OULUN AMMATTIKORKEAKOULU

Taneli Turpeinen

PILVIPOHJAINEN TEKNOLOGIADEMONSTRAATIO VIDEON SUPER-RESOLUUTIOSTA

PILVIPOHJAINEN TEKNOLOGIADEMONSTRAATIO VIDEON SUPER-RESOLUUTIOSTA

Taneli Turpeinen
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, Ohjelmistokehitys

Tekijä: Taneli Turpeinen
Opinnäytetyön nimi: Pilvipohjainen teknologiademonstraatio videon superresoluutiosta
Työn ohjaaja: Teemu Korpela
Työn valmistumislukukausi ja -vuosi: Syksy, 2021 Sivumäärä: 29 + 19

Opinnäytetyön aiheena oli pilvipohjainen superresoluution testausalusta, jota tehtiin yritysharjoittelussa Visidon Oy:lle. Työssä keskityttiin superresoluution käyttöönottoon ja siihen liittyvien skaalautumis- ja kustannustekijöihin. Lisäksi työssä tehtiin mittauksia ja laskentaa koskien alustan käyttöönottoa pilvessä ja tarvittavien GPU-resurssien määrää.

Tavoitteina oli luoda alusta, jolla voidaan demonstroida yrityksen asiakkaille tekoälyä hyödyntäviä kuvan ja videon resoluution parannukseen liittyviä teknologioita. Alustan tuli myös olla skaalautuva muille prosessointialgoritmeille. Järjestelmän tai ainakin sen prosessointiketjun tuli olla siirrettävissä asiakkaan omaan pilveen tai julkiseen pilveen tuotantovaiheessa.

Harjoittelussa tehty järjestelmä oli rakennettu Docker-konteiksi (verkkosivusto, tietokanta, aliohjelmat, sivuston ja aliohjelmien rajapinta), jossa kontit kommunikoivat järjestelmän sisällä suojatusti käänteisen välityspalvelimen kautta.

Työn teoriaosassa tutkittiin, mikä on superresoluutio ja millaisia käyttömahdollisuuksia sillä on. Lisäksi selvitettiin pilvijärjestelmien eroja ja tehtiin vertailua niiden välillä. Valittiin myös opinnäytetyön aiheeseen sopiva pilvipalvelu.

Toteutusosassa kehitettiin testausalustan skaalautuvuutta muille algoritmeille ja luotiin näkymä algoritmin tuottamien työtietojen esittelyyn ja superresoluution tulosten vertailuun. Aliohjelmien puolella tehtiin algoritmin käyttöönotto ja parametrien vienti tietokannalta aliohjelmille.

Tavoitteet saavutettiin ja sivustolle tehtiin superresoluution käyttöönotto. Docker-konttien välinen viestintä toimii odotetusti ja järjestelmän käyttöönotto pilvessä on suunniteltu teoriassa Amazonin AWS:n instanssiin. Järjestelmän käyttöönotto vaatii vielä sivuston ulkoasun kehitystä, jotta päästäisiin tuotantovaiheeseen.

Asiasanat: ohjelmistokehitys, palvelun käyttöliittymä, tietokannat, Docker, Nginx

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Taneli Turpeinen
Title of thesis: Cloud based Super resolution video technology demonstration
Supervisor: Teemu Korpela
Autumn, 2021
Number of pages: 29 + 19

In this thesis my objective was to make a cloud based super resolution testing application for a company named Visidon Oy. I made a set up for super resolution and I studied and documented the scaling factors and the expenses of the system. I calculated the application expenses on a private cloud and the GPU resources it would need for processing the workload for multiple clients.

My goal was to create a platform, where company can demonstrate AI based image and video enchanted technologies for the clients. The platform was meant to be scalable to different algorithms like the super resolution. The system or at least a part of it should be moveable to client's own cloud in production phase.

I explained what Super resolution is and how it is used in general. Also, I explained what cloud computing is and what cloud computing models there are. I found out what cloud computing model is the right choice for this application.

The system was made with Docker and it was separated to multiple containers. The containers contain a Nginx reverse proxy, a website, two Application Programming Interfaces, a database and a processing node.

I improved the system and its scaling to another algorithms. I created views for work items and comparison for input and output files. I calculated needs for GPU requirements and chose Amazon AWS instance based on pricing. I improved the processing node to take parameters from website for processing input files.

As a result, I achieved these objectives and goals, and I made a set up for super resolution. The system needs more polishing for the outlook of the website before going to the production phase.

Keywords: software development, service interface, databases, Docker, Nginx

SISÄLLYS

1 JOHDANTO

Aloitin Oulun ammattikorkeakoulun ammattitaitoa edistävän harjoittelun Visidon Oy:ssä maaliskuussa vuonna 2021. Visidon Oy on oululainen teknologiayritys, joka on erikoistunut tekoälyä hyödyntäviin kuvan- ja videonkäsittelyn algoritmeihin. Visidon kehittää algoritmeja vastaamaan nykyajan vaatimuksia, jotka ovat korkea hyötysuhde, alhainen virrankulutus ja korkea tarkkuus. (1.)

Tehtäväni oli luoda verkkosivusto, joka mahdollistaisi yrityksen algoritmien testauksen internetissä. Verkkosivuston lisäksi tein tallennustilan, välityspalvelimen ja kaksi rajapintaa. Algoritmin sisältävän aliohjelman ja tietokannan otettiin käyttöön Docker-ympäristössä. Projektissa oli paljon tehtävää, ja se sisälsi erilaisia ohjelmistoja, jotka kommunikoivat keskenään. Ohjelmistojen testausta ja ylläpitoa varten rakensin ne Docker-ympäristöön ja jaoin ohjelmistot kontteihin, jotka ovat eristettyjä ohjelmistoympäristöjä.

Tässä opinnäytetyössä otettiin käyttöön superresoluution algoritmin aikaisemmin kehittämääni verkkosivulle. Kehitettiin myös verkkosivustoa, jotta sen käyttäminen ja tulosten vertailu olisi vaivatonta asiakkaalle. Järjestelmä tulee tulevaisuudessa internetin välityksellä saataville valittuun pilvipalveluun rakennettuna. Tätä varten tutkittiin järjestelmän tarvitsemia resursseja superresoluution prosessointiin ja kustannustekijöitä tulevassa projektin käyttöönotossa.

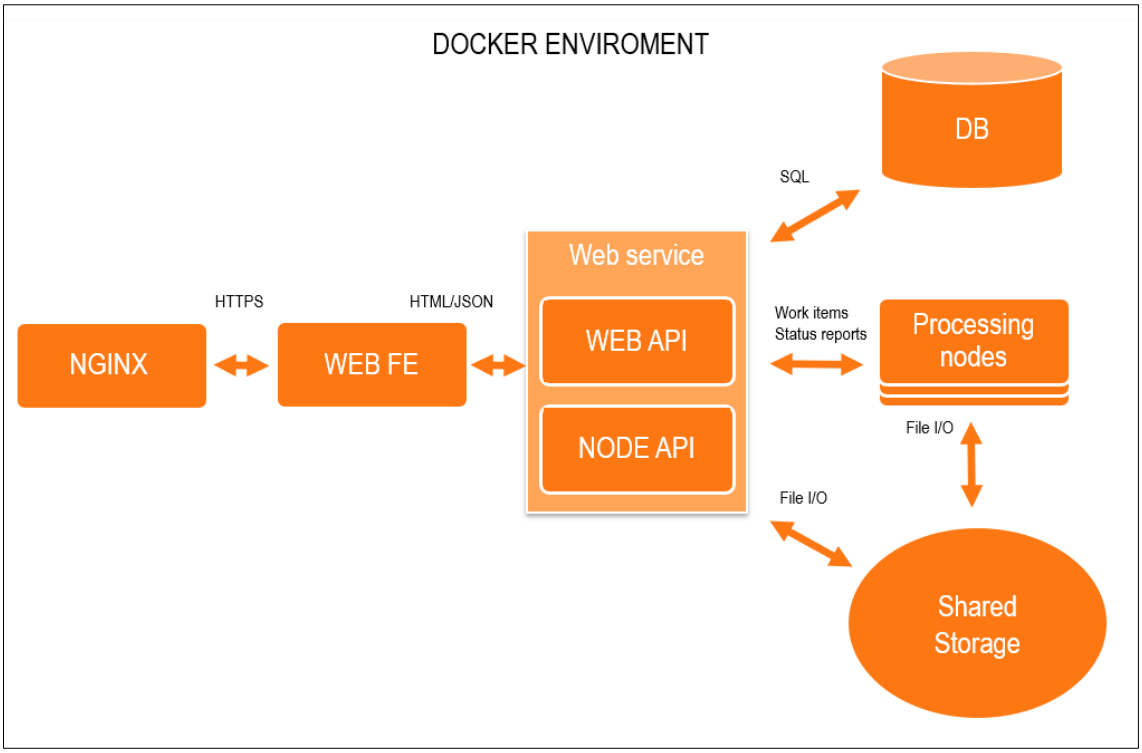
2 LÄHTÖKOHDAT

2.1 Docker

Docker on avoimeen lähdekoodiin perustuva ohjelmisto, ja se oli Stack Overflow 2021 -kyselyn mukaan toiseksi suosituin teknologia versionhallintajärjestelmä Gitin jälkeen (2). Se on avoin alusta sovellusten kehittämiseen, pakkaamiseen ja ajamiseen ja sen avulla ohjelmistokehittäjät voivat rakentaa sovelluksia alustasta riippumatta (3).

Docker yksinkertaistaa sovelluksen kehittämistä ja toteutusta pakkaamalla kaikki sovelluksen edellyttämät ohjelmistot yhdeksi ohjelmistoksi, jota kutsutaan Docker-kuvaksi. Kuva voidaan ajaa missä tahansa alustassa ja ympäristössä. (4.) Docker-tiedosto on tekstitiedosto, joka sisältää kaikki komennot, joita käyttäjä voi suorittaa komentorivillä kontin luomiseksi kuvan perusteella (5). Harjoittelussa rakennettiin välityspalvelin, sivusto, rajapinnat ja tietokanta omiin nimettyihin kontteihin ja määriteltiin Docker-tiedostot jokaiselle erikseen.

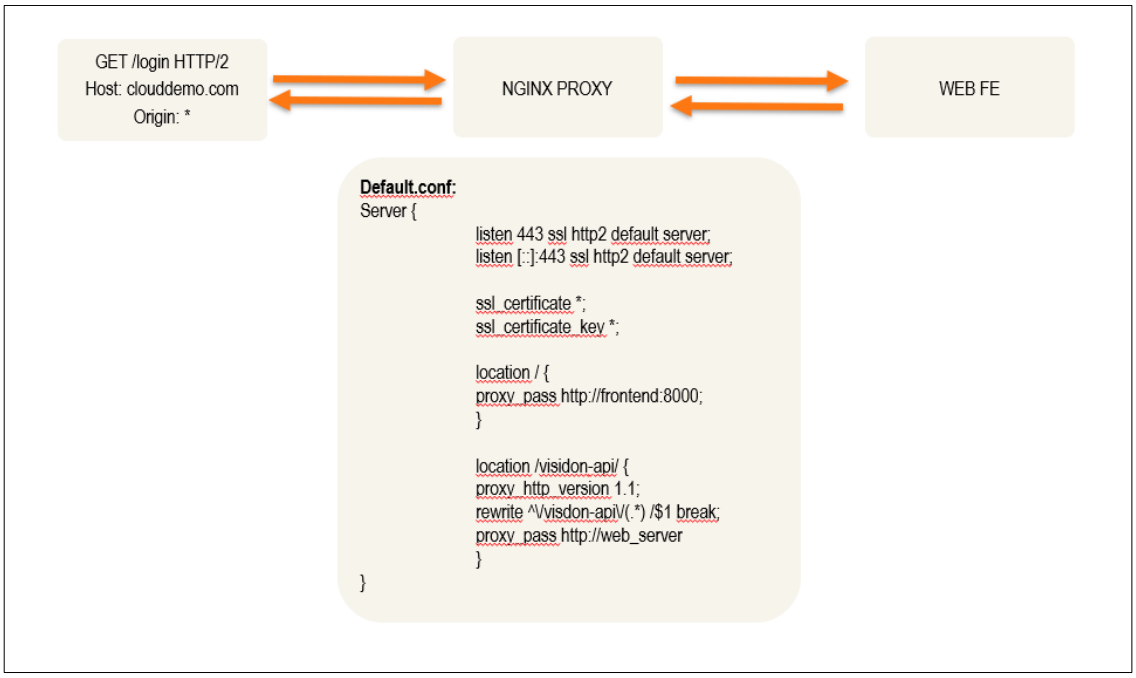
Docker Compose on työkalu, jolla voidaan määrittää ja käynnistää usean kontin järjestelmiä. Yhdellä ***docker-compose up*** -komennolla voi sitten luoda kontit ja käynnistää kaikki palvelut. (6.) Kuvassa 1 näkyvä kaavio esittää järjestelmää Docker-ympäristössä ja järjestelmän sisältyvien konttien välisiä suhteita.



