

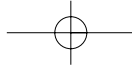
16 Collaborative Science: Astrophysics Requirements and Experiences

CHAPTER

Gabrielle Allen and Edward Seidel

Astrophysics is a natural driver for extreme computing technology. From simulations of the universe and its constituents to analysis of vast amounts of data collected by observatories of various kinds, astrophysics has always been at the forefront of computation. Calculus itself was developed by Newton as a “computational technology” to understand the dynamics of the solar system, and astrophysicists led the creation of the U.S. supercomputing centers program in the early 1980s (404). Thus it is not surprising that astrophysicists are today a driving force behind the development of the Grid. In this chapter, we explain how this symbiosis between astrophysicists and Grid technologists contributes to the development of both disciplines.

Astrophysicists use computational tools to understand exotic processes in the universe, from localized processes (star formation, solar physics, supernovae, collisions of or accretion around neutron stars and black holes, gravitational waves, and γ -ray bursts) to collections of all these (the formation and interactions of galaxies) to dynamics of and structure formation in the universe itself. Each process is complex, drawing from a vast fundamental knowledge base that includes nuclear, atomic, and particle physics, astrochemistry, relativistic magnetohydrodynamics, radiation transport, and general relativity. In some problems (e.g., γ -ray bursts), *all* of these elements may be needed to describe the real-universe processes. Modeling these different processes requires a wide range of both physical theory and algorithmic techniques (e.g., finite differences, finite elements, spectral methods, N -body approaches, smooth-particle hydrodynamics). Processes may occur on multiple and dynamically changing time and length scales that are intricately intertwined.



The resulting computational problems are immense; a realistic calculation capable of predicting what actually happens in nature dwarfs the capacity of existing supercomputers. Most of these problems are intrinsically large scale and three dimensional and thus require highly parallel codes to run efficiently on advanced computer architectures. When 3D astrophysical processes are modeled on large-scale computers, they usually generate huge amounts of output data. Storing, managing, visualizing, analyzing, and ultimately understanding the output—and its relevance to any observational data—can be overwhelming.

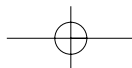
Astrophysics is also increasingly collaborative: the physics is often so rich and varied that no single group or community has enough expertise to tackle every part of a problem. Larger and larger collaborations among groups with different expertise are required to give a realistic description of these processes. As a result, astrophysicists are heavily involved in collaborative virtual organizations (281) (VOs: Chapter 4), with distributed colleagues sharing (usually in ad hoc ways) resources, data, codes, knowledge, and expertise.

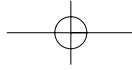
Astrophysics is also an observational science (Chapter 7). A large and varied array of detectors, satellites, and observatories is creating a vast amount of data. These data sources need to be linked together and, ultimately, connected to simulations of the sources from which data are collected. This interplay between a multitude of prodigious data sources (both archived and live from different experiments) and both simulation and analysis processes (that themselves may require computational power that dwarfs what is available on today's largest systems) is what makes Grids so important to the astrophysics communities. The available computing power of any single site is too limited, and the communities and data are widely distributed. An environment of seamlessly integrated computational resources, continuously updated data archives, and communities of scientists and engineers are exactly what is required for the astrophysics and many other disciplines—and exactly what is promised by the Grid.

16.1 NUMERICAL RELATIVITY AND ASTROPHYSICS

To provide a definite focus (and to draw on our own expertise), we concentrate on Grid approaches being used and developed *now* in the numerical relativity community. The approaches and tools we describe are, however, applicable across a wide range of applications in computational astrophysics and beyond.

Numerical relativity, the numerical solution of Einstein's equations for general relativity, is demanding computationally and requires a broad expertise base drawing from large, dispersed collaborations of both physics and computer





