TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Mats Johanson

Veebirakenduse monitoorimislahenduse realiseerimine Eesti Töötukassa infosüsteemi näitel

Bakalaureustöö (6 EAP)

Juhendajad: dotsent Helle Hein
Priit Liivak, Nortal AS

Autor: ....."...." mai 2013
Juhendaja: ....."...." mai 2013
Juhendaja: ....."...." mai 2013

Lubada kaitsmisele
Professor ....."...." mai 2013

TARTU 2013

Sisukord

Sissejuhatus

Aastal 2009 sündis ettevõttes Webmedia veebirakendus nimega EMPIS (*Employment Infosystem* – Töötukassa infosüsteem), aitamaks Eesti Töötukassal oma igapäevatööd efektiivsemalt teha. EMPISel oli suurepärane eelanalüüs, parimad arhitektid ja vankumatud testijad, mistõttu võrreldi tuliuue rakenduse kasutuselevõttu Töötukassas vankri auto vastu vahetamisega [1]. Nüüd, neli aastat hiljem, on süsteem jätkuvalt kiirelt arenemas ning uut funktsionaalsust lisandub iga päevaga.

Kuigi suurepärane analüüs ning arhitektuur seda aeglustanud on, siis veebirakenduse keerukuse ning koodibaasi kasvuga kaasneb alati selle hallatavuse langus – iga uue funktsionaalse tüki lisandumisega suureneb regressiooni oht ja efektiivsuse kadu. Vastused andmebaasist saabuvad mõned millisekundid hiljem, kuna andmete hulk on aasta-aastalt kasvanud või menüüriba avaneb uute valikute tõttu järjest kauem. Kui ainsaks indikaatoriks, mis seda paratamatut langust jälgib, on rakenduse kasutajate subjektiivne hinnang, siis selleks ajaks, kui nemad juba süsteemi aeglust tunnetavad, on liiga hilja.

Et rakenduse taset ning kliendi rahulolu kõrgel hoida, tuleb leida vahend, mis pidevalt süsteemil silma peal hoiaks ning kust oleks võimalik ammutada objektiivset informatsiooni - **monitoorimistarkvara.** Antud töö käsitlebki tervikliku monitoorimislahenduse otsimist, paigaldamist ning kohandamist reaalsele kogukale projektile – EMPISele – ning selle käigus püütakse leida vastused järgmistele küsimustele:

- Miks on tarvis tarkvara monitoorimist?
- Kuidas valida endale sobivat lahendust?
- Kuidas monitoorimistarkvarast maksimaalne kasutegur saada?

Töö on jagatud loogiliselt kahte ossa. Esimeses osas kirjeldatakse, mis on tarkvara monitooring ning milles peitub tema kasulikkus. Selleks tuuakse välja erinevaid jälgitavaid rakenduse kihte ja näidatakse, mida võib nende monitooringust võita. Esimese osa lõpetuseks on erinevaid jälgimislahendusi pakkuvate rakenduste võrdlus.

Töö teine osa käsitleb autori kogemusi ja järeldusi konkreetse EMPISele sobiva monitoorimistoote valimisel, selle paigaldamisel, seadistamisel ning kasutamisel.

1 Tarkvara jälgimine ehk monitooring

Monitooringu mõiste all peetakse silmas antud rakenduse kohta käivate teatud parameetrite jälgimist ning salvestamist, saamaks teavet süsteemi hetkeseisu kohta. Ideaalne jälgimissüsteem on vahend, mis annab ülevaatliku pildi kõikidest rakenduse kihtidest, ilma et kasutaja peaks otseselt koodi nägema. See võimaldab ka näiteks analüütikutel, projektijuhil või koguni kliendil ühe pilguga teada saada, mis olukorras tema rakendus on.

Sellise ülevaate omamine on kasulik väga mitmel viisil: leitakse kitsaskohti, mis ülejäänud programmi tagasi hoiavad; saadakse objektiivne pilt sellest, kuidas üks või teine uuendus on süsteemile mõjunud, võrreldes hetkeseisu eelnevalt kogunenud andmetega; suurendatakse kliendi usaldust ning rahulolu, tabades süsteemis esinevaid vigu kiiremini ja täpsemalt; nähakse ära, mida on süsteemi riistvarakihis tarvis uuendada või välja vahetada ja palju muud. Kasulikke oodatavaid tulemusi on mitmeid – need võib kokkuvõtvalt nimetada jälgitava rakenduse efektiivsuse tõstmiseks ning see ongi monitooringu peamiseks eesmärgiks.

1.1 Jälgitavad parameetrid

Kuna erinevaid jälgitavaid muutujaid on lihtne leida lõpmatul hulgal, tuleb teha kitsendusi, leidmaks monitooritava rakenduse äriloogikat kõige rohkem mõjutavad kohad. Antud töös on peamiseks uurimisobjektiks veebirakendus, seega kasutajakihist riistvarakihi suunas liikudes huvitavad meid järgmised peamised uurimisaspektid [2].

1.1.1 Kasutajate käitumine

Mis aegadel päevas, mis nädalapäevadel, millisest maailmajaost ning milliste sündmustega seotult rakendust enim kasutatakse? Millised veebilehitsejad on klientide seas kõige populaarsemad? Ärikriitiliste otsuste tegemiseks on selline informatsioon hädavajalik.

1.1.2 Lõppkasutaja jõudlus

Kui kaua läheb aega kasutajapoolsest hiireklõpsust kuni kuva viimase elemendi laadimiseni ning mis osa selles protsessis võtab kõige kauem aega? Üldiselt peaks rakenduse kosteaeg jääma alla 0,1 sekundi, kui tegemist on lihtsate kasutusliidese

operatsioonidega (tulbas väärtuse valimine); alla 1 sekundi lihtsamate arvutuste korral (tulba väärtuste ümber sorteerimine); alla 10 sekundi aeganõudvamate toimingute (andebaasist väärtuste laadimine) puhul. Üle 10 sekundi peab kasutajale näitama progressiriba ning võimalust tegevust katkestada, muidu võib tema tähelepanu hajuda [3].

Lõppkasutaja jõudlus annab arendajale võimaluse näha, kas lehekülg laetakse kasutajale enne neid mugavuspiiranguid ja kui ei, siis miks mitte. Tihti võib mahukate lehekülgede kokku panemisele minna arvestatav aeg näiteks piltide kuvamise või lohaka Javascripti käivitamise peale, kuid seda teadmata võib arendaja ekslikult kulutada ressursse hoopis andmebaasikihi optimeerimisele.

1.1.3 Kasutatav funktsionaalsus

Millised protsessid on kõige rohkem käivitatud või millised kasutuslood läbitud? Rakenduse optimeerimisel peitub suurim kasutegur just selliste kohtadega tegelemises, mida enim kasutatakse, kuna klientide arv, kelle jaoks rakendus kiiremaks muutub, on maksimeeritud.

Lisaks on süsteemis võimalik leida kohti, mida kasutatakse vähe või üldse mitte ning otsida põhjuseid, miks see nii on. Vastasel juhul võib nende arendamiseks kulutatud aja lugeda raisatuks.

1.1.4 Rakenduse liidestus

Paljudel süsteemidel on tarvis andmeid vahetada väliste rakendustega, mis aga ei pruugi alati kõige kiirema kosteajaga olla. Nende suhtluse jälgimine annab arendajale teada, millised liidestused on kõige ebaoptimaalsemad ning tuleks üle kontrollida.

1.1.5 Rakenduse jõudlus

Kui palju võttis aega kindla meetodi väljakutse Java kihis? Kas meil on koodi, mida käivitatakse väga tihti, aga on samas ebaoptimaalselt kirjutatud? Kui lõppkasutaja jõudlus näitab, kui palju aega võtab ühele kasutaja tegevusele reageerimine, siis rakenduse jõudluse statistikast saame leida probleeme, mis asuvad madalamal tasemel.

1.1.6 Andmebaasi jõudlus

Millised päringud võtavad süsteemis kõige rohkem aega? See on peamine küsimus, millele antud parameetri juures vastust otsida. Tõenäoliselt leidub igas rakenduses korralikult optimeerimata päringuid, kuid kui neid tihti välja ei kutsuta, siis ei mõjuta see kasutaja rahulolu kuigi tõsisel määral ning nende peale aega kulutada ei ole otstarbekas. Kui aga jälgida päringute väljakutseid, saame täpselt teada, millised neist vajavad kõige hoolikamat tähelepanu. Kuivõrd baasipäringud on tihti veebirakenduse üks ajakulukamaid tegevusi, siis võit, mida õigeid kohti optimeerides saab, on märgatav.

