Tuomas Kommeri

Pilvilaskennan suorituskyky – erityistarkastelussa Drupal-sisällönhallintasovellus Amazonin, Rackspacen ja GoGridin pilvipalvelimilla

Tietojärjestelmätieteen
pro gradu -tutkielma
13.5.2011

Jyväskylän yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Kommeri, Tuomas
Tietojärjestelmätieteen pro gradu -tutkielma / Tuomas Kommeri
Jyväskylä: Jyväskylän yliopisto, 2011.
106 s.

Pilvilaskenta on saanut paljon huomiota osakseen viime aikoina ja sitä hyödyntävien yritysten määrä on kasvanut voimakkaasti. Käytön perusteella tapahtuva IT-resurssien laskutus, nopea palveluiden käyttöönotto ja ”loputon” skaalattavuus houkuttelevat yrityksiä siirtämään IT-toimintoja pilveen. Pilvilaskentaan liittyy paljon lupauksia ja mahdollisuuksia, mutta myös haasteita, joista monet koskevat suorituskykyä. Tässä tutkielmassa selvitetään kirjallisuuteen perustuen, mitä haasteita pilvilaskennan suorituskykyyn liittyy. Lisäksi tutkielmassa testataan Drupal-sisällönhallintasovelluksen suorituskykyä eri pilvipalveluntarjoajien alustoilla.

Tutkimustulokset osoittavat, että alustaan, virtualisointiin, jaettuihin resursseihin, saavutettavuuteen, tiedonsiirtoon, skaalattavuuteen ja palvelinkeskusten sijaintiin liittyy suorituskykyä koskevia tekijöitä, jotka on hyvä ottaa huomioon pilvipalveluiden käyttöönottoa suunniteltaessa. Tutkielmassa suoritetut empiiriset testit osoittavat, että eri palveluntarjoajien alustat ovat suorituskyvyltään hyvin erilaisia. Fyysisen palvelininfrastruktuurin ominaisuudet, resurssien allokointi pilvipalvelimille ja palvelinkeskusten maantieteellinen sijainti ovat ne tekijät, jotka suorituskyvyn ensisijaisesti määräävät.

AVAINSANAT: pilvi, pilvipalvelu, pilvilaskenta, pilvipalvelin, pilvilaskenta-alusta, SaaS, PaaS, IaaS, suorituskyky, suorituskykytestaus, Drupal, Amazon, Rackspace, GoGrid

SISÄLLYSLUETTELO

1 JOHDANTO

IT-toimintojen siirtäminen pilvilaskentaa (cloud computing) hyödyntävien yritysten omistamiin palvelinkeskuksiin on voimakkaassa kasvussa. Tavoitteena on säästää oman IT-infrastruktuurin rakentamiseen ja ylläpitoon kuluvaa aikaa ja kustannuksia. IDC:n vuoden 2010 ennusteiden mukaan suurimmat muutokset IT-alalla liittyvät pilvipalveluihin. Tämä johtaa tuotteiden, markkinoiden ja liiketoimintamallien uudistumiseen. (IDC Predictions 2010). Pilvipalveluilla tarkoitetaan yleisesti verkon kautta tarjottavia tietojenkäsittelyresursseja, jotka toimitetaan palvelinkeskuksista asiakkaille. Tällaiset resurssit voidaan jakaa karkeasti kolmeen ryhmään, jotka ovat valmiit sovellukset, sovellusalustat ja infrastruktuuritason palvelut. Tällaisille tietojenkäsittelyresursseille on tyypillistä, että niistä maksetaan käytön mukaan. Periaatteessa kenellä tahansa on mahdollisuus rekisteröityä palveluihin palveluntarjoajien verkkosivustoilla (vaatii palveluntarjoajan hyväksymän luottokortin) ja aloittaa niiden käyttö välittömästi. Esimerkiksi virtuaalisen palvelininstanssin käynnistäminen web-pohjaisen käyttöliittymän kautta on nopeaa ja se kestää palveluntarjoajasta riippuen kymmenistä sekunneista muutamaan minuuttiin.

Virtualisointi on nimenomaan tärkeä, pilvelle ominainen tekijä, joka erottaa sen aiemmista teknologioista kuten hilalaskennasta (grid computing). Hilalaskennassa hyödynnetään myös virtualisointia, mutta se ei tukeudu siihen pilvilaskennan tavoin. (Foster ym. 2008; Weinhardt ym. 2009). Teknologiat sisältävät myös muita eroavaisuuksia mutta myös paljon yhtäläisyyksiä. Foster ym. (2008) ovat verranneet pilvilaskentaa ja hilalaskentaa toisiinsa ja todenneet tutkimuksessaan, että molempien teknologioiden päämääränä on vähentää tietojenkäsittelyn kustannuksia, lisätä luotettavuutta ja joustavuutta siirtämällä tietojenkäsittelyresurssit pois asiakkaiden omista tietokonesaleista. Toisaalta taas teknologiat eroavat toisistaan mm. turvallisuus-, ohjelmointi-, liiketoiminta-, laskenta-, tietomalleiltaan. Esimerkiksi liiketoimintamallin

tapauksessa pilvelle on ominaista juuri välitön pääsy tietojenkäsittelyresursseihin Internetin kautta ja maksaminen käytön mukaan (tunti- ja datamäärien perusteella). Hilalaskentaa verrataan monesti pilvilaskentaan monien yhtäläisyyksien takia, mutta pilvilaskenta sisältää aineksia myös muualta. Youseffin (2008) mukaan pilvilaskenta on eräänlainen yhdistelmä palvelukeskeistä arkkitehtuuria, hajautettua- ja hilalaskentaa sekä virtualisointia. Voidaankin sanoa, ettei pilvilaskenta ole teknologisesti uusi ja mullistava keksintö vaan se on enemmänkin uusi konsepti, jossa on hyödynnetty olemassa olevia keksintöjä.

Vaikka pilvilaskentaa hyödyntävien yritysten määrä on kasvanut voimakkaasti, siihen liittyy edelleen useita epäselvyyksiä ja epävarmuutta aiheuttavia tekijöitä. Isoja kysymyksiä on suorituskyky, turvallisuus, luotettavuus, saavutettavuus ja kustannukset. Mm. Armbrust ym. (2009) listaavat tunnetussa pilvilaskentaa koskevassa teknisessä raportissaan kymmenen suurinta mahdollisuutta ja estettä pilvilaskennan kasvulle. Näistä puolet liittyy suorituskykyyn. Toisaalta palveluntarjoajat lupaavat suorituskyvyn ja kustannustehokkuuden osalta paljon; loputonta kapasiteettia, skaalattavuutta ja maksua vain käytetyistä resursseista. Toisaalta taas ongelmia ei luonnollisesti mainosteta, ja näin ollen koko totuutta saattaa olla vaikeaa hahmottaa.

Pilvilaskennassa käytettäviin teknologioihin liittyy siis suorituskyvyn osalta haasteita, jotka koskevat kaikkia pilvipalveluntarjoajia. Näiden haasteiden lisäksi asiakkaan haasteena on löytää palveluiden ja hinnoittelumallien viidakosta tarpeilleen sopiva ratkaisu. Palveluntarjoajien erilaisuus johtaa siihen, että palveluiden kesken suorituskykyä pitää arvioida ja mitata. (Li ym. 2010). Tällä tavalla voidaan löytää optimaalinen ratkaisu tietyn tyyppiselle, esimerkiksi korkeaa laskentatehoa vaativalle sovellukselle. Myös Durkeen (2010) mukaan eri alustojen suorituskykyyn tulee kiinnittää huomiota. Palveluntarjoajien ilmoittamien ominaisuuksien perusteella on vaikeaa arvioida

todellista suorituskykyä, ja näin ollen sovelluksen ajaminen pilvialustalla on paras keino kyseisen sovelluksen suorituskyvyn määrittämiseksi.

Tässä tutkielmassa ensimmäisenä tutkimusongelmana selvitetään, mitä haasteita pilven suorituskykyyn liittyy. Tutkielmassa kartoitetaan eri lähteissä esitetyt haasteet. Ensimmäisessä tutkimusongelmassa tutkimusmenetelmänä on systemaattinen kirjallisuuskatsaus. Toisena tutkimusongelmana vertaillaan kokeellisen tutkimuksen keinoin LAMP (Linux, Apache, MySQL, PHP) -pohjaisen Drupal-sisällönhallintasovelluksen (Drupal 2010a) suorituskykyä tunnettujen pilvipalveluntarjoajien (Amazon, Rackspace ja GoGrid) alustoilla. Sovellus asennetaan testattaville alustoille ja sitä kuormitetaan simuloiduilla käyttäjillä Suomesta käsin. Suorituskyvyn arvioinnissa käytetään neljännessä luvussa esiteltäviä suorituskyvyn mittareita. Tämän tutkielman tutkimuskysymykset voidaan esittää seuraavasti 1) Mitä haasteita pilven suorituskykyyn liittyy? ja 2) Millainen on Drupal-sisällönhallintasovelluksen suorituskyky yhden palvelimen kokoonpanolla eri pilvipalveluntarjoajien alustoilla?

Tutkielma jakaantuu rakenteellisesti siten, että johdannon jälkeisessä luvussa määritellään pilvilaskenta ja siihen liittyvät keskeiset käsitteet. Lisäksi luvussa perehdytään lyhyesti tärkeimpään pilven taustalla vaikuttavaan teknologiaan, virtualisointiin. Kolmannessa luvussa esitellään niitä pilvilaskennan suorituskykyyn liittyviä haasteita, joita kirjallisuudessa on esitetty. Neljäs luku koskee tietojärjestelmien suorituskyvyn mittareita. Luvussa perehdytään kirjallisuudessa esitettyihin mittareihin, joita hyödynnetään jatkossa suorituskykytestauksen yhteydessä. Viidennessä luvussa esitellään suorituskykytestausta yleisesti ja kuudennessa luvussa perehdytään erityisesti web-pohjaisten järjestelmien suorituskykytestaukseen. Lukuihin 7-9 sisältyy tämän tutkielman kokeellinen osuus. Se koostuu testauksen valmisteluun liittyvistä asioista sekä tulosten esittämisestä ja analysoinnista. Lopuksi kymmenennessä luvussa muodostetaan yhteenveto tutkielmasta.

2 PILVIPALVELUT JA PILVEN TOTEUTUSMALLIT

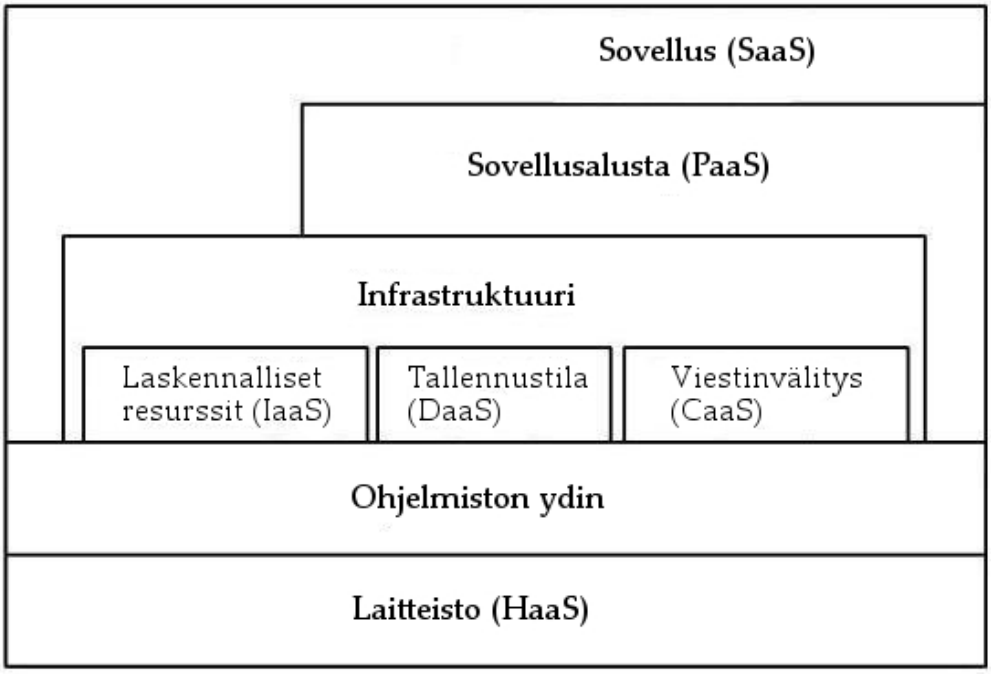
Tässä luvussa määritellään pilvilaskenta ja esitellään pilvipalveluiden kolme pääryhmää. Lisäksi luvussa esitellään pilven toteutusmallit (cloud deployment models), joista mielenkiinto tämän tutkielman osalta kohdistuu julkisiin pilviin (public clouds). Luvun lopussa esitellään pilven taustalla olevaa virtualisointiin liittyvää teknologiaa.

2.1 Pilvilaskennan määritelmä

Useat asiantuntijat ovat kehittäneet määritelmiä pilvelle ja pilvilaskennalle. Yksiselitteistä ja lyhyttä määritelmää on mahdotonta keksiä, sillä pilvilaskentaan liittyvien käsitteiden ja teknologioiden määrä on kirjava. Buyya ym. (2009, 601) ovat määritelleet pilven artikkelissaan seuraavasti: "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." Määritelmässä pidetään siis oleellisena toisiinsa kytkettyjä virtualisoituja tietokoneita, jotka tarjotaan dynaamisesti asiakkaille. Sys-con.com-sivuston (Sys-Con Media 2009) pilvilaskentaa koskevassa artikkelissa on esitelty 21 asiantuntijan määritelmät pilvilaskennalle. Useissa artikkelissa esitellyistä määritelmistä esiintyy pilven skaalattavuus (tai elastisuus) sekä käytön mukaan tapahtuva laskutus. Ideaalitilanteessahan pilvi toimii siten, että resursseja lisätään ja vähennetään vastaamaan aina tietyn hetken kuormitustasoa. Muita määritelmissä esiintyviä asioita ovat mm. verkkokeskeisyys, globaalius ja tietojenkäsittelyresurssien ulkoistus. Pilviä on luokiteltu myös laskentakapasiteetin julkiseksi palveluksi. Tällaisen luokittelun perustana on se, että laskentaresurssit (ja muut tietojenkäsittelyresurssit) toimitetaan tietystä kohteesta veteen tai sähköön verrattavalla tavalla niiden kuluttajille.

2.2 Pilvipalvelut

Yleisesti pilvipalveluilla tarkoitetaan palveluita, jotka toimitetaan verkon, tavallisesti Internetin, välityksellä asiakkaalle ja joista laskutetaan käytön perusteella. Suurempien pilvipalveluita tarjoavien yritysten omistamat palvelinkeskukset sisältävät tuhansittain virtualisointia hyödyntäviä palvelimia, joista vuokrataan resursseja eri käyttötarkoituksiin. Pilvipalvelut jaetaan yleisesti kolmeen pääryhmään: valmiit sovellukset (Software-as-a-Service, SaaS), sovellusalustat (Platform-as-a-Service, PaaS) ja infrastruktuuri (Infrastructure-as-a-Service, IaaS). (mm. Bennett ym. 2009; Fouquet ym. 2009; Mather ym. 2009, 17-22). Jako voidaan tehdä myös tarkemmin, jolloin esimerkiksi palveluna tarjottava tietovarasto tai tietokanta voitaisiin jakaa omaan ryhmään. (Linthicum 2009, 11-13; Youseff ym. 2008). Pilvipalveluiden jaottelua Youseffin ym. (2008) mukaan esitellään kuvassa 1.



Kuva 1 Pilvipalvelut (Youseff ym. 2008)

Kuvassa 1 kolme ylintä kerrosta muodostaa varsinaiset pilvipalvelut ja kaksi alinta kerrosta koskee fyysisiä palvelimia ja muuta rautatason infrastruktuuria

sekä niiden toimintaan tarvittavia ohjelmistoja. Seuraavaksi esitellään kuvan 1 kolme ylintä kerrosta tarkemmalla tasolla. Alimpiin kerroksiin liittyviä pilven taustateknologioita esitellään kohdassa 2.4.

2.2.1 Sovellukset (SaaS)

Sovelluskerros on rajapinta pilven ja erilaisten päätelaitteiden välillä. Se on siis loppukäyttäjälle näkyvin kerros ja sen kautta sovelluksiin käytettävä laskenta voidaan siirtää paikallisilta tietokoneilta pilveen. (Youseff ym. 2008). Tässä palvelumallissa asiakas ostamisen sijaan vuokraa sovelluksen. Joissakin tapauksissa palvelu voi olla asiakkaalle myös maksuton (esim. Google Gmail). Tyypillisesti palvelu kattaa laitteiston, ohjelmiston ja ylläpidon. Näin ollen palvelu on valmis käytettäväksi ja asiakas pääsee siihen käsiksi verkon kautta minkä tahansa auktorisoidun laitteen avulla. (Mather ym. 2009, 18-19).

2.2.2 Sovellusalustat (PaaS)

Toinen palvelumalli on sovellusalustat, joista tunnetuimpia ovat Google App Engine (Google 2010), Force.com (Salesforce.com 2010) ja Windows Azure (Microsoft 2010). Ne tarjoavat sovelluskehittäjille ohjelmointiympäristön, jonka avulla voidaan rakentaa verkkopalvelu toimittajan alustalle. Tyypillisesti toimittaja kehittää työkalut ja standardit sovelluskehitystä varten. (Mather ym. 2009, 19). Näin ollen kehitystyö tapahtuu rajatummassa ja kontrolloidummassa ympäristössä kuin ”perinteisessä” sovelluskehityksessä. Etuna ratkaisussa on, että kehittäminen voidaan aloittaa nopeasti valmiiksi konfiguroidussa ympäristössä ja sovellukset saadaan hetkessä verkkoon käyttäjien ulottuville.

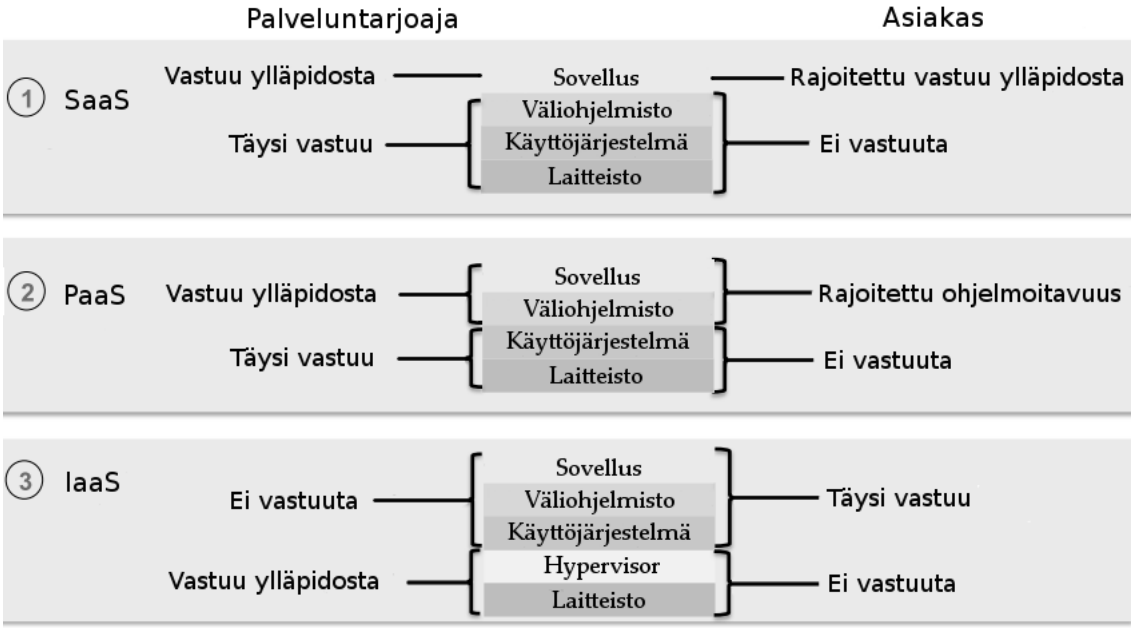
Youseffin ym. (2008) mukaan sovellusalustojen etuja ovat automaattinen skaalautuminen ja kuorman tasaus (load balancing) sekä integroituminen muihin palveluihin (mm. sähköpostipalvelut). Lisäksi kehittäjillä on

mahdollisuus integroida muita palveluita kehittämiinsä sovelluksiin tarpeidensa mukaan.

2.2.3 Infrastruktuuri (IaaS)

Kolmas palvelumalli on palveluna tarjottava infrastruktuuri. Palveluun sisältyy laitteisto ja siihen liittyvä ohjelmisto (mm. käyttöjärjestelmän virtualisointiin liittyvä teknologia). Tässä vaihtoehdossa palveluntarjoajien päätehtävänä on pitää palvelinkeskus toimintakunnossa ja jättää tarkempi konfigurointi asiakkaan vastuulle. (Bennett ym. 2009). Tämän tason palveluita tarjoavista yrityksistä ylivoimaisesti tunnetuin on Amazon. (Amazon.com 2010a). Palveluiden käyttäjällä on mahdollisuus valita esimerkiksi käyttöjärjestelmä, ohjelmointikieli ja tietokanta käyttämäänsä palvelininstanssiin. (Amazon.com 2010b; Reese 2009, 31-33). Lisäksi esimerkiksi kapasiteetin hallinta palvelininstansseja käynnistämällä ja sammuttamalla on asiakkaan vastuulla. Toimintoa varten on tosin kehitetty myös automatiikkaa (mm. Amazon.com 2010c). Kuvan 1 mukaisesti (Youssef ym. 2008) palveluna tarjottava infrastruktuuri voidaan jakaa tarkemmin laskennallisiin resursseihin, tietovarastoon ja viestinvälitykseen. Laskennallisilla resursseilla tarkoitetaan käytännössä virtualisoituja palvelimia (esim. Amazon EC2), tietovarastolla pilvestä saatavaa levytilaa (esim. Amazon S3) ja viestinvälityksellä tässä yhteydessä palveluna tarjottavia tietoliikenneyhteyksiä.

Kuvassa 2 Badger ja Grance (2010) esittelevät palveluntarjoajan ja asiakkaan vastuita edellä esitellyissä palvelumalleissa. Vastuualueet on kahdessa ensimmäisessä mallissa jaettu sovellukseen, väliohjelmistoon, käyttöjärjestelmään ja laitteistoon. IaaS-mallissa on esitelty lisäksi palvelinten virtualisoinnista vastaava hypervisor-kerros.



Kuva 2 Palveluntarjoajan ja asiakkaan vastuut eri palvelumalleissa (Badger & Grance 2010)

SaaS-palvelumallissa palveluntarjoajan vastuulla on väliohjelmisto, käyttöjärjestelmä ja laitteisto. Lisäksi palveluntarjoaja toimii sovelluksen pääylläpitäjänä. PaaS-palvelumallissa asiakkaalla on enemmän vastuita, mutta palveluntarjoaja kantaa vielä täyden vastuun käyttöjärjestelmästä ja laitteistosta. IaaS-palvelumallissa asiakkaan vastuu kasvaa edelleen ja myös vastuu käyttöjärjestelmästä siirtyy asiakkaalle. IaaS-mallissa palveluntarjoaja säilyttää edelleen kontrollin fyysiseen laitteistoon ja sen toiminnan edellyttävään ohjelmistoon.

2.3 Pilven toteutusmallit

Pilven toteutusmallit voidaan jakaa kolmeen ryhmään (mm. Bennett ym. 2009; Mather ym. 2009, 22-25; Sotomayor ym. 2009). Ensimmäinen toteutusmalli on julkinen pilvi. Käytännössä pilvilaskenta tunnettuine palveluntarjoajineen yleensä rinnastetaan tähän vaihtoehtoon. Lisäksi keskeiset pilviohjelmoinnista saatavat hyödyt on saavutettavissa tätä vaihtoehtoa hyödyntämällä (mm. pienet aloituskustannukset ja käytön mukaan tapahtuva maksaminen). Yritykset rakentavat käyttöönsä myös ns. yksityisiä pilviä (private clouds), joiden

käytöllä voidaan saavuttaa tiettyjä etuja (kohta 2.3.2). Toisaalta tällöin oman infrastruktuurin rakentamisen ja ylläpidon kustannuksilta ei säästytä, ja tällä tavalla eräs pilviohjelmoinnin keskeinen ominaisuus menettää merkityksensä. Kolmas vaihtoehto on hybridimalli, jossa osa toiminnoista pidetään talon sisällä ja osa siirretään julkiseen pilveen.

2.3.1 Julkiset pilvet

Julkiset pilvet kuvaavat pilvilaskennan pääsuuntausta, jossa resurssit toimitetaan dynaamisesti itsepalveluna verkon kautta. Organisaation ulkopuolinen toimija operoi ja ylläpitää julkisia pilviä yhdessä tai useammassa palvelinkeskuksessa. Palvelu tarjotaan yleisen infrastruktuurin yli useille asiakkaille. (Mather ym. 2009, 23). Tällä tavalla asiakkaat saavuttavat tiettyjä hyötyjä, joista tärkeimpiä ovat (Mather ym. 2009, 26-27):

1. Pienet aloituskustannukset, sillä laitteistoon ja ohjelmistoihin ei tarvitse investoida. Lisäksi, kun palvelusta maksetaan käytön mukaan ja etukäteismaksut ovat minimaaliset, vähentää se yritysten kynnystä astua markkinoille.

2. Skaalaedut. Normaalisti useimmissa sovellusprojekteissa pyritään laskemaan vaadittava tilantarve, prosessointiteho ja muistitarve kehityksen, testauksen ja tuotannon aikana. Tarkkojen arvioiden tekeminen on usein vaikeaa. Kun laskentateho saadaan pilvestä kysynnän mukaan, tätä ongelmaa ei ole.

3. Käytössä on modulaarinen arkkitehtuuri, joka pystyy kasvamaan nopeasti ja muuttumaan tarvittaessa.

4. Vakaus. Palveluntarjoajat ovat investoineet huomattavia summia luodakseen kestävän arkkitehtuurin, joka tarjoaa vakaan ympäristön.

2.3.2 Yksityiset pilvet

Yksityinen pilvi on omistettu organisaation sisäiseen käyttöön. Se toimii joko organisaation omilla laitteistoilla tai vaihtoehtoisesti se voi olla ulkoistettu. (Grossman 2009). Yksityinen pilvi voi olla myös organisaatioiden välinen, jolloin sen omistus ja kontrolli jakaantuu organisaatioiden kesken. Tällöin voidaan puhua myös ns. yhteisöpilvestä (community cloud). (Briscoe ja Marinos 2009; Linthicum 2009, 10). Matherin ym. (2009, 23-24) mukaan yksityiset pilvet emuloivat pilvilaskentaa yksityisissä verkoissa. Ne ovat tyypillisesti tuotteita, joiden avulla saavutetaan joitakin pilvilaskennan hyötyjä ilman että tarvitsisi kantaa huolta mm. julkisiin pilviin liittyvistä turvallisuus- ja luotettavuuskysymyksistä. Toisaalta organisaatioiden täytyy ostaa, rakentaa ja ylläpitää ne, joten investointi- ja ylläpitokustannuksilta ei säästytä. Reesen (2009, 19) mukaan yksityisten pilvien tärkein etu on kontrolli: yritys säilyttää täyden kontrollin IT-infrastruktuuriinsa, mutta saa myös kaikki virtualisoinnin hyödyt.

2.3.3 Hybridi

Käytännössä pilvipalveluita hyödyntävät yritykset käyttävät monesti palveluiden sekoitusta, jossa osa on talon sisäisiä ja osa ulkoistettuja. Esimerkiksi palkanlaskenta voitaisiin tarjota pilvestä ja integroida muihin sovelluksiin. (Murray 2009). Hybridi koostuu siis useista sisäisistä ja/tai ulkoisista palveluntarjoajista ja sen avulla organisaatiot voivat mm. säilyttää ydintoiminnan sovellukset ja arkaluonteisen datan talon sisällä. Samalla muita toimintoja voitaisiin siirtää julkiseen pilveen. (Mather ym. 2009, 25).

