Polyglot Vision

1st Sreeja Malladi
dept. Computer Science and Engineering
Santa Clara University School of Engineering
Santa Clara, CA, USA
smalladi@scu.edu

2nd Sourabh Deshmukh
dept. Computer Science and Engineering
Santa Clara University School of Engineering
Santa Clara, CA, USA
ssdeshmukh@scu.edu

3rd Snehal Nikam
dept. Computer Science and Engineering
Santa Clara University School of Engineering
Santa Clara, CA, USA
snikam2@scu.edu

4th Silvi Monga
dept. Computer Science and Engineering
Santa Clara University School of Engineering
Santa Clara, CA, USA
smonga@scu.edu

***Abstract*—The Polyglot Vision project is dedicated to solving the problem of language barriers in videos. We're using advanced technology like cloud computing and artificial intelligence to make translating subtitles easier. Specifically, we're focusing on translating subtitles from English to Spanish. To build our system, we rely on Amazon Web Services (AWS), a powerful platform that provides us with the tools and infrastructure we need. We've put together a mix of different AWS services like Lambda functions, EC2 instances, and Terraform to create a system that can handle real-time subtitle translation efficiently. Behind the scenes, we've developed a sophisticated backend using Flask, a web framework for Python, to handle all the processing involved in translating subtitles. This backend is seamlessly integrated with AWS, ensuring smooth communication between different parts of the system. On the user-facing side, we've designed a user-friendly website using Vue.js, a popular JavaScript framework, to provide an intuitive and enjoyable experience for viewers. Our primary achievement so far has been making videos more accessible to Spanish speakers, but we're not stopping there. We have ambitious plans to expand our language support to include more languages, allowing even more people around the world to enjoy video content in their native language. Additionally, we're working on improving our system's capabilities to handle larger video files, ensuring that our solution remains scalable and efficient as it continues to grow. Ultimately, our goal is to make the world a more connected and inclusive place by breaking down language barriers in multimedia communication. Through innovation and dedication, we're striving to empower people from diverse linguistic backgrounds to access and enjoy video content without limitations.**

***Keywords*—AWS, Video, Translate, Transcribe, Terraform**

I. Introduction

Polyglot Vision addresses the complexity of translating video content into multiple languages, a task that is both challenging and essential in our interconnected world. Traditional methods often need to be more efficient, tied to significant manual effort or automated systems that lack linguistic finesse. Additionally, the hefty infrastructure typically required for speech recognition is beyond the reach of many users and smaller enterprises. Our project simplifies this process through an automated translation system hosted on AWS, accessible at app.polyglotvision.online. We leverage a sequence of AWS Lambda functions that activate upon video uploads to an S3 bucket, utilizing Amazon Transcribe for transcription and Amazon Translate for translation. This process ends with the merging of subtitles with the original video, all facilitated by a cloud infrastructure orchestrated via Terraform. Polyglot Vision's architecture is distinct in its operational efficiency, with three dedicated containers for the backend, front end, and subtitle API, each enhancing the system's manageability and scalability. This containerized setup enables the handling of increased video file sizes and adaptability to user demand. Contrasting with existing tools that are available in the market, our service enhances the user experience by automating each step and providing updates through AWS SES when subtitles are ready. Secure user management is ensured through AWS Cognito and IAM, with Route 53 managing our domain, underscoring our commitment to a user-friendly and secure service. This paper will explore Polyglot Vision's structure, functionality, and the value it adds in making video content more accessible on a global scale, marking a significant step towards addressing the demand for multi-language video content accessibility.

II. Background and related work

In today's interconnected world, video content serves as a universal medium, transcending geographical and linguistic boundaries. Yet, the challenge of making this content accessible to diverse global audiences remains, particularly due to language barriers. Traditional methods for subtitle translation, ranging from manual efforts to semi-automated tools, often need to catch up in terms of scalability, accuracy, or both. The advent of cloud computing has dramatically shifted this landscape, offering powerful computational resources and sophisticated AI/ML services on demand. Amazon Web Services (AWS) has emerged as a leader in this space, providing a suite of services tailored to the challenges of video subtitle translation, such as AWS Transcribe for speech-to-text conversion, AWS Translate for automatic translation, and AWS Lambda for serverless computing.

Existing tools and platforms, such as Clipchamp [9] and Veed.io [10], have set preliminary standards for integrating subtitle features into video content. These platforms have been instrumental in illustrating the practical aspects of subtitle integration, from user interface design to the backend processing of video files. However, they often require manual intervention for translation tasks or need more comprehensive automation that cloud services can offer.

Open-source projects on platforms like GitHub [11] have further enriched the landscape, providing a range of solutions from standalone subtitle translation tools to comprehensive video editing suites with integrated translation capabilities. These projects highlight the community's effort to tackle the problem from various angles, offering insights into alternative approaches and the potential for collaborative innovation.

Our project, Polyglot Vision, differentiates itself by fully embracing the capabilities offered by AWS, ensuring scalability and efficiency in processing and translating video content on a large scale. A critical aspect of our architecture is the use of Terraform for infrastructure as code, allowing automated provisioning of AWS resources, complemented by Docker containers for deploying different parts of our application. This approach underscores Polyglot Vision's commitment to delivering a resilient and adaptable solution.

