This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

Adaptive Resource Management for Analyzing Video Streams from Globally Distributed Network Cameras

Anup Mohan, Ahmed S. Kaseb, Yung-Hsiang Lu, Thomas J. Hacker

Abstract—There has been tremendous growth in the amount of visual data available on the Internet in recent years. One type of visual data of particular interest is produced by network cameras providing real-time views. Millions of network cameras around the world continuously stream data to viewers connected to the Internet. This data may be used by a wide variety of applications such as enhancing public safety, urban planning, emergency response, and traffic management which are computationally intensive. Analyzing this data requires significant amounts of computational resources. Cloud computing can be a preferred solution for meeting the resource requirements for analyzing these data. There are many options when selecting cloud instances (amounts of memory, number of cores, locations, etc.). Inefficient provisioning of cloud resources may become costly in pay-per-use cloud computing. This paper presents a method to select cloud instances in order to meet the performance requirements for visual data analysis at a lower cost. We measure the frame rates when analyzing the data using different computer vision methods and model the relationships between frame rates and resource utilizations. We formulate the problem of managing cloud resources as a Variable Size Bin Packing Problem and use a heuristic solution. Experiments using Amazon EC2 validate the model and demonstrate that the proposed solution can reduce the cost up to 62% while meeting the performance requirements.

Index Terms—cloud computing, video streaming, MJPEG, resource management, cost reduction

1 Introduction

The use of visual data such as images and videos for scientific analysis to solve real-world problems has been increasing significantly over the past decade. Network cameras are of particular interest as they generate continuous real-time video data with rich and versatile content [31]. Millions of network cameras are deployed every year [24]. The video analysis market is rapidly growing and is estimated to be worth more than $1.2 billion by the year 2017 [6]. A wide variety of applications such as improving public safety [21], aiding emergency response [14], and surveillance [28] may use the large volumes of visual data. The network cameras considered in this paper consist of both indoor and outdoor cameras including traffic cameras, cameras inside shopping malls, and other institutions. These are public cameras providing free access and owned by different organizations. Modifying any configuration settings of the network cameras is impossible.

These applications may require (1) high resolution video data, (2) analysis for long durations, (3) data from multiple cameras, and (4) high frame rates. These requirements represent "big data" problems that require analysis of large amounts of visual data which needs substantial amounts of computational resources. Cloud computing has the potential to meet these resource needs by selecting many cloud instances (i.e., virtual machines, VMs) containing more cores and large memory. Many applications require streaming data from network cameras around the world, for example, studying the traffic pattern of cities, analyzing global fashion trends, and monitoring weather conditions at different regions. The distance between the network camera and cloud instance can affect the performance of the analysis. Hence there is a need to efficiently stream the data from multiple sources at different geographical locations. Cloud vendors offer many types of VM instances: with different number of cores, memory capacities, and geographical locations. The "pay-per-use" pricing model encourages the use of only a few cloud instances with small number of cores and less memory. The computational requirements of the applications may vary depending on the time of the day and the content of the scene being analyzed. The different competing factors mentioned above make resource management a challenging problem. Few studies have been devoted to selecting the most efficient cloud instances to analyze many video streams at low monetary costs. These studies do not consider the effects of the different types and locations of the instances on the overall cost and performance of the analysis.

This paper presents a method called *Adaptive Resource Management for Video Analysis in Cloud* (***ARMVAC***). ARMVAC determines the configurations (types, locations, and numbers) of cloud instances needed to meet the performance requirements at low costs. We consider Motion JPEG (MJPEG) [4] as the format of video data because most network cameras support MJPEG streaming (some newer network cameras also support H.264). ARMVAC considers the network distances (measured by the round-trip time, RTT) between cameras and cloud instances. ARMVAC models the relationships of the frame rates and CPU utilization

• *A. Mohan is with Intel Corporation Santa Clara, CA, USA. A. S. Kaseb is with Cairo University, Giza, Egypt. Y.-H. Lu, and T. J. Hacker are with Purdue University, West Lafayette, IN, USA.*
Email: anup.mohan@intel.com, akaseb@eng.cu.edu.eg, {yunglu, tjhacker}@purdue.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

on different types of cloud instances. We develop our model through the analysis of three different computer vision methods provided by the OpenCV library [7]: People Detection, Edge Detection, and Color Histogram. We model the problem of selecting cloud instances at low costs as a *Variable Size Bin Packing Problem* (VSBPP) and use a heuristic algorithm [12] to find a solution. ARMVAC predicts the maximum number of streams from cameras to be analyzed on different cloud instances for the given analysis programs. Our solution can dynamically adapt to the varying resource requirements of analysis programs running for long durations. ARMVAC monitors the utilization of the cloud resources at regular intervals and automatically scales the number of resources based on the utilization and performance requirements. ARMVAC is evaluated using Amazon Elastic Compute Cloud (EC2) instances [2]. Three analysis programs: Motion Estimation, Face Detection, and SIFT feature extraction are used for evaluation. The method can achieve the required frame rates and save up to 62% cost compared with four other cloud resource selection strategies.

This paper makes the following contributions: (1) It is one of the first papers devoted to selecting the cloud configurations for analyzing large (GB) amounts of data from multiple video streams. The sources of the streams are globally distributed. (2) Our method considers both performance requirements and costs, modelling this problem as a bin packing problem and using a heuristic solution. (3) The paper presents a prediction model based on CPU utilization for determining the number of streams that can be analyzed on different types of cloud instances for a given analysis program. (4) We evaluate the solution using Amazon EC2 and demonstrate up to 62% cost reduction compared with four other strategies for selecting cloud instances.

The rest of the paper is organized as follows. Section 2 describes the related work and compares ARMVAC with the existing studies. Section 3 explains the relationships among frame rates, locations of cloud instances, and types of cloud instances. Section 4 models the problem of selecting low-cost cloud instances while meeting the required frame rates as a Variable Size Bin Packing Problem and describes our heuristic algorithm to solve the problem. Section 5 evaluates ARMVAC using Amazon EC2. Section 6 concludes the paper and suggests directions for future work.

2 Related Work

Several papers discuss the issues related to streaming media content residing on the cloud for uses such as Video on Demand (VoD). For example, Cloudmedia [34], optimizes the cloud resources based on the server utilization and network bandwidth. Wang et al. [33] discuss data streaming with optimizing cloud resources based on cost and availability. They consider the locations of data sources. They focus only on media delivery and hence do not consider utilization and performance issues while streaming and analyzing video data flowing into the cloud. These studies focus on streaming data *out from* the cloud instances. This paper solves a different problem: streaming video data *into* cloud instances for analysis. The data come from globally distributed network cameras.

2.1 Prediction Based Resource Management

Plenty of research work exists on developing prediction models for cloud resource provisioning. Islam et al. [17] develop statistical models to predict the surge in resource requirements. Khan et al. [20] use a multiple time series to predict workload variations in a cluster of virtual machines. Xiao et al. [35] present a system which dynamically allocates resources based on the past behavior of the virtual machines. Van et al. [27] propose an autonomic resource manager for service hosting platforms using a constraint based programming approach. Most of these studies use past information to predict future resource usage and perform resource management. Each analysis program has a unique resource usage pattern. Obtaining the past resource usage information for all analysis programs is not feasible. Our solution does not use past information and instead performs resource management based on the current resource usage. Our method allocates resources and adaptively adjusts to the change in resource consumption.

2.2 Dynamic Resource Provisioning Methods

Many studies investigate resource allocation on the cloud for a variety of applications. Vijaykumar et al. [32] consider dynamic resource provisioning for data streaming. They do not consider analyzing video data from network cameras. In contrast, this paper considers how to analyze video streams from network cameras.Sharma et al. [29] use an integer linear programming formulation to select cloud configurations. Their model handles tasks with small uniform data sizes and our solution can handle tasks of different sizes.

Hossain et al. [16] propose a heuristic resource management for cloud based video surveillance systems. Song et al. [30] propose a queuing based resource management for multimedia applications. The main objective for both methods is to reduce the service wait time and long term service cost to satisfy the QoS requirements. They reduce the monetary cost by reducing the running time of the tasks. Our solution reduces the overall cost by selecting cost-efficient cloud instance types and locations. We model the performance in terms of frame rate and not the running time as the tasks we consider may run for any duration.

Hossain et al. [15] discuss resource allocation for service composition in cloud based video surveillance applications. They minimize the number of cloud instances used for service composition by modelling the problem as a multi-dimensional bin packing problem. Their method does not reduce the overall cost. Our solution considers multiple analysis programs of different characteristics analyzing data at different frame rates. The solution by Hossain et al. [15] is evaluated by simulation, but our method is evaluated using a commercial cloud vendor—Amazon EC2.

