DOI: 10.3969/j. issn. 1673-5188. 2017. 02. 005
http://kns.cnki.net/kcms/detail/34.1294.TN.20170411.0956.002.html, published online April 11, 2017

Scheduling Heuristics for Live Video Transcoding on Cloud Edges

Panagiotis Oikonomou[1], Maria G. Koziri[1], Nikos Tziritas[2], Thanasis Loukopoulos[1], and XU Cheng-Zhong[2]
(1. University of Thessaly, Lamia 35100, Greece;
2. Research Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China)

Abstract

Efficient video delivery involves the transcoding of the original sequence into various resolutions, bitrates and standards, in order to match viewers' capabilities. Since video coding and transcoding are computationally demanding, performing a portion of these tasks at the network edges promises to decrease both the workload and network traffic towards the data centers of media providers. Motivated by the increasing popularity of live casting on social media platforms, in this paper we focus on the case of live video transcoding. Specifically, we investigate scheduling heuristics that decide on which jobs should be assigned to an edge mini-datacenter and which to a backend datacenter. Through simulation experiments with different QoS requirements we conclude on the best alternative.

Keywords

video transcoding; edge computing; scheduling; heuristics; x264

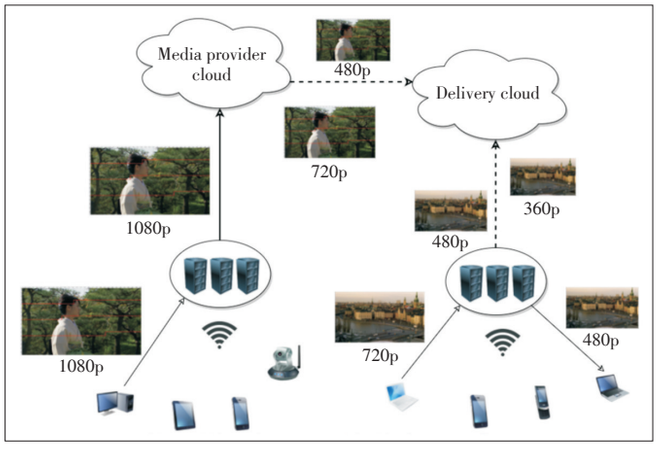
1 Introduction

Modern applications built on top of an integrated Internet of Things (IoT) environment [1], together with Cyber Physical Systems (CPSs) [2], involve heavy video traffic, e.g., in smart vehicle traffic management. At the same time, the proliferation of smart mobile devices carrying cameras of continuously higher resolution, together with the explosive growth in the popularity of social media platforms, poses great challenges in cloud resource management. As an indication, Cisco reported in [3] that during 2015, mobile Internet traffic experienced a growth of 74%, the majority of which (>50%) was video transmissions. Therefore, minimizing video related network traffic becomes of paramount importance.

Video coding is the process of compressing a raw video sequence using some standards. Examples of such standards are H.264/AVC [4] which is the most popular (but aging) standard currently in use, High Efficiency Video Coding (HEVC) [5] and VP9 [6], which are newer standards achieving higher compression ratios compared to H.264/AVC. Although video coding is a computationally demanding task, it is usually performed at the point where the initial video is captured (camera, smart device etc.), often with the aid of specialized hardware. Thus, the initial coding of a video sequence does not hinder a cloud based social media platform (SMP) computationally wise and the only overhead is the consumed bandwidth for uploading. However, in order to be able to deliver the video sequence to a variety of clients differing in screen resolutions, decoders and network capabilities, the originally uploaded sequence must be encoded into multiple output sequences of various resolutions, bitrates, quality levels and perhaps coding standards. This process is called transcoding and burdens computationally and network-wise the SMP's cloud. In particular, the case of live casting offers the most challenges since real time performance is a requirement.

Motivated by the above, we investigate the case where an SMP can take advantage of mini-data centers existing at network edges in order to offload live transcoding jobs, thus, saving resources and bandwidth. **Fig. 1** illustrates an example whereby two broadcasts are performed, one at 1080p and the other at 720p from two different edges. In the first case (1080p), the sequence is not transcoded at the edge but transmitted to one of the SMP's data centers for processing. Then two different outputs (720p and 480p) are sent to some Content Delivery Network (CDN). In contrast to this, the other input sequence (720p) is transcoded into two output sequences at the edge. Copies of the outputs are sent to the CDN and also used to satisfy local demands (480p). Clearly, the second alternative of using edge transcoding reduces both the processing and network resource consumption at the SMP's cloud.

