TITLE PAGE

Report Title:
Automatic Transformation of MPI Programs to Asynchronous, Graph-Driven Form
Final Report
Reporting Start Data: January 15, 2010
Reporting End Data: January 14, 2013
Principal Authors: Scott B. Baden, John H. Weare and Eric J. Bylaska
Date: April 28, 2013
DOE Award number: DE-SC0003443

Name and address of submitting organization:
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093

Scott Baden's address:
UC San Diego
Computer Science and Engineering Department
9500 Gilman Drive, La Jolla, CA 92093-0404

John Weare's address:
UC San Diego
Department of Chemistry and Biochemistry
9500 Gilman Drive, La Jolla, CA 92093-0303

Eric J Bylaska's address:
Environmental Molecular Sciences Laboratory
Pacific Northwest National Laboratory
P.O. Box 999
Richland, Washington 99352

DISCLAIMER

“This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.”

ABSTRACT

The goals of this project are to develop new, scalable, high-fidelity algorithms for atomic-level simulations and program transformations that automatically restructure existing applications, enabling them to scale forward to Petascale systems and beyond. The techniques enable legacy MPI application code to exploit greater parallelism though increased latency hiding and improved workload assignment. The techniques were successfully demonstrated on high-end scalable systems located at DOE laboratories. Besides the automatic MPI program transformations efforts, the project also developed several new scalable algorithms for ab-initio molecular dynamics, including new massively parallel algorithms for hybrid DFT and new parallel in time algorithms for molecular dynamics and ab-initio molecular dynamics. These algorithms were shown to scale to very large number of cores, and they were designed to work in the latency hiding framework developed in this project. The effectiveness of the developments was enhanced by the direct application to real grand challenge simulation problems covering a wide range of technologically important applications, time scales and accuracies. These included the simulation of the electronic structure of mineral/fluid interfaces, the very accurate simulation of chemical reactions in microsolvated environments, and the simulation of chemical behavior in very large enzyme reactions.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The availability of massively parallel supercomputers has supported the dramatic expansion of atomic level simulations (molecular dynamics, and *ab-initio* molecular dynamics) to new classes of scientific and engineering models that more effectively capture the full complexity of real systems. With continuing performance improvement this technology it is expected that simulation will take a more crucial role in areas such as the development of new high performance materials (batteries, magnetic recording devices), control and prevention of disease, drug discovery and the development of secure waste isolation technology, e.g., CO_2 sequestration and nuclear waste isolation [7,11,12]. However, despite impressive progress, the time scales, model system sizes (particle numbers), and model accuracies that are necessary to capture the complex behaviors encountered in many of these problem areas are still not accessible to predictive simulation analysis. Exascale systems could enable further growth in applying simulation methods. However, performance improvements of such simulations are limited by the costs of data motion as we increase the number of cores ($\sim 10^4$). Thus, managing these data motion costs is *the* significant impediment to further development of this critical tool for technological discovery.

The implementation techniques required to run at scale for these large computational problems introduce an unprecedented expansion in performance programming detail. Since present day compiler technology cannot perform the required optimizations, the task of masking communication delays entails significant, intrusive performance programming, challenging even the expert programmer. Moreover, with no assurance that current software techniques will continue to be effective in the face of rapid technological evolution, performance robustness will be troublesome.

Too meet this software need, we developed Bamboo, a source-to-source translator that transforms an MPI program into a semantically equivalent task precedence graph formulation, which automatically masks communication with available computation. Bamboo is a custom source-to-source translator that recognizes MPI calls, in effect treating the MPI entries as primitive language objects. We implemented the translator with the ROSE compiler framework [17]. Bamboo re-engineers MPI code to run as a data-driven program (like coarse-grain dataflow [16]) running under the control of runtime services which schedule tasks according to the flow of data and the availability of processing resources. The run time services rely on virtualization [19] to pipeline communication and computation.

We validated Bamboo against two important application motifs: Jacobi's method for solving Poisson's equation in 3 dimensions (structured grid) and Cannon's Matrix Multiplication Algorithm (dense linear algebra). We ran on up to 98304 processor cores of NERSC's Hopper system and demonstrated that Bamboo-generated code improved performance by masking communication delays. Performance was competitive with that of carefully optimized MPI source that was manually restructured with classic split-phase coding. Indeed, Bamboo avoids the need for classic split-phase code that complicates communication tolerant applications. Whereas classic split- phase code embeds the communication tolerance strategy into the application, Bamboo factors the communication overlap strategy out of the user's code, improving code maintainability and performance robustness.

Several new low latency parallel algorithms were implemented for atomic-level simulations, including massively parallel algorithms for hybrid DFT, parallel in time algorithms for molecular dynamics and ab-initio molecular dynamics, local density of states free energy calculations based on ab-initio molecular dynamics, and new algorithms for dynamic mean field theory. These algorithms were designed to work with the latency hiding Bamboo framework developed in this project.

The developments of the computer science and implementation side of the program were rigorously guided by application of the methods in the simulations in real technological applications of high importance. These included: the 1st principle simulation of the dynamics of a fluid/mineral fluid interface and the demonstration of the stability of the hydrated mineral surface stability (supported by the development of

efficient hybrid DFT implementations; the calculation of the microsolvation of an acid species leading to the demonstration of the need to simulate these problems using very high level accuracy in the electronic structure force calculation (supported by the development of the new parallel in time algorithm); and the application of QM/MM methods to bioenzyme reactions illustrating the effectiveness of the application accurate of the application of high level quantum chemistry methods to interpreting the detailed chemistry drug discovery targets.