2.3 Amazon Auto Scaling

Amazon EC2 provides auto scaling [1] to add or remove cloud instances based on utilization. Auto scaling lets users select a group of instances and scale them up or down based on use. It does not consider the overall cost. Auto scaling is tied to a particular region and cannot handle the effect of network delay on the input data as explained in Section 3.3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

TABLE 1: COMPARISON OF ARMVAC WITH RELATED WORK

	Data Direction with the Cloud Instance	Selects Different Types of Instances	Selects Instances at Different Locations	Evaluation Method	Use Globally Distributed Data Sources	Reduces Overall Cost
Auto Scaling [1]	N/A	No	No	N/A	N/A	No
Cloudmedia [34]	OUT	No	No	Simulation	No	Yes
Wang et al. [33]	OUT	No	Yes	Simulation	Yes	Yes
Xiao et al. [35]	N/A	No	No	Simulation	N/A	No
Van et al. [27]	N/A	No	No	Simulation	N/A	No
Vijaykumar et al. [32]	IN	No	No	Xen Virtual Environment	Yes	No
Hossain et al. [15]	IN	No	No	Simulation	Yes	No
ARMVAC	**IN**	**Yes**	**Yes**	**Amazon EC2**	**Yes**	**Yes**

TABLE 2: COMPARISON OF ARMVAC WITH OUR PREVIOUS WORK

	Adaptive Resource Manager	Selects Different Types of Instances	Selects Instances at Different Locations	Cost Optimization Method	Speedup Resource Allocation by Predicting the Maximum Number of Streams that can be Analyzed on Instances
Kaseb et al. [19]	Yes	No	No	Heuristic	No
Chen et al. [9]	Yes	No	No	Heuristic	No
ARMVAC	**Yes**	**Yes**	**Yes**	**VSBPP [13]**	**Yes**

Our method determines the types and locations of instances best suited for the given application such that the overall cost is low.

2.4 Improvements from Our Previous Work

This paper extends our previous work [10, 18, 31] that builds a software infrastructure using cloud computing to analyze visual data from thousands of network cameras. The system is referred to as Continuous Analysis of Many CAMeras (CAM^2). This work adds adaptive resource management of cloud instances for analyzing video streams. This paper extends our work [9] which gradually adds resources based on utilization and our work [19] on resource management by finding cost effective instances. This paper is an improvement over our work [25] which primarily focuses on selecting cloud instance locations. This paper implements a resource manager which considers the location of the cloud instances and cameras. The resource manager predicts the number of streams that can be analyzed using a given program on different types of cloud instances to perform resource allocation faster. The resource manager selects cloud instances with different types and locations at lower cost.

Table 1 compares ARMVAC with related work. Table 2 compares ARMVAC with our previous work. Readers are encouraged to visit the CAM^2 website at https://cam2.ecn.purdue.edu/ and register to explore this system.

2.5 Variable Size Bin Packing Problem

ARMVAC models the problem of selecting cloud instances at low cost as a Variable Size Bin Packing Problem (VSBPP) [13]. It considers a collection of bins of finite sizes. The number of bins of each size is unlimited. Each bin has a cost associated proportional to its size. The objective of VSBPP is to pack a list of items into bins so as to minimize the total cost. This is an NP-Hard problem [13]. There are different heuristic algorithms to solve VSBPP that includes First Fit algorithm, Best Fit algorithm, and Best Fit Decreasing algorithm. [23]. This paper considers a variant of the Adapted-Best First Decreasing (A-BFD) algorithm [12] as it is reported to perform better than the other methods. A-BFD sorts the bins according to the ratio of their costs and sizes. The best bin is the one with the least cost-size ratio. We consider the cloud instances to be equivalent to the bins in a VSBPP and the number of streams that can be analyzed on an instance to be equivalent to the size of the bin.

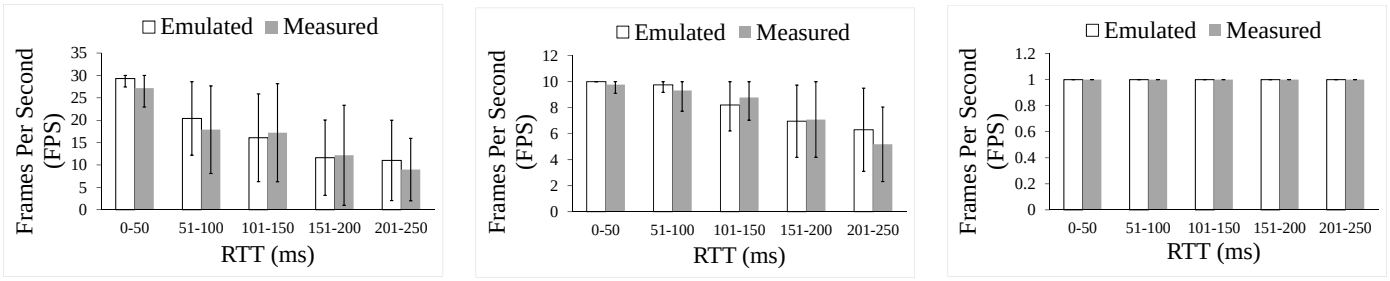
3 Problem Formulation

This paper solves the problem of resource allocation using cloud instances for analyzing video streams from many geographically distributed network cameras. This section formulates the problem based on experimental data obtained by measuring the important factors such as resource utilization and performance. Different brands (and different models) of network cameras provide different data formats. Some network cameras provide only snapshots in JPEG format (updated once every several minutes). Many network cameras provide MJPEG video and some recent models also provide H.264. Comparing different data formats is beyond the scope of this paper. Instead, this paper focuses on MJPEG as it is widely supported by many network cameras. MJPEG [4] is a video format which compresses each frame as a JPEG image.

3.1 Factors Affecting the Resource Management

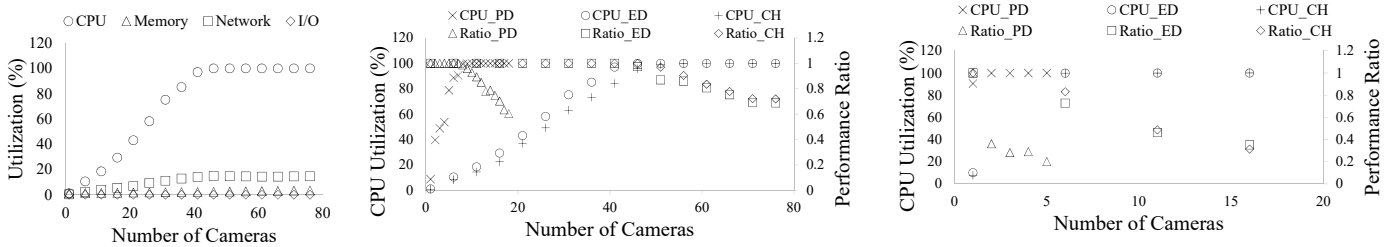
This paper uses video frame rate as the primary performance requirement because many analysis programs (such as moving object tracking) require high frame rates. Many cloud vendors provide virtual machines at different geographical locations. For example, both Microsoft Azure and Amazon EC2 provide cloud instances in USA, South America, West Europe, East Asia, Australia, etc. The network cameras are also globally distributed. Intuitively, the data from a camera should be analyzed by a cloud instance that is geographically closer in order to reduce data movement across long distances. This simple solution, however, is rarely ideal due to several factors. First, the same type of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing



(a) Target frame Rate = 30 FPS (b) Target frame Rate = 10 FPS (c) Target frame Rate = 1 FPS

Fig. 1: Measured frame rate of MJPEG data with different Round Trip Time (RTT). The figure shows the ranges and the means. The line on each bar shows the standard deviation. When the target frame rate is high, RTT has noticeable effects. When the target frame rate is low, RTT's effect is negligible. The emulated data is obtained using the *netem* utility. Please notice that the figures have different ranges for the vertical axes.



(a) ED (b) m3.2xlarge (c) m3.medium

Fig. 2: CPU utilization and number of cameras: (a) CPU is identified as the major bottleneck, (b) and (c) CPU utilization and Performance ratio (ratio of measured frame rate and target frame rate) for different analysis programs (PD, ED, and CH), while streaming MJPEG data from multiple cameras on different cloud instances. As the number of cameras increases, the CPU utilization saturates and the frame rate decreases. The performance ratio starts to decrease after the CPU utilization becomes 100%. PD has a target frame rate of 1 FPS, and ED and CH have a target frame rate of 10 FPS. An important factor is the number of cameras at which the CPU saturates.

cloud instances (same number of cores and same amount of memory) has different cost (dollars per hour) at different locations. Table 3 shows the cost of Amazon EC2 cloud instances of type *m3* at different locations [3]. There is a cost difference of 47% between the cheapest (Virginia) and the costliest (Singapore) locations.

TABLE 3: COST OF AMAZON EC2 *m3* INSTANCES PER HOUR IN US DOLLARS AT DIFFERENT GEOGRAPHICAL LOCATIONS. COST CAN BE SAVED UP TO 47% WITH EFFICIENT LOCATION SELECTION

Instance	Virginia, Oregon	Frankfurt	Singapore	Tokyo
m3.medium	0.067	0.079	0.098	0.096
m3.large	0.133	0.158	0.196	0.193
m3.xlarge	0.266	0.316	0.392	0.385
m3.2xlarge	0.532	0.632	0.784	0.770

Second, we observe in our experiments that when high frame rates are required, it is necessary to allocate cloud instances closer to the data sources. Third, if a virtual machine is capable of analyzing multiple data streams, it is preferable to consolidate the resources and reduce the costs. These factors make resource allocation a complex problem. This paper aims to address the following questions:

- What types of cloud instances should be used?
- Where are these instances?
- How many instances are needed?