In this paper, we tackle the associated scheduling problem



▲Figure 1. Example system model with edge transcoding.

induced by the scenario of Fig. 1. Namely, given edge resources and the characteristics of arriving transcoding tasks, task-server assignment must be made so that the percentage of tasks not processed by the edge (satisfied with overhead by SMP's Cloud) is minimized. We evaluate different scheduling heuristics for the scheduling problem under the constraint that each assigned task must obtain the required processing power to exhibit real time behavior. We then examine the case where the aforementioned constraint is softened, allowing for some quality loss in order to increase the number of tasks assigned to the edge. All heuristics are evaluated using a dataset of Twitch broadcasts [7] and realistic values for transcoding job characteristics obtained by using x264 codec [8] over class B and class A common test video sequences [9].

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the problem formulation. Heuristics are illustrated in Section 4 and evaluated in Section 5. Finally Section 6 concludes the paper.

2 Related Work

The number of transcoding tasks hosted by edges is dictated by their processing requirements. Related to this requirement is research concerning speeding up of video coding and transcoding. An avid research exists on parallelizing video coding with approaches varying from coarse grained parallelism, whereby parallelism is considered at the level of group of Macroblocks (H.264/AVC) or Coding Tree Units (CTUs in HEVC), to finer grained parallel approaches implementable within a block of pels. Examples of coarse grained parallelization include slices, tiles and wavefront in the HEVC standard. Efficient implementation of these parallel options are described in [10] for slices, [11] for tiles and [12] for wavefront. Fine grained techniques usually consist of applying the Single Instruction Multiple Data (SIMD) paradigm at various levels of the encoding [13] and decoding stages [14].

As far as transcoding is concerned, a straightforward method is to first decode fully the input sequence, scale its resolution and then re-encode it. More efficient approaches target at utilizing the information already coded in the input, most noticeably the one concerning motion estimation, in order to reduce the search space when transcoding to another standard. Example works in the area include [15] where an H.264/AVC to HEVC transcoding architecture is presented that achieves a nominal speedup reaching 8x, when compared to re-encoding from scratch. If a bitrate change rather than a change in standard is needed, the process is often referred to as transrating. A survey on fast transrating methods can be found in [16]. In the experiments we obtained transcoding task weights by using the straightforward approach of re-encoding without using the information already coded. This was done both for reasons of simplicity and due to code availability (ffmpeg and x264 used). However, based on the aforementioned research we scaled the values obtained to depict the case where a more efficient transcoder is used.

Concerning cloud transcoding, most works focused on providing job scheduling techniques at the level of a server cluster or a data center. In [7], the authors considered the case of live video transcoding and proposed an integer linear program (ILP) formulation to tackle scheduling decisions. An online algorithm that schedules jobs among the servers of a datacenter with the target of satisfying delay requirements while using minimum energy was proposed in [17]. In [18] the scope was a single cluster and the optimization target was to keep the servers load balanced. In [19] an admission control algorithm was developed that differs or rejects requests that cannot be satisfied based on current workload. It is worth noting that this is the contrary approach to the one used in this paper for the case of edges, whereby it might be viable to reduce quality by over-assigning tasks to servers if the relevant benefits from edge processing are deemed sufficient. Finally, in [20] a combined caching and transcoding approach is discussed, whereby transcoding jobs are partially processed to allow for efficient caching. The target considered in this paper, i.e., live transcoding excludes partial transcoding as an option. Caching and replication techniques in the cloud are surveyed in [21], while [22] and [23] concern efficient video delivery.

Overall, compared to [7], [17], [18], [19] and [20], we differ in scope since we examine transcoding at edges while we view [21], [22] and [23] as orthogonal to our approach. Perhaps the closest work in the literature is [24], where system architecture for edge transcoding is described. Nevertheless, scheduling issues were not tackled in the manner done in this paper.

3 Problem Definition

We consider the case of a media provider receiving requests for live video casting, whereby the input stream must be transcoded into a set of output streams with different resolution, bitrate and quality demands. We consider two options for the set

of transcoding tasks associated with each input. Either they are all assigned to a mini-datacenter existing at the edge of the network or they are all assigned to the backend main datacenter of the media provider. Clearly, if the tasks are processed at the edge, the processing workload at the backend datacenter is reduced and the network overhead for transmitting the input sequence is avoided.

