Cluster Computing 4, 179–188, 2001

Cactus Tools for Grid Applications

GABRIELLE ALLEN [a], WERNER BENGER [a,b], THOMAS DRAMLITSCH [a], TOM GOODALE [a], HANS-CHRISTIAN HEGE [b], GERD LANFERMANN [a], ANDRÉ MERZKY [b], THOMAS RADKE [a], EDWARD SEIDEL [a,c] and JOHN SHALF [c,d]

[a] *Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut (AEI), Golm, Germany*
[b] *Konrad-Zuse-Zentrum für Informationstechnik (ZIB), Berlin, Germany*
[c] *National Center for Supercomputing Applications (NCSA), Champaign, IL, USA*
[d] *Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA*

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, and detail how its design makes it an ideal application test-bed for Grid computing. We describe the development of tools, and the experiments which have already been performed in a Grid environment with Cactus, including distributed simulations, remote monitoring and steering, and data handling and visualization. Finally, we discuss how Grid portals, such as those already developed for Cactus, will open the door to global computing resources for scientific users.

Keywords: Cactus, Grid computing, Grid portals

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.

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

Of many applications using the Cactus framework, an important one which continues to drive its development is the solution of the Einstein Equations. The large and varied computational requirements of solving these equations for scenarios such as black hole or neutron star collisions, make them a good example for demonstrating the need for Grid computing, and an ideal testbed for developing new techniques. 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 and timestep. Considering that a sufficiently accurate solution of a full 3D

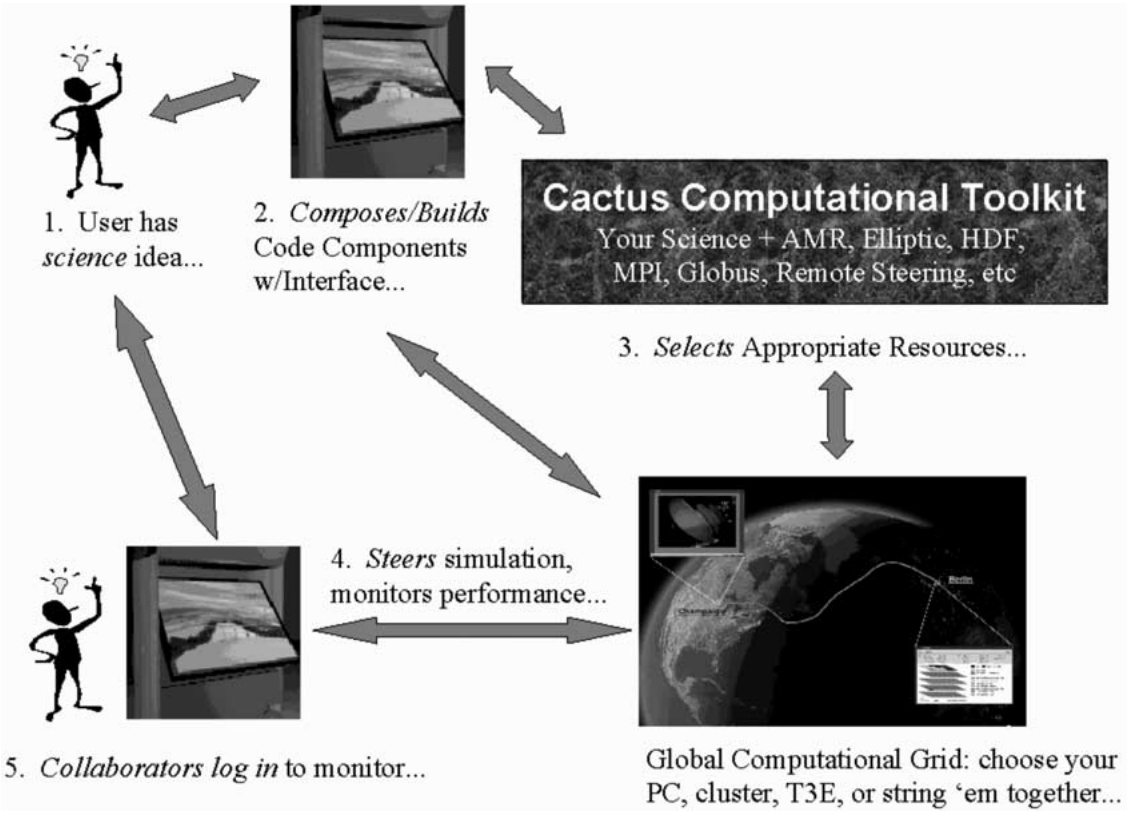

Figure 1. The dream of Grid computing: Grid infrastructure provides a transparent and flexible working environment providing access to global computing resources.

black hole problem will require at least 1000 grid points in each spatial dimension, this implies TeraByte/TeraFlop computers. Further, to analyze the large data sets created during a simulation requires advanced techniques in file management and visualization.