2.4 Taustateknologiat

Pilven taustalla vaikuttavista teknologioista keskeisin on virtualisointi. Douglasin ja Gehrmannin (2010) mukaan virtualisointi on järjestelmän

abstrahointia, jossa virtualisointilogiikan kerros käsittelee ja tarjoaa ”virtualisoidut” resurssit sen yläpuolella toimivaan asiakaskerrokseen (client layer). Asiakas pääsee käsiksi resursseihin standardien rajapintojen kautta, mutta nämä rajapinnat eivät kommunikoi suoraan resurssien kanssa. Sen sijaan virtualisointikerros käsittelee oikeat resurssit ja mahdollisesti moninkertaistaa ne useamman asiakkaan kesken. Virtualisoinnin avulla voidaan erottaa laitteisto käyttöjärjestelmästä ja tällä tavalla pilvipalveluyritysten asiakkaat saavat valita käyttämilleen pilvipalvelimille haluamansa käyttöjärjestelmän. Tällainen valinnanvapaus koskee siis palveluna tarjottavaa infrastruktuuria (IaaS) ja tarkemmin tässä tapauksessa virtualisoituja palvelininstansseja. Valmiiksi konfiguroiduissa sovellusalustoissa (PaaS) valintamahdollisuudet on varsin rajalliset.

2.4.1 Virtualisoinnin tyypit

Smith ja Ravi (2005) jaottelevat virtualisoinnin kahteen perustyyppiin, jotka ovat prosessivirtualisointi (process virtualization) ja järjestelmävirtualisointi (system virtualization). Prosessitason virtualisointi on perustavaa laatua oleva käsite lähes jokaisessa modernissa tietokonejärjestelmässä. Prosessivirtualisoinnissa käyttöjärjestelmä virtualisoi muistiosoiteavaruuden (memory address space), prosessorin, prosessorin rekisterit ja muut järjestelmäresurssit jokaiselle käynnissä olevalle prosessille. Jokainen prosessi on vuorovaikutuksessa käyttöjärjestelmän kanssa virtuaalisen rajapinnan kautta tietämättä muista prosesseista. Käyttöjärjestelmä huolehtii virtualisoinnista ja ylläpitää yhteyttä kullekin prosessille.

Järjestelmävirtualisoinnissa on taas kyse siitä, että koko järjestelmä on virtualisoitu. Tällä tavalla useat virtuaaliset järjestelmät pystyvät toimimaan eristettyinä mutta silti yhteistyössä keskenään. Virtuaalikoneen monitori (virtual machine monitor, VMM) virtualisoi kaikki oikean koneen resurssit, sisältäen prosessorin, laitteet, muistin ja prosessit. Tällä tavalla muodostuu

virtuaalinen ympäristö, jota kutsutaan virtuaalikoneeksi (virtual machine, VM). Virtuaalikoneella toimivalla ohjelmistolla on ikään kuin harhakuva, että se toimii oikealla koneella. Kaikkiin oikealla koneella oleviin resursseihin päästään käsiksi tietyn rajapinnan kautta. Virtuaalikoneen monitori käsittelee oikeat resurssit ja tarjoaa ne virtuaalikoneille. (Douglas & Gehrmann 2005; Smith & Ravi 2005). Nimenomaan järjestelmävirtualisointi mahdollistaa pilvipalvelut, jolloin tietystä määrästä fyysisiä palvelimia saadaankin useita virtuaalisia palvelininstansseja, jotka näkyvät niiden vuokraajille kuten fyysiset palvelimet. Tällä tavalla fyysiset resurssit jaetaan useiden asiakkaiden kesken eikä asiakkaan hallittavana olekaan enää perinteiseen tapaan fyysinen palvelin.

2.4.2 Virtualisointiohjelmistot

Virtualisointiohjelmistoista (tai virtuaalikoneen monitoreista) löytyy useita kaupallisia ja avoimen lähdekoodin tuotteita. Virtualisointiohjelmistot voidaan jakaa kahteen eri tyyppiin. Ensimmäisen tyypit ohjelmistot asennetaan olemassa olevan käyttöjärjestelmän päälle ja toisen tyypin ohjelmistot käyttöjärjestelmättömälle palvelimelle. Esimerkiksi Xen ja VMWare ESX toimivat suoraan laitteiston päällä. Tämä on perinteinen tapa ja tällöin ohjelmistojen täytyy tarjota myös laitteistoajurit ja muut tarvittavat komponentit ja palvelut. Toisen tyypin mahdollisena etuna on, että olemassa olevat laitteistoajurit ja palvelut ovat virtualisointiohjelmiston käytettävissä. (Douglas & Gehrmann 2010).

2.5 Yhteenveto

Pilvilaskennan määritelmä ei ole yksiselitteinen. Eri tahojen kirjoittamissa määritelmissä toistuvat kuitenkin tietyt asiat, joita ovat mm. virtualisointi, skaalattavuus ja käyttöperusteinen laskutus. Myös luokittelu laskentakapasiteetin julkiseksi palveluksi on saanut suosiota osakseen. Pilvipalvelut jaetaan yleisesti kolmeen pääryhmään, jotka ovat sovellukset,

sovellusalustat ja infrastruktuuri. Ensimmäisessä ryhmässä asiakkaalle tarjotaan valmis sovellus, toisessa ohjelmointiympäristö sovellusten kehittämistä varten ja viimeisessä laitteisto sekä vastuu käyttöjärjestelmästä. Pilven toteutusmallit jaetaan niin ikään kolmeen ryhmään, jotka ovat julkiset pilvet, yksityiset pilvet ja hybridi. Julkisilla pilvillä tarkoitetaan pilvilaskennan pääsuuntausta, jossa julkisesti tarjolla oleviin tietojenkäsittelyresursseihin päästään hetkessä käsiksi palveluntarjoajien sivustoilla olevien sovellusten avulla. Yksityiset pilvet on tarkoitettu organisaatioiden sisäiseen käyttöön ja ne toimivat joko omilla laitteistoilla tai ovat ulkoistettuja. Hybridimallissa taas osa toiminnoista tarjotaan pilvestä ja osa muualta (esim. ydintoiminnan sovellukset). Tärkein pilvilaskennan taustalla vaikuttavista teknologioista on virtualisointi. Järjestelmätason virtualisoinnin avulla fyysisestä palvelimesta saadaan useita virtuaalisia palvelininstansseja, jotka toimivat toisistaan erillään. Kullekin palvelininstanssille voidaan asentaa esimerkiksi haluttu käyttöjärjestelmä.

3 PILVEN SUORITUSKYKY

Tässä luvussa esitellään pilven suorituskykyyn liittyviä tekijöitä. Kohdassa 3.1 esitellään alustan suorituskykyä ja verkon latenssia sekä muita kirjallisuudessa mainittuja pilven suorituskykyyn liittyviä asioita. Kohdassa 3.2 perehdytään virtualisoitujen tietokoneiden suorituskykyyn ja kohdassa 3.3 suorituskyvyn vaihteluihin.

3.1 Pilven suorituskyky yleisesti

Linthicumin (2009, 226-227) mukaan pilven suorituskykyyn liittyy kaksi tärkeää tekijää: alustan suorituskyky ja verkon latenssi. Alustan osalta asiakkaalla on pilvipalvelussa oma virtuaalinen tila tai virtuaalikone mutta prosessorit ja tallennustila jaetaan muiden asiakkaiden kanssa. Jaetussa arkkitehtuurissa piilee suorituskykyyn liittyviä haasteita ja tämän vuoksi palveluntarjoajalta tulisi vaatia palvelutasosopimus tietyn suorituskyvyn takaamiseksi. Durkeen (2010) mukaan palvelutasosopimuksetkaan ei aina takaa suorituskykyä tai ole asiakkaalle edullisia, sillä sopimusrikkomustilanteissa luvatut korvaukset ovat vähäisiä. Lisäksi palvelutasosopimukset perustuvat tyypillisesti vain palvelujen saavutettavuuteen. (Schad ym. 2010).

Palveluntarjoajien ilmoittamat ominaisuudet antavat viitteitä, mutta eivät kuitenkaan tarkkaa käsitystä palvelun suorituskyvystä. Lisäksi monet palveluntarjoajat ovat epämääräisiä käyttämänsä fyysisen laitteiston ja ohjelmiston ominaisuuksista, jolla he jakavat virtuaalipalvelimen asiakkaalle. Palveluntarjoajat mm. määrittävät muistin allokoinnin ja jättävät prosessorin tiedot kertomatta, mainitsevat jaettujen resurssien enimmäismäärät yksityisten sijasta ja määrittävät mm. prosessoriteholle vaihtelualueen. (Durkee 2010). Edellä esitettyihin väitteisiin löydetään tukea palveluntarjoajien Internet-sivustoilta. Esimerkiksi Amazon (Amazon.com 2010d) ilmoittaa mm. käyttämänsä laskentayksikön prosessoritehon tietyllä vaihteluvälillä. Lisäksi

ilmoitetaan, että osa resursseista on allokoitu pelkästään tietylle palvelininstanssille ja osa taas jaetaan palvelininstanssien kesken. Tarkkaa käsitystä suorituskyvystä on siis mahdotonta muodostaa palveluntarjoajien ilmoittamien ominaisuuksien perusteella. Durkeen (2010) mukaan olisi poikkeuksellista, jos pilvet voisivat tarjota sellaisen suorituskyvyn tason, johon yritykset ovat tottuneet. Hänen mukaansa suurimmassa osassa tapauksista ainoa keino määrittää tietyn sovelluksen suorituskyky, on ajaa se pilvialustalla. Myös Amazonin web-sivustoilla (Amazon.com 2010d) mainitaan, että paras tapa löytää sovellukselle oikeanlainen alusta, on testata sitä eri palvelininstansseilla.

Verkon latenssista puhuttaessa pilviresursseista tulee hidas linkki pitkässä arkkitehtuuriketjussa. Kun perinteisissä järjestelmissä palvelin vastaa oletetulla tavalla, niin pilvijärjestelmissä verkon viiveet aiheuttavat hitaamman vastauksen. Tyypillisesti verkon latenssi ilmenee suurien tietomäärien siirrossa pilvipalveluntarjoajalle ja takaisin. (Linthicum 2009, 226-227). Myös Grossman (2009) ja Leavitt (2009) ottavat pilvilaskennan latenssiongelman artikkeleissaan esille. Armbrust ym. (2009) ovat listanneet pilvilaskennan kymmenen suurinta estettä ja mahdollisuutta. Mukana on useita suorituskykyyn liittyviä tekijöitä. Näitä ovat palvelun saavutettavuus, tiedonsiirtoon liittyvät pullonkaulakohdat, suorituskyvyn ennustamattomuus, nopea skaalattavuus ja skaalautuva tietovarasto. Seuraavaksi esitellään edellä mainittuja tekijöitä tarkemmin.

3.1.1 Saavutettavuus

Armbrustin ym. (2009) mukaan ainoa uskottava ratkaisu erittäin korkeaan saavutettavuuteen on useiden eri pilvipalveluntarjoajien hyödyntäminen. Vaikka tietyllä pilvipalveluita tarjoavalla yrityksellä olisikin useita palvelinkeskuksia maantieteellisesti eri paikoissa ja ne käyttäisivät eri verkkoja, niillä voi olla yhteinen ohjelmistoarkkitehtuuri ja kirjanpitojärjestelmät. Lisäksi yritysten lopettamisen riski on otettava huomioon.

Pilvien saavutettavuusongelmia on raportoitu eri web-sivustoilla. Esimerkiksi crn.com-sivusto (CRN 2011) on listannut kymmenen suurinta vuonna 2010 tapahtunutta katkosta pilvipalveluissa. Jotkut raportoiduista katkoksista ovat kestäneet useita tunteja. Tällaisissa tilanteissa taloudelliset tappiot voivat olla asiakkaille suuria ja kriittisten sovellusten tapauksessa tilanne on kestämätön. Pilvipalveluntarjoajat pyrkivät palvelutasosopimuksilla takaamaan tietyn saavutettavuuden asiakkaille. Esimerkiksi Amazon EC2:n palvelutasosopimuksessa (Amazon.com 2011) luvataan 99.95 %:n saavutettavuus vuoden aikana. Mikäli saavutettavuus laskee tämän lukeman alapuolelle, asiakkaalle luvataan 10 %:n hyvitys laskusta. Asiakkaan näkökulmasta hyvitys vaikuttaa melko vähäiseltä verrattuna mahdollisiin taloudellisiin tappioihin tai muihin vahinkoihin. Tässä yhteydessä mainittakoon, että palvelinkeskusten saavutettavuustasoja (ja datan replikointitasoa) varten on olemassa Telecommunications Industry Association -yhdistyksen (TIA 2011) määrittelemä TIA-942-standardi. Standardissa palvelinkeskukset luokitellaan neljään tyyppiin, joissa edellä mainitut tasot määritetään.

3.1.2 Tiedonsiirron pullonkaulakohdat ja soveltuvuus suurteholaskentaan

Pilven sisäinen verkkoteknologia ja kaistanleveyden puute mm. korkeaa laskentatehoa vaativiin sovelluksiin on eräs ongelmakohta. Lisäksi laajaverkon (Wide Area Network, WAN) kaistanleveys saattaa muodostua pullonkaulaksi. (Armbrust ym. 2009). Edellä esitettyä väitettä sisäisen verkon riittämättömyydestä tukee Hen ym. (2009) tekemä pilven suurteholaskentaa (High Performance Computing, HPC) koskeva tutkimus, jossa he ovat todenneet huonon verkon potentiaalin suurimmaksi ongelmaksi suurteholaskennan toteuttamiseksi pilvessä. Myös Napperin ja Bientinesin (2009) sekä Walkerin (2008) tutkimukset osoittavat, ettei pilven suorituskyky vielä riitä suurteholaskentaan. Pilvilaskenta onkin nykyisellään tarkoitettu

lähinnä liiketoiminnan sovelluksiin. (He ym. 2010). Pilviteknologiat kehittyvät kuitenkin nopeasti ja myös tunnettujen pilvijättiläisten ulkopuolelta voi löytyä suurteholaskentaan sopiva ratkaisu. Eräs kiinnostava yritys on Sabalcore (Sabalcore 2011), joka lupaa asiakkailleen maksuperusteista pilvipalvelua suurteholaskennan saralla.

3.1.3 Suorituskyvyn ennustamattomuus

Useat virtuaalikoneet osaavat jakaa prosessoritehon ja muistin hämmästyttävän hyvin, mutta levyoperaatioiden jakaminen on ongelmallisempaa ja niiden käyttäytymistä on vaikeampaa ennustaa. (Armbrust ym. 2009). Kohdassa 3.2 esitellään tarkemmin virtualisoitujen ympäristöjen suorituskykyä ja niihin tehtyjen suorituskykymittausten tuloksia. Lisäksi tämän kohdan aiheeseen liittyen kohdassa 3.3 esitellään pilven suorituskyvyn vaihteluun liittyviä tutkimuksia.

3.1.4 Nopea skaalattavuus ja skaalautuva tietovarasto

Pilven tulee skaalautua nopeasti kumpaankin suuntaan. Toisin sanoen kuormituksen kasvaessa ja laskiessa resurssien lisäämisen ja vähentämisen täytyy tapahtua nopeasti. Tämä on suorituskyvyn lisäksi kustannuskysymys. (Armbrust ym. 2009). Tähän liittyen Hill ym. (2010) ja Schad ym. (2010) nostavat artikkeleissaan esille palvelininstanssien käynnistymisajat. Ensimmäisessa tutkimuksessa on testattu käynnistymisaikoja Windows Azure -pilvialustalla ja jälkimmäisessä on tutkittu Amazon EC2:n käynnistymisaikojen vaihteluita. Hillin ym. (2010) mukaan palvelininstanssin käynnistymisaika on kriittinen tekijä pilvisovellusten dynaamisessa skaalattavuudessa. Windows Azurella suoritetuissa testeissä uuden instanssin käynnistäminen ja integroiminen kokoonpanoon kesti usein noin kymmenen minuuttia. Tämä voi olla liian pitkä aika, mikäli sovellukselta vaaditaan nopeaa skaalattavuutta. Schadin ym. (2010) tekemä tutkimus käynnistymisaikojen vaihteluista kahdessa

Amazonin palvelinkeskuksessa osoitti, että käynnistymisajoissa esiintyy huomattavia vaihteluita eri ajankohtina ja palvelinkeskusten kesken.

Armbrust ym. (2009) ottavat artikkelissaan erikseen esille skaalautuvan tietovaraston. Heidän mukaansa pysyvään skaalautuvaan tietovarastoon liittyy erityisiä haasteita. Avoimena tutkimusongelmana on luoda tietovarastojärjestelmä, joka täyttää tietyt tietovarastolle asetetut tarpeet ja skaalautuu mielivaltaisesti molempiin suuntiin. Lisäksi resurssienhallinnan suhteen tietovaraston tulee vastata ohjelmoijien odotuksia skaalattavuudesta, tiedon pysyvyydestä ja korkeasta saavutettavuudesta.

3.2 Virtualisoitujen tietokoneiden suorituskyky

Pilvipalveluntarjoajat hyödyntävät palvelinkeskuksissaan virtualisointitekniikkaa, jonka avulla fyysisestä palvelimesta saadaan useita virtuaalisia palvelininstansseja. Virtualisointitekniikkaa hyödyntävistä tietokoneista on tehty useita suorituskykyä koskevia tutkimuksia. Virtualisoitujen tietokoneiden suorituskykyyn liittyy haasteita ja ne ovat yleisesti tehottomampia kuin natiivit palvelimet.

Salokanto (2010) on tutkinut virtualisoitujen tietokoneiden puhdasta laskentatehoa ja levynopeutta. Hän on havainnut, että prosessoritehon suhteen virtualisoinnin tuoma lisäkuorma on mitätön; ainoastaan noin yhden prosentin. Sen sijaan levynopeustesti osoitti, että virtuaaliympäristön suorituskyky on selvästi heikompi; ”virtuaaliympäristöille ominaiseen tapaan suorituskyky vaihteli paljon, mutta mittaukset osoittivat kirjoitusnopeuden tippuvan ajoittain käyttökelvottoman hitaaksi”. (Salokanto 2010, 58). Testeissä käytettiin VMware ESX ja Sun xVM -virtuaalialustoja. Hanin ym. (2009) suorittamat testit tukevat edellä esiteltyjä havaintoja prosessoritehosta. Testien mukaan esimerkiksi VMware:n tapauksessa suurin osa prosessoritehoa vaativista operaatioista oli noin prosentista muutamaan prosenttiin hitaampia kuin natiivissa palvelimessa. Joidenkin operaatioiden osalta suorituskyky osoittautui kuitenkin

jopa 30 %:a heikommaksi. Virtuaaliympäristöjen levynopeuksista on tehty tutkimuksia, jotka puoltavat Salokannon (2010) tekemiä testejä. Shafer (2010) on tutkinut avoimeen lähdekoodiin perustuvan Eucalyptus-pilven (joitakin testejä myös kaupallisella Amazon EC2:lla) levyn suorituskykyä ja todennut, että se on merkittävästi heikompi kuin virtualisoimattomassa palvelimessa. Barkerin ja Shenoyn (2010) Amazon EC2:a koskevat testit niin ikään osoittivat, että levyn suorituskykyä ei pystytä eristämään virtuaalikoneille riittävän hyvin. Testeissä havaittiin suorituskyvyn heikkenevän jopa 75 %:a, kun tiettyä kuormitusta pidettiin yllä. Myös Yuanin ym. (2009) mukaan levynopeus on virtuaalisissa ympäristöissä kriittisin yksittäinen suorituskykyä koskeva tekijä. Wang ja Ng (2010) tutkivat virtualisoinnin vaikutusta Amazon EC2 -pilven verkon suorituskykyyn. He käsittelivät tutkimuksessaan virtuaalipalvelinten välisiä jaettuja prosessoreita, pakettien viiveitä, suoritustehoa (TCP/UDP) ja pakettien häviöitä. Tutkimus osoitti, että virtualisointi voi aiheuttaa epänormaaleja viiveitä ja merkittävää epävakaisuutta suoritustehoon.

Tietokoneiden virtualisoinnin suorituskyvystä ja sen vertaamisesta natiiveihin palvelimiin on tehty useita tutkimuksia. Edellä esitellyt tutkimukset osoittavat, ettei virtualisoitujen tietokoneiden suorituskyky ole niin hyvä ja vakaa kuin natiiveissa palvelimissa. Tutkimusten mukaan etenkin levynopeus on ongelmatekijä virtuaalisissa ympäristöissä.

3.3 Suorituskyvyn vaihtelu

Iosup ym. (2010) ovat analysoineet kaupallisten pilvipalvelujen suorituskyvyn vaihtelua pitkällä aikavälillä. Tutkimuksessa tarkasteltiin mm. palvelininstanssien käynnistymisaikoja, tiedonsiirtonopeuksia ja tietokantaan tehtyjen kyselyjen vasteaikoja. Keskeisenä tutkimustuloksena havaittiin, että noin puolessa testattavista palveluista esiintyy tiettyjä vuosittaisia ja päivittäisiä malleja, joiden mukaisesti suorituskyky vaihtelee. Joissakin palveluissa esiintyi korkeitakin vaihteluja kuukausittaisten mediaaniarvojen välillä. Lisäksi

tutkimuksessa havaittiin, että suurimmassa osassa palveluista esiintyy ajanjaksoja, jolloin suorituskyky on erityisen tasainen ja vakaa. Tutkimuksen mukaan suorituskyvyn vaihtelu voi olla tärkeä tekijä pilvipalvelun valinnassa ja sen vaikutus riippuu sovelluksesta. Tutkimuksessa käsiteltiin Amazonin ja Googlen pilvipalveluita. Myös Durkee (2010) ottaa artikkelissaan esille suorituskyvyn vaihtelun. Hän toteaa, että pilvipalveluntarjoajat antavat liikaa resursseja käytettäväksi, mikä johtaa vaihtelevaan ja ennustamattomaan suorituskykyyn. Artikkelissa todetaan, että jaetun infrastruktuurin vaarana on, että joidenkin asiakkaiden käyttömallit vaikuttavat muiden asiakkaiden suorituskykyyn. Tähän liittyen Mei ym. (2010) kirjoittavat pilvilaskentaa koskevassa artikkelissaan suorituskyvyn eristämisestä (performance isolation). Jos eristys toimisi täydellisesti, fyysisellä tietokoneella olevan virtuaalikoneen suorituskyky ei vaihtelisi, vaan pysyisi tasaisena riippumatta muista samalla tietokoneella toimivista virtuaalikoneista. Artikkelin mukaan virtuaalikoneet kilpailevat kuitenkin laskenta- ja tietoliikenneresursseista eivätkä nykyiset virtualisointitekniikat tarjoa riittävää suorituskyvyn eristystä. Tuloksena voi siis olla odottamatonta suorituskyvyn heikkenemistä. Myös Schadin ym. (2010) mukaan kilpailu virtualisoimattomista resursseista (mm. verkon kaistanleveys) on selkeästi tärkeimpiä tekijöitä pilven suorituskyvyn ennustamattomuuteen. Heidän Amazonin pilvipalveluita koskevat testinsä osoittivat, että suorituskyvyssä esiintyy suuriakin vaihteluita.

3.4 Yhteenveto

Pilven suorituskykyyn liittyy monia haasteita, joita on esitelty tässä luvussa. Alustan suorituskyky ja verkon latenssi ovat tekijöitä, jotka tulee ottaa huomioon suunniteltaessa sovelluksen toteuttamista pilvessä. Palveluntarjoajat ilmoittavat alustojen ominaisuudet eri tavoilla ja ilmoitetut ominaisuudet jättävät tulkinnanvaraa. Palveluiden viidakosta onkin haasteellista löytää tietylle sovellukselle suorituskyvyn (ja muidenkin ominaisuuksien) osalta paras ratkaisu. Verkon latenssin suhteen sovelluksen käyttäjän maantieteellinen

sijainti palvelinkeskukseen nähden on ratkaisevassa asemassa. Yleisesti pilven suorituskykyyn liittyviä haasteita ovat mm. saavutettavuuden ongelmat sekä sisäisen verkkoteknologian hitaus, joka koskee lähinnä suurteholaskentaa. Lisäksi nopeaan kaksisuuntaiseen skaalattavuuteen liittyy haasteita. Virtualisointi ja jaetut resurssit ovat asioita, joihin useat pilven suorituskykyyn liittyvät ongelmat kytkeytyvät. Tutkimusten mukaan virtualisointi aiheuttaa lisäkuorman, joka ilmenee erityisesti levynopeudessa. Esiteltyjen tutkimusten mukaan levyn suorituskyvyssä voi esiintyä suuriakin vaihteluita. Jaettujen resurssien suhteen tutkimukset osoittavat, ettei suorituskykyä pystytä eristämään riittävän hyvin virtuaalikoneille.

4 TIETOJÄRJESTELMÄN SUORITUSKYVYN MITTARIT

Tässä luvussa esitellään ensin lyhyesti tietojärjestelmän suorituskyvyn arviointia, jonka jälkeen esitellään keskeiset suorituskyvyn mittarit. Luvussa perehdytään viiteen keskeiseen käsitteeseen. Näistä käsitteistä vasteaika ja suoritusteho ovat käytetyimmät suorituskyvyn mittarit suorituskykytesteissä. Tutkimuksen kokeellisessa osassa testataan pilvipalvelinten suorituskykyä, minkä vuoksi tässä luvussa esitetään teoreettinen pohja testauksessa käytettäville mittareille.

Tietojärjestelmän suorituskykyä voidaan arvioida kolmella eri tekniikalla: analyyttisella mallinnuksella, simuloinnilla ja mittaamalla. (mm. Jain 1991, 30; Porotskiy & Fateev 1992). Analyyttinen mallinnus perustuu erilaisiin kaavoihin ja/tai algoritmeihin, joiden avulla suorituskykyä pyritään arvioimaan. Simulointimalleissa sen sijaan tietokoneohjelmien avulla jäljitellään järjestelmän dynaamista olemusta ja staattista rakennetta. (Menascé ym. 2004, 36-37). Analyyttista mallinnusta ja simulointia voidaan siis käyttää myös tilanteissa, joissa mittaus ei ole mahdollista. Yleisesti on kuitenkin uskottavampaa, että niiden käyttö perustuu aiempiin mittauksiin. Mittauksessa järjestelmän suorituskykyä ei mallinneta, vaan järjestelmän on oltava olemassa, jolloin sitä voidaan mitata halutuilla mittareilla. Kaikilla edellä mainituilla tekniikoilla on omat etunsa; esimerkiksi vaadittava aika, tarkkuus ja kustannukset vaihtelevat. (Jain 1991, 31). Tässä tutkimuksessa keskitytään palvelinkeskuksessa sijaitsevan sovelluksen suorituskyvyn mittaukseen, joten analyyttista mallinnusta tai simulointia ei esitellä enempää.

4.1 Tietojärjestelmän laatutekijät ja suorituskyvyn mittarit

Suorituskyky on yksi tietojärjestelmien keskeisistä laatutekijöistä ja se pyritään maksimoimaan mahdollisimman alhaisilla kustannuksilla. Suorituskyvyn käsite on moniulotteinen, sillä järjestelmän suorituskyvyllä voidaan tarkoittaa

useita eri asioita. Jollekin järjestelmälle tärkein suorituskyvyn kriteeri voi olla esimerkiksi palvelun saavutettavuus. Tällöin esimerkiksi 99.99 %:n saavutettavuus tarkoittaisi, että palvelu on hieman yli neljä minuuttia saavuttamattomissa 30 päivän ajanjaksolla. Verkkokaupassa tällaisella tuskin on suurtakaan merkitystä mutta esimerkiksi sairaalan kriittisissä järjestelmissä tilanne on aivan toinen. Hirvisalon (2000) mukaan suorituskyky on laajemmin ajateltuna ohjelmiston kykyä selviytyä sille kuuluvista tehtävistä käytettävissä olevilla resursseilla. Usein tärkeää on vasteaika, toisinaan taas suoritusteho tai laskennan vaatima tila. Ohjelmiston elinkaaren kannalta tärkeitä ovat suorituskyvyn skaalattavuus ja tasapaino. Tällöin suorituskyky kasvaa suhteessa laitteiston tehoon ja ohjelmiston eri komponentit ovat sopusoinnussa keskenään ja käyttävät tasaisesti laitteistoresursseja.

