Matti Rantakömi

Tallennusmedian suorituskyvyn vaikutus Wordpress-julkaisujärjestelmän toimintaan

Tietotekniikan
pro gradu -tutkielma
29. huhtikuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Matti Rantakömi
Yhteystiedot: matti@rantakomi.fi
Puhelinnumero: 040-561 2515
Ohjaaja: Risto T. Honkanen
Työn nimi: Tallennusmedian suorituskyvyn vaikutus Wordpress-julkaisujärjestelmän toimintaan
Title in English: Impact of storage media performance on Wordpress Content Management System operation
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 64
Tiivistelmä: Wordpress on julkaisujärjestelmä [38], jota käytetään verkkopalveluiden, kuten verkkosivujen, verkkokauppojen ja blogien alustana. Se on maailman ylivoimaisesti suosituin julkaisujärjestelmä [43]. Verkkopalvelun suorituskyvyllä on ratkaiseva vaikutus käyttökokemukseen, joten palvelun tulisi toimia mahdollisimman nopeasti kaikissa kuormitustilanteissa myös ruuhkahuippujen aikana. Hitaasti toimiva palvelu on yksi suurimmista kävijöitä karkottavista tekijöistä. Tässä tutkielmassa tutkittiin ja vertailtiin Wordpress-verkkopalvelun latausaikoja eri tasoisilla kovalevyillä varustetuilla virtuaalikoneilla konstruktiivisen tutkimusotteen avulla. Suorituskyky koostuu useammasta osatekijästä, kuten verkkopalvelun alustana käytettävien palvelinlaitteiden muistin, suorittimien ja tallennusjärjestelmän nopeudella. Suorituskykyä pyrittiin selvittämään ja vertailemaan eri tasoisilla laitealustoilla. Mittauksia varten testiympäristöiksi perustettiin kolme eri tasoisella kovalevyllä ja suoritinkapasiteetilla varustettua virtuaalikonetta, joihin asennettiin Wordpress-verkkopalvelu testisisältöineen. Lisäksi perustettiin neljäs virtuaalikone, johon asennettiin mittauksien suorittamiseen käytetty Locust-kuormitustestausohjelmisto. Mittauksen ensimmäisessä vaiheessa Wordpress-verkkopalvelun etusivua ladataan yhdellä käyttäjällä ilman Linuxin Page Cache -välimuistia. Toinen testi tehdään välimuistin kanssa aloittaen yhdellä käyttäjällä, jonka jälkeen käyttäjämäärää kasvatetaan tasaisesti aina 300 käyttäjään saakka. Sivunlatausaikojen mediaaneista koostetaan kuvaajat, joista latausaikojen kehitystä on helppo havainnoida. Toisen mittausvaiheen testeissä ladataan PHP-skriptiä, joka lukee levyltä samat tiedostot kuin Wordpressin etusivun latauksessa, mutta ei suorita PHP-koodia. Näin voidaan havainnoida aikaa, joka kuluu pelkästään tiedostojen lukemiseen kovalevyltä. Mittaustulokset olivat pitkälti sen suuntaisia kuin arvelinkin. Hitaimmalla kovalevyllä ja pienimmällä suoritinkapasiteetilla varustetun testivirtuaalikoneen latausajat oli-

vat kaikkein hitaimmat. Tämä korostui erityisesti ilman välimuistia tehdyissä mittauksissa. Vastaavasti nopeammilla kovalevyillä ja suuremmilla suoritinkapasiteeteilla varustettujen testivirtuaalikoneiden tulokset olivat huomattavasti nopeampia. Samaa testiä toistettaessa saadut tulokset olivat yhteneväisiä aiempien testien kanssa ja ja tulosten hajonta testikierrosten välillä oli pientä.
Avainsanat: iops, suorituskyky, wordpress, julkaisujärjestelmä, cms, verkkopalvelu, verkkokauppa
Abstract: Wordpress is a Content Management System [38], which is used as a platform for web services, such as websites, online stores and blogs. It's by far the most popular [43] Content Management System in the world. Web service performance has a decisive effect on the user experience, so the service should work as quickly as possible in all situations, even during peak traffic. Slowly responding service is one of the biggest factors driving visitors away. In this thesis, the response times of the Wordpress Content Management System on virtual machine platforms equipped with different performance levels of hard disks was investigated using a constructive research method. The performance consists of several factors, such as speed of the memory, processors and storage system used on underlying hardware. The aim was to find out and compare their performance on different platforms. Test environment consisted of three virtual machines equipped with different levels of hard disk and CPU capacity were set up as test environments, on which the Wordpress web service with test content was installed. Fourth virtual machine with Locust load testing software installed were used to perform the actual measurements. In the first stage of the performance measurement, the front page of the Wordpress web service is loaded by a single user without the Linux Page Cache cache. The second test is done with the cache, starting with one user, after which the number of users is steadily increased up to 300 users. Graphs are compiled from the medians of page loading times, from which it's easy to observe the development of loading times. In the tests of the second measurement phase, a PHP script is loaded, which reads the same files from the disk as when loading the Wordpress front page, but doesn't execute the PHP code. In this way, the time spent just reading files from the hard disk can be observed. The measurement results were largely in line with what I thought. The load times of the test virtual machine with the slowest hard drive and the lowest CPU capacity were the slowest. This was especially emphasized in measurements made without a cache. Correspondingly, the results of test virtual machines equipped with faster hard drives and larger CPU capacities were significantly faster. When

repeating the same test, the results obtained were consistent with the previous tests and the dispersion of the results between test rounds was small.
Keywords: iops, performance, wordpress, content management system, cms, web service, website, online store

Copyright © 2024 Matti Rantakömi

All rights reserved.

Sanasto

API-rajapinta	API (Application Programming Interface) on ohjelmointirajapinta, jonka avulla ohjelmistolle voidaan tehdä pyyntöjä, esimerkiksi noutaa tai tuoda tietoja tai hallita ohjelmiston asetuksia.
Hypervisor	Laitteistopohjaisen virtualisoinnin toteuttava käyttöjärjestelmäytimeen sisältyvä ohjelmistokomponentti.
IOPS	Input/Output operations Per Second. Kovalevyn tai muun tallennusmedian luku- ja kirjoitusoperaatioiden lukumäärä yhden sekunnin aikana.
Isäntäkone	Virtuaalikoneiden alustana toimiva palvelin. Useimmiten fyysinen palvelinlaite.
KVM	Kernel Virtual Machine. Linuxin käyttöjärjestelmäytimeen sisältyvä suoraan laitteistolla tavallisina prosesseina virtuaalikoneita suorittava ohjelmistokomponentti.
Käyttäjäavaruus	User Space. Käyttöjärjestelmän taso, jossa suoritetaan käyttäjien prosessit. Käyttäjäavaruudesta ei voi tehdä suoria kutsuja ytimen muistiavaruuteen, vaan se tapahtuu rajapintojen kautta.
Microsoft Hyper-V	Microsoftin julkaisema laitteistopohjainen virtualisointiohjelmisto.
Oracle VirtualBox	Oraclen julkaisema laitteistopohjainen virtualisointiohjelmisto.
Proxmox VE	Debian GNU/Linux -käyttöjärjestelmäjakeluun perustuva virtualisointikäyttöjärjestelmäjakelu.

QEMU	Quick EMUlator. KVM:n käyttäjäavaruudessa sijaitsevat toiminnallisuudet sisältävä ohjelmistokomponentti.
Räkkiyksikkö	Rack Unit. Konesaleissa käytettävien räkkikaappien rivien korkeutta kuvaava yksikkö. Yksikön korkeus on 1,75” eli 44,45 mm ja siitä käytetään lyhennettä U. Kaappeihin sijoitettavien laitteiden korkeus on useimmiten tämän yksikön mukainen tai sen kerrannainen.
Sisäkkäinen virtualisointi	Virtualisointiympäristön suorittaminen toisen virtuaalikoneen sisällä eli käytännössä virtuaalinen isäntäkone.
Virtuaalikovalevy	Virtuaalinen, fyysisellä kovalevyllä tai useammasta kovalevystä koostuvalla levyosiolla sijaitseva levynkuvatiedosto, joka esitetään sitä käyttävälle virtuaalikoneelle samaan tapaan lohkolaitteena (block device) kuin fyysinen kovalevy.
Virtualisointi	Fyysisen laitteiston ohjelmallinen jakaminen useammaksi virtuaaliseksi tietokoneeksi.
TTFB	Time To First Byte. Aika, joka kuluu käyttäjän selaimen lähettämästä HTTP-pyynnöstä web-palvelimen lähettämän HTTP-vastauksen ensimmäisen tavun vastaanottamiseen.
VMware vSphere	VMwaren julkaisema laitteistopohjainen virtualisointiohjelmisto.
Ylivaraus (Overbooking)	Isäntäkoneen resurssien (suoritin, keskusmuisti, kovalevytila) allokointi virtuaalikoneille siten, että kokonaisallokaatio ylittää isäntäkoneen resurssit.
Äänekäs naapuri	Virtuaalikone, joka joko vihamielisesti, tai suoritettavan sovelluksen ohjelmointivirheen vuoksi häiritsee muiden samalla isäntäkoneella sijaitsevien virtuaalikoneiden toimintaa isäntäkonetta ylikuormittamalla.

Sisällys

1 Johdanto

Tässä tutkielmassa havainnoidaan Wordpress-julkaisujärjestelmällä [38] toteutetun testisivuston latausnopeutta eri tasoisilla tallennusjärjestelmillä varustetuilla Linux-virtuaalikoneilla [30]. Wordpress on julkaisujärjestelmä, jota käytetään verkkopalveluiden, kuten verkkosivujen, verkkokauppojen ja blogien alustana. Se on ylivoimaisesti maailman suosituin julkaisujärjestelmä [43]. Verkkopalvelun suorituskyvyllä on ratkaiseva vaikutus palvelun käyttökokemukseen, joten palvelun tulisi toimia mahdollisimman nopeasti kaikissa kuormitustilanteissa myös ruuhkahuippujen aikana samanaikaisten käyttäjien määrästä riippumatta. Hitaasti toimiva palvelu on yksi suurimmista verkkopalvelun kävijöitä karkottavista tekijöistä. Suorituskyky koostuu useammasta osatekijästä, joista merkittävimpiä julkaisujärjestelmän ohjelmakoodin laadun ohella ovat palvelimen verkkoyhteys sekä palvelinlaitteen keskusmuistin, suorittimen ja tallennusjärjestelmän suorituskyky.

Monissa tapauksissa merkittävin yksittäinen toimintanopeuteen vaikuttava tekijä on palvelinlaitteen tallennusjärjestelmän suorituskyky. Verkkopalveluiden tuottamisessa käytetään monenlaisia ja eri laitesukupolvia edustavia palvelinlaitteita. Käytettävän laitealustan valintaa ja käyttöä ohjaavat tyypillisesti taloudelliset tekijät ja monesti myös asiakkaan tai tilaajan vähäinen tekninen osaaminen. Pilvipalveluiden yleistymisen myötä palvelinympäristöt ovat hyvin usein virtualisoituja, jolloin niitä suoritetaan isäntäkoneilla, joiden resurssit ja tallennusjärjestelmän kapasiteetti jaetaan usean virtuaalikoneen kesken.

Toimintanopeutta testataan kolmesta virtuaalikoneesta koostuvalla testiympäristöllä, joihin on asennettu Wordpress-verkkopalvelu testisisältöineen. Lisäksi neljänteen virtuaalikoneeseen on asennettu Locust-kuormitustestausohjelmisto, jonka avulla suorituskykymittaukset suoritetaan. Testivirtuaalikoneet on nopeuserojen havainnoimiseksi varustettu eri tasoisilla virtuaalikovalevyillä ja toisistaan poikkeavilla suoritinkapasiteeteilla. Suorituskykymittauksissa Wordpress-verkkopalvelun etusivua ladataan yksittäisellä käyttäjällä ilman Linuxin Page Cache -välimuistia sekä välimuistin kanssa aloittaen yhdellä käyttäjällä, jonka jälkeen käyttäjämäärää kasvatetaan tasaisesti aina 300 käyttäjään saakka. sivunlatausaikojen mediaaneista koostetaan kuvaajat, joista latausaikojen kehitystä on helppo havainnoida. Toi-

nen mittaus suoritetaan samaan tapaan kuin aiempikin testi, mutta siinä ladataan kirjoittamaani PHP-skriptiä, joka lukee levyltä samat tiedostot kuin Wordpressin etusivun latauksessa, mutta ei suorita PHP-koodia. Näin voidaan havainnoida aikaa, joka kuluu luettaessa tiedostot kovalevyltä.

Tehtyjen suorituskykymittausten tulokset olivat pitkälti sen suuntaisia, mitä ennen mittausten aloittamista arvelinkin. Hitaimmalla kovalevyllä ja pienimmällä suoritinkapasiteetilla varustetun server1-testivirtuaalikoneen latausajat olivat kaikkein hitaimmat. Nopeammalla kovalevyllä ja tuplasti suuremmalla suoritinkapasiteetilla varustetun server2-virtuaalikoneen tulokset olivat huomattavasti nopeampia. Vielä edellistäkin nopeammalla kovalevyllä ja edelleen tuplasti suuremmalla suoritinkapasiteetilla varustetun server3-testivirtuaalikoneen tulokset olivat edelleen hieman nopeampia kuin server2:n testitulokset. Ilman välimuistia suoritettujen testitulosten hajonta oli hyvin pientä ja latausajat tasaisia. Välimuistin kanssa suoritetun testin latausajat pitenivät tasaisesti kuormituksen kasvaessa, eikä suurempia heittoja suuntaan tai toiseen ilmennyt lainkaan. Kaikki kolme testipalvelinta suoriutuivat 300 samanaikaisen käyttäjän aiheuttamasta kuormasta, mutta latausajat moninkertaistuivat lähtötilanteeseen nähden, jossa käyttäjiä oli vain yksi.

Tutkielman luvussa 2 esitellään tutkimusmenetelmä sekä kuvaillaan tutkimusongelma ja tutkimusprosessi, jonka puitteissa tutkimus toteutetaan. Luku 3 käsittelee verkkopalvelukokonaisuuden rakennetta ja koostumusta. Luvussa käydään läpi verkkopalvelukäsite yleisellä tasolla ja paneudutaan tarkemmin Wordpress-julkaisujärjestelmän toteutukseen. Verkkopalvelun alustana toimivan fyysisen konesaliympäristön koostumus tarvittavine tiloineen ja laitteineen esitellään yleisellä tasolla. Sovellusympäristön osalta aiheeseen pureudutaan hieman syvällisemmin ja käydään läpi verkkopalvelun sovellusympäristön kaikki osatekijät palvelintietokoneen käyttöjärjestelmästä lähtien.

Luvussa 4 esitellään verkkopalvelun alustana toimivan palvelimen tallennusjärjestelmä. Tässä luvussa keskitytään isäntäpalvelimen tallennusjärjestelmään. Käsite käydään läpi yleisellä tasolla ja lisäksi kuvaillaan erilaisia tallennustekniikoita.

Luku 5 käsittelee tämän tutkielman tutkimusaihetta eli suorituskykymittauksia. Aluksi esitellään testiympäristön toteutus käytettävine sovelluksineen ja tekniikoineen. Suorituskykymittausten osalta keskitytään mittaamaan I/O-operaation [19] suoritusaikaa ja vastaavasti verkkopalvelun ohjelmakoodin suoritusaikaa. Mittauksia suoritetaan eri tasoilla virtuaalikovalevyillä varustetuilla virtuaalikoneilla. Mittaukset toteutetaan avoimen lähdekoodin Locust-kuormitustestausohjelmis-

tolla [45]. Varsinaisten suorituskykymittausten ja testien ohella luvussa 5 käsitellään myös muita verkkopalvelun toimintanopeuteen vaikuttavia osatekijöitä, kuten käyttäjän oman verkkoyhteyden nopeutta ja käytettävän laitteen, kuten puhelimen, tabletin tai tietokoneen suorituskyvyn vaikutusta kokonaisuuteen.

Luku 6 esittelee luvussa 5 toteutettujen suorituskykymittausten tulokset, havaitut ongelmakohteet ja pohdintaa siitä, miten havaitut ongelmat saataisiin ratkaistua ja siten parannettua verkkopalveluiden toimivuutta. Viimeisenä luku 7 sisältää yhteenvedon tutkielman aiheesta yleisellä tasolla.

2 Tutkimusmenetelmä

Tutkimusmenetelmänä tässä tutkielmassa käytetään konstruktiivista tutkimusotetta. Se tarjoaa dynaamisen lähestymistavan ongelmien ratkaisuun, joka on hyödyllistä erityisesti tämän tutkielman kaltaisten, nopeasti kehittyvää teknologiaa käsittelevien, aihealueiden tutkimuksessa. Uuden ratkaisun, työkalun tai menetelmän kehittäminen muodostaa tutkimusmenetelmän ytimen. Tutkimusmenetelmän tarkoituksena ei ole ainoastaan ymmärtää nykytietämystä, vaan myös pyrkiä laajentamaan sitä uusilla konkreettisilla ratkaisuilla.

Konstruktiivista tutkimusotetta voi soveltaa laajalti monenlaisiin tutkimuksiin, vaikka se onkin alunperin kehitetty liiketaloustieteen tarpeisiin. Tätä samaa tutkimusotetta on sovellettu paljon, joko sellaisenaan tai mukautettuna, myös tietojärjestelmätieteiden, lääketieteen ja kasvatustieteiden aloilla [27]. Konstruktiivinen tutkimusmenetelmä on kehitetty Suomessa Eero Kasasen ja Kari Lukan toimesta. Tutkimusongelma, konstruktiivisen tutkimusprosessin rakenne ja itse tutkimuksen kulku käydään läpi omissa aliluvuissaan.

2.1 Tutkimusongelma

Miten verkkopalvelun kävijää palvelevan palvelimen tallennusmedian suorituskyky vaikuttaa Wordpress-julkaisujärjestelmän toimintaan? Tutkimus lähtee etsimään vastausta kysymykseen ja tavoitteena on selvittää Wordpress-pohjaisten verkkopalveluiden alustana toimivien ja eritasoisia tallennusmedioita käyttävien palvelimien suorituskykyä ja suurinta mahdollista yhtäaikaista käyttäjämäärää, josta palvelin kykynee suoriutumaan ilman palvelun toiminnan hidastumista. Tutkimusaiheen laajuuden rajaamiseksi tutkimus rajattiin tarkoituksellisesti koskemaan vain Wordpress-pohjaisia verkkopalveluita. Se antaa kuitenkin hyvää taustatietoa myös muiden verkkopalveluiden käyttäytymisestä.

Verkkopalveluista on nykyisen tietoyhteiskunnan ja digitaalisen vallankumouksen myötä tullut merkittävä osa ihmisen jokapäiväisen toiminnan taustatekijöistä. Verkkopalveluiden toiminnan varmistaminen suurta yleisöä samanaikaisesti koskettavien tapahtumien aikana voi olla haastavaa, sillä suuri kävijämäärä on omiaan

aiheuttamaan verkkopalvelun toiminnan hidastumista tai pahimmillaan palvelun toiminnan estymisen. Tällainen tilanne on omiaan aiheuttamaan asiakkaiden turhautumista ja siten asiakastyytyväisyyden laskua. Pahimmillaan palvelu menettää asiakkaita ja sitä kautta palvelusta riippuen joko suoraan tai epäsuorasti selvää rahaa.

Ruuhkatilanteita aiheutuu suuren määrän käyttäjiä pyrkiessä samanaikaisesti käyttämään palvelua, jolloin sen kapasiteetti ylittyy. Tätä voi verrata tilanteeseen, jossa kivijalkakaupan kassalle kertyy jonoja, koska kassojen kapasiteetti on rajallinen. Kassojen kapasiteettia ei kustannussyistä ole mitoitettu palvelemaan asiakkaita rinnakkain niin suurta määrää, kuin suurimpien ruuhkahuippujen aikaan olisi tarpeellista, jotta kaikki asiakkaat saataisiin palveltua ilman jonoutumista. Tästä seuraa yksittäisen asiakkaan kassatapahtuman eli jonottamisen, tuotteiden viivakoodien lukemisen sekä maksutapahtuman yhteiskestoajan piteneminen.

2.2 Konstruktiivinen tutkimusprosessi

Konstruktiiviseen tutkimusprosessiin kuuluu kaiken kaikkiaan seitsemän eri vaihetta. Se alkaa ongelman tunnistamisesta ja tutkittavan asian määrittelystä. Hyvä tutkimusaihe on sellainen, josta on olemassa vain vähän aiempaa tutkimustietoa, ja jolla on käytännöllistä merkitystä. On myös tärkeää pyrkiä tuottamaan kontribuutiota sille tieteenalalle, jolla tutkimusta sovelletaan. Kun tutkimusaihe on määritelty, seuraavaksi selvitetään mahdollisuudet pitkän aikavälin tutkimusyhteistyöhön kohdeorganisaation kanssa. Käytännössä tutkimuksesta kannattaa pyrkiä tekemään pitkäkestoinen projekti, johon osallistuu kohdeorganisaation avainhenkilöitä, jotka tutkijan kanssa muodostavat projektityöryhmän. Tutkimussopimus tulee laatia kirjallisesti ja siihen kannattaa sisällyttää tutkimuksen kannalta oleelliset asiat, kuten ehdot tutkimustulosten julkaisemiseen ja tutkittaviin tietoihin käsiksi pääsyyn [27]. Tutkimuksen rahoittamiseen liittyvät asiat tulee myös kirjata sopimukseen. Tutkijan tulisi aina pyrkiä siihen, että tutkimusprojektin tulokset on mahdollista julkaista. Näin varmistetaan, että tutkimusprojektin akateeminen kontribuutio toteutuu, eivätkä tulokset jää vain kohdeorganisaation sisäiseen käyttöön.

Tutkimussopimuksen laatimisen jälkeen alkaa perehtyminen tutkimusaiheeseen ja kohdeorganisaatioon. Tutkijan tulee perehtyä tutkittavaan aiheeseen mahdollisimman kattavasti ja huolellisesti kirjallisten aineistojen analysoinnin, haastattelujen ja havainnoinnin menetelmin. Tässä vaiheessa on myös hyvä pyrkiä käsitteel-

listämään tutkimusongelmaa tutkijan ja muun projektityöryhmän kesken, jotta voidaan varmistaa kaikkien puhuvan ”samaa kieltä” ja ymmärtävän toisiaan [27]. Tutkijan tulee myös perehtyä saman aihepiirin mahdolliseen olemassa olevaan olevaan tutkimusaineistoon ja teorioihin, jotta tutkimuksen valmistuttua sitä on mahdollista vertailla ja analysoida muita vastaavankaltaisia tutkimuksia vasten.

Neljännessä vaiheessa kehitetään konstruktio, jolla tunnistettu ongelma pyritään ratkaisemaan. Konstruktiota innovoidessa on tärkeää pyrkiä tuottamaan sellainen ratkaisu, joka voi tuottaa myös teoreettista kontribuutiota. Konstruktiota kehitettäessä on tärkeää, että tutkijan ohella kehitystyöhön osallistuvat myös projektityöryhmän muut jäsenet [27]. Iteratiivisesti toteutettava kehitystyö on aikaa vievä prosessi, joka alkaa konstruktioideoiden keksimisestä ja suunnittelusta. Tämän jälkeen ideoiden pohjalta voidaan tehdä pienimuotoisia implementaatiota, joilla testataan konstruktiota ja kerätään kokemuksia. Saatujen kokemusten pohjalta havaittuja ongelmia pyritään korjaamaan konstruktiota parantelemalla.

Viidennessä vaiheessa siirrytään varsinaiseen toteutukseen konstruktion kehittämisen jälkeen. Tarkoituksena on testata kehitetty konstruktio teknisesti ja sen lisäksi myös kokonaisuutena siten, että tutkijan lisäksi työhön osallistuu koko projektityöryhmä [27]. Vaiheeseen sisältyy toteutuksen lisäksi henkilökunnan koulutusta, ohjeistuksen valmistelua sekä mahdollisesti myös pilottitestejä. Tutkimusprosessissa erityisesti juuri tämä vaihe poikkeaa perinteisestä akateemisen tutkimuksen linjasta, joka pyrkii olemaan puuttumaton ja neutraali, siltä osin että tutkija on tiiminsä kanssa sitoutunut ”myymään” kehitettyä innovaatiota kohdeorganisaatiolle.

Seuraavassa vaiheessa tutkija analysoi prosessin tuloksia peilaten niitä tutkimuksen ennakkoehtoihin ja lähtöasetelmaan. On tärkeää, että tutkija pystyy kontrolloimaan omaa sitoutumistaan ja ottamaan etäisyyttä omaan empiiriseen työhönsä, jotta hän pystyy analysoimaan tuloksia neutraalisti [27]. Tässä vaiheessa on tärkeää pohtia tutkimuksen myötä läpikäytyä oppimisprosessia yhdessä kohdeorganisaation kanssa. Jos tutkimusprosessissa kehitetyn konstruktion tulokset ovat odotettuja eli markkinatesti on läpäisty onnistuneesti, on tärkeää pohtia myös sitä, miten pääpiirteiltään samaa konstruktiota voisi hyödyntää vastaavanlaisiin tarpeisiin myös muissa organisaatioissa. Vastaavasti markkinatestin epäonnistuessa on hyvä pohtia mikä testin tulos olisi ollut toisissa organisaatioissa ja olisiko epäonnistumisen voinut välttää.

