

TARTU ÜLIKOOL  
Arvutiteaduse instituut  
Informaatika (2476) õppekava

**Samanta Suitsik**  
**Raspberry Pi ja pilvserverite jõudlust ning**  
**kasutuskogemust võrdlev uuring**  
**Bakalaureusetöö (9 EAP)**

Juhendaja(d): Pelle Jakovits, PhD

Tartu 2025

# **Raspberry Pi ja pilvserverite jõudlust ning kasutuskogemust võrdlev uuring**

## **Lühikokkuvõte:**

Selles töös võrreldi omavahel odavat füüsilist taristut ehk Raspberry Pi'd ning soodsat pilvtehnoloogial põhinevat virtuaalmasinat. Töö eesmärk oli välja selgitada, milline neist on parem lahendus väikese kasutajate arvu ning suuremate andmemahtudega tegeleva rakenduse majutamiseks. Selle saavutamiseks võrreldi mõlemate taristute ülesseadmise keerukust, aega, rahalist kulu ning jõudlust. Võrdluse tulemusena selgus, et nii Raspberry Pi kui ka pilvepõhine lahendus olid jõudluse poolest sarnasel tasemel. Virtuaalmasina ülesseadmine oli küll kiirem, kuid sellegipoolest oli Raspberry Pi märkimisväärselt odavam ning selle ülesseadmine oli lihtsam. Väikese kasutajate arvuga ning suurte failidega tegeleva rakenduse majutamiseks on mõistlikum kasutada Raspberry Pi'd.

## **Võtmesõnad:**

Raspberry Pi, pilvtehnoloogia, Azure virtuaalmasin, jõudlustestimine, kuluanalüüs

**CERCS:** P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

# **Comparing Performance and User Experience between Raspberry Pi and Cloud Servers**

## **Abstract:**

In this thesis, a low-priced physical infrastructure Raspberry Pi is compared to an inexpensive cloud-based virtual machine. The aim of this work is to find out which of these is the best for hosting an application that deals with large files and has a small number of users. To achieve this, both of the infrastructures' setup time, difficulty, cost, and performance were evaluated and then compared. The results of this comparison show that both Raspberry Pi and the cloud-based solution had a similar performance. The virtual machine had a faster setup time, however, the Raspberry Pi had a significantly lower cost and was easier to set up. It is better to use Raspberry Pi for hosting an application that has a small user base and works with files.

## **Keywords:**

Raspberry Pi, cloud computing, Azure virtual machine, performance testing, cost analysis

**CERCS:** P170 Computer science, numerical analysis, systems, control

## Sisukord

<b>1. Sissejuhatus</b>	<b>4</b>
<b>2. Terminid</b>	<b>6</b>
<b>3. Taust</b>	<b>7</b>
3.1 Raspberry Pi	7
3.2 Pilvtehnoloogia	8
3.3 Varasemad uuringud	9
<b>4. Tehnoloogiate valik</b>	<b>11</b>
4.1 Taristud	11
4.2 Testitav rakendus	13
4.3 Jõudlustestimise tarkvara	15
4.4 Seire tööriistad	16
<b>5. Metoodika</b>	<b>18</b>
5.1 Kasutuslugu	18
5.2 Vaadeldavad näitajad	19
5.2.1 Ülesseadmine	19
5.2.2 Kulu	20
5.2.3 Jõudlus	20
<b>6. Koormustestide disain</b>	<b>22</b>
6.1 Testide olemus	22
6.2 Ettevalmistus	22
6.3 Testide disain	23
<b>7. Tulemused</b>	<b>25</b>
7.1 Ülesseadmine	25
7.2 Kulu	26
7.3 Koormustestide tulemused	28
7.3.1 Viie kasutajaga koormustest	28
7.3.2 Kümne kasutajaga koormustest	31
7.3.3 Micro-SD kaardiga läbiviidud koormustest	33
7.3.4 Koormustestide tulemuste analüüs	35
7.4 Tulemuste kokkuvõte	38
<b>8. Kokkuvõte</b>	<b>40</b>
<b>Viidatud kirjandus</b>	<b>42</b>
<b>Lisad</b>	<b>47</b>
Lisa 1. Repositoorium	47
Lisa 2. Failinimede kogumise skripti loomine	48
Lisa 3. Failide teisaldamise skripti loomine.	49
Lisa 4. Locusti testi skript	51
<b>Litsents</b>	<b>53</b>

## 1. Sissejuhatus

Arvuti ja interneti kasutusel on märkimisväärne roll tänapäeva ühiskonnas. Datareportali andmete kohaselt kasutab 2025. aasta veebruari seisuga internetti 67.9% inimestest maailmas [1]. Arvuti ning interneti ligipääs toovad inimesteni erinevaid veebilehti ja tarkvarasid mitmetest valdkondadest nagu haridus, kaubandus või pangandus. Statistikaameti andmetel veedab järjest rohkem eestlasi oma aega internetis ning võrreldes varasemate aastatega on ka nende arvutioskused täienenud [2]. Laialdane interneti kasutus võib soodustada järjest rohkemate rakenduste loomist ja kasutust. Ükskõik, kas rakendus on loodud ise või tahetakse kasutada mingit vabavaralist tööriista, mõlemal juhul peab rakenduse kasutamiseks seda kõigepealt kusagil majutama.

Rakenduse majutamiseks on mitmeid mooduseid. Jordani [3] sõnul on üheks võimaluseks majutada rakendust ettevõtte või isiku enda poolt üles seatud füüsilisel taristul. Teise võimalusena saab oma rakenduse üles seada pilvserveril. Valiku tegemisel mängivad rolli mitmed tegurid nagu jõudlus, kulu ning ülesseadmise aeg ja keerukus. Suurema jõudlusega taristul saab majutada ressursimahukamaid rakendusi. Lisaks suudab see teenindada rohkem kasutajaid. Jõudluse mõõtmiseks saab vaadeldavatel taristutel läbi viia jõudlusteste, mille abil otsustatakse, kumb neist on võimsam. Kuid mida võimsam on taristu, seda kallim see tavaliselt on. Seega tuleks arvestada kui suur on ühe või teise kulu, et maksta täpselt nii võimsa riistvara eest, kui vaja on ning mida eelarve võimaldab. Otsustamisel mängib rolli ka taristu ülesseadmise aeg, kui tahetakse näiteks kiiresti jõudlust mastabeerida (ingl *scaling*). Kui ülesseadjal on vähene kokkupuude kindla taristu või üleüldiselt nende ülesseadmisega, on oluline võtta arvesse ka keerukust, et ülesseadmine toimuks võimalikult probleemivabalt. Otsustamine sõltub paljudest erinevatest teguritest, mille vahel tuleks teha kompromisse, et leida sobiv lahendus.

Siinses töös keskendutakse odavate ning väiksema jõudlusega taristute uurimisele ning võrdlemisele. Sellised taristud on sobilikud näiteks hobiprojektide ning muude rakenduste majutamiseks, mille kasutajate arv on väike. Statistikaameti kohaselt tegelevad enamus arvutikasutajatest failide liigutamise ühest keskkonnast teise, näiteks piltide tõstmisega nutitelefonist arvutisse [2]. Samuti selgus 2020. aastal läbi viidud uuringus, et 71% uuringus osalejatest kasutab andme talletus teenuseid nagu Google Drive isiklike piltide salvestamiseks [4]. Seega võetakse siin töös võrdlemise aluseks rakendus, mis töötab suuremate andmemahutudega näiteks pildifailidega. Töö eesmärk on välja selgitada, milline

odavatest taristutest, on kõige mõistlikum suurte andmemahtudega töötava ning väikese kasutajate arvuga rakenduse majutamiseks, kas käsitsi üles seatud füüsiline taristu või pilvserver. Selleks uuritakse võimalikke taristuid, millel majutamist teostada ning valitakse välja kaks sobivamat, mida võrdlema hakatakse. Samuti uuritakse rakendusi, mis töötavad pildifailidega ning valitakse üks, mida majutama hakatakse. Seejärel seatakse taristud üles ning selle käigus vaadeldakse, kui aeganõudev ning keeruline oli kummagi protsess. Peale seda majutatakse välja valitud rakendust mõlemal taristul ning viiakse läbi jõudlustestid, mille eesmärk on välja selgitada, kumb neist on võimsam. Lisaks arvutatakse kokku mõlema taristu aastane kulu, eesmärgiga leida, kumb on odavam. Nende kolme teguri uurimuse tulemusena tehakse otsus, kumba eelistada.

Töö koosneb kaheksast peatükist. Peatükis 2 tuuakse välja asjakohased mõisted. Peatükis 3 antakse ülevaade kahte erinevat tüüpi taristu kontseptsioonist ning kirjeldatakse varasemaid uuringuid. Töös kasutatud tehnoloogiate valikut selgitatakse peatükis 4. Peatükk 5 annab ülevaate uurimuse läbiviimise plaanist ning peatükis 6 kirjeldatakse lähemalt jõudlusteste. Peatükis 7 tuuakse välja saadud tulemused ning peatükis 8 on töö kokkuvõte.

## 2. Terminid

**Hüperhargtöötlus** (ingl *hyperthreading*): Inteli protsessorite omanduslik rööpse mitmelõimelisuse tehnoloogia [5].

**Jõudlus** (ingl *performance*): Kvantitatiivne sooritusvõime, näiteks toimingute arv ajaühikus [5].

**Jõudlustest** (ingl *benchmark*): Riist- või tarkvara jõudluse hindamiseks mõeldud test [6].

**Kohtvõrk** ehk LAN (ingl *local area network*): Piiratud territooriumil paiknev arvutivõrk, kus arvutitevaheline kaugus ei ületa kilomeetrit [7].

**Koormusejaotur** (ingl *load balancer*): Vahend, mis jaotab võrgu- või rakenduseliiklust mitmele serverile, suurendades teenuse jõudlust ja käideldavust [5].

**Läbivestimine** (ingl *end-to-end testing*): Süsteemi testimine koos ta välisliidestega, st kasutaja vaatepunktist [5].

**Mastabeerimine** (ingl *scaling*): Objekti või protsessi talitlusulatus parameetrite (jõudluse, teenindusala, toodangumahu vms) suurusjärgu muutmine [5].

**Regressioontestimine**: Muudetud versiooni töö võrdlemine algse omaga, ühilduvuse kontrollimiseks ja võimalike uute vigade avastamiseks [5].

**Salvesti**: Funktsionaalüksus, millesse saab andmed paigutada, milles neid saab hoida ja millest neid saab võtta. [7]. Salvestid on näiteks kõvaketas ja pooljuhtketas ehk SSD.

**Viik**: (ingl *pin*): Elektroonikadetaili ühenduskontakt [5].

**Võrgusalvesti** ehk NAS (ingl *network attached storage*): arvutivõrku ühendatav kõvakettaga seade failide talletuseks ja ühiskasutuseks [5].

**Välisseade** (ingl *peripheral*): Digitaalseadme võimestiku laiendamiseks ta külge ühendatud seade [5].

**Importimine**: Piltide teisaldamine ajutisest kaustast põhikausta, mille käigus jäetakse vahele kõik duplikaadid. Peale seda toimub piltide sorteerimine nende loomisaja järgi ning nende indekseerimine [8].

**Indekseerimine**: Pildi metaandmete salvestamine ja pispildi (ingl *thumbnail*) genereerimine [8].

### 3. Taust

Selles peatükis tutvustatakse Raspberry Pi'd ja pilvtehnoloogiat ning seejärel antakse ülevaade varasematest uuringutest.

#### 3.1 Raspberry Pi

Ravikirani kohaselt on Raspberry Pi pangakaardisuurune arvuti, mis on välja arendatud Raspberry Pi Foundationi poolt 2012. aastal. See loodi eesmärgiga õpetada tõhusalt programmeerimist ja arvutitehnikat. Raspberry Pi seerias on ilmunud mitmeid mudeleid näiteks Raspberry Pi Zero ja Raspberry Pi 3 [9].

Riistvara osas on Ravikirani kirjutanud, et Raspberry Pi sarnaneb tavalise arvutiga, omades protsessorit, muutmälu ehk RAM-i ja graafikakaarti. Lisaks saab selle ühendada internetiga läbi WiFi või Ethernet pordi. Samuti on Raspberry Pi jaoks loodud ka spetsiaalne operatsioonisüsteem Raspberry Pi OS. Selle kasutamiseks saab Raspberry Pi külge ühendada monitori, klaviatuuri ja hiire [9]. On ka omadusi, mis Raspberry Pi'd tavalisest arvutist eristavad. Basumallicku kohaselt pole Raspberry Pi'l kaasas kõvaketast, millele andmeid salvestada. Seega kasutatakse salvestiks peamiselt SD-kaarti. Suurimaks erinevuseks Raspberry Pi ja tavalise arvuti vahel on see, et Raspberry Pi'l on olemas GPIO ehk üldotstarbelise sisend-väljundi viigud. Tänu neile saab Raspberry Pi külge ühendada erinevaid elektroonilisi seadmeid näiteks kaamera mooduleid [10].

Raspberry Pi jaoks leidub hulganisti kasutusvõimalusi. Üheks selliseks on näiteks Raspberry kasutamine liikumise tuvastamiseks. Selle funktsiooni loomiseks lisasid Kurkovsky ja Williams oma projektis Raspberry Pi külge infrapuna sensorid, mis andsid Raspberry Pi'le signaali, kui selle lähedusse sattus soojust kiirgav keha. Lisati külge ka kaamera moodul, et tuvastada, kes vaateväljas liigub. Sel moel omandab Raspberry Pi liikuva keha kohta kaadreid. Need saab hoiustada näiteks Raspberry Pi lokaalses mälus või laadida üles pilve. Sellisel funktsioonil on mitmeid kasutusjuhte, näiteks looduses metsloomade jäädvustamine või auto numbrimärkide tuvastamine [11].

Raspberry Pile võib peale sensorite ja muude moodulite külge panna ka välise kõvaketta. See võimaldab Raspberry Pi'd kasutada eraldiseisva salvestina. Sellist viisi lähenemisel on mitmed eelised teiste salvestamise võimaluste ees nagu andmete talletamine pilves või nutitelefonis salvestusruumis. Solihin bin Ahmad Nasser jt kohaselt on keskmisel arvuti kasutajal võimalik umbes ühe terabaidi jagu erinevaid andmeid oma seadmesse salvestada. Kui see maht täis saab, on üheks lahenduseks osta endale juurde salvestusruumi pilves. See

on aga tülikas ja kallid protsessid. Lisaks on oht, et keegi teine pääseb kasutaja andmetele ligi, kuna need on internetist kättesaadavad. Raspberry Pi salvesti lahendaks aga need probleemid. See on odavam kui pilvteenus ning kui salvesti pole internetist kättesaadav, on tagatud, et igaüks kasutaja failidele ligi ei pääse. Eelisenä saab välja tuua ka Raspberry Pi väikese suuruse, mis teeb salvesti kaasas kandmise mugavaks [12].

### 3.2 Pilvtehnoloogia

Google'i poolt väljastatud artikli kohaselt oli varasemalt ettevõtetal vaja soetada oma riistvara, mille peal saadi rakendus või muu teenus kasutajatele kättesaadavaks teha. See nõudis aga lisakulusid tööjõu näol, kes riistvara haldaks. Lisaks sellele oli ressursse keeruline mastabeerida, kuna vahepeal võis esineda hüppeid kasutajate nõudluses nii, et teenus ei suuda normaalselt talitleda ning vastupidiselt sellele võis tekkida olukordi, kus teenusele eraldati rohkem ressursse, kui viimane ära kasutada jõudis, tekitades juurde ebavajalikke kulutusi. Nendele probleemidele pakub lahendust pilvtehnoloogia. Pilvtehnoloogia võimaldab ligipääsu rakendustele, andmetele või arvutusressurssidele üle interneti. Pilvtehnoloogia üks suurimatest eelistest seisneb selles, et kasutaja maksab täpselt nii palju eest, kui ta ära kasutada jõuab [13].

Eelneva artikli andmetel saab pilvtehnoloogia teenuseid ehk pilvteenuseid liigitada vastavalt sellele, kui palju kontrolli on selle kasutajal, kui paindlik see on ning kuidas seda hallatakse. Peamised pilvteenuste mudelid on tarkvara teenusena ehk SaaS, platvormi teenusena ehk PaaS ning taristu teenusena ehk IaaS [13]. Geeksforgeeks artikli kohaselt on tegemist SaaS-iga, kui kasutaja ei pea tarkvara ise üles seadma ega kuidagi haldama, vaid saab seda kasutada igal pool ja igal ajal üle interneti. Selliseks teenuseks on näiteks Google Docs [14]. Kasutaja omab rohkem kontrolli, kui ta kasutab platvormi teenusena. See on peamiselt mõeldud tarkvara arenduseks, kus teenusepakkuja annab arendajale vajalikud tööriistad rakenduste loomiseks, jättes välja selleks vajamineva riistvara ning tarkvara ülesseadmise osa. PaaS on näiteks Amazon Web Services Elastic Beanstalk [15]. Kõige rohkem on kasutajal kontrolli ja paindlikkust, kui kasutatakse taristut teenusena. IaaS-i korral rendib teenusepakkuja kasutajale arvutusressursse näiteks virtuaalmasinate näol. Kasutaja saab teha otsuseid salvestusmahu, võrgunduse või muu sellise osas. Selle käigus ei pea ta aga muretsema füüsilise riistvara halduse eest [16]. Taristu teenusena on näiteks Microsoft Azure'i virtuaalmasin [17], mida ka siin töös kasutatakse.

