LOUISIANA STATE UNIVERSITY
Center for Computation & Technology

CCT Technical Report Series
CCT-TR-2009-11

Large-scale Problem Solving Using Automatic Code Generation and Distributed Visualization

Andrei Hutanu, Louisiana State University, Baton Rouge, LA USA
Erik Schnetter, Louisiana State University, Baton Rouge, LA USA
Werner Benger, Louisiana State University, Baton Rouge, LA USA
Eloisa Bentivegna, Louisiana State University, Baton Rouge, LA USA
Alex Clary, Louisiana State University, Baton Rouge, LA USA
Peter Diener, Louisiana State University, Baton Rouge, LA USA
Jinghua Ge, Louisiana State University, Baton Rouge, LA USA
Robert Kooima, Louisiana State University, Baton Rouge, LA USA
Oleg Korobkin, Louisiana State University, Baton Rouge, LA USA
Kexi Liu, Louisiana State University, Baton Rouge, LA USA
Frank Loffler, Louisiana State University, Baton Rouge, LA USA
Ravi Paruchuri, Louisiana State University, Baton Rouge, LA USA
Jian Tao, Louisiana State University, Baton Rouge, LA USA
Cornelius Toole, Louisiana State University, Baton Rouge, LA USA
Adam Yates, Louisiana State University, Baton Rouge, LA USA
Gabrielle Allen, Louisiana State University, Baton Rouge, LA USA

Posted August 2009.

cct.lsu.edu/CCT-TR/CCT-TR-2009-11
The author(s) retain all copyright privileges for this article and accompanying materials.
Nothing may be copied or republished without the written consent of the author(s).

LARGE SCALE PROBLEM SOLVING USING AUTOMATIC CODE GENERATION AND DISTRIBUTED VISUALIZATION

ANDREI HUTANU[1,2], ERIK SCHNETTER[1,3], WERNER BENGER[1], ELOISA BENTIVEGNA[1], ALEX CLARY[1,4], PETER DIENER[1,3], JINGHUA GE[1], ROBERT KOOIMA[1], OLEG KOROBKIN[1,3], KEXI LIU[1,2], FRANK LÖFFLER[1], RAVI PARUCHURI[1], JIAN TAO[1], CORNELIUS TOOLE[1,2], ADAM YATES[1], GABRIELLE ALLEN[1,2]

Abstract. Scientific computation today is facing many scalability challenges and issues in trying to take advantage of the latest generation compute, network and graphics hardware. We present a comprehensive approach to solving four important scalability challenges: programming productivity, scalability to a large number of processors, I/O bandwidth, and interactive visualization of large data. We describe a scenario of our proposed system applied to the field of numerical relativity, in particular the simulation of a binary black hole system. Here a solver for the governing Einstein equations is generated, executed on a large computational cluster, and its output is distributed onto a temporary distributed data server using a high-speed optical grid, and finally visualized remotely using distributed visualization methods and optical networks. This system is applicable to a large number of problems. A demonstration of this system was awarded first place in the IEEE SCALE 2009 Challenge in Shanghai, China in May 2009.

1. Introduction

We describe the motivation, design, and experimental experiences of an end-to-end system for large scale, interactive and collaborative numerical simulation and visualization that addresses a set of fundamental scalability challenges for real world applications. This system was awarded first place in the IEEE SCALE 2009 Challenge in Shanghai, China in May 2009.

Our system shows 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 four scalability challenges that are addressed are described below (see Figure 1).

1.1. **Programming productivity.**

Programming productivity has long been a concern in the computational science community: 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 [21]. Our work greatly benefits from the modular design of the *Cactus* framework [18, 1], which frees domain experts from lower level programming issues, i.e., parallelization, I/O, visualization etc. In this collaborative problem solving environment based on *Cactus* and *Kranc*, application developers, either software engineers or domain experts, can contribute to a code with their expertise, thus enhancing the overall programming productivity.

[1] Center for Computation & Technology, Louisiana State University, Baton Rouge, LA 70803, USA.
[2] Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA.
[3] Department of Physics & Astronomy, Louisiana State University, Baton Rouge LA 70803, USA.
[4] Department of Electrical & Computer Engineering, Louisiana State University, Baton Rouge LA 70803, USA.

