TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu Kolledž

HAPROXY JA PGBOUNCERI RAKENDAMINE EKSAMITE INFOSÜSTEEMIS

HAPROXY AND PGBOUNCER INTEGRATION IN EXAMINATION INFORMATION SYSTEM

RAKENDUSKÕRGHARIDUSTÖÖ

Üliõpilane: Jay Martin Ploomipuu

Üliõpilaskood: 183537EDTR

Juhendaja: Sten Aus, taristu büroo juhataja

Kaasjuhendaja: Ago Rootsi, lektor

Tartu 2020

(Tiitellehe pöördel)

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"....." ..... 202…..

Autor: /allkirjastatud digitaalselt/

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

"....." ..... 202…..

Juhendaja: /allkirjastatud digitaalselt/

Kaitsmisele lubatud

"....."..... 202… .

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina Jay Martin Ploomipuu (a*utori nimi*) (sünnikuupäev: 27.12.1997)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose HAProxy ja PgBounceri rakendamine Eksamite Infosüsteemis,

(*lõputöö pealkiri*)

mille juhendaja on Sten Aus,

(*juhendaja nimi*)

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.*

/allkirjastatud digitaalselt/

_____ (*kuupäev*)

TalTech Tartu Kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Jay Martin Ploomipuu, 183537EDTR

Õppekava, peaeriala: EDTR17/18 - Telemaatika ja arukad süsteemid

Juhendajad: Taristu büroo juhataja, Sten Aus
Haridus- ja Noorteamet, 730 2117, sten.aus@harno.ee
Lektor, Ago Rootsi

Lõputöö teema:

(eesti keeles) HAProxy ja PgBounceri rakendamine Eksamite Infosüsteemis

(inglise keeles) HAProxy and PgBouncer integration in Examination Information System

Lõputöö põhieesmärgid:

1. Eksamite Infosüsteemi võimekuse parandamine
2. Rakenduste proksi väljavahetamine
3. Andmebaaside hajutamine
4. Andmebaasi puul tarkvara rakendamine
5. Infosüsteemi võimekus teenindada 4000 üheaegset kasutajat

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Uue proksi rakendamine	11.2018
2.	Andmebaaside hajutamine	07.2019
3.	Puul tarkvara rakendamine	12.2019

Töö keel: eesti keel **Lõputöö esitamise tähtaeg:** "....."..........202….a

Üliõpilane: Jay Martin Ploomipuu /allkirjastatud digitaalselt/ "....."..........202….a

Juhendaja: Sten Aus /allkirjastatud digitaalselt/ "....."..........202….a

Kaasjuhendaja: Ago Rootsi /allkirjastatud digitaalselt/ "....."..........202….a

Kinnise kaitsmise ja/või lõputöö avalikustamise piirangu tingimused formuleeritakse pöördel

SISUKORD

EESSÕNA

Lõputöö algatajaks on Haridus- ja Noorteamet, mis 2020. aasta augustikuuni kandis nime Hariduse Infotehnoloogia Sihtasutus. Lõputöö on osa asutuse aastaplaanidest. Ametipoolseks juhendajaks on tehnoloogia juhtimise osakonna taristu büroo juhataja Sten Aus ning koolipoolseks juhendajaks lektor Ago Rootsi.

Autor tänab koolipoolset kaasjuhendajat Ago Rootsit ja töökohapoolset juhendajat Sten Ausi informatiivse tagasiside ja kannatliku juhendamise eest, tiimikaaslaseid ning EISi projektijuhti ja arendajat meeldiva koostöö eest.

Mõistete ja lühendite loetelu

Backend	Eesti k tagakomponent, kasutajatele nähtamatu süsteemi põhiosa, mis töötleb informatsiooni
Frontend	Eeskomponent, mis on enamjaolt rakendatud HTTPS ühendusele vastuvõtmisele ning päringute edastamise tagakomponendile
HTTP	Turvamata veebiliiklusse protokoll
HTTPS	Sertifikaatidega krüpteeritud veebiliikluse protokoll
Load balancing	Koormuse jaotamine süsteemi komponentide vahel
Header	Eesti k päis, kasutatakse info edastamiseks näiteks veebiliikluse protokollide puhul (HTTP-HTTPS)
Pool	Eesti k puul, tarkvara tüüp või funktsiooni olek, mis hoiab endas ühendusi ning vastavalt seadistusele suunab, piirab või katkestab ühendusi
SSL offload	Ühenduse lahti kodeerimine ja edastamine sihtserverile

SISSEJUHATUS

Tänapäevastes infosüsteemides tõuseb kasutajate arv iga päevaga ning eriti suur tõus toimub haridusega seotud infosüsteemides [1]. Õpilastele luuakse suures mahus elektroonilisi materjale, mis võimaldavad luua interaktiivseid ja personaalseid õppimisvõimalusi aidates samal ajal haridusasutustel säästa loodust vähendades massilist õpikute ja töövihikute printimist.

Infosüsteemi populaarsuse kasvuga kaasneb mitmeid positiivseid külgi, kuid paratamatult tuleb andmete koguse suurenemisega kaasa palju lisatööd. On hea meel tõdeda, et hariduse poolet tekib infosüsteemidesse palju uut materjali, mis võimaldavad õpilastel õppida kasutades uusi innovatiivseid meetodeid. Andmete koguse suurenemisega muutuvad andmebaasid mahukamaks ning suureneb andmebaasi tehtavate päringute arv, lisaks muutuvad keerukamaks infosüsteemide rakendused, kuhu lisatakse juurde uusi funktsioone arendustöödega, et suurenda infosüsteemi populaarsust ja käideldavust ning tagada parimad võimalused õpetamiseks. Koormuse tõus infosüsteemide komponentidele suurendab vajadust uute ülesehituste rakendamisele ja süsteemi osade optimeerimisele. Suurema kasutuskoormuse tõusu mõjutavad suuremad sündmused - hetkelises olukorras on suureks mõjutajaks Covid-19 levik, mis sunnib paljud koolid hübriid- või distantsõppele suunduma [1].

Lõputöö eesmärgiks on suurendada Eksamite Infosüsteemi (edaspidi EIS) käideldusvõimet. EISi koormustaluvus ei olnud piisavalt optimeeritud, millest tulenevalt on tarvis leida uusi võimalusi jõudluse parandamiseks. Probleemseteks kohtadeks on ebaefektiivse koormusjaoturi rakendamine kuue infosüsteemi rakenduse ees ning ülekoormatud andmebaasi ülesehitus. Lõputöö käigus otsitakse uusi või optimeeritud lahendusi kahele eelnimetatud probleemile.

Lõputöö koosneb kolmest põhiosast. Esimene põhiosa keskendub infosüsteemi rakendusete ees kasutatava Apache'i proksi väljavahetamisele ja uue tarkvara rakendamisega kaasatulevate muudatuste rakendamisele. Teine põhiosa hõlmab andmebaasi võimekuse tõstmist optimeerimise ja uue tarkvara rakendamise abil. Viimases osas on võetud kokku infosüsteemi vastupidavus 2020. aasta septembri viimases pooles toimunud tasemetöödele. Lisaks on iga süsteemi muudatuste järel analüüsitud koormustaluvuse tõusu Apache JMeter tarkvara abil.

1. Rakenduse proksi tarkvarad

Eksamite Infosüsteem kasutab hetkel proksi tarkvarana Apache'i, mis omab kõiki vajalikke funktsioone, aga ei ole võimeline vastu pidama suurele koormusele. Halb koormustaluvus tuleneb sellest, et Apache ei ole algselt mõeldud kasutamaks kui koormusjaotur, vaid tegemist on veebiserveriga. Põhiliseks probleemiks on asjaolu, et tarkvara võimaldab küll suure koguse ühendusi ära teenindada, kuid edasist arenguruumi enam ei ole ning võttes arvesse infosüsteemi populaarsuse kasvu tuleb leida praegusele tarkvarale asendus [2].

Algset valikut proksi tarkvara puhul mõjutavad kaks funktsiooni - *SSL offload* ja *load balancing*. *SSL offload*i eesmärgiks on ühenduse lahti kodeerimine ja edastamine sihtserverile turvalises võrgus, mis ei ole ühendatud avaliku internetiga. Kasutaja, kes siseneb veebilehele, mis kasutab HTTPS protokolli, loob krüpteeritud ühenduse proksiserveriga kasutades sertifikaate. *SSL offload*i ei teostata enne, kui kasutaja ei ole proksiserveriga ühendust loonud. Liigutades ühenduse krüpteerimise rakendusserverist eraldi proksiserverisse on võimalik vähendada teenusmasina koormust, mis omakorda võimaldab suuremat koormuse taluvust. *Load balancing* ehk koormusjaotus võimaldab rakendada mitu rakendusserverit jagades sissetuleva liikluse sihtserverite vahel.

NGINX on sama populaarne ja laialdaselt kasutatud tarkvara nagu Apache. Tarkvara on samuti kasutusel veebiserverina, kuid jõudluse poolest peaks olema parem kui Apache. NGINXi eelis Apache'i ees seisneb selles, et tarkvara on loomisel on kohe algusest peale arendajad pannud rõhku ka proksi toele. Apache on võimas oma funktsionaaluste poolest, mis tulevad erinevate lisade rakendamisest, kuid funktsionaalsused on efektiivsed teatud piirideni [3].

HAProxy on täielikult proksiserver nagu viitab ka nimi. Võimeline edasi suunama nii TCP kui HTTP protokolli põhiseid ühendusi, mistõttu on võimeline tegema *SSL offload*i. Teise põhilise funktsionaalsuse poolest omab HAProxy koormusjaoturi tuge, võimaldades jagada koormust serverite vahel erinevatel viisidel. Lisaks on tarkvara kogunud palju tunnustust olles kiire ja efektiivne [4].

Proksi tarkvarasid on mitmeid ning kõiki ei ole võimalik läbi katsetada, milles tulenevalt langeb otsus praegu HAProxy kasukus, kuna sellega on EENetis kõige vähem kogemust ning uue tarkvara avastamine annab uusi kogemusi. HAProxy kasutuselevõtu ebaõnnestumisel on võimalus rakendada NGINX või liikuda tagasi vana ülesehituse juurde. HAProxy ja NGINXi eelisteks on asjaolu, et mõlemad on võimelised kasutama

mitut tööprotsessi korraga, mis võimaldab rakendustel kasutada protsessorit, millel on rohkem kui üks tuum.

1.1 Testsüsteemi ehitamine

HAProxy testimiseks tuleb ehitada testsüsteem, mis peaks ideaalis koosnema vähemalt kolmest serverist, üks server on puhtalt HAProxy käsutuses ning ülejäänud kaks serverit tuleb seadistada rakendusserveriteks, mis töötavad Apache veebiserveri tarkvaral, koos mõningate lisadega. Esimese ülesehitusena tuleb tööle saada esialgne seadistus kõikide serverite puhul, et sealt edasi liikuda erinevate funktsioonide testimiseni.

1.1.1 Tarkvara seadistuse kirjeldamine

HAProxy seadistamisel peab kasutama etteantud plokke, mis võimaldavad seadistada erinevaid funktsioone ja eesmärke. Globaalses (*global*) plokis on võimalik seadistada töötamise funktsioon, mis mõjub kogu tarkvarale. Võimalus on tuunida erinevaid limiite nagu ühenduste arv ja puhvrite suurus. Järgmiseks tähtsaks plokiks on vaikeväärtuste (*defaults*) osa, mis võimaldab seadistada vaikesätted kõikidele järgnevatele koodiplokkidele. Sissetulev liikus pannakse kirja ühendust vastuvõtvas (*frontend)* plokis ning edasisuunatud liiklus, mis peab jõudma rakendusteni on kirjeldatud *backend* osas. *Backend* ja *frontend* osasid saab olla mitmeid.

Frontend osas tuleb määrata millistel portidel ja IP-aadressidel rakendus kuulab ehk ootab ühenduste saabumist. Iga porti on võimalik seadistada erinevalt, hetkel on vaja seadistada ainult kaks porti – 80 ja 443, mis vastavad HTTP ja HTTPS veebiliiklusele. HTTPS ehk 443 tuleb lisaks juurde määrata veebiserveri sertifikaatide asukohad. Sertifikaat on vajalik turvalise veebiühenduse saavutamiseks, et veebilehel sisestatud info liiguks ainult krüpteerituna. Eduka HTTPS ühenduse saavutamiseks peab veebiserveri või ühendust vastuvõttev proksi edastama sertifikaadid ja avaliku võtme. Iga sertifikaadi jaoks on privaatne võti, mis ei tohi sattuda kolmanda osapoole kätte. Sertifikaadid on üles ehitatud usaldusahelale. Juursertifikaadid on operatsiooni-süsteemides ja/või internetilehitsejates. Juursertifikaatidega on allkirjastatud n-ö vahesertifikaat (või vahesertifikaadid) ja seejärel serverisertifikaat. Kehtiva usaldusahela jaoks on vaja esitada serveril altpoolt alustades serverisertifikaat ja kõik vahesertifikaadid kuni juursertifikaadini. Tavaliselt juursertifikaati ühendusega kaasa ei

lisata, kuna see on kliendis juba olemas. HAProxy üheks eripäraks on sertifikaatide haldus, võimalus on lisada kõik vajalikud sertifikaadid ja võtmed ühte faili või jagada domeenipõhiselt erinevatesse failidesse ning programmile edastada sertifikaatide nimekiri. Sertifikaadi kontrolli tegemine ühenduse loomisel (*frontend* osas) võimaldab proksil luua juba krüpteerimata ühenduse päringut teenindava rakendusserveriga. Proksi ja rakendusserverite vaheline suhtlus toimub sisevõrgus, mis ei ole ühendatud avaliku Internetiga, sellest tulenevalt ei pea muretsema sisevõrgus toimuva liikluse krüpteerituse üle. Vastavalt vajadusele võib teenusserverite ühenduse eraldada võrgusegmendiga (VLANiga).

*Backen*d osas saab määrata rakendusserverid ja koormusjaotuse ülesehituse. Kõik ühe infosüsteemi rakendusserverid on võimalik seadistada ühes *backend* osas, ideena iga *backend* kujutab endast ühte infosüsteemi või rakenduse mingit osa. Koormuse jagamiseks on erinevad võimalused:

- *roundrobin* – päringud jagatakse järjest erinevatele rakendusserveritele;
- *static-rr* – päring jagatakse järjest, kuid üks sessioon kinnistatakse ühele rakendusele;
- *leastconn* – päring antakse kõige vähem koormatud serverile;

Erinevaid võimalusi koormuse jagamiseks on kokku kümme, lisaks on võimalik koormust jaotada veel „kaalu" (ingl k *weight*) põhjal, viies sunniviisiliselt koormuse jaotuse erinevaks.

Backend osa võimaldab teha rakenduste kontrolli ehk kui tarkvara tuvastab, et rakendus päringule ei vasta etteantud aja jooksul, siis suunatakse sama ühendus järgmisesse rakendusserverisse. Rakenduste nimekiri koosneb rakendusele antavast nimest IP-aadressist ja pordist. Samuti on nimepõhiselt võimalik rakendada lisareeglid – näiteks kinda URLiga pöördumise korral suunata kindlasse rakendusserverisse (nt staatilise sisu kuvamine vmt).

1.1.2 Test süsteemi seadistamine

Esialgse HAProxy seadistuse saavutmiseks peab rakendama kahte *backend* ja ühte *frontend* osa. Teine *backend* on vajalik automaatseks serverisertifikaatide uuendamiseks. Sertifikaatide uuendamistarkvara töötab eraldi rakendustarkvarast, seega on tal erinev IP-aadress rakendusserveritest ja teenusele on avatud port proksiserveri sees. Tarkvarale suunamine tehakse ACL (ingl k *Access Control Lists*) ehk

reeglite kogumikuga, mis analüüsivad HTTP päringut ja URLi alusel suunavad päringu sertifikaatidega tegelevale *backend*ile.

