Latvijas Universitāte

Fizikas un matemātikas fakultāte

Datorikas nodaļa

Interneta informācijas sistēmu slodzes un mērogojamības testēšana izmantojot bezmaksas rīkus

Maģistra darbs

Autors: *Edvīns Vēzis*

Vadītājs: *Andrejs Maruškins,*

LU Datorzinātņu maģistrs,

SIA „Dyna-IT" valdes priekšsēdētājs

Rīga, 2008

Anotācija

[LV]

Mūsdienu interneta informāciju sistēmām (IIS) ir jāapstrādā arvien lielāku informācijas apjomu, kā arī jānodrošina iespēju strādāt augošam lietotāju skaitam. Savukārt augošais bezmaksas un atvērtā pirmkoda rīku klāsts piedāvā jaunas iespējas programmatūras testēšanā. Autora mērķis ir iepazīties ar IIS slodzes un mērogojamības testēšanas metodiku un atrast bezmaksas rīkus, kas varētu palīdzēt liela mēroga IIS (vairāk kā 10 000 vienlaicīgu lietotāju) testēšanā. Darbā tika apskatīti rīki, kas varētu noderēt šim mērķim, un novērtēti pēc vairākiem kritērijiem. Tika izvēlēti divi no rīkiem, CLIF un The Grinder, un ieviesti efektīvai izmantošanai reālā projektā. Izstrādāta saskarne starp šiem rīkiem, kas ļauj izmantot labākās īpašības no katra rīka. Rezultātā ir iespējams vispusīgi novērtēt IIS veiktspēju, tai skaitā dažādu IIS resursu patēriņu.

Atslēgvārdi: interneta informācijas sistēma, slodze, mērogojamība, testēšana, atvērtā pirmkoda un bezmaksas rīki

Abstract

[ENG]

Modern internet information system (IIS) must be able to handle growing amount of information as well as increasing number of users. In the same time, growing range of open-source and free of charge tools provide good facilities in testing. Author's aim is to learn methods of load and scalability testing of IIS, and to find freely available tools for load testing big scale (more than 10 000 online users) system. In this work there were inspected tools usable for this purpose. They were rated at several parameters and picked several of them for further practical evaluation. Later two tools from best tools picked earlier were tested out and implemented in real project. There were developed interface between both tools, which made possible to use best features from each tool. In result IIS performance is well tested, also with different resource measuring.

Keywords: internet information system, load, scalability, testing, open-source and free of charge tools

Autoreferāts

Autors maģistra darba gaitā ir paveicis sekojošo:

- apskatījis interneta informācijas sistēmu testēšanas īpatnības;
- iepazinies ar metodikām slodzes un mērogojamības testēšanā;
- iepazinies ar pieejamajiem bezmaksas rīkiem, kas varētu noderēt slodzes un mērogojamības testēšanā;
- pēc izvēlētiem kritērijiem atlasījis labākos rīkus tālākai novērtēšanai reālā projektā;
- Izvēlējies divus rīkus ieviešanai reālā projektā un izstrādājis saskarni starp šiem rīkiem, to kombinētai un efektīvai izmantošanai;
- Veiksmīgi ieviesis šo rīku izmantošanu reālā projektā un aprakstījis iegūto pieredzi un rezultātus šajā darbā.

Satura rādītājs

IEVADS

Mūsdienu interneta informāciju sistēmām ir jāapstrādā arvien lielāks informācijas apjoms, kā arī jānodrošina iespēju strādāt augošam lietotāju skaitam. Aktuāli kļūst paredzēt sistēmas darbību šādos apstākļos un iespēju sistēmu attīstīt neapmierinošu rezultātu gadījumā, par cik tas tiešā veidā var ietekmēt komerciālās darbības attīstību un potenciālās peļņas gūšanu vai zaudēšanu. Viens no svarīgākajiem faktoriem pakalpojumu sniegšanas sfērā ir nodrošināmā servisa kvalitāte un ātrums. Informācijas sistēmu servisu ātrumu raksturo veiktspējas un stabilitātes jēdzieni. Viens no sistēmas stabilitātes un veiktspējas noturības raksturojošiem jēdzieniem ir sistēmas mērogojamība (scalability), kas ir sistēmas spēja pie nepieciešamības efektīvi palielināt savus resursus.

Dotā darba pirmajā autors cenšas apzināt metodes interneta informācijas sistēmas darbības novērtēšanai dažādos apstākļos, kas paredz arī iepriekš minēto slodzes lielumu un sistēmas konfigurāciju variācijas. Apskatīta slodzes un mērogojamības testēšana no teorētiskās puses.

ISS slodzes testēšanai parasti tiek izmantoti specializēti testēšanas rīki. Autors novēroja, ka pašlaik kļūst pieejams ievērojami plašs bezmaksas un atvērtā koda testēšanas rīku klāsts un ir tendence arvien biežāk programmatūras izstrādes kompānijās izmantot šādus rīkus. Pirms dotā darba uzsākšanas autors arī novēroja, ka principā nav pieejami nekādi esošo bezmaksas slodzes testēšanas rīku salīdzinājumi un novērtējumi. Tas rada grūtības rīka izvēlē.

Dotā darba otrajā daļā autors apskata publiski pieejamos bezmaksas testēšanas rīkus, kas varētu noderēt slodzes un mērogojamības testēšanā. Autora mērķis ir atrast rīkus, kas gala rezultātā spētu simulēt liela mēroga sistēmas darbību ar vairāk kā 10 000 vienlaicīgiem lietotājiem. Ņemot vērā šo un citus svarīgus kritērijus, tiek atlasīti labākie rīki šim mērķim.

Trešajā darba daļā autors apraksta testēšanas rīku izvēli un izmantošanu reālam projektam. Tiek apskatīta testējamā IIS, tās arhitektūra un funkcionalitāte. Tālāk aprakstīts divu izvēlēto rīku CLIF un The Grinder pielietošana, kā tiek realizēta abu šo rīku kombinēta darbība, kuru izstrādājis šī darba autors. Beigās tiek aprakstīta slodzes un mērogojamības testēšana un tās rezultāti šajā projektā.

1. METODOLOĢIJAS APSKATS

Šajā nodaļā sākumā aplūkotas tēmas, kas raksturīgas interneta informācijas sistēmu testēšanai, un vēlāk precīzāk aprakstītas metodes slodzes un mērogojamības testēšanai.

1.1. Interneta sistēmu testēšana

Interneta informācijas sistēmu testēšanas process parasti sastāv no vairākiem posmiem:

- Sistēmas analīze un lielāko riska apgabalu noteikšana;
- Testu plānošana – viena no svarīgākajām iemaņām testējot IIS ir efektīvi saplānot testus, ņemot vērā IIS daudzpusīgo funkcionalitāti un sarežģīto arhitektūru;
- Testu izpilde – IIS atšķirībā no tradicionālajām ir lielāks uzsvars uz dažiem testu veidiem kā drošība, mērogojamība utt;
- Rezultātu analīze – IIS svarīgi apzināt dažādo IIS komponenšu ietekmi uz sistēmu;

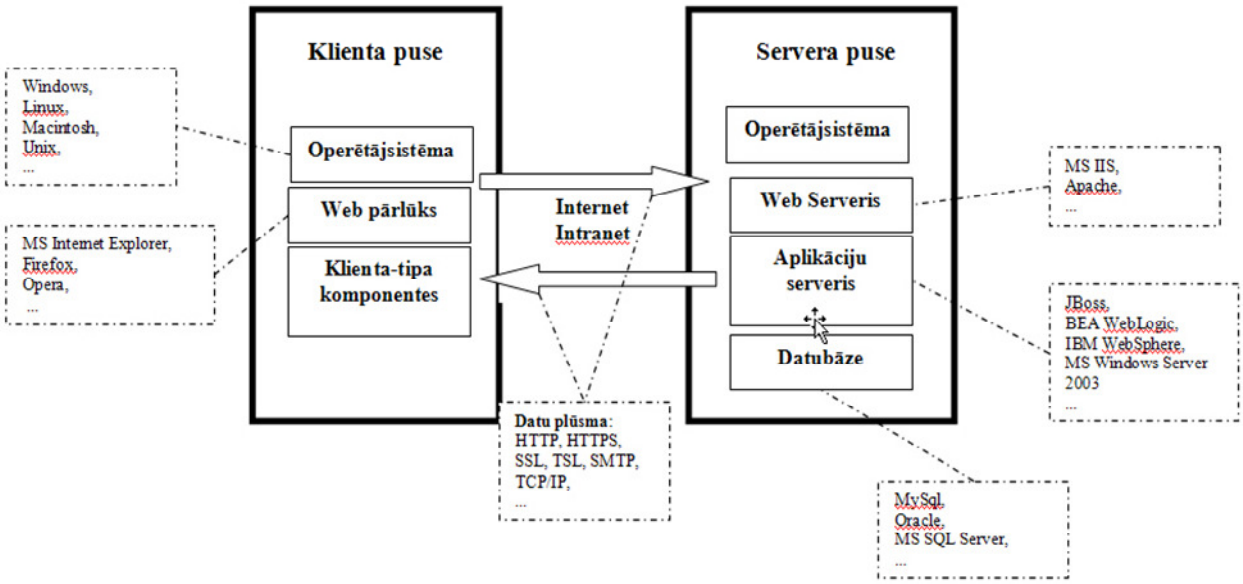
Sistēmas analīzē ietilpst zināšanas par tādā sfērām:

- Sistēmas funkcionālās prasības;
- Sistēmas darbības vide;

Klienta-servera sistēmas, uz kurām ir bāzētas interneta sistēmas, sastāv no tīkla un vairākiem datoriem: klienta datori un servera datori, kuri sūta pieprasītos datus klienta datoriem. Lielākajai daļai sistēmu lietotnēm par lietotāja saskarni kalpo interneta pārlūka programma.

Klienta-servera modelis un attiecīgi arī interneta informācijas sistēmas modelis nav viegli segmentējams. Tajā sistēmas procesus var veikt gan klienta, gan servera datori, kā arī servera puses procesi var tikt sadalīti starp vairākām fiziskām un programmatiskām daļām (lietojuma serveris, Web serveris, datubāzes serveris, tērzēšanas serveri, utt.).

Klienta-servera sistēmas arhitektūra - attēls 1.



Att. 1- Klienta-servera arhitektūra

Interneta sistēmas ir sarežģītākās nekā tradicionālās klienta-servera sistēmas. Tradicionālo sistēmu klienti parasti tika veidoti specifiskai platformai (Windows, Linux vai Macintosh, utt.), bet interneta sistēmu klienti parasti ir bāzēti uz interneta pārlūka programmu un var tikt darbināti dažādās platformās.

Interneta informācijas sistēmām iespējamas dažādas gan aparatūras, gan programmatūras kombinācijas. Sistēmas programmatiskā komplektācija var sastāvēt dažādās kombinācijās no sekojošām komponentēm:

- vairākām operētājsistēmām;
- vairākām programmatūras pakotnēm;
- vairākām programmatūras komponentēm;
- vairākiem serveru programmu tipiem, ražotājiem un modeļiem;
- vairākām interneta pārlūka programmām un versijām.

Internet sistēmu testēšanai ir specifiskas iezīmes:

- Interneta sistēmas darbojas komplicētākā vidē nekā lokālās sistēmas. Ja interneta sistēmas darbībā rodas kļūme, tad iemesls var slēpties gan sistēmas kodā, gan sistēmas kādas komponentes savietojamībā ar citu sistēmas komponenti, gan problēmās ar komunikācijām starp sistēmas komponentēm, kuras nav pieejamas izstrādātājiem.
- Bieži vien, liela daļa no sistēmas iekļauj sevī kompleksus trešās personas izstrādātāju produktus. Piemēram, interneta pārlūks, Java interpretators, datubāzes serveris, Web serveris.
- Par cik Web sistēma sastāv no tik daudzām daļām un komponentēm, kas sadarbojas savā starpā, ir iespējams radīt un pielietot testa rīkus, kas ļauj lasīt un modificēt dažādus starpposmu notikumus. Ir iespēja novērot un veidot ziņojumus starp klientu un serveri vairākos komunikāciju ķēdes posmos.

Interneta informācijas sistēmu testēšanas specifiskās sfēras ir:

- Web lietotāju saskarnes implementācija – lai arī notiek centieni standartizēt pārlūka programmu realizācijas un vienotu informācijas attēlojumu, tomēr joprojām eksistē pietiekoši daudz atšķirību starp dažādu pārlūkiem, piemēram, Internet Explorer,

FireFox un Opera.

- Sistēmas integrācija - IIS parasti sastāv no daudzām komponentēm (lietojuma servera, Web servera, datubāzes, e-pasta servera utt.) un tām savā starpā efektīvi jāsadarbojas;
- Servera un klienta uzstādīšana un konfigurācija – īpaši liela mēroga IIS parasti sastāv no daudziem serveru datoriem un/vai klasteriem, kas prasa lielus sagatavošanas darbus;
- Web-bāzēta lietotāju pamācība;
- Drošība;
- **Slodze un mērogojamība**;

Šajā darbā autors aplūko saraksta pēdējo punktu.

[1, 2]

1.2. Slodzes un mērogojamības testēšanas metodikas

Slodzes testēšana analizē, kā sistēma strādā ar lieliem datu apjomiem, lieliem aprēķiniem un procesiem. Slodzes testi parasti tiek automatizēti.

Telekomunikācijās un programmatūras izstrādē *mērogojamība* ir sistēmas, tīkla vai procesa īpašība, kas norāda spēju tikt galā ar pieaugošu slodzi un spēju atbilstoši palielināt resursus.

Tādējādi, mērogojamības testēšana ir cieši saistīta ar slodzes testēšanu. Viens no mērogojamības testēšanas posmiem ir slodzes testēšanas veikšana pie dažādām sistēmas konfigurācijām un resursiem.

[3, 4]

1.2.1. Slodzes testēšana

Pamatā slodzes testēšanā tiek simulēta daudzu lietotāju vienlaicīga darbība. Tas palīdz novērtēt daudz-lietotāju sistēmas veiktspēju apstākļos, kas ir pēc iespējas tuvāki paredzamai sistēmas lietošanai ekspluatācijas laikā. Slodzes testēšana parasti fokusējas uz:

- lielu datu apjomu apstrādi;
- daudzu procesu vienlaicīgu darbināšanu;
- sistēmas uzdevumu izpilde ilgstošā laika periodā.

Slodzes testēšana parasti notiek pēc funkcionālās testēšanas, kas nodrošina slodzes testu korektu izpildi.

[1]

1.2.2. Slodzes testēšanas pielietojums

Viena no interneta informācijas sistēmu galvenajām priekšrocībām ir iespēja vienlaicīgi daudziem lietotājiem piekļūt sistēmas resursiem. Tāpēc ir ļoti svarīgi novērtēt sistēmas spēju veikt svarīgākās funkcijas pie lielām lietotāju slodzēm.

Slodzes testu primārais mērķis ir noteikt sistēmas veiktspēju pie dažāda lietotāja skaita un noteikt to lietotāju skaitu, pie kura sistēmai ir pieņemama veiktspēja. Parasti šajos testos tiek simulēti simtiem vai tūkstošiem vienlaicīgu lietotāju darbība noteiktā laika periodā. Vēlams ir, lai minimālā sistēmas konfigurācija un maksimālie aktivitāšu līmeņi ir noteikti pirms testēšanas sākuma.

Slodzes testēšanai ir specifisks apakš veids - *Stresa testēšana.* Tā novērtē sistēmas darbību, kad to pakļauj ekstremālai slodzei, piemēram, lai novērtētu sistēmas stabilitāti pie maksimāliem vienlaicīgiem lietotāju daudzumiem. Stresa testu primārais mērķis ir noteikt vai tādās situācijās sistēma nepārstāj funkcionēt vai arī tā spēj veiksmīgi atjaunot savu darbību. Tādas situācijas palīdz atklāt sistēmas vājās vietas. [2]

Slodzes testi parasti simulē regulāru lietotāju darbību. Viens no šo testu rezultātu kritērijiem ir *reakcijas laiks*. Reakcijas laiks ir laika daudzums, kas lietotājam jāgaida pēc pieprasījuma uz sistēmas reakciju.

Slodzes testi izmanto reālu vai simulētu darbību, lai noslogotu sistēmas un citus saistītus resursus, ieskaitot:

- Atmiņu (operatīvo atmiņu, virtuālo, cietā diska, steka vietu)
- Centrālā procesora (CPU) laiku
- TCP/IP adreses

- Tīkla pārraides resursus
- Failu apstrādi

Šie testi var identificēt tādas sistēmas kļūdas:

- Aparatūras izraisītas programmatūras kļūdas
- Atmiņas darbības kļūdas
- Datubāzes ierakstu bloķēšanās
- Vienlaicīgu sistēmas darbības *pavedienu* (*threads*) problēmas

1.2.3. Slodzes prasību novērtēšana

Slodzes testēšanas galvenie testējamie jautājumi ir:

- Vai sistēma varēs tikt galā ar pieaugošu tīklā nosūtāmo datu daudzumu, nepasliktinot rādītājus reakcijas laikiem, drošībai, stabilitātei un precizitātei?
- Kurā brīdī veiktspēja samazināsies un kāda komponente būs tajā vainojama?
- Kādu iespaidu veiktspējas samazināšanās atstās uz kompānijas tirgus aspektu un tehniskā atbalsta izmaksām?

Katrs no iepriekšējiem jautājumiem prasa veikt mērījumus testējamajai sistēmai. Sistēmas atribūti, tādi kā reakcijas laiks, var tikt novērtēti pielietojot dažādus darba slodzes scenārijus sistēmai. Balstoties uz ievāktajiem datiem var veikt secinājumus. (Piemēram, kad paralēlo lietotāju skaits sasniedz X, reakcijas laiks ir vienāds ar Y. Tādējādi, sistēma nevar uzturēt vairāk kā X paralēlu lietotāju.) Sarežģījumi rodas, kad, lai gan lietotāju skaits X nemainās, tomēr Y var mainīties atkarībā no dažādām lietotāju darbībām. Piemēram, 1000 paralēlu lietotāju pieprasījums pēc 2KB HTML lapas dos ierobežota lieluma reakcijas laikus; kamēr reakcijas laiki var ļoti variēt, ja tiem paši 1000 paralēlie lietotāji vienlaicīgi veic transakcijas, kas izraisa ievērojamu servera puses apstrādi. Izveidot korektu darba slodzes modeli, kas attēlo šādu reālu izmantošanu, nav viegls uzdevums.