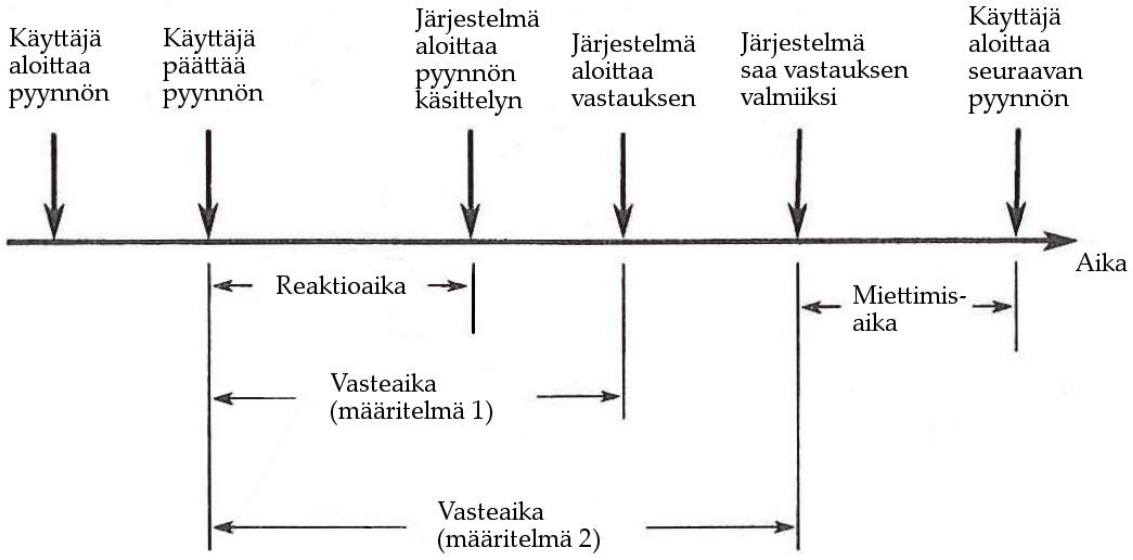
Grayn (1991, 3) mukaan järjestelmien suorituskyky vaihtelee valtavasti eri sovellusalueiden kesken. Jokainen järjestelmä on tyypillisesti suunniteltu muutamaa tehtävää varten ja voi olla kyvytön suorittamaan muita tehtäviä. Tietynlainen supertietokone voi esimerkiksi suorittaa tehokkaasti suurta laskentatehoa vaativia operaatioita, mutta samalla se voi olla soveltumaton liiketoiminnan sovelluksiin. Laitteiston nopeus vaikuttaa luonnollisesti järjestelmän suorituskykyyn, mutta pelkästään sen perusteella ei voida päätellä järjestelmän suorituskykyä. Esimerkiksi tietokantapohjaisissa järjestelmissä ohjelmoiduilla algoritmeilla on keskeinen merkitys suorituskykyyn. Lisäksi tällaisten järjestelmien sisällä esiintyy huomattavaa vaihtelua suorituskyvyssä. Tietty järjestelmä voi esimerkiksi olla erinomainen suorittamaan yksinkertaisia päivitysoperaatioita tietokantaan. Samalla se on kuitenkin hidas tekemään monimutkaisia kyselyitä samaan tietokantaan. Järjestelmien erilaisuuden vuoksi myös suorituskykytestien tulee olla sovellusaluekohtaisia. Niissä tulee määritellä tiettyä sovellusaluetta koskeva tyypillinen synteettinen kuorma. Tällä tavalla vertailemalla tiettyyn sovellusalueeseen kuuluvia sovelluksia,

saadaan karkea arvio niiden suhteellisesta suorituskyvystä tarkasteltavalla sovellusalueella.

Menascén ym. (2004, 12-19) mukaan järjestelmän laatutekijöitä ovat vasteaika, suoritusteho, saavutettavuus, luotettavuus, turvallisuus, skaalattavuus ja laajennettavuus. Jain (1991, 37-40) luokittelee yleisesti käytetyiksi suorituskyvyn mittareiksi vasteajan, suoritustehon, käyttöasteen, luotettavuuden ja saavutettavuuden. Seuraavaksi käsitellään edellä mainituista suorasti suorituskykyyn liittyvät käsitteet. Esimerkiksi turvallisuuden voidaan sanoa olevan epäsuorasti suorituskykyyn liittyvä käsite, sillä tietyillä järjestelmän turvallisuuteen liittyvillä ratkaisuilla on merkitystä myös suorituskykyyn (mm. Menascé 2003).

4.1.1 Vasteaika

Vasteajaksi (tai vastausajaksi) kutsutaan aikaa, joka järjestelmällä kestää vastata käyttäjän lähettämään pyyntöön. Jainin (1991, 37) mukaan tällainen määritelmä vasteajalle on yksinkertaistettu, sillä pyynnöt tai vastaukset eivät ole välittömiä. Vasteaika voidaan määritellä joko pyynnön lähettämisen ja vastauksen alkamisen väliseksi ajaksi tai pyynnön lähettämisen ja vastauksen päättymisen väliseksi ajaksi. Kummatkin määritelmät ovat selkeästi määriteltyinä hyväksyttäviä, mutta jälkimmäinen on parempi jos vastauksen alkamisen ja päättymisen välinen aika on pitkä. Jain (1991, 37-38). Kuvassa 3 esitetään vasteajan määritelmät sekä reaktioaika, jolla tarkoitetaan pyynnön lähettämisen ja sen käsittelyn aloituksen välistä aikaa.



Kuva 3 Vasteajan määritelmät (Jain 1991, 37)

Vasteaika koostuu järjestelmissä eri komponenttien käyttämistä ajoista. Esimerkiksi verkkopohjaisissa palveluissa selaimella lähetetyn pyynnön vasteaika voidaan jakaa kolmeen pääkomponenttiin: selaimen aika, verkon aika ja palvelimen aika. (Menascé ym. 2004, 13). Kuvassa 4 esitetään tarkemmin vasteajan jakautuminen pääkomponenttien sisällä.

Selaimen aika		Verkon aika			Palvelimen aika		
Prosessointi	I/O	Selaimen ja verkkopalvelun-tarjoajan välinen aika	Internet-aika	Verkkopalvelun-tarjoajalta palvelimelle kuluva aika	Prosessointi	I/O	Sisäinen verkon aika
Odotukseen käytetty aika							

Kuva 4 Vasteajan erittely verkkopalvelun tapauksessa (Menascé ym. 2004, 13)

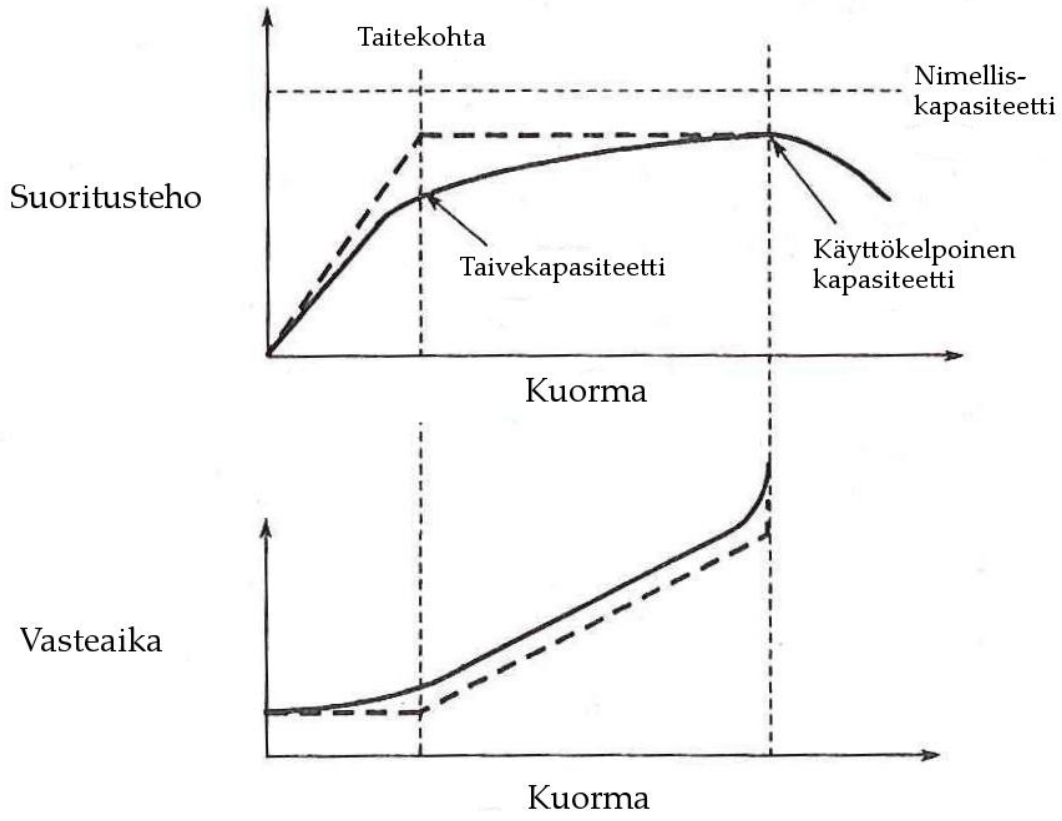
Selaimen aika sisältää pyynnön lähettämiseen ja vastaanotetun sivun näyttämiseen vaadittavan prosessointi- ja I/O-ajan. Verkon aika sisältää selaimelta verkkopalveluntarjoajalle (Internet Service Provider, ISP) kuluvan välitysajan, Internetiin käytetyn ajan ja verkkopalveluntarjoajan ja palvelimen

väliseen kommunikointiin käytetyn ajan. Viimeisenä palvelimen aika sisältää pyynnön prosessoimiseen kuluvan ajan, I/O-ajan ja sivuston sisäiseen verkon toimintaan kuluvan ajan. Mikä tahansa kolmesta komponentista sisältää eri resursseissa (prosessorit, levyt ja verkot) odotukseen käytetyn ajan. Tällöin puhutaan järjestelmässä olevasta ruuhkasta (congestion). (Menascé ym. 2004, 13).

4.1.2 Suoritusteho

Toinen tärkeä suorituskyvyn mittari on suoritusteho (throughput). Jain (1991, 38) määrittelee sen nopeudeksi (pyyntöjä tiettyä aikayksikköä kohden), jolla järjestelmä palvelee sille saapuvat pyynnöt. Esimerkiksi vuorovaikutteisissa järjestelmissä suoritustehoa mitataan pyyntöinä sekuntia kohden. Verkkopalvelussa suoritustehoa voidaan mitata esimerkiksi HTTP-pyyntöjen määrällä sekunnissa tai tavuina sekunnissa (Menascé ym. 2004, 14). Prosessorien tapauksessa suoritusteho voidaan ilmaista taas miljoonina käskyinä sekuntia kohden. Eri järjestelmissä ja komponenteissa käytetään siis eri yksiköitä ilmaisemaan suoritustehoa.

Yleisesti järjestelmän suoritusteho kasvaa kuormituksen kasvaessa. Tietyn kuormituksen jälkeen kasvu pysähtyy ja se saattaa jopa kääntyä laskuun. (Jain 1991, 38; Menascé ym. 2004, 15-16). Maksimaalista suoritustehoa ihanteellisella kuormituksella kutsutaan järjestelmän nimelliskapasiteetiksi (nominal capacity). Vasteaika maksimaalisella suoritusteholla on usein kuitenkin liian korkea, joten on kiinnostavampaa tietää maksimaalinen suoritusteho ennalta määritellyllä vasteajan hyväksyttävällä maksimiarvolla. Kuvassa 5 tätä kohtaa kutsutaan käyttökelpoiseksi kapasiteetiksi (usable capacity). (Jain 1991, 38-39).



Kuva 5 Järjestelmän kapasiteetti (Jain 1991, 38)

Monissa sovelluksissa suoritustehon tai vasteajan taitekohtaa pidetään optimaalisena pisteenä, sillä kuvan 5 osoittamalla tavalla vasteaika on tällöin suhteellisen pieni ja taitekohdasta eteenpäin suoritusteho kasvaa hitaasti. Taitekohdan suoritustehoa kutsutaan järjestelmän taivekapasiteetiksi (knee capacity). (Jain 1991, 38-39).

4.1.3 Saavutettavuus

Saavutettavuus määritellään ajanjaksoksi, jolloin järjestelmä on toiminnassa ja sen asiakkaiden käytettävissä. Kaksi pääsyytä järjestelmän saavuttamattomuuteen on toimintahäiriöt ja ylikuormitus. Ylikuormitus tapahtuu, kun kaikki järjestelmän komponentit ovat toimintakuntoisia, mutta järjestelmässä ei ole riittävästi resursseja uusien pyyntöjen käsittelemiseen. Tällaisessa tapauksessa järjestelmä tavallisesti hylkää siihen tulevat pyynnöt.

Esimerkiksi Web-palvelin saattaa hylätä uuden TCP-yhteyden, mikäli yhteyksien maksimimäärä on saavutettu. Toimintahäiriöt on syytä hoitaa nopeasti ja ensimmäinen askel niiden käsittelyyn on häiriöiden tunnistaminen. Tämän jälkeen häiriöiden syyt on tunnistettava, jotta järjestelmä saadaan normaaliin toimintakuntoon. Toimintahäiriöiden käsittely koostuu siis niiden tunnistamisesta, vianmäärityksestä ja palauttamisesta. (Menascé ym. 2004, 16-17).

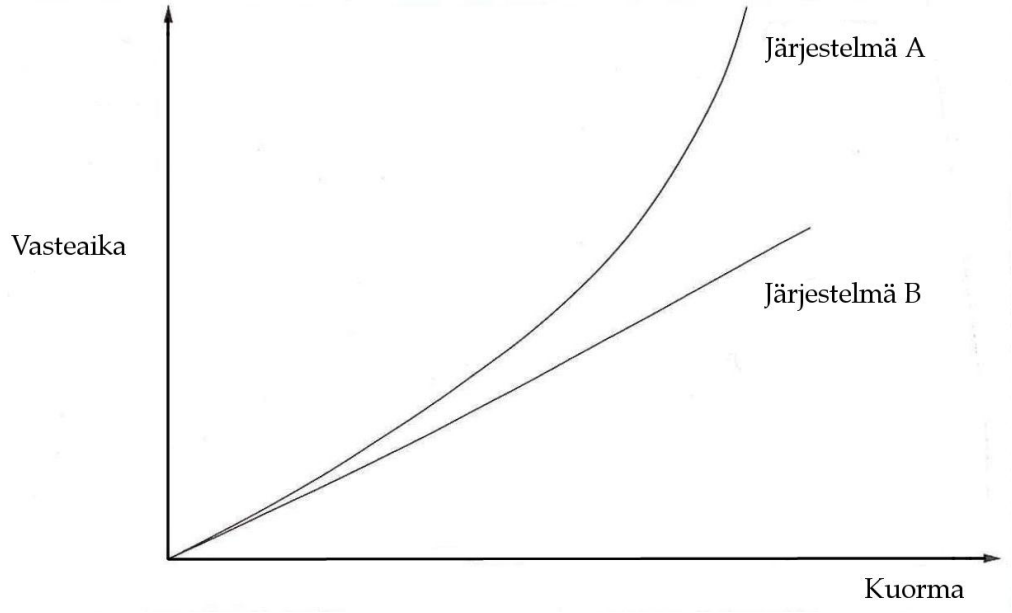
Toisinaan järjestelmissä halutaan kontrolloida ja rajoittaa saapuvien pyyntöjen määrää, jotta palvelun hyvä laatu saadaan varmistettua. Mikäli kontrollia ei ole, vasteajalla on taipumus kasvaa eksponentiaalisesti suhteessa kuormaan. Hyvä vasteaika taataan siis hylättyjen pyyntöjen kustannuksella. (Menascé ym. 2004, 17-18).

4.1.4 Käyttöaste

Jainin (1991, 39-40) mukaan resurssin käyttöastetta mitataan ajanjaksona, jona se palvelee sille saapuvia pyyntöjä. Näin ollen käyttöaste on palveluajan ja kokonaisajan suhde tietyllä aikavälillä. Järjestelmissä ollaan usein kiinnostuneita kuorman tasapainotuksesta, jolloin resursseja (mm. palvelimet, prosessorit, muistit) kuormitetaan tasaisesti. Aina tämä ei ole kuitenkaan mahdollista ja tietyn resurssin käyttöaste kasvaa muita suuremmaksi. Jainin (1991, 34) mukaan resurssia, jonka käyttöaste on suurin, kutsutaan järjestelmän pullonkaulaksi (bottleneck). Tällaisen resurssin suorituskyvyn optimointi on järjestelmälle kokonaisuudessaan varsin kannattavaa. Lisäksi on huomioitava, että järjestelmän sisäisten resurssien käyttöasteiden löytäminen on tärkeä osa suorituskyvyn arviointia.

4.1.5 Skaalattavuus

Järjestelmän voidaan sanoa olevan skaalautuva, mikäli sen suorituskyky ei heikkene merkittävästi, kun kuorma kasvaa. Esimerkiksi kuvassa 6 järjestelmän A vasteaika kasvaa epälineaarisesti kuorman suhteen, kun taas järjestelmässä B kasvu näyttää tapahtuvan selvästi kontrolloidummin. Näin ollen järjestelmän B voidaan sanoa olevan skaalautuva. (Menascé ym. 2004, 19).



Kuva 6 Skaalattavuus (Menascé ym. 2004, 20)

4.2 Yhteenveto

Tässä luvussa on esitelty tietojärjestelmän suorituskyvyn mittareita. Tärkeimpiä suorituskyvyn mittareita on vasteaika ja suoritusteho. Vasteajan määritelmä ei ole yksiselitteinen, vaan se määritellään joko aikaväliksi, jolloin käyttäjä lähettää pyynnön ja järjestelmä aloittaa vastauksen tai aikaväliksi, jolloin käyttäjä lähettää pyynnön ja järjestelmä saa vastauksen valmiiksi. Verkkopohjaisissa palveluissa vasteaika koostuu eri komponenttien käyttämistä ajoista. Pääkomponentteja on kolme: selaimen aika, verkon aika ja palvelimen aika. Vasteajan ohella suoritusteho on tärkeä suorituskyvyn mittari.

Suoritusteholla mitataan järjestelmän tai komponentin kykyä suorittaa sille saapuvat pyynnöt. Vasteajan ja suoritustehon lisäksi suorituskyvyn mittareina voidaan käyttää mm. saavutettavuutta, käyttöastetta, luotettavuutta ja skaalattavuutta.

5 SUORITUSKYKYTESTAUS

Tässä luvussa esitellään tietojärjestelmän suorituskykytestausta. Luku on jaettu kahteen pääkohtaan siten, että ensimmäisessä kohdassa esitellään suorituskykytestauksen tyyppejä. Toisessa kohdassa esitellään järjestelmän kuormittamisen periaatteita ja perehdytään lyhyesti suorituskykytesteihin.

Meierin ym. (2007) mukaan suorituskykytestauksella pyritään määrittämään järjestelmän vasteaika, suoritusteho, luotettavuus ja/tai skaalattavuus annetulla kuormituksella. Myös Gao ym. (2003, 230-232) ovat esitelleet edellä lueteltuja asioita teoksessaan. Suorituskykytestauksella pyritään yleisesti seuraaviin asioihin (Meier ym. 2007; Gao ym. 2003, 230-232):

- Tuotannon valmiuden arvioiminen
- Järjestelmän suorituskyvyn arviointi vaadittuja kriteerejä vastaan
- Suorituskyvyn ominaisuuksien vertailu eri järjestelmissä ja eri kokoonpanoilla
- Suorituskykyongelmien kartoitus
- Järjestelmän optimoinnin tukeminen
- Suoritustehon tasojen löytäminen

5.1 Suorituskykytestauksen tyypit

Tässä kohdassa esitellään suorituskykytestauksen tyyppejä. Kohdan sisältö perustuu Meierin ym. (2007) ja Subrayan (2006) teoksiin. Subrayan (2006) mukaan suorituskykytestauksen neljä perustyyppiä ovat kuormitustestaus (load testing), kestävyystestaus (endurance testing), stressitestaus (stress testing) ja piikkitestaus (spike testing). Meier ym. (2007) jaottelevat suorituskykytestauksen tyypit hieman eri tavalla. Tässä jaottelussa

suorituskykytestejä pidetään teknisinä tutkimuksina, joilla määritetään testattavan järjestelmän vasteajat, nopeus, skaalattavuus ja/tai vakaus. Jaottelussa kestävyystestaus luetaan kuormitustestauksen alalajiksi ja piikkitestaus stressitestauksen alalajiksi. Lisäksi kapasiteettitestaus (capacity testing) luetaan erilliseksi suorituskykytestauksen tyypiksi. Seuraavaksi esitellään edellä mainitut tyypit tarkemmalla tasolla.

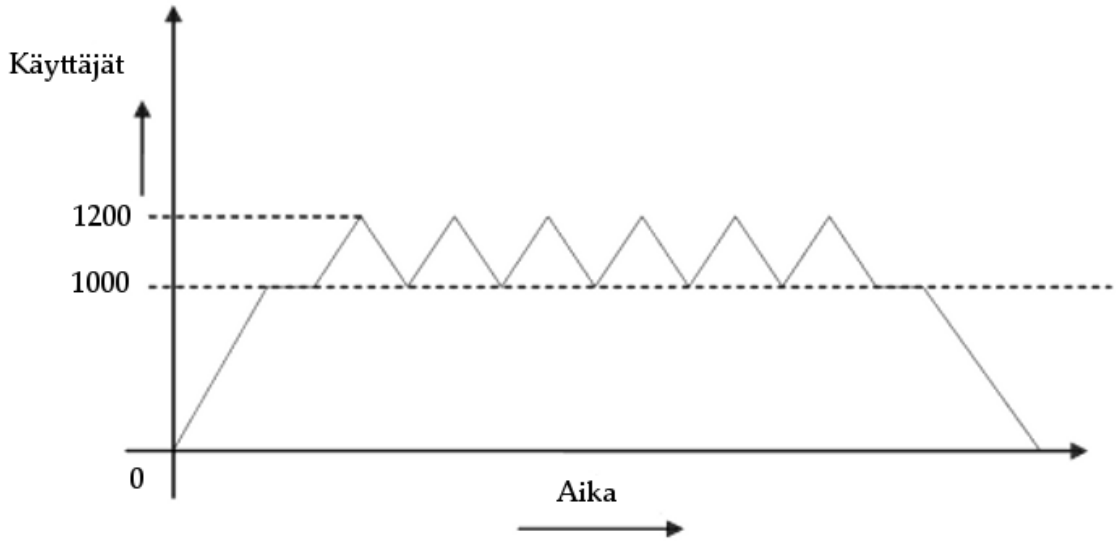
5.1.1 Kuormitus- ja kestävyystestaus

Kuormitustestauksessa testataan sovelluksen käyttäytymistä erilaisilla kuormituksilla. Kuormitustestauksen tarkoituksena on varmistaa, että sovellus täyttää sen suorituskyvylle asetetut tavoitteet. (Meier ym. 2007; Subraya 2006, 16). Nämä tavoitteet määritellään usein palvelutasosopimuksessa (Service Level Agreement, SLA). Kuormitustestaus mahdollistaa vasteaikojen, suoritustehon ja käyttöasteen mittaamisen. Lisäksi sovelluksen hajoamispiste (breaking point) voidaan määrittää. Tällöin oletuksena on, että hajoamispiste esiintyy piikkikuormituksen alapuolella. Kestävyystestaus voidaan lukea kuormitustestauksen alalajiksi. Siinä pyritään selvittämään, kuinka testattavan sovelluksen suorituskyky käyttäytyy pidemmällä aikavälillä. Tällä tavalla voidaan testata joitakin hitaasti kasvavia suorituskykyongelmia, joita lyhytkestoisilla testeillä ei voida havaita. (Meier ym. 2007).

5.1.2 Stressi- ja piikkitestaus

Stressitestauksen tarkoitus on löytää sovelluksessa olevat viat, jotka esiintyvät vain korkeilla kuormitusasteilla. Tällaiset viat voivat olla esimerkiksi synkronointiin tai muistivuotoihin liittyviä. Stressitestauksen avulla voidaan löytää sovelluksen heikot kohdat ja havaita, millainen sovelluksen käyttäytyminen on äärimmäisessä kuormituksessa. Piikkitestaus on stressitestauksen alalaji, jossa kuormitusta kasvatetaan toistuvasti korkeaan kuormitusasteeseen lyhyiden ajanjaksojen välillä. (Meier ym. 2007). Subrayan

(2006, 16) mukaan stressitestaus auttaa tunnistamaan kuormitustason, jonka järjestelmä pystyy käsittelemään ennen kuin se kaatuu tai sen suorituskyky heikkenee radikaalisti. Kuvassa 7 esitellään stressitestausta.



Kuva 7 Stressitestaus (Subraya 2006, 17)

5.1.3 Kapasiteettitestaus

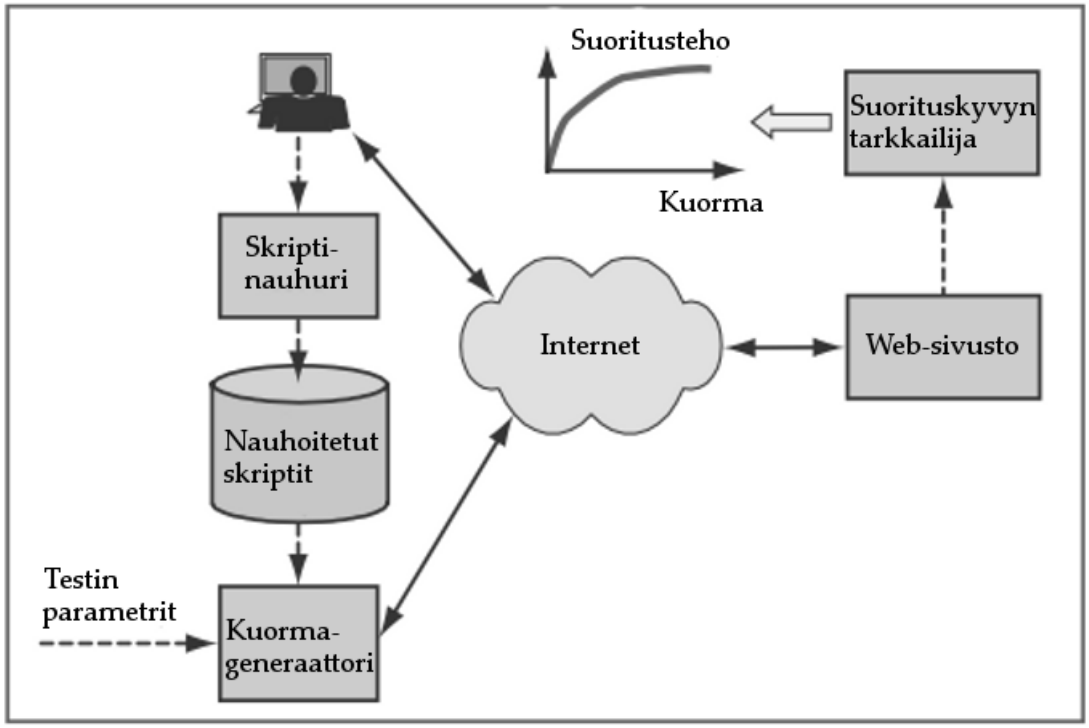
Kapasiteettisuunnittelun yhteydessä käytetään kapasiteettitestausta. Sen avulla voidaan selvittää millaisella kuormituksella sovellus pysyy sille annettujen raja-arvojen sisällä. Kapasiteettisuunnittelulla pyritään suunnittelemaan tulevaisuuden kasvua ja tällöin on tärkeää tietää, kuinka paljon lisää resursseja (mm. prosessorin ja levyn kapasiteetti, muistinkäyttö ym.) tarvitaan. (Meier ym. 2007).

5.2 Tietojärjestelmän kuormittaminen ja suorituskykytestit

Tässä kohdassa esitellään ensin testattavan järjestelmän kuormittamisen periaatteita, jonka jälkeen esitellään suorituskykytestien toimintaa.

