other. Identified and analysed problems, which were raised in development process. Also was developed Cloud Computing Service which can replace currently used compiling and execution module in Online Mathematic Programming and Simulation system.

Key words: cloud, cloud computing, service, cloud computing service, open source, online mathematical programming and simulation system.

Priedai

1. DVD turinys

- Šakniniame kataloge galima rasti elektroninės magistro darbo aprašymo versijas (rinkmenos *aprasymas.doc* ir *aprasymas.pdf*) bei ktus katalogus.
- Kataloge *Diagramos* įdėtos visos su projektais susijusios diagramos.
- Kataloge *Programine iranga* yra patalpinti du aplankai:
    - *Windows* aplanke įkelta *VirtualBox* programinės įrangos diegimo rinkmena, bei *putty.exe* porgramėlė skirta prisijungti prie *Ubuntu* serverio imituotjant *ssh* komandą.
    - *Ubuntu* aplanke taip pat įkelta *VirtualBox* programinės diegimo rinkmenos skirtos šešiasdešimt keturių bei trisdešimt dviejų bitų operacinėms sistemos. Bei pridėtas įskiepis leidžiantis panaudoti virtualios mašinos pasikrovimą iš tinklo.
- Kataloge *Projektu iseities kodai* galima rasti projektų OMPSS-Controller, OMPSSC-Service, OMPSS-Server ir OMPSS-Service išeities kodus. Projektų išieties kodai taip gali būti prieinami internetu, pasinaudojus nuoroda: https://github.com/nereliz.
- Kataloge *Testines uzduotys* įkelti, testavimo metu nudotų, užduočių išeities kodai bei papildomas failas su parametrais, kurie buvo naudojami testavimo metu.
- Kataloge *Ubuntu atvaizdai* patalpintos ISO formato rinkmenos, kuriuose *Ubuntu12.04.2 LTS* operacinės sistemos diegimo ploksteliu atvaizdai. Ikeltos rinkmenos šešiasdešimt keturių ir trisdešimt dviejų bitų architektūroms.

2. Diegimas

2.1. Reikalavimai

Norint įdiegti šį projektą minimaliai reikia penkių fizinių ar virtualių mašinų. Taip pat papildomo kompiuterio su grafine vartotojo sąsaja tam, kad būtų galima prisijungti prie MaaS portalo. Pirmasis kompiuteris, MaaS serveris, bus paskirtas virtualių bei fizinių mašinų administravimui. Antrasis kompiuteris bus naudojamas aplinkos administravimui, tai yra paslaugų diegimui į mazgus ir jų palaikymui mazguose. Aplinkos mazgas įdiegiamas automatiniu būdu, tokiu pat būdu bus įdiegti ir visi kiti mazgai. Vienas mazgas bus skirtas OMPSS-Controller paslaugai o likę du OMPSS-Server paslaugoms patalpinti. Talpinti vieną OMPSS-Server paslaugą neverta, nes ji gali būti naudojama ir be debesies platformos, nereikalaujant didelio kiekio resursų.

MaaS virtualius ar fizinius mazgus prisijungia MAC adresų pagalba, bet šio darbo ribose buvo naudojamas vienas potinklis. Kadangi reikalavimai formuojami pagal testavimo aplinką, todėl visi

kompiuteriai turėtų būti viename potinklyje. Taip pat visi mazgai testinėje aplinkoje buvo šešiasdešimt keturių bitų architektūros, šis reikalavimas nėra svarbus tačiau labai svarbu, kad visi mazgai naudoto vienodos architektūros operacines sistemas kadangi jie dalinasi jau sukompiliuotomis programomis. Taip pat jei planuojama naudoti trisdešimti dviejų bitų sistemą reikėtų perkompiliuoti OMPSS-Server bei OMPSS-Controller.

Visi kompiuteriai, išskyrus MaaS serverį, kuris bus diegiamas rankiniu būdu, turi turėti galimybę būti paleistiems per tinklą ir pradėti darbą naudojant tinklą kaip pradinį paleisties įrenginį. Jeigu kompiuteriai neturi tokios galimybės, tokiu atveju automatinis tinklo mazgų įdiegimas neveiks ir tai teks daryti rankiniu būdu. Šio darbo ribose buvo naudojamas tik automatinis mazgų diegimas, todėl rankinis mazgų diegimo būdas nebus apžvelgtas.

Taigi apibendrinus reikalingas vienas kompiuteris, kuris bus skirtas MaaS serveriui jo techninės charakteristikos turėtų būti nemažesnės negu:

- Procesorių taktinis dažnis: 700 megahercų
- Operatyvioji atmintis: 512 megabaitų
- Kietasis diskas: 10 gigabaitų ir daugiau nes jame bus saugomos visos sukompiliuotos programos, bei mazgams skirti *Ubuntu* atvaizdai.

Darbinio mazgo su OMPSS-Controller paslauga techninės charakteristikos turėtų būti nemažesnės negu:

- Procesoriaus taktinis dažnis: 700 megahercų
- Operatyvioji atmintis: 512 megabaitų
- Kietasis diskas: 5 gigabaitai

