Serverless-arkkitehtuurin haasteet verrattuna perinteisiin pilvipalvelumalleihin

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Kesäkuu 2026
Arttu Ikonen

TURUN YLIOPISTO
Tietotekniikan laitos

Arttu Ikonen: Serverless-arkkitehtuurin haasteet verrattuna perinteisiin pilvipalvelumalleihin

TkK-tutkielma, 22 s.
Tietotekniikka
Kesäkuu 2026

Serverless-arkkitehtuuri siirtää palvelinten ylläpidon ja toiminnan vastuun ohjelmiston kehittäjältä palveluntarjoajalle. Tämä helpottaa ohjelmistokehitystä ja mahdollistaa sovellusten joustavan ja automaattisen skaalaamisen, sekä nopeuttaa sovellusten kehittämistä ja käyttöönottoa. Serverless-malli tuo kuitenkin mukanaan myös sille ominaisia haasteita. Tämän kanditutkielman tarkoituksena on selvittää, miten serverless-arkkitehtuurin haasteet eroavat perinteisten pilvipalvelujen haasteista sekä millaisia uusia haasteita serverless-malli luo.
Tutkielma on toteutettu kirjallisuuskatsauksena. Aineisto on kerätty kahdesta tieteellisestä tietokannasta, Web of Sciencesta ja ACM:stä, hakutermien avulla, jotka kattoivat serverless-arkkitehtuurin haasteet, rajoitukset ja heikkoudet. Tarkasteluun valikoitui vuodesta 2020 alkaen julkaistuja vertaisarvioituja artikkeleita, jotka käsittelivät serverless-mallia ja sen haasteita.
Tutkimuksen keskeisiksi tuloksiksi nousevat erityisesti kylmäkäynnistykseen liittyvä viive, kustannukset sekä toimittajasidonnaisuus. Lisäksi serverless-alustalla toteutettujen palvelinratkaisuden testaaminen, virheenjäljitys ja monitorointi osoittautuvat huomattavasti perinteisiä pilvipalveluja haastavimmiksi. Tietoturvan näkökulmasta vastuunjako kehittäjän ja palveluntarjoajan välillä muuttuu. Tutkimuksen pohjalta osoitetaan serverless-arkkitehtuurin soveltuvan parhaiten tapahtumaohjautuviin ja epäsäännöllisesti kuormittuviin sovelluksiin. Se asettaa kuitenkin merkittäviä rajoituksia pitkäkestoisia prosesseja vaativille ohjelmille.

Asiasanat: serverless, pilvipalvelut, haasteet

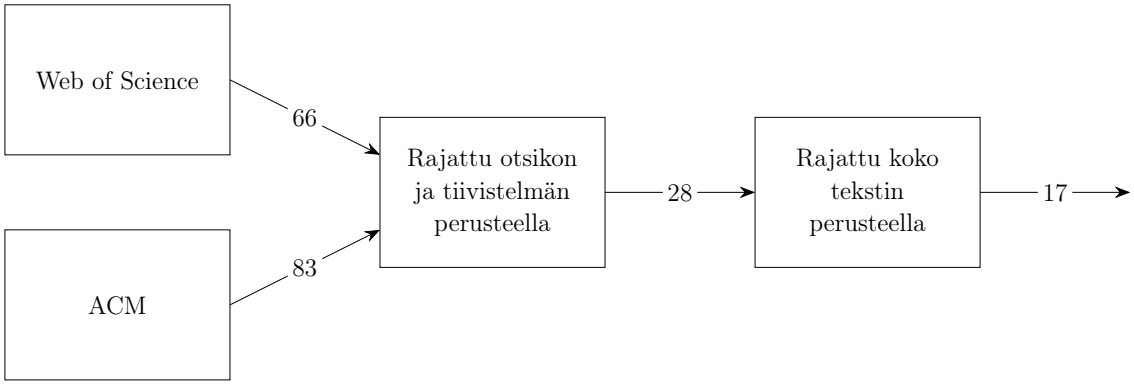
Sisällys

1 Johdanto

Pilvipalvelut tarjoavat joustavamman ja skaalautuvamman tavan kehittää ohjelmistoja verrattuna perinteisiin palvelinratkaisuihin. Abstrahointitason kasvaessa kehittäjän vastuu infrastruktuurista vähenee ja kehittäjä voi keskittyä yhä enemmän sovelluksen kehitykseen. Eräs pilvipalveluiden kehityksen uusimmista askeleista on serverless-malli, jonka suosio on kasvanut merkittävästi lähivuosina. Serverless-mallin lupaukset automaattisesta skaalautuvuudesta ja ylläpidon vastuun siirtämisestä palveluntarjoajalle houkuttelevat kehittäjiä, mutta malli tuo mukanaan sille ominaisia haasteita, jotka eroavat perinteisten pilvipalvelumallien haasteista. Tutkielmassa tutkitaan, miten serverless-arkkitehtuuri eroaa perinteisistä pilvipalvelumalleista ja mitä uusia haasteita se luo. Tutkimuskysymys on:

TK1: Mitä haasteita serverless-arkkitehtuuri tuo perinteisiin pilvipalvelumalleihin verrattuina?

Tiedonhaku pohjautuu kahteen tietokantaan, Web of Science ja ACM. Tiedonvalintaprosessia on kuvattu kuvassa 1.1. Web of Science -tietokannan hakulauseena käytettiin "*serverless architecture*" OR "*serverless computing*" (Author Keywords) AND *challenges* OR *limitations* OR *drawbacks* (Topic). ACM-tietokannan tiedonhakuun käytettiin hakulausetta (Keywords: "*serverless architecture*" OR Keywords: "*serverless computing*") AND (Abstract: *challenges* OR Abstract: *limitations* OR Abstract: *drawbacks*). Tarkasteluun sisällytettiin vuodesta 2020 alkaen julkaistut lähteet ja karsittiin pois artikkeleita joilla ei ollut viittauksia lainkaan.



Kuva 1.1: Tiedonvalintaprosessi

Hakutuloksista otsikon ja tiivistelmän perusteella karsittiin artikkeleita, jotka keskittyvät enemmän serverless-arkkitehtuurin pohjautuviin ratkaisuihin, kuin itse serverless-arkkitehtuuriin, lukuunottamatta artikkeleita, jotka tiivistelmän perusteella saattaisivat keskittyä serverless-mallin aiheuttamiin haasteisiin ja niiden ratkaisemiseen jonkin käyttökohteen kautta.

Luvussa 2 käsitellään pilvipalvelumalleja ja niiden toimintaperiaatteita. Yleisimpien pilvipalvelumalleihin tutustumisen jälkeen tutkitaan serverless-mallin määritelmää ja miten malli sijoittuu perinteisiin pilvipalvelumalleihin toiminnan ja toteutuksen pohjalta. Luvun lopussa avataan serverless-malliin liittyviä teknologioita ja käsitteitä, joiden ymmärtäminen toimii pohjana seuraavan luvun käsittelylle. Seuraavaksi luvussa 3 syvennytään eri serverless-mallin haasteiden osa-alueisiin, tavoitteena tunnistaa serveless-mallille ominaisia haasteita, kaikille pilvipalvelumalleille tyypillisten haasteiden seasta. Luvussa tutkitaan lyhyesti myös vastakohtana serverless-mallin hyötyjä.

