Tampereen yliopisto

Frank Herrala

WEB-PALVELINYMPÄRISTÖJEN VEDENMITTAUSTIEDON KÄSITTELYYN KÄYTETTYJEN TOTEUTUKSIEN VERTAILU

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Huhtikuu 2020

TIIVISTELMÄ

Frank Herrala: Web-palvelinympäristöjen vedenmittaustiedon käsittelyyn käytettyjen toteutuksien vertailu (Comparing server-side implementations for handling waterconsumption data)
Kandidaatintyö
Tampereen yliopisto
Tietotekniikka
Huhtikuu 2020

Tutkimuksen tehtävänä on selvittää, kumpi vertailtavista palvelinpuolista soveltuu paremmin vedenmittausdatan tallentamiseen relaatiotietokantaan. Vertailtavat palvelinpuolet ovat Node.js ja PHP+Apache. Tietokannan hallintajärjestelmänä toimi MySQL. Toteutuksia testattiin lähettämällä niille HTTP-viestejä, joiden tiedot niiden tehtävänä oli tallentaa tietokantaan. Viestejä lähetettiin eri määriä samanaikaisesti, jotta selviäisi, kuinka hyvin palvelinpuolet selviävät samainaikaisuuden kasvaessa. Tutkimuksessa PHP+Apache selviytyi paremmin korkeista samanaikaisuuden tasoista johtuen sen käyttämistä tuhannesta säikeestä, kun taas Node toimi yhdellä säikeellä. Node oli kuitenkin pienemmillä samanaikaisuuden tasoilla nopeampi ja vaati huomattavasti vähemmän muistia ja prosessointi tehoa kaikilla tasoilla.

Avainsanat: Node, Node.js, PHP, Apache, palvelinpuoli

SISÄLLYSLUETTELO

1 JOHDANTO

Tutkimuksessa tarkastellaan kahta vaihtoehtoa vedenmittauksesta saadun tiedon tallentamiseen. Vedenmittauslaite mittaa vedenkulutusta vedenjakamossa ja tallentaa aina ajan, jonka kuluessa vettä on mennyt tietty määrä. Nämä tiedot on tarkoituksena lähettää web-palvelimelle, jonne ne voidaan tallentaa ja josta niitä voidaan tarkastella. Tarkoituksena on selvittää, kumpi kahdesta eri palvelinpuolen ympäristöstä pystyy käsittelemään tehokkaammin vedenmittauslaittesta lähettyjä tietoja. Vertailtavina ympäristöinä ovat Node.js ja PHP, joka pyörii Apache web-palvelimen päällä. Molemmat ympäristöt käyttävät MySQL-tietokannan hallintajärjestelmää. Vertailussa lähetetään HTTP-viestejä, jotka tallennetaan MySQL-tietokantaan. HTTP-viestejä lähetetään eri määriä rinnakkain tehokkuuden selvittämiseksi. Työssä käydään läpi aluksi palvelinpuolen toimintaa ja teknologiaa, sekä avataan aikaisempia tutkimuksia Noden ja PHP+Apachen tehokkuudesta. Tämän jälkeen käydään läpi, miten eri toteutuksia testataan ja verrataan keskenään. Kerrotaan myös miten eri toteutukset on pohjustettu testausta varten. Lopussa tarkastellaan tutkimuksen tuloksia ja tarkastellaan, mistä nämä johtuvat.

2 WEB-PALVELINYMPÄRISTÖN TAUSTAA

Tässä luvussa perehdytään tutkimuksen kannalta olennaisiin teknologioihin ja metodologiaan. Palvelinpuolen tehokkuus on tutkimuksen kohteena, joten on tarpeellista käydä läpi mitä teknologioita tämä sisältää. Web-palvelimen tehtävä käydään lävitse, koska se ohjaa HTTP-viestejä oikeisiin osotteisiin. Palvelinpuolenympäristöt käydään läpi, koska nämä käsittelevät HTTP-pyyntöjen sisällön ja luovat vastauksen kyseisiin pyyntöihin. Luvussa käydään myös läpi MySQL-tietokannan hallintajärjestelmä, jota tarvitaan tiedon tallentamiseen web-palvelimelle. Molemmat toteutukset, käyttävät MySQL-järjestelmää, sillä tietokannan hallintajärjestelmän ei ole tarkoitus olla vertailun kohteena. HTTP-viestit ovat myös olennainen osa palvelinpuolen toimintaa, sillä näillä laukaistaan tapahtumia palvelinpuolella.

2.1 Palvelinpuolen määritelmä

Asiakas-palvelin-malli perustuu ohjelmiston jakamisen kahteen eri osaan. Tässä mallissa asiakas tekee palvelimelle pyyntöjä, joihin palvelin vastaa tai joiden tietojen perusteella palvelin toteuttaa toimintoja. Jos palvelin sisältää myös tietoa, sitä voidaan kutsua tiedostopalvelimeksi. Palvelinpuolen tehtävänä voi olla myös web-sivujen tarjoaminen, mutta tiedostopalvelimen tapauksessa palvelin tarjoaa ainoastaan tietoja ja muokkaa niitä asiakkaan pyyntöjen perusteella. [1]

Web-palvelimet ovat asikas-palvelin -mallin palvelinosa. Web-palvelin ohjelmistot pyörivät tietokoneella ja kuuntelevat HTTP-pyyntöjä tietystä portista. Palvelinpuoli sisältää web-palvelimen sekä web-sovelluksen koodin. Pyyntö-vastaus -mallissa toiminta alkaa siitä, kun asiakas lähettää HTTP-pyynnön web-palvelimelle. Web-palvelin ottaa vastaan pyynnön ja päättää, mille web-sovellukselle pyyntö ohjataan prosessoitavaksi. Tämän jälkeen web-sovellus käsittelee pyynnön sen parametrien perusteella. Kun sovellus on käsitellyt kyseisen pyynnön, se luo web-palvelimelle vastauksen, joka lähetetään takaisin asiakkaalle. [2]

Web-palvelimen tehtävä on prosessoida HTTP-pyyntöjä ja lähettää näihin pyyntöihin vastauksia. Web-palvelin termillä voidaan viitata joko web-palvelin ohjelmistoon tai tiettyyn tietokoneeseen, jonka tehtävä on vastata tarjoamalla web-sivuja tai tiedostoja. Web-palvelimiin on implementoitu HTTP ja TCP-yhteyden käsittely. TCP on lyhenne sanoista transmission control protocol ja se huolehtii yhteyden muodostamisesta sekä viestien vä-

littämisestä tietokoneiden välillä verkossa. HTTP tulee sanoista hypertext trasfer protocol ja se määrittelee tavan, jolla tieto liikkuu verkon välityksellä. Yleiskäyttöiset web-palvelinohjelmistot pyörivät normaaleilla tietokoneilla, joissa on internetyhteyteen mahdollisuus. Web-palvelinohjelmistoja on olemassa monia erilaisia. Osa niistä on muovattu vain tiettyyn käyttöön, ja osa niistä on tarkoitettu laajempaan käyttöön. [3]

Palvelinpuolen CGI-komentosarjat (*engl.* Common Gateway Interface) ovat ohjelmia, jotka pyörivät palvelinkoneella. CGI on aplikaatioprotokolla, jonka tehtävänä on kuljettaa dataa ja tuloksia web-palvelimen ja palvelinpuolen komentosarjojen välillä. Sen tehtäviin kuuluu myös tiedon kuljettaminen verkkoselaimelle ja muille asiakkaille. Kun web-palvelimelta pyydetään staattista tiedostoa, esimerkiksi HTML-tiedostoa, sen täytyy hakea kyseinen tiedosto palvelimen tiedostoista. Mikäli web-palvelimelle tulee pyyntö, jossa pyydetään esimerkiksi PHP-tiedostoa, palvelin aloittaa ohjelman, joka prosessoi pyynnön osoittaman tiedoston. Kun ohjelma on prosessoinut tiedoston, se palauttaa tuloksen web-palvelimelle. [4]

