Scaling and Visualization of N-Body Gravitational Dynamics with GalaxSeeHPC

David A. Joiner
Kean University
1000 Morris Ave
Union, NJ
djoiner@kean.edu

James Walters
Kean University
1000 Morris Ave
Union, NJ
walterj1@kean.edu

ABSTRACT

In this paper, we present GalaxSeeHPC, a new cluster-enabled gravitational N-Body program designed for educational use, along with two potential student experiences that illustrate what students might be able to investigate at larger N than available with earlier versions of GalaxSee. GalaxSeeHPC adds additional force calculation algorithms and input options to the previous cluster-enabled version. GalaxSeeHPC lessons have been developed focusing on two key studies, the structure of rotating galaxies and the large scale structure of the universe. At large N, visualizing the results becomes a significant challenge, and tools for visualization are presented. The canonical lesson in the original version of GalaxSee is the rotation and flattening of a cluster with angular momentum. Model discrepancies that are not obvious at the range of N available in previous versions become quite obvious at large N, and changes to the initial mass and velocity distribution can be seen more readily. For the large scale structure models, while basic clearing and clustering can be seen at around N=5,000, N=50,000 allows for a much clearer visualization of the filamentary structure at large scale, and N=500,000 allows for a more detailed geometry of the knots formed as the filaments combine to form superclusters. For the galactic dynamics simulations, we found that while a flattening due to overall angular momentum can be explored with N=1,000 or smaller, formation of spiral structure requires not only a larger number of objects, typically on the order of 10,000, but also modifications to the default initial mass and velocity distributions used in older versions of GalaxSee.

Keywords

N-Body simulations. Gravitational dynamics. Scaling, Visualization.

1. INTRODUCTION

1.1 Motivation

GalaxSee is a gravitational dynamics program initially developed by Mike South and the Shodor Education Foundation, Inc.[5]. The original version was designed for the Macintosh, and focused on allowing users to create small N-Body simulations using a point and click interface, to solve the problem of gravitational dynamics, where the force on any object i due to any other object j is given by:

$$F_{ij} = G\frac{M_i M_j}{\left|\vec{x}_j - \vec{x}_i\right|^3}\left(\vec{x}_j - \vec{x}_i\right)$$

A wide variety of approaches have been developed to solve the gravitational N-body problem[1], including many state of the art computational tools designed for research (see for example [15]), as well as many educational tools. Most research grade tools for N-Body simulation have obstacles to their adoption as a classroom tool—notably a reliance on non-standard compilers, multiple software dependencies, and non-human-readable file formats. Most educational N-Body tools, however, focus on the use of graphical user interface to remove obstacles for students, but replace those obstacles with limitations on the size of N, either hard-coded in the tool itself or self-imposed by the CPU requirements of real-time visualization of results.

Typical classroom use simulations for N-Body problems using tools with limits on the size of N range from 2-Body problems such as the orbit of the Earth around the Sun up to simulations of simple gravitational dynamics, exotic solutions of the few body problem [3], where users could create initial mass distributions with or without angular momentum and explore the disk formation that resulted from a spinning cluster of gravitationally bound masses, or collisions of disk galaxies under the assumption of small objects orbiting two massive cores [9].

1.2 GalaxSeeHPC Learning Goals

The two scenarios presented in this paper focus on studies of structure, the first of the formation and stability of spiral structure and the second of elements in large scale structure. Both of these are meant to be viewed qualitatively, as there are many physical elements left out of the model. In the case of the spiral structure scenario, the galaxy model presented does not account for drag due to the interstellar medium. The large scale structure scenario assumes Newtonian gravity in a constantly expanding universe. Even with these phenomena left out, however, key concepts in gravitational dynamics can be quickly and easily seen by students.

As the tool has been created for general purpose, specific learning goals would be largely implementation specific and would depend on the goals the instructor wanted to emphasize. An instructor focusing on performance or algorithms might have different goals than an instructor focusing on a science lesson. Like many of the tools developed at Shodor, GalaxSee has always followed the paradigm that it should be able to address both (computational science) education and computational (science education).

In the case of spiral structure, students might learn that the formation of spiral arms is a natural occurrence given a velocity profile that is gravitationally stable, and that not all velocity profiles will be gravitationally stable. Students can, in the process of exploring spiral structure get practice creating velocity curves for model galaxies which could then be compared to those of real galaxies, which might then prompt a discussion of dark matter or other issues of interest.

In the case of large scale structure, students can explore the interplay between expansion velocity and initial mass density for an expanding cube with periodic boundary conditions and "wrapped" gravity. While this leaves out some key features of the Lambda-CDM model, it will allow students to see a trend towards initial clumping along filaments, provided sufficiently high mass density and sufficiently low expansion velocity. The stability of those filaments over time can be seen to be strongly affected, with a tendency towards a "big crunch" for more dense and more slowly expanding systems.

Both of these cases lead naturally to goal-seeking exercises ("How can I change the velocity profile of this galaxy? What if this universe has more mass in a given expanding cube?") that focus on simple conceptual questions related to the balance of gravity, angular momentum, and expansion.

In terms of the computational science learning enabled by these lessons, students get practice using tools running at a command line, input file creation, management, and analysis, parallel job submission and monitoring. The data sets created are rich, with significant challenges in the visualization of results. The simulation includes a variety of force calculation methods, which, while not necessarily state-of-the-art, provide an entry level into two of the key methods used in modern N-Body, tree-based and particle-mesh methods.

1.3 GalaxSee revision history

The original GalaxSee, like many educational N-Body tools, took the approach of a graphical user interface with the ability to pre-create systems at random with a small number of parameters. Later versions of the code included GalaxSee 2.0 for Windows, which kept the look and feel of the original, but added the ability to use a Barnes-Hut force calculation, and a Java based web-start version. GalaxSee-MPI was written to explore parallel computing, removing the GUI interface, as well as the Barnes-Hut force calculation, and allowing for MPI based parallelization of a direct force calculation[8]. GalaxSee-MPI was originally intended just as an exploration of parallelism, and lacked any features to control the input to the simulation, nor did it have any advanced features for visualization, limiting itself to a non-interactive top-down-side-view image of the simulation.

1.4 GalaxSeeHPC Software Goals

The purpose in writing GalaxSeeHPC was to provide students with an N-Body code that (a) allowed students to explore the types of problems that cannot be solved at smaller values of N, (b) allowed students to see examples of some of the force calculation algorithms that have allowed for the increased use of N-Body algorithms, (c) was written in code that is designed for readability and modification, (d) had a simplified dependency stack so that some functionality would be available even without any additional code and that other features could be enabled easily as software dependencies were met, and (e) allowed for human-readable input and output files—so that students would not have to simultaneously learn modern hierarchical file structures at the same time as learning either the physics or algorithms of the N-Body problem.

GalaxSeeHPC is a re-write of the GalaxSee-MPI C++ code Lessons are available from the Blue Waters Petascale Education website[11] and source code is available from Sourceforge[6]. GalaxSeeHPC was written in C to allow for greater portability, and includes both the ability to perform a Barnes-Hut style force calculation algorithm as well as a Particle-Particle Particle-Mesh (PPPM) algorithm. While still command line based, GalaxSeeHPC allows for the user to use a text input file to specify model parameters, including changing the scaling and units used for the problem, allowing a linear expansion of the spatial units (e.g. for a simulation in an expanding universe), force calculation method and parameters, softening factors, numerical integration options, and output features. X-Window based output is still available, but a more interactive SDL-based visualization is also an option, as are multiple different graphics and text output options. CMake is used for configuration and build management, and the code can be configured at compile-time to ignore any options that require numerical or graphical libraries not present on the system.

