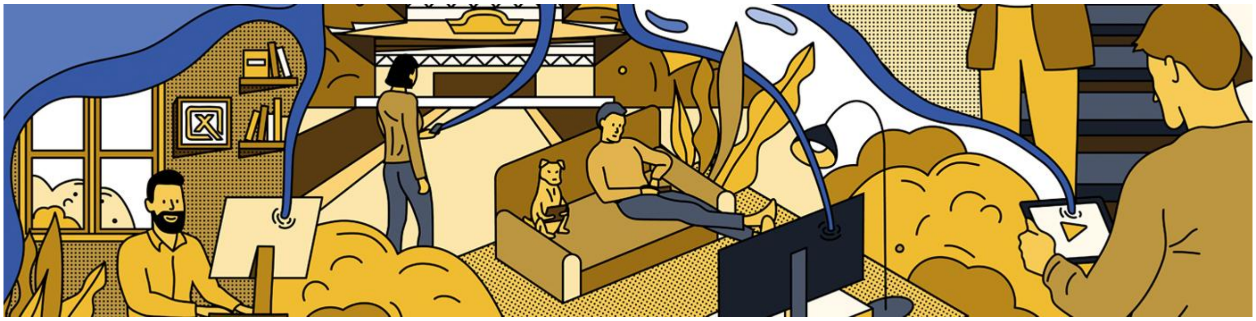


Transitioning from “Density per CPU” to “Pixel per Dollar”

Marc Baillavoine, September 24th, 2019

How Many Channels on my Dual-Xeon Gold 6140?

“How many OTT channels can you have in a Dual-xeon Gold 6140” is a question I have heard hundreds of times in the past. Why: the density (i.e. the total number of video streams that a given hardware can process in “real time”) has always been considered as a key factor for determining the overall price of a video delivery solution. Hence, many companies have been investing a lot of money in making their video delivery chain (and especially the video transcoder) more “efficient” (efficient means “reducing the number of CPU cycles but keeping the quality as is”) to lower the TCO of their solution. Open source softwares (such as libx264) also have a long history of optimizing their code to run faster on any kind of platform. x264 is now even being used as a benchmark tool for measuring the CPU speed.

New software paradigms, born with the cloud, shed a new light on this story. CPU has now become a resource that can be provisioned anywhere, at any time, … at any price, and modern software architectures can leverage that diversity to dramatically lower the sacrosanct “price per channel”. Moreover, a “channel” (the one you choose with your remote) may not be the optimal object to make this comparison with the diversity of flavours OTT brings in the game.

Let’s see how modern softwares can leverage new development and architecture paradigms to dramatically improve the video delivery TCO and transition from a “channel per cpu” to a “pixel per dollar” logic.

Video Delivery 101

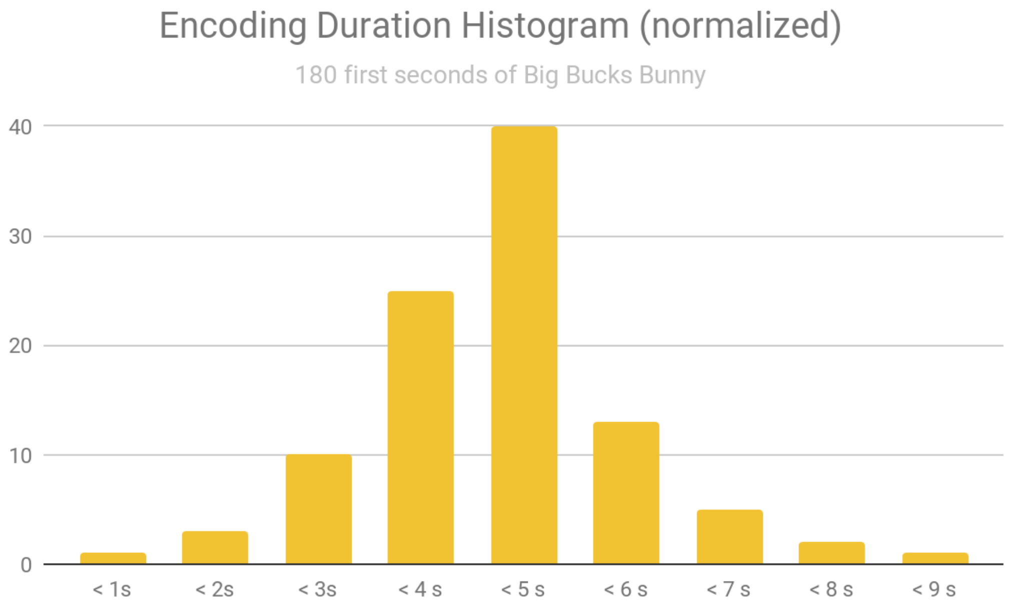
Rate Distortion Optimizations

Video transcoding is complex stuff. Video standards, since the early days of H.261 (predecessor of H.262, also known as MPEG2), define the way the decoder shall behave when receiving a bitstream. But it never describes the way the “encoding” should be done, for one reason: there are billions of billions of possibilities for encoding a video sequence, and the vendor implementing the encoder is solely responsible for making these choices. Without diving too deep into the details:

- each picture is split into ‘units’ (“macroblocks” in H.264/AVC, “Coding Tree Unit” in H.265/HEVC) that are typically 16x16 pixels
- then for each unit (there are 8160 units in a 1080p image), the encoder has to decide:

1. the type of this unit (I, P, B),
2. the size of the prediction block,
3. the size of the transform,
4. the motion vectors for each of the prediction blocks,
5. the quantizer to use,
6. etc.

Every combination will give a different quality and a different rate. The complexity of a video encoder is in choosing the mode that will lower the number of bits used to encode one “unit”, while maximising the video quality: that is what we call the “Rate-Distortion Optimization”.



In an ideal world, the encoder would try all the possible combinations and choose the best compromise. But it would literally take years to process 1 hour of content, even on the latest CPU or GPU generation. Hence, an encoder has to make “early exit” decisions to dramatically lower the number of possible combinations. This is usually done through a wise mix of thresholds that will prune some large portions of the decision tree. These thresholds are compared to metrics computed on the input content, resulting in a highly non-linear processing time. It sounds obvious, but that’s the reason why it takes significantly less time to encode a SMPTE colorbar than any real-life content, as depicted by the above histogram.

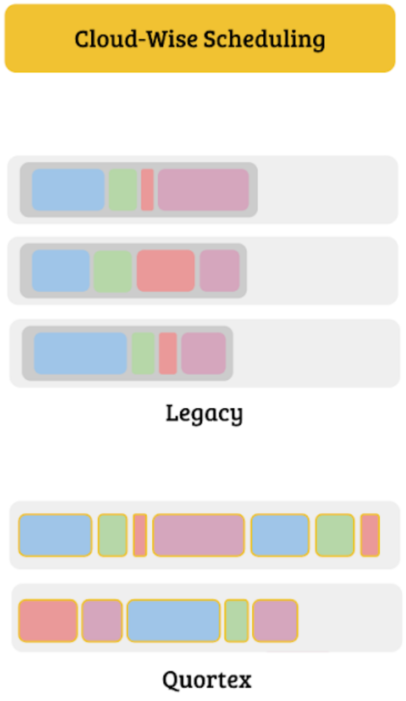
CPU Cycle Wasting

Let us say you are moving from SF to LA, and let us imagine:

- that you rent exactly one moving truck for each room of your house,
- that you have to guess the size of the truck.

What will happen ? You will probably take a lot of margin so that all the pieces of furniture of one room fill within one truck. You will end up with a lot of free space in most of the trucks. Even worst, you may be short of storage for one of your rooms, but you will have to leave the extra furniture because you cannot use the truck of another room for moving it.

Sounds stupid, right? This is exactly what happens today with legacy software running in the cloud! The several tasks of encoding a channel (the room) are tied to a hardware (the truck). Even though all the vendors claim to be pure software house, that software is physically tied to hardware. In other words, one channel will be processed on only one hardware.



This legacy architecture, combined with the heterogeneity of encoding times (as seen above) leads encoder vendors to take significant margins on the hardware, as they need to be able to sustain “real-time” encoding for the worst case (even though the worst case only occurs 1 time in 1000). If you inspect a live transcoding farm today, is is very likely that ~70% of its CPU is idle because of this margin. In a nutshell, vendors have been optimizing their code for decades to use less and less CPU cycles (using the latest SIMD instructions on the processor), but 70% of these cycles end up wasted because of the non-linearity of video transcoding processing time.