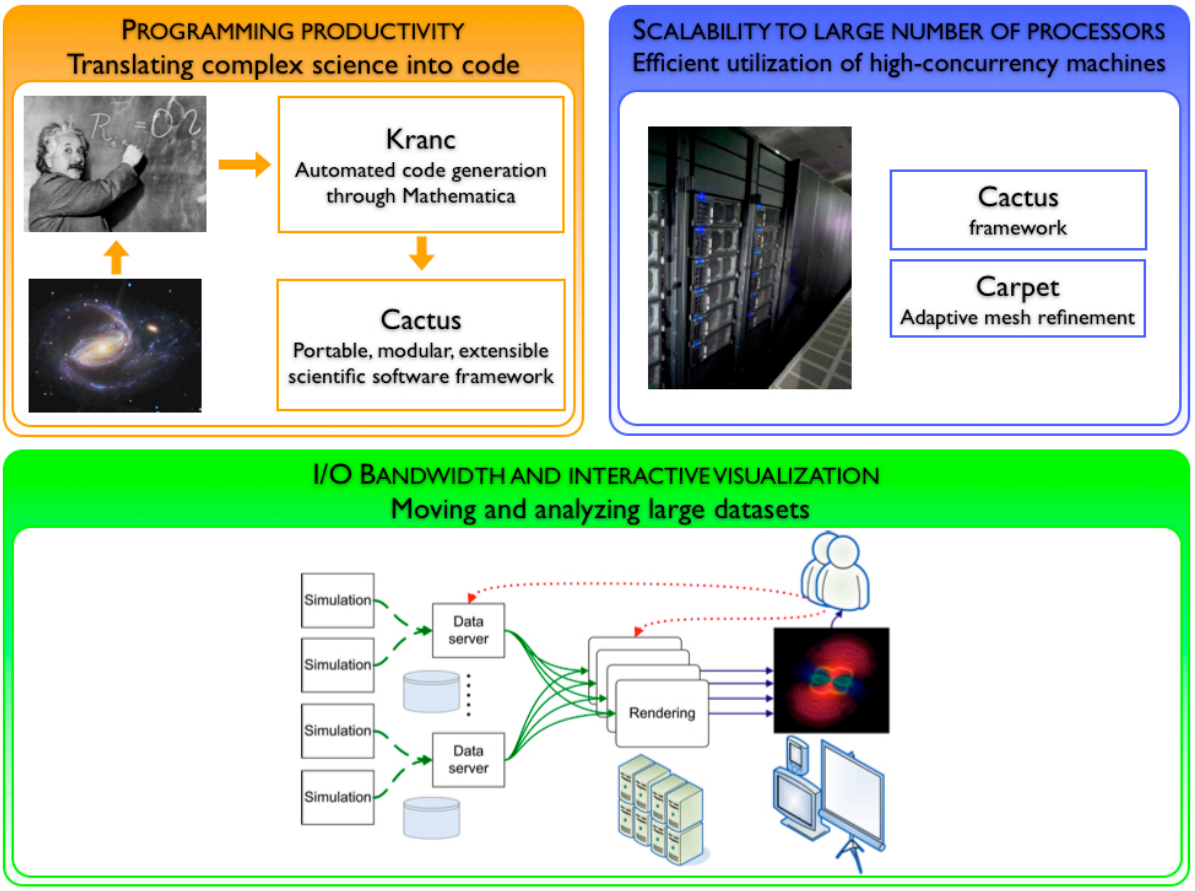

FIGURE 1. Scalability challenges involved in an end-to-end system for large scale, interactive numerical simulation and visualization for black hole modeling.

1.2. **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 this enormous computational power to be able 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 does not only hide the hardware complexity, but also provides a user friendly interface for scientific application developers to speed up scientific discoveries. In our project, we used a highly efficient computational infrastructure that is based on the *Cactus* framework and the *Carpet* AMR library [25, 27, 2].

1.3. **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.

1.4. **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 giving the user the ability to control and steer the visualization, 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 visualizing it, and propose using visualization clusters in the network to interactively visualize large amounts of data.

2. Scientific Motivation: Black Holes and Gamma-Ray Bursts

More than ninety years after Einstein first proposed his Theory of General Relativity, astrophysicists are more than ever and in greater detail studying regions of the universe where gravity is very strong and where the curvature of spacetime is large.

This realm of strong curvature is notoriously difficult to investigate with conventional observational astronomy. Some phenomena might not be observable in the electromagnetic spectrum at all, and may only be visible in the gravitational spectrum, i.e., via the gravitational waves that they emit, as predicted by Einstein's general relativity. Gravitational waves have today not yet been observed directly, but gravitational wave detectors (LIGO [3], GEO [4], VIRGO [5]) will soon reach sufficient sensitivities to observe interesting astrophysical phenomena.

In order to correctly interpret the gravitational-wave astronomy data, astrophysicists must rely on computationally challenging large-scale numerical simulations to study the details of the energetic processes occurring in regions of strong curvature. Such astrophysical systems and phenomena include the birth of neutron stars or black holes in collapsing evolved massive stars, the coalescence of compact binary systems, Gamma-Ray Bursts (GRBs), active galactic nuclei harboring supermassive black holes, pulsars, and oscillating neutron stars.

Of these, Gamma-Ray Bursts (GRBs) [23] are among the most scientifically interesting. GRBs are intense, narrowly-beamed flashes of γ-rays originating at cosmological distances, and the riddle concerning their central engines and emission mechanisms is one of the most complex and challenging problems of astrophysics today. The physics necessary in such a model includes general relativity, relativistic magneto-hydrodynamics, nuclear physics (describing nuclear reactions and the equation of state of dense matter), neutrino physics (weak interactions), and neutrino and photon radiation transport. The complexity of the GRB central engine requires a multi-physics, multi-length-scale approach that cannot be fully realized on present-day computers and will require petascale computing [24, 28].

At LSU we are performing simulations of general relativistic systems in the context of a decade-long research program in numerical relativity. One pillar of this work is focused particularly on 3D black hole physics and binary black hole inspiral and merger simulations. This includes the development of the necessary tools and techniques to carry these out, such as mesh refinement and multi-block methods, higher order numerical schemes, and formulations of the Einstein equations. A second pillar of the group's research is focused on general relativistic hydrodynamics simulations, building upon results and progress achieved with black hole system. Such simulations are crucial for detecting and interpreting signals soon expected to be recorded from ground-based laser interferometric detectors.

The specific application scenario in this demonstration is the numerical modeling of the gravitational waves produced by the inspiral and merger of binary black hole systems (see Figure 2).

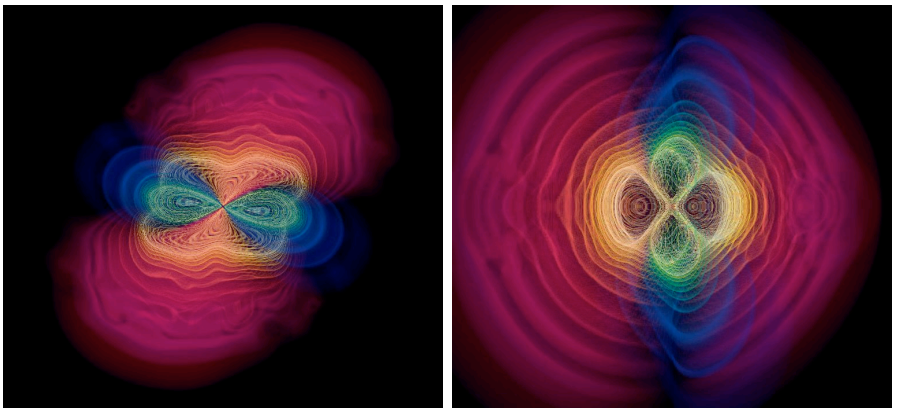

FIGURE 2. Volume rendering of the gravitational radiation during a binary black hole merger, represented by the real part of Weyl scalar $r \cdot \Psi_4$.

3. Automatic Parallel Code Generation

3.1. **Cactus–Carpet Computational Infrastructure.** *Cactus* [18, 1] is an open source software framework consisting of a central core, the *flesh*, which connects many software components (*thorns*) through an extensible interface. *Carpet* [25, 27, 2] 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 high order finite differences on block-structured grids, employing a Berger-Oliger-style adaptive mesh refinement method [13] with sub-cycling in time, which provides both efficiency and flexibility. We use explicit Runge-Kutta methods for time integration.

Cactus is highly portable and runs on all current HPC platforms as well as on workstations and laptops on all major operating systems. Codes written using *Cactus* have been run on various brands of the fastest computers in the world, such as various Intel and AMD based systems, SGI Altix, the Japanese Earth Simulator, IBM Blue Gene, Cray XT, and the SiCortex architecture, among others. 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 [6].

3.2. **Kranc Code Generation Package.** *Kranc* [22, 21, 7] is a Mathematica-based computer algebra package designed to facilitate analytical manipulations of systems of tensorial equations, and to automatically generate C or Fortran code for solving initial boundary value problems. *Kranc* generates complete *Cactus* thorns, starting from a high-level description including the system of equations formulated in high-level Mathematica notation, and discretizing the equations with higher-order finite differencing. *Kranc* generated thorns make use of the Cactus Computational Toolkit, declaring to Cactus the grid functions which the simulation will use, and computing the right hand sides of the evolution equations so that the time integrator can advance the solution in time.