Tietokannat ovat kokoelma dataa. Tietokantojen hallinnointijärjestelmät koostuvat datasta ja ohjelmista, joilla dataan päästään käsiksi. Näiden ohjelmien avulla dataa voi tallentaa ja hakea tietokannasta. Yhdessä tietokanta ja tietokannan hallintajärjestelmä muodostavat tietokantajärjestelmän. Järjestelmän käyttäjät voivat lisätä uutta tietoa tietokantaan tai poistaa, hakea ja päivittää tietokannan tietoja. Kolmiportaisessa arkkitehtuurissa asiakkaan pyydettyä tietoa palvelimelta, palvelinpuoli kommunikoi tietokantajärjestelmän kanssa. [5]

2.2 Palvelinpuolen kielet

Node.js on asynkroninen tapahtumaohjattu JavaScript-ajoaika, joka on rakennettu Googlen V8 JavaScript -moottorin päälle. Node on tehty skaalautuvien verkkosovellusten tekemiseen. Se perustuu samanaikaisuuteen, jolla pystytään hoitamaan esimerkiksi monia yhteyksiä samanaikaisesti. [11]

PHP on laajalti käytetty skriptauskieli, jonka syntaksiin on otettu vaikutteita C, Java ja Perl -kielistä. Sen päätarkoitus on mahdollistaa web-sivujen dynaamisia ominaisuuksia. PHP on avointa lähdekoodia ja sen voi hyvin sijoittaa HTML-kielellä kirjoitettuun koodiin. [12]

2.2.1 Node.js

Node.js on palvelinpuolen JavaScript-ympäristö. Node.js on tunnettu myös Node-nimityksellä. Node perustuu Googlen ajoaikaiseen implementaatioon V8 -moottoriin. Node on implementoitu suurimmaksi osaksi C ja C++ kielillä. V8 tukee pääasiallisesti JavaScriptiä selaimessa, mutta Noden tarkoituksena on tukea pitkäaikaisia palvelinprosesseja. Node perustuu tapahtumamalliin, jota se tukee kielitasolla. Tästä syystä JavaScript sopii hyvin

Nodelle, koska se tukee tapahtumavastakutsuja. Esimerkiksi kun selain lataa dokumentin käyttäjän painaessa nappia tai kun Ajax-pyyntö on täytetty, tapahtuma laukaisee vastakutsun. Tapahtumamalli asynkronisella I/O:lla toimii Noden perustana, toisin kuin useat muut modernit ympäristöt, jotka luottavat säikeistykseen. [6]

Säikeistäminen auttaa useiden I/O-lähteiden käsittelyssä. Se on mahdollistanut ohjelmien rinnakkaisuuden ja tehnyt ohjelmien logiikasta helpompaa ymmärtää, implementoida, ylläpitää sekä tarjota tehokkaamman suorittamisen. Web-palvelimet, jotka suorittavat monia I/O-operaatioita, hyötyvät säikeistämisestä, kun prosessorilla on tarjota monta ydintä, jotka mahdollistavat aidon rinnakkaisuuden. Kun ohjelma pyörii yhdellä ytimellä ja se aloittaa I/O-operaation, se vaihtaa suorituskontekstin toiselle säikeelle sen ajaksi, kun edellinen säie suorittaa I/0-operaatiota. Säikeen vaihto tapahtuu, koska muuten ohjelman täytyisi pysäyttää suorittaminen I/O-operaation ajaksi. Kun suorituskonteksti vaihdetaan toiselle säikeelle, ohjelma voi jatkaa suoritusta normaalisti. Alkuperäinen säie vapautuu uudelleen ohjelman suorittamiseen, kun I/O-operaatio on suoritettu. Säikeistämisen hyödyistä huolimatta sitä pidetään vaikeana, koska siihen liittyy heikkouksia. Säikeistämisen luomat ongelmat voivat olla vaikeita eristää. Ongelmia ovat esimerkiksi lukkiutumat ja jaettujen resurssien suojaaminen. [6]

Tapahtumaohjattu ohjelmointi luottaa tapahtumailmoituksiin. Tämä tarkoittaa sitä, että ohjelman halutessa suorittaa jokin tietty operaatio, joka ei juuri sillä hetkellä ole mahdollinen, se jatkaa suorittamista ja jää kuuntelemaan tapahtumailmoitusta. Tapahtumailmoitus tulee, kun operaatio on mahdollista suorittaa. Asynkroninen I/O toimii perustana tapahtumaohjatulle ohjelmoinnille, koska se estää ohjelman pysähtymisen I/O-operaatioiden kohdalla. Kun asynkroninen I/O ei ole käytössä ja ohjelman tarvitsee kirjoittaa tiettyyn estolliseen kantaan, ohjelman suritus pysähtyy kokonaan, mikäli kannan puskuri on täynnä. Asynkronista I/O:a käytettäessä estoton kanta ilmoittaa ohjelmalla, että se voi jatkaa suoritusta ja saa tapahtumailmoituksen, kun ohjelma voi kirjoittaa kantaan. Tapahtumaohjatulla asynkronisella I/O ohjelmoinnilla on myös ongelmansa. Yksi ongelmista on se, että kaikkia operaatiota ei voida liittää tapahtumailmoituksiin. Toinen ongelma on ohjelmointikielien kyvyttömyys käsitellä tapahtumia ja asynkronista I/O:a. [6]

Node käyttää pääasiassa asynkronista I/O-mallia ja siihen on ainoastaan käytettävyyden takia lisätty synkronisia funktioita, esimerkiksi tiedostojen uudelleen nimeämiseen ja poistamiseen. Jokaisessa I/O-operaatiossa hyödynnetään vastakutsuja, joka tarkoittaa korkeamman järjestyksen funktioiden käyttöä. Korkeamman järjestyksen funktiot ovat funktioita, jotka ottavat parametriksi funktion. [6]

```
1 var sys = require("sys"),
2     http = require("http"),
3     url = require("url"),
4     path = require("path"),
5     fs = require("fs");
6
7 http.createServer(function(request, response) {
8     var uri = url.parse(request.url).pathname;
9     var filename = path.join(process.cwd(), uri);
```

```
10      path.exists(filename, function(exists) {
11          if(exists) {
12              fs.readFile(filename, function(err, data) {
13                  response.writeHead(200);
14                  response.end(data);
15              });
16          } else {
17              response.writeHead(404);
18              response.end();
19          }
20      });
21  }).listen(8080);
22  sys.log("Server running at http://localhost:8080/");
```

Ohjelma 1. *Nodella toteutettu yksinkertainen HTTP-tiedostopalvelin [6].*

Ohjelmasta 1 voidaan nähdä miten yksikertainen web-palvelin on toteutettu Nodella. Tämän palvelimen tehtävänä on tarjota asiakkaalle tiedostoja palvelimelta. Ohjelmassa 1 on hyviä esimerkkejä siitä, miten korkeamman järjestyksen funktiot toteutetaan Nodella funktioina, jotka ottavat parametrikseen nimettömiä funktioita. Ohjelmasta huomataan myös, että mikään selkeästi kutsutuista funktioista ei tee I/O-operaatioita. Funktiot rekisteröivät sopivat vastakutsut, jotka toteutetaan, kun ohjelma saa tapahtumailmoituksen I/O-operaation suorittamisen mahdollisuudesta. Noden ytimeen implementoitu tapahtumasilmukka mahdollistaa tämän kaiken asynkronisuuden. Tästä syystä Node pystyy palvelemaan monia asiakkaita samanaikaisesti. Node luo matalan tason HTTP-protokollan implementaation http.createserver-funktiolla. Tämä funktio herätetään ainoastaan silloin kun tiedot pyynnöltä ovat valmiina luettavaksi. Yksittäinen Node-palvelin voi palvella monia asiakkaita samanaikaisesti, vaikka se toimii vain yhdellä säikeellä. Tämä onnistuu siitä syystä, että pääasiallisen koodin ympäri pyörivä silmukka suorittaa ainoastaan rekisteröintipyyntöjä itse I/O-operaatioiden suorittamista varten. [6]