Lai labāk saprastu kā pieaugoša slodze un attiecīgi arī pieaugoši reakcijas laiki var izraisīt kompānijas peļņas zaudējumos, paņemsim vienkāršotu piemēru. Pieņemsim, ka e-biznesa mājas lapa apstrādā 300 000 transakciju dienā. Tas vidēji nozīmē 3,47 transakcijas sekundē. Pieņemsim, ka mārketinga pētījums ir atklājis sekojošo:

- Transakcijas reakcijas laiks ir pieņemams, ja tas nepārsniedz 4 sekundes;
- Ja transakcijas reakcijas laiks ir lielāks par 4, bet mazāks par 9 sekundēm, 30% lietotāju pārtrauc savus darījumus;
- Ja reakcijas laiks ir lielāks par 8, bet mazāks par 10 sekundēm, 60% lietotāju pārtrauc transakciju;
- Ja transakcijas laiks ir lielāks par 10 sekundēm, vairāk kā 90% lietotāju pārtrauc transakciju.

Pieņemsim, ka nākamajos 6 mēnešos transakciju skaitam jāpieaug par 25 līdz 75 % un vienas transakcijas potenciālā peļņa ir 1$. Vadība grib uzzināt, kā veiktspēja ietekmēs kompānijas peļņu, ja transakciju skaits dienā pieaugs. Tiek veikts slodzes tests un tiek noteikts, ka sistēma nespēj apstrādāt tādus slodzes pieaugumus bez reakcijas laika pieauguma. Sekojoši, pieaugs pārtraukto un neveiksmīgo transakciju skaits. Ja transakciju skaits pieaugs kā gaidīts, kompānijas potenciālie peļņas zaudējumi būs no 112 00 līdz 472 000 $ dienā.

Šis piemērs parāda, ka lielām sistēmām nespēja strādāt pie palielinātas slodzes var nest ļoti ievērojamus zaudējumus.

1.2.4. Slodzes testu plānošana

Sekojošās ir galvenie faktori, kas jāsaprot un jānovērtē slodzes testu plānošanā:

- Plānoto lietotāju daudzums. Tas var būt pietiekami sarežģīti, noteikt lietotāju skaitu, ko sistēma uzturēs, jo lietotāju aktivitāte var mainīties.
- Apmierinošas veiktspējas definēšana.
- Datu analīze un koriģējošo darbību plānojums.

Slodzes testēšana iekļauj sevī trīs galveno elementu novērtēšanu:

- Sistēmas vide un pieejamie resursi;
- Darba slodze;
- Sistēmas reakcijas laiks;

Darba slodze ir sistēmas veikto datu apstrādes un plūsmas kvantitāte. Tā pamatā veidojas no vienlaicīgo lietotāju skaita, datu daudzuma datubāzē un datu apstrādes sarežģītības.

[1, 2]

1.2.5. Slodzes testu veidi

Slodzes testēšanas dažādiem mērķiem var izdalīt vairākus testu veidus, kas atšķiras savā starpā gan pēc slodzes apjoma un testu ilguma, gan arī pēc slodzes variācijām viena testa laikā.

Līdzenais tests (Flat test) – tiek veikts ar konstantu, iepriekš definētu slodzes apjomu. Visi virtuālie lietotāji tiek ielādeti vienlaicīgi un darbināti konstantu laiku. Tas nodrošina akurātus un atkārtojamus mērījumus. Šis testa veids labi noder dažādu IIS versiju novērtēšanai un salīdzināšanai.

Pieaugošās slodzes tests (Ramp-up test) – lietotāju skaits tiek pakapeniski palielināts. Šādi testi ir vērtīgi, lai noskaidrotu aptuvenu maksimālo lietotāju skaitu, ko pēc tam var izmantot līdzenajos testos. Šis testa tips ļauj redzēt sistēmas veiktspējas izmaiņas pie mainīgas slodzes.

Stabilitātes tests (Soak test) – ilgsoši testi ar nemainīgu paralēlo lietotāju skaitu. Šādi testi ļauj atklāt veikstējas degradāciju ilgstošā laika periodā, kas rodas nepareizas atmiņas izmantošanas, parmērīgas neizmantoto resursu savākšanas (garbage collection) vai citu sistēmas problēmu rezultātā. Vēlams darbināt ar mērēnu lietotāju skaitu, lai priekšlaicīgi nepārslogotu sistēmu.

Augtas-zemas slodzes tests (Peak_Rest tets) – šie testi ir pieaugošās slodzes un stabilitātes testu apvienojums. Galvenais mērķis šim testu tipam ir noteikt, cik veiksmīgi sistēma spēj atjaunot darbību pēc augstas slodzes periodiem. Slodze periodiski tiek paaugstināta līdz maksimumam, tad tiek samazināta līdz ļoti mērenai un tad atkal paaugstināta un samazināta.

1.2.6. Mērogojamība

Sistēma, kuras veiktspēja pēc aparatūras resursu pievienošanas pieaug proporcionāli pievienotajiem resursiem, tiek saukta par *mērogojamu sistēmu*. Lai gan mērogojamība ir svarīga, tomēr tās pazīmes un to raksturojošie parametri parasti ir nosakāmi saistībā ar citām sistēmas īpašībām.

Daudzām distributīvām sistēmām vajag būt mērogojamām. Tām vajadzētu būt spējīgām darboties dažādos mērogos tādās mērvienībās kā lietotāju un servisu skaits, uzglabājamo un apstrādājamo datu daudzums, aktivitāšu biežums, mezglu (serveru) skaits, ģeogrāfiskais

pārklājums, tīkla izmērs un datu glabāšanas ierīces. Mērogojamība nozīmē ne tikai spēju darboties dažādās konfigurācijās, bet darboties efektīvi un ar pietiekošu servisu kvalitāti.

Piemēram, mērogojama tīkla transakciju sistēma vai datubāzes pārvaldības sistēma ir tāda, kura var tikt uzlabota lielāka skaita transakciju apstrāde, pievienojot jaunus procesorus, ierīces un datu nesējus, un kura var tikt uzlabota viegli un pārredzami bez sistēmas darbības pārtraukšanas.

Ir divi mērogošanas veidi:

- Vertikālā mērogošana – pievieno resursus vienam sistēmas mezglam, parasti jaunus CPU vai atmiņu.
- Horizontālā mērogošana –pievieno jaunus mezglus sistēmai. Piemēram, pievieno jaunu serveri distributīvai sistēmai.

Datoru cenas samazinās un jauda turpina pieaugt. Salīdzinoši lētas sistēmas var tikt izmantotas lielas jaudas skaitļošanas sistēmās, tādās kuras senāk varēja darbināt tikai ar superdatoriem. Simtiem datoru var nokonfigurēt vienā klasterī, lai iegūtu summētu jaudu.

Abām pieejam ir savas priekšrocības un trūkumi. Lielāks datoru skaits nozīmē palielinātu pārvaldības sarežģītību, kā arī sarežģītāku programmēšanas modeli un rūpes par caurlaidību un reakcijas laiku starp mezgliem. Tomēr cenu atšķirība starp abiem modeļiem ar vien vairāk sliecas par labu horizontālajai mērogošanai.

1.2.7. Mērogojamības novērtēšana

Dažādas mērogojamības metrikas ir izstrādātas priekš paralēlām skaitļošanām, lai noteiktu dotā algoritma efektivitāti darbinot uz dažāda izmēra platformām, un lai salīdzinātu algoritmu mērogojamību. Šīs metrikas pieņem, ka programma patstāvīgi darbojas uz k procesoriem ar doto arhitektūru, un ka izpildes laiks T novērtē veiktspēju. Trīs veidu metrikas tiek minētas: paātrinājuma metrika, efektivitātes metrika un mērogojamības metrika. Sekojošās definīcijas minētajām metrikām dod skaidrojumu, lai gan dažādiem autoriem nedaudz var atšķirties:

- Paātrinājums S novērtē, kā padarāmā darba izpildes ražīgums pieaug ar procesoru skaita k pieaugumu salīdzinājumā ar vienu procesoru. „Ideāla" vērtība ir lineārs paātrinājums $S(k) = k$;
- Efektivitāte E novērtē darba ražīgumu uz procesoru (tas ir, $E(k) = S(k)/k$). Ideālā

vērtība ir viena vienība, 1;

- Mērogojamība $M(k1, k2)$ no viena mēroga *k1* uz citu mērogu *k2* ir attiecība divu gadījumu efektivitātes vērtībām, $M(k1, k2) = E(k2)/E(k1)$. Tam arī ideālā vērtība ir viena vienība, 1;

Šādi parametri, protams, nav pietiekami, lai novērtētu mūsdienīgu informācijas sistēmu. Piemēram, produktivitātes novērtēšanu vajag paplašināt, ņemot vērā arī darbības kvalitāti. Arī sistēmas „izmērs" ir sarežģītāks lielums nekā tikai procesoru skaits. Nāk klāt tādi lielumi kā simetriskie daudz-procesoru mezgli, replikāciju servisi, alternatīvie tīkli, dažādu tipu un cenu procesori utt. Jēdziens „izmērs" kļūst daudz-dimensiju.

Vēl viens parametrs ir *izmaksas* (cost). Šeit bez procesoru izmaksām, vajag ņemt vērā tādas kā datu glabāšanas un pārraides izmaksas, kā arī programmatūras licenču, un, iespējams, pārvaldes un apkalpošanas izmaksas.

Distributīvas sistēmas mērogošanas stratēģija ir komplicētāka par vienkāršu procesoru, datu vietņu vai pārraides kanālu pievienošanu. Tā, piemēram, var sevī iekļaut programmatūras servisu vai datu vietņu replikāciju vai komunikāciju mehānismu modifikāciju. Precīzi formulētai mērogošanas stratēģijai jābūt daļai no metrikas definīcijas.

Grāmatā [5] piedāvātā mērogojamības metrika ir bāzēta uz *produktivitāti*. Ja produktivitāte tiek saglabāta pie mēroga izmaiņām, tad sistēma tiek uzskatīta par mērogojamu. Ir doti trīs parametri:

- d(k) = caurlaidība atbildēs/sekundē pie mērogojuma *k*;
- f(k) = vidējā katras atbildes *vērtība*, aprēķināta no servisa kvalitātes pie mērogojuma *k*;
- C(k) = izmaksas pie mērogojuma (k), izteiktas kā darbināšanas izmaksas sekundē, lai būtu vienās vienībās ar *d*;

Produktivitāte F(k) tad ir atgrieztā vērtība sekundē un izdalīta ar izmaksām sekundē:

$F(k) = d(k) * f(k) / C(k);$

Tālāk sistēmas mērogojamības metrika divos mērogojumos ir definēta kā attiecība starp to produktivitātēm:

M(k1, k2) = (F(k2)) / (F(k1)).

Mērogojamības testēšana ir cieši saistīta ar slodzes testēšanu. Viens no mērogojamības testēšanas posmiem ir slodzes testēšanas veikšana pie dažādām sistēmas konfigurācijām un resursiem.

[5, 6, 7]

2. TESTĒŠANAS RĪKI

Interneta informācijas sistēmas operē dinamiskā vidē. Reizēm testēšanas rīki ir nepieciešami, lai papildinātu manuālo testēšanu. Daži testu tipi (piemēram, slodzes un veiktspējas) būtu nepraktiski bez rīku palīdzības, kas var simulēt tūkstošu lietotāju darbību. Dažādu rīku vērtība ir atkarīga no specifiskām testēšanas vajadzībām, budžeta un personāla ierobežojumiem.

Šajā darbā atbilstoši tēmai uzmanību autors pievērš rīkiem, kas varētu noderēt slodzes un mērogojamības testēšanā.

Tādiem rīkiem vajadzētu ļaut simulēt tūkstošiem lietotāju darbību, kuri piekļūst interneta informācijas sistēmai pieprasot datus un veicot transakcijas, kā arī citas e-biznesa aktivitātes. Virtuālā slodze varētu simulēt arī dažādus interneta pārlūkus un to versijas, kā arī tīkla caurlaidību. Kamēr simulētā slodze tiek pielieto serverim, tiek uzkrāti veiktspējas dati un atrādīti vairākos pārskatāmos ziņojumu formātos.

Tādi rīki ģenerē testu skriptus ierakstot lietotāju aktivitātes un kombinējot tās ar skriptu valodu. Tie var radīt vairākus sistēmas darbības pavedienus, katrs pavediens var veikt specifisku testa skriptu vai scenāriju, lai simulētu reālas situācijas pieprasījumus serverim.

Šī darba mērķis bija atrast rīkus, kas spētu veikt nopietnus testus liela mēroga sistēmai. Ar liela mēroga sistēmu autors apzīmē sistēmu, ar kuru plānots strādāt vismaz 10 000 (desmit tūkstošiem) paralēliem lietotājiem.

Darba autors centās vismaz virspusēji iepazīties ar visiem publiski pieejamajiem rīkiem un izvēlēties piemērotākos.

2.1. Atlasītie bezmaksas testēšanas rīki

Šeit apskatīti nopietnākie rīki, kas pieminēti publiski pieejamos materiālos un par kuriem ir pozitīvas atsauksmes. Dots katra rīka neliels apraksts un novērtējums ar priekšrocību un trūkumu atzīmēšanas. Rīki tika novērtēti pēc vairākiem salīdzinošiem kritējiem, tai skaitā:

- iespēja darbināt testus uz vairākiem attālinātiem datoriem;

- iegūstamās informācijas kvalitāte un dažādība, iespēja iegūt gatavas atskaites nevis tikai žurnālu failus (log) un .csv tipa failus ar uzkrātiem datiem;
- iespēja savākt informāciju automātiski no attālinātiem datoriem;
- iespēja automātiski ierakstīt testus;
- testu skriptu pielāgojamība – sevišķi sistēmas izstrādēs laikā, kad mainās sistēmas funkcionalitāte un prasības, ir svarīga iespēja ērti modificēt esošos testu skriptus;
- lietotāja saskarne;
- dokumentācijas kvalitāte – kā izrādījās rīku pētījumu laikā, tā ir ļoti svarīgs resurss, īpaši lietotājiem ar mazu pieredzi dotajā sfērā, jo funkcijām bāgātākie rīki bieži izrādījās sarežģīti lietošanā.

Viens no svarīgākajiem faktoriem rīka atlasei ir tā iespēja darbināt testus uz vairākiem attālinātiem datoriem. Tas nodrošina rīkam daudzkārt lielāku jaudu un iespēju simulēt vairāk vienlaicīgus lietotājus, kas ir nepieciešams, ja jāspēj simulēt vairāk kā 10 000 lietotāju.

2.1.1. The Grinder

(http://grinder.sourceforge.net/) Java slodzes testēšanas ietvars (framework) brīvi pieejams zem BSD stila atvērtā koda licences. Tas ļauj no grafiskas konsoles pārvaldīt testu skriptu izpildīšanos vairākos procesos uz vairākiem datoriem. Tas ļauj testēt HTTP servisus, kā arī dod iespēju automātiski ierakstīt HTTP skriptus. Skripti tiek rakstīti Jython valodā.

Rīks sevī ietver trīs veidu procesus (jeb programmas): strādnieka (worker) process, aģenta process, un konsole (skat. att. 2). Katram no procesu tipiem ir sekojošie pienākumi:

- Strādnieka process – interpretēt Jython testa skriptu un veikt testus izmantojot noteiktu skaitu strādnieka procesa pavedienus.
- Aģenta process – pārvalda strādnieku procesus.
- Konsole – Koordinē citus procesus. Savāc un parāda statistikas. Veic skriptu labošanu un izplatīšanu.

  Par cik rīks ir rakstīts Java valodā, tad katrs no procesiem ir Javas virtuālā mašīna (JVM).

  Skriptu automātiska ierakstīšana

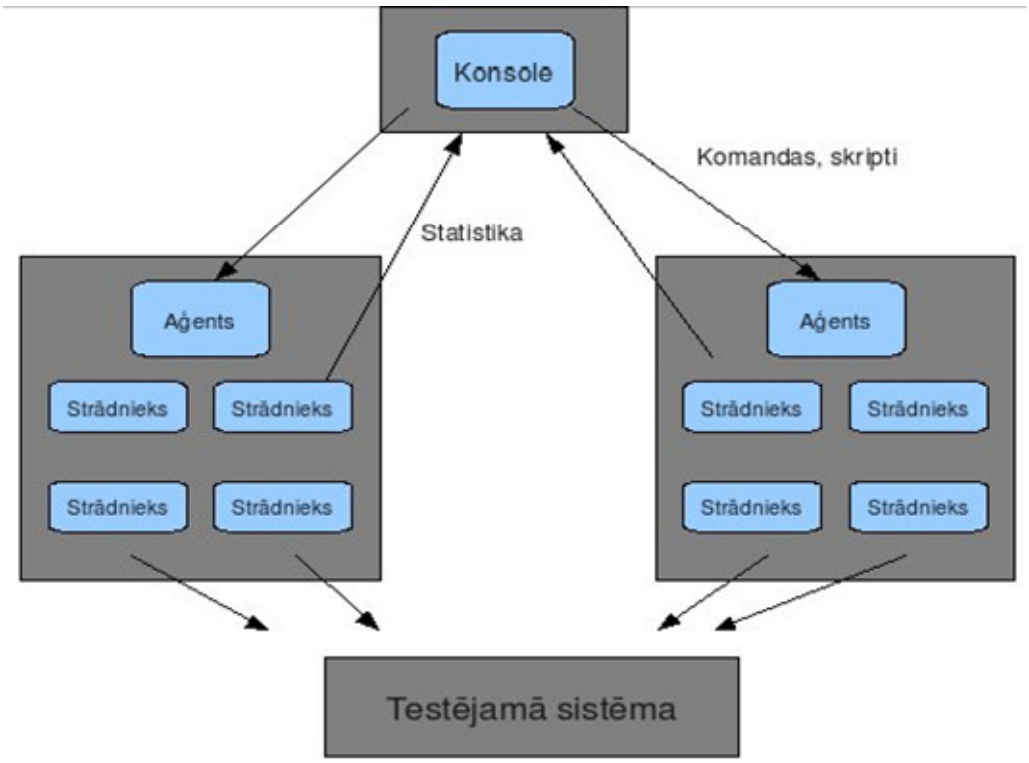
Skriptu ierakstīšana notiek izmantojot TCPProxy, kas ir HTTP starp-programma. Tā ir jānorāda interneta pārlūka uzstādījumos.

