JuxtaView – a Tool for Interactive Visualization of Large Imagery on Scalable Tiled Displays

Naveen K. Krishnaprasad*, Venkatram Vishwanath, Shalini Venkataraman, Arun G. Rao, Luc Renambot, Jason Leigh, Andrew E. Johnson
* naveen@evl.uic.edu
Electronic Visualization Laboratory **(EVL)**
University of Illinois at Chicago

Brian Davis
Earth Resource Observation Systems (**EROS**) Data Center
US Geological Survey (**USGS)**

Abstract

JuxtaView is a cluster-based application for viewing ultra-high-resolution images on scalable tiled displays. We present in JuxtaView, a new parallel computing and distributed memory approach for out-of-core montage visualization, using LambdaRAM, a software-based network-level cache system. The *ultimate* goal of JuxtaView is to enable a user to interactively roam through potentially terabytes of distributed, spatially referenced image data such as those from electron microscopes, satellites and aerial photographs. In working towards this goal, we describe our first prototype implemented over a local area network, where the image is distributed using LambdaRAM, on the memory of all nodes of a PC cluster driving a tiled display wall. Aggressive pre-fetching schemes employed by LambdaRAM help to reduce latency involved in remote memory access. We compare LambdaRAM with a more traditional memory-mapped file approach for out-of-core visualization.

1. Introduction

As the precision of data acquisition increases, scientists find it impractical to examine any significant portion of the data at an effective level of detail. For instance, scientists at the Earth Resource Observation Systems (EROS) Data Center (EDC) at US Geological Survey (USGS) [1] acquire on a regular basis, terabytes of data from satellites and aerial photography. The EDC is acquiring high-resolution color orthoimagery for 133 most populated metropolitan areas of the United States at a resolution of about 1/3 meter. As an essential element of The National Map [2], the need for up-to-date imagery is critical for Homeland Security, and Emergency Response. The imagery can be used as a base layer for updating or deriving additional geographic information, such as transportation networks, hydrographic features, elevation and land cover. While examining any large metropolitan area like Chicago, one should be able to look at an overview of a large area, interactively pan and zoom at detailed resolution and also be able to teleport to different areas within a terabyte dataset. In such scenarios it is intractable to examine any significant portion of the large image area on a single desktop and higher display resolution becomes paramount to understanding the data.

JuxtaView is an application designed to view extremely large image montages on high-resolution, scalable tiled display walls like the GeoWall2 [3] built at the Electronic Visualization Laboratory (EVL). The GeoWall2 consists of a tiled array of LCD panels, driven by a cluster of PC's, with a high-end graphics cards such as Nvidia's Quadro FX series, large disk space, dual processor CPUs and Gigabit Ethernet networking. GeoWall2 is scalable because smaller or even larger versions can be built by adjusting the number of LCD panels and computers. A separate master console is used to show an overview of the image viewed on the tiled display. The master node of the cluster accepts keyboard, mouse and joystick events from the user for interaction with the image on the tiled display. USGS is currently using a GeoWall2, with JuxtaView being deployed and used on a daily basis, for studying high resolution aerial and satellite imagery.

Figure 1: Picture of JuxtaView on the GeoWall2 demonstrated at Supercomputing 2003, Phoenix, Arizona. The image shows satellite data from ocean floor bathymetry of the Earth. Data Courtesy: USGS

High-resolution tiled display walls however solve only part of the problem. For instance, hydrologists at the USGS who study the hurricane effects along the Atlantic seashore would like to be able to interactively roam through 6 inch resolution of LIDAR data of the entire coastline from Chesapeake Bay to the tip of Florida. It is impractical to build displays big enough to examine all of the data at such levels. Instead we need means to interactively roam through a dataset, potentially distributed on different servers.

At EVL, we have developed a computing paradigm called the Optiputer as the primary means for supporting similar large scale visualization applications. The Optiputer [4] is a National Science Foundation funded project to interconnect distributed storage, computing and visualization resources using photonic networks. The main goal of the project is to exploit the trend that network capacity is increasing at a rate far exceeding processor speed, while at the same time plummeting in cost. We have developed aggressive data transfer protocols such as the Reliable Blast UDP [5], as part of the Optiputer project, to help applications take advantage of this available bandwidth.

