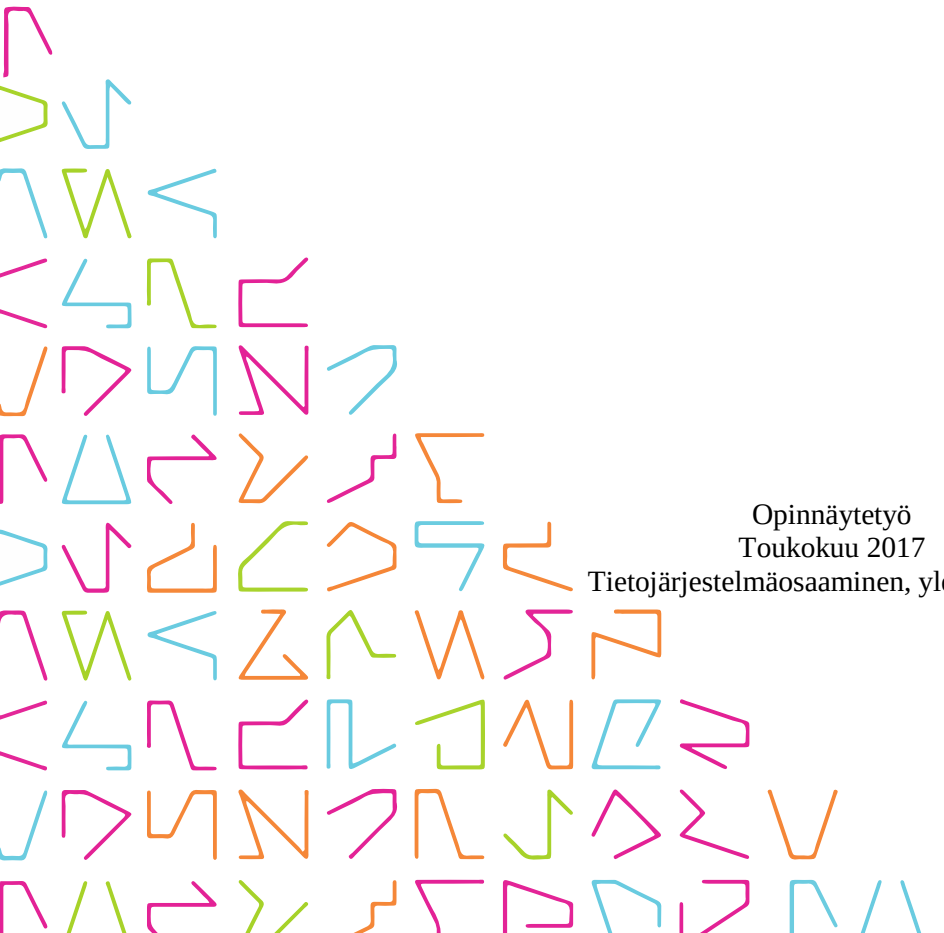
TAMPEREEN
AMMATTIKORKEAKOULU

MAGENTO-SOVELLUSPINON SUORITUSKYVYN KEHITTÄMINEN SUORITUSKYKYTESTAUKSEN AVULLA

Mika Mäkelä



Opinnäytetyö
Toukokuu 2017
Tietojärjestelmäosaaminen, ylempi AMK

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojärjestelmäosaaminen, ylempi AMK

MÄKELÄ, MIKA:
Magento-sovelluspinon suorituskyvyn kehittäminen suorituskykytestauksen avulla

Opinnäytetyö 55 sivua
Toukokuu 2017

Opinnäytetyön tavoitteena oli mitata ja parantaa Magento-sovelluspinon suorituskykyä suorituskykytestauksen avulla. Tarkoituksena oli ensin suunnitella ja toteuttaa ratkaisu Magento-sovelluspinon suorituskykytestaukselle, joka on helposti käyttöönotettava, muokattava ja toistettava. Toteutettua testausratkaisua hyödynnettiin Magento-sovelluspinon suorituskykyongelmien tunnistamiseen ja suorituskyvyn parantamiseen. Pääasiallisena kehittämismenetelmänä toimi suorituskykytestaus ja sen tuloksena syntynyt uusi tieto. Työn toimeksiantaja oli Ambientia Oy, jonka visiona on tarjota Suomen nopeinta ja luotettavinta Magento-verkkokauppaa.

Magento-sovelluspinon suorituskykytestaukseen kehitetty ratkaisu vastasi sille asetettuihin tavoitteisiin. Ratkaisussa käytettiin hyväksi valmiita avoimen lähdekoodin työkaluja, kuten JMeter-testaustyökalua, nopeuttamaan suorituskykytestausta. JMeter-testaustyökalun valintaa tuki myös sille löytynyt valmis Magenton testausprofiili. Näin ollen Magenton käyttötapauksia ei tarvinnut rakentaa itse. Aika voitiin sen sijaan hyödyntää suorituskykytestauksen suunnitteluun ja testausympäristön rakentamiseen. Kehitetty ratkaisu on laajasti muokattavissa, sillä sen rakentamisessa hyödynnettiin avoimen lähdekoodin työkaluja. Toistettavuus saavutettiin kehittämällä testiautomaatiota mahdollisimman pitkälle.

Suorituskykytestauksesta saatujen tulosten avulla Magento-sovelluspinon suorituskykyä saatiin parannettua. Se tapahtui valitsemalla Magento-sovelluspinoon ne sovelluskomponentit, joilla saavutettiin paras suorituskyky. Samalla ratkaistiin nykyisen alustan suorituskykyongelmat, joita ei ilman suorituskykytestausta olisi todennäköisesti löytynyt. Suorituskyvyn parantuminen vaikutti positiivisesti Magento-verkkokauppojen käytettävyyteen ja tätä myötä myös Ambientian asiakkaiden verkkokauppaliiketoimintaan. Suorituskykytestausprosessin tuominen osaksi Magento-sovelluspinon muutostenhallintaa, auttoi parantamaan kehityksen laatua ja vähentämään suorituskykyyn liittyviä riskejä.

Suorituskykytestauksesta saatujen tulosten profilointi sovelluskomponenttien mukaan ei toteutunut suunnitellusti. Jatkokehityksenä suunniteltiin sovellustason suorituskykymonitoroinnin käyttöönottoa osana suorituskykytestausta. Suorituskykytestauksesta saadut vasteajan mittaukset eivät täysin vastanneet loppukäyttäjän kokemusta. Tämä johtui siitä, että JMeter-testaustyökalu ei suorittanut sivupyyntöjä samalla tavalla kuin käyttäjän selain. Havaituista puutteista johtuen opinnäytetyössä pohdittiin sitä, miten reaaliaikainen suorituskykymonitorointi voisi mahdollisesti korvata erilliset suorituskykytestaukset.

Asiasanat: verkkokauppa, magento, suorituskyky, suorituskykytestaus, sovelluspino

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree Programme in Information System Competence

MÄKELÄ, MIKA:
Improving the Performance of Magento Software Stack by Performance Testing

Master's thesis 55 pages
May 2017

The purpose of this thesis was to measure and improve the performance of the Magento software stack by using performance testing methods. To achieve this, a performance test solution for the software stack needed to be designed and implemented. The requirements for the performance test solution were ease of implementation, modifiability and repeatability. The thesis was commissioned by Ambientia Ltd, whose vision is to offer the fastest and most reliable Magento hosting in Finland.

Ease of implementation was achieved by using ready-made performance testing tools. The JMeter performance testing tool was chosen because it already had a testing profile available for Magento. A lot of time was saved by not needing to program the test cases manually. Instead, the time was used to plan the performance test scenarios and build the test environment. JMeter also offered greater modifiability as it is open source software. Partial test automation was developed to increase repeatability.

By modifying the Magento software stack based on the performance tests results, the performance was increased substantially. The results were also used to fix performance bottlenecks, which most likely would not have been discovered otherwise. The increase of performance had a positive effect on Magento's usability, which in turn helped the e-commerce business of Ambientia's customers. As further development, application performance monitoring was planned to be added as part of the performance test solution.

Key words: e-commerce, magento, performance, performance testing, software stack

SISÄLLYS

LYHENTEET JA TERMIT

APM	Application Performance Monitoring
CAMS	Culture, Automation, Measurement, Sharing
CI/CD	Continuous Integration/Deployment
CDN	Content Delivery Network
DCS	Data Centre Services
DOM	Document Object Model
ESI	Edge Side Includes
FPC	Full Page Cache
FPM	FastCGI Process Manager
LAMP	Linux, Apache, MySQL, PHP/Perl/Python
RHEL	Red Hat Enterprise Linux
RHSCL	Red Hat Software Collections
RUM	Real User Measurements
SLA	Service Level Agreement
KPI	Key Performance Indicator
TTFB	Time to First Byte
TTLB	Time to Last byte

ESIPUHE

Aluksi haluan kiittää Ville Törhöstä ja Ossi Hakkaraista, jotka ovat olleet mukana kehittämistehtävään liittyvien suorituskykytestien suorittamisessa ja niiden määrittelyssä.

1 JOHDANTO

Verkkopalveluiden suorituskyky tulee huomioida, sillä se on vahvasti yhteydessä palvelun käytettävyyteen. Käytettävyysguru Jakob Nielsen (2010) onkin listannut suorituskyvyn yhdeksi tärkeimmistä asioista, jotka tulee verkkopalvelun käytettävyydessä huomioida. Nielsenin (2011) mukaan syyt tähän ovat inhimillisiä, sillä meidän huomiointikykymme ja lähimuistimme ovat rajallisia. Jos verkkopalvelu hidastelee, niin vaarana on, että käyttäjä ei jaksa odottaa sivujen latausta ja poistuu palvelusta. Kun kyseessä on verkkokauppa, niin se tarkoittaa potentiaalisen asiakkaan menettämistä. Jokainen hävitty asiakkuus tietää menetettyä rahaa, joka taas on pois verkkokaupan liikevaihdosta. Verkkokaupan liikevaihtoon voidaan pyrkiä vaikuttamaan esimerkiksi markkinoinnin avulla, mutta suorituskyky on yksi harvoista asioista, johon voidaan vaikuttaa puhtaasti teknisillä ratkaisuilla. Mutta ennen kuin suorituskykyä voidaan parantaa, tulee sitä pystyä ensin mittaamaan. Ongelman ratkaisuun on olemassa ohjelmistotestaukseen liittyvä menetelmä, joka tunnetaan nimellä suorituskykytestaus. Sen avulla voidaan systemaattisesti ja luotettavasti todistaa suorituskyvyn muutokset. Suorituskykytestaus vaatii kuitenkin suunnittelua, jotta sen avulla saadaan hyödyllisiä tuloksia.

Tämän opinnäytetyön aiheena on Magento-verkkokaupan suorituskyvyn kehittäminen. Opinnäytetyön aihe on syntynyt toimeksiantajan liiketoimintastrategian pohjalta. Strategiassa on listattu Magento-verkkokauppojen nopeus yhdeksi merkittävimmistä kehityskohteista. Magenton suorituskyvyn kasvattaminen tuo lisäarvoa verkkokauppojen liiketoimintaan parantamalla verkkokaupan käyttökokemusta. Se tuo myös kilpailuetua niillä markkinoilla, joilla kilpailu on kovaa. Kehittämisen kohteena on tarkemmin Magenton sovelluspino ja pääasiallisena kehittämismenetelmänä on suorituskykytestaus. Jotta suorituskyvyn kehittäminen olisi mahdollista, suorituskykytestaus tulee ensin suunnitella ja rakentaa. Suorituskykytestauksesta saatujen tulosten avulla on tarkoitus kehittää Magento-sovelluspinon suorituskykyä. Käytännössä tämä tapahtuu teknisillä toimenpiteillä, joilla muutetaan Magento-sovelluspinon rakennetta.

2 TAUSTA

Tässä luvussa kuvataan aihealueen tärkeimmät sidosryhmät, kuten toimeksiantajan organisaatio. Tarkoitus on myös avata opinnäytetyössä käytettyä terminologiaa, jotta se on lukijan helpommin omaksuttavissa. Tavoitteena on, että lukija saa paremman kuvan toimintaympäristöstä, johon opinnäytetyön aihealue liittyy.

2.1 Toimeksiantaja

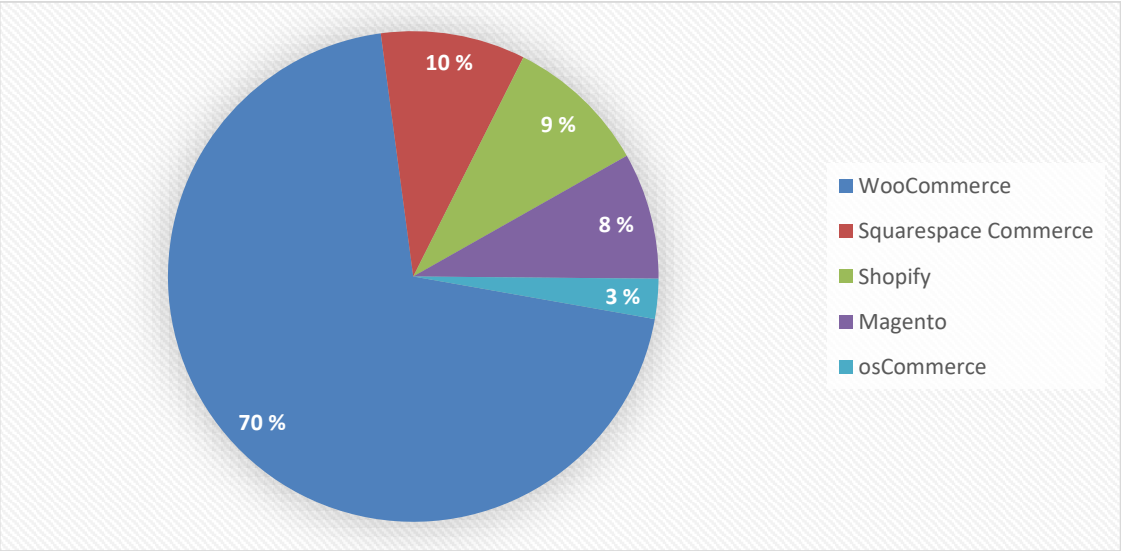
Työn toimeksiantajana toimii Ambientia, joka on vuonna 1996 perustettu IT-alan yritys. Sen palveluihin kuuluvat muun muassa erilaiset portaaliratkaisut, ketterä ohjelmistokehitys ja palvelumuotoilu. Ambientian tavoite on tehostaa asiakkaiden liiketoimintaa digitalisoimalla palveluita ja prosesseja. Ambientian erottaa muista alan toimijoista se, että sieltä asiakkaat saavat kokonaisvaltaista palvelua. Verkkokauppoihin liittyvä liiketoiminta on tullut mukaan tällä vuosikymmenellä ja siitä on syntynyt yksi yrityksen strategisesti merkittävimpiä liiketoiminta-alueita. Liiketoiminta-alueen henkilömäärä on kasvanut nopeasti rekrytointien, yritysostojen ja fuusioiden avulla. Vuonna 2015 Ambientia hankki osake-enemmistön Suomessa ja Virossa toimivasta Insolo Oy:sta (Voutilainen 2015). Seuraavana vuonna perustettiin uusi konserniyhtiö Ambientia E-commerce, jossa yhdistyivät Soprano Oyj:n ja Ambientian verkkokauppaliiketoiminnat (Soprano Oy. 2016). Vuonna 2017 Ambientia on asiantuntijatalo, jossa työskentelee yli 150 ammattilaista.

Ambientian organisaatiorakenne koostuu eri liiketoimintayksiköistä, joista opinnäytetyön kannalta oleellisimmat ovat Business Platform Services (BPS) ja E-Commerce. BPS sisältää Ambientian tukiorganisaation, joka voidaan jakaa alusta-, ylläpito- ja tukipalveluihin. Itse toimin alustapalveluiden puolella järjestelmäasiantuntijana Data Centre Services (DCS) nimisessä ryhmässä. DCS vastaa alustapalveluista, joten vastuualueeseen kuuluu luonnollisesti myös alustan suorituskyky. E-Commerce sen sijaan vastaa Magenton toiminnasta ja ohjelmakoodiin liittyvästä suorituskyvystä. Vaikka vastuut on jaettu kahden eri liiketoiminta-alueen kesken, niin tavoitteena on silti kehittää verkkokauppojen suorituskykyä yhteistyössä. Poiketen monista muista vastaavan kokoluokan toimijoista, Ambientia ylläpitää omaa palvelin- ja verkkoinfrastruktuuriaan. Tämä toimintatapa on harvinaistunut ulkoisten pilvipalveluntarjoajien yleistymisen myötä. Taustalla tässä on se, että usein on kustannustehokkaampaa vuokrata palvelinkapasiteettia ulkopuolelta kuin

ylläpitää sitä itse (Boisvert 2015). Kokonaissuorituskyvyn näkökulmasta on kuitenkin parempi, että voidaan vaikuttaa koko ympäristöön myös käyttöjärjestelmästä alaspäin. Tämä vaikutusmahdollisuus on rajoittunut niillä, jotka käyttävät ulkoista pilvipalvelun-tarjoajaa.

2.2 Magento-verkkokauppa

Ambientia hyödyntää verkkokauppa-alustanaan Magentoa. Kyseessä on avoimeen lähdekoodiin perustuva verkkokauppa-alusta, josta on olemassa ilmainen Community Edition (CE) ja kaupallinen Enterprise Edition (EE). Ilmainen CE-versio on julkaistu Open Software License (OSL 3.0) avoimen lähdekoodin ohjelmistolisenssillä. Magentosta on kaksi eri versiota: vuonna 2008 julkaistu Magento 1.0 ja vuoden 2015 lopussa julkaistu Magento 2.0 (Nyári 2016). Kun opinnäytetyössä puhutaan Magentosta, niin tällöin viitataan näistä jälkimmäiseen. Magenton vahvuuksia ovat sen avoin yhteisö ja laaja ekosysteemi, joka mahdollistaa Magenton toiminnollisuuksien laajentamisen kolmansien osapuolten kehittämillä lisäosilla. Alustan ominaisuuksien ja integraatiomahdollisuuksien skaalautuvuuden vuoksi Magento sopii sekä isoihin että pieniin verkkokauppatoteutuksiin. Asiakaskunnan laajuuden myötä Magento on yksi maailman suosituimpia verkkokauppa-alustoja (kuvio 1). Käyttöasteen perusteella Magento on neljänneksi suosituin verkkokauppa-alusta, kun mukaan lasketaan kaikki tunnetut verkkokaupparatkaisut (BuiltWith 2017). Magento on huomioitu myös suurten toimijoiden, kuten Oraclen, IBM:n ja SAPin, haastajana Gartnerin (2016) Magic Quadrant for Digital Commerce –raportissa (kuva 1).

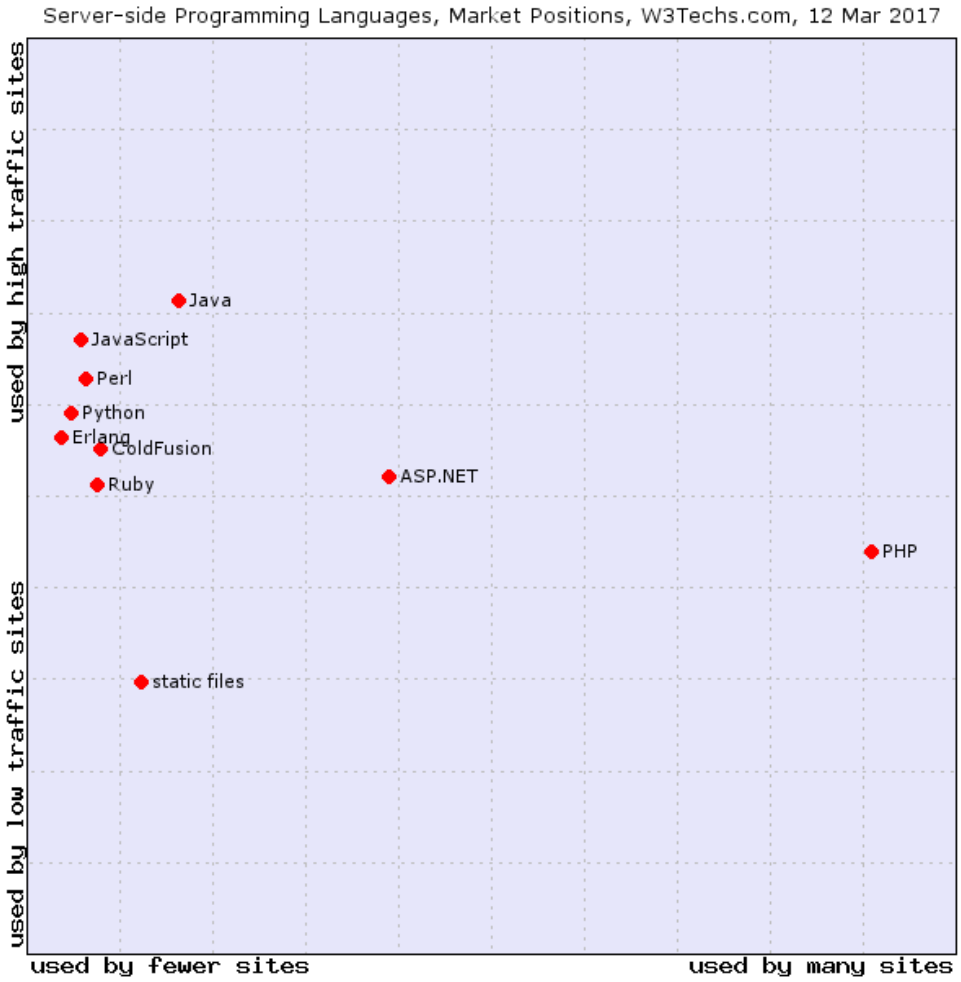


KUVIO 1. Viisi suosituinta verkkokauppa-alustaa ja näiden prosentuaaliset käyttöasteet (BuiltWith 2017)

KUVA 1. Magic Quadrant for Digital Commerce (Gartner 2016)

Magento on toteutettu PHP-ohjelmointikielellä, joka sisältää sekä vahvuuksia että heikkouksia. PHP:lla kehittäminen on nopeaa, sillä se on tulkattava ohjelmointikieli, jolloin sitä ei tarvitse erikseen kääntää. PHP on myös maailmanlaajuisesti suosituin ohjelmointikieli verkkosivustojen toteuttamiseen (kuva 2). Tulkattavan ohjelmointikielen suorituskyky on kuitenkin heikompi verrattuna sellaiseen ohjelmointikieleen, joka käännetään konekielelle ennen suoritusta (Riley 2002). Magento on ratkaissut tämän ongelman lisäämällä välimuistikerroksia suorituskyvyn parantamiseen. Magenton välimuistikerros tallentaa tietonsa joko tiedostojärjestelmään tai erilliseen tietokantaan. Välimuistikerrokset ovat kasvattaneet Magento-sovelluspinon kompleksisuutta ja tämä hankaloittaa varsinkin suorituskykyyn liittyvää vianselvitystä. Usein on hankala selvittää, missä suorituskykyyn liittyvä ongelma varsinaisesti sijaitsee, sillä välimuistien käyttö voi piilottaa alla olevat ongelmat. Yleisellä tasolla suorituskykyongelmat voidaan jakaa alustasta ja ohjelmakoodin suorittamisesta johtuviin syihin. Todellisuudessa muuttujia on paljon enemmän,

mutta tässä työssä keskitytään käyttöjärjestelmän ja sen päällä ajettavien sovelluskomponenttien suorituskykyyn. Suorituskykymittauksen toivotaan antavan parempaa näkyvyyttä alustaan siltä osin, että ongelmallisten komponenttien tunnistaminen helpottuisi.



