U.S. DEPARTMENT OF ENERGY

PNNL-15739

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Greenbook Algorithms and Hardware Needs Analysis

WA De Jong
CS Oehmen
DJ Baxter

January 2007

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

This document was printed on recycled paper.
(9/2003)

HPCS-3 Project: Greenbook Algorithms and Hardware Needs Analysis

Executive Summary

This document describes the algorithms, and hardware balance requirements needed to enable the solution of real scientific problems in the DOE core mission areas of environmental and subsurface chemistry, computational and systems biology, and climate science. The MSCF scientific drivers have been outlined in the Greenbook, which is available online at http://mscf.emsl.pnl.gov/docs/greenbook_for_web.pdf . Historically, the primary science driver has been the chemical and the molecular dynamics of the biological science area, whereas the remaining applications in the biological and environmental systems science areas have been occupying a smaller segment of the available hardware resources.

To go from science drivers to hardware balance requirements, the major applications were identified. Major applications on the MSCF resources are low- to high-accuracy electronic structure methods, molecular dynamics, regional climate modeling, subsurface transport, and computational biology. The algorithms of these applications were analyzed to identify the computational kernels in both sequential and parallel execution. This analysis shows that a *balanced* architecture is needed with respect to processor speed, peak flop rate, peak integer operation rate, and memory hierarchy, interprocessor communication, and disk access and storage. A single architecture can satisfy the needs of all of the science areas, although some areas may take greater advantage of certain aspects of the architecture. The table below summarizes the primary hardware needs for the three scientific areas.

Hardware Needs	**B**	**C**	**E**
Memory hierarchy (bandwidth, size and latency)	X	X	X
Peak flops	X	X	X
Peak integer operations	X		
Overlap computation, communication and I/O	X	X	X
Low communication latency	X	X	X
High communication bandwidth		X	
Large memory	X	X	
High I/O bandwidth		X	
Increasing disk storage needs (size)	X*		X*
B = Biology; C = Chemistry; E = Environmental Systems Science * Relatively high bandwidth I/O is needed to store large blocks of data			

The main aspects of the architecture, summarized in this table, will be discussed below in relation to the associated science areas.

Processor Architecture and Memory Hierarchy

Each of the MSCF science areas requires a CPU with high-performance *floating-point* capabilities. Several of the computational areas will also require significant integer capabilities for addressing memory. The percentage of peak performance for the chemical sciences, biological modeling with molecular dynamics methods, and environmental systems is strongly driven by the hierarchical memory access, memory bandwidth, and latency. Large amounts of low latency and high-bandwidth memory are needed to account for the 10 to 100 times increase in data set size envisioned by the scientific community in the next five years. Fast memory in combination with large cache and pre-fetching mechanisms will improve scientific application performance. The amount of memory per processor is tightly coupled to the need for local fast disk space and I/O, with memory being the preferred way to store run-time data. Environmental systems researchers can take full advantage of hardware optimized for the long vectors that are characteristic of the grid-based solution schemes. Several chemistry and biology applications could benefit from vector capabilities, as well. In summary, the science areas have the following processor and memory requirements:

- *a CPU with high sustained floating-point performance; biological sciences need good integer operation (including integer multiply) performance*
- *large memory (with low latency and high bandwidth perceived as important to high sustained floating-point performance)*
- *fast hierarchical memory access (i.e., large cache and pre-fetching, again, perceived as important to high sustained performance).*

Interprocessor Communication

High-performance computing today is synonymous with large numbers of processors. Most algorithms used for the MSCF science areas are tightly coupled and require data to be shared among processors at regular or irregular intervals—requiring efficient interprocessor communication. To obtain efficient communication among a cluster of processors (e.g., a distributed memory machine), an interconnect equipped with high bandwidth and low latency is required. Latency is one of the determining factors for biological molecular dynamics simulations, which require synchronization during every time step. For other computations (e.g., electronic structure computations), high bandwidth is required. The chemical and biological sciences areas will benefit from interconnect capabilities that enable one-sided communication to facilitate the overlap of computational operations with communications, and provide high bandwidth. Environmental systems models tend to communicate in a "nearest neighbor" communication pattern and will benefit from one- to three-dimensional interconnect topologies. Large shared memory processing systems can currently provide low latency and high bandwidth for memory access for small- to medium-sized computing problems. However, existing science problems have demonstrated shortcomings in using current shared memory architectures in terms of the scalability of I/O on single-image operating systems, the complexity and cost of maintaining cache coherency and the physical limitations of the number of simultaneous memory accesses to the same address.

In summary, the science areas require the following for interprocessor communication:

- *high bidirectional bandwidth and low latency*
- *one-sided communication and overlap of communication with computation*
- *exploration of clustered shared-memory architectures with commensurate I/O capability and operating system architecture.*

Disk Storage and Access

Of the science areas discussed in the Greenbook, those related to high-accuracy chemical sciences methods require large, temporary, high-speed local disk storage and rapid I/O capability that can be accessed in a non-blocking or asynchronous fashion. The amount of local disk storage is tightly coupled to available memory; the latter considered the preferred method for storing intermediate data. The environmental systems science area and biological molecular dynamics methodologies require up to hundreds of terabytes of disk space to store associated simulation data. Code performance in the environmental systems science area would improve through the use of efficient parallel I/O to a shared global file system. For example, Climate Resolving Models presently require 12 gigabytes of disk space per simulated day, or more than four terabytes per simulated year. To improve model performance, a higher-resolution grid (one kilometer or less) is needed, which would increase the data-storage requirements to 50 gigabytes and 18 terabytes per day and year, respectively. Thus, a 20-year simulation would produce 360 terabytes of data. The full amount of disk space would not be required at run time, although a multi-terabyte shared global file system would be required to perform these simulations. Subsurface modeling also has similar data requirements as those discussed above. In summary, the algorithm analysis for disk storage and access requires:

- *local high-speed disk storage with non-blocking access (pre-fetch)*
- *a large shared global file system with reasonably fast parallel I/O.*

Algorithms and Hardware Needs Analysis

The remainder of this document is split into three main sections, one for each of the three science areas, biological sciences, chemical sciences, and environmental systems sciences, identified in the Greenbook. In each section the major applications were identified, and the algorithms of these applications were analyzed to determine the computational kernels in both sequential and parallel execution. The algorithmic information was subsequently mapped onto hardware balance requirements. In addition, for each science area guidance on future hardware capacity and resource needs will be provided.

1 Biological Sciences

The science drivers for biology identified in the Greenbook workshop highlight several architectural needs for keeping pace with computational biology. Many of the computational algorithms needed for leading-edge biological science are fundamentally different than those used by computational chemistry, or environmental sciences. However, in many cases the computational architecture support needed to enable the next generation of high-performance biology software *is* similar to other science domains. The following section presents an analysis of benchmark data from representative applications in computational biology indicating three key needs: 1) very large memory space and memory bandwidth, 2) enhanced capacity for integer operations, 3) large globally-mounted filesystem with high I/O bandwidth.

1.1 Algorithmic information

The biology domain areas represented at the Greenbook workshop include molecular dynamics applied to biological molecules, prediction of protein structure from sequence data, building physiological models over many spatial and temporal scales, applying statistical models to experimental pipelines for data analysis, and bioinformatics applications for analyzing sequence data.

Molecular dynamics: *using molecular dynamics to understand protein function*

The use of classical molecular dynamics enables the modeling of atomic motions over time in large biomolecular systems consisting of tens to hundreds of thousands of atoms. Interactions between atoms are described by a force field (AMBER or CHARMM), or parameterized effective pair model. At each time step in the simulation, the forces of each atom in the system need to be evaluated. The sequential kernels involved in this evaluation consist of simple arithmetic operations (square root, exp, sin/cos, etc.). In addition, force fields can be extended with additional terms such as the particle-mesh Ewald (PME) correction for long range electrostatic interactions. These PME corrections require the calculation of Fast Fourier Transforms (FFT) and become the dominant step in the computations.

To improve the time-to-solution, and to enable scientists to study dynamical processes that occur on time scales longer than a nanosecond time, parallelization is essential. Parallel molecular dynamics codes use domain decomposition of the physical simulation volume to distribute the work. For large molecular systems, this distribution of the

atomic data will also reduce the memory requirements, which are generally in the hundreds of Mbytes. Domain decomposition methods utilize the locality of molecular interactions to minimize the interprocessor communication. However, it is this communication between processors, which is taking place at each of the hundreds of thousands simulation time steps, that is the main bottleneck of the calculations. For example, for the calculation of forces and energies in which atoms are involved on neighboring domains assigned to different processors, information from these domains need to be exchanged between processors. In addition, the dynamical nature of these calculations requires a periodic redistribution and associated data communication for atoms that move from one processor's domain to another. Moreover, for heterogeneous molecular systems the computational work is generally not evenly distributed over the processors. Atom and heterogeneity require the use of dynamic load balancing of domains in order for the parallelization to be efficient. In NWChem molecular dynamics simulations use point-to-point onesided asynchronous communication to make the interprocessor communication as efficient as possible.

Monte Carlo, energy minimization: *de novo prediction of protein structure*

Gene information is presented in the form of a sequence of nucleotides using a character alphabet with four elements. This can be converted, using a well-characterized mapping, to a collection of sequences of amino acids. Each of these sequences corresponds to a putative protein - the molecular machinery that drives biological systems. It is commonly believed that one must understand the structure of these putative proteins to understand their functions. The de novo approach to this involves using protein sequence data and chemical properties of the amino acid sequence to find a structure corresponding to a minimum in chemical energy. Because of the size of these systems (the average protein length is on the order of 350 amino acids, and proteins can be thousands of residues in length), one must perform a collection of randomly seeded energy minimization calculations to have any chance at capturing the true structure. Currently, this is being approached in high-performance applications such as Rosetta (Bonneau et al.). The algorithms used in performing this type of calculation involve 1) using Monte Carlo or other random-walk to seed the energy minimization calculation with a starting structure; 2) performing an energy minimization or other scoring function to assess the goodness of the structure. This approach can be implemented without communication between processors, with the exception of control information governing overall progress and task assignments.

Physiological modeling: *Integrative multiscale modeling to understand signaling and regulation*

Among the biological sciences, physiological modeling most resembles conventional computational science. It is generally implemented as a mesh or agent-based domain description on which ordinary differential equations, partial differential equations, or rule-based interactions are calculated. Physiological modeling applications tend to be the most flop-intensive of the computational biology areas, as they frequently require the solution of equations on a physiological domain. In the continuum approach, solution of biological systems is achieved in much the same way that climate modeling and subsurface transport calculations are performed. Real or idealized system is converted to a mesh or grid representation on which theoretically determined sets of equations are

solved. Visualization tools are then used to convey the results to the user. In the discrete approach, agent-based approach can be used to govern the motion and interaction of objects vested with biological meaning. In both approaches, the key bottleneck is *memory access*. State differential equations can rely on many other states, requiring run-time access to essentially a random collection of memory elements. Agent-based approaches generally localize the attributes of an agent according to their spatial relationship at startup. But interactions could conceivably occur between agents anywhere in the system, making it necessary to expose these data fragments to any agent in the simulation. If repeated load-balancing is employed to enforce some degree of locality, memory access will still be the bottleneck because one cannot predict *a priori* how agents will be partitioned to best share the load after movement occurs. Regardless of the implementation, these approaches require substantial post-processing for visualization or feature-extraction, and substantial pre-processing for setting up the mesh or agent-based domain.

Peptide identification from MS data: *Using statistical models to map experimental measurements to protein identity.*

DOE has invested heavily in high-throughput proteomics facilities which analyze the identity of proteins in biological samples using a variety of techniques. In general, the experimental pipeline produces quantitative peak data which gives information about the relative mass to charge ratio of fragments of proteins (peptides). The computational task of determining the amino acid sequence of the peptides is primarily string matching. Calculating the total mass of peptide candidates in database-searching methods such as Polygraph and Sequest requires inner-loop lookups of amino acid weights. Assessing the goodness of the candidate's match to the experimental spectrum involves limited floating-point arithmetic compared to the memory movement and lookaside required.

Sequence alignment and homology detection: *Informatics applied to multi-species systems*

Sequence alignment is the process of determining which regions of two biological sequences are equivalent with statistical confidence. Heuristic methods such as BLAST (NCBI) achieve high speed by sacrificing accuracy with respect to exhaustive methods (Smith-Waterman and Needleman-Wunsch). The heuristic employed by BLAST methods involve pattern-matching small pieces of the sequences against one another. For each possible pair of nucleotides or amino acids, one must perform a table lookup to accumulate the probability of this match happening by chance. Once the 'optimal' alignment is found, the total quality of the alignment and a statistical measure of the alignment occurring by chance are reported which together may provide evidence that the proteins are homologous (i.e., share a common ancestor) and thus, share a common function. Few floating-point operations are performed in calculating sequence alignment with respect to the integer operations (index calculation, string matching, table lookup, etc.). For example, comparing two amino acids would require 2 integer lookups (one for each amino acid), followed by integer multiply and add to determine the location in the scoring matrix from which to retrieve a floating-point probability, the lookup of that probability, and then finally an accumulate or other operation on the probability. This sort of sequence is at the heart of alignment algorithms, normally in multiply nested loops going over combinations of amino acids or nucleotides from the sequences. Hence

sequence alignment and homology detection methods rarely operate anywhere near peak machine flops. However, since so few calculations are carried out per character read, improved memory latency, memory hierarchy, size, and memory bandwidth are critical architectural characteristics for a machine designed to support sequence alignment algorithms.

