Palvelinympäristön kehittäminen hyperkonvergenssiin

Timo Lotta

2021 Laurea

Laurea-ammattikorkeakoulu

Palvelinympäristön kehittäminen hyperkonvergenssiin

Timo Lotta

Tietojenkäsittely

Opinnäytetyö

2,2021

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Timo Lotta

Palvelinympäristön kehittäminen hyperkonvergenssiin

Vuosi 2021 Sivumäärä 34

Opinnäytetyön toimeksiantajana toimi SafeIT Oy. Opinnäytetyön tavoitteena oli verrata, määritellä ja suunnitella eri vaihtoehtoja, joilla nykyiset järjestelmät voi korvata. Opinnäytetyön tarkoituksena oli löytää nykyistä järjestelmää halvempi, vikasietoisempi ja helpommin laajennettavissa oleva järjestelmä. Työhön valikoitui muutama eri ohjelmisto, jolla voi toteuttaa hyperkonvergoidun järjestelmän. Hyperkonvergoitu järjestelmä on ohjelmistopohjaisesti toteutettu hajautettu palvelinympäristö.

Opinnäytetyön aineistoksi valikoitui palvelinympäristöjen kehitys ja virtualisointi, tallennusjärjestelmät ja eri tavat toteuttaa tallennus sekä projekti- ja kehittämistyö. Tutkimus- ja kehittämismenetelminä työn tekemiseen käytettiin havainnointia, haastattelua ja hiljaista tietoa. Kehittämisprojektin vaiheistuksessa käytettiin vesiputousmallia iteratiivisesti. Työssä valittiin ohjelmisto testiympäristön toteuttamiseksi vertailun, määrittelyn ja suunnittelun tuloksena. Työssä löydettiin yrityksen tarpeisiin sopiva ohjelmisto palvelinympäristön toteuttamiseen testiympäristöstä saatujen tulosten osalta. Nykyjärjestelmän kehitysehdotuksena oli siirtyminen hybriditallennusjärjestelmään.

Asiasanat: Hyperkonvergenssi, Kehittäminen, Palvelinympäristö

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's Degree

Timo Lotta

Developing a Server Environment for Hyperconvergence

Year 2021 Pages 34

The Bachelor's thesis was commissioned by SafeIT Ltd. The aim of this study was to compare, define and design different alternatives to replace the existing systems. The purpose of this thesis was to find out a cheaper, more fault-tolerant and more easily expandable system than the current system. For the Thesis project, a couple of software options were selected to be able to implement a hyperconverged system. A hyperconverged system is a software-based, distributed server environment.

The development and virtualization of server environments, storage systems, various ways of implementing storage and project and development work were selected as material for the thesis. Observations, interviews, and tacit knowledge were used as research and development methods of the work. The Waterfall model was iteratively used in the phasing of the development project. The software to implement the test environment was chosen by means of comparison, definition and design. As a result of the work a suitable software was found for the company for implementation of the server environment in terms of the results obtained from the test environment. The development proposal for the current system was a transition to a hybrid storage system.

Keywords: Hyperconvergence, Developing, a Server Environment

Sisällys

1 Johdanto

Tämän toiminnallisen opinnäytetyön tarkoituksena on kehittää virtuaalinen palvelinympäristö, jolla voidaan korvata nykyisin käytössä oleva palvelinympäristö. Työn teoreettinen viitekehys rakentuu projektityön ja ohjelmistotuotannon keskeisimmistä vaiheistuksista sekä käsittelee työlle tärkeässä osassa olevaa virtualisointia ja palvelinkeskus-infrastruktuurin kehittämistä hyperkonvergenssi-ohjelmiston avulla.

Teoreettisessa viitekehyksessä perehdytään työlle keskeisessä osassa olevan tallennusjärjestelmän toteutukseen eri vaihtoehtojen kautta siten, että työssä esiin tulleita havaintoja hyödynnetään toteutusvaiheessa. Toteutus-osiossa tehdään esitutkimus, määrittely, suunnittelu, toteutus, integrointi ja testaus. Työstä rajautuu pois käyttöönottovaihe, koska päätöstä järjestelmän siirtämisestä tuotantoympäristöön ei ole vielä tehty.

Toteutusvaiheessa kuvaan koko projektia ja sen edistymistä. Tavoitteena on löytää ohjelmisto, joka otetaan käyttöön. Opinnäytetyössä tulen käyttämään tutkimusmenetelminä havainnointia, haastatteluja sekä hyödynnän yrityksestä löytyvää hiljaista tietoa, jota on kertynyt työntekijöille kokemuksien kautta. Opinnäytetyö tehtiin vuoden 2020 aikana.

2 Työn lähtökohdat

Opinnäytetyön toimeksiantajana toimii SafeIT Oy, joka on perustettu 2002. Yritys toteuttaa teknisiä turvallisuusjärjestelmiä asiakkaiden toiveiden mukaisesti. Opinnäytetyön tavoitteena on löytää vaihtoehtoinen tapa korvata olemassa oleva palvelin- ja tallennusjärjestelmäinfrastruktuuri HCI-ohjelmiston avulla. Työn tekijällä ei ole aikaisempaa kokemusta virtuaaliympäristöistä. Nykyinen järjestelmä lyhyesti kuvattuna on seuraavanlainen: VMware esxi vsphare -ympäristö, jossa tallennusjärjestelmä on toteutettu open-e-dss -ohjelmistolla. Tämä tarkoittaa sitä, että nykyinen järjestelmä on CI:n mukainen, jossa komponentit ovat erillisiä. Nykyisin käytössä on kaksi erillistä RAID-ohjaimella toimivaa tallennuslaitetta, kaksi palvelinta ja kaksi kytkintä.

2.1 Aiheen valinta ja rajaus

Kehittämistyön idea lähti yrityksen tarpeesta kehittää olemassa olevaa tallennusjärjestelmää tai vaihtaa koko palvelinympäristö uuteen. Tämä työ on tarpeellinen, koska nykyisessä järjestelmässä on useita erilaisia puutteita, joita on havaittu vuosien varrella.

Tallennusjärjestelmä on nykyisellään monimutkainen, kallis ylläpitää eikä se ole helposti laajennettavissa. Olisi tarpeellista löytää parempia vaihtoehtoja toteuttaa vastaavanlainen ympäristö kuin mikä yrityksellä on tällä hetkellä käytössä. Yrityksellä on käytössä useita erillään olevia palvelinympäristöjä, joita on toteutettu monilla eri tavoilla. Vaihtoehtoina pidetään siirtymistä hyperkonvergoituihin ratkaisuihin, koska ne mahdollistavat selkeämpään infrastruktuuriin siirtymisen.

Tässä työssä tarkoituksena on vertailla, määritellä ja suunnitella eri vaihtoehtoja, joilla nykyiset järjestelmät voitaisiin korvata. Vertailussa otetaan huomioon laitteistovaatimukset, koska ne ovat merkittävässä osassa laitteistojen hintaa ajatellen.

Aiheesta on rajattu pois kaikki muut HCI-ympäristön sisällään pitävät mahdollisuudet, jotka eivät liity virtualisointiin tai tallennusjärjestelmän luomiseen. Aiheesta on myös rajattu pois ohjelmistojen asennus, konfiguraatio, käyttöönotto ja testaus, koska niitä ei ollut mahdollista suorittaa opinnäytetyön osana.

2.2 Opinnäytetyön tarkoitus ja tavoitteet

Tutkimuskysymykset on määritelty toimeksiantajan tavoitteiden mukaisesti:
- Vertailla, määritellä ja suunnitella eri vaihtoehtoja, joilla nykyiset järjestelmät voitaisiin korvata.
- Onnistutaanko esitetyllä ratkaisulla säästämään yrityksen resursseja?
- Saavutetaanko vaadittava vikasietoisuus?
- Onko järjestelmä helpommin laajennettavissa kuin nykyinen järjestelmä?

Kehittämistyön tavoitteena on valita HCI-ohjelmisto, jonka avulla voidaan toteuttaa laboratorioympäristö, jossa voidaan järjestelmän vikasietoisuutta testata.

Tärkeimpänä asiana on löytää ohjelmisto vertailun tuloksena, jolla olisi mahdollista korvata olemassa olevia järjestelmiä ja keskittyä ohjelmiston kykyyn toteuttaa helposti skaalautuva vikasietoinen tallennusjärjestelmä, jonka avulla voitaisiin säästää ylläpidon elinkaarikustannuksissa. Tämän saavuttamiseksi on tarpeellista selvittää mitä tallennusjärjestelmät ja palvelinympäristöt pitävät sisällään olemassa olevien kirjallisuuden ja verkkolähteiden avulla. Aiheen perusteellinen ymmärtäminen on tarpeellista onnistuneen lopputuloksen saavuttamiseksi.

2.3 Käsitteet

DSF: Distributed File System eli hajautettu tiedostojärjestelmä

VMware: Virtualisointiohjelmisto

Esxi: virtuaalikoneiden suorittamiseen käytettävä ohjelmistotaso

vSphare: palvelimen virtualisointiohjelmisto

open-e-dss: tallennusjärjestelmän ohjelmisto

HA: High Availability tarkoittaa korkeaa saatavuutta

HCI: Hyperconverged Infrastructure on ohjelmistolla määriteltävissä oleva palvelinarkkitehtuuri, jossa palvelinkeskuksen ominaisuudet sijaitsevat yhdessä laitteessa

HDD: Hard disk drive on tietokoneen massamuisti, jonka tallennus tapahtuu mekaanisesti

SSD: Solid-state drive on tietokoneen massamuisti, jonka tallennus toimii ilman mekaanisia osia.

I/O: Tässä työssä I/O tarkoittaa input/output operaatioita, joita käytetään mittaamaan tallennuslaitteen suorituskykyä.

Klusteri: Ryhmä yhteen liitettyjä laitteita

WAN: Wide area network eli laajaverkko

SAN: Storage area network käytetään yhdistämään NAS-palvelimia

NAS: Network attached storage eli verkkotallennus

CI: Converged Infrastructure eli yhdistetty infrastruktuuri. Yhdistetyssä infrastruktuurissa palvelimet, tallennus-, verkko- ja hallintalaitteet koostuvat erillisistä komponenteista.

Pariteetti: Käytetään tietotekniikassa virheiden tarkastuksessa. Pariteetti on parillinen tai pariton riippuen siirrettävien ykkösiä sisältävien bittien määrästä.

iSCSI: Internet iSCSI eli verkon välityksellä toimiva SCSI

SCSI: Small Computer System Interface on standardi tiedon siirtämiseksi tietokoneen ja siihen liitettyjen laitteiden välillä

SMB: Tiedostonsiirtoprotokolla

3 Projektin vaiheistus vesiputousmenetelmällä

Projektien ja projektitöiden tarkoituksena on johtaa toimintaa nykytilasta tavoitetilaan siten, että yrityksen tulos paranee samalla. Onnistuneissa projekteissa tavoitteet ja perustehtävät ovat selkeät koko projektin ajan. Syitä projektin aloittamiselle voi olla useita, mutta projekti

yleensä aloitetaan jostain esille tulleesta tarpeesta. Tällainen tarve voi olla esimerkiksi toimintaa tukevan infrastruktuurin kehittäminen. Tuotekehitysprojektien tavoitteena voi olla olemassa olevan tuotteen parantaminen tai uuden tuotteen luominen. Tuotekehitysprojekti tulee useimmiten yrityksen omista tarpeista mikä johtaa siihen, että projektin aikataulutus, ajankohta ja siitä saatavat rahalliset hyödyt ovat hankalasti arvioitavissa. (Mäntyneva 2016, luku 1.)