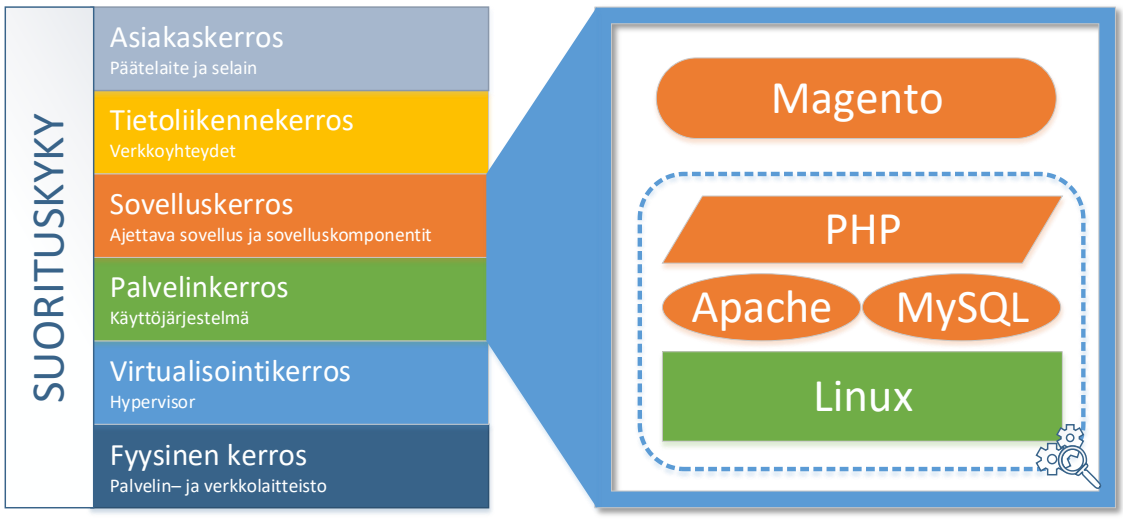
KUVA 2. Eri teknologioilla toteutettujen verkkopalveluiden lukumäärä suhteessa liikennemäärään (W3Techs 2017)

2.3 Sovelluspino

Sovelluspinoksi kutsutaan kokonaisuutta, joka rakentuu sovelluksista, joilla on riippuvuuksia toisiinsa (Liu 2009, 43). Sovelluspino koostuu erillisistä sovelluskomponenteista, jotka yhdessä luovat alustan, jonka päällä Magento-sovelusta voidaan suorittaa. Sovellusten suorituskyky ei johdu pelkästään ohjelmakoodin logiikan suunnittelusta ja toteutuksesta, vaan siihen liittyy koko sovelluspinon toiminta (Liu 2009, 43). Opinnäytetyön aihe on rajattu koskemaan Magento-sovelluspinon suorituskykyä. Tästä syystä aihealueen ulkopuolelle on rajattu Magenton oman ohjelmankoodin suoritukseen liittyvät suori-

tuskykytekijät. Magento käyttää LAMP-sovelluspinoa, joka sisältää seuraavat sovelluskomponentit: Linux (käyttöjärjestelmä), Apache HTTPd (web-palvelin), MySQL (tietokanta) ja PHP (ohjelmointikieli). LAMP on suosittu avoimeen lähdekoodiin perustuva dynaamisten verkkosivujen rakentamiseen tarkoitettu sovelluspino.

Magenton suorituskykyyn vaikuttavat sovelluspinon ulkopuoliset tekijät, kuten alla oleva verkko- ja palvelininfrastruktuuri. Verkkokerroksessa suorituskyvyn ongelmia aiheuttavat huonot tietoliikenneyhteydet ja näistä aiheutuva viive. Palvelinpuolella suorituskyvyn kannalta ratkaisevassa asemassa ovat palvelinten resurssit ja tallennusratkaisun nopeus. Mukana on usein virtualisointikerros, joka voi myös olla yksi suorituskykyyn vaikuttava asia. Kun suorituskykyä tarkistellaan asiakaspuolelta, siihen vaikuttaa muun muassa selaimessa ajettava JavaScript-komentokieli. Myös käytetty päätelaite vaikuttaa suorituskykyyn siinä määrin, että modernit verkkosivut vaativat kohtalaista prosessointitehoa, jotta verkkosivu piirtyy selaimeen ongelmitta. Kuitenkin kaikki nämä sovelluspinon ala- ja yläpuoliset tekijät on rajattu aihealueen ulkopuolelle. On kuitenkin hyvä tiedostaa, että verkkosivuston todelliseen suorituskykyyn vaikuttaa sovelluspinon lisäksi lista muita muuttujia. Oheisessa kuvassa (kuva 3) voidaan havaita mitkä eri kerrokset vaikuttavat verkkosivuston kokonaissuorituskykyyn. Kuvasta nähdään myös, että mihin kerrokseen Magenton sovelluspino sijoittuu.



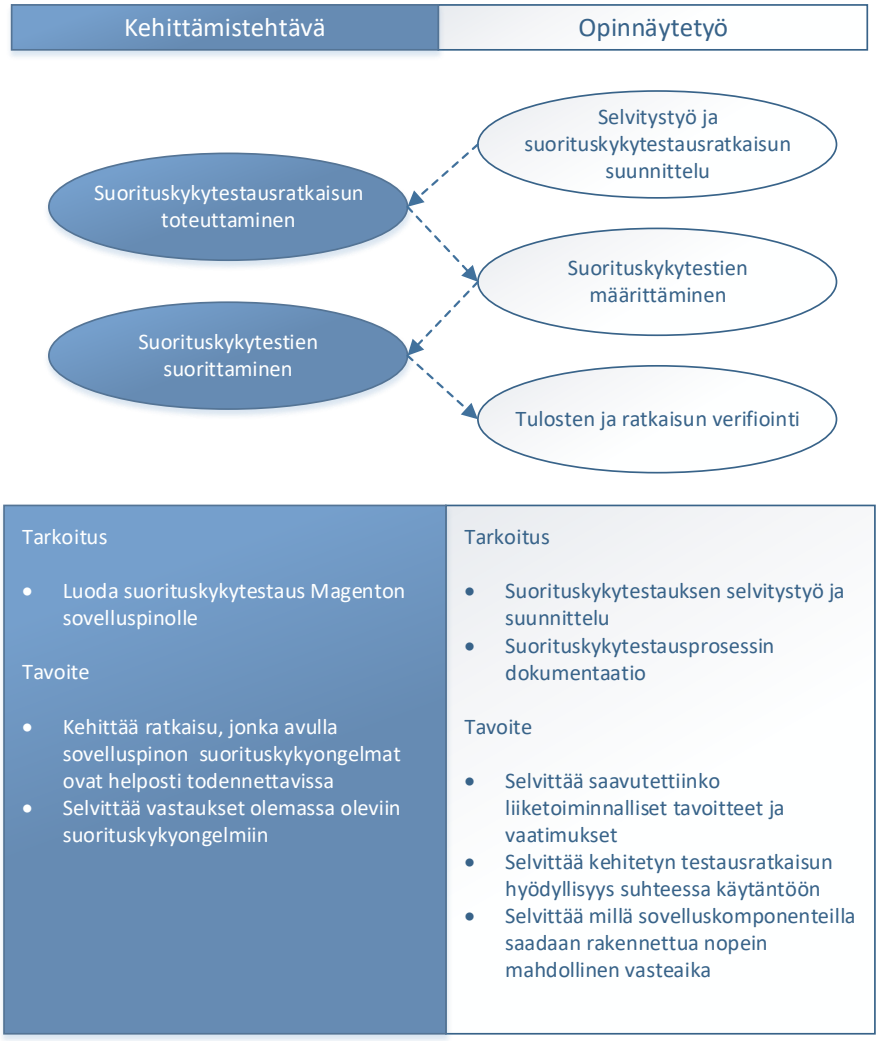
KUVA 3. Havainnointikuva verkkopalvelun kokonaissuorituskykyyn vaikuttavista asioista

3 TAVOITTEET JA MENETELMÄT

Tässä luvussa esitellään opinnäytetyön tavoitteet ja tarkoitus. Sen lisäksi kuvataan tarkemmin opinnäytetyön ja kehittämistehtävän välistä suhdetta. Luvussa esitellään myös opinnäytetyössä käytetyt tutkimusmenetelmät, joiden avulla opinnäytetyön tavoitteet pyritään saavuttamaan.

3.1 Tavoitteet

Kehittämistehtävän tavoite on luoda ratkaisu Magento-sovelluspinon suorituskykytestaukselle. Kehittämistehtävän tarkoitus on ratkaista Magenton sovelluspinon suorituskykyyn liittyviä ongelmia. Opinnäytetyön tarkoitus on suunnitella suorituskykytestaus niin, että se vastaa sille asetettuja laadullisia tavoitteita. Opinnäytetyö toimii kehittämisprosessin dokumentaationa, jossa kuvataan suorituskykytestauksen suunnittelu ja toteutus. Opinnäytetyön yksi tavoitteista on selvittää, millä komponenteilla rakennettu Magento-sovelluspino tuottaa parhaan suorituskyvyn. Sen lisäksi opinnäytetyön tavoitteena on osoittaa ratkaisun toimivuus suhteessa käytäntöön ja sille asetettuihin vaatimuksiin. Kuvassa 4 kehittämistehtävän ja opinnäytetyön suhdetta on pyritty tarkentamaan. Kehitetyn suorituskykytestausratkaisun toimivuuden todentaminen tapahtuu soveltamalla sitä toimintaympäristössä havaittujen ongelmien ratkaisemiseen. Suorituskykytestausratkaisulle asetetut laadulliset vaatimukset on määritetty yhteistyössä DCS:n ja eCommercen kanssa. Tuloksena saatiin lista laadullisista tavoitteista, jotka toimivat myös mittareina ratkaisun arvioinnissa. Näiden tavoitteiden pohjalta on laadittu tutkimuskysymys, jonka tehtävä on selvittää, kuinka luoda helposti käyttöönotettava, helposti muokattava ja toistettava suorituskykytestaus Magento-verkkokaupan sovelluspinolle.

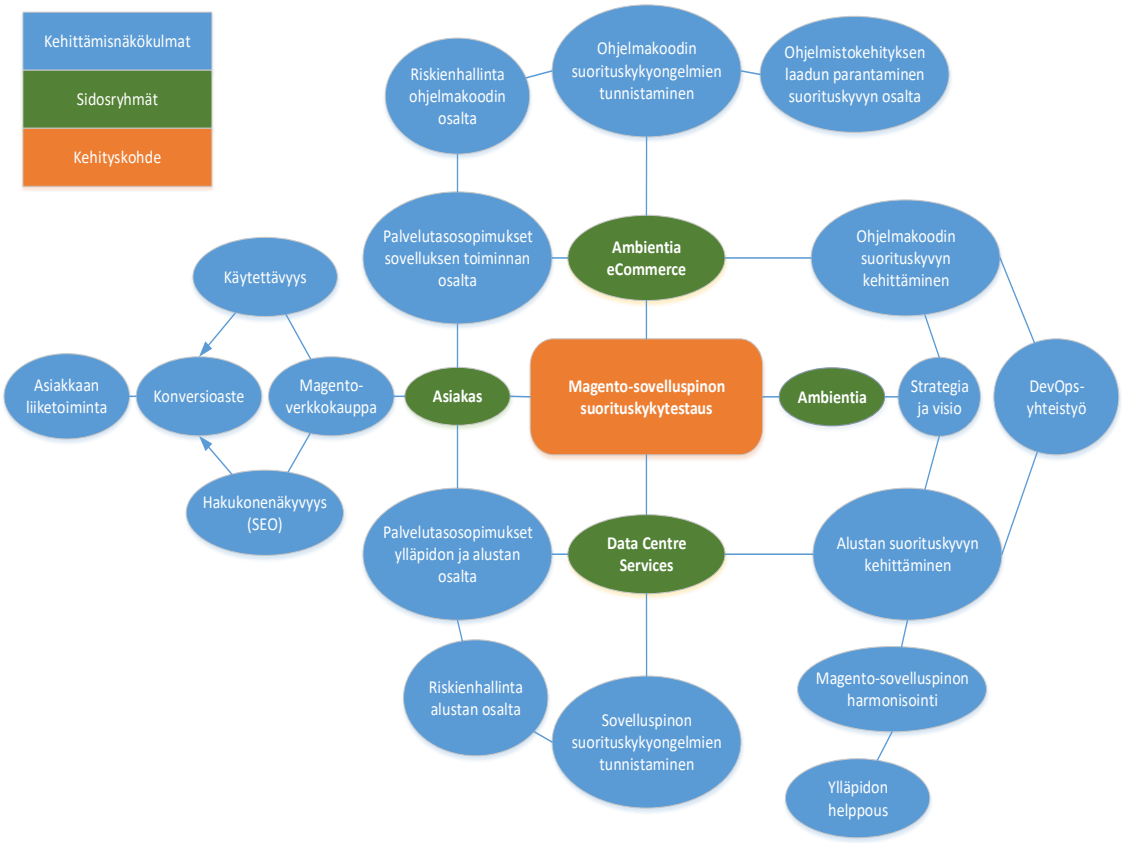


KUVA 4. Kehittämistehtävän ja opinnäytetyön välinen suhde

Helposti käyttöönotettavuus pyritään saavuttamaan sillä, että hyödynnetään valmiita testaustyökaluja suorituskykytestauksen toteuttamiseen. Näin saadaan minimaalisella kehitystyöllä suorituskykytestaukselle perusta, jota voidaan tarvittaessa laajentaa eri käyttötarkoituksiin. Muokattavuudella pyritään siihen, että suorituskykytestausta voidaan laajentaa ohjelmistokehityksen laadun varmistukseen ja riskien hallintaan. Ensisijainen käyttötarve suorituskykytestaukselle on kuitenkin sovelluspinoon liittyvien suorituskykyongelmien selvitys. Suorituskykytestauksen toistettavuudella pyritään varmistamaan tulosten johdonmukaisuus sekä mahdollistamaan testiautomaation kehittäminen. Testiautomaation avulla suorituskykytestit voidaan automatisoida, jolloin manuaalisten toimenpiteiden määrä vähenee. Tämä vähentää inhimillisten virheiden määrää ja parantaa tulosten luotettavuutta. Testiautomaatio on viime vuosina ollut ohjelmistotestausalan kasvava trendi ja merkittävimpiä hyötyjä ovat sen nopeus, tehokkuus ja regression varmistaminen (Pesonen 2016).

3.2 Tutkimusmenetelmät

Opinnäytetyön tavoitteena on tuottaa uutta tietoa ja siihen tarkoitukseen sovelletaan erilaisia tutkimusmenetelmiä. Tutkimusmenetelmien hyödyntäminen lisää opinnäytetyön tutkimuksellista luonnetta. Kun yhdistetään kehittämistoiminta ja tutkimuksellinen lähestymistapa, niin sitä kutsutaan myös tutkimukselliseksi kehittämistoiminnaksi (Toikko & Rantanen 2009, 21). Toikko ja Rantanen (2009, 123-124) toteavat, että tuotetun tiedon vakuuttavuus perustuu sen uskottavuuteen ja johdonmukaisuuteen. Tiedon laatuun liittyviä tekijöitä voi pyrkiä parantamaan käyttämällä tunnettuja ja hyväksi todettuja tutkimusmenetelmiä. Ojasalon ym. (2014, 24-25) mukaan yksi tutkimuksellisen kehitystyön keskeisimmistä asioista on löytää kehittämisen näkökulmat. Tässä apuna käytetään käsitejärjestelmää, jota kutsutaan myös käsiteanalyysiksi. Käsiteanalyysin tuloksena on käsitekartta, jossa kuvataan kehittämistoiminnan käsitteet ja niiden suhteet (Toikko & Rantanen 2009, 133). Käsitekartta avaa uusia näkökulmia ja piirteitä, joita ei ennestään ole huomioitu. Koska suorituskykytestauksen kehittämisnäkökulmia on useita, nämä jäsennetään käyttämällä apuna käsitekarttaa (kuva 5), jotta kaikki sidosryhmät tulee tunnistettua.



KUVA 5. Käsitekartta eri sidosryhmistä ja kehittämisnäkökulmista

Kehittämistehtävä voidaan mieltää kehittämisprosessiksi, joka voidaan jakaa viiteen eri osaan: perustelu, organisointi, toteutus, arviointi ja tulosten levittäminen. Perustelun lähtökohta on usein ongelma tai visio ja se vastaa kysymykseen, miksi kehitystä tulisi tehdä. Kehittämisen tavoitteet perustellaan mahdollisimman konkreettisesti, sillä se auttaa itse toteutuksen ohjauksessa. Tavoitteiden määrittelyssä käytetään tähän tarkoitettua työkalua, joka kulkee nimellä loogisen viitekehyksen matriisi (taulukko 1). Toteutusvaihe pitää sisällään konkreettisen tekemisen, jonka aikana pidetään yllä analysoivaa ja pohtivaa otetta. Arviointi on yksi tiedon tuottamisen välineitä, jolla itse kehittämisprosessia voidaan ohjata. Arviointia suoritetaan koko kehittämisprosessin ajan, sillä projektin tavoitteet voivat muuttua. Tämä varmistetaan käyttämällä iteratiivista kehittämismallia, jossa kehitystä tehdään pienissä sykleissä ja samalla projektin etenemistä ohjataan kohti tavoitteita. Viimeinen kehittämisprosessin vaihe on tulosten levittäminen. Tulosten levittämistä voidaan edistää esimerkiksi tuotteistamisen tai koulutuksen avulla. Tässä tapauksessa opinnäytetyö itsessään on tulosten levittämiseen tarkoitettu väline. (Toikko & Rantanen 2009, 56-63, 75-76.)

TAULUKKO 1. Tavoitehierarkia ja loogisen viitekehyksen matriisi

	Interventiologiikka	Indikaattorit	Todentamisen välineet	Oletukset ja riskit
PÄÄMÄÄRÄ	Päämääränä on luoda suorituskykytestausratkaisu Magento-sovelluspinolle	Suorituskykytestausratkaisulla voidaan ajaa suorituskykytestejä, joista selviää Magento-sovelluspinon suorituskyky	Suorituskykytestauksesta saadaan uutta tietoa, jota voidaan hyödyntää Magento-sovelluspinon suorituskyvyn kehityksessä	Riskit: Suorituskykytestaus ei ole tarpeeksi kattava tai se mittaa vääriä asioita. Suorituskykytestien tulokset eivät ole verrattavissa reaalimaailmaan.
TARKOITUS	Tarkoitus on ratkaista Magento-sovelluspinoon liittyviä suorituskykyongelmia sekä selvittää millä sovelluskomponenteilla saavutetaan nopein vasteaika.	Pystyttiinkö suorituskykyongelmat ratkaisemaan ja saatiinko suorituskykytestauksista tarvittava tieto Magento-sovelluspinon suorituskyvyn kehittämiseen.	Suorituskykyongelmat on ratkaistu ja Magento-sovelluspinon suorituskyky on todistettavasti nopeutunut. Pääasiallisena menetelmänä tiedon tuottamisessa käytetään suorituskykyvertailua.	Riskit: Suorituskykyongelmia ei voida todentaa suorituskykytestauksella. Suorituskykytestauksesta ei saada vertailtavia tuloksia Magento-sovelluspinon kehityksen tueksi.
ODOTETUT TULOKSET	Suorituskykytestausratkaisun tulee olla helposti käyttöönotettava, muokattava ja toistettava. Suorituskykytestauksista saatujen tulosten avulla voidaan kehittää Magento-sovelluspinon suorituskykyä.	Suorituskykytestaus ei vaadi mittavia manuaalisia toimenpiteitä. Suorituskykytestausta voidaan muokata myös muuhun käyttöön kuin sovelluspinon suorituskyvyn mittaamiseen. Magento-sovelluspinon suorituskyky on parantunut. Suorituskykytestit voidaan toistaa helposti.	Ratkaisu on täysin tai osittain automatisoitu. Suorituskykytestaus on käytössä myös ohjelmakoodin suorituskyvyn mittaamiseen. Magento-sovelluspino on määritelty uudelleen suorituskykytestien tulosten perusteella.	Oletukset: Kehittämistyötä varten tulee perustaa projekti, jotta siihen voidaan kiinnittää tarpeelliset resurssit. Riskit: Riskinä on sisäinen kehitystyön eriarvoisuus verrattuna laskutettavaan työhön.
TOIMENPITEET	Suorituskykytestausratkaisun kehityksessä käytetään valmiita ratkaisuja ja työkaluja, joita on helppo muokata eri tarpeiden mukaan. Ratkaisun tulee olla automatisoitavissa, jotta se on helposti toistettavissa. Suorituskykytestien tulokset on oltava helposti vertailtavia toisiinsa.	Suositaan valmiita ratkaisuja ja hyödynnetään avoimen lähdekoodin ohjelmistoja. Testaustyökaluissa oltava ohjelmointirajapinta (API) tai niitä voidaan kutsua komentoriviltä. Suorituskykytestien tulokset tulee pystyä visualisoimaan vertailun ja luettavuuden parantamiseksi.	Sisäistä kehitystyötä on tehty mahdollisimman vähän. Testaustyökaluja voidaan kutsua ohjelmallisesti. Suorituskykytestausten tulokset voidaan saada ulos graafisessa muodossa.	Riskit: Suorituskykytestausratkaisu joudutaan rakentamaan alusta alkaen itse, sillä valmiita ratkaisuja ei ole saatavilla. Vaihtoehtoina vain kaupallisia ratkaisuja, joiden muokkaaminen ja käyttö ovat rajoitettua. Testaustyökalut eivät tarjoa ohjelmointirajapintaa, jolloin testiautomaatiota ei voida toteuttaa. Suorituskykytestien tulokset saadaan ulos vain numeerisena datana.