Let the mini-datacenter consist of S servers, with S_i denoting the ith of them, assuming a total ordering ($1 \leqslant i \leqslant S$). Each server has an associated processing capacity (let C_i), which denotes the number of baseline transcoding tasks that can be processed concurrently at real time. Baseline tasks are the ones requiring the minimum power to process. Let B_j be the jth broadcast, assuming an ordering of the B total broadcasting events ($1 \leqslant j \leqslant B$). Similarly, let s_j and d_j be the arrival time and duration of B_j, respectively. Each broadcast entails a set of transcoding tasks. Let T be the total number of transcoding tasks for all broadcasts, and T_k be the kth such task, assuming a total ordering of them ($1 \leqslant k \leqslant T$). We represent whether T_k is a task of B_j or not, using a Boolean matrix A of $B \times T$ size, whereby $A_{jk}=1$ if and only if (iff) B_j has task T_k and 0 otherwise. Moreover, W_k depicts the relevant weight of T_k in processing terms over the baseline task. Put in other terms, W_k shows how much more computationally demanding T_k is, compared to the baseline scenario. Last, let X be an $S \times T$ Boolean matrix used to encode task server assignments as follows: $X_{ik}=1$ iff T_k is assigned for processing at S_i, otherwise $X_{ik}=0$. We assume that once assigned, a task cannot be preempted and will remain for the whole duration $[s_j, \ldots, s_j+d_j]$. We consider that we want to optimize the system starting from a clean state (no task assignments exist) over a time frame divided into E equally sized slots (s_j and d_j values are now measured in time slot terms). Let e_t be the tth such time slot, with a corresponding assignment matrix X^t. We typically formulate the problem as follows: Find all values in the E total matrices X^t, so that the objective function f given in (1) is maximized:

$$f = \sum_{t=1}^{E} \sum_{i=1}^{S} \sum_{k=1}^{T} X_{ik}^{t} (1 - X_{ik}^{t-1}), \tag{1}$$

subject to the following constraints:

$$\left(\sum_{i=1}^{S} \sum_{k=1}^{T} A_{jk} X_{ik}^{t} - \sum_{k=1}^{T} A_{jk} \right) \sum_{i=1}^{S} \sum_{k=1}^{T} A_{jk} X_{ik}^{t} = 0, \quad \forall j, \ t = s_j, \tag{2}$$

$$X_{ik}^{t} = X_{ik}^{t+1}, \quad \forall i,k, \ t | s_j \leqslant t < s_j + d_j \wedge A_{jk} = 1, \tag{3}$$

$$\sum_{i=1}^{S} \sum_{k=1}^{T} A_{jk} X_{ik}^{t} = 0, \quad \forall j, \ t | t < s_j \vee t > s_j + d_j, \tag{4}$$

$$\sum_{k=1}^{T} X_{ik}^{t} W_k \leqslant C_i, \quad \forall i, \ t, \tag{5}$$

$$\sum_{i=1}^{S} X_{ik}^{t} \leqslant 1, \quad \forall k,t . \tag{6}$$

The objective function encodes the tasks that will be assigned to the edge. Eqs. (2)–(6) give the main constraints of the problem. Constraint (2) states that either all tasks of a broadcast B_j will be assigned to the edge at the time the broadcast arrives or none. Constraint (3) enforces that the decision taken for a transcoding task at the time of its broadcast arrival remains for the duration of the broadcast. Constraint (4) ensures that neither before a broadcast arrival, nor after its end time, can a corresponding task be scheduled for edge transcoding. Constraint (5) dictates that a server can exceed its capacity at no point in time. Finally, (6) states that a task can only be scheduled at one server.

Clearly, the fact that broadcasts are known in advance reduces the applicability of the presented problem formulation to cases of prescheduled event covering, e.g., sports. Nevertheless, the formulation provides a thorough definition of the optimization target and the related constraints. These remain the same both in the static problem variation presented and in the dynamic case. A last note concerns complexity. It can be shown that the relevant decision problem is NP-complete since the processing capacity constraint at the servers effectively introduces a (0, 1) Knapsack component. Next, we present heuristics for dynamic scheduling of transcoding tasks at the network edge.

4 Scheduling Heuristics

4.1 Scheduling with Tight Task QoS Requirements

The proposed heuristics tackle the dynamic version of the scheduling problem presented in the previous section. Specifically, upon the arrival of a broadcast request, the necessary transcoding tasks are defined. Then, they are sorted according to their weight and considered either in increasing order (MIN policy) or in decreasing (MAX policy). Each task is assigned to a server (using one of the policies described in the sequel) provided the task computational demands can be met by the server as per (5). If a suitable server is found for every transcoding task of the broadcast under consideration, the assignments are committed; otherwise, even if one task fails to find a hosting server, all the tasks are sent to the SMP's datacenter for processing. The assignment policies considered are based on the well-known bin-packing heuristics:

- Best Fit (BF): Select the server where the remaining capacity, left after task assignment, is the minimum possible.
- Worst Fit (WF): Similar to BF only that the server with the maximum remaining capacity will be selected.
- First Fit (FF): The first server where the task fits will be selected.

The corresponding heuristics are named after the order with which the task list is considered and the packing method followed. For instance MAX-BF refers to the heuristic that considers the heaviest task first and assigns it using Best Fit.

