Tampereen yliopisto

Lauri Korhonen:

RESURSSIEN SKAALAUS PILVILASKENTAYMPÄRISTÖSSÄ

RESOURCE SCALING IN CLOUD COMPUTING ENVIRONMENT

Tampereen teknillinen yliopisto Kandidaatintyö, 35 sivua
Kesäkuu 2019
Tieto- ja sähkötekniikan koulutusohjelma
Pääaine: Ohjelmistotekniikka
Tarkastaja: Pia Niemelä, tutkijatohtori

TIIVISTELMÄ

Lauri Korhonen: Resurssien skaalaus pilvilaskentaympäristössä, Resource Scaling in Cloud Computing Environment
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikka
Kesäkuu 2019

Tutkielmassa on tutkittu millaisia ovat hyvän pilvilaskentaympäristön resurssien skaalauksen mallin piirteitä. Pilvilaskentaympäristö mahdollistaa pilvipalvelujen toteuttamisen. Pilvipalvelut hyödyntävät konesalien palvelimia, joilla tehdään tarvittava laskenta. Tämä mahdollistaa yritysten ja kuluttajien palvelusovellusten vaatiman laskennan ulkoistamisen, sekä parantaa niiden saatavuutta. Tarvittavia resursseja ovat palvelimien prosessorit, keskusmuisti, tallennustila ja laajakaistayhteys. Oikeanlaisella resurssien skaalauksella vähennetään hukkaa, mikä parantaa kustannustehokkuutta vähentämällä ylimääräistä sähköenergian kulutusta.

Ennakoivalla skaalaamisella varaudutaan kasvavaan resurssien tarpeeseen. Ennakoivaa skaalaamista tarvitaan, koska reaktiivinen skaalaus on liian hidasta. Ennakoivan skaalauksen tärkeys korostuu tilanteissa, joissa resurssien tarve kasvaa äkillisesti. Reaktiivisessa skaalauksessa reagoidaan muuttuneeseen tarpeeseen, mutta resurssien käyttöönotossa on viive, joten pelkällä reaktiivisella skaalauksella ei välttämättä voida vastata tarpeeseen riittävän nopeasti, mikä heikentää palvelunlaatua eli Quality of Serviceä (QoS). On tärkeää, että palvelunlaatu pysyy stabiilina, koska huono laatu johtaa asiakkaiden menettämiseen. QoS on määritelty Service Level Agreementissa eli SLA:ssa, joka on pilvipalveluntarjoajan ja konesaleja vuokraavan tahon välinen sopimus palvelun tasosta.

Resurssien skaalauksella on suora vaikutus pilvipalvelujen toteuttamisen onnistumiseen pilvilaskentaympäristössä. Skaalaus on usein jaettu horisontaaliseen ja vertikaaliseen skaalaukseen pilvilaskentaympäristöissä, joissa hyödynnetään virtuaalikoneita. Horisontaalisessa skaalauksessa säädetään virtuaalikoneiden määrää, jotka suorittavat sovellusinstansseja. Vertikaalisessa skaalauksessa säädetään virtuaalikoneiden käytössä olevia resursseja. Hyvä tapa skaalata, on hyödyntää horisontaalisen ja vertikaalisen skaalauksen yhdistelmää, joka mahdollistaa tarkan skaalauksen. Tarkalla skaalauksella saadaan aikaan korkea käyttöaste.

Hyvä resurssien skaalaus tarjoaa korkean käyttöasteen heikentämättä palvelunlaatua. Skaalauksen on reagoitava resurssien tarpeeseen skaalaamalla dynaamisesti ja reaaliaikaisesti. Myös äkillisesti kasvaneeseen työkuormaan on reagoitava nopeasti ja tarvittaessa ennakoivasti, jotta palvelunlaatu pysyy hyvänä.

SISÄLLYSLUETTELO

JOHDANTO

Pilvilaskennan avulla voi tarjota asiakkaille helposti saatavilla olevia virtuaalisia palveluita, joita kutsutaan pilvipalveluiksi. Luonteenomaisesti pilvipalvelujen vaatima laskenta toteutetaan pilvipalvelimilla. Näitä palvelimia ylläpidetään eri puolilla maailmaa konesaleissa, jotka toteuttavat pilvilaskennan. Tämä kokonaisuus muodostaa liiketoiminnallisesti suuren sektorin tietojenkäsittelyssä ja niiden ennustetaan kasvavan 206 miljardin dollarin markkinoiksi vuonna 2019 (Forbes, 2018).

Pilvipalvelujen mahdollistamiseksi tarvitaan lukuisia konesaleja ympäri maailman viiveen minimoimiseksi. Esimerkiksi Googlella on 15 konesalia eri puolilla maailmaa. Google on myös ilmoittanut investoivansa 13 miljardia dollaria konesaleihin vuonna 2019 rakentamalla neljä uutta konesalia ja laajentamalla lukuisia jo olemassa olevia (Christine Hall, 2019). Marraskuussa 2018 Microsoft ilmoitti käytössään olevan yli 100 konesalia (Microsoft Cloud, 2018). Konesali tarvitsee toimiakseen paljon sähköä, sillä tyypillinen konesali voi kuluttaa saman verran sähköä, kuin 25 tuhatta modernia kotitaloutta (Miyuru Dayarathna et al., 2015). Konesalit ovat hyvin harvoin täysin toimettomia ja niiden prosessoreiden käyttöaste vaihtelee keskimäärin 10–50 %:n välillä (Luiz André Barroso & Urs Hölzle, 2007). Maailmassa on paljon muitakin isoja pilvipalveluntarjoajia, joilla on käytössään lukuisia konesaleja. Näiden konesalien kehnot käyttöasteet kumuloivat suuren määrän hukkaenergiaa, mikä tarkoittaa turhia menoeriä pilvipalveluyrityksille.

Paremmalla käyttöasteella voi vähentää sähkönkulutusta palvelunlaadusta karsimatta. Parempi käyttöaste tekee pilvilaskennasta myös kustannustehokkaampaa, ekologisempaa ja kilpailukykyisempää. Kun hyvällä tekniikalla skaalataan pilvilaskennan käytössä olevia resursseja, parannetaan käyttöastetta. On ehdotettu useita tapoja, kuinka skaalausta pilvilaskentaympäristöissä voi toteuttaa. Tässä tutkielmassa tutustutaan muutamaan ehdotettuun skaalaustapaan. Näissä eri tavoissa on esitelty erilaisia algoritmeja, itseoppivia järjestelmiä ja virtualisointitekniikoita. Tutkielmassa pohditaan mitä hyvän skaalaustekniikan tulee käsittää pilvipalveluille asetettujen laatuvaatimusten puitteissa.

1. PILVILASKENTA

1.1 Pilvilaskennan toteutus

Konesaleissa toteutetuilla pilvipalvelimilla pidetään yllä pilvipalveluja. Pilvipalvelut saavat tarvittavan datan palvelimella tapahtuvaa laskentaa varten internet-yhteyden välityksellä. Tämä tarvittava data tulee tyypillisesti asiakkailta, mikä määrittää ohjelman toiminnan. Asiakkaalta vastaanotettava data sisältää tyypillisesti pyyntöjä, jotka liittyvät palvelimella sijaitsevaa dataan. Konesalien palvelimilla voi olla valmiina tallennettuna tarvittavaa dataa palvelun mahdollistamiseksi. Datan tallentaminen, määrä ja merkitys riippuu tarjottavasta palvelusta. Dataa käsitellään pilvipalvelimella asiakkaan pyyntöjen mukaisesti, mikä tekee palvelusta interaktiivisen ja mahdollistaa raskaan laskennan ulkoistamisen palvelimille. Palvelun toteuttamiseksi tarvitaan erilaisia resursseja. Tyypillisesti tarvitaan tallennustilaa, keskusmuistia ja prosessoriyksiköitä.

Resursseja voi varata pilvipalvelimella käyttäjien sovelluksien käyttöön esimerkiksi hyödyntämällä virtuaalikoneita. Virtuaalikoneille voi varata resursseja niiden suorittamien sovelluksien tarpeiden mukaisesti. Asiakkaan ottaessa yhteyttä pilvipalveluun, osa käytettävistä resursseista varataan dynaamisesti asiakkaan palvelemiseksi. Pilvipalvelimella tämä voi tarkoittaa virtuaalikoneessa uuden sovelluksen käynnistämistä tai uuden virtuaalikoneinstanssin käynnistämistä hyödyntäen vapaata laitteistoa. Resursseja voi varata palvelua varten myös staattisesti siten, että konesalista varataan tietty fyysinen osa käyttöön. Tällaisessa tilanteessa varataan jokin määrä tai osa tallennusvälineistä, keskusmuistista ja prosessoreista määritettyjen palvelujen käyttöön. Se, miten varaus käytännössä tehdään, riippuu palveluntarjoajista ja asiakkaiden tekemistä sopimuksista. Tutkielmassa keskitytään dynaamiseen resurssien varaukseen.

1.2 Pilvilaskennan mahdollistamat pilvipalvelut

Pilvipalvelut käsittävät Software as a Service eli SaaS:n, Infrastructure as a Service eli IaaS:n ja Platform as a Service eli PaaS:n muodostaman kokonaisuuden. SaaS:lle tyypillistä on pilvessä sijaitsevat ohjelmistot tai ohjelmat, joita asiakkaat voi käyttää verkon välityksellä. IaaS tarjoaa pilvessä sijaitsevaa laskentatehoa, mikä mahdollistaa palvelimien ulkoistamisen. PaaS tarjoaa sovellusalustan yrityksille ja ohjelmistokehittäjille.

Pilvipalveluita on paljon eri tyyppisiä, joista kuluttajille lienee tunnetuimpien joukossa SaaS-tyyppiset suoratoistopalvelut, jotka ovat mullistaneet viihteen. Muun muassa musiikkia, elokuvia ja sarjoja voi pitää palvelimilla tallennettuina ja ne voi pakata sopivaan kokoon verkon kautta siirrettäväksi. Sopivaan kokoon pakattu data voidaan siirtää asiakkaan omistaman laajakaistayhteyden puitteissa mikä mahdollistaa keskeytyksettömän toiston. Näin asiakas voi nauttia viihteestä omistamatta sitä fyysisessä formaatissa. Asiakas voi käyttää palvelua, kun hänellä on laite, joka tukee sovellusta, sekä internet-yhteys. Jotkin sovellukset antavat myös laitteiden puskuroida dataa, mikä mahdollistaa palvelun rajoitetun käytön ilman internet-yhteyttä.

SaaS-markkinat muodostavat pilvipalveluissa suurimman sektorin ja tuottavat jopa yksittäisessä kvartaalissa 20 miljardia dollaria liikevoittoa. SaaS-markkinoiden tuottama liikevoitto kasvaa vuosittain 32%:lla. (Synergy Research Group, 2018.) Vuonna 2017 IaaS-markkinat kasvoiva 29,5% nousten 23,5 miljardiin dollariin (Gartner, 2018). PaaS-markkinoiden odotetaan kasvavan 20 miljardiin dollariin 2019 (Gartner, 2019). Kyseessä ovat suuret ja yhä kasvavat markkinat, mitkä rakentuvat pilvilaskennan varaan. Pilvipalvelut ovat tulleet jäädäkseen.

1.3 Pilvilaskennan sähkönkulutus

Pilvilaskenta ja sen mahdollistamat palvelut vaikuttavat hyvin laajasti IT-alalla yritysten ja kuluttajien toimiin. Pilvilaskenta koskettaa miltei kaikkia internetin palveluja hyödyntäviä ihmisiä suorasti tai epäsuorasti. Pilvilaskennan mahdollistaa jättimäinen infrastruktuuri, joka koostuu suuresta määrästä palvelimia ja nopeista yhteyksistä. Infrastruktuurin vaatima sähkön määrä vastaa 1,1 %:n ja 1,5 %:n väliltä maailman sähkönkulutuksesta ja osuus kasvaa ennusteen mukaan tulevaisuudessa. (Toni Mastelic et al., 2014.) Sähkönkulutuksen kasvu on luultavasti väistämätöntä pilvipalveluiden kasvaessa, mutta kasvun asteeseen voi vaikuttaa optimoimalla pilvipalvelimien sähköenergian käyttöä.

Pilvipalvelimien sähköenergiankulutuksella on merkittävä rooli infrastruktuurin kustannusrakenteessa, koska prosessorit ja muistit tarvitsevat sähkövirtaa toimiakseen. Prosessorien kuluttama sähköenergian määrä korreloi prosessointitehon kanssa, mikä tarkoittaa suurempaa tarvetta sähkölle laskentaintensiivisissä sovelluksissa tai ruuhkaisissa palveluissa. Ruuhka ja laskentaintensiiviset sovellukset aiheuttavat myös suurem-

man tarpeen komponenttien jäähdyttämiselle, mikä kasvattaa energiankulutusta. Prosessorit kuumenevat sitä enemmän, mitä korkeampi niiden kellotaajuus on. Mikäli sähkönkulutusta saataisiin optimoitua, pilvipalveluista saataisiin taloudellisesti kannattavampia. Optimoinnilla olisi myös ekologinen vaikutus, sillä nykymaailman sähköntuotannosta noin 80 % on peräisin fossiilisista polttoaineista (The Word Bank, 2015).