To answer these questions, we run experiments to model the dependency of frame rates on the locations, types, and number of instances.

3.2 System Characteristics

The experiments for problem formulation use the following types of cloud instances: *m3.medium*, *m3.large*, *m3.xlarge* and *m3.2xlarge*. We consider cloud instances at East US (N. Virginia), West US (Oregon), Europe (Frankfurt), Asia Pacific (Singapore and Tokyo) as the cameras under consideration are near to these locations. The different locations affect the network round-trip time (RTT) and the achievable frame rates. In addition to the measured data, we also use *netem* [5], a Linux utility. Netem is a network emulator that supports adding network delays to the system thereby controlling the RTT. The network delays are increased in steps using netem and frame rates are measured. We measure the frame rates of MJPEG data from 500 network cameras distributed around the world. Table 4 shows the distribution of these cameras with respect to the Amazon EC2 N.Virginia instance. The instance located at Virginia is used in Table 4 as it is the cheapest Amazon EC2 instance of type *m3*.

To understand the impact of the locations, types, and number of instances on the frame rate, we use four different analysis programs with different resource utilization:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

TABLE 4: RANGE OF RTT FOR NETWORK CAMERAS IN OUR LIST WITH RESPECT TO A N.VIRGINIA INSTANCE

RTT (ms)	Camera Count
0-100	235
101-200	213
201-300	24
301-400	13
> 400	15

1) Data Retrieval (DR): Retrieve MJPEG data without saving to the disk. No analysis is performed. This establishes the upper bound of the frame rates.
2) Edge Detection (ED): Perform Canny edge detection [8] on each frame.
3) Color Histogram (CH)
4) People Detect (PD): A cascade of histogram of oriented gradients is used for people detection [36] on each frame.

The programs ED, CH, and PD are provided by the OpenCV library [7]. PD is the most computationally intensive program followed by ED and CH, and DR. DR is only used to study the relationship between frame rate and RTT.

3.3 Frame Rate and Network RTT

To understand the effect of RTT, we use DR to retrieve MJPEG data from 20 cameras at different locations. These cameras can achieve up to 30 *frames per second*, FPS. We increase the RTT using netem and measure the achieved frame rates. RTT is measured using 10 *ping* commands prior to measuring the frame rates. The experiment is repeated by reducing the target frame rate down to 1FPS. The experiment then launches cloud instances at the different geographical locations and streams MJPEG data from these cameras to measure the achieved frame rates for different target frame rates. We use the *m3.2xlarge* instance for this measurement so that the frame rate is limited by the network, not by the instance's capabilities.

Figure 1 shows the results. The achieved frame rates decrease noticeably at larger RTT when the frame rates are high. In Figure 1 (a), when the desired frame rate is 30 FPS, the achieved frame rates drop to below 15 FPS when RTT is above 250ms. In contrast, when the desired frame rate is low, the effect of location is negligible. In Figure 1 (c), the desired frame rate is 1 FPS, the frame rates do not drop at higher RTT. From this experiment we can conclude two main points.

- If the desired frame rate is high, we need to use the cloud instances closer to the source (smaller RTT), even though they may be more expensive.
- If the desired frame rate is low, we can stream data using the cloud instance at the lowest cost, even though RTT is high.

3.4 Frame Rate and Number of Cameras

The number of streams that can be analyzed on a cloud instance is limited by the CPU, memory, or network bandwidth of the instance. To quantify these limitations we stream and analyze MJPEG data at a target frame rate and measure the CPU utilization and the frame rate across multiple cameras. The streams are analyzed for a duration of 5 minutes and the average utilization is calculated. We repeat this experiment for the different analysis programs (ED, CH, and PD), and on the different types of cloud instances. In this experiment, the target frame rate for PD is 1 FPS, and for ED and CH are 10 FPS. We use a lower frame rate for PD since the program is more computationally expensive.

The experiment analyzes video data from the individual cameras and calculates the *performance ratio*: ratio of the average of the measured frame rates to the target frame rate. The performance ratio is 1.0 if the measured frame rate across all cameras is the same as the target frame rate. A performance ratio less than 1.0 implies that the measured frame rate on at least one camera is less than the target frame rate. The analysis programs are CPU intensive and the CPU gets saturated much faster than the network, memory, or I/O as shown in Figure 2(a). After the CPU gets saturated, the performance ratio decreases when the number of streams being analyzed increases because the CPU cannot analyze data at the given frame rate.

Figure 2 (b) shows the measured CPU utilization and performance ratio for different analysis programs. We observe that for each type of cloud instance and analysis program, the CPU utilization saturates at different points (number of cameras). When the CPU utilization reaches 100%, the performance ratio decreases as the number of cameras being analyzed increases because the CPU cannot analyze the data at the rate at which it arrives.

It is observed that the network, memory, or I/O does not saturate for any of the given analysis programs. From Figure 2, we can infer that the drop in the average frame rate is due to the saturation of the CPU. Figure 2 (b) shows that for ED and CH, the CPU utilization becomes 100% when the number of cameras becomes greater than 45. PD is more CPU intensive and saturates faster compared with ED and CH as shown in Figure 2 (b) and (c). Figures 2 (c) shows that *m3.medium* cannot be used to analyze data from more than one camera for PD. Figure 2 shows that the CPU utilization is the major factor that impacts the frame rate. For different analysis programs, the CPU utilization directly imposes a restriction on the number of cameras that can be analyzed.

This section shows that the types and locations of the cloud instances are important factors to be considered while streaming and analyzing MJPEG data from globally distributed network cameras. These factors present an opportunity to lower the overall analysis cost by carefully selecting the cloud instances.

4 Resource Management

We present the method of Adaptive Resource Management for Video Analysis in Cloud (***ARMVAC***) to reduce the cost of cloud instances for analyzing data from network cameras at the required frame rate.

ARMVAC maps the problem to the Variable Size Bin Packing Problem (VSBPP) [13]. As VSBPP is an NP-Hard problem [13], we use a heuristic algorithm to determine the types, locations, and number of cloud instances to reduce the overall cost while meeting the performance requirements. ARMVAC models: (1) the relation between CPU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

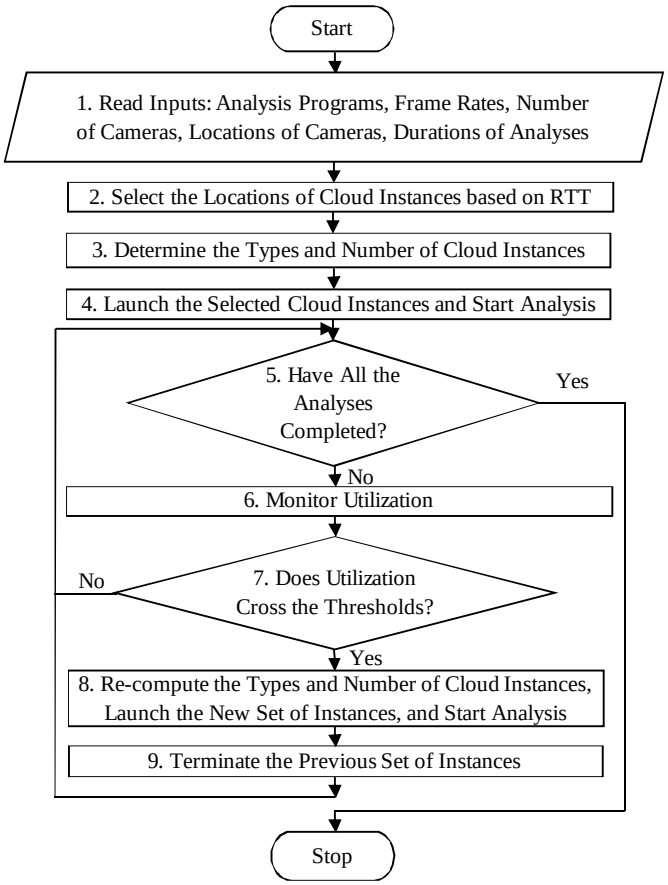


Fig. 3: Flow Diagram of ARMVAC

TABLE 5: SYMBOL AND UNITS TABLE

Symbol	Definition
t	Total number of cameras to be analyzed
V	A set of available cloud instances
v_i	A cloud instance in set V
n_i	Maximum number of cameras that v_i can analyze
K	A possible solution set having the count of all v_i instances in the solution
k_i	Number of v_i instances in the solution K
$\mathbb{S}$	Set of all possible solutions K
oc	Overall cost
c_i	cost of the cloud instance v_i
d_i	Total cost of a possible solution K
ut	CPU utilization value
p_i	Number of CPUs in an instance
m	Number of cameras being analyzed in a cloud instance, used to model ut
y	Number of cameras for which ut = 100%
α	Second order coefficient used to model ut
β	First order coefficient used to model ut
γ	Constant term used to model ut
λ	Second order coefficient used to predict n_i
μ	First order coefficient used to predict n_i
ν	Constant term used to predict n_i
Units of Measurement	
Cost	dollars per hour ($/hr)
Frame Rate	frames per second (FPS)
RTT	milliseconds (ms)
CPU utilization	percentage (%)

utilization and the number of streams being analyzed for the given analysis programs and the target frame rates, and (2) the relation between the measured frame rates and the RTT. Using the model, the method first determines the instance locations. Then the method calculates the maximum number of streams that can be analyzed on one type of instance and predicts the maximum number of streams for the other types of instances. With this information, ARMVAC can map the problem to the VSBPP. The method then uses the *Adapted Best First Decreasing* (A-BFD) approach [12] as it is reported to perform better than other heuristic algorithms for VSBPP. ARMVAC is designed for CPU intensive analyses where CPU saturates faster than other resources.