As an application on the Optiputer, JuxtaView is our work in progress to roam through large image datasets potentially residing on remote data sites interconnected by photonic networks. LambdaRAM [6] is the middleware designed based on our networking expertise designated to help Optiputer applications harness high bandwidth network connectivity and still address latency of long haul networks. LambdaRAM achieves this by managing distributed memory pools for data storage and employing aggressive pre-fetching schemes to cache data locally for applications.

JuxtaView is in the preliminary phase of the Optiputer project. The goal in this phase is to present a scalable visualization model and prototype LambdaRAM as a network level abstraction for large data storage. The main contribution of our work is,

- Presenting a scalable graphics visualization tool for large images on high-resolution tiled displays
- Presenting a novel distributed memory approach for out-of-core visualization of large imagery that outperforms traditional approaches like memory mapped IO
- Designing a network level cache in LambdaRAM, de-coupled from the application, designed to harness high network bandwidth and hide latency through aggressive pre-fetching

We present our work in this phase as a proof of concept for our future Optiputer applications over wide area photonic networks.

2. Related Work

Standard image viewers like Adobe Photoshop [7], Gimp [8] and Imagemagick [9] offer several image manipulation features and handle various file formats. Specialized software like the ArcView (of the ArcGIS suite of tools) [10] provide several data display, query and analysis capabilities for geographically distributed data. Traditional image viewers however are largely restricted by availability of memory on a single machine and the disk access speed, in the case of images that do not fit into memory. Consequently most traditional viewers are designed to effectively handle small images, but not very large images in the order of several gigabytes.

VLiv [11] is a desktop image viewer that can load very large tiff images stored in a tile format by loading only the tiles required for viewing. However an inherent requirement for viewing large images is the ability to look at a significant portion of the image at effective resolution, which calls for more high-resolution display systems like the IBM T221 display [12] or tiled display walls such as the GeoWall2.

[13]and [14] give an overview of building scalable tiled display walls using PC clusters and issues involved in writing scalable software. Commonly used software for tiled display

