Location Based Cloud Resource Management for Analyzing Real-Time Videos from Globally Distributed Network Cameras

Anup Mohan, Ahmed S. Kaseb, Yung-Hsiang Lu, Thomas J. Hacker
Purdue University, West Lafayette, IN, USA
{mohan11, akaseb, yunglu, tjhacker}@purdue.edu

***Abstract*—The use of video data for a variety of applications has gained immense popularity. These applications include traffic monitoring, surveillance, retail store management, etc. Thousands of publicly accessible network cameras distributed around the world are potential sources of video data. The applications analyzing data from network cameras have different resource requirements (CPU, memory, etc.) and performance requirements (video frame rate). A preferred way to meet these requirements is to use a cloud infrastructure and a pay-per-use model of the cloud. Cloud computing offers resources, referred to as cloud instances, with different capacities and at different locations. The cost of these instances are dependent on their capacities and locations. The frame rate of the video data obtained from a network camera impacts the accuracy of the analysis and is dependent on the location of the cloud instance. Hence it is important to select the locations of the instances to meet the application performance requirements. This paper presents a resource management approach to select the locations, types, and number of cloud instances to analyze real-time video data from the network cameras while meeting the performance requirements and reducing the analysis cost. We model the resource management problem as a variable size bin packing problem and describe a heuristic algorithm to find a solution. This paper uses Amazon EC2 to evaluate our new resource manager, and observes that our method can reduce the analysis cost up to 56% compared with two other strategies for selecting instance locations.**

1. Introduction

Video streaming and analysis have emerged as an important research area in both academic and industrial fields [12]. Network cameras are a data source of particular interest as they provide valuable real-time information and are globally distributed. Based on research conducted in 2014, there are more than 49 million network cameras active and operational around the world [9]. These include traffic cameras and cameras deployed by various organizations and universities. A wide variety of applications are possible by analyzing the data from network cameras. Surveillance applications analyze data from multiple cameras [19, 22]. Traffic management and monitoring analyzes video data to detect and classify vehicles, track pedestrians, etc. [17, 23]. The video data are also used to manage retail store operations [20].

These applications require significant computational resources. It is common to analyze data for short durations, for example in rush hours only. Hence the pay-per-use cloud pricing model is a preferred solution to meet the resource requirements of these applications. Cloud vendors offer instances at different geographical locations including North America, South America, Europe, Asia, and Australia. The costs of the instances vary depending on the location. Figure 1 shows the cost of Amazon EC2 *m3.2xlarge* instances [1] at different locations and the cost can vary by 47%.

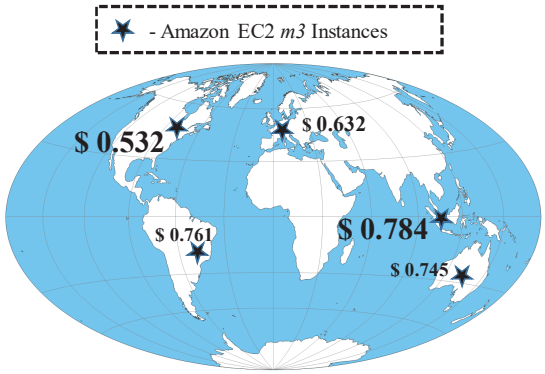

Figure 1. The cost (dollars per hour) of Amazon EC2 *m3.2xlarge* instances at different locations. The instances with the highest and lowest cost are highlighted. The differences are as much as 47% if analyses are performed in the instance with the lowest cost ($0.532/hour) compared with the highest cost ($0.784/hr).

One of the most important requirements for video data analysis is the frame rate, measured in *Frames Per Second (FPS)*. Frame rate depends on the analysis program. For example, tracking moving vehicles requires a higher frame rate compared with tracking pedestrians. The target frame rate has to be maintained as it can impact the accuracy of the analysis (e.g., the number of vehicles detected). Frame rate requirements restrict the maximum distance between the cameras and the instances (measured by Round-Trip Time, RTT) [6]. It is not always possible to select the cheapest instance location. For higher frame rates, the instances should

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

2330-2186/16 $31.00 © 2016 IEEE
DOI 10.1109/CloudCom.2016.37

be closer to the cameras. Hence it is important to select the locations of the cloud instances to meet the analysis requirements.

This paper solves the problem of selecting cloud instance locations at reduced cost for analyzing video data from network cameras. This paper presents a resource manager that models the problem of selecting instance locations as a Variable Size Bin Packing Problem [8] and proposes a heuristic algorithm. We analyze Motion JPEG video data as it is commonly available video data format in network cameras. Amazon EC2 instances are used to evaluate the solution and our method can reduce the analysis cost up to 56% compared with two other location selection strategies. The main contributions of the paper are as follows: (i) mapping the problem of selecting locations as a variable size bin packing problem (ii) proposing a heuristic algorithm to determine the locations, types, and number of instances; and (iii) evaluating our method using Amazon EC2 and demonstrating significant cost reduction compared with two other location selection strategies.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 defines the problem of location selection. Section 4 presents the resource manager and Section 5 evaluates the proposed resource manager. Section 6 summarizes the paper and suggests future work.