2. SKAALAUS

2.1 Skaalauksen tavoitteet

Pilvilaskennan skaalauksen pääasiallisena tavoitteena on resurssien korkea käyttöaste laatutavoitteiden puitteissa. Käyttöaste on suhdeluku resurssien käytön ja käytettävissä olevan kapasiteetin välillä. Hyvään käyttöasteeseen pääsee säätämällä käytettäviä resursseja suhteessa tarpeeseen. On esimerkiksi turhaa pitää prosessorilla korkeaa kellotaajuutta, kun alhaisempi kellotaajuus riittää, tai kun prosessoria ei tarvita. Tarpeeton prosessori olisi sähkönkulutuksen kannalta kannattavinta sammuttaa. Voisi ajatella, että korkeaan käyttöasteeseen pääsisi aina tarpeen mukaan sulkemalla ja käynnistämällä, tai asettamalla lepotilaan ja herättämällä prosessoreita. Korkea käyttöaste ei ole kuitenkaan ainoa skaalauksen tavoite ja siksi ennalta mainittu toiminta ei ole kannattavinta. Prosessorin käynnistäminen ja hyödyntäminen palvelun kannalta vie aikaa. Sama koskee muita palvelimen komponentteja, kuten tallennustilaa ja keskusmuistia.

Skaalauksen on otettava huomioon palvelujen responsiivisuus ja interaktiivisuus. Palvelujen käyttäjät eivät halua joutua odottelemaan saadakseen palvelua. Siksi resurssien skaalauksessa on huomioitava myös palvelun vasteajan minimointi. Huonosti käyttäjien syötteisiin reagoivat ja hitaat palvelut eivät houkuttele asiakkaita. Suoratoistopalvelu, joka olisi käytössä 90 % ajasta, voisi antaa kuluttajalle huonon kokemuksen palvelusta, sekä saada aikaan tilaussopimuksen peruuttamisen. Laskettuna 30 päivällä kuukaudessa, käytettävyysaste vastaisi 72 tuntia kuukaudesta, jolloin palvelua ei voisi käyttää. Huono palvelin voi jumiutua ruuhkan vuoksi, mikä on liiketoiminnan kannalta tappiollista ja kuluttajien kannalta harmillista. Suoratoistopalvelujen ruuhkaantumisesta johtuvien käytettävyysongelmien huono ajoittuminen tärkeisiin ajankohtiin on tappiollista kaikille sopimuksen osapuolille. Esimerkiksi suorien ja suosittujen urheilulähetysten pätkiminen tai täysi toimimattomuus aiheuttaa kuluttajille huonon käyttökokemuksen, mikä voi ohjata heidät etsimään parempaa palveluntarjoajaa. Tällaisessa tilanteessa suoratoistopalveluita tarjoava yritys menettää asiakkaita, samoin pilvilaskentaa vuokraava yritys.

Hyvä skaalaus siis takaa hyvän resurssien käyttöasteen hidastamatta palvelua. Hyvin toteutettuna se laskee hukkaenergian määrää ja pienentää kustannuksia. Sillä on myös vaikutus tuotettuun hiilijalanjälkeen ja palvelunlaatuun. On siis helppoa perustella, miksi hyvä skaalaus on tavoiteltava ominaisuus palvelimilla.

2.2 Service Level Agreement (SLA)

Service Level Agreement (SLA) on sopimus pilvipalveluntarjoajien ja asiakkaiden välillä, missä määritellään asiakkaan tarpeet ja ehdot, joita pilvipalveluntarjoajien on noudatettava. SLA:t ovat hyvin merkittäviä sellaisten asiakkaiden kannalta, jotka vuokraavat pilvilaskentakapasiteettia omien palvelun tuottamiseksi palvelimia tarjoavilta yrityksiltä. Kyseessä on kuitenkin kahden yrityksen välisestä liiketoimintasopimuksesta. Siispä palvelimia vuokralle tarjoavan yrityksen on vakuutettava, että asiakas saavuttaa liiketoiminnallista arvoa vuokraamalla palvelimia käyttöönsä. SLA on siis palvelimia tarjoavalle yritykselle tärkeä kaupankäynnin väline, jolla yritys vakuuttaa asiakkaalle palvelun laadukkuudesta. (Saravanan K & Rajaram M, 2015.)

Monet IT-alan yritykset ovat siirtäneet toimintaansa pilveen, mikä mahdollistaa liiketoiminnan harjoittamisen omistamatta palvelinrautaa ja vähentää välittömiä ylläpitokuluja. Pilveen siirtyminen korostaa SLA:n tärkeyttä etenkin niiden yritysten kannalta, jotka tarjoavat palveluita muille yrityksille. Pilveen siirtyneille yrityksille palvelujen saatavuuden takaaminen ja ehdoista sopiminen on ehdottoman tärkeää, koska yritysasiakkaiden menettäminen ja oman toimintakyvyn lamaantuminen tulee molemmille kalliiksi. Esimerkiksi jonkin IT-yrityksen tarjoama pilvessä sijaitseva toiminnanhallintajärjestelmä voi käytettävyysongelmien vuoksi hidastaa tai jopa pysäyttää teollisuusyrityksen tehdastoiminnan. Epästabiililla palvelimella pyörivä toiminnanhallintajärjestelmä tai muu kriittinen järjestelmä voi huonosti toimiessaan aiheuttaa suuriakin kuluja ja vaikeuksia yritykselle.

SLA:ssa määritellään erilaisia asioita riippuen siitä, onko kyse IaaS-, SaaS- vai PaaS-tyyppinen palvelu. Sopimukseen vaikuttaa myös se, että onko kyseessä kiinteä vai dynaaminen sopimus. Kiinteässä sopimuksessa maksetaan vakiokokoisesta palvelusta ja dynaamisessa sopimuksessa maksetaan käytön mukaan. On ehdotettuja lukuisia erilaisia standardeja ja käytäntöjä, joiden mukaan SLA tulisi strukturoida. Koska ei ole yhtä tiettyä hyväksyttyä ja vakiintunutta standardia SLA:n toteuttamiseen, sovitut asiat vaihtelevat SLA:iden välillä. (Saravanan K & Rajaram M, 2015.) Sovittuja asioita voi olla mm. vasteajat, palvelun käytettävyyden suhdeluvut, kompensointi, rangaistukset, takuut sekä muut tapauskohtaiset ehdot. Skaalauksen on huomioitava nämä ehdot, koska käytössä olevilla resursseilla on välitön vaikutus siihen, kuinka hyvin SLA:ssa määriteltyihin tavoitteisiin päästään.

2.3 Vertikaalinen ja horisontaalinen skaalaus

Skaalausta voi lähestyä eri tavoin, mutta yksi yleinen tapa on jakaa skaalaus horisontaaliseen ja vertikaaliseen skaalaukseen, sekä käyttää palvelinjärjestelmässä virtuaalikoneita. Horisontaalisessa skaalauksessa käynnistetään uusia virtuaalikoneita, jotka käynnistyttyään suorittavat tarvittuja sovellusinstansseja. Horisontaalisessa skaalauksessa haasteen luo uusien virtuaalikoneiden käynnistämisen ajoittaminen. Virtuaalikoneen käynnistäminen vie järjestelmästä riippuen tietyn määrän aikaa. Horisontaalisella skaalauksella ei voida vastata reaaliaikaiseen tarpeeseen, joten sillä ei voi tyydyttää välitöntä kapasiteettitarvetta. Virtuaalikoneen käynnistäminen vie sovelluksesta riippuen aikaa, koska sovellusten ajamisen mahdollistaa ajoympäristö, johon sovellus on suunniteltu suoritettavaksi. Jotta virtuaalikone voi käynnistää sopivan ajoympäristön, on virtuaalikoneen käyttöön ensin ladattava tarvittavia resursseja. Resursseja ovat esimerkiksi käyttöjärjestelmä sekä ohjainohjelmistot, jotka mahdollistavat määriteltyjen prosessorien, tallennustilalaitteiden ja keskusmuistin käytön. Virtuaalikoneen käynnistämän ajoympäristön jälkeen itse sovellus on ladattava virtuaalikoneen muistiin ja käynnistettävä, mikä taas vie oman aikansa.

Vertikaalisessa skaalauksessa vaikutetaan virtuaalikoneen käytettävissä oleviin resursseihin. Esimerkiksi virtuaalikoneen käytössä olevien prosessoreiden jännitettä ja kellotaajuutta säätämällä voi vaikuttaa virtuaalikoneen energiankulutukseen ja datankäsittelynopeuteen. Vertikaalisesta skaalauksesta, prosessorin jännitteen ja kellotaajuuden säätämisellä on suurin vaikutus energiankulutukseen, koska prosessorit ja niiden jäähdytinlaitteet kuluttavat enemmän energiaa kuin tallennustilalaitteet ja keskusmuistit. Vertikaalisella skaalauksella voi vastata reaaliaikaisesti kapasiteettitarpeeseen, mutta sen vaikutus kapasiteettiin on pienempi kuin horisontaalisen.

Laadukkaan palvelun mahdollistaa skaalaus, jossa yhdistyvät vertikaalinen ja horisontaalinen skaalaus. Suurempiin kapasiteetin tarpeen muutoksiin voi vastata horisontaalisella skaalauksella ja pienempiin vertikaalisella skaalauksella. Kun virtuaalikoneella ajettavan sovelluksen tarve resursseille muuttuu, voi tilanteeseen mukautua vertikaalisella skaalauksella. Esimerkiksi tilanteessa, jossa sovelluksen käyttäjä ei tarvitse nopeaa datan prosessointia, voi alikellottamalla prosessoria parantaa käyttöastetta. Vastaavasti ylikellottamalla voi vastata nopean prosessoinnin tarpeeseen. Kun palveluun ottaa yhteyden useampi määrä käyttäjiä ja syntyy suurempi tarve kapasiteetille, kuin mihin vertikaalisella skaalauksella voi vastata, on järkevää turvautua horisontaaliseen skaalaukseen.

Käynnistämällä uusia virtuaalikoneita tai herättämällä virtuaalikoneita lepotilasta, saadaan suurempi määrä kapasiteettia käytettäväksi. Tavoitteena on laadukas palvelu, joka on käyttäjien tarpeiden mukaan aina käytettävissä.

3. EHDOTETTUJA TAPOJA SKAALAUKSEEN

3.1 Kuormitusta ennustava skaalaus

3.1.1 Ennustavan skaalauksen toimintaperiaate

Tao Li et al. esittävät julkaisussaan Load Prediction-Based Automatic Scaling Cloud Computing (2016) ennakoivan lähestymisen skaalaukseen. Skaalausta ennakoi heidän kehittämä Linear Regression And Improved Knuth-Morris-Pratt (LRAIKMP) -yhdistelmäalgoritmi. Kuten nimestä voi päätellä, LRAIKMP on algoritmi, jossa yhdistyy lineaarisen regression ja Knuth-Morris-Pratt -algoritmin käyttö. Algoritmin tavoitteena on saavuttaa mahdollisimman pieni ero resurssien tarpeen ja käytettävissä olevien resurssien välillä, eli suuri käyttöaste. Ennakoinnilla pyritään päättelemään oikea ajankohta resurssien laajennukselle, jolla pyritään välttämään tilanteita, joissa resurssien tarve ylittää käytettävissä olevat resurssit. Tarpeen ylittäessä tarjolla olevat resurssit, palvelunlaatu kärsii, joten algoritmi pyrkii tällaisia tilanteita välttämällä parantamaan myös palvelun laatua. Kaava, joilla lineaarisen regression kertoimet ratkaistaan, ja parannellun Knuth-Morris-Pratt -algoritmin pseudokoodi, on esitelty julkaisussa.

Paranneltu Knuth-Morris-Pratt (KMP) -algoritmi on siten muunneltu versio alkuperäisestä, että siinä sallitaan tietty virhemarginaali. Alkuperäinen KMP on algoritmi, joka on suunniteltu löytämään tietty peräkkäinen kirjainyhdistelmä kirjainjonosta, eli siinä ei voi sallia virhettä. Tässä tapauksessa muunneltua KMP:tä käytetään löytämään historiatiedon avulla samanlaisuuksia resurssien tarpeista toistuvien ajankohtien avulla, eli etsitään trendejä. Esimerkiksi, jos perjantai-iltaisin palvelun käyttäjien määrä nousee, ja laskee yöllä, on ilman virhemarginaalia vaikea todeta kyseisen trendin olemassaoloa. On hyvin epätodennäköistä, että resurssien tarpeet täsmäävät jokaisessa mittausajankohdassa. Havaittujen trendien perusteella saadaan hyödyllistä tietoa, minkä perusteella voi ennakoida resurssien tarpeen muutoksia.