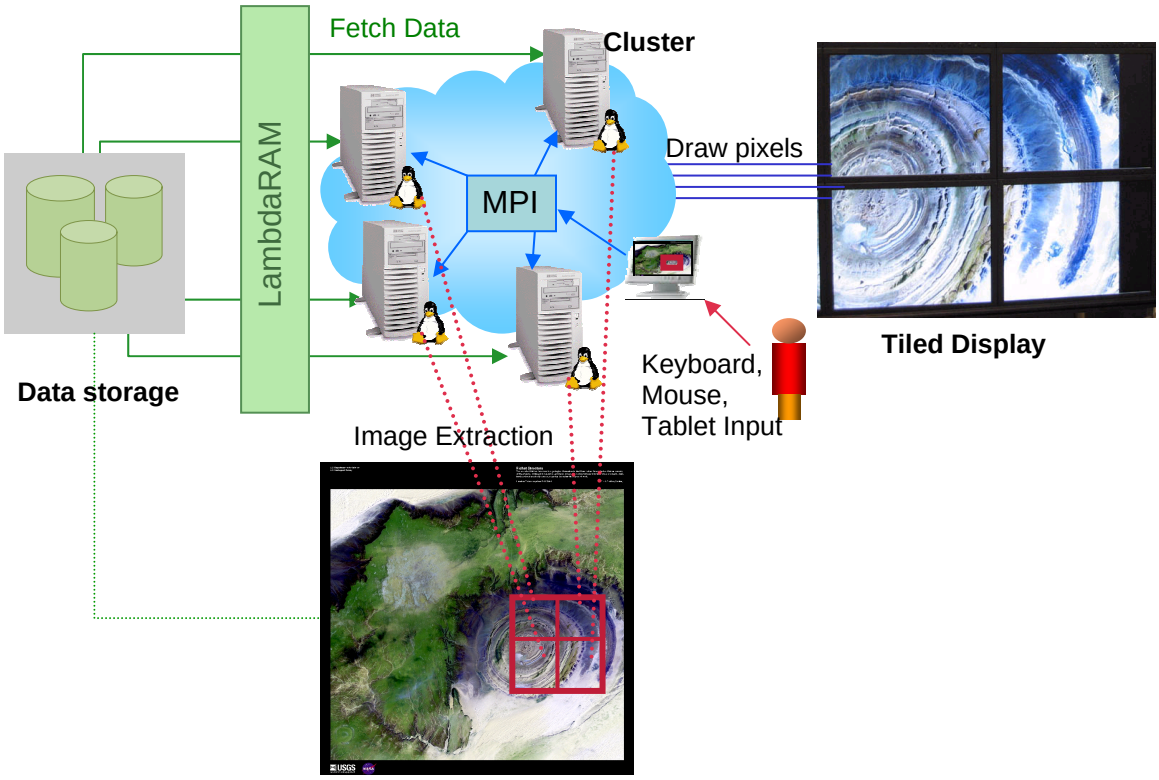

Figure 2: Overview of JuxtaView's design. The figure shows a image stored on disk loaded into LambdaRAM's memory pool distributed on all nodes of a 4 node cluster driving a 2x2 tiled display. A separate master console, which shows a preview of the image, allows users to interact with the image on the tile display. Image courtesy: USGS

environments include VNC [15] and Chromium [16], which provide generic means to distribute frame buffers and polygon rendering on a tiled display, respectively. [17]describes a cluster based image viewer developed at Argonne National Labs, for large format images and list the problems involved in designing one. Argonne's image viewer employs texture mapping through hardware to display images. While taking advantage of hardware accelerated bilinear interpolation, this however currently precludes choosing arbitrary points on the tiled display for zooming, in order to examine specific points of the image in detail. TimV developed at San Diego Supercomputing Center [18], is a cluster based image viewer for tiled displays, built over ImageMagick, MPI and OpenGL libraries. A disadvantage of TimV is its inherently dependence on virtual memory of a single machine for the image [19]. Both Argonne's image viewer and TimV fail to address the issue of viewing out-of-core data on tiled display walls.

JuxtaView addresses this issue by using LambdaRAM to store images over the distributed memory of workstations of a cluster. In future we aim to scale this over wide area networks and harness memory of several clusters of computers connected by high speed photonic networks.

3. System Design

JuxtaView is a parallel application built over MPI [20], OpenGL [21], Glut [22] and SDL [23] libraries. It loads image files stored in a raw RGBA file format, which enables it to easily map the image to a buffer in the application using either a memory mapped (mmap) interface or a distributed shared memory interface such as LambdaRAM. Figure 2 shows the overall system organization in JuxtaView, which is discussed in further detail in the following sub-sections. The image to be loaded on the tiled display is stored on a local disk. A JuxtaView process is started on each node of the cluster driving the tiled display using MPI. At start up, the image is distributed over the memory of all the nodes using LambdaRAM, based on a static two dimensional partition. Each process controlling a

tile is set up to draw a portion of the image on that tile through a configuration file.

The master node of the cluster driving the tiled display accepts user interactions with the image on the tiled display. The commands are broadcast to all the slave processes controlling the individual tiles, which re-compute the image areas corresponding to their tiles. The data corresponding to the image areas is fetched from LambdaRAM's cluster memory pool. Once the pixels to be displayed are computed, the tiles display the pixels in a synchronized fashion.

3.1 Parallel Image Extraction

JuxtaView uses an image order parallel algorithm to extract the image pixels to be drawn on each tile of the tiled display. The pixels to be displayed are computed each time a new operation is performed using a global transformation. Since the pixels of the tiled display are automatically divided equally among all the tiles, the computation is in turn evenly divided among all the nodes.

In order to implement this, we maintain three different contexts in the program; an image context in the order of the dimensions of the image, the tile-display context in tile-display coordinates including borders between display panels, if any, and the local context of each tile, in screen coordinates. A cursor which can be moved across the tiled display enables a user to choose any arbitrary point in the image as reference for zooming in and out of the image. The transformation from the image to the screen coordinates is described as follows:

$$\text{Transformation, } T = \begin{bmatrix} 1 & 0 & Z_x \\ 0 & 1 & Z_y \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} 1 & 0 & -Z_x \\ 0 & 1 & -Z_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & Z_x(1-S_x) \\ 0 & S_y & Z_y(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

If the cursor position is indicated by a point Z (Z_x, Z_y) and the current scale by S (S_x, S_y), the transformation matrix to calculate the pixel coordinates in image space to tile display space is computed, as show below, by translating the cursor position to origin, scaling and translating back to the cursor position.

The pixel indices in the image coordinates, P (P_x , P_y) are computed as a product of the transformation matrix (calculated above) and the tile display coordinate matrix T(T_x , T_y) where they are displayed.

$$P [P_x , P_y, 0] = \begin{bmatrix} S_x & 0 & Z_x(1-S_x) \\ 0 & S_y & Z_y(1-S_y) \\ 0 & 0 & 1 \end{bmatrix} X \begin{bmatrix} T_x \\ T_y \\ 1 \end{bmatrix}$$

At the end of computation, the pixels are drawn on the tiles in a lock step fashion.