2. Related Work

There are related studies dealing with the location selection of cloud instances and the placement of data and applications in distributed cloud networks. Agarwal et al. [3] present an automated data placement method for geo-distributed cloud services that seeks to reduce data-center capacity, inter-datacenter traffic and latency. Tiwana et al. [21] propose to expose the location of data in the network, and moving computation closer to the data (and vice versa) to reduce latencies. Chang et al. [5] study the optimal placement of applications in cloud to meet the latency requirements. Since the data sources considered in this paper are network cameras, the location of the real-time data cannot be changed. Instead, this paper focuses on selecting the locations of cloud instances to reduce the latency and cost.

Other studies consider the placement of web applications in cloud. Malet et al. [15] propose a cloud management middleware to adjust the placement of web applications across the cloud data centers to minimize response time. Luckeneder et al. [14] use the geographical distance, network latency, and HTTP round-trip time to select cloud regions. They execute web service workflows to decrease the execution time. Neither of the studies consider the cost of instances at different locations and only focus on reducing the execution time. Our method aims at reducing the overall cost while satisfying the performance requirements.

Pandey et al. [18] propose a non-linear programming model to minimize the total execution cost for data intensive workflows in distributed cloud services. This paper solves a different problem of reducing the overall cost of real-time analysis on streaming video data. Our method models the relationship between the frame rates, number of streams, and network distances, to select the types, locations, and number of instances.

To facilitate the large scale analysis on image and video data from network cameras, our team has developed a system called Continuous Analysis of Many CAMeras (CAM^2) [10, 11]. CAM^2 provides access to more than 120,000 publicly available network cameras. The problems related to the location selection was first explained in our previous work [6]. Our prior work, Adaptive Resource Management for Video Analysis in Cloud, *ARMVAC* [16], determines the locations, types and number of cloud instances required for a given analysis program and frame rate. ARMVAC determines the locations of instances by dividing the network cameras into regions based on the frame rate for analysis and selecting the instance locations and types separately for each region. Therefore, ARMVAC determines the instance locations locally (within a region). Our resource manager described in this paper builds upon ARMVAC to determine the locations, types, and number of instances for network cameras globally without dividing them into regions and produces better results compared with ARMVAC. Our improved method reduces the cost up to 31% compared with ARMVAC.

3. Problem Definition

Consider the case to analyze data from cameras located in Virginia, Frankfurt, and Tokyo. There are multiple ways to select locations of cloud instances such as, (i) launch instances at the three locations, (ii) launch instances at one of the locations and analyze data from all the cameras, and (iii) launch instances at two of the locations. Our goal is to determine the instance locations with the least cost. In this section we define the problem of selecting locations and identify the factors influencing the cost.

Our previous work [16] demonstrates that the measured frame rate changes based on the network round-trip time (RTT) between the cloud instances and the cameras. When the target frame rates are low, the instances can be selected based on the cost as the instances need not be near the cameras. When the target frame rates are high, the instances should be near the cameras irrespective of the cost. A range of permissible RTT's for a given range of target frames is described in our previous work and is shown in Table 1 for reference.

TABLE 1. THE MAXIMUM RTT FOR DIFFERENT TARGET FRAME RATES AS SHOWN IN OUR PREVIOUS WORK [16]

Frame Rate (F)	RTT (ms)
10 FPS $<$ F $\leq$ 30 FPS	50
5 FPS $<$ F $\leq$ 10 FPS	100
1 FPS $<$ F $\leq$ 5 FPS	200
F $\leq$ 1 FPS	400

Figure 2 adopted from our prior work [6], illustrates the challenges involved in location selection. For each camera

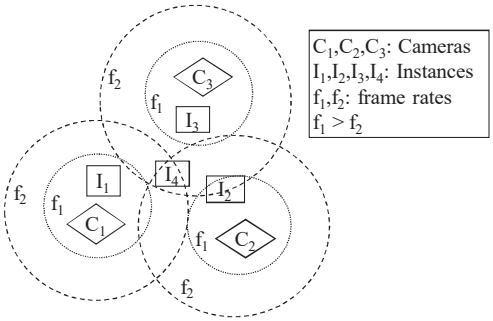

Figure 2. The instance selection is influenced by the frame rate and the network distance (RTT) between the cameras and instances [6]. The range of possible instance locations for cameras at different target frame rates are shown using circles.

and different target frame rates, the valid geographical range of possible instance locations determined using the permissible RTT's are shown using circles. As the network distance between the instance location and the camera increases, the time taken by frames from camera to arrive at the instance increases and the measured frame rate decreases. Hence the range of possible instances are wide for lower frame rates. Since the target frame rate $f_2 < f_1$, the range of possible locations for f_2 is higher than f_1. If the target frame rate is f_1, the instances I_1, I_2, and I_3 must be selected to analyze video data from cameras C_1, C_2, and C_3 respectively, as they are the only instances within the RTT range. If the target frame rate is f_2, then multiple combinations are possible like (I_1, I_2, and I_3), (I_1 and I_2), and (I_4). The combination of instances with the lowest cost should be selected. The analysis cost is determined by the types and number of instances used and is dependent on three factors, (i) the analysis program, (ii) the number of camera streams being analyzed, and (iii) the target frame rate. For example, when the target frame rate is f_2, the cost of the combination I_1, I_2, and I_3 is the sum of the cost of analyzing data from one camera at each instance. Similarly the cost of using I_4 is equal to the cost of analyzing data from 3 cameras at instance I_4. Therefore the location of an instance is determined by: (1) the location of the camera streams analyzed by the instance; (2) frame rate required for the analyses; and (3) the analysis cost at the instance location. This is a simplified example and in most cases the number of cameras are much higher than the number of instance locations which makes the location selection a complex problem.