Tutkimuksen viimeisessä vaiheessa tutkijan täytyy pystyä selittämään projektin teoreettinen kontribuutio vertaamalla sitä esimerkiksi mahdollisesti jo olemassa ole-

viin vastaaviin teorioihin. Konstruktiivisen tutkimuksen luultavasti tyypillisin tulos on teorian jalostus, sillä tutkimuksen myötä syntyy tarve muokata ja tarkistaa omia aiempia, tutkimusta edeltäviä ja voimassaolevia, käsityksiämme tutkimuksen prosesseja ja rakenteita koskevista riippuvuussuhteista [27]. Edellisen vaiheen tapaan myös tässä vaiheessa on tärkeää, että tutkija pystyy kontrolloimaan omaa sitoutumistaan ja ottamaan etäisyyttä omaan empiiriseen työhönsä, jotta tutkimuksen vertailu onnistuisi neutraalisti. Teoreettisen kontribuution tuottaminen on mahdollista konstruktion itsensä osalta, sillä kehitetyn uuden konstruktion toimiessa alkuperäisessä ympäristössään se tuottaa lisäystä aiempaan olemassa olevaan tietoon ja kirjallisuuteen. Toisaalta riittävä tulos kontribuution tuottamisen osalta on jo se, että ongelmaa ratkaistaessa hyödynnettiin aiemmin tunnettu teoriaa. Lisäksi tutkimuksessa kehitetyn konstruktion riippuvuussuhteet mahdollistavat teoreettisen tietämyksen kehittämisen, soveltamisen ja testaamisen.

2.3 Tutkimuksen kulku

Tutkimus aloitetaan konstruktiivisen tutkimusotteen periaatteiden mukaisesti määrittelemällä tutkimusaihe tai -ongelma. Tämän tutkielman tutkimusongelmaksi on jo aiemmassa luvussa määritelty kysymys: Miten verkkopalvelun kävijää palvelevan palvelimen tallennusmedian suorituskyky vaikuttaa Wordpress-julkaisujärjestelmän toimintaan? Tutkimuksessa lähdetään siis selvittämään vastauksia ja etsimään ratkaisuja tähän kysymykseen.

Määrittelyn jälkeen keskitytään tutkimusongelman aiheeseen ja pyritään hankkimaan syvällinen tuntemus sekä teoreettisesti että käytännöllisesti. Itselläni on päivätyössä hankittu yli 15 vuoden kokemus tutkimusongelman parissa työskentelystä, joten koen ymmärtäväni aiheesta varsin paljon jo entuudestaan. Toki tutkimuksen aikana tulen syventämään osaamistani entisestään tutustumalla aihepiirin julkaisuihin ja kirjallisuuteen.

Seuraavaksi pyritään kehittämään ratkaisu ongelmaan. Termi on ehkä sinällään hieman harhaanjohtava tässä yhteydessä, sillä tutkimuksessa ei varsinaisesti ratkaista ongelmaa. Sen sijaan tutkimuksessa keskitytään selvittämään minkä tasoisilla kovalevyillä varustetuilla virtuaalikoneilla Wordpress-verkkopalvelua on mahdollista ylläpitää kelvollisesti siten, että palvelin pystyy suoriutumaan tietyn kokoisen yhtäaikaisen kävijämäärän palvelemisesta ilman latausaikojen pidentymistä. Usein palvelimen suorituskyky korreloi pitkälti sen hinnan kanssa, joten pienelle verkko-

palvelulle ei kannata hankkia liian suurta palvelinympäristöä ja vastaavasti suurta verkkopalvelua ei kannata tukehduttaa ylläpitämällä sitä liian tehottomalla palvelimella. Asian havainnoimiseksi ja mittaamiseksi pystytetään useammasta Ubuntu Linux -käyttöjärjestelmällä [8] varustetusta QEMU-pohjaisesta virtuaalikoneesta [41] koostuva testiympäristö.

Ratkaisua lähdetään toteuttamaan ja testaamaan perustamalla virtuaalikoneita Proxmox VE (Virtualization Environment) -virtualisointialustalle [33], joka on asennettu Hetzneriltä vuokrattuun fyysiseen palvelinlaitteeseen. Palvelimessa on Intel i5-12500 -suoritin, 64 gigatavua keskusmuistia ja 512 gigatavun NVMe-kovalevy, joten se on riittävän suorituskykyinen näihin mittauksiin. Virtuaalikoneiden virtuaalikovalevyjen suorituskykyä rajoitetaan ohjelmallisesti, sillä tarkoituksena on jäljitellä tavallisen pyörivän kovalevyn [10], SSD-kovalevyn [47] ja NVMe-kovalevyn [1] suorituskykyä. Varsinaisten testivirtuaalikoneiden lisäksi perustetaan yksi virtuaalikone kuormitustestien suorittamista varten. Tälle virtuaalikoneelle asennetaan Locust-kuormitustestausohjelmisto [45] ja se sijoitetaan samaan 1 Gbps -nopeudella toimivaan lähiverkkoon kuin varsinaiset testivirtuaalikoneet.

Kuormitustesteissä verkkopalvelun etusivua kuormitetaan Locustin avulla yhdellä käyttäjällä aloittaen. Näin pyritään ensin jäljittelemään tilannetta, jossa yksittäinen kävijä vierailee verkkosivustolla. Kävijämäärää kasvatetaan lineaarisesti vähitellen aina 300 käyttäjään saakka. Samalla tarkkaillaan sivunlatauspyyntöjen latausaikaa sekä latausaikojen mahdollista pidentymistä käyttäjämäärän kasvaessa. Kaikille eri suorituskykyisillä kovalevyillä varustetuille virtuaalikoneille suoritetaan sama testi.

Koska sivunlatauspyyntö, suoritettava Wordpressin PHP-ohjelmakoodi ja siten palautettavan sivun sisältö on aina samanlainen, pystytään ensimmäisten sivunlatausten jälkeen seuraavat lataukset todennäköisesti palvelemaan joko kokonaan tai osittain Linux-käyttöjärjestelmäytimen Page Cache [58] -välimuistista, joka taltioi eniten käytettyjä tiedostoja suoraan keskusmuistiin. Näin ollen ensimmäinen latauspyyntö on todennäköisesti kaikkein hitain ja sitä seuraavien pyyntöjen latausajat huomattavasti nopeampia ja keskenään suunnilleen samanpituisia. Page Cachen vaikutuksen eliminoimiseksi latausaikoja testataan myös yksittäisellä käyttäjällä siten, että välimuisti tyhjennetään manuaalisesti jokaisen kierroksen jälkeen. Näin jokainen latauspyyntö aiheuttaa kaikkien pyynnön suorittamiseen liittyvien tiedostojen lataamisen palvelimen kovalevyltä, sillä niitä ei pystytä tarjoamaan välimuistista.

Normaalikäytössä Page Cache on hyödyllinen ja palvelimen toimintaa nopeuttava ominaisuus. Sen ansiosta Wordpressin tiedostoja ei tarvitse lukea levyltä uudelleen jokaisen sivunlatauksen yhteydessä, vaan riittää, kun ne luetaan kertaalleen ja tallennetaan Page Cacheen, joka käyttää keskusmuistia tallennusmedianaan. Kun tiedostot löytyvät välimuistista, ne voidaan tarjota web-palvelimelle ja PHP-koodia suorittavalle Apachen mod_php -moduulille [53] nopeasti ilman, että niitä tarvitsee lukea kovalevyltä. Esimerkiksi palvelimen uudelleenkäynnistyksen jälkeen ensimmäinen sivunlataus on hidas, koska välimuisti on tyhjä. Seuraavat sivunlataukset ovat nopeita, koska ne pystytään palvelemaan suoraan välimuistista. Page Cache on olennainen osa Linuxin käyttöjärjestelmäydintä ja sen levyoperaatioihin liittyvää toimintaa, eikä se ole kytkettävissä pois päältä.

Kuormitustestejä toistetaan riittävän monta kertaa, jotta testituloksista saadaan mahdollisimman luotettavia ja voidaan välttyä mahdollisilta satunnaisvaihteluilta. Lisäksi mahdolliset selkeästi virheelliset testitulokset suodatetaan pois. Testituloksista koostetaan taulukoita, joiden avulla eri testivirtuaalikoneiden tuloksia voidaan vertailla keskenään. Tuloksista tehdään myös graafisia kuvaajia helpottamaan luettavuutta. Testien valmistuttua tuloksia analysoidaan ja arvioidaan niistä mahdollisesti saatavaa tieteellistä kontribuutiota. On oletettavaa, että testeistä saadaan hyödyllistä ja tutkittua tietoa eri tasoisten ympäristöjen kuormituksen kestosta ja ideoita toimintanopeuden parantamiseen.

3 Verkkopalvelun rakenne ja komponentit

Tässä luvussa paneudutaan verkkopalveluun rakenteeseen ja komponentteihin, jotka yhdessä muodostavat verkkopalveluksi kutsuttavan kokonaisuuden. Verkkopalvelu käsitetään tässä kontekstissa kokonaisuutena, joka koostuu pääasiassa kahdesta osasta. Näitä ovat palvelun fyysinen alusta eli palvelinympäristö ja vastaavasti sovellusympäristö, joka käsittää palvelun tuottamiseen tarvittavat ohjelmistokomponentit.

Aliluku 3.1 käsittelee verkkopalvelua yleisellä tasolla verkkopalvelu-käsitettä taustoittaen ja verkkopalveluiden historiaa esitellen. Aliluku 3.2 keskittyy Wordpress-julkaisujärjestelmään. Wordpressin historia käydään läpi ensimmäisestä versiosta lähtien ja lisäksi käsitellään teknistä toteutusta sekä suorituskykyyn liittyviä asioita. Verkkopalveluiden palvelinympäristöihin liittyvät asiat käsitellään aliluvussa 3.3 ja sovellusympäristöt aliluvussa 3.4.

3.1 Verkkopalvelu

Verkkopalvelut ovat tärkeä osa nykyihmisen arkea ja ne tarjoavat yrityksille mahdollisuuden tavoittaa asiakkaansa verkossa ja tarjota heille palveluita ja tuotteita. Ne ovat tänä päivänä hyvin yleisiä sekä yksityishenkilöiden että yritysten käytössä. Verkkopalvelun määritelmä on melko lavea ja se on käsitteenä verrattain uusi ja siten myös vakiintumaton. Rissanen [42] kertoo verkkopalveluista puhuttaessa tarkoitettavan staattisia kotisivuja, erillisiä palveluja tarjoavia ohjelmistoja sekä erillisten rajapintojen välityksellä palveluja tarjoavia ohjelmistoja. Vastaavasti Parkkinen [36] toteaa verkkopalvelun olevan Internetin kautta käytettävä palvelu tai palvelukokonaisuus, joka saavutetaan www-selaimen avulla.

Tämän tutkielman yhteydessä verkkopalvelulla tarkoitetaan verkkosivustoa tai -kauppaa, asiakassivustoa tai muuta vastaavaa internetselaimella käytettävää palvelua. Koska tutkielma keskittyy pääosin Wordpressiin ja sen suorituskyvyn tutkimukseen, ei muihin verkkopalvelutoteutuksiin paneuduta tätä pintapuolista tarkastelua syvällisemmin. Kuten todettua, verkkopalvelu voidaan toteuttaa joko staattisena sivustona tai dynaamisesti, jolloin käyttäjälle näytettävä sisältö muodostetaan

dynaamisesti verkkopalvelun alustana toimivan ohjelmiston toimesta. Nykypäivänä tyypillisen verkkopalvelun alustana toimii julkaisujärjestelmä, joka huolehtii sisällön tuottamisesta sivuston käyttäjän nähtäville.

Nykyaikaisten julkaisujärjestelmien ansiosta verkkopalveluita voidaan käyttää hyvin monenlaisiin tarkoituksiin. Tyypillisiä käyttökohteita ovat yritysten mainos-, tiedotus- ja asiakaspalvelukanavat. Sen lisäksi verkkokaupat ovat yksi yleisimmistä verkkopalvelun ilmentymistä. Koska verkkopalveluita on ollut yleisesti saatavilla jo pitkälti yli 20 vuoden ajan ja ihmiset ovat tottuneet käyttämään niitä päivittäin, niiden oletetaan toimivan aina ja kaikissa tilanteissa, ympäri vuorokauden.

Internetin alkuaikoina verkkopalvelu tarkoitti hyvin usein käytännössä staattisista sivuista koostuvaa verkkosivustoa, jolla esitettiin tekstiä ja kuvia. Dynaamista toiminnallisuutta ei ollut välttämättä lainkaan. Itse asiassa internetin ensimmäinen vuonna 1993 julkaistu verkkosivu sisälsi ainoastaan kevyesti muotoiltua tekstiä ja hyperlinkkejä [7], jotka olivat käytännössä ankkuriviittauksia saman sivuston eri osiin, koska muihin sivustoihin ei luonnollisestikaan vielä tässä vaiheessa voinut linkata.

Yksinkertainen, pelkästään tekstiä ja kuvia sisältävä staattinen sivusto ajaa edelleen asiansa hyvin monessa tarkoituksessa ja tarvitsee toimiakseen vain vähän palvelinresursseja, joten staattisen sivuston toteutus on kokonaisuutena yksinkertainen ja edullinen ylläpitää. Tällaisen sivuston ylläpito vaatii kuitenkin jonkin verran asiantuntemusta ja perehtyneisyyttä aiheeseen, sillä sivustolle tarvitaan palvelintila, jonne teksti- ja kuvatiedostot täytyy siirtää esimerkiksi FTP- tai SCP-yhteyden [46] avulla.

Staattisten palveluiden rinnalla julkaisujärjestelmät yleistyivät nopeasti, koska ne mahdollistivat sivuston nopean ja helpon päivittymisen ja ylläpidon pelkästään internetselainta käyttäen HTML-koodin [51] kirjoittamisen sijaan. Lisäksi dynaamiset verkkopalvelut mahdollistivat interaktiiviset verkkopalvelut, joita voidaan käyttää esimerkiksi asiakaspalvelutarkoituksiin korvaamaan ja täydentämään palveluita, joiden asiakaspalvelun kommunikaatio on aiemmin hoidettu joko menemällä paikan päälle yrityksen toimipisteeseen, puhelimitse, sähköpostitse tai kirjepostin välityksellä.

3.2 Wordpress-julkaisujärjestelmä

Wordpress on avoimeen lähdekoodiin perustuva, alunperin blogien eli verkkopäiväkirjojen julkaisuun kehitetty sisällönhallintaohjelmisto eli julkaisujärjestelmä. Järjestelmä on toteutettu PHP-ohjelmointikielen [49] avulla ja sen ensimmäinen versio näki päivänvalon vuonna 2003 [55]. Wordpressin suosio on kasvanut vuosien mittaan ja se on nykyään maailman suosituin julkaisujärjestelmä [43]. Lähteistä riippuen Wordpressin markkinaosuuden väitetään olevan jopa 65 % [54]. Julkaisujärjestelmän lähdekoodi on saatavilla avoimesti ja ilmaiseksi. Myös järjestelmän käyttö ja asentaminen on ilmaista.

3.2.1 Yleistä

Wordpressin omistaja on yhdysvaltalainen, vuonna 2005 perustettu teknologiayhtiö Automattic Inc. Sen toimitusjohtajana toimii Matt Mullenweg [54], joka on yksi yhtiön perustajajäsenistä ja Wordpressin isä. Yhtiö työllistää noin 2000 henkilöä ja sen päätuote on Wordpress.com-verkkopalvelu, joka tarjoaa ylläpidettyjä Wordpress-pohjaisia verkkopalveluratkaisuja [5]. Vaikka Wordpress on saatavilla ilmaiseksi, on monelle käyttäjälle kuitenkin helpompaa ja luontevampaa ostaa palvelu valmiiksi asennettuna ja ylläpidettynä, jolloin asiakas voi itse keskittyä pelkästään sivuston sisällön tuottamiseen.

Automattic Inc. -yhtiön menestys perustuu hyvin tuotteistettuun palveluun, joka on tehty asiakkaalle mahdollisimman helppokäyttöiseksi ja vaivattomaksi. Yritys syntyi nimen omaan vastaamaan tarpeeseen Wordpressin asennuksen ja käytön tekemisestä helppokäyttöiseksi. Yhtiön arvon on vuonna 2022 arvioitu olevan noin 7,5 miljardia yhdysvaltain dollaria [54].

Suosion kasvuun vaikuttaneita tekijöitä ovat erityisesti helppokäyttöisyys, monipuolisuus ja laajennettavuus. Verkkopalvelua voi hallinnoida internetselaimella helppokäyttöisen käyttöliittymän avulla, josta palvelun sisällön lisäys, muokkaus ja poisto onnistuvat näppärästi ilman syvällisempää teknistä osaamista. Ilmaisia ja maksullisia, erinäisiä lisätoiminnallisuuksia tarjoavia, laajennuksia on saatavilla valtava määrä. Sivuston ulkoasua voi muokata niin ikään joko ilmaisilla tai maksullisilla teemoilla. Teemojen luonti myös omatoimisesti on suhteellisen helppoa.

Wordpressiä käytetään verkkopalveluiden, kuten verkkosivujen, verkkokauppojen ja blogien alustana. Verkkopalvelun toimintanopeudella on ratkaiseva vaikutus käyttökokemukseen, joten palvelun tulisi toimia mahdollisimman nopeasti kai-

kissa kuormitustilanteissa myös ruuhkahuippujen aikana. Hitaasti toimiva palvelu onkin yksi suurimmista verkkopalvelun kävijöitä karkoittavista tekijöistä. Suorituskyky koostuu useammasta osatekijästä, joista merkittävimpiä julkaisujärjestelmän ohjelmakoodin laadun ohella ovat palvelinlaitteen keskusmuistin, suorittimien ja tallennusjärjestelmän suorituskyky sekä verkkoyhteyden liikennöintinopeus ja latausaika.

3.2.2 Tekninen toteutus

Wordpress-julkaisujärjestelmän palvelimella suoritettava lähdekoodi on toteutettu PHP-ohjelmointikielellä [55]. Verkkopalvelulle ominaisesti käyttäjän selaimelle lähetettävien vastausten, joiden perusteella selain renderöi näytettävän sivun käyttäjän katsottavaksi, sisältö koostuu pääasiassa HTML- [51], CSS- [12] ja JavaScript-koodista [16]. Lähdekoodi on jaettu pieniin osiin omiin tiedostoihinsa siten, että yksittäinen tiedosto sisältää jonkin tietyn asian, kuten yhden luokan lähdekoodin. Kaiken kaikkiaan lähdekoodi käsittää noin 1000 tiedostoa ja hakemistoa [56] ja koko asennuspaketti yli 3000. Asennuspakettiin sisältyy varsinaisten PHP-lähdekooditiedostojen lisäksi CSS-tyylitiedostoja, PNG-kuvatiedostoja [2], JavaScript- ja HTML-tiedostoja, kirjasintiedostoja sekä liuta muita tiedostoja, kuten Readme-tyyppisiä tekstidokumentteja.

Wordpress on tietokantapohjainen julkaisujärjestelmä ja se käyttää tietovarastonaan MySQL/MariaDB-tietokantaa [57]. Myös PostgreSQL-tuen [57] lisäämisestä Wordpressiin on ollut keskustelua, mutta asiaa ei ainakaan tällä hetkellä aktiivisesti edistetä. Tietokantaan tallennetaan kaikki verkkopalvelun tekstisisältö, kuten artikkelit ja sivut sekä käyttäjätiedot ja asetukset. Tietokannan toimintanopeus on verkkopalvelun nopean toiminnan kannalta kriittistä, sillä yksittäinen sivunlataus saattaa aiheuttaa kymmeniä tai satoja tietokantakyselyitä. Näin ollen hitaiden tietokantakyselyiden kerrannaisvaikutuksen johdosta verkkopalvelusta tulee helposti käyttökelvoton kävijöiden kyllästyessä odottamaan sivujen latautumista.

3.2.3 Suorituskyky

Wordpress-pohjaisen verkkopalvelun suorituskyky on monen tekijän summa. Siihen vaikuttavat mm. Wordpressin ja mahdollisesti käytettävien lisäosien koodin laatu, palvelimen suorituskyky, käyttäjän ja palvelimen verkkoyhteyden nopeus ja latausaika. Wordpressin koodia kehitetään jatkuvasti ja uusien versioiden myötä jul-

kaistaan parannuksia ja bugikorjauksia myös aiempien toiminallisuuksien ja ominaisuuksien koodiin. Palvelimen suorituskykyyn vaikuttavia tekijöitä ovat suorittimen, keskusmuistin ja tallennusjärjestelmän nopeus. Verkkopalvelun suorituskyvyn kannalta on tärkeää, että kaikki osatekijät ovat kunnossa.

Sivustolla vierailevan käyttäjän verkkoselaimen tekemän yksittäisen sivunlatauksen aikaansaama prosessi alkaa selaimen lähettämästä HTTP-sivunlatauspyynnöstä, joka lähetetään sivustoa tarjoavalle palvelimelle. Kun pyyntö saapuu, palvelimen käyttöjärjestelmä välittää sen Apache-web-palvelimelle, joka aloittaa pyynnön käsittelyn. Web-palvelin käsittelee staattisiin tiedostoihin, kuten kuviin, CSS-tyylitiedostoihin ja JavaScript-koodeihin, kohdistuvat latauspyynnöt itsenäisesti ja palauttaa selaimelle pyydetyn tiedoston sisällön. Kaikki Wordpressin sivusto-osoitteisiin kohdistuvat sivunlatauspyynnöt käsitellään Apachen mod_php -moduulin toimesta. Kaikki pyynnöt ohjataan PHP-koodin sisältävälle index.php-tiedostolle, joka sisällyttää myös muita kooditiedostoja suoritettavaksi tarpeen mukaan. Koodin suorittamisen jälkeen valmis vastaus palautetaan käyttäjälle web-palvelimen toimesta.

Sivuston sisällöstä ja käytettävistä lisäosista riippuen yksittäisen sivunlatauspyynnön myötä suoritettavan index.php-lähdekooditiedoston suorittamiseksi palvelimen tallennusjärjestelmästä joudutaan hakemaan ja lataamaan jopa useampi sata lähdekooditiedostoa, joista index.php on riippuvainen, ja jotka sisällytetään siihen koodin suorituksen aikana. Koska jokaisen yksittäisen lähdekooditiedoston hakuun ja lataukseen kovalevyltä kuluu tietty aika, on hitaasti toimivalla tallennusjärjestelmällä kerrannaisvaikutuksineen merkittävä rooli kokonaislatausajan muodostumisessa. Levyoperaatioiden tehostamiseksi palvelimilla käytetään yleisesti erinäisiä välimuistiratkaisuja, jotka tyypillisesti nopeuttavat levyoperaatioita, mutta voivat myös aiheuttaa ei-toivottuja lieveilmiöitä, kuten vanhentuneiden tiedostojen tarjoamista välimuistista. Dynaamisten sisältöjen vuoksi kaikkea ei myöskään voida välimuistittaa.

3.3 Palvelinympäristö

Tyypillinen verkkopalveluiden ylläpitämiseen käytettävä palvelinympäristö koostuu useammasta eri osa-alueesta. Palvelinlaitteet sijoitetaan tyypillisesti hyvin suojattuihin konesaleihin, joihin on rakennettu hyvät tietoliikenneyhteydet. Konesalit on tyypillisesti kalustettu standardikokoisilla lukittavilla 19” -räkkikaapeilla [39],

joihin palvelin- ja verkkolaitteet saadaan kiinnitettyä. Helpoin tapa saada palvelin laitetilaan on vuokrata tilaa ja tietoliikenneyhteydet joltakin alan toimijalta. Yksittäiselle palvelinlaitteelle voi vuokrata oman laitepaikan jaetusta kaapista tai isompia kokonaisuuksia varten oman tai tarvittaessa useammankin laitekaapin.

3.3.1 Konesali

Konesali on tekninen, hyvin suojattu tila, joka on suunniteltu palvelimien, levyjärjestelmien, tietoliikennelaitteiden ja muiden toimintaan liittyvien ja sitä tukevien laitteiden säilytykseen ja aktiivikäyttöön [15]. Konesaliin sijoitettavilla palvelinlaitteilla toimii yritysten ja julkisten tahojen kriittisiä verkkopalveluita sekä muita järjestelmiä, joiden oletetaan olevan saavutettavissa kaikki aikoina ympäri vuorokauden.

Konesalin sähkönsyöttö on tyypillisesti varmistettu UPS (Uninterruptible Power Supply) -laitteistolla [15]. UPS-laitteiston lisäksi jatkona toimivat dieselkäyttöisillä generaattorit, joiden ansiosta virransaanti voidaan taata ja täten varmistaa palvelimien toiminnan jatkuvuus myös pidempien sähkökatkojen aikana. Verkkoyhteydet tulee rakentaa vikasietoisiksi siten, että ne käyttävät useamman eri operaattorin verkkoja ja fyysiset kaapeloinnit kulkevat useampaa reittiä pitkin rakennuksen sisällä ja siitä eteenpäin maan alla tontin ulkopuolelle siirryttäessä.

Konesaliympäristön palveluiden toiminnan valvonta on avainasemassa hyvän ja luotettavan konesalipalvelun tuottamisessa. Ympäristön lämpötila ja ilmankosteus tulee pitää tasaisena, jotta laitteiden mahdollisimman vakaa ja tasainen toiminta voidaan taata. Yksi konesaliympäristön toimintaa uhkaavista asioista on mahdollinen tulipalo. Vettä ei sähkölaitteiden kanssa kannata käyttää sammutusaineena, sillä se on omiaan aiheuttamaan hapettumia ja oikosulkuja elektronisissa laitteissa. Konesalit onkin usein varustettu kaasupohjaisilla sammutusjärjestelmillä, joiden toiminta perustuu hapen syrjäyttämiseen tai paremminkin sen pitoisuuden pienentämiseen konesalin ilman koostumuksessa.

3.3.2 Palvelinlaite

Palvelinlaite voi olla käytännössä mikä tahansa tietokone. Varsinkin harrastuskäytössä on yleistä, että palvelinlaitteena toimii aivan tavallinen PC-tietokone. Myös kannettava tietokone voi toimia palvelimena siinä missä pöytäkonekin. Kun kotikäyttäjät ajoittain hankkivat uusia tietokoneita, jää vanhoja ylimääräisiksi. Har-

rastuskäytössä esimerkiksi vanhan kotitietokoneen valjastaminen palvelinkäyttöön on usein houkutteleva vaihtoehto, sillä se on kustannuksiltaan edullinen ratkaisu. Myös käytetyn tavaran kauppapaikoilla vanhoja tietokoneita on saatavilla hyvin edullisesti tai jopa ilmaiseksi, joten alkuun pääsee hyvin pienillä kustannuksilla.

