Advanced Parallel Computing Techniques with Applications to Computational Cosmology

1 Introduction

Advances in our ability to build parallel computers have continued unabated over the past decade. Large parallel machines have been deployed, including a 3,000 processor machine available to the academic community, as well as a 30,000 processor system at DOE. One of the planned machines (IBM's Blue Light) for the near future includes over 65 thousand processors. At the same time, the structural complexity of parallel applications in science and engineering has been increasing. Algorithmic advances such as multiple time stepping and adaptive refinements, coupled with the unprecedented compute power available, could lead to breakthrough advances in science and engineering. However, to realize this potential, substantial advances in computer science are needed. Attaining high performance on such large machines for complex applications is difficult, and the programming effort required is unreasonably large. It is clear that tomorrow's applications demand new techniques that automate or simplify specific parallel programming tasks, to help achieve high performance.

Our experience indicates that developing computer science abstractions that have a significant impact on the state of art in parallel computing is possible only when such research is conducted in the context of real applications. Computational cosmology, as explained in the proposal, is an ideal application area from this viewpoint. It presents most of the challenging issues in the upcoming era of parallel computing: irregular and dynamically changing application structure, complex communication needs that have the potential to overwhelm parallel performance on large machines, multiple time stepping that creates new challenges for dynamic load balancing, massive parallel I/O requirements, and the need for post processing of large and complex output data.

Computational cosmology itself is poised for new breakthroughs if the power of new parallel computers and new algorithms can be used effectively. For example,

- Understanding the formation of galaxies in their detailed cosmological context.
- Understanding of the complex interaction of cluster galaxies and the intra-cluster medium.
- Constraining dark matter theories based on both large scale structure and the substructure within galaxies and clusters.
- A predictive model of solar system formation, from proto-solar disk to fully formed planets.

Accurately performing simulations of these problems, and comparing the results with new ground (e.g. SDSS, [48], and BOOMERANG, [15]) and space-based observations (e.g. MAP, CHANDRA and HST satellites), will greatly enrich our understanding of our origins.

The proposed research is aimed at making significant advances in both computational cosmology and parallel computing. New parallel computing techniques (Sec. 3) developed as a part of this research, including automatic load balancing and communication optimizations, will be released in the form of a software framework for general-purpose parallel computing. These will be used to develop a computational framework (called *N-Chilada*) for simulation of particles in general (Sec. 4) Several computational cosmology algorithms will be implemented in the particle framework. Such a particle framework will be useful in applications beyond computational cosmology. Using the framework and its algorithms, specific scientific studies will be performed (Sec. 2) to constrain theories of cosmology by simulating large scale structure, understand the sub-structure of galaxy clusters, understand galaxy formation within a cosmological context, and understand the complex interactions between dust, gas and planetesimals during the process of planet formation. The synergy between the proposed research components is shown in Figure 1.

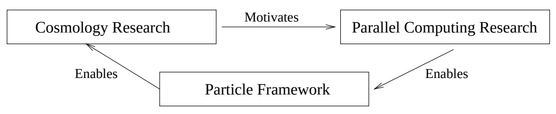


Figure 1: The synergetic Relationship between Research Components

2 Computational Astrophysics Research

The projects below demonstrate the scientific needs of computational cosmology that require breakthrough advances in parallel programing abstractions to fulfill. All of these projects also represent the continuation of our ongoing scientific programs with a number of long-time collaborators. The breadth and number of the collaborators ensures a large user community for the framework.

Large Scale Structure

Here we make the connection between the upcoming MAP satellite data and the Universe of today. MAP will measure the small cosmological fluctuations in the microwave background temperature, essentially giving us the "initial conditions" of the Universe. Simulations of large cosmological volumes are necessary to quantify how initial fluctuations of one part in 10^5 in the microwave background turn into the galaxies and clusters we observe. The spatial distribution of galaxies across a statistically significant region of the Universe gives clues about the early Universe, but this has to be disentangled from how these objects form according to environment. Making sense of the large scale surveys such as the Sloan Digital Sky Survey([48]) depends on understanding this process.

Our observational collegues have therefore thrown down the gauntlet: they have cut glass, poured concrete and launched satellites to measure structure that we *must* simulate. To unequivocally address these observations, we will simulate a volume 400 megaparsecs on a side with at least 10 billion particles and a spatial resolution of only a few kiloparsecs, well beyond the current state-of-the-art. The timescale range is similarly large: the collapsed structure has to be integrated on million year intervals while the overall Universe evolves on gigayear timescales. By firing off parallel analysis tasks periodically *during simulation* we will be able to couple the simulation to advanced semi–analytical techniques that will allow us to predict the galaxy population properties and its evolution in time.

This will require:

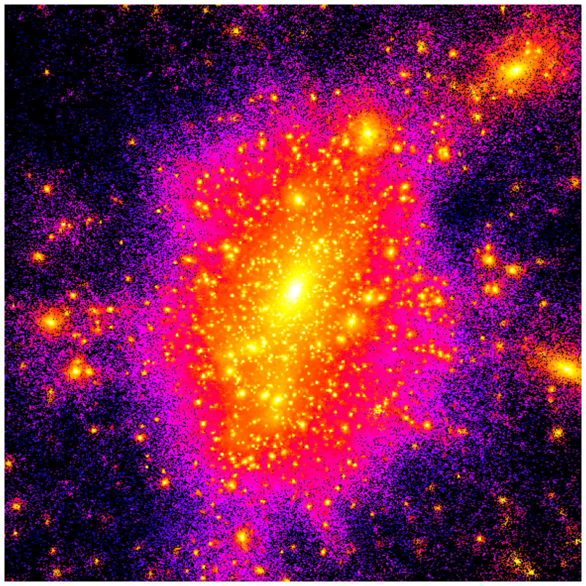
- A large amount of parallel computing time for the actual simulation.

  The challenge here will be to scale our simulation code to 10s of thousands of processors to make runs like this feasible in a reasonable amount of wall clock time.
- A large number of outputs and hence a large amount of data (many TB)
- Extensive parallel postprocessing of individual outputs, as well as on-line analysis of simulation snapshots

Interpreting the "galaxies" requires the identification of individual halos and then computing a number of different physical quantities such as density profiles and total angular momentum. Some of the postprocessing algorithms needed currently perform N_{halo}^2 calculations and are very computationally intensive once the volume contains several individual halos with $\mathrm{N}_{halo} > 10^6$. Here we have the multiple challenge of devising faster algorithms, implementing them in parallel and coupling to a parallel I/O framework in order to handle the large amount of data. Furthermore, different algorithms perform better in the early Universe when the structure is relatively smooth, and others perform better on the highly clustered state that describes the Universe today. The ability to "plug-and-play" different algorithms within a highly parallel context is necessary for optimum performance.

Galaxy Clusters

High resolution simulations of galaxy clusters are one of the hallmark achievements of the Washington team [14] [20] [29] [17] [30] [19] [31] [18] [28] [7]. Through extensive use of the "volume re-normalization" technique we were able to extend the dynamic range of our large cosmological simulations by several orders of magnitude. A simulation of a Coma like cluster was able to resolve individual dwarf galaxies (10kpc in radius) in a box of 1000 Mpc across: a dynamical range of almost 10^5! The technique requires that first a large scale simulation is done at low resolution



Simulation of a cluster of galaxies. The color encodes dark matter density. This is our groundbreaking simulation that first accurately resolved "halos within halos" and accurately modeled the central density profile, results that have fomented significant rethinking of the standard model of dark matter. The simulation was performed by PKDGRAV, and the visualization was done using TIPSY. However, it is a simulation of only the dark matter. Higher resolution and gas dynamics will allow us to make tighter connections with observations

and regions of interest (*e.g.* clusters of galaxies) identified. The initial conditions for the high resolution simulations of the regions of interest are then generated by extracting the low-frequency initial conditions associated with these regions from the large simulation and adding high-frequency density waves and high resolution particles. It is not sufficient to just extract the initial conditions for the objects of interest. The surrounding matter distribution, because of its tidal influences on the structure under study, must also be taken into account. Note how this requires highly adaptive methods to be feasible.

