Computers, Materials & Continua
DOI:10.32604/cmc.2022.023630
Article

Tech Science Press

FSpot: Fast and Efficient Video Encoding Workloads Over Amazon Spot Instances

Anatoliy Zabrovskiy[1,3], Prateek Agrawal[1,2,*], Vladislav Kashansky[1], Roland Kersche[4], Christian Timmerer[1,4] and Radu Prodan[1]

[1]University of Klagenfurt, Klagenfurt, 9020, Austria
[2]Lovely Professional University, Phagwara, 144411, India
[3]Petrozavodsk State University, Petrozavodsk, 185035, Russia
[4]Bitmovin, Klagenfurt, 9020, Austria
*Corresponding Author: Prateek Agrawal. Email: dr.agrawal.prateek@gmail.com
Received: 14 September 2021; Accepted: 18 November 2021

Abstract: HTTP Adaptive Streaming (HAS) of video content is becoming an undivided part of the Internet and accounts for most of today's network traffic. Video compression technology plays a vital role in efficiently utilizing network channels, but encoding videos into multiple representations with selected encoding parameters is a significant challenge. However, video encoding is a computationally intensive and time-consuming operation that requires high-performance resources provided by on-premise infrastructures or public clouds. In turn, the public clouds, such as Amazon elastic compute cloud (EC2), provide hundreds of computing instances optimized for different purposes and clients' budgets. Thus, there is a need for algorithms and methods for optimized computing instance selection for specific tasks such as video encoding and transcoding operations. Additionally, the encoding speed directly depends on the selected encoding parameters and the complexity characteristics of video content. In this paper, we first benchmarked the video encoding performance of Amazon EC2 spot instances using multiple ×264 codec encoding parameters and video sequences of varying complexity. Then, we proposed a novel fast approach to optimize Amazon EC2 spot instances and minimize video encoding costs. Furthermore, we evaluated how the optimized selection of EC2 spot instances can affect the encoding cost. The results show that our approach, on average, can reduce the encoding costs by at least 15.8% and up to 47.8% when compared to a random selection of EC2 spot instances.

Keywords: EC2 spot instance; encoding time prediction; adaptive streaming; video transcoding; clustering; HTTP adaptive streaming; MPEG-DASH; cloud computing; optimization; Pareto front

1 Introduction

Nowadays, most Internet traffic represents multimedia content, such as live or on-demand audio and video streaming [1]. The streaming experience over the Internet depends on several factors like user location, network speed, traffic congestion, or end-user device, which significantly vary over time [2]. Streaming platforms and services use the HAS technology [3] to adapt to these bandwidth variations that provide video sequences in multiple bitrates. The resolution pairs are divided into short-term video and audio segments (e.g., 2 to 10 s), individually as requested by a client device depending on its technical conditions (e.g., screen size, network performance) in a dynamic, adaptive manner [4]. Client devices and video players use segment bitrate selection (or rate adaptation) algorithms to optimize the user experience [5,6]. The widely used MPEG-DASH HAS implementation allows streaming providers to choose from a set of codecs for video encoding due to its codec independent [3] characteristic, including Advanced Video Coding (AVC) [7], High-Efficiency Video Coding (HEVC) [8], VP9 [9], AOMedia Video 1 (AV1) [10] and Versatile Video Coding (VVC) [11]. However, encoding video segments for adaptive streaming is a computationally-intensive process that can take seconds or even days depending on many technical aspects, such as video complexity or encoding parameters [12] and typically requires expensive high-performance computers.

Currently, most streaming services and video encoding platforms opt for less expensive and more scalable cloud resources (e.g., Amazon Web Services (AWS), Google Cloud, Microsoft Azure) rented on demand [13,14], deployed worldwide on low-latency geo-distributed infrastructures [15,16]. Amazon EC2 currently operates in eighteen geographical locations and provides different instances for general purposes (m instances), compute-optimized (c instances), memory-optimized (r instances), or burstable (t instances) [17]. EC2 spot instances are unused spare compute capacity in the AWS cloud available at a high discount compared to on-demand prices, with the limitation is that AWS can stop them at any time upon a two-minute warning. While modern encoding platforms and services can significantly leverage spot instances to reduce their encoding costs, the unavailability of intelligent models to estimate the video encoding time and costs makes the correct selection of the cloud instances for thousands of encoding tasks still critical [13,18]. Cloud infrastructures, dedicated servers and Internet of Things devices [19] are examples of predicting encoding time, cost and stability significantly impacting the provisioning and scheduling of encoding tasks. Therefore, a highly desirable system that estimates the encoding time and costs and optimizes the encoding task schedule on selected spot instances [20].

To decrease the encoding costs and maximize the utilization of Amazon EC2 spot instances, we propose a new method called the Fast approach for better utilization of Amazon EC2 **Spot** Instances for video encoding (FSpot) based on four phases: 1) instance benchmarking, 2) fast encoding time estimation, 3) instance set selection and 4) priority and numerical calculation. The first phase tests different EC2 instances using various encoding parameters, extracts the critical features from the video encodings and creates a dataset, and proposes a heuristic for selecting EC2 spot instances. The second phase uses a fast estimate of the encoding speed for videos on a master node hosted on an on-demand EC2 instance that splits video into segments, estimates the encoding time and distributes encoding tasks to worker nodes hosted on spot instances. The third phase selects the required number of EC2 spot instances recommended for optimized video encoding in the Amazon cloud. Finally, the last phase calculates the priorities and number for EC2 spot instances, such that those with the lowest predicted video encoding cost have the highest priority. We evaluated FSpot on a set of ten heterogeneous videos of different genres with different duration and frame rates using three AWS availability zones. Experimental results show that, on average, our model can reduce the encoding costs by at least 15.8% and up to 47.8% when compared to a random selection of EC2 spot instances.

The significant contributions of the FSpot work are:

1. We benchmarked on eleven commonly used Amazon EC2 spot instances using different encoding parameters and video sequences.
2. We developed a novel method for fast encoding time estimation of video segments and proposed an algorithm combining Pareto frontier and clustering techniques to find an appropriate set of spot instances.
3. We also proposed and implemented a fast method to calculate the instance number and priority for different EC2 spot instances to optimize the Amazon EC2 spot instance selection for encoding task allocation. The proposed FSpot approach reduces the encoding costs by at least 15.8% and up to 47.8% compared to a random selection of EC2 spot instances.

This paper has five sections-Section 2 highlights related work. Section 3 describes the proposed FSpot approach and its implementation, followed by results evaluation in Section 4. Section 5 concludes the paper and highlights future work.

2 Related Work

2.1 General Scheduling Techniques

Gog et al. [21] studied various scheduling architectures and proposed a min-cost max-flow (MCMF) optimization over a graph and continuously reschedules the entire workload. Authors extend Quincy's [22] original MCMF algorithm that results in task placement latencies of minutes on a large cluster. In [23], the authors propose global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. Malawski et al. [24] presented a mathematical model to optimize the cost of scheduling workflows under a deadline constraint. It considers a multi-cloud environment where each provider offers a limited number of heterogeneous virtual machines and a global storage service to share intermediate data files.Ghobaei-Arani et al. [25] presented an autonomous resource provisioning framework to control and manage computational resources using a fuzzy logic auto-scaling algorithm in a cloud environment.