Figure 3 shows the steps. Step 1 reads the inputs needed for problem modelling explained in Section 4.1. Step 2 selects the locations of cloud instances to be considered for the given analysis as explained in Section 4.2. Steps 3 and 4 together determine the types and number of cloud instances needed for the analysis and is explained in detail in Section 4.3. Steps 5 - 9 describes the adaptive solution of resource management explained in detail in Section 4.4.

The symbols used in the paper and the units of measurement are listed in Table 5. An uppercase symbol represents a set. A symbol with a subscript implies that the symbol is an element of a set. The constants are represented using lowercase symbols. The coefficients of the equations are represented using Greek letters.

4.1 Resource Allocation as a Variable Size Bin Packing Problem

Consider the example of analyzing data from 50 cameras in North America at a target frame rate of 5 FPS for a duration of one hour. Let the set of available instances be {m3.2xlarge.Virginia, m3.xlarge.Oregon, m3.large.Virginia, m3.medium.Oregon}. The cost (per hour) of the instances is given by the set {$0.532, $0.266, $0.133, $0.067}. The maximum number of data streams that can be analyzed on each instance be {15, 8, 5, 1}. The goal is to find the types and number of instances to analyze data at the target frame rate from 50 cameras at the lowest cost. The solution is a set containing the number of instances used of the given four types. A possible solution is the set {3, 0, 1, 0} using 3 m3.2xlarge.Virginia instances and 1 m3.large.Virginia instance. The cost of the solution $1.73/hour. The set {1, 0, 0, 0} is not a solution as it means that we need to analyze the data from 50 cameras on a single m3.2xlarge.Virginia instance. If more than 15 streams are analyzed on one m3.2xlarge.Virginia instance it would fail to meet the performance requirement. Let the set of all possible solutions be { {0, 5, 2, 0}, {4, 0, 0, 0}, {2, 1, 2, 2}, ..., {0, 0, 10, 0}}. The set is finite because the number of instances and the cameras are finite. A solution with the minimum overall cost is {0, 0, 10, 0} using 10 m3.large.Virginia instances and the overall cost is $1.33/hour. We model this process of allocating resources at the lowest cost as a Variable Size Bin Packing Problem.

Let t represent the total number of cameras (e.g. 50). Let $V = \{v_1, v_2, v_3, ...\}$ = {m3.2xlarge.Virginia, m3.xlarge.Oregon, m3.large.Singapore, ...} represent the set of available cloud instances. The cost of v_i is c_i. Let n_i be the maximum number of cameras from which the data can be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

analyzed using cloud instance v_i, such that the measured frame rate is equal to the target frame rate on all the cameras.

Let $K = \{ k_1, k_2, k_3, ... \}$ represent a solution consisting of the number of instances in V required to analyze the data from the t cameras at the target frame rate. The elements of K, k_i have values in the range $\left[0, \left\lceil \frac{t}{n_i} \right\rceil \right]$ (e.g. K = {3,0,1,0} from the previous example). The value of k_i is constrained using the relation given in Equation (1).

$$\sum_{i \in [1,|V|]} (k_i \times n_i) \geq t \tag{1}$$

The overall cost of renting cloud instances for the j^{th} solution, d_j is defined as,

$$d_j = \sum_{i \in [1,|K|]} (k_i \times c_i) \tag{2}$$

Let $\mathbb{S} = \{K_1, K_2, .. \}$ be the set representing all possible solutions K. The main problem is to find the minimum overall cost *oc* given by Equation (3).

$$oc = min \sum_{j \in [1,|\mathbb{S}|]} d_j \tag{3}$$

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

- The cloud instances used (v_i) are equivalent of bins.
- The maximum number of cameras (n_i) from which the data can be analyzed by the cloud instance v_i is the equivalent of the size of the bin.
- The cost of the individual cloud instances (c_i) is equivalent to the cost of selecting the bins.

In this case each bin can be repeated as many times as required ($k_i > 1$). The inputs are t, target frame rate, and V, and the outputs are *oc* and a solution set K that has an overall cost *oc*.

4.2 Selecting the Location of Cloud Instances

A cloud instance is a candidate if the instance can analyze data from any of the given cameras at the target frame rate. ARMVAC determines the candidate cloud locations for the given set of cameras and the target frame rate. Only the candidate cloud locations are retained in the set V.

Based on the measurement data obtained in Section 3.3, ARMVAC determines the RTT values for a range of target frame rates. Table 6 shows the range of RTT values for different target frame rates. An RTT of 100ms may be sufficient for a frame rate of 12 FPS but we select an RTT of 50ms to be safe.

TABLE 6: THE RTT RANGE FOR DIFFERENT TARGET FRAME RATES

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

If the target frame rate is less than or equal to 1 FPS, we can select instances at any location considered and still achieve the target frame rate. If the target frame is 5 FPS, we should consider the cloud instances within an RTT range of 200ms from the cameras. Depending on the target frame rate, ARMVAC considers the cloud instances within the given RTT range as given in Table 6. ARMVAC selects the cheapest cloud instances based on the locations within the RTT range. For example, to analyze data from 5 cameras in Virginia and 5 cameras in Singapore at a target frame rate of 0.5 FPS, ARMVAC determines the possible cloud instance locations within the RTT range (in this case anywhere in the world). Then the method determines the types and number of cloud instances to be used. Consider another example of analyzing data from the same cameras at a target frame rate of 20 FPS. Since the RTT range is small, ARMVAC launches the instances in Virginia and Singapore separately. The other cloud locations are not considered in this case.

4.3 Determining the Types and Number of Cloud Instances

The main step in resource management is to calculate the types and number of cloud instances required to reduce the overall cost. The cost of the cloud instance (c_i) and the maximum number of cameras that can be analyzed on a cloud instance (n_i) are the two important factors used to determine the types and number of the cloud instances. The method first determines n_i for an instance which is equivalent to the size of the bin in the VSBPP. Then the method predicts the n_i for other types of instances. Finally the c_i and n_i values are given as inputs to the heuristic algorithm used to determine the types and number of cloud instances needed.

4.3.1 *Determining n_i for a Cloud Instance*

ARMVAC models the relationship between CPU utilization (ut) and the number of cameras (m) being analyzed on each cloud instance v_i as,

$$ut = \begin{cases} \alpha m^2 + \beta m + \gamma & \text{for } m \leq y \\ 100\% & \text{otherwise} \end{cases} \tag{4}$$

The model is obtained based on the measurement data obtained from three different experiments using the PD, ED, and CH analysis programs as shown in Figure 2. A quadratic model is used as the error in calculating the CPU utilization is less compared with other models like linear, exponential, and polynomial (degree 3). The content of the scene affects the utilization and n_i, and cameras have versatile content. Also, the CPU utilization doesn't increase linearly with the number of threads (one per camera). Therefore, a quadratic model is more suitable. According to Equation (4), the CPU utilization increases with the number of cameras in a quadratic fashion until the number of cameras is equal to y. For the number of cameras greater than or equal to y, the CPU utilization remains at 100%. Table 7 shows the values for the coefficients α, β, and γ for different analysis programs and instances. Some of the calculated α and γ coefficients have negative values. This may cause the utilization to decrease for large values of m.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

TABLE 7: COEFFICIENTS TO MODEL CPU UTILIZATION FOR DIFFERENT TYPES OF INSTANCES AS GIVEN IN EQUATION (4)

Cloud Instances	m3.2xlarge			m3.xlarge			m3.large			m3.medium		
Analysis Programs	ED	CH	PD	ED	CH	PD	ED	CH	PD	ED	CH	PD
α	-0.03	-0.02	-0.61	0.06	-0.05	-0.83	-0.17	-0.08	-5.81	-0.89	-0.53	-1.34
β	3.59	3.06	15.67	5.37	4.65	19.05	8.42	5.63	49.9	20.71	15.32	9.89
γ	-12.10	-11.80	6.38	-3.34	-3.87	3.15	3.15	14.52	2.24	-5.18	3.64	83.10

TABLE 8: THE COEFFICIENTS FOR PREDICTING n_i FOR DIFFERENT NUMBER OF CORES p_i AS GIVEN IN EQUATION (6)

n_i for p_i = 8	λ	μ	ν
$n_i \leq 10$	0.04	0.45	0.67
$n_i \geq 15$	-0.16	6.78	-3.00
$10 < n_i < 15$	$0.04 - \left(0.20 \times \frac{(n_i - 10)}{5}\right)$	$0.45 + \left(6.33 \times \frac{(n_i - 10)}{5}\right)$	$0.67 - \left(3.67 \times \frac{(n_i - 10)}{5}\right)$

$$n_i = \max_{m \in [1,T]} m \text{ such that } ut < 90\% \quad (5)$$

Equation (5) defines the maximum number of streams, n_i. Figure 2 shows that the performance drops after the CPU utilization becomes 100%. Hence to avoid the drop in performance, we define n_i to be equal to the maximum number of data streams that results in a CPU utilization less than 90%. Using a utilization below 90% can reduce the number of streams being analyzed by an instance which may increase the number of instances used.

