# The Cactus Code: A Problem Solving Environment for the Grid

Gabrielle Allen[*] Werner Benger[*†] Tom Goodale[*] Hans-Christian Hege[†]
Gerd Lanfermann[*] André Merzky[†] Thomas Radke[*] Edward Seidel[*§]
John Shalf[§]

January 26, 2001

**Abstract**

Cactus is an open source problem solving environment designed for scientists and engineers. Its modular structure facilitates parallel computation across different architectures and collaborative code development between different groups. The Cactus Code originated in the academic research community, where it has been developed and used over many years by a large international collaboration of physicists and computational scientists.

We discuss here how the intensive computing requirements of physics applications now using the Cactus Code encourage the use of distributed and metacomputing, describe the development and experiments which have already been performed with Cactus, and detail how its design makes it an ideal application test-bed for Grid computing.

## 1 Introduction

Cactus [1], [2] is an open source problem solving environment designed to provide a unified modular and parallel computational framework for physicists and engineers.

The Cactus Code was originally developed to provide a framework for the numerical solution of Einstein's Equations [3], one of the most complex sets of partial differential equations in physics. These equations govern such cataclysmic events as the collisions of black holes or the supernova explosions of stars. The solution of these equations with computers continues to provide challenges in the fields of mathematics, physics and computer science. The modular design of Cactus enables people and institutes from all these disciplines to coordinate their research, using Cactus as the collaborating and unifying tool.

The name Cactus comes from the design of a central core (or *flesh*) which connects to application modules (or *thorns*) through an extensible interface. Thorns can implement custom developed scientific or engineering applications, such as the Einstein solvers, or other applications such as computational fluid dynamics. Other thorns from a standard computational toolkit provide a range of capabilities, such as parallel I/O, data distribution, or checkpointing.

[*]Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, Golm (AEI)
[†]Konrad-Zuse-Zentrum für Informationstechnik, Berlin (ZIB)
[§]National Center for Supercomputing Applications, Champaign, IL, (NCSA)

Cactus runs on many architectures. Applications, developed on standard workstations or laptops, can be seamlessly run on clusters or supercomputers. Parallelism and portability are achieved by hiding the driver layer and features such as the I/O system and calling interface under a simple abstraction API. The Cactus API supports `C/C++` and `F77/F90` programming languages for the thorns. Thus thorn programmers can work in the language they find most convenient, and are not required to master the latest and greatest computing paradigms. This makes it easier for scientists to turn existing codes into thorns which can then make use of the complete Cactus infrastructure, and in turn be used by other thorns within Cactus.

Cactus provides easy access to many cutting edge software technologies being developed in the academic research community, such as the Globus Metacomputing Toolkit, HDF5 parallel file I/O, the PETSc scientific computing library, adaptive mesh refinement, web interfaces, and advanced visualization tools.

## 2 The Need for the Grid

The large and varied computational requirements of relativistic problems, such as black hole collisions, make them a good example for demonstrating the need for Grid computing, and an ideal testbed for developing solutions. In developing the Cactus infrastructure to make full use of the Grid for such problems these advances are then immediately available for all applications.

Implementing the full Einstein Equations in a finite difference code amounts to a memory requirement of around one hundred 3D arrays, and a CPU requirement of thousands of floating point operations per grid point. Considering that an accurate solution of a full 3D black hole problem will require at least 1000 grid points in each spatial dimension, this implies TeraByte/TeraFlop computers. Further, to analyse the large data sets created during a simulation requires advanced techniques in file management and visualisation.

To date, the resources of the individual computers available have limited simulations to around 200-300 grid points in each spatial dimension. Even then, simulations generate huge amounts of data, and negotiating the curiosities of different supercomputers, such as batch queues and file systems, is not something that physicists relish.

