Implementation and Characterization of Protein Folding on a Desktop Computational Grid

Is CHARMM a suitable candidate for the United Devices MetaProcessor?

B. Uk[1], M. Taufer[1], T. Stricker[1], G. Settanni[2], A. Cavalli[2]

[1] Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
buk,taufer,tomstr@inf.ethz.ch

[2] Department of Biochemistry
University of Zurich
CH-8057 Zurich, Switzerland
settanni,cavalli@bioc.unizh.ch

Abstract

CHARMM is a popular molecular dynamics code for computational biology. For many CHARMM applications such as protein folding, desktop grids could become viable alternatives to clusters of PCs.
In this technical report, we present a prototype and discuss the viability of a protein folding application with CHARMM on the United Devices MetaProcessor, a platform for widely distributed computing. We identify the algorithmic approach of protein folding as a hybrid search algorithm with best-first, depth-first and breadth-first components and address the issues of task scheduling and fault tolerance.
The performance evaluation of our system indicates that the calculation is robust against the heterogeneity of compute nodes and limited communication capabilities typically found in desktop grids. We show that there is an interesting tradeoff between accuracy and task parallelism resulting in optimal work-pool size for a given platform and a given simulation. Surprisingly the platform heterogeneity of a desktop grid positively affects the quality of protein folding simulations.
Protein folding calculations with CHARMM turn out to be well suitable for desktop grids like e.g. the United Devices MetaProcessor. Our software system can make a large amount of nearly free compute cycles available to computational biologists.

Keywords: protein folding, widely distributed computing, desktop grids, workload characterization, search algorithms, CHARMM, United Devices MetaProcessor.

1 Introduction

Millions of PCs are in use everywhere but remain under-used most of the time, utilizing only 18% of their power on average during normal use (word processing, web browsing, e-mail etc.) [3, 31, 23]. At the same time, some computationally intensive problems are too large for even todays largest supercomputers to solve in reasonable amounts of time. We find examples of such computations in many fields of computational sciences, cryptography and mathematics.
The goal of a wide variety of technologies like Globus [11, 12], Condor [2], EU DataGrid [27, 16], and GriPhyN [10], is to transform the Internet into a powerful computational platform. Moreover, a significant industrial effort will provide standard software for similar systems in the near future (e.g. Entropia [5], United Devices [1]).

Many projects try to utilize the *unused computing power* available on the Intranet or the Internet for large-scale problems. The possibility of harnessing the power of idle processors in a company or all over the world has been proven by the success of technology demonstrators like *Seti@home* for the analysis of radio telescope data [24] or *distributed.net* for testing cryptographic keys against security flaws [7]. These projects involve so-called embarrassingly parallel problems, meaning that each participating node performs a lot of work that is not dependent on the work of other nodes.

Protein folding research is an area of computational biology that could greatly benefit from free computational power. Protein folding simulations using molecular dynamics require huge amounts of computing power, but unlike cryptographic key-space searches or cosmic signal analysis they are not embarrassingly parallel.

Previous research addresses the issue of protein folding simulations on distributed platforms, but no project explicitly investigates the amount of task parallelism available and the effect of tasking on the accuracy of the results. CHARMM on the distributed computing platform Legion has been studied in [20]. The implementation described there specializes in high-performance, strongly interconnected clusters and a fairly small number of tasks. Our work applies more broadly to desktop grids built from commodity PCs connected by a wide variety of stronger and weaker network technologies. The *Folding@home* project conducted by researchers at Stanford University [9, 35, 34] is based on the TINKER molecular dynamics package and uses plain task parallelism on PCs connected via Internet. For a first implementation the supercomputer code was re-designed so that only trajectories overcoming energy barriers were continued [25]. The resulting study dealt with a collection of foldings of a 16 residue β-hairpin peptide [35]. In a more advanced implementation the same team removed all dependencies among trajectories resulting in a most scalable code designed to collect as much simulation time as possible [34]. The latter version of the code was applied to the unfolded state of 3 small proteins (ranging from 12 to 36 residues) and none of the proteins showed clear folding events in the simulated time.

As a starting point to the work described in this technical report, we also use plain task parallelism to successfully migrate the CHARMM code for a protein folding simulation to the widely distributed platform [30] of United Devices (UD): the UD MetaProcessor. However, in our study of the viability and effectiveness of CHARMM on the UD MetaProcessor we focus our attention on the whole complex system including both the platform and the application parameters. In this first prototype of the protein folding application, we carefully study issues like the effect of the heterogeneity of the computation platform under investigation and the effect of several workload parameters, such as the number of molecular dynamics steps per work-unit (or work-unit size) and the maximum number of work-units in progress at any time (i.e. work-pool size).

As a result of this study on workload characteristics and application performance, we find that there are significant limitations of the scalability due to the limited amount of task parallelism generated by a protein folding calculation. In particular, large work-unit sizes providing simulations with better quality factors, and calculation methods such as PME providing higher accuracy, both increase the amount of work per work-unit and therefore the turn-around time of a single task. On the other hand, simulations with a short overall turn-around time make the biologists more productive in their research.

In Section 2, we briefly introduce the protein folding process using CHARMM code. We also present the United Devices MetaProcessor, a commercial platform for widely distributing computing that we used as a grid platform in our investigation. We show the adaptation and the migration of the protein folding process to our platform for widely distributed computing and identify the algorithmic approach of protein folding as a hybrid search algorithm with best-first, depth-first and breadth-first components. In Section 3, we present our experimental results of the study of critical issues like the effect of heterogeneity, work-unit size and work-pool size on the execution time and the quality of the results for the protein folding of a specific protein, the SH3 domain. In Section 4, we conclude summarizing our results and pointing out some relevant limits of task parallelism for protein folding application on such a widely distributed platform.