Järgnevalt vaadeldakse lähemalt pilvtehnoloogia eeliseid ja puuduseid. Maddersi andmetel on üheks pilvtehnoloogia eeliseks paindlikkus ning mastabeeritavavus. Kui ettevõtte nõudmised vahepeal suurenevad makstakse sellel perioodil rohkem, kuid kui tekib olukord, kus ressursse nii palju ei kasutata, makstakse vähem. Pilvtehnoloogia eeliseks on ka andmete tsentraliseerimine, st pilve pandud andmed saab läbi interneti kätte ükskõik millisest seadmest. Seega kasutatakse pilve tihti varukoopiate hoiustamiseks. Pilvtehnoloogia kasutamine on suurtele ettevõtetele odavam võrreldes füüsilise riistvara soetamisega. See aga on olemas ettevõttest ja selle vajadustest, näiteks väiksel asutusel võib olla odavam oma taristu ülesseadmine. Pilvtehnoloogia puuduseks võib pidada selle turvalisust. Isegi kui teenusepakkuja poolt on võetud kasutusele tõhusad meetodid turvalisuse tagamiseks, võib kasutaja oma keskkonna valesti konfigurereida, mis võib nende rakenduse ohtu seada [18].

### **3.3 Varasemad uuringud**

Raspberry Pi ja pilveserveritega on läbi viidud mitmeid uurimusi. Järgnevad uuringud annavad arusaama, kuidas on varem lähenetud Raspberry Pi ja pilveserverite võimekuse hindamisele. Töödest tuuakse välja uurimiseks kasutatud lähenemisviisid, tehnoloogiad ja tulemused.

Laumets uuris, kui hästi tuleb Raspberry Pi toime veebiserverite majutamisega. Majutati Roundcube'i meiliserverit ja MediaDropi videoplatvormi. Raspberry Pi jõudluse hindamiseks serverite jooksumise ajal kasutati Apache jõudlustestide tööriista. Uurimuse käigus leiti, et Raspberry Pi sai tõrkevabalt kasutajate päringute serveerimisega hakkama. Päringute kiirust tõstis mitme Raspberry Pi kasutamine. Lisaks toodi välja, et pilveserverite kasutamine on kallim kui manuaalselt üles seatud server, mis jookseb Raspberry Pil [19].

Gamessi ja Hernandezi töös uuriti erinevate Raspberry Pi mudelite jõudlust ja võrreldi neid omavahel. Viidi läbi mitmeid mõõtmisi kasutades nendeks erinevaid tehnoloogiaid. Protsessori temperatuuri mõõtmiseks kasutati Stressberry tarkvara ja jõudluse testimiseks kasutati Sysbench jõudlustestide tarkvara. Lisaks hinnati failide tihendamise ja helifailide vormingu teisendamise kiirust. Need viidi läbi kasutades rakendusi 7-Zip (tihendamiseks) ja Phoronix Test Suite (vormingu teisendamiseks). Testiti ka muutmälu kiirust Geekbench ja STREAM jõudlustestimise programmide abil. Eeltoodud testidega sai kõige paremad tulemused Raspberry Pi 4B mudel, mis edestas vanemaid mudeleid nagu Raspberry Pi 3B+ ja Raspberry Pi Zero 2 W [20].

Tormi töös majutati Rackspace'i pilveserveril veebirakendust ning andmebaasi, mis algselt jooksid mõlemad ühel ja samal virtuaalmasinal. Uurimistöö eesmärk oli välja selgitada, kui hästi saab server kasutajate serveerimisega hakkama, kui koormust märkimisväärselt tõsta. Selleks viidi läbi jõudlustestid, kus analüüsiti rakenduse ja andmebaasi jooksutamist nii ühel kui ka erinevatel virtuaalmasinatel. Mitme virtuaalmasina korral kasutati ka koormusejaoturit ning automaatset mastabeerimist. Suure hulga kasutajate simuleerimiseks ja seejärel testitulemuste saamiseks kasutati tööriista Load Impact. Tulemuste põhjal järeldati, et kuigi üksik virtuaalmasin saab hakkama väikese hulga külastajatega, on nende arvu suurenemisel mõistlikum kasutada serveri koormuse jaotamise ning automaatse mastabeerimise tehnoloogiaid, et serveri töö ootamatult ei lakkaks ja kasutajate teenindamine liiga kaua aega ei võtaks [21].

Eelmainitud töödest selgub, et leidub mitmeid uuringuid, mis viivad läbi jõudluste nii Raspberry Pi'l kui ka pilveserveritel. Nende lähenemised annavad ülevaate, milline mudel on kõige suurema jõudlusega või kuidas jõudlust parandada. Lisaks tuuakse välja erinevaid viise, kuidas ning milliste tehnoloogiate abil saab jõudlust testida. Vähem keskendutakse erinevate serveri jooksutamise keskkondade alternatiivide kaalutlusele ja võrdlusele.

Erinevalt ülaltoodud uurimustest, keskendutakse siinses töös kahe erineva tehnoloogia võrdlemisele. Töös proovitakse välja selgitada, milline viis on jõudluse poolest parim valik rakenduse majutamiseks. Lisaks võetakse hindamisel arvesse ka ülesseadmise ajakulu, keerukust ning rahalist väljaminekut. Testimisel kasutatakse Raspberry Pi 5 mudelit, mis anti välja 2023. aastal ning mille jõudlust pole eelmainitud töödes uuritud.

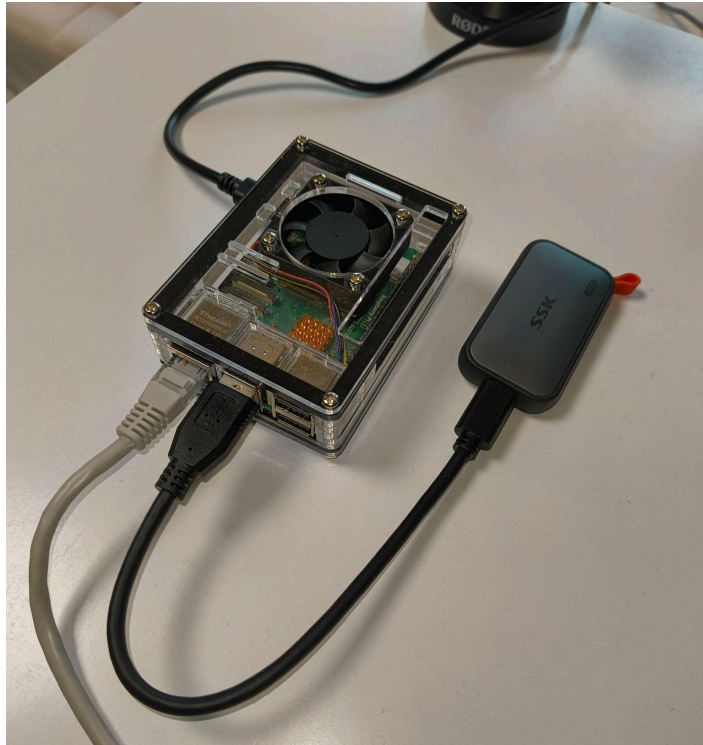
## 4. Tehnoloogiate valik

Siin peatükis antakse ülevaade kõikidest töös kasutatavatest tehnoloogiatest ning põhjendatakse nende valikut. Samuti kirjeldatakse nende konfiguratsioone.

### 4.1 Taristud

Kõigepealt valitakse välja sobiv Raspberry Pi mudel, mida töös kasutama hakatakse. Selleks toetutakse Fromage'i artiklile, kus võrreldakse erinevaid Raspberry Pi mudeleid omavahel. Seal tuuakse välja, et üleüldiselt on projektideks kõige mõistlikum kasutada 2025. aasta seisuga seeria uusimat mudelit Raspberry Pi 5. Tänu võimsamale protsessorile, on selle jõudlus võrreldes varasemate mudelitega parem [22]. Seega kasutatakse siin töös Raspberry Pi 5'te.

Järgnevalt kirjeldatakse lähemalt Raspberry Pi 5'e konfiguratsiooni. Raspberry Pi 5 kasutab Broadcomi BCM2712 nelja tuumalist Arm Cortex-A76 protsessorit [23]. Siin töös kasutatakse nelja gigabaidist muutmälu, kuna majutatav rakendus ei ole väga nõudlik arvutusressursi suhtes. Seega pole vaja teha lisakulutusi suurema muutmälu jaoks. Raspberry Pi'le pandi juurde ka mitmed lisad. Suurel mahul andmete talletamiseks otsustati kasutada 500 gigabaidist SSK poolt toodetud välist SSD-d [24]. Lisaks SSD-de suuremale salvestusmahule, on neil allika kohaselt võrreldes Micro-SD kaartidega ka kiiremad lugemis- ja kirjutamisoperatsioonide ajad [25]. Seetõttu konfigureeriti alglaadimine (ingl *booting*) toimuma SSD pealt. Antud mudeli elektritarbimine on suurem võrreldes varasemate mudelitega, seega kasutatakse selle toiteks spetsiaalselt Raspberry Pi 5'e jaoks mõeldud Qoltec Super FAST PD adapterit [26]. Lisaks varustati Raspberry Pi Miuzei korpuse ning jahutiga, et vältida ülekuumenemist [27]. Töö viimases, Micro-SD kaardiga läbiviidavas testis, kasutatakse alglaadimiseks ja andmete talletuseks Netaci 32-e gigabaidist C10 kiirusklassi Micro-SD kaarti [28]. Üles seatud Raspberry Pi kasutab operatsioonisüsteemi Raspberry Pi OS Bookworm, kuna see on optimeeritud spetsiaalselt selle riistvara jaoks [29]. Raspberry Pi ühendub võrku läbi CAT5-kaabli. Kokku pandud Raspberry Pi on toodud joonisel 1.



Joonis 1. Töös kasutatud Raspberry Pi 5 koos kõigi lisadega.

Järgnevalt uuritakse võimalikke pilvepõhiseid valikuid rakenduse majutamiseks. Slingerlandi [30] sõnul on 2024. aasta seisuga pilveteenuse pakkujate turuliider Amazon Web Services ehk AWS. AWS pakub kõige laiemat valikut erinevatele IaaS, PaaS ning SaaS teenustele nagu Amazon EC2 virtuaalmasin [31] ja Amazon Lambda [32], mida kasutatakse serverita funktsioonide juurutamiseks (ingl *deployment*). Perry artikli kohaselt on AWS-i eeliseks ka selle paindlikkus, pakkudes taristu haldajatele suuremat kontrolli oma teenuste üle. Samas tekitab see juurde keerukust, mis võib teenuste üles seadmise algajatele raskeks muuta [33].

Üheks alternatiiviks AWS-ile on Google Cloud Platform ehk GCP [34]. Samas artiklis tuuakse välja, et GCP pakub lihtsamat konteinerite halduse teenust ning selgemat hinnapoliitikat võrreldes teiste alternatiividega. Kuigi GCP-l on vähem teenuseid ning kohandamise võimalusi, peetakse seda tänu sellele kergemaks ning paremaks valikuks algajatele [33].

Siinses töös kasutatakse pilvteenusete pakkujana Microsoft Azure'i [35]. Kishore'i andmetel on Azure AWS-i kõrval populaarsuselt teine. Ka nemad pakuvad taristu haldajale suurt teenuste valikut ning kohandamisvõimalusi [36]. Töös kasutatakse Azure'i, kuna see pakub võrreldes AWS-i ja GCP-ga kõige odavamaid lahendusi õpilastele.

Microsoft Azure'i teenustest otsustati kasutada virtuaalmasinat, kuna selle ülesseadmine on võrreldes teiste teenustega nagu Azure App Service [37] kõige sarnasem Raspberry Pi'ga. Virtuaalmasina suuruseks valiti Standard B2pls v2, millel on neli gigabaiti muutmälu ja kaks vCPU-d ehk kaks virtuaalset protsessorit, mis on loodud hüperhargtöötuse abil [38]. Sellel on ARM64-arhitektuur ning operatsioonisüsteemiks on Debian 12. Valiti välja just selline konfiguratsioon, sest see on nii riistvara kui ka arhitektuuri poolest sarnane Raspberry Pi'ga. Virtuaalmasina loomisel valiti selle regiooniks Põhja Euroopa ning ülejäänud valikud jäid vaikeväärtustega. Andmete hoiustamiseks otsustati kasutada andmetalletuse teenust Blob Storage [39]. Sinna loodi omakorda konteiner, kuhu rakendus andmeid salvestama hakkab. Virtuaalmasina ja Blob Storage'i ühendamiseks kasutati BlobFuse2-te [40]. BlobFuse2 võimaldab Blob Storage'i konteinereid mauntida (ingl *mount*) virtuaalmasina külge nii, et kasutaja saab konteineri andmeid manipuleerida nii nagu need oleksid osa kohalikust failisüsteemist [41].

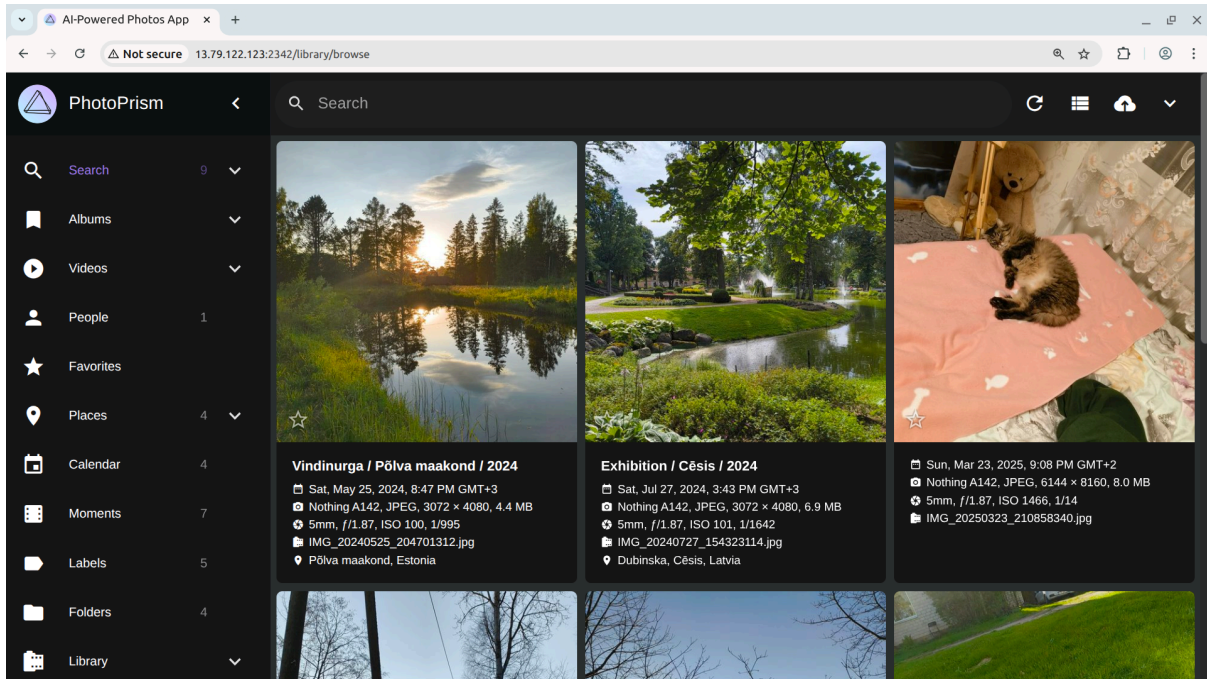
## 4.2 Testitav rakendus

Selles peatükis valitakse välja sobiv rakendus, mida jõudlustestimise käigus kasutama hakatakse. Rakendus peaks tegelema suurte andmemahtude nt pildifailide hoiustamisega. Samuti peab seda olema võimalik majutada omal käel ülesseatud serveril.

Selliseks rakenduseks on näiteks Nextcloud Files, kus on võimalik talletada mitmeid tüüpi faile nagu pilte või tekstidokumente. Nextcloudi kasutavad tuhanded organisatsioonid, sh riigiasutused, kuna lisaks andmete salvestamisele pakutakse ka teisigi teenuseid nagu Nextcloud Office tekstitöötlusprogramm või Nextcloud Talk keskkond videokonverentside tegemise jaoks [42]. Kuna selles töös on vaja tegeleda ainult piltide hoiustamise ja haldusega, ei oma Nextcloudi teenuste rohkus suurt tähtsust ning lisab nende näol juurde ebavajalikku keerukust. Järgnevalt uuritakse lähemalt ainult piltide ja videote haldusele keskenduvaid rakendusi.

