Parallel Visualization and Analysis with ParaView on a Cray XT4

John Patchett, *Los Alamos National Laboratory*
James Ahrens, *Los Alamos National Laboratory*
Sean Ahern, *Oak Ridge National Laboratory*
David Pugmire, *Oak Ridge National Laboratory*

Abstract: Scientific data sets produced by modern supercomputers like ORNL's Cray XT 4, Jaguar, can be extremely large, making visualization and analysis more difficult as moving large resultant data to dedicated analysis systems can be prohibitively expensive. We share our continuing work of integrating a parallel visualization system, ParaView, on ORNL's Jaguar system and our efforts to enable extreme scale interactive data visualization and analysis. We will discuss porting challenges and present performance numbers.

Keywords: Cray XT4, Interactive Distance Visualization, ParaView

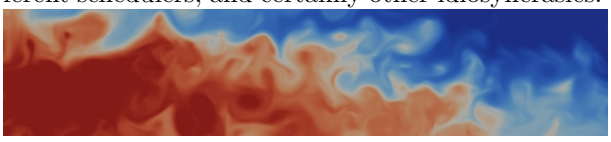
1 Introduction

Jaguar is a Cray XT4 with 7,832 compute nodes each with 4 cores and 8GB of memory, located in Oak Ridge, Tennessee. Jaguar is currently ranked #8 in the Top 500.

Jaguar, and other machines like it, enable increasingly larger data sets. Whether scientists are producing extremely massive data or small data they still have a requirement for analysis. Analysis is often difficult for larger data and it is further complicated when the user is distant to the data. Data sets that are too large to ship and/or analyze on a single desktop require a parallel analysis tool and a high performance computer to run on.

Traditionally, specialized visualization resources capable of interactively handling large-scale visualization operations are co-located with the supercomputers they support. These specialized resources, usually PC clusters with graphics cards, are becoming less useful for a number of reasons. They are expensive to purchase and maintain and as the sizes of compute machines increase, so must the size and expense of these specialized resources. The specialty visualization resources are good for local users with high bandwidth connections into them and they are also necessary for driving other specialty hardware like power walls, caves and raves by providing high frame rates.

For remote users like those at LANL using Jaguar, they are limited to significantly lower frame rates due to the network, so the ability of the specialty hardware to produce such high frame rates does not help much due to Amdahl's law. Additionally, the specialty hardware can be an irritation, as it requires another account on an additional machine that has its own rules of use, possibly different schedulers, and certainly other idiosyncrasies.



Our primary scientific contact, and Jaguar user, for this study was Mathew Maltrud. He uses interactive visualization as part of his standard work flow and we studied the use of interactive visualization via ParaView on Jaguar. Mat uses Jaguar to run the Parallel Ocean Program (POP) [2], which produces more data on Jaguar in a day than can be streamed back to LANL in a reasonable time frame, if at all. He therefore is faced with doing visualization and analysis at a distance. We tailored our usability study to his visualization needs.

We used ParaView [1], an open source, parallel visualization tool to study whether interactive distance visualization on large machines like Jaguar is technically feasible and worth pursuing. The largest barrier to using these machines interactively for visualization is the emphasis on batch scheduling, which

is generally configured to provide very little quality of service to the waiting user.

We were awarded with an NCCS Directors Discretionary Project to port ParaView to Jaguar. This paper will summarize our results to date.

We are contributing to visualization on the Cray XT4 by looking at interactive distance visualization using parallel rendering with standard OpenGL using OSMesa at a distance. We are also presenting ParaView on the Cray XT4.

2 Previous Work

Visualization work had already been done on the XT3 [4] and XT4 [7], but it did not include interactive remote visualization. These two visualization papers were presented at CUG 2008. Moreland et al [4] presented their work porting ParaView to the Cray XT3, they presented numbers for batch processing of a very large data set. Pugmire et al [7] presented their work on porting VisIt, a parallel visualization application, to the Cray XT4. They presented both batch processing and interactive processing on extremely large data sets.

Mesa's offscreen rendering component, OSMesa, allows GL rendering into user allocated memory [6]. OSMesa allows graphics on machines without graphics hardware, machines like Jaguar. Work on OSMesa is a core enabler for our work.