Ennen tutkimusmenetelmien valintaa on hyvä tunnistaa kehittämistehtävän lähestymistapa. Lähestymistavan tunnistaminen auttaa kehittämistyön suunnittelussa ja tutkimuksellisuuden liittämisessä kehitykseen. Tämän kehittämistehtävän lähestymistavassa yhdistyvät konstruktiivisen ja toimintatutkimuksen piirteet. Toimintatutkimuksen piirteet tulevat esille, kun kehityksen kohteena on ihmisten tai organisaation toiminta. Tässä tapauksessa toimintatutkimusta edustaa suorituskykytestausprosessin määrittäminen ja sen liittäminen osaksi ihmisten toimintaa. Konstruktiivinen tutkimuksessa muutos kohdistuu johonkin konkreettiseen tulokseen, eli tässä tapauksessa Magento-sovelluspinoon ja sen suorituskyvyn mittaamiseen. Kehittämistehtävän tavoitteena on luoda käytäntöön soveltuva ja teoreettisesti perusteltu ratkaisu Magento-sovelluspinon suorituskykytestaukseen.

Tämä toteutuu silloin, kun käytännön ongelma ja siihen liittyvä ratkaisu sidotaan teoreettiseen tietoperustaan, eli tässä tapauksessa suorituskykytestauksen teoriaan. (Ojasalo ym. 2014, 36-38, 65-68.)

Tuotoksen toimivuuden ja hyödyllisyyden arviointi ovat tärkeä osa konstruktiivista tutkimusta (Ojasalo ym. 2014, 68). Tämän vuoksi suorituskykytestausratkaisun toimivuuden osoittaminen ja tavoitteiden saavuttaminen ovat opinnäytetyön keskeisiä tavoitteita. Konstruktiivisen tutkimuksen haasteina ovat tuotoksen toimivuuden toteennäyttäminen ja sen tieteellisen annin osoittaminen (Ojasalo ym. 2014, 68). Tuotoksen toimivuus voidaan perustella hyvillä käytännöillä, joissa korostuu ratkaisun käytännön hyödyllisyys (Toikko & Rantanen 2009, 148). Työn tieteellisyys pyritään osoittamaan suorituskykytestauksista saatavalla datalla, jonka analysoimiseen käytetään suorituskykyvertailua. Suorituskykytestausten lähtö- ja loppuarvoja vertaillaan keskenään, jotta saadaan selville Magento-sovelluspinon muutoksista syntynyt suorituskyvyn vaihtelu. Suorituskykytestausratkaisuun liittyvien laadullisten tavoitteiden saavuttamista arvioidaan käyttöönoton helppoudella, muokattavuudella ja toistettavuudella.

4 SUORITUSKYVYN MITTAAMINEN JA KEHITTÄMINEN

Opinnäytetyön menetelmäosiossa selvitettiin käsiteanalyysin avulla useita eri verkkokauppojen suorituskyvyn mittaamiseen liittyviä sidosryhmiä ja kehittämisen näkökulmia. Tämän osion tarkoitus on keskittyä sisäisiin ja ulkoisiin sidosryhmiin, jotta saadaan laajempi kokonaisnäkemys siitä, että miksi verkkokauppojen suorituskyky ja sen mittaaminen ovat tärkeitä kehittämisen kohteita. Tärkein suorituskyvyn kehittämisnäkökulma ja sidosryhmä on luonnollisesti asiakas, sillä kaikki kehittämistoiminta tulee ensisijaisesti olla asiakaslähtöistä. Taustalla tässä on se, että asiakkaan liiketoiminnan tehostaminen on koko verkkokauppaliiketoiminnan ydin. Ilman asiakasta ei Ambientian verkkokauppaliiketoiminnalle olisi toimintaedellytyksiä. Kun verkkokaupan suorituskykyä tarkastellaan asiakkaan näkökulmasta, tulee tunnistaa asiakkaan liiketoimintaan vaikuttavat tekijät. Asiakkaan verkkokauppaliiketoimintaan vaikuttavia ja suorituskykyyn liittyviä tekijöitä on tarkoitus käsitellä tarkemmin tässä osiossa.

4.1 Suorituskyvyn mittaamisen haasteet

Suorituskyvyn mittaaminen sisältää useita haasteita, jotka liittyvät suorituskykytestauksen suunnitteluun, tekniseen toteutukseen ja tulosten arviointiin. Suorituskykytestauksen suunnittelu on tärkeässä roolissa, jotta osataan mitata oikeita asioita ja oikeilla menetelmillä. Suorituskykytestauksen suunnittelun keskeisessä osassa on tavoitteiden määrittely, eli toisin sanoen selvitetään, mitä suorituskykyyn liittyvää asiaa on tarkoitus mitata. Kun puhutaan suorituskykytestauksesta, niin tämä ei vielä yksiselitteisesti määrittele suorituskykytestauksen tavoitteita. Suorituskykytestauksen tavoitteena voi olla selvittää verkkosivuston vasteaikoja, suorituskykykapasiteettia tai niiden yhteisvaikutusta. Verkkosivuston vasteajalla kuvataan sitä aikaa, joka alkaa sivupyynnöstä ja päättyy verkkosivun latautumiseen. Suorituskykykapasiteetti kuvaa taas sitä, kuinka monta käyttäjätransaktiota verkkosivu pystyy normaalisti palvelemaan tietyllä aikavälillä. Toisin sanoen suorituskykytestauksen tavoitteita voi olla monenlaisia ja ne kannattaa sopia tärkeiden sidosryhmien kanssa. Haasteena onkin sopia kaikkien sidosryhmien kanssa yhteisiä tarkoitusperiä palvelevat suorituskyvyn mittarit.

Suorituskykytestauksen suunnittelussa tulee huomioida testitapausten kattavuus. Jotta suorituskykytestauksesta saadaan hyödyllisiä tuloksia, tulee suunnitella, miten suoritus-

kykytestausta tehdään. Se ei vielä riitä, että suorituskykytestauksen aikana tehdään sivupyyntöjä verkkopalvelun etusivulle. Testitapaukset tulisi seurata tarkasti käyttäjien tekemiä toimintoja, jotta suorituskykytestauksesta saadut tulokset olisivat reaalimaailmaan verrattavissa. Testitapausten tarkempi määrittely vaatii tukea myös testaustyökalulta. Eräät suorituskykytestaustyökalut tukevat testitapausten nauhoittamista suoraan selaimessa. Jotkin suoritustestaustyökalut sisältävät oman merkintäkielensä (eng. Domain Specific Language), jota hyödyntämällä voidaan kirjoittaa testitapauksia. Mitä laajemmin erilaisia toiminnallisuuksia kuvaavia testitapauksia on määritetty, sitä kattavampia ovat suorituskykytestauksen tulokset. Testitapauksien laatimisessa ongelmana on tunnistaa, mikä on merkityksellistä loppukäyttäjälle näkyvään suorituskykyyn (Kivinen 2009, 30).

Kun mitataan verkkosivuston suorituskykyä, niin haasteena tässä on kokosuorituskyvyn profilointi. Profiloinnilla tarkoitetaan suorituskykymittauksista saatujen tulosten jakamista pienempiin osiin. Esimerkiksi sovellustasolla suorituskyvyn profilointi mahdollistaa vasteajan jakamisen eri sovelluskomponenttien mukaan. Näin saadaan selville, miten vasteaika jakautuu esimerkiksi PHP:n ja MySQL:n kesken. Suorituskyvyn tarkempi profilointi kuitenkin vaatii valvonnan laajentamisen sovellustasolle. Tähän yhtenä ratkaisuna on sovelluksen suorituskykymonitorointi (eng. Application Performance Monitoring), jonka avulla saadaan parempi näkyvyys sovellustason suorituskyvystä. Monitorointiratkaisuja on vastaavasti tarjolla myös muihin kuin sovelluskerrokseen, jos halutaan saada selville miten kokonaissuorituskyky muodostuu.

4.2 Suorituskyvyn kehittämisnäkökulmat

Tässä osiossa käsitellään suorituskyvyn kehittämisnäkökulmia. Ensimmäinen näistä on yrityksen strateginen kehittämisnäkökulma. Ambientian verkkokauppaliiketoiminnan kehittämisestä on olemassa 5-vuotisstrategia, jossa on listattu tulevaisuuden kehittämistavoitteet. Strategian mukaan Ambientian verkkokauppaliiketoiminnan visiona on tarjota Suomen nopeinta ja luotettavinta Magento-verkkokauppaa (Tielinen 2016). Verkkokaupan nopeus on muutenkin nostettu esille strategiseksi kehittämistavoitteeksi. Nopeuden lisäksi strategiassa mainitaan verkkokaupan vakaus ja saavutettavuus. Jotta visioon pohjautuvat tavoitteet voidaan saavuttaa, tulee kerätä tarkempaa tietoa Suomessa toimivista kilpailijoista ja verkkokauppojen suorituskyvystä. Sen lisäksi tarvitaan myös menetelmä, jolla voidaan mitata ja verrata suorituskykyä. Suorituskykytestausmenetelmien kehittäminen auttaa siis myös yrityksen strategisten tavoitteiden saavuttamisessa.

Yrityksen strategisen näkökulman lisäksi verkkokauppojen suorituskyvyn kehittämiseen liittyviä sidosryhmiä ovat itse asiakas, Data Centre Services ja eCommerce. Näitä yhdistävät osapuolten väliset sopimukset. Suorituskyvyn kannalta yksi tärkeä sopimustyyppi on palvelutasosopimus (eng. Service Level Agreement). Palvelutasosopimuksissa voi olla määritetty verkkopalvelun vasteaika ja kuinka monta yhtäaikaisia käyttäjää sen tulee pystyä palvelemaan. Palvelutasosopimuksiin usein sisältyy myös sopimussanktioita, joiden perusteella voidaan määrätä esimerkiksi hinnan alennuksia, jos palvelutaso ei ole sopimuksen mukainen. Riskien realisoituminen aiheuttaisi molemmille osapuolille merkittävää haittaa. Asiakkaan liiketoiminta kärsii ja samalla toimittaja joutuu maksamaan kompensaatiota. Verkkokaupan suorituskykytestaus on osa riskienhallintaa, jonka avulla pyritään ehkäisemään riskien realisoitumista. Suorituskykytestauksen avulla voidaan selvittää jo etukäteen, että verkkokaupan suorituskyky on sopimuksen mukainen.

Verkkokauppojen suorituskyvyn kehittämistä tehdään yhteistyössä eCommercen kehittäjien ja DCS:n palvelinylläpidon kanssa. Yhteistyön toimintamallina on käytetty DevOpsia, jossa mittaaminen on keskeisessä roolissa. John Williams ja Damon Edwards (2010) ovat lanseeranneet DevOpsiin liittyvän CAMS-käsitteen, joka tulee sanoista kulttuuri (Culture), automaatio (Automation), mittaus (Measurement) ja jakaminen (Sharing). Usein CAMS-käsitteeseen lisätään myös L-kirjain, joka tarkoittaa Lean-menetelmien hyödyntämistä osana DevOpsia. Yhden määrityksen mukaan DevOps on kokoelma toimintatapoja, joiden avulla vähennetään sitä aikaa, joka kuluu muutoksen viemisestä tuotantoon (Bass ym. 2015, 4). Kun yhdistetään aika ja sen mittaaminen, niin nämä yhdessä muodostavat menetelmän, jolla toiminnan tehokkuutta voidaan systemaattisesti arvioida. Suorituskykytestauksen kannalta mittaaminen on tärkeässä osassa, sillä sen avulla suorituskyvyn kehittämistä voidaan ohjata oikeaan suuntaan.

Ambientialla DevOps-toimintamallia käytetään kehittämisprojekteissa, joissa kehittämistoiminta tapahtuu palvelin- ja sovelluspuolen välimaastossa. Tällä toiminnalla pyritään saamaan kehittäjät ja ylläpitäjät työskentelemään kohti yhteisiä tavoitteita. Perinteisesti nämä ryhmät on sijoitettu eri siiloihin ja tällä on negatiivinen vaikutus yhteistyöhön. DevOps-toiminnan korkean tason tavoite on liiketoiminnan onnistumisen mahdollistaminen. Kuten aiemmin mainittu, DevOps on vahvasti sidoksissa yrityksen työskentelykulttuuriin. On olemassa useampia määrityksiä siitä, että miten DevOpsia kuuluisi tehdä, mutta yhtä ainoaa oikeaa tapaa ei ole olemassa. Parhaimpaan tulokseen pääsee silloin,

kun DevOpsia tehdään oman työympäristön ehdoilla ja siihen sopivin menetelmin. DevOpsin parhaista käytännöistä ja menetelmistä kannattaa valita ne, jotka aidosti tuottavat mitattavaa lisäarvoa.

Suorituskyvyn mittaaminen on tärkeä osa myös verkkokauppojen järjestelmäarkkitehtuurin ja sovelluskonfiguraation kehittämistä. Suorituskykytestauksen avulla saadaan tärkeää tietoa, jonka perusteella voidaan tehdä päätöksiä, jotka edesauttavat suorituskyvyn kehittämistä. Suorituskykytestaus on työkalu, joka toimii päätöksenteon tukena, kun verkkokauppaympäristöjen järjestelmäarkkitehtuuria muutetaan suorituskykyisempään suuntaan. Kun saadaan selville mistä komponenteista koostuu suorituskykyisin Magento-sovelluspino, niin se auttaa tulevaisuudessa harmonisoimaan verkkokauppaympäristöjä. Ympäristöjen harmonisointi tehostaa toimintaa siinä määrin, että se vähentää verkkokauppaympäristöjen ylläpitoon ja pystyttämiseen vaadittua työmäärää. Sovelluskehityksen osalta suorituskykytestaus on yksi menetelmä, jolla ohjelmakoodiin liittyvä suorituskyvyn regressio voidaan havaita. Ihannetilanne tässä on se, että suorituskykytestaus olisi mukana jatkuvan integraation prosessissa (Continuous Integration), jolloin suorituskykytestaus tapahtuisi automaattisesti jokaisen ohjelmakoodimuutoksen jälkeen.

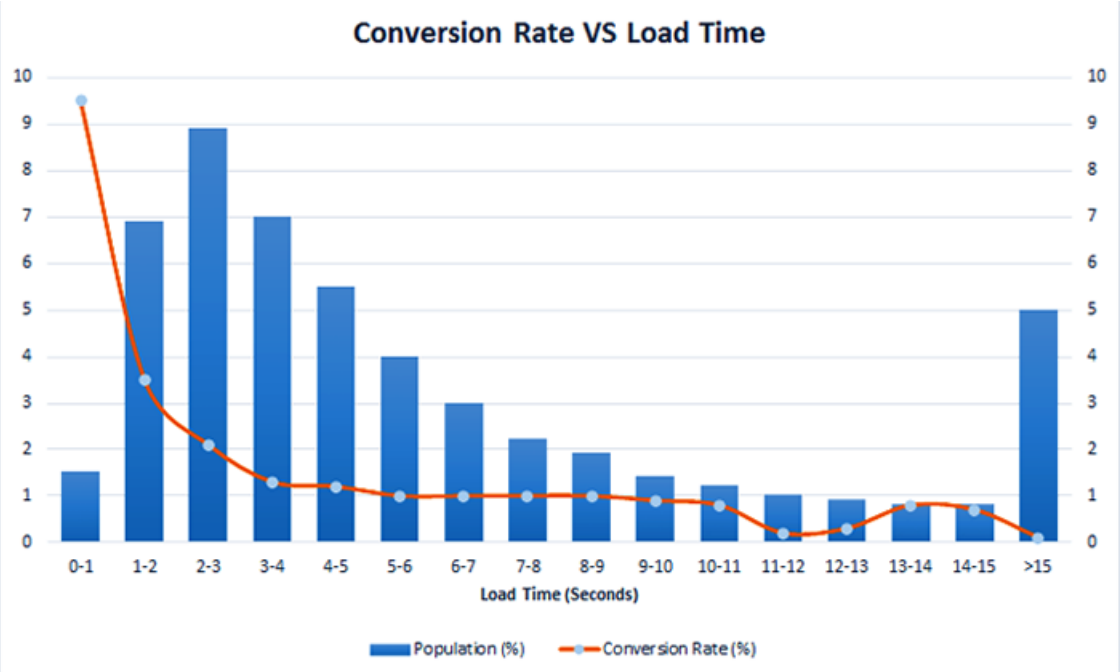
4.3 Verkkokaupan suorituskyvyn vaikutukset

Tämä osio on erillään muista kehitysnäkökulmista siitä syystä, että tässä tarkastellaan verkkokaupan suorituskykyä ja millä tavoin se vaikuttaa asiakkaan liiketoimintaan. Verkkokaupan alentunut suorituskyvy näyttäytyy negatiivisena kokemuksena verkkokaupan käyttäjille. Negatiivisella käyttäjäkokemuksella on taas vaikutusta asiakastyytyväisyyteen, joka laskee verkkokaupasta tehtyjen tilausten lukumäärää. Suorituskykyongelmat voivat vaikuttaa verkkokaupan liiketoimintaan siinä määrin, että se on este liiketoiminnan kasvulle. Yleisenä motivaattorina toimii verkkokaupan liikevaihdon ja tuloksen kasvattaminen. Verkkokaupan suorituskyky voi tuoda ratkaisevaa etua markkinoilla, jossa on paljon kilpailijoita. Jos eroavaisuuksia ei kahden eri verkkokaupan välillä ole, kuluttaja voi tehdä ostopäätöksen, joka pohjautuu verkkokauppaan liittyvään käyttökokemukseen.

4.3.1 Suorituskyvyn vaikutus konversioasteeseen

Konversiolla tarkoitetaan mitattavaa tapahtumaa, jossa käyttäjä suorittaa ennalta määritellyn tavoitteen. Verkkokaupoissa tämä tavoite on luonnollisesti tuotteen tilaaminen, mutta jossain muussa yhteydessä se voi olla esimerkiksi sähköpostilistalle rekisteröityminen tai mainosbannerin klikkaaminen. Konversiota mitataan erillisellä konversioasteella, joka tarkoittaa toteutuneiden tavoitteiden ja sivulla käyneiden välistä suhdelukua (Vainio 2014). Aberdeen Groupin (2008) tekemän tutkimuksen mukaan yhden sekunnin vasteajan kasvu tarkoittaa 7 % negatiivista muutosta konversioasteeseen. Toisen tutkimuksen mukaan konversioaste ei kulje lineaarisesti suhteessa vasteaikaan, vaan konversioaste laskee merkittävästi, kun vasteaika liikkuu 0-4 sekunnin välillä (Bixby 2012). Kuvassa 6 on esimerkki siitä, että miten sivuston vasteaika vaikuttaa konversioasteen kehitykseen.

Yllä mainittujen tutkimusten perusteella konversioastetta voidaan kasvattaa parantamalla verkkokaupan suorituskykyä. Konversioaste ja sen vaihtelu ovat yksilöllistä jokaiselle verkkokaupalle, mutta suorituskyvyn ja konversioasteen välillä on kuitenkin tunnistettava yhteys. Kun konversioastetta pyritään kasvattamaan tietoisesti, puhutaan konversio-optimoinnista. Konversio-optimointi on datalähtöinen lähestymistapa konversiosuhteen kasvattamiseen. Suosittuja menetelmiä tähän ovat AB-, split- tai nk. multivariate-testaus (Haapahovi 2015). Testausmenetelmien avulla selvitetään kahden tai useamman variaation vaikutus asiakkaiden ostoskäyttäytymiseen. Esimerkiksi verkkokaupan etusivusta voi olla kahta tai useampaa eri variaatiota ja analytiikan avulla voidaan selvittää, kummalla näistä on suurempi konversioaste. Knuutila (2016) kuitenkin varoittaa, että AB-testauksessa piilee ongelmansa. Jos taustatutkimus on tehty puutteellisesti, AB-testaus on omien hypoteesien testausta.