The problem of location selection can be defined as follows: *Given an analysis program, cameras to be analyzed and their locations, and the target frame rate, how to determine the cloud instance locations such that analysis requirements are satisfied and the overall cost is reduced?*

The video data from the network cameras are available in different data formats. In this paper we focus only on analyzing Motion JPEG (MJPEG) [2] data from cameras, as the MJPEG format is a common video format supported by the network cameras. Even though this paper focuses on the real-time analysis of video data, the location selection problem is relevant for offline analysis as well. Since the location of the instance impacts the frame rate of the video data, to ensure that the video data is being archived at the target frame rate, the locations of instances should be selected based on the RTT. To simplify the problem, we assume that for a given analysis program the data from all the cameras are analyzed at the same frame rate. We do not consider data transfer cost ($0.02 per GB) [1] as they are not significant compared with the instance costs.

TABLE 2. SYMBOL TABLE

Symbol	Definition
T	Set of cameras
L	Set of instance locations
A	Set of candidate cameras of a location
$\mathbb{A}$	Set of candidate cameras for all locations
x_{ij}	Indicator variable to assign the camera to a particular location
V	Set of instance types
$\mathbb{V}$	Set of instance types for all locations
$AC()$	Analysis Cost function
e_j	Analysis cost of location j
$\mathbb{S}$	Solution set
cs	Overall cost of a solution
h	Total number of cameras
w	Total number of locations

4. Location Based Resource Management

We present our location based resource manager in this section. The improvements from our previous work is explained in Section 4.1. Our proposed method of location selection is described in Section 4.2. The symbols used in this paper are listed in Table 2. A set is represented using an uppercase symbol and the elements of a set using a symbol with a subscript. A double struck uppercase symbol (e.g. $\mathbb{A}$) is used to represent a family of sets. Variables are represented using lowercase symbols.

4.1. Improvement from Our Previous Work

Our previous work [16] presents a method named ARMVAC to determine the types and number of instances required for video data analysis. The method takes the analysis program, number of cameras, and target frame rate as inputs, determines the types and number of instances, and outputs the overall cost. ARMVAC maps the problem of resource selection as a variable size bin packing problem. We will use this method to determine the types and number of instances required and the analysis cost for an instance location and will be referred to as *AC()*.

ARMVAC selects the locations first and then determines the types and number of instances. If an instance location is within the RTT range of the camera (Table 1) for a target frame rate, then the instance location is a *candidate location* for the camera. Similarly a *candidate camera* for a location is the camera within the RTT range of the location for a

target frame rate. ARMVAC selects the cheapest candidate locations for all the given cameras and the cameras with the same cheapest candidate location will be assigned to that location.

ARMVAC performs well for low and high target frame rates. But there is an issue with ARMVAC for intermediate frame rates (1 FPS $<$ Frame Rate $<$ 20 FPS). For example, consider 10 cameras in Virginia and 10 cameras in Tokyo and a target frame rate of 5 FPS. The candidate list of Virgnia includes cameras in Virginia, the candidate list of Tokyo includes cameras in Tokyo, and that of Frankfurt includes cameras in both Virginia and Tokyo due to the RTT range at 5 FPS. ARMVAC selects the cheapest location for cameras in Virginia to be Virginia, and for the cameras in Tokyo to be Frankfurt. Instances will be launched at Virginia and Frankfurt separately. Frankfurt is also a candidate location for cameras in Virginia for the given frame rate. Hence it is possibile to analyze data from all the cameras at Frankfurt at a lower cost compared with the cost of analyzing data separately at Virginia and Frankfurt. The number of *candidate cameras* from a location and the cost of analyzing these cameras are important factors to consider while selecting the locations. ARMVAC does not consider these factors and may not be able to produce cost-efficient results for intermediate frame rates.

To overcome this problem we propose a new method called the *Globally Cheapest Location (**GCL**)*. GCL defines the cost of instances at each location as the overall cost of analyzing data at the target frame rate from all the candidate cameras of that location. It maps the problem of selecting instance location to the Variable Size Bin Packing Problem (VSBPP) [8] and uses a heuristic algorithm to determine the types, locations, and number of cloud instances to reduce the overall cost. One key difference of our proposed method GCL compared with ARMVAC is that the locations are selected iteratively along with the types and number of instances.

4.2. Globally Cheapest Location (GCL)

This section describes the two stages of GCL. The first stage models the location selection problem as a variable size bin packing problem. The second stage proposes a heuristic algorithm based on the Adapted Best First Decreasing approach [7].