1.1.7 Arvutivõrk

Võrguliikluse jälgimine on järjest suurenevate andmemahtude juures üks tähtsamaid uurimisaspekte. Olenemata sellest, kui kiiresti toimib optimeeritud programm või riistvara, võib võrguliiklus kasutaja jaoks rakenduse kosteaja siiski talumatult pikaks venitada. Lisaks on oluline teada, kas rakenduse osad on üksteisega ühenduses – kui näiteks andmebaasiga ei saada ühendust, siis õigesti kasutatud monitooringu korral tuleb see kohe välja, vastasel juhul võib kuluda palju väärtuslikku aega, et vigade tegelik põhjus leida.

1.1.8 Riistvara

Riistvara monitoorimine annab näiteks informatsiooni selle kohta, kui palju on vaba kõvakettaruumi, operatiivmälu, protsessori aega, protsessori tuumi ja nii edasi. Selle põhjal on lihtsam näha, kas rakenduse aegluse taga on ehk liiga vähene mälukogus või on mõni muu seade liiga nõrgaks jäänud. Lisaks on operatiivmälu kasutuse stabiilse tõusu järgi leitavad mälulekked ning statistiliste andmete põhjal saab planeerida, millal järgmist uuendust vaja läheb.

1.2 Monitoorimistarkvara

Mida suuremaks rakendused kasvavad, seda keerulisemaks muutub nende kiiruse ja vigade jälgimine logide või skriptide abil – seega on efektiivseks jälgimiseks mõistlik kasutusele võtta valmisrakendus, mis teeb administraatori eest korduva töö ära ning kuvab seda kasutajale loetaval kujul – näiteks graafikutena.

Kuna veebirakendused on suures osas väga sarnastele kihtidele ehitatud (standardiseeritud on riistvara, andmebaas, serveri tarkvara ning veebikiht) [4], siis on suurt osa rakendusest võimalik jälgimistarkvaraga liidestada väga vähese vaevaga. Ainus kiht, mis vajab eraldi lähenemist, on ärispetsiifiliste protsesside kiht, kus universaalselt ehitatud tarkvara ei saa ära arvata, milliseid tegevusi omavahel kokku grupeerida ja milliseid eraldi välja tuua. Selleks on aga jälgimistarkvaral tihti kaasa antud tööriistad, mille abil oma programmis kriitilised kohad ära märkida.

Järgnevalt on välja toodud valik populaarsemaid monitoorimisrakendusi [5-7] koos nende maksumuse, eelmises peatükis väljatoodud parameetrite katvuse ning huvitavamate karakteristikutega. Valikus on keskendutud toodetele, mis on populaarsed, pikaajalise kogemusega või toonud turule olulist innovatsiooni.

1.2.1 Google Analytics

Google Analytics [8] (edaspidi GA) on ülekaalukalt kõige populaarsem veebilehtede analüüsivahend [9]. See jälgib kasutajate tegevusi, kogub statistikat lehekülgede populaarsuse ja kiiruse kohta ning on paigaldatav ühe Javascripti lõigu enda koodi lisamisega. Põhifunktsionaalsus on saadaval tasuta ning lisateenuste eest tuleb maksta $150,000 aastas ühe kasutajakonto kohta, kui selle kontoga seotud lehtede külastajate arv jääb alla miljardi kuus [10]. Uuritavatest parameetritest katab GA täielikult kasutajate käitumise, osaliselt kasutatava funktsionaalsuse ning kasutajapoolse jõudluse. GA ei sõltu arendusplatvormist.

1.2.2 Pingdom

Pingdom [11] on veebiteenus, mis spetsialiseerub eeskätt monitooritava tarkvara globaalse kättesaadavuse analüüsimisele, kasutades oma üle maailma paiknevaid servereid. See katab kõik kasutajapoolsed jälgitavad parameetrid ning on paigaldatav sarnaselt GA-ga Javascripti täienduse abil ega sõltu arendusplatvormist. On võimalik valida kolme hinnapaketi vahel – 9,95$, 39,95$ või 495$ kuus – vastavalt jälgitavate rakenduste arvule.

1.2.3 CA Application Performance Management (APM) [12]

Lew Cirne lõi antud rakenduse aastal 1998 koos firmaga Wily Technology Inc. ning see osteti CA poolt ära aastal 2006 [13]. Seega on tegemist väga kaua turul olnud lahendusega, mis katab kõik punktis 1.1 loetletud parameetrid, kuid eeldab ulatuslikku paigaldamisprotsessi, konfigureerimist ja algkapitali, kuna CA ei paku oma toodet veebiteenusena vaid ainult eraldi rakendusena. See tähendab arvestatavaid kasutajapoolseid aja- ning rahakulutusi, sest eeldab litsentside (9600$ iga jälgitava rakenduse kohta) ja uue riistvara või selle kasutamisteenuse eraldi sisseostu ning pidevat hooldust [14-15]. Toetab Java ning .Net platvormidel töötavate süsteemide jälgimist.

1.2.4 New Relic [16]

Peale Wily Technology Inc.-i müümist lõi Lew Cirne uue monitoorimistarkvara [17] nimega New Relic (edaspidi NR) ja parandas suurimad vead oma eelmise rakenduse juures – NR on saadaval veebiteenusena ning põhifunktsionaalsuse paigaldamine toimub vahetult ning sisaldab ainult rakendusserverile agendi lisamist. Piiratud funktsionaalsust saab kasutada tasuta, lisade eest tuleb maksta 24$ - 149$ kuus iga jälgitava serveri eest. NR toetab Ruby, PHP, Java, .NET, Python ja NodeJS arendusplatvorme ning katab kõik ülalpool välja toodud uurimisaspektid, kuigi riistvara kihi jälgimiseks on operatsioonisüsteemi tarvis paigaldada eraldi programm.

1.2.5 Nagios

Nagios [18] väljastati aastal 1999 [19] ning on siiani pidevalt uuenev vabavaraline tarkvara UNIX süsteemidele. Nagiosil on oma vanuse tõttu arvestatav kasutajate- ning lisanditepagas ja on spetsialiseerunud riistvara ning arvutivõrgu kihtide monitoorimisele – puudub kõrgemate kihtide, nagu andmebaas ja rakenduse jõudlus, jälgimisvõimalus, rääkimata kasutaja tegevustest. Nagiosi paigaldamine ja hooldus on keerukad tegevused, kuna tarkvara vajab sügavat konfigureerimist vastavalt jälgitavale süsteemile [20].

2 Monitoorimislahenduse realiseerimine

2.1 Jälgitav tarkvara – Eesti Töötukassa infosüsteem

EMPIS on Java platvormil mitme aasta vältel arendatud mahukas veebirakendus, mille eesmärgiks on lihtsustada ja kiirendada Töötukassa ametnike igapäevatöös ette tulevaid mehhaanilisi ning bürokraatlikke tegevusi: erinevate toetuste arvutamine, töötubadesse määramine, lepingute ning kokkulepete koostamine, statistika ja aruannete väljatrükk jne. – jättes neile rohkem aega kliendi kui inimese abistamiseks.

2.2 Sobivaima lahenduse valik

Kuna pakutavaid valmislahendusi on mitmeid, siis nende seast sobivaima leidmiseks tuleks kaaluda järgmiseid peamiseid kriteeriume: hind, paigaldatavus, hallatavus ning peatükis 1.1 välja toodud parameetrite katvus. Katvuse suurendamiseks on loomulikult võimalik erinevaid lahendusi omavahel kombineerida, kuid see mitmekordistab tõenäoliselt hinna ning suurendab eksponentsiaalselt nii paigaldatavuse kui hallatavuse raskusastet. Lisaks tuleb sellise lahenduse korral andmeid koguda mitmest allikast ning manuaalselt otsida parameetrite vahelist korrelatsiooni, selmet lasta komplektsel tööriistal see töö enda eest ära teha.

Üheks väga oluliseks boonuseks on ka see, et lahendust saaks sisse osta teenusena, mitte eraldi programmi ning litsentsidena, kuna ärilises kontekstis on oluline käia muudatustega kaasas – teenuse vahetamine on odav, aga aegunud või sobimatu rakenduse litsentside peale kulutatud raha on kadunud investeering.

Peatükis 1.2 välja toodud kiirest võrdlusest on näha, et mitmed pakutavatest variantidest on kas spetsialiseerunud kitsale alamosale kogu rakenduse kihtidest või on liiga kohmakad ja aegunud ning ei ole saadaval veebiteenusena.

Kuna EMPISes on hädavajalik katta kõik loetletud uurimiskriteeriumid, siis kõige paremini sobis kriteeriumitega **New Relic**, milles on tabatud hea tasakaal kergekaalulisuse ning võimekuse vahel.

