KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Martynas Vaidelys

UŽDUOČIŲ EFEKTYVAUS SKIRSTYMO MODELIAVIMAS HIBRIDINIO CLOUD APLINKOJE

Magistro darbas

Vadovas
doc. dr. K. Šutienė

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil. dr. V.Pekarskas
2013 06 07

UŽDUOČIŲ EFEKTYVAUS SKIRSTYMO MODELIAVIMAS HIBRIDINIO CLOUD APLINKOJE

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
doc. dr. K. Šutienė
2013 06 07

Recenzentas
doc. dr. N. Listopadskis
2013 06 07

Atliko
FMMM-1 gr. stud.
M. Vaidelys
2013 06 07

KAUNAS, 2013

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Rimantas Rudzkis, profesorius (VU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., direktoriaus pavaduotojas (UAB „Danet Baltic“)

Vaidelys M. Užduočių efektyvaus skirstymo modeliavimas hibridinio cloud aplinkoje : Master's work in Applied Mathematics / supervisor dr. assoc. prof. Šutienė K.; Department of Mathematical Research, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2013. – 61 p.

SUMMARY

The providers of Hybrid Cloud based platforms must solve the challenge of workload scheduling, meanwhile satisfying Service Level Agreements. Overview of scientific publications shows that costs for running idle resources increases significantly if scheduling is done irresponsibly. Motivated by this problem, task scheduling theory were adapted to maximize private Cloud utilization. Main property which lets any schedule customizations is task's priority or maximum postpone time. In this work scheduling also involves task's run time, required virtual machine number and from hardware side – private Cloud capacity. In conjunction with scheduling comes restrictions like preventing high priority tasks from being sent to public Cloud. This leads to investigation of the workload data characteristics while restriction management refers to prediction model that generates future values of resource occupation or incoming workflow. To establish optimal schedule according to mentioned restrictions, objective function is needed which minimizes processor's time rented from public Cloud.

Research showed that fitting any standard mathematical distribution like Pareto, Weibull or Gama to characteristics of the real-life workload data is a complicated task. Real-life data is frequently stochastic or modal and does not fit theoretical models, therefore empirical distributions are the best choice to use. In terms of workflow, the autocorrelation analysis confirmed a weekly seasonal pattern with decreasing traffic at weekends and similar tendency in workdays. However, despite seasonal component, more accurate workflow prediction using neural networks, does not prove to be very realistic.

Lastly, hybrid Cloud model was created, in which particle swarm optimization were used to schedule tasks while taking into account basic workload characteristics, scheduling restrictions and resource occupation prediction. After comparing this type of scheduling with FiFo and "short tasks first" strategies it did show some increase in private Cloud utilization. However the best results there achieved after additionally adding tasks pre-execution. This means that no matter how good the schedule is, not all tasks fits into private Cloud. Consequently, these excessive tasks could be sent to public Cloud right away. Overall, task scheduling strategies implemented in this work allowed to increase private Cloud utilization and most of all – shortened an average tasks waiting time.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Vystant lygiagretų programavimą, skaičiavimus GRID tinkle, resursų virtualizaciją ir siekiant sistemos lankstumo išsivystė naujas kompiuterinis modelis – skaičiavimai Cloud aplinkoje *(angl. Cloud Computing)*. Cloud pasiekia aukštą virtualizacijos ir abstrakcijos lygį efektyviai integruodamas techniniu požiūriu skirtingus skaičiavimo, duomenų saugojimo resursus ir programinę įrangą [8]. Vartotojui norinčiam pasinaudoti šia technologija dažniausiai pakanka tik interneto naršyklės kompiuteryje ar telefone ir galimybės prisijungti prie tinklo. Cloud tikslas dalintis duomenimis, resursais ir paslaugomis tarp vartotojų. Remiantis šaltiniu [37], struktūra Cloud apibrėžiama kaip lygiagreti, paskirstyta sistema, sudaryta iš tarpusavyje sujungtų, virtualių serverių, kurie pateikiami dinamiškai ir kaip vienas ar keli tarpusavyje suderinti skaičiavimo resursai, veikiantys pagal paslaugų sutarties sąlygas *(angl. Service Level Agreements (SLAs))*, suderėtas tarp paslaugos tiekėjo ir pirkėjo. Paslaugos tiekėjas čia yra techninės ir programinės Cloud dalies valdytojas, o vartotojas – paslaugos ir resursų nuomotojas-pirkėjas.

Nuo savo pirmtakų Cloud skiriasi, nes jungia techninės architektūros požiūriu akivaizdžiai skirtingus serverius, jų blokinius *(angl. Clusters)* ir taip sukuria virtualų blokinį. Privalumai: į bendrą sistemą galima integruoti silpus ir galingus resursus, išdėstytus skirtingose pasaulio vietose, be to paslaugos užsakovui galima skirti pageidaujamos galios ir talpos virtualius resursus. Su virtualiais resursais galima atlikti tas pačias lygiagretaus skaičiavimo darbus, kaip ir su fiziniais. Taip pat egzistuoja galimybė virtualizuoti jau virtualius resursus [52]. Kitas Cloud privalumas – jo priežiūra rūpinasi paslaugos tiekėjas. Vartotojui nereikia rūpintis techninės bei programinės įrangos atnaujinimu, diegimu ir konfigūravimu, energijos sąnaudomis, efektyvumo problemomis bei paslaugos saugumu.

Cloud tampa ypač paklausi dėl galimybės sumažinti kaštus, padidinti verslo lankstumą ir paslaugų teikimo nenutrūkstamumą [12]. Įmonių resursų planavimo (ERP), atsargų valdymo sistemos (SCM), vartotojų ryšių valdymo (CRM), medicininės, valstybinės duomenų bazės, žaidimai tinkle, mobiliosios technologijos turi milijonus vartotojų [53]. Aptarnauti vartotojus, ypač pasklidusius po visą pasaulį, reikia didelio interneto pralaidumo, greitų skaičiavimo resursų, talpių duomenų centrų, didelių energijos sąnaudų – didelių kaštų. Bendrai ši technologija leidžia mokėti tik už tuos resursus kurių vartotojui tuo metu reikia *(angl. pay-as-you-go manner)* [53], pavyzdžiui už valandą procesoriaus laiko arba už duomenų saugyklą parai *(angl. processors by the hour, storage by the day)*. Paradoksalu, tačiau tokiu būdu taupant galima žymiai paspartinti skaičiavimų eigą.

Praktiniu pavyzdžiu gali būti žiniasklaidos kompanijos The New York Times sprendimas 2007 metais [43], pateikti visus viešus 1851-1922 metų straipsnius nemokamai, t. y. suteikti viešą priėjimą prie 11 milijonų skenuotų originalių straipsnių paverstų į elektroninį PDF formatą. Straipsnius buvo galima versti iš vaizdų į PDF vartotojui pareikalavus, tačiau nuspręsta konvertuoti visus iš karto ir

vartotojams suteikti statinę prieigą. Kadangi tokia procedūra reikalauja greito ir talpaus duomenų centro arba daug procesoriaus laiko, nuspręsta šią užduotį patikėti Amazon Cloud paslaugų tiekėjui. 4 TB duomenų buvo patalpinti Amazon S3 saugykloje, o 100 Amazon Elastic Cloud (EC2) procesorių atliko konvertavimą per 24 valandas (minima, kad konvertavimas atliktas du kartus). Ši procedūra (saugojimas ir skaičiavimai) turėjo kainuoti apie 890 USD, tačiau atsižvelgus į vienkartinį poreikį – pigiau, nei įsigyti nuosavą alternatyvą.

Šiuo metu vis daugiau paslaugų keliamos į Cloud. Remiantis 2011 metų apklausomis apie 3/4 kompanijų-respondenčių savo planuose Cloud laiko prioritetu ir maždaug tiek pat bent minimaliai naudojasi jo paslaugomis [31, 44]. Ankščiau paskirstyto skaičiavimo privalumų labiausiai reikėjo stambiems skaičiavimams ir tyrimams sparčių skaičiavimo resursų reikalaujančiose *(angl. High Performance Computing (HPC))* bendruomenėse. Dabar vis dažniau į Cloud perkeliamos interneto svetainės ir jų kūrimo, talpinimo (Site2You, doTemplate) paketai, elektroninis paštas, darbastalio programos (Microsoft Web App, Google Docs), vaizdų apdorojimo (Pixlr, Adobe Photoshop), programų kūrimo, darbastalio dalinimo (LogMeIn, Mikogo, TeamViewer), telekonferencijų (Skype, Lync) programos, internetinė televizija ir bankai, failų dalinimosi paslaugos, žaidimai, socialiniai tinklai (Facebook, Google+), internetiniai dienoraščiai (WordPress, Twitter).

Reaguodamos į paklausą, šias paslaugas teikia vis daugiau kompanijų. 2013 metų tiekėjų dešimtukas [50] atrodo taip: Amazon, Rackspace, Microsoft, Google, Salesforce.com, Red hat, VMware, Citrix, Linode ir IBM. Tačiau, naudojantis Nimbus, Eucalyptus, OpenStack, Joyent ar kitais konfigūravimo įrankiais, kuriamos ir naujos Cloud struktūros, ypač skirtos organizacijos vidinėms reikmėms arba tikslinei vartotojų grupei aptarnauti. Kalbant apie realizaciją, techniniu požiūriu Cloud skirstomas pagal modelio tipą ir lokacijos vietą (plačiau 1.1 skyriuje). Modelis apibūdina kokiu pavidalu paslaugos tiekėjas resursus pateikia galutiniam vartotojui: programinės ar techninės įrangos forma. Tuo tarpu lokacijos vieta nusako, kur šie resursai egzistuoja fiziškai: privačios organizacijos viduje ar valdomi vieno iš ankščiau įvardinto viešo paslaugų tiekėjo.

Cloud struktūroje operuojama virtualiais resursais ir nėra tiesioginio priėjimo prie aparatūrinės įrangos, tačiau Cloud pajėgumai nusakomi virtualiomis mašinomis *(angl. Virtual Machine (VM))*, kas suteikia žymiai platesnes galimybes jais disponuoti. VM tai individualaus kompiuterio analogas, su fiksuotos spartos procesoriumi, darbine atmintimi, duomenų saugykla ir I/O galimybėmis. VM yra nedaloma, tačiau paslaugos tiekėjas gali realizuoti įvairių konfigūracijų virtualių mašinų, kurios tarpusavyje gali būti jungiamos į grupes, sukuriant superkompiuterio alternatyvą.

Kadangi VM konfigūravimas yra labai lankstus, atsiranda galimybė apjungti VM ne tik išskirtinai privačiame arba viešame Cloud. Galima lokacijos vieta, papildoma hibridiniu Cloud *(angl. Hybrid Cloud)*, kai techniniai resursai jau nebeturi vienintelės fizinės būvimo vietos. Kokie tokios sistemos privalumai ir kodėl nepakanka išskirtinai viešo Cloud?

1. Hibridinis Cloud suteikia galimybę naudotis teoriškai neribotu virtualių mašinų kiekiu, nes privačius resursus galima greitai išplėsti viešaisiais *(angl. Cloud bursting)*, esant minimaliam sulėtėjimui [2];
2. Panaudojami jau turimi privatūs resursai;
3. Tenkinami duomenų saugumo ir paslaugų nenutrūkstamumo reikalavimai, nes minimaliems organizacijos poreikiams patenkinti visada egzistuoja privatus Cloud;
4. Vis dar galioja visi Cloud struktūros privalumai [35, 48].

Iki šiol aptarta tik techninė Cloud dalį – resursai. Tikslas juos „įdarbinti“ vartotojų poreikiams tenkinti, o konkrečiau – užduočių vykdymui. Tariant, kad virtualios mašinos vienodos, šiame magistriniame darbe užduotis aprašoma dviem pagrindiniais parametrais: VM poreikis × vykdymo laikas. Remiantis šiuo principu apibrėžus ir Cloud struktūrą, užduotis būtų interpretuojama kaip dalis viso prieinamo Cloud ploto. Tačiau, priešingai nei VM, vartotojų užduotys nėra fiksuoto dydžio, chaotiškai pasiskirstę laike, ir kaip matysime vėliau, statistiškai sunkiai aprašomos. Iš to kyla literatūroje plačiai nagrinėjamos problemos:

- privataus Cloud resursų trūkumas, esant per dideliam ateinančių užduočių srautui [11, 27];
- privataus Cloud nepilnas panaudojimas, esant užduočių trūkumui;
- nuolatinės Cloud apkrovos užtikrinimas *(angl. Cloud balancing)* [33];
- užduočių planavimas *(angl. Scheduling)* privačiame Cloud [35, 48], viešame ir tarp jų [16, 23];
- užduočių vykdymas nuomojamuose viešuose Cloud, esant privačių trūkumui *(angl. Cloud bursting)* [17, 27];
- laiko nuostoliai ir duomenų perdavimo kaina, susidaranti dėl užduočių pateikimo ir aptarnavimo viešame Cloud [19, 40];
- energijos nuostoliai, susidarantys dėl nenaudojamų resursų [20, 21];
- užduočių kiekio, vykdymo laiko ir kitų charakteristikų prognozavimas [18, 39, 45];
- užduočių kiekio prognozavimas, rezervavimas [28];
- užduočių kiekio intensyvumo kitimo ir pikų valdymas [18, 28];
- resursų nepilnas panaudojamumas dėl virtualių mašinų diskretiškumo [46];
- skirtingi eilių modeliai [46];
- kokybės *(angl. Quality of Service (QoS))* užtikrinimas galutiniam vartotojui [28].

Visi paminėti šaltiniai teigia, kad galima pasiekti didesnį efektyvumą taikant tikslines priemones ir išreiškiamas poreikis didinti hibridinio Cloud ir Cloud, bendru atveju, efektyvumą.

Naudojant hibridinį Cloud, bendru atveju, turimos dvi dalys: vieša ir privati. Koks šių dalių santykis privačiai organizacijai būtų finansiškai naudingiausias? Atsakymas – kad privatūs resursai būtų maksimaliai išnaudoti. Tačiau tai labai priklauso nuo teikiamos paslaugos specifikos ir šis santykis gali

labai varijuoti. Iš to kyla tikslas įvesti priemones padidinančias nuosavų VM panaudojimą (arba sumažinančias užduočių vykdymo viešame Cloud poreikį), kurios lengvai adaptuotųsi nepriklausomai nuo paslaugų specifikos. Tam reikia:

- Ištirti užduočių charakteristikas ir ištirti jų prognozavimo, statistinio įvertinimo galimybes;
- Parinkti užduočių planavimo metodus ir atlikti jiems reikalingus papildomus skaičiavimus;
- Sukurti imitacinį hibridinio Cloud modelį;
- Patvirtinti planavimo metodų efektyvumą eksperimentais.

Šiame darbe apžvelgtos problemos kylančios naudojant realų hibridinį Cloud ir nagrinėjama galimybė šią struktūrą modeliuoti. 1 skyriuje pateikiami techniniai Cloud aspektai, Cloud užduočių charakteristikų aprašymo metodai, užduočių planavimo strategijos, tvarkaraščių sudarymo teorija ir alternatyvūs publikuojami problemos sprendimai. 2 skyriuje aprašyta darbo tiriamoji dalis. Realioms užduotims aprašyti pritaikomi matematiniai metodai, sprendžiamas užduočių srauto prognozavimo sezoniškumu ir neuroniniais tinklais klausimas ir tiriamos tvarkaraščių sudarymo galimybės skirtingais euristiniais metodais. 3 skyriuje pateikiamas hibridinio Cloud imitacinio modelio aprašymas, įgyvendintų užduočių planavimo strategijų algoritmai. Rezultatų ir diskusijos skyriuose pateikiami ir apibendrinami gauti rezultatus ir aptariamos problemos kilusios atliekant hibridinio Cloud užduočių planavimą.

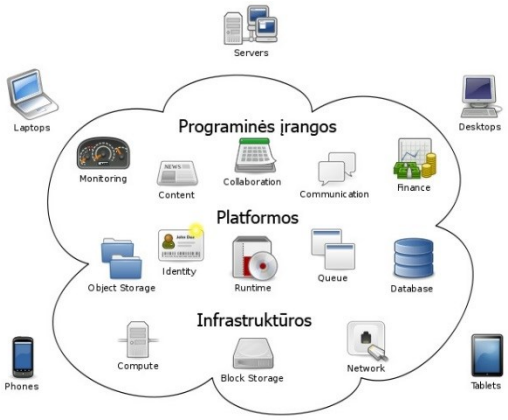
1. TEORINĖ DALIS

Įvade supažindinta su Cloud idėja ir nauda, iškeltos literatūroje nagrinėjamos problemos, bei magistrinio darbo tikslai. Šioje dalyje smulkiau aprašoma:

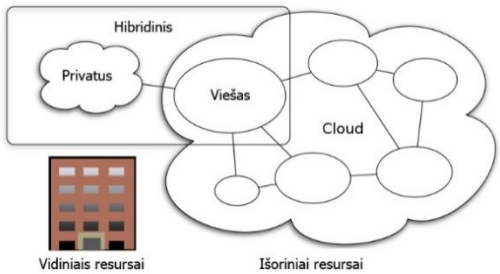
- Cloud architektūra ir tipai, terminai naudojami Cloud kontekste ir šiame darbe;
- Užduočių charakteristikų tyrimui naudotini matematiniai metodai;
- Užduočių ir jų srauto prognozavimui skirti metodai;
- Matematiniai metodai tinkami užduočių valdymo skirstymo modeliui sudaryti;
- Cloud modeliavimui tinkama ir šiame darbe naudojama taikomoji programinė įranga.

1.1. CLOUD KLASIFIKACIJA

Cloud tiekėjai siūlo paslaugas paremtas vienu iš trijų modelių [7, 35, 48] (Pav. 1.1): infrastuktūros *(angl. Infrastructure as a Service (IaaS))*, platformos *(angl. Platform as a Service (PaaS))* arba programinės įrangos *(angl. Software as a Service (SaaS))*. Serveriai veikiantys infrastruktūros modelio pagrindu, vartotojui suteikia galimybę naudotis tinklais, duomenų saugyklomis, bazine programine įranga, todėl ši paslauga dar vadinama technine *(angl. Hardware as a Service (HaaS))*. Platformos arba programinės įrangos modelio pagrindu teikiamos paslaugos yra orientuotos į konkrečių užduočių ar darbų vykdymą, atitinkamai programų kūrimui ir duomenų apdorojimui.



1.1 pav. Cloud modeliai IaaS, PaaS ir SaaS



1.2 pav. Cloud tipai: privatus, viešas ir hibridinis

Pagal savo lokacijos vietą (Pav. 1.2) [7] Cloud gali būti privatus *(angl. Private)*, viešas *(angl. Public)*, hibridinis *(angl. Hybrid)* ir bendruomenės *(angl. Community)*. Toliau pateikti aprašymai ir pritaikymo pavyzdžiai būdingi kiekvienam iš tipų:

- Privatus Cloud skirtas naudoti išskirtinai tik kompanijos ribose ir jos specifiniams poreikiams. Gali būti kuriamas ir valdomas kompanijos viduje arba privataus Cloud tiekėjo.
- Viešas Cloud. Programos, talpyklos ir kiti resursai yra pasiekiami internetu išorinių vartotojų, o sistemos aptarnavimas ir valdymas tiekėjo rankose. Standartiškai tokia paslauga būna nemokama arba apmokestinami tik resursai kurie tuo metu naudojami. Jei privatų Cloud laikysime intranetu, tai viešas atitikmuo – internetas. Pagrindinis viešo Cloud minusas – privatumo trūkumas.
- Bendruomenės Cloud infrastruktūra tarpusavyje dalinasi kelios organizacijos priklausančios tai pačiai bendruomenei, t. y. apibrėžtos tokios pat saugumo, naudojimo, teisinės ir panašios nuostatos. Kaštai paskirstomi mažesniam kiekiui rinkos dalyvių nei esant viešam Cloud, taigi realizuojamas tik dalinis kaštų taupymo potencialas.
- Hibridinis Cloud apjungia privatų ir viešą Cloud. Organizacija gali nuspręsti dalį savo resursų valdyti viduje, o dalį išorėje. Toks pasirinkimas padidina sistemos atsparumą neplanuotiems gedimams ir vartotojų pertekliui, sumažina priklausomybę nuo interneto, tačiau išlaiko visus viešo Cloud privalumus.

  Šiame magistriniame darbe aptariamas IaaS hibridinio Cloud tipas.

1.2. UŽDUOČIŲ CHARAKTERISTIKOS

Realiose IaaS sistemose, remiantis paslaugų tiekėjais Microsoft Azure arba Amazon, virtualios mašinos charakterizuojamos:

- procesoriaus sparta ir procesorių skaičiumi,
- spartinančiosios atminties kiekiu,
- saugyklos talpa,
- tinklu/internetu perduodamų duomenų kiekis,
- papildomai teikiamomis paslaugomis ir galimybėmis, tokiomis kaip SQL serveris, grafikos procesorių naudojimas lygiagretiems skaičiavimams,
- kitos paminėtų charakteristikų variacijos, kurios gali būti būtinos užduoties vykdymui arba vartotojui.

Apibendrinant, išvardintos charakteristikos įkainojamos pavieniui arba pagal iš anksto nustatytą modelį, todėl šaltiniuose [1, 34, 46] įvardijamos kaip užduotį aprašančios charakteristikos.

Aprašant užduotis papildomai įtraukiami ir su SLA sutartimi susiję parametrai. Straipsniuose [34, 38], bei viešai prieinamuose užduočių žurnaluose [41] įvardijami tokie parametrai:

- užduoties atvykimo į sistemą laikas (patekimo į sistemą, bet nebūtinai vykdymo pradžios laikas),
- prognozuojamas užduoties vykdymo laikas,
- vėliausias užduoties įvykdymo laikas *(angl. deadline)*,

- ar užduotis gali būti nutraukiama, atidedama, sustabdoma, perkeliama į kitus resursus,
- ar užduočiai suteikiamas išskirtinis priėjimas prie konkrečių resursų *(angl. exclusive access),* ar resursais dalinamasi su kitomis užduotimis *(angl. shared access)*,
- vartotojų grupė, t. y. užduočių pirmumas ir prioritetas.

Užduotį įvykdžius įvardijamos tokios charakteristikos:

- užduoties vykdymo pradžios,
- užduoties paruošimo vykdymui laiko intervalas,
- laukimo laikas,
- realus panaudotas procesoriaus laikas,
- realus vykdymo laikas,
- eilės tipas ar numeris,
- sutarties reikalavimų tenkinimo sėkmingumas, klaidos kodas.

Šiame darbe užduotis aprašoma dviem pagrindiniais parametrais: VM poreikis × vykdymo laikas. VM laikomos vienodomis, taigi techninės jų galimybės įtakos neturi. Kitos esminės darbe naudojamos charakteristikos: užduočių atvykimo į sistemą laiko pasiskirstymas, užduočių vykdymo laiko prognozavimas, užduočių atidėjimas ir visi su įvykdymu susiję rezultatai.

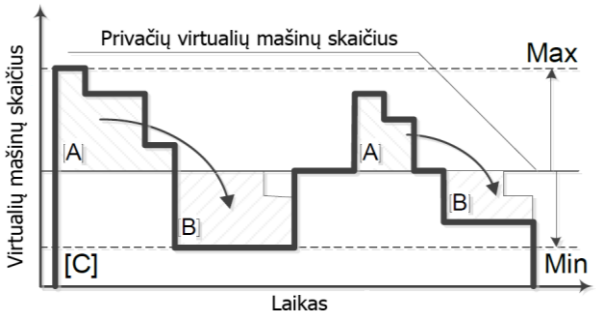
1.3. UŽDUOČIŲ PLANAVIMAS

Kai susiduriama su paskirstytais skaičiavimais, šiuo atveju užduočių vykdymu hibridinio Cloud aplinkoje, visada keliamas planavimo klausimas [54]. Vieni iš paprasčiausiu planavimo būdų – užduočių rikiavimas pagal tam tikrą charakteristiką: atėjimo laiką, dydį ar sudėtingumą – tai priklauso nuo konkrečios situacijos. Sudėtingesnis ir dažnai efektyvesnis būdas – naudotis tvarkaraščių sudarymo teorija. Jei pirmuoju atveju metodams reikia žinoti minimaliai apie užduočių ir sistemos savybes, tai sudarant tvarkaraščius svarbu kuo tiksliau apibūdinti užduotį, žinoti visas sistemos galimybes ir visus galimus ribojimus. Užduotims aprašyti dažniausiai naudojamos statistinės charakteristikos: skirstiniai ir sąryšiai, o planuojant į ateitį – prognozuojamas užduočių srautas. Tačiau net ir tiksliai žinant šią informaciją, nebus galima nieko pakeisti vykdymo plane, jei užduotys bus nepajudinamos laike, pavyzdžiui, užduotys privalo būti vykdomos iš karto patekusios į sistemą.

