Project 1: IaaS—Amazon Web Services

CSE 546: Cloud Computing

In the first project, we will build an elastic and responsive application that utilizes cloud resources and IoT devices to achieve real time object detection on videos recorded by the devices. Specifically, we will develop this application using Amazon Web Services (AWS) based cloud and Raspberry Pi based IoT.

1. Introduction

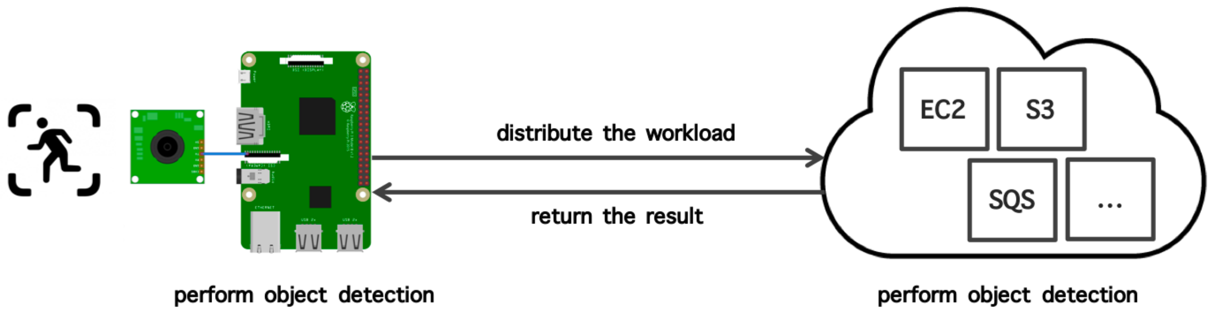
Our application will offer a meaningful service to users, and the technologies and techniques that we learn will be useful for us to build many others in the future.

We have learned in class what cloud computing is and what its benefits are. Edge computing is an emerging paradigm for enabling computation on or near the smart devices and Internet of Things (IoT) that produce the input data and provide feedback to users and the physical world. With the explosive growth of data contributed by various edge devices deployed every day, the cloud is not able to meet all the computational demands, especially when the response time requirement is tight. The continuous advancement of System-on-Chips has enabled edge devices to conduct more complicated workloads, leading to the emergence of edge computing which extends the computing from the cloud to the edge. One main advantage of edge computing is the low response time since computation is conducted close to where the input is generated and the response needs to be produced.

Therefore, cloud and edge computing are two complementary paradigms for enabling many challenging applications such as AI, AR/VR, and autonomous driving, where cloud resources can provide scalability and edge devices can deliver responsiveness.

In this project, we will utilize both cloud (using AWS) and edge (using Raspberry Pi) to develop an application that provides real-time object detection. AWS is the most widely used IaaS provider and offers a variety of compute, storage, and messaging cloud services. Raspberry Pi is a low cost, credit-card sized IoT development platform.

The architecture of the system is as follows. The Raspberry Pi records the videos using its attached camera. The object detection model can run on either the Pi or the cloud. The workload demand can be unpredictable.



The objective of this system is to minimize the end to end latency of object detection, i.e., the time from when video is recorded by the device to when the output of the object detection result is produced.

2. Instructions

2.1 Using the Pi

First please refer to this document to install OS on Raspberry Pi.

Then refer to this document to learn how to use a camera on Raspberry Pi.

We use a lightweight deep learning framework, Darknet, to perform object detection on Raspberry Pi. We will use tiny yolo as our pre-trained model for object detection.

Then follow the instructions below to run the object detection model.

1. After booting up your Raspberry Pi, follow the steps below to build Darknet on your Pi.
    ```
    git clone https://github.com/pjreddie/darknet.git
    cd darknet
    ```
    Modify line 3 in the Makefile from “OPENCV=0” to “OPENCV=1”
    ```
    make
    ```
2. Download the pre-trained model weights of tiny-yolo using the following command:
    wget https://pjreddie.com/media/files/yolov3-tiny.weights
3. If you ssh into you Pi, you need to use the command Xvfb :1 & export DISPLAY=:1 to enable x display service which enables you to run graphical applications without a display.
4. Run the Darknet program with the following command:

```
./darknet detector demo <path to cfg> <path to cfg> <path to weight> <path to
video> <option>
```

For example,

```
./darknet detector demo cfg/coco.cfg cfg/yolov3-tiny.cfg yolov3-tiny.weights
1550538579.h264
```

2.2 Using AWS

1. EC2 is used to perform object detection on the requests that the Raspberry Pi cannot handle in time. You can follow the steps below to setup Darknet on your EC2 instances:

```
git clone https://github.com/pjreddie/darknet.git
sudo apt-get update
sudo apt-get install gcc g++
cd darknet
make
```

2. SQS is used to couple the various components in the system and achieve auto-scaling. Only when SQS is not empty, EC2 will try to find the video in S3. The message in SQS should contain the key of the bucket in S3, and the EC2 will use the key to download the video from S3 bucket.

3. S3 is used to store the videos captured by the Raspberry Pi and the object detection results.

3. Requirements

1) Your app should properly utilize the resources from the Pi and AWS to minimize the end-to-end latency of object detection. At the same time, your app should not leave any AWS resources idle and wasted.

2) The Pi should always upload the recorded videos to a bucket in the form of (video_name, video_content) pairs in S3.

3) The objection detection results should be correct and they should be stored in a bucket in the form of (input, output) pairs, e.g., (“1582614401.h264”, “person”), Even though some videos are just performed on Pi, the result will also be stored in the bucket in S3.

4) Because we have limited resources from the AWS free tier, the app should use no more than 20 instances, and it should queue all the pending requests when it reaches this limit. When there is no request, the app should use only the Raspberry Pi without using EC2 instances. Note that you cannot use the AWS auto-scaling service to implement this feature. As we discussed in class, the functionality of the AWS auto-scaling service is quite limited.

4. Submission

The submission includes two steps:

1) You need to submit your implementation on Canvas by March 22nd 11:59:59pm.

   a) Your submission should be a single zip file that is named by the full names of your team members. The zip file should contain all your source code and the AWS credentials for running your application.

   b) It should also contain a simple README file that lists your group member names and S3 bucket name that stores the results of the object detection; and a detailed PDF file that describes the design of your application and any additional information that can help the instructor understand your code and run your application. A template will be provided to you for writing the report.

   c) Do not include any binary file in your zip file. Only one submission is needed per group.

   d) Failure to follow the above submission instructions will cause a penalty to your grade. The submission link will be closed immediately after the deadline.

2) You need to give a live demo to the TA by March 22nd 5pm. We will try the code that you submitted to Canvas. You have only ten minutes for the demo, and there will be no second chance if you fail the demo. We will check the following items during your demo:

   a) The end to end latency should be low. If the latency is within a reasonable range of our baseline, then you will get full marks. Otherwise, you will lose points based on how much slower the end to end latency of your application is compared to our baseline.

   b) The instances in your app that your app utilizes should automatically scale in and out on demand. You will lose points for wasted instances.

   c) The object detection (input, output) stored in S3 should be correct.

5. Policies

1) Late submissions will absolutely not be graded (unless you have verifiable proof of emergency). It is much better to submit partial work on time and get partial credit for your work than to submit late for no credit.

2) Each group needs to work independently on this exercise. We encourage high-level discussions among groups to help each other understand the concepts and principles. However, a code-level discussion is prohibited and plagiarism will directly lead to failure of this course. We will use anti-plagiarism tools to detect violations of this policy.