Tampereen yliopisto

Kalle Puutio

KONTTITEKNOLOGIAN EDUT JA HYÖDYT

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Elokuu 2021

TIIVISTELMÄ

Kalle Puutio: Konttiteknologian edut ja hyödyt
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Elokuu 2021

Viimeisen viiden vuoden aikana virtuaalikoneiden (Virtual Machine, VM) rinnalle on kehittynyt konttiteknologia (container technology), joka tarjoaa kevyen, liikuteltavan ja eristetyn vaihtoehdon virtuaalikoneille. Konttiteknologia voidaan nähdä myös virtualisoinnin luontaisena teknologian kehitysasteena. Uutena teknologiana sen etuja ei kuitenkaan vielä välttämättä tunneta laajalti. Tämän tutkielman tarkoitus on tunnistaa konttiteknologian edut, verrata konttiteknologian hyödyntämistä virtuaalikoneisiin sekä muihin kehitysmenetelmiin ja tutkia, mitä etuja konttiteknologia tarjoaa.

Tämä tutkielma on muodoltaan kirjallisuuskatsaus. Tutkielman aineistoa on etsitty tietojenkäsittelytieteisiin keskittyneistä julkaisutietokannoista. Aiheen tuoreuden vuoksi artikkelien valintakriteereissä on painotettua mahdollisimman uusia julkaisuja, mutta taustatietojen, kuten virtualisoinnin ja korkeateholaskennan (High Performance Computing), tarkastelussa on mukaan otettu myös vanhempia artikkeleita tuomaan laajempaa kontekstia tutkielman aiheelle.

Jotta konttiteknologian edut voidaan tunnistaa, täytyy tuntea sovelluskehityksen historiaa sekä keinot, jotka olivat käytössä ennen virtualisointia ja konttiteknologiaa. Tämä tutkielma sitoo aiheensa myös isompaan teknologian kehityksen kokonaisuuteen tuodakseen lukijalle kokonaiskuvan aiheen kontekstista ja teknologian kehityksestä pelkkää yksittäistä teknologiaa suurempana kokonaisuutena, vaikka monessa kohdassa keskitytäänkin Docker-kontteihin niiden suosion takia.

Kontit ovat suorituskyvyltään yhtä tehokkaita tai tehokkaampia kuin käyttöjärjestelmän ytimeen pohjautuvat virtuaalikoneet ja tarvitsevat virtuaalikoneisiin verrattuna vähemmän muistia, mikä samalla myös nopeuttaa konttien käyttöönottoa pilviympäristöissä.

Konttiteknologian etuja ovat keveys, vähäisempi lisärasite (overhead) verrattuna virtuaalikoneisiin sekä muut virtualisointiteknologialle ominaiset edut, eli osittelu (partitioning), eristäminen (isolation) sekä kapselointi (encapsulation).

Avainsanat: konttiteknologia, kontit, virtualisointi, kontitus

Sisällysluettelo

1 Johdanto

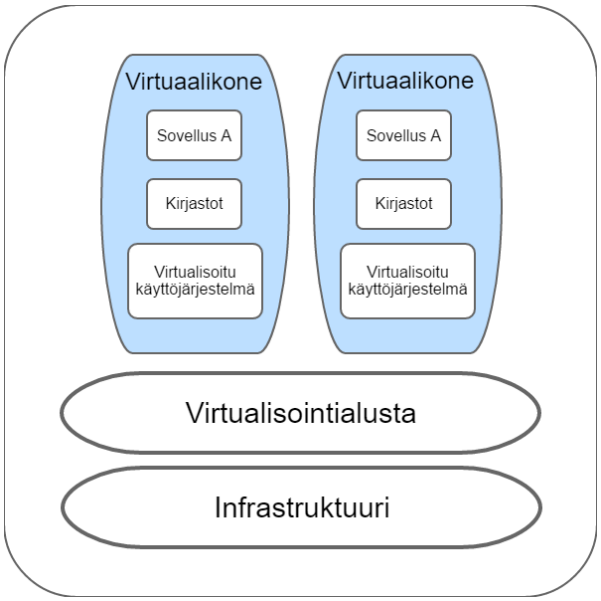
Ennen *virtualisointia* (*virtualization*) yksilöt ja organisaatiot ovat joutuneet rakentamaan omistamiinsa tiloihinsa isoja, tilaa vaativia, *suurteholaskentajärjestelmiä* (*High Performance Computing*). Perinteiset *suurteholaskentajärjestelmät* koostuvat useasta *dedikoidusta palvelimesta* (*dedicated server*). Näitä palvelimia on mahdollista jakaa usean eri organisaation tai käyttäjän välillä. Organisaatioilla tai käyttäjille voi olla erilaisia vaatimuksia palvelinympäristöille, mikä käytännössä tarkoittaa, että jokaiselle käyttäjälle tai organisaatiolle täytyy erikseen räätälöidä oma ympäristö. Haasteena on, että järjestelmät täytyy aina räätälöidä yksilön tai organisaation kunkin käyttötarkoituksen mukaisiin tarpeisiin ja tämä räätälöinti on kallista sekä resursseja, kuten työvoimaa ja aikaa, vievää. Ratkaisuksi tähän on otettu käyttöön *virtualisointi*, joka luo erillisen, käyttäjälle tai organisaatiolle räätälöidyn, virtuaalisen ympäristön vastaamaan käyttäjän tarpeita (Martin et al., 2018).

