Cover Page

National Science Foundation
Solicitation NSF 08-592:

ACCELERATING DISCOVERY IN SCIENCE AND ENGINEERING THROUGH PETASCALE SIMULATIONS AND ANALYSIS (PetaApps)

A Petascale Environment for Simulation of Protein Folding and Recognition

For the period July 1, 2009 – June 30, 2013

Submitted October 30, 2008

Principal Investigator

Karl Freed
The James Franck Institute
University of Chicago
Email: freed@uchicago.edu

Co-Principal Investigators

Tobin Sosnick
Institute for Biophysical Dynamics
University of Chicago
Email: trsosnick@uchicago.edu

Michael Wilde
Computation Institute
University of Chicago
Email: wilde@mcs.anl.gov

The scientific goal of this project is to predict protein structure and recognize docking partners at unprecedented speed, capacity and accuracy. To achieve this computationally intensive goal we utilize a petascale-proven platform-independent application-neutral framework for running loosely coupled simulations on grids and petascale systems. The unique amino acid sequence of a protein determines its secondary (2^o) structure elements (e.g., helices and strands) which assemble into a 3D tertiary (3^o) structure. This "folded" structure determines the protein's function and its docking partners. The centrality of these processes to a vast array of scientific questions makes our capability a transformative enabler.

This project is crucial for treating docking and will enable the OOPS folding code to run on all of the major open-science petascale resources of the next five years and beyond, enabling it to predict the structure of large proteins and currently unsolvable docking targets. Prototypes demonstrate that OOPS runs on Blue Gene P (BG/P) at scales ranging up to 65,536 processors, using an innovative computing framework. This framework is tested at scales up to 163,840 processors and employs petascale systems in a hybrid loosely/tightly coupled manner which will scale efficiently to future petascale systems. The OOPS user interface enables petascale use of OOPS via web and service interfaces and custom scripts.

Applicability to this program: Program timing is ideal as the OOPS code has just proven successful in predicting the structure of small proteins. Scaling to larger proteins requires greater computing resources. Our team is already doing coding and testing on petascale hardware. We have worked extensively with the BG/P and TeraGrid Ranger, and are to our knowledge the only group to date that has executed a wide variety of applications at scales above 64K cores using a generic middleware platform.

Furthermore, the proposed research requires the close interactions within our multidisciplinary chemistry/computer science team that extends beyond projects funded by Core programs. The demands of the simulations impose constraints that guide the development of the framework for loosely coupled petascale computing. The petascale framework, in turn, is necessary to meet the demands of the heavier computing associated with docking. Our collaborative effort of porting OOPS to large scale computing environments requires an integration of petascale computing and scientific skills not fundable elsewhere.

Intellectual merit: Outstanding challenges in the folding and docking fields include (a) predicting large and multi-domain proteins, particularly those where subdomains have little homology to known structures, and (b) the prediction of binding partners (and the relaxation upon binding) in order to map out interaction networks within a single organism. To address these challenges, petascale methods are essential.

While the folding/docking field has been active for decades, petascale OOPS is needed because most other algorithms rely heavily on known structures or fragments, being either homology- or template-based. Their success rapidly diminishes as the amount of known information decreases. Similarly, the reliable protein-protein docking is a hit-or-miss affair and often requires backbone relaxation for success. The identification of the strongest pairwise interactions and docking interfaces within a group of proteins remains a long-range goal. Efficient, scalable new approaches are needed.

Nearly uniquely, OOPS incorporates basic chemical principles and mimics a folding pathway to restrict search space. As folding often is robust to side chain substitution, we believe that fine side chain details are unnecessary and a "C_β" representation is the most appropriate for the problem at hand *so long as the side chain information can be properly retained*. *Our method retains this information.* The reduced representation enables a broader and faster search, and backbone relaxation is accomplished within the condensed state using our specialized move set. In contrast, most methods include all the atoms and consequently expend a majority of their computational time searching through a rugged side chain space. Given the importance in starting with the correct 2^o structure, other 3^o structure prediction algorithms can take advantage of our 2^o structure predictions which are more accurate for low homology sequences than existing methods.

Broader impacts: The flexibility and availability of OOPS will transform the way protein structure can be predicted and assessed, making petascale systems a tool readily usable for characterizing protein recognition and the ever growing number of proteins that has emerged from sequencing projects. OOPS has sufficient inherent parallelism to scale to million core processors within the decade. The petascale framework will be freely available for use with other loosely coupled simulation algorithms in many diverse areas, simply by inserting a new simulation algorithm into the petascale framework. The current toolkit runs on Blue Gene P, Constellation, and XT4, and will support the architecture of Blue Waters. User facilities at the UChicago Computation Institute will serve a large, diverse, interactive, and collaborative user group for data staging, and result analysis. The training of students (undergrads, grads, postdocs, and users) will be an essential part of our broad EOT program.

A Petascale Environment for Simulations of Protein Folding and docking

1 Overview

As the protein targets increase in size, the need for petascale computing becomes urgent. Current prediction methods have limited accuracy even for proteins on the order of 100 residues when homology-based information is minimal. To fold larger and multi-domain proteins, statistical sampling becomes a limiting factor. Additionally, current docking algorithms are inconsistent. The most glaring issue in the folding of multi-domain proteins and the docking of complexes is "induced-fit", wherein each of the constituents (either the individual domains or the binding partners) change shape and can no longer be treated as rigid entities. Their conformations must be adjusted in concert with their association. As a result, the folding and docking processes will require extra sampling rounds, which greatly increases the amount of computation power required. Furthermore, this entire class of calculations generally are going to require a much larger number of loosely coupled calculations rather than a few, massively parallel calculations. Therefore, regardless of whether our or other folding/docking codes are used, the petascale computation environment we propose will be generally useful for virtually all protein simulations.

The identification of the protein sequences in a genome has transformed biological studies. But it is only the starting point. Amino acid sequence codes for structure, which often determines function. At the next level, protein association is integral to signaling networks which control and orchestrate cellular processes. Our long-term goal is the ability to take a set of genes identified in a biological process (e.g. amino acid synthesis under low nutrient conditions) and provide the structure and function of the proteins, as well as identify their interactions and signaling connections, including the positive and negative feedback one interaction has on another. Petascale computation is critical for this far-reaching goal.

What we have done to date. OOPS – the Open Protein Simulator – is a suite of C++ programs for the prediction of the structure of proteins with minimal use of information derived from sequence similarity or homology to other proteins. OOPS derives its speed and accuracy from the use of a "C_β" model, an accurate statistical potential, and a search strategy involving iterative fixing of structure in multiple "rounds" of folding. Since OOPS uses minimal homology information and a reduced representation, its success depends crucially on describing the "protein physics" correctly. Great effort has been devoted to the energy function, e.g., interactions are conditional on backbone geometry and the relative orientation of side chains. Last year, its accuracy exceeded available all-atom potentials, and it has been significantly improved since then. Our homology-free 2^o and 3^o structure predictions for small proteins rival or exceed homology-based methods with (expensive) explicit side chains, engendering optimism for continued progress. Furthermore, we now employ sequence homology for additional accuracy when appropriate.

OOPS is currently executed in parallel on modest computing clusters using an ad-hoc Python script that submits independent parallel jobs, analyzes results, and orchestrates the iteration of a parallel solution to a folding problem. Wilde's group has, over seven years, developed a general language and execution environment to express scientific workflows and execute them at higher degrees of parallelism on platforms including petascale precursor systems such as TeraGrid's Ranger and the ALCF BG/P. This environment comprises the Swift parallel scripting system [ZH+07,S08], the Falkon resource provisioner and scheduler[RZ+07, RZ+08] and the ZeptoOS operating system [RZ+08, Z08] developed by researchers at Argonne National Laboratory for use on petascale systems. Using current versions of the middleware components proposed here, we have run over 164M million jobs on a range of cluster and grid resources utilizing over 1.4M hours of CPU time – predominantly short-duration tasks not previously runable in such environments. 49 million of these test jobs were on the ALCF BG/P, running a set of diverse science applications including OOPS[FA08]. The success of these tests confirms the feasibility of the petascale OOPS scripting environment we describe below and motivates the program we propose.

What this project will accomplish. We propose to adapt OOPS to easily and effectively utilize *diverse* and important existing petascale computing resources (Blue Gene, Constellation, XT4) and future systems (Blue Waters), thus enabling OOPS to be applied to challenging folding targets and providing the requisite number of docking simulations (10^4 – 10^6) of challenging binding targets. OOPS's performance, accuracy, capabilities and usability will be enhanced and will benefit from live scientific usage and testing on sub-petascale computing platforms.

OOPS will be embedded into a flexible petascale computing framework that will enable top-level components to be easily executed by diverse scientific communities in a flexible manner with little or no additional programming (Section 3). The innovative computing approach we employ is characterized by a

flexible parallel scripting system that enables efficient execution on diverse petascale platforms, allowing users to harness more than one platform for a given run (i,e., a grid approach to resource utilization). This allows a great deal of flexibility in adapting existing programs to new requirements and algorithmic variations. In addition, the environment will be useable for many other applications, including other groups' folding and docking algorithms, and replica exchange simulations with multiple order parameters.

Our project plan details the specific deliverables, which includes: applications of petascale folding and docking, user interface and data sharing facilities, and large-scale datasets derived from testing, calibration and reference-data creation activities. The many education and outreach efforts and benefits of this effort are described below in Section 2. The computing approach is described in Section 3, followed by technical scientific details of the OOPS algorithms in Section 4, and comparison to other folding and computing approaches in Section 5. We conclude with a brief project plan sketch and a summary of results from prior relevant NSF funding.

2 Education and Outreach Efforts and Contribution

We describe here a multi-faceted plan to integrate research and education – at many levels – throughout the proposed project, and to create and maintain a diverse and inclusive collaboration.

Development of Human Resources During Prior and Proposed Support Period. During the last four years, (bio)chemistry training and research experience was provided for one postdoctoral researcher, three graduate students, and four undergraduate students. We have and will continue to welcome women and minorities in our research groups by reaching out to members of underrepresented groups via job fairs regularly held at UChicago for the entire Midwest region, and will use this opportunity to recruit the postdoctoral research positions supported by this grant. Freed and Sosnick have a strong record of undergraduate participation in research as witnessed by the four undergraduates working last summer (with three continuing through the academic year) on joint projects. One who graduated in 2007 had two first author papers and will be a coauthor on two others. The prototype testing of OOPS on the Argonne BG/P was done by Glen Hocky, a 4th year Chemistry/Math undergraduate working with Freed and Sosnick and collaborating with Wilde, is an example of the ability of this effort to provide promising young scientists with cutting-edge computational experiences. No funds are specifically budgeted for summer students since their stipends are arranged separately. Freed participates in a new Graduate Program in Biophysical Sciences (developed and chaired by Sosnick) that requires students to work on co-mentored, interdisciplinary projects. Freed is also part of a program to enable high school students to work in research labs.

Undergraduate and Graduate Education. Our project will involve graduate students in Biophysics, Computer Science, Chemistry and Biochemistry. The proposed OOPS framework will be available in an easy-to-use web interface that is well suited for teaching advanced undergraduate and early graduate students the power of emerging petascale systems for simulation and analysis.