Vienas iš būtų padaryti planavimą lankstesnį, įvesti galimybę užduotis atidėti iki tam tikro laiko momento [38]. Tai panašu į planavimą esant užduoties pabaigos laiko ribojimui (angl. Deadline constrained scheduling) [18], kai užduotis privalo būti įvykdyta iki tam tikro laiko, tačiau šiuo atveju nesirūpinama apie užduoties pabaigą. Išskiriami keli atidėjimo planai, vadinami prioritetais: aukšto prioriteto vartotojų užduotys – vykdomos neatidedant, o žemo – atidedant. Varijuojant skirtingais atidėjimų laikais, galima pasiekti žymiai didesnį privačios hibridinio Cloud dalies apkrovimą, nei tiesiog pateikiant užduotis iš eilės. Idėja remiasi tuo, kad užduočių srautas pasižymi sezoniškumu, todėl kartais

yra užduočių perteklius, kartais – trūkumas, t. y. privataus Cloud prastovos.

Užduočių atidėjimas demonstruojamas 1.3 paveiksle, kur A žymi užduočių perteklių, B – užduočių trūkumą, o C žymi suplanuotas užduotis sistemoje. Tobulu atveju, kai maksimalus atidėjimas pakankamai didelis, o A dalis lygi B, srautą turėtų būti galima išlyginti ir išvengti visiško viešo Cloud panaudojimo. Realiu atveju, viešo Cloud panaudojimas gali būti bet sumažinamas.



1.3 pav. Užduočių atidėjimas

Tyrimo [38] metu, buvo ieškoma optimalaus privačių resursų kiekio, t. y. techninę dalį bandoma pritaikyti prie esamo užduočių srauto. Tačiau srautas kinta ir naudojamas privatus Cloud dažniausiai fiksuotas. Tokiu atveju, A dalyje gali būti gerokai daugiau užduočių, nei B dalis geba aptarnauti, kurias iš užduočių atidėti? Čia gali būti pritaikyta tvarkaraščių sudarymo teorija. Tačiau kaip minėta, susidūrus su tvarkaraščiais, svarbūs tampa apribojimai ir kuo tikslesnis užduočių charakteristikų žinojimas. Todėl toliau apžvelgiama teorija užduočių charakteristikoms nustatyti ir užduočių srauto prognozei sudaryti.

1.4. UŽDUOČIŲ CHARAKTERISTIKŲ NUSTATYMAS

Kiekvienas srautas gali būti aprašomas tam tikromis charakteristikomis (žr. skyrių 1.2). Atliekant statistinį sistemos imitavimą, laikomasi prielaidos, kad charakteristikos gali būti gaunamos naudojantis skirstiniais. Tinkamai parinkti skirstiniai leistų sukurti dirbtinį užduočių srautą, atkartojantį tikrąjį srautą, ir tai leistų korektiškai modeliuoti realią sistemą, bei atlikti matematinę analizę. Cloud sistemos užduočių parametrai dažniausiai pasižymi tokiomis savybėmis:

- Reikšmės teigiamos. Neigiamos procesoriaus spartos būti negali.
- Skirstiniai asimetriški: yra labai daug mažų reikšmių ir kelios didelės.

Šiame skyriuje aprašomi pagrindiniai matematiniai skirstiniai naudojami pritaikant užduočių srautus [54] pasižyminčius minėtomis savybėmis.

1.4.1. EMPIRINIS SKIRSTINYS

Lengviausias, visada galimas, būdas yra nesudarinėti matematinio modelio skirstiniui aprašyti, o naudoti realius duomenis tiesiogiai. Tai vadinama empiriniu skirstiniu. Empirinis skirstinys labai patogus, kai duomenims pritaikyti netinkami dažniausiai naudojami matematiniai skirstiniai. Pavyzdžiui, Cloud sistemose nusakant užduočių dydį arba perduodamų duomenų kiekį.

Empirinis skirstinys vienareikšmiškai aprašomas realių duomenų histograma. Atsitiktinių dydžių generavimas taip pat nesudėtingas: parenkama atsitiktinė reikšmė iš tolygaus intervalo $[0, 1]$, pagal kurią iš histogramos parenkama generuotina reikšmė. Tiksliau tai aprašoma šia procedūra:

1. Sudaroma histograma h. Tai reikšmių vektorius atitinkantis kiekvieną galimą duomenų reikšmę x (arba reikšmę pasirinktu žingsniu). $h(x)$ – nusako kiek kartų konkreti reikšmė pasikartojo duomenų žurnale.
2. Histograma paverčiama į pasiskirtymo formos funkciją. Tam sudaromas vektorius c, kurio reikšmės yra iš intervalo $[0, 1]$, o n – galimų reikšmių skaičius.

$$c(x) = \frac{\sum_{x' \le x} h(x')}{\sum_{x'} h(x')} = \frac{|\{x' | x' \le x\}|}{n}. \quad \textbf{(1.1)}$$

3. Parenkamas atsitiktinis skaičius u iš tolygaus intervalo $[0, 1]$.
4. Randamas mažiausias skaičius x toks, kad $c(x) \ge u$. Rastasis x ir yra atsitiktinė reikšmė sugeneruota iš empirinio skirstinio.

Aprašyta procedūra yra pritaikyta tik diskrečių reikšmių generavimui, kaip pavyzdžiui užduoties virtualių mašinų skaičius. Jei norima generuoti tolydžias reikšmes (užduoties vykdymo laiką) arba reikšmių trūksta, reiktų interpoliuoti tarp x reikšmių. Empirinis skirstinys puikiai tinka aprašyti ir modalinių reikšmių pasiskirstymą (virtualių mašinų skaičių, kuris nusakomas pvz. 2^k).

Naudoti šį skirstinį modeliuojant gali būti nekorektiška, nes rezultatai atspindi tik konkrečius duomenis. Be to empiriniais skirstiniai negali būti naudojami matematinėje analizėje.

1.4.2. GAMA SKIRSTINYS

Gama skirstinio tankio funkcija užrašoma (1.2), o Gama funkcija išreiškiama (1.3). Čia α – formos parametras, β – mastelio parametras.

$$f(x|\alpha, \beta) = \frac{1}{\beta \Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}, \quad \textbf{(1.2)}$$

$$\Gamma(\alpha) = \int_o^\infty x^{\alpha-1} e^{-x} dx. \quad \textbf{(1.3)}$$

Vienas iš pagrindinių Gama skirstinio privalumų yra jo lankstumas. Skirtingi parametrai α ir β leidžia išgauti įvairią skirstinio formą, panaudotiną dirbant su Cloud užduočių srautu. Skirstinio reikšmės teigiamos ir neribojamos, be to kai $\alpha > 1$ egzistuoja moda $\neq 0$, kas leidžia išgauti ne tik monotoniškai mažėjančios formos skirstinį.

1.4.3. VEIBULO SKIRSTINYS

Veibulo skirstinio tankio funkcija užrašoma (1.4), kur α – formos parametras, β – mastelio parametras.

$$f(x|\alpha,\beta) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-\left(\frac{x}{\beta}\right)^{\alpha}}, \quad \textbf{(1.4)}$$

Kai α didelės – moda ryškesnė $\neq 0$ ir labiau simetrinė. Kai α mažos – išryškinamos didelės skirtinio reikšmės. Veibulo skirstiniu šaltinyje [16] aprašomi užduočių vykdymo laikai.

1.4.4. PARETO SKIRSTINYS

Remiantis [25] vykdymo laiko pasiskirstymas turėtų būti aprašomas Pareto skirstiniu. Matlab pakete realizuota apibendrinto Pareto skirstinio tankio funkcija (1.5), kuri virsta Pareto, kai $\alpha > 0$ ir $\theta = \frac{\beta}{\alpha}$, kur $\alpha \neq 0$ – formos parametras, β – mastelio parametras, θ – ribinis parametras.

$$y = f(x|\alpha,\beta,\theta) = \frac{1}{\beta}\left(1 + \alpha\frac{x-\theta}{\beta}\right)^{-1-\frac{1}{\alpha}}, \quad \textbf{(1.5)}$$

Kadangi Pareto skirstinys begalinis, tam tikrais atvejais naudojama sutrumpinta jo forma, kai apibrėžiama maksimali galima reikšmė t. Tokiu atveju būtina normalizuoti skirstinį. Pareto skirstinio moda visada lygi 0.

1.4.5. KITI SKIRSTINIAI. SKIRSTINIŲ PRITAIKYMAS

Priklausomai nuo konkrečios sistemos, jų užduočių charakteristikos negali būti aprašomos vienareikšmiškai. Pateikti skirstiniai yra dažniausiai sutinkami arba jau pagrįsti, kaip Pareto – teorijoje dažnai įvardijamas kaip užduočių vykdymo laiką nusakantis skirstinys. Tačiau lygiagrečių sistemų modeliavime taikomi ir kiti skirstiniai [54]: normalusis, log-normalusis, eksponentinis, log-gamma, Erlango, bei skirstinių mišiniai, kurie šiame darbe netirti. Plačiau skirstiniai nenagrinėjami, nes naudojamos programinės priemonės turi realizuotus skirstinius, geba parinkti jų parametrus mažiausių kvadratų metodu ir generuoti atsitiktines reikšmes.

Skirstinio taikymo korektiškumas gali būti patikrinamas tikslumo testais. Šiame darbe naudojamas Kolmogorovo-Smirnovo testas, o vizualizacijai kvantinių grafikas *(angl. Q-Q plots)*, kurie lygina ar dviejų duomenų sekų X ir Y skirstiniai tarpusavyje sutampa. Lyginimas atliekamas tarp realių duomenų X ir atsitiktinių reikšmių sugeneruotų po skirstinio pritaikymo Y. Kolmogorovo-Smirnovo testo metu tikrinamos hipotezės:

H_0: X ir Y yra pasiskirstę pagal tą patį skirstinį,

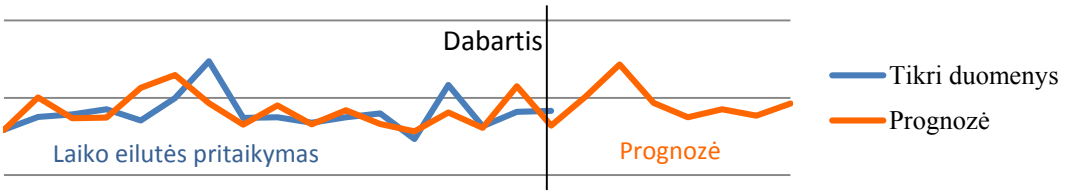
H_1: X ir Y yra sugeneruoti skirtingais skirstinių.

Kvantilių grafikas rodo, kad skirstiniai sutampa kai, reikšmės išsidėsto tiesėje.

1.5. UŽDUOČIŲ SRAUTO PROGNOZAVIMAS

Literatūroje sutinkami du terminai, kalbant apie laiko eilučių prognozę: pritaikymas *(angl. Fitting)* ir prognozavimas *(angl. Forecasting, Prediction, Extrapolation)*. Nors jie yra susiję, tačiau ne visada

naudojami vienareikšmiškai, todėl būtina pabrėžti skirtumą: pritaikymas – tai žinomos laiko eilutės atkartojimas pasirinktu metodu, prognozavimas – žinomos laiko eilutės ekstrapoliavimas į ateitį (Pav. 1.4). Šie terminai dažnai apjungiami ir vadinami tiesiog prognoze-prognozavimu. Bendru sutarimu, toliau prognoze vadinamos tik nežinomos reikšmės ateityje.



1.4 pav. Pritaikymo ir prognozavimo terminų skirtumas

Nepaisant terminologijos skirtumų, abejais tikslais, konkrečiai laiko eilutei, naudojamas tas pats modelis, t. y. prognozuojama naudojantis pritaikymo modeliu. Pažymėjus laiko eilutės reikšmės kiekvienu laiko momentu ξ_t, o prognozę – $\hat{\xi}_t$, pritaikymo-prognozavimo modelis $i'jam$ žingsniui į priekį užrašomas (1.6) išraiška. Kai prognozės horizontas $n > 1$, tada tolimesnei prognozei dažnai naudojamos ne tik praeities, tačiau ir prognozuotosios reikšmės. Išraiškoje (1.6) esantis j parametras nurodo, kad ne visada naudojama visa laiko eilutė. Dažniausiai prognozavimas vykdomas remiantis keliomis paskutinėmis praeities reikšmėmis, kurios parenkamos naudojantis reguliarų arba nereguliarų vėlinimą.

$$\hat{\xi}_{t+i} = F(\xi_t, \dots, \xi_j), \qquad kai\ i = \overline{1,n}, \qquad j \in (1,t), \tag{1.6}$$

čia t – dabartis, o F – pritaikymo-prognozavimo modelis, n – prognozės horizontas.

Pritaikymo modelio sudarymui dažnai naudojami klasikiniai prognozavimo metodai: tiesinis prognozavimas, eksponentinio glodinimo, slenkančio vidurkio, ARIMA. Kai turima daug istorinių laiko eilutės reikšmių, o jų pasiskirstymas laike šiek tiek chaotinis, įdarbinami dirbtinio intelekto metodai, pavyzdžiui neuroniniai tinklai arba Fuzzy logika. Dirbtinio intelekto metodai pasižymi gebėjimu išmokti visą laiko eilutę, todėl apmokymo procese naudojamos ne tik paskutinės laiko eilutės reikšmės, tačiau visos reikšmės, sugrupuotos pagal parinktą vėlinimo vektorių. Vėlinimų vektorius nustatomas atsitiktinai arba euristiniai metodais [24].

Pritaikymo modelio ir prognozės tikslumas nustatomas paklaidų įvertinimo metrikomis, kas dažniausiai ir pagrindžia pasirinkto pritaikymo modelio adekvatumą. Dažnai naudojamos tokios paklaidų metrikos:

- Vidutinė kvadratinė paklaida *(angl. Mean Square Error (MSE))* (1.7),
- Šaknis iš vidutinės kvadratinės paklaidos *(angl. Mean Root Square Error (RMSE))*,
- Vidutinė procentinė absoliutinė paklaida *(angl. Mean Absolute Percentage Error (MAPE))* (1.8).

$$MSE = \frac{1}{n}\sum(\xi_t - \hat{\xi}_t)^2 \tag{1.7}$$

$$MAPE = \frac{100}{n}\sum\left|\frac{\xi_t - \hat{\xi}_t}{\xi_t}\right|, \tag{1.8}$$

čia n – prognozuotų reikšmių skaičius. MAPE paklaida negali būti naudojama su lygiomis 0 reikšmėmis, o kai reikšmės artimos 0, paklaidos reikšmė taip pat stipriai iškraipoma.

Kadangi vienas iš svarbiausių prognozavimo tikslų numatyti staigius šuolius (pikus) stebimoje laiko eilute, šiame darbe pristatoma nauja paklaidų vertinimo metrika DE *(angl. Direction Error)*. DE parodo santykį tarp klaidingai prognozuotos krypties reikšmių ir visų prognozuotų reikšmių. Kryptis čia suprantama kaip laiko eilutės reikšmių kilimas arba kritimas. Matematine išraiška DE užrašoma (1.10) forma, o kuo mažesnė DE reikšmė – tuo geresnis rezultatas. Ši metrika taip pat nesunkiai gali būti pertvarkoma vertinti tik pakilimus arba kritimus laiko eilutėje.

$$DE = 1 - \frac{1}{n_p}\sum_{t=2}^{n_p} H\left[\text{sign}(\xi_t - \xi_{t-1}) \cdot \text{sign}\left(\hat{\xi}_t - \hat{\xi}_{t-1}\right)\right], \tag{1.9}$$

čia $H[s] = \begin{cases} 0, & s < 0, \\ 1, & s \geq 0, \end{cases}$ Hevisaido funkcija, o n_p – patikrintų prognozuotų reikšmių skaičius.

1.6. TVARKARAŠČIŲ SUDARYMAS

Vykdant užduotis Cloud aplinkoje susiduriama su vykdymo tvarkaraščio sudarymo problemomis. Paprasčiausiu atveju laikoma, kad užduotis ribotą laiką vykdoma konkrečių parametrų virtualiose mašinose (VM), tačiau realybėje įtraukiamos ir kitos 1.2 skyriuje aptartos užduočių charakteristikos.

Geriausio problemos sprendinio paieška vadinama optimizavimu. Optimizavimo metu, priklausomai nuo situacijos, sprendžiamas minimizavimo arba maksimizavimo uždavinys. Matematiškai, reikia surasti tikslo funkcijos minimumą (neprarandant bendrumo toliau vadinamo globaliu optimumu), įvertinant galimus apribojimus ir uždavinio apibrėžimo sritį. Problema kyla tuomet, kai apibrėžimo sritį sudaro daug parametrų (Cloud techninių parametrų ir užduoties charakteristikų), kurių reikšmių aibė yra baigtinė (tačiau didelė), o tikslo funkcija turi daug lokalių minimumų. Taigi rasti globalaus minimumo tašką standartinėmis optimizavimo priemonėmis (pvz. gradientinio nusileidimo metodu) arba pilno tiesioginio perrinkimo būdu, laiko atžvilgiu, yra nepriimtina.

Formaliai kombinatorinio optimizavimo-minimizavimo uždavinį [14] galima aprašyti struktūra (S, f), kai ieškome tokio objekto, iš baigtinės (arba galimos paversti baigtine) aibės S, kuris minimizuotų tikslo funkciją f. Tikslo funkcijos pobūdis ir išraiška priklauso nuo konkretaus sprendžiamo uždavinio. Labai dažnai kombinatorinių uždavinių tikslo funkcijos yra netiesinės, neiškilos, nediferencijuojamos, daugiaekstremės. Tokiu būdu, išspręsti uždavinį (S, f), reiškia surasti tokį sprendinį $s^* \in S$, kad galiotų (1.10).

$$s^* \in S^* = \left\{\bar{s} \,\middle|\, \bar{s} = \arg\min_{s \in S} f(s)\right\}. \tag{1.10}$$

Sprendinys s^* vadinamas uždavinio (S, f) globaliai optimaliu sprendiniu, o aibė $S^* \subseteq S$ – optimalių sprendinių aibe. Kadangi šiame darbe lyginsime kelis skirtingus problemos sprendimo būdus,

tarsime kad sprendinio reikšmė yra kiekybiškai įvertinama t. y. tikslo funkcijos rezultatą galima išreikšti skaitine forma $f: S \to \mathbb{R}$.

Optimalių tvarkaraščių radimas yra labai aktuali problema įvairiose veiklos srityse: logistikoje, vadyboje, versle, projektavime, ekonomikoje, todėl plačiai ištirta, tačiau rečiau minima Cloud kontekste. Remiantis [22], rasti leistiną tvarkaraščio sprendinį, kuris tenkintų visus ribojimus, klasikiniais metodais, santykinai nesunku. Pavyzdys – kritinių kelių, mažėjančių trukmių ir panašus algoritmai. Tačiau toks leistinas tvarkaraštis nebūtinai bus optimalus. Bendru atveju tai NP sudėtingumo problema (žr. 1.1 lentelę), kurią galima išspręsti pilnu binariniu perrinkimu arba pritaikius metodus, besiremiančius šakų ir ribų metodo idėjomis. Tačiau tokius metodus sunku arba neįmanoma realizuoti, o praktikoje sutinkamiems uždaviniams spręsti reikalauja daug laiko. Todėl tvarkaraščiams optimizuoti vystomi ir taikomi euristiniai optimizavimo metodai, dažnai vadovaujantis analogijomis su gamta, dirbtinio intelekto idėjomis ir pan. Kompleksiniu procesu, apimančiu operacijas su keliomis euristikomis, nagrinėjimas yra priskiriamas metaeuristikos sričiai. Šie procesai yra modifikuojami ir derinami tam, kad būtų gaunami efektyvūs aukštos kokybės sprendiniai.

1.1 lentelė.

Naudojamos kombinatorinio optimizavimo uždavinių sudėtingumo klasės

Žymėjimas	Aprašymas [6]
P	Uždaviniai vadinami polinominio sudėtingumo *(angl. polynomial time computable)*, kai jų sudėtingumas $\mathcal{O}(n^k)$ veiksmų. Čia n – duomenų skaičius, k – algoritmo sudėtingumo eilė. $\mathcal{O}(\alpha^k)$ – eksponentinio sudėtingumo, $\mathcal{O}(n!)$ – faktorialinio sudėtingumo
NP	Nedeterminuoto polinominio sudėtingumo uždaviniams *(angl. nondeterministic polynomial time)* nežinomas polinominio sudėtingumo sprendimo algoritmas, tačiau, atlikus $\mathcal{O}(n^k)$ veiksmų, galima patikrinti ar duotas objektas yra uždavinio sprendinys (pvz.: rikiavimo, keliaujančio pirklio uždaviniai, tvarkaraščio sudarymo [22])

Šiame darbe nagrinėjami diskretieji optimizavimo uždaviniai, kurių sprendiniai yra realūs ir parinkti iš tam tikros diskrečios aibės. Kombinatorinio optimizavimo uždavinių atveju, reikia parinkti geriausią kombinaciją iš galimų. Pagrindinė tokių uždavinių sprendimo problema – reikia palyginti visus be galo daug galimų sprendinių, nes kito sprendinio įvertinimo bendru atveju nėra. Ši procedūra gali užtrukti per ilgai realiems uždaviniams. Naudojantis euristiniais metodais, NP sudėtingumo uždavinys gali būti išsprendžiamas apytiksliai, t. y. perrenkant tik dalį sprendinių. Tokiu atveju žymiai sutrumpėja optimizavimo laikas. Dažnai naudojami euristiniai paieškos metodai kombinatorinio optimizavimo uždaviniams spręsti [22]:

- Modeliuojamojo atkaitinimo *(angl. Simulated Anneling)* algoritmas [30],
- Genetiniai algoritmai *(angl. Genetic Algorithms)* [34],
- Paieška su draudimais *(angl. Tabu Search)*,
- Skruzdžių kolonijos *(angl. Ant Colony Systems)* [10],

- Memtiniai algoritmai *(angl. Memetic Algorithms)*,
- Išbarstytoji, išsklaidytoji paieška *(angl. Scatter Search)*,
- Spiečių algoritmai *(angl. Swarm Algorithms)*,
- Dirbtinių neuroninių tinklų metodai *(angl. Neural Networks)*.

Šie metodai taip pat sutinkami literatūroje užduočių planavimui Cloud aplinkoje (šaltinis pateikiamas prie metodo pavadinimo). Toliau pateikiami šiame darbe naudoti optimizavimo algoritmai ir jų aprašymai.

1.6.1. BENDRASIS KOMBINATORINIO OPTIMIZAVIMO ALGORITMAS

Kitaip vadinamas pilnu perrinkimu [15], kai apskaičiuojamos $f(s)$ su visomis $\forall s \in S$ ir parinkamas toks s^* su kuriuo $f(s^*) \geq f(s), \forall s \in S$.

Teoriškai šis algoritmas išspręstų bet kokią baigtinę optimizavimo problemą. Tačiau daugumai realių problemų aibės S dydis yra per didelis, kad per priimtiną laiką būtų galimą sulaukti šio algoritmo rezultatų.