REPORT DETAILS

Experimental Methods

Computational results were obtained on two platforms. The first platform was *Hopper,* a Cray XE-6 located at NERSC, with 153,216 cores. The second platform is *Chinook*, a supercluster with computation nodes comprising dual-socket, quad-core AMD processors and 32 GB of memory per node.

Results and Discussions

We first discuss our results in restructuring MPI application code to hide communication. We then discuss our results with ab-initio Molecular Dynamics.

Restructuring MPI applications to hide communication

Results for Bamboo were obtained on Hopper. All source code was compiled using the *CC* wrapper, an MPI interface to the C++ compiler, with the following optimization options:
-O3 –ffastmath. The wrapper was set up to use the GNU compiler suite (GCC 4.6.1). High performance matrix multiply (dgemm) was supplied by ACML version 4.4.0.

We validated Bamboo against applications from two well-known HPC computational motifs: a 3D Jacobi iterative solver (which we will refer to as 3D Jacobi) and dense matrix multiplication. For the latter, we implemented not only the classic Cannon algorithm, but also the communication-avoiding variant, which targets small matrices [4].

Jacobi. For Jacobi, we conducted a strong scaling study, maintaining a fixed problem size as we increase the number of processors. The results, shown in Fig. 1, demonstrate that Bamboo was able to restructure the MPI application to use improve performance by masking communication. For example, on 96K (98304) cores, the restructuring improved performance by a factor of 1.27, hiding 52% of the communication delays. The improvement in performance tracks the relative fraction of communication. Thus, at smaller numbers of cores, the improvements are also smaller, 7% on 12K (12288) cores.

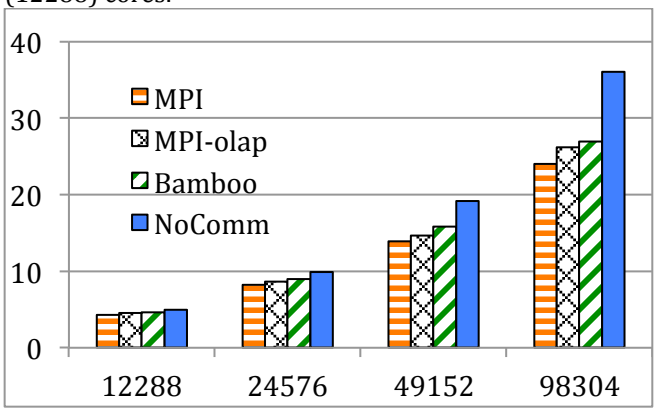

Fig. 1. Results with Jacobi3D, a 3D iterative solver. In this strong scaling study, the problem size is fixed at 3072^3. The input code is **MPI**, and appears as the leftmost bar in each series (12K through 96K cores). The rightmost bar (**NoComm**) corresponds to the ideal case when all MPI communication has been shut off; it is an upper bound on performance. **MPI-olap** corresponds to a hand coded variants that overlaps performance. Bamboo's performance (**Bamboo)** always exceeds that of **MPI-olap**.

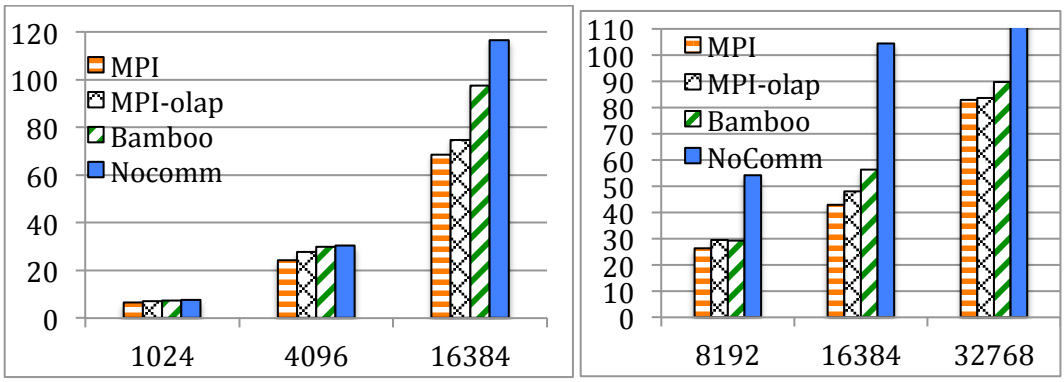

Fig. 2. Weak scaling studies of matrix multiplication (Cannon's algorithm). **Left.** The conventional (2D) algorithm. **T**he matrix size was 52000^2 on 1024 cores and the amount of computation work per core (N^3/P) remains constant. The performance of the original MPI code (**MPI**) appears as the left-most bar in each series. The rightmost bar (**NoComm**) corresponds to the ideal case when all MPI communication has been shut off and is an upper bound on performance. **Bamboo** (2nd bar from the right) improved performance by 42% on 16K cores (S=1.42) and it outperforms the handed coded variant (**MPI-olap**, 2nd from the left). **Right.** Results for the 2.5D communication avoiding variant. The matrix size was 20608^2 on 8192 cores. As with the 2D algorithm, Bamboo hid communication and outperformed the hand-coded variant. There was a slight performance inversion on 8K cores, which vanished at larger scales

Cannon's matrix multiplication algorithm. For this application, we conducted a weak scaling study. Since the application delivers a high fraction of peak performance on a single core (88%), strong scaling is of limited value. We used square matrices of size of 52000^2. As shown in Fig. 2, communication costs increase with the number of cores. This is consistent with the performance model for the application. The number of communication steps grows as $\sqrt{P}$. Since the wall clock time spent in *dgemm* remains constant, and the size of the local sub-matrices A and B grows as $P^{2/3}$, the communication to computation ratio grows as $P^{1/6}$. In fact, the observed growth in communication is a bit higher as we've ignored the increase in message starts, which also grow as $\sqrt{P}$. Under these conditions of growing communication costs, Bamboo speeds up the original MPI code by a factor of 1.15 to 1.42, eliminating the majority of the communication overhead.