Koska paranneltu KMP on laskennallisesti lineaarisen regression menetelmää raskaampi, käytetään lineaarista regressiota käyttöasteen ollessa korkea. Lineaarinen regressio ei anna yhtä tarkkoja tuloksia, mutta se on laskennallisesti kevyempi, sekä soveltuu käytettäväksi tilanteessa, jossa käyttöaste on jo korkea. Siispä käyttöasteen ollessa

huono, käytetään paranneltua KMP:tä käyttöasteen parantamiseksi, jonka jälkeen käytetään lineaarista regressiota. Näin vähennetään itse skaalauksesta koituvia kustannuksia.

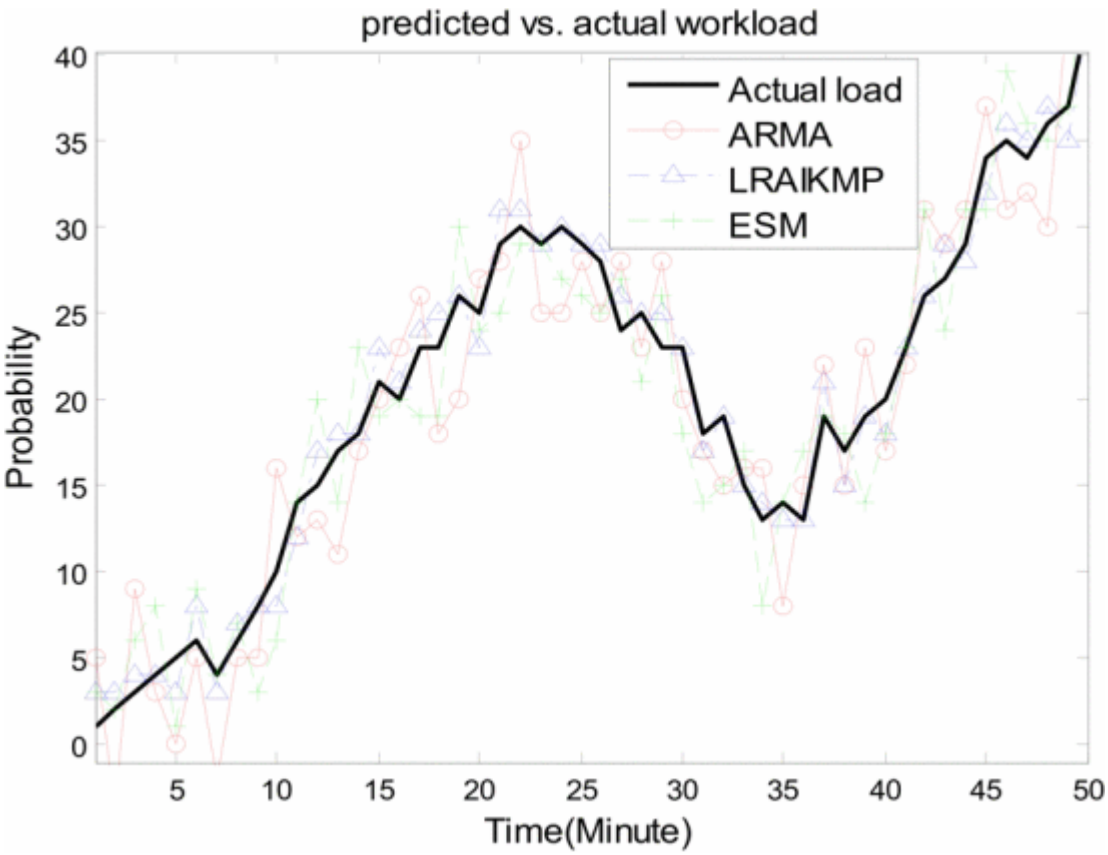
3.1.2 Skaalauksen toiminta eri tilanteissa

Li et al. esittävät skaalauksen toiminnan jaettuna kolmeen alalajiin tilanteesta riippuen: 1) Ennakoitua tietoa resurssien tarpeen muutoksesta käytetään päätöksessä lisätä tai vähentää käytettäviä resursseja, 2) Mikäli resurssien tarve ennakoinnista huolimatta on käytettävissä olevia resursseja suurempi, vastataan vertikaalisella skaalauksella välittömään tarpeeseen ja 3) Jos käytettävissä olevat resurssit ylittävät tarpeen, vapautetaan resursseja.

Julkaisun mukaan resurssien skaalauksessa on huomioitava resurssit ja kustannukset kokonaisuutena. Pyrkimyksenä on skaalata resursseja säästäen mahdollisimman paljon kustannuksia. Kun tiedetään milloin skaalata, julkaisussa esitelty algoritmi laskee kustannustehokkaimman resurssien skaalauksen käyttäen pilvipalvelun hinnastoja. Mikäli on mahdollista, vapautetaan resursseja horisontaalisesti, eli pyritään vapauttamaan virtuaalikone, jolla on käytössä vähiten resursseja. Oletuksen mukaan, ohjelmistolisenssimaksujen takia horisontaalinen skaalaus säästäisi eniten kustannuksia. Tämä johtuu yksittäisen virtuaalikoneen kiinteistä kustannuksista. Tilanne tietenkin riippuu käytettävistä ohjelmistoista. Muita kustannuksia ovat prosessorin, tallennustilan ja keskusmuistin käytöstä aiheutuvat kustannukset. Kustannustehokkain resurssien skaalaus riippuu hinnastosta, sopimuksesta ja suoritettavista ohjelmista.

3.1.3 Ennakoivan lähestymisen tuottamat ennusteet

LRAIKMP:n toimivuutta verrattiin julkaisussa eksponentiaalisen tasoituksen ja liukuvan keskiarvon malliin. Liukuvan keskiarvon malli käyttää resurssien tarpeen ennustamisessa hyödyksi tietoa viimeaikaisista kuormituksen tasoista. Eksponentiaalisen tasoituksen malli käyttää hyödyksi koko kuormituksen historiaa ja antaa eri ajanhetkille erilaisen painoarvon. Kaavio 1 havainnoi testauksen tuloksia. LRAIKMP:n lisäksi kaaviossa on merkattu ARMA, joka tarkoittaa liukuvan keskiarvon mallin ennustetta, ESM, joka tarkoittaa eksponentiaalisen tasoituksen ennustetta ja Actual load, joka on todellinen kuormituksen taso.



Kaavio 1: LRAIKMP:n vertaaminen liukuvan keskiarvon ja eksponentiaalisen tasoituksen tuottamiin ennusteisiin (Load Prediction-Based Automatic Scaling Cloud Computing, Tao Li et al., 2016).

Kaaviosta 1 voi havaita, kuinka LRAIKMP:n ennusteet ovat todellista kuormitusta lähimpänä. LRAIKMP:n ennusteet tuottavat siis stabiilimpia tuloksia. Stabiiliutta havainnoi taulukko 1, missä on lueteltu algoritmien pienin ja suurin virhe, sekä virheen keskiarvo.

Taulukko 1: LRAIKMP:n, ARMA:n ja ESM:n virhearvojen vertailu (Load Prediction-Based Automatic Scaling Cloud Computing, Tao Li et al., 2016).

Error	ARMA	ESM	LRAIKMP
minimum error	0	1	0
maximum error	67	78	56
average error	10	15	8

Taulukosta 1 voi havaita LRAIKMP:n tuottavan tarkimpia ennusteita kolmesta vertaillusta tavasta. LRAIKMP:n suurin virhe oli myös näistä kolmesta pienin. Suurimman ja keskimääräisen virheen ollessa pienin, syntyy vähiten hukkaa. Kyseinen algoritmi tuottaa siis kyseisistä kolmesta vertailluista parhaat tulokset. Toisaalta taulukkoon olisi voinut myös sisältää tiedon kumulatiivisesta tai keskimääräisestä kuormituksesta, koska voisi kuvitella, että joillekin palveluntarjoajille olisi tärkeää taata resurssien riittävyys. Jatkuva tai keskimääräinen vaje resursseista voi laskea palvelunlaatua.

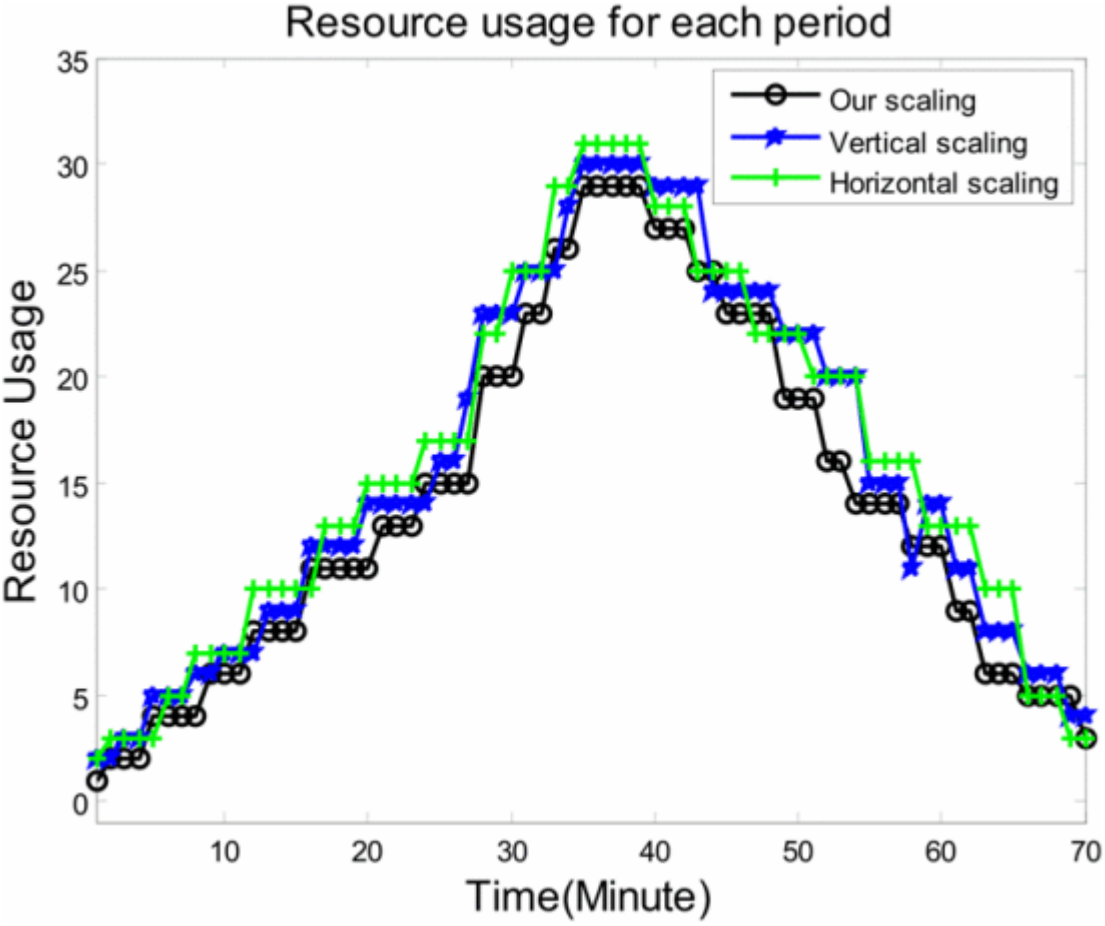
3.1.4 Ennakoivan lähestymisen skaalautuvuus

Koska LRAIKMP:n tavoitteena on hyvä käyttöaste ja hukan vähentäminen todellisessa järjestelmässä, on algoritmin skaalauksen toimivuutta testattu myös simuloidussa pilviympäristössä. Testauksessa on käytetty CloudSim nimistä pilviympäristöä simuloivaa ohjelmistoa. Referenssinä hinnastoille on käytetty Amazon elastic compute cloud -pilviympäristön hinnastoa.

Taulukko 2: Amazon elastic compute cloud -pilviympäristön hinnasto, joka ei ole päivitetty (Li et al., 2016).