4.2.1. Location Selection as a Variable Size Bin Packing Problem. The inputs to the model are: (i) a set of cameras; (ii) the locations of instances; (iii) the types of instances at each location; and (iv) the set of candidate cameras of each location.

The number of candidate cameras of an instance location can vary from zero (no cameras within the RTT range) to the total number of cameras. A possible solution is a set of locations such that each camera should be assigned to one and only one of the locations in the solution.

The cost of using an instance location depends on two factors: (1) the number of candidate cameras of the location; and (2) the types and number of instances required to analyze data from the given number of cameras. For a given solution the overall cost can be determined by adding the costs of instances present in the solution using these two factors. Multiple solutions are possible for this problem. The goal is to find the solution with the minimum overall cost.

For example, consider the case of analyzing 20 cameras at a target frame rate of 5 FPS. Let the set of cameras be {cam1, cam2, ..., cam20} and the set of locations be {Virginia, Tokyo, Frankfurt}. Cameras 1-10 are candidate cameras of Virginia and Frankfurt, and cameras 11-20 are candidate cameras of Tokyo and Frankfurt for the given target frame rate. The set of candidate cameras for each location is {{cam1, ..., cam10}, {cam11, ..., cam20}, {cam1, ..., cam20}} in the same order as the set of locations. The set of types of instances available at all locations is {{m3.2xlarge, m3.large},{m3.2xlarge, m3.large},{m3.2xlarge, m3.large}}. The number of candidate cameras of each location is 10, 10, and 20 respectively. Consider the case that all 20 cameras can be analyzed using one instance of both the types (possible for analyses like face detection). In this example m3.large is selected as the instance type as it is cheaper. The cost of the locations in the same order as the set of locations are {\$0.133/hour, \$0.196/hour, \$0.158/hour}. Three solutions are possible in this case: {Virginia, Frankfurt}, {Virginia, Tokyo}, and {Frankfurt}. The cost of each solution is {\$0.291/hour, \$0.329/hour, \$0.158/hour} respectively. In the first solution cameras 1-10 are assigned to Virginia and cameras 6-10 are assigned to Tokyo. The third solution assigns all cameras to Frankfurt. The third solution has the minimum cost and should be selected.

The problem can be modeled as follows. Let $T = \{t_1, t_2, ..., t_h\}$ be the set of all cameras. Let the set of all instance locations be given by $L = \{l_1, l_2, ..., l_w\}$. For an instance location l_j, let $A = \{t_1, t_2, ..., t_h\}$ be the set of candidate cameras of l_j. The set of candidate cameras for all the locations is given by $\mathbb{A} = \{A_1, A_2, ..., A_w\}$.

Let x_{ij} be an indicator variable to keep track of the location assignment for cameras such that x_{ij} is 1 if and only if camera i is assigned to location j. For a given camera x_{ij} should be one 1 for exactly one location.

Let $V = \{v_1, v_2...\} = \{m3.2xlarge, m3.xlarge, ..\}$ be the different types of instances. The set of different types of instances for all the locations is given by $\mathbb{V} = \{V_1, V_2, ..., V_w\}$. For a given location j, the cost of analyzing data from the given number of cameras is calculated using the function *AC(V_j, number of cameras assigned)* as explained in Section 4.1. The number of cameras assigned to a location can be determined by adding all the x_{ij} values for that location. The cost of selecting instances at location j is given by:

$$e_j = AC(V_j, \sum_{i=1}^{h} x_{ij}) \quad (1)$$

Let a possible solution be $\mathbb{S} = \{A_1, A_3, ..., A_w\}$. The overall cost of a solution $\mathbb{S}$, cs, is the sum of costs of instances (e_j) for all the locations in the solution.

The number of possible solutions is finite as the number of cameras and the instance locations is also finite. The solution with minimum overall cost is given by:

$$\mathbb{S}_y, \text{ such that } y = \underset{\forall \text{ possible } \mathbb{S}}{\operatorname{argmin}} \; cs \tag{2}$$

The formulated problem is a variant of Variable Size Bin Packing Problem. The parameters of the VSBPP can be mapped to the formulated problem as follows.

- The cloud instance locations used (l_j) are the equivalent of the bins.
- The size of the bin is represented by the maximum number of cameras ($|\mathbb{A}_j|$) from which the data can be analyzed by the instance location l_j.
- The bin cost is equivalent to the cost (e_j) of selecting cloud instances at location l_j.

4.2.2. Algorithm to Select the Locations. Our approach (GCL) is based on solving the variable size bin packing problem described previously. GCL uses a heuristic algorithm based on *Adapted Best First Decreasing* (A-BFD) [7]. A-BFD sorts the bins based on the ratio of the cost and the size of the bins. GCL sorts the instance locations based on the ratio of the cost of selecting the location (e_j) and the number of candidate cameras of the location ($|\mathbb{A}_j|$). The best instance locations are those with lower ratio values.