Virtualisointi kehitettiin mahdollistamaan järjestelmän räätälöinti ilman tarvetta räätälöidä fyysistä järjestelmää, jolloin jokaiselle käyttäjälle voidaan räätälöidä käyttäjän vaatimusten mukainen virtuaalinen järjestelmä ilman tarvetta tehdä fyysistä järjestelmää. Verrattuna perinteiseen suurteholaskentajärjestelmään virtualisointi tarjoaa mm. paremman resurssien hyödyntämisen, helpomman järjestelmän ylläpidon, paremman järjestelmän siirrettävyyden, korkeamman saatavuuden ja paremman palautumisen (Pooja ja Pandey, 2014). On mahdollista virtualisoida useita eri asioita, kuten laitteistoa, ohjelmistoa, työpöytää, tallennusjärjestelmää tai dataa (Pooja ja Pandey, 2014).

Perinteisien suurteholaskentajärjestelmien vaihtoehdoksi on kehittynyt *pilvipalvelu* (*cloud computing*). Pilvipalvelu on etänä toimitettava palvelu, resurssi tai työkalu, jonka pilvipalveluntarjoaja tarjoaa käyttäjälle. Merkittävänä etuna nähdään, ettei käyttäjän tarvitse rakentaa omaa IT-infrastruktuuriaan vaan resurssit voidaan keskittää käyttäjän omaan ydinosaamiseen. Verrattuna suurteholaskentaan pilvipalvelu tarjoaa suurlaskentatehoa mahdollisesti matalampaan hintaan. Pilvipalvelussa virtualisointia käytetään mahdollistamaan laaja-alainen jaettujen resurssien jako. Tyypillisesti *virtuaalikoneet* (*virtual machine*) muodostavat pilvipalvelun infrastruktuurin selkärangan. (Pahl et al., 2019)

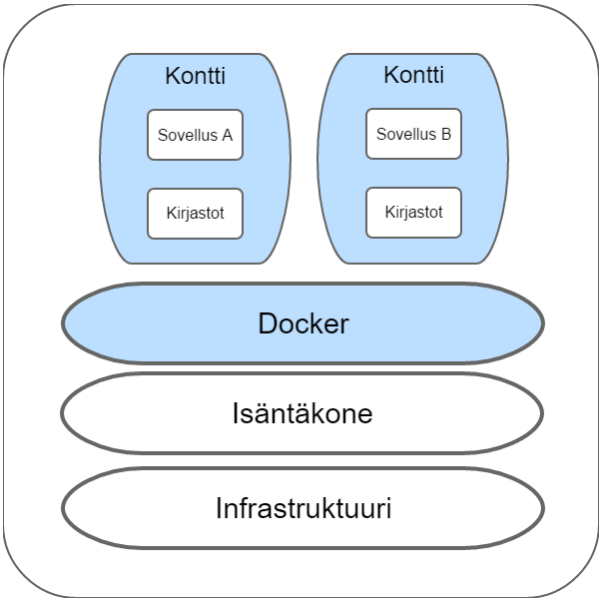
Virtuaalikone on simulaattori, joka tarjoaa käyttöympäristön määriteltävään tarkoitukseen. Virtuaalikone voi sisältää prosessorin, muistin sekä I/O-laitteet (Nguyen et al., 2017). *Virtualisointialusta* (*Hypervisor*) on ohjelmisto, joka luo ja ajaa virtuaalikoneita. Joskus *virtuaalikoneiden valvontana* (*virtual machine management*) tunnettu virtuali-

sointialusta eristää käyttöjärjestelmän ja resurssit virtuaalikoneesta ja mahdollistaa virtuaalikoneiden hallinnoinnin (Red Hat, 2021). Esimerkki virtuaalikoneen kerroksista ja arkkitehtuurista on kuvassa 1. Virtuaalikone tarjoaa luonnostaan eristäytyneisyyttä, sillä ainoa tapa virtuaalikoneelle kommunikoida ulkopuolisen maailman kanssa on joko rajattujen *hyperkutsujen* (*hypercall*) tai emuloitujen laitteiden kautta, joita molempia hallitsee *virtualisointialusta* (Felter et al., 2014). Yksi suurimpia etuja virtuaalikoneissa on mahdollisuus jakaa fyysisen koneen suuret resurssit pienempiin, käyttäjälle määriteltyihin tarpeisiin. Yhden virtuaalikoneen käyttövalmiiksi asentaminen on fyysiseen koneeseen verrattuna nopeampaa, halvempaa sekä helpompaa (Nguyen et al., 2017).



Kuva 1. Virtuaalikoneiden arkkitehtuuri ja kerrokset.

Kontitus (*containerization*) on kevyt virtualisointiteknologia, joka mahdollistaa hajautettujen järjestelmien käyttöönoton ja ajamisen pilvessä (Casalicchio ja Ianucci, 2019). Konteissa ohjelma eristetään omaan suljettuun ympäristöönsä tarvittavine kirjastoineen, ohjelmistoineen ja työkaluineen, josta ohjelma voi kommunikoida rajapintojen kautta muiden ohjelmien kanssa (kuva 2). Ohjelmistokehityksen näkökulmasta konttiteknologia tarjoaa merkittäviä etuja, sillä ohjelmointikielet, työkalut ja kirjastot voidaan valita kunkin sovelluksen tai työtehtävän tarpeiden mukaan. Eristyneisyyden myötä kehittäjä voi keskittyä vain oman sovelluksensa itsenäiseen kehittämiseen.



