Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

A design in system architecture based on mobile cloud computing for a virtual try-on solution

Duong Van Ngoc, Nguyen Tien Dat *

Modeling and Simulation, Viettel High Technology Industries Corporation, Hanoi, Vietnam

ARTICLE INFO

Article history:
Received 20 December 2021
Received in revised form
24 March 2022
Accepted 25 March 2022

Keywords:
Cloud computing
Amazon web service
Virtual try-on

ABSTRACT

Cloud computing is an emerging technology in this digital century. It provides an excellent but low-cost source for database storage, computing power, applications, and services through an internet-delivered cloud platform. Thanks to the cost savings in investing and maintaining physical data centers, as well as the stability of Quality of Service (QoS), it has no restrictions on company size or sector for enterprises to shift their operations to a cloud platform. The fashion industry, particularly the fashion e-commerce sector, is a case study in leveraging the cloud platform via a technology called "Virtual try-on" (VTO). VTO solution allows fashion brands to increase the shopping experience, however, requires installing and maintaining a bulky system for implementation. There are different methods and approaches to design architectures using cloud computing, however, there have not been many studies addressing tasks related to machine learning that uses the high Graphics Processing Unit (GPU) encountered in VTO solutions. To design a scheduler that could optimize the system performance while lowering operational expenses in VTO solutions, this research proposes a system to (1) handle synchronous model and asynchronous model separately and clearly, (2) perform multi-layered task processing architecture by hashing task ID and implementing a queue management system. This method would satisfy three major requirements: (1) Avoid complex hardware requirements for users, (2) Ensure the system stability and the ease of horizontal and vertical extension, and (3) Protect user information privacy.

1. Introduction

This research presents a solution for building and designing an architecture for use in the fashion industry. Due to the severe impact of Covid-19 and the trend in digital transformation, E-commerce, particularly Fashion E-commerce, is exploding more than ever. This rapid development has had a profound impact on global fashion business models. According to Statista (2020), the Fashion E-commerce Industry is the largest E-commerce market segment with a global market value estimated at $752,5 billion in 2020 and a projected increase to $1164,7 billion by the end of 2025 (Statista, 2020). Despite consistent sales growth in recent years, online retailers have experienced a frustrating product return rate, which is an unavoidable consequence of online shopping's disadvantage: customers do not have a trial room as in brick-and-mortar stores, and they are unable to try products on their bodies. Due to the lack of physical interaction, the size mismatch is the biggest reason for clothes returns. One of the best solutions for solving this issue is Virtual Try-On (VTO). This technology allows consumers to try clothing on a 3D model that simulates their body shape and determine whether a product suits them well. Not only does VTO resolve the misfit concerns but it also provides a joyful and personalized shopping experience, thereby increasing customer satisfaction and sellers' competitiveness. VTO can be employed in immersive and non-immersive technologies for a range of applications such as Virtual Fitting Room and Online Virtual Try-on Tool. Commercializing a VTO solution, however, could be a significant challenge because it especially necessitates the utilization of technological devices, computing power and resources. Managing a large database of 3D assets, equipping expensive hardware and

* Corresponding Author.
Email Address: datnt65@viettel.com.vn (N. T. Dat)
https://doi.org/10.21833/ijaas.2022.06.005
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-7537-8708

protecting user information are barriers that system operators face.

To address the aforementioned issues, a mobile application for both Android and IOS operating systems has been developed and integrated with the technologies in 3D human body reconstruction, physics simulation and 3D rendering. These technologies are packed under containers and then uploaded to a cloud platform. Each container on the cloud would contain the application, libraries and dependencies, then combined with the local server to store the users' database. Businesses are using cloud computing platforms for a variety of objectives. Netflix provided a video streaming service on a global scale with over 209 million subscribers in 2021, known as "Insider intelligence." The architecture of Netflix includes three main parts:

1. Client: Providing a front-end as browsers on laptops, desktops or apps on mobile, and television to send requests and receive responses.
2. Backend, where services, databases and storage are immigrated and operated on Amazon Web Services (AWS).
3. Open Connect, used for optimizing storing and streaming directly videos to users.