1.6.2. GENETINIS ALGORITMAS

Genetinių algoritmų *(angl. Genetic algorithm (GA))* veikimas pagrįstas evoliucijos, vykstančios gyvojoje gamtoje, t. y., natūraliosios atrankos proceso imitavimu [42, 49]. Pagrindinės sąvokos, naudojamos modeliuojant biologinės evoliucijos procesus, yra chromosoma ir populiacija. Chromosoma yra objektas sudarytas iš genų. Didesnė ar mažesnė chromosomų grupė sudaro populiaciją. Dar vienas svarbus dalykas yra vadinamasis „chromosomos tinkamumas“ − savotiška chromosomos vertė. Chromosomos vertę galima traktuoti kaip jos sugebėjimą sėkmingai prisitaikyti (prie aplinkos), išlikti ir reprodukuotis. Šia prasme vertingesnis (grynai biologiškai) yra ta chromosoma, kuri sugeba geriausiai prisitaikyti (būti stipresne už kitas) ir, galbūt, palikti didesnį palikuonių skaičių.

Optimizavime vietoje sąvokų populiacija, chromosoma ir jos vertė naudojamos tradicinės, įprastos sąvokos atitinkamai: sprendinių aibė (grupė, rinkinys), sprendinys ir tikslo funkcija. Taip, kaip gyvosios gamtos evoliucijoje išlieka tik vertingiausi individai, taip ir optimizavime siekiama gauti kuo geresnį sprendinį, t. y., sprendinį su kuo mažesne (ar didesne) − nelygu, koks optimizavimo uždavinys (minimizavimo ar maksimizavimo) sprendžiamas − tikslo funkcijos reikšme.

Pateiksime genetinio algoritmo bazinę procedūrą:

1 žingsnis. Atsitiktinai suformuojame pradinę populiaciją $\vec{X}(t), t = 0$, t žymi iteracijos numerį.

2 žingsnis. Apskaičiuojame tikslo funkcijos $f_i, i = 1,2, \dots, N$ reikšmes visai populiacijai $\vec{X}(t)$, N – populiacijos dydis.

3 žingsnis. Pasinaudodami pasirinkta geriausių chromosomų atrinkimo metodika, populiaciją atnaujiname geriausiomis chromosomomis. Skiriami skirtingi genetinių algoritmų tipai pagal geriausių

chromosomų parinkimą [49]. Tinkamo algoritmo tipo parinkimas gali turėti esminės įtakos konvergavimo greičiui ir apskritai konvergavimui.

4 žingsnis. $N-1$ tėvinėms chromosomų poroms $\left(M_1^k, M_2^k\right), k = 1,2, \dots, N-1$ pritaikome vieno atsitiktinio taško kryžminimo operatorių ir gauname naujas chromosomas $X_1'(t+1), \dots, X_{N-1}'(t+1)$.

5 žingsnis. Sukryžmintoms chromosomoms pritaikome mutacijos operatorių.

6 žingsnis. Jei algoritmo stabdymo sąlygos tenkinamos, stop; jei ne – einame į 2 žingsnį.

Iš tikrųjų, terminas „genetiniai algoritmai“ reiškia ne kokį nors atskirą, specifinį euristinį optimizavimo algoritmą, tačiau tam tikrą bendrą, gana universalų metodą, strategiją – metaeuristiką. GA nusako euristinių algoritmų šeimą su panašiais veikimo principais ir savybėmis.

Šaltinyje [29] nurodomas geriausių chromosomų parinkimas tiesiog procentaliai, t. y. nustatytas procentas geriausių chromosomų patenka į kitą iteraciją. Mutacijos gali būti atliekamos vienam genui, arba kažkuriai jų daliai. Kryžminama viename arba keliuose taškuose, dažniausiai taikant papildomas priemones, kaip aritmetinį, euristinį, Laplaso ar kitus kryžminimo operatorius [3, 4]. Šaltinyje [49] sutinkame geriausių chromosomų parinkimą naudojantis parinkimo tikimybe, kai kitai iteracijai parenkamos geresnė chromosoma iš atsitiktinės poros $\left(M_1^k, M_2^k\right)$. Kiekvieno geno mutacija vykdoma su tikimybe $P_m \in [0.001, 0.01]$.

Genetiniai algoritmai taikomi ir Cloud kontekste [34], kur tvarkaraštis sudaromas nepriklausomoms užduotims taikantis prie skirtingų skaičiavimo ir atminties reikalavimų.

1.6.3. DALELIŲ SPIEČIAUS OPTIMIZAVIMAS

Dalelių spiečiaus optimizavimas *(angl. Particle swarm optimization (PSO))* sukurtas remiantis gyvūnų socialiniu elgesiu, tokiu kaip paukščių arba žuvų migracija, būrimųsi [42]. PSO su GA panašūs tuo, kad prasideda iš atsitiktinai sugeneruotos populiacijos, tačiau skiriasi kryžminimo ir mutacijos operatorių nebuvimu. Kadangi algoritmo prigimtis skirtinga, naudojami ir kiti terminai: chromosomos vadinamos dalelėmis. Kiekviena dalelė juda aplink vertės funkcijos hiperpaviršių atitinkamu greičiu. Dalelės atnaujina savo greitį ir poziciją priklausomai nuo geriausio lokalaus ir globalaus sprendinio (1.11):

$$v_{m,n}^{naujas} = v_{m,n}^{senas} + \Gamma_1 \times r_1 \times \left(p_{m,n}^{lokGer} - p_{m,n}^{sena}\right) + \Gamma_2 \times r_2 \times \left(p_{m,n}^{gloGer} - p_{m,n}^{sena}\right), \quad \textbf{(1.11)}$$

čia $lokGer$ – lokalus geriausias, $gloGer$ – globalus geriausias,

Žymėjimai:

- $v_{m,n}$ – dalelės greitis,
- $p_{m,n}$ – dalelės kintamasis, pozicija,
- r_1, r_2 – nepriklausomi, tolygiai pasiskirstę atsitiktiniai skaičiai,
- $\Gamma_1 = \Gamma_2$ – pažintinio ir socialinio mokymosi faktoriai,

- $p_{m,n}^{lokGer}$ – geriausias lokalus sprendinys,
- $p_{m,n}^{gloGer}$ – geriausias globalus sprendinys,
- m – populiacijos dydis,
- n – dalelę sudarančių reikšmių skaičius (GA: genų skaičius).

PSO algoritmas atnaujina greičio vektorių kiekvienai dalelei, tada prideda šį greitį prie dalelės pozicijos. Greičio atnaujinimas įtakojamas tiek lokalaus geriausio, tiek globalaus visos optimizavimo eigos geriausio sprendinio. Jei geriausio lokalaus sprendinio vertės funkcijos reikšmė yra mažesnė (minimizavimo atveju), nei globalaus, tada geriausias globalus sprendinys atnaujinamas lokaliuoju.

Dalelių spiečiaus optimizavimo algoritmo privalumas – lengva realizacija ir mažai keičiamų parametrų, funkcijų, be to šis algoritmas, kaip ir GA, geba optimizuoti funkcijas su daug lokalių minimumų.

1.7. ALTERNATYVŪS MODELIAI

Šiame skyriuje apžvelgiami jau realizuoti Cloud modeliai. Mokslinėse publikacijose aprašomi konkretiems tikslams skirti modeliai [1, 16, 17, 18, 33, 38, 46] ir sutinkama net Cloud sistemos aprašymo kalbų [5]. Darbus galima grupuoti pagal nagrinėjamą sritį:

- užduočių srauto ir jo specifikacijų prognozavimo ir nustatymo;
- tvarkaraščių užduotims arba resursams sudarymo;
- tiriamo Cloud aplinkos tipo – bendru atveju Cloud arba hibridinio Cloud, kai nagrinėjamas užduočių paskirstymas tarp viešo ir privataus Cloud;
- kai apjungiami aukščiau pateikti punktai.

1.7.1. KRŪVIO VALDYMO IR TVARKARAŠČIŲ SUDARYMO MODELIS

Šis hibridinio Cloud modelis [38] sukurtas Kauno technologijos universiteto darbuotojų yra pradinė versija, apimanti tik bendras pradines sąlygas. Užduotys nelygiagretinamos, t. y. viena užduotis vykdoma tik vienoje VM, o pasibaigus užduoties vykdymo terminui, užduotis pašalinama iš sistemos. Užduotys skirstomos į dvi grupes: atidedamas (iki kelių valandų) ir prioritetines (vykdomas tuoj pat pasiekus sistemą). Jei užduotis negali būti įvykdoma nuosavuose resursuose – ji atidedama kiek galima ilgiau. Bendru atveju norima parodyti, kad dėl ateinančių užduočių srauto nepastovumo (bangavimo) atidėjimas leidžia bent dalinai išvengti viešo Cloud.

Eksperimentų tikslo funkcija priklauso nuo vieno parametro – per modeliavimo laikotarpį vykdytų užduočių kaštų. Kaštų funkcija apima dvi dedamąsias: kaštus už vykdymą privačiame ir viešame Cloud. Minimizuojant tikslo funkciją norima rasti nustatytoms sąlygoms adekvatų privačių virtualių mašinų skaičių, garantuojantį mažiausius vartotojų aptarnavimo kaštus.

1.7.2. UŽDUOČIŲ TVARKARAŠČIO SU SUVARŽYMAMS SUDARYMO EURISTIKOS MINIMIZUOJANT KAŠTUS

Ši realizacija aprašyta [16]. Autoriai nagrinėja privačių resursų panaudojimo didinimą, sumažinant kaštus reikalingus užduotims atlikti viešame Cloud, esant hibridiniam Cloud. Atsižvelgiama į paslaugos kokybės užtikrinimo suvaržymus, naudojamos euristikos vertina skaičiavimo, duomenų perdavimo kaštus ir perdavimo laiką. Norima nustatyti kaip skirtingi kaštų svoriai ir užduočių krūvio charakteristikos paveikia kaštus naudojant skirtingas euristikas ir kaip pasikeičia rezultatai esant užduočių vykdymo laiko prognozės paklaidoms.

Pagrindinė modelio sąlyga, kad užduotys, tokios kaip vaizdų ar lygiagreti duomenų analizė, trivialiai skaidytinos į kelis lygiagrečius nepriklausomus darbus ir paleistos nepriklausomose virtualiose mašinose. Paprastumo dėlei tariama, kad viena užduotis negali būti vykdoma lygiagrečiai kelių skirtingų Cloud. Taip pat nagrinėjamas atvejis, kai virtualioms mašinoms lygiagretintos užduoties vykdymo laikas, priklauso nuo konkrečios dalies sudėtingumo. Autoriai atsiriboja nuo užduoties vykdymo laiko, tinklo apkrovimo, duomenų kiekio prognozavimo ir lygiagretinimo efektyvumo problemos.

Dėl realių duomenų trūkumo buvo pasirinkta, kad užduotys ateina kas sekundę pasiskirstę pagal Puasono skirstinį su parametru $\lambda = 0.002$. Tokiu būdu per visą savaitės laikotarpio modeliavimo laikotarpį sukuriama ~ 1200 užduočių. Viena užduotis gali būti suskaidyta į n dalių, kai n pasiskirstęs tolygiai intervale $[1, 100]$, o kiekvienos dalies vykdymo laikai tarpusavyje nepriklauso. Tačiau bendras užduoties vykdymo laikas apibrėžtas Weibull‘o skirstiniu, kurio parametrai išskirti iš Workloads Archive duomenų bazės, nagrinėjant SDSC IBM SP2 apkrovimo duomenis.

Įvertinama papildoma sąlyga: pavyzdžiui dvigubai padidinus resursų galingumą, užduoties vykdymo laikas gali taip stipriai nepasikeisti. Todėl naudojamas Amdahl‘o dėsnis užduoties pagreitėjimui nustatyti. Įvertinami ir papildomi suvaržymai, kaip atminties trūkumo poveikis užduoties vykdymo greičiui, užduoties užimama vieta kietajame diske, užduoties lygiagretinimo laipsnis ir būtino įvykdymo laikas.

Cloud užpildymu rūpinasi trys užduočių planuotojai: privataus, viešo ir hibridinio Cloud:

- Viešas rūpinasi užduočių vykdymu, atsižvelgdamas į vykdymo greitį ir reikalaujamą atminties kiekį, kaštus skirtingus greičius reikalingus užduočiai perduoti tinklu.
- Privatus atsižvelgia į ribotą turimų resursų kiekį;
- Hibridinis Cloud priima sprendimą ar užduoties vykdymas gali būti planuojamas nuosavame ar, įvertinus kaštus, viešame Cloud. Atsižvelgiama gali būti į įvairius faktorius: privataus Cloud užimtumo, laukimo eilėje laiko, biudžeto suvaržymų.

Pateikti privataus ir viešo Cloud darbų planavimo algoritmai atsižvelgiantys į kaštus ir maksimalų užduočių įvykdymo kiekį per ribotą laiką. Pateikiami rezultatai, su skirtingomis planavimo euristikomis,

ateinant skirtingų charakteristikų užduotims, rodo, kad kaštais paremtas modelis atsiperka laiku įvykdytų užduočių kiekiu ir sumažintomis išlaidomis. Tačiau pastebima, kad naudą labai įtakoja užduočių laiko prognozavimo paklaidos.

1.7.3. TVARKARAŠČIŲ SUDARYMAS HIBRIDINIAME CLOUD SU APRIBOJIMAIS NAUDOJANTIS GAUSO PROCESU

Mokslinėje publikacijoje [18] pateikiamo hibridinio Cloud modelio pradinės sąlygos panašios į [38], tačiau tyrimas specializuojasi į užduočių vykdymo laiko įvertinimą naudojantis heteroskedastišku Gauso procesu (HGP). Norint sudaryti patikimą užduočių vykdymo tvarkaraštį reikia kuo tiksliau prognozuoti užduoties vykdymo laiką, kuris nėra žinoma. Kadangi tiksliai įvertinti neįmanoma, reikia tenkintis tam tikru pasikliovimo intervalu, kuris naudojantis HGP procesu anot autorių yra stebėtinai tikslus.

1.7.4. STOCHASTINIS KRŪVIO VALDYMO IR TVARKARAŠČIŲ SUDARYMO CLOUD MODELIS

Šiame modelyje [46] autoriai nagrinėja atvejį, kai naudojama bendrai Cloud ideologija, neįvardijant, kad egzistuoja vieša dalis. Tačiau tiriamas atvejis, kai darbų planuotojui reikia įvertinti skirtingas VM konfigūracijas (procesoriaus galią (CPU), atminties kiekį ir duomenų saugyklos dydį), paskirstyti darbus konkretiems serveriams ir taip rezervuoti reikiamus resursus.

Svarbiu apribojimu laikoma, kad serverio pajėgumai yra riboti, o užduotims reikalingos skirtingos VM. Tada reikia taip paskirstyti virtualias mašinas serveriams, kad jie būtų maksimaliai užpildyti (kad liktų kuo mažiau nepanaudojamų resursų). Pavyzdžiui galima paimti serverį su 10 CPU ir du fiksuotos galios VM tipus: 7 ir 5 CPU; tokiu atveju jei paleidome viena 7 CPU virtualią mašiną, turime 3 nepanaudojamus CPU. Analogiškai galime išplėsti ir daugiau nei vienam serveriui.

1.7.5. IAAS CLOUD APKROVIMO DIDINIMAS

Šaltinio [33] idėja skiriasi nuo ankščiau aprašytų modelių, pirmiausia tuo, kad eksperimentuojama su realia sistema ir realiais uždaviniai, antra – tariama kad yra dalis vartotojų, kurių užduotis galima nutraukti ir pratęsti vėliau. Pabrėžiama, kad norint aptarnauti visus vartotojus, reikalinga turėti pakankamai resursų, kad jų poreikiai būtų patenkinti bet kuriuo metu, atmetant kuo mažiau užklausų. Tokiu atveju vidutinis sistemos apkrovimas būna tik 37.5 %. Pasinaudoję idėja, kad tam tikri, uždaviniai vykdomi nuolat arba ilgai, o rezultato nesitikima gauti „dabar“, autoriai tikina pasiekę 100 % Cloud apkrovimą, taip pat išlaikydami daugumą „paslauga-pareikalavus“ *(angl. on-demand)* vartotojų. Tam išskiriami du paslaugų tipai:

- Paslauga-pareikalavus, nenutraukiama, lanksti nuoma *(angl. on-demand, nonpreemptible,*

fexible leases) – leidžia vartotojui susikurti norimo tipo virtualią mašiną, esamu laiku, iš anksto numatytam laiko intervalui ir atlinkti norimus darbus.

- Paslauga priklausomai nuo situacijos, nutraukiama, momentinė nuoma *(angl. opportunistic, preemptible, preset leases)* – leidžia vartotojui prieiti prie Cloud administratoriaus iš anksto nustatytos specifikacijos resursų, nežinomu momentu, neapibrėžtam laiko intervalui, t. y. tuo metu, kai atsilaisvina resursai ir iki tol kol nėra aukštesnio prioriteto vartotojų. Tokios užduotys gali būti nutraukiamos staiga ir turi būti pritaikytos staigiems resursų kiekio pokyčiams. Tariama, jog resursų kiekis nuolat svyruoja ir pastovaus kiekio tokio tipo užduotims užtikrinti negalima.

1.7.6. NEPRIKLAUSOMŲ CLOUD UŽDUOČIŲ PLANAVIMAS GENETINIAIS ALGORITMAIS

Wuhan universiteto darbuotojai [34] aprašo optimizuotų genetinių algoritmų panaudojimą planuojant nepriklausomas, nenutraukiamas ir lygiagretinamas Cloud užduotis. Techniniais užduočių reikalavimais laikomi procesoriaus sparta ir atminties kiekis, o užduočių prieinamumas prie resursų išskirtinis *(angl. exclusive access)* arba bendras *(angl. shared)*. Tvarkaraščių sudarymo uždavinys išskiriamas į dvi grupes: statinius, kai planuojamos žinomos užduotys, ir dinaminius, kai planavimas remiasi, esama sistemos būsena, bei jau vykdomomis užduotimis. Nors darbas labiau orientuotas į GA realizaciją, tačiau pažymima, kad resursų užimtumas padidėjo.

1.8. PROGRAMINĖ ĮRANGA

Struktūra Cloud nėra visiškai nauja, todėl yra prieinami tiek komerciniai, tiek atviro kodo programiniai paketai skirti realaus Cloud tinklo modeliavimui. Nagrinėjamoms problemoms tirti tiktų CloudSim [13], iCanCloud [32] arba JavaCloudware [1], tačiau norint testuoti naujus metodus, reikia išmanyti šių sistemų architektūrą, į kurią įeina ir daug funkcijų nenaudotinų šiame darbe. Taip pat tiriamų metodų validacijai reikalingas statistiškai reikšmingas pagrindimas, tam būtina atlikti pakankamai eksperimentų. Modelio lengvumas, problemos sprendimo tikslingumas ir galimybė gauti konkrečius duomenis apie esamą ir praėjusią modelio padėtį yra būtini. Remiantis kitų tyrėjų patirtimi [16, 18], paranku kurti tikslinį modelį.

Iš atliktos naudotinų matematinių metodų apžvalgos 4 skyriuje, svarbiausia tampa nesudėtinga matematinių metodų – prognozavimo, tvarkaraščių sudarymo, minimizavimo – realizacija arba galimybė pasinaudoti jau esamomis statistinėmis, funkcijomis. Patogus grafinis rezultatų ir modeliavimo eigos atvaizdavimas, objektinio programavimo galimybės imituojant skirtingus vartotojų tipus ir Cloud komponentus. Sudarant tvarkaraščius atminties poreikis gali smarkiai išaugti, o modeliavimo eigoje bus atliekama daug veiksmų su duomenimis ir matricomis, todėl reikalingas patogus atminties išskyrimas ir

veiksmai su kintamaisiais. Iš šių reikalavimų kyla taikomosios matematinės programinės įrangos poreikis, su integruotais skaitiniais metodais, t. y. aukštesnio lygio programavimo kalba, nei Java, C++ ar panašios.

Poreikius tenkintų komercinis Matlab paketas arba atviro kodo minėtojo paketo analogijos Octave, Freemat, and Scilab. Matlab paketas tikriausiai yra labiausiai naudojamas tarptautiniu mastu matematiniams skaičiavimams ir modeliavimui. Daugelyje universitetų yra pristatomas studentams viena ar kita forma, vertinamas inžinierių ir naudoja labai optimizuotus vidinius algoritmus. Prie bazinio paketo, kuriame jau yra duomenų analizės ir integruotos matematinės funkcijos, galima įsigyti papildomus įrankius duomenų aproksimavimui, funkcijos optimizavimui, neuroninių tinklų ar Fuzzy logikos kūrimui, bei kurti nepriklausomas nuo Matlab paketo programas išoriniam naudojimui *(angl. Application Deployment Toolbox)*. Taip pat integruota galimybė naudotis daug procesorių turinčios techninės įrangos privalumais atliekant lygiagrečius skaičiavimus.

Norėdami atlikti modelio validaciją, turime pasinaudoti realiai surinktais duomenimis. Plačiau apie duomenis kitoje tiriamojo darbo dalyje, paminėsime tik, kad egzistuoja duomenų rinkimui sukurtos atitinkamos priemonės ir standartai [41, 47]. Šis magistrinis darbas neapima duomenų rinkimo ir toliau bus naudojamos mokslinės bendruomenės jau pripažintos duomenų bazės. Pateikiami duomenys yra istoriniai įrašai sistemoje *(angl. Data Logs)* konkrečiais laiko momentais. Kuriamam modeliui tinkamos informacijos išskyrimui bus galima panaudoti MS Excel dėl patogios grafinės sąsajos arba, esant labai dideliam duomenų kiekiui, Matlab priemones. Kita alternatyva naudoti statistinius paketus kaip SAS, SPSS, R, kurie specializuojasi į statistinių metodų taikymą ir charakteristikų išskyrimą.

Imitacinio modelio realizacija savyje turės keletą skirtingų metodų atskiroms užduotims atlikti: prognozavimui, tvarkaraščio sudarymui ar atnaujinimui, o ir pats modelis turės imituoti pakankamai ilgą laikotarpį, kad rezultatai stabilizuotųsi. Skaičiavimų apimtis (laikas) yra vienas iš pagrindinių kriterijų, kai nerandamas optimalus sprendinys.

1.9. APIBENDRINIMAS

Teorijos skyriuje:

- apibūdinta darbe naudojama Cloud struktūra ir techninės kuriamo modelio savybės;
- apibrėžta užduoties sąvoka, jos savybės, charakteristikų nustatymo būdai;
- supažindinta su užduočių srauto prognozavimo teorija, bei užduočių planavimo galimybėmis ir idėjomis;
- pristatyti galimi naudoti tvarkaraščių sudarymo algoritmai;
- trumpai pristatyti alternatyvūs Cloud užduočių planavimo modeliai sutinkami literatūroje.

2. TIRIAMOJI DALIS IR REZULTATAI

Nagrinėjant kompiuterines sistemas reikalinga tyrinėjama kompiuterinė aplinka ir jai pritaikyti duomenys. Dažniausiai domimasi jau turimų arba esamų sistemų, darbo greičiu, ekonominiu efektyvumu, galimu tobulinimu ir galimais rezultatais. Tai galima nagrinėti realiu laiku, su realiais duomenimis ir taip gauti atsakymus į norimus klausimus. Būdas ypač tikslus, tačiau lėtas ir nelankstus. Pirma, proceso negalima pagreitinti, nes viskas vyksta realiu laiku, antra, gaunami rezultatai remiasi dabartimi, t. y. negalima eksperimentuoti norimomis situacijomis, trečia, pavojingi bet kokie neapgalvoti atnaujinimai galintys sukelti milžiniškus nuostolius, paliekant sistemos vartotojus nepatenkintus.

Šiems klausimams spręsti yra naudojami kompiuteriniai modeliai, kurie apsiriboja siaura tyrinėjama sritimi, naudojant supaprastintus duomenis. Naudojant modelius galima imituoti situacijas daug kartų per sąlyginai trumpą laiko tarpą, parinkti įvairias, anomalines situacijas, valdymo arba vykdymo strategijas, kurių realioje sistemoje būtų neįmanoma išbandyti ir atsiriboti nuo nuostolių susijusių su vartotojų patenkinimu.