A central component of the proposed research is the training of students and postdocs. The Freed group has produced 42 PhDs, has trained 29 postdocs and several undergraduates (who have coauthored papers), and has hosted numerous senior researchers. The group has included women and members of minority groups. We participate in several outreach programs (Dreyfus, REU, Beckman scholar, PCBio, etc.) for attracting undergrads to research, and group members have been very active in tutorial programs at schools with high proportions of disadvantaged minorities. The undergrads generally go on to graduate school, while the graduate students move to postdoc positions and thence (as our postdocs) on to positions in academia, national labs, industry, or software developers.

High School and Informal Education. The NSF-supported I2U2 project "Interactions in Understanding the Universe" [BG+06] on which Wilde is co-investigator, provides infrastructure and a pedagogical model for teaching the use of high-performance computing for discovery and collaboration in science, in both the high school classroom (with "e-Labs") and the informal science museum setting (i-Labs). E-Labs provide a web-based environment in which students can connect instrumentation and/or simulations, upload share and analyze datasets, and create and share reports, posters and findings, with other students around the US and in several other countries (Canada, Taiwan, Israel, India). As part of our effort we propose to create the framework for a new e-Lab on biomolecular simulation for high school use, and a visually compelling i-Lab for a museum demo on how proteins fold and interact. Our proposed collaboration environment (section 3) will provide much of the basis for the core of these e/i-Labs.

Workshop program. As part of growing the local UChicago user community for the OOPS petascale framework, we will make aggressive use of local seminars in the Computation Institute [CI08b] and James Franck Institute [JFI08] as an outreach vehicle for introductory demos and tutorials on the capabilities enables by the proposed framework. In project year 2, once the petascale OOPS framework is stable, we will begin a program of workshops to promote its use by students and researchers, with a focus on undergraduate and graduate students and early career scientists. In addition to numerous periodic short workshops (a few hours to 1-day), we will prepare and deliver 2 1-week summer workshops at UChicago open to the national and international chemistry communities as well as the numerous disciplines that require protein folding and recognition/docking as a principle tool. We will recruit participants from the many nearby universities in the midwest (to achieve a broad impact while containing travel costs) and through requested funds provide limited travel support for students throughout the US.

Investigators Freed, Sosnick and Wilde, colleagues, and participating students will all contribute to the workshops which will be held in meeting facilities of the Computation institute, where the recent OSG Midwest Grid School 2008 workshop was held. The workshops will cover a range of topics, including protein folding and biophysics, the use of OOPS, running OOPS at petascale, and general approaches to scientific petascale workflow that will be valuable to participants for general high performance scientific computing and data management. The workshops will be promoted to our large network of contacts among organizations advancing involvement in science and computing by people from under-represented groups, include HACU, SWE, Grace Hopper, and EPIC (Engaging People in Cyberinfrastructure).

3 Computing Approach for Scaling OOPS to Petascale Architectures

OOPS is currently a set of open source applications for fast simulation of protein folding and docking. It uses the C++ protein library "PL" [PL08] for representing, moving and performing energy calculations on protein structures, and also provides a set of useful analysis tools. A Monte Carlo simulated annealing (MCSA) simulation function, Mcsa() is the primary OOPS application. Given parameter files and a protein sequence it generates 2^o and 3^o structure descriptions and statistics about the final configuration (energy, etc). Throughout, we speak generally of both executable applications and library routines as "functions" and notate them as **FunctionName().**

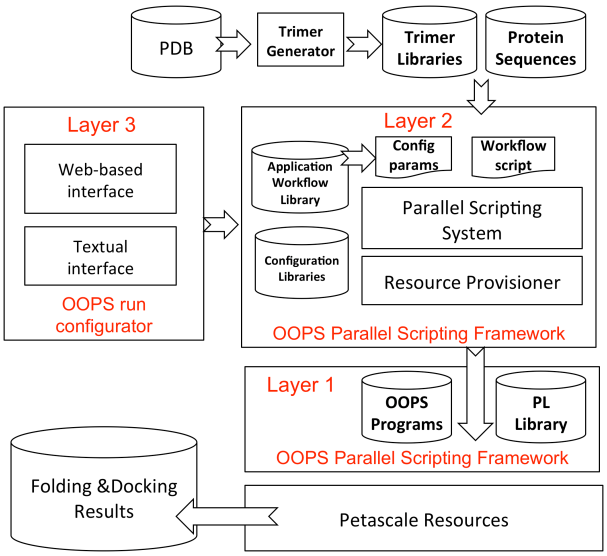

Figure 1: Petascale OOPS Framework

We will use a ***three-layer approach*** to escalate OOPS to petascale performance: **Layer 1** will parallelize the OOPS simulated annealing loop (inc. the energy calculator) using multi-core and MPI techniques, enabling a singe MCSA to efficiently use a large number of CPUs. **Layer 2** will exploit the innately parallel nature of the OOPS folding algorithm, which enables large numbers of MCSA simulation jobs to be done in parallel, both for multiple proteins, multiple parameter settings, and parallel "rounds". **Layer 3** will provide a petascale science gateway environment for the large-scale execution of OOPS for folding and docking. The architecture of this framework is shown in **Figure 1**. In the remainder of this section we first detail Layers 1 and 2, then define the projected speed and parallelism we seek, and conclude with Layer 3, the user environment.

Layer 1: Parallelizing the core algorithm. The core OOPS folding loop exposes significant parallelism for effective utilization of petascale resources. While the move-evaluate steps within MCSA jobs are at the moment serial, the operation that dominates its processing time is *calculating the energy* of each molecular configuration from the pair-wise atomic distances (**Figure 2**). We will devote significant attention to this function by parallelizing it first for a multi-threaded environment using OpenMP [CJ+07], and then at a larger scale using MPI. This operation can be readily parallelized: the contribution of each pair of atoms to the total energy can be computed in a highly parallel manner. We will use MPI as a

distribution network to partition this processing by batching subsets of the upper-triangular portion of the distance matrix based on the number of available “energy calculator” CPUs and then gather and sum the results. For a small, 76 amino protein, one eCalc() takes about .05 seconds of Pentium Q6600 CPU time and represents about 91% of the total cost of running Mcsa(). Generating the backbone move is about 7%; startup is 2%. eCalc() is inherently $O(N^2)$ where N is the number of atoms interacting, hence both the CPU and proportion of simulation time increases dramatically with sequence length (proteins are ~60-300 amino acids).

```
eCalc(config)
   eTable=ReadParameters
   for i 1 to nAtoms
      for j = i+1 to nAtoms
         d=dist(Ai, Aj)
         eIndex=(Ai,Aj)
   energy+=eTable(eIndex,Ai,Aj,d)
   return energy
```

Figure 2: OOPS Energy Calculation loop

Our proposed algorithm will perform the move and its energy calculation as a combined parallel operation, as follows. Since the moves are fast computations, each processor can maintain the full state of the protein, and do identical moves in parallel. At the start, each worker gets the same initial state, and is assigned a set of atom pairs Ai,Aj within the protein. After each move of the protein backbone, each worker calculates the energy for just the set of atom pairs that it was assigned. Each worker can maintain the set as a list of Ai,Aj pairs that it serially iterates over. Then each worker sends its energy subtotal to a master "annealer", who sums the constituent energies and decides whether or not to accept the move. The master responds to the workers with accept/reject, and then the workers do the next move. Each communication is very compact and can use fast collective primitives. Based on profile measurements cited above, we estimate that a speedup of at least 10, and likely 30 or higher, is possible with this algorithm, using 128 cores per Mcsa() job. (This will depend heavily on the ratio of parallel to serial work as the algorithm evolves. The rough 30X estimate assumes about 2% serial code and ability to utilize 100 cores for linear speedup of the parallel portion).

Layer 2: Parallel petascale scripting of Mcsa() rounds. OOPS workflows are innately suited for execution in a “loosely coupled” manner [RZ+08], both to parallelize the many independent jobs in the workflows, and to permit the science user to customize the analysis of intermediate and final results and to insert these analyses into the workflow.

The flow of a Fold() operation is shown in **Figure 3**. We define an OOPS ***workflow*** as a set of Fold() (or Dock) operations executed in parallel, possibly with intervening analysis. Each Fold or Dock consists of N ***rounds***. Each round is a set of ***jobs*** executed in parallel. (Rounds test a large sample of Mcsa() outcomes). Each job is one execution of the Mcsa() program. Each execution of the Mcsa program consists of a simulated annealing (SA) loop that iterates until it “cools” sufficiently. Each SA loop is a set of alternating move() and energy-check() operations, followed by acceptance or rejection of the move. The boundary between loosely coupled “jobs” and tightly coupled “function calls” is the Mcsa() application, which is executed as a job. Within Mcsa() we make function calls; outside Mcsa() we run programs which exchange files.

```
Fold(Protein p, int N)
   s = InitSecStr (from data file)
   round = 1
   while not converged and round < N
      foreach job j in 1..RoundSize
        model, SecStrList =
         Mcsa(p, temp, nSteps, moveSet, s )
      newS = analyze(model,SecStrList)
      converged = checkConvergence(newS, s)
      roundNum++
```

Figure 3: OOPS Folding loop

An example of analysis within a workflow is as follows. For each combination of input parameters Si (e.g. protein sequence, initial 2^o structure, starting annealing temperature, amino acid substitution matrix, etc.) we iterate multiple rounds of between 100 and 1000 runs of Mcsa(). After jobs 1-N in Si(round j) finish, we gather statistical data about that round, specifically on the average origins or assignments of the 2^o structures at each position in the sequence. Sometimes this analysis involves clustering of structures through various techniques. It is then determined whether or not to stop sampling angles from certain 2^o structure types at those positions, and jobs 1-N in Si(round j+1) are started with this information as a new input file. Additionally, at each of these analysis steps, various plots are created from the output data, including average 3D atomic contact maps (and movies), RMSD (3D position accuracy) versus energy plots, and 2^o structure prediction accuracy.

If the information being passed back at the end of round (k) for a set of parameters Si does not differ significantly from the information at the end of round (k-1) then the simulation for Si is considered converged. As each set Si finishes, overall summary data is generated. The best predictions from each round are compared and the 2^o structure fixing across rounds is visualized (Figure with ovals and arrows). Movies of the best trajectories are rendered. Comparison with other experimental and simulated data is performed, if that data is available. All of these data are compiled in tables and charts automatically (cf. the OOPS data in Table 2 below). This application profile motivates the following petascale execution framework.