Another cloud printing solution is proposed and implemented by Xeros. It allows users to access the printer from any location with unlimited authentication: NFC, QR code or Pin Code, All activities such as managing, updating and patching are operated and controlled by Xeros's cloud. Several big companies in designing and manufacturing such as Altium (with PCB and mechanical design) and Autodesk (with mechanical design, BIM) also presented full services on a cloud from sketch, modeling, and simulation to centralized cloud storage. The main contributions of this research are:

1. Proposing a complete architecture system applied for virtual try-on solutions that combine mobile agent-based cloud with a local server.
2. Providing case studies to demonstrate the performance and stability of QoS of the proposed system.

This research is organized as follows: Section 2 gives a brief description of related works. The proposed architecture is presented in Section 3. Section 4 provides and analyzes research results when deploying the proposed system on a mobile application. Finally, conclusions and future works are mentioned in section 5.

2. Literature survey

Victor's research in Ribes et al. (2020) proposed an architecture solution using mobile cloud computing in 3D designing and simulation for the traditional manufacturing industry. 3D databases such as designs and drawings are uploaded to the cloud from client computers through Transmission Control Protocol (TCP)/Internet Protocol (IP). Applications and software known as Computer-Aided Design (CAD)/Computer-Aided Manufacturing (CAM) are installed on cloud computing to run simulation and modeling in manufacturing processes. Furthermore, this research combines cloud offloading with GPU processing to improve system performance and efficiency. Benedetto et al. (2018) presented a code offloading framework named "MobiCOP." His key contribution is a new paradigm for implementing code offloading that contains a library, which is compatible with most Android devices available at that time. An architecture system connects clients with the server through the MobiCOP library contains android services and middleware by TUS protocol, an open-source of HTTP. Accordingly, it requires developers to implement and code in a thread-safe manner to ensure system synchronization and security. Zhang et al. (2012) developed an algorithm regarding code partition for mobile code offloading. The main challenges are real-time adaptability and code partition precision. A call graph model is defined to describe the calling relationship of the application. Besides, to locate the best offloading and integrating points, a depth-first search and a linear time search are applied. Biswas and Whaiduzzaman (2018) and Fernando et al. (2013) proposed an overview of different offloading techniques in the wireless heterogeneous network. They gave definitions of mobile cloud computing, provided several benefits of offloading and brought some suggestions for applying the MCC model in the future mobile industry. Kemp et al. (2010) presented a "cuckoo" computation offload framework for Android smartphones. It includes a runtime system that boosts computational speed or reduces energy consumption, as well as a resource manager application that integrates with the Eclipse build system. Alsboui et al. (2019) introduced an emerging IoT platform called "IOTA." Their main distribution is creating a new scalable and energy-efficient distribution for IoT devices. The core part of IOTA is the tangle, which is defined as a concept of a Directed Acyclic Graph. Meanwhile, cloud-based computer numerical control (C-CNC) (Lu et al., 2019) is proposed as a solution for manufacturing machines as a service. CNC functionalities are configured in local devices such as laptops or desktops via interface and then transmitted to the Google cloud platform. Liu et al. (2019) suggested a method for enabling high accuracy object detection in an AR/VR system running at 60 fps. The main idea is to employ low latency offloading techniques, then decouple the rendering pipeline and apply a fast object tracking method to maintain detection accuracy.

3. Methodology

The design goal of the proposed architecture for implementing the Virtual Try-On solution is divided into three main parts:

1. Maintaining the system's high stability and reliability so that it can handle thousands of application requests at the same time.
2. Ensuring scalability, which allows the system to scale both in the number of clients (horizontal) and the requirements of hardware specifications (vertical) without affecting customers' experiences.
3. Keeping the customer data secure.