Vienas iš pagrindinių šio darbo tikslų, yra tolygesnis privataus Cloud apkrovimas užduotimis. Remiantis įvairiomis studijomis, vidutinės sistemos prastovos varijuoja nuo 75 % iki 91 % [54], todėl problema aktuali, o jai spręsti reikia kuo daugiau žinoti apie užduotis, jų statistines charakteristikas ir metodus joms tirti, bei modeliuoti.

2.1. NAUDOTA TECHNINĖ ĮRANGA

Skaičiavimams atlikti buvo naudotas personalinis kompiuteris, kurio parametrai:

- Intel® Core™ i5-2520M Procesorius (2.5 GHz, Max turbo 3.2 GHz, 3 MB Cache);
- Operatyviosios atminties 12 GB – 1333MHz;
- Windows 8 Pro operacinė sistema,
- Matlab R2012b.

Kai kurie eksperimentai, reikalaujantys daug procesoriaus laiko, atlikti Microsoft Azure Cloud aplinkoje išnuomotose virtualiose mašinose. Be to tiriant Cloud aplinką, būtina praktiškai išmėginti Cloud galimybes ir naudojimo principus.

2.2. UŽDUOČIŲ CHARAKTERISTIKŲ RADIMAS

Svarbus etapas kuriant modelį, tiriamosios srities pradiniai duomenys. Cloud yra jauna technologija ir dažniausiai naudojama komerciniais tikslais, todėl duomenų pasiekiamumas dažnai ribotas. Kokybiški, pilnaverčiai pradiniai duomenys yra būtini, nes nuo jų tiesiogiai priklausys gaunami rezultatai. Pavyzdžiui nagrinėjant lygiagrečių užduočių paskirstymą dideliame Cloud reikalingas tikrasis ir vartotojui skirtas maksimalus užduoties vykdymo laikas, naudotų virtualių mašinų skaičius. Priklausomai nuo tiriamos srities gali būti reikalingos ir kitos 1.2 skyriuje paminėtos charakteristikos.

Šiame skyriuje apžvelgiamos užduočių charakteristikas ir sąryšiai.

Tyrimui pasirinkti SDSC, LANL, HPC2N ir LPC istoriniai užduočių įrašai iš [41] (lentelė 2.1). Pavyzdiniu duomenų rinkiniu naudojamas SDSC, nes pasižymi minimaliu išskirčių skaičiumi, didele imtimi ir laikotarpiu, bei akivaizdžiu sezoniškumu, kuris patogus prognozuojant. Kiti duomenys paeiliui vis labiau netvarkingi ir parinkti modelio testavimui esant skirtingoms situacijoms.

Istoriniuose įrašuose buvo kaupiamos ir darbe naudojamos užduoties charakteristikos:

- užduoties patekimo į sistemą laikas,
- užduoties realus vykdymo laikas,
- užduotį pateikusio vartotojo numatytas maksimalus užduoties vykdymo laikas,
- procesorių arba virtualių mašinų skaičius reikalingas užduočiai vykdyti.

2.1 lentelė.

Tiriamų sistemų statistiniai duomenys

Duomenų pavadinimas	Laikotarpis	Užduočių kiekis	Procesorių skaičius
SDSC-BLUE-2000	32 mėnesiai	243314	1152
LANL-CM5-1994	24 mėnesiai	122060	1024
HPC2N-2002	42 mėnesiai	202876	240
LPC-EGEE-2004	9 mėnesiai	234889	140

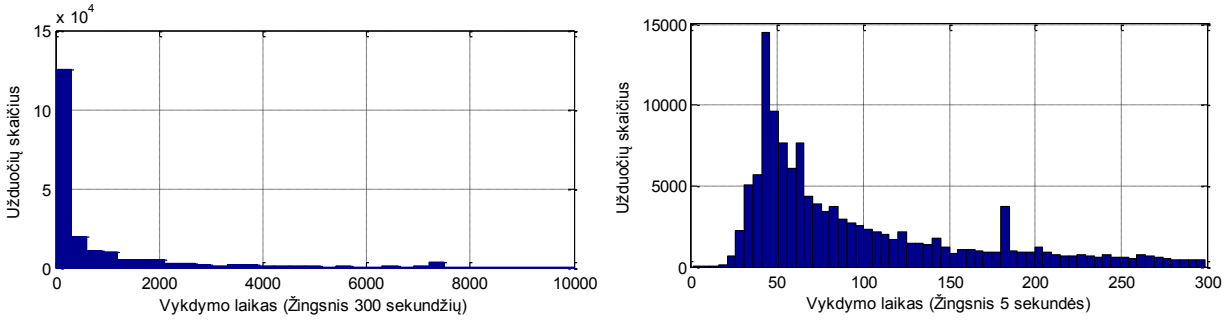
2.2.1. UŽDUOTIES VYKDYMO LAIKO TYRIMAS

Užduoties vykdymo laikas yra viena sunkiausiai nustatomų charakteristikų, nes retai kada tenkina bet kokias teorine prielaidas. Paprasčiausiu atveju galima tarti, kad užduočių vykdymo laikai yra žinomi, kaip elgiamasi [26], tačiau remiantis [39] taip būna retai. Todėl Cloud sistemos ir apskritai kalbant apie užduočių vykdymo laiką kompiuterinėse sistemose, vartotojo prašoma įvertinti galimą jo užduoties vykdymo laiką. Nors įvertis ir bus netikslus, tačiau planavimo prasme suteikia žymiai daugiau galimybių. Taigi užduoties vykdymas aprašomas žinomu įverčių $t_{numatytas}$ ir iki užduoties pabaigos momento nežinomu t_{realus} vykdymo laiku.

Norint korektiškai modeliuoti Cloud sistemą, siekiama užduočių charakteristikas aprašyti skirstiniais. Statistine prasme, šios charakteristikos turėtų būti priklausomos, taigi norint nustatyti jų skirstinius reikia tai įvertinti. Šį ryšį galima realizuoti randant vykdymo laiko įvertinimo paklaidos skirstinį t_{ε}. Tada t_{realus} būtų modeliuojamas pagal savo skirstinį, o įvertis apskaičiuojamas $t_{numatytas} = t_{realus} + t_{\varepsilon}$. Pirmiausia bus nustatomas t_{realus} skirstinys, tada patikrinama ar iš tiesų egzistuoja t_{realus} ir $t_{numatytas}$ ryšys, ir egzistavimo atveju bandomas rasti t_{ε} skirstinys iš kurio galėtų būti generuojamas $t_{numatytas}$.

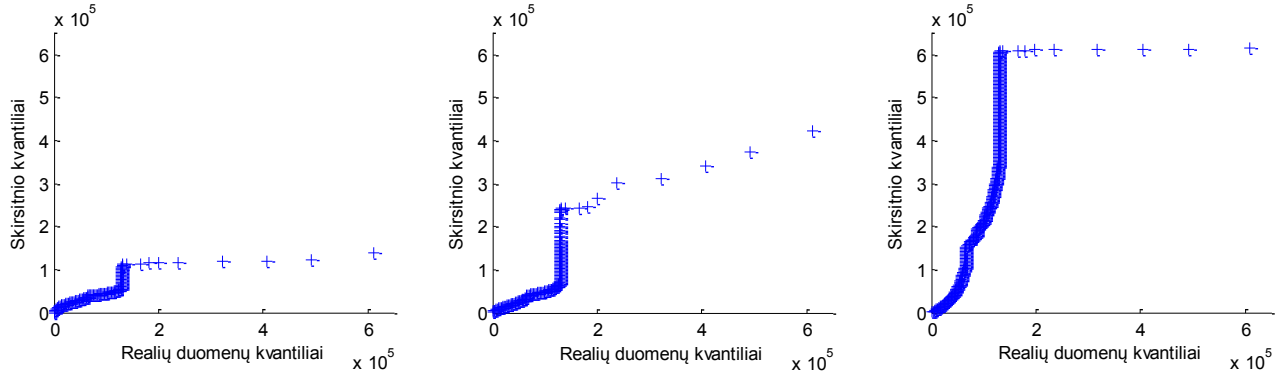
Paveiksle 2.1 pavaizduotos SDSC užduočių vykdymo laiko histogramos skirtingais žingsniais. Didžiausią problema pritaikant skirstinį kelia [54] įvardintos ir paveiksle 2.1(b) matomos savybės: užduočių kurių vykdymo laikas didelis yra mažai, o trumpai vykdomų užduočių skaičius labai didelis.

Realiuose duomenyse taip pat matoma, pasiskirstymo moda nelygi 0. Tokiu atveju teoriškai siūlomas Pareto skirstinys tikrai tiksliai neaprašys trumpų vykdymo laikų. Tokią situaciją aprašyti galėtų Veibulo arba Gama skirstiniai. Matlab priemonėmis randami skirstinių parametrai, sugeneruojami atsitiktiniai dydžiai pagal šiuos skirstinius, patikrinamas atitikimas Kolmogorovo-Smirnovo testu ir grafinei vizualizacijai atidedamas kvantilių grafikas.



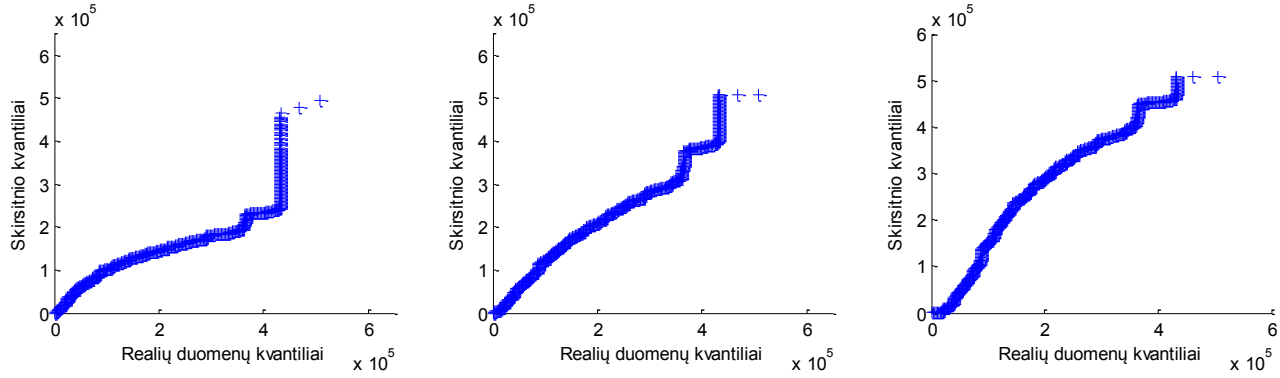
a) b)

2.1 pav. SDSC užduočių realaus vykdymo laiko pasiskirstymas



a) Gama b) Veibulo c) Pareto

2.2 pav. SDSC užduočių vykdymo realaus laiko pritaikymo skirtingais skirstiniais palyginimas kvantilių grafikais



a) Gama b) Veibulo c) Pareto

2.3 pav. HPC2N užduočių vykdymo realaus laiko pritaikymo skirtingais skirstiniais palyginimas kvantilių grafikais

Kolmogorovo-Smirnovo testas visiems (Gama, Veibulo ir Pareto) skirstiniams atmeta H_0 hipotezę su 5 % pasikliovimo lygmeniu. Taigi naudoto testo prasme, nei vienas iš skirstinių negali pakankamai tiksliai atkartoti realių užduočių vykdymo laikų. Iš kvantilių grafikų (pav. 2.2) matyti, kad didžiausia problema kyla ne dėl nelygios nuliui modos, o dėl labai didelio trumpų užduočių skaičiaus ir pakankamai mažo skaičiaus vidutinio ilgio užduočių. Kvantilių grafikas vizualiai artimiausias tiesei yra Gama arba

Pareto. Nagrinėjant 2.1 lentelėje pateiktų kitų realių duomenų užduočių vykdymo laikus, nėra vienintelio geriausiai pritaikomo skirstinio (pvz. pav. 2.3 ir 2.2 lentelę). Kolmogorovo-Smirnovo testas ir kitiems duomenims atmeta H_0 hipotezę su 5 % pasikliovimo lygmeniu. Taigi aprašyti realius duomenis skirstiniu galima su tam tikromis išlygomis, o konkretaus skirstinio pasirinkimas gali būti pagrindžiamas dažniausiai vertintojo nuomone.

2.2 lentelė.

Rekomenduojami vykdymo laiko pritaikymo skirstiniai skirtingiems duomenims

Duomenų pavadinimas	Skirstinio pavadinimas	α – formos parametras	β – mastelio parametras	θ – ribinis parametras
SDSC-BLUE-2000	Gama	0.2899	15135.7111	-
LANL-CM5-1994	Veibulo	1133.8568	0.4671	-
HPC2N-2002	Pareto	4.2068	89.9097	21.3723
LPC-EGEE-2004	Gama	0.1751	18788.1548	-

Ryšiui tarp t_{realus} ir $t_{numatytas}$ įvertinti gali būti naudojama Spearmano ranginė koreliacija, tai neparametrinis statistinės priklausomybės tarp dviejų kintamųjų įvertinimo matas. Jis nurodo ryšio tarp kintamųjų stiprumą ir gali būti išreikštas monotonine funkcija. Ideali Spearmano koreliacija lygi +1 arba –1, jei kintamieji gali būti aprašomi tobula monotonine funkcija, vienas kito atžvilgiu. Spearmano koeficientas, tinkamas tiek tolygiems, tiek diskretiems kintamiesiems ir apskaičiuojamas pagal formulę (2.1), kur kintamųjų X_i ir Y_i reikšmės keičiamos rangais x_i ir y_i, $i = \overline{1,n}$.

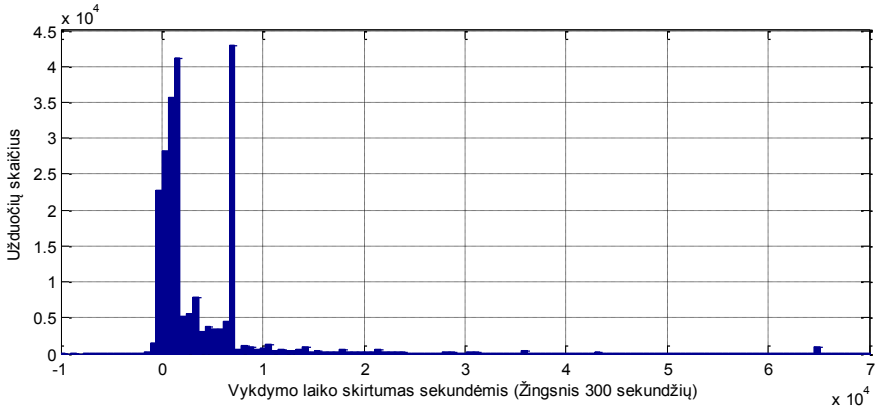
$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}. \quad \textbf{(2.1)}$$

Pagal Spearmano rangine koreliacija, ryšio stiprumas tarp t_{realus} ir $t_{numatytas}$ užduoties vykdymo laiko varijuoja esant skirtingiems duomenimis, tačiau jis egzistuoja (lentelė 2.3). Todėl tikslinga ieškoti t_ε skirstinio, kuris galėtų iš realaus užduočių vykdymo laiko generuoti numatytąjį, t. y. laiką, kuris būtų naudojamas planuojant užduotis. Histograma pavaizdavus SDSC t_ε pasiskirstymą (pav. 2.4), matoma, kad $t_{numatytas}$ gali būti didesnis arba mažesnis už t_{realus}, be to neigiami t_ε gali viršyti t_{realus} vykdymo laiką, kas sąlygotų neigiamus realius vykdymo laikus. Todėl tikslinga numatytajam vykdymo laikui išreikšti naudoti daugybos operaciją $t_{numatytas} = t_{realus} \cdot k_\varepsilon$, kur k_ε nurodo kiek kartų realus vykdymo laikas skiriasi nuo numatytojo. Kadangi $t_{numatytas} < t_{realus}$ pasitaiko gerokai rečiau, o pasiskirstymas kitoks, reikia nagrinėti teigiamą ir neigiamą t_ε dalis atskirai (pav. 2.5). Generuojant $t_{numatytas}$ iš t_{realus} teigiamą arba neigiamą paklaidą t_ε galima parinkti su tikimybe p. Patogiausiai tokius skirstinius generuoti naudojant empirinį skirstinį. $t_{numatytas}$ labai priklauso nuo pradinių duomenų, o ir jo reikšmės, sprendžiant iš histogramos 2.5(b), labai netikslios. Tikimybė, kad $t_{realus} < t_{numatytas}$, SDSC atveju, lygi 0.9.

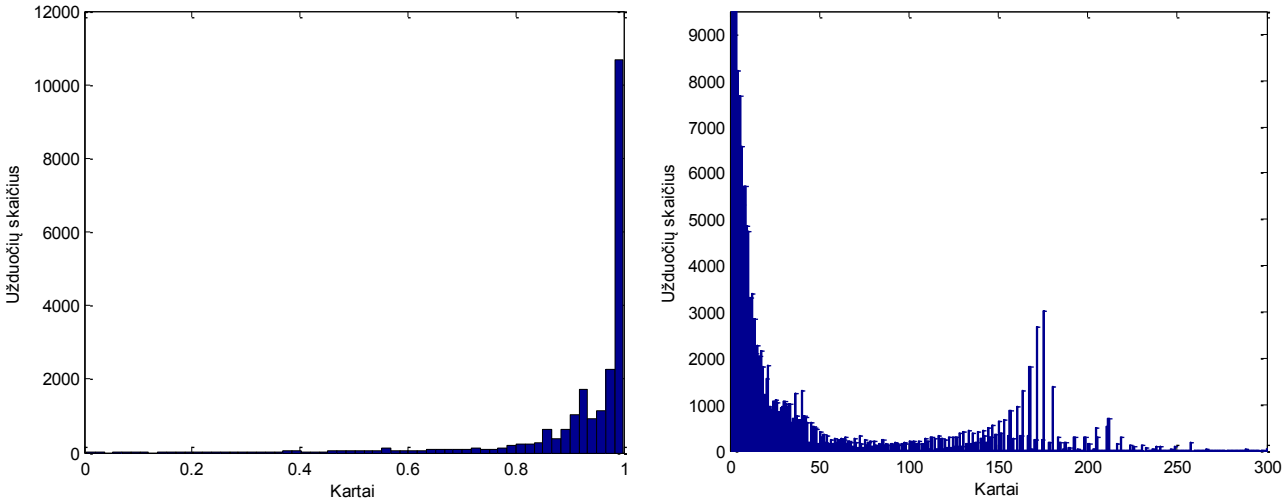
2.3 lentelė.

Realaus ir numatyto užduoties vykdymo laiko ryšys

Duomenų pavadinimas	Realus viršijo numatytą vykdymo laiką	Spearman ρ koreliacijos koeficiento reikšmė	Koreliacijos interpretacija
SDSC-BLUE-2000	10 %	0.7314	Stipri
LANL-CM5-1994	33 %	0.6094	Vidutinė
HPC2N-2002	7.3 %	0.5837	Vidutinė
LPC-EGEE-2004	35 %	0.0669	Labai silpna



2.4 pav. SDSC užduočių vykdymo laiko skirtumo pasiskirstymas



a) Kai $t_{numatytas} < t_{realus}$ b) Kai $t_{numatytas} \geq t_{realus}$

2.5 pav. SDSC užduočių vykdymo laiko skirtumo išreikšto kartais pasiskirstymas

2.2.2. VM SKAIČIAUS PASISKIRSTYMAS

Suteiktas papildomas virtualių mašinų skaičius, praktikoje gali labai žymiai įtakoti užduočių vykdymo spartą. Naudojamuose duomenyse (2.1 lentelė) užduoties vykdymo laikas tai intervalas tarp užduoties pradžios ir pabaigos laiko momentų, neatsižvelgiant į VM skaičių. Todėl užduočių vykdymo greitis dėl VM skaičiaus nekinta, bet reikalaujamų VM skaičius naudojama kaip užduoties ribojimas atliekant planavimą. VM pasiskirstymo nustatymas svarbus modeliuojant hibridinį Cloud kaip ir užduočių vykdymo laikas. VM skaičiui galioja tokios savybės:

- VM skaičius tiksliai žinomas dar prieš pradedant vykdyti užduotį.
- Reikšmė diskrečios ir modalinės. Kartais reikalaujamas VM skaičius užduotims vykdyti yra

skaičiaus 2 laipsnis: $2^i, i = \mathbb{N}$.

Kadangi VM skaičius dažnai būna pasirenkamas atsitiktinai, remiantis tam tikrais įsitikinimais arba techniniais sprendimais, paprasčiausias būdas aprašyti VM skaičiaus pasiskirstymą yra empiriniu skirstiniu [54].

Prielaidą, kad ilgiau vykdomos užduotys turėtų naudoti mažiau VM arba atvirkščiai, gali paneigti statistiškai reikšmingo ryšio tarp vykdymo laiko ir pageidaujamų procesorių skaičiaus nebuvimu (lentelė 2.4). Tai pagrindžia, kad užduočių VM skaičius tikrai atsitiktinis dydis.

2.4 lentelė.

Procesorių skaičiaus ir užduoties vykdymo laiko ryšys

Duomenų pavadinimas	Spearman ρ koreliacijos koeficiento reikšmė	Koreliacijos interpretacija
SDSC-BLUE-2000	0.4039	Silpna
LANL-CM5-1994	0.1212	Labai silpna
HPC2N-2002	-0.2238	Silpna
LPC-EGEE-2004	Visoms užduotims po 1 procesorių	

2.2.3. UŽDUOČIŲ SRAUTO TYRIMAS IR PROGNOZAVIMAS

Nustatyta jog užduočių charakteristikas atskirai galima aprašyti tam tikrais skirstiniais, nors ir nelabai tiksliai. Kai kurios charakteristikos tarpusavyje koreliuoja, tai įpareigoja charakteristikas susieti. Skirstinių nustatymas labai naudingas norint suprasti kokias Cloud situacijas reiktų įtraukti atliekant Cloud sistemos modeliavimą. Realios sistemos atkartojimui gali būti panaudojami empiriniai skirstiniai. Tačiau dažnai pavienės užduotys suteikia mažai informacijos planavimui. Net ir tiksliai aprašius užduoties, jos dar pasiskirsto laike – ir tai daro atsitiktinai. Vykdant planavimą ir siekiant patenkinti sutarties sąlygas, svarbiau žinoti užduočių srauto charakteristikas, nenagrinėjant pavienių užduočių. Pirmiausia domina kaip užduotys pasiskirsčiusios laike ir antra, koks yra užduotims reikalingų virtualių mašinų poreikis tam tikrais laiko momentais.

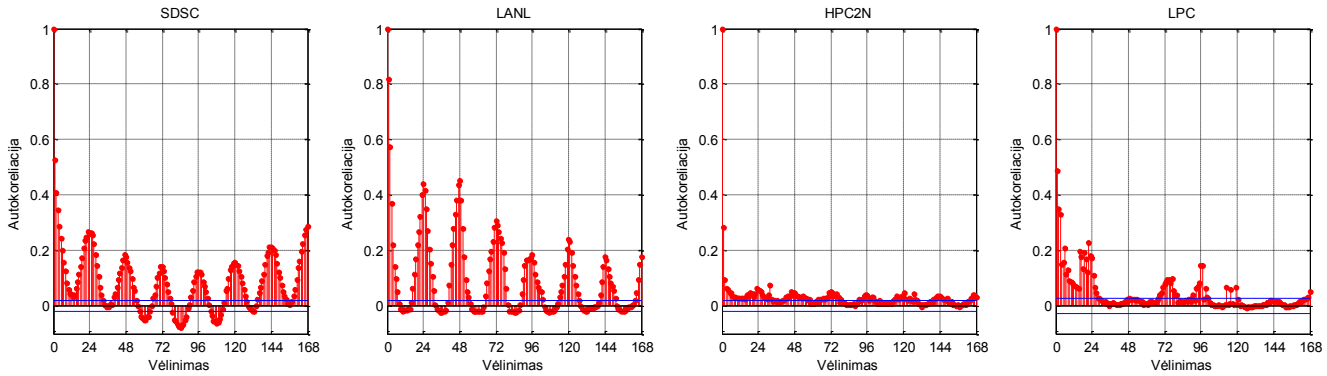
Srauto charakteristikų nustatymas labiausiai svarbus prognozavime. Prognozavimas galimas remiantis skirtingo laikotarpio periodiškumu – valandiniu, dieniniu, savaitiniu arba mėnesiniu. Dirbant su dinaminėmis sistemomis, modeliai turi būti periodiškai patikrinami, pritaikomi pasikeitusiai situacijai ir pataisomi atsižvelgiant į stebėtas paklaidas. Žinoma prognozuoti tolydžiai nerealu, todėl dažnai pasirenkamas prognozės žingsnis: minutėmis, valandomis, dienomis, savaitėmis ar ketvirčiais. Šiame darbe naudojamas valandos žingsnis tiek srauto charakteristikų aprašyme, tiek prognozavime.