2 Protein Folding on Computational Desktop Grid

2.1 Protein Folding with CHARMM

CHARMM is a code for simulating the structure of biologically relevant macromolecules (proteins, DNA, RNA) [4]. It uses classical mechanical methods to investigate potential energy surfaces derived from experimental and "ab initio" quantum chemical calculations [18]. We use CHARMM for Molecular Dynamics (MD) simulations at constant temperature to investigate the protein folding process, in which the Newton equation of motion of the system (protein + thermal bath) is discretize and solved by an integration procedure (Verlet algorithm). The force on the atoms is the negative gradient of the CHARMM potential energy [18].
Basically, in a protein folding simulation we perform a search for trajectories leading to conformations close to the native (folded) conformation of a protein, starting from an unfolded conformation. An entire trajectory for which a certain amount of sub-trajectories or work-units have been completed is characterized by its starting conformation with a starting quality factor, as well as the best calculated quality factor for the entire trajectory with the related conformation and the the set of work-units leading from the starting conformation to the best conformation.
In each work-unit, an MD simulation of a certain number of molecular dynamics steps (or *work-unit size*) and with constant temperature is performed. The entire folding process is guided by the measure of a *quality factor*, which is repeated calculated for a certain amount of snapshots along each work-unit computation. The best quality factor found in a sub-trajectory (lower quality factors are better!) guides the overall branch and bound search for the best conformation. The best conformation with lower quality factor is returned at the end when the work-unit is completed. The quality factor used here is the ϕ-RMSD introduced in [33]. In our implementation we use a slightly different definition [15]. According to this definition, the conformations minimizing the quality factor should belong to the folding transition state ensemble (i.e. they lie on the top of the free energy barrier separating the folded from the unfolded population).
During a protein folding simulation, multiple work-units are processed simultaneously in parallel. New work-units are always generated using the best result obtained so far as starting conformation and assigning to this conformation new random atom velocities at the same temperature. The effectiveness of this kind of algorithms in molecular dynamics simulations has been discussed in [26].

2.2 The United Devices MetaProcessor

The United Devices (UD) MetaProcessor platform (MP platform) [31, 32] provides an environment for running compute intensive tasks distributed over many desktop-class machines in a corporate-wide Intranet or over the worldwide Internet. The MP platform is well established for problems with low communication to computation ratio and involving coarse-grain parallelism with no dependencies between work-units. Figure 1 shows the MetaProcessor platform architecture and components.
The MP server is the link between the MP platform and the participating agents. It is responsible for scheduling, as well as for the distribution of task modules, resident data and work-units to the agents, and for receiving results returned by the agents. The management server allows access to the internal data structures via the *Management API* (Application Programming Interface), allowing for instance the submission of work-units and retrieval of results. The database contains all relevant information of the MP platform. The database, MP server and Management server form the server side of the MP platform.
The UD agent is a small program which runs on each participating device (or *worker*). The agent communicates with the MP server to request work-units, resident data and task modules as needed, executes the task module on the participating device, and returns the task's results to the server. The management console provides a web-based interface to the MP platform, allowing administrative tasks to be performed. *Work-units* are submitted to

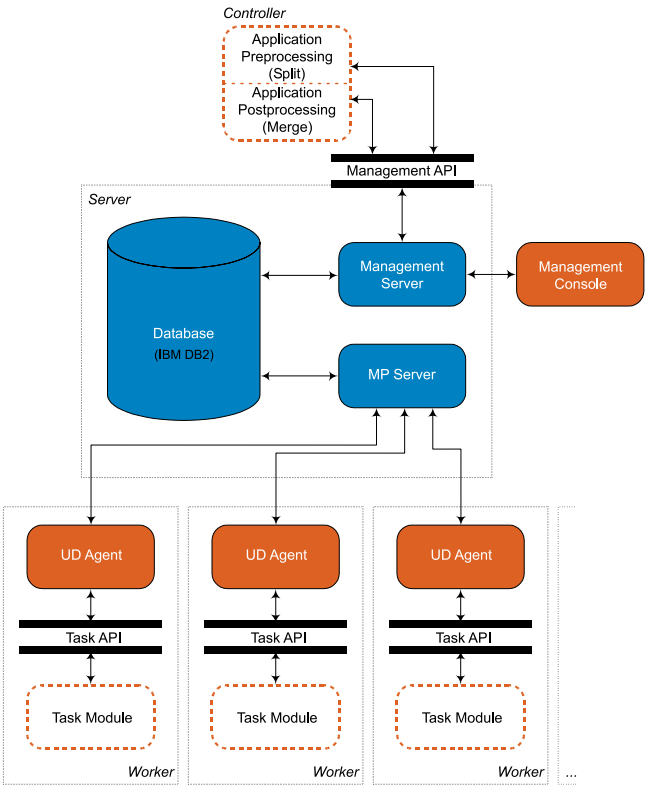

Figure 1: MetaProcessor platform overview.

the server via the Management API by a *controller process* which performs the generation of work-units and the retrieval of results.

2.3 Porting Protein Folding to the UD MetaProcessor

Many unique properties of the UD MetaProcessor platform must be taken into account to successfully port a protein folding application to this distributed platform. An application-specific controller process which generates work-units and submits them to the MP server has to be designed and implemented. However, this controller does not interact directly with the workers. Workers communicate only with the MP server, which performs the scheduling decisions.

For our protein folding application, we implemented such a controller process, taking into account both the issues related to the protein folding listed in the previous section and the features of the UD MetaProcessor.

At the beginning of a protein folding simulation, our controller process creates and submits a set of starting work-units to the MP server (initialization phase). These work-units all have the same protein conformation but a different random seed number. The work-units are also characterized by a starting best quality factor. The work-unit set is defined as the *work-pool*. The user decides the dimension of the work-pool. The number of work-units in the work-pool is called the *work-pool size*.

During the whole simulation, besides the set of work-units which have been completed and for which results have already been returned, we have the set of work-units which are either in process by a worker or waiting until they are assigned to a worker. At regular intervals during a so called update phase, the state of the work-pool is inspected by the controller process. The control process is responsible for generating and submitting to the work-pool additional work-units in substitution to the ones which have reached the end so that the sum of the work-units in process by a worker and the ones in waiting in the server queue is the *work-pool size*. The

user decides the dimension of the update interval. For the generation of new work-units, the control process sorts out whether between the set of work-units which have reached the end since the last update, there are any work-units which improve the best quality factor at that time. In case there is a work-unit with improved quality factor then the new work-units are generated using its conformation while the related quality factor becomes the new best quality factor. In the case that no improvement has been found, the last best conformation is used for generating the work-units to add to the work-pool while the best quality factor remains unchanged.
For scheduling the work-units (tasks) we incorporate the master-worker setting prescribed by the UD MetaProcessor software toolkit because it is well suitable for the task parallelism resulting from the application of CHARMM to protein folding.
Once new work-units are queued to the MP server in its work-pool, it is the server which distributes the new work-units and the ones already present in the queue across the agents in the distributed system. The scheduling decisions over which task to select next are left to the United Devices MetaProcessor and in the current version of the software there is little influence left to the application writer.
The scheduler in the MP server provides so-called *eager scheduling*, which basically means that as long as the server has work-units queued that are not being processed by other workers, it will send out one of these work-units to a worker requesting work [17]. A worker on which a finished work-unit has been running asks for and receives a new work-unit from the MP server. If however, all work-units have already been sent to workers, the server will re-send work-units for which no results have been obtained yet. This eager scheduling provides a simple method for fault tolerance, i.e. if a worker has crashed, eventually the work-unit it was processing will get resent to a different worker. This scheme can also have negative effects, for instance causing work-units of slow machines to be rescheduled, resulting in redundant results and thus a potential waste of computation time. So the rescheduling feature of the MP platform is in fact not desirable. To avoid rescheduling of work-units, we take a simple approach by ensuring that the size of the work-pool is always larger than the number of workers contributing.
Since the starting conformation of each newly generated work-unit depends on the results returned so far, there are dependencies between work-units, making the task generation and scheduling of this application a bit more challenging than typical “embarrassingly parallel” case of a statically generated search application.
The protein folding algorithm used is in itself *fault tolerant*, because the work-units generated from a certain conformation are identical except from a random number seed that is used to assign new atom velocities. Therefore the results are only marginally affected by the infrequent loss of an online work-unit.