3.1. Overview

Fig. 1 indicates the detailed architecture for the Virtual try-on system. It includes three main parts:

1. Client
2. Public cloud
3. Private cloud

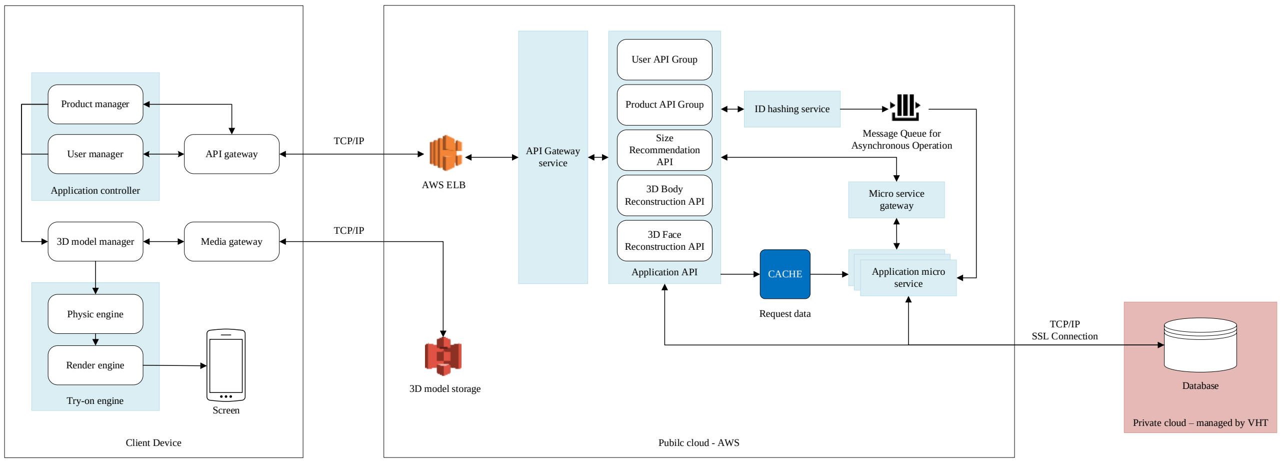

Fig. 1: The linear relationship between weight and volume

Client devices are smartphones, tablets, or other hardware equipment that are used to receive requests and provide services to customers. We developed an IOS and Android application (Virtual Try-on solution) to interact with customers by controlling the request through SDK and API services. The public cloud consists of services and several storages that are deployed and run entirely on Amazon Web Services Cloud. The private cloud is an infrastructure of a local server that is built and operated by Viettel Military Industry and Telecoms Group (Vietnam). The public cloud will package data and send them to the private cloud by using a Secure Socket Layer (SSL). The private cloud is responsible for archiving databases to protect customers' information security. Each component will be described in detail below.

3.1.1. Client

There are five components in the "client device" module:

1. API gateway
2. 3D model manager
3. Media gateway
4. Application controller
5. Try-on engine

The API gateway would handle and manage the requests to the Amazon server

The authentication token header is created and linked to the Application Programming Interface (API) of the user manager such as saving the user name or changing the password. On the other hand, the authentication token is used to create the path and connect to the cloud via Hypertext Transfer Protocol (HTTP), as well as classify the API into two types: public or authentication. By utilizing the API gateway, the latency of API requests could be controlled to a bare minimum; user behaviors are tracked and monitored by API calls as well. Furthermore, it is flexible and straightforward to scale up the system or add more APIs with no impact on clients. Similar to the API gateway, the Media gateway creates a bridge connecting clients and the cloud; however, it focuses on 3D model storage which is put on Amazon Simple Storage Service (S3). 3D storage includes images of clothing products and their 3D models in the FBX file format (which is a 3D object storage). The Media gateway saves the Product Uniform Resource Locator (Product URL) of each product in brands and saves the structure system of media files. Like the API gateway, it is in charge of adding authentication tokens into the headers of requests that require access to users' libraries. By applying the Media gateway, it is easy to replace or expand the system's memory zone without affecting the customer experience. Meanwhile, the Application Controller Module controls the whole actions of clients and connects to the AWS via the API gateway. It comprises two sections:

1. Product Manager: Used to manage information about clothing products as types, colors, sizes, and 3D cloth model indexes.
2. User Manager: used to manage information about users such as personal information, passwords, 3D human model indexes, purchase history, and selected items.