Support Vector Machine (SVM): *Using advanced optimization methods to classify biomolecules.*

Support vector machines are an increasingly popular machine-learning technique for classifying complex processes using a vector representation of their features. The SVM algorithm involves transformation of the vectors using one of many possible higher-order kernels. Using a collection of vectors of known class, one 'trains' the SVM to detect vectors which belong to the class, and those which don't. The resulting classifier is a lightweight representation of the training set which can then be used to rapidly classify new observations of unknown class with statistical confidence. The brute-force approach to calculating the SVM kernel involves creating an *m* by *m* matrix where *m* is the number of vectors in the training set. Each column corresponds to the transformed combination (*i.e.* dot product for a linear kernel) of a single vector with all other vectors. Each vector then gets its own column, resulting in a square matrix. Researchers have already used this approach to train classifiers for homology detection using tens of thousands of training vectors. The memory footprint of the GIST application for a set of 50808 training vectors was 26 Gbyte of memory, which was primarily dedicated to a large dense matrix. This is somewhat larger than the kernel matrix because of associated bookkeeping matrices required for GIST to run. The target training set size is in the millions, but is currently unreachable using the brute-force method. A newer method known as sequential minimization optimization (SMO), only calculates some pieces of the kernel uses a smaller memory footprint. But SMO is much more sensitive to memory latency and bandwidth than GIST because it retrieves a different working pair of vectors depending on a constantly updated gradient estimate. GIST can be optimized to reuse cache as much as possible because of the regularity of the square matrix. But SMO methods takes more of a random walk through memory.

1.2 Hardware balance requirements

Molecular dynamics simulation methods already make use of teraflop-scale computers and will strongly benefit from larger-scale high-performance computing resources. With the increase in processor flop count outpacing the speedup in communication networks, the interprocessor communication has become the scaling bottleneck of these simulations on massively parallel architectures. Currently, the sequential computation only takes a fraction of the time needed in each step, and each step only takes 0.1-1 second. Latency is one of the determining factors for biological molecular dynamics simulations, which require synchronization during every time step. Currently, the general approach for distributing the atoms of the molecular system over the available processors requires that (a relatively small amount of) information be interchanged between those processors at each of the large number of time steps of the simulation. For example, even the low 3-5 microsecond latency of the current Elan4 interconnect on the MSCF machine is a bottleneck for large molecular dynamics simulations on large processor counts.

The key requirement for biological applications is the need to balance integer operations and memory operations with the flop rate. Many of the applications in this domain are data-driven meaning that one cannot predict *a priori* what the memory access pattern will be. Memory operations are scheduled in response to decisions made on the basis of previous computing (*e.g.* a sequence alignment is performed only for sequence pairs with an optimal alignment which is statistically significant). For the most part, simple prefetching is not a viable solution for these applications since memory access has a tendency to be sparse. Most biological applications will rely on the architecture to support an even balance of integer operations (including integer multiply and integer compare), memory operations and flops per cycle. Ideally, the next system would support at least one memory operation, one integer multiply and one integer compare per flop.

1.3 Hardware capacity requirements

Very low latency interconnect is the main requirement for molecular dynamics codes. An increase in processor speed will reduce the time spent in the computational kernels even further, and possibly negligible, thereby making the interprocessor communication an even more dominant factor in the simulations.

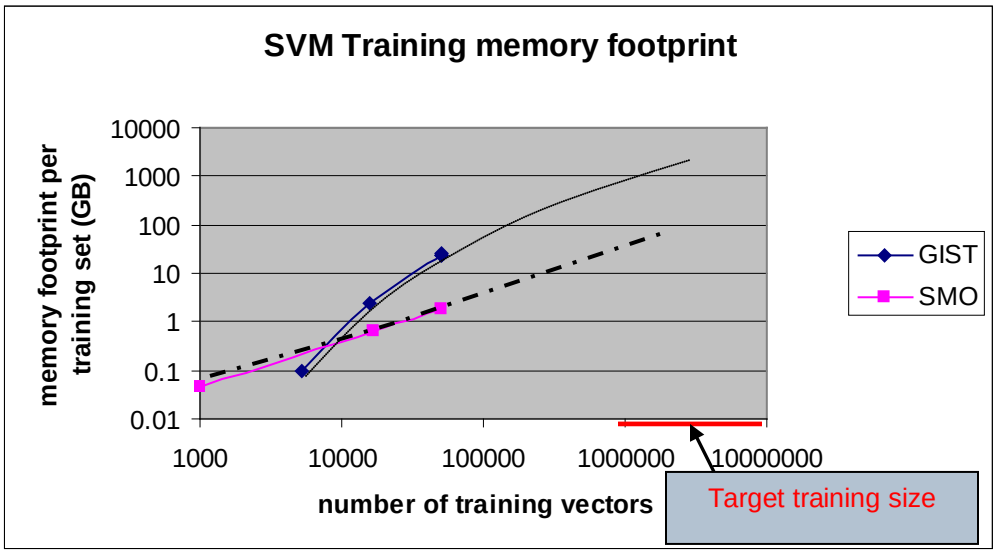
Currently, all of the listed applications in biology other than molecular dynamics are bottlenecked with respect to memory access. The GIST implementation of SVM requires 26 Gbyte of memory for training on the order of 50,000 vectors. The near future of this technology will require training on sets 20x this size, requiring 400x the memory, easily into the Tbyte range. Although researchers will more likely use the substantially smaller memory footprint SVM implementation SMO, access to memory is less predictable and will still require 100 Gbyte of memory for our target training set size (see Figure on the following page). Currently, both applications use less than 10% of the peak memory bandwidth available. Using the performance monitoring software, NWPerf, one can show that, after accounting for process migration, SMO uses 0.48 bytes per cycle (0.686 GByte/sec) and GIST uses 0.21 bytes per cycle (0.3 GByte/sec) per processor.

ScalaBLAST has been run on as many as 1500 processors on MPP2, achieving a virtually ideal (constant) work per processor rate regardless of number of CPU's involved. This is achieved by *overlapping communication and remote memory access with computation* using the Global Array toolkit through the low-latency interconnect on MPP2. Excellent scaling has been demonstrated as well on high-latency interconnect environments, but not having remote DMA gives a 5-10x drop in performance. This huge penalty for operating without remote data access motivates the need for RDMA in the next architecture. (Parallel SMO implementation will be even more sensitive to this issue because very large datasets shared between processors cannot be effectively prefetched). The target databases being searched in typical sequence alignment tasks are growing at an exponential rate. As high-throughput sequencing technology continues to mature, more genomes are coming online daily. As each new organism's genome becomes available, it is incorporated into the large curated databases against which most alignments are performed. The size of the protein database is now 1.6 Gbyte and doubles in size every 18 months. The size of the nucleotide database is approximately 12 Gbyte and doubles at the same rate. Each search through the databases requires accessing each of the sequences in its entirety. For each base pair or amino acid pair, a lookaside operation has

to be performed and accumulated to calculate the probability of the match occurring by chance. Performance monitoring of ScalaBLAST reveals that it only uses 0.05 Gbyte/sec (0.035 bytes/cycle) of memory bandwidth and the application is stalled approximately 50% of the time because of inter-register dependencies (presumably from the time requred to perform index calculations for memory retrieves). Because this algorithm requires pattern matching, the emphasis is on integer operations, not flops. Polygraph has may similarities, in that it can operate on the same databases that ScalaBLAST operates on, the emphasis is on integer operations, and scoring requires accumulated lookaside operations and lots of character comparisons. Polygraph exhibits excellent scalability and can make extensive use of large memory space to avoid recomputing quantities. Polygraph sustains only 0.143 Gbyte/sec of memory bandwidth (0.1 bytes/cycle).

Biological applications are not flop intensive, in general. SVM methods both have sustained performance at less than 3% peak flop rate, Polygraph operates at 3.6%, and ScalaBLAST sustains 0.15%. None of the applications tax the memory bandwidth capacity with demand above 10% of the peak 6.5 Gbyte/sec as discussed above, and all are plagued by FPU stalls rather than cache misses. *The real requirement is improved integer handling*, specifically integer multiplication and integer comparison. Indirect addressing requires integer multiplication, and takes on the order of 30 clock cycles to perform, relative to a 1 clock cycle cost for performing a single (or multiple) floating point multiplies. Applications in biology require a shift in balance away from flop-heavy architectures and toward an even balance between flops and iops. Assuming an architecture in which this balance is achieved, these biology applications also require memory bandwidth in proportion to the flop and iop rate. For most biology applications, cache reuse is difficult to arrange because memory operations are being determined by the results of calculations.

The specific hardware capacity requirements resulting from these applications are summarized below:



Large memory is the biggest requirement for the next generation machine with respect to applications in biology. SVM requires on the order of Terabytes for applications right

now, whereas that memory requirement is approximately 5-10 years out for sequence alignment applications, but certain to come. Polygraph could take advantage of increased memory capacity to the 10s or 100s of Tbytes today increasing its throughput by roughly a factor of 3.

High memory bandwidth is the second requirement. While the biology applications currently running on MPP2 do not push the limits of memory bandwidth, it is expected that an increased capacity for integer operations will alleviate the FPU stall problem and push the bottleneck toward memory bandwidth. Biological applications running on a float/integer balanced system will require on the order of 1 byte/cycle of memory bandwidth or better.

Integer multiply and string matching capacity are not well-supported by significant number of the current architectures, but are the main determinants in the performance of many biological applications. Biological applications need integer operations which balance the floating point arithmetic capacity of the machine.

A large global filesystem is a necessary component for these applications. Molecular dynamics trajectories and complete sequence alignments are on the order of Tbyte and must be written on-the-fly. For example, a single time step in the simulation of a protein consisting of 50,000 residues or, with each residue containing about 20 atoms, a million atoms manipulates tens of megabytes of data. The current strategy for a nanosecond simulation of small biomolecular systems of about 50,000 atoms is to store trajectory data for every 10^{th} or 50^{th} time step for post-processing analysis. One such simulation can easily generate single data files of 200 gigabytes. Larger 100 nanosecond simulations of a biomolecular system of ten million atoms could potentially generate 4,000 terabytes of data based on current state-of-the-art methods. Asynchronous I/O support would further enhance performance of these codes by allowing file-write operations to be overlapped with computation.

High I/O bandwidth to global filesystem is motivated by profiling of ScalaBLAST which reveals that it spends approximately 10% of its time in I/O. A benchmark 3.5 hour run using 1500 processors produced a total of 109 Gbytes of files. Approximating that this run spent 21 minutes per processor in I/O, the total system peak I/O to the globally mounted file system would need to be at least 0.27 Gbyte/sec if the balanced iop, flop and memory bandwidth alleviate the FPU stall problem, and if this sustained per-processor rate is extended to 2000 processors on MPP2 to model current capacity.

2 Chemical Sciences

The chemical sciences area discussed in the MSCF Greenbook describes a wide variety of chemical processes at the molecular scale using quantum chemical and molecular dynamics methodologies. Application areas range from modeling large biomolecules, catalysis, nanoscience, to highly accurate spectroscopic, energetic and dynamical properties. The algorithms used in molecular dynamics will be discussed in the biological sciences section of this document.

In this section the most widely utilized algorithms in quantum chemistry will be discussed, mapped onto hardware balance requirements, and Greenbook based examples will be presented that provide insight into future hardware capability requirements. Quantum chemistry algorithms will utilize each feature available within the hardware architecture, some of the key needs are: 1) large memory space and bandwidth, and low memory latency, 2) large local scratch disk, 3) very low latency interconnect.

2.1 Algorithmic information

Analysis of the algorithms used in quantum chemistry will be based on those available in the computational quantum chemistry software package NWChem, developed at EMSL and used by many of the users of the MSCF resources. However, the algorithms in NWChem are representative of those in the other most widely used quantum chemistry codes currently available to MSCF users, such as Molpro, GAMESS, ADF, and VASP. The chemical sciences section of the Greenbook also discusses molecular dynamics methodologies, which already have been covered in the biological sciences section above. Many of the multiscale approaches, such as QM/MM methodologies, utilize a combination of the chemistry methods analyzed here and, hence, can be analyzed based on their individual components.

Two classes of algorithms will be analyzed, those based on Gaussian functions and those based on plane waves. Methodologies based on Gaussian functions are the most widely used, but over the last two years the MSCF has seen a larger user community starting to utilize software based on plane wave methodology.

Gaussian based methodologies

The basic building block to describe the quantum mechanical wave function is the basis function. To properly describe the electronic structure of an atom, or the interactions of atoms in a molecule multiple Gaussian functions are needed. These basis functions are used to build up atomic or molecular orbitals, with Gaussian basis sets commonly used. The number of functions can range from ten to hundreds per atom, depending on the type of atom and the accuracy that the scientist wants to achieve. Also, the computational chemist has access to a wide variety of methods that scale anywhere from N to N^7 and higher with respect to the number of operations, depending on the accuracy required and/or the scientific question that needs to be answered. Here N is the number of Gaussian basis functions in the system studied.

If only low to medium level accuracy is needed, methods such as Hartree-Fock (HF) and Density Functional Theory (DFT) can be utilized. These methods scale in the range from N^2-N^3 and, hence, can be used to study molecules that contain large numbers of atoms. HF and DFT require memory to store multiple matrices of size NxN and either memory

or disk to store the Gaussian integrals of size N^4. Using permutation symmetry the integral storage can be reduced by a factor of 8. These quantities need to be accessed multiple times due to the iterative nature of these methodologies. For DFT the number of Gaussian integrals required can be reduced to N^3 through the introduction of an additional Coulomb Fitting approximation.

To achieve highly accurate information for chemical properties, and enable a direct comparison between theory and experiment, higher order methods, such as Møller-Plesset (MP) and coupled cluster theory are needed. MP2 (second-order MP) methodologies scale as approximately N^5. The coupled cluster with iterative single and double excitations and linearized triples approximation, CCSD(T), is widely used to achieve chemical accuracies of approximately one kcal/mol. Operations for the iterative CCSD part of the code scale $X*N^6$ (with X depends on the number of iterations needed for convergence, generally from 10-50), whereas the (T) part scales N^7. The scaling behavior of the MP2 and CCSD(T) severely limits the size of system that can be studied. On the current MSCF hardware, scientists can study molecules with $N\approx2000$ for MP2 and $N\approx1000$ for CCSD(T). Memory requirements for the MP2 and CCSD(T) are of the order N^4, which is for the transformed Gaussian basis function integrals. At the high accuracy CCSD(T) level the integrals can be recalculated, but this will add additional floating point operations scaling to N^5 for every iterative step.