2.5D Cannon's matrix multiplication algorithm. The 2.5D "communication avoiding" variant of matrix multiplication algorithm [4] targets small matrices. These small matrices arise in the structure calculations targeted by this project [5], [6]. The 2.5D algorithm is interesting for two reasons. First, small matrix products incur high communication costs relative to computation, especially at large scales, which stress Bamboo's ability to mask communication delays. Second, the 2.5D algorithm introduces two new communication patterns: broadcast and reduction. Supporting these new patterns broadens the scope of Bamboo. We conducted a weak scaling study on 8K, 16K and 32K cores (Fig 2). We chose problem sizes that enabled us to demonstrate the algorithmic benefit of data replication employed by the 2.5D algorithm. Bamboo improved performance by up to 31% (on 16K cores).

In all of the applications, we compared the performance of Bamboo-generated code with carefully hand coded variants that hide communication delays. These codes require expert knowledge to implement complicated split phase strategies for overlapping communication with computation. In all cases, Bamboo's generated code usually outperformed these variants. In one case the hand-coded variant slightly outperforms the Bamboo generated code, though this occurred at a smaller scale. Bamboo avoided the need for costly code restructuring to reformulate the algorithm to hide communication. Only a few lines of pragmas needed to be added (see Fig. 3, as discussed in the next section), and in some cases a few lines of code were re-arranged. The MPI code retains it original structure, and a standard C/C++ compiler ignores Bamboo pragmas.

The translator.
The Bamboo translator applies a set of transformations to MPI source code that capture MPI library semantics to (1) analyze dependence information expressed via MPI calls and (2) automatically generate a semantically equivalent task precedence graph representation, which can run asynchronously and automatically hide communication delays.

An example MPI code with Bamboo annotations appears as Fig 3.

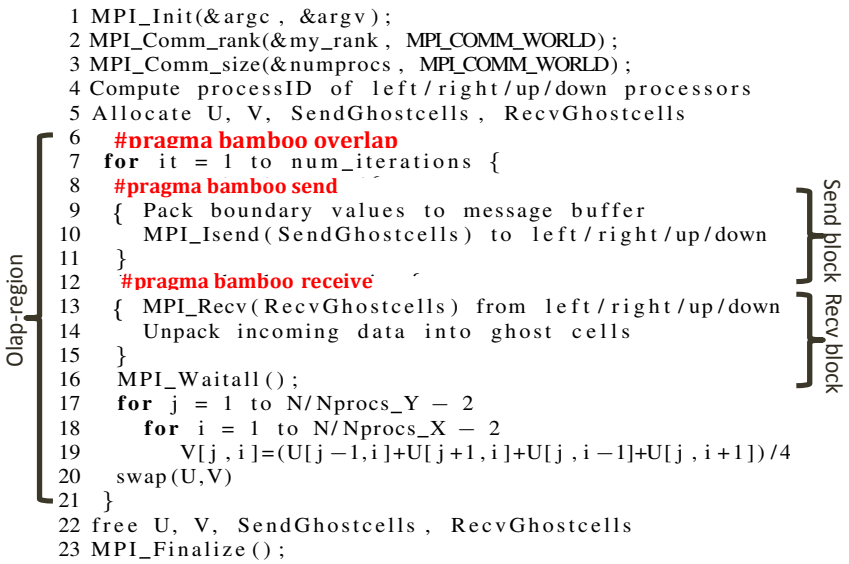

Fig. 3: Annotated MPI program for 2DJacobi. Some code has been omitted for the purposes of clarity. To save space, we employ non-standard C syntax for send and receive regions: an opening curly brace appears on the same line as the corresponding pragma.

We will refer to the task precedence graph as a *task graph*. This program runs under the control of Bamboo's run time services, which interpret MPI program execution in terms of the task graph. Bamboo requires some additional knowledge about the input source, that comes in the form of programmer annotations that we discuss next

A Bamboo program is a legal MPI program, augmented with one or more *olap-regions*. An *olap-region* is a section of code containing communication to be overlapped with computation, both located within the same *olap-region.* Lines 6-21 in Fig. 3 show an example of an *olap-region*. Bamboo can recognize opportunities to overlap communication with computation within *olap-regions* only. It preserves the execution order of *olap-regions*, which run sequentially, one after the other.

Each olap-region contains at least 2 *communication blocks*, plus a single computational block that depends upon the completion of communication. The computational block is optional and may contain other *olap-regions*. The common case is for computational blocks to be present, as in the applications we studied. There are two kinds of communication blocks: *send* and *receive*. Communication blocks specify a partial ordering of communication operations at the granularity of a block, including associated statements that set up arguments for the communication routines, e.g. establish a destination process, pack and unpack message buffers. While the statements within each block are executed in order, the totality of the statements contained within all the send blocks are independent of the totality of statements contained within all the receive blocks. This partial ordering enables Bamboo to reorder send and receive blocks. However, Bamboo will not reorder blocks of the same type.

Bamboo currently handles 6 fundamental message passing primitives inside communication regions: blocking send and receive (*Send* and *Recv*), *asynchronous* variants (*iSend* and *iRecv*) and synchronization (*Wait* and *Waitall*). *Wait* and *Waitall* specify synchronization points; their semantics are preserved by Bamboo. We note that the above restrictions do not rule out the expression of certain communication patterns. Rather, they provide a methodology, that is, guidelines for inserting Bamboo annotations that ensures code correctness.

