11th International Conference on Open Repositories
Trinity College, Dublin, Ireland
13th - 16th June, 2016

Using a GPU to Accelerate Digital Preservation Workflows

Alex Garnett
Data Curation & Digital Preservation Librarian
Simon Fraser University
garnett@sfu.ca

Misty De Meo
Software Developer / Systems Analyst
Artefactual Systems
mdemeo@artefactual.com

Session Type (select one)

Presentation

Abstract

Many current digital preservation and repository platforms are shipped with several open source tools for transforming or extracting data from various ingested file formats. Most of the CPU load of these systems is consumed by these tools, as they will by design continuously peg a CPU until an assigned job is completed. Many of these execute relatively quickly, but some, notably video encoding – typically using *ffmpeg* – and optical character recognition – typically using *tesseract* – do not. Fortunately, both ffmpeg and tesseract can be compiled with new functionality which takes advantage of *General-purpose computing on graphics processing units* (GPGPU), using a GPU (commonly referred to as a video card) to complete jobs in substantially less time and with less CPU load. In this paper/presentation, we present the results of a case study in using GPU acceleration to accelerate video encoding in the context of digital preservation and repository platforms, provide examples for using this functionality at a typical local site, and discuss the implications of supporting and utilizing a GPGPU pipeline on commodity hardware in the GLAM sector.

Conference Themes

Supporting Open Scholarship, Open Data, and Open Science
Repositories of high volume and/or complex data and collections
Managing Research Data, Software, and Workflows

Keywords

GPU Acceleration, OCR, ffmpeg, Archivematica, Optimization, Video

11th International Conference on Open Repositories
Trinity College, Dublin, Ireland
13th - 16th June, 2016

Audience

Repository Developers, Sysadmins, IT Staff/Managers, or people who run a lot of slow ffmpeg and/or OCR jobs

Background

This submission presents open source solutions for improving scalability in repositories of high volume, specifically when processing a large number of (or simply large) videos or OCR jobs. This is relevant to system administrators and other staff trying to balance the CPU load of virtual machines, as well as digital preservation librarians, archivists, repository managers, and developers who just want faster transcoding times.

Presentation content

Digital preservation and repository platforms are challenging systems to scale. While these systems perform large numbers of tasks, a few discrete categories consume disproportionate resources compared to the rest of the system. This lopsided usage of key resources such as CPU power and disk I/O can make it very difficult to determine the best way to provision hardware.

In this paper, we examine the possibility of using GPU (e.g., video card) resources to substitute for CPU resources in several key tasks via the use of GPGPU (*General-purpose computing on graphics processing units*), and provide benchmarks to analyze both the total time taken and the system resources consumed by both traditional CPU-based variants and GPGPU-based variants of several programs. By doing so, we aim to determine whether the total processing time used can be reduced, and whether some CPU resources can be diverted to the GPU in order to free up CPU for other tasks.

The tools examined in this paper are the FFmpeg video encoder, and the Tesseract optical character recognition (OCR) engine. These were chosen for the following reasons:

1) Both are commonly used in digital preservation systems and repositories;
2) Both use disproportionate amounts of resources and frequently serve as bottlenecks in systems which employ them;
3) Both are available in GPGPU-adapted versions.

For our test, we used an Nvidia Titan X GPU[1] paired with a four-core Intel i5-2500k CPU (each core of which is comparable to one core of most Intel server CPUs since 2011), on a computer with 16GB of memory and a 7200RPM spinning hard drive connected via SATA6. With the possible exception of IO performance, this represents a theoretically optimal environment. Our test file was a 720p H.264 / AC3 Matroska video at the film-standard 23.98 frames per second with a duration of slightly more than one hour, with the following output from *ffprobe:*

```
  Duration: 01:01:35.20, start: 0.000000, bitrate: 3385 kb/s

    Stream #0:0: Audio: ac3, 48000 Hz, 5.1(side), fltp, 384 kb/s
(default)

    Stream #0:1(eng): Video: h264 (High), yuv420p, 1280x720, SAR
1:1 DAR 16:9, 23.98 fps, 23.98 tbr, 1k tbn, 47.95 tbc
```

When running ffmpeg with `-vcodec h264` and `-acodec libfdk_aac` and no other options specified (i.e., using the default quality settings for both codecs), encoding purely on the CPU using all four cores, ffmpeg reported an average speed of 63 frames per second – slightly less than 3x the speed of watching the video in real time. The job completed in about 24 minutes. When running ffmpeg with `-vcodec nvenc` and otherwise identical options, using the GPU-based NVENC encoder, ffmpeg reported an average speed of 577 frames per second – almost ten times the speed of the CPU encoder – and successfully re-encoded an hour of video in under three minutes.

The GPU reported 100% utilization during this test, indicating that no other system component was bottlenecking its processing power, which suggests that traditional spinning disks may be fast enough to keep up with current GPUs when the system is otherwise idle. The CPU reported roughly 75% utilization (equivalent to 100% of three of its four cores), which was not insignificant; this was probably due to both the realtime *decoding* of the input file and the re-encoding of the *audio* stream (as only video encoding is handled by the GPU), which are both normally a very small percentage of a typical video encoding job, but are able to run much faster than normal in order to "keep up" with the video encoding being performed on the GPU. Somewhat unintuitively, using a less powerful GPU would probably result in lower CPU

1 For reference, the Titan X is the single most powerful consumer GPU available as of January 2016, retailing at around $1000 USD; however, more value can be found in the middle of the spectrum, such as the GTX 970, which retails around $330 USD but provides roughly 2/3 of the power of the Titan X.

utilization, because the decoding and audio re-encoding would not need to run as quickly, although this would of course slow down the job. Encoding a raw input video without an audio stream would likely drop CPU utilization to near-zero, but this has not been tested.