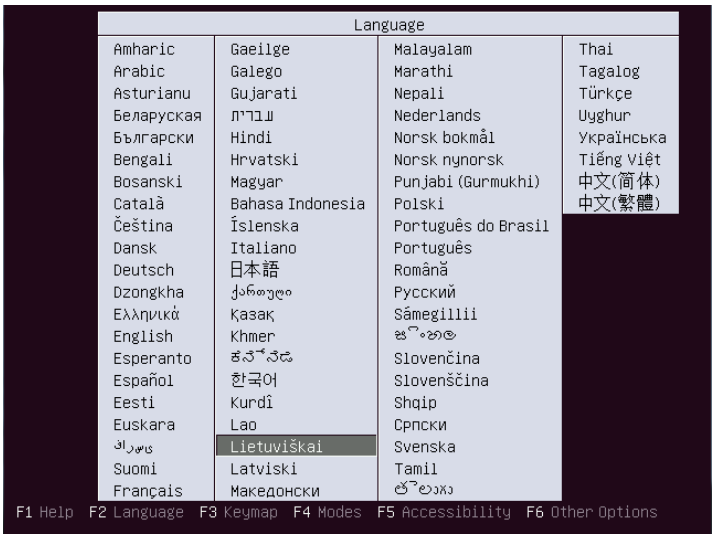
Darbinių mazgų su OMPSS-Server paslauga techninės charakteristikos turėtų būti nemažesnės negu:

- Procesoriaus taktinis dažnis: 700 megahercų ir daugiau, nes juose bus vykdomi skaičiavimai ir kompiliavimas
- Operatyvioji atmintis: 512 megabaitų ir daugiau nes juose bus vykdomi skaičiavimai ir kompiliavimas
- Kietasis diskas: 5 gigabaitai.

2.2. Debesies diegimo instrukcija

MaaS serverį galima įdiegti kartu su *Ubuntu 12.04.2 TLS*, kurį galima rasti DVD laikmenoje (žr. Preidai. 1 skyrelis). Taip pat šios operacinės sistemos diegimo kompaktinės plokštelės atvaizdą galima parsisiųsti pasinaudojus šia nuoroda: http://www.ubuntu.com/download/cloud.

Parsisiųstą rinkmeną reikia perkelti į kompaktinę plokštelę ar kitą atminties įrenginį, kurį būtų galima nustatyti kaip kompiuterio pradinį paleisties įrenginį. Pasileidus *Ubuntu* diegimo vedliui, visų pirma, reikia pasirinkti kuria kalba bus tęsiamas diegimo vedlys. Galima pasirinkti ir lietuvių kalbą.

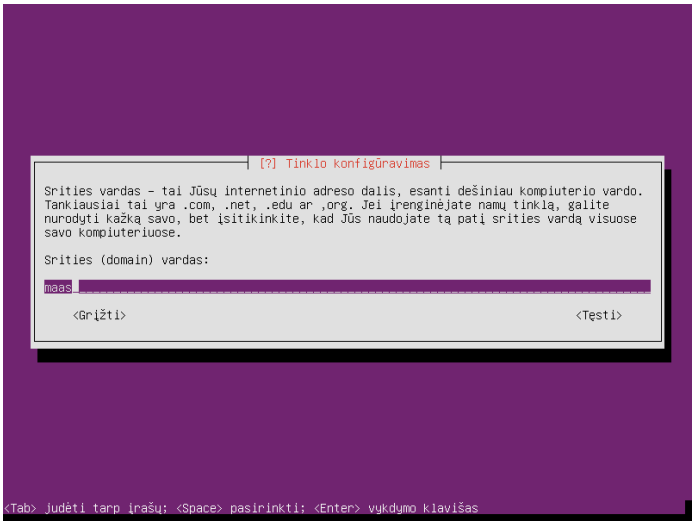


Pav. 16. Diegimo vedlio kalbos pasirinkimas

Pasirinkus vedlio kalbą, pasirodo tolimesnių veiksmų pasirinkimo sąrašas. Iš šio sąrašo reikėtų pasirinkti *Daugelio serverių įdiegimas su MAAS* punktą. Šis punktas pridės *OpenStack* debesies platformai reikalingas bibliotekas bei MaaS administravimo aplinką. Dalis įrankių debesies konfigūravimui ir palaikymui bus įdiegta iškarto, tačiau keletą teks įdiegti papildomai.

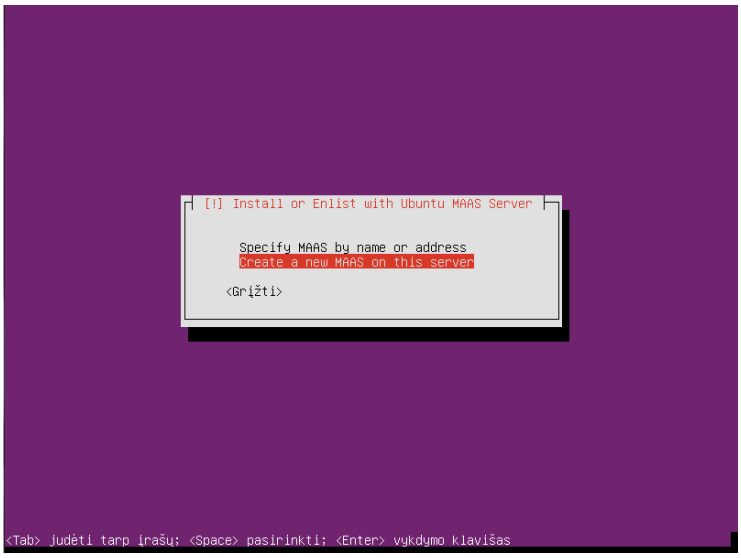
Pav. 17. Diegimo vedlio pasirinkimo galimybės

Sekantys langai susiję su gyvenamosios vietos nustatymu ir kompiuterio klaviatūros išdėstymo pasirinkimu. Po to reikės parinkti kompiuterio vardą. Pasirodžius tinklo konfigūracijos langui įveskite domeno vardą, kuris bus naudojamas vietiniame tinkle.