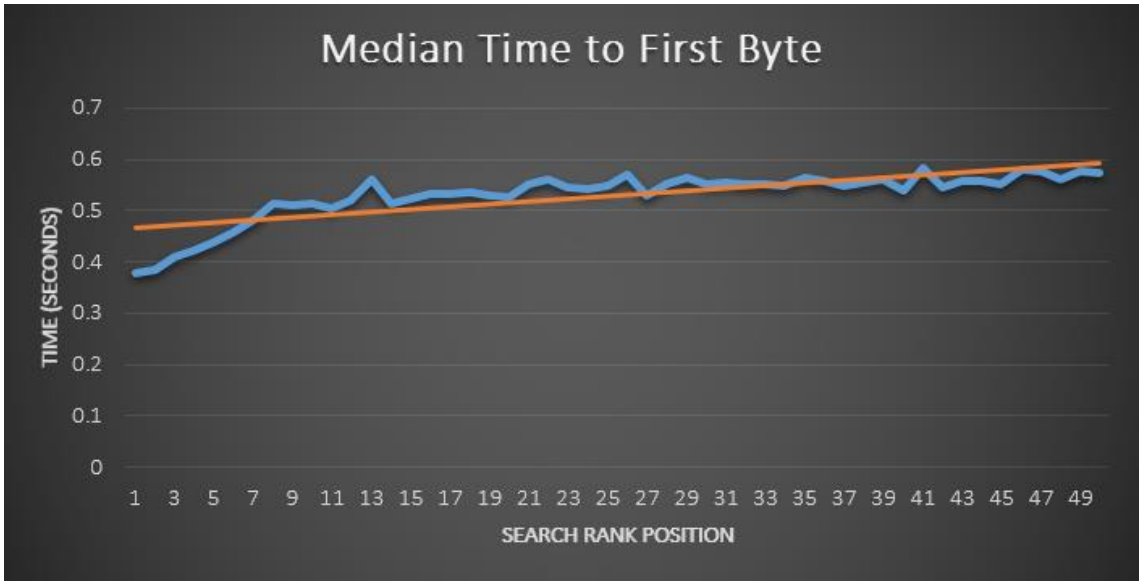
KUVA 6. Walmart.com:n konversioaste suhteessa vasteaikaan (Bixby 2012)

4.3.2 Suorituskyvyn vaikutus hakukonenäkyvyyteen

Hakukonenäkyvyys on verkkokaupoille tärkeää, jotta potentiaaliset asiakkaat löytävät perille. Internetin suuret hakukoneet, kuten Google ja Bing, sijoittavat hakutulokset erillisten hakukonealgoritmien perusteella. Hakukonealgoritmiin liittyviä painoarvoja ei ole virallisesti saatavilla, mutta esimerkiksi Googlella uskotaan olevan yli 200 eri painoarvoa, jotka vaikuttavat verkkosivuston hakukonenäkyvyyteen (Dean 2016). Yksi näistä tekijöistä on verkkosivuston vasteaika, joka on ollut osa Googlen käyttämää hakukonealgoritmia vuodesta 2010 (Singhal 2010). Vaikka verkkopalvelun vasteaika on todistetusti yksi hakukonenäkyvyyteen vaikuttaja tekijä, niin se on silti painoarvoltaan hyvin marginaalinen. Googlen hakukoneosaston entinen johtaja Matthew Cutts (2010) toteaa blogissaan, että verkkosivustojen hitaus vaikuttaa negatiivisesti vain yhteen prosenttiin kaikista hakutuloksista.

Verkkosivuston sisältöön liittyvät asiat ovat edelleen painoarvoltaan hakukonenäkyvyyden tärkein osa-alue, mutta tulevaisuudessa tämä saattaa kuitenkin muuttua. Kun verkkosivuston hakukonenäkyvyyttä pyritään kasvattamaan suorituskykyä parantamalla, niin silloin tulee ymmärtää, mitä vasteajan arvoa hakukoneet itse asiassa mittaavat. Erään tutkimusraportin mukaan koko verkkosivun latausajalla (eng. load time) ei ole merkitystä Googlen hakutulosten järjestyksen suhteen (Hoffman 2013). Sen sijaan Googlen hakutu-

lokset korreloivat sellaiseen vasteajan arvoon, joka mittaa aikaa sivupyynnöstä ensimmäiseen vastaanotettuun tavuun (kuva 7). Tätä kutsutaan myös nimellä TTFB (Time to First Byte). TTFB on looginen vaihtoehto vasteajan mittaukseen, sillä hakukoneet käyttävät verkkosivujen indeksointiin robotteja, jotka samalla mittaavat verkkosivujen vasteaikaa. Koska robotit eivät lataa sivuja samalla tavalla kuin käyttäjän selain, niin se rajoittaa vaihtoehtoja vasteajan mittaamiseen. Synteettisten suorituskykymittausratkaisujen kehittyessä voi TTFB vaihtua sellaiseen, joka kuvaa paremmin loppukäyttäjän kokemaa vasteaikaa (Hoffman 2013).



KUVA 7. Verkkosivujen vasteajan TTFB-arvon korrelaatio Googlen hakutulosten sijoituksiin (Hoffman 2013)

4.3.3 Suorituskyvyn vaikutus käytettävyyteen ja saavutettavuuteen

Kun puhutaan verkkosivujen ja käyttäjien välisestä interaktiosta, tärkein ominaisuus on nopea latautumisaika. Verkkosivujen latautumiseen liittyvä kokemus on kuitenkin subjektiivinen eli jokainen kokee sen omalla tavallaan. Joidenkin mielestä verkkosivu voi olla käyttökelvoton, kun sen lataus kestää yli kymmenen sekuntia. Tämä ei ole välttämättä ongelma niille, jotka ovat tottuneet odottamaan sivujen latautumista. Tästä syystä käyttäjän odotukset määräävät, miten suorituskyvyn kokemus syntyy. Kokemukseen voi vaikuttaa esimerkiksi käytössä oleva päätelaite tai tietoliikenneyhteydet. Jotta käyttäjä voi tuntea liikkuvansa verkkosivulla vapaasti, vasteajan on oltava alle sekunti. Tämä alle sekunnin vasteaika perustuu useisiin verkkosivujen käytettävyyteen liittyviin tutkimuksiin. (Nielsen 2000, 43-47.)

Verkkokauppojen suorituskyvystä ei ole olemassa laajalti tunnettua ohjeistusta tai standardia, mutta suuntaa-antavaa käyttäjätutkimusta on olemassa. Tunnetun CDN-tarjoajan, Akamain, teettämän tutkimuksen mukaan verkkokauppojen tavoiteltava vasteaika on alle kaksi sekuntia. Tutkimuksen mukaan 47 % käyttäjistä olettaa, että verkkokaupan sivun lataukset tulee latautua alle kahden sekunnin. Kun verkkokaupan vasteajat ylittävät kolme sekuntia, 40 % kyselyyn vastanneista kertoi poistuvansa palvelusta. Yli puolet (52 %) kyselyyn vastanneista pitää verkkokaupan suorituskykyä tärkeänä ominaisuutena tehdessään ostoksia verkossa. Tutkimuksesta voidaan päätellä, että kuinka nopea verkkokaupan vasteaika tulisi olla, jotta sen käyttökokemus vastaisi kuluttajien olettamuksia. (Akamai 2009.)

Saavutettavuus liittyy verkkokaupan suorituskykyyn siinä määrin, että joissain poikkeustilanteissa vasteaika voi kasvaa lopulta niin suureksi, että verkkokauppa ei ole saavutettavissa. Normaalissa tilanteessa verkkokaupan tulee pystyä palvelemaan tietty määrä yhtäaikaisia käyttäjiä ilman, että siitä aiheutuu haittaa sivuston saavutettavuuteen. Kuitenkin jossain poikkeustilanteissa verkkopalvelu ei pysty palvelemaan kaikkia käyttäjiä, sillä palvelinten suorituskykykapasiteetti loppuu kesken. Taustalla voi olla esimerkiksi juuri julkaistu tarjouskampanja, joka kasvattaa kävijämäärää hetkellisesti niin suureksi, että palvelu ruuhkautuu. Ongelmia voi aiheuttaa myös runsas hakurobottiliikenne, jolloin verkkopalvelua indeksoidaan eri hakukoneiden toimesta. Osa robottiliikenteestä voi olla myös haitallista, jonka seurauksena orgaanisille käyttäjille varattu suorituskykykapasiteetti ei riitä palvelemaan sivuston käyttäjiä.

5 SUORITUSKYKYTESTAUS

Suorituskykytestaus on yleinen termi sellaiselle testaustoiminnalle, jonka tavoitteena on selvittää testattavan järjestelmän nopeutta, skaalautuvuutta tai sen vakautta kuorman alla. Suorituskykytestauksesta saatavaa tietoa voidaan hyödyntää muun muassa suorituskyvyn kehittämisessä, suorituskykyyn liittyvien pullonkaulojen tunnistamisessa, suorituskykyvaatimusten toteuttamisessa, suorituskyvyn vertailuarvojen luomisessa tai päätöksenteon tukena, kun kyseessä on järjestelmään tehtävät muutokset. (Meier ym. 2007.)

5.1 Suorituskykytestausprosessi

Perinteinen suorituskykytestaus voidaan kuvata prosessina (kuva 8), jonka ensimmäisessä kohdassa suunnitellaan testiympäristö, joka sisältää testattavan järjestelmän ja käytettävissä olevat testaustyökalut (1). Sen jälkeen arvioidaan suorituskykyyn liittyvät avainmittarit ja niiden tavoitearvot (2). Prosessin seuraavassa kohdassa suunnitellaan suorituskykytestaukseen liittyvät testitapaukset (3). Tämän jälkeen konfiguroidaan testiympäristö niin, että sen suorituskykyä voidaan monitoroida suorituskykytestauksen aikana (4). Käyttötapaukset ohjelmoidaan tai ne luodaan käytössä olevan testaustyökalun avulla (5). Lopuksi suorituskykytestit suoritetaan (6) ja niiden tulokset analysoidaan ja raportoidaan (7). Sama prosessi voidaan tarvittaessa toistaa uudelleen, jotta siitä saadaan ulos vertailukelpoisia tuloksia. (Meier ym. 2007.)

KUVA 8. Suorituskykytestausprosessin pääkohdat (Meier ym. 2007)

5.2 Suorituskykytestauksen alatyypit

Suorituskykytestaus voidaan jakaa erilaisiin alatyyppeihin riippuen siitä, mitä suorituskykytestauksella halutaan saavuttaa. Verkkosivuston suorituskykytestaus mittaa verkkokaupan nopeutta eli vasteaikaa. Vasteaikaan perustuvia suorituskykymittauksia voidaan hyödyntää muun muassa suorituskykyyn liittyvien oletusten ja tosiasioiden korrelointiin. Suorituskykytestauksen tuloksista saatuja vasteaikoja voidaan esimerkiksi verrata loppukäyttäjien odotuksiin suorituskyvyn osalta. Vasteaika on hyvä kehityksen mittari, kun verkkopalveluun tehdään suorituskykyyn liittyviä optimointeja tai muutoksia konfiguraatioon. Pelkän vasteajan mittaamisen ongelmana on se, että sen avulla ei löydetä virheitä, jotka esiintyvät verkkopalvelun kuormituksen seurauksena. Vasteaika ei aina kuvaa loppukäyttäjän kokemusta, sillä kaikkia käyttötapauksia on vaikea huomioida. Suorituskykytestauksen tulokset eivät välttämättä ole verrattavissa keskenään, jos testattavien ympäristöjen välillä löytyy eroavaisuuksia. (Meier ym. 2007.)

Kun halutaan selvittää, miten verkkopalvelu käyttäytyy kuorman alla, testausmenetelmänä käytetään kuormitustestausta. Kuormitustestauksen avulla saadaan selville, kuinka monta yhtäaikaista käyttäjää verkkosivu pystyy palvelemaan. Kuormitustestausta voidaan tehdä esimerkiksi ennen verkkopalvelun julkaisua, jotta saadaan selville kestääkö verkkopalvelu määrätyn määrän käyttäjiä. Kuormitustestauksesta on apua silloin, kun verkkopalvelun resursseja on tarkoitus mitoittaa uudelleen. Tulokset auttavat kapasiteettisuunnittelussa esimerkiksi silloin, kun verkkopalvelua skaalataan useammalle palvelimelle. Kuormitustestauksella löytyvät myös ne ongelmat, jotka tulevat ilmi ainoastaan silloin, kun verkkopalvelu on kuormituksen alla. Kuormitustestauksen pääasiallinen tarkoitus ei ole mitata verkkopalvelun vasteaikaa, joten se ei sovellu siihen tehtävään kovin hyvin. Kuormitustestauksesta saatuja tuloksia ei voida vertailla keskenään kahden eri järjestelmän välillä, sillä kapasiteetti kestää kuormaa on jokaisella järjestelmällä yksilöllinen. (Meier ym. 2007.)

Rasitustestaus, josta myös stressitestaukseksi kutsutaan, on testausmenetelmä, jonka tarkoitus on kuormittaa verkkopalvelua sen normaalista toiminnasta poiketen. Verkkopalvelua kuormitetaan niin paljon, että se ei enää toimi normaalisti. Tämä voi tarkoittaa tilannetta, jossa vasteajat kasvavat niin suureksi, että verkkopalvelu ei ole enää saavutetta-

vissa. Rasituksen alla verkkopalvelu voi rikkoutua monella eri tavalla, joten rasitustestauksen avulla voidaan selvittää kuinka verkkopalvelu käyttäytyy näissä rajatapauksissa. Ongelma voi esimerkiksi syntyä palvelinten kapasiteetin loppumisesta johtuvista syistä. Tietoturvan näkökulmasta tulokset antavat hyvät mahdollisuudet varautua esimerkiksi palvelunestohyökkäyksiin. Samalla saadaan ulos indikaattori, jonka avulla voidaan mahdollinen palvelunestohyökkäys tunnistaa jatkossa. Se tuo myös suunnitelmallisuutta siihen, millaisiin vikatilanteisiin on hyvä varautua etukäteen ja miten palvelunestohyökkäyksiä voidaan ennaltaehkäistä. Koska rasitustestauksella simuloidaan sellaisia rajatapauksia, joita ei mahdollisesti koskaan tapahdu, niin silloin se voidaan nähdä ajan hukkana. Rasitustestauksen luonteesta johtuen kasvanut liikennemäärä voi aiheuttaa ongelmia myös muualla kuin testattavassa järjestelmässä. (Meier ym. 2007.)

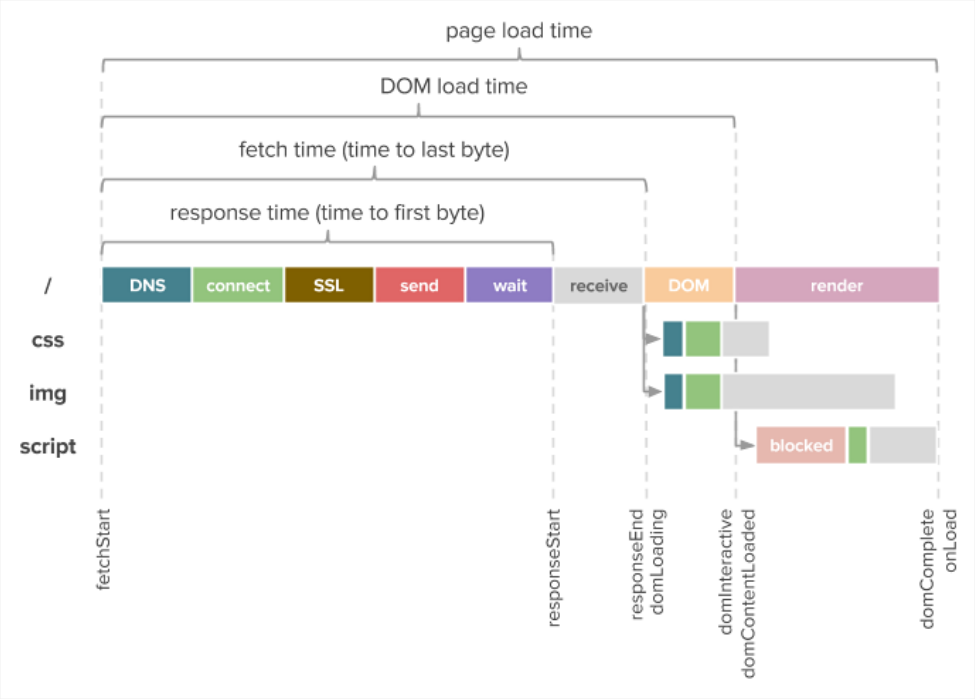
5.3 Suorituskyvyn mittarit

Suorituskyvyn mittarit on hyvä tunnistaa heti suorituskykytestausprosessin alussa. Tämä on tärkeää, että tiedetään mitä mitata ja millä menetelmällä. Suorituskyvyn mittarit on hyvä sopia yhdessä oleellisten sidosryhmien kanssa. Tärkein näistä on liiketoiminnan näkökulma, jonka avulla mittarit voidaan linjata liiketoiminnallisten tavoitteiden kanssa. Kun suorituskyvyn mittarit on valittu, niitä kutsutaan nimellä avainarvomittarit (eng. Key Performance Indicator) (Kivinen 2009, 30). Avainarvomittarit liittyvät johonkin tarkastelun alla olevaan asiaan, josta tarvitaan tarkempaa tietoa kokonaisuuden hahmottamiseen. Kun puhutaan suorituskykytestauksesta, avainarvomittarina käytetään usein vasteaikaa. Avainarvomittarit voivat vaihdella sen mukaan, mitä suorituskykytestausmenetelmää käytetään. Koska kehittämistehtävässä mitataan verkkopalvelun nopeutta, siitä syystä opinnäytetyössä käsitellään tarkemmin verkkopalvelun vasteaikaa.

Verkkopalvelun vasteaika kuvataan yksittäisenä nopeuden arvona, mutta todellisuudessa se voidaan jakaa pienempiin osiin. Ensimmäinen näistä arvioista on TTFB (Time to First Byte), jolla mitataan aikaa, joka kuluu sivupyynnön ensimmäisen tavun vastaanottamisesta. Ennen varsinaista sivupyyntöä suoritetaan DNS-kysely, joka kertoo, missä IP-osoitteessa domain sijaitsee. Sen jälkeen kohdepalvelimelle tehdään TCP- ja SSL-kättelyt (jos käytössä HTTPS-protokolla). Yhteyden muodostumisen jälkeen lähetetään itse sivupyyntö HTTP(S)-protokollan avulla ja sen jälkeen odotetaan palvelimelta vastausta. TTFB:n mittaukseen vaikuttaa verkkoliikenteen latenssi, verkkopalvelun ruuhkaisuus ja aika, joka kuluu sivupyynnön prosessointiin (Hoffman 2013). TTFB on yksinkertainen

tapa mitata vasteaikaa, kun tavoitteena on selvittää alustan suorituskykyä. Toinen vasteajan määritelmä on TTLB (Time to Last Byte). TTLB on muuten sama kuin TTFB, mutta siihen on lisätty aika, joka kuluu sivupyynnön vastaanottamiseen. Sitä voidaan hyödyntää suorituskyvyn mittarina rasitustesteissä, jotta saadaan selville, miten kuorma vaikuttaa sivupyynnön vastaanottamiseen.

Seuraavat vasteajan mittarit liittyvät enemmän selaimen toimintaan, joten ne ovat lähempänä loppukäyttäjän kokemaa vasteaikaa. Kun sivupyyntö on vastaanotettu, selain alkaa parsimaan DOM-puuta (Document Object Model), jonka mukaan ladataan verkkosivun ulkoiset elementit, kuten kuva- ja tyylitiedostot. Verkkosivun ja siihen liittyvien elementtien latautumiseen käytettyä aikaa kutsutaan DOM-latausajaksi (eng. DOM Load Time). Viimeinen vaihe liittyy verkkosivun lopulliseen lataamiseen selaimessa. Kun verkkosivu on täysin latautunut selaimeen, puhutaan sivun latausajasta (eng. Page Load Time). Mitä pidemmälle vasteajan mittauksessa edetään, sitä lähemmäksi tullaan loppukäyttäjän kokemaa vasteaikaa. Suorituskykytestauksen näkökulmasta tässä rajoituksena on usein käytössä oleva testaustyökalu. Jotta loppukäyttäjän kokemaa vasteaikaa voidaan mitata täydellisesti, tarvitaan testaustyökalu, jolla voidaan simuloida tai kutsua selaimia. Tästä käytetään termiä synteettinen suorituskykytestaus. (Kephart 2015.)



KUVA 9. Vasteajan eri mittausarvot ja niiden sisällöt (Kephart 2015)

5.4 Suorituskykytestaustyökalut

Suorituskykytestaustyökalut ovat olennainen osa suorituskykytestausta. Suorituskykytestaustyökalun tarkoitus on simuloida liikennettä testattavaan verkkopalveluun hyödyntämällä virtuaalisia käyttäjiä (Farrell-Vinay 2008, 282). Testaustyökalu raportoi myös virtuaalisten käyttäjätransaktioiden vasteajat ja mahdolliset virhetilanteet (Farrell-Vinay 2008, 282). Suorituskyvyn testaukseen käytettävät työkalut voivat olla valmiita ohjelmistoja — joko kaupallisia tai avoimeen lähdekoodiin perustuvia. Työkalut voivat olla täysin itse ohjelmoituja tai sitten ne on rakennettu jonkin testauskehyksen (eng. test harness) päälle. Testauskehyksen etu on, että se tarjoaa valmiin pohjan testien rakentamiseen ja niiden suorittamiseen. Testaustyökaluja ei kannata rakentaa itse, jos soveltuvia työkaluja tai komponentteja on valmiina käytettävissä (Loveland ym. 2005, 215).

Testaustyökalujen arvioinnissa tulee huomioida saavutettavan lisäarvon ja kustannusten välinen suhde. Kustannuksiin voi lukeutua esimerkiksi ohjelmistolisenssin hankinta tai testaustyökalun kehittämiseen ja käyttöönottoon liittyvä työmäärä. Lisäarvoa tuovat testaustyökalujen ominaisuudet, joilla testausprosessia voidaan nopeuttaa tai sen laatua parantaa. Nämä ominaisuudet kannattaa ottaa huomioon, kun arvioidaan uuden testaustyökalun käyttöönottoa. Testaustyökalun valinnassa tulisi ainakin kaksi näistä tavoitteista toteutua: kustannustehokkuuden kasvu, testausprosessin nopeuttaminen ja testauksen laadun parantaminen. (Loveland ym. 2005, 215-216.)