4.2 Scheduling with Relaxed Task QoS Requirements

The motivation for the relaxed QoS case is the following. As-

sume that all but one task of a broadcast could fit to the available servers of the edge. With strict QoS requirements, none of these tasks will be assigned. However, it might be possible to assign the remaining task to one server so that its processing capacity is exceeded by a very small margin. In practice, this means that all the tasks processed by this server will exhibit a small quality drop. For instance, if a broadcaster transmits at 30 fps (frames per second) then a 3.3% drop at the processing rate of one of its transcoding tasks means that roughly the output stream will be at 29 fps. Depending on decoder characteristics, such a drop might not even be noticeable by a human viewer. Assuming that p denotes the maximum percentage of allowable performance drop, (5) becomes:

$$\sum_{k=1}^{T} X_{ik}^{t} W_k \leqslant (1+p) C_i, \quad \forall i, t . \tag{7}$$

The heuristics first attempt to allocate all the tasks of a broadcast as per Section 4.1. In case a task does not fit, it is considered for assignment using (7) as server capacity constraint and one of the below described policies.

- Min Quality Decrease (MQD): Selects the server that incurs the minimum proportional capacity violation (equivalent to asking for the minimum quality penalty for its hosted tasks).
- First Fit (FF): The first server where the task fits as per (7) will be selected.
- View Weighted Penalty (VWP): Weights the quality penalty of each task by the number of its viewers. The server with the minimum aggregated weighted quality penalty value is selected.

5 Experiments

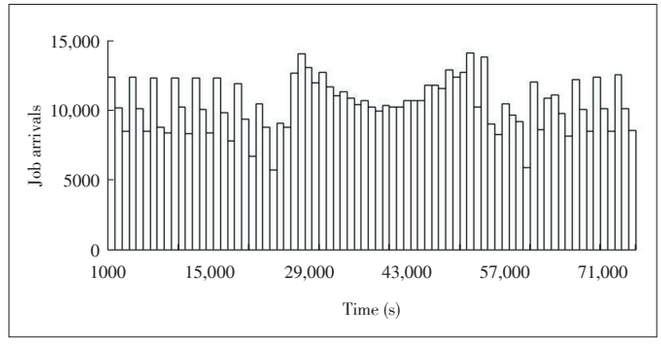
5.1 Setup

To simulate broadcasting activity, we used the same dataset from Twitch as the one described in [7]. We kept the portion of the dataset representing one day activity (Jan. 6th, 2014). We then filtered it by deleting entries with broadcasts having no viewers and the broadcasts of resolution less than 220p. To keep the simulation time manageable we considered the following 5 resolutions: 240p, 360p, 480p, 720p and 1080p. In case a broadcast in the trace did not follow one of the previously mentioned resolutions, we clustered it to its closest matching. We assumed that a broadcast must be transcoded to all the resolutions that were lower than the one it used. Clearly, with this setting the maximum number of transcoding tasks incurred by a broadcast is 4, corresponding to a 1080p stream that must be downscaled to 720p, 480p, 360p and 240p. Upscaling was not considered in the experiments. Finally, for simulation purposes we assumed that all videos used 30 fps. Furthermore, the recorded in the dataset viewing demand was split equally among the resolutions used by a broadcast, i.e., the input and all lower ones. **Table 1** summarizes some of the dataset characteristics, while **Fig. 2** plots the broadcasting job arrival rates as a histogram of a 1000 seconds (s) step. As it can be seen, the arriving jobs do not exhibit sharp peaks (at least with the used interval), but the distribution is rather uniform. This favors job scheduling at edges since it makes sizing decisions for edges less demanding. However, duration of broadcasts does not follow a similar trend. As noted in Table 1, the difference between the average and maximum duration is two orders of magnitude, implying a heavy tailed distribution. This hinders scheduling decisions, since it means that duration estimation will be hard to achieve in the general case. For this reason, none of the scheduling heuristics described in Section 4 uses such estimates.

Next, we needed to characterize the weights of the transcoding tasks. For this reason we used class A and class B common test video sequences and transcoded them to the levels for which we wanted to obtain weight values. To do so, a sequence was first fully decoded, then scaled to the desired resolution using ffmpeg and then encoded using x264. The encoding settings followed the Peak Signal to Noise Ratio (PSNR) tailored scenario of [25], which aims at maximizing quality in PSNR terms. The exact parameters are given below (Kimono example): x264 --input-depth 8 --frames 0 --input-res 1920x1080 --fps 24 --input-csp i420 --log-level debug --tune psnr --psnr --profile high --preset placebo --keyint 96 --min-keyint 96 --me

▼**Table 1. Dataset for broadcasters (general characteristics)**

Name	Value
Dataset duration	75,079 s
Average broadcast duration	23,263.4 s
Max broadcast duration	1.05E6 s
Number of broadcasts	786,100
Total transcoding tasks	1,244,450
Percentage of braodcasts at 1080p	26.21%
Percentage of braodcasts at 720p	53.24%
Percentage of braodcasts at 480p	11.18%
Percentage of braodcasts at 360p	7.49%
Percentage of braodcasts at 240p	1.86%



▲**Figure 2. Histogram for broadcasting arrival rates.**

umh --merange 240 --ref 4 --partitions all --threads 1 --subme 9 --aq-mode 0 --aq-strength 0.0 --psy-rd 0.0 --output kimono_out.264 Kimono_in.yuv.

