**Video Transcoding using MapReduce**

Tom Stordy-Allison

Supervisor: David Budgen

**1. Problem**

The conversion of raw video into different encodings is a highly CPU intensive operation. As video resolution continues to increase, it is becoming harder and harder to convert video into various different encodings and qualities for display and transmission in real-time.

This project will investigate the use of ‘MapReduce’ [DG04], a parallel processing paradigm popularised by Google several years ago, for use of video conversion. This literature survey will discuss MapReduce, ‘The Cloud’ and Distributed Video Encoding and issues surrounding it in more detail, and survey the current state of research in these areas specific to our problem.

**2. Definitions**

|  |  |
| --- | --- |
| Transcoding | “Transcoding is the process of converting a compressed video in a given format, into another compressed video stream.” [Fur08, p951] |

**3. Themes**

**3.1 MapReduce**

“MapReduce is a programming model and an associated implementation for processing and generating large data sets. [In MapReduce] Users specify a map function that processes a key/value pair [of data] to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key” [DG04].



Figure 1:

MapReduce diagram, taken from "MapReduce: Simplified Data Processing on Large Clusters" by Jeffrey Dean and Sanjay Ghemawat [DG04].

The main advantage of formulating a problem like this is that it allows for automatic parallelisation, making it ideal for problems that require either high throughput IO and/or high CPU usage. Each of the Map functions can be processed in a distributed manner, and each of the key-value pairs can be stored appropriately in a distributed file system such that they are available to the processor running the Map on local high-speed storage. The reduce phase can run concurrently as each of the Maps completes, producing the output as soon as possible [Whi09].

In Chen and Schlosser [SS08], processing is achieved through a MapReduce implementation which estimates geographic information (‘where is this scene?’) given a specific image, by leveraging a data set of around 6 million GPS-tagged images and using scene matching to find the most similar image through a reduce-less program.

One of the first public demonstrations of the power of MapReduce in a real world scenario when in 2007, was the New York Times archive conversion, where the TIFF images of the public domain scanned articles from 1851 to 1922 were converted to PDF format [NYT07]. 11 million articles were converted and glued together in less than 24 hours of processing using around 100 nodes of Amazon Web Services’ EC2 instances using the MapReduce model.

Fewer examples are available for video processing, one of them being HP Labs’ VideoToon implementation[[1]](#footnote-1), where a service is implemented to “cartoonize” videos.

There are many, many different implementations of the idea – Hadoop is the open source favourite, with the best support and features. “[Apache] Hadoop Map/Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner”. Hadoop implements the MapReduce model, and supports several languages to directly interface with its API (namely Java and C++, among others) and also the ability to run complied programs and send data in and out of the standard IO streams. Hadoop is available to run wherever Linux does, and several pre-built distributions exist for its use (Yahoo Hadoop[[2]](#footnote-2) and Cloudera[[3]](#footnote-3)).

Several applications have been built to run directly on top of MapReduce in Hadoop, two prominent examples are Hive[[4]](#footnote-4), a data warehouse infrastructure with a SQL-like query system, and Pig [GNC+09], “platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs” [[5]](#footnote-5). Both of these extend MapReduce and its power to a wider audience.

On-going research exists looking to improve areas of Hadoop, for example on distribution and load balancing [ZKJ+08]. This makes the use of the MapReduce model in Hadoop an excellent way of taking advantage of these improvements as they occur.

In the context of video transcoding, Pereira et al [PABE10] propose a ‘Split and Merge’ generalisation of the MapReduce model, and compare the use of this against several single encoders to show the performance improvements that are available. They make a strong case for using this type of model for video encoding in the ‘cloud’. Garcia, Kalva and Furht [GKF10] use MapReduce to convert video from a DVD, in real time, using a user requested quality setting, for live streaming to their device.

**3.2. The ‘Cloud’ and Utility Computing**

‘Cloud Computing’ doesn’t yet have a widely accepted definition. Many forms of computing that we now use can be called ‘cloud’ by some and not by others [AFG+09, Hil09].

