applied sciences

Article

A Cloud-Based Distributed Architecture to Accelerate Video Encoders

Juan Gutiérrez-Aguado *, Raúl Peña-Ortiz, Miguel Garcia-Pineda and Jose M. Claver

Departament d'Informàtica, Universitat de València, Avd. Universitat, SN, 46100 Burjassot, Spain; raul.penya@uv.es (R.P.-O.); miguel.garcia-pineda@uv.es (M.G.-P.); jose.claver@uv.es (J.M.C.)

* Correspondence: juan.gutierrez@uv.es

Received: 30 May 2020; Accepted: 20 July 2020; Published: 23 July 2020

Featured Application: The proposed cloud-based video encoding architecture could provide a new encoding platform decreasing the encoding time while keeping the video quality. This system could be used by any company dedicated to multimedia broadcasting, which is booming these days.

Abstract: Nowadays, video coding and transcoding have a great interest and important impact in areas such as high-definition video and entertainment, healthcare and elderly care, high-resolution video surveillance, self-driving cars, or e-learning. This growing demand for high-resolution video boosts the proposal of new codecs and the development of their encoders that require high computational requirements. Therefore, new strategies are needed to accelerate them. Cloud infrastructures offer interesting features for video coding, such as on-demand resource allocation, multitenancy, elasticity, and resiliency. This paper proposes a cloud-based distributed architecture, where the network and the storage layers have been tuned, to accelerate video encoders over an elastic number of worker encoder nodes. Moreover, an application is developed and executed in the proposed architecture to allow the creation of encoding jobs, their dynamic assignment, their execution in the worker encoder nodes, and the reprogramming of the failed ones. To validate the proposed architecture, the parallel execution of existing video encoders, `x265` for H.265/HEVC and `libvpx-vp9` for VP9, has been evaluated in terms of scalability, workload, and job distribution, varying the number of encoder nodes. The quality of the encoded videos has been analyzed for different bit rates and number of frames per job using the Peak Signal-to-Noise Ratio (PSNR). Results show that our proposal maintains video quality compared with the sequential encoding while improving encoding time, which can decrease near 90%, depending on the codec and the number of encoder nodes.

Keywords: cloud computing; video coding; hyperconvergence; video quality; jobs distribution

1. Introduction

Several high profile events, which are related to video traffic, occurred during the first half of 2019, namely the final season of Game of Thrones, the FIFA Women's World Cup, releases of new TV series and movies or new streaming services. Furthermore, the data from last year is speaking quite clearly on the trends that define 2019 a great year for the Internet video traffic, which reached over 60% of all IP downstream traffic even though 4K content streams are a drop in the bucket so far and 8K is not yet a reality [1]. Moreover, due to the COVID-19 worldwide pandemic, which has drastically affected network usage around the world, the video streaming traffic in early 2020 is 2.2% higher than in 2019 [2].

According to big companies such as Cisco Systems, Google, Apple and Microsoft, video traffic volume will account for 82% of traffic by 2022 [3]. Different types of IP video will contribute significantly to increasing this type of traffic as well as increasing processing and transmission requirements to unprecedented limits. Examples are: Internet video, IP video on demand, video files exchanged through file sharing, video-streamed gaming, video conferencing, high-definition video and entertainment applications, new multimedia-based applications in the areas of healthcare and elderly care, high-resolution video surveillance, self-driving car and e-learning.

Users consume video content by using different devices (i.e., smart TVs and phones, tablets, laptops or desktop computers) that are connected to the Internet through different network technologies (e.g., broadband, Wi-Fi, LTE, 3G, 4G, and 5G). Dynamic Adaptive Streaming over HTTP has been adopted by video content distributors to deal with this hardware heterogeneity and network conditions variability due to its adaptability, flexibility and ease to perform several encoding profiles as quickly as possible in order to provide an early device-adapted video streaming, only requiring an HTTP server [4]. However, the content preparation is characterized by high computation and temporal load because it may require using different codecs with several resolutions and bit rates. For example, in YouTube a 4 K video can be distributed in eight resolutions. Each resolution is encoded using H.264/AVC [5] and VP9 [6] codecs with several bit rates, obtaining for a video 32 different encoded videos.

Mainly due to the requirements of video resolutions demanded by new devices and society, several video codecs have been proposed. The two primary contestants in the Ultra-High Definition (UHD) space are H.265/HEVC [7] and VP9 [6]. H.265 was proposed with the aim of doubling data compression and halving bit-rate as compared to H.264/AVC standard while preserving its equivalent video quality [8], but H.265 contains patented technologies. Their developed encoders are capable of compressing high-definition and ultra-high-definition video, but they present a high bit rate and long encoding time due to its high computational complexity. For this reason, these encoders need to be accelerated, as the previous H.264/AVC encoder, by using hardware solutions as GPU accelerators [9] and multi-core platforms [10]. If more acceleration is needed, then distributed solutions where the load is distributed among nodes are required.

The National Institute of Standards and Technology defines cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resilient computing resources (e.g., networks, servers, storage, applications and services) that can be elastically (scale rapidly) provisioned and released with minimal management effort or service provider interaction to serve multiple consumers using a multi-tenant model [11]. According to this definition, a cloud-based infrastructure can be a fundamental piece for storing and encode/transcode multimedia content with a certain level of reliability and capacity [12]. For instance, Netflix uses Amazon Web Services for nearly all its computing and storage needs, including video transcoding [13].

The main contribution of this paper is the design, implementation, and validation of a cloud-based distributed architecture to accelerate video encoders. It also provides the specification of the required nodes, the tune of the network and the storage, the specification of the jobs, the development of a distributed application based on a message broker to dynamically perform encoding jobs, and the validation of the proposal in a private cloud. In the proposed solution, a video is split in chunks of fixed size, for each chunk to be encoded an encoding job is generated, and each job is consumed by a worker virtual machine (VM) node that performs the encoding of the chunk.