Table 2 summarizes the general characteristics of the test sequences, together with the coding time for each targeted resolution, measured as the number of frames per second processed by the codec. The case of 240p forms the baseline transcoding scenario, with the remaining resolutions assigned proportional weights.

Having defined the time of the baseline scenario and task weights accordingly, next we define server capacity in the following manner. In order to fully control the system environment we used a dedicated server for which we had full ownership. The server used for the x264 coding jobs carried two 6-core Intel Xeon E5-2630 CPUs running at 2.3 GHz. Since the coding speeds at Table 2 used one thread and the nominal rate considered for the simulation is 30 fps, each core of the server accounts for a processing capacity of 21.6/30 (the baseline scenario). The total server capacity is then calculated by multiplying with 12 (the total number of physical cores) and equals 8.64. Since our server setting is not of generic use, we translate it into one of the Amazon EC2 instances [26] to make our simulation setting more applicable. Specifically, we consider the C3 instances which are recommended for video coding. Comparing the processor passmark ratios between the CPU of our server and the one used in the C3 instances, i.e., Intel Xeon E5-2680 v2 (Ivy Bridge), it can be estimated that the instance c3.4xlarge will account for a speedup of 2.2x compared to our server. Since video coding is CPU-bound, the aforementioned methodology (i.e., comparing CPUs) provides a good estimation on relative performance. Last, we consider the case where specialized transcoding software is available, which accounts for a speedup of 8x (same as in [15]) compared to the simple methodology used to obtain the processing rates of Table 2.

5.2 Results for Tight QoS Requirements

Here we present results for the case where the assigned transcoding tasks at the edge must be satisfied at their nominal rate (30 fps). We consider two scenarios. In the first (homogeneous), 1000 servers each of capacity described in Section 5.1 exist in the micro datacenter of the edge, while in the second scenario (heterogeneous) 500 servers have the aforementioned capacity and 500 half of it (presumably equivalent to a c3.2xlarge EC2 instance). **Fig. 3** plots the performance of the scheduling heuristics, measured in terms of the percentage of broadcasts that are assigned for edge transcoding. Results show that the same trends are exhibited both in the homogeneous and the heterogeneous cases. The later achieves lower performance since it accounts for smaller total capacity. Furthermore, sorting the transcoding tasks of a broadcast has marginal effect. This is presumably due to the fact that the tasks are scheduled as soon as a broadcast arrives and are rather small in number, compared to the available servers. Clearly the WF policy outperforms the other alternatives by a substantially large margin (an extra 5% roughly of the arriving requests can be accommodated by the edge).

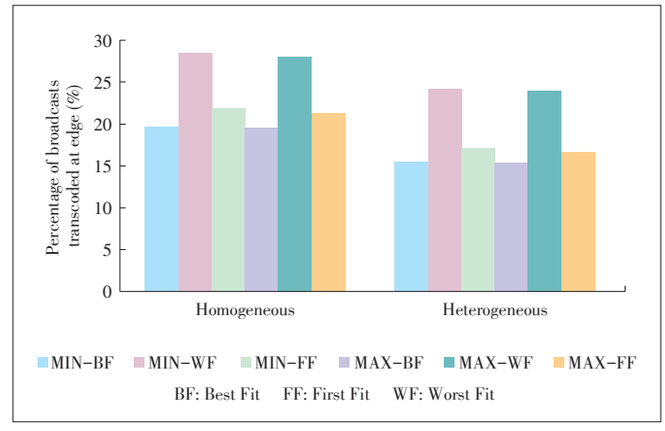
Having identified WF to be the most promising heuristic, we evaluated the impact of the arrival rate on the achievable performance. To do so, we used the same trace with above, but sampled it every 1, 2 and 3 entries. Clearly, a sampling rate of 1 is equivalent to using the whole dataset (Fig. 3), while 2 and 3 effectively account for 1/2 and 1/3 of the arrival rate. **Fig. 4** plots the performance of the WF scheme for the three arrival rates and for both the homogeneous and heterogeneous cases. As expected, the percentage of jobs that can be satisfied by the edge increases as the arrival rate decreases. It is worth noting that with 1/3 arrival rate which accounts for roughly 262,000 daily requests, roughly 60% of them (homogeneous case) can be satisfied by the edge. This translates for great load reduction at the back end datacenters.