3.2 Communication and Synchronization

We use a master-slave MPI model for communication and synchronization. The master node is in charge of user interaction and propagation of the commands via a MPI broadcast channel which is used in a lock step fashion on all the slaves. The slaves are responsible for computing the pixels required for displaying on each tile using a transformation as described above. Each slave process uses a LambdaRAM data client to fetch the portion of the data required that does not reside on local memory.

Our algorithm is designed so that the master is not responsible for calculating each slave's image view. The broadcast messages are very small, merely indicating the operation to be performed on the image and are independent of the image size or the number of tiles. This ensures scalability to larger tiled displays and image sizes.

3.3 User Interaction

Users can interact with the image on the tiled display using a keyboard, mouse or joystick, connected to the master node of the cluster driving the tiled display. Glut and SDL libraries are used for window management and registering event callbacks for keyboard, mouse

and wireless joystick interactions. A preview window is provided on the master console with an overlaid box showing the image area viewed on the tiled display. User operations in JuxtaView include the following:

- panning across the image at various speeds
- scaling with respect to any arbitrary point on the images
- teleporting to any part of the image by clicking on the overview map
- playing an animation movie or slide show, by cycling through a set of images

We have also implemented client-server architecture using the Quanta networking API [24] for third party communication from a laptop or a tablet PC. A server process executing in a separate thread on the master node, accepts multiple TCP connections from third party JuxtaView clients. Users executing these client processes, presumably on laptops or tablet PC's, can request a mini-map of the entire image, zoom in at a particular point and enter text annotations which is routed through the server on the master node to be pasted onto the image on the tiled display. The annotations serve to identify interesting areas in an image.

4. LambdaRAM

LambdaRAM is based on the concept of Network Memory. Prior work in Network Memory (NetRAM) has mainly focused on local area or system area networks because there simply has never been sufficient bandwidth over a wide area network to carry data from memory to memory at rates that are close to memory access rates [25]. The high bandwidth photonic networks interconnecting components on the Optiputer makes NetRAM over wide areas practical. The concept behind NetRAM is to provide a massive pool of physical memory that is distributed over separate computers which are interconnected using high speed photonic networks. For example, while a intra-cluster interconnection network such as Myrinet can have as much as a Gigabit of bandwidth with a latency of a few microseconds, photonic networks used by LambdaRAM will have a bandwidth of 10 Gigabits/s and latencies of approximately 2-5ms; whereas a SCSI disk drive only has 300Mbits of bandwidth with an average seek time of 8-9 ms [6]. Therefore in scenarios where data to be stored is too high for local disk storage, using NetRAM to move data between sites with high bandwidth connectivity, becomes an attractive option. LambdaRAM provides a read-only data cache, which fits most visualization and data mining applications where the original data is rarely modified. Application developers using LambdaRAM can specify the partitioning of data on distributed workstations to maximize locality of reference. The latency involved in long distance photonic networks limited by the speed of light, can be overcome by LambdaRAM through aggressive pre-fetching schemes, providing data to applications hopefully before it is accessed.

4.1 Data Partitioning for JuxtaView

For the local area case of LambdaRAM, the memory pool managed by LambdaRAM on each node is split into two parts – the Lambda Cache, which consists of the portion of the original data assigned to a node and a Local Cache, which consists of data obtained from remote nodes, cached locally for the application. The data stored in the cache is in turn organized in fixed sized blocks of smaller size, as in a paging scheme, to increase the granularity of management.

JuxtaView specifies a two dimensional partitioning for LambdaRAM, maintaining a logical spatial locality in proportion to the ratio of image sub-division on the tiled display. The general partitioning strategy can be sketched as follows:

Image Dimensions = M x N pixels, where M is the number of columns and N is the number of rows

Tile-display resolution = m x n pixels, typically (M x N) >> (m x n), the image being much larger than the tile-display resolution

Tile-display layout = a x b, where a is the number of tiles in a row and b is the number of tiles in a column

Number of nodes in the cluster driving the tile-display= P, where a x b =P, in a simple case where each node drives one tile of the display

Total Cache size of LambdaRAM on each node = X

Lambda Cache = L1 = M/a x N/b, sub-divided into smaller blocks of size y x z

Local Cache, L2 = X – L1, used for caching data locally for applications. This portion is

also used to store data pre-fetched from remote nodes

Thus every node explicitly knows the data partition and fetches data from the remote node on demand, to satisfy a memory request. LambdaRAM's intra-cluster communication is implemented using TCP scatter-gather mechanisms provided by the Quanta networking API, for gathering data distributed on other nodes of the cluster. Over a wide area network, LambdaRAM data transfer module would take advantage of protocols such as Reliable Blast UDP provided by Quanta [5], instead of TCP.