Testaustyökalujen avulla suorituskykytestausta voidaan automatisoida, jolloin aikaa jää itse suorituskykytestauksen suunnitteluun ja valmisteluun. Testaustyökalujen valinta on helpompaa, kun suorituskykytestauksen tavoitteet ovat selvät. Kun tiedossa on selkeä tavoite, myös työkalut tavoitteen saavuttamiseksi on helpommin määriteltävissä. Huomioitavaa on myös se, että testaustyökalut kehittyvät jatkuvasti eteenpäin. Jotta kehityksessä voidaan pysyä mukana, testaustyökaluja tulee jatkuvasti etsiä, tunnistaa ja evaluoida. Tässä kuitenkin piilee se riski, että aikaa kuluu enemmän suhteessa saavutettuihin hyötyihin. Testaustoiminnassa tulee muistaa sen perustavoitteet, joita testaustyökalujen tulee palvella. Liiallinen työkalupainotteisuus voi haitata itse testaustoimintaa, jos työkalut vaihtuvat jatkuvasti ja aikaa käytetään enemmän testaustyökalujen evaluointiin kun itse testaustyöhön. (Loveland ym. 2005, 215-216.)

6 SUORITUSKYKYTESTAUKSEN TOTEUTTAMINEN

Toimeksiantajan laadulliset vaatimukset huomioon ottaen on selvitetty, miten ja millä työkaluilla suorituskykytestausta tehdään. Suorituskykytestaustyökalulta odotetaan, että sillä voidaan simuloida palveluun liikennettä. Suorituskykytestaustyökalun pitää myös pystyä simuloimaan loppukäyttäjää siltä osin, että sillä voidaan suorittaa vastaavia käyttötapauksia kuten ostosten lisäämistä ostoskoriin. Näin parannetaan suorituskykytestauksen kattavuutta ja tällöin se vastaa paremmin reaalimaailman toimintaa. Ratkaisun käyttöönotettavuutta pyritään helpottamaan siten, että valitaan sellaiset työkalut, jotka ovat helposti otettavissa käyttöön. Tarkoitus on siis valita mahdollisimman valmis ratkaisu, jolloin omaa kehitystä tarvitsee tehdä mahdollisimman vähän. Muokattavuus voidaan nähdä niin, että suositaan avointa lähdekoodia, sillä se tarjoaa miltei vapaan muokattavuuden verrattuna suljettuihin ratkaisuihin. Tämä tarkoittaa myös kustannustehokkuutta verrattuna kaupallisiin ratkaisuihin vaikkakin erillisiä kustannusvaikutuksia tässä ei ole huomioitu.

Suorituskykytestauksen toistettavuus voidaan tulkita niin, että toteutettu ratkaisu pitää pystyä automatisoimaan mahdollisimman pitkälle. Manuaaliset työvaiheet vähentävät toistettavuutta ja kasvattavat virheiden mahdollisuuksia. Inhimilliset virheet voivat vääristää testaustuloksia ja niistä tehtyjä johtopäätöksiä. Koska kyseessä on kompleksisen järjestelmän testaaminen, testiautomaation toteuttaminen vaatii alkuun suuren kehityspanoksen. Jotta kehityspanos saadaan tehokkaasti hyödynnettyä, työkalujen valinnassa tulee huomioida, kuinka hyvin ne tukevat automatisointia. Käytännössä se tarkoittaa ohjelmallisesti kutsuttavien rajapintojen olemassaoloa ja niiden hyödyntämistä. Toinen vaihtoehto on komentorivikäyttöliittymä, jolloin ohjelmaa voidaan ajaa skriptien avulla. Skriptillä tarkoitetaan pientä apuohjelmaa, joka sisältää valmiiksi automatisoidun työnkulun. Suorituskykytestauksen tulokset tulee olla helposti luettavia ja helpoiten tämä saavutetaan datan visualisoinnilla. Ratkaisun tulee pystyä visualisoimaan suorituskykytestauksesta saatavaa numeerista dataa, jotta sitä on helpompi vertailla.

6.1 Valitut testaustyökalut

Suorituskykytestaus tehdään käyttämällä JMeter-testaustyökalua. JMeter on verkkopalveluiden suorituskykytestaukseen erikoistunut työkalu. JMeter valittiin testaustyökaluksi, sillä se on kustannustehokas vaihtoehto ja sen avulla suorituskykytestauksen käyttöönotto

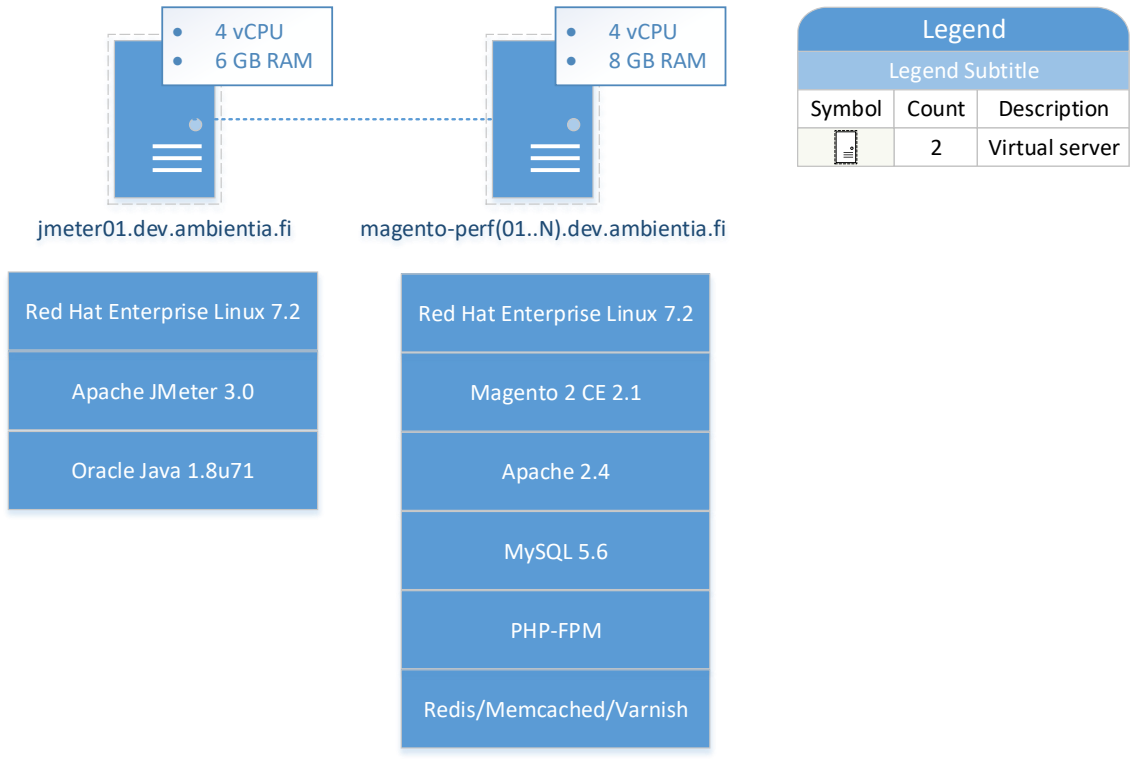
tapahtuu nopeasti. JMeteriä on julkaistu avoimen lähdekoodin lisenssillä ja sen kehittäjäyhteisönä toimii Apache Software Foundation. JMeter-testaustyökalu on Java-pohjainen ja sitä voidaan ajaa joko komentoriviltä tai graafisesta käyttöliittymästä. Suorituskykytestaus voidaan täten ottaa osaksi jatkuvan integraation työnkulkua. Näin voidaan hyvissä ajoin havaita sovelluksen muutoksista aiheutuva suorituskyvyn regressio. Työkalun avulla voidaan simuloida palveluun tulevaa liikennettä. Tämä tapahtuu nauhoittamalla käyttäjän toimintoja, jotka voidaan ohjelmallisesti laukaista uudelleen. Virtuaalisia käyttäjiä voidaan ajaa rinnakkain useampia, jolloin voidaan testata, kuinka monta yhtäaikaista käyttäjää verkkopalvelu kestää. Tällöin puhutaan kuormitustestauksesta, jonka tavoite on eri kuin suorituskykytestauksen. Koska tarkoituksena on käyttää JMeteriä suorituskykytestaukseen, niin riittää, että virtuaalisia käyttäjiä on vain muutama.

Magentolle löytyy valmiina JMeter-testausprofiili, joten sitä ei ole tarvetta rakentaa erikseen. Testausprofiili sisältää valikoiman testitapauksia, joiden avulla suorituskykytestauksesta saadaan tarpeeksi kattava. Toimintoihin kuuluu muun muassa palveluun kirjautuminen, tuotteiden lataaminen ja lisääminen ostoskoriin sekä tuotteiden tilausprosessi. Valmiiksi hyödynnettävän testausprofiilin olemassaolo oli pääasiallinen syy siihen, miksi testaustyökaluksi valittiin JMeter. JMeterin käyttöliittymä mahdollistaa tulosten esittämisen graafisesti eri muodoissa. JMeteriä on kuitenkin tarkoitus ajaa komentoriviltä, jolloin tulosten esittämiseen tarvitaan toinen työkalu. Tähän on valittu Gnuplot, joka vastaavasti on myös komentorivipohjainen työkalu. JMeterin suorituskykytestien tulokset on mahdollista saada ulos CSV-formaatissa, jonka jälkeen tuloksia voidaan käsitellä toisessa työkalulla. Gnuplot tarjoaa datan visualisointiin erilaisia diagrammeja, joista on valittu perinteinen pylväsdiagrammi. Pylväsdiagrammien avulla voidaan helposti suorittaa suorituskykyvertailu kahden tai useamman muuttujan välillä.

6.2 Testausympäristö

Testausympäristö (kuva 10) koostuu yhdestä tai useammasta virtuaalipalvelimesta, jotka ajetaan samassa virtuaali-infrastruktuurissa, jossa sijaitsee myös tuotantopalvelut. Näin testattava järjestelmä vastaa tuotantoympäristöä alla olevan laitteiston ja tietoliikenteen osalta. JMeter on asennettu omalle virtuaalipalvelimelle, jotta suorituskykytestien ajaminen ei vaikuttaisi Magenton toimintaan. Magentolle dedikoidun virtuaalipalvelimen resurssit vastaavat keskisuuren verkkokaupan vaatimuksia (4 vCPU ja 8 Gt RAM). Käyt-

töjärjestelmänä toimii Red Hat Enterprise Linux 7.2, ellei toisin mainita. Web-palvelimena käytetään Apache 2.4:sta ja tietokantana MySQL 5.6:sta. PHP on asennettu toimimaan FPM-moodissa (FastCGI Process Manager), joka ajaa PHP:ta omassa prosessissaan. Virtuaalipalvelimen konfiguraation hallintaan käytetään Puppet-työkalua. Puppetin avulla palvelimen konfiguraatiot voidaan monistaa, joka auttaa uusien testausympäristöjen pystyttämisessä. Käyttämällä konfiguraation hallintaa, on mahdollista vähentää inhimillisten virheiden määrä. Palvelimelle asennettava verkkokauppa on Magento Community Edition versio 2.1. Jotta Magento vastaisi toiminnaltaan tuotannossa olevaa verkkokauppaa, sinne täytyy saada myös sisältöä. Tätä varten Magentossa itsessään on sisäänrakennettu toiminnallisuus, jonka avulla voidaan ohjelmallisesti luoda sisältöä verkkokauppaan. Sisällön luonti pohjautuu erilaisiin profiileihin, jotka määräävät verkkokaupan koon. Näistä profiileista valittiin keskisuuri verkkokauppa, joka sisältää muun muassa 17000 tuotetta, 300 kategoriaa, 200 asiakastiliä ja 1600 tilausta.



KUVA 10. Testausympäristön kuvaus ja asennetut sovelluskomponentit

6.3 Testausskenaariot

Suorituskykymittauksia varten on rakennettu erilaisia testausskenaarioita, joiden avulla pyritään selvittämään kehitetyn mittausratkaisun toimivuutta ja hyödyllisyyttä käytännössä. Skenaariot perustuvat toimintaympäristössä havaittuihin ongelmiin ja niistä syntyneisiin kysymyksiin. Tarkoitus on pohjustaa testausskenaarioiden taustat, jotta lukija saa tarkemman kuvan selvitettävän ongelman laadusta ja sen vaikutuksista. Tulosten pohjalta tehdään suorituskykyvertailu, jonka avulla pyritään saamaan selville suorituskykyisin kokoonpano. Suorituskykyvertailusta odotetaan olevan hyötyä Ambientian verkkokauppaliiketoiminnan palveluiden ja toiminnan kehittämiseen. Jotta suorituskykyvertailusta saadaan mahdollisimman yksiselitteinen ja helposti tulkittava, jokaisessa testausskenaariossa muutetaan yhtä sovelluskomponenttia tai konfiguraatiota kerrallaan. Testausskenaarioiden kulku ja käytetyt menetelmät pyritään kuvaamaan sillä tarkkuudella, että tulokset voidaan tarvittaessa toistaa.

6.3.1 Linux-käyttöjärjestelmien vaikutus vasteaikoihin

Tämä testauskenaario on rakennettu sen olettamuksen pohjalta, että käytössä oleva Red Hat Enterprise Linux -käyttöjärjestelmä ei ole Magento-sovelluspinon suorituskyvyn kannalta paras mahdollinen alusta verrattuna muihin Linux-jakelupaketteihin. Kehittäjillä on ollut hyviä kokemuksia Debian Linuxista, joka on empiiristen havaintojen perusteella suoriutunut RHEL-käyttöjärjestelmää nopeammin. Havainnot ovat pohjautuneet kuitenkin henkilökohtaisiin kokemuksiin ja ovat täten subjektiivisia. Testausskenaarion tarkoitus on selvittää pitääkö tämä olettamus paikkansa. Vastaus tähän perustuu tietoon, joka saadaan suorituskykyvertailun tuloksena. Yksi mahdollinen Magento-sovelluspinon suorituskykyyn vaikuttava seikka johtuu Red Hatin päivityspolitiikasta. Red Hatin päivityspolitiikka korostaa käyttöjärjestelmän vakautta ja tuettavuutta uusien versioiden ja ominaisuuksien kustannuksella.

Käyttöjärjestelmän mukana tulevien ohjelmistojen versiot on lukittu RHEL:n julkaisuversioon (nk. fixed release). Tietoturvapäivitykset kuitenkin tehdään taaksepäin yhteensopivasti käytössä oleviin ohjelmistoversioihin (Red Hat Security Backporting Policy n.d.). Tämä järjestely takaa sen, että RHEL-käyttöjärjestelmällä on suhteellisen pitkä, yli kymmen vuoden elinkaari (Red Hat Enterprise Linux Life Cycle n.d.). Tämä kuitenkin

tarkoittaa sitä, että uusia ohjelmaversioita ja ominaisuuksia saadaan vasta uuden julkaisuversion myötä. Red Hatin tapauksessa se tarkoittaa keskimäärin neljää vuotta eri julkaisuversioiden välillä. Kehityksen kannalta neljä vuotta on pitkä aika, mutta onneksi Red Hatilla on kuitenkin tarjolla erillinen RHSCL-kanava (Red Hat Software Collections n.d.), joka mahdollistaa uudempien ohjelmaversioiden asentamisen käyttöjärjestelmän rinnalle. RHSCL-kanavalta tulevilla paketeilla on kuitenkin käyttöjärjestelmää lyhempi elinkaarimalli (Red Hat Software Collections Product Life Cycle n.d.).

Vaihtoehtona on käyttää sellaista Linux-jakelua, jonka julkaisusykli on nopeampi. Näin saadaan uudempia ohjelmaversioita nopeammin käyttöön. Esimerkiksi Debian Linuxin julkaisusykli on 2 vuotta (Debian Releases 2016). On myös olemassa nk. rolling release -tyyppisiä Linux-jakeluita, joissa ei ole erillisiä julkaisuversioita. Ohjelmistot päivittyvät sen mukaan, kun uusia versioita julkaistaan. Jatkuvan julkaisun periaatteella toimivia Linux-jakeluita ovat muun muassa Arch Linux ja Gentoo. Niitä harvemmin näkee yrityskäytössä, sillä niille ei ole saatavilla vastaavaa Red Hatin tukimallia. Tämä ei kuitenkaan ole se suurin ongelma vaan se, että uusia ohjelmaversioita ei testata perinpohjaisesti ennen niiden käyttöönottoa. Epävakaat ohjelmaversiot voivat sisältää ohjelmistovirheitä ja yhteensopivuusongelmia, jotka vaikuttavat negatiivisesti järjestelmän toimintaan. Nopean julkaisusyklin Linux-jakelujen elinkaari on verraten lyhyt ja se tarkoittaa sitä, että käyttöjärjestelmä tulee päivittää uuteen julkaisuversioon ennen elinkaaren päättymistä. Ylläpidon näkökulmasta se tarkoittaa päivityksistä johtuvaa lisätyötä, jotta käyttöjärjestelmä pysyisi tuettuna. Kun käyttöjärjestelmän elinkaari päättyy, uusia päivityksiä ei ole saatavilla. Tämä on varsinkin tietoturvan kannalta tärkeä asia.

Jotta saadaan selville käyttöjärjestelmän vaikutus Magento-verkkokaupan suorituskykyyn, on tähän valittu neljä eri Linux-käyttöjärjestelmää: Red Hat Enterprise Linux 7.2, Debian 8, Ubuntu 16.04 LTS ja Gentoo. RHEL on Ambientian oletuskäyttöjärjestelmä ja se pohjautuu Fedora Linuxiin. Ubuntu sen sijaan pohjautuu Debianiin ja molemmat niistä ovat tällä hetkellä suosituimpia Linux-käyttöjärjestelmiä palvelinkäytössä (Usage statistics and market share of Linux for websites 2017). Gentoo edustaa tässä ainoata rolling release –tyyppistä Linux-jakelua. Käyttöjärjestelmät konfiguroidaan mahdollisimman samalla tavalla ja niiden päälle asennetaan identtinen Magento-sovelluspino. Eri käyttöjärjestelmillä ajettaviin Magento-verkkokauppaympäristöihin tehdään samanlaiset suorituskykytestit, joiden perusteella toivottavasti selviää vaikuttaako käyttöjärjestelmä verkkokaupan nopeuteen.

6.3.2 PHP-versiopäivityksen vaikutus vasteaikoihin

Seuraavan testausskenaarion tavoitteena on selvittää, kuinka paljon PHP:n versiopäivitys vaikuttaa Magento-verkkokaupan suorituskykyyn. Tarkoituksena on päivittää palvelimen PHP lähtöversiosta 5.6 versioon 7.0. Päivitys tehdään kaikille neljälle valitulle käyttöjärjestelmälle. Suorituskykytestaus tehdään ensin PHP 5.6:lla ja sen jälkeen PHP 7.0:lla, jolloin saadaan selville suorituskykyyn liittyvä muutos. PHP:n virallisen julkaisu-uutisen mukaan odotettavissa on jopa kaksinkertainen suorituskyvyn parannus verrattuna edelliseen versioon (PHP 7.0.0 Release Notes 2015). Uutisessa ei kuitenkaan mainita sitä, millä menetelmillä suorituskyvyn muutos on todistettu. Suorituskyvyn käyttäytymisestä kompleksisessa verkkopalvelussa ei siis ole täyttä varmuutta. Amastyn laatima Magenton suorituskykyraportti antaa tähän hyödyllisempää tietoa. Raportin mukaan PHP 7.0 kasvattaa Magenton suorituskykyä keskimäärin 50 % (Tataranovich 2017). Amastyn raportin perusteella PHP kannattaisi päivittää, sillä pienikin suorituskyvyn parannus vaikuttaa myönteisesti myös verkkokaupan käytettävyyteen.

PHP:n päivittäminen RHEL:ssa ei ole kuitenkaan yksinkertaista, sillä uusin julkaisuversio (RHEL 7.3) tarjoaa korkeintaan vain version 5.6 erillisen RHSCL-kanavan kautta. Sen vuoksi joudutaan turvautumaan toisen osapuolen ohjelmistopaketteihin. Ongelma tässä on se, että Red Hat ei tarjoa tukea ulkopuolisiin ohjelmistopaketteihin, joita ei ole ladattu virallisista lähteistä. Kirjoitushetkellä Red Hat ei tarjoa PHP 7.0:aa virallisilta kanavilta, mutta se on kaavailtu julkaistavaksi RHSCL:n 2.3 versiossa. Vaihtoehtoina ovat käyttöjärjestelmän ja sen sisältämien ohjelmistojen osittainen tuen menettäminen tai suorituskyvyn paraneminen. Tilanne on kaikin puolin ongelmallinen ja päätöksenteon tueksi tarvitaan tarkempaa tietoa, jonka tämän testausskenaarion toivotaan tarjoavan. Muihin valittuihin Linux-jakelupaketteihin PHP 7.0 on saatavilla ongelmitta, mutta niihin ei ole tarjolla vastaavaa valmistajan tukimallia, kuin Red Hatilla.