▼**Table 2. Video sequences used for weight calculation**

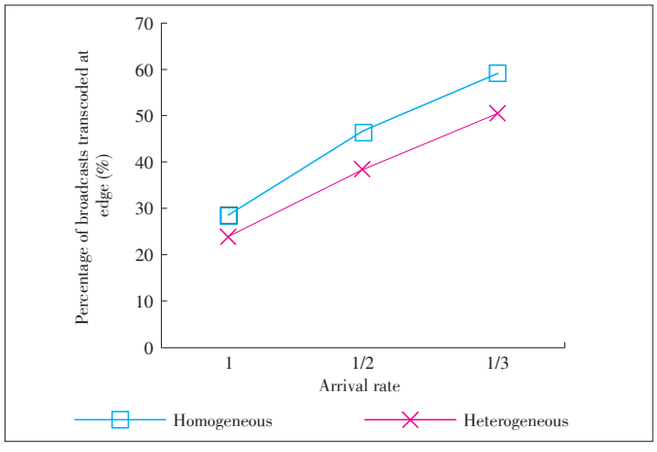
Name	Resolution	Frames	Time 240p (fps)	Time 360p (fps)	Time 480p (fps)	Time 720p (fps)
BasketballDrive	1920×1080	500	15.78	5.37	3.78	1.47
BQTerrace	1920×1080	600	29.10	9.11	6.42	2.37
Cactus	1920×1080	500	25.48	8.77	5.09	1.96
Kimono	1920×1080	240	18.54	6.05	4.33	1.62
ParkScene	1920×1080	240	24.95	7.62	5.28	1.94
PeopleOnStreet	2560×1600	150	10.01	2.82	2.00	0.74
Traffic	2560×1600	150	27.36	8.06	5.77	2.17
Average	-	-	21.60	6.82	4.66	1.75
Weights	-	-	1.00	3.16	4.63	12.34

5.3 Results for Soft QoS Requirements

We consider that the processing requirement of real time performance is relaxed as per Section 4.2. We evaluated the performance of the algorithms of Section 4.2 when combined



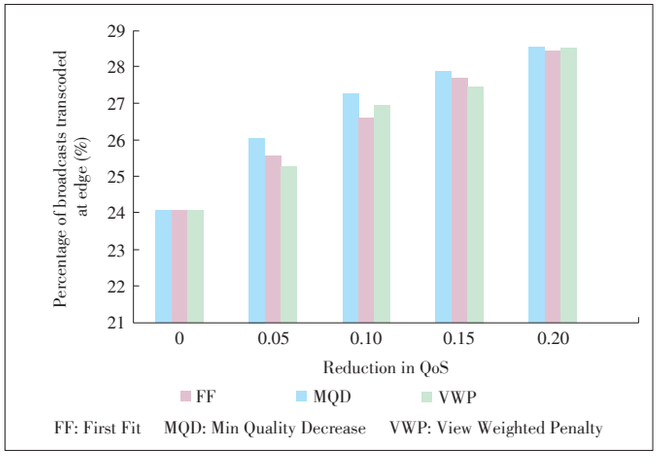
▲**Figure 3: Percentage of broadcasts processed by the edge (1000 servers, full dataset). Two different cases: homogeneous and heterogeneous.**



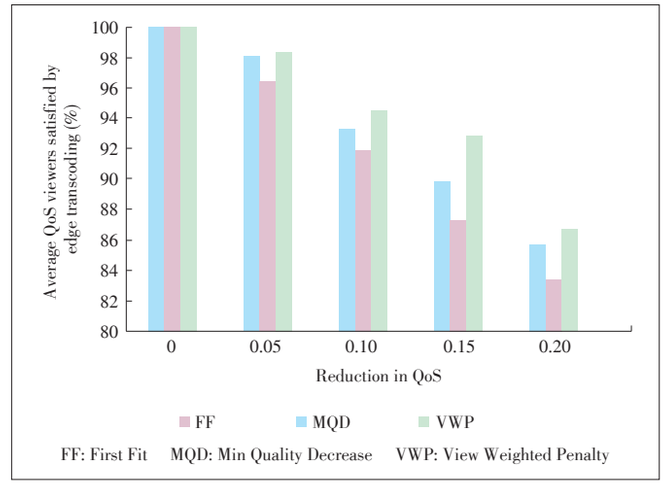
▲**Figure 4. Percentage of broadcasts processed by the edge for decreasing arrival rate (1000 servers).**

with MAX-WF for various levels of QoS reduction, namely: 0% (no reduction - real time performance), 5%, 10%, 15% and 20%. **Fig. 5** plots the percentage of broadcasts having their transcoding jobs assigned to the micro datacenter, for the heterogeneous case used in Figs. 3 and 4 and with the whole trace as input. All three heuristics assign to the edge an increasing number of jobs as QoS requirements are reduced. Among the algorithms, MQD achieves the best performance, followed by VWP and FF. As it can be observed, with a 5% decrease in processing rate, an extra 2% of jobs can be assigned to the edge, while with 20% an extra 4.5%. Although a 20% reduction seems impractical at first glance, it roughly means that instead of processing a stream at 30 fps, the stream might be processed at 24 fps. It is worth noting that this is the lowest rate for 1080p TV sets. Overall, by relaxing the nominal real time processing rate for transcoding jobs, significant extra load could be offset from back end datacenters.