The next module would manage 3D human models and 3D cloth models, in which, clothes are classified into three types: Upper body (Shirt, T-shirt, Casual Jacket, Sleeveless top); Lower body (Pants, Shorts, Skirt), Entire Body (Dress, Jumpsuit). The 3D model manager receives the indexes of the 3D cloth model or 3D human model from the Application Controller Module and then sends the request to the Media gateway to collect the 3D model's meshes and textures. The 3D models are displayed on the screen using a specific feature that automatically loads/unloads to release the computer memory. The final is the Try-on module which includes a Physics engine and a Render engine. The Physics engine takes meshes of the cloth and human body as input and provides a solver that calculates the interaction between the cloth and human body in both with motion and without motion. Besides, the Render engine receives the command of the physics engine to perform and render graphics on hardware devices. It includes shading, material, texture mapping, and special effects like fog or light reflection. Both engines are run parallel on the device GPU to ensure the application's performance.

3.1.2. Public cloud

This part outlines the system's structure deployed on the Amazon cloud. Elastic load balancing of AWS is used to arrange the input requests from client devices following a pre-configured distribution, then send them to the API Gateway. Using load balancing ensures the capacity to scale up and boosts the system's performance. Similar to the API gateway, which is deployed on the mobile application, the API gateway on the cloud serves two purposes:

1. Decoding the request from the balancer.
2. Encoding the data from API applications.

The decoding process would check the validation of requests via two steps:

1. General checking in content type and content length.
2. Specific checking in authentication token regard to human API such as Validation of JWT token, validation of JWT signature, validation of JWT exp time, validation of JWT ISS.

On the other hand, the API Gateway receives data from the API application and adds the headers to them if necessary. The next part is Application API which is configured and set up to expand horizontally. Requests received from the API gateway would pre-process like unpack/pack data, convert the data type to ensure the input's correction, and send them to micro-services. Due to the characteristics of each API service, transferring data is executed in two methods: Synchronous and Asynchronous, along with the process of ID hashing service, message queue, microservices gateway, and application service would be described in the following parts.

3.2. Asynchronous OS schedule

Fig. 2 indicates the detail in the Asynchronous Outsourcing schedule. The key factors are 2 APIs of body reconstruction and face reconstruction which require specific hardware to deploy. Here, APIs are contained and built on EC2 g4dn.xlarge of Amazon. As mentioned above, after receiving the request from the API gateway, the data would be pre-processed in the Application API and divided into two parts:

- Images data: Saved in the cache.
- Others under the text/string format: Transferred to a module named "ID Hashing Service (IDHS)," whose major objective is to arrange, classify and select the clients' request ID.

IDHS uses the hashing algorithm to map request ID (32 bytes) to queue ID (2 bits). Each queue has the corresponding node ID, APIs get the requests from the queue including request ID and data that would not be saved in the cache, and collect the data in the cache with the corresponding request ID. After that, 3D Fault Face/Pose Detection (3DFP) checks the input images to ensure the quality of microservices. If satisfied, APIs process the requests and transfer the results to the database. By adding IDHS, a general service for storage to map ID with queue ID would be diminished, reducing the complexity of the system and ensuring system stability. Besides, using a random ID with length-32 bytes (32x8 bits) makes the identical statistic ID becomes approximately zero.

