Opinnäytetyö (AMK)

Tieto- ja viestintätekniikan koulutus

2021

Saara Alho

RASPBERRY PI -KLUSTERI WWW-PALVELIMENA

– suunnittelu, toteutus ja testaus

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan koulutus

2021 | 31 sivua, 3 liitesivua

Saara Alho

RASPBERRY PI -KLUSTERI WWW-PALVELIMENA

- suunnittelu, toteutus ja testaus

Tämän opinnäytetyön tarkoituksena oli suunnitella ja rakentaa edullinen WWW-palvelin opiskelijakäyttöön sekä lisäksi tutkia, riittäisikö muutaman Raspberry Pin klusteripalvelimesta tarpeeksi tehoa sitä varten.

Työ aloitettiin tutkimalla, mikä on klusteripalvelin ja esitettelemällä myös vaihtoehtoisia edullisia palvelintyyppejä klustereille. Lisäksi etsittiin, mitä tutkimuksia ja toteutuksia klusteripalvelimiin liittyen löytyi jo valmiina. Raspberry Pi -laitteista, suositusta yhden piirilevyn tietokoneesta, koostuvia klustereita on toteutettu paljon jo silloin, kun laitteet olivat vielä nykyiseen verrattuna erityisen matalatehoisia. Toisaalta palvelimena toimivia toteutuksia löytyi vain muutama, ja näissä laitteita on käytetty huomattavasti enemmän kuin mitä tässä työssä oli tarkoitus käyttää.

WWW-palvelimen toteutus aloitettiin suunnitelmalla, jossa valittiin ja esitettiin sekä laitteet että ohjelmistot, joita oli tarkoitus käyttää rakennettavassa klusteripalvelimessa. Kriteerinä oli helppokäyttöisyyden lisäksi myös ohjelmiston keveys, jotta laitteina käytettävien Raspberry Pi -koneiden rajallisista resursseista jäisi mahdollisimman paljon itse WWW-palvelimen pyörittämiseen. Valituista ohjelmistoista klusteriohjelmistoksi valittiin Docker Swarm, sekä WWW-palvelimeksi vähän resursseja käyttävä NGINX. Tavoitteena oli, että palvelin kestäisi noin 200 käyttäjän kuorman.

Varsinainen toteutus oli melko mutkaton prosessi, jossa pääasiassa Raspberry Pit yhdistettiin toisiinsa kytkimen kautta ja niille asennettiin ja konfiguroitiin valitut ohjelmistot. Valmiin palvelimen testaus ApacheBench-ohjelmistolla osoitti, että kolmen laitteen klusteri riittää opiskelijakäyttöön hyvin.

ASIASANAT:

Palvelin, Klusteri, Raspberry Pi, Docker Swarm

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and communications technology

2021 | 31 pages, 3 pages in appendices

Saara Alho

RASPBERRY PI CLUSTER AS A WEB SERVER

- planning, implementation and testing

The purpose of this thesis was to plan and build an inexpensive web server for student use and study whether a cluster of few Rasperry Pis is powerful enough for such purpose.

The first part of this thesis explains what a cluster server is and also showcases other types of affordable servers. A cluster is a unit that consists of computers capable of communicating and working together on a common task and a cluster server describes the state when these computers work as servers. Clusters created using Raspberry Pi, a popular single-board computer, are common and were implemented even when the boards were still very inefficient when compared to the newest models. On the other hand, out of these projects only a few were found that work as servers, and most of them use significantly more Raspberry Pis than the Raspberry Pis that were used in this thesis.

The implementation of this thesis work began with a plan in which the used hardware and software were chosen. The criteria for this implementation were the simplicity of use and the lightness of the software, mostly because of the limited resources that should be saved for the use of the server itself. The chosen software include Docker and specifically the Swarm mode for clustering, and NGINX for the webserver part.

The actual building of the server was a fairly straightforward process where the main steps were connecting the Raspberry Pis using a switch and installing and configuring the software. The testing of the finished server using ApacheBench indicates that the three-device cluster is adequate for student use.

KEYWORDS:

Server, Cluster, Raspberry Pi, Docker Swarm

SISÄLTÖ

LIITTEET

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET

PoE	Virta Ethernetin kautta (Power over Ethernet)
AWS	Amazonin verkkopalvelut (Amazon Web Services)
RAM	Luku- ja kirjoitusmuisti (Random Access Memory)
ARM	Prosessoriarkkitehtuuri
OS	Käyttöjärjestelmä (Operating System)
GUI	Graafinen käyttöliittymä (Graphical User Interface)
HTTP	Hypertekstin siirtoprotokolla (HyperText Transfer Protocol)
TLS	Protokolla salausta varten (Transport Layer Security)

1 JOHDANTO

Raspberry Pi on yhden piirilevyn tietokone, joka tuli markkinoille vuonna 2012. Koska se on edullinen ja monipuolinen, monet harrastelijat ovat käyttäneet sitä projekteissaan jo julkaisusta alkaen. Raspberry Pi on pienoiskokoinen tietokone, jolla voi tehdä melkein mitä tahansa samoja asioita kuin standardikokoisella koneellakin. Sille valmistetaan tuhansia eri lisävarusteita aina kameramoduuleista RFID-lukijoihin, joten se inspiroi harrastelijoita keksimään koko ajan uusia projekteja ja käyttötarkoituksia laitteelle. Jo ensimmäisistä versioista lähtien ihmiset ovat pyrkineet selvittämään, mitkä ovat näiden käyttömahdollisuuksien rajat. Uusien, tehokkaampien mallien ilmestyessä ovat yhden piirilevyn tietokoneet alkaneet jo olla kykeneväisiä asioihin, joihin vanhat mallit eivät vielä pystyneet.

Tämän opinnäytetyön tavoitteena on selvittää, kuinka voisi toteuttaa edullisen WWW-palvelimen Turun ammattikorkeakoulun Salon insinöörikoulutuksen opiskelukäyttöön, sekä tutkia, riittääkö 2–4 Raspberry Pi 4:stä koostuvasta klusteripalvelimesta tarpeeksi suoritustehoa palvelinkäyttöön.

Teoriaosuudessa käsitellään sitä, mikä on klusteripalvelin, esitellään muiden jo toteuttamia Raspberry Pi -klustereita sekä vaihtoehtoisia matalakustanteisia palvelintyyppejä.

Teoriaosuuden tietojen pohjalta on tarkoitus ensin suunnitella WWW-palvelimena toimiva klusteripalvelin ja sen jälkeen esitellä opinnäytetyön käytännön osuudessa, kuinka sellaisen voi rakentaa.

2 PALVELIMET

Palvelimella tarkoitetaan tietokonetta, tietokoneella pyörivää ohjelmistoa tai näiden molempien yhdistelmää, joka tarjoaa palveluja joko lokaalisti tai internetin kautta asiakkaille eli toisille laitteille tai ohjelmistoille [1]. Tämä asiakas-palvelinarkkitehtuuri on useimmiten toteutettu siten, että asiakas lähettää pyynnön, johon palvelin reagoi ja vastaa mallilla pyyntö-vastaus (engl. request-response).

Palvelimilla on lukuisia eri tehtäviä ja käyttötarkoituksia, joihin on olemassa oma palvelintyyppinsä. Yleisimpiin käytettyihin palvelimiin lukeutuvat esimerkiksi WWW-palvelimet, jotka pyörittävät verkkosivuja, tietokantapalvelimet, jotka ylläpitävät tietokantoja sekä sähköpostipalvelimet, joiden tarkoituksena on välittää viestit lähettäjältä vastaanottajalle.

2.1 Klusteripalvelin

Termi klusteri tulee englannin kielen sanasta ”cluster”, joka suoraan suomennettuna tarkoittaa rykelmää tai ryhmää. Nimensä mukaisesti klusteripalvelimella tarkoitetaan arkkitehtuuria, johon kuuluu useampi tietokone, jotka usein toimivat yhden IP-osoitteen alaisuudessa muodostaen yhden palvelinyksikön käyttäjän näkökulmasta [2]. Yleinen toimintamalli klusterissa on se, että yksi kone on vastuussa kommunikaatiosta ja tehtävien jakamisesta muille koneille, tai jokin variaatio tästä toimintatyypistä. Jokainen klusteriin kuuluva tietokone muodostaa yhden noden eli solmun, jossa täytyy olla tietty ohjelmisto ja kytkentä toisiin solmulaitteisiin ollakseen osa klusteripalvelinta. Klusterissa jokaisen solmun ohjelmiston täytyy olla sama.

Eri klusterityyppejä on monenlaisia, mutta niistä kolme tärkeää päätyyppiä ovat ”load balancing” -klusterit, ”high availability” -klusterit sekä ”high performance” -klusterit. Näille englanninkielisille nimityksille ei ole virallisia suomenkielisiä vastineita, mutta vapaasti käännettynä ne ovat kuormantasaus-klusterit, korkean tavoitettavuuden klusterit sekä tehokkaat klusterit [3], [4]. Yksi klusteripalvelin voi hyödyntää yhtä tai montaa eri klusterityyppiä.

