BEAMR

Integrating Beamr Video Into a Video Encoding Workflow

By: Jan Ozer

Beamr Video is a perceptual video optimizer that significantly reduces the bitrate of video streams without compromising quality, enabling a better streaming user experience and significant savings on delivery costs. This document demonstrates how Beamr Video can be integrated seamlessly into your video encoding workflow.

Beamr Video Overview

Beamr Video is a perceptual video optimizer that significantly reduces the bitrate of video streams while preserving their full resolution and quality. The optimization process reduces storage and bandwidth costs, and enables the delivery of higher quality video over bandwidth-constrained links. Beamr Video runs on 64-bit Ubuntu 12.04 and RHEL 6.x and compatible systems, and is applied to existing MP4/H.264 files after adaptive bitrate (ABR) encoding and before packaging for streaming, or to H.264 elementary streams targeted for Blu-ray production.

This document provides an overview of how Beamr Video works and the options available for integrating Beamr Video into a production workflow, as well as an example of an existing Beamr customer that is integrating Beamr Video into its workflow.

This document is intended as an overview of the Beamr Video implementation process. It refers frequently to the Beamr Video User Manual.

Beamr Video Optimization

The Beamr Video optimization process is applied to compressed H.264 files, prior to any ABR packaging. Beamr Video offers two settings, Best and High, which are selected via the quality flag in the command string. Best quality provides the optimum quality output, and an output stream that is perceptually identical to the input stream, even when viewed by an *expert* viewer. High quality delivers a more compact stream that is perceptually identical to the input stream when viewed by an *average* user.

As shown in Figure 1 below, the optimization process is a closed loop system built around the perceptually identical concept. During the process, the original compressed frames are decoded, and then re-encoded using more aggressive compression parameters, delivering a more compact frame. This frame is then decompressed, and compared to the original using Beamr's Quality Measure Analyzer. This returns a score that numerically represents whether the frame is perceptually identical to the original frame for an average

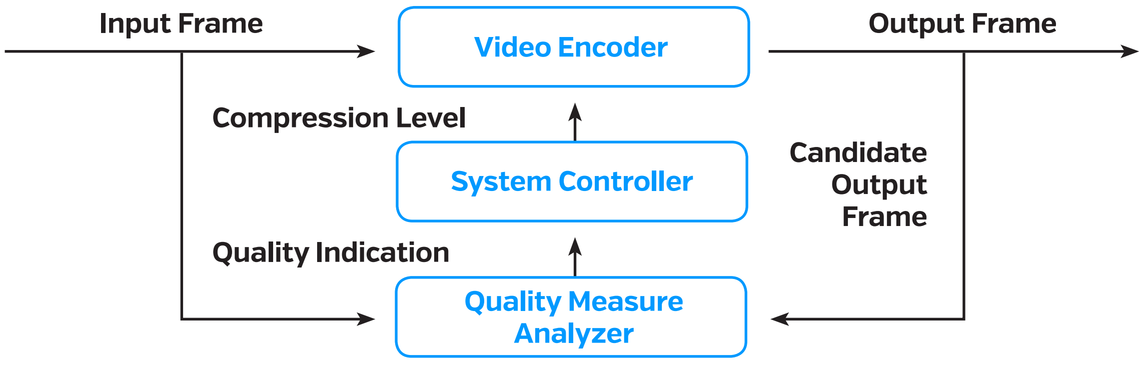


Figure 1. *Beamr Video optimization process*

or expert viewer, depending upon the selected quality setting. If the score is within a certain threshold, Beamr Video accepts the frame and moves to the next.

If the score exceeds this threshold, it indicates that more savings may be achievable, so Beamr Video encodes at a higher compression level and analyzes again until the score is within the target threshold. Conversely, if the score is less than the threshold value, it indicates that the frame is no longer visually identical to the original, so the optimizer re-encodes at a lower compression level until the frame is within the target quality threshold.

