KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Saulius Petrauskas

Užduočių valdymo tyrimas heterogeninėje skaičiavimų aplinkoje

Magistro darbas

Darbo vadovė
doc. dr. Regina Misevičienė

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
dr. V. Pilkauskas

Užduočių valdymo tyrimas heterogeninėje skaičiavimų aplinkoje

Magistro darbas

Recenzentas
dr. K. Paulikas
2009 01

Darbo vadovė
doc. dr. R. Misevičienė
2009 01

Atliko
IFM-3/4 gr. stud.
Saulius Petrauskas
2009 01

Kaunas, 2009

SUMMARY

Distributed and parallel computing on heterogeneous clusters is the hot topic nowadays. Author of this thesis presents methods and tools used for task management and scheduling. Computing environment and middleware capabilities were analysed to better fit proposed methodology. Various user requirements were structured and combined into single generic solution to fit most frequent uses.

Research combines knowledge about *BalticGrid* distributed computing infrastructure from user perspective. Author proposes, that it is possible to gather desired results from heterogeneous computing resources. Finally, results from task management and resource allocation measurements are provided.

TURINYS

Lentelių sąrašas

Paveikslų sąrašas

TERMINŲ IR SANTRUMPŲ ŽODYNAS

CE (*Computing Element*) – servisas valdantis vidinius telkinio resursus ir eiles.

Grid – technologija apjungianti heterogeninius resursus į bendrą skaičiavimams skirtą aplinką ir atliekanti globalios resursų informacijos duomenų bazės vaidmenį.

JDL (*Job Description Language*) – užduočių aprašymo kalba.

Middleware – šiame kontekste tai programinė įranga valdanti skaičiavimų aplinką. Ji valdo resursus, užduočių vykdymą, būsenas, teises. Taip pat apjungia kompiuterių telkinius į bendrą skaičiavimų tinklą ir stebi jų būklę. Darbe tiriama gLite *middleware* aplinka.

SE (*Storage Element*) – didelės apimties duomenų saugykla.

Telkinys – šio darbo kontekste tai vienoje vietoje dislokuota kompiuterių ar serverių grupė sujungta į tinklą ir paruošta skaičiavimams atlikti.

VOMS (*Virtual Organization Management Service*) – sistema valdanti vartotojų autorizavimą tarp bendradarbiaujančių institucijų.

WN (*Worker Node*) – darbinis mazgas, kuriame fiziškai vykdoma užduotis.

WMS (*Workload Management System*) – valdantis Grid servisas, kuris tarpininkauja priskiriant resursus užduočiai.

ĮVADAS

Spartus progresas vyksta lygiagrečiųjų ir paskirstytų skaičiavimų technologijose. Pasaulyje technologija apjungianti heterogeninius resursus į bendrą skaičiavimams skirtą aplinką bei atliekanti globalios resursų informacijos duomenų bazės vaidmenį (Grid) jau tapo pagrindine skaičiavimų infrastruktūra mokslo įstaigose ir pramonėje. Šiuo tikslu apjungiama vis daugiau heterogeninių resursų į bendrą skaičiavimų tinklą.

Tokio skaičiavimų tinklo naudotojams iškyla problema kaip efektyviai ir greitai pasiekti resursus. Atliekant didesnius tyrinėjimus, siunčiant sudėtingus uždavinius skaičiavimams reikia papildomos programinės įrangos, kuri uždavinių valdymą automatizuotų. Tuo tarpu, sudėtingi uždaviniai, reikalaujantys įvairių resursų tampa vis svarbesni Grid'e [12].

Anksčiau heterogeninė skaičiavimų aplinka pati savaime nebuvo labai palanki skaičiavimams dėl nuolatinės resursų kaitos. Tačiau dabar, kai ši aplinka sugeba nuolat įjungtų resursų galia pralenkti pavienius superkompiuterius (kurie taip pat gali būti įjungiami į šį tinklą), atsiranda poreikis tuos resursus naudoti. Per pastarąjį dešimtmetį labai sparčiai gerėjo serverių ir kompiuterių, kurie naudojami skaičiavimų tinkle, parametrai. Uždaviniai taip pat keitėsi ir tapo vis labiau suskaidyti, komponentiniai. Tai įgalina juos perkelti iš homogeninės aplinkos, kur įprastai jie vis dar testuojami į heterogeninę. Heterogeninė skaičiavimų aplinka, kaip galima įsitikinti [11] darbe, turi daugybę privalumų kainos / kokybės atžvilgiu.

Pagrindiniai veiksniai, kurie inicijavo šį tyrimą yra keli: skaičiuojant elektroninių schemų modelius vis nepavykdavo pasiekti patikimų rezultatų naudojant vieną kompiuterį. Net perkėlus skaičiavimus į Grid, didesnių schemų modeliai vis dar nepasiekdavo reikiamo tikslumo. Kito uždavinio atveju naudojant centralizuotą duomenų surinkimą ir tam uždaviniui pritaikytą valdymo mechanizmą Grid iš dalies pasiteisino. Daugėjant uždavinių prireikė vieningos metodikos ir įrankių toms užduotims valdyti. Norint sukurti tokią metodiką buvo būtina išanalizuoti uždavinių apribojimus, naudotojų reikalavimus, infrastruktūros teikiamas galimybes ir nesėkmingus scenarijus. Uždavinių valdymo automatizavimo tyrimus skatina ne tik naudotojų poreikis tokioms sistemoms, bet ir Europos naujos kartos Grid'ų vizija, skelbianti autonominio skaičiavimo (*Autonomic Computing (AC)*) paradigmą [12].

Šiame darbe tiriami metodai padedantys naudotojui atlikti užduočių planavimą ir valdymą heterogeninėje skaičiavimų aplinkoje. Visapusiškai išanalizuoti galimi sprendimai tiek iš vartotojo, tiek iš užduočių valdymo įrangos *middleware* pusės. Darbe sutelkti ir apibendrinti vartotojo reikalavimai uždaviniams. Nustatyti trys pagrindiniai uždavinių suskaidymo tipai ir pateikti siūlymai bei metodika valdymo problemai spręsti.

Šio darbo tikslas – ištirti heterogeninę skaičiavimų aplinką ir pasiūlyti metodiką užduočių valdymui.

Tyrimo objektas – heterogeninė skaičiavimų aplinka sujungta į Grid.

Darbo uždaviniai:

- Išanalizuoti esamo paskirstytų skaičiavimų tinklo BalticGrid Baltijos šalių regione veikimą;

- Atlikti standartinių Grid programinių priemonių (skirtų užduočių valdymui) galimybių analizę;
- Surinkti ir apibendrinti esamų vartotojo sąsajų privalumus ir trūkumus;
- Pasiūlyti metodologiją užduočių valdymui;
- Įvertinti pagal pasirinktą metodologiją gautus skaičiavimų valdymo rezultatus.

Tyrimo metu keliama hipotezė, kad uždavinius, kurių nepavyksta išspręsti taikant ir optimizuojant sukurtus algoritmus, galima išspręsti (iki tam tikro numatyto tikslumo) efektyviai apjungiant didelius skaičiavimų resursus. Tokiu būdu heterogeninėje ir nuolat kintančioje skaičiavimų aplinkoje gaunami patikimi rezultatai. Ši hipotezė pasitvirtina atliekant eksperimentus su schemų modeliavimo uždaviniu §3.4.

Tyrimo metu sukaupta medžiaga buvo paskelbta mokslinėse KTU konferencijose „Informacinės technologijos 2007“ ir „Informacinės technologijos 2008“. Atspausdinti straipsniai KTU konferencijų „IT’ 2007“ ir „IT’ 2008“ medžiagoje, kuriuose akcentuojami skaičiavimų rezultatai, ir straipsnis KU konferencijoje „Fundamentiniai tyrimai ir inovacijos mokslų sandūroje“ 2008m.. Perskaitytas pranešimas VVK konferencijoje „Innovative Infotechnologies for Science, Business and Education“ 2008 m. tema „Computations on heterogeneous clusters“. Sukurta metodika naudojasi KTU Informacinių Technologijų Plėtros Instituto (ITPI) kompiuterių sistemų skyriaus darbuotojai. Tikimasi kurti naujas sąsajas ir įrankius naudotojams šios metodikos pagrindu.

1. SKAIČIAVIMŲ APLINKOS ANALIZĖ

Šiame skyriuje apžvelgiama paskirstytų skaičiavimų aplinka. Pateikiama teorijos analizė susijusi su tyrimo uždavinių problematika. Taip pat pateikiamos įvairių autorių nuomonės sprendžiant panašias problemas.

1.1. Literatūros apžvalga

Pirmiausia svarbu apibrėžti kas tai yra heterogeninė aplinka ir kodėl ji svarbi šiame darbe. Kalbant apie užduočių valdymą, heterogeninė aplinka yra ta skaičiavimų aplinka, kurioje vykdomos užduotys. Šiame darbe tiriamos aplinkos realizacija remiasi Grid technologijų pagrindu. Pagrindinis uždavinys – tokios aplinkos efektyvus panaudojimas. Heterogeniniai skaičiavimai, tai koordinuotas skirtingų tipų mašinų, tinklų ir sąsajų naudojimas su tikslu maksimizuoti jų bendrą našumą ir/arba rentabilumą [26].

Didžioji šio tyrimo dalis skirta apžvelgti esamas metodikas ir įrankius. Daugelis užduočių valdymo mechanizmų remiasi euristika [26, 13, 8, 6]. Taip yra tikriausiai todėl, kad užduočių valdymas, nenaudojant valdymo metodikos ar meta-valdymo įrankių, atliekamas pačio naudotojo – bandymų ir klaidų keliu. Projektuojant užduočių valdymo sistemas bandoma modeliuoti veiksmus, kuriuos atlieka naudotojas ir šiuos veiksmus optimizuoti.

Grid skaičiavimų technologijos yra pakankamai brandžios, kad galėtų būti sėkmingai taikomos intensyvių skaičiavimų reikalaujančiuose moksliniuose uždaviniuose [2].

Tokio tipo tyrimų svarbą lemia tai, kad nuo to gali priklausyti daugybės kitų tyrimų sėkmė. O technologija daranti Grid aplinką lengviau pasiekiamą galutiniam vartotojui bus reikšminga daugelyje mokslo sferų [2]. Meta-valdymo sprendimai nėra nauji ir remiasi esamais valdymo įrankiais, kad palengvintų naudotojų darbą bei suteiktų aplinkai didesnį stabilumo laipsnį. Pagrindinis skirtumas tarp resursų paskyrimo Grid'e lyginant su kitomis aplinkomis yra tam tikro lygio heterogeniškumo įsitraukimas [7].

Egzistuoja valdymo sistemų, kurios naudoja kelių pakopų darbų planavimą, kai žinoma konkrečių užduočių aibė [1]. Tokios užduočių aibės parametrai kiekvienam uždaviniui yra specifiniai ir nustatomi iš anksto DAG (angl. *Directed Acyclic Graph*) grafo pavidalu. Tokie grafų modeliai yra populiariausi modeliai, naudojami statiniame užduočių planavime [27].

Viena iš rekomendacijų, kaip turėtų būti kuriama užduočių valdymo sistema – agentų sistemos. Tokiu pat principu kaip naudotojas išmoksta optimaliai dirbti su Grid per savo bandymus ir klaidas, galėtų apsimokyti ir kažkuris valdymo sistemos agentas [10]. Tačiau tokių bandymų gali būti per daug produkcinei sistemai, o jų modeliavimas gan sudėtingas dėl dinamiškos Grid prigimties [17, 8] ir kitų sunkiai nuspėjamų veiksnių. Panašiu principu veikiantis genetinis algoritmas irgi gali būti naudojamas modeliuojant užduočių suskirstymą. Genetiniai algoritmai su tam tikrais operatoriais yra pranašesni už statinius planavimo algoritmus [6]. Minėtame darbe efektyviausi operatoriai nėra įvardinti kol dar nėra eksperimentinių rezultatų.

Grid infrastruktūra teikia skaičiavimų ir duomenų saugojimo paslaugų virtualizaciją ir abstrakciją. Grid aplikacijos šiuo metu randasi kuo arčiau šių esminių resursų [21]. Turbūt dėl šios

priežasties dar labai populiarus centralizuotas skaičiavimų valdymo portalų teikimas. Dabartiniu metu skaičiavimų tinklai plečiasi ir vis daugiau partnerių jungiasi į virtualias organizacijas. Kai dauguma mechanizmų resursų rezervavimui pagrįsti centralizuotu priėjimu, greičiausiai sulauks veikimo problemų Grid sistemoms plečiantis [7].

Internetinė sąsaja nutolusių *Grid* resursų pasiekimui siūlo lengvą interakciją su naudotoju, kuri paslepia *Grid* sistemų heterogeniškumą ir sudėtingumą [19].

Darbe siūlomos trys bendros metodikos resursų išrinkimui: resursų prioritizavimas, resursų pašalinimas ir darbų dubliavimas [13].

Heterogeninėje aplinkoje skirtingi resursai, priklausantys įvairioms organizacijoms su skirtingomis taisyklėmis ir plačiai geografiškai išsiskirstę, yra naudojami kartu [5].

Grid dominuoja virtualios organizacijos (VO), kurios susieja naudotojus su heterogeniniais resursais. Grid aplinka nuolat kinta. Keičiasi naudotojų ir resursų skaičius. Būtina atkreipti dėmesį, kad valdymo metodika turi būti pakankamai lanksti, kad galėtų prisitaikyti prie padidėjusio resursų pasiekiamumo dėl VO plėtros. Sunku tiksliai nustatyti kiek VO gali plėstis, tačiau manoma, kad jos galės apjungti daugiau kaip dešimtis tūkstančių vartotojų [10].

Norint įsigilinti į užduočių valdymą, pirmiausia reikia išsiaiškinti kas tai yra kompiuterių telkinys. Kompiuterių telkinys apibrėžiamas kaip grupė sujungtų kompiuterių, dirbančių kartų labai artimai tam, kad daugeliu atžvilgių suformuotų vieną kompiuterį [3].

Vienas iš sunkumų su kuriuo susiduriama kuriant užduočių valdymo sistemą, tai kad trūksta tikslios informacijos apie resursų ir užduočių būklę iš pačios *middleware* (šiame kontekste tai programinė įranga valdanti skaičiavimų aplinką. Ji valdo resursus, užduočių vykdymą, būsenas, teises. Apjungia kompiuterių telkinius į bendrą skaičiavimų tinklą ir stebi jų būklę).

Darbe vystomos euristikos suteikia pagrindą efektyvioms užduočių valdymo strategijoms.

1.2. Grid middleware galimybių studija

Atliekant šį tyrimą buvo bandoma rasti galimus sprendimus panaudojant esamas *middleware* savybes. Dideliam kiekiui užduočių išsiųsti numatyti keli būdai. Šiame skyrelyje atskleidžiama jų tiksli paskirtis, taikymas ir trūkumai.

DAG uždaviniai. Tai toks uždavinių tipas, kuris skaidant į užduotis atvaizduojamas DAG (*Directed Acyclic Graph*) grafo pavidalu. Užduočių grafas dažniausiai naudojamas atvaizduoti komunikacijas, reikalavimus ir prioritetus tarp užduočių [27]. Pagrindinė tokio vaizdavimo ir suskaldymo paskirtis aprašyti užduotis, kurių vykdymo eiliškumas ir tarpusavio komunikacija yra griežtai iš anksto nuspręsta. Pagal šiuos principus realizuotas P-GRADE įrankis (žr. 1.3.6 skyrelį), kurio pagalba vartotojas gali interaktyviai kurti ir keisti užduočių grafą.

Esminis šio uždavinių tipo trūkumas yra tai, kad nepavykus įvykdyti bent vienos užduoties – visa užduočių aibė nutraukiama, o dalies rezultatų grąžinimas retai pasiseka. Jau nuo 2006-ųjų metų situacija nepasikeitusi ir, kaip buvo pasakyta viename seminare: „DAG uždaviniai beveik veikia“.[1]

[1] Informacija iš skaidrių „Joining the GRID as Application“ (http://www.litgrid.lt/naujienos/files/AppSupport-Vilnus17Jan06.pdf) [žiūrėta 2008-01-05].

Parametriniai uždaviniai.

Šis užduočių tipas geriausiai atspindi šiame darbe siūlomos metodikos (žr. 2 skyrelį) tikslą ir paskirtį. Parametriniai uždaviniai, tai specializuotas DAG užduoties atvejis. Šis uždavinių tipas numatytas dideliam užduočių skaičiui paleisti aprašant jas vienu šablonu. Užduotys paleidžiamos perduodant parametrą, kurio rėžiai aprašomi JDL (*Job Description Language* – užduočių aprašymo kalba) faile (žr. 2.1 skyrelį). Kaip ir DAG atveju, visa užduočių aibė nebus sėkmingai įvykdyta jeigu bent viena užduotis dėl kokių nors priežasčių nesibaigs sėkmingai. Remiantis QoS (Quality of Service) arba paslaugų kokybės tyrimu, aukščiausiai įvertinto telkinio pupa.elen.ktu.lt QoS yra lygus 0.85 [25]. Galime apskaičiuoti, kokia tikimybė, kad pasirinktos užduotys bus įvykdytos sėkmingai remdamiesi binominiu skirstiniu:

$$\binom{n}{k} p^k (1-p)^{(n-k)} \tag{1}$$

Pasirinkime, kad užduotys nedubliuojamos, nes tai nenumatyta standartinėse gLite valdymo priemonėse. Tarkime, kad atliksime $n = 500$ bandymų. Kadangi užduotys vykdomos heterogeninėje aplinkoje, sakykime, kad jos tolygiai paskirstomos po visus telkinius. Taip pat įsivaizduokime, kad telkiniai modeliuojamame Grid'e jau pasiekė tokį patį servisų kokybės lygį, kaip ir geriausias iš jų. Todėl pasirenkame geriausio telkinio QoS reikšmę, kuri yra $p = 0.85$. Tam, kad visas parametrinis uždavinys būtų įvykdytas sėkmingai, visos $k = 500$ užduočių turi pavykti. Įstatę šias reikšmes gauname, kad tokia tikimybė yra 5.12×10^{-36}. Gautoji tikimybė labai maža, todėl pastebime, kad neįvertinome galimybės pakartoti kiekvieną užduotį. Tai nustatoma parametru „*ShallowRetryCount*“ apie kurio panaudojimą plačiau skaitykite 2.1 skyrelyje. Pagal nutylėjimą šio parametro (pažymėkime raide r) reikšmė yra $r = 3$ ir gali kisti intervale $-1 \leq r \leq 10$ su tam tikromis išlygomis. Taigi kiekviena užduotis gali būti pakartojama r kartų. Įvertinę šį pakeitimą papildome formulę (1) nauja informacija. Dabar lengviau apskaičiuoti tikimybę $q = 1 - p$, kad užduotis nebus įvykdyta per r pakartojimų: $q = (1-p)^r$. Įvykdę pakeitimus ir supaprastinę išraišką (2) gauname uždavinio įvykdymo tikimybę P:

$$P = (1 - (1-p)^r)^n \tag{2}$$

Įstatę reikšmes gauname, kad $P = 0.18$. Taigi galima teigti, kad su šiais vykdymo apribojimais, parametriniai uždaviniai yra nepraktiški.