2.4 Protein Folding as a Search Algorithm

The process of protein folding for a given protein is a search through a tree of a large number of possible configuration of the atoms in the protein, a so called conformation, under investigation. Each node in the tree corresponds to a different configuration of the atoms in the molecule. For each configuration there is an heuristic function to guide further decisions in the search. This function calculates the quality factor based on the total energy of the atom configuration.
In the process of looking for best conformations, we perform a search in breadth and in depth on a tree of three-dimensional atom configurations. Correspondingly there are two kinds of changes to the three-dimensional atom configurations.

2.4.1 Energy minimization as search in depth

The first kind of changes are smooth changes and due to the mechanical forces between the atoms. The process of calculation for these changes is a physical simulation called energy minimization. A sub-trajectory of states within such an energy minimization is a linear chain without any branches and is encapsulated in a *work-unit*

or a *task*, which will be our basic quantum for the distribution and the scheduling of the protein folding computation. Since it is too expensive to evaluate every configuration during energy minimization, the calculation of the quality factor is limited to certain snapshots during the energy minimization. In the calculations considered for this report, snapshots were taken every 100 simulation steps.

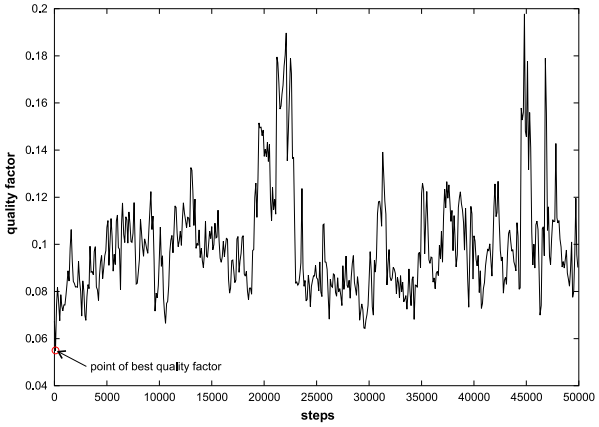

Figure 2: Quality factor along the sub-trajectory computed by a work-unit (50'000 steps, a snapshot every 100 steps). The snapshot with the best quality factor (minimum) is at the beginning.

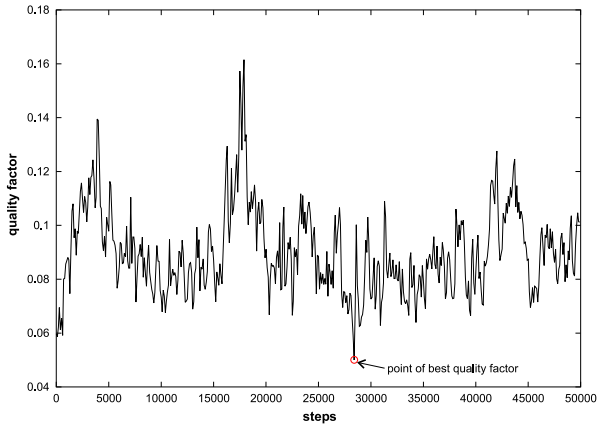

Figure 3: Quality factor along a sub-trajectory computed by a work-unit (50'000 steps, a snapshot every 100 steps). The snapshot with the best quality factor (minimum) is in the second half of the sub-trajectory.

Figures 2 and 3 show examples of the path the quality factor takes along two different sub-trajectories of an energy minimization with the same work-unit size of 50000 steps. One property of this simulation is that it is not possible to predict if or when an improvement of the quality factor will occur. Specifically in our cases a simulation concludes after a fixed number of steps but in general adaptive criteria based on the quality factors are also supported. The result of the simulation is the snapshot of configuration with the best quality factor obtained within the entire sub-trajectory. At a higher level of abstraction, the process of energy minimization for a single conformation candidate can be collapsed into a single edge in the higher level search tree.

2.4.2 Randomized conformations as search in breadth

The search in breadth is triggered by a second kind of changes to the configurations of a molecule under investigation. The folding process intentionally introduces disturbances to the molecular system by reassigning the atom velocities of a conformation with random numbers while preserving the temperature. Each work-unit generated in the process of randomization receives a different random seed. Multiple work-units are generated from one single conformation using different seeds. This results in the branches of a breadth-first search.
Figure 4 shows a possible configuration of the search tree. In the picture, we have three update phases for which new work-units are generated and queued in a work-pool.
In Figure 4, the protein conformation (Mol_1) in the first update is the best found at that time and we assume that an empty work-pool is fully completed by new work-units each one with a different random seed. In the second update phase only two work-units have reached the end while two others are either in progress by workers or waiting at the MP server. Between the work-units which reach the end, the conformation with best quality factor (Mol_2) is chosen and two new work-units are generated and added to the server queue by the control process. Note that the work-units not finished yet remain in the work-pool queue. During the third update,