Üheks populaarsemaks neist on Immich, mis on mõeldud majutamiseks isiklikul taristul. Sellel on olemas enamus pildigaleriis olevaid funktsionaalsuseid nagu albumite loomise võimalus, piltide otsing ja filtreerimine ning näotuvastus. Samuti ei pea nutitelefonis kasutama sellele ligipääsemiseks brauserit, vaid saab alla tõmmata vastava mobiilirakenduse, mis pakub paremat kasutuskogemust. Immichi puuduseks on see, et kuna 2025. aasta seisuga on see veel arengufaasis, võib seal esineda programmivigu (ingl *bug*). Seetõttu ei soovitata kasutada seda ainsa failide varunduse lahendusena [43]. Seega kasutatakse selles töös

rakendust Photoprism [44], mis on võrreldes Immichiga vanem ning stabiilsem. Photoprismi kasutajaliidest on näha joonisel 2.



Joonis 2. Photoprismi kasutajaliides.

Photoprismis on mitmeid funktsionaalsusi nagu piltide üles- ja allalaadimine, võimalus jagada neid albumitesse ning sildistada neid. Leidub ka lisafunktsioone näiteks slaidilugude loomine. Samuti saab seal hallata kasutajate rolle, mis tähendab, et näiteks vaataja rollis kasutaja ei saa pilte üles laadida. Photoprism pakub nii tasulisi kui ka tasuta versioone. Järgnevalt uuritakse lähemalt, kuidas oli Photoprism sätestatud selles uurimuses.

Töös kasutatakse Photoprism Community tasuta versiooni. Selle ülesseadmiseks kasutatakse Docker Compose'i, mis võimaldab ühe ainsa konfiguratsioonifaili abil defineerida ning seejärel käivitada korraga mitu Docker'i konteinerit [45]. Töös kasutatakse serveri käivitamiseks Photoprismi dokumentatsioonis Raspberry Pi jaoks mõeldud näidis Docker Compose faili. Konfiguratsioonifailis muudeti testimise lihtsustamise eesmärgil autentimise režiim (ingl *public mode*) avalikuks. Kui konteiner on jooksmas pandud, pääseb Photoprismi rakendusele ligi pordilt 2342. Lisaks eemaldati läbi kasutajaliidese kvaliteedifilter (ingl *quality filter*) selleks, et halvema kvaliteediga pilte ei eraldataks ülejäänud piltidest erinevasse kausta. Töös on tähtis, et kõik pildid oleks samas kaustas, et simuleerida kõigi nende kuvamist.

### 4.3 Jõudlustestimise tarkvara

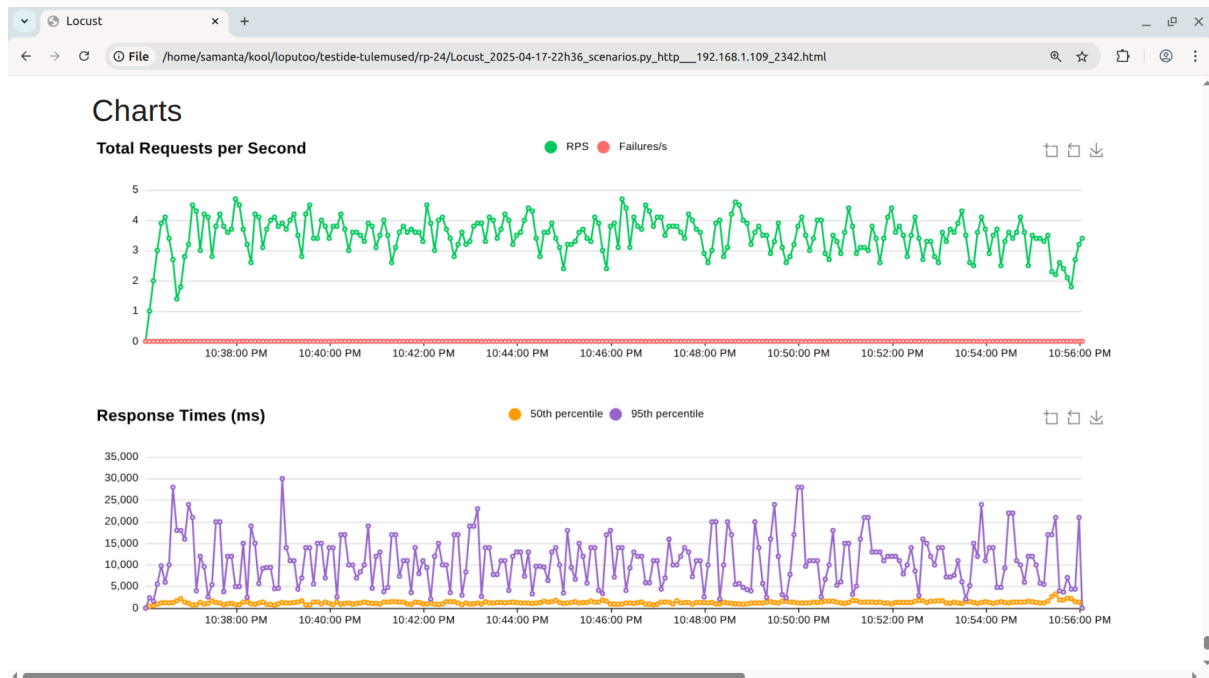
Selles peatükis uuritakse erinevaid tarkvarasid, mille abil saab läbi viia jõudlusteste ning valitakse neist välja sobiv. Järgnevalt antakse ülevaade kolmest jõudlustestimise tarkvarast, mis võimaldavad koormustestimist. Coheni kohaselt aitavad koormustestid analüüsida, kas testitav server saab hakkama eeldatava kasutajate arvu teenindamisega. Koormustestimise käigus luuakse hulk sünteetilisi kasutajaid, mis hakkavad tegema päringuid testitavale serverile. Nii simuleeritakse suurt koormust ning seejärel analüüsitakse, kas server on piisavalt võimas ning saab oma tööga suure koormuse all olles hakkama [46].

Apache JMeter [47] tarkvara võimaldab koormustestida erinevaid tüüpi rakendusi, servereid või protokolle, näiteks REST-põhist veebiteenust, andmebaase või tavalist kehtaskripti (ingl *shell script*). Selleks, et luua vastavad testimallid (ingl *test cases*) ehk stsenaariumid, mille järgi sünteetilised kasutajad päringuid tegema hakkavad, kasutatakse JMeteris peamiselt graafilist kasutajaliidest. Testi saab jooksutada kasutajaliidestest või käsurealt ning selle lõpus genereeritakse aruanne HTML-i failina. Aruandes on toodud testide tulemuste graafikud ja statistika näiteks selle kohta, kui kaua võttis päringute täitmine keskmiselt aega. Seega on autori arvates JMeter sobilik tööriist neile, kes ei soovi oma rakenduse testimiseks skripte kirjutada.

Teine koormustestimise tarkvara on k6 [48]. See toetab peale koormustestimise ka teisi testimistüüpe nagu läbivtestimist (ingl *end-to-end testing*) ning jõudluse regressioontestimist. Erinevalt JMeter tarkvarast, toimub k6 tööriistaga testide loomine Javascripti programmeerimiskeele abil, kus kõik testimallid defineeritakse koodi tasemel. Kui k6-e kasutajal on kokkupuude Javascripti keelega, aitab eelnev kogemus tal kiiremini esimesi k6-e teste luua. Lisaks sellele on k6-l üle 50-ne erineva laienduse (ingl *extension*), mis võimaldavad näiteks testide tulemuste vormingut muuta [49]. Kuna tarkvara pakub palju kasutusvõimalusi, võib see tekitada juurde ka keerukust.

Koormustestimise tööriist Locust [50], toimib sarnaselt k6-le, pakkudes kasutajale võimalust kõik testimallid ja konfiguratsioonid defineerida ühe skriptina. Erinevalt k6-st kasutab Locust programmeerimiskeelt Python. Locusti eeliseks on see, et testi jooksutamise tulemusi saab ilma laiendusteta reaalajas kasutajaliidestest jälgida. Andmed on visualiseeritud graafikutel ning lisaks saab testi käigus silma peal hoida ka logidel, et tuvastada probleeme. Testi lõppedes koostatakse saadud tulemustest aruanded, mis näitavad erinevat statistikat näiteks õnnestunud ja ebaõnnestunud päringute kohta, päringute täitmise keskmiseid aegu ning

läbilaskevõimet. Aruanded on eksporditavad CSV-failidena kui ka kokkuvõtliku HTML-failina. Locusti aruande näide HTML-failina on toodud joonisel 3.



Joonis 3. Locusti aruanne HTML-failina.

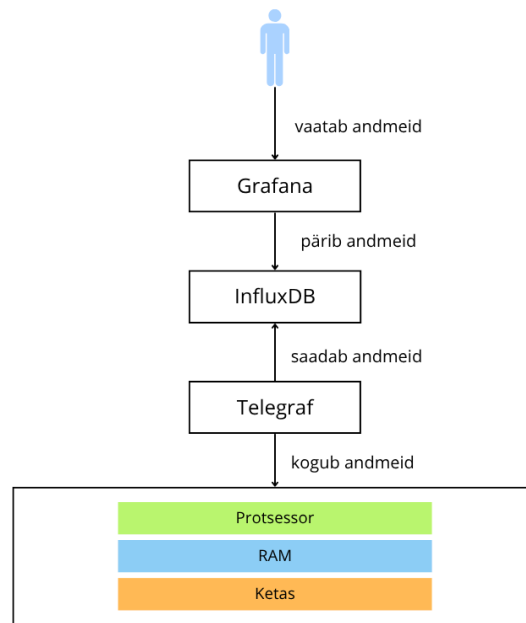
Joonisel 3 on näha, kuidas visualiseeritakse graafikuid läbilaskevõime ning päringu täitmise aegade kohta. Kuna Locust võimaldab teste luua Pythoni skriptide abil ning pakub mugavat viisi testi tulemuste jälgimiseks ning eksportimiseks, kasutatakse siin töös just seda.

#### 4.4 Seire tööriistad

Järgnevalt uuritakse, milliste tehnoloogiatega on võimalik süsteemi seiret teostada ning selgitatakse väljavalitud lahendust. Taristu seiret teostatakse tavaliselt mitme erineva tööriista abil. Üheks neist võib olla näiteks eraldiseisev andmete kogumise agent, st programm, mis kogub sissetulevaid andmeid ning suunab need edasi järgmisele komponendile näiteks andmebaasile [51]. Üldiselt kasutatakse seirel aegridade andmebaasi (ingl *time series database*) ehk andmebaasi, mis salvestab kogutud andmeid koos nende esinemise või toimumise ajaga [52]. Peale andmete salvestamist tuleks kasuks tarkvarast, mis saadud tulemusi nende analüüsimise eesmärgil visualiseeriks.

Siinses töös kasutatakse süsteeminäitajate kogumise agendina Telegrafi [53]. Selle eelisteks alternatiivide nagu Prometheus [54] kõrval on konfigureerimise lihtsus ja madal ressursikasutus. Samuti on selle jaoks loodud üle 300 plugina, mis pakuvad erinevaid viise andmete vastuvõtmiseks, muutmiseks või väljasaatmiseks [55]. Andmete salvestamiseks

kasutatakse InfluxDB [56] aegridade andmebaasi, mis on mõeldud näitajate või logide reaalajas talletamiseks. Süsteeminäitajate tõhusamaks analüüsimiseks kasutatakse Grafanat [57], mis võimaldab salvestatud andmeid visualiseerida. Selleks saab Grafanas luua koondpaneeli (ingl *dashboard*), mis sisaldab vaadeldavaid süsteeminäitajaid. Koondpaneeli saab kohandada vastavalt vajadustele, lisades sinna eritüüpi graafikuid või kujundades üldist väljanägemist. Seire komponentide ülesehitust illustreerib joonis 4.



Joonis 4. Seire komponentide ülesehitus.

Siinses töös kasutatakse Provosti [58] poolt loodud Docker Compose'i konfiguratsioonifaili, kuhu on kokku pandud kõik eelmainitud tööriistad nagu Telegraf, InfluxDB ja Grafana. Konfiguratsioonifaili abil on võimalik kõik kolm teenust ühe käsuga üles seada. Kui tööriistad on ülesseatud kasutatakse Rivera poolt loodud Grafana koondpaneeli [59], mis pakub valmisolevat lahendust Linux-i põhise süsteemi näitajate visualiseerimiseks. Grafana kasutajaliidesele pääseb ligi pordilt 3000.

## 5. Metoodika

Selles peatükis luuakse kasutuslugu, et selgitada miks ja kuidas kasutatakse testitavat rakendust päriselulises olukorras. Samuti aitab kasutuslugu seada eesmärged jõudluse hindamiseks. Peatüki teises pooles antakse ülevaade kõigist näitajatest, mida mõõdetakse ning seejärel võrdlemisel kasutatakse.

### 5.1 Kasutuslugu

Töös majutatakse eespool mainitud pildigalerii rakendust Photoprism. Kuna isiklikku pildigaleriid kasutab peamiselt selle omanik, tehakse oletus, et galeriid kasutab enamasti ajast korraga 1-2 inimest. Sellegipoolest võib tekkida olukordi, kus tahetakse pilte jagada teiste inimestega, kes tehtud fotodel esinevad või soovivad neid lihtsalt näha või jagada. Seetõttu eeldatakse, et siinset pildigaleriid võib kasutada väiksematel ajavahemikel ka kuni 10 inimest. Kasutajad viivad rakenduses läbi mitmeid toiminguid. Nendeks on piltide üles- ja allalaadimine, nende vaatamine ning filtreerimine. Pildigaleriile pääseb ligi ainult kohtvõrgust. Nii tagatakse üleslaetud piltide parem privaatsus. Sellise stsenaariumiga proovitakse simuleerida päriselulist rakenduse kasutust.

Selleks, et otsustada, milline keskkond on parim pildigalerii majutamiseks, tuleks seada eesmärgid rakenduse jõudluse osas. Üldine eesmärk rakenduse majutamisel on tagada selle hea kasutuskõlblikkus suuremate kasutajate arvu korral (siinses töös 10). Nielsen on kirjutanud, et kui mingi tegevus rakenduses võtab sooritamiseks aega ühe sekundi, siis kasutaja tähelepanu selle aja sees ei kao ning ta tunneb, et rakenduses navigeerimine on kiire [60]. Seega võiks siinse töö pildigaleriis üle saja pildi laadimine toimuda vähem kui kahe sekundiga. Nielsen kohaselt hakkab kasutaja kaotama tähelepanu, kui rakenduses tehtud tegevuse ooteaeg on pikem kui 10 sekundit. Sellised tegevused vajaksid edenemisriba (ingl *progress bar*), et parandada kasutajakogemust [61]. Photoprismi pildigaleriis on piltide üleslaadimist kasutajaliideses illustreeritud edenemisribaga, seega võib järeldada, et pildi üleslaadimine on pikem protsess ning võib võtta aega üle 10-nd sekundi. Üleslaadimise muudab ajamahukaks see, et pildifailid on tavaliselt üpris suured ning Photoprismi pildigaleriis toimub pildi üleslaadimise ajal selle indekseerimine ja importimine [8]. Siinses töös võiks kümne aktiivse kasutaja korral ühe pildi üleslaadimine toimuda vähem kui kümne sekundiga. Pildi allalaadimise aeg sõltub samuti pildi suuruselt, kuid see ei nõua tavaliselt enam serveri poolset töötlust. Seega võiks pildi allalaadimine toimuda vähem kui viie

sekundiga. Eelnevalt mainitud stsenaariumid ning nende ajalised eesmärgid on toodud tabelis 1.

Stsenaarium	Ajaline eesmärk (s)
Kõikide piltide laadimine	2
Ühe pildi allalaadimine	5
Ühe pildi üleslaadimine	10

Tabel 1. Stsenaariumite ajalised eesmärgid

Järgmises peatükis antakse ülevaade näitajatest, mida töös uurima hakatakse.

## 5.2 Vaadeldavad näitajad

Raspberry Pi ja Azure'i virtuaalmasina võrdlemisel pööratakse tähelepanu kolmele aspektile. Need on taristu ülesseadmise keerukus ja aeg, selle maksumus ning jõudlus. Selles peatükis kirjeldatakse, kuidas mainitud külgi uurima hakatakse ning selgitatakse nende olemust ja tähtsust võrdluse tegemisel.

### 5.2.1 Ülesseadmine

Isiklikuks tarbeks või väikese kasutajate arvuga rakenduse ülesseadjal ei pruugi olla varasemat kogemust rakenduste majutamise kohta Linuxil põhisel taristul. Samuti võib tekkida olukordi, kus on vaja rakenduse töö taastada, kui virtuaalmasin kokku jookseb või Raspberry Pi SD-kaart korrumpereb. Nendel põhjustel vaadeldakse selles töös kui keeruline on ühel või teisel taristul rakendust majutada ning kui kiiresti saab selle tööle.