Turvalise ülesehituse tagamiseks tuleb teha automaatne suunamine HTTP pealt HTTPSi peale (juhul kui serveriga luuakse ühendus kasutades porti 80). Esialgne ülesehitus on välja toodud lisades (lisa 1). Rakendusepoolne seadistus on lühem, kuna Apache peab ootama sissetulevat liiklust ja suunama õigesse kasuta vastavalt domeenile – Apache'i esialgne seadistus on lisades (lisa 2).

Algseadistusega suudab HAProxy suunata kliendi päringud kordamööda erinevatele rakendustele. Rakenduste päringute teenindamist saab testida näiteks erinevate lehtede kuvamisega või lehe tiitli muutmisega. Ühenduste testimiseks saab kasutada veel veebilehitseja lehe värskendusfunktsiooni või ühenduda serveriga mitmel korral erinevate veebilehitseja akendega. Tulemuseks on positiivne *roundrobin* rakendamine, mis tuleb rakendada Eksamite Infosüsteemis sarnaselt. HAProxy jagab sissetuleva liikluse rakendusserverite nimekirja põhjal järjest, välja arvatud juhtudel kui mõni rakendus ei ole käsitsi märgitud kõlbmatuks või proksi tarkvaral puudub ühendus nimetatud rakendusega.

1.2 ID-kaardi tugi

ID-kaardiga autentimine põhineb sertifikaadi kontrollimisel. Kasutusel on kaks sertifikaati, üks isikutuvastamiseks ja teine allkirjastamiseks. Proksis peab rakendama isikutuvastamise sertifikaadi kontrolli. Sertifikaat põhineb salajasel võtmel, mis on omakorda kaitstud parooliga, millest tulenevalt sertifikaadi esitamiseks tuleb sisestada PIN kood (parool). ID-kaardiga autentimine on kahe faktoriga isiku tuvastamine, kasutaja peab omama füüsiliselt plastikkaarti ning teadma salajase võtme parooli. Autentimise esimeseks etapiks on sertifikaadi väljastaja kontrollimine, mille käigus kontrollitakse, kas sertifikaat pärineb usaldatud sertifikaadiahelast. Tulenevalt töö kirjutamise ajal kehtivatest seadustest ja kokkulepetest on Eestis kehtivad ID-kaartide sertifikaatide koostööpartner SK ID Solutions AS [5].

SK ID Solutions AS-i poolt allkirjastatud sertifikaatide juursertifikaadid on kättesaadavad Internetis SK ID Solutions AS veebilehelt. Peale sertifikaadi usaldusahela kontrolli järgneb sertifikaadi kehtivuse kontroll. Sertifikaadi kehtivuse kontrollimiseks peab proksi suutma teha kehtivuse kontrolli tühistusnimekirja vastu. Lisaks lokaalsele

tühistusnimekirjale on olemas OCSP (ingk k *Online Certificate Status Protocol*) sertifikaatide oleku kontrollimine reaalajas võimalik rakendada proksis ja rakenduses, kuid teenus on tasuline. Tühistusnimekirja sertifikaadid saab alla laadida SK ID Solutions veebilehelt, failides on kirjas kõik Eestis väljastatud ID-kaartide sertifikaadid, millel on kehtivus peatatud. Kehtivuse kaotanud sertifikaadi jõudmine nimekirja võib aega võtta kuni 12 tundi [6]. Tühistusnimekirja negatiivseks küljeks on nimekirjas olevate sertifikaatide uuenemine. Peale igat tühistusnimekirja täiendamist vajab proksi tarkvara taaskäivitamast, kuna tarkvara loeb ja salvestab sertifikaatide info mällu, pidev taaskäivitamine võib hakata segama infosüsteemi kasutamist, sest tulenevalt tühistusnimekirja suurusest toimub proksi taaskäivitamine aeglaselt (üle ühe minuti). ID-kaardiga sisselogimist saab rakendada ilma tühistusnimekirjata, kuid siis peab sertifikaadi kehtivuse kontrolli rakendama kas rakenduse poolel või jätta sertifikaadid kontrollimata.

ID-kaardi toe rakendamine on järgmine ülesanne, millega HAProxy peab hakkama saama. Varasemalt on EISis ID-kaardiga autentinud kasutajad suunatud eraldi domeenile, millest tulenevalt peab rakendama sarnase ülesehituse testsüsteemi proksi puhul. Lisaks tuleb seadistada rakendusservereid kuvama informeerivat lehekülge juhtudel, kui süsteemiadministraator on ühendatud ID-kaardiga, aga samas peab veenduma, et turvatud veebi osa ei oleks kättesaadav avalikult liideselt.

HAProxy tarkvaraga on mitu erinevat võimalust tühistusnimekirja ja juursertifikaatide rakendamiseks. Üheks lahenduses on seadistada domeenide nimekiri eraldi faili, kus määrata ID-kaarti kasutavata domeeni juurde vastavad parameetrid, et toimuks ID-kaardiga autentimine. Tulenevalt Eksamite Infosüsteemi ülesehitusest ei ole vaja rakendada tühistusnimekirja kontrolli, kuid vastav funktsionaalsuse võimekus on tarkvaral olemas. Testsüsteemis tuleb lisada Apache'i seadistusfaili juurde üks lisaplokk, mis on mõeldud neile kasutajatele, kes on ennast tugevalt autentinud ehk sisenenud ID-kaardiga. Turvatud osale on ligipääs lubatud ainult domeeni põhjal ning domeeni külastamisel nõuab HAProxy kasutajalt ID-kaarti, vastasel juhul veebile ligipääs puudub ning kasutajale kuvatakse veebilehitseja poolt vastav veateade.

1.2.1 Päiste edastamine

Vaikeseadistusega ei edastata rakendusele ID-kaardi sertifikaati ega muud ID-kaardiga seotud infot (veebipäringu päistega). Eksamite Infosüsteemil on kasutusel kindlad

päised isiku tuvastamiseks ning millest tulenevalt seatakse üles isikupärane keskkond. EISis kasutatavad päised on loetletud koodilõigus 1.1.

```
SSL_CLIENT_CERT
SSL_CLIENT_I_DN
SSL_CLIENT_I_DN_CN
SSL_CLIENT_M_SERIAL
SSL_CLIENT_S_DN
SSL_CLIENT_S_DN_CN
SSL_CLIENT_VERIFY
```

Koodilõik 1.1. EISis kasutatavad SSL päised

Päiste edastamiseks tuleb seadistada soovitud päiste ülesehitus proksi seadistusfailis. Tulenevalt rakenduste eripärast on vaja osasid päiseid täiendada. Kliendi sertifikaati ehk SSL-CLIENT-CERT päis tuleb rakendusele saata base64 vormingus, kuna sellise ülesehitusena töötab Apache, vaikimisi edastatakse päise sisu binaarvormingus. Binaar-vormingus saadab proksi veel SSL-CLIENT-M-SERIAL päise, mis tuleb rakenduse toimimiseks muuta kuueteistkümnendik vormingus. SSL-CLIENT-VERIFY päis saadetakse edasi numbrilise väärtusena – kui usaldusahel on kinnitatud, siis on väärtus 0, muudel juhtudel on väärtus midagi muud. Koodilõigus 1.2. on välja toodud HAProxy päiste seadistused, vasakpoolne tulp on proksi poole ülesehitus koos vajaminevate täiendustega ning paremal pool on päise seadistus koos nimega. Nimi on jäetud üks-ühele Apache ülesehitusega, et päise sisu oleks üheti mõistetav. Kõiki EISis kasutatavaid päiseid on võimalik edastada rakendusele, välja arvatud ühte – SSL-CLIENT-VERIFY, mida rakendus soovib saada sõnana SUCCESS.

```
http-request set-header SSL-CLIENT-CERT          %{+Q}[ssl_c_der,base64]
http-request set-header SSL-CLIENT-I-DN          %{+Q}[ssl_c_i_dn]
http-request set-header SSL-CLIENT-I-DN-CN       %{+Q}[ssl_c_i_dn(cn)]
http-request set-header SSL-CLIENT-M-SERIAL      %[ssl_c_serial,hex]
http-request set-header SSL-CLIENT-S-DN          %{+Q}[ssl_c_s_dn]
http-request set-header SSL-CLIENT-S-DN-CN       %{+Q}[ssl_c_s_dn(cn)]
http-request set-header SSL-CLIENT-VERIFY        %[ssl_c_verify]
```

Koodilõik 1.2. HAProxy päised

Tulenevalt süsteemi ülesehitusest kasutab Eksamite Infosüsteem päiste väärtustena Apache'i väärtuseid, mis on vaja edastada HAProxyil samamoodi rakendusele loetavalt. Proksi seadistusfailis saab määrata soovitud nimed päistele, kuid Apache'i tarkvara täiendab neid nimesid veel omakorda HTTP prefiksi väärtusega, sest tegu on väliste muutujatega. Proksi ja veebiserveri eripärast tingitult on vaja teha päiste

ümberseadistamist Apache'i seadistusfailis ka rakendusserveris. Sissetulevate päiste seadistus on välja toodud 1.3. koodilõigus. Tulenevalt proksi mittevõimekusest on vaja seadistada SSL-CLIENT-VERIFY käsitsi. Väärtuse käsitsi seadistamise tulemusena asendatakse kliendi kontrolli väärtus alati SUCCESSiga kui sissetulev väärtus on 0. Teistel juhtudel päringuid ei jõua rakenduseni, kuna kliendi kontroll ebaõnnestus ning sertifikaadi kontrolli teeb HAProxy. HAProxy ülesehitus võimaldab kasutaja edasi saata valitud veakoodide piires. Päise seadistus tehakse SetEnvIf lause põhjal, kus sissetuleva päise väärtus lisatakse Apache'i kohaliku päise väärtuse asemele.

```
SetEnvIf SSL-CLIENT-CERT "(..*)" SSL-CLIENT-CERT=$1
SetEnvIf SSL_CLIENT_I_DN "(..*)" SSL-CLIENT-I-DN=$1
SetEnvIf SSL-CLIENT-I-DN-CN "(..*)" SSL-CLIENT-I-DN-CN=$1
SetEnvIf SSL-CLIENT-M-SERIAL "(..*)" SSL-CLIENT-M-SERIAL=$1
SetEnvIf SSL-CLIENT-S-DN "(..*)" SSL-CLIENT-S-DN=$1
SetEnvIf SSL-CLIENT-S-DN-CN "(..*)" SSL-CLIENT-S-DN-CN=$1
SetEnvIf SSL-CLIENT-VERIFY "0" SSL-CLIENT-VERIFY=SUCCESS
```

Koodilõik 1.3. Sissetulevate päiste ümberseadistamine Apache's

1.3 HAProxy rakendamine EISis

HAProxy server tuleb EISi jaoks seadistada sarnaselt nagu sai tehtud testsüsteemis. Kuna kasutatakse pikema kehtivusajaga sertifikaate, siis ei ole vaja eraldi *backend*i veebisertifikaatide taotlemiseks. Infosüsteemi suurusest tulenevalt on vaja täienda rakenduste *backend* plokki, kuhu tuleb lisada kõik kuus kasutusel olevat rakendust. Mõned URLid on määratud kindlate rakendusserverite pihta, nende jaoks on eraldi ACLid loodud.

Katkestus Eksamite Infosüsteemi testkeskkonnas koormusjaoturi vahetuseks on lubatud kaks tundi tööpäeva jooksul. Katkestuse käigus vahetatakse välja Apache HAProxy vastu. Vahetuse käigus muutub test.ekk.edu.ee domeeni IP-aadress. Apache'i proksiserver jääb seisvas olekus alles juhuks kui üleminek peaks ebaõnnestuma.

Proksi vahetus testkeskkonna jaoks oli edukas ning järgnevalt on vaja jälgida mälukasutuse koormusgraafikuid. Üheks põhiliseks probleemiks Apache'i proksi rakendamise juures oli selle mälukasutus. Proksiserveril oli eelnevalt 8 GB mälu, mis suure koormuse all kõik ära kasutati, mistõttu tipphetkedel suurendati mälu 16 GB-le.

Arvestades äripoolelt vajaminevat koormustaluvust on koormusjaoturi mälukasutust vaja vähendada.
Tootekeskkonna proksi uuenduseks on kokkulepitud ajakulu üks tööpäev. Kahe nädala jooksul testiti süsteemi ja teostati vastavad seadistusmuudatused, et süsteemi oleks võimalik edukalt kasutada. Uuendus oli edukas, kuid edasiste päevade jooksul ilmnesid mõned probleemid seoses üleslaetavate failide suurustega, sessioonide kestvusajaga ning koormuse jaotuse ülesehitusega, kuid proksi vahetuse tagajärjel langes mälukasutus tunduvalt. Seadistused ei olnud lõplikud, neid muudeti vastavalt ajale, vajadusele ning infosüsteemi kasutuskoormusele.

1.4 Koormustestid peale HAProxy rakendamist

Eelnevalt on Eksamite Infosüsteemile koormusteste tehtud üliharva. Edasist HAProxy rakendamise parandamist ja täiustamist saab testida Apache JMeter tarkvaraga, milles on kirjutatud koormustestid süsteemi näitliku koormuse tekitamiseks. Koormustesti sisuks on kasutaja tegevuse jäljendamine ehk tegevused, mida tavakasutaja võiks süsteemis teha. Koormustestil määratakse kasutajate arv ning koormustestide sisu (tegevusi) ei muudeta erinevate testide ajal, küll aga võimalikke kasutajate arvu.

Koormustestide tulemusest on võimalik välja lugeda tehtud päringute koguse ja palju päringuid sekundis käivitati. Põhiliseks võrdlusarvuks on päringute kestvus millisekundites – keskmine ja maksimaalne. Maksimaalne päringu kestvus ehk reageerimisaeg sisaldab endas infot kui kaua kestis kõige pikem päring ning keskmine reageerimisaeg on kõikide päringute tulemsute keskmine väärtus. Lisaks on võimalik koormustesti tulemustest välja lugeda ebaõnnestunud päringute arv ja protsent kogupäringutest. Ebaõnnestunud päringud tekivad kui rakendus ei saa andmebaasiga ühendust, proksi ei suuda päringut teenindada või rakenduse protsesside arv ületas etteantud limiidi.

Eksamite Infosüsteemis tootekeskkonnas tehti koormusteste kokku 19 korral, testide käigus täiendati proksi seadistust ning arendaja täiendas rakenduse koodi. Koormusteste tehti vahemikus 2018 november kuni 2019 juuli. Koormustestide tulemused, mis tagastasid ebaõnnestunud päringuid oli kokku viis – selles ajavahemikus tehtud testide tulemused on tabelina välja töötud lisades (lisa 10). Kõige mahukam test tehti 16. mail 2019, kui süsteemi suunas saadeti 3000 kasutajat tehes süsteemis natuke

alla poole miljoni päringu, millest 441 (0.09 %) ebaõnnestusid. Süsteemi keskmise päringu reageerimisaeg oli 7 sekundit ning kõige pikem päring võttis aega 10 minutit.

Süsteemi parimaks tulemuseks saab lugeda testi, kui ei toimunud ühtegi ebaõnnestunud päringut ning süsteemi suunati suurim võimalik arv ühendusi, selleks oli 16. mail 2019 sooritus, kui süsteem teenindas korraga 2500 ühendust. Tulemuseks oli keskmise päringu kestvuseks 0.7 sekundit ning kõige pikem päring võttis aega 57 sekundit. Hetke tulemus on süsteemi võimekus, mida tuleb parandada, et süsteem toetaks korraga 4000 üheaegset kasutajat.

2. Andmebaasi jõudluse parandamine

Eksamite Infosüsteemis kasutatakse andmebaasimootorina PostgreSQL tarkvara. Serveris on kokku neli erineva eesmärgiga andmebaasi. Andmebaas eisdb1 hoiab endas infosüsteemi andmeid, eisdb1tunnistus omab infot tunnistuste kohta, eisdb1sess on sessiooniinfo (ja vormide seadistused jmt) ning eisdb1log on infosüsteemi logide andmebaas. Algselt olid kõik andmebaasid ühes serveris koos ja see tekitas päringute töötlemisel piiranguid.