Kuva 2. Docker-konttien arkkitehtuuri ja kerrokset.

Yleisesti konttiteknologialla ajatellaan olevan etuja kuten keveys, suorituskyky ja tehokkuus. Konttiteknologian suorituskykyä onkin tutkittu ja verrattu vaihtoehtoisiin teknologioihin, kuten virtuaalikoneisiin. Tämän tutkielman tutkimuskysymys on ”*Mitä on konttiteknologia ja mitä etuja tai hyötyjä siinä on*?” eli tarkoituksena on tunnistaa konttiteknologian edut, mutta samalla myös tuoda esille mahdollisia konttiteknologian puutteita tai haasteita.

Tutkielman luvussa 2 kuvataan tutkimusmenetelmä, jolla tutkielman aineisto on valittu. Luvussa 3 käydään läpi konttiteknologian tarkempaa sisältöä, eri konttityyppejä, konttien käyttämistä osana kehitysprosessia sekä avataan konttien hallinnointia ja siinä käytettyjä ohjelmistoja. Luvussa 4 käydään läpi konttiteknologian edut ja hyödyt sekä tutkitaan tarkemmin, miten nämä edut ilmenevät. Luvussa 5 tuodaan esille konttiteknologian haastekohdat ja puutteet. Luvussa 6 pohditaan konttiteknologian etuja, etujen määritelmiä ja tulkintaa sekä pohditaan konttien hallinnointia ja soveltamista yritysmaailmassa. Luvussa 7 on lopuksi tutkielman yhteenveto sekä lista tutkielmassa esille tulleista eduista ja hyödyistä.

2 Tutkimusmenetelmä

Valintakriteerinä tässä katsauksessa käyttämissäni artikkeleissa oli keskittyminen konttiteknologiaan, sen suorituskykyyn ja sovelluksiin. Tämän kirjallisuuskatsauksen lähdemateriaalina käyttämäni artikkelit etsin IEEE XPLORE, Computer Science Database Proquest ja Elsevier – tietokannoista. Käyttämiäni hakusanoja olivat ”Golang”, ”Docker”, ”Container technology”, ”Docker container”, ”Virtualization”, ”Performance”, ”Container Security” ja ”Virtual Machine”. Pyrin rajaamaan lähteet 2005 vuodesta eteenpäin ja

suosimaan mahdollisimman ajankohtaisia lähteitä, sillä teknologia kehittyy jatkuvasti ja on saattanut muuttua artikkelin julkaisun ja tämän tutkielman kirjoittamisen välisenä aikana.

Kaikki käyttämäni lähdemateriaalit ovat yhtä mieltä: konttiteknologialla on etuja verrattuna virtuaalikoneisiin. Useissa lähteissä ei kuitenkaan usein suoraan mainittu tai määritelty, mitä konttiteknologian edut ovat; usein vaan tyydyttiin toteamaan, että hyötyjä, kuten liikuteltavuus ja parempi resurssien hallinta, on olemassa. Tavoitteenani oli siis löytää aineistoa, jossa tutkitaan, mitä nämä hyödyt käytännössä ovat, miten nämä hyödyt määritellään ja voidaanko konttiteknologian suorituskykyä verrata muihin vastaaviin teknologioihin.

3 Konttiteknologia

3.1 Kontin sisältö

Yksittäinen kontti pitää sisällään eristetyt, muusta ympäristöstä riippumattomat ja käyttöönottovalmiit sovelluksen osat sekä mahdollisesti ajamiseen tarvittavat *väliohjelmistot* (*middleware*) ja *käyttölogiikan* (*business logic*), kuten binäärit ja kirjastot (Pahl et al., 2019). Docker-kontit hyödyntävät LXC:ia (*Linux containers*), jotka koostuvat *Linux-ytimen* (*Linux kernel*) ominaisuuksista, kuten *hallintoryhmistä* (*cgroups*) ja *nimiavaruuksista* (*namespaces*) tehokkaaseen prosessin eristämiseen sekä resurssinhallintaan (Biradar et al., 2018). Verrattuna perinteiseen virtualisointiin konttien etuina ovat myös pieni koko, ohjelman siirto järjestelmästä toiseen on nopeampi, resurssien kulutus on vähäisempää ja resurssien hyödyntämisen aste korkeampi (Xie et al., 2017).