These simulations resolve many thousands of subhalos within clusters, and resolution tests show we may have converged on their inner structure, putting strong constraints on the hierarchical structure formation model and the nature of dark matter. Our current highest resolution halo contains five million particles inside its virial radius. ([18])

We now have a host of new observational tests of the hierarchical structure formation paradigm that include: the extent of galactic halos in clusters and satellites in galactic halos, the orbital distribution of cluster/satellite galaxies, the number of satellites as a function of their circular velocity or mass, the spatial and velocity distribution of satellites, and the central density profiles of clusters and galaxies. These properties have heretofore been the subject of endless debates among cosmologists because of the lack of a predictive theory: contradictory results can be had just by turning a few "knobs" on a phenomenological model. Our accurate numerical models will allow us to unambiguously interpret these observations.

The next step to take is to perform high resolution gas dynamical simulations of these clusters. This is particularly critical in order to make reasonable comparisons with data from X-ray satellites like CHANDRA. The simulations are needed in order to interpret what constraints the X-ray data can put on physical quantities such as the size of the X-ray core and the temperature structure of the cluster. We already have the capability to generate X-ray maps from our simulated clusters, but it is very computationally intensive. So, this capability will be moved into our parallel framework. Similar resolution and dynamic range will be needed in order to follow galaxy intra-cluster gas interactions, but the computational challenges will be greater because of the greater dynamical ranges in timescales introduced by the gas. New space-adaptive and time-adaptive parallel algorithms will allow us to rise to these challenges.

Likewise, we wish to study in detail the morphological evolution of galaxies as they orbit inside the gravitational potential of a galaxy cluster or a larger galaxy. The strong tidal field and rapid fly-bys with other orbiting galaxies drive a dramatic morphology evolution. Model spiral galaxies get destroyed or transformed into spheroidal systems with properties remarkably similar to those of observed galaxies in clusters or small satellites in the Local Group. Hence, we are able to construct a unified theory of the morphological diversity among these nearby galaxies.

The cluster simulations introduce a unique set of challenges, particularly in a massively parallel context. The large dynamic range in densities implies a large range in timescales: over a factor of 1000. We have developed time-adaptive algorithms for this problem, but making them efficient in parallel remains a challenge. Any small "fixed cost" quickly becomes a bottleneck because of the enormous dynamic range. Efficiently performing these simulations also requires a few instances of computational steering. First our "volume renormalization" technique requires the steering of the simulation so that a region of interest (where the cluster forms) is resolved sufficiently while keeping information about the cosmological context in low resolution. Secondly, as galaxies fall into the cluster, they need to be replaced with high resolution individual galaxy models if we are to properly follow their morphological transformation. These techniques are currently implemented in serial. Implementations in a parallel framework are needed for futher breakthroughs in this field.

Galaxy Formation

Galaxies are the most distinctive objects in the universe, containing almost all the luminous material. They are remarkable dynamical systems, formed by non-linear collapse and a drawn-out series of mergers and encounters. And yet, there are tight empirical relations, drawn from observations, that seem to apply to most of them.

We and our collaborators have a long experience in simulating galaxy formation. Navarro was among the first to simulate galaxy formation in a proper cosmological context ([32]). However, these simulations failed to create realistic spiral galaxies: the disk angular momentum was a factor of ten lower than in real spiral galaxies. After rapid cooling of gas inside the ubiquitous dense and small halos at high redshift the scene is set for a nasty surprise: gas loses too much angular momentum when it is trapped inside these halos as they spiral together to form the main body of the galaxy. The disk that forms from the assembly of gas has a scale length that is several times too small compared with real spiral galaxies. Somehow the gas has to remain in a diffuse state all the way up to the time it is accreted onto the central disk.

In the following years Quinn ([34]) introduced a realistic UV background (created by forming stars) and Steinmetz & Navarro ([33]) explored simple recipes for supernova feedback as possible solution. If the gas remains hot or is expelled out of the smaller halos by supernova winds its angular momentum loss should be significantly lower. Still, the problem of the "over-cooling catastrophe" remains, and no one has yet simulated the formation of a realistic disk galaxy from a cosmological simulation. This is a major embarrassment for the numerical cosmology community. We plan to join forces, take advantage of the large improvement in computing power afforded by the N–Chilada framework and tackle the problem again.

Galaxy formation is indeed a challenging computational problem: it requires a dynamic range of millions in each dimension, produces spatially varying timescales with a similar dynamic range, and involves resolution-sensitive interstellar medium physics. To form a stable Milky Way-like galaxy, tens of millions of resolution elements must be simulated to the current epoch. Locally adaptive timesteps reduce the CPU work by orders of magnitude, but not evenly throughout the computational volume: a considerable challenge for parallel load balancing. No existing N-body/Hydro solver can handle this regime efficiently; However, our *PKDGRAV/GASOLINE* [38] code must be considered the state of the art. A major thrust in the development of our framework is the creation of parallel algorithms that are fully spatially and temporally adaptive; thus making galaxy simulation a routine application that makes full use of parallel resources. This will be a watershed activity: only by thoroughly understanding galaxy formation will we be able to disentangle the light vs. mass problem that is the bane of cosmology.

Planet Formation

We have modified the PKDGRAV cosmology code to study the formation of planetary systems [37]. Here we are modeling the pairwise accretion of smaller bodies into proto-planets. In order to follow this process it is necessary to predict collisions (and near misses), and resolve the consequences of such collisions, whether they be mergers, fragmentations, or something in between. The computational challenge is to find an adaptive timestep scheme that is stable over millions of dynamical times yet that can handle close encounters accurately and is suitable for simulations involving millions of planetesimals. We have algorithms in hand, but implementing them in parallel has always been a challenge. A separate issue is handling the details of the collision physics. The parallel framework will be extensible enough to handle the small asteroid regime where material properties play an important role. Accurate simulations of this regime will allow us to correctly parameterize collisions in the large scale simulations.

The scientific payback here will be enormous. There are a number of outstanding fundamental questions about the origins of planetary systems which these simulations will be able to answer. What is the relationship between hot Jupiters and terrestrial-like planets? Do asteroid belts form around other stars (if so, this could be trouble for the Terrestrial Planet Finder, as dust may overwhelm the planet's signal)? How common are other small body reservoirs such as Kuiper Belts and Oort clouds? How effective are planetary systems at cleansing small bodies and thus mitigating sterilizing impacts on any terrestrial planets? What is the origin of the spin of the terrestrial planets? We are literally opening whole new worlds here.

3 Parallel Computing Research

The ambitious cosmology research proposed requires a combination of unprecendented raw compute power, advanced algorithms, and effective parallelization. As shown in Figure 2, changes in algorithms have actually outpaced the phenomenal increase in computer performance over the past few decades. However, both of these phenomena accentuate the parallelization problem. The disparity in the increase in computational and communication performance means the communication issues become more challenging over time; Complex algorithms are harder to parallelize, and often lead to severe load imbalances. Further, developing high performance parallel applications requires herculean efforts today, and applications developers need to be experts in numerical modeling (and algorithmic techniques) for the application domains, as well as in parallelization. These problems are exacerbated when the number of processors in individual parallel machines rises to tens of thousands, as it will in the next few years.

The Computer Science research is motivated by and will be demonstrated by improving programmer productivity and computer performance on the cosmology applications. However, because of the representative nature of this application, the parallel programming techniques developed will be useful for a wide variety of applications.

