LARGE SCALE PROBLEM SOLVING USING AUTOMATIC CODE GENERATION AND DISTRIBUTED VISUALIZATION

Abstract. Scientific computation today is facing many scalability challenges and issues in trying to take advantage of latest generation compute, network and graphics hardware. We are proposing a comprehensive method to solve four important scalability challenges: programming productivity, scalability to a large number of processors, I/O bandwidth and interactive visualization of large data. We will show how our proposed systems can be combined in a single scenario where *McLachlan* code [1] is generated, executed on a large computational cluster and its output is distributed in a temporary distributed data server using a high-speed optical grid and visualized remotely using distributed visualization methods and optical networks. The system is applicable to a large number of problems and the specific scientific application that we will present in our demonstration is binary black hole simulation.

1. Introduction

We are looking at methods of solving four fundamental scalability challenges, each of them important on their own and propose a comprehensive system that addresses all of them. We will show a single end-to-end application capable of scaling to a large number of processors and whose output can be visualized remotely by taking advantage of high speed networking capabilities and of GPU-based parallel graphics processing resources.

The challenges that we are addressing are the following:

Programming productivity. The programming productivity has long been a concern in the computational science community. In addition to possible human errors, the limit for code writing and the ever-growing complexity of many scientific codes make the development and maintenance of many large scale scientific applications an intimidating task. Things get even worse when one is dealing with extremely complicated systems such as the Einstein equations which usually have more than 20 evolved variables and thousands of terms in the source terms. In addressing these issues, we present our latest work on generic methods for generating code that solves a set of coupled nonlinear partial differential equations using the *Kranc* code generation package [7]. Our work greatly benefits from the modular design of the *Cactus* framework [5]. which frees domain experts from lower level programming issues, i.e., parallelization, parameter parsing, I/O, visualization etc. In this collaborative problem solving environment based on *Cactus* and *Kran*, application developers, either software engineers or domain experts, can contribute to a code with they expertise at their maximum scale, thus enhance the overall programming productivity.

Scalability to large number of processors. With the advent of Roadrunner, the first supercomputer that broke the petaflop/s mark in year 2008, we officially entered the petascale era. There are a great number of challenges to overcome in order to fully leverage the enormous computational power to solve previously unattainable scientific problems. The most urgent of all is the design and development of highly scalable and efficient scientific applications. However, the ever growing complexity in developing such efficient parallel software always leaves a gap for many application developers to cross. We need a bridge, a computational infrastructure, which can not only hide the hardware complexity, but also provide a user friendly interface for scientific application developers to speed up scientific discoveries. In this proposal, we present a highly efficient computational infrastructure that is based on the *Cactus* framework and the *Carpet* AMR library [8].

I/O bandwidth. We are faced with difficult challenges in moving data when dealing with large datasets, challenges that arise from I/O architecture, network protocols and hardware resources. I/O architectures that do not use a non-blocking approach are fundamentally limiting the I/O performance Standard network protocols such as *TCP* cannot utilize the bandwidth available in

emerging optical networks and cannot be used efficiently on wide-area networks. Single disks, or workstations are not able to saturate high-capacity network links. We propose a system that combines an efficient pipeline-based architecture, can take advantage of non-standard high-speed data transport protocols such as *UDT* and use distributed grid resources to increase the I/O throughput.

Interactive visualization of large data. Bringing efficient visualization and data analysis power to the end users' desktop while visualizing large data and maintain interactiveness, by as having the ability to control and steer the visualization by the user is a major challenge for visualization applications today. We are looking at the case where sufficiently powerful visualization resources are not available at either the location where the data was generated or at the location where the user is located, and propose using visualization clusters in the network to interactively visualize large amounts of data.

The specific application scenario in our demonstration is the numerical modeling of the gravitational waves produced by binary black hole systems. Such simulations are crucial for detecting and interpreting signals soon to be recorded from ground-based laser interferometric detectors (LIGO, GEO600, VIRGO).

2. Automatic Parallel Code Generation

2.1. **Cactus-Carpet Computational Infrastructure.** *Cactus* is an open source software framework consisting of a central core, the *flesh*, which connects many software components (or *thorns* through an extensible interface. *Carpet* serves as a driver layer of the *Cactus* framework providing adaptive mesh refinement, multi-patch capability, as well as memory management, parallelization, and efficient I/O. In the *Cactus-Carpet* computational infrastructure, the simulation domain is discretized using finite differences on block-structured grids, employing a Berger-Oliger block-structured adaptive mesh refinement method [2] which provides both efficiency and flexibility. The time integration schemes used are explicit Runge-Kutta methods.