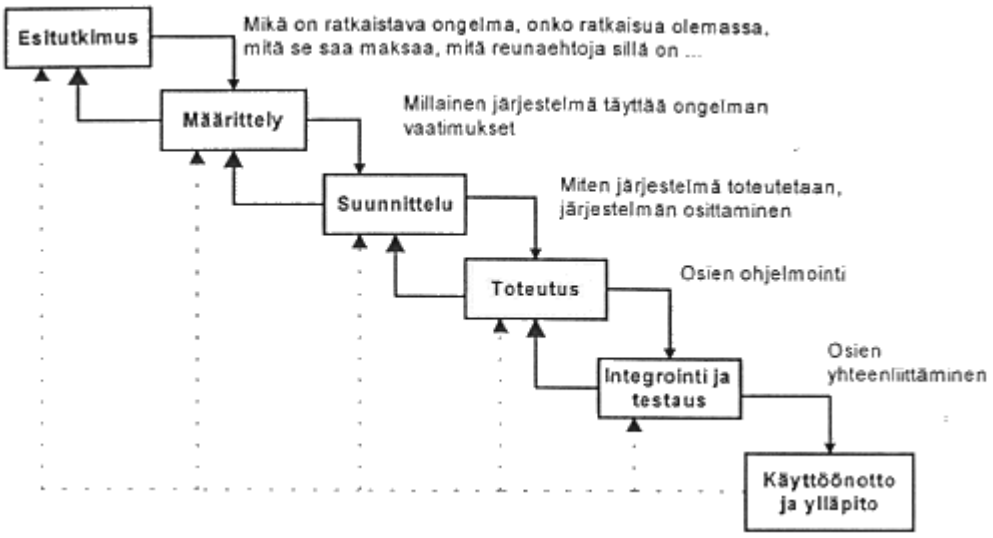
Projekteilla on aina aloitus ja lopetus, kuitenkin projektia voidaan valmistella vuosia ennen sen varsinaista aloittamista. Valmisteluvaiheeseen sisältyy projektin aloittamista ennen tehdyt työt. Hyvän valmistelutyön tuloksena on helpompi suunnitella projektia. (Mäntyneva 2016, luku 2.)

Suunnittelussa määritellään projektin laajuus. Tavoitteiden saavuttamiseksi tarvitaan esimerkiksi resursseja, henkilöitä ja muita toimia. Suunnittelussa määritellään projektille aikataulu, kustannukset ja resurssit tarkasti, jotta projektisuunnitelma voidaan toteuttaa. Suunnittelussa on tarpeellista havainnoida mahdolliset riskit sekä tunnistaa ongelmakohdat, jotta ongelmakohtien ja riskien varalle voidaan tehdä suunnitelma. (Mäntyneva 2016, luku 2.)

Toteutuksessa projektin etenemistä ja resurssien käyttöä valvotaan ja seurataan. Valvonnan ja seurannan avulla pyritään löytämään projektin etenemistä hidastavat ongelmat. Ongelmat ovat tärkeä havainnoida, että niihin voidaan puuttua ajoissa. (Mäntyneva 2016, luku 2.)

Projektin päättäminen tulee siinä vaiheessa ajankohtaiseksi, kun lopputuote on saatu valmiiksi ja projektipäällikkö on tehnyt loppuraportin. Projektityön loputtua työn tulokset dokumentoidaan ja projektin onnistuminen arvioidaan. Loppuraportissa on tiivis yhteenveto työn toteutuksesta ja maininnat siitä, miten projekti poikkeaa projektisuunnitelmasta. Loppuraportista ja dokumenteista voidaan saada tärkeitä tietoja, joita voidaan hyödyntää tulevissa projekteissa. Lopussa, projektin ohjausryhmä tarkastaa tuloksen ja että projekti vastaa haluttua kokonaisuutta. (Mäntyneva 2016, luku 2.)

Ohjelmistotuotannolla tarkoitetaan tapaa toteuttaa ohjelmisto, sen kehittäminen, ylläpito ja toiminta järjestelmällisesti. Ohjelmistotuotannon osa-alueet ovat vaatimusten määrittely, suunnittelu, toteutus, testaus ja ylläpito. (Haikala & Märijärvi, 2004, 35.) Kuviossa 1 esitellään vesiputousmalli, joka on lineaarinen yksinkertainen prosessimalli kuvaamaan työvaiheita. Lineaarisesti etenevä prosessi etenee vaihe kerrallaan, ja kun vaihe on saatu valmiiksi, siirrytään seuraavaan vaiheeseen. Lineaarista prosessia toteutetaan monella tuotannon alalla. Vesiputousmallia voidaan käyttää myös siten, että tarvittaessa siirrytään askelissa taaksepäin.



Kuvio 1: Vesiputousmalli (Haikala & Märijärvi, 2002)

Esitutkimuksessa tarkoituksena on kartoittaa järjestelmän vaatimukset ja toteuttamismahdollisuudet siten, että voidaan vastata kysymyksiin mitä järjestelmän pitää tehdä, onko järjestelmä mahdollista tehdä ja onko järjestelmää kannattavaa tehdä. Koko projektin kannalta tämä on tärkeä vaihe, koska esitutkimuksessa päätetään myös se, onko projekti tarpeellista aloittaa. (Haikala & Märijärvi, 2004, 37.)

Määrittelyvaiheessa kootaan ja analysoidaan asiakkaan tarpeita. Apuna tarpeiden kartoittamisessa voidaan käyttää esimerkiksi kyselylomakkeita sekä haastatteluja. Märittelyn tuotoksena pitäisi tulla selvyys siitä mitä järjestelmä tekee. Määrittelyvaiheessa voidaan selvittää onko projekti toteutettavissa ja asettaa projektille tavoitteet ja vaatimukset. Vaatimusmäärittelyssä voidaan esimerkiksi määrittää ne vaatimukset, miten järjestelmän tulee toimia vikatilanteissa. Muita osa-alueita voi olla järjestelmän toiminnallisuus eli miten järjestelmä on integroitavissa muihin järjestelmiin tai käytettävyyteen liittyvät asiat. (Haikala & Märijärvi, 2004, 78-80.)

Suunnittelussa esitellään millä tavalla järjestelmä toteutetaan. Suunnittelun tulos esittelee ratkaisun, miten järjestelmä pystyy suorittamaan tehtävät, järjestelmän toiminnot ja määritykset ilmaistaan teknisellä kielellä. Suunnittelun osa-alueita ovat esimerkiksi ohjelmiston, testauksen ja käyttöönoton suunnittelu. Moduuleiksi jaetun järjestelmän ideana on saada järjestelmä toimimaan siten, että moduulit olisivat mahdollisimman vähän riippuvaisia toisista moduuleista. Vähäinen riippuvuus mahdollistaa moduulin korvaamisen tai vaihtamisen ilman, että sillä on suurta kokonaisvaikutusta toisiin moduuleihin nähden. (Haikala & Märijärvi, 2004, 81-83.) Toteutuksessa tehdään suunnittelun ja määrittelyn mukainen ohjelmisto.

Testaus koostuu suunnittelusta, testiympäristön tekemisestä, testaamisesta ja tulosten tarkastelusta. Testauksessa etsitään suunnitelmallisesti virheitä ohjelmasta, mutta testaus voi pitää myös sisällään dokumentaation tarkastuksen. Testaus vähentää ylläpidon tarvetta, mutta se ei poissulje virheiden olemassaoloa, koska perusteellinen testaus on käytännössä mahdotonta. (Haikala & Märijärvi, 2004, 283-285.)

4 Datakeskusten infrastruktuurin kehittyminen

Datakeskukset ovat kehittyneet paljon viime vuosina. Kaikki alkoi kuitenkin keskustietokoneesta, jonka avulla voitiin hallita laitteistoresursseja sisäisellä redundanssilla. Tällä tarkoitetaan sitä, että laitteistojen vikasietoisuutta ja tietojen päällekkäisyyttä ei lisätty ulkoisilla laitteilla. Keskustietokoneet olivat kuitenkin kalliita monimutkaisia laitteita, jotka skaalautuivat huonosti. (Poitras 2020, luku 1.1-1.11.)

Skaalautuvuus parani, kun siirryttiin Itsenäisiin palvelimiin. Itsenäisiin palvelimiin voitiin suoraan liittää DAS-tallennustila. Palvelimien hallinta voitiin tehdä verkon yli. Verkon yli tapahtuva hallinta kuitenkin lisäsi tarvetta verkkolaitteille. Itsenäisissä palvelimissa oli huono hyötysuhde resurssien käytölle, ja tallennus ja laskentatilalle oli vain yksi epäonnistumispiste. (Poitras 2020, luku 1.1.2 .)

Keskitetty varastointi korvasi keskusyksikön sekä erilliset palvelimet, koska se tarjosi suuremman jaettavan tallennustilan. Keskeisimpänä ominaisuutena keskitetyssä varastoinnissa on RAID, jonka avulla tallennuksen käyttö parani. Keskitetty varastointi lisäsi kuitenkin jälleen kustannuksia ja monimutkaisuutta. (Poitras 2020, luku 1.1.3.)

Virtualisointi ja useat käyttöjärjestelmät hyödyntävät tehokkaasti laitteiston suorituskyvyn, koska useampi käyttöjärjestelmä toimii yhdessä koneessa. Virtualisoinnin kehittyessä hypervisorista tuli hyödyllinen ratkaisu, koske se sisälsi korkean saatavuuden, reaaliaikaisen laitteen siirron järjestelmästä toiseen ja hajautetun resurssien aikataulutuksen. Kehittyneiden ratkaisujen myötä virtuaalikoneille saatiin korkea saatavuus. Virtualisointi lisäsi riippuvuutta keskitetystä varastoinnista, jolloin virtualisointi ja keskitetty varastointi sulautuivat keskenään. Tallennusryhmän lisääntynyt kuormitus ja virtuaalikoneiden levinneisyys johti I/O-varastointiin, jonka seurauksena tarvittiin nopeampia tallennuslaitteita. (Poitras 2020, luku 1.1.5.)

SSD-asemat auttoivat lieventämään I/O:sta johtuvaa pullonkaulaa tarjoamalla paremman suorituskyvyn kuin HDD-levyt, vähentäen samalla laitteistojen tilantarvetta. Seuraava pullonkaula muodostui ohjaimista ja verkosta, koska ne eivät pysyneet kasvaneen suorituskyvyn mukana. (Poitras 2020, luku 1.1.6.)

Hajautettujen itsenäisten järjestelmien lähtökohtana on ajatus, että laitteisto epäonnistuu lopulta. Hajautetut järjestelmät on suunniteltu ratkaisemaan laitteiden epäonnistumisesta johtuvaa ongelmaa jakamalla roolit klusterin solmujen kesken siten, että vian ilmentyessä vika voidaan korjata häiriöttömästi. Mikäli hajautettu järjestelmä ei onnistu korjaamaan vikaa itsenäisesti käyttäjä saa siitä tiedon. Hajautetut järjestelmät, sisältävät useampia epäonnistumispisteitä ja ovat lineaarisesti skaalautuvia järjestelmiä, jotka hyödyntävät samanaikaisuutta ilman pullonkauloja. (Poitras 2020, luku 1.4.3.)

4.1 Virtualisointi

Virtualisointi toimi jo 1970-luvulla IBM:llä, mutta ensimmäinen kaupallinen ratkaisu X86 -tietokoneiden virtualisoimiseksi tapahtui VMwaren toimesta vuonna 2001. Kaksi vuotta myöhemmin tuli avoimeen lähdekoodiin perustuva Xen. (Portnoy 2012, luku 1.)