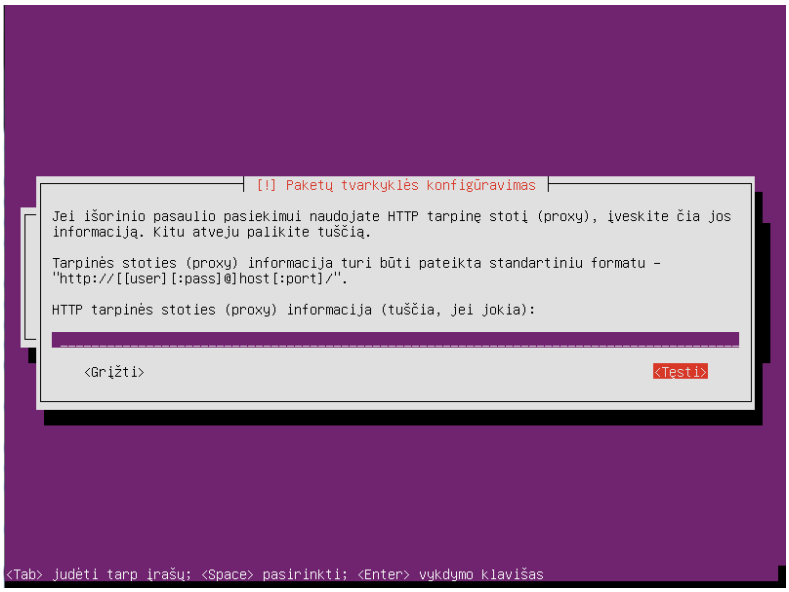
Pav. 18. Diegimo vedlio pasirinkimo galimybės

Sekančiame žingsnyje reikia pasirinkti ar diegiamas serveris yra valdantysis MaaS serveris ar rankiniu būdu diegiamas darbinis mazgas. Šiuo atveju reikėtų pasirinkti *Create a new MAAS on this server*.



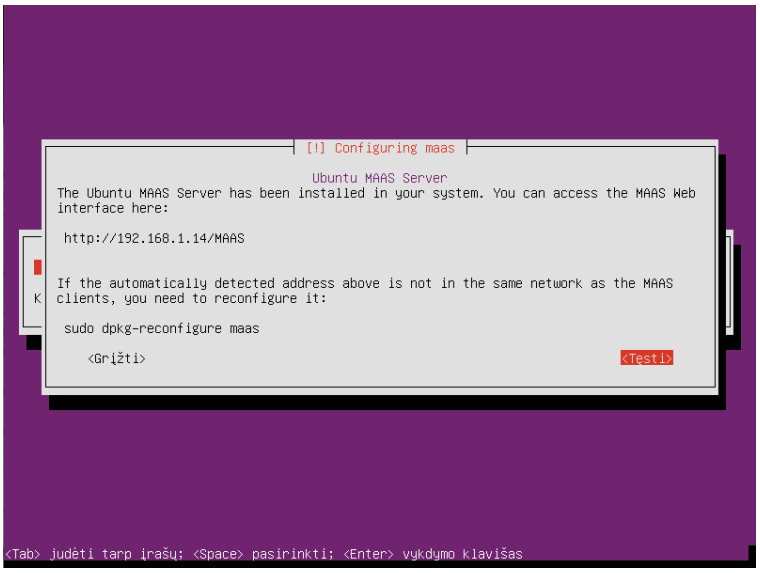
Pav. 19. Serverio tipo pasirinkimas

Sekantis vedlio žingsnis, tai vartotojo sukūrimas. Pageidautina naudoti slaptažodį, kuris gali būti ir nesudėtingas. Po to pasirodo laiko juostos patvirtinimo dialogas ir kietojo disko konfigūravimo žingsniai. Pasirodžius dialoginiam langui su HTTP tarpinės stoties informaciją, palikite laukelį tuščią.



Pav. 20. HTTP tarpinės stoties informacijos nustatymas

Toliau nustatomas serverio atnaujinimo tipas, pasiūlomas kalbų paketo diegimas. Sekantis žingsnis labai svarbus, jame pateikiamas adresas, kuriuo bus galima prisijungti prie debesies fizinių bei virtualių mazgų administravimo aplinkos. Pateiktas adresas naudoja HTTP protokolą, todėl prie MaaS administravimo aplinkos bus galima prisijungti naudojant internetinę naršyklę.



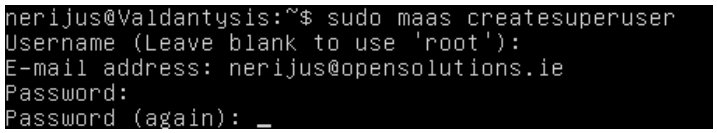
Pav. 21. MaaS administravimo adresas

Paskutinis vedlio dialoginis langas leis pasirinkti ar įdiegti GRUB. Po šios operacijos diegimo vedlys primins išimti kompaktinę plokštelę ir sistema pasileis iš naujo. Persikrovus sistemai reikės įvesti prisijungimo vardą ir slaptažodį, kad galėtumėte naudoti komandinę eilutę. Prisijungimo vardas ir slaptažodis tokie kokie buvo nurodyti serverio diegimo metu. Prisijungus prie terminalo reikėtų įvykdyti komandas *sudo apt-get update* ir *sudo apt-get upgrade*, šios komandos atnaujins

prieinamų programų sąrašus bei atnaujins sistemą. MaaS serverio nustatymai gali būti pakoreguoti iškvietus komandą *sudo dpkg-reconfigure maas*. SSH serverio paslauga bus įdiegta sistemos diegimo metu, todėl iškarto galima prisijungti prie serverio naudojant *ssh* komanda. Prisijungimo vardas ir slaptažodis naudojami tokie pat kaip ir prisijungimo prie terminalo metu.