3.2 Järjestelmä- ja sovelluskontit

Kontit voidaan jaotella kahteen pääluokkaan: *järjestelmäkontteihin* (*system container*) sekä *sovelluskontteihin* (*application container*). Järjestelmäkontit pitävät sisällään kokonaisen käyttöjärjestelmän siinä missä ohjelmistokontti sisältää vain sovelluksen tai osan sovellusta. Sovelluskontti koostuu useasta kerroksesta *näköistiedostoja* (*image layer*), jotka hyödyntävät Union File System -palvelua sekä Copy-on-Write -tekniikkaa, joilla *tiedostojärjestelmä* (*file system*) saadaan näyttämään *kirjoitettavalta* (*writable*) ilman, että kirjoittaminen muuttaa tiedostojärjestelmän sisältöä. (Casalicchio ja Ianucci, 2019). Tyypillisesti sovelluskehittäjä aloittaa kehittämisen *pohjanäköistiedostosta* (*base image*), joka sisältää käyttöjärjestelmän ytimen sekä käyttöjärjestelmän vakiokirjastot. Jälkeenpäin luotu sovellus voidaan kopioida uudessa kerroksessa.

Kuten sovelluskontissa, järjestelmäkontissakin hyödynnetään käyttöjärjestelmän näköistiedostoa alimpana kerroksena, jota on mahdollista muokata lisäämällä kirjastoja, työkaluja ja tietoa uusissa kerroksissa. Kun monikerroksinen näköistiedosto on valmis, kaikki paitsi viimeisenä lisätty kerros muutetaan vain luettaviksi (*read-only*). Viimeinen lisätty kerros on luettava ja kirjoitettava, muttei persistoiva (*persistent*), eli kun kontti poistetaan niin tämä viimeisen kerroksen data menetetään (Casalicchio ja Ianucci, 2019).

3.3 Kehitysprosessi kontteja käyttäessä

Tyypillisesti sovelluskehittäjä aloittaa kehittämisen pohjanäköistiedostosta, joka sisältää käyttöjärjestelmän ytimen sekä käyttöjärjestelmän vakiokirjastot (Casalicchio, Ianucci, 2019). Tämän jälkeen kehittäjä lisää mahdollisesti sovelluksen tai järjestelmän tarvitsemia kirjastoja konttiin. On myös mahdollista, että kehittäjä hyödyntää *monivaiheista rakentamista* (*multi-stage build*), jossa sovellus käännetään ensin *rakennusnäköistiedostossa* (*builder image*) ja siirretään myöhemmin varsinaiseen, käyttöönotettavaan näköistiedostoon.

Etuna ylläkuvatussa monivaiheisessa rakentamisessa on, että kääntämisessä tarvittavat kirjastot, työkalut ja muut ylimääräiset, ajamiseen tarpeettomat tiedostot voidaan jättää pois lopullisesta, käyttöön otettavasta kontista, jolloin näköistiedoston koko saadaan pidettyä mahdollisimman pienenä.

3.4 Konttien hallinnointi

Konttien käyttöönottoon sekä elinkaaren hallinnointiin käytetään erilaisia *kehysympäristöjä* (*framework*). Näillä kehysympäristöillä hallinnoidaan konttien tarvitsemia resursseja ohjelmisto- ja laitteistotasolla. Kontit tarjoavat sekä fyysisien resurssien, eli prosessorin, muistin jne., että käyttöjärjestelmätason (prosessorin ajoituksen, vuorottelun) resurssit. Hallinnointikehysympäristöjen tärkein ominaisuus on kyky käynnistää sovelluksia mahdollisimman alhaisella viiveellä. Tämän lisäksi hallinnointikehysympäristöt määrittelevät, miten kontit käynnistetään, milloin ja millä ehdoilla käynnistäminen, alasajo sekä uudelleenkäynnistäminen suoritetaan, sekä mihin kontit asetetaan ajoon. (Watada et al., 2019)

3.5 Yleisimpiä konttiteknologioita

Vaikka monia konttiteknologioita on ollut olemassa jo pidemmän aikaa, käytännössä Docker nosti konttiteknologian yleiseen suosioon, erityisesti pilvipalveluissa (Abraham et al., 2020). Tämän hetken suosituin kontitusratkaisu onkin Docker (Lin et al., 2020).

Taulukossa 1 on listattu erilaisia konttiteknologioita. Mukana ovat järjestelmä- ja sovelluskonttiteknologioiden lisäksi konttien hallinnointiin tarkoitetut teknologiat ja konttien

orkestraation kehysympäristöt. Teknologiat, jotka ovat tyypiltään sovellus- tai järjestelmäkontteja, ovat tarkoitettu itse konttien luomiseen. Konttien hallinnointiteknologiat ovat tarkoitettu avustamaan ja automatisoimaan konttien hallinnointia koko konttien elinkaaren ajan. Konttien elinkaaren hallinnointiin kuuluu työtehtäviä kuten konttien käyttöönotto, konttien käynnistymisen ja alasajon ajastaminen, resurssien varaaminen, kontin tilan tarkkailu ja skaalauksen hallinta.

Listaa ei ole tarkoitettu kaiken kattavaksi ja täydelliseksi, vaan antamaan esimerkkejä yleisesti käytössä olevista teknologioista. Listaan on pyritty valitsemaan tällä hetkellä käytössä olevia ja aktiivisesti tuettuja teknologioita. Hylätyt tai teknologiat, joiden tuki on lakkautettu, on pyritty jättämään pois.

Taulukko 1. Lista yleisimmistä konttiteknologioista.