Similarly, Rodriguez et al. [26] described a plan-based offline auto-scaler that partitions workflows into bags-of-tasks and then applied a MIP-based approach to make the allocation plan. Another work of Malawski et al. [27] considered the problem of task planning on multiple clouds formulated but in the more general framework of the mixed-integer nonlinear programming problem (MINLP). Garcia-Carballeira et al. [28] combined randomized techniques with static local balancing in a round-robin manner for tasks scheduling. Chhabra et al. [29] combined multi-criteria meta-heuristics to schedule HPC tasks on the IaaS cloud. Ebadifard et al. [30] proposed a dynamic load balancing task scheduling algorithm for a cloud environment that minimizes the communication overhead. Wang et al. [31] performed an empirical analysis of amazon EC2 spot instance features affecting cost-effective resource management.

2.2 Video Transcoding-specific Scheduling Techniques

Some recent remarkable works contributed to scheduling the video transcoding tasks [32–34]. Kirubha et al. [35] implemented a modified controlled channel access scheduling method to improve the quality of service-based video streaming. Similarly, Jokhio et al. [36] presented a distributed video transcoding method to reduce video bitrates. Li et al. [37] presented a QoS-aware scheduling approach for mapping transcoding jobs to heterogeneous virtual machines. Recently, Sameti et al. [38] proposed a container-based transcoding method for interactive video streaming that automatically calculates the number of processing cores that maintain a specific frame rate for any given video segment and transcoding resolution. The authors performed benchmarking to find the optimal parallelism for interactive streaming video. Li et al. [39] proposed a HAS delivery scheme that combines caching, transcoding for energy and resource-efficient scheduling. Ma [40] proposed a scheduling method for transcoding MPEG-DASH video segments using a node that managed all other servers in the system (rather than predicting the transcoding times) and reported a saving time of up to 30%.

2.3 State-of-the-art Analysis

Previously listed general and transcoding-specific scheduling techniques are capable of processing a large amount of different computational workloads. Such systems use various scheduling algorithms ranging from general mixed-integer programming (MIP) techniques, flow-based formulations and workload-agnostic techniques to video-specific heuristics [41,42] that maximize the use of processing units and minimize the associated costs. Companies currently prefer on-demand and spot instances by utilizing state-of-the-art video codecs to enable cost-effective video encoding. As the cost of such computing units depends on the time of use (ph or ps), the customers strive to keep the highest possible utilization for all computing resources. They typically deploy the encoding tasks using opportunistic load balancing (OLB) algorithms to utilize the resources at all times. It is relatively easy to achieve maximum resource utilization if all the encoding tasks have similar complexity, require similar execution times on the underlying computing units and all computing units have the same price. However, a problem arises when a simple scheduling algorithm randomly assigns specific encoding tasks to expensive spot instances with a low availability probability or is not optimized for selected encoding parameters. This can lead to load imbalance, increased encoding time and costs and degraded video quality on the viewer side. The motivation for our work is to maximize the Amazon EC2 spot instances utilization for video encoding and provide the encoding infrastructure with advanced information on the various video encoding tasks to ensure their fast completion with reduced cost. The relatively straightforward case for the methods mentioned earlier is when all the encoding tasks have similar complexity, require similar execution times on the underlying computing units and all computing units have the same price. However, a problem arises when the scheduling algorithm misses specific knowledge about encoding workload and underlying computational resources behavior. Some methods are simply incapable of solving the problem directly in the case of the even bigger video workloads and smaller segment sizes of 2–4 s. Natural extension led to the flow-based formulations and workload-agnostic techniques that can work on significantly larger scales. However, it can quickly happen that those methods will assign segment encoding tasks to spot instances with a low availability probability or not optimized for selected encoding parameters. It will result in additional expenses and sub-optimal performance. This can also lead to load imbalance, increase encoding time and costs and degrade video quality on the viewer side. Further, some approaches consider only a single objective to optimize. Our multi-objective approach maximizes the Amazon EC2 spot instances utilization, reduces the related costs and increases the execution reliability for **large-scale video encoding workloads** by reinforcing decisions with **advanced information** on the various video encoding tasks obtained via the fast benchmark algorithm.

3 Proposed FSpot Approach

3.1 EC2 Instance Benchmarking

3.1.1 Dataset Selection

First, we selected ten video sequences of different visual complexity from the publicly available dataset [12]. Fig. 1 shows the SI and TI metrics of the selected videos. The average TI and SI metrics confirm the varying video content complexity. We used video sequences that represent a wide range of possible visual scenes and use cases. Tab. 1 presents video categories (or genres) and critical file characteristics of original videos. Using the FFmpeg [42] software v4.1.3, we uncompressed all video sequences into raw Y4M format and divided them into 80 video segments of 4 s duration each. Typically, each segment is a switching point to other video representations. Therefore the segment length becomes an important parameter in *HTTP Adaptive Streaming*. The 4 s segments are widely used in real video streaming deployments because they show a good trade-off between encoding efficiency and video streaming performance [43].

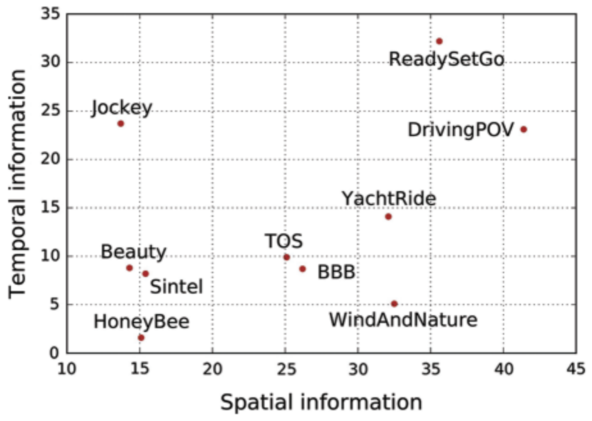

Figure 1: Average spatial information (SI) and temporal information (TI) for video sequences

Table 1: Original video file characteristics

Video description	Video category	Frames per second	Duration (in sec)
BBB	Animation	30	60
Beauty	Moving head	30	20
DrivingPOV	Moving cars	60	20
HoneyBee	Nature (flying bee)	30	20
Jockey	Sports (running jockey)	30	20
Sintel	Animation	24	60
TOS	Animation and real	24	60
WindAndNature	Rotating wind vanes	60	20
ReadySetGo	Sports (horse racing)	30	20
YachtRide	Moving yacht	30	20

3.1.2 EC2 Instance Performance Analysis

We encoded each Y4M segment using the FFmpeg ×264 video codec implementation with the veryslow encoding preset to get the highest possible quality compared to the original videos. The ×264 video codec contains nine encodings presets: ultrafast, superfast, veryfast, faster, fast, medium (default preset), slow, slower, veryslow, placebo [44]. Encoding bitrate with a slower ×264 encoding preset for the same video usually has a slower encoding speed but better visual quality [45]. We considered these generated video segments as source files and used them to encode different Amazon EC2 instances. We developed a framework using Python programming language to encode video sequences in the Amazon cloud automatically. Tab. 3 shows all encoded video segments on eleven different Amazon 2 × large instances (presented in Tab. 2) using various encoding parameters, i.e., bitrates and resolutions. All EC2 spot instances have eight vCPUs and RAM size ranges from 15 GiB for the c5a.2 × large instance to 64 GiB for the r5.2 × large and r5a.2 × large instances. We used multiple Amazon 2 × large instances commonly used for video transcoding [43]. We then extracted several features from the video encodings and created the Amazon EC2 instance encoding dataset. The raw dataset contains 16720 encoding tasks (80 segments ∗ 19 bitrates ∗ 11 EC2 instances) for the 4 s length video segments on medium encoding preset. Each record in our dataset contains EC2 instance name, EC2 instance availability, EC2 instance price, video segment name, encoding bitrate, file size, segment width, segment height, encoding time.

