Computational Challenges For Large-Scale Astrophysics Calculations

B. Fryxell

ASCI Center for Astrophysical Thermonuclear Flashes, University of Chicago, Chicago, IL

Abstract. Numerical astrophysics is filled with a wide range of computational challenges. Many of these relate to developing a code for solving a given problem, or preferably a code that can address a large class of problems. Once the code is built, it is necessary to test it to make sure that the results are believable. Direct validation of astrophysics codes is virtually impossible, so other techniques must be employed. Finally, there are many challenges associated with performing the simulations, in particular managing the huge quantities of data produced and performing analysis and visualization of the results.

INTRODUCTION

Working in numerical astrophysics usually involves trying to perform the largest simulations possible with current technology. This means pushing every aspect of computation to the limits. It is only natural that many obstacles will have to be overcome along the way. Many of the challenges faced by computational astrophysicists are common to all fields of large-scale computing. However, the extreme conditions found in astronomical objects offers many challenges unique to this field.

The following section will discuss issues related to developing a general purpose high-performance astrophysics code. In many cases, the physics and numerics required for dealing with astrophysical objects is in direct conflict with achieving good performance on parallel computers, so compromises must be made. The next section will be devoted to techniques that can be used to verify and validate astrophysics codes, including comparison with standard test problems and with laboratory experiments. This will be followed by an analysis of a hypothetical astrophysics calculation, which illustrates the challenges presented by the immense amounts of data generated by a typical astrophysics simulation.

CODE DEVELOPMENT ISSUES

Frequently, astrophysics calculations are characterized by enormous ranges in length and time scales. It is not unusual for important length scales to vary by ten or more orders of magnitude in a single object. For example, in a Type Ia supernova, the initial radius of the star is 10^9 cm, while the thickness of the nuclear burning front near the center of the star can be less than 1 mm. Resolving both length scales using a uniform grid would require 10^{10} grid points in each spatial dimension. It is unlikely that we will ever have computers capable of dealing with this situation. Another common example is caused by the extremely small viscosity in most astrophysical fluids, which permits the formation of fully developed turbulence. These flows are characterized by a cascade of energy into small length scales that cannot be resolved on a typical grid.

There are a number of techniques available for dealing with multiple length and time scales. One is Adaptive Mesh Refinement (AMR), which concentrates grid points in regions of the flow where they are most needed. This method can significantly reduce time to solution and decrease memory usage, allowing higher-resolution simulations to be performed. Unfortunately, the increase in dynamic range of length scales provided by AMR is highly problem dependent and does not approach the needs discussed above. To make matters worse, the benefit comes at the cost of code complexity and poorer performance on parallel computers. Data must continually be redistributed between processors as the refinement pattern changes, increasing the ratio of communication to computation. In addition, these communication patterns are highly irregular and difficult to predict, making efficient implementation on parallel computers difficult.

Another effective way to improve the dynamic range of length scales in a simulation is front tracking. This technique can prevent unphysical mixing due to numerical diffusion at sharp interfaces. In the supernova example described above, the burning front could be replaced by an interface, so that the code would not need to resolve the width of the front. In this case, a physical model of how the flow changes across the front would also be required. Front tracking also adds significantly to the complexity of the code and can be difficult to implement efficiently on parallel computers. The grid points that contain the front are much more expensive to update than those in smooth portions of the flow, and this can lead to load balancing problems. As the front moves through the grid, one must continually redistribute data so that each processor has the same amount of work. This is similar to the problems encountered with AMR.

A third way to simulate flows with multiple length scales is the use of sub-grid models. This has been used primarily for turbulent flows. The approach is to replace all the physics that takes places on a length scale smaller than the grid spacing by a physical model. The main problem with this method is that the optimal model is problem dependent, and reliable sub-grid models are available only for relatively simple flows. For example, no good sub-grid models exist for magnetized flows.

In addition to challenges associated with multiple length and time scales, many astrophysical calculations also have to deal with a wide range of flow velocities. Efficient solution of low Mach number flows requires very different techniques from supersonic flows. It is not uncommon for both regimes to appear in a single simulation. Low Mach number, incompressible flows are usually calculated with some variety of spectral technique, while the supersonic regime is most often calculated using finite difference or finite volume shock capturing methods. One possibility would be to follow the low Mach number portion of the calculation with an incompressible spectral code and then switch to a compressible code after the Mach number becomes too large. However, it is not clear at this time that there is an intermediate region in Mach number space where the two codes give identical answers. In addition, the data structures used by the two types of code are very different. Because of these issues, mapping smoothly from one code to the other without creating a sudden jump in the solution may not be possible.

An alternate approach would be to use a purely implicit or an implicit/explicit hybrid algorithm for following the fluid flows. These techniques would allow treatment of flows of any Mach number within a single code. However, they are extremely difficult to program and maintain. They also require global communications and are therefore less efficient on parallel computers. For a hybrid method, the need for load balancing would provide an additional complication.