Revisiting the code in Fig. 2, a single *send* pragma at line (8) groups four *MPI Isend* invocations together. The *iSend()* at (10) consumes the data produced by code at (9) that linearizes data into a buffer. Similarly, the *receive* pragma (12) groups four *MPI Recv* calls. The unpacking code inside the receive block at (14) consumes data delivered by the *Recv()* at (13). By grouping the four *iSends* and four *Recvs* into a *send* and a *receive* block, respectively, the user informs Bamboo that the sending and the receiving of ghost cells are two independent activities. Once the two communication blocks complete, the computations to update the local mesh at lines (17)-(19) may execute.

Bamboo generates source code to produce a new program represented as a task graph that runs under a data-flow like execution model. This program relies on a task graph library, *Tarragon* [14,15], to define, manage, and execute task precedence graphs, which are objects of type *TaskGraph*. Bamboo expects the task graph library to export this abstract base `class`, and generates concrete instances as needed. The nodes of a TaskGraph correspond to tasks to be executed, which are objects of type *Task*. The edges correspond to data dependencies among tasks. The taskgraph library's runtime system interprets these dependencies as communication channels and tasks communicate via active messages.

Each MPI process will be effectively transformed into a set of tasks that are executed by a group of *worker threads*. The library supports task execution via one or more *service threads*, including scheduling. In order to improve the success of hiding latency, we create more tasks than processing cores; as noted by others, virtualization is important in hiding latency [19]. Worker threads run to completion but the tasks do not. The combination of task virtualization and run to completion behavior requires that we take measures to avoid deadlock. To this end, a task does not explicitly wait on communication. The effect is to control when data becomes visible to the task.

The underlying execution model of Bamboo is like dataflow, but with the provision for task *state.* Tasks will generally alternate between the running and suspended states, for example, in an iterative method. When a task finishes computing, it moves data along output edges and then it suspends, at which point it is waiting on arriving data. The runtime services recognize task state and will swap in a runnable task, which has met the conditions of its *firing rule.* A task's firing rule is simple: the task becomes runnable once it receives all required messages. When a task runs-or becomes runnable-all its input data are guaranteed to be available. Any newly-arriving data will not be visible to the task until after it has suspended; the task has received all the data it needs to run, and any newly arriving data will be visible the next time the task runs. The resultant delays in data visibility are fundamentally different in a task graph program and an MPI program and required code motion transformations that are performed by the translator [13].

We used the ROSE compiler infrastructure [17] to implement Bamboo, which includes the EDG front-end used to parse standard C source. This front-end generates an Intermediate Representation (IR), which is an in-memory Abstract Syntax Tree (AST). Since EDG considers the MPI calls as ordinary C function calls, we built a custom module sitting between the front and middle ends to extract information about the parameters passed to MPI functions. The four main modules of Bamboo built on top of ROSE's middle-end modify the IR to create a new form that conforms to the Tarragon API. The *annotation handler* extracts information from each Bamboo directive along with the corresponding location within the IR; the *analyzer* and *transformer* modify the IR to conform to the Tarragon model and the *optimizer* applies various transformations to

improve the quality of the generated source code. Finally, the back-end completes the translation process.

The communication avoiding multiplication application not only uses point-to-point message passing calls,, but collective calls as well. We currently hand translate the collective calls into their point-to-point counterparts, and automated translation is in progress.

New algorithms for ab-initio molecular dynamics

New massively parallel algorithms for hybrid DFT. We have developed a scalable parallel algorithm, called the incomplete butterfly, that overcomes inefficiencies associated with calculating the exact exchange interaction from Hartree-Fock theory and improves implementation of a hybrid exchange-correlation functionals in density functional theory (DFT) – a quantum mechanical modeling method. DFT is used to calculate the properties of solid state and molecular systems, and predicts structures, properties, and reactivities for a wide variety of systems important to solar, hydrogen storage, catalytic, nuclear, and environmental remediation technologies. But it frequently gives inaccurate predictions. In this work parallel algorithms to distribute data and replicated data to overcome bottlenecks associated with calculating the exact exchange term were developed. These algorithms were implemented in a module in PNNL's NWChem high-performance chemistry software package that is used to calculate surfaces and condensed phase systems. The developments were then used to perform hybrid-DFT calculations on several surface and condensed phase systems.

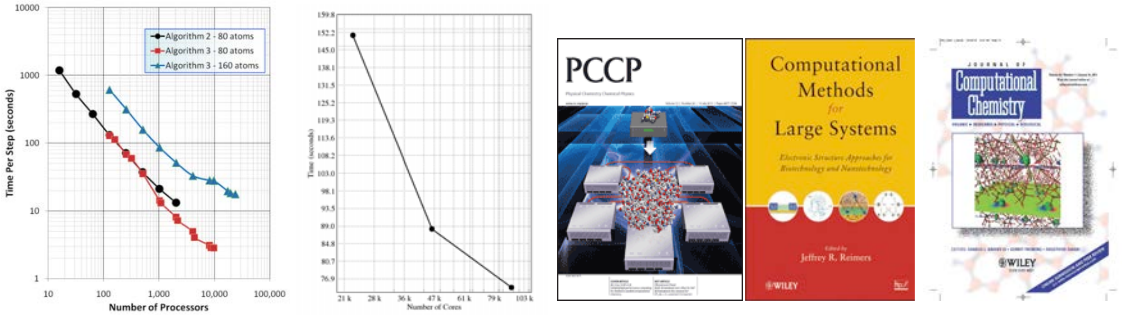

Fig. 4 (left) Exact exchange timings – 80 atom cell of hematite (cutoff energy=100Ry). These calculations were performed on the Franklin Cray-XT4 computer system at NERSC. (Middle) Exact exchange timings – 576 atom cell of water (cutoff energy=100Ry). These calculations were performed on the Hopper Cray-XE6 computer system at NERSC. (Right). Covers that highlighted our work on hybrid DFT.