Užduočių rinkiniai.

Užduočių rinkinys (angl. collection) tai aibė nepriklausomų užduočių, kurios siunčiamos, stebimos ir valdomos kaip viena. Pagrindinė rinkinių savybė – bendri pradinių duomenų failai. *WMS* tokioms užduotims stengiasi atlikti kuo mažiau užklausimų, taupyti tinklo resursus perduodant po vieną failo kopiją į telkinį. Užduočių rinkinius įmanoma priversti elgtis panašiai kaip ir parametrines užduotis naudojant papildomus programinius skriptus. Kaip ir DAG tipo užduočių atveju, rinkiniuose užduotys taip pat dalinai priklausomos viena nuo kitos. To privalumas – greitas ir mažiau resursų reikalaujantis užduočių išsiuntimas, negu siunčiant pavienes užduotis. O pagrindinis trūkumas: panašiai kaip ir DAG tipo užduočių – neįmanoma parsisiųsti

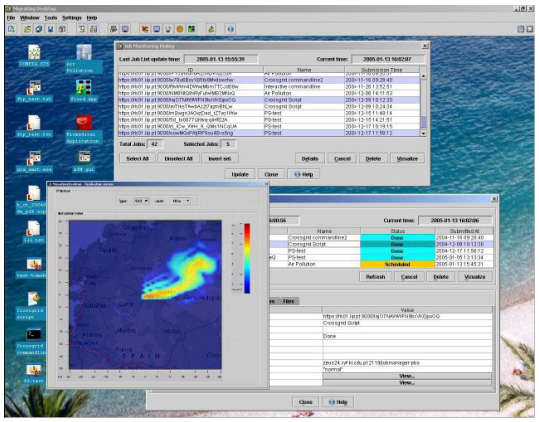
dalinių rezultatų, kol visos užduotys nebaigtos. Tik visoms užduotims pasibaigus, pagrindinė (*master*) užduotis pereina į pabaigos būseną.

Ekonominis modelis „Tycoon“. Vienas iš bandymų išspręsti užduočių valdymą *middleware* pusėje remiasi rinkos dėsniais resursų paskirstyme. Tam buvo suprojektuota ir realizuota *Tycoon* sistema. Ši sistema remiasi aukciono principu užduočių susiejimui su resursais, o į užduočių skirstymą žvelgiama per ekonominę prizmę. Tokios sistemos naudotojas nurodo kreditų skaičių, galutinį užbaigimo terminą ir kiek mazgų rezervuoti vykdymui [15]. Sistema pagal iš anksto numatytas rinkos strategijas bando rasti optimalų užduočių paskirstymo variantą, kuris tenkintų kuo daugiau naudotojų apribojimų. Strategijos vykdymas atliekamas agentų principu: imituojamas aukcionas, kurio metu agentai „atstovauja“ naudotojui. Sistemoje įvesta „banko“ rolė, kuri seka visus vykdomus sandorius ir kiekvieno agento balansą. Kol kas neteko matyti šios sistemos pagrindu realizuoto serviso, tačiau panašius valdymo principus bandyta modeliuoti 2003-iaisiais metais aprašytame darbe [7] „The application of bioinspired engineering principles to Grid resource allocation“.

1.3. Uždavinių valdymo įrankių studija

1.3.1. Valdymo aplinka „Migrating Desktop“

Migrating Desktop pateikia vienodą grid darbo aplinką nepriklausomai nuo specifinės Grid infrastruktūros. Šis įrankis taip pat nepriklausomas ir nuo naudotojo operacinės sistemos, nes yra parašytas Java programavimo kalba. Naudotojui užtenka turėti arba interneto naršyklę ir jos pagalba pasileisti JVM (Java Virtual Machine) per naršyklės įskiepį, arba naudoti JWS (Java Web Start). Autorių teigimu, JWS aplinkoje Migrating Desktop veikia kur kas sparčiau dėl programos kešo (angl. *caching*) lokaliame kompiuteryje [19]. Užduočių aprašymas ir valdymas atliekamas interaktyvių dialogų pagalba. Įrankyje puikiai demonstruojamos Grid galimybės. Įgyvendintas patogus duomenų ir rezultatų failų valdymas naudojant „Grid Commander“ komponentą. Galima vykdyti interaktyvias užduotis. Yra numatyta galimybė plėsti šią aplinką (Desktop) įskiepių pagalba. Vienas tokių įskiepių skirtas rezultatų vizualizacijai (pav. 1).



1 pav. Grafinė *„Migrating Desktop“* vartotojo sąsaja

Migrating Desktop būtinas priėjimas prie RAS (Roaming Access Server) serverio, kurio

programinis kodas viešai prieinamas pagal CROSSGRID licenciją [19]. Tai reiškia, kad ši aplinka yra dalinai centralizuota ir jos veikimas priklausomas nuo tarpinio serverio. Visas užduočių valdymas patikimas RAS, kuris šiuo metu veikia kaip tarpininkas tarp įvairių *middleware*. Šiam tarpiniam servisui patikėta: užduočių siuntimas ir monitoringas, naudotojų ir duomenų valdymas, autorizavimas. Atliekamas informacijos apie uždavinius valdymas.

Įrankis puikiai tinka mokytis dirbti ir susipažinti su Grid technologija. Šis įrankis orientuotas į naudojimo paprastumą ir nėra autonominė produkcinio lygio užduočių valdymo sistema.

1.3.2. Internetinė sąsaja „gridcom“

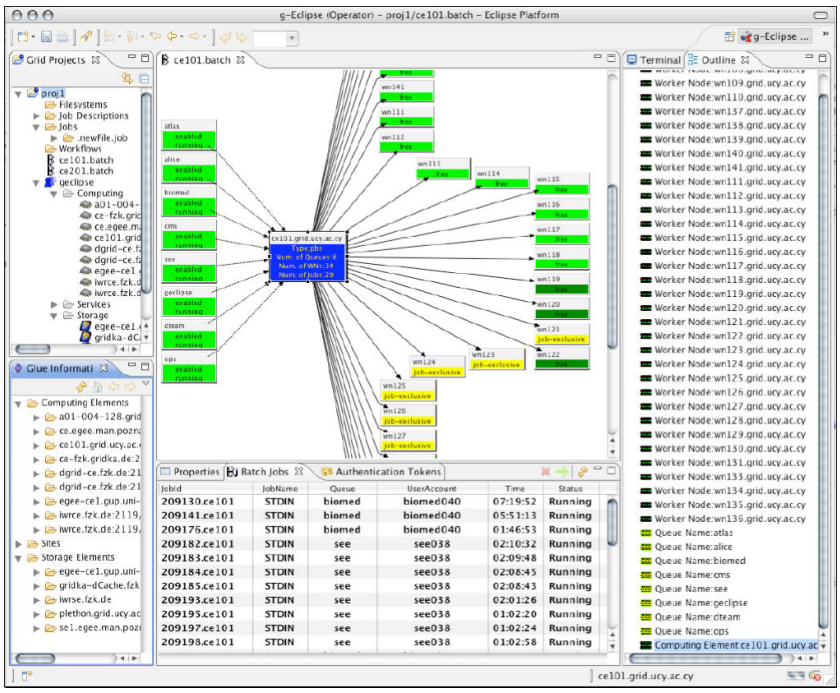
Tai novatoriškas užduočių valdymo ir uždavinių informacijos valdymo įrankis realizuotas per internetinę sąsają. Naudotojai turi galimybę įkelti ir aprašyti vykdomus uždavinius. Priemonė realizuota PHP programavimo kalba. Grid aplinkos heterogeniškumas ir sudėtingumas paslėptas po paprasta vartotojo sąsaja, išskyrus klaidų pranešimus. Valdant užduotis nurodoma kiek kartų uždavinys paleidžiamas. Klaidų pranešimai naudotojui rodomi tuo pačiu originaliu pavidalu (pav. 2), kaip ir dirbant konsolės aplinkoje. Paveikslėlyje pateiktas klaidos pranešimas gautas, nes gridcom prieš kelis mėnesius tapo uždara aplinka su priėjimu tik autorizuotiems naudotojams. Demonstracinė šios sistemos versija šiuo metu išjungta.

2 pav. Internetinė *„gridcom“* vartotojo sąsaja

Pagrindinis šio valdymo įrankio privalumas yra tai, kad jis padeda atlikti esmines užduočių valdymo operacijas paprastai, greitai ir autonomiškai. Įrankis centralizuotas, uždaro kodo ir sukurtas Vilniaus universitete. Naudotojas privalo įkelti per internetinę sąsają (HTTPS protokolu) ir patikėti savo slaptą raktą, slaptažodį ir sertifikatą prisijungimui prie Grid.

1.3.3. Valdymo įrankis „gEclipse“

Siekiant sumažinti naudotojo pusėje gLite UI programinės įrangos kiekį ir supaprastinti instaliavimą galima naudotis *gEclipse* (http://www.geclipse.org) įrankiu arba jo kodo fragmentais (gEclipse yra atviro kodo produktas). Šiuo metu gEclipse uždavinių valdymo funkcijos BalticGrid infrastruktūroje neveikia. Tai įrankis, kurį galima vadinti karkasu Grid valdymui suvienodinti. Pagrindinis gEclipse privalumas yra įvairių *middleware* palaikymas (gLite, UNICORE, Globus toolkit) ir jų technologijų integravimas į vieną valdymo metodiką. Įrankis realizuotas Java programavimo kalba ir parsiunčiamas į naudotojo kompiuterį kaip programa. Valdymo sąsaja realizuota grafinių dialogų principu ir interaktyviai valdoma naudotojo (pav. 3).



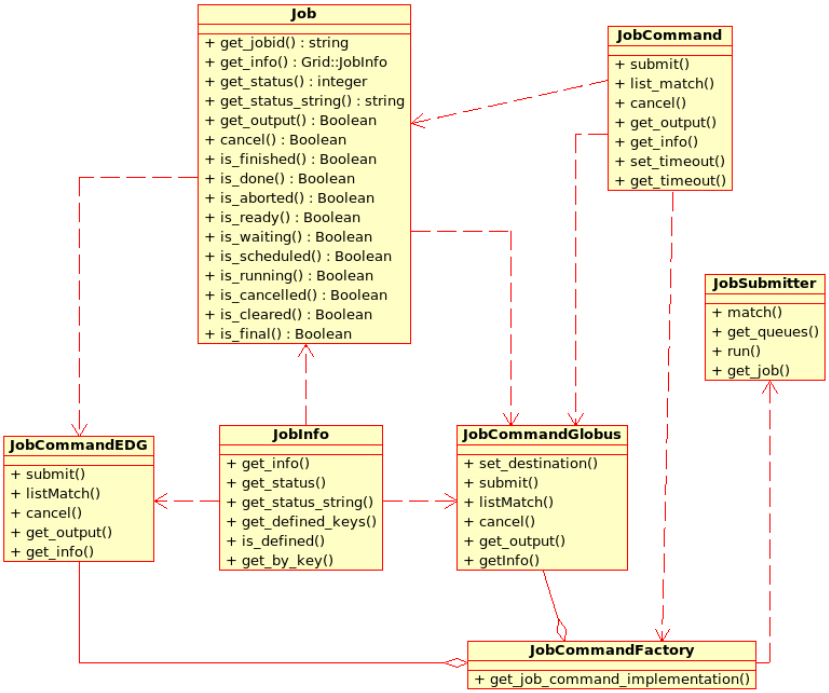
3 pav. Grafinė *„gEclipse“* vartotojo sąsaja

Nebuvo galimybės išbandyti uždavinių išsiuntimo ir kitų su tuo susijusių funkcijų dėl suderinamumo klaidos gEclipse bendraujant su *WMS*. Įrankis gali veikti visiškai decentralizuotai bendraudamas tiesiogiai su standartinius Grid servisais. gEclipse projektuotojų pagrindiniai tikslai, kuriant šią sąsają:

- Naudotojai galės pasiekti Grid resursus standartizuota, bet vartotojui draugiška ir pritaikoma sąsaja.
- Resursų teikėjai galės sumažinti operacijų išlaidas skiriamas naudotojų palaikymui, kai Grid naudojimas taps paprastesnis.
- Programų kūrėjai galės paspartinti Grid programų kūrimo ciklą.

1.3.4. Įrankių rinkinys „Zeus Grid toolkit“

Zeus Grid toolkit pagrindinis tikslas atskirti naudotojo programinį kodą nuo *middleware* įrankių. Šis paketas – tai įrankių rinkinys, kuris atlieka sąsajos vaidmenį tarp programos ir *middleware*. Naudotojui suteikiama galimybė naudoti tuos pačius programinius skriptus su skirtingomis *middleware*. Taip pat suteikiami baziniai įrankiai vienodam duomenų valdymui tarp skirtingų LCG-2 (gLite pirmtako) versijų. Klaidų atveju šie įrankiai suteikia galimybe naudotojui pakartoti nepasisekusių užduočių vykdymą. Suteikiama galimybė aprašyti užduotis, stebėti jų vykdymą, atnaujinti rezultatus ir baigti darbą. Šis įrankis dirba tik su edg-job-* komandomis, kurių dauguma gLite nebenaudojamos. Realizacijai naudota PERL programavimo kalba. Projekto užduočių valdymo klasių diagrama pateikta žemiau (pav. 4).



4 pav. Valdymo įrankių *„Zeus Grid Toolkit“* klasė

Užduočių valdymas Zeus Grid toolkit visiškai autonominis:

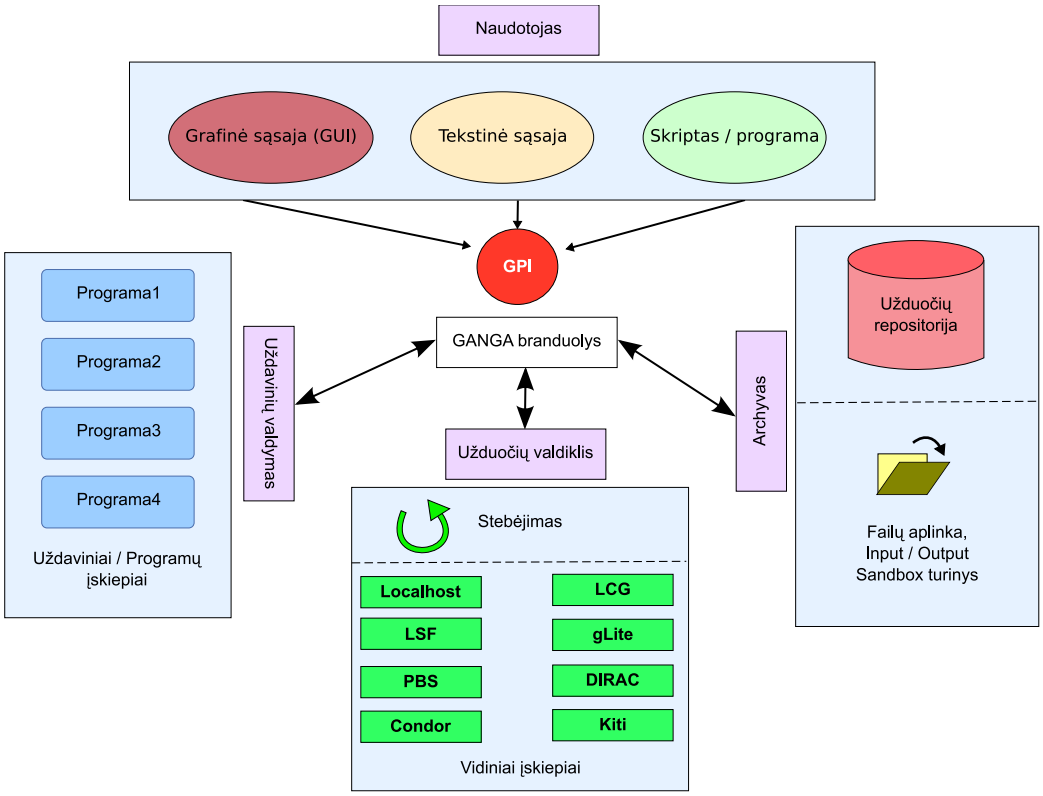
```
./jdl.pl --d 7 --vo <your_vo>     # užduočių išsiuntimas
./jobq.pl --all                   # eilių būklės peržiūra
./jobkill.pl 1                    # užduoties nutraukimas
```

Valdymui (įskaitant duomenų valdymą) naudojama apie 20 komandų. Įrankis decentralizuotas ir remiasi UI instaliuota programine įranga (LCG-2). Perrašius valdymo komandas ir atlikus struktūrinius pakeitimus tokį įrankį įmanoma pritaikyti gLite, nes tai atviro kodo programa. Panašius įrankius labai patogu naudoti programiniuose skriptuose.