In order to further quantify the impact of QoS reduction, in **Fig. 6** we plot the average viewing quality (for the viewers satisfied by the edge) as a percentage of the achieved fps when compared to real time 30 fps. VWP achieves the best performance, which for a 15% allowable reduction (0.15 point in x-axis) leads to an average viewing quality of 93%. Put it in other terms the average processing rate will be almost 28 fps. For the same QoS reduction (15%), Fig. 5 depicts that VWP assigns an extra ~3.5% of jobs, or roughly 27,000 more broadcasts at the edge micro datacenter. Thus, an interesting tradeoff is present whereby VWP can offset substantial load towards the edge micro datacenter at only a small decrease on average viewing quality.



▲**Figure 5. Percentage of broadcasts processed by the edge for varying QoS reduction percentages (full dataset, heterogeneous servers).**



▲**Figure 6. Average QoS of viewers as the allowable reduction in QoS for edge transcoding jobs is increased (full dataset, heterogeneous servers).**

6 Conclusions

In this paper we examined scheduling heuristics for the problem of assigning live transcoding jobs at an edge micro datacenter. We considered two main cases. The first accounts for real time performance, while the second allows small quality degradation on the output video streams in order to increase the assigned jobs to the micro datacenter. Through simulation experiments using a realistic dataset, it is concluded that interesting tradeoffs can be obtained by a method (VWP) that takes into account viewer perceived QoS.

References

[1] J. Gubbi, R Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sept. 2013. doi: 10.1016/j.future.2013.01.010.

[2] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyber physical systems: a survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, Jun. 2015. doi: 10.1109/JSYST.2014.2322503.
[3] Cisco Systems Inc. (2017, Jan. 30). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020* [Online White Paper]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html
[4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003. doi: 10.1109/TCSVT.2003.815165.
[5] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012. doi: 10.1109/TCSVT.2012.2221191.
[6] D. Grois, D. Marpe, A. Mulayoff, *et al.*, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," in *Picture Coding Symposium (PCS)*, San Jose, USA, Dec. 2013, pp. 394–397. doi: 10.1109/PCS.2013.6737766.
[7] R. A. Pardo, K. Pires, A. Blanc, and G. Simon, "Transcoding live adaptive video streams at a massive scale in the cloud," in *ACM SIGMM Conference on Multimedia Systems (MMSys)*, Portland, USA, Mar. 2015, pp. 49–60. doi:10.1145/2713168.2713177.
[8] VideoLAN. (2017, Jan. 30). *x264 home page* [Online]. Available: http://www.videolan.org/developers/x264.html
[9] F. Bossen, "Common test conditions and software reference configurations," JCT-VC, San Jose, USA, Document: JCTVC-H1100, Feb. 2012.
[10] M. G. Koziri, P. Papadopoulos, N. Tziritas, *et al.*, "Slice-based parallelization in HEVC encoding: realizing the potential through efficient load balancing," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Montreal, Canada, Sept. 2016, pp. 1–6. doi: 10.1109/MMSP.2016.7813354.
[11] M. Shafique, M. U. K. Khan, and J. Henkel, "Power efficient and workload balanced tiling for parallelized high efficiency video coding," in *IEEE International Conference on Image Processing (ICIP)*, Paris, France, Oct. 2014, pp. 1253–1257. doi: 10.1109/ICIP.2014.7025250.
[12] C. C. Chi, M. A. Mesa, B. Juurlink, *et al.*, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012. doi: 10.1109/TCSVT.2012.2223056.
[13] Y.-J. Ahn, T.-J. Hwang, D.-G. Sim, and W.-J. Han, "Implementation of fast HEVC encoder based on SIMD and data-level parallelism," *EURASIP Journal Image and Video Processing*, vol. 16, Dec. 2014. doi: 10.1186/1687-5281-2014-16.
[14] M. G. Koziri, D. Zacharis, I. Katsavounidis, and N. Bellas, "Implementation of the AVS video decoder on a heterogeneous dual-core SIMD processor," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 673–681, May 2011. doi: 10.1109/TCE.2011.5955207.
[15] J. F. Franche and S. Coulombe, "Fast H.264 to HEVC transcoder based on post-order traversal of quadtree structure," in *IEEE International Conference on Image Processing (ICIP)*, Quebec, Canada, Sept. 2015, pp. 477–481. doi: 10.1109/ICIP.2015.7350844.
[16] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005. doi: 10.1109/TMM.2005.854472.
[17] W. Zhang, Y. Wen, J. Cai, and D. O. Wu, "Toward transcoding as a service in a multimedia cloud: energy-efficient job-dispatching algorithm," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2002–2012, Jun. 2014. doi: 10.1109/TVT.2014.2310394.
[18] S. Lin, X. Zhang, Q. Yu, *et al.*, "Parallelizing video transcoding with load balancing on cloud computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Beijing, China, May 2013, pp. 2864–2867. doi: 10.1109/ISCAS.2013.6572476.
[19] A. Ashraf, F. Jokhio, T. Deneke, *et al.*, "Stream-based admission control and scheduling for video transcoding in cloud computing," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Delft, Netherlands, May 2013, pp. 482–489. doi: 10.1109/CCGrid.2013.21.
[20] G. Gao, W. Zhang, Y. Wen, *et al.*, "Towards cost-efficient video transcoding in media cloud: insights learned from user viewing patterns," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1286–1296, Aug. 2015. doi: 10.1109/TMM.2015.2438713.
[21] S. U. R. Malik, S. U. Khan, S. J. Ewen, *et al.*, "Performance analysis of data Intensive cloud systems based on data management and replication: a survey," *Distributed and Parallel Databases*, vol. 34, no. 2, pp. 179–215, Jun. 2016. doi: 10.1007/s10619-015-7173-2.
[22] W. Ji, Z. Li, and Y. Chen, "Joint source-channel coding and optimization for layered video broadcasting to heterogeneous devices," *IEEE Transactions on Multimedia*, vol. 14, no. 2, pp. 443–455, Apr. 2012. doi: 10.1109/TMM.2011.2177645.
[23] W. Ji, Z. Li, and Y. Chen, "Content-aware utility-fair video streaming in wireless broadcasting networks," in *IEEE International Conference on Image Processing (ICIP)*, Brussels, Belgium, Sept. 2011, pp. 145–148. doi: 10.1109/ICIP.2011.6115717.
[24] M. T. Beck, S. Feld, A. Fichtner, *et al.*, "ME-VoLTE: network functions for energy-efficient video transcoding at the mobile edge," in *International Conference on Intelligence in Next Generation Networks (ICIN)*, Paris, France, Feb. 2015, pp. 38–44. doi: 10.1109/ICIN.2015.7073804.
[25] J. De Cock, A. Mavlankar, A. Moorthy, and A. Aaron, "A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications," *SPIE Applications of Digital Image Processing XXXIX*, vol. 9971, 997116. Sept. 2016. doi: 10.1117/12.2238495.
[26] Amazon Web Services. (2017, Jan. 30). *Amazon EC2 Instance Types* [Online]. Available: https://aws.amazon.com/ec2/instance-types/