The algorithms scaled up to 25,000 CPUs for a modest-sized problem and can be applied to a wide variety of plane wave programs. Recently they were demonstrated to scale to 100,000 CPUs on the Hopper computer system at NERSC (Fig. 4). This work was featured on the [9] of the *Journal of Computational Chemistry* [8]. This work was also featured in a cover article and a book chapter [10]. These covers are shown in Fig. 4.

Our initial translation of this algorithm with Bamboo did not show improvement. We also saw this problem when translating the 3D FFT kernel. We are currently working on Bamboo as well as Tarragon to improve the performance of the 3D FFT and the incomplete buttery kernels. This work has contributed to three publications [8-10] that were featured as covers.

New parallel in time algorithms for molecular dynamics and ab initio molecular dynamics. Several parallel in time algorithms have been developed and tested for molecular dynamics (MD) and *ab-initio* molecular dynamics (AIMD) simulations. These algorithms transform standard forward ODE solvers (i.e. $x_{i+1} = f_i(x_i)$) into fixed point root problems, $F(X) = 0$, which are then solved using a variety of optimization techniques, including quasi-Newton and preconditioned quasi-Newton root finding methods. To parallelize the algorithms the evaluation of F(X) was done in parallel. We also showed that solving the root finding problem by preconditioned fixed point algorithm leads to the parareal algorithm of Lions and Maday[2].

Very encouraging results were obtained with the application of these algorithms to AIMD simulations based on the computationally expensive MP2 electronic structure method to HCl+4H2O clusters. We demonstrated that our parallel in time algorithms were able to provide theoretical speedups up to 30 and actual speedups to 14.3. We were able to obtain an effective cost per time-step of less than 10 and 30 seconds in our AIMD simulations at the MP2/6-31G* and MP2/6-311+G* levels respectively. With these timings, meaningful dynamic simulations can be done. We found that the preconditioned quasi-Newton algorithm converged in fewer iterations. However, the measured speedup was highly sensitive to the cost of preconditioner relative to the cost of the AIMD force. At the highest level (MP2/6-311+G*) the parallel in time algorithms with a HF preconditioner scaled to 100 processors. When the smaller the 6-31G* basis was used it was found that while the HF preconditioner greatly improved the theoretical speedup (from 8.8 to 30) the best possible actual speedup (8.2) was obtained without HF preconditioning.

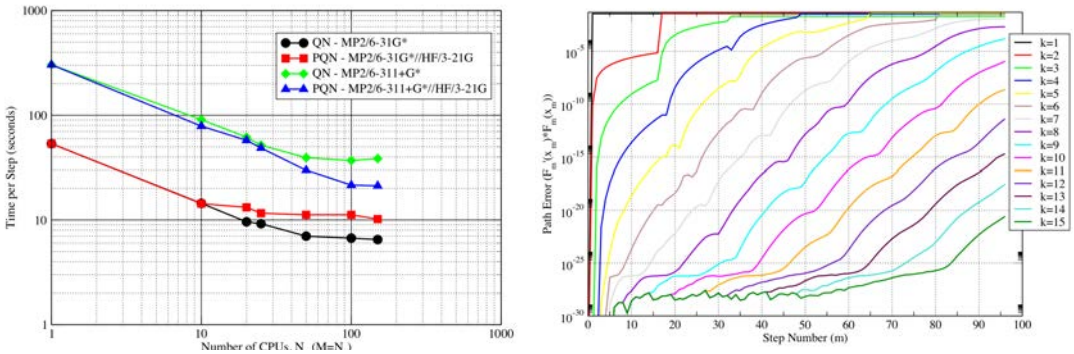

Fig. 5. Timings for HCl+4H2O parallel in time AIMD MP2/6-31G and MP2/6-311+G* simulations (preconditioner- HF/3-21G) simulations (△t = 5.0 a.u., and M = Np). (right)* Path error for each sweep of a quasi-Newton parallel in time algorithm of a Stillinger-Weber 1000 Si atom simulation (△t=20.0 au, initial T=500K).

As an additional test of the performance of the algorithms we ran the HCl+4H2O clusters with an even larger basis set (6-311++G(2d,2p) on the PNNL Chinook platform (Fig. 5). Using NWChem the force parallelization saturates at approximately 64 processors. With this number of processors and this basis the single step force calculation requires 32 second per step. By extending these calculations with our parallel in time algorithms we were able to run the problem on 2560 processors and obtain a step time of 6.9 seconds. With this step time realistic dynamical simulations are now possible at this level of theory. This work has been submitted to publication to the *Journal of Chemical Physics* and is currently under review.

We are currently finishing up work to extend this approach to use of algorithms based on implicit integrators. This type of approach might be able to resolve the multiple time-scales in a hierarchal fashion as in multigrid solvers and by doing so could overcome the broadcast bottleneck since only a fraction, △t/△T, of the geometries need to be propagated to the finest level. The use of implicit integrators that can produce a significant amount of energy drift when simulating long times is a drawback of this approach. However, these problems can be overcome by using explicit integrators at the finest level of time integration. Publication of this extended work is in preparation.