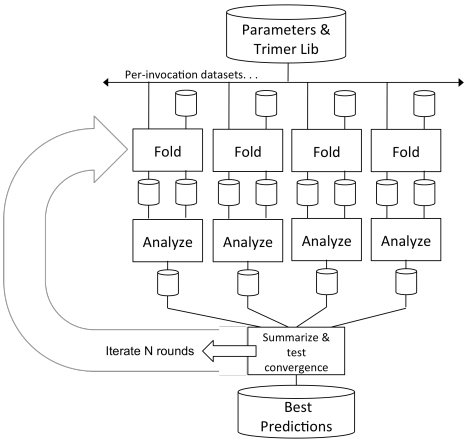

Figure 4: OOPS Folding Dataflow Pattern

Parallel Scripting.The need for a flexible scripting model to support this style of analysis in a flexible manner motivates our proposal to implement petascale OOPS workflows with the loosely-coupled parallel scripting paradigm of passing data via files between dependent tasks [RZ+08, ZE+08] (as shown in Figure 4). This is a natural approach to the task-independent (single-program, multiple data, or SPMD [SP08]) dataflow aspects of the workflow of a typical large-scale OOPS run. It has the advantage of being very fast, easier to implement and maintain, far more flexible, and dynamically adapts to workloads of widely varying task durations. Our work to date has proven this to be highly effective on petascale systems [RZ+08, ZE+08], challenging the conventional wisdom that petascale systems can only be efficiently utilized with a tightly coupled message-passing programming model. In contrast, we will employ a more pragmatic hybrid model, in which Layer 1 is tightly coupled and Layer 2 is loosely coupled: in essence, a loosely coupled scripted workflow of tightly coupled parallel jobs. This approach is well supported by the viewpoint of a highly regarded 2006 study of the future of parallel computation from UC Berkeley [AB+06, section 5].

The Swift parallel scripting system [S08, ZH+07] provides a useful method of programming large-scale computing resources in a uniform high-level manner. It demonstrates an approach that treats grids and petascale clusters as if they are complex chips, and takes a code-template approach to automating the mapping of scientific workflow, loosely coupled by files instead of tightly coupled by messages, to these large-scale systems. By treating entire scientific applications as if they were functions, Swift leverages the simplicity and uniformity of the functional model to solve complex data flow specification problems. Swift makes large-scale computing resources more accessible to both researchers and students, and lowers the barrier to large-scale data analysis. In this role, Swift provides a vitally needed easier-to-use "on-ramp" for large grids and evolving petascale systems. The Swift parallel scripting system makes this programming style possible with a highly scalable runtime system and a notation that enables the concise specification and reliable and efficient execution of large loosely coupled computations. With a few lines of Swift scripting code, one can specify computations involving extremely large numbers of files and tasks, and which execute efficiently and reliably on petascale computers.

Lightweight scheduling. Below Swift, we will integrate into the OOPS framework a petascale-tested parallel task execution framework called Falkon [RZ+07, RZ+08]. Falkon is a fast and light-weight task execution framework that combines dynamic resource provisioning and a streamlined task dispatcher, and has been proven to work with high efficiency on petascale platforms (BG/P) up to 164K cores [RZ+08, FA08]. Falkon enables loosely coupled applications to be efficiently executed on petascale systems that are otherwise restricted to efficiently handling only large, tightly-coupled applications such as MPI programs.

To handle the hierarchical architecture of the Blue Gene/P [IBM08] in which there are 640 I/O nodes, and 40K quad-core compute nodes, Falkon distributes its dispatcher function to the many I/O nodes. This architecture (described in Raicu et.al [RZ08]) has allowed us to scale to 164K processors with sufficient headroom for scaling to much larger systems, and to maintain the thousands of tasks per second needed to fully utilize a system with hundreds of thousands of CPUs.

Collective I/O. Loosely-coupled parallel scripting, while productive for the developer, imposes a high performance burden on large scale systems. We address this issue with a collective I/O model for file-based many-task computing [ZE+08] that we have prototyped on the BG/P and which enables efficient

distribution of input data files to computing nodes and gathering of output results from them. This approach broadcasts common input data, and uses efficient scatter/gather and caching techniques for input and output.

Resilience. When running loosely coupled applications via Swift and Falkon, the failure of a single node only affects the task(s) that were being executed by the failed node at the time of the failure. I/O node failures only affect their respective processor sets (groups of 256 processors). Falkon can suspend offending nodes with high task failure rates. Swift maintains persistent state that allows it to restart a parallel application script from the point of failure, re-executing only uncompleted tasks and eliminating the need for explicit checkpointing.

Application packaging and platform support**.** In the scope of this proposed 4-year effort, we will first focus on packaging the OOPS framework for standalone execution on these platforms: 1) The TeraGrid system "Ranger" at TACC and future TeraGrid petascale expansions; 2) The Argonne LCF BG/P system "Intrepid"; 3) The ORNL LCF Cray XT4 system; 4) small and TeraScale Linux clusters located throughout the TeraGrid and Open Science Grid, including the CI Petascale Active Data Server (PADS) system and 5) a small set of popular desktop Linux workstation environments. Our work to date has shown both the OOPS code itself as well as the Falkon and Swift components to be highly portable and platform neutral. The OOPS framework will include the core components for the runtime environment and a Java-based framework for the workflow submission (client) environment and lightweight task scheduler.

The framework will be distributed with a built-in set of top-level scripts to perform the most popular variations of folding and docking simulation tasks. A rich set of analysis scripts will be provided. Both the simulation and analysis scripts will serve as exemplars for users to customize their own scripts. While we will package the Petascale OOPS framework for simple "one command" automated compilation and installation on a modest set of the most needed runtime environments, we will also collaborate with the major petascale environments to maintain shared copies of the OOPS framework for their users.

Projected speed and effective use of petascale parallelism. Having described the algorithms for parallelizing Layers 1 and 2, we can now estimate the speeds we expect to achieve on our target petascale platforms. Several estimates below are compared to measurements of a "stock" processor (one core of an Intel Q6600), running an average protein (1UBQ.pdb). Projections are based on ALCF BG/P "Intrepid", TeraGrid Constellation "Ranger" and press speculation on Blue Waters [BW08]. As described above, for folding runs, we do P proteins in parallel (constituting a Round). At Layer 2, each simulation of a fold takes about 12 minutes (almost entirely CPU time) on the stock processor and roughly 60 minutes on a BG/P core.

Table 1: Projected Petascale Speedups

Operation *#proteins #cores x 1K, #cores for eCalc()*			**Time (hours)** *BG/P, Constellation, BlueWaters*		
#prot	KCore	#eC	BG	Cn	BW
1	10	1	5.3	1.1	0.8
1	40	4	1.3	0.3	0.2
1	60	16	0.9	0.2	0.1
1	164	16	0.3		0.1
1	250	16	0.2		0.0
10	60	16	8.9	1.8	1.3
10	60	160	8.9	1.8	1.3
10	164	160	3.3		0.5
100	250	16	21.3		3.2

The terminology and timings of the nested iterations that comprise a single protein folding or single 2-protein docking simulation are*: 1 fold = 10 rounds; 1 round = 100 to 1,000 Mcsa() invocations: 10^3 to 10^5 jobs; 1 dock = 10,000 to 1,000,000 Mcsa() invocations: 10^4 to 10^6 jobs; 1 Mcsa() invocation = ~15,000 to ~100,000 move()/eCalc() steps.*

The degree of parallelism inherent in OOPS folding is *# proteins x roundSize x corePerAnnealer.* At the low end, roundSize = 100 and coresPerAnnealer = 1. At the high end, these can both be 1000, utilizing 1M cores for a single Fold() job. To effectively use, e.g., the ALCF BG/P for one protein, we can use roundsize = 1,000 and corePerAnnealer = 160. This would cut the 60 minute duration of a Mcsa() job to 2 minutes; a 10-round fold would take 20 minutes.

Folding 10 proteins with a fold size of 10 rounds and a roundSize of 1,000 can utilize 100,000 CPUs working in parallel with no data dependencies and hence near-perfect scaling. If we devote between 4 and 16 cores to each Mcsa() function for the parallel computation of the energy of each configuration, we can effectively utilize 40,000 to 160,000 compute cores for this task, or 4,000 to 16,000 cores per protein. Realistically, even on petascale systems with 50K to 300K cores, most user jobs will run with allocations far less than the full system. This scale fits well for today's usage, and can expand to greater utilization, even for single proteins, as the parallelization of the energy computation increases.

While folding presents excellent opportunities for petascale parallelism, predicting protein docked conformations is expected to require far more simulation and can effectively utilize an entire BG/P system to dock a single protein pair. Docking will require $O(10^{4}$-$10^{6})$ trajectory calculations – each one an *independent* Mcsa() call that can be given its own core (and, when the energy function is parallelized, its own set of cores). Thus docking will immediately generate sufficient independent parallel jobs to utilize a petascale system and is an excellent candidate for utilizing a million-core system in the coming decade.

Based on this analysis, our tentative targets for OOPS execution (comparing the CI TeraPort cluster, ALCF BG/P, TeraGrid Ranger Constellation CN as a metric) are shown in **Table 1**. By comparison, folding 1 protein on the 256 core TeraPort cluster at UChicago (Opteron) is approximately a 40 hour computation. We caution that all of these numbers are extremely speculative estimates, based on many assumptions (especially for Blue Waters). 1 Mcsa() of 0.2 hrs on a stock 2.6GHz processor is estimated here as 1 hr on BG/P, .2 on Constellation, 0.15 hrs on blue waters assuming 4GHz cores and no superscalar behavior. This is a conservative estimate if we actually scale with clock speed. **While speculative, these performance goals represent a transformative change in the ways that protein folding and docking can be applied as a tool across diverse sub-disciplines.**

Overall confidence in the scalability of our proposed framework is supported by recently published tests[RZ08,ZE+08,FA08]. We have since January 2008 been testing, measuring and improving the components of the framework proposed here, focusing on the petascale ALCF BG/P system. In this time we have run 164M jobs through Falkon, across grid clusters, a 5800 core SiCortex, and BG/P systems, with an average task length of 31 seconds. We have utilized 1.2M hours of BG/P time, running 49M Falkon tasks, and 105K CPU hours of OOPS on the BG/P with 102K tasks averaging 3700 seconds. Our results indicate excellent utilization, using Falkon at scales up to the full BG/P (163,840 cores).

As an example of the scalability of this approach on an application very similar in profile to OOPS, we tested a DOCK5 workload that consisted of 934,803 jobs, which we ran on 116K CPU cores in 2.01 hours [RZ+08]. The per-task execution time was quite varied, with a minimum of 1 second, a maximum of 5030 seconds, and a mean of 713±560 seconds. The two-hour run had a sustained utilization of 99.6% (first 5700 seconds of experiment) and an overall utilization of 78% (due to workload dropoff at the tail end of the experiment). OOPS jobs are much more uniform in run time and will not exhibit this "tail effect". These preliminary results support our belief in the value and feasibility of loosely-coupled petascale computation.

We have been running OOPS itself on the ALCF BG/P system since Aug. 2008 at levels up to 64K cores with near-perfect processor efficiency. Our results from these tests indicate that our design yields an easy-to-run system, delivers linear scaling consistent with the results obtained at much larger levels for DOCK and synthetic benchmarks, and shows the feasibility of our Layer-2 architecture.

User interfaces to the petascale OOPS framework. Layer 3, the user interface layer, will provide workflow hosts, web-based OOPS request interfaces, and data, metadata and provenance catalogs. An OOPS "run configurator" mechanism packaged for use both from a web-form-based interface as well as via a simple textual command specification will enable users to specify OOPS runs with no programming. The web interface will be runable locally by any user or community as a service of the OOPS workflow framework. Users seeking more control over customizing their runs can write their own Swift scripts based on the models provide for the supplied library of scripted applications.

