Title of Project: Analysis of Video in Distributed Environment for Object Detection.

List of Project Participants: Anirudh Prashant Kalghatkar, Sachin Kashinath Rathod.

Project Overview:

This project was undertaken as part of the course CSCI 5253: Datacenter Scale Computing at CU Boulder during the Fall 2023 semester. The primary objective was to implement a distributed system for processing video frames with the goal of object detection using a pre-trained Deep Neural Network (DNN).

Project Goals:

Our project focused on implementing a distributed video frame processing system for Object detection using a pre-trained Deep Neural Network (DNN). The objective was to input a video into the system and receive an output video where objects are highlighted, along with their count in each frame where the objects can be cars, people, or any other object supported by pre-trained DNN. This was achieved by dividing the video into individual frames, independently processing each frame for object detection using the pre-trained DNN, and then merging the frames while maintaining the original order. The outcome is a modified video with highlighted objects and their respective counts. The project leverages distributed processing for scalability and efficiency, contributing to the intersection of data center scale computing, deep learning, and video analytics.

For this project, we have configured pre-trained DNN to detect humans.

Software Components:

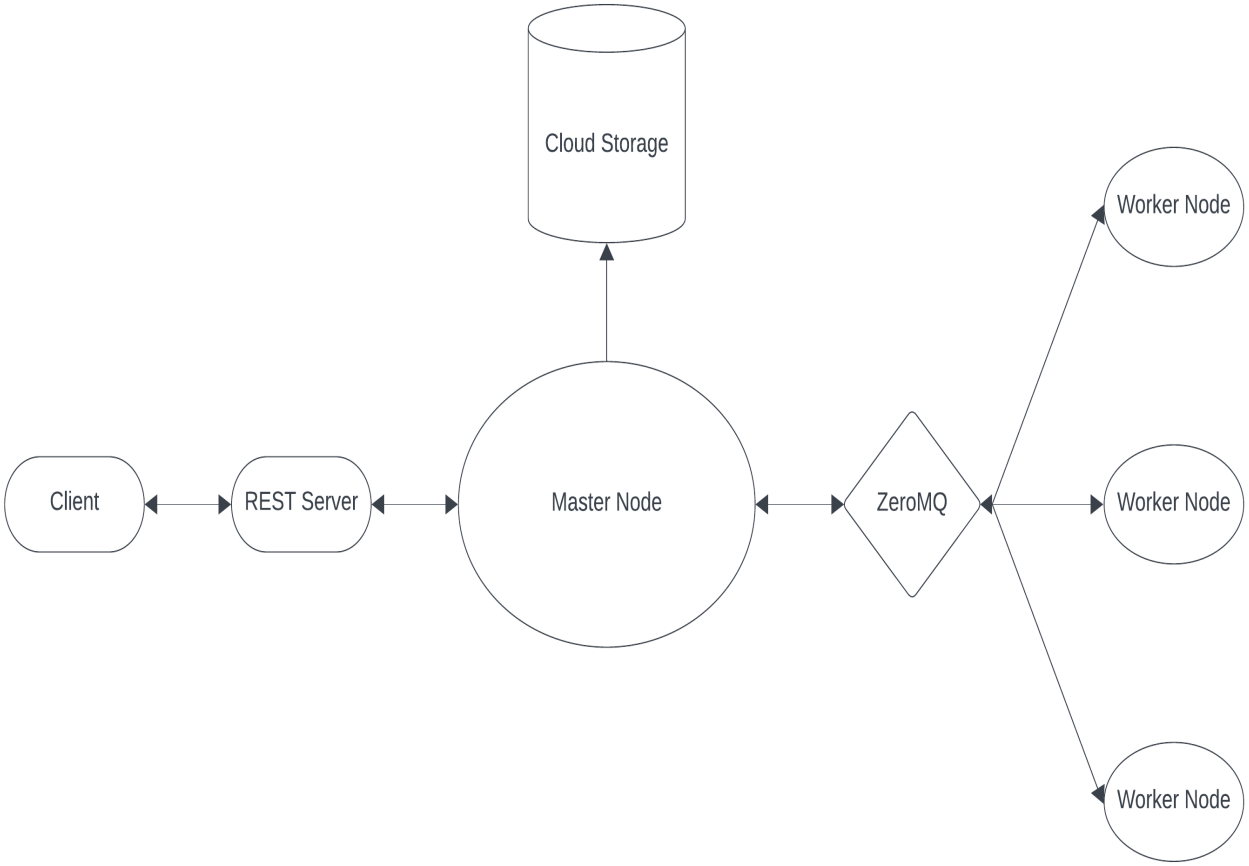
1. ZeroMQ

2. VM Instance

3. Google Cloud Storage

4. REST API Interface

Architecture Diagram:



Description of each Component:

1) ZeroMQ:

Purpose:

In our project, ZeroMQ serves as a crucial tool for implementing message queues. Having gained familiarity with broker systems like RabbitMQ and Redis in our course, we opted to explore brokerless systems for specific reasons outlined below.