To date, the resources of the individual supercomputers 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 could 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 analyze 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 a web portal on her PDA to select the needed Cactus thorns, and to estimate the required computer resources. 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 simulation, 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 analyze 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 [1,4–6].

3. Grid Computing 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 parallelization across any number of supercomputers with precisely the same interface as it does to the resources of a single machine [7].

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.

Grid Computing 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 [8], remote steering and distributed I/O [9]. 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 utilize Grid computing for problems such as the collisions of neutron stars. The GrADs project [10], also funded by the NSF in the USA, is using Cactus as one of its applications for developing a Grid based computing environment. The European Grid Forum [11] has chosen Cactus as one of its 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 on the Grid

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, visualized, 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.

The latency and bandwidth are characteristic features, which determine the speed of a network. Cactus is aware of these features and can be fine-tuned in order to optimize communication. For example, if a network has a high latency but also a high bandwidth, many small messages can be coalesced into fewer bigger ones. When running in a metacomputing environment, one has to deal with different types of networks (shared-memory, distributed-memory, high-speed-network, LAN, WAN/internet) with different latency/bandwidth characteristics. Here, we also have the possibility to distinguish between these different types of network-connections in one single distributed run and tune Cactus communication patterns adequately. Partly this is already achieved by using `MPICH-G2`, the next-generation `MPI`-implementation, which can distinguish between processors located on one host (with native `MPI` installed) and processors separated by a LAN or WAN. According to the location, `MPICH-G2` can choose different protocols (`TCP` or vendor's `MPI`) for communication in one single distributed parallel run. Cactus can be used with this new technology without problem, which was demonstrated in many metacomputing experiments last year, including Supercomputing 2000 in Dallas.

A further aspect is load-balancing. Since different architectures provide different types of processors at different speeds, Cactus provides the ability to decompose the whole computational problem into sub-problems of different size, which fit the local processor power.

For running in a metacomputing environment at a real production-level, a User Portal has been built (described in section 5), making it possible to configure and start Cactus runs from one machine via a special web-based GUI. This greatly simplifies the situation for the scientist, since she does not have to deal with every detail of the local supercomputer, such as batch-systems, username/password. The portal provides automatic staging and compilation of the code on the local supercomputer, the distributed machines appearing as a single virtual machine.

3.2. Checkpointing distributed simulations on the Grid

Grid computing events in the past have been often understood as one-time, well prepared attempts to harness several machines at the same time. In a more realistic setting, compute resources of considerable size can not be expected to be available at a given time, instead their availability is a dynamic process. A true Grid application has to be capable of dealing with such dynamic allocations of resources.

The Cactus framework addresses this challenge by providing a sophisticated cross-platform checkpointing mechanism. In general, checkpointing technology allows the user to freeze the state of an application by writing a checkpoint file to disk, from which the application can be restored and continued at a later time.

In Cactus the checkpoint is not just the memory image of the application written to disk, as found in several other checkpointing systems, but the total set of user defined objects (variables, scalars, etc.). While memory images tend to be quite huge and are only compatible within the same class of operating systems and architectures, this approach allows for smaller, architecture-independent check-

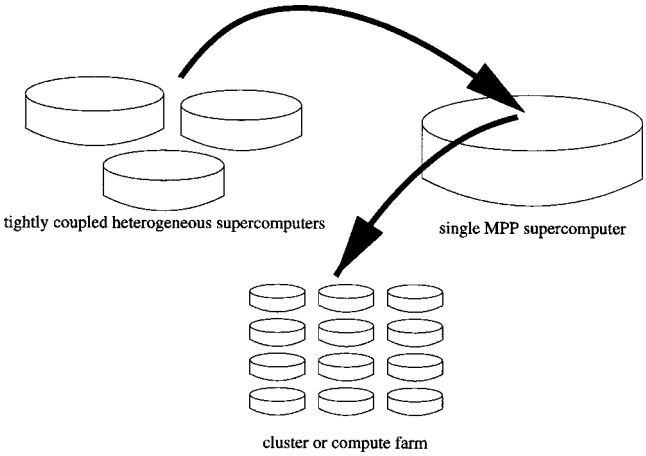

Figure 2. Migration scenario for a distributed simulation: the simulation starts on three tightly coupled supercomputers from where it is checkpointed and migrated to a single machine. The computation migrates again to finish on a cluster.

points. The cross-platform checkpoints of Cactus can be transferred between arbitrary architectures, operating systems and numbers of processors for restarting and continuing simulations.

The checkpointing mechanism is completely transparent to the user, who can request a checkpoints to be written at regular timestep intervals, at the end of the requested compute time allocation, or using a steering interface immediately at the current timestep. All of the internal technicalities of parallel I/O are hidden from the user. The user can control checkpoint behavior (as frequency or parallel I/O) by means of steerable parameters.