The Grid provides a way to access the resources needed for these simulations. It provides a uniform access layer to supercomputers and computing resources, making these resources far more useful to scientists who want to use them for simulations. Given the appropriate permissions, networks, allocations and Grid enabling software, a scientist can in principle run a simulation on a set of supercomputers, all connected by the Grid, and thus be able to run far larger problems than would be possible on a routine basis without the Grid. With proper software tools, the Grid provides the necessary infrastructure to connect the machines, and to deal with the resulting large data sets, and, ultimately, the resources to analyse such volumes of data.

The dream for physicists is that Grid computing will provide a scientific programming environment similar to that shown in Figure 1, allowing working scenarios such as:

> *A physicist, sitting in a cafe in Berlin has an idea for a colliding black hole run, maybe to try a new initial data set, or to test a new evolution method. She uses GUIs on her PalmTop to select the Cactus thorns needed, and to estimate the computer resources she requires. Her Grid software takes over, selecting the most appropriate machine or set of machines to use from those available to her. This Grid software automatically*

*creates or transfers executables and parameter files and starts the run on the remote resources. After several coffees, she connects to the running Cactus, using one of the remote access thorns, and sees that things are going better than expected. She rings up colleagues in the USA, who watch the isosurfaces being streamed out from Cactus. They want to save some 3D data sets to analyse later, so they connect to the Cactus run using their web browser, and turn on output for the grid functions they are interested in.*

As futuristic as such a scenario sounds, all the pieces already exist in a prototype form, and are being further developed and integrated, as described below and in [1], [4], [5] and [6].

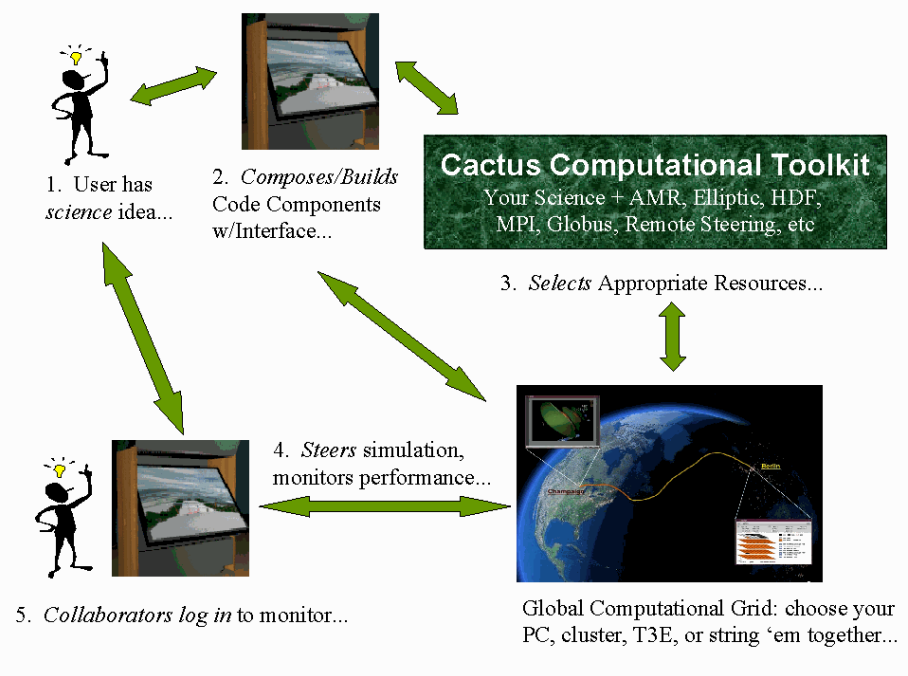


Figure 1: The dream of Grid computing.

# 3 Metacomputing with Cactus

Cactus was designed with the Grid and Grid applications in mind. It provides a layer on top of the Grid, giving a programming interface which allows the user to be completely ignorant of the nature of the machine or machines that the simulation runs on. The code provides access to Grid resources such as distributed I/O and parallelisation across any number of supercomputers with precisely the same interface as it does to the resources of a single machine.

Cactus thus provides a convenient laboratory for computer scientists to develop metacomputing techniques, which can then be tested with real physics applications and also by real users without

requiring changes to the physics application code. When a new technique is perfected, it can immediately be made available to the whole community of Cactus Users.