Basic computational kernels underpinning all Gaussian based methodologies are the evaluation of N^4 integrals, the construction of matrices (Fock Matrix), and large matrix multiplication for the CCSD(T). The basic structure of these kernels will be discussed first, followed by a discussion on the use of semi-direct algorithms and parallelism.

Gaussian integral evaluation

Gaussian integral evaluation accounts for up to 80% of the time for medium precision electronic structure calculations such as HF, DFT, and high precision MP2. The integral codes tend to be large, complex, and generally consist of many critical loops. These loops can contain recursion algorithms that can be vectorized, many short DAXPY-like (BLAS level-1) operations, and some matrix-matrix and matrix-vector operations (i.e. level-2 and level-3 BLAS). Integrals are build up from common components that are calculated separately, stored in local memory. Assembling the components of the integrals requires gather-scatter operations from local memory. Although the integral algorithms can be tuned to utilize larger cache, memory latency is an important factor to achieve single processor efficiency.

Fock matrix construction

Both HF and DFT combine the Gaussian integrals described above with a so-called density matrix to form a Fock matrix that will be subsequently diagonalized. Building a Fock matrix is similar to a sparse matrix-vector product. A single integral gets combined with multiple density matrix elements (collected through a gather operation),and the product will contribute to multiple Fock matrix elements (distributed through a scatter operation). The gather-scatter of matrix elements requires integer index or address evaluations.

Matrix multiplication kernels

High accuracy methodologies such as CCSD(T) have been formulated such that they can utilize DGEMM and similar operations that can run at near theoretical peak speed. In fact, about 90% of the time-to-solution for a CCSD(T) calculation is spend in these kernels. Both DFT (quadrature) and the plane wave methodology make significant use of matrix multiplication operations.

Semi-direct approaches

Gaussian integral evaluations are computationally expensive, and most quantum chemistry codes will use semi-direct algorithms to reduce the recomputation in every iteration by storing expensive Gaussian integrals, and partly transformed molecular integrals in memory or on disk. The data size needed is in the order of N^4, which will quickly consume all available memory and generally requires the integrals to be stored on disk. Data access patterns for the HF and CCSD(T) are to write once and read many times large records sequentially, whereas MP2 uses a strided write and sequential read. The read/write block sizes depend on the available processor memory.

Hence, the Gaussian based methodologies are set up to maximize use of all available memory and disk space to store integrals and other intermediate products of the calculations. Once the storage is exhausted, the remaining integrals will be recomputed in every iteration (i.e. direct).

Parallelism

High-performance computing today is synonymous with large numbers of processors. These large numbers of processors do not only provide the large number of floating point operations to the workhorse N^3-N^7 algorithms, they also provide an aggregate of memory and disk storage that can be exploited.

The parallel communication protocol used by quantum chemistry applications are MPI and Global Array (GA) tools. In both protocols the availability of a fast network is essential in order for the computational chemistry software to scale to large numbers of processors. NWChem relies on GA for global memory access, message passing, and parallel linear algebra operations. Its essential network communication layer, ARMCI, is built on low level network APIs and will utilize the fastest communication, including non-blocking and asynchronous network operations, available on a given architecture. Most computational chemistry algorithms require (globally distributed) data to be shared among processors at regular or irregular intervals—requiring very efficient interprocessor communication. GA and ARMCI are part of the Global Array Toolkit, which also includes functionality for local memory allocation and parallel I/O. Parallel I/O can consist of independent files on each processor, or globally shared files. Many major computational chemistry codes that are scalable use GA, and there is an increased use in other scientific fields.

Gaussian integrals are distributed over the processors to be evaluated and stored (using the aggregate I/O available), or recomputed. Both the density matrix and Fock matrix, needed in the Fock matrix construction, can be replicated on each processor, or distributed or mirrored globally over all the processors. The latter becomes the only viable solution once the matrix becomes to big to be stored in the local memory of a

single processor. Replication and distribution each bring their own challenges with respect to parallelization. If the Fock matrix is replicated a global summation will be required to assemble the complete matrix. Globally distributed density and Fock matrices require put, get, and accumulate operations between processors, making the Fock matrix construction network latency sensitive. This also applies to other matrices and vectors that are stored and accessed globally by all processors. In addition, to diagonalize a distributed matrix requires very efficient parallel linear algebra libraries. Most DGEMM-like operations are performed locally, but require the data to be retrieved from global memory (possibly on other processors). Using non-blocking network or I/O communication prefetching techniques can be utilized to minimize the latency penalty.

In recent years a second level of parallelism has been introduced in quantum chemistry codes, where multiple calculations are performed in parallel, each using a subset of the available processors. An example is the calculation of part of the potential energy surface (PES), or numerical hessians and gradients, which are needed to allow predictions of structural, spectroscopic, and thermodynamic properties, and the determination of dynamic properties such as rate constants. A numerical hessian or gradient requires $9*N_a*N_a$ or $3*N_a$ (N_a is the number of atoms in the molecular system) individual energy calculations respectively. Each of these individual energy calculations might not scale to a large number of processors, due to network latency and bandwidth, and performing multiple individual energy calculations in parallel on a smaller number of processors can provide a faster time to solution. Chemical dynamics simulations can use parallelization in a similar fashion. These types of calculations consist of a large ensemble of individual trajectory calculations, which can be run efficiently in parallel.

Plane wave based methodologies

Density functional based plane wave methodologies can provide modeling results that are of a similar accuracy as the Gaussian DFT methodology. The main time consuming sequential kernel of the pseudo potential plane wave density functional methodology, which includes Car-Parinello, is the calculation of terms from the so-called non-local pseudopotential. This kernel mainly consists of BLAS operations of the type DGER. The remaining time is spend in 3-D Fast Fourier Transforms (FFT) and orthogonalization, the latter consisting of DDOT and DAXPY operations. Recently, plane wave codes with exact exchange have been emerging, which allow scientists to use the popular hybrid functionals currently used in Gaussian DFT calculations. The exact exchange requires the extensive use of FFT, which then becomes the most time consuming kernel in the calculation.

Alternative formalism of plane wave density functional theory is the projector augmented plane wave (PAW) methodology. Within this formalism the non-local pseudopotential term is by far the dominant term in the computation due to the large numbers of grid points, i.e. long vectors, that need to be operated on and summed. PAW algorithms are highly vectorizable, and easy to parallelize. Both plane wave and PAW sequential algorithms require fast CPUs, large cache, and low memory latency, but do not need large memory or disk.

All the plane wave codes currently used on the MSCF resources use MPI for their parallel communication. The expensive non-local pseudopotential calculation can be

easily parallelized through distribution of the data, and can be scaled to large numbers of processors. However, parallel 3-D FFT's and orthogonalization operations require access to the distributed data. Hence, at large processor counts the scalability of the calculation will depend on the scalability of the parallel 3-D FFT (2-D FFT of a local slab of a 3-D grid) and the movement of data for DDOT operations.

2.2 Hardware balance requirements

The computational chemistry algorithms described in the previous section utilize a cache-blocked architecture. Therefore, cache latency, bandwidth, and size are important to performance and scalability. Data reuse is limited and access to cache and memory becomes a driving factor. Currently NWChem applications use between 0.2-0.5 bytes/cycle (0.3-0.75 GBytes/sec), only a fraction of the available memory bandwidth. Analysis of hardware counters of single processor MP2 and DFT runs shows a large stall rate (40% and higher), and that the execution units get only partly utilized per cycle. Block sizes in the integral routines can be tuned to improve cache locality, although most operations will need to run well out of cache.

Large blocks of temporary data are used by the quantum chemistry algorithms, and therefore large local memory with low latency and high bandwidth greatly enhances the science that can be achieved, as does access to large amounts of fast storage capabilities for temporary storage of intermediate results. Currently, most quantum chemistry applications will utilize a large part of the available 3-4 GByte of memory per processor on MPP2, in addition to the local scratch disk when local memory is insufficient (which is generally the case). A symmetric local scratch system is necessary for balanced computations. For example, the MPP2 machine has some nodes with 300+ Gbyte of disk, while others have only 12 Gbyte. When both nodes are used in a single computation, for example when a calculation is run on the whole machine, the resources used are those of the smallest denominator, i.e. 12 Gbyte. A key issue is the ability to handle distributed data structures on a fully distributed memory system.

Most computational chemistry algorithms require data to be shared among processors at regular or irregular intervals, requiring efficient interprocessor communication. To obtain efficient communication among a cluster of processors an interconnect equipped with high bandwidth and low latency is required. For example, NWChem's standard DFT benchmark improved time to solution by approximately 30 percent when the Quadrics interconnect on MPP2 was upgraded to $QsNet^{II}$ (with the bandwidth going from 350 Mbyte/sec to 900 MByte/sec and the ARMCI put latency going from 4.7 to 2.5 microseconds). Large shared memory processing systems can currently provide low latency and high bandwidth for memory access to small- to medium-sized computing problems. However, existing science problems have demonstrated shortcomings in using current shared memory architectures in terms of the scalability of I/O on single-image operating systems, increasing the complexity and cost of maintaining cache coherency and the physical limitations of the number of simultaneous memory accesses to the same address.

Key to performance is the ability to overlap computation, communication, and I/O operations (e.g., use of asynchronous I/O, use of non-blocking communication operations). The chemical sciences area will benefit from interconnect capabilities that

enable one-sided communication, facilitating the overlap of computational operations with communications, and provide high bandwidth. Plane wave methodologies require very efficient parallel 3-D FFT libraries to effectively scale to large numbers of processors.

In summary, analysis of the algorithms used in NWChem, representative for a wide variety of quantum chemistry application currently used on the MSCF resources, shows that a *balanced* architecture in terms of processor speed, memory and cache size, latency and bandwidth, processor interconnect bandwidth and latency, and latency and bandwidth to I/O is needed. The current balance of the MPP2 machine is very suitable for chemistry. Limiting factors on the current architecture are the amount of memory, the asymmetric local scratch, limiting its use to only a small fraction when a large fraction of the machine is used in a single run. Quantum chemistry codes can and will exploit each part of the available hardware architecture to obtain the fastest time to solution.

2.3 Hardware capacity requirements

It is generally recognized that computational chemistry can consume all hardware resources of the MSCF in the foreseeable future by scaling computations to achieve higher accuracy, modeling more extended and realistic systems, and introducing dynamics as a variable in the computations. In this section some examples of the future application hardware capacity needs of the Greenbook chemical sciences area will be described.

Depending on the method used, scaling of computational effort with system size varies from nearly linear order N scaling to N^7, and increases in computational effort with system size ultimately limit the size of the system that can be studied. Today, it is possible using quantum mechanical approaches to make predictions of molecular properties of *modest-sized* systems (tens of atoms) that are as reliable as the most detailed experiments, especially for thermodynamics, structure, and many spectroscopic properties. For systems with hundreds of atoms approximate methods can be employed to reduce the computational effort. However, establishing the validity of computational methods and models, as well as the uncertainty and limitations of the methods requires high accuracy benchmark calculations.

With the scaling of current accurate methods ranging from N^5 (MP2) to N^7 (CCSD(T)), the need for access to large computational resources for accurate predictions is evident. An example of the state of the art with the current MSCF computational hardware and software is the calculation of the heat of formation of the octane molecule, a hydrocarbon that is part of automobile fuel. The most computationally intensive part of this calculation required 1,400 processors or 8.4 Tflops for 23 hours, yielding 75 percent CPU efficiency as the kernel is mainly DGEMM operations. However, current computational resources are not enough to obtain these properties for even bigger hydrocarbons such as cetane, which is twice as large as octane and for which there is practically no crucial experimental data available. The octane calculation required 8.7 Tbytes of aggregate storage, whereas the same calculation on the cetane molecule will require at least 63 Tbytes. If the cetane calculation would be performed on today's MSCF hardware the calculation would take over 2200 hours to complete on 1400 processors. Hence, a large increase in computational resources is required to make these calculations feasible. In

general, high accuracy problem sizes that are of interest to the computational chemistry community, but cannot be addressed by the current state-of-the-art computing resources, range from 2000-5000 orbitals and 100-200 electrons. Although the size of systems for which accurate calculations can be performed is growing (e.g., accurate energetics can be obtained for clusters with more than 20 water molecules), extension of these types of calculations to more reliable models of condensed-phase systems will require significant increases in computational resources. For example, explicit high accuracy calculations of up to 100 water molecules would require an approximate 600 - 3,000 fold increase in the computational resources (storage and CPU power) depending on the method used.

In addition, the octane example above involved a single energy at a single nuclear configuration. A simple kinetics calculation at this level of accuracy for a similar sized system would require finding the stationary points (reactant, transition state, and product), which requires approximately thirty energy evaluations as a conservative estimate. Hence, one would need approximately 200 Tflops to obtain the kinetics of one reaction involving the octane molecule.

Another drive is to perform simulations on more realistic models with fewer approximate methods. Although the size of systems for which accurate calculations can be performed is growing (e.g., accurate energetics can be obtained for clusters with more than 20 water molecules), extension of these types of calculations to reliable models will require significant increases in computational resources. For example, condensed-phase chemical problems involve hundreds to thousands of atoms, and can include computationally expensive atoms such as transition metals or heavy elements.

Another example requiring large computational resources is the modeling of the energetics of large molecular systems such as metalloproteins or nanoparticles. The structure of a protein determines its functionality and activity (for example biological processes for the transfer of electrons and energy in proteins in cells), and the structure of a nanoparticle controls its properties. Researchers are currently pushing the boundaries modeling a 690 atom metalloprotein that consists of 8770 basis functions at the low accuracy HF level of theory. This calculation requires the recomputation of the integrals for every iteration as there is simply not enough storage on the machine to hold them all.

