Extending The Scientific Template Library With A Shared-Memory Non-Uniform Memory Access Programming Model Coupled With HPF Language Support

Abstract: In this proposal, we build upon a set of powerful and proven paradigms for managing the complex memory hierarchy in MPPs, and fully integrate that functionality into key areas of the SciTL toolkit, greatly extending its capabilities for applications requiring non-uniform memory access and for computer systems having non-uniform memory architecture (essentially all present and future MPPs). Our approach combines the strengths of the shared-memory-programming model with the scalability and performance of the distributed-memory model. What has emerged is a highly successful Global Arrays model (GA), whose one-sided communications model yields clean code with outstanding performance and scalability, as well as compatibility with standards and common linear algebra packages. Current applications span a wide range, including molecular dynamics, computational chemistry, graphics rendering, and security value forecasting. GA is used by major centers and applications development groups nationwide and worldwide. In the spirit of a standards-based approach, this toolkit will be integrated with High Performance Fortran, in a completely standards compliant and portable fashion, so that HPF based applications can readily exploit both data and task parallelism on NUMA architectures.

TABLE OF CONTENTS

1 Background and Significance

All current and proposed MPPs (e.g., ASCI computers) have or will have a non-uniform memory architecture. The hardware in a particular system may support a shared-memory or message-passing programming model, but these factors affect only the relative costs of local and remote accesses and do not affect not the system's fundamental non-uniform memory access (NUMA) characteristics. In future systems, including the ASCI systems with their shared-memory multiprocessor (SMP) nodes, both the number of memory levels and the cost in processor cycles of accessing deeper memory levels can be expected to increase **[26,29]**. Essential to achieving high efficiency on such systems are non-uniform memory-access (NUMA) aware programming tools.

Although the efficient management of memory hierarchies is fundamental to high performance in scientific computing, existing parallel languages and tools provide only limited support for this management task **[31,24]** and focus primarily on the issues of control flow, communication structures, and load balancing. Memory-related issues such as data locality and variable cost of accessing data have been largely ignored by a flat memory model (see for instance references [Fortran90,PCN,DataParallelC,27]). However, this simple memory model hinders the understanding and tuning of parallel program performance. The variable cost of accessing data that reside at different memory levels cannot be ignored if we are to construct efficient parallel programs.

Systems that address locality management include High Performance Fortran (HPF) **[hpfbook],** the Message Passing Interface (MPI) **[MPI]**, Split-C [10], and Compositional C++ [8]. HPF augments Fortran 90's flat memory model with data distribution directives, which can provide an implicit specification of locality. However, HPF can specify only data-parallel algorithms, and requires advanced compiler technology. MPI distinguishes local and remote memory explicitly, but requires that remote memory access be performed by means of explicit send calls with matching receives. Split-C and CC++ use global pointers to represent references to remote data.

Recognizing the deficiencies of the existing programming models, we developed abstractions and programming tools, that can facilitate the explicit management of memory hierarchies by the programmer, and hence the efficient programming of scalable parallel computers. These abstractions comprise local arrays, global (distributed) arrays, and disk resident arrays located on secondary storage. The tools comprise the Global Arrays library [21,22,23], which supports the transfer of data between local and global arrays, and the Disk Resident Arrays library [18], its extension to high-performance secondary storage, for transferring data between global and disk resident arrays.

The Global Array (GA) NUMA programming model and supporting toolkit has been proven as an effective and easy-to-use parallel programming tool for developing real large-scale applications [4,5,11,15,32]. These applications were able to obtain impressive speedups and scalability on massively-parallel computers thanks to support that GA and Disk-Resident Arrays (DRA) provide for high-performance programming in NUMA environments: control of data locality, recognition of the memory hierarchy, and explicit mechanisms for such transfers. GA is already optimized for shared-memory multiple processing SMP nodes of MPP computers and should match very well the architecture of the ASCI systems.

From the user perspective, GA does not require expert skills in parallel programming. The high-level array-oriented API is close to the matrix notation that describes mathematical formulation of physical problems. Scientists who are not parallel programming experts find GA operations on arrays much more familiar, easier to learn and use than the low-level message-passing model. Efficient and scalable parallel algorithms are also readily designed and implemented, since any necessary blocking of accesses to data is fully supported by GA. These attributes of GA are the keys to achieving very-high productivity levels in the development codes using GA, such as NWChem [6].

From the system architecture perspective, GA is a memory model and as such, it can deliver the highest possible performance because its operations map directly to hardware capabilities (designed to move data between different memory locations), avoiding the additional S/W overhead from abstractions such as threads or active messages. On a shared-memory machine, native shared-memory operations are used (e.g., CRAY T3E, SGI SMP). On a distributed-memory machine, the highest-performance message-passing tools are used. On a cluster of SMPs, shared-memory access is used wherever possible, and otherwise massage passing is used. GA programs are fully portable between all of these architectures. Furthermore, the performances of such parallel programs are readily modeled in terms of the fundamental machine characteristics (e.g., latencies, bandwidths, etc.) and therefore are easier to understand and optimize.

Global Arrays complement other tools in SciTL. Unlike P++ (and toolkits based on P++), that targets data-parallel execution model, GA supports both task- parallelism and data-parallelism. Unlike Tulip [3] that uses polling for remote memory access on message-passing systems (such as IBM SP-2), the GA implementation uses platform-specific low-level communication mechanisms that do not require polling. While polling is feasible in a run-time system supporting languages used to develop complete applications, it is not practical for many applications that use large standard libraries. This is because it would require modifications (inserting polling calls) to *all* of the application code including such libraries [17].

High Performance Fortran is a parallel language that is highly relevant to the DoE scientific community, where most application codes are still written in Fortran. It offers advantages of being an accepted industry standard, with vendor support, and a wider user-base. Unfortunately, HPF-1 is limited to data parallel operations, while many algorithms including computational chemistry and subsurface transport simulations have both and task parallel components. A simple example would be a common second order perturbation method used computational chemistry, called MP2. The first part of such a calculation is a “four-index transformation”, which is task parallel with an on-the-fly transposition of a distributed, sparse matrix. It involves a complex, nested, loop structure, which can be executed in parallel. Because task durations vary widely, dynamic load balancing of this loop structure is essential to obtain reasonable performance. The second major part of an MP2 calculation, the “energy evaluation,” is readily implemented in a data parallel fashion.

A revision of the HPF standard is now under consideration, which will include some support for task parallel operations, reduction variables, and more general data distributions. Unfortunately the designers of HPF-2 resisted introducing any non-deterministic features (needed to support dynamic load balancing) into the language, so the HPF-2 implementation of task parallel algorithms such as the MP2 four-index transformation would have to rely on the INDEPENDENT directive and on the compiler and runtime to provide high performance. Given the current philosophy of the language designers, and the complexity of compiler- or runtime-level support for dynamic load balancing, it is unlikely that acceptable performance could be obtained from mass-market implementations. The most feasible approach to supporting task and data-parallelism, while taking advantage of language support, is to integrate GA with HPF in order to take advantage of the best features of both.

We propose two major activities:

- The integration of the Global Array NUMA programming toolkit into SciTL, including full interoperability with key existing SciTL components, and the extension of the Global Array toolkit to provide enhanced support for applications outside of the domains in which it is already extensively used. This work will be conducted primarily at PNNL with input and feedback from the SciTL user community driving requirements analysis for the extensions to be performed.
- The standards based interoperability and integration of GA and HPF. It will include development of GA library interfaces to HPF arrays to support task-parallelism in the framework of HPF programs. In the next phase, language support for such interfaces will be developed to replace explicit GA library calls with language constructs supported with extended HPF run-time system and GA library. This work to be primarily performed at NPAC in collaboration with PNNL.

Integration with SciTL will require definition of data descriptors and data distributions conforming to PADRE specifications, which in turn will allow for the coupling of GA or HPF/GA applications with other SciTL modules within the PAWS environment.

