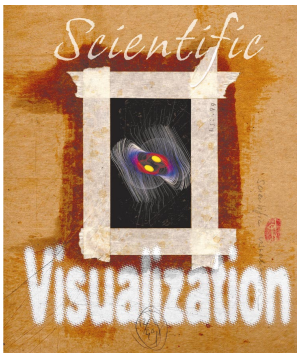
Solving Einstein's Equations on Supercomputers



Globally distributed scientific teams, linked to the most powerful supercomputers, are running visual simulations of Einstein's equations on the gravitational effects of colliding black holes.

Gabrielle Allen
Tom Goodale
Gerd Lanfermann
Thomas Radke
Edward Seidel
Max Planck Institute for Gravitational Physics

Werner Benger
Hans-Christian Hege
Andre Merzky
Konrad Zuse Center for Information Technology, Berlin

Joan Massó
University of the Balearic Islands

John Shalf
National Center for Supercomputing Applications

In 1916, Albert Einstein published his famous general theory of relativity, which contains the rules of gravity and provides the basis for modern theories of astrophysics and cosmology. It describes phenomena on all scales in the universe, from compact objects such as black holes, neutron stars, and supernovae to large-scale structure formation such as that involved in creating the distribution of clusters of galaxies. For many years, physicists, astrophysicists, and mathematicians have striven to develop techniques for unlocking the secrets contained in Einstein's theory of gravity; more recently, computational-science research groups have added their expertise to the endeavor.

Those who study these objects face a daunting challenge: The equations are among the most complicated seen in mathematical physics. Together, they form a set of 10 coupled, nonlinear, hyperbolic-elliptic partial differential equations that contain many thousands of terms. Despite more than 80 years of intense analytical study, these equations have yielded only a handful of special solutions relevant for astrophysics and cosmology, giving only tantalizing snapshots of the dynamics that occur in our universe.

Scientists have gradually realized that numerical studies of Einstein's equations will play an essential role in uncovering the full picture. Realizing that this work requires new tools, developers have created computer programs to investigate gravity, giving birth to the field of *numerical relativity*. Progress here has been initially slow, due to the complexity of the equations, the lack of computer resources, and the wide variety of numerical algorithms, computational techniques, and underlying physics needed to solve these equations.

A realistic 3D simulation based on the full Einstein equations is an enormous task: A single simulation of coalescing neutron stars or black holes would require more than a teraflop per second for reasonable performance, and a terabyte of memory. Even if the computers needed to perform such work existed, we must still increase our understanding of the underlying physics, mathematics, numerical algorithms, high-speed networking, and computational science. These cross-disciplinary requirements ensure that no single group of researchers has the expertise to solve the full problem. Further, many traditional relativists trained in more mathematical aspects of the problem lack expertise in computational science, as do many astrophysicists trained in the techniques of numerical relativity. We need an effective community framework for bringing together experts from disparate disciplines and geographically distributed locations that will enable mathematical relativists, astrophysicists, and computer scientists to work together on this immensely difficult problem.

Because the underlying scientific project provides such a demanding and rich system for computational science, techniques developed to solve Einstein's equations will apply immediately to a large family of scientific and engineering problems. We have developed a collaborative computational framework that allows remote monitoring and visualization of simulations, at the center of which lies a community code called Cactus (see the "Cactus's Modular, Extensible Code" sidebar). Many researchers in the general scientific computing community have already adopted Cactus, as have numerical relativists and astrophysicists. This past June, an international team of researchers at various sites around the world ran, over several weeks, some of the largest such simulations in numerical relativity yet undertaken, using a 256-processor SGI Origin 2000 supercomputer at NCSA (see http://access.ncsa.uiuc.edu/Features/o2krun/ for a description).

VISUALIZING RELATIVITY

The scientific visualization of Einstein's equations as numerical simulations presents us with many chal-

0018-9162/99/$10.00 © 1999 IEEE

lenges. The fundamental objects that describe the gravitational field form components of high-rank, four-dimensional tensors. Not only do the tensor components depend on the coordinates in which they are written, and thus have no direct physical meaning themselves, they are also numerous: The quantity describing space-time curvature has 256 components. Further, interpreting these components is not straightforward: Pure gravity is nothing more than curved "vacuum," or empty space without matter. Standard intuition in terms of, say, fluid quantities such as density or velocity, do not carry over easily to relativity. Thus, new paradigms for visualization must be developed. The many quantities needed to describe the gravitational field require an enormous volume of data that must be output, visualized, and analyzed.