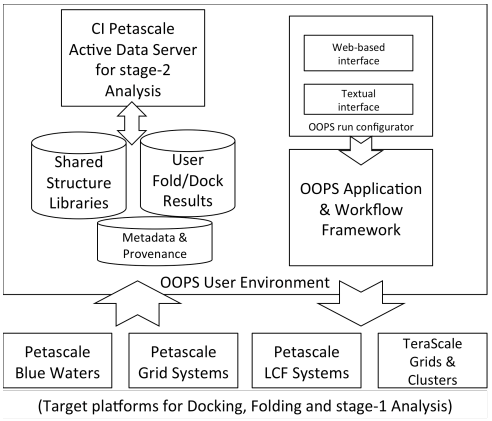

Figure 5: OOPS User Environment

As a natural by-product of our OOPS petascale development effort, we will create an environment, hosted at the UChicago Computation Institute (on its Petascale Active Data Store and TeraPort systems) to serve as a protein science gateway to the large-scale execution of OOPS for folding and docking. This will provide workflow hosts, web-based OOPS request interfaces, and data, metadata and provenance catalogs. As needed, based on security considerations (i.e, the ability to securely host web interfaces and access external file repositories), we will either replicate this framework within, or, preferably, integrate this environment with DOE-based LCF BG/P and XT4 systems. While this environment will be created initially to serve the needs of the local UChicago / Computation

Institute research community, we are eager to open it up for use by scientists across the US and around the world, to the extent that we can support this.

The collaboration environment will enable scientists in diverse disciplines to catalog and share input and output datasets, and to compare results from variations on parameter sets and algorithms. The collaboration environment will leverage the CI's new NSF-supported Petascale Active Data Server (PADS) – a .5PB storage system integrated with a 384-node cluster, with another .2PB of storage and ample RAM on the cluster nodes [CI08, H08] (NSF grant OCI-0821678). This will be an ideal facility for "stage-2" analysis (where stage-1 analysis is done on the target petascale systems themselves as part of the OOPS workflow, as described above).

Among the many advantages of our script-based approach to the "outer" parallelization of OOPS is that we get automated data provenance tracking [CF+08, ZWF06] from the workflow execution engine, as well as the ability to leverage multiple petascale execution resources within a single workflow.

While development of new visualization software is outside the scope of the proposed effort, existing visualization tools can be readily integrated into the framework (via analysis scripts) and can leverage many existing tools to visualize protein conformations, 2^o structure, and statistics. Tools such as R, Octave, and MatLab can be readily integrated into analysis scripts (as many Swift users do today). Such analysis scripts can utilize the same parallel scripting language as the OOPS run-time framework, and can run both on the target petascale systems as well as the backend "stage-2 analysis" environments such as PADS, clusters and workstations.

4 Science Approach – Protein Simulation Methods and Algorithms

In this section we describe the computational chemistry approach to folding and docking that is implemented by the OOPS framework. The level of detail is intended for those familiar with the field.

Background. While there has been great progress with predicting 2^o and 3^o structure using homology-based methods, the goal of predicting structure from sequence alone remains elusive. Furthermore, the accuracy of 2^o structure prediction methods has reached a plateau despite their crucial role as input to most methods for predicting 3^o structure. A major barrier in structure prediction has been the reliance on known structures with significant sequence similarity, or homology, to the target sequence. Many successful prediction methods begin with an alignment of homologous sequences (e.g., PSI-BLAST [ASF97]). Because the accuracy of the 2^o structure prediction degrades for target sequences with low homology, the accuracy of 3^o structure predictions similarly diminishes for both template-based modeling and simulations using homologous protein fragments. The reliance on homology precludes identifying the underlying physiochemical principles that govern protein folding, including determining the minimal essential information and model of protein structure that are required for accurate structure prediction.

A deficiency of many 2^o structure prediction methods[JDT99,PP+02] is the failure to explicitly include 3^o context. Context dependence can override local biases[MK96,KD05,AH+07], and its neglect has limited 2^o structure accuracy to about 80% for decades[DZ07]. Previous attempts to improve prediction by including 3^o structure predictions achieve rather limited success [MB03].

We have devised a strategy in which 2^o structure prediction is integral to and a consequence of the folding process. Consequently, our strategy may share some benefits that real proteins gain by folding along a robust and efficient pathway. While others have integrated the determination of 2^o and 3^o structure[LK+05,YC+07] with an iterative fixing of structure[SR95,SF+04,OW+07], our approach differs by not using any exogenous 2^o structure prediction or homology, and our model lacks side chain degrees of freedom, which greatly reduces computation time, and the entire chain interacts at all stages. Our results suggest that models that do not include explicit side chains or utilize homology can be as accurate while requiring far less computing time. In addition, information about folding pathways can be extracted from simulations (DeBartolo, Freed, Sosnick, submitted).

Integration of 2^o and 3^o structure prediction. Our ItFix algorithm focuses on three fundamental protein properties, the sequence dependent torsional preferences of the polypeptide backbone, its hydrogen bonding requirements, and the different chemical properties and packing preferences of the twenty amino acid side chains. As each factor strongly influences the other two, a major challenge lies in simultaneously incorporating all three factors into a folding algorithm. The model retains the backbone heavy atoms and the side chain C_β atoms [CJ+06,ES06,SS+06,FJ+08], so the backbone dihedral

angles ϕ,ψ are the only degrees of freedom in this reduced representation. Hence, only *2N* parameters are needed to describe a chain of *N* residues.

Iterative fixing and trimer selection. A critical aspect of the algorithm is the selection of trimers from an increasingly refined trimer library, similar in spirit to earlier studies [SR95,SF+04,OW+07]. During the initial round of the simulations, trimer selection is conditional only on the amino acid identity of the three residues (Fig. 6). Trimer selection in subsequent rounds depends on the 2^o structure type at each position that is identified from the previous round by the prescriptions described below. The specification of 2^o structure is enabled because the three residues of each trimer in the trimer library are labeled by their 2^o structure assignments in the PDB structure in which they originate using the Dictionary of Protein Secondary Structure (DSSP) definition. The frequencies of occurrence for each originating 2^o structure type, H(elix), E(xtended), or C(oil), are calculated from the last inserted trimer at each position in the $O(10^2)$ final structures emerging from each round of folding. Following Sherlock Holmes' deductive strategy "Eliminate all other factors, and the one which remains must be the truth."[DAC90], if the frequency of occurrence for a particular 2^o structure type falls below a 5-10% threshold at a given position, any trimer with that 2^o structure at this position is removed from the trimer library used in subsequent folding rounds. The process continues until no additional positions can be further restricted. After the last round, the best structures are identified.

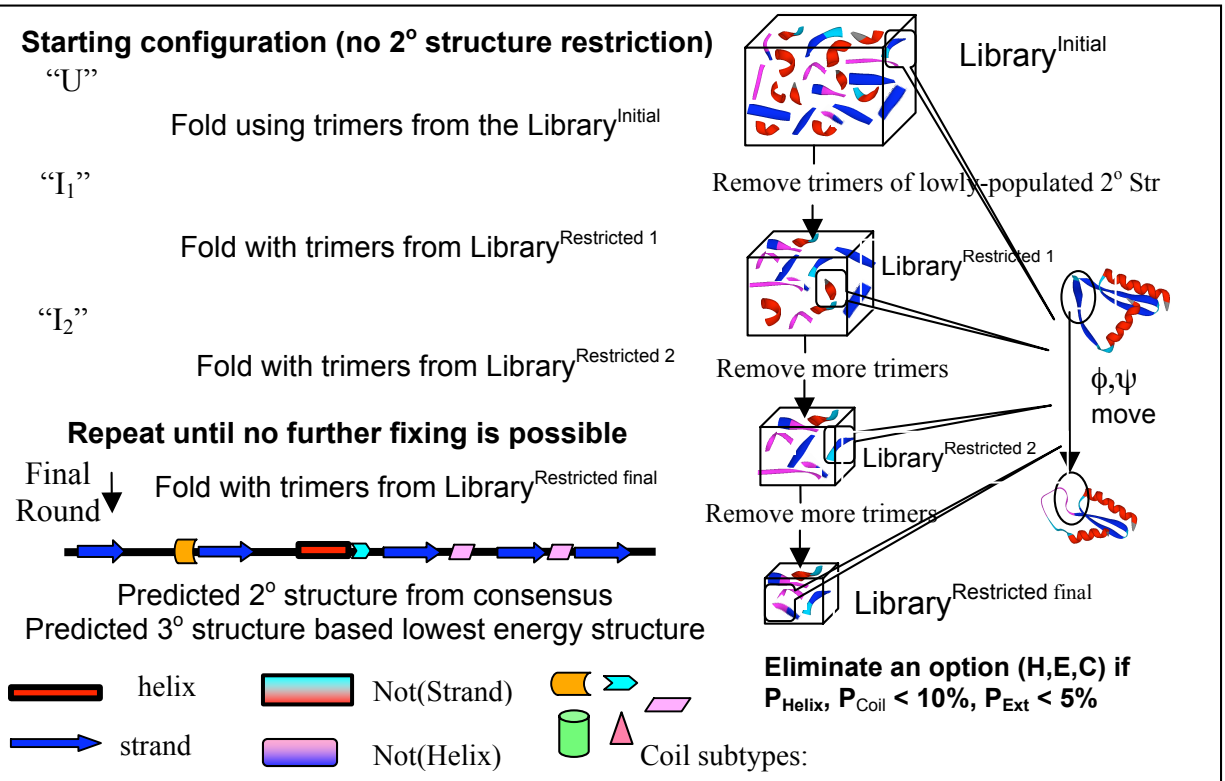

Fig. 6 ItFix Algorithm. Trimer-based moves are obtained from a library that is increasingly restricted during the multiple rounds. In the initial round, all sequence-compatible trimers are allowed. In subsequent rounds, trimers are removed if their SecStr type is observed at low frequency at the end of the previous round of folding. The starting configuration (black line) is built from the insertion of a trimer at each position based only on its amino acid identity. The frequencies of SecStr are calculated and used to restrict trimer selection in future rounds For some regions, H or E are eliminated as an option in early rounds (two-color ovals), eventually becoming fixed to a single SecStr type (solid). At the end of the final round, the lowest energy structure is selected as the predicted structure, while the predicted SecStr is an outcome of the iterative fixing process during multiple rounds.

Our MCSA algorithm is designed to mimic rather than replicate the true folding pathway. Each round in our iterative fixing (ItFix) process begins from a configuration devoid of any 3^o structure. The backbone

geometry is simulated by replacing one pair of ϕ,ψ dihedral angles at a randomly chosen position with those from the equivalent position in a ϕ,ψ trimer selected from a library. In principle, these angles could be obtained from all-atom simulations for tripeptides, but current methods are much less reliable [ZS+03]. The starting chain is built using trimers initially specified solely by the amino acid sequence. However, the library becomes increasingly conditional on 2^o structure type as the rounds proceed. Each round of the iterative folding consists of $O(10^2)$ individual folding trajectories. Each trajectory involves a global search guided by single ϕ,ψ insertion moves whose acceptance is governed by a Metropolis criterion with a statistical potential for a scoring function. The trajectory ends when the collapsed structure cannot undergo additional moves. The final product is a folding-enhanced 2^o structure prediction that emerges simultaneously with an ensemble of 3^o structures.