Pohdintaa katsauksessa esiin tulleista ajatuksista käsitellään luvussa 4. Pohdinnassa mietitään mihin käyttökohteisiin serverless-malli soveltuu, miten mallin pintapuoliset edut eroavat todellisuudesta, sekä vastuunjaosta. Lopuksi löydetyt havainnot tuodaan yhteen luvussa 5. Yleisen yhteenvedon lisäksi luvussa mietitään

löydettyjen lähteiden ulkopuolelle jääneitä haasteita, sekä tuodaan mukaan omaa pohdintaa jatkotutkimuskysymyksistä.

2 Tausta

Pilvipalveluiden ja niiden haasteiden ymmärtämiseksi on ensin tarkasteltava, mitä pilvipalvelut ovat, miten ne toimivat ja miten ne eroavat toisistaan. Tässä luvussa käsitellään pilvipalvelumalleja sekä perehdytään erityisesti serverless-malliin ja sen toteutustapoihin.

2.1 Pilvipalvelumallit

Pilvipalveluiden käyttö perustuu erilaisiin palvelumalleihin (engl. X as a service, *aaS), jotka määrittävät mitä osia järjestelmästä käyttäjä itse hallitsee ja mitkä on ulkoistettu palveluntarjoajalle. Pilvipalveluiden keskeisimmät palvelumallit ovat infrastriktuuri palveluna (engl. Infrastructure as a Service, IaaS), alusta palveluna (engl. Platform as a Service, PaaS) ja ohjelmisto palveluna (engl. Software as a Service, SaaS) [1]. Näiden lisäksi on olemassa malleja jotka rakentuvat näiden päälle kuten funktio palveluna (engl. Function as a Service, FaaS) ja taustapalvelu palveluna (engl. Backend as a Service, BaaS). FaaS- ja BaaS-mallien kattokäsitteenä toimii serverless-malli. Serverless-malliin tutustutaan luvussa 2.2.

Infrastructure as a Service (IaaS) tarjoaa käyttäjälle ohjelmiston ajamiseen tarvittavan infrastuktuurin. Palveluntarjoaja vastaa fyysisestä laitteistosta ja sen ylläpidosta, kun taas käyttäjän vastuulle jää käyttöjärjestelmä sekä ajettavat sovellukset. IaaS-malli voidaan toteuttaa kahdella tavalla. Joko koko laitteisto varataan

yhdelle käyttäjälle, tai tyypillisemmin laitteiston resursseja jaetaan usean käyttäjän kesken. Resurssien jakaminen tapahtuu virtualisoinilla.

Virtualisointi on teknologia, jossa laskentaresursseja jaetaan useiksi virtuaalisiksi ympäristöiksi. Se muodostaa keskeisen perustan pilvipalveluille sekä monille mikropalveluarkkitehtuureille. IaaS-mallissa virtualisointia toteutetaan virtuaalikoneilla (eng. *virtual machine*, VM). Virtualisointiin, virtualisoinnin tekniikoihin ja niiden toimintaan perehdytään luvussa 2.3.[2]

Software as a Service (SaaS) -mallissa sovellus tarjotaan käyttäjälle palveluna verkon yli. Palveluntarjoaja vastaa sovelluksesta, sen ylläpidosta sekä taustalla olevasta infrastruktuurista, eikä käyttäjän tarvitse huolehtia teknisestä toteutuksesta. SaaS-malli kattaa laajan joukon sovelluksia, joita voidaan käyttää internetin välityksellä ilman paikallista asennusta. Esimerkkejä SaaS-mallia toteuttavista palveluista ovat pilvitallennuspalvelut, kuten *Google Drive* ja *Microsoft OneDrive*.[3]

Platform as a Service (PaaS) tarjoaa valmiin ympäristön ohjelmien kehittämiseen, testaamiseen ja ylläpitoon. Palveluntarjoaja hallitsee infrastruktuurin sekä alustan ja käyttäjän vastuulle jää sovellus. PaaSin tarkoitus on helpottaa kehittämistä keskittämällä kaiken sovelluksen kehittämiseen tarvittavan yhdelle alustalle[1]. Esimerkkinä toimii AWS Elastic Beanstalk, joka automatisoi web-sovellusten käyttöönoton ja skaalauksen[4]. Kehittäjä lataa alustalle oman sovelluksensa ja alusta vastaa automaattissesti kuormantasauksesta, kapasiteetin hallinnasta ja sovelluksen monitoroinnista. [5]

Function as a Service (FaaS) -mallin toiminta perustuu käyttäjän määrittämien tilattomiin funktioihin. Funktioiden ajaminen tapahtuu tapahtumapohjaisesti (engl. *event driven*). Tapahtuma on tyypillisesti HTTP-kutsu, mutta voi myös olla esimerkiksi ajastettu tapahtuma tai toinen funktiokutsu.[6] FaaS on toiminnaltaan vertailtavissa PaaSiin. Toisin kuin PaaS-palvelut, jossa koko ohjelmisto ajetaan omassa virtuaaliympäristössään, FaaS-mallissa funktiokutsuille luodaan ympäristö-

jä tarpeen mukaan. Tätä varten käytetään virtuaalikoneiden sijaan vaihtoehtoista virtualisoinnin menetelmää, konttiteknologioita. [7]

Kontit (engl. containers) ovat vaihtoehtoinen virtualisointimenetelmä, jossa virtualisointi tapahtuu laitteistotason sijaan käyttöjärjestelmätasolla. Konttiteknologioihin ja niiden toimintaan perehdytään luvussa 2.3. FaaS-funktiokutsujen ajamiseen preferoidaan kontteja tai erikoistuneita virtuaalikoneratkaisuja niiden homattavasti lyhyempien käynnistysaikojen takia, verrattuna perinteisiin virtualisoinnin muotoihin, joka on FaaS-mallin toiminnan kannalta tärkeää, koska käynnistysaika vaikuttaa suoraan funktion vasteaikaan [8].

Backend as a Service BaaS on palvelumalli, joka sijoittuu PaaS- ja SaaS-mallien väliin. BaaS-palvelut tarjoavat valmiita, yleisesti tarvittuja backend-toiminnallisuuksia, kuten autentikointia, käyttäjänhallintaa, ulkoisia integraatioita ja ilmoituspalveluita. BaaS-mallin tarkka toteutus vaihtelee palveluntarjoajien välillä. Jotkut antavat käyttäjän valita käytettävän tietokannan itse, kun taas toiset rajoittavat tietokannan yhteen ennalta määrättyyn vaihtoehtoon. BaaS-kattaa siis monenlaisia palveluita ja sen määrittelyn rajaamien yksiselitteisesti on haastavaa. [9]