16.1 Numerical Relativity and Astrophysics

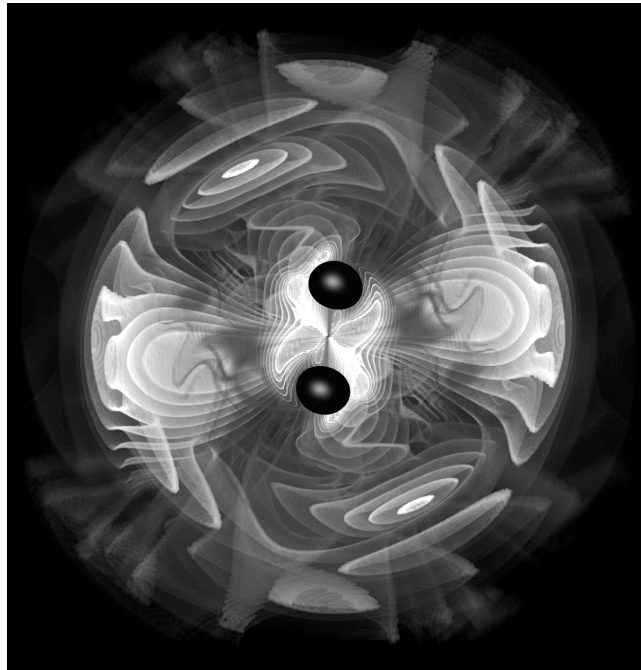
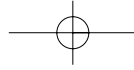
203

scientists (73). Interest in the field is currently high, since we will soon be observing gravitational waves with large-scale laser interferometric detectors (98).

Today, crude simulations of sources of gravitational waves, such as colliding black holes or neutron stars, supernova explosions, and other phenomena, are barely within reach of available computational resources and the communities that use them. Large groups and collaborations, such as those clustered around the European Astrophysics Network, run simulations on a daily basis on both workstations and supercomputing resources worldwide. Sharing a common code base (based on the Cactus framework: see the following section), groups of researchers create simulation codes by assembling modules, for example, for evolving the gravitational field, hydrodynamics, analysis of gravitational waveforms, parallel I/O and communication, and remote control and visualization. These codes must be extremely portable: The same code must run equally well on an un-networked home laptop in Greece for development and testing; a workstation in Germany; or on any subset of a dozen large (1000 + processor) supercomputers worldwide. Such needs motivate interdisciplinary and international common formats, protocols, and access, especially for data and information (Chapters 22 and 23).

Computational resource needs will increase far beyond those just described as projects move from development to real production: for example, to calculate accurate waveforms for gravitational wave observatories. Such projects will also require even larger collaborations, because the mathematical, physical, and computational knowledge needed to perform a single simulation—with as much realistic physics as is required to model a real astrophysics event—is so vast. The difficulties just described all then increase dramatically, and functioning Grid technologies and tools will be crucial for success.

Computational astrophysicists have different requirements for the Grid according to whether they *develop* or *run* codes. Those developing Grid-enabled codes need tools and procedures for diagnosing and solving the problems that arise in this complex environment. Those running codes are more interested in reliability, ease-of-use, and new functionality such as remote techniques for interaction and collaboration. For example, a user simulating some exotic astrophysical process must be able to submit a job to an appropriate resource. When the job starts, it must be able to notify interested users distributed across the world; while it runs, various users may need to monitor its status and change the parameters; and after it has completed, the potentially huge amounts of data generated must be archived, visualized, and analyzed, again by many scientists. We emphasize that technologies that meet some of these needs exist: they were used, for example, in one of the largest simulations to date of two orbiting, coalescing black holes: see Figure 16.1.

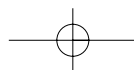


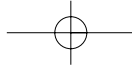
16.1
FIGURE

Two orbiting black holes, at center, are about to collide in this visualization of one of the largest simulations to date of this process. The emitted gravitational waves (which researchers hope to detect this decade) are shown as swirls of color. This simulation required well over 100 Gb of memory, generated a terabyte of output data, and required more than a day on thousand-processor machines at the National Center for Supercomputing Applications (NCSA) and the National Energy Research Scientific Computing Center (NERSC). Grid technologies were used to remotely monitor and perform basic visualizations of this production simulation, while it was running, by a collaboration of astrophysicists in different countries. (Image provided by AEI/ZIB).

16.2 CACTUS: AN APPLICATION FRAMEWORK FOR THE GRID

Developing large codes is challenging even on today's independent resources due to the need to understand potential different problems arising on different architectures or different numbers of processors. On the Grid, the problems are multiplied. We require clear and complete logging information, the ability to recreate a specific environment in order to reproduce specific results, and Grid-enabled





16.2 Cactus: An Application Framework for the Grid

frameworks and toolkits that hide the complexity of the Grid and provide application-oriented APIs for functionality. The Cactus application framework that we describe here addresses some of these concerns. Others are being addressed in the European GridLab project (71, 581), which is developing a “Grid Application Toolkit” that will provide abstract APIs for common Grid operations, such as moving and archiving files, searching for suitable machines, and migrating running codes.