Palaišana

Lai palaistu konsoli, aģenta procesu vai TCPProxy starp-progammu nepieciešams uzstādīt vairākus vides mainīgos un palaist attiecīgos .jar failus. Tas nozīmē, ka nepieciešama neliela konfigurācija. Automātiskai minēto darbību veikšanai Windows vidē izmanto .cmd failus, bet Linux vidē - .sh failus.

Novertējums:

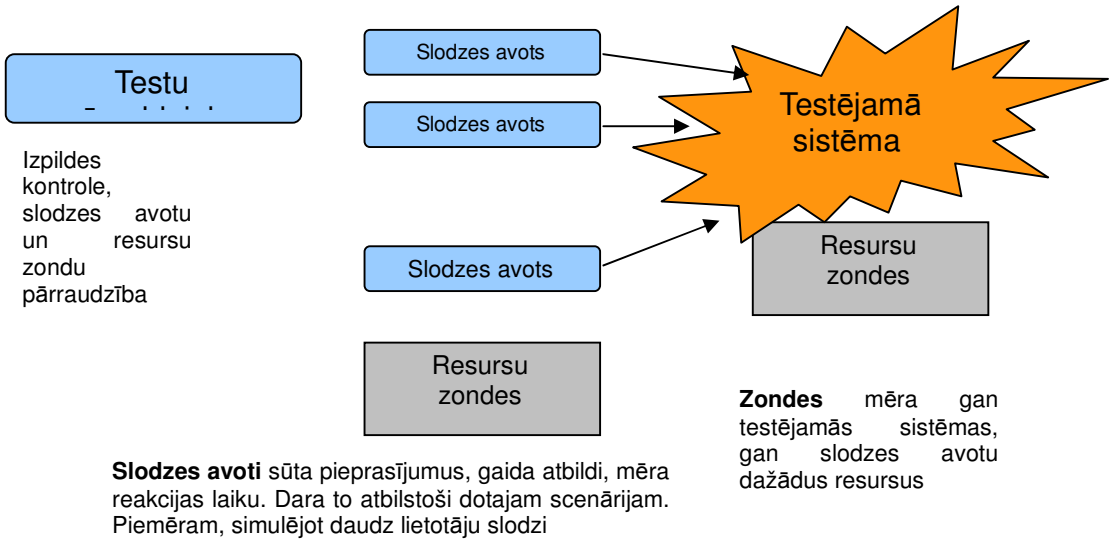
1. Slodzes sadalīšana - darbināms uz attālinātām iekārtām.
2. Slodzes ģenerētāji – JVM procesi, kurus var palaist ar pēc kārtas ar intervāliem.
3. Sistēmas resursu novērošana – nav pieejama;
4. Lietotāja saskarne – Vadības konsolē iespējams palaist, apturēt slodzes ģenerētājus. Tur arī parāda tekošos testu datus, kas nāk no slodzes ģeneratoriem.
5. Dokumentācijas kvalitāte – pieejama laba dokumentācija.
6. Iespēja automātiski ierakstīt skriptus – ir iespējams.
7. Servera atbildes korektuma pārbaude – ir pieejams, var izmantot regulārās izteiksmes.
8. Skriptu pielāgojamība - testa skriptus iespējams rakstīt ērtā, Python un Java valodām līdzīgā, skriptu valodā Jython.
9. Skriptu parametrizēšana – ir iespējama ar iekodētām konstantēm.
10. Automātiska informācijas savākšana no attālinātiem datoriem – konsole var ievākt tekošo informāciju no slodzes ģeneratoriem.
11. Atskaišu sagatavošana – ir pieejama viena atskaite, kas tiek ģenerēta no konsoles ievāktajiem datiem.



Att. 2. – The Grinder rīka arhitektūra

2.1.2. CLIF

(http://clif.objectweb.org/) modulāra un pielāgojama distributīvas slodzes testēšanas platforma (skat. att. 3.). To var pielietot jebkurai sistēmai, kas ir sasniedzama no Javas programmas (HTTP, DNS, TCP/IP...). CLIF piedāvā 3 lietotāja interfeisus (Swing vai Eclipse GUI, komandas rinda), lai izvietotu, kontrolētu un novērotu slodzes injektoru (slodzes avotu) kopu un resursu patēriņa zondes (CPU, atmiņas...).



Att. 3. – rīka CLIF arhitektūra

Palaišana

Gan rīka kompilācija, gan rīku sastāvdaļu startēšana notiek izmantojot konstruēšanas rīku Ant.

Skriptu automātiska ierakstīšana

Ir iespēja izmantot atsevišķu funkcionālās testēšanas atvērtā koda rīku MaxQ, kas kalpo kā HTTP starp-serveris, lai ierakstītu HTTP pieprasījumus uz serveri.

Novertējums:

1. Slodzes sadalīšana - darbināms uz attālinātām iekārtām. Iespējams injektorus sūtīt no vadības konsoles uz slodzes avotu datoriem pa tīklu un automātiski aktivizēt un apturēt;
2. Slodzes ģenerētāji - iespējams veidot pielāgotus slodzes ģenerētājus;
3. Sistēmas rerursu novērošana - pieejamas resursu zondes dažādu sistēmas resursu novērošanai un analīzei;
4. Lietotāja saskarne – pieejamas trīs veidu lietotāju saskarnes:
   - Eclipse izstrādes vides spraudņa veidā. Šī ir vis funkcionālākā CLIF rīka saskarne.
   - Atsevišķa grafiskā konsole, kas realizēta izmantojot Java Swing bibliotēkas; nav tik funkcionāla kā Eclipse bāzētā;
   - komandas rinda.
5. Dokumentācijas kvalitāte – pieejama diezgan plaša dokumentācija, bet nav pilnīga.

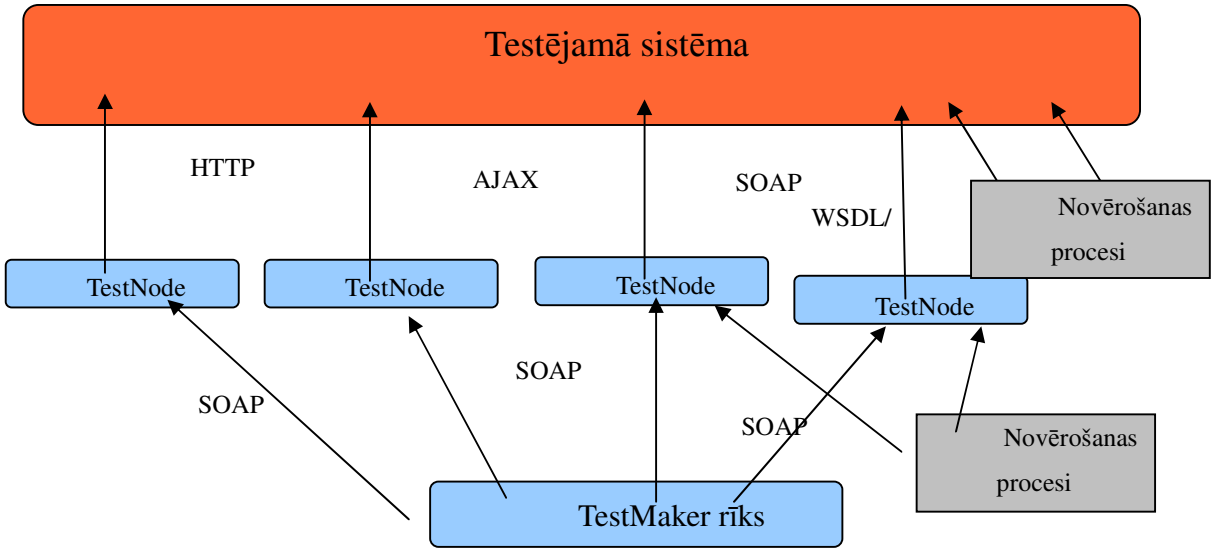
Mīnuss ir arī tas, ka dokumentāciju orientēta vairāk uz pieredzējušu lietotāju un daudzas lietas nav pārāk saprotami aprakstītas. Prasīja salīdzinoši ilgu laiku apguvei.

6. Iespēja automātiski ierakstīt skriptus – ir iespējams, izmantojot atvērtā koda rīku MaxQ, kas darbojas kā starpserveris HTTP pieprasījumu reģistrēšanai.
7. Servera atbildes korektuma pārbaude – nav paredzēta.
8. Skriptu pielāgojamība – ierakstītie skripti ir labojami XML formātā.
9. Skriptu parametrizēšana – nav iespējama.
10. Automātiska informācijas savākšana no attālinātiem datoriem -
11. Atskaišu sagatavošana -

2.1.3. TestMaker

(http://www.pushtotest.com/) bezmaksas atvērtā koda rīks Web lietojumu veiktspējas, mērogojamības un funkcionālajai testēšanai. Ietvars un rīks testa aģentu izveidei un darbināšanai, kuri simulē sistēmas lietotāju darbību. Tas piedāvā uz XML bāzētu skriptu valodu un testa objektu bibliotēku testa aģentu izveidei. Ietver sevī iespējas pārbaudīt un kontrolēt e-pastu sistēmas izmantojot SMTP, POP3, IMAP protokolus. Uz Java bāzēts rīks.

Rīks ļauj veikt distributīvus testus uz vairākiem testu mezgliem (TestNode) (skatt. att. 4.).



Att. 4. – rīka TestMaker arhitektūra

TestNode mezgls ir izpildes vide realizēta programmēšanas valodā Java un darbojas visur, kur izpildās Java programmas. TestMaker vadības pults sazinās ar TestNode izmantojot SOAP protokolus caur portu 80, kas nozīmē ka parasti nav vajadzības mainīt ugunsmūra iestatījumus.

Rīks piedāvā arī novērošanas (monitoring) iespējas. Novērotāju procesi seko līdzi procesoru, tīkla, un atmiņas lietojumam un ir paplašināmi citu testējamās sistēmas resursu novērošanai.

Priekšrocības:

- Pieejami novērošanas procesi dažādu sistēmas resursu kontrolei;
- Iespējams darbināt uz attālinātām iekārtām.

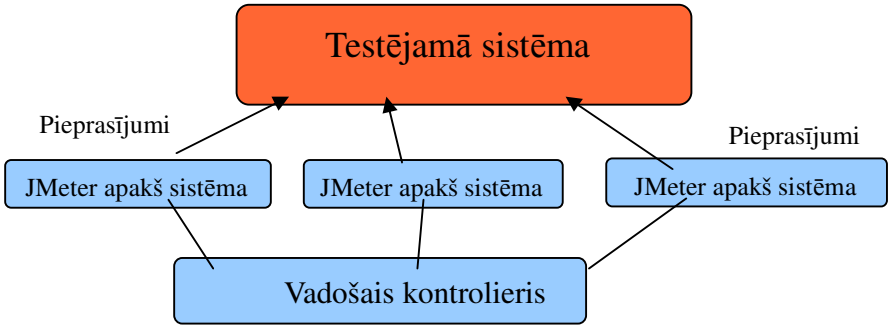
Trūkumi:

- Bezmaksas tikai līdz 200 lietotāju simulācijai;

2.1.4. Jmeter

(http://jakarta.apache.org/jmeter/) Java bāzēts rīks no Apache Software Foundation slodzes un veiktspējas testēšanai. Sākotnēji izstrādāta Web lietojumu testēšanai, bet ir paplašinājusies līdz citām testa funkcijām; var tikt lietota veiktspējas testēšanai gan statiskos, gan dinamiskos resursos (failos, servletos, Perl skriptos, Java Objektos, datubāzēs, FTP serveros un citur). Var lietot lai simulētu lielu slodzi uz servera, tīklā vai objektam. Var veikt grafisku analīzi.

Rīks darbojas tādā veidā, ka viens vadošais kontrolieris iniciē testus uz vairākām apakš sistēmām (skat. att. 5.).



Att. 5. – rīka JMeter arhitektūra

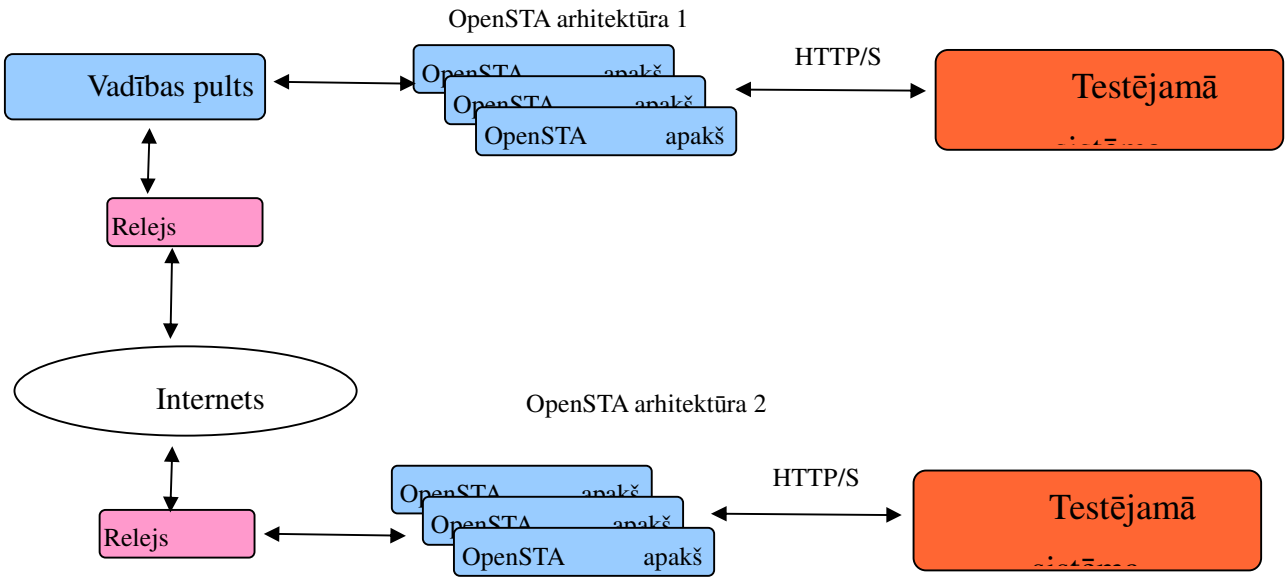
Priekšrocības:

- Iespējams darbināt uz vairākām attālinātām iekārtām.

2.1.5. OpenSTA

(http://www.opensta.org/) Bezmaksas atvērtā koda Web slodzes/stresa testēšanas rīks. Izmanto distributīvās programmatūras arhitektūru bāzētu uz CORBA. Tekošais rīku klāsts var veikt HTTP un HTTPS slodzes testus no Win32 platformām. Testi var sastāvēt no skriptiem un kolektoriem. Skripti definē operācijas, kuras izpilda virtuālie lietotāji. Kolektori definē SNMP, NT Perfomance un citu datu kopu, ko iegūst laižot testu.

Rīks ļauj veikt testu izpildi uz attālinātiem datoriem. Tas tiek panākts izmantojot Web Relay Daemon iespējas, kas ļauj OpenSTA arhitektūras CORBA-bāzētās komunikācijas pārraidīt internetā. Tiek izmantota Web Relay Daemon arhitektūra (skat. att. 6.).



Att. 6. – rīka OpenSTA arhitektūra

Priekšrocības:

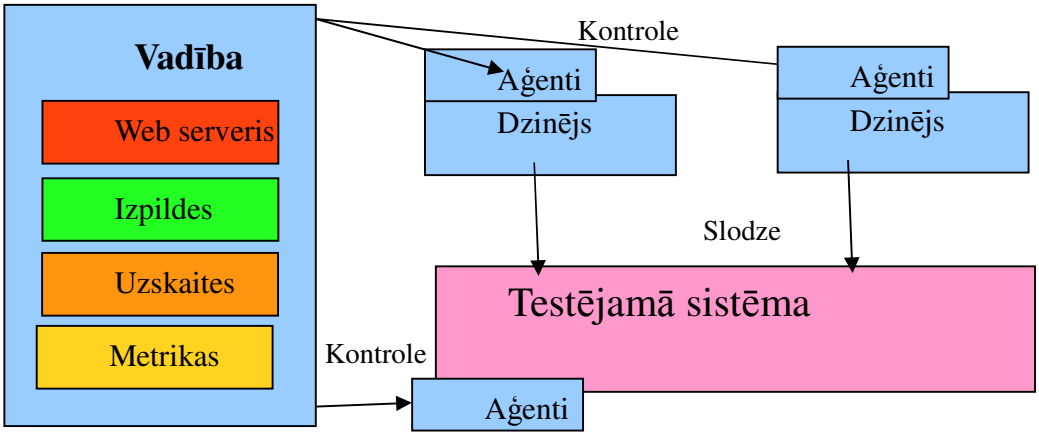
- Iespējams darbināt uz attālinātām iekārtām;
- Iespējams darbināt testus uz attālinātiem datoriem, kas nav lokālajā tīklā ar vadības pulti, bet ir saistīti ar interneta palīdzību.

Trūkumi:

- Rīks darbojas tikai Microsoft Windows vidē.

2.1.6. Faban

(http://faban.sunsource.net/) Rīks izstrādāts kompānijā Sun. Rīks nodrošina Web saskarni testu darbināšanai (skat. att 7.).



Att. 7. – rīka Faban arhitektūra

Vadības pults satur Web serveri, kas nodrošina piekļuvi Faban arhitektūrai caur Web saskarni. Tas nodrošina visas lietotāja saskarnes testu izveidošanai, kontrolei, kā arī rezultātu apskatei. Izpildes rinda ir galvenais dzinējs, kas kontrolē testu izpildi. Uzskaites serveris saņem izpildes laika ziņojumus no visām rīka komponentēm. Vadības pults var arī nodrošināt dažādas metrikas.

Faban aģenti darbojas gan uz dzinēju sistēmām, gan uz testējamās sistēmas. Tie ļauj darbināt dzinējus, kā arī izpildīt individuālas komandas serveru palaišanai, apstādināšanai, konfigurēšanai un sistēmas informācijas ievākšanai.