3.3. Synchronous OS schedule

Fig. 3 describes the structure of the synchronous outsourcing schedule with regard to APIs such as Login service, Sign up service, Size recommendation service, and Product service APIs. These APIs are deployed on the Amazon cloud without any specific hardware requirements.

Similar to the Asynchronous OS, Application API receives input requests from the API gateway, managing and distributing them to microservices by a module called "microservices gateway." The agents regarding the client in the Application API include: signing in/signing up, managing 3D models and products, and managing passwords. Different from the Asynchronous Os schedule, the microservices gateway is created to receive request data and distribute them to microservices. Its primary goal is to manage the lists and addresses of microservices, even if services change the IP address or add-on, the microservices gateway would monitor and update correspondingly without affecting the Application API. Because the configuration now is completed in the gateway, it is easy to maintain and upgrade the whole system in the future.

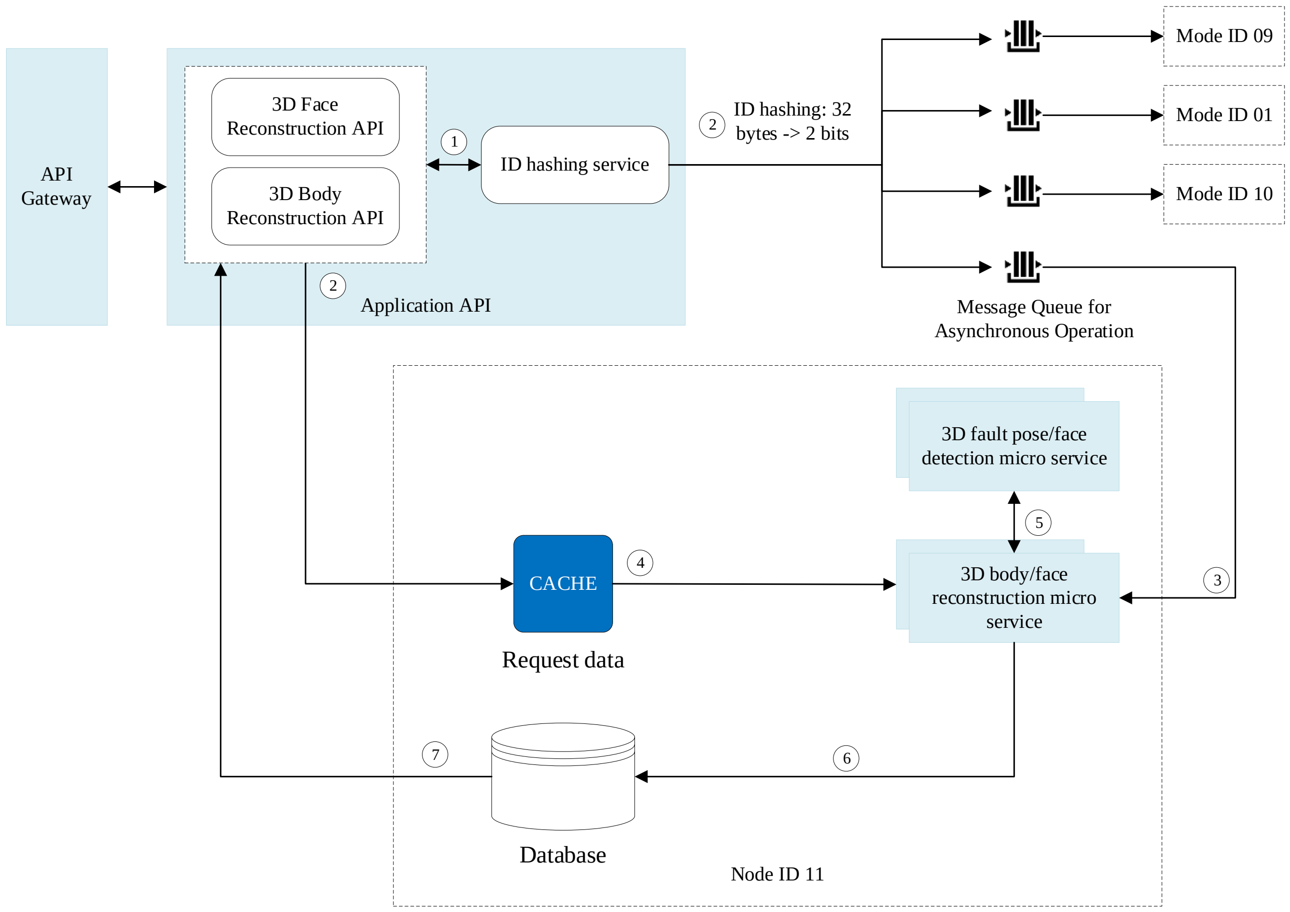
API Gateway
3D Face Reconstruction API
3D Body Reconstruction API
Application API
1
ID hashing service
2
ID hashing: 32 bytes -> 2 bits
Mode ID 09
Mode ID 01
Mode ID 10
Message Queue for Asynchronous Operation
2
3D fault pose/face detection micro service
5
4
CACHE
Request data
3D body/face reconstruction micro service
3
7
6
Database
Node ID 11