Cactus is highly portable and runs on all variants of the Unix operating system as well as the Windows platform. Codes written using *Cactus* have been run on various brands of the fastest computers in the world, such as the SGI Altix, the Japanese Earth Simulator, the IBM Blue Gene/L, the Cray XT5, the Sun Constellation Linux Cluster, and the SiCortex system, just to name a few. Very recently, the *Cactus* team successfully carried out benchmark runs on 131,072 cores on the IBM Blue Gene/P at the Argonne National Laboratory.

2.2. **Kranc Code Generation Package.** *Kranc* is a suite of Mathematica-based computer-algebra packages designed to facilitate analytical manipulations of tensorial systems of equations, and to automatically generate C or Fortran code for solving initial boundary value problems. *Kranc* provides a Mathematica function for creating *Cactus* thorns. A thorn generated by *Kranc* will (i) define the grid functions which the simulation will use; (ii)Compute the right hand sides of evolution equations so that the time integrator can compute the evolved variables at the next time step. (iii) perform a user-specified calculation at each point of the grid.

2.3. **McLachlan Code.** The *McLachlan* code was created in the XiRel project [3], which is aiming at developing a highly scalable, efficient and accurate adaptive mesh refinement layer for the *Cactus* Framework, based on the *Carpet* driver, and optimized and supported for numerical relativitists studying the physics of black holes, neutron stars and gravitational waves. *McLachlan* [1] is automatically generated using the *Kranc* code generation package from a high level set of partial differential equations. The automation is of particular importance for experimenting with new numerical methods or particular machine specific code optimizations as well as the hybrid MPI and OpenMP parallel programming method. *On Ranger,* McLachlan *scales to more than 8,000 cores which is the largest run done with* Cactus-Carpet *computational infrastructure so far.* As we can see from figure 1, the performance sharply drops from 1,000 cores to 2,000 core on Franklin, while the drop starts from about 4,000 cores on Ranger. Both Ranger and Queen Bee show similar scaling behavior at least up to 2,000 cores.

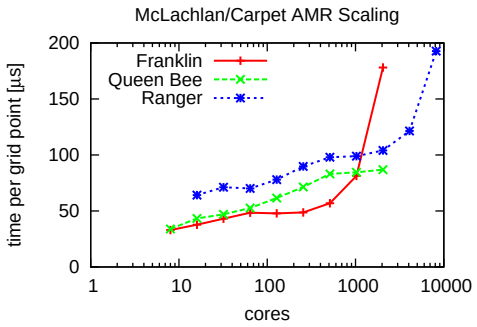

FIGURE 1. Weak scaling benchmark runs with the *McLachlan* code on Ranger, a Sun Constellation Linux Cluster at TACC on more than 8,000 cores.

3. Distributed Visualization

3.1. **Visualization Scenario.** Our scenario is the following: The visualization user is connected over a network link to a grid system of various types of resources (visualization, network, compute, data). The data to be visualized is located on a data server that is also connected to the grid system.

There are various ways in which a visualization application can be created to solve the problem, such as running the visualization on the data server and running a video stream to the client, or run the visualization on the local client and run a data stream between the data server and the client.

These two solutions are limited by the visualization power available near the data server or near local machine respectively. Since powerful visualization resources are not available at the client and may not be available near the data server we propose to build a three-way distributed system that uses a visualization cluster in the network, data streaming from the data server to the visualization cluster and video streaming from the visualization cluster to the local client.

We have taken the problem one step further and considered the case where the network connection of the data server is a relatively slow one — much slower than the network capacity of the rendering machine. In this situation we are dealing with I/O scalability issues and the solution we propose is to create a temporary distributed data server in the grid. The distributed data server uses compute and data resources that are not dedicated for this application but are allocated on-demand to support it when it needs to execute. The distributed data server can sustain much higher data transfer rates than a single data source. Data will be loaded in advance from the source on the distributed data server. *We have already demonstrated during SC08 how using this approach we can transfer data at speeds exceeding 5Gbps.* The architecture of this approach is illustrated in Figure 2. In this scenario, because visualization resources that are not local to the end client, a remote interaction system is necessary in order for the user to be able to connect and steer the visualization.