2.2.1. MaaS serverio konfigūravimas ir papildomų paslaugų diegimas

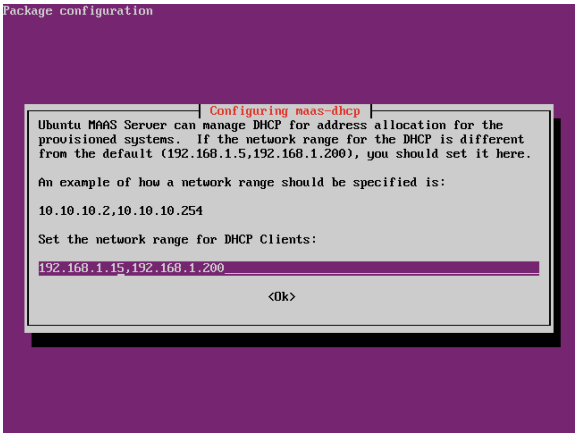
Įdiegus MaaS serverį reikia susikurti MaaS administratorių, kurio paskirtis administruoti debesies fizines bei virtualias mašinas. Administratoriaus sukūrimui, komandinėje eilutėje, reikia įrašyti komandą: *sudo maas createsuperuser*. Toliau vadovaujantis pasirodžiusiomis komandomis pateikti reikiamą informaciją (žr. Pav. 22).



Pav. 22. MaaS administratoriaus sukūrimas

Toliau reikia parsisūsti rinkmenas reikalingas automatiniam mazgų įdiegimui. Parsiunčiamos rinkmenos, iš tikro yra *Ubuntu* kompaktinių plokštelių atspindžiai skirti tiek trisdešimt dviejų tiek šešiasdešimt keturių bitų architektūros mašinoms. Reikalingos rinkmenos parsiunčiamos iškvietus komandą sudo *maas-import-isos*.

Parsiuntus reikiamas rinkmenas mazgų įdiegimui, reikia įdiegti DHCP (Dynamic Host Configuration Protocol) paslaugą, kad mazgai galėtų prisijungti prie MaaS serverio ir automatiškai įsidiegti operacinę sistema, reikiamas paslaugas ir RSA raktus. DHCP paslaugai įdiegti terminale reikia iškviesti komanda *sudo apt-get install -y maas-dhcp*. Diegimo metu pasirodys keletas dialoginių langų. Pirmajame reikės nurodyti IP adresų intervalą, kuriame bus aptinkami darbiniai mazgai (žr. pav. 23). Antrajame lange reikia nurodyti tinklo išėjimo vartus (gateway). Jeigu ši informacija nežinoma, prieš pradedant DHCP diegimą, reikia iškviesti komanda *netstat –rn* ir reikiamą IP adresą galima rasti *gateway* skiltyje. Paskutiniame dialogo lange informacija susijusi su papildomu domeno vardu, skirtu darbiniams mazgams jį galima palikti tuščią.

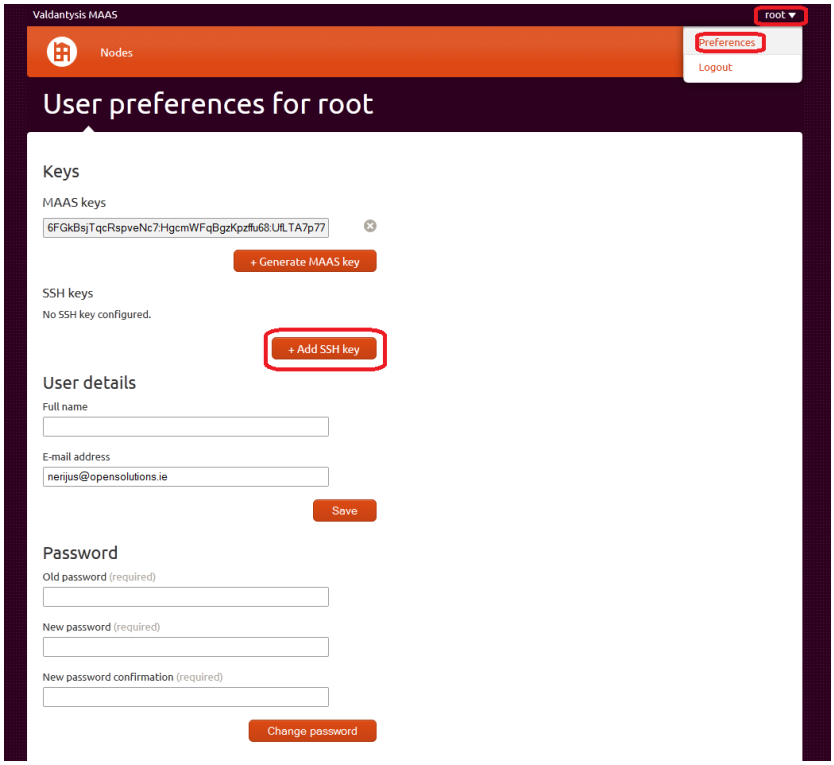


Pav. 23. DHCP IP adresų intervalas

Jeigu diegimo metu DHCP paslauga buvo sukonfigūruota netinkamai, šį dialogą galima iškviesti pakartotinai pasinaudojus komanda: *sudo dpkg-reconfigure maas-dhcp*.