Relevant features of the proposed solution are: elasticity (horizontal scalability), hyperconvergence (software defined storage, computation, and networks that are deployed on top of off-the-shelf servers), adaptability (can work with different codecs), fault-tolerance (if a node fails, then another node will perform the job), and automation (when a node is started, no intervention is required to deploy, configure, and start the application). Architecture and application have been tested with H.265 and VP9 encoders with different chunks sizes (full video, 25, 50, 100, and 200 frames), and bit rates (0.75 Mb, 1.5 Mb, 3 Mb, and 4.5 Mb).

The rest of the paper is structured as follows. Section 2 reviews the state of the art in this field. Section 3 explains both cloud-based architecture and application. Section 4 describes the experimental setup used to deploy the architecture and analyses its scalability and resource consumption, by varying the number of virtual machines to encode videos, and quality metrics for different video chunk sizes, bit rates and codecs. A qualitative comparison of the previous work with our proposal is discussed in Section 5. Finally, some concluding remarks and future work are drawn in Section 6.

2. Related Work

In order to perform a correct analysis of the existing work related to the topic of the paper we must emphasize the difference between encoding and transcoding videos. On the one hand, the process of encoding involves taking raw, original and uncompressed source content and converting it to a compressed digital format. On the other hand, the process of transcoding involves taking an encoded video file in one of the digital formats and converting it to another digital format, size or bitrate. Our proposal is focused on the encoding, activity that requires a greater computing capacity due to the weight of RAW files, especially if we work with HD resolutions (720p, 1080p) or higher.

Taking into account the previous point and after an exhaustive search we can indicate that there are few published papers related to RAW video encoding over cloud environments, and they are mainly based on the MapReduce paradigm [14,15]. In [14], the authors propose the Split&Merge architecture for deploying video encoding over cloud infrastructures. The proposal of this work is based on splitting the video into several chunks and distributing these chunks in the cloud infrastructure to decrease the coding time. They select the GOP size as an input to determinate the size of the chunks (it is fixed on 250 frames). This paper does not provide quantitative measurements of the communication overheads that the cloud execution entails and also completely bypass the effects on output quality. The authors of [15] improve the performance of video coding with content-aware video segmentation and scheduling over cloud. Their segmentation is based on several chunks with the same size, but it considers the video content in order to schedule encoding them. They determine that each chunk may contain a type of video content and therefore each chunk requires an encoding time. The chunks with long encoding times are arranged first, and chunks with short encoding times are gradually arranged, reducing the idle time. As with the previous paper [14], the authors only focus on the encoding time without taking into account the processing time of each chunk to select the type of video content and communication time are not valued either, aspect to bear in mind when we use UHD videos due to their size.

The applicability of the Function as a Service programming model to the problem of video coding has been analyzed in [16]. This work adapts the VP8 algorithm to encode multiple chunks that are further split, for instance chunks of 96 frames are divided in chunks of 6 frames executed in parallel using 16 workers. Each worker is a function executed in AWS lambda and the 16 workers encoding a chunk must be coordinated to produce only a keyframe per chunk. The proposal uses a coordinator VM that starts jobs in workers and a VM that provide rendezvous for the communication among workers encoding the same chunk. The main problem of this approach is that it requires to modify the encoder, thus new and better encoders such as VP9 or AV1 must be adapted to run in the proposed model. Besides, while lambda functions can scale with no prior allocation of resources, they have some limitations. The most important ones are the limited execution time and storage capabilities. Therefore, in scenarios where the function must perform long-running tasks (for example, if they must produce multiple bitrate encodings to reduce data transfer or because the encoder is complex) or the chunk size is increased this programming model is not appropriate. Finally, this model cannot be easily replied in private clouds because of its reliance on provider services.

If we pay attention to video transcoding on cloud infrastructures, there are few works in the literature that are directly related to the work proposed in this paper. Previous proposed architectures for distributed video encoding/transcoding in the cloud has been proposed for H.264 [17–21]. These approaches rely on MapReduce or on cluster of machines to perform the encoding, therefore the main difference with the proposed work is that the number of encoders cannot be modified dynamically once the encoding is started.

3. Proposed Architecture

This section introduces the proposed architecture, which is an elastic cloud-based distributed approach to accelerate video encoding, and the application that is executed on the top to evaluate the whole solution. Firstly, the logical view of the architecture is presented in Section 3.1 by describing the adopted virtual infrastructure based on the open source private cloud OpenStack technology (https://www.openstack.org): the type of VMs, networks, storage, and network tune to improve performance. Finally, details of the application are given in Section 3.2.

3.1. Virtual Infrastructure

The logical view of the architecture is shown in Figure 1. Two networks are defined: a self service job network used to send and receive messages that encapsulate video encode jobs and a provider media network used to transfer video data.

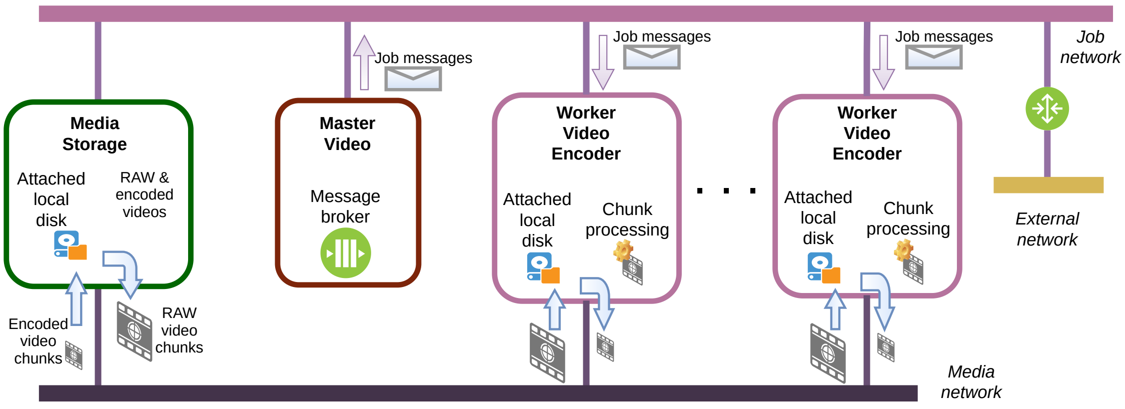

Figure 1. Logical view of the proposed virtual infrastructure.