A third set of issues centers around the wide variety of physical processes that take place in astrophysical objects. The physics can be highly localized and non-uniform. In the supernova example described earlier, nuclear burning occurs only in the small number of grid points located near the flame front. It would be a waste of computer resources to call the nuclear reaction module at other grid points. However, without the use of load balancing techniques, processors containing only non-burning regions would have to wait while other processors were performing the burning calculations,, destroying parallel efficiency. Unfortunately, load balancing will require frequent redistribution of data, thereby increasing communication costs. In addition, many of the physical processes are non-local and require global communication. Examples are radiative transfer and self-gravity, which requires the solution of Poisson's equation.

An additional challenge in constructing a code involves the rapidly changing computing environment that we must deal with. New computer architectures are constantly being developed. The time scale for developing a complex code can be comparable to or longer than the life time of a given computer architecture. Development of portable codes rather than ones designed for a particular type of computer is essential. The emergence of Beowulf clusters requires incorporation of latency tolerant algorithms. Clusters of Shared Memory Processors (SMP's) may require the use of hybrid programming algorithms, such as using threaded code within a single box and MPI communication between boxes. This is a severe challenge to portability. Even without computer architecture changes, hardware and software is frequently unstable and subject to frequent changes. It is not unusual for "improvements" in operating systems or compilers to reduce the performance of a code or cause it to stop working entirely.

CODE VERIFICATION AND VALIDATION

In order to do believable science with complex simulation codes, it is first necessary to test them as thoroughly as possible with an intense verification and validation effort. There are many ways to test a code, but none of them can definitively determine if a code is giving the correct answer for a real astrophysics problem. Nevertheless, by careful use of a combination of these methods, it is possible to obtain some degree of confidence in the results of the code.

The most obvious test of an astrophysics code would be to validate it by comparing to observations of the event being simulated. Unfortunately, direct validation of astrophysics codes is extremely difficult. The stars we wish to simulate are so far away that they appear as tiny points in the sky, even using the most powerful telescopes. Even if the surfaces of these objects could be resolved, we could not see into their interiors, where much of the interesting physics is taking place, since the stellar gases are completely opaque. There are a number of events, such as supernovae, where one could try to reproduce observations of light curves and spectra. However, this gives only indirect clues to the physical processes that cause the event. Even if a model is found that reproduces the observations, it is unclear if the result is unique. Perhaps a completely different model would give similar agreement with observations.

The first step in checking a code is to run a standard set of verification tests. These are important in finding obvious flaws in a code, but there usefulness is fairly limited. Many problems with analytic solutions exist, but generally these test only the simplest aspects of the code. Few good test problems exist for multi-dimensional flows, and multi-physics tests are virtually non-existent. Convergence tests, in which the resolution of the calculation is increased to see if the answers change, can also be useful, but it is possible for a code to converge to the wrong solution or to converge to multiple solutions as the resolution is increased. To make matters worse, due to the complexity of astrophysical objects, it is almost never possible to reach convergence for a complete simulation. Another way to find errors in a code is to compare the results with those obtained from other codes. The problem with this approach is that, if the codes disagree, it can be difficult to determine which, if any, of them is producing the correct solution.

One additional approach to testing codes is validation through the use of laboratory experiments. Although it is impossible to obtain the physical conditions in a laboratory that one encounters in stellar astrophysics, except perhaps in high-energy laser experiments, much of the important physics, such as fluid flow, obeys simple scaling laws. In other words, the same physical phenomena can occur in both astrophysical objects and in laboratory experiments, but with different length and time scales. The chief difficulty with this approach results from the poor resolution of diagnostics in many laboratory experiments. It is usually possible to get agreement with the experiments for relatively large length scales, but little information can be obtained on the small scale phenomena, which are most likely to be in error.

DATA MANAGEMENT

Another major challenge that must be faced in performing large-scale astrophysics calculations is how to deal with the enormous quantity of data that is produced. In many types of large-scale computing, the result of a very long calculation can be just a few important numbers. This is not the case in most astrophysics simulations. A complete analysis of a simulation will require writing to disk the entire data set, not only at the final time, but also at a large number of intermediate stages. We are interested not only in the final result, but also how the system evolved into that state.

For a typical high-resolution three-dimensional astrophysics calculation, we would like to use a grid size of perhaps 1000^3 for a total of 10^9 grid points. If we store 25 variables at each grid point and limit the precision of the output data to 4 Byte words, each output file will require 0.1 TBytes of disk space. A 30 second movie of the results at 30 frames per second requires about 1000 output files – a total disk space of 100 TBytes for a single calculation. This will obviously put a severe stress on mass storage systems.

Even using parallel I/O, on currently available computers, the best throughput that could be hoped for is 100 MBytes per second. At this rate, it would take more than 10 days to write all of the data to disk. To retrieve all of the data from a remote computer at current network speeds would require more than a year. Data could also be retrieved by mailing tapes, but writing the tapes would also be time consuming and labor intensive. The tapes would also require significant storage space at the local site. Clearly, an alternate approach is needed, which will probably require analysis and visualization of the data at the remote site.

These large data sets also put a strain on visualization and data analysis capabilities. Full three-dimensional volume rendering of the data will require a large parallel computer. Even trying to visualize two-dimensional slices through a three-dimensional data set will be beyond the capability of a typical desktop workstation.

ACKNOWLEDGMENTS

This work was supported by the Department of Energy under Grant No. B341495 to the Center for Astrophysical Thermonuclear Flashes at the University of Chicago and by NASA/Goddard Space Flight Center.