Scalable Distributed Image Transcoding using Python-WorkQueue

Jeffrey Westphal and Peter Bui □ Department of Computer Science □ University of Wisconsin – Eau Claire

Abstract
Transcoding large amounts of digital media from one format to another is a common data intensive workflow. We present a scalable image transcoding system based on Python-WorkQueue that significantly reduces the amount of time required to convert images from one format to another by mapping transcoding tasks across a distributed pool of remote workers. We test our system using a Condor cluster and varying amounts of files and number of workers. Our results show that we are able to achieve speed increases up until a certain limit.

Evaluation
The above chart shows the speedup gained for 10 1MB images. The x-axis shows the number of workers while the y-axis shows the speedup as the time to convert given one worker divided by the time for N-workers. Given a small numbers of files, adding more workers minimally increases speedup.

System Architecture
After mapping the conversion to each image, the WorkQueue master converts the conversion command to a WorkQueue task. The WorkQueue master then schedules each task to run on a remote worker on a Condor cluster.

Evaluation
The above chart shows that for medium sized files, strong speedup can be gained by continuing to add workers.

Our system uses a self-extracting executable of ImageMagick Convert and a set of images as input. Using a map abstraction, our system creates a command to apply a conversion to each image that will be applied in parallel on many remote workers.

Solution
Map Abstraction

WorkQueue

Strong speedup is gained for medium sized files even if given a very large number of files.

For very large files, adding workers minimally increases speedup after a small number of workers have been added.

As reflected in the graphs, for small sized files, adding more workers seems to continually increase speedup. For larger sized files, adding workers will slow the operation after a certain number are added. Sixteen workers appears to be the near optimal number of workers for sets of large files. This is most likely due to a network bottleneck at the master.

Future Work
Add more workers to test how far speedup continues to scale.
YouTube-like video transcoding service.
Expand and generalize existing system to use any application, rather than just Convert.

Publications
Midwest Instruction & Computing Symposium 2013

We thank the Office of Research and Sponsored Programs for supporting this research, and Learning & Technology Services for printing this poster.