GalaxSeeHPC has been used and tested in multiple sessions for Physics faculty at the SC09 and SC10 education programs. The visualization of results from GalaxSeeHPC has been a feature of multiple SC and NCSI workshops on scientific visualization. GalaxSeeHPC has been used in two successive summer camp environments with high-school age students.

2. GALAXSEEHPC ALGORITHMS

As every object can interact with every other object, this potentially leads to $N(N-1)$ forces that need to be calculated, though in practice half of these forces will be redundant as each force pair is equal and opposite. As an $O(N^2)$ problem, as N grows large the computational time requirements of the problem can quickly grow beyond the limitations of a typical classroom PC or laptop.

The three approaches that are used to alleviate this problem are parallelism to spread the work over multiple processes, binary tree based sorting of masses to determine which forces can be approximated by substituting a point mass in place of a large number of distance masses, and spectral techniques that interpolate onto a density grid which can be solved using Fourier techniques.

2.1 Barnes-Hut

The Barnes-Hut algorithm is a tree based approach to approximating the force field due to distant particles[2]. An oct-tree is constructed for the space modeled, with the tree recursively refined until each sub-element contains only one object. As the force is calculated, nearby objects, which typically will be close by on the oct-tree and can be located quickly, are used in a direct force calculation, and as objects are further away, branches of the tree can be approximated as a point mass, averaging the masses and positions of many masses into a single force calculation.

Tree methods work on the principle that one can organize an n-body model in a data structure that ensures that nearest neighbors can be easily defined for any one body, and that distant neighbors can be easily approximated using a center of mass treatment. In one dimension, this can be thought of as a binary tree, which can be extended to three dimensions using an oct-tree structure.

A simple implementation of a tree-based structure might assume that physical proximity is equivalent to being leaves on the same branch, but problems can occur for particles at the edge of a high level branch boundary, that are physically close to each other, but separated by many branching on the oct-tree. A modification of the tree algorithm to take into account issues like this might check to see if a node being tested is close enough to the object of interest to be suspect. As one descends the tree, this "closeness radius" can get smaller and smaller. If we consider s_l to be the scale of a tree segment at depth l, one might attempt the following force calculation method

```
1. For a given object, start at the top of the tree
2. Descend tree
    a. If child node is not a predecessor (along the same branch) of the object being calculated AND the object in question is at a distance greater than $ks_l$ from the center of mass of the node, stop and use the total mass and center of mass of that node
    b. If child node is a predecessor (along the same branch) of the object being calculated OR the object in question is at a distance less than $ks_l$ from the center of mass of the node, but does not SOLELY contain the object being calculated, descend all children of node
```

The accuracy of the method can be controlled by the closeness criterion k. Figure 1 gives a visualization of this in 1-dimension using a binary tree structure. Note that in the case $k = 0$ this reduces to the previous algorithm, and in the case $k \to \infty$ this approaches a direct force calculation. The total number of forces to be calculated will scale as $N\log(N)$ in this situation instead of N^2 for large models, and the accuracy of the tree calculation (and associated trade-off in speed) can be adjusted by use of the closeness criteria.

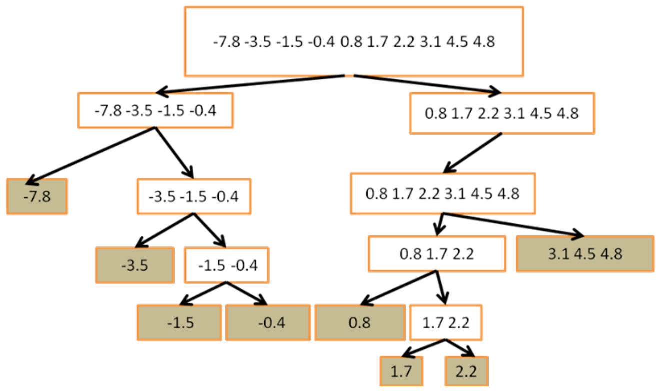

Figure 1: Use of a closeness checking factor can eliminate errors due to aggressive tree pruning

2.2 Particle-Particle Particle-Mesh

Spectral methods, typically solved using the FFT algorithm, reduce the discrete n-body problem to a continuous gravitational problem solved on periodic boundary conditions[4]. Computationally, the advantage of spectral techniques is that it allows you to separate the long-range forces from the short-range forces, and use a direct calculation of short range forces while replacing long range forces with the solution of a potential function that satisfies Poisson's equation.

$$\nabla^2\Phi = 4\pi G\rho$$

If a function for the density of space can be approximated, this can be solved easily as the Laplacian of the Fourier transform of a function is given by

$$\nabla^2\hat{\Phi} = -4\pi^2|\mathbf{k}|^2\hat{\Phi}$$

where $\hat{\Phi}$ is the Fourier transform of Φ. This gives for the solution of Poisson's equation

$$\hat{\Phi} = \frac{-G}{\pi|\mathbf{k}|^2}\hat{\rho}$$

which can be solved using a discrete Fourier transform, typically the Fast Fourier Transform (FFT) algorithm.

For the implementation in GalaxSeeHPC, the point mass distribution is first interpolated onto a density grid at evenly spaced intervals in x, y, and z. Each mass is treated as if it's mass is spread out over a Gaussian with standard deviation $\sigma = k_\sigma n / S$ where n is the number of grid points in each dimension (assumed to be equal in all dimensions in GalaxSeeHPC), S is the scale of a periodic box in the model, and k_σ is a user supplied parameter.

The Particle-Particle correction is applied to all points within some distance $\sigma_{near} = k_{near} n / S$, where k_{near} is a user supplied constant. Default values of $k_\sigma = 2.0$ and $k_{near} = 1.0$ are used in the code. For the purposes of the periodic boundary conditions in the PPPM algorithm, particles are "ghosted" across a periodic boundary if it results in a particle being closer to a second for the purposes of force calculation.

2.3 Parallelism

2.3.1 *Direct Force Calculation*

The wall-time when using a direct force calculation is dominated by the nested loop over all particles. This is parallelized in GalaxSeeHPC using MPI, and a round robin scheduling scheme to determine which particle's forces are calculated by which process.

2.3.2 *Tree-Based Force Calculation*

The tree creation takes sufficiently little time compared to the force calculation that we parallelize only the calculation of the forces from the built tree. The tree is typically built every timestep, but this can be reduced by the user. The loop over all particles to calculate forces from the tree is scheduled using MPI in a round-robin fashion.

2.3.3 *PPPM Method*

The creation of the density grid and the interpolation of forces from the density grid both consume a significant portion of the force calculation in the PPPM method. Each of these processes are parallelized in MPI using a round robin scheduled loop.

2.4 Softened Potentials

An issue occurs due to the $1/r$ potential in the gravitational N-Body problem in that there is a singularity in the force as particles get very close to each other. Typically, one uses some method of altering the potential to remove any singularities. This can be done by one of two methods in GalaxSeeHPC. The first is through use of a shield radius, as is done in previous versions of GalaxSee, in which the user specifies a parameter which defines a cutoff radius, within which forces are ignored. In practice GalaxSeeHPC uses an adaptive algorithm that depends on the central mass causing the force and the timestep being used, and the actual shield radius is given by

$$r_s = k_{sr}\sqrt[3]{GM\Delta t^2}$$

where the shield radius scaling factor k_{sr} is taken to be 5 by default.

