# COMPUTING RESOURCE MINIMIZATION WITH CONTENT-AWARE WORKLOAD ESTIMATION IN CLOUD-BASED SURVEILLANCE SYSTEMS

*Peng-Jung Wu and Yung-Cheng Kao*

Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan

**ABSTRACT**

As cloud computing platform provides computing power as utilities, it is important to develop a mechanism to adaptively adjust the resources needed for handling cloud service. In this paper, a computing resource minimization framework for cloud-based surveillance video analysis systems is proposed. Videos streams are divided into clips and multiple processing nodes are used to handle clips. While the quality-of-service (QoS) is maintained, the proposed framework dynamically adjusts the number of processing nodes based on a proposed content-aware workload estimation mechanism. Experimental results show that the proposed mechanism successfully predicts the variability of system workload while QoS is maintained and outperforms other mechanisms in terms of average virtual machine (VM) quantity and job failure ratio.

***Index Terms***— Cloud computing, auto-scaling, resource-minimization

## 1. INTRODUCTION

Cloud computing has been a promising technology for hosting large scale video surveillance systems, which allows users to place IPCAMs at home and to stream surveillance video to cloud platform for storage. Video analysis service is often considered as an add-on feature that actively analyzes video stream and sends out alarms when suspicious events are detected. Due to the characteristics of video analysis, the processing time of a video clip containing suspicious events could be much longer than that of a video clip without suspicious events. Thus, the computing power needed for analyzing video streams varies from time to time. In addition, since cloud computing platform provides computing power as utilities, e.g., pay-as-you-go, minimizing the computing power means saving cost. For example, VMs can be consolidated into few physical machines and thus those physical machines without VM running on top of them can be turned off to save energy. For the above reasons, it is important to develop a mechanism that dynamically allocates computing power without sacrificing QoS.

One way to deploy a live video processing system on cloud platform is to statically assign video streams to processing nodes. This method, however, is not a desirable solution as the processing time of a video stream varies from time to time. Thus, to meet the real-time requirements of surveillance systems, video streams are usually divided into small clips. According to the workload, clips are then dispatched to processing nodes [1]. Since job scheduling over multiple processing nodes is an NP-hard problem, several heuristic algorithms are proposed to address it. More specifically, in [2], each processing node equips with a task queue. A high computation task is divided into multiple sub-tasks and inserted into queues of different processing nodes for being processed in parallel to shorten the entire finish time. In [3], a Map-Reduce-based framework for video transcoding is proposed. While tasks are assigned to computer based on a proposed minimal-complete-time algorithm, high computation tasks are assigned to powerful processing nodes. The above works, however, deal with fixed number of processing nodes without considering the capability of dynamically resource provisioning of cloud computing.

To better utilize the cloud resources, many researches have been conducted that dynamically acquires VMs from platform when existing computing power is insufficient to handle the increasing workload. More specifically, Marshall [4] increases the number of VMs when jobs waiting in queue cannot be finished before deadline. Assuncao [5] proposes several policy-based mechanisms to decide when to acquire new VM instance under different workloads. In [6], both user performance requirements and budget concerns are taken into consideration. By allocating/deallocating VMs dynamically and assigning tasks to the most cost-efficient VMs, system can finish tasks within the deadlines at minimum financial cost. However, the above works make an assumption that application owners need to be able to estimate the task processing time before a task being submitted to the system, which in many cases, is not practical.

In this paper, a computing resource minimization framework for cloud-based surveillance video analysis systems is proposed. Instead of asking users to estimate the workload of tasks, the proposed system pre-analyzes the

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

coming video clips and extracts useful information for estimating system workload. Based on the workload estimation result, system derives the number of VMs needed to process the tasks within the deadline.

The rest of this paper is organized as follows. Section 2 gives an architecture reference of cloud-based surveillance video analysis systems. Section 3 presents the proposed adaptive workload estimation mechanism. The performance evaluation is provided in Section 4. Conclusions are drawn in Section 5.