2.1 Andmebaasi töö põhimõte

Andmebaasi seadistuses on vaja määrata mitu ühendust andmebaasiserver korraga suudab teenindada. Ühenduste koguarv on seotud serverile eraldatud protsessorituumade ja mäluga. Päringute teenindamiseks jagatakse ressurss päringute vahel võrdselt, st mida rohkem koguühendusi on lubatud, seda väiksem on kasutatav ressurss ühele päringule. Kui ühenduste piirarv on täis, siis edastatakse rakendusele sellekohane veateade. PostgreSQL tarkvaral puudub võimalus tekitada päringutest ootejärjekorda, seega on selle lahendamiseks vaja kas lisatarkvara andmebaasi või rakendusserveri juurde.

Järjekord moodustatakse päringute saabumise ajast tarkvarasse. Andmebaasi päringu kestvust arvestatakse alates ülesande loomisest kuni eduka vastuse saatmisest rakendusele. Päringud mõjutavad üksteise kestvust olukordades, kus ootejärjekord hakkab kuhjuma. Suur süsteemi koormus toob endaga kaasa ummiku ning iga aeglane päring mõjutab järjekorras olevate päringute sooritusaega, see omakorda mõjutab kasutaja kogemust infosüsteemis

2.2 Andmebaasi koormuse hajutamine

Kerge lahendus andmebaasi jõudluse parandamiseks on serveri ressursside suurendamine, tänu millele on võimalus suurendada sissetulevate ühenduste arvu ja päringute mälu kasutust. Ainukeseks takistuseks on serveri maksumus ja füüsiliselt saadaolevate ressursside kogus. Andmebaasi võimekuse piir on välja tulnud erinevate koormustestide käigus, kui süsteem ei saavuta enam päringute kestvuses ajalisi võitusid, isegi kui kasutatavat ressursi on saadaval. Lisaks päringute teenindamisele

peab baasiserver tegelema ka administratiivse tööga, mis kasutab veel omakorda ressursse. Efektiivse andmebaasi töö tagamiseks tuleb hajutada serveri koormust vähendades ühel ajahetkel sissetulevate ja töödeldavate päringute arvu, sest väiksem ühenduste arv võimaldab suuremat mälukasutust protsessile ning vaba mälu olemasolu on oluline mahukate päringute korral.

Esimene samm päringute vähendamiseks on andmebaaside jagamine erinevatesse andmebaasiserveritesse. Tulenevalt süsteemi ülesehitusest saab eisdb1 kõige suurema koormuse, millele järgnevad logide ja sessioonide andmebaasid. Paigutades kaks vähem nõutud andmebaasi eraldiseisvasse serverisse, saab vähendada põhiandmebaasi serveri koormust ning eisdb1 andmebaasil on võimalik kasutada rohkem protsesse ja seeläbi suurendada efektiivsust. Olemas on ka neljas andmebaas eisdb1tunnistused, mille kasutus on minimaalne, seega see jääb pragu eisdb1 andmebaasiga samasse serverisse. PostgreSQL tarkvara kõige efektiivsema kasutamise saavutab, kui andmebaasi suunas tehakse umbkaudu ainult sada üheaegset päringut [7]. Tegelik infosüsteemi kasutajate arv on kümme korda suurem, mille tõttu tuleb rakendada andmebaasi ees *pool* ehk puul, mis hoiaks ühenduste arvu andmebaasi suunas kindla suuruse peal.

2.2.1 Koormustestid hajutatud andmebaasidega

Eksamite Infosüsteemi ühest suurest andmebaasi serverist tehti kaks serverit – põhibaas sisuga eisdb1 ja eisdb1tunnistus ning logibaas andmebaasidega eisdb1log ja eidb1sess. Andmebaaside ümberliigutamine toimus 12. juulil 2019. Süsteemi töös tehti kolmetunnine katkestus. Andmebaaside liigutamine kõrvalisse serverisse sujus probleemideta. Sama päeva jooksul tehti kolm koormustesti, et hinnata muudatuse mõju päringute kiirusele. Kolmest testist üks oli andmebaasi ja rakendusserveri peenhäälestusseadistuse mõttes edukas ning see käivitati 3000 sooritajaga, mis tegi süsteemis kokku samapalju päringuid nagu 16. mail ebaõnnestunud 3000 sooritajaga test. Testide käivitamisel oli üks erinevus – kolme JMeteri serveri asemel käivitati testid kahest serverist. Tulemused olid tunduvalt paremad, keskmise päringu aeg liikus 7 sekundi pealt 0.4 sekundile ning kõige pikem päring võttis aega 73,3 sekundit võrreldes varasema testi 10 minutiga, ühtegi ebaõnnestunud päringut ei esinenud.

Sama aasta augustis tehti veel neli testi, millest kaks ebaõnnestusid ning ülejäänud kaks said halvema tulemuse. Väljatoomist väärib 4000 üheaegse ühendusega koormustest, mis õnnestus vigadeta. Päringu keskmine reageerimisaeg jäi 2,5 sekundi

juurde ning päringu maksimaalne reageerimisaeg oli 5,2 minutit. Selle testiga tõestati, et süsteem on võimeline teenindama ära 4000 kasutajat ning edasiseks eesmärgiks oli saada reageerimisajad lühemaks. Kõige viimane test tehti 17. detsembril, kui käivitati 3000 sooritajaga testsüsteemi suunas. Kuna vahepeal olid infosüsteemis toimunud arendusest tulenevalt muutused, siis muudeti testide ülesehitust. Uue testiga oli päringute koguarv suurem, päringu keskmise aeg tõusis võrreldes juuliga, kuid päringu maksimaalse kestvus langes kolmekordselt.

2.3 Andmebaasiühenduste puulerid

Kaks kõige levinumat PostgreSQL andmebaasi puuler tarkvara on Pgpool ja PGBouncer [8]. Puuler tarkvara võimaldab lisada juurde funktsionaalsusi andmebaasi ühenduse loomisel, kuid igal tarkvaral on omad plussid ja miinused. Tarkvarade erinevus seisneb asjaolus, et PgBounceri ülesseadmine on otsekohene ja ühenduste loomiseks ei pea seadistama midagi lisaks, kuid Pgpooli kasutamiseks peab seadistusfailis määrama täpsed parameetrid ning tihti tegema lisakatsetusi ja koormusteste, et leida kõige parem seadistus infosüsteemi jaoks [8].

Infosüsteemide puhul on tavapärane puul tarkvara rakendamine juba rakenduse tasemel. Sama on varasemalt tehtud Eksamite Infosüsteemi puhul, kuid kuna infosüsteemis on kuus rakendusserverit ja serverid omavahel ei koordineeri päringute tegemist andmebaasi, siis kujunes varasem lahendus liiga ebaefektiivseks. Rakendusserverisse ühenduste järjekorra vmt halduse tekitamine vähendaks rakendusserveri efektiivsust veelgi. Rakendusserveri ülesanne on klientide päringute töötlemine, seega oleks otstarbekas andmebaasimootori ühenduste ette rakendada puuler lisatarkvara.

2.3.1 Puul tarkvara valimine

Võimalus on rakendada mõlemat Pgpool ja PgBouncer tarkvara koos kuid see toob endaga kaasa erinevaid probleeme. Esiteks suureneb süsteemi ülesehituse keerukus ja komponentide arv, seega võimaliku rakenduse vea tuvastamine läheb keerulisemaks ja ajakulukamaks. Kuna lahenduse kasutamine tõstab keerukust ja testimine on ajakulukas, siis otsustatakse testida neid eraldi. Samuti langeb päringu töötlemise kiirus, kuna päring peab läbima mitmeid servereid ja tarkvarasid. Edaspidi testitakse

tarkvarasid eraldi ja tehakse samasuguseid koormusteste, et hinnata kas puuler tarkvarast on kasu EISi jõudluse tõstmiseks.

PgBounceri ideaalne rakendamine tuleb kasuks siis, kui mitmed erinevad rakendused või nende osad võtavad ühendust andmebaasiga [8]. EISis võtab andmebaasidega ühendust kuus rakendusserverit, millest igaühel on veel omakorda kolm kasutajat (üks iga andmebaasi jaoks). Eksamite Infosüsteemi PostgreSQL baasi kasutuse kirjelduse järgi sobiks PgBouncer täitma puuleri rolli. Pgpool võetakse tavaliselt kasutusele tänu oma headele lisafunktsioonidele, mis võivad võtta algul aega, et saaksid korralikult üles seatud, kuid võivad hiljem palju jõudlust infosüsteemile juurde anda. Lisafunktsioonid on just need, mis räägivad Pgpooli valiku kasuks, kuna sama aegselt saab vähendada nii koormust andmebaasi suunas ning soovi korral teha tuleviks päringute optimeerimist ja rakendada lisa funktsioone. Otsustavaks faktoriks on aeg, kuna uue süsteemi lüli rakendamine peab olema võimalikult väikese ajakuluga ja efektiivne ning Pgpool seda hetkeolukorras ei ole, tuleb testida PgBouncer tarkvara rakendamist [8][9].

2.4 Paigaldamine

EENetis on valdav enamus servereid paigaldatud Arch Linux operatsioonisüsteemile, mis ehitatakse kokku ServeriVabrikus, mis on BASH ja Ruby skriptide kogum, mis võimaldab kettale paigaldada eelseadistatud operatsioonisüsteeme, omades nii Preseedi kui konfiguratsioonihaldustööristade (näiteks Puppet) funktsionaalsusi. Seda kasutatakse virtuaalmasinate loomiseks ja versioonihalduseks. ServeriVabrikut hoitakse EENeti Gitlabis, tagades töötajatele ligipääsu loodud serverite seadistusele. Vabrik lubab uue virtuaalmasina laadimiseks vajalikke ressursse (tuumasid, pakke, konstruktoreid ja spetsiifilisi konfiguratsioonifaile) hoida ühes asukohas, kasutades efektiivselt andmemahtu ja lihtsustades virtuaalmasinate loomist ning haldust [10][1].

Iga serveri jaoks on Buildfile (e.k. konstruktor), milles defineeritakse sellele omistatavad väärtused. Konstruktor käivitatakse läbi skripti, mis loob Ehitaja kettale serveri nimelise kausta kuupäevaga, pakib sellesse lahti operatsioonisüsteemi ja konstruktori alusel teeb konfiguratsioonimuudatused, paigaldab pakid, loob vajalikud sümbolnimed. KVM virtualiseerimist kasutav VM sooritab alglaadimise, kasutades PXEd ja haagib

[1] * Märkus 1. Tegemist on kollektiivselt loodud KKK rubriigi tekstiga, lõputöö autor on üks teksti loojatest.

konstruktoriga tekitatud kausta NFS jaona külge, laadides sealt juurfailisüsteemi. Käivitamiseks vajalikud seaded saab VM PXELinux konfiguratsioonifailist, mis on VMi virtuaalse võrguseadme MAC aadressi nimeline. Fail sisaldab lisaks kerneli versioonile ka parameetreid nagu külgehaagitavad kettad ja IP aadressid. VM on alglaadimise sooritanud, võimaldades külgehaagitud ketaste initsialiseerimist, näiteks selle partitsioneerimist, failisüsteemi loomist ja failide kopeerimist [10][1].

Eksamite Infosüsteemi puuler serveri konstruktor on lühikese ülesehitusega, kuna serveris töötab ainult üks põhirakendus, milleks on PgBouncer. Serveri ülesehituse säilitamiseks paigaldatakse serverisse varunduse pakk, mis on eelnevalt seadistatud tegema kirjutavast kettast varukoopia igal öösel. Server hakkab paiknema privaatses võrgus, mistõttu ei vaja see avalikku liidest ega tulemüüri. Serverisse ligipääsuks lisatakse eelseadistatud administraatorite kontod ning arendaja ligipääs läbi VPN lahenduse. Lisaks tehakse operatsioonisüsteemi ülesehituses muutatus, et lubada süsteemil kasuda serveriga ühendust võtnud klientide *port*e uuesti iga sekundi tagant. Kui rakendus võtab ühendust puul serveriga, avatakse suhtluseks eraldi port ning kui suhtlus lõppeb jääb port veel tegelikult mitteaktiivsesse olekusse (kuni minutiks), kuni nimetatud port vabastatakse ja lubatakse tagasi ringlusesse. Ühe päringu jaoks avatakse kaks porti, üks ühendus on rakenduse ja puuli vahet ning teine on puuli ja andmebaasi vahel.

2.5 Testsüsteemi ehitamine

PgBounceri testimiseks tuleb ehitada testsüsteem, mis peaks ideaalis koosnema vähemalt kahest serverist, üks puhtalt PostgreSQL baasimoori käsutusse ning teine server PgBounceri jaoks Funktsioonide järgi eraldatud teenusserveritel on lihtsam jälgida ressursside kasutust. Kahe serveri rakendamine tagab võimaluse ka rakendused tagasi lülitada nn otseühendusega andmebaasi külge. Lisaks saab jätta mõlemad programmid kasutama PostgreSQLi vaikimisi porti 5432 ja seadistamisel ei pea hakkama jagama serveri ressursse kahe eri tarkvara vahel. Paigutades andmebaasi ja puuli tarkvarad ühte serverisse on vaja jälgida, et need ei kasutaks samu porte, täiendada mõlema tarkvara konfiguratsioonifaile jm.

Peale serverite ehitamist ServeriVabrikus on võimalik teha tarkvarade seadistamised. PostgreSQL masinas peab käivitama andmebaasi initsialiseerimise käsu, mis määrab kuhu kausta baasimootor hakkab andmebaaside faile salvestama. Paigalduskausta

tekivad andmebaasi seadistusfailid, milles tuleb teha muudatusi, et optimeerida andmebaasi tööd ja lubada ligi ühendusi teistest serveritest. PostgreSQL hakkab ühendusi ootama kõikidel võimalikel liidestel, milleks on server ise ning külge lisatud sisevõrgu liides, mis ei ole ühendatud avaliku internetiga. Peale seadistuste sisseviimist tuleb taaskäivitada PostgreSQL teenus serveris.

2.6 Testsüsteemi koormustestid

PostgreSQL koormuse testimiseks saab kasutada Pgbench tarkvara, mis tuleb kaasa andmebaasi paki paigaldamisel. Koormustestide tegemiseks võib kasutada eelnevalt loodud andmebaasi või lasta Pgbench tarkvaral luua testide tegemiseks uus andmebaas koos sisuga. Kõige kergem on lasta Pgbench tarkvaral luua uus andmebaas, kuna Eksamite Infosüsteemi andmebaasi kasutamiseks tuleks kõigepealt tarkvara testserverisse taastada. Pgbench tarkvara poolt loodav andmebaas on 16 MB suur, mis on liiga väike ning ei ole ligilähedal EISi andmebaasi suurusega. Andmebaasi loomisel on võimalik koormustestide tarkvarale öelda kui mitu korda suurendada algselt 16 MB andmebaasi. Eksamite Infosüsteemi ametlik põhiline andmebaas eisdb1 on 178 GB suur, mistõttu tuleb kasutada testandmebaasi loomisel kordajat 11125. Sellega tagatakse enam-vähem sarnane maht.

Optimaalseks koormuseks paigaldatakse Pgbench tarkvara eraldi serverisse andmebaasist. PostgreSQL initsialiseerimist ei pea tegema. Esimesed koormustestid tuleb teha ilma puul tarkvarata, et leida andmebaasi kõige efektiivsem seadistus. Eksamite Infosüsteemi andmebaasi masinal on kasutada 48 GB mälu ja 32 protsessori tuuma, serveris hoitakse kahte andmebaasi eisdb1 ja eidb1tunnistused, millest viimane ei saa erilist koormust. Koormustesti käivitamisel tuleb määrata sihtserver, kuna andmebaas on eraldiseisvas masinas. Lisaks saab määrata mitmeid erinevaid väärtusi koormustesti läbiviimiseks – loodavate ühenduste arv, palju protsesse võib luua, ühe ühenduse poolt tehtavate transaktsioonide arv.