Load balancing -klusterissa kuormantasausalgoritmi jakaa palvelimelle tulevat pyynnöt klusterin eri palvelinsolmuille työkuorman helpottamiseksi, usein vähiten kiireiselle [3],

[4]. Monet load balancing -klusteripalvelimet hyödyntävät lisäksi fail-over-teknologiaa, jossa yhden klusterin solmun pettäessä tai kaatuessa muut vapaana olevat laitteet ottavat sen paikan pitäen palvelimen edelleen pystyssä.

High availability -klusterista käytetään joskus nimitystä fail-over ja toisin päin. Aina nämä kaksi eivät kuitenkaan välttämättä tarkoita samaa asiaa. Sekä fail-overissa että high availability -palvelimessa tavoitteena on sen vikasietoisuus – fail-overilla tarkoitetaan sitä, että toiset tietokoneet ottavat toimimattoman solmun paikan, mutta high availability voi sisältää myös muita tapoja eliminoida vika-alttiuksia ja siten toteuttaa korkean tavoitettavuuden klusteri [3].

High performance- klusterin tarkoitus on nopeuttaa erityisesti laskutehoa [3]. Tämä on hieman harvinaisempi klusterityyppi kuin load balancing tai high availability, mutta high performance -klustereita käytetään erityisesti supertietokoneiden kanssa sekä suurien datamäärien kanssa työskennellessä.

Näiden lisäksi joskus mukaan lasketaan myös ”storage”- klusterit eli tallennustilaklusterit jotka yleensä ovat load balancing -teknologiaa hyödyntäviä, suurta määrää tallennustilaa sisältäviä klustereita [5].

Näihin kaikkiin klusterityyppeihin on kohdistunut paljon tutkimustyötä, jolla pyritään parantamaan erityisesti toimintanopeutta ja tehokkuutta. Esimerkiksi erityisesti kuormantasausalgoritmeista on esitetty monenlaisia parannettuja ja uusia versioita.

Klusterin eduksi lasketaan usein modulaarisuus; palvelimen kokoa on yksinkertaista skaalata joko lisäämällä tai poistamalla tietokoneita klusterista. Toinen klusteripalvelimen hyvä puoli on sen luotettavuus; klusteripalvelin on vikatilanteissa varmempi kuin yhdestä tietokoneesta koostuva palvelin, sillä yhden solmun pettäessä muut pitävät palvelimen edelleen pystyssä. Yksin toimivassa palvelinlaitteessa yhden osan vioittuminen voi kaataa palvelimen. Tässä voi säästää myös rahaa, koska palvelimen kaatuminen voi tarkoittaa yrityksille käyttökatkon pituudesta riippuen isojakin menetyksiä, jolloin klusteripalvelimen vikasietoisuus on houkutteleva vaihtoehto.

Klusteri voi olla muillakin tavoin edullisempi vaihtoehto: yksi palvelinkone voi maksaa useita tuhansia dollareita, mutta klusterin pystyy rakentamaan useista halvoista tietokoneista jolloin hankintakulut ovat pienemmät. Riippuen siitä, millaisia koneita klusterissa käyttää, voi energiankulutus myös olla pienempi.

2.2 Raspberry Pi -klusterit

Ajatus matala-tehoisten tietokoneiden klusteroinnista ei ole uusi. Erityisesti Raspberry Pi -klustereista on lukuisia variaatioita ja siten myös ohjeita ja kirjallisuutta, joista moni tosin keskittyy laskentatehon klusterointiin, mikä voi poiketa joiltain osin palvelinklusteroinnista. Jo vuonna 2013, kun Raspberry Pi -koneista oli tullut vasta kaksi ensimmäistä mallia, B ja A, julkaistiin ohjekirja *Raspberry Pi super cluster* oman ”superklusterin” rakentamisesta Raspberry Pi B:llä [6]. Nykyään Raspberry Pi on jo moninkertaisesti tehokkaampi, ja niin ovat myös siitä tehdyt klusterit. MagPi, virallinen Raspberry Pi -lehti, kirjoitti vuonna 2019 artikkelin, kuinka voidaan rakentaa itse klusteritietokone käyttämällä Raspberry Pi 4:ää ja erikseen näiden klusterointia varten valmistettavaa telinesuojusta [7].

Raspberry Pi -klusterin edut verrattuna perinteisiin klustereihin ovat laitteen hinta, sen koko ja sen virrankulutus. Nämä seikat puoltavat sitä, että Raspberry Pi -klusterin voi tehdä kuka tahansa, melkein missä tahansa. Klusteria on myös helppo siirtää paikasta toiseen.

Yhden piirilevyn tietokone -klustereista kertova artikkeli *Commodity single board computer clusters and their applications* listaa myös näiden heikkoudet verrattuna perinteisiin klustereihin: Jokaisessa solmussa on rajoitetut laitteistoresurssit, laitteiston hajoamistahti on suurempi, verkon latenssi on korkea ja nämä klusterit tarvitsevat nimenomaiselle arkkitehtuurille tarkoitetut kääntäjät sekä epästandardisoidut työkalut [8]. Artikkeli tosin on vuodelta 2018, jolloin uusin Raspberry Pin malli oli 3+. Raspberry Pin malli 4 on parantanut tilannetta verkon latenssin kanssa, sillä siinä on 1 Gb:n Ethernet, jossa ei ole sisällä USB2:ta, joka 3+:ssa rajoitti suoritustehoa. Näistä artikkelin esittämistä ongelmista klusterin rajoitetut laiteresurssit sekä vikojen ilmentymisen nopeuden voi ainakin osin ratkaista klusterin kokoa laajentamalla. Mitä enemmän on solmuja, sitä enemmän palvelimessa on yhteensä tehoa ja sitä pienempi vaikutus sillä on, jos jokin solmu menee rikki. Tietysti riippumatta siitä, onko klusterissa kolme vai kolmekymmentä solmua, täytyy rikki mennyt laite jossain vaiheessa korvata, mutta jos verrataan Raspberry Pi -konetta tavalliseen tietokoneeseen, ei vaiva ole erityisen suuri. Yhden laitteen hinta on vähäinen, ja Raspberry Pi kertoo avoimesti, kuinka pitkään sen malleja luvataan valmistaa. Esimerkiksi Raspberry Pi 4 model B poistuu tuotannosta aikaisintaan tammikuussa 2026 [9], joten siihen asti on saatavilla vielä korvaava laite.

Joitain ongelmia on kuitenkin myös isommilla klustereilla. Yksi näistä on virtakaapelit. Neljän laitteen klusterissa ei ole ongelmaa siinä, että niistä jokainen täytyy kytkeä erikseen seinään kiinni, mutta siinä vaiheessa kun klusterin koko ylittää kymmenen solmua, voi olla että täytyy miettiä vaihtoehtoisia ratkaisuja jatkojohdoille. Power over ethernet eli virta ethernetin kautta on yleinen ratkaisu tähän, mutta se vaatii kytkimen, joka tukee PoE:tä sekä lisäosan Raspberry Pihin. Toinen ongelma on ympäristö. Laitteiden säilytykseen pitää käyttää enemmän vaivaa, ja viilennys on otettava huomioon.

2.3 Vaihtoehtoisia matala-kustanteisia palvelimia

Riippuu täysin siitä, mihin palvelinta tarvitaan ja kuinka pitkään, jotta voidaan päätellä, mikä olisi paras ratkaisu. Tässä esitellään harrastelijatarpeille tapoja hankkia halpa palvelin.

Erillisellä tietokoneella suoritettava palvelin

Yleisin tapa, jolla ihmiset tekevät palvelimia kotikäyttöön, on ottaa vanha kone ja tehdä siitä palvelin [10]. Nykyään usealla on kaapissa vanha kannettava tietokone, jolla ei enää ole käyttöä, tai tietokoneen osia jotka ovat pöytäkoneen päivittämisen jälkeen jääneet turhiksi. Näistä tehdään yleensä Linux-pohjaisia palvelimia, jotka saavat olla jossain kodin nurkassa. Nämä ovat yleensä hyviä ihmisille, joiden projektit vaativat huomattavaa ja jatkuvaa palvelimen käyttöä tai niille, joilla on tarve jakaa helposti tiedostoja eri tietokoneiden välillä. Tällaisessa ratkaisussa on kuitenkin joitain asioita, joita kannattaa ottaa huomioon.

Yksi näistä on virrankulutus. Oman palvelimen ylläpitäminen ei nimittäin ole ilmaista vaikka laitteisto olisikin jo omasta takaa. Jos palvelin on käynnissä kellon ympäri, näkyy sen aiheuttamat kustannukset viimeistään sähkölaskussa. Keskimääräisen kerrostaloasujan sähkö maksoi vuoden 2021 tammikuussa 9,46 snt/kWh ja sähkönsiirto 10 snt/kWh, laskematta mukaan sähköveroa [11]. Näiden tietojen avulla voi laskea, kuinka paljon palvelimen ylläpitäminen voi käytännössä maksaa, kaavalla *laitteen sähkönkulutus (kW/h)* × *(sähkön hinta (€/kWh)* + *sähkön siirtohinta (€/kWh))* × *tunnit päällä* × *päivät päällä*. Jos palvelin olisi päällä 24 tuntia vuorokaudessa, tulisi kuukaudessa (30 päivää) sille hintaa noin kahdeksan ja puoli euroa jos kyseessä olisi

kannettava tietokone, jonka kulutus olisi keskimäärin 60 W/h ja noin neljätoista euroa mikäli kyseessä olisi pöytätietokone, jonka keskimääräinen kulutus olisi 100 W/h.