Virtualisoinnilla tarkoitetaan sitä, että yhteen fyysiseen laitteistoon, kuten palvelimeen, asennetaan useita käyttöjärjestelmiä siten, että ne jakavat kyseisen laitteen laitteistoresurssit kuten muistin, prosessorin, levyt ja asemat. Näitä käyttöjärjestelmiä kutsutaan vieraskäyttöjärjestelmiksi. Vieraskäyttöjärjestelmät toimivat virtualisointiohjelmiston päällä, joka toimii isäntäjärjestelmänä. Vieraskäyttöjärjestelmässä olevat sovellukset pitävät käyttöjärjestelmää kokonaisena. Tämä tarkoittaa sitä, että vieraskäyttöjärjestelmät eivät havaitse eroa sen välillä, että ne toimivat virtualisointiohjelmiston päällä eikä suoraan fyysisellä laitteistolla. Virtualisointiohjelmisto ohjaa pääsyä taustalla olevaan laitteistoon siten, että monet käyttöjärjestelmät voivat toimia toisistaan riippumatta. (Golden 2011, luku 1.1.)

Virtualisointi on kuitenkin laaja-alainen käsite, joka pitää sisällään eri tasoja, kuten palvelin, työpöytä, tallennustila, tiedosto ja verkko. Jokaisella tasolla on omat monimutkaisuudet etuineen. Virtuaalikone on ainoastaan fyysisen tietokoneen datatiedosto, joka voidaan kopioida tai siirtää toiseen koneeseen samalla tavalla kuin normaalit datatiedostot. (Techopedia 2017.)

Virtualisointi tarjoaa IT-toimintaan joustavuutta, koska ohjelmiston voi ohjelmoida pitämään huolta virtuaalikoneiden kunnosta siten, että se voi käynnistää virtuaalikoneita uudelleen tai siirtää palveluita koneista toiseen palvelun tai koneen kaatuessa. Virtualisoinnin avulla voidaan vähentää IT-toimintakustannuksia, koska ei tarvita useita fyysisiä koneita vaan ne voivat toimia yhdellä tietokoneella. Tämä vähentää ylläpitoon liittyviä kustannuksia, yksinkertaistaa infrastruktuuria ja keskittää hallintaa. (Golden 2011, luku 1.2.)

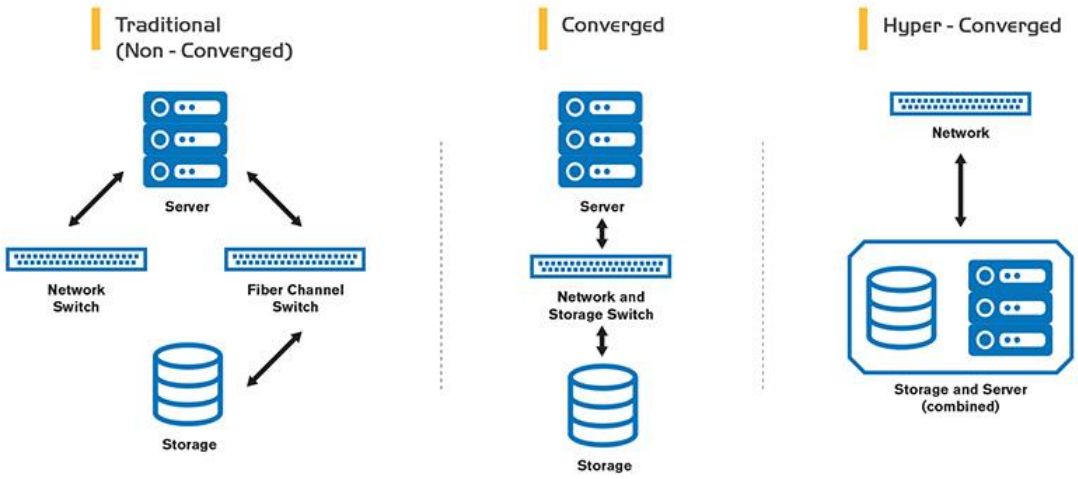
Hypervisor tarjoaa välimuistitekniikkaa, jonka avulla käyttäjä voi hylätä käyttöjärjestelmään tehdyt muutokset käynnistämällä käyttöjärjestelmän uudelleen tunnetusta tilasta. Virtualisoinnin hyötyinä voidaan pitää edullista tai maksutonta käyttöönottoa ja resurssien täyttä hyödyntämistä. (Techopedia 2017.)

4.2 Hyperkonvergenssi

Hyperkonvergenssi (HCI) pohjautuu CI-järjestelmän idealle ja sen arkkitehtuurille. Yhdistetyssä järjestelmässä eli CI:ssa komponentit kuten tallennus-, verkko-, palvelin-, ja hallintalaitteet ovat erillisiä, mutta toisistaan riippumattomia osia. Hyperkonvergoidun infrastruktuurin ideana on, että kaikki palvelinkeskuksessa tarvittavat ominaisuudet sijaitsevat yhdessä rungossa ilman erillisiä laitteita. Rungolla tarkoitetaan tavallista PC-palvelinta, jossa tallennus-, verkko-, palvelin- ja hallintatoiminnot voidaan määrittää ohjelmistokerroksesta. Hyperkonvergoidun infrastruktuurin avulla voidaan korvata useita erillisiä tuotteita kuten palvelimet, tallennuslaitteet (HDD, SSD ja varmuuskopio), tallennusverkkojen ohjauslaitteet (SAN), kuormituksen tasapainottimet ja WAN-optimointi. (Azeem & Sharma 2017.)

Hyperkonvergoidun infrastruktuurin hyötyinä pidetään sitä, että sillä voidaan saavuttaa palvelinkeskusten käytettävyys ja luotettavuusvaatimukset. Infrastruktuurin hallinta ja työmäärät otetaan käyttöön yhden käyttöliittymän kautta. HCI yksinkertaistaa hallintaa ja lisää resurssien käyttöastetta hyvällä virtuaalisen työpöydän suorituskyvyllä. HCI helpottaa työpöydän virtualisointia, parantaa I/O:ta ja vähentää käynnistysmyrskyjen vaikutuksia. (Azeem & Sharma 2017.)

Kuviossa 2 on kuvattuna HCI-ympäristön ero tavanomaiseen ei-konvergoituun järjestelmään, jossa on erillinen tallennusverkko, palvelin ja kytkimet. Konvergoitu infrastruktuuri yhdistää SAN-pohjaisen tallennusverkon yhdeksi laitteeksi. Hyperkonvergoidussa järjestelmässä kaikki ovat yhdessä laitteessa.



Kuvio 2: Palvelinarkkitehtuurit (Helixstorm 2020)

4.3 Nutanix

Nutanix on HCI-ohjelmiston toimittaja, joka keskittyy pääsääntöisesti seuraaviin asioihin: HCI, Cloud ja Hypervisor. Älykkään ohjelmiston (AOS) avulla tarjotaan käyttöliittymä (Prism). Nutanixin käyttöliittymään on panostettu paljon ja se on kirjoitettu HTML5:llä. Prism-käyttöliittymän avulla hallitaan ja käytetään koko Nutanix-alustaa. Nutanix on suunniteltu laajennettavaksi ja liitettäväksi arkkitehtuuriksi, jossa voidaan ottaa uusia ominaisuuksia käyttöön ohjelmistopäivitysten avulla. Nutanix tukee monia laitteistoalustoja ja useita hypervisoreita kuten AHV, ESXi ja Hyper-V. Tästä syystä siihen voidaan integroida monia suurimpia pilvipalvelutoimijoita. AHV on Nutanixin suunnittelema hypervisor, joka yksinkertaistaa päivityksiä ja tarjoaa palveluina suojausta ja salausta. (Poitras 2020, luku 3.)

Nutanix tarjoaa helpon hajautetun ohjelmistoarkkitehtuurin, jonka avulla päästään eroon SAN-verkoista. Tämän avulla säästetään mm. elinkaarikustannuksissa. Kyseinen arkkitehtuuri toimii Googlella ja monella muulla suurella yrityksellä, joten sitä voidaan pitää luotettavana. Hajautettu ohjelmistoarkkitehtuuri tarjoaa korkeaa suorituskykyä, koska suorituskyky jakaantuu useamman laitteen kesken. Nutanix DSF yhdistää tallennustilan, laskentaresurssit, ohjainlogiikan ja hypervisorin integroiduksi järjestelmäksi. Tämä tarjoaa korkeaa vikasietoisuutta ja käytettävyyttä. Ohjelmistoarkkitehtuuri tukee VMware vSphereä, KVM-tekniikoita ja muita teollisuuden standardien mukaisia hallintajärjestelmiä. Solmut yhdistyvät helposti klusteriksi ja skaalautuvat ilman käyttökatkoja. (Nutanix Data Sheet 2020.)

Tallennusta Nutanixissa tapahtuu DSF:n avulla. DSF yhdistää solmuihin suoraan liitetyt tallennuslaitteet ohjelmistoon, jolloin saadaan objekti-, lohko- ja tiedostotallennusominaisuudet käyttöön. DSF näyttää keskitetyltä tallennusryhmältä, vaikka suoritusteho tapahtuu paikallisesti. (Mellor 2019.)

Ryhmä Nutanix-solmuja luo klusterin, joka vastaa Prism- ja AOS-ominaisuuksien tarjoamisesta. Kaikki palvelut ja komponentit jaetaan klusterissa oleville CVM-moduuleille korkean käytettävyyden ja lineaarisen suorituskyvyn saavuttamiseksi. Metatietojen ja tietojen jakaminen kaikkiin solmuihin varmistaa hyvän suorituskyvyn tiedonsiirron aikana. MapReduce Frameworkin tarkoituksena on hyödyntää klusterin koko kapasiteettia siten, että toimintoja suoritetaan samanaikaisesti. Klusterin solmujen määrän kasvaessa tietyt toiminnot tehostuvat, koska työkuorma jakautuu solmujen kesken. (Poitras 2020, luku 3.4.)

Nutanix Community Edition on ilmainen versio Nutanixista, se voidaan asentaa 1,3 tai 4 solmuun. Nutanix vaatii neliytimisen prosessorin, joista kaksi ydintä on määritetty CVM:n käyttöön. Keskusmuistia on oltava vähintään 16GB. Maksimimäärä SSD- tai HDD-laitteita on 4/solmu. Minimissään 500GB ja maksimissaan 18TB cold tier. Hot tier 200GB SSD tai suurempi. 8GB boot device/solmu. (Getting Started with Nutanix Community Edition 2020, 5-6.)

4.4 Proxmox

Proxmox VE on avoimeen lähdekoodiin perustuva Debian Linux -jakelu, jota kutsutaan myös hypervisoriksi tai Virtual Machine Monitoriksi palvelimen virtualisointiin. Proxmoxiin pystyy asentamaan monia erilaisia käyttöjärjestelmiä, kuten Windows, Unix, ja Linux. Proxmoxilla voidaan klusteroida monia tietokoneita tai palvelimia ryhmiksi, jotka jakavat tietokoneiden laitteistoresurssit keskenään. Proxmoxilla voidaan tehdä reaaliaikaista virtuaalikoneiden siirtoa eli migraatiota ilman käyttökatkosta. Korkean saatavuuden avulla yhden solmun epäonnistuttua, korkean saatavuuden ryhmässä olevat virtuaalikoneet siirtyvät klusterissa olevaan toimivaan solmuun. Silloitetun verkon avulla voidaan rakentaa yksityisiä verkkoja virtuaalikoneiden kesken, sekä jakaa verkko useampaan virtuaaliseen lähiverkkoon. Joustava tallennus voidaan toteuttaa usealla tavalla kuten LVM, ISCI, NFS, Gluster filesystem tai Cdph:n avulla. Varmuuskopioita voidaan ajastaa ja tallentaa paikallisesti mihin tahansa edellä mainituista tallennusvaihtoehdoista. Proxmoxin graafisen käyttöliittymän lisäksi muutoksia järjestelmään voidaan tehdä komentoriviä käyttäen. (Cheng 2014, luku 1.)