Efektiivse PostgreSQL seadistuse leidmises tuleb testida andmebaasi erinevate seadistusväärtustega. Põhiliseks muutujaks on andmebaasi maksimaalsete ühenduste arv, mis mõjutab omakorda work_mem väärtust ehk mälukasutamist protsessi kohta. Üks koormustest võiks kesta 15 kuni 30 minutit, mis võimaldab andmebaasil tagastada stabiilse tulemuse tänu pikaajalisele tööle ja selle aja põhjal on võimalik ka koormusgraafikutelt mingi järeldus teha. Põhjaliku ülevaate saamiseks tuleb teha

koormustestid iga lisanduva 10 ühenduse kohta kuni 1000 üheaegse ühenduseni. Kui testi tulemused näitavad, et süsteemil on veel ruumi tagastada paremaid tulemusi kui tuhat ühendust, siis tuleb suurendada koormustesti piire. Maksimaalne ühenduste arv 1000 tähendab testi sooritamist 100 korral, mille käsitsi tegemine on ajaliselt kulukas ja ebaefektiive, seepärast tuleb koormustestimine automatiseerida.

2.6.1 Testimise automatiseerimine

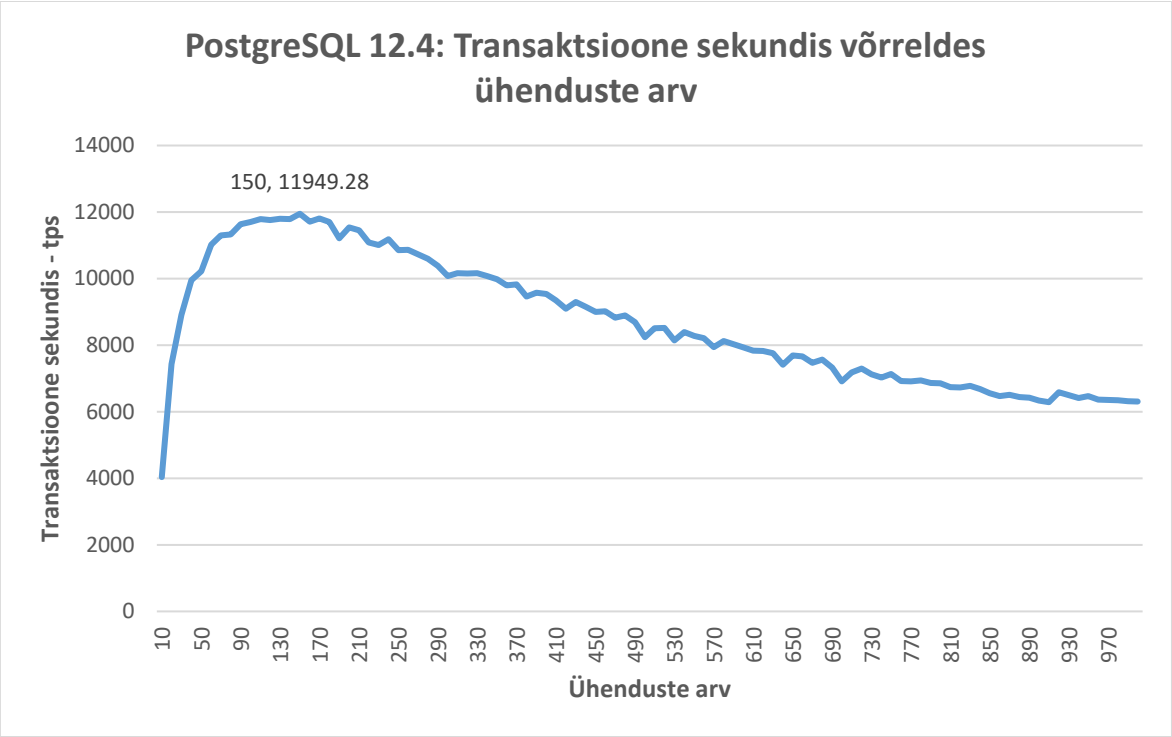
Skript tuleb paigaldada koormustestide serverisse tagades võimaluse muuta skripti sihtserverit vastavalt kas koos või ilma puul tarkvarata. Skriptile tuleb määrata piirid, et vältida protsessi lõpmatut töötamist. Piiriks on maksimaalsete ühenduste arv ehk 1000 ühendust, algseadena on see 10 ning suureneb iga kui-tsükliga kümne võrra. Iga tsükli käigus tehakse SSH andmebaasi serverisse kus käivitatakse Python programmeerimiskeeles kirjutatud PGTune skript, mis kombineerib siestatava info põhjal optimaalse PostgreSQL seadistuse. Skripti sisendiks tuleb anda kaasa soovitud ühenduste arvu. Serveri ressursid, mälu kogus ja protsessori tuumade arvu, tuvastab skript iseseisvalt.

PGTune tarkvara loojaks on Greg Smith, kelle versioon põhines Python programmeerimisele, kuid skript on tänaseks aegunud ning ei toeta enam uuemaid PostgreSQL versioone. Seadistustarkvara põhja on kasutatud paljude uute PGTune järglaste loomisel, paljud nendest kannavad sama nime. Koormustestide skripti käsutuses on üks PGTune seadistustarkvara järglastest. Vastav transaktsioonide arv arvutatakse iga uue ühenduvate klientide arvu muutumisel – andmebaasi tehakse kokku vähemalt 30 000 ühendust. Koormustesti tulemused kirjutatakse ekraanile, mis veel omakorda dubleeritakse serveris olevale andmeladustuskettale hilisemaks analüüsimiseks.

2.6.2 PostgreSQL koormustest

Koormustesti tulemus põhineb kahe arvu võrdlemisel, kui palju transaktsioone suudab andmebaas teha määratud ühenduste korral. Testi tulemused on esitatud graafikul 2.1. Tulemuste põhjal saab öelda, et andmebaas on kõige efektiivsem 150 ühenduse juures tehes 11 949 transaktsiooni sekundis. Peale parima tulemuse saavutamist toimub ühtlane langus. Parim tulemus ei tähenda seda, et sellise lõppseadistusega tehakse andmebaas, kuna tihtipeale loeb süsteemi sisene ülesehitus ja infosüsteemi ametlikud koormustestid, et milliseks kujuneb andmebaasi lõplik seadistus. Tehtud andmebaasi

koormusteste saab võrrelda tulevaste infosüsteemi koormustestide tulemusega, mille põhjal on võimalus luua seoseid ja teha edasised optimeerimisi.



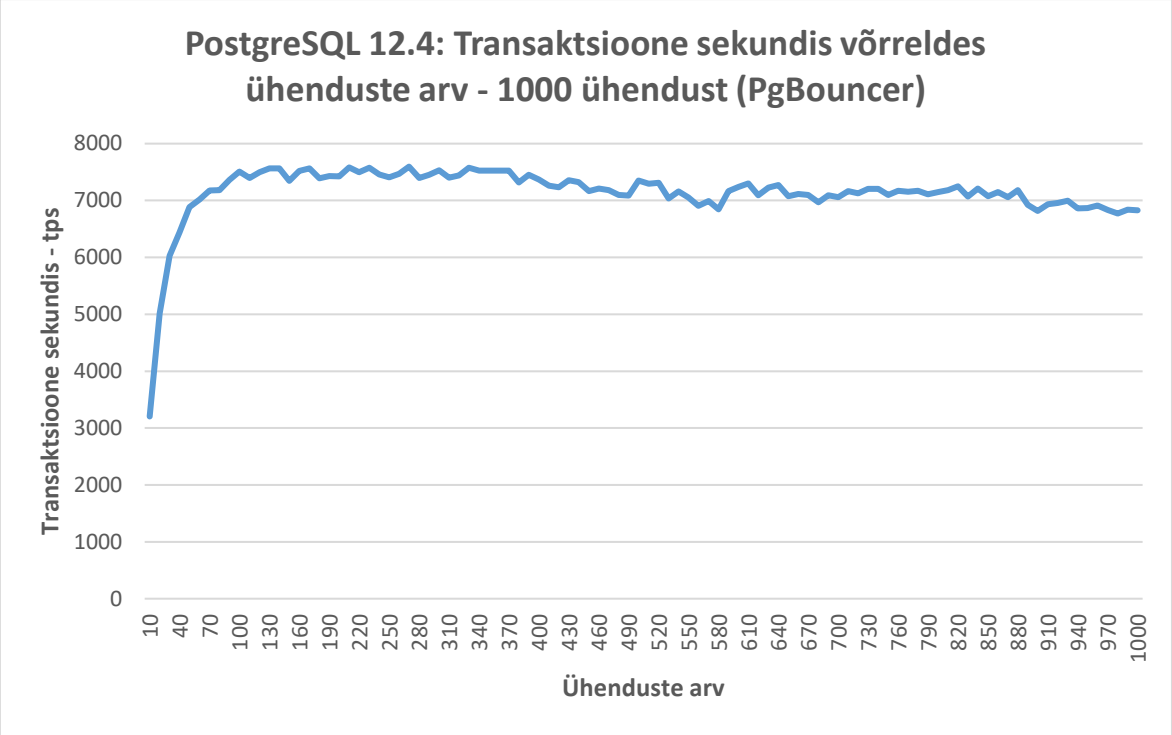
Graafik 2.1. PostgreSQL koormustest

2.6.3 PostgreSQL ja PgBounceriga koostöö koormustest

Järgmiseks etapiks on rakendada andmebaasi ette PgBouncer ja testida kuidas seadistada ja koormus jaguneb andmebaasile koos puul tarkvaraga. Eesmärgiks on suunata puuli suunas tuhat kasutajaid ja võrrelda, kuidas saab ülesandega hakkama andmebaas iseseisvalt ja koos puul tarkvaraga. Sarnaselt eelmise korraga teeb skript kokku saja erineva ülesehitusega testi, lisaks andmebaasi muutmisele tuleb juurde lisada puul tarkvara sees toimuvad tegevused – puuli seadistuses andmebaasi suunas olevate parameetrite muutmine, et need oleksid sarnased andmebaasi maksimaalsete ühenduste arvuga ja puul tarkvara taaskäivitamine ning päringud tuleb saata andmebaasi asemel nüüd puul serveri suunas ehk tuleb vahetada IP-aadress. Testi vahemik jääb samaks, aga puuli serveri suunas tehakse iga seadistuse korral 1000 ühendust.

Koostöö koormustesti tulemused on graafikul 2.2. Tulemuste põhjal saab öelda, et olenemata andmebaasi seadistusest suudab PgBouncer hoida ühtlast transaktsioonide arvu sekundis. Andmebaasi seadistused, mis põhinevad ühenduste arvu vahemikust 10

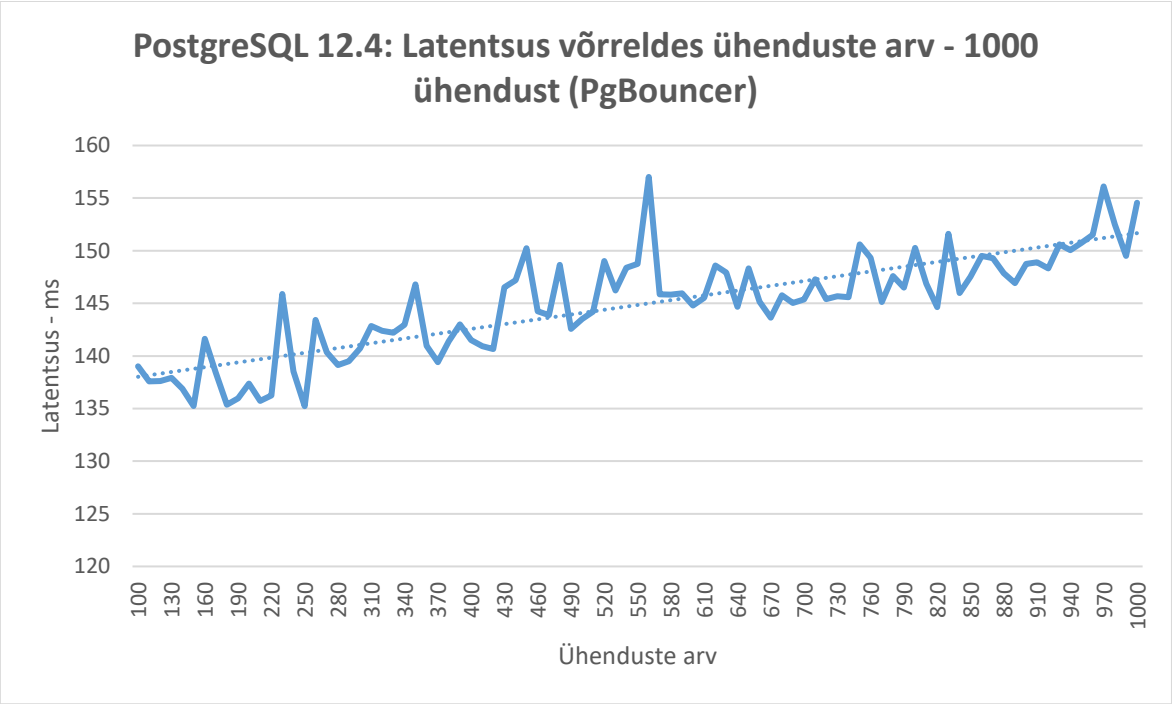
kuni 750 said parema tulema iseseisvalt ilma puul tarkvara abiga, kuid peale 750 ühenduse arvu, suutis puul tarkvara tõsta andmebaasi transaktsioonide töötlemise kogust sekundis. Testi tulemuste põhjal saab öelda, et toodud riistvaralise ülesehituse, andmebaasi suuruse ja infosüsteemi kasutajate arvu poolest tasuks puuli tarkvara ennast kindalt ära. Eksamite Infosüsteemis on oodata tulevikus samaaegsete kasutajate arvu 2000 ja 4000 vahel, misläbi tuleb andmebaasi ülesehitust testida suuremate ühenduste arvuga, et teha kindlaks, mis on süsteemi maksimaalne piir.



Graafik 2.2. PostgreSQL ja PgBounceriga koostöö koormustest

Andmebaasi seadistuse valmine on hetkel keerukam ning ei ole ühte kindalt ülesehitust, mis oleks kindalt teistest parem. Viimase testi põhjal saab öelda, et andmebaasi võib seadistada töötama vahemikus 100 kuni 370 ühendust, kuna see vahemik andis kõige parema tulemuse – umbes 7500 transaktsiooni sekundis. Kõige kindlam on valida väiksema ühenduste arvuga ülesehitus, kuna see võimaldab jätta protsessile suurema koguse mälu. Kõige parema tulemuse andis 270 ühendusega ülesehitus, mis saab olema järgmise koormustesti aluseks. Võimalus oleks veel arvestada latentsuse mõju, kuid see väärtus suureneb pidevalt kui andmebaas on seadistatud sajast suurema maksimaalse üheaegsete ühenduse teenindamiseks [11]. Kindalt latentsuse väärtust ei saa valida, kuna tulemus kõigub suuresti iga testiga. Segavates teguriteks on kõikuv võrguliikluse olukord testi tegemise hetkel ning virtualiseerimisklastris teiste virtuaalmasinate

koormus. Latentsuse ja ühenduse arvu suhe viimaste testi tulemused on kajastatud graafikul 2.3.