4.2 Cache Management

The Cache Management module in LambdaRAM comprises of the following:

a. Data-Server

The data-server thread services requests from other nodes, for data present in its Lambda Cache. The smallest unit of data transfer is one block of data, which can be specified by the user.

b. Data-Client

The Data-Client is the interface used by applications like JuxtaView to request data. The Data-Client checks if the data requested is contained in either the Lambda Cache or the Local Cache. In the case of a hit, the pointer to the memory requested is returned. In the case of a miss, requests are sent to remote nodes which contain the data blocks. Once the data blocks are received, a pointer to the memory requested is returned back to the application.

c. Cache Replacement

LambdaRAM currently implements a Least Recently Used (LRU) block replacement strategy for garbage collection in the Local Cache area. Each data request made by the application is identified using an Operation Id (OID).

d. Data Pre-fetcher

We have implemented a data pre-fetching scheme based on the locality of reference of the data. When a block of data is accessed, the adjacent blocks surrounding the requested block are also pre-fetched from remote memory. We have experimented with different block sizes in order to determine the optimal balance of high hit ratio and memory usage.

Our current design of LambdaRAM assumes that the image fits into the collective memory of a PC cluster.

5. Results

We conducted experiments to quantify the performance of JuxtaView on a 16 node Intel Xeon 1.8 GHz cluster running a Red Hat Linux 2.4.18-3smp kernel, with 512 MB RAM, 512KB cache and using NVIDIA GeForce4 graphic cards. The dataset used for our experiments was a 1.8 GB image file of dimensions 30,000 x 15, 000 pixels, showing aerial photography of the Chicago downtown area at 1 foot resolution.

We provide comparative results of loading the image in JuxtaView using a mmap API, traditionally used for out-of-core visualization in image viewers such as ImageMagick [19], versus using LambdaRAM to use the collective distributed memory of the cluster. The dataset was replicated on all the individual disks of the nodes in the mmap case, in order to increase disk access speed.

We performed hundred random pan and zoom operations and recorded the latencies for the operations. The latency calculations in our results indicate the time taken by JuxtaView to complete the data access and computation, recorded just before drawing. The value used is the maximum (worst case) latency among all the nodes in the cluster, since that is the synchronization cost for displaying the image on the tiled display. The following were the test cases:

1. *Panning:* The image was panned at full resolution, and the image area requested was the same as the area to be displayed on the screen. The latency of panning reduces as the image areas already visited are accessed again, for example when a user pans right and left immediately.
2. *Zooming:* The zoom operations in this test are performed by choosing any point in the image to teleport to, and zooming in at full resolution and followed by two zoom out operations to examine a larger area of the image. Tele-porting to an arbitrary point in the image to zoom in triggers a random memory accesses into the image, usually resulting in the peak latencies shown in the graphs.
3. *Pre-fetching*: In this case LambdaRAM fetched adjacent blocks surrounding the data requested, so that the subsequent pan or zoom operation would find the data in the local cache. We found pre-fetching to be useful in reducing the latencies further. The test results are tabulated:

In summary, we find that the LambdaRAM version of JuxtaView outperforms the mmap version by a factor of 2 in most cases. The graphs obtained from the tests are plotted below. The graphs show a histogram of operations lying in different ranges of latencies. A normal curve is provided in each graph to show the distribution of numbers around the mean latency.

