Technologies for Collaborative, Large Scale Simulation in Astrophysics and a General Toolkit for solving PDEs in Science and Engineering

Edward Seidel, for the NCSA/Potsdam/WashU Numerical Relativity Collaboration
Max-Planck-Institut für Gravitationsphysik
(Albert-Einstein-Institut), Potsdam

Abstract

We are developing a system for collaborative research and development in computational science for a distributed group of researchers at different institutions around the world. In a new paradigm for *collaborative* computational science, the computer code and supporting infrastructure itself becomes the collaborating instrument, just as an accelerator becomes the collaborating tool for large numbers of distributed researchers in particle physics. The design of this "Collaboratory" allows many users, with very different areas of expertise from diverse areas such as computer science, theoretical physics, and computer graphics, to work coherently together, on distributed computers around world. Different supercomputers may be used separately, or for problems exceeding the capacity of any single system, multiple supercomputers may be networked together through high speed gigabit networks. Central to this Collaboratory is a new type of community simulation code, called "Cactus", developed jointly by researchers at the Albert-Einstein-Institut (AEI), Washington University in St.Louis, the National Center for Supercomputing Applications (NCSA), and elsewhere. The scientific driving force behind this project is the simulation of Einstein's equations for studying black holes, gravitational waves, and neutron stars, which has brought together researchers in very different fields from many groups around the world to make advances in the study of relativity and astrophysics. But the system is also being developed as the "Cactus Computational Toolkit" to provide scientists and engineers, without expert knowledge of parallel or distributed computing, mesh refinement, and so on, with a simple framework for solving any system of partial differential equations. The toolkit is now being used at NCSA in

Illinois to provide transparent access to many parallel computer systems, from Origin 2000's and T3E's to networks of NT workstations.

1 Introduction and Overview

Computational Science is moving rapidly in many directions, driven by advances in computer hardware, software, visualization technology and also by the needs of high performance scientific and engineering applications. In many cases, these efforts are moving together, driving each other forward. In this essay, I describe an ongoing computational project to solve Einstein's equations, that govern such exotic objects as black holes and neutron stars. The project pulls together many elements of high performance computing and pure science, and allows (or rather requires) researchers from very different fields of expertise to work together. In some ways the present state of the project is a snapshot of what is just barely possible given today's technology, but it also represents a vision of seamless computing capability that enables complex scientific and engineering calculations that we hope will one day be routine. This vision has developed over many years, with influences of many of my collaborators and colleagues around the world. I am presenting the collective work of large group in this essay, for which many people share the credit, and I focus on the current state of the project and the vision for where we believe computational science can take us. The science itself, which is enabled by this computational environment, is discussed only briefly, although it is actually the major effort of these groups. (See [1, 2, 3, 4, 5, 6] for a review of dozens of papers detailing this science).

1.1 The Science Driving the Collaboratory

For a number of years, researchers in my group, together with other groups in physics, astrophysics, and computational science, have been striving to develop computational codes to solve Einstein's equations for the gravitational field. These equations for the structure of spacetime were first published in 1916 when Einstein introduced his famous general theory of relativity. This theory of gravity has remained essentially unchanged since its discovery, and it provides the underpinnings of modern theories of astrophysics and cosmology. It is essential in describing phenomena such as black holes, compact objects, supernovae, and the formation of structure in the Universe. Unfortunately, the equations are a set of highly complex, coupled, nonlinear, hyperbolic-elliptic partial differential equations involving many functions of 4 independent spacetime variables. They are among the most complicated equations in mathematical physics. For this reason, in spite of more than 80

years of intense study, the solution space to the full set of equations is essentially unknown.

Over the last 30 years a growing research area, called Numerical Relativity, has developed, where computers are employed to construct numerical solutions to these equations. Although much has been learned through this approach, progress has been slow due to the complexity of the equations, inadequate computer power, and the wide variety of numerical algorithms, computational techniques, and underlying physics needed to solve the problem.

Two most important astrophysical applications of Numerical Relativity are simulations of 3D spiraling coalescence of two black holes (BH) or neutron stars (NS), which should generate strong sources of waves. The best candidates for early detection by the laser interferometer network are considered to be BH mergers[7, 8]. The imminent arrival of data from the long awaited gravitational wave interferometers has provided a sense of urgency in understanding these strong sources of gravitational waves. These wave signals can be properly interpreted, or perhaps even detected, only with a detailed comparison between the observational data and a set of numerically computed "waveform templates".

However, a realistic 3D simulation based on the full Einstein equations is a highly non-trivial task. One can easily show that the required computational power for a single realistic simulation of coalescing binaries exceeds a Teraflop/sec and a Terabyte of memory! However, even if the necessary computers were available today, we are not yet ready to solve the problem, as still more research is required in the underlying physics, mathematics, numerical algorithms, high speed networking, and computational science. For this reason, no single group of researchers has the expertise to solve this problem.

Furthermore, the relativity and astrophysics communities themselves have historically been very specialized: most traditional relativists are very mathematical and not versed in computational science, and hence are not able to take advantage of advanced parallel computing resources, while most astrophysicists are not versed in the techniques of numerical relativity needed to study problems involving strong, dynamic gravitational fields. What is needed is an effective community framework for bringing together experts from such disparate disciplines, and harnessing their expertise in a single project, enabling mathematical relativists, astrophysicists, and computer scientists to work together side by side.

Over the last years we have developed several 3D numerical relativity projects towards such a system, culminating in the last two years' work, originating at AEI, on a system called Cactus[9, 10, 11, 12]. Cactus, a central piece of the Collaboratory, is already producing excellent scientific results in relativistic astrophysics, and is being released as a public resource for the rel-

ativity and astrophysics communities in October, 1998. Its unique and open collaborative "plug-in" architecture, described below, enables groups from around the world to build on the work of other groups, bringing high performance computing capabilities to a community without expert knowledge in scientific computing. It has the potential to act as a computational analog of Mathematica or Maple, which have become standard community tools, with many specialized "Einstein libraries", for mathematical relativity.

It is also being used by groups worldwide to develop new techniques in computational science, ranging from distributed computing through high speed networks to 3D virtual reality visualization to novel adaptive mesh refinement techniques and parallel I/O, etc. This has had the effect enabling computer scientists to see directly their algorithms applied to exotic scientific applications, and to enable physicists to solve problems for which they lacked the necessary computational expertise.

Because the underlying scientific project provides such a demanding and rich system for computational science, touching many of its different aspects, techniques that are developed to solve Einstein's equations have immediate application to a large family of scientific and engineering problems. Indeed, the scientific computing tools that we developed are now being used in the general scientific computing community, to enable easy access to a highly diverse set of parallel computing platforms. We expect the technology developed in our project to be useful both to single research groups needing access to parallel computing platforms, and to large collaborations of distributed researchers needing a central tool to integrate their work. As advances in computational technology are integrated into the system, they become available to users in all communities utilizing the system.

1.2 Overview of the Collaboratory

The application of the Einstein theory to the study of realistic astrophysical phenomena is such a complex and rich problem that progress requires major advances in computation, visualization technology, and numerical algorithms. Just as importantly, the multidisciplinary nature of both the code development process and the associated relativistic astrophysical applications demands advances in the technologies used to develop and run complex simulation codes. Hence, we have focussed on both fundamental problems in astrophysics and a set of computer science problems designed to enable *the collaborative development and use of a simulation code*.

The basic idea of the Collaboratory is to integrate the advances in these technologies so as to enable the collaborative development and use of the simulation code. Historically, simulation codes have mostly been developed and used by small groups. Even successful "community codes" are typically

developed by just a small group of developers. In our present case, with our base code containing relativity, hydrodynamics and microphysics, (1) the construction of this code is so multidisciplinary, and (2) there are such a large variety of astrophysical applications that can be built on it, that the full development and utilization of this Collaboratory needs to support a large group (at multiple institutes) of code and application developers, as well as an even larger user community. This calls for advances in software engineering, distributed computing, and collaboration technologies.