Given an analysis program we can determine n_i by calculating the coefficients (α, β, and γ). by measuring the CPU utilization for the different values of m. Using Equations (4) and (5), we calculate n_i for the given analysis program.

Since we consider CPU-intensive programs, the average frame rate will start decreasing after the number of cameras being analyzed is greater than n_i. Hence if we analyze data from the given cameras at the same target frame rate, then n_i is the maximum number of streams that can be analyzed on a cloud instance such that the target frame rate is achieved on all the cameras. The different steps involved in determining n_i for an instance are summarized as follows:

1) Analyze MJPEG data at the target frame rate and measure the CPU utilization (ut) for different number of cameras (m). A minimum of 3 utilization values are required.
2) Use the CPU utilization information obtained in step 1 to compute the values for α, β, and γ in Equation (4) by solving the system of equations.
3) Use α, β, and γ obtained from step 2 to compute n_i using Equation (5)

4.3.2 Predicting n_i for Instances

Given an analysis program to analyze data from t cameras, we need to find n_i for different types of cloud instances. Calculating n_i on each type of cloud instances individually is time consuming. ARMVAC develops a prediction model to determine n_i on different types of cloud instances to save time.

Figure 4 shows n_i for the same analysis program on different types of cloud instances represented using the number of cores in the instance (in this case *m3.medium* = 1 CORE, *m3.large* = 2 CORES, *m3.xlarge* = 4 CORES, *m3.2xlarge* = 8 CORES). Equation (6) models the relationship between the n_i and the number of cores in a cloud instance based on Figure 4. In Equation (6), p_i represents the number of cores and λ, μ, and ν represent the coefficients of the model. A quadratic model is used instead of a linear model to reduce the error in predicting n_i.

$$n_i = \lambda {p_i}^2 + \mu p_i + \nu \quad (6)$$

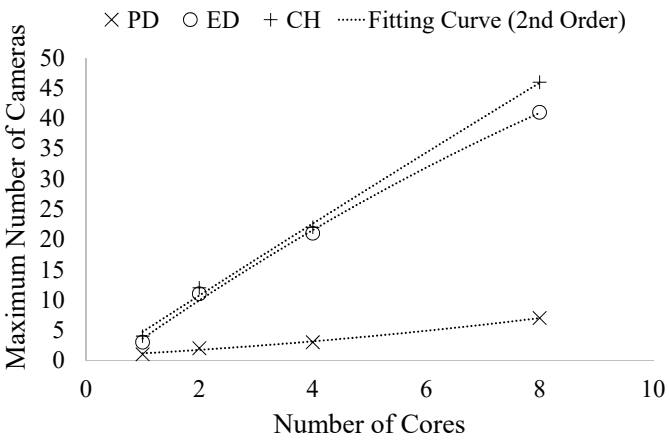


Fig. 4: The maximum number of cameras (n_i) while executing PD, ED, and CH analysis programs on cloud instances with different number of cores. The fitting curves as per Equation 6 are shown. Note that the X axis has discrete values.

Equation 6 can be used to predict n_i for different instances by substituting the values of p_i corresponding to the instances. The λ, μ, and ν coefficients for a given analysis program are determined using the value of n_i for the instance with maximum number of cores (in this case *m3.2xlarge*, $p_i = 8$). The method described in Section 4.3.1 is used to determine n_i. Based on the calculated n_i we select λ, μ, and ν using Table 8. It has the pre-determined values for the coefficients λ, μ, and ν based on the experimental data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

using the PD, CH, and ED programs. The coefficients are pre-determined for $n_i \leq 10$ and $n_i \geq 15$. For $10 < n_i < 15$ we linearly interpolate the values for the coefficients as shown in Table 8 as we did not have data points in that range. For $n_i \geq 15$, the λ and ν coefficients have a negative value. This implies that for such programs, the instances with fewer cores have large values for n_i making them more cost-efficient. The numbers *10* and *15* are obtained based on our experimental data.

Since λ, μ, and ν in Table 8 are obtained based on the analysis programs (PD, ED, and CH), for a new analysis program we need to reduce the error in prediction. The Levenberg-Marquadt (LM) algorithm [26] is used to minimize the error in prediction. The calculated value of n_i for $p_i = 8$ is used with the LM algorithm to determine the μ coefficient that results in minimum error. With one calculated value for n_i, only one coefficient in Equation (6) can be optimized using the LM algorithm. We choose μ since the average error in prediction is the lowest with optimizing the μ coefficient compared with λ and ν coefficients. The new value for μ along with λ and ν as given in Table 8 is used in Equation (6) to calculate n_i for other cloud instances (in this case $p_i = 1, 2, \text{and } 4$). The error due to using pre-determined values for λ and ν does not affect the final result (overall cost). We observe that determining n_i for the instance with the maximum number of cores and predicting the n_i values for other instances with fewer number of cores gives the best prediction results.

The different steps involved in predicting n_i for instances are summarized as follows:

1) Calculate n_i for the instance with maximum number of cores (in this case $p_i = 8$) using the steps described in Section 4.3.1.
2) Use the n_i value obtained from step 1 and the λ, μ, and ν values in Table 8 to calculate the optimized μ coefficient using the Levenberg-Marquadt (LM) algorithm.
3) Predict n_i for the other instances (in this case $p_i = 1, 2, \text{and } 4$) using Equation (6) with the new μ coefficient obtained from step 2.

4.3.3 Heuristic Algorithm

ARMVAC solves the variable size bin packing problem modelled in Section 4.1 using a heuristic algorithm based on the *Adapted Best First Decreasing* (A-BFD) approach [12]. An outline of our heuristic algorithm is given in Figure 5. ARMVAC considers the general case where the target frame rates and analysis programs are different for cameras. First, the *per-core-utilization*: defined as the ratio of the number of streams that can be analyzed using the instance (n_i) and the number of cores (p_i) is calculated for each camera. The cameras with the same *per-core-utilization* are grouped together. The next step is to sort the cloud instances based on the ratio of the instance cost (c_i) and n_i for each group. The *best cloud instances* are the ones with lower ratio values. After sorting the cloud instances V based on the $\frac{c_i}{n_i}$ ratio, the best instance is given by v_1. Since we can use multiple cloud instances of the same type, our approach uses k_1 instances of type v_1, where $k_1 = \left\lceil \frac{t}{n_1} \right\rceil$. Each instance of type v_1 can analyze data from n_1 cameras. The process is repeated until all cameras are assigned.

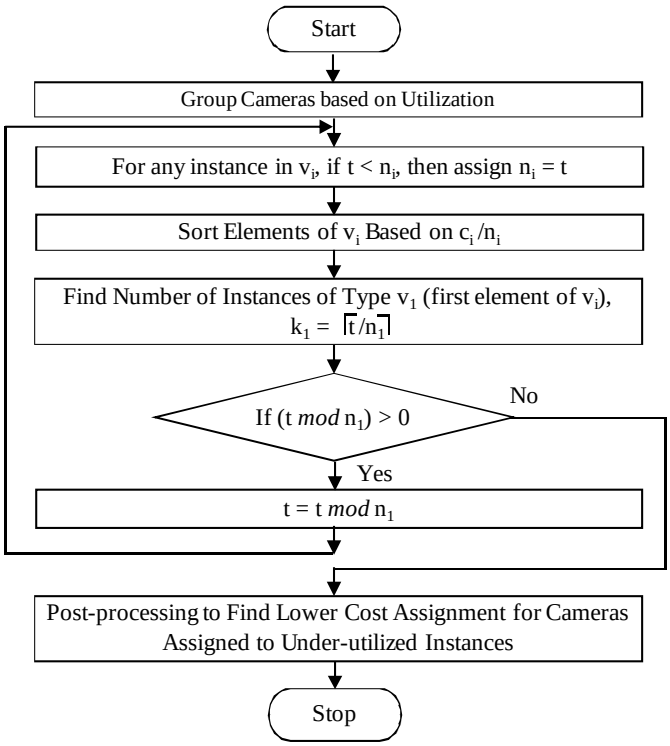


Fig. 5: Flow Diagram of the Heuristic Algorithm. The symbols used are explained in Table 5.