The optimal determination of molecular structures of these large molecules is a difficult problem and currently requires repeated sampling of the potential energy surface. The computational effort scales nonlinearly with dimensionality of the system. For example, in the most simpleminded approach of sampling over n points in each dimension, the effort scales as n^D, where D is the number of degrees of freedom (formally 3 times the number of atoms) in the system. For the condensed-phase or chemical interfaces this may involve systems of hundreds and possibly thousands of atoms.

The calculation of dynamic properties, such as diffusion, energy transfer, and chemical reaction rates, requires following the motions of atoms and molecules as a function of time. Treating the atoms classically, atomic motion is followed by integrating the classical equations of motion to determine the classical trajectory of the system. Instead of following a single trajectory, one can also calculate an ensemble of trajectories for a chemical dynamics simulation, and this ensemble may be as small as 50 trajectories or as large as several thousand trajectories, depending on the nature of the problem.

Parallelization of such a multi-trajectory simulation is straightforward by simply running multiple trajectories in parallel on groups of processors.

For each time step in each of these classical trajectories the forces on the atoms need to be evaluated on the potential energy surface. Depending on the accuracy of the method used do the evaluation, with a computational expense ranging from order N or N^7, each time step could take seconds or minutes to many hours on current MSCF hardware resources. For these types of simulations, hours is simply to long. Minutes may be too long as well. In addition, time scales for dynamical properties of interest, especially rare events such as passing over a barrier in a chemical reaction or gas-to-liquid nucleation, can be much larger than the inherent time scales in the order of 10^{-15} seconds of motions in molecules, requiring large numbers of steps along the trajectory. The computational effort scales linearly with the number of time steps in a calculation, which could be in the order of 10^3-10^{10}. Current methods, such as Car-Parinello dynamics, have provided unique insights into dynamic properties on short-time scales, yet the electronic structure on which the dynamics are based is only approximate and the time scale is very short. Yet, such methods are needed to explore reactivity in solution and biochemical systems.

3 Environmental Systems Sciences

Knowledge of the chemical and biological reactions of environmentally important processes can be used to resolve critical DOE environmental challenges such as bio-remediation and carbon sequestration. Thus, large multi-scale subsurface and climate models must be developed to simulate long-term events that may influence policy decisions concerning natural and human impacts on the environment. Challenges representing subsurface and climate modeling are often intertwined—for example, atmospheric modeling may feed into watershed and surface water models that link to the subsurface.

3.1 Algorithmic information

Both climate modeling and subsurface chemistry model information laid out on three dimensional grids that evolve in time and are distributed over a collection of processors. Those grids may be refined globally or locally to include multi-scale computations and may move to follow fronts. Most of the computations involve large sparse matrix vector products and large vector-vector operations that may involve contiguous, strided, or indexed memory accesses. The wall clock time required to run these simulations requires that history files containing all of the data necessary for a restart be generated at regular intervals.

For both the climate and subsurface domains, the low level nature of serial operations involved tend to be dot-product, and vector-triple operations like DAXPY.

The subsurface chemistry arena needs small dense matrix solves of which tens of thousands to millions are done, with matrices modified at each step of a nonlinear outer iteration. It also utilizes large sparse system solves such as those provided by the PETSc library to solve linear sub-problems of Newton-like nonlinear solvers.

Both domains have advection and diffusion of species (chemical and/or precipitate) over their modeling space. This feature along with the large sparse system solves require frequent nearest neighbor style communications. The size of the communications tends to increase with the square of a linear spatial dimension, hence with the 2/3 power of the product of the spatial dimensions.

Both of these environmental system simulation areas have data that needs to be aggregated into a single result and then redistributed at frequent intervals. These aggregations and broadcasts are used to guide future steps in the computation. They also both share a multi-scale flavor with the desire to run much finer resolution grids on parts of or all of their domain.

3.2 Hardware balance requirements

The vector triples operations, dot product operations and sparse matrix operations are all memory bandwidth limited. The rate limiting component of the computations is not how fast floating point operations on register arguments can be performed. The speed at which operands can be loaded into the floating point registers from memory is the current bottleneck. There is very little data reuse in existing codes which are millions of lines long. Most operations could utilize two to three double words of memory bandwidth per machine clock (16 to 24 bytes/cycle). Current architectures can only deliver a fraction of

that – one to two bytes per cycle. Lots of high speed cache, with at least 4 way associativity and large numbers of floating point registers can help to hide this latency some what. It is also important to be able to queue enough outstanding memory requests to allow prefetching to help hide the high latency to memory. Currently some of the climate codes achieve about 10% of peak floating point speed on our Itanium IA64 cluster while the subsurface chemistry and regional climate codes only get 3% to 5% of peak efficiency.

The history keeping features of the climate modeling codes and the basic nature of output for spatial grids progressing through time with several variables per grid point calls for significant parallel input/output capacity on each processor. A globally visible file system that allows multiple writers and readers per file and has an efficient file locking mechanism is required. A significant portion of time (5% to 30%) for these runs is spent writing history data for restart and output data making asynchronous I/O capability critical. The large volume of output from these applications also requires an archive system capable of handling tens of Tbytes of data per day for long term storage. That archive file system will need to be network available for remote user access at very high bandwidths (minimally tens to hundreds of Mbytes per second).

The advection and diffusion of material species in both climate and subsurface modeling as well as the sparse matrix-vector products in the large sparse linear system solves for the subsurface modeling arena will require low latency point to point communications for nearest neighbor style communication patterns. Sufficient bandwidth for all processors to be able to communicate concurrently in this point to point capacity is important. Non-blocking, one-sided communications (with remote direct memory access and zero-copy communications) will be important features.

Currently all of the environmental systems modeling codes using the MSCF require the MPI paradigm for communicating, so MPI 1.1 compliance is required and MPI 2.0 capabilities are desired. The other flavor of communication that is critical to the applications is the global reduction and broadcast operation for global dot products. This global reduction and broadcast is done several times per nearest neighbor communications and requires low latency communication that should only grow logarithmically in elapsed time with the number of processors. The ability to aggregate results over subgroups of processes will also be important as underlying algorithms are migrated to resource fault tolerance.

The problems of the future in the environmental systems arenas will grow by two to five orders of magnitude. Thus they will need to run for longer periods of time on more components increasing the likelihood of at least one component failing during the execution of a single run. This suggests that some form of hardware redundancy in the form of hot spare processors and memory, raided disks for global file systems and possibly communication logging capacity will be desirable. Also crucial in the capacity to provide fault resilience will be resource managers capable of detecting hardware faults and allowing computations to proceed in spite of them. Substantial effort from the application side will also be required.

3.3 Hardware capacity requirements

The three-dimensional Hanford Sitewide Groundwater Model currently uses inverse simulations to identify geologic zonation and hydrologic parameters, including boundary conditions. Future pilot-scale analyses for the Hanford Site will require significantly more (10 to 100 times) grid resolution, will include more detailed transport processes (e.g., chemistry), and will address 10 to 100 times more estimated parameters. These additional capabilities could require more than a 1,000 times increase in the number of flops to be done, and storage requirements can be expected to increase by a factor of 10 to 100.

Multiscale and hybrid approaches (molecular-scale modeling of mineral surfaces, in silico models of metal reducing bacteria, and pore-scale models of multiphase flow and multi-component reactive transport) coupled with longer time simulations and increasing from two to three dimensional models, are expected to increase computational requirements by three to five orders of magnitude. In addition, parameter estimation and uncertainty computations will add one to two more orders of magnitude to obtain the reliability necessary for policy decisions.

For example, the Hanford Site composite analysis encompasses several different models and solution techniques that address multiple environmental pathways on the Hanford Site. The risk analysis is based on 1,000-year simulations of 14 contaminants originating at 1,053 waste sites using 100 realizations to address variability and uncertainty in the parameter space. A need exists to (1) increase dimensionality and resolution in the modeled areas to better represent the existing and future plumes; (2) incorporate additional process models (e.g., multicomponent geochemistry to reflect spatially and temporally variable contaminant concentrations, and bank storage to identify release and migration of contaminants through seepage faces); (3) add more sophisticated time-dependent boundary conditions; and (4) increase the number of realizations to account for more of the parameter space. These additional capabilities could increase the computational requirements by 100 to 1000 times the current use.

The memory requirements for the subsurface chemistry codes are currently 200 Gbytes to 300 Gbytes (or about 1 Gbyte per processor). With increases in spatial refinement and multiscale models, this can be expected to grow to 2 Tbytes to 30 Tbytes of memory per run. Systems with less memory will require out of core solutions which tend to be 10 to 100 times slower on existing architecture balances and hence would require very high bandwidth, and lower latency globally visible disk solutions. These codes currently generate 10's of Gbytes of simulation output per run and can be expected to generate Tbytes of data per run as they grow in complexity. The total volume of data communicated will grow approximately two orders of magnitude in moving from the two dimensional simulations to three dimensional simulations. Currently, the size of the nearest neighbor communications are tens of Kbytes to hundreds of Kbytes per message per neighbor, for a total volume of tens of Mbytes per nearest neighbor communication. Increasing the volume by one hundred times increases the need to be able to do non-blocking one-sided (preferably zero-copy) communication.

In the last few years, a new approach to the treatment of clouds has replaced conventional cloud parameterizations in a few models. This new approach consists of embedding a

cloud resolving model (CRM) in each climate model grid cell. Each CRM explicitly resolves the cloud circulation and its influence on microphysical and radiative processes. The resulting coupled system is often called the Multiscale Modeling Framework (MMF). Preliminary results showing quantitative improvements in climate simulations suggest that the MMF methodology will become the approach of choice for estimating the radiative forcing by anthropogenic greenhouse gases and aerosols, investigating physical feedbacks in the climate system, and understanding the relationship between global climate forcing and regional hydrological changes. It is a computationally expensive solution, increasing the run time of a simulation by a factor of 200 in coarse resolution models. However, because the CRMs dominate the computations and interact with each other only through the outer spatial (coarsest resolution) grid, the MMF scales with respect to the number of processors much better than conventional climate models, permitting efficient parallelization on thousands of processors on some computing systems. The MSCF, with its concentration on chemical and physical processes such as aerosol chemistry, is used to improve and develop the physics and boundary conditions of the model clouds in the CRM.

Climate modeling with computationally intensive models began in the 1960s and has been limited by computing resources during its history. Even development of the improved model, which is the focus here, requires large computational resources for correcting and validating changes in the models. Current global climate models have been parallelized to run on hundreds of processors at coarse (200-kilometer) resolution and thousands of processors at fine (10-kilometer) resolution. Most climate simulations are run at coarse resolution to permit multicentury simulations and to store the resulting data. Fine resolution simulations are severely limited by the lack of data storage ability. Coarse resolution simulations are limited by the cost of communication, which in turn limits scalability to less than 1,000 processors. This feature again highlights the need for the ability to overlap communication with computation via non-blocking and one sided communication capacity.

Future simulations with advanced and improved versions of the MMF will require enhanced computational resources. Refining the resolution of the CRMs by a factor of four will require at least a 16-fold increase in computing power and storage to achieve the same simulation throughput. Three-dimensional cloud geometry would increase the computational needs by another factor of 1.5. Refining the global model grid size by a factor of two will require another factor of four-increase in computing power and storage. Thus, future improved MMF simulations will require a 100-fold increase in computing power to maintain the same simulation throughput.

Some improvement in the throughput can be achieved by using a larger number of processors. At the current 2^{o}x2.5^{o}xL26 resolution of CAM, the number of processors is limited to 240. Increasing the number of processors can be accomplished in the model either by increasing the resolution of CAM (which would render the MMF even slower) or by using a different domain decomposition that permits up to 8,000 processors for the same grid resolution. Using a different domain decomposition and allowing columns to be shared across processors and/or using a task scheduled approach could thus provide a 30-fold increase in time-to-solution of the simulation. The processor count also could be increased by increasing the number of processors sharing memory on a single node. A

factor of 16 is unrealistic, but a factor of 2 to 4 might be achievable. Thus, a 100-fold increase in throughput is achievable using next-generation microprocessor-based machines.

The amount of memory currently used by MMF models is 30 Gbyte to 250 Gbyte per run (about 1 Gbyte per processor). Refinements of grid spacing in the horizontal directions and moving towards three dimensional modeling on the cloud scale will require terabytes to tens of terabytes of memory. Memory latency for the CAM is an issue, limiting the performance of climate models on high-performance computing systems to only five to ten percent of peak theoretical performance; however, the MMF was developed for cache based machines and consumes approximately 90% of the computational time for the simulations. Although the standard CAM has been vectorized to run efficiently on vector machines, the MMF version of CAM has not. Further work would be required to develop a vector version of the MMF that can run efficiently on vector systems.

Replacing the cyclical lateral boundary conditions in the CRMs with one that allows propagation between CRMs presents a significant communication challenge to the computing system. The frequent communication requires a low latency interconnect. A significant restructuring of the MMF code will also be required to ensure that message passing between CRMs are only required if the CRMs reside on different processors or nodes (the latter can consist of multiple processors).

Volume data for model history will become an issue at high resolution. Currently, climate modeling history is comprised of 12 Gbytes per simulated day, or 4.4 Tbytes per simulated year. This would increase to 50 Gbytes per day (18 Tbytes per simulated year) at 1 kilometer CRM resolution. Thus, 20 years of simulation will produce a total of 360 Tbytes of data/run. Needing an ensemble of runs for model verification will easily reach the Pbyte range of data production. That data needs to be accessible to on site and remote users for subsequent analysis and visualization.

4 Summary

An analysis of the algorithms in all the three science areas has been presented in the three sections above. The analysis shows that a *balanced* architecture is needed with respect to processor, memory hierarchy, interprocessor communication, and disk access and storage. A single architecture can satisfy the needs of all of the science areas, although some areas may take greater advantage of certain aspects of the architecture.