KUVA 1. Järjestelmän kaavio Docker-ympäristössä. Sivusto: HTML, CSS, JS + Node, Sivuston ja aliohjelmien rajapinnat: PHP Slim V.4, Tietokanta: PostgreSQL v. 13, Aliohjelmat ja algoritmit: C++/ Python.

2.2 Nginx

Nginx-käänteisen välityspalvelimen tarkoitus oli tasapainottaa järjestelmän kuormitusta ja lisätä konttien turvallisuutta SSL/TSL-viestinnässä. Kaikki viestintä tapahtuu HTTP-pyyntöjen avulla. Käyttäjien tiedostot tallennetaan sivuston rajapinnan konttiin (WEB-API), josta ne ovat saatavilla HTTP-pyynnöllä. Tuotantovaiheessa internetin kautta tuleva viestintä käyttää HTTPS-protokollaa kaikissa yhteydenotoissa ja mahdolliset HTTP-pyynnöt ohjataan uudestaan välityspalvelimen avulla käyttämään HTTPS-protokollaa (7; 8). Kuvassa 2 näkyvä kaavio esittää käänteisen välityspalvelimen toimintaa konttien välillä. Kehitysvaiheessa käytettiin itsetehtyjä SSL-sertifikaatteja.

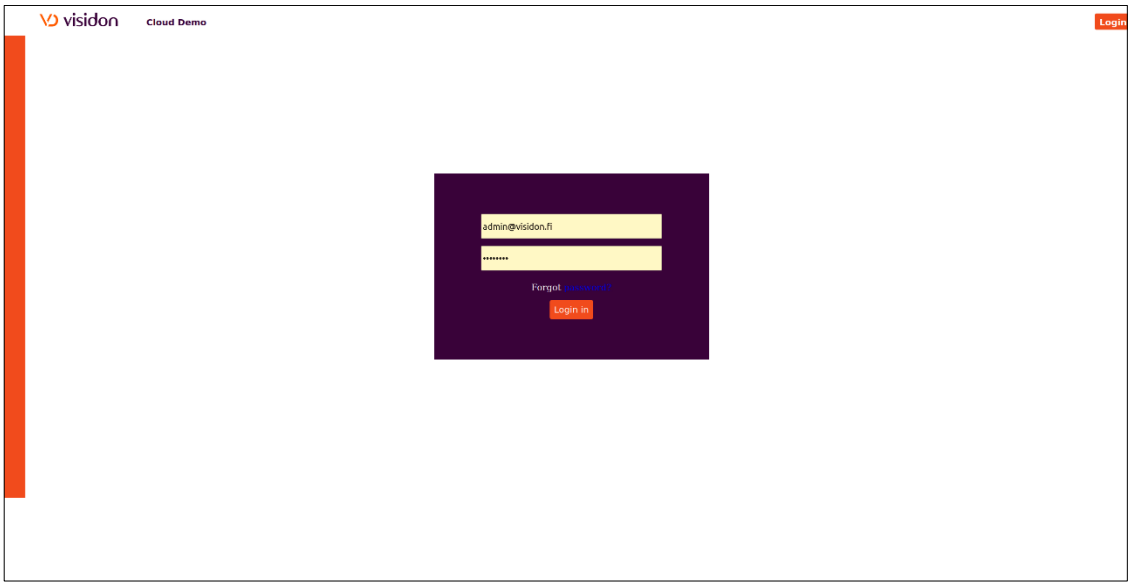


KUVA 2. NGINX käänteinen välityspalvelimen toiminta konttien välillä.

2.3 Verkkosivu

Verkkosivulle tehtiin jo harjoittelun aikana sisäänkirjautumisnäkymä ja tiedostojen vertailunäkymä. Käyttäjän antamat käyttäjätunnus ja salasana tarkistetaan tietokannasta ja istunto tallennetaan tietokantaan. Käyttäjälle annetaan istuntoavain, jolla sivusto ja rajapinnan toimintojen käyttö on mahdollista. Peruskäyttäjällä ja järjestelmänvalvojalla ovat erilliset HTML-sivut, koska järjestelmänvalvojalla on laajennetut toiminnallisuudet verkkosivustolla ylläpidon takia ja tukitehtävien takia. Ylläpitotehtävien aikana järjestelmänvalvoja voi estää muiden käyttäjien sisäänkirjautumisen ja käyttäjille esitetään sivuston huoltoilmoitus.

Onnistuneen sisäänkirjautumisen jälkeen käyttäjä ohjataan pääsivulle, jossa käyttäjä pystyy hallitsemaan ladattuja tiedostoja ja aliohjelman tuottamia prosessoituja tiedostoja. Lisäksi käyttäjä pystyy lataamaan ja poistamaan tiedostoja. Kuvassa 3 on sisäänkirjautuminen. Jokainen sisäänkirjautuminen tarkistetaan tietokannasta. Onnistuneen sisäänkirjautumisen jälkeen käyttäjä saa 30 minuutin mittaisen istunnon, joka tarkistetaan annetulla istuntoavaimella. Kuvassa 4 on pääsivu, jossa käyttäjä voi ladata tiedostoja. Ladatut tiedostot tulevat näkyville alhaalla vasemmalle ja prosessoidut tiedostot tulevat alhaalla oikealle. Tiedostolle voi luoda työtehtävän ja sen voi ladata tai poistaa.



KUVA 3. Sisäänkirjautuminen.

KUVA 4. Pääsivu

2.4 Tietokanta

SQL-tietokannan tehtävä on tallentaa käyttäjien, työtehtävien ja tiedostojen tietoja pysyvästi järjestelmän toiminnan takia (9, s. 6). Lisäksi käyttäjien tunnistetut sisäänkirjautumisen istunnot tallennetaan tietokantaan. Salasanoja ei tallenneta suoraan tietokantaan, vaan ne salataan käyttämällä Hash-algoritmia (9, s. 6–7). Tietokanta asettaa poistettaessa työtehtävät, asiakkaiden tilit ja tiedostot erillisiin tauluihin työtehtäviksi aliohjelmalle.

2.5 Rajapinnat

Verkkosivun rajapinnan (WEB API) tehtävä on toimia verkkosivun ja tietokannan välillä ja hallinnoida käyttäjien istuntoja istuntoavaimilla. Aliohjelman rajapinnan (NODE API) tehtävä on vastata aliohjelman viesteihin koskien työtehtävien asettamista työvaiheeseen ja tulosten tallentaminen oikeaan paikkaan. Lisäksi se tasapainottaa erilaisia aliohjelmien vastaanottamia työtehtäviä, jotta työtehtäviä suoritetaan tasaisesti eikä samanlaisia työtehtäviä jäisi liikaa jonoon suhteessa muihin työtehtäviin (9., s.10–14).

2.6 Aliohjelmat

Aliohjelmat ovat itsenäisiä ohjelmistoja, jotka toimivat omissa Dockerin konteissa. Niiden tehtävä on ottaa työtehtäviä vastaan tietokannasta oman rajapinnan (NODE-API) kautta ja prosessoida annetut tiedostot parametreilla (9, s. 14). Kun aliohjelma löytää työtehtävän tietokannasta, se asettaa työtehtävän itselleen, aloittaa työtehtävän prosessoinnin ja päivittää tehtävän vaiheita tietokantaan ja tuottaa prosessoidun tiedoston jaettuun tallennustilaan.

2.7 Tallennustila

Järjestelmän tallennustila tehtiin testausvaiheessa rajapinnan konttiin kansiomuodossa. Järjestelmän käyttöönoton vaiheessa sen paikka tulee olemaan valitussa pilvessä, johon viitataan URL-osoitteella tiedoston tallentamisen ja haun aikana (9, s. 5–6). Tiedostojen poistaminen tallennustilasta vaatii tiedoston siirtämisen poistettavien tauluun tietokannassa, josta sitten itsenäinen aliohjelma hoitaa poistamisen sille sopivalla ajanhetkellä. Tällä on tarkoitus keventää sivuston toimintaa, jotta työtehtävien poistaminen ei vaadi tiedostojen samanaikaista poistamista, vaan niiden poistaminen tapahtuu aliohjelmalle sopivana ajankohtana.

2.8 Opinnäytetyön päämäärä ja tavoitteet

Opinnäytetyön päämääränä oli alustalle määrätyn superresoluution algoritmin käyttöönotto ja näkymä, jossa tuloksien vertailu alkuperäisen ja prosessoidun tiedoston välillä on mahdollista. Lisäksi koko järjestelmä piti toimia internetin välityksellä, jotta asiakas voi lähettää tiedostonsa ja saada prosessoidun tuloksen käyttöönsä. Järjestelmän tuli myös olla skaalautuva muille

algoritmeille ja järjestelmän, tai ainakin osan sen prosessointiketjusta, tuli olla siirrettävissä asiakkaan omaan pilveen tai julkiseen pilveen.

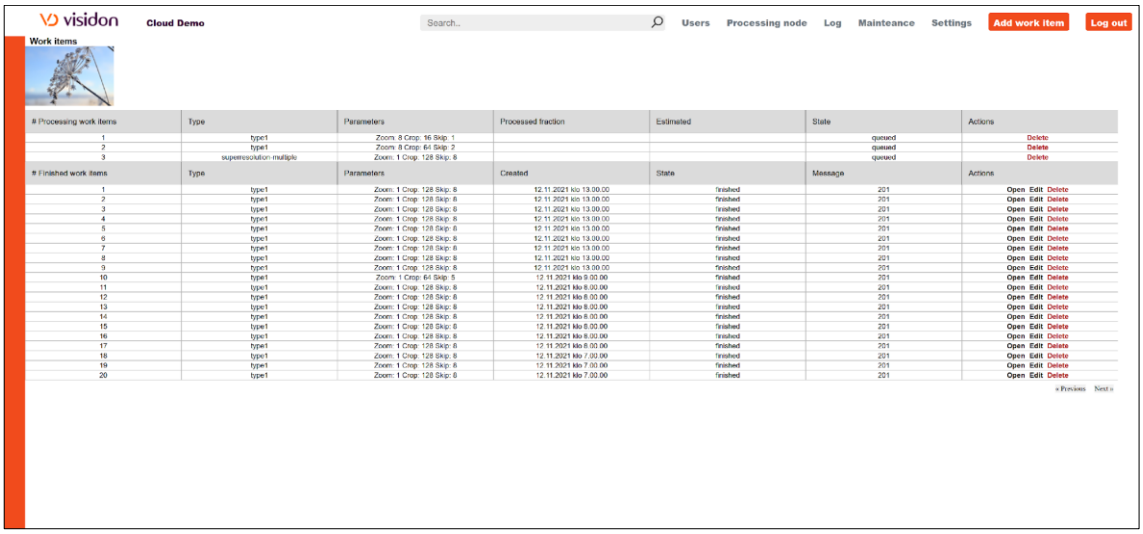
Opinnäytetyössä laskettiin myös riittävät GPU-resurssit vaadittavaan tiedostojen prosessointiin ja arvioitiin prosessoinnin kustannuksia. Lisäksi valittiin pilvipalvelu, joka soveltuu järjestelmän tarkoitukseen parhaiten. Muuta tutkittavaa oli järjestelmän integraatio muihin järjestelmiin ja se, miten edetään tuotantovaiheeseen mahdollisimman helposti ja kustannustehokkaasti.