In summary, GA would complement and extend the capabilities of tools in the current SciTL by providing a NUMA-aware, portable, efficient and easy to use shared memory programming model fully compatible with MPI and interoperable with HPF.

In the following subsections, we provide some more background material about GA and DRA.

1.1 Global Arrays

The two predominant programming models are message passing and shared memory. The message-passing programming model is widely used because of its portability, yet some applications are too complex to code in it while also trying to maintain a balanced computation load and avoid redundant computations. The shared-memory programming model simplifies coding, but it is not portable and often provides little control [1,2,7,Linda] over interprocessor data transfer costs. The Global Arrays (GA) combine the better features of message-passing and shared-memory, leading to both simple coding and efficient execution for a class of applications that operate on large distributed dense arrays, require dynamic load balancing or exhibit attendant unpredictable data reference patterns. The Global Array toolkit provides an efficient and portable “shared-memory” programming interface for distributed and shared-memory systems. Unlike other shared-memory environments, the GA exposes to the programmer the nonuniform memory access (NUMA) characteristics of high performance computers, acknowledges that access to remote data is slower than to local data, and it allows data locality to be explicitly specified and used. From the user perspective, a global array can be used as if it were stored in shared memory, except that explicit library calls are required to access it.

Development of Global Arrays has been motivated by real applications. Current applications of GA include graphics rendering [16], financial calculations (security value forecasting), molecular dynamics **[Tjerk]** and many computational chemistry algorithms which calculate the electronic structure of molecules and other small or crystalline chemical systems. Such calculations can predict many chemical properties that are not directly accessible by experiments, and play a dominant role in the number of supercomputer cycles currently used for computational chemistry. Some examples of applications that attain high parallel scalability and demonstrate the ease of use of GAs are included in the following references **[4,5,6,7,8]**.

GA were developed as part U.S. Federal High Performance Computing and Communications Initiative (HPCCI) DoE Grand Challenge Applications project in computational chemistry, and turned out to be useful for research and production code development around the world: at PNNL’s Environmental Molecular Science Laboratory (EMSL), elsewhere in the U.S. (including Argonne National Laboratory, NCSA, Pittsburgh Supercomputing Center, Cornell Theory Center, San Diego Supercomputing Center, University of Illinois Urbana Champaign), and other countries including United Kingdom (Daresbury Laboratory), Austria (University of Vienna), Germany (Stuttgart University), Australia (Australian National University), Sweden (University of Lund) and Italy (University of Perugia). In last three years, many applications have adopted GAs, totaling nearly 1,500,000 lines of code, including some 400,000 lines of new code.

Important to the successful employment of the SciTL in the DoE scientific community and beyond, is its compatibility with languages, libraries and tools that scientists use. Most scientific applications have been and continue to be developed in Fortran. It is important for the tools to support this language and follow its evolution. In particular, the High Performance Fortran (HPF) is a new industry standard for developing data parallel programs. While several present SciTL object-oriented programming toolkits implemented in C++ offer some interfaces to Fortran, their main strength can only be exploited within the C++ object-oriented framework. Although GA has been designed using object oriented principles it does not suffer from the performance problems associated with C++ implementations (e.g., the reported 50% performance loss compared to the Fortran implementation [27]). Some of these problems can be addressed by expression templates however the sophisticated C++ compiler technology required is not yet widely available. GA offers its full capabilities and equally good performance in both Fortran and C/C++ applications.

The GA was developed in light of emerging standards: the API follows Fortran 90 array notation, the library has been designed based on object oriented principles, GA is compatible with MPI, and has interfaces to parallel linear algebra packages such as ScaLAPACK or PLAPACK. The programmer is free to use both shared-memory and message-passing paradigms in the same program and to take advantage of existing message-passing software libraries. Extending the interoperability of GA with other tools and facilities is a subject of this proposal and will significantrly enhance the value of GA to DoE projects.

The GA are widely available for variety of computer systems:

- distributed-memory, message-passing parallel computers with interrupt-driven communications or Active Messages (IBM SP and Intel Paragon);
- networked clusters of single- and multi-processor UNIX (Sun, IBM, SGI, DEC, HP) and Windows NT workstations;
- shared-memory computers (Convex/HP SPP, SGI Origin-2000 and Power Challenge); and
- globally-addressable distributed-memory computers (Cray T3D/E).

Several computer vendors are collaborating with PNNL researchers in developing optimized ports of GA. For instance, IBM has provided a new low-level API (LAPI) with the sole-purpose of supporting GA-like operations. This API was designed in recognition of the very broad set of applications requirements that the GA model addresses. Cray Research and PNNL are currently cooperating on optimizations of GA and DRA libraries for the current and forthcoming Cray/SGI systems. In the past, Intel and PNNL developed extensions to the OS to better support GA on the Paragon.

1.1.1 Mirrored Arrays: metacomputing extensions of GA

In recent years there has been increasing interest in metacomputing. Metacomputing environments comprise multiple supercomputers and other devices (mass storage systems, display devices) connected via local area networks (LANs) or wide area networks (WANs). They can potentially provide significant increases in the computational power accessible to a single application. From the perspective of the memory hierarchy, a metacomputing environment introduces one or more additional layers, typically with much higher latencies and lower bandwidth than in a supercomputer. Mirrored Arrays [19,20] have been developed to address the performance characteristics of metacomputers in a manner similar to replicated shared memory [25]. Arrays are fully distributed within each machine, so the amount of data held by each processor increases by a factor roughly proportional to the number of networked supercomputers. Each supercomputer operates on its own mirrored array independently, and a special operation is provided for enforcing consistency of the different mirrored arrays. This primitive is a collective operation across all supercomputers that merges user-specified sections of the mirrored arrays. Upon completion of this operation, all machines have identical copies of the specified array or array section. The result is that most communication occurs locally within the supercomputers. Total network traffic is lower and average network message size is larger than in simple distributed approach. The mirrored arrays approach was very successful for a large application (Self-Consistent Field) that was able to effectively exploit multiple supercomputers connected with WANs (Intel Paragons and IBM SPs).

1.2 Disk Resident Arrays: Extension of Global Arrays to Secondary Storage

Disk Resident Arrays (DRA) extend the GA model to another level in the storage hierarchy, namely, secondary storage. DRA introduces the concept of a disk resident array—a disk-based representation of an array—and provides functions for transferring blocks of data between global arrays and disk resident arrays. Hence, it allows programmers to access data located on disk via a simple interface expressed in terms of arrays rather than files. It extends the benefits of global arrays (in particular, the absence of complex index calculations and a high-level array interface) to programs that operate on arrays that are too large to fit into memory.

By providing distinct interfaces for accessing objects located in main memory and on the disk, GA and DRA consistently expose the different levels of the memory hierarchy in which objects are stored. Hence, programs can take advantage of the performance characteristics associated with access to these levels. Recall that memory hierarchies consist of multiple levels, but are managed between two adjacent levels at a time [26]. For example, a page fault causes the transfer of a data block (page) to main memory while a cache miss transfers a cache line. Similarly, GA and DRA allow data transfer between adjacent levels of memory which combined with collective I/O strategy made this model very efficient [18**,**9] on the parallel I/O systems. DRA read and write operations can be applied both to entire arrays and to sections of arrays (disk and/or global arrays); in either case, they are collective and asynchronous. For the efficiency reasons, this collective I/O strategy has been adopted in many other projects [**MPIO**, **PASSION**, **Panda**] as well. The DRA asynchronous interface to I/O operations permits applications to overlap time-consuming I/O with computation. The DRA has been successfully used in out-of-core matrix multiplication and two chemistry applications [14].

2 Preliminary Studies

The Global Array concept has been already thoroughly tested in a variety of application programs with some containing hundreds of thousands lines of code.

The functionality extensions (support for higher dimensional arrays and split-phase interface) that this proposal describes fit well within the design framework.

Some interfaces between GA and third-party linear algebra libraries (ScaLAPACK, SUMMA, PLAPACK) have been developed but not all of them have been thoroughly tested in the application codes. Further work is required to make these interfaces complete (for example providing full BLAS-3 functionality through PLAPACK) and testing to make them robust and suitable for production code development.