Our research in parallel computing is not based on the dominant processor-oriented paradigm exemplified by MPI or OpenMP. Instead, we decompose a parallel program into a set of communicating objects. Our system then maps these objects to processors, as shown in Figure 3. We share some goals with innovative parallel programming approaches such as [2, 42, 6, 40, 39] but

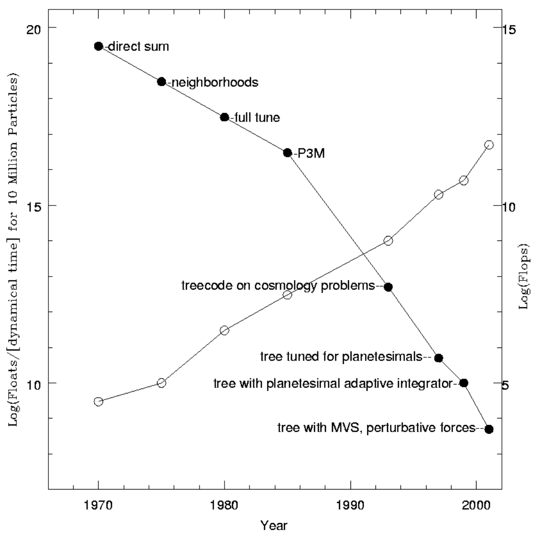


Figure 2: The evolution of algorithms and machines over three decades. Adaptive algorithms have provided gains that exceed the phenomenal increase in computer performance.

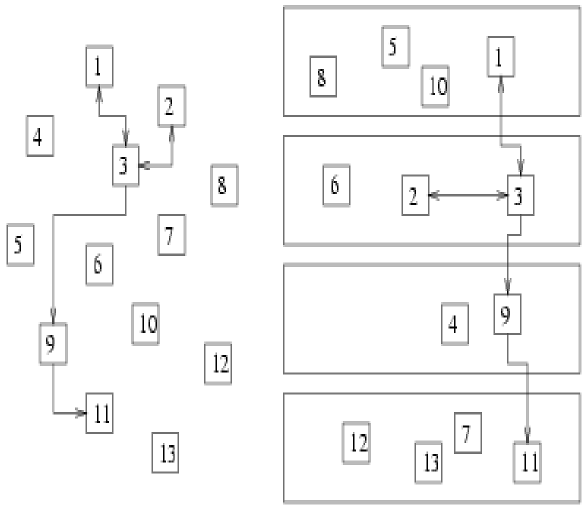


Figure 3: A Charm++ program consists of communicating objects. Charm++ system maps the objects to different processors, and handles the message delivery for the communication.

the specifics of our approach are different as they are based virtualization based on objects and the principle of persistence, as explained below.

We have successfully applied this object-based decomposition approach to several demanding application domains, including molecular dynamics (e.g. [24, 10, 22]), Finite Element computations ([3]), and rocket simulations (e.g. [16]).

3.1 Previous work: Object-Based Parallel Programming

The essential idea in object-based parallel computing is to create a separation of concerns between the application developer, who specifies algorithms as a set of communicating objects; and the parallel runtime system, which determines where and when to execute those objects to maximize performance.

Since the system typically places several objects on each processor, execution takes place in a *message driven* manner under the control of the scheduler, which automaticaly allows only those objects to continue that are not waiting for remote data. We have demonstrated that this approach naturally leads to an adaptive overlap of communication and computation. Objects can be migrated at run-time as needed for load balancing or other functions.

Basic communications (method invocations, reductions and broadcasts) are implemented using novel strategies to ensure they work in presence of such migrations [25]. Object migration has been used in a cluster environment with shared workstations, to "vacate" a workstation or to reduce the load on a workstation when the owner starts using it [8]. One of the most important uses of object based decomposition has been in automatic load balancing. We have used this capability for a molecular dynamics application, with an unprecedented speed up (1250 on 2,000 PEs), which led to a runners-up position for the Gordon Bell award in 2000 [21, 10].

Several additional capabilities, such as automatic checkpointing and automatic out-of-core execution, have also been developed based on this idea. Although the virtualization capabilities

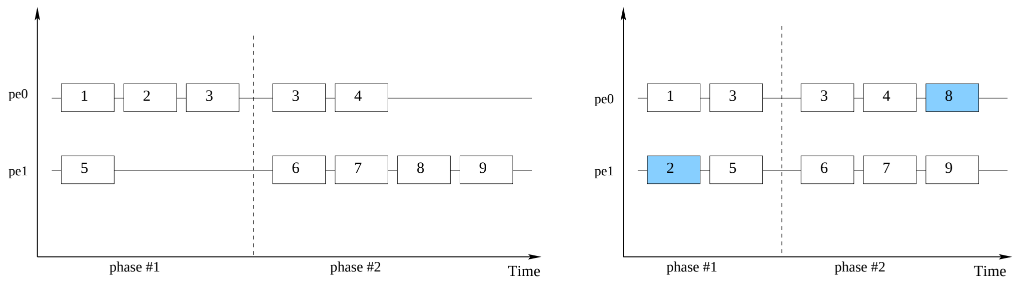


Figure 4: Balancing based on total load is inadequate in Multiphase Computation.

provided by the objects can be mimicked by explicit programming using traditional paradigms (MPI), we have shown that such an approach leads to complex programming and often negatively impacts modularity [23].

Currently, parallel object programs are written using C++ via the Charm++ system. A more recent development is that of AMPI [5] which allows MPI programs in C, Fortran 90, or C++ to utilize the object based approach. AMPI uses user-level threads to implement MPI processes, and migrates these threads across nodes as needed. A part of our effort will be spent on improving the AMPI interface and its thread migration capabilities. The detailed, application level instrumentation offered by the Charm++ runtime system will be used to develop performance visualizations and analysis techniques to aid in improving performance.

3.2 Principle of persistence

In computations expressed using communicating objects, an interesting property typically holds: we call it the principle of persistence, which can be thought of as an analog to the idea of principle of locality in sequential computing. This heuristic principle states that in a object-based program the computation times of individual objects and the inter-object communication patterns tend to persist over time. This means the recent past is a good predictor of near future. Interestingly, the principle still holds for dynamic applications involving adaptive refinements or system evolution. Typically, adaptive refinements are either infrequent or confined to regions of space; even evolving systems typically change slowly compared with their time steps or iterations. Note that we only claim that it is a heuristic principle, just like the principle of locality: it doesn't hold all the time, but often enough to permit systematic exploitation[24, 3].

As the subsections below describe, we propose to develop techniques to dramatically improve application performance and programmer productivity, by exploiting the separation of concerns provided by Charm++'s objects and the principle of persistence. The techniques will lead to automatic load balancing for complex, multi-phase applications on large parallel machines (with 10k-100k processors), automatic optimization of complex communication patterns, and the ability to connect multiple parallel modules in a flexible, dynamic and plug-and-play-manner.

3.3 Load Balancing

Load balancing issues can substantially affect the performance of complex applications such as those in cosmology. Based on the principle of persistence, we have already developed a load balancing framework ([9, 8]) that works effectively on machines with hundreds of processors for a significant class of applications. Automatic instrumentation via the runtime system is used to record object computation times and inter-object communication data: for each pair of communicating objects, the baseline instrumentation records the number of method invocations (messages) and the total number of bytes communicated. This information is refreshed periodically and used by a load balancing *strategy* that remaps objects to processors so as to minimize inter-processor communication and load imbalance.

A suite of load balancing strategies has been developed, because different strategies are better in different circumstances. For example, when load balancing is needed very infrequently, strategies that assigns objects "from scratch" are better, whereas for more frequent balancing *refinement-based* strategies that involve moving only a few objects are suitable. Some of the strategies are *centralized* (all performance data is brought to one processor, where the decisions are made) while others are *distributed*. However, all the current strategies aim at equalizing the *total* amount of work assigned to each processor– more precisely, they aim at minimizing the maximum load.