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

This document was printed on recycled paper.
(9/2003)

HPCS-3 Project: Greenbook Algorithms and Hardware Needs Analysis

Executive Summary

This document describes the algorithms, and hardware balance requirements needed to enable the solution of real scientific problems in the DOE core mission areas of environmental and subsurface chemistry, computational and systems biology, and climate science. The MSCF scientific drivers have been outlined in the Greenbook, which is available online at http://mscf.emsl.pnl.gov/docs/greenbook_for_web.pdf . Historically, the primary science driver has been the chemical and the molecular dynamics of the biological science area, whereas the remaining applications in the biological and environmental systems science areas have been occupying a smaller segment of the available hardware resources.

To go from science drivers to hardware balance requirements, the major applications were identified. Major applications on the MSCF resources are low- to high-accuracy electronic structure methods, molecular dynamics, regional climate modeling, subsurface transport, and computational biology. The algorithms of these applications were analyzed to identify the computational kernels in both sequential and parallel execution. This analysis shows that a *balanced* architecture is needed with respect to processor speed, peak flop rate, peak integer operation rate, and memory hierarchy, interprocessor communication, and disk access and storage. A single architecture can satisfy the needs of all of the science areas, although some areas may take greater advantage of certain aspects of the architecture. The table below summarizes the primary hardware needs for the three scientific areas.

Hardware Needs	**B**	**C**	**E**
Memory hierarchy (bandwidth, size and latency)	X	X	X
Peak flops	X	X	X
Peak integer operations	X		
Overlap computation, communication and I/O	X	X	X
Low communication latency	X	X	X
High communication bandwidth		X	
Large memory	X	X	
High I/O bandwidth		X	
Increasing disk storage needs (size)	X*		X*
B = Biology; C = Chemistry; E = Environmental Systems Science * Relatively high bandwidth I/O is needed to store large blocks of data			

The main aspects of the architecture, summarized in this table, will be discussed below in relation to the associated science areas.

Processor Architecture and Memory Hierarchy

Each of the MSCF science areas requires a CPU with high-performance *floating-point* capabilities. Several of the computational areas will also require significant integer capabilities for addressing memory. The percentage of peak performance for the chemical sciences, biological modeling with molecular dynamics methods, and environmental systems is strongly driven by the hierarchical memory access, memory bandwidth, and latency. Large amounts of low latency and high-bandwidth memory are needed to account for the 10 to 100 times increase in data set size envisioned by the scientific community in the next five years. Fast memory in combination with large cache and pre-fetching mechanisms will improve scientific application performance. The amount of memory per processor is tightly coupled to the need for local fast disk space and I/O, with memory being the preferred way to store run-time data. Environmental systems researchers can take full advantage of hardware optimized for the long vectors that are characteristic of the grid-based solution schemes. Several chemistry and biology applications could benefit from vector capabilities, as well. In summary, the science areas have the following processor and memory requirements:

- *a CPU with high sustained floating-point performance; biological sciences need good integer operation (including integer multiply) performance*
- *large memory (with low latency and high bandwidth perceived as important to high sustained floating-point performance)*
- *fast hierarchical memory access (i.e., large cache and pre-fetching, again, perceived as important to high sustained performance).*

Interprocessor Communication

High-performance computing today is synonymous with large numbers of processors. Most algorithms used for the MSCF science areas are tightly coupled and require data to be shared among processors at regular or irregular intervals—requiring efficient interprocessor communication. To obtain efficient communication among a cluster of processors (e.g., a distributed memory machine), an interconnect equipped with high bandwidth and low latency is required. Latency is one of the determining factors for biological molecular dynamics simulations, which require synchronization during every time step. For other computations (e.g., electronic structure computations), high bandwidth is required. The chemical and biological sciences areas will benefit from interconnect capabilities that enable one-sided communication to facilitate the overlap of computational operations with communications, and provide high bandwidth. Environmental systems models tend to communicate in a "nearest neighbor" communication pattern and will benefit from one- to three-dimensional interconnect topologies. Large shared memory processing systems can currently provide low latency and high bandwidth for memory access for small- to medium-sized computing problems. However, existing science problems have demonstrated shortcomings in using current shared memory architectures in terms of the scalability of I/O on single-image operating systems, the complexity and cost of maintaining cache coherency and the physical limitations of the number of simultaneous memory accesses to the same address.

In summary, the science areas require the following for interprocessor communication:

- *high bidirectional bandwidth and low latency*
- *one-sided communication and overlap of communication with computation*
- *exploration of clustered shared-memory architectures with commensurate I/O capability and operating system architecture.*

Disk Storage and Access

Of the science areas discussed in the Greenbook, those related to high-accuracy chemical sciences methods require large, temporary, high-speed local disk storage and rapid I/O capability that can be accessed in a non-blocking or asynchronous fashion. The amount of local disk storage is tightly coupled to available memory; the latter considered the preferred method for storing intermediate data. The environmental systems science area and biological molecular dynamics methodologies require up to hundreds of terabytes of disk space to store associated simulation data. Code performance in the environmental systems science area would improve through the use of efficient parallel I/O to a shared global file system. For example, Climate Resolving Models presently require 12 gigabytes of disk space per simulated day, or more than four terabytes per simulated year. To improve model performance, a higher-resolution grid (one kilometer or less) is needed, which would increase the data-storage requirements to 50 gigabytes and 18 terabytes per day and year, respectively. Thus, a 20-year simulation would produce 360 terabytes of data. The full amount of disk space would not be required at run time, although a multi-terabyte shared global file system would be required to perform these simulations. Subsurface modeling also has similar data requirements as those discussed above. In summary, the algorithm analysis for disk storage and access requires:

- *local high-speed disk storage with non-blocking access (pre-fetch)*
- *a large shared global file system with reasonably fast parallel I/O.*

Algorithms and Hardware Needs Analysis

The remainder of this document is split into three main sections, one for each of the three science areas, biological sciences, chemical sciences, and environmental systems sciences, identified in the Greenbook. In each section the major applications were identified, and the algorithms of these applications were analyzed to determine the computational kernels in both sequential and parallel execution. The algorithmic information was subsequently mapped onto hardware balance requirements. In addition, for each science area guidance on future hardware capacity and resource needs will be provided.

1 Biological Sciences

The science drivers for biology identified in the Greenbook workshop highlight several architectural needs for keeping pace with computational biology. Many of the computational algorithms needed for leading-edge biological science are fundamentally different than those used by computational chemistry, or environmental sciences. However, in many cases the computational architecture support needed to enable the next generation of high-performance biology software *is* similar to other science domains. The following section presents an analysis of benchmark data from representative applications in computational biology indicating three key needs: 1) very large memory space and memory bandwidth, 2) enhanced capacity for integer operations, 3) large globally-mounted filesystem with high I/O bandwidth.

1.1 Algorithmic information

The biology domain areas represented at the Greenbook workshop include molecular dynamics applied to biological molecules, prediction of protein structure from sequence data, building physiological models over many spatial and temporal scales, applying statistical models to experimental pipelines for data analysis, and bioinformatics applications for analyzing sequence data.

Molecular dynamics: *using molecular dynamics to understand protein function*

The use of classical molecular dynamics enables the modeling of atomic motions over time in large biomolecular systems consisting of tens to hundreds of thousands of atoms. Interactions between atoms are described by a force field (AMBER or CHARMM), or parameterized effective pair model. At each time step in the simulation, the forces of each atom in the system need to be evaluated. The sequential kernels involved in this evaluation consist of simple arithmetic operations (square root, exp, sin/cos, etc.). In addition, force fields can be extended with additional terms such as the particle-mesh Ewald (PME) correction for long range electrostatic interactions. These PME corrections require the calculation of Fast Fourier Transforms (FFT) and become the dominant step in the computations.

To improve the time-to-solution, and to enable scientists to study dynamical processes that occur on time scales longer than a nanosecond time, parallelization is essential. Parallel molecular dynamics codes use domain decomposition of the physical simulation volume to distribute the work. For large molecular systems, this distribution of the

atomic data will also reduce the memory requirements, which are generally in the hundreds of Mbytes. Domain decomposition methods utilize the locality of molecular interactions to minimize the interprocessor communication. However, it is this communication between processors, which is taking place at each of the hundreds of thousands simulation time steps, that is the main bottleneck of the calculations. For example, for the calculation of forces and energies in which atoms are involved on neighboring domains assigned to different processors, information from these domains need to be exchanged between processors. In addition, the dynamical nature of these calculations requires a periodic redistribution and associated data communication for atoms that move from one processor's domain to another. Moreover, for heterogeneous molecular systems the computational work is generally not evenly distributed over the processors. Atom and heterogeneity require the use of dynamic load balancing of domains in order for the parallelization to be efficient. In NWChem molecular dynamics simulations use point-to-point onesided asynchronous communication to make the interprocessor communication as efficient as possible.

Monte Carlo, energy minimization: *de novo prediction of protein structure*

Gene information is presented in the form of a sequence of nucleotides using a character alphabet with four elements. This can be converted, using a well-characterized mapping, to a collection of sequences of amino acids. Each of these sequences corresponds to a putative protein - the molecular machinery that drives biological systems. It is commonly believed that one must understand the structure of these putative proteins to understand their functions. The de novo approach to this involves using protein sequence data and chemical properties of the amino acid sequence to find a structure corresponding to a minimum in chemical energy. Because of the size of these systems (the average protein length is on the order of 350 amino acids, and proteins can be thousands of residues in length), one must perform a collection of randomly seeded energy minimization calculations to have any chance at capturing the true structure. Currently, this is being approached in high-performance applications such as Rosetta (Bonneau et al.). The algorithms used in performing this type of calculation involve 1) using Monte Carlo or other random-walk to seed the energy minimization calculation with a starting structure; 2) performing an energy minimization or other scoring function to assess the goodness of the structure. This approach can be implemented without communication between processors, with the exception of control information governing overall progress and task assignments.

Physiological modeling: *Integrative multiscale modeling to understand signaling and regulation*

Among the biological sciences, physiological modeling most resembles conventional computational science. It is generally implemented as a mesh or agent-based domain description on which ordinary differential equations, partial differential equations, or rule-based interactions are calculated. Physiological modeling applications tend to be the most flop-intensive of the computational biology areas, as they frequently require the solution of equations on a physiological domain. In the continuum approach, solution of biological systems is achieved in much the same way that climate modeling and subsurface transport calculations are performed. Real or idealized system is converted to a mesh or grid representation on which theoretically determined sets of equations are

solved. Visualization tools are then used to convey the results to the user. In the discrete approach, agent-based approach can be used to govern the motion and interaction of objects vested with biological meaning. In both approaches, the key bottleneck is *memory access*. State differential equations can rely on many other states, requiring run-time access to essentially a random collection of memory elements. Agent-based approaches generally localize the attributes of an agent according to their spatial relationship at startup. But interactions could conceivably occur between agents anywhere in the system, making it necessary to expose these data fragments to any agent in the simulation. If repeated load-balancing is employed to enforce some degree of locality, memory access will still be the bottleneck because one cannot predict *a priori* how agents will be partitioned to best share the load after movement occurs. Regardless of the implementation, these approaches require substantial post-processing for visualization or feature-extraction, and substantial pre-processing for setting up the mesh or agent-based domain.

Peptide identification from MS data: *Using statistical models to map experimental measurements to protein identity.*

DOE has invested heavily in high-throughput proteomics facilities which analyze the identity of proteins in biological samples using a variety of techniques. In general, the experimental pipeline produces quantitative peak data which gives information about the relative mass to charge ratio of fragments of proteins (peptides). The computational task of determining the amino acid sequence of the peptides is primarily string matching. Calculating the total mass of peptide candidates in database-searching methods such as Polygraph and Sequest requires inner-loop lookups of amino acid weights. Assessing the goodness of the candidate's match to the experimental spectrum involves limited floating-point arithmetic compared to the memory movement and lookaside required.

Sequence alignment and homology detection: *Informatics applied to multi-species systems*

Sequence alignment is the process of determining which regions of two biological sequences are equivalent with statistical confidence. Heuristic methods such as BLAST (NCBI) achieve high speed by sacrificing accuracy with respect to exhaustive methods (Smith-Waterman and Needleman-Wunsch). The heuristic employed by BLAST methods involve pattern-matching small pieces of the sequences against one another. For each possible pair of nucleotides or amino acids, one must perform a table lookup to accumulate the probability of this match happening by chance. Once the 'optimal' alignment is found, the total quality of the alignment and a statistical measure of the alignment occurring by chance are reported which together may provide evidence that the proteins are homologous (i.e., share a common ancestor) and thus, share a common function. Few floating-point operations are performed in calculating sequence alignment with respect to the integer operations (index calculation, string matching, table lookup, etc.). For example, comparing two amino acids would require 2 integer lookups (one for each amino acid), followed by integer multiply and add to determine the location in the scoring matrix from which to retrieve a floating-point probability, the lookup of that probability, and then finally an accumulate or other operation on the probability. This sort of sequence is at the heart of alignment algorithms, normally in multiply nested loops going over combinations of amino acids or nucleotides from the sequences. Hence

sequence alignment and homology detection methods rarely operate anywhere near peak machine flops. However, since so few calculations are carried out per character read, improved memory latency, memory hierarchy, size, and memory bandwidth are critical architectural characteristics for a machine designed to support sequence alignment algorithms.