To perform the H.264 encoding, Beamr Video comes pre-installed with the H.264/AVC reference (JM) encoder, yet users can install other plug-in encoders such as the x264 encoder for faster operation. During re-encoding, Beamr Video places IDR frames in the output video stream at all locations that have IDR frames in the input video stream, ensuring that files prepared for ABR stream switching, remain synchronized. Similarly, Beamr Video also preserves the H.264 profile, level and maximum number of B frames from the original stream.

Beamr Video Workflow Integration

Beamr Video is integrated into an existing video processing workflow after the encoding stage, and before the packaging and DRM stage. The initial encode is required in order to provide a quality reference to Beamr Video, which preserves the quality of the encoded file while reducing its bitrate as much as possible. Beamr Video also preserves all the attributes of its input file, so the DRM & packaging step is performed on the optimized file in exactly the same way as it would have been performed for the regular encoded file.

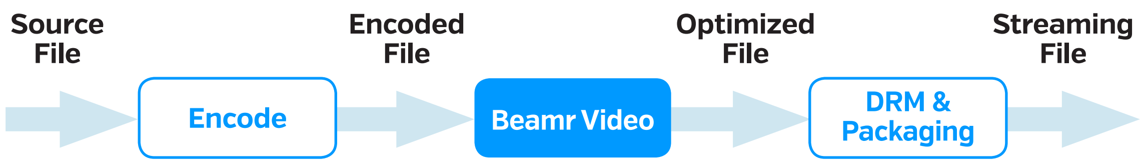


Figure 2. *Beamr Video integration into video processing workflow*

Getting Started With Beamr Video

Briefly, Beamr Video is supported on 64-bit versions of Ubuntu 12.04 and compatible, and RHEL 6.x and compatible. The Beamr Video User Manual contains complete installation instructions.

Beamr Video uses a cloud-based licensing system, which controls the number of instances that can run simultaneously on a single server (node) or on several nodes. When running Beamr Video for the first time, enter the License ID received from Beamr and the number of required licenses to activate on the current node. The number of licenses determines the maximum number of concurrent instances of Beamr Video that can run on the current node. For servers that are not connected to the Internet, an alternative licensing system based on a USB dongle is available. License management is detailed in the Beamr Video User Manual.

In operation, Beamr Video supports H.264-encoded video in MP4 and MOV containers, and AAC or AC-3 audio streams. Beamr Video also supports elementary stream input formats like H.264, 264, AVC and BSF, outputting an H.264-encoded elementary stream.

Note ***Beamr Video processes only the first video track. Also, Beamr Video only supports progressive video and not interlaced video.***

Beamr Video generates the highest optimization ratios for high-quality video streams, such as high bitrate streams bound for Blu-ray production, or 1080p and 720p streams within adaptive groups. Using Beamr Video on lower quality streams typically results in lower optimization ratios. If Beamr Video detects that the quality of the input video stream is very low – and it cannot be further optimized without degrading visual quality – the input file is copied to the output.