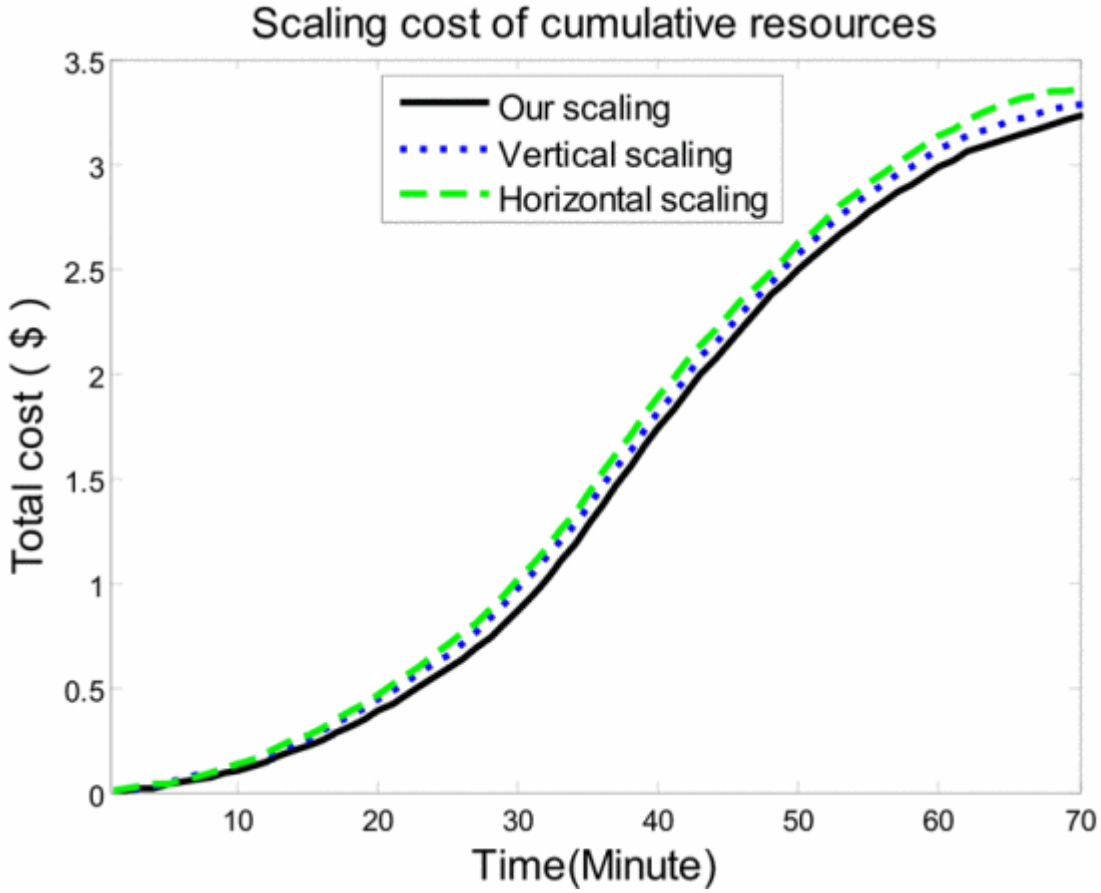
VM types	memory	number of CPU	size of disk	price per hour
small	2GB	1	30GB	$0.0065
medium	4GB	2	200GB	$0.062
large	8GB	2	400GB	$0.124

Pilvijärjestelmää testattiin simuloimalla kahdenlaisia palveluita, sosiaalisen median palveluja ja suoratoistopalveluja. Sosiaalisen median palvelu simuloi kuormitusta, joka koostui kuvien, tekstien ja kommenttien jakamisesta sekä muista sosiaalisen median toiminnoista. Suoratoistopalvelu simuloi kuormitusta, joka koostui videoiden jakamisesta palveluun, videoiden lataamisesta ja katselemisesta. Simuloinnin resurssien käyttö on kuvattu kaaviossa 2.



Kaavio 2: Resurssien käyttö simuloinnissa. Our scaling tarkoittaa LRAIKMP:tä, Vertical scaling vertikaalista skaalausta ja Horizontal scaling horisontaalista skaalausta. (Li et al., 2016)

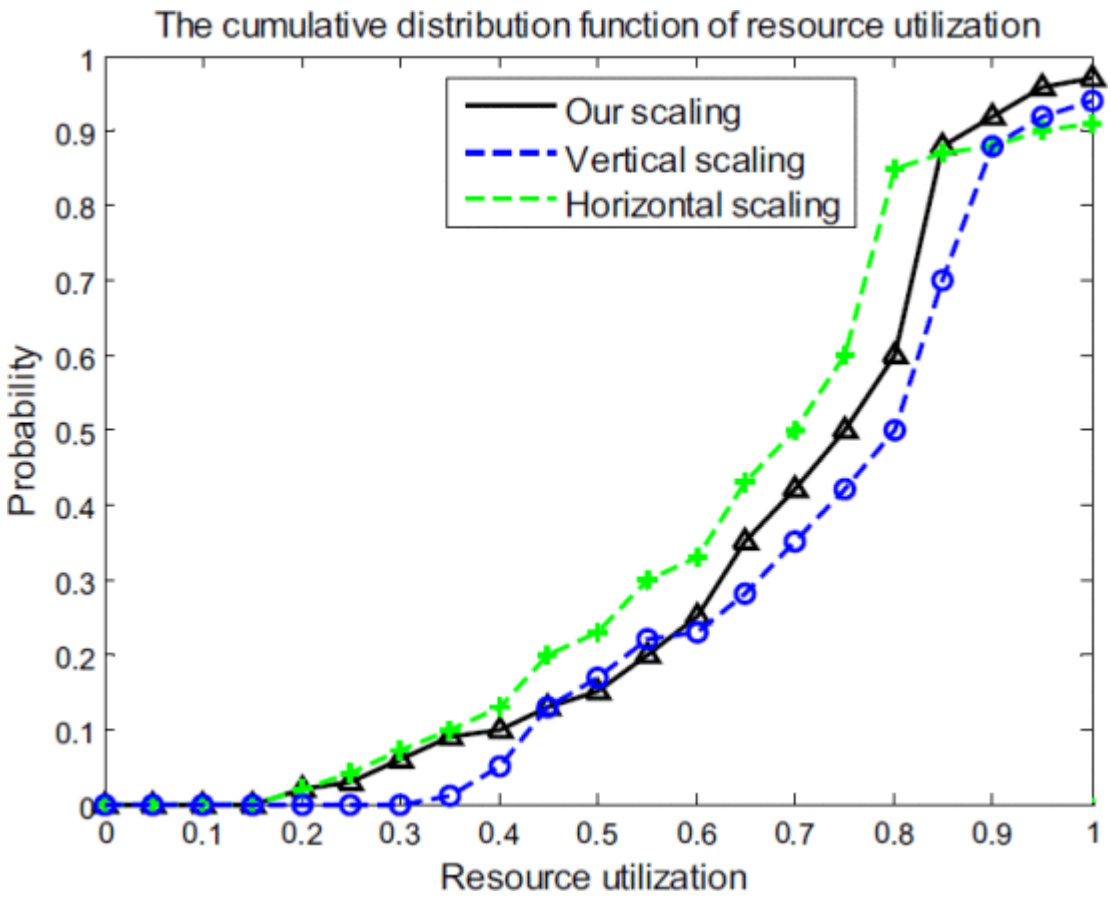
Kuten kaaviosta voi havaita, LRAIKMP käytti vertailluista kolmesta vähiten resursseja. Kyseinen algoritmi pystyy siis skaalaamaan käytettäviä resursseja kustannustehokkaimmin saman palvelun tuottamiseksi. Kustannustehokkuus on hyvin havaittavissa kaaviossa 3, jossa on kuvattu resurssien käytöstä johtuva kumulatiivinen hinta.



Kaavio 3: Resurssien käytöstä johtuva kumulatiivinen hinta. Selitteet ovat samat kuin kaaviossa 2. (Li et al., 2016)

Simulaation loppuajankohdasta voi havaita LRAIKMP:n tulleen halvimmaksi vaihtoehdoksi. Kaaviosta 3 voi havaita algoritmin olevan siis kustannustehokkain, vaikka simulointia kesti vaivaiset 70 minuuttia. Mikäli simulointia olisi jatkettu pidemmän aikaa tai simulaation ajankulkua olisi kiihdytetty, voisimme nähdä selkeämmän eron, joka havainnollistaisi kustannustehokkuutta pidemmältä ajanjaksolta. Pilvijärjestelmät ovat kuitenkin lähtökohtaisesti pitkäikäisiä, joten tulokset pidemmältä ajanjaksolta olisivat kiinnostavia.

Julkaisussa on viimeisenä esitelty kaavio, jossa on esitelty LRAIKMP:n, horisontaalisen skaalauksen ja vertikaalisen skaalauksen kertymäfunktiot.



Kaavio 4: LRAIKMP:n, horisontaalisen skaalauksen ja vertikaalisen skaalauksen kertymäfunktiot käyttöasteen suhteen. Selitteet ovat samat kuin kaaviossa 2. (Li et al., 2016)

Kaaviosta voi havaita LRAIKMP:n todennäköisimmäksi vaihtoehdoksi saavuttaa korkea käyttöaste. Tämä tarkoittaa sitä, että kyseinen algoritmi on korkeaa käyttöastetta tavoitellessa parempi vaihtoehto pelkän vertikaalisen tai horisontaalisen skaalauksen käyttöön verrattuna. Kun huomioidaan vielä kyseisen algoritmin tuottavan vähemmän virheitä kuin lineaarinen regressio tai eksponentiaalinen tasoitus, voi algoritmin todeta olevan edistyksellinen perinteisiin aikasarjaennustamismenetelmiin verrattuna. Voimme käyttöasteen, resurssienkäytön, hinnan ja ennustustarkkuuden perusteella todeta LRAIKMP:n olevan paras vaihtoehto verratuista.

3.2 Kuormituksen jakamiseen perustuva skaalaus

3.2.1 Kuormituksen jakamisen periaate ja tavoite

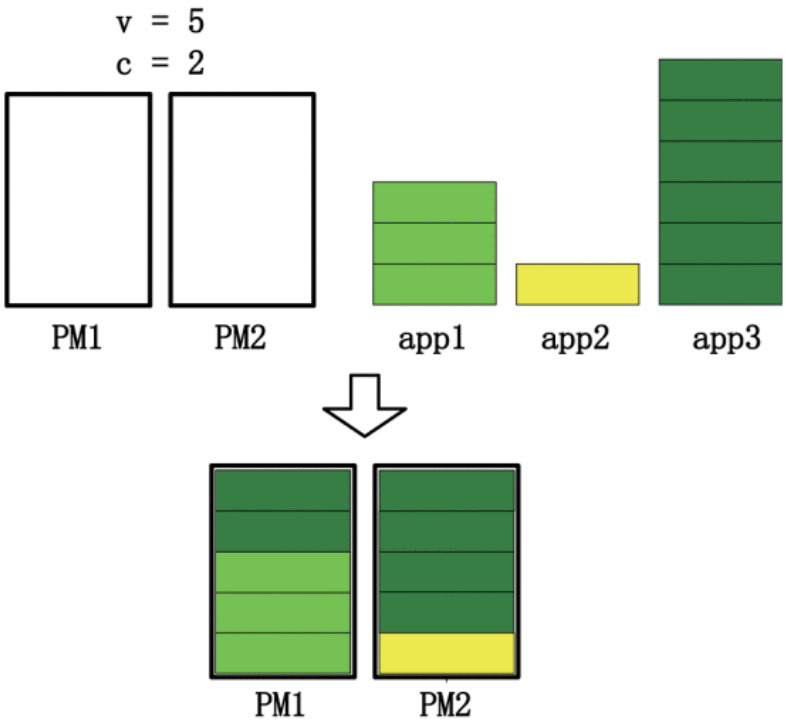
Zhen Xiao et al. esittävät julkaisussaan Automatic Scaling of Internet Applications for Cloud Computing Services (2012) mallin, joka skaalaa pilvipalvelusovelluksia automaattisesti pilvilaskentaympäristössä. Se on mallinnettu ratkaisuna englanniksi nk. Class Constrained Bin Packing (CCBP) -ongelmalle. Yksinkertaistettuna kyseessä on ongelma, jossa on käytössä vakiokokoisia säiliöitä, joihin koitetaan pakata asioita käyttämällä mahdollisimman vähän säiliöitä, eli tavoite on pakata asiat säiliöihin mahdollisimman tiiviisti. Kyseisen mallin säiliöt ovat pilvilaskentajärjestelmän palvelimia ja niihin pakattavat asiat ovat sovellusinstansseja. Korkeaan resurssien käyttöasteeseen ja energiatehokkuuteen päästään minimoimalla palvelimien määrää, jotka ovat käynnissä. Käyttämättömiä säiliöitä pidetään valmiustilassa, joita voi ottaa tarpeen mukaan käyttöön hyödyntämällä Wake-on-Lan-lähiverkkostandardia, joka mahdollistaa valmiustilassa olevien tietokoneiden nopean käynnistämisen lähiverkossa etänä.

Malli koostuu lähettäjästä, sovelluksen skedulointiliitännäisestä, sovelluspalvelimista ja datan säilytyspalvelimista. Lähettäjä vastaanottaa verkosta saapuvat asiakkaiden sovelluspyynnöt ja reitittää ne skedulointiliitännäiselle. Liitännäinen pitää yllä tietoa sovellusjakaumasta, palvelimien tilasta ja sen toiminta-algoritmin mukaisesta sovellusjoukon sijoittelusta. Palaamme itse toiminta-algoritmiin seuraavassa luvussa. Liitännäinen reitittää vastaanotetut sovelluspyynnöt ylläpitämän tiedon mukaisesti sovelluspalvelimille. Sovelluspalvelimet ovat itsessään tilattomia ja sovellusinstanssien dataa säilötään erikseen tallennustilapalvelimilla. Tilattomien sovelluspalvelimien tarkoituksena on parantaa toipumista virhetilanteista. Näin asiakassovelluksen istunto on mahdollista palauttaa nopeasti sovelluspalvelimen virhetilanteessa.