Besides, three types of VMs have been defined: a *Master Video* (master), *Worker Video Encoders* (workers), and a *Media Storage* (media). The roles of these VMs are the following:

- Master VM has installed a message broker, RabbitMQ (https://www.rabbitmq.com/) in our deployment, and a Java Runtime Environment (JRE) (https://www.oracle.com/java). This node hosts a Java application that allows the user to specify: the video to be encoded, the encoder, the location of the video in media, the bit rates, the chunk size, or the total number of frames, among others. The number of chunks is calculated, and for each chunk a message with all the information to encode that chunk (chunk number, bitrates, location of the video, where to put the encoded videos, the encoder, etc.) is sent to the message broker.
- Each worker VM has installed a JRE and `ffmpeg` [22] with support for H.265/HEVC using the `x265` encoder [23] and for VP9 using the `libvpx-vp9` encoder [24]. These nodes run a Java application that subscribes to the queue where video job messages are stored. When the application in a worker node receives a message to encode a particular chunk, it downloads the video chunk from the media and starts an encoder process using the configuration specified in the message. Finally, the encoded chunk is uploaded to the media.
- Media VM contains the videos to be processed, provides chunks of these videos, and stores the encoded chunks.

With respect to the networking, the *Media Network* uses Single Root I/O Virtualization (SR-IOV) in order to achieve near native performance in the transference of video chunks. Before starting a new VM connected to the *Media Network*, a new OpenStack neutron port is created in this network. This port creates a new Virtual Function in a SR-IOV capable network interface at the compute node. This port is assigned at boot time to the VM. Each worker is connected to two networks: the *Media Network*, to access the media VM, and the *Job Network* to connect to the master VM. The media VM is connected to the *Media Network* and to the *Job Network*. The master VM only requires a connection to the *Job Network*.

Regarding storage, when the media or a worker is started, a new disk volume is created in the compute node where the VM is allocated and, using `virsh` in the compute node, the disk is attached to the virtual machine. In this way, we achieve locality between data and computation.

All the VMs started in the infrastructure are configured using a `cloud-init` (a multi-distribution package that handles early initialization of the cloud instance) script provided at boot time. These scripts format the attached disk volume, initialize directories, obtain and install compressed file artifacts from a web server, and finally run the application. VMs have access to the Internet through the external network, but only master and media can be reached from outside.

3.2. Dynamic Distributed Execution of Encoding Tasks

The application deployed on the top of the virtual infrastructure creates a direct exchange in RabbitMQ named *Video*, which performs load balancing among message consumers, with two queues:

- *Jobs Queue* that stores `VideoJob` messages sent by the master and consumed by a worker. A `VideoJob` message contains all the information and parameters needed to retrieve and encode a video chunk, and the sort of encoders that has to be used.
- *Times Queue* that stores `ProcessingTimes` sent by the workers and consumed by the master.

Firstly, the application in the master node parses the command line arguments, obtains a channel associated to RabbitMQ to create the exchange and the queues, and performs the bindings. Then, for each chunk to be encoded a `VideoJob` message is created and sent (serialized in JSON format) to the RabbitMQ exchange.

Secondly, the application deployed in each worker carries out according to Algorithm 1. Initially, it obtains a RabbitMQ channel and subscribes to the Jobs queue. Thus, after a `VideoJob` message is received, it is deserialized. Then, a port number is selected and a `netcat`, `nc`, process is started in `localhost` to receive the chunks of video from the previously selected port. After that, a `nc` process in the media is started to extract and send the video chunk from the RAW video to the worker. Note that, as each frame has a fixed size, it is possible to extract the portion of the video using for instance the `dd` command from the number of chunk and the number of frames per chunk. When the chunk has been completely downloaded in the disk volume attached to the VM, *ffmpeg* is used to encode each chunk for each value of bit rate. Each encoded chunk is uploaded to the media server. If the encoding process has terminated in the worker node and the encoded chunks have been uploaded to the media server, the message is acknowledged and a message with the time data is sent. On the contrary, when an error occurs, a message is sent informing about the error, so the master node can send the corresponding job message with higher priority.

Finally, the new encoded videos are obtained by joining the encoded video chunks.

Notice that, our proposal is able to work with other codecs. The required adjustments are:

- Creating a new worker VM image with the libraries required for the new codecs and `ffmpeg` recompiled to support them.
- Creating a new script with the call to `ffmpeg` for the new encoder.
- Adapting particular parameters to the new encoder.
- Changing the call to the application running in the master node to pass the name of the new encoder as an argument.

Algorithm 1: Worker video encoder code

```
Obtain a channel from RabbitMQ in Master node;
Create exchanges, queues, and bindings;
/* To receive just one message                    */;
Set prefetch to 1;
Subscribe to RabbitMQ Jobs queue;
for ( each received message ) {
    Deserialize VideoJob from message payload;
    port ← Select a port number;
    Start nc in localhost and listen in selected port;
    Start a nc process in media to send video chunk, using direct access, to worker;
    Wait for video chunk to download;
    for ( each bit rate ) {
        Start a ffmpeg process to encode the video chunk
        ; Wait for encoding process to finish;
        Upload encoded video to media server;
    }
    Send a message to Times exchange with processing times;
    if not error then
        ACK the message;
    else
        NACK the message;
    end
}
```

4. Experimental Setup and Results

This section describes the experimental setup (testbed, measured metrics and experiments) used to validate the proposed architecture. Then a scalability and resource usage study is performed varying the number of worker encoder nodes. Finally, the quality of the encoded videos is obtained for different chunk sizes and bit rates.