Edellä mainitut kotikäyttöön suunnitellut laitteet eivät kuitenkaan ole luonteva vaihtoehto kaupalliseen käyttöön, sillä niitä ei tyypillisesti ole suunniteltu ympärivuorokautista käyttöä silmällä pitäen. Kotikäyttöön tarkoitettujen laitteiden valmistuksessa on käytetty edullisempia komponentteja ja niiden testaus ei ole yhtä kattavaa kuin ympärivuorokautiseen palvelinkäyttöön tehdyissä laitteissa. Lisäksi laitteiden fyysinen kotelointi ei ole optimaalinen konesaliasennuksissa käytettävien räkkien mitoituksiin nähden.

Kaupallisessa käytössä palvelimena toimii tyypillisesti räkkiasennukseen tarkoitettu yksittäinen laite tai suurempaan blade-palvelinkehikkoon asennettava korttipalvelin. Kukin laite varustetaan halutun määrän suoritinytimiä tarjoavalla suorittimella sekä halutulla muisti- ja tallennuskapasiteetilla. Laitteisiin asennetaan käyttöjärjestelmä ja virtualisointikerros (hypervisor) apuohjelmineen. Tätä kokonaisuutta kutsutaan yleisesti isäntäkoneeksi.

3.3.3 Palvelinlaitteen resurssit

Palvelinlaitteen resurssit koostuvat suoritin-, keskusmuisti- ja tallennuskapasiteetista. Nykyiset palvelinkäyttöön tehtyjen tietokoneiden emolevyt on tyypillisesti varustettu joko yhdellä tai kahdella suoritinkannalla ja niihin asennettavien palvelinkäyttöön suunniteltujen suorittimien sisältämä suoritinytimien määrä vaihtelee muutamasta aina useisiin kymmeniin ytimiin per suoritin. Näin ollen yksittäiseen fyysiseen palvelimeen saadaan tarvittaessa käyttöön vähintään useita kymmeniä suoritinytimiä. Nykyaikaiset suorittimet tukevat säikeistystä [26], jonka avulla loogisten suorittimien lukumäärä saadaan tuplattua. Säikeistyksen avulla yksittäisessä suoritinytimessä voidaan suorittaa kahta tehtävää rinnakkaisesti, jolloin suorittimen suorituskykyä saadaan merkittävästi parannettua [23]. Intel käyttää omasta säikeistystekniikastaan nimitystä Hyper Threading [9] ja AMD:n vastaava tekniikka kantaa nimeä Simultaneous Multi-Threading [6].

Muistikapasiteetin määrä vaihtelee hyvin paljon kulloisenkin käyttötarpeen mukaan. Palvelinemolevyt on suunniteltu kuluttajatietokoneiden emolevyihin nähden suurempaa muistikapasiteettia varten ja ne sisältävät tyypillisesti 8-24 muistipaikkaa, joten isäntäkoneen muistin kokonaismäärä voi olla jopa usean teratavun luok-

kaa. Isolla muistikapasiteetilla varustettu palvelin pystyy luonnollisesti toimimaan isäntäkoneena suuremmalle määrälle muisti-intensiivisiä virtuaalikoneita. Tallennuskapasiteetin määrään pätevät samat lainalaisuudet kuin muistikapasiteettiinkin eli niitä hankitaan tapauskohtaisesti sen mukaan, mitä käyttötarve edellyttää. Mahdollisesti myös palvelimen hankintaprosessin aikana tapahtuvat heilahtelut komponenttien hinnoissa saattavat vaikuttaa siihen minkä laajuisilla resursseilla palvelin varustetaan.

3.3.4 Isäntäkone (hypervisor)

Isäntäkone (physical host, host node, hypervisor) on fyysinen palvelinlaite, jossa on tyypillisesti asennettuna vain käyttöjärjestelmä ja virtualisointiohjelmistokerroksen (hypervisor) suorittamisen kannalta välttämättömät apuohjelmat [3]. Tässä tutkielmassa keskitytään x86-pohjaiseen KVM:ään (Kernel Virtual Machine) [41] ja lisäksi KVM:n kanssa yhdessä käytettävään käyttäjäavaruuden komponentti QEMU:n (Quick EMUlator) [41]. Muita yleisesti käytettäviä x86-pohjaisia alustoja ovat mm. VMware [41], Oracle VirtualBox [3] ja Microsoft Hyper-V [41]. Saatavilla on myös muita, mm. BSD-käyttöjärjestelmää [41] hyödyntäviä virtualisointiratkaisuja.

Tyypillisesti fyysiseen palvelinlaitteeseen tehtävä virtualisointiohjelmistokerroskokonaisuuden asennus on mahdollista tehdä myös virtuaalikoneen sisälle. Tällaista ratkaisua nimitetään sisäkkäiseksi virtualisoinniksi (nested virtualization) [21]. Sisäkkäinen virtualisointi on mielekäs vaihtoehto esimerkiksi opiskelutarkoituksissa, koska sen asentamiseen tai käyttöön ei tarvita erillistä fyysistä palvelinlaitetta, vaan oppilaitos voi tarjota opiskelijoille käyttöön omat virtuaalikoneet, joiden sisälle kukin opiskelija voi asentaa oman virtualisointiohjelmistokerroksen ja käyttää sitä tietyin rajoittein samaan tapaan kuin suoraan fyysiseen laitteeseen tehtyä asennusta.

KVM sisältyy viralliseen avoimen lähdekoodin Linux-käyttöjärjestelmäytimeen omana moduulinaan ja sen asennus ja käyttöönotto onnistuvat helposti yleisimpiin Linux-käyttöjärjestelmäjakeluihin. Yksi yleinen ja suosittu virtualisointiohjelmisto on Proxmox VE (Virtualization Environment) [33], joka perustuu Debian GNU/-Linuxiin [11] sisältäen täydellisen hypervisor-toiminnallisuuden käsittäen KVM- ja QEMU-ohjelmistoineen valmiiksi asennettuna ja lisäksi Proxmox-projektin tuottaman selainpohjaisen hallintapaneelin, vastaavat toiminnot tarjoavan komentorivityökalun ja lisäksi muita virtuaaliympäristön hallintaan tarvittavia toimintoja ja ominaisuuksia.

Isäntäkoneen suorituskyky muodostuu ja määrittyy fyysiseen palvelinlaitteeseen asennettujen komponenttien perusteella ja on siten rajallista. Nykyaikaiset isäntäkoneet sisältävät tyypillisesti useita satoja gigatavua keskusmuistia, kymmeniä suoritinytimiä ja useamman teratavun verran tallennuskapasiteettia. Myös isäntäkoneista erilliset, joko lähiverkon tai erillisen, usein kuitupohjaisen, tallennusverkon kautta isäntäkoneisiin liitettävät levyjärjestelmät ovat yleisiä. Tällöin isäntäkone voidaan varustaa vain pienellä kovalevyllä, joka sisältää isäntäkoneen käyttöjärjestelmän ja virtualisointiohjelmistokerroksen (hypervisor) suorittamisen kannalta välttämättömät apuohjelmat.

Käytännössä mikä tahansa riittävillä resursseilla varustettu x86-pohjainen tietokone voi toimia isäntäkoneena. Kaupallisessa käytössä hyödynnetään enimmäkseen erityisesti palvelinkäyttöön suunniteltuja laitteita, sillä ne on tehty kestämään jatkuvaa ympärivuorokautista käyttöä vuosikausia. Lisäksi tällaisiin laitteisiin on saatavilla takuu- ja huoltopalveluita nopealla latausajalla, joka on tärkeää kriittisessä tuotantokäytössä oleville laitteille. Myös tavallisia PC-pöytätietokoneita käytetään palvelinkäytössä niiden edullisuuden vuoksi. Tällaiset laitteet toimivatkin varsin hyvin myös ympärivuorokautisessa käytössä kunhan laitteiston lämpötilasta, puhtaudesta ja ilmankierrosta huolehditaan tunnollisesti. Jopa kannettavia tietokoneita voi käyttää isäntäkoneena, mutta toisaalta kuluttajatason laitteita ei ole suunniteltu ympärivuorokautista käyttöä varten.

3.4 Sovellusympäristö

Tässä verkkopalvelukontekstissa sovellusympäristöllä tarkoitetaan kokonaisuutta, joka tarvitaan palvelemaan verkkopalvelua. Kokonaisuuden pohjana on virtuaalikone, johon asennetaan Ubuntu Linux -palvelinkäyttöjärjestelmä. Käyttöjärjestelmän lisäksi palvelimelle asennetaan Apache-webpalvelin [48] ja MariaDB-tietokantapalvelinsovellukset [29]. Molemmat ohjelmistot ovat avointa lähdekoodia, ilmaiseksi saatavilla ja kuuluvat oman kategoriansa käytetyimpien tuotteisiin.

Sovellusympäristöön kuuluvat komponentit käsitellään omissa aluluvuissaan. Aliluku 3.4.1 käsittelee virtuaalikonetta, joka muodostaa sovellusympäristön perustan. Aliluku 3.4.2 puolestaan keskittyy virtuaalikoneeseen asennettavaan käyttöjärjestelmään, aliluku 3.4.3 sovellusympäristön web-palvelimeen ja aliluku 3.4.4 kertoo sovellusympäristön tietokantapalvelimesta. Muita olennaisia sovellusympäristöön liittyviä asioita käsitellään aliluvussa 3.4.5, jossa käsitellään tiedostojärjestelmää, ali-

luvussa 3.4.6 verkkoyhteyttä ja aliluvussa 3.4.7 käyttäjän päätelaitetta.

3.4.1 Virtuaalikone

Virtuaalikone on ohjelmallisesti toteutettu palvelinympäristö, joka jäljittelee fyysistä palvelinlaitetta. Toiminnallisuudeltaan se on samankaltainen kuin fyysinenkin palvelin. Käyttöjärjestelmälle näytetään virtualisoituina fyysisestä palvelimesta löytyvät komponentit, kuten prosessori, muisti, kovalevy, verkkokortti sekä muut mahdolliset oheislaitteet [41]. Myös fyysisen laitteen läpivienti (pass through) suoraan virtuaalikoneelle on mahdollista, jolloin virtuaalikoneelle näytetään suoraan jokin fyysinen laite, kuten kovalevy tai näytönohjain, sellaisenaan ilman virtualisoitua laiteajuria.

Kuten edellisessä aliluvussa jo mainittiinkin, tässä tutkielmassa virtualisointiohjelmistona käytetään Proxmox VE:tä (Virtualization Environment) [33], joka perustuu Debian GNU/Linuxiin [11]. Virtuaalikoneen alustana käytetään x86-pohjaiseen KVM:ään (Kernel Virtual Machine) [41] ja sen kanssa yhdessä käytettävään käyttäjäavaruuden komponentti QEMU:n (Quick EMUlator) [41] pohjautuvaa virtuaalikonetta. Käytettävän virtualisointiohjelmiston ja -tekniikan valintaan vaikuttivat pääosin valitun ratkaisun suosio, avoin lähdekoodi ja ilmaisuus. Proxmox sisältää myös selainpohjaisen hallintapaneelin, vastaavat toiminnot tarjoavan komentorivityökalun sekä muita virtuaaliympäristön hallintaan tarvittavia toimintoja ja ominaisuuksia.

Yhdellä isäntäkoneella voidaan suorittaa useita virtuaalikoneita rinnakkain, jolloin osa isäntäkoneen resursseista allokoidaan kunkin virtuaalikoneen käyttöön. Käytännössä virtuaalikoneelle määritellään tietty muisti-, suoritin- ja tallennuskapasiteetti ja mahdollisesti verkkoyhteyden nopeutta rajoitetaan. Edellisten määritysten lisäksi tallennuskapasiteetin I/O- ja IOPS-suorituskyvyn rajoittaminen on tärkeää, jotta äänekäs naapuri ei pääse häiritsemään muiden samalla isäntäkoneella ajettavien virtuaalikoneiden toimintaa. Kaikki isäntäkoneen resurssit on mahdollista myös ylivarata eli allokoida resursseja virtuaalikoneiden yli isäntäkoneen kapasiteetin. Koska tämä on omiaan aiheuttamaan virtuaalikoneiden toiminnan hidastumista ruuhkatilanteissa, on suotavaa allokoida kunkin virtuaalikoneen käyttöön taatut resurssit ja jättää hieman suoritin- ja muistikapasiteettia myös isäntäkoneen omien prosessien suorittamiseen.

3.4.2 Käyttöjärjestelmä

Tässä kontekstissa käyttöjärjestelmä on palvelintietokoneeseen asennettu ohjelmisto, joka toimii kaiken kyseisessä tietokoneessa tapahtuvan ohjelmallisen toiminnan pohjana halliten tietokoneen resursseja ja tarjoten käyttäjälle tai muiden ohjelmistojen sovelluksille rajapinnan tietokoneen resurssien hyödyntämiseen. Se toimii tietokoneen keskeisenä ohjausohjelmistona, joka mahdollistaa muiden ohjelmistojen ja käyttäjän vuorovaikutuksen laitteiston kanssa. Yksinkertaisimmillaan käyttöjärjestelmä vastaa perustoiminnoista, kuten tiedostojen hallinnasta, muistinhallinnasta ja prosessien hallinnasta.

Linux-käyttöjärjestelmä on valta-asemassa verkkopalveluiden ylläpitoon käytettävissä ympäristöissä [13], joten se oli luonteva valinta myös tämän tutkielman palvelinten käyttöjärjestelmäksi. Erilaisia Linux-käyttöjärjestelmäjakeluita on saatavilla vähintään useita kymmeniä, joista osa on ilmaiseksi saatavilla olevia ja osa kaupallisia. Tässä tutkielmassa käytetään Ubuntu Linux -jakeluversiota, joka on saatavilla ilmaiseksi. Käytettävä versio on Ubuntu Server 22.04 LTS (Long Term Support), joka on Ubuntun viimeisin pitkän tuen omaava julkaisu ja tarkoitettu erityisesti palvelinkäyttöön. Ubuntua julkaiseva Canonical Ltd tarjoaa sille myös maksullista tukea.

Työpöytäkäytössä käytännössä kaikissa järjestelmissä käytössä olevaan graafiseen käyttöliittymään törmää Linux-palvelimien kanssa harvemmin, sillä järjestelmiä hallinnoidaan tyypillisesti komentorivipohjaisten sovellusten avulla. Sinänsähän työpöytä- tai palvelinkäyttöjärjestelmä ei ole kuin käsite, joten graafisen työpöytäympäristön voi aivan hyvin asentaa myös palvelimena toimimaan Linuxiin. Myös erilaisia selainpohjaisia hallintaohjelmistoja on laajasti saatavilla. Windows-palvelimissa graafinen käyttöliittymä on ylläpitokäytössä yleisempi ja graafisen työpöydän etäkäytön mahdollistava RDP-yhteys on usein pääasiallinen tapa hallinnoida palvelimia.

3.4.3 Web-palvelin

Web-palvelin on verkkopalvelua tarjoavalla palvelimella toimiva sovellus, jonka tehtävä on kommunikoida käyttäjän internetselaimen kanssa internetin välityksellä. Yksi ensimmäisistä web-palvelinsovelluksista on Apache Software Foundationin julkaisema Apache HTTP Server [48], jonka ensimmäinen versio näki päivänvalon jo vuonna 1995. Ohjelmisto on edelleen aktiivisessa kehityksessä ja se on interne-

tin alkuajoista lähtien aina tähän päivään saakka ollut yksi suosituimmista web-palvelinohjelmistoista [31]. Pääosin juuri siksi päädyin käyttämään Apachea tämän tutkielman testiympäristöissä.

Apache on saatavilla useiden käyttöjärjestelmäjakeluiden mukana ja löytyy siten myös Ubuntun jakeluversiosta, josta se asennetaan testivirtuaalikoneille. Apache on modulaarinen ohjelmisto ja siihen on saatavilla kattava kirjo erilaisia lisäominaisuuksia tarjoavia moduuleja. Testiympäristöjen asennuksen yhteydessä Apacheen asennetaan oletusarvoisten lisäosien lisäksi mod_php -moduuli PHP-lähdekooditiedostojen suorittamista varten. Kaikki liikenne käyttäjän ja palvelimen välillä tapahtuu HTTPS-yhteyden välityksellä SSL-salattuna. SSL-toiminnallisuus toteutetaan Ubuntun Apache-asennukseen oletusarvoisesti sisältyvän mod_ssl -moduulin toimesta.

Apachen ja PHP-koodia suorittavan mod_php -moduulin käyttö on varsin tavanomainen ja yleinen ratkaisu Wordpress-pohjaisen palvelun ylläpidossa. Se on siten myös hyvin tuettu ja edelleen aktiivisesti kehittyvä ohjelmistoratkaisu. Apache on testipalvelimella jatkuvasti käynnissä käyttöjärjestelmän palveluna, jonka myötä se saadaan käynnistymään automaattisesti palvelimen uudelleenkäynnistyksen yhteydessä. Apache kuuntelee oletusarvoista HTTPS-liikenteelle tarkoitettua TCP-porttia 443. Apachen toimintaa rajoittamassa ei tässä testissä ole minkäänlaista palomuuria, koska kaikki testaukset suoritetaan suljetussa lähiverkossa, jossa ei ole muita käyttäjiä.

3.4.4 Tietokantapalvelin

Koska Wordpressin toiminta nojaa vahvasti tietokannassa säilytettäviin tietoihin, verkkopalvelun ylläpitämiseen tarvitaan web-palvelimen lisäksi myös tietokantapalvelin. Tietokantapalvelimia on saatavilla suuri määrä sekä kaupallisina, että ilmaisina avoimen lähdekoodin sovelluksina. Tässä tutkielmassa tietokantapalvelimena käytetään avoimen lähdekoodin MariaDB-tietokantaa [29]. Apachen tapaan myös MariaDB on laajasti käytetty ja saatavilla suoraan käyttöjärjestelmäjakeluiden mukana. PHP-ohjelmointikielestä löytyy valmiit funktiot tietokantayhteyksien muodostamiseen, joita Wordpress käyttää osana omia koodejaan.

Wordpress-verkkopalvelun kaikki tekstimuotoinen sisältö ja asetukset on tallennettu tietokantaan, joten luotettavasti toimiva tietokantapalvelin on elinehto nopeasti toimivalle verkkopalvelulle. Toteutuksesta riippuen tietokantapalvelinta voidaan ajaa samalla palvelimella web-palvelimen kanssa tai vaihtoehtoisesti erillisel-

lä palvelimella. Yhdistetyn web- ja tietokantapalvelimen käyttö on yleistä etenkin pienten sivustojen kanssa, koska se on erillisiä palvelimia edullisempi ratkaisu. Varjopuolena tällaisessa toteutuksessa on se, että palvelinresurssit ovat jaettuja, jolloin web-palvelimen ylikuormittuminen voi pahimmassa tapauksessa estää tietokantapalvelimen toiminnan kokonaan.

Varsinkin suurempia käyttäjämääriä palvelevien verkkopalveluiden kanssa kannattaa käyttää erillisiä web- ja tietokantapalvelimia, jotta palvelimiin kohdistuvaa kuormaa saadaan jaettua tasaisemmin. Toisistaan eriytetyt web- ja tietokantapalvelimet mahdollistavat palvelimien paremman skaalattavuuden ja tarvittaessa myös rinnakkaistamisen, jolloin useampia keskenään replikoituja palvelininstansseja voidaan valjastaa suorittamaan samaa tehtävää. Useampaa rinnakkaista tietokantapalvelinta käytettäessä tarvitaan lisäksi myös kuormantasaaja tarjoamaan yhteysosoite tietokantaan ja ohjaamaan liikennettä tietokantapalvelimille. Vaihtoehtoisesti Wordpressiin voidaan asentaa HyperDB-lisäosa [20], jonka avulla tietokantakyselyitä ja kirjoituksia voidaan jakaa useammalle tietokantapalvelimelle.

3.4.5 Tiedostojärjestelmä

Tiedostojärjestelmä on fyysisen tallennusmedian eli tässä tapauksessa kovalevyn tai useammasta kovalevystä koostuvan levyjärjestelmän päälle luotu kerros, jonka avulla palvelimella toimivat ohjelmistot voivat lukea ja kirjoittaa tiedostoja. Tiedostojärjestelmään tallennetaan tiedostojen ja hakemistojen metatiedot, kuten nimi, koko, luku-, kirjoitus- ja suoritusoikeudet sekä aikaleimat, joista selviää luontiaika, viimeisimmän muutoksen ajankohta ja viimeisin käyttöajankohta.

Tätä nykyä yleisimmin käytetyt tiedostojärjestelmät Linux-käyttöjärjestelmissä ovat EXT4 [32] ja XFS [14]. EXT4 (fourth EXTended filesystem) on Linux-käyttöjärjestelmään kuuluva neljännen sukupolven tiedostojärjestelmä, joka julkaistiin alunperin vuonna 2008 [32]. Se on oletusarvoinen käyttöjärjestelmä Debian- ja Ubuntupohjaisissa Linux-käyttöjärjestelmissä. EXT4-tiedostojärjestelmä on kehitetty aiemman EXT3:n seuraajaksi ja suunniteltu parantamaan suorituskykyä ja luotettavuutta.

Käyttöönotto on helppoa, sillä EXT4 on yhteensopiva ext3:n kanssa ja aiempi tiedostojärjestelmä on mahdollista konvertoida uuden version mukaiseen formaattiin. EXT4 liitettiin osaksi Linuxin kerneliä heti julkaisuvuoden 2008 aikana.

XFS-tiedostojärjestelmä ensimmäinen versio julkaistiin suljettuna versiona Silicon Graphicsin toimesta alunperin jo vuonna 1993 ja sitä käytettiin IRIX-käyttöjär-

jestelmän oletusarvoisena käyttöjärjestelmänä XFS [14]. Avoimen lähdekoodin versio julkaistiin vuonna 1999 ja se liitettiin osaksi Linux-käyttöjärjestelmää vuonna 2001. XFS on nykyään oletusarvoinen tiedostojärjestelmä Red Hat- ja CentOS-pohjaisissa käyttöjärjestelmissä. Se on hyvä valinta tiedostojärjestelmäksi suurten tiedostojen tallennukseen ja skaalautuu hyvin soveltuen varsinkin erittäin suurta tallennuskapasiteettia edellyttäviin tarpeisiin kuten varmuuskopiointiin.

3.4.6 Verkkoyhteys

Verkkopalvelua tarjoava palvelin on liitettävä internetiin tai sisäisessä käytössä suljettuun yksityisverkkoon, jotta käyttäjien on mahdollista ottaa siihen yhteyttä. Verkkoyhteys ja sen toimivuus on yksi keskeinen osa rakennettaessa nopeasti ja luotettavasti toimivaa verkkopalvelua. Varsinkin internetin alkuaikoina verkkoyhteyden merkitys verkkopalvelun toimivuudessa oli korostuneempi sillä yhteydet eivät tuolloin olleet niin luotettavia kuin tänä päivänä. Nykyään kaupallisessa käytössä toimivat palvelimet on liitetty internetiin lähes poikkeuksetta nopeiden kuituyhteyksien avulla ja ne toimivat varsin luotettavasti. Suurin yhteyksien toimintaa heikentävä uhka ovat mahdolliset palvelunestohyökkäykset.

Palvelinyhteyksien ohella merkittävä osatekijä verkkopalvelun latausnopeudessa on käyttäjän oma verkkoyhteys ja sen toimivuus. Tilanteesta riippuen käyttäjän päätelaite voi käyttää palvelimista tuttua kiinteää kuituyhteyttä, naapurin tai nettikahvilan suojaamatonta wifi-verkkoa, mobiiliverkkoa tai vaikkapa satelliitin välityksellä toteutettua internetyhteyttä. Erinäisistä tekijöistä johtuen verkkoyhteydet voivat toimia moitteettomasti ja nopeasti, pätkien ja tökkien tai olla poikki kokonaan. Tässä tilanteessa nopeimmastakaan palvelimesta ei ole mitään, jos käyttäjän oma yhteys ei toimi. Verkkoyhteys on sinänsä sovellusympäristön ulkopuolinen osa, eikä sitä käsitellä tässä tutkielmassa sen syvällisemmin.

3.4.7 Käyttäjän päätelaite

Käyttäjän päätelaite ei varsinaisesti liity sovellusympäristöön, mutta myös sen toiminnalla on olennainen merkitys verkkopalvelun latausnopeuteen ja yleiseen toimivuuteen. Nykyisellään suuri osa verkkopalveluiden käyttäjistä käyttää mobiililaitteita, kuten matkapuhelinta tai tablettia. Näiden laitteiden kirjo on suuri ja eri laitesukupolvia useita, joten jo pelkästään päätelaitteiden suorituskyvyn ja ominaisuuksien eroavaisuudet aiheuttavat merkittävää hajontaa verkkopalveluiden toiminta-

nopeuteen ja käyttäjäkokemukseen.

Uudet laitteet ajantasaisella ohjelmistolla varustettuina toimivat pääosin hyvin. Vanhempia laitteita on kuitenkin käytössä hyvin laajalti ja niiden suorituskyky sekä vanhentuneet ohjelmistot voivat aiheuttaa ongelmia palveluita käytettäessä. Verkkopalveluiden tuottajat joutuvatkin tasapainoilemaan viimeisimpien tekniikoiden ja ominaisuuksien käytön ja toisaalta taaksepäin yhteensopivuuden kanssa, jotta palvelut ovat myös vanhempien laitteiden käyttäjien ulottuvilla.

Loppujen lopuksi ei ole kenenkään etu, että verkkopalvelu toimii hitaasti tai pahimmassa sen käyttö estyy kokonaan epäkelvon päätelaitteen vuoksi. Tällaisessa tilanteessa palvelun käyttäjä päätyy usein ajattelemaan, että ongelman aiheuttaja on kyseinen palvelu, vaikka todellinen syy olisikin käyttäjän oma, sekä laite- että ohjelmistokomponenttien osalta vanhentunut puhelin tai tabletti.