The checkpoint mechanism allows for the output of a single, global checkpoint file as well as multiple checkpoint files for each of the distributed machines. The mechanism makes use of parallel I/O where possible. For restarting, the multiple checkpoint files can be recombined into a single file which can be used to restart on an arbitrary set of machines. The parallel restart operation from multiple files is currently restricted to the same topology of machines. Future developments will add intelligent components to immediately restart from multiple checkpoint files across arbitrary machine topologies.

With respect to distributed simulations, a Cactus user has the ability to perform a distributed run and checkpoint this simulation even though is it being run on a heterogeneous machine set. A checkpoint file can then be transferred to a new configuration of machines to continue the simulation. The new pool of machines can differ from the previous one in type and number of machines involved as well as the number of processors. This flexible chain of distributed simulations is illustrated in figure 2: an initial simulation run across three tightly coupled supercomputers is checkpointed. The checkpoint file is transferred to a single MPP machine and restarted. After a second checkpointing event the third stage of the simulation is continued on a cluster system.

4. Grid-enabled communication and I/O techniques

The parallel driver layer in Cactus, which manages the allocation and domain decomposition of grid variables as well as their synchronization between processor boundaries, 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.

The current standard driver thorn is called `PUGH`, which uses `MPI` to provide parallelism. In order to perform distributed Cactus simulations on the Grid, this `PUGH` thorn is simply linked against the Grid-enabled `MPICH-G` [12] implementation of `MPI` which is available with the Globus toolkit. Thus, preparing a Grid-enabled version of Cactus is a compilation choice, and it is completely transparent for application thorn programmers to add their own code to a Grid-enabled Cactus. Using the Globus job submission tools, Cactus users can start their Cactus runs in a Grid environment just as easily as they do on a single machine.

The Cactus I/O subsystem is implemented in a similar, generic manner: the flesh provides a runtime interface for arbitrary I/O thorns to register their own, specific I/O methods. These methods can then in turn be invoked by the flesh or any application thorn to read external data into Cactus variables or dump their contents to a storage medium for postprocessing analysis and visualization purposes.

The I/O thorns currently available in the computational toolkit provide methods to write simulation data in different formats (1D traceline plots, 2D slices and JPEG images, full N-dimensional arrays, arbitrary hyperslabs of N-dimensional arrays, reduction scalars (e.g., minimum/maximum values), isosurface geometry data, particle trajectories, runtime standard output) also using different I/O libraries (`FlexIO` [13], `HDF5` [14], JPEG, ASCII). Further methods or libraries can easily be added by thorn programmers.

In the following sections we will describe in more detail the Grid software techniques we have developed to date which allow Cactus users to easily perform postprocessing analysis on data produced by a remote Cactus simulation, and also to monitor and steer running Cactus jobs remotely. A general overview of the final proposed architecture of our Grid-enabled I/O system is shown in figure 3.

The *Hierarchical Data Format version* 5 (`HDF5`) plays a key role in this overall picture. `HDF5` has become a widely accepted standard in the scientific computing community for storing data. It defines a very flexible file format and provides an efficient software library for managing arbitrary multidimensional datasets of various types. Raw data access is accomplished via a generic *Virtual File Driver* (VFD) layer in `HDF5`. Beneath this abstraction layer exists a set of low-level I/O drivers which provide different ways of accessing the raw data of an `HDF5` file, either located on a local disk or on other storage media. We have added our