The interface to HPF has been recently investigated and we have determined that the language standard provides sufficient information necessary to retrieve distribution information and construct a GA descriptor associated with an HPF array. Also, the EXTRINSIC interface of HPF has been used before in other projects [36] and was proven as a reliable mechanism for interfacing HPF applications with non-HPF software.

Our recent study at NPAC determined that the recent HPF-2 standard, which is nearing approval, would provide some support for task-parallelism in chemistry applications. Unfortunately, the new features are not sufficient to preserve and support dynamic load balancing schemes in these applications. The code restructuring involved in translation of a sample computational chemistry algorithm, implemented with GA, to HPF-2 required replacement of load-balancing, task-farming loop with deterministic parallel loop. It is not clear if optimizing HPF-2 compilers would be capable of performing automatic dynamic load balancing when translating such loops. Moreover, HPF compilers often concentrate on parallelizing simple loops, where communication can be reduced to cooperative or collective operations. On encountering a complex loop where communication cannot sensibly be reduced to such simple patterns, a compiler may well fall back on a default strategy of sequentializing the loop. These considerations indicate that Global-Array shared-memory operations will be needed even when HPF-2 compilers become available. In particular, coupling the GA library with a standard, vendor-supported HPF (2) compiler could offer a powerful tool combination capable of supporting task- and data-parallelism more effectively than any of the both facilities would be capable of alone.

We now examine briefly the two major application areas that have driven the current development of GA the prototype HPF-GA preprocessor.

2.1 Computational Chemistry

Global arrays were originally designed to support computational chemistry applications which fall into two classes; the electronic structure of molecules and solids, and the classical molecular dynamics simulation of macromolecules and fluids.

In electronic structure most matrices are dense, one- or two-dimensional arrays, and, to-date, higher-dimensional or sparse arrays have been packed into dense one-dimensional structures. Nearly all of the algorithms are parallelized inside nested loops that require mangement of sparsity and dynamic load balancing, and map to dense matrices at the lower levels of the program. The original requirements for GA were simply to facilitate access to distributed matrices from MIMD parallel code, including *put*, *get* and *accumulate* (reduction) operations. As the development proceeded, other primitive MIMD operations were required including a large selection of data-parallel operations and interfaces to third party libraries. Most of the parallel electronic structure algorithms have successfully used blocking of nested loops to improve data reuse, thereby providing the tolerance of latency necessary for high efficiency on massively parallel machines. Global arrays have been employed in several manners:

- Distribution of matrices in dynamically load-balanced MIMD loops with simultaneous update (accumulate) into overlapping regions.
- As a convenient framework to create data parallel operations and manage distributed data.
- To map logically shared-files into distributed global arrays providing rapid access to the aggregate resources of the entire parallel machine.
- As an easy-to-use interface into complex parallel BLAS and linear algebra libraries.

- As a tool for rapid data-redistribution – the one-sided access makes it much easier to design efficient, application specific, redistribution algorithms than, for instance, using message passing.

Molecular dynamics typically has a much smaller ratio of computation to data then electronic structure, and is therefore sensitive much more sensitive to management of data locality, and both the bandwidth and latency of communication. The very capable NWChem molecular dynamics code [30] makes extensive use of GA to implement a spatial-decomposition algorithm **[reference]**. The simulation data (atomic coordinates, velocity, forces, connectivity, etc.) are distributed by mapping to a virtual array of processors. The major advantages provided by GA in this instance, are increased asynchrony of execution due to the one-sided access to data, and improved ease-of-implementation since GA can handle most of the complexity of addressing the distributed data. The one-sided communication also facilitates the data redistribution that must occur during dynamic load balancing, as "ownership" of atoms is transferred from one processor to another.

2.2 Coupled fluid flow, transport and chemistry in porous media

The major application area here has been the simulation of contaminated subsurface sires for remediation, e.g., modelling the migration of contaminant plumes, interaction with local biogeochemical environment and the impact of remediation stresses. A continuum model is used and material properties of the solid matrix are typically highly variable with features that range over the entire spectrum of length scales ($\sim10^{-2}$–10^{3} m). In these systems, Darcy's law can typically be used to describe the bulk fluid movement. The continuum is discretized into grid cells using finite elemnent or finite difference schemes, and the numerical treatment of flow, transport and reactions is similar to computational fluid dynamics. Each grid cell must deal with up to $O(10^2)$ unknowns and adaptive mesh refinement is necessary to recover important features of contaminant behaviour and also to improve the overall computational efficiency. A simultaneous globally implicit solution for all unknowns results in a single matrix that is locally dense but globally sparse ($\sim10^{6}$–10^{8} unknowns) that is solved iteratively. More typically, operator splitting is used to decouple the problem, allowing the fluid flow, transport and reactions to be isolated and solved with less dependecy between them. The prinipal trade-off is between accuracy and efficieny. Material property heterogeneities are irregularly distributed in space and tend to cause computational hot spots, though these are most demanding only when the reative plume front contacts these zones. The fluid flow and transport are inherently data parallel, while the reactions require task parallelism. Load balancing is required by the automatic mesh refinement and the reaction computation.

A single program multiple data model is used with elements of both data and task parallelism. The prototype HPF–GA preprocessor is being used to provide backward compatibility with code in CM-Fortran code. It also enables new project members to make more rapid progress in a familiar programming language (HPF) before they must confront issues such as management of NUMA that might not be essential except in compuational kernels.

3 Research, Design and Methods

Although the GA NUMA programming model and toolkit has been very successful for many applications, the value of GA to DOE the scientific community can be improved. We will target improvements to the following areas:

- integration and interoperability with other tools and libraries including those already in the SciTL
- extensions of the GA functionality and programming interfaces, and
- interoperability with High Performance Fortran and language support for NUMA model

3.1 Integration, interoperability of GA with SciTL and standard tools and libraries

Many applications require access to parallel linear algebra algorithms. GA provides interfaces to the state-of-the-art parallel libraries such as ScaLAPACK, SUMMA, PLAPACK, and PeIGS. **{REFERENCES}** The interfaces hide the complexity of the solver interfaces (and sometimes variation from one release to another). The user only specifies an array descriptor for input and output and GA handles the rest. The data is rearranged to the format required by the particular library, and GA makes a call to the appropriate solver, and returns the results in the output array. The GA already furnishes interfaces to several parallel linear algebra algorithms available in ScaLAPACK, including LU and Cholesky factorizations, linear equation solvers, and matrix inverse.

PETSc **[reference]** is set of numerical libraries and data structures developed for solution of partial differential equations. In addition to the distributed data structures, it contains iterative solvers, preconditioners, and unconstrained minimization algorithms.

We propose

- providing additional GA interfaces to ScaLAPACK and BLAS-3 matrix operations in PLAPACK,
- develop an interface between GA and PETSc to make the numerical algorithms supported by PETSc available to GA applications. By combining PETSc with Disk Resident Arrays in the GA framework, out-of-core solvers can be provided, and
- define data descriptors and data destributions for GA and HPF arrays conforming to PADRE specificatins to allow coupling of GA/HPF applications in PAWS environment.

3.2 Proposed extensions and improvements to GA

3.2.1 Higher-dimensional arrays

The current implementation of GA provides support only for 1- and 2-dimensional dense arrays distributed (uniformly or nonuniformly) in the block fashion. This is a significant limitation for computational fluid dynamics and other applications with 3-dimensional problem domain. Such applications require support for arrays with three- to five-dimensions to represent the physical quantities such as temperature, velocity, pressure, etc in a 3-dimensional spatial domains. We propose to extend the GA to support arrays up to seven dimensions (the Fortran limit). This is a natural extension of the current toolkit capabilities and will fit well in the current design framework.

3.2.2 Additional distribution types

To improve static load balancing, many parallel linear algebra algorithms adopt block cyclic distribution. The current interfaces between GA and parallel linear algebra packages, such as ScaLAPACK, that use such distributions require data reorganization. For some applications, such reorganizations could be avoided if GA supported block cyclic distribution.