## 2. SYSTEM ARCHITECTURE

In this section, a cloud-based surveillance video analysis system is presented which is designed to analyze multiple surveillance video streams on the fly. As shown in Fig. 1, each stream is divided into small clips. The *clip pre-analysis module* analyzes the clips and packs clips along with metadata as small tasks. Multiple processing nodes get task one-by-one from the queue and perform the predefined video analysis algorithms such as object detection and license plate recognition. Whenever an object is detected or an unauthorized license plate is recognized, the information is inserted into a database for further handling, e.g. sending an alarm to security guards. After finishing a task, a processing node gets the next task from queue and repeats the above process. Different from [4][5][6], the proposed system does not require users to estimate the task processing time before a task being submitted to the system since we believe it is not practical. More specifically, the *workload estimation module* estimates current system workload based on the metadata provided by the clip pre-analysis module.

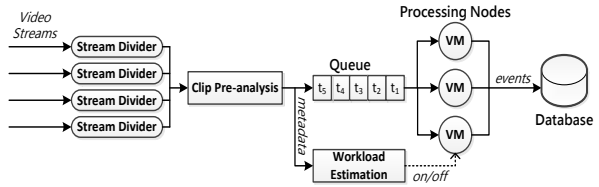


**Fig. 1.** System architecture

In general, video analysis algorithms usually have several stages. Some stages may have useful information that can be used to estimate the processing time. For example, in license plate recognition algorithm we used in this paper [8], there are two stages including (a) license plate detection and (b) character recognition. Several candidates may be identified in license plate detection stage. Actually, the license plate detection does nothing but identifying rectangle objects, thus the candidates could be just rectangle objects that are not necessary to be license plates. Even though candidates could be just rectangle objects appeared in the picture, they are all needed to be sent to character recognition stage for further processing. The number of candidates can thus be seen as an indicator for estimating the character recognition processing time. Although the license plate recognition is used as an example in this paper, the proposed mechanism can easily be applied to all object detection based video analysis algorithms.

## 3. ADAPTIVE WORKLOAD ESTIMATION

In this section, we describe how to decide the number of needed VMs for processing video clips within the deadline. Let $q_i$ represent the time task $i$ waiting in the queue and $p_i$ represent the time task $i$ is processed. The turnaround time of task $i$ can be calculated as $\delta_i=q_i+p_i$. Assume the expected turnaround time, i.e., deadline, is $\delta$, the goal is then to minimize the number of running processing nodes ($k$) while $\delta_{i\leq}\delta$ for all tasks.

```
1:  algorithm CalculateNumberOfNeededVM(k)
2:  input:
      k: the number of running VMs
3:  begin
4:    Num=k
5:    loop do
6:      MaxTurnaround=FindMaxTurnaround(Num,T,E,A)
7:      if MaxTurnaround>Threshold_H then
8:        increase Num by 1
9:      else if MaxTurnaround<Threshold_L then
10:       decrease Num by 1
11:     else
12:       goto done
13:     end if
14:   end loop
15:   label done
16:   return Num
17: end
18: function FindMaxTurnaroundTime(k,T,E,A)
19: input:
20:   k: the number of running VMs
21:   T={T_1,T_2,…,T_k}, where T_i is the expected
      release time of VM_i
22:   E={e_1,e_2,…,e_n}, where e_i is the estimated
      processing time of task i
23:   A={a_1,a_2,…,a_n}, where a_i is arrival time of
      task i
24: begin
25:   initialize Max=0
26:   for all task i
27:     find j where T_j=minimum(T)
28:     if T_j+e_i-a_i>Max then
29:       Max=T_j+e_i-a_i
30:     end if
31:     update T_j=T_j+e_i
32:   end for
33:   return Max
34: end
```

Algorithm 1. Calculate the number of needed VMs.

To achieve the goal, the workload estimation module simply simulates the task dispatching process to find the maximum turnaround time of tasks, as shown in Algorithm 1. If the maximum turnaround time is greater than a predefined threshold, denotes as $Threshold_H$, system acquires more VMs form cloud platform and does the simulation again. On the other hand, when the maximum turnaround time drops to

be lower than $Threshold_L$, system releases VMs from cloud platform.

The reason to define $Threshold_H$ and $Threshold_L$ is to prevent the system from acquiring and releasing VMs frequently since there is a cost of starting up and shutting down a VM. Normally, it takes 1~2 minutes for a just-turned-on VM to be ready for use. However, researchers of live VM cloning have shown that the time for cloning a live VM can be as fast as hundreds of milliseconds [8].