5.2.1 Testattavan järjestelmän kuormittaminen

Käyttäjien simulointi tietokoneilla on suorituskykytestauksen keino, jolla suorituskykyä voidaan testata käyttäjän näkökulmasta. Yhdellä tietokoneella voidaan simuloida useita käyttäjiä kontrolloidusti ja toistaa testit. Tällaisia tietokoneita on kutsuttu mm. pääte-emulaattoreiksi (remote terminal emulator, RTE). (Jain 1991, 132). Pääte-emulaattori toimii siten, että se lähettää komentoja testattavaan järjestelmään. Komennot luetaan ns. skriptitiedostoista, jotka sisältävät komentojen lisäksi ohjeita. Ohjeissa voidaan määritellä esimerkiksi milloin komento lähetetään. Mittausoperaatio koostuu kolmesta vaiheesta: esiemulointi (pre emulation), emulointi ja jälkiemulointi (post emulation). Ensimmäisessä vaiheessa määritellään mm. käyttöskenaariot, joiden pohjalta luodaan käyttäytymistä vastaava skripti. Tässä kohdassa tunnistetaan mm. päätteiden ja käyttäjien ominaisuudet. Toisessa vaiheessa suoritetaan itse emulointi ja kerätään data. Käyttäjien suorittamien toimintojen jäljittely pääte-emulaattorilla voidaan suorittaa porrastetusti. Tyypillisesti käyttäjät esimerkiksi kirjautuvat järjestelmään satunnaisessa järjestyksessä. Tällaista käyttäytymistä voidaan jäljitellä määrittämällä tietty viive kirjautumisten välille. Viimeisessä vaiheessa luodaan mm. raportti ja päätetään kokeen toistosta. (Jain 1991, 133-135). Pääte-emulaattori -termiä on käytetty tällaisten laitteiden yhteydessä mm. useissa TPC:n suorituskykytesteissä (TPC 2010a). Myös muita nimityksiä vastaavan tyyppisille laitteille on olemassa. Vaikka pääte-emulaattori on jo vanha keksintö, sen keskeiset toimintaperiaatteet ovat edelleen voimassa nykyaikaisessa suorituskykytestauksessa. Kuvassa 9 esitellään järjestelmän kuormittamista Menascén (2002) mukaan.



Kuva 8 Järjestelmän kuormittaminen (Menascé 2002)

Kuvassa 8 skriptinauhurilla (script recorder) nauhoitetaan käyttäjän suorittamat toiminnot ja ne tallennetaan kuormageneraattoria (load generator) varten. Kuormageneraattori lähettää nauhoitettuihin skripteihin ja muihin testin parametreihin perustuvan kuorman web-sivustolle ja suorituskyvyn tarkkailija (performance monitor) mittaa suorituskyvyn. Toimintaperiaatteista voidaan havaita, että ne on hyvin samankaltaisia aiemmin esitellyn pääte-emulaattorin kanssa. Tämän tutkielman kokeellisessa osiossa simuloidaan käyttäjiä web-pohjaisessa ympäristössä edellä esitetyn kaltaisella tavalla. Kuormitukseen käytettävää laitetta nimitetään jatkossa yksinkertaisesti kuormitustyökaluksi.

5.2.2 Suorituskykytestit

Suorituskykytestit (benchmarks) ovat suorituskyvyn mittaamiseen tarkoitettuja ohjelmia tai algoritmeja, joiden tarkoituksena on mitata järjestelmän tiettyjä ominaisuuksia tai suorituskykyä laajemmassa merkityksessä. Krishnaswamy ja

Scherson (2000) jakavat suorituskykytestit kahteen kategoriaan: hienojakoiset (fine-grained) ja karkeajakoiset (coarse-grained) suorituskykytestit. Esimerkiksi tietokoneen yksittäisten operaatioiden suorituskyvyn mittaukseen käytettäviä suorituskykytestejä voidaan pitää hienojakoisina, kun taas laajempien ohjelmistojen suorituskykyä testaavat luetaan karkeajakoisiksi. Myös muunlaisia jaotteluja on olemassa, tosin perusajatus pysyy pitkälti samana. Esimerkiksi Ponder (1990) jakaa suorituskykytestit kolmeen perustyyppiin: mikroskooppisiin, makroskooppisiin ja yhdistelmiin. Ensimmäisellä tarkoitetaan hienojakoisia testejä, toisella karkeajakoisia ja viimeisellä näiden yhdistelmää.

Hienojakoiset suorituskykytestit mittaavat siis tiettyjen operaatioiden suorituskykyä. Tällaisia operaatioita ovat esimerkiksi kokonaislukujen käsittelyyn liittyvät operaatiot, liukulukuoperaatiot ja muistioperaatiot. Hienojakoisista suorituskykytesteistä saatavia tuloksia käytetään mm. järjestelmän pullonkaulakohtien tunnistamiseen ja niillä voidaan perustella järjestelmän soveltuvuutta tietylle sovellukselle. Lisäksi ne tarjoavat käsityksen järjestelmästä ja sovelluksesta. Tämän tyyppiset suorituskykytestit eivät kuitenkaan anna hyviä arvioita todellisesta käyttäjien luomasta kuormasta. Karkeajakoisissa suorituskykytesteissä tulokset koostuvat suhteellisen suurista määristä ohjelmakoodia. Esimerkiksi suorituskykyä testaavat sovellukset, kuten tunnetut TPC:n sovellukset, kuuluvat tähän ryhmään. Karkeajakoiset suorituskykytestit mallintavat järjestelmän luonnollista kuormaa ja ne soveltuvat hyvin, kun halutaan mitata suorituskykyä käyttäjän näkökulmasta. Ne tarjoavat kuitenkin vain vähän tietoa järjestelmän suunnittelua varten. (Krishnaswamy & Scherson 2000; Shan & Strohmaier 2010).

Suorituskykytestejä kehittävät tahot antavat toisinaan vain toteutusohjeet itse suorituskykytesteille. Mm. standardoiduissa TPC:n suorituskykytesteissä annetaan tarkat toteutusohjeet, joiden perusteella sovellus voidaan kehittää. Esimerkiksi elektronisen kauppapaikan toteutusohjeet on annettu TPC-W:n

määrittelydokumentissa (TPC 2010b). Näiden ohjeiden avulla voidaan rakentaa sovellus, jota kuormitetaan pääte-emulaattorin tyyppisillä laitteilla. TPC-W:n tapauksessa määrittelydokumentti on varsin laaja ja toteuttaminen on vaikeaa ja aikaa vievää. Tietyt tahot ovat kuitenkin julkaisseet valmiita toteutuksia Internetissä (esim. Cain ym. 2003). On myös hyvä huomioida, että esimerkiksi TPC vaatii auditoinnin, ennen kuin tiettyä järjestelmää koskevan testin tulokset saatetaan julkisuuteen. Tällä halutaan varmistaa tulosten oikeellisuus.

Tämän tutkielman kokeellisessa osassa suorituskykytestausta ei toteuteta valmiin suorituskykytestin avulla. Sen sijaan testattavana sovelluksena on tiettyyn sovellusalueeseen kuuluva web-pohjainen sovellus, jonka suorituskykyä testataan kuormittamalla sovelluksessa olevia toimintoja eri käyttäjämäärillä. Suorituskykytestauksessa käytettävä testisetti määritellään itse.

5.3 Yhteenveto

Tässä luvussa on esitelty suorituskykytestausta. Luvun ensimmäisessä pääkohdassa esiteltiin suorituskykytestauksen tyyppejä ja toisessa pääkohdassa tietojärjestelmän kuormittamista sekä suorituskykytestejä. Suorituskykytestauksen tyypit jaettiin ryhmiin siten, että kuormitus- ja kestävyystestaus sekä stressi- ja piikkitestaus muodostivat oman ryhmänsä. Lisäksi kapasiteettitestaus esiteltiin erillisenä tyyppinä. Ryhmiin jaon perusteena on se, että tietyt testaustyypit ovat samankaltaisia. Esimerkiksi kestävyystestaus toteutetaan kuormitustestausta vastaavalla tavalla, mutta nimensä mukaisesti se on pitkäkestoisempi. Luvun toisessa pääkohdassa esiteltiin testattavan järjestelmän kuormittamista simuloiduilla käyttäjillä. Tällä tavalla järjestelmää on mahdollista kuormittaa sen luonnollista kuormitusta vastaavalla tavalla. Kuormittaminen suoritetaan tietokoneilla, jotka lähettävät ennalta määritellyt pyynnöt testattavaan järjestelmään. Luvun lopuksi esiteltiin vielä suorituskykytestejä, joilla tarkoitetaan valmiita suorituskyvyn

mittaamiseen tarkoitettuja ohjelmia tai algoritmeja. Testit voidaan jakaa hieno- ja karkeajakoisiin testeihin. Hienojakoiset testit mittaavat yksittäisten operaatioiden suorituskykyä kun taas karkeajakoiset testit mittaavat suorituskykyä laajemmin.

6 WEB-POHJAISTEN JÄRJESTELMIEN SUORITUSKYKYTESTAUS

Tässä kohdassa esitellään web-pohjaisten järjestelmien suorituskykytestausta käytännönläheisestä näkökulmasta. Luvun sisältö toimii pohjana tutkielmassa myöhemmin toteutettavalle suorituskykytestaukselle.

Subraya (2006) jakaa suorituskykytestauksen vaiheet viiteen osaan, joista kolme ensimmäistä koskee testien valmistelua. Näissä vaiheissa määritellään mm. testiympäristö, kuorma ja työkalut. Neljäs vaihe koskee testien ajoa ja viides on ajon jälkeinen vaihe. Meier ym. (2007) ovat jaotelleet suorituskykytestauksen vaiheet seitsemään osaan, joissa tunnistetaan testiympäristö ja hyväksymiskriteerit, suunnitellaan testit, konfiguroidaan testiympäristö, toteutetaan testisuunnitelma, ajetaan testit sekä analysoidaan ja raportoidaan tulokset. Seuraavaksi esitellään testauksen vaiheet. Seuraavien kohtien sisältö perustuu Meierin ym. (2007) teokseen.

6.1 Testiympäristön tunnistus

Tässä kohdassa tunnistetaan fyysinen testiympäristö ja tuotantoympäristö. Lisäksi tunnistetaan ne työkalut ja resurssit, joita testissä käytetään. Testiympäristön kokonaisvaltainen ymmärtäminen mahdollistaa tehokkaamman testien suunnittelun ja auttaa tunnistamaan aikaisessa vaiheessa testaukseen liittyvät haasteet. Seuraavaksi listataan testiympäristöön liittyviä kriittisiä tekijöitä Meierin ym. (2007) mukaan.

- Laitteisto
    - Konfiguraatio ja fyysinen laitteisto (prosessori, keskusmuisti ym.)
- Verkko

    - Verkon arkkitehtuuri ja loppukäyttäjän sijainti. Lisäksi kuormantasaus sekä klusterin ja nimipalvelimen konfiguraatio on syytä huomioida.

- Työkalut
    - Kuormitustyökalun rajoitukset ja monitoroinnin vaikutus testiympäristöön.
- Ohjelmisto
    - Muut asennetut tai käynnissä olevat sovellukset, sovellusten lisenssirajoitukset ym.
- Ulkoiset tekijät
    - Muun verkossa tapahtuvan liikennöinnin määrä ja tyyppi, vuorovaikutus muiden järjestelmien kanssa, ajastetut eräprosessit, päivitykset ja varmuuskopiot.

Edellä lueteltujen asioiden lisäksi tässä vaiheessa tunnistetaan sen datan määrä ja tyyppi, jolla sovelluksen todellista käyttöä emuloidaan. Lisäksi kriittiset järjestelmän komponentit on tunnistettava ja arvioitava, mitkä komponentit ovat mahdollisia pullokaulakohtia. Tässä yhteydessä on myös hyvä arvioida, löytyykö asioita, joita testissä ei pystytä kontrolloimaan. Lisäksi tulisi myös varmistaa, että mm. palomuurit, nimipalvelin ja reititys ym. toimivat niin, että se vastaa mahdollisimman tarkasti tuotantoympäristön tilannetta.

6.2 Hyväksymiskriteerien tunnistus

Tässä kohdassa tunnistetaan tai ainakin arvioidaan niitä kriteereitä, joita sovelluksella halutaan suorituskyvyn suhteen olevan. Tyypillisesti vasteaikaan, suoritustehoon ja resurssien käyttöasteeseen asetetaan tiettyjä kriteerejä. Esimerkiksi suoritustehon kriteeriksi voidaan määrittää, että järjestelmän täytyy

käsitellä 25 kirjatilausta sekunnissa. Tällöin myös vasteajalle on syytä asettaa kriteerit.

6.3 Testien suunnittelu

Testien suunnitteluvaiheessa tunnistetaan käyttöskenaariot (usage scenarios), määritellään sopiva vaihtelevuus käyttäjien välille, tunnistetaan ja luodaan testidata sekä päätetään millaisia mittareita käytetään. Nämä asiat tarjoavat perustan kuormille ja kuormaprofiileille (workload profiles). Testejä suunniteltaessa on muistettava, että niiden tulisi vastata mahdollisimman tarkasti tuotantoympäristön tilannetta. Mikäli tuotantoympäristöä ei ole aiemmin käytetty, on luonnollisesti vaikeaa määrittää tarkkoja siihen perustuvia testejä.

Tämä vaihe voidaan jakaa kahteen osaan, joista ensimmäinen koskee sovelluksen käytön mallintamista ja toinen yksilöllistä käyttäjädatan ja vaihteluiden määrittelyä. Ensimmäinen osa käsittelee kuorman mallinnusta (workload modeling). Kuorman mallinnus voidaan määritellä yhden tai useamman käyttöprofiilin tunnistamisen prosessiksi suorituskykytestauksessa. Se koostuu seitsemästä vaiheesta, joita esitellään seuraavaksi. Tämän vaiheen toista kohtaa ei käsitellä tässä yhteydessä tarkemmalla tasolla. Toisen vaiheen pääasiallinen sisältö koostuu käyttäjien viiveiden mallintamisesta, yksilöllisen datan määrittämisestä ja käyttäjän poistumisesta sivustolta.

- Tavoitteiden tunnistus
    - Tavoitteiden tunnistamiseksi täytyy miettiä mm. ennustetun käytön määrää, eli kuinka paljon esimerkiksi tilauksia käsitellään tietyllä ajanjaksolla. Lisäksi tulee ottaa huomioon mm. kuormitushuippujen tasot, kuinka nopeasti oletetut kuormitushuiput saavutetaan ja kuinka pitkään ne kestävät.

- Tärkeimpien käyttöskenaarioiden määritys
    - Jokaisen mahdollisen tehtävän simuloiminen on epäkäytännöllistä, ellei jopa mahdotonta. Käyttöskenaarioihin on siis asetettava rajoitteita. Testeissä voidaan päättää, että käytetään esimerkiksi kaikkein yleisintä, suorituskykyä vaativaa tai liiketoiminnan kannalta kriittisintä skenaariota. Samantyyppisten sovellusten ajamiseen käytettävien web-palvelinten lokitiedostoja voidaan käyttää skenaarioiden määrittämisen apuna.
- Navigointipolkujen määritys käyttöskenaarioille
    - Tässä kohdassa määritellään, kuinka yksittäiset käyttäjät suorittavat skeenaarioihin liittyvät tehtävät. Navigointipolut voidaan määrittää tunnistamalla olemassa olevasta web-sovelluksesta merkittävästi suorituskykyyn vaikuttavat polut, lukemalla ohjekirjoja, yrittämällä ratkaista ne omalla päättelyllä tai havainnoimalla muita.
- Yksilöllisen käyttäjädatan ja vaihteluiden määritys
    - Vaikka navigointipolkuja ja käyttöskenaarioita esittävä malli olisi miten tarkka tahansa, se ei ole valmis. Käyttäjien yksilöllisyys ja käyttäjiin liittyvät vaihtelut on otettava huomioon. Tätäkin käyttäytymistä voidaan tutkia web-palvelimen lokitiedostoista. Esimerkiksi tietyn aikavälin sivulataukset, käyttäjäsessiot, sessioiden kestot, pyyntöjen hajonta, interaktioiden välinen aika (miettimisaika) ja poistuminen sivustolta ovat mittareita, jotka tarjoavat korkean tason näkemyksen sivuston käytöstä.
- Käyttöskenaarioiden suhteellisen jakautumisen määritys

  - Tässä kohdassa tulee määrittää, kuinka usein käyttäjät suorittavat kunkin mallin esittämän toiminnon suhteessa muihin toimintoihin.

- Tavoitteena olevien kuormitustasojen tunnistaminen
  - Tässä kohdassa määritellään sovelluksen käytön volyymi erilaisten käyttömallien (usage models) avulla.

- Valmistautuminen kuormamallin toteutukseen
  - Kuormamallin toteutus ajettavaksi testiksi on sidottu toteutusmenetelmään. Tyypillisesti toteutus tapahtuu luomalla skriptit siihen tarkoitetulla kuormitustyökalulla (vrt. pääte-emulaattori).

6.4 Testiympäristön konfigurointi

Tässä kohdassa valmistellaan testiympäristö, työkalut ja resurssit testisuunnitelman toteutukselle ja testien ajolle. Kuorman luomiseen ja sovelluksen monitorointiin tarkoitettavia työkaluja ei ole aina helppoa saada toimimaan halutulla tavalla. Ongelmat saattavat johtua esimerkiksi eristetystä verkkoympäristöstä, laitteistosta tai yhteensopimattomuudesta. Tässä vaiheessa on hyvä määritellä mm. kuinka paljon kuormaa voidaan luoda ennen kuin pullonkaulakohta saavutetaan. Tyypillisesti pullonkaulat löytyvät ensin muistista ja sitten prosessorista. Lisäksi kuormaa generoitaessa tulee harkita mm. useiden IP-osoitteiden simulointia (yleensä työkaluissa on tällainen ominaisuus) ja tarkkailla resurssien käyttöastetta palvelinten välillä.

6.5 Testisuunnitelman toteutus

Ajettavan testin luomisen yksityiskohdat ovat äärimmäisen työkalukohtaisia. Testeissä käytettävästä työkalusta riippumatta testin luominen sisältää

tyypillisesti yksittäisen käyttöskenaarion kirjoittamisen ja sen yhdistämisen muihin skenaarioihin valmiin kuormamallin esittämiseksi. Monesti suurin haaste tässä vaiheessa on luoda suhteellisen realistinen testi, jossa testattava järjestelmä ei huomaa eroa simuloitujen ja oikeiden käyttäjien välillä.

6.6 Testien ajo

Testien ajo ei tarkoita ainoastaan testien ja monitoroinnin käynnistämistä. Se on huomattavasti monimutkaisempi prosessi. Testien ajo voidaan nähdä seuraavaksi esiteltävien tehtävien yhdistelmänä.

- Testien ajon ja monitoroinnin koordinointi.
- Testien, konfiguraatioiden ja testiympäristön sekä datan validointi.
- Testien ajon aloittaminen.
- Skriptien, järjestelmän ja datan monitorointi ja validointi testien ollessa käynnissä.
- Ajon jälkeen tulosten nopea läpikäynti siltä varalta, että testissä on ollut virheitä tai puutteita.
- Testien, testidatan, tulosten ja muun testin toistamiseksi vaadittavan informaation arkistointi.
- Aloitus- ja päätösaikojen sekä tulosdatan kirjaaminen.

6.7 Tulosten analysointi ja raportointi

Lopuksi tulokset analysoidaan ja raportoidaan. Tässä kohdassa mm. kerättyä dataa verrataan hyväksymiskriteereihin sen määrittämiseksi, onko testattavan sovelluksen suorituskyky tavoitteiden kanssa linjassa vai ei. Lisäksi analyysi paljastaa monesti, että tulosten täydelliseen ymmärtämiseen tarvitaan lisää

mittareita. Tulosten esittämisen lisäksi vaaditaan tuloksiin perustuvia johtopäätöksiä ja dataa, joka vahvistaa johtopäätökset. Teknisestä näkökulmasta katsottuna analyysi, vertailut ja tiedot testien yksityiskohdista ovat tärkeitä.

6.8 Yhteenveto

Tässä luvussa on esitelty käytännönläheistä lähestymistapaa web-pohjaisten järjestelmien suorituskykytestaukseen. Testausprosessi etenee vaiheittain siten, että aluksi tunnistetaan testiympäristöön liittyvät asiat, kuten testattava järjestelmä ja käytettävissä olevat työkalut. Tämän jälkeen testausprosessissa edetään hyväksymiskriteereistä testisuunnitelmaan ja testiympäristön konfigurointiin. Testisuunnitelman toteutusvaiheen jälkeen prosessin loppuvaiheessa suoritetaan testien ajo sekä tulosten analysointi ja raportointi.

7 PALVELUNTARJOAJAT JA TESTATTAVA SOVELLUS

Tässä luvussa esitellään ne palveluntarjoajat, joiden alustoja tähän tutkielmaan sisältyvässä suorituskykytestauksessa testataan. Lisäksi kunkin palveluntarjoajan osalta esitellään tarjolla olevat palvelininstanssit ja niiden ominaisuudet. Testattavat palvelininstanssit esitellään tarkemmin seuraavassa luvussa testiympäristön määrittelyn yhteydessä. Toinen tässä luvussa esiteltävistä asioista on Drupal 6 -sisällönhallintasovellus (Drupal 2011a), joka asennetaan testattaville alustoille suorituskykytestausta varten.

7.1 Aikaisemmat tutkimukset

Drupalin käyttämisestä pilvialustoilla on keskusteltu jonkin verran Internetin eri sivustoilla. Sisällönhallintasovellusten alueella Drupal on yksi suosituimmista sovelluksista ja sen vieminen pilvialustoille herättää kiinnostusta. Myös Drupalin suorituskyvystä pilvialustoilla on tehty joitakin yksittäisiä testejä. Testejä on kuitenkin vain vähän ja ne ovat pääasiassa varsin suppeita. Lisäksi niissä ei vertailla eri palveluntarjoajien alustoja, vaan kaikki löydetyt testit koskevat Amazon EC2:n eri palvelininstansseja. Eräs tällaisista testeistä löytyy johnandcailin.com-sivustolta (John & Cailin 2011). Siinä on testattu Drupalin suorituskykyä kolmen eri Amazon EC2:n instanssityypin kesken ja kokeiltu myös erään optimointitekniikan vaikutusta suoritustehoon ja vasteaikoihin. Tässä yhteydessä on myös hyvä huomioida, että testien vasteajat vaihtelevat maantieteellisten sijaintien mukaan. Esimerkiksi edellä esitellyssä tutkimuksessa testien ajopaikkaa ei mainittu lainkaan.

7.2 Palveluntarjoajat ja palvelininstanssien ominaisuudet

Suorituskykytestiä varten on valittu kolme tunnettua pilvipalveluntarjoajaa. Kullakin palveluntarjoajalla on tarjolla erilaisia palvelininstansseja, joiden ominaisuuksia esitellään tässä kohdassa.

7.2.1 Amazon EC2

Amazon EC2 (Amazon.com 2010a) on tällä hetkellä selvästi tunnetuin infrastruktuuritason pilvipalvelu. Amazonin omistamat palvelinkeskukset sijaitsevat Yhdysvalloissa, Irlannissa ja Singaporessa. Vakioinstanssien (Standard On-Demand Instances) lisäksi palvelussa on tarjolla mm. suurella muistikapasiteetilla varustettuja instansseja (High-Memory On-Demand Instances) sekä korkean laskentatehon omaavia instansseja (High-CPU On-Demand Instances). Lisäksi kevyempää käyttöä varten on tarjolla ns. mikroinstansseja (Micro On-Demand Instances). Näiden instanssiryhmien sisällä on mikroinstansseja lukuun ottamatta mahdollisuus valita erikokoisia instansseja. Taulukossa 1 esitellään edellä mainittujen palvelininstanssien ominaisuuksia. (Amazon.com 2010d).

Taulukko 1 Palvelininstanssien ominaisuudet Amazon EC2:ssa

Tyyppi	**Muisti (GB)**	**Prosessoriydinten lukumäärä**	**Tallennuskapasiteetti (GB)**
Mikro	0.6	Ei ilmoitettu (max. 2 ECU-laskentayksikköä)	1-1000 (Elastic Block Store, EBS)
Vakio	1.7-15	1-4 (1-8 ECU-laskentayksikköä)	160-1690
Korkea muisti	17-68	2-8 (6.5-26 ECU-laskentayksikköä)	420-1690
Korkea laskentateho	1.7-7	2-8 (5-20 ECU-laskentayksikköä)	350-1690

Virtuaalisten prosessoriydinten lisäksi Amazon ilmoittaa kunkin instanssin prosessoritehon Amazonin omalla laskentayksiköllään (EC2 Compute Unit, ECU). Yhden tällaisen laskentayksikön prosessoritehon ilmoitetaan vastaavan 1.0-1.2 GHz:n Opteron tai Xeon -prosessoria.

7.2.2 Rackspace

Rackspace (Rackspace 2010) on tunnettu pilvipalveluita tarjoava yritys. Sen infrastruktuuritason pilvipalveluita tarjoavat palvelinkeskukset sijaitsevat Yhdysvalloissa, mutta myös Eurooppaan on avattu vuoden 2011 alkupuolella oma palvelinkeskus. Palvelininstanssien luokittelu Rackspacessa on toisenlainen kuin Amazon EC2:ssa. Seuraavaksi esitettävät tiedot perustuvat Rackspacen web-sivustoilla (Rackspace 2010; Rackspace 2011) esitettäviin palvelininstanssien ominaisuuksiin ja Rackspacen edustajan kanssa käytyyn keskusteluun. Palvelininstanssin prosessori on aina neliytiminen (Opteron 2374 HE) mutta siitä saatava vähimmäislaskentateho määräytyy instanssityypin mukaan. Esimerkiksi suurimmalla muistimäärällä (16GB) varustetusta palvelininstanssista saadaan laskentatehoa 100 %, kun taas puolella tästä muistimäärästä saavutetaan vähintään 50 %:n laskentateho. Lisäksi levyn tallennuskapasiteetti ja siirtokapasiteetti vaihtelee. Esimerkiksi pienimmälle palvelininstanssille luvataan 10MB/s siirtokapasiteetti ja suurimmalle 70MB/s. Lisäksi Rackspacessa on automaattisesti prosessoriaikaa lisäävä toiminto (CPU Bursting), jolla mahdollista ylimääräistä prosessoriaikaa kohdennetaan instansseille ilmaiseksi. Taulukossa 2 esitellään palvelininstanssien ominaisuuksia Rackspacessa.

Taulukko 2 Palvelininstanssien ominaisuudet Rackspacessa

Tyyppi	**Muisti (GB)**	**Prosessorista saatava laskentateho**	**Tallennuskapasiteetti (GB)**
1	0.25	1/64	10
2	0.5	1/32	20
3	1	1/16	40
4	2	1/8	80

(jatkuu)

Taulukko 2. (jatkuu)

5	4	1/4	160
6	8	1/2	320
7	16	1/1	620

Taulukossa 2 esitellyt prosessorin laskentatehot ovat tietylle palvelininstanssille luvatut vähimmäistehot. Jokaisella instanssilla on kuitenkin mahdollisuus käyttää kaikki saatavilla oleva laskentateho. Todellinen laskentateho määräytyy sen mukaan, miten laskentatehoa on käytettävissä tietyllä fyysisellä palvelimella. Näin ollen instansseja käytettäessä laskentatehon voidaan olettaa olevan korkeampi kuin ilmoitettu vähimmäislaskentateho. Tyypin 7 palvelininstanssilla laskentateho on siis aina täydet 100 % ja siitä alaspäin vähimmäislaskentateho puolittuu keskusmuistin mukaan. Palvelininstanssien taustalla oleva fyysinen palvelin sisältää neljä neliytimistä prosessoria eli yhteensä 16 ydintä.

7.2.3 GoGrid

GoGrid (GoGrid 2010) on myös yksi suurista ja tunnetuista Yhdysvaltalaisista pilvipalveluntarjoajista. GoGridillä on kaksi palvelinkeskusta, jotka sijaitsevat Yhdysvaltojen itäisellä ja läntisellä puolella. Tarjolla olevat palvelut vaihtelevat hieman palvelinkeskusten välillä. Läntisessä palvelinkeskuksessa palveluiden määrä on kirjavampi, kun taas itäisessä palvelinkeskuksessa kaikki palvelut eivät ole käytettävissä. Palvelininstanssien luokittelu GoGridissa on suhteellisen yksinkertainen; valittavana on kuusi erilaista palvelininstanssia muistin, prosessoriydinten ja tallennuskapasiteetin mukaan. Muisti ja tallennuskapasiteetti tuplaantuvat aina suurempaan instanssityyppiin siirryttäessä. Prosessoriydinten lukumäärä taas määräytyy hieman toisella tavalla. Taulukossa 3 esitellään palvelininstanssien ominaisuudet GoGridissa.