4 Tallennusjärjestelmä

Tässä luvussa käsitellään eri tyyppisiä kovalevyjä ja tallennusjärjestelmiä, kovalevyjen historiaa ja kehitystä. Tallennusjärjestelmä on olennainen osa palvelinlaitteen ympärille nivoutuvassa kokonaisuudessa. Tässä tapauksessa se voi tarkoittaa joko palvelimen sisäistä, yhtä tai useampaa kovalevyä tai erillisestä levyjärjestelmästä tarjottavaa tallennuskapasiteettia. Olennaista kuitenkin on, että se tarjoaa mahdollisuuden pysyväistallennukseen mahdollistaen palvelimen käyttöjärjestelmän, sovellusten ja niiden käsittelemän datan luotettavan tallennuksen.

Tämä luku sisältää neljä alilukua, joista aliluku 4.1 käsittelee perinteistä magneettista pyörivää kovalevyä esitellen niiden historiaa, tekniikkaa ja nykytilaa. Aliluku 4.2 keskittyy tuoreempiin SSD- ja NVMe-kovalevytyyppeihin ja edellisen alilu-vun tapaan kertoo näiden kovalevytyyppien historiasta, tekniikasta ja nykypäivän asemasta. Aliluku 4.3 kertoo levyjärjestelmistä ja aliluku 4.4 ramdiskistä ja heittovaihtotiedostosta.

4.1 Magneettinen (pyörivä) kovalevy

Magneettisuuteen perustuvan pyörivän kovalevyn historia ulottuu aina vuoteen 1957 saakka, jolloin IBM esitteli ensimmäisen kovalevyn. Ensimmäinen kovalevy oli kooltaan lähes kahden kuutiometrin kokoinen [10]. Tekniikan kehittyessä kovalevyjen fyysinen koko pieneni ja tallennuskapasiteetti kasvoi. Yleisimmät kovalevyissä tänä päivänä käytettävät koot ovat 2,5″ ja 3,5″. Aiemmin käytössä on ollut myös 8″:n, 5,25″:n ja 1,8″:n kokoisia laitteita. Kovalevyjen koosta puhuttaessa on huomioitava, että koolla on alun perin tarkoitettu kovalevyjen sisällä olevien metalli- tai lasikiekkojen halkaisijaa, jolloin kotelon leveys ja korkeus ovat ilmoitettua mittaa suurempia [40]. Myöhemmin SSD-levyjen vallatessa markkinoita niiden valmistuksessa päädyttiin käyttämään pääosin samoja ulkomittoja kuin pyörivien kovalevyjen kanssa, jotta levyjä voitaisiin asentaa tietokonekoteloissa pyöriville kovalevyille ja levyasemille tarkoitettuihin paikkoihin.

Pyörivien kovalevyjen kierrosnopeudet ovat tyypillisesti 7200-15000 kierrosta minuutissa. Tiedon lukeminen tapahtuu pienen lukupään avulla, joka liikkuu mag-

neettisen pinnan yläpuolella [10]. Lukupää mittaa magneettikentän muutoksia ja tulkitsee ne binäärimuotoiseksi dataksi. Vastaavasti tiedon tallennus tapahtuu magnetisoimalla pieniä alueita vastaamaan binäärikoodia 1 tai 0. Sekä luku- että kirjoitusoperaatiot tapahtuvat samalla lukupäällä.

Pyörivä, tyypillisesti 3,5”:n kokoinen kovalevy (Hard Disk Drive, HDD), oli pitkään PC-tietokoneiden pääasiallinen tallennusmedia. Sittemmin liikkuvia osia sisältämättömät SSD- ja NVMe-kovalevyt ovat pääosin syrjäyttäneet pyörivät kovalevyt. Pyöriviä kovalevyjä käytetään nykyään lähinnä varmuuskopiointi- ja arkistointikäytössä silloin, kun tarvitaan erittäin suurta tallennuskapasiteettia. Pyöriviä levyjä on lisäksi edelleen käytössä suuria määriä vanhoissa järjestelmissä, joten uusia levyjä tarvitaan myös varaosiksi käytössä rikkoutuneiden tilalle.

4.2 SSD- ja NVMe-kovalevyt

SSD-kovalevyjen historia ulottuu pyörivien kovalevyjen tapaan aina 1950-luvulle saakka [47]. Pyörivät levyt olivat kuitenkin käytännössä ainoa käytössä oleva kovalevytyyppi aina 2000-luvun taitteeseen saakka ja kuluttajatason laitteissa SSD-levyt alkoivat yleistymään vuoden 2010 paikkeilla. SSD-kovalevy on suorituskyvyltään ylivertainen pyörivään kovalevyyn verrattuna, joten levyjen yleistyminen kuluttajatason tietokoneissa nopeutti laitteiden toimintaa merkittävästi. Toimintanopeuden lisäksi toinen selkeä parannus pyöriviin levyihin nähden on huomattavasti parempi tärinänkesto, joka on erityisen hyvä asia varsinkin kannettavien koneiden kanssa.

SSD-levyjen toimintaperiaate eroaa merkittävästi pyörivistä kovalevyistä, sillä levyt eivät sisälly liikkuvia osia. Tieto tallennetaan flash-muistipiireihin, jotka sisältävät soluja, joita voidaan lukea sähköisten varausten avulla [17]. Kun tietoa halutaan lukea SSD-levyltä, ohjainpiiri lähettää signaaleja oikeisiin soluihin, jotka sisältävät tarvittavan tiedon, joka tulkataan edelleen binääriseksi tiedoksi. Prosessi on nopea, koska se ei, toisin kuin pyörivissä kovalevyissä, vaadi liikkuvia osia. Kirjoitettaessa ohjainpiiri valitsee solut joihin tieto tallennetaan ja muuttaa niiden sähkövarausta vastaamaan haluttua bittikoodia. Kirjoittamisen myötä solujen sähkövaraus muuttuu pysyvästi, jotta tiedot säilyvät myös virran ollessa katkaistuna. Lukuoperaation tapaan myös kirjoitusoperaatio on nopea, koska se ei vaadi fyysistä liikettä eikä mekaanista toimintaa.

SSD-levyjen seuraajaksi kehitettyjen NVMe-levyjen ensimmäinen versio julkaistiin vuonna 2008, mutta niiden yleistyminen otti oman aikansa. NVMe-levyt yleis-

tyivät hiljalleen ja ne ovat olleet laajassa käytössä noin vuoden 2015 paikkeilta saakka [1]. Aiempiin SSD-levyihin verrattuna NVMe-levyt ovat huomattavasti suorituskykyisempiä ja lisäksi fyysisiltä mitoiltaan pienikokoisempia.

NVMe-liitäntä tarjoaa huomattavasti suuremman tiedonsiirtonopeuden perinteiseen SATA-liitäntäiseen SSD-levyyn nähden [22]. Tämä johtuu siitä, että NVMe-protokolla on suunniteltu toimimaan PCIe-väylän kautta, mikä tarjoaa huomattavasti suuremman kaistanleveyden ja vähäisemmän viiveen kuin perinteinen SATA-liitäntä. Luku- ja kirjoitusoperaatiot tapahtuvat samaan tapaan SSD-levyissä. Pelkkä levy itsessään ei sinänsä ole nopeampi kuin SSD-levy, vaan nopeusero aiheutuu nimen omaan paremmasta väylästä ja liitettävyydestä emolevyyn ja edelleen prosessoriin.

4.3 Levyjärjestelmä

Levyjärjestelmästä puhuttaessa tarkoitetaan itsenäistä, palvelinlaitteesta erillistä laitetta tai useammasta laitteesta koostuvaa kokonaisuutta, joka tarjoaa tallennuskapasiteettia yhdelle tai useammalle palvelimelle [44]. Levyjärjestelmät koostuu tyypillisesti räkkiasennettavasta laitteesta, joka sisältää levyohjaimet sekä muut järjestelmälle yhteiset komponentit, kuten liitettävyyden tarjoavat kytkimet. Sen yhteyteen voidaan liittää levyhyllyjä, eli käytännössä räkkiasennettavia kehikoita, jotka kalustetaan tarpeen mukaan halutulla määrällä kovalevyjä. Levyjärjestelmän tallennuskapasiteetista voidaan luoda useamman eri kovalevyn tallennustilaa hyödyntäviä loogisia yksiköitä halutulla RAID-tasolla tarvittavan vikasietoisuuden tai tallennuskapasiteetin saavuttamiseksi.

Levyjärjestelmä liitetään sitä hyödyntäviin palvelinlaitteisiin tyypillisesti joko ethernet- tai SAN-verkon välityksellä. Käytettävät palvelinlaitteet voivat toimia kokonaan ilman omia kovalevyjä ja tarjota tallennuskapasiteetti kokonaisuudessaan levyjärjestelmästä tai varustaa palvelimet pienillä kovalevyillä, jotka tarjoavat tallennuskapasiteetin vain käyttöjärjestelmälle sekä mahdolliselle heittovaihtopartitiolle ja tarjota varsinainen tallennuskapasiteetti levyjärjestelmästä. Levyjärjestelmän käyttö on perusteltua ja järkevää myös monessa muussa eri käyttötarkoituksessa. Sen avulla voidaan sama, useamman fyysisen kovalevyn päälle rakentuva, looginen levy tuoda samanaikaisesti useamman palvelinlaitteen käyttöön, jolloin esimerkiksi klusteroitujen virtualisointijärjestelmien päällä ajettavien virtuaalikoneiden ajonaikainen siirto toiselle fyysiselle palvelimelle helpottuu ja nopeutuu mer-

kittävästi. Varmuuskopiointitarkoituksissa levyjärjestelmän edut ovat kiistattomat, sillä sen avulla saadaan edullisesti tuotettua suuri tallennuskapasiteetti pyöriviä levyjä käyttämällä.

4.4 Ramdisk ja heittovaihtotiedosto

Ramdisk on Linux-käyttöjärjestelmäytimeen sisältyvä ominaisuus, joka mahdollistaa keskusmuistin käytön tavallisen kovalevyn tapaan [24]. Huomioitavaa tässä on kuitenkin se, että ramdisk-pohjaisen kovalevyn sisältö menetetään, kun tietokoneesta katkaistaan virta. Näin ollen ramdiskiä ei voi käyttää pysyväistallennukseen, mutta toisaalta se soveltuu hyvin lyhyt- ja väliaikaiseen tallennustarpeeseen tarjoten kovalevyihin nähden erittäin suuren suorituskyvyn.

Heittovaihtotiedosto toimii keskusmuistin jatkeena ja sitä käytetään tyypillisesti tarjoamaan ylimääräistä keskusmuistikapasiteettia järjestelmän muistinkäytön ylittäessä keskusmuistin määrän esimerkiksi ajastettujen ajojen tai muiden kuormittavien tehtävien suorittamisen aikana [28]. Heittovaihtotiedosto sijaitsee kovalevyllä, joten sen toimintanopeus ja suorituskyky on tyypillisesti huomattavasti keskusmuistia heikompi. Palvelimen keskusmuistin määrä tulisikin mitoittaa siten, että heittovaihtotiedostoa ei ainakaan merkittävissä määrin tarvitse koskaan käyttää.

5 Suorituskykymittaukset

Tässä luvussa käsitellään tutkielman suorituskykymittausten käytännön toteutus ja tulokset. Testiympäristön valmistelu aloitetaan perustamalla testivirtuaalikoneet automatisoidusti Terraform-nimisen infrastruktuuri-koodina (Infrastructure as Code, IaC) -ohjelmiston [18] ja Ansible-automaatI/O-ohjelmiston [34] avulla Proxmox-isäntäkoneille. Testit suoritetaan ja testitulosten pohjalta koostetaan taulukoita ja graafeja testitulosten havainnollistamiseksi.

Aliluku 5.1 taustoittaa suorituskykymittauksia yleisellä tasolla. Siinä kerrotaan suorituskykymittausten sisällöstä ja rakenteesta sekä taustoitetaan mittaustulosten kannalta oleellisia asioita. Aliluku 5.2 keskittyy käsittelemään testiympäristön teknistä toteutusta ja siinä kerrotaan tarkemmin minkälaisilla testivirtuaalikoneilla testi suoritetaan. Aliluvussa 5.3 käsitellään varsinaiset suorituskykymittaukset tuloksineen ja alivussa 5.4 suorituskykymittausten yhteenveto.

5.1 Yleistä

Suorituskykymittausten tarkoituksena on selvittää kuinka suurta kuormaa eri tasoisilla kovalevyillä varustetut virtuaalikoneet kestävät ilman toiminnan hidastumista ja vertailla tuloksia keskenään. Koska palvelin toimii aina rajallisilla resursseilla, se pystyy suoriutumaan samanaikaisten käyttäjien palvelemisesta tiettyyn pisteeseen saakka, jonka jälkeen toiminta alkaa hidastua. Hidastuminen voi johtua monesta eri syystä.

Ongelmaa lähestytään perustamalla kolme eri tasoista Linux-virtuaalikonetta, jotka sisältävät 16 gigatavua keskusmuistia, kaksi, neljä tai kahdeksan suoritinydintä, ja joiden virtuaalikovalevyt on rajoitettu eri suorituskykytasoille. Testipalvelimeen asennetaan web- ja tietokantapalvelin sekä Wordpress-julkaisujärjestelmä. Testiympäristöissä Apachen ja MariaDB:n asetukset pidetään oletusarvoissaan sillä poikkeuksella, että molempien sovellusten maksimiyhteysmäärää kasvatetaan, jotta käyttäjämäärän kasvatus enimmillään 300 käyttäjään olisi mahdollista. Wordpress asennetaan virallisesta asennuspaketista ja asennuksen sisällöksi tuodaan WP Test -testisisältö [37], joka vastaa tyypillistä Wordpress-verkkopalvelua ja on varta vasten

kehitetty testitarkoituksiin. Virtuaalikoneiden kovalevyille asetettavien rajoitusten avulla pyritään jäljittelemään eri tyyppisten fyysisten kovalevyjen (HDD, SSD, NVMe) suorituskyvyn tasoa. Myös fyysisten testitietokoneiden ja -kovalevyjen hankintaa harkittiin, mutta se todettiin taloudellisesti liian suureksi investoinniksi varsinkin, kun virtualisoiduilla ympäristöillä saadaan melko tarkasti jäljiteltyä fyysisten laitteiden toimintaa.

Kaikki suorituskykymittaukset suoritetaan kuormittamalla testivirtuaalikoneille asennettua Wordpress-julkaisujärjestelmään perustuvaa verkkopalvelua Locust-kuormitustestausohjelmistolla [45]. Testivirtuaalikoneiden Wordpress-asennuksiin on ladattu sama WP Test [37] -testisisältö. Suoritettavia testejä on kaksi erilaista, jotka molemmat sisältävät kaksi vaihetta. Ensimmäisessä testissä testiympäristön kaikki välimuistiratkaisut pyritään eliminoimaan, koska tarkoituksena on tutkia kovalevyn suorituskyvyn vaikutusta julkaisujärjestelmän toimintanopeuteen. Käytännössä Linux-käyttöjärjestelmän Page Cache -välimuistia [58] ei voi kytkeä pois käytöstä, joten sitä täytyy tyhjentää aina testikierroksen lopussa ennen uuden kierroksen aloittamista. Toisessa vaiheessa ladataan kirjoittamani PHP-skripti fstat.php, joka kuvaillaan tarkemmin suorituskykymittausten toteutuksesta kertovassa luvussa. Skripti lukee levyltä Wordpress-verkkopalvelun etusivun latauksessa tarvittavat tiedostot niitä kuitenkaan suorittamatta. Näin saadaan selville pelkästään tiedostojen lukemiseen kuluva aika. Toisessa testissä välimuisti pidetään normaalisti käytössä ja havainnoidaan käyttäjämäärän lisääntymisen vaikutusta latausaikaan.

Mittauksissa palvelun etusivua ladataan säännöllisin väliajoin ja mitataan vastauksiin kuluva aika. Ensimmäisen mittauksen osalta testi tehdään vain yhdellä käyttäjällä, sillä välimuistin tyhjennyksen jälkeen täytyy olla varma siitä, että mahdollisen toisen rinnakkaisen sivunlatauksen toimesta välimuistiin päätyneiden tiedostojen nopeat hakuajat eivät sotke testituloksia. Toisen testin osalta Locustin parametrejä muuttamalla simuloitua käyttäjämäärää voidaan kasvattaa halutuin askelin. Jotta kunkin testivirtuaalikoneen suorituskyvyn vaikutus latausaikoihin kuormituksen noustessa saadaan selville, Locustin simuloitua käyttäjämäärää kasvatetaan alun yhdestä käyttäjästä aina 300 käyttäjään saakka.

Jotta suoritus- ja vastausaikoja saadaan mitattua luotettavasti, täytyy kuormitustestien toistokertoja olla paljon, sillä kasvatettaessa käyttäjämäärää kuormitustestin aikana palvelin joutuu käsittelemään useita samanaikaisia latauspyyntöjä rinnakkain. Käyttöjärjestelmä ja ohjelmakoodia suorittavat suoritinytimet käsittelevät pyyntöjä erinäisten jonojen kautta, jolloin kahden yksittäisen kuormitustestin tu-

lokset eivät ole keskenään täysin samanlaisia. Riittävällä toistolla voidaan kuitenkin saavuttaa hyvä ja luotettava tulos. Riittäväksi toistoksi päätettiin 100 latauskertaa eli ilman välimuistia suoritettavia latauksia tullaan yhden testin aikana suorittamaan 100 kertaa, jolloin mahdollisista satunnaisvaihteluista päästään eroon. Koko testi voidaan myös tarvittaessa uusia, mikäli epäilyttäviä poikkeamia ilmenee.

Edellä mainittujen seikkojen vuoksi minimi- tai maksimilatausaikoja ei käytetä mittarina, koska ne eivät kuvaa suorituskyvyn perustasoa. Myöskään latausaikojen keskiarvon käyttö ei palvele tarkoitusta parhaalla mahdollisella tavalla, sillä latausaikojen jakaumat ovat usein vinoja ja yksittäiset poikkeamat saattavat vääristää tulosta liiaksi [50]. Näin ollen mediaani on keskiarvoa parempi suure kuvaamaan toistokerroista koostettua suorituskykymittauksen tulosta. Jos esimerkiksi viiden latauskerran vastausajat ovat 0,5, 0,6, 0,5, 1,8 ja 0,7 sekuntia, on keskiarvo 0,82 ja mediaani 0,6 sekuntia. Tästä voidaan huomata, että yksittäinen pitkittynyt 1,8 sekunnin latausaika ei sekoita mediaanitulosta liiaksi, mutta näkyy keskiarvossa turhan paljon tulosta nostavana ja vääristävänä tekijänä.

Mittaustulosten perusteella luodaan kaavioita, joista voidaan havainnoida ilman välimuistia suoritettavien testien latausaikoja ja välimuistin kanssa suoritettavien testien osalta latausajan lisäksi myös käyttäjämäärän vaikutusta verkkopalvelun toimintanopeuteen. Kunkin testivirtuaalikoneen testitulosten pohjalta luodaan kaaviot, joista käy ilmi yksittäisen koneen tulokset. Sen lisäksi luodaan kaaviot, joihin on yhdistetty kaikkien kolmen testivirtuaalikoneen testitulokset rinnakkain, jolloin niistä on helppoa tarkastella eri tasoisten ympäristöjen suorituskyvyn ja toimintanopeuden eroavaisuuksia.

5.2 Testiympäristön tekninen kuvaus

Testiympäristöä suunniteltaessa toteutuksen lähtökohdiksi otettiin tulevan ympäristön edullisuus, tarvittavien laite- ja ohjelmistokomponenttien hyvä saatavuus ja suoritettavien testien helppo toistettavuus. Erityisesti suorituskykymittausten toistettavuus ja testiympäristöjen nopea uudelleenasennus olivat ehdottomia vaatimuksia, koska samoja testejä tultaisiin suorittamaan kerta toisensa jälkeen uudelleen ja uudelleen. Lisäksi mahdollisten konfiguraatI/O- ja kokoonpanomuutosten jälkeen tarvittavien asennusten tulisi olla helppoja toteuttaa.

Testiympäristön tekninen kuvaus on jaettu neljään eri alilukuun. Aliluvussa 5.2.1 kuvataan suorituskykymittauksissa käytettyjen virtuaaliympäristöjen kokoonpano

ja käytettävät ohjelmistot. Aliluvussa 5.2.2 paneudutaan tarkemmin suorituskykymittausten toteutukseen ja niiden sisältöön. Aliluku 5.2.3 sisältää magneettisen kovalevyn esittelyn ja suorituskykyyn vaikuttavat parametrit. Aliluvussa 5.2.4 käsitellään vastaavat SSD- ja NVMe-kovalevyihin liittyvät asiat.

5.2.1 Käytetty virtuaaliympäristö

Yhdistettyinä web- ja tietokantapalvelimina toimivien virtuaalikoneiden kokoonpanoa määriteltäessä peruslähtökohdaksi otettiin se, että ympäristöjen täytyy vastata tallennusmedian suorituskyvyn osalta sellaisia palvelimia, joita Wordpress-pohjaisten verkkopalveluiden ylläpidossa yleisesti käytetään. Pilvipalveluiden myötä virtuaalikoneet ovat hyvin laajasti käytössä tänä päivänä, joten niitä päädyttiin käyttämään myös testipalvelimien alustana. Virtuaalikoneilla toteutettu testiympäristö on myös huomattavan paljon edullisesti ja vaivattomampi tapa testiympäristön pystyttämiseen kuin vastaavanlaisten fyysisten palvelimien hankkiminen ja asennus käyttökuntoon.

Testiympäristöjen isäntäkoneen käyttöjärjestelmänä toimii Proxmox VE (Virtualization Environment) [33] -virtualisointiohjelmisto, joka perustuu Debian GNU/Linuxiin [11]. Se sisältää hypervisor-toiminnallisuuden KVM- ja QEMU-ohjelmistoineen valmiiksi asennettuna. Ohjelmisto sisältää myös API-rajapinnan, jonka ansiosta palvelimien asennus ja konfigurointi voidaan tehdä Terraform- ja Ansible-ohjelmistojen avulla. Ympäristöjen perustaminen, asennus ja konfigurointi tapahtuu automatisoidusti Github CI/CD -julkaisuputkessa, jolloin ympäristöjen perustaminen ja poistaminen onnistuu vaivattomasti käden käänteessä.

Testiympäristöjen käyttöjärjestelmäksi valittiin Ubuntu 22.04 LTS -Linux-käyttöjärjestelmä. Kaikkiin testivirtuaalikoneisiin päätettiin allokoida 16 gigatavua keskusmuistia, joka on tällaisen testipalvelimen kaltaiseen yhdistettyyn web- ja tietokantapalvelimeen hyvinkin riittävä määrä. Riittävällä keskusmuistikapasiteetilla varmistetaan, että muisti riittää kaikissa tilanteissa, eikä muistin mahdollinen loppuminen pääse vääristämään mittaustuloksia tai aiheuttamaan muita häiriöitä testejä suoritettaessa.

Virtuaalikoneisiin liitettävien virtuaalikovalevyjen osalta päätettiin käyttää levyjä, joiden levykuva on sijoitettu NVMe-levylle. Kovalevyjen suorituskykyä päätettiin rajoittaa ohjelmallisesti kolmeen eri suorituskykyluokkaan siten, että ensimmäisen testivirtuaalikoneen levy rajoitettiin hitaimmaksi ja vastaamaan suorituskyvyltään perinteistä pyörivää 3,5” kovalevyä (100 IOPS) [25]. Toisen testivirtuaalikoneen

levyn suorituskyky rajoitettiin vastaamaan edullista kuluttajatason SSD-kovalevyä (20000 IOPS) [25]. Kolmannen testivirtuaalikoneen kovalevyn suorituskyky rajoitettiin edullisen kuluttajatason NVMe-levyn tasolle (200000 IOPS).

Suoritinkapasiteetin osalta testivirtuaalikoneisiin päätettiin allokoida kaksi, neljä tai kahdeksan suoritinydintä. Hitaimmalla virtuaalikovalevyllä eli suorituskyvyltään perinteistä pyörivää 3,5" kovalevyä (100 IOPS) vastaavalla levyllä varustettu testivirtuaalikone varustettiin kahdella suoritinytimellä. Toinen edullista kuluttajatason SSD-kovalevyä (20000 IOPS) vastaavalla virtuaalikovalevyllä varustettu virtuaalikone varustettiin neljällä suoritinytimellä ja kolmannen edullisen kuluttajatason NVMe-levyn tasoa (200000 IOPS) vastaavalla virtuaalikovalevyllä varustettu virtuaalikone kahdeksalla suoritinytimellä.

5.2.2 Suorituskykymittausten toteutus

Ensimmäisessä testissä testiympäristön kaikki välimuistiratkaisut pyritään eliminoimaan, koska tässä testissä on tarkoituksena tutkia kovalevyn suorituskyvyn vaikutusta julkaisujärjestelmän toimintanopeuteen, eikä niinkään välimuistin toimintanopeutta toistuvien latausten yhteydessä. Ensimmäisen testin mittausten osalta testi tehdään vain yhdellä käyttäjällä, sillä välimuistin tyhjennyksen jälkeen täytyy olla varma siitä, että mahdollisen toisen rinnakkaisen sivunlatauksen toimesta välimuistiin päätyneiden tiedostojen nopeat hakuajat eivät sotke testituloksia.

Ennen varsinaisen testin aloittamista suoritetaan tätä tarkoitusta varten kirjoittamani PHP-skripti drop.php, joka tyhjentää Page Cachen [58] varmuuden vuoksi. Drop.php:n suorituksen jälkeen odotetaan muutama sekunti, jotta testipalvelimen välimuisti on varmasti tyhjentynyt. Tämän jälkeen aloitetaan varsinainen testi, joka suoritetaan Locustin toimesta Python-skriptin (locustfile.py) pohjalta.