All the experiments have been performed applying the H.265/HEVC and VP9 codecs to a Sintel raw video with a resolution of 1080p and YUV 4:2:0 format [25]. This video has a duration of 236.30 s with 6144 frames at 25 fps and a file size of 19.1 GB. The libraries used for each codec have been x265 [23] for HEVC/H.265 and libvpx-vp9 [24] for VP9. The parameters used in the encoding process for each codec is shown in Table 1.

Table 1. Parameters used in the encoding process.

Codec	Tool	Input	Resolution	Encoder	Preset	Threads	fps	Bit Rate (Mbps)
H.265 **VP9**	ffmpeg	RAW video	1080p	libx265 libvpx-vp9	slow -cpu-used 1	4	25	{0.75, 1.5, 3, 4.5}

The encoding has been carried out increasing the number of workers from 1 to 15 to analyze the scalability of the proposed architecture. On the server side, load metrics (CPU and memory utilization) and executed jobs metrics (e.g., number of executed jobs, job execution time, etc.) have been collected for each worker VM during the experiments. Video quality results are evaluated for different video chunk sizes, bit rates, and codecs, and the total codification time has been obtained for each combination.

4.1. Testbed Overview

The experiments have been performed in a private cloud based on OpenStack Ocata (https://docs.openstack.org/ocata) hat has been deployed on five physical nodes with 2 Intel XEON CPU E5-2630-v3@2.40GHz processors with 8 cores per processor, 32 GB DDR3-2200 SDRAM, and a 2 TB HD. Therefore, 16 virtual CPUs (vCPUs) are available per node.

OpenStack controller and network nodes are VMs running in one of the physical nodes [26]. Both VMs are configured with 8 vCPUs, but the controller has 10 GB of RAM while the network node has 6 GB.

During the experiments, VMs of the proposed architecture are distributed among the other four physical nodes as follows. The master and the media VMs are hosted by the same node, while the other three nodes allocate the same amount of workers. Each worker VM is started with 4 GB of RAM and 4 vCPUs and, as stated in Section 3, a local disk of 4 GB is attached to it at boot time. The Management and Overlay networks use a 1 GbE while the Media network use a 10 GbE connection.

4.2. Scalability and Resource Usage

The first experiment evaluates the scalability of the proposed solution. With this purpose, the Sintel video is encoded at a requested bit rate of 1.5 Mb/s with a chunk size of 50 frames with 1, 3, 6, 9, 12 and 15 workers for each codec. The workers are distributed evenly on the compute nodes in order to balance the workload. The master node generates 123 `VideoJob` messages in this configuration. Each message is processed by a worker, which invokes `ffmpeg` with appropriate parameters to apply the two encoders.

4.2.1. Time to Encode and Speedup

Table 2 shows execution time for each codec as the number of workers increases. VP9 is almost 5.5 times faster than H.265, which is maintained when the number of workers increases. Results denote a significant decrease of the execution time in comparison to the sequential version (1 worker), which ranges from 63% to 89% when using from 2 to 15 workers.

Table 2. Execution time (minutes) as the number of workers increases.

Workers	H.265	VP9
1	131.40	23.98
3	45.67	8.80
6	24.93	4.72
9	17.37	3.28
12	15.81	2.73
15	14.41	2.44

Figure 2 depicts the speedup as the number of workers increases, which is obtained from data in Table 2 as the ratio of the sequential version using one worker to the time taken by increasing the number of workers. The speedup is almost ideal until the use of 9 workers (using a total of 12 vCPUs per compute node out of 16 physical cores) with an efficiency greater than 80%. However, there is a reduction for 12 and 15 workers, although the efficiency in these cases is still equal or greater than 70% for 12 workers and greater than 60% for 15 workers. For 12 workers, there are 4 workers per compute node demanding the use of 16 vGPU among the 16 physical existing cores, therefore some of them will be shared with the operating system and the applications running in the compute node. This is even worse for 15 workers, as there are 5 workers per compute node demanding the use of 20 vCPUs of the only 16 physical existing cores, thus some of them will be shared with the operating system, applications, and other encoding processes. A more detailed view of this effect is analysed in the next subsection.

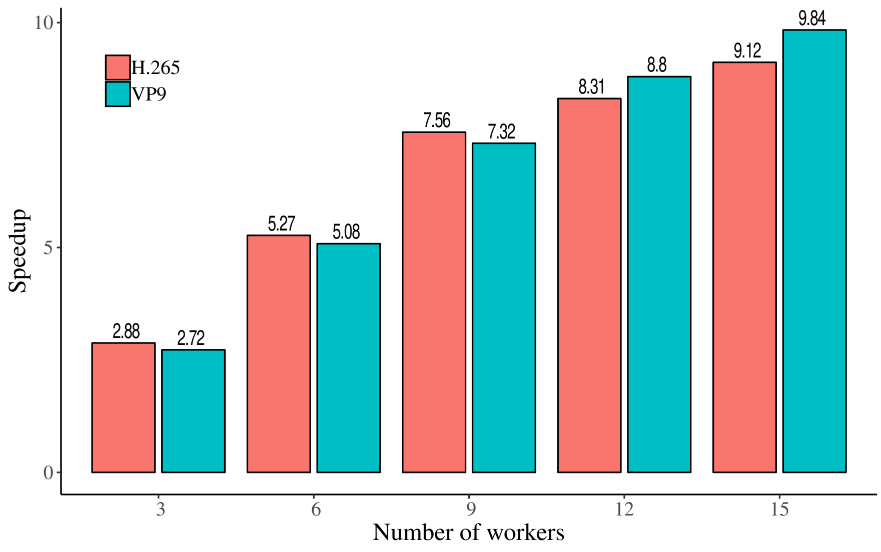

Figure 2. Speedup as the number of workers increases.

Figure 3 shows the number of executed jobs by each worker when encoding using 15 workers.