3.2.3 Support for sparse data structures

The current GA applications handle sparse data structures by explicitly packing data into one- or two-dimensional dense arrays. Support for sparse arrays in GA would greatly simplify implementation of such algorithms. The details of the sparse storage would be encapsulated in the GA object and access to the data simplified.

3.2.4 Split-phase operations

In order to improve latency tolerance in systems with multiple layers of NUMA memory hierarchy we propose to extend GA shared-memory communication operations with split-phase operations. The non-blocking interface to the one-sided communication operations will be developed. This technique will allow overlapping computations with communications in the applications that are capable of using such mechanisms.

3.3 Interface to HPF and language support

Most scientific applications have been and continue to be developed in Fortran. It is important for the programming tools and libraries to support and interoperate with this language, and follow its evolution. In particular, High Performance Fortran (HPF) is a new industry standard for developing data parallel programs. Based on Fortran 90 syntax, it introduces a portable way to explicitly express parallelism in the code. Intuitive data mapping directives and a global name space greatly simplifies development of parallel applications. One of the greatest strength of HPF is that it hides low-level details of the target machine architecture from the programmer. The actual parallel implementation is done by the compilation system, with the machine-specific run-time support taking responsibility of accessing remote (off-processor) data. Therefore, the efficiency of the resulting codes depends on the sophistication of the compilation system and quality of the run-time support.

The integration and interoperability between GA and HPF will be performed in two steps:

- Implementation of GA as a HPF extrinsic library. This step includes development of a benchmark application, using the HPF interpreter [AFH97] as a dynamical software integration tool.
- Based on the experience of using the GA library within the HPF we will define language extensions to be implemented using Programmable Compiler technology.

3.3.1 Extrinsic library interface to HPF

The design and implementation of HPF reflects the current status of the compiler technology. In order to guarantee portability of the codes in the sense that performance is similar on machines with a similar computational power regardless their architecture, HPF-1 does not support directly some features which are crucial for certain classes of applications. In particular, there is no support for features required in dynamic, task-parallel programs, such as random access to regions of distributed arrays from within a MIMD parallel subroutine call-tree, and reduction into overlapping regions of distributed arrays. The HPF Forum judged such features as too specific to be included in the Fortran standard, or found it premature to guarantee performance at this time. This does not limit the scope of the language since HPF provides a portable way to incorporate libraries of parallel codes that are implemented in languages different from HPF, or following parallel programming paradigms different than that employed by HPF. The difference between the directly supported HPF features and such extrinsic procedures is that the latter are to be developed by the user (or a third party), and it is the user's responsibility to provide cross-platform portability. Development of parallel extrinsic procedures necessarily requires significant expertise in parallel computing. Therefore, we propose to develop HPF extrinsic libraries that become a part of the SciTL.

As the first step, we propose complementing HPF with the Global Arrays library. In particular, the GA non-collective one-sided communication on array sections would allow random access to regions of distributed arrays and reduction into overlapping regions of distributed arrays from within a MIMD parallel subroutine call-tree. Since GA has been designed to be as compatible as possible with HPF and GA API follows Fortran 90 and HPF array notation, most GA collective operations can be translated directly into single-line statements in HPF.

3.3.2 Language support

The concept of integration of stand-alone libraries, such as Global Arrays, with HPF can be pushed even further. One strategy is to provide support for user's extensions to the language. In this approach, a dedicated, programmable language processor translates user's directives into HPF statements. In particular, the language processor generates calling sequences for the user's run-time support implemented as HPF extrinsic library. We propose to develop language extensions that makes, from the user point of view, Global Arrays an integral part of the HPF run-time system accessible via HPF-like statements and/or directives. To implement the language processor to handle codes written in the extended HPF we are going to use the infrastructure developed by the Parallel Compiler Runtime Consortium [**PCRC**]. It includes HPF parser (and also C++ and Java) that transforms the source code into an intermediate representation (IR), and vice versa. The IR is implemented in an object-oriented fashion, and is modeled on Sage++ [Ganon]. A class library to perform operations on the IR is available as well. This infrastructure has been used by the PCRC to implement a HPF compiler [Li]. Here we propose to reimplement the ideas of the Programmable Array Compiler [28] as the class library to perform operations on IR that correspond to the language extensions. The result is then reparsed to HPF, compiled using a commercial HPF compiler, and linked against the extrinsic Global Array library. The essential concept here is that the proposed language processor is programmable; that is, the functionality of the class library is described in a Language Definition File. Consequently, the same language processor can be used to integrate other run-time libraries with HPF just by modifying the language definition. Since the PCRC infrastructure is able to process C++ and Java sources, this mechanism can also be used to integrate other components of SciTL.

3.4 Timetable for the project

The proposed activities will span three years.

In the first year the following capabilities will be developed:

- support for higher dimensional arrays and split-phase operations in GA (PNNL),
- initial GA-HPF extrinsic library interface (NPAC and PNNL), and
- definition of data descriptors and data distributions for GA conforming to PADRE specifications (NPAC and PNNL).

In addition, requirements from the SciTL applications will be gathered to set priorities for the work in second and third year (NPAC).

During the second year the activities will target:

- interfaces to other SciTL components (NPAC and PNNL),
- language extensions to support one-sided communication capabilities in the HPF framework (NPAC and PNNL), and
- sparse storage and additional distribution types in GA (PNNL).

Finally, in the third year we will:

- continue integration of GA and language support system into SciTL (NPAC), and
- develop additional GA interfaces to numerical libraries and solvers (PNNL).

4 Consortium Arrangements

The work in this proposal will be performed by a consortium comprising researchers at PNNL and NPAC. PNNL will be the home institution for the project. Both institutions will contribute significant intellectual and software resources to the project:

- PNNL expertise is in the arena of non-uniform memory access programming models, the Global Array toolkit. There are also large groups of scientists independently using and developing code for massively parallel computers in the areas of computational chemistry, CFD, and the modeling of contaminated subsurface sites. These will help maintain the applications focus that has characterized the development of GA to date.

- NPAC researchers have a long and prestigious history in parallel computation, with relevant current expertise in HPF compiler technology, run-time support, application integration, and computational steering.

There has been a close and successful working relationship between PNNL and NAPC for over two years. One of the NPAC staff was formerly a post-doctoral researcher at PNNL, and NPAC staff have performed research and S/W development under subcontract to PNNL. This subcontract is expected to be continued (independently of this proposal) for another two years (in support of the DoE HPCC project described in the Section "Other Support of Investigators).