Realiose sistemose užduotys, kaip ir VM užimtumas dažnai pasiskirsto laike stochastiškai. Tačiau kai kuriose sistemose, kaip SDSC ir LANL, vis dėlto galima aiškiai stebėti bent pagrindinę srauto charakteristiką – sezoniškumą.

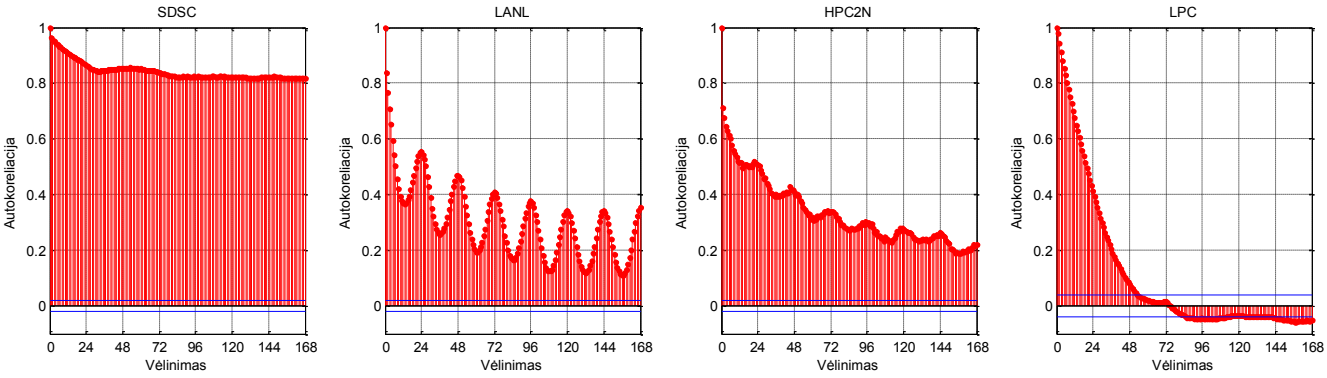
Pasinaudojant užduočių patekimo į sistemą laiku galima apskaičiuoti kiek užduočių pateikiama sistemai per valandą. Sezoniškumui atpažinti apskaičiuojamos autokoreliacijos atitinkamai dienos ir

savaitės sezoniškumui nustatyti. Iš pav. 2.6 matoma, kad statistiškai reikšmingos paros ir savaitės autokoreliacijos egzistuoja SDSC, LALN duomenims. Nagrinėjant vidutinį VM užimtumą (pav. 2.7), statistiškai reikšmingos paros ir savaitės priklausomybes stebimos LALN ir HPC2N duomenims. Iš autokoreliacijų galima spręsti, apie sezoniškumo egzistavimą.

Sezoniškumo egzistavimas pirmiausia gali būti naudojamas kaip statistinis modeliavimo duomenų aprašymo būdas, bet svarbiausiai prognozavimo tikslais. Todėl vidutinio VM užimtumo laiko sezoniškumo nebuvimas daugumoje (HPC2N duomenų sezoniškumas labai nežymus) laiko eilučių kelia problemų. Cloud užduočių planavimo idėja remiasi galimybe prognozuoti VM užimtumą.



2.6 pav. Savaitės dienų užduočių skaičiaus autokoreliacijos



2.7 pav. Savaitės dienų vidutinio užimtų VM skaičiaus autokoreliacijos

Paros ir savaitės sezoniškumo egzistavimo atveju galima sudaryti multiplikatyvų arba adityvų modelį ir suteikti sezoniškumui matematinę išraišką. Paprasčiausia, valandinį užduočių kiekį apskaičiuoti vidurkinant duomenis per visą laikotarpį ir sutraukiant į 168 valandas, t. y. vieną savaitę. Šiuo atveju naudojamasi Furje skleidiniu (2.2) dienos sezoniškumo ir n-ojo laipsnio polinomu (2.3) savaitės sezoniškumo komponentei aprašyti.

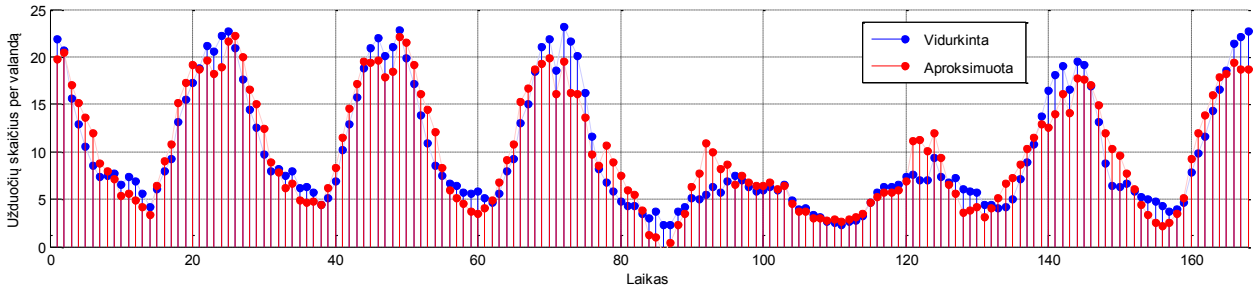
$$z_{diena}(t) = \frac{a_0}{2} + \sum_{k=1}^{m} (a_k \cos(kt) + b_k \sin(kt)), \tag{2.2}$$

čia parametrai a_0, a_k, b_k, m įvertinami pradinei laiko eilutei (užduočių srautui) $z(t)$ mažiausių kvadratų metodu, o $t = \overline{1,24}$ žymi dienos valandas.

$$z_{savaitė}(t) = p_1 t^n + p_2 t^{n-1} + \cdots + p_n t + p_{n+1}, \tag{2.3}$$

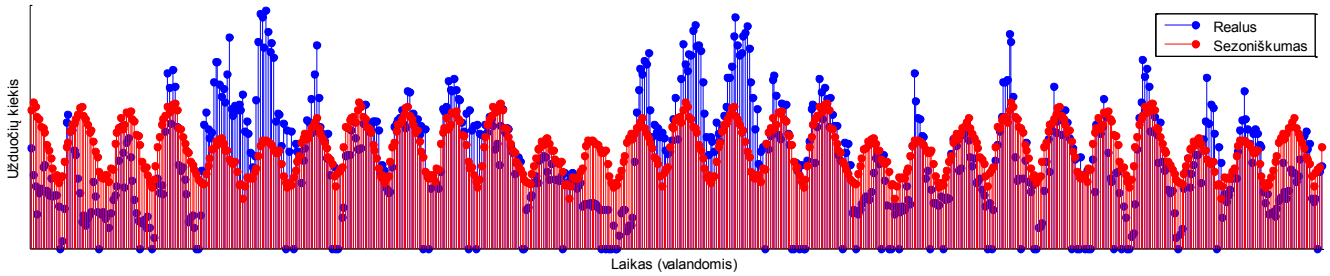
čia p_i, $i = \overline{1, n+1}$ žymi polinomo parametrus, kiekvienai laiko eilutei nustatomi vis kiti, $t = \overline{1,7}$ žymi savaitės dienas.

Adityviai apjungus savaitės ir dienos komponentes gauname savaitės valandinį pritaikymą *(angl. Fitting)*. Iliustracijoje 2.8 pateiktas pritaikymo pavyzdys, kai paklaida tarp adityvaus modelio ir vidurkintų duomenų $RMSE = 1.9795$. Kadangi dienos skirtingos, todėl kiekvienai dienai naudojami skirtingi Furje skleidinio parametrai.



2.8 pav. SDSC savaitinės-dienos sezoniškumo pritaikymo palyginimas su vidurkiais

Sezoniškumo išskyrimas yra paprasčiausias prognozavimo būdas, tačiau realybėje tai retai kada tinka visose situacijose (pav. 2.9). Skyriuje 1.5 pateikti galimi sezoniškumo patikslinimo – prognozavimo metodai. Šiame skyriuje nagrinėjamos nestacionarios laiko eilutės ir slenkančio vidurkio klasės metodai negali būti taikomi. Taigi atliekamas prognozavimas neuroniniais tinklais, naudojant nereguliarus laiko vėlinimus. Prognozavimo tikslumui palyginti naudojama RMSE ir DE paklaidų metrikos.



2.9 pav. SDSC savaitinės-dienos sezoniškumo pritaikymo palyginimas su realiais duomenimis

Prognozavimo metu neuroninis tinklas *(angl. Neural Network (NN))* buvo sukonfigūruotas taip:

- Laiko vėlinimų vektorius $\tau = [1\ 1\ 3\ 3\ 3\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1]$,
- Du paslėpti lygmenys *(angl. Hidden Layer)* po 20 ir 25 neuronus,
- Apmokymui atskirta 70 % duomenų,
- Validavimui ir testavimui 15 % duomenų,
- Prognozuojamas vienas žingsnis (valanda) į priekį.

Prognozavimo rezultatai pateikiami LALN užduotims, nes egzistuoja užduočių skaičiaus ir VM užimtumo sezoniškumas. Vertinamos pritaikymo ir prognozavimo paklaidos atskirai (skirtumas apibrėžtas 1.5 skyriuje), naudojant sezoniškumą, NN ir NN prieš tai eliminavus sezoniškumą. Rezultatai pateikiami 2.5 ir 2.6 lentelėje. Sprendžiant iš DE paklaidų, tiksliausiai prognozuoti srauto kritimą ir

kilimą galima tik su ~0.5 tikimybe, kas reiškia visišką atsitiktinumą. Tačiau iš RMSE paklaidų matoma, kad neuroninis tinklas žymiai pagerina kokybinę prognozės dalį, be to neuroniniu tinklui neturi įtakos sezoniškumo eliminavimas.

Įdomūs rezultatai gaunami nagrinėjant srautus, nepasižyminčius sezoniškumu, pavyzdžiui LPC. Prognozuojant LPC užimtų VM skaičių tik neuroniniais tinklais gaunama žymiai geresnė teigiamo šuolio paklaida $DE = 0.3361$.

2.5 lentelė.

LALN užduočių skaičiaus prognozavimo paklaidos naudojant skirtingus metodus

Metodas		RMSE	DE	Tik teigiamo šuolio DE
Pritaikymas	Sezoniškumas	7.0599	0.5168	0.5852
	Neuroninis tinklas	5.2843	0.6365	0.6855
	NN–sezoniškumas	5.0205	0.6008	0.6226
Prognozavimas	Sezoniškumas	5.2209	0.5351	0.6002
	Neuroninis tinklas	4.8880	0.6762	0.7068
	NN–sezoniškumas	3.9425	0.6294	0.6697

2.6 lentelė.

LALN užimtų VM skaičiaus prognozavimo paklaidos naudojant skirtingus metodus

Metodas		RMSE	DE	Tik teigiamo šuolio DE
Pritaikymas	Sezoniškumas	741.17	0.4392	0.4620
	Neuroninis tinklas	388.92	0.5010	0.5239
	NN–sezoniškumas	377.95	0.4928	0.5244
Prognozavimas	Sezoniškumas	609.63	0.4802	0.5133
	Neuroninis tinklas	270.97	0.5752	0.5969
	NN–sezoniškumas	291.61	0.5732	0.6032

2.2.4. APIBENDRINIMAS

Tiriant užduočių charakteristikas nustatyta, kad daugumą realių užduočių charakteristikų standartiniais skirstiniais tiksliai aprašyti negalima. Geriausiai aprašomas realus užduoties vykdymo laikas. Tačiau įvertinus, kad užduočių vykdymas laiko pasiskirstymo moda dažnai $\neq 0$, o ilgiausios užduotys sukuria begalinę pasiskirstymo uodegą, nei vienas iš tirtų skirstinių nebuvo pajėgus aprašyti realaus užduoties vykdymo laiko. Problema kilo ir aprašant realaus vykdymo laiko ir numatytojo laiko skirtumo skirstinį. Virtualių mašinų skaičiaus aprašymas dėl modalinių ir diskrečių reikšmių, gali būti aprašomas tik empirinių skirstiniu.

Atlikus užduočių kiekio ir užimtų VM skaičiaus laike tyrimą, pastebėta, kad dauguma šių realių laiko eilučių nepasižymi sezoniškumu. Iš nagrinėtų laiko eilučių galima išskirti tik LALN duomenis, kurie pasižymėjo sezoniškumu užduočių kiekio ir užimtų VM skaičiaus prasme. Tačiau bendru atveju, sezoniškumo nustatymas didelės įtakos laiko eilučių prognozavimui neturėjo. Tikintis nustatyti ar laiko eilutės reikšmės kris ar kils, DE paklaida rodė visišką atsitiktinumą visais trimis bandytais metodais.

Prognozavimo pagerėjimas fiksuotas naudojant neuroninius tinklus ir tik RMSE prasme.

2.3. TVARKARAŠČIO SUDARYMAS

Apibrėžkime struktūrą (2.4) sistemai, kuriai kuriamas tvarkaraštis, aprašyti.

$$\alpha|\beta|\gamma, \tag{2.4}$$

čia α – virtualių mašinų įranga, β – darbų vykdymo apribojimai, γ – tikslo funkcija.

Šiame darbe nagrinėsime atvejus, kai turime fiksuotą lygiagrečių vienodų mašinų (toliau virtualių mašinų (VM)) skaičių m ir užduočių skaičių n, kurioms įvykdyti reikia $1 \leq k < m$ mašinų. Tokiu atveju žymima $\alpha = P_m$. Užduočių vykdymo laikai p_j ir VM kiekio M_j reikalavimai žinomi iš anksto, t. y. tvarkaraščio sudarymo metu nekinta. Laikas tolydus, o VM kiekis M_j – sveikasis skaičius. Užduotys pradedamos ir baigiamos vykdyti visose joms reikalingose virtualiose mašinose tuo pačiu metu. Optimalaus tvarkaraščio paieškos pabaigoje užduotį aprašo: virtualių mašinų skaičius M_j, užduoties vykdymo pradžios momentas r_j ir vykdymo pabaigos momentas $d_j = r_j + p_j$. Šių duomenų pakaks tvarkaraščiui atvaizduoti Gantt‘o diagrama.

2.3.1. TIKSLO FUNKCIJA

Ieškant optimalaus tvarkaraščio, tarpusavyje lyginami sprendiniai-pretendentai, o jų kokybei įvertinti reikalinga tikslo funkcija. Realizacijos prasme patogu atskirti du tikslo funkcijos tipus: tvarkaraščio ir algoritmo. Tvarkaraščio tikslo funkcija γ vadinsimas kriterijus, kuriuo vertinamas tvarkaraščio gerumą *(angl. Fitness)*, o algoritmo tikslo funkcija $f(s)$ – funkciją, kurią minimizuos vienas iš 1.5 skyriaus algoritmų. Optimizavimo algoritmas minimizuoja f, kuri pateikia γ reikšmę.

Tyrimo metu tvarkaraščio tikslo funkcijomis γ naudojamos Makespan (2.5) ir bendro įvykdymo laiko (2.6) funkcijas. Algoritmo tikslo funkcijos $f(s)$ aprašymas pateikiamas 2.3.2 skyrelyje, apibūdinant sprendinio s formą.

$$C_{max} = \max(C_1, \dots, C_n), \tag{2.5}$$

$$\sum_j w_j C_j, \tag{2.6}$$

čia, w_j – užduočių svoriai, paprasčiausiu atveju lygūs 1, C_i – užduoties pabaigos laikas.

2.3.2. TVARKARAŠČIO PROGRAMINĖ REALIZACIJA

Naudojant 1.5 skyriuje paminėtus algoritmus praktiškai tikimasi sudaryti tvarkaraštį, tačiau nei vienas iš minėtų algoritmų nėra tikslingai parašyti tvarkaraščiams kurti. Algoritmų sprendiniai gali būti pateikti tik vektorine arba, bendru atveju, matricine forma. Kaip šiuo atveju apibrėžti tvarkaraščio sprendinį s algoritmo viduje?

Sprendinio struktūros sudėtingumas priklausytų nuo poreikių, tačiau remiantis uždavinio

„pirmiausia vykdyti ilgiausias užduotis“ analogija, sprendinį galima aprašyti svorių vektoriumi. Tai patogu, nes euristiniai algoritmai lengvai generuoja vektorinius sprendinius iš pasirinktos apibrėžimo aibės. Vektorių surikiavus, galima lengvai nustatyti užduočių vykdymo eilę viena kitos atžvilgiu. Taigi optimizavimo algoritmas parenka vektorių s, kurio ilgis n, reikšmės realios ir apibrėžimo sritis lygi intervalui $[0, 1]$. Svorius s pateikus tikslo funkcijai f, vektorius s rikiuojamas, gaunant užduočių vykdymo eilę.

Jei tvarkaraščio sudaryme dalyvauja sistemos ribojimai β, tai turėtų būti atliekama algoritmo tikslo funkcijoje. Konkrečių ribojimų įgyvendinimo algoritmai pateikti 3 skyriuje, tačiau visi remiasi bazinėmis taisyklėmis:

1. Mažiausio svorio užduotis pirmiausia,
2. Užduotis vykdoma tik tada kai tenkina visus ribojimus β,
3. Užduotis p_j negali būti pradėta vykdyti ankščiau nei p_{j-1}.

Daugeliu atveju galimos sistemos prastovos, tačiau tai sprendžia optimizavimo algoritmas, perrenkantis skirtingus galimus sprendinius.

2.3.3. EKSPERIMENTAI

Šiame skyrelyje lyginamas 1.5 skyriaus algoritmų efektyvumas, kai susiduriama su standartiniais tvarkaraščių sudarymo uždaviniais. Pagrindinė sąlyga planuojamų užduočių kiekiui: galimybė, per priimtiną laiką, atlikti pilną tvarkaraščio sprendinių perrinkimą. Dirbant su evoliuciniais algoritmais, kaip GA ir PSO, sprendiniai ne visuomet konverguoja į globalųjį minimumą. Taip nutinka dėl atsitiktinės pradinės populiacijos ir atsitiktinės prigimties geriausių sprendinių parinkimo metodikos (ruletės principas). Šie algoritmai teoriškai geba optimizuoti daugiaekstremes funkcijas, tačiau praktiškai geriausiu sprendiniu dažnai tampa vienas iš lokaliųjų. Taigi norint gauti korektišką sprendinį s^*, kai užduočių skaičius didelis, algoritmas paleidžiamas bent 5 kartus.

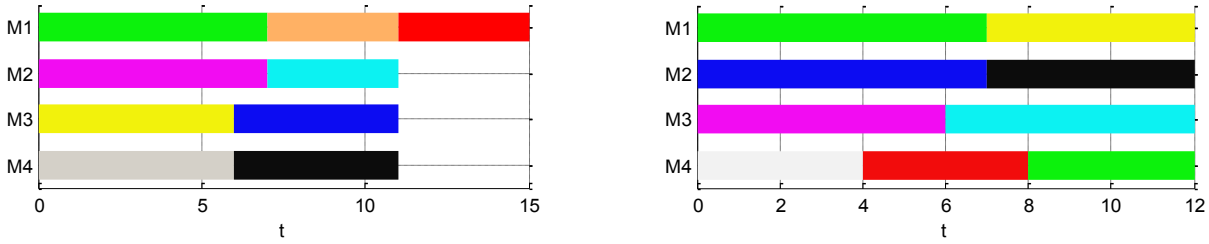
2.3.3.1 STANDARTINIS PAVYZDYS

Standartinis pavyzdys pateikimas kaip blogiausias taisyklės „pirmiausia vykdome ilgiausias užduotis“ *(angl. Longest processing time (LPT))* rezultatas. $m = 4$, $n = 9$, ir užduočių vykdymo laikai pateikiami 2.7 lentelėje, kur j žymi užduoties numerį. LPT ir optimalus (OPT) uždavinio sprendiniai pateikiami 2.10 paveiksle.

2.7 lentelė.

LPT taisyklės blogiausias pavyzdys

j	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4



a) b)

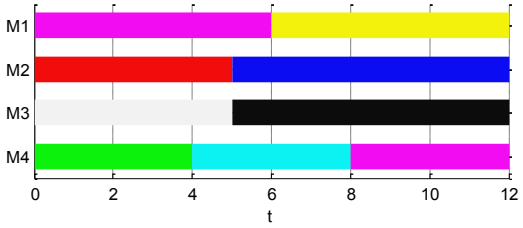
2.10 pav. a) LPT ir b) OPT sprendinio palyginimas

Tą patį uždavinį galima spręsti GA ir PSO metodais. Esant 2.8 lentelėje pateiktiems pradiniams metodų parametrams, kai tikslo funkcija $\gamma = C_{max}$, abu metodai pateikia optimalų sprendinį (Pav. 2.11). Pagrindinis metodų skirtumas sprendžiant šią problemą – laikas. Palyginimui pateikiami pilno perrinkimo, GA ir PSO metodų laikai sprendžiant šį uždavinį (žr. 2.10 lentelę). Šiuo atveju, net ir pilno perrinkimo metodas yra realiu laiku įgyvendinamas, tačiau Cloud sistemose užduočių dažnai būna gerokai daugiau, taigi GA ir PSO sparta ypač aktuali.

2.8 lentelė.

GA ir PSO metodų pradiniai parametrai

Genetinis algoritmas	Dalelių spiečiaus optimizavimas
Populiacijos dydis = 15 Generacijų skaičius = 20 Dominavimo koeficientas = 1 Kryžminimosi laipsnis = 0.8 Mutavimo laipsnis = 0.002	Populiacijos dydis = 15 Iteracijų skaičius = 20 Pažintinio mokymosi faktorius = 1 Socialinio mokymosi faktorius = 3



2.11 pav. Problemos sprendinys rastas GA metodu

2.9 lentelė.

Problemos sprendimo greičio palyginimas

Metodas	Vidutinis eksperimento laikas* (sek.)	Pagreitėjimas (kartais)
Pilnas perrinkimas	179.001696	Atskaitos taškas
Genetinis algoritmas (GA)	1.882570	95
Dalelių spiečiaus optimizavimas (PSO)	1.826641	97
*Su 2.1 skyriuje aprašyta techninė įranga.		

2.3.3.2 UŽDUOTIMS REIKIA $M_j \geq 1$ VM

Problema tampa sudėtingesnė, jei užduotims reikalingas skirtingas virtualių mašinų skaičius. GA ir PSO algoritmai optimizuoja $\gamma = C_{max}$ tikslo funkciją. Algoritmų parametrai aprašyti 2.8 lentelėje, o užduočių aprašymas, modifikuojant 2.3.3.1 skyrelio pavyzdį, pateiktas 2.10 lentelėje.

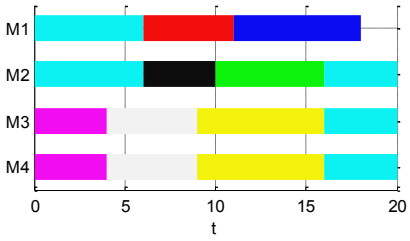
Atliekamas pilnas perrinkimas optimaliojo sprendinio radimui (Pav. 2.12), tokia galimybė vis dar

įmanoma per priimtiną laiką, ir randami GA ir PSO metodų sprendiniai (Pav. 2.13).

2.10 lentelė.