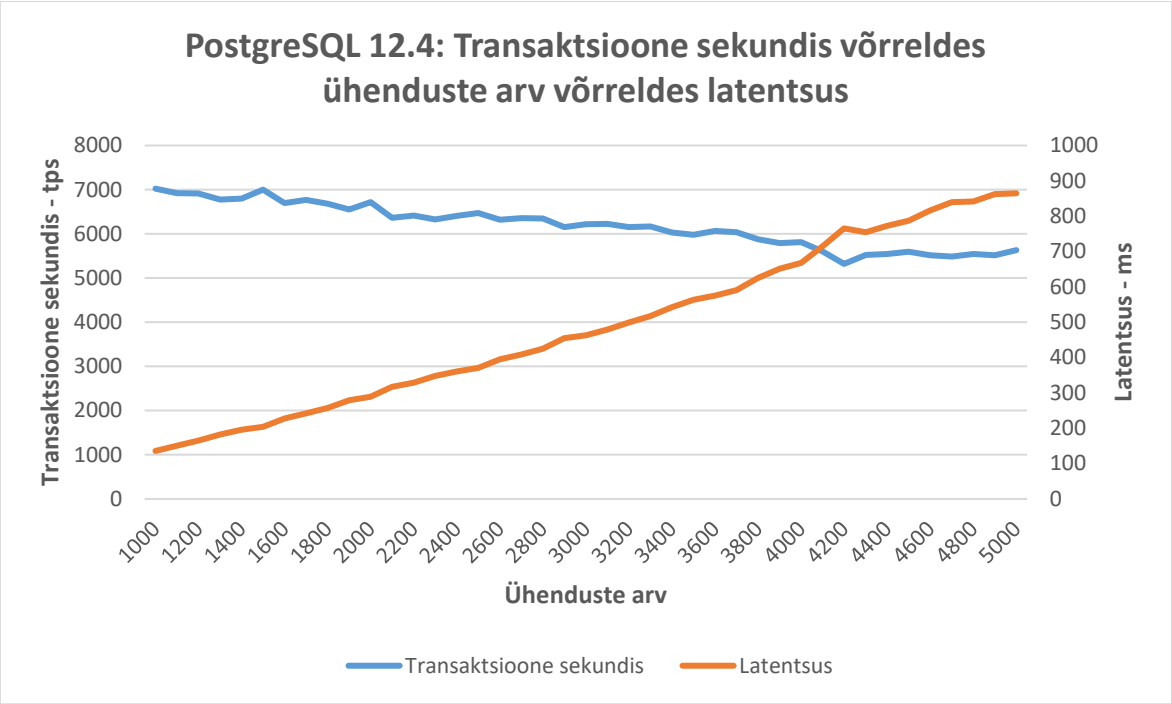
Graafik 2.3. PostgreSQL ja PgBouncer koostöö koormustesti latentsus

2.6.4 Lõplik tarkvarade koostöö tulemus

Maksimaalse süsteemi võimekuse leidmiseks tuleb kirjutada sarnane skript esimesega. Erinevusteks on ühenduste arvu vahemik, milleks on 1000 kuni 5000 ja testide vaheliseks sammuks on 100 ehk teste tehakse iga lisanduva 100 ühenduse kohta. Maksimaalse ühenduste arvu määramine 5000 tuleneb sellest, et hetkel on eesmärgiks toetada EISis 4000 üheaegset külastajat ning selle mõttega sai juurde lisatud 1000 ekstra ühendust. Skript ei pea andmebaasi seadistama, kuna eelmise testi põhjal otsustati, et andmebaas ja puul tarkvara seadistatakse käsitsi 270 üheaegse ühenduse teenindamiseks. Testi eesmärgiks on näha, kui palju langeb transaktsioonide arv sekundis ning kui palju suureneb päringute latentsus.

Tulemused on kajastatud graafikul 2.4. Saadud info põhjab saab väita ootuspäraselt, et transaktsioonide arv ja latentsus muutuvad halvemaks ühenduste arvu kasvuga, kuna iga testiga suurenes PgBounceri ootejärjekorras olevate päringute kogus. Lõpliku ülevaate annaavad JMeter koormustestid, mis aitavad kaasa andebaasi ja puul tarkvara

täpse seadistuse leidmiseks, mis sobib Eksamite Infosüsteemi ülesehitusse kõige paremini.



Graafik 2.4. Lõplik tarkvarade koostöö tulemus

2.7 PgBounceri rakendamine EISis

EISis süsteemis on kokku 18 erinevat andmebaasikasutajat, seega on vaja puul tarkvara seadistada natukene erinevalt võrreldes eelnevate testidega. Kõik kasutajad peab lisama puul serveri seadistuses määratud faili – iga kasutaja on omaette real koos parooliga. Eesmärgiks on lubada igast rakendusest kindel arv ühendusi andmebaasi suunas, tagades, et üks rakendus ei hõivaks kõiki andmebaasi ühendusi enda kasutusse. Tulenevalt HAProxy ülesehitusest toimub koormuse jagamine rakendusserverite nimekirja alusel ehk ühendused suunatakse serveritesse korda mööda. Võib tekkida olukordi, kus kasutaja teeb nõudlikumaid päringuid ühes rakenduses ning selle rakenduse andmebaasi ühendused hakkavad kuhjuma, kuid sellisele probleemile ei ole hetkel lahendust.

Andmebaasi testide põhjal sai otsustatud, et PostgreSQL seadistus tuleb teha 270 üheaegse ühenduse teenindamiseks. Võimalus on kasutada kõik ühendused andmebaasi suunas ära, kuid see ei jätaks ruumi olukordadeks kui ühenduste arv läheb üle lubatud

piiri. Testimise käigus õnnestus tekitada olukord, kus PgBouncer lubas andmebaasi suunas rohkem ühendusi kui lubatud, millest tulenevalt hakkasid tekkima ebaõnnestuvad päringud. Jagades 270 ühendust kuue rakenduse vahel saame 45 ühendust ning eemaldades ühe ühenduse rakenduse kohta varusse on tulemuses 44 ühendust, mis tuleb jagada kahe EISi andmebaasi vahel, milleks on eisdb1 ja eisdb1tunnistus. Arendaja tagasiside põhja tehakse tunnistuste baasi ühendusi väga minimaalselt, millest tulenevalt on võimalik igal rakendusel teha kaks ühendust tunnistuse baasi suunas. Lõpptulemusena on põhibaasi suunas võimalik avada igal rakendusel 42 ühendust, jättes PostgreSQL serveril 6 ühendust varusse.

Logiandmebaas, mis hoiab endas baase eisdb1sess ja eidb1log peab olema puuli poolest sarnase ülesehitusega nagu esimene baasiserver. Sarnane ülesehitus on tingitud baasi-kasutajate ülesehitusest. Sama kasutaja, mis teeb päringuid eisdb1 kirjutab sessioonide andmebaasi infosüsteemi kasutajal hetkel pooleli olevate asjade ülesehituse. Tulenevalt logi andmebaasi ülesehitusest ja vajadusest infosüsteemi poolt ei pea rakendama nimetatud andmebaasi kõike efektiivsemat ülesehitust. Logi serveris asub kaks andmebaasi ning mõlemale on vaja sarnast ühenduste arvu nagu põhi andmebaasi ehk siis 42 ühendust andmebaasi kohta annab tulemuseks 504 ning jättes sarnaselt 6 ühendust varusse saab PostgreSQL seadistuse määrata 510 ühendusele. PgBouncer andmebaasi ühendused on illustreerivalt esitatud koodilõigus 2.1.

```
[databases]
eisdb1 = host=10.64.0.2 port=5432 pool_size=42 max_db_connections=252
eisdb1tunnistus = host=10.64.0.2 port=5432 pool_size=2 max_db_connections=12

eisdb1sess = host=10.64.0.3 port=5432 pool_size=42 max_db_connections=252
eisdb1log = host=10.64.0.3 port=5432 pool_size=42 max_db_connections=252
```

Koodilõik 2.1. PgBounceris rakendatud puul ülesehitus

Eksamite Infosüsteemi ümberseadistamine otse ühenduselt andmebaasiga puul serveri suunas toimus 18. detsembril 2019. Põhilise ümberlülituse tegevuse oli IP-aadressi muutus rakenduses, mille muudatused kandis koodi sisse arendaja. Andmebaasi seadistusete poole pealt pidi kindlaks tegema, et lubatud ühenduste vahemikus on puul serveri IP-aadress ning seadistama põhi- ja logiandmebaasi ülesehituse vastavalt plaanitud ülesehitusele kasutades PGTune seadistust. Väljavahetamine õnnestus koos mõningate tagasilöökidega - mõne rakenduse seadistus jäi korrektselt muutmata ning puul tarkvaras oli vaja peenhäälestusega seadistada päringute optimeerimist. Lõplik Eksamite Infosüsteemi PgBounceri seadistusfaili on lisades.

2.8 Koormustestid peale PgBouncer rakendamist

Peale PgBouncer rakendamist on Eksamite Infosüsteemis tehtud kokku 34 koormustesti, millest 18 on tagastanud ebaõnnestunud päringuid. Koormustest tehti ajavahemikus 19. detsember 2019 kuni 18. september 2020. Võrreldes hajutatud andmebaasi ülesehituse korral tehtud koormustest, kus suunati süsteemi suunas 4000 üheaegset kasutajat on tulemus paranenud mitmekordselt – arvestama peab süsteemi muutuste ja koormustesti skripti täiendustega. Kahekordselt on vähenenud reageerimise keskmise päringu kestvus kahekordselt ning pikim päring vähenes 45 korda – 5 minutilt 7 sekundile. Hetkel on süsteemi maksimaalseks koormustaluvuseks 4998 ühendust, test tehti 21. jaanuaril – päringu keskmine reageerimis aeg oli 2 sekundit ja maksimaalne päring kestis 45 sekundit.

3. Infosüsteemi vastupidavus reaalsuses

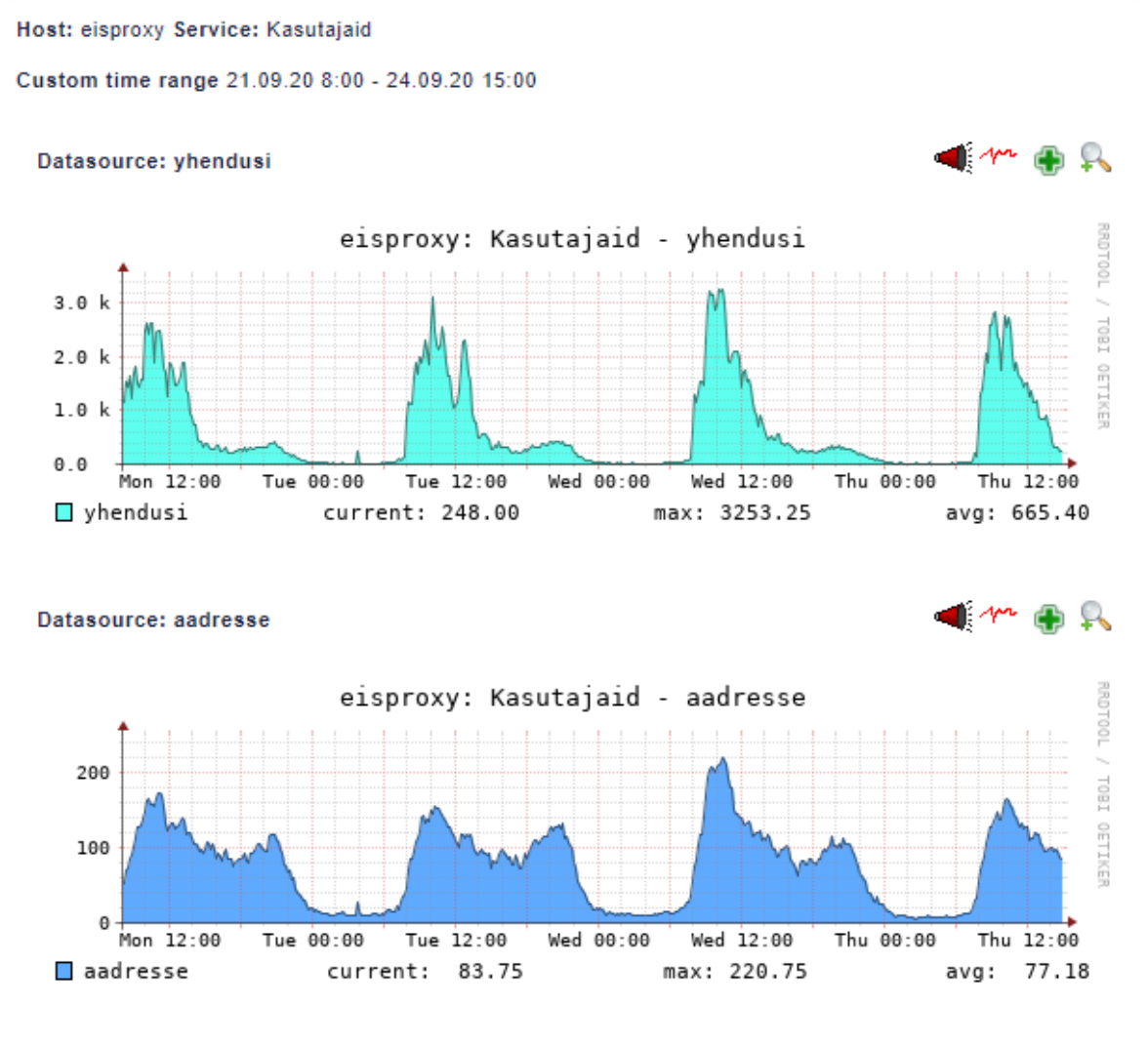
Infosüsteemi kindla vastupidavuse üheks mõõdupunktiks on vastupidavus reaalsele koormusele. Eksamite Infosüsteemis on selleks tasemetööde või eksamite sooritamine, mis koosnevad kas automaatsest kontrollist või kõrvalise kasutaja tagasisidest. Lisaks osade tulemuste puhul tehakse arvutusi tulemuste piiride loomiseks või korrigeerimiseks, mis mõjutavad süsteemi tööd, kuna arvutused on mahukad. EISis tehti septembris kokku kahes õppeaines tasemetööd – loodusõpetus ja matemaatika. Tasemetöö sooritajateks olid 7.klassi õpilased. Kokkuvõtted tehakse koormusgraafikute põhjal, mis on tehtud CheckMK tarkvaraga, mis kogub andmeid süsteemi protsesside kasutuse kohta ja kuvab saadud informatsiooni graafikutel. Süsteemi tehnilised andmed on kajastatud tabelis 3.1. Rakendusi on kokku kuus, igaüks tabelis 1 toodud konfiguratsiooniga.