## 4. PERFORMANCE EVALUATION

In this section, experiments are conducted to evaluate the performance of the proposed mechanism. The license plate recognition is used to verify the performance of the proposed mechanism.

### 4.1 LICENSE PLATE RECOGNITION

There are two main stages: (1) license plate detection and (2) character recognition. The following describes the details.

(1) License Plate Detection: the purpose of license plate detection is to identify candidates of license plate. The algorithm described in [7] is adopted where a predefined-size window is used to scan the whole image and to decide if the scanned region is a candidate based on the following three conditions: (a) edge density, (b) width/height ratio, and (c) width and height limit.

**(2)** *Character Recognition*: to recognize license plate, each candidate is first segmented into possible character regions using the X-Y cut algorithm described in [7], which performs the x-direction scan on the candidate to find the highest and lowest horizontal positions of possible character region, and then performs a y-direction scan to find the local peak value of the vertical gray-level projection. The possible character regions can thus be determined based on highest and lowest horizontal positions and width/height ratio of the character region. Each possible character region is then processed to extract character features including Zone, Cross, Histogram and Profile. And the Support Vector Machine [9] is used to recognize the character based on character features. Finally, string grammars are applied to characters to filter out incorrect candidate, and the a voting mechanism is adopted to improve the accuracy of the character recognition.

In the experiments, 10~50 video sources of cars entering a company are used. Example images of sources are shown in Fig. 2. As most cars entering the company between 8AM to 9AM and leaving the company between 5PM to 6PM, the occurrence of license plates is likely to be bursty. The video source is encoded using H264/AVC codec, the resolution is 720x480 and the frame rate is 30 frames/sec. The time of a just-started VM to be ready for use is assumed to be 400ms, which is the suggested value of live VM cloning in [8].

**Fig. 2.** The video content of cars entering company gate.

### 4.2 EXPERIMENTAL RESULT

In the experiments, the proposed mechanism is compared with two heuristic mechanisms: (a) Fixed VM Quantity and (b) Queue-Size. In Fixed VM Quantity (FVQ) mechanism, the number of VMs is fixed. More specifically, system administrator pre-decides the number of VMs based on historical experiences or the budget. This mechanism is easy to implement but is not adaptive to the variance of system workload. In Queue-Size (QS) based mechanism, jobs in the queue are assumed to have the same processing time, which is obtained by calculating the mean value of processing time of historical jobs. That is, the mechanism adjusts the number of VMs based on the number of jobs waiting in queue. However, in video analysis, each job may have different processing time, thus this mechanism cannot capture the real system workload. For example, video clips containing suspicious objects will consume more computing power than those without suspicious objects. That is, two systems having the same number of jobs waiting in queue are not necessary to have the same workload.

Two metrics are used for performance evaluation: (a) Failure Job Ratio: the percentage of jobs that are failed to be finished before the deadline and (b) Average VM Quantity: the average number of running VMs. The deadline is set to 3 seconds in the experiments, which means once a job comes in to the system; it needs to be finished within 3 seconds including the time waiting in queue and the time it is processed.

We first evaluate the performance of Failure Job Ratio. As shown in Fig. 3, the proposed mechanism performs much better than the other two mechanisms in terms of Failure Job Ratio. There are 99.81% of jobs can be finished within the pre-defined deadline in the proposed mechanism while only 97.58% of jobs can be finished within deadline in FVQ mechanism. The QS mechanism performs even worse than FVQ mechanism which is only 93.67%~85.32% of jobs can be finished with deadline. This indicates that the proposed mechanism can successfully predicts the system workload and thus precisely adjust the number of VMs.