3 JÄRJESTELMÄN KEHITYS

3.1 Pääsivuston kehitys

Sivustolle luotiin uusi pääsivu, jossa käyttäjä voi onnistuneen sisäänkirjautumisen jälkeen hallinnoida työtehtäviä tiedostojen sijaan. Uudella pääsivulla käyttäjä voi luoda uuden työtehtävän, selata työtehtäviä listanäkymässä, asettaa työtehtävän uudestaan aliohjelman käsiteltäväksi uusilla tiedoilla ja poistaa työtehtäviä tietokannasta ja tallennustilasta. Vanhasta pääsivusta kehitettiin vertailunäkymä, johon käyttäjä pääsee klikkaamalla avaa-nappia ja jossa käyttäjä voi tutkia ja vertailla tarkemmin valitun työtehtävän tiedostoja.

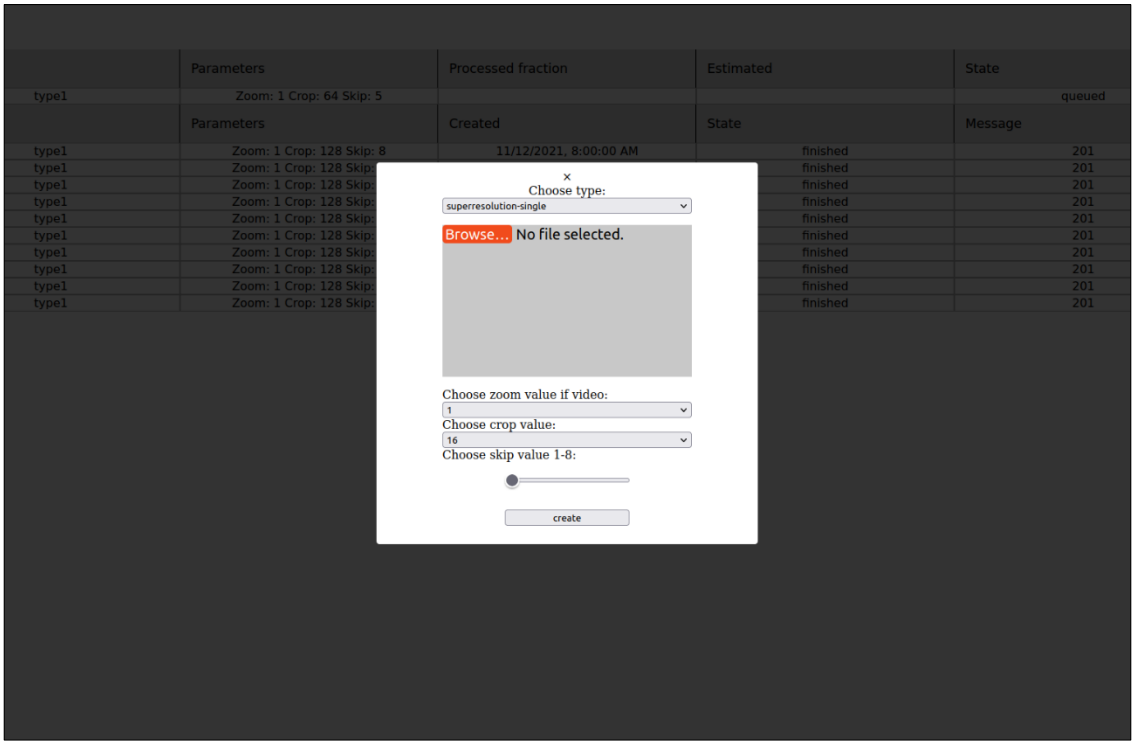
Kuvassa 5 on pääsivu, jolla käyttäjä voi hallinnoida aliohjelmalle asetettuja työtehtäviä. Prosessoinnissa olevat työtehtävät ylempänä, valmiit työtehtävät alhaalla. Yläpalkissa löytyvät toiminnot, käyttäjähallinta, aliohjelmien hallinta, lokitiedot, asetukset, työtehtävän asettaminen ja uloskirjautuminen. Valitun työtehtävän sisääntulotiedosto ilmestyy ylhäälle vasemmalle. Työtehtävän voi avata tiedostojen vertailusivulle, tehdä uuden työtehtävän asettamalla erilaiset parametrit ja poistaa työtehtävän.



KUVA 5. Järjestelmänvalvojan etusivu kirjautumisen jälkeen.

3.2 Työtehtävien luominen sivustolla

Aikaisemmin kaikki tiedostot tallennettiin tallennustilaan ennen työtehtävän parametrien asettamista ja työtehtävän tallentamista tietokantaan. Se mahdollisti käyttämättömien tiedostojen tallentamisen ilman rajoituksia, mikä kasvattaa tallennustilan tarvetta ilman syytä. Tähän kehitettiin ratkaisu, jossa sisääntulotiedoston lataaminen tallennustilaan vaatii työtehtävän asettamisen tiedostolle samanaikaisesti. Tiedostojen latausikkunassa pitää säätää työtehtävän parametrit, ja käyttämättömien tiedostojen lataaminen ei ole enää mahdollista. Kuvassa 6 asetetaan työtehtävä aliohjelmalle. Zoom-parametri mahdollistaa prosessoinnin tiedoston tiettyyn alueeseen, Crop-parametrilla säädetään prosessoitavan tiedoston käsittelylohkon kokoa ja Skip-parametrilla säädetään tiedoston ulkoreunan pikselien paksuutta.



KUVA 6. Superresoluutio-työtehtävän asettaminen aliohjelmalle.

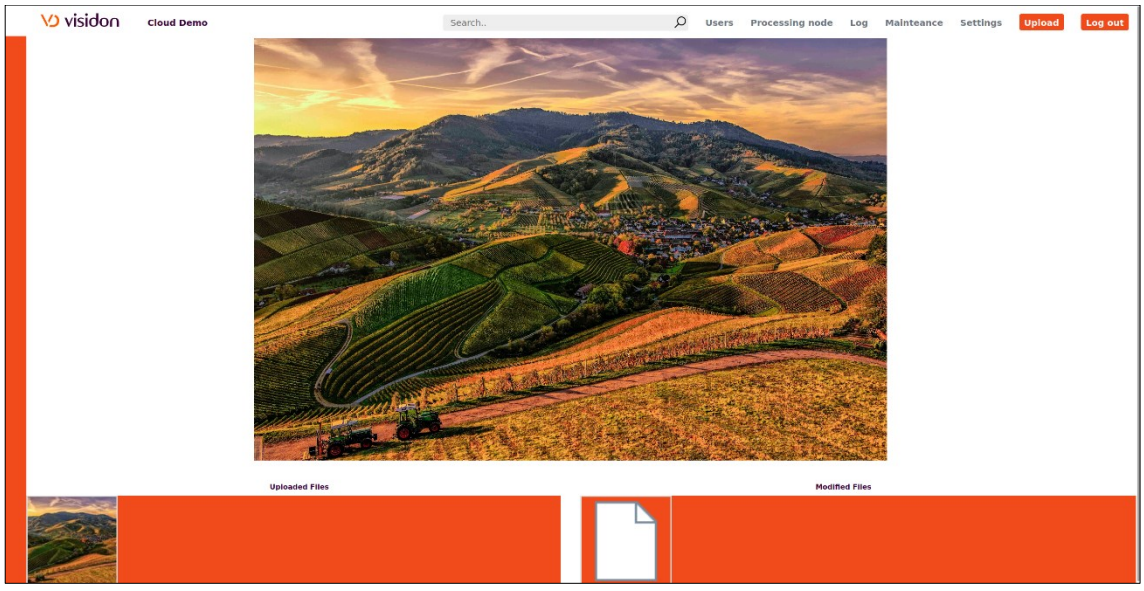
3.3 Superresoluutio

Superresoluutio-algoritmilla luodaan tiedostosta uusi tiedosto, jonka resoluutio on korkeampi. Näin parannetaan tiedoston kuvanlaatua ja mahdollistetaan parempi tarkennus yksityiskohtiin. Korkean resoluution tiedostot ovat hyödyllisiä esimerkiksi lääketieteessä, satelliiteissa, puolustusvoimissa, veden alla, kaukokartoituksessa ja teräväpiirtotelevisiossa (10; 11). Useimmiten digitaalisen

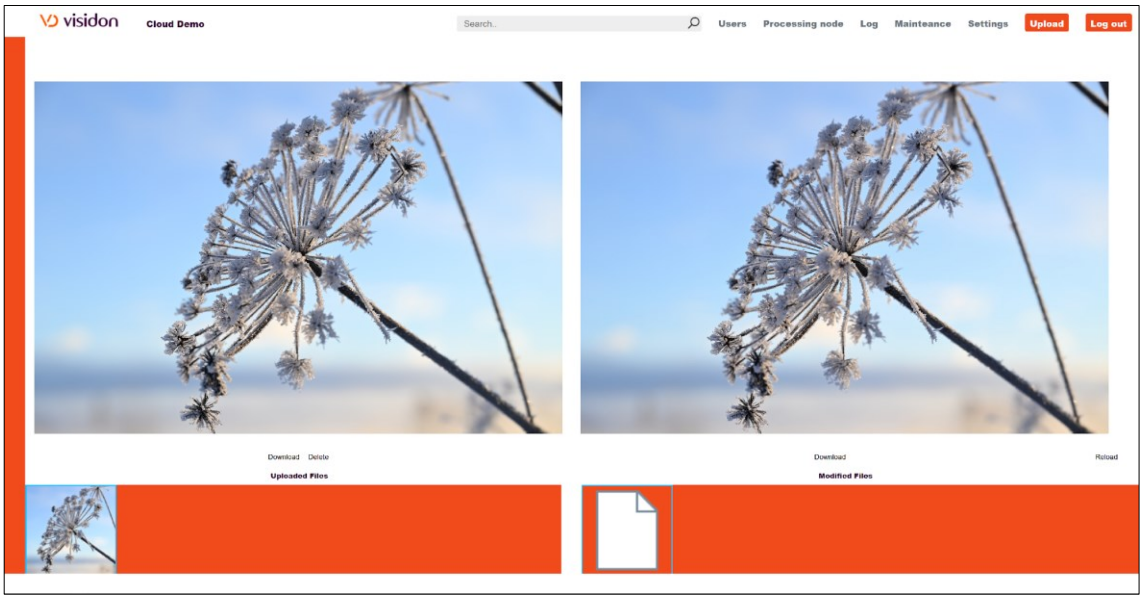
kuvantamisen sovelluksissa korkean resoluution kuvia ja videoita tarvitaan myöhempää kuvan käsittelyä ja analysointia varten. Tarve kuvallisten tietojen laadun parantamiseen tulee kahdesta eri sovellusalueesta: parannetaan ihmisen mahdollisuuksia tehdä tulkintaa ja edistetään koneen automaattisen havaintokyvyn käyttöä tiedon tulkinnassa (12, s. 1–2).

Superresoluutio-algoritmin tuottamien tiedostojen vertailu alkuperäisten tiedostojen kanssa tehtiin mahdolliseksi erillisessä vertailunäkymässä, jonne pääsee pääsivun kautta. Vertailunäkymä toteutettiin käyttäen HTML Canvasta. Aliohjelma tuottaa myös vertailutiedoston, jossa alkuperäinen ja prosessoitu tiedosto ovat yhdistetty toisiinsa. Tällä vertailutiedostolla käyttäjä voi klikkaamalla nähdä nopeasti prosessoinnin tulokset suhteessa alkuperäiseen tiedostoon. Kuvassa 7,8 ja 9 havainnollistetaan vertailunäkymän toimintaa.

KUVA 7. Aliohjelman tuottama kuva.



KUVA 8. Vertailukuva, jossa HTML Canvas piirtää yhdistetyn kuva toinen puoli kerrallaan (Kuva 7).



KUVA 9. Kuvien vertailu vierekkäin erillisinä tiedostoina.