Teknologia	Tyyppi	Kotisivu
Amazon Elastic Container Service (ECS)	Konttien hallinnointi	https://aws.amazon.com/ecs/
Azure Kubernetes Service (AKS)	Konttien hallinnointi	https://azure.microsoft.com/en-us/services/kubernetes-service/
Cloudify	Konttien hallinnointi	https://cloudify.co/
Docker	Sovelluskontti, konttien hallinnointi	https://www.docker.com/
Docker Swarm	Konttien hallinnointi	https://www.docker.com/
Kubernetes	Konttien hallinnointi	https://kubernetes.io/
Linux Container (LXC)	Järjestelmä- ja sovelluskontti	https://linuxcontainers.org
LXD	Konttien hallinnointi	https://linuxcontainers.org/
Mesosphere Marathon	Konttien hallinnointi	https://mesosphere.github.io/marathon/
OpenVZ	Järjestelmäkontti, konttien hallinnointi	https://openvz.org
Oracle Solaris Container	Järjestelmä- ja sovelluskontti, konttien hallinnointi	https://www.oracle.com/solaris/technologies/solaris-containers.html
Windows Hyper-V Container (WHC)	Sovelluskontti	https://docs.microsoft.com/en-us/virtualization/windowscontainers
Windows Server Container (WSC)	Sovelluskontti	https://docs.microsoft.com/en-us/virtualization/windowscontainers/

4 Konttiteknologian edut

4.1 Konttiteknologialle ja virtualisoinnille yhteiset edut

Virtuaalikoneet virtualisoivat laitteiston abstraktilla tasolla, mutta kontit ovat emulaatio käyttöjärjestelmän abstraktilla tasolla (Chung et al., 2016). Virtuaalikoneissa siis pyritään virtualisoimaan tietokoneen täysi laitteisto, kun taas konteissa virtualisoidaan vain käyttötarkoitukseen tarvittavan laitteiston osa. Seuraavat edut ovat yleisiä virtualisoinnin etuja, jotka täten ovat myös konttiteknologiaan kuuluvia etuja, vaikka eivät suoranaisesti ole konttiteknologiasta lähtöisin olevia etuja.

4.1.1 Osittelu

Virtualisoinnissa *osittelu* (*partitioning*) tarkoittaa käytössä olevien resurssien, kuten prosessorin tai muistin, jakamista niin, että useita ohjelmistoja ja käyttöjärjestelmiä voidaan ajaa samanaikaisesti.

4.1.2 Eristäminen

Eristäminen (*isolation*) on virtualisoinnin ominaisuus, jossa yhden virtuaalikoneen kaatuessa kaatuminen ei vaikuta muihin virtuaalikoneisiin, sillä jokainen virtuaalikone on eristetty fyysisestä *isäntäkoneesta* (*host*) sekä muista virtuaalikoneista (Pooja, 2014). Yhden kontin kaatuminen ei vaikuta muihin kontteihin, ellei kontteja ole tehty toisistaan riippuvaisiksi.

4.1.3 Kapselointi

Kapseloinnilla (*encapsulation*) tarkoitetaan sitä, että virtuaalikone voidaan esittää ja tallettaa yhtenä tiedostona, jotta se voidaan helposti tunnistaa tarjoamiensa palveluiden perusteella. Kapseloitu virtuaalikone voidaan esittää ohjelmistolle yhtenäisenä, kokonaisena entiteettinä. Täten kapselointi estää virtuaalikoneita häiritsemästä toistensa toimintaa (Pooja, 2014). Myös kontit voidaan esittää yhtenä, kokonaisena näköistiedostona.

4.2 Lisärasite

Docker-kontit ovat suorituskyvyltään joko yhtä tehokkaita tai tehokkaampia kuin käyttöjärjestelmän ytimeen pohjautuvat virtuaalikoneet (*Kernel-based Virtual Machine*). On huomioitava, että vaikka kontit ja virtuaalikoneet eivät aiheuta melkein yhtään *lisärasitetta* (*overhead*) suoraan prosessorille tai muistin käytölle, kontit ja virtuaalikoneet aiheuttavat lisärasitetta *siirräntään* (*I/O*) sekä vuorovaikutukseen käyttöjärjestelmän kanssa. Jokainen siirräntäoperaatio vaatii ylimääräisiä prosessorin käskysyklejä, joten pienet siirrännät kärsivät enemmän kuin isot. Tästä aiheutuva ylimääräinen lisärasite lisää siirräntöjen *viivettä* (*latency*) ja vaatii prosessorin käskysyklejä (Felter et al., 2014, Kozhirbayev ja Sinnot, 2017). Toisin sanoen, kontit ja virtuaalikoneet tuovat lisärasitetta

siirräntäoperaatioiden kautta epäsuorasti prosessorille. Konttiteknologian käytön tuoma lisärasite on lähes olematon, mutta siirräntäoperaatiot ovat suurin pullonkaula suorituskyvyssä (Morabito et al., 2015).

4.3 Muistin käyttö

Koska sovelluskontti ei pidä yllä ylimääräisiä hallintoprosesseja, sovelluskontti ei tuhlaa *keskusmuistia* (*Random Access Memory*) tarpeettomien hallintaprosessien ylläpitoon ja täten käyttää vähemmän keskusmustia kuin vastaava järjestelmäkontti tai virtuaalikone (Felter et al., 2014).

