

# The Cactus Computational Collaboratory: Enabling Technologies for Relativistic Astrophysics, and a Toolkit for Solving PDE's by Communities in Science and Engineering

Gabrielle Allen, Tom Goodale, and Edward Seidel  
Max-Planck-Institut-für-Gravitationsphysik (Albert-Einstein-Institut)  
1 Schlaatzweg  
14473 Potsdam, Germany

## Abstract

*We are developing a system for collaborative research and development 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, to work coherently together, on distributed computers around the 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”. 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 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 on many parallel computer systems, from traditional supercomputers to networks of 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. Here, we

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. We focus on the computational aspects of the project, and only briefly mention the science itself, although it is actually the major effort of the groups involved. (See [1, 2, 3, 4, 5, 6] for a review of dozens of papers detailing this science).

*The Science Driving the Collaboratory.* For many years, research 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, and hence numerical studies of Einstein’s equations are absolutely essential. But progress in this area has been very 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 equations.

Two most important astrophysical applications of Numerical Relativity are simulations of 3D spiraling coales-

cence of two black holes or neutron stars, which should create strong sources of gravitational waves. 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: 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, many traditional relativists concentrate on the mathematics and are not versed in computational science, and hence are not able to take advantage of advanced parallel computing resources, while many 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.

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.

*Overview of the Collaboratory.* The basic idea of the Collaboratory is to integrate the advances in emerging 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, hydrodynam-

ics 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* provides 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, and *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 Cactus Code

A central part of the Collaboratory is the Cactus code, developed for the numerical general relativity and astrophysics communities. 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, which are in turn being integrated into the larger Collaboratory for use by the community. Ultimately they will be 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.

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.

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. The thorns register with the flesh using a method that eliminates actual changes to the flesh code. Further, users choose at run time which

compiled thorns will actually be used, with unused thorns not contributing to computational overheads such as memory or CPU time.

*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.

*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 contain 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 the previous version, 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 existing code base (where possible). This is an obvious goal, but often different code versions are created for computational science projects, which cannot be reintegrated into the original code base. The modular thorn structure minimizes the risk of incompatible code development.

Cactus has also 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 all cases scaling is ex-

cellent, achieving well over 90% of the maximum linear scaling. We have recently tested a 1024 node T3E-1200 (provided for the NASA Neutron Star Grand Challenge Project (<http://wugrav.wustl.edu/Relativ/nsgc.html>), in which Cactus plays a central role), and achieve 142GFlops on a complete spacetime evolution, coupled to a Riemann solver based relativistic hydrodynamic thorn, that has recently been released (available at <http://wugrav.wustl.edu/Codes/GR3D/>).

## 2.1 Computational Tools

In this section we mention a sample of the underlying computational tools that are being developed and tested in conjunction with the Collaboratory development. For example, adaptive mesh refinement will be essential in providing the necessary resolution to accurately simulate systems like coalescing neutron stars, which are highly compact objects spiraling together, emitting both short and long wavelength gravitational waves, while also collapsing catastrophically to a black hole which has infinite density at its core. There are several independent efforts ongoing in 3D mesh refinement for relativity, which are being coupled into the Collaboratory. A system for distributing computing on large parallel machines, called Distributed Adapted Grid Hierarchies, or DAGH, has been developed by Parashar and Browne (see <http://www.cs.utexas.edu/users/dagh/>). Among other things, DAGH provides a framework for parallel AMR, which is being integrated in Cactus. Another AMR system is under development at Potsdam, called HLL, or Hierarchical Linked Lists, developed by Wild and Schutz. Finally, 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, has been developed and implemented as a thorn in Cactus. This nested box system has been merged with a parallel multi-grid solver to provide a truly portable, scalable system for solving *coupled* elliptic and hyperbolic equations on parallel computers. 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.

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, on some machines

a crippling one. PANDA (<http://drl.cs.uiuc.edu/panda>), led by 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. 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 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.

Finally, Numerical Relativity also 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.

## 2.2 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. 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 will include:

- 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 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 clusters, Origin, and T3E, and access to metacomputing tools (e.g. Globus);
- a fixed mesh refinement code, called Box-in-Box, including coupled elliptic-hyperbolic solves;
- 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;
- two full *parallel*, MPI based AMR implementations, including DAGH and HLL

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

### 3 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 simulation 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.

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 which happen to be available, say in San Diego, in order to continue the calculation. 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.

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. Globus, developed by a team headed by Ian Foster and Carl Kesselman, 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. 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.

This work has culminated in a demonstration performed at SC'98: 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 Immersedesk Virtual Reality system. 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 achieved by other means.

## 4 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, multi-disciplinary, and distributed community.

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.

We hope that this work will transform the regimes within which relativistic simulations can be performed, and will increase the scope of problems that can be addressed, and 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 im-

pact on other research communities.

## 5 Acknowledgments

This is clearly a highly collaborative project, and we are indebted to a great many people at different institutes for this work. The basic design and implementation of the Cactus code was by Joan Massó and Paul Walker, and it was further developed at AEI, NCSA, and Washington University in St. Louis. Ian Foster and the team at Argonne National Lab have provided a vision and the technical expertise needed to actually carry out the distributed computing experiments, and many others have contributed throughout the development of this project.

## 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, 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.