5 Literature Cited

1. C. Amza and A.L. Cox and S. Dwarkadas and P. Keleher and H. Lu and R. Rajamony and W. Yu and W. Zwaenepoel, "TreadMarks: Shared Memory Computing on Networks of Workstations", IEEE Computer, 29:2, 1996.
2. H.E. Bal and M.F. Kaashoek and A.S. Tanenbaum, "Orca: A language for parallel programming of distributed systems", IEEE Trans. Software Eng., 18(3), 1992.
3. P. Beckman, D. Gannon, Tulip: A portable run-time system for object-oriented systems, Proc. IPPSi'96, pp. 532:536, 1996.
4. D. Bernhold and R. Harrison. Orbital-invariant second-order many-body perturbation theory on parallel computers: An approach for large molecules. Journal of Chemical Physics, 102(24):9582-9589, 1995.
5. D. Bernhold and R. Harrison. Large-scale correlated electronic structure calculations: The RI-MP2 method on parallel computers. Chemical Physics Letters, 250:477-484, 1996.
6. D. Bernhold et al. Parallel computational chemistry made easier: The development of NWChem. Intl J. Quantum Chem. Symp., 29:475-483, 1995.
7. B. Bershad and M. Zekauskas and W. Sawdon, The Midway distributed shared memory system, Proc. '93 CompCon Conference, 1993.
8. K.M. Chandy and C. Kesselman, CC++: A declarative concurrent object oriented programming notation, in Research Directions in Object Oriented Programming, The MIT Press, 1993.
9. Y. Chen, I.T. Foster, J. Nieplocha, and M. Winslett, Optimizing Collective I/O Performance on Parallel Computers: A Multi-System Study, Proc. 11th ACM Int. Conf. Supercomputing, 1997.
10. D.E. Culler and A. Dusseau and S.C. Goldstein and A. Krishnamurthy and S. Lumetta, T. von Eicken and K. Yelick, Parallel programming in Split-C, Proc. Supercomputing'93, 1993.
11. H. Dachsel, H. Lischka, R. Shepard, J. Nieplocha, and R. Harrison. A massively parallel multireference configuration interaction program - the parallel COLUMBUS program. Journal of Chemical Physics, 1997.
12. E.F. D'Azevedo and C.H. Romine, DOLIB: Distributed object library, tech. rep. ORNL/TM-12744, 1994.
13. MPI Forum, MPI-2, draft document available at ftp://info.mcs.anl.gov/pub/mpi/mpi2
14. H. Fruechtl, R. A. Kendall, R. J. Harrison, and K. G. Dyall. A scalable implementation of RI-SCF on parallel computers. Intl J. Quantum Chem. Symp., 1997 (in press).
15. R.J. Harrison, M. Guest, R. Kendall, D. Bernholdt, A. Wong, M. Stave, J. Anchell, A. Hess, R. Littlefield, G. Fann, J. Nieplocha, G. Thomas, D. Elwood, J. Tilson, R. Shepard, A. Wagner, I. Foster, E. Lusk, and R. Stevens. Toward high-performance

computational chemistry: II. A scalable self-consistent field program. Journal of Computational Chemistry, 17(1):124-132, 1996.

16. R. Heiland, Object-Oriented Parallel Polygon Rendering, Proc. Graphics & Visualization GViz'94 Conf., Sept. 8, 1994, pages 19-26. 1994.
17. K. Langendoen, J. Romein, R. Bhoedjang, H. Bal, Integrating polling, interrupts and thread management, Proc. *Frontiers'96*, pp13-22, 1996.
18. Jarek Nieplocha and Ian Foster, Disk Resident Arrays: An Array Oriented Library for High Performance IO, Proc. Frontiers on Massively Parallel Comput. *Frontiers'96*, 1996.
19. J. Nieplocha and R. J. Harrison, Shared Memory NUMA Programming on I-WAY, Proc. of IEEE High Perform. Distr. Comp. HPDC-5, 1996. (HPDC-5 Best Paper award)
20. J. Nieplocha and R.J. Harrison, Shared memory programming in metacomputing environments: The Global Array approach, The Journal of Supercomputing,1997 (in press).
21. J. Nieplocha, R. Harrison, and R. Littlefield. Global Arrays: A portable 'shared-memory" programming model for distributed memory computers. In Proc. Supercomputing 1994, pages 340-349. IEEE Computer Society Press, 1994.
22. J. Nieplocha, R. J. Harrison, R. J. Littlefield. The Global Array programming model for high performance scientific computing. SIAM News, 28(7):12-14, 1995.
23. J. Nieplocha, R. J. Harrison, R. J. Littlefield. The Global Array: A Non-Uniform-Memory-Access Programming Model For High-Performance Computers, The Journal of Supercomputing, 10:197-220, 1996.
24. J. Nieplocha, R.J. Harrison, and I.T. Foster, "Explicit Management of Memory Hierarchy", in "Advances in High Performance Computing",Eds. J. Kowalik, L. Grandinetti and M. Vajtersic, Kluwer Academic, 1997.
25. M. Oguchi and H. Aida and T. Saito, A Proposal for a DSM Architecture Suitable for a Widely Distributed Environment and its Evaluation", Proc. 4-th IEEE Int. Symp. HPDC, 1995.
26. D.A. Patterson and J.L. Hennessy, Computer Architecture: {A} Quantitative Approach, Morgan Kaufmann, 1990.
27. D. Quinlan, A++/P++ Manual, version 0.6.8, tech. report LA-CC-96-1, Los Alamos Nat. Laboratory. 1996.
28. M. Rosing, Programmable Array Compiler, http://www.emsl.pnl.gov:2080/docs/parsoft/pac.html., 1996.
29. T. Sterling and P. Messina and P. Smith, Enabling Technologies for Petaflops Software, MIT Press, 1995
30. T. Straatsma
31. E.F. Van der Velde, Book Review on `Studies in Computational Science: Parallel Programming Paradigms', IEEE Computational Science and Engineering, 2(4), 1995.

32. A. Wong, R. Harrison, and A. Rendell. Parallel direct four-index transformations. Theoretica Chimica Acta, 1996.

33. Parallel Compiler Runtime Consortium, http://www.npac.syr.edu/projects/pcrc/demo1997/index.html

34. Zhang http://www.npac.syr.edu/projects/pcrc/demo1997/DOC/compilerDoc/sc97.ps

35. Xiaoming HPFfe: a Front-end for HPF, NPAC technical report SCCS-771

36. E.Akarsu, G. Fox, T. Haupt, Particle-in-Cell Simulation Codes in High Performance Fortran, in proceedings of Supercomputing '96.

37. [AFH97] E. Akarsu, G. Fox, T. Haupt, Integrated Environment for HPF Compiler and Interpreter, http://www.npac.syr.edu/users/haupt/HPFI/

38. T. Straatsma, Molecular Dynamics chapter in NWChem User Guide, http://www.emsl.pnl.gov/docs/nwchem/doc/user/user.html

39. Message Passing Interface Forum. MPI: A Message-Passing Interface. University of Tennessee, Knoxville, Ten., May 5, 1994.

40. K. E. Seamons, Y. Chen, P. Jones, J. Jozwiak, and M. Winslett. Server-directed collective I/O in Panda. In Proc. Supercomputing '95, December 1995

41. P. Corbett, D. Feitelson, S. Fineberg, Y. Hsu, W. Nitzberg, J.-P. Prost, M. Snir, B. Traversat, and P. Wong. Overview of the MPI-IO parallel I/O interface. In IPPS '95 Workshop on Input/Output in Parallel and Distributed Systems, 1995.

42. A. Choudhary, R. Bordawekar, M. Harry, R. Krishnaiyer, R. Ponnusamy, T. Singh, and R. Thakur. PASSION: Parallel and scalable software for input-output. Technical Report SCCS-636, NPAC, Syracuse, NY, 1994.

43. N. Carriero and D. Gelernter. How To Write Parallel Programs. A First Course. The MIT Press, Cam bridge, Mass., 1990.

44. C. Koelbel, D. Loveman, R. Schreiber, G. S. Jr., and M. E. Zosel. The High Performance Fortran Handbook. The MIT Press, Cambridge, MA, 1994.

45. S.Plimpton, "Fast parallel algorithms for short range molecular dynamics", Technical report SAND91-1144, Sandia National Laboratories, Albuquerque, New Mexico 87185, May 1993.

46. I. Foster, R. Olson, and S. Tuecke. Productive parallel programming: The PCN approach. Scientific Programming, 1(1):51-66, 1992.

47. P. Hatcher and M. Quinn. Data-Parallel Programming on MIMD Computers. MIT Press, 1991.

48. R. van de Geijn and J. Watts. Summa: Scalable universal matrix multiplication algorithm. Technical Report TR-95-13, Dept of Computer Sciences, The University of Texas, 1995.

49. R. van de Geijn, "Using PLAPACK:Parallel Linear Algebra Package," Scientific and Engineering Computation Series, The MIT Press, Cambridge, MA, 1997

50. D. Gannon, Sage++, http://www.extreme.indiana.edu/sage/

51. J. Choi, J. Dongarra, R. Pozo, and D. Walker. ScaLAPACK: A scalable linear algebra for distributed memory concur rent computers. In Proceedings of the 4th Symposium on the Frontiers of Massively Parallel Computations, pages 120-127. IEEE Computer Society, 1992.