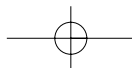
Although Grid technologies should work with legacy codes and current working practices, ultimately applications need to be written and developed in new ways if they are to fully leverage the potential of the Grid. Further, the way that we as scientists use our codes, utilize our resources, and conduct our collaborations also needs to be rethought. The Cactus application framework (318) includes features suitable both for prototyping Grid use and for exploiting the Grid as technologies mature. We find that many capabilities needed for Grid execution coincide with those required for running and working successfully in today’s computing collaborative environments, such as abstraction, portability, modularity, checkpointing and restart, flexible I/O, information interfaces, and parameter steering.

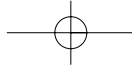
Cactus is an open source, generic problem-solving environment designed to provide scientists and engineers with a portable and collaborative framework for high-performance computing. Cactus emerged from the numerical relativity community, but is now used by a growing number of applications in science and engineering. The simulations of black hole collisions presented in Figure 16.1 were performed using a code developed in the Cactus framework.

The Cactus design was motivated primarily by computational issues facing researchers needing large-scale resources and involving sizable development and user teams. Its resulting modular structure easily enables parallel computation across different architectures and collaborative code development between different groups. Modules, or “thorns” in Cactus nomenclature, can implement different parts of custom-developed scientific or engineering applications, such as numerical relativity. Other thorns from standard computational toolkits provide a range of computational capabilities, such as parallel I/O, data distribution, or checkpointing.

Although Cactus is used today mainly in traditional supercomputing environments, many of its design features enable it to make good use of Grid technologies and to prototype new Grid scenarios. In particular, Cactus is highly portable (for example, it runs on iPAQs and Playstations) and has a configurable and scriptable build system, a built-in information Web server and steering API, and robust platform-independent checkpointing and restart capabilities.

The Cactus I/O layer maps parallel I/O to different libraries and new file formats. HDF-5 support for platform-independent hyperslabbing and downsampling





has proved particularly useful for Grid work. We describe Grid usage scenarios requiring these capabilities in Section 16.6. The parallel driver layer is implemented by a thorn, and the default MPI-based PUGH driver can be replaced by drivers based on other paradigms. PUGH can be linked with MPICH-G2 (403) (Chapter 24) for distributed computing (see Section 16.5).

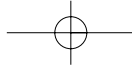
16.3 RESOURCE SHARING WITHIN VIRTUAL ORGANIZATIONS

In the remainder of this chapter, we use a series of usage scenarios to illustrate opportunities for Grid usage in computational science. Although motivated by computational astrophysics, these scenarios are applicable across many disciplines. In each scenario, we ground our presentation firmly in reality by describing how today's technologies have been used to enable the activity now, and then present our vision of how the activity can evolve as Grid technologies are fully deployed. Based on what is possible now, and sometimes even in prototype form, we believe that these apparently futuristic scenarios are not far from being realized.

We first consider issues relating to resource sharing within collaborative communities. The VO concept introduced in Chapter 4 is central to our work. We consider a VO as a collection of resources managed independently, but cooperatively, and available to some user community. Effective organization and use of these resources is of prime importance to this community.

The simplest and currently most important usage scenario for computational astrophysics on the Grid is probably that of *resource discovery and job submission*. A scientist in the EU Astrophysics Network has prepared an executable and input parameter file to simulate a black hole merger. With access to many large-scale computing facilities around the world, typically all with different userIDs, passwords, operating, queuing, and file systems, simply choosing where to run this simulation is complicated. Often researchers will use the resource they used the day before, regardless of whether it is the *best* available now (e.g., least loaded, most appropriate for current job).

Grid technologies simplify the process of accessing these resources. As discussed in Chapter 4, certificate-based authentication allows users to access all systems in their VO with a single log-in, which is then mapped to their local account. Similarly, other local idiosyncrasies can be removed: common Grid-enabled interfaces to local batch systems, file systems, data archiving, and so forth create the notion of unified user commands for all resources. At present the basic Grid technologies to support this process are well developed and robust in the Globus Toolkit, and are rapidly being deployed across many sites.