Traditionally, most codes in the literature use what is referred to as a softened potential, in which the potential (and hence force) functions can be modified to include a softened distance, effectively treating all distances as if they were some small distance ε greater than they actually are.

$$P = -\frac{GM}{\left(R^2+\varepsilon^2\right)^{\frac{1}{2}}}$$

with accelerations

$$\frac{\vec{F}_i}{M_i} = \sum_{j\neq i} GM_j \frac{\vec{x}_j - \vec{x}_i}{\left|\vec{x}_j - \vec{x}_i\right|^2 + \varepsilon^2}$$

and potential energy

$$PE = -\sum_{i=1}^{N}\sum_{j=1\neq i}^{i-1} GM_iM_j \frac{\left|\vec{x}_j - \vec{x}_i\right|^2}{\left(\left|\vec{x}_j - \vec{x}_i\right|^2 + \varepsilon^2\right)^{3/2}}$$

3. SCENARIOS

One question that has arisen in many presentations of GalaxSee-MPI to faculty, particularly Physics faculty interested in the science that could be learned by such a simulation rather than computer science faculty interested in scaling properties, has been whether or not students working on projects involving N-body simulations need to run models with enough points to warrant high performance computing resources. A large class of astrophysical problems traditionally fit into what are often described as "million-body" problems—problems that require enough points for study that statistical or hydrodynamical approaches are not appropriate, but for which using too few points in an N-body solution will result in approximation error such that results are qualitatively incorrect[7]. Two problems are presented here that fit into this category, the modeling of galactic structure and the modeling of large scale structure in the universe.

3.1 Galactic Structure

3.1.1 Potential Learning Goals, Science

Students performing this exploration might, depending on implementation, focus on the velocity profiles required to maintain a gravitationally stable structure and the patterns that develop, as well as how the patterns that develop depend on the initial anisotropy of the mass distribution.

3.1.2 Potential Learning Goals, Skills

As N is increased, the computational overhead of a direct force calculation rapidly will increase the computational requirements of each run. The use of a tree-based method would be appropriate in this case as a periodic solution is not needed and the problem domain will have large regions of physical space in which there are few stars. Students can explore performance of tree-based methods as compared to direct force calculations. The parallelization method currently implemented does not truly split bodies across processors but merely shares the results of force calculation at each step. Students can explore the effect of communication on scaling as the code moves from a computation bound problem to a communication bound problem when increasing the number of processes.

3.1.3 Overview

Galaxies are large collections of stars, gas, and dust surrounded by relatively empty space, typically on the order of many kiloparsecs in size and containing hundreds of billions of stars. A key feature of galactic structure is the shape as classified on a tuning-fork diagram, categorizing galaxies as elliptical, spiral, or barred spiral[12]. (Teachers and students can find public domain images of many of these objects online, organized by galaxy type[13].) A feature of the original GalaxSee code was the exploration of how the interplay between gravity and angular momentum tended to flatten a large rotating mass of gravitationally bound objects. However, running models larger than a few thousand points was impractical, both due to hard coded features in early version of the code and the lack of an ability to operate in a command line mode with saved snapshots for models that required longer to run. Additionally, while it was possible to create models with different mass distributions and rotation curves, the default initial mass distributions and rotation curves in GalaxSee did not produce results that could be easily compared to images of spiral galaxies.

As GalaxSeeHPC makes for a more practical approach to running models with larger N, simulations were run to test the results at N=5,000, 50,000, and 500,000. Additionally, models were run with the default initial distribution and velocity profile in GalaxSee, with a mass distribution that is more heavily weighted to the center of the initial distribution, and with a velocity profile that is lowered for object near the center of the mass distribution.

3.1.4 Initial Conditions

The original windows GalaxSee used as its initial conditions a random uniform distribution within a sphere, and a velocity distribution associated with a circular orbit with centripetal acceleration equal to the central force being provided by gravity.

As the number of particles is increased, certain issues related to the default GalaxSee initial conditions are seen. In particular, a uniform distribution does not have enough mass in the core to keep the entire structure cohesively bound, and the distribution breaks up into many small clusters in orbit around each other. Additionally, the assumption of velocity set to centripetal acceleration works well at the edges of the galaxy, but towards the center this overestimates the actual orbital speeds, and simulations see a clearing effect wherein a ring structure is formed as opposed to something that looks like an elliptical, spiral, or lenticular galaxy.

As a result, our initial conditions are taken to be normal distributions in x, y, and z for position, parameterized by the standard deviations of the normal distributions σ_x, σ_y, and σ_z.

Velocities are calculated by modifying the assumption of centripetal acceleration caused by gravitational force to allow for a slower velocity towards the center.

$$\vec{a}_i = \sum_j -GM_j \frac{\vec{r}_{ij}}{\left|r_{ij}\right|^3}$$

$$\vec{a}_{Ti} = \left[a_{xi}, a_{yi}, 0\right]$$

$$\vec{v}_{Ti} = \frac{1}{2}\left(1 + erf\left(\frac{\rho_i}{\mathrm{P}} - 1\right)\right)\sqrt{a_{Ti}\rho_i}$$

where $\rho_i = \left[x_i, y_i, 0\right]$ assuming the entire mass distribution is centered at the origin, and P is the point at which the slower velocities towards the core switch over to a more typical centripetal acceleration-based velocity towards the edges. For each of the models here, we have assumed $\mathrm{P} = \sigma_x / 5$.

3.1.5 Results of Galactic Structure Simulations

A simulation was run with an initial distribution with $\sigma_x = 383pc$, $\sigma_y = 0.8\sigma_x$, and $\sigma_z = 0.1\sigma_x$, at sizes of 1,000, 5,000, 50,000, and 500,000 points (see Figure 2**Error! Reference source not found.**). At 1,000 points, typical of the problem sizes one would use with the Windows version of GalaxSee, the possibility of a spiral structure is hinted at by the results, but cannot be clearly seen with so few points. Increasing the size to 5,000 points makes the spiral structure more visible, and 50,000 points allows for a clear structure of spiral arms with clusters along the arms. Models were run for 1 billion years at a timestep of 500,000 years, using an Adams-Bashforth-Moulton integration scheme and a Barnes-Hut force calculation scheme.

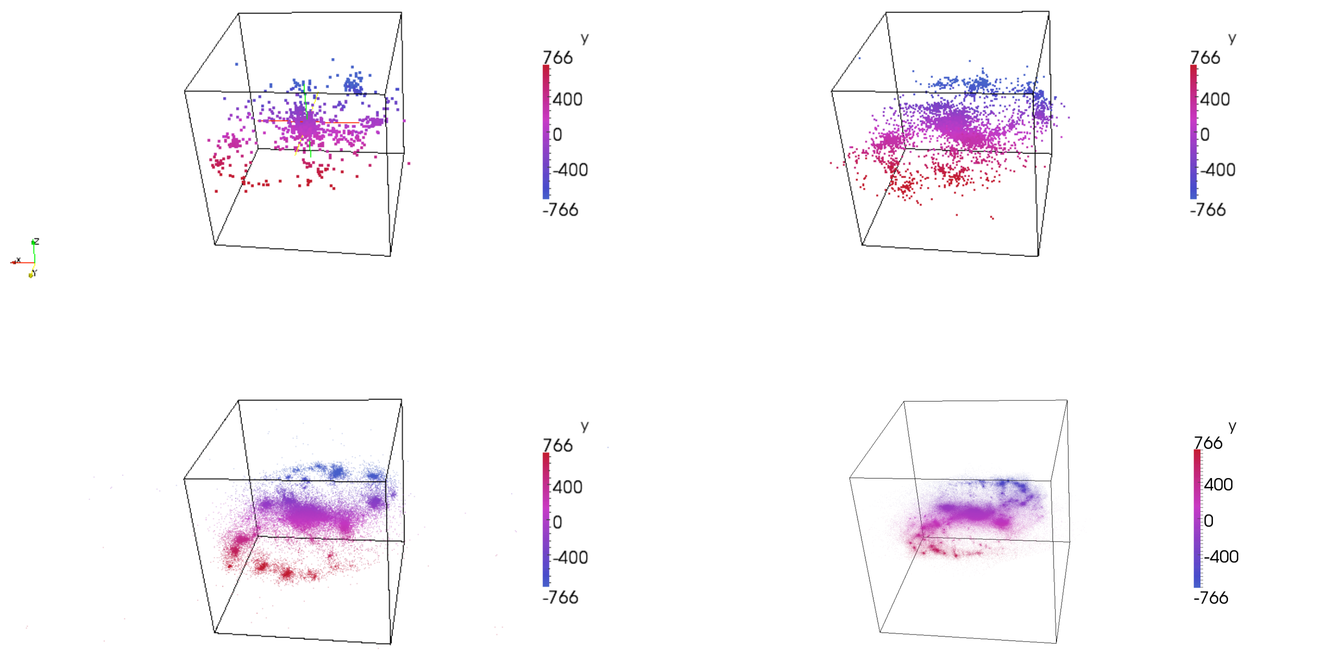

Figure 2 Spiral Galaxy Model with varying values of N. From top left to bottom right N= 1000, 5000, 50000, and 500000.