In a nutshell, this *simulation Collaboratory* aims to provide a single coherent but modular interface to diverse simulation, collaboration, and distributed computing technologies, allowing researchers to:

- create application-specific codes by composing only modules needed for a particular application;
- develop and add highly efficient parallel modules to the Collaboratory, *without* requiring knowledge of the underlying parallel infrastructure, *without* interfering with other code modules. These modules are maintained through a verification test suite designed to ensure integrity of their original function, even through future modifications by the community;
- conveniently and transparently monitor, select and carry out "simulation experiments" on available resources, regardless of location;
- analyze data resulting from the simulation experiment, both in real-time (via computational steering) and offline (via visualization analysis of data files);
- obtain access to the simulation data archive and to simulation code modules; and
- interact with other researchers at other institutions and on other continents.

A central, unifying, piece of this Collaboratory is the Cactus Computational Toolkit, which provides the underlying infrastructure for the simulation modules themselves. The Toolkit is being connected to various technology efforts in computational science, such as adaptive mesh technologies, distributed computing technologies (through Globus, described below), parallel I/O and visualization technologies, and so on. A very large set of simulation modules comprise the Cactus code for numerical relativity and astrophysics, which was the driving force behind much of the work described here.

2 The Einstein Equations

The generality and complexity of the Einstein equations make them an excellent and fertile testing ground for a variety of broadly significant computing issues. They form a system of dozens of coupled, nonlinear equations, with thousands of terms, of mixed hyperbolic-elliptic type, and even undefined

types, depending on coordinate conditions. This rich and general structure of the equations implies that the techniques developed to solve our problems will be immediately applicable to a large family of diverse scientific applications. Conversely, it also implies that solving problems of interest requires an immense variety of expertise not found in any researcher or single group, which has been the primary motivation for the entire collaborative project described here. In this section we describe the system of equations in some detail to show the depth and breadth of the problems they pose in terms of mathematical physics, computational science, and numerical analysis.

The system of Einstein equations breaks up naturally into a set of constraint equations, which are elliptic in nature, "hyperbolic" evolution equations, and gauge equations, which can be chosen arbitrarily (often leading to more elliptic equations). The evolution equations guarantee (mathematically) that the elliptic constraints are satisfied at all times provided the initial data satisfied them.

2.1 Constraint Equations

The constraint equations consist of 4 coupled nonlinear elliptic equations that must be solved for appropriate initial data. Techniques for doing so have been developed over the years for special cases in which they decouple, but the fully general case has only recently been attempted. A number of different numerical methods for solving these equations have been tried, including finite differences with multigrid or Krylov subspace iterative methods, finite elements, adaptive grids, and so forth. Numerous methods for solving the coupled elliptic equations are provided through the Cactus Computational Toolkit, including an advanced multigrid solver developed by Brügmann, the Petsc library[13], developed at Argonne National Lab, and several others under development for Cactus at the University of Illinois.

Although these equations are then guaranteed to be preserved throughout the evolution, numerically this is not the case. Many techniques have been studied to enforce the constraints during evolution, but at present there is no established consensus on how best to do this. Several techniques have been studied within our collaboration, leading to one PhD and two Masters theses in both Physics and Computer Science at the University of Illinois and the Chinese University of Hong Kong. Such techniques are planned to be made available to the Collaboratory at a later time, after the community release.

2.2 Evolution Equations

The evolution equations are used to determine the future time development of an initial data set. They are second order in space and time, nonlinear,

coupled equations, containing thousands of terms, for the six components of the metric tensor describing the spacetime geometry. In 3D, depending on how they are written, they generally form a set of dozens of coupled first order PDE's.

There are are infinitely many analytically equivalent ways of writing these equations, which will generally have quite different numerical properties. Which system is suitable for which problem is a matter of trial and error, and it is impossible for a single group to explore even a fraction of the possible formulations of equations or the different numerical methods.

The traditional approach, developed by Arnowitt, Deser, and Misner in the early 1960's, is called "ADM", and has served the numerical community well for 30 years. However, the equations were quite difficult to analyze from a numerical point of view, and typically one applied an ad hoc numerical scheme and hoped for the best in simulations. Unfortunately, most simulations using this approach break down when evolving systems such a black holes.

Over the last decade a new formulation was developed by our group and collaborators [14], where the full 3D Einstein equations are written in an *explicitly first order, flux-conservative, hyperbolic* form:

$$\partial_t D(u) + \partial_k F^k(u) = S(u) \ , \tag{1}$$

where the densities D, fluxes F^k and sources S are vector valued functions of the set of variables u. The characteristic matrix of the system has real eigenvalues and a full set of eigenvectors. This allows advanced techniques, developed after many years of research in computational fluid dynamics, to be used on the Einstein equations for the first time. This discovery has led other groups to develop independent hyperbolic formulations. At present there are at least a half dozen new formulations of the equations that may have numerical and theoretical advantages over traditional ones.

Developing and testing such formulations takes years for single groups, as a code is developed, debugged, tested, and redeveloped for different computer architectures. In many cases, groups developing such mathematical formulations do not have access to or expertise in high performance parallel computers. The Collaboratory was designed specifically to allow many different formulations of equations, along with different numerical methods that may be suitable for a variety of equations, to be implemented easily, with only a knowledge of Fortran or C, within a *common, optimized, debugged, parallel infrastructure*. In this way, such formulations and methods can be rapidly tested and compared on the *same* initial data sets, with the *same* analysis and computational tools. At present, the Collaboratory is allowing different formulations of the equations and different numerical methods to be tested by groups at AEI, WashU, NCSA, U. of Chicago, Bowling Green State University, Penn State University, Palma, and the U. of Washington/Seattle. Once

the code is released to the entire community, we expect a much broader set of methods to be tested and made available.

2.3 Gauge Conditions

Just as Maxwell's theory of electrodynamics provides a free choice of gauge in the vector potential, Einstein's theory provides four free functions called the lapse function $\alpha(x, y, z, t)$ and the 3-dimensional shift vector $\beta^i(x, y, z, t)$. The lapse function provides a physical meaning to the time coordinate t, allowing time to advance at different rates locally, while the shift vector provides a meaning for the physical movement of the spatial grid points, labeled by (x, y, z), from one time slice to the next. Conditions for the lapse and shift *must* be specified for the evolution equations. The lapse and shift can be chosen *arbitrarily* on the initial slice and thereafter. Einstein did not specify these quantities; they are up to the numerical relativist to choose at will, and on each time slice!

There are many ideas on how to best choose these functions, but none has been found to provide satisfactory evolutions in general. In fact, many seemingly obvious choices fail miserably. It is a major research topic in general relativity to find suitable choices that will allow, e.g., a black hole coalescence simulation to be carried out without having the code crash well before the interesting part of the simulation is reached. Many of the best choices to date have involved solving elliptic equations for all four gauge functions! These are very time consuming, especially considering that they must in principle be solved on each time slice, and usually dominate the entire run time of a simulation.

It has been a major research effort to develop effective methods to efficiently and accurately solve these elliptic conditions, as well as to find suitable gauge conditions in the first place. Hence, the Cactus code already has available a large family of lapse and shift conditions, as well as advanced computational infrastructure for solving elliptic equations that often arise. At present, gauge conditions are being initiated, developed, and implemented in Cactus by researchers at AEI, WashU, NCSA, Palma, Caltech, the University of Chicago, and elsewhere. It is interesting to note that Caltech and Chicago are highly mathematically based relativity efforts without numerical groups.