Retaining lost side chain information. Retention of the side chain information lost by the use of the C_β-level representation poses an extreme challenge. Central to this goal is our ϕ/ψ dihedral angle sampling procedure that is conditional on both the chemical identity and the increasingly refined 2^o structure specificity for each position and those of the neighboring residues. The backbone dihedral angles are strongly correlated with the side chain rotomer angles and both the neighboring residues' side chain identities and conformations [JC+05,JC+05b,CJ+06]. Hence, even without explicitly depicting the side chains, much of their influence is retained by choosing ϕ,ψ values using our trimer selection strategy.

Given this retention of the interplay of side chain/backbone interactions, the other elements of our algorithm focus on optimizing 3^o interactions and to a lesser degree, backbone hydrogen bonding. Tertiary interaction energies are obtained from the statistical potential "DOPE-C_β"[CJ+06,FJ+07] derived from an all-atom pairwise potential [ES+06,SS+06] that uses a novel reference state and distinguishes the backbone atoms according to amino acid type. Our version removes from the all-atom potential contributions involving side chain atoms beyond the C_β atom, as appropriate to the reduced C_β chain representation. To eliminate bias towards specific 2^o structure types, the attractive potential is removed between atoms in contiguous units of 2^o structure. The repulsive portion of this term is retained between all atom pairs to prevent steric overlap. In addition, interactions are conditional on backbone geometry and the relative orientation of the C_α-C_β bonds of the two interacting side chains. The latter feature is particularly helpful in setting up the over-all chain topology so that collapse generates native-like structures. Beyond the prescription used to eliminate a 2^o structure option in the trimer library, the only adjustable parameters are the four relative weights of the components of the statistical potential. Due to computational costs, the weights are derived semi-empirically but will be optimized with petascale facilities. Our C_b energy calculation is ~four-fold faster than any all atom potential, and is able to take into account the most important interactions in the folding process. The energy calculation is the most expensive part of any molecular simulation code, so we foresee a great interest in incorporating this function in other researchers' work.

Performance. Our methods have been benchmarks on two diverse sets of proteins. The first set of targets originates from a previous study that integrates 2^o and 3^o structure prediction [MB03]. The first set contains proteins with fourteen diverse folds and relatively low sequence homology. Our accuracy of predicting the three major 2^o structure types, termed "Q3 level", generally surpasses that from the prediction servers SSPRO[PP+02] and PSIPRED[JDT99] and the previous study[MB03], with some exceptions. A second set of targets originates from a study focusing on improving 3^o structure prediction using extensive homology and side chain refinement. The high homology of these sequences is responsible for the 2^o structure prediction accuracy to exceed 80% for both the SSPRO and PSIPRED servers, well above the large scale benchmarking for these methods. Nevertheless, our iterative fixing protocol achieves comparable accuracy without invoking any homology information (Table 2, Fig. 7). Furthermore, we also predict all eight types of 2^o structure (aka Q8 level) by subdividing coil into the six of the DSSP-defined subtypes. This ability also is available using SSPRO, but it is less accurate for most targets.

The ItFix algorithm describes both protein sets with comparable accuracy for α, α/β, and β proteins (Fig. 7, Table 2). These predictions are comparable in accuracy with the highly successful Rosetta fragment-based insertion algorithm, as implemented in the papers from which the test sets are obtained[MB03,BM+05]. Our structures are more comparable in quality for the low homology set than the high homology set, as expected given that Rosetta extensively uses homology to obtain the increased accuracy. In addition, the Rosetta algorithm includes extensive side chain refinement and requires over

an order of magnitude more computation time[BM+05] than our algorithm which does not include side chains. We applied the ItFix algorithm in the CASP8 experiment. A notable success was the prediction of T0482 (2k4v.pdb), a 120 aa α/β protein (Fig. 7).

Table 2: Comparison with existing prediction methods for proteins taken from [BM+05]

Protein			Secondary Structure Prediction (% accuracy)			3° Structure Prediction-(C_a-RMSD, Å)	
PDB ID	Length	Fold	ItFix Q3 (Q8)	SSPRO Q3 (Q8)	PSIPRED Q3	ItFix	Baker et al. (best[1])
1af7	69	α	96 (87)	86 (81)	90	2.6	10.4
1b72A	50	α	84 (80)	68 (72)	84	2.4	1.1
1csp	67	β	87 (75)	75 (67)	88	6.5	4.7
1di2	68	αβ	88 (84)	74 (75)	97	6.1	2.6
1mky	77	αβ	88 (75)	87 (71)	90	5.2	6.3
1o2Fb	77	αβ	83 (70)	79 (66)	75	5.8	10.1
1r69	61	α	93 (89)	84 (72)	92	3.9	1.2
1shfA	59	β	85 (63)	85 (69)	80	6.7	10.8
1tif	57	αβ	89 (79)	86 (70)	93	4.6	4.1
1tig	86	αβ	81 (67)	69 (67)	74	5.3	3.5
1ubq	73	αβ	90 (69)	88 (67)	90	3.7	1.0

[1]Values shown are taken from ref. [BM+05], Table 1, Column 7 "Best C_a-RMSD of the centers of the largest five clusters from the low-energy models from Round 1".

Future Studies. We propose to continue optimizing the algorithm, focusing on better methods to include homology-based information and ensure a topologically correct collapse. The energy function will be improved by adding a penalty for burial of unsatisfied H-bond donors and acceptors and by renormalizing energies as a function of burial to account for multi-body effects and cooperativity. Success in the template-free prediction of pathways and structure using a C_β-level model will imply that we have identified many of the important principles that govern the folding process and will be well-positioned to determine the structure of proteins with no known template.

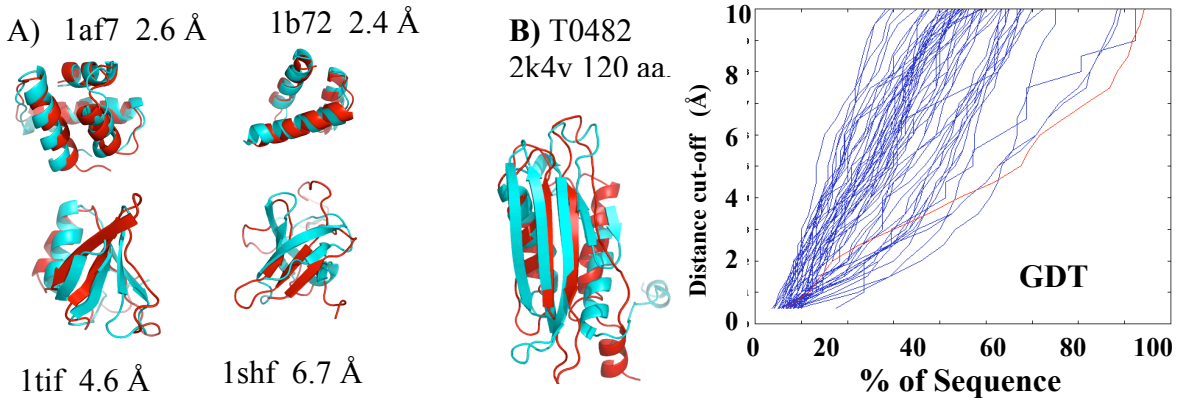

Fig. 7. ItFix 3° structure prediction. A) Comparison between homology-free ItFix predicted (lowest energy among compact structures) and native structures w/ PDB code and C_α-RMSD. **B)** ItFix prediction for CASP8 target T0482 (for this prediction only, the trimer library was enhanced using trimers from homologous sequences, but no structural fragments from homologous *structures* were used). RMSD calculation ignores the isolated helix at the terminus. Global Distance Test (GDT value is the percentage of the sequence within a distance of the native structure. This distance is the y-value on the plot, e.g., for the ItFix prediction, 50% and 92% of the residues are predicted to within 4.3 and 10 Å of the native structure, respectively. The ItFix prediction (red, with human intervention) compares extremely favorably to top server predictions (other human predictions are currently unavailable).

Utilizing the known sequence universe. Although our algorithm can function admirably without homology-based information, we have found that utilizing the vast amount of known sequences produces superior results. Our original sampling is restricted to trimers having the same amino acid sequence as

the parent. To increase the number of trimers and take advantage of homology information, the library is enriched using data from sequence-homologous stretches of 20+ residues as determined using PSI-BLAST. The aligned regions identify other possible amino acids at each position. These other possibilities are combined with the possibilities at the other two positions to create new trimer combinations. Though the new set of trimers are in general better than the original trimers even in abnormal situations, they provide increased diversity that often is required to provide a native set of angles at a few critical positions in the sequence. This methodology was used during the blind protein structure prediction challenge CASP (Fig. 7) and further studies are under way.

Adapting the OOPS Method to Docking. The conversion of the OOPS folding algorithm to perform docking calculations is simple. The two partners are tethered together with a virtual poly-glycine tether. This tether undergoes the normal "Type 1" moves that involve inserting a single ϕ,ψ angle pair, and the standard energy is calculated, although only between the two complexes (just inter-protein, ignoring the tether). These studies will take advantage of the accuracy and speed of our C_β-level energy calculation. After predicting a possible set of docked conformations (e.g. 10^3) using Type 1 moves, the conformations will then be subjected to *Type 2* moves wherein the backbone is relaxed.

Although we are new to the docking field, we effectively have already been conducting docking studies during the CASP8 experiment: we often predicted two subdomains which were then docked to each other by folding the connecting region.

Refinement Move Type 2. Although useful for folding an extended chain, the ϕ,ψ insertion move is inadequate for backbone relaxation within the condensed state. What is required for structural refinement either in the context of folding or docking is a move set that allows residues to execute moves with large changes in backbone dihedral angles *while leaving the flanking regions largely unchanged*. The move of most relevance to the present work is a crankshaft motion wherein the ψ_{i-1} dihedral angle of the preceding residue is rotated in an equal but opposite direction as the ϕ_i of the residue of interest: $(\psi_{i-1},\phi_i) \rightarrow (\psi_{i-1}+\delta,\phi_i-\delta)$. This move is used infrequently in structural refinement and then apparently only for minor angular changes for a single peptide group, $\Delta\phi$, $\Delta\psi \leq 3°$[RS+04]. Our refinement method, in contrast, allows arbitrarily large δ while maintaining optimal backbone geometry.

A large scale crankshaft motion can locally improve the protein's backbone geometry while maintaining the RMSD of the entire backbone within ~ 1 Å *even with backbone rotations of 180°*. Although the single crankshaft move is extremely useful, we have developed a move that combines the best predictive capabilities of fragment insertion methods with the fine-scale refinement possible with molecular mechanics methods. This move is a concerted "double-crank" move of 4 consecutive dihedral angles: $[(\text{--},\psi_{\iota-1}), (\phi_\iota,\psi_\iota), (\phi_{\iota+1},\text{--})] \rightarrow [(\text{--},\psi_{\iota-1}-\delta), (\phi_\iota+\delta,\psi_\iota+\Delta), (\phi_{\iota+1}-\Delta,\text{--})]$. The move is particularly powerful because it alters both the ϕ,ψ angles of the center residue, unlike the single-crank move, which only alters one angle. Because the center residue's ϕ,ψ values can both be changed by an arbitrary amount, the double-crank move can be combined with our extensive knowledge of dihedral angle preferences [JH+05,JH+05b,CJ+06] wherein the angles ϕ_ι,ψ_ι are selected according to the amino acid identity and its nearest neighbors' sequence and conformation. The double-crank move, when combined with PDB-based ϕ,ψ frequencies, significantly improves the backbone stereochemistry without disrupting the fold. We have used it to refine crystal structures, improving published R_{free} values. Because these moves are conducted without side-chains, the calculations are fast and the structure are much less likely to be trapped in local minima. Folding of multi-domain also will profit form Type 2 moves. We stress that our utilization of homology is at the sequence level, and there are ~ $3x10^7$ sequences, whereas normal implementation of homology uses structures and there are only ~10^4 unique structures in the PDB.