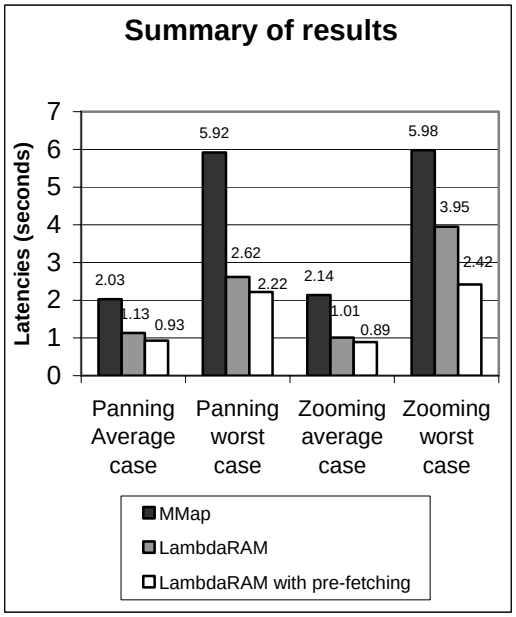

Figure 3: Summary of results

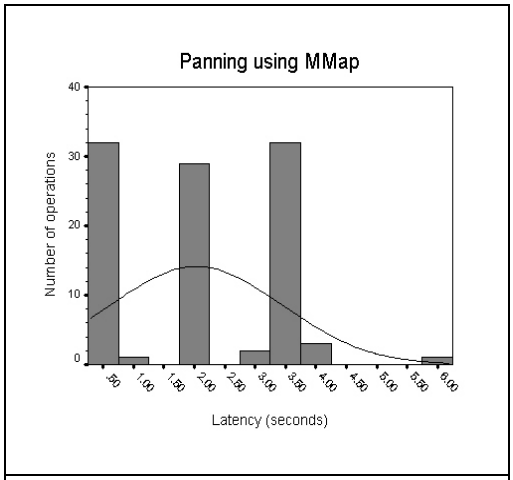

Figure 4: Panning using MMap: Approximately 70% of the latencies lie in the range of 1.8 – 3.8 seconds

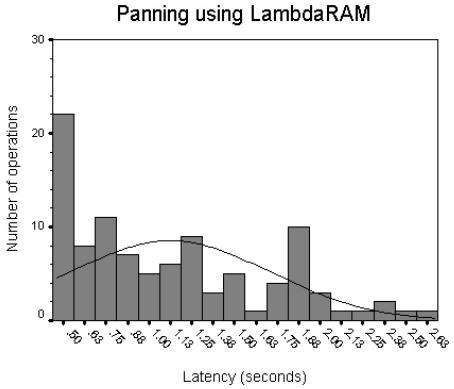

Figure 5: Panning using LambdaRAM: Approximately 70% of the latencies lie in the range of 0.5 – 1.5 seconds

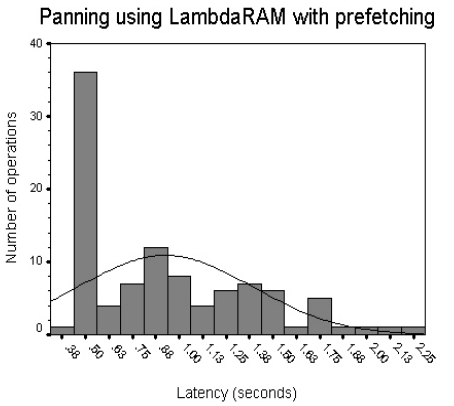

Figure 6: Panning using LambdaRAM with pre-fetching: About 70% of the latencies lie in the range of 0 – 1 second

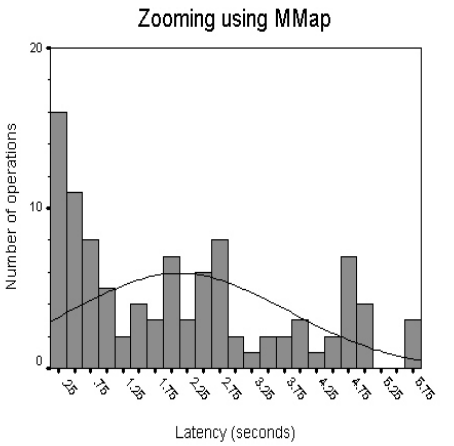

Figure 7: Zooming using mmap: The latencies are more evenly distributed, with 25% lying in the range of 0-0.5 seconds, 25% between 0.5-2 seconds and 30% between 2-4 seconds

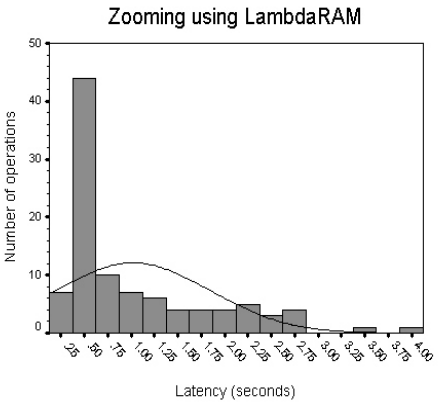

Figure 8: Zooming using LambdaRAM: Approximately 70% of the latencies lie in the range of 0 – 1 second