Table 2: Amazon instances

Amazon EC2 instance	vCPUs	RAM(GiB)	Processor	Optimized
c5a.2 × large	8	16	AMD	Compute
c5.2 × large	8	16	×86	Compute
c4.2 × large	8	15	×86	Compute
r5.2 × large	8	64	×86	Memory
m5.2 × large	8	32	×86	General
m5a.2 × large	8	32	AMD	General
r5a.2 × large	8	64	AMD	Memory
t3.2 × large	8	32	×86	General
t3a.2 × large	8	32	AMD	General
r4.2 × large	8	61	×86	Memory
m4.2 × large	8	32	×86	General

Table 3: Bitrate ladder (bitrate/resolution pairs). Bitrate values are in kbps

#	Bitrate	Resolution	#	Bitrate	Resolution
1	100	256 ∗ 144	11	4300	1920 ∗ 1080
2	200	320 ∗ 180	12	5800	1920 ∗ 1080
3	240	384 ∗ 216	13	6500	2560 ∗ 1440
4	375	384 ∗ 216	14	7000	2560 ∗ 1440
5	550	512 ∗ 288	15	7500	2560 ∗ 1440

(Continued)

Table 3: Continued

#	Bitrate	Resolution	#	Bitrate	Resolution
6	750	640 ∗ 360	16	8000	3840 ∗ 2160
7	1000	768 ∗ 432	17	12000	3840 ∗ 2160
8	1500	1024 ∗ 576	18	17000	3840 ∗ 2160
9	2300	1280 ∗ 720	19	20000	3840 ∗ 2160
10	3000	1280 ∗ 720			

3.1.3 EC2 Spot Instance Selection Heuristic

Let us assume we have over one hundred different spot instances to encode segments of a single video. Further, we only want to select the top N spot instances that will minimize the cost. Our work proposes a method that selects a set of computing units, for example, 5, for optimized video encoding. The main goal of this method is to reduce the number of computing units for further analysis quickly. We calculate the price ratio β_i and the speed factor r_i for each EC2 instance with respect to c5.2 × large base EC2 instance (see Tab. 5), as shown in Eqs. (1) and (2), respectively.

$$\beta_i = \frac{c_i}{c^{base}} \tag{1}$$

$$r_i = \frac{e^{base}}{e_i} \tag{2}$$

We then calculate the instance availability speed ratio H_i as shown in Eq. (3).

$$H_i = G_i + \alpha * (1 - p_i) \tag{3}$$

$$G_i = \frac{1}{r_i} \tag{4}$$

Eq. (3) reflects the adequate speed information of the EC2 spot instance i by analyzing its actual speed against the availability probability p_i. α is an adjusted weighting coefficient. We use the availability and pricing information of EC2 spot instances in our proposed FSpot model from the Amazon website [46,47]. Instead of the availability metric, Amazon uses the term *frequency of interruption*. For example, if the frequency of interruption is <5%, it means that the spot instance interruption of Amazon services based on historical information of the last three months before being terminated intentionally by a client is less than 5%. The G_i parameter in Eq. (3) is a relative time to encode a single video on EC2 spot instance i and is calculated by Eq. (4). The availability probability p_i of EC2 spot instance i is Amazon frequency of interruption between 0 to 1. Tab. 4 shows the availability probability calculation from the amazon frequency of interruption converted to percentage. Further, in our proposed work, we use H_{i} and β_{i} to select a set of computing units for optimized video encoding

3.2 Fast Encoding Time Estimation

We use a sample video file segment to calculate the encoding speed for the different Amazon EC2 instances. First, the system encodes a middle segment of a video sequence at the base node-the master node or the fastest available EC2 instance for a few seconds with selected encoding parameters. It then uses obtained encoding time data $t^*_{i,j}$ for the middle segment and instance availability speed ratio (H_i) to estimate the encoding speed $v^{encode}_{i,j}$ (in segments/sec) for different EC2 spot instances and video segments as shown in Eq. (5).

$$v^{encode}_{i,j} = \frac{1}{H_i * t^*_{i,j}} \tag{5}$$

Table 4: The amazon frequency of interruption converted to percentage

EC2 spot instance	Frequency of interruption	Availability probability p_i
c5.2 × large (base)	<5%	0.955
r5.2 × large	5%–10%	0.925
c5a.2 × large	15%–20%	0.825
r5a.2 × large	>20%	0.800

We assume that the encoding time of all segments of the same video sequence has similar values. Recent research [20] shows that the encoding times of segments of the same video file with the same encoding parameters have similar values and do not exceed one second for the ×264 video codec. Our approach uses a quick estimate of the encoding speed for each new video and a new set of encoding parameters.

From our dataset, we extracted ×264 codec encoding times for the base EC2 instance (c5.2 × large) for middle segments and all unique combinations of encoding parameters for each video sequence. We then used the instance availability speed ratio (H_i) to estimate the encoding speed for video segments on different EC2 spot instances. We only used the information about the encoding time of the middle segment on the base c5.2 × large EC2 instance. We can use our approach to make predictions for different video codecs, for example, for ×265. To do this, we need to automatically benchmark EC2 instances for the ×265 video codec and then use the results for the calculations. The output of this implementation phase is an array of estimated encoding speeds $v^{encode}_{i,j}$ for different video segments j and EC2 spot instances i.

3.3 EC2 Instance Selection Using Pareto Fronts and Clustering

We used our dataset to calculate H_i for all eleven EC2 spot instances and then calculated their β_i using the pricing information retrieved from Amazon [46,47]. Tab. 5 presents an example of different calculated parameters for all selected EC2 spot instances for the Amazon *Europe* (*Frankfurt*) region and *eu-central-1b* availability zone. Then, we applied Pareto fronts and a clustering approach to finding five EC2 spot instances for optimized video encoding in the cloud. The selected EC2 spot instances used to minimize the encoding costs are t3a.2 × large, t3.2 × large, c4.2 × large, c5a.2 × large, c5.2 × large. Please note that pricing information on the Amazon website changes in real-time, so in the entire encoding system, our proposed model will ask for new EC2 Spot prices every minute and recalculate the selected EC2 spot instance set.