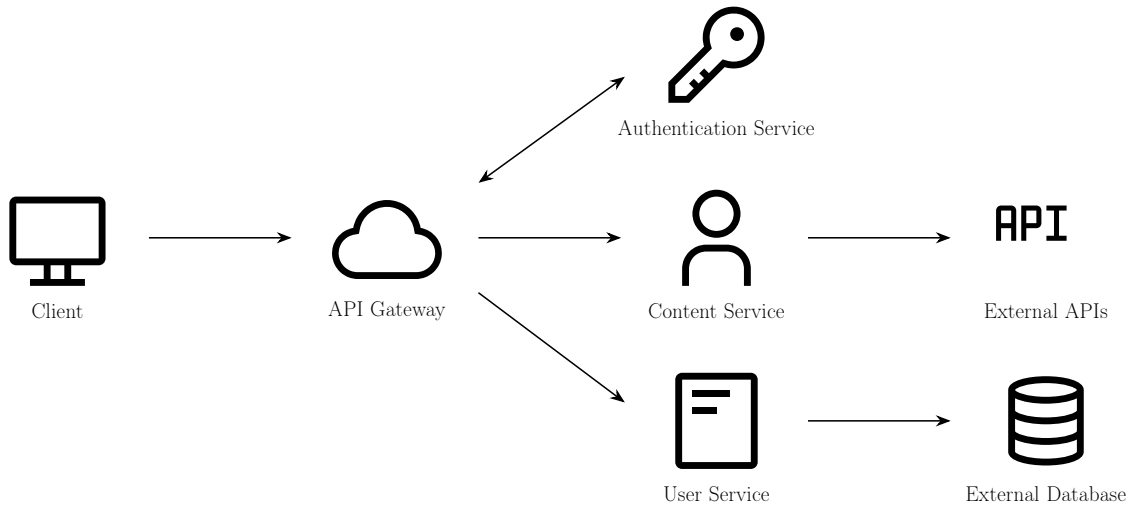
2.2 Serverless

Palveluntarjoajat tarjoavat erilaisia malleja eri tarpeisiin. Abstraktiotason kasvaessa kehittäjien tehokkuus ja mahdollisuus keskittyä uusien ja oleellisten ongelmien ratkaisemiseen paranee, kun yleisimmät pilvipalveluiden ongelmat tarjotaan valmiiksi ratkaistuna. Samalla kuitenkin kyky räätälöidä järjestelmä omiin tarpeisiin vähenee, kun yhä suurempi osa toteutuksesta siirtyy palveluntarjoajalle.

Yksi uusimmista ja suosiota keräävistä pilvipalvelumalleista on serverless-malli (myös serverless-arkkitehtuuri). Varsinaista määritelmää serverless-mallille ei ole olemassa, mutta yleisesti sen ajatellaan olevan FaaS- ja BaaS-mallien yhdistelmä [10].

Käytännössä serverless-malli tarkoittaa, että perinteinen palvelininfrastruktuuri ja sen hallinta ulkoistetaan kokonaan pilvipalveluntarjoajalle.

Serverless-mallin toteutukset yhdistävät FaaS-mallin funktiopohjaisen toimintamallin ja BaaS-mallin palvelut yhteen alustaan. Käyttäjä tekee kutsuja alustan API-yhdyskäytävään (engl. API Gateway). API Gateway toimii reitittimenä ja ohjaa kutsun kohteen perusteella sopivalle FaaS-funktiolle. Funktio käsittelee kutsun ja voi kommunikoida alustan muiden palveluiden, kuten tietokanta- tai auktorisointipalveluiden kanssa, tai tehdä kutsuja ulkoisiin rajapintoihin. Esimerkki serverless-alustan rakenteesta on kuvattu kuvassa 2.1.



Kuva 2.1: Esimerkki serverless-alustan rakenteesta

Serverless-mallin suosio perustuu sen joustavuuteen ja automaattisen skaalautuvuuteen. Resursseja ei varata etukäteen, vaan niitä allokoidaan ja vapautetaan funktiokutsujen mukaan. Tämä tekee serverless-mallista sopivan mallin sovelluksiin, joissa ruuhka on epäsäännöllistä, vaikeasti ennustettavissa tai käytönaste on suuren osan ajasta pientä.

2.3 Tekniikat

Virtualisointi on teknologia, jossa fyysisiä laskentaresursseja jaetaan tai yhdistetään yhdeksi tai useammaksi virtuaaliseksi käyttöympäristöksi. Virtualisointia voidaan toteuttaa useilla eri tavoilla, kuten laitteiston ja ohjelmiston osioinnilla, simulaatiolla, emulaatiolla sekä resurssien osituksella. Kun puhutaan virtualisoinnista yleisimmin puhutaan virtuaalikoneista (engl. *virtual machine*, VM) Virtuaalikone on eristetty ajoympäristö, joka emuloi fyysistä tietokonetta. [6]

Virtuaalikoneiden luomiseen ja ajamiseen käytetään hypervisor-ohjelmia eli virtualisointiympäristöjä (engl. *hypervisor, virtual machine monitor, VMM.* Virtualisointiympäristöt hallinnoivat virtuaalikoneita ja jakavat niille isäntälaitteiston resursseja, siten että ne eivät puutu toisiinsa. Tämä mahdollistaa muiden virtuaalikoneiden normaalin toiminnan esimerkiksi tilanteessa, jossa yksi virtuaalikoneista lakkaa toimimasta.

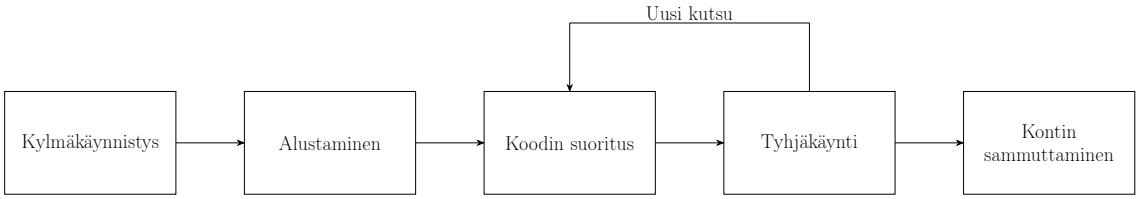
Konttiteknologiat ovat vaihtoehto virtuaalikoneisiin perustuvalle virtualisoinnille, jossa eristäminen tapahtuu laitteistotason sijaan käyttöjärjestelmätasolla konttien avulla. Virtuaalikoneista poiketen kontit jakavat isäntäjärjestelmän käyttöjärjestelmän. Konttiteknologioiden etu virtuaalikoneisiin on jaetun isäntäjärjestelmän mahdollistamat nopeat käynnistysajat ja kyky jakaa resursseja konttien välillä tehokkaammin.[11]

Konttiteknologioihin liittyy erinäisiä käsitteitä, joiden ymmärtäminen on olennaista teknologian hahmottamisen kannalta. Levykuva (engl. *image*) on staattinen tiedosto, joka sisältää kontin käynnistämiseen tarvittavat resurssit, kuten paketit, konfiguraation ja muut riippuvuudet [12]. Levykuvan avulla sovellus ja sen suoritusympäristö voidaan paketoida yhteen, jolloin levykuva voidaan ajaa yhtenäisesti eri alustoilla riippumatta isäntäjärjestelmästä.