Support Vector Machine (SVM): *Using advanced optimization methods to classify biomolecules.*

Support vector machines are an increasingly popular machine-learning technique for classifying complex processes using a vector representation of their features. The SVM algorithm involves transformation of the vectors using one of many possible higher-order kernels. Using a collection of vectors of known class, one 'trains' the SVM to detect vectors which belong to the class, and those which don't. The resulting classifier is a lightweight representation of the training set which can then be used to rapidly classify new observations of unknown class with statistical confidence. The brute-force approach to calculating the SVM kernel involves creating an *m* by *m* matrix where *m* is the number of vectors in the training set. Each column corresponds to the transformed combination (*i.e.* dot product for a linear kernel) of a single vector with all other vectors. Each vector then gets its own column, resulting in a square matrix. Researchers have already used this approach to train classifiers for homology detection using tens of thousands of training vectors. The memory footprint of the GIST application for a set of 50808 training vectors was 26 Gbyte of memory, which was primarily dedicated to a large dense matrix. This is somewhat larger than the kernel matrix because of associated bookkeeping matrices required for GIST to run. The target training set size is in the millions, but is currently unreachable using the brute-force method. A newer method known as sequential minimization optimization (SMO), only calculates some pieces of the kernel uses a smaller memory footprint. But SMO is much more sensitive to memory latency and bandwidth than GIST because it retrieves a different working pair of vectors depending on a constantly updated gradient estimate. GIST can be optimized to reuse cache as much as possible because of the regularity of the square matrix. But SMO methods takes more of a random walk through memory.

1.2 Hardware balance requirements

Molecular dynamics simulation methods already make use of teraflop-scale computers and will strongly benefit from larger-scale high-performance computing resources. With the increase in processor flop count outpacing the speedup in communication networks, the interprocessor communication has become the scaling bottleneck of these simulations on massively parallel architectures. Currently, the sequential computation only takes a fraction of the time needed in each step, and each step only takes 0.1-1 second. Latency is one of the determining factors for biological molecular dynamics simulations, which require synchronization during every time step. Currently, the general approach for distributing the atoms of the molecular system over the available processors requires that (a relatively small amount of) information be interchanged between those processors at each of the large number of time steps of the simulation. For example, even the low 3-5 microsecond latency of the current Elan4 interconnect on the MSCF machine is a bottleneck for large molecular dynamics simulations on large processor counts.

The key requirement for biological applications is the need to balance integer operations and memory operations with the flop rate. Many of the applications in this domain are data-driven meaning that one cannot predict *a priori* what the memory access pattern will be. Memory operations are scheduled in response to decisions made on the basis of previous computing (*e.g.* a sequence alignment is performed only for sequence pairs with an optimal alignment which is statistically significant). For the most part, simple prefetching is not a viable solution for these applications since memory access has a tendency to be sparse. Most biological applications will rely on the architecture to support an even balance of integer operations (including integer multiply and integer compare), memory operations and flops per cycle. Ideally, the next system would support at least one memory operation, one integer multiply and one integer compare per flop.

1.3 Hardware capacity requirements

Very low latency interconnect is the main requirement for molecular dynamics codes. An increase in processor speed will reduce the time spent in the computational kernels even further, and possibly negligible, thereby making the interprocessor communication an even more dominant factor in the simulations.

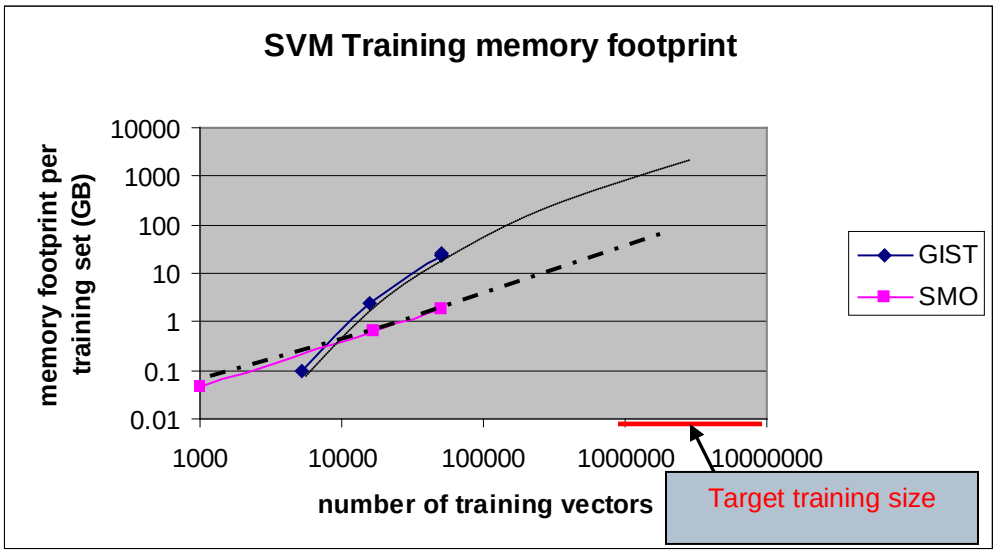
Currently, all of the listed applications in biology other than molecular dynamics are bottlenecked with respect to memory access. The GIST implementation of SVM requires 26 Gbyte of memory for training on the order of 50,000 vectors. The near future of this technology will require training on sets 20x this size, requiring 400x the memory, easily into the Tbyte range. Although researchers will more likely use the substantially smaller memory footprint SVM implementation SMO, access to memory is less predictable and will still require 100 Gbyte of memory for our target training set size (see Figure on the following page). Currently, both applications use less than 10% of the peak memory bandwidth available. Using the performance monitoring software, NWPerf, one can show that, after accounting for process migration, SMO uses 0.48 bytes per cycle (0.686 GByte/sec) and GIST uses 0.21 bytes per cycle (0.3 GByte/sec) per processor.

ScalaBLAST has been run on as many as 1500 processors on MPP2, achieving a virtually ideal (constant) work per processor rate regardless of number of CPU's involved. This is achieved by *overlapping communication and remote memory access with computation* using the Global Array toolkit through the low-latency interconnect on MPP2. Excellent scaling has been demonstrated as well on high-latency interconnect environments, but not having remote DMA gives a 5-10x drop in performance. This huge penalty for operating without remote data access motivates the need for RDMA in the next architecture. (Parallel SMO implementation will be even more sensitive to this issue because very large datasets shared between processors cannot be effectively prefetched). The target databases being searched in typical sequence alignment tasks are growing at an exponential rate. As high-throughput sequencing technology continues to mature, more genomes are coming online daily. As each new organism's genome becomes available, it is incorporated into the large curated databases against which most alignments are performed. The size of the protein database is now 1.6 Gbyte and doubles in size every 18 months. The size of the nucleotide database is approximately 12 Gbyte and doubles at the same rate. Each search through the databases requires accessing each of the sequences in its entirety. For each base pair or amino acid pair, a lookaside operation has

to be performed and accumulated to calculate the probability of the match occurring by chance. Performance monitoring of ScalaBLAST reveals that it only uses 0.05 Gbyte/sec (0.035 bytes/cycle) of memory bandwidth and the application is stalled approximately 50% of the time because of inter-register dependencies (presumably from the time requred to perform index calculations for memory retrieves). Because this algorithm requires pattern matching, the emphasis is on integer operations, not flops. Polygraph has may similarities, in that it can operate on the same databases that ScalaBLAST operates on, the emphasis is on integer operations, and scoring requires accumulated lookaside operations and lots of character comparisons. Polygraph exhibits excellent scalability and can make extensive use of large memory space to avoid recomputing quantities. Polygraph sustains only 0.143 Gbyte/sec of memory bandwidth (0.1 bytes/cycle).

Biological applications are not flop intensive, in general. SVM methods both have sustained performance at less than 3% peak flop rate, Polygraph operates at 3.6%, and ScalaBLAST sustains 0.15%. None of the applications tax the memory bandwidth capacity with demand above 10% of the peak 6.5 Gbyte/sec as discussed above, and all are plagued by FPU stalls rather than cache misses. *The real requirement is improved integer handling*, specifically integer multiplication and integer comparison. Indirect addressing requires integer multiplication, and takes on the order of 30 clock cycles to perform, relative to a 1 clock cycle cost for performing a single (or multiple) floating point multiplies. Applications in biology require a shift in balance away from flop-heavy architectures and toward an even balance between flops and iops. Assuming an architecture in which this balance is achieved, these biology applications also require memory bandwidth in proportion to the flop and iop rate. For most biology applications, cache reuse is difficult to arrange because memory operations are being determined by the results of calculations.

The specific hardware capacity requirements resulting from these applications are summarized below:



Large memory is the biggest requirement for the next generation machine with respect to applications in biology. SVM requires on the order of Terabytes for applications right

now, whereas that memory requirement is approximately 5-10 years out for sequence alignment applications, but certain to come. Polygraph could take advantage of increased memory capacity to the 10s or 100s of Tbytes today increasing its throughput by roughly a factor of 3.

High memory bandwidth is the second requirement. While the biology applications currently running on MPP2 do not push the limits of memory bandwidth, it is expected that an increased capacity for integer operations will alleviate the FPU stall problem and push the bottleneck toward memory bandwidth. Biological applications running on a float/integer balanced system will require on the order of 1 byte/cycle of memory bandwidth or better.

Integer multiply and string matching capacity are not well-supported by significant number of the current architectures, but are the main determinants in the performance of many biological applications. Biological applications need integer operations which balance the floating point arithmetic capacity of the machine.

A large global filesystem is a necessary component for these applications. Molecular dynamics trajectories and complete sequence alignments are on the order of Tbyte and must be written on-the-fly. For example, a single time step in the simulation of a protein consisting of 50,000 residues or, with each residue containing about 20 atoms, a million atoms manipulates tens of megabytes of data. The current strategy for a nanosecond simulation of small biomolecular systems of about 50,000 atoms is to store trajectory data for every 10^{th} or 50^{th} time step for post-processing analysis. One such simulation can easily generate single data files of 200 gigabytes. Larger 100 nanosecond simulations of a biomolecular system of ten million atoms could potentially generate 4,000 terabytes of data based on current state-of-the-art methods. Asynchronous I/O support would further enhance performance of these codes by allowing file-write operations to be overlapped with computation.

High I/O bandwidth to global filesystem is motivated by profiling of ScalaBLAST which reveals that it spends approximately 10% of its time in I/O. A benchmark 3.5 hour run using 1500 processors produced a total of 109 Gbytes of files. Approximating that this run spent 21 minutes per processor in I/O, the total system peak I/O to the globally mounted file system would need to be at least 0.27 Gbyte/sec if the balanced iop, flop and memory bandwidth alleviate the FPU stall problem, and if this sustained per-processor rate is extended to 2000 processors on MPP2 to model current capacity.

2 Chemical Sciences

The chemical sciences area discussed in the MSCF Greenbook describes a wide variety of chemical processes at the molecular scale using quantum chemical and molecular dynamics methodologies. Application areas range from modeling large biomolecules, catalysis, nanoscience, to highly accurate spectroscopic, energetic and dynamical properties. The algorithms used in molecular dynamics will be discussed in the biological sciences section of this document.

In this section the most widely utilized algorithms in quantum chemistry will be discussed, mapped onto hardware balance requirements, and Greenbook based examples will be presented that provide insight into future hardware capability requirements. Quantum chemistry algorithms will utilize each feature available within the hardware architecture, some of the key needs are: 1) large memory space and bandwidth, and low memory latency, 2) large local scratch disk, 3) very low latency interconnect.

2.1 Algorithmic information

Analysis of the algorithms used in quantum chemistry will be based on those available in the computational quantum chemistry software package NWChem, developed at EMSL and used by many of the users of the MSCF resources. However, the algorithms in NWChem are representative of those in the other most widely used quantum chemistry codes currently available to MSCF users, such as Molpro, GAMESS, ADF, and VASP. The chemical sciences section of the Greenbook also discusses molecular dynamics methodologies, which already have been covered in the biological sciences section above. Many of the multiscale approaches, such as QM/MM methodologies, utilize a combination of the chemistry methods analyzed here and, hence, can be analyzed based on their individual components.

Two classes of algorithms will be analyzed, those based on Gaussian functions and those based on plane waves. Methodologies based on Gaussian functions are the most widely used, but over the last two years the MSCF has seen a larger user community starting to utilize software based on plane wave methodology.

Gaussian based methodologies

The basic building block to describe the quantum mechanical wave function is the basis function. To properly describe the electronic structure of an atom, or the interactions of atoms in a molecule multiple Gaussian functions are needed. These basis functions are used to build up atomic or molecular orbitals, with Gaussian basis sets commonly used. The number of functions can range from ten to hundreds per atom, depending on the type of atom and the accuracy that the scientist wants to achieve. Also, the computational chemist has access to a wide variety of methods that scale anywhere from N to N^7 and higher with respect to the number of operations, depending on the accuracy required and/or the scientific question that needs to be answered. Here N is the number of Gaussian basis functions in the system studied.

If only low to medium level accuracy is needed, methods such as Hartree-Fock (HF) and Density Functional Theory (DFT) can be utilized. These methods scale in the range from N^2-N^3 and, hence, can be used to study molecules that contain large numbers of atoms. HF and DFT require memory to store multiple matrices of size NxN and either memory

or disk to store the Gaussian integrals of size N^4. Using permutation symmetry the integral storage can be reduced by a factor of 8. These quantities need to be accessed multiple times due to the iterative nature of these methodologies. For DFT the number of Gaussian integrals required can be reduced to N^3 through the introduction of an additional Coulomb Fitting approximation.

To achieve highly accurate information for chemical properties, and enable a direct comparison between theory and experiment, higher order methods, such as Møller-Plesset (MP) and coupled cluster theory are needed. MP2 (second-order MP) methodologies scale as approximately N^5. The coupled cluster with iterative single and double excitations and linearized triples approximation, CCSD(T), is widely used to achieve chemical accuracies of approximately one kcal/mol. Operations for the iterative CCSD part of the code scale $X*N^6$ (with X depends on the number of iterations needed for convergence, generally from 10-50), whereas the (T) part scales N^7. The scaling behavior of the MP2 and CCSD(T) severely limits the size of system that can be studied. On the current MSCF hardware, scientists can study molecules with $N\approx2000$ for MP2 and $N\approx1000$ for CCSD(T). Memory requirements for the MP2 and CCSD(T) are of the order N^4, which is for the transformed Gaussian basis function integrals. At the high accuracy CCSD(T) level the integrals can be recalculated, but this will add additional floating point operations scaling to N^5 for every iterative step.