5 Comparison to state of the art, and other work and approaches

Protein simulation approach. Most folding algorithms rely heavily on known structures or fragments (homology- or template-based) and can be extremely successful. Their success, however, rapidly decreases as the amount of known information decreases[SRF08]. Nearly uniquely, our folding algorithm focuses on incorporating principles of protein chemistry and on mimicking a folding pathway to restrict the search space. We rely minimally on homology, and even then, only on sequence but not structural homology. Nevertheless, the homology-free predictions from OOPS rival or exceed homology-based methods for small proteins which use explicit side chains, engendering optimism of continuing progress.

Our algorithm is able to predict the structures of two sets of proteins with comparable accuracy for α, α/β, and β proteins (Table 2, only one set shown). These predictions are comparable in accuracy to the highly successful Rosetta fragment-based insertion algorithm, as implemented in the papers from which the test sets are obtained[MB-3,BM+05].

Computing approach. Until recently, most of the applications run on emerging petascale computing systems, such as IBM's Blue Gene/P [IBM08] were "tightly-coupled", and were commonly implemented by using the Message Passing Interface (MPI) to achieve the needed inter-process communication. In contrast we enable the use of these resources for task-parallel applications, which are linked into useful workflows through the looser task-coupling model of passing data via files between dependent tasks. This larger class of task-parallel applications was typically precluded from leveraging the increasing power of modern parallel systems due to the lack of efficient support in those systems for the "scripting" programming model [O98]. We claim (and have proven) that many-task computing (MTC) [RZ08] applications *can* be executed efficiently on today's supercomputers. Our previous work [RZ08,ZE08] provided empirical evidence to prove our claim by testing and measuring two systems, Swift [ZH+07] and Falkon [RZ07] and various applications such as BLAST, DOCK [M+06], MARS [H06], OOPS (discussed above), and CNARI [C+08]. We point out various factors that motivate the support of MTC applications on petascale HPC systems. The I/O subsystem of petascale systems offers unique capabilities needed by MTC applications. [RZ08,ZE08] The cost to manage and run on petascale systems like the Blue Gene/P is less than that of conventional clusters or Grids. [AU08] Large-scale systems inevitably have utilization issues, and hence it is desirable to have a community of users who can leverage traditional back-filling strategies to run loosely coupled applications on idle portions of petascale systems. Perhaps most important, some applications are so demanding that only petascale systems have enough compute power to make certain problems tractable in a reasonable timeframe, or to leverage new opportunities.

Others have come to the same conclusion we have, identifying that HPC machines can be efficiently used for non-traditional workloads, and have begun work on building the necessary middleware to support these new kinds of applications. IBM has proposed in their internal project Kittyhawk [AUW08] that the Blue Gene/P can be used to run non-traditional workloads, such as those found in the general Internet, which are by definition part of a loosely coupled system. Cope et al. [CO+07] aimed at integrating their solution in the Cobalt [D05] scheduling system (as opposed to bringing in another system such as Falkon); their implementation was on the Blue Gene/L using the HTC-mode [IBM08b] support in Cobalt. Peter's et al. from IBM also did a similar study using the HTC-mode native support in Cobalt [PK+08]. Condor [TTL05] is also in the process of supporting high throughput computing on the IBM Blue Gene, but there has not been any formal citable literature other than presentation slides outlining their efforts. Our previous work [RZ+08] compares and contrasts the performance between our proposed system using Falkon on the Blue Gene/P and the results presented by Cope at al. [CO+07] and Peters et al. [PK+08], and found at least one order of magnitude better performance, and several orders of magnitude better scalability. This improved performance and scalability of the middle-ware can translate into direct improvements in scalability and performance for applications, with finer grained task parallelism and reduced end-to-end application execution times.

6 Project plan

This proposed effort will deliver: 1) A protein folding application that minimally annotates the 2^o structure (and local 3^o structure) of low-homology protein sequences. 2) A protein-protein docking tool which includes backbone refinement, to determine the most-likely pairings within groups of proteins. Our efforts will include the rigorous testing and performance/accuracy tuning of these applications. 3) A supported open source version of OOPS enabled for and tested on specific major available petascale systems and related support architectures (workstation to terascale grids). 4) A library of scripts for executing these applications in diverse ways and for flexibly adapting them to new uses. 5) The OOPS parallel energy function as a readily integrated module for other parallel and petascale applications with similar needs. 6) A general petascale computing framework and application toolkit useful for multi-order parameter replica exchange and other similar problems. 7) Initial limited support for a community of users, including a "science gateway" for data sharing and web-based invocation of the tools. 8) Published libraries of large-scale runs on proteins with limited homology, and other datasets of value to the broader community.

A high-level summary of development milestones is: Year 1 (2009-10) Support for BG/P (ALCF) and Sun Constellation (Ranger); initial web interface; initial parallel multicore energy function with OpenMP (eCalc,

multicore); Scaling to 500K-task workflows; Integration of Collective I/O ; prototype docking/recognition. Year 2 (2010-11): Parallel MPI eCalc; Extend to ORNL Cray XT4; scaling to 1M task workflows; Intelligent combined multicore/MPI eCalc; PADS-based user environment; Scaling to 500K-task workflows; production docking/recognition. Year 3 (2011-12): ;Prototypes on Blue Waters; Create a web service; I2U2 e-Lab prototypes; i-Lab for MSI and elsewhere?; Scaling to 10M task workflows; Data/metadata and provenance catalogs in user environment. Year 4 (2012-13): Maintenance, testing, tuning; I2U2 e-Lab and i-Lab interfaces; enhancements of science gateway environment and end-user documentation. Transition to open source development and support.

Our staff plan is described in the budget justification. Our proposed approach of using a parallel scripting method greatly reduces the programming staff-hours needed for the project, and the fact that we have successfully demonstrated the technique and middleware stack for OOPS makes us confident that the staff resources we request are adequate to deliver what we propose here. For computing resources, we currently possess for various development efforts 200K (scaled) CPU hours of Ranger time and are conducting our ALCF work using a director's discretionary allocation on the 164K-core ALCF BG/P which will continue through the duration of this project. To supplement these allocations we will apply for extended allocations on TeraGrid Ranger and on both LCF systems via INCITE proposals.

Within our local development environment, OOPS, Swift and Falkon are already packaged for easy end-user installation, and all three are maintained under version management using "Subversion" [SFP08]. Issue tracking is already in place for Swift and Falkon, and will be added for OOPS (using the Bugzilla system [BUG08]). For continuous build-and-test, Swift already makes use of the NSF NMI "Build and Test" facility "Metronome" [NMI08], which on a regular basis performs and automated build of an application on a set of supported target environments, and executes an automated test suite on each. All of these industry-standard approaches to methodical development will ensure that the petascale OOPS framework can reliably serve the needs of a large, international user community.

7 Results from prior NSF support

Research supported, in part, by NSF Grant CHE-0416017, 7/1/04 – 6/30/08 to Freed, "Self-assembly in associating fluids: Transition coupling and branching" (total cost: $465,000) and by CHE-0749788, 3/1/08 – 2/28/09 (total first year: $180,000), "Systematic Theoretical Description of Thermodynamics and Dynamics of Self-Assembly".

Support from the previous grant has contributed to 17 papers [FL1-17] and 6 are from the current grant. [FL18-23] Since there is no overlap with the current proposal, only a brief summary is provided for some of the projects concerning self-assembly and the related topic of glass formation in polymer systems. The self-assembly of dynamic clusters is a ubiquitous phenomenon in which the constituent molecules or particles form and disintegrate clusters of various geometries in a dynamical equilibrium. This dynamic clustering process is prevalent in biopolymer, polyelectrolyte, and ionic solutions, solutions of amphiphiles and molecules exhibiting supramolecular assembly with highly directional (hydrogen bonding, pi-pi, polar, and multipole) interactions. Self-assembly is a basic element of bottom-up nano-manufacturing, and elucidating the factors controlling the structure and the stability of dynamical particle assemblies is key to successfully applying this fabrication approach. **This NSF supported research has led to significant advances:** Our theory provides insight into numerous previously unexplained observations for, e.g., ionic fluids, where large scale correlation length amplitudes and narrow Ising critical regimes have been an enigma. Our theory of glass-formation in polymer fluids is the first to relate thermodynamic properties of polymer glasses to the monomer molecular structure, explaining many enigmatic trends in glass-formation and in the fragility of polymer glasses.

Research supported, in part, by NSF Grant OCI-0721939, 9/1/2007 – 8/31/2009, to Wilde, "SDCI NMI: Improvement of the Swift Science and Engineering", (total cost: $599,907). The main deliverable of this project is the middleware-level software tool "Swift" (which is an important component of the work proposed here, and is described briefly above and in the references) Just past its first year of funding, Swift is being used by scientists in biochemistry to do molecular ligand-protein docking at massive scale, neuroscientists to study language and other behavioral mechanisms, medical physicists to develop automated image pattern detection mechanisms, computer scientists to do automated image analysis for surgical planning, bioinformaticists to study human genetics, and economists to study energy policy. Publications include [ZE+08, ZH+07, ZRF08, ZR+08, ZWF07, CF+08]. **Research supported, in part, by NSF Grants PHY-0736126, 0538356 and 0636265, covering 10/1/2005 through 9/30/2011, to**

PI M. Bardeen, "I2U2: Interactions in Understanding the Universe" (cost: $1,604,640). As co-I of I2U2, Wilde supervises the design, development and support of grid services for the I2U2 science education project, serving approx. 200 high schools [BG+06, I08,LT07]. **Education supported in part by PHY 0621704, "Sustaining and Extending the Open Science Grid: Science Innovation on a PetaScale Nationwide Facility" 09/01/2006 to 8/31/2001, PI M. Livny ($8,427,000).** Wilde served 2006-2008 as the Education Coordinator of the Open Science Grid. **Workshop supported by NSF Grant OCI0736291 Cyberinfrastructure Learning and Workforce Development: CI-TEAM Community Building Workshop 06/01/2007, PI S. Lathrop ($144,151).** As co-I, Wilde assisted in the development and delivery of the NSF 2008 CI-TEAM PI's Workshop, facilitated workshop sessions, and spoke on OSG and TeraGrid cyberinfrastructure.

8 Concluding Summary

High speed, high accuracy and high capacity simulation of protein folding and docking/recognition are vital tools for advancing important problems in chemistry and biology. For instance, the description of docking site recognition and relaxation cannot proceed without efficient petascale facilities. This proposal will adapt OOPS to operate in a petascale environment, tuned, tested and supported on the 4 major open science platforms of the next half-decade. The OOPS tools have distinguished themselves in providing improved 2^o structure and comparable 3^o structure predictions to the best of these methods without the need for homology information. Thus, OOPS is poised to predict structures of the ever growing body of totally unknown, unsolved proteins. The many existing groups with their own folding codes will find our 2^o structure predictions (particularly in probabilistic form) essential for low-homology sequences. In addition, several components of the petascale OOPS framework will be highly reusable by other computational science software developers: the workflow framework (and the work that will enable this framework to be easily installable in a simple and automated manner) and the parallel energy calculator.