The parallel driver layer in Cactus, which manages the grid variables and their communications, is provided by a thorn. This means that different thorns can be used to implement different parallel paradigms, such as `PVM`, pthreads, `OpenMP`, `CORBA`, etc. Cactus can be compiled with as many driver thorns as required (subject to availability), with the one actually used chosen by the user at run time through the parameter file. Currently the standard driver thorn is called `PUGH`, which uses `MPI` to provide parallelism.

The I/O subsystem is implemented in a similar, generic manner within Cactus: the flesh provides an interface for arbitrary I/O thorns to register their specific I/O methods. These methods can then in turn be invoked by any application thorn to read data into Cactus variables or dump their contents for postprocessing analysis. The I/O thorns available in the computational toolkit provide methods to write in different formats (1D traceline plots, 2D slices, full 3D arrays, reduction (e.g. maximum value) scalar, isosurfaces, screen output) and using different I/O libraries (FlexIO [8], HDF5 [9], ASCII). Further methods or libraries can easily be employed by thorn programmers.

Metacomputing developments and experiments have been performed using Cactus for several years, some of which are described in the sections below. Capabilities are being further developed in connection with Cactus through several projects. A DFN-Verein project [4] at the AEI in Germany is funded to exploit high speed networks for colliding black hole simulations, and is concentrating on remote visualization [11], remote steering and distributed I/O [10]. A project funded by the so-called KDI program of the American National Science Foundation (NSF) joins five institutes to develop an *Astrophysics Simulation Collaboratory* [5] which will provide an environment for physicists to utilise Grid computing for problems such as the collisions of neutron stars. The GrADs project, also funded by the NSF in the USA, is using Cactus as one of its applications for developing a Grid based computing environment. The Europen Grid Forum [12] has chosen Cactus as one of its 4 exemplary applications running on the European Grid-TestBed. These technologies are being brought into the scientific and engineering communities as they are developed.

## 3.1 Distributed Simulations

We are actively working to develop techniques which allow researchers to harness computational resources wherever they may be on the Grid. This could include a distributed set of machines connected by high speed networks, allowing larger or faster simulations than would be possible on a single machine. At Supercomputing 98 a neutron star collision was run with Cactus, using the Globus [6] metacomputing toolkit to split the domain across two T3Es on different sides of the Atlantic, one in Munich, Germany and one in San Diego, California. In this simulation the neutron stars collided somewhere in cyberspace, over the Atlantic Ocean. The simulations were launched, visualised, and steered from the show floor in Orlando. The scaling across the two machines used for this simulation was roughly 50%, which we believe is excellent considering the large amount of communication required between the machines to solve these equations and the latencies in the transatlantic link.

### 3.2 Direct Remote Partial File Access

Large scale computer simulations often generate large scale data sets. Even a single such simulation may generate files containing several hundreds of GBytes, up to the order of a TByte as machines grow. Conventional postprocessing analysis is then extremely resource-intensive when remote simulation data has to be staged for local processing. Also, in many cases, for example for visualization purposes, only a small fraction of the overall data is really needed. For example, in a simulation of the time evolution of two colliding black holes, the output may contain a dozen variables representing the state of the gravitational field at perhaps 1000 time steps during the evolution. For visualization purposes one might want to analyze only the first time step of one or two of the variables.

By combining the HDF5 I/O library [9] with the Data Grid software component from the Globus toolkit we enable existing I/O layers to operate on remote files directly — completely transparent to the application. Using the data selection capabilities of HDF5 (defining so-called hyperslabs as arbitrary rectangular subregions in the multidimensional data fields, optionally with downsampling applied) individual time steps and zones of interesting data can be read and visualised in a very efficient and convenient way.

Currently the Data Grid client software supports remote partial file access to Distributed Parallel Storage Systems (DPSS). Access to files which are located on any webserver in the world will be added soon via the standard HTTP-1.1 protocol. During Supercomputing '99 and at CeBIT 2000 we successfully demonstrated the feasibility of such a Data Grid Infrastructure. In these demonstrations, Cactus simulation data residing on remote DPSS data servers was visualised by an HDF5-enabled version of the visualization package Amira [7]. This is illustrated in Figure 2.