Fig. 2: Asynchronous OS schedule

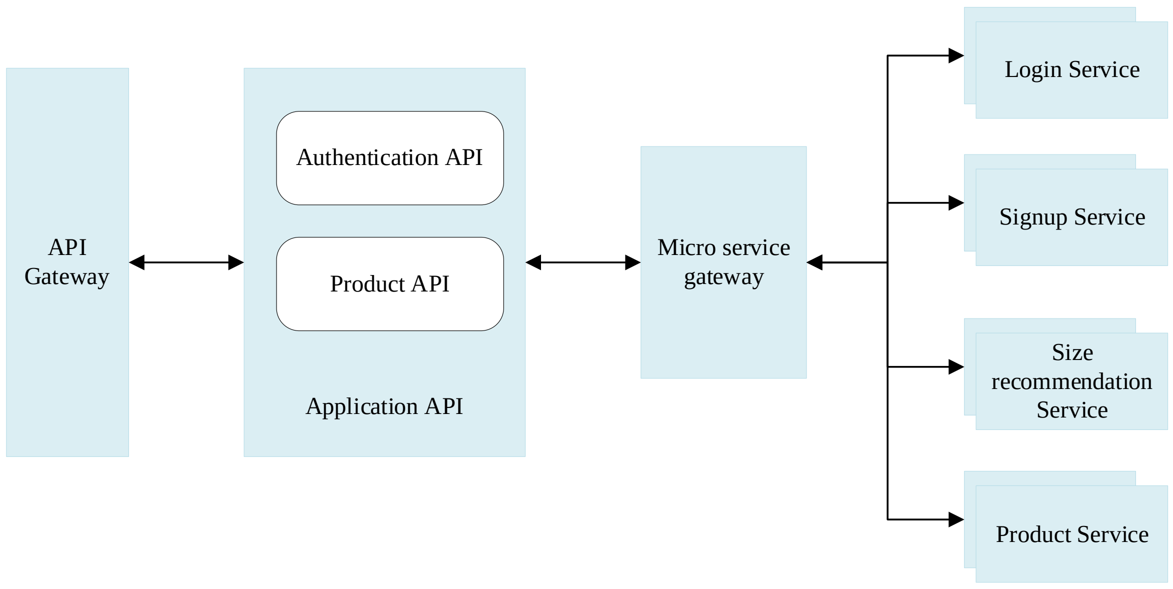

Fig. 3: Synchronous OS schedule

4. Results

A set of experiments has been designed and configured to verify the performance of the proposed architecture. The test cases involve three steps:

- Comparing the performance of APIs which requires a specific GPU when computing on a local server versus on AWS.
- Reviewing the communication time with variation of technologies.
- Validating the performance of the system.