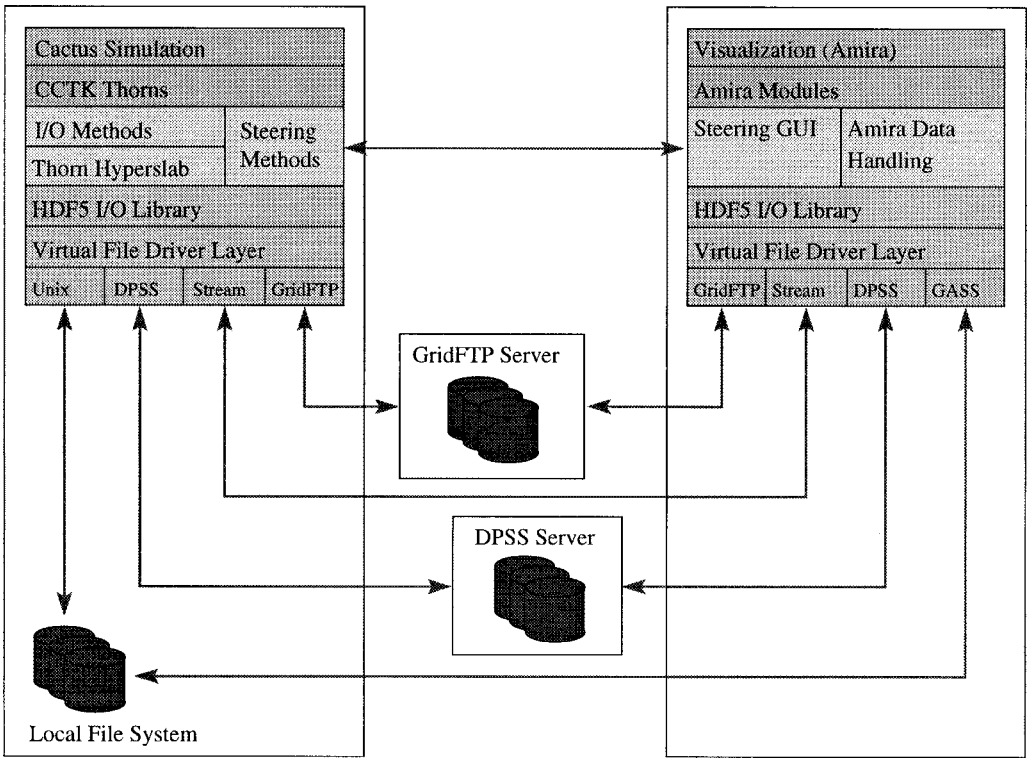

Figure 3. General overview of the Grid-enabled I/O architecture.

own drivers to this layer which enable existing applications to have the additional capability of accessing remote data residing anywhere on the Grid.

4.1. Direct remote file access

`HDF5` already has a `GASS` driver (Global Access to Secondary Storage) which automatically stages complete remote files first to the local machine and then operates on their local copies via standard UNIX file I/O. This method is feasible for small or medium sized data files. However, large-scale computer simulations often generate large-scale data sets – single simulations may generate files containing several hundreds of GBytes, up to the order of a TByte as machine resources increase. Conventional postprocessing analysis then becomes prohibitively resource-intensive when remote simulation data must be staged for local processing. Further, in many cases, for example for first-sight visualization purposes, only a small fraction of the overall data is really needed. For example, in a simulation of the 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 one might want to analyze only the first time step of one or two of the variables. Or, in order to perform a quick pre-analysis of high-resolution data, it might be sufficient to downsample the array variables and fetch data at only every other grid point.

By enhancing the `HDF5` VFD layer with a driver that builds on top of the Data Grid software components [15] from the Globus toolkit we enable existing I/O layers to operate on remote `HDF5` files directly. These are uniquely addressed by their URL, and after opening them with the appropriate driver, all read and write operations are performed as network transactions on the Grid – completely transparent to the application. Using the data selection capabilities of `HDF5` (defining so-called hyperslabs as arbitrary rectangular subregions in the multidimensional data sets, optionally with downsampling and type conversion applied) individual time steps and zones of interesting data can be read and visualized in a very efficient and convenient way.

The Data Grid client software only supports remote partial file access to *Distributed Parallel Storage Systems* (DPSS) [16]. During Supercomputing 1999 in Portland and at CeBIT 2000 in Hannover we successfully demonstrated the feasibility of such a DPSS Data Grid Infrastructure. In these demonstrations, Cactus simulation data residing on remote DPSS data servers was visualized by an `HDF5`-enabled version of the visualization package Amira [17]. This is illustrated in figure 4.

Remote access to files which are located anywhere on the Grid will soon be provided by a `GridFtp` driver [18] which supports the standard FTP protocol, enhanced with partial file access, parallel streaming capabilities, and Grid security mechanisms.

Another challenge occurs when 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, using a global address space having pointers to pieces at other locations. We plan to also tackle these problems with the Data