Figure 1 shows one of the first detailed simulations depicting a full 3D merger of two spinning and orbiting black holes of unequal mass. (The "Black Holes and Gravitational Waves" sidebar describes black holes in greater detail.) We define the black holes by the surfaces of their respective horizons—in our case, and for rather complex reasons, we compute the *apparent horizon*. The horizon forms the critical 2D surface that separates everything trapped forever inside from everything that could possibly escape. This surface responds to local variations in the gravitational field by distorting and oscillating. In Figure 1,

Cactus's Modular, Extensible Code

Our experiences with highly complex numerical relativity simulation codes have led us to create a new paradigm for developing and reusing numerical software in a collaborative, portable environment. Cactus embodies this new approach, which can be applied to all areas of computational science and engineering. Cactus lets you extend traditional single-processor codes to full-blown parallel applications that can run on all supercomputers, but that can still be developed on a laptop. Cactus also provides access to a myriad of computational tools, such as advanced numerical techniques, adaptive mesh refinement, parallel I/O, live and remote visualization, and remote steering. We're also working to integrate computational technologies such as metacomputing toolkits like Globus, and performance and steering tools like Autopilot, described elsewhere in this issue. The various computational layers and functionalities, as well as the different scientific modules, can be selected at runtime via parameter choices.

Design principles

The name "Cactus" comes from the initial design principle of a module set—the thorns—that could be plugged easily into a basic kernel, the flesh. The flesh has evolved into a metacode with its own language to enable dynamic assembly of many different application codes from a set of thorns. The flesh controls how the thorns work together, coordinating dataflow between them and deciding when any routine should be executed.

Developers build Cactus applications dynamically, using a metacode with a new object-oriented language that describes how the different pieces of codes written in any common computational language—such as C, C++, Fortan 77, and Fortran 90—interweave.

Our desire to provide a solid framework for collaborative code use and development heavily influenced Cactus's design principles. The modular approach enables both scientists and computational scientists to design, use, and add thorns that work together, through the flesh API. Cactus lets you apply collaborative considerations not only to the code itself, but to the wider working environment. Thorn mailing lists, a versioning system, test-suite checking technology, and an open bug-tracking system help connect and inform users. To better address these collaborative needs, we made the flesh and computational tool thorns open source so that all developers can access them. In this sense, Cactus represents a huge collaboration between all users in all fields.

Cactus synergies

It's not always clear in emerging research areas, such as numerical relativity, what techniques will be best suited to solving a given scientific problem. Further, both the software and hardware used to run advanced scientific simulations change rapidly. These factors, combined with the complexity of the work undertaken, result in huge simulations that require intensive and often widely distributed collaboration.

To support such work, we needed an environment such as that which Cactus provides: a flexible, extensible, and portable computational infrastructure into which different components, from different communities, could be plugged. Although costly and labor-intensive, implementing such a system provides unique synergies. For example, while a physicist may develop a new formulation of Einstein's equations, a numerical analyst may have a more efficient algorithm for solving a required elliptic equation, a computer scientist may develop a more efficient parallel I/O layer, a software team may develop a new paradigm for parallel programming, a hardware vendor may develop a more efficient parallel-computer architecture, and so on. All such developments take place independently on relatively short time scales, and all may significantly affect the scientific simulation.

Without the ability to effortlessly connect such components and move about in "architecture space," projects can be delayed repeatedly for component retooling. Cactus allows the computational application scientist, such as a physicist or engineer, to assimilate the most appropriate available components into a solution tailored specifically to the problem at hand, and to execute a simulation using the best computer architecture available. You can find out more about Cactus at http://www.cactuscode.org.