6.3.3 Magenton sisäisten välimuistien vaikutus vasteaikoihin

Tämän testausskenaarion tavoitteena on selvittää kuinka suuri osa Magento-verkkokaupan suorituskyvystä johtuu sisäisten välimuistien hyödyntämisestä. Tämä muuttuja halutaan eliminoida, jotta saadaan selville vasteaika, joka perustuu Magenton puhtaaseen suorituskykyyn. Samalla selviää, ovatko muiden suorituskykytestien tulokset vertailukelpoisia, vaikka niissä välimuistit ovatkin käytössä. Magento 2:ssa on yhteensä 12 kappaletta

erilaisia välimuistityyppejä (Magento DevDocs: Manage the cache n.d.). Jokainen näistä vaikuttaa suorituskykyyn vaihtelevalla tavalla. Tarkoitus ei ole kuitenkaan selvittää yksittäisten välimuistien vaikutusta suorituskykyyn, sillä ne kaikki otetaan pois käytöstä. Näin saadaan selville, miten Magenton vasteaika käyttäytyy ilman välimuistikerroksia. Todennäköistä on, että palvelinresurssien käyttöaste ja vasteaika tulee kasvamaan huomattavasti. Välimuistikerroksen teho perustuu siihen, että kertaalleen prosessoitu tieto tallennetaan välimuistiin, josta se on nopeasti saatavilla seuraavalla kerralla. Magento tallentaa välimuisteihinsa erilaista tietoa, kuten asetuksia ja arvoja, jolloin niitä ei tarvitse erikseen hakea tietokannasta. Useat rinnakkaiset tietokantakyselyt ovat suorituskyvyn kannalta kalliita operaatioita, sillä tietoa joudutaan hakemaan levyltä, aiheuttaen kuormaa palvelimelle. Tietoa tallentuu välimuistiin ensimmäisen sivupyynnön yhteydessä, jolloin havaittavissa on vasteajan hetkellinen nousu. Seuraavalla sivunlatauksella vasteaika laskee, sillä sivun uudelleenprosessoinnista aiheutuvaa viivettä ei synny.

Magenton toimintaa voidaan tehostaa myös sovelluksen ulkopuolisilla välimuistiratkaisuilla. Yksi näistä on Varnish-välimuistipalvelin, joka tallentaa välimuistiin staattisia sivuja ja osia dynaamisista sivuista. Käyttäjäkohtaiset dynaamiset sisältöalueet merkitään erillisillä ESI-tageilla (Edge Side Includes), jotta tiedot eivät päädy välimuistiin ja sitä kautta muille käyttäjille. Magenton sisäisistä välimuisteista poiketen Varnish toimii palvelun edessä ulkoisena sivuvälimuistikerroksena. Magenton oma logiikka määrittelee, kuinka pitkään sivu pysyy välimuistissa. Muutosten yhteydessä Magento tyhjentää sivut välimuistista, jotta palvelu ei tarjoaisi vanhentunutta sisältöä. PHP-ohjelmakoodin suoritusta voidaan vielä nopeuttaa OpCache-lisäosalla. Ensimmäisen suorituksen yhteydessä PHP-ohjelmakoodi käännetään tavukoodiksi palvelimen välimuistiin. Seuraavan suorituksen yhteydessä välimuistissa oleva tavukoodi suoriutuu nopeammin, vaikka se sisältää vastaavan logiikan. Suorituskyky syntyy siitä, että palvelin suorittaa välimuistissa tavukoodia nopeammin verrattuna tulkattavaan ja selkokieliseen ohjelmakoodiin (Bolton 2014).

6.3.4 Klusteroidun ympäristön vaikutus vasteaikoihin

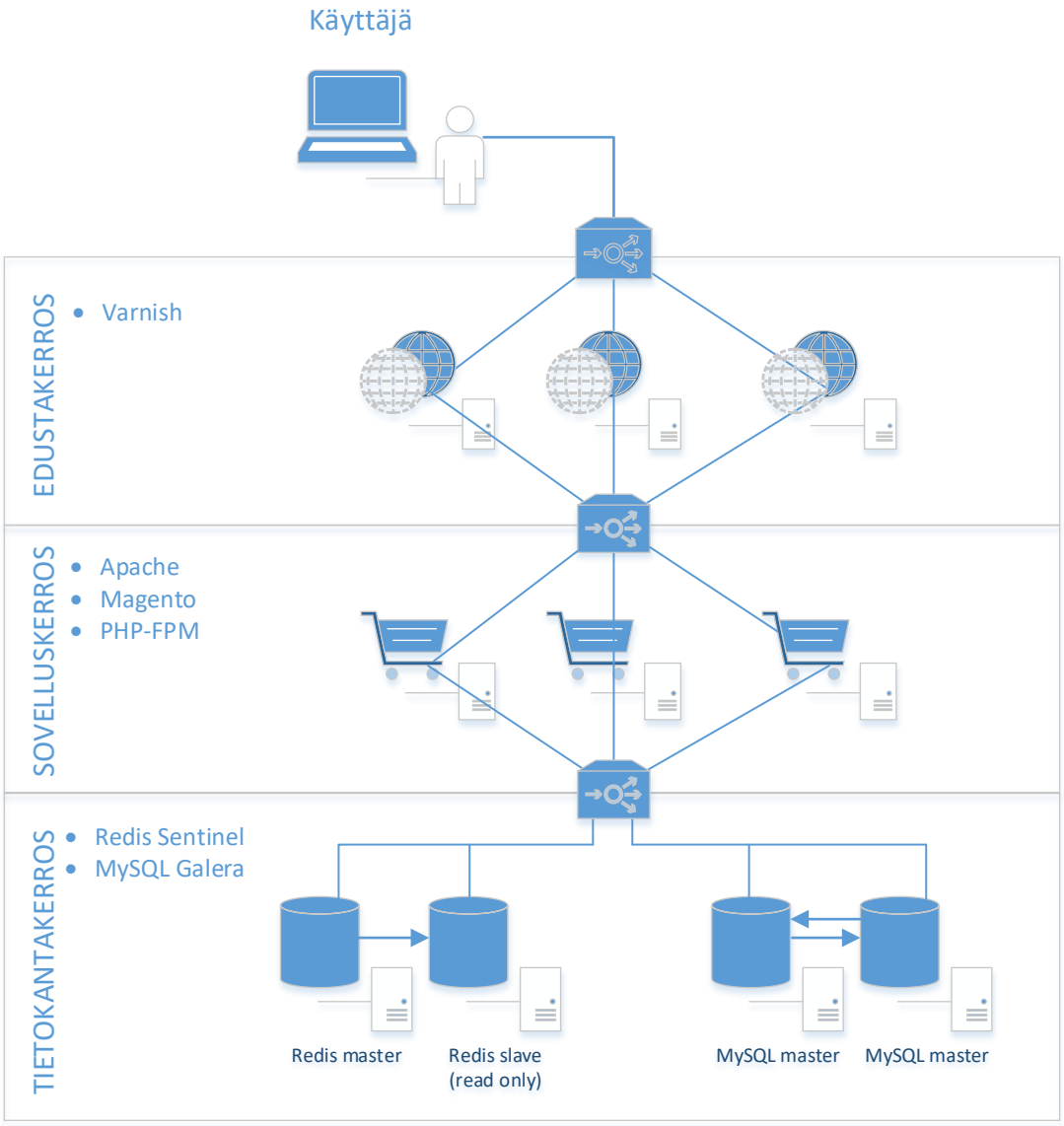
Kun asiakkaiden määrittämät palvelutasosopimukset ovat vaatineet parempaa saavutettavuutta tai suorituskykykapasiteettia, Magento-palvelinympäristöjen arkkitehtuuria on jouduttu suunnittelemaan uudelleen. Magenton suorituskyky laskee, kun palvelimen resurssit loppuvat kesken. Yhden palvelimen ajoympäristöä voidaan skaalata vertikaalisesti

kasvattamalla resursseja. Vastaan tulee kuitenkin yhden palvelimen laskentatehon ja muistimäärän maksimirajat. Virtuaalisessa ympäristössä vertikaalinen skaalautuminen on helpompaa kuin fyysisissä palvelinympäristössä, sillä resursseja voidaan lisätä ajon aikana. Yksittäisen palvelimen rajoitteista päästään eroon muuttamalla järjestelmäarkkitehtuuria niin, että sovelluskomponentit hajautetaan eri palvelimille. Magentolle toimiva tapa on tehdä kolmikerroksinen järjestelmäarkkitehtuuri (eng. three-tier architecture), jossa eriteltyinä ovat edusta-, sovellus- ja tietokantapalvelimet (kuva 11). Edustapalvelin ottaa vastaan liikenteen, purkaa mahdollisen SSL/TLS-salauksen ja toimii usein staattisten sivujen välimuistivarastona. Edustapalvelimelta liikenne välitetään sovelluspalvelimelle, joka sisältää itse sovelluslogiikan. Tietokantapalvelin toimii nimensä mukaisesti dynaamisesti muuttuvan datan tietovarastona. Kolmikerrosarkkitehtuurin etuna on se, että eri palvelimille hajautetut sovelluskomponentit eivät kilpaile resursseista toistensa kanssa. Liikennemäärän kasvaessa myös hajautetussa palvelinarkkitehtuurissa tulee yksittäisen palvelimen resurssirajat vastaan ennen pitkään.

KUVA 11. Esimerkkikuva kolmikerroksisesta järjestelmäarkkitehtuurista (3-tier)

Seuraava kehitysaskel on skaalata kolmikerroksista järjestelmäarkkitehtuuria horisontaalisesti, joka vaatii palvelinympäristön klusteroinnin. Myös sovelluksen tulee tukea klusterointia siinä määrin, että se pystyy jakamaan tilatietoja klusterin eri jäsenille. Tilatiedolla tarkoitetaan esimerkiksi käyttäjäsessioita, välimuisteja ja sovelluksen jaettua dataa. Magenton osalta sessiot ja välimuistit on mahdollista säilyttää jaetussa tietovarastossa, jota klusterin kaikki sovelluspalvelimet pääsevät käyttämään. Sovellusten staattinen sisältö, kuten kuvatiedostot, on mahdollista jakaa klusterin jäsenten kesken erillisessä jaetussa tiedostojärjestelmässä. Klusteroidun ympäristön tärkeä komponentti on kuormanjakaja, joka jakaa liikennettä erillisen kuormanjakoalgoritmin avulla. Kuormanjaon myötä ympäristön suorituskapasiteetti kasvaa ja yksittäisten palvelinten resurssirajoituksesta päästään eroon. Klusteroidussa palvelinympäristössä (kuva 12) jokaisessa kerroksessa voi olla rinnakkain useampia palvelimia, jotka tasaisesti jakavat kuorman. Korkeasti saavuttava (eng. highly available) ympäristö toteutuu silloin, kun jokaisessa kerroksessa on

vähintään kaksi tai useampi palvelin vastaanottamassa liikennettä. Tietokantojen osalta tämä tarkoittaa joko master-slave-replikointia tai multi-master-toteutusta.



KUVA 12. Esimerkkikuva klusteroidusta järjestelmäarkkitehtuurista (N-tier)

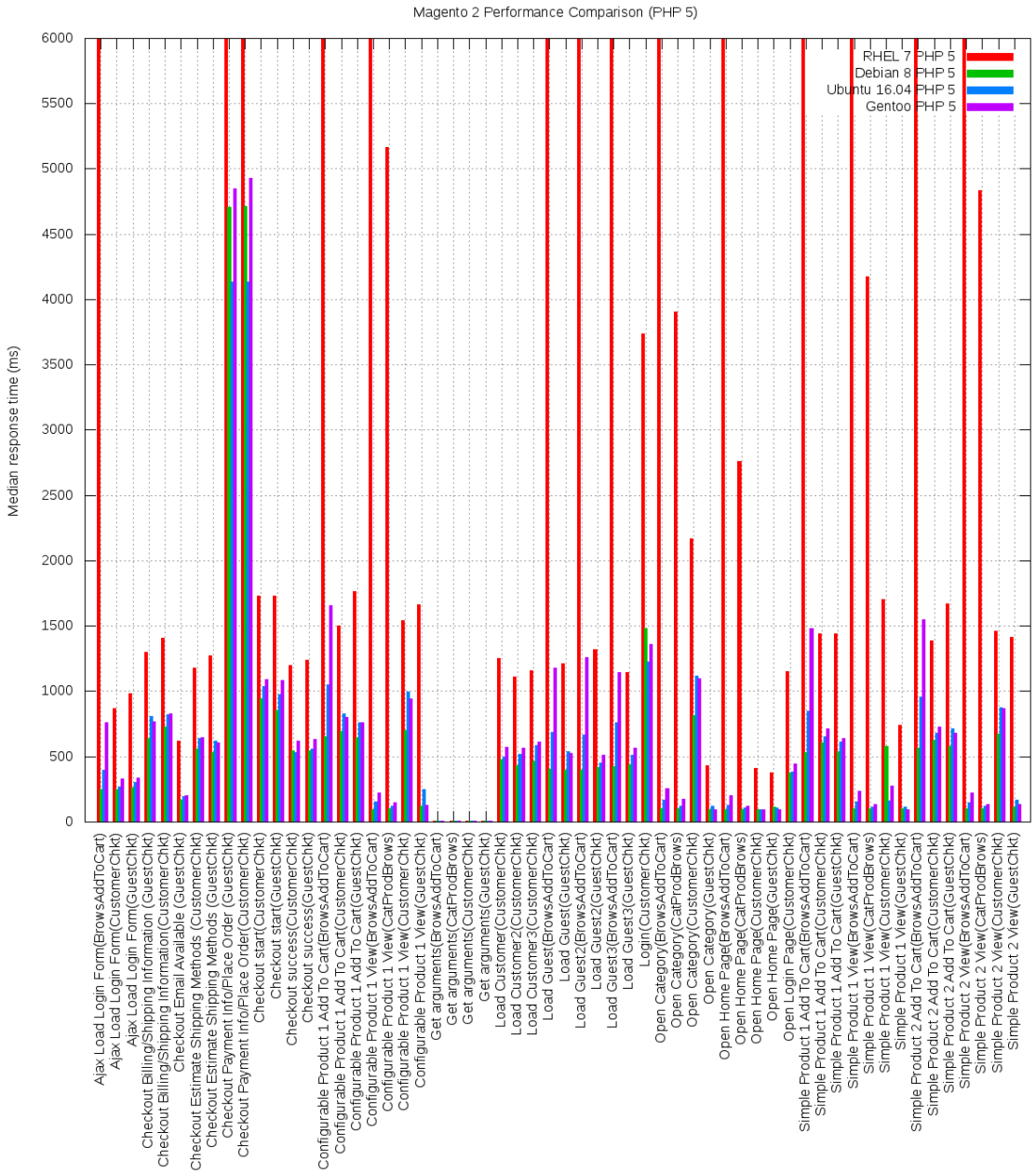
Testausskenaarion tarkoituksena on selvittää, kuinka klusteroidun Magento-ympäristön suorituskyky käyttäytyy, kun sivuvälimuistin (FPC) ja sessioiden tallentamiseen liittyviä tietovarastoja muutetaan. Suorituskykykapasiteettia ei tässä skenaariossa tulla selvittämään, sillä silloin tulisi suorittaa järjestelmän rasitustestaus. Vaikka JMeter-testaustyökalulla tämä voitaisiin tehdä, niin silti tavoitteena tässä on saada selville, millä sivuvälimuistin ja sessioiden tallennusratkaisulla saavutetaan nopeimmat vasteajat. Klusteroidussa ympäristössä sessioiden tallentamiseen voidaan käyttää muistinvaraisia

Memcached- tai Redis-tietokantoja. Sivuvälimuistin (FPC) tallennukseen voidaan sen sijaan käyttää Varnish-välimuistipalvelinta tai Redis-tietokantaa. Kaikki vertailussa olevat tallennusratkaisut on konfiguroitu käyttämään muistinvaraista tallennustilaa, jolloin suorituskyvyn vertailu on mahdollisimman tasaveroinen.

6.4 Suorituskykyvertailu

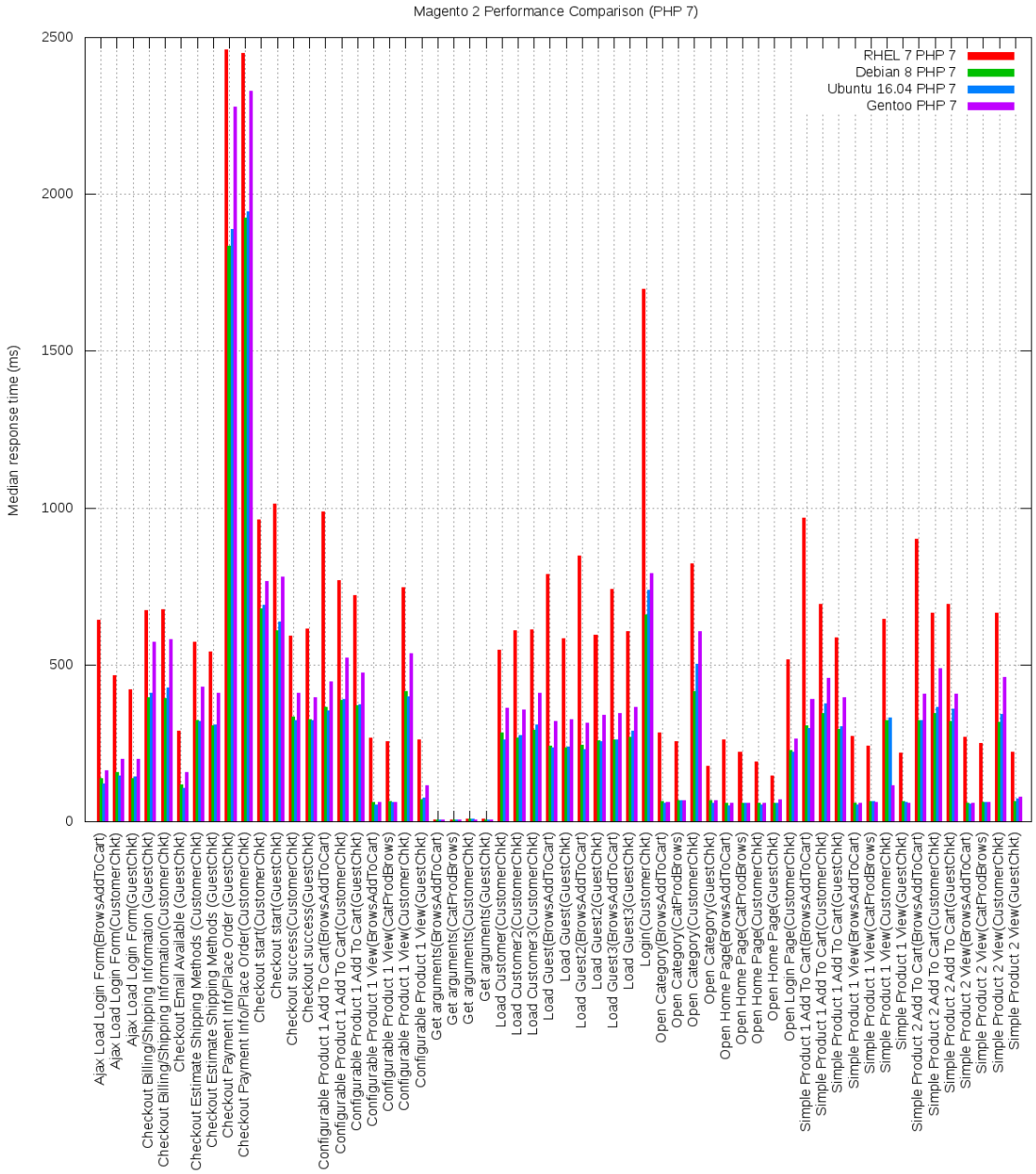
Seuraavassa käsitellään testausskenaarioiden tulokset ja niiden suorituskykyvertailut. Tulokset esitetään pylväsdiagrammeina, jotka on luotu suorituskykytestien numeerisesta datasta. Suorituskykytestin aikana suoritetut käyttötapaukset näkyvät pylväsdiagrammin X-akselilla. Erilaisia käyttötapauksia on yhteensä n. 60 kappaletta ja ne kuvaavat verkkokaupan käyttäjien suorittamia toimintoa. Jotkin monimutkaisemmat toiminnot, kuten tilauksen tekeminen ja palveluun kirjautuminen, ovat normaalisti hitaampia verrattuna staattisten sivujen vasteaikoihin. Testattavat käyttötapaukset kattavat Magento-verkkokaupan keskeisimmät toiminnot asiakkaan näkökulmasta. Verkkokaupan ylläpitäjän näkökulmaa käyttötapauksissa ei ole huomioitu. Käyttötapauksia tulisi laajentaa, jos mukaan halutaan verkkokaupan ylläpitoon tarkoitettu hallintapaneeli. Jokainen testitapaus on suoritettu useita kymmeniä kertoja, jolloin satunnaiset ääripään tulokset eivät vaikuta lopputulokseen dramaattisesti. Pylväsdiagrammin Y-akselilla nähdään keskiarvo sivunlatauksen tai toiminnon suorittamisen vasteajasta. Vasteaika tässä tapauksessa tarkoittaa millisekunneissa sitä aikaa, joka kuluu yhteyden muodostamisesta ensimmäisen tavun vastaanottamiseen. Toisin sanoen mittausarvona käytössä on TTFB, joka tarjoaa kaiken tarvittavan informaation liittyen sivupyynnön prosessointiin kuluvasta ajasta.

Kuvasta 13 selviää kuinka Linux-käyttöjärjestelmän valinta vaikuttaa Magenton vasteaikoihin, kun käytössä on PHP 5.6. Suorituskykyvertailun mukaan Magenton vasteajat ovat huonoimpia RHEL-käyttöjärjestelmällä. Osassa käyttötapauksissa vasteaika kasvoi jopa yli vertailuasteikon (6 sekuntia). Muut Linux-käyttöjärjestelmät suoriutuivat testistä tasaisemmin ja niiden välillä ei suuria eroja syntynyt. Magenton eri käyttötapauksista saatujen vasteaikojen keskiarvon perusteella nopein vasteaika saavutettiin Debian Linuxilla.