2.3 New Relicu paigaldamine

2.3.1 Põhifunktsionaalsus

Esmalt tuli endale teha konto NR veebilehel – konto avamine ning põhifunktsionaalsuse kasutamine on tasuta, lisaks pakutakse kahenädalast prooviperioodi, mille vältel on kasutajale avatud kõikide hinnaklasside võimalused.

Autoriseeritud kasutajana oli võimalik alla laadida pakkefail, mis sisaldas kõike monitoorimise alustamiseks vajalikku: monitooritava rakenduse serveri külge paigaldatavat agenti (`newrelic.jar`) ning konfiguratsioonifaili (`newrelic.yml`). Konfiguratsioonifail sisaldab ka kasutaja võtit, mille abil agent teab, millisele kontole andmeid saatma hakata.

EMPIS kasutab oma rakendusserverina Apache Tomcat'i [21], millele NR agendi lisamine toimus järgmiselt:

1. Alla laetud agent ning konfiguratsioonifail tuli paigaldada Tomcat'i juurkausta;
2. Käivitada käsk "`java -jar newrelic.jar install`";
3. Failis `newrelic.yml` väärtustada parameeter „`app_name`" enda tarkvara nimega;
4. Käivitada server

Sellega oli põhifunktsionaalsus paigaldataud – NR kasutajakonsoolilt oli näha, et rakendusest kogutud statistika jõuab veebiteenuseni.

Põhifunktsionaalsus katab peatükis 1.1 välja toodud parameetritest viis: kasutatav funktsionaalsus, rakenduse liidestus, rakenduse jõudlus, andmebaasi jõudlus ning arvutivõrk. Kuna serveri agent ei näe otseselt ei veebilehitsejat ega riistvara, mille peal ta töötab, siis täieliku katvuse saavutamiseks on tarvis lisategevusi.

2.3.2 Lõppkasutaja jälgimine

Et jälgida lõppkasutajate käitumist ning jõudlust, kasutab NR kahte Javascripti, mis käivitatakse vastavalt iga lehekülje laadimise alustamisel ning lõpetamisel [22]. Nende skriptidega kogutakse andmeid lehekülje laadimiskiiruse, kasutaja asukoha ning

veebilehitseja tüübi kohta, andes rikkaliku konteksti hilisemaks andmete analüüsiks.

Vastavad Javascripti lõigud tuli lisada EMPISe jsp-failide [23] päisesse ning jalusesse [24], et saada iga lehekülje laadimise kiirusest võimalikult täpne ülevaade. Paigaldamise protsess oli kiire, kuid kuna rakenduse kood muutus, vajas lõppkasutaja jälgimise sisse lülitamine kogu rakenduse versiooni uuendust, mis aeglustas monitooringu esialgset käivitamiskiirust.

2.3.3 Riistvara jälgimine

Ainsaks katmata jälgimispiirkonnaks oli veel riistvara kiht. Rakendusserveril ei ole vaikimisi mingit võimalust näha, millise operatsioonisüsteemi, mälumahu või protsessoritega riistvara peal see töötab. Probleemi lahendamiseks pakub NR tarkvara, mis tuleb küll eraldi paigaldada, kuid andmed saadetakse samale juhtpaneelile teiste monitoormiskihtide statistikaga ning ei vaja eraldi haldamist.

Tarkvara on saadaval enamikele moodsatele Linuxi või Windowsi põhistele operatsioonisüsteemidele [25] ning kuna EMPISe server kasutab operatsioonisüsteemi, millel on kasutatav yum [26] pakkehaldussüsteem, siis kogu protsess koosnes neljast käsust [27] ning riistvara andmed hakkasid NR juhtpaneelile ilmuma kohe.

2.3.4 Paigaldamisel esinenud probleemid

Kuna serverid, mille peal EMPIS töötab, ei ole otseselt Nortal AS hallatavad, siis on enamik NR paigaldamise ning haldamise probleemidest bürokraatliku loomuga. Kui lokaalsetesse testkeskkondadesse sai NR tööle panna maksimaalselt ühe päevaga, siis välise teenusepakkuja korral läks selleks kordades rohkem aega – tuli koostada paigaldusjuhend, uurida täpselt välja, mis operatsioonisüsteeme teenusepakkujad kasutavad ja vastata küsimustele. Juba ainuüksi kirjavahetuse peale kulus väga arvestatav aeg. Lisaks selgus, et serverites kasutatav operatsioonisüsteem oli sedavõrd aegunud, et NR riistvarajälgimistarkvara seda ei toeta. Süsteemi uuendamine päästis valla uue bürokraatia-ahela, millele kulus veel mitu nädalat.

Seega monitoorimislahenduse valimisel tuleks tähepanu pöörata ka sellele, kui palju paigaldamis- ja haldamistegevusi võib jääda väljapoole arendaja enda mõjuulatust.

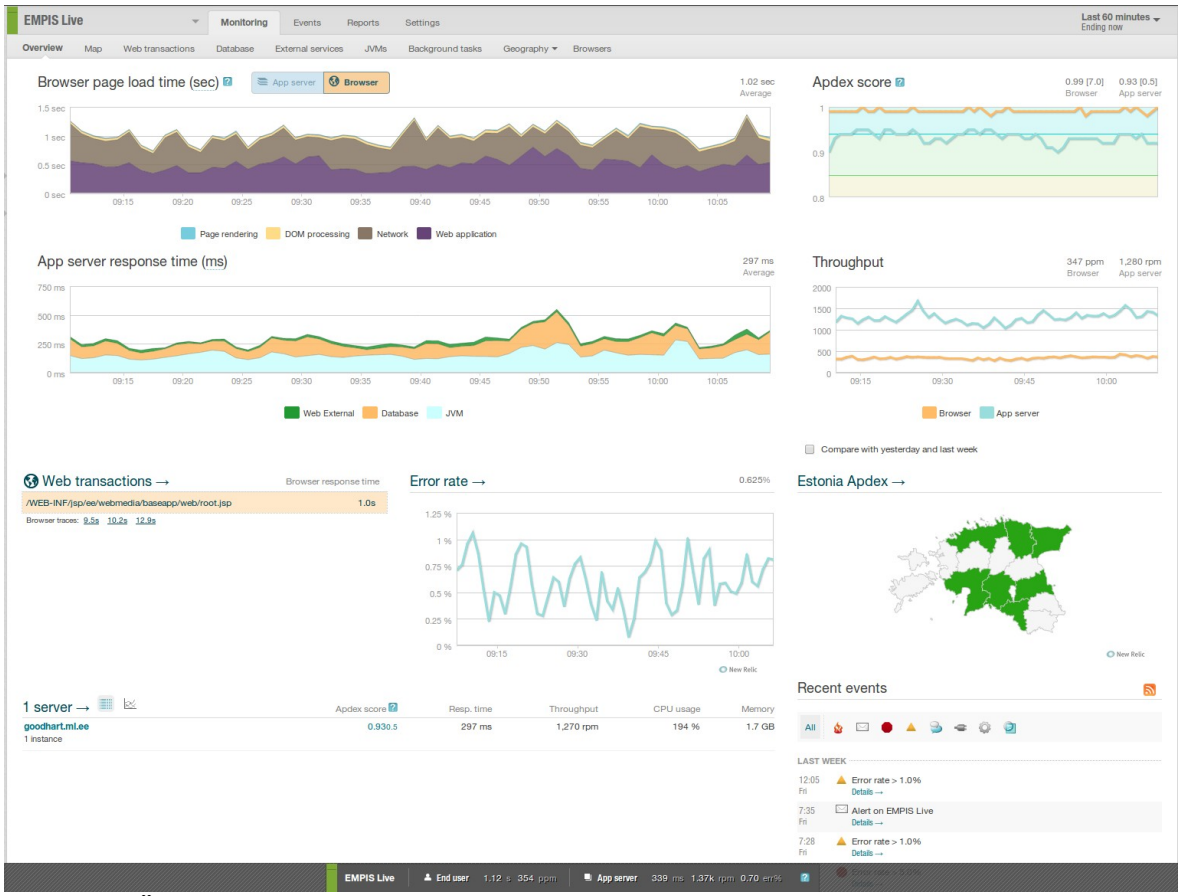
2.4 New Relicu kasutamine

NR näol on tegemist eelkõige statistika analüsaatoriga – mida suurem on andmete hulk, seda täpsemini saab selle põhjal leida süsteemi kõige nõrgemaid kohti. Tuleks vältida otsuste tegemist vaid testkeskkonna või vähese tööaktiivsusega perioodide kohta, kuna need ei pruugi väljendada reaalse olukorra probleeme. Kui on kogunenud arvestatav hulk informatsiooni, saab seda hakata süstemaatiliselt läbi käima.