2.4 Relativistic Hydrodynamics

In order to make numerical relativity a tool for computational general relativistic astrophysics, it is important to combine numerical relativity with traditional tools of computational astrophysics, and in particular relativistic hydrodynamics. While a large amount of 3D studies in numerical relativity

have been devoted to the *vacuum* Einstein equations, the spacetime dynamics with a non-vanishing source term remains a large uncharted territory. General relativistic hydrodynamics will become an increasingly important subject as astrophysicists begin to study more relativistic systems, as relativists become more involved in studies of astrophysical sources. This promises to be one of the most exciting and important areas of research in relativistic astrophysics in the coming years.

Previously, most work in relativistic hydrodynamics has been done on *fixed* metric backgrounds, as most hydrodynamicists lacked necessary training in numerical relativity, and vice versa. In this approximation the fluid is allowed to move in a relativistic manner in strong gravitational fields, but its effect on the spacetime is not considered. However, a fixed background approximation is inadequate in describing a large class of problems which are of most interest to gravitational wave astronomy, namely those with substantial matter motion generating gravitational radiation, like the coalescences of neutron star binaries. Cactus contains a multi-purpose 3D code partly funded by the NASA Neutron Star Grand Challenge Project [15] that couples the full Einstein equations to general relativistic hydrodynamics. The spacetime part can be solved by any of the methods or formulations described above; the hydrodynamic part consists of both an artificial viscosity module, [16] and a module based on modern shock capturing schemes [17], employing approximate Riemann solvers to account explicitly for the characteristic information of the equations. These schemes are particularly suitable for astrophysics simulations that involve matter in (ultra)relativistic speeds and strong shock waves.

With the development of a hyperbolic formulation of the Einstein equations, the *entire* system can be treated as a single system of hyperbolic equations, rather than artificially separating the spacetime part from the fluid part, and will be possible for the first time in the Cactus code community release.

2.5 Analysis Tools

We now turn briefly to a sample of several important tools that have been developed to analyze the results of a numerical evolution, carried out by some numerical evolution scheme. The evolution will generally provide metric functions on a grid, which are usually the functions actually evolved, but they are highly dependent on both the coordinate system and gauge in which the system is evolved.

Determining *physical* information, such as whether a black hole exists in the data, or what gravitational waveforms have been emitted, are highly complex problems, requiring detailed analysis and study. As in other parts of the Collaboratory, the philosophy is to let expert groups develop such analysis

tools and provide them to the community, whether the tools are primarily theoretical (e.g. determining the intrinsic geometry of a horizon surface in curved spacetime), or computational (e.g., how does one visualize the results of an adaptive mesh refinement simulation containing dozens of variables?). Each of these tools could take years for a single group to develop. Through the Collaboratory, they are developed through the community and deployed for all to use.

- Apparent Horizons: determining if black holes exist in a spacetime is a difficult problem. A concept called the apparent horizon defines, on a given time slice, a surface that is trapped inside the black hole. The surface, if it exists, is governed by a complex partial differential equation. Many strategies have been developed over the years to find them, but they are very difficult to locate. At present three such algorithms have been implemented and tested in Cactus[12].
- Event Horizons: The event horizon is the boundary that separates light rays that can reach infinity from those that cannot. The global character of such a definition implies that the position of an EH can only be found if the whole history of the spacetime is known. Our group recently developed a method to find this surface, and produced the first simulation of colliding wormholes showing the structure of the merging horizon (shown on the cover of *Science*[18]).
- Waveform extraction: we use a gauge-invariant technique to extract gravitational waves in numerical spacetimes, and developed and tested an analysis module over the last two years.
- Principal Null Directions, Newman-Penrose quantities, Riemann-Invariants: such tools are quite complex to develop, typically used previously only in mathematical relativity, and are now available to the computational community.

These quantities and many others will be useful in analyzing a numerical spacetime. Many of these tools have been developed by the mathematical community, but never before applied in the numerical relativity community. Their introduction in the Cactus code should provide the community with a new set of tools for interpreting the results of numerical simulations.

3 The Cactus Code

A central part of the Collaboratory is the Cactus code, developed for the numerical general relativity and astrophysics communities. This code was designed and initially implemented by Joan Massó and Paul Walker in my group, based on years of experience with previous generations of 3D codes of the NCSA/WashU group, and then made available to a half dozen “seed”

groups around the world to develop the code into a true community tool. This development phase is nearing completion, with a community release slated for early 1999.

But what began as a tool for these specific scientific communities has become a unifying framework for various other research communities. It provides a central connecting point for many developing technologies in computational science community, including Globus (a system developed for distributed computing across different sites), DAGH (a system for parallel computing with adaptive mesh refinement), PETSC (a major library developed at Argonne for solving elliptic equations), PANDA (a parallel I/O library developed for efficient I/O in parallel applications), and others. All of these groups have found the various computational demands of solving the Einstein equations an excellent development ground for different techniques in computational science. In fact, each of these projects has been developed by using Cactus, and its forerunners, as examples of demanding applications. Hence the Cactus project (and its history) has had major impact on these developing technologies, which are in turn being integrated into the larger Collaboratory for use by the community. Ultimately they are being made available to computational and simulation scientists through the Cactus Computational Toolkit, providing infrastructure for solving virtually any set of PDE's. Thus, not only does Cactus provide a laboratory for numerous major projects in pure computational science, but it integrates them into a single, powerful resource for simulation scientists in many fields of science and engineering.

3.1 Concepts

Cactus is a modular framework for creating portable parallel finite-difference simulation codes. It has been designed from the start to support the development efforts of many programmers working on independent projects by providing heavy support for the CVS code revision system and a very modular system for adding simulation capabilities. The modules that plug in to Cactus are commonly referred to as "thorns". Parallelism and portability are achieved by hiding MPI, the I/O subsystem, and the calling interface under a simple abstraction API. The Cactus API supports both C and F77/F90 programming languages for the modules completely natively so that Fortran programmers need not know any C to plug into the framework, nor do they require understanding of object-oriented programming techniques. This makes it considerably easier for physicists to turn existing codes (the majority of which are implemented in Fortran) into modules (thorns) which plug into the Cactus framework. All of the benefits of a modern simulation code are available without requiring major technique changes for the programmers.

As a 3D code for numerical relativity, it represents our third generation. Based on lessons learned from earlier generations, it is the first code to effectively address the difficult software engineering problems of collaborative code development, maintenance and management. Our earlier generations of 3D numerical relativity codes [19, 20], have made us keenly aware of the issues and difficulties involved in distributed collaborative code development. For Cactus, a collaborative infrastructure has been essential. As of this writing, over two dozen collaborators at nearly a dozen institutions are using the code for various research projects in relativity, astrophysics, and computational science, and we aim at further making it a truly community code for the investigation of general relativistic astrophysics, and a general toolkit for solving PDE's of nearly any type.

It is hence designed to minimize barriers to the community development and use of the code, including the complexity associated with both the code itself and the networked supercomputer environments in which simulations and data analyses are performed. This complexity is particularly noticeable in large multidisciplinary simulations such as ours, because of the range of disciplines that must contribute to code development (relativity, hydrodynamics, astrophysics, numerics, and computer science) and because of the geographical distribution of the people and computer resources involved in simulation and data analysis. The collaborative technologies that we are developing within Cactus include:

• *A modular code structure and associated code development tools*. Cactus defines coding rules that allow one, with only a working knowledge of Fortran or C, to write new code modules that are easily plugged in as "thorns" to the main Cactus code (the "flesh"). The "flesh" contains the parallel domain decomposition software, I/O, utilities, and so forth. Then, users have the ability to plug in "thorns" to the "flesh", such as the basic Einstein solver, other formulations of the equations, analysis routines, etc. A thorn may be any code that the user wants, in order to provide different initial data, different matter fields, different gauge conditions, visualization modules, etc.