LPT taisyklės blogiausias pavyzdys

j	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4
M_j	1	2	1	2	1	2	1	2	3



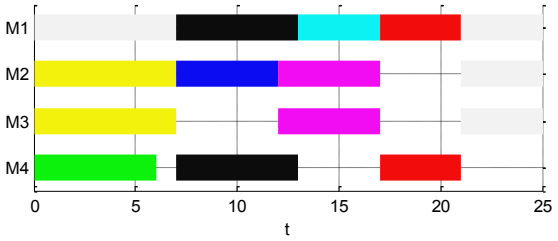
2.12 pav. Optimalus sprendinys pavaizduotas Gantt'o diagrama

a) $C_{max}(OPT_{GA}) = 20$ b) $C_{max}(OPT_{PSO}) = 20$

2.13 pav. a) GA ir b) PSO algoritmų sprendiniai atvaizduoti Gantt'o diagramomis

Uždavinio $P_m||\gamma = C_{max}$ euristikų efektyvumo nustatyti naudojamas įvertis $1 \leq \frac{C_{max}(LPT)}{C_{max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m}$. Spręsto uždavinio LPT sprendinys pavaizduotas pav. 2.14, kai $C_{max}(LPT) = 25$. Atlikus skaičiavimus (2.7), galima teigti, kad euristikų GA ir PSO rezultatas tenkina teorinius įverčius.

$$1 \leq \frac{C_{max}(LPT)}{C_{max}(OPT_{GA})} = \frac{C_{max}(LPT)}{C_{max}(OPT_{PSO})} = \frac{25}{20} = 1.25 \leq \frac{4}{3} - \frac{1}{3 \cdot 4} = 1.25 \quad \textbf{(2.7)}$$



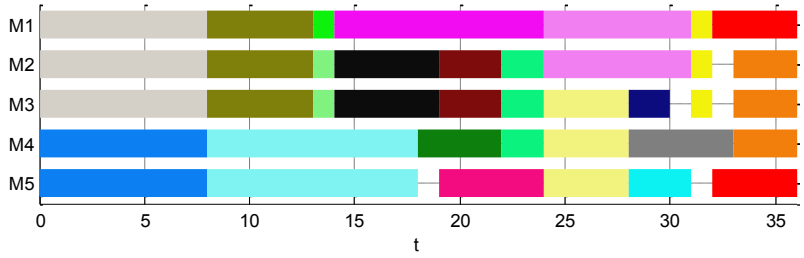
2.14 pav. LPT sprendinys

2.3.3.3 SUDĖTINGAS $M_j \geq 1$ ATVEJIS

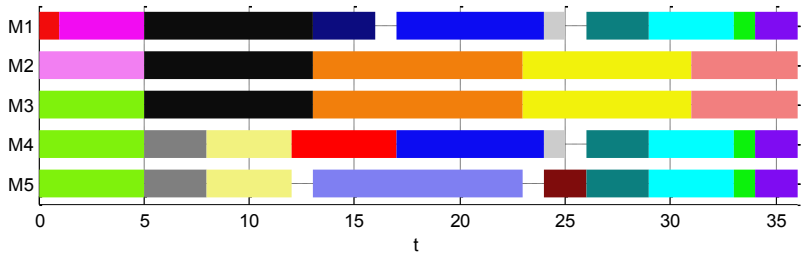
Kadangi Cloud sistemoje tuo pačiu metu laukia daug užduočių, o euristiniai algoritmai kuriami sudėtingiems uždaviniams spręsti, kurių neįmanoma išspręsti per priimtiną laiką pilno perrinkimo būdu, atlikimas eksperimentas su daug užduočių. Algoritmų nustatymai nekinta (2.8 lentelė). Tikslo funkciją $\gamma = C_{max}$, mašinų skaičius $m = 5$, užduočių skaičius $n = 30$. Pav. 2.15 ir 2.16 pateikti sprendiniai, kuriuose matome, kad abu algoritmai pateikė skirtingus tvarkaraščius, tačiau tikslo funkcijos reikšmės

sutampa $C_{max} = 36$, o iš teorinio įverčio (2.8) galime spręsti, kad rezultatas tenkinamas.

$$1 \leq \frac{C_{max}(LPT)}{C_{max}(OPT)} = \frac{42}{36} = 1.1667 \leq \frac{4}{3} - \frac{1}{3 \cdot 5} = 1.2667 \quad \textbf{(2.8)}$$

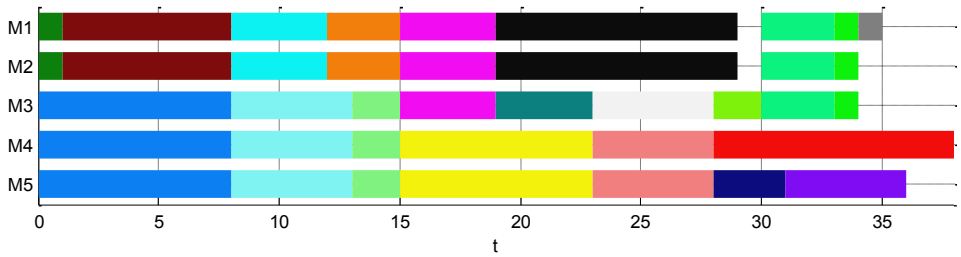


2.15 pav. GA sprendinys pavaizduotas Gantt‘o diagrama

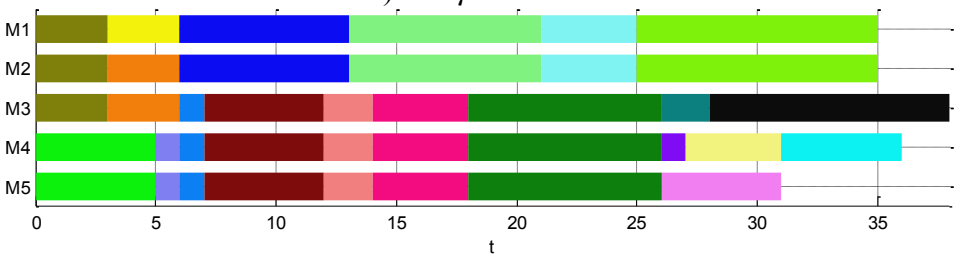


2.16 pav. PSO sprendinys pavaizduotas Gantt‘o diagrama

Pakeitus tvarkaraščių sudarymo tikslo funkciją į $\gamma = \sum C_j$, galima stebėti kitokius rezultatus (Pav. 2.17). Atlikti eksperimento pilno perrinkimo būdu, laiko prasme, nepriimtina, tačiau problemą nesunkiai sprendžia GA ir PSO, tiesa optimalaus sprendinio radimas nėra garantuotas. Šios problemos atveju, su nustatytais metodo parametrais, genetinis algoritmas nesugeba konkuruoti su PSO. PSO visados pateikia geresnį sprendinį $\gamma = \sum C_j$ tikslo funkcijos atžvilgiu, be to rezultatas pastovesnis.



a) GA $\gamma = 179$



b) PSO $\gamma = 175$

2.17 pav. GA ir PSO sprendiniai, bendro įvykdymo laiko tikslo funkcijos atžvilgiu

2.3.4. APIBENDRINIMAS

Tvarkaraščių sudarymo uždaviniai yra NP sudėtingumo, todėl pilno perrinkimo metodas rasti geriausiam šio uždavinio sprendiniui laiko atžvilgiu yra nepriimtinas. Pateikiamos alternatyvos

uždavinio optimaliam sprendiniui rasti genetiniais ir dalelių spiečiaus optimizavimo algoritmais.

Eksperimentinėje dalyje pateikti paprasti ir nagrinėjamai Cloud sistemai artimi pavyzdžiai, kuriuos vykdant algoritmai pasirodė skirtingai. Pirma, evoliuciniai algoritmai negali pirmuoju bandymu pateikti optimalaus sprendinio, todėl būtina atlikti pakartotines paieškas ir iš jų išrinkti geriausią sprendinį. Antra, GA algoritmas, pateikia prastesnius ir nepastovesnius rezultatus nei PSO. Didėjant užduočių skaičiui, kyla poreikis keisti pradinius euristinių metodų parametrus, pagrinde – pradinės populiacijos imtį.

Nors euristiniai metodai negarantuoja optimalaus sprendinio, tačiau bendru atveju yra gerokai greitesni nei pilnas perrinkimas. Be to, tikslaus sprendinio žinojimas netenka prasmės, jei užduočių vykdymo laikas nėra tiksliai žinomas, kaip dažniausiai būna Cloud sistemose.

3. PROGRAMINĖ REALIZACIJA

Hibridinio Cloud užduočių planavimui sukurtas imitacinis modelis Matlab aplinkoje. Modelyje įgyvendinamos pagrindinės Cloud galimybės ir savybės, panaudotos tirtų užduočių charakteristikos ir sąryšiai, alternatyvių modelių idėjos. Paruošiamieji modelio kūrimo darbai, šiame skyriuje nepristatomi, todėl toliau kalbama apie galutinę programą. Cloud sistemai funkcionuoti svarbūs šie pagrindiniai komponentai: vartotojai, resursų brokeris ir Cloud. Vartotojai generuoja užduotis, kurių charakteristikos nustatytos tiriamojoje dalyje. Resursų brokeris, tai valdančioji Cloud dalis, atsakinga už užduočių paskirstymą resursams, t. y. tarpinė programinė įranga, valdanti informaciją apie Cloud virtualių mašinų užimtumą ir kiekį, bei eilėje vykdymo laukiančias užduotis. Cloud tai privačios arba/ir viešos virtualios mašinos (VM). Sukurtame modelyje realizuoti vartotojai pateikiantys įvairių tipų užduotis, vienas resursų brokeris ir du Cloud: viešas ir privatus. Privatus Cloud visada riboto pajėgumo, o viešo galimybės neribojamos.

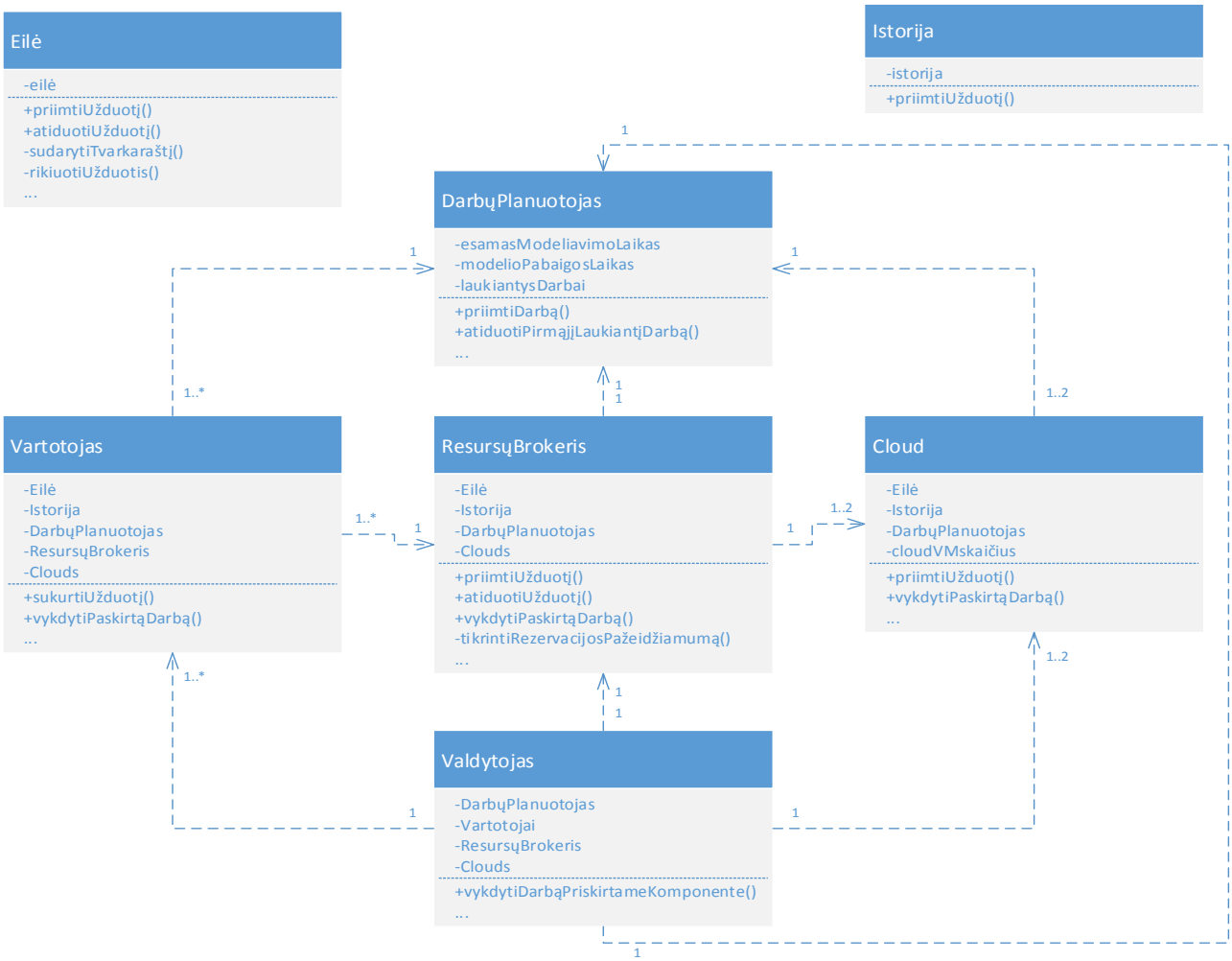
Realybėje laikas tolydus, o komponentai tarpusavyje nepriklausomi. Imitaciniame modelyje ši idėja taip pat realizuota. Tuo tikslu pagrindinius komponentus valdo už laiką atsakinga posistemė. Trumpai tariant, visus veiksmus, kuriuos reikia vykdyti dabar, komponentai atlieka patys arba pateikia vykdyti jiems pavaldiems objektams. Veiksmai, kurie turi būti atliekami vėliau, yra saugomi šios posistemės ir apdorojami reikiamu metu. Tokiu būdu komponentai tampa nepriklausomi vienas nuo kito, gali tarpusavyje konkuruoti dėl VM ir neprivalo laukti užduoties pabaigos.

Cloud modelio komponentai Matlab aplinkoje aprašomi objektais, kurių klasių diagrama pateikiama 3.1 paveiksle. Eilės ir istorijos klasės 3.1 paveiksle pavaizduotos atskirai tik dėl diagramos vaizdumo, iš tiesų jos siejasi su vartotojų, resursų brokerio ir Cloud klasėmis *..1 ryšiu. Klasių diagramoje pateikiami tik pagrindiniai kintamieji ir funkcijos būtinos suprasti modelio ryšius, parametrai ir standartiniai klasių metodai nepateikiami.

Modelis kurtas laikantis tam tikrų sistemos ir užduočių prielaidų. Technine prasme resursai traktuojami kaip vienodos virtualios mašinos, kurių pajėgumai nėra svarbūs, nes užduoties techniniai reikalavimai nusakomi diskrečiu VM kiekiu. Taip galima teigti, nes Cloud resursai visuomet virtualūs, ir tiek viešame, tiek privačiame Cloud gali būti vienodi. Taip pat atsiribojama ir nuo tinklo pralaidumo, užduočių paruošimo vykdymui laiko klausimų, o remiantis [27] saugumo problemos išsisprendžia ir toliau nėra nagrinėjamos. Remiantis šaltiniais [19, 40] užduotis galima taip pat patogiai pateikti ir privatiems, ir viešiems resursams, todėl privatūs ir vieši resursai technine prasme nekonkuruoja. Užduoties nutraukimas ir padalinimas vykdymo metu yra ekonomiškai ir spartos prasme nuostolingas [46], todėl šių galimybių vienareikšmiškai atsisakoma.

Pagal nutylėjimą, užduočių yra pakankamai daug, todėl privačių resursų dažniausiai trūksta. Į sistemą ateinančios užduotys atsitiktinės, tačiau iš tiriamosios dalies žinoma, kad tam tikros jų charakteristikos yra prognozuojamos. Tarp iš anksto tiksliai žinomų charakteristikų yra tik VM kiekis ir

užduoties prioritetas. Prioritetas nusako galimą maksimalų užduoties atidėjimo laiką: kuo mažiau užduotis gali būti atidedama, tuo prioritetas aukštesnis. Esant ribiniam atidėjimo laikui, modelis privalomai vykdo užduotis, t. y. jei tuo metu užduotis netelpa privačiame, vykdoma viešame Cloud. Remiantis šiomis savybėmis, vykdomas užduočių planavimas laike.



3.1 pav. Modelio klasių diagrama, jų pagrindinės funkcijos ir kintamieji

Modelio tikslas kuo optimaliau apkrauti privatų Cloud užduotimis arba minimizuoti užduočių vykdymą viešame Cloud. Užduočių vykdymo perplanavimas atliekamas, įvykus pasikeitimui resursų brokerio eilėje. Priklausomai nuo pasirikto planavimo tipo, gali būti minimizuojamos skirtingos tikslo funkcijos. Tvarkaraščio sudarymo atveju, tikslo funkcija minimizuojama į viešą Cloud išsiųstų užduočių kiekį. Jei užduotis apibūnama kaip: VM poreikis × vykdymo laikas, o VM vienodos, tai kiekybine prasme užduotį aprašo procesoriaus laikas *(angl. processor time)*. Pavyzdžiui, užduočiai įvykdyti reikia 2 VM ir 3 val., tai tokios užduoties procesoriaus laikas lygus 6 val. Tokiu atveju, minimizuojama tikslo funkcija $f(s)$ skaičiuos suminį viešo Cloud užduočių procesoriaus laiką.

Šiame skyriuje plačiau aprašoma sukurta programinė įranga, jos komponentai, naudoti algoritmai ir praktiškai realizuotos idėjos.

3.1. UŽDUOČIŲ APRAŠYMAS IR JŲ KŪRIMAS

Modeliavimo metu pateikiamas didelis kiekis užduočių, kurios aptarnaujamos skirtinguose komponentuose, kartais laukia eilėje arba yra papildomos nauja informacija. Siekiant optimalaus modelio spartos ir patogumo, operuojant su užduotimi ar jų grupe, santykio, užduotis aprašomas kaip vektorius eilutė, o užduočių sąrašas – matrica, kurios kiekviena eilutė aprašo užduotį. Pagrindinė taisyklė, norint nepamesti užduočių – užduotis vienu metu gali būti tik vienoje vietoje: viename sąraše, viename objekte. Taigi užduotis-vektorius **P** „keliauja“ per modelį su visomis realiais skaičiais jį aprašomomis charakteristikomis:

1. id – Užduotį unikaliai identifikuojantis numeris;
2. tpr – Sukūrimo, patekimo į sistemą laiko momentas, $tpr \in [0, \infty)$;
3. $tbpr$ – Patekimo į brokerį laiko momentas, $tbpr = tpr$;
4. $tbpb$ – Brokerio palikimo laiko momentas, $tbpr \leq tbpb$;
5. $tdpr$ – Patekimo į Cloud laiko momentas, $tbpb \leq tdpr$;
6. $tdpb$ – Cloud palikimo laiko momentas $tdpr \leq tdpb$, kai užduotis buvo įvykdyta arba paliko dėl kitų priežasčių;
7. pr – Prioritetas išreiškiamas maksimaliu užduoties atidėjimo laiku. Dvi pagrindinės užduočių grupės: 0 – aukšto prioriteto, užduotis vykdomas iš karto patekus į sistemą, > 0 – žemo prioriteto, gali laukti;
8. $tdprv$ – Vėliausias paleidimo Cloud laikas (išskiriamas kaip atskira charakteristika dėl patogumo):
   a. Aukšto prioriteto užduotims $tdprv = tpr$,
   b. Žemo prioriteto užduotims $tpr \leq tdprv \leq tpr + pr$;
9. tvr – Realus užduoties vykdymo laikas Cloud, $0 \leq tvr$;
10. tvp – Prognozuojamas užduoties vykdymo laikas Cloud, $0 \leq tvp$:
    a. Idealiu atveju (testavimo metu) sutampa su tvr,
    b. Užduoties savininkas pateikia apytikslį vykdymo laiką,
    c. tvp įvertinamas modeliavimo eigoje;
11. vm – Vykdymui būtinas VM skaičius;
12. tpb – Įtraukimo į istoriją laiko momentas, $tpr \leq tpb$. Nuo šio momento užduotis modeliavime nedalyvauja, naudojamas tik statistinėms reikmėms.

Toliau $\mathbf{S}_j$ žymės j-ąją matricos-sąrašo $\mathbf{S} \in \mathbb{M}(ns, 12)$ užduotį $\mathbf{P}$, $\mathbf{S}_{j,i}$ – užduočių matricos-sąrašo j-osios užduoties i-ąją charakteristiką arba $\mathbf{P}_i$ sutrumpintai žymės i-ąją konkrečios užduoties charakteristiką. Čia i gali būti skaičius arba trumpinys atitinkantis vieną iš pateiktų charakteristikų. Laikas, pagal nutylėjimą, skaičiuojamas sekundėmis, nebent nurodyta kitaip.

Užduotys gali būti sukuriamos pagalbinių funkcijų, remiantis tiriamosios dalies (2 skyrius) rezultatais, ir pateikiamos kaip atskiras duomenų failas. Keli testavimo skirti užduočių kūrimo metodai realizuoti modelio programinėje dalyje. Abiem atvejais modelis pats rūpinasi užduočių id unikalumu, o užduočių prioritetai p nustatomi modelio parametrais.

3.2. UŽDUOČIŲ VYKDYMO PLANO SUDARYMAS

Sukurtas užduotis vartotojai siunčia resursų brokeriui, kuris prieina prie privataus ir viešo Cloud. Resursų brokeris, priklausomai nuo užduoties charakteristikų ir Cloud užimtumo gali:

1. užduotį aptarnauti iš karto – perduoti užduotį vykdyti į privatų/viešą Cloud arba atmesti ją kaip netenkinančią būtinų sąlygų;
2. atidėti užduotį, įtraukiant ją į laukiančiųjų sąrašą.

Pirmuoju atveju, užduotis tiesiogiai neįtakoja planavimo todėl šis atvejis labai parankus. Dažniausiai, siekiant ekonominės naudos, privataus Cloud resursų trūksta ir aptarnauti užduotis galima tik jas atidedant ir sudarant eilę. Aukšto prioriteto užduočių, kurių $pr = 0$ į eilę patekti negali, nes jos būtinai vykdomos tik atėjusios. Iš eilės užduotys paimamos po vieną, taigi paprasčiausiu atveju naudojant FiFo principą, užduotys lauks savo eilės, ir jei atidėjimo laikas bus pakankamas, t. y. $t_{dabar} \leq tdprv$, užduotis bus aptarnauta privačiame Cloud, jei ne – viešame. Tačiau pasitaiko, kad daug VM reikalaujanti ir ilga užduotis stabdo likusias eilėje. Taigi užduočių tvarką eilėje paranku keisti, pavyzdžiui rikiuojant pagal užduočių VM poreikį vm, jų ilgį tvp arba sudarant tvarkaraštį.

Kadangi rikiavimo algoritmai yra integruoti Matlab pakete, jie neaptariami. Svarbu pažymėti, kad kokia bebūtų planavimo strategija, užduočių perplanavimas vykdomas po kiekvieno pakitimo Cloud aplinkoje: užduotis pradėta arba baigta vykdyti. Kitais atvejais galima naudotis tuo pačiu planu. Sudarant naująjį planą, senasis neturi jokios įtakos. Toliau aprašomas planavimas sudarant optimalų tvarkaraštį.

3.2.1. TVARKARAŠČIO SUDARYMAS

Tvarkaraščio sudarymui reikalingi pradiniai duomenys:

1. dabarties laikas,
2. laukiančių užduočių sąrašas,
3. iki kada VM bus užimtos (tariama, kad užduoties vykdymo laikas žinomas),
4. virtualių mašinų kiekio prognozė,
5. esamas Cloud virtualių mašinų užimtumas (atskirai aukšto ir žemo prioriteto užduotims) (poskyris 3.2.1.1),
6. svorių vektorius, žymintis užduočių pirmumą viena kitos atžvilgiu (generuoja optimizavimo algoritmas) (poskyriai 3.2.1.2 ir 3.2.1.3).