Testissä ladataan Wordpress-verkkopalvelun etusivu ja mitataan sen latausaika. Sitten välimuisti tyhjennetään uudelleen, odotetaan muutama sekunti ja ladataan PHP-skripti fstat.php, joka lukee fstat.txt-tekstitiedostosta listauksen tiedostoista, jotka Wordpress-verkkopalvelun etusivun latauksessa ladataan. Tämän jälkeen tiedostot etsitään ja luetaan kovalevyltä, mutta Wordpressin PHP-koodia ei kuitenkaan suoriteta, joten näin saadaan selville pelkästään tiedostojen lukemiseen kuluva aika. Lopuksi välimuisti tyhjennetään vielä uudemman kerran, jotta uusi testikierros voidaan aloittaa puhtaalta pöydältä tyhjällä välimuistilla. Sama testi toistetaan kullakin testivirtuaalikoneella sata kertaa peräkkäin, jotta mahdollista latausaikojen vaihtelua pystytään havainnoimaan mahdollisimman kattavasti.

Toisessa testissä välimuisti pidetään normaalisti käytössä ja havainnoidaan käyttäjämäärän lisääntymisen vaikutusta latausaikaan. Toisen testin kulku on muuten sama kuin ensimmäisessä vaiheessa, mutta välimuistia ei tyhjennetä ollenkaan. Ensin latadaan Wordpress-verkkopalvelun etusivu ja mitataan sen latausaika. Seuraavaksi ladataan PHP-skripti fstat.php, joka lukee fstat.txt-tekstitiedostosta listauksen tiedostoista, jotka Wordpress-verkkopalvelun etusivun latauksessa ladataan. Tämän jälkeen tiedostot etsitään ja luetaan kovalevyltä, mutta Wordpressin PHP-koodia ei kuitenkaan suoriteta, joten näin saadaan selville pelkästään tiedostojen lukemiseen kuluva aika. Testi suoritetaan siten, että Locustin simuloitua käyttäjämäärää kasvatetaan alun yhdestä käyttäjästä aina 300 käyttäjään saakka, jotta kunkin testivirtuaalikoneen suorituskyvyn vaikutus latausaikoihin kuormituksen noustessa saadaan selville.

Välimuistin tyhjentämisen suorittavan drop.php-skriptin lähdekoodi:

```
<?php
shell_exec("sudo screen -dmS drop_caches bash -c 'sync && \
echo 3 > /proc/sys/vm/drop_caches'");
echo "OK\n";
?>
```

Fstat.php-skriptin lähdekoodi:

```
<?php

$time_start = hrtime(true);

if ($file = fopen("fstat.txt", "r")) {
    while(!feof($file)) {
        $line = fgets($file);
        if(is_file($line)){
          $handle = fopen($line, "r");
          $fstat = fstat($handle);
          $contents = fread($handle, filesize($line));
          fclose($handle);
        }
    }
    fclose($file);
}

$time_end = hrtime(true);
$time_total = ($time_end - $time_start)/1e+6;

$execution_time = "Execution time for index.php test: " . $time_total . " milliseconds";
echo $execution_time . "\n";
syslog(LOG_WARNING, $execution_time);

?>
```

Locustin käyttämä Python-kielinen locustfile.py, jolla määritellään testin sisältö:

```
import time, logging
from locust import HttpUser, task

class QuickstartUser(HttpUser):
    @task
    def index(self):
        self.client.get("/")
        self.client.get("/drop.php")
        time.sleep(2)
        self.client.get("/fstat.php")
        self.client.get("/drop.php")
        time.sleep(2)
```

5.2.3 Magneettinen (pyörivä) kovalevy

Magneettiseen pyörivään levyyn perustuva kovalevytyyppi. Kuten aiemmin jo mainittiinkin, mittauksissa käytetyn virtuaalikovalevyn IOPS-suorituskyky rajoitetaan 100 tapahtumaan sekuntia kohden. Se vastaa 7200 kierrosta minuutissa pyörivän kuluttajatason 3,5" SATA-kovalevyn suorituskykyä. Koska IOPS-suorituskyky on verrattain heikko, tällaisella kovalevyllä varustetun palvelimen suorituskyky alkaa hidastua jo varsin pienellä kuormituksella ja I/O-pyyntöjä alkaa kasaantua jonoon, jolloin niihin ei enää pystytä vastaamaan nopeasti.

Tämä kovalevytyyppi on tänä päivänä vuonna 2024 jo pitkälti poistunut käytöstä uusien pöytätietokoneiden ja kannettavien tallennusmediana [4]. Pyöriviä kovalevyjä käytetään yleisesti lähinnä varmuuskopiointiin tarkoitetuissa levyjärjestelmissä niiden tarjoaman suuren ja muihin kovalevytyyppeihin nähden edullisen tallennuskapasiteetin vuoksi. Pyöriviä kovalevyjä on kuitenkin edelleen paljon käytössä vanhoissa laitteissa ja varsinkin kotikäytössä olleissa tietokoneissa, joita käytetään myös palvelinkäytössä esimerkiksi opiskelu- ja harrasteprojektien yhteydessä.

5.2.4 SSD- ja NVMe-kovalevyt

Suorituskyvyltään edullista kuluttajatason SSD-kovalevyä (20000 IOPS) vastaavalla virtuaalikovalevyllä varustetun testivirtuaalikoneen pitäisi suoriutua paljon paremmin kuin edellisessä aliluvun 100 IOPS -levyn. Vastaavasti edullisen kuluttajatason NVMe-levyn tasolle (200000 IOPS) rajoitetulla virtuaalikovalevyllä varustetun testivirtuaalikoneen pitäisi suoriutua vielä huomattavasti paremmin.

SSD- ja NVMe-kovalevyt ovat nykytietokoneissa käytettäviä pääasiallisia kovalevytyyppejä. Edullisimmillaan pienen SSD- tai NVMe-kovalevyn hinta on vain muutamia kymmeniä euroja, joten pyörivän kovalevyn päivittäminen uudemman teknologian levyyn on helppo ja edullinen vaihtoehto myös vanhan tietokoneen nopeuttamiseen.

5.3 I/O-operaation suoritusaika

I/O-operaatiolla tarkoitetaan tapahtumaa, jossa tietoa joko luetaan kovalevyltä tai kirjoitetaan kovalevylle. [52] Operaatio tapahtuu käyttöjärjestelmän pyynnöstä ja se suoritetaan fyysisen kovalevyn tai levyjärjestelmän toimesta, kun käyttöjärjestelmässä toimiva prosessi pyytää tietyn tiedoston tietojen käyttämistä tai kun tietoa tallennetaan kovalevylle.

Tässä testissä on tarkoituksena havainnoida pelkkiin I/O-operaatioihin kuluvaa aikaa ilman PHP-koodin suorittamista. Latausajan havainnoimiseksi Locustilla ladataan fstat.php:tä, joka suorittaa samanlaiset levytapahtumat, jotka tapahtuvat web-palvelimen ladatessa index.php-sivun käyttäjän selaimen ladatessa verkkopalvelun etusivun. Näin voidaan havainnoida miten palvelimen tallennusjärjestelmä toimii, kun samaa testiä toistetaan kerta toisensa jälkeen. Sama testi suoritetaan ilman välimuistia yhdellä käyttäjällä ja välimuistin kanssa 1-300 käyttäjällä samaan tapaan kuin ohjelmakoodin suoritusaikoja mittaavat testit.

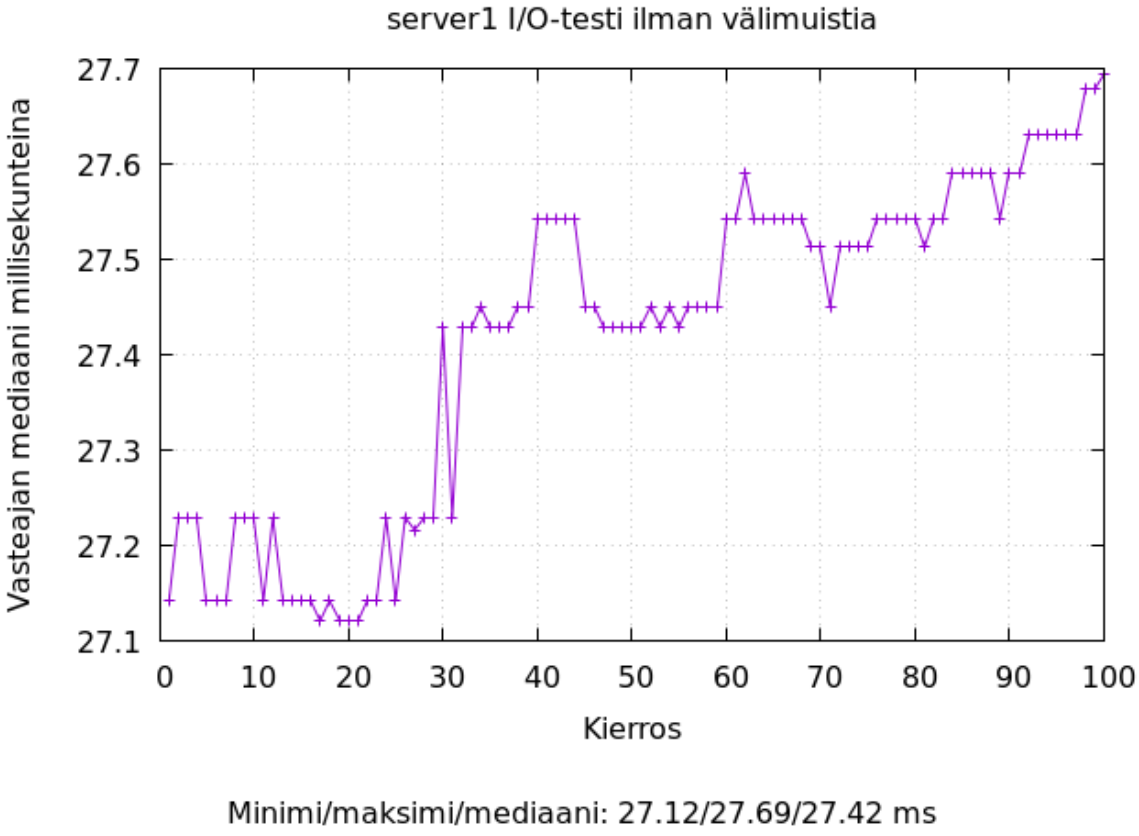
Testivirtuaalikoneiden mittaustulokset esitetään omina kuvaajinaan omissa aluvuissaan. Magneettisen pyörivän kovalevyn osalta tulokset löytyvät aliluvusta 5.3.1 ja SSD- ja NVMe-kovalevyjen aliluvusta 5.3.2. Alilukuun 5.3.3 on koottu mittausten yhteenveto yhdistettyine kaavioineen.

5.3.1 Magneettinen (pyörivä) kovalevy

Magneettisen kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server1-testivirtuaalikoneen testituloksen ensimmäisen testin tulos (Kuva 5.1) ilman välimuistia oli kolmesta testipalvelimesta heikoin. Tämä oli odotettua, sillä IOPS-suorituskyvyn rajat tulevat helposti vastaan etenkin pieniä tiedostoja luettaessa, joita tässä testissä joudutaan lukemaan useampi sata kappaletta jokaista index.php-sivunlatausta kohden. Näin ollen hitaan kovalevyn heikko suorituskyky korostuu entisestään, koska jokaisen tiedoston lukeminen levyltä aiheuttaa aina uuden lu-

kuoperaation. Testin latausajat olivat hyvin tasaisia vaihdellen 27-28 millisekunnin välillä, joten mittaustuloksia voidaan pitää luotettavina.

Tässä tutkielmassa ei mitattu suoritinkäyttöä, joten näiden testien osalta jäi epäselväksi, kuinka merkittävästi kovalevyn suorituskyky ja vastaavasti suoritinkapasiteetti vaikuttavat latausaikaan. Tein kuitenkin pintapuolista havainnointia testipalvelimen suoritinkäytöstä testin ollessa käynnissä ja tekemieni havaintojen perusteella voi sanoa, että ainakin server1:n suoritinkapasiteettia on liian vähän mahdollisimman nopean latausajan saavuttamiseksi.

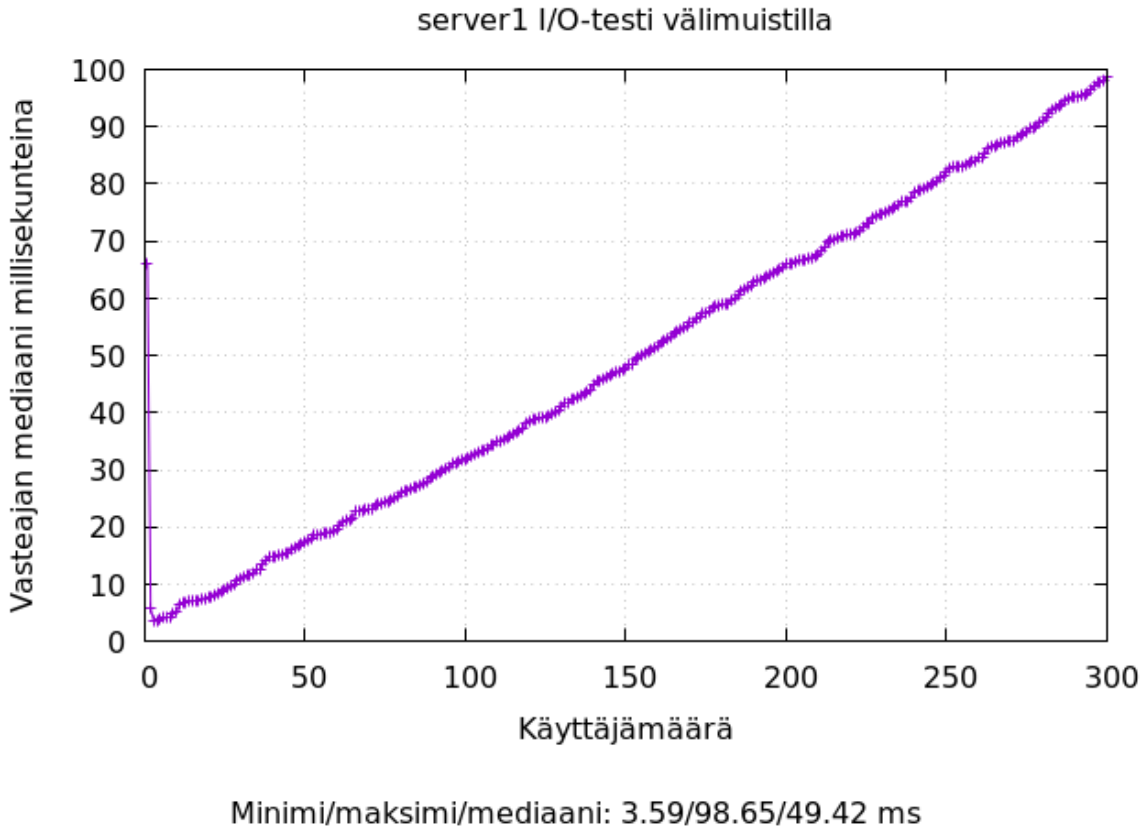


Kuva 5.1: Server1-palvelimen I/O-operaation suoritusaika ilman välimuistia

Myös välimuistia käytettäessä tulos (Kuva 5.2) oli odotetusti kolmen testipalvelimen osalta heikoin. Pienellä käyttäjämäärällä latausaika oli välimuistin ansiosta nopeampi kuin ilman välimuistia. Tämä oli odotettua, sillä välimuistin ansiosta palvelin pystyy ensimmäisten hitaiden ja suoraan levyltä luettavien vastausten jälkeen vastaamaan nopeasti hitaammasta kovalevystä huolimatta. Käytännössä ko-

valevyn hitaus ei tässä vaiheessa haittaa lainkaan tarvittavien tiedostojen löytyessä välimuistista.

Suuremmalla kävijämäärällä suoritinkapasiteetin rajat kuitenkin tulevat vastaan jossain vaiheessa ja latausaika alkaa pidentyä, koska välimuistikaan ei enää pysty yksin pelastamaan tilannetta. Latausaika alkoikin kuorman vaikutuksesta hidastumaan tasaisesti, ollen ilman välimuistia suoritettua testiä hitaampi käyttäjämäärän ylittäessä 80 käyttäjän rajan. 300 käyttäjän kohdalla latausaika nousi jo lähes sataan millisekuntiin.



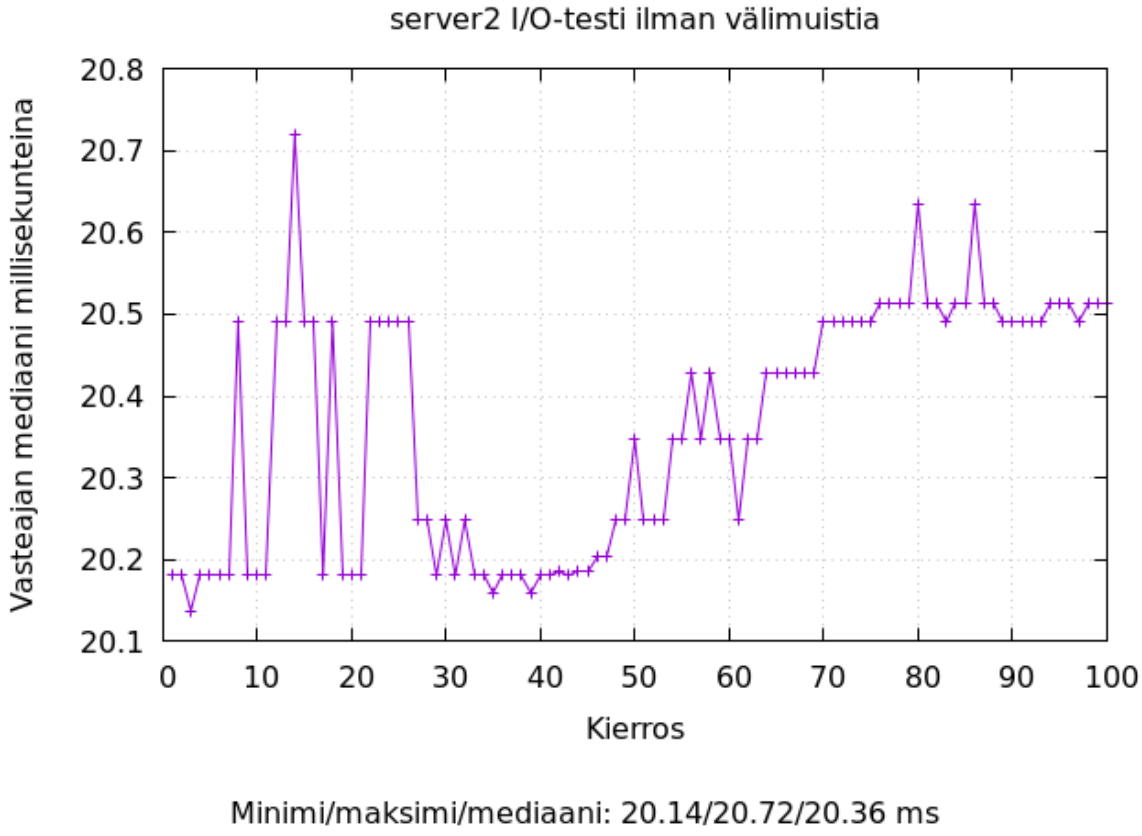
Kuva 5.2: Server1-palvelimen I/O-operaation suoritusaika välimuistilla

5.3.2 SSD- ja NVMe-kovalevyt

SSD-kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server2-testivirtuaalikoneen testitulos (Kuva 5.3) ilman välimuistia oli odotetusti magneettisen kovalevyn suorituskyvyn tasolle rajoitettua levyä parempi. Levyltä luettavat

tiedostot ovat verrattain pieniä eli maksimissaan muutaman kymmenen kilotavun kokoisia.

Latausajat olivat 19-20 millisekuntia ja server1:n tapaan ne pysyivät hyvin tasaisina koko testin ajan. Myös latausaikojen vaihtelu oli pientä, noin 0,5 millisekunnin luokkaa, ollen samaa tasoa server1:n kanssa. Vaikuttaa siltä, että valtaosan latausajasta muodostaa tiedostoja levyltä lukevan PHP-koodin suorittaminen, sillä server1:n ja server2:n latausajoissa ei kuitenkaan ole kovin suurta eroa, vaikka server2:n IOPS-suorituskyky on 200-kertainen server1:een nähden.

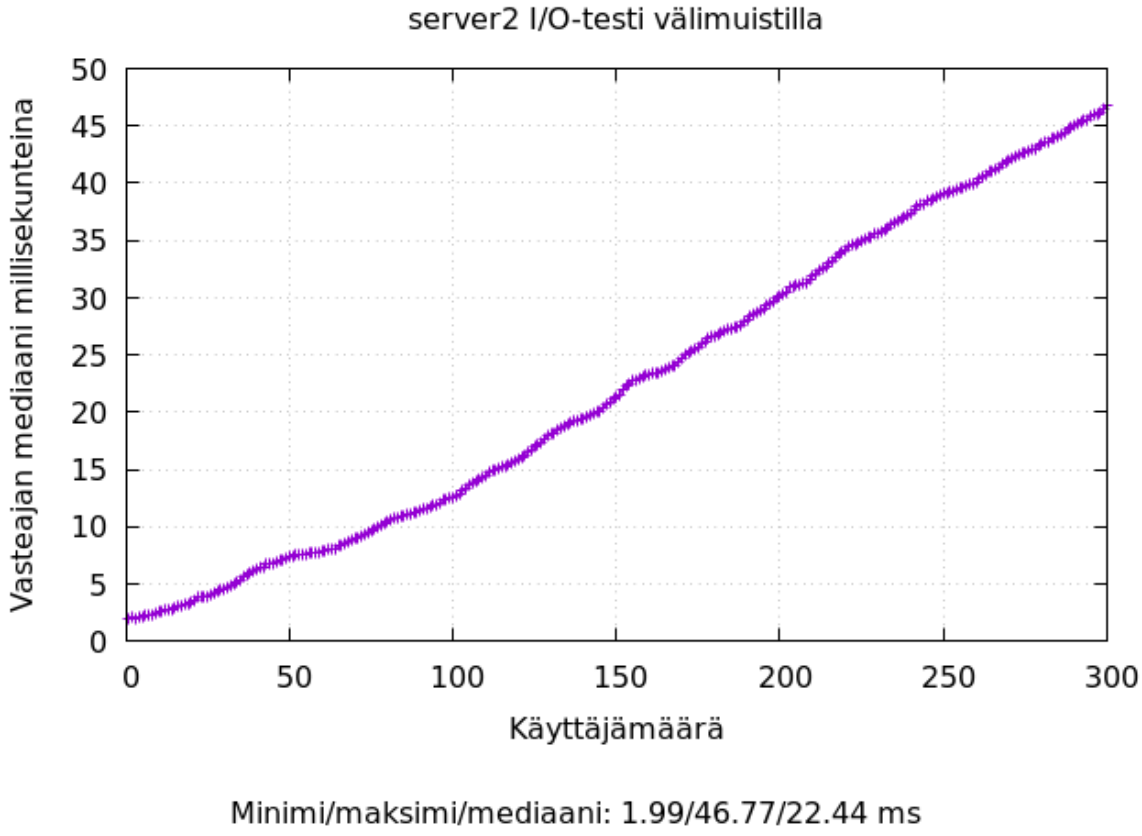


Kuva 5.3: Server2-palvelimen I/O-operaation suoritusaika ilman välimuistia

Myös välimuistia käytettäessä server2:n tulos (Kuva 5.4) oli server1:n magneettisen kovalevyn suorituskyvyn tasolle rajoitettua levyä parempi. Latausaika oli välimuistin ansiosta nopeampi kuin ilman välimuistia aina noin 150 käyttäjään saakka, jonka jälkeen latausaika jatkoi samaa tasaista hidastumistaan. 300 käyttäjän kohdalla latausaika nousi noin 46 millisekuntiin. Tämän testin osalta on nähtävissä, että

server2:n latausaikojen pituus on käytännössä koko testin ajan noin puolet server1:n testiaikojen pituudesta.

Koska tässä testissä käytettiin välimuistia, ei server1:n hitaamman kovalevyn pitäisi ensimmäisen latauksen jälkeen juurikaan vaikuttaa testitulokseen. Server1 on kuitenkin varustettu vain kahdella suoritinytimellä ja server2 neljällä, joten suoritusaikojen ero johtuu luultavasti siitä. Vaikka suoritinkäyttöä ei näissä testeissä mitattu, testien suorittamisen aikana tekemäni havainnot puoltavat sitä, että latausaikojen hidastuminen johtuu pääosin suoritinkapasiteetin loppumisesta, jolloin latauspyyntöjä alkaa kertyä jonoon ja niihin vastaaminen hidastuu.