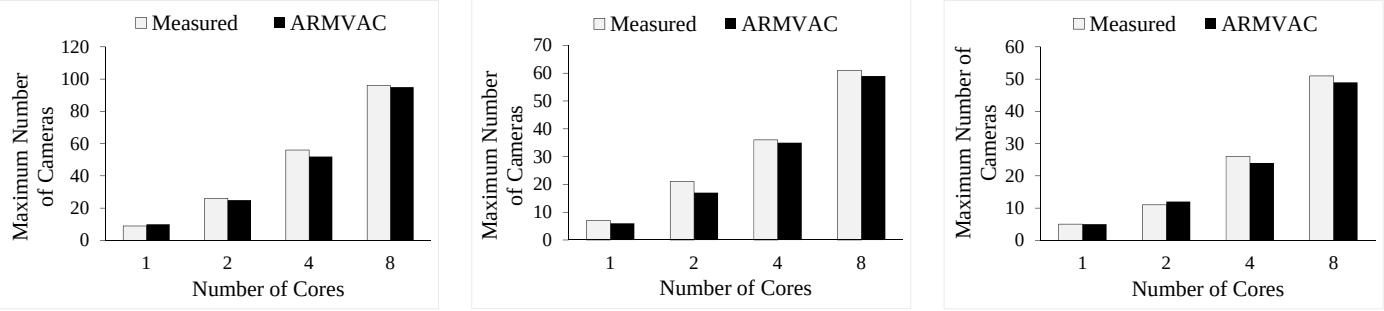
The post processing stage determines the remaining utilization for instances in each group and checks the possibility to combine the under-utilized instances and reduce cost. Each group will only have a maximum of one under-utilized instance. Except for the last iteration, each step assigns n_i cameras to the best instance. The last iteration will add the remaining cameras to the best instance. The total number of cameras assigned to the under-utilized instances are significantly smaller than the total number of cameras. Therefore an exhaustive search is performed to determine a lower cost assignment for these.

The heuristic algorithm of ARMVAC determines the number and types of cloud instances for analyzing data at the target frame rate from the given number of cameras such that the overall cost is low.

4.4 Adjusting Resources at Run-Time

The requirements of the applications may vary over time. For example, counting pedestrians are more computationally intensive during rush hours with more people than at other times. The duration of analysis may also be different for analysis programs. Hence the solution needs to adapt to these changes in requirements. We start the analysis by launching the cloud instances selected by the cost effective solution and monitor the utilization of the instances continuously. If the utilization increases above 95% because of the scene becoming more computationally intense, or the utilization drops below 50% because of change in requirements, ARMVAC recomputes the cost effective solution.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing



(a) Target frame Rate = 5 FPS (b) Target frame Rate = 10 FPS (c) Target frame Rate = 15 FPS

Fig. 6: Accuracy of the estimate of the maximum number of cameras (n_i) necessary for input scenarios 1, 2, and 3. The FD analysis program is used with 50 cameras at three target frame rates: 5 FPS, 10 FPS, 15 FPS. As target frame rates increases, n_i decreases. ARMVAC determines n_i accurately with an average error less than 2 cameras. Please notice that the figures have different ranges for the vertical axes.

Since the target utilization is 90% (Equation 5), an upper threshold of 95% is selected allowing a safe margin. As recomputing the solution requires running the largest instance type (m3.2xlarge) for a few minutes (Section 4.3 - 4.3.3), lower threshold of 50% is selected to reduce recomputing the solution. It is important to prevent data loss while switching to the newly computed solution. If the current cloud instances are terminated and then the new instances are launched, there is a possibility for data loss. Hence the cloud instances selected by the new solution are launched first, and then the old instances are terminated. This way of switching to the new solution may result in duplicate data being generated from the network cameras present in both old and new configuration, but will not result in data loss. Launching the new instances first slightly increases the cost as both old and new instances are running together for a short duration ($0.532 per an EC2 Virginia m3.2xlarge instance). Removing duplicate data and efficiently switching to the new solution considering the cost and time taken to determine and launch the instances are part of the future work.

5 Evaluation of ARMVAC

This section evaluates ARMVAC based on the accuracy of the method in Section 5.1, cost and performance in Section 5.2, and based on the cloud instance location selection in Section 5.3. We observe that ARMVAC can estimate the maximum number of cameras for each instance type with an average error of less than 1 camera for PD, ED, and CH analysis programs. Evaluating these analyses using ARMVAC against the different scenarios results in significant cost reduction. In order to demonstrate that ARMVAC can handle different analysis programs and does not have any bias towards a single analysis, we use three new analyses provided by the OpenCV library for evaluation:

- Motion Estimation (ME): Use background subtraction to estimate motion. Requires a minimum of two frames to build the background model.
- Face Detection (FD): Haar Feature based cascade classifiers are used for face detection on each frame.
- SIFT (SIFT): Extracts scale invariant features [22] on each frame.

It should be noted that using PD, ED, and CH with ARMVAC yields similar cost reduction as given in Section 5.2. We use the input scenarios given in Table 9 by varying the target frame rate and the number of cameras for different analysis programs to evaluate our method.

TABLE 9: INPUT SCENARIOS TO EVALUATE ARMVAC

Input Scenarios	Analysis Program	Number of Cameras	Target Frame Rate (FPS)
Scenario 1	FD	50	5
Scenario 2	FD	50	10
Scenario 3	FD	50	15
Scenario 4	SIFT	100	1
Scenario 5	SIFT	200	1
Scenario 6	SIFT	500	1
Scenario 7	ME	10	25
	SIFT	100	1
	FD	50	10

The overall costs of our method for the input scenarios are compared with the strategies ST1 - ST4 described in Table 10. Each strategy uses only the specified instance type. For example, ST1 always uses the *m3.medium* instance type. The maximum number of streams analyzed per instance for each strategy is same as that of ARMVAC for the different scenarios to ensure a fair comparison. We use strategy ST5 to evaluate the importance of considering RTT while selecting cloud configurations. In the evaluation, we analyze MJPEG data of size up to 14GB and the number of images up to 150,000.

TABLE 10: STATIC ALLOCATION STRATEGIES TO COMPARE ARMVAC

Name	Instance Type Used	Number of Cores per Instance
ST1	m3.medium	1
ST2	m3.large	2
ST3	m3.xlarge	4
ST4	m3.2xlarge	8

5.1 Accuracy of the Model

The accuracy of ARMVAC depends on how accurately n_i can be calculated for each type of cloud instance. Since n_i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

directly impacts the overall cost of the method, it is important to evaluate the accuracy of the method. We calculate n_i for each type of instance and for the input scenarios. To obtain the actual data, we measure the CPU utilization and determine n_i for which the utilization exceeds 90%, for each type of cloud instance and the input scenarios. The calculated values of n_i are compared with the actual data to evaluate the accuracy of the method.

The accuracy of calculating an estimate of n_i for input scenarios 1 - 3 using ARMVAC is shown in Figure 6. As the target frame rate increases, n_i for each type of instance decreases. ARMVAC determines n_i accurately as the average error for scenarios 1, 2, and 3 is below 2 cameras. For input scenarios 4, 5, and 6, the target frame rate is fixed and the number of cameras changes. So we need to calculate n_i only once for each type of instance. The accuracy of calculating an estimate of n_i for input scenarios 4, 5, and 6 is shown in Figure 7 (a). In this case ARMVAC can determine n_i with an average error of one camera. Scenario 7 represents a general case with three different analysis programs, frame rates, and number of cameras. As we already measured the accuracy for FD and SIFT programs, we only need to determine the accuracy of calculating n_i for ME analysis on data from 10 cameras at 25 FPS. The results are shown in Figure 7 (b). Our method can determine n_i with an average error of one camera.

The results show that ARMVAC has high accuracy in determining the maximum number of streams an instance can analyze when evaluated against analysis programs ME, FD, and SIFT for different target frame rates and number of cameras.

5.2 Cost and Performance

The cost and the performance of the method are evaluated in two stages. The algorithm explained in Section 4.3.3 is responsible for selecting cloud instances at lower costs, hence it is important to evaluate the algorithm first. The algorithm is compared with CPLEX [11] to check the correctness of the algorithm. ARMVAC is tested against the different input scenarios in Table 9 and the overall cost of cloud instances selected by ARMVAC for the input scenarios is compared with the strategies ST1 - ST4 in Table 10.

5.2.1 Comparison of the Heuristic Algorithm with CPLEX

CPLEX [11] is an optimization software package developed by IBM that we used to evaluate ARMVAC. In order to use CPLEX we formulate our problem as an integer linear programing problem. CPLEX will output the number of cloud instances of each type. The total cost of instances selected by CPLEX for all the input scenarios are then compared with ARMVAC. The comparison of heuristic algorithm of ARMVAC and CPLEX for the input scenarios are given in Table 11. We observe that ARMVAC selects cloud instances with the same total cost as that of CPLEX. The types and number of instances selected by ARMVAC and CPLEX differ in some cases but the overall cost remains the same. Note that CPLEX only solves the integer linear programming problem and it requires n_i and locations determined by ARMVAC to produce the result.