Although successful so far, these strategies are not adequate for the upcoming era of parallel computing, as exemplified by the cosmology applications. Fundamentally, there are two independent reasons for this: the large number of processors (10k-100k), and the application complexity. However, object based decomposition and automatic run-time instrumentation create the potential for handling even these situations automatically, as outlined below.

3.3.1 Proposed work: Load Balancing for Large machines

We have shown that centralized load balancers perform better than distributed ones, since they can globally optimize performance. Although many applications permit centralized stop-and-go load balancing (especially if it is infrequent), such a strategy is infeasible on large machines. For example, there simply may not be enough memory on one processor to hold all the performance data. One solution that we plan to explore is to develope continuous, fully distributed load balancers that attempt to equalize load within a local neighborhood. However, we expect that the best strategies will combine the global-optimization features of a centralized strategy with scalability of fully distributed ones. Development of such hybrid strategies is one of our major goals.

On machines with hundreds (or even a few thousand) processors, load balancers can typically ignore the communication topologies. (Although communication volume minimization is typically achieved by modeling send and receive costs as computational costs.) This is because with modern communication infrastructure (with wormhole routing), the communication delay between processors is almost independent of the number of hops traveled. However, on very large machines another effect starts to dominate: the more hops a message travels, the larger the fraction of the total machine bandwidth it uses. As large machines are bandwidth constrained because of physical factors, it becomes necessary to minimize the number of hops traveled by messages on the average. (For example, the bisection bandwidth of a 3-D toroid with P processors increases only as $P^{\frac{2}{3}}$). We plan to develop new object remapping techniques that will address this concern as well.

3.3.2 Proposed work: Load Balancing for Complex Application Structures

Multiple time stepping in tree codes provides one example of application complexity that impacts load balancing. Even when dual time stepping is used, the computation can be seen to break down into two phases, which repeat. It is not enough then to ensure that the total amount of work assigned to each processor is equal: it must be balanced within each phase individually (See figure 4). It is further complicated by the fact that some objects may be active in both phases of the computation. Further, phases may not be completely sequentialized. E.g. in an application with 3 parallel modules (A,B, and C), objects of module B may executes after those of module A, module C objects may execute concurrently with both A and B. We believe it is possible to add new instrumentation to create a relatively small amount of additional run-time data that captures the dependence structure among object computations. With use of such instrumentation, then, we will develop a new generation of load balancing strategies. Such strategies could explore a new degree of freedom to be effective: the load balancing decision may now involve some objects being continuously moved in a pre-planned manner between phases. So, the load balancer may dictate that object X be placed on processor 5 during phase 1 and processor seven during phase 2. Of course, the load balancer then must take into account the overhead of periodic object movement, and use this mechanism only when it leads to overall improved performance.

Although such strategies will be highly challenging to develop, we believe that they represent the logical culmination of the separation of concerns in object-based decomposition. With such strategies, programmers can truly ignore load balancing issues and focus on parallelization alone, making it much easier to develop complex high-performance parallel applications.

3.4 Adaptive Optimization of Commmunication Patterns

Parallel applications in general, and the cosmology codes in particular, exhibit several specific communication patterns. For example, in a tree code based on oct-trees, one may wish to send all the particles that have moved outside the box represented by their tree node to their new "home nodes". This leads to a communication pattern that may be dubbed "each-to-many individualized messages". The transmissions of particle coordinates from one box to all the others that need to calculate direct particle-to-particle intractions leads to "each-to-many multicast". In a k-d tree, repartitioning of the domain may (potentially) lead to a "permutation", with each particle moving to a new location, Unless carefully optimized, such communication operations may limit scalability severely. Object based programming, and the principle of persistence create an opportunity to automate such optimization.

Several algorithms are known for such optimizations [41]. For example, each-to-many-individualized pattern implemented in a straightforward fashion may lead to huge overheads: if each entity is sending messages to all others (say P), the overhead is proportional to P. If the messages are short (as they will be in case of particle migrations), this can be optimized by organizing processors in a 2D virtual grid (almost independet of the actual communication topology). If processor (x1,y1) wishes to send a message to (x2,y2), it sends it via (x1,y2). (x1,y2) waits for all contributions by processors within its row (x1,*), sorts the data receieved and sends separate messages to all the processors in its column. Thus each processor sends $2\sqrt{P}$ messages instead of P, although each message is larger by a factor of $\sqrt{P}$. Since per byte cost of messages (a few ns) is much lower than per-message cost (tens of μs), this repesents a substantial win for short messages. One can take this idea to extreme by using a binary hypercube instead of a 2-d grid (or several variations in between: k-ary n-cubes). Other variations such as spanning tree based distribution are also possible. When the comunicating entities are objects, instead or processors, another degree of complexity is added.

Deciding which algorithm among these will be appropriate in a situation depends on several factors including α, the per message cost, β, the per byte cost, size of messages, number of processors, the interconnection topology and the kind of communication hardware, for example. However, in an object based system, the run-time system is in a position to measure all these parameters. To take advantage of this, we will first develop interfaces for commonly needed communication operations. In addition to those defined above, we will focus on several other communication patterns, including multicasts and reductions on (possibly small) object subgroups. These are especially useful in tree codes, to send coordinates of particles to multiple recipents and to receieve force contributions back from them. For each communication pattern, we will develop several alternative algorithms, and determine regions of the parameter space where they are effective. This knowledge will be built into intelligent libraries that automatically select, and switch between alternative strategies at runtime in response to the characteristics of the communication pattern.

3.5 Component Technology

Cosmology simulations need to switch algorithms depending upon the amount of clustering. Initially, with weak clustering, it is found that grid-based algorithms are faster, but, as the clustering becomes stronger, tree-based algorithms are more efficient. Thus, the framework for such simulations needs to have a plug-and-play architecture for switching between algorithms. Also, in the analysis phase of large amounts of raw data, one needs to perform particle-based, neighborhood-based, or tree-based computations in parallel. The actual computations on the raw-data are application-specific, but the structures of these computations are similar. One could build specialized applications for each type of analysis, but this would entail loading terabytes of data for each instance of analysis of the same raw data. In order to increase researcher productivity, it would be desired to insert the analysis code into the running parallel program, which already has the data, rather than move the data to the code.

For efficient access to raw data, the plugged-in components (simulation algorithms, or analysis computations) have to be part of the same processes that comprise the parallel application. These are called *in-process components* in literature. Since these computations are inherently parallel, this presents challenges concerning distributed data-transfer, as well as parallel control-transfer. Existing component technologies either use process-based model for components, thus leaving the control transfer to the operating system with costly context switches, or they use a sequential blocking method invocation semantics, which are inadequte for parallel control transfers.

Message-driven paradigms, such as the core of Charm++, execute computations based on availability of data, so that control transfer between components is implicit in the data interchange between components. Thus, these paradigms unify the two aspects of parallel component interfaces: distributed data interchange, and parallel control transfer. For the distributed data transfer to be flexible, it is required that the parallel component interfaces be specified only at run-time. This late binding of interfaces is not possible for message-driven extensions of existing component architectures, because the interface model for these component architectures is based on a static object-interface model.

We have done some preliminary work on design of a novel interface architecture for parallel components [4]. Our proposed interface model separates the specification of behavior of parallel components from specification of connections between these components. It also provides flexibility and control for the runtime system to effectively load-balance the plugged-in components. It will be extended to dynamically load the parallel components into a running application, and to execute them in-process. We also propose to make this component architecture more efficient so that access to local raw cosmological data in the analysis phase is through more efficient shared memory, than the current message-based transfer. Also, we plan to investigate the commonly needed data-transfer patterns, and provide them as pluggable communication components, available for the runtime system to substitute in place of default libraries.