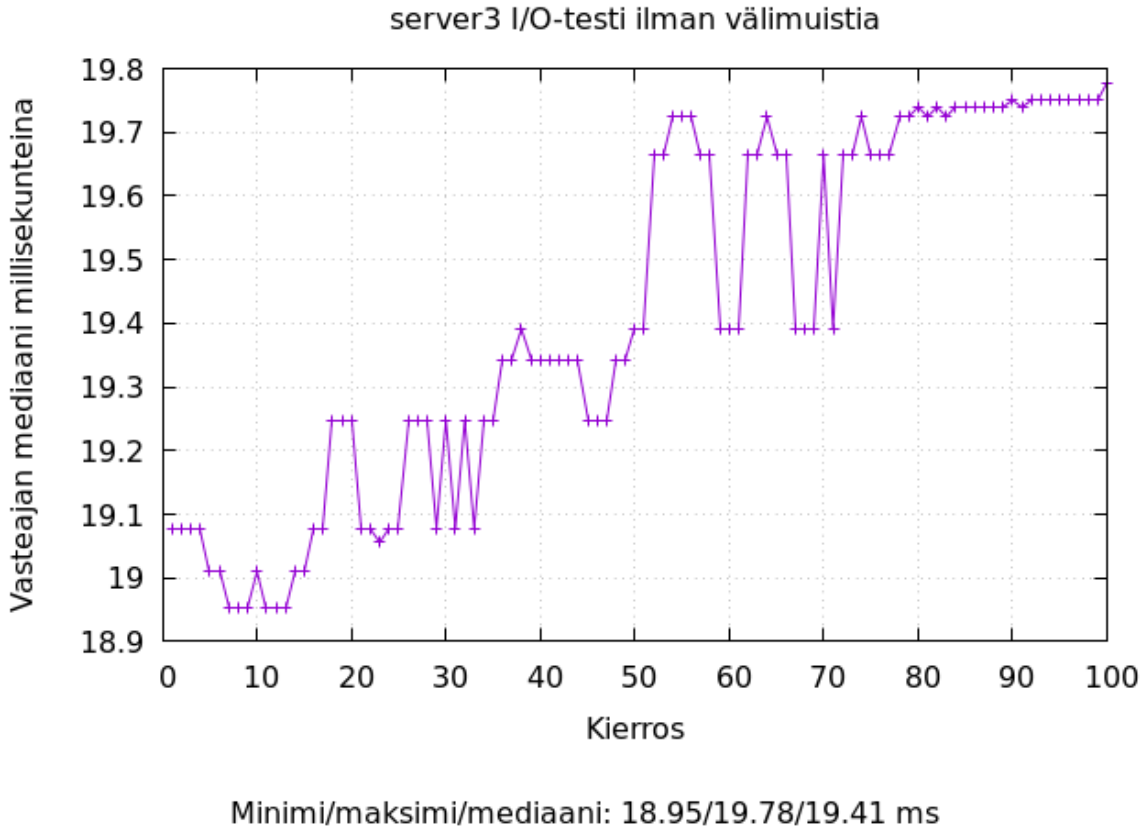


Kuva 5.4: Server2-palvelimen I/O-operaation suoritusaika välimuistilla

NVMe-kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server3-testivirtuaalikoneen testitulos (Kuva 5.5) ilman välimuistia oli lähes samalla tasolla server2:n kanssa. Latausajat vaihtelivat 19-20 millisekunnin välillä pysyen tasaisina koko testin ajan. Nopeampi levy ei siis enää juurikaan nopeuttanut I/O-

operaation suoritusta.

Näissä testeissä ei mitattu suoritinkäyttöä sivunlatausten aikana, joten jäi epäselväksi onko server3:n nopeammalla kovalevyllä vai suuremmalla suoritinkapasiteetilla suurempi vaikutus hieman parantuneeseen latausaikaan server2-palvelimeen nähden. Voisin kuvitella, että suuremmalla suoritinkapasiteetilla on tässä tapauksessa enemmän vaikutusta, sillä server2:n kovalevyn nopeus on jo verrattain suuri ja luettavat tiedostot pieniä, joten niiden prosessointi näyttelee suurempaa osaa latausajan muodostumisessa.

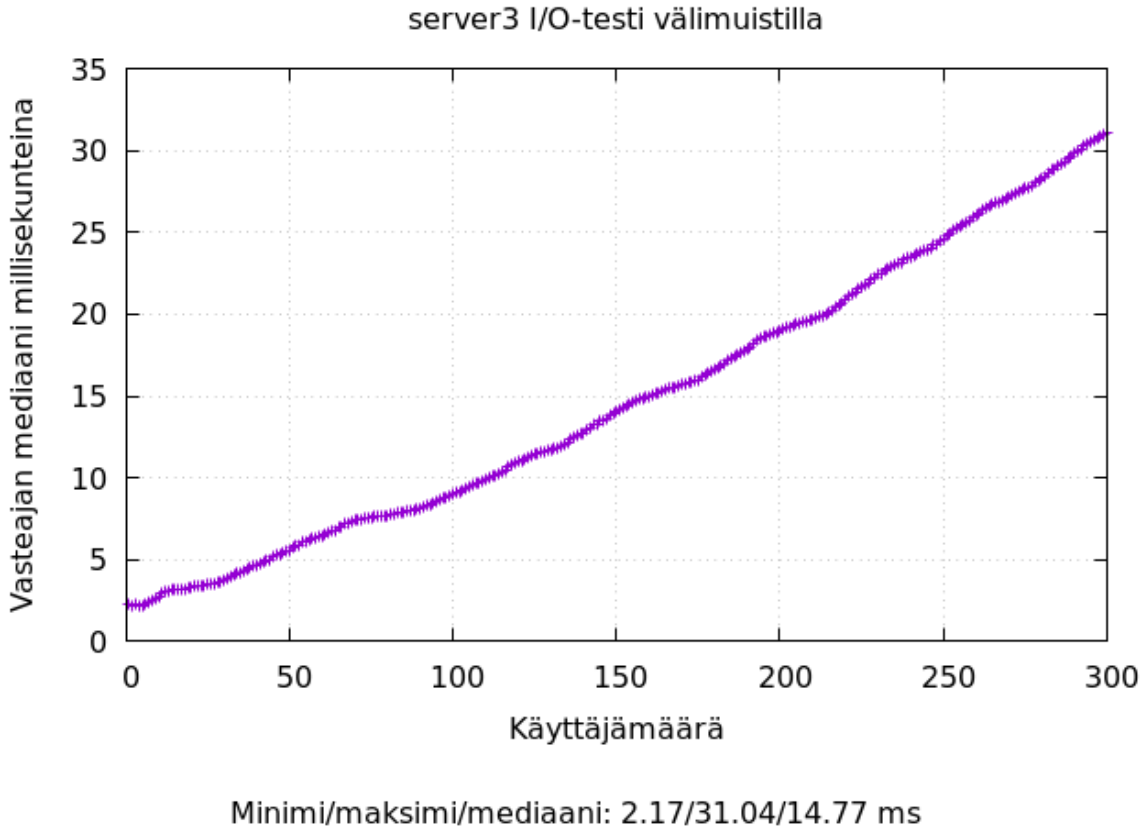


Kuva 5.5: Server3-palvelimen I/O-operaation suoritusaika ilman välimuistia

Välimuistia käytettäessä server3:n tulos (Kuva 5.6) oli kuitenkin merkittävästi server2:ta parempi latausajan ollessa 300 käyttäjän kuormalla noin 31 millisekuntia. Ilman välimuistia suoritettua testiä vastaava latausaika saavutettiin noin 200 käyttäjän kohdalla. Tässä testissä voi huomata suuremman suoritinkapasiteetin vaikutuksen. Vaikka yksittäisellä käyttäjällä ilman välimuistia suoritetun testin osalta

server2:n ja server3:n testitulokset olivat yhteneväisiä, oli server3 välimuistia käytettäessä kuitenkin selkeästi nopeampi suuremmilla käyttäjämäärillä.

Server3:n latausaika server2:een nähden ei enää puolitu, vaikka suoritinkapasiteetti tuplaantuu. Koska tässä testissä käytettiin välimuistia, en usko nopeammalla kovalevyllä olevan juurikaan vaikutusta testituloksiin. Näin ollen voidaankin päätellä, että hidastaviksi tekijöiksi muodostuvat mod_php:n ja PHP-funktioiden suoritukseen liittyvät asiat. On kuitenkin huomioitavaa, että näitä asioita ei tässä tutkielmassa selvitetty, joten kyseessä on vain oma oletukseni.



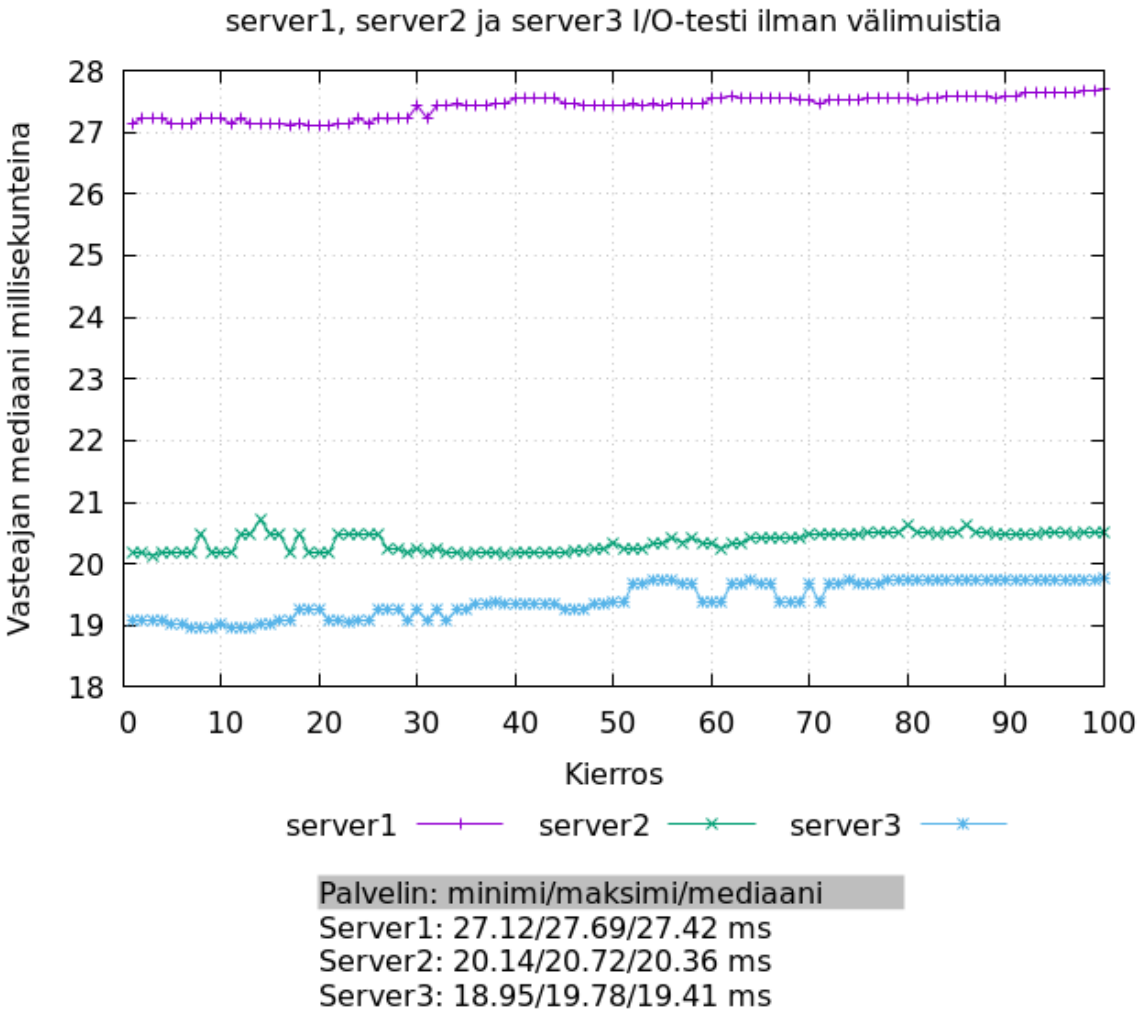
Kuva 5.6: Server3-palvelimen I/O-operaation suoritusaika välimuistilla

5.3.3 Yhteenveto

Kun kaikkien kolmen testipalvelimen I/O-operaatioiden suoritusajat ilman välimuistia kootaan samaan kaavioon (Kuva 5.7), on helppo havaita, että latausajat pysyvät tasaisina ja vaihteluväli alle yhden millisekunnin mittaisena kaikkien kol-

men testipalvelimen osalta. Server1:n latausaika oli 27 millisekunnin paikkeilla, server2:n 20 millisekunnin ja server3:n 19 millisekunnin paikkeilla. Tästä voidaan havaita, että server2:n ja server3:n latausajat ovat hyvin lähellä toisiaan.

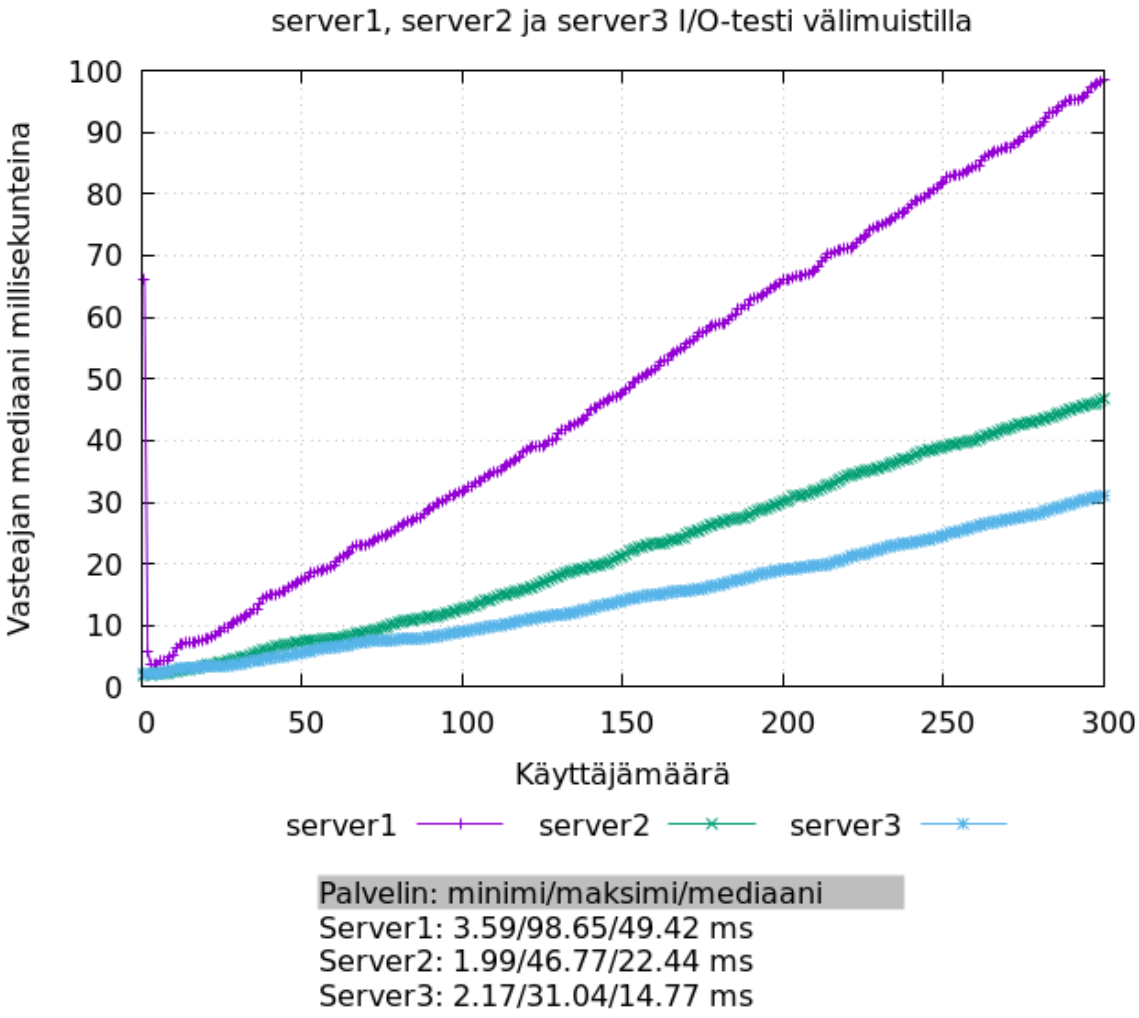
Server2 sisältää neljä suoritinydintä ja server3 kahdeksan. Vaikka ydinten määrä tuplaantuu, ei latausaika juurikaan enää lyhene. Vastaavasti server3 sisältää nelinkertaisen määrän suoritinkapasiteettia server1-palvelimeen nähden (kaksi vs. kahdeksan ydintä), ei latausaika kuitenkaan lyhene vastaavassa suhteessa. Asiaa ei tässä tutkielmassa selvitetty, mutta oman päätelmäni mukaan latausaikojen tasoittuminen johtuu mod_php:n ja PHP-funktioiden suoritukseen liittyvistä asioista sekä Linux-ytimen ja välimuistitoteutuksen toimintaan liittyvistä tekijöistä, joihin parempi suorituskyky ei enää juurikaan vaikuta.



Kuva 5.7: Testipalvelimien I/O-operaatioiden suoritusaika ilman välimuistia

Vastaavasti välimuistia käyttäen suoritetuista testeistä voidaan kaikkien kolmen testivirtuaalikoneen osalta havaita (Kuva 5.8), että latausaika kasvaa tasaisesti käyttäjämäärän lisääntyessä, eikä poikkeamia esiinny käytännössä ollenkaan. Välimuistia käytettäessä ensimmäinen sivunlataus on huomattavasti hitaampi kuin sitä seuraavat, suoraan välimuistia palveltavat lataukset.

Suuremmalla kävijämäärällä suoritinkapasiteetin rajat tulevat vastaan jossain vaiheessa. Server1:n latausaika kasvaa odotetusti kaikkein jyrkimmin, sillä se on varustettu pienimmällä suoritinkapasiteetilla, eikä välimuistikaan enää pysty pelastamaan tilannetta. Sama ilmiö on havaittavissa myös neljä suoritinydintä sisältävällä server2-palvelimella ja kahdeksan ytimen server3-palvelimella, mutta niiden osalta latausajan piteneminen ei ole yhtä suurta.



Kuva 5.8: Testipalvelimien I/O-operaatioiden suoritusaika välimuistilla

5.4 Ohjelmakoodin suoritusaika

Aiemmassa osiossa käsiteltiin I/O-operaation suoritusaikaa, joka on yksi osa-alue koko ohjelmakoodin suoritusajasta. Ohjelmakoodin suorittamiseksi koodi täytyy ensin lukea levyltä, jotta se voidaan toimittaa suorittimen suoritettavaksi. Näin ollen I/O-operaatioiden hidastuminen vaikuttaa merkittävästi kokonaissuoritusaikaan aina HTTP-pyynnön lähettämisestä HTTP-vastauksen vastaanoton valmistumiseen. Tässä osiossa havainnoidaan tätä suoritusaikaa lataamalla Wordpress-verkkopalvelun etusivua Locustin avulla. Mittaus suoritetaan I/O-testin tapaan ensin yksittäisellä käyttäjällä ilman välimuistia. Sen jälkeen suoritetaan testin toinen vaihe välimuistin kanssa yhdellä käyttäjällä aloittaen. Käyttäjämäärää kasvatetaan tasaisesti ja vähitellen aina 300 käyttäjään saakka ja samalla havainnoidaan latausaikojen kehitystä.

Testivirtuaalikoneiden mittaustulokset esitetään omina kuvaajinaan omissa aluvuissaan. Magneettisen pyörivän kovalevyn osalta tulokset löytyvät aliluvusta 5.4.1 ja SSD- ja NVMe-kovalevyjen aliluvusta 5.4.2. Alilukuun 5.4.3 on koottu mittausten yhteenveto yhdistettyine kaavioineen.

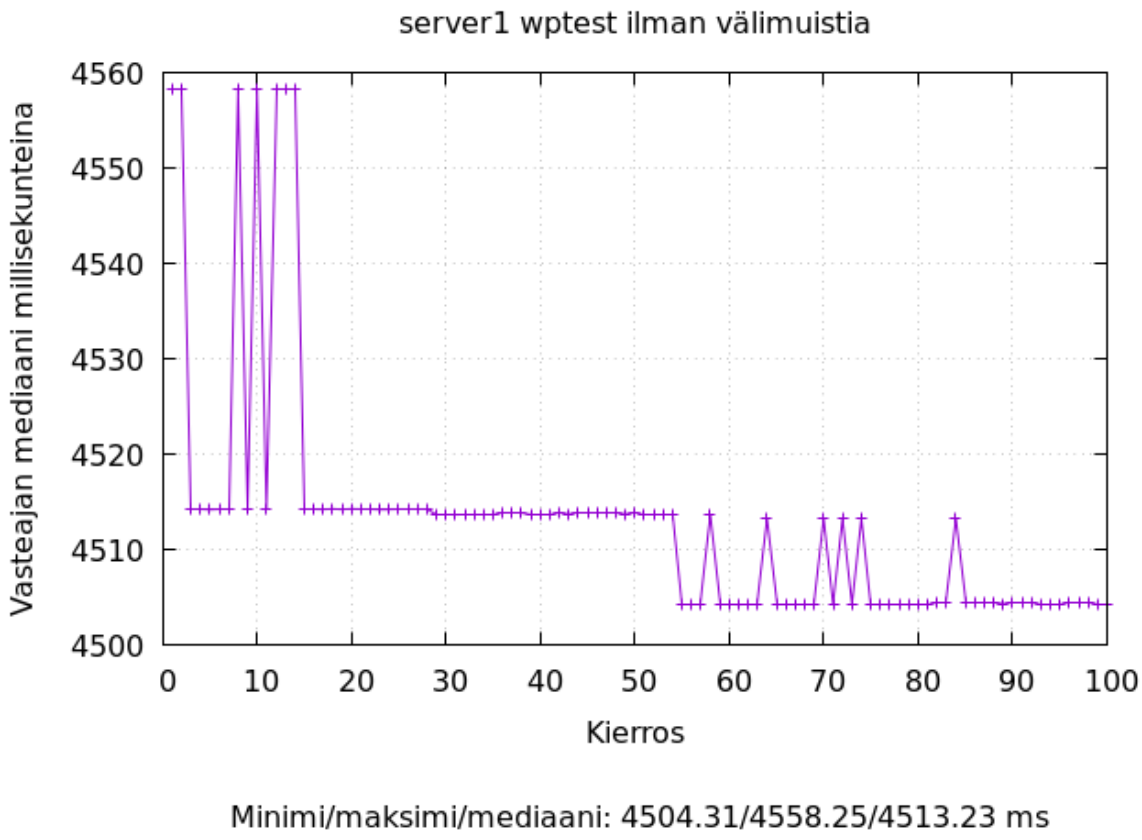
5.4.1 Magneettinen (pyörivä) kovalevy

Magneettisen kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server1-testivirtuaalikoneen lataustestin tulos (Kuva 5.9) oli I/O-testin tapaan kolmesta testipalvelimesta heikoin. Tulos oli sinänsä odotettu, koska IOPS-suorituskyvyn rajojen tullessa vastaan ohjelmakoodin suorittaminen luonnollisesti hidastuu. Tämä johtuu yksinkertaisesti siitä, että koodia ei tietenkään voida suorittaa, jos sitä ei ole vielä saatu luettua levyltä. Latausajat pysyivät 4,5 sekunnin tasolla ja hyvin tasaisina koko testin ajan.

Koska tutkielmassa ei mitattu suoritinkäyttöä, jäi tässäkin testissä I/O-testin tavoin epäselväksi, kuinka merkittävästi kovalevyn suorituskyky ja vastaavasti suoritinkapasiteetti vaikuttavat latausaikaan server1:n osalta. Käytännössä kyseisen palvelimen kovalevyn suorituskyky ja suoritinkapasiteetti ovat nykymittapuulla riittämättömiä Wordpress-verkkopalvelun ylläpitoon, sillä latausaika jo yksittäisellä käyttäjällä on liian pitkä mielekkään käyttäjäkokemuksen aikaansaamiseksi.

Täytyy toki muistaa, että tämä testi suoritettiin ilman välimuistia, joten normaalikäytössä välimuistin ollessa toiminnassa tilanne parantuu usein merkittävästi. Wordpress-verkkopalvelut kuitenkin sisältävät usein sellaista sisältöä, jota ei voida

tarjota välimuistista, vaan se joudutaan lukemaan suoraan levyltä. Tällöin pullonkaulaksi muodostuvat hidas kovalevy ja liian pieni suoritinkapasiteetti.

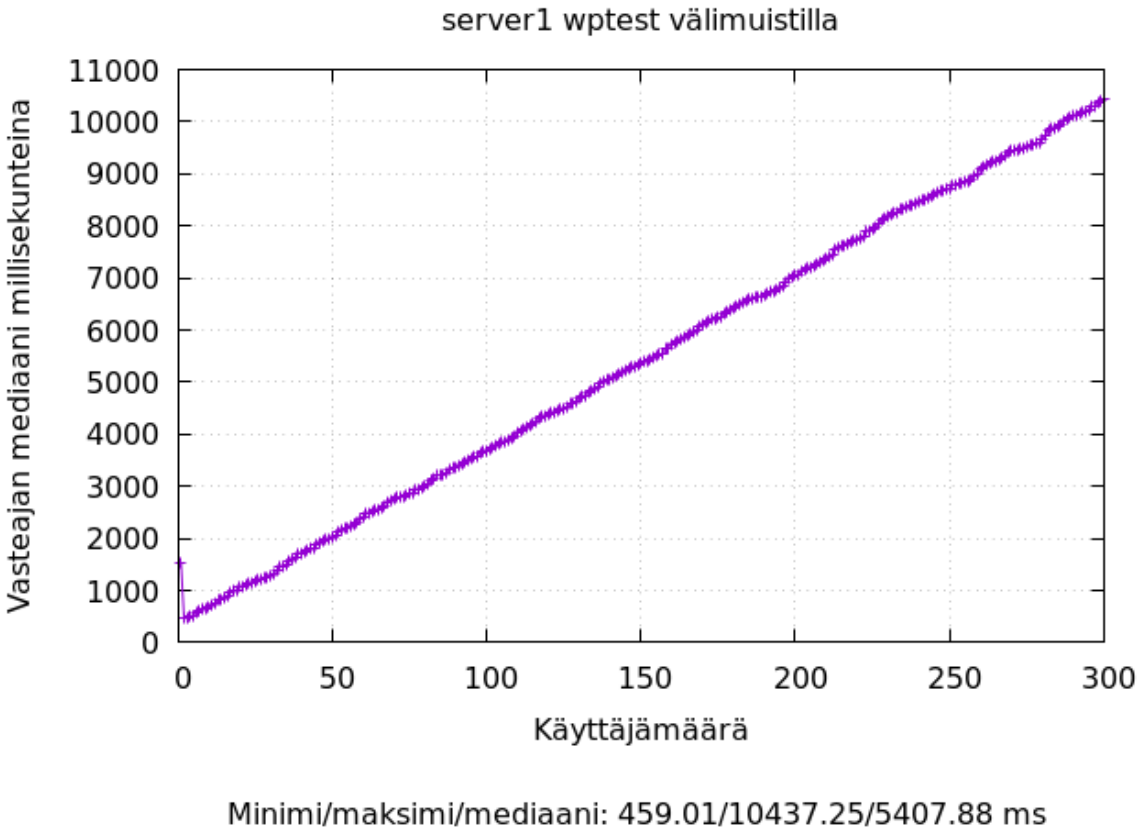


Kuva 5.9: Server1-palvelimen lataustestin suoritusaika ilman välimuistia

Välimuistia käytettäessä latausaika testin (Kuva 5.10) alkuvaiheessa oli 0,5 sekunnin paikkeilla. Tämä on vielä varsin kelvollinen ja nopea latausaika. Käytännössä käyttäjä kokee palvelun toimivan nopeasti ja ilman mitään ylimääräistä viivettä. Varsinkin mobiililaitteita käytettäessä on laitteen verkkoyhteydestä aiheutuva viive on suurempi olosuhteissa, joissa radiosignaalin kuuluvuus on heikko.