Po to reikia sugeneruoti RSA raktų poras. Reikėtų sugeneruoti dvi raktų poras vieną naudojant esamą vartotoją kitą su administratoriaus teisėmis. Komanda *ssh-keygen* sugeneruos raktų porą skirtą prisijungusiam vartotojui. Pirma reikia nurodyti rinkmenos pavadinimą kuriame bus išsaugotas raktas, šioje vietoje reikia nieko nerašyti ir palikti rinkmenos pavadinimą pagal nutylėjimą. Paskui reikės įvesti slaptažodį ir jį pakartoti. Sukūrus pirmąją raktų porą reikia sukurti ir antrąją, tačiau naudojant komandą *sudo ssh-keygen*. Rinkmenos pavadinimą vėl reikėtų pasirinkti pagal nutylėjimą.

Pridėjus raktus reikėtų naudojant internetinę naršyklę prisijungti prie MaaS serverio. Reikės prisijungimo vardo ir slaptažodžio, kurie buvo įvesti kuriant MaaS administratorių. Prisijungus prie portalo dešinės pusės viršutiniame kampe, matomas prisijungimo vardas ir mažas trikampėlis. Reikia pažymėti prisijungimo vardą ir išsiskleidusiame meniu pasirinkti *Preferences*. Pasirodžiusioje formoje reikia pažymėti *Add SSH key*. Šioje vietoje reikia įkelti anksčiau sukurtų raktų porų viešuosius raktus. Pirma nukopijuokite */home/$vartotojas/.ssh/id_rsa.pub*, kur *$vartotojas* yra prisijungusio vartotojos vardas, rinkmenos turinį į įvesties laukelį ir paspausti + *Add key*. Pridėjus pirmąjį raktą reikia pridėti ir antrąjį, vėl paspaudus *Add SHH key* ir į pasirodžiusį laukelį šį kartą įkelti */root/.ssh/id_rsa.pub*. Rakto frazės bus reikalingos kiekvieną kartą, kai *JUJU* bandys prisijungti prie debesies aplinkos. Kai bus kviečiama *juju ...* komanda reikės naudoti pirmąjį raktą. Naudojant komanda *sudo juju ...* arba jeigu vartotojas tuo metu yra prisijungęs kaip administratorius (root) tuomet reikia naudoti antrojo rakto frazę.

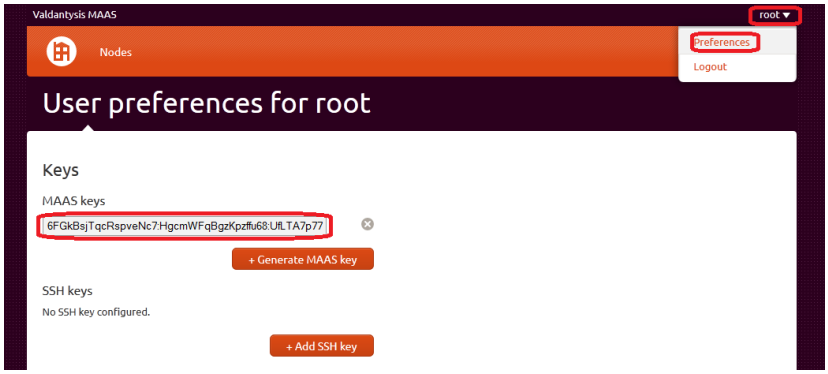


Pav. 24. Raktų įkėlimas į MaaS administravimo aplinką.

Sekančiame žingsnyje reikėtų įdiegti *JUJU Orchestration* įrankius debesies paslaugų administravimui. Norint įdiegti *JUJU* reikia iškviesti komandą: *sudo apt-get install -y juju*. Pasibaigus diegimo procesui reikia sukonfigūruoti MaaS aplinką, prie kurios bus prijungiami darbiniai mazgai. Aplinkos konfigūravimui reikia sukurti rinkmeną aplanke *~/.juju*, pavadinimu *environments.yaml*. Jos viduje turėtų būti išdėstytos konfigūracijos, pavyzdžiui:

```
juju: environments
environments:
  maas:
    type: maas
    maas-server: 'http://localhost:80/MAAS'
    maas-oauth: 'api-key'
    admin-secret: 'nothing'
    default-series: precise
```

Šį pavyzdį galima naudoti tokį, koks jis yra tačiau *api-key* reikėtų pakeisti MaaS raktu, kuris pateiktas *MaaS* administravimo aplinkoje, toje pačioje formoje kaip ir *SSH* raktai.



Pav. 25. API rakto vieta.

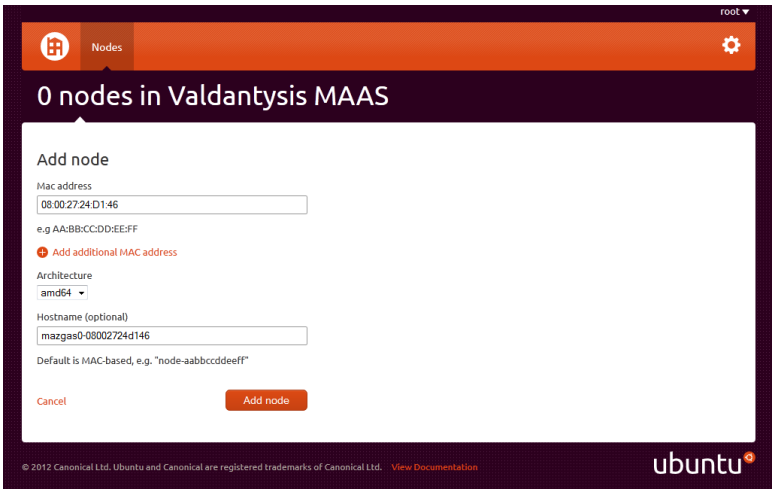
Paruošus *MaaS* aplinką reikia įdiegti *NFS* serverinę paslaugą, kad OMPSS-Server paslauga galėtų būti pilnai išnaudota. Šiam tikslui reikia iškviesti komandą *sudo apt-get -y install nfs-kernel-server*. Įdiegus šią paslaugą reikia sukurti aplanką, kuris bus pateiktas bendram naudojimui. Aplankas turi būti pasiekiamas tokiu pat keliu kaip ir pateikta šiame pavyzdyje. Pirma reikia sukurti patį aplanką pasinaudojus komandomis *sudo mkdir /srv/nfs4* ir iš kart po to *sudo mkdir /srv/nfs4/ompss*. Kadangi aplankas bus bendro naudojimo ir kitos mašinos atliks pakeitimus aplanko viduje, todėl jam reikia sukurti visiškai laisvas prieigos teises, pasinaudojus komanda *sudo chmod 777 /srv/nfs4/ompss*. Sukūrus aplanką, reikia nurodyti sistemai, kad jis bus bendro naudojimo. Tai galima įgyvendinti pakoreguojant rinkmeną */etc/export* ir pridedant eilutę „*/srv/nfs4/ompss *(rw,async,insecure,no_subtree_check)*". Atlikus pakeitimus, paslaugą reikia paleisti iš naujo pasinaudojant komanda: *sudo service nfs-kernel-server restart*. Ir ją aktyvuoti iškvietus komandą: *sudo exportfs -ra*.