3 ParaView

ParaView is a VTK based, open source, serial and parallel visualization tool. In a parallel mode, it is best run in a client server configuration where a parallel server is started using MPI and a client connects to it over sockets. The client, data server and render servers can all be run on separate machines and then communicate with each other over sockets [8]. Our goal with Jaguar was to run the data and render servers on Jaguar and run the client on a local machine.

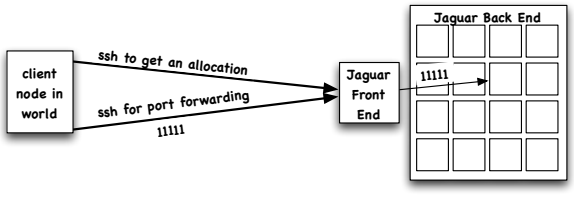
3.1 Ineractive vs. Batch

Supercomputers primarily rely on batch processing for process scheduling. The problems arise when we try and use batch schedulers in ways that weren't necessarily envisioned in their design, such as interactive visualization. The interactivity model for scheduling appears to have been designed for debugging and it is less than adequate for visualization and analysis. It creates too much time between the hypothesis and verification analysis loop. Batch based visualization is an iterative process that involves editing a script, submitting a job, waiting, looking at an image or video, modifying the script and starting the process again. Scientists spend more time interacting with the bureaucracy of the computer scheduler than they do with their data. The visualization process is significantly improved by interactivity and this improvement facilitates scientific discovery.

3.2 Building and Running ParaView

Running ParaView on the Cray XT4 was easier than it was historically for the XT3. Socket support made it very convenient to connect the client to the back-end for interactivity. We desired interactivity over batch processing, so we didn't have to delve into building and linking against a static python. Cross compiling for Jaguar was relatively simple and didn't require hardly anything but cmake settings. We did have to build a native ParaView for the Jaguar front-end server using the make pvHostTools target documented by Moreland et al 2008 [4]. Jaguar has a set of front end nodes that can be used as launching points for ssh port forwarding. The problem is a desktop at LANL is behind several levels of firewalls, and must connect to a specific Cray XT4 back end node at ORNL. This is accomplished using ssh port forwarding.

In order to set up port forwarding the user must find the hostname that will get assigned process 0. This can be found after getting the interactive allocation by running "aprun -n 1 hostname". The user must then start a new ssh tunnel from the LANL desktop to a Jaguar front end that will forward the port to the process 0 back end node: "ssh -L 11111:MPIProc0:11111 Jaguar.ccs.ornl.gov" This will forward any port 11111 loopback connection to the desktop and send it to MPI Process 0 port 11111 through Jaguar.ccs.ornl.gov.



Then the XT4 ParaView server is started: "aprun

-n 16 pvserver –use-offscreen-rendering". A client will get started on the local desktop. The client connects to localhost port 11111 and is forwarded to the remote server where the connection is established.

The interactive startup process looks like this:

1. Connect to Jaguar and request an allocation with qsub.
2. Find hostname of process 0.
3. Start pvserver.
4. Connect to Jaguar forwarding port 11111 to MPI Process 0.
5. Start a local client.
6. Connect client to localhost port 11111.

This process will have to be repeated for each new allocation of nodes. The server and client can be restarted at any time without reconfiguration of the port forwarding.

4 Approach

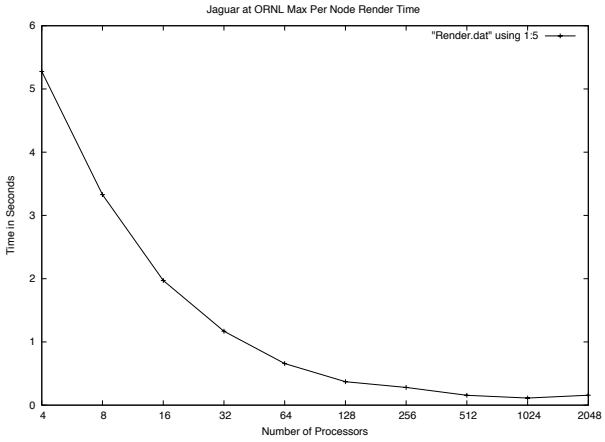
We were driven by the use case of our user, whose models run on a 3600x2400x42 rectilinear grid. He regularly wants to run a visualization pipeline of read, isosurface and render of a single variable data set. This visualization pipeline consists of parallel reader that supplies data to a parallel isosurfacing algorithm. This in turn feeds polygons to a parallel renderer, whose results are composited together to form a single image. The final image is then sent to the client application and displayed. This series of events is run for every image shown on the client, though through last modification short-circuiting in the pipeline, some steps can be skipped, speeding up the process. For instance, the reader does not have to read on the next iteration, unless the input file is changed.