The outline of the heuristic algorithm is shown in Figure 3. Consider the same example as given in Section 4.2.1. The elements of the set $\mathbb{A}$ are {{cam1, ..., cam10}, {cam11, ..., cam20}, {cam1, ..., cam20}}. The weight of each element is 10, 10, and 20 respectively. The cost of each element is \$0.133/hour, \$0.196/hour, and \$0.158/hour respectively. The elements of the set $\mathbb{A}$ are sorted based on the ratio of its cost and the number of cameras. The sorted $\mathbb{A}$ is {{cam1, ..., cam20}, {cam1, ..., cam10}, {cam11, ..., cam20}}. After sorting, the best location ($\mathbb{A}_1$) is added to the solution. For the example $\mathbb{A}_1$ = {cam1, ..., cam20} corresponds to Frankfurt.

The cameras in $\mathbb{A}_1$ are assigned to its corresponding location by setting the indicator variable x_{ij}. The indicator variables for this example are {{0, 0, 1}, {0, 0, 1}, ..., {0, 0, 1}} as Frankfurt is the selected location. The cameras assigned to $\mathbb{A}_1$ are removed from the candidate list of other elements of $\mathbb{A}$. The element $\mathbb{A}_1$ is removed from $\mathbb{A}$ for the next iteration. In the example since all cameras are assigned, $\mathbb{A}$ becomes empty after this step. To illustrate this step clearly, consider $\mathbb{A}_1$ to be {cam1, ..., cam10}. This means that cameras 1 - 10 are assigned to Virginia in the solution and the indicator variables become {{1, 0, 0}, {1, 0, 0}, ..., {1, 0, 0}, {0, 0, 0}..., {0, 0, 0}}. After removing the assigned cameras from the candidate list and removing $\mathbb{A}_1$, $\mathbb{A}$ becomes {{cam11, ..., cam20}, {cam11, ..., cam20}}. Note that the candidate cameras of Frankfurt changes from {cam1, ..., cam20} to {cam11, ..., cam20} as cameras 1-10 are assigned to Virginia. The process is repeated until all the given cameras are assigned to the locations in the solution.

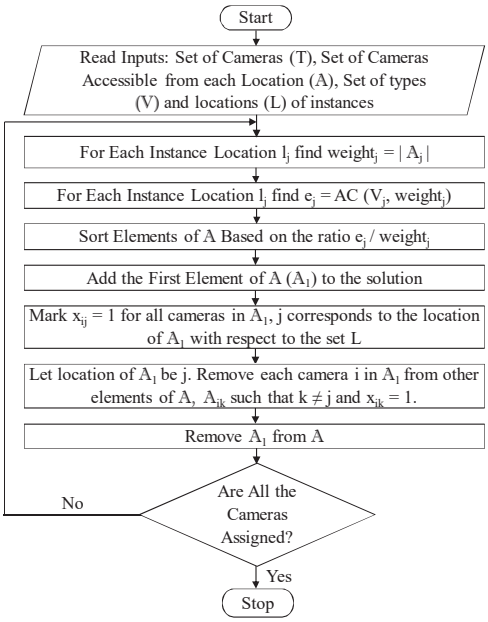

Figure 3. Flow Diagram of GCL

5. Evaluation

We evaluate our solution ***GCL*** by changing the target frame rates, analysis programs, distribution of cameras among the locations, and the number of cameras. We also compare our solution with the optimal solution. The network cameras used for evaluation are distributed across North America, Europe, and Asia. Amazon EC2 *m3* cloud instances are used for the evaluation. The cloud instances are launched at the selected locations and the MJPEG data from the network cameras are analyzed and frame rates are measured. The instances located at *Virginia (**V**), Oregon (**O**), Frankfurt (**F**), and Tokyo (**T**)* are used.

The results are compared against another location selection strategy where the instance location nearest to the camera is always selected. This strategy will be referred to as *Nearest Location (**NL**)*. For example, to analyze two cameras in Virginia and Oregon, the strategy NL would always launch instances at Virginia and Oregon separately irrespective of the frame rate. The results are also compared with our previous work ***ARMVAC***. Another strategy where the cheapest instance location is always selected irrespective of the distance is not considered as it may not meet the performance requirements at high frame rates. Three different analysis programs used for evaluation are as follows:

- Motion Estimation (ME): Background subtraction is used to estimate the amount of motion.
- Face Detection (FD): Based on Haar Feature based cascade classifiers.
- SIFT (SIFT): Scale invariant features [13] are extracted from the image.

TABLE 3. COMPARISON OF LOCATIONS SELECTED FOR DIFFERENT FRAME RATES

FPS	NL	ARMVAC	GCL
0.5	V,O,F,T	V	V
1	V,O,F,T	V	V
5	V,O,F,T	V,O,F	F
10	V,O,F,T	V,F	O,F
15	V,O,F,T	V,T	V,O,T
20	V,O,F,T	V,O,F,T	V,O,F,T
30	V,O,F,T	V,O,F,T	V,O,F,T

TABLE 4. SETUP AND RESULTS FOR EVALUATION WITH DIFFERENT LOCATIONS OF NETWORK CAMERAS. WE USE 20 CAMERAS AND THE PROGRAM "ME". LOCATIONS OF INSTANCES ARE VIRGINIA (**V**), OREGON (**O**), FRANKFURT (**F**), AND TOKYO (**T**).