The exploration of existing tools and open-source projects underscored a market gap for a fully automated, cloud-based subtitle translation service—a gap Polyglot Vision aims to fill. Inspired by the potential of AWS services and the flexibility of containerized applications, we strive to deliver a service that not only surpasses existing solutions in efficiency, accuracy, and scalability but also sets a new standard for making video content universally accessible and understandable. Polyglot Vision stands as a testament to the power of leveraging cutting-edge cloud technologies and infrastructure management practices to address the intricate challenges of content accessibility, paving the way for a future where video content can be enjoyed by a global audience, regardless of linguistic barriers.

Public Cloud Services

AWS offers a wide range of services catering to diverse computing needs, spanning from Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) to Software as a Service (SaaS). The advantages of AWS include scalability, flexibility, reliability, and cost-efficiency. Polyglot Vision leverages various AWS services like EC2, Lambda, IAM, Cognito, DynamoDB, S3, Transcribe, and Translate for its video subtitle translation solution.

AWS Services Utilized in Polyglot Vision

- AWS EC2: Elastic Compute Cloud (EC2) provided the virtual computing infrastructure for our application's server-based tasks, including video processing and scalable resource management. Our setup incorporated Application Load Balancers, Target Groups for frontend, backend, and worker services, and a dedicated security group, ensuring efficient traffic distribution, enhanced security, and optimal service delivery. While we currently scale EC2 instances manually, we plan to implement Auto Scaling Groups to adjust resources based on demand dynamically, maintaining Polyglot Vision's robust and responsive subtitle translation services.
- AWS Lambda: Lambda enabled serverless computing, allowing us to execute code without managing servers. This service automatically scales, offering a cost-effective solution for automating the transcription and translation of video subtitles. The serverless model enhanced our system's efficiency, enabling seamless video subtitle translation without the complexity of server management.
- AWS IAM: Identity and Access Management (IAM) controlled access to AWS resources, managing users, security credentials, and permissions. IAM ensured secure and streamlined operations within our cloud infrastructure by providing precise permission controls for AWS services and resources.
- AWS Cognito: Cognito offered user sign-up, sign-in, and access control capabilities for our mobile and web applications. It seamlessly integrates with Lambda for custom authentication challenges, creating a secure user directory that scales to millions of users, ensuring secure and seamless user access to Polyglot Vision's features.
- AWS DynamoDB: DynamoDB, a fully managed, scalable NoSQL database service, manages user information and video data for Polyglot Vision. Known for its low latency and flexible data model, DynamoDB automatically scaled to meet the demands of our application, ensuring high performance, reliability, and efficient data retrieval, supporting our video subtitle translation service's dynamic requirements without extensive database management.
- AWS S3: Simple Storage Service (S3) provided scalable object storage for data backup, collection, and analytics. Within Polyglot Vision, S3 played a pivotal role in securely storing and retrieving video content, ensuring easy access to data for our subtitle translation processes.
- AWS Transcribe: Transcribe utilized advanced speech recognition to convert video audio into text, facilitating seamless subtitle creation. This service was crucial for automating video transcription in our project, laying the groundwork for our subsequent translation process, and making video content universally accessible.
- AWS Translate: Translate, a neural machine translation service, delivers fast, high-quality, and cost-effective language translation. In Polyglot Vision, we utilized Amazon Translate to convert transcribed subtitles into Spanish, enabling our solution to support global accessibility by providing video content understandable to a diverse international audience.
- Route53: Route 53 is a highly available and scalable Domain Name System (DNS) service offered by Amazon Web Services (AWS). Route 53 directs user traffic to your web applications, ensuring users end up at the right place. It can route traffic to various AWS resources like EC2 instances or S3 buckets (storage for websites). Our