Kõikides NR vaadetes on võimalik valida jälgitav ajavahemik vastavalt enda soovile ning vahemiku laius võib korraga ulatuda ühest minutist kuni kolme kuuni. Lisaks on vaadeldav aeg terve NR ülene seade, mis tähendab, et olles valinud kindla sündmuse ajavahemiku, on väga mugav kontrollida, mis igas rakenduse kihis sel hetkel toimus.

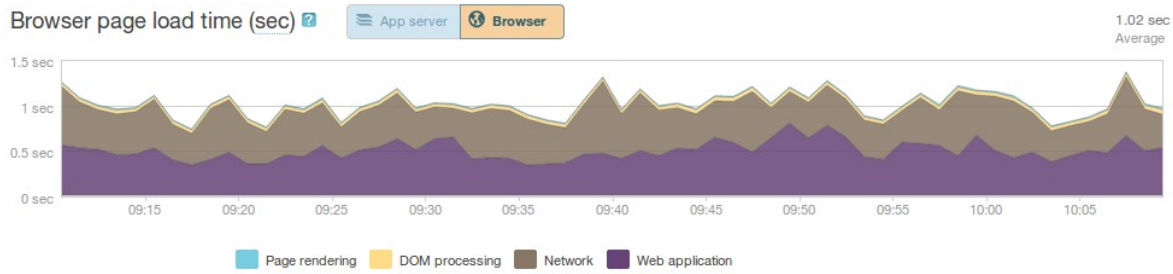
Järgnevalt tuuakse EMPISe näitel esile NR peamised tööriistad koos nende tutvustusega ning reaalsete andmete põhjal tehtud ekraanitõmmistega. Kogu NR funktsionaalsust antud töö maht katta ei võimalda ning seega on antud töös keskendutud kõige olulisemale. Tööriistade kuvad on väga intuitiivsed ning kokkuvõtlikud, kuid kuvatõmmise resolutsiooni tõttu antud töös raskesti loetavad. Tähtsamad ja huvitavamad osad tuuakse eraldi välja.

2.4.1 Ülevaade



Joonis 1: Ülevaade

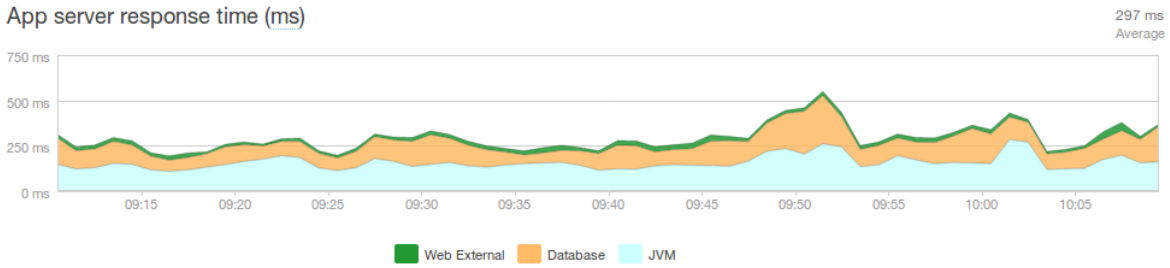
Ülevaade ehk *overview,* nähtaval joonisel 1, on esimene lehekülg, mida NR kasutaja kohtab. Siia on kokkuvõtvalt toodud kõik põhilised mõõdikud, et silmapilguga anda pilt sellest, mis olukorras jälgitav rakendus on. Kuna informatsiooni on palju, on see järgnevalt tükkhaaval läbi analüüsitud.



Joonis 2: Lehekülje laadimiskiirus

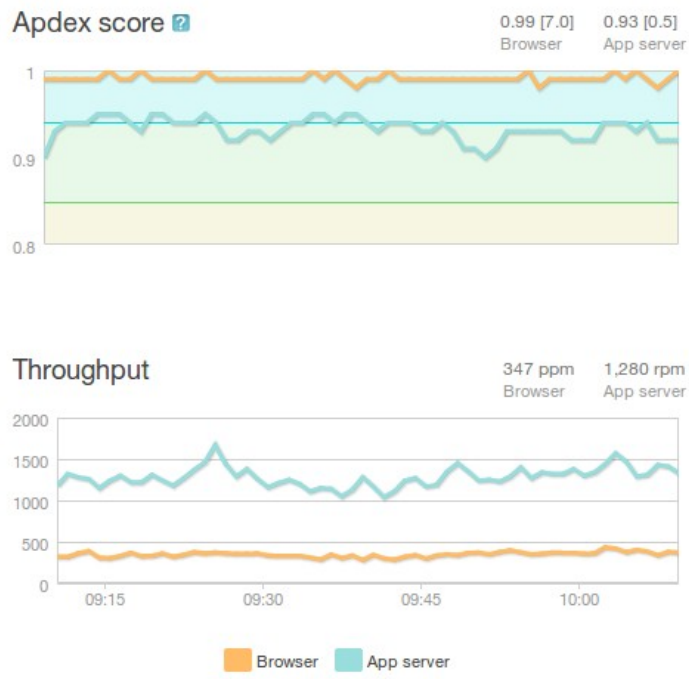
Browser page load time ehk veebilehe laadimise aeg näitab kihiliselt kõiki tegevusi, mis tehti antud lehe laadimiseks. Jooniselt 2 on näha, et EMPISe lehe keskmine laadimiskiirus

oli 1.02 sekundit ning ligi pool sellest on kulunud võrgukiiruse alla. Sellest saab teha olulise järelduse – klienti tuleb informeerida nende võrguprobleemidest, et saada arvestatav võit lehtede laadimisel. Positiivse külje pealt on näha, et väga vähe aega läheb lehekülje parsimise (*DOM processing*) ja piltide laadimise ning Javascripti käivitamise (*page rendering*) peale.



Joonis 3: Rakendusserveri kosteaeg

App server response time ehk rakendusserveri kosteaeg näitab, kui kaua läks rakendusel aega ühe käsu täitmiseks ning vastuse andmiseks. Selle mõõdiku järgi on võimalik hinnata kirjutatud koodi optimaalsust. Joonise 3 järgi paistab, et jõudlus on jaotatud andmebaasi ning Java koodi vahel üsna võrdselt ning kosteaeg on ühtlaselt madal – keskmiselt 297 millisekundit.



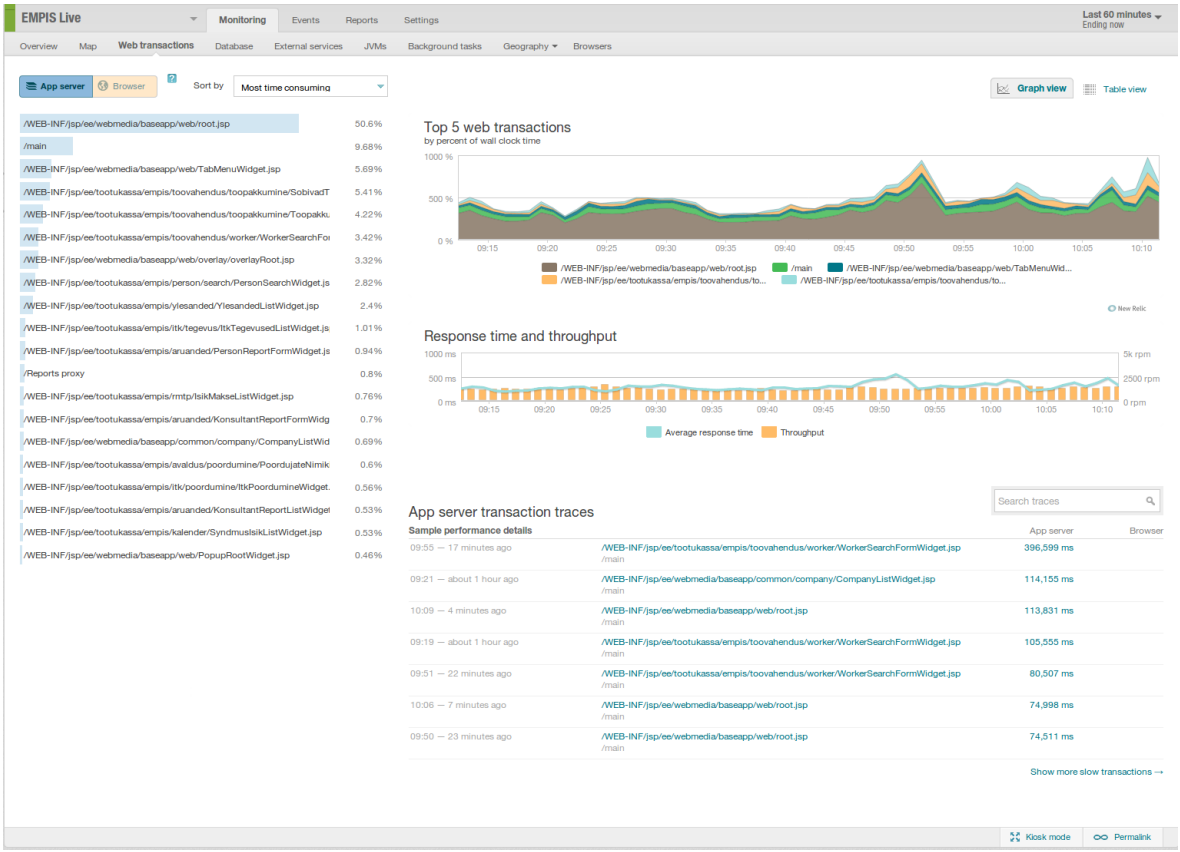
Joonis 4: Apdex & Läbilase

Apdex (*application performance index)* [28], nähtaval joonis 4 ülemise graafikuna, on mõõdik, mis arvutab kasutaja määratud ajapiiride järgi välja, kas klientide keskmine teenindamiskiirus on adekvaatne.