Toinen vaikutus on ympäristöön. Palvelimen ollessa rankassa käytössä lämpenee laitteen lämpötila jolloin se joutuu viilentämään itseään komponenttien vioittumisen estämiseksi. Tämä voi tehdä palvelimesta jäähdytysjärjestelmästä riippuen äänekkään.

Koska äänihaitan mahdollisuus on olemassa ja vaikutus sähkölaskuun joissain tapauksissa huomattava, ei kaikille oma palvelin ole välttämättä paras vaihtoehto.

Vuokrattu palvelintila, pilvipalvelut

Joskus voi olla halvinta ja helpointa vuokrata palvelin. Jos tarvitsee palvelintilaa, jotta voisi käyttää moninpelipalvelinta hetken aikaa, kannattaa sellainen vuokrata. Esimerkiksi Minecraft-pelipalvelinta vuokrataan netissä alle kymmenen euron kuukausihintaan, ja Counter-Strike-pelipalvelinta noin viiden euron kuukausihintaan [12]. Muuta satunnaista käyttöä varten palvelin, jossa maksaa sen mukaan, kuinka paljon sitä käyttää, voi olla hyvä.

Aina ei vuokratusta palvelimesta tarvitse edes maksaa. Amazonin verkkopalvelut eli AWS tarjoaa useista pilvipalveluistaan ilmaista versiota tai vuoden ilmaista kokeiluaikaa, joka on riittävä moneen harrastelu- tai satunnaiskäyttöön [13]. Microsoftin Azure tarjoaa samankaltaisia palveluja kuin AWS [14]. Kuten AWS:llä, myös Azurella on ilmaiset kokeilujaksot joihinkin palveluihin ja tietyt palvelut ovat aina ilmaisia. Opiskelija voi saada Azurelle rekisteröityään lisäksi 100 dollarin krediitin. Nämä palvelut sopivat esimerkiksi pieniin projekteihin ja oppimiskäyttöön.

Virtuaalikonepalvelin

Virtuaalikone tarkoittaa fyysisellä koneella ohjelmiston avulla simuloitavaa tietokonetta, joka toimii käyttämällä isäntäkoneen laitteistoa ja laskentavoimaa. Virtuaalikonetta luotaessa voidaan määrittää, kuinka paljon isäntäkoneen resursseja se voi viedä.

Virtuaalikoneiden hyöty on siinä, että yhdellä tietokoneella voi olla monta käyttöjärjestelmäinstanssia, isäntäkäyttöjärjestelmä sekä virtuaalikoneiden pyörittämät vieraskäyttöjärjestelmät [15]. Halutessaan voi jokaiselle vieraskäyttöjärjestelmälle laittaa

oman palvelimensa. Jos käytössä on vain yksi tietokone ja tarvitsee useamman palvelimen, voi virtuaalipalvelin olla vaihtoehto. Virtuaalikoneet käyttävät isäntäkoneen resursseja, joten mitä enemmän instansseja on käynnissä, sitä vähemmän tehoa ja laskentavoimaa yhdellä virtuaalikoneella voi olla käytettävissään. Siksi joko isäntäkoneen pitää olla erityisen tehokas, virtuaalikoneiden määrän tarpeeksi pieni tai virtuaalipalvelimien niin matalatehoisia, että isäntäkoneen laskentavoima riittää niiden ylläpitämiseen.

3 KLUSTERIPALVELIMEN SUUNNITTELU

WWW-palvelimena toimiva klusteripalvelin koostuu laitteista ja ohjelmistoista, jotka valitaan ennen varsinaista toteutuksen aloittamista.

3.1 Raspberry Pi

Tässä WWW-palvelimessa on laitteistona tarkoitus käyttää Raspberry Pi Foundationin Raspberry Pi 4 Model B:tä, mikä on vuonna 2018 julkaistu yhden piirilevyn pieni, noin 85 mm × 56 mm kokoinen tietokone, jota edelsi suosittu Raspberry Pi 3. Raspberry Pi 4:ää on saatavilla kolmea eri mallia: 2GB luku- ja kirjoitusmuistia eli RAM:a tarjoava malli, 4GB RAM-malli sekä 8GB RAM-malli. RAM:a lukuun ottamatta jokaisessa mallissa on sama sisältö: tärkeimpänä 1.5 GHz neljä-ytiminen ARM-arkkitehtuurin prosessori ja yksi gigabit-Ethernet-paikka [9].

Raspberry Pi 4 on kalleimmillaankin halvempi kuin moni tavallinen tietokone: Halvin malli eli 2GB RAM on 35 Yhdysvaltojen dollaria ja kallein, 8GB RAM, 75 dollaria. 4GB:n malli on 55 dollaria. Raspberry Pin pakkauksessa mukana ei kuitenkaan tule virtalähdettä tai microSD-korttia, jolle käyttöympäristö tulisi asentaa [16].

3.2 Ohjelmistojen valinta

Raspberry Pi -laitteiden klusterissa ohjelmistoilla on enemmän väliä kuin jos koneet olisivat tehokkaampia. Koska jokaisessa laitteessa on rajalliset resurssit ja näiden klusterissa on todennäköisesti useitakin pullonkauloja, kannattaa ohjelmistojen olla mahdollisimmat kevyet, jotta resursseja säästyy palvelimen käyttöä varten.

Käyttöjärjestelmä

Raspberry Pi Foundationin tarjoama Raspberry Pi OS, aikaisemmin Raspbian, on usein suositelluin Raspberry Pille asennettava käyttöympäristö, koska se on ainoa Raspberry Pin virallinen käyttöjärjestelmä. Raspberry Pi OS:n desktop-versio on kuitenkin suhteellisen raskas pyörittää pienitehoisella tietokoneella, muuan muassa kaikkien

mukana tulevien valmiiksi asennettujen ohjelmistojen takia. Tarkoituksena on tehdä mahdollisimman tehokas klusteripalvelin, jolloin on hyvä löytää kevyempi käyttöjärjestelmä. Raspberry Pi OS:stä on olemassa Lite-versio, jonka asennuspaketti on kooltaan 480MB verrattuna desktopin riisutumman version 1171MB:seen ja suositeltujen sovellusten kanssa tulevan version 2863MB:seen [17]. Näiden suurin eroavaisuus toisistaan on se, että Litestä puuttuu Desktop-versiosta löytyvä graafinen käyttöliittymä eli GUI, jonka takia Lite vie vähemmän laitteen resursseja mutta aiheuttaa myös sen, että Lite ei käyttöjärjestelmänä ole yhtä aloittelijaystävällinen kuin Desktop. Toinen kevyt vaihtoehto on Debianista kehitetty DietPi, joka tarjoaa läpinäkyvästi myös useiden laitteiden mittaustestitulokset [18]. Koska suurin kriteeri ohjelmistovalinnassa on kevyys, lähdetään palvelinta rakentamaan DietPin päälle.

Klusteri-ohjelmisto

Tämän palvelimen tarkoitus olisi hyödyntää load balancing- sekä fail-over-teknologiaa.

Microsoft on tarjonnut Microsoft Cluster Server eli MCSC:ää joka myöhemmin nimettiin uudelleen Microsoft Server Failover Clustering:ksi eli lyhyesti MSFC:ksi. Tämän ohjelmiston huono puoli on se, että palvelintietokoneet tarvitsevat Windows Server -alustan toimiakseen, eli käyttöympäristö on maksullinen ja rajoittaa huomattavasti [19].

Docker on avoimen lähdekoodin ohjelmisto, jonka Docker Swarm -ominaisuus tarjoaa sekä high availability- että load balancing -klusteriratkaisuja monille eri käyttöalustoille [20]. Googlen kehittämä Kubernetes tarjoaa hyvää vikasietoisuutta, erinomaisia klusterin skaalausmahdollisuuksia ja Dockerista täysin puuttuvaa klusterin monitorointia ja lokikirjausta [21], mutta sen käyttäminen on huomattavasti monimutkaisempaa kuin Dockerin. Eli Docker Swarm on se, jolla pyritään tässä toteutuksessa klusteroimaan Raspberry Pit.

HTTP-palvelin

HTTP-palvelin eli WWW-palvelin suorittaa nettisivuja. NGINX Open Source on vapaan lähdekoodin HTTP-palvelin, joka tarjoaa myös load balancing -mahdollisuuksia. Se toimii asynkronisesti tapahtumavetoisesti (engl. event-driven), mikä kuluttaa vähemmän palvelinlaitteen resursseja kuin esimerkiksi Apachen HTTP-palvelin, jossa jokainen

pyyntö palvelimelle vaatii oman säikeensä [22]. Raspberry Pi 4:ssä on neliytiminen prosessori, mikä ei ole Apachea ajatellen erityisen hyvä, erityisesti ottaen huomioon sen, että klusteri on suhteellisen pienikokoinen joten säikeiden kokonaismääräkään ei ole välttämättä riittävä. NGINX sopii arkkitehtuuriltaan hyvin matalatehoiseen laitteistoon. Open Source -versio on ilmainen, mutta NGINX tarjoaa lisäksi maksullisia palveluita ja sovelluksia. NGINX on erityisesti hyvä staattisia nettisivuja varten.

