Automatic Resource Management in Silverstack Lab

Managed Resources

Silverstack Lab brings together data management and dailies creation on one computer. Both activities come with competing demands for the performance of the used machine. This paper outlines how Silverstack Lab handles the performance needs of different activities, how the different resources such as GPU, CPU and input/output (IO) play together, and what features for resource management Silverstack Lab provides to the user.

Performance, Resources, and Activities

The performance requirements for laptops and desktop computers in use at today's film projects are relatively high even on smaller projects due to high data volumes and processing needs, but the "power" of the used computer system will always be limited in one way or the other.

The overall performance is always limited by one "slowest" resource in a certain situation. That can be a slow source drive, a graphics card with a not-so-powerful GPU, or too few CPU cores. But this is always relative in the given context: A notebook's CPU can be "slow" for theoretically unlimited IO performance, but it may be not even the bottleneck, when the attached storage devices have limited IO capabilities. In that case a faster CPU could not make the overall process faster, but would lead to a CPU that waits even more for the attached drives. So a "fast" computer system for one task might be not so well suited for another task, because different activities simply require quite different combinations of resource types.

The *offloading activity* creates multiple backups of camera cards, creates hash values for each file and logs everything in a database. The main resources utilized by offloading processes are:

- *Read IO (input/output resources)*: Reading of source files from the camera card via a card reader
- *Write IO:* Writing the read data to (multiple) storage devices
- *Read IO*: Verifying the correctness of all data on all destinations
- *CPU:* Creating and verifying hash values

The *creation of dailies* is a combination of two activities, namely transcoding and rendering. The *transcoding* consists of the decoding of the input format and the encoding to the output format.

The main resources utilized by the transcoding processes are:

- *Read IO:* Reading of source files from storage
- *CPU*: Decoding frames from the source format
- *CPU*: Encoding of frames to target format(s)
- *Write IO:* Writing of created files to storage

An example of "decoding" would be the extraction of images from a ProRes QuickTime file, which involves reading the file from the file system, decompressing the ProRes data, and converting the resulting image data to an RGB image.

The amount of Read IO needed is heavily depending on the source format. For example the ARRIRAW format has high Read IO demands, because the files are uncompressed – which leads to relatively large files that on the other hand do not need too much CPU for decoding. CPU usage is depending also on the destination format(s), for example creating an H.264 files is more "CPU-expensive" than creating a DNxHD file of the same resolution.

The *rendering* comprises all steps between decoding and encoding, which can include resizing, applying a look (e.g. via a 3D LUT), conversion between color spaces, and drawing overlays and burn-ins. The main resource used in Silverstack for that is:

- *GPU*: Processing the images

The GPU requirements are influenced by the number of effects/ processing steps used, but mainly depend on the resolution. Creating 4K deliverables simply requires the processing of four to five times the number of pixels than creating 1080p material.

The resources needed by the *playback* activity are quite similar to resources needed by the creation of dailies. As the images are displayed on the monitor or HD-SDI out instead of being written to an output format, the only thing that is missing compared to dailies creation is basically the encoding part.

Overlapping Resource Needs

When several activities should be performed simultaneously, the resource needs obviously do overlap. That could be for example simultaneous reading and writing to the same volume, or the competing requirements for high CPU performance.

A lot of storage media devices don't have the same read or write performance when exclusively read/written when compared to being read or written simultaneously. For example a SSD RAID may have a 50% reduced write speed if another process reads from that device at the same time. This can be even worse for spinning drives, when the read/write head might move a lot for competing IO.

When dailies creation and offloading is happening at the same time, the overlapping resource needs are mainly IO to/from the same volume and CPU utilization. When looking at dailies creation and playback, the overlap is even more obvious: It's the Read IO, the GPU, and the CPU requirements that are competing. Usually while transcoding and rendering is happening for dailies creation, basically no playback is possible.

Need for Management

The requirement of managing competing activities with limited resources can be compared to a chef creating eight different meals with just two pots and one burner. To get the meals out of the kitchen, the chef will either need to constantly switch back and forth between pots and heating, or simply finish all meals one after another sequentially – which will let some of the customers be waiting for a very long time.

For computer systems there are different strategies available for managing resources. The “unmanaged” approach hands the problem of competing resources over to the operating system (OS). The OS can only assure that the available resources are used in a way that the outcome is still correct, but cannot ensure performance-optimized scheduling. So for example parallel IO may reduce the overall sum of IO way below the maximum possibility of a single, undisturbed IO (especially on spinning drives), and managing too many processes with high load is additional work for the OS as well and will require a multitude of RAM than running just one process at a time.

At any time, processes without built-in resource management will usually act after the pirates’ rule of resource management: “Take what you can, give nothing back”. To optimize that, the user either has to schedule the start of different tasks manually to provide exclusive resources, or pause processes (if possible at all) to free resources temporarily for other, momentarily higher prioritized activities.

Cooperative Resource Management

Silverstack Lab (but also Silverstack and Silverstack XT) performs a lot of resource management – even besides the dailies creation process. So for example during offloading of camera cards the number of parallel offloads and transfers is limited to maximize throughput by queueing them in a job list. Even within such an offload job the copy and verification tasks for each file are scheduled in a configurable way. In Silverstack Lab you also find a lot of capabilities for dailies creation. On that account even more management is in place under the hood:

Transcoding and rendering is scheduled in jobs:
Silverstack Lab takes care that dailies creation jobs are automatically started sequentially one after another without user interaction (e.g. creating dailies for multiple reels). This maximizes overall transcoding performance by avoiding competing resource needs between multiple transcoding and rendering tasks (e.g. CPU and GPU).

Transcoding is paused during offloading:
Offloading and backup in order to free camera cards as early as possible has a high priority on a film set. So transcoding is paused automatically when an offload starts and resumes immediately after the offload has finished.

Transcoding is paused during playback:
As soon as the user plays clips in Silverstack's player, all transcoding processes are paused. After a few seconds without playback the transcoding automatically resumes again.

The operator can create jobs and tasks within Silverstack without worrying when to start, stop or pause them:

- The scheduling guarantees that only so many jobs run at a time that make sense on a given system without the need for manual planning of the best job start time (for power users who want to tune their system even further, that can be configured in the preferences).
- The automatic prioritization of offload and playback over transcoding allows a transparent use of Silverstack (e.g. simply hit playback even while a transcoding job is running with high IO/CPU/GPU load) without the need of searching for and shutting down running tasks to free performance resources.

The benefit of the automated resource management is two fold: The performance of the overall system is optimized not only for laboratory setups where e.g. just one transcoding job is running at a time, but for real world scenarios, where parallel tasks and changing priorities create constant need for management of resources. Dealing with the complexity of different resource types and needs of different tasks is taken off the users shoulder and resolved in a optimized way for any given situation. The minimized need for user-involved resource management tasks leads to increased focus on the important things and the automated scheduling runs and finishes prepared work following after a performance-optimized schedule in an organized manner.