Konttien luomiseen on olemassa ohjelmistoja, joista suosituin on Docker [13]. Docker on avoimen lähdekoodin alusta, joka mahdollistaa konttien rakentamisen,

käyttöönoton ja ajamisen [13]. Konttien määrän kasvaessa syntyi tarve automatisoida niiden hallintaa, käyttöönottoa ja skaalautumista. Tätä varten kehitettiin konttien orkestrointiautomaatiotyökaluja (engl. *container orchestration*), joista suosituin on Kubernetes. Kubernetes automatisoi konttien käyttöönoton, skaalautumisen ja vikasietoisuuden hallinnan, ja toimii perustana monille nykyaikaisille pilvipalvelualustoille, mukaan lukien serverless-toteutuksille. [14]

Virtuaalikoneiden ja konttien erot vaikuttavat niiden ominaisuuksiin. Virtuaalikoneiden käynnistäminen kestää konttien käynnistämistä pidempään huomattavasti. Tämä johtuu virtuaalikoneiden sisältämästä omasta käyttöjärjestelmästä ja virtuaalisesta ympäristöstä. Jotta voimme paremmin hahmottaa konttien käyttöä serverless-mallissa, tarkastellaan kontin elinkaarta kuvassa 2.2.



Kuva 2.2: Kontin elinkaari serverless-alustalla

Kuvassa kuvataan kontin eri vaiheita sen käynnistyksestä sammuttamiseen. Jos kutsulle ei ole valmiina olevaa konttia, syntyy kylmäkäynnistys. Tällöin uusi kontti luodaan lataamalla levykuva konttivarastosta, minkä jälkeen siitä rakennetaan ajoympäristö, ladataan riippuvuudet ja ajetaan alustuskoodi; kuvassa alustaminen. Kontin ollessaan valmis, se suorittaa kutsun mukaisen koodin ja jää odottamaan uusia kutsuja tyhjäkäynnillä. Jos uusia kutsuja ei tule määriteltyyn odotusaikaan mennessä, kontti sammutetaan.

3 Serverless-arkkitehtuurin haasteet

Serverless-mallin toteutus tuo mukanaan sille ominaisia haasteita, jotka eroavat perinteisten pilvipalvelumallien ongelmista. Tässä luvussa tutkitaan näitä haasteita useasta taulukossa 3.1 esitellyistä näkökulmista. Nämä näkökulmat ovat kylmäkäynnistys ja viiveet, suorituskyky ja resurssien hallinta sekä kustannukset. Luvussa 3.5 käsitellään muutamia luokittelemattomia haasteita.

Taulukko 3.1: Aineistot ja niiden näkökulmat

	Kylmäkäynnistys ja viiveet	Suorituskyky ja resurssien hallinta	Kustannukset	Turvallisuus ja yksityisyys	Muita haasteita			
					Toimittajaloukku	Testaus ja debuggaus	Tilanhallinta	Ympäristövaikutukset
Li et al. 2022 [10]	x	x	x	x	x	x		
Wen et al. 2023 [15]	x	x	x	x			x	
Barrak et al. 2022 [16]	x	x	x	x	x	x		
Golec et al. 2024 [17]	x	x	x		x			
Wen et al. 2021 [18]	x	x	x	x		x		
Gajanin et al. 2025 [19]	x	x	x	x				
Moreno-Vozmediano et al. 2023 [20]	x		x					
Tusa et al. 2024 [21]	x	x	x					
Wang et al. 2024 [22]		x	x					
Akour ja Alenezi 2025 [23]	x	x	x	x	x			x
Ni et al. 2024 [24]				x	x			
Roy et al. 2024 [25]	x	x					x	x
Farahani et al. 2024 [26]	x	x	x				x	
Shafiei et al. 2022 [27]	x	x	x	x		x	x	
Qiu et al. 2021 [28]	x	x	x					
Lin ja Shahrad 2025 [29]	x	x	x					x

3.1 Kylmäkäynnistys ja viiveet

Yksi serverless-mallin eduista perinteisiin pilvipalvelumalleihin on sen skaalautuvuus [10, 15]. Serverless-alustat kykenevät skaalautumaan automaattisesti alustan työkuorman vaihdellessa. Tarvittaessa serverless-malli mahdollistaa skaalautumisen nollaan (engl. *scale to zero*), jolla tarkoitetaan kaikkien sovelluksen suoritusympäristöjen eli instanssien sammuttamista, kun niille ei ole tarvetta [15]. Tämä luo uuden haasteen: Alustan skaalautuessa nollaan, voi esiintyä tilanne, jolloin alustalla ei ole

ainuttakaan instanssia käynnissä. Tällöin joudutaan käynnistämään uusi instanssi, joka vie aikaa ja aiheuttaa viivettä. [15] Tätä ilmiötä kutsutaan kylmäkäynnistykseksi (engl. *cold start*). [10, 15–17, 19]

Kylmäkäynnistyksen aiheuttama viive koostuu monista osista. Alustan on varattava instansseja varten resursseja, käynnistettävä uusi kontti ja ladattava ajoympäristö, kirjastot ja lähdekoodi. Kontin käynnistysviivettä kasvattaa erityisesti kontin levykuvan hakeminen konttivarastosta silloin, kun kontti luodaan ensimmäistä kertaa [21]. Kylmäkäynnistyksen aiheuttamat viiveet voivat tyypillisesti olla 300-800 ms, mikä voi olla merkittävä haitta viivekriittisissä sovelluksissa [23].

Sen lisäksi, että kylmäkäynnistyksiä tapahtuu nollaan skaalautumisen seurauksena, serverless-mallin automaattinen skaalautuminen aiheuttaa niitä myös toisella tavalla. Jos alustan työkuorma kasvaa äkillisesti, tarvittavien instanssien määrä voi ylittää käynnissä olevien instanssien kapasiteetin, jolloin alustan on käynnistettävä uusia instansseja [10]. Tätä skaalautumista kutsutaan äkilliseksi skaalautumiseksi (engl. *burst scaling*). Äkillinen ruuhka saattaa aiheuttaa kylmäkäynnistyksiä myös instanssien resurssien jakamisen myötä. Resurssien loppuessa kesken, joutuu alusta käynnistämään uuden instanssin kutsun käsittelyä varten tai odottamaan resurssien vapautumista. [10]

Kylmäkäynnistyksen aiheuttavat viiveet eivät ole vakioaikaisia. Esimerkiksi käytettävä ohjelmointikieli vaikuttaa viiveisiin. Korkean tason ohjelmointikielet, kuten Python ja Java käyttävät omia ajonaikaisia ympäristöjään, jotka aiheuttavat viiveitä. Tulkittavat ohjelmointikielet (engl. *interpretative languages*) käynnistyvät nopeammin kuin käännetyt ohjelmointikielet (engl. *compiled languages*). [10]

Koska serverless-toteutukset eivät ennalta provisioi resursseja toisin kuin perinteiset pilvipalvelumallit, suoritusympäristö luodaan vasta funktiokutsun yhteydessä [21]. Tämä tekee kylmäkäynnistyksestä keskeisen ongelman serverless-mallissa. Vastaavia käynnistysviiveitä tapahtuu myös muissa pilvipalvelumalleissa, mutta ne

eivät serverless-palveluiden tavoin vaikuta sovelluksen pyyntöjen käsittelyyn. Tämän vuoksi niiden vaikutus käyttäjän kokemaan vasteaikaan on vähäisempi, eikä samalla tavalla haaste, kuten serverless-mallissa.