1.3.5. Užduočių valdymo aplinka „GANGA“

GANGA suteikia vartotojui galimybę lengvai persijungti tarp darbų testavimo lokalioje sistemoje ir užduočių išsiuntimo plačiu mastu analizei į Grid, paslepiant Grid technines detales

[9]. GANGA buvo sukurta CERN tyrimų centre ir pritaikyta ten vykdomiems eksperimentams: Atlas ir LHC*b*. GANGA suteikia vartotojui galimybę greitai persijungti tarp uždavinių testavimo lokalioje eilių sistemoje ir siuntimo į plataus mąsto skaičiavimų aplinką apjungtą Grid. Šis įrankis padeda naudotojui mažiau rūpintis užduočių vykdymo eiga ir stabilumo problemomis. GANGA teikia objektinį komandų ir užduočių valdymo mechanizmą. Šios sistemos architektūra objektiškai išskaidyta į atskirus komponentus ir servisus (pav. 5).



5 pav. Valdymo priemonių karkasas GANGA

Vartotojo sąsaja realizuota interaktyviu tekstiniu režimu. Yra galimybė naudoti GANGA tiesiogiai perduodant reikiamus argumentus programai. Užduočių skaidymui realizuotos šešios funkcijos:

GenericSplitter – užduočių skaidymas su kintamu argumentu;

GaussSplitter – specifinių „Gauss“ užduočių skaidymas, nedokumentuotas.

OptionsFileSplitter – nedokumentuotas;

SplitByFiles – nedokumentuotas;

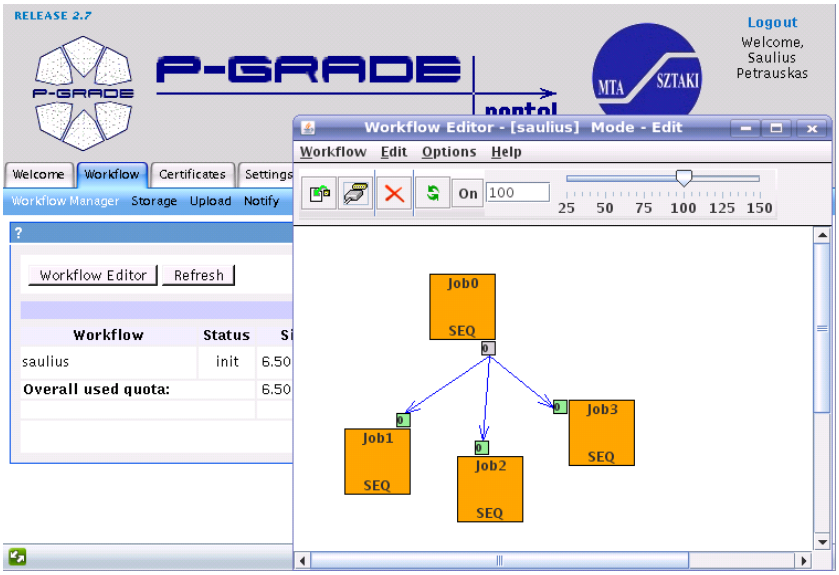
ArgSplitter – skaidymas pagal argumentų skaičių;

DiracSplitter – užklausiamas LFC (LCG File Catalog) failų meta-informacijos katalogas, kad būtų atliktas optimalus užduočių suskaidymas pagal duomenų failų grupavimą.

Konfigūraciniai šablonai gali būti išsaugomi ir automatiškai konvertuojami pagal skaičiavimo aplinkas (pvz.: PBS, Condor-G ir kt.) [4], o pati valdymo konsolė tinka naudojimui programiniuose skriptuose. GANGA realizuota PYTHON programavimo kalba.

1.3.6. Valdymo portalas „P-GRADE“

P-GRADE Grid portalas yra internetinio portalo principu veikianti aplinka, skirta uždavinių vystymui, vykdymui ir priežiūrai. Portalas suteikia naudotojui galimybę kurti darbų sekos grafus, kuriuos galima vykdyti įvairiose Grid platformose. P-GRADE portalas oficialiai palaikomas BalticGrid ir tarnauja regioninei naudotojų bendruomenei. Šis įrankis tinkamas mokytis dirbti Grid aplinkoje ir pademonstruoti šios aplinkos galimybes bei paskirstytų programų kūrimo aspektus. Portalas puikiai suderintas su gLite *middleware*. Ši aplinka turi labai patogų grafinį užduočių sekų redaktorių (pav. 6). Kadangi tai portalo principu realizuota aplinka ji yra centralizuota. P-GRADE parašytas JAVA programavimo kalba. Įrankis pats suformuoja JDL šablonus DAG užduotims (žr. 1.2 skyrelį), kurios siunčiamos į Grid. Dėl šios priežasties užduočių valdymas tampa daug paprastesnis: naudotojui nebereikia aprašinėti DAG užduočių (tai reikalauja gilių techninių žinių ir kvalifikacijos), nebereikalinga priežiūra tarpinių užduočių vykdymui ir nereikia rūpintis kokioje būsenoje jas galima paleisti į Grid [23]. Portale realizuota pranešimo elektroniniu paštu apie uždavinio vykdymo būsenos pasikeitimą (pvz.: pabaigos) funkcija.



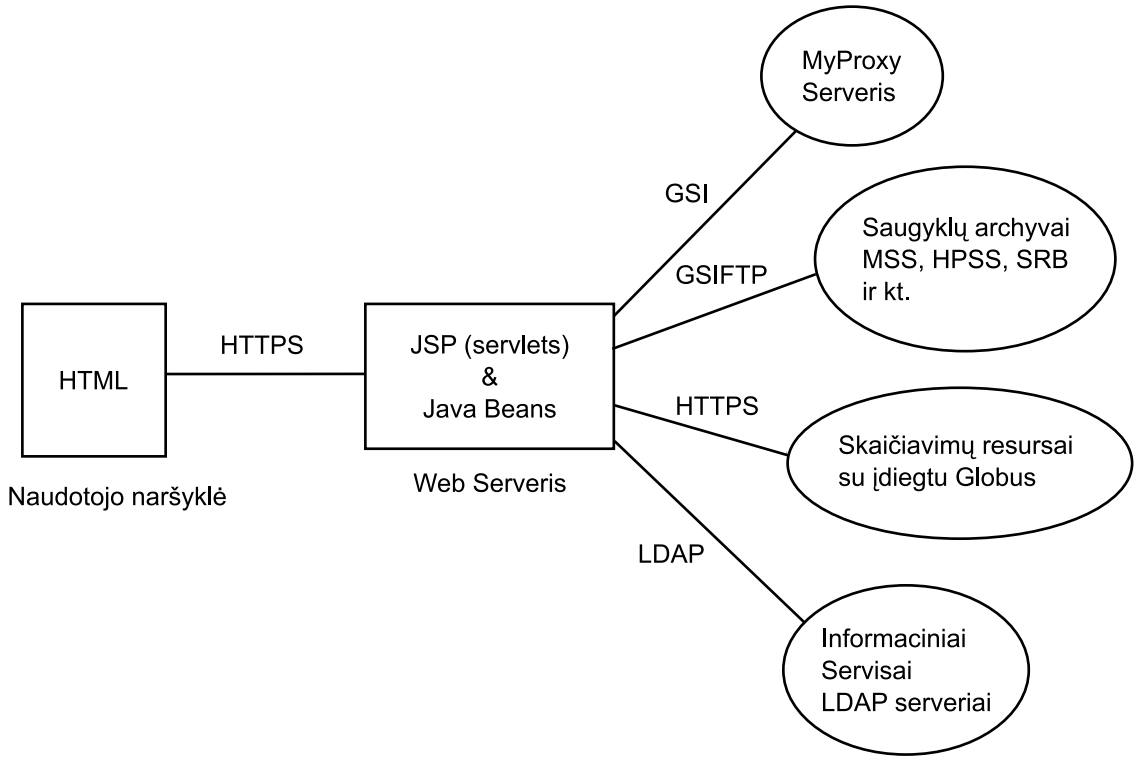
6 pav. Portalo „*P-GRADE*“ vartotojo sąsaja

Portalas suteikia naudotojui abstraktų ir homogenišką visos Grid infrastruktūros vaizdą. Žemo sisteminio lygio Grid valdymo mechanizmai paslepiami po grafine sąsaja. Todėl net ir nepatyrę vartotojai gali formuoti ir vykdyti paskirstytas programas heterogeninėje skaičiavimų aplinkoje. P-GRADE jau dabar palaiko kelias skirtingas *middleware* versijas. Užduočių sekas ir jų aprašus portalo viduje galima naudoti nepriklausomai nuo *middleware* įrangos. Tai suteikia naudotojams galimybę pasiekti kelias Grid sistemas vienu metu ir dar plačiau paskirstyti

uždavinius per kelias platformas. Sėkmingai įvykdyti užduočių grafą šiuo įrankiu nepavyko dėl klaidos susijusios su MyProxy delegavimu.

1.3.7. Įrankių rinkinys „GPDK“

Įrankių rinkinys *GPDK* skirtas naudotojams, kurie nori valdyti užduotis iš kelių kompiuterių. Tai centralizuota paslauga, kurią galima įkelti vykdymui Tomcat serveryje kaip WAR (Web programs ARchive). Tai gali būti patogu pernešant programų rinkinį tarp kompiuterių. Sistema naudoja nutolusį MyProxy serverį naudotojo autorizavimui. Šis serveris atsakingas už proxy sertifikato pratęsimą ir delegavimą. Inicijavus MyProxy naudotojas gali saugiai pasidėti raktus ir slaptažodžius kitoje vietoje, o priėjimo prie resursų autorizaciją atliks MyProxy servisas. GPDK dar nėra išbaigtas produktas, o tik komponentų rinkinys, kurį galės panaudoti specializuotų portalų kūrėjai [18]. Jo principinė veikimo schema pateikta žemiau (pav. 7). Iš paveikslo matome, kad sistemai reikalingas TomCat serveris, kurį galima paleisti kartu su esamu serveriu Apache ar Microsoft IIS.



7 pav. Portalo *„GPDK“* architektūra

Įrankis naudoja JSP (Java Server Pages) ir Java Beans. Naudotojui arba sistemos administratoriui dar tektų pasirūpinti LDAP arba DBMS informacinės sistemos saugumu, kurią naudoja Java Beans komponentai naudotojo profilių ir įgaliojimų informacijai saugoti. Realizacijai naudota JAVA programavimo kalba.

1.3.8. Valdymo įrankių apibendrinimas

Atlikus išsamią užduočių valdymo įrankių analizę, galima aktualias tiriamai sričiai savybes susisteminti ir pateikti lentelės pavidalu (žr. 1 lent.). Svarbiausiomis savybėmis parinktos: įrankio galimybė autonomiškai veikti, be naudotojo įsikišimo ir galimybė būti visiškai nepriklausomam nuo tarpinių centrų ir servisų (išskyrus standartinius Grid palaikomus servisus kaip *WMS*). Informatyvumo tikslu lentelėje dar pateikta grafinės sąsajos galimybė.

1 lentelė. Valdymo įrankių savybės

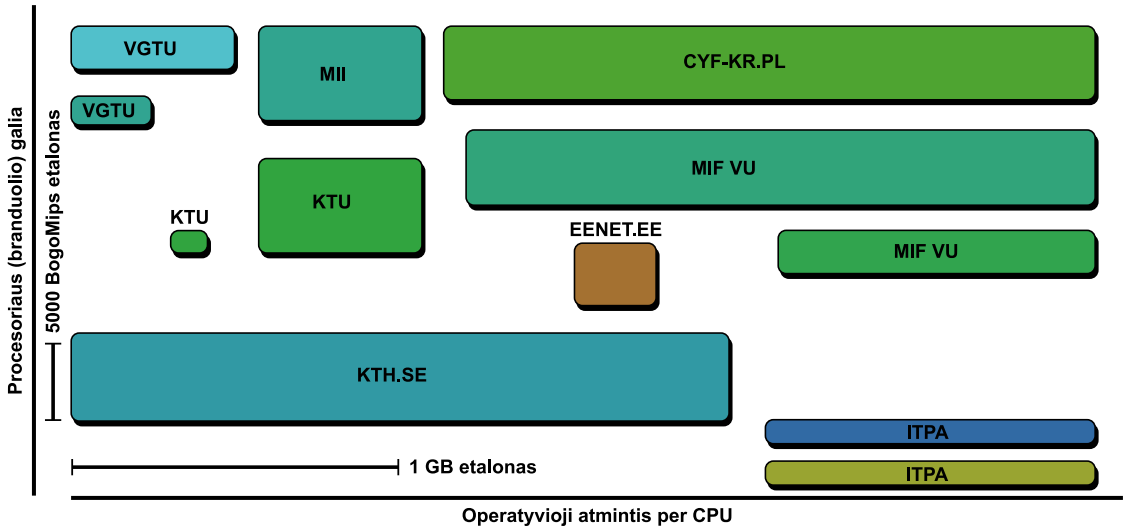
Pavadinimas	**Decentralizuotas**	**Autonominis**	**Turi grafinę sąsają**
Migrating Desktop	−	±	+
gridcom	−	±	+
gEclipse	+	+	+
Zeus Grid toolkit	+	−	−
GANGA	+	+	−
P-GRADE	−	±	+
GPDK	−	−	+

Pagal pasirinktus įvertinimus matome, kad decentralizuotų ir autonominių įrankių kriterijus geriausiai atitinka gEclipse ir GANGA. Pagrindinis skirtumas tarp šių įrankių, tai grafinės sąsajos tipas. gEclipse naudoja GUI (Grafical User Inferface) tipo sąsają, o GANGA pati savaime turi tekstinę sąsają, tačiau turi galimybę būti naudojama iš aukštesnio lygio sąsajų ir programinių skriptų.

1.4. Infrastruktūros analizė

BalticGrid infrastruktūroje daugiausia naudojami Intel/AMD procesoriai, tačiau yra naudojami ir kitų tipų procesoriai. Dalis į Grid įjungtų telkinių sudaryti iš mažai kur kitur naudotojams prieinamos aparatūros (pvz.: Itanium2 procesorių). Naudotojui būtinas trumpas ir nuspėjamas uždavinio vykdymo ciklas, arba interaktyvus priėjimas prie resursų, kad jis galėtų kurti ir derinti savo uždavinius. Interaktyviam priėjimui realizuoti tinkamos Globus sąsają naudojančios programos kaip *glogin*.

Gali būti sunku įsivaizduoti kaip visa heterogeninė skaičiavimų aplinka atrodo. Pagrindinis parametras nuo kurio priklauso užduočių vykdymas yra laikas. Pagrindiniai kintamieji, kurie įtakoja užduoties vykdymo laiką jai startavus – procesoriaus greitis ir operatyviosios atminties kiekis. Taigi šie du kintamieji labiausiai apsprendžia resursų rūšį. Šiame darbe nagrinėjamos tarpusavyje nesusietos užduotys, o tai reiškia, kad nekreipsime dėmesio į tarpusavio komunikavimo tarp užduočių vėlinimą, kuris taip pat labai įtakotų vykdymo laiką. Turėdami šias dvi dimensijas: procesoriaus greitį (tiksliau jo branduolio, nes šiuo metu procesoriai dažniausiai turi po kelis branduolius) ir atminties kiekį galime šias resurso ypatybes pavaizduoti koordinačių plokštumoje (pav. 8).



8 pav. Heterogeniniai resursai BalticGrid aplinkoje

Matome, kad kai kurie BalticGrid infrastruktūros partneriai turi po du telkinius. Dažniausiai taip yra todėl, kad viename telkinyje resursai paprastai būna homogeniški. Taip lengviau išlaikyti resursų informacinės sistemos vientisumą, o valdymo įrankiams lengviau paskirstyti užduotis.

Iš aukščiau pateiktos diagramos matome, kad resursai labai skirtingi. Tikslius duomenis naudotus šioje diagramoje galima rasti prieduose (žr. 3 lent.). Vienas KTU telkinys (KTU-BG-GLITE) dar tebenaudoja Pentium III procesorius ir turi tik 128 MB operatyviosios atminties, todėl diagramoje pavaizduotas labai mažas. Tokie resursai vis tiek yra naudingi: juos galima panaudoti uždavinių testavimui. Be to, diagramoje neįvertintas mazgų skaičius, kad aiškiau matytųsi kokioje aplinkoje vykdoma kiekviena užduotis. Įvedus trečią – laiko dimensiją, galima būtų stebėti tiriamos aplinkos pokyčius laike. Tam reikėtų reguliariai apklausinėti visus telkinius ir surinkti resursų charakteristikas arba pamėginti išgauti tokius duomenis iš monitoringo sistemų.

Kai kurie partneriai perka mažiau, bet galingų serverių ar kompiuterių, kuriuos jungia į bendrą tinklą. Tai priklauso nuo kiekvienos institucijos vidinės politikos ir išteklių, todėl tokiuose tinkluose tam tikras heterogeniškumo lygis visada išliks. Kaip tokią aplinką efektyviai panaudoti skaitykite 1.5 skyrelyje.