Taulukko 3 Palvelininstanssien ominaisuudet GoGridissa

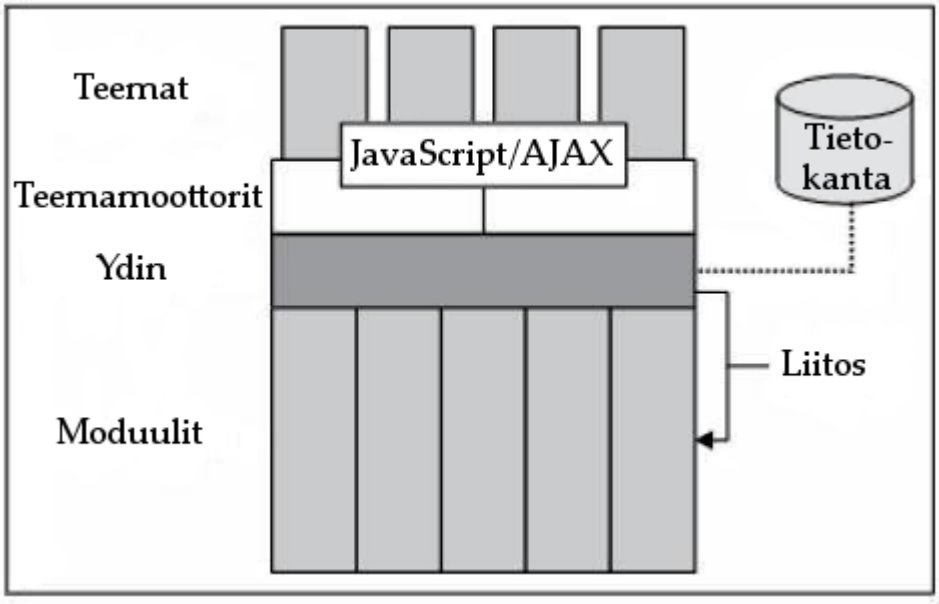
Tyyppi	Muisti (GB)	Prosessoriydinten vähimmäismäärä	Prosessoriydinten enimmäismäärä	Tallennuskapasiteetti (GB)
1	0.5	0.5	1	25
2	1	1	1	50
3	2	1	2	100
4	4	3	4	200
5	8	6	8	400
6	16	8	16	800

Ilmoitettu prosessoriydinten lukumäärä aiheuttaa hämmennystä. GoGrid ilmoittaa web-sivustoillaan (GoGrid 2011), että prosessoritehon allokointi riippuu sen fyysisen laitteiston profiilista, jolla pilvipalvelin sijaitsee. Tämän tiedon perusteella on mahdotonta sanoa, miten prosessoriteho määräytyy milloinkin. Otin henkilökohtaisesti yhteyttä GoGridin edustajaan ja kysyin asiasta. GoGridin edustaja kertoi, että pilvipalvelinten taustalla toimivan fyysisen palvelimen prosessoriydinten lukumäärä on joko 16 tai 32 ja prosessoriytimet jaetaan yllä olevan taulukon mukaisesti. Tiettyä fyysistä laitteistoa, jolla palvelinstanssit toimivat, on kuitenkin mahdotonta valita itse. Näin ollen kokeileminen on nähtävästi ainoa varma keino varmistaa kunkin palvelininstanssin prosessoritehon allokointi. Kysyin edustajalta myös palvelininstanssien kaistanleveydestä (siirtokapasiteetti Internetiin), sillä palvelininstanssien ominaisuuksissa ei tätä ilmoitettu. Edustajan mukaan kaistanleveyden allokointi tapahtuu siten, että kullekin käyttäjätilille varataan 1GB/s kaistanleveys. Tämä siis jaetaan käyttäjätilillä toimivien palvelininstanssien kesken.

7.3 Testattava sovellus

Tässä tutkimuksessa suorituskykyä testataan kuormittamalla web-pohjaista Drupal 6 -sisällönhallintasovellusta. Se on suosittu PHP-ohjelmointikielellä toteutettu avoimen lähdekoodin sovellus. Maailmanlaajuisesti julkaistaan satoja Drupal-sovelluksia päivittäin. Suomessa sovellusta käyttää mm. YLE ja Nokia (Drupal 2010a). Drupalin perustoiminnallisuuksiin kuuluu mm. sisällön lisääminen ja muokkaaminen, kommentointi, hakutoiminnot ja oikeuksien hallinta. Lisäksi eri tahot ovat kehittäneet sovellukseen lukuisia moduuleja eri käyttötarkoituksia varten. Sovellusta käytetään useimmiten palvelimilla, joissa käyttöjärjestelmänä on Linux, web-palvelimena Apache ja tietokantana MySQL. Käyttöjärjestelmäksi kelpaa myös Windows ja OS X. Web-palvelimena voidaan käyttää myös Microsoft IIS:ää ja tietokantana MySQL:n sijasta mm. PostgreSQL:ää. (Drupal 2010b).

Drupalin arkkitehtuuri voidaan jakaa teemoista, teemamoottorista (theme engine), ytimestä, moduuleista ja tietokannasta koostuviin osiin. Kuvassa 9 esitellään Drupalin arkkitehtuuria. (Butcher 2008, 8).



Kuva 9 Drupalin arkkitehtuuri (Butcher 2008, 8)

Ylimpänä arkkitehtuurissa on ns. teemakerros (teemat ja teemamoottorit), jonka tehtävänä on luoda HTML (tai JavaScript, XML ym.), jonka selain vastaanottaa. Tiedon esittäminen pidetään tämän kerroksen avulla erillään sisällöstä. Kuvan keskellä olevan Drupalin ytimen tehtävänä on tarjota sovelluksen perustoiminnallisuudet ja yhteys tietokantaan. Perustoiminnallisuuksia käytetään muiden järjestelmän osien tukemiseen. (Vandyk 2008, 2-5). Sovellus on rakennettu modulaariseksi, joten lisämoduulit on mahdollista liittää sovellukseen kätevästi. Drupalin (6.x) ydin koostuu kuudesta välttämättömästä moduulista sekä 28 valinnaisesta moduulista. Taulukossa 4 esitellään Drupalin pakolliset moduulit.

Taulukko 4 Drupalin pakolliset moduulit

Moduuli	**Kuvaus**
Block	Pääsisällön ympärillä näytettävien laatikoiden kontrollointi
Filter	Sisällön suodatus sen näyttämistä varten
Node	Mahdollistaa sisällön lähettämisen ja näyttämisen sivustolla
System	Yleinen sivuston konfiguraation hallinta ylläpitäjille
User	Käyttäjien rekisteröinnin ja kirjautumisen hallinta

7.4 Yhteenveto

Tämän luvun alussa esiteltiin lyhyesti aiempaa aiheesta tehtyä tutkimusta ja todettiin, että tutkimuksia on tehty vain vähän. Lisäksi tutkimusten todettiin keskittyvän Amazonin EC2 -pilvialustalle. Pilvipalveluntarjoajien ja palvelininstanssien esittelyssä havaittiin, että palvelininstanssien ominaisuudet ovat hyvin erilaisia eri palveluissa. Amazon esimerkiksi luokittelee palvelinstanssit täysin omalla tavallaan, kun taas Rackspacen ja GoGridin

luokittelusta voidaan löytää yhteisiä piirteitä. Amazonilla on myös suurimpana palveluntarjoajana laajin palvelininstanssien valikoima. Palvelinstanssien ominaisuuksien esittely vahvisti Durkeen (2010) väitteet palveluntarjoajien epämääräisyydestä; ominaisuudet jättävät tulkinnanvaraa ja kaikkia tietoja on vaikeaa saada edes selville. Luvussa esiteltiin myös testattavaa sovellusta (Drupal) ja sen arkkitehtuuria sekä perustoiminnallisuuksia. Sovelluksen todettiin tukevan eri käyttöjärjestelmiä ja tietokantoja. Lisäksi Drupalin kehittyneessä modulaarisessa arkkitehtuurissa esityslogiikka on erotettu sisällöstä ja myös uusien moduulien liittäminen osaksi arkkitehtuuria onnistuu kätevästi.

8 SUORITUSKYKYTESTAUKSEN VALMISTELU

Tässä luvussa käydään yksityiskohtaisesti läpi suorituskykytestauksen testiympäristö ja testisuunnitelma. Luvun sisällössä hyödynnetään luvussa 6 esiteltyä (Meier ym. 2007) lähestymistapaa suorituskykytestaukseen. Itse testaus toteutetaan kuormitustestauksena ja toisaalta myös piikkitestauksena (luku 5). Testin tarkoitus on selvittää kunkin palvelininstanssin suoritusteho, vasteajat ja muut tarpeelliset tiedot eri käyttäjämäärillä. Huippukuormituksessa testausta voidaan pitää piikkitestauksena, sillä siinä kuormitus nostetaan korkeaksi lyhyen ajanjakson sisällä.

8.1 Testiympäristö

Tässä kohdassa esitellään testiympäristö. Testiympäristöön kuuluu testattavat palvelininstanssit, testattava sovellus ja sen kokoonpano, testien ajopaikka ja käytettävä verkko, kuormittamiseen käytettävä laitteisto ja ohjelmisto sekä muut testiympäristöön liittyvät asiat.

8.1.1 Testattavat palvelininstanssit

Jokaisesta testattavasta palvelusta valitaan kolme erilaista palvelininstanssia suorituskykytestausta varten. Palvelininstanssit ovat ominaisuuksiltaan hyvin erilaisia jokaisessa palvelussa, joten toisiaan vastaavien instanssien valitseminen eri palveluista ei ole mahdollista. Lisäksi palvelininstanssien ominaisuudet ja käytettävissä olevat resurssit ilmoitetaan eri palveluissa eri tavoilla (kohta 7.2), ja näin ollen todellista suorituskykyä on vaikeaa arvioida. Palvelininstanssien valinta tehdään siten, että kultakin palveluntarjoajalta valitaan kustannuksiltaan edullisimpia instansseja. Näin ollen suurimmat ja tehokkaimmat palvelininstanssit jäävät testauksen ulkopuolelle. Taulukossa 5 esitellään testattavat palvelininstanssit ja niiden ominaisuudet. Taulukossa on listattu myös palvelininstanssin (käyttöjärjestelmänä Linux) yksikköhinta tuntia

kohden (palvelininstanssin olemassaoloajalta). Tarkempaa hinnan arviointia varten myös tiedonsiirtomaksut tulisi huomioida. Lisäksi palveluissa voi olla muitakin vaihtoehtoja kuin tuntiveloitteinen hinnoittelu. Tässä tutkielmassa palveluiden käytön hintaa ei kuitenkaan arvioida tarkemmalla tasolla.

Taulukko 5 Testattavat palvelininstanssit ja niiden ominaisuudet

Palvelu	**Muisti (GB)**	**Prosessori**	**Prosessoriydinten lukumäärä**	**Levy (GB)**	**Hinta/h ($)**
Amazon EC2 (micro)	0.6	Xeon E5430 2.66	Ei ilmoitettu (max. 2 ECU-laskentayksikköä)	EBS (1-1000)	0.025
Amazon EC2 (small)	1.7	Xeon E5430 2.66	1 (1 ECU-laskentayksikkö)	160	0.095
Amazon EC2 (High-CPU Medium)	1.7	Xeon E5410 2.33	2 (5 ECU-laskentayksikköä)	350	0.19
Rackspace	0.5	Opteron 2374 2.2	4	20	0.03
Rackspace	1	Opteron 2374 2.2	4	40	0.06
Rackspace	2	Opteron 2374 2.2	4	80	0.12
GoGrid	0.5	Xeon E5520 2.27	0.5-1	25	0.095
GoGrid	1	Xeon E5520 2.27	1	50	0.19
GoGrid	2	Xeon E5520 2.27	1-2	100	0.38

Taulukko 5 osoittaa, että palvelininstanssit ovat hyvin erilaisia eri palveluntarjoajilla. Amazon ilmoittaa esimerkiksi laskentatehon omalla ECU-yksiköllään, jolla on tietty vaihteluväli (kohta 7.2.1). Näiden laskentayksiköiden määrä vaihtelee palvelininstanssien kesken. Sitä vastoin Rackspacella kaikille palvelininstansseille luvataan neliytiminen prosessori, mutta siitä saatava laskentateho vaihtelee (kohta 7.2.2). GoGridin instansseille prosessoriteho määritetään taas prosessoriytiminä, joilla on tietty vähimmäis- ja enimmäismäärä (kohta 7.2.3). Edellä taulukossa 5 esiteltyjen ominaisuuksien lisäksi LAMP-ympäristön kokoonpanojen yksityiskohtaiset tiedot esitellään palveluntarjoajittain taulukossa 6.

Taulukko 6 LAMP-ympäristön yksityiskohtaiset tiedot palveluntarjoajittain

Palvelu	**Käyttöjärjestelmä**	**Apache-versio**	**MySQL-versio**	**PHP-versio**
Amazon EC2	Basic 32-bit Amazon Linux 2010.11.1 Beta	2.2.16	14.14	5.3.3
Rackspace	32-bit CentOS 5.4	2.2.3	14.12	5.1.6
GoGrid	32-bit CentOS 5.3	2.2.3	14.12	5.1.6

8.1.2 Testattavan sovelluksen kokoonpano

Drupal 6 -sovellusta testataan siten, että kaikki sovelluksen sisäiset ja ulkoiset suorituskyvyn optimointitekniikat ovat pois käytöstä. Sovelluksen kokoonpanossa tarvitaan pakollisten ydinmoduulien lisäksi Kommentointimoduuli (Comment module), hakumoduuli (Search module) ja CCK (Content Construction Kit) -moduuli, jolla kuvat voidaan liittää sivuston tekstien yhteyteen. Tämä onnistuu Content-, FileField- ja ImageField -moduulien avulla, jotka ovat osa CCK-moduulia. Testiä varten sovellukseen on

generoitava myös testitietokanta. MySQL-tyyppiseen tietokantaan luodaan 1000 käyttäjää ja 1000 kirjoitusta. Lisäksi erillisiin tietokantatauluihin luodaan linkit kuvatiedostoihin. Kuvat esitetään sivuston kirjoitusten yhteydessä. Tietokannan generointi tapahtuu työkalulla, jolla tarvittavat SQL-lauseet ja niihin sisältyvät satunnaiset käyttäjänimet, salasanat, sähköpostiosoitteet ja muut asiat voidaan luoda. Liitteissä 1-4 esitellään testattavan sovelluksen etusivu, satunnainen sivu, hakusivu ja kommentointisivu.

8.1.3 Verkko, sijainnit ja ajoajat

Testit ajetaan Jyväskylän yliopistossa (Agora) sijaitsevilta tietokoneilta, jotka on kytketty yliopiston verkkoon. Testattavat pilvipalvelimet sijaitsevat Euroopassa ja Pohjois-Amerikassa. Amazonin palvelininstanssit sijaitsevat Irlannissa, Rackspacen Chicagossa tai Dallasissa ja GoGridin Yhdysvaltojen länsipuolella San Franciscossa. Kaikki testit ajetaan arkipäivisin klo 8-20 välillä (Suomen aikaa). Testit toteutetaan ns. piikkitestauksena, jolloin ne ovat suhteellisen lyhytkestoisia (max. n. 30 min / palvelininstanssi / käyttäjämäärä). Korkeammilla käyttäjämäärillä yksittäinen testi on pitkäkestoisempi, sillä käyttäjämäärän nostaminen huippukuormitukseen vie enemmän aikaa. Testit ajetaan tammikuussa 2011.

8.1.4 Testauslaitteisto ja kuormitustyökalu

Testauslaitteistona käytetään Intel Core 2 Duo -suorittimilla (E6550 2.33GHz) varustettuja tietokoneita. Kuormitustyökaluna käytetään Apache JMeter -sovellusta (Apache Software Foundation 2010). Se on avoimeen lähdekoodiin perustuva suosittu Java-ohjelmointikielellä kirjoitettu sovellus. JMeterillä voidaan testata sekä staattisia että dynaamisia resursseja. Testattavan palvelimen ja siellä olevan sovelluksen kuormittaminen tapahtuu siten, että työkalulla luodaan useita samanaikaisia asiakasyhteyksiä (tässä tutkimuksessa TCP/IP + http) palvelimeen. Kuormitustyökalun sisällä tämä on mahdollista

monisäikeisen ohjelmoinnin ansiosta. JMeter voidaan asentaa myös useaan tietokoneeseen, mikä mahdollistaa samanaikaisten käyttäjien simuloinnin laajemmassakin mittakaavassa. Tällaisessa kokoonpanossa yksi tietokone toimii isäntäkoneena ja muut sen alaisuudessa. JMeterissä on mahdollista käyttää ohjelmoinnin perusrakenteita, kuten ehtolauseita ja silmukoita. Näiden avulla voidaan rakentaa mm. navigointipolkuja, joissa esimerkiksi tiedostossa olevia sivuston osoitteita käydään läpi halutulla tavalla. Lisäksi mm. viiveiden mallintaminen on mahdollista ajastimien avulla. Kuuntelijoita (listeners) käyttämällä saadaan esille pyyntöihin kuluvat vasteajat, suoritusteho ja muut tarpeelliset raportteihin tarvittavat tiedot.

8.1.5 Muut asiat

Testiin liittyvä ulkoinen tekijä on verkossa tapahtuva muu liikennöinti, jota ei voida kontrolloida. Lisäksi pyyntöjen reititystä palveluntarjoajille ei voida kontrolloida. Kussakin testissä käsiteltävän datan määrä on suhteellisen vähäinen; yksittäisten instanssien testauksessa pyyntöjen määrän arvioidaan olevan enintään joitakin kymmeniä pyyntöjä sekuntia kohden. Ladattavien sivujen koot ovat myös suhteellisen pieniä, joten verkon kaistan käyttöaste ei nouse kovin suureksi edes täydessä kuormituksessa.

8.2 Testisuunnitelma

Tässä kohdassa esitellään testisuunnitelma Meieriä ym. (2007) mukaillen. Suunnitelma koostuu tavoitteiden tunnistuksesta, tärkeimpien käyttöskenaarioiden määrityksestä, navigointipolkujen määrityksestä käyttöskenaarioille, yksilöllisen käyttäjädatan ja vaihteluiden määrityksestä, käyttöskenaarioiden suhteellisen jakautumisen määrityksestä sekä kuormamallin toteutuksen valmistelusta.

8.2.1 Tavoitteiden tunnistus

Tässä kohdassa tulee määrittää sovelluksen ennustetun käytön määrä ja kuormitushuippujen tasot sekä kuinka nopeasti oletetut tasot saavutetaan ja kuinka kauan ne kestävät. Koska kyseessä ei ole tuotantokäytössä oleva sovellus ja aiempaa dataa ei siis ole saatavilla, ennusteita ei voida toteuttaa niihin pohjautuen. Tässä tutkielmassa ajettavien suorituskykytestien tarkoituksena on selvittää, miten palvelininstanssit suoriutuvat eri käyttäjämäärien luomasta kuormasta. Näin ollen kuormitusta lisätään asteittain, kunnes palvelininstanssin rajat tulevat vastaan. Käyttäjämäärän nosto tapahtuu kasvattamalla jokaisen käyttöskenaarion samanaikaisten käyttäjien lukumäärää, kunnes testattava palvelininstanssi kaatuu tai ei pysty enää käsittelemään saapuvia pyyntöjä.

8.2.2 Tärkeimpien käyttöskenaarioiden määritys

Tässä suorituskykytestissä testataan viittä Drupalin perustoiminnallisuutta. Ensimmäinen käyttöskenaario on kirjautumattoman käyttäjän suorittama sivujen selaus (vain luku-operaatiot). Toinen skenaario on ensimmäistä vastaava, mutta siinä käyttäjät kirjautuvat järjestelmään. Kolmannessa skenaariossa kirjautumaton käyttäjä hakee tietoa sivuston hakutoiminnon avulla ja neljännessä vastaavan toiminnon suorittaa kirjautunut käyttäjä. Viidennessä skenaariossa kirjautunut käyttäjä luo kommentteja sivustolla oleviin kirjoituksiin.

8.2.3 Navigointipolkujen määritys käyttöskenaarioille

Kirjautumattoman käyttäjän suorittamassa sivujen selauksessa selattavat sivut valitaan satunnaisesti tietokannasta. Satunnainen järjestys on tallennettu ennalta tiedostoon, joten testi on toistettavissa samanlaisena. Kirjautuneen käyttäjän tapauksessa testi etenee samalla tavalla siten, että satunnainen

järjestys luetaan eri tiedostosta. Kolmannessa ja neljännessä käyttöskenaariossa sisältöä haetaan satunnaisilla tiedostoon generoiduilla hakusanoilla, joita esiintyy sivuston kirjoituksissa. Myös sellaisia hakusanoja, jotka eivät anna tuloksia, on generoitu hakusanojen joukkoon (n. 17% hakusanoista). Viimeisessä käyttöskenaariossa kommentoidaan sivuston kirjoituksia siten, että jälleen tiedostosta luetaan generoituja tekstejä, jotka lisätään tiettyjen satunnaisesti valittujen kirjoitusten kommenteiksi.

8.2.4 Yksilöllisen käyttäjädatan ja vaihteluiden määritys

Kuormitus pyritään jakamaan tasaisesti palvelimelle siten, että jokaiselle käyttöskenaariolle on määritetty tietyt viiveet aina kun uusi simuloitava käyttäjä lisätään testiin. Lisäksi kussakin käyttöskenaariossa suoritettavien interaktioiden välille on määritetty ajat, joiden välein pyyntöjä lähetetään. Pyyntöjen väliset ajat ovat satunnaisesti valittuja tietyltä aikaväliltä ja ne luetaan tiedostosta. Ne ovat siis toistettavissa samanlaisina eri testikierroksilla. Ajastusten avulla eri käyttäjien pyynnöt jakautuvat realistisemmin kuin tilanteessa, jossa käyttäjät suorittaisivat pyynnöt tasaisin väliajoin. Meier ym. (2007) ovat demonstroineet teoksessaan käyttäjien viiveiden mallinnusta kummassakin tapauksessa.

8.2.5 Käyttöskenaarioiden suhteellisen jakautumisen määritys

Käyttöskenaarioissa suoritettavien pyyntöjen lukumäärät jaetaan siten, että kahdessa ensimmäisessä skenaariossa pyyntöjä suoritetaan eniten ja lopuissa vähemmän. Hakutoiminto ja kommenttien kirjoittaminen ovat siis harvemmin tapahtuvia toimintoja järjestelmässä. Eräänä vertailukohtana tämän kaltaisille käyttöskenaarioille mainittakoon Blogbench-suorituskykytesti (Blogbench 2010), jolla testataan eräänlaisen blogijärjestelmän lukemista, kommentointia ja päivittämistä. Siinä on määritetty mm. 20 lukijaa kohden yksi kommentteja luova käyttäjä. Tässä yhteydessä on syytä huomata, että käyttöskenaarioiden

jakautuminen on sovelluskohtaista ja todellista jakautumista on mahdotonta määrittää tarkasti, mikäli järjestelmä ei ole käytössä tai sitä ei voida monitoroida. Tässä tutkielmassa neljän eri käyttöskenaarion suhteellista jakautumista havainnollistetaan taulukossa 7.

Taulukko 7 Käyttöskenaarioiden suhteellinen jakautuminen

Käyttöskenaario	**Suhteellinen jakautuminen (%)**
Selaus (kirjautumaton)	40
Selaus (kirjautunut)	40
Haku (kirjautumaton)	8
Haku (kirjautunut)	8
Kommentointi (kirjautunut)	4

Pyynnöt lähetetään satunnaisesti generoidun ajan mukaan kullakin määritetyllä aikavälillä. Ajat on generoitu tiedostoon, joten ajon aikana ei generointia tehdä. Tällä tavalla testi voidaan myös toistaa samanlaisena. Suhteellisessa jakautumisessa saattaa esiintyä pientä vaihtelua. Oleellista kuitenkin on, että sivuston selaus käsittää n. 80 %:a, haut n. 16 %:a ja kommentointi n. 4 %:a kaikista pyynnöistä.

8.2.6 Kuormamallin toteutuksen valmistelu

Kuormamallia kuvaava skripti kirjoitetaan ja ajetaan JMeter-kuormitustyökalulla. Jmx-päätteinen skriptitiedosto sisältää n. 70 elementtiä, joihin sisältyvät mm. HTTP-pyyntöjen määrittelyyn tarkoitetut elementit, ajastimet, erilaiset ohjaimet (controllers) ja kuuntelijat. Elementtejä käyttäen pyynnöt on mahdollista lähettää tietyssä järjestyksessä halutuin aikavälein. Kuuntelijoiden avulla tuloksia voidaan monitoroida. JMeter-ohjelmalla luotu elementtipuu esitellään liitteessä 5.

8.3 Yhteenveto

Tässä luvussa on esitelty suorituskykytestauksen testiympäristö ja testisuunnitelma. Luvun ensimmäisessä pääkohdassa testiympäristön yhteydessä on määritelty kunkin palveluntarjoajan testattavat palvelininstanssit, testattavan sovelluksen kokoonpano, testien ajopaikka ja testattavien laitteistojen maantieteellinen sijainti. Lisäksi kohdassa on esitelty testauslaitteisto, kuormitustyökalu ja muut testiympäristöön liittyvät asiat. Kultakin palveluntarjoajalta valittiin kolme erilaista palvelininstanssia testiä varten. Valitut palvelininstanssit ovat hinnoittelultaan edullisimmasta päästä, ja näin ollen suurimmat ja tehokkaimmat palvelininstanssit jätettiin testistä pois. Testit ajetaan Suomessa olevalla testilaitteistolla ja testattavat palvelininstanssit sekä niille asennetut Drupal-sovellukset sijaitsevat joko Irlannissa tai Yhdysvalloissa.

Luvun toisessa pääkohdassa on esitelty testisuunnitelma. Sen yhteydessä on määritelty käyttöskenaariot, joilla tarkoitetaan niitä toimintoja, joita simuloidut käyttäjät suorittavat testattavassa järjestelmässä. Käyttöskenaarioille on määritelty myös navigointipolut, vaihtelut ja suhteellinen jakautuminen. Näiden määrittelyiden tarkoituksena on luoda kuormamalli, joka jäljittelee mahdollisimman tarkasti oikeiden käyttäjien toimintaa. Kun käyttöskenaarioihin liittyvät asiat on määritelty, kuormamalli on valmis ja se voidaan toteuttaa siihen tarkoitetulla työkalulla.

9 TULOKSET

Testit ajettiin yhteensä yhdeksällä erilaisella palvelininstanssilla (kultakin palveluntarjoajalta kolme palvelininstanssia). Testit koostuivat eri toiminnoista ja niiden tavumääräiset koot olivat kaikkien toimintojen osalta suhteellisen pienet. Rackspacen neliytimisellä suorittimella varustetut palvelininstanssit suoriutuivat testistä suoritustehon (valmistuneiden pyyntöjen määrä sekuntia kohden) suhteen parhaiten. Sen sijaan parhaat vasteajat mitattiin Amazonin instansseilla. Taulukossa 8 esitellään ping-komennolla testattu verkon latenssi (kiertoaika, round-trip time) eri palvelinkeskuksiin.

Taulukko 8 Verkon latenssi palvelinkeskuksittain

Palveluntarjoaja ja sijainti	**Keskimääräinen kiertoaika (millisekuntia)**
Amazon EC2 (EU)	66
Amazon EC2 (Yhdysvallat itä)	121
Amazon EC2 (Yhdysvallat länsi)	188
Amazon EC2 (Aasia)	314
GoGrid (Yhdysvallat itä)	122
GoGrid (Yhdysvallat länsi)	189
Rackspace (Yhdysvallat Dallas/Chigago)	132

Taulukon 8 mukaisesti Amazonin Irlannin palvelinkeskuksella on testien ajopaikan sijainnin takia ylivoimaisesti pienin verkon latenssi. Sekä GoGridin että Amazonin yhdysvaltojen palvelinkeskuksiin mitatut latenssit ovat lähes samat. Rackspacen latenssi on Amazonin ja GoGridin Yhdysvaltain itäisiin palvelukeskuksiin nähden aavistuksen verran korkeampi, sillä palvelinkeskukset ovat kauempana itärannikosta. Lisäksi on huomioitava, että Rackspacen tapauksessa palvelinkeskusta ei ole mahdollista valita itse. Näin

ollen latenssia ei voitu myöskään mitata erikseen kumpaankin palvelinkeskukseen. Taulukossa 8 esitellyistä palveluntarjoajista ja palvelinkeskuksista testeissä oli mukana Amazonin Euroopan palvelinkeskus, GoGridin Yhdysvaltojen läntinen palvelinkeskus ja Rackspacen palvelinkeskukset. Verkon latenssin vaikutus on siis suurin GoGridin palvelininstansseilla.

Seuraavaksi tulosten esittelyssä on hyvä huomioida, että vasteajat, suoritusteho ja virheet kattavat kaikki testissä ajetut toiminnot. Ne ovat siis keskiarvoja eri toimintoihin käytetyistä suoritusajoista. Testissä vasteajat vaihtelivat toimintojen mukaan. Vasteaikojen vaihtelut voidaan havaita tuloksia esittävistä kuvaajista. Eri toimintoihin käytettyjä vasteaikoja esitellään tiettyjen palvelininstanssien osalta erikseen kohdassa 9.2.