3.3. **McLachlan Code.** The *McLachlan* code [14, 8] was developed in the context of the XiRel project. XiRel [30, 9] is a collaboration between several numerical relativity groups worldwide to develop a highly scalable, efficient, and accurate adaptive mesh refinement layer for the *Cactus* framework, based on the *Carpet* driver, enabling numerical relativists to study the physics of black holes, neutron stars and gravitational waves. The *McLachlan* code is automatically generated using the *Kranc* code generation package (see above) from a high-level description of the underlying set of partial differential equations. The automation is of particular importance for experimenting with new formulations of the equations, new numerical methods, or particular machine specific-code optimizations. *McLachlan* employs a hybrid MPI/OpenMP parallelization.

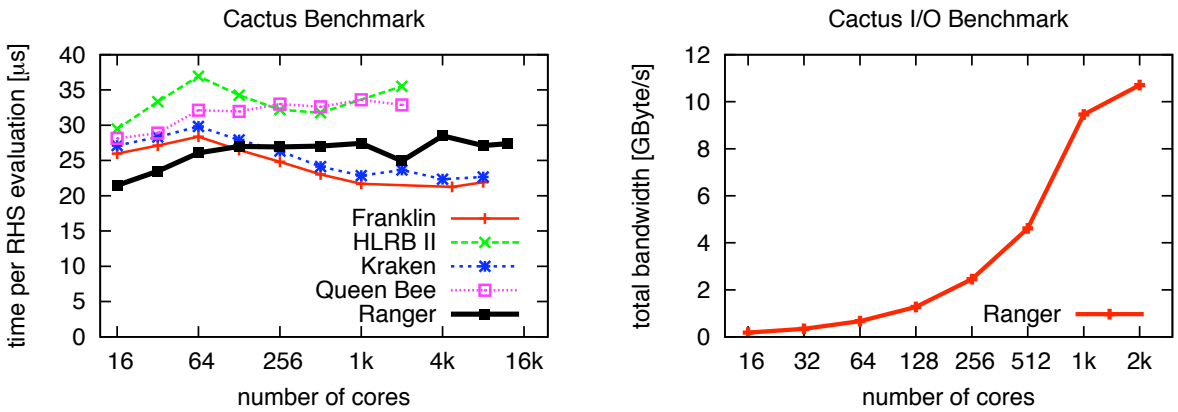

FIGURE 3. **Left:** Weak scaling benchmark results of the *McLachlan* code on several current leadership HPC systems. This benchmark simulates a single black hole with nine levels of mesh refinement. The code scales well up to more than 12,000 cores of Ranger at TACC. **Right:** I/O benchmark on Ranger, showing the total I/O bandwidth vs. the number of cores. Cactus achieves a significant fraction of the maximum bandwidth already on 1,000 cores.

As can been seen from Figure 3, *on TACC's Ranger,* McLachlan *and the supporting infrastructure scale well up to more than 12,000 cores. Cactus–Carpet* is also able to use a significant fraction of the theoretical peak I/O bandwidth already on 1,000 cores.

4. Interactive and Collaborative Simulations

4.1. **Monitoring, Profiling and Debugging.** Supporting performance and enforcing correctness of the complex, large-scale codes that Cactus generates is a non-trivial task, targeted by the NSF-funded Application-Level Performance and Correctness Analysis (Alpaca) project [26, 12].

In order to reap the benefits of the high-concurrency machines available today, it is not sufficient that a code's parallel efficiency remain constant as the size of the problem is scaled up, but also that its ease of control remains close to that of simulations carried out on a few number of computing cores; if this is not the case, the process of debugging and optimizing the code may be so time-consuming as to annul the speed-up obtained through parallelism. The Alpaca project addresses this issue through the development of application-level tools, i.e., high-level tools that are aware of the Cactus data structures and execution model.

In particular, Alpaca's objectives concentrate on three areas: (i) high-level debugging, devising debugging strategies that leverage high-level knowledge about the execution actors and the data processing, and develop tools that extract such information from a simulation and provide it in an abstract format to the user; (ii) high-level profiling, devising algorithms for extracting high-level information from timing data; and (iii) remote visualization, using visual control over the simulation data as a high-level correctness check. In particular, work within the Alpaca project includes the development of HTTPS, a Cactus module that spawns an SSL webserver, with X.509 certificate authorization, at the beginning of a simulation and uses it to receive incoming connections, expose the simulation's details and provide fine-grained control over its execution.

4.2. **Leveraging Web 2.0 for Collaborations.** Not only computer simulations themselves are becoming more complex, requiring more powerful resources, but the way science itself is carried out is changing in a similar way.

For a long time, scientists have worked either alone or in small groups and collected hand-written data in notebooks; this has changed with the growing use of computers and world-wide networks.

When Stephen Hawking worked out the basic theoretical framework for two colliding black holes [20] in the early seventies and Larry Smarr carried out early numerical simulations [29] a few years later, both involved only very small teams and generated perhaps a megabyte of data. The same problem has been studied in full 3D [11], now with a team size of perhaps 15 researchers, a growing number of involved institutes and an increase in generated data by a factor of about a million. Currently unsolved problems like the Gamma-Ray Burst riddle [28] will require still larger collaborations, even from different communities, and generate even more data.

In order for scientific collaborations to work at this scale, for the large amounts of data to be handled properly, and for the results to be reproducible, new methods of collaboration must be developed or already existing tools from other fields must be leveraged. Cactus can now use two tools from the latter class to announce information about simulations to existing Web 2.0 services, as described in the following.

Twitter's [31] main service is a message routing system that enables its users to send and read each others' updates, known as *tweets*. Tweets have to be very short (at most 140 characters in length) and can be sent and read via a wide range of devices, e.g. mobile texting, instant message, the web, or external applications.

Twitter provides an API [32] which allows the integration of Twitter with other web services and applications. One of the most important functions is the "statuses/update" API call, which is used to post a new Twitter message from the specified user. This Twitter API is used in a Cactus thorn to announce live information from a simulation 7.

Flickr [16] was launched as an image hosting website targeted at digital photographs, but short videos can be uploaded today as well. Flickr can be used at no charge with limits on the total size of images that can be uploaded (currently 100 MByte) and on the number of images which can be viewed (currently 200), along with other potential services available.

One important functionality, besides the image upload, is to be able to group images. Flickr offers a capability to group images into "Sets", and also can group different "Sets" into a "Collection". This provides a hierarchical structure for organizing simulation images.