2.2.2 PHP

PHP on ohjelmointikieli, joka on luotu web-sivujen tekemiseen. Se muodostui aluksi C-kielen kirjastoista, joista se myöhemmin kehittyi omaksi ohjelmointikieleksi. PHP:n alkuperäinen tarkoitus oli nostaa tuottavaisuutta, kun PHP:n kehittäjän, Rasmus Leodorfin, täytyi tehdä web-aplikaatioita monille eri asiakkaille. Leodorf päätyi jakamaan luomaansa PHP-kirjastoa muille kollegoille, josta sen suosio lähti nousuun. Kollegat löysivät bugeja ja korjasivat niitä ja lähettivät ne edelleen Leodorfille. Muut käyttäjät alkoivat kirjoittamaan päivityksiä ja korjaamaan virheitä PHP:ssa. PHP:n käyttäjien määrän kasvaessa siitä tuli joukoille ulkoistettu ohjelmointikieli. PHP:n käyttäjät pystyvät luomaan uusia toiminnalisuuksia ja ne implementoidaan ydinkoodiin, mikäli ne todetaan toimiviksi. [7]

PHP pyörii tyypillisesti web-palvelimella, jolloin sen se ei kuormita asiakkaan tietokonetta. Kun web-palvelin saa HTTP-pyynnön, se hakee kyseisen tiedoston palvelimen tiedostoista. Mikäli tiedosto sisältää PHP:ta tai on kokonaan PHP:ta, tiedosto siirtyy PHP-moottorille, joka käsittelee kyseisen tiedoston PHP-kieliset komennot ja palauttaa niiden

mukaista tietoa. Koska PHP-komennot käsitellään palvelimen puolella, asiakas ei näe miten kyseinen PHP-komentosarja on kirjoitettu, vaan asiakas näkee ainoastaan sen outputin.[8]
PHP-kieltä kuvaillaan helposti opittavaksi ja yli 200 miljoonaa sivustoa hyödynsi sitä vuonna 2016. PHP toimii myös monella eri käyttöjärjestelmällä, eikä käyttöjärjestelmä vaikuta siihen, miten PHP-ohjelmat kääntyvät, vaan saman ohjelman voi suoraan kopioida toiselle käyttöjärjestelmälle. Monet eri web-palvelimet tukevat myös PHP:ta, joista Apache on suosituin. Kaikki web-palvelimet, jotka tukevat CGI-standardia, tukevat myös PHP:ta. PHP toimii myös monien eri tietokantojen kanssa, kuten MySQL:n ja MongoDB:n. [8]

2.3 MySQL

MySQL on tietokannan hallinnointijärjestelmä, joka hallinnoi relaatiotietokantoja. MySQL tarjoaa ominaisuuksia, jotka takaavat ympäristön datan turvalliseen varastoimiseen, ylläpitoon ja käsiksi pääsyyn. MySQL on skaalautuva, joten se pystyy käsittelemään isojakin tietokantoja. MySQL on myös siirrettävä, koska se tukee monia eri käyttöjärjestelmiä. Sen yhdistettävyyden takaa ohjelmointirajapinta, joka mahdollistaa yhteyden eri ohjelmointikielien kanssa, joita ovat esimerkiksi PHP, Java ja Python. MySQL sisältää myös järjestelmän, jolla hallinnoidaan pääsyä dataan. Avoin lähdekoodi mahdollistaa käyttäjien osallistumisen lähdekoodin tarkasteluun, testaamiseen ja kehittämiseen. [9]
SQL on konekieli, jonka tehtävänä on hallinnoida dataa relationaalisessa tietokannassa. "SQL toimii yhdistelmänä relaatiotietokannan hallintajärjestelmän kanssa tietokannan rakenteen määrittämiseen, datan varastoimiseen kyseisessä tietokannassa, datan manipuloimiseen, datan noutamiseen, dataan pääsyn kontrolloimiseen ja datan eheyden takaamiseen."[9] Relaatiotietokantojen hallintajärjestelmät on rakennettu esimerkiksi C ja C++ ohjelmointikielillä, koska tällä mahdollistetaan yhdistyneisyys, tarjotaan ohjelmointirajapintoja, mahdollistetaan pääsy verkkoon tai kanssakäynti asiakkaan välineiden kanssa. Relaatiotietokantojen hallintajärjestelmissä SQL-kielen tehtävä on mahdollistaa kanssakäynti tietokannan kanssa. [9]

2.4 HTTP-viestit

HTTP viestejä on olemassa kahdenlaisia, pyyntö- ja vastausviestejä. HTTP-viestit ovat yksinkertaisia tekstimuotoisia viestejä. Viestit, jotka lähetetään verkkoasiakasohjelmistosta, ovat pyyntöjä. Palvelimien lähettämät viestit ovat vastauksia ja ne vastaavaat verkkoasiakasohjelmistojen lähettämiin pyyntöihin. HTTP-viestit koostuvat kolmesta osasta: aloitusrivistä, header-soluista ja kehosta. Aloitusrivi pyyntöviestissä kertoo mitä pyynnölle pitää tehdä ja vastausviestissä se kertoo mitä on tapahtunut. Header-soluja viesteissä on nolla tai enemmän ja ne antavat lisätietoja viestistä. Yksittäinen solu koostuu nimestä ja arvosta. Keho on valinnainen osa viestiä ja se voi sisältää mitä tahansa dataa. Pyyntöviesteissä keho vie dataa palvelimelle ja vastausviesteissä se tuo dataa verkkoasiakasohjelmistolle. [3]

Molemmilla viestityypeillä on oma syntaksinsa. Niillä on kuitenkin samanlainen perusrakenne.

Pyyntöviestin formaatti:

```
1 <method> <request-URL> <version>
2 <headers>
3
4 <entity-body>
```

Vastausviestin formaatti:

```
1 <version> <status> <reason-phrase>
2 <headers>
3
4 <entity-body>
```

Method-osa kertoo mitä asiakas haluaa palvelimen tekevän osoitetulle resurssille. Method-arvoja ovat mm. "GET", "HEAD"tai "POST". Request-URL kertoo minkä resurssin asiakasverkko-ohjelmisto haluaa. Version-osa kertoo viestin käyttämän HTTP-version. Status-code on kolmen numeron yhdistelmä, joille on olemassa omat merkityksensä. Reason-phrase antaa luettavan muodon status-code -solun antamasta numeroyhdistelmästä. Headers-solu sisältää nolla tai useamman nimen ja arvon yhdistelmän erotettuna kaksoispisteellä. Entity-body sisältää vapaasti määrättyä dataa, mutta tätäkään ei ole pakko olla. [3]

2.5 Node ja PHP+Apache ympäristöjen tehokkuus

Nodea ja PHP+Apachea on vertailtu aikaisemmin keskenään ja näissä Node on aina selvinnyt tehookkaammaksi [10, 11]. Näissä tutkimuksissa ei kuitenkaan vertailtu tietokantaan tallentamista ja vain toisessa vertailun kohteena oli tietokannasta hakeminen.
Lein, Man ja Tanin tutkimuksesssa Node oli selkeästi paras palvelin, koska se suoritti eniten HTTP-pyyntöjä lähes kaikissa eri vertailuissa. Tutkimuksessa palvelimena toimi tietokone Ubuntu-käyttöjärjestelmällä, jolle palvelimet oli rakennettu. Tässä täytyy ottaa myös huomioon, että Apache oli konfiguroitu käyttämään prefork MPM -moduulia, joka ei ole säikeistetty. Tämä aiheuttaa sen, että viestejä ei käsitelty samaan aikaan eri säikeillä, vaan ne laitettiin odottamaan säikeen vapautumista. Tutkimuksessa verrattiin ensiksi palvelimen nopeutta vastata "hello world"asiakkaalle, toisessa testissä verrattiin palvelimien nopeutta kymmenennen Fibonaccin luvun laskemiseen ja kolmannessa testissä verrattiin kuinka nopeasti palvelimet suorittivat select-operaation tietokantaan ja palauttivat operaation vastauksen. Kaikki testit suoritettiin, kun samanaikaisia HTTP-viestejä lähetettiin 10, 100, 200, 500 ja 1000. Ainut tilanne, jossa PHP+Apache käsitteli pyyntöjä nopeammin kuin Node, oli tilanne, jossa laskettiin Fibonaccin 10. kertalukua ja samanaikaisia pyyntöjä lähetettiin 100. Muissa tilanteissa Node oli selkeästi parempi. Tuloksista myös huomataan kuinka Noden tehokkuus laskee huomattavasti hitaammin samanaikaisten pyyntöjen kasvaessa, kun taas PHP+Apachen tehokkuus laskee nopeasti. [10]
Chaniotisin, Kyriakoun ja Tselikasin tutkimuksessa testattiin palvelimien nopeutta vastata "hello world"HTTP-pyyntöihin. Toisessa testissä vertailtiin maantieteellisen sijainnin

