Parallel Analysis and Visualization on Cray Compute Node Linux

David Pugmire, *Oak Ridge National Laboratory and*
Hank Childs, *Lawrence Livermore National Laboratory*
and **Sean Ahern**, *Oak Ridge National Laboratory*

ABSTRACT: *Capability computer systems are deployed to give researchers the computational power required to investigate and solve key challenges facing the scientific community. As the power of these computer systems increases, the computational problem domain typically increases in size, complexity and scope. These increases strain the ability of commodity analysis and visualization clusters to effectively perform post-processing tasks and provide critical insight and understanding to the computed results. An alternative to purchasing increasingly larger, separate analysis and visualization commodity clusters is to use the computational system itself to perform post-processing tasks. In this paper, the recent successful port of VisIt, a parallel, open source analysis and visualization tool, to compute node linux running on the Cray is detailed. Additionally, the unprecedented ability of this resource for analysis and visualization is discussed and a report on obtained results is presented.*

KEYWORDS: ORNL, NCCS, Cray XT4, analysis, visualization

1. Introduction

ORNL NCCS

The National Center for Computational Sciences (NCCS) was established at Oak Ridge National Laboratory (ORNL) in 1992. In 2004 the Secretary of Energy designated the center as the Leadership Computing Facility for the nation, with the mission of delivering world class computing facilities for open scientific research.

The primary goal of the NCCS is to support open science and research in areas that the Department of Energy (DOE) deems worthy of investigation. This is primarily accomplished through the DOE's Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program, with over 145 million processing hours awarded on ORNL systems for the 2008 year. INCITE is one of the United States' primary programs for computationally intensive research.

Analysis and Visualization at the NCCS

One significant challenge to supercomputing at these scales is understanding the enormous amount of information output by these simulations. A typical simulation can output data files that are larger than the total size of the entire Library of Congress. One powerful method of understanding the information contained in these outputs is the use of scientific analysis and visualization. Creating a visual representation of the data uses the power of the human visual system to serve as a guide in exploring the results of a calculation. Being able to see visual representations of a calculation can serve as a power tool enabling scientists to draw correlations and gather insight into their data. Supporting INCITE users with analysis and visualization needs is one of the primary goals of the Visualization Team in the NCCS.

As the computational resources available to researchers increases, the size, complexity and scope of the scientific problem domain typically increases as well. These increases strain the ability of traditional modes of analysis and visualization. In this paper, several methods of addressing this strain on system resources are discussed

and the recent port of VisIt, a popular open source analysis and visualization tool developed by the Department of Energy is detailed. Additionally, results of a scaling study are presented.

2. Background

Large scale production analysis and visualization

In order to service the output of large computational resources, one common method employed by analysis and visualization software is to use a distributed, client-server architecture. In this type of architecture, shown in Figure 1, the visualization application is composed of two components, one component running on the users work station and the other component running in parallel on a cluster. A number of full featured, production analysis and visualization tools use this type of architecture, including, VisIt [1], Paraview [3] and EnSight [2].

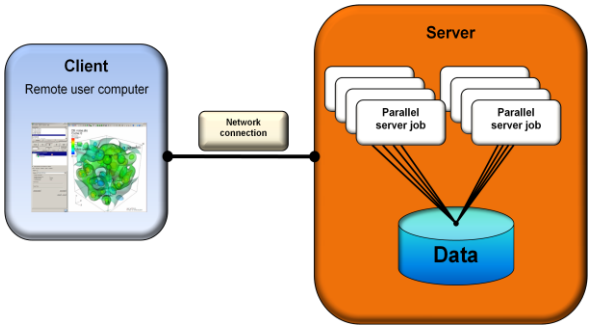

Figure 1. Client Server architecture. A distributed architecture where client and scalable server components run on different resources. Synchronization and communication is done over a connection.