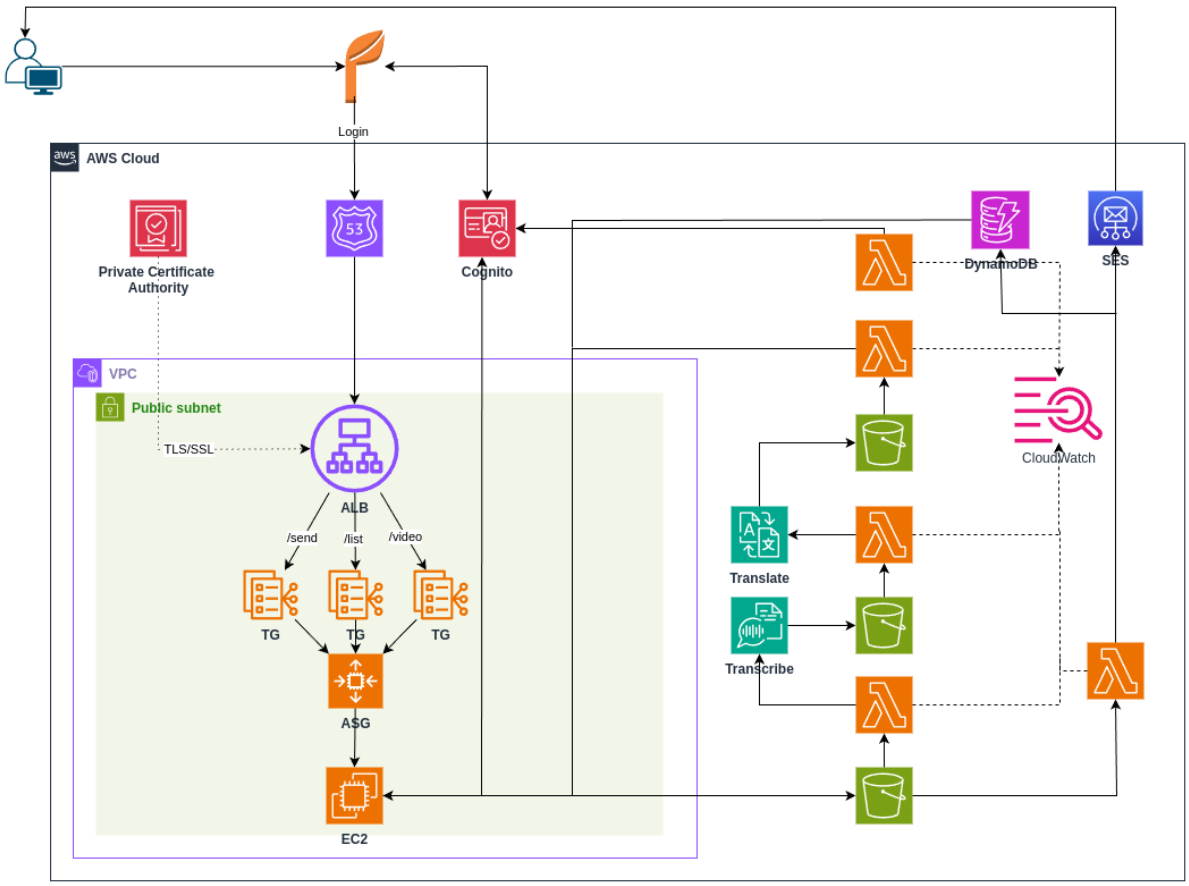

Fig. 1. System Architecture

project leverages Route 53 for intelligent domain routing. Based on the path requested in the URL, Route 53 directs traffic to the appropriate EC2 instance and its specific port where the corresponding module is running.

- AWS SES: AWS Simple Email Service (SES) is a cost-effective email-sending and receiving platform offered by Amazon Web Services. It leverages the same reliable infrastructure Amazon uses for its own communication so that you can be confident in its scalability and uptime. Our project integrates Amazon SES to automate the notification process for users. Once the targeted video subtitle merging is complete, SES triggers an email informing the user that their final video with subtitles is ready for download.

By integrating these AWS services, Polyglot Vision ensures a robust, scalable, and reliable platform for deploying its video subtitle translation solution while benefiting from the flexibility and cost-effectiveness of cloud computing services provided by AWS.

Approach

Overview of High-level architecture of Polyglot Vision

- Domain and Access: Users interact with the system through the dedicated domain app.polyglotvision.online, which serves as the entry point to the services offered by Polyglot Vision.
- Registration and Authentication: New users register on the website. This registration data is securely handled by AWS Cognito, which provides robust user authentication and management. Registered users go through a verification process, after which they can log in to access the translation services.
- Video Upload and Storage: Upon successful login, users can upload videos to the platform. Uploaded videos are securely stored in an AWS S3 bucket, ready for processing.
- Video Upload and Storage: Upon successful login, users can upload videos to the platform. Uploaded videos are securely stored in an AWS S3 bucket, ready for processing.
- Transcription and Translation Workflow: AWS Lambda functions are triggered in response to new video uploads. AWS Transcribe is engaged in transcribing the audio content of the video to text. The transcribed text is then translated into the desired language using AWS Translate.
- Subtitle Integration: Another set of AWS Lambda functions is responsible for chunking the video and merging the translated subtitles with the original video.

- Database Management: Throughout this process, AWS DynamoDB is utilized to store and manage user data and video processing statuses.
- Notification and Delivery: Once the video with translated subtitles is ready, AWS Simple Email Service (SES) notifies the user. The user can then download the translated video directly from the platform.
- Infrastructure Automation: The entire AWS resource stack is managed and provisioned using Terraform, which allows for infrastructure as code, ensuring that all cloud resources are consistently deployed and managed.
- Containerization: The system's backend, frontend, and subtitle processing services are encapsulated within Docker containers, providing isolation, scalability, and ease of deployment.

SOFTWARE ARCHITECTURE:

The software stack is engineered to be resilient and modular, employing containerization and serverless technologies to ensure both flexibility and reliability.