Priekšrocības:

- Iespējams darbināt uz vairākām attālinātām iekārtam;
- Iespējams ievākt daudzveidīgu informāciju par sistēmu un veidot dažādas metrikas.

2.2. Citi slodzes testēšanas rīki

Šeit aprakstīti bezmaksas testēšanas rīki, kas dažādu faktoru dēļ netika atlasīti uzstādītajam mērķim, bet kas arī varētu noderēt vienkāršākos gadījumos.

Pylot - (http://www.pylot.org/) Bezmaksas atvērtā koda rīks web servisu veiktspējas un mērogojamības testēšanai. Ģenerē paralēlu HTTP slodzi. Testa piemērus definē XML failos, specificē pieprasījumus – adresi, metodi, saturu, utt un verifikāciju. Verifikācija notiek salīdzinot ar regulārajām izteiksmēm un ar HTTP statusa kodiem. Novēro un palaiž testus no grafiskas lietotāja saskarnes. Tiek rādīta statistika un kļūdu ziņojumi reālā laikā.

Trūkumi:

- Nav iespējams automātiski ierakstīt testa piemērus.
- Testus iespējams darbināt tikai no lokālā datora.

Httperf - (http://www.hpl.hp.com/research/linux/httperf/) Bezmaksas atvērtā koda Web serveru veiktspējas/novērtēšanas rīks no HP Research Labs. Nodrošina pielāgojamu rīku dažādu HTTP darba slodžu ģenerēšanai. Rīks darbojas tikai no komandrindas, ar ļoti nepārskatāmu lietotāja saskarni.

Trūkumi:

- Vāja lietotāja saskarne;
- grūti definējami dažādie slodzes testu parametri;

Jcrawler – atvērtā koda Web lietojumu stresa testu rīks. Rīks ir ar ložņāšanas/izpētes funkciju. Tam var iedot sākuma adrešu kopu un tas sāks staigāt caur visām adresēm, kuras var atrast savā ceļā, ģenerējot slodzi Web lietojumam.

Trūkumi:

- nav konkrēti definējami slodzes veidi;
- testus iespējams izpildīt tikai no viena datora

Curl-Loader - (http://curl-loader.sourceforge.net) Atvērtā koda rīks rakstīts C valodā. Var simulēt vairāku tūkstošu HTTP/HTTPS un FTP/FTPS klientu slodzi. Katrs klients var būt ar savu IP adresi.

Trūkumi:

- izpētes laikā nebija pieejams lejupielādēšanai no interneta;
- nav grafiskā lietotāja saskarne;
- nav iespējams automātiski ierakstīt testa piemērus.

2.3. Secinājumi

Izpētes darba laikā kopsummā tika apskatīti vairāk kā divi desmiti bezmaksas slodzes testēšanas rīku. Liela daļa no rīkiem tika uzstādīta un palaista, lai varētu novērtēt piedāvāto

funkcionalitāti. Var secināt, ka ir pieejams pietiekoši plašs bezmaksas rīku klāsts un tai skaitā arī ar pietiekoši bagātu funkcionalitāti. Tika atlasīti seši rīki, kurus autors uzskata par labākajiem lielu sistēmu testēšanai. Tomēr galējo izvēli vēl nebija iespējams veikt, jo katrs no šiem rīkiem piedāvā lielu jaudu un dažādas funkcijas. Galējā izvēle tiek paredzēta pēc atlasīto rīku izmēģināšanas reāla projektā, ko paredzēts veikt tuvākajā laikā.

Izvēlētos rīkus raksturo sekojošās kopējās īpatnības:

- iespēja darbināt testus no vairākiem attālinātiem datoriem, tādējādi sasniedzot lielu testu slodzi;
- iespēja uzkrāt informāciju par dažādiem sistēmas resursiem (procesoru, atmiņas, utt.)
- iespēja pielāgot testus un ievācamo informāciju.

3. RĪKU IZMANTOŠANA REĀLĀ PROJEKTĀ

Autora darba vietā tiek izstrādāts projekts, kurš bija par motivāciju dotā darba izstrādei. Projektā izstrādājamai sistēmai paredzēts uzturēt lielu skaitu vienlaicīgu lietotāju, kas varētu sasniegt pat 10 000 un vairāk, tāpēc slodzes testēšana un mērogojamības plānošana ir nepieciešama.

3.1. Rīku izvēle

Projektā slodzes un mērogojamības testēšanā tika nolemts izmēģināt divus no iepriekš atlasītajiem bezmaksas rīkiem - The Grinder un CLIF.

3.1.1. CLIF

CLIF tika izvēlēts dēļ jau realizētām dažādas resursu zondēm (CPU, RAM, datu nesēju, tīkla, utt.), kas paralēli IIS veiktspējas novērtēšanai ļaus arī noteikt sistēmas vājās vietas. CLIF piedāvā arī plašas pielāgojamības iespējas, tai skaitā, sniedz iespējas veidot pašam savus slodzes ģeneratorus.

Kā negatīvās CLIF īpašības tika atzītas ierobežotā un specifiskā testa skripta valoda XML formātā. Tā nepiedāvā pārbaudīt servera reakcijas pareizību uz pieprasījumiem.

3.1.2. The Grinder

Tādēļ papildus tam tika izvēlēts arī The Grinder rīks. Tas ļauj testa skriptus rakstīt valoda Jython, kas ir Python valodas paplašinājums ar iespēju izmantot visus valodas Java objektus. Tāda pieeja ir ļoti ērta un saprotama Java programmētājiem, kas arī ļoti noderēja dotajā projektā.

Kā vēl viens faktors The Grinder izvēlei bija iepriekšēja pieredze šī rīka lietošanā.

3.1.3. Kāpēc ne citus rīkus

PushToTest TestMaker rīks netika izvēlēts, jo pēc dziļākas izpētes atklājās, ka par testēšanu ar vairāk kā 200 ģenerētiem lietotājiem tomēr ir jāmaksā.

Jmeter rīku autors atzina par grūti izmantojamu iesācējam. Dokumentācija vāji palīdzēja šajā jautājumā.

OpenSTA rīks netika izvēlēts, jo tas paredzēts darbināšanai no Windows platformām.

Faban rīks netika izvēlēts, jo izmanto XML valodu testu skriptu rakstīšanai, kas salīdzinoši

ar Jython ir daudz neērtāk un ar mazāk iespējām.

3.2. Projekta apraksts

Projektā tiek izstrādāta uz valodu Java bāzēta klienta-servera sistēma. Tā kalpo kā sociālā tīkla sistēma, kas nodrošina tās lietotājiem daudzveidīgu funkcionalitāti:

- informāciju par lietotājiem un tās kontaktiem, lietotāja kontaktu pārvaldību;
- ziņojumu, failu, mobilo tālruņu īsziņu sūtīšanu starp lietotājiem;
- lietotāju failu glabāšanu internetā;
- reālā laika tērzēšanas (chat) iespēju;
- lietotāju projektu, uzdevumu, laika pārvaldību;
- interešu grupas;
- dažādu multimediju publicēšanu, atskaņošanu;

Lietojumā tiek izmantotas sekojošās tehnoloģijas:

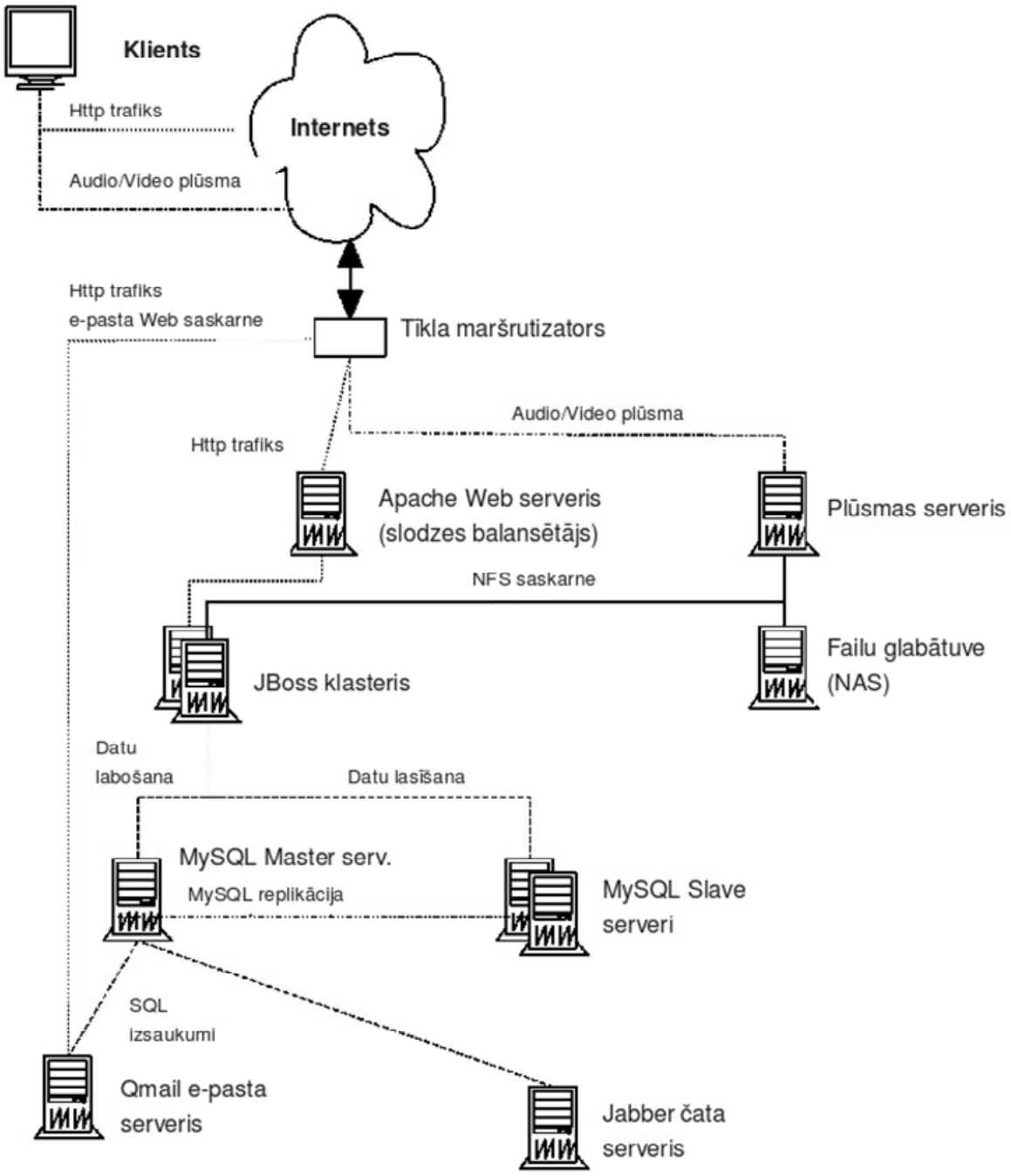
- IIS lietojuma pamata ietvars: Spring Framework 2.0. Tas ir dažādu Java tehnoloģiju un komponenšu kopums, ko var izmantot visdažādākajos dalītas biznesa sistēmas līmeņos, sākot ar datubāzes/biznesa līmeni un beidzot ar klienta līmeni. Izstrādātājs var lietot atsevišķas Spring Framework komponentes un atteikties no citām. Kalpo kā pamats dažādu sistēmas komponenšu integrēšanai.
- Lietojuma Web līmeņa ietvars: Apache Struts 1.2;
- Ajax tehnoloģiskā realizācija: DWR 2.0;
- Web servisu realizācijai tiek izmantots darba ietvars Codehaus XFire 1.2.
- Lietojuma un datubāzes objektu/relāciju sasaitei tiek izmantots darba ietvars; Hibernate 3.2. ar EJB3 anotācijām.
- Drošība: Acegi framework.
- Žurnalifikācija: Apache Commons – Logging;
- Junit vienīb-testēšanas ietvars;

- Maven projekta kompilācijas un uzbūves rīks.

Serveros izmantotā programmatūra:

- Datubāzes pārvaldības sistēma: MySQL 5.1;
- Java virtuālā mašīna (JVM): Sun JVM 1.5;
- Operētājsistēma: openSuse 10.2 Linux;
- Web serveris: Apache 2.2;
- Lietojuma serveris: Jboss 4.2.2.;
- Čata serveris: Ejabberd 1.3.1.;
- Datu plūsmas (streaming): Darwin Streaming Server (atvērtā pirmkoda versija Apple QuickTime Streaming Server tehnoloģijai);
- E-pasta serveris: Qmail;

IIS klienta-servera kopējā arhitektūra redzama sekojošajā attēlā (skat. att. 8.). Tā izmanto visdažādāko programmatūru, un to ir paredzēts darbināt uz liela skaita serveriem un serveru klasteriem, kas spētu nodrošināt pietiekošu darba jaudu.



att. 8. - projektā izstrādājamās sistēmas arhitektūra

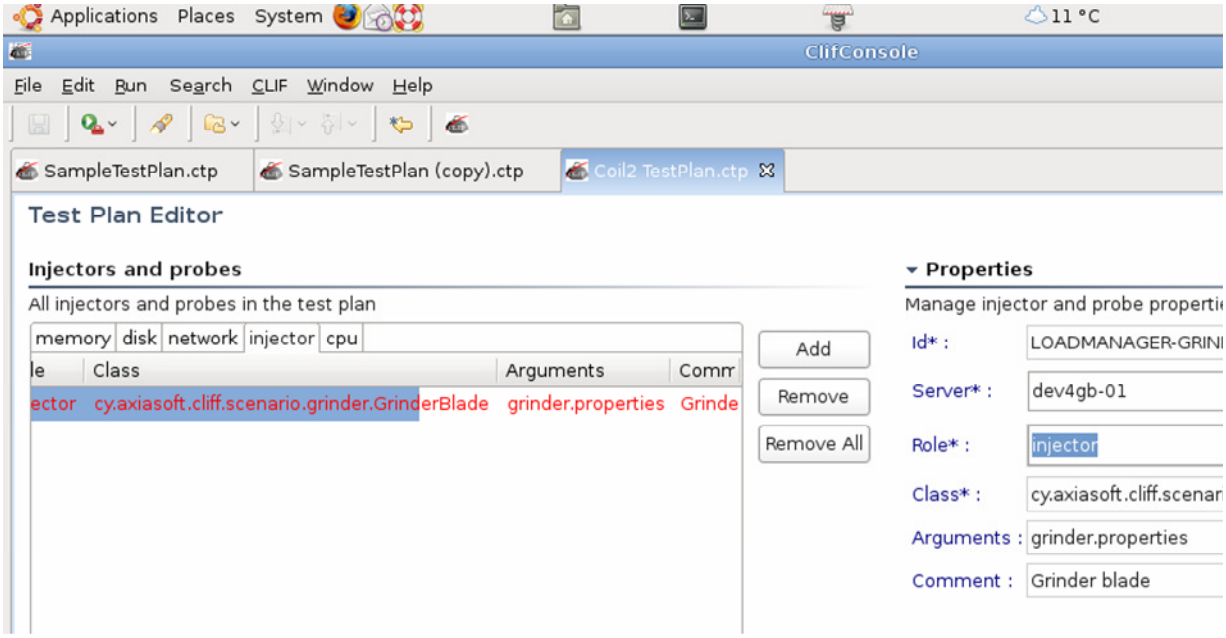
3.3. CLIF un The Grinder rīku lietojumu apvienošana

Rīks CLIF piedāvā iespēju izmantot pašam savus slodzes ģeneratorus. Pateicoties tam, ka abi rīki ir ar atvērto pirmkodu, ir iespējams izpētīt to uzbūvi un uzprogrammēt dažādas modifikācijas. Tika realizēta CLIF slodzes ģeneratora programmātiskā saskarne (tās pirmkods

dots 1. pielikumā), kura ļauj izmantot The Grinder rīku ar definētiem parametriem par slodzes ģeneratoru. Kā parametrs tiek padots Grinder testa parametrus saturoša faila nosaukums. Tas ļauj izmantot The Grinder iespējas Jython skriptu darbināšanā un slodzes definēšanā.

Rīku saskarnes galvenajās funkcijās ietilpst Grinder parametra faila un skripta faila pārsūtīšana uz slodzes ģeneratoriem un Grinder aģenta palaišana un apturēšana.

Grinder slodzes ģeneratora saskarne tiek definēta Clif testa plānā (skat. att. 9.).

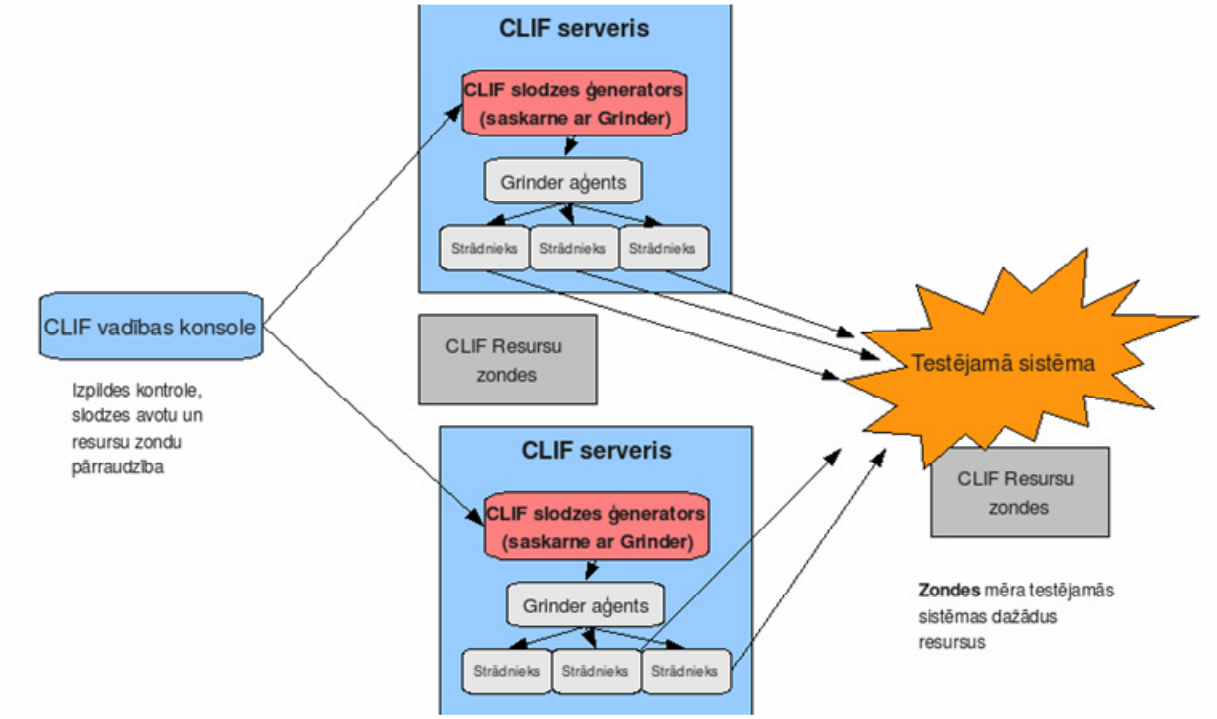


att. 9 – CLIF-Grinder saskarnes izmantošana CLIF testa plānā

CLIF-Grinder saskarnes darbības princips (skat. att. 10.):

- slodzes ģeneratora parametru uzstādīšana:
  1. CLIF serveris pēc padotā izpildes argumenta pieprasa un saņem grinder.properties failu no CLIF vadības konsoles;
  2. Atsūtītais fails tiek nolasīts un tiek iegūta informācija par Grinder skripta faila nosaukumu un atrašanās vietu;
  3. CLIF serveris pieprasa, saņem un saglabā Grinder skripta failu no vadības konsoles;
  4. grinder.properties failā tiek ierakstīta atsūtīta skripta faila saglabāšanas vieta.