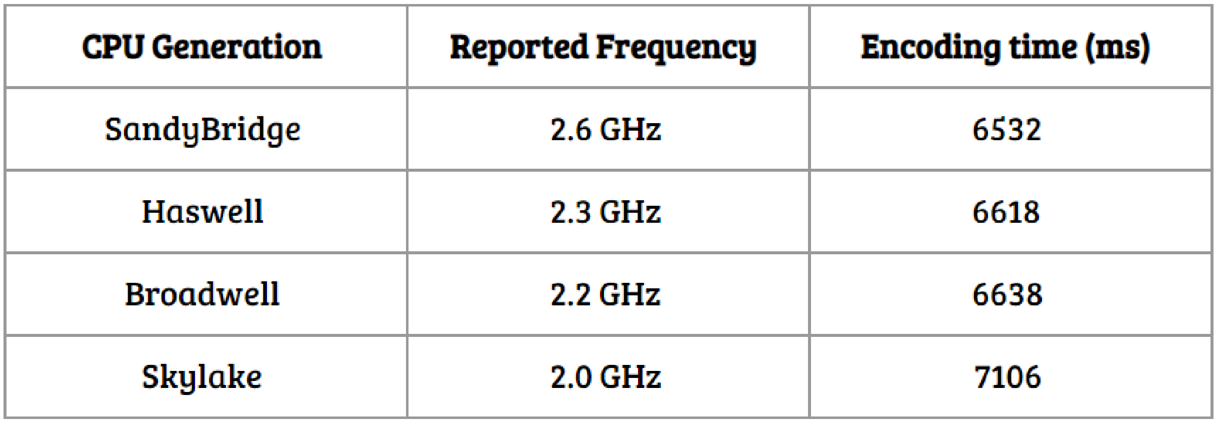
A modern video delivery architecture needs to truly segregate those stacks. By doing so, the cloud can be considered as a very large CPU where tasks (process) can be scheduled, regardless of the underlying hardware. This has many excellent properties, the most immediate one being a significant CPU cycles saving. Keep in mind that 70% of the CPU cycles are wasted today because of the margins that are taken because of that software adherence to one server. If you can dynamically distribute the workload amongst all the CPUs (i.e. any process can run on any server), you will save a lot of CPU cycles.

The Cloud is the CPU

CPU Abstraction

Hardware virtualisation is one of the key characteristics of the cloud. Virtualisation, in that sense, means “abstraction”: although some cloud providers give reasonable information on the underlying hardware, it is almost impossible to precisely and repeatedly know the type of CPU that you will be assigned when reserving a Virtual Machine. Moreover, you are likely to get a portion of a CPU that will be logically split over several users, and you may not get the same CPU tomorrow morning or if you spin up a VM in a different region. Trying to map “channels per CPU” is meaningless because you don’t have a clue what the CPU model is. For the same price, in the same region and at the same time, you may even get pretty different results. On top of that, cloud providers have varying definitions of a “vCPU”: while Google offers a unified definition of a vCPU, AWS has a definition of a vCPU that explicitly depends on the machine type (c5, t4, …) you request.

As an example, GCP can offer “Sandy Bridge” (released in 2011), “Haswell” (2013), “Broadwell” (2014) or “Skylake” (2015) SKUs. As you can guess, such a disparity will lead to significant performance difference, even though Google did its best on the virtualization layer to balance the allocated CPU frequency so that those different SKUs appear similar in performance. The table below reports average encoding time for 10 seconds segments. Differences of up to 10% can be seen in performance for the exact same price (surprisingly, the oldest CPU generation give the best results!)



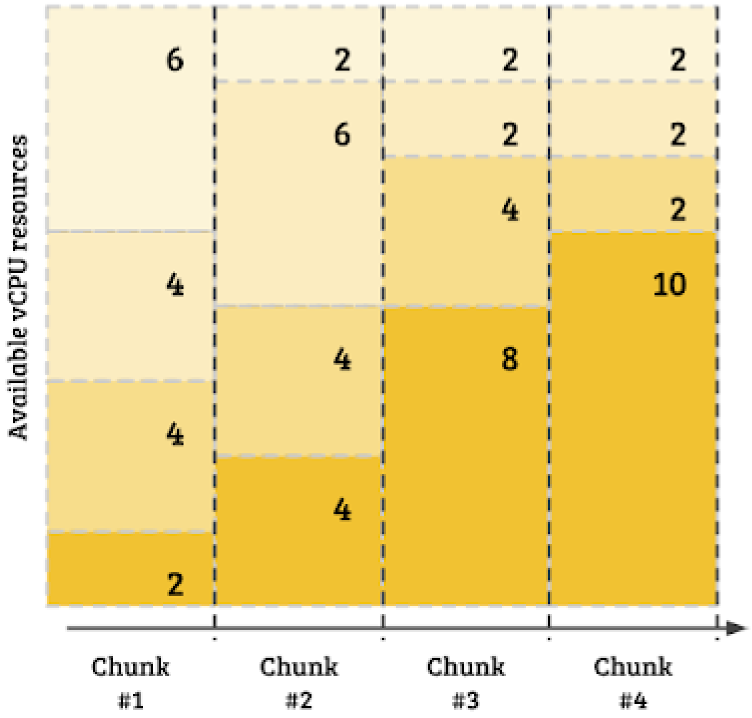
CPU Generation	Reported Frequency	Encoding time (ms)
SandyBridge	2.6 GHz	6532
Haswell	2.3 GHz	6618
Broadwell	2.2 GHz	6638
Skylake	2.0 GHz	7106

What is true for one single cloud provider becomes obvious when you mix cloud providers. For instance, AWS has c5 machines which are based on the latest Xeon Platinum CPUs. One vCPU of such instances will transcode much faster than any of the above machines (while, of of course, being more expensive).

