Computational Scalability of Large Size Image Dissemination

Rob Kooper*[a], Peter Bajcsy[a]
[a]National Center for Super Computing Applications
University of Illinois, 1205 W. Clark St., Urbana, IL 61801

ABSTRACT

We have investigated the computational scalability of image pyramid building needed for dissemination of very large image data. The sources of large images include high resolution microscopes and telescopes, remote sensing and airborne imaging, and high resolution scanners. The term 'large' is understood from a user perspective which means either larger than a display size or larger than a memory/disk to hold the image data. The application drivers for our work are digitization projects such as the Lincoln Papers project (each image scan is about 100-150MB or about 5000x8000 pixels with the total number to be around 200,000) and the UIUC library scanning project for historical maps from 17th and 18th century (smaller number but larger images). The goal of our work is understand computational scalability of the web-based dissemination using image pyramids for these large image scans, as well as the preservation aspects of the data. We report our computational benchmarks for (a) building image pyramids to be disseminated using the Microsoft Seadragon library, (b) a computation execution approach using hyper-threading to generate image pyramids and to utilize the underlying hardware, and (c) an image pyramid preservation approach using various hard drive configurations of Redundant Array of Independent Disks (RAID) drives for input/output operations. The benchmarks are obtained with a map (334.61 MB, JPEG format, 17591x15014 pixels). The discussion combines the speed and preservation objectives.

Keywords: gigapixel images, image pyramids

1. INTRODUCTION

There have been several efforts underway to digitize many old manuscripts since they are too fragile to be accessed by everyone. Many libraries are scanning historical manuscripts and maps using high resolution scanners that generate very large images. These digital scans serve the purpose of preserving the information as well the purpose of making the manuscripts available to a large research community. As the historical materials are digitized, problems arise related to access and on-demand updates of the repository of digital scans.

We have been involved in two projects that deal with large size and large volume of digital image scans. The first project involves image scans of Lincoln's papers and legal documents (~5000x8000 pixels/each, ~150MB/each, ~25,000 images, TIFF file format). The second project focuses on the digitization of historical maps from the 17th and 18th century at the University of Illinois library (~20000x15000 pixels/each, ~350MB/each, compressed jpeg file format). The challenge is to automatically update the repositories of image scans and make them easily accessible for scholars and general public.

While these two scanning efforts were the main drivers of our computational scalability study, there are many other applications where image pyramids have to be built and computational scalability has to be understood. Applications processing airborne and satellite imagery or sky survey imagery would typically lead to Giga and Tera pixel images (see Costa Rica imagery in [1] and Large Synoptic Survey Telescope (LSST) generating 15 terabytes of daily raw data and approximately 150 petabytes of raw data by program end [2]). These very large size images have to be disseminated in a form of an image pyramid similar to Google Map. With the increasing resolution of commercial off-the-shelf cameras and their decreasing cost per pixel (e.g., Kodak EasyShare C195 14.0-Megapixel Digital Camera costs $100 as 10/10/2010), panoramic images or images of cylindrical 3D objects imaged on a turn table generate very large images using these regular cameras and have to be converted into an image pyramid in order to view them efficiently. Example of images generated on a turn table can be found in imaging historical artifacts such as cylinder seals from Mesopotamia [3].

Our work is motivated by the lack of understanding about computational scalability of building image pyramids needed for dissemination of very large image data and the tradeoffs between preservation driven replication of data and an

Parallel Processing for Imaging Applications, edited by John D. Owens, I-Jong Lin, Yu-Jin Zhang, Giordano B. Beretta,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7872, 78720N · © 2011 SPIE-IS&T · CCC code: 0277-786X/10/$18
doi: 10.1117/12.872834

overhead incurred due to input/output operations of image pyramids to disks. Pyramid representation allows delivering image tiles to a client at the requested spatial resolutions and image location instead of sending the entire image.