3.6 Parallel I/O

Cosmology computations and analysis manipulate large amounts of data. Our current simulations involve 270 millionis particles and produce about 300 GB of output. For any parallel application to be scalable over thousands of processors, all its components need to be scalable, and I/O has traditionally been the biggest bottleneck for scalability. A scalable simulation has to have a scalable software component to supply data, to record intermediate checkpoints, and to save data for analysis. Also, since it is likely that the entire dataset may not fit completely in memory, the software framework should provide for out-of-core computation, and a parallel I/O system is a must for that.

Today's parallel I/O systems have not been shown to be scalable beyond a thousand processors. This is mainly because they are cumbersome to use, and they need to be tuned for every application and hardware. Tuning I/O systems is very challenging, and is almost impossible for average users. Our focus for the last five years has ben the development of automatic tuning of I/O parameters for a parallel I/O system, Panda. We will build upon this research to improve the scalability of parallel I/O in Cosmology computations.

To promote use of parallel I/O systems in parallel computations, one needs to have a perfect impedance match between the abstractions provided by the I/O systems, and the abstractions provided by the parallel language or simulation framework. Today's parallel I/O systems support abstractions such as dense arrays, which are useful for grid-based computations. However, no known parallel I/O system provides tree-based abstractions, which are employed in computational astrophysics. A major focus of our proposed work would be to provide tree-based I/O abstractions.

Panda can utilize hints about upcoming I/O requests for optimizing I/O. Principle of persistence of computational load and communication structure also extends to I/O patterns. The Charm++ system currently records communication and computational behavior of applications. We plan to extend it to record I/O requests as well, and based on the previous I/O behavior, the Charm++ system will be made to provide hints to the Panda library.

Objects are a natural abstraction of parallelism in Charm++. They also provide the right abstraction for out-of-core execution with their ability for data encapsulation, and their broadly predictable access patterns (since objects are scheduled by the Charm++ scheduler). We have done preliminary work to show that Charm++ can support out-of-core execution transparently by enabling the scheduler to have lookahead capabilities, and issuing asynchronous prefetching commands for pending computations. We plan to integrate Panda to carry out the asynchronous fetching for Charm++ in parallel.

4 Tree Algorithms, Particle Framework and N-Chilada

New research is clearly needed to develop scalable and adaptive variants of tree codes that will work effectively on very large machines. However, the significant programming effort involved in such research impedes experimentation and often forces application developers to commit to their algorithmic choices early on. To alleviate this problem, we plan to research and develop new parallel algorithms, create efficient implementations of known algorithms, and incorporate them all into a comprehensive framework. Our proposed framework, named *N-Chilada*, will allow application developers to quickly build integrated applications by assembling parallel modules using a high-level scripting language. It will thus be a general purpose framework for supporting simulation, analysis and visualization of irregularly structured particle data on massively parallel computers.

Research on the framework will require extensive collaboration between the Washington and Illinois researchers. The component framework will be developed at Illinois; modules will contributed by each team, and some modules will be developed collaboratively.

4.1 Scalable parallelization of tree codes

Tree codes based on Barnes-Hut and other algorithms have been parallelized on hundreds of processors; PKDGRAV embodies the best-known techniques to yield a parallel efficiency of over 70

percent on 256 processors. To increase efficiency, and to scale to a much larger number of processors, new parallelization techniques will be developed.
Efficient Octree algorithms: Octrees are shown to have theoretical advantages over binary (k-d) trees for N-body algorithms [1] However, in traditional parallel implementations, octrees pose problems such as requiring a power of eight processors, which lead to the adoption of binary trees. The object-based approach enables an octree to be distributed over any number of processors, and octree codes become easier to develop. We plan to create new and efficient parallel implementations of such codes.
Tree rebuilding will become a severe bottleneck on large machines. We will develop fully distributed algorithms that parallelize this phase effectively, for both binary and oct trees. This research will benefit from (and provide directions for) the communication research described in Sec. 3.4.

While parallelizing molecular dynamics, the CS group at Illinois has developed a new **hybrid space-and-force decomposition** technique that is likely to be beneficial to the gravity codes as well. In this technique, the pairwise interactions between two cells are themselves thought of as a separate computational object that can be assigned to a third processor if necessary. This provides a degree of freedom that leads to better load balance, and optimized tree decompositions. Complex tradeoffs present themselves, when one considers combinations of depth of the object-tree, and depth of the tree within each object. A thorough analysis will be conducted, and efficient choices for such parameters identified.

4.2 Dynamic Components, Scripting and Steering

N-chilada will provide abstract APIs for commonly needed operations on particle data sets. Using a modular design, the details of the parallel implementation will be hidden from the end user. This will allow rapid development of new simulations and high performance analysis tools without requiring extensive knowledge of the complicated underlying parallel issues. New analysis procedures will be able to be run in parallel by writing only serial code: either at a high level to control the flow of the computation or at the individual processor level for instructions to operate on individual particles. The framework will allow the user to choose different algorithms to evolve the data set. For example, gravity calculations on weakly clustered particle distributions are most efficiently done on a grid; as the clustering gets stronger, trees become more efficient. Within the framework, we can easily switch between these two techniques at run time.

Our current production codes *PKDGRAV* and *GASOLINE* have been designed (at Univ. of Washington) with modularity and extensibility in mind. The success of the design was proven by the ease with which we have taken the code originally designed for cosmological dark matter simulations and applied it to new physical regimes such as Solar System formation, asteroid collisions, and even collisional dynamics in grains of sand. *N-Chilada* will build on the design of these codes, and extend them to make them useful to a broader community.

Based on the ability to add, remove and substitute parallel components runtime (Sec. 3.5), we will develop a high level scripting system that allows user to control the application easily. This will allow dynamic switching of algorithmic modules, dynamic spawning of analysis and I/O modules, and ability to replace regions with high resolution models at runtime, under the control of a script that may be pre-written, or dynamically loaded into a running application.

Based on the client-server interface in Charm++, we will build web-based steering interfaces that allow scientists to monitor an ongoing simulation and interactively trigger analysis, visualization snapshot, outputs, as well as algorithmic controls.

4.3 Framework Support

Structured Grids: Structured grids are often used, for example in more uniform astrophysical gas dynamical simulations, and in particle-mesh algorithms. We will develop methods that use spatial decomposition information to effectively interface with such grids, so that data can be easily transferred and interpolated between the two representations.
File Format and I/O: Standard (XML based) file formats for particle data will be developed for a variety of data structures (such as trees), and supported by parallel I/O primitives, and their adaptive implementations (See Sec. 3.6).
Post-processing support: Analyzing the simulation data is currently the bottle neck: transferring and translating huge data files, writing specialized code and laborious analysis in serial. Harnessing the power of a parallel machine will allow us to drastically increase our scientific output.
Other disciplines: Our framework will be used in other scientific disciplines which we cannot discuss in detail here. For example, Markiel and Quinn are currently collaborating with G. Seidler of the UW physics department on a granular dynamics project. *PKDGRAV* simulations are being compared to laboratory experiments to determine the dynamics of packing spheres. Quinn has an ongoing collaboration with V. Daggett of UW Medicinal chemistry to incorporate protein folding physics into the parallel framework.

Specific plans with milestones for research on the particle framework, and how they relate to the research on parallel computing and cosmology are described in the management plan section.

5 Broader Impact

5.1 Education and Outreach

The PIs are committed to making graduate and undergraduate education a significant factor in their research activities. Each PI has trained and are training several graduate students in their research groups, and have supervised undergraduate students, some supported by the NSF REU program. Graduate students will be involved in both the building the framework and using it for science in the work proposed here. Summer undergraduate researchers will be users of the framework.