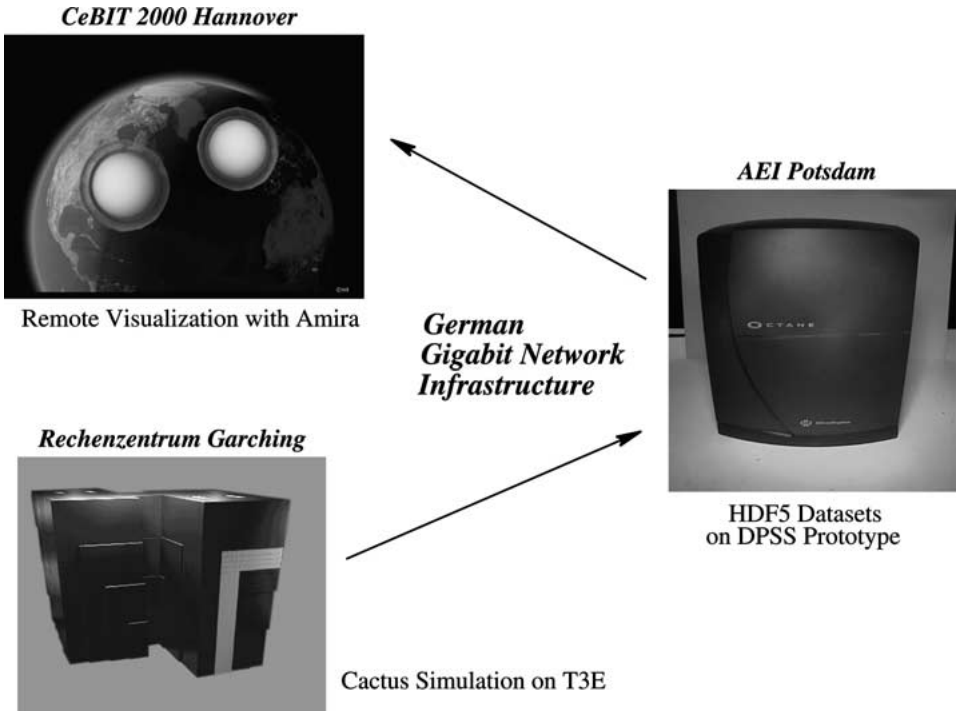

Figure 4. Remote file access and visualization demo presented at CeBIT 2000.

Grid components, by organizing related files as collections of logical file instances. The DataGrid project of the Globus group is investigating such techniques [15].

4.2. Remote online data streaming and visualization

Cactus also provides the capability to stream online-data from a running simulation via `TCP/IP` socket communications. This can be used for many purposes. To date, the most common use for live data streaming is for remote visualization, which is our focus here. However, in our vision of future Grid simulations, we expect running simulations to communicate with each other, migrate from machine to machine, spawn off additional processes on the Grid, etc. Hence, we expect that data streaming will be a fundamental enabling technology for future Grid simulations. We are now exploiting the data streaming capabilities we describe here to enable such advanced Grid simulations, as demonstrated in our "Cactus Worm" scenario where a running Cactus simulation was able to migrate itself, using the data streaming techniques described below, from site to site across the European Egrid [19]. This is a simple example of more sophisticated types of Grid simulations, based on data streaming, that we will be developing in the future. But in the remainder of this section we focus on data streaming for use in remote visualization.

Multiple visualization clients can then connect to a running Cactus executable via a socket from any remote machine on the Grid, request arbitrary data from the running simulation, and display simulation results in real-time, visualizing for example photons falling into a black hole, or isosurfaces of gravitational waves which are emitted during a black hole collision.

Data streaming is integrated into Cactus in several different ways. One method is to access Cactus output files while they are being written by the running simulation. Those files are registered with the `HTTP` control interface, described in the following section, and can be downloaded to any web browser. For example, simple 1D data graphs can be viewed by simply clicking on a download file and firing off, for example, an xgraph program. Two-dimensional JPEG images can be viewed directly in a web browser, and continuous time sequences of JPEGs can be displayed using the auto-refresh option of the capable browsers.

Another technique implements a proprietary communication protocol for sending specific geometry data such as isosurfaces or particle trajectories down a raw socket connection to a visualization program [20]. This is illustrated in figure 6. Precomputing such data at the simulation side not only allows for parallel rendering of images but also reduces the amount of data to be transferred to remote visualization clients.

The most generic approach for streaming arbitrary data of any type is again based on the `HDF5` I/O library and its VFD layer. We have developed a *Stream* driver which holds the `HDF5` data to be streamed out of the Cactus simulation as an in-memory `HDF5` file. On a flush/close operation the entire file is sent through a socket to the connected client. In the client application, the same driver is used to reconstruct the in-memory file which then can be accessed as usual to read the `HDF5` datasets.