For each EC2 spot instance, we calculate the (i) instance availability speed ratio H_i and (ii) price ratio β_i and use them as input parameters for our Pareto-fronts and clustering model. We calculate different Pareto fronts between H_i and β_i for all selected EC2 spot instances and rank each front in ascending order (see Fig. 2).

Table 5: Example of calculated parameters for different amazon EC2 spot instances

EC2 instance name	Encoding time e_i	Speed factor r_i	Relative time G_i	Frequency of interruption	Availability probability p_i	EC2 price c_i	EC2 price ratio β_i	EC2 availability speed ratio H_i
c5a.2 × large	4.3648	1.220	0.820	15%–20%	0.825	0.1345	0.990	0.821
c5.2 × large (base)	5.3431 (e^{base})	1.000	1.000	<5%	0.955	0.1358 (c^{base})	1.000	1.000
c4.2 × large	5.9900	0.892	1.121	<5%	0.955	0.1422	1.047	1.122
r5.2 × large	6.0300	0.886	1.129	5%–10%	0.925	0.1508	1.110	1.129
m5.2 × large	6.0400	0.885	1.130	<5%	0.955	0.1435	1.057	1.130
m5a.2 × large	6.3600	0.840	1.191	<5%	0.955	0.1902	1.401	1.191
r5a.2 × large	6.3700	0.839	1.192	>20%	0.800	0.1981	1.459	1.194
t3.2 × large	6.4400	0.829	1.206	<5%	0.955	0.1152	0.848	1.207
t3a.2 × large	6.7400	0.793	1.261	<5%	0.955	0.1037	0.764	1.261
r4.2 × large	7.0600	0.757	1.320	<5%	0.955	0.1523	1.122	1.321
m4.2 × large	7.0600	0.757	1.322	<5%	0.955	0.1596	1.175	1.322

We then apply K-means clustering on Pareto fronts points to form K clusters (see Fig. 2) such that the centroid of each cluster.

$$K = \min\left(x, quotient\left(\frac{n}{2}\right)\right) \tag{6}$$

where x is the number of Pareto fronts and n is the total number of EC2 spot instances. For example, if the number of EC2 spot instances is ten, clusters will be four. Our algorithm first selects EC2 spot instances belonging to the first Pareto front to find a set of optimized EC2 spot instances. Depending on the optimization problem (minimizing encoding time or cost of encoding), the algorithm selects points from the bottom or the top of the first Pareto front. If all EC2 spot instances of the same type in one Pareto front are already in use, the proposed algorithm selects other EC2 instances belonging to the same cluster and same front. If no EC2 spot instance from the same front and the same cluster is available, the proposed algorithm searches different EC2 instances within the same cluster but from another front. If all EC2 spot instances of one cluster are already in use, it requests the remaining EC2 spot instances from the first front, which belong to different cluster(s). If all EC2 spot instances of the first Pareto front are already in use, the algorithm will move to the second Pareto front and so on. We proposed Algorithm 1 to find a set of appropriate EC2 spot instances. This phase results in a set of preselected EC2 spot instances for optimized video encoding in the cloud.

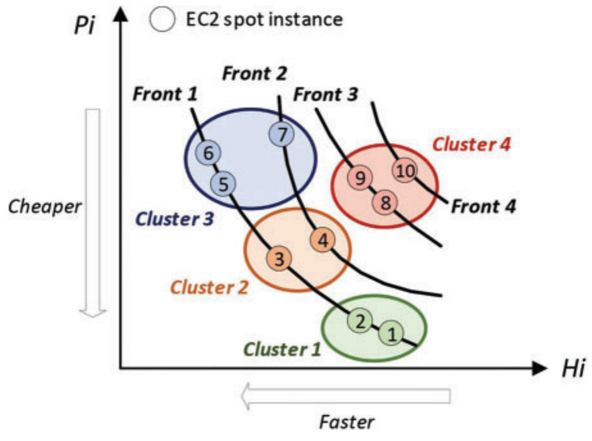

Figure 2: EC2 spot instance selection by using Pareto fronts and clusters

3.5 Calculating Priorities and Numbers for EC2 Spot Instances

This phase only uses EC2 spot instances that belong to the set selected by Algorithm 1. First of all, we represent the constraints. We consider the disk speed d^{copy} and the network speed k^{copy} from the master node to a cluster of EC2 spot instances as two main parameters influencing segments' distribution time. In actual encoding infrastructure, the open-source tool IPerf can be used to measure the network speed k^{copy}. The transmission speed of video segments w^{copy} is the minimum value between d^{copy} and k^{copy}, as shown in Eq. (7).

$$w^{copy} = \min(d^{copy},\ k^{copy}) \tag{7}$$

Algorithm 1: Algorithm for selecting EC2 spot instances using Pareto front and clustering techniques.

Input: array of EC2 spot instances (ec2[ ])
Input: number of Pareto fronts (n_fronts)
Input: number of EC2 to select (n)
Output: array of selected instances (s_ec2)

```
// Indexing of array ec2[ ] starts from one. ec2[ ] array is sorted by price ratio in ascending order.
1.  ec2[ ] ← {'val1', 'val2', 'val3', ... 'valP'}
2.  n_fronts[ ] ← val
3.  n ← val
4.
5.  Function select_ec2_set(ec2, n_fronts, n) :
6.      output[ ] ← null
7.      for i = 1 to n_fronts do
8.          current_cluster ← null
9.          for ec2 in ec2[ ] do
```

(Continued)

Algorithm 1: Continued

```
10.         if ec2 $\epsilon$ (i front) then
11.             output ← ec2
                remove ec2 from ec2[ ] array
                current cluster = cluster of ec2
                if len(output) == n then
12.                 Break
13.             End
14.             for all ec2i in (i+1) front do
15.                 if ec2i $\epsilon$ current cluster then
16.                     output ← ec2i
                        remove ec2i from ec2[ ] array
                        if len(output) == n then
17.                         Break
18.                     End
19.                 End
20.             End
21.         End
22.     End
23.     return output
24. End Function
25. s_ec2 ← select_ec2_set(ec2, n_fronts, n)
```

Eq. (8) calculates the minimum time to copy all segments of one video $T_i^{copy\ all}$ to multiple EC2 spot instances, where l is the number of segments for encoding and s is the average segment size. In turn, Eq. (9) calculates the minimum time to encode all segments of a video on z_i EC2 spot instances, where $v_{i,j}^{encode}$ is the estimated encoding speed (in segments/sec) of EC2 Spot instance type i.

$$T_i^{copy\ all} = \frac{l * s}{w^{copy}} \tag{8}$$

$$T_i^{encode\ all} = \frac{l}{v_{i,j}^{encode} * z_i} \tag{9}$$

For continuous encoding of video segments on EC2 spot instances of type i, the following constraint must be met:

$$T_i^{copy\ all} < T_i^{encode\ all} \tag{10}$$

Then the number z_i of EC2 spot instances to use can be represented as inequality 11. Also, z_i must be less than or equal to the number of l segments to encode and the maximum number z^{max} of EC2 spot instances that the system can request simultaneously (See 11). The value of z^{max} can be defined by the encoding infrastructure administrator or be a maximum number of EC2 instances of the specific type available in the cloud.