The client side of the architecture utilizes the graphics hardware on the user workstation to render the geometry extracted from the data as well as provide a graphical user interface (GUI). The server side of the architecture runs in parallel, allowing it to scale to the size of the simulation data. Each parallel server will interact with the data directly, perform the analysis and send the resulting geometry to the client side for rendering and viewing. This architecture has served the production analysis and visualization community well for some time and will likely continue into the future.

Client-server deployment options

One common method of deploying client-server architecture type analysis tools is to have a separate cluster dedicated to analysis and visualization tasks. In this method, as shown in Figure 2, the supercomputer outputs results of a simulation to disk. The analysis cluster is then used to read data from disk (ideally, a shared disk) and perform the analysis. Since analysis and visualization is almost always I/O bound, these systems are typically designed to include features such as fast parallel file system and a very large memory foot print.

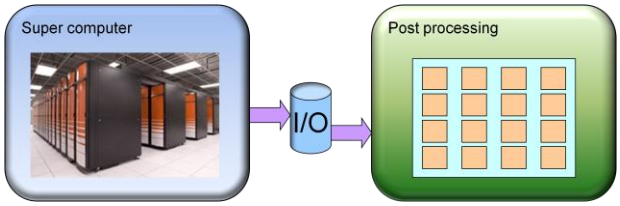

Figure 2. Separate computing and analysis resources. Using this method, two independent computing resources are used, one for simulation computation and the second for post processing analysis and visualization.

This method of deploying a separate cluster dedicated to analysis has worked well in the past, but is unlikely to continue to be a viable work path going forward. As the size of super computers and the amount of data being output continues to grow, deploying increasingly larger clusters dedicated to analysis becomes prohibitively expensive.

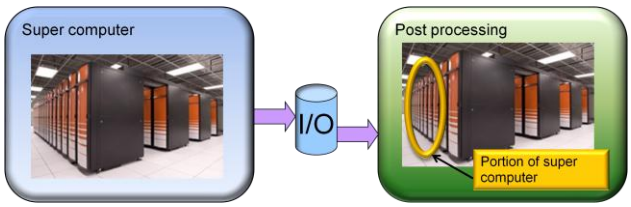

Figure 3. Shared computing and analysis resources. Using this method, the computing resource is shared, performing both computation and post-processing analysis and visualization.

An alternative to having a dedicated analysis cluster is run both computation and analysis on the same system. As shown in Figure 3, a portion of the super computer is used to perform analysis and visualization. By using the same resource for both computation and analysis, scalability of the analysis is no longer an issue.

However, doing both computation and analysis on the same system has historically been complicated. In order to maximize performance, the operating systems on computational systems typically consist of a very optimized but restricted subset of a full featured operating system. Analysis and visualization tools are often feature rich, requiring operating system functionality that is not always supported in optimized computational operating systems. Porting a production visualization tool to run on these computational operating systems can be challenging.

Recently, as reported in [5], VisIt was ported to run on the Catamount operating system on a Cray XT3. Catamount, a light weight computational operating system lacks two key features used by VisIt. First, support for shared libraries and second, support for sockets. The first limitation can be overcome, at the expense of creating a large executable, by building VisIt statically. The second limitation proved more challenging. In the VisIt implementation of the client-server architecture, the connection between the client and the server is made using TCP/IP over sockets, which are not supported in Catamount. Altering the communication protocol in the tool would require changes to both the client and server side and would be a serious maintenance, compatibility and support issue for the future. To minimize the impact, a small change in the server side component was made to communicate with a socket-*like* feature in Catamount called portals. A client side daemon would then be installed to translate the communication done with portals to sockets. The client side components could then function without modification.

3. Porting VisIt to CNL

Jaguar is the National Center for Computational Science's 263 TFLOPS Cray XT4 running at Oak Ridge National Laboratory. The system runs Compute Node Linux (CNL) on the computational nodes. CNL is a recently developed, optimized operating system that has a feature set richer than Catamount but with nearly similar performance

CNL provides two key features that were important in simplifying the port of VisIt to Jaguar; full support for sockets and partial support for shared libraries. Full support of sockets allows the VisIt implementation of the client-server architecture to run directly, unmodified on Jaguar. Direct communication can occur between the client and the server using TCP/IP sockets. The partial support for shared libraries allows for a much smaller executable footprint that loads components on an as-needed basis and simplifies the VisIt build process as the default mode is to use shared libraries.