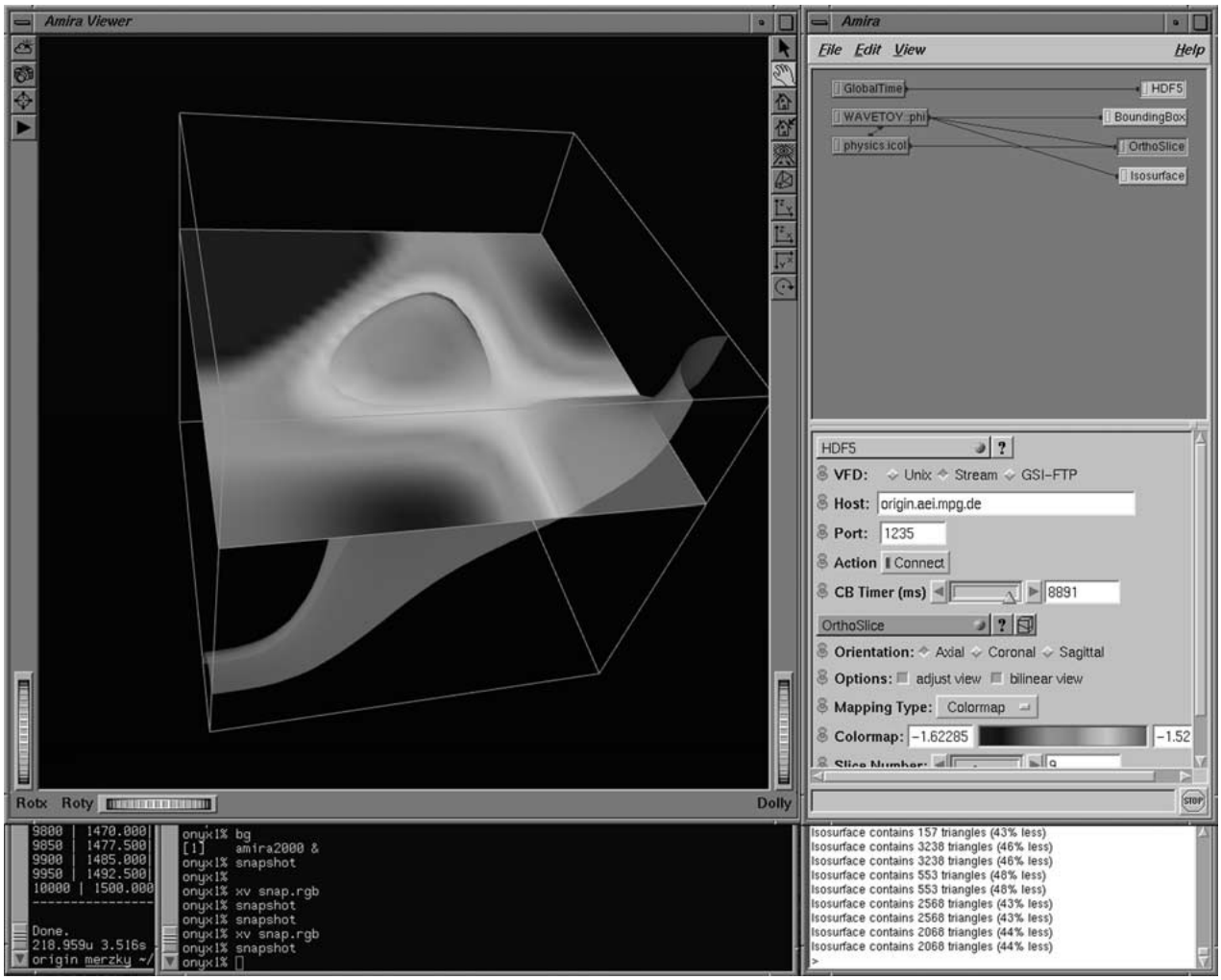

Figure 5. Online visualization: The Amira visualization toolkit [17] allows a user to visualize slices through a complete 3D data set streamed from a running Cactus simulation, and at the same time to display an isosurface obtained online for the same 3D field.

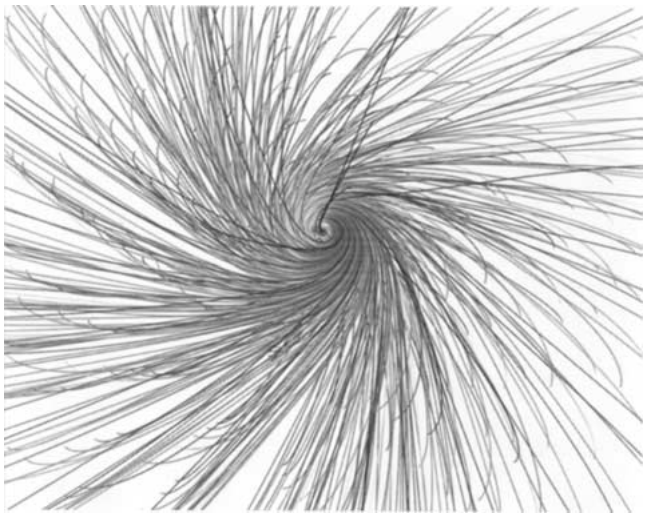

Figure 6. Trajectories of freely falling particles in the vicinity of a rotating black hole. The particle positions are streamed to the visualization tool in real-time during computation. This was demonstrated with Cactus and Amira at IGrid 2000 in Yokohama [20].

Since the VFD layer hides all low-level I/O operations from the upper layers of the `HDF5` library and from the application that builds on top of it, applications can use their existing `HDF5` file-based I/O methods immediately for online remote data access without changing their I/O interfaces. This has been demonstrated using different visualization toolkits, including Amira [17], the IBM Data Explorer [21], and LCA Vision [22].

The *Stream* driver is capable of sending data simultaneously to multiple clients. This is one key component for building a collaborative visualization environment where scientists at different sites can analyze the results of a remote simulation either by looking simultaneously at the same data or by requesting different views of it. We are working on a design for a more sophisticated I/O request protocol and the implementation of an external data server which will handle multiple clients and can also serve requests individually. By integrating intelligent data management and caching strategies, such a server would relieve the simulation from communication overhead and help to reduce data traffic in general.

4.3. Remote monitoring and steering

The Cactus Computational Toolkit contains a thorn `HTTPD` which can be added to any Cactus simulation to provide an inbuilt `HTTP` server. Pointing their web browsers to a URL identifying a running Cactus job on a remote machine, any number of collaborators can connect to monitor and steer the simulation online.