4.1. Comparison between the performance of cloud server and local server

Table 1 provides the detailed configuration of the cloud server and physical server in the experiment. Three APIs:

- Body Reconstruction with Image.
- Body Reconstruction with measurements.

- Face Reconstruction (which require massive specification for processing) and the 3D model's outcome are used to verify the performance in response time.

In the scope of this experiment, the communication time (sending requests and receiving responses) is not included in the response time.

Table 2 indicates the average time in 100 processing between the cloud server and the local server. It could be seen that the local server with RTX 8000 brings better results when handling APIs with regard to images, 0.63s (9.3%) and 0.58s (21.3%) respectively. Overall, the difference between processing APIs on the cloud and on local is small, and the unequal time is acceptable to the user's requirements.

Table 1: Computing platform in detail

No	Computing platform	Specification Detail
1	AWS cloud g4dn.xlarge	- vCPUs: 4 - Memory: 16 GB - GPU: 1 NVIDIA Tesla T4
2	Dell Precision 5820 Tower	- vCPUs: 16 - Memory: 64 GB - GPU: 1 NVIDIA Quadro RTX 8000

Table 2: Average processing time (s)

	AWS cloud	Dell Precision 5820 Tower
/predict_body/from_image	6.77	6.14
/predict_body/from_measu-rements	33.11	33.88
/predict_face	2.72	2.14

4.2. The effect in difference of communication technologies

In this experiment, the communication time that includes sending the load and receiving the results is considered based on the variation of communication technologies. There are four types used to compare:

- Lan (on local server)
- Wi-Fi
- 4G
- 3G

Table 3 illustrates the average communication time 100 times continuously. 3G provides the worst results, with approximately 30%, 25%, and 100% slower than Lan, Wi-fi, and 4G, respectively. Meanwhile, the difference in the fastest and slowest communication time between LAN, Wi-Fi, and 4G is quite small, 7%, 2%, and 8% respectively with each API. It proved the similarity in deploying the system on the cloud and physical server.

Table 3: Communication time with multi-communication technologies(s)

	Local (dell)	Wifi	4G	3G
/predict_body/from_image	7.67	7.50	7.11	10.72
/predict_body/from_measu-rements	36.25	36.91	36.56	43.51
/predict_face	3.97	3.96	4.30	10.72

4.3. Stability and reliability of system

In this part of the research, two experiments are designed to observe the response time and the stability of the system. There are 7 APIs with multi-output and input which are used to simulate the main flow of clients. Two experiments are:

- 2k clients request all APIs (one by one) at the same time.
- 2k clients request one of the APIs at the same time.

Table 4 and Table 5 indicate the average time and error rate of these experiments. It could be seen that the time for processing images and reconstructing the face in Table 4 is the largest due to the serial configuration, and vice versa for other APIs which are configured to process parallel. Meanwhile, when 20k clients request 1 API at the same time, the time for processing parallel APIs is much slower, approximately 6 times than before. However, two serial APIs bring similar results. The most important factor of this experiment is the error rate with a non-error rate. It proves the efficiency and stability of Virtual Try-on architecture.

Table 4: Communication time with multi-communication technologies

No	API	Client	Duration (s)	Sample	Average (ms)	Min (ms)	Max (ms)	Error (%)
1	from_image	2000	1	2000	54865	6209	103004	0
2	predict_face	2000	1	2000	35354	4511	61533	0
3	predict_body /from_measurements	2000	10	20000	99	87	115	0
4	get_cloth_model	2000	10	20000	91	78	108	0
5	get_hair_model	2000	10	20000	90	77	108	0
6	change_human_skin	2000	10	20000	3438	80	14553	0
7	recommend_size	2000	10	20000	94	80	111	0

Table 5: The performance of system in receiving of client's request to process one of APIs at the same time