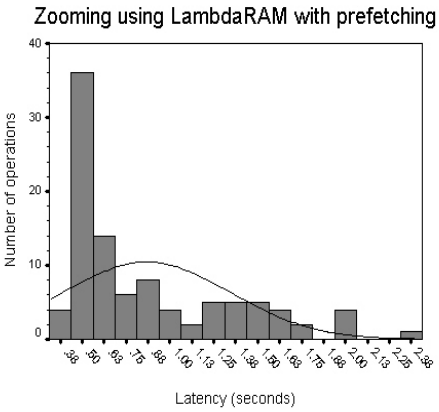

Figure 9: Zooming using LambdaRAM with pre-fetching: Here too, about 70% of the latencies lie in the range of 0 – 1 second

6. Current Applications and Future Work

JuxtaView is being used by US Geological Survey (USGS) for Geosciences' applications involving satellite imagery and aerial photography. The Institute of Geophysics and Planetary Physics at the Scripps Institution of Oceanography [26] employs JuxtaView to analyze IKONOS[27] satellite imagery, aerial photography and panoramic images from the Mars Rover missions. The National Center for Medical Imaging and Research (NCMIR) [28] uses JuxtaView to visualize large cerebral montages obtained from Confocal Microscopy. We are also working with the Joint Oceanographic Institutes [29] in using JuxtaView to visualize images of extremely long core samples extending to several miles, obtained by drilling ocean and lake beds.

We have loaded image sizes up to 15 GB, corresponding to a 65,000 X 55,000 pixel aerial photography image on our 3 node 64 bit Opteron cluster. In future, as we proceed towards a tera-byte scale of data, we need to introduce an additional layer in LambdaRAM to pre-fetch data stored on disk at distributed sites. Moreover, intuitive user interfaces are needed to roam through them, at various resolution levels specified by the user.

LambdaRAM should be able to load portions of a big image dataset, using a configuration provided by the user. The dataset configuration would specify mapping from the image space to a specific image file and an IP address to indicate the location of the file on a remote site. More intelligent data pre-fetching schemes that adapt to the memory access patterns in an image will increase the efficiency of LambdaRAM.

In serving the Geosciences community, we are currently implementing overlaying of multiple geo-referenced images with different transparency levels to merge different types of data from the same geographical location. We have developed a prototype Optiputer test-bed between Chicago and Amsterdam over OMNInet [30], the Optical Metropolitan Network Initiative, currently deployed in Chicago and Evanston, linked to StarLight [31] and a transatlantic high-performance link provided by SURFnet [32], on to NetherLight in Amsterdam [33]. We are currently conducting LambdaRAM experiments using JuxtaView on this test-bed between Chicago and Amsterdam, with a cluster of computers at Amsterdam serving as a data source for images, visualized at EVL in Chicago on our GeoWall2.

7. Conclusion

We have presented JuxtaView, an interactive visualization tool for large imagery on scalable tiled displays. JuxtaView presents a new distributed memory approach for out-of-core visualization of image datasets that do not fit on the memory of a single computer. We have provided results to show that LambdaRAM outperforms memory mapped IO by at least a factor of 2. We have also prototyped LambdaRAM as a network level cache to provide seamless access to distributed data storage. The experience in integrating JuxtaView

and LambdaRAM will help designing other visualization applications over the Optiputer.

8. Acknowledgements

The advanced networking and visualization research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), awards EIA-9802090, EIA-0115809, ANI-9980480, ANI-0229642, ANI-9730202, ANI-0123399, ANI-0129527, EAR-0218918, as well as the NSF Information Technology Research (ITR) cooperative agreement (ANI-0225642) to the University of California SanDiego (UCSD) for "The OptIPuter" and the NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement (ACI-9619019) to the National Computational Science Alliance. EVL also receives funding from the US Department of Energy (DOE) ASCI VIEWS program and by the Office of Naval Research through an award from the Technology Research Education and Commercialization Center (TRECC). In addition, EVL receives funding from the State of Illinois, Microsoft Research, General Motors Research, and Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan.
The USGS Land Remote Sensing Program and the Cooperative Topographic Mapping (CTM) Program fund Information Technology Research at the EROS Data Center, concentrating on Scientific Visualization Investigations.