3.3 Palvelimen laajuus

WWW-palvelimen tehokkuuden voi määrittää muutaman asian avulla. Näihin lukeutuvat muun muassa kuinka nopeasti se vastaa, kuinka nopeasti se lataa nettisivun, kuinka paljon resursseja se käyttää. Eri käyttötarkoituksia varten WWW-palvelimella on eri standardit. Koska tämä palvelin tulisi opiskelijakäyttöön, ei nopeus ole kriittinen mutta mahdollisimman nopeaan on silti hyvä pyrkiä. Tarpeeksi tehokas siis voisi olla tässä työssä keskinopea palvelin, joka kestäisi pahimmassa tapauksessa maksimimäärän opiskelijoita. Tavoitteena olisi, että palvelimen vastausaika (engl. response time) olisi noin 200 ms tai alle.

Suunnittelulle tärkeää on siis selvittää, kuinka paljon opiskelijoita voisi parhaimmassa tapauksessa olla, jotta voidaan selvittää kuinka laajasti WWW-palvelimen tulisi palvella. Koska tämän palvelimen ideana olisi toimia Salon kampuksen insinöörikoulutuksen opiskelukäytössä, täytyy opiskelijoiden määrä selvittää pääpiirteittäin. Vuonna 2019 pilotin käynnistyessä opiskelijoita aloitti hieman alle 30 opiskelijaa [23]. Turun ammattikorkeakoulun alkuvuodesta 2021 julkaiseman uutisen perusteella opiskelijamäärä saattaa kuitenkin kasvaa. Tavoitteena olisi noin 30 opiskelijaa Uzbekistanissa sijaitsevan yliopiston ja Turun ammattikorkeakoulun kaksoiskoulutukseen, mikä on siis normaalin koulutuksen lisäksi [24]. Näiden perusteella voisi arvioida Salossa siis joskus mahdollisesti olevan noin 50 insinööriopiskelijan vuosikursseja. Insinöörikoulutus on suunniteltu ammattikorkeakoulussa nelivuotiseksi ja kaksoistutkintoa tekevät suorittaisivat siitä Salossa kolme vuotta, joten arvio voisi olla, että tulevaisuudessa opiskelijoita olisi noin 170 kerrallaan. Kaikki näistä hypoteettisista opiskelijoista eivät todennäköisesti samaan aikaan palvelinta käyttäisi, mutta sen olisi silti hyvä kestää noin parin sadan samanaikaisen käyttäjän kuorma. Tällöin palvelinta ei tarvitsisi aivan heti laajentaa, vaikka käyttäjien määrä tulevaisuudessa kasvaisikin. Käyttäjämäärän lisäksi palvelimeen voi vaikuttaa myös se, mitä palvelin suorittaa. Jotkin

nettisivut vievät enemmän resursseja kuin toiset, mutta yleisesti ellei sivusto ole erityisen raskas, pitäisi vaikutuksen olla melko pieni.

Etukäteen on hankala arvioida, kuinka monta Raspberry Pi -konetta klusteri vaatii tukeakseen noin kahdensadan käyttäjän maksimia. Oletetaan, että palvelimella on yksi nettisivu, jossa ihmiset klikkaavat jotain tai muuten lataavat sivua uudestaan kaksi kertaa minuutissa, ja että sivu on raskas tai palvelimen nopeus keskinkertainen ja sivun lataus kestäisi 1000ms. Yhdessä Raspberry Pi 4:ssä on neljä ydintä. Tällöin yksi ydin voi sekunnissa ladata sivun keskimäärin kerran. Jos pyyntöjä lähetetään palvelimelle 2 kertaa minuutissa, tarvittaisiin kahdeksan ydintä, että palvelin jaksaisi yli 200 käyttäjän kuorman. Tämä tarkoittaisi kahta Raspberry Pi -laitetta. Tämä on kuitenkin kaikki arvailua; palvelimen todellisen maksimikuorman ja tehokkuuden voi mitata vasta, kun sen on rakentanut ja tietää, millainen sen suorittama nettisivu on. Tässä arviossa ei myöskään otettu huomioon, kuinka paljon RAM:a yksi pyyntö vie – lisäksi nopeuteen voi vaikuttaa vielä moni asia, kuten tallennustila, paikallinen verkon nopeus tai muu laitteisto. Myös muut Raspberry Pi -laitteella olevat ohjelmistot voivat viedä resursseja, joita palvelimella ei siinä tapauksessa ole enää käytettävissään. Näistä epävarmuustekijöistä johtuen arvio on kahden sijaan kolme Raspberry Pi 4 -konetta.

3.4 Suunnitelman yhteenveto

Suunnitelman mukaan toteutusta varten tarvitaan seuraavat:

- 3 Raspberry Pi 4:ää, mielellään 4GB tai 8GB RAM-mallia
- 3 microSD-korttia sekä SD-kortin lukija
- virtalähteet Raspberry Pi -koneille
- 3 ethernet-kaapelia, mielellään cat 5/6
- kytkin mielellään 1 Gb:n Ethernet-paikoilla.

Laitteiden ja kaapeleiden olisi hyvä tukea gigabitin nopeutta, jotta palvelimen nopeus ei turhaan hidastuisi niiden puolelta. Vaikka Raspberry Pi 4:llä maksiminopeus on 1 Gb/s, olisi se saavuttamaton mikäli esimerkiksi Ethernet-johto sattuisi olemaan vääränlainen. Tämän takia kaapeleiden valinta on melkein yhtä tärkeä kuin itse laitteen tai sen ohjelmistojen, vaikka niitä ei heti ensimmäiseksi tulisi ajatelleeksikaan.

Klusteri on tarkoitus toteuttaa manager-worker-arkkitehtuurilla, jossa yksi solmu toimii tehtävänjakajana muille. Jokaiseen Raspberry Pi -laitteeseen ladataan DietPin

Raspberry Pi -koneita varten tarkoitettu 32-bittinen käyttöjärjestelmä, mikäli sen kanssa tulee ongelmia voidaan vaihtoehtoisesti asentaa Raspberry Pi OS Lite. Klusterointia varten asennetaan Dockerin Swarm-tila, ja Dockerin container laitetaan sisältämään NGINX-palvelin. Kun toteutus on saatu tehtyä, testataan palvelimen tehokkuus sekä mahdollisen fail-over-mekaniikan toimivuus.

4 KLUSTERIPALVELIMEN RAKENTAMINEN

Toteutuksessa käytetyt laitteet sisältävät kolme Raspberry Pin 4GB RAM -mallia, EdgeRouter X:n, neljä cat 5 Ethernet-kaapelia, yhden SD-kortin, kolme 8GB USB 3.0 -muistitikkua sekä virtalähteet Raspberry Pi:lle. Lisäksi käytössä on myös SD-kortin lukija, HDMI – microHDMI adapteri sekä näyttö että näppäimistö, joita ei tarvitse enää ssh-yhteyden luomisen jälkeen.

4.1 Toteutus

Kytkimen käyttöönotto

Kytkimenä toteutuksessa toimii EdgeRouter X. Se ei tehdasasetuksilla toimi suoraan, toisin kuin monet kotikäyttöön tarkoitetut kytkimet. Edgerouterin hallintasivulla voi suorittaa setup wizardilla paketin basic setup, joka on riittävä tähän tarkoitukseen. Edgerouter ehdottaa automaattisesti IP-osoitteeksi 192.168.1.1/24, mutta tätä voi joutua muokkaamaan hieman joissain tapauksissa. Esimerkiksi tässä toteutuksessa kotiverkon reitittimen IP-osoite on sama, mikä voisi aiheuttaa konflikteja jos oletukset pidettäisiin ennallaan.

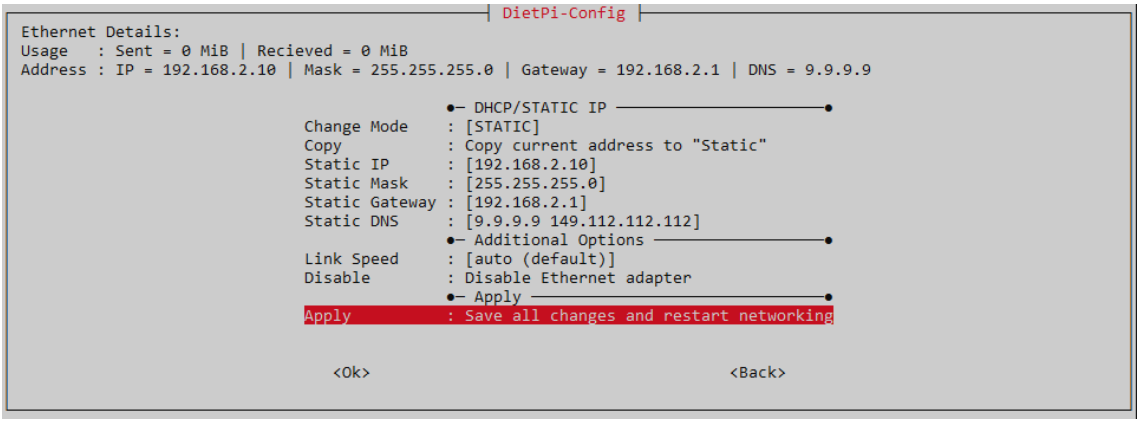
Käyttöjärjestelmän asennus

Yksi poikkeus suunnitelmasta on vaihtaa SD-kortit USB-tikkuihin. SD-kortit korruptoituvat helpommin kuin muistitikut, ja muistitikkuja on helpompi myöhemmin päivittää kuin SD-kortteja. Yhtä microSD-korttia kuitenkin tarvitaan siihen, että Raspberry Pi:n saa käynnistämään itsensä USB-tikulta. Uusimmat Raspberry Pi -laitteet pystyvät automaattisesti tunnistamaan, mikäli käynnistyksen voi tehdä mustitikulta, mutta tässä tapauksessa laitteet ovat sen verran vanhoja, että ne täytyy konfiguroida itse. Raspberry Pin käynnistysasetuksia voi muuttaa kahdella tapaa: näistä ensimmäinen on käynnistää Raspberry Pi OS SD-kortin kautta ja avata komennon *sudo raspi-config* avulla asetusvalikko, josta löytyy vaihtoehto Boot Over USB, jonka valitsemalla Raspberry Pi pyrkii jatkossa ensisijaisesti käynnistämään itsensä muistitikulta. Jos Raspberry Pi OS:n kuvan sisältävää SD-korttia ei valmiiksi löydy, on vaihtoehto kaksi vielä yksinkertaisempi. Raspberry Pin kotisivuilta voi ladata tietokoneelle ohjelman nimeltä Raspberry Pi Imager,