As can be seen in the comparison of the N=1,000 point and N=5,000 point simulation, 5,000 points was the bare minimum to begin seeing clearly any spiral structure that formed in these models, and on the order of 10,000 points is preferred. A 5,000 point model for GalaxSeeHPC with 8 processes ran in 3 minutes 54 seconds, and a 50,000 point model with 16 processes ran in 43 minutes. For practical use in a classroom lab, 5,000 point models are best run on a multi-core workstation or small cluster, and 50,000 point models are best done as either a single model run at the beginning of a class and analyzed afterwards or overnight or as part of longer term student projects. Models with 500,000 points showed more detail, but did not have qualitatively different features for this problem than those with N=50,000, while requiring significantly longer to run.

3.1.6 Scenarios for students to investigate

One key issue in the formation of classic spiral and barred spiral structures is the need for some difference in the scale in the x and y directions for the initial conditions. The lower the eccentricity of the initial material, the less likely it is that the resulting galaxy will have a classic two-armed spiral structure.

A second issue for students to study is the distribution of mass in the forming galaxy—looking at the difference between normally distributed matter and uniformly distributed matter, without an elevated density towards the center of the galaxy there will not be enough gravity to hold the center together, and students will see systems that fragment into many smaller rotating clusters. Additionally, it is possible to overestimate the acceleration of objects towards the center if one simply sets centripetal force equal to the gravitational force exerted on each object. Both of these are shown in Figure 3.

Testing

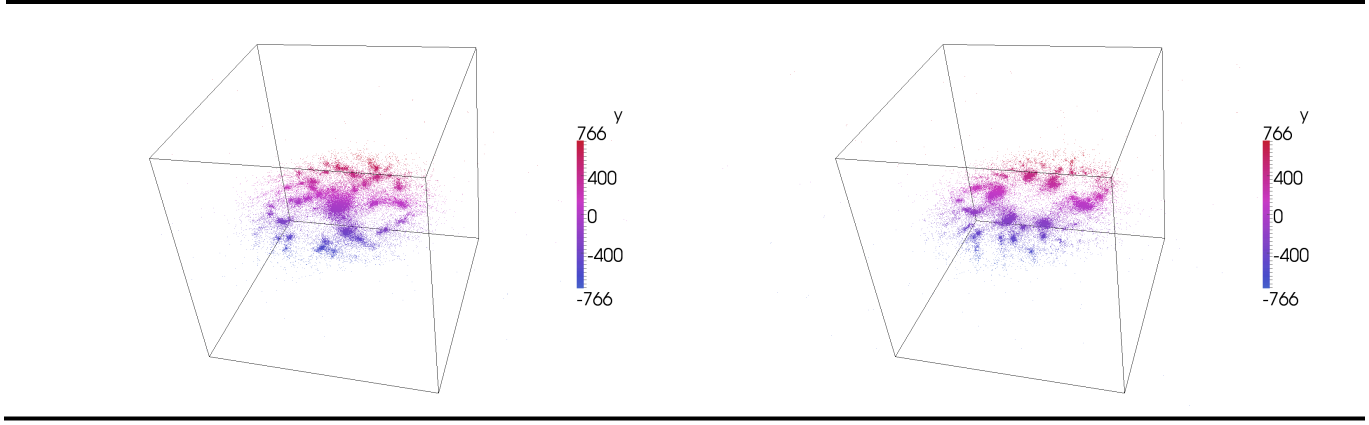

Figure 3 Simulation of galaxy formation without any eccentricity to induce spiral arm formation (left) and without a higher density in the central region to form a core (left)

This can be seen by example with a though experiment in which two equal stars orbit each other. Since it is not a case of a single object orbiting a more massive one, the actual velocities required to maintain a stable orbit are half what it would be otherwise.

This is addressed in the initial velocity function used in this paper by using an error function to create an interior zone where the objects are treated as if they are orbiting each other, and an exterior zone in which objects are orbiting a central mass. Having too little mass in the center can lead to fragmentation of the galaxy being modeled, and having too high of a speed for the interior objects can lead to clearing of the inner regions—and thus fragmentation of the galaxy being modeled.

3.2 Large Scale Structure of the Universe

Issues of cosmology on a large scale are both of interest to many students and are well reported in current media and research literature. Recent advances in computational simulations have led to understandings of the structure of the universe and the connection to the $\Lambda - CDM$ model of big-bang cosmology[4]. One of the largest N-body simulations ever run—the Millennium Simulation—focuses on this problem[14].

Modeling of large scale structure is complex—the distance scales change as the universe expands, and results depend sensitively on both the initial anisotropy of the mass distribution as well as the density. Computationally the problem requires treating the space modeled as a unit cell with periodic boundary conditions. However, students can explore at some level conceptual ideas with a simple Newtonian model. Our approach in GalaxSee is to let the students explore self-gravitation of a random anisotropic initial mass distribution in an expanding periodic box.

Initial student exploration into large scale structure can include an overview of the existing data on large scale structure, and attempts to fit models of big-bang expansion, gravitational condensation of galaxies, and freezing out of structures as the universe expands to that data, particularly with regards to the eventual end fate of our universe. While recent studies suggest that there is sufficient inflation to sustain the universe and have it continue its expansion, until recently it was unknown by scientists whether the universe's gravitational pull would ever result in an eventual "big crunch" collapse. This provides a compelling question for students to investigate, and allows them to understand the process by which computational science has informed us about this phenomenon. Even without allowing for either an expanding universe or any inflation to that expansion, students can, with only Newtonian gravity, explore the creation of filamentary structure and the eventual progression to a collapse event without expansion to prevent it. (As of version 1.1, GalaxSeeHPC supports the ability to model an expanding universe with a constant expansion rate, but does not allow for inflation—though this is a modification that a student could make.)

The models used in studying cosmological structure are often referred to as "universe-in-a-box" models, in that they take what might be considered a unit-cell of the universe, and approximate the gravitational effect of the surrounding universe by assuming that things are isotropic enough that whatever is happening on the left side of the cell is just as likely as anything else as to be a representation of what might be happening beyond the right edge of the unit cell. As such, periodic boundary conditions are applied, in effect giving us a toroidal geometry in order to approximate a piece of a larger universe. Students can change the initial mass density and size of this universe in a box, start with a random initial distribution, and simulate the initial clustering and eventual collapse that occurs. Students can see an interim stage before collapse where the types of structures formed closely resemble both the more accurate cosmological models being run on research codes.

3.2.1 Initial Conditions

The initial mass density and unit cell size were chosen so to ensure that the simulation would results in visible creation of filamentary structure, and the image shown are taken at the peak of the filamentary nature of the structure before further collapse occurred. The models shown here were run with no expansion and 1.0e14 solar masses randomly distributed in a 1 megaparsec cubed box. (Note that these numbers are chosen simply to produce qualitative results and are not meant to be physical. While these initial conditions can qualitatively show filamentary structure it results in a mass density of the universe that is orders of magnitude greater than observed and not stable for the lifetime of the universe.)