HCI-ympäristön tekoon Proxmoxin ja Ceph-klusterin avulla tarvitaan ainakin kolme tietokonetta tai palvelinta, ja niiden olisi mieluiten oltava samanlaisia. Prosessorin korkea ydintaajuus vähentää viivettä, joten prosessoriin kannattaa panostaa. Hyvänä muistisääntönä pidetään, että Ceph-palvelulle olisi syytä varata yksi prosessorin ydin ja 1TB dataa varten 1GB keskusmuistia per OSD. Daemon vaatii oletuksena 3-5 Gt keskusmuistia, mutta se voidaan itse määrittää. Palveluiden toiminnan kannalta verkon olisi syytä olla 10 GbE, jottei verkosta muodostu pullonkaulaa, koska verkossa voi liikkua 25, 40 tai jopa 100GT/s. SSD-asemien avulla voidaan vähentää palautumisaikaa ja parantaa kirjoitus- ja lukunopeutta HDD-levyihin verrattuna. Suositeltavaa on käyttää tasakokoisia ja hajautettuja levyjä tai asemia klusterissa, koska 4 x 500Gt per solmu on parempi kuin esimerkiksi 1TB ja 250gt per solmu. RAID-ohjainta olisi syytä välttää, koska Ceph on suunniteltu käsittelemään koko HDD-levyä tai SSD-asemaa itsenäisesti. Tällöin Ceph voi käsitellä dataobjektien redundanssia ja useita rinnakkaisia kirjoituksia. RAID-ohjaimen korvaajana voidaan käyttää Host bus -adapteria. (Proxmox VE Administration Guide 2020, 139-141.)

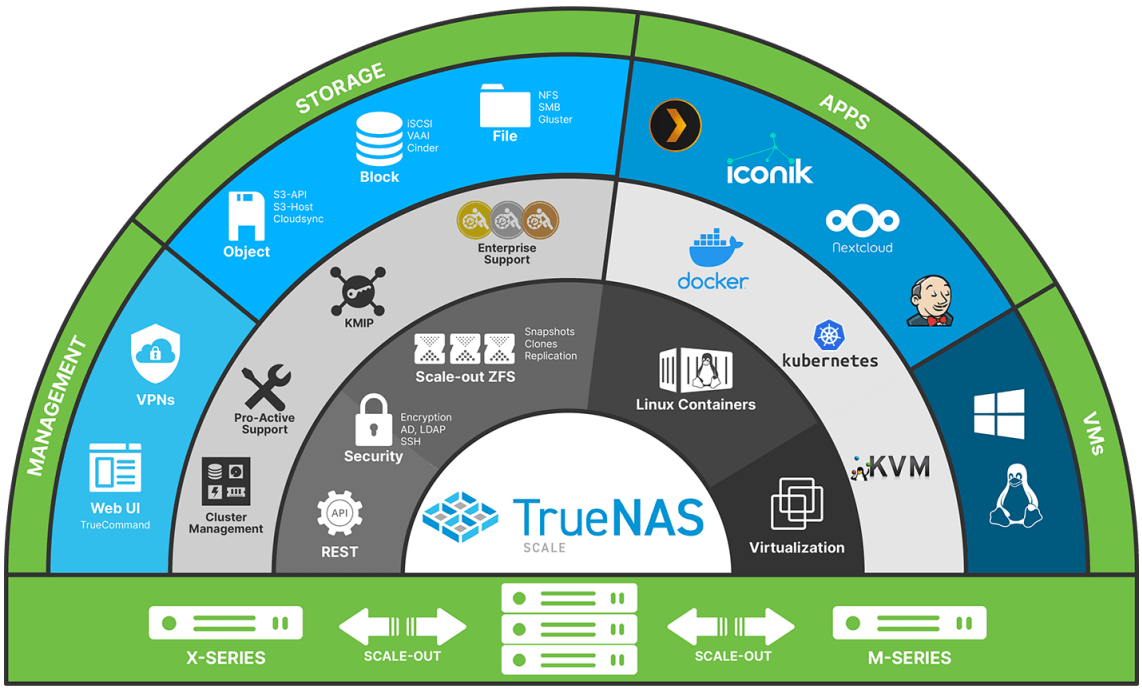
Proxmox VE on ilmainen, mutta jos halutaan saada vakaa Proxmox VE Enterprise -tietovarasto luotettavilla ohjelmistopäivityksillä, suojausparannuksilla ja tarvittaessa teknistä tukea tai apua, siitä pitää maksaa. Hinnat määräytyvät sen mukaan, miten ja millaista apua tarvitsee. Community tuki maksaa 85 €/vuosi. Community pitää sisällään edellä mainitut asiat. Basic 270 €/vuosi poikkeaa Communitysta siten, että sen avulla saadaan tuki suoraan asiakasportaalin kautta ja kolme tukilippua yhden arkipäivän vastausajalla. Standard 398 €/vuosi, kymmenen tukilippua/vuosi, neljän tunnin vastausajalla arkipäivisin ja etähallintayhteys SSH:n kautta. Premium 796 €/vuosi, rajoittamattomat tukiliput kahden tunnin vastausajalla arkipäivisin sekä SSH-etätuki. (Proxmox 2020b.)

4.5 TrueNAS SCALE

TrueNAS SCALE on iXsystemsin uusin projekti. SCALE on avoimen lähdekoodin HCI-ratkaisu, joka käyttää suurta osaa TrueNAS 12.0 -lähdekoodista, mutta siihen on lisätty muutama ominaisuus. SCALE tulee sanoista: Scale-out, Converged, active-active, Linux containers, Easy-to-manage. Laajennettava, yhdistetty, aktiivisesti aktiivinen, Linux-kontit ja helposti hallittava infrastruktuuri. Edellä mainitut asiat ovat samalla SCALE:n tavoitteet. SCALE on tarkoitus julkaista vuonna 2021. (Mulford 2020.)

SCALE eroaa muusta TrueNAS-perheestä siten, että se käyttää Debian Linux 11:tä eikä FreeBSD:tä. Debian Linux mahdollistaa SCALE-nimessä mainittujen tavoitteiden saavuttamisen. Open ZFS 2.0, FreeBSD ja Linux tarjoavat TrueNAS-perheelle monimuotoisuutta. SCALE ei vaikuta TrueNAS CORE:n ja TrueNAS Enterprisen tukeen tai kehittämiseen FreeBSD:llä vaan SCALE:n on tarkoitus täydentää TrueNAS-perhettä. SCALE:n lähdekoodi on saatavilla GitHubissa ja ohjelmiston kehitysversio on ladattavissa verkosta ilmaiseksi. (Moore 2020.) OpenZFS ja Gluster mahdollistavat ZFS-laajennuksen, joka on erittäin vakaa ja tehokas pakkaamaan. HCI mahdollistaa klusterin tekemisen satojen laskenta- ja tallennussolmujen avulla. KVM VM-, Kubernets ja Docker-tuella on helppo lisätä sovelluksia tarpeen mukaan ja järjestelmää käytetään TrueCommandilla. (iXsystems 2020.)

Kuviossa 3 on kuvattuna TrueNAS SCALE:n ominaisuudet. Tärkeintä on huomioida, että SCALE koostuu moduleista. Modulit on jaoteltu hallintaan, tallennukseen, ohjelmistoihin ja virtualisointiin ja kaiken pohjalla on skaalautuva järjestelmä.



Kuvio 3: TrueNAS SCALE infrastruktuuri (iXsystems 2020)

5 Tallennusjärjestelmät

Kasvavan tiedon määrä vaatii yhä enemmän tallennuskapasiteettia ja sen parempaa hyödyntämistä. Tallennuskapasiteettivaatimukset määräävät sen paljonko tallennustilaa vaaditaan. Kapasiteettivaatimuksissa on tarpeellista huomioida tallennettavan tiedon tietotyypit, koska asiakirjat tarvitsevat vähemmän tilaa kuin esimerkiksi videotiedostot. SSD-asemien avulla päästään tietoihin nopeammin käsiksi kuin HDD-levyillä, koska SSD-asemien luku- ja kirjoitusnopeudet ovat nopeammat. Virranmenetys ei hävitä tietoja flash-muistista toisin kuin mekaanisien levyjen kanssa. Tästä syystä SSD-asemat ovat vikasietoisempia kuin HDD-levyt, koska mekaaniset levyt tarvitsevat sisäisen akun tai varmuuskopion pitääkseen tiedot muistissa. Edellä mainituista syistä monet yritykset ovat ottaneet käyttöön flash-matriisit, joissa on NAND flash -pohjaisia SSD-asemia lisärakenteena tai korvikkeena levyasemille. SSD-asemat eivät ole kuitenkaan niin kestäviä kuin mekaaniset levyt. Tämä tekee hybridiratkaisuista varteenotettavan vaihtoehdon. Yleisimpiä tallennustapoja ovat tiedosto-, objekti- ja lohkotallennus. (Rouse 2018.)

Tiedostotallennuksessa tiedot tallennetaan yhtenä tietona kansion sisällä samalla tavalla kuin paperinpalaset järjestetään kansioon. Tietoja hakiessa tietokoneen on tiedettävä polku tietojen löytämiseksi. Tiedostoihin tallennetut tiedot haetaan ja järjestetään käyttämällä rajoitettua määrää metatietoja. Metatiedot kertovat tietokoneelle missä tietoja säilötään ja toimivat samalla tavalla kuin kirjastokortisto, mutta datatiedostoille. Tiedostotallennus on vanhin ja eniten käytetty tietojen tallennusjärjestelmä suorille ja verkkoon liitetyille tallennusjärjestelmille. Tiedostotallennus sisältää laajat ominaisuudet ja sen avulla voidaan tallentaa melkein mitä tahansa. Ongelmana on, että tiedostopohjainen järjestelmä on vaikeasti laajennettava, koska tiedostopohjainen järjestelmä vaatii laajentuessaan lisää järjestelmiä eikä pelkän kapasiteetin lisäys riitä. (Red Hat 2020.)

Lohkotallennustila pilkkoo tiedot lohkoiksi ja tallentaa tiedot erillisiksi paloiksi. Jokaiselle tietolohkolle annetaan yksilöllinen tunniste, jota hyödyntämällä tallennusjärjestelmä sijoittaa pienemmät tiedonpalat niille sopivimpaan paikkaan. Lohkotallennus ei ole riippuvainen yhdestä tietopolusta, kuten tiedostojen tallennus, ja tämä mahdollistaa nopeamman tietojen noudon. Jokainen lohko toimii itsestään ja ne voidaan itse osioida. Tämän avulla lohkotallennusta voidaan käyttää eri käyttöjärjestelmissä. Itsenäinen toiminta ja osiointi antaa käyttäjälle täydellisen vapauden määrittää tietoja. Lohkotallennus on luotettava ja tehokas tapa tallentaa tietoja helpolla käytettävyydellä ja hallinnalla. Erityisesti lohkotallennus toimii hyvin suurien muuttuvien tietokantojen kanssa, mutta se voi olla kuitenkin kallista. Lohkotallennuksella on rajoitettu kyky käsitellä metatietoja saamaan tapaan kuin tiedostotallennuksellakin. (Red Hat 2020.)