Electronic structure of highly correlated systems. In the last year of the project a student from UCSD (Y. Chen) and PNNL have developed a new algorithm to model electron transfer in highly correlated materials (These materials are essential components in many important applications, e.g., catalysis, oxidation and reduction, respiration, etc.). This algorithm uses a local (or atomic) density of states as a collective variable, ***S***, in a free energy simulation based on Metadynamics (MTD) [3]. MTD is a non-equilibrium molecular dynamics method that accelerates the sampling of the multidimensional free energy surfaces of chemical reactions on a simulation time scale that is easily accessible on modern computers. This is achieved by adding an *external time-dependent bias potential* which is a function of the *collective variables*, ***S***, to the Hamiltonian of the system. The bias potential discourages the system from sampling previously visited values of ***S*** (i.e., encourages the system to explore new values of ***S***),

and its accumulation in low energy wells allows the system to cross energy barriers much more quickly than would occur in standard molecular dynamics. This work is included in the latest release of NWChem, version 6.4.

We are also developing new application software implementing dynamical mean field theory (LDA+DMFT). This approach is designed to provide a manageable solution method for the electronic structure problem for highly correlated materials. These materials (e.g., transition metal oxides) are essential components in many important applications (e.g., catalysis, oxidation and reduction, respiration, etc.) The lack of efficient methods of solution is a critical need for the development of new high performance materials [1]. The DMFT approximation produces a highly heterogeneous computational environment. The principal bottleneck is the solution to the electronic structure problem for single highly correlated elements (e.g. Fe^{3+}) interacting with an independent particle band (LDA or DFT). Our objective is to develop a preliminary parallel impurity solver, which we will incorporate in our mature DFT codes. UCSD and PNNL are currently working to complete the development of new parallel algorithms for dynamical mean field theory (LDA+DMFT).

Application progress

The interaction between the application side and the computer science/implementation effort of the program is critical to insure that the developments lead to actual improvement in simulation performance for problem that are important to the DOE mission. In the following we highlight some of the progress that was partially supported by this program during this period.

Calculation of electronic structure of the mineral/fluid interface of high correlated materials:
Iron oxide (hematite (α-Fe_2O_3)) and oxihydroxide minerals (goethite FeOOH) commonly occur in natural environments and are associated with natural and biochemical oxidation/reduction processes, biomineral respiration, etc.. In addition, hematite is easily synthesized and because of its favorable electronic structure properties is an attractive candidate for material applications such a solar energy conversion materials, catalysis, gas sensors etc. A long-term goal of this program is to evaluate electron transport/transfer processes between transition metal oxides and reduced metal ions (e.g., Fe^{2+}(aq)/Fe^{3+}(crys.) electron exchange) in the interface region. Specific targets for this program period were are the calculation and optimization of surface interface reconstructions using various higher levels of electronic structure methods, particularly the methods discussed above for highly correlated materials. The breaking of symmetry at the mineral surface, the disorder of the interface regions, the extreme dynamics of the fluid phase, and the highly correlated nature of the solid phase create a difficult modeling problem for condensed matter theories. The software tools developed in the program and discussed above were essential to the progress during the period of this project. Fig. 6 illustrates the results of a 1st principle dynamics calculation of the fluid/mineral interface.

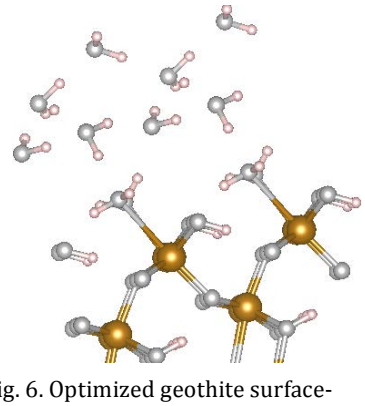

Fig. 6. Optimized geothite surface-water interface. Gold ; iron: White; oxygen: Pink; hydrogen.

Simulation of reaction dynamics in microsolvated environments (e.g., aerosols, fluid interfaces): $HCl+4H_2O$ clusters are a well-studied example of microsolvation, and they are thought to relate to acid chemistry at the air/water interface. As shown Fig. 7 the parallel-in-time simulation of the proton shuttling very little energy drift was seen in both simulations up to 1.2 and 2.5 ps maintaining energy conservation. Prior AIMD simulations (at the less accurate DFT+GGA level) of this "smallest acid droplet" have shown that adding a fourth water to $HCl\text{-}3H_2O$ cluster results in a spontaneous dissociation of the proton into the water. In our calculations we found that the initial potential energy of the cluster was very sensitive to the

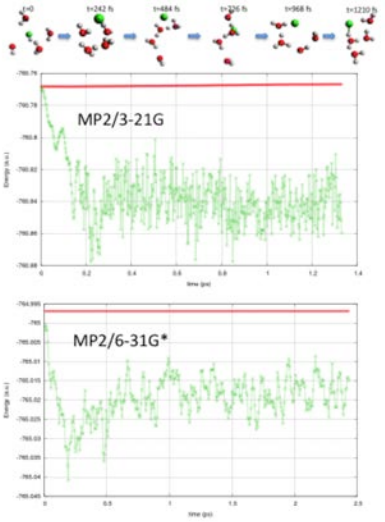

Fig. 7. MP2 dynamics of the HCl-4 H_2O system. Note the conservation of the total energy (red line).

basis set size, less stable at the MP2/3-21G level compared to the MP2/6-31G* level. This lead to almost immediate dissociation when using the smaller basis set. Whereas at the MP2/6-31G* no dissociation was observed up to 2.5 ps. The simulations here provide a very preliminary view of the dynamics at the MP2 level and are meant to illustrate the performance of the algorithms. However, the basis set sensitivity seen in these simulations suggests that more complete calculations at the MP2 level could lead to new interpretations of acid dissociation in microclusters. Fig. 7 (illustrates the dynamics of this system at different levels of approximation (MP2 levels).