Töös hinnatakse keerukust järgmise nelja tasemega:

1. Lihtne - keskmise arvutioskusega inimene, kes ei puutu igapäevaselt kokku spetsiifilisemate operatsioonisüsteemiga seotud ülesannetega, saab tegevusega hakkama.
2. Keskmine - inimene, kes on kokku puutunud Unixi käsureaga (ingl *command line*) saab ülesseadmisega hakkama.
3. Raske - inimene, kellel on kogemus Unixi käsureaga ning kes omab üldiseid teadmisi arvuti riistvarast, saab tegevusega hakkama.
4. Väga raske - inimene, kellel on kogemus Unixi käsureaga ja kellel on laiemad teadmised arvuti riistvarast ning arhitektuurist, saab ülesseadmisega hakkama.

Sel moel keerukuse hindamine aitab kategoriseerida, milliste teadmiste ja oskustega inimesed saavad hakkama vaadeldava taristu ülesseadmisega. Siinses töös on keerukuse hindamine subjektiivne. Ülesseadmise vältel märgitakse üles keerulisemad etapid ning vaadeldakse, kui palju peab eeltööd ja lisalugemist tegema, et taristu üles seada.

Lisaks keerukusele uuritakse, kui kaua võtab aega kummagi taristu ülesseadmine. Aega vaadeldakse eraldi keerukusest, kuna tegevuse käigus võivad mängida rolli inimese oskustest sõltumatud tegurid. Siinses töös mõõdetakse aega päevades.

Ülesseadmise käsitlusalasse kuulub Raspberry Pi puhul riistvara omandamine ja kokkupanemine koos Dockeri [62] konteinerite püsti panemisega. Virtuaalmasina korral arvestatakse siin käsitlusalasse sobiva masina otsimist, selle algset konfigureerimist Azure'i portaalis ja Dockeri konteinerite jooksutamist. Ülesseadmise keerukuse ja selleks kulunud aja mõõtmise eesmärgiks on välja selgitada, millise taristu saab kõige lihtsamini ja kiiremini üles seatud.

### **5.2.2 Kulu**

Raspberry Pi ja virtuaalmasina võrdlemisel arvestatakse ka mõlema taristu üles seadmisele kulunud rahalist maksumust. See on oluline tegur, mida valiku tegemisel kaaluda, kuna üksikisiku poolt loodud rakendused või muud hobiprojektid ei ole tavaliselt mõeldud tulu tootmiseks. Seega on tähtis, et taristu, millel rakendus jookseb oleks võimalikult odav.

Raspberry Pi serverina kasutamise jooksva kulu hindamiseks liidetakse kokku kõikide vajalike riistvara osade hinnad. Samuti arvutatakse sinna juurde Raspberry Pi elektrikulu. Azure'i virtuaalmasina korral arvutatakse kõikide kasutatatud teenuste (sh andmete talletuse teenus) maksumused kokku ühes kuus. Selleks, et võrrelda saadud summasid omavahel, arvutatakse, kui palju läheb maksuma ühe või teise taristu kasutamine aastas. Maksumuse uurimise eesmärk on välja selgitada, kui kallid on vaadeldavad taristud ning milline neist on odavam.

### **5.2.3 Jõudlus**

Üks olulisematest teguritest sobiva taristu valimisel on jõudlus, kuna see tagab rakenduse korraliku töötamise. Sellel põhjusel uuritakse siinses töös Raspberry Pi ja Azure'i virtuaalmasina jõudlust. Kui mõlema taristu peal on pildigalerii rakendus jooksva pandud, hakatakse seda koormustestima. Fahimi kohaselt on koormustestimine oluline selleks, et rakenduse ülesseadja oleks teadlik, kas ja kui hästi tuleb tema rakendus toime suure hulga

kasutajatega ning teha saadud tulemuste põhjal vastavaid järeldusi ja muudatusi [63]. Siinses töös kasutatakse koormustestide jooksutamiseks Tartu Ülikooli poolt antud sülearvutit HP EliteBook 840 G8, mis on varustatud Intel Core i5 protsessori ja 16 GB-i muutmäluga. Arvuti operatsioonisüsteem on Ubuntu 22.04 ning see on ühendatud samasse kohtvõrku koos uuritava Raspberry Pi'ga. Koormustestimiseks kasutatakse tööriista Locust. Kui testid on läbi viidud analüüsitakse saadud tulemusi testi lõpus genereeritud aruande põhjal. Uuritakse kahte näitajat: päringute arvu sekundis ja päringu täitmise aega. Analüüsi põhjal proovitakse välja selgitada, milline taristu reageerib kasutajate päringutele kiiremini.

Testide jooksutamise vältel toimub ka süsteemi seire. Pandora FMS tiimi kohaselt aitab seire varakult tuvastada süsteemis kitsaskohti, mis võivad tulevikus rakenduse kasutamist negatiivselt mõjutada [64]. Siinses töös on seire eesmärk välja selgitada, kui ressursimahukas on pildigalerii jooksutamine, kui seda kasutab korraga palju kasutajaid. Samuti tahetakse teada saada, kas leidub mingeid eraldiseisvaid riistvara komponente, mis pärsivad kogu süsteemi normaalset toimimist. Seiramilisel jälgitakse näitajaid nagu protsessori ja muutmälu kasutus ning ketta sisend/väljundi (ingl *disk I/O*) tööaeg. Need annavad ülevaate süsteemi tähtsamate osade tervisest.

## 6. Koormustestide disain

Koormustestimist viiakse läbi kasutades Pythoni [65] skripte. Pythoni pakett Locust võimaldab defineerida erinevaid stsenaariumeid, mida sünteetilised kasutajad pildigalerii rakenduses sooritama hakkavad. Stsenaariumid koosnevad ühest või mitmest ülesandest, mis hõlmavad päringute tegemist rakendusele kui ka abifunktsioonide täitmist. Kõik testimiseks kasutatud skriptid on esitatud GitHubi repositooriumis, mis on kättesaadav lisas 1.

### 6.1 Testide olemus

Testide käigus edastatakse pildigaleriise peamiselt nutitelefoni kaameraga tehtud fotosid. Nende suurused varieeruvad ühest megabaidist kuni 18-ne megabaidini. Iga testi alguses on galeriis kokku 100 pilti.

Töös viiakse nii Raspberry Pi kui ka virtuaalmasina peal läbi kaks koormustesti. Esimene test kestab 20 minutit ja hõlmab viit sünteetilist kasutajat. Koormuse valik on tehtud toetudes kasutusloole, kus otsustati, et kuni kümme inimest võib väiksel ajavahemikul galeriid kasutada, kuid kuna taristute riistvara pole väga võimas, viiakse kõigepealt läbi väiksema kasutajate arvuga koormustest. Teises testis simuleeritakse kümmet kasutajat ning ka selle testi kestus on 20 minutit. Eelmainitud testide abil proovitakse analüüsida, kui hästi saavad taristud hakkama väiksema ja suurema arvu kasutajatega ning kas ja kui suurel määral erinevad saadud tulemused üksteisest.

### 6.2 Ettevalmistus

Selleks, et testid annaksid igal jooksumisel tõele vastavaid ja reprodutseeritavaid tulemusi, peaks iga test olema sõltumatu varasemate testide tulemustest ning kõrvalmõjudest. Viimasteks võivad olla üles laetud pildid või andmed vahemälus. Seetõttu kustutakse iga testi lõppedes manuaalselt kõik testimise käigus lisatud pildid galeriist ära. See meede ei pruugi aga olla piisav. Photoprismi kohaselt võib serveri vahemälu hoiustada kasutamata pisipilte või indekseid [66]. See võib tähendada, et ka kustutatud pildid võivad maha jätta jäänukeid pisipiltide või indeksite näol. Kui ühe testi käigus on kasutatud üleslaadimiseks sama pilti, mida varemgi, võib vahemälu tõttu olla uue testi tulemus jõudluse poolest parem kui tavaliselt.

Probleemi ennetamiseks laetakse igas testis üles ainult selliseid pilte, mida varem kasutatud pole. Nii ei teki kahtlust, et pildid on rakenduse vahemälus juba olemas. Selleks luuakse kaks Bashi [67] skripti, mis aitavad enne ja pärast testi jooksumist kasutatud pilte hallata. Kõigepealt luuakse saadaval olevate piltide kaust nimega „pildid“, kuhu lisatakse manuaalselt

kõik pildid, mida testimiste jooksul üles laadima hakatakse. Esimene skript „leiaPiltideNimed“ loeb kõikide eelmainitud kaustas olevate failide nimesid ning salvestab need uude faili „piltideNimed“. Skript on genereeritud tehisintellekti ChatGPT o3-mini [68] poolt, mille viip ja tulemus on saadaval lisas 2. Skripti peab jooksutama enne igat testi. Testi käivitamisel kasutatakse loodud faili „piltideNimed“, et leida süsteemist üles pilt, mida galeriisse üles laadima hakatakse. Testimise käigus logitakse vastavasse logifaili iga üleslaetud pildi nimi. See on vajalik järgmise skripti jaoks, mis käivitatakse manuaalselt siis, kui test on lõppenud. Skript „teisaldaKasutatudPildid“ otsib viimati jooksnud testi logifailist üles kõikide üleslaetud piltide nimed ning liigutab need kaustast „pildid“ kausta „kasutatudpildid“. Skript on genereeritud tehisintellekti DeepSeek R1'e [69] poolt, mille viiba ning tulemuse leiab lisast 3. See meede tagab, et järgmine test ei saa kasutada varasemalt üles laetud pilte, mis omakorda mõjutaks testi tulemusi.

### 6.3 Testide disain

Locusti testide kood on genereeritud ChatGPT o3-mini-high [68] poolt, mille viip ja tulemus on saadaval lisas 4. Võimalikult reaalsete tulemuste saamiseks on loodud kolm stsenaariumit.

Esimene stsenaarium hõlmab piltide sirvimise ning filtreerimise simuleerimist. Kõigepealt tehakse päring, mis tagastab kõik galeriis olevad pildid. Seejärel saadetakse rakendusse päring, mis tagastab kasutajale ainult need pildid, mille põhi värvitooniks on hall. Filtreeritakse halli värvi pilte, kuna neid on palju kasutatud piltide seas.

Teine stsenaarium on pildi allalaadimine. Selleks, et allalaadimist teostama hakata, peab kõigepealt leidma juhusliku pildi koos selle identifikaatoriga. Selle jaoks tehakse päring, mis tagastab kõik galeriis olevad pildid. Saadud vastusest ekstrahitakse juhusliku pildi identifikaator. Identifikaatorit kasutatakse allalaadimise päringu koostamiseks, mis välja saatmisel tagastab kõik vastava pildi kohta käivad andmed.

Viimane stsenaarium on pildi üleslaadimine. Selles ülesannete komplektis genereeritakse kõigepealt abifunktsiooniga juhuslik kuue sümboli pikkune jada, millest saab pildi identifikaator. See on vajalik pildi üleslaadimiseks ja hiljem töötlemiseks. Seejärel saadetakse päring kõikide piltide galeriist kättesaamiseks. Nii simuleeritakse pildigalerii avamist ja piltide sirvimist enne üleslaadimise sooritamist. Peale seda valitakse varem koostatud piltide nimekirja järjendist juhuslik pildifaili nimi ning selle järgi otsitakse süsteemist välja õige pildifail. Pildifail lisatakse päringu lasti (ingl *payload*). Päringu saatmiseks on vaja ka kasutaja identifikaatorit, sest ainult kindlatel kasutajatel on õigus pilte üles laadida. Seejärel

koostatakse POST-päring, mis saadab välja valitud pildi rakendusse. Photoprismi otspunkt, kuhu pilt saadetakse võib välja näha selline: <http://192.168.1.146:2342/api/v1/users/ustwjk273i6ivct8/upload/a2y1g9>. Peale seda luuakse sama pildi identifikaatorit kasutades ka PUT-päring, mis on vajalik pildi töötlemiseks ehk importimiseks ja indekseerimiseks. Pildi üleslaadimise funktsioon on toodud koodiplokis 1.

```
def upload_file(self, upload_id):
    file_name = random.choice(file_list)
    logging.info("Photo: %s", file_name)
    file_path = os.path.join("./pildid", file_name)

    with open(file_path, "rb") as f:
        file_content = f.read()

    files = {"files": (file_name, file_content, "image/jpeg")}

    url = f"/api/v1/users/ustwjk273i6ivct8/upload/{upload_id}"
    with self.client.post(url, files=files, catch_response=True) as res:
        if res.status_code != 200:
            res.failure("POST-päring ebaõnnestus pildi üleslaadimisel!")

    put_payload = {"albums": []}
    with self.client.put(url, data=json.dumps(put_payload), catch_response=True) as
put_res:
        if put_res.status_code != 200:
            put_res.failure("PUT-päring ebaõnnestus pildi üleslaadimisel!")
```

Koodiplokk 1. Pildi üleslaadimise funktsioon.

Tavaliselt Locusti testi jookustades valitakse loodud stsenaariumitest juhuslikult üks, mida sünteetiline kasutaja parasjagu läbiviima hakkab [70]. Kuid kuna siinses töös viiakse testid läbi väikse hulga kasutajatega, võib tekkida olukordi, kus jooksutatakse näiteks ühes testis enamus ajast pildi üleslaadimise stsenaariumit ja teises testis piltide kuvamise stsenaariumit. Sellisel juhul võivad tulemused, sõltumata taristust, erineda suurel määral, sest piltide üleslaadimine on ressursimahukam protsess piltide kuvamisest. Seega on stsenaariumite arv koodis ettemääratud. Viie sünteetilise kasutajaga testis tegeleb üks kasutaja pildi üleslaadimisega, kaks kasutajat piltide kuvamisega ning ülejäänud kaks pildi allalaadimisega. Kümne kasutaja puhul tegeleb kaks kasutajat pildi üleslaadimisega, neli piltide kuvamisega ning ülejäänud neli pildi allalaadimisega.

## 7. Tulemused

Selles peatükis kirjeldatakse Raspberry Pi ja Azure'i virtuaalmasina ülesseadmise protsessi aega ja keerukust, tuuakse välja nende ühe aasta maksumus ning antakse ülevaade nende jõudlustestide tulemustest. Lõpus võrreldakse mõlemate taristute tulemusi ja otsustatakse, milline neist on antud olukorras parem.

### 7.1 Ülesseadmine

Järgnevalt antakse ülevaade mõlema taristu ülesseadmise protsessist ning tuuakse välja nende keerulisemad aspektid. Seejärel hinnatakse ülesseadmist 5. peatükis "Metoodika" mainitud nelja raskusastme abil.

Kõigepealt uuritakse Raspberry Pi ülesseadmise protsessi. Selle juures oli üks keerulisim osa sobiva riistvara otsimine, mis hõlmas erinevate otsuste tegemist näiteks SSD mahu või Raspberry Pi mälumahu osas. Valikute tegemisel tuleb arvestada näiteks sellega, kui palju salvestusruumi võib tulevikus vaja minna, kui suur on selle koormus ning mis eesmärgil Raspberry Pi'd kasutatakse. Peale riistvara ostu tuli käsitsi kokku panna Raspberry Pi'le korpus koos ventilaatoriga ning lisada sinna külge muud välisseadmed nagu SSD ja interneti kaabel. Toiming oli selgesti mõistetav tänu riistvaraga kaasas olevatele juhenditele. Seejärel ühendati Micro-SD kaart vastava adapteri abil sülearvuti SD-kaardi porti ning sellele installiti operatsioonisüsteem Raspberry Pi OS [29]. Peale Micro-SD kaardi lisamist Raspberry Pi'sse sai isiklikust arvutist läbi SSH võrguprotokolli logida Raspberry Pi'sse sisse. Seejärel kopeeriti kõik andmed Micro-SD kaardilt SSD-le. Peale seda oli Raspberry Pi valmis kasutuseks ning sellele pandi jooksva vajalikud Docker'i konteinerid. Ülesseadmine vajab mõningaid teadmisi arvuti riistvara kohta ning Linuxi käsurea tundmist. Protsessile lisas keerukust juurde SSD paigaldamine nii, et sellelt toimuks alglaadimine. Selle kohta pidi lisainformatsiooni otsima. Kuna riistvara valimine vajab mõningaid eelnevaid teadmisi ning ülesseadmisel oli oluline tunda ka Linux'i käsurida, saab järeldada, et peatüki "Metoodika" raskusastmete kohaselt on Raspberry Pi ülesseadmise tase raske. Üles seadmisele kulus ajaliselt umbes üks päev. Kuna riistvara osad telliti läbi interneti, arvestatakse sisse ka ostu kohaletoimetamise aeg, mis oli neli päeva. Kokku kulus kogu protsessile viis päeva.