Käyttäjämäärän kasvaessa latausajat pitenivät tasaisesti ja loppujen lopuksi 300:n käyttäjän kuormalla latausaika oli jo yli kymmenen sekunnin mittainen. Latausaika oli yhtä pitkä ilman välimuistia suoritetun testin latausajan kanssa noin 130 käyttäjän paikkeilla. Noin 50 käyttäjään saakka latausaika pysyy alle kahdessa sekunnissa ja kolmen sekunnin rajapyykki, jota voidaan pitää mielekkään latausajan ylärajana, saavutetaan noin 75 käyttäjän kohdalla.

I/O-testin tapaan suuremmalla kävijämäärällä suoritinkapasiteetin rajat tulevat vastaan ja latausaika alkaa pidentyä melko paljon, sillä latauspyyntöjen jonoutuessa suoritinkapasiteetin puutteen vuoksi välimuistikaan ei enää pysty pelastamaan tilannetta. Jos server1:een lisättäisiin suoritinytimiä, sen latausajat todennäköisesti lyhenisivät merkittävästi, mutta välimuistin ”ohi” menevät latauspyynnöt olisivat silti tuskaisen hitaita varsinkin suuremmilla käyttäjämäärillä.



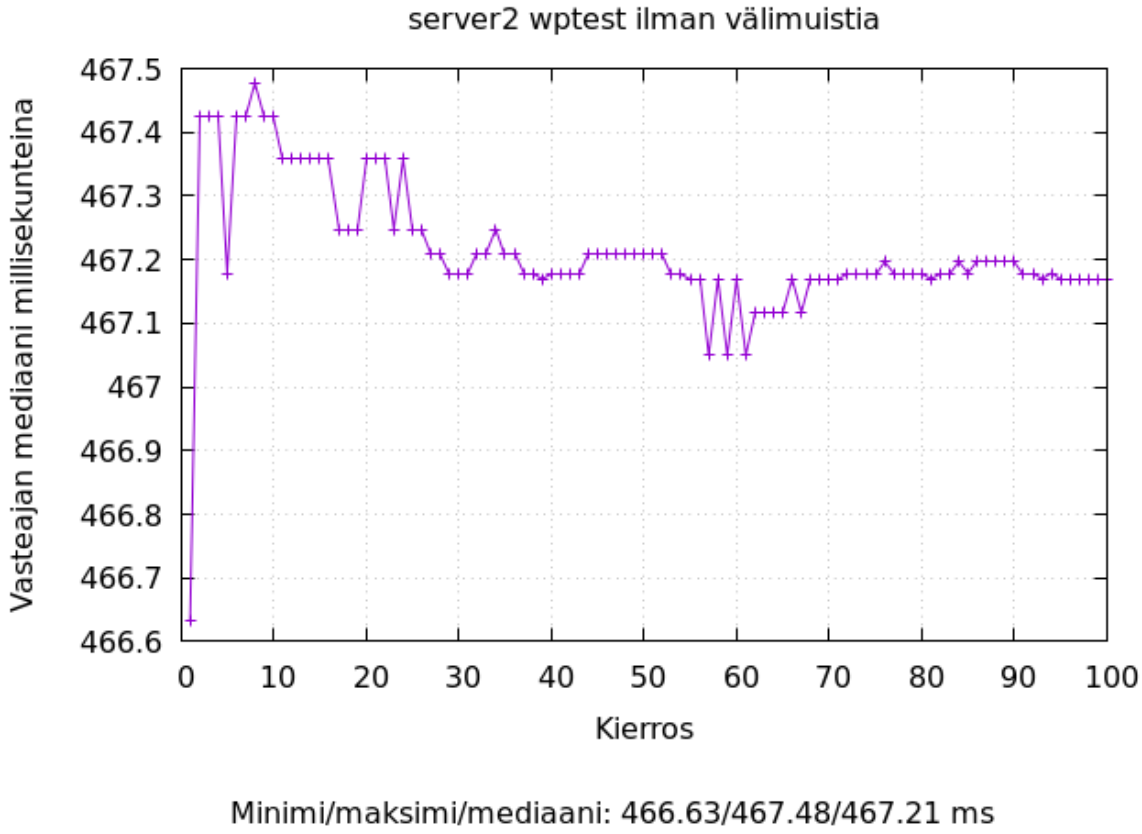
Kuva 5.10: Server1-palvelimen lataustestin suoritusaika välimuistilla

5.4.2 SSD- ja NVMe-kovalevyt

SSD-kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server2-testivirtuaalikoneen lataustestin tulos (Kuva 5.11) ilman välimuistia oli huomattavasti parempi kuin magneettisen kovalevyn suorituskyvyn tasolle rajoitetun levyn. Latausajat olivat 0,45 sekunnin tasolla ja pysyivät tasaisina koko testin ajan.

Tästä testituloksesta voidaan hyvin huomata, että kovalevyn lisääntynyt IOPS ja

suurempi suoritinkapasiteetti vaikuttavat huomattavan paljon. Server2:n latausaika on käytännössä kymmenesosa server1:n latausajasta, joten voidaan puhua erittäin merkittävästi parannuksesta. Lisäksi täytyy muistaa, että tämä testi tehtiin ilman välimuistia, jolloin normaalissa käyttötilanteessa välimuistin ollessa toiminnassa latausaika paranee entisestään.

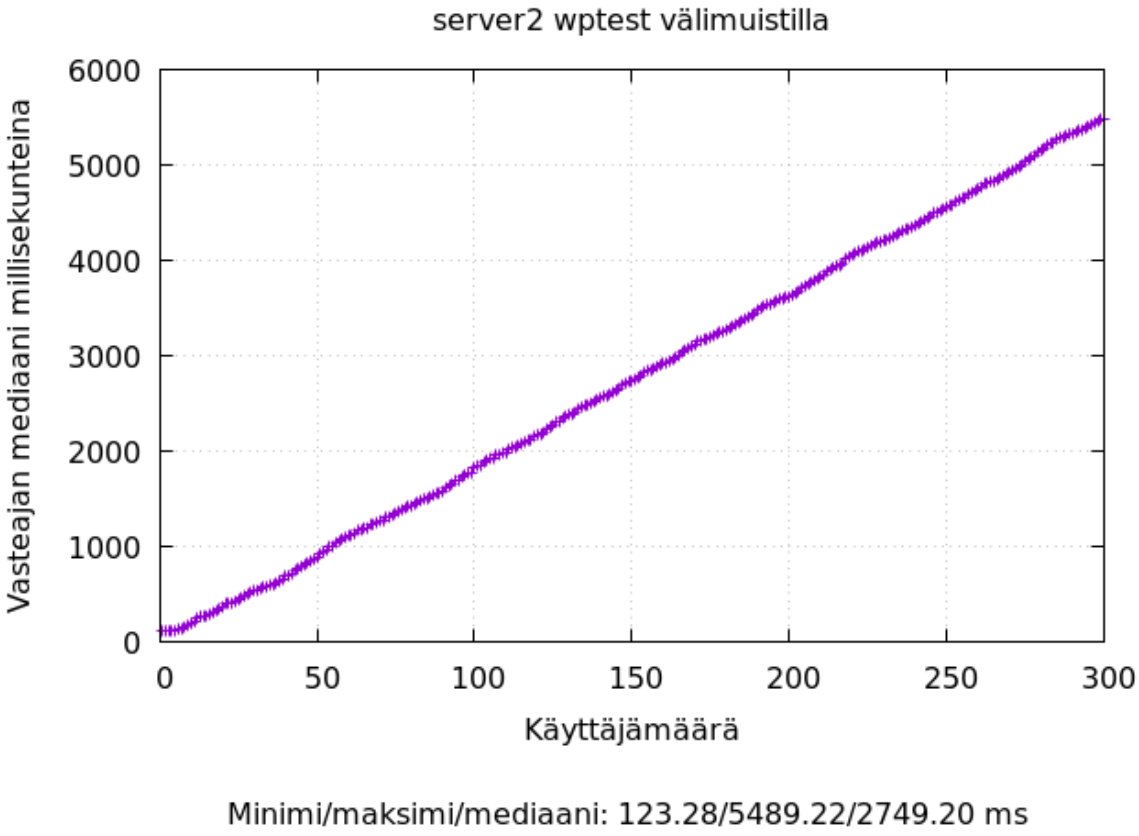


Kuva 5.11: Server2-palvelimen lataustestin suoritusaika ilman välimuistia

Välimuistin kanssa suoritetussa testissä (Kuva 5.12) latausaika oli testin alussa noin 0,15 sekunnin paikkeilla ja kasvoi tasaisesti käyttäjämäärän lisääntyessä. 300:n käyttäjän kuormalla latausaika oli noin 5,5 sekuntia. Noin 30:n käyttäjän kohdalla latausaika oli yhtä pitkä kuin ilman välimuistia suoritetun testin latausaika.

Tämän testin osalta on I/O-testin tapaan havaittavissa, että suoritinkapasiteetin tuplaantuminen vaikuttaa latausaikaan hyvin merkittävästi. Tuplaantuneen suoritinkapasiteetin turvin server2:n latausaika on noin puolet lyhyempi kuin server1:llä. Toki server2:ssa on myös merkittävästi nopeampi kovalevy, mutta sillä ei välimuis-

tia käytettäessä ole niin suurta merkitystä kuin suoritinkapasiteetilla. Toki välimuistin ”ohi” menevien latauspyyntöjen osalta auttaa, kun ne saadaan luettua levyltä nopeammin.

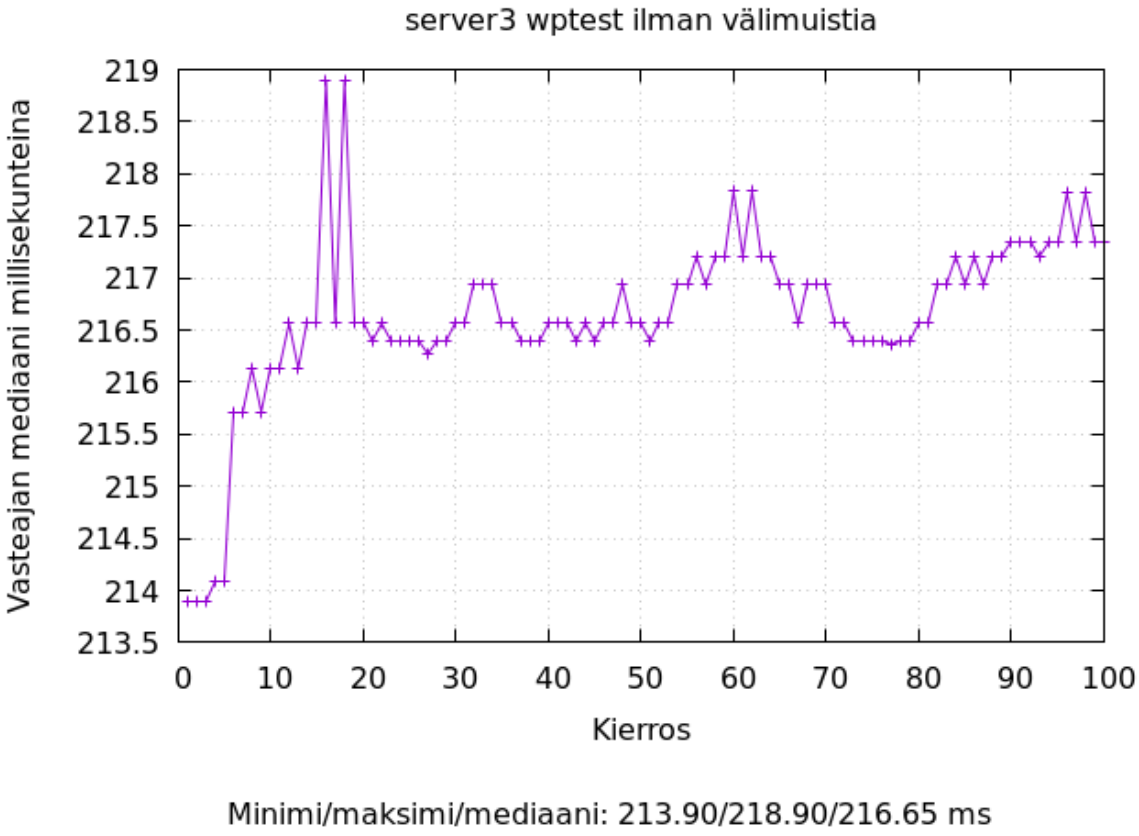


Kuva 5.12: Server2-palvelimen lataustestin suoritusaika välimuistilla

NVMe-kovalevyn suorituskyvyn tasolle rajoitetulla kovalevyllä varustetun server3-testivirtuaalikoneen lataustestin tulos (Kuva 5.13) ilman välimuistia oli huomattavasti parempi server2:een nähden. Vaikka kovalevyn IOPS kymmenkertaistui, ei latausaika kuitenkaan lyhentynyt vastaavassa suhteessa. Mittauksen latausajat olivat noin puolet lyhyempiä pysyen koko testin ajan 0,22 sekunnin paikkeilla.

Koska tässä mittauksessa käytettiin välimuistia, ei kovalevyn nopeuden lisääntymisellä ole juurikaan vaikutusta mittaustuloksiin, joista on nähtävissä jo I/O-testeissäkin tehty havainto siitä, että suoritinkapasiteetin tuplaantuminen käytännössä puolittaa latausajan. On todennäköistä, että lisääntyneen kovalevyn IOPS-kapasiteetin ja tuplaantuneen suoritinkapasiteetin myötä välimuistin kanssa toimit-

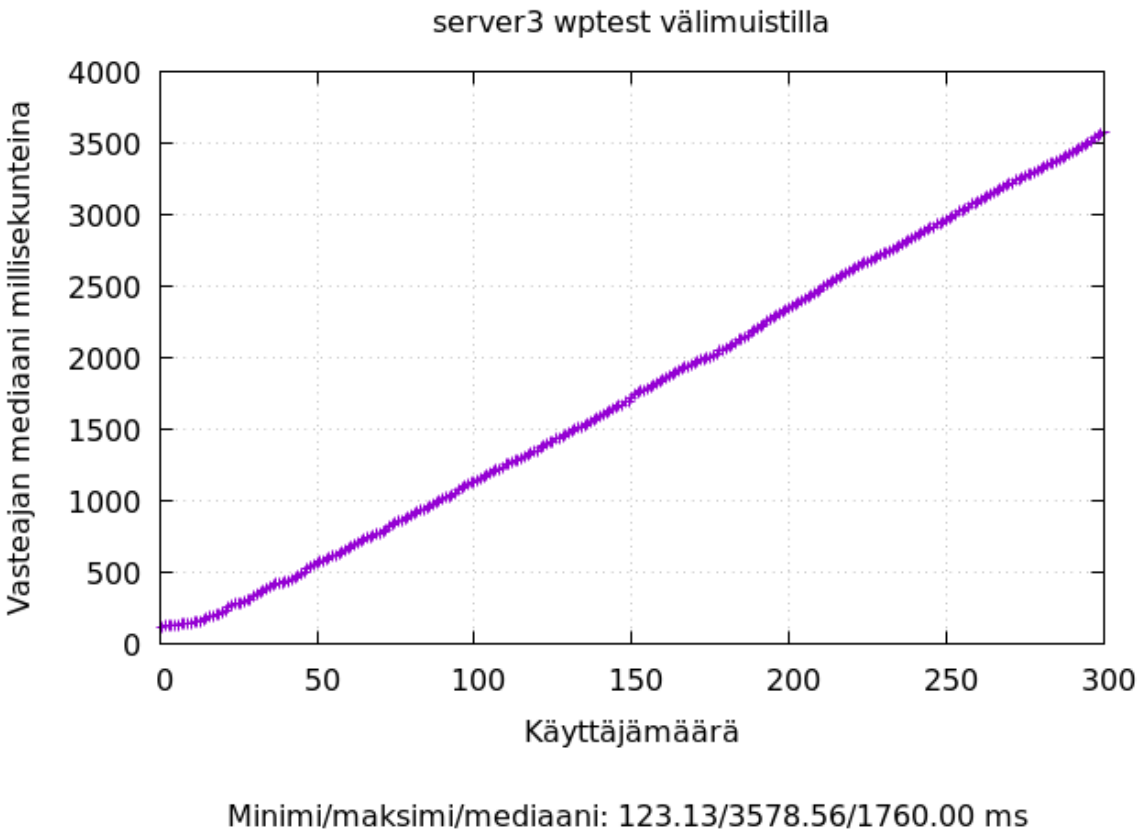
taessa latausajat ovat huomattavasti lyhyempiä.



Kuva 5.13: Server3-palvelimen lataustestin suoritusaika ilman välimuistia

Välimuistia käytettäessä latausaika oli testin (Kuva 5.14) aluksi noin 0,15 sekunnin paikkeilla. Latausaika kasvoi tasaisesti käyttäjämäärän lisääntyessä ollen lopulta noin 3,5 sekuntia 300:n käyttäjän kuormalla. Tämän testin ja ilman välimuistia suoritetun testin latausajat kohtaavat 30:n käyttäjän kohdalla. Testituloksesta voi havaita, että suoritinkapasiteetin tuplaantuminen ei enää puolita latausaikaa, kuten tapahtui server1:n ja server2:n välisiä latausaikoja vertailtaessa.

Mittaustuloksen muutos oli samansuuntainen myös vastaavassa I/O-testissä. Kuten jo siinä yhteydessä totesin, oman päätelmäni mukaan latausaikojen tasoittuminen johtunee mod_php:n ja PHP-funktioiden suoritukseen liittyvistä asioista sekä Linux-ytimen ja välimuistitoteutuksen toimintaan liittyvistä tekijöistä, joihin suurempi IOPS- ja suoritinkapasiteetti ei enää juurikaan vaikuta.



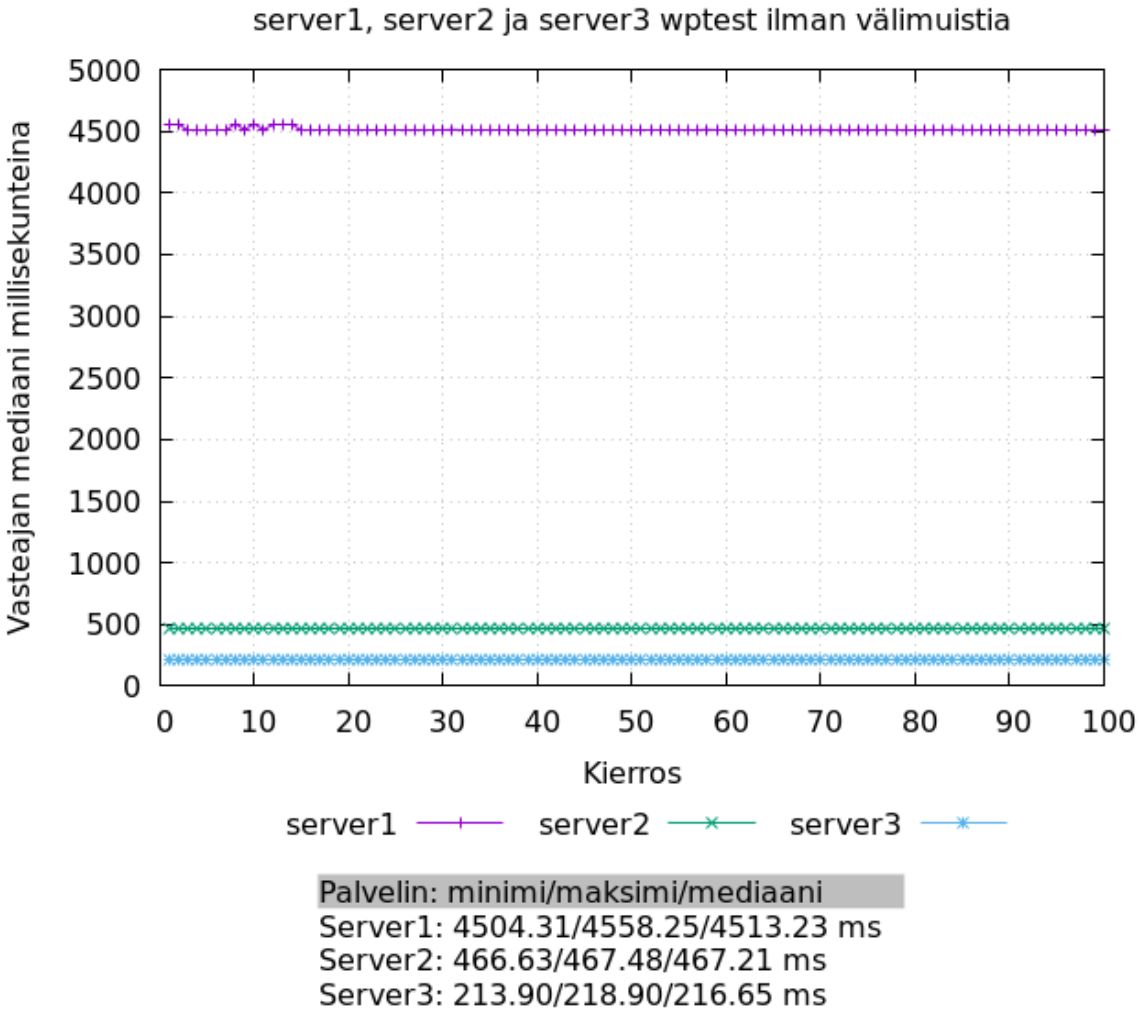
Kuva 5.14: Server3-palvelimen lataustestin suoritusaika välimuistilla

5.4.3 Yhteenveto

Mittausten latausajat eri testipalvelimilla (Kuva 5.15) ilman välimuistia olivat siinä mielessä odotusten mukaisia, että server1 oli kaikkein hitain, server2 toinen ja server3 nopein. Näin pitää palvelimille allokoidun kapasiteetin valossa ollakin. Jokaisen palvelimen osalta latausajat pysyivät hyvin tasaisina koko testin ajan, eikä suurempia vaihteluita ilmennyt.

Ohjelmakoodin suoritusaikamittauksien osalta on nähtäväissä sama kehitys kuin I/O-mittauksissakin: vaikka ydinten määrä tuplaantuu tai nelinkertaistuu, ei latausaika juurikaan enää lyhene. Koska asiaa ei tässä tutkielmassa sen tarkemmin selvitetty, jää asian käsittely vain oman pohdintani varaan. Sen perusteella latausaikojen tasoittuminen hitaampiin palvelimiin verrattuna kapasiteettia kasvatettaessa johtuu mod_php:n ja PHP-funktioiden suoritukseen liittyvistä asioista. Lisäksi Linux-ytimen ja välimuistitoteutuksen sisäiseen toimintaa liittynee tekijöitä, joihin

parempi suorituskyky ei enää juurikaan vaikuta.



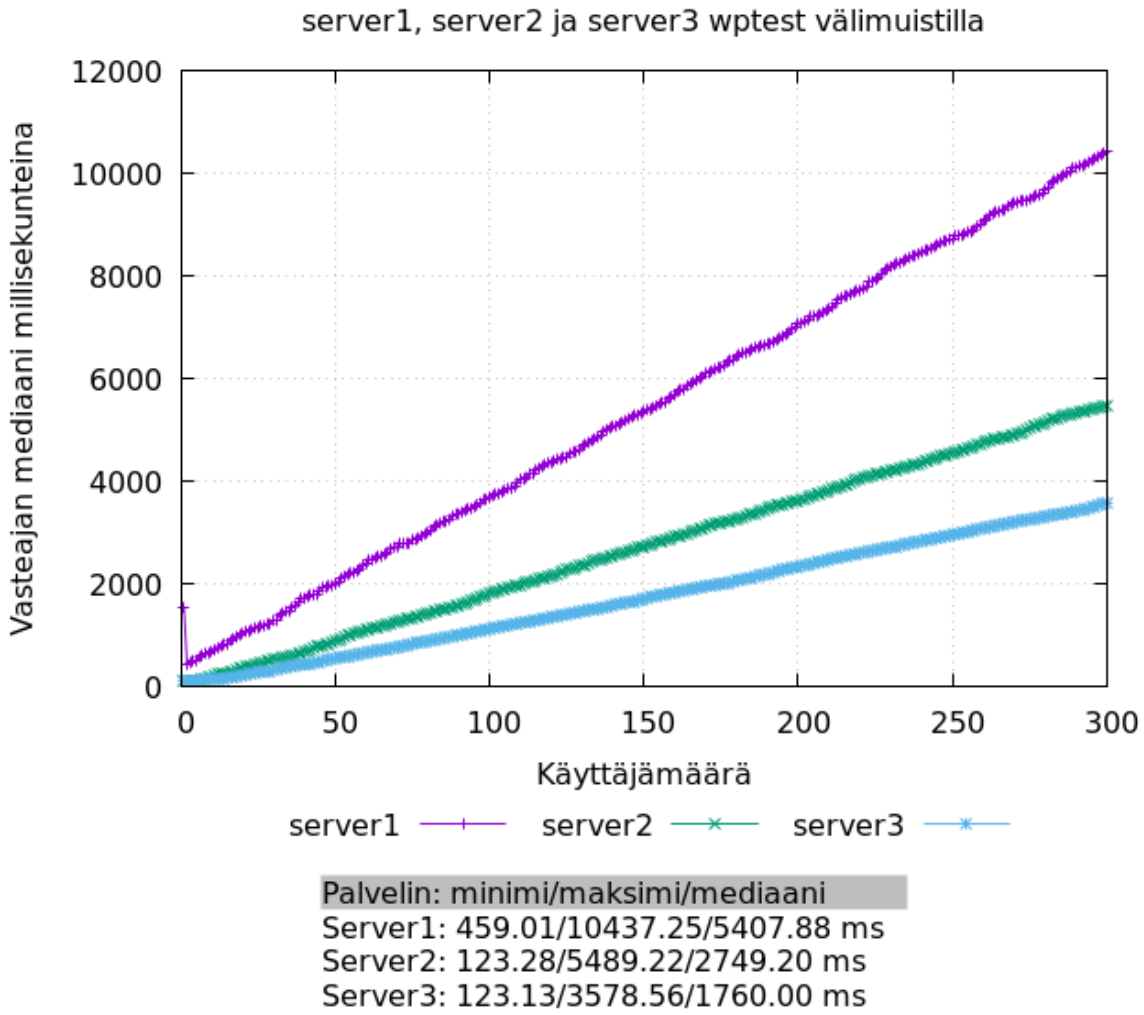
Kuva 5.15: Testipalvelimien lataustestien suoritusajat ilman välimuistia

Myös välimuistia käytettäessä latausajat 5.16 olivat palvelimille allokoidun kapasiteetin mukaisia, eikä yllätyksiä ilmennyt. Latausajat kasvoivat tasaisesti kävijämäärän lisääntyessä kaikilla kolmella testipalvelimella. Tuloksista on hyvin nähtävissä, kuinka hitaampi levy ja pienempi suoritinkapasiteetti pidentää latausaikoja varsinkin suuremmilla kävijämäärillä. Mittaustulokset olivat tältä osin hyvin samansuuntaisia kuin I/O-testeissäkin, eikä tehdyissä havainnoissa ilmennyt poikkeuksia.