With these enhancements to CNL, only two very minor challenges remained in the Visit port. First, CNL has access only to the Lustre parallel file system and second, a partial support of shared libraries. While shared libraries are supported on the compute nodes running CNL, passing information by means of the job launch system into the compute nodes about where the shared libraries are located proved difficult.

The solution to the first challenge was found by placing the VisIt installation and copying all the supporting system libraries into a location on lustre.

The solution to the second challenge was a minor modification to VisIt's parallel component launching mechanism. When the parallel computational component is launched on the server-side with qsub/aprun, instead of running the server-side application a proxy application is run. This proxy application sets the environment variable required for shared libraries and then launches the parallel component.

With these two minor changes in place, VisIt was successfully installed and deployed as a production tool for users on Jaguar.

4. Results

As results from simulations get larger and larger, we anticipate a desire to run visualization and analysis software on the only place that has sufficient resources to handle this data: on the supercomputer that generated them. This study was an attempt to better understand what the performance characteristics would be on such a machine and how viable this machine will be for visualization purposes in the future. Two strong scaling cases were used, an interactive session using ray casting and a detailed timing of the iso-contouring of a data set at two different resolutions.

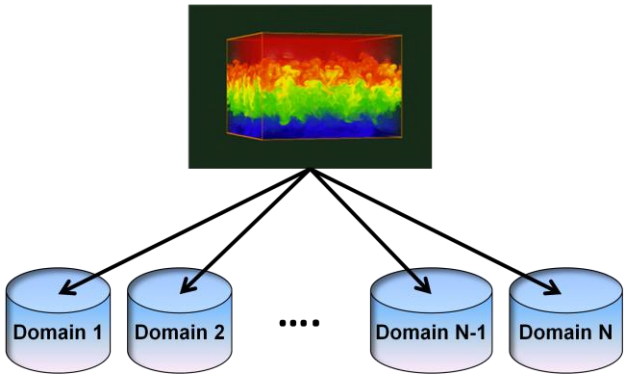

Figure 4. Domain decomposed data. The simulation is composed of a set of data files on disk.

The examples used consisted of domain decomposed data sets. As shown in Figure 4, each domain is in a separate file on disk. Each test consisted of loading all of the domains from disk into memory, execution of an analysis algorithm on the data and then rendering the resulting image.

Interactive exploration

For the interactive exploration test case, a simulation of a Richtmyer-Meshkoff instability computed on a mesh of size 2048x2048x1920 with 960 domains was used [4]. The exploration was done using ray casting and an image size of 1024x1024. In order to maximize the amount of work done by the ray casting algorithm, the camera view was set so that the simulation data filled the entire viewing window. At each pixel in the image, a number of rays are cast into the data for sampling. A transfer

function is used to specify the mapping from data value to color and opacity. In this study a range of between 500 and 2500 samples per pixel was used.

Interactive frame rates for differing sampling and processor configurations are shown in Figure 5. While not exhibiting ideal scaling, it is possible for a user to interact with and explore a very large data set.

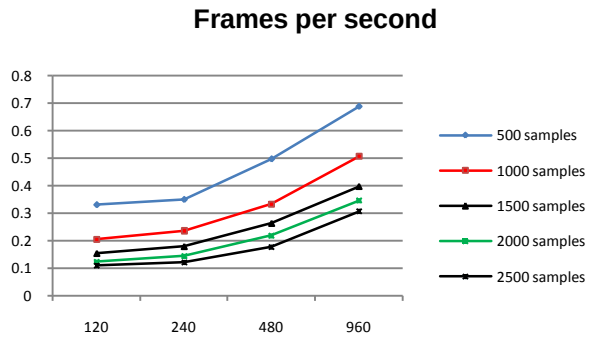

Figure 5. Frames per second ray casting a 1024 x 1024 pixel image. Results are shown on sampling rates between 500 and 2500 samples per pixel and between 120 and 960 processors.