Prediction of the reaction mechanisms of important enzyme reactions:

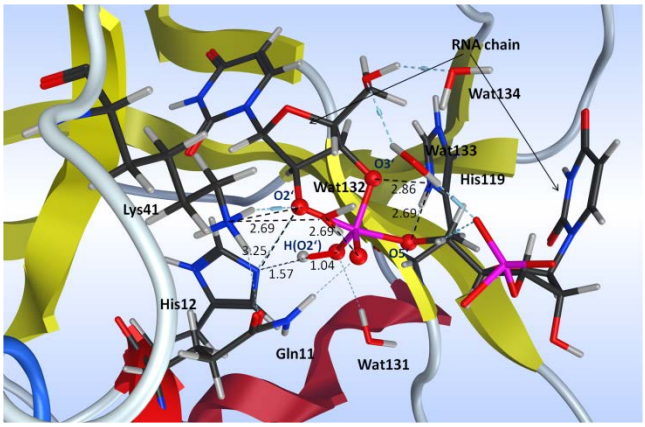

Fig. 8. TPB structure of the stabile intermediate in the active site RNase A. The active site residues, waters and two nucleotides of the RNA trimer are represented by sticks and the TBP phosphorane is depicted as balls and sticks. A zoom-up of the rest of the enzyme is shown as ribbon.

While the general acid/base character of the trans esterification step of the RNase A enzyme reaction is well accepted, the nature and roles of the structures encountered in the transition region are still not well established (even after 60 years of study) and greatly affect the interpretation of reaction rates. Because of the ubiquitous nature of this reaction, it is important to have a correct interpretation of the mechanism. With this objective we have calculated two thermally optimized minimum energy free energy reaction paths based on a thorough analysis of all existing structural data for this enzyme and identified by an unbiased search of reaction coordinates. A highly reliable quantum chemistry method is used to calculate the interactions in the QM region, which includes the catalytic pocket and the reactive part of the substrate (roughly 100 atoms). The remaining enzyme, substrate and solvent are calculated with a molecular mechanics model (30,000 atoms). Together these dynamical calculations pose a very challenging computational environment.

The results support a novel reaction mechanism in the proton initially coordinating the O2´ on the ribose migrates to the non-bridging O1P in the initial part of the reaction path rather than directly to the general base His12. With this transfer the structures in the transition region are associative, barriers are low (endocyclic TS1 with barrier 9.84 kcal/mol and exocyclic TS2 with barrier 7.54 kcals/mol) with a metastable intermediate, see Fig. 8, in agreement with many observations. Since this is an exceedingly well-studied reaction mechanism the still existing controversies over essential points in the interpretation are especially important to resolve.

Conclusion

We have developed a translator, called *Bamboo*, to re-engineer MPI code to tolerate latency. We found that Bamboo improved performance at scale on two important application motifs: structured grid and dense matrix linear algebra. We have validated the premise that, by interpreting an MPI program in terms of data driven execution, we can overlap communication with computation and thereby improve performance. Moreover, performance meets or exceeds that of labor-intensive hand coding, at scale. Bamboo also improved performance of communication avoiding matrix multiplication. The translated code not only avoids communication, but also tolerates what it cannot avoid. We believe that this dual strategy will become more widespread as data motion costs continue to grow.

Though Bamboo relies on programmer annotations to guide the transformation process, our experience is that these annotations are intuitive and modest in number. We target MPI because it is the dominant means of building parallel scalable applications. However, our approach is not limited to MPI and also applies to other data motion libraries such as GasNet [18]. More generally, our technique for treating MPI illustrates an approach for delivering an embedded domain specific language, in which the methods of a library API in effect

become primitive language constructs. In the case of Bamboo, the translator treats the MPI API as an embedded domain specific language. However, Bamboo is more than a means of hiding latency that avoids costly code restructuring. It also serves as an example of the power of semantic level optimization in realizing domain specific optimization.

Bamboo has some limitations, which we are currently addressing. First, it handles only the fundamental communication primitives covering a small subset of MPI. We are adding support, for example, for communicators and for collectives. We plan to translate collectives into their components, i.e. point-to-point primitives, to take advantage of improved finer grained pipelining. Second, we have validated Bamboo against just two application motifs. In the future we plan to look new motifs such as the Fast Fourier Transform that exhibit vastly different communication patterns.

Besides using the Bamboo framework to maximize the overlap between computation and communication in existing algorithms for molecular simulation, new algorithms also need to be designed which reduce the amount data being communicated relative to the computation being done. In this project we have developed several new algorithms for molecular simulation along these lines, including massively parallel algorithms for hybrid DFT, and parallel in time algorithms for molecular dynamics and ab-initio molecular dynamics. These algorithms have the advantage over traditional algorithms for ab-initio molecular dynamics in that the amount data being communicated is significantly less than what is being computed. As expected these algorithms have been shown to scale to very large numbers of cores (e.g. 100K for hybrid DFT), however, the drawback of these types of algorithms is that they contain very complex communication patterns and are highly sensitive to the scheduling and processor layout used. It is expected that as the Bamboo framework becomes more mature will be able to alleviate most of these problems, and these new algorithms will be able to take advantage of the exascale computing.