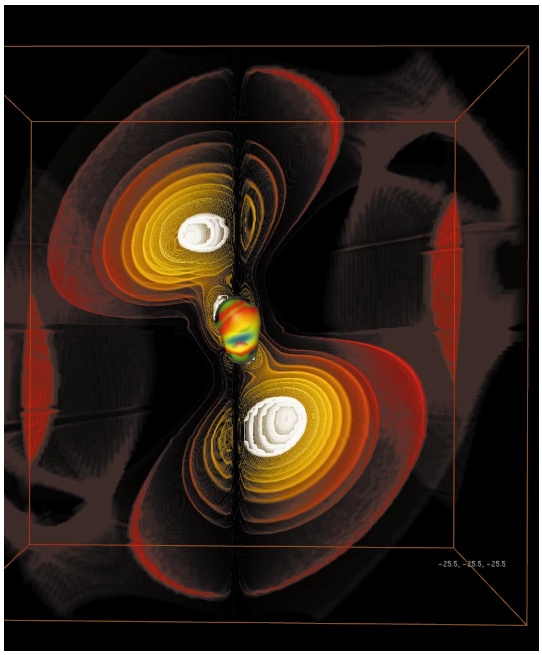


Figure 1. Gravitational waves from a full 3D grazing merger of two black holes. The image shows the objects immediately after the coalescence, when the two holes (seen just inside) have merged to form a single, larger hole at the center. The distortions of the horizon (the Gaussian curvature of the surface) appear as a color map, while the resulting burst of gravitational waves (the even-parity polarization or real part of Ψ_4) appears in red and yellow.

the two black holes have just merged in a grazing collision; the merged single black hole horizon, with its two black holes inside, is at the center. The collision emits a burst of gravitational waves that radiates away toward infinity.

The content of gravitational radiation in the system can be calculated from the 10 independent components of the so-called Riemann curvature tensor (in turn represented by five complex numbers $\Psi_0, \ldots \Psi_4$). For simplicity, we refer loosely to the real and imaginary parts of the quantity as the two independent physical polarizations of the gravitational waves. In Figure 1, the real part of Ψ_4 shows how some energy from the cataclysm radiates away. (Most of the visualizations in this article, as well as the cover image, were produced with Amira, described in the "Amira" sidebar.)

To fully evaluate the science in this complex simulation, we must examine all aspects of the gravitational waves and other quantities. Figure 2 shows the other polarization of the wave field, the imaginary part of Ψ_4. This additional component of the gravitational radiation does not occur in previously

Black Holes and Gravitational Waves

Among the most exotic objects believed to populate the universe, black holes have not been directly observed yet. They are typically expected to form when a very massive star exhausts its nuclear fuel and then collapses in on itself. This process may lead to a supernova explosion, the creation of a compact object called a neutron star, or both phenomena. But if the star is massive enough, its stronger self-gravity will pull it into a runaway collapse that continues until all the original star's mass and energy collapse to a point of infinite density. This event leaves behind a *black hole* in space. Travel too close to the black hole and you reach the point of no return: No signal, including light, can escape if caught within an imaginary surface called the *horizon*.

Many stars in the universe pair into binary systems: two stars that orbit each other. Some fraction of these binaries will be very massive stars; a fraction of these massive stars will ultimately collapse and form black holes; a yet smaller fraction of black holes will form binary black-hole pairs that orbit each other. These exotic orbiting holes of intense gravity would go around each other forever, but another novel prediction of Einstein's theory, gravitational radiation, slowly but relentlessly drains the system of energy and angular momentum. The gravitational waves form ripples in space-time itself, traveling at the speed of light. Their strength is related to the strength of the gravitational field, its time rate of change, and its nonsphericity. Hence, as the binary holes orbit, they emit a steady stream of waves that cause their orbit to decay. After eons pass, the orbit decays almost completely, and the holes approach each other in a violent final plunge, as shown in Figure A, spiraling wildly together and creating the larger, rapidly spinning black hole and a burst of radiation. A worldwide effort is under way to develop the computing capability and algorithms needed to solve those of Einstein's equations that describe this process. Achieving this goal will help us interpret and detect such events using the new gravitational-wave observatories being constructed on several continents.

Einstein's theory predicts that gravitational waves, although normally weak, may—if they are strong enough and exposed to certain conditions—collapse under their own self-gravity and form black holes! Such a scenario seems unlikely in normal astrophysical processes, but can be simulated on a computer as a way to probe Einstein's theory itself. In the main text we show examples of such calculations.