A. Frontend

The front end of our project as seen in Fig. 3 and Fig. 4 is meticulously crafted using Vue.js, a dynamic JavaScript framework renowned for its intuitive nature, adaptability, and exceptional performance in constructing interactive user interfaces. Leveraging the robust capabilities of Vue.js, we've engineered a seamlessly navigable and responsive design that facilitates effortless video uploads and streamlined translation option selection. This framework empowers us to forge an immersive user experience characterized by swift updates and a well-organized structure, thereby amplifying the accessibility of our video subtitle translation service. At the core of our frontend architecture lies Vue.js, which empowers us to create a compelling user interface with unparalleled efficiency and sophistication. By harnessing Vue.js's reactive components and leveraging its expansive ecosystem replete with tools for state management and development facilitation, our application exhibits unparalleled responsiveness and scalability. This strategic utilization of Vue.js not only enhances the overall user experience but also lays a robust foundation for future feature enhancements and seamless scalability. The front interface comprises three distinct pages, each meticulously designed to fulfill specific functionalities and enhance user interaction. The login page serves as the gateway to the application, providing users with a secure authentication mechanism to access their accounts. Meanwhile, the registration page offers a seamless onboarding experience for new users, facilitating swift account creation and entry into the application ecosystem. Finally, the main application video processing page represents the heart of our service, where users can effortlessly upload videos and select translation options, transforming English audio into Spanish subtitles with unparalleled ease and efficiency.

B. Backend

The backend of Polyglot Vision is engineered with Flask, a lightweight yet powerful web framework that is the driving force behind our application's server-side logic. Its primary responsibilities include the management of video uploads, user authentication, and seamless interaction with a suite of AWS services.

Utilizing Flask's straightforward routing system, the backend handles crucial endpoints such as '/send' for video uploads and '/list' for listing videos associated with a user. This system allows for clear and efficient management of user interactions with the platform. Security and authentication are at the forefront of our operations, with AWS Cognito rigorously managing user access, thus ensuring that only verified users can interact with the system.

Video content, once uploaded, is securely stored in an Amazon S3 bucket, while metadata pertaining to the video and user is systematically organized in AWS DynamoDB. This not only aids in achieving a high level of performance and scalability but also enables dynamic data management, where the environment variables configured within Flask app settings ensure secure and controlled access to AWS resources.

This backend framework is designed to be both resilient and agile, facilitating the rapid upload and retrieval of content. Environment variables, loaded through the dotenv main module, are crucial for maintaining the security and accessibility of AWS services, allowing our Flask application to adapt to evolving infrastructure requirements without exposing sensitive configuration details.

The code snippet provided outlines the fundamental operations facilitated by the Flask backend, from initiating video processing to updating database records and notifying users of the job's commencement. This robust system architecture ensures that Polyglot Vision's backend remains efficient, secure, and primed for the demands of an automated video subtitle translation service.

C. Lambda Functions

- Pre-Signup Lambda: This Lambda function is triggered when a user signs up for your service. Its purpose is to verify the user's identity and ensure that they are authorized to use the service. This involves checking the user's credentials, such as their email address or password, and verifying that they have the necessary permissions to access the service.
- Transcribe Lambda: This Lambda function is triggered when a video file is uploaded to an S3 bucket. Its purpose is to transcribe the video file using Amazon Transcribe, which is a service that converts spoken language into text in audio or video files. The transcribed text is then placed back into the S3 bucket.
- Translate Lambda: This Lambda function is triggered when a JSON file containing the transcribed text is placed in an S3 bucket. Its purpose is to translate the transcribed text into multiple languages using Amazon Translate, which is a service that translates text from one language to another. The translated text is then placed back into the S3 bucket.