Advantages:

- **Brokerless Architecture:** ZeroMQ distinguishes itself by not requiring a dedicated message broker, setting it apart from other message queue implementations like RabbitMQ. This brokerless approach reduces infrastructure complexity, making it a low-cost and straightforward solution to implement.
- **Positive Acknowledgment:** ZeroMQ utilizes positive acknowledgment for frame transmission, ensuring the integrity of video frames. This acknowledgment mechanism enhances reliability by minimizing the risk of frame loss during communication between components.
- **Image Transfer Facilitation:** In our project, where image transfer between different virtual machines is a core requirement, ZeroMQ proves highly efficient. The availability of a dedicated wrapper, ImageZMQ, tailored for image transfer simplifies the implementation process, contributing to the overall ease of use.
- **Scalability:** ZeroMQ exhibits excellent scalability, a critical characteristic for our distributed application. Its ability to seamlessly handle the complexities of a distributed environment aligns with the demands of our project, providing a robust foundation for scaling the system as needed.

Disadvantages:

- **Management Complexity:** While ZeroMQ's brokerless architecture offers flexibility, it requires careful management of relationships between different components. Unlike systems with dedicated brokers, where the broker manages interactions, in ZeroMQ, developers need to handle these relationships more

explicitly. This can introduce a higher level of management complexity, especially in larger and more intricate distributed systems.

Interaction with each component:

- Video frames are transmitted between different virtual machines, and the brokerless nature of ZeroMQ simplifies the communication flow.

2) VM Instance:

Purpose:

In our project, the strategic utilization of Virtual Machine (VM) instances is pivotal, with specific roles assigned to different instances. This includes VMs acting as worker nodes, a Master node, and a client.

Advantages:

- **Familiarity and Prior Experience:** Given our previous experience with VMs in various assignments, their inclusion in the project aligns with our expertise, making the implementation more efficient. The familiarity with VMs allows for a smoother integration process.
- **Ease of Debugging:** VMs offer a straightforward debugging process. Accessing VMs via Secure Shell (SSH) provides an easy means to inspect logs and troubleshoot issues directly. This ease of debugging is particularly advantageous compared to more complex environments like Kubernetes, where direct access can be challenging.
- **Simple Deployment and Software Installation:** Deploying VMs with the required operating system on platforms like Google Cloud Platform (GCP) is straightforward. Furthermore, installing additional software on VMs is uncomplicated, providing a flexible and customizable environment for our project requirements.

Disadvantages:

- **Manual Scaling:** The manual addition and removal of worker nodes based on system load is a notable disadvantage. Unlike more automated systems such as

Kubernetes, where load balancing can be handled seamlessly, VMs necessitate manual intervention for scaling operations.

- **Inefficient Resource Utilization:** VMs may lead to suboptimal resource utilization, especially when the computational demands of the application are modest. Virtual machines typically come with a certain allocation of resources, and if these are not fully utilized, it can result in inefficiencies, both in terms of memory and processing capacity.

Interaction with each component:

- Video frames are transmitted between VMs using ZeroMQ, ensuring efficient and reliable communication.
- The REST API serves as the gateway for users to interact with the system, enabling the submission of video processing tasks and retrieval of results.
- Processed and original video data are stored and retrieved using Google Cloud Storage, providing a scalable and secure storage solution.

3) Google Cloud Storage:

Purpose:

Google Cloud Storage is integrated into our project for storing both the processed and initial video files, playing a crucial role in the data management aspect.

Advantages:

- **Efficient Handling of Large Video Files:** Google Cloud Storage excels in handling very large video files efficiently. It provides a scalable and reliable solution for both uploading and downloading, accommodating the storage demands of our video processing application.
- **Fast Access within Google Cloud Ecosystem:** Since our Virtual Machines (VMs) are hosted within the Google Cloud ecosystem, leveraging Google Cloud Storage ensures swift access to stored videos. This access speed is notably enhanced when VMs and storage are located within the same Region or Zone, minimizing latency.

Disadvantages:

- **User Convenience in Service Mode:** In a scenario where this project is deployed as a service, relying solely on Google Cloud Storage for video upload and retrieval might be perceived as cumbersome by users. Users might prefer a direct upload from their devices rather than an indirect process through Google Cloud Storage.
- **Cost Implications with Large Storage Needs:** While Google Cloud Storage accommodates very large files, there exists a theoretical limit. As storage needs grow, costs may escalate. Acquiring a larger storage limit would involve additional expenses, potentially impacting the overall operating cost of the service.

Interaction with each component:

- Virtual Machines store processed and original video data and retrieve using Google Cloud Storage API.

4) REST API Interface

Purpose:

REST API plays a pivotal role in facilitating communication between the client and the master node in our project. The client initiates the communication by sending a request, represented by a URL pointing to the Google Cloud Storage location of the video, which the master node then processes.