jonka avulla voi SD-kortille ladata ”misc utility images” eli toisin sanoen vain tietyn asetuksen, kuten juuri vaikka käynnistysjärjestyksen. Kun Raspberry Pin laittaa seuraavan kerran verkkovirtaan kyseisen SD-kortin kanssa, muuttaa se automaattisesti asetuksiaan kortin sisällön mukaisesti, eli seuraavalla käynnistyskerralla se hakee ensimmäisenä USB-tikkua.

Kun jokainen laite pystyy käynnistämään itsensä muistitikulta, ladataan USB-muistiin BalenaEtcherin avulla kuva Raspberry Pi -laitteille tarkoitetusta 30-bittisestä DietPi -käyttöjärjestelmästä. DietPin käynnistäessä ensimmäisen kerran se pyytää vaihtamaan käyttäjän salasanan avaa näkymän, jossa voi ladata erinäköisiä ohjelmistoja jos haluaa. DietPin oletusohjelmistot toimivat hyvin ja ylimääräisiä ei tarvitse vielä ladata. Tämän jälkeen DietPi päivittää itse itsensä jos päivityksiä on olemassa, ja avaa sen jälkeen oletusnäkymän eli terminaalin. Ensimmäisellä käynnistyskerralla DietPi vaatii yhteyden internetiin käynnistyäkseen oikein, eli Raspberry Pin kannattaa jo ennen käynnistystä olla Ethernetin kautta kiinnitettynä.

Terminaalista avataan komennolla *sudo dietpi-config* laiteasetusnäkymä. Täältä avataan ensin Network Options: Adapters ja sen sisältä Ethernet, jossa todennäköisesti oletusasetus on DHCP. Tämä vaihdetaan Static IP:seen, joka määritetään itse (kuva 1).



Kuva 1. DietPi:n Static IP-näkymä.

IP-osoitteen vaihdon jälkeen kokeillaan, toimiiko internet-yhteys vielä, joko Dietpi-configin Network-asetusten kautta tai sitten komentoriviltä *ping*-komennon avulla. DietPi-configin Security Optionsista vaihdetaan vielä lisäksi selvyyden takia laitteen nimi eli hostname. Tämän jälkeen ohjelma pyytää käynnistämään laitteen uudelleen. Nämä tehdään jokaiselle laitteelle, jotta ne ovat helppo löytää ja erottaa toisistaan.

Koska laitteita on paljon, kannattaa työskennellä ssh-yhteyden kautta, sillä se on huomattavasti helpompaa ja nopeampaa kuin työskentely oman monitorin kautta. Eli kun nämä asetukset on saatu tehtyä, ei pitäisi olla tarvetta enää kytkeä Raspberry Pi -laitetta omaan monitoriinsa HDMI:n kautta. Mikäli ensimmäisen käynnistyskerran yhteydessä ei muuta oletusasetuksia, pitäisi DietPi -käyttöjärjestelmässä olla automaattisesti asennettuna Dropbear-niminen ssh-palvelin. Jotta komentokehotteen saa näkyviin omalla tietokoneella, pitää myös sillä olla ssh-palvelin asennettuna sekä yhteys samaan verkkoon. Kun nämä ovat hoidettuna, sen laitteen komentokehotteella jossa halutaan saada Raspberry Pi näkyviin, suoritetaan komento *ssh (pi-käyttäjä)@(ip-osoite)*. Jos kyseessä on ensimmäinen kerta kun yhdistää tähän osoitteeseen, tulee varoitus. Kun siitä jatkaa eteenpäin, kysytään käyttäjän salasanaa. Tämän jälkeen pitäisi näkymän olla Raspberry Pin komentokehoite.

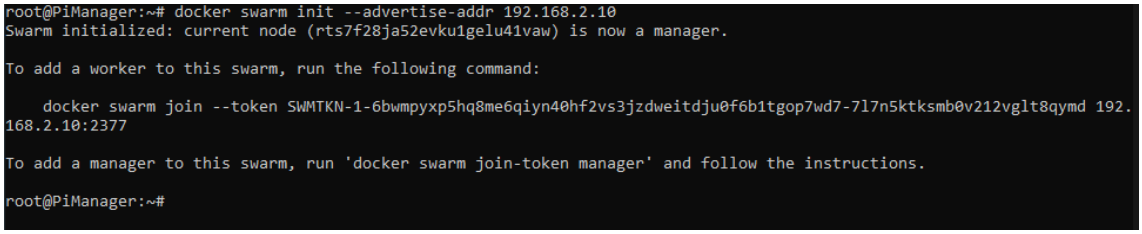
Kun kaikissa laitteissa on asennettuna DietPi ja niiden asetukset on määritelty, testataan pystyvätkö ne kommunikoimaan keskenään lähettämällä *ping*-komento jokaisen laitteen IP-osoitteeseen. Mikäli laitteet löytävät toisensa, voi jatkaa eteenpäin, mutta jos Raspberry Pit eivät pysty kommunikoimaan keskenään, kannattaa käydä läpi vielä EdgeRouterin asetukset, koska vika on silloin todennäköisimmin palomuurin määrityksissä.

Dockerin asennus

Docker tukee virallisesti Raspbiania eli nykyistä Raspberry Pi OS:ää ja ARM-arkkitehtuuria. DietPi käyttää pohjanaan Raspbianin lite-versiota, eli Dockerilla ei pitäisi tulla mitään yhteensopivuusongelmia. DietPi ehdottaa dietpi-software-näkymässä erilaisia ohjelmistoja, jotka pitäisi pystyä sen kautta helposti lataamaan. Docker löytyy tästä listasta, mutta syystä tai toisesta ohjelmistojen lataaminen tämän näkymän kautta ei onnistu. Dockerin voi kuitenkin helposti ladata myös terminaalin kautta komennoilla *curl -fsSL https://get.docker.com -o get-docker.sh* ja *sh get-docker.sh*. Jos haluaa, että Dockeria pystyy käyttämään muullakin kuin root-käyttäjällä tai *sudo*-komennolla, voi erikseen lisätä tässä vaiheessa vielä käyttäjän komennolla *sudo usermod -aG docker (käyttäjänimi)*. Auktorisoiduksi käyttäjäksi lisätään tällä käyttäjänimi dietpi.

Dockerin swarm-tila käynnistetään komennolla *docker swarm init*, ja siitä koneesta tulee automaattisesti manager eli työnjohtaja. Tämä komento antaa vahvistuksen siitä,

että swarm-tila on käynnistetty sekä lisäksi ohjeet siihen, kuinka lisätä worker eli työntekijä klusteriin (kuva 2).



Kuva 2. Docker Swarmin luominen.

Ohjeessa annettu komento kirjoitetaan sen Raspberry Pin komentokehotteelle, jonka halutaan liittyvän worker- eli työntekijäsolmuna klusteriin. Sama toistuu kaikilla lisättävillä worker-solmuilla. Worker-solmun lisäämiseen saa tokenin myös komennolla *docker swarm join-token worker*, mikäli swarm-tilan luomisen yhteydessä annettu token on unohtunut. Jos haluaa liittää klusteriin toisen manager-solmun, saa siihen tokenin samaan tapaan, komennolla *docker swarm join-token manager*. Swarmin tiedot voi tarkistaa komennolla *docker info*, jolloin se näyttää muun muassa kuinka monta manageria löytyy ja kuinka monta solmua yhteensä klusterissa on. Tässä tapauksessa managereita on yksi ja solmuja kolme – vaikka yhden solmun tehtävänimike onkin manager, ovat ne Dockerissa myös automaattisesti työntekijöitä (taulukko 1).

Taulukko 1. Klusterin laitteet.