Another challenge comes into play if simulations are carried out on a distributed computer and generate physically distributed files. This would occur, for example, in order to exploit parallel I/O methods. It is desirable to access and transfer such distributed data sets as consistent single files, with a global address space that has pointers to other pieces at other locations. We plan to tackle those problems also with the Data Grid components, by organizing related files as collections of logical file instances [13].

### 3.3 Remote Visualisation

Some Cactus I/O thorns already provide the capability to stream online-data via socket connections. Multiple visualization clients such as Amira can then connect to this socket from a remote machine and display simulation results in real-time, e.g. isosurfaces of gravitational waves emitted during a black hole collision, or photons falling into a black hole.

Currently the data streaming is implemented as a proprietary protocol. This will be made more generic by extending the HDF5 I/O library to stream online data. For that the *Virtual File Driver* (VFD) concept of HDF5 is exploited which allows applications to choose between different low-level drivers for reading/writing raw data. We developed our own VFD which keeps the HDF5 data to be streamed out of Cactus as an in-memory file. On a flush/close operation this file is sent through a socket to any connected clients. At the client side, the same VFD is used to reconstruct the in-memory file which then can be accessed as usual to read HDF5 data.

Since the VFD layer hides all low-level I/O operations from the upper layers of the HDF5 library and from the application, the already existing I/O methods in Cactus can be used transparently

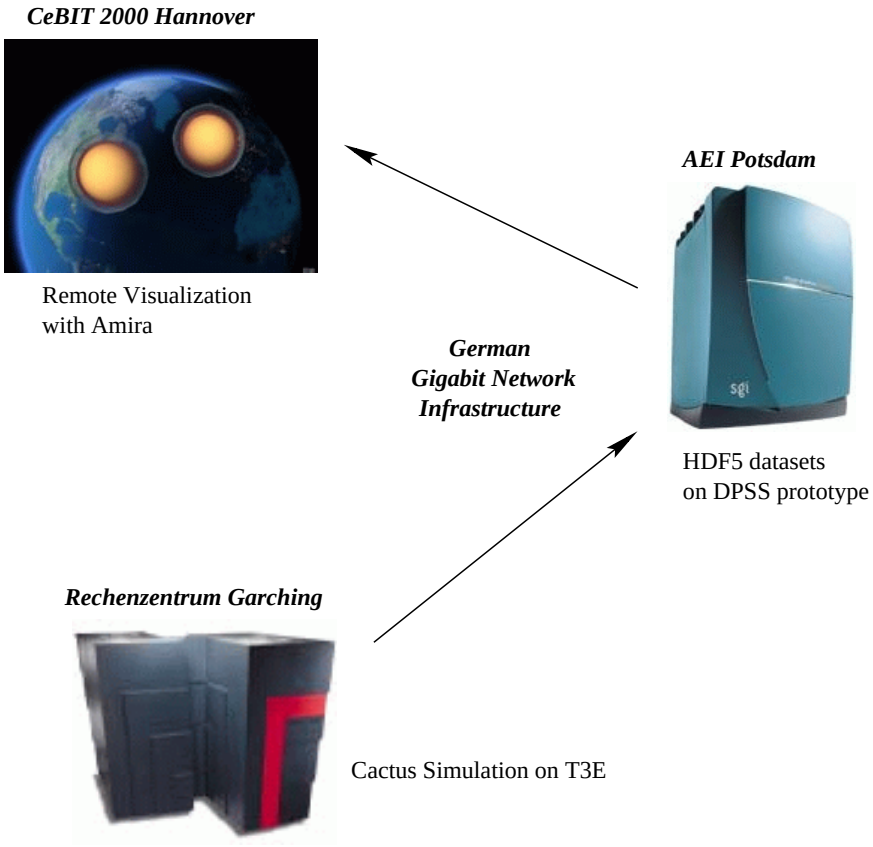


Figure 2: Remote file access and visualization demo presented at CeBIT 2000

for both file-based I/O and online data streaming.