$$\begin{cases} z_i < \dfrac{w^{copy}}{v_{i,j}^{encode}.\ s} \\ z_i \leq l \\ z_i \leq z^{max} \end{cases} \tag{11}$$

By satisfying defined constraints for z_i, we calculate the maximum possible value of z_i for each EC2 spot insblence i. The next step is to prioritize EC2 spot instances. To do this, we calculate the predicted encoding time T_i^{pred} as shown in Eq. (12) for all segments l of a video file for different EC2 spot instances i. Next, the model uses the predicted encoding time to compute the predicted encoding cost V_i^{pred} of a video for each type i of EC2 instance, as shown in Eq. (13).

$$T_i^{pred} = \frac{l}{v_{i,j}^{encode}\ .\ z_i} \tag{12}$$

$$V_i^{pred} = T_i^{pred}.\ z_i.\ c_i \tag{13}$$

We sort the predicted encoding cost for all EC2 Spot instances in ascending order. The EC2 spot instance type with the lowest predicted video encoding cost has the highest priority and vise versa.

We calculated the priorities and optimized the number of EC2 spot instances of each type (i) required for encoding different video sequences (see Tab. 1). First, using defined constraints, the model finds the maximum possible number of EC2 spot instances z_i to use. Then the model uses the calculated number of EC2 spot instances z_i, EC2 spot instance price c_i and the predicted encoding time T_i^{pred} to calculate the predicted encoding cost V_i^{pred} of a video for different EC2 spot instances. We sorted the predicted encoding cost for all EC2 spot instances in ascending order in a manner that the EC2 spot instance i with the lowest predicted video encoding cost has the highest priority and vise versa. Tab. 6 shows the predicted video encoding cost for TOS video and different Amazon EC2 spot instances. We used 285 encoding tasks (*15 video segments* $*$ *19 bitrates*) and assumed that a video segment should be delivered to an EC2 spot instance again for each encoding operation. As we can see from Tab. 6, the proposed model recommends using eight *c5a.2* $\times$ *large* EC2 spot instances to minimize the encoding cost of the TOS video sequence, which results in a cost of \$ 0.04.

Table 6: Predicted video encoding costs for TOS video sequence *Eu-central-1b* availability zone

EC2 spot instance i	Number of EC2 spot instances (z_i)	Predicted encoding cost (V_i^{pred}) in \$
c5a.2 $\times$ large	8	0.040
t3a.2 $\times$ large	13	0.049
t3.2 $\times$ large	12	0.050
c5.2 $\times$ large (base)	10	0.052
c4.2 $\times$ large	12	0.053

4 Results and Analysis

This Section presents the proposed *FSpot* approach results to analyze the performance and examine its advantages for utilizing Amazon EC2 spot instances better. We compare the predicted encoding time and cost with the actual encoding time available in the dataset. We calculate the actual encoding time T_i^{actual} of a video for each EC2 spot instance using the same number of EC2 spot instances

z_i predicted by the model. We calculate the actual encoding cost using Eq. (14) and compare it with the predicted encoding cost V_i^{pred}.

$$V_i^{actual} = T_i^{actual} \, . \, z_i \, . \, c_i \tag{14}$$

Finally, we check how the predicted encoding times and costs are correlated with actual values. The predicted priorities for the EC2 spot instance have to be correct for the actual and predicted values.

Tab. 7 shows various parameters and their defined test values to evaluate our proposed FSpot model performance. We used the encoding times and prices for Amazon EC2 spot instances from our dataset. Tab. 8 shows the selected EC2 spot instances marked as '+' and their count calculated by the proposed FSpot model for Sintel video sequences and three availability zones *eu-central-(1a|1b|1c)* of AWS *Frankfurt* region. We see that the *1a* and *1b* zones have eleven, while the *1c* zone has only nine different Amazon EC2 spot instances. It occurs due to the dynamic availability of EC2 spot instances and dependency on the selected zone. The last column of Tab. 8 shows that the calculated numbers for the same EC2 spot instance and different availability zones have the same values. This is because the calculated numbers for EC2 spot instances primarily depend on the encoding speed and availability probability of EC2 spot instances, which remain unchanged for the same EC2 spot instance and Amazon region.

Table 7: Test input parameters for Sintel video sequence

Parameter name	Parameter value
Middle segment size (in MB), s	80
Maximum number of EC2 spot instances z^{max}	70
Disk speed of master node (in MB/sec), d^{copy}	180
Network speed (in MB/sec), k^{copy}	220
Number of segments to encode, l	15
Number of encoding bitrates, h	19
Number of transcoding tasks, $r = (h * l)$	285

Table 8: Predicted numbers for EC2 spot instances. Sintel video sequence

EC2 spot instance i	Zone 1a	Zone 1b	Zone 1c	Number of instances
c5a	+	+	+	9
t3a	+	+	+	14
t3	+	+	n/a	13
c5	+	+	+	11
c4	+	+	+	12
m5	-	-	-	12
r5	-	-	-	12
r4	-	-	+	14
m4	-	-	n/a	14
m5a	-	-	-	13
r5a	-	-	-	13

Fig. 3 depicts the estimated numbers of different EC2 spot instances located in *the eu-central-1b* availability zone. It clearly shows that the number of EC2 spot instances for the Sintel video sequence varies from nine to fourteen. The proposed FSpot model calculates a minimum of nine EC2 spot instances for c5a.2 × large type and a maximum of fourteen EC2 spot instances for t3a.2 × large, m4.2 × large and r4.2 × large instance types. Fig. 4 shows the predicted and actual encoding time for Sintel video on different EC2 spot instances of *the eu-central-1b* availability zone. The predicted encoding time for all EC2 spot instances is slightly higher than the actual encoding time extracted from the dataset. There is a slight difference of less than 4% between the predicted and actual encoding times. It occurred because we used only one middle segment encoding information of the video sequence and replicated it to the rest of the segments to estimate the encoding time.