3.2 Suorituskyky ja resurssien hallinta

Pilvipalveluita vertaillessa ratkaisun suorituskyky on yksi merkittävimmistä kriteereistä. Serverless-mallilla on sille ominaisia suorituskykyhaasteita liittyen sen funktiopohjaisuuteen. Funktiopohjaisuus aiheuttaa myös resurssien hallintaan liittyviä ongelmia.

Resurssien hallinta serverless-mallissa eroaa perinteisistä pilvipalvelumalleista. Virtuaalikoneisiin perustuvissa palvelumalleissa isäntäjärjestelmän resurssit kuten muisti ja suorituskapasiteetti on varattu jatkuvasti käyttäjälle. Serverless-mallissa resursseja taas varataan funktiokutsuille tarpeen mukaan. Tällöin isäntäjärjestelmän resursseja voidaan käyttää useaan funktiokutsuun samanaikaisesti ja jakaa usean käyttäjän kesken. [21, 22] Serverless-mallissa käyttäjän on mahdollista määrittää ainoastaan serverless-funktion muistimäärää, jonka perusteella alusta määrittää muut resurssit kuten laskentakapasiteetin ja verkon kaistanleveyden. [22] Serverlesss-mallissa suorituskyvyn optimointi ja resurssitehokkuus ovat usein vastakkaisia tavoitteita. Konttien lämpimänä pitäminen parantaa vasteaikaa, mutta kasvattaa käytettyjen resurssien määrää ja energiankulutusta. [25]

Serverless-mallissa funktioita ajetaan samanaikaisesti samalla laitteistolla käsittelemään kutsuja ja samanaikaiset kutsut jakavat isäntäjärjestelmän resursseja. Palveluntarjoajat asettavat rajoituksia samanaikaisten funktiokutsujen määrälle (engl. *concurrency limits*). Ruuhkan noustessa äkillisesti serverless-alusta ei pysty käsittelemään asetetun rajan ylittäviä funktiokutsuja, jolloin ne evätään kokonaan tai laitetaan jonoon odottamaan, mikä aiheuttaa äkillistä viiveen nousua ylitystilanteissa.

Toisin kuin perinteisissä pilvipalvelumalleissa, suorituskyky ei heikkene asteittain vaan romahtaa äkillisesti rajan ylittyessä.

Välttääkseen tilanteita, joissa yksittäiset funktiokutsut varaisivat alustan resursseja pitkiä aikoja, asettavat palveluntarjoajat suoritusaikarajoituksia. Palveluntarjoajat toisin sanoen rajaavat tarjottavia resursseja ajallisesti. Käyttäjien tulee ottaa tämä huomioon funktioita määrittäessä ja toiminnallisuuksia saattaa joutua jakamaan toiminnallisuutta useaan funktioon. Aikarajoitukset eroavat serverless-alustojen välillä ja ovat tyypillisesti muutaman kymmenen ja tuhannen sekunnin väliltä. [15]

3.3 Kustannukset

Serverless-mallin skaalautuvuus tekee siitä usein kustannustehokkaan vaihtoehdon, kun käyttäjä maksaa vain allokoiduista tai käytetyistä resursseissta [15, 22–24, 26]. Kuitenkin serverless-malliin liittyy sille ominaisia kustannushaasteita, kuten ennaltalämmitysmenetelmät. Ennaltalämmitys (engl. prewarming) on kylmäkäynnistysten välttämistä varten kehitetty tekniikka, jossa serverless-alusta pitää instansseja käynnissä, jolloin kutsuja varten ei tarvitse käynnistää uusia instansseja. Ennaltalämmitys lievittää kylmäkäynnistyksien aiheuttamia viiveitä, mutta instanssien käynnissäpito aiheuttaa lisäkustannuksia [21]. Ennaltalämmityksen kustannukset ovat alustariippuvaisia, koska palveluntarjoajat käyttävät eri ennaltalämmitysstrategioita ja systeemiarkkitehtuuriratkaisuja. [20, 30]

Koska serverless-mallin kustannukset määräytyvät funktiokutsujen määrän, resurssien käytön ja funktioiden suoritusajan perusteella, on serverless-mallin kustannusten ennustaminen haastavaa. Kustannusten ennustamiseen on kuitenkin olemassa ratkaisuja. Yleisimpiin ratkaisuihin kuuluu tilastollisia- sekä regressiomalleja, mutta vakiintunutta ratkaisua ei vielä ole. Kustannusten ennustaminen on haaste sekä palveluntarjoajalle että käyttäjälle. Palveluntarjoajille haasteena on hinnoitte-

lumallin suunnittelun monimutkaisuus. Käyttäjän näkökulmasta haasteita luo kustannusten arvaamattomuus ennustustekniikoista riippumatta, joka vaikeuttaa budjetointia ja heikentää käyttäjäkokemusta. [15, 18, 22, 27]

Yleisesti ottaen serverless on kustannusten kannalta edullisempi ratkaisu, kuin perinteiset palvelintoteutukset ruuhkan ollessa alhaista ja epätasaista. Tapahtumapohjaiset sovellukset voivatkin saavuttaa jopa 70 % kustannussäästöt. Kuitenkin tasaisilla kuormilla perinteiset palvelinmallit voivat olla edullisempia. [23, 28]

3.4 Turvallisuus ja yksityisyys

Serverless-mallin toteutus avaa ovia uusille turvallisuuden ja yksityisyyden haasteille. Virtualisoinnissa turvallisuutta ja yksityisyyttä taataan eristyksellä. Perinteiset pilvipalvelumallit usein toteutetaan virtuaalikoneilla, joiden eristystaso on korkea. Virtuaalikoneiden korkea käynnistysviive, tekee niistä huonon vaihtoehdon serverless-mallille, joten serverless-mallin toteutuksissa käytetään lähes aina konttiteknologioita. Konttiteknologioiden suuri etu on niiden pieni käynnistysviive, mutta niillä on heikko eristystaso.

Konttien heikko eristystaso toimii etuna tiedon jakamisen kannalta funktioiden välillä, mutta luo riskin turvallisuuden ja yksityisyyden näkökulmasta. Kontit ovat riskialttiita erilaisille kernelibugeille, sekä sivukanavahyökkäyksille (engl. *side channel attack*), kuten Meltdown [31], Zombieload [32] ja Spectre [33]. Sivukanavahyökkäykset ovat tietoturvahyökkäyksiä, jotka hyödyntävät laitteiston mikropiiristä saatavia fyysisiä tietoja, kuten virrankulutusta tai sähkömagneettista säteilyä.[10]

Konttien heikkoa eristystasoa varten on kehitetty vaihtoehtoisia virtualisointivaihtoehtoja, jotka tarjoavat korkeamman eristystason ilman korkeaa käynnistysviivettä. Nämä ratkaisut kuten secure container tai unikernel tarjoavat korean eristystason ja alhaisen käynnistysviiveen, mutta verrattuna kontteihin eivät ole yhtälailla joustavia. Serverless-mallin riippuvuus alhaisesta käynnistysviiveestä, kylmä-

käynnistysaikoja välttäeksi pakottaa palveluntarjoajan kompromisoimaan jostakin virtualisoinnin piirteestä.[10]

Virtualisointiin liittyvien haasteiden lisäksi Serverless-mallin turvallisuutta uhkaa mallin FaaS-tyyppiset funktiokutsut. Funktioiden suoritusrooleille annetaan funktioiden toiminnan pohjalta oikeuksia käyttää ja kommunikoida muiden serverless-alustan osien kanssa, kuten tietokanta- tai auktorisointipalvelun kanssa. Jos roolille annetaan oikeuksia, jotka eivät ole funktion toiminnan kannalta tarpeellisia, tekee tämä kutsun alttiiksi mahdollisille tietoturvahyökkäyksille. Serverless-mallissa oikeuksien määrittäminen on käyttäjän vastuulla ja on FaaSille ja serverless-mallille ominainen haaste. [15, 16, 27]

3.5 Muita haasteita

Serverless-mallilla on ongelmia, joita on haastavampi jakaa laajempiin kategorioihin. Tässä alaluvussa käsitellään, muita serverless-mallissa esiintyviä haasteita, jotka eivät kuulu katsauksen muihin lukuihin.