### 3.4 Remote Steering

The Cactus Computational Toolkit contains a thorn *http* which allows any number of collaborators to connect a web browser to a running simulation and interactively change *steerable* parameters, such as I/O options. The web browser can also query information about the run, such as the current iteration step, available thorns and variables, and parameter choices. Simple line graphs of selected grid functions can be directly displayed.

In addition to this thorn *http* which lets Cactus act as a web server, we are working on more generic interfaces for remote monitoring and steering of Cactus simulations. The first approach is similar to thorn *http* but would use a generic data description protocol [14] based on XML rather than HTML. Optimized for communicating small amounts of data such as Cactus steering parameters, this layer will allow a wide range of already existing XML aware client applications to contact and control a remote Cactus simulation.

Another solution builds on top of HDF5 and our *Stream* VFD described above. For this the streaming interface is simply used in a bidirectional way: Cactus receives steering requests from remote visualization clients and sends back the requested data on the same socket connection using the standardized HDF5 API.

We are planning to integrate the remote steering capabilities as GUIs into the client applications to allow for more flexibility and interactivity for remote visualizations of Cactus simulations.

## Acknowledgments

The development of the Cactus Code is a highly collaborative effort, and we are indebted to a great many experts at different institutions for their advice, visions and support. The original design of Cactus was by Joan Massó and Paul Walker, since when it has been extensively developed at the AEI, NCSA and Washington University.

It is a pleasure for us to thank Ian Foster, Steve Tuecke, Warren Smith, Brian Toonen, and Joe Bester from the Globus team at Argonne National Labs (ANL) for their Globus and Data Grid work; Mike Folk and his HDF5 development group at NCSA who helped us in implementing the requirements of remote file access into their HDF5 code; Brian Tierney from Lawrence Berkeley Labs for his DPSS support; Jason Novotny at NLANR for his help with Globus and graphical user interfaces. Computing resources and technical support have been provided by AEI, NCSA, ANL, Rechenzentrum Garching/Germany, and ZIB.

## References

[1] Cactus Code: `http://www.cactuscode.org`

[2] Allen, G., Goodale, T., Lanfermann, G., Seidel, E., Benger, W., Hege, H.-C., Merzky, A., Massó, J., Radke, T. and Shalf, J., *Solving Einstein's Equation on Supercomputers*, IEEE Computer, p.52-59, December, 1999.
`http://www.computer.org/computer/articles/einstein_1299_1.htm`

[3] Seidel, E. and Suen, W.M., **J. Comp. Appl. Math.**, **109**, (1999), 493-525.

[4] DFN Gigabit Project: Tele Immersion, Collision of Black Holes: `http://www.zib.de/Visual/projects/TIKSL/`

[5] Astrophysics Simulation Collaboratory: `http://wugrav.wustl.edu/ASC/`

[6] Globus Metacomputing Toolkit: `http://www.globus.org/`

[7] Amira: `http://amira.zib.de/`

[8] FlexIO: `http://zeus.ncsa.uiuc.edu/~jshalf/FlexIO/`

[9] Hierachical Data Format Version 5: `http://hdf.ncsa.uiuc.edu/HDF5/`

[10] W. Benger, H. -C. Hege, A. Merzky, T. Radke, E. Seidel, *Efficient Distributed File I/O for Visualization in Grid Environments.* Proc. PDC99 *Simulation and Visualization on the Grid.*, Stockholm (1999) (unpublished)

[11] W. Benger, H.-C. Hege, A. Merzky, T. Radke, E. Seidel: "Schwarze Löcher sehen", DFN-Mitteilungen, Bd. 52, 2000

[12] The European Grid-Forum: `http://www.egrid.org`

[13] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets,* (1999) Submitted to NetStore '99

[14] A. Merzky, *Describing Data.*, Proc. 1st EGrid Workshop, Poznan (2000)