4.4 Käynnistymisnopeus

Usein Docker-näköistiedostot käyttävät vähemmän kovalevytilaa ja tiedonsiirtoa kuin vastaavat virtuaalikoneet. Tämä mahdollistaa nopeamman käyttöönoton pilviympäristössä, jossa tiedostot täytyy kopioida paikalliselta levyltä pilvipalveluun. Felter ja muut (2014) toteavat tutkimuksessaan, että heidän koeympäristössään kontti voi käynnistyä jopa alle yhdessä sekunnissa, siinä missä virtuaalikone vaati 11 sekuntia. Pääasiallinen syy tähän nopeuteen on, että päinvastoin kuin virtuaalikoneiden, konttien ei tarvitse käynnistää kopioita käyttöjärjestelmästä.

4.5 Yhtenäinen kehityspohja hajautetuissa järjestelmissä

Näköistiedostojen (*image*) koostuminen kerroksista mahdollistaa yhtenäisen ja helposti liikuteltavan kehityspohjan käyttämisen yrityksen sisällä. Pohjana käytetyn näköistiedoston ominaisuuksia voidaan laajentaa yhtenäisellä ja ylläpidettävällä tavalla (McGee, 2016). Yksittäistä näköistiedostoa voidaan käyttää usean kontin pohjana, sallien kuitenkin jokaiselle kontille mahdollisuuden muokata käyttämiään ohjelmistoja ja tiedostoja käyttötarpeiden mukaan (Felter et al., 2014).

4.6 Turvallisuus

Konteilla ei ole pääsyä mihinkään, mitä ne eivät voi nähdä, joten potentiaali mahdollisesti liian laajoille käyttöoikeuksille on alhaisempi. Käyttäessä *Linux-nimiavaruuksia* (*Linux namespace*) kontissa olevaa *juurikäyttäjää* (*root user*) ei nähdä kontin ulkopuolella juurikäyttäjänä, mikä tarjoaa lisää turvallisuutta (Bui, 2015). Turvallisuuden ongelmakohdista lisää kohdassa 5.2.

5 Konttiteknologian haastekohdat, ongelmat ja puutteet

5.1 Konttien siirtäminen käyttöjärjestelmästä toiseen

Konttiteknologiassa on joitakin rajoituksia. Konttien *siirtäminen* (*porting*) käyttöjärjestelmäperheestä toiseen, esimerkiksi Linuxista macOS:ään, on monimutkaisempaa kuin

virtuaalikoneiden vastaava siirtäminen. Ei ole natiivia tapaa ottaa kontitettuja sovelluksia käyttöön macOS:llä, vaan tällöin täytyy käyttää virtuaalikonetta kontitetun sovelluksen ja käyttöjärjestelmän välissä (Watada et al., 2019).

Windowsiin pohjautuvia kontteja ei ole mahdollista ajaa Linuxiin pohjautuvassa isäntälaitteessa (Morabito et al., 2015). Sen sijaan on mahdollista ajaa Linux-kontteja Windowsilla hyödyntäen Dockeria, Hyper-V:tä tai WSL:ia.

5.2 Turvallisuuden haasteet

Kontit eivät välttämättä eristä resurssejaan yhtä hyvin kuin virtualisointialustat, sillä isäntälaitteen käyttöjärjestelmän ydin on paljastettu konteille. Tämä voi olla ongelma tilanteissa, joissa yksi palvelin palvelee useaa käyttäjää (konttia). Konttien pääasiallisin turvallisuushaavoittuvaisuus on järjestelmäkutsut, jotka eivät ole tietoisia nimiavaruuksista ja täten mahdollistavat vuodon konttien välillä (Felter et al., 2014). Jos kontti ajetaan täysillä oikeuksilla, on tilanne lähes sama kuin prosesseilla, jotka ajettaisiin natiivisti isäntälaitteessa. Turvallisuutta voidaan lisätä ajamalla kontteja ilman korkeampia oikeuksia sekä käyttämällä lisäksi Linuxin ytimen turvamoduuleja, kuten AppArmor ja SELinux (Bui, 2015).

6 Keskustelu

Moni tutkimus ja artikkeli mainitsee ohimennen joitakin konttiteknologian etuja, mutta yksikään ei ole, parhaimman tietoni mukaan, koostanut ja määritellyt näitä konttiteknologian etuja. Usein tutkimuksissa eduista mainitaan vain ne edut, jotka ovat kyseisen tutkimuksen tutkimuskysymyksen kannalta olennaisia.

Konttiteknologian eduista mainitaan usein keveys, mutta harvemmin tuodaan esille, mitä keveys tässä kontekstissa tarkoittaa, ellei se ole osa tutkimuksen tutkimuskysymystä tai liity olennaisesti tutkimuksen tutkimuskysymykseen. Joskus konttien keveydellä tarkoitetaan konttien käyttämien näköistiedostojen pientä kokoa, joskus konttien ajamisen vähäistä lisärasitetta, joskus yhdistelmää näistä kahdesta edellä mainitusta. Keveys on tulkinnanvarainen termi, ja tulisi aina määritellä, jotta sen arviointi olisi mahdollista.