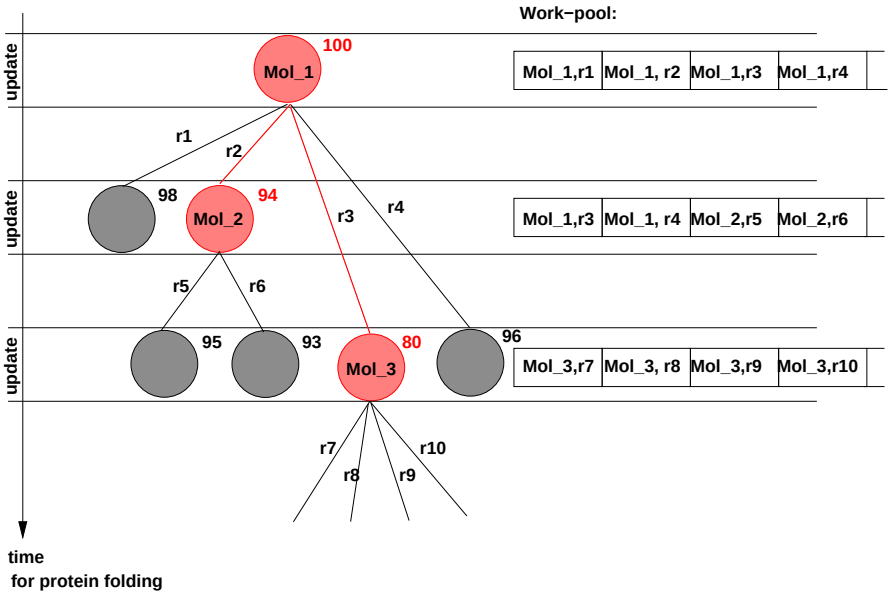

Figure 4: Example of a piece of a search tree for a protein folding simulation in which for each update phase new work-units are generated from improvements of the quality factor observed and queued in a work-pool.

all the four work-pools are finished. This time the best conformation found (Mol_3) is used to add four new work-units to the work-pool.

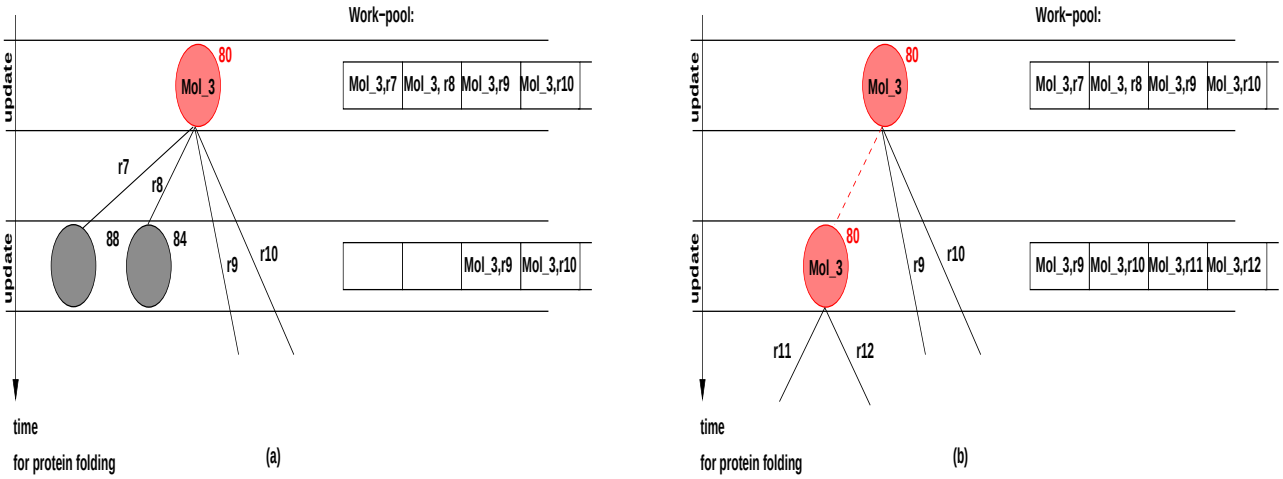

Figure 5: Example of a piece of a search tree for a protein folding simulation in which at the end of an update phase no improvement of the quality factor can be observed for the work-units completed.

Figure 5.a shows the case in which no improvement of the quality factor can be observed for the work-units completed between two update phases. In this case, the two new conformations are discarded and Mol_3, which is still the conformation with the best quality factor, is used again as the starting conformation of the two newly generated work-units (Figure 5.b). If this situation occurs repetitively, the protein folding simulation has reached a local minimum, and new strategies are needed to escape this minimum.

2.4.3 Parameters affecting the search process

The branching factor of the search in breadth is adaptive and depends on the work-pool size and the number of task completed in the mean time, while the search in depth determines primarily the waiting time and turn-around time of the work-units in the queues.
The waiting time depends on the ratio of the work-pool size over the number of workers that are available on a particular platform. The turn-around time of the single work-units is due to the work-unit size as well as the CPU clock rates and the network technologies of the nodes on the heterogenous platform.

3 Characterization of Protein Folding on UD MetaProcessor

3.1 The Biological Structure Folded

As an example of workload for our study we chose the src-SH3 domain. It is a 56 residue protein consisting of two anti-parallel β-sheets packed to form a single hydrophobic core (see Figure 6). The folding characteristics of src-SH3 domain (and structurally homologous proteins) have been thoroughly studied both from the experimental and the theoretical point of view [6, 13, 19, 22], and a detailed picture of the folding nucleus [21] and the transition state ensemble [22] is now available. The SH3 domain represents a sort of testing table for theories or algorithms dealing with protein folding. Its short length and the abundance of comparable studies make this protein the optimal target for our computations. The size of this protein does not allow to simulate its behavior with an explicit treatment for water; nevertheless the implicit solvation model used here, that is based on the solvent accessible surface area of the atoms (SASA) [8] successfully described several aspects of the dynamic properties of this protein [13, 14].

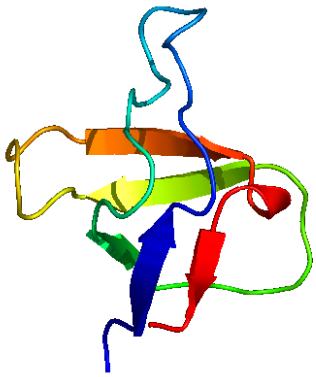

Figure 6: The src-SH3 domain is a 56 residue protein consisting of two anti-parallel β-sheets packed to form a single hydrophobic core.

3.2 Platform and Application Parameters

We use different testbed configurations with machines at ETH Zurich and at the private residences of several researchers for studying the effect that platform and application parameters have on quality and total turn-around time of the protein folding of the SH3 domain.

First of all, we investigate the impact of the heterogeneity on the quality of the protein folding results. In particular, we look at the kind of CPU clock rates and network technology as a representative of the platform parameters of the heterogenous computational grid.
We also study the effect of aspects directly related to the application (i.e. application parameters) like the work-unit size and work-pool size on the quality and turn-around time of the protein folding.