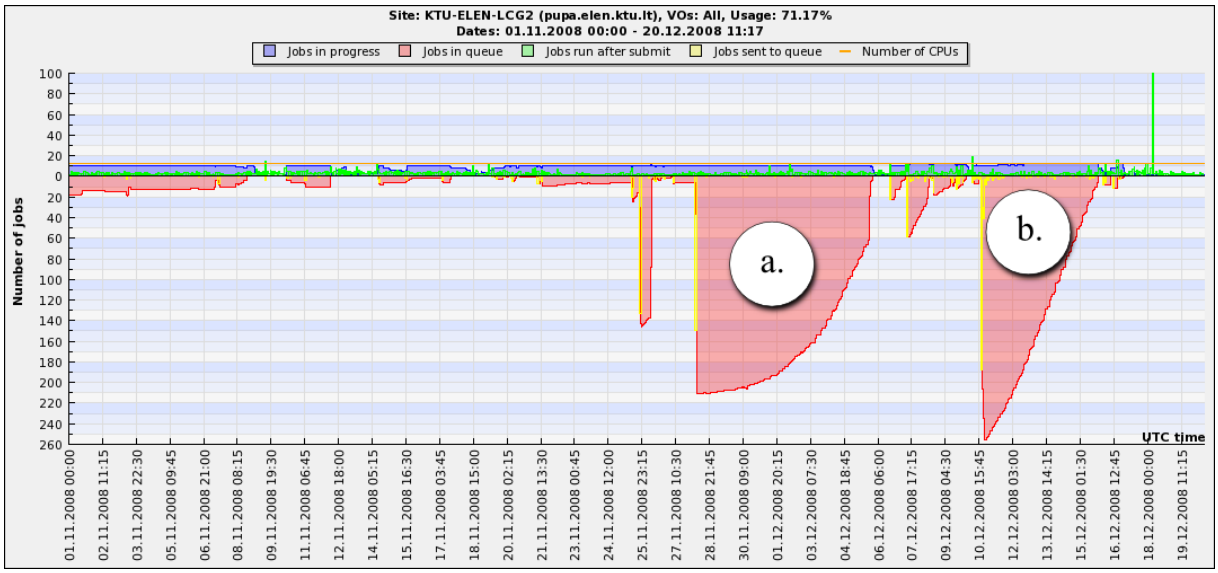
1.4.1. Apribojimai

1.4.2. Telkinių valdymas

Šio darbo kontekste kompiuterių telkinys arba klasteris – tai grupė sujungtų serverių ar kompiuterių, kurie dirba kartu labai glaudžiai. Todėl daugeliu atžvilgių ši panašių kompiuterių grupė gali būti traktuojama kaip vienas kompiuteris. Telkiniai dažniausiai apjungiami į tinklus. Telkiniai paprastai skirstomi į skaičiavimų greitį spartinančius, patikimumą palaikančius

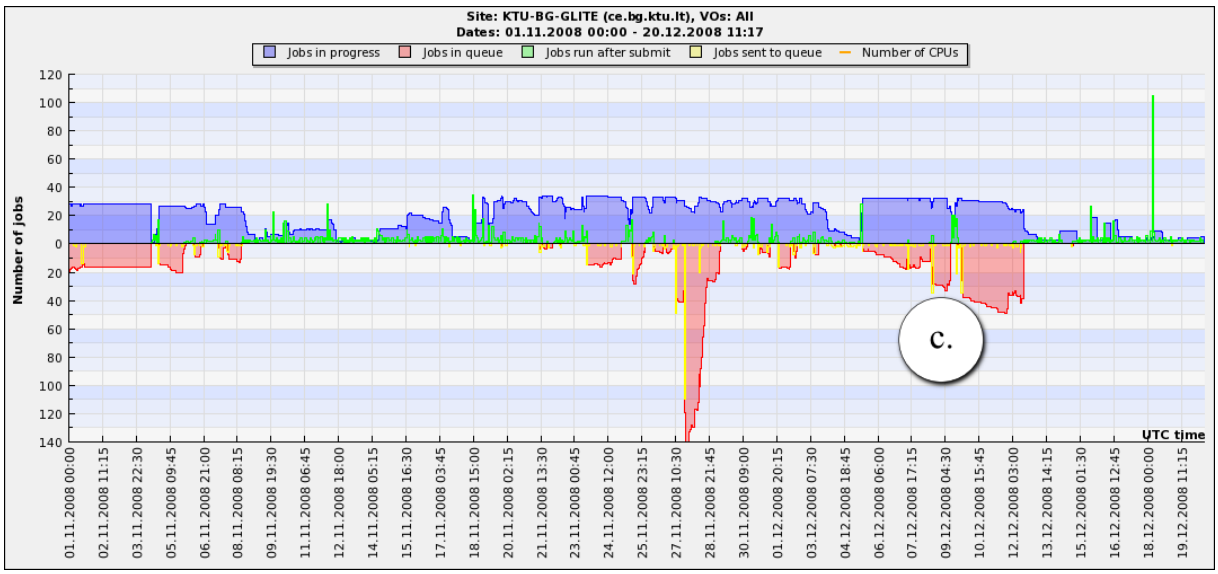
arba didelio kiekio duomenų saugojimo funkciją atliekančius. Gali būti ir kelių savybių junginys. Kainos prasme toks kompiuteris yra žymiai naudingesnis ir efektyvesnis už pavienius kompiuterius lyginant greitį ir patikimumą.

Atliekant užduočių valdymo tyrimą, pravartu išanalizuoti realią eilių vykdymo statistinę informaciją. Tam labai patogu pasinaudoti *BAT* įrankiu ir jo teikiama informacija [24] apie telkinius. Šio įrankio kliento dalį įdiegę nemaža dalis BalticGrid infrastruktūroje veikiančių partnerių. Pagal šio įrankio pateikiamą informaciją lengvai pastebimos augančios ir senkančios užduočių eilės telkiniuose (pažymėtos a., b., c.) diagramose (pav. 10) ir (pav. 9).



9 pav. Eilių statistinė informacija *(KTU-BG-GLITE telkinys)*

Pagal nutylėjimą paleistos užduotys vykdomos 12 valandų, nebent naudotojas atskirai nurodo *VOMS proxy* (leidimą tam tikram laikui naudotis VO resursais Grid'e) galiojimo laiką (kuris gali būti iki 96 valandų). Kaip matosi pažymėtoje vietoje *c.* telkinys buvo maksimaliai apkrautas 5 dienas ir užduočių eilėje susikaupė daugiau nei yra skaičiavimo mazgų šiame telkinyje. Taip pat matome, kad tos sukauptos užduotys tikriausiai buvo nutrauktos, nes viršutinėje koordinačių ašies pusėje nesimato, kad eilė „išsektų". Gali būti, kad baigėsi naudotojo *VOMS* proxy galiojimo laikas užduotims arba telkinio administratorius, pamatęs, kad eilė užstrigo ir neišsenka, užduotis nutraukė. Užduotys gali patekti į tokią būseną, kai informacinė sistema praneša, kad jos vykdomos, nors iš tiesų jų vykdymas nutrūkęs.



10 pav. Eilių statistinė informacija *(KTU-ELEN-LCG telkinys)*

Aukščiau pateiktame paveikslėlyje (pav. 9) matomas priešingas reiškinys: per labai trumpą laiką susikaupė eilė užduočių (*a.*, *b.*) ir per keletą dienų ji tolygiai išsenka. Paprastai tokie reiškiniai atsiranda dėl kelių priežasčių:

- naudotojai vykdo labai intensyvius skaičiavimus;
- sutriko eilių valdymo sistema (PBS) telkinyje – lokali problema;
- naudotojo autonominiai užduočių valdymo įrankiai veikia neefektyviai;
- *WMS* valdymo sistema netolygiai paskirsto resursus;
- informacinės sistemos (BDII) pranešimai apie laisvus resursus vėluoja;
- laike kintančios informacijos nesutapimai dėl laikrodžių fliuktuacijos;
- įvairių kitų priežasčių (pvz.: telkinių dalyvavimas keliose VO).

Dėl aukščiau paminėtų priežasčių naudotojų įrankiai atliekantys autonominį užduočių valdymą turi turėti „saugiklius“, kad užtikrintų galimybę nutraukti perteklines užduotis ir nesudarytu didžiulių eilių godžiai rezervuodami resursus. Kartais ir mažas pakeitimas užduočių valdyme gali sukelti drugelio efektą (angl. *Butterfly Effect*) [28]. Tai reiškinys, kai nedideli pakeitimai vienoje sistemoje gali sukelti daug platesnio masto pokyčius kitoje sistemoje ar sistemose.

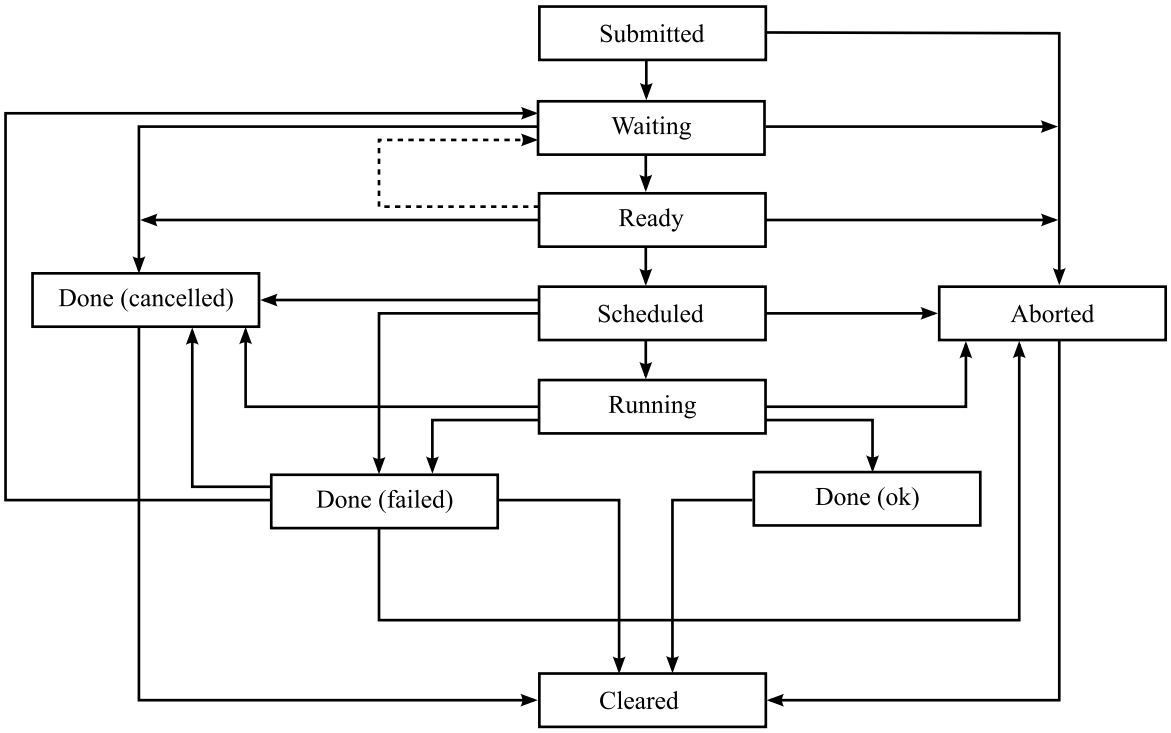
1.4.3. Infrastruktūros patikimumas

Kaip paminėta skyrelyje 1.4.2, telkinių administratoriai turi teisę nutraukti užduotis, jei jų vykdymas užstrigęs ir rezultatai greičiausiai nebus gaunami. Heterogeninė skaičiavimų aplinka ir pati savaime nuolat keičiasi: resursai išjungiami, atnaujinami, vėl įjungiami atskirais

skaičiavimų mazgais ir net telkiniais. Resursai sudaromi iš skirtingose šalyse esančių institucijų resursų, kuriems galioja tiek bendros tinklo naudojimo taisyklės, tiek vidinė politika. Dėl šių priežasčių heterogeninė aplinka negali būti laikoma patikima. Tai nėra jos trūkumas, tai tiesiog tokios aplinkos natūrali savybė. Šią savybę galima pamatuoti įvertinant atskirų telkinių ar net visos infrastruktūros QoS parametrus, kurie plačiau išnagrinėti [25] darbe. Nepatikimumo savybę reikia įvertinti kuriant metodikas ir įrankius, kurie veiks šioje aplinkoje.

Užduočių valdymo programinė įranga iš *middleware* pusės pritaikyta naudotojui dirbti tiesioginiu dialogu. Dėl šios priežasties sisteminiai pranešimai ir standartinių įrankių komandos nelabai tinkamos tiesiogiai naudoti meta-valdymo įrankiams. Pateikiama informacija naudotojui dažniausiai yra tekstinio pavidalo, o pateikimo formatas ne visada vienodas. Taigi meta-valdymo įrankių patikimumas priklauso nuo gebėjimo tinkamai reaguoti į nenumatytus pranešimus ir situacijas.

Nagrinėjant užduočių valdymą būtina išsiaiškinti kokiose būsenose vykdoma užduotis gali būti. Žemiau pateikiama užduočių vykdymo būsenų diagrama (pav. 11) sudaryta pagal gLite telkinių administratorių techninę dokumentaciją.



11 pav. Užduoties vykdymo būsenų diagrama

Žemiau pateikiamas trumpas būsenų, kuriose gali būti užduotis, reikšmių aprašymas:

Submitted užduotis jau išsiųsta vartotojo, užklausa užregistruota *WMS* WMProxy serviso, bet dar nepriimta;

Waiting užduotis perkelta į *WMS* ir priimta atrinkimui pagal reikalavimus aprašytus JDL, ir laukia kol *WMS* WM (Workload Manager) ją apdoros;

Ready užduotis jau apdorota *WMS* WM, bet dar neperkelta į telkinio CE;

Scheduled užduotis jau perkelta į CE ir laukia jo eilėje;

Running užduotis vykdoma WN elemente;

Done užduotis baigė darbą;

Aborted užduoties vykdymas buvo nutrauktas *WMS*;

Canceled užduotis nutraukta naudotojo pareikalavimu;

Cleared rezultatai buvo persiųsti naudotojui arba ištrinti jų laikymo trukmei pasibaigus.

Iš šios (pav. 11) diagramos matome, kad užduoties vykdymo procesas turi gana daug vykdymo šakų. Svarbiausios iš būsenų atliekant užduočių valdymą yra „Scheduled“, „Done“ ir „Cleared“. Šiame darbe pristatomos metodikos atveju (žr. 2 skyrelį), būtent šios būsenos apsprendžia kokius veiksmus toliau atlikti. Diagramoje pavaizduotos dvi situacijos: Done(failed) → Waiting ir Ready → Waiting, dėl kurių galėtų susidaryti ciklai. Tokios situacijos apribotos *middleware* pagal tai kiek kartų galima kartoti kiekvieną užduotį. Nepavykus užduočiai naudojamas *RetryCount* ir pati *middleware* bando užduotį įvykdyti iš naujo kol nepasiekiamas *RetryCountMax*. Iš *Ready* būsenos grįžta užduotys, kurias siunčiant iš *WMS* į telkinį dėl kažkokių priežasčių siuntimas nepavyko.

1.5. Uždavinių apribojimų analizė

Atliekant valdymo metodikų tyrimą natūraliai susiduriama su naudotojo uždavinių dalykine sritimi. Efektyvus užduočių valdymas priklauso ne tik nuo valdymo priemonių ir metodikos, bet ir pačių uždavinių apribojimų ir struktūros. Gali būti tokių uždavinių, kurių visai nepavyks suskaidyti neatlikus re-inžinerijos.

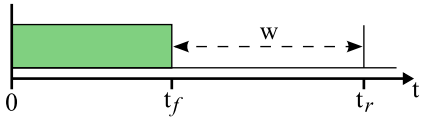
Uždaviniai, kurie paprastai sprendžiami pasitelkiant Grid resursus savo imtimi yra didesni už infrastruktūros pajėgumus perrinkti visus variantus. Todėl uždaviniai projektuojami arba pritaikomi tam tikriems apribojimams. Tokie apribojimai gali būti kelių tipų:

- ribojamas iteracijų skaičius;
- ribojamas rezultato tikslumas;
- ribojamas skaičiavimų laikas;
- dinamiškai apsprendžiamas vykdymo apribojimas.

Išnagrinėsime, kurie uždavinių apribojimo tipai geriausiai tinka vykdant juos heterogeninėje aplinkoje. Reikėtų pažymėti, kad kiekvienos užduoties vykdymą apriboja likęs naudotojo *VOMS* proxy galiojimo laikas ir resursų apribojimai WN mazge (nustatomi administratorių *ulimit* pagalba). Užduotis gali būti vykdoma, kol suveiks vienas iš paminėtų apribojimų. Po to ji automatiškai nutraukiama. Vykdymo laikas kiekvienai užduočiai bus *skirtingas*, nes užduotys

eilėje lauks skirtingus laiko intervalus, o šis laikas yra įskaitomas į bendrą užduoties vykdymo ciklo laiką.

Apribojant užduoties vykdymą pagal iteracijų skaičių galimi keli scenarijai. Pirmasis, kad užduoties iteracijų skaičius pakankamai mažas, ir skaičiavimo laikas t_f visada tilps į vykdymo laiką t_r. Tokiu atveju rezultatai bus gaunami, tačiau skaičiavimams skirtas laikas panaudojamas neefektyviai (pav. 12), nes daugiau laiko bus sugaištama komunikacijoms: užduočių siuntimams ir rezultatų grąžinimui. Be to, skaičiuojant greituose telkiniuose užduotys baigs darbą daug anksčiau nei lėtuose, todėl dalies uždavinio rezultatų teks laukti. Antrasis scenarijus, kai iteracijų apribojimas pritaikytas greitiems telkiniams. Tada skaičiavimai lėtuose telkiniuose pagal tą patį iteracijų skaičių užtruks ilgiau ir viršys skaičiavimams skirtą laiką per τ, nei buvo numatyta (pav. 13), ko pasekoje rezultatai iš šių telkinių bus prarasti. Visas laikas t_r ir resursai, kuriuos užduotis buvo užėmusi lėtame telkinyje, bus iššvaistyti. Toks apribojimas pagal iteracijas buvo taikomas 3.4 skyrelyje aprašytame uždavinyje ir dalinai pasiteisindavo su mažais schemų modeliais, tačiau tapo visiškai netinkamas didėjant uždavinio imčiai ir kiekvienai iteracijai užtrunkant vis ilgiau.