We address the problem by collecting and interpreting computational benchmarks for (a) building image pyramids to be disseminated using the Microsoft Seadragon library, (b) a computation execution approach using hyper-threading to generate image pyramids and to utilize the underlying hardware, and (c) an image pyramid backup approach using various hard drive configurations of Redundant Array of Independent Disks (RAID) drives for input/output operations.

Image pyramids are built by cutting the original image into 256x256 tiles, saving the tiles to a disk, sub-sampling the original image and repeating the 'cut' and 'save' steps till 1x1 pixel tile is formed. The process of building an image pyramid is highly parallelizable since each tile at one level is independent of any of the other tiles at the same level. We take an advantage of this when spawning threads on multi-core machines. The data transfers to and from disk can be optimized with respect to speed and storage reliability for low-cost disk drive components such as the RAID. We experimented with RAID 0 (improved performance and additional storage but no redundancy or fault tolerance) and RAID 1 (provided fault tolerance from disk errors and failure of all but decreased performance and storage to one of the drives).

2. BACKGROUND ON IMAGE PYRAMIDS, HYPER-THREADING AND DISK ARRAY CONFIGURATIONS

2.1 Background on building Image Preview and Image Pyramid

The concept of image pyramids has been introduced in 1983 by Burt and Adelson [6], and later patented as a pyramid processor method [7]. The initial Laplacian pyramid has been advanced by the introduction of Gaussian pyramid [8], Wavelets and quadrature mirror filters (QMFs) [10], and steerable pyramids [9]. There are clear advantages in building image pyramids of very large images allowing a user to download only a view of his interest but not the entire image.

We have been working with the Seadragon library created by Microsoft [4] to allow users to smoothly and interactively view high resolution images using image pyramids. A user can zoom in and out of the image without having to pay the overhead of downloading the full image first. The following illustration shows how an image pyramid works. The image itself is available in full resolution at the bottom of the pyramid, and lower resolution versions down to 1x1 pixels are stored alongside the full resolution image. The images at each level of the pyramid are stored in NxN pixel tiles, indicated by the white lines in Figure 1 (where N is often 256).

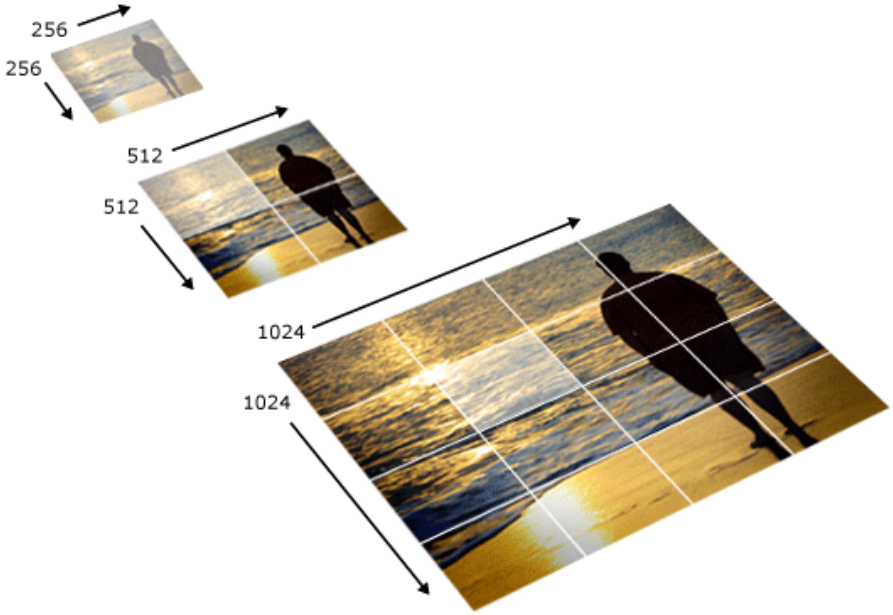

Figure 1: Illustration of an image pyramid and its tiles.