5.2.2 Comparison of ARMVAC with Different Strategies

The maximum number of streams that can be analyzed on each instance for the different input scenarios is calculated using the steps mentioned in Section 4. Then the heuristic algorithm is applied to select the cloud instances such that the target frame rate is achieved on all cameras, and the overall cost is low. We launch the cloud configurations selected by ARMVAC as given in Table 11 and analyze data from the cameras. The achieved frame rate is measured on all the cameras. If the average frame rate achieved is equal to the target frame rate, the performance is 100%.

The cloud instances in N. Virginia are used for this evaluation as these instances are within the RTT range of the required number of cameras which support the target frame rates given in Table 9. The overall costs of our method for the input scenarios are compared with the strategies ST1 - ST4 described in Table 10. The method achieves the desired performance as the average frame rate is measured to be equal to the target frame rate on all cameras for the input scenarios.

Figure 8 (a) shows the comparison of overall cost of ARMVAC for input scenarios 1, 2, and 3 compared with the strategies ST1 - ST4. ARMVAC can reduce cost up to 50%, 26%, and 44% for 5 FPS, 10 FPS, and 15 FPS respectively. For scenarios 1 - 3, *m3.2xlarge* is the least cost efficient instance and hence ST4 has the highest cost. For 15 FPS, the cost of ARMVAC is less than all the strategies as ARMVAC selects different instance types

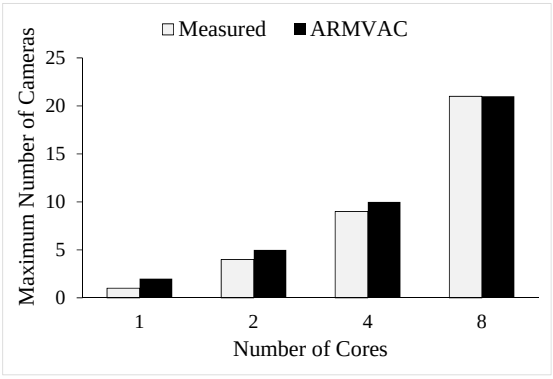
The impact of frame rates on the overall cost of ARMVAC is shown in Figure 8 (b). It can be seen that as the frame rate increases from 5 FPS to 15 FPS, the overall cost increases almost linearly by 125%. This is due to the fact that as the target frame for analysis increases, the CPU utilization for analyzing data from a single camera also increases and the number of streams that can be analyzed in an instance decreases.

The comparison of the overall cost of ARMVAC for input scenarios 4, 5, and 6 with the strategies ST1 - ST4 is shown in Figure 9 (a). In this case, ARMVAC can reduce the cost up to 60%, 61%, and 62% respectively for the number of cameras equal to 100, 200, and 500. The effect of the number of cameras on the overall cost of ARMVAC is shown in Figure 9 (b). It can be seen that as the number of cameras increases from 100 to 500, the overall cost increases almost linearly by 392%. This is due to the fact that as the number of input cameras increases, more resources are required to perform the analysis. Figure 8 (b) and 9 (b) show that the number of input cameras has more significant impact on overall cost compared with the target frame rate.

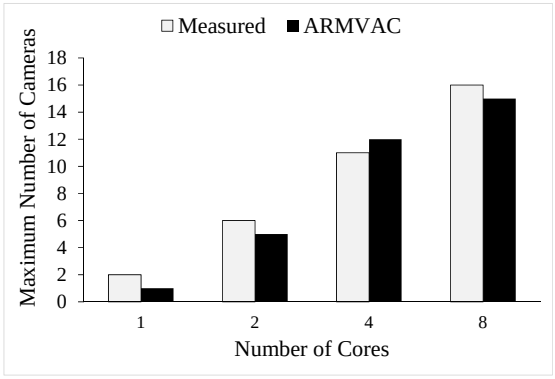
The impact of ARMVAC on overall cost for input scenario 7 compared with the strategies ST1 - ST4 are shown in Figure 10. The capability of ARMVAC to select different instance types along with the post processing stage in the heuristic algorithm to find lower cost assignment provides a cost reduction of up to 57%. For scenarios 4 - 7, *m3.medium* is the least cost efficient instance and hence ST1 has the highest cost.

Our evaluation shows that ARMVAC can reduce the overall cost up to 62% compared with the different strategies and achieve 100% performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

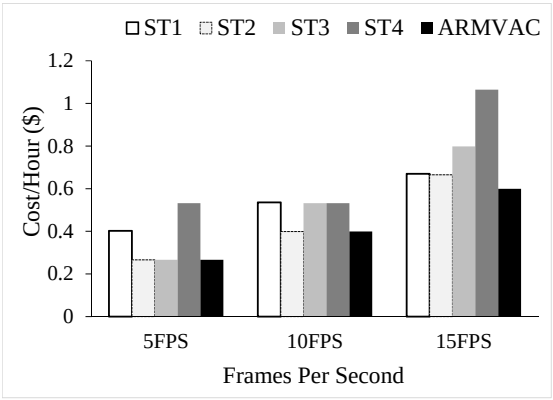


(a) SIFT

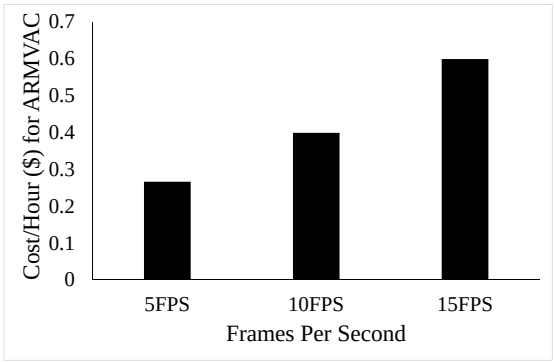


(b) ME

Fig. 7: Accuracy of n_i estimated by ARMVAC for: (a) the input scenarios 4, 5, and 6. The SIFT analysis program is used with 100, 200, and 500 cameras at a target frame rate of 1 FPS. ARMVAC determines n_i accurately with an average error of 1 camera. (b) input scenario 7. The ME analysis program is used with 10 cameras at a target frame rate of 25 FPS. ARMVAC determines n_i accurately with an average error of 1 camera.



(a)



(b)

Fig. 8: Overall cost of ARMVAC: (a) compared with the four strategies (ST1 - ST4). The FD analysis program is used with 50 cameras at three target frame rates: 5 FPS, 10 FPS, 15 FPS. ARMVAC provides a cost reduction of up to 50%. (b) for three target frame rates: 5 FPS, 10 FPS, and 15 FPS. The overall cost increases by 125% as the frame rate increases by 200%.

TABLE 11: COMPARISON OF OUTPUTS OF CPLEX AND ARMVAC

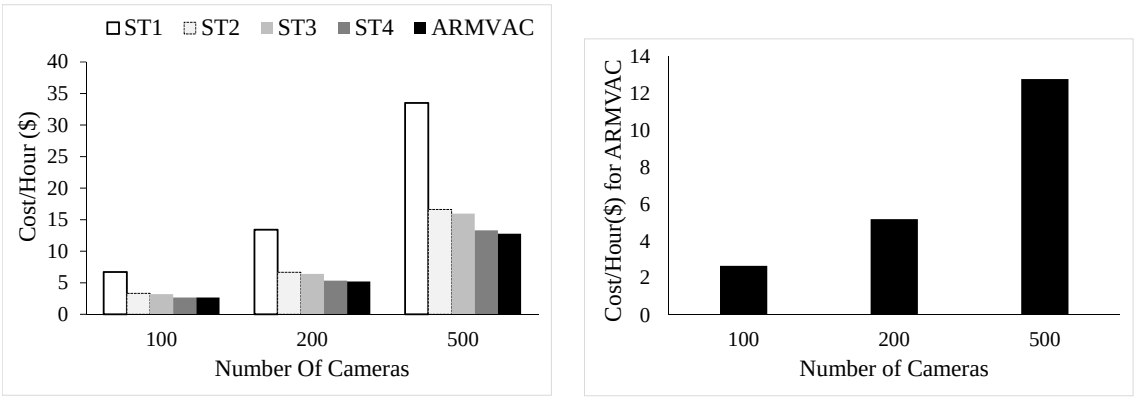
Input Scenario		m3.2xlarge	m3.xlarge	m3.large	m3.medium	Cost ($/hr)
Scenario 1		1	0	0	0	0.266
Scenario 2		0	1	1	0	0.399
Scenario 3	CPLEX	0	0	4	1	0.599
	ARMVAC	1	0	0	1	0.599
Scenario 4		4	1	1	1	2.594
Scenario 5		9	0	2	1	5.121
Scenario 6		23	1	1	1	12.702
Scenario 7	CPLEX	4	1	6	1	3.259
	ARMVAC	4	1	6	1	3.259

5.3 Instance Location and Cost

To evaluate the importance of considering RTT while selecting cloud instances, we use the strategy ST5 and compare it with ARMVAC. We select 20 cameras located at different parts of the continents North America, Europe, and Asia. The DR program is used to analyze the MJPEG data from these network cameras. We consider the cloud instances from the same three continents as the cameras. The data from the cameras are analyzed at different frame rates as shown in Table 12. The *m3.2xlarge* instances are used. Since DR only retrieves the data and does not perform any analysis, we can retrieve data from all 20 cameras using one *m3.2xlarge* instance.