Basic computational kernels underpinning all Gaussian based methodologies are the evaluation of N^4 integrals, the construction of matrices (Fock Matrix), and large matrix multiplication for the CCSD(T). The basic structure of these kernels will be discussed first, followed by a discussion on the use of semi-direct algorithms and parallelism.

Gaussian integral evaluation

Gaussian integral evaluation accounts for up to 80% of the time for medium precision electronic structure calculations such as HF, DFT, and high precision MP2. The integral codes tend to be large, complex, and generally consist of many critical loops. These loops can contain recursion algorithms that can be vectorized, many short DAXPY-like (BLAS level-1) operations, and some matrix-matrix and matrix-vector operations (i.e. level-2 and level-3 BLAS). Integrals are build up from common components that are calculated separately, stored in local memory. Assembling the components of the integrals requires gather-scatter operations from local memory. Although the integral algorithms can be tuned to utilize larger cache, memory latency is an important factor to achieve single processor efficiency.

Fock matrix construction

Both HF and DFT combine the Gaussian integrals described above with a so-called density matrix to form a Fock matrix that will be subsequently diagonalized. Building a Fock matrix is similar to a sparse matrix-vector product. A single integral gets combined with multiple density matrix elements (collected through a gather operation),and the product will contribute to multiple Fock matrix elements (distributed through a scatter operation). The gather-scatter of matrix elements requires integer index or address evaluations.

Matrix multiplication kernels

High accuracy methodologies such as CCSD(T) have been formulated such that they can utilize DGEMM and similar operations that can run at near theoretical peak speed. In fact, about 90% of the time-to-solution for a CCSD(T) calculation is spend in these kernels. Both DFT (quadrature) and the plane wave methodology make significant use of matrix multiplication operations.

Semi-direct approaches

Gaussian integral evaluations are computationally expensive, and most quantum chemistry codes will use semi-direct algorithms to reduce the recomputation in every iteration by storing expensive Gaussian integrals, and partly transformed molecular integrals in memory or on disk. The data size needed is in the order of N^4, which will quickly consume all available memory and generally requires the integrals to be stored on disk. Data access patterns for the HF and CCSD(T) are to write once and read many times large records sequentially, whereas MP2 uses a strided write and sequential read. The read/write block sizes depend on the available processor memory.

Hence, the Gaussian based methodologies are set up to maximize use of all available memory and disk space to store integrals and other intermediate products of the calculations. Once the storage is exhausted, the remaining integrals will be recomputed in every iteration (i.e. direct).

Parallelism

High-performance computing today is synonymous with large numbers of processors. These large numbers of processors do not only provide the large number of floating point operations to the workhorse N^3-N^7 algorithms, they also provide an aggregate of memory and disk storage that can be exploited.

The parallel communication protocol used by quantum chemistry applications are MPI and Global Array (GA) tools. In both protocols the availability of a fast network is essential in order for the computational chemistry software to scale to large numbers of processors. NWChem relies on GA for global memory access, message passing, and parallel linear algebra operations. Its essential network communication layer, ARMCI, is built on low level network APIs and will utilize the fastest communication, including non-blocking and asynchronous network operations, available on a given architecture. Most computational chemistry algorithms require (globally distributed) data to be shared among processors at regular or irregular intervals—requiring very efficient interprocessor communication. GA and ARMCI are part of the Global Array Toolkit, which also includes functionality for local memory allocation and parallel I/O. Parallel I/O can consist of independent files on each processor, or globally shared files. Many major computational chemistry codes that are scalable use GA, and there is an increased use in other scientific fields.

Gaussian integrals are distributed over the processors to be evaluated and stored (using the aggregate I/O available), or recomputed. Both the density matrix and Fock matrix, needed in the Fock matrix construction, can be replicated on each processor, or distributed or mirrored globally over all the processors. The latter becomes the only viable solution once the matrix becomes to big to be stored in the local memory of a

single processor. Replication and distribution each bring their own challenges with respect to parallelization. If the Fock matrix is replicated a global summation will be required to assemble the complete matrix. Globally distributed density and Fock matrices require put, get, and accumulate operations between processors, making the Fock matrix construction network latency sensitive. This also applies to other matrices and vectors that are stored and accessed globally by all processors. In addition, to diagonalize a distributed matrix requires very efficient parallel linear algebra libraries. Most DGEMM-like operations are performed locally, but require the data to be retrieved from global memory (possibly on other processors). Using non-blocking network or I/O communication prefetching techniques can be utilized to minimize the latency penalty.

In recent years a second level of parallelism has been introduced in quantum chemistry codes, where multiple calculations are performed in parallel, each using a subset of the available processors. An example is the calculation of part of the potential energy surface (PES), or numerical hessians and gradients, which are needed to allow predictions of structural, spectroscopic, and thermodynamic properties, and the determination of dynamic properties such as rate constants. A numerical hessian or gradient requires $9*N_a*N_a$ or $3*N_a$ (N_a is the number of atoms in the molecular system) individual energy calculations respectively. Each of these individual energy calculations might not scale to a large number of processors, due to network latency and bandwidth, and performing multiple individual energy calculations in parallel on a smaller number of processors can provide a faster time to solution. Chemical dynamics simulations can use parallelization in a similar fashion. These types of calculations consist of a large ensemble of individual trajectory calculations, which can be run efficiently in parallel.

Plane wave based methodologies

Density functional based plane wave methodologies can provide modeling results that are of a similar accuracy as the Gaussian DFT methodology. The main time consuming sequential kernel of the pseudo potential plane wave density functional methodology, which includes Car-Parinello, is the calculation of terms from the so-called non-local pseudopotential. This kernel mainly consists of BLAS operations of the type DGER. The remaining time is spend in 3-D Fast Fourier Transforms (FFT) and orthogonalization, the latter consisting of DDOT and DAXPY operations. Recently, plane wave codes with exact exchange have been emerging, which allow scientists to use the popular hybrid functionals currently used in Gaussian DFT calculations. The exact exchange requires the extensive use of FFT, which then becomes the most time consuming kernel in the calculation.

Alternative formalism of plane wave density functional theory is the projector augmented plane wave (PAW) methodology. Within this formalism the non-local pseudopotential term is by far the dominant term in the computation due to the large numbers of grid points, i.e. long vectors, that need to be operated on and summed. PAW algorithms are highly vectorizable, and easy to parallelize. Both plane wave and PAW sequential algorithms require fast CPUs, large cache, and low memory latency, but do not need large memory or disk.

All the plane wave codes currently used on the MSCF resources use MPI for their parallel communication. The expensive non-local pseudopotential calculation can be

easily parallelized through distribution of the data, and can be scaled to large numbers of processors. However, parallel 3-D FFT's and orthogonalization operations require access to the distributed data. Hence, at large processor counts the scalability of the calculation will depend on the scalability of the parallel 3-D FFT (2-D FFT of a local slab of a 3-D grid) and the movement of data for DDOT operations.

2.2 Hardware balance requirements

The computational chemistry algorithms described in the previous section utilize a cache-blocked architecture. Therefore, cache latency, bandwidth, and size are important to performance and scalability. Data reuse is limited and access to cache and memory becomes a driving factor. Currently NWChem applications use between 0.2-0.5 bytes/cycle (0.3-0.75 GBytes/sec), only a fraction of the available memory bandwidth. Analysis of hardware counters of single processor MP2 and DFT runs shows a large stall rate (40% and higher), and that the execution units get only partly utilized per cycle. Block sizes in the integral routines can be tuned to improve cache locality, although most operations will need to run well out of cache.

Large blocks of temporary data are used by the quantum chemistry algorithms, and therefore large local memory with low latency and high bandwidth greatly enhances the science that can be achieved, as does access to large amounts of fast storage capabilities for temporary storage of intermediate results. Currently, most quantum chemistry applications will utilize a large part of the available 3-4 GByte of memory per processor on MPP2, in addition to the local scratch disk when local memory is insufficient (which is generally the case). A symmetric local scratch system is necessary for balanced computations. For example, the MPP2 machine has some nodes with 300+ Gbyte of disk, while others have only 12 Gbyte. When both nodes are used in a single computation, for example when a calculation is run on the whole machine, the resources used are those of the smallest denominator, i.e. 12 Gbyte. A key issue is the ability to handle distributed data structures on a fully distributed memory system.

Most computational chemistry algorithms require data to be shared among processors at regular or irregular intervals, requiring efficient interprocessor communication. To obtain efficient communication among a cluster of processors an interconnect equipped with high bandwidth and low latency is required. For example, NWChem's standard DFT benchmark improved time to solution by approximately 30 percent when the Quadrics interconnect on MPP2 was upgraded to $QsNet^{II}$ (with the bandwidth going from 350 Mbyte/sec to 900 MByte/sec and the ARMCI put latency going from 4.7 to 2.5 microseconds). Large shared memory processing systems can currently provide low latency and high bandwidth for memory access to small- to medium-sized computing problems. However, existing science problems have demonstrated shortcomings in using current shared memory architectures in terms of the scalability of I/O on single-image operating systems, increasing the complexity and cost of maintaining cache coherency and the physical limitations of the number of simultaneous memory accesses to the same address.

Key to performance is the ability to overlap computation, communication, and I/O operations (e.g., use of asynchronous I/O, use of non-blocking communication operations). The chemical sciences area will benefit from interconnect capabilities that

enable one-sided communication, facilitating the overlap of computational operations with communications, and provide high bandwidth. Plane wave methodologies require very efficient parallel 3-D FFT libraries to effectively scale to large numbers of processors.

In summary, analysis of the algorithms used in NWChem, representative for a wide variety of quantum chemistry application currently used on the MSCF resources, shows that a *balanced* architecture in terms of processor speed, memory and cache size, latency and bandwidth, processor interconnect bandwidth and latency, and latency and bandwidth to I/O is needed. The current balance of the MPP2 machine is very suitable for chemistry. Limiting factors on the current architecture are the amount of memory, the asymmetric local scratch, limiting its use to only a small fraction when a large fraction of the machine is used in a single run. Quantum chemistry codes can and will exploit each part of the available hardware architecture to obtain the fastest time to solution.

2.3 Hardware capacity requirements

It is generally recognized that computational chemistry can consume all hardware resources of the MSCF in the foreseeable future by scaling computations to achieve higher accuracy, modeling more extended and realistic systems, and introducing dynamics as a variable in the computations. In this section some examples of the future application hardware capacity needs of the Greenbook chemical sciences area will be described.

Depending on the method used, scaling of computational effort with system size varies from nearly linear order N scaling to N^7, and increases in computational effort with system size ultimately limit the size of the system that can be studied. Today, it is possible using quantum mechanical approaches to make predictions of molecular properties of *modest-sized* systems (tens of atoms) that are as reliable as the most detailed experiments, especially for thermodynamics, structure, and many spectroscopic properties. For systems with hundreds of atoms approximate methods can be employed to reduce the computational effort. However, establishing the validity of computational methods and models, as well as the uncertainty and limitations of the methods requires high accuracy benchmark calculations.

With the scaling of current accurate methods ranging from N^5 (MP2) to N^7 (CCSD(T)), the need for access to large computational resources for accurate predictions is evident. An example of the state of the art with the current MSCF computational hardware and software is the calculation of the heat of formation of the octane molecule, a hydrocarbon that is part of automobile fuel. The most computationally intensive part of this calculation required 1,400 processors or 8.4 Tflops for 23 hours, yielding 75 percent CPU efficiency as the kernel is mainly DGEMM operations. However, current computational resources are not enough to obtain these properties for even bigger hydrocarbons such as cetane, which is twice as large as octane and for which there is practically no crucial experimental data available. The octane calculation required 8.7 Tbytes of aggregate storage, whereas the same calculation on the cetane molecule will require at least 63 Tbytes. If the cetane calculation would be performed on today's MSCF hardware the calculation would take over 2200 hours to complete on 1400 processors. Hence, a large increase in computational resources is required to make these calculations feasible. In

general, high accuracy problem sizes that are of interest to the computational chemistry community, but cannot be addressed by the current state-of-the-art computing resources, range from 2000-5000 orbitals and 100-200 electrons. Although the size of systems for which accurate calculations can be performed is growing (e.g., accurate energetics can be obtained for clusters with more than 20 water molecules), extension of these types of calculations to more reliable models of condensed-phase systems will require significant increases in computational resources. For example, explicit high accuracy calculations of up to 100 water molecules would require an approximate 600 - 3,000 fold increase in the computational resources (storage and CPU power) depending on the method used.

In addition, the octane example above involved a single energy at a single nuclear configuration. A simple kinetics calculation at this level of accuracy for a similar sized system would require finding the stationary points (reactant, transition state, and product), which requires approximately thirty energy evaluations as a conservative estimate. Hence, one would need approximately 200 Tflops to obtain the kinetics of one reaction involving the octane molecule.

Another drive is to perform simulations on more realistic models with fewer approximate methods. Although the size of systems for which accurate calculations can be performed is growing (e.g., accurate energetics can be obtained for clusters with more than 20 water molecules), extension of these types of calculations to reliable models will require significant increases in computational resources. For example, condensed-phase chemical problems involve hundreds to thousands of atoms, and can include computationally expensive atoms such as transition metals or heavy elements.

Another example requiring large computational resources is the modeling of the energetics of large molecular systems such as metalloproteins or nanoparticles. The structure of a protein determines its functionality and activity (for example biological processes for the transfer of electrons and energy in proteins in cells), and the structure of a nanoparticle controls its properties. Researchers are currently pushing the boundaries modeling a 690 atom metalloprotein that consists of 8770 basis functions at the low accuracy HF level of theory. This calculation requires the recomputation of the integrals for every iteration as there is simply not enough storage on the machine to hold them all.