Throughput ehk läbilase, kujutatud joonis 4 alumisel graafikuna, näitab kahte mõõdikut: mitu lehekülge on minutis laetud (*pages per minute*) ning mitu päringut on üldse serverile saadetud (*requests per minute*). See annab ülevaate kasutajate hulgast ning aktiivsusest.

Ülevaate leheküljel on veel lisaks välja toodud nimekiri kõige aeglasematest transaktsioonidest süsteemis, vigade sagedus, serveri koormus, geograafiline kaart ning viimased hoiatused ja teadaanded.

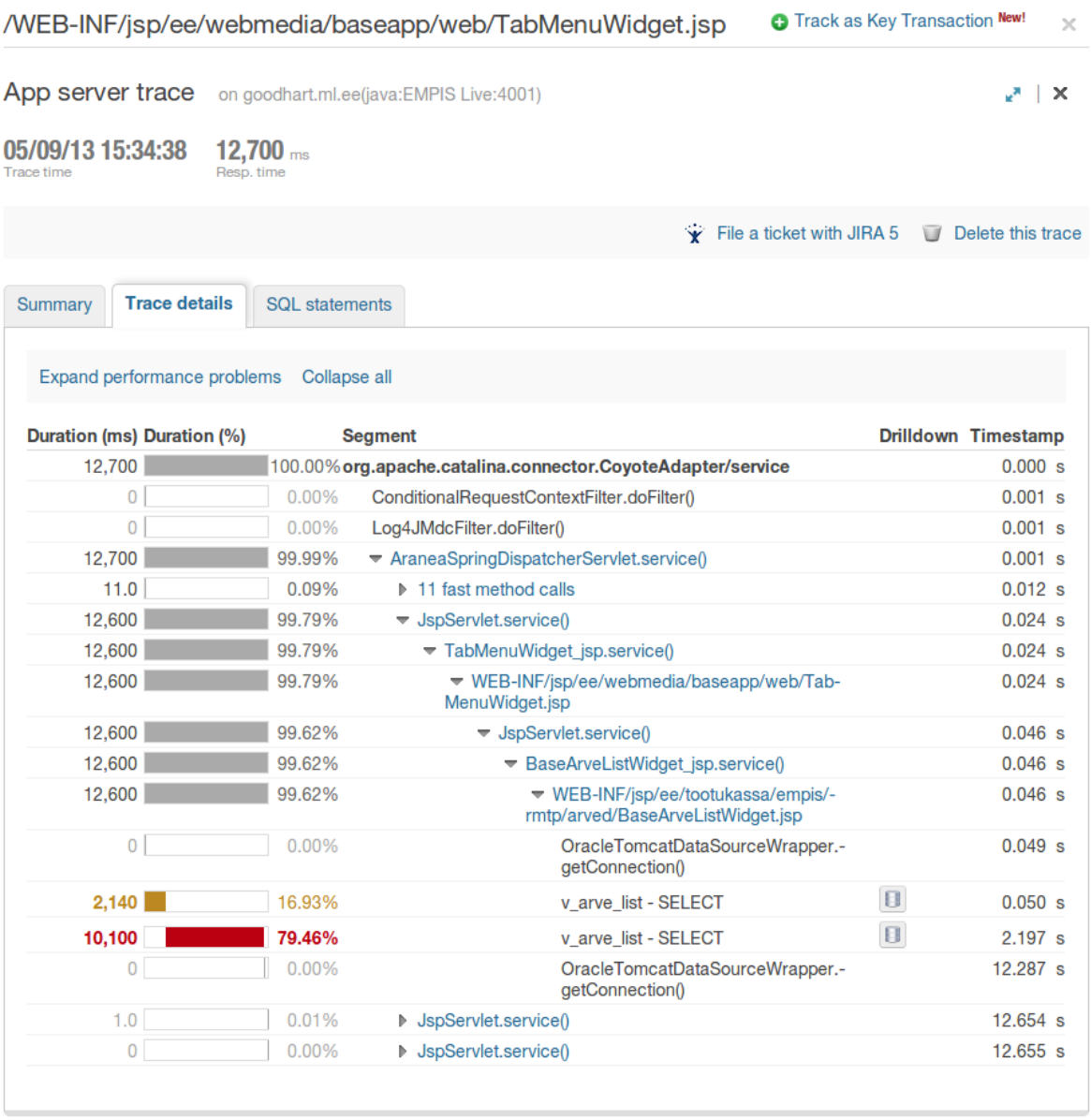
2.4.2 Veebitransaktsioonid



Joonis 5: Veebitransaktsioonid

Web transactions ehk veebitransaktsioonide vaates on ära toodud kõikide käivitatud transaktsioonide läbilõige (nähtaval joonis 5 vasakpoolse tulba ning graafikuna), andmaks ülevaadet selle kohta, mis osi süsteemist kõige tihedamini kasutatakse ning kui kaua võtab neil aega, et päringutele vastata. Kuna andmeid kogutakse otse Tomcatist ning see tegeleb peamiselt jsp-formaadis lehtede kuvamisega, siis NR arvutab veebitransaktsioone jsp-lehtede nimede järgi. EMPISes suunatakse aga väga suur osa liiklusest läbi ühe pealehe (`root.jsp`), ning seetõttu on selle konkreetse vaate tulemused tugevalt kallutatud. On näha, et üle 50% transaktsioonidest on NR arvates pärit ühest allikast. Et veebitransaktsioonide vaatest rohkem informatsiooni saada, tuleb Java koodi instrumenteerida NR API abil, millest tuleb juttu hilisemas peatükis.

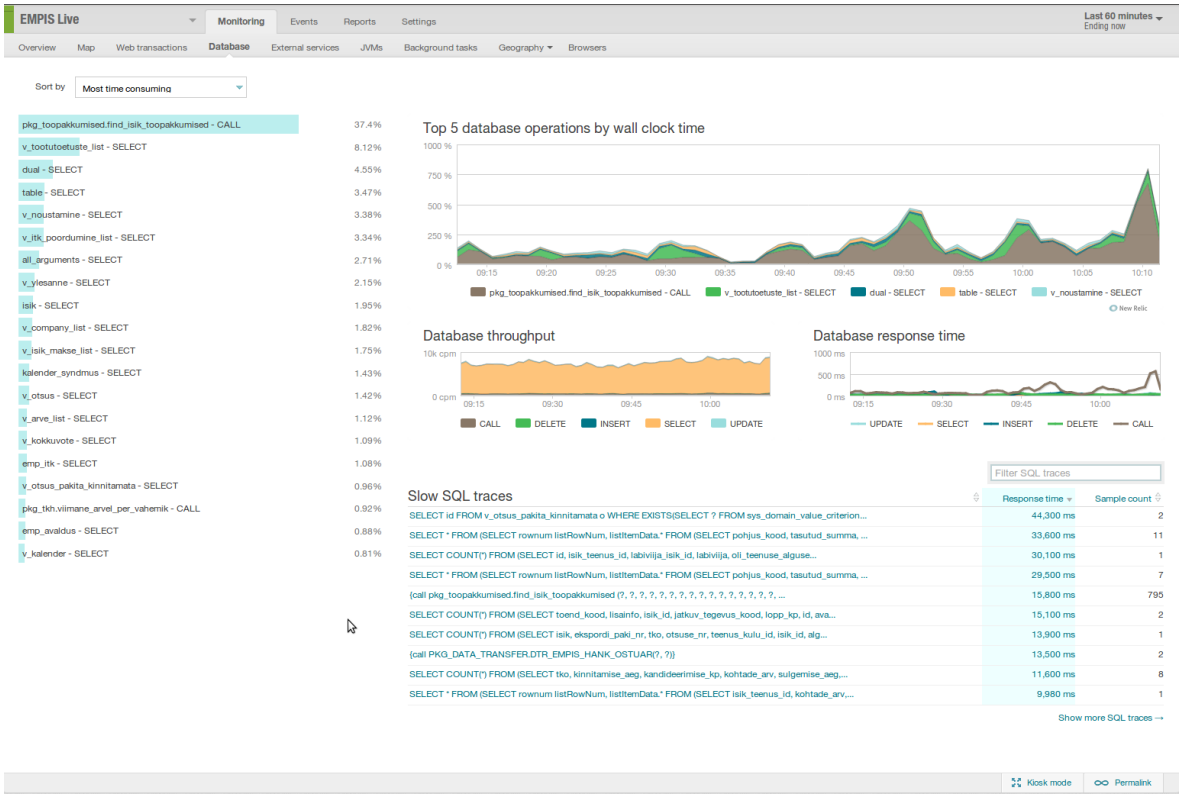
Siin vaatel on aga üks tööriist, mida saab objektiivselt kasutada ilma instrumenteerimiseta – aeglase transaktsiooni jälitus (*transaction trace*).