Running Beamr Video

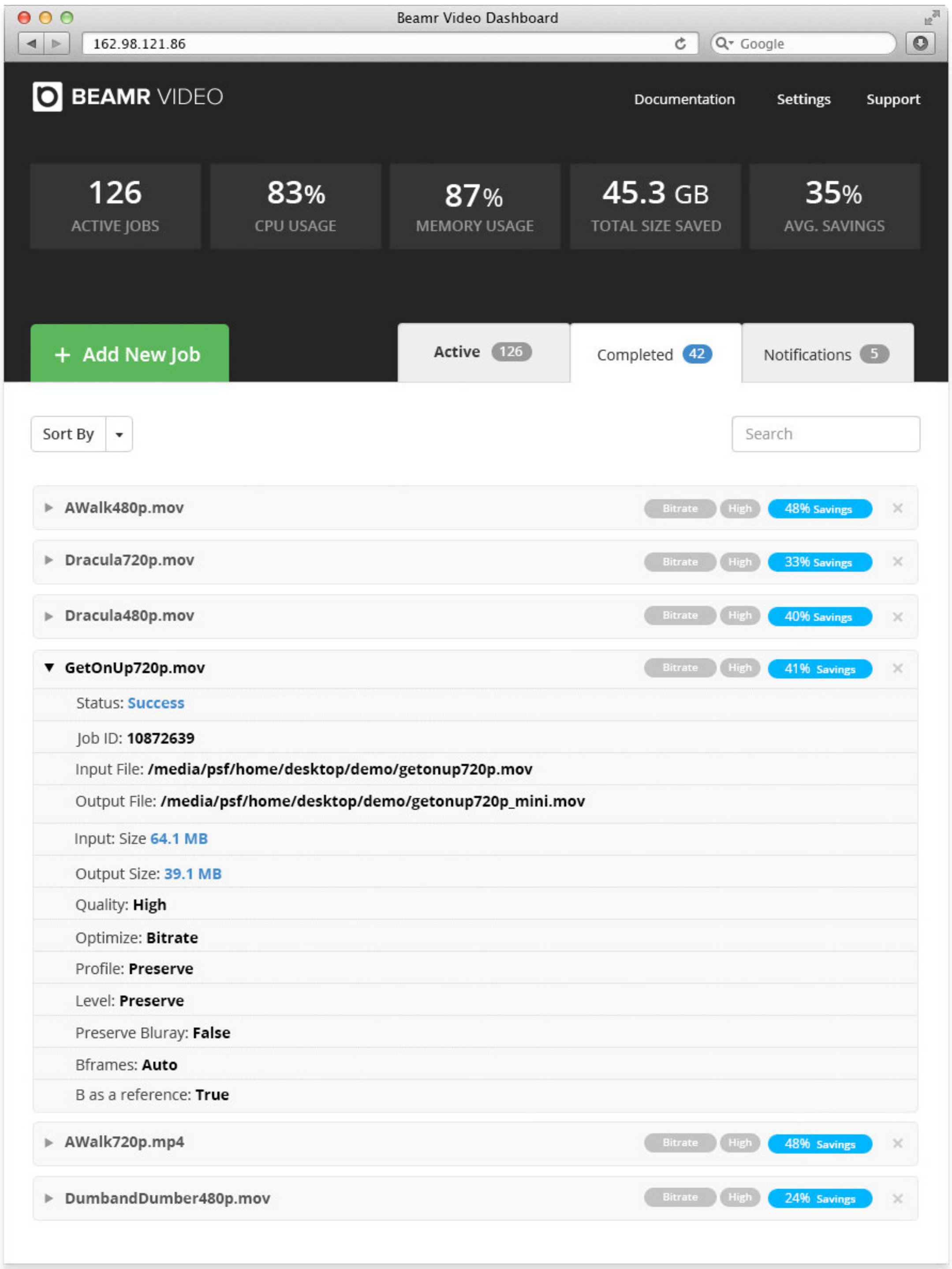
Beamr Video offers multiple modes of manual operation, including command line and web-based GUI. Those integrating Beamr Video into their existing workflows will find it useful to learn the command line operating modes first. At a high level, there are two ways to run Beamr Video; as a standalone operation and as a server-based operation, which spawns instances of Beamr Video on multiple cores in parallel. The implications of these modes are shown in Table 1.

	Server-based Operation	**Standalone Operation**
Command	`beamrvideo_mgr` `beamrvideo`	`beamrvideo_si`
Instances invoked	Multiple	Single
What happens	Launches a server that can manage multiple instances of Beamr Video processing multiple files	Initiates standalone job (one input/one output) managed by a single Beamr Video instance
Processing	Segmented – each input file is divided into multiple segments processed using multiple instances	Input file is processed from start to finish
Queue management	Queue managed by server	External
Fault tolerance	File processing and queue	None
Dashboard (web management)	Full access to management and reporting functions	None

Table 1: *Beamr Video operating modes*

Starting a job via the **`beamrvideo_mgr`** or **`beamrvideo`** commands launches a server that can manage multiple inputs and multiple instances. In this mode, all input jobs are divided into segments processed by the Beamr Video instances available on the computer for faster single-file operation. The server manages the input queue and all processing, providing fault tolerance for both; closing the terminal window does not interrupt processing, however if the machine reboots for some reason, processing restarts where it left off. The server provides status updates and enables job management via the web-based dashboard (see Figure 3).