2.2.2. Debesies aplinkos konfigūravimas

Norint prijungti darbinius mazgus prie debesies visų pirma reikia pakoreguoti bazinę įvesties ir išvesties sistemos (Basic Input / Output System – *BIOS*) nustatymus ir pakeisti sistemos pradinius paleisties įrenginius taip, kad sistema pradėtų darbą naudojant tinklo įrenginį. Taip pat reikėtų įgalinti pabudimo nuo tinklo aktyvumo (Wake On LAN – WOL) funkciją, tokiu atveju *MaaS* valdiklis pats galėtų įjungti mazgą atsiradus poreikiui jį naudoti. Jeigu WOL funkcija nėra palaikoma mazguose, tai nėra didelė kliūtis tik jį reikės paleisti rankiniu būdu. Toliau bus aprašomas atvejis jeigu *WOL* paslauga mašinose nėra įgalinta.

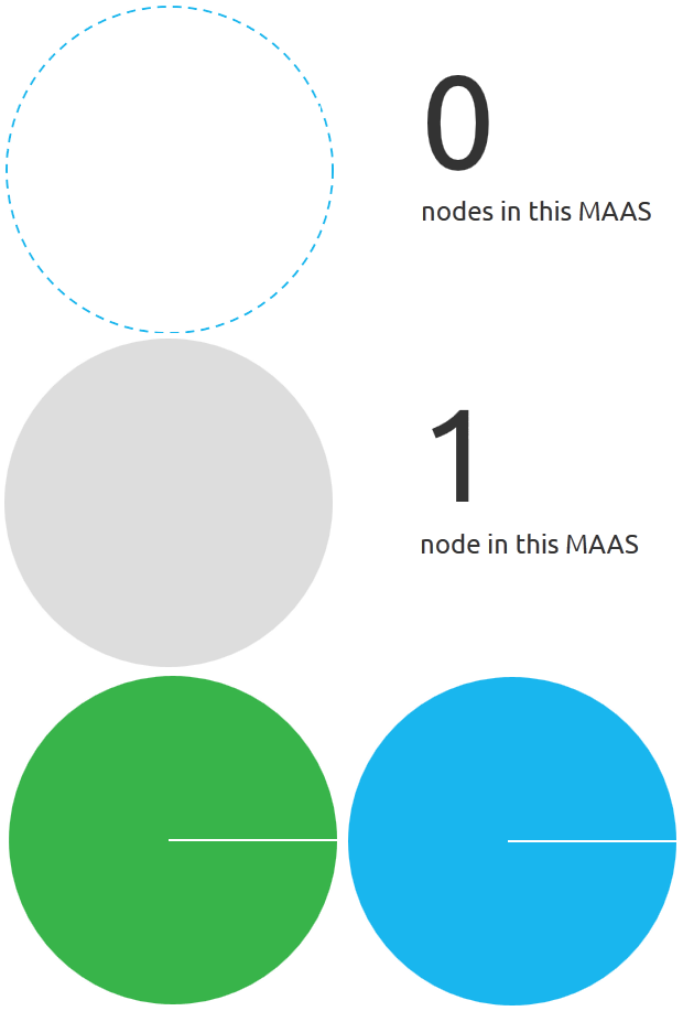
Prieš pradedant kelti paslaugas į debesį, pirmiausia reikia pasirūpinti jog aplinka, kuri buvo sukonfigūruota ankščiau būtų pasiekiama. Aplinkos palaikymui bus pilnai išnaudojamas vienas mazgas, todėl reikia prijungti pirmąją mašiną. Norint prijungti darbinį mazgą prie sistemos reikia žinoti jo MAC adresą, tada prisijungus prie MaaS administracinės aplinkos *Nodes* skiltyje reikia

paspausti + *Add node*. Pasirodžiusioje formoje reikia įvesti mazgo *MAC* adresą, pavadinimą, nurodyti architektūros tipą (žr. Pav. 26).



Pav. 26. *Mazgų pridėjimo forma*.