Flickr has many features that can be taken advantage of for providing a collaborative repository for Cactus-produced images and information. All of them are accessed through a comprehensive web service API for uploading and manipulating images [17].

A new Cactus thorn uses the Flickr API to upload live images from the running simulation. Images generated by one simulation are grouped into one "Set". It is also possible to change the rendered variables, or to change the upload frequency on-line through an Cactus-internal webserver (see section 4.1).

5. Distributed Visualization

5.1. **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 this 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 transferring a video stream to the client, or running the visualization on the local client and transferring 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 have built 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.

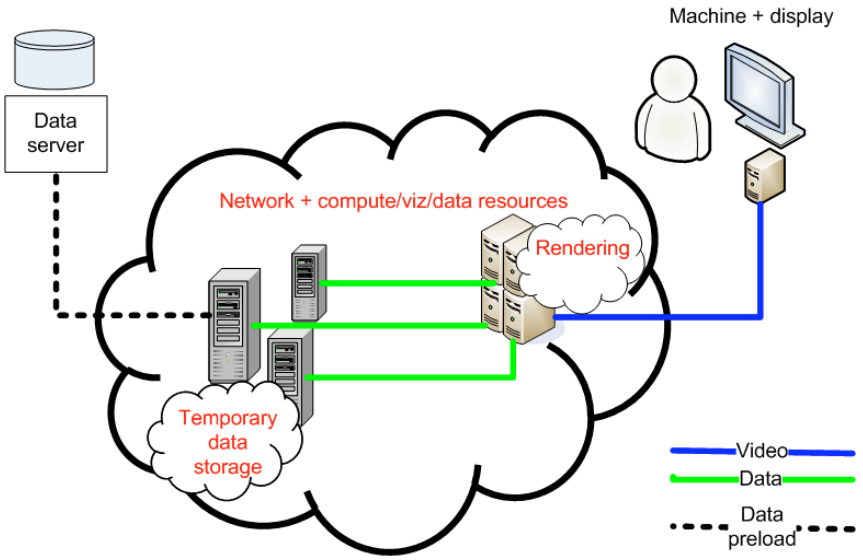

FIGURE 4. Architecture of demonstration system which involves a temporary distributed data server allocated on-demand to improve sustained data transfer rates.

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 is loaded in advance from the source on the distributed data server. The architecture of this approach is illustrated in Figure 4. Because the visualization resources are not local to the end client, a remote interaction system is necessary in order for the user to be able to connect to and steer the visualization.

5.2. **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 Mbit/s throughput on a 10 Gbit/s dedicated network connection using *TCP*. The application needs to use protocols that are suitable for the network that is utilized. Our system uses the *UDT* [19] library for wide-area transfers in order to achieve high data transmission rates. Another issue is blocking on I/O operations. Blocking I/O reduces the performance that is seen by the application, and the solution we use is based on a completely non-blocking architecture using a large number of parallel threads to keep the data flow moving.

5.3. **Parallel Rendering.** Parallel rendering on HPC or visualization clusters is utilized to visualize large datasets. In the past we have used eVolve for Equalizer (a parallel rendering framework) [15] on a rendering cluster at LSU to interactively visualize 2 GByte of visualization data at 5.5 fps, where a single machine can only visualize 256 MByte of data. *VisIt*'s [10] volume rendering plot does not support parallel hardware-accelerated volume rendering, thus largely limits its ability to accommodate native visualization of large-scale volumes. For the SCALE 2009 demonstration we have used a self-developed parallel GPU ray-tracing volume rendering implementation to interactively visualize the time-dependent numerical relativity dataset, where each timestep has a size of about 1 GByte. GPU ray-tracing does floating point compositing in a single pass using a fragment shader. The trade-off between rendering time and visual quality can be steered directly by

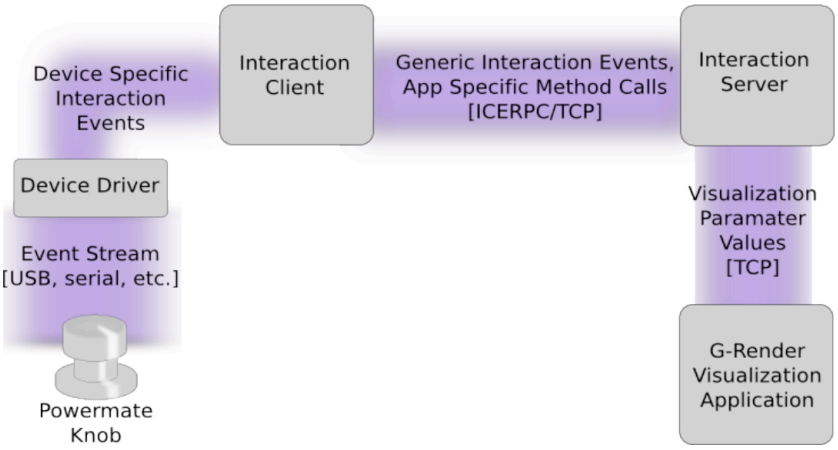

FIGURE 5. Architecture of interaction system

the user(s). Our rendering system overlaps communication with (visual) computation, in order to achieve maximum performance. Parallelism is achieved by data domain decomposition (each node renders a distinct subsection of the data), and compositing of the resulted partial view images in order to create a single view of the entire dataset.

5.4. **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 into the application workflow (see Fig. 5).

For our demonstration we used specialized interaction devices developed by the Tangible Visualization Laboratory at CCT [33] 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* (software from EVL). Our system supports various video streaming methods including SAGE, a self developed video streaming subsystem, or VNC.

6. SCALE 2009 DEMONSTRATION

The resulting Cactus application *McLachlan* used the Carpet Adaptive Mesh Refinement infrastructure to provide scalable, high order finite differencing, in this case running on 2048 cores of the Texas Advanced Computing Center (TACC) Ranger machine. The simulation ran for altogether 160 hours on Ranger, generating 42 TByte of data. Live interaction with the simulation was shown, via the application-level web interface HTTPS (Fig. 6, Section 4.1). The simulation also used new thorns co-developed by an undergraduate student at LSU to announce runtime information to Twitter and real-time images of the gravitational field to Flickr (Fig. 7).