Konttien ajatellaan olevan lisärasitteeltaan alhaisia. Yllättävää kuitenkin on, että konttien siirräntäoperaatiot sekä järjestelmäkutsut nostavat lisärasitetta epäsuorasti. Erityisesti suuri määrä siirräntäoperaatioita nostaa lisärasitteen määrää. Täten on mahdollista ajatella, että lukumäärältään vähäisiä, mutta tiedostokooltaan suuria siirräntäoperaatioita suorittavat kontitetut sovellukset voivat olla lisärasitteeltaan alhaisempia kuin useita, mutta tiedostokooltaan pieniä siirräntäoperaatioita suorittavat kontitetut sovellukset. Kontitettuja sovelluksia kehittäessä on suositeltavaa huomioida siirräntäoperaatioiden

vaikutus konttien resurssivaatimukseen, mikäli kyseessä on käyttötarkoitus, jossa korkea suorituskyky on olennaista.

Monet yritykset ovat havainneet konttiteknologian edut ja hyödyntävät kontitusta kasvavissa määrin. Sovelluskehityksen näkökulmasta konttien ominaisuudet mahdollistavat ohjelmistojen helpon skaalauksen, sillä konttien lisääminen tai poistaminen ympäristöstä on nopeaa. Myös perinteiset, suuret ja monimutkaiset sovellusmonoliitit on mahdollista jakaa pienempiin ja helpommin hallittaviin modulaarisiin *mikropalveluihin* (*microservice*).

Konttien hallinnointi on yksi suurimpia haasteita konttien käyttöönotossa yrityksissä. Kehitysprosessi kontteja käyttäessä eroaa huomattavasti virtuaalikoneiden käytöstä ja täten voi vaatia sopeutumista erilaiseen kehitystoimintaan. Yrityksen tai organisaation, joka on tottunut virtuaalikoneiden käyttämiseen järjestelmänsä tai sovelluksensa käyttöönotossa, olisi hyvä tutustua konttiteknologian kehittämiseen ja käyttöönottoon liittyviin prosesseihin ennen siirtymistä virtuaalikoneista kontteihin sekä mahdollisesti arvioida, onko yrityksellä tai organisaatiolla resursseja suorittaa siirtyminen onnistuneesti.
Tässä tutkimuksessa konttien hallinta jäi vähemmälle huomiolle, sillä halusin keskittyä puhtaasti konttiteknologiaan perustasolla. Konttiteknologian käyttöönotosta ja hallinnoinnista isossa skaalassa onkin olemassa useita tutkimuksia, ja jatkotutkimuksen kohdentaminen johonkin tiettyyn hallinnointiteknologiaan olisikin hyvä jatkotutkimuksen kohde. Olisi esimerkiksi mahdollista tutkia eroja ja hyötyjä eri hallinnointi- ja käyttöönottoteknologioiden välillä.

Jatkotutkimuksen kannalta olisi myös mielekästä tutkia konteissa käytettäviä näköistiedostoja, niiden turvallisuusriskejä sekä verrata näköistiedostojen mahdollisia eroja sekä vaikutuksia suorituskykyyn. Tällaisia tutkimuksia onkin joitakin. Konttien turvallisuutta kehitetään jatkuvasti, joten haavoittuvuuksien ajantasainen tutkiminen on mielekästä.

7 Yhteenveto

Konteissa ohjelma eristetään omaan suljettuun ympäristöönsä tarvittavine kirjastoineen, ohjelmistoineen ja työkaluineen, josta ohjelma voi kommunikoida rajapintojen kautta muiden ohjelmien kanssa. Konttiteknologia tarjoaa kevyen ja tehokkaan vaihtoehdon virtuaalikoneille. Verrattuna käyttöjärjestelmän ytimeen pohjautuviin virtuaalikoneisiin kontit ovat joko yhtä tehokkaita tai tehokkaampia. Kontit ja virtuaalikoneet eivät kumpikaan aiheuta melkein yhtään lisärasitetta suoraan prosessorille tai muistinkäytölle, mutta siirräntäoperaatiot sekä vuorovaikutus käyttöjärjestelmän kanssa nostavat lisärasitetta epäsuorasti.

Konttiteknologia on virtualisointiteknologia, ja täten tarjoaa samoja ominaisuuksia ja etuja kuin virtualisointi eli osittelun, eristämisen ja kapseloinnin. Kontit ovat suorituskyvyltään yhtä tehokkaita tai tehokkaampia kuin käyttöjärjestelmän ytimeen pohjautuvat virtuaalikoneet. Virtuaalikoneisiin verrattuna kontit tarvitsevat vähemmän muistia, mikä samalla myös nopeuttaa konttien käyttöönottoa pilviympäristöissä. Konteissa käytettävät pohjanäköistiedostot mahdollistavat yhtenäisen ja tarpeiden mukaan muokattavan kehityspohjan hajautetuissa järjestelmissä. Konttien voidaan sanoa olevan turvallisia, mutta kehittäjien on hyvä olla tietoisia mahdollisista hyökkäysvektoreista sekä tietoturva-aukoista.

Taulukkoon 2 on listattu tässä tutkielmassa esille tulleet konttiteknologian edut ja hyödyt sekä tärkeimmät lisähuomiot. Etuja on käsitelty tarkemmin luvussa 4.

Taulukko 2. Lista konttiteknologian eduista ja hyödyistä.