3.2.2 Kuormituksen jakamisen algoritmi ja toiminta

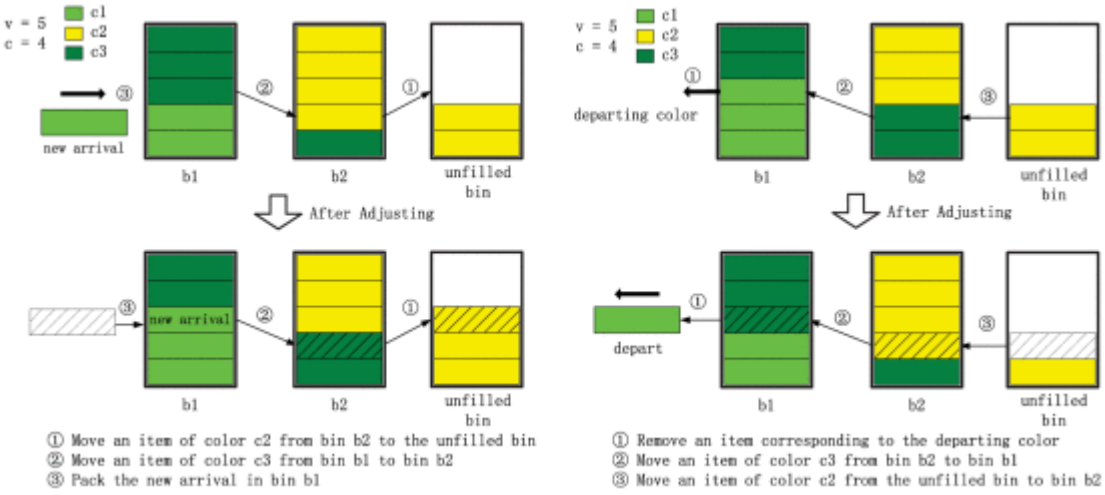
Skedulointialgoritmi reitittää pyynnöt hyödyntäen tietoa sovellusjakaumasta, joka on pyritty säätämään siten, että käynnissä olevien palvelimien sovellusinstanssit käyttäisivät mahdollisimman paljon palvelimien resursseista. Palvelimilla on rajoite sille, kuinka monta eri sovellusta niillä voi suorittaa. Rajoite johtuu siitä, että usean eri tyyppisen sovellusinstanssin suorittaminen samalla palvelimella aiheuttaa suuren tarpeen muistille.

Seuraavassa kaaviossa on havainnollistettu palvelimien täyttö. Kaavion merkintä v tarkoittaa kuinka monta kuormituksen yksikköä on palvelimen maksimikapasiteetti. Tässä tapauksessa yksikkö on 20% palvelimen kapasiteetista, koska yksikön koko on 100% jaettuna v:n lukumäärällä. Merkintä c ilmoittaa kuinka monta eri sovellusta palvelin voi suorittaa. PM tarkoittaa palvelinta ja app tarkoittaa sovellusta.



Kaavio 5: Palvelimien täyttäminen sovelluksilla. (Automatic Scaling of Internet Applications for Cloud Computing Services, Zhen Xiao et al., 2012)

Huomioitavaa on, että kaavion yksi väritetty kappale ei tarkoita yhtä sovellusinstanssia. Saman värin kappaleet muodostavat palvelimella yhden sovellusinstanssin vaatiman resurssien kokonaistarpeen. Siispä sovelluksella 3 on kaksi instanssia kyseisillä palvelimilla. Palvelimella 1 sovellusinstanssin resurssientarve on 40% kyseisen palvelimen resursseista ja palvelimella 2 sijaitsevan sovellusinstanssin tarve on 80% palvelimen resursseista. Seuraavassa kaaviossa on havainnollistettu, kuinka kuormituksen muutos käsitellään.



Kaavio 6: Palvelimilla tapahtuvan resurssien skaalauksen havainnollistus. (Zhen et al., 2012)

Kaavion 6 vasemmassa puoliskossa on havainnollistettu mallin toiminta tilanteessa, jossa käynnissä olevan sovellusinstanssin resurssien tarve kasvaa. Oikeassa puoliskossa on havainnollistettu tilanne, jossa resurssien tarve laskee.

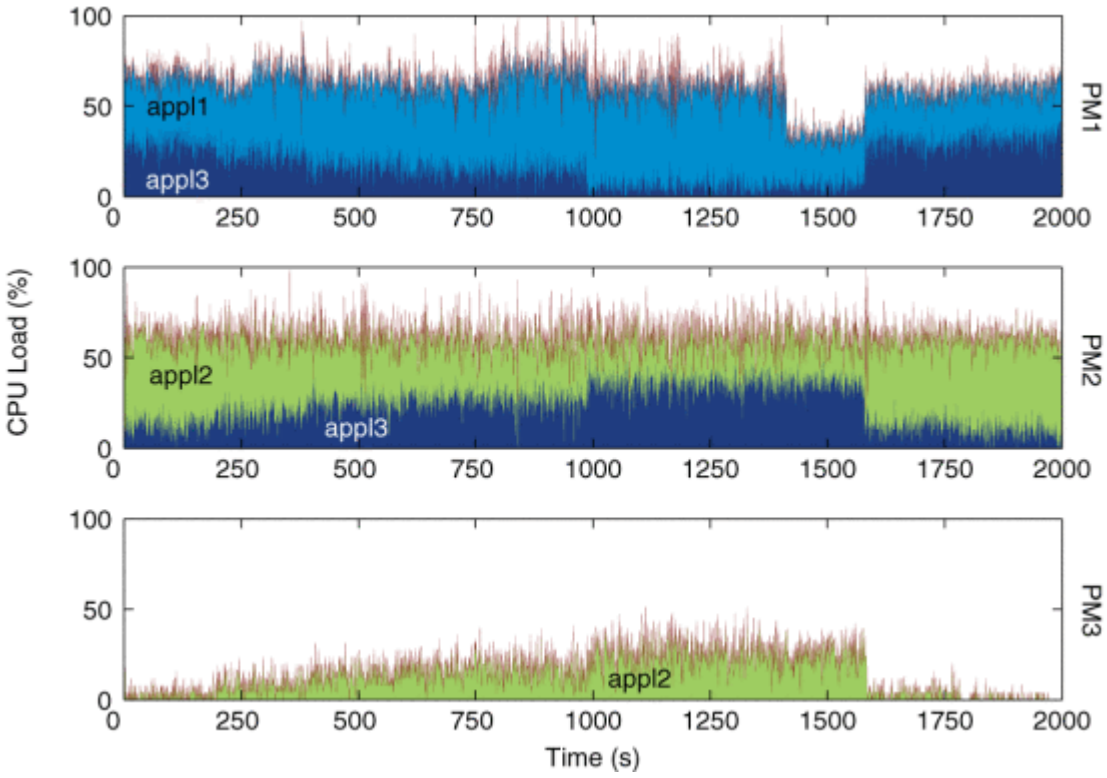
Kaavion vasemman puolen ylemmässä osassa kuvaa new arrival vaaleanvihreällä värillä tarkoittaa kasvanutta resurssien tarvetta sovellukselle c1. Tällainen tilanne kuvaa esimerkiksi kasvanutta käyttäjien määrää tai raskaan operaation alkua. Koska sovelluksella c1 on vain yksi instanssi käynnissä, eli sovellus on käynnissä vain yhdellä palvelimella, pyritään kyseiselle instanssille varaamaan lisää resursseja tällä palvelimella. Tämä onnistuu siten, että sovellukselle c3 varattuja resursseja vähennetään palvelimella b1 ja vastaavasti lisätään palvelimella b2. Sovellukselle c2 varattuja resursseja palvelimella b2 vähennetään ja vastaavasti nostetaan palvelimella b3. Lopputuloksena on tilanne, jossa sovelluksilla c2 ja c3 on käytettävissä yhtä paljon resursseja kuin alkutilanteessa, sekä sovellus c1 on skaalattu asianmukaisesti. Kaaviossa tämä näyttää siltä, että saapuvan vaaleanvihreän kappaleen edestä siirretään järjestyksessä kappaleita, jotta sille saataisiin tilaa. Huomioitavaa on, että ns. siirrot pyritään tekemään kappaleille kohdesäiliöihin, joista löytyy kyseistä väriä. Tämän saman värin löytymisen idea selitetään seuraavassa kappaleessa. On myös huomioitava, että mitään fyysistä siirtoa ei tapahdu, tapahtuu vain muutos siinä, kuinka paljon resursseja millekin sovellusinstanssille on eri palvelimilla varattu.

Idea siinä, että kohdesäiliöstä löytyy jo kyseisen sovelluksen instanssi eli kaavion 6 mukaisesti samaa väriä, on säästää aikaa ja resursseja. Tämä tarkoittaa sitä, että kohdesäiliössä on jo käynnissä kyseisen sovelluksen instanssi. Tällaisessa tilanteessa tarvitsee vain skaalata jo käynnissä olevien sovellusinstanssien käytettävissä olevia resursseja. Tilanteessa, jossa tällainen ketjuttaminen ei ole mahdollista, täytyy käynnistää uusi sovellusinstanssi. Siirtoja suositaan, koska uuden sovellusinstanssin käynnistäminen on hidas operaatio. Muutoin tehty skaalaus on nopea operaatio, joka ei laske palvelun laatua. Mikäli sovelluksen resurssientarve kasvaa tilanteessa, jossa kaikki palvelimet ovat täynnä, herätetään palvelin valmiustilasta ja käynnistetään uusi sovellusinstanssi. Palvelimen herättäminen valmiustilasta on myöskin hyvin nopea operaatio, joka ei itsessään laske palvelunlaatua.

Sovellusten resurssientarpeen vähentyessä, toimitaan samoin kuin lisääntyessä, mutta päinvastaisessa järjestyksessä. Palvelimet, joille ei ole millään sovelluksella tarvetta, asetetaan valmistilaan. Tilanteessa, jossa on kaksi tai useampi vajaa palvelin, pyritään niiden kuormituksia siirtelemään saadakseen täysiä palvelimia, vaikka se vaatisi uusien sovellusinstanssien käynnistämistä. Sovellusinstanssin käynnistäminen on hidas operaatio, mutta se on tehtävä tällaisessa tilanteessa, koska tavoitteena on säästää energiaa asettamalla tarpeettomia palvelimia valmistilaan.

3.2.3 Kuormituksen siirron ja kapasiteetin skaalauksen testaus

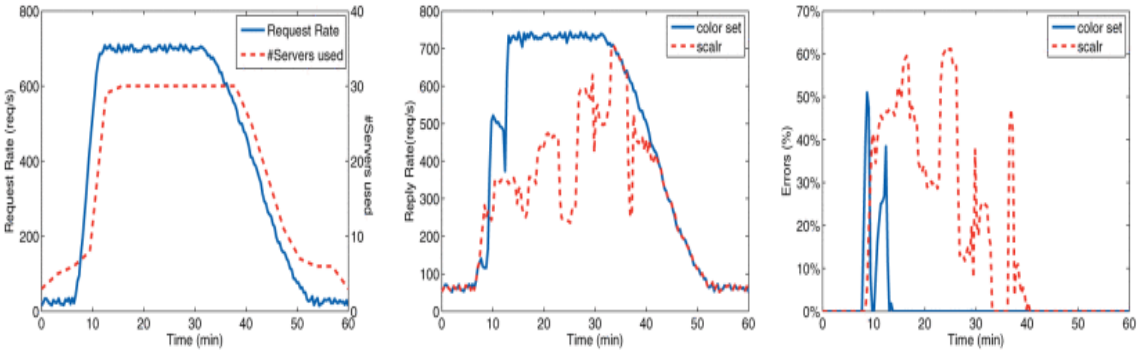
Mallin kuormituksen siirtoa on testattu kolmella palvelimella ja kolmella sovelluksella. Testauksessa yksittäinen palvelin oli varustettu Intel E5420 -prosessorilla ja 8 GB:n keskusmuistilla. Testi oli tarkoituksella pieni laajuudeltaan, koska tulokset voi yleistää eri tyyppisille palvelimille. Tulokset voi yleistää, koska erilaisten palvelimien suorituskykyparametrit eivät vaikuta merkittävästi itse siirtojen toimivuuteen. Seuraavassa kaaviossa on esitelty palvelimien prosessoreiden kuormitusasteiden vaihtelut 2000 sekuntia kestäneessä testissä.