3.2.2 Results of Universe in a Box simulations

Simple effects can be seen with a fairly modest value of N. Consider the following simulation result, using GalaxSeeHPC with the PPPM algorithm and N=5,000. Figure 4 shows the results of a model with N=5,000 using the PPPM algorithm required roughly 1 second per timestep running in serial on a Xeon-based machine, with parallel performance peaking at only a few processes, though larger values of N were able to scale to more processes. With a typical model requiring on the order of a thousand timesteps, this is well within the range of what a student might do in a lab setting, running a simulation every 10-20 minutes on typical hardware

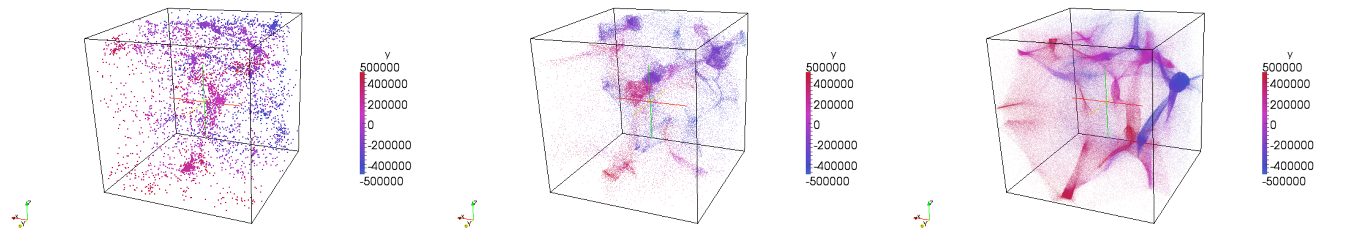

Figure 4 Large scale structure simulation, N=5000, 50000, 500000

Looking at the same model for greater values of N, students will be able to see more detail. At N=50,000, the knots in the middle of the filaments become more readily apparent and additional structure in the filaments can be seen The connectedness of the filaments is much clearer. The typical CPU time for models of this size in our tests was on the order of 2 days. Scaling up to 8 processes for this problem size on our test cluster was reasonably efficient; making this a simulation that students could run multiple times in one day on a quad-core or 8-core system.

When looking at the simulation results with N=500,000 the structure of the filaments themselves becomes much more clear, as does the morphology of the knots where filaments intersect. Scaling of this problem to 16 processes was reasonably efficient, and while models with millions of objects might run in days to weeks, depending on the number of timesteps required, students with access to a 8-core system or small cluster could run models in less than a day to a few days.

3.2.3 Scenarios for Students to Investigate

Two key questions students can try to address with these models are the sensitivity to the initial mass density of the universe of large scale structure and the effect of the expansion of the universe on large scale structure.

An initial study students may make is to look at the timescales needed for gravitational collapse of a large area of the universe with the current mass density and without any expansion. Starting with a random initial configuration, students should see that there is an initial clustering into a filamentary structure and that these filaments feed into superclusters which then themselves combine, but that the timescale for this happening is so short compared to the age of the universe that some degree of expansion is required to understand the structure of our current universe. The mass density in **Error! Reference source not found.-Error! Reference source not found.** shown in the previous section, for example, require a mass 4 orders of magnitude greater than observed, and would result in gravitational collapse within a few billion years.

GalaxSeeHPC has an EXPANSION variable in the input file which allows for a constant expansion rate. The timescales in which major change occurs will vary greatly as the universe expands, so for practical purposes it is useful to also have a scaled timestep that gets larger as the model progresses, and for that reason the student has an option of setting the timestep as a ratio of the current time using the TIMESTEP_RATIO variable rather than as a fixed number. Tracking the initial formation of anisotropy back to the point when gravitation began will likely require timesteps and numbers of objects that go beyond the architecture students have available, however the students can still start with a largely anisotropic random distribution of points at some later time, such as 1/100th the age of the universe, and evolve forward with mass densities near the current mass density of the universe. By changing the initial mass, they can see that the difference between structure never forming, filamentary structure of the type seen today, or a "big crunch" is only a few orders of magnitude, and that the qualitative types of structures found in more detailed models can be seen as naturally resulting from a combination of self-gravitation, mass density, and expansion.

Care must be taken in interpretation of results. While it is possible to independently set expansion rate and mass density in the GalaxSeeHPC input file, in practice it would be expected that these two parameters are related.

3.3 Scaling of the N-Body problem

An important concern with the N-Body problem is the scaling of the problem, both in terms of how the computing requirements scale with algorithm and problem size as well as how well the parallel implementation of the problem scales across a parallel architecture.

The first type of scaling is often referred to in terms of the "Big O" of the problem—if one were to write a function of the number of total computations needed as a function of the problem size, what term in that will dominate as the problem size gets large. In this sense, a direct force calculation is order N^2, and tree and PPPM methods are both order $N\log(N)$.

Parallel scaling, on the other hand, is typically referred to as either weak or strong. Parallel implementations with weak scaling allow for larger problems to be solved in roughly equal time on larger (i.e. more CPU cores) systems. Parallel implementations with strong scaling allow for same sized problems to be solved in less time on larger systems.

GalaxSeeHPC allows students to explore the big O scaling of direct, tree-based, and PPPM methods, and to begin exploring questions related to parallel scaling. It should be noted that the parallel implementation used in GalaxSeeHPC is limited in its parallel scaling, particularly for moderate and large clusters.

Students and teachers interested in pursuing questions related to state-of-the-art tools that exhibit strong scaling on larger systems are encouraged to look at the many professional-grade N-Body solvers. Of particular note is Gadget-2, which compiles with standard C compilers on many systems and has a fairly small number of dependencies required to run. GalaxSeeHPC includes an option to translate its own input files into Gadget-2 format.

3.3.1 Timing and Scaling of Galactic Structure Simulations

Running a simulation with 1,000 points and a Barnes-Hut calculation as described in the previous section, GalaxSee for Windows required roughly 17 minutes on an EEE PC with 1 1.7GHz Atom chip running Windows XP. Similar speeds with GalaxSee for Windows were seen on a HP EliteBook with a 2.5 GHz Centrino running Windows Vista. GalaxSeeHPC running on a single process on a Dell PowerEdge 1850 with a 2.7GHz Xeon running RedHat Linux finished in 1.7 minutes, on 4 processes finished in 31 seconds. For comparison on similar hardware, GalaxSee-MPI (which was largely based on the Windows GalaxSee codebase) using Barnes-Hut and a 4th order Runge Kutta took 4 minutes 18 seconds (GalaxSee-MPI does not currently support ABM integration methods). GalaxSeeHPC using Runge Kutta 4 took 3 minutes 24 seconds.

Many of these models could be run with a larger timestep, bringing running times on all platforms down (typical class presentations for the rotation and flattening of a spherical cluster are done with timesteps of 8 million years as opposed to 0.5 million years), however even for larger timesteps running models with on the order of 1,000 points is the practical classroom application limit of GalaxSee Windows.

Wall times per timestep for serial jobs are shown for N=5,000, 50,000, and 500,000 in **Error! Reference source not found.**. The tree-based implementation in GalaxSeeHPC scales closer to N log N than the N squared scaling expected of a direct force calculation. The parallel scaling of GalaxSeeHPC with the tree-based force calculation method was consistent across problem sizes, scaling to speedups of on the order of 10-15 on our cluster. Efficiency typically peaked once a few 8 core nodes were involved in the solution of the problem. For each of the problem sized tested, parallel efficiency dropped to 50% at about 16 processes. All are shown in Figure 5.