Pridėjus naują mazgą pradinis MaaS administravimo puslapis pasikeičia. Jame atvaizduojama prie debesies prijungtų mašinų skaičius ir jų būsena. Kol mazgų nėra, pagrindiniame administravimo puslapyje matomas tuščiaviduris apskritimas. Pridėjus naują mašiną, mazgų skaičius perskaičiuojamas ir apskritimas užsipildo pilka spalva, be to naujai pridėto mazgo būsena yra – ruošiamas naudojimui (Commissioning). Jeigu mazgų yra daugiau, tuomet apskritimas sudalinamas į tiek dalių, kad kiekvienas mazgas atitinka tam tikrą procentinę apskritimo dalį. Pilka spalva reiškia jog mazgas buvo pridėtas prie sistemos, bet dar nė karto nebandė prisijungti prie MaaS aplinkos. Kai mazgas prisijungia pirmą kartą, MaaS valdiklis paruošią sistemą mazgo įdiegimui. Tuo metu pagrindiniame administravimo puslapyje, šio mazgo atitikmuo iš pilkos spalvos pasikeičia į žalią, įgydamas būseną pasiruošęs (ready). Mazgas yra iškart išjungiamas pačios sistemos. Išsijungus mazgui MaaS serverio komandinėje eilutėje reikia įrašyti komanda: *juju bootstrap*. Ši komanda aplinkos konfigūracijas perkels į darbinį mazgą jam vėl įsijungus. Labai svarbu atliekant šią komanda turėti bent vieną darbinį mazgą su būsena arba ruošiamas naudojimui arba pasiruošęs. Mazgo saugojančio aplinkos konfigūracija paskirtis yra kontroliuoti paslaugų talpinimą debesyje ir naujų mazgų paskirstymą. Jeigu ši komanda gražina klaidą kurios numeris 409 tai reiškia, kad MaaS administracinėje aplinkoje nėra nei vieno laisvo mazgo. O klaida su numeriu 22 reiškia, kad aplinkos mazgas nepasiekiamas, tai yra išjungtas arba neteisingai sukonfigūruotas. Jei komanda įvykdyta sėkmingai reikia įjungti prieš tai debesiui priskirtą mašiną. Tuo metu į mazgą bus įdiegta operacinė sistema ir visa kita reikalinga informacija. Kai paslauga patalpinta mazgo spalva apskritime tampa mėlyna jo būsena pasikeičia: į priskirtas vartotojui (allocated to user) (žr. Pav. 27).



Pav. 27. Pagrindinių MaaS administracinės aplinkos diagramų pavyzdžiai

2.2.3. Darbinių mazgų prijungimas ir paslaugų talpinimas

Pirmiausia reikėtų parsiųsti paslaugų aprašymus į valdantįjį, MaaS, serverį, tam reikia įdiegti GIT sistemą. Tą galima padaryti pasinaudojus komanda: *sudo apt-get -y install git*. Įdiegus *GIT* sistemą reikia sukurti aplanką, kuriame bus patalpinti paslaugų aprašymai. Aplankas turėtų būti pasiekiamas pasinaudojus keliu *~/src/juju/charms/precise*. Sukūrus reikiama aplanką reikėtų pereiti į jį pasinaudojant komanda *cd ~src/juju/charms/precise*. Po to reikia iškviesti komandą *git clone https://github.com/nereliz/OMPSS-Service.git ompss* ir taip pat komandą *git clone https://github.com/nereliz/OMPSSC-Service.git ompssc*, kurios parsiunčia paslaugų aprašymus.

Parsiuntus paslaugas jas reikėtų truputį pakoreguoti, kad būtų suderinamos su aplinka, į kurią jos bus diegiamos. Pirmiausia reikėtų pakoreguoti OMPSS-Service, pakeičiant NFS paslaugos serverio IP adresą *~src/juju/charms/precise/ompss/hooks/install* rinkmenoje. Taip pat reikėtų patikrinti ar OMPSSC-Service scenarijuose *~src/juju/charms/precise/ompssc/hooks/ompss-relation-joined* ir *~src/juju/charms/precise/ompssc/hooks/ompss-relation*-departed mazgų vidiniai domeno vardai

apdorojami teisingai. Kaip yra sudaromi darbinių mazgų domenai galima patikrinti pasinaudoję komanda *juju-status* ir išvestame atsakyme paieškoti eilutės *public-address*. Tai reikia padaryti tik po to kai paslauga bus patalpinta į debesį, bet prieš sukuriant ryšį tarp paslaugų.

Parsiuntus ir Pakoregavus paslaugas, jas reikia perkelti į debesies aplinką. Šiam tikslui reikia atverti aplanką *~src/juju/charms* pasinaudojus komanda *cd ~src/juju/charms*, ir po to iškviesti komandas: *juju deploy --num-units N --repository . local:precise/ompss* ir *juju deploy --repository . local:precise/ompssc*, kur N norimas darbinių mazgų skaičius. Šiuo metu reikėtų atlikti reikiamus pakeitimus OMPSSC-Service paslaugos ankščiau minėtuose scenarijuose. Jei reikalingi pakeitimai buvo atlikti, tada iškvietus komandą *juju upgrade-charm --repository . ompssc* pakeitimai bus atnaujinti debesies aplinkoje, tai yra darbiniame mazge, kuriame buvo įdiegta OMPSSC-Controller paslauga.

Paslaugos jau patalpintos debesies aplinkoje, tačiau dar nėra sujungtos tarpusavyje ir prieinamos iš išorės. Paslaugas sujungiamos tarpusavyje reikia iškvietus komandą: *juju add-relation ompss ompssc*. Ir pasinaudojus komanda *juju expose ompssc*, OMPSS-Controller paslauga tampa prieinama iš išorės.