6 Budget Explanation

The budget for this project predominantly supports scientist labor, including staff at PNNL (0.7 FTE), and faculty (1FTE), system support personnel (0.167FTE) and (graduate) students at NPAC. It also covers the appropriate fraction of software and computer maintenance/upgrades for desktop systems used by the scientists on the project, and anticipated software dissemination costs for the DOE2000 ACTS community. Travel (approx. three trips/year) is required to coordinate work between PNNL and NPAC, to attend ACTS PI meetings, and to present the work at scientific conferences. Miscellaneous costs include office and computer supplies and publication page charges. Itemized annual requests and overhead rates are on the attached budget sheets.

It is requested that each institution, PNNL and NPAC, be directly funded.

J. Nieplocha [60%]: Principal Investigator

R.J. Harrison [10%]: Application input to design, prototyping, and testing.

G. Fox [5%]: Co-Principal Investigator

T. Haupt [30%]: Padre/PAWS integration, HPF extrinsic GA library.

B. Carpenter [33%]: HPF library and language extensions.

D. Bernholdt [35%]: HPF extrinsic GA library, prototyping.

7 Other Support of Investigators

7.1 PNNL

DoE (OCTR/MICS), High Performance Computing and Communications Grand Challenge project, "Computational Chemistry for Nuclear Waste Characterization and Processing: Relativistic Quantum Chemistry of Actinides ", awarded March 1997 with expected three year duration, $450K/annum. Currently Harrison 60%, Nieplocha 50%. If this proposal is successful support for Nieplocha will be reduced to 20%.

DoE Environmental Molecular Sciences Laboratory operations core funding in support of high performance computing and computational chemistry activities at the EMSL, commenced July 1997. Currently Harrison 40%, Nieplocha [50%]. If this proposal is successful these will be reduced to Harrison 30%, Nieplocha 20%.

7.2 NPAC

Tom Haupt is involved in two projects:

1. ARPA: Common Runtime Support for High Performance Parallel Languages, 8/15/94 - 6/30/98, 50% FTE.

2. NSF Grand Challenge: "Black Hole Binaries: Coalescence and Gravitational Radiation", 10/2/93-8/31/98, 50% FTE.

David Bernhold in involved a project "Molecular Science Software/Advanced Computer Architecture", funded by Pacific Northwest National Laboratory,12/1/96--7/31/97 at 4.5 month effort. Contract expected to be extended with additional funding. In addition, he is supported by "Nason Research Fellowship", Alex. G. Nason Foundation, Inc., maximum annual salary of $57,500 comprised of outside support supplemented by fellowship, 9/6/95--8/31/98.

8 Biographical Sketches

8.1 Jarek Nieplocha, Pacific Northwest National Laboratory

Jarek Nieplocha received his MS in Electrical Engineering from Warsaw University of Technology in 1985 and Ph.D. from The University of Alabama in 1993, where he specialized in parallel processing. He has been a senior research scientist at Pacific Northwest National Laboratory since 1993. His recent research has been in interprocessor communications, high-performance I/O and metacomputing. He is affiliated with IEEE Computer Society and ACM. His last 12 papers are:

J. Nieplocha and R.J. Harrison, "Shared memory programming in metacomputing environments: The Global Array approach", The Journal of Supercomputing,1997 (in press).

J. Nieplocha, R.J. Harrison, and I.T. Foster, "Explicit Management of Memory Hierarchy", in Advances in High Performance Computing, Eds. J. Kowalik, L. Grandinetti and M. Vajtersic, Kluwer Academic, 1997.

Y. Chen, I.T. Foster, J. Nieplocha, and M. Winslett, "Optimizing Collective I/O Performance on Parallel Computers: A Multi-System Study", Proc. 11th ACM Int. Conf. Supercomputing, 1997.

J. Nieplocha and RJ Harrison, "Shared memory NUMA programming on I-WAY", Proc. IEEE High Performance Distributed Computing HPDC-5. 1996.
(*HPDC-5 Best Paper Award*)

J. Nieplocha and I.T. Foster, "Disk Resident Arrays: An array oriented I/O library for out-of-core computations", Proc. IEEE Conf. Frontiers of Massively Parallel Computation, 1996.

J. Nieplocha, RJ Harrison, and RJ Littlefield. "Global Arrays: A Nonuniform-Memory-Access Programming Model for High-Performance Computers", The Journal of Supercomputing, 1996.

H. Dachsel, H. Lischka, R. L. Shepard, J. Nieplocha, and R.J. Harrison, "A massively parallel multireference configuration interaction program -the parallel COLUMBUS program," J. Comp. Chem., 1997.

J. Nieplocha, RJ Harrison, and RJ Littlefield. "The Global Array Programming Model For High Performance Scientific Computing", SIAM News, August 95.

J. Nieplocha, W.C. Schreiber, "The Use of Conjugated Gradient Methods with a Segregated Finite Volume Procedure for Solving Transient Incompressible Navier-Stokes Equations", Communications in Numerical Methods in Engineering, vol 11, pp. 849-56, 1995.

J. Nieplocha, R.J. Littlefield, and M. Rosing, "Beyond Message Passing: A case for one-sided communication in MPI", Proceedings of MPI Developers Conference, http://www.cse.nd.edu/mpidc95/proceedings/papers/html/nieplocha,June 21-23, 1995.

R.J. Harrison, M.F. Guest, Rick A. Kendall, D.E. Bernholdt, A.T. Wong, M. Stave, J. Anchell, A.C. Hess, R. Littlefield, G.I. Fann, J. Nieplocha, G.S. Thomas, D. Elwood, J. Tilson, R. Shepard, A.F. Wagner, I.T. Foster, E. Lusk and R. Stevens, "High Performance Computational Chemistry:(II) A Scalable SCF Program," J. Comp. Chem, 17, 124, 1995.

J. Nieplocha, R.J. Harrison, R.J. Littlefield, "Global Arrays: a portable shared-memory programming model for distributed-memory computers", Proc. Supercomputing'94.

8.2 Robert J. Harrison, Pacific Northwest National Laboratory

Education:

- B.A., First Class Honors, Natural Sciences Tripos, Churchill College, University of Cambridge, England, 1981.
- Ph.D., High Accuracy Correlated Wavefunctions, Department of Organicand Theoretical Chemistry, University of Cambridge, England, 1984.

Professional Background:

- Staff scientist, EMSL, Pacific Northwest National Laboratory (1992-present)
- Assistant scientist, Theoretical Chemistry Group, Argonne National Laboratory (1988-92)
- Research Associate, SERC Daresbury Laboratory (1986-8)
- Research Associate, Quantum Theory Project, University of Florida(1984-6)

Selected Recent Publications:

J. Nieplocha and R.J. Harrison, "Shared memory programming in metacomputing environments: The Global Array approach", The Journal of Supercomputing,1997 (in press).

J. Nieplocha, R.J. Harrison, and I.T. Foster, "Explicit Management of Memory Hierarchy", in Advances in High Performance Computing, Eds. J. Kowalik, L. Grandinetti and M. Vajtersic, Kluwer Academic, 1997.

H. Dachsel, H. Lischka, R. L. Shepard, J. Nieplocha, and R.J. Harrison, "A massively parallel multireference configuration interaction program -the parallel COLUMBUS program," J. Comp. Chem., 1997.

J. Nieplocha, R. J. Harrison, R. J. Littlefield. The Global Array: A Non-Uniform-Memory-Access Programming Model For High-Performance Computers, The Journal of Supercomputing, 10:197-220, 1996

J. Nieplocha, RJ Harrison, and RJ Littlefield. "The Global Array Programming Model For High Performance Scientific Computing", SIAM News, August 95.

A.T. Wong, R.J. Harrison and A.P. Rendell, Parallel Direct Four Index Transformation, Theor. Chim. Acta, accepted for publication, 1996.

.E. Bernholdt and R.J. Harrison, Large-Scale Correlated Electronic Structure Calculations: The RI-MP2 Method on Parallel Computers, DChem. Phys. Lett, 250, 1996, 477.