- slodzes ģeneratora inicializācija. Grinder žurnalifikācijas objekta un Grinder aģenta izveidošana ar no vadības konsoles iegūtajiem parametriem;
- slodzes ģeneratora startēšana. Jauna darbības pavediena (thread) izveidošana un palaišana, kurā tiek darbināts iepriekš izveidotais Grinder aģents.
- slodzes ģeneratora apturēšana.



att. 10. – CLIF-Grinder apvienotās darbības arhitektūra

Neatrisinātās problēmas:

- Grinder aģentam nosūtīto failu atjaunošana. Problēma saistās ar failu kešošanu CLIF serverī. Kad tiek sūtīts jauns testa skripta fails ar tādu pašu nosaukumu, kāds jau eksistē uz slodzes ģeneratora, tas netiek uzkopēts pa virsu esošajam. Pašlaik problēma tiek risināta ar CLIF serveru restartu.
- Automātiska informācijas savākšana no Grinder dzinēja. Pašlaik Grider aģenta ģenerēto informāciju no slodzes ģeneratoru serveriem nākas kopēt manuāli, izmantojot Unix Shell skriptus. Mērķis ir to realizēt izmantojot CLIF konsoles „Collect" komandu;

3.4. Testu plānošana

Testēšanas mērķi

Testēšanas uzdevums ir noskaidrot:

- cik vienlaicīgus lietotājus sistēma spēj apkalpot;
- vai sistēma spēj strādāt ilgstošu laiku pie definētas slodzes;
- noskaidrot optimālu sistēmas konfigurāciju;
- noteikt sistēmas mērogojamības formulu, lai paredzētu nepieciešamo konfigurāciju dotam lietotāju skaitam.

Simulētā lietotāja veicamās darbības

- pieslēgšanās sistēmai un iziešana no tās;
- ziņojumu rakstīšana/saņemšana;
- attēlu un dokumentu augšupielādēšana;
- rakstu komentēšana;
- lietotāju profilu skatīšanās/labošana;
- statistikas aplūkošana, citas darbības.

Izvirzītās prasības sistēmai

- vidējais sistēmas reakcijas laiks testā – ne vairāk kā 7 sek;
- maksimālais reakcijas laiks – 10 sek;
- maksimālā pieprasījumu/kļūdainu reakciju attiecība – 0.01 %.

Sākotnējie testējamās sistēmas noslodzes dati

- lietotāju skaits: 300 līdz 1000;
- faili: 10 failu katram lietotājam;
- vēstules: 70 vēstules katram lietotājam;

- dienasgrāmatu ieraksti: 10 katram lietotājam;
- Grupu skaits: 500;
- Lietotāju skaits vienā grupā: 100;
- Kontaktu skaits katram lietotājam: 40.

3.5. Testu sagatavošana

Testa skripta sagatavošana

Ar rīku The Grinder tika ierakstītas sistēmas lietotāja darbības. Tā rezultātā tika iegūts Jython valodā rakstīts skripts. Skripts tika modificēts un pielāgots. Skriptā tika pievienotas servera reakciju verifikācijas kļūdainu reakciju identificēšanai. Skriptā realizēta vairāki dažādu lietotāju tipu profili (skat. pielikmu 2.).

Testa sistēmas datubāzes sagatavošana

Lai testu rezultāti vienmēr būtu adekvāti, vienmēr ir vajadzīga identiski dati. Tika uzprogrammēta programma "DB-populator" Java valodā, kas ļauj pēc definētiem dažādiem parametriem sagatavot testa datubāzi. Programma iztīra visu esošo datubāzi un ieraksta tajā par jaunu visus testam nepieciešamos datus, izveido sistēmā nepieciešamos lietotājus, vēstules, dienasgrāmatu ierakstus, grupas, kontaktus, un citus testa objektus.

Testa vides sagatavošana

Visos sistēmas serveros tiek uzstādīts CLIF rīks. Tiek izmantotas CLIF sistēmas RAM, CPU, tīkla un cieto datu nesēju resursu zondes. Katra no šī tipa zondēm tika uzstādīta katrā serverī un cieto datu nesēju zonde pa vienai uz katru datu nesēju.

Slodzes ģeneratoru serveros CLIF serverī tiek iekopēta CLIF-Grinder saskarnes bibliotēka un nepieciešamās Grinder bibliotēkas.

3.6. Testa darbināšana

Jboss un CLIF serveru apturēšanai un palaišanai, kā arī žurnalifikācijas informācijas ievākšanai un dzēšanai tiek izmantoti Unix Shell skripti, kas veic minētās darbības automātiski uz

visiem serveriem.

1. Apturam JBoss serverus;
2. Izdzēšam žurnalifikācijas informāciju no JBoss serveriem;
3. Apturam CLIF serverus;
4. Izdzēšam žurnalifikācijas informāciju no CLIF serveriem;
5. Atjaunojam datubāze ar programmu DB-populator;
6. Palaižam Jboss lietojumu serverus;
7. Palaižam CLIFF serverus;
8. No CLIFF konsoles palaižam testu;
9. Ievācam informāciju no Jboss serveriem;
10. Ievācam informāciju no CLIF serveriem;

Testu palaišanai tiek izmantota CLIF konsole. Ar tās palīdzību palaiž testus un resursu zondes visos serveros.

3.7. Rezultāti

Rezultātus pēc testu izpildes savāc žurnalificēto informāciju no CLIF slodzes ģeneratoru serveriem un no JBoss aplikāciju serveriem. Testu savākšanai tiek izmantoti Unix Shell skripti, kas automātiski savāc, saarhivē un nokopē informāciju uz vadības datoru.

3.4.1. Rezultātu piemērs

Vispārējie dažādu testu rezultāti

Vispārējos testa rezultātos tiek attēlota tikai testa parametri un testa vides konfigurācija, un vidējie ātrdarbības un kļūdu attiecības rezultāti (skat. tabulu 11.). Atskaites kolonu raksturojums:

- Date – testa izpildes datums;
- Test Run id – testa izpildes identifikators, kas darbojas arī kā saite uz detalizētu testa atskaiti;

- Test script version – Grinder testa skripta versija;
- Jboss configuration – Jboss lietojumu servera konfigurācijas piezīmes, izmantoto serveru skaits utt;
- MySql configuration – MySql dabutāzes konfigurācijas piezīmes, izmantoto serveru skaits utt. M – Master serveris, S – Slave serveris;
- Source configuration – izejas koda konfigurācija un izmaiņas, kas varētu ietekmēt sistēmas ātrdarbību;
- Avarage response time – vidējais servera reakcijas laiks;
- Tests/Errors ratio – testu/kļudu kopējais skaits;

Date	**Test Run id**	**Test script version**	**Test run configuration**	**JBoss configuration**	**MySql configuration**	**Source configuration**	**Average response time (ms)**	**Tests/Errors ratio**
2008.05.10 17:00	tp1.run.16	1.3	threads: 300 waitTime: 0	Same as tp1.run.15 database connection pool increased from 200 to 400	1M+1S, Same as tp1.run.15	Same as tp1.run.109 Person security settings optimized, connection closing strategy changed to after_transaction	11 204	132 601/0
2008.05.11 13:00	tp1.run.17	1.3	threads: 400 waitTime: 0	Same as tp1.run.16 database connection pool: 300	1M+1S, Same as tp1.run.15	Same as tp1.run.16	17 553	263 638/**157**

att 11. - Vispārējais testu apraksts un rezultāti

Kā redzams pēc kolonas "Tests/Errors ratio", vienlaicīgo lietotāju skaarits 400 jau rada kļūdas sistēmā, kā arī vidējais reakcijas laiks ir pieaudzis par aptuveni 50 %, salīdzinot ar 300 vienlaicīgu darbību.

Konkrēta testa detalizēta atskaites

Konkrētu testu detalizētās atskaitēs tiek attēlota sīkāka informācija par sistēmas funkcijas izpildes laikiem, kā arī grafikos attēlota informācija no dažādām CLIF resursu zondēm.

Ilgāko laiku aizņemošās klašu metodes (skat. att. 12.):

Hits per second: 26.5202

TOP 20

Most time consuming operations according to execution_times.log

Class Name	Method Name	Number of ocourances in log	Averag time logge (ms)
cy.axiasoft.coil.server.webspace.dao.hibernate.UserEducStepsDaoHibernate	getUserEducSteps	1	122 345
cy.axiasoft.coil.server.webspace.service.impl.EducStepsManagerImpl	getSortedActiveUserEducSteps	5	95 950
cy.axiasoft.coil.server.webspace.dao.hibernate.UserEducStepsDaoHibernate	getEducStepsEntries	4	89 096
cy.axiasoft.coil.server.webspace.service.impl.UserManagerImpl	registerUserLoginEvent	13	72 866
cy.axiasoft.coil.server.webspace.dao.hibernate.UserLoginDaoHibernate	saveObject	13	72 857
cy.axiasoft.coil.server.webspace.dao.hibernate.ProfileDaoHibernate	getProfileByPersonAndType	3	52 164
cy.axiasoft.coil.server.webspace.dao.hibernate.GalleryDaoHibernate	saveObject	21	50 586
cy.axiasoft.coil.server.dao.hibernate.AssociatedFileDaoHibernate	getUsedBytes	10	45 876
cy.axiasoft.coil.server.webspace.service.impl.PhotoManagerImpl	getUsedSpaceInMb	15	45 850
cy.axiasoft.coil.server.webspace.dao.hibernate.GalleryDaoHibernate	getGalleriesForPersonCount	2	45 232
cy.axiasoft.coil.server.service.AssociatedFilesServiceImpl$$EnhancerByCGLIB$$5423a6e9	getLimits	22	37 003

att. 12 - ilgāko laiku aizņemošās klašu metodes

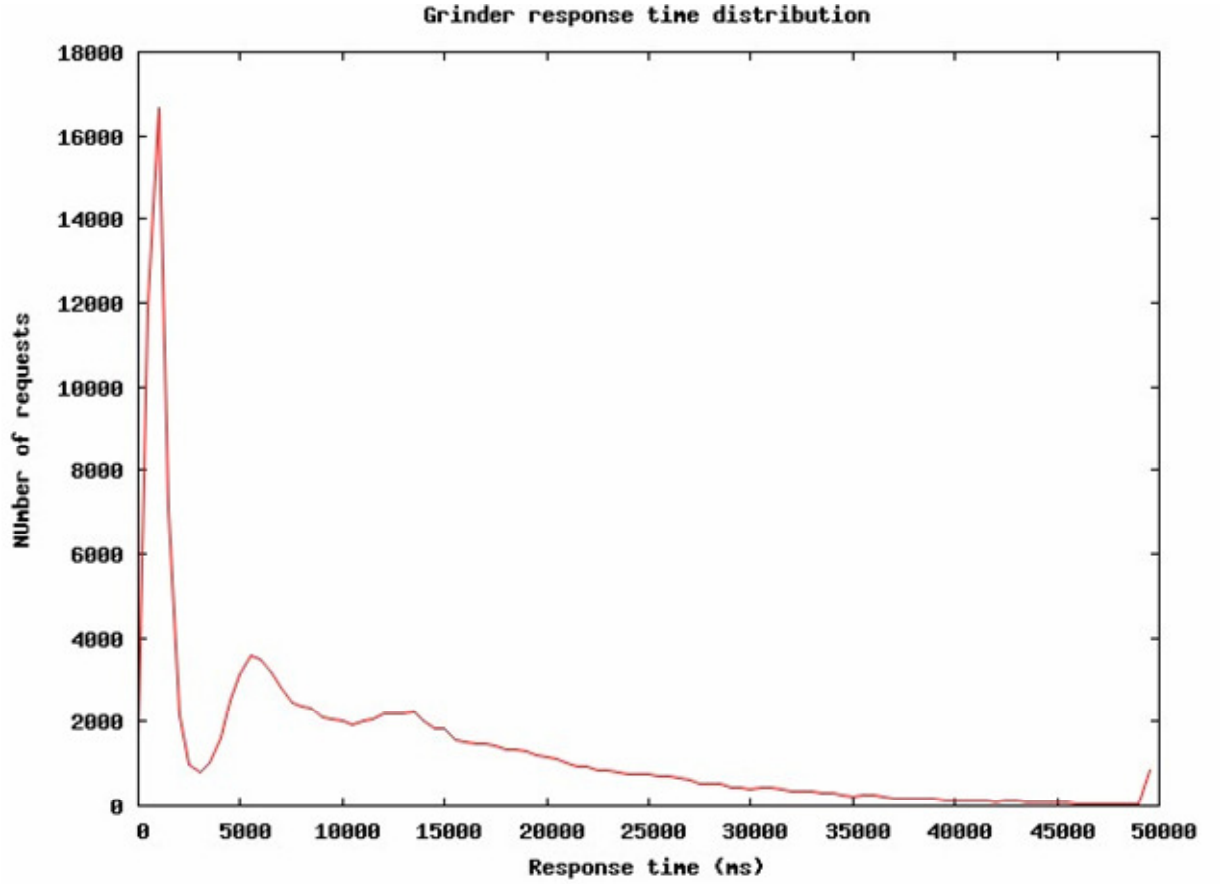
Visbiežāk izpildītās metodes (skat. att. 13.):

Operations that occurs most often in execution_times.log

Class Name	Method Name	Number of ocourances in log	Averag time logge (ms)
cy.axiasoft.coil.server.webspace.service.impl.MessageManagerImpl	getNewMessageDirectContactCount	10 642	7 776
cy.axiasoft.coil.server.webspace.dao.hibernate.ReceivedMessageDaoHibernate	getNewMessageDirectContactCount	8 905	7 652
cy.axiasoft.coil.server.webspace.service.impl.PersonSecurityManagerImpl	setSecuritySettingsByPersonId	8 550	10 286
cy.axiasoft.coil.server.webspace.dao.hibernate.ContactDaoHibernate	getContactBetweenPersons	7 101	7 753
cy.axiasoft.coil.server.webspace.service.impl.MessageManagerImpl	getNewMessageNotFromDirectContactCount	6 801	7 764
cy.axiasoft.coil.server.webspace.service.impl.PersonManagerImpl	getOnlineListWithChat	6 554	8 236
cy.axiasoft.coil.server.webspace.service.impl.MessageManagerImpl	getNewMessageDirectContact	5 733	7 644
cy.axiasoft.coil.server.webspace.dao.hibernate.ReceivedMessageDaoHibernate	getNewMessageDirectContact	5 529	7 616
cy.axiasoft.coil.server.webspace.service.impl.PersonSecurityManagerImpl	setSecuritySettingsForGroupUsers	5 249	10 246
cy.axiasoft.coil.server.webspace.dao.hibernate.ReceivedMessageDaoHibernate	getNewMessageNotFromDirectContactCount	5 234	7 616
cy.axiasoft.coil.server.webspace.dao.hibernate.PersonDaoHibernate	getOnlineList	5 011	8 025
cy.axiasoft.coil.server.webspace.dao.hibernate.UserSettingsDaoHibernate	getUserSettingsForPersons	5 011	8 834
cy.axiasoft.coil.server.webspace.service.impl.MessageManagerImpl	getNewMessageNotFromDirectContact	4 917	7 544
cy.axiasoft.coil.server.webspace.dao.hibernate.ReceivedMessageDaoHibernate	getNewMessageNotFromDirectContact	4 632	7 483