Toimittajaloukku (engl. *vendor lock-in*) on tilanne, jossa käyttäjän on vaikeaa tai jopa lähes mahdotonta vaihtaa palveluntarjoajaa, koska alustan tarjoamat rajapinnat ja ajoympäristöt toimivat eri tavalla muiden palveluntarjoajan toteutuksista. Toimittajaloukku on erityisesti ongelma serverless-mallissa, koska malli kannustaa käyttämään palveluntarjoajan omia BaaS-palveluita, kuten tietokanta- tai auktorisointipalveluita, osana sovellusta. Tiivis integraatio näihin palveluihin tekee sovelluksesta riippuvaisen valitun palveluntarjoajan järjestelmään, jolloin toiseen järjestelmään tai yksittäisen palvelun vaihtaminen vaatisi merkittävää koodin uudelleenkirjoittamista. [10, 15, 17, 23, 24]

Testaus ja debuggaus on välttämätön osa sovellusten kehittämistä. Serverless-alustat tarjoavat työkaluja FaaS funktiokutsujen testaamisen ja debuggaukseen, mutta näiden ajatellaan olevan usein puutteellisia [15, 18, 27]. Tämä saattaa johtua

serverless-mallin mahdollistamasta monimutkaisesta sovellusarkkitehtuurista, joka tekee suoritusvirrasta vaikeammin uudelleentoistettavan ja täten vaikeammin testattavan. Serverless-alustojen testausta ja debuggausta toteutetaan tallentaa ja toistaa menetelmällä, jossa koodia ajetaan ja tapahtumia tallennetaan niiden tapahtuessa. Tämän pohjalta käyttäjä voi tunnistaa haluamattomia tilanteita ja virheiden lähteitä [15, 16].

Tilanhallinta on kehittämisen näkökulmasta serverless-mallin funktiopohjaisuuden aiheuttama haaste. Serverless-funktiot ovat tilattomia, mikä tekee tilan säilyttämisestä monimutkaisempaa, kuin perinteisillä menetelmillä. Jotta tila voidaan tallentaa, tulee se tallentaa tietokantaan funktion ajon yhteydessä. Muiden funktioiden tulee taas selvittää tila hakemalla se tietokannasta. Alustat tarjoavat ongelmaan perinteisiä tietokantapalveluja nopeampia integroituja ratkaisuja, kuten AWS s3 [34], mutta tilanhallinta kuitenkin lisää viiveitä. [15]

Ympäristövaikutusten merkitys on jatkuvassa nousussa. Kasvava osuus informaatiosektorin kasvihuonekaasupäästöistä on pilvipalveluiden aiheuttamia [29]. Vaikka serverless-mallin eduksi nähdään sen funktiopohjaisuus ja kyky käyttää resursseja vain tarvittaessa, mallin toteutuksissa on kuitenkin erilaisia ympäristöongelmia ja parannuskohteita päästöjen vähentämisen näkökulmasta.

Konttien ennaltalämmitys on esimerkki serverless-mallille ominaisesta haasteesta, jolla on ympäristöhaasteita. Ennaltalämmitys ja konttien lämpimänä pitäminen käyttää resursseja ja aihettaa täten päästöjä vaikka resursseille ei varsinaista käyttöä olekkaan. Pidemmät lämpimänäpitoajat johtavat jatkuvasti kasvavaan hiilijalanjälkeen ja ennaltalämmityksen osio serverless-funktioiden hiilijalanjäljestä onkin suurempi kuin varsinaisen funktion ajamisen hiilijalanjälki [23, 25].

Serverless-alustojen toteutuksissa resurssien allokointi konttien kesken pohjautuu yleisesti maksimaalisen suorituskyvyn saavuttamiseen ja kustannustehokkuuteen. Alustojen tavoitteena ei siis tyypillisesti ole resurssien säästämiseen tai minimaa-

liseen kuluttamiseen. Päin vastoin alustat hukkaavat resursseja paremman käyttäjäkokemuksen takaamiseksi. Resurssien kuluttaminen kuitenkin aiheuttaa päästöjä, joilta voitaisiin välttyä. [29]

Serverless-alustojen kustannusmalleilla on vaikutus niiden ympäristövaikutukselle. Palveluntarjoajat asettavat minimi kustannusaikoja, jonka pohjalta laskutetaan, jos serverless-funktion ajoaika on rajaa pienempi [29]. Tämä pakottaa käyttäjää maksamaan käyttämättömistä resursseista, kun koodin suoritus kestää vähemmän kuin asetettu kustannusaikaraja. Tämä lannistaa koodin optimointia, johtaen keskimääräisesti pidempiin ajoaikoihin ja suurempaan resurssien kuluttamiseen ja täten nostaen päästöjä. [29]

Kuitenkin serverless-malli vähentää energiankulutusta merkittävästi perinteisiin pilvipalvelumalleihin verrattuna. Tutkimusten mukaan energiankulutus voi laskea jopa 70 % ja operatiiviset kustannukset jopa 60 % verrattuna virtuaalikoneisiin pohjautuviin ratkaisuihin. [23]

4 Pohdinta

Edellisessä luvussa tarkasteltiin serverless-mallin haasteita aineiston pohjalta. Tässä luvussa tarkastellaan tehtyjä havaintoja eri näkökulmista ja pohdintaan niiden vaikutuksia serverless-mallin soveltuvuuteen ja vastuunjakoon.

Serverless-malli ei sovellu hyvin sovelluksiin, jotka vaativat tasaista ja jatkuvaa kuormaa. Tällaisissa tilanteista perinteisen palvelinratkaisut ovat kustannustehokkaampia, koska serverless-mallin hinnoittelu perustuu käyttömäärään eikä hyödynnä tasaisen kuorman ennustettavuutta. Myöskään reaaliaikaiset sovellukset eivät sovellu hyvin serverless-malliin, sillä kylmäkäynnistykset voivat aiheuttaa vaikeasti ennustettavia viiveitä.

Katsauksen myötä on ilmentynyt selkeä vastuunjako. Palveluntarjoajan vastuulla on palvelimen taustalla oleva infrastuktuuri, funktiopohjaisuuden mahdollistava ohjelmisto sekä muut BaaS-palvelut kuten tietokannat. Käyttäjälle jää serverless-funktioiden määrittäminen ja eri BaaS-palveluiden yhdistäminen. Vaikka serverless-malli markkinoi itseään kehitysprosessia yksinkertaistavana ratkaisuna, voidaan katsauksen perusteella todeta mallin monimutkaistavan ja vaikeuttavan perinteisesti yksinkertaisia asioita. Perinteisillä palvelintoteutuksilla helposti toteutettavat asiat saattavat vaatia serverless-mallissa huomattavan määrän konfiguraatiota. Tämä herättää kysymyksen, siitä miksi serverless-arkkitehtuurin suosio on jatkuvassa kasvussa. Onko kyse aidosta tarpeesta ulkoistaa palvelininfrastruktuurin toteutusta vai onko kyse trendistä.