3.3 Effect of Heterogeneity on the Quality Parameter

3.3.1 Testbed for the study of heterogeneity effect

Two long-running tests using machines available around the clock on a first testbed are conducted over a period of 20 days. The machines range in CPU clock rate from 350 MHz to 1.6 GHz, providing a wide performance spectrum. Also, a widely varying range of network connectivity technologies are in use, from machines in the same LAN segment as the server (100 Mbit/s switched Ethernet) to machines at home connected via ADSL (128 kbit/s to 512 kbit/s).
Most of the machines involved are used during the course of the tests for normal day-to-day tasks. Because the UD agent spawns the task module processes at low priority, only otherwise unused processor cycles are consumed by the tests.
The following application parameters have been chosen for these tests:

- 100'000 simulation steps per work-unit,
- platform size: 45 machines,
- a work-pool of 50 work-units,
- polling frequency: once every 5 minutes.

The two tests are subject to a number of unforeseen disruptions (e.g. network failures, failure of machines, including the machine running the controller process, server's disk full), which cause the test to stall occasionally, exhibiting many properties which usually characterize real widely distributed grid platforms.

3.3.2 Experimental results of heterogeneity effect

An important question in a heterogeneous environment like the one we are studying, is whether slow machines are able to provide enough contributions to the overall computation to justify their participation. In our case, work-units are generated based on the results of previous work-units. The time at which a result is returned is therefore a parameter which can not be ignored. It is possible that, during the processing time of a work-unit on one machine, the state of the entire system has progressed so that the result of the work-unit of the considered machine is obsolete and therefore not used for the generation of new work-units configurations. If slow machines return a disproportionately large amount of obsolete results, their contribution to the entire calculation will be less valuable.
The long-running tests show that results returned by all machines, fast and slow, are accepted as good configurations. Figures 7 and 8 show the CPU clock rate for each result which was accepted in the long-running tests. Most results are from machines with CPU clock rates between 910 MHz and 1010 MHz. This is due to the fact that many such machines are used, and thus the total number of work-units completed by these machines was large. Nevertheless, slower machines make contributions during all phases of the simulation. Note that in Figure 8, no machine faster than 1010 MHz participated in the calculation.

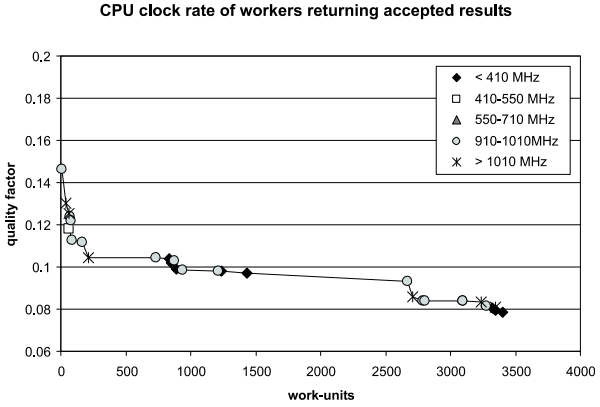

Figure 7: CPU clock rate of nodes returning accepted results, for the first long-running test.

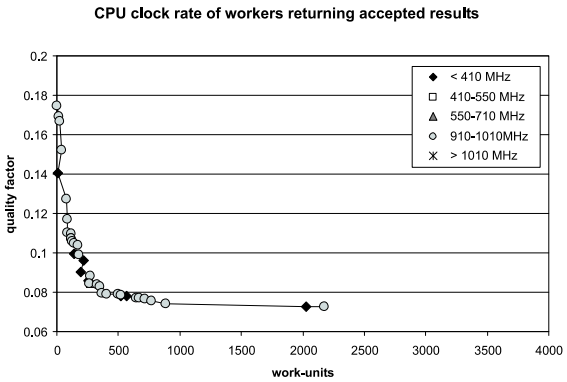

Figure 8: CPU clock rate of nodes returning accepted results, for the second long-running test.

3.4 Effect of Work-Unit Size on the Execution Time and the Quality Parameter

3.4.1 Testbed for the study of work-unit size effect

An important parameter of our simulations is the work-unit size. To study the effect of the *work-unit size* on the quality factor and the resource usage, a number of short-running tests of 12 hours are conducted on two testbed configurations with different numbers of machines connected by 100 Mbit/s switched Ethernet. The first testbed comprises:

- 12 machines 400 MHz Pentium III (Katmai)
- 12 machines 500 MHz Pentium III (Katmai)
- 12 machines 600 MHz Pentium III (Katmai)
- 12 machines 933 MHz Pentium III (Coppermine)
- 12 machines 1000 MHz Pentium III (Coppermine)

The second testbed uses a subset of the above listed machines and comprises:

- 6 machines 400 MHz Pentium III (Katmai)
- 6 machines 500 MHz Pentium III (Katmai)
- 6 machines 600 MHz Pentium III (Katmai)
- 6 machines 933 MHz Pentium III (Coppermine)
- 6 machines 1000 MHz Pentium III (Coppermine)

In these tests, two different work-unit sizes are considered and their effect on the turn-around time as well as the quality factor is investigated. The work-unit sizes are: 20000 simulation steps and 40000 simulation steps per work-unit.

3.4.2 Experimental results of work-unit size effect

Figure 9 shows the results of the quality factor for the two different work-unit size chosen (20000 and 40000 simulation steps). The tests with 40000 steps reach good quality factors after fewer processed work-units. Although the other tests with 20000 steps also reached similar quality factors after larger amounts of processed work-units, in two of the three cases presented in Figure 9, the test with 40000 steps still obtained the best quality factor at the end.
The amount of data communicated from the server to the worker per work-unit for the application chosen is around 15 kilobytes. The amount of data returned by the worker to the server depends on whether the work-unit simulation actually generates an improvement of the quality factor. If this is the case, the worker sends the entire sub-trajectory to the server. Figure 10 shows the data communicated in kilobytes for different work-unit sizes for this case. In case of no improvement of the quality factor by a work-unit computation, which is the most common case, the amount of data returned to the MP server is drastically reduced.
The application performance is limited by the CPU clock rate. As the work-unit size increases, we have observed an unexpected non-linear rise of the total floating point operations per work-unit. Figure 11 shows the amount of floating point operations for different work-unit sizes and the related non-linear behavior (the dashed line shows the linear behavior). The data related to the resource usage was measured by means of a framework for monitoring performance on limited wide area testbed developed by our group [29, 28].

