

The Cactus Computational Toolkit and Using Distributed Computing to Collide Neutron Stars

Gabrielle Allen, Tom Goodale, Joan Massó and Edward Seidel
Max-Planck-Institut-für-Gravitationsphysik (Albert-Einstein-Institut)
Am Mühlenberg 5
14476 Golm, 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 Motivation for the Cactus Computational Toolkit

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. This project, based around the *Cactus Computational Toolkit*, 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] for a review).

The Science Driving the Development of Cactus. 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.

An Application: Colliding Neutron Stars. One of the most important astrophysical applications of Numerical Relativity is the simulation of the 3D coalescence of neutron stars, which should create strong sources of gravita-

tional 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”.

The Need for a Flexible, Collaborative Framework. An accurate 3D simulation based on the full Einstein equations is a highly non-trivial task: using current techniques a single realistic simulation of coalescing binaries would exceed a Teraflop/sec and a Terabyte of memory! Even if all the necessary computer hardware were available today, we are not yet ready to solve the problem, as still more research is required in the underlying physics, mathematics and numerical algorithms. 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. Historically, simulation codes have mostly been developed and used by small groups. Increasing the number of developers or users can easily lead to management problems, and comparisons or sharing of routines with other codes is problematic.

2 The Cactus Code

The Cactus code was originally developed for the numerical general relativity and astrophysics communities. But what began as a tool for specific scientific communities has become a unifying framework with the potential to be utilised by other research communities.

Cactus is a modular framework for creating portable parallel 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 the driver layer (supplying parallelization), the I/O subsystem, and the calling interface under a simple abstraction API. The Cactus API supports C/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 pro-

gramming 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” handles the interfaces between the various thorns, provides functions to handle the calling-order of modules, and provides a standard interface layer for utilities such as memory allocation, parallelization and I/O. The “thorns”, taken either from the Cactus Computational Tool Kit or the users own personal tool kit, supply most of the functionality of the code. These thorns can range from an Einstein solver, to basic analysis routines, to a more complicated module providing a parallelization driver. 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 could be thorns.

Effective code management for collaboration. The Make system for Cactus uses configuration scripts, which each thorn must provide, to generate information about the variables, parameters, argument lists, and scheduling information for the particular chosen set of thorns. This information is needed for the code compilation, and is also used for the range of consistency checks which are performed at compile time. 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. Cactus provides scripts that allows each developer to create a test/validation suite for their own thorn. These tests can be 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.

A testing ground for computational science techniques. The Cactus code is used as a testcase for various computational science projects. Once effective techniques are developed within Cactus, e.g., for AMR or distributed computing through gigabit networks, the modular thorn structure allows the new technology to be made available to the entire user community in a transparent way, without having to modify existing code base (where possible).

Portability and efficiency Cactus has been developed from the start to be a highly portable 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 excellent, 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 achieved over 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

The computational tools which are currently available in the Cactus Computational Toolkit include:

- parallelization via an interface to a unigrid MPI driver layer for partial differential equations, implemented as a thorn. The use of this layer in application thorns requires no knowledge of the underlying method of parallelization (here MPI) itself. The generality of the interface means that as soon as new drivers are available, they can be used after simply recompiling and making a change to the parameter file;
- parallel interpolator thorns;
- several basic parallel elliptic solvers thorns;
- proven scaling and code portability on NT and Linux clusters, Origin, and T3E, and access to metacomputing tools (e.g. Globus);
- a fixed mesh refinement thorn;
- thorns implementing routines for I/O, using either the FlexIO (<http://bach.ncsa.uiuc.edu/IEEEIO>) or HDF5 (<http://bach.ncsa.uiuc.edu/IEEEIO>) libraries. I/O options include 0D, 1D, 2D or 3D parallel output, writing or reading checkpoint files to allow restarting runs, and general file readers.

- sample codes to illustrate using and developing modules;

With the general Cactus framework, users also have access to

- 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;
- support facilities, including a help-desk for the Cactus Computational Toolkit and Mailing Lists.
- Users and developers documentation.

A number of new or emerging computational science technologies are being developed and tested with the Cactus framework. Since they are already being integrated as “thorns” they will be readily available to all users as soon as they reach maturity.

Adaptive Mesh Refinement. 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 Cactus code. 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 currently 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 by Lanfermann et al. and implemented as a thorn in Cactus. This nested box system has been merged with a parallel multigrid solver to provide a truly portable, scalable system for solving *coupled* elliptic and hyperbolic equations on parallel computers.

Parallel Input/Output. 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 multi-dimensional 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 which will become part of the Computational Toolkit. It can be used as a file format itself, or can operate with PANDA for higher performance. Through close collaborations with this group over the years, we have influenced the capabilities or even initiated development of these systems.

Visualization. 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.

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, with a distributed collaboration at remote sites, one would like 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 will assist 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 will eventually allow us to execute simulations in a location-independent fashion, and information about available computational resources and network performance will allow us to intelligently select which resources to use.

So far, the use of Globus with the Cactus code 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 Immersadesk 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 prob-

lems that cannot be achieved by other means.

4 Conclusions, and the Future

We are following an ambitious program of astrophysical and computer science research to bring the numerical treatment of the Einstein theory of general relativity to the general astrophysics community and beyond. 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.

The Cactus framework 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 concept of Code Collaboration 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 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 comput-

ers but also between researchers, will have a significant impact 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, in *Gravitation and Relativity: At the Turn of the Millennium. The Proceedings of GR15*, edited by N. Dadich and J. Narliker (1998). (<http://www.lanl.gov/abs/gr-qc/9806088>).
- [2] C. Bona, J. Massó, E. Seidel and P. Walker, submitted to Phys. Rev. D. (<http://www.lanl.gov/abs/gr-qc/9806088>).