We want to take advantage of the availability of distributed computing in an efficient way, through the use of ‘utility computing’, “a service model of computing where computational resources (such as CPU time, storage or network bandwidth) are sold as on-demand and metered resources, like public utilities” [Hil09]. We can obtain resources we want from a remote company, and use it in a distributed manner to get the results we want quicker, being charged the same for 10 computers for one hour as we would one computer for 10 hours. We also don’t have infrastructure in an idle state that we have paid for – efficiently using only the resources we need [PABE10].

Amazon Elastic Compute Cloud (EC2)[[6]](#footnote-6) and Rackspace Cloud Servers[[7]](#footnote-7), both of which provide virtualized servers in the cloud, can be used to create a cluster of machines that the MapReduce model can be used on, using Hadoop. Amazon Elastic MapReduce[[8]](#footnote-8) offers such a service that is specific to the MapReduce problem, allowing the cluster to be setup for us and the jobs submitted almost immediately.

Several commercial solutions for video encoding in ‘the cloud’ have come about – Zencoder[[9]](#footnote-9), Panda[[10]](#footnote-10), Encoding.com[[11]](#footnote-11), aviberry[[12]](#footnote-12) to name a few. Zencoder and Panda both use Amazon EC2, encoding.com uses both Rackspace Cloud Servers and Amazon EC2, and all 4 offer up the ability for the user to submit a job via a website or API and have it encoded on one of many of their machines.

However, all of these seem to use a single machine to single encoding job relationship, rather than spreading out the load where possible between machines. Zencoder and Encoding.com support the use of multiple outputs – using several machines – one for each quality setting.

Using the MapReduce model and inexpensive ‘utility computing’ platforms for video encoding may only really be available for research and for initially web submitted content for now – upload links to cloud are not good enough yet for this to be the only solution [PABE10]. In most cases it takes longer to transmit the video to the cloud than to encoding it on one machine. In time this problem may cease to exist as upload bandwidth availability increases. For encoding jobs that require a diverse set of outputs, and where the input video is used over and over again, the single upload may however be worthwhile.

**3.3 Distributed/Parallel Video Encoding**

Many video encoding parallelization improvements are done at the encoding level, for specific codecs, and sometimes even specific processor architectures [AAL95, FCOG05, HFWG09] – and are designed to work with multiple processors in the same machine, or at the very least in a high-speed cluster. These improvements work well – but what we want is something that can scale to problems that are much larger, and do this using machines that are much less expensive to run.

If we can formulate the problem in such a way that adding hardware is easy, then we can scale-out for little cost. The MapReduce model gives us this and the tools to do this easily in both private clusters, and now also ‘public’ clusters in the cloud.

Distributing the task among machines by splitting the input into ‘chunks’ of work [PABE10] allows for standard decoders/encoders to be used, whilst improving the time taken for the encode. Using non-parallel decoders/encoders in our implementation, that have had much larger amounts of development time, allows us to take advantage of the optimizations, container format support and codec support they offer – e.g. ffmepg/libavcodec[[13]](#footnote-13).

When splitting the input, one of the key problem areas is temporal compression [PABE10]. Many of the codecs that we want to transcode to and from use this type of compression to remove redundant data. Rather than only performing compression on each frame (essentially image compression, usually based on JPEG) the codec uses references to previous and future frames to make up the image. This takes advantage of the small comparative differences between frames due to the nature of their use. When needed or at a set interval, the encoder places a ‘key-frame’ in the stream, a frame that does not reference any other and holds all of the required information to make up the image. Other frames can then reference this to make up themselves [Rich02, Rich03].

The exact format of the temporal compression needs to be understood by any distributed video encoding system, so that it does not split the video at a point that would leave it unable to reference data to make up the images.

Hughes and Walkerdine [HW05] propose a lightweight peer-to-peer system, for the encoding of video using ‘regular PCs’ on a network. Their solution shows the improvements that can be gained through distributing the encode job across multiple computers. They show that as the number of nodes increases, the time taken reduces to a point where it reaches a limit for that file size where extra nodes no longer decrease the time taken.