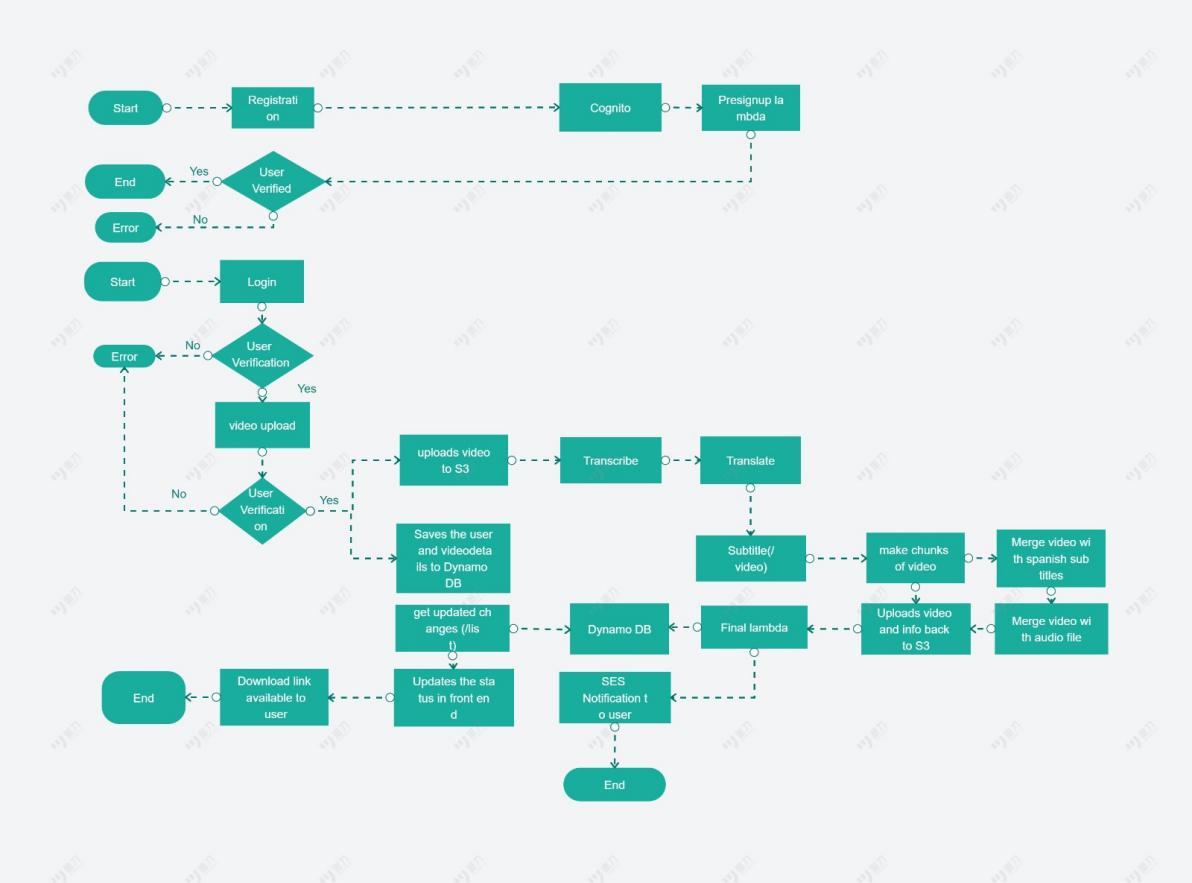

Fig. 2. Software Architecture

- Subtitle Lambda: This Lambda function is triggered when a VTT file (a file containing the transcribed text and translations) is placed in an S3 bucket. Its purpose is to generate subtitles for the video file based on the transcribed text and translations. The subtitles are then placed back into the S3 bucket.
- Final Lambda: This Lambda function is triggered when the subtitle generation process is complete and the final video, along with its information, is uploaded to the S3 bucket. Its purpose is to update the video information in DynamoDB, which is a NoSQL database service, and mark the job as complete. Additionally, it initiates SES, an email service, to notify the user that the job is finished.

D. Subtitle API

The Subtitle API is the heart of our project, orchestrating the transformation of raw videos into captivating, multilingual masterpieces. Similar to our backend, it leverages Flask to configure the /video endpoint. This endpoint serves as the entry point, receiving the necessary data to generate the final video.

- Streamlined Data Acquisition: The API seamlessly retrieves video files directly from Amazon S3, ensuring efficient and reliable access. Downloaded video files are temporarily stored in local folders for optimal processing.
- Precise Subtitle Integration: To achieve flawless subtitle timing, the API meticulously divides the video into manageable chunks, each lasting 180 seconds. For each chunk, relevant subtitles are extracted with precision from the VTT file. We utilize the powerful CompositeVideoClip method from the MoviePy Python library[5] to seamlessly merge the extracted subtitles with the corresponding video chunk.
- Effortless Concatenation and Delivery: Once all chunks are processed with their respective subtitles as shown in Fig. 5, the API expertly concatenates them into the final video masterpiece. This consolidation process is triggered by a dedicated Lambda function, accessible through the /videos endpoint.
- Secure Cloud Storage and Organized Management: The final, subtitle-enriched videos are securely stored in Amazon S3, a reliable and scalable cloud storage solution. To facilitate easy management and retrieval, detailed information (metadata) about each video is stored in DynamoDB.
- Key features:

1) Efficient Processing: By segmenting videos into manageable chunks, the API achieves highly accurate subtitle synchronization, eliminating any out-