Test Cases	Input				Output								
	Number of Cameras				Strategy	Number of Cameras Assigned for 5 FPS				Number of Cameras Assigned for 10FPS			
	E. US	W. US	Europe	Asia		V	O	F	T	V	O	F	T
Case 1	5	7	6	2	NL	5	7	6	2	5	7	6	2
					ARMVAC	6	7	7	0	5	7	8	0
					GCL	0	0	20	0	5	0	15	0
Case 2	2	2	14	2	NL	2	2	14	2	2	2	14	2
					ARMVAC	7	3	10	0	2	2	16	0
					GCL	0	0	20	0	0	0	20	0
Case 3	9	3	6	2	NL	9	3	6	2	9	3	6	2
					ARMVAC	16	2	2	0	16	1	3	0
					GCL	16	0	4	0	16	0	4	0

These programs are provided by the OpenCV library [4]. In all our evaluations, the measured frame rates are equal to the target frame rates.

5.1. Different Frame Rates

We change the target frame rate from 0.5 FPS to 30 FPS and calculate the analysis cost using GCL, ARMVAC, and NL. The data from 10 cameras are analyzed using ME. Among the cameras three are located in East U.S., three in Europe, and two each in West U.S. and Asia.

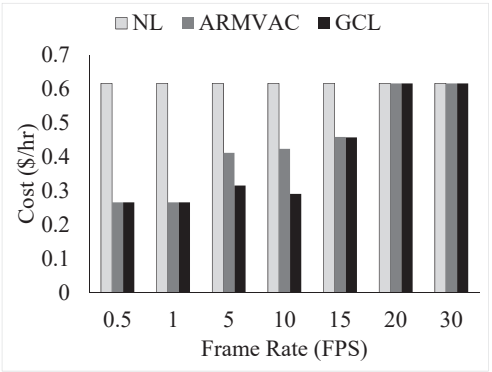

Figure 4. Cost comparison of Globally Cheapest Location (GCL) with Nearest Location (NL) and our prior work, ARMVAC for different frame rates. GCL can reduce cost up to 56% and 31% compared with NL and ARMVAC respectively.

Figure 4 shows the results. The instances selected by each method are shown in Table 3. For 0.5 FPS and 1 FPS, GCL and ARMVAC reduces the cost by 56% compared with NL as all the analysis is performed using the Virginia instance. For 5 FPS, GCL consolidates all of the computations to Frankfurt and hence reduces the cost by 48% and 23% compared with NL and ARMVAC respectively. In case of 10 FPS, ARMVAC assigns 8 cameras to an m3.xlarge instance (4 cores) at Virginia and 2 cameras to an m3.large instance (2 cores, half the price of m3.xlarge) at Frankfurt. But GCL assigns 5 cameras each to an m3.large instance at Oregon and Frankfurt and hence reduces the cost by 31% compared with ARMVAC. GCL reduces the cost by 48% compared with NL for 10 FPS. Similarly for 15 FPS, by dividing the cameras among instance locations and therefore using cheaper instance types, GCL reduces the cost by 25% compared with NL and 0.2% compared with ARMVAC. For frame rates higher than 15 FPS, the instances should be closer to the cameras and hence the locations selected by ARMVAC and GCL are same as that of NL and hence no cost reduction is provided.

5.2. Different Analysis Programs

To study the impact of the analysis programs on location selection, we use the same 10 cameras as before. The results for only 10 FPS are shown as the other frame rates also produce similar results. We use FD and SIFT in addition to ME for the evaluation.

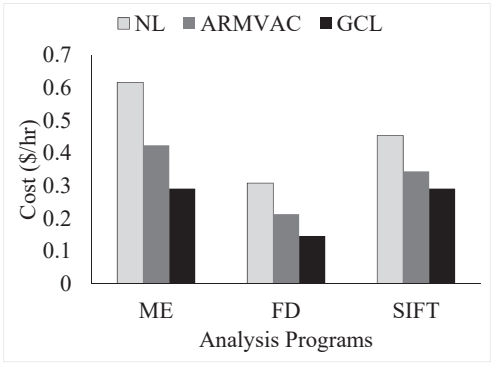

Figure 6. Cost comparison of GCL with NL and ARMVAC for different analysis programs. GCL can reduce cost up to 52% and 31% compared with NL and ARMVAC respectively. Location selection does not depend on the analysis program.

Figure 6(a) shows the results. For SIFT, GCL reduces the cost by 35% and 15% and for FD, GCL reduces the cost by 52% and 31% compared with NL and ARMVAC respectively. The locations selected by GCL are the same for all three analysis programs, as the analysis programs only affect the cost and not the location selection.