Web based interaction with cosmological simulation data: The web will be used to diseminate educational material in cosmology. Results of our simulations will be made accessible via the web to interested students, and possibly K-12 schools. Snapshots of the universe, galaxies, planets in their formation stages, as simulated starting from basic assumptions, will be highly educational to students, and will increase their scientific curiosity. More advanced questions, such as how slightly differing assumptions may lead to very different spatial structures, will also be illustrated via the web-based simulation snapshots. The dynamic and 3-dimensional nature of the structures involved requires parallel computational resources to properly convey their properties. Allowing simulation states to be viewed via the web interactively requires a parallel cluster that houses a parallel program that interactively processes the queries. We will use existing clusters at Illinois and Washington in the PI's laboratory for this purpose initially, and upload them to other clusters where available.

5.2 Software tools

The computational techniques developed in this project will be embodied in software frameworks, and distributed for use by the wider community. Parallel computing framework will be distributed via a server in Illinois, the particle framework from both Illinois and Washington, and cosmology-specific software from Washington.

Although the research here is focused on cosmology, particle based simulation techniques have a broad range of applications. The design of the framework is general, so as to support many of these applications, including:
Granular dynamics, where the particles are grains of sand or beads, and we are modeling how they pack (in collaboration with G. Seidler of the UW physics department),
Molecular dynamics, where the particles are atoms (Quinn has an ongoing collaboration with V. Daggett of UW Medicinal chemistry to incorporate protein folding physics into the framework),
Microelectronics, where the particles are sections of a chip surface (in collaboration with Vikram Jandhyala of the UW Electrical Engineering department).

6 Results From Prior NSF Support

We briefly describe some of the relevant results achieved by the PIs in prior NSF grants.

Previous NSF Support: Prof. Laxmikant Kale:

OPAAL: Simulation and Optimization of Casting and Extrusion Processes. Oct. 1998 through Sep. 2001. Award No. 9873945. Total Budget: $2,239,401. 1 of 11 PIs.

This project was funded by the OPAAL program, which aimed at bringing together pure mathematicians, Engineers and Computer Scientists together in interdisciplinary projects. Our part of this project involved supporting parallelization of multiple applications. Significant results include development of a framework for Unstructured Mesh (e.g. Finite Element) Computations [3] which was used by other co-PIs to parallelize their applications succesfully. In addition, a collision detection ("contact") algorithm with high parallel performance has also been developed [26], to work with the FEM framework in structural simulation. The resultant codes automatically adapt to application induced or external load variances to restore high performance. Preliminary work on Adaptive MPI ([5])was also supported in part by this grant.

Very recently, we have received another NSF ITR grant, involving many of the PIs of the above grant. The project has recently started, and as yet there are no results to report. One of the early projects we have started on is that of parallelization of the novel space-time meshes.

Although the support amount on these grants was relatively small, these efforts have given us invaluable experience in ensuring our techniques are relvant for a specturum of applications.

Previous NSF Support: Prof. Marianne Winslett:

Prof. Winslett was supported between 1989 and 1994 by NSF IRI 8958582 ($62,500 per year), a Presidential Young Investigator Award. This award funded her research in several areas of databases and AI, with an emphasis on the problems of updating logical knowledge bases and of database security. The PhD research of three students was supported by this prior grant.

The publications resulting from work on updating logical knowledge bases explore semantics and algorithms that can be used to revise knowledge expressed as logical formulas, with an emphasis on the case where the new knowledge acquired contradicts what was previously believed. The work also explored the close ties between the update problem and non-monotonic formalisms for reasoning, such as circumscription, and exploits the connection to suggest a new non-monotonic reasoning approach that allows one to reach default conclusions involving the equality predicate that are not obtainable using traditional circumscriptive techniques.

Prof. Winslett has been working on parallel I/O in recent years, with funding from other agencies. This work, which has already proved valuable in simulation of rockets, for example, is highly relevant to the research proposed here.

Publications: [11, 36, 46, 45, 47, 12, 13, 44, 43, 35]

Previous NSF Support: Prof. Thomas Quinn:

Quinn has recieved most of his previous High Performance Computing support from the NASA HPCC program and the NASA AISR program. The exception is NSF Award 9973209 *Direct Simulation of Planetary System Formation* (award amount: $ 239,993, period: 7/1/99-6/30/02) from the NSF Planetary Astronomy Program. In this project, Quinn's high performance parallel cosmology code was adapted to integrate the orbits of planetesimals in a protoplanetary disk, as well as detecting and resolving collisions between the planetesimals. We have been successful in simulating a disk of millions of planetesimals, and are implementing implementing better algorithms and running on faster machines to increase the timespan and particle number. Preliminary results indicate that Jupiter acts as a suppressor of planet formation in the Asteroid belt, *i.e.*, the Asteroid belt is a "failed planet" because the proximity of Jupiter prevents the buildup of larger bodies in this region. Details of these results are given in [37].

We have also used the same code to study details of the collisions between asteroids, modeling each asteroid as a "rubble pile", *i.e.* a loose agregate held together only by gravity. In [27] we present results from a parameter study of collisions between kilometer-sized spherical rubble piles. The results will assist in parameterization of collision outcomes for Solar System formation models and give insight into disruption scaling laws. We find that our rubble piles are relatively easy to disperse, even at low impact speed. This may provide a way of constraining the energy dissipation parameter and related properties of the initial planetesimal population.

7 Management Plan

The research will involve coordination of plans at multiple sites. We will use web based tools for continuous interactions, internet-accessible source control software (CVS) for managing common code base, monthly scheduled teleconferences. In addition, we expect to have 3-4 weeklong trips (2 hosted at each site) per year for software and research coordination. We also will have a workshop style meeting once a year, where recent results will be presented and plans will discussed and developed.

The staff and students at UI will be supervised by Profs. Kale and Winslett, while those at UW will be supervised by Prof. Quinn. However, reflecting the close interaction required in developing the particle framework, we will employ task-specific team structures, with different personnel taking the lead position on each task, as decided by the PIs.

The research in parallel computing, particle algorithms and cosmology will be coordinated using the milestones indicated below.

Year	Milestone
1	Design of particle framework in CHARM++
1	Comprehensive study of Centralized and fully distributed load balancers
1	Communication patterns catalogued, and APIs developed
1	I/O requirements studied, and detailed research plan made
1	Static Components technology with distributed data exchange
1	Implementation of PKDGRAV's tree algorithm in framework
1	Instrumentation for performance testing
1	Preliminary performance studies of prototype tree algorithms
2	Testing of a variety of tree walk algorithms
2	Incorporation of Topology-conscious load balancers
2	Development and evaluation of Communication algorithms
2	Incorporating FMM type algorithms into the framework
2	Implementing gas dynamics within framework
3	Runs of large-scale structure with gas performed with .5 billion particles
3	Multiphase load balancers demonstrated
3	Dynamic components with runtime binding of communication demonstrated
3	Galaxy cluster simulations with gas, cooling and stars with millions of particles
3	Intelligent strategy selection for Communication Libraries
3	High level distributed and remote/interactive scripting via dynamic components
3	Runtime instrumentation for parallel I/O and Communication patterns
3	New multistep algorithms incorporated and tested
3	Planetesimal collision detection incorporated into framework
3	Scaling tested to 2,000 processors
4	Complete and integrated suite of load balancing capabilities incorporated
4	Communication libraries, with intelligent selection, demonstrated on large parallel machines
4	Planet formation run with 10 million particles for millions of years
4	Galaxy formation simulations performed with 10s of millions of particles
4	Scaling demonstrated on 10,000+ processors. (largest available machine)
5	Run of large-scale structure with 10 billion particles

7.1 Management of Relevant External Collaborations

Several other collaborations that the PIs are involved in will be impacted by the proposed work. We describe how these collaborations will be managed in the context of the proposed work.