Fig. 3. Application Frontend

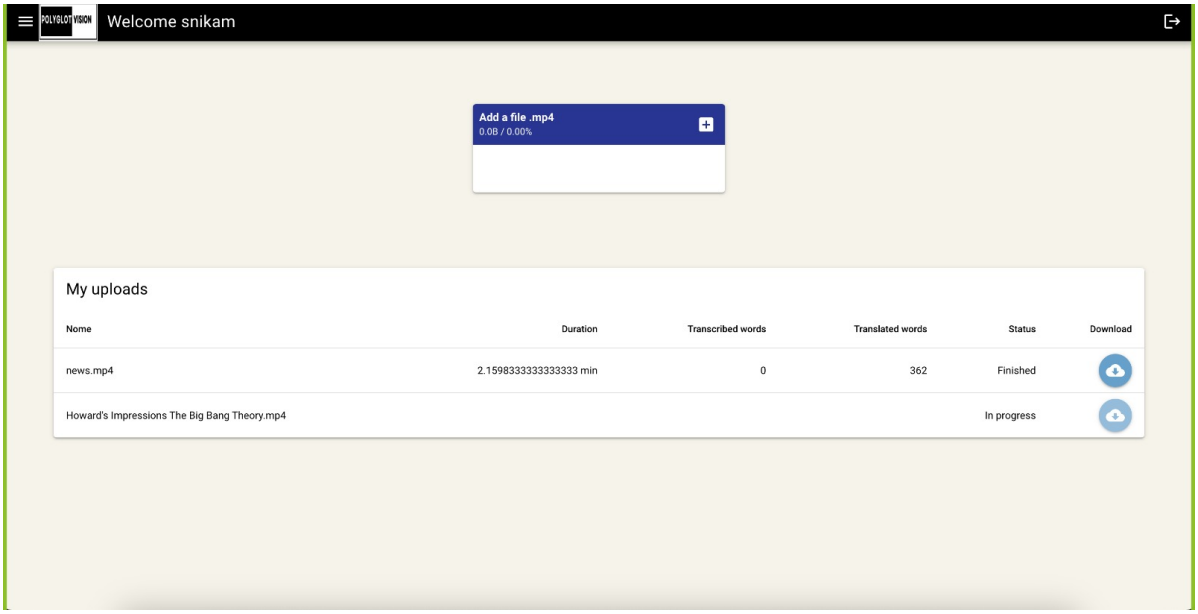

Fig. 4. Application Interface

of-sync frustration for viewers.

2) Cloud Storage and Scalability: We leverage the power of Amazon S3 and DynamoDB to guarantee secure storage for your videos and efficient management of associated metadata.
3) Multilingual Powerhouse: The streamlined process empowers you to cater to a global audience by offering multilingual subtitle support, making your videos truly accessible and engaging for viewers worldwide.

E. Docker

Docker's containerization technology brings significant advantages to our application. By packaging our application and its dependencies into standardized units called containers, Docker streamlines development, deployment, and overall manageability. This approach also empowers us to achieve superior scalability. In our project, we have containerized all our modules i.e. frontend, backend, and subtitle-API, into containers and executed it on a single EC2 instance. With Docker, we can effortlessly scale our application up or down based on real-time demands, optimizing resource utilization within our infrastructure. Furthermore, Docker's

Fig. 5. Video Processing

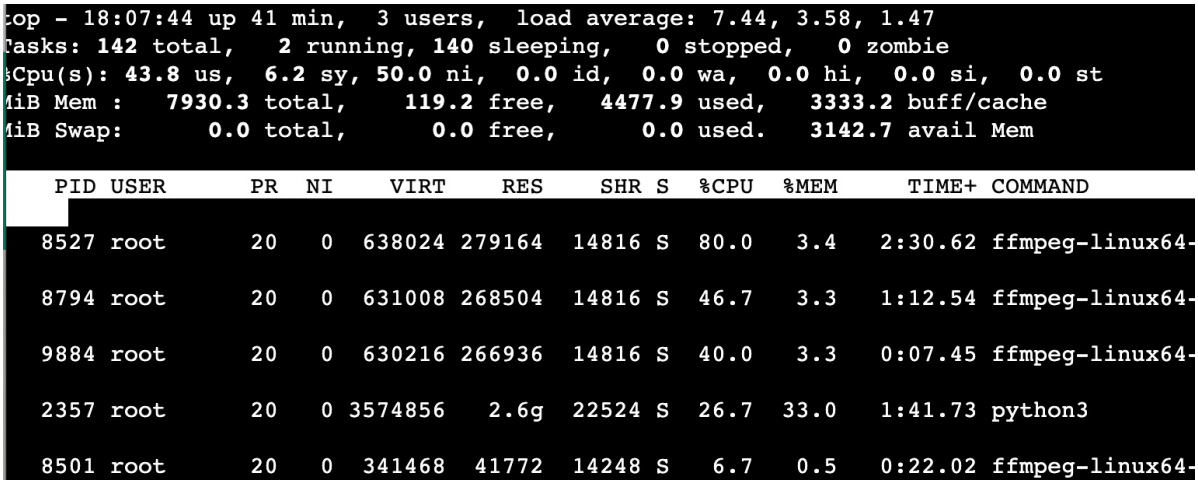

Fig. 6. System Metrcics

"restart=always" policy ensures automatic service restarts in the event of disconnections, promoting application resilience. Additionally, we leverage Docker's efficient logging mechanism to monitor meticulously and track logs generated by our application services, facilitating troubleshooting and performance optimization.

F. Terraform

Terraform, an open-source Infrastructure as Code (IaC) tool from HashiCorp, empowers us to streamline infrastructure provisioning and management. It replaces manual configuration with a human-readable language (HashiCorp Configuration Language or JSON), allowing users to define the desired infrastructure state. This includes resources like virtual machines, networks, storage, and more across various cloud providers like AWS, Google Cloud Platform, and Azure. At the heart of Terraform lies the state file, the single source of truth that tracks all deployed resources. Terraform offers a powerful suite of commands to manage your infrastructure as code. The most commonly used commands are -

- terraform init: Initializes a new Terraform project in the current directory.
- terraform plan: Create an execution plan that outlines the infrastructure changes Terraform will make based on your configuration files.
- terraform apply: Applies the planned changes to your infrastructure, provisioning resources as defined in your code.
- terraform destroy: Tears down your infrastructure based on the current state file, removing resources.