9.1 Testin tulokset palveluntarjoajittain

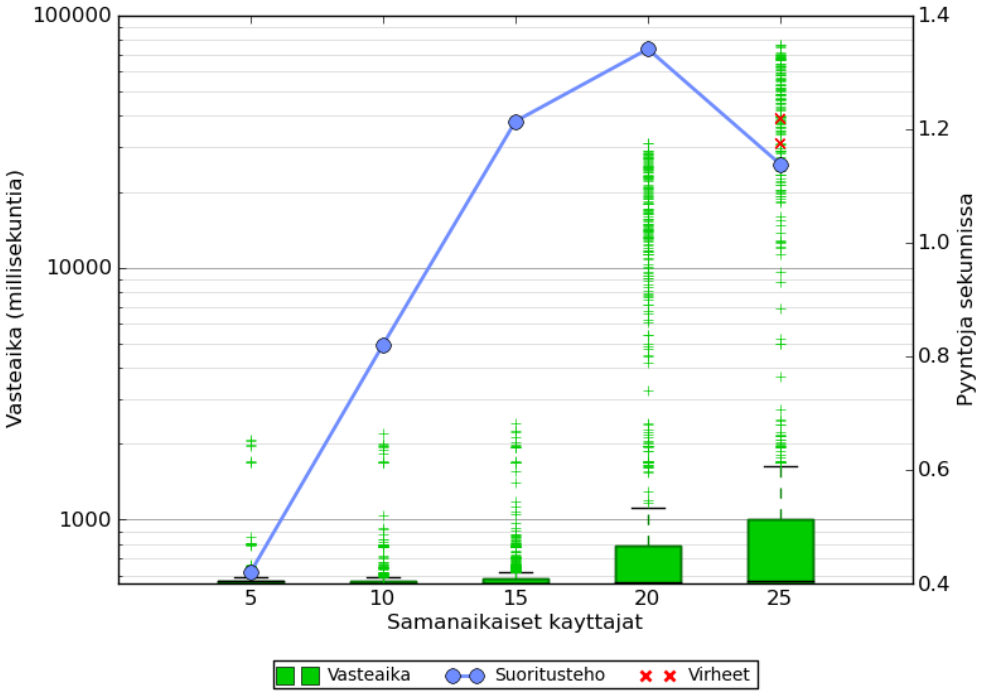
Testin tulokset esitellään seuraavaksi palveluntarjoajittain. Testattavat käyttäjämäärät vaihtelevat palvelininstanssien suorituskyvystä riippuen kymmenistä muutamaan sataan. Tulosten esittämisen apuna käytetään kuvaajaa (Metal Toad Media 2011), joka näyttää kunkin testattavan palvelininstanssin samanaikaisten käyttäjien lukumäärän, vastejat, suoritustehon ja virheet.

9.1.1 Amazon EC2

Testit ajettiin kolmella erilaisella Amazon EC2 -instanssilla. Sekä mikroinstanssi (Micro Instance) että pieni instanssi (Small Instance) osoittautuivat testissä suhteellisen tehottomiksi. Sen sijaan korkeammalla laskentateholla varustettu keskikokoinen instanssi (High-CPU Medium Instance) osoittautui selvästi suorituskykyisemmäksi. Pienimmän mikroinstanssin tulokset esitellään taulukossa 9 ja kuvassa 10.

Taulukko 9 Amazon EC2 mikroinstanssi

Käyttäjät	Vasteaikojen keskiarvo	Vasteaikojen mediaani	Vasteaikojen keskihajonta	Virheet	Suoritusteho
5	599	568	225	0.0 %	25.3/min
10	597	560	225	0.0 %	49.3/min
15	609	560	236	0.0 %	1.2/s
20	3466	563	6795	0.0 %	1.3/s
25	8256	573	17925	0.26 %	1.2/s



Kuva 10 Amazon EC2 mikroinstanssi

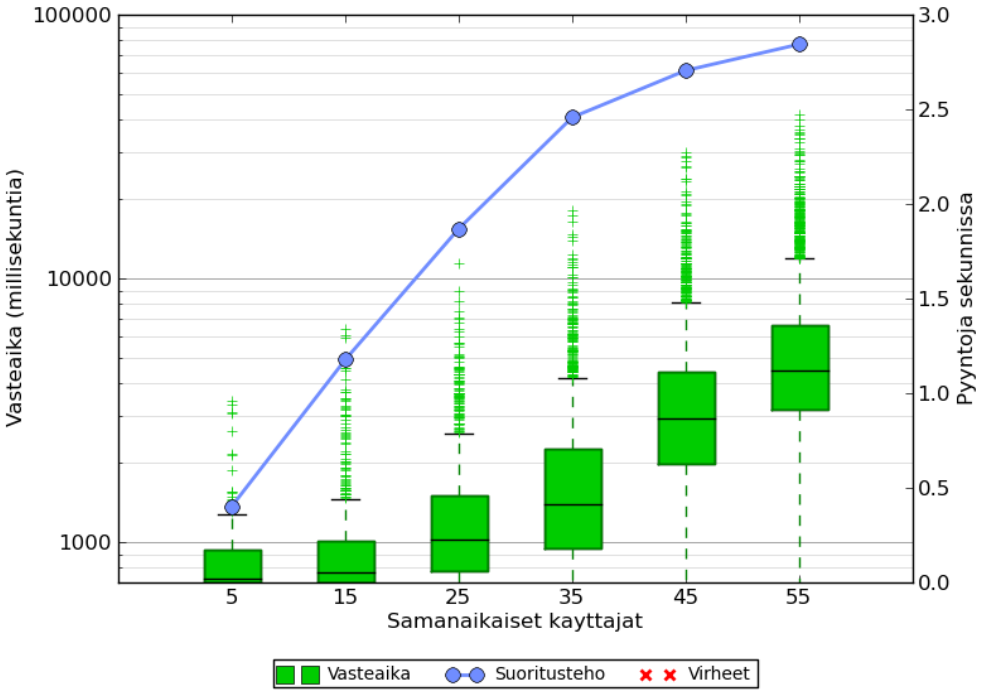
Amazon määrittelee mikroinstanssin laskentatehoksi enintään 2 laskentayksikköä. (Amazon.com 2010d). Tämän tiedon perusteella

laskentatehon voidaan olettaa vaihtelevan ja myös testit osoittivat, ettei suorituskyky ole tasainen ja vakaa. Kuvan 10 osoittamalla tavalla vasteaikojen hajonta 15 käyttäjään asti pysyy vielä kurissa, kun taas 20 käyttäjällä hajonta on jo aivan toista luokkaa. Taulukosta 9 nähdään, että vasteaikojen mediaaniarvot pysyvät lähes tarkalleen samoina kaikilla kuormituksilla, mutta keskihajonta kasvaa voimakkaasti. Tämä vahvistaa suorituskyvyn epätasaisuuden; suurin osa pyynnöistä suoritetaan suhteellisen alhaisilla vasteajoilla, kun taas osalla pyynnöistä vasteajat nousevat erittäin korkeiksi. Vielä 25 käyttäjällä suurin osa palvelupyynnöistä suoritetaan 600 millisekunnin sisällä, mutta vasteaikojen keskiarvo nousee silti suureksi, koska lopuilla pyynnöistä vasteajat ovat pääasiassa hyvin korkeita.

Amazon EC2:n pienen 1.7GB muistia sisältävän instanssin suorituskyky osoittautui mikroinstanssia vakaammaksi. Tämän tyyppisessä testissä suorituskyky jäi kuitenkin varsin vaatimattomaksi, sillä luvattu yhden ytimen ja yhden laskentayksikön prosessoriteho ei riitä käsittelemään suuria määriä pyyntöjä. Pienen instanssin tulokset esitellään taulukossa 10 ja kuvassa 11.

Taulukko 10 Amazon EC2 pieni instanssi

Käyttäjät	**Vasteaikojen keskiarvo**	**Vasteaikojen mediaani**	**Vasteaikojen keskihajonta**	**Virheet**	**Suoritusteho**
5	881	728	403	0.0 %	24.2/min
15	989	770	638	0.0 %	1.2/s
25	1367	1024	1013	0.0 %	1.9/s
35	1988	1388	1841	0.0 %	2.5/s
45	3920	2956	3354	0.0 %	2.7/s
55	6219	4472	5220	0.0 %	2.9/s



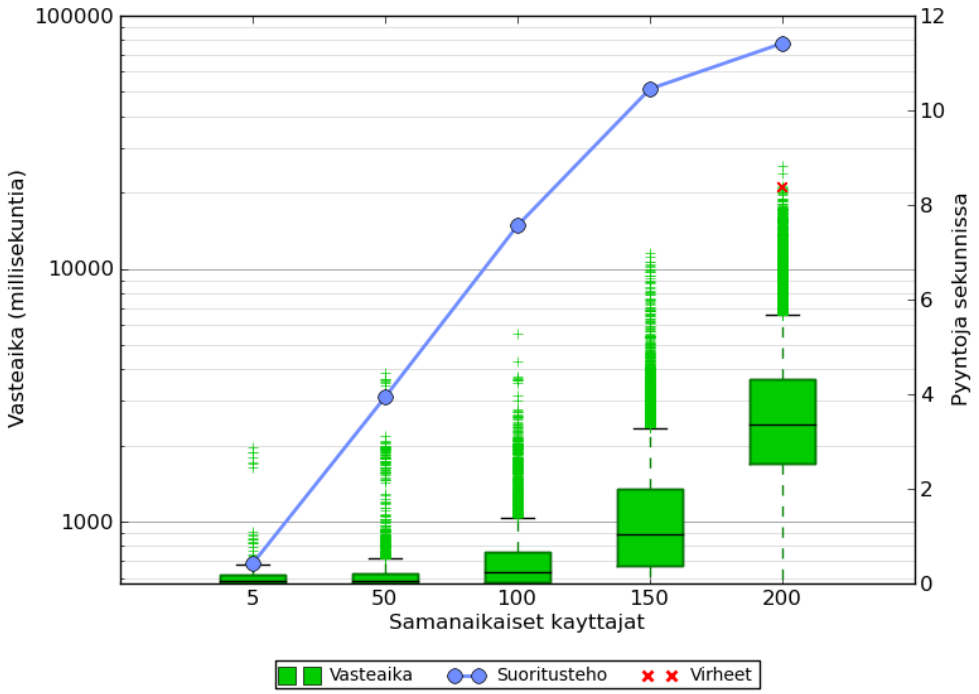
Kuva 11 Amazon EC2 pieni instanssi

Pientä instanssia ja mikroinstanssia vertaillessa havaitaan, että pienen instanssin suorituskyky on alhaisilla käyttäjämäärillä vasteajoissa mitattuna heikompi kuin mikroinstanssin. Korkeammilla käyttäjämäärillä pienen instanssin vasteajat kasvavat tasaisesti eikä mikroinstanssin testauksessa havaittuja heilahteluja tapahdu. Tosin kuormituksen kasvaessa riittävästi palvelin ”tukkeutuu” ja vasteajat nousevat jyrkemmin. Maksimaalinen suoritusteho pienellä instanssilla tässä testissä on noin kolme pyyntöä sekunnissa.

Keskikokoinen korkean laskentatehon 1.7GB:n keskusmuistilla varustettu palvelininstanssi (High-CPU Medium) osoittautui suoritustehon osalta selvästi edellä testattuja instansseja suorituskykyisemmäksi. Viisi laskentayksikköä käsittävä tuplaydinprosessori on tässä ratkaisevassa asemassa. Käyttäjämäärä on mahdollista nostaa noin neljä kertaa pientä instanssia korkeammaksi. Taulukossa 11 ja kuvassa 12 esitellään tulokset.

Taulukko 11 Amazon EC2 korkean laskentatehon instanssi (High-CPU Medium)

Käyttäjät	Vasteaikojen keskiarvo	Vasteaikojen mediaani	Vasteaikojen keskihajonta	Virheet	Suoritusteho
5	624	586	211	0.0 %	25.2/min
50	637	581	268	0.0 %	3.9/s
100	740	632	334	0.0 %	7.6/s
150	1213	888	977	0.0 %	10.5/s
200	3302	2409	2908	0.01 %	11.5/s



Kuva 12 Amazon EC2 korkean laskentatehon instanssi (High-CPU Medium)

Korkean laskentatehon instanssilla vasteaikojen keskiarvot pysyvät hyvin matalina n. 100 käyttäjään asti. Tämän jälkeen ne nousevat vähitellen siten, että maksimaalinen suoritusteho asettuu lähes 12 pyyntöön sekunnissa n. 200 käyttäjällä. 150 käyttäjällä keskimääräinen vasteaika on hieman yli sekunnin.

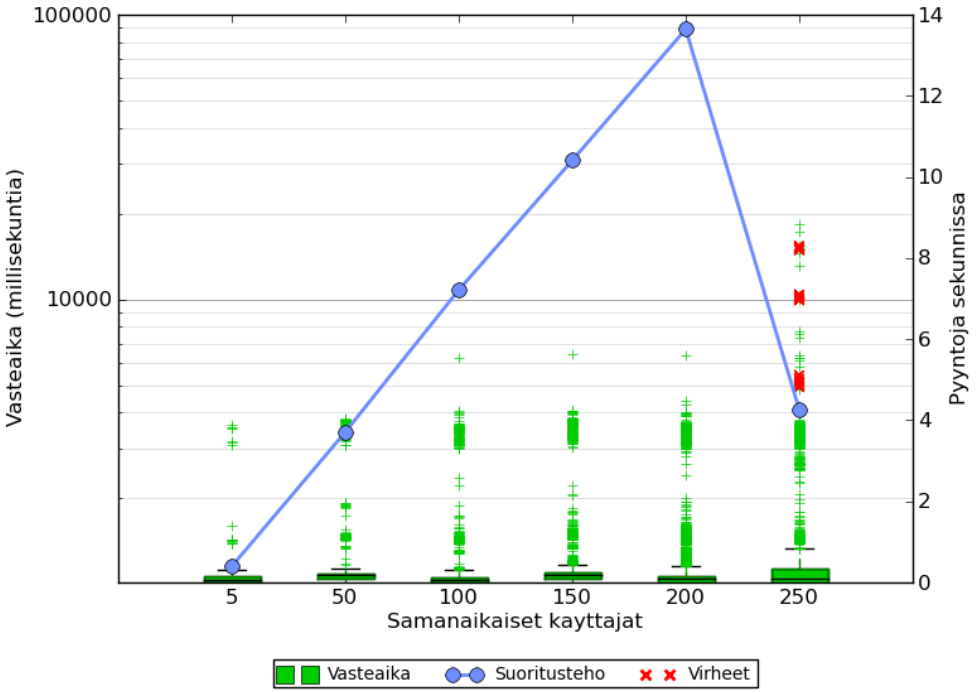
Tosin tällöin kuvan 12 osoittamalla tavalla osalla pyynnöistä vasteaika nousee jo kymmenen sekunnin tuntumaan ja vasteaikojen keskihajonta on huomattavasti korkeampi kuin edellisessä mittauspisteessä. Järjestelmän taivekapasiteetin voidaan sanoa asettuvan jonkin verran 150 käyttäjän kuorman alapuolelle, eli hieman alle 10 pyyntöön sekunnissa.

9.1.2 Rackspace

Rackspacen pilvipalvelimista testattavana oli 0.5GB:n, 1GB:n ja 2GB:n keskusmuisteilla varustetut instanssit. Jokainen näistä palvelininstansseista käyttää neliytimistä suoritinta. Kohdan 7.2.2 tietojen mukaisesti kullakin instanssilla on mahdollisuus hyödyntää kaikki saatavilla oleva prosessoriteho siten, että se on vähintään taulukossa 3 esitelty minimilaskentateho. Näin ollen tarkkaa instanssien käytössä olevaa laskentatehoa on mahdotonta sanoa. 0.5GB:n instanssin tulokset esitellään taulukossa 12 ja kuvassa 13.

Taulukko 12 Rackspace 0.5GB

Käyttäjät	**Vasteaikojen keskiarvo**	**Vasteaikojen mediaani**	**Vasteaikojen keskihajonta**	**Virheet**	**Suoritusteho**
5	1057	1038	435	0.0 %	23.9/min
50	1080	1073	455	0.0 %	3.7/s
100	1041	1032	441	0.0 %	7.2/s
150	1085	1076	455	0.0 %	10.4/s
200	1055	1040	429	0.0 %	13.7/s

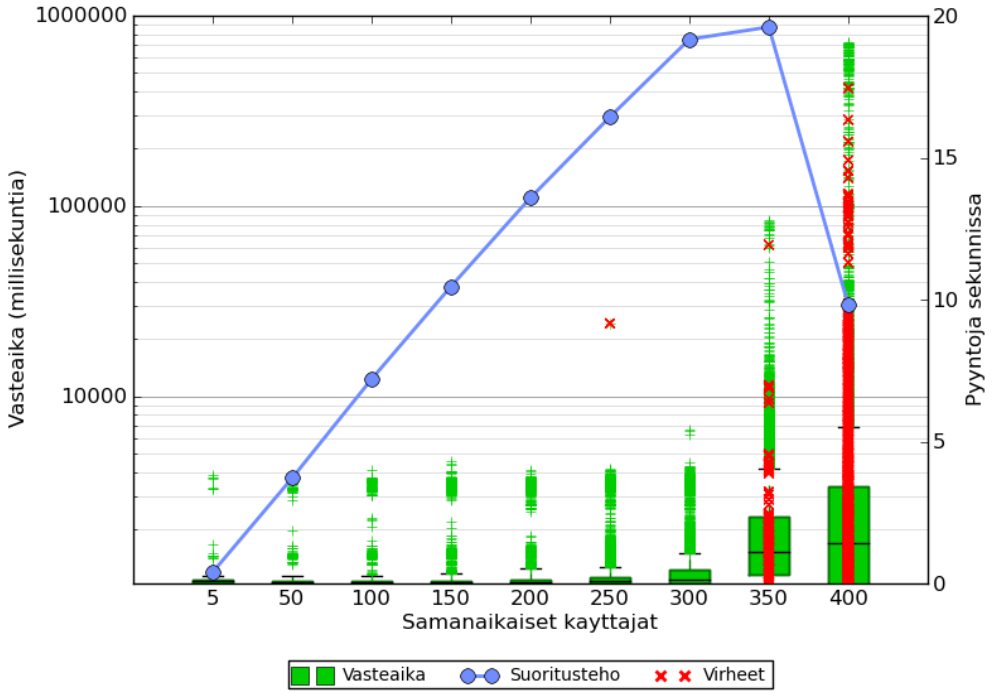


Kuva 13 Rackspace 0.5GB

Taulukosta 12 ja kuvasta 13 nähdään, että suorituskyky kasvaa erittäin tasaisesti 200 käyttäjään saakka. Nostettaessa käyttäjämäärää 250:een, virheiden määrä kasvaa voimakkaasti ja instanssi kaatuu. Korkeammilla muistimäärillä varustetut instanssit pysyvät vakaampana, joten kaatuminen viittaa muistin loppumiseen. Lisäksi suoritustehon kasvu ei hidastu missään vaiheessa. Tämä viittaa siihen, ettei prosessori ole järjestelmän pullonkaula. 1GB:n instanssilla maksimaalinen suoritusteho 300 käyttäjällä on n. 19 pyyntöä sekunnissa. Vasteaikojen keskiarvot pysyttelevät tällöin hieman alle 1200 millisekunnissa. Tästä eteenpäin virheiden (503 Service unavailable) määrä kasvaa voimakkaasti. 350 käyttäjällä palvelininstanssi ei pysty enää käsittelemään sille tulevien pyyntöjen määrää, ja lähes kaikki pyynnöt päättyvät virheeseen. Toisin kuin 0.5GB:n instanssi, 1GB:n instanssi ei kuitenkaan kaadu. 2GB:n instanssin suorituskyvyssä ei ole merkittävää eroa 1GB:n instanssiin nähden; maksimaalinen suoritusteho ja vasteajat ovat lähes samoja kuin 1GB:n instanssilla. Taulukossa 13 ja kuvassa 14 esitellään 2GB:n instanssin tulokset.

Taulukko 13 Rackspace 2GB

Käyttäjät	Vasteaikojen keskiarvo	Vasteaikojen mediaani	Vasteaikojen keskihajonta	Virheet	Suoritusteho
5	1090	1070	459	0.0 %	23.9/min
50	1064	1058	447	0.0 %	3.7/s
100	1063	1057	441	0.0 %	7.2/s
150	1067	1059	445	0.0 %	10.5/s
200	1076	1064	434	0.0 %	13.6/s
250	1115	1076	483	0.01 %	16.5/s
300	1174	1096	456	0.0 %	19.2/s
350	2210	1523	3121	1.23 %	19.9/s



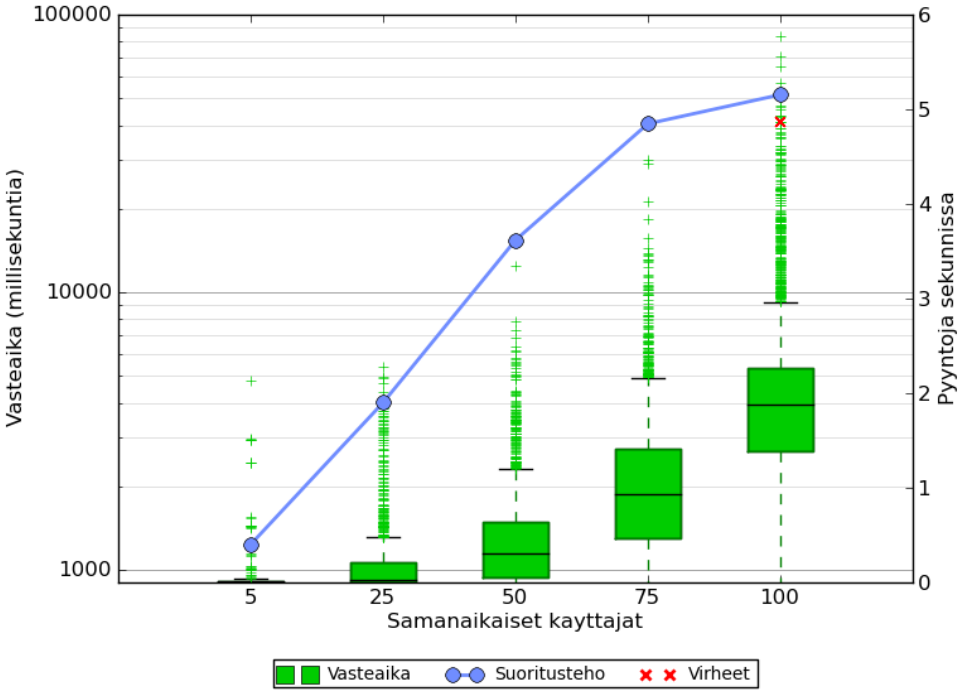
Kuva 14 Rackspace 2GB

9.1.3 GoGrid

GoGridilla testattiin Rackspacen tapaan 0.5GB:n, 1GB:n ja 2GB:n keskusmuisteilla varustetut palvelininstanssit. Suorituskyky tässä testissä vaihteli vain vähän 0.5GB:n ja 1GB:n instanssin välillä. Esimerkiksi suoritusteho lähellä maksimikuormitusta (100 käyttäjää) mitattiin lähes samaksi (molemmilla palvelininstansseilla n. 5 pyyntöä sekunnissa). Vasteajoissa ei esiintynyt merkittäviä eroja, mutta hieman yllättäen 0.5GB:n instanssin vasteajat olivat jopa aavistuksen verran pienempiä kuin 1GB:n instanssilla. Kohdan 7.2.3 mukaisesti 0.5GB:n instanssille luvataan 0.5-1 prosessoriytimen suoritusteho ja 1GB:n instanssille yksi ydin. Testien mukaan molempien instanssien suorituskyky on kuitenkin lähes sama. 0.5GB:n instanssin tulokset esitellään taulukossa 14 ja kuvassa 15.

Taulukko 14 GoGrid 0.5GB

Käyttäjät	**Vasteaikojen keskiarvo**	**Vasteaikojen mediaani**	**Vasteaikojen keskihajonta**	**Virheet**	**Suoritusteho**
5	961	906	416	0.0 %	24.2/min
25	1081	916	519	0.0 %	1.9/s
50	1345	1150	716	0.0 %	3.6/s
75	2259	1877	1764	0.0 %	4.9/s
100	4914	3940	5067	0.05 %	5.2/s



Kuva 15 GoGrid 0.5GB

Vasteajat kasvavat tasaisesti 50 käyttäjään asti, jonka jälkeen 75 käyttäjällä ollaan jo hyvin lähellä maksimaalista suoritustehoa. Tästä eteenpäin suoritusteho kasvaa erittäin hitaasti n. 100 käyttäjään asti. Tässä pisteessä saavutetaan järjestelmän maksimaalinen suoritusteho, joka on hieman yli viisi pyyntöä sekunnissa. 2GB:n instanssi suoriutui testistä selvästi 0.5GB:n ja 1GB:n instansseja paremmin kahden prosessoriytimensä ansiosta. Tulokset esitellään taulukossa 15 ja kuvassa 16.

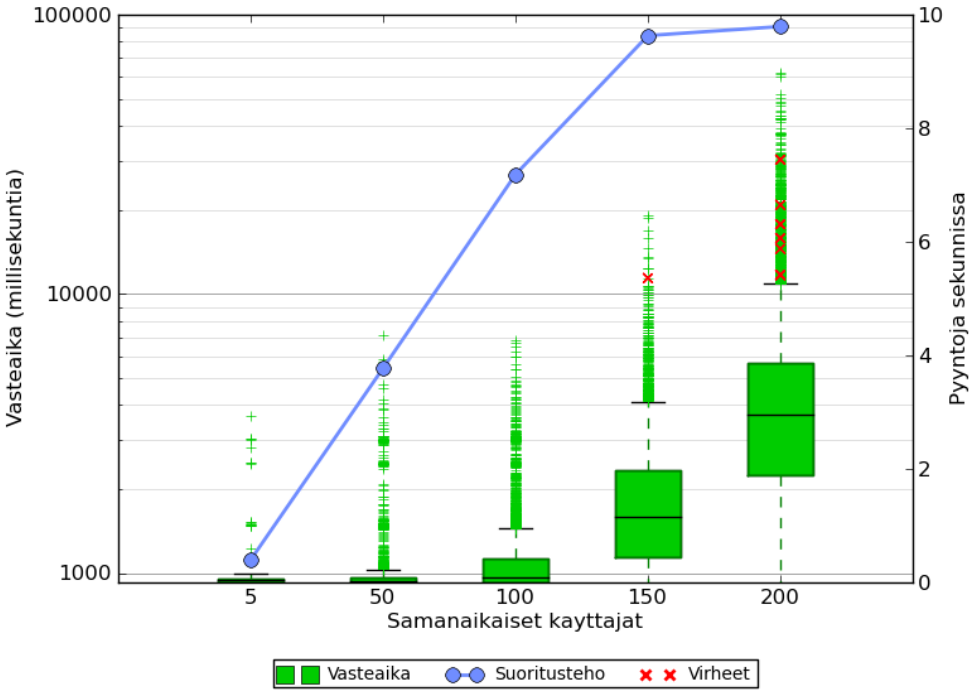
Taulukko 15 GoGrid 2GB

Käyttäjät	**Vasteaikojen keskiarvo**	**Vasteaikojen mediaani**	**Vasteaikojen keskihajonta**	**Virheet**	**Suoritusteho**
5	997	943	376	0.0 %	24.0/min

(jatkuu)

Taulukko 15. (jatkuu)

50	1004	937	438	0.0 %	3.8/s
100	1117	968	506	0.0 %	7.2/s
150	1916	1593	1235	0.01 %	9.7/s
200	5023	3701	5131	0.09 %	9.8/s



Kuva 16 GoGrid 2GB

2GB:n instanssin vasteajoissa ei tapahdu 100 käyttäjään asti merkittäviä muutoksia. 150 käyttäjään nostettaessa sen sijaan suoritustehon kasvu hidastuu tuntuvasti ja vasteajat kasvavat nopeammin. Järjestelmän maksimaalinen suoritusteho on n. 10 pyyntöä sekunnissa.