• *Extensibility to many fields of computational science.* Thorns need not have anything to do with relativity, and hence the immediate application to other fields of science and engineering: the flesh could be used as basic infrastructure for any set of PDE's, from Newtonian hydrodynamics equations to Yang Mills equations, that are coded as thorns. The user inserts the hook to their thorn into the flesh code in a way that the thorn will not be compiled unless it is designated to be active.

• *Effective code management for collaboration.* We have developed a makefile and perl-based thorn management system that, through the use of preprocessor macros that are appropriately expanded to the arguments of the flesh and additional arguments defined by each thorn, is able to create a code

which can configure itself to have a variety of different modules. This ensures that *only* those variables needed for a particular simulation are actually used, and that no conflicts can be created in subroutine argument calling lists. In this sense Cactus is a "*meta-code*": a desired code is specified by the user, and the system automatically generates a code containing *only* those routines requested. The resulting code has no knowledge of the multitude of other thorns that could be included.

• *A consistency test suite library.* An increased number of thorns makes the code more attractive to its community but also increases the risk of incompatibilities. Hence, we provide technology that allows each developer to create a test/validation suite for their own thorn. These tests are run prior to any check-in of new code to the repository, ensuring that it reproduces results consistent with previous versions, and hence cannot compromise the work of other developers relying on a given thorn.

• *Providing a testing ground for computational science techniques, with redeployment into the same code base for the community.* This is a key concept for success of the system. The code is used in many different computational science projects around the world, as described throughout this essay. Once effective techniques are developed with Cactus, e.g., for AMR or distributed computing through gigabit networks, the technology is made available to the entire user community in a transparent way, without having to modify the existing code base (where possible). This is an obvious goal, but often different code versions are created for the computational science projects, which cannot be reintegrated into the original code base. The modular thorn structure minimizes the risk of incompatible code development.

Our experiences with Cactus up to now suggest that these techniques are very effective. It allows a code of many tens of thousands of lines, but with a compact flesh that is possible to maintain despite the large number of people contributing to it. The common code base has enhanced the collaborative process, having important and beneficial effects on the flow of ideas between remote groups.

3.2 The Developer Community and Global Computing Issues

The Cactus developer community presently spans the globe with users in the USA, Germany, Spain, India, and Hong Kong. This will be greatly expanded when the code is made public for the relativity and astrophysics communities, and even more as the underlying infrastructure in the Cactus Computational Toolkit is made available for other fields of science and engineering.

Likewise the desired computational resources are also quite widely distributed, ranging from Cray T3E's, SGI Origin 2000's, Hewlett Packard/Convex Exemplars, IBM SP-2's, experimental clusters of Pentium based NT and linux

workstations, to single DEC alpha workstations, at different sites throughout the world. For these reasons it is essential to develop scalable and highly portable code that can be used effectively on many kinds of machines at highly distributed locations. Furthermore, as parallel software environments and computer architectures develop and sometimes change drastically, it is essential to have a highly adaptable framework for different architectures. Developing code for multiple platforms aids both the user community, and the inevitable transition from one architecture to another.

In the next section we discuss the strategies we have used to make the Cactus code perform well on many different architectures by themselves. However, the highly distributed nature of the community, coupled with the need for much larger computational resources than are present on any single system today (e.g.,the need for Teraflop computing), lead us naturally from high performance applications on a single scalable machines, to scaling across multiple machines, wherever they may be, connected by gigabit networks. We shall deal with that scenario in section 4 below.

3.3 Performance, Scalability, and Portability

Cactus has been developed to be a highly portable and efficient code for numerical relativity and astrophysics. It has been most extensively tested on three very different parallel architectures: the SGI Origin 2000 system, the SGI/Cray T3E system, and a cluster of 128 NT workstations, developed at NCSA, running Pentium II processors. In Fig. 1 we show scaling results achieved on a Cray T3E-600, and in Fig. 2 the results achieved on both the Origin 2000 and the NT cluster. These results indicate that codes like this can be run efficiently in parallel on a wide variety of machines.

The collaborative nature of this code has been essential in developing this performance and scalability on different machines. The optimizations for the T3E were developed by Tom Clune of Cray (in Minnesota), the Origin 2000 optimizations were carried out by Paul Walker at AEI and NCSA, and NT cluster development was done by Paul Walker and Rob Pennington at NCSA on the experimental cluster there. All optimizations were checked back in to the central CVS repository maintained in Potsdam, and all users worldwide benefit. Based on this experience with the NCSA model, the Palma group has acquired funding for a cluster of at least 64 Pentium II NT workstations, and will use the Toolkit as a foundation for supercomputing there, both for relativity and other fields of science.

We have recently tested a 1024 node T3E-1200 (provided for the NASA Neutron Star Grand Challenge Project [15], in which Cactus plays a central role), achieving 142GFlops and linear scaling, on a complete spacetime evolution, coupled to a Riemann solver based relativistic hydrodynamic thorn,

that has recently been released[1]. The full set of the Einstein equations with the perfect fluid source, involved a large number of 3D arrays. The huge number of grid points used (644 x 644 x 1284, or 500 x 500 x 996 respectively for 32 and 64 bits) is made possible by reduced memory of the code.

Previously, the largest production simulations in 3D numerical relativity (also done with Cactus) have been limited to about 350^3 or less, and applied to distorted 3D black hole systems [21, 22, 23, 24]. When such very large machines are made available for routine production simulations, we expect to further improve the results of such black hole simulations, and perform more general 3D calculations involving distorted rotating black holes, coalescences of neutron stars, gravitational waves, as well as other interesting problems in general relativistic astrophysics.

As we will see below, the ability to run and scale well on a variety of machines brings with it the realistic possibility to utilize resources distributed across different sites connected by gigabit networks, to attack problems that simply are too large to fit on any single system in the world. We also note that as the Computational Toolkit, which provides the backbone of the relativity application, will provide scientists and engineers from many other disciplines access to the same configuration of machines, including machines like the NT cluster that might otherwise be considered difficult to program.

3.4 Computational Tools

In this section we discuss a sample of the underlying computational tools that have been developed and tested in conjunction with the Collaboratory development.

3.4.1 Adaptive Mesh Refinement

3D simulations of Einstein's equations are very demanding computationally: 3D codes to solve the full Einstein equations have typically 100 double precision variables, and one can show [25] that an adequately resolved NS collision requires of order 1000 Gbytes, and machines with such capacity will not be routinely available for some years! Therefore some form of adaptive mesh refinement (AMR) that places resolution only where it is required is not only desirable, but essential. The basic idea of AMR is to use some set of criteria to evaluate the quality of the solution on the present time step. If there are regions that require more resolution, then data are interpolated onto a finer grid in those regions; if less resolution is required, grid points are destroyed. Then the evolution proceeds to the next time step on this hierarchy of grids, where the process is repeated. These ideas, developed originally by Berger

[1] available at http://wugrav.wustl.edu/Codes/GR3D/

and Oliger[26], have been refined and applied in many applications now in computational science.

There are several efforts ongoing in 3D mesh refinement for relativity, all of which are coupled into the Collaboratory. A large collaboration, begun by the NSF Black Hole Grand Challenge Alliance (of which my group has been a member), has been developing a system for distributing computing on large parallel machines, called Distributed Adapted Grid Hierarchies, or DAGH. Among other things, DAGH provides a framework for parallel AMR. Developed by Parashar and Browne, in collaboration with many subgroups within and without the Alliance, it is now being applied to many problems in science and engineering (see http://www.cs.utexas.edu/users/dagh/).