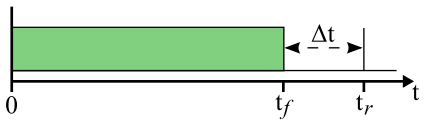


12 pav. Nepanaudotas laikas skaičiavimams

Ribojant rezultato tikslumą iškyla panašios problemos kaip ir apribojant vykdymą pagal iteracijas. Kol tikslumas pasiekiamas per t_r laiką, užduotys bus vykdomos sėkmingai. Tačiau egzistuoja ta pati resursų švaistymo problema, kai tikslumas nepasiekiamas. Be to gali būti sunku nuspręsti ar tikslumas jau pasiektas visam uždaviniui, nes užduotys vykdymo metu neturi pilnos informacijos apie kitų dalių pasiektą rezultatą.

13 pav. Viršytas skaičiavimams skirtas laikas

Skaičiavimų laiko apribojimas reiškia, kad užduotis skaičiuos tiek, kiek liko laiko, nuo tada kai pradėjo vykdymą. Galima palikti atsarginį laiko rezervą užduoties užbaigimui ir rezultatų failų grąžinimui, kol nepasibaigė naudotojo *VOMS* proxy galiojimo laikas. Vykdant užduotis su tokiu apribojimu optimalus skaičiavimų laiko panaudojimas vyksta kai $\Delta t = t_r - t_f$ kuo mažesnis: $\Delta t \longrightarrow 0$ (užduotį sėkmingai įvykdyta laikome kai $t_f \leq t_r$) (pav. 14). Šis apribojimas turi keletą privalumų: užduotys lanksčiai prisitaiko prie heterogeninių resursų galimybių ir rezultatai iš visų telkinių grįžta beveik vienu metu su nedidele fliuktuacija dėl Δt ir komunikacijų greičio skirtumų Grid'e.



14 pav. Optimalus skaičiavimų laiko panaudojimas

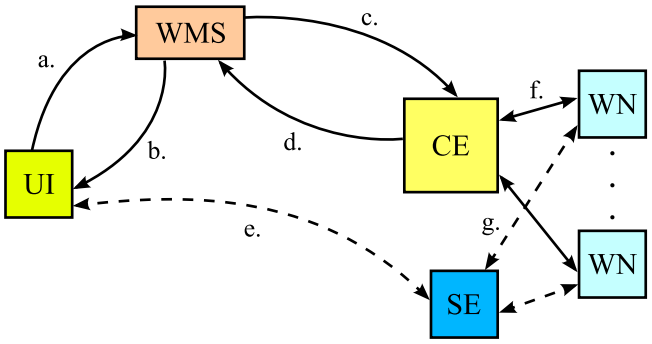
Dinaminiai apribojimai gali būti taikomi uždaviniams, kurių nepavyksta suskaidyti remiantis laiko intervalais. Dalis užduočių gali būti ilgos ir išnaudoti visą laiko rezervą, kitos dėl dinaminių apribojimu darbą baigs anksčiau. Toks suskaidymas gali būti pakankamai efektyvus, jei trumposios užduotys tolygiai užpildys nepanaudotą skaičiavimų laiką (pav. 12), kol galioja naudotojo *VOMS* proxy sertifikatas. Tačiau vis tiek atsiras laiko praradimai, kurį užtruks užduoties išsiuntimas ir rezultatų grįžimas.

Apibendrinant galima teigti, kad užduočių skaičiavimo laiko apribojimas yra efektyviausias.

1.6. Tipinis užduoties vykdymo modelis

Norėdami išsiaiškinti kaip užduotis paskiriama į vieną ar kitą telkinį ir ten įvykdoma, išanalizuokime tipinį vienos užduoties vykdymo modelį. Grid aplinkoje naudojami specialūs sutartinai sutrumpinimai elementams žymėti, jų aprašymas pateiktas terminų žodynėlyje.

Žemiau pateiktoje diagramoje (pav. 15) sudarytas tipinis telkinio modelis, kuriame yra šie elementai: CE, SE ir WN$\times n$. SE dalyvavimas vykdant užduotį nėra privalomas ir priklauso nuo uždavinio.



15 pav. Tipinis užduoties vykdymo modelis

Pavaizduotas modelis gali būti paaiškintas tipiniu užduoties vykdymo scenarijumi:

1. Naudotojas (arba programa) prisijungia prie UI elemento ssh kliento pagalba.
2. Užduotis pagal aprašo failą siunčiama į *WMS* ($UI \rightarrow a. \rightarrow WMS$).
3. *WMS* pagal vidinį atrankos algoritmą suranda tinkamiausią telkinį pagal skaičiavimo elemento (CE) informaciją, gaunamą iš informacinės sistemos (site-BDII), ir siunčia į CE užduoties apraše nurodytus failus, programinį kodą bei paleidimo metu vykdomas komandas ($WMS \rightarrow c. \rightarrow CE$).

4. CE parenka laisvą darbinį elementą WN ir siunčia užduotį į jį ($CE \rightarrow f. \rightarrow WN$). Jei laisvo WN nėra, užduotis pastatoma į eilę.

5. Užduotis paleidžiama WN elemente.

6. *WMS* nuolat tikrina užduoties vykdymo būklę ($WMS \rightarrow c. \rightarrow CE \rightarrow d. \rightarrow WMS$) ir, naudotojui pareikalavus, ją praneša ($WMS \rightarrow b. \rightarrow UI$).

7. Naudotojas kreipiasi į *WMS*, kad sužinotų užduoties vykdymo būklę ($UI \rightarrow a. \rightarrow WMS$).

8. Gavus informaciją apie užduoties baigimą $WN \rightarrow f. \rightarrow CE \rightarrow d. \rightarrow WMS$, CE vykdomas iš *WMS* gautas užduoties įpakavimo (angl. wrapper) skriptas, kuris inicijuoja rezultatų grąžinimą iš WN.

9. Kai *WMS* gauna rezultatus, užduoties būsena pakeičiama į „Done“ ir naudotojas gali parsisiųsti rezultatus iš *WMS* ($WMS \rightarrow b.UI$).

Jeigu uždavinys naudoja SE duomenims saugoti, naudotojas gali iš anksto įkelti reikiamus duomenų failus į SE $UI \rightarrow e. \rightarrow SE$. Kai užduotis bus paleista WN elemente, duomenis saugomus SE galima parsisiųsti į WN $SE \rightarrow g. \rightarrow WN$ arba pasiekti pasinaudojus API (Application programming interface) kreipiniais. Taip taupomas persiunčiamų per WMS duomenų srautas.

Šiame skyriuje išanalizavę užduočių valdymo įrankius ir aplinką galime padaryti šias išvadas:

- Naudotojai pasiekia Grid resursus standartizuota bei vartotojui draugiška ir pritaikoma sąsaja.

- Resursų teikėjai gali sumažinti operacijų išlaidas skiriamas naudotojų palaikymui, kai Grid naudojimas taps paprastesnis.

- Programų kūrėjams spartėja Grid programų kūrimo ciklas.

2. UŽDUOČIŲ VALDYMO METODOLOGIJA

2.1. Užduočių aprašymas ir šablonų kūrimas

Užduotys siunčiamos skaičiavimams į Grid aprašomos JDL failuose. Kiekviena individuali užduotis turi būti aprašoma atskirame faile, jei nenaudojami DAG, parametriniai ar rinkinio tipo aprašymai (žr. 1.2 skyrelį). Norint vykdyti užduotis Grid'e naudojant gLite *middleware* jas būtinai reikia aprašyti JDL kalba [14]. Bendra failo struktūra:

```
atributas1 = skaičius;
atributas2 = "eilutė";
atributas3 = { "eilutė1", "eilutė2" };
atributas4 = ( išraiška );
```

Užduočių valdymo komandas galima suskirstyti į grupes pagal jų atliekamas funkcijas. Skirtingus prefiksus (globus-*, edg-job-*, glite-job-*, glite-wms-job-*) turinčios komandos gali naudoti skirtingus API, todėl geriau naudoti komandas su vienodais prefiksais.

Užduoties aprašo tikrinimas

```
Komandos: edg-job-list-match, glite-job-list-match,
glite-wms-job-list-match
```

Paskirtis: šios grupės komandos parodo, kurie CE tenkina JDL faile aprašytos užduoties reikalavimus. Šalutinis efektas - patikrinama JDL failo sintaksė nesiunčiant užduoties į Grid'ą.

Užduoties gridtest.jdl sintaksės patikrinimas ir tinkamų CE sąrašo išvedimas:

```
glite-wms-job-list-match -a gridtest.jdl
```

Galima nurodyti RB/*WMS* serverį į kurį bus siunčiama užduotis, kaip tai padaryti skaitykite atitinkamų komandų man puslapiuose.

Užduoties gridtest.jdl tikrinimas nurodant *WMS* serverį:

```
glite-wms-job-list-match -a -e \
https://broker.eenet.ee:7443/glite_wms_wmproxy_server gridtest.jdl
```

Užduoties siuntimas vykdymui

Komandos: globus-job-submit, edg-job-submit, glite-job-submit, glite-wms-job-submit

Paskirtis: šios komandos skirtos užduočių siuntimui į Grid.

Testuojant ar kokiu kitu tikslu, galima nurodyti telkinį, į kurį norima siųsti užduotį. Jei telkinys nenurodytas, jį parenka *WMS* LB serveris. Komandos grąžina užduoties identifikatorių.

Užduoties gridtest.jdl siuntimas:

```
glite-wms-job-submit -a gridtest.jdl
```

Atributas	**Galimos reikšmės**	**Prasmė**
Type	Job	Failo paskirtis (*Užduotis*)
JobType	"Normal", "MPICH", "Interactive", "DAG", "Parametric", "Checkpointable"	Užduoties tipas
Executable	/kelias/programa, "programa"	Programa, kuri bus vykdoma
Arguments	arg arba "arg1 arg2"	Programos argumentai
Environment		Programos aplinkos kintamieji
StdInput	failo vardas	Failas, kuris bus perduotas programai vietoje įvedimo iš konsolės (stdin)
StdOutput	failo vardas	Kur bus saugomi programos į ekraną išvedami rezultatai (stdout)
StdError	failo vardas	Kur bus saugomi programos pranešimai apie klaidas (stderr)
InputSandbox	{ "failas1", "failas2" }	Programa ir jos duomenys, kurie persiunčiami kartu su užduotimi (iki 10 MB)
OutputSandbox	{ "failas1", "failas2" }	Failai, kurie grįžta su užduoties rezultatais
InputData	lfn:/... arba "guid:..."	Papildomi duomenys; padeda atrinkti arčiau WN esantį SE.
DataAccessProtocol	"gridftp", "rfio"	Protokolas, kuris bus naudojamas InputData duomenų persiuntimui
RetryCount	N	N - skaičius, kiek kartų bandyti pakartotinai paleisti užduotį, jei užduotis nutrūko (*Aborted*).
ShallowRetryCount	N	N - Bandymų paleisti užduotį skaičius, kai užduotis nepasiekia WN
Requirements	(loginė išraiška)	Reikalavimai, pagal kuriuos atrenkami užduoties vykdymui tinkami telkiniai.
Rank	(aritmetinė išraiška)	Išraiška, pagal kurią rūšiuojami reikalavimus atitinkantys telkiniai
NodeNumber	N	Norimas telkinio branduolių skaičius (MPICH užduotims, branduoliai išskiriami viename telkinyje)
Lrms_Type	PBS	*gLite* klaidos apėjimas (MPICH užduotims)
VirtualOrganisation	VO vardas	Virtualios organizacijos (VO) pavadinimas
MyProxyServer	myproxy_serveris	MyProxy serveris
Max_Nodes_Running	N	DAG užduoties maksimalus vienu metu veikiančių mazgų skaičius
Nodes		DAG užduoties grafo aprašymas
ListenerPort	N	*Interactive* užduočių serverio prievado numeris, į kurį jungiasi startavusi užduotis
Parameters	N	Galinis rėžis parametrinėms užduotims
ParameterStart	N	Pradinis rėžis parametrinėms užduotims
ParameterStep	N	Žingsnis parametrinėms užduotims

Lentelėje (žr. 2 lent.) pateikti galimi JDL atributai, pagal kuriuos bus kuriamas užduočių siuntimo šablonas. Naudotojas turės galimybę šabloną papildyti, pagal savo poreikius.

Žemiau pateikti uždavinio tipų aprašymai:

JobType = „Normal“

Būtini atributai: Executable, StdOutput, StdError, InputSandbox, OutputSandbox.

Tokios užduotys tinka nuoseklių programų vykdymui Grid'e.

job-normal.jdl:

```
Executable = "/bin/sh";
Arguments = "job-normal.sh";
StdOutput = "job-normal.out";
StdError = "job-normal.err";
InputSandbox = {"job-normal.sh"};
OutputSandbox = {"job-normal.err","job-normal.out"};
```

job-normal.sh:

```
#!/bin/sh
date
hostname
uname -a
gcc -v
uptime
id
```

JobType = „MPICH“

Būtini atributai: JobType, NodeNumber, Executable, StdOutput, StdError, InputSandbox, OutputSandbox.

MPICH užduotys tinka lygiagrečių užduočių, kurios turi komunikuoti tarpusavyje, vykdymui Grid aplinkoje. gLite (LCG-CE) tik rezervuoja telkinio mazgus užduoties vykdymui, o šių mazgų valdymas paliekamas pačiam naudotojui.

job-mpich.jdl:

```
Type = "Job";
JobType = "MPICH";
NodeNumber = 2;
Executable = "job-mpich.sh";
Arguments = "job-mpich";
StdOutput = "job-mpich.out";
StdError = "job-mpich.err";
InputSandbox = {"job-mpich.sh","job-mpich.c"};
OutputSandbox = {"job-mpich.err","job-mpich.out","mpiexec.out"};
Requirements = (other.GlueCEInfoLRMSType == "PBS");
```

```
RetryCount = 0;
Lrms_Type = "PBS";
```

job-mpich.sh:

```
#!/bin/sh -x
EXE=$1
mpicc -o ${EXE} ${EXE}.c
mpiexec `pwd`/$EXE > mpiexec.out 2>&1
```

job-mpich.c:

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char *argv[]){
   int numprocs;  /* Number of processors */
   int procnum;   /* Processor number */
   /* Initialize MPI */
   MPI_Init(&argc, &argv);
   /* Find this processor number */
   MPI_Comm_rank(MPI_COMM_WORLD, &procnum);
   /* Find the number of processors */
   MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
   printf ("Hi! from processor %d out of %d\n", procnum, numprocs);
   /* Shut down MPI */
   MPI_Finalize();
   return 0;
}
```

JobType = „Interactive“

Reikalingi atributai: JobType, Executable.

job-interactive.jdl:

```
Type="job";
JobType = "interactive";
Executable = "/bin/csh";
ListenerPort=6740;
```

Išsiuntus interaktyvią užduotį, UI elemente paleidžiama programa, kuri klausosi ant nurodyto prievado *ListenerPort* TCP protokolu. Kai užduotis startuoja WN, ji jungiasi prie UI *ListenerPort*. Taigi būtina, kad nurodytas UI prievadas būtų pasiekiamas iš išorės.

JobType = „DAG“

Būtini atributai: Type, JobType, Executable, Nodes, Description, Dependencies.

InputSandbox ir OutputSandbox gali būti bendri ir tarpusavyje susieti.

DAG tipo uždaviniai aprašomi grafo pavidalu, plačiau (žr. 1.2 skyrelį). Užduotys susiejamos parametru „dependencies“.

Jei neįvykdoma bent viena užduotis, visa „DAG“ užduotis baigiama su klaida ir pavienių rezultatų gavimas tampa painus.

job-dag.jdl:

```
[
Type = "dag";
Nodes = [
uzdA = [
Description = [
JobType = "Normal";
Executable="uzdA.sh";
StdError="job-dag.err";
StdOutput="job-dag.out";
InputSandbox = {"uzdA.sh"};
OutputSandbox={"job-dag.err","job-dag.out"};
];
];
uzdB = [
Description = [
InputSandbox = {};
JobType = "Normal";
Executable = "uzdB.sh";
StdError="job-dag.err";
StdOutput="job-dag.out";
InputSandbox = {"uzdB.sh"};
OutputSandbox={"job-dag.err","job-dag.out"};
];
];
];

Dependencies = { { uzdA, uzdB }};

];
```

jobA.sh:

```
#!/bin/sh -x
date
hostname
```

```
TL=`voms-proxy-info -timeleft`
# Ivertiname telkinio apribojimus:
UL=`ulimit -t`
if [ "$UL" -gt 0 2>/dev/null ] && [ "$UL" -lt "$TL" ]; then
    echo "UL < TL"
    TL=$UL
fi
# Is dvieju apribojimu pasirenkamas grieztesnis

echo -n "Site: $SITE_NAME Likes laikas vykdymui: $TL"
```

jobB.sh:

```
#!/bin/sh -x
date
hostname
echo -n "Site: $SITE_NAME CE: $GLITE_WMS_LOG_DESTINATION"
pwd
```

JobType = „Parametric“

Būtini atributai: JobType, Executable, Parameters, ParameterStart, ParameterStep

Tai specializuotas DAG užduoties atvejis. Užduotis tinka norint paleisti dideli skaičių panašių užduočių, kurios skiriasi tik įėjimo duomenimis. _PARAM_ JDL faile pakeičiamas parametro reikšme. Parametras kinta nuo pradinės reikšmės iki priešpaskutinės.

Jei neįvykdoma bent viena užduotis, visa „Parametric“ užduotis pasibaigia su klaida ir rezultatų gavimas tampa labai painus.

Siunčiamas per gLite-WMS serveri (LCG-RB netinka), t.y. naudojantis glite-wms-* komandomis.

job-par.jdl:

```
JobType = "Parametric";
Executable = "job-par.sh";
Parameters = 3;
ParameterStart = 1;
ParameterStep = 1;
Arguments = "_PARAM_";
StdOutput = "job-par_PARAM_.out";
StdError = "job-par_PARAM_.err";
InputSandbox = { "job-par.sh" };
OutputSandbox = { "job-par_PARAM_.out", "job-par_PARAM_.err" };
```

job-parametric.sh:

```
#!/bin/sh -x
date
echo "SITE: $SITE_NAME"
echo "GIIS: $SITE_GIIS_URL"
echo "LOG: $GLITE_WMS_LOG_DESTINATION"
echo "SE: $VO_BALTICGRID_DEFAULT_SE"
echo "PWD: $PWD"
uname -a
exec 2>&1
( env ; set ) |sort |uniq
echo "Arguments: $*"
```

Reikalavimai resursams

JDL faile galima nurodyti apribojimus, į kuriuos telkinius siųsti užduotį. Galima aprašyti sudėtingesnius apribojimus, apjungiant juos „&&“ (ir) ir „||“ (arba) loginiais operatoriais bei nurodant „!“ (inversija).

Telkinius, tenkinančius nurodytus reikalavimus galima pamatyti su *-job-list-match komandomis.

Reikalavimų pavyzdžiai

Telkiniai, kuriuose žinoma, kad HOME katalogai pasiekiami bendrai (tai aktualu MPI tipo užduotims, jei nesiimama priemonių kopijuoti vykdymui reikalingų failų į visus darbinius mazgus):

```
Requirements =
  other.GlueCEUniqueID == "ce.bg.ktu.lt:2119/jobmanager-pbs-balticgrid" ||
  other.GlueCEUniqueID == "pupa.elen.ktu.lt:2119/jobmanager-lcgpbs-balticgrid" ||
  other.GlueCEUniqueID == "atomas.itpa.lt:2119/jobmanager-lcgpbs-balticgrid" ||
  other.GlueCEUniqueID == "kriit.eenet.ee:2119/jobmanager-pbs-balticgrid";
```

Tik IA64 architektūros telkiniai (skirtingi telkiniai tą patį procesoriaus modelį vadina skirtingai, todėl ši sąlyga tik apytikrė):

```
Requirements = other.GlueHostProcessorModel == "IA64";
```

Tik tie telkiniai, kurie turi 2 GB ir daugiau operatyviosios atminties:

```
Requirements = other.GlueHostMainMemoryRAMSize >= 2096;
```

Prioritetai

JDL faile galima nurodyti, pagal kokius prioritetus bus pasirenkami telkiniai. Tai yra: jei keli telkiniai palaiko naudotojo VO ir tenkina nurodytus „Requirements“ – patikslinti į kurį iš jų bus siunčiama užduotis.

Pagal nutylėjimą „Normal“ tipo užduotims naudojama išraiška:

```
Rank = -other.GlueCEStateEstimatedResponseTime;
```

Siųsti į greičiausią telkinį (sąlyga nebūtinai teisinga, nes ne visi telkiniai teisingai praneša SI00 reikšmę):

```
Rank = other.GlueHostMainMemoryRAMSize;
```

Norint peržiūrėti, kokie prioritetai priskirti tinkamiems telkiniams naudojama komanda:

```
glite-job-list-match --rank job.jdl
```

MyProxy valdymas

Komandos: myproxy-init, myproxy-info, myproxy-destroy.

Proxy sertifikatas, sukurtas naudojant *-voms-proxy-init komandas, galioja ribotą laiką (pagal nutylėjimą 12 valandų). Jei užduotis proxy sertifikato galiojimo metu nebaigiama (laukė eilėje arba per ilgai truko skaičiavimai) – ji nutraukiama su klaidos pranešimu. To galima išvengti naudojant MyProxy servisą. Naudotojas inicijuoja MYPROXY sertifikato sukūrimą. Sukurtas sertifikatas saugomas MyProxy mazge. Užduotis gali pratęsti savo proxy sertifikatą naudodama MyProxy serveryje esantį naudotojo MYPROXY sertifikatą.

```
myproxy-init -s broker.eenet.ee -l saulius -a
```

Aukščiau pateikta komanda ne tik sukuria MYPROXY sertifikatą, bet ir nurodo pageidaujamas prisijungimo vardą (login) prie serviso. Prisijungimo vardą būtina nurodyti kai kuriuose valdymo įrankiuose vietoje DN (Distinguished Name). Paskutinis parametras „-a“ nustato, kad sertifikatą gali pasiekti ir trečiosios šalys, jei žino prisijungimo vardo ir slaptažodžio porą. Tokiu principu MyProxy naudojamas P-GRADE portale (žr. 1.3.6 skyrelį).

Užduočių kartojimas klaidos atveju

Užduočių pakartotinas siuntimas klaidos atveju nustatomas parametrais „RetryCount“ ir „ShallowRetryCount“. Pakartojimas laikomas „giliu“, jei naudotojo užduotis buvo paleista vykdyti WN elemente, tačiau pačios užduoties arba ją gaubiančio *WMS* skripto vykdymas buvo nesėkmingas. Kitu atveju toks kartojimas laikomas „negiliu“, kai užduoties vykdymas nepavyko dar prieš ją paleidžiant. Naudotojas gali nurodyti didesnes „RetryCount“ ir „ShallowRetryCount“, kad išvengtų atsitiktinių sutrikimų. Plačiau apie šių parametrų įtaką uždavinio vykdymo sėkmingumui aprašyta skyrelyje 1.2. Pagal nutylėjimą maksimalios šių parametrų reikšmės lygios 10. Gali būti nurodoma ir neigiama reikšmė pvz.: -1, tada pakartojimo mechanizmas visiškai išjungiamas.

Užduoties aprašo šablono kūrimas

Atliekant autonominį užduočių valdymą, geriausia susikurti pakankamai abstraktų šabloną, kuris tiktų daugumai uždavinių. Tokio šablono pavyzdys pateikiamas žemiau. Šablonui buvo pasirinktas „Normal“ užduoties tipas, nes jo veikimas paprastas ir stabilus. Siunčiant uždavinius nedidelėmis 2-5 užduočių grupelėmis galima būtų naudoti ir užduočių rinkinius arba parametrinį tipą. Vykstant nenumatytiems pasikeitimams vykdymo eigoje, lengviausia atlikti

operacijas su pavienėmis užduotimis. Kai užduotys yra visiškai nepriklausomos, bet kuriuo momentu galima susigrąžinti dalį rezultatų iš baigusių darbą užduočių. Remiantis šiame darbe atlikta heterogeninės aplinkos analize sudarome šabloną, pagal kurį galėsime generuoti JDL aprašus kiekvienai užduočiai:

```
executable = "/bin/sh";
Arguments = "-c '/bin/tar zxf grid.tar.gz; rm -f grid.tar.gz; p1'";
StdOutput = "stdout-p2.txt";
StdError = "stderr-p2.txt";
InputSandbox = {"grid.tar.gz"};
OutputSandbox = {"stdout-p2.txt","stderr.txt-p2","p3"};
Requirements = other.GlueCEPolicyMaxCPUTime > p4;
```

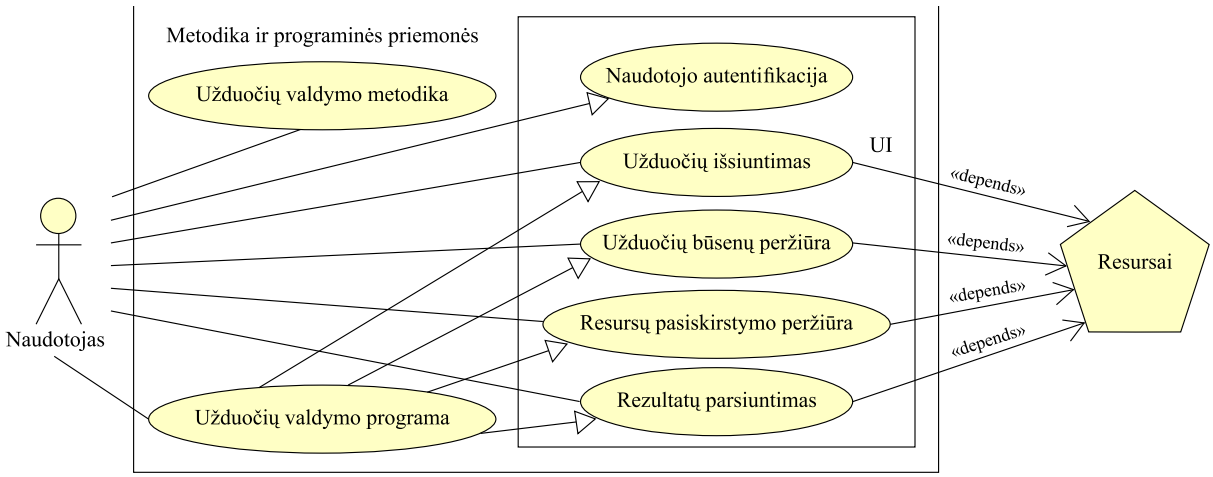
Parinktas universalus būdas išsiųsti uždavinį nepriklausomai nuo jo failų struktūros:

- sukuriamas viso uždavinio katalogo archyvas;
- nurodomas šio archyvo perdavimas per „InputSandbox“;
- užduočiai pasiekus vykdymo vietą uždavinys pirma išpakuojamas ir tik tada paleidžiamas. Tai nustatoma „Arguments“ pradžioje. Vykdomasis failas įterpiamas vietoje p_1.

Naudojant tokį šabloną daug paprasčiau testuoti uždavinius lokalioje eilių sistemoje, o paskui nekeičiant uždavinio failų struktūros, nei šablono, išsiųsti jį vykdyti į Grid. Atliekant autonominį siuntimą vietoje parametro p_1 bus įrašytas naudotojo nurodytas vykdomasis failas. Vietoje parametro p_2 bus parenkamas užduoties numeris. O vietoje p_3 lauko bus įrašoma naudotojo nurodytas rezultatų failas, kuris turi grįžti (standartiniai pranešimai iš *stdin* ir *stderr* nurodyti statiškai). Šiame pavyzdyje p_4 skirtas apriboti užduočių siuntimą į trumpas eiles. Jei uždavinys vykdomas bent kelias valandas, toks apribojimas apsaugos nuo bereikalingo užduočių paleidimo trumpose eilėse ir jų nutraukimo. Naudotojas gali priskirti ir daugiau apribojimų pagal taisykles aprašytas 2.1 skyrelyje.

2.2. Naudotojo poreikių studija

Reikalingas mechanizmas, kuris patikimai dirbtų esant nepilnai, neaiškiai ir ne visada pasiekiamai informacijai. Užduočių valdymas turi būti greitas, lanksčiai plečiamas ir kuo mažiau susietas su konkrečiais uždaviniais ar naudotojo parašytomis programomis. Naudotojas taip pat norės būti tikras, kad neautorizuoti asmenys negalės valdyti jo užduočių ar manipuliuoti jo duomenimis [16].



16 pav. Panaudos atvejų diagrama

Šiame darbe aprašyta metodika ir įrankiai skirti naudotojams, turintiems teisę pasiekti skaičiuojamuosius resursus *BalticGrid* ar *LitGrid* tinkle. Metodika padeda autonomiškai vykdyti užduotis pagal naudotojo pasirinktus parametrus.

2.3. Panaudojimo atvejų scenarijai

Panaudos atvejų diagramoje (pav. 16) pavaizduota kokias funkcijas naudotojas gali atlikti naudodamas Grid užduočių valdymo metodiką ir įrankį. Žemiau pateikiami šių funkcijų panaudojimo scenarijai:

1. PANAUDOJIMO ATVEJIS: naudotojo autentifikacija

Aktorius: Grid naudotojas.

Aprašas: apima procesą kuriame naudotojas autentifikuojamas sukuriant *VOMS* proxy, kad galėtų naudotis Grid servisais.

Prieš sąlyga: naudotojas neturi galiojančio laikinojo *VOMS* proxy; Naudotojas neautentifikuotas.

Sužadinimo sąlyga: naudotojas įvesdamas privataus rakto slaptažodį sukuria *VOMS* proxy.

Po sąlyga: galimas autorizuotas Grid valdymo komandų vykdymas.

2. PANAUDOJIMO ATVEJIS: užduočių išsiuntimas

Aktorius: Grid naudotojas.

Aprašas: apima parametrų patikrinimo procesą, naudotojo patvirtinimo užklausimą ir užduočių išsiuntimą.

Prieš sąlyga: naudotojo *VOMS* proxy galioja.

Sužadinimo sąlyga: naudotojas paleidžia programą *Sietas* (žr. 3.4 skyrelį) su parametru „submit“.

Po sąlyga: patikrinus parametrus uždavinys siunčiamas į Grid.

3. PANAUDOJIMO ATVEJIS: užduočių būsenos peržiūra

Aktorius: Grid naudotojas.

Aprašas: apima patikrinimą ar egzistuoja užduočių sąrašas ir būsenų tikrinimo procesą.

Prieš sąlyga: naudotojo *VOMS* proxy galioja. Užduočių sąrašas egzistuoja.

Sužadinimo sąlyga: naudotojas paleidžia programą *Sietas* (žr. 3.4 skyrelį) su parametru „status“ arba parametru „status all“.

Po sąlyga: gaunama bendra arba detali informacija apie užduočių vykdymą Grid'e.

4. PANAUDOJIMO ATVEJIS: priverstinis užduočių nutraukimas

Aktorius: Grid naudotojas.

Aprašas: apima patikrinimą ar egzistuoja užduočių sąrašas, naudotojo patvirtinimo užklausimą ir visų arba tik „Scheduled“ būsenos užduočių nutraukimą.

Prieš sąlyga: naudotojo *VOMS* proxy galioja. Užduočių sąrašas egzistuoja.

Sužadinimo sąlyga: naudotojas paleidžia programą *Sietas* (žr. 3.4 skyrelį) su parametru „abort“ arba parametru „abort scheduled“.

Po sąlyga: nutraukiamos visos arba tik „Scheduled“ būsenos užduotys (priklausomai nuo prieš tai pasirinkto parametro).

4. PANAUDOJIMO ATVEJIS: rezultatų parsiuntimas

Aktorius: Grid naudotojas.

Aprašas: apima patikrinimą ar egzistuoja užduočių sąrašas, ir „Finished“ būsenos užduočių rezultatų parsiuntimo inicijavimą.

Prieš sąlyga: naudotojo *VOMS* proxy galioja. Užduočių sąrašas egzistuoja. Yra baigusių darbą užduočių.

Sužadinimo sąlyga: naudotojas paleidžia programą *Sietas* (žr. 3.4 skyrelį) su parametru „status“ arba parametru „status all“.

Po sąlyga: inicijuojamas rezultatų parsiuntimas užduotims, kurių būsena „Finished“.

4. PANAUDOJIMO ATVEJIS: užduočių valdymo metodika

Aktorius: Grid naudotojas.

Aprašas: apima valdymo metodikos panaudojimą užduočių reikalavimams ar apribojimams aprašyti.

Prieš sąlyga: naudotojas nori paruošti uždavinį vykdyti paskirstytoje skaičiavimų aplinkoje ir nustatyti tam tikrus apribojimus.

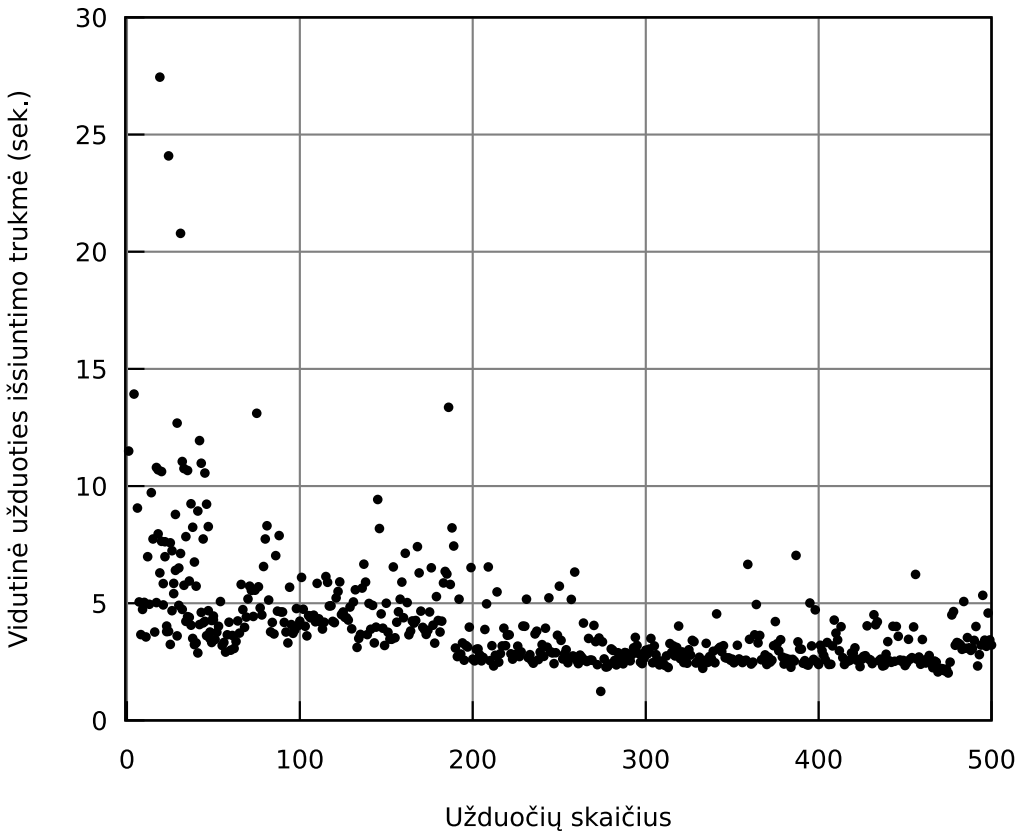
Sužadinimo sąlyga: naudotojas radęs reikalingus reikalavimus (žr. 2 skyrelį) aprašo juos valdymo įrankio *Sietas* (žr. 3.4 skyrelį) JDL (žr. 2.1 skyrelį) šablone.

Po sąlyga: naudotojas turi savo reikmėms pritaikytą uždavinio valdymo mechanizmą.

3. ATLIEKAMŲ TYRIMŲ REZULTATAI

3.1. Užduočių siuntimo analizė

Tyrimo metu buvo atlikti automatizuoti užduočių siuntimai ir išmatuota šių siutimų trukmė (pav. 17). Čia išmatuotas siuntimo laikas tarp UI → *WMS*. Pagal diagramą galima pastebėti, kad siunčiant didesnį skaičių užduočių siuntimo trukmė konverguoja į 3-4 sekundžių reikšmę. Tai galima paaiškinti didesniu sistemos apkrovimu pradedant siųsti užduotis, kol informacinė sistema dar neturi pakankamai informacijos apie skaičiavimų aplinkos būklę.



17 pav. Užduočių išsiuntimo trukmė

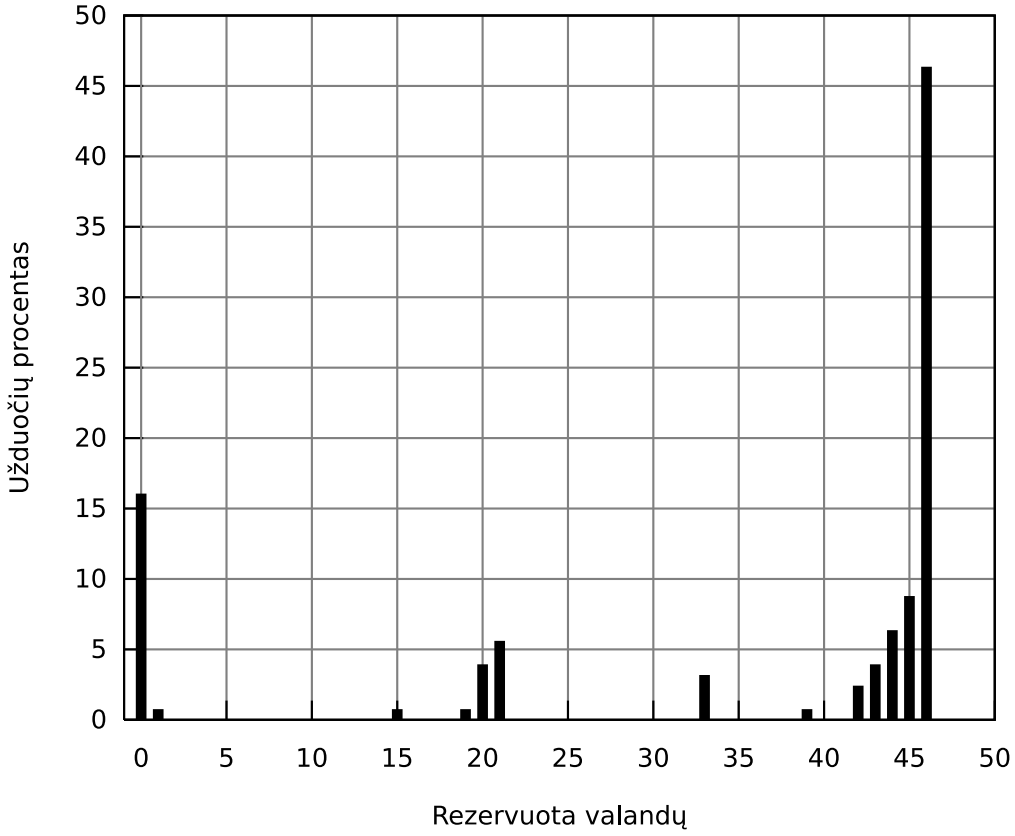
Išrinkimo algoritmai *WMS* servise turi apklausti top-BDII servisus, kad galėtų atnaujinti duomenis apie resursų būseną. Kitos užduotys jau gali būti išsiunčiamos ir priimamoms *WMS* WMProxy greičiau, kai turima pakankamai nauja informacija pačio *WMS* serviso informacinėse sistemose. Naudojant GANGA užduočių išsiuntimas į LCG (*gLite* pirmtakas) vienai užduočiai užtrunka 10-20 sekundžių [9]. Darbe neparašyta ar tai laikas visas laikas, kurį užtrunka užduotis keliaudama iki CE telkinio, ar tik trukmė iki RB. Šis eksperimentas patvirtina, kad *middleware* įrankiai tobulėja ir veikia sparčiau. Žemame lygmenyje nuo jų labai priklauso užduočių aibės vykdymo trukmė.

Galime charakterizuoti kokia bus užduočių aibės išsiuntimo trukmė. Užduočių išsiuntimo trukmė e_i užduočiai u_i gali būti apibrėžta laikų, kuriuos užtrunka užduoties išsiuntimas į *WMS* u_{is}, užduoties laukimas eilėje u_{iw} bei užduoties siuntimas į UI u_{in}, suma. Iš čia $e_i = u_{is} + u_{iw} +$

u_{in}. Tada užduoties u_i tikėtinas užbaigimo laikas c_i nuo išsiuntimo momento bus $c_i = e_i + t_p - t_r$. Čia t_p – naudotojo *VOMS* proxy galiojimo laikas, o t_r rezervinis laikas, paliktas užduoties rezultatų grąžinimui: $0 \leq t_r < t_p$. Tegul $\underline{\mathrm{M}}$ bus aibė užduočių, kurias norima įvykdyti. Visa užduočių aibės vykdymo plano trukmė gali būti apibrėžta kaip $max_{u_i \in M}(c_i)$. Ši trukmė parodo heterogeninės skaičiavimų aplinkos našumą, bet neišmatuoja individualių užduočių vykdymo kokybės *QoS* (Quality of Service). Reikia pabrėžti, kad šiame darbe dėmesys sutelktas į bendrą uždavinio, o ne kelių konkuruojančių užduočių vykdymo laiko optimizavimą.

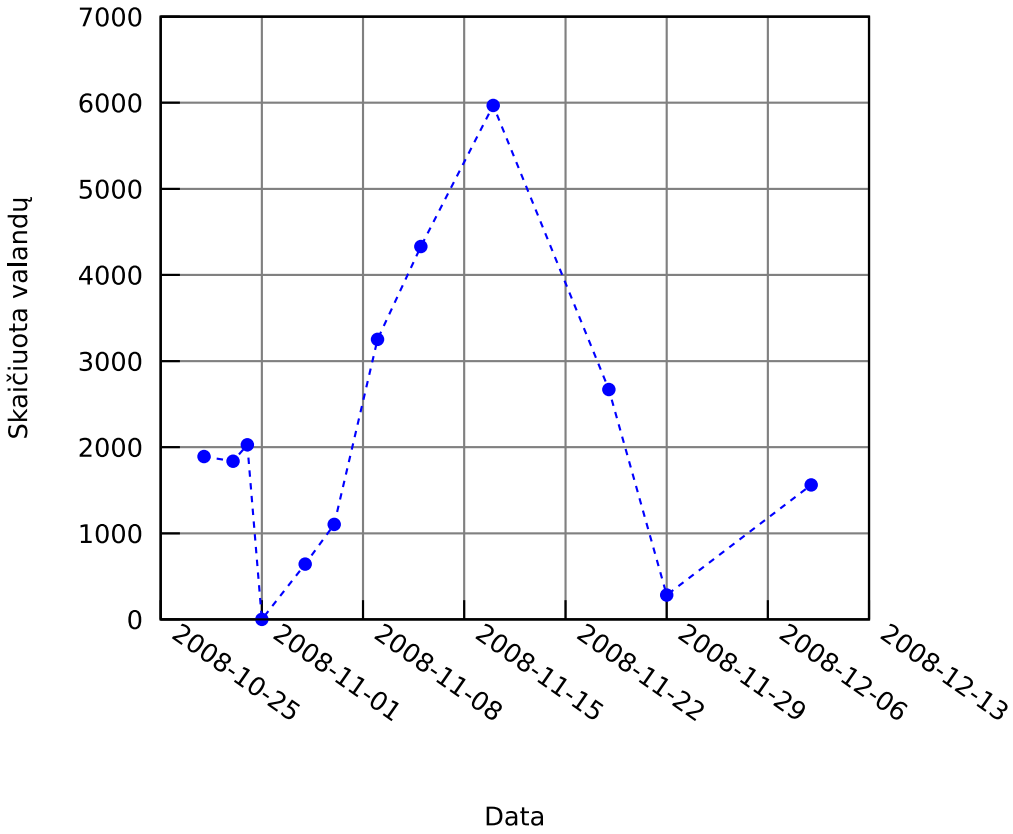
3.2. Užduočių pasiskirstymas valandomis

Vykdant eksperimentus su schemų modeliavimu (žr. 3.4 skyrelį) buvo atliktas užduočių pasiskirstymo procentais pagal tai kiek valandų jos gavo skaičiavimų vykdymui, tyrimas. Likęs laikas skaičiuotas nuo startavimo momento darbiniame elemente WN. Be to palikta viena valanda rezervui, kad būtų užtikrintas rezultatų grįžimas. Buvo išsiųsta lygiai 200 užduočių. Iš diagramos (pav. 18) matome, kad 16% iš jų teko 0 arba mažiau valandų skaičiavimo laiko (jei įskaitant rezervuotą valandą grįžimui).



18 pav. Užduočių pasiskirstymas pagal valandas

Panašus pasiskirstymas buvo gaunamas ir kartojant kitus eksperimentus. Taip pat matome, kad beveik pusei užduočių pavyko rezervuoti visą skaičiavimų laiką (buvo naudojamas 48 valandas galiojantis *VOMS* proxy sertifikatas). Likusi užduočių dalis pasiskirstė per visą likusį *VOMS* proxy galiojimo laiką.

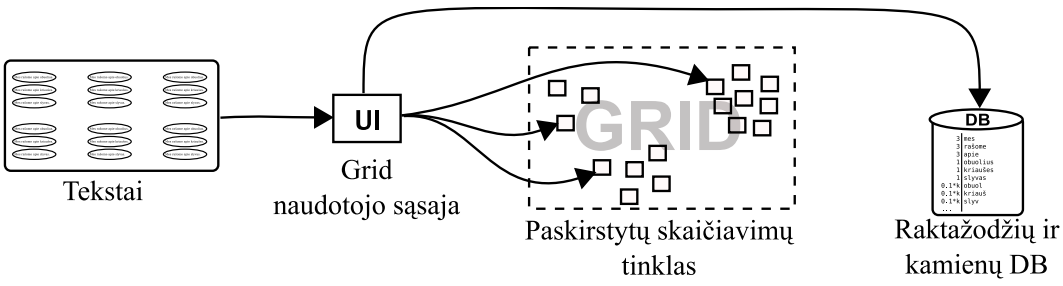


19 pav. Rezultatai iš skaičiavimų protokolo

Šiame grafike (pav. 19) pateikiama statistika iš skaičiavimų protokolo ištraukos. Kas savaitę buvo siunčiama maždaug po 200 užduočių. Taškai indikuoja uždavinio užbaigimą ir užduočių rezultatų grįžimą. Ne visos užduotys būdavo užbaigtos sėkmingai, tačiau iš grįžusių, skaičiavimų rezultatai buvo sėkmingai kaupiami ir skaičiuotų valandų skaičius rašomas į protokolą.

3.3. Lietuviško teksto santraukos automatizavimas

Teksto raktažodžių išrinkimas pasitarnauja sprendžiant informacijos pertekliškumo ir apžvalgos problemas. Šio uždavinio tikslas atlikti raktažodžių išrinkimą ir vertinimą.



20 pav. Tekstų apdorojimo mechanizmas

Ilgiems dokumentams apdoroti šiuo metu naudojama paskirstyta skaičiavimo sistema. Didžiausią laiko dalį užima prefiksų ir sufiksų atskyrimas [22]. Todėl būtent ši dalis skaičiavimų

ir atliekama Grid'e (pav. 20). Kiekvienas žodis sutikrinamas bent su 4000 šablonų, o prefiksų atskyrimas atliekamas rekursiškai. Uždavinio skaldymas atliekamas tokiu principu: tekstas išskaidomas į žodžius, tuomet priklausomai nuo laisvų skaičiavimo mazgų skaičiaus, žodžiai grupėmis siunčiami kamienų išskyrimui. Rezultatai spausdinami tekstiniame faile bei išsaugomi duomenų bazėje. Toliau lokaliai atliekamas raktažodžių perrinkimas atsižvelgiant į jau esamus jų koeficientus duomenų bazėje. Kol kas duomenims saugoti naudojamas vienas MySQL serveris. Afiksų lentelės naudojamos iš atviro kodo lietuviškos rašybos tikrinimo įrankio aspell.

Uždavinio reikalavimai:

Programinė įranga: PERL $\geq$ 5.6.

Techninė įranga:

- RAM $\geq$ 128 MB;
- HDD $\leq$ 30 MB disko vietos WN.

Skaičiavimų rezultatai:

Duomenų bazėje saugomi apdoroti ir įvertinti 51679 unikalūs žodžių kamienai;

Pagrindinės problemos kylančios naudojant Grid:

- užduočių replikavimas (dalinai išspręsta);
- interaktyvumo trūkumas;
- aukšto patikimumo paslaugų galimybė.

3.4. Elektroninių schemų modelių simuliavimas

2008-ais metais buvo tobulinami ir kuriami lustų testų sudarymo uždaviniai. Realizavus vykdymo laiko apribojimą aprašytą 1.5, įėjimų-išėjimų sąryšių nustatymo uždavinys tapo tinkamas vykdymui visuose į BalticGrid infrastruktūrą įjungtuose telkiniuose. Atlikti eksperimentus su didesnėmis schemomis sudėtinga dėl labai didelių laiko ir operatyvinės atminties sąnaudų [20].

Anksčiau modelių simuliavimui vykdyti būdavo naudojamas KTU lygiagrečių skaičiavimų klasteris (6 PC su Linux operacine sistema, viso 12 CPU) [20]. Dabar šis telkinys jau įjungtas į Grid infrastruktūra ir pasiekiamas ne tik KTU naudotojams.

Uždavinio reikalavimai:

Programinė įranga:

- PYTHON $\geq$ 2.3 rezultatų apjungimui UI pusėje;
- Bet koks ISO C++ palaikantis kompiliatorius WN pusėje.

Techninė įranga:

- RAM $\geq$ 128 MB;

- HDD $\leq$ 10 MB disko vietos WN.

Skaičiavimų rezultatai:

- c7552 (206×107) funkcijos modelis išsemtas;
- b17 (1452×1512) funkcija arti pabaigos (likę 39 išėjimai iš 1512);
- b17_1 (1452×1512) pradėta analizuoti;
- b20 (522×512) pradėta analizuoti;
- b20_1 (522×512) pradėta analizuoti.

Dar apie 40 schemų modelių laukia apdorojimo.
Pagrindinės problemos kilusios naudojant Grid aplinką:

- TAR_UI įdiegimas sudėtingas paprastiems vartotojams;
- naudotojo dokumentacija išmėtyta po visą internetą ir daugybę *wiki* sistemų;
- trūksta centralizuotos ir aiškios informacijos apie UI komandų naudojimą autonomiškai.

IŠVADOS

1. Literatūros analizė parodė, kad užduočių valdymo įrankiai padeda naudotojams pasiekti Grid resursus standartizuota, draugiška ir lengvai pritaikoma sąsaja. O resursų teikėjai gali sumažinti operacijų išlaidas skiriamas naudotojų palaikymui. Valdymo įrankiai paspartina programų kūrėjų darbo ciklą.

2. Apibendrinus užduočių valdymo įrankių privalumus ir trūkumus išrinkti du įrankiai, geriausiai atspindintys autonominio užduočių valdymo paradigmą. Tai *gEclipse* valdymo aplinka su grafine vartotojo sąsaja ir GANGA įrankis su tekstine vartotojo sąsaja.

3. Darbe sukurtos užduočių valdymo heterogeninėje aplinkoje metodikos privalumai:

   (a) pasiekti geresni taikomųjų uždavinių rezultatai negu naudojant ankstesnes metodikas;

   (b) yra abstrakti ir gali būti pritaikoma daugeliui uždavinių.

4. Tyrimo metu prieita išvados, kad tam tikras į Grid įjungtų resursų heterogeniškumo lygis visada išliks, dėl skirtingos dalyvių vidinės politikos ir išteklių.

REZULTATAI

- Sukurtas užduočių valdymo įrankis „*Sietas*“ (žr. 3.4 skyrelį).
- Sudarytas tipinis užduočių vykdymo modelis ir pasiūlyta metodika, kuri remiasi parametriniu pavienių užduočių valdymu ir uždavinio skaičiavimo laiko apribojimu.
- Darbo metu sukaupta medžiaga ir tyrimų rezultatai paskelbti:
  - atspausdintas straipsnis ir perskaitytas pranešimas KTU konferencijoje „Informacinės Technologijos 2007“;
  - atspausdintas straipsnis ir perskaitytas pranešimas KTU konferencijoje „Informacinės Technologijos 2008“;
  - atspausdintas straipsnis ir perskaitytas pranešimas KU konferencijoje „Fundamentiniai tyrimai ir inovacijos mokslų sandūroje“ 2008 m.;
  - 2008 m. perskaitytas pranešimas VVK konferencijoje „Innovative Infotechnologies for Science, Business and Education“;
  - priimtas pranešimas į „4th EGEE User Forum“ 2009.
- Sukurta metodika naudojasi KTU Informacinių Technologijų Plėtros Instituto kompiuterinių sistemų skyriaus darbuotojai.
- Išanalizuotas esamas ir prieinamas BalticGrid infrastruktūros ir aplinkos veikimas;
- Darbe siūlomą metodiką numatyta kryptingai plėtoti ateityje, papildant ją pagal kintančius naudotojų poreikius.

LITERATŪRA

[1] Ammar H. Alhusaini, Cauligi S. Raghavendra, ir Viktor K. Prasanna. Run-time adaptation for grid environments. Iš *IPDPS*, 87 p. IEEE, 2001. ISBN 0-7695-0990-8.

[2] José M. Alonso, Vicente Hernández, ir Germán Moltó. Towards on-demand ubiquitous metascheduling on computational grids. Iš *PDP*, p. 84–90. IEEE Computer Society, 2007.

[3] David A. Bader ir Robert Pennington. Cluster computing: Applications. 2002.

[4] Haresh S. Bhatt, Dharmesh Bhansali, Sonal Shah, P R Patel, VH Patel, ir Arup Dasgupta. GANGA: Grid application information gathering and accessing framework. *International Journal of Information Technology*, 11(4), 2007.

[5] Rajkumar Buyya, Steve J. Chapin, ir David C. DiNucci. Architectural models for resource management in the grid. Iš Rajkumar Buyya ir Mark Baker, red., *GRID*, tomas 1971 iš *Lecture Notes in Computer Science*, p. 18–35. Springer, 2000. ISBN 3-540-41403-7.

[6] Javier Carretero. Use of genetic algorithms for scheduling jobs in large scale grid applications. 2006.

[7] Simon Davy, Karim Djemame, ir Jason Noble. The application of bioinspired engineering principles to grid resource allocation. 2003.

[8] Jörg Decker ir Jörg Schneider. Heuristic scheduling of grid workflows supporting co-allocation and advance reservation. Iš *CCGRID*, p. 335–342. IEEE Computer Society, 2007.

[9] Johannes Elmsheuser. Distributed analysis within the LHC computing grid. 2007, gegužė.

[10] Aram Galstyan, Karl Czajkowski, ir Kristina Lerman. Resource allocation in the grid using reinforcement learning. Iš *AAMAS*, p. 1314–1315. IEEE Computer Society, 2004. ISBN 1-58113-864-4.

[11] Brighten Godfrey ir Richard M. Karp. On the price of heterogeneity in parallel systems. Iš Phillip B. Gibbons ir Uzi Vishkin, red., *SPAA*, p. 84–92. ACM, 2006. ISBN 1-59593-452-9.

[12] Next Generation GRIDs Expert Group. Future for european grids: GRIDs and service oriented knowledge utilities – vision and research directions 2010 and beyond, 2006, sausis. report #3.

[13] Derrick Kondo, Andrew A. Chien, ir Henri Casanova. Resource management for rapid application turnaround on enterprise desktop grids. Iš *SC*, 17 p. IEEE Computer Society, 2004. ISBN 0-7695-2153-3.

[14] Miroslaw Kupczyk. *EGEE - gLite Tutorial Riga*. PSNC, 2006, spalis.

[15] Kevin Lai, Bernardo A. Huberman, ir Leslie R. Fine. Tycoon: A distributed market-based resource allocation system. *CoRR*, cs.DC/0404013, 2004. informal publication.

[16] Daniel Lorenz, Peter Buchholz, Christian Uebing, Wolfgang Walkowiak, ir Roland Wismüller. Secure communication for computational steering of grid jobs. Iš *PDP*, p. 209–217. IEEE Computer Society, 2008.

[17] N. Muthuvelu, I. Chai, ir C. Eswaran. An adaptive and parameterized job grouping algorithm for scheduling grid jobs. Iš *Advanced Communication Technology. ICACT 2008*, 2008.

[18] Jason Novotny. The grid portal development kit. *Concurrency and Computation: Practice and Experience*, 14(13-15):1129–1144, 2002.

[19] Bartek Palak, Marcin Plóciennik, ir Miroslaw Kupczyk. *Running On the Grid Using Migrating Desktop*. PSNC, 1.1 edition, 2006.

[20] Kęstutis Paulikas. *Testų generavimas pagal algoritminius aprašus*. Daktaro disertacija, KTU, 2007.

[21] Saulius Petrauskas. GRID perspektyva - SOKU ir žinių paslaugos. *Informacinės technologijos 2007*, p. 369–373, 2007. ISSN 1822-6337.

[22] Saulius Petrauskas ir Regina Misevičienė. Lithuanian text summarization based on keyword cross-occurrence. *Information Technologies' 2008*, p. 49–52, 2008. ISSN 2029-0063.

[23] R. Plėštys, G. Vilutis, ir K. Paulikas. The analysis of the tools of grid middleware. Iš *ITI 2008 : proceedings of the 30th International Conference on Information Technology Interfaces,*, p. 873–877, 2008, birželis. ISBN 978-953-7138-12-7.

[24] Marcin Radecki et al. PBS statistics (BAT). https://sam-web.cyfronet.pl/bat/. [žiūrėta 2008-12-21].

[25] Donatas Sandonavičius. GRID teikiamų servisų efektyvumo analizė. Magistro darbas, KTU, 2007.

[26] Howard Jay Siegel ir Shoukat Ali. Techniques for mapping tasks to machines in heterogeneous computing systems, 1999.

[27] Oliver Sinnen ir Leonel Sousa. List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures. *Parallel Computing*, 30(1):81–101, 2004.

[28] Dan Tsafrir ir Dror G. Feitelson. Instability in parallel job scheduling simulation: the role of workload flurries. Iš *IPDPS*. IEEE, 2006.

PRIEDAI

1 PRIEDAS

Užduočių valdymo įrankio „Sietas“ programinis kodas.