We have internally created the software as part of a larger system to create these image pyramids once a user has uploaded their image. The system will take the uploaded image and create two thumbnails (100x100 and 800x600 pixels) and then the image pyramid. The speed of the preview and image pyramid computations are of our interest since a user desires to view the large images using the Seadragon technology immediately after the upload of the data to the server.

Creating the tiles of each level of the pyramid lends itself for simple parallelism. One the larger image is loaded into memory, each tile can be cut out and store to disk by separate threads. Because of the potential large number of tiles that are created we are interested in how fast we can write the tiles to disk since that is where the bottleneck of our system is. Our approach to computational scalability of image pyramid building is to investigate the use of hyper-threading for computation execution and various hard drive configurations such as RAID drives (Redundant Array of Independent Disks) for input/output operations in order to achieve the fastest execution of the computation of interest.

2.2 Background on Hyper-threading

Hyper-threading works by duplicating certain sections of the processor—those that store the architectural state—but not duplicating the main execution resources. This allows a hyper-threading processor to appear as two "logical" processors to the host operating system, allowing the operating system to schedule two threads or processes simultaneously. When execution resources would not be used by the current task in a processor without hyper-threading, and especially when the processor is stalled (the processor has to wait idle for example due to a cache miss, branch inaccurate prediction or data dependency or anything that prevents the processor from executing the next command), a hyper-threading equipped processor can use those execution resources to execute another scheduled task.

2.3 Background on hard drive configurations

RAID (Redundant Array of Independent Disks) [5] is a technology to create high levels of storage reliability and space using low-cost disk drive components (see Figure 2). RAID 0, provides improved performance and additional storage but no redundancy or fault tolerance. Because there is no redundancy, this level is not actually a Redundant Array of Independent Disks, i.e. not true RAID. However, because of the similarities to RAID (especially the need for a controller to distribute data across multiple disks), simple stripe sets are normally referred to as RAID 0. Any disk failure destroys the array, which has greater consequences with more disks in the array (at a minimum, catastrophic data loss is twice as severe compared to single drives without RAID). A single disk failure destroys the entire array because when data is written to a RAID 0 drive, the data is broken into fragments. The number of fragments is dictated by the number of disks in the array. The fragments are written to their respective disks simultaneously on the same sector. This allows smaller sections of the entire chunk of data to be read off the drive in parallel, increasing bandwidth. RAID 0 does not implement error checking so any error is unrecoverable. The more disks in the RAID array, the higher the bandwidth and the greater risk of data loss.

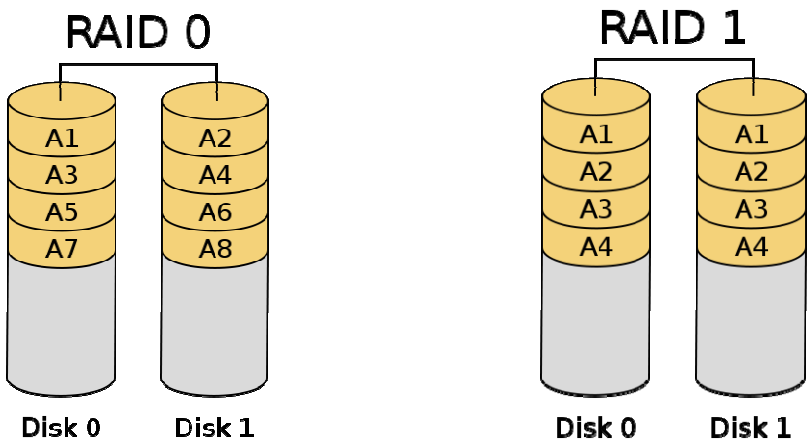

Figure 2. Illustration of RAID 0 (left) and RAID 1 (right) from [5].