9.2 Eri toimintojen vasteajat

Tässä kohdassa esitellään eri toimintoihin käytettyjä vasteaikoja kunkin palveluntarjoajan suurimmalla testissä mukana olleella palvelininstanssilla. Testi ajettiin Amazonin Irlannissa sijaitsevan palvelinkeskuksen lisäksi myös Yhdysvaltojen itäisessä (Virginia) palvelinkeskuksessa. Tällä tavalla maantieteellisen sijainnin vaikutusta vasteaikoihin voidaan arvioida paremmin, sillä palvelinkeskusten tarjoamat palvelininstanssit ovat samanlaisia. Virginian palvelinkeskuksen testi ajettiin muista testeistä erillään toukokuussa 2011. Taulukon 8 mukaisesti sinne mitattiin lähes tuplasti korkeampi verkon latenssi kuin Irlannissa sijaitsevaan palvelinkeskukseen.

Testattavana oli sovelluksen viisi eri toimintoa, jotka olivat kirjautumattoman/kirjautuneen käyttäjän suorittama sivujen selaus, kirjautumattoman/kirjautuneen käyttäjän sisällön haku ja kirjautuneen käyttäjän suorittama sisällön kommentointi. Vasteajat ilmoitetaan matalalla kuormitusasteella (50 käyttäjää), jolloin käyttäjämäärä ei vielä juurikaan vaikuta niihin. Keskimääräiset koot on merkitty taulukoihin. Vasteajat ilmoitetaan millisekunteina. Taulukossa 16 esitellään kirjautumattoman käyttäjän ja taulukossa 17 kirjautuneen käyttäjän tulokset.

Taulukko 16 Toimintojen vasteajat (kirjautumaton)

Palvelininstanssi	**Satunnainen sivu (17kt)**	**Sisällön haku (7kt)**
Amazon EC2 High-CPU Medium (Irlanti)	596	596
Amazon EC2 High-CPU Medium (Yhdysvallat)	1059	750
Rackspace 2GB	1056	620
GoGrid 2GB	966	702

Taulukko 17 Toimintojen vasteajat (kirjautunut)

Palvelininstanssi	**Satunnainen sivu (16kt)**	**Sisällön haku (6kt)**	**Kommentointi (17kt)**	**Etusivu (62kt)**
Amazon EC2 High-CPU Medium (Irlanti)	592	624	904	1625
Amazon EC2 High-CPU Medium (Yhdysvallat)	1045	738	1467	3306
Rackspace 2GB	1050	622	1445	3245
GoGrid 2GB	951	695	1535	2575

Satunnaisten sivujen ja etusivun latauksessa Amazonin Irlannissa sijaitseva palvelininstanssi osoittautui selvästi muita palvelininstansseja suorituskykyisemmäksi. Sen sijaan sisällön haussa tulokset olivat tasaisemmat. Tuloksista on havaittavissa, että pienimpien toimintojen tapauksessa maantieteellisen sijainnin vaikutus on vähäisin. Suuremmissa toiminnoissa sijainnin merkitys korostuu ja vasteaikojen erot näkyvät selvemmin. Suurempien datamäärien tapauksessa verkon latenssin lisäksi sen siirtokapasiteetti vaikuttaa vasteaikoihin. Taulukon 17 mukaisesti etusivun lataus oli selvästi hitain testissä suoritetuista toiminnoista. Tämä on selitettävissä sillä, että etusivun koko on muita sivuja suurempi ja sille on asetettu enemmän sisältöä (linkit teksteihin ja kuvia), jolloin tietokannasta joudutaan hakemaan useampi kirjoitus ja palvelimelta useita kuvia. Datamäärältäänhän etusivukin on suhteellisen pieni. Tässä yhteydessä on myös hyvä huomioida, että tietokantakyselyitä ei ole optimoitu mitenkään. Kirjautuneen käyttäjän vasteajoilla ei ole satunnaisten sivujen ja sisällön haun suhteen juurikaan eroa kirjautumattoman käyttäjän vasteaikoihin. Tuloksissa

on huomioitava, että sivujen keskimääräiset koot kirjautuneella käyttäjällä ovat aavistuksen verran pienempiä. Tämä on selitettävissä sillä, että kirjautuneelle käyttäjälle esitetään hieman eri tiedot ja elementit kuin kirjautumattomalle käyttäjälle. Esimerkiksi sivuston sivupalkissa olevaa kirjautumislomaketta ei ole kirjautuneella käyttäjällä.

9.3 Optimoinnista

Optimointia hyödyntämällä voidaan saavuttaa moninkertaisesti paremmat tulokset kuin tässä testissä saavutettiin. Testistä haluttiin kuitenkin sulkea pois optimointitekniikat, kuten PHP-kiihdyttimet (PHP accelerators), erilaisten välimuistien käyttö ja tietokannan optimointi. LAMP-ympäristön optimoinnista löytyy tutkimusaineistoa usealta eri www-sivustolta. Seuraavaksi esitellään lyhyesti optimointitekniikoita ja niiden käytöllä saavutettuja tuloksia.

Drupalin sisäisen välimuistitoiminnon avulla palvelimen kuormaa saadaan kevennettyä ja nopeutettua sivujen luomiseen kuluvaa aikaa. Välimuisti toimii siten, että Drupal tallentaa ladatun sivun HTML-koodin suoraan tietokantaan. Kun sivua ladataan uudestaan, se haetaan tietokannasta eikä sitä generoida uudelleen alusta. Tällaisen menettelyn tuloksena sadat tietokantakyselyt korvataan yhdellä kyselyllä ja web-palvelimen kuormaa saadaan kevennettyä merkittävästi. (Drupal 2011b).

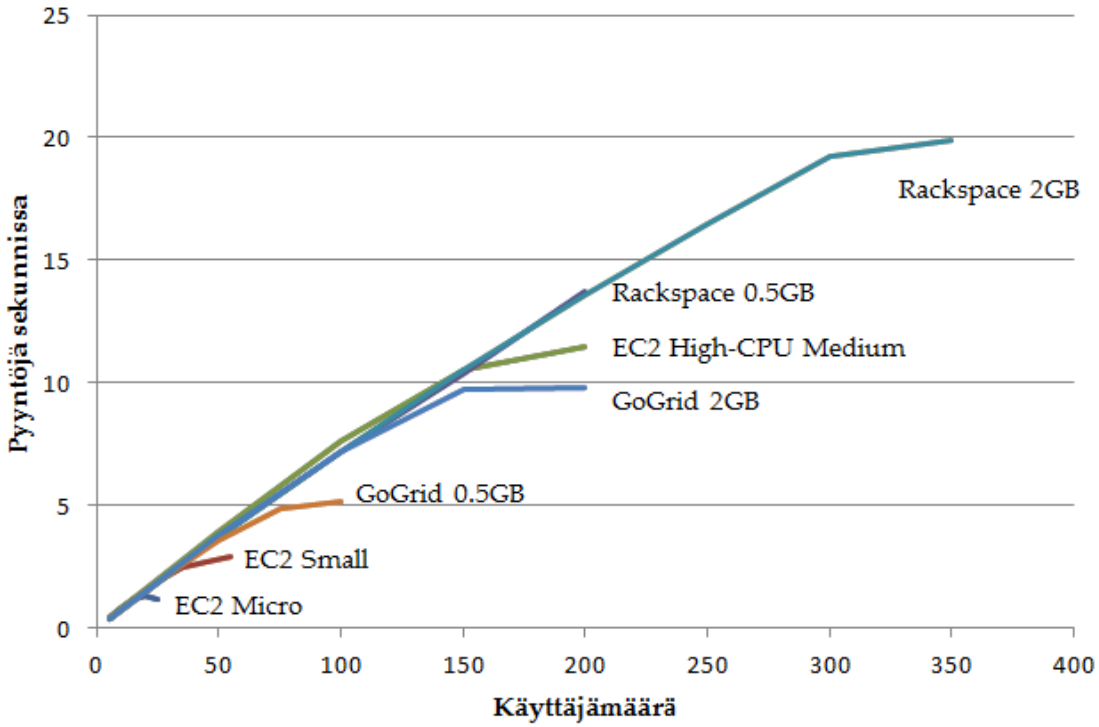
PHP-kiihdyttimet (esim. APC, eAccelerator, XCache) ovat hyvin suosittu keino nostaa PHP-sovellusten suorituskykyä. Niiden käyttöönotto on helppoa ja toiminta perustuu siihen, että PHP-skriptit tallennetaan välimuistiin valmiiksi käännettyinä. Tällä tavalla kääntämisestä aiheutuva kuorma saadaan lähes kokonaan poistettua. PHP-kiihdyttimillä palvelinten kuormitusta saadaan siis kevennettyä ja PHP-koodi voidaan suorittaa jopa kymmenen kertaa nopeammin (EAccelerator 2011). Esimerkiksi 2bits.com-sivuston (2bits 2011) suorittamissa testeissä suoritusteho saatiin noin kolme kertaa suuremmaksi ja

vasteajat kolmasosaan verrattuna tilanteeseen, jossa PHP-kiihdyttimiä ei käytetä.

Eräs suosittu suorituskyvyn nopeuttamisen keino on tallentaa dataa muistiin, josta se saadaan nopeasti haettua. Memcached on suosittu dynaamisten web-sovellusten nopeuttamiseen tarkoitettu järjestelmä. Sen tarkoituksena on vähentää tietokantaan kohdistuvaa kuormitusta tallentamalla muistiin dataa mm. tietokantakyselyiden tuloksista. (Memcached 2011). Memcachedin ja muiden optimointitekniikoiden käyttöä Drupalin kanssa on testattu mm. coldfrontlabs.ca-sivustolla (Coldfront Labs 2011). Testeissä saavutettiin huikeat tulokset Drupalin sisäisen välimuistitoiminnon, PHP-kiihdyttimen ja Memcached-järjestelmän yhteistoiminnalla. Suoritusteho kasvoi noin seitsenkertaiseksi ja vasteajat laskivat murto-osaan verrattuna tilanteeseen, jossa mitään optimointitekniikoita ei käytetä.

9.4 Yhteenveto

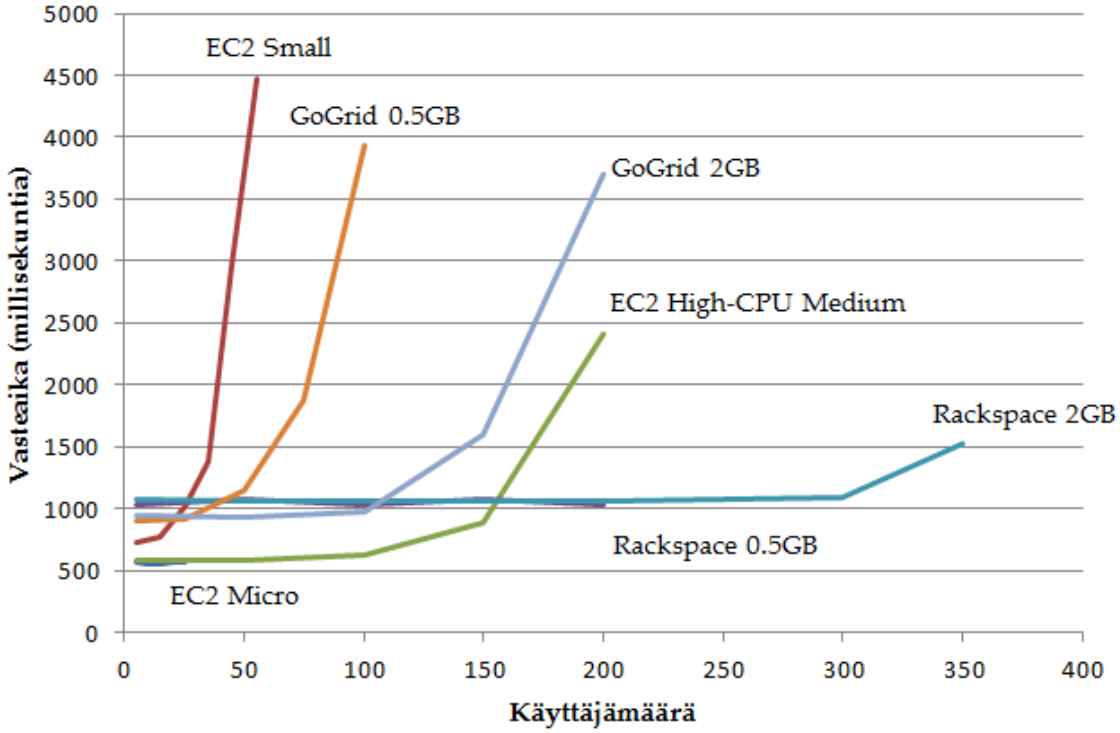
Tulokset osoittavat, että Rackspacen palvelininstanssit ovat tässä testissä muiden palveluntarjoajien instansseja suorituskykyisempiä, kun kysymys on suoritustehosta. Kuvassa 17 esitellään vertailuyhteenveto testeissä mitatuista suoritustehoista.



Kuva 17 Suoritusteho palvelininstansseittain

Suoritusteho on testissä vahvasti kytköksissä prosessoritehoon (ja prosessoriydinten lukumäärään) ja näin ollen Rackspace suoriutui testistä tehokkaiden neliytimisten prosessorien ansiosta muita palveluntarjoajia paremmin. Amazonin ja GoGridin tuplaydinprosessorilla varustetut instanssit osoittautuivat myös suhteellisen tehokkaiksi. Kun käytössä on yhden palvelimen kokoonpano eli web-palvelin ja tietokantapalvelin toimivat samalla tietokoneella, moniydinprosessorilla varustetuissa tietokoneissa prosessoriaika pystytään jakamaan paremmin näiden kesken. (Vandyk 2008, 542). Testit osoittivat, että myös muistin määrällä on merkitystä, kun kuormitus on riittävän suuri. Rackspacen pienimmällä muistilla (0.5GB) varustetulla palvelininstanssilla suoritusteho ei laskenut missään vaiheessa, mutta instanssi kuitenkin kaatui yllättäen, kun kuormitusta nostettiin tarpeeksi.

Vasteajoissa mitattuna Amazonin instanssit osoittautuivat tiettyihin käyttäjämääriin asti testin suorituskykyisimmiksi instansseiksi. Kuvassa 18 esitellään vertailuyhteenveto vasteaikojen mediaaniarvoista.



Kuva 18 Vasteaikojen mediaaniarvot palvelininstansseittain

Vasteaikoihin vaikuttaa keskeisesti palvelinkeskuksen maantieteellinen sijainti. Amazonin Irlannin palvelinkeskukselle mitattiin noin tuplasti pienempi verkon latenssi kuin Rackspacen palvelinkeskukselle. GoGridin testissä olleilla palvelininstansseilla latenssi oli vielä selvästi Rackspacea korkeampi. Tästä huolimatta GoGridin vasteajat olivat tiettyyn käyttäjämäärään asti keskimäärin Rackspacea pienempiä. Kuvan 18 osoittamalla tavalla pienempitehoisilla palvelininstansseilla vasteajat nousevat korkeiksi jo suhteellisen alhaisilla käyttäjämäärillä.

Amazonin palvelinstanssien kesken suorituskyvyssä on selvät erot, sillä instanssit ovat ominaisuuksiltaan varsin erilaisia. Sen sijaan Rackspacen ja GoGridin tapauksissa tilanne on toisenlainen. Näillä palveluntarjoajilla

palvelinstanssien luokittelu (kohdat 7.2 ja 8.1) tapahtuu kuusi- tai seitsenportaisen asteikon mukaan, jonka perusteella resurssit jaetaan palvelininstansseille. Esimerkiksi Rackspacen tapauksessa käytössä on palvelin, josta saatavien virtuaalisten palvelininstanssien ominaisuudet määräytyvät instanssityypin mukaan siten, että aina seuraavalle portaalle siirryttäessä hinta, vähimmäislaskentateho, muisti ja tallennuskapasiteetti tuplaantuvat. Testin tulokset kuitenkin osoittavat, että laskentatehon osalta suorituskyvyssä instanssityyppien välillä ei esiinny merkittäviä eroja. Mikäli siis halutaan puhdasta laskentatehoa, näyttäisivät pienemmät instanssit olevan selvästi suurempia instansseja kustannustehokkaampi ratkaisu. Tästä löytyy viitteitä myös muista tutkimuksista (CloudHarmony 2010; Li ym. 2010; The Bitsource 2010). Vastaavasti GoGridin tapauksessa laskentatehossa ei esiintynyt ensimmäisen ja toisen instanssityypin (0.5GB ja 1GB) osalta merkittäviä eroja, vaikka mm. hinta on jälkimmäisellä instanssilla tuplasti korkeampi. GoGridin suuremmilla palvelininstansseilla (2GB ja suuremmat) tilanne on kuitenkin erilainen kuin Rackspacessa, sillä myös prosessorin suorituskyky kasvaa hinnan ja muiden ominaisuuksien ohella.

Vasteaikojen toimintokohtainen tarkastelu osoitti, että datamäärältään pienemmissä toiminnoissa verkon latenssin vaikutus on vähäisempi. Esimerkiksi pienimmän toiminnon tapauksessa vasteaikojen ero nopeimman ja hitaimman instanssin välillä oli n. 150 millisekuntia, kun taas suurimmassa toiminnossa se oli yli 1.5 sekuntia. Lisäksi toimintokohtainen tarkastelu osoitti, että toimintojen suoritusajoissa oli mielenkiintoisia eroja palveluntarjoajien välillä. Esimerkiksi GoGridin suurin testattava instanssi osoittautui korkeammasta verkon latenssista huolimatta vasteajoissa mitattuna keskimäärin Rackspacen suurinta instanssia suorituskykyisemmäksi, sillä se suoritti yleisimmän toiminnon (satunnaisten sivujen lataus) Rackspacea nopeammin. Rackspace sen sijaan suoritti sisällön haun ja kommentoinnin GoGridiä pienemmillä vasteajoilla.

Testattavana olleen sovelluksen suorituskyky olisi saatu huomattavasti paremmaksi erilaisia optimointitekniikoita hyödyntämällä. Tutkielman laajuus huomioon ottaen niiden vaikutusta ei raportoitu (vaikka joitakin yksittäisiä testejä tehtiinkin). Optimointitekniikoiden avulla mm. tietokannan kuormitusta saadaan kevennettyä tallentamalla dataa muistiin. Eri tutkimukset osoittavat, että optimointitekniikoiden avulla suoritusteho voidaan saada moninkertaiseksi ja vasteajat huomattavasti pienemmiksi verrattuna tilanteeseen, jossa mitään optimointitekniikoita ei käytetä.

10 YHTEENVETO

Pilvilaskenta on kasvattanut räjähdysmäisesti suosiotaan, ja yhä useampi yritys siirtää IT-toimintoja pilvipalveluntarjoajien jättiläismäisiin palvelinkeskuksiin. Menettelyllä pyritään vähentämään oman IT-infrastruktuurin rakentamiseen kuluvaa aikaa ja kustannuksia. Kun alkuinvestointeja ei tarvita, vaan maksaminen tapahtuu käyttöperusteisesti, on pilvilaskenta houkutteleva vaihtoehto esimerkiksi aloitteleville yrityksille. Sen lisäksi, että IT-investoinnit voidaan vaihtaa liiketoiminnan operatiivisiksi kuluiksi, on ”loputtomasti” skaalautuva arkkitehtuuri kiinnostava ratkaisu moniin web-pohjaisiin järjestelmiin. Pilvilaskenta on varsin kiinnostava vaihtoehto esimerkiksi sovelluksissa, joiden käyttöasteet vaihtelevat paljon. Tällaisissa tapauksissa resursseja voidaan lisätä korkeammille kuormitustasoille ja vastaavasti vähentää, kun kysyntäpiikit menevät ohi. Toisaalta taas tapauksissa, joissa kuormitustasot pysyvät tasaisina, pilvet eivät välttämättä tarjoa optimaalista ratkaisua.

Suorituskyky on eräs pilvilaskennan keskeisistä kysymyksistä, johon liittyy mahdollisuuksia, mutta myös ongelmia ja epäilyksiä. Tässä tutkielmassa selvitettiin kirjallisuuteen perustuen, mitä haasteita pilven suorituskykyyn liittyy. Tutkimus osoitti, että keskeisiä asioita ovat virtualisointiin ja jaettuun arkkitehtuuriin liittyvät tekijät (mm. suorituskyvyn ennustamattomuus), saavutettavuus, tiedonsiirron ja skaalattavuuden haasteet, verkon latenssi sekä alustan suorituskyky. Virtualisoiduissa ympäristöissä levynopeuden todettiin olevan tärkein suorituskyvyn ongelmatekijä. Eri tutkimuksissa tehtyjen testien mukaan levynopeudessa esiintyy huomattavia vaihteluita, kun taas esimerkiksi virtualisoinnin vaikutus prosessoritehoon osoittautui suhteellisen vähäiseksi. Lisäksi useiden asiakkaiden kesken jaettavat resurssit osoittautuivat tekijäksi, joka lisää suorituskyvyn ennustamattomuutta pilvilaskennassa.

Palveluiden saavutettavuuden osalta tutkimukset osoittivat, etteivät pilvipalveluntarjoajat pysty nykyisellään takaamaan erittäin korkeaa saavutettavuutta. Lisäksi on riskinä, että palveluissa esiintyy pidempiäkin käyttökatkoja. Pilven sisäisen verkkoteknologian kehittymättömyys todettiin ongelmaksi, joka koskee erityisesti pilvessä tapahtuvaa suurteholaskentaa. Palvelininstanssien välinen kommunikointi ei ole tieteelliseen laskentaan vielä riittävän nopeaa (joitakin poikkeuksia lukuun ottamatta) ja pilvet onkin suunniteltu lähinnä liiketoiminnan tarpeisiin. Edellä esiteltyjen asioiden lisäksi kaksisuuntaiseen skaalattavuuteen liittyy haasteita (mm. palvelininstanssien käynnistymisajat) ja eräänä keskeisenä ongelmana on verkon latenssi, jonka vaikutus riippuu palvelinkeskusten maantieteellisestä sijainnista. Sijainnin merkitys korostuu etenkin suurempia tietomääriä siirrettäessä.

Alustan suorituskyky on tärkeä tekijä pilvipalvelun valinnassa. Palveluntarjoajilla on hyvin erilaiset alustojen ominaisuudet ja niiden vertailu on haasteellista. Lisäksi joitakin ominaisuuksia ei ilmoiteta alustojen tiedoissa lainkaan tai ne on ilmoitettu epämääräisesti (mm. vaihteluväli prosessoriteholle). Asiakkaan kannalta on siis vaikeaa valita ihanteellisin ratkaisu tiettyä sovellusta varten. Paras keino valinnan tekemiseen onkin ajaa sovellus eri palveluntarjoajien alustoilla. Tässä tutkielmassa testattiin Drupal-sisällönhallintasovelluksen suorituskykyä eri alustoilla. Sovellusta kuormitettiin eri määrillä simuloituja käyttäjiä, jotka suorittivat sovelluksen toimintoja. Testit osoittivat, että vasteaikojen suhteen Amazonin Irlannissa sijaitsevat palvelininstanssit ovat kokonaisuudessaan tiettyyn käyttäjien lukumäärään asti selvästi muita testattavana olleita palvelininstansseja suorituskykyisempiä. Sen sijaan tulosten toimintokohtainen tarkastelu (palveluntarjoajien suurimmat palvelininstanssit) osoitti, että datamäärältään pienimpien toimintojen suhteen vasteaikojen erot palveluntarjoajien välillä olivat pienet. Suuremmissa toiminnoissa vasteaikojen erot korostuivat. Tulosta voidaan selittää palvelinkeskusten maantieteellisellä sijainnilla.

Suoritustehon suhteen Rackspacen palvelininstanssit osoittautuivat selvästi testin suorituskykyisimmiksi instansseiksi. Myös Amazonin ja GoGridin suurimmat testissä olleet palvelininstanssit osoittautuivat suhteellisen suorituskykyisiksi. Yhden palvelimen kokoonpanolla prosessoriteho ja prosessoriydinten lukumäärä on tämän mittarin kohdalla ratkaisevassa asemassa. Saman palveluntarjoajan palvelininstanssien kesken suoritusteho ja vasteajat vaihtelivat joissakin tapauksissa vain vähän. Esimerkiksi Rackspacen tapauksessa 1GB:n ja 2GB:n muistilla varustetut palvelininstanssit suoriutuivat testistä käytännössä yhtä hyvin. Lähes samanlaiset tulokset mitattiin myös GoGridin kahdelle pienimmälle palvelininstanssille. Kumpienkin palveluntarjoajien tapauksessa hinta on suuremmalla palvelininstanssilla tuplasti kalliimpi, joten kustannustehokkuuksissa voi esiintyä suuriakin eroja.

Kokonaisuudessaan testien tulokset olivat melko vaatimattomat. Tämä johtuu osaltaan siitä, että kaikki optimointekniikat suljettiin testien ulkopuolelle. Eri lähteissä esitellyt tutkimukset osoittavat, että optimointitekniikoiden avulla suorituskyky voidaan saada moninkertaisesti paremmaksi. Optimointitekniikoiden lisäksi useamman palvelimen kokoonpanolla sovelluksen skaalattavuutta voitaisiin parantaa tuntuvasti. Mielenkiintoinen jatkotutkimuskohde onkin useamman palvelimen kokoonpanolla suoritettavat testit ja mahdollisesti optimointitekniikoiden hyödyntäminen näissä testeissä. Testit voitaisiin suorittaa aluksi erillisellä tietokantapalvelimella, jonka jälkeen kokoonpanoon lisättäisiin kuormantasaaja ja useampi web-palvelin. Samalla palvelininstanssien lisäämisen ja poistamisen nopeutta kokoonpanosta voitaisiin testata palveluntarjoajittain. Toinen mahdollinen jatkotutkimuskohde on pitkäkestoiset testit, joilla testataan eri palveluntarjoajien alustojen suorituskykyä ja sen vaihteluita pidemmällä aikavälillä.

LÄHDELUETTELO

2bits 2011. Benchmarking Drupal with PHP op-code Caches: APC, eAccelerator and XCache Compared [online]. [viitattu 25.1.2011]. Saatavilla www-osoitteessa <http://2bits.com/articles/benchmarking-drupal-with-php-op-code-caches-apc-eaccelerator-and-xcache-compared.html>.

Amazon.com 2010a. Amazon Elastic Compute Cloud (Amazon EC2) [online]. Amazon.com Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://aws.amazon.com/ec2/>.

Amazon.com 2010b. Amazon Machine Images (AMIs) [online]. Amazon.com Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://aws.amazon.com/amis>.

Amazon.com 2010c. Auto Scaling [online]. Amazon.com Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://aws.amazon.com/autoscaling/>.

Amazon.com 2010d. Amazon EC2 Instance Types [online]. Amazon.com Corporation [viitattu 2.11.2010]. Saatavilla www-osoitteessa <http://aws.amazon.com/ec2/instance-types/>.

Amazon.com 2011. Amazon EC2 Service Level Agreement [online]. Amazon.com Corporation [viitattu 2.2.2011]. Saatavilla www-osoitteessa <http://aws.amazon.com/ec2-sla/>.

Apache Software Foundation 2010. Apache JMeter [online]. Apache Software Foundation [viitattu 28.10.2010]. Saatavilla www-osoitteessa <http://jakarta.apache.org/jmeter/>.

Armbrust M., Fox A., Griffit R., Joseph A.D., Katz R.H., Konwinski A., Lee G., Patterson D.A., Rabkin A., Stoica I. & Zaharia M. 2009. Above the Clouds:

A Berkeley View of Cloud Computing. University of California at Berkeley, Electrical Engineering and Computer Sciences. Technical Report UCB/EECS-2009-28.

Badger L. & Grance T. 2010. Standards Acceleration to Jumpstart Adoption of Cloud Computing (SAJACC). Teoksessa Cloud Computing Forum & Workshop, Washington, May 20. National Institute of Standards and Technology (NIST), USA, 1-40.

Barker S. & Shenoy P. 2010. Empirical Evaluation of Latency-sensitive Application Performance in the Cloud. Teoksessa W. Feng & M-P. Ketan (toim.) Proceedings of the first annual ACM SIGMM conference on Multimedia systems (MMSys '10), New York, USA, ACM, 35-46.

Bennett S., Bhuller M. & Covington R. 2009. Architectural Strategies for Cloud Computing. Oracle White Paper julkaisussa Enterprise Architecture [online], [viitattu 5.6.2010]. Saatavilla www-muodossa <http://www.oracle.com/technology/architect/entarch >.

Blogbench 2010. Blogbench - About [online]. [viitattu 5.11.2010]. Saatavilla www-osoitteessa <http://www.pureftpd.org/project/blogbench>.

Briscoe G. & Marinos A. 2009. Digital Ecosystems in the Clouds: Towards Community Cloud Computing. Teoksessa O. Kaynak & M. Mohania (toim.) 3rd IEEE International Conference on Digital Ecosystems and Technologies, 1-3 June, IEEE DEST, 103-108.