3.5 Effect of Work-Pool Size on the Execution Time and the Quality Factor

3.5.1 Testbed for the study of work-pool size effect

Different values for the work-pool size on different platform sizes are chosen to investigate the work-pool effect on the quality factor. For the investigation of the effect of the *work-pool size*, a number of short-running tests of 12 hours are conducted on two different testbed configurations each one characterized by a different subset of work-pool sizes. Both the testbed configurations are connected by 100 Mbit/s switched Ethernet while the work-pool size is constant (10000 simulations steps).
The fist testbed comprises the following machines:

- 12 machines 500 MHz Pentium III (Katmai)
- 12 machines 600 MHz Pentium III (Katmai)
- 24 machines 933 MHz Pentium III (Coppermine)

The work-pool sizes considered for the first testbed are: 70, 120, 200.
The second testbed has the following structure:

- 12 machines 400 MHz Pentium III (Katmai)
- 12 machines 1000 MHz Pentium III (Coppermine)

The different work-pool sizes considered for the second testbed are: 30, 45, 60.

3.5.2 Experimental results of work-pool size effect

Figure 12 shows the minimum, average and maximum quality factors obtained using three different levels of the work-pool size for the short-running tests on the larger testbed under investigation. The work-unit size is maintained constant (10000 simulation steps per work-unit). Figure 13 shows the same quality factors (minimum, average and maximum) for the smaller testbed. Again the work-unit size is 10000. For the larger

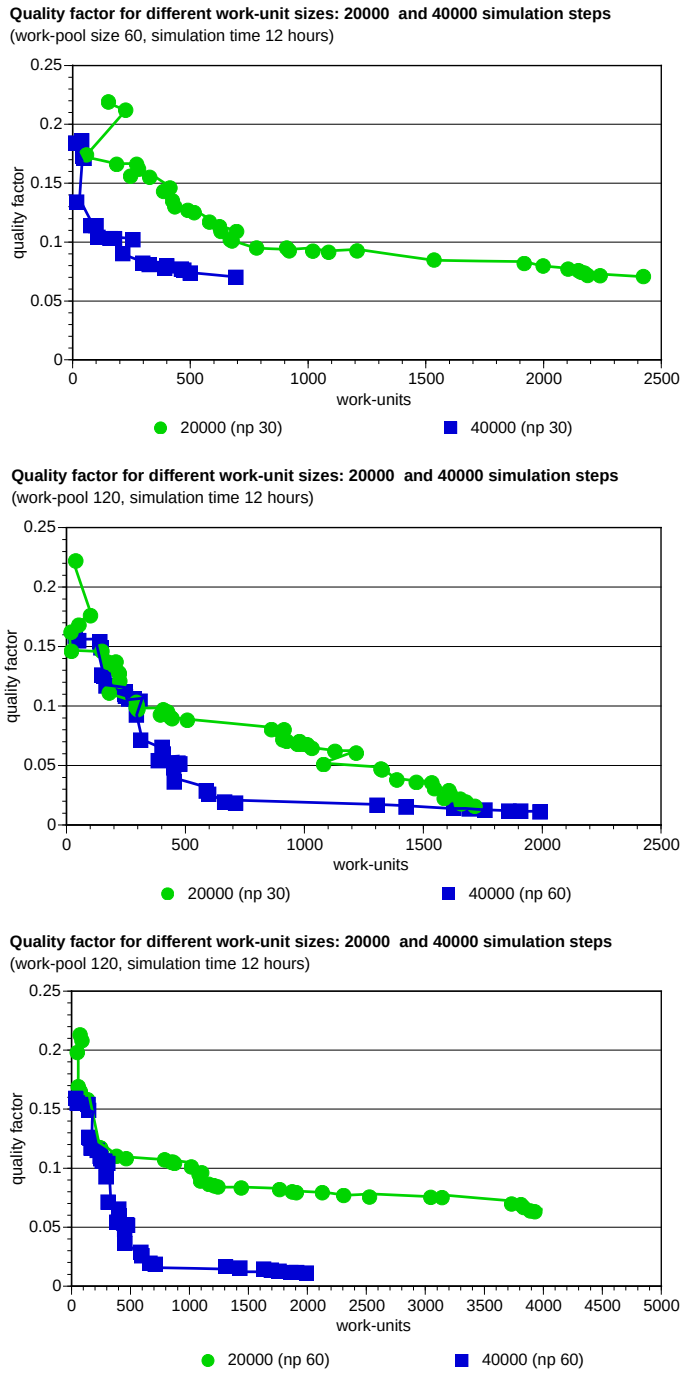

Figure 9: Effect of work-unit size on the quality factor. Two platforms with different numbers of nodes are taken into account, i.e. a platform with 30 nodes (np 30) and a platform with 60 nodes (np 60). Each point in the pictures is a work-unit which returned a result accepted due to improvement of the quality factor. In all cases, the tests with 40000 simulation steps per work-unit reached good quality factors after fewer work-units than the tests with less steps per work-unit.

testbed, the tests with a work-pool size of 120 reached the best quality factor in average. On the smaller testbed, the best quality factor in average is reached with a work-pool size of 45. This implies that for each platform

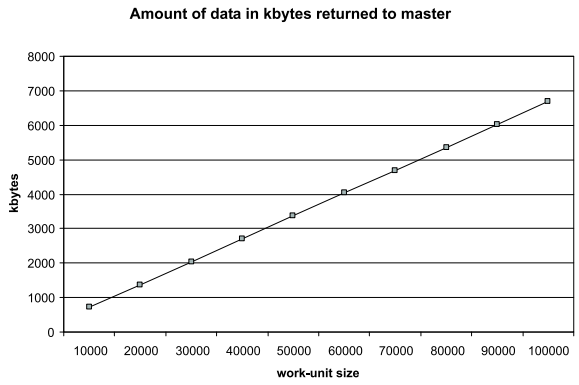

Figure 10: Size of the data communicated for different work-unit sizes when a sub-trajectory is accepted because it provides an improvement of the quality factor.

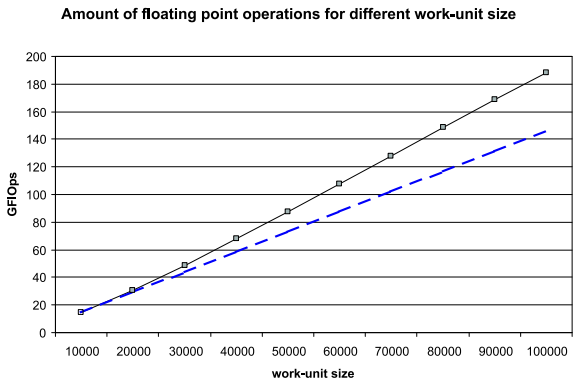

Figure 11: Amount of floating point operations for different work-unit sizes and the related non-linear behavior (the dashed line shows the linear behavior).