Interactive visualization of the data produced was shown using a visualization system distributed across the Louisiana Optical Network Initiative (LONI), see Fig. 8. A data server deployed on the Eric and Louie LONI clusters cached 20 GByte of data at any time in RAM using a total of 8

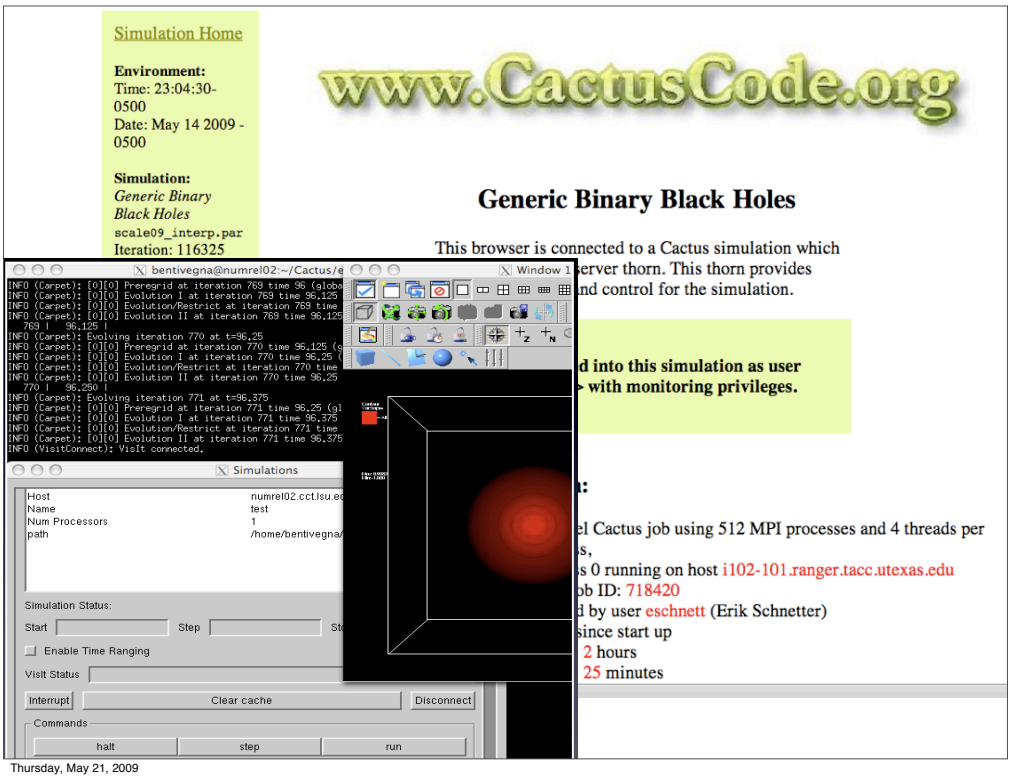

FIGURE 6. The Alpaca tools provide real-time access to simulations running on remote machines, allowing monitoring, interactive visualization, steering, and high-level debugging of large-scale simulations.

compute nodes. This data was then transferred using TCP and UDT protocols over the 10 Gbit/s LONI network to rendering nodes at a visualization cluster at LSU, with an average aggregate I/O rate of 4 Gbit/s. Loading time from the remote distributed resources was six times faster than local load from disk (2 s remote vs. 12.8 s local). Here a new parallel renderer used GPU acceleration to render images, which were then streamed using the SAGE software to the final display. VNC (Fig. 9) was used instead of SAGE in Shanghai due to local network limitations. Tangible interaction devices (also located in Shanghai) provided interaction with the renderer.

The visualization system demonstrated its capability for interactive, collaborative and scalable visualization, achieving around 5 frames per second and data transfer time of 2 seconds. This achieved the team's goal of showing how distributed systems can provide enhanced capabilities over local systems. This system is also applicable to a large number of other applications, and its demonstration was awarded first place in the IEEE SCALE 2009 Challenge. Figure 10 shows the happy winners.

7. Conclusion

Our demonstration shows the viability of the proposed approach of using the *Cactus* framework, automated code generation, and modern numerical methods to scale to a large number of processors. It also shows how distributing the visualization system into separate components increases the amount of data that can be handled, increases the speed at which the data can be visualized compared to local techniques, and improves data transfer rates and interactivity.

Our integrated approach provides a solution and a model for future scientific computing, and a large set of applications will be able to benefit from such an approach.