Briefly, starting a job via the **`beamrvideo_si`** command initiates a single instance of Beamr Video that processes a single file from start to finish. The Beamr Video instance accepts one job at a time, and the processing queue is managed by an external process like a shell script or watch folder. If processing is interrupted, such as by the terminal window closing or the computer rebooting, all processing is lost. Since the file is managed by a single instance of Beamr Video, it does not appear in the web-based dashboard, and hence cannot be managed there.



126 ACTIVE JOBS | 83% CPU USAGE | 87% MEMORY USAGE | 45.3 GB TOTAL SIZE SAVED | 35% AVG. SAVINGS

AWalk480p.mov	Bitrate	High	48% Savings
Dracula720p.mov	Bitrate	High	33% Savings
Dracula480p.mov	Bitrate	High	40% Savings
GetOnUp720p.mov	Bitrate	High	41% Savings

Status: Success
Job ID: 10872639
Input File: /media/psf/home/desktop/demo/getonup720p.mov
Output File: /media/psf/home/desktop/demo/getonup720p_mini.mov
Input: Size 64.1 MB
Output Size: 39.1 MB
Quality: High
Optimize: Bitrate
Profile: Preserve
Level: Preserve
Preserve Bluray: False
Bframes: Auto
B as a reference: True

AWalk720p.mp4	Bitrate	High	48% Savings
DumbandDumber480p.mov	Bitrate	High	24% Savings

Figure 3. *Beamr Video dashboard, available only with server-based operation*

Beamr Video and Adaptive Streaming Groups

Adaptive streaming delivers the optimal viewing experience to viewers connecting over a range of connection speeds and watching on a range of playback devices. To meet this goal, when encoding for adaptive streaming, a single source file is encoded into multiple files with configurations customized for different playback environments and delivery bandwidths.

Multiple factors are considered when creating these configurations. For example, lower-resolution streams (under 480p) are typically configured to stream smoothly to and play well on mobile devices. Middle-resolution (480p to 540p) streams are often customized for specific window sizes on producer websites, while higher-quality streams (720p and higher) are usually configured to create meaningful qualitative upgrades at higher data rates, and to avoid the frequent stream switching that can occur when file data rates are too close together.

M-GO, the premium digital video on demand service created as a joint venture between DreamWorks Animation and Technicolor, has integrated Beamr Video optimization into its video processing workflow.

Since Beamr Video often dramatically reduces the data rate differences between files in the adaptive group, this has made it possible for M-GO to eliminate some streams from the adaptive group, reducing storage and encoding costs while delivering the same or better experience to the viewer. "Rebuffer events were reduced up to 50%, start time decreased by up to 20%, and as a result of optimization, higher quality layers became available at lower bandwidths, enabling many more of our viewers to watch 1080p and enjoy a Blu-ray like experience," M-GO reported.

Stream reduction can obviously be performed manually, and M-GO has created its own internal algorithms for layer elimination. In addition, after researching adaptive streaming configuration recommendations in industry white papers, and reviewing the published practices of online video platforms and some broadcasters, Beamr has formulated the following layer reduction logic that can be integrated into an automated workflow. A sample script written in Python is available for this purpose.

The logic implemented in the script is as follows:

- The script retains at least one iteration for each resolution, no matter how close the data rates. This ensures that streams created for specific window sizes or devices are preserved.
- The script retains all streams configured with unique frame rates, profiles and levels, for the same reason.
- For streams with identical resolutions, the script starts with the highest data rate stream (after Beamr Video) and eliminates any lower quality stream within 25% of the data rate of that stream. The script then restarts the analysis for each stream that is preserved, again eliminating any lower quality streams within 25% of its data rate. This step eliminates any streams that likely will not represent a significant qualitative difference with the next highest quality stream, and those that could cause excessive stream switching.

Obviously, customers are free to adjust the script as desired.

Automating Beamr Video

After understanding the Beamr Video operating modes, we can discuss how to automate Beamr Video processing and integrate it into an existing workflow.