Auto Adaptive, Self Scalable Software

If you want to run software that will behave the same, regardless of the allocated CPU type, you need to have an architecture that will adapt and scale to accommodate this heterogeneity. If you do not do that, you will end up with a software that will be specifically designed for a hardware type, thus losing one of the main benefits of the cloud: segregation between the software and the hardware stacks.

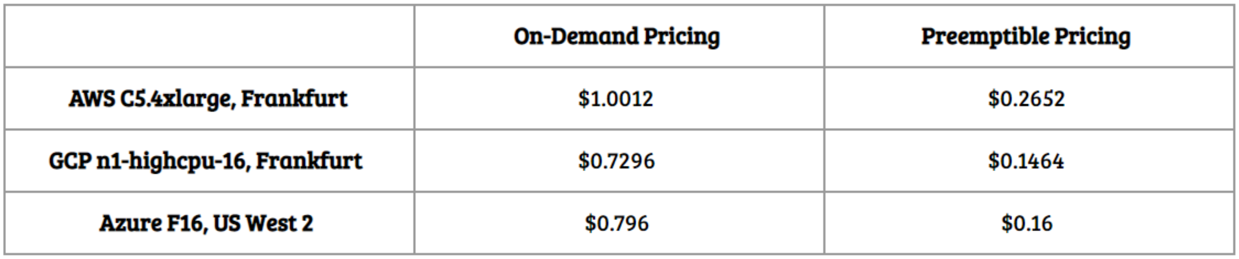
CPU Statmux



As said earlier, legacy software process one channel on one given server: they cannot migrate their processing from server to server. Cloud-Native software, as opposed to that, have the ability to “load balance” their processing amongst several “processing units” running on different hardware. Enabling such a capability on a video processing saves a lot of CPU cycles. Similarly to the famous “Bitrate StatMux” (where you can dynamically adjust the bitrate for each channel within a satellite, fixed-bandwidth transponder), Quortex has the ability to perform “CPU Statmux” so as to adjust, in real time, the number of allocated CPU cycles per stream. This is done at the cloud level and is not limited to the channels running on a single hardware. Deep Learning is a key tool to predict the CPU cycles that will be used by the transcoder. This is a great answer to the non-linear encoding time issue, and, according to our tests, can save up to 30% of the CPU cycles.

CPU Abstraction

According to Google, “Preemptible VMs are highly affordable, short-lived compute instances suitable for batch jobs and fault-tolerant workloads. Preemptible VMs offer the same machine types and options as regular compute instances and last for up to 24 hours”. AWS has the same concept with “Spot” instances, while Azure has this in Beta program (“Low priority VMs”).



	On-Demand Pricing	Preemptible Pricing
AWS C5.4xlarge, Frankfurt	$1.0012	$0.2652
GCP n1-highcpu-16, Frankfurt	$0.7296	$0.1464
Azure F16, US West 2	$0.796	$0.16

To go back to our moving truck story, let us imagine that the renter has an offer for trucks that would cost 20% of the normal price, but those trucks can stop anywhere between SF and LA. The renter ensures that another truck will immediately pop-up to take over. You will of course say “no” because you do not want to take out the furniture from one truck to put it in the replacement truck. But what if this is automatic ? What if the replacement truck can take over while the first truck is still moving ? Then you will of course consider that option seriously.

Modern software can run on preemptible VMs because they were designed to be fault-tolerant. This is a killer feature, as the price of the infrastructure suddenly drops by 80%.

How many channels per $2 500 monthly?

CPU Abstraction

We have seen that truly cloud-native software can leverage new software development paradigms:

- By using an architecture that breaks the adherence between the processing and the underlying hardware,
- By statmuxing the CPU demanding tasks (the more channels, the more efficient),
- By using preemptible VMs.

It is easy to understand that giving a “Channel density by CPU” is absolutely irrelevant with that approach. At Quortex, the software we have developed makes use of these properties and we have transitioned to a “Pixel rate per dollar” approach. It is of course still worth spending time on optimizing the core of a software, but the cloud-native nature of our solution dramatically reduces the infrastructure price. An example is shown below for an infrastructure TCO for 20 channels encoded following the HLS Authoring Specifications, showing up to 80% of cost savings on the infrastructure.

	Legacy Software	Quortex Just-In-Time
Number of C5.4x large instance for 1 channel	1	1
Number of C5.4x large instance for 20 channel	20	16
Number of C5.4x large instance for 20 channel w/ CPU Statmux	20	12
Hourly Price per Instance	$1.0012	$0.2652
Monthly TCO for 20 channels	$14617	$2323