Compressor[[14]](#footnote-14) is a commercial solution from Apple Inc, for transcoding and supports distributed encoding – but, it can only run on Mac OS X and the details behind its implementation are somewhat unknown. It uses Quicktime[[15]](#footnote-15) for its transcoding, limiting its container and codec support a little (although all the major codecs are covered). It uses Apple Qmaster, a proprietary solution to actually distribute the jobs between machines.

x264farm is an open source distributed encoder that “utilizes x264 in a distributed environment to improve encoding speed. It is designed mainly to be portable and relatively easy to use” [Wil06]. x264farm can only encode video into H.264, and does not support any container formats, leaving it up to the user to actually put the video in its final format. It does not support audio either, and has not been actively developed since 2006. ELDER[[16]](#footnote-16) (parallEL encoDER) is another distributed encoder that whilst being technically closed source, was developed on the doom9.org[[17]](#footnote-17) forums by several members as a solution to low encoding speeds, and given away free as an executable. It again has not been developed actively since 2006, but the binaries are still available for Windows.

**4. Project Direction and Solution**

This project will use Apache Hadoop as the system for implementing the MapReduce model, as it is has the largest community and best set of features.

ffmpeg will be used for the actual reading and writing of the video and audio – giving us a common interface to manipulate the data with, and good support for many codec and container formats.

The system will be tested on both cloud clusters and private clusters. In the ‘cloud’, the intention is to us Amazon Elastic MapReduce – as it offers a solution tailored specifically to our needs, running Hadoop. Private clusters will be set up using virtual machines, making it easy to use machines that don’t run Hadoop or Linux for testing.

The intention is to create a system that is capable of transcoding from AVI/MJPEG-MP3 and AVI/DivX-MP3 to MP4/H.264-AAC and MPEG2/H.264-AAC. The performance of the clusters as file size grows, and as the number of nodes is increased will be assessed, with the optional extension of also assessing the trade off between quality and speed through a user study that will focus on subjective video quality.

*See “Project Specification Document for ‘MapReduce based Audio and Video Transcoding’” for more specifics on the specification.*

**References**

|  |  |
| --- | --- |
| AAL95 | Akramullah, S.M. and Ahmad, I. and Liou, M.L., 1995. A data-parallel approach for real-time MPEG-2 video encoding. *Journal of parallel and distributed computing,* 30(2), pp 129-146. |
| AFG+09 | Armbrust, M. and Fox, A. and Griffith, R. and Joseph, A.D. and Katz, R.H. and Konwinski, A. and Lee, G. and Patterson, D.A. and Rabkin, A. and Stoica, I. and others, 2009. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28.* |
| DG04 | Dean, J. and Ghemawat, S, 2004. MapReduce: Simplified data processing on large clusters. *Communications of the ACM,* 51(1), pp 107-113. |
| FCOG05 | Fernaindez-Escribano, G. and Cuenca, P. and Orozco-Barbosa, L. and Garrido, A. Computational Complexity Reduction of Intra-Frame Prediction in MPEG-2/H. 264 Video Transcoders. *2005 IEEE International Conference on Multimedia and Expo July 6-9, 2005, Amsterdam, The Netherlands*, pp 707-710. |
| Fur08 | B. Furht, 2008, *Encyclopaedia of Multimedia*, 2nd ed. Springer*.*  ISBN: 0387747249 |
| GKF10 | Garcia, A. and Kalva, H. and Furht, B. A study of transcoding on cloud environments for video content delivery, *Proceedings of the 2010 ACM multimedia workshop on Mobile cloud media computing, October 29-29, 2010, Firenze, Italy*, pp13-18. |
| GNC+09 | Gates, A.F. and Natkovich, O. and Chopra, S. et al, 2009. Building a high-level dataflow system on top of Map-Reduce: the Pig experience. *Proceedings of the VLDB Endowment*, 2(2), pp 1414-1425. |
| HFWG09 | He, X. and Fang, X. and Wang, C. and Goto, S. Parallel HD encoding on CELL,  *Proceedings of Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium, Kos, Greece,* pp *1065-1068.* |
| Hil09 | D. Hilley, 2009. Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings, *CERCS, Georgia Institute of Technology, Tech. Rep. GIT-CERCS-09-13.* |
| HW05 | D. Hughes, J. Walkerdine, Distributed video encoding over a peer-to-peer network, in: *Proceedings of PREP 2005, Lancaster, UK, 30th March–1st April, 2005.* |
| NYT07 | New York Times, November 1st, 2007. *Self-service, Prorated Super Computing Fun!*, [online] Available at: [http://open.blogs.nytimes.com/2007/11/01/self-service- prorated-super-computing-fun/](http://open.blogs.nytimes.com/2007/11/01/self-service-%20prorated-super-computing-fun/) [last accessed: 21/06/2001] |
| PABE10 | Pereira, R & et al, An Architecture for Distributed High Performance Video Processing in the Cloud, *Proceedings of Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference, July 2010, Miami, Florida, USA.,* pp 482-289. |
| Rich02 | Iain E. G. Richardson, 2002. *Video codec design: developing image and video compression systems,* Wiley and Sons.  ISBN: 978-0-471-48553-7 |
| Rich03 | Iain E. G. Richardson, 2003, *H.264 and MPEG-4 video compression: video coding for next-generation multimedia*, Wiley and Sons.  ISBN: 9780470848371 |
| SS08 | Shimin Chen, Steven W. Schlosser. Map-Reduce Meets Wider Varieties of Applications. *Intel Labs Pittsburgh Tech Report, IRP-TR-08-05, May, 2008.* |
| Whi09 | T. White, 2009. *Hadoop: The Definitive Guide*, O’Reilly Media, Inc.,  ISBN: 978-0-596-52197-4 |
| Wil06 | Wilson, R. 2006. *x264farm*. [online]  Avaliable at: <http://omion.dyndns.org/x264farm/x264farm.html>  [last acessed 22/06/2011] |
| ZKJ+08 | Zaharia, M. and Konwinski, A. and Joseph, A.D. and Katz, R. and Stoica, I.  Improving MapReduce performance in heterogeneous environments,  *Proceedings of the 8th USENIX conference on Operating systems design and implementation, San Diego, CA, USA 2008.* pp 29-42. |