Tabel 3.1. EISi serverite ressursid

Serveri nimi	*Mälumaht (GB)*	*Protsessori tuumade arv*
eisproxy	8	8
eisrakendus	16	8
eispgbouncer	6	4
eisdb	48	32
eislogdb	24	8

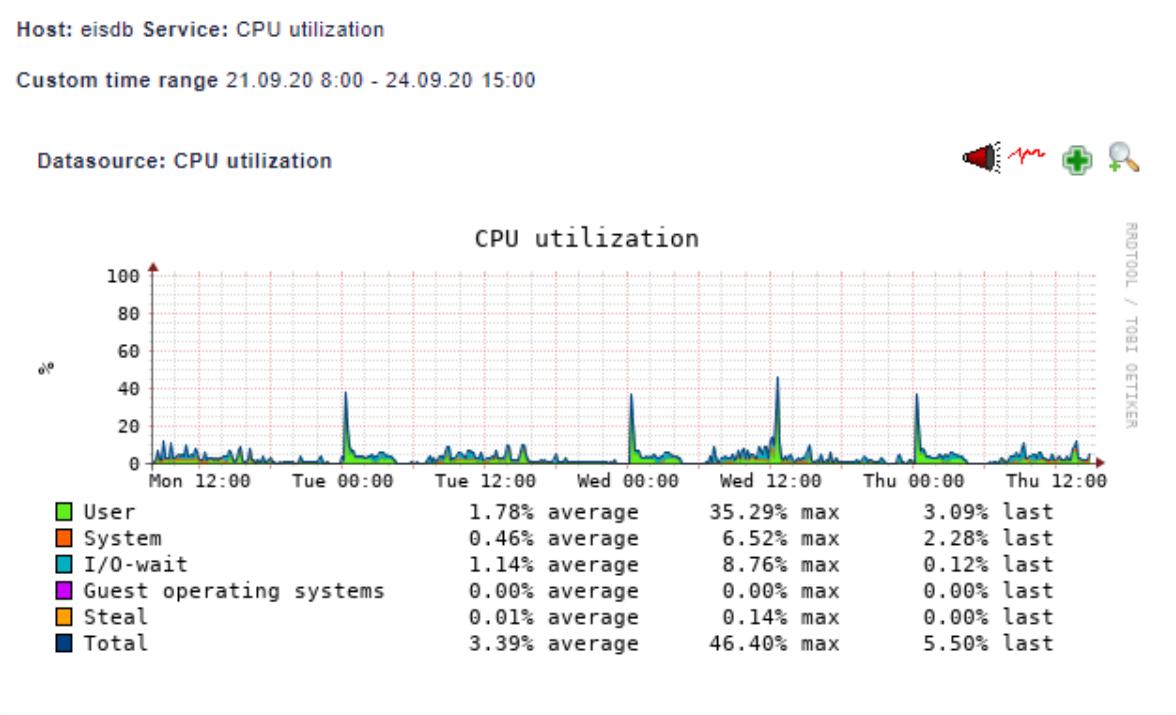
3.1 Loodusõpetuse ja matemaatika tasemetööd

Loodusõpetuse tasemetöö toimus ajavahemikul 21. kuni 24. september – teste tehti nimetatud päevadel kella 8:00 ja 15:00 ajavahemikul. Test oli üheosaline ehk kõik tegevused oli võimalik teha samal päeval õpilastel arvutiklassis. Tasemetöös oli kokku 8 süsteemipoolt hinnatavat ülesannet ja 13 loomingulist ülesannet, mida pidi hindama kooli poolt määratud kontrollija. Loodusõpetuse testi perioodi käigus saavutati üheaegsete maksimaalsete ühenduste arvuna 3253 ühendust ning süsteemiga oli üheaegselt ühendatud 220 erinevat IP-aadressi. Tasemetöö kasutajate arvu visuaalne ülesehitus koos ajateljega on leitav jooniselt 3.1.

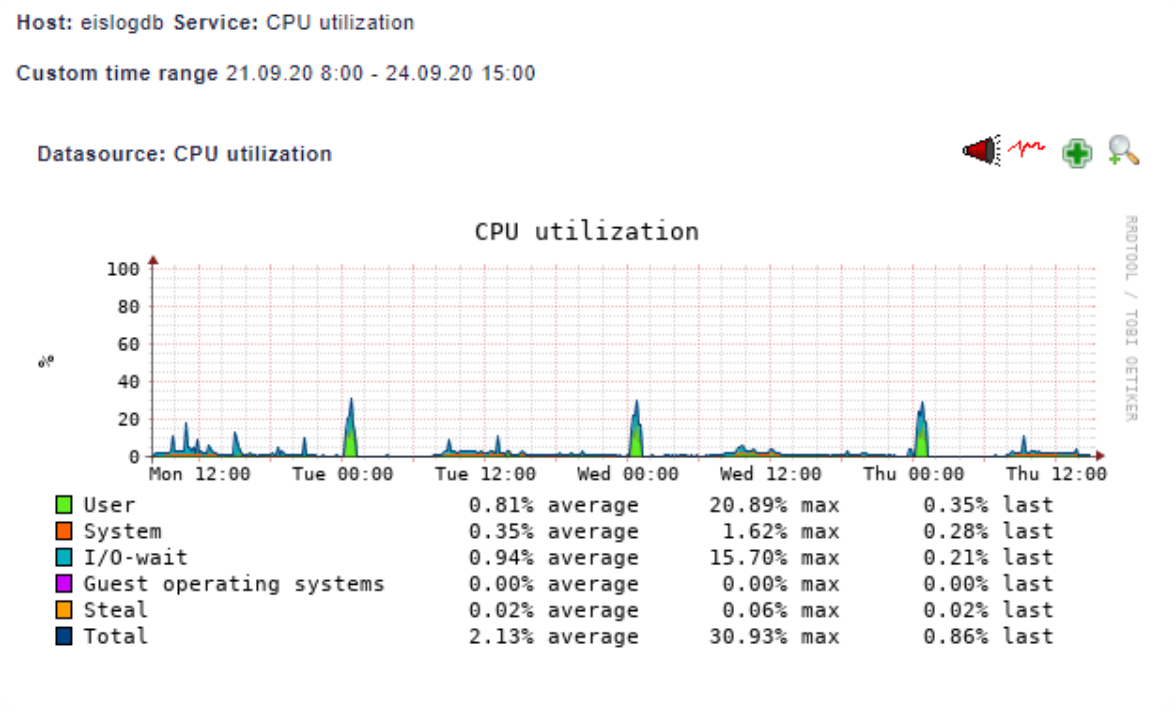


Joonis 3.1. EIS kasutajad - loodusõpetuse tasemetöö

Andmebaaside monitooringu põhjal saab öelda, et põhibaasi maksimaalne koormus oli 42,02% ning logibaasil 17,68%. Üldine andmebaaside kasutusprotsent oli 2% ja 3% vahel. Põhi andmebaasi töö on joonisel 3.2. ja logibaasi kasutus on joonisel 3.3. Mõlema andmebaasi graafikul on näha kesköö andmebaasi koormuse tõus seoses andmebaasi varundamisega. Tulenevalt jälgimistarkvara eripärast ei ole võimalik hinnata andmebaasi serverite mälukasutust graaflikudelt.



Joonis 3.2. EIS põhibaasi protsessori kasutus - loodusõpetuse tasemetöö

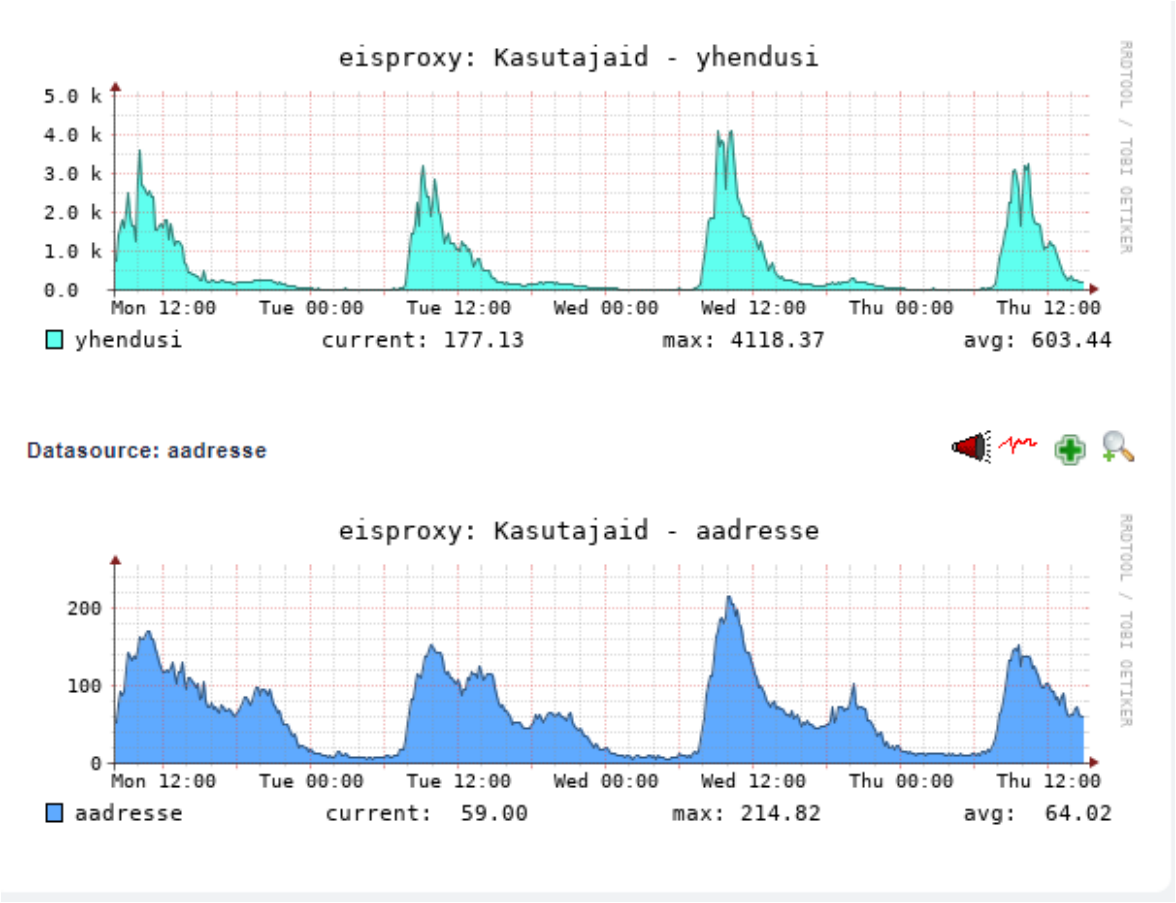


Joonis 3.3. EIS logibaasi protsessori kasutus - loodusõpetuse tasemetöö

Andmebaasi koormusjaoturi protsessor oli maksimaalselt kasutuses 13,58% ning mälu-kasutus oli minimaalne. Rakenduste protsessori kasutused jagunesid kahte gruppi. Kolm esimest rakendust said portsessori maksimaalseks koormuseks kindlal hetkel üle 60% ning ülejäänud kolm olid alla 10 %. Kolm rakendust, mis ületasid 60 % piirid vaheldusid

iga päev. Tulenevalt protsessori kasutusest jagunesid rakendused mälu kasutuses sarnaselt kahte gruppi – esimesed kolm kasutasid 6GB mälu ning ülejäänud 4,5 GB. Viimaseks süsteemi komponendiks on HAProxy server, mis kasutas tasemetöö perioodil 3% oma protsessori koguvõimsusest ning 1,8 GB mälu, millest tulenevalt saab väita, et HAProxy on väga efektiivne.

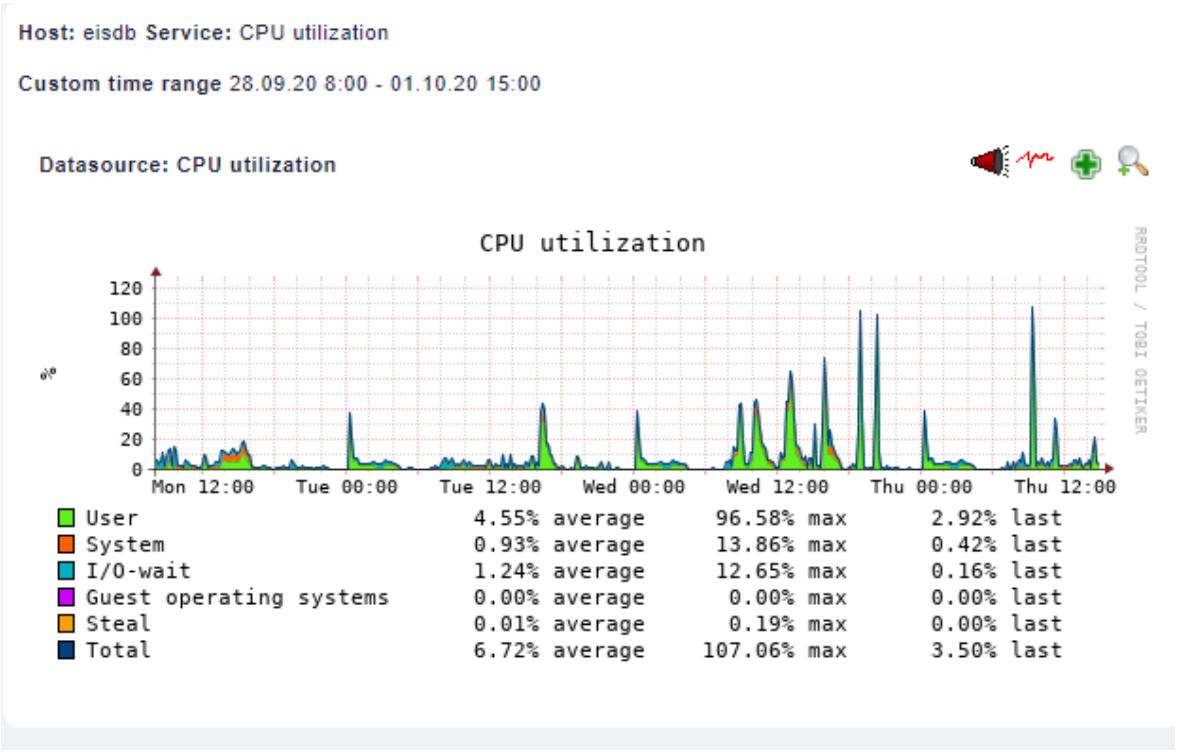
Matemaatika tasemetöö toimus ajavahemikul 28. september kuni 1. oktoober. Sarnaselt loodusõpetusega sooritati testi samal ajavahemikul. Matemaatika tasemetöö läbimine oli tunduvalt nõudlikum. Tasemetöö üheaegsete ühenduste arvuks kujunes 4118 ning IP-aadresside arv oli seekord väiksem – 214. Matemaatika tasemetöö ühenduste jagunemist on kujutatud joonisel 3.4.



Joonis 3.4. EISi kasutajad - matemaatika tasemetöö

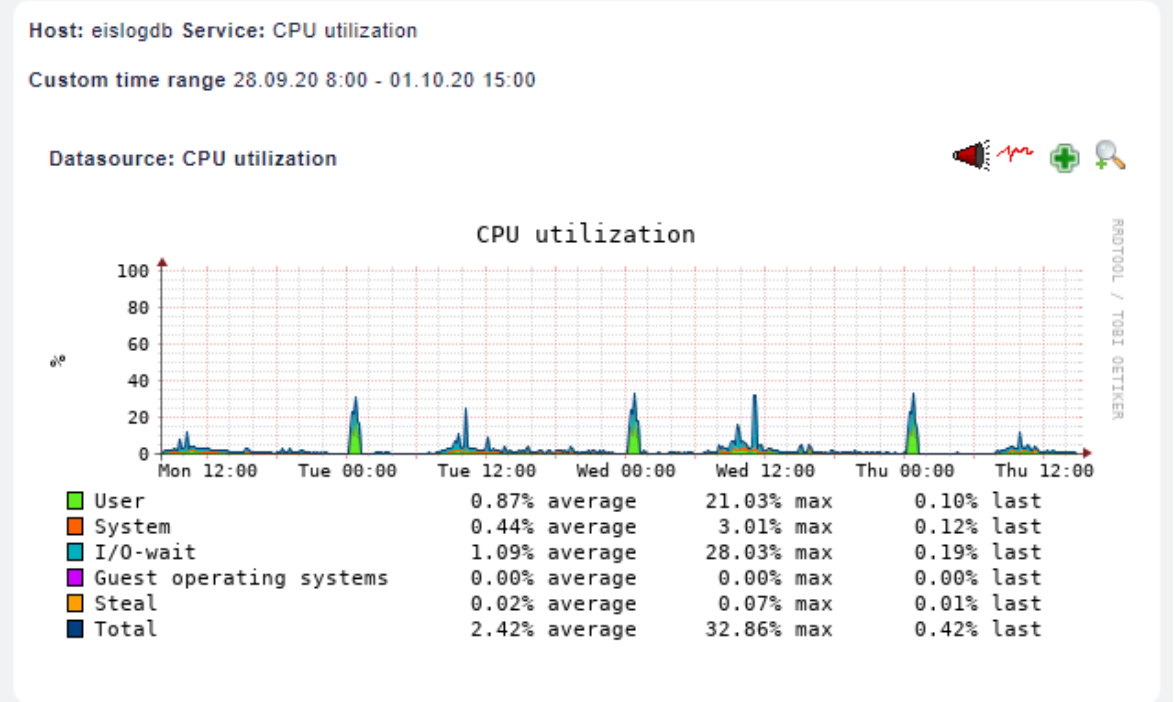
Põhiliseks erinevuseks kujunes süsteemi poolt teostatud arvutuste koormus. Kolmapäeval, 30. septembril kasutati esimese rakenduse protsessor täielikult eraldatud ressursside piiris ära – protsessori kasutus jõudis 100%-ni. Lisaks oli sarnases olukorras kuues rakendusserver, mis kasutas protsessori ressursi ära 80%. Kuuenda rakendusega esines sama olukord uuesti neljapäeval. Arvutuste info võetakse ja sisestatakse

andmebaasi, millest tulenevalt tõusis põhiandmebaasi koormus kolmapäeval ja neljapäeval, mis on visuaalselt kujutatud joonisel 3.5.