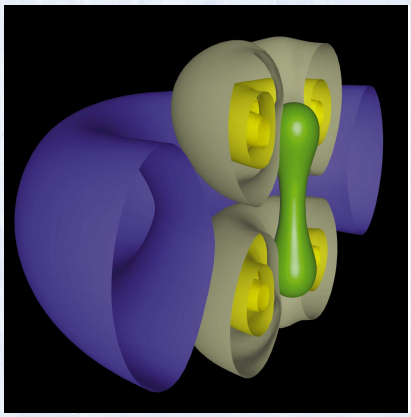


Figure A. Anatomy of a black-hole collision. The visualization depicts a 3D-simulated head-on collision between equal-mass black holes. The (event) horizon surface appears green, while the isosurfaces of the gravitational waves emitted appear as yellow and purple surfaces.

Amira

Amira is a 3D visualization and modeling system developed at the Konrad Zuse Center for Information Technology to support interactive applications in science, medicine, and engineering. Designed for nonexperts, Amira balances complexity with ease of use and should be accessible to a broad range of users. Complex tasks require more familiarity but can be achieved by experimenting with the interactive facilities.

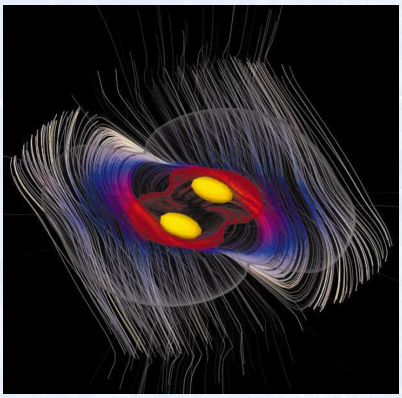


Figure B. Collision of neutron stars showing isolevels of matter density and illuminated integral lines of the momentum vector field. The field lines and proper illumination greatly improve spatial perception of complex scenes.

Amira does not adhere to the dataflow concept; instead, the user loads data and simply activates appropriate display or computational modules via context-sensitive menus. Instantiated objects and their mutual dependencies appear automatically as a graphical network. By modifying this representation and selecting suitable parameters, users can control the visualization process with minimal interaction.

Amira displays its visual results in one or several graphics windows that allow users to interact directly, in three dimensions, with the depicted data objects. Amira can also arbitrarily combine different display techniques in a single window. The system's 3D graphics have been implemented using Open Inventor and OpenGL. Graphics hardware, if available, can be used to display very large data sets at interactive speed. Although typically used interactively, Amira's functions can also be scripted via a Tcl-command interface.

The system currently comprises about 25 different data-object types and more than 100 modules that can be dynamically loaded at runtime. Visualization techniques include hardware-accelerated volume rendering for display of scalar fields, and advanced vector field visualization algorithms, like the illuminated streamlines shown in Figure B. (For more on illuminated streamlines, see http://www.zib.de/Visual/projects/vector.) The illuminated streamlines greatly improve spatial perception of complex 3D vector fields. Amira intrinsically supports various grid types, like hexahedral, curvilinear, and unstructured tetrahedral grids. AMR data types for numerical relativity data, adapted to those in Cactus, are currently under development. (For more on Amira, see http://www.amiravis.com.)

computed axisymmetric collisions and shows interesting phenomena uniquely associated with 3D calculations.

Studying the horizon surface of the black hole by itself provides further insights. The coordinate location or coordinate shape of the horizon does not reflect its physical properties; those properties change with different coordinate choices. However, the (Gaussian) curvature of the surface is an intrinsic property of the horizon's true geometry and thus does reflect its physical properties. We can compute this quantity and color-map it to the surface of the horizon to understand how deformed it becomes during the collision. Figure 3 shows the results.

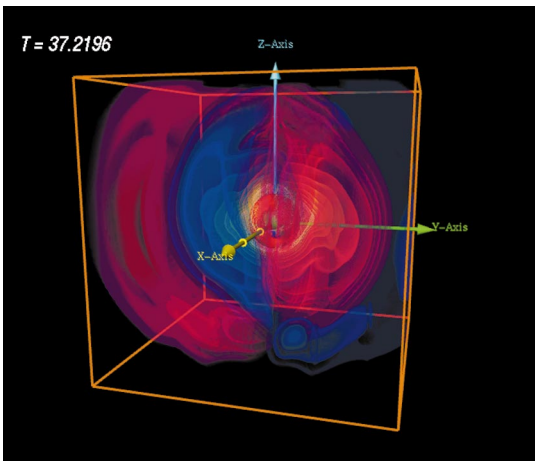


Figure 2. Additional polarization of gravitational waves (imaginary part of Ψ_4) from a 3D merging collision of two black holes. The merged single black-hole horizon can just be seen through the cloud of radiation emitted in the process.