Joonis 6: Transaktsiooni jälitus

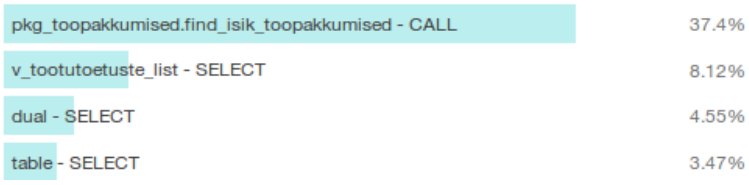
Kui NR näeb, et transaktsioon võtab erakordselt kaua aega, käivitatakse automaatselt transaktsiooni jälitus [29], mille järgi on hiljem lihtne leida, kuhu täpselt päringu aeg kulutati. Jooniselt 6 ilmneb, et jsp-leheküljelt nimega `TabMenuWidget` välja kutsutud päring võttis aega 12,7 sekundit ning 96,39% sellest ajast kulus `v_arve_list` andmebaasivaate kahele väljakutsele. Ilma monitoorimistööriistata oleks sellise informatsiooni kogumine võinud võtta tunde.

2.4.3 Suhtlus andmebaasiga



Joonis 7: Suhtlus andmebaasiga

Andmebaasi vaade, kujutatud joonisel 7, pakub hulgaliselt informatsiooni kõige aeglasemate ning enim kasutatavate päringute kohta, järjestades need kokkuvõtvalt kulutatud aja järgi.



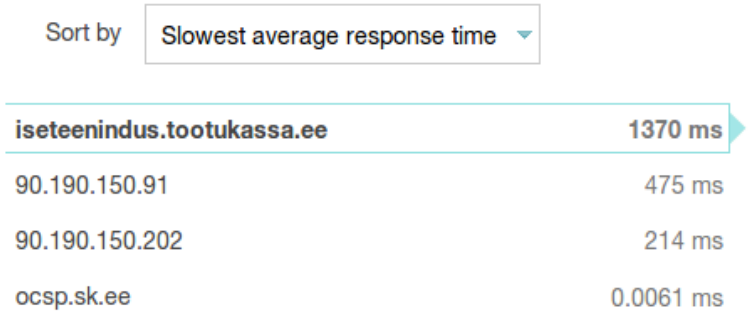
Joonis 8: Aeganõudvamad päringud

Jooniselt 8 võib näha, et üle kolmandiku EMPISe andmebaasi ajast läheb ainult ühe protseduuri, `pkg_toopakkumised.find_isik_toopakkumised`, väljakutsete peale – teisisõnu seda päringut optimeerides on meil lootus saada väga oluline ajavõit.

Sarnaselt veebitransaktsioonide vaatele jälitatakse ka aeglaseid andmebaasipäringuid ehk logitakse terve päring. See teeb tunduvalt lihtsamaks päringute optimeerimise, kuna on täpselt näha, mis parameetritega ning mis kujul aeglane päring kokku pandud oli.

2.4.4 Välised teenused

NR jälgib kogu suhtlust, mis toimub serveriväliste ühenduste kaudu ning mõõdab nende ühenduste kiirust. See võimaldab väga hõlpsasti kindlaks teha, kas ja kui tihti me väliste teenuste vastuste taga ootame .

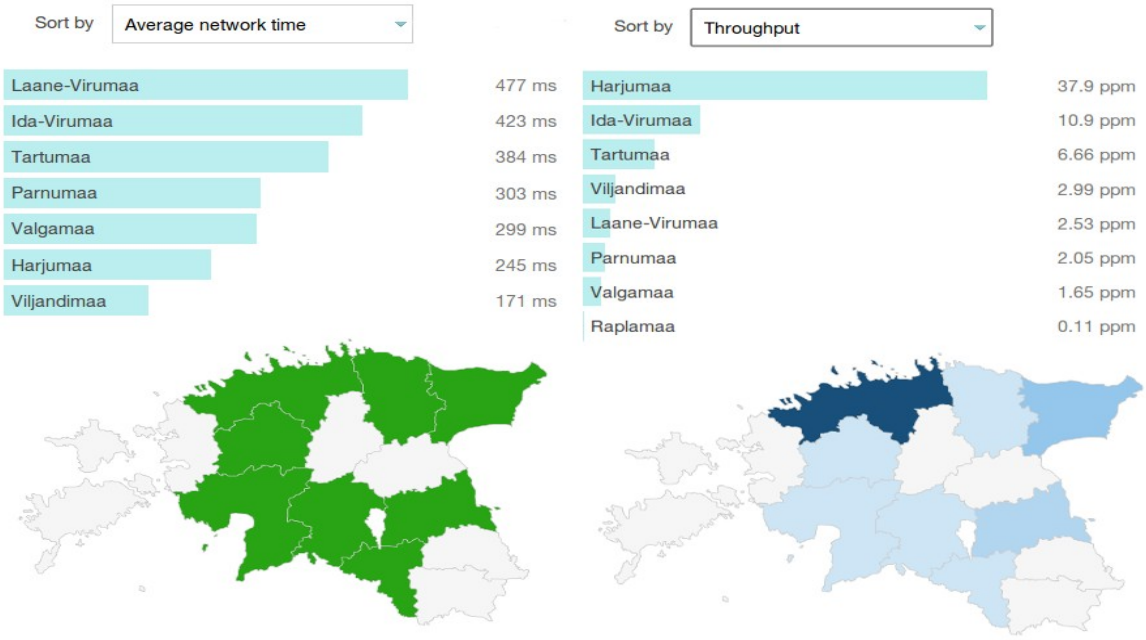


Joonis 9: Välised teenused

Jooniselt 9 selgub, et keskmiselt kõige kauem ootab EMPIS vastust aadressilt iseteenindus.tootukassa.ee. Selle informatsiooniga varustatult saab kas ühendust võtta iseteeninduse rakenduse meeskonnaga või teavitada enda kliente potentsiaalsest aeglustustest teenustel, mis selle aadressiga suhtlevad.

2.4.5 Geograafiline kaart

Kuna NR jälgib klientide IP-aadresse, oskab ta kogutud monitoorimisandmeid seostada geograafiliste asukohtadega ning seda teavet eri kategooriates kas globaalsel või riiklikul tasandil kaardina kuvada. EMPISe kontekstis annab see haruldase võimaluse näiteks võrgukiirust ning kasutajate hulka maakonniti võrrelda.



Joonis 10: Kaart

Jooniselt 10 on näha, et Harjumaa on kõige suurema kasutajate hulgaga maakond ning selle võrgukiirus on üks parimatest Eestis. Seevastu Ida-Virumaa ühendusega võib tekkida probleeme, kuna kõrge kasutajate hulk on kombineerunud üsna pika võrgu viivitusega, mis võib suure päringute hulga juures kasutajatele problemaatiliseks osutuda.

2.5 New Relicu kohandamine enda rakendusele ja vajadustele

Kogu eelnevas peatükis välja toodud funktsionaalsus on saavutatav vaikimisi NR paigaldusega. Et aga jälgimistööriistast viimast võtta, on järgnevalt toodud mõned kasulikud lisategevused.

2.5.1 Java instrumenteerimine

Kuna NR ei tea, millised rakenduse osad on kriitilise monitoorimisvajadusega ning millised mitte, siis vaikimisi konfiguratsiooniga võib statistika olla liiga üldsõnaline, et selle põhjal otsuseid teha või vigu leida. Heaks näiteks sellest fenomenist on alampeatükis 2.4.2 leitud `root.jsp` rohke kasutus, mis ei anna tegelikult mingit kasulikku informatsiooni.