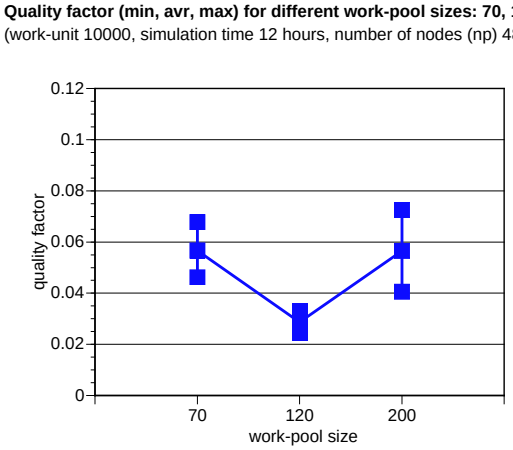

Figure 12: Minimum, average and maximum quality factor for different work-pool sizes (i.e.70, 120, 200) on a testbed with 48 machines.

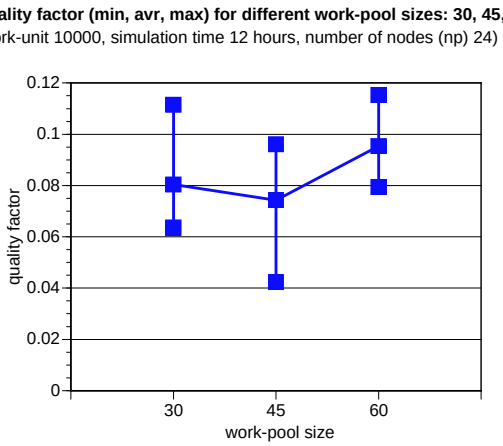

Figure 13: Minimum, average and maximum quality factor for different work-pool sizes (i.e. 30, 45, 60) on a testbed with 24 machines.

size there is an optimal work-pool size which leads to the best results (i.e. quality factor). To keep all workers busy the work-pool size should be larger than the number of participating workers. Therefore, the dependency between the quality factor and the work-pool size inherently limits the amount of task-parallelism available in the application and the potential for speed-up in a widely distributed computation.

Overly large work-pool sizes can cause a slow convergence of the quality function and can therefore result in longer turn-around time as displayed in Figure 14. The figure reports the quality factor of repeated tests on the smaller testbed. For the several tests, the work-pool size is 10000 simulation steps. On the other hand, we can also see comparing the three pictures that as the work-pool size increases, the contribution to the simulation provided by the slower machines (400 MHz) becomes more and more relevant. Moreover, larger work-pool sizes help to avoid local minimum and facilitate the resolution of simulations stuck in a local minimum.

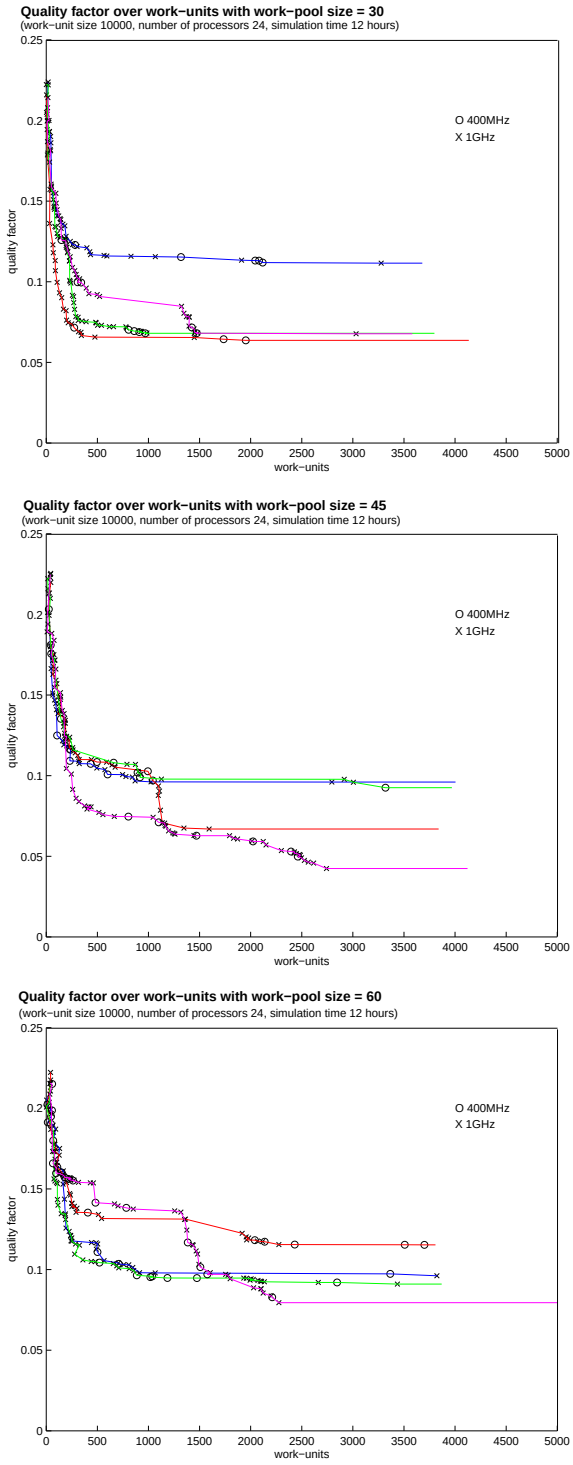

Figure 14: Effect of work-pool size on the quality factor and the kind of machines providing accepted work-units (i.e. CPU clock rate). The several tests last about 12 hours and runs on 24 nodes (12 nodes with 400 MHz CPU clock rate and 12 nodes with 1 GHz CPU clock rate).

4 Conclusion

In this report, we migrate a protein folding simulation using CHARMM from PC clusters to the United Devices MetaProcessor - a commercial platform for widely distributed computing on Intra- and Internet. We also look

at the performance characterization of the migrated application on the heterogenous platform.
The performance evaluation with a 56 residue protein (the *src-SH3* domain) shows that there are interesting trade-offs in work-unit size and work-pool size which even affect the quality of the results in the folding simulation. Larger work-unit sizes provide better quality factors with a smaller number of work-units, but result in longer turn-around times and less parallelism. We prove that the quality of the protein folding simulation is sensitive to the platform configuration, i.e. the number of workers and the clock rates of the workers. In particular, the proper choice of work-pool size for a given platform size (number of workers) can improve the final quality factor by 20%-30% with the same total amount of computation performed. We demonstrate that with larger work-pool sizes the active contribution of slower machines to the entire folding process become more significant and the computation becomes less susceptible to local minima. But choosing work-pools that are too large will adversely affect the convergence of the quality factor.
The experience with our research prototype indicates that the application of protein folding is robust against the heterogeneous environments and the limited communication capabilities of desktop grids. In fact, all compute nodes involved do provide contributions to the progress of the protein folding simulation, despite their different CPU clock rates and slower network interconnections. Based on our performance study presented in this paper, we can conclude that the platform heterogeneity positively affects the quality of protein folding simulations, because of its positive influence on task generations and scheduling. The differences in clock rates and network speeds seem to contribute additional randomness to the search for an optimal conformation by simulated annealing and help to get better results.
In short, we can say that protein folding calculations with CHARMM are well suitable candidates for widely distributed computation on a desktop computational grid. The suggested software system can make a large amount of nearly free compute cycles available to computational biologists in need of a large amount of additional compute power for protein folding calculations for their research.