Duomenys 1-3 visada žinomi ir lengvai prienami. VM prognozavimui naudojamas maksimalus

aukšto prioriteto užduočių virtualių mašinų užimtumas išreikštas sezoniškumo komponente. 4-5 reikalauja būti atnaujinti kiekvieno naujo plano sudarymo metu. Tvarkaraščių sudarymui naudojamos dvejos tikslo funkcijos, viena įvertinama tik galimybė užduotis atidėti, kita papildomai vertinama VM rezervacija aukšto prioriteto užduotims. Tikslo funkcijos minimizuojamos modelyje realizuotu PSO algoritmu.

3.2.1.1 ALGORITMAS TUŠČIOMS VM APSKAIČIUOTI

VM poreikio prognozė gali būti generuojama tik diskrečiu žingsniu $t_{žingsnis}$, todėl patogu žinoti neužimtų VM kiekį $t_{žingsnis}$ žingsniu. Modeliavimo metu užduotys gali būti vykdomos ilgai (tariama, kad vykdymo laikas yra žinomas), todėl pateiktas algoritmas apskaičiuojantis kiek VM bus užimta kiekvienu laiko momentu. Laiko žingsnis kurių užrašomas užimtumas, priklauso nuo vėliau naudojamos VM poreikio prognozės, kuri gali būti tik diskrečiu žingsniu $t_{žingsnis}$.

Apibrėžiamos laiko konvertavimo funkcijos (3.1) ir (3.2) naudojamos tolesniame aprašyme. Kai $t_{žingsnis} = 3600$, (3.1) ir (3.2) atitinkamai nusako: sekundėmis skaičiuojamą laiką suapvalintą į valandas ir sekundėmis skaičiuojamą laiką suapvalintą į sekundėmis išreikštas valandas.

$$ValH(t) = \left[t/t_{žingsnis}\right],\ \ t \in [0, \infty\},\ \ ValH: \mathbb{R} \to \mathbb{S}, \quad \textbf{(3.1)}$$

$$ValM(t) = \left[t/t_{žingsnis}\right] \cdot t_{žingsnis},\ \ t \in [0, \infty\},\ \ ValM: \mathbb{R} \to \mathbb{S}. \quad \textbf{(3.2)}$$

Esamo VM užimtumo matrica pažymėta $\mathbf{U} \in \mathbb{M}(nu, 4)$, kurios kiekviena eilutė k skirta vis kitai to pačio Cloud virtualiai mašinai, o stulpeliai žymi:

1. Laiką iki kada užimta VM skaičiuojant nuo 0;
2. Užimančios užduoties prioritetą. 0 – aukštas prioritetas, 1 – žemas;
3. Užimamos paskutinės valandos numerį skaičiuojant nuo $ValH(t_{dabar})$;
4. Laiką kiek VM dar bus užimta pagal prioritetą.

Neužimtų VM kiekio apskaičiavimo algoritmas pateikiamas 3.1 lentelėje, kur:

- $\mathbf{F} \in \mathbb{S}(1, nf)$ – žingsnio $t_{žingsnis}$ suminė aukšto prioriteto užduočių VM prognozė;
- $\mathbf{Uz} \in \mathbb{S}(1, nf)$ – žingsnio $t_{žingsnis}$ suminis užimtų VM skaičius žemo prioriteto užduotys;
- $\mathbf{Ua} \in \mathbb{S}(1, nf)$ – žingsnio $t_{žingsnis}$ suminis užimtų VM skaičius aukšto prioriteto užduotys;
- $\mathbf{L} \in \mathbb{S}(1, nf)$ – žingsnio $t_{žingsnis}$ laisvų VM skaičius.

3.1 lentelė.

Kiekvieno laiko žingsnio neužimtų VM kiekio apskaičiavimo algoritmas

1	$\mathbf{U}_{k,3} = ValH(\mathbf{U}_{k,1}) - ValH(t_{dabar}),\ \forall k.$
2	$\mathbf{Uz}_{if} = 0,\ \mathbf{Ua}_{if} = 0,\ \forall if.$
3	Kol $k = 1 \le n_u$
4	Jei $\mathbf{U}_{k,3} > 0$ tai
5	Jei $\mathbf{U}_{k,2} = 0$ tai

6	$\mathbf{Ua}_{if} = \mathbf{Ua}_{if} + 1, kai\ if = \overline{1, \mathbf{U}_{k,3}}.$
7	Priešingai
8	$\mathbf{Uz}_{if} = \mathbf{Uz}_{if} + 1, kai\ if = \overline{1, \mathbf{U}_{k,3}}.$
9	$\mathbf{L} = nu - \max(\mathbf{F}, \mathbf{Ua}) - \mathbf{Uz}.$

3.2.1.2 TIKSLO FUNKCIJA SU ATIDĖJIMU

Tvarkaraščių sudaryme tikslo funkcijos naudojamos apskaičiuoti tikrinamo sprendinio kokybinę išraišką. Priešingai nei 2.3 skyriuje naudotos tikslo funkcijos, čia tikslo funkcija pritaikyta konkretaus uždavinio sprendimui ir minimizuoja reikalingą viešo Cloud procesoriaus laiką. 3.2 lentelėje pateikiamas tikslo funkcijos algoritmas, kur sprendinys $\mathbf{W} \in \mathbb{R}(1{:}\,ns)$ – svorių vektorius (kiekvienai užduočiai iš eilės-sąrašo **S**), **Cid** – netelpančių į privatų Cloud užduočių id, F – tikslo funkcijos kiekybinė reikšmė.

Ieškoma F reikšmė kiekybine prasme aprašo užduotis vykdomas viešame Cloud, taigi algoritmas papildomas šalutine funkcija – viešame Cloud vykdomų užduočių ieškojimu. Kadangi iš anksto žinoma (prognozuojama), kad šios užduotys nebepateks į privatų Cloud, jas galima iš karto vykdyti viešame Cloud. Taip galima sumažinant nereikalingą laukimo laiką.

3.2 lentelė.

Tikslo funkcijos su užduočių atidėjimu algoritmas

1	iw priskirti surikiuoto **W** didėjančia tvarka indeksus.
2	Išvalyti **Cid** vektorių.
3	$\mathbf{u} = \mathbf{U}_{k,1},\ \forall k.$
4	Tikslo funkcijos reikšmė $F = 0$.
5	Su visais iw paeiliui
6	iu priskirti surikiuoto **u** didėjančia tvarka indeksus.
7	$iup = iu_h,\ h = \overline{1, \mathbf{S}_{iw,11}}.$
8	$t_{start} = iu_h,\ h = \mathbf{S}_{iw,11}.$
9	Jei $t_{start} \leq \mathbf{S}_{iw,8}$ tai
10	$\mathbf{u}_{iup} = t_{start} + \mathbf{S}_{iw,10}.$
11	Priešingai
12	$F = F + \mathbf{S}_{iw,10} \cdot \mathbf{S}_{iw,11}.$
13	Į vektorių **Cid** įtraukti užduoties $id = \mathbf{S}_{iw,1}.$

3.2.1.3 TIKSLO FUNKCIJA SU ATIDĖJIMU IR REZERVACIJA

Tikslo funkciją su atidėjimu (poskyris 3.2.1.2), galima papildyti aukšto prioriteto užduotims reikalingų virtualių mašinų rezervacija. Taip būtų siekiama maksimaliai tenkinti reikalavimą aukšto prioriteto užduotis vykdyti privačiame Cloud.

VM rezervacija įgyvendinama dviem etapais:

1. Sudaromas užduočių vykdymo tvarkaraštis, atsižvelgiant į rezervaciją;
2. Prieš pradedant vykdyti užduotį, tikrinamas rezervacijos pažeidžiamumas.

Antrasis etapas nebus aptariamas, nes jam realizuoti naudojama ta pati idėja kaip pirmajame etape.

VM rezervacija sudarant tvarkaraštį, integruota į tikslo funkciją, kurios algoritmas pateiktas 3.3 lentelėje, jo paaiškinimas 3.4 lentelėje. Rezervacijos principas panašus į laisvų VM radimo algoritmą (poskyris 3.2.1.1), nes rezervacija tai tiesiog neįtraukiamas į planavimą VM skaičius kiekvienu laiko žingsniu $t_{žingsnis}$.

3.3 lentelė.

Neužimtų VM kiekio apskaičiavimo algoritmas

1	iw priskirti surikiuoto $\mathbf{W}$ didėjančia tvarka indeksus.
2	Išvalyti **Cid** vektorių.
3	$\mathbf{u} = \mathbf{U2} = \mathbf{U}_k, \ \forall k.$
4	Tikslo funkcijos reikšmė $F = 0$.
5	$t_{tęsti} = 0, \ th_{tęsti} = 1.$
6	Su visais iw paeiliui
7	$\mathbf{P} = \mathbf{S}_{iw}.$
8	Jei $\mathbf{u} = \mathbf{U2}$
9	Apskaičiuoti **L** pagal 3.1 lentelėje pateiktą algoritmą.
10	$\mathbf{u} = \mathbf{U2}.$
11	$ih = th_{tęsti}.$
12	Kol $ih \leq ns - ValH(\mathbf{P}_{10})$
13	Jei $\mathbf{L}_{ih} < \mathbf{P}_{11}$
14	Eiti į 12 žingsnį.
15	$t_{dabar2} = \max\left(ValM(t_{dabar}) + (ih-1) \cdot t_{žingsnis}, t_{tęsti}\right).$
16	$\mathbf{u}_{k,1} = t_{dabarN}, \ kai \ \mathbf{u}_{k,1} < t_{dabarN}, \ \forall k.$
17	$\mathbf{u}_{k,4} = \left(\mathbf{u}_{k,1} - t_{dabar}\right) \cdot (-1)^{\mathbf{u}_{k,2}}, \ \forall k.$
18	Rikiuoti $\mathbf{u}$ eilutes pagal 4 stulpelį mažėjančia ir pagal 2 – didėjančia tvarka.
19	$iul = \overline{(nu - \mathbf{L}_{ih} + 1), nu}.$
20	$t_{start} = \mathbf{u}_{iul_{\mathbf{P}_{11}},1}.$
21	Jei $t_{start} > \mathbf{P}_8$ tai
22	$F = F + \mathbf{P}_{10} \cdot \mathbf{P}_{11}.$
23	Į vektorių **Cid** įtraukti užduoties $id = \mathbf{P}_1.$
24	Eiti į 6 žingsnį.
25	$th_{pradžia} = ValH\left(t_{start} - ValM(t_{dabar})\right).$
26	$th_{pabaiga} = ValH\left(t_{start} + \mathbf{P}_{10} - ValM(t_{dabar})\right).$
27	$il = \overline{th_{pradžia} + 1, th_{pabaiga}}.$
28	Jei $\mathbf{L}_{il} \geq \mathbf{P}_{11}, \ \forall il$ tai
29	Eiti į 12 žingsnį.
30	$iu = \overline{iul_1, iul_{\mathbf{P}_{11}}}.$
31	$\mathbf{u}_{iu,1} = t_{start} + \mathbf{P}_{10}.$
32	$\mathbf{u}_{iu,2} = 1.$
33	$\mathbf{U2} = \mathbf{u}.$
34	$t_{tęsti} = t_{start}, \ th_{tęsti} = ih.$
35	Eiti į 6 žingsnį.

3.4 lentelė.

Neužimtų VM kiekio apskaičiavimo algoritmas žodžiu

1	Su kiekviena užduotimi iš laukiančiųjų sąrašo S:
2	Iš E parenkama mažiausio svorio užduotis.

3	Iš aukšto prioriteto užduočių valandinio VM poreikio prognozės *F* ir esamo VM užimtumo **u** apskaičiuojamas valandinis laisvų VM skaičius **L**.
4	Kas valandą nuo max(pirma *F* valanda; paskutinės suplanuotos užduoties paleidimo valanda):
5	Ar laisvų VM kiekis **L** šią valandą pakankamas?
6	Ne: į 4 žingsnį.
7	Pagal **u** randamas tikslus laiko momentas šią valandą (neankstesnis nei paskutinės vykdytos užduoties pradžia), tenkinantis užduoties VM poreikį.
8	Ar viršijamas maksimalus užduoties atidėjimas?
9	Taip: Užduotis atmetama. Įsimename perteklinę užduotį. Į 1 žingsnį.
10	Ar vykdant užduotį bus pažeista rezervacija *F* ateityje?
11	Taip: į 4 žingsnį.
12	Atnaujinamas **u**. Užduotis suplanuota vykdyti. Tikslo funkcijos reikšmė didinama užduoties procesoriaus laiku. Į 1 žingsnį.

3.3. INSTRUKCIJA VARTOTOJUI

Hibridinio Cloud modelis parašytas Matlab aplinkoje ir veikia komandiniu rėžimu. Programuojant naudotasi objektinio programavimo galimybėmis, todėl klasės universalios ir dažnai atlieka savo funkcijas skirtingame kontekste. Imitacinis modelis neturi specialiai sukurtos grafinės sąsajos, todėl parametrų nustatymas vykdomas programos tekste, pradinėje funkcijoje, iš kurios galima valdyti visas modelio galimybes. Galimų komponentuose realizuotų funkcijų ir parametrų sąrašas su paaiškinimais pateikiamas kiekvienos klasės pradžioje. Modeliavimo rezultatai pateikiami komandiniame lange ir gali būti pritaikomi vartotojo poreikiams, keičiant išvedamą informaciją pradinėje funkcijoje.

3.4. APIBENDRINIMAS

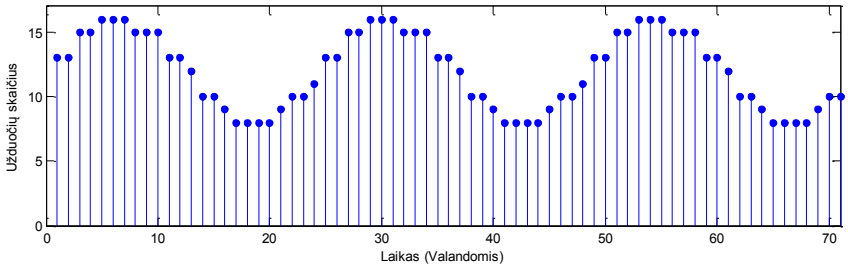
Programinės įrangos skyriuje pateiktas sukurto hibridinio Cloud imitacinio modelio aprašymas ir pagrindiniai naudoti algoritmai. Modelyje įgyvendintos užduočių kūrimo, jų planavimo ir vykdymo funkcijos, bei užduočių perdavimo tarp komponentų-objektų, statistinių duomenų kaupimo operacijos. Nors modelis realizuotas kaip tiesinio programavimo uždavinys, tačiau dėl modelį valdančios laiko valdymo sistemos puikiai imituojama lygiagretaus programavimo – Cloud sistema.

4. REZULTATAI

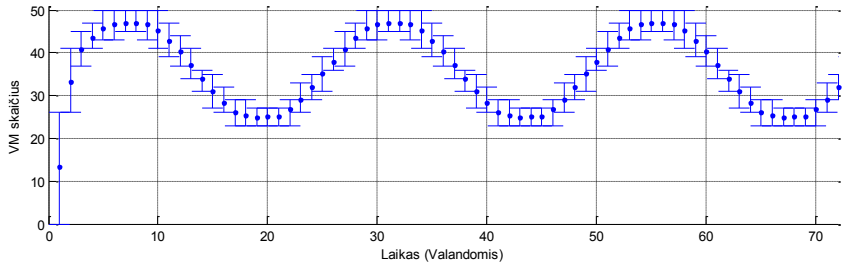
Hibridinio Cloud modelyje realizuotos kelios užduočių planavimo strategijos, naudojant skirtingus užduočių srautus ir pradines prielaidas. Šiame skyriuje praktiškai patikrinamos modelio galimybės ir pasiekiami rezultatai. Kadangi modelis integruoja daugumą šiame darbe nagrinėtų temų, todėl aptariamos pradinės sąlygos naudojamos testavimo aplinkoje. Testavimui nenaudojami realios tirtos užduotys dėl didelės įvairovės ir nevienareikšmiškai aprašomų skirstinių. Esant labai sudėtingam (realiam) užduočių srautui, neleistų korektiškai įvertinti vieno ar kito metodo galimybių. Todėl modeliuojamai Cloud sistemai nustatomos paprastesnės sąlygos.

- Užduotys generuojamos remiantis šiomis pradinėmis sąlygomis:
  - Realus vykdymo laikas = 3600 arba 7200 sek. vienodomis tikimybėmis;
  - Numatytasis vykdymo laikas apskaičiuojamas iš realaus, įtraukiant klaidos tikimybę;
  - VM skaičius = 1, 2 arba 3 VM vienodomis tikimybėmis;
  - Užduoties maksimalus atidėjimas = 0, 5400 arba 10800 sek. vienodomis tikimybėmis;
  - Užduočių srautas pasižymi vienos paros sezoniškumu (pav. 4.1) ir generuojamas pagal (4.1) formulę;
  - Užimamų VM skaičius pasižymi sezoniškumu (pav. 4.2);
  - Užduočių yra pakankamai daug, kad privatus Cloud nespėtų jų aptarnauti.
- Užduočių planavimas remiasi šiomis taisyklėmis:
  - Planavimas vykdomas naudojant numatytąjį vykdymo laiką;
  - Nulinio atidėjimo užduotys vykdomos iš karto;
  - Jei užduotys negali būti vykdomos privačiame Cloud, ir negali būti atidėtos, jos vykdomos viešame Cloud.
- Modeliavimo sąlygos:
  - Modeliuojamas laikotarpis – 4 paros. Pirmos 2 paros modeliui apsimokyti, likusios – rezultatams;
  - Privatus Cloud sudarytas iš 20 VM

$$t_{ikiKitosUzduoties} = t_{dabar} + \frac{3600}{A + B\sin(2t_{dabar}\pi/24 \cdot 3600)}, \quad A = 12, \quad B = 4. \tag{4.1}$$



4.1 pav. Į Cloud ateinančių užduočių skaičiaus per valandą



4.2 pav. Vidutinis, maksimalus ir minimalus užimtų VM skaičius kiekvieną valandą. Užduotys neatidedamos ir planuojamos FiFo metodu

Visų atliktų testų rezultatai vertinami pagal šiuos kriterijus:

- Privataus Cloud užimtumą (PCU), apskaičiuojamas pagal formulę (4.2);
- Aukšto prioriteto užduotims skirto procesoriaus laiko viešame Cloud procentas (APVC);
- Vidutinis užduoties laukimo eilėje laikas (ULL);
- Modeliavimo laikas ML, naudojantis 2.1 skyriuje aprašyta techninę įranga.

$$PCU = \frac{Privačiame\ Cloud\ vykdomų\ užduočių\ procesoriaus\ laikas}{20\ VM \cdot\ modeliavimo\ laikotarpis} \cdot 100\%. \qquad \textbf{(4.2)}$$

Užduočių planavimui realizuotos 5 strategijos. Vykdymo laikas planavimo metu tiksliai žinomas. Eksperimentų, atliktų vienodomis sąlygomis, rezultatai pateikti 4.1 lentelėje. Geriausius rezultatus PCU, APVC ir ULL kriterijų atžvilgiu galima pasiekti pirmiausiai vykdant trumpas užduotis ir tik tada ilgas. Vykdant pirmiausia ilgas užduotis, pasiekiamas toks pats privataus Cloud užimtumas, tačiau ženkliai padidėja užduočių laukimo eilėje laikas. Baziniu atskaitos tašku laikoma strategija, kai užduotys vykdomos iš eilės, pagal patekimo į sistemą momentą.

4.1 lentelė.

Skirtingų planavimo strategijų palyginimas

Strategijos aprašymas	PCU (%)	APVC (%)	ULL (sek.)	ML (sek.)
Pirma atėjo pirma išėjo (FiFo)	92.74	16.06	4843	2.123780
Pirmiausia vykdomos užduotys, kurios eilėje gali laukti trumpiausiai	93.08	18.46	5155	2.131609
Pirmiausia trumpos užduotys	95.45	14.51	2797	2.142381
Pirmiausia ilgos užduotys	95.44	17.79	4140	2.164849
Sudarant optimalų užduočių vykdymo tvarkaraštį PSO metodu	93.57	17.20	4310	18.722843

Planavimo metu, norint patenkinti sutartinius įsipareigojimus, aukšto prioriteto užduotis reikia vykdyti maksimaliai privačiame Cloud. Modelyje tai realizuojama rezervuojant virtualias mašinas išskirtinai aukšto prioriteto užduotims. Realizavus šį ribojimą, kiekvienai planavimo strategijai atlikti eksperimentai, o rezultatai pateikti 4.2 lentelėje. Geriausias rezultatas gaunamas pirmiausia vykdant trumpiausias eilėje laukiančias užduotis. Dėl VM rezervacijos ženkliai krito privataus Cloud apkrovimas, padidėjo laukimo eilėje laikas, tačiau tenkinamas naujai įvestas ribojimas.

4.2 lentelė.

Skirtingų planavimo strategijų, atliekant resursų rezervavimą, palyginimas

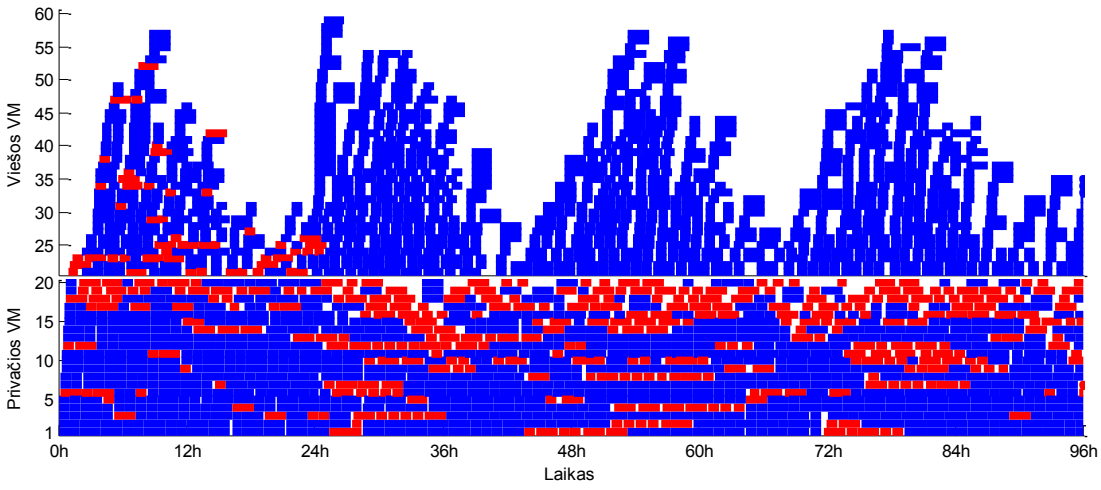
Strategijos aprašymas	PCU (%)	APVC (%)	ULL (sek.)	ML (sek.)
Pirma atėjo pirma išėjo (FiFo)	88.22	0	5126	2.359667
Pirmiausia vykdomos užduotys, kurios eilėje gali laukti trumpiausiai	87.73	0	5194	2.430095
Pirmiausia trumpos užduotys	90.72	0	3804	2.423751
Pirmiausia ilgos užduotys	87.85	0	4831	2.462681
Sudarant optimalų užduočių vykdymo tvarkaraštį PSO metodu	89.12	0	4357	71.693612

Iš pastarųjų eksperimentų iškyla šių planavimo strategijas trūkumas, ženkliai sumažinti užduočių laukimo laiko nepavyksta. Planavimas gali būti papildomas išankstiniu užduočių vykdymu, kai užduotis išlaukus eilėje maksimaliai vis tiek negalės būti vykdoma privačiame Cloud. Eksperimento rezultatai 4.3 lentelėje pateikiami tik bazinei ir dviem geriausioms strategijoms, kurių planavimo kokybė panaši. Didžiausiu privataus Cloud užimtumu pasižyminčios strategijos planavimo rezultatas iliustruotas 4.3, kur raudonai žymimos aukšto prioriteto užduotys, o mėlynas visos galimos atidėti užduotys. Iliustracijoje puikiai matoma rezervacijos įtaka jau po pirmosios modeliavimo paros.