9. References

[1] The EROS Data Center (EDC), USGS, http://edc.usgs.gov/

[2] The National Map, US Geological Survey http://nationalmap.usgs.gov

[3] The GeoWall 2, http://www.evl.uic.edu/cavern/optiputer/geowall2.html

[4] T.A. DeFanti, J. Leigh, M.D. Brown, D.J. Sandin, O. Yu, C. Zhang, R. Singh, E. He, J. Alimohideen, N.K. Krishnaprasad, R. Grossman, M. Mazzucco, L. Smarr, M. Ellisman, P. Papadopoulos, A. Chien, J. Orcutt, "Teleimmersion and Visualization with the OptIPuter," Proc. of the 12th International Conference on Artificial Reality and Telexistence (ICAT 2002), The University of Tokyo, Japan, December 3-6, 2002, to be published by Ohmsha/IOS Press

[5] Eric He , Jason Leigh , Oliver Yu , Thomas A. DeFanti, Reliable Blast UDP: Predictable High Performance Bulk Data Transfer, Proceedings of the IEEE International Conference on Cluster Computing, p.317, September 23-26, 2002

[6] C. Zhang, J. Leigh, T. A. DeFanti, TeraScope: Distributed Visual Data Mining of Terascale Data Sets over Photonic Networks, .In Future Generation Computer Systems, Elsevier Science Press. 2003

[7] Adobe Photoshop software, http://www.adobe.com/products/photoshop/main.html

[8] Gimp, The GNU Image Manipulation Program, http://www.gimp.org

[9] ImageMagick image manipulation toolkit and API, http://www.imagemagick.org/

[10] ArcView GIS and mapping software and the ArcGIS suite of tools, http://www.esri.com/software/arcgis/arcview/index.html

[11] VLiv desktop image viewer for large images, http://delhoume.frederic.free.fr/vliv.htm

[12] IBM T221 display, http://www.research.ibm.com/deepview/

[13] Hereld, I. R. Judson, and R. L. Stevens, "Introduction to Building Projection-based Tiled Display Systems," IEEE Computer Graphics & Applications, vol.20, pp. 22 - 28, 2000

[14] Y. Chen, H. Chen, D. W. Clark, Z. Liu, G. Wallace, K. Li., "Software environments for cluster-based display systems", First IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2001

[15] Virtual Network Computing, http://www.uk.research.att.com/archive/vnc/

[16] Greg Humphreys, Mike Houston, Ren Ng, Sean Ahern, Randall Frank, Peter Kirchner, and James T. Klosowski, Chromium: A Stream Processing Framework for Interactive Graphics on Clusters of Workstations. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002), pp 693-702, July 2002

[17] Justin Blinns, Michael E. Papka, Rick Stevens, Cluster-based image viewer http://www.fusiongrid.org/research/papers/hpdc02-viewer.pdf

[18] Tiled Image Viewer (TimV), http://graphics.ucsd.edu/~cdonner/timv/

[19] Image Magick FAQ, www.imagemagick.com/www/FAQ.html#C9

[20] MPICH implementation of the Message Passing Interface (MPI), http://www-unix.mcs.anl.gov/mpi/

[21] Open GL cross platform graphics library http://ww.opengl.org

[22] GL Utility Toolkit (GLUT), www.opengl.org/resources/libraries/glut.html

[23] Simple DirectMedia Layer (SDL), http://www.libsdl.org

[24] Eric He, Javid Alimohideen, Josh Eliason, Naveen Krishnaprasad, Jason Leigh, Oliver Yu,

Thomas A. DeFanti, “Quanta: A Toolkit for High Performance Data Delivery over Photonic Networks,” Journal of Future Generation Computer Systems (FGCS), Elsevier Science Press, Volume 19, Issue 6, August 2003, pp. 919-934

[25] K. Li. IVY: A Shared Virtual Memory System for Parallel Computing. In Proceedings of the International Conference on Parallel Processing, 1988

[26] The Scripps Institution of Oceanography, http://www.igpp.ucsd.edu/

[27] IKONOS Satellite imagery, http://www.spaceimaging.com/

[28] The National Center for Medical Imaging and Research (NCMIR), University of California, San Diego, http://ncmir.ucsd.edu/

[29] Joint Oceanographic Institutions, http://www.joiscience.org

[30] http://www.icair.org/omninet

[31] http://www.startap.net/starlight

[32] http://www.surfnet.nl

[33] http://www.uva.nl