Similar to ffmpeg's use case for video encoding, the OCR library tesseract can be compiled with GPGPU support by following the provided documentation, after which it can be run on the GPU by setting the environment variable `TESSERACT_OPENCL_DEVICE=1`. Precise improvements are more difficult to measure because OCR jobs are generally not as arbitrarily large as video files – a 2000x2000 PNG of a scanned newspaper was successfully analyzed in about three seconds on the four-core CPU, and about one second on the GPU, though this is obviously an imprecise estimate. Still, these preliminary results are promising in comparison to anecdotes about days-long OCR tasks.

The implementation of normalization or extraction "micro-services" – individual jobs which utilize various CLI applications, typically on a per-filetype basis – is an area of active development within the digital preservation community. Various solutions have been employed in current open source applications such as Archivematica and Islandora, using queuing and messaging frameworks like Gearman and Apache Camel to make these external application calls more programmatic, more stable, and more distributed. This makes the idea of having a dedicated GPU machine configured for running ffmpeg and tesseract jobs much more palatable than it might be otherwise. At present, a simple solution can be configured in Archivematica's dashboard, without requiring any modifications to the codebase, replacing a local ffmpeg call with a *ssh -t* script to run the same call on a remote server using the NVENC encoder like so:

Default: `ffmpeg -i filename.avi -vcodec h264 -acodec libfdk_aac outfile.mkv`

Remote: `scp filename.avi username@remote.server:~/.; ssh -t username@remote.server "ffmpeg -i filename.avi -vcodec nvenc -acodec libfdk_aac outfile.mkv && rm filename.avi"; scp username@remote.server:~/outfile.mkv .`

The network copy operational would add some additional delays to the job, but should be relatively negligible on modern gigabit (or better) networks.

In addition to greatly speeding up the execution time of most ffmpeg or tesseract jobs, there is also a plausible advantage to lowering the overall CPU usage of digital preservation and repository platforms. The computational load of these systems is currently very bursty and unpredictable, leading to difficult conversations with (and later, headaches for) IT staff or systems administrators who are provisioning the virtual machines which GLAM server software will run on; right now it is entirely likely that a given server will not tax its virtual CPU at all beyond serving a web frontend for weeks at a time before suddenly being pegged at 100% utilization for days to handle a large batch ingest. If system administrators have some mechanism in place to prevent individual VMs from slowing down other infrastructure, it is likely that this 100% will actually represent much less than 100% of a physical CPU core (or cores), further slowing down the job. Moving these ffmpeg and tesseract jobs off of the existing stack towards GPU commodity hardware would thus have substantial benefits.

The most significant, and most obvious limitation of this work is that trying to run even a single GPU-capable server within current university or library network infrastructure is generally not trivial, particularly if most of your systems are virtualized and management responsibility at your institution is highly centralized. Justifying both the additional hardware cost and the additional support responsibility thus becomes doubly difficult. However, integration of these GPU boxes into virtualized stacks is not unprecedented; Amazon Web Services, for example, currently offers GPU-attached EC2 instances.

Another limitation is that GPGPU code paths can be more brittle than functionally identical pure-CPU implementations due to developers' unfamiliarity with OpenCL APIs, among other things. For example, H.264 video *decoding* is often performed on the CPU rather than the GPU in many end user applications by default because the CPU-based algorithms will generally be more tolerant of errors in the source file than a GPU decoder would be – this is particularly salient for file-level preservation workflows which require a high degree of confidence in being able to produce good output from questionable input. Fortunately, this is not an issue in ffmpeg's implementation, which exclusively uses the CPU for *decoding* an input file, and only uses the NVENC GPU functionality for *encoding* it to the H.264 output format. However, there is currently an analogous bug in tesseract's GPGPU functionality, which as of this writing produces corrupted output when using GPU acceleration on any input images whose dimensions are not perfectly square. This can be worked around and will very likely soon be fixed, but it does reinforce how immature some GPGPU implementations may be.

Conclusion

Although GPGPU implementations are still a relatively niche use case for digital preservation, the support recently added to two of the most commonly-used, computationally intensive open source libraries may provide a compelling use case for some institutions who are running a significant number of OCR or video encoding jobs. They can provide a ten-fold increase in efficiency on commodity hardware, are relatively easy to configure within current open repository platforms, and make you feel more technically sophisticated than you probably are.

Some of the equipment used to perform this research was generously provided by Nvidia. ffmpeg's GPU encoder was developed by Nvidia and will only function on Nvidia hardware, whereas tesseract uses the generic OpenCL libraries and will theoretically work with any GPU.

References

Soua, M., Kachouri, R., & Akil, M. (2014). GPU parallel implementation of the new hybrid binarization based on Kmeans method (HBK). Journal of Real-Time Image Processing, 1–15. http://doi.org/10.1007/s11554-014-0458-2.

Stensland, H. K., Wilhelmsen, M. A., Gaddam, V. R., Mortensen, A., Langseth, R., Griwodz, C., & Halvorsen, P. (2015). Using a Commodity Hardware Video Encoder for Interactive Applications: International Journal of Multimedia Data Engineering and Management, 6(3), 17–31. http://doi.org/10.4018/ijmdem.2015070102.

Wilhelmsen, M.A., Stensland, H.K., Gaddam, V.R., Halvorsen, P., & Griwodz, C. (2014). Performance and Application of the NVIDIA NVENC H.264 Encoder. GPU Technology Conference. http://on-demand.gputechconf.com/gtc/2014/poster/pdf/P4188_real-time_panorama_video_NVENC.pdf.