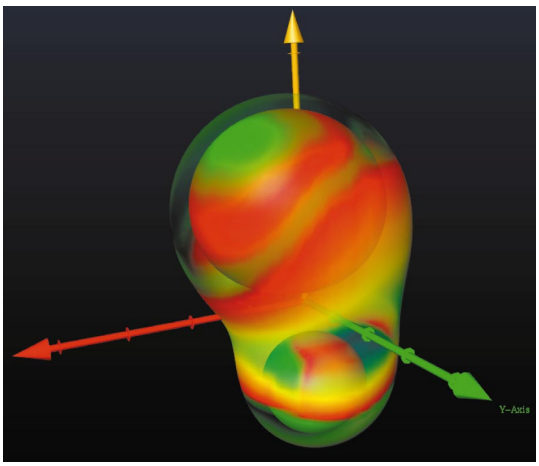


Figure 3. Close-up of the horizon—with Gaussian curvature to show the distorted nature of the surface—of a black hole formed by the collision of two black holes. The two individual horizon surfaces can just be seen inside the larger horizon formed during the collision process.

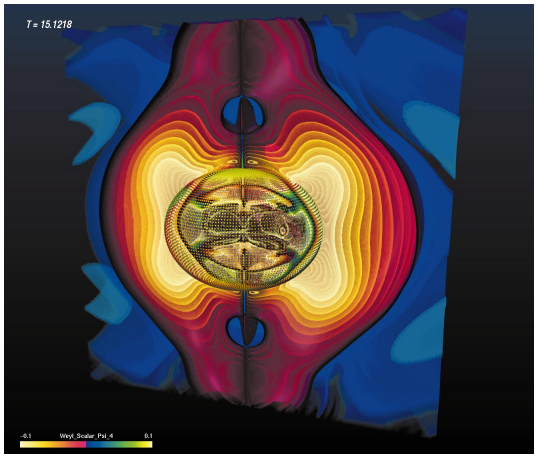


Figure 4. Gravitational waves and horizon of a newly formed black hole, caused by massive collapse of a strong gravitational wave on itself. The dotted surface shows the horizon, where green colors indicate high curvature and yellow means zero curvature. The highest curvature indicates the largest gravitational radiation. The "distortion" (the non-sphericity) of the black hole radiates away over time, in accordance with mathematical theorems about black holes.

The fabric of space-time may also change depending on the initial concentration of pure gravitational waves. A weak concentration lets the evolving gravitational radiation just "flow away," like shallow ripples on a pond. However, a large-enough energy concentration in the gravitational wave can implode violently to create a black hole with, again, small ripples flowing away. At this point, comparing the gravitational waves' field indicators and the oscillations of the Gaussian curvature of the newly formed black hole can be especially informative. For the simulation presented here, the black hole assumes for its final state a static spherical "Schwarzschild black hole," but the initial black hole that forms will be highly distorted. These distortions radiate away during the evolution. We thus expect to see gravitational radiation correlated with the structures of the Gaussian curvature. By displaying the positive and negative values of the Ψ_4 scalar in full 3D volume, using special volume rendering techniques and overlaying the apparent horizon, we can make this correlation visible, as Figures 4 and 5 show.

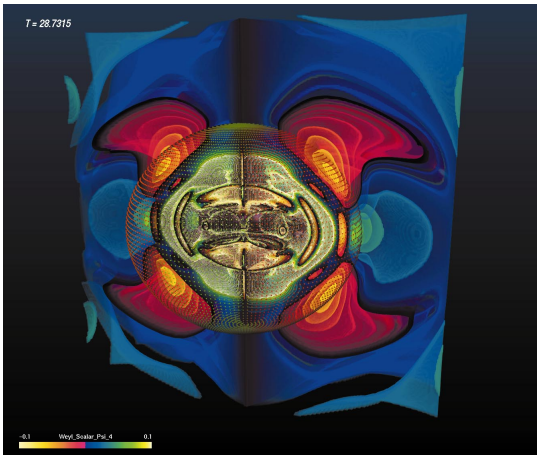


Figure 5. Horizon of a gravitational wave that implodes to form a black hole, with leftover waves escaping, shown at a later time in the same evolution presented in Figure 4. We see that the horizon curvature is affected by, and correlated with, the evolving gravitational wave.