Joonis 3.5. EIS põhibaasi protsessori kasutus - matemaatika tasemetöö

Andmebaasi ülesehituse poolest said suurema koormuse osaliseks matemaatika tasemetöö ajal PgBouncer ja logibaas. Kui loodusõpetuse tasemetöö ajal oli logibaasi protsessori ressursid kasutuses 17%, siis matemaatika tasemetöö ajal oli kasutus kasvanud kahekordseks, logibaasi koormus on märgitud joonisel 3.6. Koormuse kasvu on võimalik tuvastada ka PgBounceri graafikutelt, kus protsessori kasutus kasvas esimese tasemetööga võrreldes 10%. Arvutuste tegemine on süsteemi üheks suurimaks pudelikaelaks olnud siiamaani ja saab olema veel suuremaks edaspidi. Takistus seisneb selles, et järjest rohkem tehakse elektroonilisi test ning populaarsuse kas suurendab töödeldavate andemete kogum, mis omakorda hakkab tulevikus veel rohkem koormama andmebaase.



Joonis 3.6. EISi logibaasi protsessori kasutus - matemaatika tasemetöö

3.2 Tasemetööde kokkuvõte

Süsteem pidas vastu suurele koormusele ning saavutas uusi rekordeid üheaegsete kasutajate teenindamises. Maksimaalsete ühenduste arv oli 4118. Reaalne kasutus annab just kõige kindlama ülevaate süsteemi käideldavuse kohta, kuna automaat- ja/või koormustestidega on jäigalt kirjutatud mida simuleeritud kasutaja kavatseb teha ning see ei peegelda alati reaalse kasutaja käitumist. Sarnaselt toob reaalne kasutus välja kitsaskohad, milleks on arvutuste tegemine, mis mõjutab süsteemi tööd ning võib muutuda segavaks teguriks. Koormustulemused, millel puuduvad graafikud on välja toodud koodi lõikudena lisades (lisa 13 ja 14). Graafiku puudumine tuleneb sellest, et graafikud on kas liiga ühtlased või omavad ainult ühete ajahetke, kus koormustase kerkis kõrgele.

KOKKUVÕTE

Rakenduskõrgharidustöö eesmärgiks oli Eksamite Infosüsteemi võimekuse parandamine. Eesmärgi saavutamiseks tehti uute süsteemi komponentide testimiseks eraldiseisvad arendusserverid, rakendati uusi ülesehitusi EISi test- ja toodangukeskkondades, testiti süsteemi vastupidavust simuleeritud koormustestidele ja analüüsiti vastupidavust reaalsele kasutusele. EISi võimekuse parandamiseks tuli välja vahetada rakenduste proksiserver ja optimeerida andmebaaside ülesehitus. Eksamite Infosüsteemi koormustaluvuse tõstmine kuulutatakse edukaks, kui süsteem on võimeline teenindama 4000 simuleeritud kasutajat korraga.

Süsteemi lõplikuks koormustestiks oli 2020. aasta septembri viimases pooles tehtud reaalne süsteemi kasutus, kus 7. klassi õpilased läbisid matemaatika ja loodusõpetuse tasemetöid. Süsteem teenindas tipphetkel ära 4118 üheaegset ühendust ning tasemetööde käigus ei esinenud ühtegi tõrget süsteemi töös. Saadud katsetulemuste põhjal võib töö eesmärgi lugeda saavutatuks.

Optimeerimisvõimalusi on veel mitmeid nii rakenduse koodi kui ka kasutusel oleva tarkvara seadistuse poolelt. Eksamite Infosüsteemi jõudluse taluvus on tõusnud iga uue süsteemi komponendi rakendamisega, kuid kogu au ei kuulu rakendatud tarkvaradele. EISi rakendusi on arendatud ja optimeeritud edasi ka Harno arendaja ja projektijuhi poolt. Optimeerimisvõimalusi on tekkinud tänu koormustestide tegemise ja nende tulemuste analüüsimisel. Eksamite Infosüsteemi edasiseks jõudluse tõstmiseks tuleks uurida Redise andmebaasi rakendamist ühenduste sessioonide talletamiseks PostgreSQL andmebaasi asemel.

SUMMARY

This thesis was aimed at improving the capabilities of the Examination Information System. To fulfill the objective new application servers were built, a new setup was implemented in both test and production environments, the system was stress tested with simulated load and compared against real use case. To improve the system's capabilities application's proxy server was swapped out and the database setup was optimized. The thesis will be confirmed as successful if the system is able to sustain 4000 concurrent users.

The final stress test was conducted in late September 2020 when 7th grade students performed evaluation examinations in math and biology. System maintained 4118 concurrent connections during highlight. During examinations, no errors were observed. Based on the results of this final use case, it can be concluded thesis was successful.

Even though the thesis was successful, there are still possibilities to optimize the system even further by improving application code as well as current software setups. Examination Information System's capabilities have improved with every new component implementation, but the system's developer and project lead have developed and optimized the system even further as well. With every stress test and its analysis, new possibilities of optimization have arisen for instance by considering applying Redis database for storing session data rather than the current PostgreSQL setup.

KASUTATUD KIRJANDUSE LOETELU

[1] Ilker Koksal. The rise of online learning. [*Online*]
https://www.forbes.com/sites/ilkerkoksal/2020/05/02/the-rise-of-online-learning/?sh=6bbf37ca72f3 (08.12.2020)
[2] Apache – HTTP Server Project. [*Online*]
https://httpd.apache.org/ (15.04.2020)
[3] Kavya. Apache Vs NGINX – Which Is The Best Web Server for You? [*Online*]
https://serverguy.com/comparison/apache-vs-nginx/#:~:text=Apache%20Vs%20NGINX%202020,multiple%20requests%20within%20one%20thread. (30.07.2020)
[4] HAProxy The Reliable, High Performance TCP/HTTP Load Balancer. [*Online*]
http://www.haproxy.org/ (21.01.2020)
[5] Ettevõttest. [*Online*]
https://www.skidsolutions.eu/ettevottest/ (05.02.2020)
[6] Tühistusnimekiri. [*Online*]
https://www.skidsolutions.eu/repositoorium/CRL/CRL (14.10.2020)
[7] Tuning Your PostgreSQL. [*Online*]
https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server (09.10.2019)
[8] PostgreSQL Connection Pooling: Part 1 – Pros & Cons [*Online*]
https://scalegrid.io/blog/postgresql-connection-pooling-part-1-pros-and-cons/ (08.12.2020)
[9] PostgreSQL Connection Pooling: Part 4 – PgBouncer vs. Pgpool-II [*Online*]
https://scalegrid.io/blog/postgresql-connection-pooling-part-4-pgbouncer-vs-pgpool/ (08.12.2020)
[10] Serverivabrik. [*Online*]
https://arendus.eenet.ee/w/arendusveeb/serverivabrik/ (10.11.2020)
[11] PostgreSQL-based application performance: latency and hidden delays. [*Online*]
https://www.2ndquadrant.com/en/blog/postgresql-latency-pipelining-batching/ (08.12.2020)

LISAD

Lisa 1 Testsüsteemi HAProxy esialgne seadistus

```
Global
    daemon

Defaults
    mode http
    timeout connect 5s
    timeout client 500s
    timeout server 500s

frontend proxy
    bind *:80
    redirect scheme https if !{ ssl_fc }
    acl letsencrypt path_beg /.well-known/acme-challenge/
    use_backend letsencrypt if letsencrypt
    bind *:443 ssl crt /srv/certs/haproxy.eenet.ee.pem
    default_backend rakendusserverid

backend rakendusserverid
    balance roundrobin
    server haproxyrakendus1 10.40.19.51:80 check
    server haproxyrakendus2 10.40.19.52:80 check

backend letsencrypt
    server letsencrypt 127.0.0.1:8080
```

Lisa 2 Testsüsteemi Apache esialgne seadistus

```
<VirtualHost 10.40.19.51:80>
    DocumentRoot "/srv/http/avalik"
    ServerName haproxy.eenet.ee
</VirtualHost>
```

Lisa 3 Testsüsteemi HAProxy täiendatud seadistus

```
Global
    daemon

defaults
    mode http
    timeout connect 5s
    timeout client 500s
    timeout server 500s

frontend proxy
    bind *:80
    # Headerid, mida edastada tagaolevale rakendusele - ID-Kaart
    http-request set-header SSL-CLIENT-I-DN       %{+Q}[ssl_c_i_dn]
    http-request set-header SSL-CLIENT-I-DN-CN    %{+Q}[ssl_c_i_dn(cn)]
    http-request set-header SSL-CLIENT-VERIFY     %[ssl_c_verify]
    http-request set-header SSL-CLIENT-S-DN       %{+Q}[ssl_c_s_dn]
    http-request set-header SSL-CLIENT-S-DN-CN    %{+Q}[ssl_c_s_dn(cn)]
    http-request set-header SSL-CLIENT-S-DN-O     %{+Q}[ssl_c_s_dn(o)]
    http-request set-header SSL-CLIENT-M-SERIAL   %[ssl_c_serial,hex]
    http-request set-header SSL-CLIENT-CERT       %{+Q}[ssl_c_der,base64]
    http-request set-header SSL                   %[ssl_fc]
    http-request set-header SSL-Client-SHA1       %{+Q}[ssl_c_sha1,hex]
    http-request set-header SSL-Client-NotBefore  %{+Q}[ssl_c_notbefore]
    http-request set-header SSL-Client-NotAfter   %{+Q}[ssl_c_notafter]
    http-request set-header SSL-Client-Version    %{+Q}[ssl_c_version]
    http-request set-header X-Forwarded-Port %[dst_port]
    http-request add-header X-Forwarded-Proto https if { ssl_fc }
    redirect scheme https if !{ ssl_fc }
    acl letsencrypt path_beg /.well-known/acme-challenge/
    use_backend letsencrypt if letsencrypt
    bind *:443 ssl crt-list /srv/certs/crt-list
    # Kasutaja IP edastamine
    option forwardfor
    option http-server-close
    # Rakenduste juurde suunamine
    default_backend rakendusserverid

backend rakendusserverid
    balance roundrobin
    server haproxyrakendus1 10.40.19.51:80 check
    server haproxyrakendus2 10.40.19.52:80 check

backend letsencrypt
    server letsencrypt 127.0.0.1:8080
```

Lisa 4 Testsüsteemi Apache täiendatud seadistus

```
<VirtualHost 10.40.19.51:80>
      DocumentRoot "/srv/http/avalik"
      ServerName haproxy.eenet.ee
</VirtualHost>

<VirtualHost 10.40.19.51:80>
   DocumentRoot "/srv/http/id-kaart"
   ServerName haproxy-id.eenet.ee

   SetEnvIf SSL_CLIENT_I_DN "(..*)" SSL-CLIENT-I-DN=$1
   SetEnvIf SSL-CLIENT-I-DN-CN "(..*)" SSL-CLIENT-I-DN-CN=$1

   SetEnvIf SSL-CLIENT-VERIFY "0" SSL-CLIENT-VERIFY=SUCCESS
   #SetEnvIf X-SSL-CLIENT-VERIFY "(..*)" SSL-CLIENT-VERIFY=$1

   SetEnvIf SSL-CLIENT-S-DN "(..*)" SSL-CLIENT-S-DN=$1
   SetEnvIf SSL-CLIENT-S-DN-CN "(..*)" SSL-CLIENT-S-DN-CN=$1
   SetEnvIf SSL-CLIENT-S-DN-O "(..*)" SSL-CLIENT-S-DN-O=$1
   SetEnvIf SSL-CLIENT-M-SERIAL "(..*)" SSL-CLIENT-M-SERIAL=$1
   SetEnvIf SSL-CLIENT-CERT "(..*)" SSL-CLIENT-CERT=$1
</VirtualHost>
```

Lisa 5 HAProxy domeenide nimekirja faili ülesehitus

```
/srv/certs/haproxy-id.eenet.ee.pem [ca-file /srv/certs/id.crt verify required alpn h2,http/1.1] haproxy-
id.eenet.ee
/srv/certs/haproxy.eenet.ee.pem [alpn h2,http/1.1] haproxy.eenet.ee
```

Lisa 6 Eksamite Infosüsteemi HAProxy seadistusfail

```
Global
    daemon
    maxconn 100000
    tune.ssl.cachesize 2000000
    tune.maxrewrite 16384
    tune.bufsize 32768

defaults
    mode http
    timeout connect 5000ms
    timeout client 180000ms
    timeout server 180000ms
    errorfile 503 /srv/haproxy/comeback.html
    maxconn 20000

frontend haproxy
    # Kuulame 80 porti
    bind :80
    # Domeeni nime vahetus
    redirect prefix https://testid.edu.ee code 301 if { hdr(host) -i eis.innove.ee }
    # Ümbersuunamine HTTPS'le
    redirect scheme https if !{ ssl_fc }
    # Kuulame 443 porti ja määrame bundle sertifikaadi asukoha
    bind :443 ssl crt /srv/certs/testid.edu.ee-bundle.crt
    # Rakendusservri valiku juurde suunamine
    default_backend rakendusserverid

backend rakendusserverid
    balance roundrobin
    option forwardfor
    option http-server-close
    http-request set-header X-Forwarded-Port %[dst_port]
    http-request add-header X-Forwarded-Proto https if { ssl_fc }
    acl network_allowed src 10.40.0.31
    acl restricted_page path_beg /adapter
    http-request deny if restricted_page !network_allowed
    acl path_static path_beg /ekk/muud/clufs
    use-server rakendus01 if path_static
    server rakendus01 10.64.0.111:80 check
    server rakendus02 10.64.0.112:80 check
    server rakendus03 10.64.0.113:80 check
    server rakendus04 10.64.0.114:80 check
```

Lisa 7 Andmebaasi esimene skript

```
#!/bin/bash
con=10
while [ $con -lt 1010 ]; do
echo -e "\e[96mDB masinas tegevused\e[0m"
ssh -o "StrictHostKeyChecking no" jay@10.40.19.56 -T << EOF
sudo -i
/srv/pgtune/pgtune.py -c $con -l "*" | grep -v "log" > /srv/postgres/data/tune.conf
systemctl restart postgresql
exit
exit
EOF
tran=$((30000/$con))
echo -e "\e[96m---------- $con ----------\e[0m"
i=0
while [ $i -lt 20 ]; do
pgbench -U postgres -h 10.40.19.56 -n -c $con -j 4 -t $tran testdb
i=$(($i+1))
done
con=$(($con+10))
done
```

Lisa 8 Andmebaasi teine skript

```
#!/bin/bash
con=10
while [ $con -lt 1010 ]; do
echo -e "\e[96mPOOL masinas tegevused\e[0m"
ssh -o "StrictHostKeyChecking no" jay@10.40.19.55 -T << EOF
sudo -i
/srv/conf.sh $con
systemctl restart pgbouncer
exit
exit
EOF
echo -e "\e[96mDB masinas tegevused\e[0m"
ssh -o "StrictHostKeyChecking no" jay@10.40.19.56 -T << EOF
sudo -i
/srv/pgtune/pgtune.py -c $con -I "*" | grep -v "log" > /srv/postgres/data/tune.conf
systemctl restart postgresql
exit
exit
EOF
tran=$((30000/1000))
echo -e "\e[96m---------- $con tps and latency ----------\e[0m"
i=0
while [ $i -lt 20 ]; do
pgbench -U postgres -h 10.40.19.55 -n -c 1000 -j 4 -t $tran testdb
i=$(($i+1))
done
con=$(($con+10))
done
```