Järgnevalt uuritakse Azure'i virtuaalmasina ülesseadmise keerukust ja aega. Sarnaselt Raspberry Pi'le oli ka virtuaalmasina ülesseadmisel üheks raskemaks osaks sobivate ressursside otsimine ja selle käigus valikute tegemine. Lisaks tavapärastele riistvaralistele ressurssidele nagu mälumaht või protsessori võimsus, pakub Azure palju teisi erinevaid

valikuid ning võimalusi virtuaalmasina loomisel. Nendeks on näiteks turvalisuse taseme valik, koormusjaoturi kasutuse võimalused või võrguga seotud konfiguratsioonid. Nende detailide mõistmine, et nõuetele vastav virtuaalmasin luua, vajab laialdasi teadmisi arvuti riistvara ja muu sellega seonduva kohta. Vastasel juhul tuleb jääda vaikeväärtuste juurde, kuid see võib kaasa tuua soovitud funktsioonide puudumise või hoopis lisakulusid soovimatute funktsioonide olemasolu korral. Kui valikud parameetrite osas on tehtud, saab virtuaalmasina luua mõne klikiga. Selleks, et koduvõrgust saaks kasutada Photoprismi rakendust ning teostada seiret, peab teenuses Network Security Group avama ka vastava pordid. Photoprismile pääseb ligi pordi 2342 kaudu ning Grafana on pordil 3000. Seejärel saab isiklikust arvutist SSH võrguprotokollil abil luua Azure'i virtuaalmasinaga ühenduse. Teine kõige keerulisem osa oli seotud Azure'i Blob Storage andmesalvestuse teenuse ja virtuaalmasina vahelise ühenduse loomisega, et tagada piltide Blob Storage salvestisse liigutamine. Selleks installiti virtuaalmasinasse BlobFuse2 ning selle abil sai Blob Storage salvesti konteineri mauntida virtuaalmasina külge. Järgmisena installiti Docker ning jooksutati vajalikud konteinerid. Üldiselt vajas virtuaalmasina ülesseadmine võrreldes Raspberry Pi'ga rohkem eeltööd ja informatsiooni kogumist, et soovitud teenused tööle saada. Seetõttu on ka protsess keerukam, kuna see nõuab laialdasemaid teadmisi nii Azure'i enda teenuste kui ka arvuti arhitektuuri ja riistvara kohta. Järelikult on Azure'i virtuaalmasina ülesseadmise keerukuse tase peatüki "Metoodika" alusel väga raske. Kui eeltöö välja arvata, käib virtuaalmasina ülesseadmine koos vastavate lisateenustega kiiresti ning selle saab üles seada paari tunniga ehk ülesseadmisele kulub vähem kui üks päev.

Kokkuvõttes on Raspberry Pi ülesseadmise protsess kergem kui Azure'i virtuaalmasinas. Virtuaalmasina seadistamise teeb keeruliseks suur hulk parameetreid, mida selle loomisel kaaluma peab. Kui kasutada ka eraldiseisvat andmesalvestus teenust tuleb tegeleda selle ühendamisega, mis lisab veel keerukust. Raspberry Pi's tuleb tegeleda klassikaliste tegevustega nagu riistvara osade kokku panemise ja SSD paigaldamisega. Seetõttu on Raspberry Pi ülesseadmine tunduvalt ajamahukam võrreldes virtuaalmasinaga.

## **7.2 Kulu**

Selles alampeatükis uuritakse, kui palju lähevad maksma mõlemad taristud aastas. Kõigepealt uuritakse Raspberry Pi kulu. Kõik Raspberry Pi komponendid ning nende hinnad on toodud tabelis 2.

<b>Komponendi nimi</b>	<b>Maksumus (€)</b>
Raspberry Pi 5	68.54
SSD	61.5
Toiteadapter	13.75
Korpus koos jahutiga	12.3
Micro-SD kaart	6.14
<b>Kokku</b>	<b>162.23</b>

Tabel 2. Raspberry Pi komponendid ja nende hinnad.

Tabelis 2 on toodud ka kõikide komponentide kogusumma, mis on 162.23 eurot. Nendele lisaks arvutatakse ka tarbitud elektri hind. Kui Raspberry Pi 5'e koormus on suur võib see tarbida kohati kuni 12 vatti elektrit [71]. Arvestades, et Eesti konkurentsiameti andmetel oli 2024. aasta keskmine elektri hind 8.73 s/kWh [72] ning Raspberry Pi töötab täiskoormusel 24 tundi ja 365 päeva, tuleb elektri eest tasuda umbes 9.18 eurot aastas. Arvutatud elektrikulu on tõenäoliselt maksimum, kuna Raspberry Pi ei pruugi kogu aeg täiskoormusega töötada. Eeldades, et Raspberry Pi eluiga võib Fleetstacki artikli andmetel olla kuni 5 aastat [73], tuleb Raspberry Pi kuluks ligikaudu 41.63 eurot aastas. Kui aga Raspberry Pi väljavahetatakse, ei pea suure tõenäosusega uut korpust ega vooladapterit soetama, kuna nende eluaeg ei ole nii lühike.

Järgnevalt uuritakse töös kasutatud pilvteenuste aastast kulu. Selle teada saamiseks saab kasutada Azure'i portaalis olevaid tööriistu, mis annavad ülevaate kasutatud teenustest ning nende kuludest ühes kuus. Vaatluse alla võetakse kuu, mille igal päeval virtuaalmasin töötas ning kümnel päeval jookсутati üks või rohkem kordi koormusteste. Selle kuu kuluanalüüsi andmed on toodud tabelis 3.

<b>Teenuse nimi</b>	<b>Maksumus (€)</b>
Virtual Machines Bpsv2 Series	15.55
Premium SSD Managed Disks	9.78
Blob Storage	1.12
IP Addresses	6.67

<b>Kuus kokku</b>	<b>33.12</b>
-------------------	--------------

Tabel 3. Azure'is kasutatud pilvteenused ja nende maksumused ühes kuus.

Tabelist 3 on näha, et ühes kuus maksavad kõik teenused kokku 33.12 eurot. Eeldades, et majutatud rakendust kasutatakse terve aasta umbes kümme korda kuus, kulub Azure'i pilvteenuste kasutamisele ühes aastas kokku 397.44 eurot. Tegelik kulu võib aga arvatust erineda, kuna pilvteenuste puhul maksab kasutaja täpselt nii paljude ressursside eest, kui ta ära kasutab. Samuti sõltub hind sellest, kui suurel hulgal andmeid talletatakse. Azure Blob Storage'i teenuses maksab andmete talletamine 0.0194 eurot ühe gibibaidi kohta ning andmete manipuleerimine, näiteks failide lisamine, toob sisse veel lisakulusid [74].

Kokkuvõttes tuleb Raspberry Pi kasutamine odavam võrreldes pilvteenustega. Raspberry Pi kulu on 41.63 eurot aastas ning pilvteenuste kulu võib olla umbes 397.44 eurot aastas. Azure'i kulud sõltuvad sellest, kui palju teenuseid kasutatakse, lisades juurde keerukust ja ebamäärasust maksumuse arvutamisel. Kuna pilvteenuste kulu on umbes üheksa korda kallim, on oluline mõõta, kas ja kui palju on pilves olev virtuaalmasin võimsam Raspberry Pi'st.

### **7.3 Koormustestide tulemused**

Selles peatükis uuritakse kahe erineva koormusega testi jooksutamisel saadud tulemusi. Selleks analüüsitakse iga testi lõpus Locusti poolt genereeritud aruannet. Kõigi testide aruandeid on võimalik näha Githubi repositooriumis, mis asub lisas 1. Igat erinevat testi jooksutati vähemalt kolm korda, et tuvastada erindeid ning võtta vaatluse alla ainult selle testi tulemused, mis on sarnased eelnevatega.

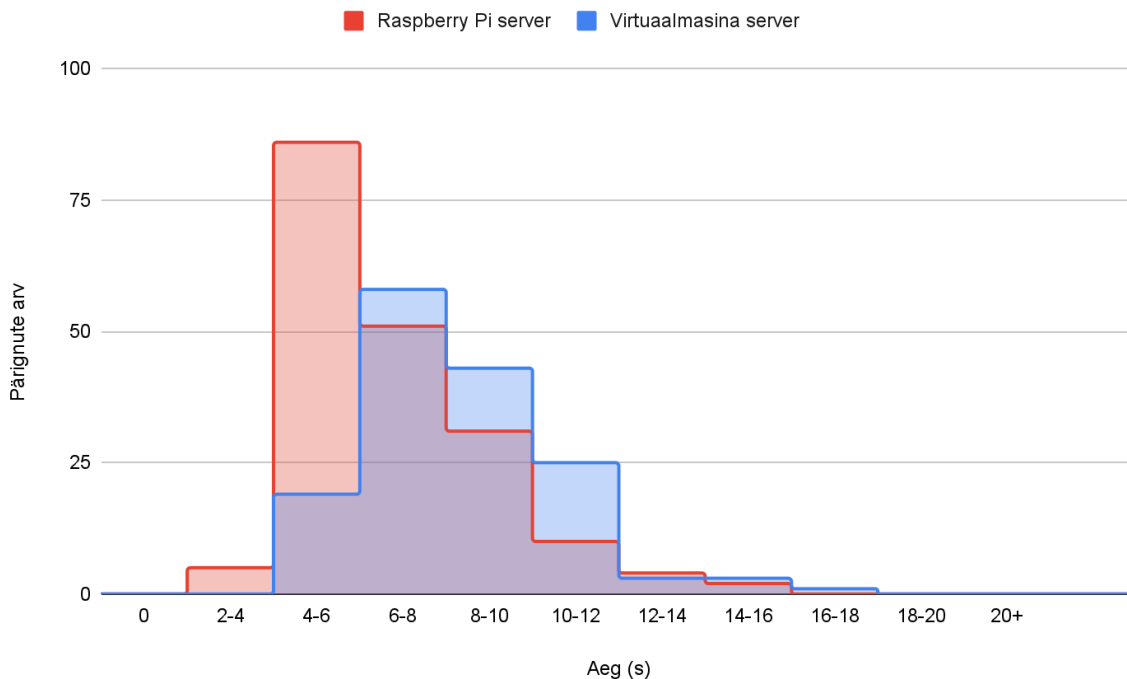
#### **7.3.1 Viie kasutajaga koormustest**

Selles alampeatükis võrreldakse Raspberry Pi ja virtuaalmasina serveri tulemusi viie sünteetilise kasutajaga testi korral. Kõigepealt vaadeldakse üldiseid tulemusi ning peale seda uuritakse iga stsenaariumi (nagu pildi üles- ja allalaadimine) raames, kui kaua läks aega päringute täitmiseks ehk kui pikad olid reaktsiooniajad. Päringu täitmise aja all peetakse silmas aega, mis kulub algse päringu saatmisest kuni serverilt vastuse saamiseni [75]. Peale seda võrreldakse mõlema taristu tulemusi omavahel. Locusti testide aruannetest selgub, et mõlema taristu läbilaskevõimed ühes sekundis olid sarnased. Raspberry Pi serveri läbilaskevõime oli 3.03 päringut sekundis ning virtuaalmasina serveris 2.97 päringut sekundis. Mõlema taristu keskmised päringu täitmiseks kulunud ajad olid 0.56 sekundit.

Järgnevalt uuritakse iga stsenaariumi kohta eraldi, kui kaua läks keskmiselt nende päringute täitmiseks aega. Kõigepealt vaadeldakse kõikide piltide laadimise stsenaariumit. Selleks toetutakse Locusti aruandele, kus on näha, et Raspberry Pi serveril kulus kõikide piltide laadimiseks ligikaudu 0.69 sekundit ja virtuaalmasinal 0.74 sekundit. Päringu täitmise ajad on mõlemal taristul sarnased ning need täitsid kasutusloos seatud eesmärgi, mis väitis, et piltide laadimine peaks toimuma vähem kui kahe sekundiga.

Järgmisena vaadeldakse pildi allalaadimiseks kulunud aegu. Raspberry Pi serveril kulus pildi allalaadimise päringu täitmiseks keskmiselt ligikaudu 4.7 sekundit ning virtuaalmasinal 4 sekundit. Järelikult on virtuaalmasina tulemused siin stsenaariumis kiiremad. Mõlemate taristute serverid täitsid kasutusloos seatud kriteeriumi, mis väitis, et piltide allalaadimine peaks toimuma viie sekundi vältel.

Viimaseks vaadeldakse lähemalt, kui pikad olid pildi üleslaadimiseks sooritatud päringute ajad. Selleks kasutatakse teistsugust lähenemist võrreldes eelmiste stsenaariumitega. Locusti aruandes on toodud kõik päringud välja eraldi, kuid kuna Photoprism teeb pildi üleslaadimisel kaks päringut POST ja PUT meetodiga, siis peab kogu ühe pildi üleslaadimisele kulunud aja leidmiseks arvutama kokku nii POST kui ka PUT meetodi täitmise aja. Järgnevalt uuritaksegi ühe pildi üleslaadimisele kuluvat terviklikku aega, mis arvestab nii pildi algset üleslaadimist, mis teostatakse POST-päringuga kui ka sellejärgset indekseerimist ja importimist, mis toimub PUT-päringu vältel. Selleks arvutati iga üleslaetud pildi POST- ja PUT-päringute reaktsiooniaegade summa. Testi tulemuste põhjal koostati mõlema serveri kohta histogrammid, mis näitavad, milliseid reaktsiooniaegu esineb kõige rohkem. Mõlema taristu histogrammid on toodud joonisel 5.



Joonis 5. Pildi üleslaadimise aegade histogrammid Raspberry Pi ja virtuaalmasina serveri kohta viie kasutaja korral.

Joonisel 5 on horisontaalteljel toodud reaktsiooniaegade vahemikud ning vertikaalteljel, mitu päringut vastavasse ajavahemikku jäi. Punased tulbad viitavad Raspberry Pi serveri ning sinised virtuaalmasina reaktsiooniaegadele. Samuti on näha nende vahelist kattuvust. Jooniselt 5 selgub, et Raspberry Pi serveril laeti enamuse pilte üles 4-6 sekundiga. Seevastu on näha, et virtuaalmasina serveris esines kõige rohkem juhte, kus pildi üleslaadimisele kulus 6-8 sekundit. Raspberry Pi serveris täideti peaaegu ligikaudu 92% sooritatud päringutest kasutusloos seatud ajalise eesmärgi ehk kümne sekundi raames ära. Seevastu jäi virtuaalmasinas alla kümne sekundi ligikaudu 79% pildi üleslaadimise päringutest. Keskmine pildi üleslaadimise aeg on Raspberry Pi serveril ligikaudu 5.7 sekundit. Erinevalt Raspberry Pi'ist on virtuaalmasina aeg ligikaudu 7.3 sekundit. Saadud tulemustest saab järeldada, et Raspberry Pi serveril kulus piltide üleslaadimiseks vähem aega. Sellegipoolest täitsid mõlemad serverid kasutusloos seatud eesmärgi enamuse ajast ära. Tabelis 4 on toodud kõikide eelmainitud stsenaariumite päringute täitmise ajad.

	<b>Raspberry Pi server</b>	<b>Virtuaalmasina server</b>
Keskmine päringu täitmise aeg (s)	0.56	0.56
Kõikide piltide laadimiseks kulunud keskmine aeg (s)	0.69	0.74
Pildi allalaadimiseks kulunud keskmine aeg (s)	4.7	4.0
Pildi üleslaadimiseks kulunud keskmine aeg (s)	5.7	7.3

Tabel 4. Päringute täitmiseks kulunud ajad Raspberry Pi ja virtuaalmasina serveritel.

Kokkuvõttes said nii Raspberry Pi kui ka virtuaalmasina server viie sünteetilise kasutajaga koormustestidega hakkama. Mõlemad taristud andsid sarnaseid tulemusi kõikide piltide kättesaamisel ja piltide allalaadimisel. Piltide üleslaadimise toimus kiiremini Raspberry Pi serveril. Sellegipoolest täitsid mõlemad taristud ära kõik kasutusloos seatud kriteeriumid.