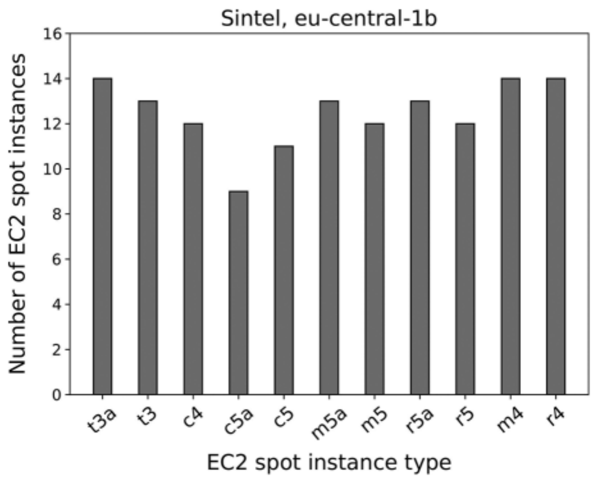

Figure 3: The calculated number of EC2 spot instances for the sintel video sequence

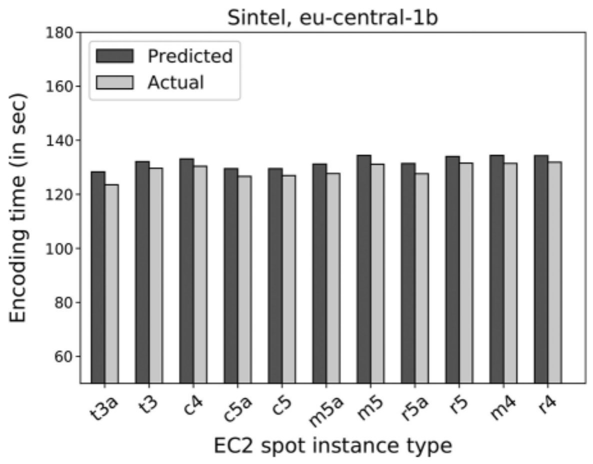

Figure 4: Predicted and actual encoding time for different EC2 spot instances for the sintel video sequence

Tab. 9 presents the predicted and actual encoding time results for three video sequences (BBB, Sintel, TOS) on five different EC2 spot instances. We can see that for Sintel and TOS videos, the difference between the average predicted and actual values for all five EC2 spot instances is relatively small, 3 and 4 s, respectively. However, for the BBB video sequence, the difference reaches 24 s. This is because the actual encoding time of the middle segment of the BBB video sequence has a significant difference from the average encoding time of all video segments. Tab. 10 shows the average actual encoding times for all segments of three video sequences compared to the average encoding times of middle segments of the videos. Tab. 10 presents the results for the *c5.2* $\times$ *large* EC2 spot instance and *eu-central-1b* AWS availability zone. We see that the BBB video sequence has the highest difference of 0.64 s (3.94–3.30) between the average actual encoding time for all segments and the middle segment. The difference for Sintel and TOS videos is only 0.09 and 0.17 s, respectively.

Table 9: Predicted and actual encoding time (in sec) for different video sequences and eu-central-1b availability zone

EC2 instance	BBB		Sintel		TOS	
	Pred. time	Act. time	Pred. time	Act. time	Pred. time	Act. time
t3a	132	156	128	124	132	131
t3	142	166	132	130	137	130
c5a	129	151	130	127	139	134
c5	135	161	130	127	136	131
c4	132	158	133	130	127	122
Average	**134**	**158**	**131**	**128**	**134**	**130**

Table 10: Average encoding times for segments of three video sequences

Video sequence	Average actual encoding time (sec)	
	All segments	Middle segment
BBB	3.94	3.30
Sintel	4.89	4.98
TOS	4.59	4.76

Fig. 5 shows the predicted and actual encoding costs for the Sintel video sequence on the *eu-central-1b* availability zone. We see that the predicted encoding times for all EC2 spot instances are slightly higher than the actual encoding times. This is because the predicted encoding times for the EC2 spot instances are slightly higher than the actual encoding times (see Fig. 4). Our proposed FSpot model selects different EC2 spot instances by prioritizing the low cost. Tab. 11 shows that the *c5a.2* $\times$ *large* spot instance has the highest priority. Both the predicted and actual encoding costs for *c5a.2* $\times$ *large* are the lowest compared to other EC2 spot instances. This means that the proposed FSpot model can select the appropriate EC2 spot instance type and the number of EC2 spot instances with minimum video encoding costs.

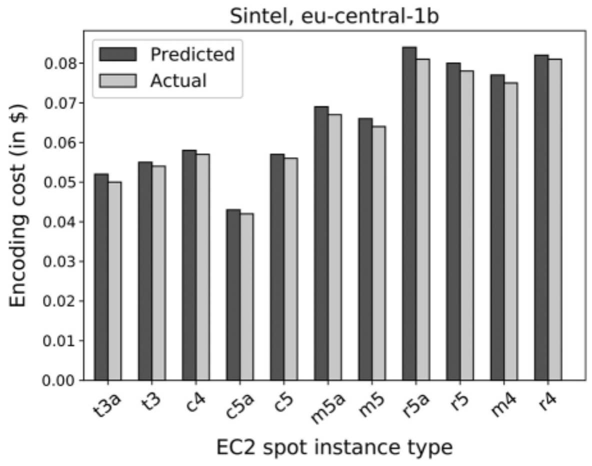

Figure 5: Predicted and actual encoding cost (in $) for different EC2 spot instances for sintel video sequence and *eu-central-1b* availability zone

Table 11: Predicted and actual encoding cost for the Sintel video sequence and eu-central-1b availability zone

EC2 spot instance *i*	Selected for zone 1*b*	Predicted cost, $	Actual cost, $	Priority of instance
c5a	Yes	0.043	0.042	1
t3a	Yes	0.052	0.050	2
t3	Yes	0.055	0.054	3
c5	Yes	0.057	0.056	4
c4	Yes	0.058	0.057	5
m5	No	0.066	0.064	6
m5a	No	0.069	0.067	7
m4	No	0.077	0.075	8
r5	No	0.080	0.078	9
r4	No	0.082	0.081	10
r5a	No	0.084	0.081	11

Additionally, Tab. 11 shows all predicted priorities for all EC2 spot instances in ascending order in the last column table. Interestingly, all predicted and actual costs are mapped as per their priority and arranged in ascending order. This shows that the model assigned the correct priorities to all EC2 spot instances. Additionally, the first five EC2 spot instances (from *c5a* to *c4*) belong to a set selected by our FSpot approach. Thus, our FSpot approach outperforms in quickly reduce the number of EC2 spot instances for further and in-depth analysis.

We compared our proposed FSpot approach to a random method where the system randomly selects *2* × *large* EC2 spot instances to encode video segments. With the proposed FSpot approach, the *percentage decrease of cost* (PDC) for Sintel video sequence ranges from 16% for *t3a.2* × *large* spot instances to 48% for *r4.2* × *large* and *r5a.2* × *large* spot instances. Fig. 6 presents the PDC values for ten EC2 spot instances compared to *c5a.2 × large* spot instances. We also compared our FSpot approach with another approach where the lowest price EC2 spot instance has the highest priority. According to Tab. 5, the EC2 spot instance *t3a.2 × large* has the lowest price of 0.1037 $. The proposed FSpot model selects c5a.2 × large spot instance type and achieves PDC to 16% with the highest priority compared to the lowest price EC2 spot instance (*t3a.2 × large*). This means that the model can choose the appropriate EC2 spot instance, even with a higher price. The higher price EC2 spot instances typically have higher video encoding speed and vice versa. Tab. 12 shows PDC for all ten video sequences compared to the random approach. We can see that the *ReadySetGo* video sequence has the lowest PDC of 11.8%, while *the Beauty* video sequence has the highest PDC of 20.8%. The results show that, on average, our approach can reduce the encoding cost by at least 15.8% and the maximum by 47.8% (see the last row in Tab. 12). Ideally, the PDC value will be zero if the random approach selects the best EC2 spot instance and the correct number of EC2 instances. However, the chances of choosing both values correctly are meager. Our proposed FSpot model can select the best EC2 spot instances between different AWS availability zones.