Kaavio 7: Palvelimien prosessoreiden kuormitusasteet kuormituksen siirron testissä. (Zhen et al., 2012)

Testin alussa alettu lisätä sovelluksen 1 kuormitusta tasaisesti. Koska kuormituksen siirto ei ollut vielä mahdollista, on palvelimella 3 käynnistetty uusi sovelluksen 2 instanssi. Käynnistynyt instanssi mahdollisti palvelimen 1 sovelluksen 3 kuormituksen siirron palvelimelle 2, joka oli mahdollista, koska palvelimen 2 sovelluksen 2 kuormitusta voitiin nyt siirtää palvelimelle 3. Näin mahdollistui resurssien lisääminen sovelluksen 1 käyttöön palvelimella 1. Kun stabiili tila oli saavutettu, sovelluksen 1 kuormaa alettiin laskea ja käänteinen prosessi alkoi. On huomattavaa, kuinka 1450 sekunnin kohdalla palvelimen 1 prosessorin kuormituksen taso jää alhaiseksi noin 100 sekunnin ajan. Tämä johtuu siitä, että skeduloija ei kerää tietoa jatkuvasti sovellusjakaumasta kuormituksen muuttuessa, vaan jatkaa käyttäjien sovelluspyyntöjen reitittämistä palvelimille viimeisimmän tiedon mukaan. Skeduloijan tietoa sovellusjakaumasta päivitetään tietyn säädettävissä olevan ajanjakson välein, näitä ajanhetkiä kutsutaan päätöksentekohetkiksi, jolloin algoritmi muokkaa reititystaulua sovellusjakauman perusteella. Hieman ennen 1600 sekunnin ajankohtaa voi havaita tilan stabiloitumisen, mikä johtuu algoritmin päätöksentekohetkestä.

Automaattista kapasiteetin skaalausta on testattu yhdeksällä sovelluksella ja 30 palvelimella. Yksittäinen palvelin on varustettu Intel E5620 -prosessorilla ja 24 GB:n keskusmuistilla. Mallia on verrattu avoimen lähdekoodin toteutusversioon Amazon EC2:n käyttämästä Scalr-skaalausalgoritmista. Testin alussa nostettiin yhden sovelluksen kuormitusta rajusti. Kuormituksen raju kasvu kuvaa esimerkiksi hyvin raskaan operaation alkua, usean samanaikaisen operaation alkua tai vaikkapa suurta käyttäjämäärän kasvua. Seuraavassa kaaviossa nähdään testauksen tulokset.



Kaavio 8: Kapasiteetin skaalauksen testauksen tulokset. (Zhen et al., 2012)

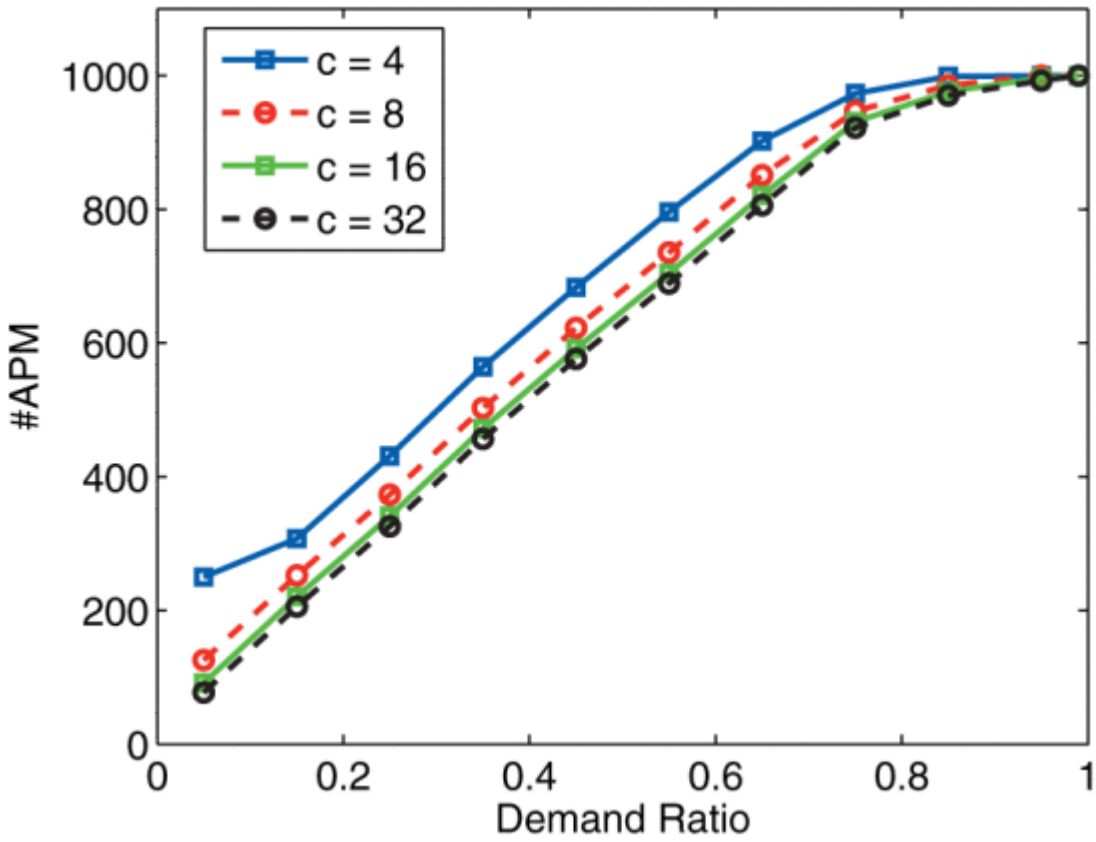
Kaavion 8 vasemmassa viivakaaviossa on sinisellä merkattu sovelluspyyntöjen määrä sekunnissa ajan funktiona ja punaisella käytettyjen palvelimien määrä kyseiselle mallille. Keskimmäisessä viivakaaviossa sinisellä on merkattu kyseisen mallin sovelluspyyntöihin lähetettyjen vastausten määrä sekuntia kohden ajan funktiona ja punaisella Scalr:n lähettämien vastausten määrä. Oikean puoleisessa viivakaaviossa on sinisellä merkattu käsittelemämme mallin virheprosentti ajan funktiona ja Scalr:n punaisella.

Sovelluspyyntöjen kasvaessa malli ottaa käyttöön kaikki 30 palvelinta. Samalla voi havaita nopean kasvun vastausten taajuudessa. Vastausten taajuuden kasvun yhteydessä voi havaita mallin kohdalla virhettä, mistä palaudutaan viidessä minuutissa. Scalr alkaa myös samoihin aikoihin kärsimään virheistä ja kärsii niistä vielä 25 minuutin jälkeen. Kyseinen malli vaikuttaisi suoriutuvan hyvin jopa näin vaativassa tilanteessa. Riippuen mahdollisen QoS:n määritelmästä SLA:ssa, voi paremman laadun luvata kyseisen mallin mukaisella skaalauksella, kuin Amazonin käyttämän Scalr:n skaalauksella. On kuitenkin huomattava, että testi on tehty 2012 ja tulokset voisivat näyttää erilaisilta, jos testi tehtäisiin tänä päivänä.

3.2.4 Toimivuuden testaus simulaatiossa

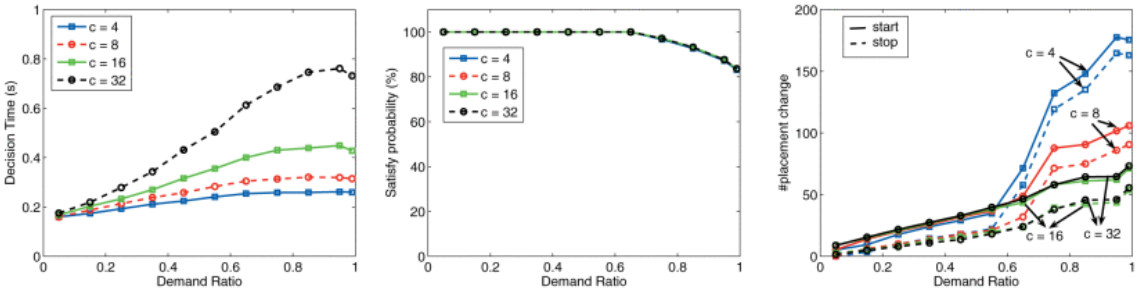
Isolla skaalalla toimivuutta on testattu simuloimalla suurta palvelinjoukkoa ja sovellusten määrää käyttäen samaa koodipohjaa mitä testauksissa käytettiin. Seuraavaksi määrittelemme kyseisen simulaation rajoitteita ja sääntöjä. Simulaatiolle on annettu tieto palvelimien kapasiteetista, sekä sovellusinstanssien muistin ja prosessorin tarpeista. Palvelimien oletettiin olevan homogeenisia. Kysyntäsuhde D on määritelty sovellusten kokonaisresurssientarpeen ja palvelimien kokonaiskapasiteetin välillä. Esimerkiksi D:n ollessa 90 %, olisi palvelimien käyttöaste oltava 90 %, jotta kaikkien sovellusten tarpeet olisi tyydytetty. CPU:n tarve varattiin suhteessa sovellusten normalisoituun painoarvoon. Sovellusten käytettävissä oleva muisti on rajattu jakamalla palvelimen muisti sillä ajettavien erilaisten sovellusten lukumäärällä tasan. Sovellusten yhteenlaskettu prosessointikapasiteetin tarve saatiin kertomalla D sekä palvelimien kokonaisprosessointikapasiteetti. Jokainen ajettava sovellus valitsi satunnaisen liukuluvun nollan ja yhden välillä painoarvoksi. Jokaisena hetkenä, kun skeduloija keräsi tietoa sovellusjakaumasta ja muutti reititystä sen mukaisesti, muutettiin samalla satunnaisten sovellusten kysyntää 20%. Mikäli palvelimella oli yksikin sovellusinstanssi suorituksessa, se luokiteltiin aktiiviseksi fyysiseksi palvelimeksi, joka on englanniksi Active Physical Machine, josta käytetään lyhennettä APM. Jos palvelimella ei ollut sovellusinstanssia suorituksessa, se asetettiin lepotilaan energiansäästämiseksi. Simulaatiossa skedulointialgoritmille annettiin valmisteltu syöte, mikä kuvasi palvelinpyyntöjä. Simulaatio ylläpiti tietoa järjestelmän tilasta ja laski mittarit kuten APM:ien määrä järjestelmässä, skedulointialgoritmin päätöksentekohetket, tarpeen tyydytyssuhteet ja sovellusjakauman muutosten määrän. Testit toistettiin 200 kertaa samoilla syötteillä, joista tuloksiin poimittiin keskiarvo.

Ensimmäinen simulaatio suoritettiin tuhannen palvelimen ja tuhannen sovelluksen joukolla. Simulaation alussa D:n arvoa alettiin nostaa tasaisesti 0,05:stä 0,99:ään. Seuraavassa kaaviossa on havainnollistettu kuinka APM:ien määrä muuttuu D:n muuttuessa. Kaaviossa on nähtävissä tulokset neljälle eri c:n arvolle, mikä tarkoitti rajausta sille, kuinka monta erityyppistä sovellusinstanssia yksi palvelin saa suorittaa.



Kaavio 9: APM:n määrän muuttuminen D:n suhteen (Zhen et al., 2012)

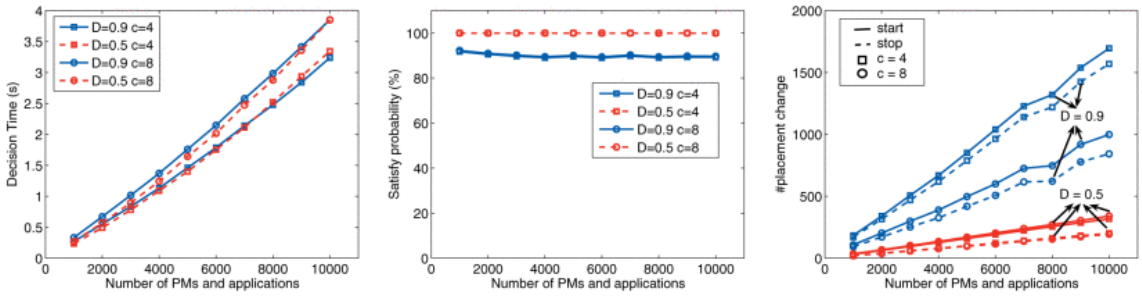
Kaaviosta APM:n voi huomata saavuttavan 100 % hieman sen jälkeen, kun D ohittaa arvon 0,8. Suuremmalla c:n arvolla vaikuttaisi olevan pienentävä vaikutus APM:n määrään ja pienemmällä taas suurentava vaikutus. Seuraavassa kaaviossa nähdään vasemmalla D:n vaikutus eri c:n arvoilla skeduloijan päätöksentekohetken pituuteen, keskellä pyyntöjen tyydytyssuhteeseen ja oikealla sovellusten siirtojen määrään. Siirrot on eritelty instanssien käynnistyksiin jatkuvalla viivalla ja lopetuksiin katkoviivalla.