Manuscript received: 2017-02-09

Biographies

Panagiotis Oikonomou (paikonom@uth.gr) received his Diploma degree (2008) and M.Sc. degree (2010) from the Department of Electrical and Computer Engineering, University of Thessaly, Greece. He is currently a Ph.D. candidate at the same Department. His research interests include optimization algorithms and fuzzy logic methods.

Maria G. Koziri (mkoziri@uth.gr) received her Diploma degree in computer engineering from the Technical University of Crete, Greece in 2003 and Ph.D. degree in computer science from the University of Thessaly, Greece in 2007. She is currently a visiting lecturer in the Computer Science Department of the University of Thessaly. Her research interests include video compression, scalable video coding, rate-distortion optimization and computer architecture.

Nikos Tziritas (nikolaos@siat.ac.cn) received his B.Sc. degree from the Technological Educational Institute of Serres, Greece in 2004, and M.Sc. and Ph.D. degrees from the University of Thessaly, Greece in 2006 and 2011, respectively. He is currently an associate professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. His work has appeared in over 35 publications. He is the recipient of the Award for Excellence for Early Career Researchers in Scalable Computing from IEEE Technical Committee in Scalable Computing in 2016.

Thanasis Loukopoulos (luke@dib.uth.gr) received his Ph.D. degree in computer science from the Hong Kong University of Science and Technology, China. He is currently a lecturer at the Department of Computer Science and Biomedical Informatics of the University of Thessaly, Greece. His research interests are in green computing, cloud computing, WSNs, scheduling, load balancing and video coding parallelization. His work appeared in over 50 publications. He had the best paper award in ICPP 2001.

XU Cheng-Zhong (cz.xu@siat.ac.cn) received the Ph.D. degree in computer science from the University of Hong Kong, China in 1993. He is currently a professor in the Department of Electrical and Computer Engineering of Wayne State University, China and the director of Cloud and Internet Computing Laboratory (CIC) and Sun's Center of Excellence in Open Source Computing and Applications (OSCA). His research interest is mainly in scalable distributed and parallel systems and wireless embedded computing devices. He has published two books and more than 160 articles in peer-reviewed journals and conferences in these areas.