Laitteen nimi	IP-osoite	Klusterirooli
PiManager	192.168.2.10	Manager
PiWorker1	192.168.2.11	Worker
PiWorker2	192.168.2.12	Worker

WWW-palvelimen luominen

Nyt Raspberry Pit ovat swarm-tilassa, mutta mitään tehtävää ei niillä vielä ole. Dockerin avulla voi helposti luoda erilaisia palveluja (engl. services). Yksinkertaisimmillaan ei palvelua luodessa tarvitse komennossa määrittää kuin se, millaisen palvelun haluaa luoda. Jos ei anna edes palvelulle nimeä, antaa Docker sille itse jonkin satunnaisen nimen. Koska tarkoituksena on luoda palvelu, joka toimisi WWW-palvelimena, on kuitenkin hyvä määritellä nimen lisäksi komennossa myös portit, joiden kautta palvelu toimii – isäntälaitteen portti sekä containerin portti (kuva 3).

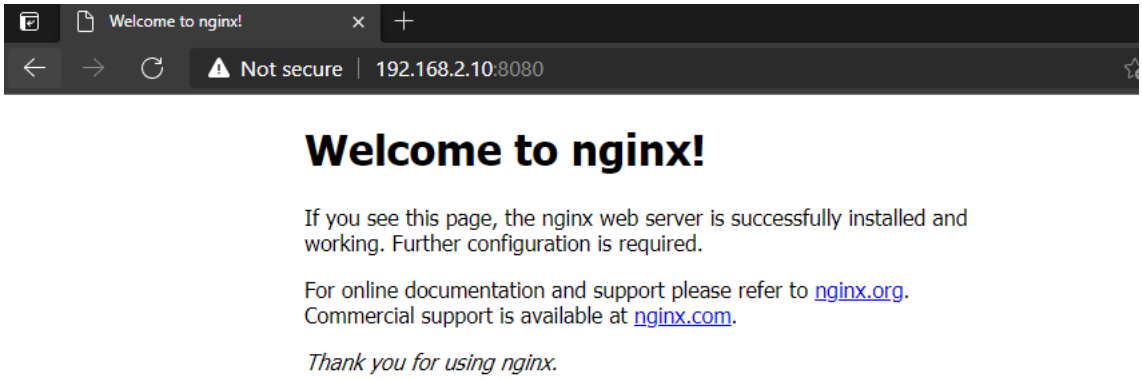
```
root@PiManager:~# docker service create --name testiserveri --publish 8080:80 nginx
yjc5eskqg2695pfouit305d8o
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@PiManager:~# docker service ps testiserveri
ID             NAME             IMAGE          NODE        DESIRED STATE   CURRENT STATE
zs2pljqttdkv   testiserveri.1   nginx:latest   PiManager   Running         Running about a minute ago
root@PiManager:~#
```

Kuva 3. Docker Servicen luominen.

Jos ei ole palvelua luodessa jo määrittänyt modea eli tilaa, on suoritettavana aluksi vain yksi tehtävä (engl. task), yhdellä laitteella. Tällöin siis klusterin koosta riippumatta vain yhdellä sen solmuista on tehtävä. Palvelun kokoa voi skaalata luomisen jälkeenkin komennolla *scale*, jolloin luodaan kopiot kuvasta (engl. image), jota pyöritetään. Eli tässä tapauksessa jos palvelua on skaalattu kokoon kolme, pitäisi jokaisella solmulla olla yksi kopio kuvasta. Yhdellä solmulla voi olla enemmänkin kuin yksi kopiota, maksimissaan kuitenkin kahdeksan. Jos yksi solmuista ei olekaan enää käytettävissä, pitäisi sen sisältämä tehtävä siirtyä jonkin toisen solmun suoritettavaksi. Palvelua luodessa voi kopioiden määrän jo määrittää lisäämällä komentoon *--replicas (kopioiden määrä).*

Jos palvelua luodessa määrittääkin tilan kuitenkin globaaliksi lisäämällä *--mode global,* on jokaisella klusterin solmulla tehtävä automaattisesti, ja uutta solmua lisättäessä klusteriin saa sekin heti tehtävän itselleen. Tässä tilassa ei kuitenkaan yhdellä solmulla voi olla yhtä tehtävää enempää, eli jos yksi solmulaite ei olekaan käytettävissä, on swarmissa aina yksi tehtävä vähemmän. Tässä toteutuksessa palvelu tehdään globaaliksi, muun muassa siksi, että palvelulle on määritelty portit, ja jos yksi solmu suorittaa useampaa tehtävää, ei niistä todennäköisesti toimisi kumpikaan, koska ne yrittäisivät molemmat käyttää samaa porttia.

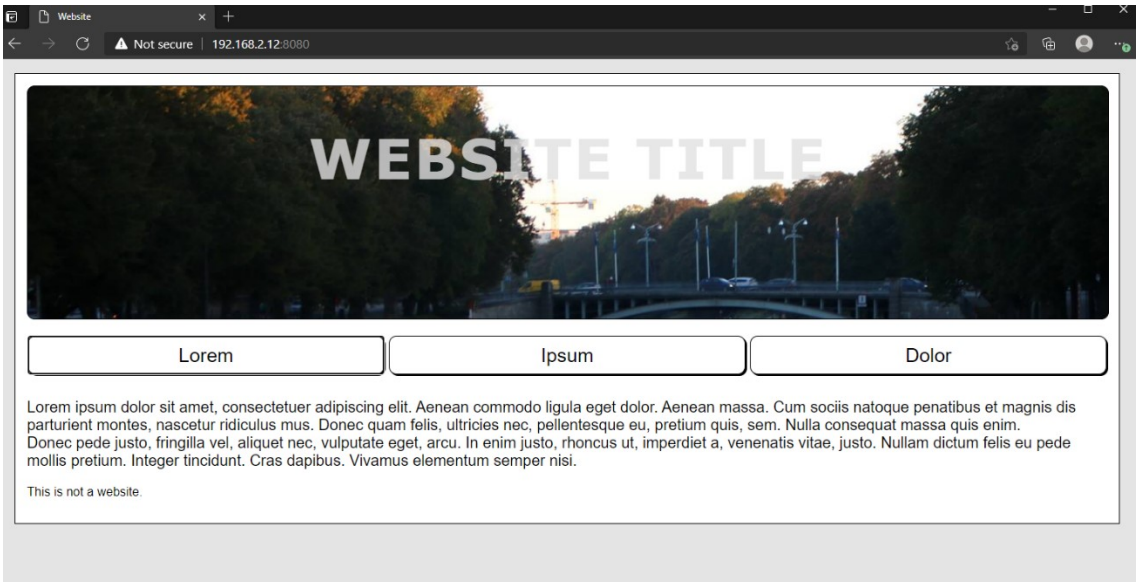
Jos samassa verkossa olevan tietokoneen kautta laittaa nyt palvelun luomisen ja määrittelyn jälkeen hakukenttään vaikka PiManagerin IP-osoitteen ja palvelua luotaessa määritetyn portin, pitäisi selaimen näyttää NGINX:in oletussivu mikäli WWW-palvelin toimii (kuva 4). Sivu tulee näkyviin, eli kaikki on tähän mennessä hyvin ja WWW-palvelin toimii kuten pitääkin. Dockerille ominaisesti palvelin toimii jos osoitteena käyttää minkä tahansa klusterin solmun IP-osoitetta. Varteen otettavaa on kuitenkin se, että vaikka osoitteena käyttäisikin tietyn solmun IP-osoitetta, voi palveleva tehtävä olla mikä tahansa vapaana olevista, eri solmuilla suoritettavista kopioista.



Kuva 4. NGINX-oletussivu.

WWW-palvelimen käyttöä ajatellen harva todennäköisesti kuitenkaan tyytyy suorittamaan vain NGINX:n oletussivua. Siksi olennaista on myös varmistaa, että haluamansa sivun saa myös näkymään. Tämän testaamista varten luodaan siis pieni staattinen nettisivu, jota Raspberry Pi -klusterin on tarkoitus suorittaa. Tämä testisivu koostuu yhdestä HTML-tiedostosta [liite 1] sekä yhdestä .jpg-tiedostosta. Tähän päivittämiseen on muutama erilainen toteutustapa: ensimmäinen on ladata halutut tiedostot laitteelle esimerkiksi Githubin kautta ja muokata konfiguraatiotiedoston polun viittaamaan haluttuun kansioon. DietPi -käyttöjärjestelmässä ei ole oletuksena ladattuna gitiä, eli se täytyy ladata komennolla *sudo apt-get install git*. Githubissa sijaitseva repository kopioidaan laitteelle *git clone* -komennolla. Tämä täytyy kuitenkin toistaa jokaiselle laitteelle erikseen, sillä muuten palvelin voisi ladata eri asioita – toisille päivitetyn sivun ja toisille NGINX-sivun. Koska tässä klusterissa solmuja on kolme, on

melko nopeaa ladata niille jokaiselle sivu kerrallaan, joten käytetään päivittämiseen ensimmäistä tapaa, mutta useamman solmun klusterissa kannattaa käyttää toista tapaa – omaa custom imagea private registryn kautta. Esimerkiksi Docker Hub on suosittu kuvien tallentamispaikka, josta voi hakea jo olemassa olevia kuvia tai tallentaa omansa, joko julkisena tai yksityisenä. Tällöin kuvan sisältöä on helppo ylläpitää, ja klusterissa olevan palvelun kuvan voi päivittää komennolla *docker service update --image (kuvan osoite) (palvelun nimi).* Tässä tapauksessa jokaiselle solmuista on päivitetty uusi sisältö, ja palvelimen osoite näyttää uutta sivua (kuva 5). WWW-palvelimen on siis todistettu toimivan halutusti.