References

[1] D. Anderson et al. United Devices - Building the worlds largest computer, one computer at a time. http://www.ud.com.

[2] J. Basney and M. Livny. Deploying a High Throughput Computing Cluster. In Rajkumar Buyya Editor, editor, *High Performance Cluster Computing*, volume 1, chapter 5. Prentice Hall PTR, 1999.

[3] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In *Proc. of the International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS)*, Santa Clara, California, Jun 2000.

[4] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[5] A. Chien et al. Entropia - High Performance Internet Computing. http://www.entropia.com.

[6] C. Clementi, H. Nymeyer, and J. N. Onuchic. Topological and Energetic Factors: What Determines the Structural Details of the Transition State Ensemble and “en-route” Intermediates for Protein Folding? An Investigation for Small Globular Proteins. *J. Mol. Biol.*, 298:937, 2000.

[7] distributed.net Project Page. http://distributed.net/.

[8] P. Ferrara, J. Apostolakis, and Caflisch A. Evaluation of a fast implicit solvent model for Molecular Dynamics Simulations. *Proteins*, 46(1):24–33, 2002.

[9] Folding@home Project Page. http://folding.stanford.edu.

[10] I. Foster et al. Grid Physics Network. http://www.griphyn.org.

[11] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

[12] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Intl J. Supercomputer Applications*, 15(3), 2001.

[13] J. Gsponer and A. Caflisch. Role of Native Topology Investigated by Multiple Unfolding Simulations of Four SH3 Domains. *J. Mol. Biol.*, 309:285–298, 2001.

[14] J Gsponer and A. Caflisch. Role of Native Topology Investigated by Multiple Unfolding Simulations of four SH3 Domains. *J. Mol. Biol.*, 309(1):285–98, 2001.

[15] J. Gsponer and A. Caflisch. Molecular Dynamics Simulations of Protein Folding from the Transition State. *Proc Natl Acad Sci U S A.*, 99(10):6719–24, May 2002.

[16] W. Hoschek. A Database for Dynamic Distributed Content and its Application for Service and Resource Discovery. In *Proc. of Int. IEEE Symposium on Parallel and Distributed Computing (ISPDC 2002)*, Iasi, Romania, Jul 2002.

[17] M. Karaul. *Metacomputing and Resource Allocation on the World Wide Web*. PhD thesis, New York University, May 1998.

[18] A.D. MacKerell Jr. and et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[19] J. C. Martinez and L. Serrano. The Folding Transition State between SH3 Domains is Conformationally Restricted and Evolutionarily Conserved. *Nat. Struct. Biol.*, 6:1010, 1999.

[20] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M. Humphrey, A. Fox, A. Grimshaw, and C. Brooks. Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion. In *Proc. of the 10th IEEE Symposium on High-Performance Distributed Computing (HPDC-10)*, San Francisco, California, Aug 2001.

[21] J.G.B. Northey, A.A. Di Nardo, and A.R. Davidson. Hydrophobic Core Packing in the SH3 Domain Folding Transition State. *Nature Structural Biology*, 9(2):126–130, Feb 2002.

[22] D. S. Riddle, V. P. Grantcharova, J. V. Santiago, E. Alm, I. Ruczinski, and D. Baker. Experiment and Theory Highlight Role of Native State Topology in SH3 Folding. *Nat. Struct. Biol.*, 6:1016, 1999.

[23] Kyung Dong Ryu. *Exploiting Idle Cycles in Networks of Workstations*. PhD thesis, University of Maryland, 2001.

[24] SETI@Home: Search for Extraterrestrial Intelligence at Home. http://setiathome.ssl.berkeley.edu/.

[25] M.R. Shirts and V.S. Pande. Mathematical Analysis of Coupled Parallel Simulations. *Phys Rev Lett.*, 86(22):4983–7, May 2001.

[26] M.R. Shirts and V.S. Pande. Mathematical Analysis of Coupled Parallel Simulations. *Phys Rev Lett*, 86(22):4983–7, May 2001.

[27] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney. File and Object Replication in Data Grids. In *Proc. of the 10th IEEE Symposium on High Performance and Distributed Computing (HPDC-10)*, San Francisco, California, Aug 2001.

[28] M. Taufer, T. Stricker, and R. Weber. Inverting Middleware Framework: a Framework for Performance Analysis of Distributed OLAP Benchmarks on Clusters of PCs by Filtering and Abstracting Low Level Resource Usage. Technical Report 367, Institute of Computer Systems, ETH Zurich, Switzerland, Mar 2002.

[29] M. Taufer, T. Stricker, and R. Weber. Scalability and Resource Usage of an OLAP Benchmark on Clusters of PCs. In *Proc. of SPAA 2002, Fourteenth ACM Symposium on Parallel Algorithms and Architectures*, Winnipeg, Manitoba, Canada, Aug 2002.

[30] B. Uk. Migration of the Molecular Dynamics Application CHARMM to the Widely Distributed Computing Platform of United Devices. Diploma Thesis, ETH Zurich, Switzerland, 2002.

[31] United Devices, Inc. Edge Distributed Computing with the MetaProcessor Platform, 2001. http://www.ud.com/products/documentation/.

[32] United Devices, Inc. MetaProcessor Platform, Version 2.1 Application Developer's Guide, 2001.

[33] M. Vendruscolo, C.M. Paci, E.and Dobson, and M. Karplus. Three Key Residues form a Critical Contact Network in a Protein Folding Transition State. *Nature*, 409(6820):641–5, Feb 2001.

[34] B. Zagrovic, C.D. Snow, S. Khaliq, M.R. Shirts, and V.S. Pande. Native-like Structure in the Unfolded Ensamble of Small Proteins. *J. Mol. Biol.*, 323:153–164, 2002.

[35] B. Zagrovic, E. Sorin, and V. Pande. Beta Hairpin Folding Simulations in Atomistic Detail Using an Implicit Solvent Model. *Journal of Molecular Biology*, 2001.