koodausta. Tässäkin tutkimuksessa palvelimet oli asennettu Ubuntu käyttöjärjestelmälle. Huomioitavaa on myös että Apache käytti worker-moduulia, jolloin se pystyi suoriutumaan testeissä ilmenevistä korkeista samanaikaisuuden tasoista hyödyntämällä useampia säikeitä. Node palvelimessa oli hyödynnetty klusterointimoduulia, joka pystyy hyödyntämään jokaista prosessorin ydintä. Tutkimuksen "hello world-testissä Node suoritti pyyntöjä reilu kaksi kertaa enemmän kuin PHP+Apache samanaikaisten viestien määrän kasvaessa 0:sta 30000:een. Testissä tarkasteltiin myös kuinka paljon prosessorin tehoa ja tietokoneen muistia palvelimet käyttivät. Node käytti prosessoria 5000 samanaikaisen pyynnön jälkeen enemmän kuin PHP+Apache, mutta tämä näkyy myös pyyntöjen käsittelyn nopeudessa. PHP+Apachen prosessorin käyttöaste vaihtelee välillä 30-70 kun samanaikaisten yhteyksien määrä on 2000 ja 30000 välillä. Noden prosessorin käyttö taas nousee tasaisen jyrkästi alkuun kunnes tasoittuu samanaikaisten yhteyksien kasvaessa. Muistin käytössä Node on myös parempi sillä sen muistinkäyttö nousee noin 1,7 gigatavusta 2,5 gigatavuun, kun taas Apache+PHP nousee noin 2,5 gigatavusta 3 gigatavuun. Tutkimuksen testissä, jossa käytetään tiettyä algoritmia maantieteellisen sijainnin enkoodaamiseen, Node pystyy käsittelemään noin neljä kertaa enemmän pyyntöjä verrattuna Apache+PHP:hen. Prosessorin käyttöasteet noudattavat lähes samanlaista trendiä tässä testissä kuin tutkimuksen "hello world-testissä. Muistinkäyttö enkoodaus-testissä kasvaa Apache+PHP-palvelimella, kun taas Nodella se pysyy lähes samana. Enkoodaus testissä Apache+PHP:n muistinkäyttö nousee noin 2,4 gigatavusta 4,1 gigatavuun, kun Nodella nousu on noin 1,7 gigatavusta noin 2,4 gigatavuun. [11]

3 TUTKIMUSMENETELMÄT JA AINEISTO

Tutkimuksen tavoitteena on selvittää, kumpi palvelinpuolen ympäristöistä soveltuu paremmin vedenmittauksesta saadun datan tallentamiseen, Node.js vai PHP. Palvelinpuolen tehtävä on pystyä tallentamaan HTTP-viesteillä tulevat mittaustiedot mahdollisimman tehokkaasti tietokantaan. Tehokkuus nousee tärkeimmäksi kriteeriksi, HTTP-viestien samanaikainen määrä kasvaa. Tutkimuksessa selvitetäänkin, kuinka kaksi eri palvelinpuolen toteutusta suoriutuu mittaustietojen tallentamissa samanaikaisten HTTP-pyyntöjen määrän kasvaessa.
Testeissä vertaillaan kuinka nopeasti palvelinpuolet käsittelevät HTTP-viestejä. Palvelimen tehtävänä on vastaanottaa POST-metodilla tuleva HTTP-viesti ja tallentaa sen sisältämät mittaustiedot tietokantaan. Testit tullaan tekemään neljällä eri samanaikaisuuden tasolla. Tämä tarkoittaa sitä, kuinka monta samanaikaista HTTP-viestiä testityökalu tulee lähettämään. Palvelinpuolta vertaillaan tällä mittarilla, koska se antaa hyvän kuvan siitä kuinka ne suoriutuvat mittausdatan hallinnointiin. Palvelimien on mahdollista käsitellä jopa tuhannen eri vedenmittauslaitteen lähettämiä tietoja, joten samanaikaisten HTTP-viestien käsittely on tärkeää. Muita mitattavia parametrejä ovat prosessorin käyttöaste sekä muistin käyttöaste. Näitä on myös tärkeä tarkastella, jotta mahdolliset palvelinpuolien resurssientarpeen erot näkyvät.
Palvelinpuolen toteutukset toimivat Windows 10 Education -käyttöjärjestelmän päällä, jonka versio on 10.0.17763. Tietokoneen prosessori on Intel i5-4690k 4,3GHz, muisti 2x4GB DDR3 1883MHz ja emolevy Asus Z-97a.
Testaustyökaluna käytetään Apache HTTP server benchmarking tool -työkalua, joka tulee Apachen mukana. Tämän työkalun avulla pystytään määrittämään parametrit halutuille testeille. Työkalu löytyy Apachen kansiosta /Apache24/bin/ab.exe. Kun mennään komentorivillä tähän tiedostoon, pystytään antamaan komentoja jotka ovat muotoa:

```
1 ab.exe -n 100000 -c 10 -m POST "localhost:3000/data?id=1&timestamp
    =20190112143631"
```

Kyseisessä komennossa -n kertoo kuinka monta HTTP-viestiä lähetetään yhteensä. Komennon -c osa kertoo kuinka monta samanaikaista viestiä lähetetään ja POST on HTTP-viestin metodi. Komennon lopussa on osoite, johon kyseinen viesti halutaan lähettää.

3.1 Node-ympäristö

Node.js ympäristön asentaminen aloitetaan asentamalla Noden uusin versio 12.13.1 (Osoite: `www.nodejs.org/en/download/`). Tämän jälkeen ladataan kaksi Node-moduulia

palvelimen toteuttamisen helpottamiseksi. Moduulit asennetaan koneelle NPM-komentoa (*engl*. Node Package Manager) käyttäen. NPM toimii siis Noden pakettien hallinnointina. Kaksi asennettavaa moduulia ovat express (Osoite: `www.npmjs.com/package/express`) ja mysql (Osoite: `www.npmjs.com/package/mysql`). Express-moduuli helpottaa palvelimen eri polkujen reitittämisessä. Mysql-moduulin tehtävänä on helpottaa tietokantaan liittyvien komentojen kirjoittamisessa. Seuraavaksi luodaan kansio, johon Node-palvelin luodaan. Navigoidaan terminaalissa kyseiseen kansioon ja suoritetaan komennot:

```
npm init
npm install express
npm install mysql
```

Kansioon on tullut uusi kansio node_modules sekä package.json tiedosto. Package.json tiedostosta muutetaan rivi:

```
"main": "index.js",
```

muotoon:

```
"main": "main.js",
```

Tämä rivi kertoo mikä on projektin juuritiedosto ja tämän jälkeen voidaan kirjoittaa tarvittava koodi kyseiseen tiedostoon. Lisätään myös scripts-muuttujaan yksi rivi lisää:

```
"start": "node main.js",
```

Tämä mahdollistaa palvelimen käynnistämisen kommennolla "npm start". Seuraavaksi kirjoitetaan Node-palvelimen toiminnallisuus.

```
const port = 3000;
const mysql = require("mysql");
var pool = mysql.createPool({
  connectionLimit : 1000,
  queueLimit: 10000,
  host: "localhost",
  user: "kandi_testi",
  password: "salasana123",
  database: "waterconsumption"
});

const express = require("express");
const app = express();

app.post("/data", (req,res) => {
  var device_id = req.query.id;
  var timestamp = req.query.timestamp;

  pool.getConnection(function(err, connection) {
    if (err){
      console.log(err.code);
      res.end();
    }else{
      connection.query("INSERT INTO measurement VALUES (" + device_id + "," +
    timestamp + ")",
      function (error, results, fields) {
```

```
26          connection.release();
27          res.statusCode = 200;
28          res.end();
29        });
30      }
31    });
32 });
33
34 app.listen(port, () =>{
35   console.log(‘The server has started and is listening on port number:${port}‘)
36 });
```

Ohjelma 2. *Nodella toteutettu palvelin.*

Rivillä 1 määritetään portin numero, jota palvelin tulee kuuntelemaan. Rivit 2-10 määrittävät MySQL-yhteyksien parametrit. Rivi 15 määrittää mitä tapahtuu, kun palvelin saa viestin polulle /data. Seuraavaksi tallennetaan muuttujiin viestissä saadut parametrit ja otetaan yhteys MySQL-tietokantaan. Kun tiedot on saatu tallennettua onnistuneesti tietokantaan, vapautetaan tietokantayhteys uudelleenkäytettäväksi rivillä 26 ja lähetetään vastaus asiakkaalle rivillä 28.