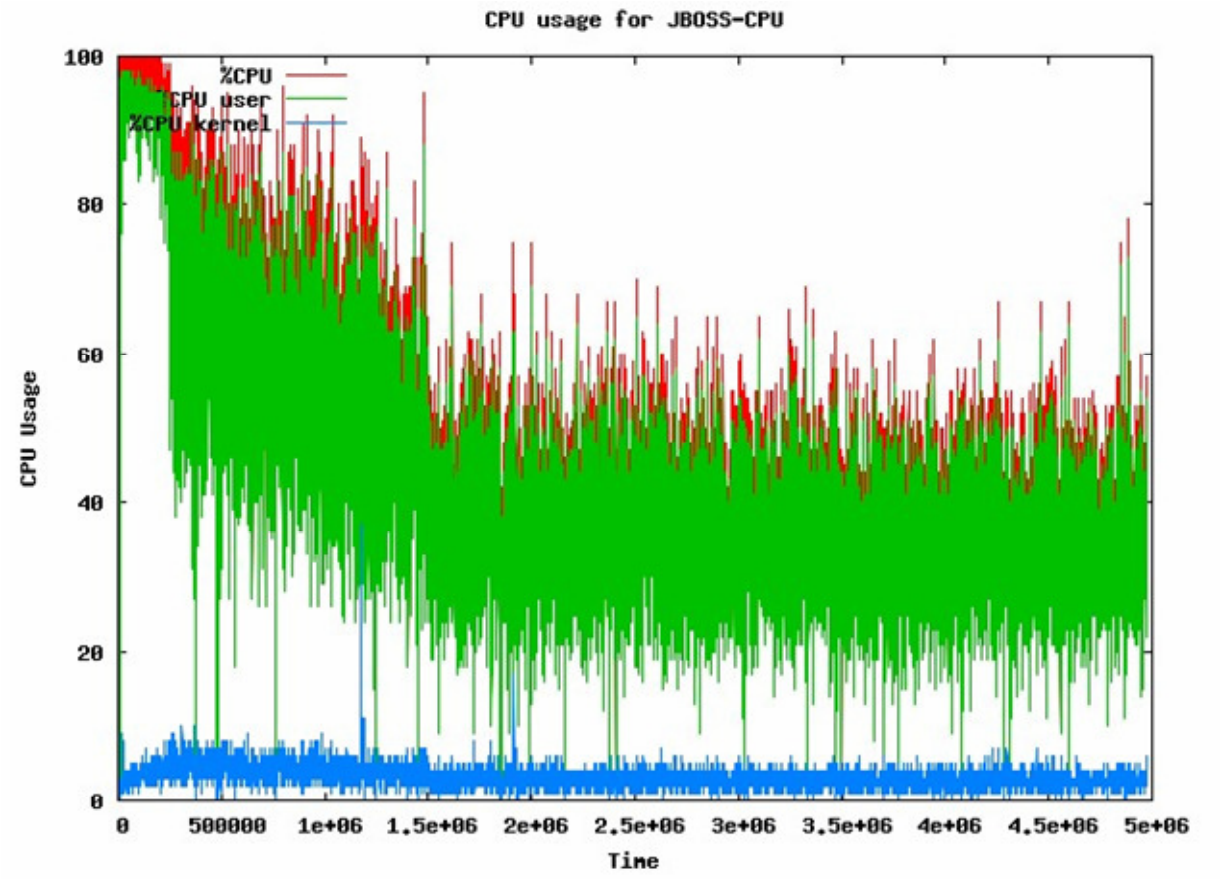
att. 13 - visbiežāk izpildītās metodes

Dažādo reakcijas laiku attiecība (skat. att. 14.):

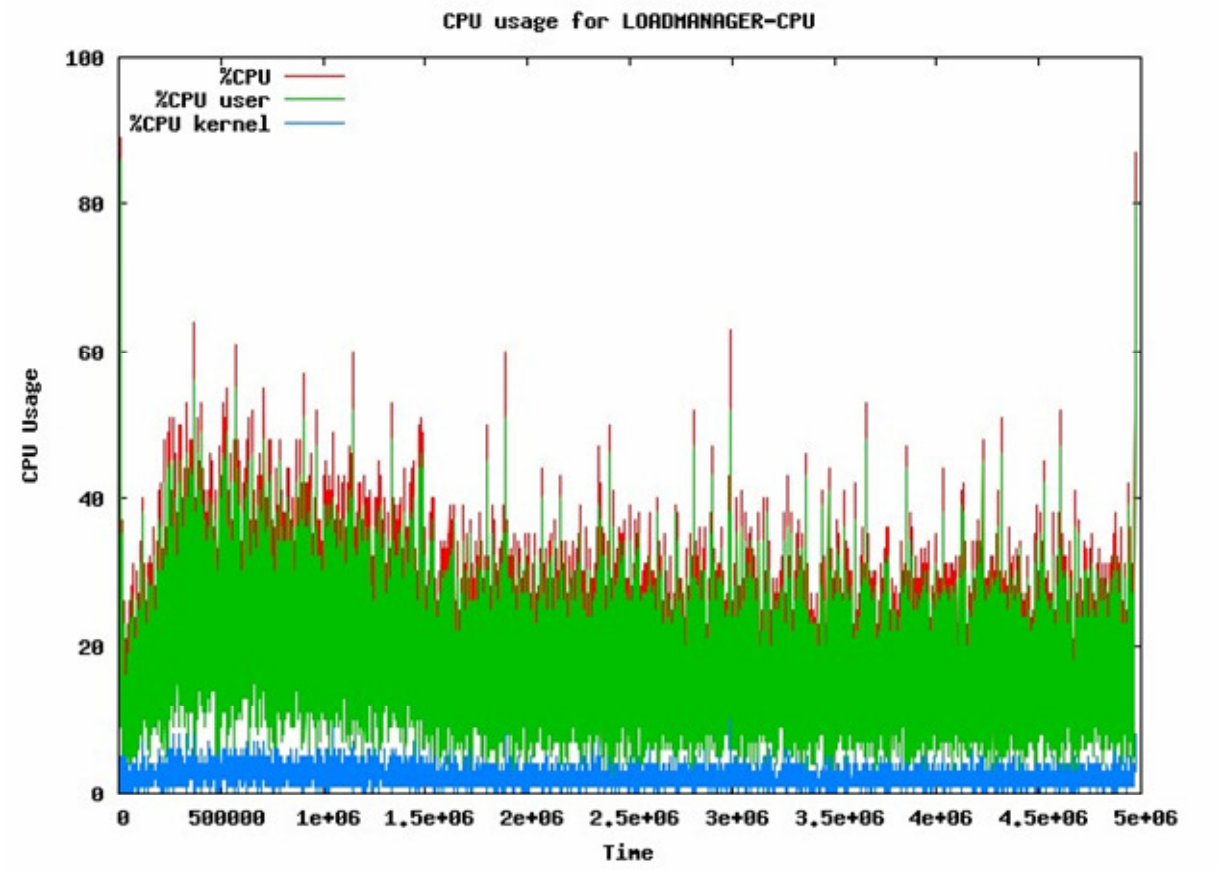


att. 14. - dažādo reakcijas laiku attiecība

Un citi grafi (par piemēru, skat. att. 15. un 16.):



att. 15. – CPU noslodze lietojuma serverī



att. 16. – CPU noslodze slodzes ģeneratora datorā

3.8. Mērogojamības testēšana

Mērogojamības testēšana tiek veikta ar mērķi prognozēt nepieciešamo konfigurāciju un resursus dažādām slodzēm. Noslodzes galvenā mērvienība tiek uzskatīta vienlaicīgi ar IIS strādājošo lietotāju skaits.

Dažādie konfigurāciju varianti sastāv no sekojošajām variācijām:

- datubāzes serveru skaits;
- lietojumu serveru skaits;
- Web serveru skaits;
- slodzes ģeneratoru skaits;
- datubāzes servera aparatūras konfigurācija;
- lietojuma servera aparatūras konfigurācija;

Mērogojamības testēšanas gaita un rezultāti šajā darbā nav apskatīti, jo tam paredzētā sistēmas konfigurācija tam vēl tiek gatavota. Tiek testēta sistēma, kas maksimālajā konfigurācijā sastāv no 16 serveriem.

NOSLĒGUMS

Motivācija šim darbam un šī darba paredzamajam turpinājumam autoram radās pie nepieciešamības iemācīties un atrast līdzekļus notestēt veiktspēju liela mēroga interneta sistēmai (vairāk kā 10 000 paredzamo vienlaicīgo lietotāju). Autora profesionālās darbības sfēra ir saistīta ar bezmaksas un atvērtā koda programmatūras risinājumiem, tāpēc arī testēšanas līdzekļus tika nolemts meklēt šāda veida risinājumos. Veicot pētījumus atklājās, ka šādiem mērķiem ir pieejami vairāki nopietni bezmaksas atvērtā koda rīki.

Šajā darbā autors iepazinās ar slodzes un mērogojamības testēšanas metodēm un īpatnībām, kas saistītas ar interneta informācijas sistēmām, kā arī apskatīja pieejamos bezmaksas slodzes testēšanas rīkus. Liela daļa no rīkiem tika uzstādīta un palaista, lai varētu novērtēt piedāvāto funkcionalitāti.

Autoram izdevās novērtēt un atlasīt sešus rīkus, kuri nodrošina pietiekošu kvalitāti un jaudu lielu sistēmu testēšanai. Rīku lielā veiktspēja tiek sasniegta balstoties uz dalīto arhitektūru, kas ļauj noslogot testējamo sistēmu no vairākām darbstacijām vienlaicīgi.

Autors izvēlējās divus no iepriekš atlasītajiem rīkiem, kurus izmēģināt un ieviest reālā projektā. Pirmais rīks ir CLIF, kas ir rīks ar plašām realizētām iespējām, gan ar labām modifikācijas īpašībām. Otrais rīks ir The Grinder, kas nodrošina ērtu skriptu valodu, kas balstās uz Jython un ir arī viegli konfigurējams. Autors veiksmīgi izmēģināja šos rīkus projektā un realizēja iespēju šos rīkus izmantot kombinētā veidā, tādējādi izmantojot abu rīku labākās īpašības. Tika gatavota lielas slodzes testēšana ar detalizētas informācijas ievākšanu, kurā tiks izmantoti 16 serveri vienlaicīgi.

Rezultātā tika atrasti bezmaksas rīki, kas kombinēti darbojoties dotu plašu informāciju par sistēmas veiktspēju un tās dažādu resursu noslodzi, un tai pašā laikā testu veidošana un uzturēšana būtu pietiekoši ērta un funkcionāla. Šiem rīkiem tika atrasta iespēja veiksmīgi kombinēti darboties, un tas tika ieviests un izmēģināts reālā projektā.

Turpmāk paredzēts veikt testus uz vēl daudzkārt jaudīgākām sistēmām.

LIETOTIE TERMINI UN SAĪSINĀJUMI

BSD licence – bezmaksas programmatūras licenču saime;

CORBA – Common Object Request Broker Architecture; Objektu pieprasījumu vispārēja arhitektūra, atļauj programmām vai to daļām sadarboties attālināti caur tīklu. Līdzīgas iespējas piedāvā arhitektūras COM/DCOM no Microsoft un RMI no Sun Microsystems;

Distributīva sistēma – dalītas arhitektūras sistēma, kas var sastāvēt no vairākiem savā starpā savienotiem datoriem;

Darbības pavediens – (*thread*) sistēmas darbības izpildes paralēls process.

HTTP – Hiper Text Transfer Protocol. Teksta informācijas pārraides protokols starp serveri un pārlūka programmu.

HTTPS – Hypertext Transfer Protocol over Secure Socket Layer. HTTP protokols ar SSL izmantošanu drošākai datu pārraidei.

Lietojumu serveris – programmatūras dzinējs, kas nodrošina programmas darbināšanu un tās komunikāciju ar klientiem.

SSL - Secure Socket Layer. Kriptogrāfisks protokols, kas nodrošina drošu datu pārraidi internetā.

Klasteris – vairāku datoru apvienojums, kuri strādā vienotos uzdevumos un ko kopumā daudzējādā ziņā var uzskatīt kā vienu sistēmu. Parasti klasterus veido jaudas un stabilitātes palielināšanai.

Serveris – datorsistēma vai programma, kas apkalpo citus datorus vai programmas.

Servlets – Java klase, kas apstrādā Http pieprasījumus.

SNMP - Simple Network Management Protocol. To izmanto tīkla pārvaldības sistēmās, lai novērotu tīklam pieslēgtas ierīces.

SOAP – protokols XML-bāzētu ziņojumu apmaiņai datoru tīklos. Nav atkarīgs no izmantotās operētājsistēmas vai programmēšanas valodas.

TCP/IP – interneta protokolu kopa, kas sastāv no TCP (Transmission Control Protocol) un

IP (Internet Protocol), kuri ir pirmie divi definētie tīkla protokoli.

Web – interneta tīkls;

Web serveris – programma, kas ir atbildīga par HTTP pieprasījumiem no klientiem jeb interneta pārlūka programmām un HTTP atbilžu nosūtīšanu.

XML – paplašināmās iezīmēšanas valoda. Programmēšanas valodas SGML apakškopa, kas speciāli izstrādāta darbam ar tīmekļa dokumentiem. [7]

LITERATŪRAS SARAKSTS

[1] Hung Q. Nguyen, *Testing Applications on the Web*, John Wiley & Sons, 2001, p. 402;

[2] „The Web Testing Companion – The Insider's Guide to Efficient and Effective Tests." Lydia Ash. 2003.

[3] Wikipedia: Load testing [tiešsaiste]. – [atsauce 22.03.2008.] Pieejams: http://en.wikipedia.org/wiki/Load_testing;

[4] Wikipedia: Scalability [tiešsaiste]. – [atsauce 22.03.2008.] Pieejams: http://en.wikipedia.org/wiki/Scalability;

[5] André B. Bondi, „Characteristics of scalability and their impact on performance", *Proc. 2nd int'l workshop Software and performance,* Ottawa, Ontario, Canada, 2000, pp. 195 – 203;

[6] Leticia Duboc, David S. Rosenblum, Tony Wicks, „A framework for modelling and analysis of software systems scalability" in *Proc. 28th int'l conf. Software eng. ICSE '06*, 2006, pp. 949 – 952;

[7] P. Jogalekar and M. Woodside, „Evaluating the scalability of distributed systems", IEEE Trans. Parallel and Distributed Systems, pp. 589-603, 2000;

[8] D. N. Kolisničenko, „Angļu – krievu skaidrojošā datoru terminu vārdnīca", 2. izdevums, 2008

PIELIKUMI

CLIF un The Grinder rīku saskarnes pirmkods

```
package cy.axiasoft.cliff.scenario.grinder;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import net.grinder.common.GrinderException;
import net.grinder.common.Logger;
import net.grinder.engine.agent.Agent;
import net.grinder.util.SimpleLogger;

import org.objectweb.clif.datacollector.lib.GenericFilter;
import org.objectweb.clif.scenario.isac.util.BooleanHolder;
import org.objectweb.clif.server.api.BladeControl;
import org.objectweb.clif.server.util.EventStorageState;
import org.objectweb.clif.storage.api.ActionEvent;
import org.objectweb.clif.storage.api.AlarmEvent;
import org.objectweb.clif.supervisor.api.ClifException;
import org.objectweb.clif.util.ClifClassLoader;

public class GrinderBlade extends GrinderController implements BladeControl {

  private enum TestState {                                          // testa stāvokļi
        uninitialized, initialized, started, suspended
  }

  class ExecutionThread extends Thread {                    // Grinder aģenta darbināšanas
pavediens
        @Override
        public void run() {
              System.out.println("Starting Grinder");
              try {
                    agent.run();                                                  // Grinder
aģenta palaišana
                    System.out.println("Grinder stopped, shutting down");
                    // statistics?

                    agent.shutdown();                                             // Grinder
aģenta apturēšana
                    System.out.println("Done !!!");

                    done();                                                       //
procesa beigšanas signāla padošana CLIF vadības konsolei
                    System.out.println("Exiting");
```

```
                        } catch (GrinderException e) {
                                e.printStackTrace();
                                AlarmEvent alarm = new AlarmEvent(System.currentTimeMillis(),
AlarmEvent.FATAL, e.getMessage());
                                bladeInsertResponse.alarm(alarm);
                                bladeInsertResponse.aborted();
                        }
                }
        }

        /* test scenario id */
        private String id;
        /* Event storage variables */
        private final Map<String, BooleanHolder> eventStorageStatesMap = new HashMap<String,
BooleanHolder>();
        private final BooleanHolder storeLifeCycleEvents = new BooleanHolder(true);
        private final BooleanHolder storeAlarmEvents = new BooleanHolder(true);
        private final BooleanHolder storeActionEvents = new BooleanHolder(true);

        /* test state */
        private TestState testState;

        /* Grinder Agent */
        private Agent agent;        // Grinder testu darbināšanas aģents

        /* Temporary files used by Grinder */
        private File scriptFile;                    // Grinder skripta fails
        private File propertiesFile;                // grinder.properties fails

        public GrinderBlade() {                             // Grinder saskarnes instruktors
                eventStorageStatesMap.put("store-lifecycle-events", storeLifeCycleEvents);
                eventStorageStatesMap.put("store-alarm-events", storeAlarmEvents);
                eventStorageStatesMap.put("store-" + ActionEvent.EVENT_TYPE_LABEL + "-events",
storeActionEvents);
                testState = TestState.uninitialized;
        }

        public void changeParameter(String parameter, Serializable value) throws ClifException {
                if (EventStorageState.setEventStorageState(eventStorageStatesMap, parameter, value)) {
                        dataCollectorWrite.setFilter(new GenericFilter(storeActionEvents.getBooleanValue(),
storeAlarmEvents.getBooleanValue(),
                                        storeLifeCycleEvents.getBooleanValue(), false));
                }
        }

        @SuppressWarnings("unchecked")
        public Map getCurrentParameters() {
                Map parameters = new HashMap();
                EventStorageState.putEventStorageStates(parameters, eventStorageStatesMap);
                return parameters;
        }

        public String getId() {
                return id;
        }

        /**
         * Method for parsing blade arguments.<br>
```

```
	 * Grinder blade expects to receive grinder properties file name
	 */
	public void setArgument(String arg) throws ClifException {  // CLIF slodzes ģeneratora argumentu izmantošanas funkcija
		if (arg == null)
			throw new ClifException("Grinder properties file name should be passed as blade's argument");
		ClifClassLoader ccl = ClifClassLoader.getClassLoader(); // tiek iegūts CLIF ielādētājs
		InputStream propsIn = ccl.getResourceAsStream(arg);		// tiek atsūtīts Grinder properties faila saturs no CLIF vadības konsoles
		if (propsIn == null)
			throw new ClifException("Could not find file named '" + arg + "' on the classpath!");
		Properties grinderProps = new Properties();
		try {
			grinderProps.load(propsIn);                                                                    // nolasa parametrus no faila
		} catch (IOException e) {
			e.printStackTrace();
			throw new ClifException(e);
		}
		String scriptFileName = grinderProps.getProperty("grinder.script");	// ieguust skripta faila nosaukumu
		InputStream scriptIn = ccl.getResourceAsStream(scriptFileName);		// tiek atsūtīts skripta fails no vadības konsoles
		scriptFile = writeScriptToTmpFile(scriptIn);                                                   // skripta fails tiek saglabāts uz slodzes ģeneratora datora
		grinderProps.setProperty("grinder.script", scriptFile.getAbsolutePath()); // Grinder parametors tiek uzstādīta atsūtītā skripta faila atrašanāš vieta
		propertiesFile = writePropertiesToTmpFile(grinderProps);		// tiek saglabāti labotie Grinder parametri
	}

	public void setId(String id) {
		this.id = id;
	}

	public void init(Serializable testId) throws ClifException {	// CLIF slodzes ģeneratora inicializēšanas funkcija
		System.out.println("**** Init: "+testId);
		try {                                                                                          // tiek izveidots Grinder loggers
			final Logger logger = new SimpleLogger("agent", new PrintWriter(
					System.out), new PrintWriter(System.err));
			agent = new Agent(logger, propertiesFile);  // tiek izveidots jauns Grinder aģents
			testState = TestState.initialized;			// tests tiek uzskatīts par inicializētu
		} catch (GrinderException ge) {
			ge.printStackTrace();
			throw new ClifException(ge);
		}
	}

	public void join() {
		// TODO Auto-generated method stub
	}

	public void resume() {
		if (testState != TestState.suspended) {
			// maybe throw exception?
```

```
                return;
        }
    }

    public void start() {                                   // CLIF slodzes ģeneratora startēšanas funkcija
        System.out.println("**** Start ");
        if (testState != TestState.initialized) {
            // maybe throw exception?
            System.out.println("Incorect state: "+testState);
            return;
        }
        Thread runner = new ExecutionThread();    // tiek izveidots jauns Grinder aģenta darbināšanas pavediens
        runner.start();                                     // pavediens tiek palaists
        testState = TestState.started;                      // tests tiek uzskatīts par palaistu
    }

    public void stop() {
        if (testState != TestState.started) {
            return;
        }
        // implement methods for stopping test
        testState = TestState.uninitialized;
    }

    public void suspend() {
        if (testState != TestState.started) {
            return;
        }
        // implement suspend
        testState = TestState.suspended;
    }

    public void done() {                                    // CLIF konsolei tiek padots signāls ka Grinder aģenta process ir beidzis darboties
        bladeInsertResponse.completed();
    }

    private File writeScriptToTmpFile(InputStream scriptIn)
            throws ClifException {
        try {
            File file = File.createTempFile("grinder", "py");
            FileOutputStream fos = new FileOutputStream(file);
            byte[] buffer = new byte[1024];
            int bytesRead = scriptIn.read(buffer);
            while (bytesRead != -1) {
                fos.write(buffer, 0, bytesRead);
                bytesRead = scriptIn.read(buffer);
            }
            fos.flush();
            fos.close();
            scriptIn.close();
            return file;
        } catch (IOException e) {
            e.printStackTrace();
            throw new ClifException(e);
        }
```

```
    }

    private File writePropertiesToTmpFile(Properties grinderProps) throws ClifException {
            try {
                    File file = File.createTempFile("grinder", "properties");
                    PrintWriter writer = new PrintWriter(file);
                    grinderProps.list(writer);
                    writer.flush();
                    writer.close();
                    return file;
            } catch (IOException e) {
                    e.printStackTrace();
                    throw new ClifException(e);
            }
    }

}
```