1. <http://www.hpl.hp.com/open_innovation/cloud_collaboration/cloud_demo_transcript.html> [last accessed: 19/06/2011] [↑](#footnote-ref-1)
2. <http://developer.yahoo.com/hadoop/distribution/> [last accessed: 18/06/2011] [↑](#footnote-ref-2)
3. <http://www.cloudera.com/> [last accessed: 18/06/2011] [↑](#footnote-ref-3)
4. <http://hive.apache.org/> [last accessed: 18/06/2011] [↑](#footnote-ref-4)
5. <http://pig.apache.org/> [last accessed: 19/06/2011] [↑](#footnote-ref-5)
6. <http://aws.amazon.com/ec2/> [last accessed: 18/06/2011] [↑](#footnote-ref-6)
7. <http://www.rackspace.com/cloud/cloud_hosting_products/servers/> [last accessed: 19/06/2011] [↑](#footnote-ref-7)
8. <http://aws.amazon.com/elasticmapreduce/> [last accessed: 18/06/2011] [↑](#footnote-ref-8)
9. <http://zencoder.com/> [last accessed: 18/06/2011] [↑](#footnote-ref-9)
10. <http://www.pandastream.com/> [last accessed: 18/06/2011] [↑](#footnote-ref-10)
11. <http://www.encoding.com/> [last accessed: 18/06/2011] [↑](#footnote-ref-11)
12. <http://www.aviberry.com/> [last accessed: 18/06/2011] [↑](#footnote-ref-12)
13. <http://www.ffmpeg.org/general.html#SEC6> [last accessed: 22/06/2011] [↑](#footnote-ref-13)
14. <http://www.apple.com/finalcutpro/compressor/> [last accessed: 22/06/2011] [↑](#footnote-ref-14)
15. <http://www.apple.com/quicktime/> [last accessed: 22/06/2011] [↑](#footnote-ref-15)
16. <http://www.funknmary.de/bergdichter/projekte/index.php?page=ELDER> [last accessed: 22/06/2011, page last changed: 22/12/2006] [↑](#footnote-ref-16)
17. <http://forum.doom9.org/> [last accessed: 23/06/2011] [↑](#footnote-ref-17)