Lisa 9 Andmebaasi kolmas skript

```
#!/bin/bash
con=1000
while [ $con -lt 10010 ]; do
tran=$((30000/$con))
echo -e "\e[96m---------- $con ---------- tps and latency\e[0m"
i=0
while [ $i -lt 20 ]; do
pgbench -U postgres -h 10.40.19.55 -n -c $con -j 4 -t $tran testdb
i=$(($i+1))
done
con=$(($con+100))
done
```

Lisa 10 JMeter koormustestid peale HAProxy rakendamist

Date		User	Req	Time	reg/s	Avg	Min	Max	Err	Err (%)
02.11 2018	1	600	94200	0:20:02	78.4	102	1	5730	0	0.00%
02.11 2018	1	2000	314000	0:22:56	228.2	539	0	100018	66	0.02%
02.11 2018	1	2500	392500	0:25:30	256.6	1433	0	100386	408	0.10%
14.05 2019	1	100	15700	0:20:37	12.7	81	1	2359	300	1.91%
14.05 2019	1	100	78500	0:20:07	65	103	0	4314	0	0.00%
14.05 2019	1	500	78500	0:21:31	60.8	437	0	35354	0	0.00%
14.05 2019	1	700	109900	0:22:56	79.9	1114	0	102692	0	0.00%
14.05 2019	1	900	141300	0:24:19	96.9	1703	0	142166	0	0.00%
14.05 2019	1	1500	235500	0:22:06	177.6	231	0	16021	0	0.00%
16.05 2019	3	1000	157000	0:38:11	68.5	6806	0	600019	133	0.08%
		1000	157000	0:38:24	68.2	6720	0	600025	138	0.09%
		1000	157000	0:39:06	66.9	6549	0	600028	170	0.11%
16.05 2019	2	1250	196250	0:21:51	149.7	694	0	57577	0	0.00%
		1250	196250	0:22:16	146.9	696	0	55285	0	0.00%
16.05 2019	2	1250	196250	0:22:34	144.9	1124	0	122605	0	0.00%
		1250	196250	0:24:10	135.3	1109	0	119987	0	0.00%
16.05 2019	2	1250	196250	0:24:43	132.3	1765	0	221524	0	0.00%
		1250	196250	0:25:06	130.3	1794	0	216256	0	0.00%
20.05 2019	2	1250	196250	0:24:47	132	1735	0	135483	0	0.00%
		1250	196250	0:24:49	131.8	1640	0	136675	0	0.00%
25.06 2019	1	500	78500	0:21:18	61.4	361	1	30339	0	0.00%
25.06 2019	1	1000	157000	0:23:11	112.8	1021	1	141270	1	0.00%
09.07 2019	1	400	62800	0:20:39	50.7	262	1	24751	0	0.00%
09.07 2019	1	400	62800	0:20:49	50.3	259	1	21684	0	0.00%
09.07 2019	1	400	62800	0:20:31	51	255	1	19117	0	0.00%

Lisa 11 JMeter koormustestid peale andmebaaside hajutamist

Date		User	Req	Time	reg/s	Avg	Min	Max	Err	Err (%)
12.07 2019	2	1500	235500	0:21:23	183.6	381	0	71669	0	0.00%
		1500	235500	0:21:37	181.5	385	1	73548	0	0.00%
12.07 2019	2	2000	314000	0:46:59	111.4	9390	0	600029	8	0.00%
		2000	314000	0:47:45	109.6	9778	1	600457	23	0.01%
12.07 2019	2	1500	235500	0:32:02	122.5	4589	1	339788	3	0.00%
		1500	235500	0:32:45	119.8	4468	0	528817	4	0.00%
28.08 2019	2	1500	235500	0:23:19	168.3	1264	0	138574	0	0.00%
		1500	235500	0:24:05	162.9	1309	1	135779	0	0.00%
28.08 2019	2	2000	314000	0:30:10	173.5	3350	0	293795	0	0.00%
		2000	314000	0:30:10	173.5	3528	1	300038	1	0.00%
28.08 2019	2	2000	314000	0:27:13	192.2	2476	0	313079	0	0.00%
			314000	0:27:33	189.9	2494	1	298087	0	0.00%
29.08 2019	2	2000	314000	0:27:55	187.4	2590	1	387558	0	0.00%
			314000	0:27:53	187.6	2357	0	388284	1	0.00%
17.12 2019	2	1500	214500	0:21:56	163	490	0	24640	0	0.00%
			214500	0:21:41	164.9	584	1	25499	0	0.00%

Lisa 12 JMeter koormustestid peale PgBouncer rakendamist

Date		User	Req	Time	reg/s	Avg	Min	Max	Err	Err (%)
19.12 2019	2	1500	187191	0:29:57	104.2	7118	0	300397	4255	2.27%
			189187	0:30:00	105.1	6963	0	300113	4686	2.48%
20.12 2019	2	1500	214500	0:25:58	137.7	2213	0	106639	77	0.04%
			214500	0:26:13	136.4	2255	1	108133	61	0.03%
20.12 2019	2	1500	214500	0:25:42	139.1	2298	0	99291	0	0.00%
			214500	0:26:04	137.2	2253	1	99725	0	0.00%
20.12 2019	2	1500	214500	0:23:43	150.7	1272	0	71563	143	0.07%
			214500	0:24:19	147	1323	1	68423	173	0.08%
20.12 2019	2	1500	214500	0:24:38	145.2	1493	0	216648	708	0.33%
			214500	0:25:04	142.6	1524	1	216693	716	0.33%
20.12 2019	2	1500	214500	0:23:30	152.1	1089	0	113382	672	0.31%
			214500	0:23:45	150.6	1114	1	113536	818	0.38%
02.01 2020	2	1500	214500	0:34:58	102.2	1361	1	300098	8098	3.78%
			214500	0:34:53	102.5	1184	0	300035	5653	2.64%
03.01 2020	2	1500	214500	0:28:45	124.4	1308	0	300012	6146	2.87%
			214500	0:35:47	99.9	1409	1	300066	7024	3.27%
08.01 2020	1	1500	214500	0:20:33	173.9	102	0	2206	0	0.00%
9.01 2020	2	1500	214500	0:31:15	114.4	4740	0	68762	0	0.00%
			214500	0:31:26	113.7	4699	1	67570	0	0.00%
9.01 2020	2	1500	214500	0:32:18	110.7	5473	0	66456	0	0.00%
			214500	0:32:21	110.5	5484	0	66303	0	0.00%
10.01 2020	2	2000	286000	0:43:53	108.6	10397	0	110455	0	0.00%
			286000	0:44:06	108.1	10390	0	110007	0	0.00%
14.01 2020	2	1500	147000	0:14:30	168.9	4906	0	69204	0	0.00%
			147000	0:14:33	168.4	4911	0	69159	0	0.00%
14.01 2020	2	2000	196000	0:21:56	148.9	7943	0	151862	283	0.14%
			196000	0:21:41	150.6	7949	0	151376	332	0.17%
14.01 2020	1	2000	190711	0:09:15	343.6	2129	0	47332	0	0.00%
15.01 2020	2	1500	147000	0:07:59	306.8	519	0	27422	0	0.00%
			147000	0:08:04	303.6	623	0	27725	0	0.00%
15.01 2020	2	2000	196000	0:09:50	332.2	1670	0	65200	24	0.01%
			196000	0:10:07	322.6	1619	0	65509	42	0.02%
15.01 2020	3	2000	196000	0:14:29	225.7	4652	0	154507	2014	1.03%

			196000	0:14:32	224.8	4683	0	148112	1986	1.01%
			196000	0:15:51	206.1	4957	1	171744	2260	1.15%
16.01 2020	2	2000	196000	0:11:02	296.2	2411	0	49227	0	0.00%
			196000	0:10:56	298.6	2452	0	48007	0	0.00%
16.01 2020	2	2000	196000	0:11:34	282.3	2851	0	55064	0	0.00%
			196000	0:11:31	283.7	2856	0	53944	0	0.00%
17.01 2020	2	2000	196000	0:09:39	338.7	1503	0	56558	0	0.00%
			196000	0:09:48	333.2	1464	0	57120	0	0.00%
21.01 2020	2	2000	196000	0:09:17	351.9	1335	0	43179	0	0.00%
			196000	0:09:21	349.5	1283	0	43137	0	0.00%
21.01 2020	2	2000	196000	0:24:47	131.8	10842	0	168639	114	0.06%
			196000	0:24:45	132	10895	0	160587	105	0.05%
21.01 2020	3	2000	196000	0:10:30	311.1	2040	0	62456	210	0.11%
			196000	0:10:37	307.6	2055	0	61330	224	0.11%
			196000	0:15:21	212.9	2982	1	130585	1994	1.02%
21.01 2020	3	1666	163268	0:09:45	278.9	1678	0	60361	47	0.03%
			163268	0:09:45	279	1610	0	60024	41	0.03%
			163268	0:15:11	179.2	2677	0	130420	1516	0.93%
21.01 2020	3	1666	163268	0:10:20	263.3	1923	0	44096	0	0.00%
			163268	0:10:21	263.1	1921	0	44647	0	0.00%
			163268	0:10:18	264.2	2112	1	44234	0	0.00%
21.01 2020	3	2000	196000	0:11:51	275.5	2847	0	68896	426	0.22%
			196000	0:11:57	273.4	2918	0	69062	446	0.23%
			196000	0:12:05	270.5	2859	0	67576	460	0.23%
21.01 2020	4	1500	147000	0:12:01	204	2919	0	76978	833	0.57%
			147000	0:11:57	205.1	2922	0	73345	811	0.55%
			147000	0:12:11	201	2941	0	93711	727	0.49%
			147000	0:11:51	206.8	2997	0	77390	781	0.53%
07.05 2020	2	2000	89803	0:02:47	537.9	756	0	132969	434	0.48%
			148553	0:03:46	656.7	306	0	3021	801	0.54%
07.05 2020	2	2000	195900	0:14:54	219	1496	0	300013	2463	1.26%
			195851	0:14:54	219.1	382	0	300042	1805	0.92%
07.05 2020	2	2000	196000	0:10:59	297.5	342	0	300018	1	0.00%
			196000	0:11:33	282.7	1501	0	300012	1495	0.76%
07.05 2020	2	2000	196000	0:09:00	363.1	1173	0	6890	0	0.00%
			196000	0:08:59	363.5	1107	0	6112	0	0.00%
17.09 2020	2	1500	103500	0:08:19	207.6	1359	1	41286	0	0.00%
			103500	0:08:20	207.2	1367	0	41007	0	0.00%

18.09 2020	2	2000	354000	0:22:07	266.8	1603	0	11934	0	0.00%
			354000	0:22:10	266.2	1577	0	10054	0	0.00%

Lisa 13 Tasemetööde käigus tekkinud koormused – loodusõpetus

Kuupäev	**21.09**	**22.09**	**23.09**	**24.09**
Maksimaalne ühenduste arv	2630.08	3109.00	3253.25	2825.00
Keskmine ühenduste arv	1476.14	1492.15	1590.97	1359.22
Maksimaalne aadresside arv	172.30	154.25	220.75	163.75
Keskmine aadressite arv	121.36	113.81	140.27	113.13
Maksimaalne eisdb CPU koormus	10.39 %	10.13 %	42.02 %	10.30 %
Keskmine eisdb CPU koormus	3.20 %	3.66 %	3.01 %	2.33 %
Maksimaalne eislogdb CPU koormus	17.68 %	10.53 %	5.57 %	11.05 %
Keskmine eislogdb CPU koormus	2.87%	2.27 %	1.77 %	1.60%
Maksimaalne rakenduste CPU koormus				
Rakendus 1	8.84 %	68.45 %	9.32 %	65.16 %
Rakendus 2	11.42 %	8.51 %	60.08 %	7.55 %
Rakendus 3	9.78 %	62.78 %	9.18 %	7.69 %
Rakendus 4	7.96 %	7.82 %	8.99 %	21.12 %
Rakendus 5	8.96 %	10.44 %	8.89 %	7.19 %
Rakendus 6	7.26%	7.81 %	9.23 %	8.25 %
Keskmine rakenduste CPU koormus				
Rakendus 1	4.25 %	9.56 %	3.64 %	5.63 %
Rakendus 2	3.74 %	4.11 %	5.11 %	2.89 %
Rakendus 3	3.80 %	5.29 %	3.50 %	3.02 %
Rakendus 4	3.85 %	4.34 %	3.53 %	3.11 %
Rakendus 5	4.08 %	4.44 %	3.63 %	3.14 %
Rakendus 6	3.80 %	4.12 %	3.39 %	3.00 %
Maksimaalne rakenduste RAM kasutus	**21.09 – 24.09**			
Rakendus 1	6.0 GB			
Rakendus 2	5.4 GB			
Rakendus 3	6.0 GB			
Rakendus 4	4.7 GB			
Rakendus 5	4.4 GB			
Rakendus 6	4.4 GB			
Maksimaalne pgbouncer CPU koormus	13.58 %			
Maksimaalne eisporxy CPU koormus	3.09 %			
Maksimaalne eisproxy RAM kasutus	1.8 %			

Lisa 14 Tasemetööde käigus tekkinud koormused – matemaatika

Kuupäev	**28.09**	**29.09**	**30.09**	**01.10**
Maksimaalne ühenduste arv	3603.53	3209.20	4118.37	3226.75
Keskmine ühenduste arv	1447.02	1300.84	1676.29	1243.80
Maksimaalne aadresside arv	171.00	153.00	214.82	153.35
Keskmine aadressite arv	119.51	113.39	122.02	92.61
Maksimaalne eisdb CPU koormus	14.35 %	10.32 %	64.48 %	99.98 %
Keskmine eisdb CPU koormus	5.38 %	2.87 %	15.17 %	4.83 %
Maksimaalne eislogdb CPU koormus	11.66 %	24.37 %	31.70 %	11.50 %
Keskmine eislogdb CPU koormus	2.32 %	3.01 %	3.89 %	1.61 %
Maksimaalne rakenduste CPU koormus				
Rakendus 1	7.08 %	26.88 %	100.58 %	26.46 %
Rakendus 2	9.50 %	24.44 %	21.74 %	25.34 %
Rakendus 3	5.79 %	21.46 %	21.92 %	59.99 %
Rakendus 4	7.52 %	22.97 %	16.61 %	17.34 %
Rakendus 5	5.89 %	27.80 %	66.33 %	47.03 %
Rakendus 6	6.84 %	27.95%	80.96 %	80.13 %
Keskmine rakenduste CPU koormus				
Rakendus 1	3.72 %	4.04 %	11.36 %	2.36 %
Rakendus 2	3.41 %	3.78 %	3.26 %	2.05 %
Rakendus 3	3.18 %	3.58 %	3.24 %	4.11 %
Rakendus 4	3.22 %	3.22 %	3.09 %	2.03 %
Rakendus 5	3.14 %	3.96 %	6.19 %	2.26 %
Rakendus 6	3.22 %	3.78 %	9.22 %	2.28 %
Maksimaalne rakenduste RAM kasutus	**28.09 – 01.10**			
Rakendus 1	6.5 GB			
Rakendus 2	5.8 GB			
Rakendus 3	5.0 GB			
Rakendus 4	4.3 GB			
Rakendus 5	5.2 GB			
Rakendus 6	9.0 GB			
Maksimaalne pgbouncer CPU koormus	24.20 %			
Maksimaalne eisporxy CPU koormus	3.57 %			
Maksimaalne eisproxy RAM kasutus	1.8 GB			