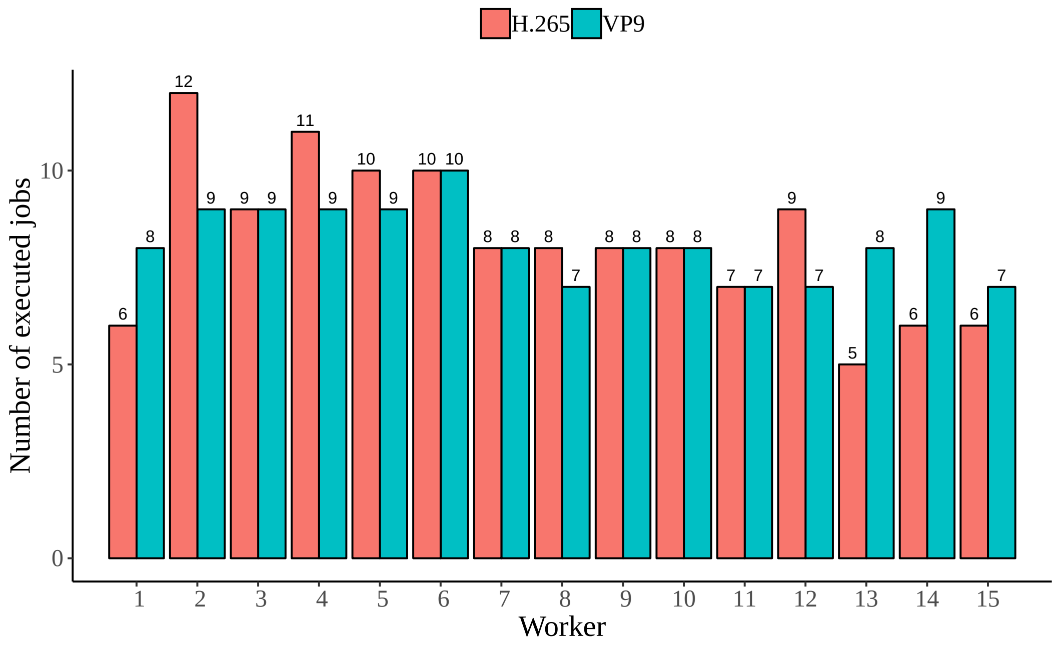

Figure 3. Number of executed jobs per worker with a configuration of 15 workers for the two codecs with chunks of 50 frames, which generates 123 jobs.

This number depends on the computational load of the chunk encoding. Although the chunk size is uniform in all jobs, the encoding time will depend on its content. For example, one worker encodes 5 chunks for H.265, while other encodes 12.

Figure 4 depicts the job scheduling for H.265 when encoding using 15 workers.

As can be seen in this figure, worker 4 performs 5 time-consuming encoding, while worker 12 performs 12 shorter encoding jobs. The total encoding time is determined by the last ending worker. In order to avoid congestion in the media network, when the workers download their video chunks, the master generates job messages at a rate of 1 message per second (this value is provided when the master application is started). Therefore, as can be seen in Figure 4, not all workers start simultaneously.

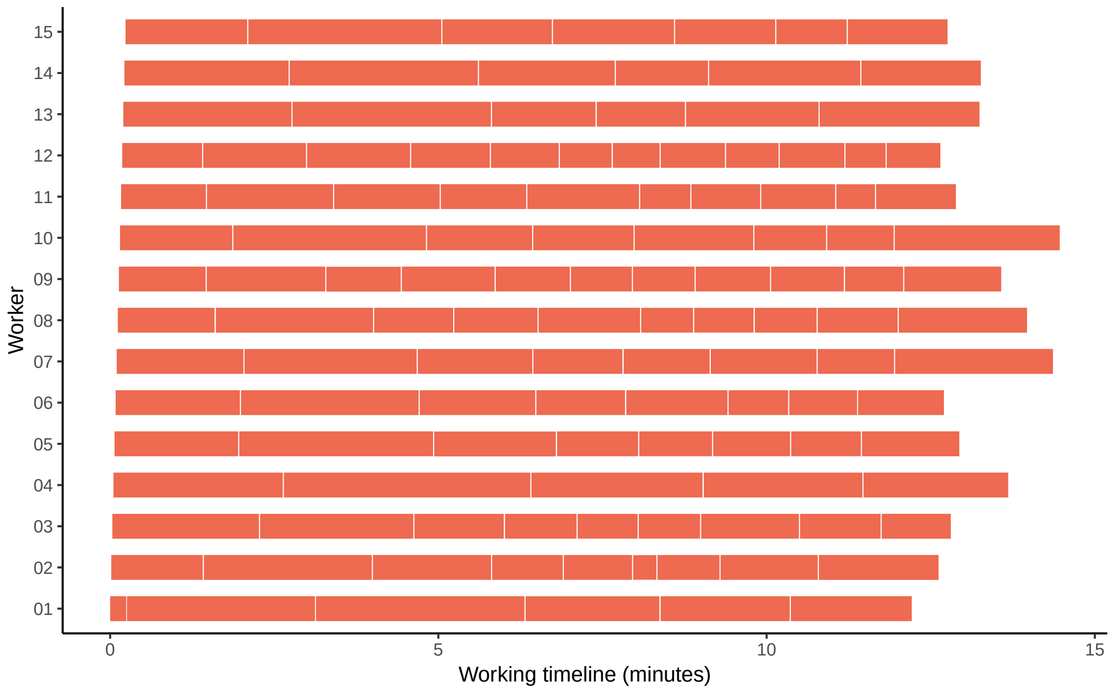

Figure 4. Job scheduling with 15 workers for H.265.

4.2.2. Download, Coding and Upload Times

Mean download, coding and upload times for 3, 9 and 15 workers are shown in Table 3.

Table 3. Mean time in seconds to perform different stages depending on the number of workers, with a chunk size of 50 frames, for H.265 and VP9.