Kuva 5. Testisivu.

4.2 Testaus

Yksi tärkeimmistä klusterin eduista on fail-over -mekaniikka, jonka pitäisi Dockerin kautta toimia swarmin luomisen jälkeen automaattisesti. Helpoimmin tämän näkee replica- eli kopiotilassa, koska silloin yksi solmu voi pyörittää useampaa kuin yhtä tehtävää. Jos palvelu on kopio-tilassa, jossa tehtäviä pyörii kolme ja jokainen niistä omalla laitteellaan, nähdään fail-over käytännössä palvelun tiedot näyttävän komennon *docker service ps (palvelun nimi)* avulla. Kun yhden Raspberry Pin irroittaa verkosta, näkyy tämä tilana shutdown. Solmulla olleen tehtävän voi huomata siirtyneen toiselle laitteelle suoritettavaksi (kuva 6). Globaalissa tilassa olevaa palvelinta voi myös testata irroittamalla yhden solmun verkkoyhteydestä, mutta silloin sama komento näyttää

palvelussa pyörivän vain yhden tehtävän vähemmän. Tämä kaikki kuitenkin koskee vain worker-solmuja. Jos manager-solmuja on vain yksi ja se kaatuu, kaatuu myös koko palvelin. Siksi jos haluaa palvelinta laajentaa, kannattaa lisätä myös toinen manager.

```
root@PiManager:~# docker service ps testiserveri
ID             NAME                IMAGE          NODE        DESIRED STATE   CURRENT STATE
p9cd1fr7wtsw   testiserveri.1      nginx:latest   PiManager   Running         Running 4 minutes ago
tgwli51775hi   testiserveri.2      nginx:latest   PiWorker1   Running         Running 4 minutes ago
xybirr4hvlew   testiserveri.3      nginx:latest   PiWorker1   Running         Preparing 8 seconds ago
uy1ldjthbr27    \_ testiserveri.3  nginx:latest   PiWorker2   Shutdown        Running 3 minutes ago
root@PiManager:~#
```

Kuva 6. Replica-palvelun tehtävät yhden laitteen irroittamisen jälkeen.

Palvelimen kantokyvyn testaamista varten voi joko kirjoittaa itse ohjelman, joka esimerkiksi suorittaisi samanaikaisesti tietyn määrän kutsuja palvelimelle ja mittaisi palvelimen vastausajat, mutta mikäli testeillä ei haluta mitata mitään erikoisempaa, on helpompaa ja aikaa säästävämpää käyttää jotain valmista ohjelmaa. Apachella on ohjelma nimeltä ApacheBench, mikä on Apachen HTTP-palvelimille tarkoitettu testausohjelmisto. ApacheBenchin testit pitäisi kuitenkin olla tarpeeksi yksinkertaisia, jotta niitä voisi soveltaa muihinkin kuin Apachen palvelimiin, kuten tässä työssä käytettyyn NGINX:iin. Koska testeillä halutaan tietää lähinnä ajat sekä sen, vastaanottaako palvelin kaikki pyynnöt, on ApacheBench riittävä näihin testauksiin.

ApacheBenchin käyttöönotto on erilainen prosessi riippuen siitä, mikä käyttöjärjestelmä on sillä tietokoneella, jolta käsin testaukset halutaan tehdä. Esimerkiksi Mac OS:llä Apache on valmiiksi asennettu, mutta tässä tapauksessa käyttöjärjestelmä on Windows 10 ja sille Apache täytyy ladata erikseen. ApacheBenchiä varten täytyy ladata ja asentaa Apache2.4 esimerkiksi sivulta ApacheLounge. Asennuksen jälkeen siitä kansiosta käsin, josta Apache löytyy, voi komennolla *ab* kutsua testejä. Näissä testeissä halutaan määritellä kaksi asiaa: kutsujen määrä (*-n*) ja kuinka monta kutsua samanaikaisesti lähetetään (*-c*), eli esimerkiksi komentona voisi toimia *ab -n 15000 -c 250 http://192.168.2.10:8080/,* jolloin ApacheBench lähettää annettuun osoitteeseen 15 000 kutsua, 205 kutsua kerrallaan ja kokoaa tiedot ylös.

Kutsujen määrän ei pitäisi vaikuttaa juurikaan lopputulokseen, toisin kuin samanaikaisten kutsujen määrä. Testataan varmuuden vuoksi kuitenkin myös eri kokonaiskutsujen määrillä. Tässä testissä hyväksyttävä tuloksen rajoiksi on asetettu se, että epäonnistuneiden kutsujen määrä on nolla ja keskimääräinen palvelimelle tehtävien

pyyntöjen aika on 200ms tai alle. Ladattava verkkosivu on tässä tapauksessa palvelimelle aiemmin ladattu testisivu ja sen koko on 3651 B.

Jokainen testi ajettiin kolmeen kertaan, ja niistä kehnoin tulos kirjattiin ylös (taulukko 2). 300 samanaikaista kutsua vaihteli alle sekä yli kahdensadan millisekunnin, mutta koska vain kehnoin tulos huomioitiin, kirjattiin näiden testi epäonnistuneeksi. Näiden testien lisäksi kokeiltiin vielä 200 samanaikaista kutsua suhteessa 30 000:een ja 40 000:een kokonaiskutsumäärään. Näissä keskimääräinen pyyntöjen aika pysyi edelleen alle kahdensadan millisekunnin, välillä 142–158ms.

Taulukko 2. Apache Bench -testien tulokset.

Kutsujen määrä	Samanaikaisten kutsujen määrä	Testitulos hyväksytty/hylätty	Aika per pyyntö, keskiarvo	Epäonnistuneiden pyyntöjen määrä
1000	50	Hyväksytty	42.419ms	0
1000	100	Hyväksytty	86.502ms	0
1000	200	Hyväksytty	157.044ms	0
1000	300	Hylätty	225.363ms	0
10000	50	Hyväksytty	35.533ms	0
10000	100	Hyväksytty	65.355ms	0
10000	200	Hyväksytty	156.790ms	0
10000	300	Hylätty	216.374ms	0

Suurin mahdollinen samanaikaisten kutsujen määrä näillä rajoituksilla on siis noin kaksisataa, todennäköisesti hieman yli, mikä on erinomainen tulos haluttuun tarkoitukseen. Suunnitelman mukaisesti palvelimen tulisi pahimmassa tapauksessa kestää 200 samanaikaista käyttäjää, ja näistä kahdesta sadasta käyttäjästä kaikki eivät pyyntöjä todennäköisesti samaan aikaan palvelimelle lähetä, jolloin todellinen maksimikäyttäjämäärä voi olla vielä suurempi.

Syy siihen, minkä takia testissä haettiin alle 200ms vastausaikaa on se, että nämä tehtiin samassa lokaalissa verkossa palvelimen kanssa olevan tietokoneen kautta – riippuen siitä, miten palvelinta halutaan tulevaisuudessa käyttää, voi olla että yhteys ja siten testitulokset ovat nopeampia tässä testitilanteessa. 200ms antaa ikään kuin pelivaraa

sitä varten, jos yhteys onkin hitaampi tai ladattava verkkosivu suurikokoisempi kuin nyt. Silloin 200 samanaikaista pyyntöä pysyvät todennäköisesti edelleen hyväksyttävän vastausnopeuden rajoissa.

4.3 Klusteripalvelimen kehittämisen jatkaminen

Tässä vaiheessa rakennettu WWW-palvelin on hyvä pohja, jota voi kehittää eteenpäin omiin tarpeisiin sopivaksi. Jos palvelin haluttaisiin oikeaan käyttöön tämän jälkeen, on joitain muutoksia joista kannattaa aloittaa, esimerkiksi USB-tikut voi vaihtaa tallennustilaltaan hieman suurempiin – kahdeksan gigatavua toimii tässä versiossa hyvin mutta oikeassa käytössä raja tulisi todennäköisesti nopeasti vastaan. Jos haluaa käyttää palvelinta muutenkin kuin lokaalissa ympäristössä, kannattaa tulevaisuudessa ylimääräiseksi load balancing -tasoksi ja käänteiseksi (engl. reverse) proxyksi lisätä esimerkiksi vielä HAProxy, jolloin kaikki liikenne välitetään sen kautta varsinaiselle palvelimelle. Lisäksi yleistä käyttöä varten palvelimen voi tehdä turvallisemmaksi lisäämällä käyttöön TLS-sertifikaatin. WWW-palvelimena toimiva NGINX on hyvä käsittelemään staattisia nettisivuja, mutta jos tulevaisuudessa haluaa käyttöön dynaamisen nettisivun, voi sen vaihtaa johonkin toiseen.

Jos palvelinta haluaa vielä laajentaa tulevaisuudessa, kannattaa ympäristöön myös panostaa. Ylikuumenemista helpottamaan voi Raspberry Pi -laitteisiin asentaa esimerkiksi jäähdytyselementin, ja johtojen valtavan määrän takia kannattaa harkita myös PoE:tä eli virtaa Ethernetin kautta tukeviin kytkimiin ja Raspberry Pi -lisävarusteisiin sijoittamista.