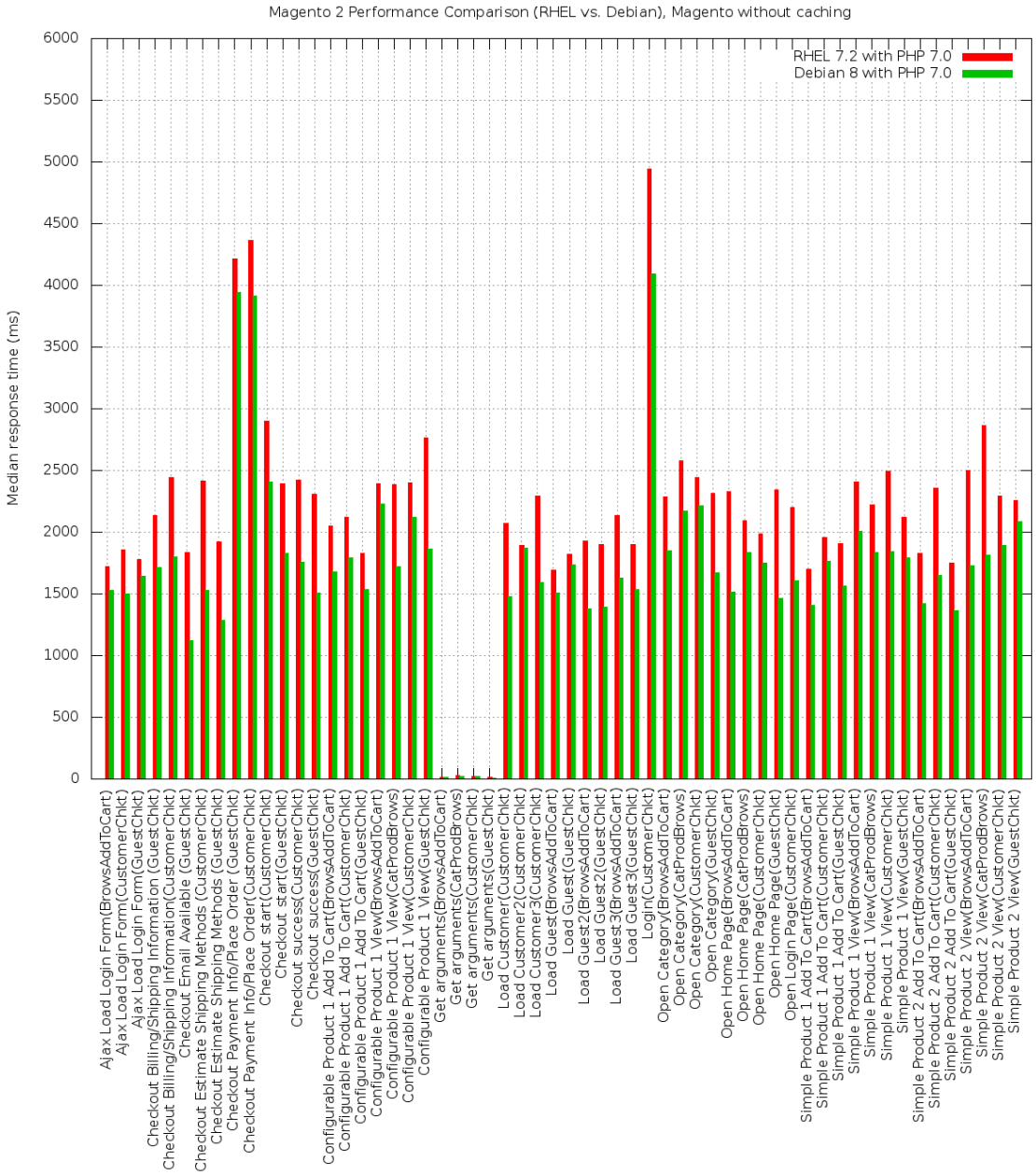
KUVA 13. Eri Linux-käyttöjärjestelmien vaikutus Magenton vasteaikoihin, kun käytössä PHP 5.6

Kuvasta 14 selviää kuinka PHP 7.0 päivityksen jälkeen vasteajat paranivat kaikissa Linux-käyttöjärjestelmissä. Kuten edellisessä testiskenaariossa, niin myös tässä RHEL oli vertailun hitain. Toisaalta kuitenkin suurin prosentuaalinen suorituskyvyn parannus tapahtui RHEL:lla. Muiden prosentuaalinen suorituskyvyn kasvu oli odotettu ~50 %, kun taas RHEL:ssa se oli useita satoja prosentteja. PHP 7.0 liittyvää suorituskyvyn parannusta osattiin jo odottaa Amastyn tekemän suorituskykymittauksen perusteella. Kuten edellisessä vertailussa, niin myös tässä Debian Linux pärjäsi parhaiten vasteaikojen osalta.



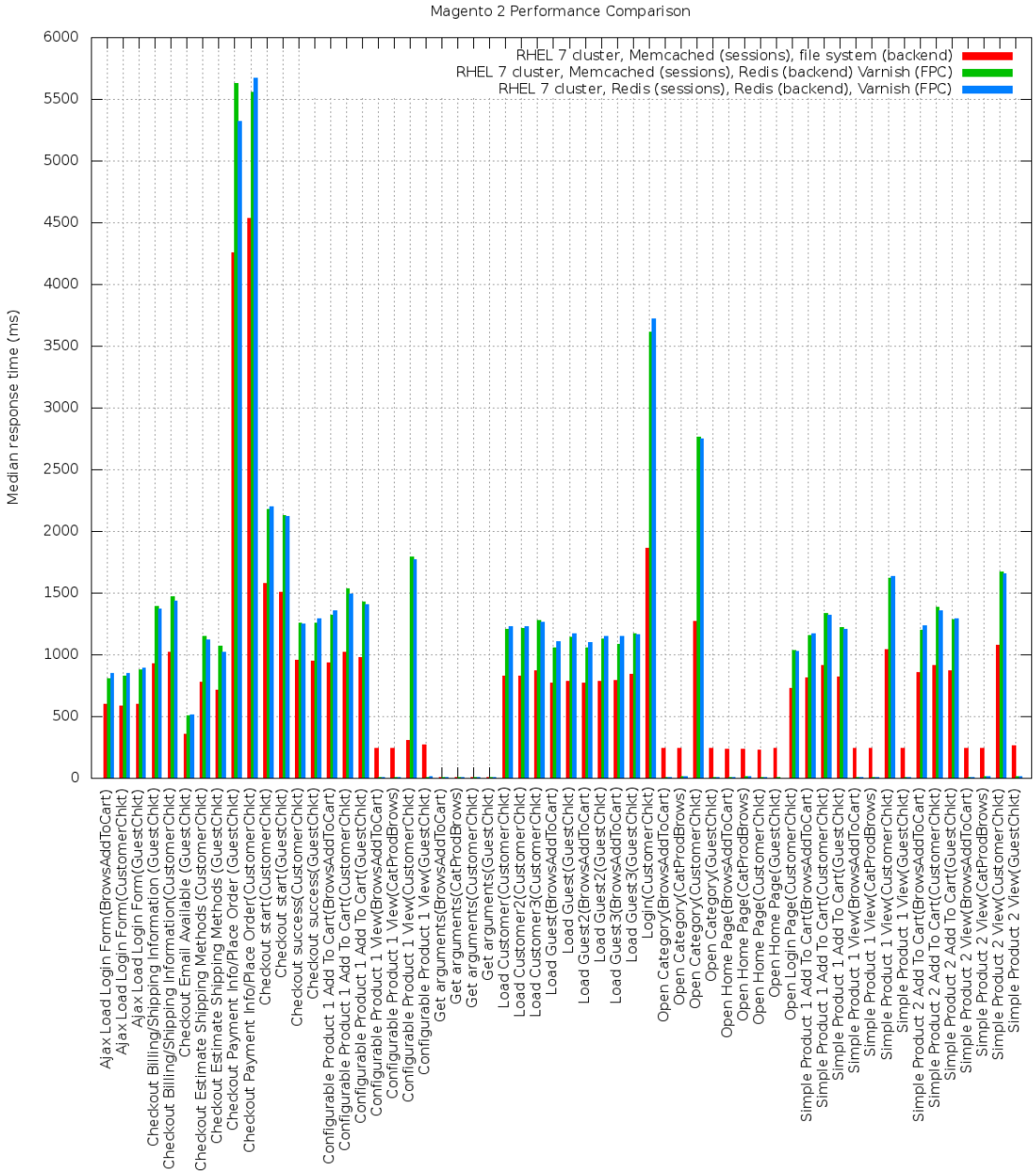
KUVA 14. Eri Linux-käyttöjärjestelmien vaikutus Magenton vasteaikoihin, kun käytössä PHP 7.0

Kuvasta 15 selviää suorituskykytestien tulokset Debian- ja RHEL-käyttöjärjestelmillä, kun molemmista niistä on Magenton sisäiset välimuistit poistettu käytöstä. Tuloksista saadaan selville, millainen on Magenton raaka suorituskyky ilman sen sisäisiä välimuistikerroksia. Suorituskykytestin aikana palvelimen kuorma nousi huomattavan korkealle ja vertailukelpoisten tulosten saamiseksi JMeter-testausprofiilia piti muokata niin, että sivupyyntöjä testiympäristöön tehtiin kerran sekunnissa. Magenton suorituskykykapasiteetti laski oletetusti, kun sen sisäiset välimuistit otetiin pois käytöstä.



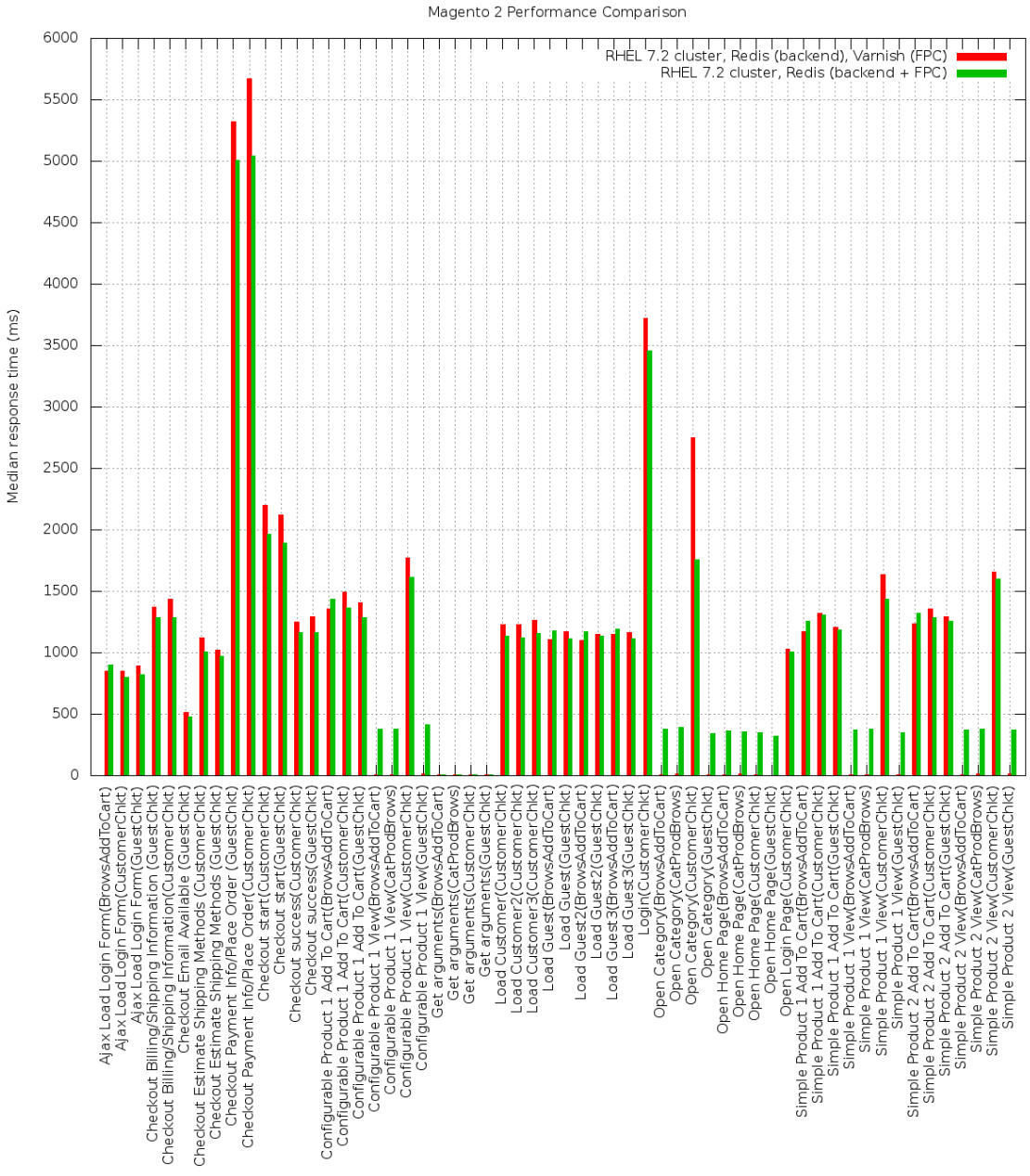
KUVA 15. Vasteajat ilman Magenton sisäisiä välimuisteja (RHEL ja Debian Linux)

Kuvasta 16 selviää klusteroidun ympäristön suorituskykytestauksen tulokset, kun käytössä on eri tallennusratkaisut käyttäjäsessioiden varastoimiseen (Memcached ja Redis). Suorituskykyvertailusta voidaan päätellä, että käyttäjäsessioiden tallennusratkaisu ei oleellisesti vaikuta klusteroidun Magento-ympäristön vasteaikoihin. Mahdollisesti kuormitustestauksen avulla näiden kahden tallennusratkaisun väliset erot olisi havaittavissa.



KUVA 16. Käyttäjäsessioiden eri tallennusratkaisujen (Redis ja Memcached) väliset nopeuserot klusteroidussa Magento-ympäristössä

Kuvasta 17 esittää Magento-klusteriin tehtyä suorituskykytestausta, jossa vertailun kohteena ovat eri sivuvälimuistien tallennusratkaisut. Hyödyllisin tieto tästä suorituskykyvertailusta on se, että kun käytössä on Varnish FPC, staattisten sivujen vasteajat laskivat alle sataan millisekuntiin (100 ms). Kun käytössä on taas Redis FPC, staattisten sivujen vasteaika lähenee neljääsataa millisekuntia (400 ms).



KUVA 17. Eri sivuvälimuistien (FPC) tallennusratkaisujen (Varnish ja Redis) nopeuserot klusteroidussa Magento-ympäristössä

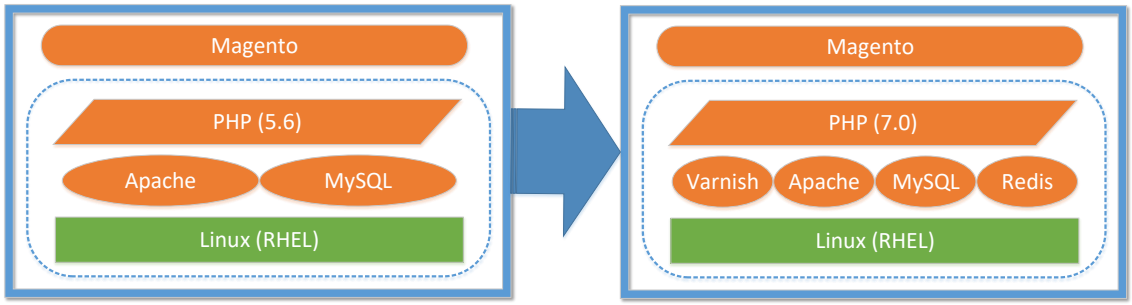
6.5 Johtopäätökset

Ajetuista suorituskykytesteistä saatiin vertailukelpoisia tuloksia, joten siltä osin suorituskyvyn testausratkaisu on onnistunut. Tulokset olivat osittain täysin päinvastaisia, mitä aluksi oli arveltu. Varsinkin Debianin suorituskyky verrattuna RHEL:iin oli todella suuri yllätys. Jopa niin suuri, että suorituskykytestin tuloksia epäiltiin aluksi virheellisiksi. Tulosten perusteella Debian suoriutui PHP 5.6:lla nopeammin kuin RHEL PHP 7.0:lla. Koska Ambientia on Red Hatin kumppani, niin tulokset päätettiin lähettää Red Hatille tutkittavaksi. Jälkikäteen selvisi, että eron aiheutti poikkeava konfiguraatio PHP:n suorittamisessa. Testiin valitut Linux-jakelut ovat valmiiksi konfiguroitu toisistaan poikkeavalla tavalla. Vaikka käyttöjärjestelmien konfiguraatioita pyrittiin yhtenäistämään, niin silti yksi asetus aiheutti mittavan suorituskykyongelman. Kyseessä oli PHP:n hitaiden kyselyjen suorittamisen kirjaaminen lokiin. Jokaiseen PHP-prosessiin kiinnittyi virheenkorjaustyökalu, joka kirjaa suorituksen aikana tehdyt järjestelmäkutsut. Kun tämä asetus poistettiin käytöstä, niin Magenton vasteajat saatiin vastaavalle tasolle muiden vertailukohteiden kanssa. Johtopäätös on siis se, että käyttöjärjestelmä ei oleellisesti vaikuta Magenton vasteaikaan, vaan erot johtuivat järjestelmän konfiguraatiosta. Jos suorituskykytestejä ei olisi tehty, niin PHP:n konfiguraatioon liittyvää ongelmaa ei välttämättä olisi löydetty. Päätös vaihtaa alustaa sen olettamuksen perusteella, että RHEL-käyttöjärjestelmä olisi muita hitaampi, olisi vaatinut huomattavia investointeja.

PHP-versiopäivityksiin liittyvissä suorituskykytesteissä selvisi, että vasteajat paranivat kaikissa testitapauksissa. Tämän tiedon perusteella päätettiin, että PHP 7.0 otetaan käyttöön kaikissa Magento 2 asennuksissa. Koska Red Hat ei kirjoitushetkellä vielä virallisesti tarjoa PHP 7.0:aa, se päätettiin asentaa ulkopuolisista lähteistä. Väliaikaista järjestelyä on tarkoitus jatkaa niin pitkälle kunnes PHP 7.0 on saatavilla Red Hatin virallisista lähteistä. Se tarkoittaa myös sitä, että kyseisen ajanjakson ajan Magenton sovelluspino ei ole täysin Red Hatin tukema. Kyseessä on tiedostettu riski, mutta ylipääsemättömissä ongelmatilanteissa on mahdollista palata takaisin käyttämään Red Hatin tukemaan PHP 5.6:sta. Tässä päädyttiin kompromissiin, jossa hyväksytään osittainen käyttöjärjestelmän tuen menetys, jotta saavutetaan yli puolet nopeampi Magenton vasteaika. Liiketoiminnan näkökulmasta tehty päätös palvelee paremmin yrityksen visiota, jossa verkkokauppojen nopeus on listattu yhdeksi päätavoitteista.

Magenton ajamisesta ilman välimuisteja opittiin se, että tuotantokäytössä välimuistien käyttö on suorituskyvyn kannalta käytännössä pakollista. Ilman välimuisteja Magenton suorituskapasiteetti palvella useita käyttäjiä on huomattavasti alentunut. Vastaavasti myös vasteajat nousivat, vaikka suorituskykytestaus suoritettiin vain muutamalla yhtäaikaisella käyttäjällä. Kehitystoimenpiteenä Magento-palveluiden julkaisuun liittyvälle tarkistuslistalle lisättiin kohta, jossa tarkistetaan, että kaikki sisäiset välimuistit ovat päällä ennen palvelun julkaisua. Suorituskykytestin aikana selvisi epäsuorasti myös se seikka, että miten Magenton ajotapa vaikutti suorituskykyyn. Magento tulee erikseen määrittää tuotantotilaan, jotta staattista sisältöä ei generoitaisi sivupyynnön aikana (Magento Dev-Docs: Magento Modes n.d.) Tämä asetus oli jäänyt ensimmäisten suorituskykytestien aikana oletustilaan, jonka vuoksi vasteajat eivät vastanneet tuotantokäyttöä. Jotta vastaavaa ei pääsisi tapahtumaan, niin myös tämä kohta lisättiin tarkistuslistalle.

Magenton klusteroituun ympäristöön ajetuista suorituskykymittauksista opittiin, että vasteaikojen kannalta ei ole juurikaan merkitystä, mitä tietovarastoa käytetään välimuistien tallentamiseen. Pääasia on se, että kaikki sisäiset ja ulkoiset välimuistit pitää olla käytössä, jotta optimaalinen vasteaika voidaan saavuttaa. Magenton sivuvälimuistin (FPC) osalta Varnishin käyttäminen laski staattisten sivujen vasteajan alle sataan millisekuntiin. Vaihtoehtoisesti sivuvälimuisti voidaan tallentaa myös Redis-tietokantaan, mutta silloin staattisten sivujen latausaika oli luokkaa 500 millisekuntia. Tähän tietoon nojautuen Magento-ympäristöihin tullaan jatkossa asentamaan Varnish-välimuistipalvelin oletuksena. Redistä tullaan jatkossa käyttämään sessioiden ja Magenton muiden välimuistin tallentamiseen. Memcached päätettiin jättää vaihtoehdoista pois, sillä se ei tarjonnut suorituskyvyn kannalta merkittävää etua. Redis on näistä kahdesta muistinvaraisista tietokannoista tuoreempaa teknologiaa ja se on ominaisuuksiltaan edistyneempi verrattuna Memcachediin (Haber 2016).