3.4 Aliohjelmien toiminnan kehittäminen

Aliohjelma tarvitsee käyttäjän antamat työtehtävän parametrit prosessointia varten. Aliohjelman sisällä superresoluutio-algoritmi tuottaa prosessoidun tiedoston näiden parametrien perusteella. Lisäksi aliohjelma tuottaa Ffmpeg-kirjaston avulla tiedostot H264-muodossa sivustoa varten ja liittää videotiedostoissa ääniraidan alkuperäisestä tiedostosta (13). Isommat tiedostot skaalataan FullHD-laatuisiksi, jotta niiden esittäminen sivustolla olisi kevyempää. Kahden 4K-videon esittäminen yhtä aikaa HTML-canvasilla olisi hyvin raskasta ja käyttäjäkokemus olisi epämiellyttävä.

3.5 Superresoluutio rajapintojen puolella

Verkkorajapinta sisältää HTTP-pyynnöt, joilla verkkosivusto kommunikoi rajapinnan ja tietokannan välillä. Superresoluution rajapinta ottaa vastaan sisääntulotiedostot ja sijoittaa ne yhteiseen tallennustilaan. Rajapinta käsittelee lisäksi työtehtävien asettamisen tietokantaan, seuraa työtehtävien kehitystä aliohjelmissa ja hakee prosessoidut tulokset, jotka aliohjelma tuottaa yhteiseen tallennustilaan. Lisäksi rajapinta tallentaa sivustolla tapahtuvia HTTP-pyyntöjä lokiin tekstitiedostoihin, joilla voidaan kerätä tietoja sivustolla tapahtuvista toiminnoista.

Aliohjelmat tekevät HTTP-pyyntöjä omaan aliohjelma-rajapintaan, joka hakee tietokannasta avoimena olevia työtehtäviä ja tasaa erilaisten työtehtävien kuormitusta. Työtehtävän saatuaan aliohjelmat hakevat sisääntulotiedoston tai -tiedostot ja prosessoivat ne annetun työtehtävän ja parametrien mukaisesti ja tuottavat ulostulotiedoston tai -tiedostoja. Työtehtävän suoritettuaan aliohjelma hakee taas avoimen tehtävän ja aloittaa uuden prosessin. Myös tämä rajapinta kerää lokitietoja aliohjelmien toiminnasta.

3.6 Skaalautuvuus

Sivustolle on tarkoitus ottaa käyttöön myös muita yrityksen algoritmeja. Tulevat algoritmit voivat vaatia yhden tai useamman tiedoston sisääntulossa ja aliohjelmien prosessoinnin jälkeen lopputuloksena voi olla yksi tai useampi tiedosto. Tämän mahdollistin toteuttamalla useamman tiedoston rinnakkaisen lataamisen samanaikaisesti työtehtävän asettamisen kanssa. Tässä useamman sisääntulotiedoston työtehtävässä asetetaan halutut tiedot aliohjelman parametreiksi ja sisääntulotiedostot merkitään mukaan työtehtävään ja ne löytyvät yhteisestä tallennustilasta. Tiedostojen vertailussa tuodaan kaikki tiedostot näkymään, jotka olivat prosessoinnissa mukana.

4 JÄRJESTELMÄN PILVEN VALINTA JA JÄRJESTELMÄN KÄYTTÖÖNOTTO PILVESSÄ

Pilvi on kielikuva, jolla viitataan internettiin. Pilvipalveluilla tarkoitetaan mallia, jossa tietotekniikkaresursseja tarjotaan verkon välityksellä käyttöön ilman, että käyttäjän tarvitsee tietää resurssien sijaintia tai huolehtia niiden toiminnasta ja ylläpidosta (14., s. 16–17).

Yhdysvalloissa julkishallinnon standardeja pohtiva ja paikallisen elinkeinoministeriön alainen National Institute of Standards and Technology (NIST) määrittelee pilvipalvelun ominaispiirteet seuraavasti (15):

- itsepalvelu
- pääsy palveluihin eri päätelaitteilla
- resurssien yhteiskäyttö
- nopea joustavuus
- käytön tarkka mittaaminen.

4.1 Saas, Paas tai IaaS

NIST listaa myös pilvipalvelumallit: 1. ohjelmisto palveluna (SaaS), 2. alusta palveluna (PaaS) ja 3. infrastruktuuri palveluna (IaaS).

SaaS-mallissa (engl. Software as a service) tarjotaan ohjelmistoa, jossa asiakas toimii tarjotun käyttöliittymän kautta. Asiakas ei hallinnoi tai ohjaa pilvi-infrastruktuuria, kuten verkkoa, servereitä, käyttöjärjestelmää, tallennustilaa tai yksittäisiä sovelluksia. (16.)

PaaS-mallissa (engl. Platform as a service) asiakas voi ottaa käyttöön pilvessä asiakkaan luoman tai hankkiman sovelluksen, joka on luotu käyttäen tuettuja ohjelmointikieliä, kirjastoja, palveluita ja työkaluja. Kuluttaja ei hallitse verkkoa, servereitä, käyttöjärjestelmää tai tallennustilaa, mutta kontrolloi käyttöönotettua sovellusta ja mahdollisesti sovellukseen liittyviä asetuksia sovellusympäristössä. (16.)

IaaS-mallissa (engl. Infrastructure as a service) asiakkaalla on mahdollista hallinnoida järjestelmää enemmän kuin aikaisemmissa malleissa. Esimerkiksi asiakas voi hallita tallennustilaa, verkkoa ja muita tietokoneresursseja. Asiakas voi myös luoda mallissa käyttöjärjestelmiä ja sovelluksia. (16.)

Valitsin pilvipalvelumalleista IaaS:n, koska rakentamani järjestelmä tarvitsee serverin ja tallennustilan sisään- ja ulostulotiedostoille. Tallennustilan- ja resurssien hallinta on tarpeen, koska aliohjelmat käyttävät GPU-resursseja ja järjestelmän tallennustilan määrää voidaan tarvita skaalautuvasti. Järjestelmä käynnistetään pilvipalvelussa Docker-ympäristössä ja tallennustilaan viitataan sen IP-osoitteella.

4.2 Yksityinen, hybridi tai julkinen pilvi

NIST määrittelee myös pilvipalveluiden pilvityypit. Julkinen pilvi on yleisin pilvipalvelun malli. Käyttäjä pääsee käsiksi palveluihin julkisen internetin kautta ja sitä käytetään yhdessä muiden käyttäjien kanssa. Hallinnoinnista, laitteistosta, ohjelmistosta ja palveluista vastaa palveluntarjoaja. (15; 16.)

Yksityinen pilvi (VPC, Virtual Private Cloud) on malli, jossa pilvipalvelu on yksityisen organisaation käytössä. Pilvipalveluinfrastruktuuri voi olla käyttäjän tai kolmannen osapuolen omistuksessa ja hallinnassa, tai osittain käyttäjän ja osittain kolmannen osapuolen omistuksessa ja hallinnassa. Laitteisto voi sijaita käyttäjän tiloissa tai kolmannen osapuolen tiloissa. (15; 16.)

Yhteisöpilvi on tietyn kuluttajayhteisön yhteinen pilvi, joka on rakennettu vastaamaan yhteisön tarpeita ja käyttöä. Hallinnoinnista voi vastata ulkopuolinen taho ja laitteisto voi sijaita organisaatioiden ulkopuolella (15; 16).

Hybridipilvi on yhdistelmä, jossa yhdistetään osia kahdesta tai useammasta mallista. Osa arkkitehtuurista on yksityistä tai yhteisöllistä ja osa julkista. (14; 15; 16).

Valitsin yksityisen pilven, koska soveltuu parhaiten järjestelmän, yrityksen ja sen asiakkaiden tarpeisiin. Yksityinen pilvi mahdollistaa järjestelmän hallinnan turvallisessa ympäristössä, jossa voidaan tehdä järjestelmän testausta ja kehittää sen erilaisia menetelmiä ja algoritmeja (17).

Yksityinen pilvi ulkoiselta palveluntarjoajalta antaa myös mahdollisuuden pienentää laite- ja ylläpitokustannuksia paremmin kuin muut vaihtoehdot (18).

4.3 Järjestelmän soveltuvuus yksityiseen pilveen

Järjestelmä on siirrettävissä tuotantovaiheessa yksityiseen pilveen. Dockerin verkkosivulta löytyvät ohjeet järjestelmän käyttöönotosta pilvessä Azuren ja Amazonin tuetuissa pilviratkaisuissa (19; 20). Kontit, jotka sisältävät ohjelmistokoodin, voidaan käynnistää serverin sisällä ja viestintä tapahtuu suojatussa ympäristössä ja tietoliikenne kulkee Nginx-kontin kautta. Sivusto toimii kutsumalla järjestelmän Nginx-kontin IP-osoitetta, jolle voidaan myös asettaa verkko-osoite. Linux-käyttöjärjestelmä vaatii Docker Compose CLI:n asennuksen, jolla saadaan tarvittavat paketit Azuren ja Amazonin käyttöönottoon Dockerin kanssa (19; 20).

Jaettu tallennustila järjestelmän käyttäjille voidaan ottaa käyttöön pilvessä tarkoitukseen sopivassa ratkaisussa. Tallennustilan koko riippuu samanaikaisten käyttäjien määrästä, tiedostojen koosta ja algoritmien toiminnasta. Yrityksen arvion mukaan samanaikaisia käyttäjiä olisi noin kymmenen ja lisäksi tarvitaan testikäyttäjiä. Kaikki huomioon ottaen tallennustilan minimi on 16 teratavua (9, s. 5–6).

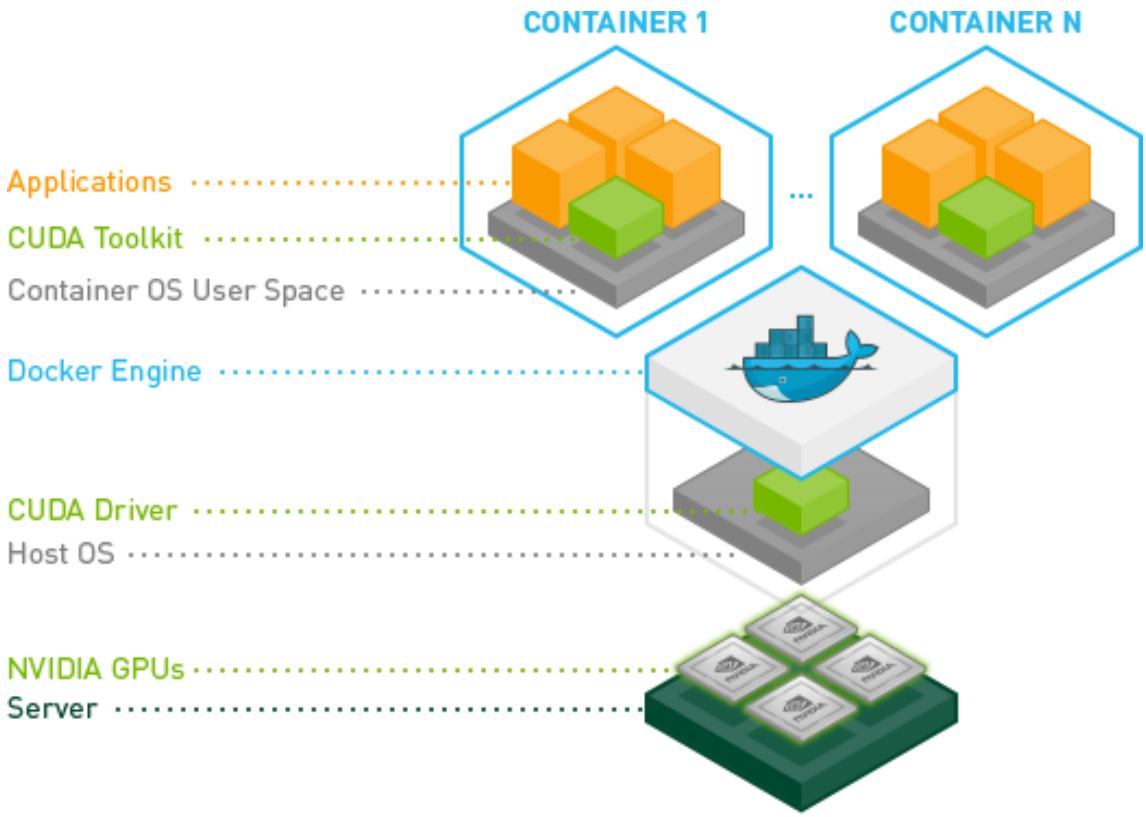
Tallennustilan tarkoitus on säilyttää käyttäjien lataamat tiedostot ja niistä aliohjelmien prosessoimat tiedostot lopputuloksena. Kuvien koko on 100 kB – 10 MB. Videoiden koko voi vaihdella välillä 100 MB – 10 GB. Ulostulotiedostot voivat olla jopa neljä kertaa isompia, jos esimerkiksi videon ulostulotiedoston bittinopeutta kasvatetaan prosessoinnin aikana. Tietokannassa jokaiselle käyttäjälle on määritelty tallennustila, jonka koko on 1024 GB (9, s. 5–6).