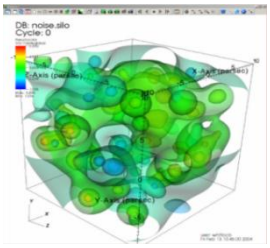

Figure 6. Isocontour test dataset. A number of random seed values are distributed within a volume. Values at each individual cell are calculated by interpolation of the random seed points.

Isocontouring scaling study

In the isocountouring scaling study, detailed timings were taken of loading a dataset, extracting isocontour surfaces and rendered the resulting image. Three data sets were used, each an interpolated noise data set, shown in Figure 6, resampled onto a larger grid. Test cases included a 1 billion cell data set with 1000 domains and a 10 billion cell data set, one with 1000 and the other with 2744 domains. In each case, timings of the major computational tasks were taken while executing a script. The steps in the process are as follows:

1. Load data from disk
2. Extract ten iso contour surfaces
3. Render image
    a. Render triangles
    b. Image compositing

Steps 1, 2 and 3a are each handled in parallel by a separate processing running on the server side and are expected to scale as additional nodes are added. The performance of step 1 is dependent on the performance of the underlying lustre file system and the load across the system of other jobs competing for I/O, and so actual scaling may vary. The image compositing step in 3b is not expected to scale. Each parallel process on the server renders the geometry for its subset of the total dataset, producing a partial image. These partial images must be composited together to form the final image. As more processors are added, more sub-images must be composited together and so the time required will increase.

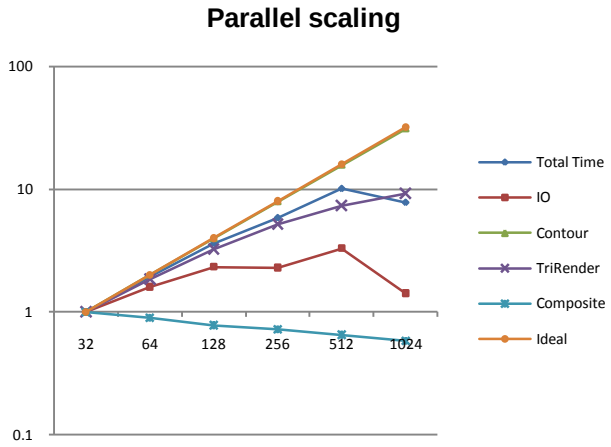

Figure 7. Parallel scaling on a 1 billion zones, 1000 domain dataset.

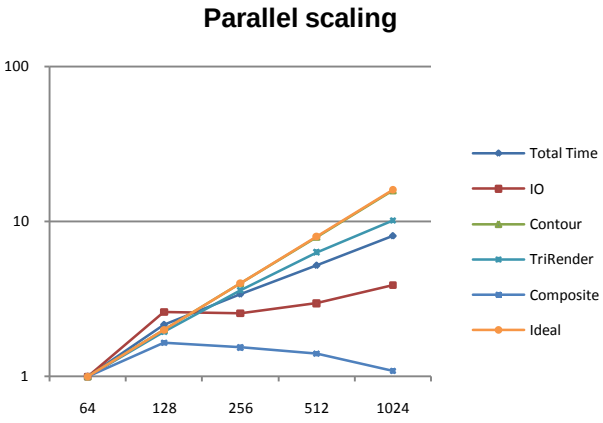

Figure 8. Parallel scaling on a 10 billion zones, 1000 domain dataset.