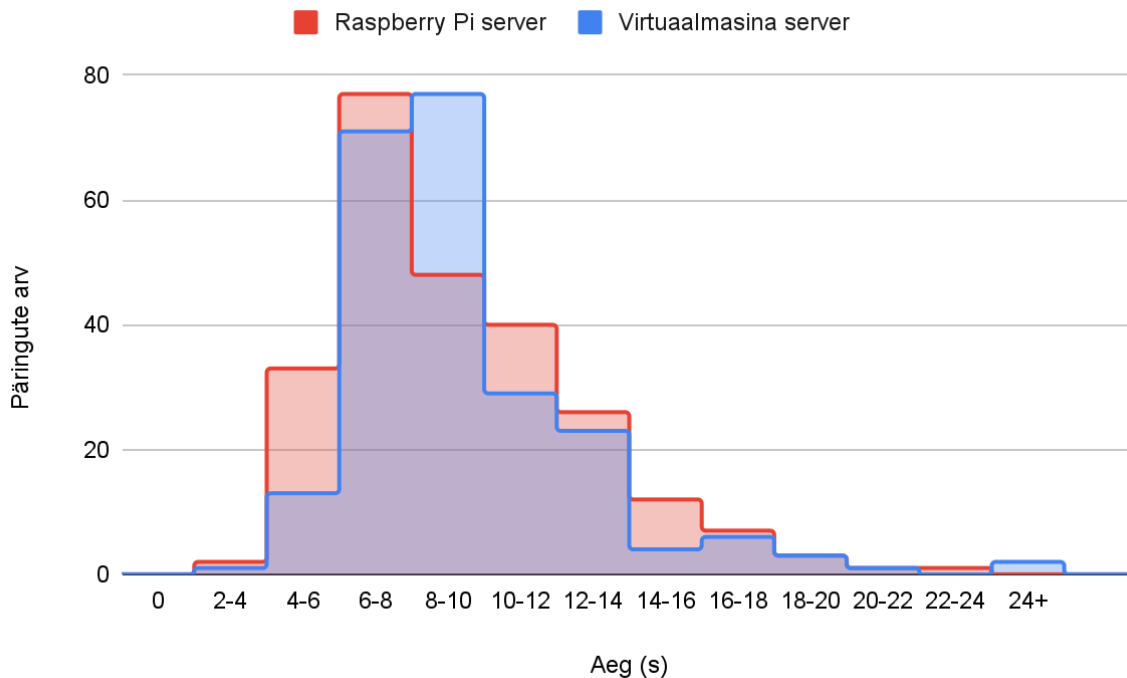
### 7.3.2 Kümne kasutajaga koormustest

Siin alampeatükis vaadeldakse kümne sünteetilise kasutajaga läbi viidud koormustesti tulemusi. Locusti aruandest selgub, et Raspberry Pi serveri läbilaskevõime oli 3.49 päringut sekundis ning virtuaalmasinal 2.95 päringut sekundis. Siin on näha, et Raspberry Pi täitis ühes sekundis rohkem päringuid võrreldes virtuaalmasinaga. Keskmine reaktsiooniaeg Raspberry Pi serveril oli 1.2 sekundit ning virtuaalmasinal 1.6 sekundit. Seega olid Raspberry Pi serveri ajad üldiselt natuke kiiremad. Järgnevalt uuritakse iga stsenaariumi kohta eraldi, kui kaua kulus nende päringute täitmiseks aega.

Piltide laadimise päringute keskmine reaktsiooniaeg Raspberry Pi serveril oli ligikaudu 1.5 sekundit ning virtuaalmasinal 2.1 sekundit. Raspberry Pi ajad on kiiremad ning kasutusloos seatud kriteerium, mis väitis, et piltide laadimine peaks toimuma vähem kui kahe sekundiga, sai täidetud. Virtuaalmasin eesmärki ei täitnud.

Piltide allalaadimiseks kulunud ajad olid mõlemal serveril üpris sarnased. Raspberry Pi'l kulus pildi allalaadimise päringu täitmiseks keskmiselt ligikaudu 15 sekundit ning virtuaalmasinal 15.2 sekundit. Ajad olid märkimisväärselt pikemad kasutusloos seatud kriteeriumist, mis seadis pildi allalaadimise piiriks viis sekundit. Seega ei täitnud kumbki taristutest selle stsenaariumi eesmärki.

Viimasena vaadeldakse lähemalt piltide üleslaadimiseks kulunud aegu. Selleks esitatakse joonisel 6 nii Raspberry Pi kui ka virtuaalmasina serverite tulemuste põhjal loodud histogrammid, mis näitavad kui sagedasti esines mingeid reaktsiooniaegade vahemikke.



Joonis 6. Pildi üleslaadimisele kulunud aegade histogrammid Raspberry Pi ja virtuaalmasina serverite kohta kümne kasutaja korral.

Jooniselt 6 on näha, et kümne kasutaja korral on pildi üleslaadimise ajad pikemad võrreldes eelmise testi tulemustega. Raspberry Pi serveril kulus piltide üleslaadimiseks peamiselt 6-8 sekundit. Virtuaalmasina serveris esines kõige rohkem juhte, kus pildi üleslaadimiseks kulus 8-10 sekundit. Vaadeldes keskmised reaktsiooniaegu, tuleb välja, et Raspberry Pi'l toimus pildi üleslaadimise päringu täitmine keskmiselt ligikaudu 8.3 sekundi ning virtuaalmasinal 8.4 sekundiga. Lähtudes kasutusloo eesmärgist, et pildi üleslaadimisele peaks kuluma vähem kui 10 sekundit, täitis Raspberry Pi server seatud eesmärgi 64%-il juhtudest ning virtuaalmasina server ligikaudu 70%-il juhtudest. Järelikult täitsid mõlemad taristud enamikel juhtudel seatud eesmärgi. Tabelis 5 on näha kõikide eelmainitud stsenaariumite päringute täitmise ajad kümne kasutaja korral.

	<b>Raspberry Pi server</b>	<b>Virtuaalmasina server</b>
Keskmine päringu täitmise aeg (s)	1.2	1.6
Kõikide piltide laadimiseks kulunud keskmine aeg (s)	1.5	2.1
Pildi allalaadimiseks kulunud keskmine aeg (s)	15	15.2
Pildi üleslaadimiseks kulunud keskmine aeg (s)	8.3	8.4

Tabel 5. Päringute täitmiseks kulunud ajad Raspberry Pi ja virtuaalmasina serveritel kümne kasutajaga koormustesti korral.

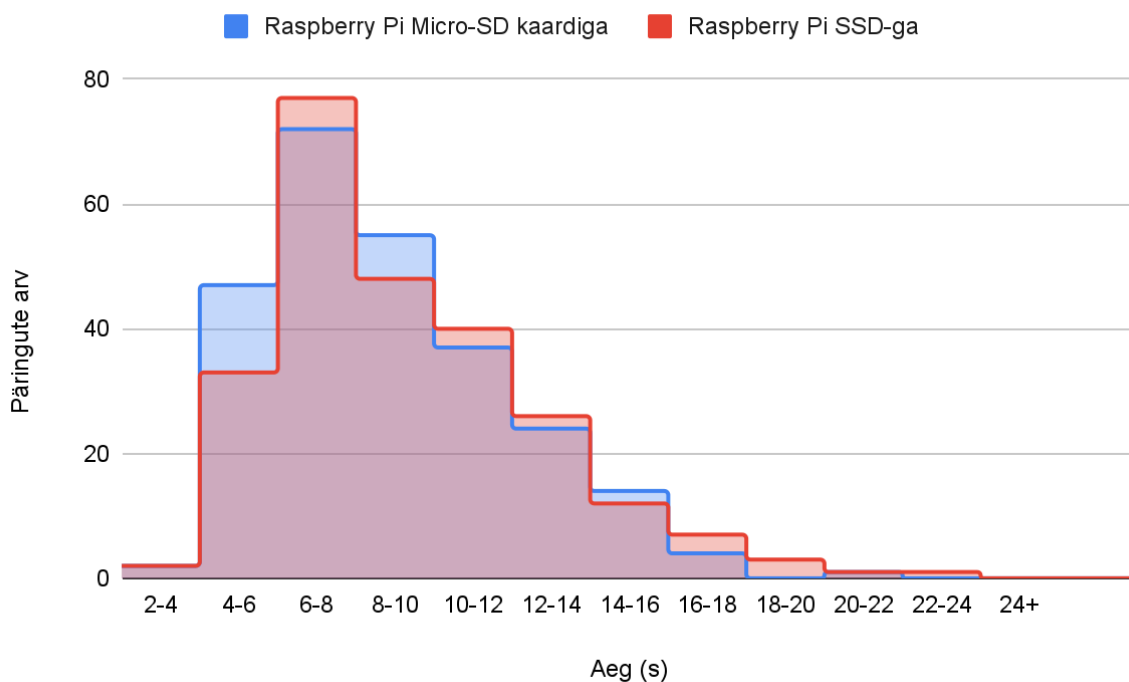
Kokkuvõttes on kümne kasutajaga testi tulemused märgatavalt aeglasemad viie kasutajaga testi tulemustest. Raspberry Pi server täitis piltide üles- ja allalaadimise päringuid võrreldes virtuaalmasinaga kiiremini, kuid nende aegade vahe ei olnud suur. Mõlemad täitsid pildi üleslaadimise ajalise kriteeriumi, kuid kumbki ei saavutanud pildi allalaadimise ajalist eesmärki. Raspberry Pi näitas kiiremaid tulemusi kõikide piltide kuvamisel ning täitis ka seatud kriteeriumi. Virtuaalmasin seda eesmärki ei täitnud.

### 7.3.3 Micro-SD kaardiga läbiviidud koormustest

Järgnevalt uuritakse, kuidas toimib rakenduse töö kasutades SSD asemel Micro-SD kaarti. Nii proovitakse välja selgitada, kas SSD soetamine on vajalik. Selleks viiakse kümne kasutajaga test läbi ka muudetud konfiguratsiooniga. Seejärel võrreldakse saadud tulemusi varasemate tulemustega, mis eelmises, kümne kasutajaga testis läbi viidi. Nendes oli Raspberry Pi varustatud SSD-ga.

Locusti aruandest selgub, et Micro-SD kaardiga varustatud Raspberry Pi täitis ühe sekundiga 3.53 päringut, mis on väga lähedal SSD-ga saadud läbilaskevõimele, mis oli 3.49 päringut sekundis. Micro-SD kaarti kasutades oli keskmine päringu täitmise aeg 1.3 sekundit ning SSD-ga 1.2 sekundit. Seega olid mõlemate ajad üpris sarnased. Locusti aruandest selgub, et kõikide piltide laadimiseks kulus Micro-SD kaarti kasutades keskmiselt ligikaudu 1.6 sekundit ning SSD-d kasutades 1.5 sekundit. Järelikult on SSD-ga saadud tulemused natukene kiiremad. Suuremaid erinevusi on näha aga pildi allalaadimiste stsenaariumis. Micro-SD kaardiga oli pildi allalaadimise keskmine päringu täitmise aeg ligikaudu 13.2 sekundit. See on kiirem SSD-ga saadud tulemusest, mis oli 15 sekundit.

Järgnevalt vaadeldakse lähemalt, kui kaua läks Micro-SD kaarti kasutades aega piltide üleslaadimiseks ning võrreldakse neid eelnevas testis saadud tulemustega, kus kasutati SSD-d. Selleks koostati histogrammid, mis näitavad kui sagedasti esines mingeid reaktsiooniaegu Micro-SD kaarti või SSD-d kasutades. Mõlema konfiguratsiooni histogrammid on toodud joonisel 7. Sinised tulbad viitavad Micro-SD kaardi ja punased SSD-ga saadud aegade esinemiste arvule.



Joonis 7. Pildi üleslaadimiseks kulunud aegade histogrammid Micro-SD kaardi ja SSD kasutuse korral.

Jooniselt 7 selgub, et mõlema konfiguratsiooniga sooritatud tulemustes on palju kattuvust. Mõlemal juhul täideti kõige rohkem päringuid 6-8 sekundi jooksul. Keskmine reaktsiooniaeg Micro-SD kaardiga oli ligikaudu 7.8 sekundit ning SSD-d kasutades 8.3 sekundit. Jooniselt 7 on näha, et Micro-SD kaardiga esines rohkem päringuid, mis võtsid aega 4-6 sekundit. Järelikult toimus piltide üleslaadimine Micro-SD kaarti kasutades kiiremini võrreldes SSD-ga. Tabelis 6 on kokku võetud kõikide stsenaariumite keskmised päringu täitmise ajad mõlema konfiguratsiooniga.

	<b>Raspberry Pi Micro-SD kaardiga</b>	<b>Raspberry Pi SSD-ga</b>
Keskmine päringu täitmise aeg (s)	1.3	1.2
Kõikide piltide laadimiseks kulunud keskmine aeg (s)	1.6	1.5
Pildi allalaadimiseks kulunud keskmine aeg (s)	13.2	15
Pildi üleslaadimiseks kulunud keskmine aeg (s)	7.8	8.3

Tabel 6. Päringu täitmiseks kulunud ajad Micro-SD kaardi ja SSD-ga sooritatud kümne kasutajaga testis.

Kokkuvõttes olid Micro-SD kaardiga sooritatud tulemused kümne kasutajaga testis sarnased SSD-ga saadud tulemustele. Kõikide piltide laadimise ajad olid mõlemal sarnased, kuid piltide üles- ja allalaadimisel olid reaktsiooniajad natuke kiiremad Micro-SD kaarti kasutades.

#### **7.3.4 Koormustestide tulemuste analüüs**

Selles alampeatükis analüüsitakse koormustestide tulemusi. Kõigepealt arutletakse Raspberry Pi'l ja virtuaalmasinal läbiviidud testide tulemuste üle. Seejärel kaalutletakse Micro-SD kaardiga sooritatud tulemusi ja arutletakse selle kasutuse üle.

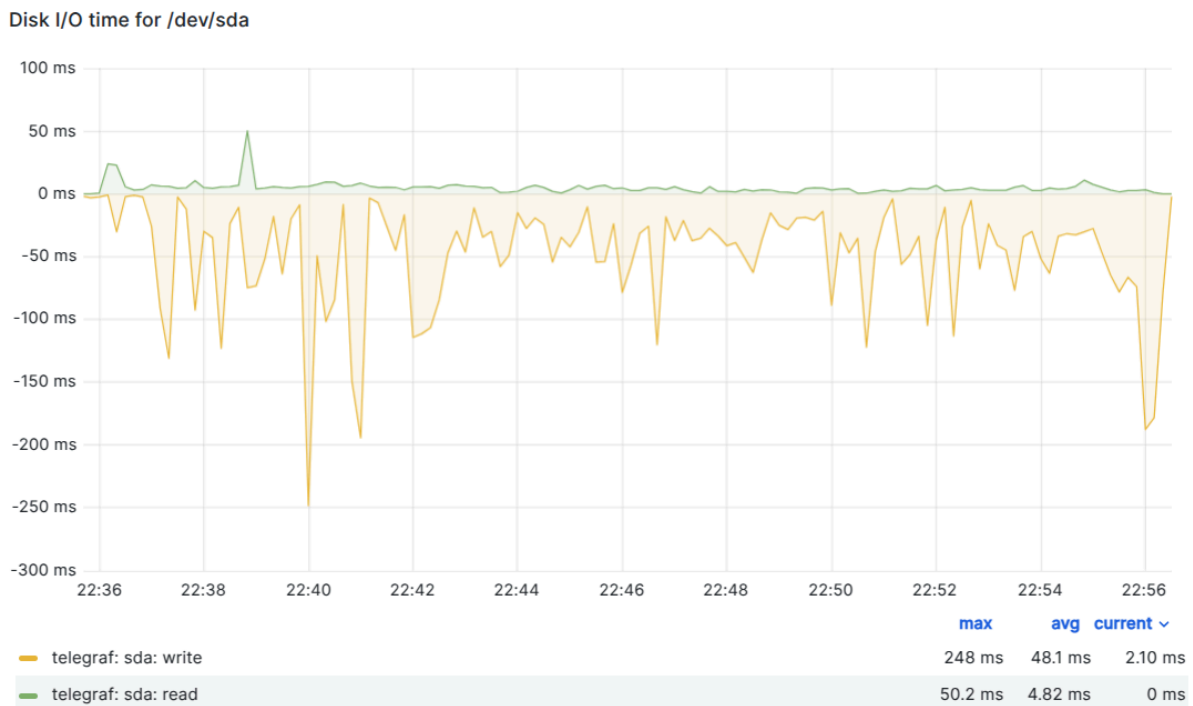
##### **Raspberry Pi ja virtuaalmasin**

Viie sünteetilise kasutajaga läbiviidud testidega said mõlemad taristud hakkama ning päringu täitmise ajad olid üldiselt üpris sarnased. Kümne kasutajaga testi reaktsiooniajad olid eelmisest testist märgatavalt aeglasemad, kuid mõlemate taristute tulemused olid sellegipoolest sarnased. Järgnevalt analüüsitakse kümne kasutajaga testi tulemusi.