7.1.1 The N-Chilada collaboration

In an effort to build a community of potential users of the parallel framework and assess their needs, Quinn has organized a collaboration of computational scientists. These include members of the C4 collaboration (below) former UW post-docs and graduate students, and other colleagues. Two meetings have been held to organize the effort and to solicit feedback. Continual discussions are taking place via a collaboration website, with more than twenty contributors. The breadth and number of this community is an indication of the felt need for the development that we are proposing.

Quinn and Kale have submitted a proposal to NASA's Applied Information Systems program which focuses on the visualization and analysis needs of this collaboration. This is in marked contrast to the goals of this proposal where we are focusing on break-through parallel algorithm research. Nevertheless there will be a synergy between the efforts particularly in the development of a framework upon which both visualization systems and new massively parallel algorithms can be built.

7.1.2 The C4 collaboration

Collaborators include a number of participants in the Canadian Computational Cosmology Collaboration (C4): Babul, Navarro, Stadel, Couchman and Wadsley. This is a Canadian-based group specializing in massively parallel cosmological simulations of galaxy systems. Quinn and his group at the University of Washington, have had a particularly close relationship with C4, sharing scientific goals, postdoctoral positions, codes, and parallel machine expertise (see supporting letter). Babul, the director of the C4 is a scientific collaborator on the Large Scale Structure and Cluster projects. Joachim Stadel and Wadsley, both formerly at UW are now making *PKDGRAV* and *GASOLINE* available to C4 members.

In this proposal we seek to continue to support this successful international collaboration. Stadel and Wadsley will be key developers of the computational cosmology code. Other members of the collaboration, in particular, Navarro, Babul and their graduate students will be "power users" of the framework. This ensures both a community of users, and quick feedback on usability and scientific correctness.

References

[1] R. J. Anderson. Tree data structures for n-body simulation. In *Proc. 37th Ann. Symp. Foundations of Comp. Sci.*, pages 224–223, 1997.

[2] P. Beckman and D. Gannon. Tulip: A portable run-time system for object-parallel systems. In *Proceedings of the 10th International Parallel Processing Symposium*, April 1996.

[3] Milind Bhandarkar and L. V. Kalé. A Parallel Framework for Explicit FEM. In M. Valero, V. K. Prasanna, and S. Vajpeyam, editors, *Proceedings of the International Conference on High Performance Computing (HiPC 2000), Lecture Notes in Computer Science*, volume 1970. Springer Verlag, December 2000.

[4] Milind Bhandarkar and L. V. Kale. An Interface Model for Parallel Components. In *Proceedings of the Workshop on Languages and Compilers for Parallel Computing (LCPC), Cumberland Falls, KY*, August 2001.

[5] Milind Bhandarkar, L. V. Kale, Eric de Sturler, and Jay Hoeflinger. Object-Based Adaptive Load Balancing for MPI Programs. In *Proceedings of the International Conference on Computational Science, San Francisco, CA, LNCS 2074*, pages 108–117, May 2001.

[6] F. Bodin, P. Beckman, D. Gannon, S. Narayana, and S. Yang. Distributed pC++: Basic Ideas for an Object Parallel Language. *Scientific Programming*, 2(3), 1993.

[7] S. Borgani, F. Governato, J. Wadsley, N. Menci, P. Tozzi, G. Lake, T. Quinn, and J. Stadel. Preheating the Intracluster Medium in High-Resolution Simulations: The Effect on the Gas Entropy. *Astrophys. J. Lett.*, 559:L71–L74, October 2001.

[8] Robert K. Brunner and Laxmikant V. Kalé. Adapting to load on workstation clusters. In *The Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 106–112. IEEE Computer Society Press, February 1999.

[9] Robert K. Brunner and Laxmikant V. Kalé. Handling application-induced load imbalance using parallel objects. Technical Report 99-03, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1999. Submitted for publication.

[10] Robert K. Brunner, James C. Phillips, and Laxmikant V. Kale. Scalable Molecular Dynamics for Large Biomolecular Systems. In *Proceedings of Supercomputing (SC) 2000, Dallas, TX, November 200 0. Nominated for Gordon Bell Award.*, November 2000.

[11] T. S-C Chou and M. Winslett. A model-based belief revision system. *Journal of Automated Reasoning*, 12(2):157–208, June 1994.

[12] Timothy S.-C. Chou and M. Winslett. Immortal: A model-based belief revision system. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge MA*, pages 99–110. Morgan Kaufmann, April 1991.

[13] Timothy S.-C. Chou and M. Winslett. The implementation of a model-based belief revision system. In *AAAI Spring Symposium on Implemented Knowledge Representation and Reasoning Systems, Palo Alto CA*, pages 59–72. March 1991.

[14] M. M. Crone, F. Governato, J. Stadel, and T. Quinn. Prospects for Cosmology with Cluster Mass Profiles. *Astrophys. J. Lett.*, 477:L5–+, March 1997.

[15] P. de Bernardis, P. A. R. Ade, J. J. Bock, J. R. Bond, J. Borrill, A. Boscaleri, K. Coble, C. R. Contaldi, B. P. Crill, G. De Troia, P. Farese, K. Ganga, M. Giacometti, E. Hivon, V. V. Hristov, A. Iacoangeli, A. H. Jaffe, W. C. Jones, A. E. Lange, L. Martinis, S. Masi, P. Mason, P. D. Mauskopf, A. Melchiorri, T. Montroy, C. B. Netterfield, E. Pascale, F. Piacentini, D. Pogosyan, G. Polenta, F. Pongetti, S. Prunet, G. Romeo, J. E. Ruhl, and F. Scaramuzzi. Multiple Peaks in the Angular Power Spectrum of the Cosmic Microwave Background: Significance and Consequences for Cosmology. In *15 pages, 6 figures, submitted to Ap.J*, pages 5296+, May 2001.

[16] Eric deStruler, Jay Hoeflinger, L. V. Kale, and Milind Bhandarkar. A New Approach to Software Integration Frameworks for Multi-physics Simulation Codes. In *Proceedings of IFIP TC2/WG2.5 Working Conference on Architecture of Scientific Software, Ottawa, Canada*, pages 87–104, October 2000.

[17] S. Ghigna, B. Moore, F. Governato, G. Lake, T. Quinn, and J. Stadel. Dark matter haloes within clusters. *Mon. Not. R. astr. Soc.*, 300:146–162, October 1998.

[18] S. Ghigna, B. Moore, F. Governato, G. Lake, T. Quinn, and J. Stadel. Density Profiles and Substructure of Dark Matter Halos: Converging Results at Ultra-High Numerical Resolution. *Astrophys. J.*, 544:616–628, December 2000.

[19] F. Governato, A. Babul, T. Quinn, P. Tozzi, C. M. Baugh, N. Katz, and G. Lake. Properties of galaxy clusters: mass and correlation functions. *Mon. Not. R. astr. Soc.*, 307:949–966, August 1999.

[20] F. Governato, C. M. Baugh, C. S. Frenk, S. Cole, C. G. Lacey, T. Quinn, and J. Stadel. The seeds of rich galaxy clusters in the Universe. *Nature*, 392:359–361, 1998.

[21] Attila Gursoy. Simplified expression of message-driven programs and quantification of their impact on performance. Technical Report UIUCDCS-R-94-1852, 1994.

[22] L. V. Kalé, Milind Bhandarkar, Robert Brunner, N. Krawetz, J. Phillips, and A. Shinozaki. Namd: A case study in multilingual parallel programming. In *Proc. 10th International Workshop on Languages and Compilers for Parallel Computing*, Minneapolis, Minnesota, August 1997.

[23] L. V. Kale and Attila Gursoy. Modularity, reuse and efficiency with message-driven libraries. In *Proc. 27th Conference on Parallel Processing for Scientific Computing*, pages 738–743, February 1995.