ST5 always analyzes data from a network camera using a cloud instance geographically closer to the camera. One cloud instance can retrieve data from multiple cameras. For lower target frame rates, ARMVAC consolidates all the instances to the cheapest instance. For higher frame rates, ARMVAC selects the geographically closest instance. Table 12 compares the overall cost of m3.2xlarge cloud

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing



(a) (b)

Fig. 9: Overall cost of ARMVAC: (a) compared with the four strategies (ST1 - ST4). The SIFT analysis program is used with 100, 200, and 500 cameras at a target frame rate of 1 FPS. ARMVAC provides a cost reduction of up to 62%. (b) for three different input number of cameras: 100, 200, and 500. The overall cost increases by 392% as the number of cameras increases by 400%.

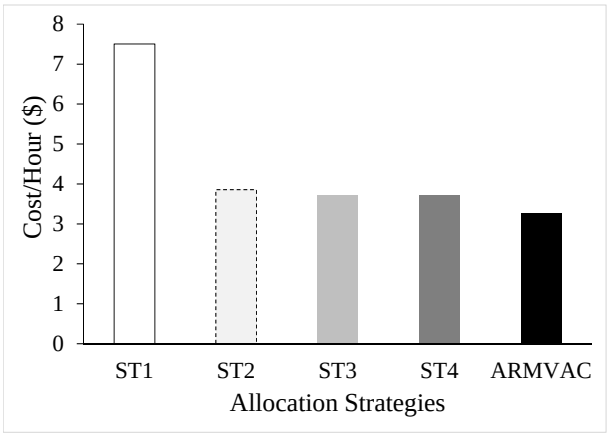


Fig. 10: Overall cost comparison of ARMVAC with the four strategies for Scenario 7. Three different analysis programs, number of cameras, and target frame rates are considered. ARMVAC provides a cost reduction of up to 57%.

instances selected by ST5 and ARMVAC for different target frame rates. For frame rates 1 FPS or lower, ARMVAC retrieves the data using the cheapest instance (Virginia) and can reduce the cost by 78% when compared with ST5. For the target frame rate of 5 FPS, due to RTT constraints ARMVAC uses instances from Virginia and Oregon. In this case, ARMVAC can reduce the cost by 56%. For the frame rate of 10 FPS, ARMVAC uses instances from Virginia, Oregon, and Frankfurt and reduces the cost by 31%.

TABLE 12: COST COMPARISON BETWEEN ST5 AND ARMVAC. THE INSTANCE LOCATIONS ARE OREGON(OR), FRANKFURT(FT), TOKYO (TO), VIRGINIA (VA)

Frame Rate (FPS)	Instances Used by ST5	Instances Used by ARMVAC	Cost ST5 ($)	Cost ARMVAC ($)	Cost Savings (%)
0.3	OR,FT,TO,VA	VA	2.466	0.532	78.4
1.0	OR,FT,TO,VA	VA	2.466	0.532	78.4
5.0	OR,FT,TO,VA	OR,VA	2.466	1.064	56.9
10.0	OR,FT,TO,VA	OR,FT,VA	2.466	1.696	31.2
15.0	OR,FT,TO,VA	OR,FT,TO,VA	2.466	2.466	0.00
20.0	OR,FT,TO,VA	OR,FT,TO,VA	2.466	2.466	0.00
30.0	OR,FT,TO,VA	OR,FT,TO,VA	2.466	2.466	0.00

6 Conclusion and Future Work

This paper presents ARMVAC, an adaptive resource manager to select low-cost cloud instances for analyzing MJPEG data from globally distributed network cameras. Inputs to ARMVAC are the analysis programs, the required number of cameras, the locations of the cameras, the target frame rates, and the durations of the analyses. The outputs are the types, locations, and number of cloud instances to be launched to achieve the target frame rate on all the cameras. ARMVAC includes a model to predict the maximum number of streams that can be analyzed on different types of instances. We evaluate ARMVAC using *Amazon EC2* cloud instances and observe that the achieved frame rate on all cameras is equal to the target frame rate for different input scenarios thereby satisfying the performance requirements. We observe that ARMVAC lowers the overall cost up to 62% when compared with four other reasonable strategies (ST1 - ST4) for selecting cloud configurations. Our evaluation demonstrates that our method is not ad-hoc and can be applied to different analysis programs.

As part of our future work, we will extend the method to handle analysis programs which are memory intensive, bandwidth intensive, or I/O intensive. We would also like to improve our method of adaptively launching instances while adjusting to the run-time conditions. We also plan to study effect of adaptive nature of H.264 streams on resource selection.

Acknowledgments

We would like to thank Amazon and Microsoft Azure for providing the cloud infrastructure, and the organizations that provide the camera data. This project is supported in part by NSF ACI-1535108, CNS-0958487, and IIP-1530914. 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 Auto Scaling. https://aws.amazon.com/autoscaling/.
[2] Amazon EC2. http://aws.amazon.com/ec2/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2836907, IEEE Transactions on Cloud Computing

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

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

[5] netem, Network Emulation Utility. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[6] Network camera and video analytics market. http://www.marketsandmarkets.com/Market-Reports/visual-communication-market-775.html, 2011.

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

[8] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[9] W. Chen et al. Adaptive cloud resource allocation for analysing many video streams. In *IEEE International Conference on Cloud Computing Technology and Science*, 2015.

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

[11] I. I. CPLEX. V12. 1: Users manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[12] 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.

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

[14] R. Gargees et al. Incident-supporting visual cloud computing utilizing software-defined networking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):182–197, 2017.

[15] Hossain et al. Resource allocation for service composition in cloud-based video surveillance platform. In *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, pages 408–412, 2012.

[16] M. A. Hossain and B. Song. Efficient resource management for cloud-enabled video surveillance over next generation network. *Mobile Networks and Applications*, 21(5):806–821, 2016.

[17] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, 2012.

[18] 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.

[19] A. S. Kaseb et al. Cloud resource management for image and video analysis of big data from network cameras. In *International Conference on Cloud Computing and Big Data*, 2015.

[20] A. Khan, X. Yan, S. Tao, and N. Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1287–1294.

[21] Y. Koh et al. Improve safety using public network cameras. In *IEEE Symposium on Technologies for Homeland Security (HST)*, pages 1–5, 2016.

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[23] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[24] A. Mohan et al. Internet of video things in 2030: A world with many cameras.

[25] A. Mohan et al. Location based cloud resource management for analyzing real-time videos from globally distributed network cameras. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 176–183, 2016.

[26] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[27] H. Nguyen Van, F. Dang Tran, and J.-M. Menaud. Autonomic virtual resource management for service hosting platforms. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 1–8. IEEE Computer Society.

[28] Regazzoni et al. Video analytics for surveillance: Theory and practice. *Signal Processing Magazine, IEEE*, 27(5), 2010.

[29] U. Sharma et al. A cost-aware elasticity provisioning system for the cloud. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 559–570. IEEE, 2011.

[30] B. Song et al. A two-stage approach for task and resource management in multimedia cloud environment. *Computing*, 98(1-2):119–145, 2016.

[31] W.-T. Su et al. Harvest the information from multimedia big data in global camera networks. In *Multimedia Big Data, 2015 IEEE International Conference on*, pages 184–191.

[32] Vijayakumar et al. Dynamic resource provisioning for data streaming applications in a cloud environment. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 441–448.

[33] Wang et al. Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities. In *INFOCOM, 2012 Proceedings IEEE*, pages 199–207.

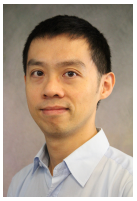
[34] Wu et al. Cloudmedia: When cloud on demand meets video on demand. In *Distributed Computing Systems, 2011*.

[35] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1107–1117, 2013.

[36] Zhu et al. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*.

Anup Mohan obtained his Ph.D. from the School of Electrical and Computer Engineering at Purdue University in 2017. His research interests include large-scale video analysis, cloud computing, and big data analysis. Anup Mohan is currently working at Intel Corporation, Santa Clara, U.S.A.

Ahmed S. Kaseb is an assistant professor of computer engineering in the Faculty of Engineering at Cairo University. He obtained the Ph.D. in computer engineering from Purdue University in 2016. He obtained the M.S. and B.E. in computer engineering from Cairo University in 2013 and 2010 respectively. He is conducting research in Big Data, Cloud Computing, and Computer Vision.



Yung-Hsiang Lu is a Professor in the School of Electrical and Computer Engineering of Purdue University. He is an ACM distinguished scientist and ACM distinguished speaker. He is the lead investigator of a software system for analyzing thousands of network cameras worldwide. He is the lead organizer of the first Low-Power Image Recognition Challenge in 2015-2017. He obtained the Ph.D. from Stanford University.

Thomas J. Hacker is an Professor of Computer and Information Technology at Purdue University. Dr. Hackers research interests center around high-performance computing and networking on the operating system and middleware layers. Recently his research has focused on cloud computing, reliability, and cyberinfrastructure. Thomas has a M.S. and Ph.D. in Computer Science & Engineering from the University of Michigan in Ann Arbor, Michigan.