Objektipohjaisessa tallennuksessa tiedostot hajotetaan palasiksi hajautettujen laitteistojen kesken. Tietojen hajotessa erillisiksi yksiköiksi eli esineiksi, esineet säilytetään yhdessä arkistossa, kun taas tiedot säilytetään tiedostoina kansioissa tai lohkoina palvelimilla. Objektit löytyvät hajautetusta järjestelmästä yksilöllisen tunnisteen avulla. Objektitallennus toimii hyvin staattiseen dataan, koska objektin kirjoittaminen on hidasta ja monimutkaisempaa kuin tiedostojen tallennus. Objektitallennuksessa objektin muuttuessa se on kirjoitettava kokonaan uudelleen, mutta se on ketterä, tasainen ja sen skaalautuvuus on hyvä. (Red Hat 2020.)

Tärkeä ero objekti- ja lohkotallennustilan välillä on se, miten ne käsittelevät metatietoja. Objektitallennuksessa metatiedot sisältävät hyvin yksityiskohtaista tietoa objektiin tallennetuista datatiedostoista. Metatiedot voivat sisältää erittäin yksilöllistä tietoa esimerkiksi siitä, missä video on kuvattu, millä kameralla se on kuvattu tai mitä missäkin videon kehyksessä on. (IBM 2019.)

5.1 ZFS

ZFS on Sun Microsystems Inc. -insinöörien vuonna 2001 aloittama kehitystyö Unix-pohjaiseen Solaris-käyttöjärjestelmään. Vuonna 2005 Sun julkaisi ZFS-lähdekoodin yleisellä kehitys- ja jakelulisenssillä (CDDL) osana OpenSolaris-käyttöjärjestelmää. Lähdekoodin parannusten jälkeen ZFS siirrettiin myös muihin käyttöjärjestelmiin, kuten FreeBSD, Linux, Mac OS X. (Rouse 2017.)

ZFS-tiedostojärjestelmille on ominaista tietojen eheys, korkea skaalautuvuus ja sisäänrakennetut tallennusominaisuudet. Replikoinnilla voidaan tehdä kopio jostain, Deduplicationilla poistetaan tarpeettomat kopiot, pakkaamalla voidaan vähentää tietojen esittämiseen vaadittavien bittien määrää, tilannekuva on joukko viitemerkkejä tietylle ajankohdalle ja kloonaamalla voidaan tehdä identtinen kopio jostain. (Rouse 2017.)

ZFS on suunniteltu toimimaan yhdessä palvelimessa, johon voidaan liittää suuria määriä tallennusasemia. ZFS yhdistää käytettävän tallennustilan ja hallitsee kaikkia levyjä yhtenä kokonaisuutena. Käyttäjä voi lisätä tallennusasemia altaaseen, josta tiedostojärjestelmän kapasiteettia voidaan kasvattaa. ZFS on erittäin skaalautuva ja se tukee suuria tiedostokokoja. ZFS tallentaa vähintään kaksi kopiota metatiedoista joka kerta kun tiedot kirjoitetaan levylle tai asemalle. Metatiedot sisältävät tietoja kuten levysektorit joihin data on tallennettu, datalohkojen koon ja binääristen numeroiden tarkistussumman. (Rouse 2017.)

Käyttäjän pyytäessä pääsyä tiedostoon, tiedoston tarkastussumman algoritmi suorittaa laskennan varmistaakseen, että haetut tiedot vastaavat levylle kirjoitettuja alkuperäisiä bittejä. Jos tarkastussumma havaitsee epäjohdonmukaisuuden, virheelliset tiedot merkitään. Järjestelmissä, joissa on peilattu tallennus tai RAID:n ZFs versio ZFS noutaa oikean kopion toisesta asemasta ja korjaa vioittuneen datan. (Rouse 2017.)

5.2 RAID

RAID tulee sanoista Redundant array of independent disk, ja sillä tarkoitetaan redundanttia ryhmää itsenäisiä levyjä. Redundantti sisältää saman tiedon ilman datan päällekkäisyyksiä paremmalla vikasietoisuudella. RAID on joukko tallennuslaitteita, jotka voidaan liittää toisiinsa. Tällä tavalla saavutetaan parempi vikasietoisuus, parempi suorituskyky ja voidaan lisätä tallennustilaa. Tämä tapahtuu siten, että kaksi tai useampi fyysinen levy yhdistetään yhdeksi loogiseksi yksiköksi, joka kuitenkin näkyy yhtenäisenä tallennuslaitteena käyttöjärjestelmässä. RAID voidaan toteuttaa ohjelmiston tai laitteen avulla, jota kutsutaan RAID-ohjaimeksi. Kun liitetään useampi levy RAID-ohjaimeen, saadaan laitteistosovellus. Kun osioidaan levy useampaan loogiseen levyyn, saadaan ohjelmistosovellus. RAID-tasoja on monia ja niitä voidaan yhdistää keskenään. (Rathnam 2020.)

RAID 0:ta käytetään palvelimien suorituskyvyn parantamiseksi. Tällaisessa kokoonpanossa tiedot kirjoitetaan usealle levylle kaistaleeksi. Jokainen näistä levyistä voi lukea tai kirjoittaa tietoja samaan aikaan, mikä parantaa I/O-suorituskykyä. Haittapuolena on, että RAID 0:lla ei saavuteta redundanssia, joten jos yksi levy vikaantuu se vaikuttaa koko ryhmään. Lisäksi on suuri riski datan vioittumiseen tai katoamiseen. (Rathnam 2020.)

RAID 1:tä käytetään vikasietoisuuden parantamiseen. Tässä käytettävää tekniikkaa kutsutaan datapeiliksi, jossa yhden levyn tiedot peilataan ja kopioidaan toiselle. Ensisijaisen levyn epäonnistuessa, toissijainen levy tulee käyttöön toimittaen samat tiedot saumattomasti. Haittapuolena on, että suorituskyky kärsii. RAID 1 aiheuttaa lisäkustannuksia, koska käyttöönotettava tallennustila puolittuu. (Rathnam 2020.)

RAID 2:ta käytetään virheenkorjaukseen. Siinä käytetään tiedon hajautusta: kun data on hajautettu eri laitteille, jotkut levyt sisältävät virheen tarkastus- ja korjaustiedot (ECC). RAID 2:ta ei kuitenkaan usein käytetä, koska se on samankaltainen kuin RAID 3 eikä sillä ole huomattavaa etua RAID 3:een nähden. RAID 3:n ja RAID 2:n ero on oikeastaan se, että ECC:lle on erillinen levy, jolla RAID 3 havaitsee virheitä. Tämä kokoonpano tekee tietojen palauttamisesta helpompaa, koska se laskee muiden levyjen pariteettitiedot ja vertaa niitä ECC:hen virheiden tunnistamiseksi. RAID 3 ei kuitenkaan pysty käsittelemään päällekkäisiä kirjoitus- ja lukutoimintoja. (Rathnam 2020.)

RAID 4 on samanlainen kuin RAID 3, paitsi että se tukee suurempaa tietojen hajautusta ja näin saavutetaan nopeampi päällekkäinen I/O- lukutoiminto, mutta päällekkäinen I/O-kirjoitusoperaatio ei ole mahdollinen, koska kaikkien kirjoitusoperaatioiden on päivitettävä pariteettitiedot. (Rathnam 2020.)

RAID 5 on suosituin kokoonpano, jota käytetään yrityksissä ja NAS-palvelimissa, koska siinä on korkea suorituskyky sekä vikasietoisuus. RAID 5:n data- ja pariteettitiedot tallennetaan yhdessä ja levitetään eri levyille. Yhden levyn epäonnistuessa, data voidaan luoda saumattomasti uudelleen muilta levyiltä. Nämä rekonstruoidut tiedot ovat virheettöminä jokaisessa levyssä olevien pariteettilohkojen takia. Tämä mahdollistaa samanaikaisen tietojen lukemisen sekä kirjoittamisen. Jos palvelimella suoritetaan useita kirjoitusoperaatioita, suorituskyky heikkenee, koska tässä tapauksessa tiedot kopioituvat monille palvelimille. Lisäksi pariteettitarkastusten vuoksi tietojen palauttaminen varmuuskopiosta voi viedä paljon aikaa. (Rathnam 2020.)

RAID 6 on samantapainen kuin RAID 5, mutta se lisää toisen pariteetin, joka on jaettu kaikille asemille. RAID 6:ssa ei haittaa, vaikka kaksi levyä epäonnistuu. Toisaalta on harvinaista, että kaksi tai useampi levy vikaantuu samanaikaisesti. RAID 6 on paljon hitaampi kuin RAID 5, joten sitä harvemmin käytetään. (Rathnam 2020.)

RAID 10 yhdistää RAID 0:n ja RAID 1:n tarjotakseen paremman suorituskyvyn, koska siinä käytetään peilausta ja tietojen hajautusta. Tässä kokoonpanossa peilausta seuraa tietojen hajautus, mikä tarjoaa sekä redundanssia että parempaa suorituskykyä. Tässä vaaditaan kuitenkin vähintään neljää taulukkoa, joista kaksi ensimmäistä peilaa dataa ja loput kaksi hajauttaa tietoja. RAID 10 on suosittu yrityksissä, joissa käsitellään arkaluonteista tietoa ja joissa vaaditaan korkeita transaktiotietokantoja. (Rathnam 2020.)

5.3 Ceph

Ceph-projekti alkoi vuonna 2004 Sage Weilin Kalifornian yliopiston tohtoriprojektina. Vuonna 2006 Ceph tuli saataville avoimen lähdekoodin lisenssillä, ja vuonna 2012 Sage Weil perusti yhdessä insinööritiimin kanssa yrityksen, joka tarjosi Ceph-version yritysten käyttöön. Vuonna 2014 Red Hat inc. osti yrityksen, jolloin Weil otti Cephin pääarkkitehdin roolin Red Hatissa. (Rouse 2016.)

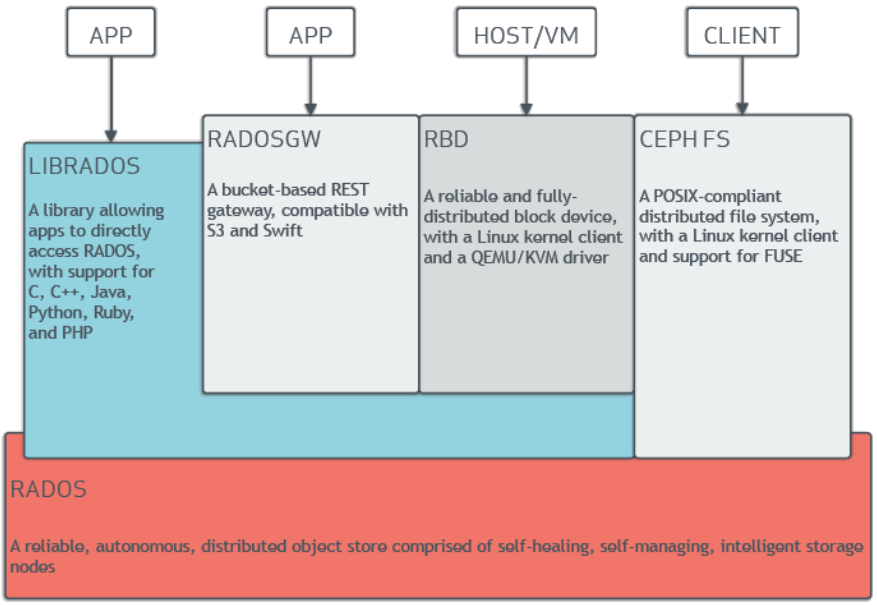
Ceph on suunniteltu tarjoamaan erittäin skaalautuvaa objekti-, lohko- ja tiedostopohjaista tallennustilaa yhtenäisessä järjestelmässä. Cephin hajautetulla tiedostojärjestelmällä saavutetaan skaalautuvaa tallennustilaa useaan petatavuun asti luotettavasti hyvällä suorituskyvyllä. (Rouse 2016.)