TOWARD SCIENTIFIC PORTALS FOR LARGE-SCALE COLLABORATIVE SIMULATION

Keeping such a widely distributed group of researchers working together requires infrastructure support for collaboration and resource sharing. This emerging world of metacomputing, now referred to as the Grid, brings together extraordinary computing resources, wherever they may be located in the world, to attack extraordinary scientific problems that cannot be solved by other means.

Although distributed resources offer several advantages, there are downsides as well. Even something as simple as running a supercomputing job becomes a huge task when you and the supercomputer are on different continents. The enormous terabyte data sets produced by these simulations tax bandwidth limits particularly. Even with the best available international networking resources, downloading the data from the simulation run may take longer than it took to run the simulation itself. To load terabyte data sets for interactive visual analysis, the workstation assigned that task must be nearly as powerful as the supercomputer performing the simulation. These delays and resource limitations can be costly to research when a simulation goes awry or researchers select a nonoptimal set of simulation parameters. These problems have motivated many of our remote monitoring and steering efforts—an endeavor greatly simplified by Cactus's modular architecture.

From a visualization standpoint, traditional researchers remain essentially blind while the supercomputer runs a simulation. This limitation becomes intolerable when we consider the hours or days that large-scale problems run and the exceedingly expensive computational resources these problems consume. If we equate these limitations to cooking a roast without a timer or thermometer, computational monitoring provides the "window into the oven" we need to control these simulations. Our own efforts in this area have focused on parallel geometry extraction inline with the simulation code, using, initially, the Cactus isosurfacer thorn. This approach allowed us to remotely visualize a large ($300 \times 300 \times 300$) data set running on a T3E in Germany using an Electronic Visualization Laboratory (EVL) ImmersaDesk VR system in San Jose, Calif., over a rather meager 5-megabits-per-second network connection.

Remote and distributed visualization simply decouples the parallel visualization pipeline from the simu-

lation code. The enormous data sizes generated by these simulation codes overwhelm even the largest of our workstations. The most desirable way around this challenge is to put the supercomputer that generated the data set to work as a "visualization supercomputer." Doing so

- takes advantage of the data locality as well as the supercomputer's disk I/O bandwidth and memory capacity to speed the loading of these data sets, and
- lets us bring to bear the parallel processing capabilities of these remote supercomputing resources to accelerate expensive visualization algorithms.

The data transport to the client must employ compression techniques (lossless and lossy) for geometries, textures, and images. Further, it must adapt to bandwidth and client capabilities by offering, for example, progressive transmission and multilevel resolution. To this end, we are experimenting with adding parallel computing and offscreen rendering extensions to popular open-source visualization packages like VTK. We're also adding the protocols needed to convey remote-visualization control information for the DFN Gigabit Testbed project.

Naturally, we would like to do more than watch our code "roasting in the oven." Computational steering takes monitoring one step further by allowing you to manipulate the simulation as it runs. This can be useful when some event in the simulation requires a decision about the physical or computational parameters during runtime. For example, adaptive mesh refinement may exhaust a machine's memory, and the researcher will be asked to choose between ignoring less interesting portions of the simulation or bringing online more computing resources to handle the additional load.

The Cactus "http" thorn turns the simulation code into a Web server that allows anyone to monitor all aspects of the running code using a Web browser. Its steering capabilities are currently limited to login-and-password authenticated code access to kill the simulation—so that we can, metaphorically, stop the oven when we see smoke. In the future, the steering capabilities will be expanded to allow manipulation of many parameters. We have also begun to work with AutoPilot, which will let us use various sensors and actuators for steering and for visualization of the code performance on distributed computers (see article by Eric Shaffer and colleagues on pages 44-51).

Perhaps the supercomputer visualization community has restricted itself too much by considering theory, computation, and visualization as separate entities. Doing so keeps everyone's work area compartmentalized and discourages close interactions between each stage in the pipeline. Portals may let us do away with this partitioning and offer more open access to each piece of this scientific puzzle. A portal is a scientific workbench widely accessible through networking and optimized for a particular set of problems or a problem-solving paradigm.