5 LOPUKSI

Tämän opinnäytetyön tarkoituksena oli toteuttaa edullinen WWW-palvelin ja tutkia, riittääkö muutaman Raspberry Pin klusterista tarpeeksi tehoa siihen. Kolmesta laitteesta muodostuvaan klusteriin käytettiin niitä yhdistävänä ohjelmistona Dockeria ja HTTP-palvelimena NGINX:ää.

Lopputulos on toimiva WWW-palvelin, joka tosin oikeaa käyttöönottoa varten vaatii vielä joitain muokkauksia riippuen siitä mitä, palvelimelta halutaan. Tässä työssä palvelinta kehitettiin pienessä lokaalissa verkossa, jossa siihen sai yhteyden vain samassa verkossa kytkettynä oleva laite. Tämä vaikutti toteutuksessa kahteen asiaan: palvelimen turvallisuuteen ja testaukseen. Salauksiin ja muihin varotoimenpiteihin ei tässä työssä juurikaan keskitytty, mutta jos palvelin halutaan avata yleiseen käyttöön, on suositeltavaa implementoida joitain salausmekanismeja. Testauksen tulokset taas eivät välttämättä päde mikäli palvelinta ei käytettäisikään vain lokaalisti. Testitulosten perusteella palvelimen voi kuitenkin todeta olevan siinäkin tapauksessa todennäköisesti riittävän hyvä opiskelijakäyttöä varten, eli kolmen Raspberry Pin klusteri täyttää ehdot sekä edullisuudesta että toimivuudesta.

LÄHTEET

[1] HANNU JAAKOHUHTA, MOT IT-Ensyklopedia.

[2] YANG, J., JIN, D., LI, Y., HIELSCHER, K. and GERMAN, R., 2006. Modeling and simulation of performance analysis for cluster-based web server. Simulation Modelling Practice and Theory, 14(2), pp. 188-200.

[3] PARK, J., KIM, J., AHN, C., WOO, Y. and CHOI, H., 2008. Cluster Management in a Virtualized Server Environmet, 2008 10th International Conference on Advanced Communication Technology, Feb 17-20, 2008 2008, IEEE.

[4] LI, B., SHANG, J., DONG, M. and HE, Y., 2020. Research and Application of Server Cluster Load Balancing Technology, 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Jun 12-14, 2020 2020, IEEE.

[5] JIGUANG, W., XIAOMING, D. and LING, Z., 2005. Research and design of a self-managing storage cluster, 2005 IEEE International Conference on Granular Computing, 25-27.7.2005 2005, IEEE.

[6] DENNIS, A.K., 2013. Raspberry Pi Super Cluster. UK: Packt Publishing, Limited.

[7] EVANS, P.J., Nov 13, 2019-last update, Build a Raspberry Pi cluster computer. Saatavilla: https://magpi.raspberrypi.org/articles/build-a-raspberry-pi-cluster-computer [Luettu 17.3.2021].

[8] JOHNSTON, S.J., BASFORD, P.J., PERKINS, C.S., HERRY, H., FUNG, P.T., PEZAROS, D., MULLINS, R.D., YONEKI, E., COX, S.J. and SINGER, J., 2018. Commodity single board computer clusters and their applications. Future Generation Computer Systems, 89, pp. 201-212.

[9] RASPBERRY PI FOUNDATION, January, 2021-last update, Raspberry Pi 4 product brief . Saatavilla: https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf [Luettu 16.3.2021].

[10] CAMPBELL, A., Apr 3, 2017-last update, How to have a Linux home server on the cheap. Saatavilla: https://www.pcworld.com/article/3184925/how-to-have-a-linux-home-server-on-the-cheap.html [Luettu 20.3.2021].

[11] Sähkön hintatilastot. Saatavilla: https://energiavirasto.fi/sahkon-hintatilastot [Luettu 17.3.2021].

[12] Peliserverit. Saatavilla: https://net9.fi/ [Luettu 20.3.2021].

[13] AWS Free Tier. Saatavilla: https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc [Luettu 20.3.2021].

[14] Azure Free Account. Saatavilla: https://azure.microsoft.com/en-gb/free/ [Luettu 20.3.2021].

[15] User Manual - 1.2 Some Terminology. Saatavilla: https://www.virtualbox.org/manual/UserManual.html#virt-why-useful [Luettu 20.3.2021].

[16] RASPBERRY PI FOUNDATION, , Raspberry Pi 4. Saatavilla: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ [Luettu 17.3.2021].

[17] RASPBERRY PI FOUNDATION, , Raspberry Pi OS. Saatavilla: https://www.raspberrypi.org/documentation/raspbian/ [Luettu 17.3.2021].

[18] DietPi. Saatavilla: https://dietpi.com/ [Luettu 18.3.2021].

[19] What's new in Failover Clustering, 18.11.2018-last update. Saatavilla: https://docs.microsoft.com/en-us/windows-server/failover-clustering/whats-new-in-failover-clustering [Luettu 20.3.2021].

[20] Swarm Mode key concepts. Saatavilla: https://docs.docker.com/engine/swarm/key-concepts/ [Luettu 19.3.2021].

[21] What is Kubernetes. Saatavilla: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ [Luettu 19.3.2021].

[22] NGINX Wiki. Saatavilla: https://www.nginx.com/resources/wiki/ [Luettu 19.3.2021].

[23] KOMULAINEN, M., Jan 20, 2021-last update, Turun AMK vie ICT-insinöörikoulutusta Keski-Aasiaan. Saatavilla: https://www.turkuamk.fi/fi/ajankohtaista/2663/turun-amk-vie-ict-insinoorikoulutusta-keski-aasiaan/ [Luettu 18.3.2021].

[24] Ulkomaalaiset IT-opiskelijat helpottamaan Varsinais-Suomen osaajapulaa Jan 11, 2019-last update. Saatavilla: https://www.turkuamk.fi/fi/ajankohtaista/2048/ulkomaalaiset-it-opiskelijat-helpottamaan-varsinais-suomen-osaajapulaa/ [Luettu 18.3.2021]

Liite 1 – index.html

```
<!DOCTYPE html>
<html>
<head>
          <title>Website</title>
</head>
<style>
div.head{
          background-image: url('turku.jpg');
          background-repeat: no-repeat;
          background-size: cover;
          background-attachment: fixed;
          height: 300px;
          width: 100%;
          border: 1px solid black;
          border-radius: 10px;
}
h1{
          font-family: verdana, sans-serif;
          font-size: 450%;
          color: rgba(225,225,225,0.8);
          text-align: center;
          text-transform: uppercase;
          letter-spacing: 3px;

}
p{
          font-family: arial;
          font-size: 125%;
          color: black;
          display: block;
}
p.nonvisible{
          display: none;
}
p.footnote{
          font-size:100%;
}
div.buttons{
          padding:20px 0px 10px 0px;
}
div.content{
          padding: 15px;
          background-color: white;
          border: 1px solid black;
}
button{
          height:50px;
    width: 33%;
    text-align: center;
    text-decoration:none;
    font-family: arial;
    font-size: 25px;
```

```
    color: black;
    border: 1px solid ;
          border-radius: 10px;
    background-color: white;
    margin: 1px;
          box-shadow: 2px 2px;

}
button:hover, button:active{
          background-color: rgb(228,228,228);
}
body{
          padding: 10px 20px 10px 20px;
          background-color: rgb(228,228,228);
}
</style>
<script>
function loremClick(){
          document.getElementById("lorem").style.display = "block";
          document.getElementById("ipsum").style.display = "none";
          document.getElementById("dolor").style.display = "none";
};

function ipsumClick(){
          document.getElementById("lorem").style.display = "none";
          document.getElementById("ipsum").style.display = "block";
          document.getElementById("dolor").style.display = "none";
};

function dolorClick(){
          document.getElementById("lorem").style.display = "none";
          document.getElementById("ipsum").style.display = "none";
          document.getElementById("dolor").style.display = "block";
};
</script>
<body>
<div class="content">
          <div class="head">
          <h1>Website Title</h1>
          </div>
          <div class="buttons">
                    <button id="btn_lorem"
onclick="loremClick()">Lorem</button>
                    <button id="btn_ipsum"
onclick="ipsumClick()">Ipsum</button>
                    <button id="btn_dolor"
onclick="dolorClick()">Dolor</button>
          </div>
          <p id="lorem">Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.
Cum sociis natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus. Donec quam felis, ultricies nec,
pellentesque eu, pretium quis, sem. Nulla consequat massa quis
enim. <br>
           Donec pede justo, fringilla vel, aliquet nec, vulputate
eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis
```

```
vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer
tincidunt. Cras dapibus. Vivamus elementum semper nisi.</p>
          <p id="ipsum" class="nonvisible">Aenean vulputate
eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae,
eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis,
feugiat a, tellus. Phasellus viverra nulla ut metus varius
laoreet.<br>
          Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi
vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui.
Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus,
sem quam semper libero, sit amet adipiscing sem neque sed
ipsum.</p>
          <p id="dolor" class="nonvisible">Nam quam nunc, blandit
vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et
ante tincidunt tempus. Donec vitae sapien ut libero venenatis
faucibus.<br>
          Nullam quis ante. Etiam sit amet orci eget eros faucibus
tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec
sodales sagittis magna.</p>
          <p class="footnote">This is not a website.</p>
</div>
</body>
</html>
```