Butcher M. 2008. Learning Drupal 6 Module Development. Birmingham: Packt Publishing.

Buyya R., Yeo C. S., Venugopal S., Broberg J. & Brandic I. 2009. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for

Delivering Computing as the 5th Utility. Future Generation Computer Systems 25(6), 599-616.

Cain T., Martin M., Heil T., Weglarz E. & Bezenek T. 2003. TPC-W Java Implementation 2003 [online]. [viitattu 14.9.2010]. Saatavilla www-osoitteessa < http://mitglied.multimania.de/jankiefer/tpcw/news.html>.

CloudHarmony 2010. What is an ECU? CPU Benchmarking in the Cloud [online]. [viitattu 10.12.2010]. Saatavilla www-osoitteessa <http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html>.

Coldfront Labs 2011. Making Drupal fly with APC, Memcache and Squid [online]. [viitattu 25.1.2011]. Saatavilla www-osoitteessa <http://coldfrontlabs.ca/blog/making-drupal-fly-apc-memcache-and-squid>.

CRN 2011. 10 Biggest Cloud Outages of 2010 (so far) [online]. [viitattu 2.2.2011]. Saatavilla www-osoitteessa <http://www.crn.com/slide-shows/applications-os/225701829/10-biggest-cloud-outages-of-2010-so-far.htm>.

Douglas H. & Gehrmann C. 2010. Secure Virtualization and Multicore Platforms State-of-the-Art Report. Swedish Institute of Computer Science (SICS). Technical Report T2009:14A.

Drupal 2010a. Drupal Suomi [online]. [viitattu 26.10.2010]. Saatavilla www-osoitteessa <http://drupal.fi/fi/drupal-suomi>.

Drupal 2010b. System requirements [online]. [viitattu 27.10.2010]. Saatavilla www-osoitteessa <http://drupal.org/requirements>.

Drupal 2011a. Drupal core [online]. [viitattu 6.5.2011]. Saatavilla www-osoitteessa <http://drupal.org/project/drupal>.

Drupal 2011b. Cache Support [online]. [viitattu 25.1.2011]. Saatavilla www-osoitteessa <http://drupal.org/node/15367>.

Durkee D. 2010. Why Cloud Computing Will Never Be Free. ACM Queue 8(4), 1-10.

EAccelerator 2011. eAccelerator [online]. [viitattu 25.1.2011]. Saatavilla www-osoitteessa <http://eaccelerator.net>.

Foster I., Yong Z., Raicu I. & Lu S. 2008. Cloud Computing and Grid Computing 360-Degree Compared. Teoksessa M. Pierce (toim.) Grid Computing Environments Workshop, GCE '08, Austin, USA, November 12-16, IEEE, 1-10.

Fouquet M., Niedermayer H. & Carle G. 2009. Cloud Computing for the Masses. Teoksessa J. Liebeherr & G. Ventre (toim.) Proceedings of the 1st ACM Workshop on User-provided Networking: Challenges and Opportunities (U-NET '09), New York, USA, ACM, 31-36.

Gao J.Z., Tsao H.S. & Wu Y. 2003. Testing and Quality Assurance for Component-Based Software. Norwood, Massachuttes: Artech House.

GoGrid 2010. Cloud Hosting, Cloud Servers, Hybrid Hosting, Cloud Infrastructure from GoGrid [online]. GoGrid Corporation [viitattu 26.10.2010]. Saatavilla www-osoitteessa < http://www.gogrid.com>.

GoGrid 2011. Frequently Asked Questions [online]. GoGrid Corporation [viitattu 16.2.2011]. Saatavilla www-osoitteessa <http://wiki.gogrid.com/wiki/index.php/Frequently_Asked_Question>.

Google 2010. Google App Engine [online]. Google Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://code.google.com/appengine/>.

Gray J. 1991. The Benchmark Handbook. San Mateo, California: Morgan Kaufmann Publishers.

Grossman R.L. 2009. The Case for Cloud Computing. IT Professional 11(2), 23-27.

Han S-M., Hassan M., Yoon C-W. & Huh E-N. 2009. Efficient Service Recommendation System for Cloud Computing Market. Teoksessa S. Sohn, K. Um, F.I.S. Ko, K.D. Kwack, J.H. Lee, G. Kou, K. Nakamura, A. Fong, P. Ma, L. Chen, S. Hwang, K. Cho & S. Kawata (toim.) Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09), New York, USA, ACM, 839-845.

He Q., Zhou S., Kobler B., Duffy D. & McGlynn T. 2010. Case Study for Running HPC Applications in Public Clouds. Teoksessa S. Hariri & K. Keahey (toim.) Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10), New York, USA, ACM, 395-401.

Hill Z., Li J., Mao M., Ruiz-Alvarez A. & Humphrey M. 2010. Early Observations on the Performance of Windows Azure. Teoksessa S. Hariri & K. Keahey (toim.) Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10), New York, USA, ACM, 367-376.

Hirvisalo V. 2000. Tehoa ohjelmiin. Suunnitellen eroon suorituskykyongelmista. Prosessori/ES, Marraskuu, 98-99.

IDC Predictions 2010. IDC Top 10 Predictions 2010: Recovery and Transformation [online]. International Data Corporation. [viitattu 14.5.2010]. Saatavilla www-osoitteessa <http://www.idc.com/research/predictions10/predictions10.jsp>.

Iosup A., Yigitbasi N. & Epema D. 2010. On the Performance Variability of Production Cloud Services. Delft University of Technology, Department of Technical Mathematics and Informatics. Technical Report PDS-2010-002.

Jain R. 1991. The Art of Computer Systems Performance Analysis. Techniques for Experimental Design, Measurement, Simulation, and Modeling. Yhdysvallat: John Wiley & Sons.

John & Cailin 2011. Lamp Performance on the Elastic Compute Cloud: Benchmarking Drupal on Amazon EC2 [online]. [viitattu 14.2.2011]. Saatavilla www-osoitteessa <http://www.johnandcailin.com/blog/john/lamp-performance-elastic-compute-cloud-benchmarking-drupal-amazon-ec2>.

Krishnaswamy U. & Scherson I.D. 2000. A Framework for Computer Performance Evaluation Using Benchmark Sets. IEEE Transactions on Computers 49(12), 1325-1338.

Leavitt N. 2009. Is Cloud Computing Really Ready for Prime Time?. Computer 42(1), 15-20.

Li A., Yang X, Kandula S. & Zhang M. 2010. CloudCmp: Comparing Public Cloud Providers. Teoksessa M. Allman (toim.) Proceedings of the 10th Annual Conference on Internet Measurement (IMC '10), New York, USA, ACM, 1-14.

Linthicum D. 2009. Cloud Computing and SOA Convergence in Your Enterprise. Crawfordsville, Indiana: Addison-Wesley.

Mather T., Kumaraswamy S. & Latif S. 2009. Cloud Security and Privacy. Sebastopol: O'Reilly Media.

Mei Y., Liu L., Pu X. & Sivathanu S. 2010. Performance Measurements and Analysis of Network I/O Applications in Virtualized Cloud. Teoksessa

S.S. Yau & L-J. Zhang (toim.) Proceedings of the 3rd International Conference on Cloud Computing (CLOUD '10), Washington, USA, IEEE Computer Society, 59-66.

Memcached 2011. Memcached - a Distributed Memory Object Caching System [online]. [viitattu 25.1.2011]. Saatavilla www-osoitteessa <http://memcached.org/>.

Menascé D.A. 2002. Load Testing of Web Sites. IEEE Internet Computing 6(4), 70-74.

Menascé D.A. 2003. Security Performance. IEEE Internet Computing 7(3), 84-87.

Menascé D.A., Almeida V.A.F. & Dowdy L.W. 2004. Performance by Design. Computer Capacity Planning by Example. Upper Saddle River, New Jersey: Prentice Hall PTR.

Meier J., Farre C., Bansode P., Barber S. & Rea D. 2007. Performance Testing Guidance for Web Applications: Patterns & Practices. Microsoft Press.

Metal Toad Media 2011. Plotting Your Load Test with JMeter [online]. Metal Toad Media Corporation [viitattu 4.3.2011]. Saatavilla www-osoitteessa <http://www.metaltoad.com/blog/plotting-your-load-test-jmeter>.

Microsoft 2010. Windows Azure Platform [online]. Microsoft Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://www.microsoft.com/windowsazure/>.

Murray P. 2009. Enterprise Grade Cloud Computing. Teoksessa C. Pu, M. Kersten, R. Oliveira & P. Murray (toim.) Proceedings of the Third Workshop on Dependable Distributed Data Management (WDDM '09), New York, USA, ACM, 1-1.

Napper J. & Bientinesi P. 2009. Can Cloud Computing Reach The TOP500?. Teoksessa A. Hast, R. Buchty, J. Tao & J. Weidendorfer (toim.) Proceedings of the Combined Workshops on Unconventional High Performance Computing Workshop Plus Memory Access Workshop (UCHPC-MAW '09), New York, USA, ACM, 17-20.

Ponder C. 1990. Performance Variation Across Benchmark Suites. SIGMETRICS Performance Evaluation Review 18(3), 42-48.

Porotskiy S.M. & Fateev A.E. 1992. System and Real Performance Evaluation of Computer. SIGMETRICS Performance Evaluation Review 20(2), 43-46.

Rackspace 2010. Cloud Servers - Virtual Server Hosting & Dedicated Server Hosting [online]. Rackspace Corporation [viitattu 25.10.2010]. Saatavilla www-osoitteessa <http://www.rackspacecloud.com/cloud_hosting_products/servers>.

Rackspace 2011. Frequently Asked Questions - Cloud Servers Wiki [online]. Rackspace Corporation [viitattu 16.2.2011]. Saatavilla www-osoitteessa <http://cloudservers.rackspacecloud.com/index.php/Frequently_Asked_Questions>.

Reese G. 2009. Cloud Application Architectures. Sebastopol: O'Reilly Media.

Sabalcore 2011. HPC Cloud Premium - Sabalcore Computing Inc. Rent a High Performance Cluster On-Demand. Linux Cluster [online]. Sabalcore Corporation [viitattu 2.2.2011]. Saatavilla www-osoitteessa <www.sabalcore.com>.

Salesforce.com 2010. Force.com: The leading cloud platform for business apps [online]. Salesforce.com Corporation [viitattu 1.6.2010]. Saatavilla www-osoitteessa <http://www.salesforce.com/platform/>.

Salokanto H. 2010. Sovelluskehitysympäristön virtualisoinnin tuomat edut ja haitat. Aalto-yliopiston teknillinen korkeakoulu. Tietoliikenne- ja tietoverkkotekniikan diplomityö.

Schad J., Dittrich J. & Quiané-Ruiz J-A. 2010. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. Teoksessa H.V. Jagadish (toim.) Proceedings of the VLDB Endowment 3(1-2), USA, VLDB Endowment, 460-471.

Shafer J. 2010. I/O Virtualization Bottlenecks in Cloud Computing Today. Teoksessa A.L. Cox, S. Rixner & M. Ben-Yehuda (toim.) Proceedings of the 2nd Conference on I/O Virtualization (WIOV'10), Berkeley, USA, USENIX Association, 31-37.

Shan H. & Strohmaier E. 2010. A Multi-dimensional Micro-benchmark for Performance Study and Prediction. Teoksessa W-K. Ching, H. Jin, J. Li, S.K. Mishra & L. Yu (toim.) Third International Joint Conference on Computational Science and Optimization (CSO), 28-31 May, Washington, USA, IEEE Computer Society, 156-160.

Smith J.E. & Ravi N. 2005. The Architecture of Virtual Machines. Computer 38(5), 32-38.

Sotomayor B., Montero R.S., Llorente I.M. & Foster I. 2009. Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing 13(5), 14-22.

Subraya B.M. 2006. Integrated Approach to Web Performance Testing - a Practitioners Guide. United Kingdom: IRM Press.

Sys-Con 2008. Twenty-One Experts Define Cloud Computing [online]. Sys-Con Media Corporation [viitattu 13.1.2011]. Saatavilla www-osoitteessa <http://cloudcomputing.sys-con.com/node/612375>.

TIA 2011. Standards Development Overview [online]. Telecommunications Industry Association [viitattu 5.5.2011]. Saatavilla www-osoitteessa <http://www.tiaonline.org/standards>.

The Bitsource 2010. Rackspace Cloud Servers versus Amazon EC2: Performance Analysis [online]. [viitattu 10.12.2010]. Saatavilla www-osoitteessa <http://www.thebitsource.com/featured-posts/rackspace-cloud-servers-versus-amazon-ec2-performance-analysis>.

TPC 2010a. TPC Benchmarks [online]. Transaction Processing Performance Council Corporation [viitattu 11.8.2010]. Saatavilla www-osoitteessa <http://www.tpc.org/information/benchmarks.asp>.

TPC 2010b. TPC Benchmark W (Web Commerce) Specification [online]. Transaction Processing Performance Council Corporation [viitattu 14.9.2010]. Saatavilla www-osoitteessa <http://www.tpc.org/tpcw/spec/TPCWV2.pdf>

Vandyk J. 2008. Pro Drupal Development. USA: Apress.

Walker E. 2008. Benchmarking Amazon EC2 for High-Performance Scientific Computing. Usenix Login 33(5), 18–23.

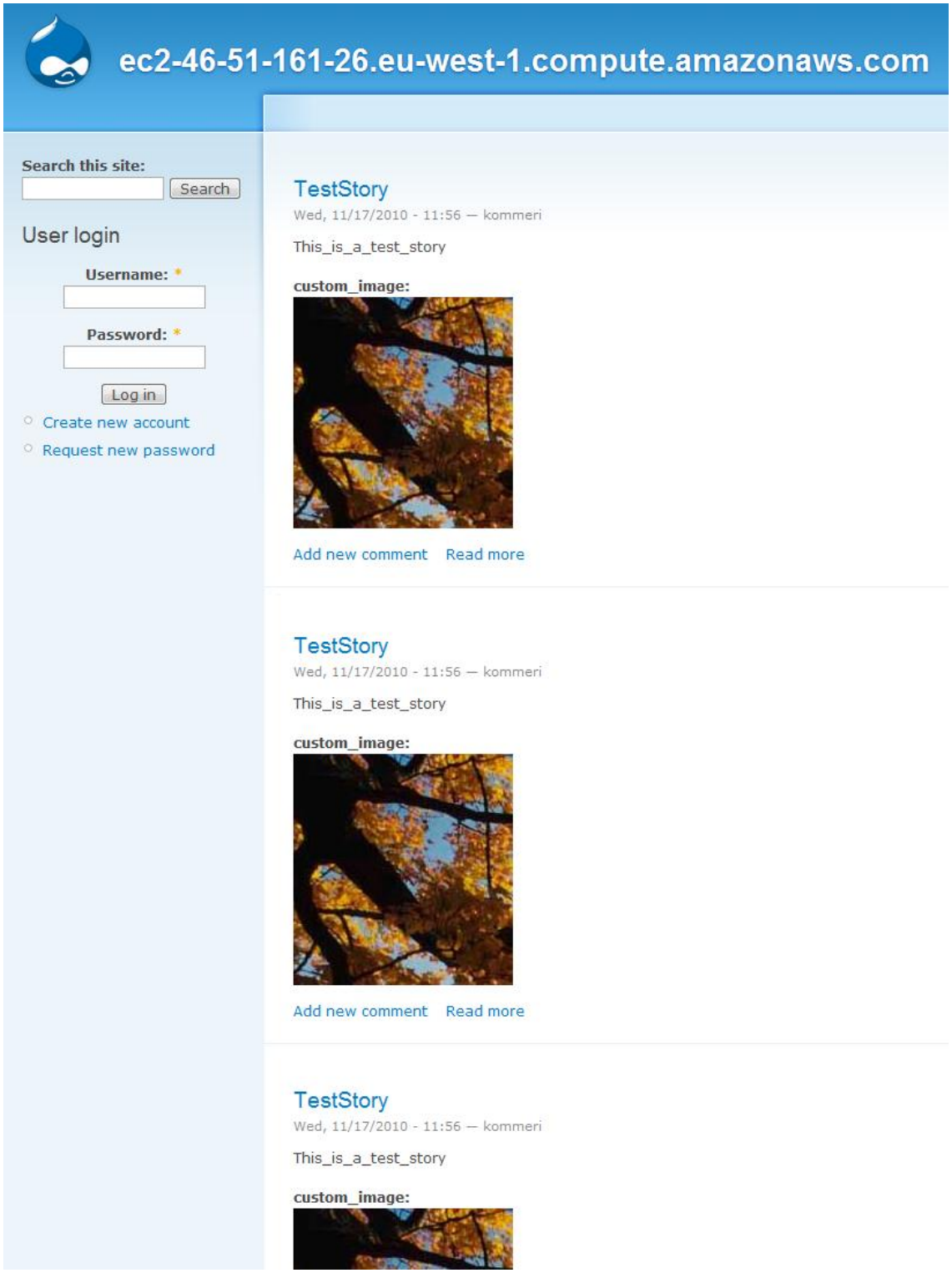
Wang G. & Ng T. 2010. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. Teoksessa G. Mandyam (toim.) Proceedings of the 29th Conference on Information Communications (INFOCOM'10), New York, USA, IEEE Press, 1163-1171.

Weinhardt C., Anandasivam A.,Blau B. & Stößer J. 2009. Business Models in the Service World. IT Professional 11(2), 28-33.

Youseff L., Butrico M. & Da Silva D. 2008. Toward a Unified Ontology of Cloud Computing. Teoksessa M. Pierce (toim.) Grid Computing Environments Workshop, GCE '08, Austin, USA, November 12-16, IEEE, 1-10.

Yuan P., Jin H., Ye D., Cao W., Yan Y. & Xie X. 2009. vBench: A Micro-Benchmark for File - I/O Performance of Virtual Machines. Teoksessa T. Chen Khong & F. Sew Bun (toim.) Services Computing Conference (APSCC 2009), 7-11 December, IEEE, 168-173.

LIITE 1: TESTATTAVAN SOVELLUKSEN ETUSIVU

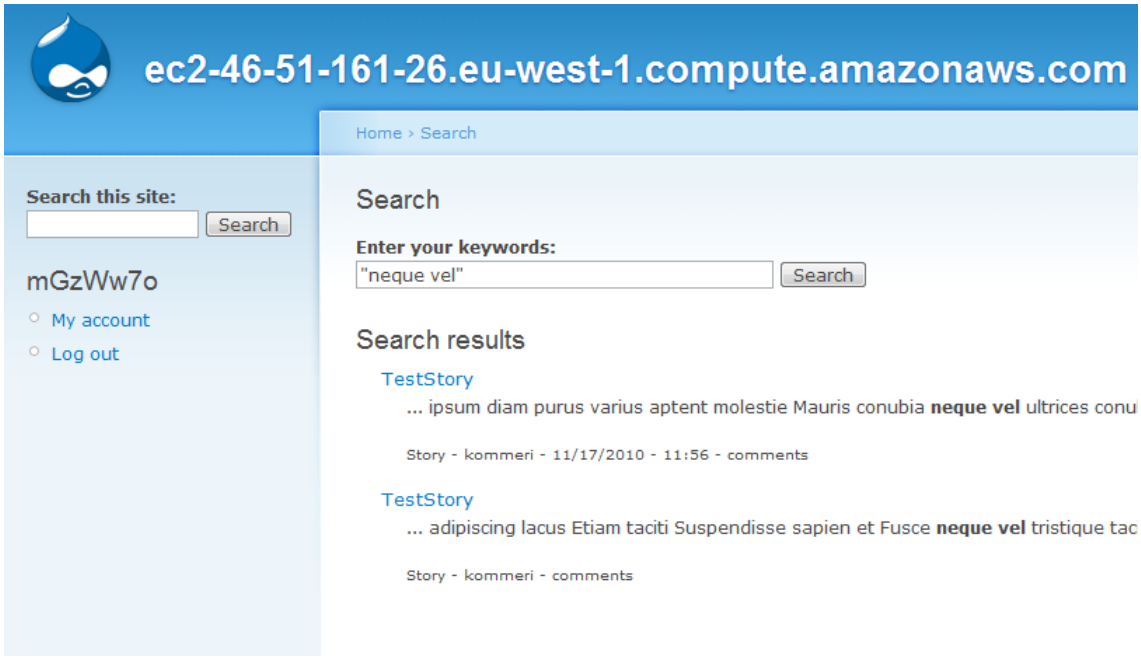


Kuva 19 Testattavan sovelluksen etusivu

LIITE 2: SATUNNAINEN SIVU TESTATTAVASSA SOVELLUKSESSA

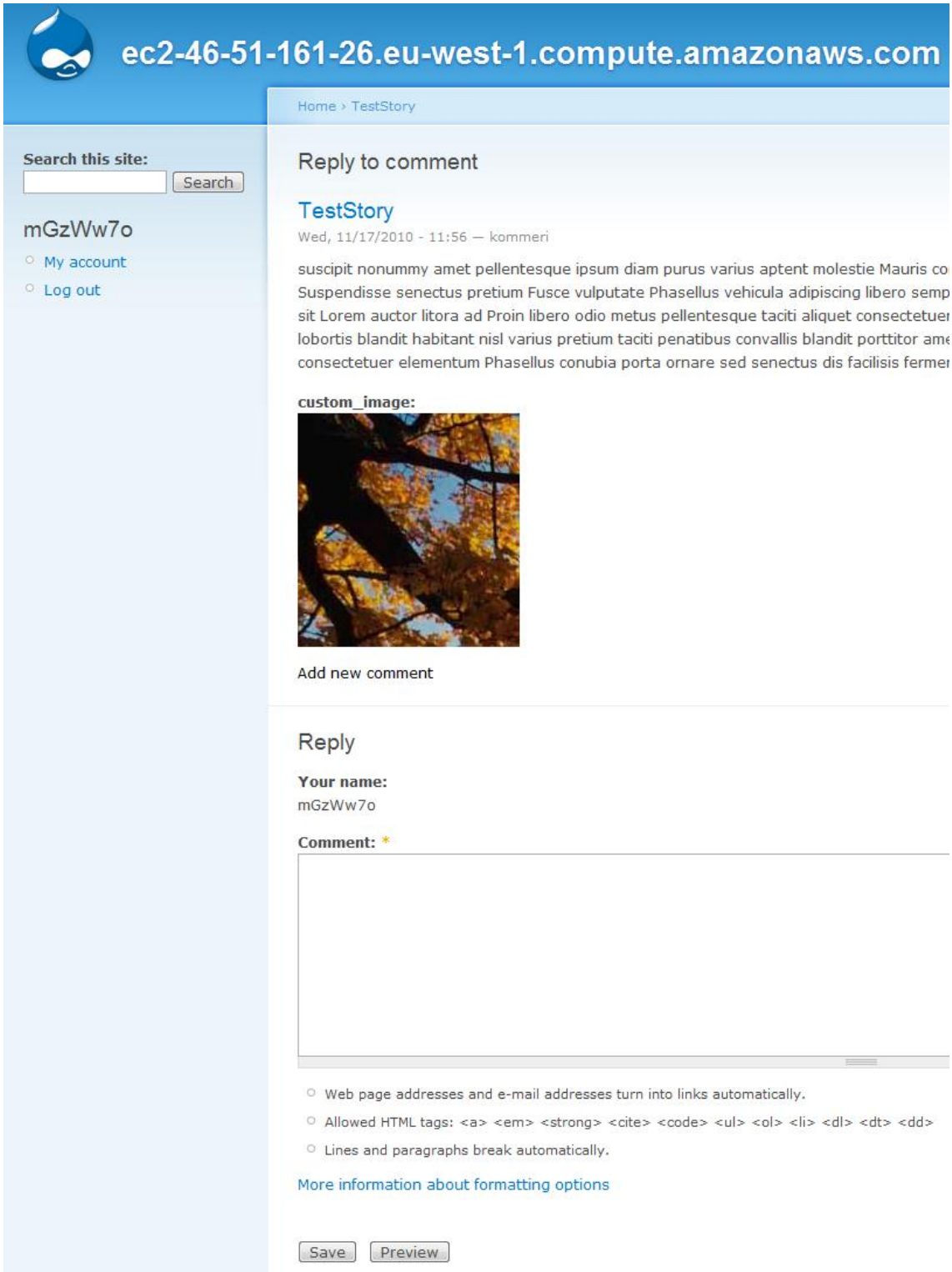
Kuva 20 Satunnainen sivu testattavassa sovelluksessa

LIITE 3: SISÄLLÖN HAKU TESTATTAVASSA SOVELLUKSESSA



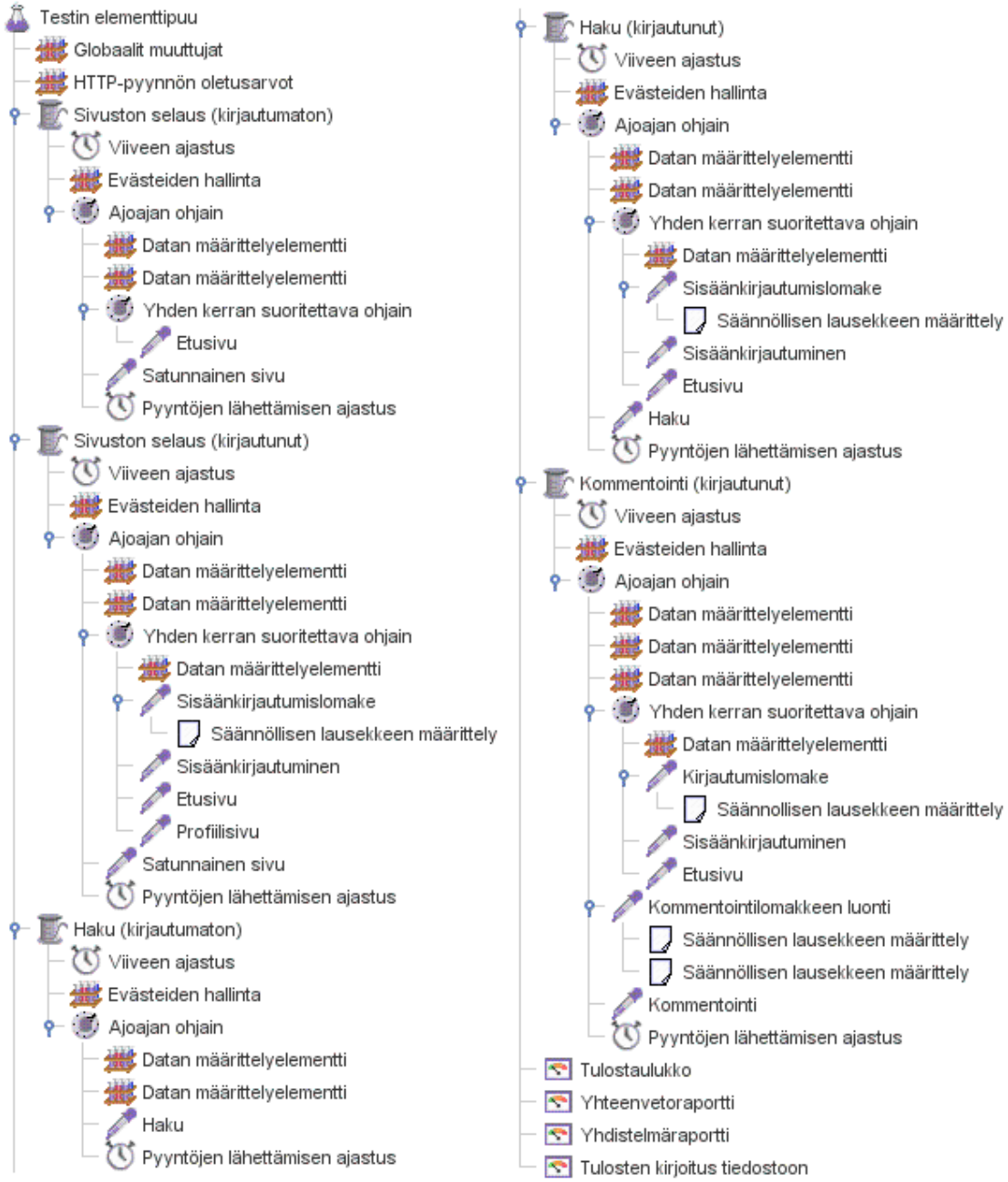
Kuva 21 Sisällön haku testattavassa sovelluksessa

LIITE 4: KOMMENTOINTI TESTATTAVASSA SOVELLUKSESSA



Kuva 22 Kommentointi testattavassa sovelluksessa

LIITE 5: ELEMENTTIPUU JMETER-OHJELMASSA



Kuva 23 Elementtipuu JMeter-ohjelmassa