We are just beginning work on a Collaborative Science Portal that will conveniently tie together interfaces for composing codes from community-developed repositories. This will provide a single interface to assemble a simulation code and resources, control the running code, and visualize the results, either while it is running or afterward. We need to incorporate new collaborative technologies such as CavernSoft that provide an infrastructure that supports multiway collaborative visual data analysis. The Cactus code's modular nature makes possible this level of service integration and lets us meet the needs of our worldwide distributed band of researchers. We firmly believe that most major scientific codes will be viewed through some form of portal interface and that we will find visualization systems increasingly embedded in a ubiquitous service-based Grid fabric. ❖

Acknowledgments

We thank Ralf Kähler, Hermann Lederer, Jason Novotny, Manuel Panea, Warren Smith, Detlev Stalling, Ulrich Schwenn, Meghan Thornton, Paul Walker, Malte Zöckler, and Ian Foster for many important contributions to the computational science presented in this article, and Miguel Alcubierre, Steve Brandt, Bernd Brügmann, Lars Nerger, Wai-Mo Suen, and Ryoji Takahashi for many contributions to the science presented here. This work was supported by NCSA, the NSF, Max Planck Institute for Gravitational Physics, Konrad Zuse Center for Information Technology, and the DFN-Verein. Our international networking exploits would not have been possible without assistance and applications support from the Star Tap high-performance network access point.

Gabrielle Allen is a Cactus developer at the Max Planck Institute for Gravitational Physics. She received a PhD in astrophysics from the University of Wales and has worked in the field of numerical relativity.

Tom Goodale is a PhD student at the Max Planck Institute for Gravitational Physics, where he is a Cactus developer. His research interests include general relativistic hydrodynamics, parallel computational physics, and adaptive mesh refinement. He received an MSc in computational fluid dynamics

from Cranfield University after working as a software engineer developing simulation code for the oil and gas industry.

Gerd Lanfermann is a PhD student at the Max Planck Institute for Gravitational Physics, where he is a member of the Cactus Development Team. He received a diploma degree in theoretical physics from the Free University, Berlin. His research interests include parallel and distributed computing.

Thomas Radke is staff member at the Max Planck Institute for Gravitational Physics. He is involved in the TIKSL project, a German gigabit-testbed research project on the teleimmersion of black hole collisions. His research interests include multithreading, message passing programming, and Linux kernel development. He received a diploma in information technology from the University of Technology, Chemnitz.

Edward Seidel is a professor at the Max Planck Institute for Gravitational Physics, holds adjunct positions in the Astronomy and Physics Departments at the University of Illinois, Urbana-Champaign, and is a senior research scientist at the National Center for Supercomputing Applications. His research interests include general relativity, astrophysics, and computational science. He received a PhD in relativistic astrophysics from Yale.

Werner Benger is a researcher at the Konrad Zuse Center for Information Technology in cooperation with the Max Planck Institute for Gravitational Physics. He works on visualization techniques for general relativity, including tensorfield visualization and relativistic ray tracing. He received a Magister degree from the University of Innsbruck, Austria. His images and movies have appeared in various journals and television broadcasts.

Hans-Christian Hege is the director of the Scientific Visualization Department at the Konrad Zuse Center for Information Technology and is the managing director of Indeed-Visual Concepts, a company specializing in visualization software and hardware. He studied physics at Free University, Berlin, and was a research assistant in quantum field theory and numerical physics.

Andre Merzky is a researcher at the Konrad Zuse Center for Information Technology. He studied elementary particle physics in Leipzig, Edinburgh, and Berlin. His research interests include distributed computing, remote steering, and visualization.

Joan Massó is an associate professor in the Department of Physics at the University of the Balearic Islands, where he also is the head of supercomputing. He is affiliated with the National Center for Supercomputing Applications and with the Department of Physics at Washington University. His research interests include numerical relativity, scientific and high-performance computing, and software development. He received a PhD in theoretical physics from the University of the Balearic Islands.

John Shalf works for the National Center for Supercomputing Applications' Visualization and Virtual Environments group as well as the Star Tap high-performance network access point. His degree is in electrical engineering, but he has contributed to a variety of projects in networking, distributed computing, and application code frameworks.

Contact the authors at cactus@cactuscode.org.