We propose a two-pronged strategy for utilizing petascale systems to achieve ultra-high capacity: internal parallelization of the OOPS simulation's costly energy calculations, and the embedding of OOPS into an already petascale capable, parallel scripting systems The resulting unique framework and approach will make OOPS run very fast on petascale machines with higher accuracy/resolution than previously possible, thus serving as an excellent example of the capabilities and benefits of petascale computing.

The proposed packaging approach will enable the parallel OOPS code to run in a wide range of environments – a necessity for supporting petascale systems with limited availability even to users with awards of time on them – and will provide a fully ready end user environment that will enable immediate use on the 4 main systems targeted here (at ALCF, ORNL, TeraGrid, and eventually Blue Waters).

Because so many of the risks of this effort have been mitigated through prototyping, petascale OOPS will be ready for initial use in very short order – and is demonstrable now at scales of 64k cores. Regardless of whether our or other existing folding/docking codes are used, the computational environment created will be generally useful for every group performing loosely coupled calculations.

Our multi-disciplinary collaborative team has the necessary combination of expertise in chemistry, computing, and computational science, a highly usable framework, and great scientific insight. The work benefits from technologies developed by collaborators within the UChicago Computation Institute and which can be adapted and supported to meet the needs of the petascale OOPS framework. Our CS team works closely with ALCF and the TeraGrid, is already running on BG/P and Ranger, and is applying a similar approach to applications across many disciplines. Thus, the likelihood of success is excellent, and the cost of the development and support of the core technologies is amortized over many projects.

Broader impact for education and its integration with research. We propose a compact but sufficient staffing plan – weighted to involve students and early-career scientists, and the project will provide excellent multidisciplinary training for graduate students, postdocs, and undergraduates. The broader impact of the proposed work is manifest both in our approach and demonstrated commitment (and results) in training, education and mentoring, and in the significant scientific and software fringe benefits of the proposed approach. In the TEAM arena, we propose an education plan that is effective and comprehensive for integrating research and education, and for utilizing the science and computational deliverables of this project immediately in science education from the high school through postdoctoral level – through e- and i-Labs (for high schools and science museums), courses and workshops (for graduate and advanced undergraduate education. Our plan for teaching the techniques of computational science will have broad benefits far beyond the specific science focus of the petascale OOPS framework.

REFERENCES

[AB+06] Asanovic, K., R. Bodik, et. al. The Landscape of Parallel Computing Research: A View from Berkeley, EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2006-183 December 18, 2006, http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

[AH+07] Alexander, P. A., Y. He, et al. (2007). "The design and characterization of two proteins with 88% sequence identity but different structure and function." Proc. Natl. Acad. Sci. U S A.

[ASF97] Altschul, S. F., et al (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nuc. Acids Res. **25**: 3389-3402.

[AUW08] J. Appavoo, V. Uhlig, A. Waterland. "Project Kittyhawk: Building a Global-Scale Computer," ACM Sigops Operating System Review, 2008

[BG+06] Marjorie Bardeen, Eric Gilbert, Thomas Jordan, Paul Nepywoda, Elizabeth Quigg, Michael Wilde, Yong Zhao: The QuarkNet/Grid Collaborative Learning e-Lab. Future Generation Comp. Syst. 22(6): 700-708 (2006)

[BM+05] Bradley, P., K. M. Misura, et al. (2005). "Toward high-resolution de novo structure prediction for small proteins." Science **309**(5742): 1868-71.

[BUG08] Bugzilla Defect Tracking System, http://www.bugzilla.org/about/, 2008.

[BW08] Blue Waters Project web site: http://www.ncsa.uiuc.edu/BlueWaters/, 2008.

[BW08b] Vance, Ashley. “IBM's eight-core Power7 chip to clock in at 4.0GHz, Will fuel 300,000-core, 10 petaflop giant”, 11 July 2008, http://www.theregister.co.uk/2008/07/11/ibm_power7_ncsa/

[C+08] CNARI – Computational Neuroscience and Aphasia Research Infrastructure, http://www.cnari.org, 2008.

[CF+08] Ben Clifford, Ian T. Foster, Jens-S. Vöckler, Michael Wilde, Yong Zhao: Tracking provenance in a virtual data grid. Concurrency and Computation: Practice and Experience 20(5): 565-575 (2008)

[CJ+06] Colubri, A., A. K. Jha, et al. (2006). "Minimalist Representations and the Importance of Nearest Neighbor Effects in Protein Folding Simulations." J. Mol. Biol. **363**: 835-857.

[CJ+07] B.Chapman, G. Jost, R. vanderPas, D.J. Kuck, Using OpenMP: Portable Shared Memory Parallel Programming. The MIT Press (October 31, 2007). ISBN 0262533022

[CI08] PADS – CI Petascale Active Data Store, NSF OCI Grant #, http://pads.ci.uchicago.edu/index.php,

[CI08b] University of Chicago Computation Institute, Colloquia and Events Calendar. http://www.ci.uchicago.edu/events/index.php, 2008.

[CO+07] J. Cope, M. Oberg, H.M. Tufo, T. Voran, M. Woitaszek. “High Throughput Grid Computing with an IBM Blue Gene/L,” Cluster 2007

[D05] N. Desai. “Cobalt: An Open Source Platform for HPC System Software Research,” Edinburgh BG/L System Software Workshop, 2005

[DAC90] Doyle, A. C. (1890). The Sign of the Four.

[DZ07] Dor, O. and Y. Zhou (2007). "Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training." Proteins **66**(4): 838-45.

[ES+06] Eramian, D., M. Y. Shen, et al. (2006). "A composite score for predicting errors in protein structure models." Protein Sci. **15**(7): 1653-66.

[FA08] Falkon historical activity, http://dev.globus.org/wiki/Incubator/Falkon/Activity

[FJ+07] Fitzgerald, J. E., A. K. Jha, et al. (2007). "Reduced Cbeta statistical potentials can outperform all-atom potentials in decoy identification." Protein Sci. **16**(10): 2123-39.

[FJ+08] Fitzgerald, J. E., A. K. Jha, et al. ((accepted)). "Reduced Cβ statistical potentials can outperform all-atom potentials in decoy identification." Protein Sci.

[FL1] The Glass Transition Temperature of Polymer Melts. J. Dudowicz, K. F. Freed, and J. F. Douglas, J. Phys. Chem. B **109**, 21285-92 (2005).

[FL2] Direct Computation of Characteristic Temperatures and Relaxation Times for Glass-Forming Polymer Liquids. J. Dudowicz, K. F. Freed, and J. F. Douglas, J. Chem. Phys. **123**, 111102 1-4 (2005).

[FL3] Statistical Coil Model of the Unfolded State: Resolving the Reconciliation Problem. A Jha, A. Colubri, K. F. Freed, and T. R. Sosnick, Proc. Natl. Acad. Sci. (US) **102**, 13099-104 (2005).

[FL4] Fragility of Glass-forming Polymer Liquids. J. Dudowicz, K. F. Freed, and J. F. Douglas, J. Phys. Chem. B **109**, 21350-6 (2005).

[FL5] Compressible Models of Equilibrium Polymerization. M. N. Artyomov and K. F. Freed, J. Chem. Phys. **123**, 194906 1-13 (2005).

[FL6] Entropy Theory of Polymer Glass-Formation: I. General Formulation. J. Dudowicz, K. F. Freed, and J. F. Douglas, J. Chem. Phys. **124**, 064901 1-14 (2006).

[FL7] Lattice Model of Equilibrium Polymerization. V. Scattering Properties and the Width of the Critical Regime for Phase Separation. K. Rah, K. F. Freed, J. Dudowicz, and J. F. Douglas, J. Chem. Phys. **124**, 144906 (2006); also appears in Virt. J. Nanoscale Sci. Techn. Apr. 26 (2006).

[FL8] Ab initio Description of the Ground and Excited States of Cyanogen Isomers, R. K. Chaudhuri, S. L. N. G. Krishnamachari, K. F. Freed, J. Mol. Spectr. THEOCHEM **768**, 119-26 (2006).

[FL9] Small Proteins Fold through Transition States with Native-like Topologies, A. Pandit, A. Jha, K. F Freed, and T. R. Sosnick, J. Mol. Biol. **361**, 755-70 (2006).

[FL10] Generalized Entropy Theory of Polymer Glass-Formation. J. Dudowicz, K. F. Freed, and J. F. Douglas, Adv. Chem. Phys. (in press).

[FL11] Minimalist representations and the importance of nearest neighbor effects in protein folding simulations. A. Colubri, A. Jha, M.-y. Shen, A. Sali, R. S. Berry, T. R. Sosnick, and K. F Freed, J. Mol. Biol. **363**, 835—57 (2006).

[FL12] Does Equilibrium Polymerization Describe the Dynamic Heterogeneity of Glass-forming Liquids? J. F. Douglas, J. Dudowicz, and K. F. Freed, J. Chem. Phys. **125**, 144907 (2006). Also appears in Virt. J. Nanoscale Sci. Techn. Oct. 30 (2006).

[FL13] Minimal Model of Relaxation in an Associating Fluid: Viscoelastic and Dielectric in Equilibrium Polymer Solutions. E. B. Stukalin and K. F. Freed, J. Chem. Phys. **125**, 184905 (2006).

[FL14] Polypeptide motions are dominated by peptide group oscillations resulting from dihedral angle correlations between nearest neighbors. J. E. Fitzgerald, A. Jha, T. R. Sosnick, and K. F. Freed, Biochemistry **46**, 669-82 (2007).

[FL15] Actin Polymerization under Pressure: A Theoretical Study. M. N. Artyomov and K. F. Freed, J. Chem. Phys. **126**, 024908 (2007).

[FL16] Colubri, A., Jha, A. K., Shen, M. Y., Sali, A., Berry, R. S., Sosnick, T. R. & Freed, K. F. (2006). Minimalist Representations and the Importance of Nearest Neighbor Effects in Protein Folding Simulations. *J. Mol. Biol.* **363**, 835-857.

[FL17] Lattice Model of Equilibrium Polymerization. VI. Measures of Fluid 'Complexity' and Search for Generalized Corresponding States. J. F. Douglas, J. Dudowicz, and K. F. Freed, J. Chem. Phys. **127**, 224901 (2007).

[FL18] Lattice Model of Equilibrium Polymerization: VII. Understanding the Role of "Cooperativity" in Self-assembly, J. F. Douglas, J. Dudowicz, and K. F. Freed, J. Chem. Phys. **128**, 224901 (2008).

[FL19] Reappraisal of *Cis* Effect in 1,2-dihaloethenes: An Improved Virtual Orbital Multireference Approach. R. K. Chaudhuri, K. F. Freed, S. K. Chattopadhyay, U. K. S. Mahaatra, and J. R. Hammond, J. Chem. Phys. **129**, 064101 (2008).