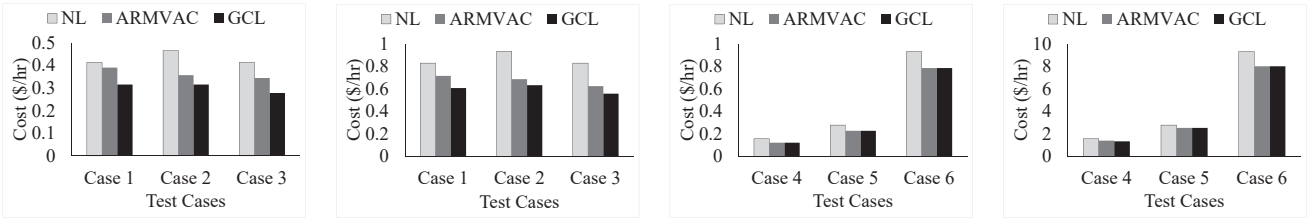

Figure 5. Cost comparison of Globally Cheapest Location (GCL) with Nearest Location (NL) and our prior work, ARMVAC for: (a) different distribution of cameras for a target frame rate of 5 FPS, GCL can reduce cost up to 32% and 19% compared with NL and ARMVAC respectively; and (b) different distribution of cameras for a target frame rate of 10 FPS, GCL can reduce cost up to 32% and 15% compared with NL and ARMVAC respectively. (c) different number of cameras for a target frame rate of 1 FPS, GCL can reduce cost up to 22% compared with NL; and (d) different number of cameras for a target frame rate of 10 FPS, GCL can reduce cost up to 14% and 4% compared with NL and ARMVAC respectively.

5.3. Different Distribution of Cameras among Locations

Another factor that determines the location selection is the distribution of the network cameras among the locations. We evaluate this by changing the number of cameras selected for analysis from a particular location. Two target frame rates 5 FPS and 10 FPS are used for this evaluation. Table 4 shows the configurations for this evaluation.

Figure 5(a) and Figure 5(b) shows the results for the target frame rate of 5 FPS and 10 FPS respectively. The instances selected for each case are shown in Table 4. As it can be seen the distribution of cameras among locations impacts the location selection and the analysis cost. For *Case 1* at 5 FPS, GCL reduces the cost by 23% and 19% compared with NL and ARMVAC respectively. Similarly for *Case 2* GCL reduces the cost by 32% and 11%. In both cases the candidate list of Frankfurt has all the cameras, and therefore GCL assigns all cameras to Frankfurt providing the cost reduction. For *Case 3*, GCL reduces the cost by 32% compared with NL and 19% compared with ARMVAC. In this case the Virginia instance is preferred as its candidate list has the maximum number of cameras and is one of the cheapest instances. The cost difference between GCL and ARMVAC in this case is because GCL assigns 4 cameras to Frankfurt but ARMVAC assigns two each to Oregon and Frankfurt. For 10 FPS, the candidate cameras of an instance changes resulting in a different distribution of cameras among locations and hence different costs. For example, in *Case 3* ARMVAC assigns 1 camera to Oregon and 3 cameras to Frankfurt. A single core instance can now be used at Oregon instead of a two core instance as used for 5 FPS. GCL reduces the cost by 26%, 32%, and 32% compared with NL for all cases. Compared with ARMVAC, GCL reduces the cost by 15%, 7%, and 10% for all the cases. The analysis cost for 10 FPS is higher than that of 5 FPS as more resources are required.

5.4. Different Number of Cameras

We evaluate GCL for 50, 100, and 350 cameras and compare the results. The analysis program ME is used at a low frame rate, 1 FPS and an intermediate frame rate, 10 FPS. The configurations for this evaluation and the locations selected are described in Table 5.

TABLE 5. SETUP AND RESULTS FOR EVALUATION WITH DIFFERENT NUMBER OF CAMERAS. LOCATIONS OF INSTANCES ARE VIRGINIA (**V**), OREGON (**O**), FRANKFURT (**F**), AND TOKYO (**T**).

Test Cases	Input					Output				
	Number of Cameras					Strategy	Number of Cameras Assigned for 10 FPS			
	Total	E. US	W. US	Europe	Asia		V	O	F	T
Case 4	50	1	7	37	5	NL	1	7	37	5
						ARMVAC	13	10	27	0
						GCL	12	11	27	0
Case 5	100	30	7	58	5	NL	30	7	58	5
						ARMVAC	63	10	27	0
						GCL	63	10	27	0
Case 6	350	145	49	179	4	NL	145	49	179	4
						ARMVAC	318	10	22	0
						GCL	318	24	8	0

Figure 5(c) and 5(d) shows the results for 1 FPS and 10 FPS respectively. For 1 FPS, the candidate lists of the given locations have all the cameras. Hence GCL and ARMVAC will assign all the cameras to the Virginia instances. GCL reduces the cost by 22%, 18%, and 15% compared with NL for cases 4, 5, and 6 respectively. For *Case 4* at 10 FPS, GCL reduces the cost by 14% and 4% compared with NL and ARMVAC. ARMVAC assigns 13 cameras to Virginia and 10 to Oregon whereas GCL assigns 12 cameras to Virginia and 11 to Oregon. Even though the total number of cameras assigned to Virginia and Oregon are same for both the methods, the combination selected by GCL results in lower cost due to the types of instances selected. For cases 5 and 6, GCL reduces the cost by 9% and 14% compared with NL. Due to distribution of the given cameras, GCL and ARMVAC have the same analysis cost for cases 5 and 6.