3.2 Php ja Apache -ympäristö

PHP:n ja Apachen palvelinpuolen pystytys aloitetaan lataamalla ja asentamalla sekä PHP että Apache. Ladataan PHP:n uusin vakaa versio 7.3.12 VC15 x64 Thread Safe (2019-Nov-27 12:47:05) (Osoite: `www.windows.php.net/download`). Ladataan ja asennetaan Apachen uusin stabiili versio Apache 2.4.41 Win64 (Osoite: `www.apachelounge.com/download/VC15/`). Puretaan sekä PHP, että Apache tiedostot C: polkuun. Pystyäksemme käyttämään PHP:ta Apachella, täytyy tehdä muutoksia Apachen konfigurointitiedostoon. Muokattava tiedosto löytyy polusta /Apache24/conf/httpd.conf. Kyseiseen tiedostoon poistetaan kommenteista rivi:

```
1 Include conf/extra/httpd-mpm.conf
```

Tällä muutoksella otetaan käyttöön Multi-Processing Modules, jolloin saamme lisättyä työntekijäsäikeitä suorittamaan HTTP-viestejä. Jotta PHP toimisi Apachen kanssa, lisätään LoadModule-komentojen perään rivit:

```
1 LoadModule php7\_module "c:/php/php7apache2\_4.dll"
2 PHPIniDir "c:/php"
```

Poistetaan myös seuraava rivi kommenteista ja vaihdetaan muuttuja:

```
1 ServerName localhost
```

Lopuksi muokataan tiedostoa httpd-mpm.conf, joka löytyy polusta /Apache24/conf/extra. Tästä tiedostosta muokataan WinNT MPM -kohdasta ThreadsPerChild:

```
1 ThreadsPerChild 1000
```

Nyt Apache pystyy suoriutumaan tuhannen samanaikaisen HTTP-pyynnön kanssa. Lopuksi kirjoitetaan PHP-tiedosto, joka sijoitetaan polkuun: /Apache24/htdocs.

```
1  <?php
2    function new_connection(){
3      $connection = new mysqli("localhost", "kandi_testi", "salasana123", "
       waterconsumption");
4      return $connection;
5    }
6
7    $device_id = $_GET["id"];
8    $time = $_GET["timestamp"];
9    $link = new mysqli("localhost", "kandi_testi", "salasana123", "
       waterconsumption");
10   while($link->connect_errno == True) {
11     echo "Failed to connect to MySQL: (" . $link->connect_errno . ") " . $link
       ->connect_error;
12     $link = new_connection();
13   };
14   $res = mysqli_query($link, "INSERT INTO measurement VALUES ($device_id, $time
       )");
15   mysqli_kill( $link,  $link->thread_id);
16 ?>
```

Ohjelma 3. *PHP:lla toteutettu vedenmittausdatan tallentaminen.*

Riviltä 2 alkaen määritellään funktio, joka ottaa uuden yhteyden tietokantaan, mikäli sitä ei pystytä heti luomaan. Rivi 9 luo yhteyden tietokantaan ja seuraavaksi tarkastetaan, että yhteys on onnistunut. Rivillä 14 tallenetaan tiedot tietokantaan ja rivillä 15 lopetetaan yhteys tietokantaan, jolloin tämä yhteyspaikka vapautuu uudelleen käytettäväksi.

3.3 MySQL-ympäristö

Tietokantana tutkimuksessa toimii SQL-tietokanta, jonka hallintajärjestelmänä toimii MySQL. Ladataan ja asennetaan uusin versio 8.018 MySQL-tietokannan hallintajärjestelmästä. Luodaan uusi tietokanta, jota käytetään mittaustietojen tallentamiseen. Uuden tietokannan luominen onnistuu MySQL Shell -terminaalissa. Aluksi yhdistetään MySQL-palvelimeen komennolla:

```
1 \connect localhost:3306
```

jonka jälkeen shell pyytää käyttäjätunnuksia. Kun terminaalin sisällä on yhdistetty MySQL-palvelimeen, syötetään ensin komento:

```
1 \sql
```

jonka jälkeen pystymme käyttämään SQL-komentoja. Luodaan uusi tietokanta komennolla:

```
1 CREATE DATABASE waterconsumption;
```

Seuraavaksi otetaan kyseinen tietokanta käyttöön komennolla:

```
1 USE waterconsumption;
```

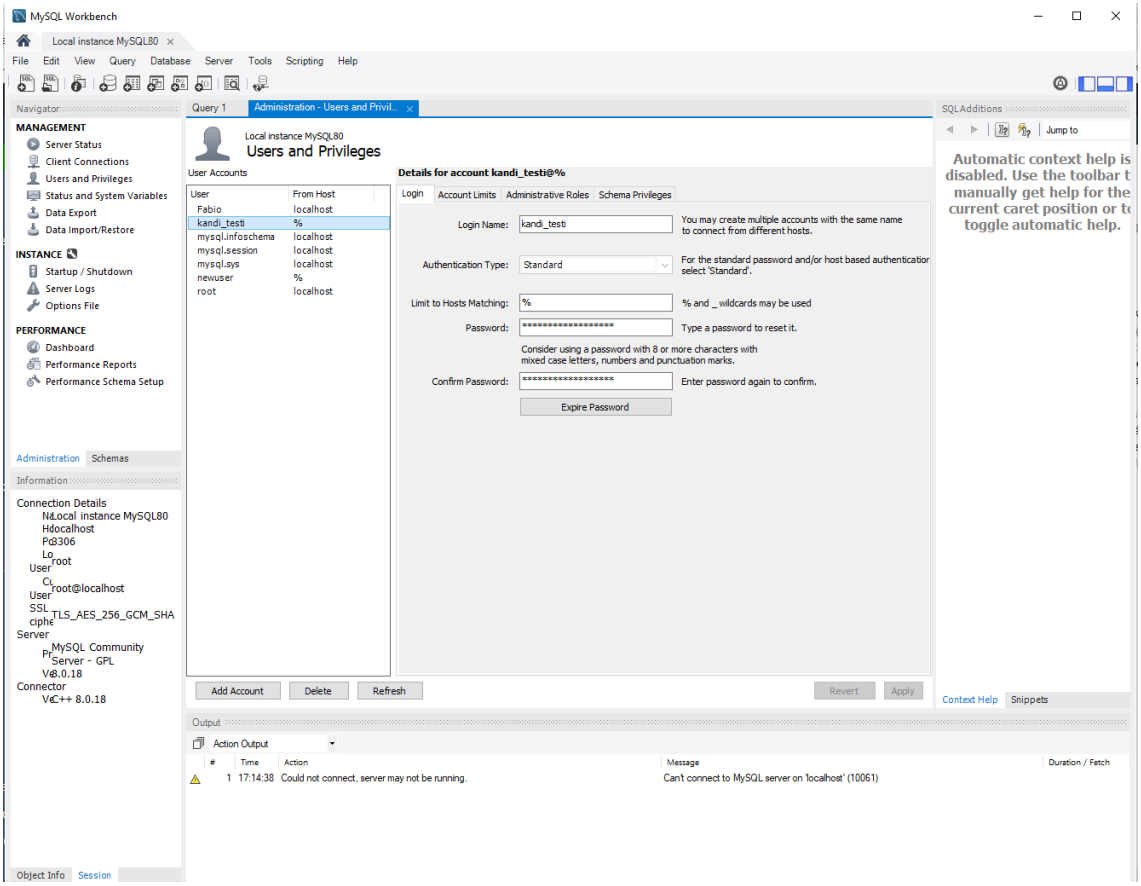
Tämän jälkeen luodaan taulukko, joka sisältää kaksi saraketta, joista toinen on mittalaitteen identifioiva numero ja toinen vesi-impulssin aikaleima. Taulukko luodaan komennolla:

```
1 CREATE TABLE measurement (id int, timestamp date);
```