Stage	Workers	H.265	VP9
Download	3	1.70	1.72
	9	1.77	1.80
	15	1.87	1.91
Coding	3	63.66	10.29
	9	68.80	11.03
	15	92.82	13.0
Upload	3	0.66	0.66
	9	0.65	0.68
	15	0.66	0.71

The download time increases with the number of workers as the as it is more likely to have more than one worker simultaneously downloading chunks. The time required to encode the chunks also increases with the number of workers. In the 15-workers configuration (5 workers per compute node allocating 20 vCPUs) per compute node so the physical system is subject to a higher overload than in the 9-workers configuration (3 workers per compute node allocating 12 vCPUs). This has an impact in efficiency reduction, as commented in the previous subsection, and specially for an encoder with high computational demand as H.265, but it could be solved increasing the number of nodes in the infrastructure. Upload times show that, the number of worker nodes does not have such a big impact as the download times. The reason is that the quantity of data to be uploaded is much lower compared to the quantity of data to be downloaded.

4.2.3. Resource Usage

During the experiments, the workers were monitored to obtain CPU and memory usage. All the data collected from the workers is concatenated to obtain the sample Probality Density Function (PDF).

Table 4 shows the CPU usage and memory consumption for the two codecs with 15-workers encoding the video at 1.5 Mb/s. As can be seen, x265 uses all the CPU available, being the encoder with the more intensive CPU use of the two encoders, while libvpx-vp9 uses around a 73% of the available CPU. Moreover, both encoders are similar in terms of memory consumption.

Table 4. Mean memory and CPU usage for 15 workers to encode at a bit rate of 1.5 Mb/s.

Encoder	Memory (MB)	CPU (%)
x265	1213.7	99
libvpx-vp9	1012.4	77.5

4.3. Quality Evaluation

This section analyzes the impact of the chunk size on the final size and quality of the coded video. All the experiments in this section were carried out using 15 workers. The video has been encoded using two codecs H.265 and VP9, four bit rates $\{0.75, 1.5, 3, 4.5\}$ Mb/s, and the frames per chunk varies in the range $\{25, 50, 100, 200\}$. In this experiment, each worker downloads its assigned video chunk only once and encodes it at the four requested bit rates. The goal is to minimize data transference between media and workers.

To analyze the quality of the encoded videos, we have selected the PSNR [27] full-reference metric. It computes the quality by comparing the original video signal against the encoded video signal.

Table 5 shows the PSNR values in dB when the encoding process is done sequentially (without splitting the video) by one worker, while Figure 5 displays the measured quality using PSNR of the coded videos for different chunk sizes.

Table 5. Average PSNR (dB) encoding the video sequentially in one worker at several bit rates with x265 and VP9 encoders.

Encoder	0.75 Mbps	1.5 Mbps	3 Mbps	4.5 Mbps
x265	33.1818	33.4019	33.4218	33.4799
libvpx-vp9	44.7656	47.0066	49.1259	51.2586

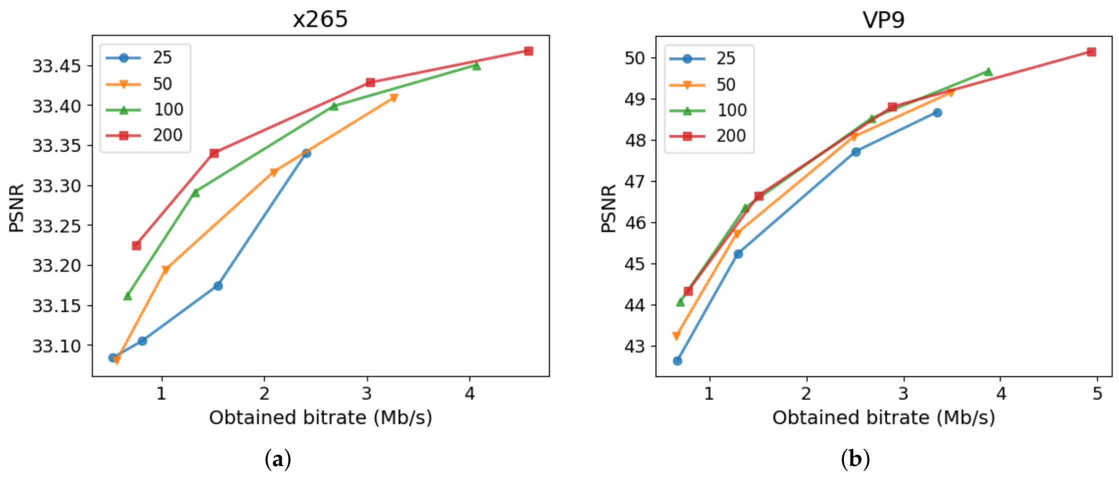

Figure 5. PSNR for the two encoders and four video chunk sizes (25, 50, 100, and 200 frames): (**a**) PSNR for x265 (**b**) PSNR for VP9.

According to the presented information, we can say that the qualities obtained, in general terms, are suitable for H.265 codec and very good for VP9. The difference between PSNR values in both codecs is due to the coding process, since H.265 uses I, P and B frames while VP9 only uses I and P frames, which will introduce more information in each frame and therefore more quality. Besides, in our experiments we have used the x.265 implementation of HEVC, which is not as optimal as the reference software for HEVC called HM (HEVC Test Model).

The PSNR for H.265 (see Figure 5a) is above 33 dB in all cases, improving between 0.10 and 0.25 dB for the same target bit rate when the chunk size is equal to 200 versus size 50. In the case of VP9 (see Figure 5b), these values are higher than 43 dB in most cases. These PSNR values are very high, so we can consider that the encoded videos have a high quality.

If we make a general analysis of the quality metrics, on one hand we can indicate that the codec most affected by the change in chunk size and bit rate is H.265, while VP9 has a more stable behavior. On the other hand, the codec with the best quality characteristics is VP9, because it has a higher PSNR values. As we can see in Figure 5a, the H.265 codec has a different behavior when the chunk size is very small. In this case, for a size of 25, the encoder is never able to reach the target bit rates and therefore the PSNR decreases compared to the traditional encoding process (values presented on Table 5). On the contrary, when the chunk size is increased to 100 and 200, the PSNR values are already more similar. When we analyze the VP9 codec, we can see another behavior (see Figure 5b): with low bit rates (0.75 and 1.5 Mbps), we can see that any chunk size reaches that target bit rate; but for the other bit rates, only the chunk size 200 can reach the target bit rate. For VP9, the proposed solution does not reach the quality obtained for a sequential encoding (see Table 5), but this small decrease of quality is compensated by the big improvement obtained in the encoding time.