No	API	Client	Duration (s)	Sample	Average (ms)	Min (ms)	Max (ms)	Error (%)
1	from_image	2000	1	2000	56890	6903	114601	0
2	predict_face	2000	1	2000	40923	18045	63693	0
3	predict_body /from_measurements	2000	10	20000	617	80	3554	0
4	get_cloth_model	2000	10	20000	815	85	19806	0
5	get_hair_model	2000	10	20000	584	77	8537	0
6	change_human_skin	2000	10	20000	12408	1416	23747	0
7	recommend_size	2000	10	20000	749	85	18920	0

5. Conclusion

The research presents a state-of-the-art cloud-based architecture for Virtual Try-on solutions. A hybrid design combining Amazon Web Services with a local server is proposed to ensure the flexibility and security of the system. Deploying APIs has diversity in data's input and output, especially with 3D data that requires a specific process and hardware device to ensure system synchronization and stability. The research has proposed a synchronous outsourcing schedule combined with an asynchronous out-sourcing schedule to guarantee the performance of the whole system with 50k clients using at the same time. Besides, the architecture would ensure scalability both in horizontal and vertical extension in the future. The results from different case studies have validated and verified the performance and efficiency of the proposed architecture. In the future, this application would be public on App Store and CH Play to collect users' opinions, together with deploying more services to enhance user experiences. In addition, the researchers would focus on improving the performance of APIs and invest in deploying several APIs on the client's device with thread parallelization techniques to reduce operating costs.

Acknowledgment

The authors would like to thank all members of the 3DR team for their contribution. This research is fully funded by Viettel High Technology Industries Corporation.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Alsboui TA, Qin Y, and Hill R (2019). Enabling distributed intelligence in the internet of things using the IOTA tangle architecture. In the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019), Science and Technology Publications, Heraklion, Greece: 392-398. https://doi.org/10.5220/0007751403920398

Benedetto JI, Valenzuela G, Sanabria P, Neyem A, Navon J, and Poellabauer C (2018). MobiCOP: A scalable and reliable mobile code offloading solution. Wireless Communications and Mobile Computing, 2018: 8715294. https://doi.org/10.1155/2018/8715294

Biswas M and Whaiduzzaman MD (2018). Efficient mobile cloud computing through computation offloading. International Journal of Advancements in Technology, 10(1): 1000225.

Fernando N, Loke SW, and Rahayu W (2013). Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1): 84-106. https://doi.org/10.1016/j.future.2012.05.023

Kemp R, Palmer N, Kielmann T, and Bal H (2010). Cuckoo: A computation offloading framework for smartphones. In the International Conference on Mobile Computing, Applications, and Services, Springer, Santa Clara, USA: 59-79. https://doi.org/10.1007/978-3-642-29336-8_4

Liu L, Li H, and Gruteser M (2019). Edge assisted real-time object detection for mobile augmented reality. In The 25th Annual International Conference on Mobile Computing and Networking, Association for Computing Machinery, Los Cabos, Mexico: 1-16. https://doi.org/10.1145/3300061.3300116

Lu X, Kumaravelu G, and Okwudire CE (2019). An evaluation of data size reduction techniques for improving the reliability of cloud-based CNC for a 3D printer. Procedia Manufacturing, 34: 903-910. https://doi.org/10.1016/j.promfg.2019.06.157

Ribes VS, Mora H, Sobecki A, and Gimeno FJM (2020). Mobile cloud computing architecture for massively parallelizable geometric computation. Computers in Industry, 123: 103336. https://doi.org/10.1016/j.compind.2020.103336

Statista (2020). Fashion eCommerce report 2021: Statista digital market outlook. Available online at: https://www.statista.com/study/38340/ecommerce-report-fashion/

Zhang Y, Liu H, Jiao L, and Fu X (2012). To offload or not to offload: An efficient code partition algorithm for mobile cloud computing. In the 1st International Conference on Cloud Networking (CLOUDNET), IEEE, Paris, France: 80-86. https://doi.org/10.1109/CloudNet.2012.6483660