5.5. Comparison with the Optimal Solution

We evaluate GCL by comparing it with the optimal solution determined by exhaustively searching all possible combinations of the locations and the cameras. The execution time for the optimal solution is an order of magnitude

higher than GCL and ARMVAC. We observe that for the evaluation scenarios described in this section, GCL selects the locations at the same cost as the optimal solution. We also observe some cases where the analysis cost of GCL is higher than the optimal solution. Table 6 describes these cases by comparing the instances selected by GCL, ARMVAC, and the optimal solution. For *Case 7*, GCL assigns 36 cameras to Virginia in the first iteration. In the second iteration, since the cost to weight ratio of Frankfurt is lesser than Oregon, Frankfurt is selected. But the optimal solution requires Oregon to be selected. Since ARMVAC assigns the cameras to the three locations in the beginning itself, the analysis cost is equal to the optimal cost. A similar explanation can be applied to *Case 8* as well. The cost difference between GCL and the optimal solution is 0.9% and 2% for cases 7 and 8. We plan to overcome this by implementing a post-processing procedure [7] that searches for other possible low cost combinations.

TABLE 6. COMPARISON OF GCL WITH THE OPTIMAL SOLUTION (OS)

Test Cases	Strategy	Number of Cameras					Cost ($/hr)	Execution Time (ms)
		Total	V	O	F	T		
Case 7	OS	50	36	1	13	0	1.26	209.554
	GCL	50	36	0	14	0	1.272	0.391
	ARMVAC	50	36	1	13	0	1.26	0.151
Case 8	OS	100	63	10	27	0	2.519	13214.106
	GCL	100	2	59	39	0	2.569	0.664
	ARMVAC	100	63	10	27	0	2.519	0.211

Our evaluation shows that GCL can reduce the cost up to 56% while selecting locations compared with NL and up to 31% compared with ARMVAC.

6. Conclusion

In this paper we present a resource manager to select the locations, types, and number of instances at reduced cost for analyzing real-time video data from network cameras. The problem of selecting locations is mapped as a variable size bin packing problem. A new heuristic algorithm is proposed to solve the problem. We use Amazon EC2 to evaluate our solution and demonstrate up to 56% cost reduction compared with two other location selection strategies. We would like to extend this work by handling the case where the target frame rates will be different for the individual network cameras. We also plan to perform this study on H.264 video format.

Acknowledgments

This project is supported in part by National Science Foundation ACI-1535108, CNS-0958487, and IIP-1530914. The authors would like to thank Amazon and Microsoft Azure for providing the cloud infrastructure, and the organizations that provide the camera data. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing/.

[2] Motion JPEG Video Codec. http://www.digitalpreservation.gov/formats/fdd/fdd000063.shtml.

[3] S. Agarwal et al. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*, pages 17–32, 2010.

[4] G. Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11):120, 122–125, 2000.

[5] F. Chang et al. Placement in clouds for application-level latency requirements. In *2012 IEEE International Conference on Cloud Computing (CLOUD)*, pages 327–335.

[6] W. Chen et al. Analysis of large-scale distributed cameras using the cloud. *Cloud Computing, IEEE*, 2(5):54–62, 2015.

[7] Crainic et al. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38(11):1474–1482, 2011.

[8] D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM journal on computing*, 15(1):222–230, 1986.

[9] N. Jenkins. 245 million video surveillance cameras installed globally in 2014. *IHS Technology*.

[10] A. S. Kaseb et al. A system for large-scale analysis of distributed cameras. In *IEEE Global Conference on Signal and Information Processing*, pages 340–344, 2014.

[11] A. S. Kaseb et al. An interactive web-based system for large-scale analysis of distributed cameras. In *Imaging and Multimedia Analytics in a Web and Mobile World*, 2015.

[12] B. Li et al. Two decades of internet video streaming: A retrospective view. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1s):33, 2013.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004.

[14] M. Luckeneder and A. Barker. Location, location, location: Data-intensive distributed computing in the cloud. In *2013 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, volume 1, pages 647–654.

[15] B. Malet and P. Pietzuch. Resource allocation across multiple cloud data centres. In *Proceedings of the International Workshop on Middleware for Grids, Clouds and e-Science*, page 5. ACM, 2010.

[16] A. Mohan et al. Adaptive resource management for analyzing video streams from globally distributed network cameras. *Submitted for Publication*.

[17] Naito et al. Robust license-plate recognition method for passing vehicles under outside environment. *IEEE Transactions on Vehicular Technology*, 49(6):2309–2319, 2000.

[18] S. Pandey et al. Minimizing execution costs when using globally distributed cloud services. In *2010 IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 222–229.

[19] Regazzoni et al. Video analytics for surveillance: Theory and practice. *Signal Processing Magazine, IEEE*, 2010.

[20] Senior et al. Video analytics for retail. In *Advanced Video and Signal Based Surveillance*, pages 423–428, 2007.

[21] B. Tiwana et al. Location, location, location!: modeling data proximity in the cloud. In *ACM SIGCOMM Workshop on Hot Topics in Networks*, page 15, 2010.

[22] M. M. Trivedi et al. Distributed video networks for incident detection and management. In *Proceedings of IEEE Intl Conference on Intelligent Transportation Systems*, pages 155–160, 2000.

[23] Zhang et al. Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras. *Transportation Research Record: Journal of the Transportation Research Board*, 1993(1):138–147, 2007.