D.E. Bernholdt and R.J. Harrison, Orbital Invariant Second-Order Many-Body Perturbation Theory on Parallel Computers: An Approach for Large Molecules, J. Chem. Phys., 102, 1995, 9582-9.

A.T. Wong and R.J. Harrison, Approaches to Large Scale SCF, J. Comp. Chem, 16, 1995, 1291.

I.T. Foster, J.L. Tilson, A.F. Wagner, R. Shepard, R.J. Harrison, R.A. Kendall, and Rik J. Littlefield, Toward High-Performance Computational Chemistry: (I) Scalable Fock Matrix Construction Algorithms, J. Comp. Chem, 17, 1995, 109.

R.J. Harrison, M.F. Guest, Rick A. Kendall, D.E. Bernholdt, A.T. Wong, M. Stave, J. Anchell, A.C. Hess, R. Littlefield, G.I. Fann, J. Nieplocha, G.S. Thomas, D. Elwood, J. Tilson, R. Shepard,A.F. Wagner, I.T. Foster, E. Lusk and R. Stevens, High Performance Computational Chemistry:(II) A Scalable SCF Program, J. Comp. Chem, 17,1995, 124.

R.A. Kendall, R.J. Harrison, R.J. Littlefield, and M.F. Guest, High Performance Computational Chemistry: Methods and Machines, chapter 4, pp. 209-316, in Reviews in Computational Chemistry, Volume VI, Eds. K.B. Lipkowitz and D.B. Boyd, VCH Publishers, Inc., New York(1995).

M.F. Guest, E. Apra, D.E. Bernholdt, H.A. Fruechtl, R.J. Harrison, R.A. Kendall, R.A. Kutteh, X. Long, J.B. Nicholas, J.A. Nichols, H.L. Taylor, A.T. Wong, G.I. Fann, R.J. Littlefield and J. Nieplocha, High Performance Computational Chemistry; NWChem and Fully Distributed Parallel Algorithms, in High Performance Computing: Issues, Methods, and Applications, Eds. J. Dongarra, L. Gradinetti, G. Joubert, and J. Kowalik, (1995).

J. Nieplocha, R.J. Harrison, R.J. Littlefield, "Global Arrays: a portable shared-memory programming model for distributed-memory computers", Proc. Supercomputing'94.

8.3 Geoffrey Charles Fox, Northeast Parallel Architectures Center

Fox is an internationally recognized expert in the use of parallel architectures and the development of concurrent algorithms. He leads a major project to develop prototype high performance Fortran(Fortran90D) compilers. He is also a leading proponent for the development of computational science as an academic discipline and a scientific method. His research on parallel computing has focused on development and use of this technology to solve large scale computational problems. Fox directs InfoMall, which is focused on accelerating the introduction of high speed communications and parallel computing into New York State industry and developing the corresponding software and systems industry. Much of this activity is centered on NYNET with ISDN and ATM connectivity throughout the state including schools where Fox is leading developments of new K-12 applications that exploit modern technology with the Living SchoolBook project.

Collaboraters: Fox has participated in HPCC since 1981 and in that time has collaborated in various fashions with many scientists. In particular current work involves scientists in CRPC (Argonne, Caltech, Los Alamos, Rice, Tennessee), and Grand Challenge scientists at Texas and NASA Goddard

Postdocs and Students: 49 Postdocs, Staff Scientists Faculty and Students who have or are working in NPAC are listed on http://www.npac.syr.edu/homepages/index.html Similarily 50 participants in the Caltech Concurrent Computation Project directed by Fox 1983-1990 are listed with biographic information in Appendix B of the book Parallel Computing Works.

Education:

B.A. in Mathematics from Cambridge Univ., Cambridge, England (1961-1964)

Ph.D. in Theoretical Physics from Cambridge University (1964-1967)

M.A. from Cambridge University (1968)

Professional Experience:

1990-	Professor of Computer Science, Syracuse University
1990-	Professor of Physics, Syracuse University
1990-	Director of Northeast Parallel Architectures Center
1979-1990	Professor of Physics, California Inst. of Tech.
1986-1988	Associate Provost for Computing, California Inst. Of Tech.
1983-1985	Dean for Educational Computing, California Inst. Of Tech.
1981-1983	Executive Officer of Physics, California Inst. of Tech.
1974-1979	Associate Professor of Physics, California Inst. Of Tech.

1971-1974	Assistant Professor of Physics, California Inst. Of Tech.
1970-1971	Millikan Research Fellow in Theoretical Physics, Caltech
1970	Visiting Scientist (April-May), Brookhaven National Laboratory
1969-1970	Research Fellow at Peterhouse College, Cavendish Lab.,Cambridge
1968-1969	Research Scientist, Lawrence Berkeley Lab., Berkeley,Calif.
1967-1968	Member of School of Natural Science, Inst. for Advanced Study, Princeton, New Jersey

Awards and Honors:

Senior Wrangler, Part III Mathematics, Cambridge (1964)

Alfred P. Sloan Foundation Fellowship (1973-75)

Fellow of the American Physical Society (1990)

Journal Editorships:

Principal: Concurrency: Practice and Experience (John Wiley, Inc.),

Physics and Computers, International Journal of Modern Physics

(World Scientific)

Associate: Journal of Supercomputing,

Selected List of Publications:

1. Fox G. and Furmanski W., "Towards Web/Java-Based High Performance Distributed Computing -- an Evolving Virtual Machine", IEEE Conference on High Performance Distributed Computing, pp 308-317, 1996

2. Fox G., "Software and Hardware Requirements for Some Applications of Pa rallel Computing to Industrial Problems", Proceedings of Icase Workshop on Parallel Computing (to be published 1996).

3. Fox G. and Furmanski W., "The Use of the National Information Infrastr ucture and High Performance Computers in Industry" , Proceedings of IEEE Conference on Massively Parallel Optical Interconnects (MPPOI'95), pp 298-312,

4. Fox G. and Mills K., "Information Processing and Opportunities for HPCN Use in Industry", pages 1-14. Number 796 in Lecture Notes in Computer Science, Springer-Verlag, New York, April 1994. Proceedings of "High Performance Computing and Networking(HPCN) Conference, Mun ich, 1994.

5. Fox G.,Chrisochoides N.P.,Haupt T.,"Computational Toolkit for Colliding Black Holes and CFD", Invited Talk at 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO, June, 1994, AIAA-94-2249.

6.Fox, G. C., Messina, P., Williams, R., "Parallel Computing Works!", Morgan Kaufmann, San Mateo, CA, 1994.

7.Fox G.,Bozkus Z.,Choudhary A.,Haupt T.,Ranka S., "Compiling HPF for distributed memory MIMD computers", in Impact of Compilation Technology on Computer Architecture pp 191-217. Kluwer, 1993.

8.Fox, G. C. "Parallel Computing and Education," Daedalus, Journal of the American Academy of Arts and Sciences, Vol. 121, No. 1, pp 111-118, Winter 1992. C3P-958, CRPC-TR91 123.

9. Fox, G. C. "Approaches to Physical Optimization," in Proceedings of 5th SIAM Conference on Parallel Processes for Scientific Computation, pp 153-162, March 25-27, 1991, Houston, TX,J. Dongarra, K. Kennedy, P. Messina, D. Sorensen, R. Voigt, editors,SIAM, 1992. C3P-959, CRPC-TR91124

10.Fox, G.C., Johnson, M.A., Lyzenga, G.A., Otto, S.W., Salmon, J.K., Walker, D.W., "Solving Problems on Concurrent Processors", Vol. 1,Prentice-Hall, Inc. 1988, Vol. 2, 1990.