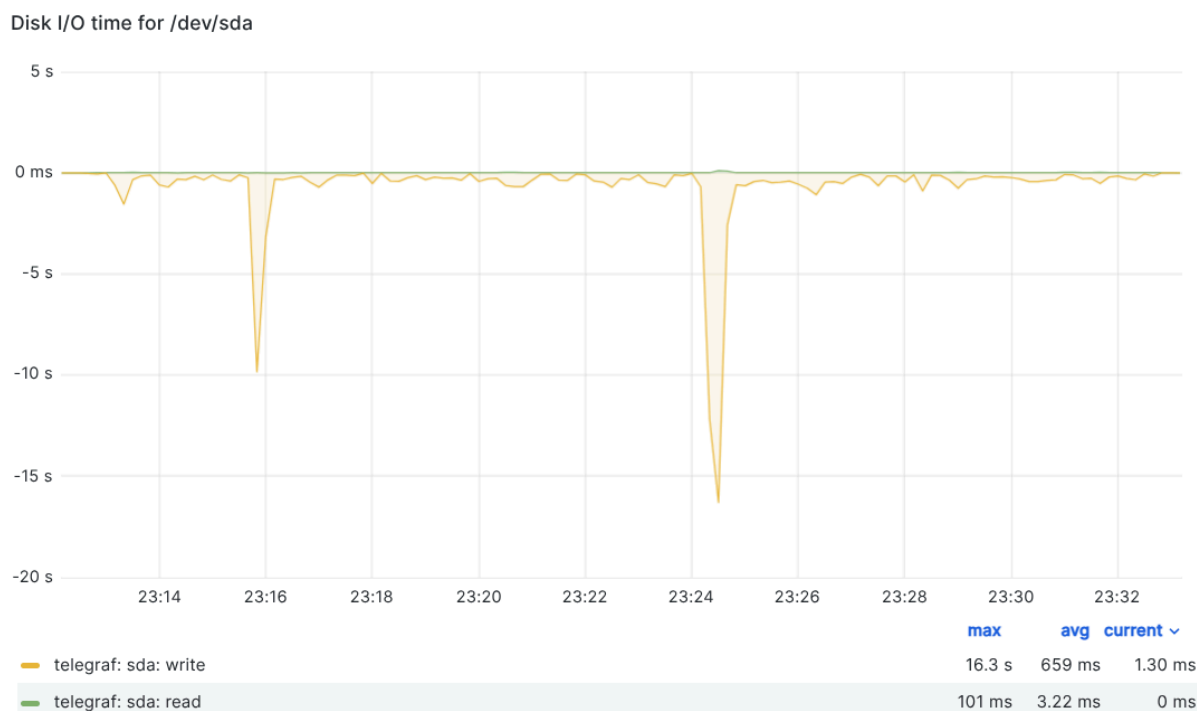
Kümne kasutajaga testis viitas probleemidele mõlema serveri läbilaskevõime. Kui viie kasutajaga testis oli Raspberry Pi läbilaskevõime 3.03 päringut sekundis, siis selles testis täideti 3.49 päringut sekundis. Sarnane muster ilmnis ka virtuaalmasina puhul, mille läbilaskevõime viie kasutajaga testis oli 2.97 päringut sekundis ning kümne kasutajaga testis oli 2.95 päringut sekundis. Kuna sünteetiliste kasutajate arv oli kaks korda suurem eelmise testi kasutajate arvust, oleks pidanud ka läbilaskevõime olema kaks korda suurem. Raspberry Pi server täitis võrreldes eelmise testiga küll natuke rohkem päringuid sekundis, kuid virtuaalmasina läbilaskevõime ei suurenenud üldse. See näitab, et mingi tegur pärsib serverite

tööd, tekitades probleeme päringute täitmisega. Järgnevalt uuritakse lähemalt selle testi seire tulemusi ning analüüsitakse, mis võis testide tulemusi mõjutada.

Seirest selgus, et Raspberry Pi protsessori kasutus testi vältel oli keskmiselt 24% ja virtuaalmasinal 33%. Seega kasutasid taristud oma protsessoreid mõõdukalt ning ressursse jäi ülegi. Muutmälu kasutus Raspberry Pi'l oli keskmiselt 53% ja virtuaalmasinal 60%. Ka see näitaja on mõlemal taristul normaalsel tasemel. Kõige rohkem on taristute vahelisi erinevusi märgata ketta kirjutamisoperatsioonide aegades. Joonisel 8 on toodud Raspberry Pi ketta operatsioonide ajad ning joonisel 9 virtuaalmasina ajad.



Joonis 8. Raspberry Pi'l ketta operatsioonidele kulunud ajad.



Joonis 9. Virtuaalmasinal ketta operatsioonidele kulunud ajad.

Jooniselt 8 ja 9 on näha, et Raspberry Pi keskmine kettale kirjutamise aeg on 48.1 millisekundit ning virtuaalmasinal 659 millisekundit. Vastavalt joonisele 9 selgub, et kahel juhul on kettale kirjutamine võtnud oluliselt rohkem aega, kui tavaliselt. Kõige pikem kirjutamise aeg kestis 16.3 sekundit. See näitab, et kirjutamisoperatsioonid olid virtuaalmasinal aeglasemad ja ebastabiilsed. Tuleb tähele panna, et Joonisel 9 pole arvestatud andmete Azure Blob Storage'i konteinerisse salvestamist, vaid andmete ajutiselt kettale kirjutamist. Azure'i portaalis vastava Storage Account teenuse seiramisel selgub, et failide ühekaupa kettalt Blob Storage'i konteinerisse salvestamine võtab keskmiselt ligikaudu 5 millisekundit. Üldiselt saab sellest järeldada, et üheks kitsaskohaks virtuaalmasinas on ketta kirjutamisoperatsioonide suur ajakulu, kuid sellegipoolest ei selgita eelnevalt mainitud seire tulemused, miks oli mõlema taristu läbilaskevõime oodatust nõrgem.

Testide tulemusi, sh läbilaskevõimet võis mõjutada ka interneti kiirus ning see, et Raspberry Pi oli testiva arvutiga samas kohtvõrgus, kuid virtuaalmasin mitte. Kui rakenduse kasutaja seade on samas kohtvõrgus Raspberry Pi'ga, tekib väiksem latentsus, kuna seadmed on üksteisele lähedal. Cloudflare'i dokumentatsiooni põhjal võivad tänu sellele olla päringute täitmise ajad kiiremad [76]. Azure'i andmekeskused paiknevad väljaspool Eestit [77], põhjustades seega suuremat latentsust ning aeglasemaid päringu täitmise aegu. Sellegipoolest

on nende eeliseks see, et majutatud rakendus on kättesaadav igalt poolt, kus internetti on. Raspberry Pi'le saab antud konfiguratsiooniga ligi ainult kohtvõrgust.

Kokkuvõttes näitas viie kasutajaga läbiviidud test, et mõlemate taristute serverid saavad ootuspäraselt hakkama. Kümne kasutajaga testis sooritatud päringu täitmise ajad olid üldiselt aeglasemad, kuid sellegipoolest näitasid mõlemad taristud sarnaseid tulemusi. Seire tulemuste põhjal ei suudetud välja selgitada, mis pärssis taristute läbilaskmisvõimet ja päringute täitmise kiirust. Üldiselt saab testi tulemustest järeldada, et mõlema taristu jõudlused on sarnased.

### **Micro-SD kaart ja SSD**

Erineva konfiguratsiooniga testi Raspberry Pi peal läbiviies selgus, et Micro-SD kaardiga on päringu täitmise ajad üpris sarnased algse konfiguratsiooni ehk SSD-ga saavutatud aegadega. Sellegipoolest on SSD-l mitmeid eeliseid võrreldes Micro-SD kaardiga, mis teevad sellest töökindlama valiku. Allika kohaselt on üldiselt Micro-SD kaartide salvestusmaht limiteeritud ja väiksem kui SSD-l [25]. Ka siin töös on salvestusmahtude vahe üpris suur – Micro-SD kaardi maht on 32 gigabaiti ja SSD-l 500 gigabaiti. Seega saab SSD-d kasutades rohkem faile talletada. Kuigi nende mõlema eluiga on umbes 10 aastat [78,79], võib Micro-SD kaart kiiresti kuluda ning seejärel korrumpeeruda, kui seda kasutatakse operatsioonisüsteemi jooksutamiseks, mis nõuab palju kirjutamisoperatsioone [80]. Ka SSD eluiga on sõltuv kirjutamisoperatsioonide arvust, kuid tänu suuremale salvestusmahule, suudab see taluda ka rohkem kirjutamisoperatsioone võrreldes väiksemahulise Micro-SD kaardiga [81]. Samuti on allika kohaselt SSD-d keskmiselt kiiremad andmete lugemisel ja kirjutamisel, kui Micro-SD kaardid. Kuna SSD-d on üldiselt mahult suuremad ning võimsamad, on nad seetõttu ka kulukamad [25]. Sellegipoolest soovitab autor kasutada SSD-d Micro-SD kaardi asemel.

## **7.4 Tulemuste kokkuvõte**

Selles peatükis võetakse kokku kõigi kolme uuritud tegurite tulemused ning proovitakse nende põhjal leida vastus küsimusele: Kumb on parem väikese kasutajate arvuga ning suurte andmemahutudega töötava rakenduse majutamiseks, kas Raspberry Pi 5 või odav Azure'i virtuaalmasin? Töös uuriti mõlema taristu ülesseadmise keerukust ning aega, selle rahalist kulu ning jõudlust. Jõudluse uurimiseks läbiviidud koormustestid näitasid, et kuni viie kasutaja teenindamisega saavad mõlemad taristud hästi hakkama. Kui kasutajate arv suureneb kümneni tekib mõlemal taristul probleeme päringute täitmisega ning nende kiirused on mõlemal aeglased. Seega on taristute jõudlused sarnased ning valiku tegemisel peab lähtuma

teistest teguritest nagu ülesseadmine ja maksumus. Kuigi mõlemate jõudlused on sarnased, on nende kulu väga erinev. Raspberry Pi aastane kulu on võrreldes virtuaalmasinaga umbes üheksa korda odavam. Siinse töö kulu tulemused sarnanesid Laumetsa tööga, kus uuriti Raspberry Pi 2'e kulu Azure'i ühetuumalise virtuaalmasinaga ning selgus, et ka sel juhul on Raspberry Pi soodsam [19]. Seega on Raspberry Pi eeliseks selle madal hind. Kui vaadelda mõlema taristu ülesseadmise keerukust, siis selgus, et Raspberry Pi ülesseadmine on üldiselt lihtsam, kuna see ei vajanud nii palju eeltööd ja Azure'i platvormipõhiseid teadmiseid kui ülesseatud virtuaalmasin. Azure'i virtuaalmasina eelis peitub ülesseadmise ajas. Virtuaalmasin seati üles ühe päevaga, kuid Raspberry Pi ülesseadmine võttis aega viis päeva. Töös saadud tulemused on kokku võetud tabelis 7.

	<b>Raspberry Pi 5</b>	<b>Azure'i virtuaalmasin Standard B2pls v2</b>
Ülesseadmise aeg	5 päeva	1 päev
Ülesseadmise keerukus	Raske	Väga raske
Kulu	41.63 eurot aastas	397.44 eurot aastas
Jõudlus (kümne kasutajaga testi keskmine päringu täitmise aeg)	1.2 s	1.6 s

Tabel 7. Raspberry Pi ja Azure'i virtuaalmasina võrdlus.

Saadud tulemustest saab järeldada, et kuigi mõlema taristu jõudlused on sarnased, on Raspberry Pi odavam ning seda on lihtsam ülesseada. Kui aga kasutaja peab rakenduse majutamisel oluliseks ülesseadmise aega, võib sel juhul sobivaks lahenduseks olla ka pilves olev virtuaalmasin. Kuid kokkuvõttes on väikese kasutajabaasiga ning suuremate failidega tegeleva rakenduse majutamiseks mõistlikum kasutada Raspberry Pi 5'te.

## 8. Kokkuvõte

Selles töös võrreldi omavahel odavat füüsilist taristut ning soodsat pilvtaristut. Töö eesmärk oli välja selgitada, millisel taristul on mõistlikum majutada rakendust, mis tegeleb suurte andmemahtudega ning mille kasutajate arv on väike. Võrdlemisel võeti arvesse nelja aspekti - taristu jõudlust, kulu ning ülesseadmise aega ja keerukust.

Kõigepealt valiti välja võrreldavad taristud. Füüsiliseks taristuks kasutati Raspberry Pi 5'te ning pilvlahendusena Azure'i virtuaalmasinat. Seejärel otsustati, millist rakendust majutatakse ning mis tööriistu kasutada selle testimiseks ning üldiseks seireks. Kui sobivad tehnoloogiad olid välja valitud seati mõlemad taristud üles ning neile pandi jooksma väljavalitud pildigalerii rakendus Photoprism. Selle käigus märgiti mõlema taristu puhul üles ülesseadmise raskemad etapid, et hinnata nende ülespanemise keerukust. Lisaks mõõdeti, kui palju kulus ülesseadmiseks aega. Kui rakendus oli kuu aega töös olnud, sai selle põhjal välja arvutada mõlema taristu aastase kulu. Selleks, et nende jõudlust võrrelda, viidi läbi koormustestid viie ning kümne kasutajaga ning nende vältel teostati ka süsteemiseiret.

Taristute jõudlust uurides selgus, et nende koormustestide tulemused olid sarnased. Mõlemad said hästi hakkama viie kasutajaga tehtud koormustestiga. Kümne kasutaja testis näitasid mõlemad aeglaseid, kuid sellegipoolest sarnaseid päringu täitmise aegu. Seire tulemusena selgus, et virtuaalmasina kettaoperatsioonid olid oluliselt aeglasemad võrreldes Raspberry Pi'ga. See võis olla üks põhjus, miks virtuaalmasina tulemused ei olnud ideaalsed. Lisaks viidi läbi koormustest ka Raspberry Pi's Micro-SD kaarti kasutades ning võrreldi tulemusi SSD-ga tehtud sooritusega. Ka sel juhul olid tulemused üpris sarnased, kuid üldiselt on SSD-l rohkem muid eeliseid näiteks suurem salvestusmaht. Tänu sellele on see eelistatum variant. Kasutuskogemuse uurimise käigus tuli välja, et Raspberry Pi aastane kulu on üheksa korda odavam virtuaalmasina ülalpidamise kuludest. Lisaks oli Raspberry Pi'd võrreldes virtuaalmasinaga lihtsam ülesseada. Sellegipoolest oli virtuaalmasina eeliseks selle ülesseadmise kiirus. Kuna Raspberry Pi on sarnase jõudlusega, kuid kordades odavam ja lihtsamini ülesseatav, on see järelikult võrreldes virtuaalmasinaga parem lahendus suurte andmemahtudega ja väikese kasutajate arvuga rakenduse majutamiseks. Sellega sai töös seatud eesmärk täidetud.

Edaspidistes töödes võiks võrrelda omavahel Raspberry Pi'd ja Azure App Service [37] teenust, mille ülesseadmine võib olla lihtsam ning odavam võrreldes siin töös kasutatud virtuaalmasinaga. Samuti võiks uurida ning võrrelda Raspberry Pi ja avalike pilvlahenduste

kõrval ka privaatpilve näiteks ETAISI [82] jõudlust ning kasutuskogemust. Tulevikus võib teste läbi viia mingi teist tüüpi rakenduse näiteks protsessori jõudlust vajava rakendusega.

## Viidatud kirjandus

- [1] DataReportal. Digital Around the World. <https://datareportal.com/global-digital-overview> (09.03.2025)
- [2] Statistikaamet. Eestimaalased kasutavad internetti järjest rohkem. <https://stat.ee/et/uudised/infotehnoloogia-leibkondades-2023> (09.03.2025)
- [3] Jordan S. The Pros and Cons of Cloud vs. In-House Servers. [https://sysgen.ca/cloud-vs-in-house-servers/#cloud\\_vs\\_in-house\\_which\\_is\\_right\\_for\\_me](https://sysgen.ca/cloud-vs-in-house-servers/#cloud_vs_in-house_which_is_right_for_me) (09.03.2025)
- [4] Armstrong M. What's in the Cloud? 2021. <https://www.statista.com/chart/25896/gcs-cloud-storage-services-usage/> (07.05.2025)
- [5] Andmekaitse ja infoturbe leksikon. Cybernetica AS. <https://akit.cyber.ee/>.
- [6] Webopedia. <https://www.webopedia.com/>.
- [7] Interactive Terminology for Europe. <http://iate.europa.eu>.
- [8] Indexing Your Library. <https://docs.photoprism.app/user-guide/library/> (20.04.2025)
- [9] Ravikiran A. S. What Is Raspberry Pi? Here's The Best Guide To Get Started. <https://www.simplilearn.com/tutorials/programming-tutorial/what-is-raspberry-pi> (06.12.2024).
- [10] Basumallick C. What Is Raspberry Pi? Models, Features, and Uses. 2022. <https://www.spiceworks.com/tech/networking/articles/what-is-raspberry-pi/> (06.12.2024).
- [11] Kurkovsky S, Williams C. Raspberry Pi as a Platform for the Internet of Things Projects: Experiences and Lessons. In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. Bologna Italy: ACM, 2017. lk. 64–9. <https://dl.acm.org/doi/10.1145/3059009.3059028>
- [12] Solihin bin Ahmad Nasser M, Attarbashi ZS, Aman AHM, Abuzaraida MA. Building an Affordable Portable Storage Area Network (SAN) with Raspberry Pi: Design, Implementation, and Performance Evaluation. In: 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA). 2024. lk. 787–92. <https://ieeexplore.ieee.org/document/10599641> (14.11.2024).
- [13] What is Cloud Computing? <https://cloud.google.com/learn/what-is-cloud-computing>

(26.04.2025)

[14] Google Docs. <https://docs.google.com/> (26.04.2025)

[15] AWS Elastic Beanstalk. <https://aws.amazon.com/elasticbeanstalk/> (26.04.2025)

[16] Cloud Based Services. <https://www.geeksforgeeks.org/cloud-based-services/>  
(26.04.2025)

[17] Azure Virtual Machines. <https://azure.microsoft.com/en-us/products/virtual-machines/>  
(26.05.2025)

[18] Madders C. The Advantages and Disadvantages of Cloud Computing: Is Your Head in the Cloud? 2021.  
<https://info.cybertecsecurity.com/advantages-and-disadvantages-of-cloud-computing>  
(27.04.2025)

[19] Laumets A. Migration from virtual environment to physical infrastructure of Raspberry Pis. TÜ arvutiteaduse instituudi bakalaureusetöö. Tartu: 2016.  
[https://comserv.cs.ut.ee/ati\\_thesis/datasheet.php?id=53807](https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53807) (06.12.2024).