3.3.2 Timing and Scaling of Large Scale Structure Calculations

Like tree-based methods, the PPPM method in GalaxSeeHPC scales as roughly N log(N). The compute time required for the force calculation is dominated by the mapping of points to a grid and the interpolation of forces on that grid back onto the points, combined with nearest neighbor direct force calculations.

Speedup peaked at around 8 for models with N ranging from 5,000 to 500,000 on the cluster used in this study, with parallel efficiency dropping off somewhat faster for the PPPM methods compared to tree-based methods. Results are shown in Figure 5.

4. VISUALIZATION

4.1 Need for higher end hardware and software at large N

In addition to the computational challenges of increasing N in GalaxSee by many orders of magnitude, the resulting data also poses challenges in how it can be visualized, as traditional method of filling in a pixel if there is a mass in the line of sight for that pixel quick saturates at large N, even for very high resolution images. Even in relatively low-density regions of the simulation, foreground objects can obscure more important details. Masking the image by only showing a subset of points can result in loss of detail for structures of interest. This can impact both the type of hardware and software that is needed for students to work with large datasets. While modest computers with embedded video may be able to load and render larger datasets, such hardware can experience much longer frame rates when loading data for a new time or when attempting to re-render data for a different perspective (such as by rotating a rendered dataset in ParaView.) Figure 6 shows the effect of not allowing for any opacity when drawing a large number of point masses, as well as the loss of resolution and structure that can occur from masking points.

Many visualization packages exist that are available to students that allow for advanced features such as changing the opacity of points, volume rendering, and creating contours and slices of regular gridded data. ParaView[10] and VisIt[16] are two such examples that are available as open source, and will work with a variety of input data types included methods of opening simple comma separated files.

The images created for this paper were made using ParaView. ParaView is multi-platform, and has been designed to work in a distributed fashion for massive data sets. Developed by Kitware Inc. and Los Alamos National Laboratory, ParaView is also supported by Sandia National Laboratory and the Army Research Laboratory.

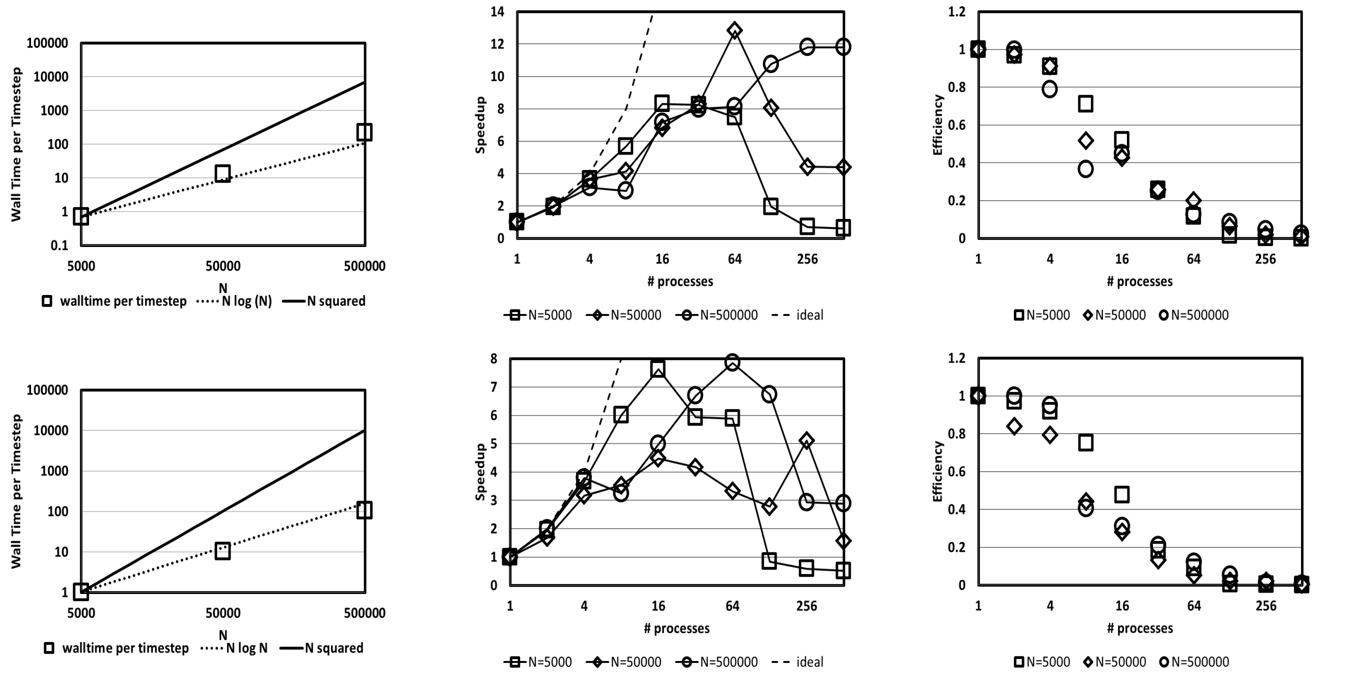

Figure 5 Scaling properties of example problems. Top row shows serial performance of tree-based algorithm run in serial relative to direct force calculation, followed left to right by speedup and efficiency (ideal would be 1.0) of tree algorithm in parallel. Bottom row shows serial performance of PPPM relative to problem size followed by speedup and performance.

4.2 Use of CAVE for visualization

Additionally, a CAVE system was used with students to visualize the results of GalaxSeeHPC, using a simple package written in OpenGL with CAVELib. Rendering was limited to no lighting effects and pixels for each mass, and up to N=500,000 could be viewed with zero masking and a frame rate high enough for the user to walk through the image without noticeable lag. The CAVE system used was a three wall system with ART head tracking and a dedicated render node using separate NVidia Quadro cards for each wall.

Our initial use of the CAVE has focused on the feasibility of using it for education. Technically, we wanted to know whether there were easy methods of getting student data into the CAVE and whether it would provide an obstacle that interrupted class flow.

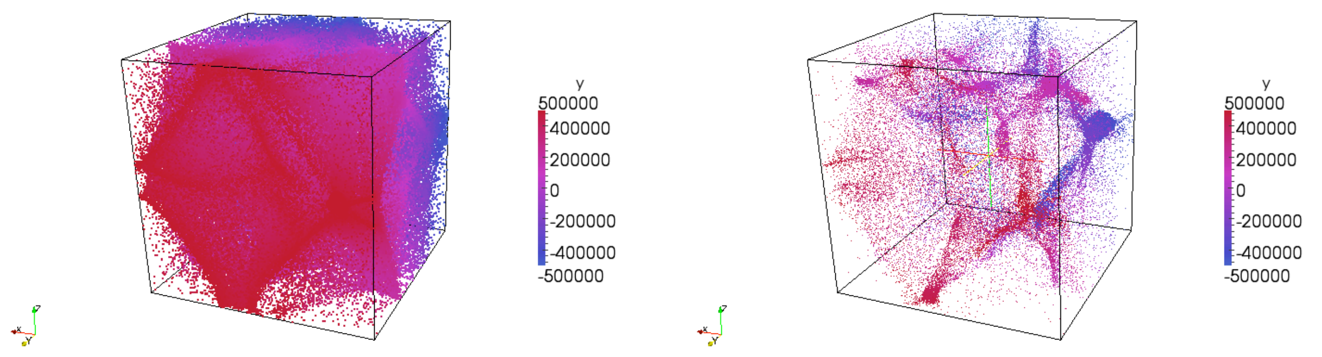

Figure 6 N=500,000. Shown at left is without any masking or opacity. At right masking is used, but no opacity is enabled to enhance visualization.

5. PEDAGOGICAL CONCERNS

5.1 Sample Lesson Plan

GalaxSeeHPC is meant to be a general purpose pedagogical tool around which a variety of lessons might be built, focusing on both topics in computational science education as well as topics in physics and astronomy.