At least two other 3D software environments for AMR have been developed for relativity: one is called HLL, or Hierarchical Linked Lists, developed by Wild and Schutz[27]; another, called BAM, was the first AMR application in 3D relativity developed by Brügmann [28, 29]. The HLL system has recently been applied to the demanding test problem describing perturbations of black holes[25], and is now being developed for Cactus by Tom Goodale.

Finally, an intermediate possibility exists for some applications: fixed mesh refinement, using nested grids that do not adapt to the computation, but instead are designed ahead of time to put resolution where it is *expected* to be required. Research in my group and elsewhere has shown that such techniques can be effective for some systems with centrally located features, e.g., isolated black holes, neutron stars, or gravitational waves which collapse to form black holes. A system of nested grids, called Box-in-Box, has been developed by Lanfermann at AEI, and applied to all three problems as it was developed. Box-in-Box has been merged with BAM, to provide a truly portable, scalable system for solving *coupled* elliptic and hyperbolic equations on parallel computers. The algorithms developed for this coupling, which to my knowledge are not available in any other 3D application, provide a major step forward for such systems, and are in preparation for publication[30]. This represents a first step towards a system capable of fully adaptive mesh refinement with coupled elliptic and hyperbolic equations, which is presently an open research topic in computational mathematics and software development.

Box-in-Box is now being used and tested for a variety of problems by the different groups in our collaboration, and will be part of the Computational Toolkit release. The DAGH and HLL systems are currently under intense development for the Collaboratory at AEI and Rutgers University in New Jersey, and will be made available to the community later this year. All three systems will be available to users of the Cactus Toolkit, without essential modification to input thorns. This will lead to a wide variety of input problems, which in

turn will drive improvements in the AMR algorithms. It is such feedback that, among other features, makes the Collaboratory so powerful.

3.4.2 Parallel I/O and AMR File Structures

Another important component in a computational science toolkit for parallel machines is parallel I/O capability. It is a very nontrivial task to design a system that can effectively and efficiently output a large number of files on parallel machines, especially considering the needs of numerical relativity: AMR, of order 100 3D variables, and parallel machines, possibly connected by gigabit networks at different sites. I/O is frequently a bottleneck, in some machines a crippling one.

My group has worked closely with several I/O research teams at NCSA (HDF and FlexIO) and the Illinois Computer Science Department (PANDA) to develop and test efficient I/O strategies on parallel machines, and also to develop solutions to the important problems of file formats, which for AMR introduces additional complications in the creation of efficient data structures that must be contained in the files. Both of these problems have provided serious challenges to tackle. Here there is only room to summarize briefly the efforts.

PANDA (http://drl.cs.uiuc.edu/panda), led by Professor Marianne Winslett at Illinois, has focused on developing new data management techniques for I/O intensive applications, such as ours, that will be useful in high-performance scientific computing. They have studied our codes to develop efficient support for applications that make use of extremely large multidimensional arrays on secondary storage, developing advanced I/O libraries supporting efficient layout alternatives for multidimensional arrays on disk and in main memory, and to support high performance array I/O operations [31]. Using what they have learned from our codes, they modify their libraries to provide better performance. In the process, they have written a thorn for Cactus that can be used both in the relativity simulations or in the Computational Toolkit. The FlexIO library (http://bach.ncsa.uiuc.edu/IEEEIO) was developed by John Shalf at NCSA specifically for the Cactus project, and interoperates with both PANDA and HDF. It can be used as a file format itself, or can operate with PANDA for higher performance. Finally HDF is a worldwide file format standard developed at NCSA. Through close collaborations with this group over the years, we have influenced the capabilities or even initiated development of these systems.

At present, we are working with all three groups to develop efficient and general structures for fast I/O on parallel machines, with all three different AMR systems (DAGH, HLL, and Box-in-Box), which each have different internal data structures. As this is developed, it will become available to the

community through the Collaboratory. These systems are developed independently at various sites, and the Collaboratory is the unifying effort that connects them.

3.5 Visualization

Numerical Relativity provides an excellent driver for scientific visualization and virtual environment technology. These 3D simulations can now routinely generate dozens of gigabytes of 3D data, over dozens of fundamental tensor fields, making the analysis of simulations extremely difficult. 3D immersive virtual environments can significantly enhance our ability to comprehend the results of such simulations. With AMR simulations, having many levels of nested grids, the problems become even more severe, and without interactive, immersive 3D environments understanding the simulations may be very difficult. In the past we have worked closely with Defanti and students at the Electronic Visualization Laboratory at the University of Illinois at Chicago, and also now with Hege at the ZIB to develop visualization techniques using virtual environments. The old-fashioned postprocessing of results can become impossible with the volume and complexity of the present data structures, and the interactive Collaboratory concept becomes important in this context.

In previous work at NCSA, a series of 3D virtual environments, utilizing the CAVE and later the ImmersaDesk systems, were developed to find new ways of visualizing the results of spacetime simulations. These experiments were very successful in developing both new techniques for visualizing multiple fields resulting from spacetime evolutions, and also in developing realtime interactive visualization and control of a simulation, but the Collaboratory did not exist at that time and the system was used only to prototype the technology.[2]

To address these problems, researchers at NCSA and RZG created a remote visualization thorn to Cactus which does the visualization computations inline with the code and can be steered using a remote client application. This allows the visualization computations to run with the same degree of parallelism as the simulation codes resulting in exceptionally fast performance. The resulting geometric information is sent via a network socket connection to a visualization client application which runs on an ImmersaDesk. The geometry data requires considerably less network bandwidth than raw data under most circumstances. The client has widgets for real-time control of selected simulation parameters in addition to the visualization controls. These experiments have together demonstrated clearly the advantages of tightly cou-

[2] See http://www.evl.uic.edu/EVL/supercomp/WAVE/EINSTEIN.html for a description of this work.

pled simulation-visualization technologies which have extended our visualization capabilities to datasets which are far larger than what can be managed using offline techniques.

This capability is being extended to allow both raw data and geometry transfer, depending on network bandwidth. An array of visualization tools is under development for these purposes, in collaboration with AEI, NCSA, RZG, and ZIB. As these tools are developed, they are immediately available to all users of the Collaboratory. Users at remote sites can also in principle simultaneously view and discuss results obtained from an ongoing simulation.

Interactive steering of AMR codes requires innovative techniques for the efficient, accurate, and concise representation of AMR hierarchies, in ways that aid the user both to understand mesh structure and its relationship to the underlying physics, and to act on this understanding to improve simulation quality. These are challenging problems due to the complexity of the visualization problem itself, the need for quasi-real-time response, our as-yet only limited understanding of how users can effectively guide refinement, and the resource management issues that arise when refinement decisions change computational resource requirements.

Effective AMR code development requires a real-time dynamic interplay between the simulation code and the visualization of the algorithm's progress where the visualization is used to constantly retune the clustering and refinement algorithms. Our present AEI-ZIB-NCSA-Garching collaboration in this area builds on this code base and on basic mechanisms incorporated into Cactus for remote realtime, AMR visualization and steering from an immersive environment such as a CAVE or ImmersaDesk.

3.6 Cactus Computational Toolkit

The infrastructure developed for these collaborative projects in relativistic astrophysics has led to a very general and powerful structure for both collaboration and computational science that should be useful to many researchers. We have therefore begun work with the staff at NCSA to make this general toolkit available to the user community.

The Computational toolkit provides support for researchers to write C or Fortran codes that solve their specific science or engineering problem, without detailed knowledge of the underlying MPI layers or computer architectures. It provides access to many utilities that we have found important in developing solvers for the Einstein equations, and hence for many other systems. The toolkit provides support for solving virtually any set of finite differenced PDE's, and includes:

– an interface to efficient, general, tested MPI layers, without knowledge of MPI itself.