In our project, Terraform played a pivotal role in building our AWS infrastructure. Its key strengths, scalability, and flexibility ensured consistency throughout the development lifecycle. Terraform not only minimized manual effort but also led to significant cost savings. With Terraform code in place, provisioning or destroying infrastructure becomes a matter of executing a few commands, simplifying the entire process.

OUTCOMES

Polyglot Vision has been meticulously designed to deliver optimal performance for video subtitle translation. The system,

tested on an EC2 t2.large instance, exhibits robust capabilities and certain limitations, as evidenced by the monitoring data and logs provided.

- Runtime Details and System Configurability: The system's versatility allows for configuration adjustments to accommodate various video sizes and concurrent processing demands. Processing a 20 MB video file typically takes around 5 minutes, encompassing upload, processing by AWS services, and the return of the translated video. However, it's important to note that a 35 MB video file can consume around 80% of CPU usage, and concurrent processing of a 25 MB video on the same instance may lead to CPU saturation, resulting in request failures during stress tests.
- End-to-end System Performance: The end-to-end performance of the system has been rigorously tested. Typical video processing, involving upload, transcription, translation, and download, takes approximately 5 minutes for a 20 MB video file. This duration is an aggregate of various stages, including network latency, processing time by AWS Lambda functions, and the time taken by AWS Transcribe and Translate services. The system's performance has been evaluated through a series of metrics as shown in Fig. 7:
    1) Latency: In Fig. 8, the 'hey' command output reflects the system's response times under load, with most requests completed in under a second. This underscores the system's ability to handle transactions promptly.
    2) Memory Distribution: Fig. 6 depicts system logs which showcases memory allocation and usage by various processes. The memory usage patterns from the system logs reflect the intensive demand during the video processing stages. High memory consumption by Python and ffmpeg processes denotes the resource-intensive nature of video processing tasks.
    3) Network Traffic: The 'Network out' in Fig. 7, showcases a fluctuating pattern of outgoing network traffic, with peaks suggesting periods of intense activity, likely corresponding to video file uploads to S3. Conversely, the 'Network in' graph indicates incoming traffic to our EC2 instance, with the sharp peak possibly representing a large batch of data received, such as video content being processed. 'Network Packets In' and 'Network Packets Out' provide additional granularity on the data packets transferred during these operations.
    4) CPU Performance: The 'CPU Utilization' graph in Fig. 7 shows the system's CPU load over time. A spike to 100% usage signifies maximum capacity reached, which correlates with the intensive processing required for video transcription and translation tasks, a threshold we aim to increase with future optimizations.
- Performance under Load: The system maintains functionality with impressive performance for file sizes up to the tested limit of 20 MB. Beyond this threshold, as concurrent requests increase, the system encounters resource constraints, as showcased by the CPU credits graph where credit usage increases during peak processing times. In our application, given the crucial role of the subtitle API in merging subtitle files with videos and its substantial consumption of CPU resources during video processing, we performed a sample load test on the API. This involved utilizing the 'hey' command with specific parameters: hey -z=300s -q 10 -c 7 -m POST -H "Content-Type: application/json" -d @data.json https://app.polyglotvision.online/video. The outcome of this test is depicted in the accompanying figure.

Polyglot Vision's current deployment on a t2.large instance is calibrated for video files of up to 20 MB to ensure quality performance. The gathered data indicates a well-functioning system under designated conditions, while also highlighting areas for enhancement, particularly in scaling to accommodate larger files. The detailed analytics provide a foundation for ongoing improvements, aimed at extending the system's capabilities to meet and exceed the evolving needs of video subtitle translation services.

Analysis and Future Work

There's always room for growth! As we strive to deliver the best possible experience, here are a few areas where we can prioritize further improvements and explore exciting future possibilities:

Reaching a Wider Audience:

Multilingual Support: We're adding language support to cater to a broader audience and their linguistic needs. Users will have the flexibility to choose their preferred language for the user interface (UI) and access content in multiple languages, enriching the viewing experience for diverse users.

Enhanced Accessibility:

Integration with Amazon Polly will provide a dubbed experience with multiple languages, bringing translated subtitles to life with realistic-sounding audio. We'll also be making UI improvements with adjustable subtitles for better readability.

Optimizing Performance and Scalability:

Application Scaling: To address limitations with large video files (over 20MB), we're exploring application scaling via Elastic Kubernetes Service (EKS) for backend processing. This will ensure our system can efficiently handle high-volume content without sacrificing performance.

Data-Driven Improvement:

Analytics for Improvement: By analyzing user behavior patterns, we can continuously refine our services to meet user demands better and provide an exceptional experience.