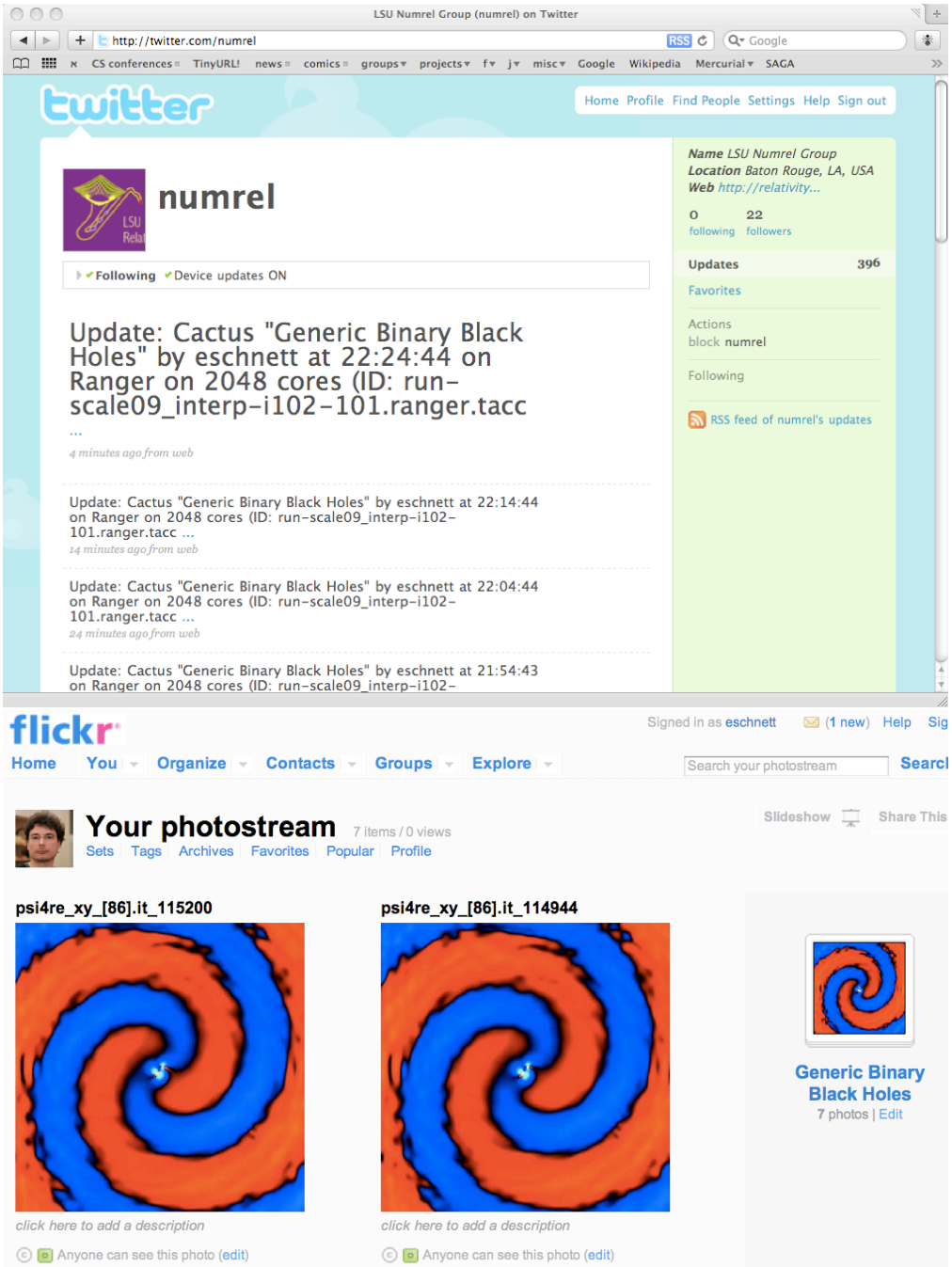

FIGURE 7. New Cactus thorns allow simulations to announce live information and images to (top) Twitter, (bottom) Flickr enabling a new mode of scientific collaboration using Web 2.0 technologies.

8. ACKNOWLEDGEMENTS

We thank the current and former members of the AEI/LSU numerical relativity group who developed the physics code that made this work possible, and the extended Cactus team for their computational infrastructure. We also thank the original authors of the Kranc code generation package for making their work publicly available. We acknowledge Jorge Ucan and Kristen Sunde for preparing the high-definition video for the SCALE 2009 Challenge, and Debra Waters for her

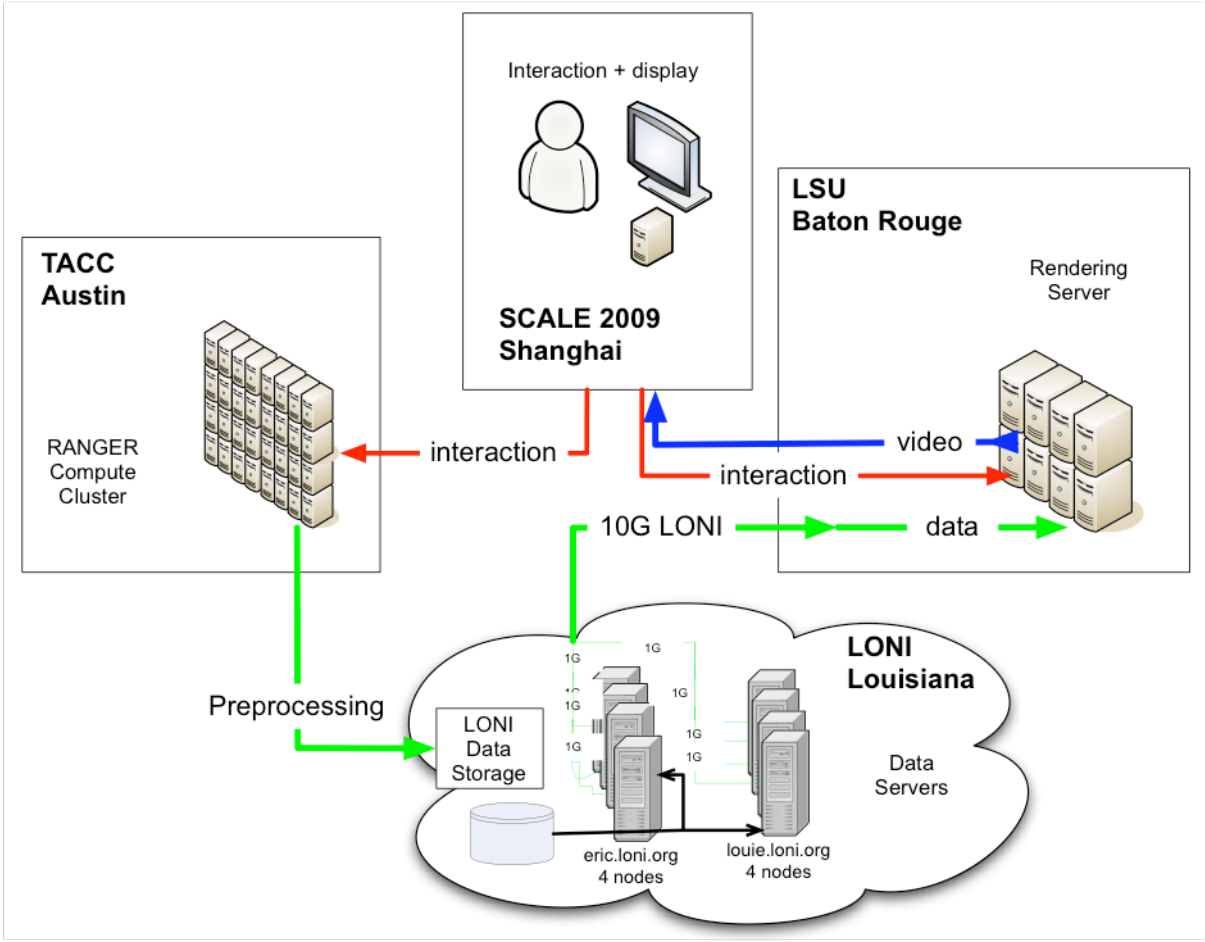

FIGURE 8. Set up of the SCALE 2009 demonstration that involved the resources of the NSF TeraGrid, the Louisiana Optical Network Initiative (LONI) and the Center for Computation & Technology.

(unfortunately ultimately unsuccessful) attempts to secure Chinese visas for our team. The demonstration would not have been possible without the help of our colleagues at TACC and LONI.

The work forming the SCALE 2009 demonstration was funded by the National Science Foundation (Alpaca #0721915, Blue Waters #0725070, Viz Tangibles #0521559, XiRel #0701566, Louisiana RII *CyberTools* #0701491) and the Center for Computation & Technology at LSU. The simulations and benchmarks were performed on Queen Bee at LONI under allocations `loni_cactus03` and `loni_numrel03`, and on Ranger at TACC under the NSF TeraGrid allocation TG-MCA02N014. The distributed visualization development was supported by the NSF TeraGrid allocation TG-CCR080027T.

REFERENCES

[1] Cactus Computational Toolkit home page, URL http://www.cactuscode.org/.
[2] Mesh refinement with Carpet, URL http://www.carpetcode.org/.
[3] LIGO: Laser Interferometer Gravitational Wave Observatory, URL http://www.ligo.caltech.edu/.
[4] GEO 600, URL http://www.geo600.uni-hannover.de/.
[5] VIRGO, URL http://www.virgo.infn.it/.
[6] Cactus Benchmark Results and Papers, URL http://www.cactuscode.org/Benchmarks.
[7] Kranc: Automated Code Generation, URL http://numrel.aei.mpg.de/Research/Kranc/.