Asetetaan myös yhteyksien maksimi määrä 1000 yhteyteen, jotta MySQL ei olisi rajoittava tekijä testeissä. Yhteyksien maksimimäärä asetetaan komennolla:

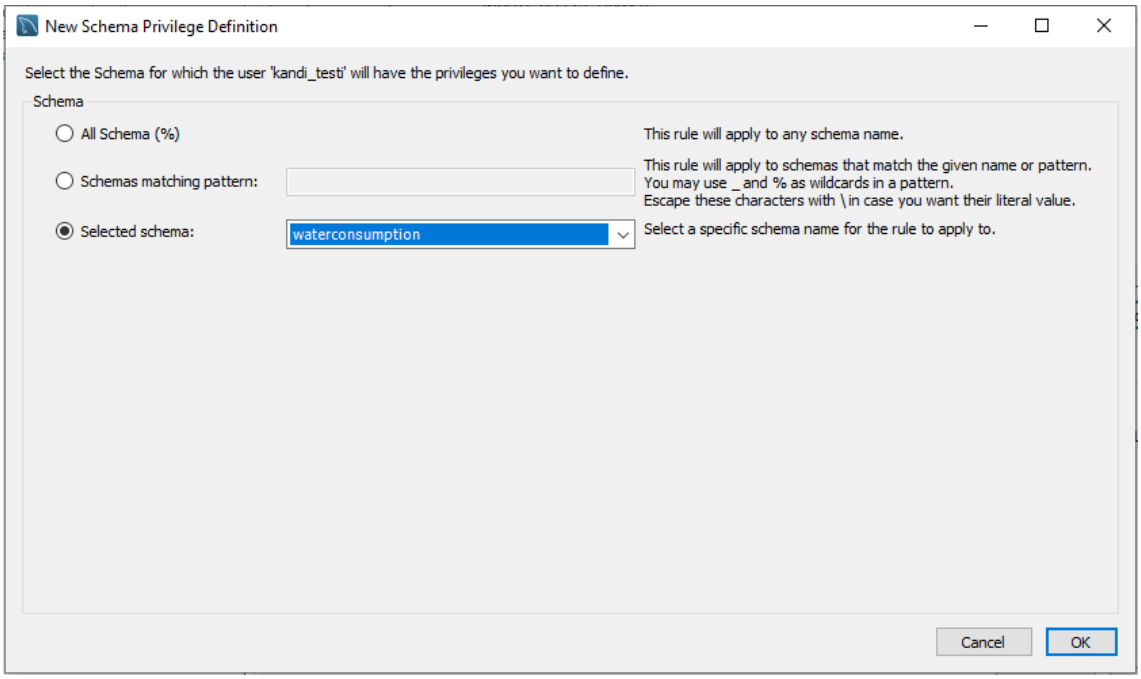
```
1 SET GLOBAL max\_connections = 1000;
```

Luodaan palvelinpuolien käyttöön uusi käyttäjä tietokannalle, jolla on oikeus tallentaa tietokantaan ja hakea sieltä tietoa. Käyttäjän voi luoda avaamalla MySQL Workbench -ohjelman ja valitsemalla alkuun localhost:3306-palvelin. Seuraavaksi valitaan vasemmalta Users and Privileges -osio, jonka vasemmasta alareunasta valitaan Add Account. Nyt valitaan käyttäjän nimeksi kandi_testi, Authentication Type: Standard, Limit to Hosts Matching: % ja salasanaksi: salasana123.



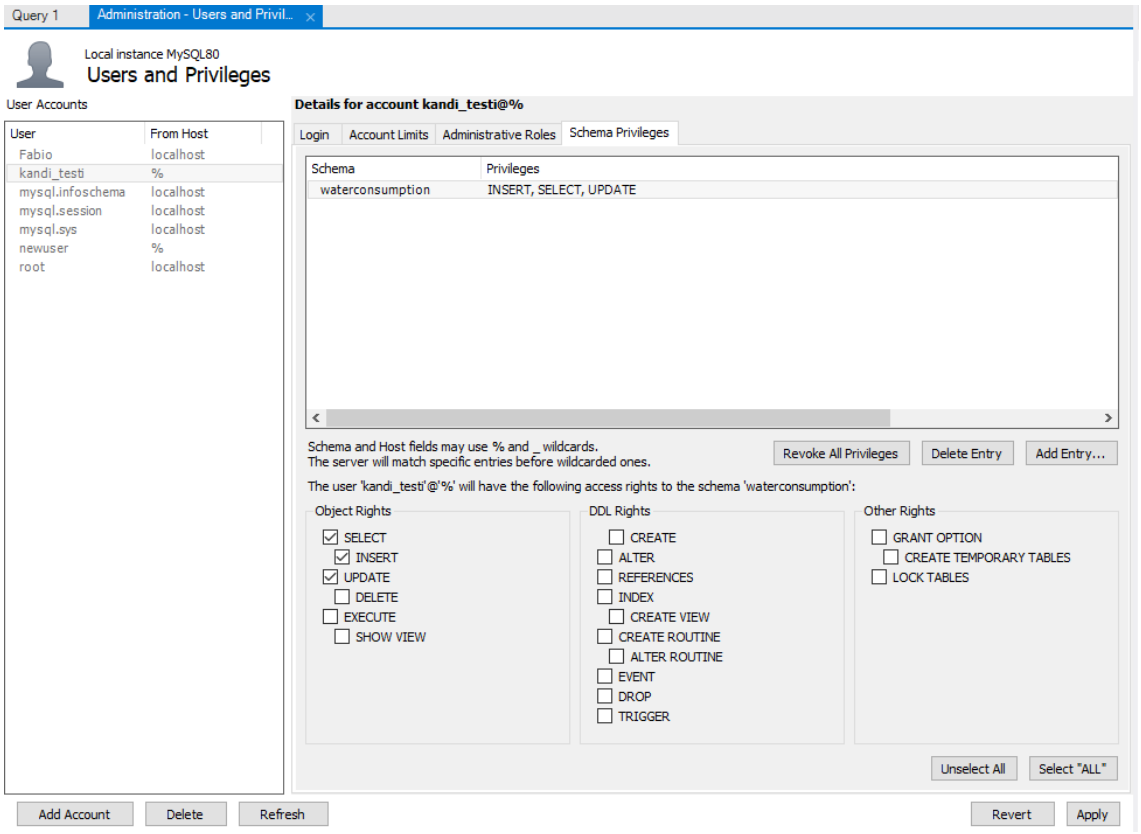
Kuva 1. *MySQL-käyttäjätiedot.*

Tämän jälkeen valitaan oikeasta yläkulmasta Schema Privileges -välilehti. Schema Privileges -välilehdellä valitaan oikealta keskeltä Add Entry. Ponnahdusikkunasta etsitään Selected schema -kohtaan waterconsumption -tietokanta ja painetaan OK.



Kuva 2. *MySQL-tietokannan valinta.*

Seuraavaksi valitaan SELECT ja INSERT -oikeudet ja painetaan Apply.