Et detailsemaid andmeid saada, tuleb kasutada NR pakutud rakendusliidest (*API*), mille

abil on võimalik eraldi meetodeid selliselt ära märgistada, et nendest läbi käiv transaktsioon monitooringus alati eraldi välja tuuakse [30]. Märgistamisega peab olema aga ettevaatlik, kuna saadetava informatsiooni hulga suurendamisel kasvab ka monitoorimise ressursikasutus. Mõistlik on eraldi välja tuua kõige veaohtlikumad kohad ning aeg-ajalt kõik märgistatud kohad üle vaadata, et asjatut ressursikulu vältida.

Siiani on EMPISes eraldi jälgimise alla võetud vaid mõned kriitilisemad kohad, mis tegelevad failide kõvakettale kirjutamise ja lugemisega. Aegamööda ning ettevaatlikult on plaanis jälgitavate protsesside arvu suurendada.

2.5.2 Rakenduse vigade jälgimine

Vaikimisi oskab NR püüda ning analüüsida vigu, mis ei ole süsteemi töödeldud – näiteks puuduvad ressursid, katkised lingid jne. Sarnased vead grupeeritakse ning näidatakse kasutajale sageduse järjestuses. EMPISes aga on enamik vigu rakenduse töödeldud, et süsteemi töö ei katkeks – klient saab vea otse süsteemist raporteerida ning tal on võimalik koheselt edasi töötada. Et NR ka selliseid vigu oma statistikas arvestada oskaks, tuleb need taaskord rakenduseliidese abil eraldi välja tuua. EMPISe puhul oli see väga lihtne – kuna meil on tsentraalne erinditöötluspunkt, siis oli vaid üks koht, kus tarvis koodimuudatust teha ning NR liides välja kutsuda.

2.5.3 Sidumine ülesandehaldussüsteemiga

EMPISe meeskond kasutab igapäevatöös tööülesannete jagamiseks ja haldamiseks vahendit nimega Jira [31] ning selgus, et NRst on Jirasse võimalik otse tööülesandeid tekitada – kasutajakonto seadetes on alampunkt *integrations* (integratsioonid), kuhu sisestada oma Jira kasutaja andmed ning valida projekt, mille alla ülesanded lisatakse. Ülesannete genereerimist toetab nii peatükis 2.4.2 välja toodud transaktsioonide jälitaja kui ka eelmises punktis nimetatud vigade vaade. Selline integreerimine suurendab tunduvalt tõenäosust, et kõik optimeerimisprobleemid ning vead jõuavad arendajateni, kuna üks hiireklikk võtab tunduvalt vähem aega kui uue ülesande manuaalne loomine.

2.5.4 Paigalduste jälgimine

Ülimalt oluline on omada ülevaadet sellest, millised muudatused tulid kaasa uue versiooni toodangusse paigaldamisega. NRl on sisseehitatud tööriistad erinevate versioonide võrdlemiseks, kuid vaikimisi puudub tal informatsioon selle kohta, millal vahetus toimus. Hetkel on ainus võimalus seda teavet monitoorimisvahendile edastada nii, et teha väljakutse otse vastu NR agenti ehk vastu `newrelic.jar` faili järgmiste parameetritega:

```
java -jar newrelic.jar deployment –revision={versiooni number}
```

Kuna EMPISe puhul on toodanguversiooni server sisse ostetud teenus, siis on väga keeruline antud teavitust automatiseerida – ei ole lihtsalt võimalik `newrelic.jar`-le ligi pääseda. Tegemist on siiski aga väga olulise featuuriga, nii et kuni parema lahenduse leidmiseni on mainitud väljakutset tehtud manuaalselt.

2.6 Juurutamine tööprotsessi

Paigaldamise ning konfigureerimise kõrval ei tohi unustada monitoorimistarkvara tõenäoliselt kõige olulisemat osa – selle juurutamist meeskonda ning tööprotsessidesse. Maksimaalse kasuteguri välja toomiseks peaks monitoorimine saama kõigi meeskonnaliikmete hariliku tööprotsessi osaks.

2.6.1 Kes peaks tegelema monitoorimistarkvara kasutamisega?

Monitooringu jälgimine ei tohiks kindlasti saada vaid ühe töötaja ülesandeks – kasulik on monitoorimisvahend põhjalikult selgeks teha kõigile meeskonna liikmetele. Arendajad ning testijad, kes on rakenduse sisemusega põhjalikult tuttavad, leiavad tehnilisi kitsaskohti; analüütikud näevad, kuidas nende planeeritud muudatused on rakendusele mõjunud ja projektijuht saab ülevaatlikku informatsiooni, mida kliendiga arutada. Mida rohkem informeeritud silmapaare monitooringut igapäevaselt kasutab, seda täpsem on kõigi projekti liikmete arusaam sellest, mis seisus nende rakendus parajasti on.

2.6.2 Monitooringuvahendi kasutamise sagedus

Lihtne on monitoorimisrakendust näha kui vahendit, mille poole pöördutakse vaid vigade põhjuste otsimiseks, kuid tegelikkuses on see vaid väike osa kogu monitoorimise potentsiaalist. EMPISe kogemusel peaks monitooringuvahendit külastama minimaalselt kord päevas, et hoida end kursis rakenduse käekäiguga. Lisaks on mõistlik määrata iga nädal muutuv korrapidaja, kes käib vastava nädala igal õhtul kõik tähtsamad mõõdikud ning vaated süstemaatiliselt läbi. Sellist graafikut tööprotsessi sisse harjutades väheneb tõenäosus, et klient peab vigu või rakenduse aeglust ise raporteerima – pideva jälgimise teel saab meeskond varakult ohtlikest olukordadest aimu ning jõuab need enne kriitiliseks muutumist parandada.

2.6.3 – Vahendi ajakohastamine

Oluline on hoida monitoorimisvahendit ühes tempos jälgitava rakenduse arendamisega. Peatükis 2.5.1 välja toodud instrumenteerimisega peaksid arendajad regulaarselt tegelema, et NRs oleks alati nähtaval optimaalsed tulemused – kui rakenduse uut kriitilist funktsionaalsust ei ole monitoorimisvahendis eraldi välja toodud, kaotab vahend aegamööda efektiivsust.

Kriitiline on ka hoida silm peal toote uutel versioonidel. Kuna NR ei ole operatsioonisüsteemi installeeritud, vaid fail rakendusserveri kaustas, siis ei ole sellel võimalik ennast automaatselt uuendada ning seda tuleb teha manuaalselt.

2.7 Tulemused ja järeldused

Kuigi NR on äärmiselt kergekaaluline rakendus, leidub mitmeid nüansse, mida selle kasutuselevõtul tasub silmas pidada .

Nagu peatükis 2.3.4 mainitud, toimub NR paigaldamise protsess väga kiiresti seni, kuni paigaldajal on otsene ligipääs jälgitavale serverile – vastasel juhul kulub väga oluline aeg serveri haldajaga läbirääkimistele ning tema instrueerimisele. Sama probleem kerkib esile siis, kui on tarvis toote versiooni uuendada. Õnneks on NR konfiguratsiooni haldamiseks kõik vajalik saadaval nende enda kasutajaliideses ning ei puuduta reaalseid faile.

Täielikult paigaldatud rakenduse kasutegurist – kuigi EMPISe monitoorimine on alles algjärgus, on juba mitmest jälgimiskihist välja tulnud äärmiselt kasulikku informatsiooni. Mõned näited:

- Leiti tõenäoline põhjendus sellele, miks Ida- ja Lääne-Virumaalt on kõige rohkem rakenduse aeglust kurtvaid teateid saabunud – NRst paistab, et nendes maakondades on kõige pikemad EMPISest sõltumatud võrguviivitused.
- Keskmiselt pool lõppkasutaja lehekülje laadimise ajast võtab enda alla võrguliiklus – sellest ning eelmisest punktist saab järeldada, et olulist võitu süsteemi kiiruses annaks Töötukassa sisevõrgu optimeerimine.
- Avastati, et üks andmebaasipäring võtab enda alla üle kolmandiku kogu andmebaasiliiklusest – see on suurepärane võimalus optimeerimise teel klientidele väärtuslikke sekundeid võita.

Lisaks on kõigile meeskonna liikmetele saanud oluliselt selgemaks, kust kõige tõenäolisemalt aegluse ja vigade põhjused EMPISes pärinevad ning kuidas neile kiiresti jälile jõuda.

NR kohandamine EMPISe vajadustele on pidev protsess ning tuleb eraldi rõhutada, et lisaks esialgsele seadistamisele tuleb peatükis 2.6.3 kirjeldatud ajakohastamine sisestada regulaarsesse tööprotsessi, et see muutuks projektis harjumuseks.