Ceph käyttää CRUSH-nimistä algoritmia (Controlled replication uder scalabe hashah), jonka avulla tiedot jakautuvat tasaisesti jokaiselle klusterissa olevalle SSD-asemalle tai HDD-levylle mahdollistaen sen, että kaikki klusterissa olevat solmut voivat hakea tietoja nopeasti. (Rouse 2016.) CRUSH-algoritmi hakee ja tallentaa tietoja sekä mahdollistaa kommunikoinnin OSD:n kanssa. Algoritmin avulla Ceph täyttää hyperkonvergenssin vaatimukset skaalautuvuuden, mo-

nien virhepisteiden ja suorituskyvyn osalta. CRUSH-kartan avulla monitoroidaan tallennusjärjestelmän tilaa ja kartan avulla voidaan esimerkiksi havaita mahdolliset viat järjestelmässä. (Ceph 2020.)

Ceph OSD (Objeckt Storage Daemons) mahdollistaa tallentamisen objekteihin. OSD:llä hallitaan tietojen replikointia, palauttamista ja tasapainottamista. OSD:n avulla toimitetaan Ceph Monitorile tilatietoja. (Ceph 2014a.) Ceph Monitor ylläpitää klusterin yleistä terveyttä ja pitää huolta klusterikartan tilasta. Näihin tiloihin kuuluvat monitorikartta, OSD-kartta, sijoitusryhmäkartta (PG) ja CRUSH-kartta. Monitorin tarkoituksena on vastaanottaa tilatietoja muilta komponenteilta. Monitori kykenee ylläpitämään karttoja ja tarjoaa niitä muille OSD- ja Monitor-solmuille. (Ceph 2014a.)

Kuviossa 4 on kuvattuna Cephin arkkitehtuuri. RADOS on se, minkä päällä kaikki toimivat mahdollistaen luotettavan objektitallennuksen. Librados-kirjasto mahdollistaa suoran pääsyn Cephiin, ja se tukee monia ohjelmointikieliä. Ceph RGW (Objeckt Gateway / Rados Gateway) on sovellusliittymä, joka käyttää RESTful API -rajapintaa. Ceph RBD (Raw Block Device) tarjoaa hajautetun lohkotallennuksen virtuaalikoneille. CephFS on hajautettu tiedostotallennusalusta, jonka avulla hallitaan tiedostojen jakamista.



Kuvio 4: Ceph-arkkitehtuuri (Ceph 2020b)

6 Tutkimuksessa käytetyt menetelmät

Tutkimuksellista kehittämistyötä voidaan käyttää yrityksen kehittämistarpeeseen käytännöllisen ongelman ratkaisemiseksi. Tyypillisesti työn tarkoituksena on etsiä parempia vaihtoehtoja

tai ottaa käytäntöön uusia ratkaisuja. Tutkimuksellisen kehittämistyön eroavaisuus tieteelliseen tutkimukseen nähden on toiminnan päämäärässä. Kehittämistyön tarkoituksena on saada aikaan käytännön parannuksia tai uusia ratkaisuja. (Ojasalo, Moilanen & Ritalahti 2014, luku 2.1.)

Opiskelijan kehittämistyö voi olla osa muutostyötä. Edellä mainitussa tapauksessa kehittämistyötä voidaan hyödyntää muutostyön apuvälineenä. Kehittämistyö etenee prosessinomaisesti. Kehittämiskohde tunnistetaan ja kerätään tarpeeksi tietoa, että aihe ymmärretään. Kehittämistyön tavoitteena voi olla esimerkiksi uusi palvelu tai sen kehittäminen. Kehittämiskohteeseen perehdytään käytännön ja teorian avulla, jolloin kehittämiskohteesta saa kattavamman kuvan. Aiheen rajauksen ja määrittelyn tarkoituksena on saada tietoon mihin keskitytään. Kehittämistyötä lähestytään tietoperäisesti menetelmiä hyödyntäen. Toteutus on osa kehittämistyötä, jossa edellä kuvaavat asiat tehdään käytännössä, jonka jälkeen prosessia ja lopputulosta voidaan arvioida. Pitää huomioida, että kehittämistyö on jatkuvaa ja tarpeen mukaan aloitetaan uusi sykli. (Ojasalo, Moilanen & Ritalahti 2014, luku 2.2.)

Kehittämistehtävän määrittämisessä pohditaan päämäärää huolellisesti, että onnistumiselle voidaan asettaa mittarit. Asetettuja mittareita käytetään hyödyksi prosessin ja lopputuloksen arvioinnissa. Mittareina käyvät myös laadulliset mittarit, kuten havainnointi tai haastattelu. Kehittämistyö voi kuitenkin muuttua tai suuntautua uudelleen prosessin edetessä. Tutkimustyön raportin mallina voi olla Oivalluttava-Perinteinen malli, jossa tietoperustan kuvaamiseksi kehittämistyössä sisältää referoinnin lisäksi tekijä omaa ajattelua siten, että tietoperustan ja kehittämistyön väliltä löytyy yhteys. (Ojasalo, Moilanen & Ritalahti 2014, 2.2.)

6.1 Havainnointi

Havainnoinnin avulla kerätään havaintoja tutkittavasta aiheesta siten, että tietoisesti tarkkaillaan. (Uusitalo 1995, Vilkan 2006, 37 mukaan.) Laadullisessa tutkimuksessa voidaan hyödyntää havainnointia, koska se on hyvin yleinen tapa tutkia tekstejä ja kuvia. Tieteellisen havainnoinnin avulla pyritään järjestelmällisesti tarkkailemaan. Tieteellinen havainnointi mahdollistaa sen, että tieto saadaan heti. (Hirsjärvi ym. 2004, 201-203.) Havainnointi voi olla osallistuvaa tai ei-osallistuvaa suoraa havainnointia. Näitä on kuitenkin hyvin hankala erottaa toisistaan, koska vasta kun tutkija on selvillä tutkittavasta kohteesta tutkimuksesta vai tulla osallistuvaa. (Vilkka, 2006, 42.) Järjestelmällinen tarkkailu on perusteellista ja edellyttää tutkijalta osaamista laatia ja jaotella ongelmia. Tutkija voi tunneperäisesti sitoutua tutkittavaan ja tästä syystä havainnointia on kritisoitu, koska tutkimus ei silloin ole täysin puolueeton. Havainnoidessa ei välttämättä pystytä tallentamaan havaintoja välittömästi, ja tästä syystä havainnoista voi tulla muistiperäisiä. (Hirsjärvi ym. 2004, 202-203.)

6.2 Haastattelu

Haastattelu on yksi yleisimmistä tavoista kerätä tietoa laadulliseen tutkimukseen. Haastattelu on yksinkertaisuudessaan sitä, että toinen henkilö esittää toiselle kysymyksen, jolloin haastattelu on eräänlainen keskustelu, joka tosin tapahtuu tutkijan johdattamana ja hänen aloitteestaan. Haastattelu on vuorovaikutteinen tapahtuma, jossa keskusteluun osallistujat ovat vaikutuksessa keskenään. Avoimessa haastattelussa tilanne muistuttaa kaikkein eniten edellä mainittua vapaamuotoista keskustelua aiheesta. Syvähaastattelulla tarkoitetaan sitä, että haastateltavan kanssa käydään läpi useita avoimia haastatteluja. Avoin haastattelu on joustava ja sen avulla voidaan selvittää jokin asia syvällisesti, mutta avoin haastattelu vaatii kuitenkin paljon aikaa. (Eskola & Suoranta 1998, luku 3.)

6.3 Hiljainen tieto

Hiljainen tieto on Michael Polanyitän 1966 esittelemänä sitä, että tiedämme enemmän kuin pystymme kertomaan. Tällä tarkoitetaan, että suurin osa tiedosta on piilossa ja se mitä ilmaisemme, on vain hyvin pieni määrä siitä kokonaistiedosta mitä on olemassa. Hiljaisella tiedolla tavanomaisesti kuvataan kokemusperäistä tietoa, jota kerääntyy vuosien saatossa. (Pohjalainen 2012, 2.) Hiljaista tietoa voidaan kerätä tarkkailemalla työntekijän työntekoa sekä sitä voidaan kerätä haastattelemalla ja kyselyiden avulla. Hiljaista tietoa voi olla hankalaa ilmaista, koska se ei useinkaan ole konkreettista. (Pohjalainen 2012, 9.) Hiljainen tieto on tarpeen saada yrityksissä talteen, koska ikääntyvien työntekijöiden ja ammattitaitoisten osaajien poistuttua työyhteisöstä myös arvokas kokemuksella saatu tietotaito katoaa, koska sitä ei ole aikaisemmin ilmaistu. (Jalonen 2014.)

6.4 Reliabiliteetti ja validiteetti

Laadullisen tutkimuksen tutkijan täytyy jatkuvasti pohtia tekemiään ratkaisuja ja samaan aikaan ottaa kantaa analyysin laajuuteen ja tekemänsä työn luotettavuuteen. Apuna analyysiä tehdessä ei ole kuin omia ja kollegojen oletuksia, jotka pohjautuvat teoreettiseen oppineisuuteen. Laadullisessa tutkimuksessa tutkija on oman tutkimuksensa keskeinen tutkimusväline. Laadullisessa tutkimuksessa arviointi koskee koko tutkimusprosessia ja sisältää paljon enemmän omaa pohdintaa toisin kuin määrällinen tutkimus. Perinteisesti ymmärrettyinä validiteetti ja reliabiliteetti eivät sellaisinaan sovellu kvalitatiivisen tutkimuksen luotettavuuden perusteiksi. (Eskola & Suoranta 1998, luku 5.)

7 Toteutus

Projektille asetettiin alustava aikataulu. Koska kyseessä on yrityksen omaan tarpeeseen tarkoitettu kehittämistyö, aikataulu on joustava ja suuntaa antava. Taulukossa 5 on Gantt-kaavion avulla esitetty projektin aikataulu sekä eri työvaiheet. Vihreällä on merkattu kunkin työvaiheen kesto. Osassa työvaiheet on aikataulutettu päällekkäin, koska vaiheet liittyvät suoranaisesti toisiin vaiheisiin.

Projektin aikajana	Toukokuu	Kesäkuu	Heinäkuu	Elokuu	Syyskuu	Lokakuu
Esitutkimus						
Määrittely						
Suunnittelu						
Toteutus						
Integrointi						
Testaus						

Taulukko 1: Gantt-kaavio

Projekti koostui esitutkimuksesta, määrittelystä ja suunnittelusta, jolle varasin kolme kuukautta. Toteutusvaiheelle varasin yhden kuukauden. Integrointi- ja testausvaiheelle varasin kaksi kuukautta. Vaadittavat henkilöresurssit rajoittuvat opinnäytetyön tekijään ja tarvittavat laitteistoresurssit määräytyivät sen mukaan mitä varastosta oli saatavilla. Laitteistoresurssien tarkempi kuvaus oli kolme tavallista pc-tietokonetta, joissa jokaisessa oli yksi kappale 256gb SSD-asema käyttöjärjestelmälle, kaksi 2TB:n kovalevyä hajautettua tallennusjärjestelmää varten ja kaksi verkkokorttia. Hallintaverkolle ja klusteriverkolle oli molemmille omat kytkimet, joihin verkkokortit kytkettiin. Prosessoreina toimi intel i5 ja keskusmuistia oli jokaisessa koneessa 16gb.