The provided Cactus web interface allows users to query certain information about the run, such as the current iteration step, a list of available thorns and variables, and a full description of all parameters and their current settings. After successful authorization a user can also interactively change parameters which are marked as *steerable*. At each simulation cycle these parameters are checked, and the appropriate thorns may react on changes individually. Most of the I/O parameters are steerable. This enables users to selectively switch on or off specific output at runtime, dynamically choosing which variables are output using which I/O method. I/O options such as hyperslabbing or downsampling parameters may also be modified in order to adjust online data streaming to remote visualization clients. The web interface can also be used to pause the simulation, optionally when a chosen condition is satisfied, and to advance the simulation by single timesteps.

The Web interface provided by thorn `HTTPD` is dynamically extensible in that any thorn can register and update its own `HTML` pages at runtime. Besides a download page for Cactus output files there is a also viewport available which embeds dynamically generated JPEG images.

Another steering interface again builds on top of `HDF5` and the *Stream* driver described above. For this interface the data streaming is simply used in a bidirectional way: Cactus writes parameters into an `HDF5` file which is then streamed to any connected steering client. After some user interaction, this client sends back a modified version of the parameter file which is read and evaluated by Cactus.

Because of its self-describing data format and the flexibility to add additional, user-defined information, `HDF5` also provides the possibility to build more advanced steering clients with graphical user interfaces. As an example, minimum/maximum values could be assigned to numerical parameters to create sliders for more convenient user interactions. Parameters belonging to one thorn could be sorted into a group hierarchy for building menus.

These features of `HDF5` make it relatively easy to implement dynamic graphical user interfaces for arbitrary Cactus parameter sets which adapt themselves to the current Cactus configuration. We are actively working on including such user interfaces into existing visualization clients. Their steering capabilities would then not only be limited to exchanging `HDF5` parameter files but could also be extended to feed back any kind of data fields into Cactus, for instance to add photons to a black hole simulation to locate an event horizon.

5. Portals onto the Grid

The Grid is only useful as a concept if its services can be used to create the illusion that all of the resources are centralized to the user's workstation. So the most successful distributed applications on the Grid will paradoxically be those which make the user least aware that they are in fact operating in distributed fashion. The motivation for producing a Grid Portal interface to a PSE like Cactus is derived from the desire to hide distributed applications and immensely complex distributed/parallel software architectures behind a single point of presence, and to make them accessible through comparatively simple client-side interfaces.

A portal is a single point of presence (typically hosted on the web) which can be customized for a particular user and remembers particular aspects of the customizations regardless of where the user accesses it from. Yahoo and HotMail are typical consumer-oriented examples of this capability and are in fact the originators of this new meaning for the term *portal*. It doesn't matter where you are, when you login to the URL of these portals, you get access to the same view of your personalized environment and data (i.e., your email).

Placing the PSE like Cactus within a portal creates a universally accessible interface to your scientific computing platform. The GUI is user-customizable, as if it were a desktop application on the user's own workstation, except that the same customized environment is accessible from virtually any location by simply connecting to the same URL address. A *science portal* has the additional implied function of automating the entire workflow for a particular scientific application, from initial data generation, to selecting resources to run the application, to archival storage management and analysis of the results of those simulations.

This replaces a series of perhaps loosely (or usually poorly) integrated tools with a comprehensive environment which is customized around a particular application. Finally, a *collaboratory* provides additional facilities for sharing information either online or asynchronously among users of the portal.

Cactus has several key advantages which make it very suitable as the basis for a portal design. Its modular design supports dynamic assembly of applications online through a simplified GUI. Its sophisticated multiplatform compilation makes it very simple to run the code on any available Grid resource without the complexities of Imake or the performance penalty of a Virtual Machine. Its centralized revision control mechanism permits efficient sharing of code, software updates, and bug fixes. Finally, the integrated visualization for remote monitoring and steering of the code through a web interface allows seamless integration of these capabilities with the portal's web-GUI.

The Astrophysics Simulation Collaboratory Portal [5] is a concrete use of Cactus within a web portal GUI which leverages off of technology which was developed originally for e-commerce applications. The architecture utilizes a commercial-grade StrongHold (Apache) webserver which

offers SSL encryption using a site certificate from a commercial Certificate Authority. Running side-by-side with the webserver is a TomCat JSP engine which offers a cleaner means to manage automation in an elegant and easily maintainable fashion. JSP allows us to directly execute methods of server-side Java beans rather than the typical CGI-script methodology of parsing the state of form elements individually after an `HTTP 'POST'` event. The Java beans directly call Java CoG [6] which is a pure Java implementation of the Globus toolkit to extend the automation to Grid resources. The user-state within the system is maintained by a backend database system (`OpenLDAP` or `mySQL`) which allows simple replication of portal state allowing the web services to be scaled through server replication.