– access to optimized and highly tested parallel interpolators
– access to parallel I/O libraries, including the FlexIO, Panda, and ultimately HDF
– access to several different parallel elliptic solvers, including a multigrid solver and iterative solvers through Petsc and solvers under development at Illinois and AEI
– proven scaling and code portability on the NT cluster, Origin, and T3E, and access to metacomputing tools (e.g. Globus: see below)
– a fixed mesh refinement code, called Box-in-Box, including coupled elliptic-hyperbolic solvers
– checkpoint restarting from files written previously
– sample codes to show how to use it, and extensive documentation
– advanced visualization capabilities, through IDL utilities, AVS networks, Amira (developed at ZIB), and the VTK based AMR visualization library, developed jointly by NCSA, AEI, and ZIB

In addition to the above list, the Toolkit will soon include access to these advanced systems:

– two full *parallel*, MPI based AMR implementations, including DAGH and HLL
– access to Globus, which was begun through our many Supercomputing demos

With these developments and others, the Toolkit will provide very powerful capability for parallel simulations of many systems on a variety of platforms.

4 Interactive Metacomputing and High Speed Networking

The management and code development tools discussed so far have evolved to the point where a large group of distributed researchers can effectively contribute to a single code, but typically the systems are designed for a simulations on a single supercomputer. However, the concept of the *Collaboratory* demands much more than a single code running on a local machine. With a distributed collaboration at remote sites, one needs seamless access to facilities around the world, for additional computational power, for collaborative simulation and analysis, and for remote visualization of the simulation.

For example, in a seemingly trivial example of metacomputing, consider a user in Berlin who needs access to an available T3E, which happens to be in San Diego. One needs adequate bandwidth to simply get the code to San Diego, but also one needs to negotiate the idiosyncrasies of the local environment, batch system, mass storage, disk quota, and so on. In our experience, such details can become so difficult that it becomes essentially impossible for a distant user to make progress on the system.

Assuming the user overcomes these hurdles and succeeds in running a simulation, managing remote data poses one of the most difficult challenges to using distant resources. It can take longer to transfer the data that results from a simulation than to actually run the simulation code. For the largest of problems, the computational resources needed to visualize the resulting data may be nearly as large as supercomputer that was originally used to create the data. But the visualization computer may well be located on a different continent!

Now consider a problem that is simply too large for the largest computer. (This is the chronic problem of numerical relativity!) In such cases, if possible the user would like to harness multiple supercomputers together to enable simulation of the desired problem. But with few exceptions, such computers are in computing centers far from each other, perhaps on different continents. If such computers were connected with high enough bandwidth networks, and the underlying computational science infrastructure permits it, the user will want to harness the greatest possible resources to bear on the problem at hand, and will want to visualize the results *as they are computed* to ensure that such an extraordinary calculation succeeds.

Finally, in another logical step in this progression, a user may have a simulation in progress, say, in Garching with an AMR system, and numerous features develop in the simulation that require additional resources. Interactive steering and visualization of the simulation is required, since only the user will be able to determine which regions are actually interesting, given limited resources. Furthermore, the user may require dynamic acquisition of additional resources, say in San Diego, in order to continue the calculation which happens to be available. *It is such a series of increasingly complex possibilities which the Collaboratory is designed to make possible*, and we have embarked on a series of experiments to show that this is feasible.

Over the last six years my group has participated in a number of distributed computing experiments, developing the technology that will ultimately enable users to pick and choose a set of geographically distributed computers, connected by high speed networks, to simulate and visualize complex systems such as colliding BHs and NSs.

We began in 1992, developing a simulation code that ran on a CM-5 parallel supercomputer, while displaying results of the simulation–as they were computed–in a CAVE 3D Virtual Reality display. The simulation was also able to be controlled from within the virtual environment, from the choice of initial data to the data fields to be visualized and the display technique (e.g., isosurface values, height fields, etc., could all be controlled live from within the virtual environment).

This technology was first demonstrated at Supercomputing ‘93 (SC93) in Portland Oregon, and subsequently refined for SIGGraph’94. For SC95 in

San Diego, we experimented with using the I-WAY high speed network in the US to prototype distributing the single simulation across a heterogeneous network of supercomputers (including SP-2's, CM-5's, T3D's, and SGI Power Challenges) [32]. This was a particularly difficult experiment, but from this work Globus, described below, was born to provide a framework to make such simulations "routine".

As my group moved to Potsdam in 1996, and the code development and simulation effort became distributed across different continents, we extended these capabilities to include different machines available anywhere in the world, provided they are connected with sufficiently fast networks. Our distributed Cactus code, enabled by Globus, was demonstrated first at SC97 in San Jose, California, where a series of high speed ATM networks connected the RZG-Garching across the ocean, ultimately to an ImmersaDesk display in California. With this system, from San Jose we were able to launch, control, and visualize, in 3D, a simulation of gravitational waves running on a 512 node T3E in Garching. This demonstration showed that remote computing and visualization is possible. This demonstration was followed up in April, 1998, with a similar demonstration between ZIB and NCSA (http://jean-luc.aei-potsdam.mpg.de/Alliance98/).

A key element of all this work was the Collaboratory, allowing researchers from these very different research communities to work together to develop the technology that will ultimately enable scientists or engineers to have convenient access to high performance facilities, and visualize the results, wherever they may be. All the necessary code development was carried out with the same Cactus code system, and is therefore immediately available for any users in their production environment.

4.1 Globus

Every supercomputing resource that the Cactus users would like to have access to has different resource reservation procedures, data storage systems, security requirements and programming environments. The Globus system provides a single set of tools to bring control of these worldwide distributed resources to each user's desktop. Globus consists of a core set of low-level services upon which higher-level services are constructed. The low-level services used in this work are services for authentication, information distribution, resource management, and access to remote data.

Globus consists of layers of high- and low-level tools. Low-level tools for authentication [33], interfacing to local schedulers [34], discovering properties of computers and networks [35], and remote access to data and executables are available. A higher-level tool for gaining simultaneous access to multiple distributed computers is available and an implementation of MPI

using the Nexus [36] communication library allows MPI computations to be formed using distributed computers with optimized communication. In addition to these core Globus components, a high-level graphical interface is used to describe the resources needed by a particular Cactus run, locate candidate sets of resources, allow the user to select which of the candidate sets to use, start the application, and monitor it as it runs.

Using the Globus toolkit greatly assists in executing Cactus in a distributed environment, allowing us to run larger simulations with less time waiting for resources to become available. The common interface to local schedulers and tools for remote access to data allows us to execute simulations in a location-independent fashion, and information about available computational resources and network performance allows us to intelligently select which resources to use. Further, our graphical interface combines all of these components for easy use.

4.2 SC'98: True Global Simulation Capability

This work has culminated in the demonstration currently under development: Using tightly coupled supercomputers in Europe and America, we performed an intercontinental, distributed simulation of the full 3D Einstein equations of general relativity, calculating the collision of neutron stars. The simulation itself was distributed across machines on both continents, utilizing Globus, and was controlled and displayed live on an ImmersaDesk Virtual Reality system at SC'98.

Two German Cray T3E's (at ZIB and RZG) and one American Cray T3E (at SDSC), all large parallel supercomputers, worked together to perform this simulation. The domain decomposition involved one star in Europe, and one in America. Fully utilizing an OC-3 transatlantic ATM network, the two objects collided and merged (in a virtual space "somewhere" over (under) the Atlantic Ocean). Rather than having two disconnected simulations at the remote sites, this was a fully coupled calculation that treated the multiple supercomputers as a single computational system, pushing the limits of achievable bandwidth on such a transatlantic network. An additional computer at SC'98 was used to control and display the simulation.