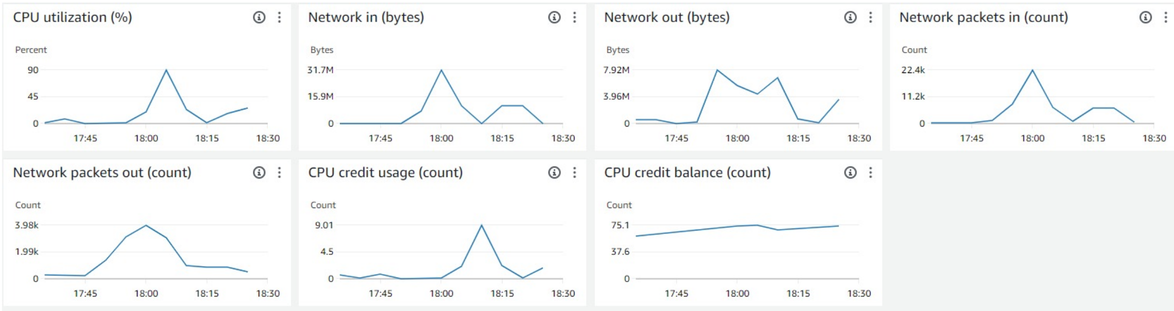

Fig. 7. Performance Metrics

```
❯ hey -z=300s -q 10 -c 7 -m POST  -H "Content-Type: application/json" -d @data.json  https://app.polyglotvision.online/video

Summary:
  Total:        300.0488 secs
  Slowest:      16.8998 secs
  Fastest:      0.0325 secs
  Average:      0.1245 secs
  Requests/sec: 34.8343

  Total data:   1741977 bytes
  Size/request: 167 bytes

Response time histogram:
  0.033 [1]     |
  1.719 [10367] |■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
  3.406 [21]    |
  5.093 [7]     |
  6.779 [0]     |
  8.466 [3]     |
  10.153 [7]    |
  11.840 [11]   |
  13.526 [0]    |
  15.213 [0]    |
  16.900 [14]   |


Latency distribution:
  10% in 0.0409 secs
  25% in 0.0442 secs
  50% in 0.0499 secs
  75% in 0.0676 secs
  90% in 0.1185 secs
  95% in 0.1353 secs
  99% in 1.0153 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0003 secs, 0.0325 secs, 16.8998 secs
  DNS-lookup:   0.0001 secs, 0.0000 secs, 0.0458 secs
```

Fig. 8. Load Test Results

Conclusion

In conclusion, Polyglot Vision stands as a significant achievement in addressing the need for accessible, automated video subtitle translation. Our project has successfully developed a comprehensive system that integrates sophisticated AWS services to streamline the translation process from video upload to the final delivery of subtitled content.

Through the diligent application of cloud architecture, we've learned that scalability and efficiency can coexist with complexity. The use of AWS Lambda, Transcribe, and Translate has demonstrated the power of cloud services to handle computationally intensive tasks with ease. Our approach with Flask and Vue.js has provided us with valuable insights into creating responsive and robust applications that maintain simplicity for the user while managing complexity under the hood.

Our achievements are multifaceted. We've created a user-friendly platform that automates the time-consuming process of subtitle translation. The use of containerization with Docker and infrastructure management with Terraform has enabled us to build a system that is not only responsive and scalable but also consistent and reliable across different environments.

Despite these successes, we recognize the limitations of our current system. The project is initially configured to translate from English to Spanish, and while it is designed to be adaptable to other languages, this remains an area for future development. Additionally, while the system efficiently handles the video files within a certain size range, scaling up to accommodate much larger files presents a challenge we aim to address moving forward.

Overall, Polyglot Vision has laid a solid foundation for a cloud-based solution that makes video content more accessible

to international audiences. The project serves as a testament to the potential of cloud computing and AI/ML in overcoming language barriers and enhancing global communication. As we look to the future, we are inspired to continue refining our system, expanding language options, and increasing our processing capabilities to meet the evolving needs of video content creators and viewers around the world.

PROJECT SOURCE CODE URL

https://github.com/Snehal-Nikam/polyglot-vision-csen241

REFERENCES

[1] Blog by Rob Dachowski, “Create video subtitles with translation using machine learning“

[2] Amazon Web Services Official Documentation By aws.amazon.com “https://aws.amazon.com/documentation-overview/“

[3] Setting up Amplify Auth by Amplify Dev center “https://docs.amplify.aws/vue/prev/build-a-backend/auth/set-up-auth/“

[4] Terraform AWS Infrastructure Documentation by terraform.com “https://registry.terraform.io/providers/hashicorp/aws/latest/docs“

[5] MoviePy package official documentation “https://zulko.github.io/moviepy/ref/videotools.html“

[6] Boto3 Package official documentation by AWS “https://boto3.amazonaws.com/v1/documentation/api/latest/index.html“

[7] William Huster, Automatically Caption Your Videos with Whisper and ffmpeg “https://williamhuster.com/automatically-subtitle-videos/“

[8] Darshan Majithiya, “Generate SRT File (Subtitles) using Google Cloud’s Speech-to-Text API“ Published in Searce on Medium.

[9] Blog by ClipChamp “https://clipchamp.com/en/blog/“

[10] Veed.io resources “https://www.veed.io/blog-category/veed-guides“

[11] Extremq, Automatically subtitle any video spoken in any language to a language of your choice using AI. “https://github.com/extremq/gptsubtitler“