Etu ja hyöty	Huomio
Osittelu.	Virtualisoinnille ominainen etu.
Eristäminen.	Virtualisoinnille ominainen etu.
Kapselointi.	Virtualisoinnille ominainen etu.
Suorituskyvyltään yhtä tehokkaita tai tehokkaampia kuin virtuaalikoneet.	Suorituskykyyn vaikuttavat muut konttiteknologian edut, kuten alhaisempi yleisrasite, alhaisempi keskusmuistin käyttö sekä nopeampi käynnistyminen.
Alhaisempi lisärasite verrattuna virtuaalikoneisiin.	Siirräntäoperaatiot suurin pullonkaula.
Alhainen keskusmuistin käyttö.	Konteissa ei ole ylimääräisiä hallintoprosesseja, joten näiden ylläpitoon ei tarvita keskusmuistia.
Nopea käynnistyminen.	Kontti voi käynnistyä jopa alle sekunnissa.
Yhtenäinen kehityspohja hajautetuissa järjestelmissä.	Näköistiedostojen koostuminen kerroksista mahdollistaa yhtenäisen ja helposti liikuteltavan kehityspohjan.
Turvallisuus.	Konteilla ei pääsyä mihinkään, mitä ne eivät voi nähdä.

Lähdeluettelo

Abraham S., Paul A. K., Khan R. I. S. and Butt A. R. (2020). On the Use of Containers in High Performance Computing Environments. In Proceedings of the *2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020, pp. 284-293*. doi: 10.1109/CLOUD49709.2020.00048.

Biradar S. M., Shekhar R. and Reddy A. P. (2018). Build Minimal Docker Container Using Golang. In Proceedings of the *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 1-4.* doi: 10.1109/ICCONS.2018.8663172.

Bui T. (2015). Analysis of Docker Security. Aalto University School of Science *https://arxiv.org/abs/1501.02967*

Casalicchio E., Iannucci S. (2020). The state-of-the-art in container technologies: Application, orchestration and security. *Concurrency Computat Pract Exper. 2020; 32:e5668. Volume 32, Issue 17, September 10, 2020.* https://doi-org.libproxy.tuni.fi/10.1002/cpe.5668

Chung M. T., Quang-Hung N., Nguyen M. and Thoai N. (2016). Using Docker in high performance computing applications. In Proceedings of the *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 2016, pp. 52-57*. doi: 10.1109/CCE.2016.7562612

Felter W., Ferreira A., Rajamony R. and Rubio J. (2014). An Updated Performance Comparison of Virtual Machines and Linux Containers. Technical Report No. RC25482(AUS1407-001). Austin, TX: IBM Research Division, Austin Research Laboratory; 2014.

Kozhirbayev Z., Sinnot R. O. (2017). A performance comparison of container-based technologies for the Cloud. Future Generation Computer Systems Volume 68, March 2017, Pages 175-182.

Lin C., Nadi S. and Khazaei H. (2020). A Large-scale Data Set and an Empirical Study of Docker Images Hosted on Docker Hub. In Proceedings of the *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* doi: 10.1109/ICSME46990.2020.00043

Martin, J. P., Kandasamy A. and Chandrasekaran K. (2018). Exploring the support for high performance applications in the container runtime environment. *Human-centric Computing and Information Sciences; Volume 8, Issue 1. https://doi.org/10.1186/s13673-017-0124-3*

McGee, J. (2016). The 6 steps of the container lifecycle. *https://www.ibm.com/blogs/cloud-computing/2016/02/the-6-steps-of-the-container-lifecycle/*. 2016. Luettu 15.6.2021.

Morabito R., Kjällman J. and Komu M. (2015). Hypervisors vs. Lightweight Virtualization: A Performance Comparison. In Proceedings of the *2015 IEEE International Conference on Cloud Engineering*. 9-13 March 2015, Tempe, AZ, USA. https://ieeexplore.ieee.org/document/7092949

Nguyen B., Chung M. T., Quang-Hung N., Nguyen M-T. and Thoai N. (2017). Scale-down methods for optimizing resource allocation in providing Virtual Laboratory environment by cloud computing. In Proceedings of the *2017 International Conference on Advanced Computing and Applications (ACOMP), 2017, pp. 54-61.*

doi: 10.1109/ACOMP.2017.18

Pahl C., Brogi A., Soldani J. and Jamshidi P. (2019). Cloud Container Technologies: A State-of-the-Art Review. In Proceedings of the *IEEE Transactions on Cloud Computing, Volume 7, Issue 3*, pp. 677 – 692, 1 July-Sept. 2019. *doi: 10.1109/TCC.2017.2702586*

Pooja, Pandey A. (2014). Virtual Machine Performance Measurement. *Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06 – 08 March*. 2014.

Red Hat. (2021). Virtualization: What is a hypervisor? *https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor*. Luettu 10.6.2021.

Watada J., Roy A., Kadikar R., Pham H. and Xu B. (2019). Emerging Trends, Techniques and Open Issues of Containerization: A Review. *IEEE Access Volume 7, p. 152443 - 152472*. 07 October 2019. doi: 10.1109/ACCESS.2019.2945930

Xie X., Wang P. and Wang Q. (2017). Performance comparison between Linux containers and virtual machines. In Proceedings of the *13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2015, pp. 342-346*. doi: 10.1109/ICACEA.2015.7164727