RAID 1, provides fault tolerance from disk errors and failure of all but one of the drives. Increased read performance occurs when using a multi-threaded operating system that supports split seeks, as well as a very small performance

reduction when writing. Array continues to operate so long as at least one drive is functioning. Using RAID 1 with a separate controller for each disk is sometimes called *duplexing*.

Finally, the non-raid setup is exactly like the RAID 1 setup but we have no overhead of making sure the data is copied correctly to the second drive. We have created a setup like the RAID 0 with no redundancy of the data which in the case of a failure could result in total data loss.

3. EXPERIMENTAL RESULTS

3.1 Experimental Setup

We have used as test system a standard desktop PC equipped with an I7 processor running at 2.8 Ghz, 8 cores when using hyper threading, 16 GB of memory and two 1TB drives. We tested a non-raid setup where we only used a single drive, a raid 1 configuration with both drives used resulting in 1TB drive space, and a raid 0 configuration with both drives resulting in 2TB of drive space.

We installed Ubuntu 9.10 64-bit server on the system. The application that creates the image pyramid is written in java and was given 10GB of memory to work. We used the openJDK version 1.6 to run the application. When switching the drive configuration the system was completely wiped and reinstalled using the same process and same software.

3.2 Test Data

We have performed benchmarking of computational scalability with the map of North America created by Arrowsmith in 1814 (see Figure 3). The image that was given to us is a 334.61 MB JPEG. The image measures 17591x15014 pixels (WxH).

Figure 3: The image scan of 1814 map of North America by Arrowsmith is a 334.61 MB JPEG; 17,591 x 15,014 pixels.

3.3 Experimental Results

For the test file (Arrowsmith 1814 map), we had to create a pyramid with 15 levels, 4071 tiles at the highest resolution and a total of about 5500 tiles total. Each tile was created on a separate thread. We used the java ExecutorService to limit the number of parallel threads. We had a minimum of one thread (in which case we did not use the ExecutorService) and a maximum of 20 threads.

The figures that show the results of the experiment have on the vertical axis the time it took to finish that particular task in hours : minutes : seconds and on the horizontal axis the number of threads that are used in parallel. The markers indicate the number of threads used (1, 2, 4, 5, 7, 10, 15, 20).

Figure 4 shows the results for creating the two thumbnails. Based on the number of threads allocated to the system the two thumbnails are either created in parallel (number of threads > 1) or sequential (number of threads = 1). We can see in Figure 4 that no matter what drive configuration is used it is more optimal to use more than one thread to create the thumbnails. For example, when using two threads to create the thumbnails, there is a 30% decrease in computation time.