4.4 GPU-resurssit

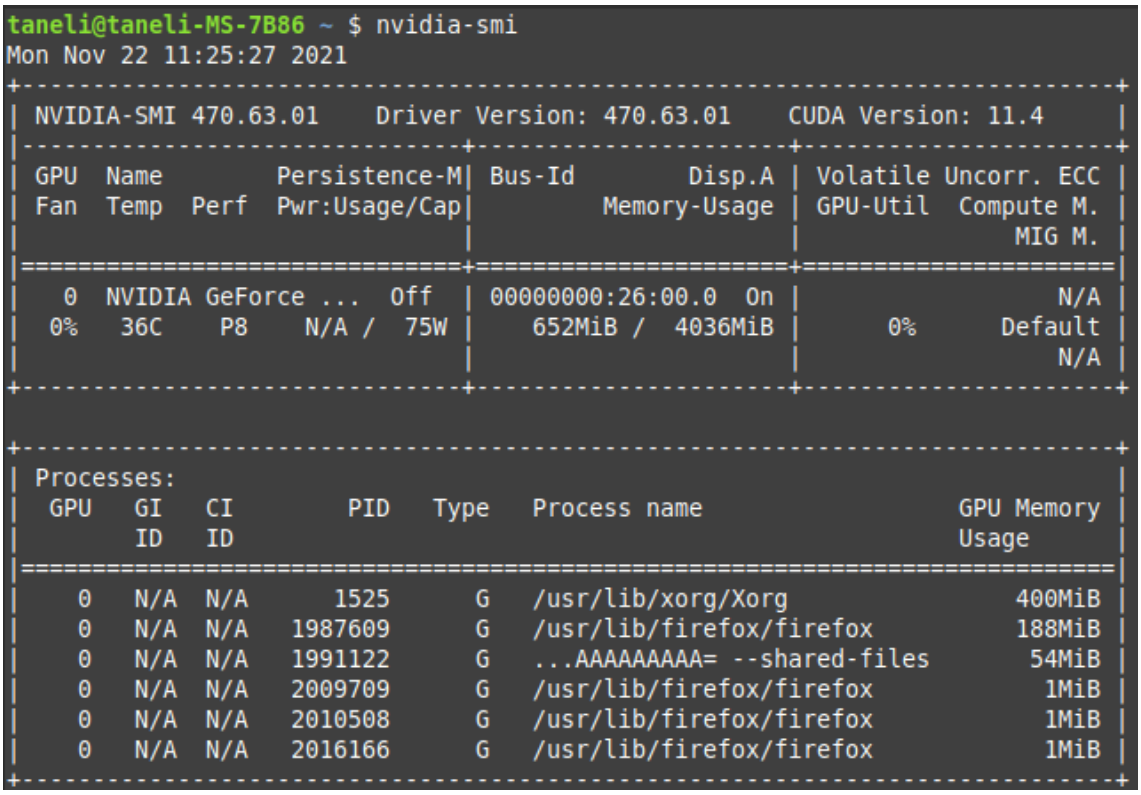
Yrityksen arvio järjestelmän samanaikaisista käyttäjämääristä on kymmenen yritysasiakasta ja heidän käyttötarpeensa on noin 100 tuntia videota kuukaudessa eli noin 25 tuntia viikossa (liite 5). Videon laadusta riippuen datan määrä voi vaihdella 50–375 gigatavun välillä viikon aikana Päiväkohtainen data voi siis olla välillä 7–54 gigatavua. Arvioissa on otettu huomioon mahdollinen minimi ja maksimi: 1080p25 = 8 Mbps-4Kp24 = 250 Mbps (liite 2).

Testasin aliohjelman laskentaa tietokoneeni prosessorilla ja totesin, että sen laskentateho ei riittänyt isompien videoiden prosessoimiseen kohtuullisessa ajassa. Aliohjelma tarvitsee näytönohjain-resursseja, koska näytönohjaimet on optimoitu graafiseen laskentaan. Ne tarjoavat enemmän tietokoneen säikeitä ja hyödyntävät niitä tehokkaammin kuin prosessorit (21).

Näytönohjaimen laskennan käyttöönotto vaati aliohjelman kontin sisälle Nvidian CUDA-työkalukirjaston ja paikalliselle koneelle tarvittiin Cuda:n ajurit. Aliohjelma käytti algoritmin koodissa Tensorflow-versiota 2.7, johon tarvittiin yhteensopivat Cuda versio 11.2 ja Cudnn versio 8.0. Asentaminen tapahtui helposti käyttämällä Nvidian virallista nvidia/cuda-kuvaa aliohjelman Docker-tiedostossa, joka asensi tarvittavat paketit kontin sisälle (22). Tietokoneelleni asensin CUDA:n ajurit Nvidian verkkosivulta. Kuvassa 10 ovat tarvittavat kirjastot ja ajurit, jotta aliohjelman kontissa voidaan käyttää GPU-resursseja (23). Kuvassa 11 Nvidian ajurit ja Cuda ovat asennettuina testauksessa tietokoneella.



KUVA 10. Dockerin ympäristössä tarvittavat kirjastot ja ajurit.



KUVA 11. Nvidian ajurit ja Cuda asennettuna paikallisella tietokoneella.

Testasin omalla tietokoneellani paikallisesti aliohjelmien videotiedostojen prosessointia näytönohjaimella. Superresoluution algoritmissa parametrit olivat: Zoom: 1 (Prosessoitava osa tiedostosta), Crop = 128 (lohkon koko) ja Skip = 1 (reunapikselien reunan paksuus). Tarkoituksena tällä testauksella on määrittää riittävä näytönohjaimien määrä yksityisessä pilvessä, jotta järjestelmä pystyy suoriutumaan vaadituista tehtävistä.

Testeissä videomateriaalina käytin puhelimella (Samsung Galaxy S20 Ultra) kuvattua videotiedostoa, joka oli kuvattu toimistolla. Video oli 4K MPEG-4-laatua ja tiedoston koko oli noin 75 MB.

GPU-resursseilla on ympärivuorokautisessa toiminnassa prosessoinnille aikaa 720 tuntia kuukaudessa, ja sen aikana pitäisi prosessoida asiakkaille noin 100 tuntia videota. Koska videotiedostojen kokoon vaikuttaa eniten bittinopeus, sen avulla voidaan tehdä karkea arvio asiakkaiden videoiden koosta tapauksessa, jossa video on korkeinta mahdollista laatua (24). Yrityksen asiakkaiden ennakkoon antamien tietojen mukaan prosessoinnin laadun tarve on

korkeimmillaan 50 Mbit/s, mistä voidaan olettaa, että suurin mahdollinen prosessoitavien tiedostojen yhteiskoko on 2 250 GB (liite 2).

Paikallisessa testauksessa työkoneeni näytönohjaimella (25) 4K-videon prosessointi kesti 44 minuuttia ja 35 sekuntia. Kaavalla voidaan karkeasti laskea vaadittavan työn määrä, jos työ olisi ollut vaadittavat sata tuntia (liite 3). Päädyin lopputulokseen, että näytönohjaimella Gtx 1050 TI yhden tunnin videon prosessoimiseen menisi aikaa noin 222 tuntia. Tämä on arvio, joka ei ota huomioon suurempien tiedostokokojen vaikutusta videotiedoston purkamisessa ja muita prosessoinnin ohessa tarvittavia toimintoja.

Laskin myös, että näytönohjaimen GTX 1080 (26) käytöllä prosessointiaikaa voidaan lyhentää noin 75 %, näytönohjaimella RTX 2080 TI (27) aika olisi 84,4 % lyhyempi ja näytönohjaimella RTX 3090 (28) aika olisi 94,1 % lyhyempi (liite 1). Lopulta tulin siihen lopputulokseen, että tarvittavien GPU-resurssien määrä olisi vähintään kaksi RTX 3090 -näytönohjainta valittuun pilvipalveluun, jotta järjestelmän vaatimukset täyttyisivät (liite 4, liite 5).

Yrityksen serverillä oli näytönohjaimia tehokkaaseen laskentaan, joten päätin testata tehokkaampia näytönohjaimia käytännössä ja vertailla tuloksia teorian kanssa. Tätä varten rakentamani järjestelmä piti ottaa käyttöön serverillä, asentaa tarvittavat kirjastot ja ottaa sivusto käyttöön SSH-protokollan kautta. Valitettavasti testaus ei onnistunut, koska aliohjelman käyttämä Tensorflow 2.7 (29) tarvitsi yhteensopivat Nvidian ajurit asennettuina serverille ja aikaisempi versio Tensorflow 2.6 ei ollut yhteensopiva aliohjelmalle.

4.5 Pilven kustannukset

Pilven kustannukset vaihtelevat paljon palveluntarjoajan mukaan, ja hinta muuttuu nopeasti järjestelmän tarpeiden ja asiakkaiden todellisen käytön mukaan. Onneksi monien palveluntarjoajien sivustoilla on mahdollisuus tehdä laskelmia järjestelmien ennakkotietojen perusteella. Pilvessä tarvitaan Docker-järjestelmälle serveri, jolla on mahdollista käyttää aliohjelmien kautta GPU-resursseja. Yhteensä GPU-resursseja tarvitaan noin 65 TFLOPS. (liite 2)

Tein kustannusarviot Dockerin tuetuista pilvipalveluista Azuresta ja Amazon AWS ja lisäksi Google Cloudista (liite 5). Amazonin AWS tarjosi mahdollisuuden ottaa järjestelmä käyttöön EC2-

instanssissa, jolle voi ottaa käyttöön näytönohjaimia. Instanssia valitessa otin huomioon näytönohjainten tarvittavan määrän, tallennustilan tiedostoille ja vertailin suoraan pakettihintoja toisiinsa. Paras tarjous oli g4dn.metal -instanssi, jossa oli 8 kappaletta Nvidia T4 -näytönohjaimia 3 vuoden suunnitelmalla 1918 euroa kuukaudessa kolmen vuoden sopimuksella. Google Cloud ja Microsoft Azure eivät pystyneet kilpailemaan hinnoissa. Hinta jäi isoksi, koska se on laskettu ennalta-arvioidun asiakkaiden maksimitarpeiden mukaan. Lisäksi GPU-resurssien määrää voidaan optimoida algoritmin kehittämisellä ja hinta-arviota tarkentaa jatkotestausten avulla.

4.6 Tuotantovaiheeseen eteneminen

Tuotantovaiheeseen on mahdollista mennä nopeasti vain superresoluutio-algoritmilla, kun sivuston järjestelmänvalvojan hallinta- ja ylläpitotoiminnallisuudet ovat valmiita ja valittu GPU-instanssi on otettu käyttöön. Myöhemmin järjestelmään voidaan lisätä muita algoritmeja ohjelmistopäivityksillä ja kehittää järjestelmän muita ominaisuuksia järjestelmällisesti. Sivuston ulkoasua ja käytettävyyttä on myös kehitettävä lisää, jotta se täyttää käyttöönoton vaatimukset. Sivuston visuaalisen ulkoasun pitäisi olla huoliteltu ja antaa käyttäjälle vaikutelman, että kyseessä on korkealaatuinen tuote (9, s. 10). Tämän jälkeen järjestelmä on valmis tuotantoon ja siirrettävissä valittuun instanssiin.