2. Pielikums

Grinder rīka Jython valodā rakstītais slodzes testu skripts

```
from net.grinder.script.Grinder import grinder
from net.grinder.script import Test
from net.grinder.plugin.http import HTTPRequest
from HTTPClient import NVPair, Codecs, Cookie, CookieModule, CookiePolicyHandler
from java.util.regex import Pattern
from java.util import Random;
from jarray import zeros

WEBSPACE_URL = "http://localhost:8080/webspace/"
LOAD_IMAGES_AND_JS_FILES = 1 # 0 = false; 1 = true
DEBUG = 1 # 0 = false; 1 = true
WAIT_TIME = 0
LOAD_PROFILE_ID = 1; # 0 - Normal, 1 - Business, 2 - Multimedia, 3 - Test

#Messaging, Image/Video upload/dwnload, File upload/dwnload, Blog reading, Blog writing, Profiles/Contacts, Tops/Stats, Groups
PROFILE_CONFIG = (
            (20,  5,  5, 30,  5, 15, 10, 10), # Normal user profile
            (35,  0, 20,  5,  0, 40,  0,  0), # Business user profile
            (10, 20,  0, 35, 10,  5, 10, 10), # Multimedia user profile
           ( 0,  0,  0,  0,  0,  1,  0,  0)  # Test profile
            )

DEFAULT_USERNAME = "user"
DEFAULT_PASSWORD = "test"

exceptionPattern = Pattern.compile("Exception")
errorPattern = Pattern.compile("Error")
sourcePattern = Pattern.compile("src=\"([^\"]*)\"")
groupIDPattern = Pattern.compile(">group(\d*)<")
personIDPattern = Pattern.compile("FirstName([\d]*) LastName")
dwrRemoteCallbackPattern = Pattern.compile("dwr\.engine\._remoteHandleCallback");

rand = Random()

DWR_SCRIPT_SESSION_ID = "92AEB577581C03CE06C638A1BADEE0E6"
# ---------------- Profile handling ----------------
def getProfile():
    profile = []
    profileConfig = PROFILE_CONFIG[LOAD_PROFILE_ID]
    counter = 0;
    for actionCount in profileConfig:
        for index in range(actionCount):
            profile.append(counter)
        counter = counter + 1
    return profile

def pickupAction(profile):
```

```
    if len(profile) < 1:
        return -1;
    return profile.pop(rand.nextInt(len(profile)))
# ---------------- Utility methods ----------------
def log(mesasge):
  if (DEBUG):
            grinder.logger.output(mesasge);

class WebspaceCookiePolicyHandler(CookiePolicyHandler):
    SESSION_COOKIE_NAME = "JSESSIONID"
    sessionId = "none"

    def acceptCookie(self, cookie, request, response):
        if cookie.name == self.SESSION_COOKIE_NAME:
            self.sessionId = cookie.value
            log("Session id : %s" % self.sessionId)
        return 1

    def sendCookie(self, cookie, request):
        log("send cookie: %s" % cookie)
        return 1

#simple get request for text
def doGetRequest(request, url):
    log("doing request to %s -> %s"%(request.url, url))
    result = request.GET(url)
    assert result.statusCode == 200
    return result.text

#simple get request binary data
def getResource(request, url):
    if url is None or url == "null" or url == "about:blank":
        log("Url is null")
        return
    log("doing request to %s -> %s"%(request.url, url))
    result = request.GET(url)
    assert result.statusCode == 200
    return result.getHeader("Content-type")

#simple post request
def doPostRequest(request, url, postArguments):
    log("doing request to %s -> %s"%(request.url, url))
    log("======================== postArguments :")
    for pair in postArguments:
        log("Argument: %s" % (pair))
    result = request.POST(url, postArguments)
    assert result.statusCode == 200
    return result.text

#multipart upload request
def doUploadFile(request, url, fileParamName, fileName, postArguments):
    log("uploading file %s to %s -> %s"%(fileName, request.url, url))

    filesToUpload = ( NVPair(fileParamName, fileName), )
    headers = zeros(1, NVPair)

    data = Codecs.mpFormDataEncode(postArguments, filesToUpload, headers)
    log("======================== Headers :")
```

```
            for pair in headers:
               log("Header: %s" % (pair))
            log("======================== postArguments :")
            for pair in postArguments:
               log("Argument: %s" % (pair))
            #log("======================== Data :")
            #log("Data: %s" % (data))

            result = request.POST(url, data, headers)
            assert result.statusCode == 200
            return result.text

         #simple dwr request
         def doDWRRequest(request, cookies, className, methodName, extraArgs):
            log("loading %s - %s " % (className, methodName))
            hardcodedPart = (NVPair("Content-type", "text/plain"),
                     NVPair("callCount","1"),
                     NVPair("scriptSessionId", DWR_SCRIPT_SESSION_ID),
                     NVPair("httpSessionId", cookies.sessionId),
                     NVPair("c0-scriptName", className),
                     NVPair("c0-methodName", methodName),
                     NVPair("c0-id", "0"),
                     NVPair("c0-param0", "boolean:false"),
                     NVPair("batchId", "0"),
                     )

            postArgs = []
            for element in hardcodedPart:
               postArgs.append(element)
            for element in extraArgs:
               postArgs.append(element)

            result = doPostRequest(request, "dwr/call/plaincall/%s.%s.dwr"%(className,methodName),
tuple(postArgs))
            assert not exceptionPattern.matcher(result).find()
            assert not errorPattern.matcher(result).find()
            assert dwrRemoteCallbackPattern.matcher(result).find()
            return result;

         #gets username/password from somewhere
         def getCredentials():
            userName = "%s%d"%(DEFAULT_USERNAME,grinder.threadID + 1)
            log("using user name %s"%userName)
            return  NVPair(userName, DEFAULT_PASSWORD)

         #Parses html text and load all images and other files from it.
         def loadResources(request, result):
            if not LOAD_IMAGES_AND_JS_FILES:
               return
            matcher = sourcePattern.matcher(result)
            while matcher.find():
               src = matcher.group(1)
               log("loading source %s"%src)
               getResource(request, src)

         #validates respone against list of regular expressions
         def validateResponse(request, result, patterns):
            log(result)
```

```
            for pattern in patterns:
               log('validating against pattern: "%s"'%pattern.pattern())
               assert pattern.matcher(result).find()
            loadResources(request, result)


         # loads left we-r-you pane - new messages, new direxT contact messages, online list
         WHO_IS_ONLINE_PATTERN = Pattern.compile("who is online from my friends")
         def loadLeftSide(request, cookies):
            result =  doDWRRequest(request, cookies, "Messages", "getNewMessageDirectContact", (NVPair("c0-e1",
"number:0"),  NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3}")))
            validateResponse(request, result, ())

            result =  doDWRRequest(request, cookies, "Messages", "getNewMessage", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3}")))
            validateResponse(request, result, ())

            result  =  doDWRRequest(request,  cookies,  "Contacts",  "getOnlineContacts",  (NVPair("c0-e1",
"number:0"),  NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3}")))
            validateResponse(request, result, (WHO_IS_ONLINE_PATTERN,))

         #loads right we-r-you pane in main page
         RANDOM_PHOTOS_PATTERN = Pattern.compile("Photos")
         TOP_STUFF_PATTERN = Pattern.compile("top stuff in global we-r-you")
         LAST_RECEIVED_FILES_PATTERN = Pattern.compile("Last received")
         LAST_UPLOADED_FILES_PATTERN = Pattern.compile("Last uploaded")
         def loadRightSide(request, cookies):
            result =  doDWRRequest(request, cookies, "Tops", "getRandomPhotos", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-e4",  "boolean:true"),  NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3, mainPage:reference:c0-e4}")))
            validateResponse(request, result, (RANDOM_PHOTOS_PATTERN,))

            result  =  doDWRRequest(request,  cookies,  "SparcFiles",  "getFilesForMainpage",  (NVPair("c0-e1",
"number:0"),  NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-e4",  "number:7"),
NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1,  currentPage:reference:c0-e2,
pageToNavigate:reference:c0-e3, filterMode:reference:c0-e4}")))
            validateResponse(request,  result,  (LAST_RECEIVED_FILES_PATTERN,
LAST_UPLOADED_FILES_PATTERN))

            #result  =  doDWRRequest(request,  cookies,  "Tops",  "getTopStuff",  (NVPair("c0-e1",  "number:0"),
NVPair("c0-e2",  "number:0"),  NVPair("c0-e3",  "number:1"),  NVPair("c0-e4",  "boolean:true"),  NVPair("c0-e5",
"boolean:false"),  NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1,  currentPage:reference:c0-e2,
pageToNavigate:reference:c0-e3, findAll:reference:c0-e4, inMyNetwork:reference:c0-e5}")))
            #validateResponse(request, result, (TOP_STUFF_PATTERN,))

         #                                                Test                                                methods
=============================================================================================
================
         # loginPage ----------------------------------------------------------------------------------------------
         LOGIN_FORM_PATTERN = Pattern.compile("loginForm")
         def loginPage(request):
```

```
        result = doGetRequest(request, "signup.do");
        validateResponse(request, result, (LOGIN_FORM_PATTERN,))

    # doLogin ---------------------------------------------------------------------------------------------------
    TOP_MENU_PATTERN = Pattern.compile("TOP_MENU_ACTIVE")
    HOME_PATTERN = Pattern.compile("home")
    UPDATE_PROFILE_PATTERN = Pattern.compile("update profile")
    FRAMES_PATTERN = Pattern.compile("your browser must support frames and javascript to use this application.")
    def doLogin(request, cookies, username, password):
        authenticationFormData = ( NVPair("username", username),
                                   NVPair("password", password),)
        result = doPostRequest(request, "login.do", authenticationFormData);
        #validateResponse(request, result, (FRAMES_PATTERN, ))
        validateResponse(request, result, (TOP_MENU_PATTERN, HOME_PATTERN, UPDATE_PROFILE_PATTERN))
        loadLeftSide(request, cookies)
        loadRightSide(request, cookies)

    # mainPage --------------------------------------------------------------------------------------------------
    MY_PARTNERS_PATTERN = Pattern.compile("my contacts")
    SHORTCUTS_PATTERN = Pattern.compile("my shortcuts")
    VISITORS_PATTERN = Pattern.compile("visitors")
    REFRESH_PATTERN = Pattern.compile("r_refresh.png")
    EXPAND_CONTACTS_PATTERN = Pattern.compile("icon-contacts-maximize.gif")
    CONTACT_PROFILE_PATTERN = Pattern.compile("viewProfile.do")
    def mainPage(request, cookies):
        result = doGetRequest(request, "start.do");
        #validateResponse(request, result, (FRAMES_PATTERN, ))
        result = doGetRequest(request, "start.do");
        validateResponse(request, result, (MY_PARTNERS_PATTERN, SHORTCUTS_PATTERN, VISITORS_PATTERN))

        result = doDWRRequest(request, cookies, "Blogs", "getFriendsBlogs", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-e3}"),NVPair("c0-param2", "string:hole"), ))
        #can be empty blog, will validate this after blog post
        #validateResponse(request, result, (REFRESH_PATTERN,))
        validateResponse(request, result, ())

        result = doDWRRequest(request, cookies, "Contacts", "getNewNetworkContacts", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-e4", "number:5"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-e3, itemQuantity:reference:c0-e4}")))
        contactsResult = result
        validateResponse(request, result, (EXPAND_CONTACTS_PATTERN,CONTACT_PROFILE_PATTERN ))

        loadLeftSide(request, cookies)
        loadRightSide(request, cookies)
        return contactsResult

    # popularNetworkContactsDWR ---------------------------------------------------------------------------------
    def popularNetworkContactsDWR(request, cookies):
        result = doDWRRequest(request, cookies, "Contacts", "getMostVisitedPersons", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-e4", "number:5"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2,
```

```
pageToNavigate:reference:c0-e3, itemQuantity:reference:c0-e4}"), ))
            validateResponse(request, result, ())


        # createMessagePage ---------------------------------------------------------------------------------------------
        MESSAGE_PATTERN = Pattern.compile("message:")
        def createMessagePage(request, cookies, personId):
            result = doGetRequest(request, "createMessage.do");
            validateResponse(request, result, (MESSAGE_PATTERN, ))

        # sendMessage ---------------------------------------------------------------------------------------------
        MESSAGE_SENT_PATTERN = Pattern.compile("your message has been sent successfully")
        def sendMessage(request, cookies, personId):
            formData = (NVPair("messagePerson", "%d"%personId),
                    NVPair("messageText","hello mister"),
                    NVPair("messages.action", "messages.replay"),
                    )
            result = doPostRequest(request, "sendMessage.do", formData);
            validateResponse(request, result, (MESSAGE_SENT_PATTERN, ))

        # saveBlog ---------------------------------------------------------------------------------------------
        PRIVATE_MESSAGES_PATTERN = Pattern.compile("private messages")
        SMS_MESSAGES_PATTERN = Pattern.compile("SMS messages")
        INVITATIONS_PATTERN = Pattern.compile("invitations")
        RECEIVED_MESSAGES_PATTERN = Pattern.compile("received messages")
        def saveBlog(request, cookies):
            formData = (NVPair("blogContent","kalimera :) "),
                    NVPair("blogScope", "hole"),
                    NVPair("blogScopeView", "hole"),
                    NVPair("blogTitle", "whatever..."),
                    NVPair("livejournal", "true"),
                    )
            result = doPostRequest(request, "saveBlog.do", formData);
            validateResponse(request, result, (MY_PARTNERS_PATTERN, SHORTCUTS_PATTERN,
VISITORS_PATTERN))

            result = doDWRRequest(request, cookies, "Blogs", "getFriendsBlogs", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3}"),NVPair("c0-param2", "string:hole"), ))
            validateResponse(request, result, (REFRESH_PATTERN,))

            loadLeftSide(request, cookies)
            loadRightSide(request, cookies)

        # messagesPage ---------------------------------------------------------------------------------------------
        def messagesPage(request, cookies):
            result = doGetRequest(request, "messages.do");
            validateResponse(request, result, (PRIVATE_MESSAGES_PATTERN, SMS_MESSAGES_PATTERN,
INVITATIONS_PATTERN))

            result = doDWRRequest(request, cookies, "Messages", "getMessages", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:1"), NVPair("c0-e3", "string:received"), NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, pageToNavigate:reference:c0-e2,
messageType:reference:c0-e3}")))
            validateResponse(request, result, (RECEIVED_MESSAGES_PATTERN,))

            loadLeftSide(request, cookies)
```

```
        # groupsPage ---------------------------------------------------------------------------------------------------
        MY_GROUPS_PATTERN = Pattern.compile("my groups")
        ALL_GROUPS_PATTERN = Pattern.compile("all groups")
        SORT_BY_PATTERN = Pattern.compile("Sort by")
        GROUP_MEMBERS_PATTERN = Pattern.compile("group members")
        def groupsPage(request, cookies):
            result = doGetRequest(request, "groups.do");
            validateResponse(request, result, (MY_GROUPS_PATTERN, ALL_GROUPS_PATTERN))

            result = doDWRRequest(request, cookies, "Groups", "getPersonGroups", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"),NVPair("c0-e4", "boolean:true"), NVPair("c0-e5",
"null:null"), NVPair("c0-e6", "string:MEMBERS_COUNT"), NVPair("c0-e7", "number:1"), NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3, mainPage:reference:c0-e4, targetPersonId:reference:c0-e5, sortColumnName:reference:c0-e6,
sortDirection:reference:c0-e7}")))
            validateResponse(request, result, (MY_GROUPS_PATTERN, SORT_BY_PATTERN,
GROUP_MEMBERS_PATTERN))

            loadLeftSide(request, cookies)
            return result

        # groupPage ----------------------------------------------------------------------------------------------------
        JOINING_GROUP_PATTERN = Pattern.compile("Joining to group:")
        SHOWING_GROUP_PATTERN = Pattern.compile("Showing members:")
        DESCRIPTION_GROUP_PATTERN = Pattern.compile("description:")
        REMOVE_FROM_MANAGERS_GROUP_PATTERN = Pattern.compile("remove from managers")