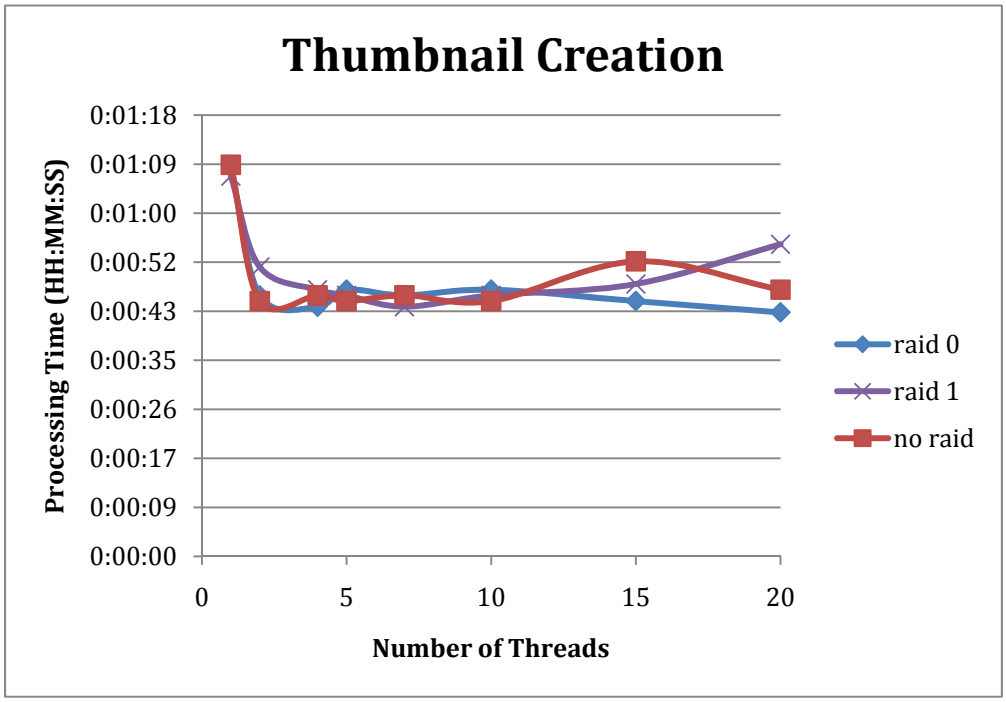

Figure 4. Processing time needed for preview (thumbnail) creation for three different drive configurations and eight different numbers of values of threads. The input image is in JPEG file format (334.61 MB) and of image size 17591x15014 pixels (WxH).

Figure 5 shows the results for creating the pyramid. Each level of the pyramid is created sequentially and not until all tiles in a level have been created will the software continue to the next level of the pyramid. In the case of a single thread, all tiles will be created sequentially and the ExecutorService is not used. If more than one thread is used the system will create all the threads first and then execute them using the ExecutorService.

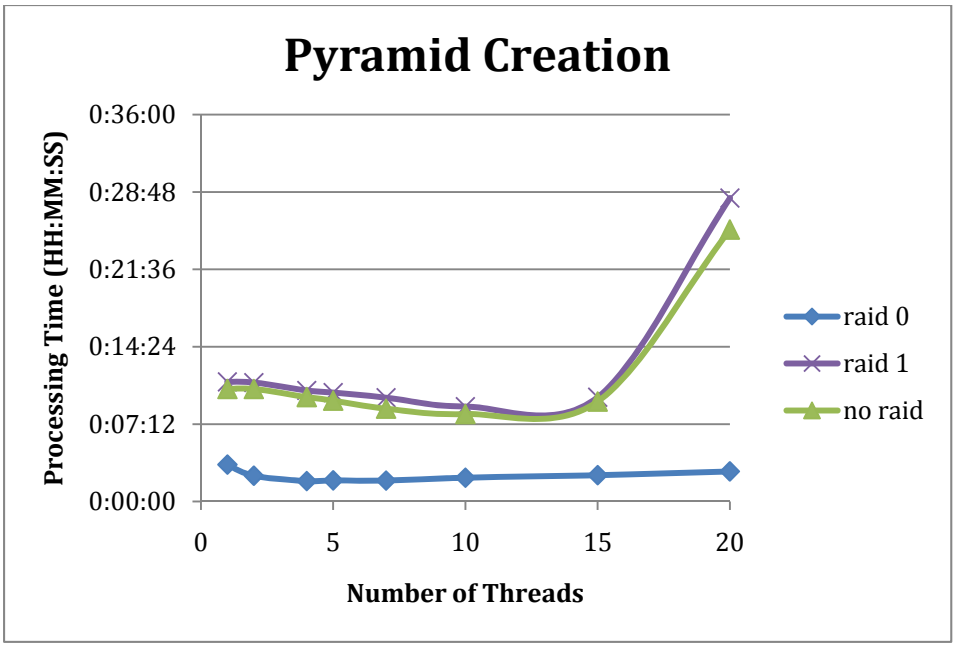

Figure 5. Processing time needed for pyramid creation for three different drive configurations and eight different numbers of values of threads. The input image is in JPEG file format (334.61 MB) and of image size 17591x15014 pixels (WxH).