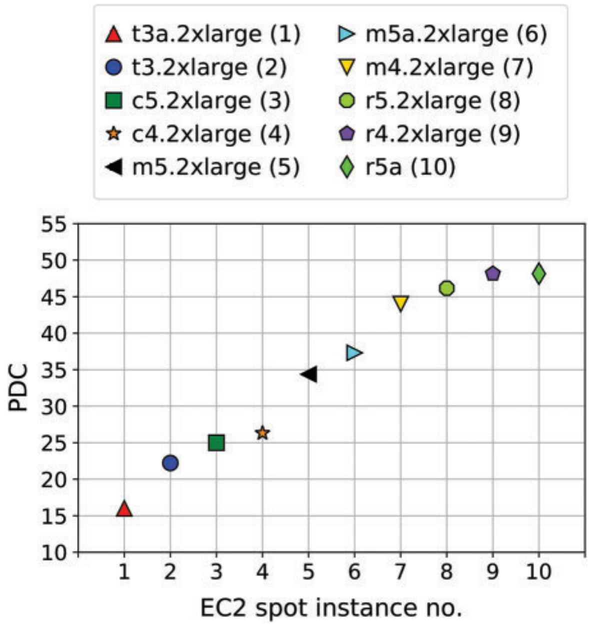

Figure 6: PDC for ten EC2 spot instances compared to *c5a.2 × large* spot instance

We proposed the FSpot method by combining the Pareto front with clustering techniques to optimize the AWS EC2 spot instance selection for encoding tasks allocation to minimize the encoding costs. Our model, on average, can reduce encoding costs by at least 15.8% and up to 47.8% compared to the random approach. FSpot can be customized and applied to the Google Cloud, and Microsoft Azure platforms with their own spare compute capacity instances. Deploying our model in an existing encoding infrastructure requires the development of an application programming interface. The encoding infrastructure will interact via the API with the model to calculate the predictions for upcoming encodings.

Table 12: Percentage decrease of cost (PDC) for all ten video sequences compared to the random approach. Eu-central-1b Amazon availability zone

Video sequence		Ready SetGo	Driving POV	Wind And Nature	BBB	Jockey	Yacht Ride	Sintel	Honey Bee	TOS	Beauty	Avg value
PDC in %	Min	11.8	13.3	14.8	15	15	15.8	16	16.7	18.4	20.8	**15.8**
	Max	46.4	48	47.7	47.7	48.5	48.4	48.1	48.3	47.4	47.2	**47.8**

5 Conclusion and Future Work

In this research, we performed benchmarking on Amazon EC2 instances using different encoding parameters and video sequences. We used video sequences and segments of different genres and visual complexity. We proposed a novel FSpot approach for fast estimation of video segments encoding time at the master node and selecting the appropriate set of EC2 spot instances for video encoding. We developed an algorithm by combining Pareto front and clustering techniques to find a set of appropriate EC2 spot instances for video encoding. Our approach calculates the EC2 spot instance count and priorities for optimized video encoding in the cloud. We implemented and tested our FSpot approach to optimize the Amazon EC2 spot instance selection for encoding tasks allocation. Results show that the FSpot approach optimizes Amazon EC2 spot instances utilization and minimizes the video encoding costs in the cloud. On average, FSpot can reduce the encoding costs ranging from 15.8% to 47.8% compared to a random selection of EC2 spot instances.

We plan in the future to extend our method for predicting the encoding time using multiple video codecs on different cloud computing instances and infrastructures. We will test our model on ARM and GPU processing instances in the cloud. In addition, we plan to develop an intelligent scheduler and auto-tuner to automate the process of optimized video encoding in the cloud.

Funding Statement: This work has been supported in part by the Austrian Research Promotion Agency (FFG) under the APOLLO and Karnten Fog project.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Robbins, *Cisco Annual Internet Report (2018–2023). Cisco*, 170 West Tasman Dr. San Jose, CA 95134 USA, Tech. Rep., 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] A. Javadtalab, M. Semsarzadeh, A. Khanchi, S. Shirmohammadi and A. Yassine, “Continuous one-way detection of available bandwidth changes for video streaming over best effort networks,” *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 1, pp. 190–203, 2015.

[3] I. Sodagar, “The MPEG-DASH standard for multimedia streaming over the internet,” *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[4] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer and R. Zimmermann, “A survey on bitrate adaptation schemes for streaming media over HTTP,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 562–585, 2019.

[5] A. Zabrovskiy, E. Petrov, E. Kuzmin and C. Timmerer, “Evaluation of the performance of adaptive HTTP streaming systems,” *CoRR*, vol. abs/1710.02459, pp. 1–7, 2017. [Online]. Available: http://arxiv.org/abs/1710.02459.

[6] A. Zabrovskiy, E. Kuzmin, E. Petrov, C. Timmerer and C. Mueller, "AdViSE: Adaptive video streaming evaluation framework for the automated testing of media players," in *8th ACM on Multimedia Systems Conf. (MMSys'17)*, New York, NY, USA: Association for Computing Machinery, pp. 217–220, 2017.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard 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, 2003.

[8] G. J. Sullivan, J. Ohm, W. 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, 2012.

[9] D. Mukherjee, J. Han, J. Bankoski, R. Bultje, A. Grange *et al.,* "A technical overview of VP9: The latest open-source video codec," *Annual Technical Conf. Exhibition (SMPTE'2013)*, Hollywood, CA, USA, pp. 1–17, 2013.

[10] C. Chen, Y. Lin, S. Benting and A. Kokaram, "Optimized transcoding for large scale adaptive streaming using playback statistics," in *25th IEEE Int. Conf. on Image Processing (ICIP)*, Athens, Greece, pp. 3269–3273, 2018.

[11] B. Bross, J. Chen and S. Liu, "Versatile video coding (Draft 7)," *JVET-P2001*, Geneva, CH, 2019.

[12] A. Zabrovskiy, C. Feldmann and C. Timmerer, "Multi-codec DASH dataset," in *9th ACM Multimedia Systems Conf. (MMSys '18)*, New York, NY, USA: ACM, pp. 438–443, 2018.

[13] D. Vatolin, D. Kulikov, E. Sklyarov, S. Zvezdakov and A. Antsiferova, "Video transcoding clouds comparison 2019," *Moscow State University, Technical Report*, 2019.

[14] M. G. Koziri, P. K. Papadopoulos, N. Tziritas, T. Loukopoulos, S. U. Khan *et al.,* "Efficient cloud provisioning for video transcoding: Review, open challenges and future opportunities," *IEEE Internet Computing*, vol. 22, no. 5, pp. 46–55, Sep. 2018.

[15] T. Pham, S. Ristov and T. Fahringer, "Performance and behavior characterization of Amazon EC2 spot instances," in *11th IEEE Int. Conf. on Cloud Computing (CLOUD 2018)*, San Francisco, CA, USA, pp. 73–81, 2018.