Käyttöönottovaihe rajautuu opinnäytetyöstä pois, koska päätöksiä järjestelmän siirtämisestä toimintaympäristöön ei ole vielä tehty. Tämäntapaisissa projekteissa on hankalaa erotella täysin esitutkimusta, määrittelyä ja suunnittelua toisistaan, joten niitä on tärkeämpää katsoa yhtenä kokonaisuutena. Kehittämistyö ei etene täysin lineaarisesti vesiputousmallin mukaisesti vaihe kerrallaan, vaan tarpeen vaatiessa palattiin vaiheissa taaksepäin. Vesiputousmallia voi käyttää myös iteratiivisesti. Vaiheissa jouduttiin palaamaan taaksepäin, kun haluttiin käyttöön lisämäärityksiä. Muutosten ja vaatimusten vaihtuessa aikataulu myös muuttui.

7.1 Esitutkimus

Projektin kehittämiskohteen tunnistuksessa kartoitettiin nykyisen olemassa olevan virtuaalisen palvelinympäristön keskeisimmät ongelmakohdat haastattelun ja havainnoinnin avulla. Haastatteluja oli monia ja ne vastasivat eniten avoimia haastatteluja. Haastateltavana oli yrityksen IT-vastaava. Tietoa voidaan pitää luotettavana, vaikka otanta on pieni ja haastateltavia vain yksi, koska jokainen järjestelmä on omanlaisensa ja havaitut ongelmat useasti todettuja. Haastattelujen ja havainnoinnin avulla tulokseksi saatiin, että nykyinen järjestelmä on RAID-riippuvainen, huonosti skaalautuva ja kustannuksiltaan kallis järjestelmä, joka koostuu monista monimutkaisista komponenteista, jossa virrankatkeaminen ympäristöstä aiheuttaa ongelmia tietojen palauttamisessa tai palauttaminen on hidasta.

Esitutkimuksessa käytettiin haastattelua ja hiljaista tietoa. Hiljaista tietoa kerättiin järjestelmästä vastaavalta henkilöltä tarkkailemalla työn tekemistä ja esittämällä tarkentavia kysymyksiä järjestelmän toiminnasta. Tulokseksi saatiin, että uudesta järjestelmästä halutaan skaalautuva, vikasietoinen, halpa, nykyaikainen järjestelmä. Järjestelmän tulisi suoriutua sille annetuista tehtävistä samalla tai paremmalla tavalla kuin vanha järjestelmä, sen olisi syytä olla helpommin laajennettavissa ja järjestelmä pitäisi voida tarvittaessa kahdentaa. Tarkoituksena oli löytää muutama eri vaihtoehto, jotka täyttävät edellä mainitut vaatimukset. Esitutkimuksen avulla koettiin, että projekti on tarpeellista aloittaa.

7.2 Määrittely

Määrittelyssä käytettiin avuksi haastattelua ja hiljaista tietoa. Hiljaista tietoa ja haastattelua käytettiin hyödyksi samalla tavalla kuin aikaisemminkin eli tarkkailemalla IT-vastaavan työntekoa olan yli ja esittämällä tarkentavia kysymyksiä. Haastattelut olivat avoimia ja niitä oli useita. Haastattelun ja hiljaisen tiedon avulla kartoitettiin mitä uudelta järjestelmältä vaaditaan, mitä se tekee ja listattiin tavoitteet ja vaatimukset sekä määriteltiin, miten sen täytyy toimia vikatilanteessa, miten sitä on käytettävä ja pohdittiin integrointimahdollisuuksia.

Määrittelyssä todettiin, että uuden järjestelmän tarkoituksena on olla HCI-palvelininfrastruktuuri, jonka avulla asiakkaille tarjotaan palveluita. Järjestelmän on oltava erittäin vikasietoinen palvelun jatkuvuuden takaamiseksi. Vikatilanteessa, esimerkiksi virran katketessa, uuden järjestelmän tietojen tulisi säilyä, sen ylläpitäminen olisi halpaa ja se palautuisi omatoimisesti. Järjestelmällä täytyisi pystyä tekemään virtualisointia, sen voisi liittää muihin järjestelmiin ja se avulla voisi yksinkertaistaa infraa.

7.3 Suunnittelu

Suunnittelussa listattiin niitä teknisiä ominaisuuksia, joita uudesta järjestelmästä pitäisi löytyä. Siinä listattiin myös asioita, joita halutaan ottaa ohjelmistosta käyttöön. Näistä saatujen

tulosten pohjalta pidettiin tärkeänä, että ohjelmistosta otetaan käyttöön seuraavat ominaisuudet: korkea saatavuus (HA), klusterointi, varmuuskopiointi, virtualisointi, verkkolevyjako, tiedosto-, lohko- ja objektipohjainen tallennusjärjestelmä.

Suunnitelmassa myös otettiin kantaa käyttöönoton suunnitteluun ja pidettiin tärkeänä, että valituista HCI-ohjelmistoista tehdään testiympäristö. Testiympäristössä testataan miten ohjelmisto suoriutuu sille annetuista tehtävistä. Kriittiset testit liittyvät siihen, miten ympäristö toimii vikatilanteissa, kuten tallennuslaitteen hajoamisessa. Ympäristön on syytä lähettää sähköpostia havaitessaan kriittisen korjausta vaativan vian. Kriittisten vikojen testiin kuuluvat myös seuraavat asiat: Säilyvätkö tiedot eheinä ja palautuuko järjestelmä automaattisesti. Miten hankala on vaihtaa tallennuslaite? Vaatiiko se muita toimenpiteitä palautuakseen, mikä on sen palautumisaika?

Taulukossa 2 on esitettynä Nutanix CE ja Proxmox -ohjelmistojen vähimmäisvaatimukset solmujen ja tallennuksen osalta. Oleellista on huomata, että Nutanix CE vaatii 8GB USB muistin käynnistämiseen. (Nutanix 2020.) Proxmox tarvitsee vähintään 1 kovalevyn. (Proxmox 2020c.)

	solmut	Tallennus
Nutanix	1	200GB SSD & 500GB HDD & 8GB USB per solmu
Proxmox	1	1 x HDD

Taulukko 2: Minimit

Taulukossa 3 on esitettynä Nutanix CE ja Proxmox -ohjelmistojen maksimit solmujen ja tallennustilan osalta. Solmut ovat PC-palvelimia, joihin voidaan asentaa HCI-ohjelmisto. Solmujen maksimimäärä siis määrittää sen, kuinka monta palvelinta voidaan liittää klusteriin. Nutanix CE rajoittuu neljään solmuun. Huomion arvoista on myös Nutanix CE:n maksimi tallennuslaitteiden osalta, joskin jotkut CE:n käyttäjät ilmoittavat, että ovat saaneet toimimaan yli 4 tallennuslaitetta per solmu. (Nutanix 2020.) Proxmox:lla raja tulee oikeastaan vastaan vasta verkkokapasiteetissa, mutta maksimina voidaan pitää 32 solmua. (Proxmox 2020a.)

	Solmut	tallennustila
Nutanix	4	4 per 1 solmu / max 18TB
Proxmox	32, voi olla rajoittamaton, mutta riippuu verkosta.	1GB RAM per 1TB per OSD

Taulukko 3:Maksimit

Suunnittelussa verrattiin opinnäytetyössä esitettyjä ohjelmistoja ja valittiin testiympäristön tekemiseen Proxmox. Vertailu tapahtui siten, että molemmat ohjelmistot asennettiin testiksi testiympäristöön, jotta niiden käyttöä voitiin arvioida kokonaisuutena. Kokonaisuuteen liittyi

asennusprosessi, konfiguraatio, laitteistoresurssit ja käyttö. Eniten ohjelmiston valintaan vaikutti Nutanix CE:n ja Proxmoxin välillä skaalautuvuus, koska Nutanix ei mahdollista kuin neljän solmun ympäristön ilmaisella Community Editionilla. Vertailussa Nutanix CE oli liian rajoitettu. Tähän johtopäätökseen päädyttiin siitä syystä, että tallennustilaa per solmu on mahdollista saada 18TB, kun taas Proxmoxissa ei kyseistä rajoitusta ole. Koimme, että avoimeen lähdekoodiin perustuva ohjelmisto sopii yrityksen tarpeeseen paremmin kuin maksullisen järjestelmän ilmaisversio. Nutanix CE:ssa oli kuitenkin konfiguraatio paljon selkeämpää ja käyttöliittymästä huomasi, että siihen on panostettu.

Nutanix CE osoittautui ongelmalliseksi myös siinä, että ohjelmiston käyttö tapahtuu Prism-tunnusten avulla, joiden kautta järjestelmän käyttöoikeudet. (Nutanix 2020.) Tästä syystä ei voida täysin poissulkea, että ilmaisversioon ei tule käyttöä rajoittavia toimenpiteitä jatkossa enemmän. TrueNAS SCALE tuli opinnäytetyöhön vertailukohdaksi, koska yrityksellä on käytössä TrueNAS Core-ympäristöjä, joissa käyttöliittymä on todettu toimivaksi. Käytettävyyden kannalta SCALE voisi olla hyvä vaihtoehto yritykselle, koska siihen siirtymiseen on pienempi kynnys kuin siirtyminen kokonaan uuteen järjestelmään. SCALE:sta ei ollut vielä valmista versiota ladattavaksi, joten sen testaaminen ei ollut vielä ajankohtaista. Mikäli yritys haluaa avaimet käteen -ratkaisun niin Nutanix on hyvä valinta.

Suunnittelussa päädyttiin siihen, että Proxmoxiin voidaan integroida muita järjestelmiä, sillä voidaan tehdä objekti-, lohko- ja tiedostotallennusta ilman RAID ohjaimia. Suunnittelussa valittiin tallennusjärjestelmäksi Ceph, koska Proxmoxissa on paras käyttöliittymätuki Ceph:iin, jolloin järjestelmän ylläpito on helpompaa ja järjestelmän tilaa voidaan tarkastaa käyttöliittymän kautta visuaalisesti.

7.4 Toteutus

Projektin toteutuksessa esitetyssä aikataulussa ei onnistuttu täysin pysymään, vaan uuden järjestelmän etsiminen on vielä kesken. Tämä johtuu siitä, että TrueNAS SCALE ei ole vielä valmis ja sen vertailu Proxmoxiin nähdään tarpeelliseksi.

Kyseessä on valmiin HCI-ohjelmiston toteutus. Varsinaista ohjelmointityötä ei tarvita. HCI ohjelmisto koostuu kuitenkin useista moduuleista kuten verkko, korkea saatavuus, klusterointi jne. Testiympäristön toteuttamiseksi tehtiin kolmen PC-tietokoneen klusteri, jolloin koneiden resurssit yhdistyivät.

Klusteriin kirjautuminen tapahtui IP-osoitteen avulla selaimen kautta. Ohjelmistolla ei kuitenkaan voida kaikkea konfiguraatiota tai moduulien asennusta tehdä käyttöliittymän kautta vaan osa täytyy suorittaa komentorivin kautta. Toteutuksessa dokumentoitiin tärkeimmät komennot, joilla moduulit otettiin käyttöön. Ceph otettiin käyttöön ja asennettiin OSD levyiksi koneiden 2TB kovalevyt.