Kaavio 10: D:n vaikutus päätöksentekohetken pituuteen, pyyntöjen tyydytyssuhteeseen ja kuormituksen siirron määrään (Zhen et al., 2012)

Kaavion 10 vasemmassa viivakaaviossa voi havaita, kuinka suurempi c:n arvo pidentää skeduloijan päätöksentekohetkeä. Huonoimmillaan kuitenkin päätöksentekohetki pysyy alle 0,8 sekunnin, joka on nopea näinkin isolle järjestelmälle. Paras päätöksentekohetken pituus saavutetaan pienellä c:n arvolla. Keskimmäisen viivadiagrammin perusteella voi todeta, ettei testatuilla c:n arvoilla ole vaikutusta käyttäjien sovelluspyyntöihin vastaamiseen. Kysynnän tyydytyssuhde pysyy 100%:ssa kunnes D ylittää 0,65:n, jonka jälkeen tyydytyssuhde laskee hieman. Tämä viittaa siihen, ettei algoritmi ole herkkä c:n arvolle. Oikeanpuoleisen viivakaavion perusteella voimme todeta, että suuremmalla c:n arvolla vähennetään tarvetta siirtää kuormitusta, sekä tarvetta käynnistää ja sammuttaa sovelluksia. Kun palvelimilla on käynnissä useampia sovelluksia, vähenee tarve käynnistää uusia sovelluksia, koska on useampia vaihtoehtoja toteuttaa kuormituksen siirto.

Skaalautuvuutta on arvioitu nostamalla sovelluksien ja palvelimien määrää 1000:sta 10000:een. Seuraavassa kaaviossa on esitelty kuinka palvelinkoneiden ja sovellusten määrä vaikuttaa päätöksentekohetken pituuteen (vasen viivakaavio), tyydytyssuhteeseen (keskimmäinen viivakaavio) ja sovellusten siirtoihin (oikeanpuoleinen viivakaavio).

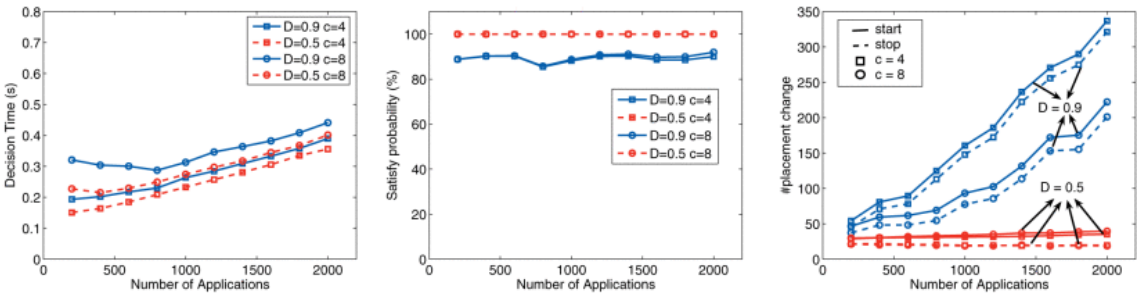


Kaavio 11: Palvelinkoneiden ja sovelluksien määrän vaikutus päätöksentekohetken pituuteen, pyyntöjen tyydytyssuhteeseen ja kuormituksen siirron määrään eri D:n arvoilla (Zhen et al., 2012)

Vaikka sovellusten ja palvelinkoneiden määrä nousee kymmeneen tuhanteen, pysyy skeduloijan päätöksentekohetken pituus alle neljän sekunnin. Tyydytyssuhde pysyy korkeana eri D:n ja c:n arvoilla, mikä on tärkeää isolla skaalalla. Voi jälleen huomata, että isommalla c:n arvolla tarvitsee tehdä vähemmän sovellusten siirtoja. Tämän perusteella voimme todeta mallin olevan hyvin skaalautuva.

Seuraavassa simulaatiossa on testattu mallin toimivuutta säätelemällä sovelluksien ja palvelimien suhdetta. Palvelimia oli käytössä 1000 ja sovellusten määrää kasvatettiin

200:sta 2000:een. Seuraavassa kaaviossa on nähtävissä tulokset sovellusmäärän suhteen.



Kaavio 12: Sovelluksien määrän vaikutus päätöksentekohetken pituuteen, pyyntöjen tyydytyssuhteeseen ja kuormituksen siirron määrään eri D:n ja c:n arvoilla (Zhen et al., 2012)

Sovellusten määrä vaikuttaa skeduloijan päätöksentekohetken pituuteen, mutta se pysyy alle 0,5 sekunnin jopa 2000:lla sovelluksella kaikilla testatuilla D:n ja c:n arvoilla. Palvelupyyntöjen tyydytyssuhde ei kärsi korkeasta määrästä sovelluksia, joten palvelunlaatu ei suurestakaan määrästä sovelluksia laske huomattavasti. D:n ollessa korkea, aiheutuu useampia sovelluksien siirtoja. Suuremmalla c:n arvolla on enemmän sovellusinstansseja ajossa sovelluksia kohden, joten sovelluksien siirtoja tarvitsee tehdä vähemmän. Kysynnän ollessa pieni, sovellusten lukumäärällä ja c:n arvolla on pienempi vaikutus tarvittavien siirtojen määrään.

Simulaation perusteella malli vaikuttaisi skaalautuvan todella hyvin. Se tarjoaa oikeanlaisilla asetuksilla skaalautuvuutta, joka ei huononna palvelunlaatua. Malli tarjoaa myös hyvän käyttöasteen säätelemällä käytettävissä olevaa kapasiteettia. Jatkoa ajatellen, olisi mielenkiintoista nähdä, kuinka loppukäyttäjän kannalta tämän mallin käyttö eroaisi esimerkiksi Amazon EC2:n käyttämästä Scalr-implementaatiosta kysynnän ollessa korkea. Myös tämän mallin vaikutus konesalin kustannuksiin olisi mielenkiintoinen tieto.

3.3 Palvelimen tilaan perustuva autonominen skaalausmalli

3.3.1 Mallin rakenne ja toimintaperiaate

Al-Sharif Z.A. et al. esittävät julkaisussaan ACCRS: autonomic based cloud computing resource scaling (2016) autonomisen resurssien skaalausmallin. Autonomisuudella pyritään siihen, että skaalaus toimisi mahdollisimman vähällä ihmisen vuorovaikutuksella. Malli on suunniteltu huomaamaan virhetilanteet ja toipumaan niistä itsestään, mikä vähentää tarvetta ihmisen vuorovaikutukselle. Malli säätelee resurssienkäyttöä pyrkien hyvään käyttöasteeseen sekä vastaamaan SLA:ssa määriteltyä QoS:ssää. Resurssienkäyttöä säädellään tarkastelemalla palvelimien tilaa ja tekemällä päätökset tilatiedon perusteella.

Mallin mukainen järjestelmä on jaettu viiteen osatekijään: 1. System State Monitoring (SSM) eli järjestelmän tilan valvoja. Valvoja kerää tietoa prosessoreiden tilasta, keskusmuistin tilasta, kaistanleveyden tilasta, suoritustehosta ja virrankulutuksesta. 2. Decision Making Algorithm (DMA) eli päätöksentekoalgoritmi. DMA prosessoi SSM:n keräämää tietoa ja tekee päätöksen resurssien skaalauksesta kerätyn tiedon perusteella. DMA pyrkii pitämään prosessorin käyttöasteen 70 %:n ja 80 %:n välillä, keskusmuistin käytön 70 %:n ja 86 %:n välillä sekä kaistanleveyden 50 %:n ja 63 %:n välillä. Alarajan määrittely tässä tutkielmassa perustuu olettamukseen, että julkaisussa on tehty virhe, joka ilmenee tarkastellessa kuormaa arvioivaa algoritmia. Algoritmi on esitelty seuraavassa kuvassa.

Algorithm 2 System State Analyses and Workload Classification

```
1: procedure SSA- WCA
2:     cpu ← current level of CPU
3:     ram ← current level of RAM
4:     bw ← current level of BandWidth
5:
6:     ▷ CPU, RAM, & Bandwidth Thresholds are: 80%, 86%, & 63%, respectively
7:     if cpu > 80% ∨ ram > 86% ∨ bw > 63% then return HEAVY
8:     else if cpu < 70% ∨ ram < 70% ∨ bw < 50% then return HEAVY
9:     else return SafeZone
```

Kuva 1: Työkuorman arvioiva algoritmi (ACCRS: autonomic based cloud computing resource scaling, Al-Sharif Z.A. et al., 2016)

DMA:n algoritmi tekee päätöksen resurssien skaalauksessa yllä olevan algoritmin mukaisesti. Yllä olevan algoritmin rivillä 8 palautetaan ehdon mukaisesti tieto raskaasta kuormasta eli pseudokoodin mukaisesti palautetaan arvo HEAVY. Olisi loogista, että näillä ehdoilla palautetaan arvo kevyestä kuormasta eli LIGHT. Esitellyssä algoritmissa ei ole ehtoa, jonka mukaan palautettaisiin arvo LIGHT, vaikka DMA:n algoritmi on varautunut kevyeen kuorman tietoon, jolloin käytettävissä olevia resursseja vähennetään.

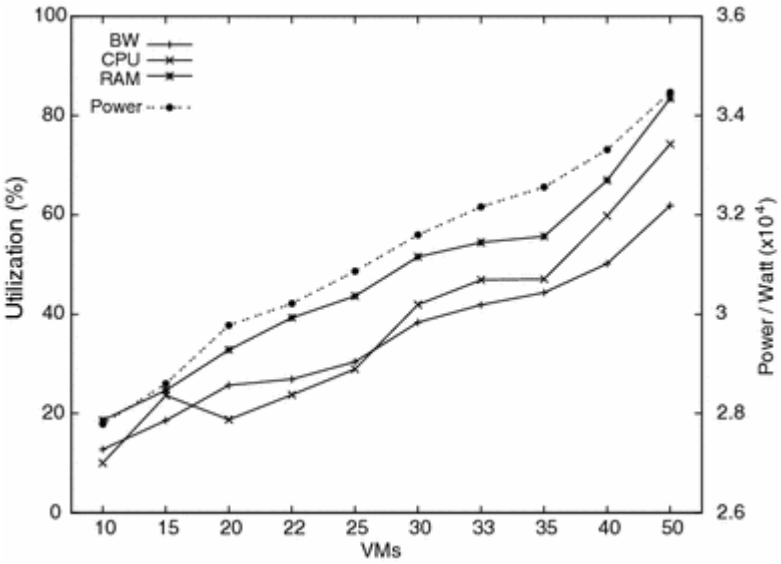
Kyseiset rajat resurssienkäytölle on todettu kokeellisesti optimaalisiksi. Näiden rajojen mukaisesti työkuorma jaetaan kevyeen tai raskaaseen työkuormaan. Raskaan työkuorman tapauksessa käytettävissä olevia resursseja lisätään, jotta käyttöaste laskisi tavoitellulle tasolle ja kevyen kuorman tapauksessa resursseja vähennetään käyttöasteen nostamiseksi. Erillinen algoritmi ennakoi vähennettävien tai lisättävien resurssien määrää laskien tarpeen sen hetkisen tilan perusteella ennen skaalausta. 3. Multi-Level Resource Scaling, joka säätelee resurssien käyttöä. 4. Host-level resource scaling (H-LRS) eli isäntätason resurssien skaalain. Järjestelmän isännät koostuvat niiden hallinnoimista virtuaalikoneista. Optimaalinen määrä virtuaalikoneita isäntää kohden on todistettu kokeellisesti olevan 5 – 6. H-LRS säätelee sen hallinnoiman virtuaalikoneiden määrää ja on siis kuin horisontaalinen skaalain, jonka toiminta rajoittuu pienelle määrälle virtuaalikoneita. 5. VM-Level Resource Scaling (VM-LRS) vastaa yksittäisen virtuaalikoneen resurssien skaalauksesta ja on siis vertikaalinen skaalain.

3.3.2 Mallin testaus simuloidussa ympäristössä

Mallin toimivuutta on testattu CloudExp-työkalulla, joka on rakennettu CloudSim:n päälle. Simulaatiossa järjestelmälle syötettiin työmäärä, joka on tuotettu Rain-nimisellä työkalusarjalla, jolla voi tuottaa vaihtelevia työmääriä. Simulaation käyttäjille osoitetaan oma työmäärän tuottaja. Simulaation isäntien käyttöön on varattu 4,5 GB keskusmuistia, tuplaydinprosessori ja miljoona yksikköä kaistanleveyttä. Yhdelle virtuaalikoneelle on varattu satunnainen määrä keskusmuistia 512 – 1024 MB:n väliltä, virtuaalinen yksiytiminen prosessori ja satunnainen määrä kaistanleveyttä 100 000 – 150 000 yksikön väliltä. Testiympäristö koostui kymmenestä isännästä, joiden käytettävissä oli kiinteä määrä resursseja.