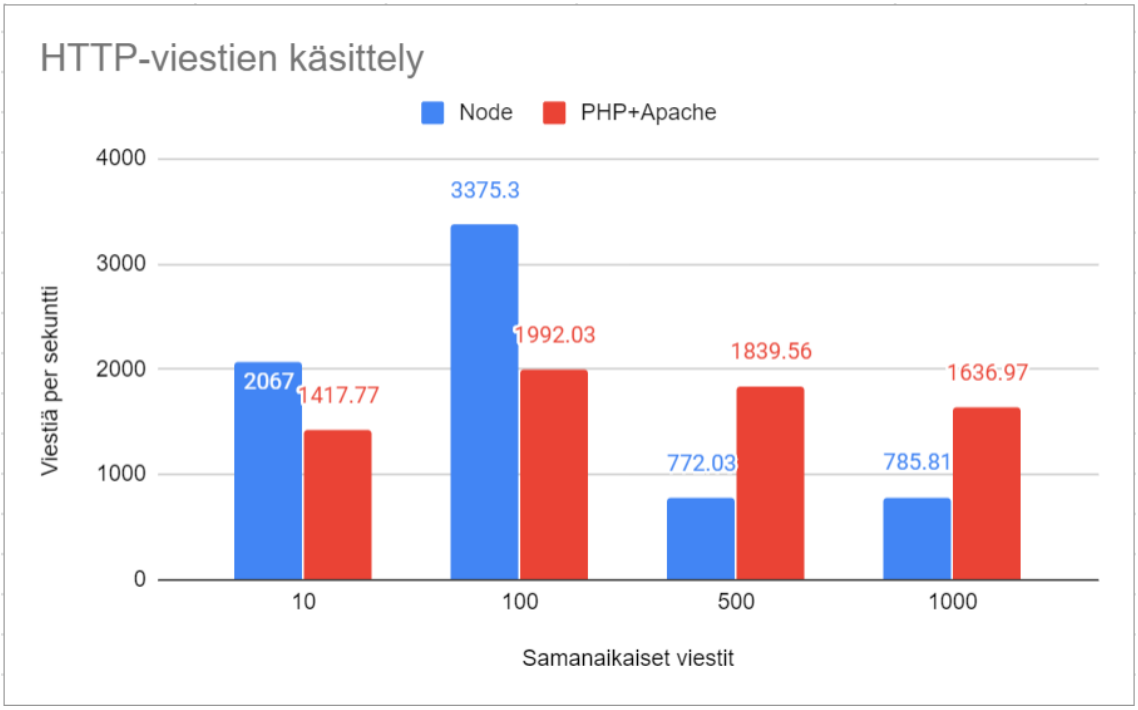


Kuva 3. *MySQL-tietokannan oikeuksien asettaminen.*

Nyt MySQL on konfiguroitu ja se on valmis suorittamaan komentoja, joita palvelimet lähettävät sille.

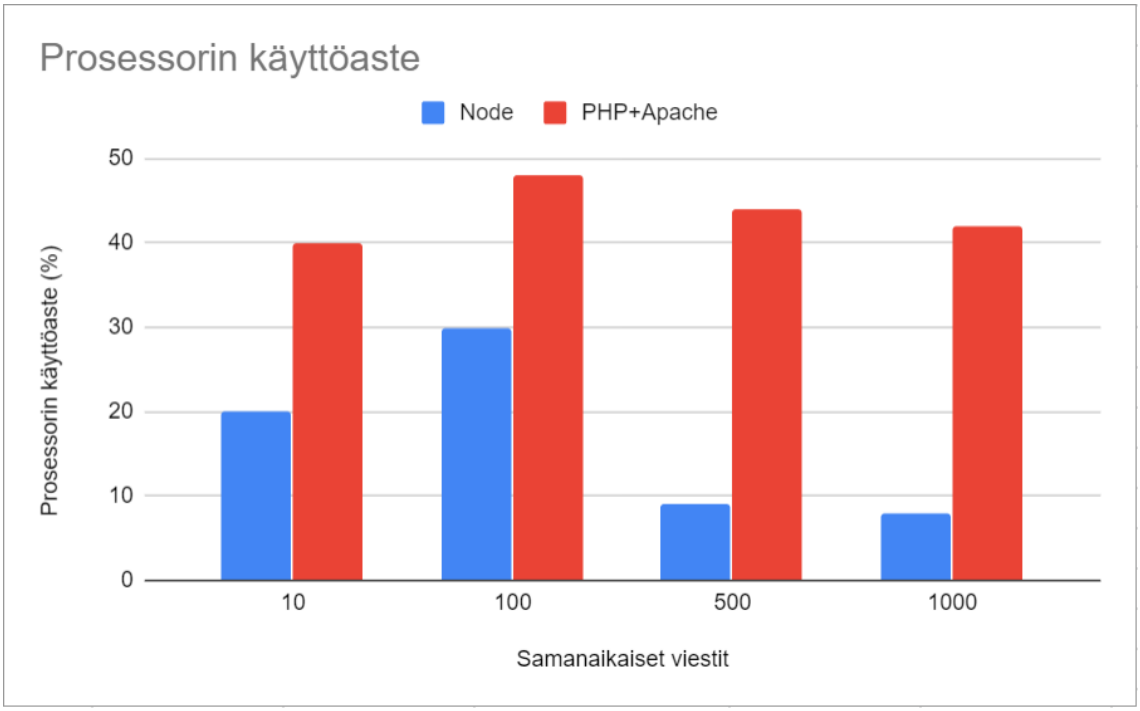
4 TULOKSET JA NIIDEN TARKASTELU

Testin tuloksista voidaan huomata kuinka Node käsittelee viestejä huomattavasti nopeammin 10 ja 100 samanaikaisuuden tasolla. Tämän jälkeen Noden nopeus jää huomattavasti vähäisemmäksi kuin PHP+Apachen käsittely nopeus. Noden hitaudesta huolimatta se pysyy lähes samana riippumatta siitä tuleeko samanaikaisia viestejä 500 vai 1000. Tästä voidaan päätellä että Noden parhaaseen käsittelynopeuteen päästään, kun samanaikaisuuden taso on 100 ja 500 välillä. Testien aikana huomattiin, että Node loi maksimissaan vain noin 250 yhteyttä MySQL-tietokantaan, vaikka sen oli mahdollista luoda 1000 yhteyttä.



Kuva 4. *Kuvaaja kuinka monta HTTP-viestiä palvelimet käsittelivät sekunnissa eri samanaikaisuuden tasoilla.*

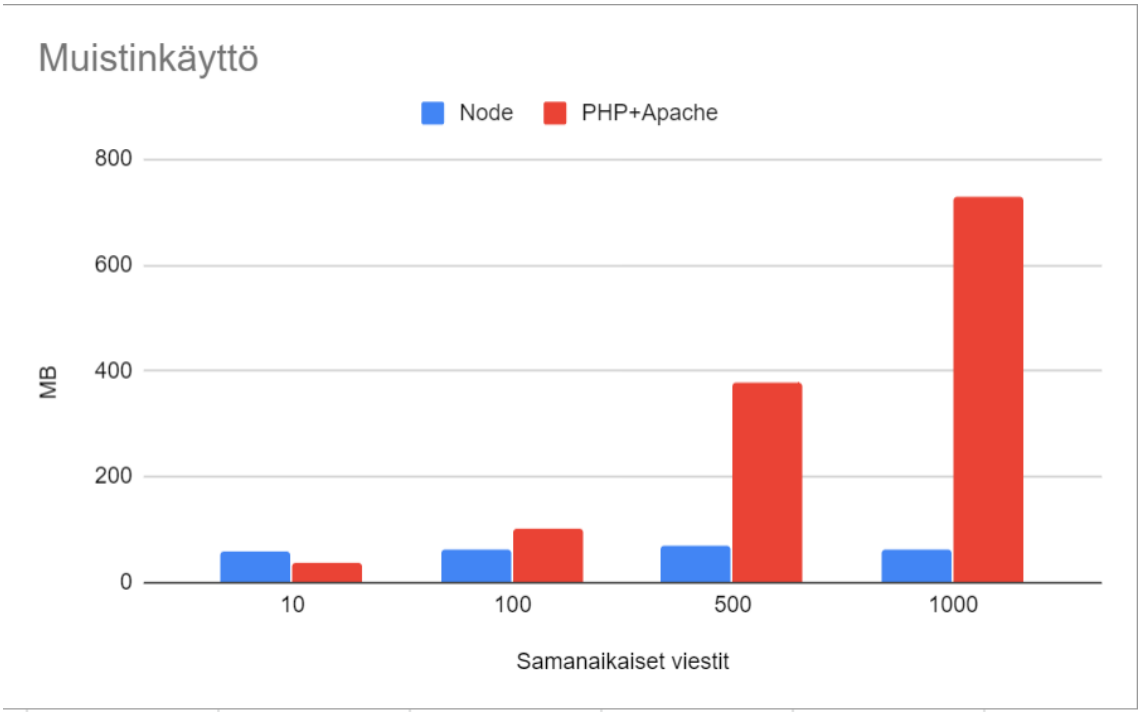
Prosessorin käyttöasteessa erot olivat huomattavat. Niistä voidaan myös huomata kuinka käyttöaste korreloi jossain määrin siihen, kuinka monta viestiä sekunnissa palvelimet käsittelivät. Havaittavissa on myös, että vaikka Node käsitteli enemmän viestejä sekunnissa 100 samanaikaisen viestin tasolla, se käyttää silti lähes 20% vähemmän prosessorin tehoa. Jokaisella tasolla huomataan, että Node vaatii selvästi vähemmän tehoa prosessorilta kuin PHP+Apache.