Monitorointi asetettiin kaikkiin koneisiin ja hallinta yhteen koneeseen. Ceph onnistui jakamaan lohkopohjaista tallennustilaa virtuaalikoneilla ja CephFS:n avulla luotiin hajautettu tiedostopohjainen tiedostopolku klusterissa oleville koneille, jonne pystyi lataamaan virtuaalikoneiden ISO-tiedostot. ISO-tiedostojen löytyminen tiedostopolun kautta helpotti ja nopeutti virtuaalikoneiden asentamista.

7.5 Integrointi ja testaus

Integrointi- ja testausvaiheessa kokeilimme järjestelmän vikasietoisuutta siten, että koneista katkottiin virrat pois useissa eri tilanteissa. Yksi tällainen katkos tehtiin kesken tiedostojen siirron, jolloin järjestelmän palautumiskykyä tarkasteltiin havaintojen avulla. Järjestelmän palautumista pystyy tarkkailla verkkoselaimen kautta visuaalisesti reaaliajassa. Järjestelmän palautus vikatilasta onnistui automaattisesti, mikäli kaksi konetta kolmesta toimii. Palautumisaika oli alle 4 minuuttia, mutta tulos riippuu käytettävissä olevasta laitteen kokoonpanosta. Virran katketessa korkean saatavuuden ryhmässä olleet virtuaalikoneet siirtyivät toimiviin solmuihin ja käynnistyivät automaattisesti. Järjestelmän kykyä pitää tiedot eheinä ja tallennuslaitteen hajoamista testattiin. Tallennuslaitteen irrottaminen ja liittäminen Ceph:iin oli helppoa käyttöliittymästä käsin. Tiedon eheyttä tarkasteltiin verkkolevyjaon kautta siten, että muista laitteista pääsee käsiksi tietoihin, vaikka yksi tallennuslaite oli epäkunnossa. CephFS:n ja SMB:n avulla pystyi tekemään tiedostopohjaisen verkkolevyjaon Windows tietokoneille, mutta ongelmallista siinä oli, että SMB täytyi asentaa suoraan johonkin Proxmox-koneeseen ja näin ollen sen korkeaa saatavuutta ei voinut taata.

Toinen vaihtoehto toteuttaa tiedostopohjainen verkkolevyjako on virtuaalikoneen kautta, jolloin edellä mainittu ongelma ratkeaa. Kahdentaminen eri paikkakuntien kesken on mahdollista, mutta se vaatii korkeita nopeuksia verkolta, koska dataa liikkuu solmujen kesken merkittävästi. Järjestelmään pystytään liittämään monin eri tavoin kohteita esimerkiksi CIFS:n ja iSCSI:n avulla. Tällä tavalla järjestelmään voidaan integroida nykyisin käytössä olevia VMware- ja TrueNAS-ympäristöjä. Tämä mahdollistaa ohjelmiston laajemman käytön yrityksessä.

Prosessin arvioinnissa todettiin, että Proxmox suoriutuu sille annetuista tehtävistä. Uusimpien päivitysten myötä sen käytettävyys parani käyttöliittymän osalta. Vertailussa käytettiin hyödyksi kolmea viimeisintä versiota. Arvioitaessa järjestelmä vastasi määrittelyssä esiteltyihin vaatimuksiin eli Proxmoxilla saavutetaan riittävä vikasietoisuus, sen skaalautuvuutta voidaan pitää erittäin hyvänä, sillä saavutetaan kustannussäästöjä ja se yksinkertaistaa infrastruktuuria. Testit osoittivat, että testiympäristöstä voidaan siirtyä tuotantoympäristöön.

Järjestelmän käyttöä kriittisten järjestelmien korvaajaksi ei kuitenkaan vielä tällä testausmäärällä voida pitää mahdollisena. Miinuspuolena mainittakoon ohjelmistosta verkosta löyty-

vät hajanaiset, yleistävät ja suppeat dokumentaatiot. Nykyistä järjestelmää voi kuitenkin kehittää siirtymällä tallennusjärjestelmänä hybridi-ratkaisuun, jossa osa tallennuslaitteista korvataan SSD-asemilla. Tämä nopeuttaisi tietojen palauttamista vikatilanteissa, koska flash-muisti ei kadota tietoja samalla tavalla kuin HDD-levyt virran katketessa sekä flash-muistin nopeampi luku- ja kirjoitusnopeus toimii paremmin muuttuvien tietokantojen kanssa.

8 Yhteenveto

HCI-ohjelmistot ja hajautettu tallennusjärjestelmä ovat toimiva tapa toteuttaa virtuaalinen palvelinympäristö ilman RAID-riippuvuutta. Opinnäytetyön tekeminen oli opettavainen matka palvelinympäristön keskeisimpiin käsitteisiin ja tieteelliseen tutkimiseen. Työtä tehtäessä tuli yllätyksenä kuinka laaja-alainen käsite on tallennusjärjestelmä ja kuinka keskeisessä osassa se on tietojärjestelmissä. Työn tarkoitus täyttyi ja varteenotettavia järjestelmiä löytyi, joilla voisi toteuttaa vastaavanlainen ympäristö nykyistä yksinkertaisemmin. Työn reliabiliteetti on pitkälti riippuvainen ohjelmistoversioista ja siitä mihin suuntaan HCI-ohjelmistot ovat siirtymässä. Kuitenkin klusteroinnin ja hajautettujen tallennusjärjestelmien osalta voidaan todeta, että HCI-ohjelmistot toteuttavat yksinkertaisemman, skaalautuvamman ja vikasietoisemman ympäristön kuin CI-järjestelmät.

Käytössä olevan järjestelmän voi korvata Proxmox-ohjelmistolla. Proxmox on ilmainen ohjelmisto, jonka avulla saadaan säästöjä lisenssimaksuista. Sillä saavutetaan erittäin vikasietoinen järjestelmä, koska sen avulla voidaan hajauttaa järjestelmä tallennuksen ja laitteistojen osalta. Proxmox tarjoaa helposti laajennettavan järjestelmän, johon voidaan liittää ainakin 32 laitetta ja tallennuskapasiteetti määräytyy käytettävissä olevan keskusmuistin mukaan.

Lähteet

Painetut

Haikala, I. & Märijärvi, J. 2002. Ohjelmistotuotanto, 8. Uudistettu painos. Helsinki: Talentum Media Oy.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki: Talentum

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2004. Tutki ja kirjoita. Helsinki: Tammi

Vilkka, H. 2006. Tutki ja havainnoi. Helsinki: Tammi

Sähköiset

Azheem, S. & Sharma, S. 2017. Study of Converged Infrastructure & Hyper Converge Infrastructure As Future of Data Centre. Viitattu 11.10.2020. https://www.researchgate.net/profile/Shaikh_Azeem/publication/335910358_Study_of_Converged_Infrastructure_Hyper_Converge_Infrastructre_As_Future_of_Data_Centre/links/5d83379ca6fdcc8fd6f3c7fa/Study-of-Converged-Infrastructure-Hyper-Converge-Infrastructre-As-Future-of-Data-Centre.pdf

Ceph 2020b. Architecture. Viitattu 11.10.2020. https://docs.ceph.com/en/latest/architecture/

Ceph 2020. Crush Map. Viitattu 11.10.2020. https://docs.ceph.com/en/latest/rados/operations/crush-map/

Ceph 2014. Zero to hero guide :: For Ceph Cluster Planning. Viitattu 11.10.2020. https://ceph.com/geen-catego-rie/zero-to-hero-guide-for-ceph-cluster-planning

Cheng, S. 2014. Proxmox High Availability. E-kirja. Birmingham: Packt Publishing Ltd.

Eskola, J. & Suoranta, J. 1998. Johdatus laadulliseen tutkimukseen. E-kirja. Tampere: Vastapaino.

Golden, B. 2011. Virtualization For Dummies. E-kirja. Indianapolis: Wiley Publishing, Inc.

Helixstorm 2020. Hyperconverged Infrastructure. Viitattu 26.10.2020. https://www.helixstorm.com/blog/hyperconverged-infrastructure/

IBM 2019. Block Storage. Viitattu 28.10.2020. https://www.ibm.com/cloud/learn/block-storage

iXsystem 2020. TrueNAS SCALE. Viitattu 20.10.2020. https://www.truenas.com/truenas-scale/

Jalonen, H 2014. Mitä hiljainen tieto on ja voiko sitä johtaa? Viitattu 1.11.2020. https://www.researchgate.net/publication/268744450_Mita_hiljainen_tieto_on_ja_voiko_sita_johtaa

Mellor, C. 2019. Nutanix kick the Buckets into object storage. Viitattu 31.1.2021. https://blocksandfiles.com/2019/11/07/nutanix-objects-storage-service/

Moore, K. 2020. Starting our next Open Source Project- TrueNAS SCALE. Viitattu 20.10.2020. https://www.ixsystems.com/community/threads/starting-our-next-open-source-project-truenas-scale.85203/

Mulford, J. 2020. iXsystem TrueNAS SCALE revealed. Viitattu 20.10.2020. https://www.storagereview.com/news/ixsystems-truenas-scale-revealed

Mäntyneva, M. 2016. Hallittu projekti. E-kirja. Helsinki: Helsingin seudun kauppakamari.

Nutanix 2020. Nutanix Data Sheet. Viitattu 19.10.2020. http://go.nutanix.com/rs/nutanix/images/Nutanix_Datasheet_Standard.pdf

Nutanix 2020. Getting Started With Nutanix Community Edition. Viitattu 19.10.2020. https://portal.nutanix.com/page/documents/details?targetId=Nutanix-Community-Edition-Getting-Started:Nutanix-Community-Edition-Getting-Started

Ojasalo, K., Moilanen, T. & Ritalahti, J. 2015. Kehittämistyön menetelmät. 4. painos. E-kirja. Helsinki: Sanoma Pro.

Pohjalainen, M. 2012. Hiljaisen tiedon käsite ja hiljaisen tiedon tutkimus. Viitattu 21.1.2021. https://journal.fi/inf/article/view/7079/5613

Poitras, S. 2020. The Nutanix Bible. Viitattu 19.10.2020. https://nutanixbible.com/

Portnoy, M. 2012. Virtualization Essentials. E-kirja. Indianapolis: John Wiley & Sons, Inc.

Proxmox 2020a. Cluster Manager. Viitattu 25.1.2021. https://pve.proxmox.com/wiki/Cluster_Manager

Proxmox 2020b. Pricing. Viitattu 16.10.2020. https://www.proxmox.com/en/proxmox-ve/pricing

Proxmox 2020. Proxmox VE Administration Guide. Viitattu 16.10.2020. https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf

Proxmox 2020c. System Requirements. Viitattu 25.1.2021. https://pve.proxmox.com/wiki/System_Requirements

Rathnam, L. 2020. Raid levels explained: How they can benefit your business. Viitattu 11.10.2020. http://techgenix.com/raid-levels-explained/

Red Hat 2020. File storage, block storage, or object storage?. Viitattu 28.10.2020. https://www.redhat.com/en/topics/data-storage/file-block-object-storage

Rouse, M. 2016. Ceph. Viitattu 11.10.2020. https://searchstorage.techtarget.com/definition/Ceph

Rouse, M. 2018. Storage. Viitattu 28.10.2020. https://searchstorage.techtarget.com/definition/storage

Rouse, M. 2017. ZFS. Viitattu 12.10.2020. https://searchstorage.techtarget.com/definition/ZFS

Techopedia 2017. Viitattu 20.10.2020. Virtualization. https://www.techopedia.com/definition/719/virtualization

Kuviot

Taulukot