4.3 lentelė.

Skirtingų planavimo strategijų, atliekant išankstinį užduočių vykdymą, palyginimas

Strategijos aprašymas	PCU (%)	ULL (sek.)	ML (sek.)
Pirma atėjo pirma išėjo (FiFo)	87.07	625	3.065748
Pirmiausia trumpos užduotys	88.68	739	2.862234
Sudarant optimalų užduočių vykdymo tvarkaraštį PSO metodu	89.51	783	12.830674



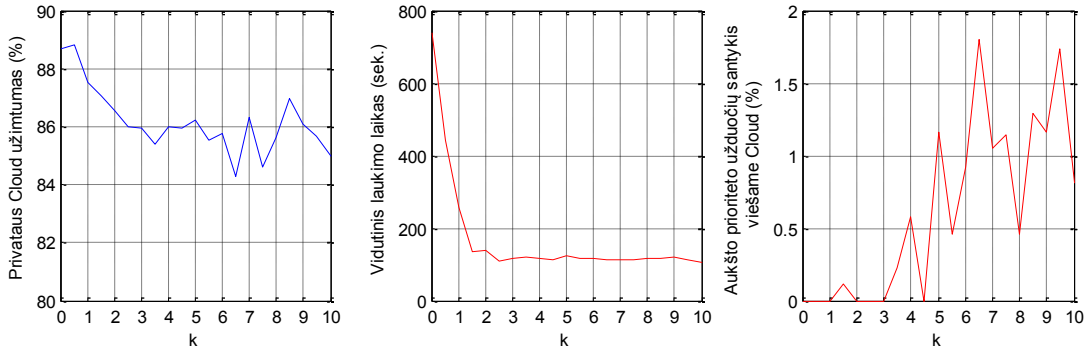
4.3 pav. Užduočių planavimo pavyzdys sudarant tvarkaraštį PSO algoritmu, atliekant rezervavimą ir išankstinį vykdymą

Visų planavimo strategijų metu, norint atlikti išankstinį užduočių vykdymą, remiamasi faktu, kad užduoties vykdymo laikas žinomas. Užduoties vykdymo laiku taip pat remiasi planavimas sudarant optimalų tvarkaraštį. Tačiau 2.2 skyriaus rezultatais, žinoma, kad vykdymo laikas nėra tiksliai žinomas. Vykdymo laiko netikslumas eksperimento metu nusakomas pridedant atsitiktinę dedamąją, kaip užrašyta

(4.3) formulėje. Čia a – atsitiktinis skaičius, generuojamas kiekvienai užduočiai, k – atsitiktinumo įtaka netikslumui. (4.3) formulė, remiantis 2.2 skyriaus rezultatais, orientuota pervertinti realų užduoties vykdymo laiką.

$$tvp = tvr + tvr \cdot (0.9 - a) \cdot k, \;\; a \in [0, 1]. \tag{4.3}$$

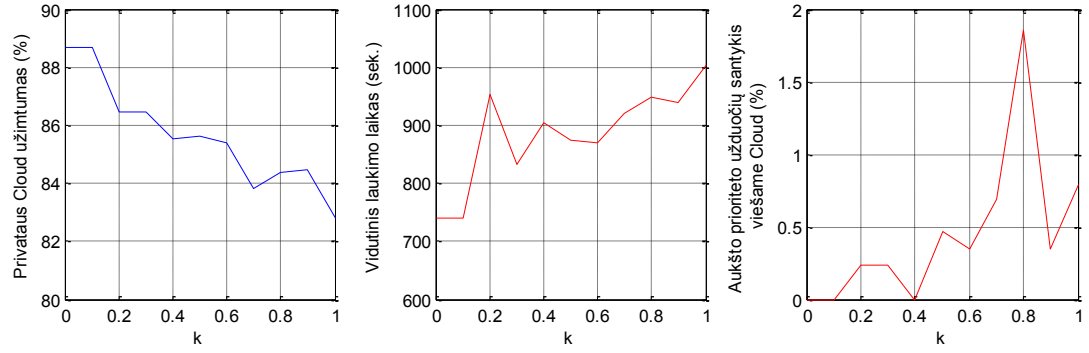
Keičiant k reikšmę (4.3) formulėje, galima stebėti užduoties vykdymo laiko paklaidos įtaką planavimo rezultatams. Eksperimentui parinkta „Pirmiausia trumpos užduotys“ strategija su VM rezervacija ir išankstiniu užduočių vykdymu. Rezultatai su skirtingomis k reikšmėmis, vertinant PCU, APVC ir ULL, pateikti 4.4 iliustracijose. Vykdymo laiko paklaidos žymiai sumažina privataus Cloud užimtumą, o laukimo laiką įtakoja tik iki tam tikros ribos.



4.4 pav. Užduoties vykdymo laiko palaidos įtaka galutiniam planavimo rezultatui

VM rezervacijai didelę įtaką turi prognozės tikslumas, todėl atliekamas tyrimas VM prognozės paklaidos planavimo tikslumui įtakai nustatyti. Čia a – atsitiktinis skaičius, generuojamas kiekvienam prognozės žingsniui, k – atsitiktinumo įtaka paklaidai. Rezultatai su skirtingomis k reikšmėmis, vertinant PCU, APVC ir ULL, pateikti 4.5 iliustracijose. Aukšto prioriteto užduočių VM prognozavimo paklaidos neigiamai įtakoja visus vertinimo kriterijus.

$$VM = VM + VM \cdot (0.5 - a) \cdot k, \;\; a \in [0, 1]. \tag{4.4}$$



4.5 pav. VM prognozavimo paklaidos įtaka galutiniam planavimo rezultatui

Apibendrinant, užduočių planavimas naudojant skirtingas strategijas pateikia skirtingus rezultatus, o jų pranašumas atsispindi nevisuose kriterijuose vienodai. Tačiau pagal visu trys kriterijus: privataus Cloud užimtumas, vidutinis užduoties laukimo laikas, aukšto prioriteto užduočių santykis viešame Cloud – planavimas yra naudingas. Planavimas ypač aktualus norint patenkinti sutarties sąlygas ir

įsipareigojimus vartotojui, kas iš tiesų veikė puikiai. Užduočių vykdymo laiko ir VM prognozavimo paklaidos labai įtakoja planavimo kokybę.

5. DISKUSIJA

Metodika su tvarkaraščio sudarymu veiktu ir esant skirtingoms virtualioms mašinoms ir jei užduotis aprašytu daugiau parametrų: atminties poreikis, procesoriaus sparta ir branduolių skaičius, saugyklos dydis. Taip pat esant daugiau nei tik vienam privačiam ir vienam viešam Cloud. Papildomų kintamųjų įvedimas akivaizdžiai pasunkintų optimizavimo užduotį, kuri pasirodė ir taip nepaprasta: optimizavimo metodo tikslumą sugebėdavo atkartoti paprastas užduočių rikiavimas pagal vykdymo trukmę.

Netikėtas buvo strategijos „pirmiausia vykdome trumpas užduotis“ rezultatas. Tai galima paaiškinti tik tuo, kad susidarančias prastovas ir trumpus langus galima užpildyti trumpesnėmis užduotims. Klausimas: kaip tokia strategija veiktų realiose sistemos, kur trumpų užduočių yra labai daug arba sistemose kur labiau varijuoja reikalaujamų VM skaičius. Iš to kyla idėja planuoti tik ilgas/dideles užduotis, o trumpas naudoti susidariusiems langams užpildyti. Taip būtų sumažintas planuojamų užduočių skaičius ir išspręstas pagrindinis kombinatorinio optimizavimo trūkumas – sparta ieškant globalaus minimumo.

Užduoties vykdymo laikas šiame darbe buvo laikomas kaip žinomas, o planavimo metu buvo naudojama tik numatyto vykdymo laiko santykinė paklaida. Literatūroje sutinkamas vykdymo laiko prognozavimo būdas naudojantis skirstiniais. Žinoma, tikslių skirstinių šio darbo metu nustatyti nepavyko, tačiau tai veiktu ir su empiriniais skirstiniais. Ateityje galima patyrinėti, kokią įtaką turi toks prognozavimo metodas planavimo rezultatams.

Bendrai, planavimo strategija sudarant tikslų vykdymo tvarkaraštį pasitvirtino, tačiau ne taip efektyviai kaip buvo tikėtasi. Nors modelis ir pasižymėjo visomis galimybėmis atlikti puikų planavimą, tačiau su realiais duomenų srautais nebuvo išbandytas. Realūs duomenys pasižymi žymiai didesniu užduočių srautu ir labai trumpomis užduotimis. Toks planavimas trunka ilgai, paklaidos kaupiasi, o sprendžiant iš testavimo duomenų rezultatas abejotinas. Paspartinti tvarkaraščio sudarymo uždavinio irgi negalima, nes trumpinant evoliucinio algoritmo darbą, labai nukenčia tikslo funkcijos sprendinio kokybė.

Darbe nustatyta, kad planavime geriausius rezultatus pateikė išankstinis užduočių vykdymas. Šis metodas nors ir remiasi prognozės rezultatais, nevisai efektyviu užduočių vykdymo tvarkaraščiu, tačiau beveik visuomet garantuoja užduoties laukimo laiko sumažėjimą. Taip, tokiu atveju krinta privataus Cloud panaudojimas, tačiau gautus nuostolius galima būtų kompensuoti pagreitėjusiu užduočių įvykdymu.

IŠVADOS

Magistrinio darbo tikslas buvo įvesti priemones padidinančias nuosavų hibridinio Cloud virtualių mašinų (VM) užimtumą ir taip sumažinti viešo Cloud poreikį. Tai reiškia trijų skirtingų uždavinių apjungimą: sistemos užduočių tyrimą, planavimo metodo parinkimą ir eksperimentavimą imitaciniu modeliu. Magistrinio darbo metu buvo atlikti tokie darbai ir gautos išvados:

- Atlikta hibridinio Cloud techninės dalies ir galimybių apžvalga. Išanalizuota tiriama problema remiantis atliktų tyrimų rezultatais, bei pagrįstas problemos aktualumas. Apžvelgtos programinės priemonės tinkamos Cloud aplinkos modeliavimui, matematinės priemonės tinkamos užduočių planavimui ir susipažinta su alternatyviais modeliais.
- Tirtos realių užduočių charakteristikos surinktos iš įvairių lygiagrečias užduotis sprendžiančių sistemų. Realūs duomenys pasižymėjo dideliu atsitiktinumu ir stipria priklausomybe nuo nagrinėjamos sistemos, todėl užduočių charakteristikoms tiksliai aprašyti standartiniai matematiniai skirstiniai netinka. Užduočių vykdymo laikui teorijoje rekomenduojamas Pareto, Veibulo ar Gama skirstiniai geba atkartoti tik bendrus bruožus tačiau ne labiausiai problemines sritis. Užduočių VM aprašymas įmanomas tik naudojant empirinius skirstinius, nes VM skaičius diskretus ir modalinis.
- Tirtas užduočių charakteristikų prognozavimas, kurių svarbiausia – aukšto prioriteto užduotims reikalingų VM skaičius. Daliai realių užduočių srautų buvo nustatytas paros ir savaitės sezoniškumas, tačiau kai kurie pasižymėjo visišku atsitiktinumu. Dėl atsitiktinumo duomenys nestacionarūs, todėl prognozavimui buvo naudojamas neuroninis tinklas. Nors RMSE prasme neuroninis tinklas prognozavo žymiai geriau nei naudojant tik sezoniškumą, tačiau DE paklaidų metrikos prasme tikslumas prastėjo.
- Efektyviam užduočių planavimu pasiekti buvo susipažinta su tvarkaraščių sudarymo teorija. Įgyvendinti Genetinis ir PSO euristiniai algoritmai jiems spręsti, bei atliktas jų palyginimas sprendžiant konkrečius tvarkaraščių sudarymo uždavinius.
- Sukurtas imitacinis hibridinio Cloud modelis, kuriame realizuotas laiko valdymas leidžia registruoti ateityje būtinus atlikti darbus, o objektai gali konkuruoti tarpusavyje. Taip įgyvendintas lengvas užduočių perdavimas tarp objektų, užduočių planavimas, vykdymas ir resursų rezervavimas. Prognozavimui remtasi sezoniškumo nustatymu, o tvarkaraščių sudarymui naudotas PSO algoritmas.
- Modeliuota su specialiai aprašytomis užduotimis, garantuojančiomis modelio galimybių panaudojimą. Tirtos 5 planavimo strategijos, kurių geriausiomis nustatytos: „pirmiausia vykdyti trumpiausias užduotis“ arba optimalaus užduočių vykdymo tvarkaraščio sudarymas. Didžiausias planavimo strategijų pranašumas prieš FiFo metodą pasiekiamas trumpinant užduočių laukimo laiką. Tuo tarpu privataus Cloud užimtumas padidėja nežymiai. Taip pat įgyvendintos papildomos

funkcijos: išankstinis perteklinių užduočių vykdymas viešame Cloud, ignoruojant didžiausią galimą atidėjimo laiką, ir aukšto prioriteto užduotims reikalingų VM rezervacija. Tai garantavo žymų pagerėjimą užduočių laukimo mažinimo ir sutarties sąlygų tenkinimo prasme.

- Nagrinėta užduočių vykdymo laiko ir VM rezervavimo paklaidų įtaka galutiniam užduočių planavimo rezultatui. Prognozės tikslumas pasirodė ypač svarbus pagal visus kriterijus: privataus Cloud užimtumą, vidutinį užduoties laukimo laiką ir į viešą Cloud išsiunčiamų užduočių kiekį. Esant netiksliai prognozei labiausiai suprastėja tvarkaraščių sudarymo metodų rezultatai, o dėl galimybės iš anksto į viešą Cloud išsiusti perteklines užduotis, atsiranda neplanuotos privataus Cloud prastovos.

Apibendrinant, planavimo rezultatas labai priklauso nuo hibridinio Cloud konfigūracijos ir užduočių charakteristikų. Esant realioms sąlygoms, planavimas tampa labai sudėtingu uždaviniu dėl tiksliai nežinomo užduoties vykdymo laiko ir didelių VM poreikio prognozavimo paklaidų.

PADĖKOS

Noriu padėkoti magistrinio darbo vadovei doc. dr. Kristinai Šutienei už manęs pastebėjimą dar bakalauro studijų metu ir kurstymą išvažiuoti Erasmus studijų. Užsiimdamas bendra moksline veikla išmokau matematinio modeliavimo ir Matlab programinio paketo subtilybių, o Erasmus metu įgijau patirties dirbant su neuroniniais tinklais. Taip pat dėkoju prof. habil. dr. Vidmantui Pekarskui už didelį norą paversti mane mokslo žmogumi ir už supažindinimą su prof. habil. dr. Minvydu Ragulskiu iš kurio mokiausi optimizavimo metodų ir laiko eilučių prognozavimo. Ir žinoma džiaugiuosi artimaisiais, kurie iškentė mano bjaurų būdą šio magistrinio darbo metu, ir, tikėtina, priims mane atgal į savo „ratą“.

ŠALTINIAI IR LITERATŪRA

1. Z. Yang, C. Yin, Y. Liu. A Cost-Based Resource Scheduling Paradigm in Cloud Computing / *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2011, p. 417-422.
2. T. Bicer, D. Chiu, G. Agrawal. A Framework for Data-Intensive Computing with Cloud Bursting / *2011 IEEE International Conference on Cluster Computing (CLUSTER)*, 2011, p. 169-177.
3. Kusum Deep, Manoj Thakur. A new crossover operator for real coded genetic algorithms / *Applied Mathematics and Computation*, Vol. 188, Issue 1, 2007, p. 895-911.
4. T. Yalcinoz, H. Altun. A new genetic algorithm with arithmetic crossover to economic and environmental economic dispatch.
5. D. Liu, J. Zic. Cloud#: A Specification Language for Modeling Cloud / *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 2011, p. 533-540.
6. V. Starikovičius. Algoritmų analizės specialieji skyriai, VGTU Matematinio modeliavimo katedra, VGTU SC Lygiagrečiųjų skaičiavimų laboratorija.
7. Wikipedia / Cloud Computing. http://en.wikipedia.org/wiki/Cloud_computing. 2012-07-05.
8. Xu Wang, Beizhan Wang, Jing Huang. Cloud computing and its key techniques / *2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, Vol. 2, 2011, p. 404-410.
9. J. Hurwitz, R. Bloor, M. Kaufman, F. Halper. Cloud Computing For Dummies, John Wiley & Sons, 2009, p. 336.
10. Soumya Banerjee, Indrajit Mukherjee, P. K. Mahanti. Cloud Computing Initiative using Modified Ant Colony Framework / *World Academy of Science, Engineering and* Technology, Vol. 32, 2009.
11. Y. Zhai, M. Liu, J. Zhai, X. Ma, W. Chen. Cloud versus in-house cluster: Evaluating Amazon cluster compute instances for running MPI applications / *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, No. 10, 2011, p. 1-10.
12. H. Heier, H. P. Borgman, B. Bahli. Cloudrise: Opportunities and Challenges for IT Governance at the Dawn of Cloud Computing / *Cloudrise: Opportunities and Challenges for IT Governance at the Dawn of Cloud Computing*, 2012, p. 4982-4991.
13. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms / *Software – Practice & Experience*, Vol. 41 Issue 1, 2011, p. 23-50.
14. C. H. Papadimitriou, K. Steiglitz. Combinatorial optimization: algorithms and complexity, Prentice-Hall, 1982, 296 p.
15. L. R. Foulds. Combinatorial optimization for undergraduates, Springer-Verlag, 1984, 227 p.

16. R. Van den Bossche, K. Vanmechelen, J. Broeckhove. Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds / *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011, p. 320-327.
17. R. Van den Bossche, K. Vanmechelen, J. Broeckhove. Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads / *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, 2010, p. 228-235.
18. A. Zinnen, T. Engel. Deadline constrained scheduling in hybrid clouds with Gaussian processes / *2011 International Conference on High Performance Computing and Simulation (HPCS)*, 2011, p. 294-300.
19. C.-H. Suen, M. Kirchberg, B. S. Lee. Efficient Migration of Virtual Machines between Public and Private Cloud / 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, p. 549-553.
20. L. Xu, G. Tan, X. Zhang, J. Zhou. Energy aware cloud application management in private cloud data center / *2011 International Conference on Cloud and Service Computing (CSC)*, 2011, p. 274-279.
21. C.-C. Lin, P. Liu, J.-J. Wu. Energy-efficient Virtual Machine Provision Algorithms for Cloud Systems / *2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, 2011, p. 81-88.
22. G. Felinskas. Euristinių metodų tyrimas ir taikymas ribotų išteklių tvarkaraščiams optimizuoti, Daktaro disertacija, Vilnius, 2007.
23. M. D. de Assuncao, A. di Costanzo, R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters, ACM, New York, NY, USA, 2009, p. 141-150.
24. Kristina Lukoseviciute, Minvydas Ragulskis. Evolutionary algorithms for the selection of time lags for time series forecasting by fuzzy inference systems / *Neurocomputing*, Vol. 73, Issues 10-12, 2010, p. 2077-2088.
25. D. G. Feitelson, D. Tsafrir, D. Krakov. Experience with the Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/. 2012-03-04.
26. D. G. Feitelson. Experimental analysis of the root causes of performance evaluation results: a backfilling case study / *IEEE Trans. Parallel & Distributed Syst.*, Vol. 16, Issue 2, 2005, p. 175-182.
27. J. Yue, Z. Zhang, J. Fu, S. Lu, X. Li, Y. Shen. Extensible architecture for high-throughput task processing based on hybrid cloud infrastructure / *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2011, p. 1452-1455.
28. D. Ardagna, S. Casolari, B. Panicucci. Flexible Distributed Capacity Allocation and Load Redirect Algorithms for Cloud Systems / *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 2011, p. 163-170.

29. J. Mockus. Genetinis Algoritmas (MethodGA, KnapGen2), KTU. http://proin.ktu.lt/~mockus /balnys/gmj2003/readme/GeneticAlgorithm.html. 2013-01-14.
30. Gan Guo-ning, Huang Ting-lei, Gao Shuai. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment / *2010 International Conference on Intelligent Computing and Integrated Systems (ICISS)*, 2010, p. 60-63.
31. Avanade Research & Insights / Global Survey: Has Cloud Computing Matured? http://www.avanade.com/Documents/Research%20and%20Insights/FY11_Cloud_Exec_Summary .pdf. 2012-12-01.
32. Alberto Nunez, Jose L. Vazquez-Poletti, Agustin C. Caminero, Gabriel G. Castane, Jesuus Carretero, Ignacio M. Llorente. iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator / *Journal of Grid Computing*, Springer, Vol. 10, No. 1, 2012, p. 185-209.
33. P. Marshall, K. Keahey, T. Freeman. Improving Utilization of Infrastructure Clouds / *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2011, p. 205-214.
34. Chenhong Zhao, Shanshan Zhang, Qingfeng Liu. Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing / *5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09*, 2009, p. 1-4.
35. W. Voorsluys, J. Broberg, R. Buyya. Introduction to CLOUD Computing / *Cloud Computing: Principles and Paradigms*, New York, USA, Wiley Press, 2011, p. 1-44.
36. M. Mattess, C. Vecchiola, R. Buyya. Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market / *2010 12th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2010, p. 180-188.
37. R. Buyya, C. S. Yeo, S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities / *10th IEEE International Conference on High Performance Computing and Communications,* 2008, p. 5-13.
38. G. Vilutis, L. Daugirdas, R. Kavaliūnas, K. Šutienė, M. Vaidelys. Model of Load Balancing and Scheduling in Cloud Computing / *Proceedings of the ITI 2012 34th International Conference on Information Technology Interfaces (ITI)*, 2012, p. 117-122.
39. D. Tsafrir, Y. Etsion, D. G. Feitelson. Modeling User Runtime Estimates / *In 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005)*, Springer-Verlag, 2005, p. 1-35.
40. D. Niyato. Optimization-Based Virtual Machine Manager for Private Cloud Computing / *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011, p. 99-106.
41. D. Feitelson. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/. 2012-01-10.

42. Randy L. Haupt, Sue Ellen Haupt. Practical Genetic Algorithms (2nd ed.), Wiley-Interscience, 2004, 272 p.
43. D. Gottfrid. Self-Service, Prorated Supercomputing Fun!, New York Times, 2007. http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/. 2012-11-10.
44. Symantec / State of Cloud Survey, 2011. http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=stateofcloud2011. 2012-09-23.
45. M. A. Iverson, F. Ozguner, L. C. Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment / *Heterogeneous Computing Workshop, 1999 (HCW '99)*, 1999, p. 1374-1379.
46. S. T. Maguluri, R. Srikant, L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters / *INFOCOM*, 2012, p. 702-710
47. S. Anoep, C. Dumitrescu, D. Epema, A. Iosup, M. Jan, H. Li, L. Wolters. The Grid Workloads Archive, TU Delft. http://gwa.ewi.tudelft.nl. 2012-01-15.
48. P. Mell, T. Grance. The NIST Definition of Cloud Computing / *Recommendations of the National Institute of Standarts and Technology*, NIST Special Publication 800-145, 2011.
49. W. Zhao, D.-S. Huang, G. Yunjian. The structure optimization of radial basis probabilistic neural networks based on genetic algorithms / *Proceedings of the 2002 International Joint Conference on Neural Networks, 2002. IJCNN '02*, Vol. 2, 2002, p. 1086-1091.
50. Cloud Computing Advices / Top 10 Cloud Computing Companies in 2013. http://cloudcomputingadvices.com/top-cloud-computing-companies-2013/. 2013-04-26.
51. Mu'alem A. W., Feitelson D. G. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling / *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 6, 2001, p. 529-543.
52. S. Malik, F. Huet. Virtual Cloud: Rent Out the Rented Resources / *2011 International Conference for Internet Technology and Secured Transactions (ICITST)*, 2011, p. 536-541.
53. T. J. Lehman, S. Vajpayee. We've Looked at Clouds from Both Sides Now / *SRII Global Conference (SRII)*, 2011, p. 342-348.
54. G. Feitelson. Workload Modeling for Computer Systems Performance Evaluation, 2011.