[FL20] IVO-MRMP Study of the Ground and Excited Electronic States of Protonated Acetylene, $C_2H_3^+$. R. K. Chaudhuri and K. F. Freed, J. Chem. Phys. **129**, 054308 (2008).

[FL21] Multi-step Relaxation in Equilibrium Polymer Solutions: A Minimal Model of Relaxation in "Complex" Association Fluids. E. B. Stukalin, J. F. Douglas, and K. F. Freed, J. Chem. Phys. **129**, 094901 (2008).

[FL22] Self-Assembly in a Polymer Matrix and its Impact on Phase Separation. J. Dudowicz, K. F. Freed, and J. F. Douglas, J. Phys. Chem. B (in press).

[FL23] Self-Assembly by Mutual Association: Basic Thermodynamic Properties. J. Dudowicz, J. F. Douglas, and K. F. Freed, J. Phys. Chem. B (in press).

[FL24] A. V. Davis, R. M. Yeh, and K. N. Raymond, Supramolecular chemistry and self-assembly special feature: Supramolecular assembly dynamics, Proc. Natl. Acad. Sci. 99, 4793-6 (2002)

[FL25] J.-M, Lehn, Supramolecular chemistry and self-assembly special feature: Toward complex matter: Supramolecular chemistry and self organization, Proc. Natl. Acad. Sci. 99, 4763-8 (2002).

[FL26] W. H. Binder, V. Barragan-Montero, and F. C. Menger, Domains and rafts in lipid membranes, Angew. Chem. 42, 5802-27 (2003).

[FL27] V. Percec, G. Ungar, and M. Peterca, Self-assembly in actin, Science 313, 55-56 (2006).

[H06] D. Hanson. "Enhancing Technology Representations within the Stanford Energy Modeling Forum (EMF) Climate Economic Models," Energy and Economic Policy Models: A Reexamination of Fundamentals, 2006

[H08] PADS Talk: http://pads.ci.uchicago.edu/ SOME PDF

[I08] I2U2 – Interactions in Understanding the Universe. Project Web Site: http://ww.i2u2.org

[IBM08] IBM BlueGene/P (BG/P), http://www.research.ibm.com/bluegene/, 2008

[IBM08b] IBM Coorporation. "High-Throughput Computing (HTC) Paradigm," IBM System Blue Gene Solution: Blue Gene/P Application Development, IBM RedBooks, 2008

[JC+05] Jha, A. K., A. Colubri, et al. (2005). "Statistical coil model of the unfolded state: Resolving the reconciliation problem." Proc. Natl. Acad. Sci. U S A **102**(37): 13099-104.

[JC+05b] Jha, A. K., A. Colubri, et al. (2005). "Helix, Sheet, and Polyproline II Frequencies and Strong Nearest Neighbor Effects in a Restricted Coil Library." Biochemistry **44**(28): 9691-702.

[JDT99] Jones, D. T. (1999). "Protein secondary structure prediction based on position-specific scoring matrices." J. Mol. Biol. **292**(2): 195-202.

[JFI08] University of Chicago, James Franck Institute, Colloquia and Events Calendar. http://jfi.uchicago.edu/events/talks.shtml, 2008.

[KD05] Kihara, D. (2005). "The effect of long-range interactions on the secondary structure formation of proteins." Protein Sci. **14**(8): 1955-63.

[LK+05] Liwo, A., M. Khalili, et al. (2005). "Ab initio simulations of protein-folding pathways by molecular dynamics with the united-residue model of polypeptide chains." Proc. Natl. Acad. Sci. U S A **102**(7): 2362-7.

[LT07] T. Loughran, "International Research Opportunities through Web Browsers: The I2U2 e-lab Suite", AAPT Summer Meeting, Greensboro, NC, June 2007.

[M+06] D.T. Moustakas et al. "Development and Validation of a Modular, Extensible Docking Program: DOCK 5," J. Comput. Aided Mol. Des. 20, 2006, pp. 601-619

[MB03] Meiler, J. and D. Baker (2003). "Coupled prediction of protein secondary and tertiary structure." Proc. Natl. Acad. Sci. U S A **100**(21): 12105-10.

[MF+05] Moreau, L., Zhao, Y., Foster, I., Voeckler, J. and Wilde, M. (2005) XDTM: The XML Dataset Typing and Mapping for Specifying Datasets. Proceedings of the 2005 European Grid Conference (EGC'05), Amsterdam, The Netherlands.

[MK96] Minor, D. L., Jr. and P. S. Kim (1996). "Context-dependent secondary structure formation of a designed protein sequence." Nature **380**(6576): 730-4.

[O98] J. Ousterhout, "Scripting: Higher Level Programming for the 21st Century", IEEE Computer, March 1998

[OW+07] Ozkan, S. B., G. A. Wu, et al. (2007). "Protein folding by zipping and assembly." Proc. Natl. Acad. Sci. U S A **104**(29): 11987–11992.

[PB+08] Quan T. Pham, Atilla S. Balkir, Jing Tie, Ian Foster, Mike Wilde, Ioan Raicu. "Data Intensive Scalable Computing on TeraGrid: A Comparison of MapReduce and Swift", Poster Presentation, TeraGrid Conference 2008.

[PK+08] A. Peters, A. King, T. Budnik, P. McCarthy, P. Michaud, M. Mundy, J. Sexton, G. Stewart. "Asynchronous Task Dispatch for High Throughput Computing for the eServer IBM Blue Gene® Supercomputer," Parallel and Distributed Processing (IPDPS), 2008

[PL08] PL protein library, http://protlib.uchicago.edu/, 2008

[PP+02] Pollastri, G., D. Przybylski, et al. (2002). "Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles." Proteins **47**(2): 228-35.

[RS+04] Rohl, C. A., C. E. Strauss, et al. (2004). "Protein structure prediction using Rosetta." Methods Enzymol **383**: 66-93.

[RZ+07] Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. "Falkon: a Fast and Light-weight tasK executiON framework", IEEE/ACM SuperComputing 2007.

[RZ+07b] Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. "A Data Diffusion Approach to Large Scale Scientific Exploration", Microsoft Research eScience Workshop 2007.

[RZ+08] Ioan Raicu, Zhao Zhang, Mike Wilde, Ian Foster, Pete Beckman, Kamil Iskra, Ben Clifford. "Toward Loosely Coupled Programming on Petascale Systems", to appear at IEEE/ACM Supercomputing 2008.

[RZ+08b] Ioan Raicu, Zhao Zhang, Mike Wilde, Ian Foster. "Enabling Loosely-Coupled Serial Job Execution on the IBM BlueGene/P Supercomputer and the SiCortex SC5832", Technical Report, Department of Computer Science, University of Chicago, April 2008.

[RZ+08c] Ioan Raicu, Yong Zhao, Ian Foster, Mike Wilde, Zhao Zhang, Ben Clifford, Mihael Hategan, Sarah Kenny. "Managing and Executing Loosely Coupled Large Scale Applications on Clusters, Grids, and Supercomputers", Extended Abstract, GlobusWorld08, part of Open Source Grid and Cluster Conference 2008.

[RZ+08d] Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. "Accelerating Large-scale Data Exploration through Data Diffusion", International Workshop on Data-Aware Distributed Computing 2008, co-locate with ACM/IEEE International Symposium High Performance Distributed Computing (HPDC) 2008.

[S08] Swift Workflow System Project web site, www.ci.uchicago.edu/swift, 2008

[SFP08] Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, Version Control with Subversion, http://svnbook.red-bean.com/, 2008.

[SP08] Single program, multiple data. http://en.wikipedia.org/wiki/SPMD, 2008.

[SR+06] Dinanath Sulakhe, Alex Rodriguez, Michael Wilde, Ian T. Foster, Natalia Maltsev: Using Multiple Grid Resources for Bioinformatics Applications in GADU. CCGRID 2006: 41

[SRF08] Service, R. F. (2008). "Problem solved* (*sort of)." Science **321**(5890): 784-6.

[SS06] Shen, M. Y. and A. Sali (2006). "Statistical potential for assessment and prediction of protein structures." Protein Sci. **15**(11): 2507-24.

[SF04] Srinivasan, R., P. J. Fleming, et al. (2004). "Ab initio protein folding using LINUS." Methods Enzymol **383**: 48-66.

[SR95] Srinivasan, R. and G. D. Rose (1995). "LINUS: a hierarchic procedure to predict the fold of a protein." Proteins **22**(2): 81-99.

[TTL05] D. Thain, T. Tannenbaum, M. Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience 17(2-4), 2005, pp. 323-356

[WP+08] Wiehe, K., M. W. Peterson, et al. (2008). "Protein-protein docking: overview and performance analysis." Methods Mol. Biol. **413**: 283-314.

[YC+07] Yang, J. S., W. W. Chen, et al. (2007). "All-atom ab initio folding of a diverse set of proteins." Structure **15**(1): 53-63.

[Z08] ZeptoOS Project Web. http://www-unix.mcs.anl.gov/zeptoos/, 2008.

[ZD+05] Y. Zhao, J. Dobson, I. Foster, L. Moreau, M. Wilde. *A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data*. SIGMOD Record, Sep 2005.

[ZE+08] Zhao Zhang, Allan Espinosa, Kamil Iskra, Ioan Raicu, Ian Foster, Michael Wilde. "Design and Evaluation of a Collective I/O Model for Loosely-coupled Petascale Programming", to appear at IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008, co-located with IEEE/ACM Supercomputing 2008.

[ZH+07] Zhao Y., Hategan, M., Clifford, B., Foster, I., vonLaszewski, G., Raicu, I., Stef-Praun, T. and Wilde, M., Swift: Fast, Reliable, Loosely Coupled Parallel Computation IEEE International Workshop on Scientific Workflows 2007.

[ZRF08] Yong Zhao, Ioan Raicu, Ian Foster. "Scientific Workflow Systems for 21st Century e-Science, New Bottle or New Wine?", Invited Paper, IEEE Workshop on Scientific Workflows 2008, co-located with IEEE International Conference on Services Computing (SCC) 2008.

[ZR+08] Yong Zhao, Ioan Raicu, Ian Foster, Mihael Hategan, Veronika Nefedova, Mike Wilde. "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", to appear as a book chapter in Grid Computing Research Progress, ISBN: 978-1-60456-404-4, Nova Publisher 2008.

[ZS+03] Zaman, M. H., M. Y. Shen, et al. (2003). "Investigations into sequence and conformational dependence of backbone entropy, inter-basin dynamics and the Flory isolated-pair hypothesis for peptides." J. Mol. Biol. **331**(3): 693-711.

[ZWF06] Zhao, Y., Wilde, M. and Foster, I., Applying the Virtual Data Provenance Model. International Provenance and Annotation Workshop, Chicago, Illinois, 2006.

[ZWF07] Y. Zhao, M. Wilde, and I. Foster. *Virtual Data Language: A Typed Workflow Notation for Diversely Structured Scientific Data*, In Taylor, I.J., Deelman, E., Gannon, D.B. and Shields, M. eds. Workflows for eScience, Springer, 2007, 258-278.

[ZWF06] Y. Zhao, M. Wilde, I. Foster. *Applying the Virtual Data Provenance Model*, International Provenance and Annotation Workshop, Chicago, IL USA, May 3-5, 2006.