[16] X. Li, M. A. Salehi, Y. Joshi, M. K. Darwich, B. Landreneau *et al.,* "Performance analysis and modeling of video transcoding using heterogeneous cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 910–922, 2019.

[17] R. Matha, D. Kimovski, A. Zabrovskiy, C. Timmerer and R. Prodan, "Where to encode: A performance analysis of x86 and arm-based Amazon EC2 instances," *17th International Conference on eScience (eScience)*, Innsbruck, Austria, IEEEXplore, pp. 118–127, 2021. 10.1109/eScience51609.2021.00022.

[18] S. Sameti, M. Wang and D. Krishnamurthy, "Stride: Distributed video transcoding in spark," in *37th Int. Performance Computing and Communications Conf. (IPCCC)*, Orlando, FL, USA: IEEE, pp. 1–8, 2018.

[19] O. Barais, J. Bourcier, Y. Bromberg and C. Dion, "Towards microservices architecture to transcode videos in the large at low costs," in *Int. Conf. on Telecommunications and Multimedia (TEMU)*, Heraklion, Greece, pp. 1–6, 2016.

[20] P. Agrawal, A. Zabrovskiy, A. Ilangovan, C. Timmerer and R. Prodan, "FastTTPS: Fast approach for video transcoding time prediction and scheduling for HTTP adaptive streaming videos," *Cluster Computing*, vol. 24, pp. 1605–1621, 2021.

[21] I. Gog, M. Schwarzkopf, A. Gleave, R. N. Watson and S. Hand, "Firmament: Fast, centralized cluster scheduling at scale," in *12th USENIX Symp. on Operating Systems Design and Implementation (OSDI-16)*, Savannah, GA, USA, pp. 99–115, 2016.

[22] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar *et al.,* "Quincy: Fair scheduling for distributed computing clusters," in *22nd Symp. on Operating Systems Principles*, Big Sky, Montana, USA: ACM SIGOPS, pp. 261–276, 2009.

[23] A. Tumanov, T. Zhu, J. W. Park, M. A. Kozuch, M. HarcholBalter *et al.,* "Tetrisched: Global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters," in *11th European Conf. on Computer Systems*, London, United Kingdom, pp. 1–16, 2016.

[24] M. Malawski, G. Juve, E. Deelman and J. Nabrzyski, “Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds,” *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.

[25] M. Ghobaei-Arani, A. A. Rahmanian, A. Souri and A. M. Rahmani, “A Moth-flame optimization algorithm for web service composition in cloud computing: Simulation and verification,” *Software: Practice and Experience*, vol. 48, no. 10, pp. 1865–1892, 2018.

[26] M. A. Rodriguez and R. Buyya, “Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods,” *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 12, no. 2, pp. 1–22, 2017.

[27] M. Malawski, K. Figiela and J. Nabrzyski, “Cost minimization for computational applications on hybrid cloud infrastructures,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1786–1794, 2013.

[28] F. Garcia-Carballeira, A. Calderon and J. Carretero, “Enhancing the power of two choices load balancing algorithm using round-robin policy,” *Cluster Computing*, vol. 24, pp. 1–14, 2020.

[29] A. Chhabra, G. Singh and K. S. Kahlon, “Multi-criteria HPC task scheduling on IaaS cloud infrastructures using metaheuristics,” *Cluster Computing*, vol. 24, pp. 1–34, 2020.

[30] F. Ebadifard and S. M. Babamir, “Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment,” *Cluster Computing*, vol. 24, pp. 1–27, 2020.

[31] C. Wang, Q. Liang and B. Urgaonkar, “An empirical analysis of Amazon EC2 spot instance features affecting cost-effective resource procurement,” *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, no. 2, pp. 1–24, 2018.

[32] R. Aparicio-Pardo, K. Pires, A. Blanc and G. Simon, “Transcoding live adaptive video streams at a massive scale in the cloud,” in *6th ACM Multimedia Systems Conf.*, Portland, Oregon, pp. 49–60, 2015.

[33] P. Oikonomou, M. G. Koziri, N. Tziritas, A. N. Dadaliaris, T. Loukopoulos, G. I. Stamoulis *et al.*, “Scheduling video transcoding jobs in the cloud,” in *Int. Conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax, NS, Canada: IEEE, pp. 442–449, 2018.

[34] L. Wei, J. Cai, C. H. Foh and B. He, “QoS-Aware resource allocation for video transcoding in clouds,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 49–61, 2016.

[35] D. Kirubha and K. Ramar, “MCCA scheduling for enhancing QoS based video streaming for video surveillance applications,” *Cluster Computing*, vol. 22, no. 6, pp. 13945–13955, 2019.

[36] F. Jokhio, T. Deneke, S. Lafond and J. Lilius, “Bitrate reduction video transcoding with distributed computing,” in *20th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing*, Munich, Germany: IEEE, pp. 206–212, 2012.

[37] X. Li, M. A. Salehi, M. Bayoumi, N. -F. Tzeng and R. Buyya, “Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 556–571, 2017.

[38] S. Sameti, M. Wang and D. Krishnamurthy, “Contrast: Container-based transcoding for interactive video streaming,” in *Network Operations and Management Symp. (NOMS’2020)*, Budapest, Hungary: IEEE, pp. 1–9, 2020.

[39] L. Li, D. Shi, R. Hou, R. Chen, B. Lin *et al.*, “Energy-efficient proactive caching for adaptive video streaming via data-driven optimization,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5549–5561, 2020.

[40] L. Ma, “An efficient scheduling multimedia transcoding method for DASH streaming in cloud environment,” *Cluster Computing*, vol. 22, pp. 1043–1053, 2017.

[41] “Use Spot Instance Pricing for Your Video Encoding Workflows with Bitmovin,” https://aws.amazon.com/blogs/startups/use-spot-instance-pricing-for-your-video-encoding-workflow/-with-bitmovin-containerized-encoding/. [Online; accessed 14-Jan-2021], 2017.

[42] “FFprobe Documentation,” https://ffmpeg.org/ffprobe.html. [Online; accessed 25-July-2020], 2020.

[43] S. Lederer, C. Muller and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *3rd Multimedia Systems Conf. (MMSys'12)*, New York, NY, USA: ACM, pp. 89–94, 2012.

[44] A. Zabrovskiy, P. Agrawal, R. Matha, C. Timmerer and R. Prodan, "ComplexCTTP: Complexity class-based transcoding time prediction for video sequences using artificial neural network," in *Sixth Int. Conf. on Multimedia Big Data (BigMM)*, New Delhi, India, pp. 316–325, 2020.

[45] "H.264 Video Encoding Guide," https://trac.ffmpeg.org/wiki/Encode/H.264. [Online; accessed 10-September-2020], 2020.

[46] Amazon, "Amazon EC2 spot instances pricing," https://aws.amazon.com/ec2/spot/pricing. [Online; accessed 14-Jan-2021], 2021.

[47] "Spot Instance Advisor," https://aws.amazon.com/ru/ec2/spot/instance-advisor. [Online; accessed 14-Jan-2021], 2021.