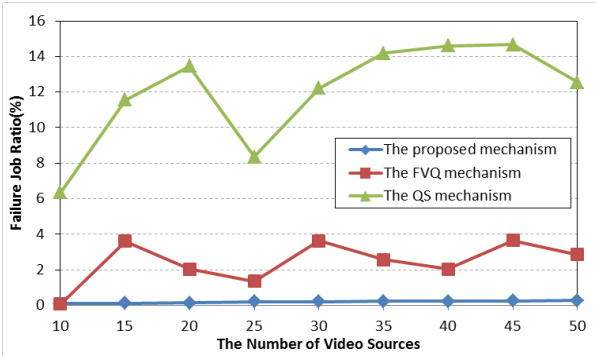


**Fig 3.** The failure job ratio of the three mechanisms under the different number of video sources.

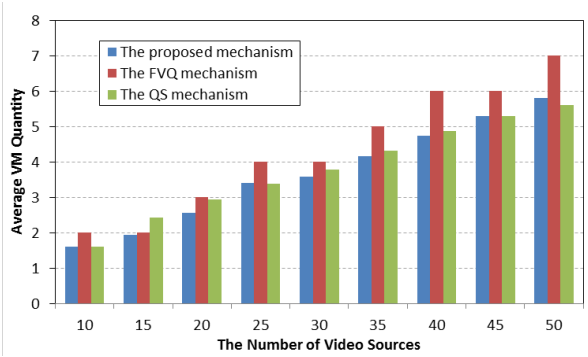


**Fig. 4.** The average VM quantity of the three mechanisms under the different number of video sources.

Then we evaluate the performance of Average VM Quantity under different number of video sources. As shown in Fig. 4, the proposed mechanism outperforms the FVQ mechanism regardless of the number of video sources. In 50 video sources case, the proposed mechanism improves 22% in terms of Average VM Quantity.

When comparing with QS mechanism, the proposed mechanism outperforms the QS mechanism by 21% and 13% in 15 video sources and 20 video sources cases, respectively. Even though in other cases, the performance of QS and the proposed mechanisms seem to be similar, the QS mechanism fails to maintain the QoS as shown in Fig. 3. However, detecting suspicious events in real-time is critical in surveillance video analysis systems.

In summary, the experimental results show that the proposed content-aware workload estimation mechanism is capable of correctly estimating the system workload. And thus the computing resources can be precisely allocated while the QoS is maintained. As the workload estimation is based on the number of suspicious objects in the images, the proposed framework can easily be applied to most of object-based surveillance video analysis systems, especially for those requiring high computing power such as dynamic video synopsis [10].

## 5. CONCLUSION

In conclusion, we have shown that with carefully design, pre-analyzing video clips can derive insightful indicators that lead to better system workload estimation. With accurate workload estimation, one can thus have a better planning on computing resource usage. While the proposed workload estimation mechanism can easily be applied to object-based video analysis algorithms, for those that are not object-based video analysis algorithms are needed to be further investigated.

## 6. REFERENCE


[1]. M. Saini, X. Wang, P. K. Atrey, and M. Kankanhalli, "Adaptive Workload Equalization in Multi-Camera Surveillance Systems," *IEEE Transactions on Multimedia*, Vol. 14, No. 3, June, 2012.

[2]. S. Lin, X. Zhang, Q. Yu, H. Qi and S. Ma, "Parallelizing Video Transcoding With Load Balancing On Cloud Computing," In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2864-2867, May, 2013.

[3]. F. Lao, X. Zhang and Z. Guo, "Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing," In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2905-2908, May, 2012.

[4]. P. Marshall, K. Keahey and T. Freeman, "Elastic Site Using Clouds to Elastically Extend Site Resources," In *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 43-52, May, 2009.

[5]. M. D. Assuncao, A. D. Costanzo and R. Buyya, "Evaluating the Cost-benefit of Using Cloud Computing to Extend the Capacity of Clusters," In *Proceedings of ACM international symposium on High Performance Distributed Computing*, pp. 141-150, June, 2009.

[6]. M. Mao and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," In *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1-12, Nov., 2011.

[7]. C. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Trans. on Intelligent Transportation System*, vol. 9, no. 3, pp. 377–391, 2008.

[8]. Y. Sun, Y. Luo, X. Wang, Z. Wang, B. Zhang, H. Chen and X. Li, "Fast Live Cloning of Virtual Machine Based on Xen," In *Proceedings of IEEE International Conference on High Performance Computing and Communications (HPCC)*, pp. 392-399, June, 2009.

[9]. C. C. Chang and C. J. Lin, "LIBSVM : a library for support vector machines," *ACM Trans. on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1-27:27, 2011.

[10]. A. Rav-Acha , Y. Pritch and S. Peleg, "Making a Long Video Short: Dynamic Video Synopsis," In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 435-441, Jun., 2006.