Advantages:

- **Ease of Implementation**: REST API is chosen for its ease of implementation. Compared to more complex alternatives like gRPC, coding a REST API is relatively straightforward. The simplicity of REST lends itself well to quick development and integration.
- **Small Payload**: The payload of a REST API request is minimal, typically consisting of a single URL. This lightweight nature, coupled with the assumption of a low volume of requests, contributes to performance that is comparable to gRPC. This observation is substantiated by a previous assignment's comparative study within the course.

- **Flexibility for Varied Calls and File Formats**: REST provides greater flexibility, capable of handling various types of calls and managing different file formats seamlessly. This flexibility aligns with the diverse requirements of our project, where the REST API must accommodate multiple types of calls and handle various file formats efficiently.

Disadvantages:

- **Connection Overhead**: REST's approach of creating a new TCP connection for each request introduces a potential performance bottleneck, especially when the client and master nodes are physically distant from each other. This connection overhead may result in slower response times compared to more optimized alternatives like gRPC.

Interaction with each component:

- The REST server, hosted on a Virtual Machine, facilitates client connections through RESTful web services.

Debug and Testing in the project:

1. **VM Logs for Fault Detection:**

- **Approach:** VM logs were extensively utilized for debugging the Master and worker nodes. SSH access to the VMs allowed close inspection of processing details.
- **Verification:** Logs were examined to ensure proper video frame splitting and object detection within individual frames.

2. **ZeroMQ Logging via SSH:**

- **Approach:** Similar SSH-based access was used to inspect ZeroMQ logs. The logging mechanism was scrutinized to ensure correct message queuing and handling of requests.
- **Verification:** Verification involved confirming that messages were appropriately queued and that all requests were handled correctly.

3. **Verbose Flag for Enhanced Debugging:**
- **Approach:** A Verbose flag was incorporated into the code, enabling the printing of detailed processes and requests during execution.
- **Verification:** This verbose output greatly facilitated debugging and testing, providing detailed insights into the system's internal workings.

4. **Verification through Final Video Comparison:**
- **Approach:** The final verification step included comparing the processed video with the original. If the new video matched in terms of frames, it signified correct frame splitting and rejoining by the master node.
- **Verification:** Ensured that the entire video processing pipeline, from frame splitting to object highlighting, was functioning accurately.

5. **Object Highlighting Verification:**
- **Approach:** Object highlighting in the final video was inspected. A successful outcome indicated that each worker node correctly identified objects within individual frames.
- **Verification:** Checked that the processed video contained highlighted objects, confirming the successful completion of tasks by worker nodes.

6. **Pre-trained Neural Network Model Assurance:**
- **Approach:** The pre-trained neural network model, having undergone previous training and testing, was incorporated into the project without the need for additional training or testing mechanisms.
- **Verification:** Relied on the model's pre-established accuracy, focusing on parameter adjustments such as confidence measures for object identification in the video.

7. **Unit and Integration Testing:**
- **Approach:** During the development phase, each software component/module underwent unit testing. Upon passing individual unit tests, an integration test was performed to ensure seamless functionality when all modules were integrated.
- **Verification:** Confirmed that the entire system operated cohesively and met the specified requirements after successful unit and integration testing.

Capabilities of the Solution:

- The proposed solution exhibits the capability to comprehensively process and analyze any given video. It achieves this by leveraging a pre-trained Deep Neural Network (DNN) configured for object detection. The system is adept at identifying and highlighting objects previously defined within the pre-trained DNN, offering a robust and versatile solution for video analysis with a focus on targeted object detection.

Limitations of the Solution:

Despite its capabilities, the solution has certain limitations that merit consideration:

- **Manual Scalability:** The scalability of the system is constrained by the manual addition and removal of worker nodes based on system load. Unlike more automated systems such as Kubernetes, where scaling operations can be handled seamlessly, our solution requires manual intervention for scaling, impacting scalability in dynamic environments.
- **Time Complexity:** The time complexity of the solution may vary based on the computational demands of the video and the efficiency of the underlying processing infrastructure. Analyzing large videos or videos with high object density may result in longer processing times, affecting real-time responsiveness.
- **Limited Object Range:** One notable limitation is that the pre-defined DNN has constraints on the types of objects it can detect. Any object outside the scope of the DNN's training data may not be recognized or may yield less accurate results. This limitation emphasizes the importance of understanding the applicability of the solution to a predefined set of objects.

Future Enhancements:

- **Transition to Automated Scaling with Kubernetes:** Implementing automated scaling systems, notably Kubernetes, to replace manual intervention in the addition and removal of worker nodes. This transition enhances scalability, ensuring efficient resource utilization and responsiveness to varying workloads.
- **Advanced DNN for Enhanced Object Detection:** Developing an advanced Deep Neural Network (DNN) capable of detecting a broader spectrum of objects with increased precision and time efficiency. This involves expanding the range of recognized objects and optimizing the model for quicker and more accurate analyses.

Meeting Project Requirements:

This Project will meet the Project Requirements as we are using 4 different Cloud technologies as specified below:

1. Message queues (ZeroMQ) - To handle processing of video frames.
2. Virtual Machines - Master and worker nodes
3. Storage services (Google Cloud Storage) - To store the input and output video.
4. REST API interface - To service the client.