Korkea abstraktiotaso tarkoittaa, että käyttäjältä jää piilon merkittävä osa alustan toiminnasta. Kylmäkäynnistys on tästä hyvä esimerkki. Käyttäjän havaitsee viiveen vaikutuksen, mutaa sen taustalla olevat prosessit eivät näy. Palveluntarjoajat tarjoavat tyypillisesti lokitiedot ainoastaan itse funktion suorituksesta, ei sitä edeltävistä alustan sisäisistä vaiheista. Puutteelliset lokitiedot vaikeuttavat vianselvitystä, koska oman koodin ja alustan vaikutuksia suorituskykyyn on vaikea erottaa toisistaan. Tämä kyseenalaistaa jälleen serverless-mallin helppokäyttöisyyden yhtenä sen keskeisistä vetonauloista.

Serverless-alustoihin perehtymisen myötä nousi esiin myös eettisiä kysymyksiä vastuunjaosta. Alustat priorisoivat tyypillisesti käyttäjäkokemusta resurssitehokkuuden, energiatehokkuuden ja ympäristövaikutusten yli, eivätkä yleensä tarjoa käyttäjille mahdollisuutta valita ympäristöystävällisempiä toteutuksia, vaikka se teknisellä tasolla olisi mahdollista. Tämä herättää kysymyksen siitä, kenelle vastuu ympäristövaikutuksista kuuluu. Vaikka palveluntarjoaja hallitsee infrastruktuuria ja siihen liittyvää optimointia, käyttäjä kantaa vastuun valitsemistaan palveluista. Nykyisessä mallissa käyttäjällä on kuitenkin rajalliset mahdollisuuden vaikuttaa alustan tekemiin päätöksiin. Tämän vuoksi vastuun voidaan katsoa siirtyvän osittain palveluntarjoajalle.

5 Yhteenveto

Tässä katsauksessa tutkittiin serverless-arkkitehtuurin haasteita verrattuna perinteisiin pilvipalvelumalleihin eri osa-alueiden kautta. Tarkastelun kohteina oli muun muassa kylmäkäynnistys, suorituskyky ja resurssien hallinta, turvallisuus sekä toimittajaloukku, testattavuus ja tilanhallinta.

Tutkielmassa tutkittiin tutkimuskysymystä TK1: *Miten serverless-arkkitehtuurin haasteet eroavat perinteisistä pilvipalvelumalleista?* Voidaan todeta, että serverless-mallin ominaiset haasteet johtuvat sen perusominaisuuksista ja toteutustavoista. Automaattinen skaalautuminen, funktiopohjaisuus ja infrastruktuurin ulkoistaminen ovat muutama esimerkki serverless-mallin ominaisuuksista, jotka luovat ongelmia, joita ei esiinny yhtä lailla perinteisissä pilvipalvelumalleissa.

Jatkotutkimus aiheesta on oleellista ja monet haasteista on mahdollista ratkaista uusilla teknologioilla. On olemassa jo tutkimusta AI-pohjaisista ennustusmalleista. Ennustusmallien parantamisella olisi valtava vaikutus serverless-mallin eri ongelmaosa-alueille. Paremmilla ennustusmalleilla voitaisiin vähentää kylmäkäynnistysaikoja ja täten parantaa palvelun suorituskykyä ja vähentää kustannuksia. Erikoistuneilla virtualisointimenetelmillä voidaan taas parantaa mallin toiminnallisuuta spesifeille käyttökohteille. Palveluntarjoajilla on myös mahdollisuus kehittää käyttäjäkokemusta parantamalla testaustyökaluja ja tarjoamalla parempaa yhteensopivuutta muiden pilvipalvelualustojen kanssa. Tämä ei ole kuitenkaan todennäköistä kehitystä, sillä pilvipalveluntarjoajien tarkoituksena on rajoittaa toimintaa

vain omille alustoilleen ja vaikeuttaa ohjelmiston siirtoa omalta alustalaan pois. Ratkaisuna toimittajaloukulle voisi toimia jonkinlainen kolmannen osapuolen ratkaisu.

Jatkotutkimusta tulisi toteuttaa myös tiedonhaun ulkopuolelle jääneistä haasteista. Esimerkiksi erilaiset regulaatiot aiheuttavat haasteita serverless-mallissa, kun mallin skaalautuvuus tekee datan fyysisen sijainnin seuraamisesta vaikeaa. Toinen tietoturvalainsäädännön aiheuttama haaste on taustajärjestelmien saavuttamattomuus käyttäjälle. Lainsäädäntö edellyttää, että tietoturvasyistä palveluiden tulee kerärätä lokidataa ja että lokidataa ei saisi jälkikäteen muokata vaan niistä tulee jäädä muuttumaton ketju. Tämä on helpompaa perinteisissä malleissa, mutta serverless-mallin hajautuneisuus ja taustapalveluiden abstrahointi vaikeuttaa seurantaa käyttäjälle. Useissa serverless-palveluissa on käyttäjän vastuulla erikseen konfiguroida seurantapalvelu tallentamaan lokit vaatimusten mukaisesti. Tämä on hyvä esimerkki siitä, kuinka serverless-mallin lupaus siirtää vastuuta infrastruktuurista, turvallisuudesta ja toteutuksesta on harhaanjohtava ja monin tavoin malli monimutkaistaa ja siirtää vastuuta.

Lähdeluettelo

[1] R. Prodan ja S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers", teoksessa *2009 10th IEEE/ACM International Conference on Grid Computing*, ISSN: 2152-1093, lokakuu 2009, s. 17–25. DOI: 10.1109/GRID.2009.5353074.

[2] Microsoft Corporation, *What is Infrastructure as a Service (IaaS)?* Viitattu 19. toukokuuta 2026. url: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas.

[3] Microsoft Corporation, *What is Software as a Service (SaaS)?* Viitattu 19. toukokuuta 2026. url: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas.

[4] *Web App Deployment - AWS Elastic Beanstalk - AWS*. viitattu 27. maaliskuuta 2026. url: https://aws.amazon.com/elasticbeanstalk/.

[5] IBM, *What Is Platform as a Service (PaaS)?* Viitattu 19. toukokuuta 2026. url: https://www.ibm.com/think/topics/paas.

[6] S. N. T.-c. Chiueh ja S. Brook, "A survey on virtualization technologies", *Rpe Report*, vol. 142, tammikuu 2005.

[7] IBM, *What Is Function as a Service (FaaS)?*, lokakuu 2021. viitattu 19. toukokuuta 2026. url: https://www.ibm.com/think/topics/faas.

[8] X. Li, X. Leng ja Y. Chen, *Securing Serverless Computing: Challenges, Solutions, and Opportunities*, arXiv:2105.12581 [cs.CR], toukokuu 2021. DOI: 10.48550/arXiv.2105.12581. viitattu 16. kesäkuuta 2026. url: http://arxiv.org/abs/2105.12581.