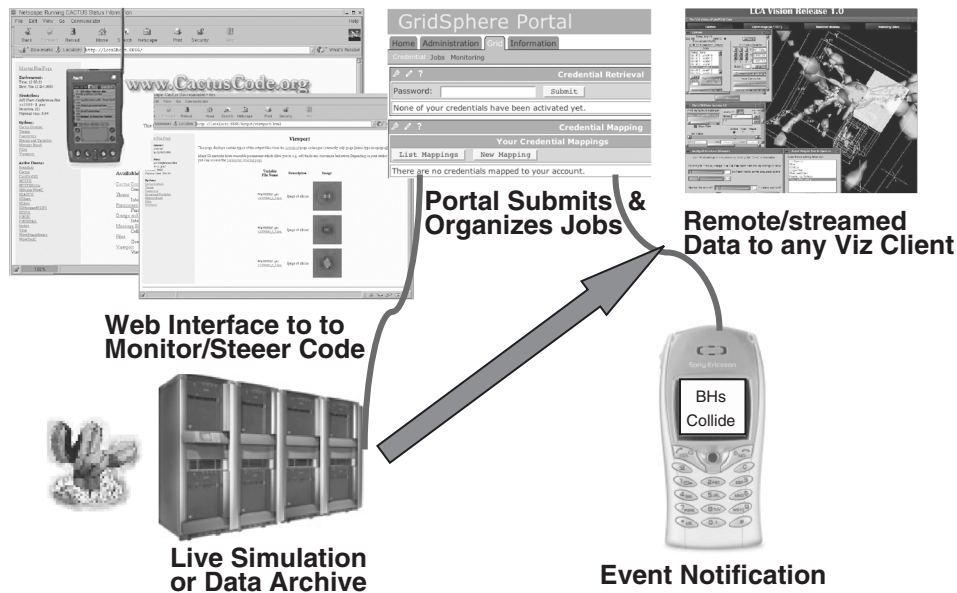
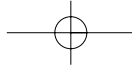
Going one step further, it is natural to provide access to, and interaction with, all the resources in one's VO through a Web-based *portal* (Chapter 24). Once authenticated, the user can in principle see all available resources and submit preconfigured jobs to a given machine. We use such a portal developed within the NSF Astrophysics Simulation Collaboratory project (567) to perform these and other functions.

A unified, single Grid log-in system, with all relevant information and job submission mechanisms collected in a point-and-click Web interface, is a major step forward. The next step is to allow the user to formulate a resource request (generated manually or automatically, based, for example, on statistics of the last similar job) that is then forwarded to a VO-aware resource broker service (Chapter 18) that can automatically route the job to the *most appropriate* resource. If these resource brokering services can be accessed by the applications themselves, then changing application needs or machine loads can trigger requests for other resources, to be found and used on demand, as we describe next.

16.4 INTERACTING WITH GRID JOBS

Traditionally, the computational scientist running a large simulation first submits a job to the batch system on a remote machine and then periodically logs onto that machine to check status. Maybe the job is still waiting in the queue, maybe it is actually running, or—as frequently occurs—maybe it has been terminated for an unexpected reason. Such working practices are frustrating for users. A job sits in a queue for days and then fails upon execution because of a simple error in initialization. Or perhaps it runs for 72 h and then aborts because it had been preset to output too much data and a disk quota was exceeded. Worst of all, it may have used 100,000 CPU-h but produced nonsensical results because of an incorrect parameter setting. These considerations motivate the development of more sophisticated interactive job monitoring and steering that will soon enable much more innovative Grid applications, such as those described below.

An information and steering interface has been developed for Cactus that allows any user with a Web browser to interact with a running job. The HTTPD Cactus thorn is a Web server that can display all current information about the running job, including active routines, their version numbers, time step, estimated time to completion, and written data files and can even include visualizations embedded in the Web pages. Any parameters declared *steerable* can be changed on the fly through a Web form interface; I/O frequency, variables output, downsampling, and any other such parameter can be changed at will, allowing the



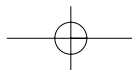
16.2

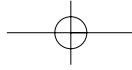
FIGURE

A user can submit jobs from a portal, and jobs are registered with the portal when they start, allowing access by all participating collaborators. Jobs can be monitored, and steered or their output data visualized, either live or from data archives. Important events in the life of a job trigger notification to various types of devices.

user to correct many problems, *without having to restart*. When a job starts, its URL is broadcast to a portal, which then notifies a user-defined group of collaborators via e-mail or SMS message with contact information that they can then access the simulation from a browser. Significant simulation events such as the merging of two black holes can be programmed to trigger notification of a user or group of collaborators.