Create a Script

One alternative is to write a shell script that runs the command line alternatives detailed in the Beamr Video User Manual. The script can be written in any programming language and then triggered whenever new content is available for processing.

Watch Folder Integration

Another alternative is to automate via watch folders, dropping files into the watch folder for Beamr processing, and monitoring the Beamr output folder to process the files further. To provide watch folder functionality, Beamr recommends the external tool `incrond`, which can be installed as any Ubuntu program.

Here is a short FAQ regarding using `incrond` with Beamr Video.

How is a watch folder designated?

To designate a watch folder, place a configuration file in the `/etc/inconf.d/ folder`. For example, create the file `/etc/inconf.d/beamr.conf` and insert this line in the file to execute a `beamrvideo` processing:

```
/mnt/input IN_CREATE beamrvideo-i
$@/$# -o /tmp/$#
```

This designates `/mnt/input` as a watch folder. Whenever a new file is added to `/mnt/input`, `incrond` immediately runs `beamrvideo` on the new file with the specified `-i` and `-o` parameters. In `incrond`, the character `@` is the watch folder name, and the character `#` is the watched (newly uploaded) file name. Accordingly, `$@/$#` is the full path to the new file, and this is the input file parameter to `beamrvideo`.

Which folders can be monitored?

A watch folder can be on a local drive or network share – anywhere that a file path can be specified in the config file. S3 buckets and remote FTPs are not supported by this mechanism.

How are parameters like output folder and naming conventions designated?

Add them at the end of the `beamrvideo` command line in the config file. For example:

```
/mnt/input IN_CREATE beamrvideo-i
$@/$# -o /tmp/$# –quality high –
optimize bitrate –overwrite.
```

The output folder and file name are designated in the `beamrvideo` command line using the `-o` parameter. In this example, `-o /tmp/$#` is specified, meaning that the output folder is `/tmp`, and the output file name is the same as the input file name. If just `-o /tmp/` is specified, then the output folder is `/tmp`, and the output file name is the default (which is the input file name with `_mini` appended at the end before the file extension).

What happens to the source file?

In the above example, nothing happens to the source file; it stays in the input folder. To delete the source file after processing, add an `rm` command at the end, with `&&` (which ensures that the `rm` is executed only if the `beamrvideo` command succeeded).

```
/mnt/input IN_CREATE beamrvideo -i
$@/$# -o /tmp/$# && rm $@/$#
```

- Alternatively, add an `mv` command that moves the input file to another location after it is processed, etc. Anything after `IN_CREATE` in the `incrond` config file is executed like a regular Linux command. A more complex logic can also be added, which moves the input file to a success or failed folder depending on the result of `beamrvideo`.

Can multiple watch folders be created?
Yes, multiple lines can be inserted in the config files with different watch folder names, and each of them executes the same `beamrvideo` command or different commands. They can also be put in separate config files, as long as the config files are in the folder `/etc/iconf.d`.

As an example, using a single config file, creates a watch folder called `bv_high`, which stores files processed in High quality mode, and another watch folder called `bv_best` for files processed in Best quality mode. A config file to enable such functionality looks something like this:

```
/mnt/input_bv_high IN_CREATE
beamrvideo –i $@/$# –o /mnt/output_bv_
high/ –quality high –optimize bitrate
```

```
/mnt/input_bv_best IN_CREATE
beamrvideo –i $@/$# –o /mnt/output_bv_
best/ –quality best –optimize bitrate
```

Where can one get additional help with `incrond`?

Documentation of the tool can be found in http://www.cyberciti.biz/faq/linux-inotify-examples-to-replicate-directories/.

About Beamr

Beamr is the global leader in media optimization solutions, powering some of the world's top web publishers, social networks and media companies. Beamr offers a patent-pending perceptual video optimizer, which reduces the bitrate of H.264 and HEVC streams by up to 50%, preserving their full resolution and quality. By reducing video bitrates, Beamr enables content and service providers to distribute exceptionally high-quality video, with faster downloads and smoother streaming on bandwidth constrained connections.

For more information, visit www.beamr.com