4.1 Reading

Poor read scaling was expected, we still have more work to do optimizing lustre reads. As our tests read a single monolithic file, we altered the stripe count to 32 and stripe size to 1 MB, which offered a good improvement over no striping or a stripe count of 4. Our user generally has a single file for each variable in each time step. For good performance on very large processor counts, in the future, we should alter ParaView's reader to restrict the nodes performing reads and then use the high performance network to balance the data across the process space.

4.2 Rendering

Rendering is the process of turning 3D geometry into a 2D image. Rendering using OSMesa, Mesa's offscreen rendering interface, was used as Jaguar lacks graphics hardware. The numbers reported by Moreland et al. [4] for producing frames of a 2048^3 dataset using ParaView batch processing with OSMesa reassured us that we would be able to get interactive rates in our study. Parallel rendering has good scaling properties as it is trivially parallel. So, given enough processors for a problem size, we expect, the per processor rendering performance should adequate.



4.3 Compositing

Compositing is the process of gathering fractional images, rendered on each node in the parallel job, and assembling them. Jaguar is interconnected using a Cray SeaStar network. Compositing consists of both network and CPU time. ParaView uses Z-depth compositing, so whenever two images overlap and need to be combined, each pixel's z value has to be compared to find which pixel is closest to the viewer. We believe the binary swap algorithm [3] provides a worst case compositing time for an improved compositor, IceT [5], used in ParaView. Binary swap's network consumption asymptotically approaches 2 * render window size for each process as the number of processors grows. This means compositing is a function of render window size rather

than a function of data size.

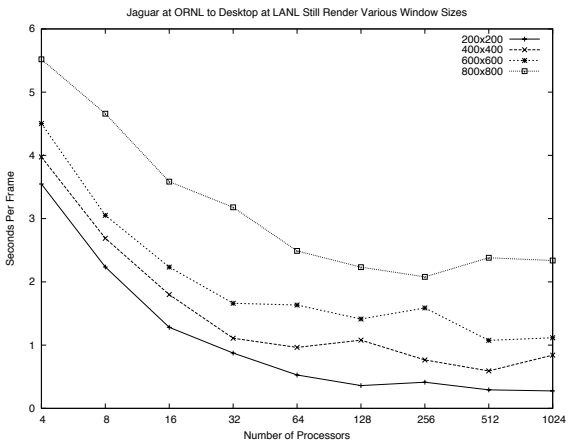
We ran binary swap scaling tests for a 1024x1024 image composed of 8 bit chars to represent RGBA and a float for z, totaling 8 MB/image. These tests made it clear that compositing can be expected to scale, in terms of not hindering interactive rates. This is good, as larger data sets will require more nodes to get rendering performance sufficient for interactive rendering rates. Finally, Results from ParaView show the average composite time plateaus at under .03 seconds per frame for the 400x400 render window.

4.4 Sending

Once images are composited they must be sent to the client. Bandwidth between the user and the supercomputer can help or severely impair an interactive user experience. Bandwidth into LANL is somewhat restricted for many reasons.

Using NetPIPE we found that the unidirectional streaming between LANL and a Jaguar compute node through an ssh tunnel to a back end Jaguar node is between 10 and 11 Mbps. This is about 1.375 MB/s. The ParaView client machine used for testing rarely showed more than 1 MB/s bandwidth. For one of our tests we used a 400x400 Render Window, which using 8 bits each for RGBA is 640 KB. From this, we can then estimate that, at best, we could get a frame from a Jaguar back end node in approximately 1/2 second + the time to create the image on Jaguar.