A portal thus becomes the organizing instrument for *collaborative* computational science. Jobs can be grouped by collaboration, by topic, by status (currently running or archived output data), and so forth. Hyperlinks to data produced by the simulation provide instant access for visualization. When such links are clicked, data residing in a file is downloaded from the remote site to the local user machine, and the appropriate visualization client is automatically launched. If, on the other hand, the data reside in memory allocated to a running simulation, it will be streamed over a socket directly to the local visualization client. The overall picture of interactions with remote data or running jobs is shown in Figure 16.2.





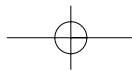
These technologies are seeing a growing use in numerical relativity projects. Several problems must be addressed, however, before such interaction mechanisms are fully embraced. First, firewall issues often conflict with the user's need to interact directly with remote data or simulations from any location. Second, such tools need to be enhanced to exploit Web and Grid services mechanisms (Chapter 17) so that, for example, applications can announce themselves, not just to a portal, but to any other compliant information servers or applications, exchanging data, contacting resource brokers to find new resources, starting other applications, notifying users or other applications when certain events take place, and so forth.

16.5 DISTRIBUTED COMPUTING

We describe various methods for distributing applications across resources: task farming, metacomputing, migration, and spawning.

Task farming involves farming out (a great many) independent or loosely coupled tasks, to resources scattered across a VO (see Chapters 13, 19, and 24). Typically tasks require little or no communication between tasks, and often return little data. Task farming can be used in our community for parameter studies, both to tune theoretical and computational parameters and to vary physical parameters over a large range of possibilities to search for the most interesting regime to be studied in depth. For example, the investigation of critical phenomena for a pure gravitational wave collapsing to form a black hole requires knowledge of the precise critical value for the amplitude of the initial wave. Slightly above this amplitude, the wave will collapse to a black hole; slightly below, it will disperse. Dozens or hundreds of jobs may be needed to find the critical value to sufficient accuracy. Grid task farming makes use of underlying Grid technologies to discover appropriate resources in a VO, to handle the staging and starting a set of tasks, and to archive task results, all in as short a time as possible.

Metacomputing involves the distribution of one or more tightly coupled tasks across a small number of large machines (72, 110). Metacomputing can be used both to increase not only peak capability but also available capacity: if, say, 1024 processors are needed but not immediately available on any one machine, four machines might be able to provide 256 processors each. We have demonstrated the feasibility of metacomputing with real production applications, on production machines and networks, and with no special conditions being created to enhance performance. For example, in 2001, we ran a simulation of colliding black holes across multiple remote machines, running on different operating systems, using adaptive techniques that automatically adjust messages sent across the network



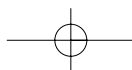
between machines to increase the efficiency, from 15 to over 70% as the simulation ran (72). A *Grid-enabled* version of the message passing layer (403) (Chapter 24) allowed the Cactus-based application code to be run without modification. Such experiments show that such Grid-based metacomputing scenarios can be run with a high degree of efficiency even for tightly coupled simulations such as Einstein's equations requiring many communications. Although these technologies are quite advanced and robust, they are used primarily for development—but only because of infrastructure deployment and scheduling issues. As Grid technologies are deployed across production sites, these capabilities can become a regular mode of operation.

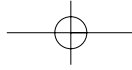
More complex distributed computing scenarios build on the technologies described so far. *Migration* involves moving a simulation from one site to another (239), for example because contention slows the simulation, or because the simulation needs more memory as adaptive meshes resolve a developing black hole. Migration proceeds with the help of a resource brokering service. If a new resource is found, a checkpoint file is written and transferred, a new executable is started, and the portal is notified of the new location. Robust prototypes of migration (69) use unmodified Cactus-based relativity applications.