3.2. **I/O.** High-performance data transmission over wide-area networks is difficult to achieve. One of the main factors that can influence performance is the network transport protocol. Using unsuitable protocols on wide area network, can result in bad performance — for example a few Mbps throughput on a 10Gbps dedicated network connection using *TCP*. The application needs to use protocols that are suitable for the network that is utilized. We propose using the *UDT* [6] library for wide-area transfers. Another issue is blocking on I/O operations. Blocking I/O reduces the performance that is seen by the application and the solution we propose is to use a completely non-blocking architecture using a large number of parallel threads to keep the data flow moving.

3.3. **Parallel rendering.** Parallel rendering on HPC or visualization clusters is utilized to visualize large datasets. *We have used eVolve for Equalizer (a parallel rendering framework)* [4] *on a rendering cluster at LSU to interactively visualize 2 Gbytes of visualization data at 5.5 fps. where a single machine can only visualize 256 Mb of data*

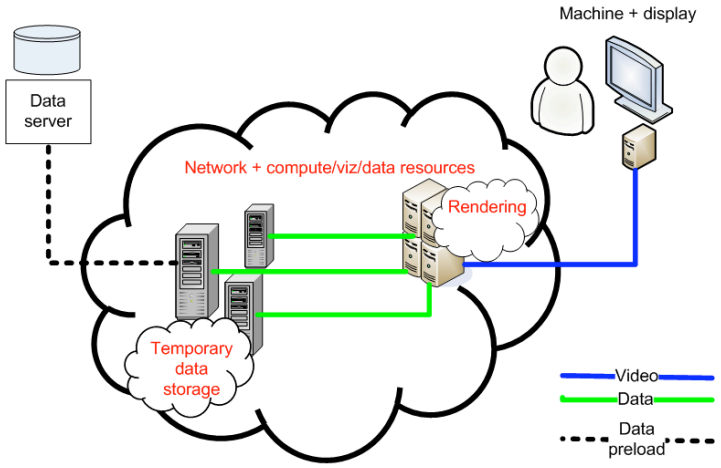

FIGURE 2. Architecture of demonstration system which involves a temporary distributed data server allocated on-demand to improve sustained data transfer rates.

VisIT's volume rendering plot doesn't support parallel hardware-accelerated volume rendering, thus largely limits its ability to accommodate native visualization of large-scale volumes. *eVolve* is a volume renderer based on 3D texture and pre-integration algorithm. Visual quality is controlled by how many slices it renders. *eVolve*'s parallel rendering only de-compose the volume along the z direction, which means we need to use more slices for each node in order to get a good visual quality. In this demonstration we propose to use *eVolve* or a self-developed parallel GPU ray-tracing volume rendering algorithm. GPU ray-tracing does float point compositing in a single pass using a fragment shader. The trade-off between rendering time and visual quality, can be steered directly by the user(s).

3.4. **Video Streaming and Interaction.** Interaction with the remote parallel renderer is necessary to modify navigation parameters such as the direction of viewing or the level of zoom, and to control the trade-off of visual quality and image deliver time. Since the visualization application is not local, an interaction system consisting of three components was developed. The components are a local interaction client running on the local machine, an interaction server running on the rendering cluster and an application plug-in that connects the interaction server to the application and inserts interaction commands in the application workflow. For our demonstration we propose to use specialized interaction devices developed by the Tangible Visualization Laboratory at CCT that are very useful in long-latency remote interaction systems and can support collaboration (collaborative visualization) from multiple sites.

The final component of the system is the video streaming. Images that are generated from the remote visualization cluster need to be transported to the local client for the user to see. In the past we have successfully utilized hardware-assisted systems running videoconferencing software (*Ultragrid*) and software-based video streaming using *SAGE*. For this demonstration, depending on the network bandwidth available we will use *VirtualGL* or *SAGE* to stream the video from LSU to SCALE'09 show floor.

4. Expected outcome. Deliverable. Impact

The precise hardware configuration for the demonstration is yet to be determined. The simulation will be executed on QueenBee or Ranger. We are planning to have a visualization streaming client available on the show-floor, connected using network services to LSU in Baton Rouge. Data will possibly be distributed on resources in LONI, Brno, Czech Republic and Oak Ridge National Laboratory (ORNL). We propose to connect the data servers to the rendering cluster using a new type of high-speed network services, for example Dynamic Circuit Networks (DCN).