The *science portals* and `collaboratories` will play an increasingly important role in HPC as the Grid evolves. The natural point of organization for user communities in an HPC environment is a particular field of science or a particular application code, just as experimental laboratories bring together top researchers who are interested in the same or similar lines of research. The internet has provided us with access to enormous remotely located resources, but this has shifted the center of focus to the particular HPC site and its operating environment, batch queues and security policies rather than the science that is computed there. The single point of presence offered by an Grid portal recreates the traditional laboratory environment where scientists who share similar interests and applications are brought together under the umbrella of a shared resource; a `collaboratory`. The portal itself is a distributed Grid application for a specific community of scientists rather than a general-purpose resource. So unlike traditional web-portals where you have an implicit offer that *if you go to* `www.<my_portal_location>.org` *we'll do everything for you here using our compute resources*, the Grid portal's business plan can be simply stated as *go to* `www.<my_portal_application>.org` *and we will do everything you need for* `<my_application>` *regardless of your location and that of your compute resources*. This returns the focus of a scientific community to the scientific application rather than the location of the HPC resources.

If the Grid is really working, in another 5 years we will no longer think of the, for example, NSF supercomputing centers as distinct sites like SDSC, NCSA, or PSC. We will instead think only of the particular application collaboratories which have been set up to study different scientific applications. That resource will merely be a *name* rather than a *place*.

6. Summary

The Cactus Code and Computational Toolkit and the large scale applications which it serves provide a ideal laboratory for developing and testing new Grid techniques and working practices. Cactus can be very easily configured and run in a Grid environment, and the tools developed so far already provide many capabilities for exploiting global computing resources. The infrastructure and tools developed are immediately available to the user community for testing, and many are already being successfully and beneficially used by collaborations researching computationally intensive problems such as black hole and neutron star collisions.

Acknowledgements

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; and Michael Russell at the University of Chicago for his Portal work. Computing resources and technical support have been provided by AEI, ANL, NCSA, Rechenzentrum Garching/Germany, and ZIB.

We greatly acknowledge financial support for André Merzky and Thomas Radke as well as provision of a gigabit network infrastructure in the course of the TIKSL research project by DFN (German Research Network).

References

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

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

[3] E. Seidel and W.M. Suen, Numerical relativity as a tool for computational astrophysics, 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://www.ascportal.org/ASC`

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

[7] W. Benger, I. Foster, J. Novotny, E. Seidel, J. Shalf, W. Smith and P. Walker, Numerical relativity in a distributed environment, in: *Proc. of the 9th SIAM Conf. on Parallel Processing for Scientific Computing*, March, 1999.

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

[9] W. Benger, H.-C. Hege, A. Merzky, T. Radke and E. Seidel, Efficient distributed file I/O for visualization in Grid environments, in: *Simulation and Visualization on the Grid*, Lecture Notes in Computational Science and Engineering, Vol. 13, eds. B. Engquist, L. Johnsson, M. Hammill and F. Short (Springer, 2000) pp. 1–16.

[10] Grid Adaptive Development Software (GrADS), http://www.isi.edu/grads/
[11] The European Grid-Forum, http://www.egrid.org
[12] Grid-enabled MPICH Implementation, http://www.globus.org/mpi
[13] FlexIO, http://zeus.ncsa.uiuc.edu/~jshalf/FlexIO/
[14] Hierachical Data Format Version 5, http://hdf.ncsa.uiuc.edu/HDF5
[15] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, The data Grid: towards an architecture for the distributed management and analysis of large scientific datasets (1999), submitted to NetStore '99.
[16] Distributed Parallel Storage System, http://www-didc.lbl.gov/DPSS
[17] Amira – Users Guide and Reference Manual, *AmiraDev – Programmers Guide*, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) and Indeed-Visual Concepts, Berlin, http://amira.zib.de
[18] The Globus Project: GridFTP: Universal Data Transfer for the Grid, White Paper, http://www.globus.org/datagrid/deliverables/C2WPdrafts.pdf
[19] G. Allen, T. Dramlitsch, T. Goodale, G. Lanfermann, T. Radke, E. Seidel, T. Kielmann, K. Verstoep, Z. Balaton, P. Kacsuk, F. Szalai, J. Gehring, A. Keller, A. Streit, L. Matyska, M. Ruda, A. Krenek, H. Frese, H. Knipp, A. Merzky, A. Reinefeld, F. Schintke, B. Ludwiczak, J. Nabrzyski, J. Pukacki, H.-P. Kersken and M. Russell, Early experiences with the Egrid testbed, in: *IEEE Int. Symp. on Cluster Computing and the Grid*, 2001.
[20] Geodesies in Kerr Space-Time, Presentation at the IGrid 2000 conference in Yokohama, Japan, http://www.zib.de/geodesics
[21] IBM Data Explorer, http://www.research.ibm.com/dx
[22] LCA Vision, http://zeus.ncsa.uiuc.edu/~miksa/LCAVision.html