Järjestelmä on käynnistettävissä Docker-tiedostojen avulla helposti yhdellä komennolla: ***docker-compose up***. Aliohjelmat on rekisteröitävä tietokantaan ja asetettava niiden tunnistautumistiedot aliohjelman Config-tiedostoon, jotta rajapinta tunnistaa ne oikeiksi aliohjelmiksi. Aliohjelmien käynnistäminen tapahtuu erikseen ajamalla niiden käynnistystiedosto. Lisäksi yhteinen tallennustila on asetettava aliohjelmiin ja verkkosivulla, jotta tiedostot tallentuvat oikeaan paikkaan.

5 YHTEENVETO

Tässä opinnäytetyössä tavoitteena oli tehdä pilvipohjainen superresoluution testausalusta. Sivustolla piti pystyä kirjautumaan sisään ja asettamaan työtehtäviä halutuille kuva- ja videotiedostoille. Tiedostot tallennettiin järjestelmässä ja työtehtävä asetettiin aliohjelmalle käsiteltäväksi. Aliohjelma tuotti superresoluutio-algoritmin kautta prosessoidun tiedoston, jonka käyttäjä pystyi näkemään sivustolla ja vertailemaan saavutettuja muutoksia alkuperäisen tiedoston kanssa. Työtehtävä oli mahdollista asettaa uudestaan ja tiedostojen lataus piti olla mahdollista. Työtehtävien poistaminen asetti liitetyt tiedostot poistettaviksi. Lisäksi järjestelmälle tuli tutkia pilvimalleja, skaalautuvuutta ja kustannuksia mahdollisista käyttöönotosta tuotantovaiheessa.

Opinnäytetyön tuloksena suurin osa tavoitteista saavutettiin ja saatiin toimiva järjestelmä Docker-ympäristössä. Docker-konttien välinen viestintä toimi ja järjestelmän käyttöönotto pilvessä on tutkittu teoriassa. Projektin jatkon kannalta kehitettävää jäi verkkosivuston toimintoihin ja visuaaliseen puoleen, jotta tuotantovaiheeseen päästäisiin superresoluutio-algoritmilla. Verkkosivustolla tulisi ottaa käyttöön myös superresoluution lisäksi myös muut yrityksen algoritmit.

Opinnäytetyön aihe oli laaja ja työtehtävät vaihtelivat paljon. Dockerin käyttöjärjestelmässä verkkosivuston kehittäminen tapahtui JavaScriptillä, HTML:llä ja CSS:llä, Rajapintojen kehittäminen tapahtui PHP:llä, tietokannan toteuttaminen tapahtui käyttäen PostgreSQL:ää ja aliohjelmien C++-ohjelmointikieltä. Opin siis paljon harjoittelun ja opinnäytetyön aikana ja työ oli erittäin mielenkiintoista.

LÄHTEET

1. Visidon. What We Do. Hakupäivä 24.11.2021. https://www.visidon.fi/.

2. Stackoverflow 2021. Developer Survey. Hakupäivä 21.11.2021. https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies.

3. Docker. Hakupäivä 29.11.2021. https://www.docker.com/.

4. Vohra, Deepak 2015. Pro Docker. Appress. Hakupäivä 26.11.2021. O'Reilly Media. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/pro-docker/9781484218303/9781484218297_Ch01.xhtml.

5. Docker docs. Dockerfile reference. Hakupäivä 26.11.2021. https://docs.docker.com/engine/reference/builder/.

6. Docker docs. Overview of Docker Compose. Hakupäivä 2.12.2021. https://docs.docker.com/compose/.

7. NginX Docs. NginX Reverce Proxy. Hakupäivä 12.11.2021. https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/.

8. Linuxide. Redirect HTTP to HTTPS in Nginx. Hakupäivä 24.12.2021. https://linuxize.com/post/redirect-http-to-https-in-nginx/.

9. Jauhiainen, Jukka 2021. Visidon. requirement specification for visidon web demo platform. Sisäinen lähde.

10. Visidon. Image and video quality enhancement. Hakupäivä 20.9.2021. https://www.visidon.fi/technologies/image-and-video-enhancement-algorithms/.

11. El-Samie, Hadhoud, El-Khamy 2012. Image Super-Resolution and Applications. Hakupäivä 29.11.2021. O'Reilly Media. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/image-super-resolution-and/9781466557970/.

12. Milanfar, Peyman 2017. Super-resolution imaging. Google Books. Hakupäivä 20.9.2021. https://books.google.fi/books?hl=fi&lr=&id=fjTUbMnvOkgC&oi=fnd&pg=PP1&dq=super+resolution&ots=56AWDsjMOA&sig=3-NU7-Ey_vzboHRkhgKc0e0fwQ0&redir_esc=y#v=onepage&q=super%20resolution&f=false

13. Ffmpeg. Documentation. Hakupäivä 29.11.2021. http://www.ffmpeg.org/

14. Salo, Immo 2010. Cloud Computing: palvelut verkossa. Jyväskylä: Docendo.

15. Mell, Grance 2011. The National Institute of Standards and Technology. U.S Department of Commerce. Definition of Cloud Computing. Hakupäivä 11.10.2021. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

16. Lisdorf, Anders 2021. Cloud Computing Basics. A Non-Technical introduction. Hakupäivä: 1.10.2021. O'Reilly Media. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/cloud-computing-basics/9781484269213/

17. SmartBear BitBar. 10 Considerations to Choose Between Public or Private Cloud. Hakupäivä 23.11.2021. https://bitbar.com/blog/public-vs-private-cloud-which-one-works-you-the-best/

18. VmWare 2017. Can private cloud be cheaper than public cloud? Hakupäivä 23.11.2021. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vrealize-suite/vmware-paper1-can-private-cloud-be-cheaper-than-public-cloud.pdf

19. Docker docs. Deploying Docker containers on Azure. Hakupäivä: 24.11.2021 https://docs.docker.com/cloud/aci-integration/

20. Docker docs. Deploying Docker containers on ECS. Hakupäivä 24.11.2021 https://docs.docker.com/cloud/ecs-integration/

21. Fastvideo. GPU Image Processing. Hakupäivä 30.11.2021 https://www.fastcompression.com/blog/gpu-vs-cpu-fast-image-processing.htm

22. Docker hub. Nvidia Cuda. 11.21-cudnn8-devel-ubuntu20.4. Hakupäivä 30.11.2021 https://hub.docker.com/layers/nvidia/cuda/11.2.1-cudnn8-devel-ubuntu20.04/images/sha256-fd1afca8e3960642c58db2f2b1694fc8c34625fd13612ad600c62fe833052aa4?context=explore

23. Github. NVIDIA/nvidia-docker. NVIDIA Container Toolkit. Hakupäivä 24.11.2021. https://github.com/NVIDIA/nvidia-docker

24. Frame.io. The Simple Formula to Calculate Video Bitrates. Hakupäivä: 24.11.2021 https://blog.frame.io/2017/03/06/calculate-video-bitrates/

25. Techpowerup. Nvidia GeForce GTX 1050 Ti. Hakupäivä 24.11.2021 https://www.techpowerup.com/gpu-specs/geforce-gtx-1050-ti.c2885.

26. Techpowerup. Nvidia GeForce GTX 1080. Hakupäivä 24.11.2021 https://www.techpowerup.com/gpu-specs/geforce-gtx-1080.c2839/

27. Techpowerup. Nvidia GeForce RTX 2080 Ti. Hakupäivä 24.11.2021 https://www.techpowerup.com/gpu-specs/geforce-rtx-2080.c3224

28. Techpowerup. Nvidia GeForce RTX 3090. Hakupäivä 25.11.2021 https://www.techpowerup.com/gpu-specs/geforce-rtx-3090.c3622

29. Tensorflow. Hakupäivä 30.11.2021 https://www.tensorflow.org/

LIITTEEN NIMI

LIITTEET

Näytönohjainten nopeus suhteessa GTX 1050 TI.liite 1

Teoreettinen laskennan tarvitsema teho. liite 2

Gtx 1050 TI: teoreettinen laskennan aika 100 h/kk videota. liite 3

Näytönohjainten tarvitsema aika ja lukumäärä. liite 4

Pilvipalveluiden hintavertailu. liite 5

Visidon. requirement specification for visidon web demo platform. liite 6

Näytönohjainten nopeus suhteessa GTX 1050 TI

Näytönohjain: GTX 1050 TI 2.138 TFLOPS

Testissä video: koko 75,4 MB, bittinopeus: 29610 kb/s (30 Mbps), Aika 20 s

Videon prosessointiin käytetty aika (Zoom:1, Crop:128, Skip:1): 44 min 35 s = 2675 s

2675 s * 2.138 TFLOPS = 5617,5 TFLOPS

GTX 1080 8.873 TFLOPS:

5617,5 TFLOPS / 8.873 TFLOPS = 633,100 s = 10 min 33 s

633,1 / 2675 = 0,236

(1 – 0,236) * 100= + 76%

GTX 1080 8.873 TFLOPS:

22 173,5 h * 0.25 = 5543,38 h

Näytönohjainten määrä: 5543,38 / 720 = 7,6

RTX 2080 TI 13.45 TFLOPS:

5617,5 TFLOPS / 13.45 TFLOPS = 417,658 = 6 min 78s

417,658 / 2675 = 0,156

(1 – 0,156) * 100 = + 84,4%

RTX 3090 35.58 TFLOPS:

5617,5 TFLOPS / 35.58 TFLOPS = 157,884 = 2 min 38 s

157,884 / 2675 = 0,059

(1 – 0,059) * 100 = + 94,1 %

Teoreettinen laskennan tarvitsema teho

GTX 1050 TI: Videon prosessointiin käytetty aika: 44 min 35 s = 2675 s

Kuukaudessa tunteja: 30 d * 24 h = 720 h

Aika = 100 h = 6000 min

Bittinopeus = 50 Mbps / 8 = 6.25 MB/s * 60 = 375 MB/m

Työn määrä = 5617,5 TFLOPS

375 MB/m * 6000 min = 2 250 000 MB = 2,25 TB

2 250 000 MB / 75,4 MB = 29 840,85

29 840,85 * 2675 s = 79 824 270,6 s = 22 173 h

720 h / 22 173 h = 0,03247193

0,03247193 * 2675 = 86,86 s

5617.5 TFLOPS / 86,86 s = 64,67 TFLOPS

GTX 1050 TI: teoreettinen laskennan tarvittava aika 100 h/kk videota

Näytönohjain: GTX 1050 TI 2.138 TFLOPS:

Testissä video: koko 75,4 MB, Aika 20 s

Videon prosessointiin käytetty aika: 44 min 35 s = 2675 s

100 h/kk videota koko arviointi:

Aika = 100 h = 6000 min

FPS = 60 fps

Bittinopeus = 50 Mbps / 8 = 6.25 MB/s * 60 = 375 MB/m

Kaava (suuntaa antava) tiedoston koolle:

Tiedoston koko = bittinopeus * aika

Kaavan lähde: (https://blog.frame.io/2017/03/06/calculate-video-bitrates/)

375 MB/m * 6000 min = 2 250 000 MB = 2,25 TB

2 250 000 MB / 75,4 MB = 29 840,84

29 841 * 2675 s = 79 824 675 s = 22 173,5 h

Näytönohjainten tarvitsema aika ja lukumäärä

Aika = 22 173,5 h

GTX 1080 = 0.25

RTX 2080 = 0,156

RTX 3090 = 0,059

Kuukaudessa tunteja: 30 d * 24 h = 720 h

GTX 1050 TI

22 173,5 h / 720 = 30,8

GTX 1080 8.873 TFLOPS:

22 173,5 h * 0.25 = 5543,38 h

Näytönohjainten määrä: 5543,38 / 720 = 7,6

RTX 2080 TI 13.45 TFLOPS:

22 173,5 * 0,156 = 3459,066 h

Näytönohjainten määrä: 3459,066 / 720 = 4,8

RTX 3090 35.58 TFLOPS:

22173,5 * 0,059 = 1308,2365 h

Näytönohjainten määrä: 1308,2365 / 720 = 1,82

Pilvipalveluiden hintavertailu

Amazon Aws

Operating system (Linux)
Amazon Elastic Block Storage (EBS) (16 TB)
DT Inbound: Internet (3 TB per month)
DT Outbound: Internet (3 TB <<per month)
DT Intra-Region: (3 TB per month)
Workload (Consistent, Number of instances: 1)
Pricing strategy (On-Demand and Reserved 1/3 Year All upfront)
Snapshot Frequency (Weekly)
Amount changed per snapshot (3 GB)

Model	Type	GPUs	vCPU	Memory (GiB)	GPU Memory (GiB)	GPU P2P	Storage	Dedicated EBS Bandwidth (Gbps)	Network Gbps	Instance Storage GB
Tesla V100	p3.8xlarge	4	32	244	64	NVLink	EBS	7	10	
Tesla M60	g3.16xlarge	4	64	488	32		EBS		25	
NVIDIA T4	g4dn.metal	8	96	384	128		EBS		100	1800
Tesla A100	p4d.24xlarge	8	96	1152						

Microsoft Azure 1v

1 NC6s v3 (6 vCPUs, 112 GB RAM)
Linux
(Pay as you go);
1 managed disk – E4, 100 transaction units;
Inter Region transfer type
30000 GB outbound data transfer from East US to East Asia

Service Type	Region	Monthly	Year
Virtual Machines	East US	2522.650914	30271.8109693
Storage Accounts	East US	899.51	10794.12
		Total	**Total**
		3422.160914	41065.9309693

Microsoft Azure 3v

1 NC6s v3 (6 vCPUs, 112 GB RAM)
Linux
(Pay as you go);
1 managed disk – E4, 100 transaction units;
Inter Region transfer type
30000 GB outbound data transfer from East US to East Asia

Service Type	Region	Monthly	Year
Virtual Machines	East US	1912.00301828217	22944.0362194
Storage Accounts	East US	899.511038289065	10794.1324595
		Total	**Total**
		2811.51405657123	33738.1686789

Google Compute engine

Region Iowa	**Price Month**	**Price Year**
Server use:730 hours/month		
VM class: preemptible		
Intance type: a2-highgpu-4g-preemtible	557.58	6690.96
Operating System / software: free		
GPU dies: 4 NVIDIA Tesla A100	2212.22	26546.64
Local SSD: 4x375 Gib	61.97	743.64
Persistent Disk 16 GiB	563	6756
	3394.77	**40737.24**

Price comparison:

	Month	1 Year plan	
Google Compute engine	3394.77	40737.24	e
Aws NVIDIA T4	3449.97666667	41399.72	e
Microsoft Azure	3422.1619524	41065.94343	e
Aws Tesla V100	5850.8025	70209.63	e
Aws Tesla A100	14735.0175	176820.21	e

	Month	3 year plan	
Aws NVIDIA T4	1918.33277778	69059.98	e
Microsoft Azure	2811.51405657	101214.506	e
Google Compute engine	3394.77	122211.72	e
Aws Tesla V100	4247.32416667	152903.67	e
Aws Tesla A100	8874.93527778	319497.67	e

VISIDON CLOUDDEMO SPECIFICATION

Requirement specification for Visidon Web Demo platform

Table of Contents

1 INTRODUCTION

This document describes system and software requirements for a Visidon Web Demo platform.

1.1 Version history

Date, version	Author(s)	Changes
2021-03-19, 0.9.0	JukkaH	Initial proposal
2021-03-22, 0.9.1	JukkaH	Clarifications and minor error corrections
2021-03-24, 1.0.0	JukkaH, VaidaJ, MikaH	Finalizing requirements

1.2 Purpose

Visidon Web Demo Platform is a new service that allows Visidon to demonstrate its image and video analysis capabilities for selected customers. Interface should be easy to use for the customer as well as provide visual results for review and compare within web interface as well as available to download.

1.3 Service description

Demo platform consists of an administrative web interface that is used to manage service accounts and an interface for customers. Both interfaces are password protected.

Account and other related information, like metadata of processed videos is stored in a database.

It consists of an administrative web interface that is used to manage accounts that is internal to Visidon employees. Web interface for customers allows them to run and view the results of the algorithms on the data they upload themselves to the service.

2 GLOSSARY

Administrator: User with special privileges on the system.

Back-end (web): Software that is run inside web server as a result of http(s) requests.

Client: Business or organization that shares work items and quotas to the service. A collection of user accounts.

CRUD: Create, Read, Update and Delete (CRUD) are basic editing functions.

Database server: Typically a SQL database that maintains structured information in permanent storage.

Front-end (web): Component of the service that user interacts with. Software and content that exists in browser. Typically consisting of JavaScript and HTML.

GDPR: The General Data Protection Regulation (EU) 2016/679 (GDPR) is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA).

Input file: File needed to process a work item. Input file is stored in Shared Storage.

Processing node: Computer system that is capable of running the analysis software.

Shared Storage: A service that provides object storage where files can be stored and retrieved from using identifiers like file names. This could be implemented as a shared network file system or some software-defined storage solution.

User Account: Information about each user like name, email address, password hash, last login, last EULA acceptance and any session management information.

Web service: Service accessible through web browsers such as Firefox or Chrome.

Work item: Data packet that describes what kind of analysis should be done in processing node and with what data. Each work item is tied to a particular customer identity.

Work item status: Status indicating that it is currently waiting in queue, processed with X% complete, canceled, finished with success or finished with an error.

XSS: Shorthand for Cross-Site Scripting. A vulnerability where other users are exposed into content injected into the web page that is controllable by the attacker. This enables running malicious JavaScript commands on other user's browser.

3 SYSTEM REQUIREMENTS

This chapter lists the requirements for the final system when it is used to serve customers.

System consists of the following components:

- (Optional) reverse-proxy server directing traffic to web server
- Web server providing user interface (web application) for the customer
- Web server back-end API for supporting user interface
- Web server back-end API for supporting processing nodes
- File storage accessible from API back-end and computing node(s)
- Processing node(s) that process the input data and produce the results.

See Drawing 1: Service architecture component diagram for a high-level illustration of the architecture.

Drawing 1: Service architecture component diagram

3.1 Generic system requirements

Generic requirements not associated to any particular component of the system.

3.1.1 SYSREQ1: GDPR -compatible

Note: Person writing this is not a lawyer – interpretation of the law may be incorrect

Generic Data Protection Regulation (EU) 2016/679 (GRPD) is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA).

It covers things like transfer of personal data outside of EU and EEA areas.

According to GDPR, this regulation affects any enterprise processing personal data of individuals regardless of citizenship or residence.

Controllers and processors of personal data must put in place appropriate technical and organizational measures to implement the data protection principles. No personal data may be processed unless this processing is done under one of the six lawful bases specified by the regulation (concent, contract, public task, vital interest, legimate interest or legal requirement).

The basis for us will be concent for any personal data associated with Client and user accounts. Accounts will only be made for people who want to try out this system. They can also remove their account data at their will or it will be removed automatically (some time) after valid test time for Client runs out.

This personal data must only be available to the employees of the company in charge of maintaining this system and handling customer services here and similar valid uses.

User account information and any customer data must be destroyed after we have no valid information to hold hit.

Data Protection Officer (DPO) must be named that customers can contact in case of any GDPR requests. This information should be readily available on the web page as well as what information is stored (“rekisteriseloste” in Finnish).

Additionally Businesses must report data breaches to national supervisory authorities within 72 hours if they have an adverse effect on user privacy.

We also act as a processor of data (in addition to being a controller). This happens when the data we accept from the user accounts itself contains personal information (i.e. images or videos of people). In this case, account owner itself is a controller and we process the data on their behalf. In this case, it is our responsibility that the personal information we process is not visible to other parties and the privacy of the people in those videos and images are respected.

There may be other requirements not listed here but that would be the gist of it.

3.1.2 SYSOPT1: (Optional) Prefer dockerized services

Docker should be used for containment of different parts of the services as much as possible to make upgrading and maintaining a simpler process.

3.2 Web service

Web service contains all components that are run through Apache or other web servers.

3.2.1 WSSYSREQ1: HTTPS encryption available

When service is accessed from the internet, HTTPS encryption is used for all requests.

3.3 Shared storage

Shared storage is used for storing input data from users and generated outputs from the service.

3.3.1 SSSYSREQ1: No direct access from Internet

All access to shared storage must be authenticated. Users of the system can access only the parts of the storage that are relevant to their account and permissions.

3.3.2 SSSYSREQ2: Minimum disk space 16 TB

To store both input data as well as generated results.

Incoming data files can be either images or videos. In addition to the input data, we need to store the result images/videos and perhaps additional comparison visualization data for each work item that is valid in the system.

Image sizes are in scale of 100kB – 10 MB. Video sizes can range from 100 MB – 10 GB and should be capped to some sane upper limit (for example 2-5 GB input size).

We estimate that there are max 10 concurrent customers testing the demo site at any given time.

We estimate that generated outputs can be as large as 4x of the inputs if we generate comparison videos with high bit rate. 1 TB per customer should probably be enough to store several larger test files and their results. With 10 customers, that is a total of **10 TB**.

We will also need to have some test accounts. Reserve **3 TB** for these.

As per-user disk space should be implemented with soft quota checking, there should be extra margin available so that we can provide the results even when the total disk space goes over the limit. 20% margin should probably be enough for that meaning extra **2 TB** per customer..

3.3.3 SSSYSOPT1: Support direct file download, upload and delete only

Nothing in this specification should require that there is a file listing functionality available in the Shared Storage system and the naming conventions can generate filenames that are not easy to guess.

This would make it more secure against bulk transfers of data if an attacker gains access to a Processing Node that needs to access these files. For work items, it is possible to specifically list the files that are needed for processing and Processing Node can in turn generate a list of files it created during the processing for web interface.

This would allow downloading of all customer data only if database access is compromised as it contains all information about relevant files and other customer data that exists in the system.

Deletion is required by any cleanup processing.

3.4 Database service

Database will hold information about user accounts, work items available for each customer and any other needed information for the service.

3.4.1 DBSYSREQ1: Some SQL database accessible only from web back-end

SQL database (for example PostgreSQL v13) instance is configured so that web system has access to it.

Additionally there can be intermediate barriers between web back-end and database itself if they are deemed necessary for security purposes.

3.4.2 DBSYSREQ2: Generation of initial database state on install

When the system is first installed, it needs to generate the initial database state that is otherwise empty but has a valid Administrator-level account so that other clients and customers can be added using appropriate tools.

3.4.3 DBSYSREQ3: Support for schema updates for maintenance

Back-end may require some database schema updates when versions are updated: Adding new tables, views, columns, indices, modifying and removing them.

3.4.4 DBSYSREQ4: Sensitive information should be hashed

Passwords and other sensitive information should NOT be available as plaintext in the database.

With passwords, a secure way is to compare hashes of the password (or recursive hashes to make password cracking slower) or do challenge-response checks with the hash instead of comparing them directly. The latter strategy would indicate that plaintext password (or a string derived from it) is provided for the server only at password change and later checks only verify that both parties know that same hash. These strategies significantly reduce the likelihood that a database leak would also leak the password information in an easy-to-use format.

4 SOFTWARE REQUIREMENTS

Software component requirements that need implementation.

4.1 Web service front-end

Web service front-end consists of HTML and JavaScript interface that is run on client browser.

4.1.1 WSFEREQ1: User authentication

Any user sessions should be authenticated against the database. If user is not authenticated, he only has access to the authentication services of the Web service UI back-end.

4.1.2 WSFEREQ2: Maintenance mode support

In maintenance mode, only system administrators should be able to log in. Others will receive a configurable message with information about maintenance break.

4.1.3 WSFEREQ3: Terms of Service support

When user first logins the system he must accepts the terms of service.

Initial consensus was that we do not want to claim any ownership to the data but we may need to tell users in Terms of Service that if they have personally identifiable data in those files, they act as a controllers in GDPR terminology for that data.

Some lawyer will probably need to write the actual text.