Oli odotettua, että suuremmalla kävijämäärällä suoritinkapasiteetin rajat tulevat vastaan tietyssä vaiheessa testiä, kun kävijämäärä kasvaa niin suureksi, että suori-

tinytimet eivät enää pysty suoriutumaan kaikista sivunlatauspyynnöistä ilman jonoutumista. Välimuistista palveltavien latauspyyntöjen osalta nopeampi kovalevy ei vaikuta latausaikoihin, sillä kovalevyltä ei näissä tilanteissa jouduta lukemaan mitään.

I/O-testin kanssa yhteneväisesti myös tässä testissä server1-palvelimen latausaika kasvaa jyrkimmin. Tämä selittyy palvelimen testipalvelimista pienimmällä suoritinkapasiteetilla eli kahdella ytimellä. Välimuistikaan ei enää tietyn pisteen jälkeen auta tässä tilanteessa, koska suoritinytimet eivät kykene käsittelemään kaikkia latauspyyntöjä ilman jonoutumista. Sama tapahtuu myös server2- ja server3-palvelimen kohdalla, mutta niiden latausajat eivät pitene niin oleellisesti.



Kuva 5.16: Testipalvelimien lataustestien suoritusajat välimuistilla

6 Analyysi

Työtä aloittaessa mietin muutamia eri vaihtoehtoja testiympäristön toteutukseen. Yksi vaihtoehto oli jonkin julkisen, ison ja tunnetun pilvipalvelun, kuten Google Cloudin käyttö. Tämä vaihtoehto kuitenkin haudattiin kustannussyistä, sillä varsinkin nopeampia levyjä käytettäessä kustannukset nousevat helposti ja lisäksi testiympäristöjä joutuisi kustannussyistä sammuttamaan ja tuhoamaan aina testien päätteeksi. Minulla oli jo itselläni muutamia vanhoja PC-tietokoneita, joita voisi käyttää testiympäristöjen isäntäkoneina. Yksi vaihtoehto oli myös vuokrata edullinen dedikoitu palvelin esimerkiksi Hetzneriltä, jota voisi sitten käyttää testiympäristöjen isäntäkoneena.

Alunperin käytin isäntäkoneina kahta omaa tietokonetta, joihin asensin Proxmoxin ja perustin virtuaalikoneet siten, että testin kohteena olevat virtuaalikoneet sijaitsivat toisella ja kuormitustestejä suorittava virtuaalikone toisella isäntäkoneella. Tällä kokoonpanolla tehtiinkin lukuisia testejä, mutta ongelmaksi muodostui lopulta se, että jo hieman vanhentuneissa koneissa oli vain SSD-kovalevyt, enkä näin ollen pystynyt testaamaan latausnopeuksia NVMe-kovalevyn tasoisella suorituskyvyllä varustetulla virtuaalikovalevyllä varustetun virtuaalikoneen kanssa. Sen johdosta hankin Hetzneriltä dedikoidun vuokrapalvelimen, jossa on NVMe-levy ja lisäksi enemmän muistia ja suoritinytimiä.

Tutkielmaan päätyneet lopulliset suorituskykymittaukset suoritettiin edellä mainitulla Hetznerin vuokrapalvelimella, jonka suorituskyky riitti hyvin kaikkiin suoritettuihin testeihin. Testien työstämiseen meni yllättävän paljon aikaa, vaikka ne eivät sinänsä ole mitenkään monimutkaisia. Pienten yksityiskohtien hiominen ja havaittujen virheiden korjaaminen on toisinaan hidasta ja aikaa vievää. Enemmän tai vähemmän epäonnistuneita testikierroksia on aloitettu luultavasti useampi sata kappaletta. Suurin osa näistä on lopetettu kesken jonkin testin aikana havaitun ongelman vuoksi. Sen jälkeen ongelma on korjattu ja aloitettu testikierros uudelleen. Samaa kaavaa toistaen suorituskykymittaukset on lopulta saatu toimimaan luotettavasti ja halutulla tavalla.

Suorituskykymittauksista käy hyvin ilmi, että hitaalla kovalevyllä varustettu palvelin on latausajoiltaan merkittävästi nopeammilla kovalevyillä varustettuja palve-

limia hitaampi. Latausajat myös kasvavat jyrkemmin kuin nopeammalla kovalevyllä ja suuremmilla suoritinresursseilla varustetuilla palvelimilla. Kovalevyn toimintanopeuden lisäksi palvelimen suoritinkapasiteetilla on merkittävä vaikutus palvelimen toimintanopeuteen varsinkin suurempien käyttäjämäärien kuormittaessa palvelinta samanaikaisesti. Palvelimille allokoitujen resurssien valossa mittaustulokset ovat osapuilleen sen suuntaisia, mitä niiden sopii olettaa olevan.

Mittaustulosten perusteella on helppo hahmottaa minkä tasoisilla resursseilla varustettu palvelin tarvitaan kevyehkön Wordpress-verkkopalvelun ylläpitoon, jonka halutaan suoriutuvan esimerkiksi 100 samanaikaisesta kävijästä kelvollisesti ja järkevillä latausajoilla kaikissa tilanteissa. Wordpress-testiympäristöillä suoritettujen testien tulokset eivät ole suoraan vertailukelpoisia muiden julkaisujärjestelmien resurssitarpeiden kanssa, mutta toimivat silti suuntaa antavana vertailukohtana. Lisäksi tutkielmassa kuvatun testiympäristön rakennetta hyödyntämällä vastaavat testit on helppo toteuttaa jonkin muun julkaisujärjestelmän tai verkkokauppaohjelmiston kanssa.

Julkisissa pilvipalveluissa erittäin suorituskykyisten kovalevyjen hinnat ovat hyvin kalliita [35] verrattuna suorituskykymittauksissa käytettyyn dedikoituun palvelimeen NVMe-kovalevyineen, sillä pilvipalveluiden kovalevyjen hinta koostuu tallennuskapasiteetin lisäksi suurimmasta mahdollisesta IOPS-nopeudesta. Se on perustasoisessa levyssä verrattain pieni ja IOPS-nopeuden kasvattaminen nostaa levyn hintaa merkittävästi. Vastaavasti dedikoidussa vuokrapalvelimessa palvelimen kovalevyn koko kapasiteetti on omassa käytössä, eikä suuremmassa IOPS-suorituskyvystä peritä mitään erillistä maksua.

Tämän tutkielman suorituskykymittauksissa ei käytetty purskeittaisella (burst) kapasiteetilla varustettuja virtuaalikovalevyjä, joita monet julkiset pilvipalvelut tarjoavat. Burst-ominaisuudella varustetulla kovalevyllä on tietty peruskapasiteetti, jonka lisäksi se voi hetkellisesti ja rajatun ajanjakson ajan käyttää myös huomattavasti korkeampaa kapasiteettia asetettujen rajoitteiden puitteissa. Tällaiset levyt saattaisivat hyvinkin olla toimiva ratkaisu Wordpress-verkkopalveluja ylläpitävien palvelimien tallennusmediaksi, sillä hiljaisempina aikoina riittää usein hitaampikin kovalevy. Toisaalta reservissä on hyvä olla kapasiteettia ruuhka-aikojen kävijäryntäyksiä ajatellen.

Park ja muut [35] ovat omassa tutkimuksessaan selvittäneet eri tyyppisten kovalevyjen toimintaa Amazon AWS -pilvipalvelussa. Testissä käytettiin AWS:n normaaleja GP2-kovalevyjä, joka on yleisin perustason levytyyppi, joita vertailtiin burst-

ominaisuudella varustettuihin levyihin. Niitä käyttämällä saavutettiin tavalliseen GP2-levyyn nähden monikymmenkertaisia tuloksia, mutta huomattavasti edullisemmalla kustannuksella AWS:n IO1-tyyppiseen, vastaavalla taatulla IOPS-kapasiteetilla varustettuun SSD-levyyn nähden.

Esimerkkinä todettakoon, että tämän tutkielman server3-palvelimessa käytetyn kuluttajatason NVMe-levyn tasolle (200000 IOPS) rajoitetun virtuaalikovalevyn kustannus AWS-pilvipalvelussa IO1-tyyppisenä kovalevynä toteutettuna on huomattavan kallis kustantaen useita tuhansia euroja kuukaudessa. Burst-tyyppisiä levyjä käyttäen voitaisiin päästä jopa yli 90 %:n kustannussäästöihin käytettäessä server3:n kaltaista palvelinta AWS:n pilvessä, joten kyseessä on todella merkittävä ja jokseenkin helposti saavutettavissa oleva rahallinen hyöty.

7 Yhteenveto

Tässä tutkielmassa tutkittiin tallennusmedian suorituskyvyn vaikutusta Wordpress-julkaisujärjestelmän toimintaan. Suorituskykymittaukset suoritettiin kolmella eri tasoisella virtuaalikoneella, joiden kovalevyjen suorituskyvyt on rajattu eri tasoille, jotka jäljittelevät pyörivän HDD-, SSD- ja NVMe-kovalevyn suorituskykyä. Lisäksi suoritinytimiä on allokoitu eri määrä kullekin palvelimelle, jotta resurssien vaikutusta latausaikoihin pystytään paremmin havainnoimaan.

Tutkielman suorituskykymittauksilla pyrittiin löytämään vastauksia siihen miten verkkopalvelun kävijää palvelevan palvelimen tallennusmedian suorituskyky vaikuttaa Wordpress-julkaisujärjestelmän toimintaan. Jo ennen tutkielman aloitusta oli tiedossa, että Wordpress-pohjaisen sivuston tai minkä tahansa verkkopalvelun toimintanopeus hyvin todennäköisesti hidastuu, kun sitä kuormitetaan suuremman käyttäjämäärän toimesta. Tältä osin en siis olettanut näkeväni itselleni uusia asioita tai testituloksia. Myös Page Cache -välimuistin toiminta oli pitkälti tiedossa entuudestaan, mutta sen aiheuttamat ”vääristymät” testituloksissa olivat kuitenkin jonkin verran yllättäviä ja sen myötä alkuperäisiä testejä jouduttiin muokkaamaan.

Suorituskykymittauksissa testataan kolmen eri tasoisilla virtuaalikovalevyillä ja toisistaan poikkeavilla suoritinkapasiteeteilla varustetuille testipalvelimille asennettujen Wordpress-verkkopalvelujen latausaikoja. Testit suoritetaan testipalvelimien kanssa samaan lähiverkkoon sijoitetulle virtuaalikoneelle asennetulla Locust-kuormitustestausohjelmistolla. Mittauksissa Wordpress-verkkopalvelun etusivua ladataan yksittäisellä käyttäjällä ilman Linuxin Page Cache -välimuistia sekä välimuistin kanssa aloittaen yhdellä käyttäjällä, jonka jälkeen käyttäjämäärää kasvatetaan tasaisesti aina 300 käyttäjään saakka. Testien pohjalta koostetaan kuvaajat, joiden avulla latausaikoja on helppo havainnoida. Toinen mittaus suoritetaan samalla kaavalla kuin Wordpress-verkkopalvelun latausaikaa mittaava testi, mutta siinä mitataan PHP-skriptin avulla aikaa joka kuluu, kun levyltä luetaan samat tiedostot kuin Wordpressin etusivun latauksessa itse PHP-koodia suorittamatta.

Mittaustulokset olivat suurimmalta osin sellaisia kuin kuvittelinkin. Hitaimmalla kovalevyllä ja pienimmällä suoritinkapasiteetilla varustetun server1-testivirtuaalikoneen latausajat olivat odotetusti kaikkein hitaimpia. Server2-virtuaalikoneen tu-

lokset olivat nopeamman kovalevyn ja tuplasti suuremman suoritinkapasiteetin ansiosta huomattavasti nopeampia. Vielä nopeammalla kovalevyllä ja server2:een nähden tuplasti suuremmalla suoritinkapasiteetilla varustetun server3-testivirtuaalikoneen olivat vielä hieman nopeampia, mutta erot eivät enää olleet niin merkittäviä kuin server1:n ja server2:n välillä. Kaikkien testivirtuaalikoneiden osalta ilman välimuistia suoritettujen testitulosten hajonta oli hyvin pientä ja latausajat tasaisia. Välimuistin kanssa suoritetun testin latausajat pitenivät tasaisesti kuormituksen kasvaessa, eikä suurempia heittoja suuntaan tai toiseen ilmennyt lainkaan. Kaikki kolme testipalvelinta suoriutuivat 300 samanaikaisen käyttäjän aiheuttamasta kuormasta, mutta latausajat moninkertaistuivat lähtötilanteeseen nähden, jossa käyttäjiä oli vain yksi.

Mahdollisten jatkotutkimuskohteiden osalta näen useampiakin mahdollisuuksia. Nyt tehdyissä testeissä kukin testipalvelin toimi yhdistettynä web- ja tietokantapalvelimena yksittäisenä instanssina kiinteillä resursseilla. Testiä voisi laajentaa siten, että web- ja tietokantapalvelimet eriytettäisiin omille virtuaalikoneilleen, jolloin niiden resurssit voitaisiin allokoida itsenäisesti. Web-palvelimen eteen voisi lisätä kuormantasaajan, joka mahdollistaisi web-palvelimien skaalaamisen useamman instanssin kokonaisuudeksi. Testiympäristöä voisi myös parantaa pystyttämällä sen kokonaisuudessaan Kubernetes-ympäristöön, jolloin kuormantasaus- ja skaalautumisominaisuudet saataisiin toteutettua natiivityökaluilla.

Lähteet

[1] ABDULLAHI, K. What Is NVMe? How Does NVMe Work?, 2024. URL https://recoverit.wondershare.com/hard-drive/what-is-nvme.html, viitattu 6.3.2024.

[2] ADOBE INC. What are PNG files and how do you open them?, 2024. URL https://www.adobe.com/creativecloud/file-types/image/raster/png-file.html, viitattu 14.4.2024.

[3] AGARWAL, A., RAO, S. K. S., JA MAHENDRA, B. Comprehensive review of virtualization tools. *Int. Res. J. Eng. Technol 7*, 6 (2020).

[4] ANDY KLEIN. SSD 101: How to Upgrade Your Computer With an SSD, 2023. URL https://www.backblaze.com/blog/ssd-upgrade-guide/, viitattu 17.2.2024.

[5] AUTOMATTIC INC. Press, 2023. URL https://automattic.com/press/, viitattu 30.11.2023.

[6] BAKITA, J., AHMED, S., OSBORNE, S. H., TANG, S., CHEN, J., SMITH, F. D., JA ANDERSON, J. H. Simultaneous multithreading in mixed-criticality real-time systems. Julkaisusarjassa *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)* (Nasville, Tennessee, USA, 2021), 278–291.

[7] BERNERS-LEE, T. World Wide Web, 1993. URL https://info.cern.ch/hypertext/WWW/TheProject.html, viitattu 12.11.2023.

[8] BIN UZAYR, S. *Mastering Ubuntu: A Beginner's Guide*. CRC Press, Florida, United States, 2022.

[9] BIR, P., KARATANGI, S. V., JA RAI, A. Design and implementation of an elastic processor with hyperthreading technology and virtualization for elastic server models. *The Journal of Supercomputing 76*, 9 (2020), 7394–7415.

[10] BLYTH, A. Digital Recording Methods for Electromechanical Computer Hard Drives (HDD), 2020. URL https://adisarc.com/wp-content/uploads/

2020/04/Digital-Recording-Methods-for-Computer-Hard-Drives.pdf, viitattu 18.11.2023.

[11] CLAES, M., MENS, T., DI COSMO, R., JA VOUILLON, J. A historical analysis of Debian package incompatibilities. Julkaisusarjassa *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories* (Florence, Italia, 2015), IEEE, 212–223.

[12] DOWDEN, M., JA DOWDEN, M. *Architecting CSS: The programmer's guide to effective style sheets*. Apress, 2020.

[13] ESCOBEDO, J. Best Server OS for Your Website: Linux or Windows?, 2023. URL https://www.liquidweb.com/blog/best-operating-system-for-web-hosting/, viitattu 1.1.2024.

[14] GALLI, R. Journal file systems in Linux. *Update 2*, 6 (2001), 50–56.

[15] GENG, H. *Data center handbook*. John Wiley & Sons, Palo Alto, California, USA, 2014.

[16] GYIMESI, P., VANCSICS, B., STOCCO, A., MAZINANIAN, D., BESZÉDES, A., FERENC, R., JA MESBAH, A. Bugsjs: a benchmark of javascript bugs. Julkaisusarjassa *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)* (2019), IEEE, 90–101.

[17] HARRIS, W. What Is an SSD and How Does It Work?, 2024. URL https://computer.howstuffworks.com/solid-state-drive.htm, viitattu 14.4.2024.

[18] HASHICORP. What is Terraform?, 2024. URL https://developer.hashicorp.com/terraform/intro, viitattu 4.2.2024.

[19] HAYDEN, J. Linux server performance: Is disk I/O slowing your application?, 2017. URL https://linuxblog.io/linux-server-performance-disk-io-slowing-application/, viitattu 13.1.2024.

[20] HYPERDB. Readme, 2022. URL https://github.com/Automattic/HyperDB/blob/trunk/readme.txt, viitattu 6.3.2024.

[21] IQBAL, M. I., MEHDI, S., JA WAHEED, I. Hardware Virtualization In Nested Virtualization. *TIJ's Research Journal of Science IT Management - RJSITM* (2014).

[22] KINGSTON TECHNOLOGY EUROPE CO LLP. 2 Types of M.2 SSDs: SATA and NVMe, 2023. URL https://www.kingston.com/en/blog/pc-performance/two-types-m2-vs-ssd, viitattu 14.4.2024.

[23] KOUFATY, D., JA MARR, D. T. Hyperthreading Technology in the Netburst Microarchitecture. *IEEE Micro* (2003).

[24] KOUTOUPIS, P. The Linux ram disk. *LiNUX+ Magazine* (2009), 36–39.

[25] KOZLOWICZ, J. Know Your Storage Constraints: IOPS and Throughput, 2017. URL https://www.lunavi.com/blog/know-your-storage-constraints-iops-and-throughput, viitattu 25.12.2023.

[26] LEE, S. H. Real-time edge computing on multi-processes and multi-threading architectures for deep learning applications. *Microprocessors and Microsystems 92* (2022).

[27] LUKKA, K. Konstruktiivinen tutkimusote, 2001. URL https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/, viitattu 17.1.2024.

[28] MADDEN, M. M. Challenges using Linux as a real-time operating system. Julkaisusarjassa *AIAA Scitech 2019 Forum* (2019).

[29] MARIADB. What is MariaDB?, 2024. URL https://mariadb.com/kb/en/what-is-mariadb/, viitattu 6.3.2024.

[30] MASDARI, M., NABAVI, S. S., JA AHMADI, V. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications 66* (2016), 106–127.

[31] NETCRAFT. January 2023 web server survey. URL https://www.netcraft.com/blog/january-2023-web-server-survey/, viitattu 25.10.2023.

[32] NORDVIK, R. Ext4. Kirjassa *Mobile Forensics–The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*. Springer, 2022, ss. 41–68.

[33] OLEKSIUK, V., JA OLEKSIUK, O. The practice of developing the academic cloud using the Proxmox VE platform. *Educational Technology Quarterly 2021*, 4 (2021), 605–616.

[34] OPENSOURCE.COM. What is Ansible?, 2024. URL https://opensource.com/resources/what-ansible, viitattu 4.2.2024.

[35] PARK, H., GANGER, G. R., JA AMVROSIADIS, G. More IOPS for less: Exploiting burstable storage in public clouds. Julkaisusarjassa *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)* (2020).

[36] PARKKINEN, J. *Hyvään verkkopalveluun!* Infor, Helsinki, 2002.

[37] POST STATUS, LLC. WP Test - The Best Tests For WordPress. URL https://github.com/poststatus/wptest/blob/master/README.md, viitattu 14.10.2023.

[38] PRICE, S. What Is WordPress? A Beginner's Guide, 2023. URL https://blog.hubspot.com/website/what-is-wordpress, viitattu 13.1.2024.

[39] RACK SOLUTIONS, INC. EIA-310: What Does It Mean and is It Still Relevant? URL https://www.racksolutions.com/news/data-center-optimization/eia-310-definition/, viitattu 7.10.2023.

[40] RAHUL, A., JA SILVA, C. HDD form factor (hard disk drive form factor), 2023. URL https://www.techtarget.com/searchstorage/definition/HDD-form-factor-hard-disk-drive-form-factor, viitattu 11.12.2023.

[41] RANDAL, A. The ideal versus the real: Revisiting the history of virtual machines and containers. *ACM Computing Surveys (CSUR) 53*, 1 (2020), 1–31.

[42] RISSANEN, P. Kohti saavutettavaa verkkopalvelua, 2011. URL https://jyx.jyu.fi/bitstream/handle/123456789/40302/URN%3aNBN%3afi%3ajyu-201211132990.pdf, viitattu 12.11.2023.

[43] SEARCH ENGINE JOURNAL. CMS Market Share Trends: Top Content Management Systems In 2022. URL https://www.searchenginejournal.com/cms-market-share/454039/, viitattu 28.9.2023.

[44] SHELDON, R. Storage 101: Data Center Storage Configurations, 2020. URL https://www.red-gate.com/simple-talk/databases/sql-server/database-administration-sql-server/storage-101-data-center-storage-configurations/, viitattu 23.3.2024.

[45] SHRIVASTAVA, S., JA PRAPULLA, S. Comprehensive Review of Load Testing Tools. *International Research Journal of Engineering and Technology 7*, 3392-3395 (2020), 43.

[46] SMALLCOMBE, M. The Complete Guide to FTP, FTPS, SFTP, and SCP, 2023. URL https://www.integrate.io/blog/the-complete-guide-to-ftp-ftps-sftp-and-scp/, viitattu 13.4.2024.

[47] SYMMETRY ELECTRONICS. The Development and History of Solid State Drives (SSDs), 2014. URL https://www.symmetryelectronics.com/blog/the-development-and-history-of-solid-state-drives/, viitattu 11.12.2023.

[48] THE APACHE SOFTWARE FOUNDATION. What is the Apache HTTP Server Project?, 2024. URL https://httpd.apache.org/ABOUT_APACHE.html, viitattu 31.3.2024.

[49] THE PHP GROUP. History of PHP, 2024. URL https://www.php.net/manual/en/history.php.php, viitattu 13.4.2024.

[50] TOUATI, S., WORMS, J., JA BRIAIS, S. *The speedup test*. Tekninen raportti, Université Versailles Saint-Quentin-En-Yvelines (UVSQ), 2010.

[51] TUTORIALS FREAK. Full History of HTML: All Versions, Founder, Latest Version, 2024. URL https://www.tutorialsfreak.com/html-tutorial/html-history, viitattu 13.4.2024.

[52] VEMULAPALLI, R., JA MADA, R. K. Performance of Disk I/O operations during the Live Migration of a Virtual Machine over WAN, 2014. URL http://www.diva-portal.se/smash/get/diva2:829720/FULLTEXT01.pdf, viitattu 10.11.2023.

[53] W3 DOCS. What is mod_php?, 2024. URL https://www.w3docs.com/snippets/php/what-is-mod-php.html, viitattu 13.4.2024.

[54] WISHPOND. The Business Behind WordPress, 2022. URL https://www.linkedin.com/pulse/business-behind-wordpress-wishpond/, viitattu 30.11.2023.

[55] WORDPRESS.ORG. Learn about WordPress origins and version history, 2023. URL https://wordpress.org/documentation/article/learn-about-wordpress-and-version-history/, viitattu 28.1.2024.

[56] WORDPRESS.ORG. The WordPress Codebase, 2023. URL https://make.wordpress.org/core/handbook/contribute/codebase/, viitattu 28.1.2024.

[57] WORDPRESS.ORG. Using Alternative Databases, 2024. URL https://codex.wordpress.org/Using_Alternative_Databases, viitattu 11.2.2024.

[58] XU, W. Deep Into Linux And Beyond - Page Cache, 2022. URL https://wxdublin.gitbooks.io/deep-into-linux-and-beyond/content/page_cache.html, viitattu 20.3.2024.