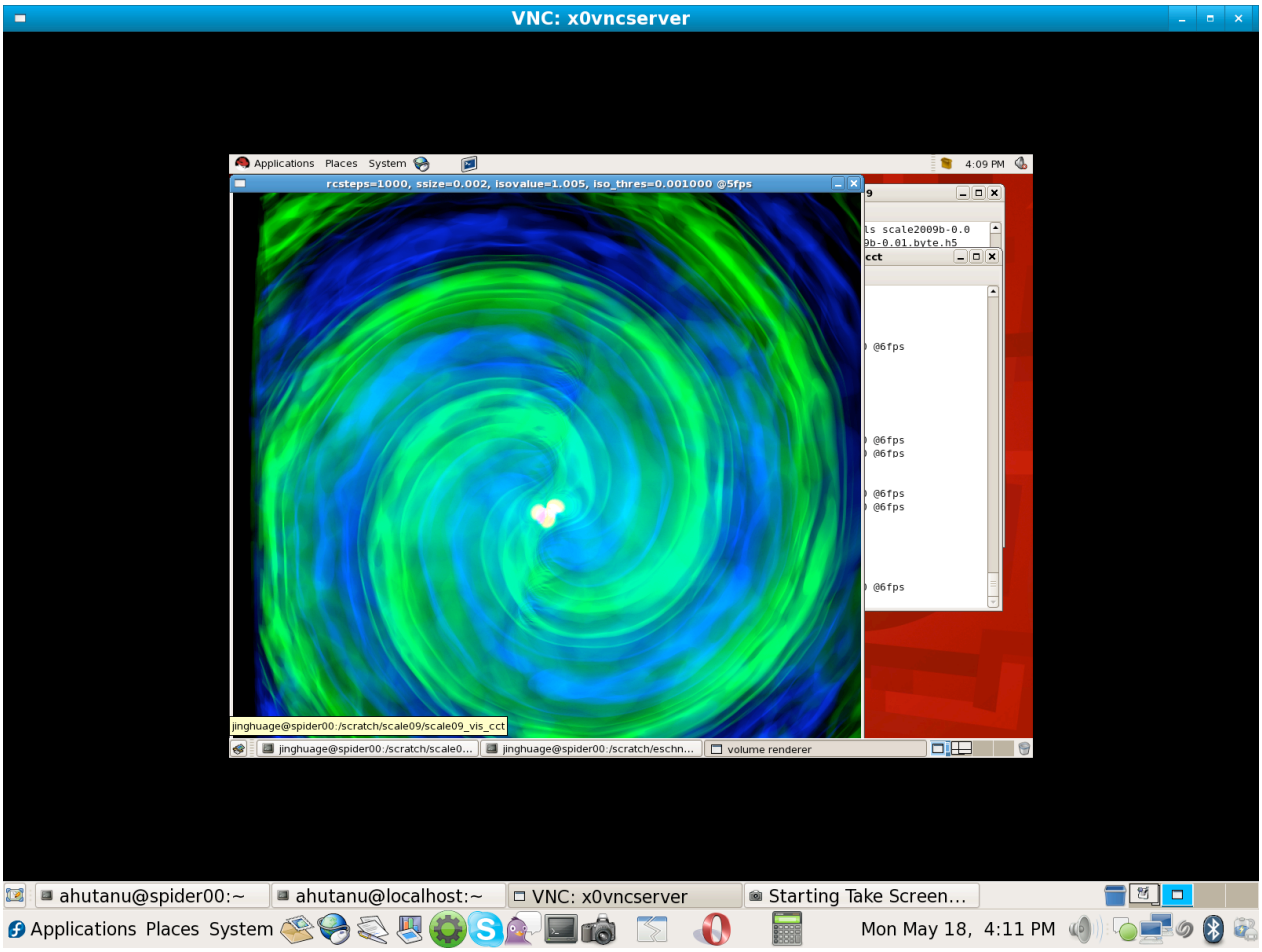

FIGURE 9. Visualization client on the end display showing rendering of gravitational waves emitted from the inspiral collision of two black holes.

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

[9] XiRel: Next Generation Infrastructure for Numerical Relativity, URL http://www.cct.lsu.edu/xirel/.

[10] VisIt Visualization Tool, URL https://wci.llnl.gov/codes/visit/.

[11] Miguel Alcubierre, W. Benger, B. Brügmann, G. Lanfermann, L. Nerger, E. Seidel, and R. Takahashi, *3D Grazing Collision of Two Black Holes*, Phys. Rev. Lett. **87** (2001), 271103, eprint gr-qc/0012079.

[12] Alpaca: Cactus tools for Application-Level Profiling and Correctness Analysis, URL http://www.cct.lsu.edu/~eschnett/Alpaca/.

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

[14] David Brown, Peter Diener, Olivier Sarbach, Erik Schnetter, and Manuel Tiglio, *Turduckening black holes: an analytical and computational study*, Phys. Rev. D **79** (2009), 044023, eprint arXiv:0809.3533 [gr-qc], URL http://arxiv.org/abs/0809.3533.

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

[16] Flickr, URL http://www.flickr.com/.

[17] Flickr API Web page, URL http://www.flickr.com/services/api/.

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

[19] Yunhong Gu and Robert L. Grossman, *Udt: Udp-based data transfer for high-speed wide area networks*, Comput. Networks **51** (2007), no. 7, 1777–1799.