A variation on migration is *spawning*, which involves moving just a *part* of an application to an appropriate remote resource. For example, when simulating black hole collisions, analysis tasks must be carried out to locate the black hole horizons or compute the gravitational waves emitted. These time-consuming tasks may not feed back to the main simulation or be easily parallelized and hence can be spawned to a more appropriate resource, allowing the primary resource to concentrate on advancing the main simulation. We demonstrated spawning scenarios in 2001, with a black hole simulation running in Germany spawning analysis tasks to resources in Europe, Asia, and North America.

16.6 DATA MANAGEMENT

The large 3D simulations that we have been considering here generate correspondingly large amounts of output data, which must be analyzed, visualized (often by several different members of a VO), and archived for later use. Each simulation can generate hundreds of files with different file formats, and the discovery and manipulation of these files are complicated by the fact that users are running on different machines, with different file systems, quotas, and archiving capabilities. For Grid applications, the data management problem is exacerbated—users may not even know on which machine their simulation is





16.7 Summary

211

running, or the simulation may be moving between resources and leaving data in multiple locations.

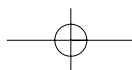
Even when the location of a file is known, its size may make moving the file to a local machine for analysis inconvenient or impossible. This was the case for the huge black hole collision simulations described previously (see Figure 16.1). Although Grid technologies were important for the international group of astrophysicists to remotely and collaboratively run, monitor, and adjust those simulations while they were running, the huge output generated could not be transported to local sites in other countries for analysis; instead, a team of scientists flew from Berlin to the United States to visualize and retrieve the data!

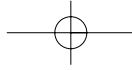
This problem is rapidly being solved. Tools for manipulating remote data are already being used by astrophysicists. One example is GridFTP servers (64) (Chapter 22), containing extensions from the German GriKSL project. When run on the file systems of machines holding data, these servers allow remote HDF5 data files to be analyzed with local visualization systems. Downsampling or zooming can be used to match the amount of transported data to the available bandwidth. Any visualization systems incorporating a GridFTP client along with the usual HDF5 reader can display these remote data, and such readers already exist for OpenDX and Amira software. These tools address the remote data problem and can be used to create, from thousands of miles away, the visualization such as the one shown in Figure 16.1.

16.7 SUMMARY

We have outlined how Grid technologies are being used in computational astrophysics today and how we expect these technologies to be used in the future. Although we have focused on the needs and applications of the numerical relativity community, the tools and experiences that we describe are quite general and apply to many other computational science disciplines. All of these scenarios are motivated by the needs of present-day astrophysicists, are working now in prototype form, and have been tested on real codes and in real production environments using today's Grid technologies.

The best way to bring these scenarios into everyday production use is to develop Grid applications in close collaboration with Grid infrastructure developers. Such a partnership ensures that application requirements found by prototyping and testing scenarios are addressed and solved and that the applications are ready to exploit the Grid as soon as possible. We envision that the scenarios described here will be running in production within a few years at most.





Still more advanced scenarios will become reality when applications take full advantage of a more mature Grid. Dynamic applications will themselves become Grid services, interacting with other services, moving about, expanding and contracting, adapting to their needs and to local resource and Grid conditions. Huge amounts of data will be collected from experiments and sensors around the world. In gravitational wave astronomy, Grid applications may process these data as they are collected, using Grid-enabled data analysis applications, then simulate the astrophysical sources of the data using Cactus-based applications, which in turn steer astronomical observatories generating the data, tuning their frequency characteristics to be more sensitive to a signal expected in a few hours. Likewise, other Grid-enabled applications will warn forecasters of impending earthquakes based on recent data acquired from satellites and ground-based geodetic data collected just hours before (581).

ACKNOWLEDGMENTS

Many colleagues contributed to the work and ideas presented here, in particular Werner Benger, Thomas Dramlitsch, Tom Goodale, Gerd Lanfermann, Andre Merzky, Thomas Radke, Michael Russell, and John Shalf, as well as our friends in the ASC, GriKSL, and GridLab projects. We are pleased to acknowledge support from NSF PHY-9979985 (ASC), DFN-Verein TK 6-2-AN 200 (GriKSL), and EU IST-2001-32133 (GridLab). Black hole simulation images resulted from computations at NCSA and NERSC.

FURTHER READING

For more information on the topics covered in this chapter, see <http://www.mkp.com/grid2>.