The optimal determination of molecular structures of these large molecules is a difficult problem and currently requires repeated sampling of the potential energy surface. The computational effort scales nonlinearly with dimensionality of the system. For example, in the most simpleminded approach of sampling over n points in each dimension, the effort scales as n^D, where D is the number of degrees of freedom (formally 3 times the number of atoms) in the system. For the condensed-phase or chemical interfaces this may involve systems of hundreds and possibly thousands of atoms.

The calculation of dynamic properties, such as diffusion, energy transfer, and chemical reaction rates, requires following the motions of atoms and molecules as a function of time. Treating the atoms classically, atomic motion is followed by integrating the classical equations of motion to determine the classical trajectory of the system. Instead of following a single trajectory, one can also calculate an ensemble of trajectories for a chemical dynamics simulation, and this ensemble may be as small as 50 trajectories or as large as several thousand trajectories, depending on the nature of the problem.

Parallelization of such a multi-trajectory simulation is straightforward by simply running multiple trajectories in parallel on groups of processors.

For each time step in each of these classical trajectories the forces on the atoms need to be evaluated on the potential energy surface. Depending on the accuracy of the method used do the evaluation, with a computational expense ranging from order N or N^7, each time step could take seconds or minutes to many hours on current MSCF hardware resources. For these types of simulations, hours is simply to long. Minutes may be too long as well. In addition, time scales for dynamical properties of interest, especially rare events such as passing over a barrier in a chemical reaction or gas-to-liquid nucleation, can be much larger than the inherent time scales in the order of 10^{-15} seconds of motions in molecules, requiring large numbers of steps along the trajectory. The computational effort scales linearly with the number of time steps in a calculation, which could be in the order of 10^3-10^{10}. Current methods, such as Car-Parinello dynamics, have provided unique insights into dynamic properties on short-time scales, yet the electronic structure on which the dynamics are based is only approximate and the time scale is very short. Yet, such methods are needed to explore reactivity in solution and biochemical systems.

3 Environmental Systems Sciences

Knowledge of the chemical and biological reactions of environmentally important processes can be used to resolve critical DOE environmental challenges such as bio-remediation and carbon sequestration. Thus, large multi-scale subsurface and climate models must be developed to simulate long-term events that may influence policy decisions concerning natural and human impacts on the environment. Challenges representing subsurface and climate modeling are often intertwined—for example, atmospheric modeling may feed into watershed and surface water models that link to the subsurface.

3.1 Algorithmic information

Both climate modeling and subsurface chemistry model information laid out on three dimensional grids that evolve in time and are distributed over a collection of processors. Those grids may be refined globally or locally to include multi-scale computations and may move to follow fronts. Most of the computations involve large sparse matrix vector products and large vector-vector operations that may involve contiguous, strided, or indexed memory accesses. The wall clock time required to run these simulations requires that history files containing all of the data necessary for a restart be generated at regular intervals.

For both the climate and subsurface domains, the low level nature of serial operations involved tend to be dot-product, and vector-triple operations like DAXPY.

The subsurface chemistry arena needs small dense matrix solves of which tens of thousands to millions are done, with matrices modified at each step of a nonlinear outer iteration. It also utilizes large sparse system solves such as those provided by the PETSc library to solve linear sub-problems of Newton-like nonlinear solvers.

Both domains have advection and diffusion of species (chemical and/or precipitate) over their modeling space. This feature along with the large sparse system solves require frequent nearest neighbor style communications. The size of the communications tends to increase with the square of a linear spatial dimension, hence with the 2/3 power of the product of the spatial dimensions.

Both of these environmental system simulation areas have data that needs to be aggregated into a single result and then redistributed at frequent intervals. These aggregations and broadcasts are used to guide future steps in the computation. They also both share a multi-scale flavor with the desire to run much finer resolution grids on parts of or all of their domain.

3.2 Hardware balance requirements

The vector triples operations, dot product operations and sparse matrix operations are all memory bandwidth limited. The rate limiting component of the computations is not how fast floating point operations on register arguments can be performed. The speed at which operands can be loaded into the floating point registers from memory is the current bottleneck. There is very little data reuse in existing codes which are millions of lines long. Most operations could utilize two to three double words of memory bandwidth per machine clock (16 to 24 bytes/cycle). Current architectures can only deliver a fraction of

that – one to two bytes per cycle. Lots of high speed cache, with at least 4 way associativity and large numbers of floating point registers can help to hide this latency some what. It is also important to be able to queue enough outstanding memory requests to allow prefetching to help hide the high latency to memory. Currently some of the climate codes achieve about 10% of peak floating point speed on our Itanium IA64 cluster while the subsurface chemistry and regional climate codes only get 3% to 5% of peak efficiency.

The history keeping features of the climate modeling codes and the basic nature of output for spatial grids progressing through time with several variables per grid point calls for significant parallel input/output capacity on each processor. A globally visible file system that allows multiple writers and readers per file and has an efficient file locking mechanism is required. A significant portion of time (5% to 30%) for these runs is spent writing history data for restart and output data making asynchronous I/O capability critical. The large volume of output from these applications also requires an archive system capable of handling tens of Tbytes of data per day for long term storage. That archive file system will need to be network available for remote user access at very high bandwidths (minimally tens to hundreds of Mbytes per second).

The advection and diffusion of material species in both climate and subsurface modeling as well as the sparse matrix-vector products in the large sparse linear system solves for the subsurface modeling arena will require low latency point to point communications for nearest neighbor style communication patterns. Sufficient bandwidth for all processors to be able to communicate concurrently in this point to point capacity is important. Non-blocking, one-sided communications (with remote direct memory access and zero-copy communications) will be important features.

Currently all of the environmental systems modeling codes using the MSCF require the MPI paradigm for communicating, so MPI 1.1 compliance is required and MPI 2.0 capabilities are desired. The other flavor of communication that is critical to the applications is the global reduction and broadcast operation for global dot products. This global reduction and broadcast is done several times per nearest neighbor communications and requires low latency communication that should only grow logarithmically in elapsed time with the number of processors. The ability to aggregate results over subgroups of processes will also be important as underlying algorithms are migrated to resource fault tolerance.

The problems of the future in the environmental systems arenas will grow by two to five orders of magnitude. Thus they will need to run for longer periods of time on more components increasing the likelihood of at least one component failing during the execution of a single run. This suggests that some form of hardware redundancy in the form of hot spare processors and memory, raided disks for global file systems and possibly communication logging capacity will be desirable. Also crucial in the capacity to provide fault resilience will be resource managers capable of detecting hardware faults and allowing computations to proceed in spite of them. Substantial effort from the application side will also be required.

3.3 Hardware capacity requirements

The three-dimensional Hanford Sitewide Groundwater Model currently uses inverse simulations to identify geologic zonation and hydrologic parameters, including boundary conditions. Future pilot-scale analyses for the Hanford Site will require significantly more (10 to 100 times) grid resolution, will include more detailed transport processes (e.g., chemistry), and will address 10 to 100 times more estimated parameters. These additional capabilities could require more than a 1,000 times increase in the number of flops to be done, and storage requirements can be expected to increase by a factor of 10 to 100.

Multiscale and hybrid approaches (molecular-scale modeling of mineral surfaces, in silico models of metal reducing bacteria, and pore-scale models of multiphase flow and multi-component reactive transport) coupled with longer time simulations and increasing from two to three dimensional models, are expected to increase computational requirements by three to five orders of magnitude. In addition, parameter estimation and uncertainty computations will add one to two more orders of magnitude to obtain the reliability necessary for policy decisions.

For example, the Hanford Site composite analysis encompasses several different models and solution techniques that address multiple environmental pathways on the Hanford Site. The risk analysis is based on 1,000-year simulations of 14 contaminants originating at 1,053 waste sites using 100 realizations to address variability and uncertainty in the parameter space. A need exists to (1) increase dimensionality and resolution in the modeled areas to better represent the existing and future plumes; (2) incorporate additional process models (e.g., multicomponent geochemistry to reflect spatially and temporally variable contaminant concentrations, and bank storage to identify release and migration of contaminants through seepage faces); (3) add more sophisticated time-dependent boundary conditions; and (4) increase the number of realizations to account for more of the parameter space. These additional capabilities could increase the computational requirements by 100 to 1000 times the current use.

The memory requirements for the subsurface chemistry codes are currently 200 Gbytes to 300 Gbytes (or about 1 Gbyte per processor). With increases in spatial refinement and multiscale models, this can be expected to grow to 2 Tbytes to 30 Tbytes of memory per run. Systems with less memory will require out of core solutions which tend to be 10 to 100 times slower on existing architecture balances and hence would require very high bandwidth, and lower latency globally visible disk solutions. These codes currently generate 10's of Gbytes of simulation output per run and can be expected to generate Tbytes of data per run as they grow in complexity. The total volume of data communicated will grow approximately two orders of magnitude in moving from the two dimensional simulations to three dimensional simulations. Currently, the size of the nearest neighbor communications are tens of Kbytes to hundreds of Kbytes per message per neighbor, for a total volume of tens of Mbytes per nearest neighbor communication. Increasing the volume by one hundred times increases the need to be able to do non-blocking one-sided (preferably zero-copy) communication.

In the last few years, a new approach to the treatment of clouds has replaced conventional cloud parameterizations in a few models. This new approach consists of embedding a

cloud resolving model (CRM) in each climate model grid cell. Each CRM explicitly resolves the cloud circulation and its influence on microphysical and radiative processes. The resulting coupled system is often called the Multiscale Modeling Framework (MMF). Preliminary results showing quantitative improvements in climate simulations suggest that the MMF methodology will become the approach of choice for estimating the radiative forcing by anthropogenic greenhouse gases and aerosols, investigating physical feedbacks in the climate system, and understanding the relationship between global climate forcing and regional hydrological changes. It is a computationally expensive solution, increasing the run time of a simulation by a factor of 200 in coarse resolution models. However, because the CRMs dominate the computations and interact with each other only through the outer spatial (coarsest resolution) grid, the MMF scales with respect to the number of processors much better than conventional climate models, permitting efficient parallelization on thousands of processors on some computing systems. The MSCF, with its concentration on chemical and physical processes such as aerosol chemistry, is used to improve and develop the physics and boundary conditions of the model clouds in the CRM.

Climate modeling with computationally intensive models began in the 1960s and has been limited by computing resources during its history. Even development of the improved model, which is the focus here, requires large computational resources for correcting and validating changes in the models. Current global climate models have been parallelized to run on hundreds of processors at coarse (200-kilometer) resolution and thousands of processors at fine (10-kilometer) resolution. Most climate simulations are run at coarse resolution to permit multicentury simulations and to store the resulting data. Fine resolution simulations are severely limited by the lack of data storage ability. Coarse resolution simulations are limited by the cost of communication, which in turn limits scalability to less than 1,000 processors. This feature again highlights the need for the ability to overlap communication with computation via non-blocking and one sided communication capacity.

Future simulations with advanced and improved versions of the MMF will require enhanced computational resources. Refining the resolution of the CRMs by a factor of four will require at least a 16-fold increase in computing power and storage to achieve the same simulation throughput. Three-dimensional cloud geometry would increase the computational needs by another factor of 1.5. Refining the global model grid size by a factor of two will require another factor of four-increase in computing power and storage. Thus, future improved MMF simulations will require a 100-fold increase in computing power to maintain the same simulation throughput.

Some improvement in the throughput can be achieved by using a larger number of processors. At the current 2^{o}x2.5^{o}xL26 resolution of CAM, the number of processors is limited to 240. Increasing the number of processors can be accomplished in the model either by increasing the resolution of CAM (which would render the MMF even slower) or by using a different domain decomposition that permits up to 8,000 processors for the same grid resolution. Using a different domain decomposition and allowing columns to be shared across processors and/or using a task scheduled approach could thus provide a 30-fold increase in time-to-solution of the simulation. The processor count also could be increased by increasing the number of processors sharing memory on a single node. A

factor of 16 is unrealistic, but a factor of 2 to 4 might be achievable. Thus, a 100-fold increase in throughput is achievable using next-generation microprocessor-based machines.

The amount of memory currently used by MMF models is 30 Gbyte to 250 Gbyte per run (about 1 Gbyte per processor). Refinements of grid spacing in the horizontal directions and moving towards three dimensional modeling on the cloud scale will require terabytes to tens of terabytes of memory. Memory latency for the CAM is an issue, limiting the performance of climate models on high-performance computing systems to only five to ten percent of peak theoretical performance; however, the MMF was developed for cache based machines and consumes approximately 90% of the computational time for the simulations. Although the standard CAM has been vectorized to run efficiently on vector machines, the MMF version of CAM has not. Further work would be required to develop a vector version of the MMF that can run efficiently on vector systems.

Replacing the cyclical lateral boundary conditions in the CRMs with one that allows propagation between CRMs presents a significant communication challenge to the computing system. The frequent communication requires a low latency interconnect. A significant restructuring of the MMF code will also be required to ensure that message passing between CRMs are only required if the CRMs reside on different processors or nodes (the latter can consist of multiple processors).

Volume data for model history will become an issue at high resolution. Currently, climate modeling history is comprised of 12 Gbytes per simulated day, or 4.4 Tbytes per simulated year. This would increase to 50 Gbytes per day (18 Tbytes per simulated year) at 1 kilometer CRM resolution. Thus, 20 years of simulation will produce a total of 360 Tbytes of data/run. Needing an ensemble of runs for model verification will easily reach the Pbyte range of data production. That data needs to be accessible to on site and remote users for subsequent analysis and visualization.

4 Summary

An analysis of the algorithms in all the three science areas has been presented in the three sections above. The analysis shows that a *balanced* architecture is needed with respect to processor, memory hierarchy, interprocessor communication, and disk access and storage. A single architecture can satisfy the needs of all of the science areas, although some areas may take greater advantage of certain aspects of the architecture.