[20] Gamess E, Hernandez S. Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests. IJACSA. 2022; 13(2).  
<https://doi.org/10.14569/IJACSA.2022.0130295>

[21] Torm R. Cloud-based data analysis and server scalability for SportID. TÜ arvutiteaduse instituudi bakalaureusetöö. Tartu: 2014.  
[https://comserv.cs.ut.ee/ati\\_thesis/datasheet.php?id=40961](https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=40961) (07.12.2024).

[22] Fromaget P. Which Raspberry Pi Model to Buy in 2025: A Complete Guide.  
<https://raspberrypi.com/best-raspberry-pi-model/> (27.04.2025)

[23] Raspberry Pi 5. <https://www.raspberrypi.com/products/raspberry-pi-5/> (27.04.2025)

[24] Portable SSD External Solid State Drive-SD500. SSK.  
[https://eng.ssk.cn/Product\\_Center\\_details/21.html](https://eng.ssk.cn/Product_Center_details/21.html) (27.04.2025)

[25] Amy. SSD vs SD Card: Similarities and Differences Between Them.  
<https://www.partitionwizard.com/partitionmagic/ssd-vs-sd-card.html> (09.05.2025)

[26] Qoltec Super FAST PD charger for Raspberry PI 5 | USB-C | 27W | 5.1V | 5A | Black.  
<https://www.qoltec.com/product/qoltec-super-fast-pd-charger-for-raspberry-pi-5-usb-c-27w-5-12v-18-5a-black-1> (27.04.2025)

[27] Miuzei Case for Raspberry Pi 5, Raspberry Pi 5 Case with 4 Pieces Copper Heatsink and 40 x 10 mm PWM Fan, Balance Between Active and Passive Heat Dissipation, for Raspi 5 4GB 8GB, Clear Acrylic.  
[https://www.amazon.de/dp/B0CM36G2GG?ref=ppx\\_yo2ov\\_dt\\_b\\_fed\\_asin\\_title&th=1](https://www.amazon.de/dp/B0CM36G2GG?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1)

(27.04.2025)

[28] Netac Micro SD Card 32 GB Fat32, Memory Card for Smartphone, Switch, Tablet, Camera, Drone, Raspberry Pi.

[https://www.amazon.de/dp/B0833XXLJR?ref=ppx\\_yo2ov\\_dt\\_b\\_fed\\_asin\\_title&th=1](https://www.amazon.de/dp/B0833XXLJR?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1)

(27.04.2025)

[29] Raspberry Pi OS. <https://www.raspberrypi.com/documentation/computers/os.html>

(09.05.2025)

[30] Slingerland C. AWS Vs. Azure Vs. Google Cloud: Which Is Right For Your Organization? 2024. <https://www.cloudzero.com/blog/aws-vs-azure-vs-google-cloud/>

(02.05.2025)

[31] Amazon EC2. [https://aws.amazon.com/ec2/?nc2=type\\_a](https://aws.amazon.com/ec2/?nc2=type_a) (02.05.2025)

[32] AWS Lambda. [https://aws.amazon.com/lambda/?nc2=type\\_a](https://aws.amazon.com/lambda/?nc2=type_a) (02.05.2025)

[33] Perry M. AWS vs GCP vs Azure: Which Cloud Platform is Best for your Business? 2024. <https://www.qovery.com/blog/aws-vs-gcp-vs-azure/> (02.05.2025)

[34] Google Cloud. The new way to cloud starts here. <https://cloud.google.com/?hl=en> (02.05.2025)

[35] Microsoft Azure. Code it possible. <https://azure.microsoft.com/en-us/> (02.05.2025)

[36] Kishore N. Amazon Web Services vs. Azure vs. Google Cloud in 2025. <https://www.mgt-commerce.com/blog/amazon-web-services-vs-azure-vs-google-cloud/>

(02.05.2025)

[37] Azure App Service. <https://azure.microsoft.com/en-us/products/app-service> (12.05.2025)

[38] Cogan S. The naming is not related to nested virtualisation, it is related to the fact that the CPU provided to your VM. Microsoft Learn. 2018.

<https://learn.microsoft.com/en-us/archive/msdn-technet-forums/5a2a0e1b-8439-4c43-ae7f-11c50ad94cb0> (04.05.2025)

[39] Azure Blob Storage. <https://azure.microsoft.com/en-us/products/storage/blobs> (04.05.2025)

[40] Blobfuse2 - A Microsoft supported Azure Storage FUSE driver. <https://github.com/Azure/azure-storage-fuse> (04.05.2025)

[41] Microsoft. What is BlobFuse? - BlobFuse2. 2024. <https://learn.microsoft.com/en-us/azure/storage/blobs/blobfuse2-what-is> (04.05.2025)

[42] Nextcloud. Regain control over your data. <https://nextcloud.com/> (07.05.2025)

[43] Immich. <https://immich.app/> (08.05.2025)

- [44] PhotoPrism. <https://www.photoprism.app/> (09.03.2025)
- [45] Docker Compose. <https://docs.docker.com/compose/> (08.05.2025)
- [46] Cohen N. Performance Testing vs. Load Testing vs. Stress Testing. 2024. <https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing> (10.05.2025)
- [47] Apache JMeter. <https://jmeter.apache.org/> (10.05.2025)
- [48] k6. The best developer experience for load testing. <https://k6.io/> (10.05.2025)
- [49] Explore k6 extensions. <https://grafana.com/docs/k6/latest/extensions/explore/> (10.05.2025)
- [50] Locust. <https://locust.io/> (20.04.2025)
- [51] Agent-based versus agentless data collection: what's the difference? 2023. <https://www.logicmonitor.com/blog/agent-based-versus-agentless-data-collection-whats-the-difference> (11.05.2025)
- [52] Dix P. Time series database explained. <https://www.influxdata.com/time-series-database/> (11.05.2025)
- [53] Telegraf. <https://www.influxdata.com/time-series-platform/telegraf/> (20.04.2025)
- [54] Prometheus. From metrics to insight. <https://prometheus.io/> (14.05.2025)
- [55] Clifford J. Infrastructure Monitoring Basics with Telegraf, InfluxDB, and Grafana. 2023. <https://www.influxdata.com/blog/infrastructure-monitoring-basics-telegraf-influxdb-grafana/> (14.05.2025)
- [56] InfluxDB. <https://www.influxdata.com/> (20.04.2025)
- [57] Grafana. <https://grafana.com/> (20.04.2025)
- [58] Provost A. L. TIG stack (Telegraf/InfluxDB/Grafana). Github, 2020. <https://github.com/alekece/tig-stack> (20.04.2025)
- [59] Rivera L. Telegraf: system dashboard. <https://grafana.com/grafana/dashboards/928-telegraf-system-dashboard/> (20.04.2025)
- [60] Nielsen J. Powers of 10: Time Scales in User Experience. 2009. <https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/> (20.04.2025)
- [61] Nielsen J. Response Times: The 3 Important Limits. 1993. <https://www.nngroup.com/articles/response-times-3-important-limits/> (20.04.2025)
- [62] Docker. <https://www.docker.com/> (20.04.2025)
- [63] Fahim F. 8 Reasons Why Load Testing is Important? 2022.

- <https://serverguy.com/importance-of-load-testing/> (20.04.2025)
- [64] Importance of having a good monitoring system. <https://pandorafms.com/blog/why-you-need-a-monitoring-system/> (20.04.2025)
- [65] Python. <https://www.python.org/> (20.04.2025)
- [66] Indexing Your Originals. <https://docs.photoprism.app/user-guide/library/originals/> (20.04.2025)
- [67] GNU Bash. <https://www.gnu.org/software/bash/> (20.04.2025)
- [68] OpenAI. ChatGPT. <https://chat.openai.com> (01.03.2025)
- [69] DeepSeek. <https://chat.deepseek.com/> (28.02.2025)
- [70] Writing a locustfile. <https://docs.locust.io/en/stable/writing-a-locustfile.html> (18.04.2025)
- [71] Bret. How to Power the Raspberry Pi 5: A Complete Guide. 2025. <https://bret.dk/how-to-power-the-raspberry-pi-5-a-complete-guide/> (04.05.2025)
- [72] Kingsepp E. 2024. aastal keskmised elektri ja maagaasi börshinnad langesid. 2025. <https://www.konkurentsiamet.ee/uudised/2024-aastal-keskmised-elektri-ja-maagaasi-borshinnad-langesid> (04.05.2025)
- [73] Ben. How Long Does a Raspberry Pi Last? Understanding the Lifespan of Your Device. 2024. <https://fleetstack.io/blog/how-long-does-a-raspberry-pi-last> (04.05.2025)
- [74] Azure Blob Storage pricing. <https://azure.microsoft.com/en-us/pricing/details/storage/blobs/> (04.05.2025)
- [75] Gupta T. Load Testing With Locust. 2020. <https://medium.com/swlh/load-testing-with-locust-3e74349f9cbf> (15.05.2025)
- [76] What is latency? | How to fix latency. <https://www.cloudflare.com/learning/performance/glossary/what-is-latency/> (07.05.2025)
- [77] Azure Datacenters. <https://datacenters.microsoft.com/globe> (07.05.2025)
- [78] How Long Do Micro SD Cards Last? <https://colesclassroom.com/how-long-do-micro-sd-cards-last/> (11.05.2025)
- [79] Ontrack. SSD Lifespan: How Long do SSDs Really Last? 2024. <https://www.ontrack.com/en-ie/blog/how-long-do-ssds-really-last> (11.05.2025)
- [80] Turner N. Using an SSD with a Raspberry Pi. 2024. <https://neilturner.me.uk/2024/07/19/using-an-ssd-with-a-raspberry-pi/> (11.05.2025)
- [81] Shukla G. What Does "TBW" Mean for SSDs? 2022. <https://www.howtogeek.com/806926/what-does-tbw-mean-for-ssds/> (11.05.2025)
- [82] ETAIS. <https://etais.ee/> (15.05.2025)

## **Lisad**

### **Lisa 1. Repositoorium**

Töös kasutatud skriptid ning testide aruanded on saadaval GitHubi repositooriumis, mis asub veebiaadressil <https://github.com/SamantaSuitsik/joudlus-testimine>

## Lisa 2. Failinimede kogumise skripti loomine

Failinimede kogumise skripti genereerimiseks kasutati järgmist viipa: „using k6 testing tool. I have a lot of pictures in my folder and i want the k6 VUs to use these files for uploading to the site. How can i do that with my script?“. Peale mõne lisaküsimuse esitamist andis ChatGPT o3-mini rahuldava vastuse. Vastus on toodud koodiplokis 2.

```
#!/bin/bash
files=()
while IFS= read -r file; do
    files+=("${basename "$file"}")
done <<(find ./scripts -type f \( -iname "*.jpg" -o -iname "*.jpeg" -o -iname "*.png"
\))
echo "["${IFS=,; echo "${files[*]}"}"]" > fileList.json
```

Koodiplokk 2. Tehisintellekti ChatGPT 03-mini poolt loodud skript failinimede kogumiseks.

Genereeritud skriptis kohandati kaustade ning failide nimesid. Töös kasutatud lõplik skript on näha koodiplokis 3.

```
#!/bin/bash
files=()
while IFS= read -r file; do
    files+=("${basename "$file"}")
done <<(find ./pildid -type f \( -iname "*.jpg" -o -iname "*.jpeg" -o -iname "*.png"
\))
echo "["${IFS=,; echo "${files[*]}"}"]" > piltideNimed.json
```

Koodiplokk 3. Töös kasutatud skript failinimede kogumiseks.

### Lisa 3. Failide teisaldamise skripti loomine.

Skripti, mis teisaldab faile ühest kaustast teise, genereerimiseks kasutati järgmist viipa: „Can you write me a script that moves a photo from one dir to another. The script should read from a log file and filter out lines that contain the photo's name. The log file looks like this: time="2025-02-28T17:22:46Z" level=info msg="Photo:tumblr\_nrz3d8GPpU1uo1g48o5\_r1\_1280.jpg" source=console“

Logifaili muutmise tõttu esitati tehisintellektile DeepSeek R1 uue logifaili formaat, mille järel saadi rahuldav vastus, mis on toodud koodiplokis 4.

```
#!/bin/bash

# Configuration variables
SOURCE_DIR="/home/samanta/kool/loputoo/locust/photos"
DEST_DIR="/home/samanta/Pictures/thesis"
LOG_FILE="/home/samanta/kool/loputoo/locust/locust.log"

# Create destination directory if it doesn't exist
mkdir -p "$DEST_DIR"

# Process log file and move photos
grep 'Photo: ' "$LOG_FILE" | \
  sed -n 's/.*Photo: \(.*\)\/\1/p' | \
  sort -u | \
  while IFS= read -r filename; do
    source_path="$SOURCE_DIR/$filename"
    dest_path="$DEST_DIR/$filename"

    if [ -f "$source_path" ]; then
      mv -- "$source_path" "$dest_path"
      echo "Moved: $filename"
    else
      echo "Warning: File not found - $filename" >&2
    fi
  done

echo "Operation completed"
```

Koodiplokk 4. Tehisintellekti DeepSeek R1 poolt genereeritud skript failide teisaldamiseks.

Saadud vastuses kohandati kaustade nimesid ning lõplik skript on koodiplokis 5.

```
#!/bin/bash

# Configuration variables
SOURCE_DIR="./pildid"
DEST_DIR="./kasutatudpildid"
LOG_FILE="./locustrp.log"

# Create destination directory if it doesn't exist
```

```

mkdir -p "$DEST_DIR"

# Process log file and move photos
grep 'Photo: ' "$LOG_FILE" | \
sed -n 's/.*Photo: \(.*\)\/\1/p' | \
sort -u | \
while IFS= read -r filename; do
    source_path="$SOURCE_DIR/$filename"
    dest_path="$DEST_DIR/$filename"

    if [ -f "$source_path" ]; then
        mv -- "$source_path" "$dest_path"
        echo "Moved: $filename"
    else
        echo "Warning: File not found - $filename" >&2
    fi
done

echo "Operation completed"

```

Koodiplokk 5. Töös kasutatud skript pildifailide teisaldamiseks.

## Lisa 4. Locusti testi skript

Terve testimiseks kasutatud skript genereeriti tehisintellekti ChatGPT o3-mini-high poolt. Kõigepealt prooviti luua test pildi üleslaadimiseks kasutades järgmist viipa:

„I want to use locust to performance test my self hosted photoprism app. I am trying to write a test to upload a file.

I have file names in a certain dir and i have the name of the files listed in a file called fileList.json.

I want to get all the files from that fileList.json file and give them to different users to be uploaded to photoprism.

The photoprism accepts a POST request like this: `${BASE_URL}/api/v1/users/usrzn0i6w9iu0075/upload/${uploadId}`,

You also have to make a PUT request for the same url afterward so the photos can be processed.

Can you write me a locust test for this?“

Kuna esimene vastus ei toimunud, esitati tehisintellektile täpsustusi ja lisaküsimusi. Seejärel saadi sobiv vastus, mille leiab koodiplokist 6.

```
import os
import json
import random
import string
from locust import HttpUser, task, between

class PhotoPrismUser(HttpUser):
    wait_time = between(1, 3)

    with open("fileList.json") as f:
        file_list = json.load(f)

    def generate_upload_id(self):
        return ''.join(random.choices(string.ascii_letters + string.digits, k=10))

    @task
    def upload_file(self):
        upload_id = self.generate_upload_id()
        file_name = random.choice(self.file_list)
        file_path = os.path.join("your_files_dir", file_name)
        url = f"/api/v1/users/usrzn0i6w9iu0075/upload/{upload_id}"

        with open(file_path, "rb") as f:
            file_content = f.read()
```

```

files = {"files": (file_name, file_content, "image/jpeg")}
with self.client.post(url, files=files, catch_response=True) as res:
    if res.status_code not in [200, 201]:
        res.failure("POST failed")

put_payload = {"albums": []}
headers = {"Content-Type": "application/json"}
with self.client.put(url, data=json.dumps(put_payload), headers=headers,
catch_response=True) as res:
    if res.status_code not in [200, 201]:
        print("PUT failed:")
        print("Status code:", res.status_code)
        print("Response text:", res.text)
        res.failure("PUT failed")

```

Koodiplokk 6. Tehisintellekti ChatGPT o3-mini-high poolt loodud testi skript.

Hiljem küsiti abi koodi ülesehituse kohta ning loodi ka teised stsenaariumid. Samuti genereeriti ChatGPT 03-mini-high abiga testidele sobiv konfiguratsioon. Lõplik kood on toodud Githubi repositooriumis failina „scenarios.py“. Repositooriumi leiab lisast 1.

## Litsents

### Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Samanta Suitsik,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Raspberry Pi ja pilvserverite jõudlust ning kasutuskogemust võrdlev uuring, mille juhendaja on Pelle Jakovits, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commons'i litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Samanta Suitsik

**15.05.2025**