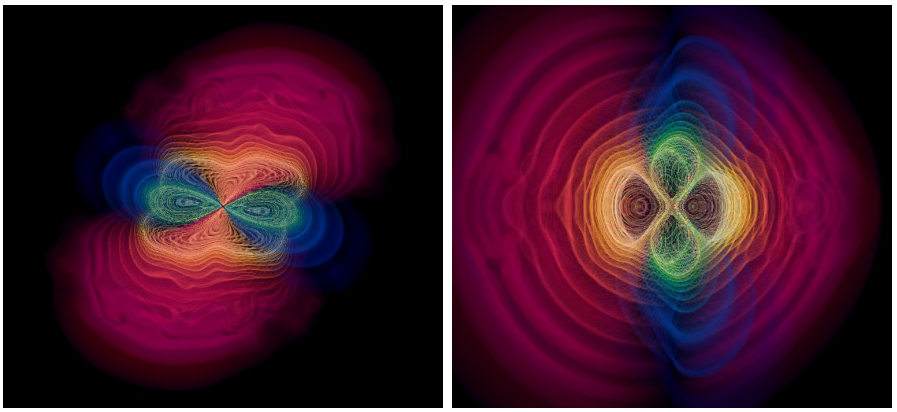

FIGURE 3. Volume rendering of the gravitational radiation during a binary black hole merger, represented by the real part of Weyl scalar $r \cdot \Psi_4$.

Our demonstration will show the viability of the proposed approach of using a code generator and *Cactus* to scale to a large number of processors and how distributing the visualization system in three components will increase the amount of data and the speed at which the data can be visualized compared to local techniques, as well as improved data transfer rates and interactivity.

Our integrated approach will provide a solution and a model for future scientific computing, and many and diverse applications involving the resolution of initial value boundary problems will be able to benefit from the proposed method.

5. TEAM

Eloisa Bentivegna[(1)], Peter Diener[(1,3)], Jinghua Ge[(1)], Andrei Hutanu[(1,2)], Cornelius Toole[(1,2)], Ravi Paruchuri[(1)], Erik Schnetter[(1,3)], Jian Tao[(1)] and Gabrielle Allen[(1,2)] {bentivegna, diener, jinghuage, ahutanu, corntoole, ravi9, schnetter, jtao, gallen}@cct.lsu.edu, with support from LONI, Internet2 DCN, ORNL, Teragrid, Masaryk University Brno.

[(1)] Center for Computation & Technology, Louisiana State University, Baton Rouge, LA, USA

[(3)] Department of Computer Science, Louisiana State University, Baton Rouge, LA 70808, USA

[(2)] Department of Physics & Astronomy, Louisiana State University, Baton Rouge LA 70808, USA

REFERENCES

[1] McLachlan, a Public BSSN Code: http://www.cct.lsu.edu/ eschnett/McLachlan/index.html.

[2] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[3] The XiRel project: `http://www.cct.lsu.edu/xirel`.

[4] Stefan Eilemann, Maxim Makhinya, and Renato Pajarola. Equalizer: A scalable parallel rendering framework. In *IEEE Transactions on Visualization and Computer Graphics*, 2008.

[5] Tom Goodale, Gabrielle Allen, Gerd Lanfermann, Joan Massó, Thomas Radke, Edward Seidel, and John Shalf. The Cactus framework and toolkit: Design and applications. In *High Performance Computing for Computational Science - VECPAR 2002, 5th International Conference, Porto, Portugal, June 26-28, 2002*, pages 197–227, Berlin, 2003. Springer.

[6] Yunhong Gu and Robert L. Grossman. Udt: Udp-based data transfer for high-speed wide area networks. *Comput. Networks*, 51(7):1777–1799, 2007.

[7] Sascha Husa, Ian Hinder, and Christiane Lechner. Kranc: a Mathematica application to generate numerical codes for tensorial evolution equations. 2004.

[8] E. Schnetter, S. H. Hawley, and I. Hawke. Evolutions in 3D numerical relativity using fixed mesh refinement. *Class. Quantum Grav.*, 21(6):1465–1488, 21 March 2004. gr-qc/0310042.