The following lesson plan is designed based on past use of GalaxSeeHPC with high school students. It assumes the use of a helper code "GalaxSeeUI," available on the Sourceforge site for

GalaxSeeHPC, to generate input files for the investigation of spiral galaxy shapes.

Subject: Physics

Grade: 11-12

Lesson Length: 90 minutes (2 classes of 45 minutes)

Title: Galaxy Structure Simulations Using Computer Applications

Overview:

Galaxies are large collections of stars rotating around a central point in space, while moving about in the universe. These bodies of stars tend to crash and collide with each other, and take on new and varying forms. Through Hubble, galaxies have gained classifications based on their structures as they form over time. This lesson will have the students learn about how to classify galaxies by their structure, using Hubble's model and computer simulations of their own design.

Preparations and Materials:

- The teacher should become familiar with the GalaxSeeUI application, GalaxSeeHPC application, and the ParaView application. (GalaxSeeUI application is available on Sourceforge site along with GalaxSeeHPC and can be used to generate input files for this lesson.)
- The teacher will need an internet browser in order to access this site

  http://cosmictimes.gsfc.nasa.gov/teachers/guide/1929/guide/classifying_nebulae.html
- The teacher should have a text editor, such as notepad, loaded along with the applications, and some way of displaying all of the applications on the monitor to the class.
- The students will need access to computers to utilize text editing software, in order to generate their initial conditions.

Objective:

- Students will be able to distinguish the different galactic structures, using the tuning fork model and computer simulations.
- Students will be able to apply their previous computer knowledge to generate input files for GalaxSeeUI and utilize ParaView.
- Students will be able to compose an argument about their own observations and defend their point of view.
- Students will be able to infer things about natural phenomenon based off of the activities conducted during this lesson.

Standards:

- NSES.9-12.A: Science as inquiry.
  - Use technology and mathematics to improve investigations and communications.
  - Communicate and defend a scientific argument.
- NSES.9-12.D:
  - Origin and evolution of the universe.
- NSES.9-12.E:
  - Understanding about science and technology.
- DoDD.Science.9:
  - Use of computational models.
  - Use careful systematic observation and data collection to obtain valid information.
  - Relate force, motion, energy, and power.

Procedure and Activities:

Day 1 – 45 minutes

1. The teacher will define the term galaxy.
   a. Galaxy - a system of stars, numbering in the millions to billions that, along with gas and dust, are held together by gravitational attraction.
   b. An example that can be given is the Andromeda galaxy, the closest spiral galaxy to the Milky Way galaxy.
2. The teacher will define the types of galaxies:
   a. Elliptical - a galaxy, generally having an elliptical shape and no obvious inner structure or spiral arms
   b. Spiral - a galaxy, that exhibits a central nucleus from which many curved arms extend
   c. Bar Spiral – a galaxy, that contains a central bar structure from which two large arms extend
   d. Irregular – a galaxy, that cannot be labeled by the previous definitions
3. The teacher will utilize the Hubble Classification, or the Tuning Fork Diagram, to discuss the development of galactic structure over time. A version of this diagram can be found on:

   http://skyserver.sdss.org/dr1/en/proj/advanced/galaxies/tuningfork.asp
4. The teacher will show students a general input file of GalaxSeeUI in Notepad, and will explain to the students the proper way to input and save the data. All files that are being submitted to GalaxSeeUI are .in files and can be saved with this extension when saving and asked to name file (Example: test.in).
5. The students will be paired into small, 3-4 person, groups to work on their own input files.
6. The students will utilize the computers to create an input file, using Notepad, following the teacher's example on how to setup the text file and save it with the proper extension.
7. The teacher will tell the students to finish what they are doing, and to return to their groups. The teacher will, then, have the students choose one member of their group to submit their file to GalaxSeeUI, and save the file to a folder, the teacher should have access to this folder. Advise that this folder should be a shared folder that the entire class can access, but the teacher can control.
8. The teacher will show a variety of galaxy pictures to the students, and ask the students to make a classification of the galaxy's structure, as well as provide their reasoning for coming to that conclusion.

Example images can be found on: http://hubblesite.org/gallery/album/galaxy/

9. The teacher will ask if there are any final questions or comments, and conclude the lesson. This time can also be used to aid the students with any errors that may arise.

Day 2 – 45 minutes

10. The teacher will show the students how to start ParaView, and how to configure ParaView to read in their data files. The teacher will, then, show the students how to play their animation, and how to download the images needed to examine the structure of the galaxies.
11. The teacher will have the students retrieve their data from, a flash drive the teacher controls or, the folder used previously. The folder should have individual folders with the group of students' names, and inside the folders should be the input file the students created, and the output from GalaxSeeHPC.
12. The students will observe their galaxies, analysis the results they note, and make an educated conclusion on the structure of their galaxy.
13. The teacher will instruct the students to use ParaView to take a picture of their "initial" step and their "final" step.
14. The student groups will share their results with the class, having the students present a small summary of their results and making their final classification of their galaxy.
15. During the ending to the period, make references to stable and unstable initial conditions. Care must be taken to differentiate between a student's set of initial conditions and actual data. Possible wording would be to always refer to the students simulations as models and never as "a galaxy."
    a. Stable – initial conditions such that the model does not exhibit overall change in structure or makeup as the simulation evolves. A stable simulation that additionally exhibits behavior similar to data is one in which the initial conditions are likely to correspond with real galaxies.
    b. Unstable – simulation exhibits behavior that changes greatly during evolution, particularly changes in the size, rotational speed, and overall geometric makeup. This may be due to numerical instability (have students try reducing timestep), or it may be due to initial conditions that are not physically likely.
16. The teacher will ask if there are any final questions or comments, and conclude the lesson. This time can also be used to aid the students with any errors that may arise anywhere during the lesson.

Extensions:

17. Show the students how to plot the velocity of their galaxies in ParaView as star color. Show them how the velocity curve of their galaxies plays a role in how stable their structure is.

5.2 Choice of Time-step

Currently none of the versions of GalaxSee (GUI-based, the original command-line MPI, or the latest GalaxSeeHPC release) allow for an adaptive timestep in solution. While this change is planned in the future, this makes it especially important that an appropriate time-step is used. Even with professional grade codes using higher order and/or adaptive integration schemes, great care must be taken with choice of time-step.

If students are not familiar with time-stepping methods, they should get some information on the drawbacks of a time-step that is either too large or too short. Any of the versions used with some form of visualization (GUI-based have built in visualization, GalaxSee-MPI and GalaxSeeHPC have the option of compiling X-based visualization into the program if supported by your platform) will show this clearly, with demonstrably wrong results and instabilities occurring with too large a time-step, and with visibly slower computation occurring with too small a time-step.

5.3 Choice of Integration Method

Integration methods available in GalaxSeeHPC mirror some of the more standard options used, as well as some options that are pedagogically easy to introduce yet not stable enough for professional work. The Euler method is included for pedagogical purpose as it is often the first numerical integration method students learn, and the easiest to code. The so-called "improved Euler" or second order Runge-Kutta scheme as well as the mid-point Euler method and leapfrog methods are also allowed in the code as these are often introduced in numerical analysis classes as incremental improvements to Euler's method. In practice, however, one would not want to run professional integration with these schemes. The fourth order Runge-Kutta algorithm is generally considered the simplest numerical integration scheme one would want to use for professional work, and is a standard method used across computational science disciplines. Additionally, predictor-corrector schemes, such as the Adams-Bashforth-Moulton available in GalaxSeeHPC attempt to use previous timesteps to better predict future behavior. For anything other than investigating the numerical impact of using lower order integration schemes, students should use either the fourth order Runge-Kutta or Adams-Bashforth-Moulton integrators.