The communications were handled by MPICH-G, a new Globus-enabled implementation of the Message Passing Interface. MPICH-G incorporates a number of features designed to support efficient execution in wide area environments, including dynamic selection of communication methods and topology information that allows optimized implementations of collective operations. The Argonne and NCSA groups continue working during this project to optimize MPICH-G performance for the transatlantic environment, for example studying the utility of specialized techniques for overlapping very long

latencies. Integrated Paradyn instrumentation will be used to guide this optimization process. Optimizations are integrated back into the Globus and Cactus systems so that users will be able to take advantage of this work for their own simulations using distributed computers, making a long term impact.

The simulation was launched from the show floor in Orlando, using a newly developed control interface displayed on an ImmersaDesk Virtual Reality system, developed at EVL. Once the simulation is launched, it can be visualized as it is computed, showing in full 3D various isosurfaces of the evolved functions indicating the merger and emission of gravitational waves.

This technology prototypes an emerging world of metacomputing, bringing together extraordinary computing resources, wherever they may be located in the world, to attack extraordinary scientific problems that cannot be solved by other means. To this end, the DFN-Verein has just funded a project between AEI, ZIB, and RZ-Garching to develop this distributed computing technology with the Collaboratory concept.

5 Physics Results

In this section I describe very briefly a few of the scientific applications of the Cactus code that have been enabled by this collaboration. Through the Collaboratory concept, users at many sites in 3 different continents work together on a daily basis to develop the code, run simulations, and explore the physics. Users at all sites contribute to and use the same code base, making rapid progress possible on a diverse set of problems.

- The Cactus code has been tested very extensively on a number of spacetimes, including dynamically sliced flat space, black holes, and weak gravitational waves [10]. This was the first extensive investigation of a hyperbolic formulation of Einstein's equations for numerical relativity in 3D, through a collaboration between AEI and our Spanish colleagues in Palma.
- The code is further being tested on highly dynamical spacetimes, including 3D distorted black holes and gravitational waves, and a series of theoretical analysis tools is being tested for a paper in preparation [11]. This is a collaboration between the AEI, NCSA, WashU, and PRL (India) groups, with tools developed and numerical experiments run regularly at all four sites.
- The code is now being used to study the collapse of pure gravitational waves to black holes, in 3D, for the first time. This problem has been under investigation for some years, but through the Collaboratory the different groups at NCSA/Potsdam/WashU have contributed coherently, making much more rapid progress as different techniques and initial data are de-

veloped and deployed. This uses Box-in-Box techniques for maximum resolution, which is required in 3D.
– The code has been used to study and compare techniques for finding 3D apparent horizons, and parameter space is being mapped out for pure gravitational wave data sets to determine under which conditions black holes exist [12].
– While the above results have been obtained with a vacuum version of the code, we have also developed fully relativistic hydrodynamics codes, which have been extensively tested, as described in a series of two papers [17, 37]. Further, some of the the first fully relativistic collisions ever performed of coalescing neutron stars, using thorns developed jointly at AEI, NCSA, Palma, Valencia, and WashU, were achieved recently. Box-in-Box is being used to resolve the stars.
– Boson Stars. The NCSA/Potsdam/WashU collaboration has developed a new module to study fully relativistic self-gravitating scalar fields, as model problems for compact object studies, for studies of gauge conditions and other techniques for numerical relativity, and as interesting problems in their own right.
– Black Holes. Much of the testing of Cactus has been aimed at evolutions of black holes, including Schwarzschild, highly distorted, and colliding black holes. Initial data sets developed at all sites have been evolved, and waveforms determined. Again, Box-in-Box techniques are being used for these simulations, which require high resolution near the holes.

6 Conclusions, and the Future

We have made an important beginning in several important areas of computational science, developing an ambitious program of astrophysical and computer science research to bring the numerical treatment of the Einstein theory of general relativity to the astrophysics community at large. The project has produced a new collaborative community framework for research on problems involving strongly gravitating astrophysical systems (e.g., neutron stars and black holes). It has also developed new computing technologies that allow massively parallel simulation codes to be simultaneously developed and used by a large, multidisciplinary, and distributed community. There have been two key challenges in building this simulation Collaboratory:

1. *The complexity and richness of the Einstein theory when applied to realistic astrophysical phenomena demands simultaneous major advances in computation, visualization, and numerical algorithms.* Our scientific research involves 3D simulations to construct numerically the full dynamical spacetime containing, e.g., neutron stars and black holes.

2. *The complexity of the resulting simulation code and the range of problems to which it can be applied demands new collaborative approaches.* Both the integration of the Einstein theory with the traditional tools of astrophysics (e.g., relativistic hydrodynamics, nuclear astrophysics) and the application of the resulting code to astrophysical problems are inherently multidisciplinary problems. Traditional code development and execution approaches discourage such multidisciplinary interactions. Responding to these needs, we have developed a *simulation-centered Collaboratory* that can support a large user *and* developer community and that reduces dramatically the barriers to experimentation and exploration.

The Collaboratory enables a wide spectrum of researchers in the community to cooperate on code development and use. This has the effect to (1) drastically increase scientific productivity by fostering collaboration, cutting down redundant efforts by different research groups, and (2) maximize the benefit of massively parallel computing to the community. Furthermore, the introduction of this new concept of Collaboratory to different communities, with researchers in many different groups working together with the same computational infrastructure, will have a beneficial sociological impact on the community and beyond. We expect this *community code co-development* to focus and foster collaboration on a scale not previously possible.

The central and unifying part of this project is the collaborative code called Cactus, which encourages collaboration among a large number of developers. This code consists of a collection of routines, called thorns, that solve physics and engineering problems (in our case the Einstein equations), and a central computational infrastructure, called the Cactus Computational Toolkit, that provides an efficient, parallel system for computation. To make this system effective, we have had to develop new code and resource management techniques designed to enable the collaborative development and use of simulation codes. This research addresses fundamental issues of modular code construction, code distribution, resource location, interface design, and computational steering, all in the demanding context of high-performance simulation applications, and ties to together many developing technologies, such as Globus, DAGH, PANDA, PETSC, and other projects that themselves use our scientific problems as major drivers in their development.

While we are fundamentally motivated to push the frontiers of the study of Einstein's equations, this work has by necessity also pushed the frontiers of computational science. In the future, we expect the computational work to continue in the following directions:

• Computational Technology: *We are working to achieve advances in large-scale parallel 3D adaptive mesh refinement (AMR).* The astrophysical simulations involve many drastically different dynamical time and length scales. In order to meet these requirements, we will develop innovative interactive,

steerable AMR technology that supports on-the-fly, on-demand adjustment of refinement criteria and refinement region.

• Visualization Technology: *We are developing new visualization and distributed computing technologies allowing collaborative real-time interactive "window into the oven" visualization of large simulation codes.* These capabilities are required both for the numerical construction of dynamical spacetimes in the Einstein theory and for interactive AMR, and will support the monitoring of a simulation in real time by groups of distributed users. The features to be visualized can be chosen or changed on the fly, with feature extraction carried out locally by each computational node in parallel, for efficiency in large-scale parallel computation.

• Numerical Algorithms: *We will bring the high resolution shock capturing methods developed in computational fluid dynamics [38] to the numerical treatment of the Einstein equations*. We also *develop broadly applicable highly parallel methods for evolving coupled hyperbolic-elliptic systems*.

• Network Computing Technology: *We are developing innovative resource brokering and configuration technology to support efficient and flexible execution of large simulation codes in distributed environments.* Building on the Globus metacomputing toolkit, we are developing new techniques for the location and scheduling of compute resources for simulation and data analysis tasks. We will also explore the *dynamic* allocation of additional networked computing resources, wherever they may be physically located, for example when needed *during* an adaptive mesh refinement simulation.