[24] Laxmikant Kalé, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151:283–312, 1999.

[25] O. Lawlor and L. V. Kalé. Supporting dynamic parallel object arrays. In *Proceedings of International Symposium on Computing in Object-oriented Parallel Environments*, Stanford, CA, Jun 2001.

[26] Orion Lawlor. A grid-based parallel collision detection algorithm, 2001. http://charm.cs.uiuc.edu/papers/CollisionThesis01.html.

[27] Z. Leinhardt, D. C. Richardson, and T. Quinn. Direct N-body Simulations of Rubble Pile Collisions. *Icarus*, 146:133–151, July 2000.

[28] G. F. Lewis, A. Babul, N. Katz, T. Quinn, L. Hernquist, and D. H. Weinberg. The Effects of Gasdynamics, Cooling, Star Formation, and Numerical Resolution in Simulations of Cluster Formation. *Astrophys. J.*, 536:623–644, June 2000.

[29] B. Moore, F. Governato, T. Quinn, J. Stadel, and G. Lake. Resolving the Structure of Cold Dark Matter Halos. *Astrophys. J. Lett.*, 499:L5–+, May 1998.

[30] B. Moore, G. Lake, T. Quinn, and J. Stadel. On the survival and destruction of spiral galaxies in clusters. *Mon. Not. R. astr. Soc.*, 304:465–474, April 1999.

[31] B. Moore, T. Quinn, F. Governato, J. Stadel, and G. Lake. Cold collapse and the core catastrophe. *Mon. Not. R. astr. Soc.*, 310:1147–1152, December 1999.

[32] J. F. Navarro, C. S. Frenk, and S. D. M. White. The assembly of galaxies in a hierarchically clustering universe. *Mon. Not. R. astr. Soc.*, 275:56–66, July 1995.

[33] J. F. Navarro and M. Steinmetz. Dark Halo and Disk Galaxy Scaling Laws in Hierarchical Universes. *Astrophys. J.*, 538:477–488, August 2000.

[34] T. Quinn, N. Katz, and G. Efstathiou. Photoionization and the formation of dwarf galaxies. *Mon. Not. R. astr. Soc.*, 278:L49–L54, February 1996.

[35] P. K. Rathmann and M. Winslett. Circumscribing equality. In *Proceedings of the International Joint Conference on Artificial Intelligence, Detroit*, pages 468–476. Morgan Kaufmann, August 1989.

[36] P. K. Rathmann, M. Winslett, and M. Manasse. Circumscription with homomorphisms: Solving the equality and counterexample problems. *Journal of the ACM*, 41(5):819–873, September 1994.

[37] D. C. Richardson, T. Quinn, J. Stadel, and G. Lake. Large-Scale N-body Simulations of Planetesimal Dynamics. *Icarus, 143, 45*, 2000.

[38] J. G. Stadel. *Cosmological N-body Simulations and their Analysis*. PhD thesis, Department of Astronomy, University of Washington, March 2001.

[39] Xinan Tang and Guang Gao. Automatically partitioning threads for multithreaded architectures. *Journal of Parallel and Distributed Computing*, 58:159–189, 1999.

[40] Kevin B. Theobald, Rishi Kumar, Gagan Agrawal, Gerd He ber, Ruppa K. Thulasiram, and Guang R. Gao. Developing a communication intensive application on earth multithreaded architecture. In *Proceedings of Europar 2000*, Munich, Germany, 2000.

[41] Sathis Vadhiyar, Graham Fagg, and Jack Dongarra. Towards and Accurate Model for Collective Communications. In *Proceedings of the International Conference on Computational Science, San Francisco, CA, LNCS 2073*, pages 41–50, May 2001.

[42] Chih-Po Wen, Soumen Chakrabarti, Etienne Deprit, Arvind Krishnamurthy, and Katherine Yelick. Run-time support for portable distributed data structures. In *Third Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers*, Rensselaer Polytechnic Institute, NY, May 1995.

[43] M. Winslett. Sometimes updates are circumscription. In *Proceedings of the International Joint Conference on Artificial Intelligence, Detroit*, pages 859–863. Morgan Kaufmann, August 1989.

[44] M. Winslett. *Updating Logical Databases.* Cambridge University Press, 1990.

[45] M. Winslett. Circumscriptive semantics for updating knowledge bases. In *Annals of Mathematics and Artificial Intelligence*, volume 3, pages 429–450. J. C. Baltzer AG Scientific Publishing Company, March 1991.

[46] M. Winslett. Epistemic aspects of databases. In Y. Shoham, P. Gärdenfors, and D. M. Gabbay, editors, *The Handbook of Logic and Artificial Intelligence 3.* Oxford University Press, 1994.

[47] M. Winslett and T. S.-C. Chou. Updates with equality: Beyond the herbrand universe assumption. In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems, Charlotte, NC*, pages 276–285. Springer-Verlag, October 1991.

[48] D. G. York, J. Adelman, J. E. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, J. A. Bakken, R. Barkhouser, S. Bastian, E. Berman, W. N. Boroski, S. Bracker, C. Briegel, J. W. Briggs, J. Brinkmann, R. Brunner, S. Burles, L. Carey, M. A. Carr, F. J. Castander, B. Chen, P. L. Colestock, A. J. Connolly, J. H. Crocker, I. ;. Csabai, P. C. Czarapata, J. E. Davis, M. Doi, T. Dombeck, D. Eisenstein, N. Ellman, B. R. Elms, M. L. Evans, X. Fan, G. R. Federwitz, L. Fiscelli, S. Friedman, J. A. Frieman, M. Fukugita, B. Gillespie, J. E. Gunn, V. K. Gurbani, E. de Haas, M. Haldeman, F. H. Harris, J. Hayes, T. M. Heckman, G. S. Hennessy, R. B. Hindsley, S. Holm, D. J. Holmgren, C. Huang, C. Hull, D. Husby, S. Ichikawa, T. Ichikawa, Željko Ivezić, S. Kent, R. S. J. Kim, E. Kinney, M. Klaene, A. N. Kleinman, S. Kleinman, G. R. Knapp, J. Korienek, R. G. Kron, P. Z. Kunszt, D. Q. Lamb, B. Lee, R. F. Leger, S. Limmongkol, C. Lindenmeyer, D. C. Long, C. Loomis, J. Loveday, R. Lucinio, R. H. Lupton, B. MacKinnon, E. J. Mannery, P. M. Mantsch, B. Margon, P. McGehee, T. A. McKay, A. Meiksin, A. Merelli, D. G. Monet, J. A. Munn, V. K. Narayanan, T. Nash, E. Neilsen, R. Neswold, H. J. Newberg, R. C. Nichol, T. Nicinski, M. Nonino, N. Okada, S. Okamura, J. P. Ostriker, R. Owen, A. G. Pauls, J. Peoples, R. L. Peterson, D. Petravick, J. R. Pier, A. Pope, R. Pordes, A. Prosapio, R. Rechenmacher, T. R. Quinn, G. T. Richards, M. W. Richmond, C. H. Rivetta, C. M. Rockosi, K. Ruthmansdorfer, D. Sandford, D. J. Schlegel, D. P. Schneider, M. Sekiguchi, G. Sergey, K. Shimasaku, W. A. Siegmund, S. Smee, J. A. Smith, S. Snedden, R. Stone, C. Stoughton, M. A. Strauss, C. Stubbs, M. SubbaRao, A. S. Szalay, I. Szapudi, G. P. Szokoly, A. R. Thakar, C. Tremonti, D. L. Tucker, A. Uomoto, D. Vanden Berk, M. S. Vogeley, P. Waddell, S. Wang, M. Watanabe, D. H. Weinberg, B. Yanny, and N. Yasuda. The Sloan Digital Sky Survey: Technical Summary. *Astron. J.*, 120:1579–1587, September 2000.