        NO_FILES_IN_GROUP_PATTERN = Pattern.compile("no files in group")
        FIRST_NAME_PATTERN = Pattern.compile("FirstName(\d*).")
        EMPTY_BLOG_PATTERN = Pattern.compile("Use the stage as you would any open forum with the
knowledge that you are posting your words")
        NO_PHOTOS_IN_GROUP_PATTERN = Pattern.compile("no photos in group")
        def groupPage(request, cookies, groupId):
            formData = (NVPair("groupId", groupId),
                    )
            result = doPostRequest(request, "groups.do", formData);
            validateResponse(request, result, (JOINING_GROUP_PATTERN, SHOWING_GROUP_PATTERN,
DESCRIPTION_GROUP_PATTERN))

            result = doDWRRequest(request, cookies, "Groups", "getPersonGroups", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"),NVPair("c0-e4", "boolean:true"), NVPair("c0-e5",
"null:null"), NVPair("c0-e6", "string:MEMBERS_COUNT"), NVPair("c0-e7", "number:1"), NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
e3, mainPage:reference:c0-e4, targetPersonId:reference:c0-e5, sortColumnName:reference:c0-e6,
sortDirection:reference:c0-e7}")))
            validateResponse(request, result, (MY_GROUPS_PATTERN, SORT_BY_PATTERN,
GROUP_MEMBERS_PATTERN))

            result = doDWRRequest(request, cookies, "GroupMedia", "getGroupFiles", (NVPair("c0-e1",
"number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"),NVPair("c0-e4", "number:%s"%groupId),
NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2,
pageToNavigate:reference:c0-e3, groupId:reference:c0-e4}")))
            validateResponse(request, result, (NO_FILES_IN_GROUP_PATTERN, ))

            result = doDWRRequest(request, cookies, "Contacts", "getGroupMembers", (NVPair("c0-e1",
"number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"),NVPair("c0-e4",
"number:%s"%groupId),NVPair("c0-e5", "number:20"), NVPair("c0-
param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-
```

```
e3, groupId:reference:c0-e4, itemQuantity:reference:c0-e5}")))
            validateResponse(request, result, (FIRST_NAME_PATTERN, ))

            result = doDWRRequest(request, cookies, "Blogs", "getFriendsBlogs", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-e3}"), NVPair("c0-param2", "string:%s"%groupId)))
            validateResponse(request, result, (EMPTY_BLOG_PATTERN, ))

            result = doDWRRequest(request, cookies, "GroupMedia", "getGroupPhotos", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"), NVPair("c0-e4", "number:%s"%groupId), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-e3, groupId:reference:c0-e4}")))
            validateResponse(request, result, (NO_PHOTOS_IN_GROUP_PATTERN, ))

            loadLeftSide(request, cookies)

        # rDrivePage ----------------------------------------------------------------------------------------------------
        ALL_MY_GALLERIES_PATTERN = Pattern.compile("all my galleries")
        MY_PRIVATE_GALLERY_PATTERN = Pattern.compile("My Private Gallery")
        CREATE_NEW_GALLERY_PATTERN = Pattern.compile("create new gallery")
        UPLOAD_NEW_PHOTO_PATTERN = Pattern.compile("upload new photo")
        VIEW_SLIDESHOW_PATTERN = Pattern.compile("view slideshow")
        def rDrivePage(request, cookies):
            result = doGetRequest(request, "listPhotos.do");
            validateResponse(request, result, (ALL_MY_GALLERIES_PATTERN, ))

            result = doDWRRequest(request, cookies, "Galleries", "getGalleriesForMediaSection", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "null:null"),NVPair("c0-e4", "number:1"), NVPair("c0-e5", "null:null"), NVPair("c0-e6", "null:null"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, targetPersonId:reference:c0-e3, pageToNavigate:reference:c0-e4, targetGalleryId:reference:c0-e5, targetSparcFolderId:reference:c0-e6}")))
            validateResponse(request, result, (MY_PRIVATE_GALLERY_PATTERN, CREATE_NEW_GALLERY_PATTERN))

            result = doDWRRequest(request, cookies, "Photos", "getPhotosForMediaSection", (NVPair("c0-e1", "number:0"), NVPair("c0-e2", "number:0"), NVPair("c0-e3", "null:null"),NVPair("c0-e4", "number:1"), NVPair("c0-e5", "null:null"), NVPair("c0-e6", "null:null"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1, currentPage:reference:c0-e2, targetPersonId:reference:c0-e3, pageToNavigate:reference:c0-e4, targetGalleryId:reference:c0-e5, targetSparcFolderId:reference:c0-e6}")))
            validateResponse(request, result, (UPLOAD_NEW_PHOTO_PATTERN, VIEW_SLIDESHOW_PATTERN))

            loadLeftSide(request, cookies)

        # image upload popup --------------------------------------------------------------------------------------------
        MAX_FILE_SIZE_PATTERN = Pattern.compile("maximum size for image uploads is 10 MB")
        def imageUploadPopup(request, cookies):
            result = doGetRequest(request, "uploadOnPhotoPopup.do");
            validateResponse(request, result, (MAX_FILE_SIZE_PATTERN, ))

        # upload image --------------------------------------------------------------------------------------------------
        IMAGE_UPLOAD_SUCCESSFULL_PATTERN = Pattern.compile("your photo has been uploaded successfully")
        def uploadImage(request, cookies):
            fileUploadFormData = ( NVPair("formName", "fileUpload"),
                                   NVPair("comment", "Poor hedgehog :)"))
            result = doUploadFile(request, "addPhoto.do", "formFile", "1002.jpg", fileUploadFormData)
```

```
        validateResponse(request, result, (IMAGE_UPLOAD_SUCCESSFULL_PATTERN, ))

    # sparcFilesPage ---------------------------------------------------------------------------------------------------
    MY_GALLERIES_PATTERN = Pattern.compile("my galleries")
    FILES_PATTERN = Pattern.compile("files")
    WEBLINKS_PATTERN = Pattern.compile("weblinks")
    ALL_FILES_PATTERN = Pattern.compile("all files")
    RECEIVED_FILES_PATTERN = Pattern.compile("received")
    SENT_FILES_PATTERN = Pattern.compile("sent")
    FILE_TABLE_PATTERN = Pattern.compile("file Size.*modified")
    def sparcFilesPage(request, cookies):
        result = doGetRequest(request, "listSparcFiles.do");
        validateResponse(request, result, (MY_GALLERIES_PATTERN, FILES_PATTERN,
WEBLINKS_PATTERN))

        result = doDWRRequest(request, cookies, "SparcFiles", "getFolders", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:-1"),NVPair("c0-e4", "number:1"), NVPair("c0-e5",
"number:0"), NVPair("c0-e6", "number:-1"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1,
currentPage:reference:c0-e2, targetPersonId:reference:c0-e3, pageToNavigate:reference:c0-e4,
filterMode:reference:c0-e5, parentFolderId:reference:c0-e6}")))
        validateResponse(request, result, (ALL_FILES_PATTERN, RECEIVED_FILES_PATTERN,
SENT_FILES_PATTERN))

        result = doDWRRequest(request, cookies, "SparcFiles", "getFiles", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:-1"),NVPair("c0-e4", "number:1"), NVPair("c0-e5",
"number:0"), NVPair("c0-e6", "number:-1"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-e1,
currentPage:reference:c0-e2, targetPersonId:reference:c0-e3, pageToNavigate:reference:c0-e4,
filterMode:reference:c0-e5, parentFolderId:reference:c0-e6}")))
        validateResponse(request, result, (FILE_TABLE_PATTERN,))

        loadLeftSide(request, cookies)

    # file upload popup ------------------------------------------------------------------------------------------------
    FILE_WILL_BE_PRIVATE_PATTERN = Pattern.compile("This file will be stored to a PRIVATE storage")
    def fileUploadPopup(request, cookies):
        result = doGetRequest(request, "preUploadSparcFile.do");
        validateResponse(request, result, (FILE_WILL_BE_PRIVATE_PATTERN, ))

    # upload file ------------------------------------------------------------------------------------------------------
    FILE_UPLOAD_SUCCESSFULL_PATTERN = Pattern.compile("file has been successfully uploaded to your
secure r-drive")
    def uploadFile(request, cookies):
        fileUploadFormData = ( NVPair("formName", "sparcUploadForm"),)
        result = doUploadFile(request, "uploadSparcFile.do", "formFile", "TDDCycle.doc", fileUploadFormData)
        # TODO Make it to create new file every time, not to override existing one
        validateResponse(request, result, (FILE_UPLOAD_SUCCESSFULL_PATTERN, ))

    # raitings page  ---------------------------------------------------------------------------------------------------
    PUBLIC_MEDIA_PATTERN = Pattern.compile("public media")
    TOP_TODAY_TAB_TITLE_PATTERN = Pattern.compile("top <br> today")
    def raitingsPage(request, cookies):
        result = doGetRequest(request, "ratings.do");
        validateResponse(request, result, (PUBLIC_MEDIA_PATTERN, ))

        result = doDWRRequest(request, cookies, "Tops", "getPhotosTop", (NVPair("c0-e1", "number:0"),
NVPair("c0-e2", "number:0"), NVPair("c0-e3", "number:1"),NVPair("c0-e4", "boolean:false"), NVPair("c0-e5",
"boolean:false"), NVPair("c0-e6", "number:25"), NVPair("c0-param1","Object_Object:{pageQuantity:reference:c0-
e1, currentPage:reference:c0-e2, pageToNavigate:reference:c0-e3, findAll:reference:c0-e4,
```

```
inMyNetwork:reference:c0-e5, itemQuantity:reference:c0-e6}")))
            validateResponse(request, result, (TOP_TODAY_TAB_TITLE_PATTERN, ))

            loadLeftSide(request, cookies)


        # persons profile page  ----------------------------------------------------------------------------------------------
        PERSONS_PUBLIC_PROFILE_PATTERN = Pattern.compile("Public stuff")
        PERSONS_CONTACTS_PATTERN = Pattern.compile("My people")
        PERSONS_FILTER_PATTERN = Pattern.compile("filter")
        PERSONS_VIDEO_PATTERN = Pattern.compile("video")
        PERSONS_DOCUMENTS_PATTERN = Pattern.compile("documents")
        PERSONS_TAG_PATTERN = Pattern.compile("tagValue")
        PERSONS_CONTACT_PATTERN = Pattern.compile("FirstName(\d*).LastName(\d*)")

        def personsProfilePage(request, cookies, personId):
            result = doPostRequest(request, "viewProfile.do",(NVPair("targetPersonId", personId),));
            validateResponse(request, result, (PERSONS_PUBLIC_PROFILE_PATTERN,
PERSONS_CONTACTS_PATTERN, ))

            result =  doDWRRequest(request, cookies, "TargetProfile", "getMultimediaForTargetPerson", (NVPair("c0-
e1", "number:%d"%personId), NVPair("c0-param1","Object_Object:{targetPersonId:reference:c0-e1}")))
            validateResponse(request, result, (PERSONS_FILTER_PATTERN, PERSONS_VIDEO_PATTERN, ))
            #TODO maybe put some multimedia here?

            result =  doDWRRequest(request, cookies, "TargetProfile", "getAllFilesForTargetPerson", (NVPair("c0-
e1", "number:%d"%personId), NVPair("c0-param1","Object_Object:{targetPersonId:reference:c0-e1}")))
            validateResponse(request, result, (PERSONS_FILTER_PATTERN,
PERSONS_DOCUMENTS_PATTERN, ))
            #TODO maybe put some documents here? Why there are not documents?

            result = doDWRRequest(request, cookies, "Tag", "getFillRequest", (NVPair("c0-e1",
"number:%d"%personId), NVPair("c0-param1","Object_Object:{targetPersonId:reference:c0-e1}"), NVPair("c0-
param2", "number:%d"%personId),NVPair("c0-param3", "string:person"),))
            validateResponse(request, result, (PERSONS_TAG_PATTERN, ))
            #TODO maybe put some tags here?

            result =  doDWRRequest(request, cookies, "Contacts", "getContactsForTargetPerson", (NVPair("c0-e1",
"number:%d"%personId), NVPair("c0-param1","Object_Object:{targetPersonId:reference:c0-e1}")))
            validateResponse(request, result, (PERSONS_CONTACT_PATTERN, ))

            loadLeftSide(request, cookies)


        # doLogout ----------------------------------------------------------------------------------------------
        def doLogout(request):
            result = doGetRequest(request, "logout.do");
            validateResponse(request, result, (LOGIN_FORM_PATTERN,))

        # Grinder wrapers for test methods
        loginPageTest =  Test(1, "WeAreYou login page").wrap(loginPage)
        doLoginTest =  Test(2, "WeAreYou login").wrap(doLogin)
        mainPageTest =  Test(3, "WeAreYou main page").wrap(mainPage)
        popularNetworkContactsDWRTest = Test(4, "WeAreYou popular user list
(dwr)").wrap(popularNetworkContactsDWR)
        createMessagePageTest =  Test(5, "WeAreYou send message window").wrap(createMessagePage)
        sendMessageTest =  Test(6, "WeAreYou message sending").wrap(sendMessage)
        saveBlogTest =  Test(7, "WeAreYou blog post").wrap(saveBlog)
```

```
messagesPageTest =  Test(8, "WeAreYou Messages main page").wrap(messagesPage)
groupsPageTest =  Test(9, "WeAreYou Groups main page").wrap(groupsPage)
groupPageTest =  Test(10, "WeAreYou Group page").wrap(groupPage)
rDrivePageTest =  Test(11, "WeAreYou R-Drive main page").wrap(rDrivePage)
imageUploadPopupTest =  Test(12, "WeAreYou Photo upload popup").wrap(imageUploadPopup)
uploadImageTest =  Test(13, "WeAreYou Photo upload").wrap(uploadImage)
sparcFilesPageTest =  Test(14, "WeAreYou Sparc files page").wrap(sparcFilesPage)
fileUploadPopupTest =  Test(15, "WeAreYou Sparc file upload popup").wrap(fileUploadPopup)
uploadFileTest =  Test(16, "WeAreYou Sparc file upload").wrap(uploadFile)
raitingsPageTest =  Test(16, "WeAreYou Raitings main page").wrap(raitingsPage)
personsProfilePageTest = Test(17, "WeAreYou Persons profile page").wrap(personsProfilePage)
doLogoutTest =  Test(18, "WeAreYou logout action").wrap(doLogout)

#Test bootstrap class
profile = getProfile()
class TestRunner:

   # This method is called for every run.
   def __call__(self):
      request = HTTPRequest(url = WEBSPACE_URL)
      cookies = WebspaceCookiePolicyHandler()
      CookieModule.setCookiePolicyHandler(cookies)
      messageReceiverId = grinder.threadID + 2

      credentials = getCredentials()

      loginPageTest(request)
      grinder.sleep(WAIT_TIME)

      doLoginTest(request, cookies, credentials.name, credentials.value)
      grinder.sleep(WAIT_TIME)


      testId = 0
      while testId >= 0:
         contactsDwrResult = mainPageTest(request, cookies)
         grinder.sleep(WAIT_TIME)
         #popularNetworkContactsDWRTest(request, cookies)

         testId = pickupAction(profile)
         log("testId: %d"%testId)
         if testId == 0:
            log("Messaging")
            createMessagePageTest(request, cookies, messageReceiverId)
            grinder.sleep(WAIT_TIME)

            sendMessageTest(request, cookies, messageReceiverId)
            grinder.sleep(WAIT_TIME)

            messagesPageTest(request, cookies)
            grinder.sleep(WAIT_TIME)
         elif testId == 1:
            log("Image/Video upload/dwnload")
            rDrivePageTest(request, cookies)
            grinder.sleep(WAIT_TIME)

            imageUploadPopupTest(request, cookies)
            grinder.sleep(WAIT_TIME)
```

```
        uploadImageTest(request, cookies)
        grinder.sleep(WAIT_TIME)
    elif testId == 2:
        log("File upload/dwnload")
        rDrivePageTest(request, cookies)
        grinder.sleep(WAIT_TIME)

        sparcFilesPageTest(request, cookies)
        grinder.sleep(WAIT_TIME)

        fileUploadPopupTest(request, cookies)
        grinder.sleep(WAIT_TIME)

        uploadFileTest(request, cookies)
        grinder.sleep(WAIT_TIME)

        sparcFilesPageTest(request, cookies)
        grinder.sleep(WAIT_TIME)
    elif testId == 3:
        log("Blog reading")
    elif testId == 4:
        log("Blog Writing")
        saveBlogTest(request, cookies)
        grinder.sleep(WAIT_TIME)
    elif testId == 5:
        log("Profiles/Contacts")
        matcher = personIDPattern.matcher(contactsDwrResult)
        matcher.find();
        personId = matcher.group(1)
        log("Using person with id %s"%personId)

        personsProfilePageTest(request, cookies, personId)
    elif testId == 6:
        log("Tops/Stats")
        raitingsPageTest(request, cookies)
        grinder.sleep(WAIT_TIME)
    elif testId == 7:
        log("Groups")
        groupsResult = groupsPageTest(request, cookies)
        grinder.sleep(WAIT_TIME)

        #Finding some group id to use later
        matcher = groupIDPattern.matcher(groupsResult)
        matcher.find();
        groupId = matcher.group(1)
        log("Using group with id %s"%groupId)

        groupPageTest(request, cookies, groupId)
        grinder.sleep(WAIT_TIME)

doLogoutTest(request)
```

DOKUMENTĀRĀ LAPA

Maģistra darbs „Interneta informācijas sistēmu slodzes un mērogojamības testēšana izmantojot bezmaksas rīkus”.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai. Piekrītu sava darba publicēšanai internetā.

Autors: _____

(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījusi augstāk minēto maģistra darbu un atzīstu to par piemērotu/nepiemērotu (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____

(Vadītāja paraksts)

Darbs iesniegts Datorikas nodaļā _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Metodiķe: _____

(Metodiķes paraksts)

Recenzente: _____

(Recenzentes paraksts)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)