4.1.4 WSFEREQ4: CRUD operations for Clients (Administrators only)

Administrators need to be able to manage status of Clients: Name, current quota limits and used resources, validity times etc.

If Client is removed through this interface, is is marked for removal. A separate background process should do the actual removal work as it needs to clean up external results like file storage as well.

Marking Client to be removed should also cancel any work items in waiting state.

4.1.5 WSFEREQ5: CRUD operations for User Accounts (Administrators only)

Administrators need to be able to manage status of user accounts. Each user is part of some Client.

When a new user account is created, it is associated immediately with the selected Client. Basic information like name and email address are filled by the administrator. The service should then provide a Relatively Secure™ method for the account owner to start using that account. A typical way is to send a secure link to the email address with informative message indicating what this is about. When user clicks that link, he enters to a password setting page. After setting passwords, user is considered to have logged in for the first time and link becomes nonfunctional.

Password reset functionality should work the same way: A link is sent to the address of the email owners that allows one-time password reset for some time period.

4.1.6 WSFEREQ6: Option for Administrators to view other Client's work items

Normally all work items and their listings should be visible only for the user accounts of the Client that created them. This exception is needed for customer support.

4.1.7 WSFEREQ7: Read, Update and Delete options for User Accounts

Non-Administrator level users should be able to view their own account information as well as update parts of it. At least password changes should be possible. Email and name are considered to be protected information that user can't change by themselves.

Users must have an option to delete their own account. If that account is the last account of the Client, the Client will be marked for removal as well.

4.1.8 WSFEREQ8: Listing of work items

Site will provide a listing of work items indicating their information and current processing status (queued, processing with X% complete, canceled, finished with success, finished with error).

If work item is queued or processing, it can be canceled. If it is in any of the finished states, it can be marked for removal. Work items marked for removal should be visualized as such so that users will know that any files in that will be released soon.

List allows users to pick a work item to view results from. This option is available only for work items in *finished with success*-state. This option is not available for work items that are marked for removal.

4.1.9 WSFEREQ9: Support creation of new work items

Each work item will have its own user interface that user has to select before anything else. Initial consensus was that we do not need a support for limiting, what work item types are available per customer but they can try anything they want to.

If work item is about processing a video, that video has to be first uploaded to the Shared Storage. If it is about analyzing images, relevant images and possibly other data files must be first transferred to the Shared Storage.

Interface should make it explicit, what properties are required from the files like maximum size, file format and supported codecs.

Actual work item is made by referencing uploaded files in the Shared Storage and setting any other relevant work item type-specific parameters.

Work item can't be created before relevant files are fully uploaded. This condition should be indicated to the user.

Alternatively interface can support creation of new work item of the same type by using the same inputs as some existing work item of the same type with finished-with-success status. In this case, interface will auto-fill input file information and only processing parameters need to be selected.

4.1.10 WSFEREQ10: Support chunked file uploading into Shared Storage

When user wants to upload new input files, they have to be transferred into Shared Storage first. This upload interface should be a component of new work item creation forms. Additional metadata like filename should be stored as well.

Chunked upload will allocate a new file (with a know size) and will get a unique handle to upload chunks into. At allocation, Client quota information is updated to match the final file size needed. If quota would get over while transferring a new file, file creation must fail and this must be reported to the user.

Chunk size should be returned by the web back-end with the creation handle. Limit should be something like 1MB. Sending larger chunks than the server indicated will fail.

File upload status should be indicated with a progress meter and possibly time estimate of completion.

If a temporary error occurs during upload (route error, host not found etc.), upload should resume with an exponential backoff up to 30 seconds. For error status indicating a permanent error (quota limit exceeded, permission errors etc.) upload process must stop.

4.1.11 WSFEREQ11: View for finished work item

When work has been finished successfully and it is not marked for removal and permissions match, this view is available.

Visualization depends on the actual work item type but in general should consist of downloadable hiqh-quality results, comparison visualizations and visualizations viewable from the web.

View should also show the parameters and information about input files used for processing.

This view should also be a suitable place for an operation that creates another work item using selected item's inputs. Some kind of "Process again with different parameters" operation.

Note that work items may have been generated by a Processing Node that is an older version that may produce somewhat incompatible outputs to new versions. View system should be made compatible between multiple versions that we expect inputs from. If versioning is needed for this, versioning should be made per work item type that directly reflects the compatibility of the output.

4.1.12 WSFEREQ12: Data from the user accounts should never be embedded into the view as is

The only exception is text that is properly escaped to avoid any XSS issues.

4.1.13 WSFEREQ13: Visual presentation suitable for a product

Visual outlook of the service should look polished and give impression that the user is working with a high-quality product.

4.1.14 WSFEOPT1: (Optional) Other administration views

It may be interesting to see a status of the computing nodes or other useful views to verify the health of the system.

4.2 Web service UI back-end

Web service back-end consists of components that are run on web server itself.

4.2.1 WSUIBEREQ1: Implement functionalities needed by UI front-end

User interface front-end needs many different queries from authentication to file upload procedures to file downloads and reports. Back-end code should support the generation of these views.

4.2.2 WSUIBEREQ2: Verify authentication and permissions for all requests

All operations except login-related views and requests require that user has logged in. Depending on the Client status of the user account, he has access to only files created for that Client account.

Nothing in the request should be trusted blindly. Validity of all parameters should be always verified by the back-end independently. For example, request to create a new work item requires that there is a session indicating a logged in user. Request must not be able to select a Client to associate the work item with but it is selected by the back-end based on the Client user belongs to. All input files must be associated with the Client of the new request etc.

4.2.3 WSUIBEREQ3: Require GET requests for views and POST requests for changes

HTTP GET requests should be used when operation is just requesting a view to something. HTTP POST requests are reserved for changing states.

4.2.4 WSUIBEREQ4: Associate expected Processing Node version with work items

When work item was generated, it was intended to be processed by a Processing Node available at that time. New requests may only be compatible with newer versions of the algorithm and this way it is possible to direct old work items to Processing Nodes still running older versions of the software and new items to newer nodes.

4.2.5 WSUIBEREQ5: Require a special server-generated parameter in POST requests

HTTP POST requests should be protected by a hidden form element value indicating that it was generated for the current session that posting it. This is to protect users (especially administrators) from clicking somewhere else in the net that sends a request to this service and modifies something there. In the worst case, a button like that could create a new admin account for the attacker as the request would automatically fill out any session information stored typically in the cookies.

4.2.6 WSUIBEOPT1: Operations can be implement as a REST API

Implementation is allowed to implement this as a REST API that JavaScript interface just calls through the browser if that makes more sense for the user. This would allow us to use the service using proper API keys from any external system.

However, as this changes the authentication method, any relevant security issues regarding API key management etc. must be solved.

4.3 Web service Scheduled Events

Some operations can take a long time. Web interface operations should be limited to ones that can be reasonably finished in seconds.

Operations like cleaning up terabyte(s) of files from Shared Storage require longer processing times and should be run separately from Web Server requests that have timers stopping the operation after too many seconds of processing.

4.3.1 WSSEREQ1: Client removal process

If Client is enough (some documented time value) past its valid demo time, it must be marked for removal. This is a GDPR requirement, as we have no valid reason to hold onto this information any more. Information is held only two weeks after Client access to the service has expired. During that time, marketing can contact the people and discuss about extending the time.

If a Client is marked for removal, any files in storage for that client should be removed. Additionally any users and work items from the database associated with that client will be removed.

Removal should be delayed until all work items of the Client are in a finished, error or canceled state to avoid race conditions with the processing nodes.

This operation could be done for example every 24 hours.

4.3.2 WSSEREQ2: Work item removal process

If a work item is marked for removal, all files associated with it will be removed from the Shared Storage after which database is updated.

If work item is the last user of some input file, those input files should be marked for removal as well.

This requires much higher frequency that Client removal process as it affects how quickly extra space is made available for the Clients. Delay between these operations should be at maximum one minute.

4.3.3 WSSEREQ3: Input file removal process

A special work item removal process should be run separately from web request schedules (for example through cron services) so that any web server timeouts do not affect it.

If input file is marked for removal (work-items using it are being removed) or it has no using work-items and upload is at least 24 hours old (interrupted upload), it should be removed and any space allocated for it should be returned back to the Client it is associated with.

This requires much higher frequency that Client removal process as it affects how quickly extra space is made available for the Clients. Delay between these operations should be at maximum one minute.

4.4 Web service Processing Node API

Processing node API is responsible for distributing queued work items for Processing Nodes as well as receiving status reports and other signals from them.

4.4.1 WSPNREQ1: API should allow multiple concurrent Processing Node versions

There are two separate issues for versions here: Work item compatibility and communication path compatibility. We need a separate API entry point when Processing Node wants to communicate with the API in a different way. Another issue is that there can be either same or different versions of algorithms for processing work items between Processing Node versions.

Alternative design approach would be that there would be different processing nodes for different work-unit types and they would only request work for those types (and versions). This would make it more difficult to make computations with minimal resource use but would have benefits of having different queues for faster and slower operations.

In order to create an upgrade path, Processing Nodes should call a versioned API (indicated by URL path component). This way, APIs can be updated and any missing information correctly handled by older version of the API when multiple versions of the Processing Node are running at the same time.

Alternatively, if we can always wait for any existing works to finish (even in maintenance mode on), then no Processing Nodes with older versions are never present after upgrade procedure making this less of an issue.

4.4.2 WSPNREQ2: Processing Nodes must be authenticated

All processing node requests must be authenticated that they do not come from external systems.

4.4.3 WSPNREQ3: Processing Node heartbeat

Processing node requests act as their heartbeat as well. Information about last contact should be updated.

4.4.4 WSPNREQ4: Request for new work item

Processing Nodes can request for new work items when they have finished their old work.

If Processing Node starts requesting new work and there is an old work currently being processed, that should be put back to the queue as it was obviously not finished for some reason.

Processing nodes are only given work items that match the exact Processing Node version associated with the work item.

If there is queued work that Processing Node can handle, that work item is reserved for the requesting Processing Node atomically. If two nodes are requesting for work simultaneously, only one of them can get each item.

Work that is reserved by a node with too long heartbeat timeout is also considered to be queued instead.

As a response, a data packet containing information about work item type, processing parameters is returned if some new work is reserved for the requester. Otherwise, a reply indicating that no work is available is provided.

Return message could also indicate extra information like next time a status update should be sent or next item to be queried to make these times centrally configurable without Processing Node updates. These status updates also act as a heartbeat for the Processing Nodes so that it is still working.

4.4.5 WSPNREQ5: Status report

Status report messages provide information about the current status of the processing. They should also contain a percentage value as a floating point value to indicate, how far the processing has gone.

4.4.6 WSPNREQ6: Reporting finished work item

If work has been successfully finished, all information about produced output files should also be included in the message.

If an error occurred or work had to be stopped for some other reason (for example, work cancelled), this status should be reported instead of final results.

4.4.7 WSPNOPT1: (Optional) Fair queuing between clients

Web back-end should implement fair queuing procedures per Client basis if possible when distributing work items to Processing Nodes. With limited computing resources, all clients should process work items evenly if one has queued hundreds of them before others.

4.5 Processing node

Processing node is any node that is capable of performing actual analysis for the work items that are created into the system.

4.5.1 PNREQ1: Requests work items periodically

Each computing node should periodically request queued work items from the system when it is not working on an item by itself.

A request includes necessary versioning information so that some requests for work can be ignored. This is necessary for doing rolling update for Computing Nodes where old versions might not be able to handle some new work items generated by the service. On the other hand, if service contains unfulfilled requests intended for the older version of the back-end with perhaps incompatible parameters, that exact version should be used to process these work items.

4.5.2 PNREQ2: Process work items

If a request for work indicated a success, then it should prepare by downloading the input files associated with the work items from Shared Storage, run the algorithm with the selected parameters and finally upload the results to Shared Storage in work-item identifier-specific directory.

When upload of the results is complete, Processing Node must signal through API the necessary information of the results.

It should then immediately request for new work after the previous work is finished without any additional delays.

4.5.3 PNREQ2: Sending status reports

While the processing is running, Processing Node should send out status reports periodically through the Processing Node API indicating current progress.