We hope that these innovations will transform the regimes within which relativistic simulations can be performed and increase the scope of problems that can be addressed. We also hope that the successful establishment of this new model for the *collaborative development and use of simulation codes*, exploiting high-speed connectivity not only between computers but also between researchers, will have a significant impact on other research communities.

Acknowledgements

By nature this is clearly a highly collaborative project, and I am indebted to a great many people at different institutes who share the credit for the results. I am most indebted to Joan Massó, Larry Smarr, and Wai-Mo Suen, and Paul Walker, who have each provided enthusiasm, inspiration, vision, and especially expertise to all aspects of this work. The basic design and implementation of the Cactus code was by Joan and Paul, based on years of collaborative experience between my group at NCSA and Wai-Mo's at Washington University in St. Louis. Ian Foster and his team at Argonne National Lab have provided a vision and the technical expertise needed to actually carry out our distributed computing experiments. For the local support and continuation of the Cactus code, Gabrielle Allen, Bernd Brügmann, Tom Goodale, and Gerd Lanfermann have become its life blood, without whom the Collaboratory would not be possible. Finally, and most importantly, I am indebted to many physicists who use the Cactus code, at AEI and elsewhere, for supporting it and making the vision work.

References

[1] E. Seidel, Acta Helvetica **69**, 454 (1996).
[2] E. Seidel, in *Relativistic Astrophysics*, edited by F. Hehl, H.-P. Nollert, and H. Riffert (Vieweg, 1997).
[3] E. Seidel, P. Walker, and J. Massó, in *Supercomputer 97*, edited by H.-W. Meuer (K. G. Sauer-Verlag, 1997).
[4] E. Seidel, in *On the Black Hole Trail*, edited by B. Iyer and B. Bhawal (Kluwer, ADDRESS, 1998).
[5] E. Seidel, in *Proceedings of GR15*, edited by N. Dadich (1998).
[6] E. Seidel and W.-M. Suen, J. Comp. Appl. Math. (1999), in press.
[7] Éanna É. Flanagan and S. A. Hughes, Phys. Rev. D **57**, 4566 (1998), gr-qc/9710129.
[8] Éanna É. Flanagan and S. A. Hughes, Phys. Rev. D **57**, 4535 (1998), gr-qc/9701039.
[9] *http://cactus.aei-potsdam.mpg.de*.
[10] C. Bona, J. Massó, E. Seidel, and P. Walker, (1998), gr-qc/9804065. Submitted to Physical Review D.
[11] M. Alcubierre *et al.*, (1998), in preparation.
[12] M. Alcubierre *et al.*, (1998), gr-qc/9809004.
[13] S. Balay, W. Gropp, L. C. McInnes, and B. Smith, PETSc - The Portable, Extensible Toolkit for Scientific Compuation, 1998, http://www.mcs.anl.gov/petsc/.
[14] C. Bona, J. Massó, E. Seidel, and J. Stela, Phys. Rev. Lett. **75**, 600 (1995).
[15] For a description of the NASA Neutron Star Grand Challenge Project, see, e.g., http://wugrav.wustl.edu/Relativ/nsgc.html.
[16] E. Wang and F. D. Swesty, in *Proceedings, Texas Symposium on Relativistic Astrophysics* (1997).
[17] J. A. Font, M. Miller, W. M. Suen, and M. Tobias, (1998), submitted to Phys. Rev. D.
[18] R. Matzner *et al.*, Science **270**, 941 (1995).
[19] P. Anninos *et al.*, Phys. Rev. D **52**, 2059 (1995).
[20] P. Anninos *et al.*, Phys. Rev. D **56**, 842 (1997).
[21] K. Camarda and E. Seidel, Phys. Rev. D **57**, R3204 (1998), gr-qc/9709075.
[22] K. Camarda and E. Seidel, gr-qc/9805099. To appear in Physical Review D.
[23] G. Allen, K. Camarda, and E. Seidel, (1998), gr-qc/9806014. Submitted to Phys. Rev. D.
[24] G. Allen, K. Camarda, and E. Seidel, (1998), gr-qc/9806036. Submitted to Phys. Rev. D.
[25] P. Papadopoulos, E. Seidel, and L. Wild, Physical Review D (1998), submitted, gr-qc/9802069.
[26] M. Berger and J. Oliger, Journal of Computational Physics **53**, 484 (1984).
[27] L. Wild and B. Schutz, (1998), in preparation.
[28] B. Brügmann, Phys. Rev. D **54**, 7361 (1996).
[29] B. Brügmann, (1997), gr-qc/9708035.
[30] B. Brügmann *et al.*, , in preparation.
[31] S. Kuo *et al.*, in *Proceedings of the SIAM Conference on Parallel Processing for Scientific Computing* (SIAM, Minneapolis, MN, 1997).
[32] R. Gjertsen *et al.*, Forefronts **11**, (1996), cornell Theory Center.
[33] I. Foster *et al.*, in *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing* (1997).
[34] K. Czajkowski *et al.*, Lecture Notes on Computer Science (1998).
[35] S. Fitzgeranld *et al.*, in *Sixth IEEE International Symposium on High Performance Distributed Computing* (1997).
[36] I. Foster, C. Kesselman, and S. Tuecke, Journal of Parallel and Distributed Computing 70 (1996).
[37] E. Bougleux and et al, (1998), in preparation.
[38] F. Banyuls *et al.*, ApJ **476**, 221 (1997).

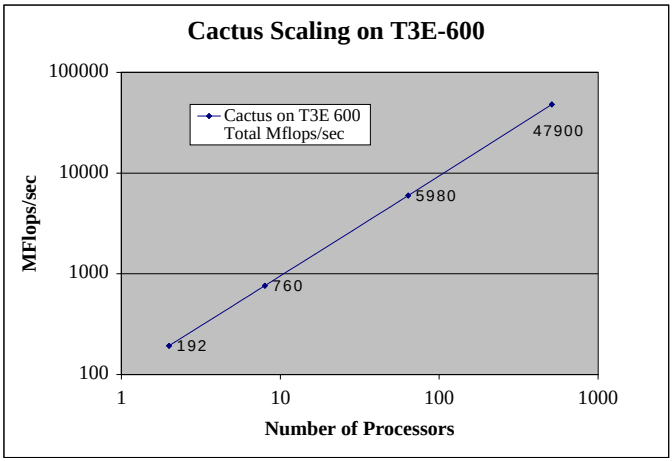

Fig. 1: We show the scaling of the Cactus code on the Cray T3E-600, giving the total Mflops/sec as a function of the number of processors. The grid size per processor is kept constant as the number of processors is increased (so the total problem size scales with the size the machine).

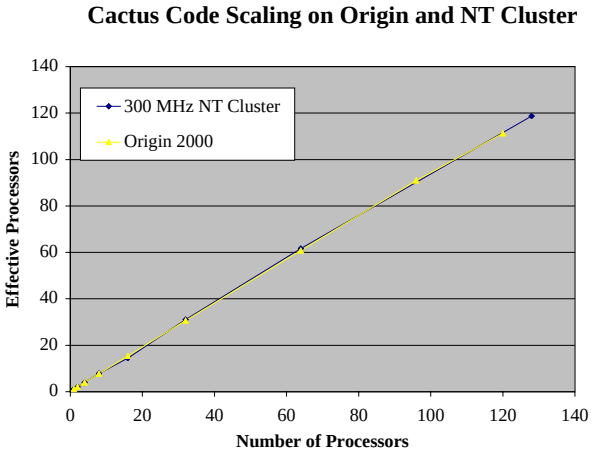

Fig. 2: We show scaling on two very different architectures: an SGI/Cray Origin 2000 DSM architecture with 128 processors, and a cluster of Compaq workstations running Windows NT.