Kõige suurem katsumus monitoorimislahenduse juurutamisel ongi seega mitte selle paigaldamine ega kasutama õppimine, vaid pidevasse töövoogu integreerimine. Kui õnnestub kogu meeskond korralikult seadistatud töövahendit regulaarselt kasutama saada, on saavutatud maksimaalne kasutegur.

Kokkuvõte

Käesoleva töö peamiseks eesmärgiks oli leida ja juurutada terviklik monitoorimislahendus Eesti Töötukassa infosüsteemile (EMPIS) ning hinnata kogu protsessi lõplikku kasutegurit.

Esmalt tutvustati üldisemalt monitoorimise mõistet ning selle kasulikkust, tuues välja erinevaid võimalikke jälgitavaid kihte ja nende jälgimisest saadavat tulu. Seejärel uuriti põgusalt erinevaid lahendusi, mida välja toodud kihtide monitooringuga katmiseks pakutakse.

Töö teises pooles võrreldi EMPISe monitoorimisvajadusi uuritud lahenduste pakutavaga ning valiti välja sobivaim toode – New Relic – misjärel analüüsiti läbi kogu juurutamise protsess: paigaldamine, seadistamine, kasutamine ning meeskonda integreerimine. Lõpuks arutati läbi tehtud töö tulemused ning toodi välja iga protsessi sammu mõtteainet pakkuvamad kohad.

Töö tulemusena on nüüd EMPISel reaalselt kasutusel olev, korralikult konfigureeritud ja projektile olulist lisaväärtust tootev NR teenus ning dokument, mis tervet juurutamise protsessi kirjeldab. Sellest dokumendist on loodetavasti kasu nii projektijuhtidele, kes klientidele monitooringu olulisust põhjendavad, kui ka üliõpilastele ning Nortali arendajatele, kellel on tarvis sarnane lahendus realiseerida, et oma veebirakenduste kohta objektiivset jõudlusinformatsiooni saada.

Kuivõrd käesolev uurimus suutis NR funktsionaalsusest katta vaid kõige põhilisema, on siin võimalus tööd edasi arendada spetsiifilisema lahenduse suunas. Töös mainiti korduvalt, et monitoorimisest maksimaalse kasu lõikamiseks peab süsteemi jälgimist nägema kui pidevalt arenevat protsessi, mitte ühekordset paigaldust ja kuna autor EMPISega igapäevaselt edasi tegeleb, oleks suurepärane võimalus tööd ka teoreetilises võtmes jätkata, püüdes leida optimaalset instrumenteerimislahendust või võrdlusmomendi huvides katsetada mõnd teist monitoorimisvahendit.

Implementing Web Application Monitoring on the Basis of the Estonian Employment Information System

Bachelor's thesis (6 EAP)

Mats Johanson

Summary

The Estonian Employment Information System (EMPIS) is a web application that has been in development for four years. As it grows, so does its complexity – each new feature brings an ever greater risk of regression, slowdowns and errors in the software. To combat this type of decay, a proper monitoring system had to be implemented.

The main purpose of the present Thesis was to explain the necessity of a good monitoring solution, find, deploy and configure a suitable monitoring solution for EMPIS and document the whole process.

The thesis is divided into two main parts:

1. The first section explains what application monitoring is, which application layers can be covered by it and how the data from each layer can be benefited from. It also compares some of the more widely recognised products on the market today.

2. The second section describes the actual process of adding a monitoring system to EMPIS and customizing it for the system's needs. The main use cases of New Relic are then covered with examples from the live version of EMPIS and are followed by an analysis of the process.

The final result of the thesis is a fully functional monitoring system and a document that explains the full process of implementation which can be used as a guideline by students or people in the IT sector who are considering whether or how to protect their software from loss of efficiency with monitoring.

Kasutatud kirjandus

1. Meelis Pavel, Uus töötukassa infosüsteem, http://www.tootukassa.ee/index.php?id=13297, [Viimati vaadatud 27.04.2013].
2. Larry Dragich, Prioritizing Gartner's APM Model, http://apmdigest.com/prioritizing-gartners-apm-model, [Viimati vaadatud 27.04.2013].
3. Jakob Nielsen, Response Times: The 3 Important Limits, http://www.nngroup.com/articles/response-times-3-important-limits/, [Viimati vaadatud 01.05.2013].
4. Microsoft Application Architecture Guide, 2nd Edition, http://msdn.microsoft.com/en-us/library/ee658127.aspx, [Viimati vaadatud 01.05.2013].
5. Who's Who in Application Performance Management for Virtualization and the Cloud, http://www.virtualizationpractice.com/whos-who-in-application-performance-management-for-virtualization-and-the-cloud-14336/ [Viimati vaadatud 02.05.2013].
6. A Comparison of Application Performance Management Suites from CA, HP and Oracle, http://www.oracle.com/us/products/enterprise-manager/crimson-apm-comp-400465.pdf, [Viimati vaadatud 27.04.2013].
7. APM Market Disruptors - AppDynamics and New Relic, http://www.bbramley.com/2012/01/apm-market-disruptors-appdynamics-and.html, [Viimati vaadatud 25.04.2013].
8. Google Analytics, http://www.google.com/analytics/, [Viimati vaadatud 25.04.2013].
9. Usage of traffic analysis tools for websites,

http://w3techs.com/technologies/overview/traffic_analysis/all [Viimati vaadatud 25.04.2013].

10. Google Analytics Premium, http://www.google.com/analytics/premium/faq.html#faq-cost, [Viimati vaadatud 25.04.2013].
11. Pingdom, https://www.pingdom.com/, [Viimati vaadatud 25.04.2013].
12. CA Technologies Application Performance Manager, http://www.ca.com/us/application-performance-management.aspx, [Viimati vaadatud 25.04.2013].
13. CA Acquires Privately Held Wily Technology, http://investor.ca.com/releasedetail.cfm?ReleaseID=316670, [Viimati vaadatud 25.04.2013].
14. Bernd Harzog, APM as a Service and CA's Reaction, http://www.virtualizationpractice.com/apm-as-a-service-19240/, [Viimati vaadatud 26.04.2013].
15. The Real Cost of Application Performance Management Ownership, http://www.appdynamics.com/blog/2011/09/02/the-real-cost-of-application-performance-management-apm-ownership/, [Viimati vaadatud 26.04.2013].
16. New Relic, http://newrelic.com/, [Viimati vaadatud 26.04.2013].
17. Lewis Cirne: Executive Profile & Biography, http://investing.businessweek.com/research/stocks/private/person.asp?personId=471247&privcapId=44316034&previousCapId=44316034&previousTitle=New%20Relic,%20Inc., [Viimati vaadatud 03.05.2013].
18. Nagios, http://www.nagios.org/, [Viimati vaadatud 28.04.2013].
19. NetSaint (Nagios) Changelog http://web.archive.org/web/20060501150621/http://www.netsaint.org/changelog.php, [Viimati vaadatud 28.04.2013].

20. The Nagios Setup Explained, http://www.linuxforu.com/2011/07/nagios-setup-guide/, [Viimati vaadatud 28.04.2013].

21. Apache Tomcat, http://tomcat.apache.org/, [Viimati vaadatud 04.05.2013].

22. Real User Monitoring in New Relic, https://newrelic.com/docs/features/real-user-monitoring, [Viimati vaadatud 04.05.2013].

23. What is a JSP page?, http://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html, [Viimati vaadatud 04.05.2013].

24. Enabling Real User Monitoring in Java, https://newrelic.com/docs/java/real-user-monitoring-in-java, [Viimati vaadatud 04.05.2013].

25. New Relic for Server Monitoring, https://newrelic.com/docs/server/new-relic-for-server-monitoring, [Viimati vaadatud 04.05.2013].

26. Yum, http://yum.baseurl.org/, [Viimati vaadatud 04.05.2013].

27. New Relic, Server Monitor Installation for Yum, https://newrelic.com/docs/server/server-monitor-installation-redhat-and-centos#yum, [Viimati vaadatud 04.05.2013].

28. Apdex, http://apdex.org/, [Viimati vaadatud 04.05.2013].

29. Transaction Traces, https://newrelic.com/docs/features/transaction-traces, [Viimati vaadatud 04.05.2013].

30. Custom Instrumentation with the Java Agent, https://newrelic.com/docs/java/custom-instrumentation-with-the-java-agent, [Viimati vaadatud 04.05.2013].

31. Jira, http://www.atlassian.com/software/jira/overview, [Viimati vaadatud 05.05.2013].

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Mats Johanson

(sünnikuupäev: 11.03.1988)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Veebirakenduse monitoorimislahenduse realiseerimine Eesti Töötukassa infosüsteemi näitel,

mille juhendajad on Priit Liivak ja Helle Hein,

1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **13.05.2013**