5.4 Limitations of GalaxSee-MPI

The primary limitations of GalaxSee-MPI from a classroom perspective was the inability to use it to teach any concept beyond which it was originally intended. GalaxSee-MPI as first written was designed to show scaling of the parallelization of direct force calculation using MPI, however all of the features of previous versions of GalaxSee that made it a useful tool for classroom exploration had been removed—the ability to easily modify input for new scenarios, the ability to design input files to meet your own problem, the ease of visualization had been removed in making a command line version of the program. Moving the program to a command line version in a HPC environment, however, did allow for much larger values of N—which the visualization abilities of early versions of GalaxSee would not handle well anyway.

Additionally, over many years of using GalaxSee-MPI in faculty workshops with Physics faculty, many faculty expressed skepticism as to whether there would be benefit for their students to running N-Body simulations with a larger value of N—whether there was anything the students would learn at large N that they would not learn at small N. Also, the lack of a feature to allow for periodic boundary conditions limited the types of situations that could be modeled.

From a technical perspective, the use of GalaxSee-MPI in new environments was often hampered by the choice of C++ as a language. While C++ is largely standard and widely adopted as a language, the C++ version of GalaxSee-MPI suffered from portability issues as it was deployed on different clustering platforms. The dependency on specific standard libraries often caused software to fail to run as expected, and different mpicxx executables, from one MPI implementation to another, often required minor code changes to in order to deploy the software on a new platform.

5.5 Changes Made

The following feature comparison shows changes made in GalaxSeeHPC compared to previous GUI based and command line based versions.

Feature	GUI versions	GalaxSee MPI	GalaxSee HPC
Runs from input file	✓		✓
Users can specify individual particle properties	✓		✓
Problem scale	Choose from menu list		User specified
Change integration method (Euler, Improved Euler, RK4, ABM)	✓		✓
Barnes-Hut	✓		✓
PPPM			✓
Passive visualization	✓	(with X11)	(with X11)
Interactive visualization	✓		(with SDL)
Command Line option		✓	✓
MPI		✓	✓
Write to snapshot files			✓
Additional output options			✓
Softened potential	Adaptive shield radius	Adaptive shield radius	Adaptive shield radius or fixed softened potential

5.6 Effect of Modifications

One concern in moving to GalaxSeeHPC was whether the removal of the GUI component would make exploration of science questions significantly more difficult for students using GalaxSee. In previous workshops with students, typical use was to use the Windows, Mac, or Java version of GalaxSee when exploring science questions and to use the GalaxSee-MPI version of the code when exploring problems with parallel efficiency and scaling. Our first use of GalaxSeeHPC in a informal education setting in summer 2010 did show that the constant flow back and forth between windowed versus command line environments slowed the pace of activities down, and when given the choice students tended to stick with the GUI driven tools. In summer 2011, we focused more specifically on using the command line tools, with more instruction on the use of the command line interface and activities that included visualization of results solved with larger N in a CAVE environment, which seemed to make for a more natural use of the command line driven HPC tools.

Since the move to C, we have seen significantly reduced issues with portability. The new version of the code has been tested on multiple platforms with both GNU and Intel compilers.

5.7 Visualization tools

Any effort to bring scalable supercomputing applications into the classroom will need problems of significant size to (a) require supercomputing resources, and (b) scale on those resources. This provides an additional concern for the educator in that large problems produce large sets of results, and visualization of those results will need to be part of the plan for implementing the use of such tools in the classroom. The use of common data formats is encouraged in order to be able to make the best possible use of open source visualization tools. Comma Separated Value text files provide a low barrier for creation of files, and are readable by many visualization tools, but will typically require the configuration of many options within the tool to define how the CSV file should be interpreted. Other Common Data Formats, such as NetCDF or HDF, are well supported by the open source community, and are standard input file formats for most visualization tools, however this will provide an additional challenge for implementation as code libraries for those formats may have to be installed on the systems on which students are computing their results.

5.8 Storage limitations

Another concern for problems involving large N, particularly in a classroom situation in which many students will be running multiple sets of such models, is disk storage. For our N=500,000 models, 30Mbytes per snapshot was typical, stored in NetCDF format. Keeping enough snapshots to create a smooth animation for N=500,000 typically required 3Gbytes per simulation. Storage requirements were linear with N.

6. FUTURE WORK

6.1 CAVE Visualization

Our initial work in incorporating the CAVE into the visualization of GalaxSeeHPC has focused primarily on technical issues of how to get the data into the CAVE as well as the feasibility of incorporating a CAVE system into the flow of a class. While our general finding is that stereo immersive visualization, as it is inherently focused on one individuals point of view, is difficult to use in a large class setting it can be inspirational for students. We noticed a clear “wow factor” when bringing participants into the CAVE. It is easier to incorporate immersive visualization into individual student projects, as there is less of an issue with contention for the resource.

Our initial work with students has used custom written software, and we are investigating whether we can replace this by using VisIt, for which a Conduit interface exists, or ParaView, which has been ported to other CAVE systems using FreeVR.

We have not yet investigated whether participants learn differently in an immersive environment from a non-immersive environment, or from viewing 3-D data in other, non-immersive, stereo visualization systems.

While CAVE systems are unlikely for typical classroom use, students may consider using non-immersive stereo rendering in ParaView through more readily available 3D monitors, TVs, or projectors.

7. ACKNOWLEDGEMENTS

The GalaxSeeHPC revisions were funded as a module project under the Blue Waters Petascale Education program. The cluster used for this project was funded by NSF award OCI-722790. The CAVE used in this project was funded by NSF award OCI-0959504.

8. REFERENCES

[1] Aarseth, S. 2003. *Gravitational N-Body Simulations*. Cambridge Monographs on Computational Physics.

[2] Barnes, J.E. and Hut, P. 1989. Error analysis of a tree code. *Astrophysical Journal Supplement*. 70, (Jun. 1989), 389–417.

[3] Christian, W. 2010. EJS CSM Textbook Chapter 5: Few-Body Problems. *An Introduction to Computer Simulation Methods - Draft EJS edition*.

[4] Efstathiou, G. and Eastwood, J.W. 1981. On the clustering of particles in an expanding universe. *Monthly Notices of the Royal Astronomical Society*. 194, (Feb. 1981), 503–525.

[5] GalaxSee Curriculum Resources: *http://www.shodor.org/master/galaxsee/*.

[6] GalaxSeeHPC: *http://sourceforge.net/projects/galaxseehpc/*.

[7] Heggie, D. and Hut, P. 2003. *The Gravitational Million-Body Problem*. Cambridge University Press.

[8] Joiner, D.A. et al. 2008. Supercomputer based laboratories and the evolution of the personal computer based laboratory. *American Journal of Physics*. 76, 4 (2008), 379.

[9] Mihos, C. et al. 1999. GalCrash: N-body Simulations on the Student Desktop. *American Astronomical Society Meeting Abstracts* (Dec. 1999), #101.04.

[10] ParaView - Open Source Scientific Visualization: *http://www.paraview.org/*. Accessed: 2012-04-25.

[11] Petascale: GalaxSeeHPC: 2011. *http://www.shodor.org/petascale/materials/UPModules/NBody/*.

[12] Rood, H.J. and Sastry, G.N. 1971. ”Tuning Fork” Classification of Rich Clusters of Galaxies. *Publications of the Astronomical Society of the Pacific*. 83, (Jun. 1971), 313.

[13] SEDS Messier Database: *http://messier.seds.org/*.

[14] Springel, V. et al. 2005. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*. 435, 7042 (Jun. 2005), 629–636.

[15] Springel, V. 2005. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*. 364, (2005), 1105–1134.

[16] VisIt Visualization Tool: *https://wci.llnl.gov/codes/visit/*. Accessed: 2012-04-25.