3.3.3 Testin tulokset

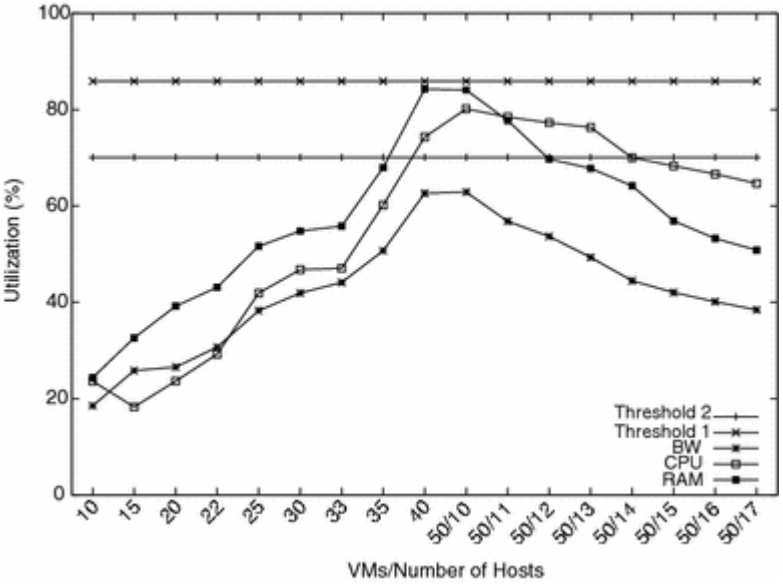
Seuraavassa kaaviossa on esitetty tulokset käyttöasteen, sähkötehon ja virtuaalikoneiden määrän suhteesta kasvavalla työkuormalla ilman mallin mukaista skaalausta. Utilization tarkoittaa käyttöastetta prosentteina, VMs tarkoittaa virtuaalikoneiden määrää ja Power / Watt tarkoittaa sähkötehoa wateissa 10^4-kertaisena. Kaavion merkintä BW tarkoittaa kaistanleveyttä, CPU prosessoria, RAM keskusmuistia ja Power sähkötehoa.



Kaavio 13: Kasvavan työkuorman vaikutus käyttöasteeseen, virrankulutukseen ja virtuaalikoneiden määrään (Al-Sharif Z.A. et al., 2016)

Ensimmäisen testin tarkoitus oli havainnollistaa kuinka kasvava työkuorma aiheuttaa kasvun sähkötehossa ja käyttöasteen kasvun. Kaaviosta voi havaita, että suuremmalla käyttöasteella tarvittava sähköteho kasvaa. Myös virtuaalikoneiden määrää on lisätty käyttöasteen noustessa.

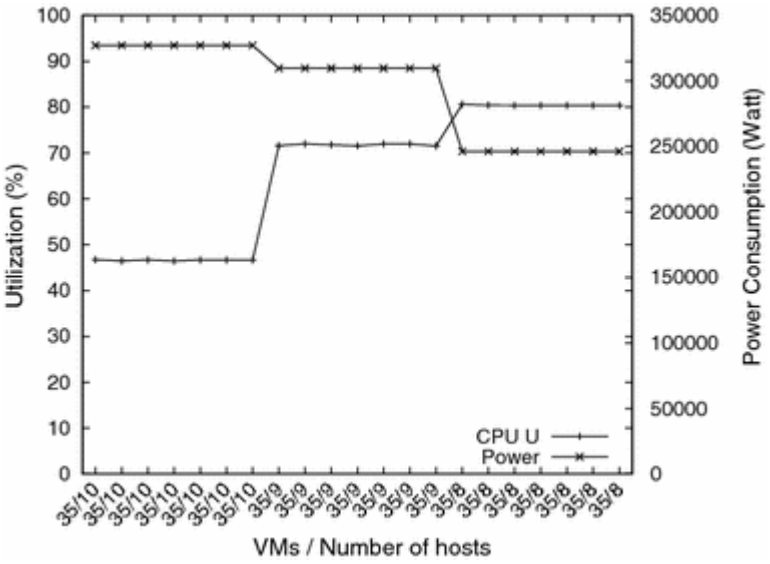
Seuraavassa testissä skaalaus on aktivoitu ja työkuorman määrää on lisätty vakiomäärällä isäntiä. Skaalauksen vaikutus kasvavalla työkuormalla käyttöasteeseen on esitetty seuraavassa kaaviossa. Kaavioon merkattu Threshold 2 tarkoittaa käyttöasteen turvallisen alueen alarajaa ja Threshold 1 ylärajaa.



Kaavio 14: Skaalauksen vaikutus käyttöasteeseen kasvavalla työkuormalla (Al-Sharif Z.A. et al., 2016)

Vaikkakin julkaisussa on ilmoitettu, että kyseessä on testi vakiomääräisillä isännillä, on epäselvää, kuinka monta isäntää testissä käytettiin. Virtuaalikoneiden määrä ilman isäntien määrää on ilmoitettu 40 virtuaalikoneeseen asti, jonka jälkeen isäntien määrä onkin ilmoitettu ja kasvaa. Kaaviosta voi kuitenkin havaita, että käyttöasteen noustessa korkeaksi, skaalain lisää resurssien määrää, jolloin käyttöaste palaa takaisin turvalliselle alueelle.

Seuraavaksi on testattu kuinka malli skaalaa resurssien käyttöä alhaisessa työkuormassa käyttöasteen parantamiseksi. Seuraavassa kaaviossa on esitetty testin tulokset prosessorin käyttöasteen kannalta.



Kaavio 15: Skaalauksen vaikutus prosessorin käyttöasteeseen alhaisella työkuormalla (Al-Sharif Z.A. et al., 2016)

Kuvasta voi havaita kuinka virtuaalikoneiden määrää ja isäntien suhdetta säätämällä saavutetaan korkeampi käyttöaste ja pienempi sähköteho, joka vähentää virrankulutusta. Paras tila saavutetaan tässä tilanteessa säätämällä isäntien määrää kahdeksaan, jonka jälkeen isäntien määrää ei kannata laskea, koska prosessorin käyttöaste ylittäisi määritellyn turvallisen rajan.

Malli onnistuu skaalaamaan resursseja testien perusteella. Mallin mukaisella skaalauksella voi dynaamisesti skaalata resursseja, jolla saadaan aikaan haluttu käyttöaste. Riippuen määritellystä SLA:sta ja siinä määritellystä QoS:stä voi turvallisen kuormituksen rajoja säätää päästäkseen haluttuun lopputulokseen.

4. YHTEENVETO

Pilvilaskennan resurssien skaalausta voi lähestyä eri tavoin, mutta tavoitteet ovat samat. Tavoitteena on korkea resurssien käyttöaste laadusta karsimatta, mikä vähentää sähkönkulutusta ja pienentää kustannuksia. SLA:ssa määritellyn QoS:n määrittely asettaa tavoitteet laadulle. Pienemmällä sähkönkulutuksella on myös vaikutus hiilijalanjälkeen.

Tässä tutkielmassa on esitelty kolme erilaista skaalausmallia, joissa skaalausta on lähestytty ennakoiden ja reaktiivisesti, sekä horisontaalisen että vertikaalisen skaalauksen tavoin. Kaikissa näissä malleissa on skaalauksella onnistuttu parantamaan käyttöastetta vertailukohteisiin nähden. Vaikka mallit ovat jo tämän tutkielman aikaan muutamia vuosia vanhoja ja vanhin vuodelta 2012, on kyseisissä malleissa nähtävissä samanlaisia lähestymistapoja ja tavoitteita kuin tuoreimmissa malleissa. Esimerkiksi julkaisussa HAS: Hybrid auto-scaler for resource scaling in cloud environment (Bibal Benifa J.V. & Dharma D., 2018), yhdistyy ennakoiva ja reaktiivinen menetelmä skaalaukseen. Julkaisussa Performance modelling and verification of cloud-based auto-scaling policies (Evangelidis A. et al., 2018), tavoitellaan QoS:n tyydyttämistä dynaamisella resurssien skaalauksella. Julkaisussa VMDFS: virtual machine dynamic frequency scaling framework in cloud computing (Shojaei K., et al., 2018), tavoitellaan sähkönkulutuksen pienentämistä ja SLA:n ehtojen rikkomisen vähentämistä säätämällä dynaamisesti virtuaalikoneiden prosessoreiden laskentatehoa. Mallit, joihin tässä tutkielmassa on tutustuttu, ovat edistyksellisiä ja niissä on elementtejä, jotka voisivat täydentää toisiaan. Kuormitusta ennustava skaalaus hyötyisi tarkemmasta reaktiivisesti skaalauksesta, joka on esitelty palvelimen tilaan perustuvassa skaalauksessa. Kuormituksen jakamiseen perustuva skaalaus hyötyisi ennakoivasta algoritmista, joka luultavasti vähentäisi virhettä äkillisessä kuormituksen kasvussa, kuten hyötyisi myös palvelimen tilaan perustuva skaalaus. Palvelimen tilaan perustuva skaalaus hyötyisi myös tarkemmasta skaalauksesta, joka skaalaisi eri resursseja itsenäisinä osina.

Hyvä skaalausmalli on varautunut äkillisiin kuormituksen kasvuihin, parantaa käyttöastetta, vähentää kustannuksia, toipuu virheistä ja minimoi niitä, sekä parantaa yrityksien kilpailukykyä. Esitellyissä malleissa on hyvien skaalausalgoritmien piirteitä, mutta parempi malli hyödyntäisi piirteitä näistä kaikista. Tällainen suuntaus skaalauksessa lienee pilvilaskennan tulevaisuus, kun datamäärät ja pilvipalvelut jatkavat kasvamista.

LÄHTEET

Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.M. and Vasilakos, A.V., 2015. Cloud computing: Survey on energy efficiency. *Acm computing surveys (csur)*, *47*(2), p.3.

Barroso, L.A. and Hölzle, U., 2007. The case for energy-proportional computing.

Dayarathna, M., Wen, Y. and Fan, R., 2015. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, *18*(1), pp.1.

Xiao, Z., Chen, Q. and Luo, H., 2012. Automatic scaling of internet applications for cloud computing services. *IEEE Transactions on Computers*, *63*(5), pp.1111-1123.

Li, T., Wang, J., Li, W., Xu, T. and Qi, Q., 2016, July. Load prediction-based automatic scaling cloud computing. In *2016 International Conference on Networking and Network Applications (NaNA)* (pp. 330-335). IEEE.

Saravanan, K. and Rajaram, M., 2015. An Exploratory Study of Cloud Service Level Agreements-State of the Art Review. *KSII Transactions on Internet & Information Systems*, *9*(3).

Al-Sharif, Z.A., Jararweh, Y., Al-Dahoud, A. and Alawneh, L.M., 2017. ACCRS: autonomic based cloud computing resource scaling. *Cluster Computing*, *20*(3), pp.2479-2488.

JV, B.B. and Dharma, D., 2018. HAS: Hybrid auto-scaler for resource scaling in cloud environment. *Journal of Parallel and Distributed Computing*, *120*, pp.1-15.

Evangelidis, A., Parker, D. and Bahsoon, R., 2018. Performance modelling and verification of cloud-based auto-scaling policies. *Future Generation Computer Systems*, *87*, pp.629-638.

Shojaei, K., Safi-Esfahani, F. and Ayat, S., 2018. VMDFS: virtual machine dynamic frequency scaling framework in cloud computing. *The Journal of Supercomputing*, *74*(11), pp.5944-5979.

Fossil fuel energy consumption (% of total). The World Bank. https://data.worldbank.org/indicator/eg.use.comm.fo.zs Luettu: 14.5.2019

Costello K., Meulen R., 2018. Gartner Says Worldwide IaaS Public Cloud Services Grew 29.5 Percent in 2017. https://www.gartner.com/en/newsroom/press-releases/2018-08-01-gartner-says-worldwide-iaas-public-cloud-services-market-grew-30-percent-in-2017 Luettu: 14.5.2019

Costello K, 2019. Gartner Says Nearly 50 Percent of Paas Offerings Are Now Cloud-Only. https://www.gartner.com/en/newsroom/press-releases/2019-02-27-gartner-says-nearly-50-percent-of-paas-offerings-are- Luettu 14.5.2019

Hall C., 2019. Google to Spend $13B on US Data Center and Office Construction This Year. DataCenter Knowledge.
https://www.datacenterknowledge.com/google-alphabet/google-spend-13b-us-data-center-and-office-construction-year Luettu: 14.5.2019

Columbus L., 2018. Roundup of Cloud Computing Forecasts And Market Estimates, 2018. https://www.forbes.com/sites/louiscolumbus/2018/09/23/roundup-of-cloud-computing-forecasts-and-market-estimates-2018/#166a09d2507b Luettu: 14.5.2019

RENO, NV, 2018. Quarterly SaaS Spending Reaches $20 billion as Microsoft Extends its Market Leadership. Synergy Research Group.
https://www.srgresearch.com/articles/quarterly-saas-spending-reaches-20-billion-microsoft-extends-its-market-leadership Luettu: 14.5.2019

Microsoft Cloud. 2.11.2018. Microsoft Global Datacenters and Network Infrastructure. Haettu: https://www.youtube.com/watch?v=s5I4wcQ6n0g