```
#!/usr/bin/perl
#      Easy interface to the Grid.
#    Saulius Petrauskas (c) 2008-2009
#
#       Compatible with gLite 3.1
$version = 'v1.35'; # Produkto versija.

# Irankis skirtas uzdaviniams paleisti Grid'e autonomiskai.
# Uzdaviniai atskiriami katalogais. Sietas siuncia visa
# katalogo archyva, todel uzdavinio specialiai pritaikyti
# nereikia.

# Nustatymai pagal nutylejima:
$RT = 96;                       # 96 valandu proxy galiojimas
$VO = 'balticgrid';             # Virtuali organizacija
$vykdyti = 'sh saulius.sh';     # vykdomoji komanda uzduociai startavus
$reziai = '1,100';              # nuo 1 iki n (kol kas tik sveiki sk.)
$rezfile = 'res';               # rezultatu failas, kuris turi įgrzti
$timeout = 60;                  # kiek sekundziu galima laukti gLite
                                # valdymo komandos vykdymo
# ------------------
my ($minr, $maxr);

if (($ARGV[0] eq '') || ($ARGV[0] eq 'help')) {
    help();
    exit;
}

$timeleft = voms_proxy_info();
if ($timeleft < 1) {
    print "Inicializuojamas voms-proxy (palaukite ~30 sek.)...\n";
    voms_proxy_init(); # jei liko maziau nei 1 valanda - pratesiame proxy.
    # palaukiame 30 sek. kad isvengti klaidu del laiko skirtumu:
    sleep(30);
}

if ($ARGV[0] eq 'status') {
    status();  # jei "status all" <- rodomas ir resursu pasiskirstymas
    exit;
} elsif ($ARGV[0] eq 'abort')  {
    abort();   # abort - nutraukti vykdomas uzduotis
    exit;
}
```

```
while (parametrai()) { }    # ivedami/keiciami siuntimo parametrai
init_vars();                # inicializuojami darbu sablonai (template)
if ($timeleft < ($RT-2)) { # siuntimui reikalinga laiko atsarga: 2 h.
    print "Inicializuojamas voms-proxy (palaukite ~30 sek.)...\n";
    # jei liko maziau nei $RT - 2 valandos - pratesiame VOMS proxy:
    voms_proxy_init();
    # palaukiame 30 sek. kad isvengti klaidu del laiko skirtumu
    sleep(30);
}
submit_jobs();              # siunciame uzduotis

sub help {
    print "Sietas ($version) galimi parametrai: submit, status,".
          " status all, abort, abort scheduled, help\n\n";
}

sub vykdymas {
# gLite komandu vykdymas su klaidu tikrinimu ir apribojimu uzbaigimo laikui
    my $cmd = shift;
    eval {
        local $SIG{ALRM} = sub { die "alarm\n" };
        # jei komanda "uzstrigs", jos vykdyma nutrauksime po $timeout sek.
        alarm $timeout;
        $ret = join('',`$cmd`);
        alarm 0;
    };
    return $ret
}

sub voms_proxy_init { # sukuria proxy, galiojanti $RT valandu
    $_ =vykdymas("voms-proxy-init -voms $VO -hours $RT -valid $RT:0 2>&1");
    if (/valid until/) {
        return
    }
    die "Nepavyko inicializuoti voms-proxy del: $ret\n";
}

sub voms_proxy_info {
    $_ = vykdymas("voms-proxy-info");
    if (/timeleft.*?(\d+):/) {
        return $1
    }
    return -1
}

sub parametrai { # interaktyvus uzdavinio parametru ivedimas / keitimas
    my $ok = 'n';
    ($minr,$maxr) = split(/,/,$reziai);
    print "-----------------------------------------\n";
```

```
    print "Pasitikrinkite ar duomenys teisingi:\n";
    print "Bus vykdoma komanda: $vykdyti\n";
    print "Bus siunciama ".($maxr - $minr + 1)." uzduociu\n";
    print "Intervalas: [$minr, $maxr]; Zingsnis: 1\n";
    print "Rezultatu failas: $rezfile\n";
    ask("Siusti? [y/n] ", \$ok);
    if ($ok ne 'y') {
        print "Siuntimo parametru keitimas:\n";
        ask("Komanda vykdoma uzduociai startavus: [$vykdyti] ",\$vykdyti);
        ask("Kokie argumento reziai? [$reziai]: ",\$reziai);
        ask("Nurodykite grazinamu rezultatu faila: [$rezfile] ",\$rezfile);
        $reziai =~ s/[^\d\,]//g;
        ($minr, $maxr) = split(/,/, $reziai);
        return 1
    }
    return 0
}

sub submit_jobs {
    print "Siunciamos uzduotys i Grid:\n";
    if (-r 'grid.jobs') { # tebera uzduociu sarasas
        my $ok = 'y';
        ask("Like nebaigtu uzduociu is praeito siuntimo. ".
            "Ar tikrai norite siusti naujas? ".
            "(Senos bus istrintos is output/!) [y/n] ", \$ok);
        if ($ok eq 'n') {
            print "Pasirinkote baigti darba.\n";
            exit
        }
        unlink('grid.jobs')
    }
    `rm -rf output/`; # trinamas rezultatu katalogas!
    create_archive(); # archyvuojam esamas katalogas
    if ($maxr < $minr) { # patikrinime rezius
        print "Intervalas neigiamas!\n";
        exit
    }
    for ($i = $minr; $i <= $maxr; $i++) {
        my $job = '';
        # iterpiame iteracija: ('rysys.sh %d')->('rysys.sh 1'),2,3...
        my $komanda = sprintf($vykdyti, $i);
        # istatome reiksmes i JDL sablona:
        $job = sprintf($jdl, $komanda, $i, $i, $i, $i, $rezfile);

        open(O,">grid.jdl");
        print O $job;
        close(O);

        # siunciame įuzduot į Grid:
```

```
        my $ret =vykdymas("glite-wms-job-submit -a -o grid.jobs grid.jdl");
        if ($ret =~ /Success/) {
            print "Uzdavinio dalis $i sekmingai issiusta!\n";
            if ($ret =~ /\n(https:\/\/[^\n]+)/) {
                $jobid = $1;
                print "jobID: $jobid\n";
            }
        } else {
            print "Nepavyko issiusti dalies $i del $ret\n";
        }
        sleep(3) # uzdelsimas tarp siuntimu (nebutinas)
    }
}

sub status {
    if (! -r 'grid.jobs') {
        print "Uzduociu nera.\n";
        exit
    }
    my @ret = `glite-wms-job-status --noint -i grid.jobs`;
    my ($run,$done,$sch,$other,$clr,$abort,$read,$wait)=(0,0,0,0,0,0,0,0);
    my %clusters; # resursu pasiskirstymas
    my $clust_int = 0;
    foreach(@ret) {
        # if (/Status info for/) { $jid = $_; }
        if (/Current Status: /) {
            if (/Running/)      { $run++ }
            elsif (/Done/)      { $done++ }
            elsif (/Scheduled/) { $sch++ }
            elsif (/Cleared/)   { $clr++ }
            elsif (/Waiting/)   { $wait++ }
            elsif (/Ready/)     { $read++ }
            elsif (/Aborted/)   { $abort++ } # ;print "aborted: $jid\n"; }
            else { $other++; } # $other_what=$_; }
        }
        if (/Destination: +([^:]+):/) {
            $clusters{$1}++; $clust_int++;
        }
    }
    print "Status for jobs:\n";
    print "  Scheduled: $sch\n";
    print "  Waiting: $wait\n";
    print "  Ready: $read\n";
    print "  Running: $run\n";
    print "  Finished: $done\n";
    print "  Cleared: $clr\n";
    print "  Aborted: $abort\n";
    print "  Other: $other\n";
    if ($done > 0) { # try to get output
```

```
        print "Downloading finished jobs output:\n";
        get_output();
    }
    if (($clr > 0) && (count_jobs() == $clr)) { # baigtos visos uzduotys
        unlink('grid.jobs'); # istrinamas uzduociu sarasas
    }
    if ($ARGV[1] eq 'all') {
        print "Status for cluster usage:\n";
        my @srt = sort {$b <=> $a} keys %clusters;
        foreach(@srt) { # spausdinama vykdymo statistine informacija
            printf("  %2d (%2.f %%) %s\n",$clusters{$_},
                    $clusters{$_}/$clust_int*100,$_);
        }
    }
}


sub abort { # uzduociu nutraukimas
    if (! -r 'grid.jobs') {
        print "Uzduociu nera.\n";
        exit
    }
    my $ok = 'n';
    my $uzd = 'visus';
    if ($ARGV[1] eq 'scheduled') {
        $uzd = 'scheduled';
    }
    ask("Ar tikrai nutraukti *$uzd* uzdavinius? [y/n] ", \$ok);
    if ($ok ne 'y') {
        exit
    }

    my @ret = `glite-wms-job-status --noint -i grid.jobs`;
    my ($run, $done, $sch, $other, $clr) = (0, 0, 0 ,0, 0);
    foreach(@ret) {
        if (/Status info for the Job : (https:.*)/) {
            $jobid = $1
        }
        if (/Current Status: /) {
            if ((! /Done/) && (! /Cleared/) && ($uzd eq 'visus')) {
                push(@abort, "$jobid\n")
            }
            if ((/Scheduled/) && ($uzd eq 'scheduled')) {
                push(@abort, "$jobid\n")
            }
        }
    }
    open(F,">grid_abort.jobs");
    print F @abort;
```

```
    close(F);
    @ret = `glite-wms-job-cancel --noint -i grid_abort.jobs`;
    foreach(@ret) {
        print "$_";
    }
    print "abort done\n";
    unlink("grid_abort.jobs");
}

sub count_jobs { # skaiciuojame kiek darbu sarase
    my $cnt = -1;
    open(F, "grid.jobs");
    while(<F>){
        $cnt++
    }
    close(F);
    return $cnt
}

sub get_output { # rezultatu parsiuntimas
    if (! -d "output") {
        mkdir("output")
    }
    `glite-wms-job-output --noint -i grid.jobs --dir output 2>/dev/null`
}

sub ask { # interaktyvus vartotojo apklausimas
    $what = shift;
    $where = shift;
    print $what;
    my $input = '';
    $input = <STDIN>;
    chop($input); # nukerpamas naujos eilutes simbolis
    if ($input) {
        $$where = $input;
    }
}

sub create_archive {
    unlink("grid.tar.gz");
    # sukuriamas esamo katalogo archyvas (be valdymo failu)
    my $arhcmd = 'tar czf grid.tar.gz *'.' --exclude="sietas.pl"'.
    " --exclude=\"$rezfile\" --exclude=\"grid.jobs\" --exclude=\"*.jdl\"".
    " --exclude=\"output\"";
    `$arhcmd`;
}

sub init_vars { # JDL sablonas.  600 = 10 val. (Naudotojo reikalavimas)
    $jdl = <<EOFF
```

```
Executable = "/bin/sh";
Arguments = "-c '/bin/tar zxf grid.tar.gz; rm -f grid.tar.gz; %s'";
StdOutput = "stdout-%s.txt";
StdError  = "stderr-%s.txt";
InputSandbox = {"grid.tar.gz"};
OutputSandbox = {"stdout-%s.txt","stderr.txt-%s","%s"};
Requirements = other.GlueCEPolicyMaxCPUTime > 600;

EOFF
;
}
# Pabaiga. sietas.pl
```

3 lentelė. Telkinių darbinių mazgų parametrai

Telkinys	**Operatyvioji atmintis, Kb**	**BogoMips**	**Branduolių skaičius**	**CPU, MHz**	**Mazgų skaičius**
ce.bg.ktu.lt	119744	1470	1	734	32
test.ce.grid.vgtu.lt	1034500	2794	2	1396	20
grid.mii.lt	1026368	6004	2	3000	20
ce.cyf-kr.edu.pl	8166024	4671	4	2333	558
atomas.itpa.lt	2068196	1594	2	796	10
grid6.mif.vu.lt	7875396	4784	4	2393	76
ce.grid.vgtu.lt	255752	1871	1	935	15
kriit.eenet.ee	516156	4000	2	2000	6
grid9.mif.vu.lt	63839104	2834	64	1394	68
g03n02.pdc.kth.se	2058548	5583	1	2795	66
spektras.itpa.lt	2068196	1594	2	796	44
pupa.elen.ktu.lt	1033460	5985	2	3000	12

* Techniniai duomenys lentelėje pateikiami apie *vieną* tipinį telkinio darbinį mazgą (WN).