5 Results and Scaling



5.1 Interactive ParaView

We find that getting nodes interactively happens reasonably quickly for smaller numbers of nodes. We have gotten up to 2048 processors in less than a minute. Though there can be down times while getting interactive nodes, we have usually gotten them pretty quickly. ParaView comes equipped with tools for handling large data and/or data at a distance like LOD and client server communications data compression. ParaView can be used to send data, geometry or imagery. The user, while running the tool, can change these settings easily. If imagery is used, the fps count will be a function of the network bandwidth assuming the supercomputer can keep up.

5.2 Scalability

We ran interactive visualization jobs on as many as 2048 Jaguar processors. Most graphs in this paper show promising performance scaling. We have not yet explored IO and this needs to be studied further so we can make positive changes to allow for end to end scalability. Specifically, until we can read quickly, out of core operations, like time series stored in single files for each step, will not be very interactive. The previous work on the XT3 and XT4 plus our work make us believe scaling for interactive visualization is completely possible on Jaguar.

6 Conclusion

Running ParaView interactively on Jaguar is a positive improvement for both local and remote users. We have shown that it is feasible for our user at LANL to visualize his simulation data created on jaguar. Local users may face fewer obstacles and get improved performance, due to their higher bandwidth to Jaguar. For future work, we would like to see higher priority queues that enable interactive visualization. A more interactive user-friendly qsub that estimates time to allocation would be very useful. Hopefully, our work will lead to more users attempting interactive visualization on Jaguar, resulting in quicker turnaround times for visualization and analysis. As users migrate to using supercomputing platforms for visualization, we expect future research possibilities in this area.

7 References

References

[1] James Ahrens, Berk Geveci, and Charles Law. Paraview: An end user tool for large data visualization. In *Visualization Handbook*. Academic Press, 2005.

[2] J. K. Dukowicz and R. D. Smith. Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *Journal of Geophysical Research*, 99:7991–8014, 1994.

[3] Kwan liu Ma, James S. Painter, and Charles D. Hansen. Parallel volume rendering using binary-swap compositing. *IEEE Computer Graphics and Applications*, 14:59–68, 1994.

[4] Kenneth Moreland, David Rogers, John Greenfield, Berk Geveci, Patrick Marion, Alexander Neundorf, and Kent Eschenberk. Large scale visualization on the cray xt3 using paraview. *CUG 2008*, May 2008.

[5] Kenneth Moreland, Brian Wylie, and Constantine Pavlakos. Sort-last parallel rendering for viewing extremely large data sets on tile displays. In *PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 85–92, Piscataway, NJ, USA, 2001. IEEE Press.

[6] Brian Paul. The mesa 3d graphics library. *http://www.mesa3d.org/*, May 2009.

[7] David Pugmire, Hank Childs, and Sean Ahern. Parallel analysis and visualization on cray compute node linux. *CUG 2008*, May 2008.

[8] Amy Henerson Squillacote. *The ParaView Guide*. Kitware, Inc., 2007.

8 About The Authors

John Patchett has worked at the Advanced Computing Laboratory at the Los Alamos National Lab since 1999 exploring visualization hardware, practices and methods. Los Alamos National Laboratory, B287 PO Box 1663, Los Alamos, NM 87544, patchett@lanl.gov

James Ahrens received his Ph.D. in Computer Science in 1996 from the University of Washington. He is the Visualization Team Leader at the Advanced Computing Laboratory where he has worked since 1997. Los Alamos National Laboratory, B287 PO Box 1663, Los Alamos, NM 87544, ahrens@lanl.gov

David Pugmire is a Computer Scientist in the National Center for Computational Sciences at the Oak Ridge National Laboratory. He Can be reached at Oak Ridge National Laboratory, Building 5600, Room B203, P.O. Box 2008 MS6008, Oak Ridge, TN 37831-6008, E-Mail:pugmire@ornl.gov

Sean Ahern is the Visualization Task Leader in the National Center for Computational Sciences at Oak Ridge National Laboratory. He can be reached at Oak Ridge National Laboratory, Building 5600, Rm B205, P.O. Box 2008 MS6016, Oak Ridge, TN 37831-6016, E-Mail:ahern@ornl.gov