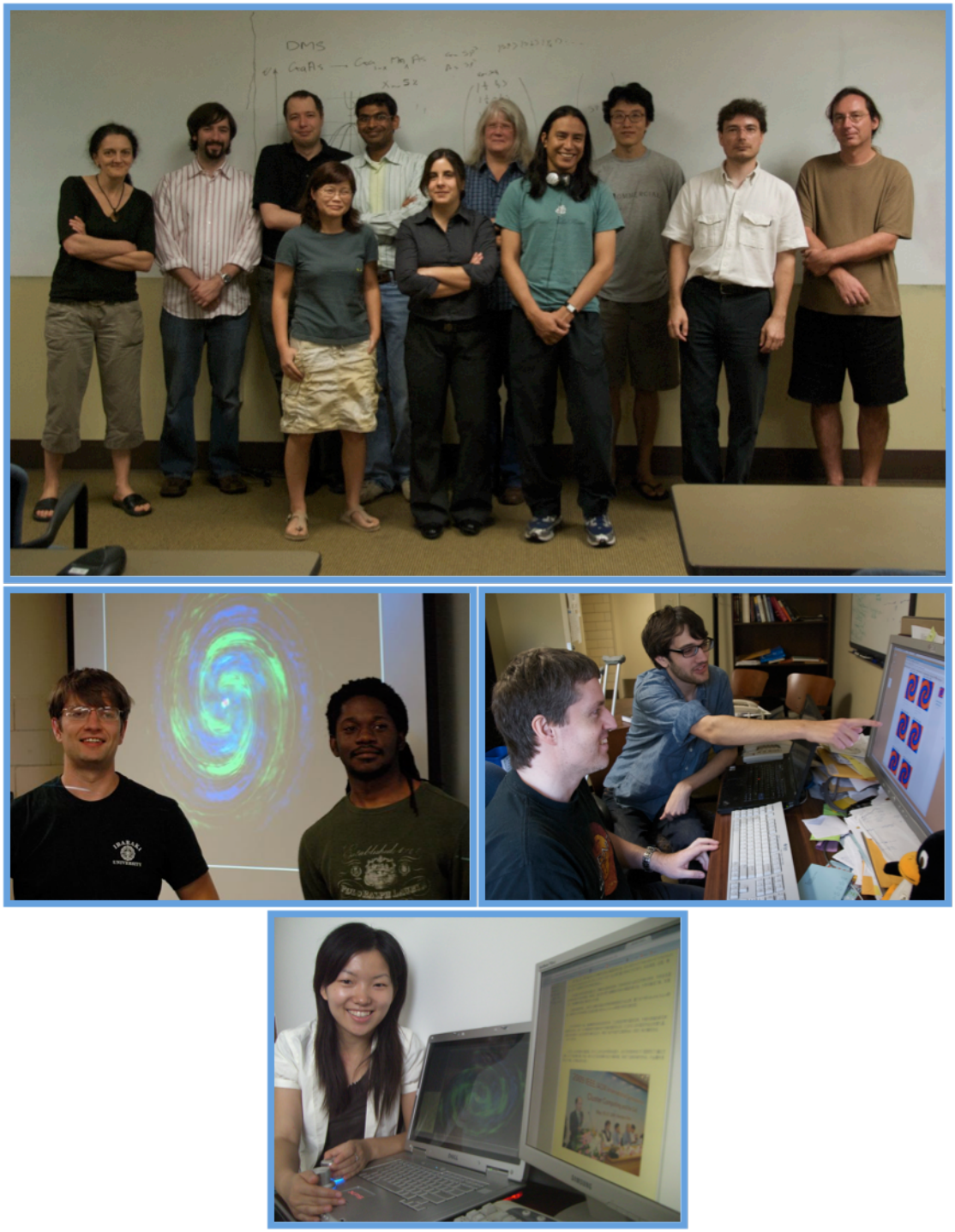

FIGURE 10. The LSU SCALE 2009 Team: Left to right, (top) Gabrielle, Adam, Andrei, Jinghua, Ravi, Eloisa, Debra, Jorge, Jian, Erik, Werner, (middle left), Oleg, Cornelius, (middle right) Peter, Alex, (bottom) Kexi.

[20] S. W. Hawking, *Black holes in general relativity*, Comm. Math. Phys. **25** (1972), 152.
[21] Sascha Husa, Ian Hinder, and Christiane Lechner, *Kranc: a Mathematica application to generate numerical codes for tensorial evolution equations*, Comput. Phys. Comm. **174** (2006), 983–1004, eprint gr-qc/0404023.
[22] Christiane Lechner, Dana Alic, and Sascha Husa, *From tensor equations to numerical code — computer algebra tools for numerical relativity*, SYNASC 2004 — 6th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 2004, eprint cs.SC/0411063, URL http://arxiv.org/abs/cs.SC/0411063.
[23] Peter Mészáros, *Gamma-ray bursts*, Rep. Prog. Phys. **69** (2006), 2259–2321, eprint astro-ph/0605208.
[24] Christian D. Ott, Erik Schnetter, Gabrielle Allen, Edward Seidel, Jian Tao, and Burkhard Zink, *A case study for petascale applications in astrophysics: Simulating Gamma-Ray Bursts*, Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities (Baton Rouge, Louisiana), ACM International Conference Proceeding Series, no. 18, ACM, 2008, URL http://doi.acm.org/10.1145/1341811.1341831.
[25] E. Schnetter, S. H. Hawley, and I. Hawke, *Evolutions in 3D numerical relativity using fixed mesh refinement*, Class. Quantum Grav. **21** (2004), no. 6, 1465–1488, gr-qc/0310042, URL http://arxiv.org/pdf/gr-qc/0310042.
[26] Erik Schnetter, Gabrielle Allen, Tom Goodale, and Mayank Tyagi, *Alpaca: Cactus tools for application level performance and correctness analysis*, Tech. Report CCT-TR-2008-2, Louisiana State University, 2008, URL http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-2.
[27] Erik Schnetter, Peter Diener, Nils Dorband, and Manuel Tiglio, *A multi-block infrastructure for three-dimensional time-dependent numerical relativity*, Class. Quantum Grav. **23** (2006), S553–S578, eprint gr-qc/0602104, URL http://stacks.iop.org/CQG/23/S553.
[28] Erik Schnetter, Christian D. Ott, Gabrielle Allen, Peter Diener, Tom Goodale, Thomas Radke, Edward Seidel, and John Shalf, *Cactus Framework: Black holes to gamma ray bursts*, Petascale Computing: Algorithms and Applications (David A. Bader, ed.), Chapman & Hall/CRC Computational Science Series, 2008, eprint arXiv:0707.1607 [cs.DC], URL http://arxiv.org/abs/0707.1607.
[29] L. Smarr, *Spacetimes generated by computers: Black holes with gravitational radiation*, Ann. N. Y. Acad. Sci. **302** (1977), 569–604.
[30] Jian Tao, Gabrielle Allen, Ian Hinder, Erik Schnetter, and Yosef Zlochower, *XiRel: Standard benchmarks for numerical relativity codes using Cactus and Carpet*, Tech. Report CCT-TR-2008-5, Louisiana State University, 2008, URL http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-5.
[31] Twitter, URL http://www.twitter.com/.
[32] Twitter Application Programming Interface, URL http://apiwiki.twitter.com/Twitter-API-Documentation.
[33] Brygg Ullmer, Rajesh Sankaran, Srikanth Jandhyala, Blake Tregre, Cornelius Toole, Karun Kallakuri, Christopher Laan, Matthew Hess, Farid Harhad, Urban Wiggins, and Shining Sun, *Tangible menus and interaction trays: core tangibles for common physical/digital activities*, TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction (New York, NY, USA), ACM, 2008, pp. 209–212.