Paviešinus paslaugą, internetinio matematinio programavimo ir modeliavimo sistemą reikia pakoreguoti taip, kad ji kreiptųsi į OMPSS-Controller. Tam reikia pakeisti rinkmenų */www/math/controller/aprogram.php* 122 eilutėje ir */www/math/controller/execute.php* 41 eilutėje esančius *IP* adresus į mazgo, turinčio OMPSS-Controller paslaugą, *IP* adresą. Jei šio mazgo IP adresas nėra žinomas tuomet reikėtų į jį prisijungti ir patikrinti. Su komanda *juju status* galima sužinoti į kurį mazgą (unit) buvo įdiegta OMPSS-Controller paslauga, tada pasinaudojus komanda *juju ssh N*, kur N mazgo numeris, prisijungti prie mazgo. Komanda *ifconfig* suteiks informacijos apie mazgo tinklo nustatymus, jeigu OMPSS ir debesis skirtinguose potinkliuose reikia žinoti tinklo nustatymus ir juos pakoreguoti, kad išoriniai kompiuteriai galėtų pasiekti OMPSS-Controller mazgą naudojantis 4645 prievadu.

2.3. Papildomi punktai

Ši skiltis nėra būtina debesies eksploatavimui, tačiau gali būti naudinga tolimesniems sukurtų paslaugų priežiūros ir tobulinimo procesams.

2.3.1. NFS Kliento diegimas

Esant poreikiui peržiūrėti sukompiliuotas programas, programų kodus ar patalpintus bendrus OMPSS-Server paslaugos nustatymus, nenaudojant valdančiojo mazgo, reikia prisijungti MaaS serveryje sukonfigūruotą bendro naudojimo aplanką. Pirma įdiegiama NFS kliento sistemą *sudo apt-get -y install nfs-common*. Įdiegus naują sistemą reikia sukurti aplanką kuris bus naudojamas,

kaip vartai į bendro naudojimo aplanką, pavyzdžiui *sudo mkidr /srv/ompss*. Sukūrus aplanką reikia pakeisti jo prieinamumo nustatymus, kad pakeitimus galėtų daryti bet kas, nes sistemoje šis aplankas bus matomas kaip vietinis, o aplanko viduje pakitimus darys darbiniai mazgai. Pakeisti nustatymus galima pasinaudojant šia komanda: *sudo chmod 777 /srv/ompss*. Paruošus aplanką reikia iškviesti komandą *sudo mount $IP:/srv/nsf4/ompss /srv/ompss*, kur *$IP* NFS serverio IP adresas. Tuomet bendro naudojimo failai bus prieinami naudojant kelią */srv/ompss*. Tačiau perkrovus sistemą NFS klientas nebesujungia vietinio aplanko su bendro naudojimo aplanku nutolusiame serveryje. Todėl reikia pakoreguoti */etc/fstab* rinkmeną pridedant eilutę: „*$IP:/srv/nfs4/ompss /srv/ompss nfs auto 0 0“*, kur *$IP* NFS serverio IP adresas.

2.3.2. Prisijungimas prie mazgų

Tobulinant ir keičiant paslaugos diegimo scenarijus gali pasitaikyti klaidų ir kritinės klaidos atveju, paslauga nėra įdiegiama. Paslaugos statusą galima pamatyti iškvietus komandą: *juju status*. prie konkrečios paslaugos (service) sekcijos, mazgo (unit) dalyje, agento statuso (agent-status) parametras. Jei paslauga buvo įdiegta sėkmingai statusas įgauna reikšmę: prasidėjęs (started). Jeigu mazgas išjungtas ar nepasiekiamas tada statuso reikšmė atjungtas (down). Statusas diegimo klaida (install-error) pasako apie diegimo scenarijuje esančią kirtinę klaida, dėl kurios scenarijus nebuvo įvykdytas iki galo. Tokiu atveju reikėtų pašalinti paslaugą pasinaudojus komanda *juju destroy-service $vardas*, kur vardas paslaugos pavadinimas, pavyzdžiui *ompss*. Tuomet rikėtų įjungti paslaugos testavimo režimą, pasinaudojus *komanda juju debug-hooks ompss*. Ir vėl patalpinti paslaugą atgal į debesį.

Kai paslauga vėl patalpinta reikia iškviesti *juju status* komandą, kad nustatyti į kurį mazgą buvo bandoma įdiegti paslaugą. Tuomet iškviesti komandą *juju ssh $id*, kur *$id* mazgo numeris. Šios komandos metu kitaip nei vykdant kitas *JUJU Orchestration* komandas RSA rakto frazę reikės pakartoti du kartus. Pirmą kartą prisijungiant prie debesies aplinkos ir antrą kartą prisijungiant prie mazgo. Ir pirmą ir antrą kartą rakto frazė ta pati kaip ir naudojama prisijungiant prie debesies aplinkos. Prisijungus prie darbinio mazgo reikėtų atsidaryti aplanką */var/lib/juju/units*. Tada iškvietus komandą *ls* bus pateiktas paslaugų esančių mazge aplankų sąrašas. Paslaugų aplankų vardai susideda iš paslaugos vardo ir identifikatoriaus, pavyzdžiui, paslauga ompss gali būti patalpinta aplanke ompss-12. Ta pati paslauga gali turėti kelis aplankus, tačiau reikėtų kreipti dėmesį į tą kurio identifikatorius didžiausias. Aplanke yra rinkmena *charm.log* ir *charm* aplankas. *Charm* aplanke yra paslaugos aprašymas ir diegimo scenarijai. O *charm.log* rinkmenoje saugomi visi įrašai kurie buvo atlikti diegiant paslaugą, paleidžiant paslaugą bei atliekant paslaugų

tarpusavio ryšių veiksmus. Taigi peržiūrint šiuos įrašus galima pakankamai greitai nustatyti kodėl diegimo ar kito proceso metu įvyko klaida ir paslaugos agentas negali pradėti darbo.