Finally, to sum up this quality evaluation, if we compare the values of Table 5 and Figure 5, we can indicate, in general terms, that the encoding system presented in this paper provides a similar quality to the sequential encoding process. When the chunk is smaller than 100, the quality drops a bit from the values presented in the Table 5. This difference is so small that it is not noticeable to an end user, but the coding speed is greatly improved (see Table 2 and Figure 2).

5. Discussion

A qualitative comparison between the related papers presented in Section 2 and our proposal is summarized in Table 6.

Table 6. Qualitative comparison of the previous work with our proposal.

Proposal	Type of Enconding (RAW or Transcoding)	Codec Used	Chunk Size (Frames)	Paradigm	Video Resolution	Tested on...	Quality Analysis
[14]	RAW encoding	H.264	250	Map-Reduce	854 × 480 (ED)	Public cloud Amazon EC2	No
[15]	RAW encoding	H.265	Variable, content-aware video segmentation	Map-Reduce	3840 × 2160 (4K)	Private architecture 1 master—20 workers	No
[16]	RAW encoding	VP8	24 and 96	Serverless	3840 × 2160 (4K)	Public cloud AWS Lambda	Yes
[17]	transcoding	H.264	GOP size	Map-Reduce	-	-	No
[18]	transcoding	H.264	32, 64, 128, 256 and 512 MB	Map-Reduce	320 × 240	Private architecture 1 master—28 workers	No
[19]	transcoding	H.264	32, 64 and 128 MB	Map-Reduce	-	-	No
[20]	transcoding	H.264	-	Map-Reduce	-	Private architecture 4 nodes	No
[21]	transcoding	H.264	-	Cluster of machines	-	Cluster of 2 to 20 machines	No
Our proposal	RAW encoding	H.265 and VP9	25-50-100-200	Message broker	1920 × 1080 (FHD)	Private cloud 5 nodes—1 to 15 workers	Yes

There are some proposals that work with RAW videos and others work performing only transcoding tasks. Our proposal is based on the processing of RAW content, whose videos take up more disk space and the management of these files must be as efficient as possible to improve encoding times. Moreover, our proposal uses the latest stable codecs (H.265 and VP9), while most works use the previous codecs such as H.264 and VP8. Regarding the paradigm used, most works use Map-Reduce or cluster of machines, while our proposal is based on the asynchronous message broker paradigm over a virtual infrastructure. Therefore, our proposal is well adapted to leverage the elasticity of computing resources offered by the cloud and has been tested on a private cloud infrastructure with several workers distributed in different nodes. We have performed a video quality analysis to verify that the encoded videos have a similar quality to the videos encoded using a sequential encoding system.

Some approaches use hardware accelerators, as GPUs, to increase the performance of the encoder on a compute node. In these approaches, the GPU parallelizes specific time consuming parts of the video encoding process (mainly Motion Estimation and Intra-prediction) [9,28–30]. However, the availability of nodes with GPU is not always possible, particularly on current heterogeneous cloud infrastructures, and the virtualization and remote use of GPUs on the cloud is another active area of research today [31]. Nevertheless, our solution could exploit the presence of GPUs in the VMs, and their combination could provide higher performance.

6. Conclusions

We have proposed a cloud-based distributed architecture to accelerate video encoders that could be offered by cloud providers as a coding platform to their users or can be deployed on premises. Its main features are: elasticity, fault-tolerance, hyperconvergence, adaptability to different video codecs, and automation.

The architecture relies on three node types (master node that schedules jobs, media node that stores raw and coded videos, and worker nodes that perform the coding) to be deployed on a cloud infrastructure. This paper has also detailed the deployment of our proposal on a private cloud to fine tune the infrastructure. The goal is to achieve hyperconvergence and near bare metal network performance by using SR-IOV in a provider network. Moreover, an application based on a message middleware is deployed on the top of this virtual infrastructure to parallelize the execution of video encoders `x265` and `libvpx-vp9`. The proposal has been evaluated in terms of scalability, workload, and job distribution ranging the number of workers from 1 to 15 at different bitrates and chunk sizes.

According to the quality evaluation, in terms of PSNR, the proposed cloud-based encoding system provides a similar quality to the sequential encoding process while improving encoding time, which can decrease up to 89% when using 15 workers. Moreover, the speedup is almost ideal until the use of 9 workers with an efficiency greater than 80%.

As future work, we are testing the behavior of the architecture with AV1 codec and the impact on performance and quality of other non-uniform video splitting schemes and other higher resolutions.

Author Contributions: Conceptualization, J.G.-A. and M.G.-P.; methodology, J.G.-A. and J.M.C.; software, J.G.-A. and R.P.-O.; validation, J.G.-A., R.P.-O. and M.G.-P.; investigation, all authors; resources, J.M.C.; data curation, R.P.-O.; writing—original draft preparation, all authors; writing—review and editing, all authors; supervision, J.M.C.; funding acquisition, J.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Research and Development Grants RTI2018-098156-B-C55 and the Generalitat Valenciana under the project GV-2016-002.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Sandvine. Global Internet Phenomena Report. White Paper. 2019. Available online: https://www.sandvine.com (accessed on 10 march 2020).
2. Sandvine. The COVID-19 Global Internet Phenomena Report. White Paper. 2020. Available online: https://www.sandvine.com/press-releases/sandvine-releases-covid-19-global-internet-phenomena-report (accessed on 20 may 2020).

3. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. White Paper 1551296909190103, CISCO. 2019. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html (accessed on 10 march 2020).
4. Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimed.* **2011**, *18*, 62–67. [CrossRef]
5. Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 560–576. [CrossRef]
6. Mukherjee, D.; Bankoski, J.; Grange, A.; Han, J.; Koleszar, J.; Wilkins, P.; Xu, Y.; Bultje, R. The latest open-source video codec VP9—An overview and preliminary results. In Proceedings of the 2013 Picture Coding Symposium (PCS), San Jose, CA, USA, 8–11 December 2013; pp. 390–393. [CrossRef]
7. Sullivan, G.J.; Ohm, J.; Han, W.; Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1649–1668. [CrossRef]
8. Pastuszak, G.; Abramowski, A. Algorithm and Architecture Design of the H.265/HEVC Intra Encoder. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 210–222. [CrossRef]
9. Rodríguez-Sánchez, R.; Martínez, J.L.; De Cock, J.; Sánchez, J.L.; Claver, J.M.; Van de Walle, R. Low delay H.264/AVC bidirectional inter prediction on a GPU. In Proceedings of the 2013 IEEE International Conference on Image Processing, Melbourne, Australia, 15–18 September 2013; pp. 2111–2115. [CrossRef]
10. Asif, M.; Majeed, S.; Taj, I.A.; Bin Ahmed, M.; Ziauddin, S.M. Exploiting MB level parallelism in H.264/AVC encoder for multi-core platform. In Proceedings of the 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), Doha, Qatar, 10–13 November 2014; pp. 125–130. [CrossRef]
11. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing (NIST Special Publication 800–145)*; Technical Report; NIST: Gaithersburg, MD, USA, 2011. [CrossRef]
12. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A View of Cloud Computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
13. Amazon Web Services. Netflix on AWS. Available online: https://aws.amazon.com/solutions/case-studies/netflix/ (accessed on 20 march 2020).
14. Pereira, R.; Azambuja, M.; Breitman, K.; Endler, M. An Architecture for Distributed High Performance Video Processing in the Cloud. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, 5–10 July 2010; pp. 482–489. [CrossRef]
15. Jeon, M.; Kim, N.; Lee, B. MapReduce-Based Distributed Video Encoding Using Content-Aware Video Segmentation and Scheduling. *IEEE Access* **2016**, *4*, 6802–6815. [CrossRef]
16. Fouladi, S.; Wahby, R.S.; Shacklett, B.; Balasubramaniam, K.V.; Zeng, W.; Bhalerao, R.; Sivaraman, A.; Porter, G.; Winstein, K. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA, USA, 27–29 March 2017; USENIX Association: Berkeley, CA, USA; pp. 363–376. Available online: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi (accessed on 10 may 2020).
17. Lao, F.; Zhang, X.; Guo, Z. Parallelizing video transcoding using Map-Reduce-based cloud computing. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS), Seoul, Korea, 20–23 May 2012; pp. 2905–2908. [CrossRef]
18. Kim, M.; Cui, Y.; Han, S.; Lee, H. Towards efficient design and implementation of a hadoop-based distributed video transcoding system in cloud computing environment. *Int. J. Multimed. Ubiquitous Eng.* **2013**, *8*, 213–224. [CrossRef]
19. Song, C.; Shen, W.; Sun, L.; Lei, Z.; Xu, W. Distributed video transcoding based on MapReduce. In Proceedings of the 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS), Taiyuan, China, 4–6 June 2014; pp. 309–314. [CrossRef]
20. Mingang, C.; Wenjie, C.; Zhenyu, L.; Lizhi, C. Parallel video transcoding using Hadoop MapReduce. *J. Netw. Comput. Appl.* **2016**, *1*, 7–11. [CrossRef]
21. Elkabbany, G.F.; Moussa, M.M. Accelerating video encoding using cluster computing. In *Multimedia Tools and Applications*; Springer: Berlin/Heidelberg, Germany, 2020. [CrossRef]
22. FFmpeg: A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video. Available online: https://www.ffmpeg.org/ (accessed on 10 december 2019).

23. MulticoreWare Inc. x265 HEVC Encoder/H.265 Video Codec. Available online: https://bitbucket.org/multicoreware/x265/wiki/Home (accessed on 10 december 2019).
24. WebM Project. The WebM VP8/VP9 Codec SDK. Available online: https://github.com/webmproject/libvpx/tree/master/vp9 (accessed on 10 december 2019).
25. Blender Foundation. Sintel, the Durian Open Movie Project. Available online: https://media.xiph.org (accessed on 10 december 2019).
26. Chirivella-Perez, E.; Gutiérrez-Aguado, J.; Claver, J.M.; Calero, J.M.A. Hybrid and Extensible Architecture for Cloud Infrastructure Deployment. In Proceedings of the 2015 IEEE International Conference on Computer and Information Technology, Liverpool, UK, 26–28 October 2015; pp. 611–617. [CrossRef]
27. Winkler, S.; Mohandas, P. The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. *IEEE Trans. Broadcast.* **2008**, *54*, 660–668. [CrossRef]
28. Rodríguez-Sánchez, R.; Martínez, J.L.; Fernández-Escribano, G.; Sánchez, J.L.; Claver, J.M. A Fast GPU-Based Motion Estimation Algorithm for H.264/AVC. In *Advances in Multimedia Modeling*; Schoeffmann, K., Merialdo, B., Hauptmann, A.G., Ngo, C.W., Andreopoulos, Y., Breiteneder, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 551–562.
29. Xiao, B.; Wang, H.; Wu, J.; Kwong, S.; Kuo, C.J. A Multi-Grained Parallel Solution for HEVC Encoding on Heterogeneous Platforms. *IEEE Trans. Multimed.* **2019**, *21*, 2997–3009. [CrossRef]
30. Luo, F.; Wang, S.; Wang, S.; Zhang, X.; Ma, S.; Gao, W. GPU-Based Hierarchical Motion Estimation for High Efficiency Video Coding. *IEEE Trans. Multimed.* **2019**, *21*, 851–862. [CrossRef]
31. Gutiérrez-Aguado, J.; Claver, J.M.; Peña-Ortiz, R. Toward a transparent and efficient GPU cloudification architecture. *J. Supercomput.* **2019**, *75*, 3640–3672. [CrossRef]