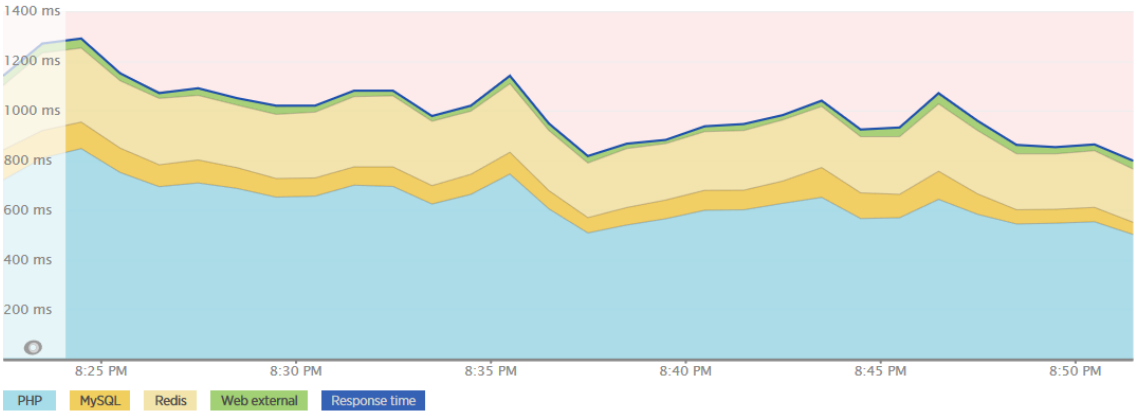
KUVA 18. Suorituskykyvertailun perusteella uudelleen määritelty Magento-sovelluspino

6.6 Arviointi ja jatkokehitys

Toteutettu suorituskyvyn testausratkaisu vastasi osittain sille asetettuihin laadullisiin tavoitteisiin. Suorituskykytestauksen käyttöönottoon kulutettua aikaa pystyttiin onnistuneesti vähentää käyttämällä olemassa olevia testaustyökaluja. Ratkaisua varten ei tarvinnut rakentaa mitään alusta asti uudelleen, joten aikaa voitiin käyttää hyödyksi suorituskykytestausprosessin määrittelyyn ja suunnitteluun. Ratkaisun käyttöönottoa helpotti suuresti Magenton oma JMeter-testausprofiili, jonka avulla suorituskykytestauksen käyttötapaukset saatiin katettua ilman mittavaa lisätyötä. Tässä kuitenkin nähtiin haasteena JMeter-testausprofiilin XML-formaatti, jonka manuaalinen muokkaaminen on hankalaa. Jos suorituskykytestauksen käyttötapauksia halutaan laajentaa, on helpompaa luoda kokonaan uusi JMeter-testausprofiili. JMeter-testausprofiilissa havaittiin myös toinen ongelma. Käytössä oleva JMeter-testausprofiili olettaa Magenton sisällön olevan määrämuotoista, joka tarkoittaa sitä, että suorituskykytestausta ei voida ajaa tuotannossa oleviin verkkokauppoihin. Magenton testidata pitää olla ensin paikallaan, jotta suorituskykytestaus voidaan ajaa. Tämä aiheuttaa tilanteen, jossa testiympäristössä ajetut suorituskykytestit eivät ole vertailukelpoisia tuotantoympäristön kanssa, sillä testattavan järjestelmän sisältö on erilainen (vaikka alusta olisi muuten identtinen).

Suorituskykytestaukseen liittyviä parametreja voidaan muuttaa eri tarkoitusten mukaisesti. JMeterillä voidaan suorittaa esimerkiksi kuormitustestausta, kun simuloitujen käyttäjien määrää kasvatetaan. Koska testausratkaisu perustuu avoimen lähdekoodin ohjelmistoihin, niitä voidaan muokata vapaammin kuin kaupallisia ohjelmistoja. Ohjelmistolisenssit eivät rajoita ratkaisun käyttöä kaupallisesti, joten testausratkaisua voidaan hyödyntää vapaasti koko organisaatiossa. Suorituskykytestit ovat helposti toistettavissa, sillä ne voidaan laukaista aina uudelleen testipalvelimen komentoriviltä. Tämä tarkoittaa myös, että testiautomaation toteuttaminen on mahdollista. Testiautomaatiota ei kuitenkaan tämän kehittämistehtävän puitteissa saatettu valmiiksi. Testiautomaation kehittäminen vaatii lisää suunnittelua, sillä testausympäristöjen monistaminen on vielä ratkaisematta. Tällä hetkellä uudet Magento-testausympäristöt pitää manuaalisesti asentaa. Tämä tarkoittaa virtuaalipalvelimen asennusta ja konfigurointia. Kun siihen lisätään vielä Magenton asennus, manuaalisia työvaiheita on vielä paljon jäljellä. Tähän on kaavailtu Vagrantin käyttöä, jonka avulla testausympäristö voidaan asentaa käyttäjän omalle työasemalle. Vagrantin avulla paikallisen virtuaalipalvelimen provisiointi ja konfigurointi voidaan automatisoida.

Suorituskykytestien tuloksista saatiin luettavia ja helposti tulkittavia muuttamalla ne graafiseen muotoon. Vaikka JMeter itsessään tarjoaa mahdollisuudet datan visualisointiin, tämä ominaisuus olisi vaatinut JMeterin ajamista graafisessa ympäristössä. Koska tätä ei testiautomaation vuoksi haluttu tehdä, päätettiin ottaa toinen komentorivityökalu ratkaisuun mukaan. Tämä kuitenkin monimutkaistaa prosessia siinä määrin, että dataa pitää kuljettaa kahden eri työkalun välillä. Myös tämä vaatisi lisäkehitystä, jotta datan visualisointiin liittyvä osuus voitaisiin automatisoida. Suorituskykytestauksen tuloksista saatiin selville uutta tietoa, joka auttoi ratkaisemaan toimintaympäristön ongelmia. Vaikka suorituskykyvertailun pohjalta voitiin tehdä johtopäätöksiä, silti tulosten informaatioarvoa pitäisi kasvattaa. Yksi suorituskykytestauksen tavoitteista on tunnistaa suorituskyvyn kannalta ongelmalliset sovelluspinon komponentit. Koska suorituskykytestauksen tuloksena saadaan vain yksi vasteajan arvo, tarkempi tulosten instrumentointi eri sovelluskomponenttien välillä ei tällä ratkaisulla onnistu. Tämä vaatisi sovellustason suorituskykymonitoroinnin käyttöönottoa, jonka avulla sovelluskomponenttien suorituskyvyn profilointi on mahdollista saavuttaa (kuva 19).

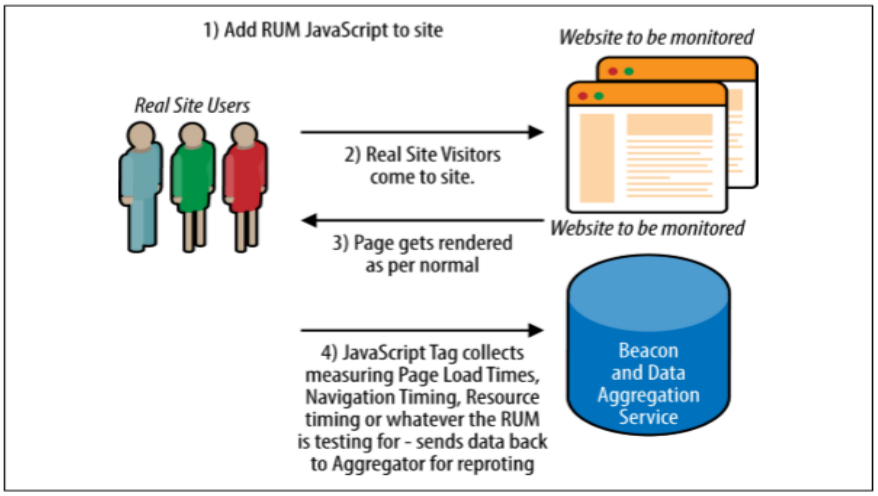


KUVA 19. Havainnointikuva New Relic APM:n toiminnasta ja vasteajan jakautumisesta eri sovelluskomponenttien mukaan

JMeter-testaustyökalun hyödyntäminen on ollut testausratkaisun käyttöönoton helppouden näkökulmasta järkevä vaihtoehto. Kun asiaa tarkastellaan toisesta näkökulmasta, JMeterin ominaisuudet eivät ole samalla tasolla modernien testaus- ja monitorointityökalujen kanssa. Varsinkin vasteajan mittauksiin on olemassa parempia työkaluja, sillä JMeter jättää osan käyttäjäkokemuksesta huomioimatta. Koska JMeter-testaustyökalu mittaa vasteajan TTFB-arvoa, mittauksesta jää pois täydellinen sivunlatausaika. Tämä johtuu siitä, että JMeter ei suorita sivunlatausta samalla tavalla kuin käyttäjän selain. JMeter ei

esimerkiksi huomioi sivun staattista tai asynkronisesti latautuvaa sisältöä. Nämä voivat kuitenkin vaikuttaa merkittävästi loppukäyttäjän kokemaan vasteaikaan. Jotta loppukäyttäjän kokema vasteaika saataisiin täysin tallennettua, seuraava kehitysaskel olisi siirtyä käyttämään synteettistä selainmonitorointia. Synteettinen selainmonitorointi tarkoittaa sitä, että selaimia komennetaan ohjelmallisesti tai selaimen toimintaa emuloidaan. Tämä laajentaisi suorituskykytestauksen myös käyttöliittymän puolella vaikkakin sen osuus on jätetty aihealueen ulkopuolelle.

Suorituskykytestauksen perusperiaatteet eivät ole muuttuneet mihinkään ajan saatossa, mutta samaa ei voi sanoa testaustyökalujen kehityksestä. Varsinkin big datan alueella on tapahtunut merkittävästi kehitystä reaaliaikaisen suorituskyvyn monitoroinnin suhteen. Yksi mainitsemisen arvioisista teknologioista on RUM (Real User Measurements). Se tarkoittaa tapaa, jolla kerätään suorituskykyyn liittyvä dataa (kuva 20). Poikkeuksellista siinä on se, että vasteaikaan liittyvä data kerätään suoraan käyttäjien selaimesta (Mastlin 2016). Tämän jälkeen selaimesta saatu data käsitellään niin, että sitä voidaan seurata miltei reaaliajassa. Kun yhdistetään kaikki edellä mainitut teknologiat, kuten synteettinen selainmonitorointi (Synthetic Browser Monitoring), käyttäjälähtöinen suorituskykymittaus (Real User Measurements) ja sovellustason suorituskykymonitorointi (Application Performance Monitoring), niin nämä yhdessä muodostavat kokonaisuuden, jonka avulla voidaan järjestelmän suorituskykyä seurata kokonaisvaltaisesti. Aktiivisen monitoroinnin avulla suorituskykyyn liittyvää metriikkaa voidaan kerätä koko järjestelmän laajuudelta ja jopa sen ulkopuolelta. Itse uskon, että aktiivinen suorituskykymonitorointi tulee tulevaisuudessa ainakin osittain syrjäyttämään perinteisen suorituskykytestauksen.



KUVA 20. RUM:n toimintaperiaate (Mastlin 2016)

7 YHTEENVETO

Kehittämistehtävän tärkeimmät tulokset asiakkaan ja täten myös toimeksiantajan näkökulmasta ovat suorituskykytestauksista johdetut kehitystoimenpiteet, joilla on ollut positiivinen vaikutus Magento-verkkokauppojen suorituskykyyn. Muuttamalla sovelluspinon konfiguraatiota ja järjestelmäarkkitehtuuria dynaamisesti generoitujen sivujen vasteajat laskivat arviolta noin puolella. Vastaavasti staattisten sivujen vasteajat laskivat viisinkertaisesti lähtötilanteeseen verrattuna. Asiakkaan näkökulmasta näillä suorituskyvyn parannuksilla on positiivinen merkitys asiakkaan verkkokauppaliiketoimintaan. Näin jälkikäteen ajateltuna tätä koettua hyötyä olisi voitu seurata jollain mitattavalla määreellä, kuten esimerkiksi konversioasteen muutoksella. Ylläpidon näkökulmasta suorituskykytestaus antoi tärkeää tietoa jo tunnettujen ja tuntemattomien ongelmien ratkaisemiseen. Suuri työmäärällinen investointi säästettiin sillä, että nykyisen alustan ongelmat ratkaistiin, kun vaihtoehtona oli koko alustan vaihtaminen toiseen. Magento-sovelluspinon järjestelmäarkkitehtuurin uudelleen määrittäminen, suorituskykytestauksesta saatujen tietojen avulla, auttaa jatkossa ympäristöjen harmonisoinnissa. Tällä saavutetaan etuja, jotka liittyvät pienentyneisiin Magento-ympäristöjen perustamis- ja ylläpitokustannuksiin.

Suorituskykytestaus on parantanut alustapalveluissa tapahtuvaa Magento-ympäristöjen muutosten hallintaa. Nämä toimenpiteet ovat vähentäneet riskejä, jotka liittyvät muun muassa palvelutasosopimuksiin ja niihin liittyviin sopimussanktioihin. Suorituskykytestaus palvelee vielä tulevaisuudessakin silloin, kun Magento-ympäristöihin otetaan uutta teknologiaa tai päivitetään olemassa olevia sovelluskomponentteja. Näin saadaan tarkempaa tietoa siitä, miten Magenton suorituskyky käyttäytyy ympäristöön tehtyjen muutosten seurauksena. Vastaavasti tämä testausratkaisu tulisi viedä sovelluskehittäjien käyttöön, jotta muutosten hallintaa voidaan suorittaa myös ohjelmakoodin osalta. Tämä kuitenkin vielä vaatii lisäkehitystä varsinkin testiautomaation ja testiympäristöjen monistamisen osalta. Samalla nousi esille tarve kehittää sovellus- ja käyttäjätason monitorointia, jolla voidaan laajentaa suorituskykytestauksesta saatuja tuloksia. Ambientian visiota Suomen nopeimmasta Magento-verkkokaupasta ei voi ainakaan vielä sanoa toteutuneeksi, johtuen puutteellisesta tiedosta. Se voidaan kuitenkin todeta, että lähemmäksi vision toteutumista on päästy ja nyt vihdoin on olemassa ratkaisu, jolla tavoite voidaan todistettavasti saavuttaa.

LÄHTEET

Aberdeen Group. 2008. The performance of web-applications: customers are won or lost in one second. Viitattu. 4.1.2017. http://www.aberdeen.com/research/5136/ra-performance-web-application/content.aspx

Akamai. 2009. Akamai reveals 2 seconds as the new threshold of acceptability for ecommerce web page response times. Viitattu 4.1.2017. https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp

Bass, L., Weber, I. & Zhu, L. 2015. DevOps: A Software Architect's Perspective.

Bixby, J. 2012. 4 awesome slides showing how page speed correlates to business metrics at Walmart.com. Viitattu 25.3.2017..http://www.webperformancetoday.com/2012/02/28/4-awesome-slides-showing-how-page-speed-correlates-to-business-metrics-at-walmart-com/

Bixby, J. 2012. Viitattu 24.3.2017. http://www.webperformancetoday.com/2012/02/13/non-geeky-guide-to-performance-measurement/

Boisvert, G. 2015. Total cost of ownership of servers: On-Premise vs. IaaS. Viitattu 21.3.2017. http://www.sherweb.com/blog/total-cost-of-ownership-of-servers-iaas-vs-on-premise/

Bolton, R. 2014. Why Every PHP Application Should Use an OpCache. Viitattu 2.4.2017. https://blog.appdynamics.com/engineering/why-every-php-application-should-use-an-opcache/

BuiltWith. 2017. Ecommerce Usage Statistics. Viitattu 12.3.2017. https://trends.builtwith.com/shop

Cutts, M. 2010. Google incorporating site speed in search rankings. Viitattu 22.1.2017. https://www.mattcutts.com/blog/site-speed/

Dean, B. 2016. Google's 200 Ranking Factors: The Complete List. Viitattu 22.1.2017: http://backlinko.com/google-ranking-factors

Debian Releases, 2016. Debian Wiki. Viitattu 29.1.207. https://wiki.debian.org/DebianReleases

Farrell-Vinay, P. 2008. Manage Software Testing. Boca Raton: Auerbach Publications.

Gartner. 2016. Magic Quadrant for Digital Commerce . Viitattu 12.3.2017. https://www.gartner.com/doc/3243418/magic-quadrant-digital-commerce

Haapahovi, S. 2015. Mitä on AB-testaus? Näin parannat konversiota AB-testauksella. Viitattu 5.1.2017. http://www.klopal.fi/2015/03/11/mita-on-ab-testaus-nain-parannat-konversiota-ab-testauksella/

Haber, I. 2016. Why Redis beats Memcached for caching. Viitattu 18.3.2017. http://www.infoworld.com/article/3063161/application-development/why-redis-beats-memcached-for-caching.html

Hoffman, B. 2013. How Website Speed Actually Impacts Search Ranking. Viitattu 25.3.2017. https://moz.com/blog/how-website-speed-actually-impacts-search-ranking

Kephart, N. 2015. 5 Must-Track Web Performance Metrics. Viitattu 2.4.2017. https://blog.thousandeyes.com/5-must-track-web-performance-metrics/

Kivinen, M. 2009. Suorituskyky on monitahoinen asia. Systeemityö-lehti 4/2009, 29-32. http://www.sytyke.org/wordpress/wp-content/uploads/2013/06/Systeemity%C3%B6-lehti_4-2009.pdf

Knuutila, J. 2016. Näin teet konversio-optimointia verkkokaupassa oikein. Viitattu 5.1.2017. https://www.paytrail.com/blog/tulos-helsingin-jaakko-knuutila-nain-teet-konversio-optimointia-verkkokaupassa-oikein

Liu, H. 2009. Software performance and scalability. Hoboken: John Wiley & Sons, Inc,

Loveland, S. Miller, G. Prewitt, R & Shannon, M. 2005. Software testing techniques. Boston: Charles River Media.

Magento DevDocs: Manage the cache. N.d. Viitattu 5.3.2017. http://devdocs.magento.com/guides/v2.1/config-guide/cli/config-cli-subcommands-cache.html

Magento DevDocs: Magento Modes. N.d. Viitattu 12.3.2017. http://devdocs.magento.com/guides/v2.1/config-guide/bootstrap/magento-modes.html

Mastlin, P. 2016. Real User Measurements. Viitattu 28.3.2017. http://www.oreilly.com/webops-perf/free/files/real-user-measurements.pdf

Meier, J., Farre, C., Bansode, P., Barber, S. & Rea, D. 2007. Performance Testing Guidance for Web Applications. Viitattu 4.1.2017. https://msdn.microsoft.com/en-us/library/bb924375.aspx

Nielsen, J. 2000. WWW-suunnittelu. Helsinki: Oy Edita Ab

Nielsen, J. 2010. Website response times. Viitattu 4.1.2017. https://www.nngroup.com/articles/website-response-times/

Nielsen, J. 2011. Top 10 mistakes in web design. Viitattu 4.1.2017. https://www.nngroup.com/articles/top-10-mistakes-web-design/

Nyári, I. 2016. Magento eCommerce: history and features of the most popular online store platform. Viitattu 12.3.2017. https://aionhill.com/magento-ecommerce-complete-overview

Ojasalo, K., Moilanen, T. & Ritalahti, J. 2014. Kehittämistyön menetelmät: uudenlaista osaamista liiketoimintaan. 3. painos. Helsinki: Sanoma Pro.

Pesonen, T. 2016. Infograafi: Testing Assemblyn kävijäkyselyn tulokset. Viitattu 3.1.2017. http://www.valagroup.com/fi/2016/10/infograafi-testing-assembly-kysely/

PHP 7.0.0 Release Notes. 2015. Viitattu 19.2.2017. http://php.net/archive/2015.php#id2015-12-03-1

Red Hat Enterprise Linux Life Cycle. N.d. Red Hat Customer Portal. Viitattu 29.1.2017. https://access.redhat.com/support/policy/updates/errata

Red Hat Security Backporting Policy. N.d. Red Hat Customer Portal. Viitattu 29.1.2017. https://access.redhat.com/security/updates/backporting

Red Hat Software Collections Product Life Cycle. N.d. Red Hat Customer Portal. Viitattu 29.1.2017. https://access.redhat.com/site/support/policy/updates/rhscl/

Riley, J. 2002. Getting Started in Programming. Interpreted vs. Compiled Languages. Viitattu 5.1.2017. http://dsbscience.com/freepubs/start_programming/node6.html

Singhal, A. 2010. Using site speed in web search ranking. Viitattu 22.1.2017. https://webmasters.googleblog.com/2010/04/using-site-speed-in-web-search-ranking.html

Soprano Oyj. 2016. Pörssitiedote. Soprano ja Ambientia yhdistävät verkkokauppayhtiönsä. Viitattu 4.1.2017. https://cns.omxgroup.com/cdsPublic/viewDisclosure.action?disclosureId=718696&messageId=900856

Tataranovich, A. 2017. Magento 1 & Magento 2 performance/speed comparison on PHP 5.6.27, PHP 7.0.13, and PHP 7.1. Viitattu 25.2.2017. https://blog.amasty.com/wp-content/uploads/2017/01/Magento-1-2-speed-comparison.pdf

Tielinen, T. 2016. Ambientia strategia 2020. Liite: Digital Commerce. Sisäinen dokumentti.

Toikko, T. & Rantanen, T. 2009. Tutkimuksellinen kehittämistoiminta. 3. painos. Tampere: Tampere University Press.

W3Techs. 2017. Server-side programming languages market position report. Viitattu 12.3.2017. https://w3techs.com/technologies/market/programming_language/10

Vainio, S. 2014. Konversioasteen mittaaminen: näin mittaat konversioasteesi. Viitattu. 25.3.2017. https://sampsavainio.fi/blogi/konversioaste/

Willis, J. & Edwards, D. 2010. DevOps Culture (Part 1). Viitattu 15.2.2017. http://itrevolution.com/devops-culture-part-1/

Voutilainen, K. 2015. Ambientia osti enemmistön Suomessa ja Virossa toimivasta Insolosta. Viitattu 30.1.2017. https://blog.ambientia.fi/2015/05/11/ambientia-osti-enemmiston-suomessa-ja-virossa-toimivasta-insolosta/