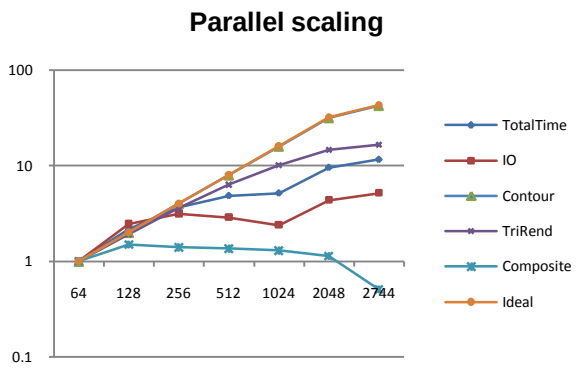

Figure 9. Parallel scaling on a 10 billion zones, 2744 domain dataset.

Log scale plots of results can be seen in Figure 7, Figure 8 and Figure 9. The scaling of the entire execution is captured by the "TotalTime" graph in each figure. In all three cases, the overall scaling of the entire execution is determined by the scaling of the I/O. Since analysis and visualization is typically an I/O bound task, this result was surprising. As discussed previously, the performance of the I/O exhibited a lot of variability as it is dependent on the overall system load and demand of other concurrently running jobs for I/O resources. In all three cases, I/O seems to exhibit good scaling to around 128 processors and then begins to degrade. As expected, contouring exhibited very good scaling in all three cases. Contouring is an *embarrassingly* parallel algorithm with no dependence on communication or I/O. Triangle rendering exhibited fairly good scaling but better scaling on larger number of processors was expected. It is possible that the tests cases chosen resulted in an imbalance in the number of triangles to be rendered for each processor, but this requires further investigation. Finally, as discussed previously, the composite step did not scale, as was expected

5. Conclusions

A scalable, production analysis and visualization tool is running on the Cray XT4 at Oak Ridge National Laboratory. Efforts to port the tool to the Compute Node Linux operating system were fairly straightforward and the scaling performance on large data sets was in line with our initial expectations. The scalable portions of the visualization pipeline exhibited good scaling and I/O scaled to a certain point before becoming a bottleneck.

One very valuable lesson from this work was information learned about the behavior of VisIt running on a very large number of processors. As timing results were being analysed, several optimizations were identified for incorporation into VisIt.

In conclusion, Cray Compute Node Linux proved to be an adequate, optimized computational kernel for performing analysis and visualization on the computational platform. The availability of such a large scale resource will be an invaluable tool for scientists to explore and understand larger and more complex simulations.

Acknowledgments

Thanks to Kevin Thomas of Cray Inc. for discussions on his work porting VisIt to Catamount and for answering questions on Compute Node Linux. Thanks also to Jeff Larkin of Cray Inc. for answering questions about Compute Node Linux.

About the Authors

David Pugmire is a Computer Scientist in the National Center for Computational Sciences at 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.

Hank Childs is a Computer Scientist at Lawrence Livermore National Laboratory. He can be reached at Lawrence Livermore National Laboratory, L-557, Livermore, CA 94550, E-Mail: child3@llnl.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, Room B205, P.O. Box 2008 MS6016, Oak Ridge, TN 37831-6016, E-Mail: ahern@ornl.gov.

References

[1] Childs, Brugger, Bonnell, Meredith, Miller, Whitlock, Max, "A Contract Based System for Large Data Visualization", VIS '05: Proceedings of the Conference on Visualization '05", Minneapolis, MN, Oct 2005.

[2] Computational Engineering International, "EnSight User Manual", May 2003.

[3] Law, Henderson, Ahrens, "An Application Architecture for Large Data Visualization: A Case Study", PVG '01: Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization, San Diego, CA, 2001.

[4] Mirin, Cohen, Curtis, Dannevik, Dimits, Duchaineau, Eliason, Schikore, Anderson, Porter, Woodward, and White, "Very High Resolution Simulation of Compressible Turbulence on the IBM-SP System," Supercomputing 99 Conference, Portland, OR, November 1999.

[5] Thomas, "Porting of VisIt Parallel Visualization Tool to the Cray XT3", Cray Users Group Meeting, Seattle, WA, May 2007.