[9] Cloudflare, *What is BaaS? | Backend-as-a-Service vs. Serverless*. viitattu 19. toukokuuta 2026. url: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/.

[10] Z. Li, L. Guo, J. Cheng, Q. Chen, B. He ja M. Guo, "The Serverless Computing Survey: A Technical Primer for Design Architecture", *ACM Comput. Surv.*, vol. 54, nro 10s, 220:1–220:34, syyskuu 2022, ISSN: 0360-0300. DOI: 10.1145/3508360.

[11] O. Bentaleb, A. S. Z. Belloum, A. Sebaa ja A. El-Maouhab, "Containerization technologies: taxonomies, applications and challenges", *The Journal of Supercomputing*, vol. 78, nro 1, s. 1144–1181, tammikuu 2022, ISSN: 1573-0484. DOI: 10.1007/s11227-021-03914-1.

[12] Avnish, *Container Images*, maaliskuu 2025. viitattu 29. huhtikuuta 2026. url: https://bovem.medium.com/container-images-761087aba4ba.

[13] IBM, *What Is Docker?*, kesäkuu 2024. viitattu 28. huhtikuuta 2026. url: https://www.ibm.com/think/topics/docker.

[14] IBM, *What Is Kubernetes?*, maaliskuu 2024. viitattu 28. huhtikuuta 2026. url: https://www.ibm.com/think/topics/kubernetes.

[15] J. Wen, Z. Chen, X. Jin ja X. Liu, "Rise of the Planet of Serverless Computing: A Systematic Review", *ACM Transactions on Software Engineering and Methodology*, vol. 32, nro 5, 131:1–131:61, heinäkuu 2023, ISSN: 1049-331X. DOI: 10.1145/3579643.

[16] A. Barrak, F. Petrillo ja F. Jaafar, ”Serverless on Machine Learning: A Systematic Mapping Study”, *IEEE Access*, vol. 10, s. 99 337–99 352, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3206366.

[17] M. Golec, G. K. Walia, M. Kumar, F. Cuadrado, S. S. Gill ja S. Uhlig, ”Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions”, *ACM Computing Surveys*, vol. 57, nro 3, 65:1–65:36, marraskuu 2024, ISSN: 0360-0300. DOI: 10.1145/3700875.

[18] J. Wen et al., ”An empirical study on challenges of application development in serverless computing”, teoksessa *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, sarja ESEC/FSE 2021, New York, NY, USA: Association for Computing Machinery, elokuu 2021, s. 416–428, ISBN: 978-1-4503-8562-6. DOI: 10.1145/3468264.3468558.

[19] R. Gajanin, C. Marcelino ja S. Nastic, ”Performance Isolation for Serverless Functions”, *IEEE Transactions on Services Computing*, vol. 18, nro 6, s. 4408–4424, marraskuu 2025, ISSN: 1939-1374. DOI: 10.1109/TSC.2025.3619151.

[20] R. Moreno-Vozmediano, E. Huedo, R. S. Montero ja I. M. Llorente, ”Latency and resource consumption analysis for serverless edge analytics”, *Journal of Cloud Computing*, vol. 12, nro 1, s. 108, heinäkuu 2023, ISSN: 2192-113X. DOI: 10.1186/s13677-023-00485-9.

[21] F. Tusa, S. Clayman, A. Buzachis ja M. Fazio, ”Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time IoT analytics”, *Future Generation Computer Systems*, vol. 155, s. 204–218, kesäkuu 2024, ISSN: 0167-739X. DOI: 10.1016/j.future.2024.02.006.

[22] W. Wang, Q. Wu, Z. Zhang, J. Zeng, X. Zhang ja M. Zhou, ”A probabilistic modeling and evolutionary optimization approach for serverless workflow

configuration", *Software: Practice and Experience*, vol. 54, nro 9, s. 1697–1713, 2024. DOI: 10.1002/spe.3268.

[23] M. Akour ja M. Alenezi, "Reducing Environmental Impact with Sustainable Serverless Computing", *Sustainability*, vol. 17, nro 7, s. 2999, tammikuu 2025, ISSN: 2071-1050. DOI: 10.3390/su17072999.

[24] K. Ni, S. K. Mondal, H. M. D. Kabir, T. Tan ja H.-N. Dai, "Toward security quantification of serverless computing", *Journal of Cloud Computing*, vol. 13, nro 1, s. 140, syyskuu 2024, ISSN: 2192-113X. DOI: 10.1186/s13677-024-00703-y.

[25] R. B. Roy, R. Kanakagiri, Y. Jiang ja D. Tiwari, "The Hidden Carbon Footprint of Serverless Computing", teoksessa *Proceedings of the 2024 ACM Symposium on Cloud Computing*, sarja SoCC '24, New York, NY, USA: Association for Computing Machinery, marraskuu 2024, s. 570–579, ISBN: 979-8-4007-1286-9. DOI: 10.1145/3698038.3698546.

[26] R. Farahani, F. Loh, D. Roman ja R. Prodan, "Serverless Workflow Management on the Computing Continuum: A Mini-Survey", teoksessa *Companion of the 15th ACM/SPEC International Conference on Performance Engineering*, sarja ICPE '24 Companion, New York, NY, USA: Association for Computing Machinery, toukokuu 2024, s. 146–150, ISBN: 979-8-4007-0445-1. DOI: 10.1145/3629527.3652901.

[27] H. Shafiei, A. Khonsari ja P. Mousavi, "Serverless Computing: A Survey of Opportunities, Challenges, and Applications", *ACM Computing Surveys*, vol. 54, nro 11s, 239:1–239:32, marraskuu 2022, ISSN: 0360-0300. DOI: 10.1145/3510611.

[28] H. Qiu et al., "Is Function-as-a-Service a Good Fit for Latency-Critical Services?", teoksessa *Proceedings of the Seventh International Workshop on Ser-*

verless Computing (WoSC7) 2021, sarja WoSC '21, New York, NY, USA: Association for Computing Machinery, joulukuu 2021, s. 1–8, ISBN: 978-1-4503-9172-6. DOI: 10.1145/3493651.3493666.

[29] C. Lin ja M. Shahrad, "Bridging the Sustainability Gap in Serverless through Observability and Carbon-Aware Pricing", *SIGENERGY Energy Inform. Rev.*, vol. 4, nro 5, s. 120–126, huhtikuu 2025. DOI: 10.1145/3727200.3727218.

[30] "Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions", *ACM Computing Surveys*, DOI: 10.1145/3700875.

[31] M. Lipp et al., "Meltdown: Reading Kernel Memory from User Space", teoksessa *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[32] M. Schwarz et al., "ZombieLoad: Cross-Privilege-Boundary Data Sampling", teoksessa *CCS*, 2019.

[33] P. Kocher et al., "Spectre attacks: exploiting speculative execution", *Commun. ACM*, vol. 63, nro 7, s. 93–101, kesäkuu 2020, ISSN: 0001-0782. DOI: 10.1145/3399742.

[34] Amazon.com, Inc, *Amazon S3*. viitattu 12. toukokuuta 2026. url: https://aws.amazon.com/pm/serv-s3/.