Kuva 5. *Kummankin palvelinpuolen käyttämä prosessorin käyttöaste.*

Muistinkäytöstä huomataan, että Node käyttää kaikilla tasoilla suurin piirtein saman verran muistia. PHP+Apache taas tarvitsee jatkuvasti lisää muistia, mitä enemmän sen täytyy palvella useita viestejä samanaikaisesti. Tässäkin huomataan, kuinka jo 100 samanaikaisen viestin palvelemisessa PHP+Apache tarvitsee enemmän muistia kuin Node.

PHP+Apachen nousevan resurssientarpeen selittää säikeiden aktivointi. Koska Apachelle on konfiguroitu käyttöön 1000 säiettä, niitä käytetään enemmän mitä korkeammaksi samanaikasten viestien määrä nousee. Node taas pääsee maksimiin 100-500 samanaikaisen viestin käsittelyn välillä, mutta tässä täytyy ottaa huomioon, että se käyttää vain yhtä säiettä. Apachea konfiguroidessa huomattiin myös, että mikäli samanaikaisten viestien määrä kasvaa liian suureksi siten, että sillä ei ole tarpeeksi säikeitä käytössä, Apache-palvelin meni jumiin eikä se pystynyt suorittamaan testiä loppuun.



Kuva 6. *Palvelinten muistin käyttö eri samanaikaisuuden tasoilla.*

5 YHTEENVETO JA PÄÄTELMÄT

Tutkimuksessa vertailtiin Node- ja PHP+Apache-palvelinympäristöjen tehokkuutta mittaustietojen tallentamiseen HTTP-viesteistä tietokantaan. HTTP-viestejä lähetettiin aina 100000 neljässä eri erässä, jossa samanaikaisia viestejä lähetettiin yhä enemmän. Nopeudesta, jolla palvelimet käsittelivät viestejä voidaan päätellä, että tällaisessa ympäristössä Noden raja on 100-500 samanaikaisen viestin välillä, jolloin Node pääsee huippuunsa. Apache+PHP pärjää tähän asti Nodea huonommin kunnes 500 samanaikaisen viestin käsittelyssä se on Nodea huomattavasti nopeampi. Tämä johtuu siitä, että Apachella on käytössään 1000 säiettä, joka myös näkyy sen muistin kulutuksessa. Node on muistin- ja prosessorinkäytön suhteen huomattavasti parempi ratkaisu, kun puhutaan noin sadasta samanaikaisesta yhteydestä. Mikäli samanaikaisia viestejä on odotettavissa enemmän kuin 500, Apache+PHP on parempi ratkaisu, kun prosessorin ja muistin käyttö ei ole ongelma. Tässä täytyy ottaa myös huomioon, että Node on mahdollista klusteroida, joka tarkottaisi sitä, että se käyttäisi prosessorin kaikkia ytimiä. Tällöin Node voisi olla mahdollisesti PHP+Apachea nopeampi myös samanaikaisten viestien määrän ollessa yli 500. Jatkossa tutkimusta voisi laajentaa eri kokoisen datan vastaanottamiseen ja lähettämiseen. Tilanne, jossa esimerkiksi vedenmittauslaitteelle ei ole mahdollista saada internet-yhtettä ja tiedot täytyy hakea manuaalisesti laitteelta aina tietyin ajoin, voidaan joutua lähettämään monia tuhansia rivejä mittausdataa palvelimelle. Mittausdatan analysointi on myös tärkeää, joten olisi hyvä selvittää miten palvelimet selviävät, kun niiden pitää hakea tuhansia mittaustietoja tietokannasta ja antaa ne asiakkaalle.

LÄHTEET

[1] RAJ, P., RAMAN, A., SUBRAMANIAN, H., Architectural Patterns, Packt Publishing, 2017, Saatavilla: https://learning.oreilly.com/library/view/architectural-patterns/9781787287495/ (Viitattu: 1.11.2019)

[2] LOPEZ, A., Learning PHP 7, Packt Publishing, 2016, Saatavilla: https://www.packtpub.com/eu/application-development/learning-php-7 (Viitattu 11.11.2019)

[3] GOURLEY, D., TOTTY, B., SAYER, M., HTTP: the definitive guide, O'Reilly, 2002, Saatavilla: https://learning.oreilly.com/library/view/http-the-definitive/1565925092/ (Viitattu: 13.11.2019)

[4] LUTZ, M., Programming Python, 4th Edition, O'Reilly Media, 2010, Saatavilla: https://learning.oreilly.com/library/view/programming-python-4th/9781449398712/ (Viitattu: 13.11.2019)

[5] VIDHYA, V., JEYARAM, G., ISHWARYA, K.R., Database management systems, Alpha Science International, 2016, Saatavilla: http://www.alphasci.com/books_display.asp?title=978-1-78332-213-8 (Viitattu: 14.11.2019)

[6] TILKOV, S., VINOSKI, S., Node.js: Using JavaScript to Build High-Performance Network Program, IEEE Internet Computing, vol. 14, no. 6, 2010, pp. 80-83, Saatavilla: https://ieeexplore.ieee.org/document/5617064 (Viitattu: 7.11.2019)

[7] SEVERANCE, C., Inventing PHP: Rasmus Lerdorf. Computer, vol. 45, no. 11, 2012, pp. 6-7 Saatavilla: https://ieeexplore.ieee.org/document/6353442 (Viitattu: 13.11.2019)

[8] Sklar, D., Learning PHP: A Gentle Introduction to the Web's Most Popular Language, First edn, O'Reilly, 2016, Saatavilla: https://learning.oreilly.com/library/view/learning-php/9781491933565/ (Viitattu: 12.11.2019)

[9] SHELDON, R., MOES, G., BOOKS24X7, I., Beginning MySQL, Indianapolis, IN: Wrox, 2005, Saatavilla: https://www.scribd.com/document/25916830/Beginning-MySQL (Viitattu 14.11.2019)

[10] LEI, K., MA, Y., TAN, Z., 2014 Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js, 2014, Saatavilla: https://ieeexplore.ieee.org/document/7023652 (Viitattu 3.2.2019)

[11] CHANIOTIS, I., KYRIAKOU, K., TSELIKAS, N., Is Node.js a viable option for building modern web applications? A performance evaluation study. Computing; Archives for Scientific Computing, vol. 97, no. 10, pp. 1023-1044, 2015, Saatavilla: https://dl.acm.org/doi/10.1007/s00607-014-0394-9 (Viitattu 3.2.2019)

[12] PHP Manual Preface, https://www.php.net/manual/en/preface.php (Viitattu 14.4.2020)