Presently the application of simulation technology to real technological problems such as drug discovery, development of high performance materials, and enhanced performance of wastes isolation technologies (CO_2 sequestration, nuclear waste cleanup, etc.) is limited because of the chemical complexity, particle number, and force interaction accuracy required for predictive reliability. In this program an effort was made to use the developments of the program in actual simulations of real technological problems. This both rigorously constrained the path of development and provided positive feedback to the computer science and implementation aspects of the program. However, in addition to this there was significant progress in the interpretation of real problems. These included:

- The first time simulation of the dynamics of the liquid mineral surface interface. This has application to problems such as the development of reliable environmental waste isolation strategies and the development of more energy efficient catalytic processes.
- The first time simulation of microsolvation using a high level solution of the electronic structure calculation. These calculations demonstrated of the importance of accurate (and very expensive) force calculation for accurate simulation. This class of simulation is important to the interpretation of many environmental and biochemical problems.
- Calculation of the reactive mechanism for bio enzyme reactions using very high particle number models (including the enzyme, the substrate and the solution) and high accuracy 1st principle force calculations. A novel enzyme reaction mechanism was discovered for a highly studied key phosphoryl transfer reaction.

REFERENCES

[1] *Discovery in Basic Energy Sciences: The Role of Computing at the Extreme Scale*, G. Galli and T. Dunning, Eds., (2010), DOE publication.

[2] J. L. Lions, Y. Maday, and G. Turinici, Comptes Rendus De L Academie Des Sciences Serie I-Mathematique 332, 661 (2001).

[3] A Laio and M Parrinello, *Proc. National Academy of Sciences*, ***99***, pp.12562 (2002).

[4] E. Solomonik and J. Demmel." Communication-optimal parallel 2.5d matrix multiplication and lu factorization algorithms." Tech. Re. UCB/EECS-2011-72, EECS Dept., Univ. Calif., Berkeley, June 2011.

[5] D. Marx and J. Hutter. "Ab-initio molecular dynamics: Theory and implementation." In J. Grotendorst, editor, *Modern Methods and Algorithms of Quantum Chemistry*, NIC, chapter 13, pp. 301–449. Forschungszentrum Jűlich, i edition, 2000. Publicly available at http://www2.fz-juelich.de/nic-series/Volume3/marx.pdf.

[6] E. Bylaska, K. Tsemekhman, N. Govind, and M. Valiev, "Large-scale plane-wave-based density functional theory: Formalism, parallelization, and applications." In J. R. Reimers, Ed., *Computational Methods for Large Systems: Electronic Structure Approaches for Biotechnology and Nanotechnology*. John Wiley and Sons, Inc., 2011.

[7] *Computational Materials Science and Chemistry: Accelerating Discovery and Innovation through Simulation Based Engineering and Science*, DOE Report: (2010).

[8]. Bylaska EJ, KL Tsemekhman, SB Baden, JH Weare, and H Jonsson. 2011. "Parallel implementation of Γ-point pseudopotential plane-wave DFT with exact exchange." *J. Computational Chemistry*, **32**(1):54–69. ***Cover***

[9] WA De Jong, E Bylaska, N. Govind, CL Janssen, K Kowalski, T Müller and R Lindh, R. "Utilizing high performance computing for chemistry: parallel computational chemistry." *Physical Chemistry Chemical Physics*, **12**(26):6896-6920 (2010). ***Cover***

[10] Bylaska, E. J., Tsemekhman, K., Govind, N., and Valiev, M. (2011). "Large-scale plane-wave-based density-functional theory: formalism, parallelization, and applications." In *Computational methods for large systems: electronic structure approaches for biotechnology and nanotechnology.* Wiley, Hoboken, 77-116. ***Cover***

[11] Basic Research Needs for the Geosciences, Grand Challenge (2005).

[12] B. Harmon, K. Kirby, CW McCurdy, "Opportunities for Discovery: Theory and Computation in Basic Energy Science," DOE Office of Science (2005).

[13] T. Nguyen, P. Cicotti, E. Bylaska, D. Quinlan and S. B. Baden. "Bamboo: translating MPI applications to a latency-tolerant, data-driven form," *Proc Intl Conf on High Performance Computing, Networking, Storage and Analysis (SC '12).* IEEE Computer Society Press, Los Alamitos, CA, USA, Article 39 , 11 pages.

[14] Pietro Cicotti and Scott B. Baden. "Latency Hiding and Performance Tuning with Graph-Based Execution." *Proc. First Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM '11)*. IEEE Computer Society, Washington, DC, USA, pp. 28-37 (2011).

[15] Pietro Cicotti, "Tarragon: a Programming Model for Latency-Hiding Scientific Computations." Ph. D. Dissertation, Dept. of Computer Science and Engineering, Univ. of California, San Diego (2011).

[16] R. G. Babb, II. Parallel processing with large-grain data flow technique. *Computer*, 17(7):55–61, July 1984.

[17] D. Quinlan, D. Miller, B. Philip, and M. Schordan. Treating a user- defined parallel library as a domain-specific language. *Proc. 16th international Parallel and Distributed Processing Symp., IPDPS 2002*, Los Alamitos, CA, USA, April 2002. IEEE.

[18] D. Bonachea. Gasnet specification, v1.1. Technical Report CSD-02- 1207, Univ. of California, Lawrence Berkeley Lab., October 2002.
[19] L. V. Kalé. The virtualization model of parallel programming : Runtime optimizations and the state of art. In LACSI 2002, Albuquerque, October 2002.

LIST OF ACRONYMS AND ABBREVIATIONS

MPI: Message Passing Interface
NUMA: Non-uniform Memory Access
MD: Molecular Dynamics
AIMD: *ab-initio* molecular dynamics
MTD: Metadynamics
AST: Abstract Syntax Tree