Figure 5 illustrates that the processing time does not differ too much when using NO RAID or RAID 1 configuration. The penalty paid in this particular experiment for using the RAID 1 configuration is only 8% when compared to a NO RAID configuration. However, we gain the redundancy of the data, i.e., if one of the drives fails we do not lose any data. When we compare the RAID 0 configuration to either the RAID 1 or NO RAID configuration we can see a dramatic improvement (about 4.5 times faster). This clearly indicates that the system is Input/Output limited.

4. CONCLUSIONS AND FUTURE WORK

We explored the computational benchmarks for building image pyramids using hyper-threading with the number of threads used (1, 2, 4, 5, 7, 10, 15, 20) and three hard drive configurations of Redundant Array of Independent Disks (RAID) drives for input/output operations (no RAID, RAID 0, and RAID 1). By analyzing all results, RAID 0 is clearly superior in terms of execution time compared to the other configurations. In all cases we could observe the I/O transfer becoming a bottleneck of the pyramid processing at a certain number of threads (4 threads in a case of RAID 0 and 8 threads in a case of RAID 1 and no RAID setups). From the preservation perspective, there are many applications requiring not only fast execution of the computation but also high confidence in preserving the data in the event of hardware failure. When considering both objectives (execution time and data preservation), a failure in a RAID 0 configuration will result in complete data loss. A RAID 1 configuration can withstand the failure of one drive without losing the data. The NO RAID configuration is not significantly faster when compared to RAID 1. If data preservation is critical then a RAID 1 configuration is definitely a feasible solution while taking into account the difference in costs.

Future analyses of this topic would include (a) the difference between software and hardware RAID configurations, (b) the difference between write and read execution time, and (c) how many Input/Output connections can each RAID disk configuration handle. These considerations are critical to understand when priorities on execution time shift from the pyramid creation (focus on writing) to pyramid viewing (focus on reading) or require to accommodate both (upload images for pyramid creation while viewing other existing image pyramids).

ACKNOWLEDGEMENTS

This research has been funded through the National Science Foundation Cooperative Agreement NSF OCI 05-25308 and Cooperative Support Agreement NSF OCI 05-04064 by the National Archives and Records Administration (NARA)

REFERENCES

[1] Kooper R. and Bajcsy P., "Parallel Image Stitching of Airborne Imagery," IS&T/SPIE Electronic Imaging 2011, January 23-27 (proceedings are in press).
[2] Jewett B. "NCSA: Sky Yields Big Data," HPC Wire, July 6, 2010; http://www.hpcwire.com/offthewire/NCSA-Sky-Yields-Big-Data-97863504.html
[3] Lundberg M. J., "Cylinder Seals and the West Semitic Research Project,", http://www.usc.edu/dept/LAS/wsrp/educational_site/ancient_texts/cylinder_seals.shtml
[4] Microsoft Seadragon library for web-based delivery of image pyramids, http://seadragon.com
[5] Standard RAID levels, Wikipedia; http://en.wikipedia.org/wiki/Standard_RAID_levels
[6] BURT P. J, and ADELSON E. H. "The Laplacian Pyramid as a Compact Image Code," IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-3l, NO. 4, APRIL 1983
[7] Burt P.J. "Pyramid processor for building large are high resolution images by parts,"- US Patent 4,797,942, 1989
[8] Adelson E. H. , Anderson |C. H., Bergen J. R. , Burt P. J., and Ogden J. M., "Pyramid methods in image processing," RCA Engineer • 29-6 • Nov/Dec 1984
[9] E. P. Simoncelli and W. T. Freeman, *The steerable pyramid: a flexible architecture for multi-scale derivative computation*, 2nd Annual IEEE International Conference on Image Processing, Washington, DC. October, 1995.
[10] Akansu A.N, Haddad R. A., and Caglar H., "The Binomial QMF-Wavelet Transform for Multiresolution Signal Decomposition, " IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 41. NO. I . JANUARY 1993