8.4 Tom Haupt, Northeast Parallel Architectures Center

Tom Haupt's research is concentrated on developing parallel programming methodologies, with a recent shift towards support for distributed computing based on the emerging Web technologies. For several years he served as the project leader of the Fortran 90D group, the ARPA sponsored project to develop a new language for parallel and distributed computer systems to develop HPF/Fortran 90D compiler, the compiler run-time support (within PCRC, and a benchmark suite (including the HPFA project. He participated in the High Performance Fortran Forum, MPI committe, as well as the International Parallel Benchmark Working Group, PARKBENCH. Currently Tom Haupt's research activities concentrate on transferring the new technologies to applications developers. Most of his time is devoted to building a support for distributed adaptive multigrid methods within Binary Black Hole Grand Challenge (both HPF and C++ with MPI). Other areas include computational chemistry, Particle-in-Cell methods, and numerous application kernels. The latter are used by him for evaluation of commercial HPF compilers. He is involved in research to developmethods to support distributed computations in a heterogeneous environments using AVS as a software integration tool, as well as Web based computations.

Haupt graduated from Jagiellonian University, and got his Ph.D. in Physics from Institute of Nuclear Physics in Krakow, Poland, in 1985. From 1985 to 1990 he did research in experimental high energy physics and computation physics at INP in Krakow, and NIKHEF-H, Nijmegen, the Netherlands. From 1990 he has been at Syracuse University, where experience working with Geoffrey Fox resulted in changing his profile from an experimental and computational physicist to a computer scientist.

The most relevant publications:

1. Erol Akarsu, Geoffrey Fox, Tomasz Haupt, Particle-in-Cell Simulation codes in High Performance Fortran, in Proceedings of Supercomputing'96

2. Tomasz Haupt, Shravanthi Reddy, Giriraj Vengurlekar, Low Level HPF Compiler Benchmark Suite, in Fortran Journal, November 1996.

3. R. M. Hockney and M. W. Berry, editors, Public International Benchmarks for Parallel Computers, in Scientific Programming 3(2), 1994.

4. Manish Parashar, Salim Hariri, Tomasz Haupt, Geoffrey Fox, "Interpreting the Performance of HPF/Fortran 90D", in proceedings of the International Workshop on Parallel Processing, December, 1994.

5. Nikos Chrisochoides, Tomasz Haupt, Geoffrey Fox, Computational Toolkit for Colliding Black Holes and CFD, in proceedings of 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO, June 1994, AIAA-94-2249.

6. Edward Bogucz, Geoffrey Fox, Tom Haupt, Kenneth Hawick, Sanjay Ranka Preliminary Evaluation of High-Performance Fortran as a Language for Computational Fluid Dynamics, in proceedings of 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO, June 1994, AIAA-94-2249.

7. Z Bozkus, A Choudhary, G Fox, T Haupt, S Ranka, Compiling HPF for Distributed Memory MIMD Computers in "The Interaction of Compilation Technology and Computer Architecture", Kluwer Academic Publishers;

8. M. Parashar, S. Hariri, T. Haupt and G. C. Fox, A Study of Software Development for High Performance Computing," in "Programming Environments for Massively Parallel Distributed Systems," Karsten M. Decker and Rene M. Rehmann, editors, Birkhauser Verlag, Basel, Switzerland, 1994. (also in Proceedings of the IFIP WG10.3 Working Conference on Programming Environments for Massively Parallel Distributed Systems, Switzerland, April, 1994.)

9. Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, S. Ranka, "Fortran 90D/HPF Compiler for Distributed Memory MIMD Computers: Design, Implementation, and Performance Results", in proceedings of Supercomputing '93

10. Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, S. Ranka "A Compilation Approach for Fortran 90D/HPF Compilers on Distributed Memory MIMD Computers", Sixth Anuall Workshop on Languages and Compilers for Parallel Computing, Portland, OR, August 12-14, 1993

11. M. Parashar, S.Hariri, T. Haupt and G. C. Fox, An Interpretive Framework for Application Performance Prediction," in the Proceedings of the1993 International Conference On Parallel and Distributed Systems ICPADS'93, December, 1993

12. A Choudhary, G Fox, T Haupt, S Ranka, Which Applications Can Use High Performance Fortran and FortranD - Industry Standard Data Parallel Languages? in the proceedings of Fifth Australian Supercomputing Conference, Melbourne, Australia, December 1992

8.5 David E. Bernhold, Northeast Parallel Architectures Center

Dr. Bernholdt is interested in all aspects of computational chemistry, including method development, efficient implementations, and interesting chemical applications. His present focus is on the efficient use of massively parallel processors (MPPs) in electronic structure calculations and new methods suited to the large-scale simulations enabled by MPPs.

Education:

> Ph.D. in Chemistry with minors in Physics and Mathematics and Certification in Chemical Physics, University of Florida, Gainesville, FL, 1993.
>
> B.S. in Chemistry Cum Laude with Highest Distinction, University of Illinois, Urbana, IL, 1986.

Experience:

1995-- Alex G. Nason Fellow, Northeast Parallel Architectures Center, Syracuse University

1995-- Affiliate Staff Scientist, Pacific Northwest National Laboratory

1993--1995 AWU Postdoctoral Fellow, Pacific Northwest National Laboratory

Latest publications:

David E. Bernholdt and Geoffrey C. Fox, Internet Resource Discovery for Chemistry -- Where are Those Vast Untapped Resources?, TrendsAnal. Chem. 16, 230 (1997).

M. F. Guest, E. Apra, D. E. Bernholdt, H. A. Fruechtl, R. J. Harrison, R. A. Kendall, R. A. Kutteh, X. Long, J. B. Nicholas, J. A. Nichols, H. L. Taylor, A. T. Wong, G. I. Fann, R. J. Littlefield, and J. Nieplocha, Advances in Parallel Distributed Data Software; Computational Chemistry and NWChem, 1041 (1996).

David E. Bernholdt and Robert J. Harrison, Large-Scale Correlated Electronic Structure Calculations: The RI-MP2 Method on Parallel Computers, Chem. Phys. Lett. 250, 477 (1996).

David Feller, Adoardo Apra, Jeff A. Nichols, and David E. Bernholdt, The Structure and Binding Energy of K+-Ether Complexes: A Comparison of MP2, RI-MP2 and Density Functional Methods, J. Chem. Phys. 105, 1940 (1996).

M. F. Guest, E. Apra, D. E. Bernholdt, H. A. Fruchtl, R. J. Harrison, R. A. Kendall, R. A. Kutteh, X. Long, J. B. Nicholas, J. A. Nichols, H. L. Taylor, A. T. Wong, G. I. Fann, R. J. Littlefield, and J. Nieplocha, High-performance Computing in Chemistry; NWChem, Future Generation Computer Systems 12, 273 (1996).

David E. Bernholdt and Robert J. Harrison, Orbital Invariant Second-Order Many-Body Perturbation Theory on Parallel Computers: An Approach for Large Molecules, J. Chem. Phys. 102, 9582 (1995).

M. F. Guest, E. Apra, D. E. Bernholdt, H. A. Fruechtl, R. J. Harrison, R. A. Kendall, R. A. Kutteh, X. Long, J. B. Nicholas, J. A. Nichols, H. L. Taylor, A. T. Wong, G. I. Fann, R. J. Littlefield, and J. Nieplocha, High Performance Computational Chemistry:

NWChem and Fully Distributed Parallel Algorithms, in High Performance Computing: Technology, Methods, and Applications, volume 10 of Advances in Parallel Computing, pages 395--427, Elsevier, Amsterdam, 1995.

D. E. Bernholdt, E. Apra, H. A. Fruechtl, M. F. Guest, R. J. Harrison, R. A. Kendall, R. A. Kutteh, X. Long, J. B. Nicholas, J. A. Nichols, H. L. Taylor, A. T. Wong, G. I. Fann, R. J. Littlefield, and J. Nieplocha, Parallel Computational Chemistry Made Easier: The Development of NWChem, Int. J. Quantum Chemistry: Quantum Chem. Symposium 29, 475 (1995).

Rodney J. Bartlett, Magnus Rittby, John D. Watts, and David E. Bernholdt, Carbon Clusters: The Synergism Between Theory and Experiment, in P. J. Reynolds, editor, On Clusters and Clustering, From Atoms to Fractals, pages 23--31, Elsevier, 1993.

S. Ajith Perera, David E. Bernholdt, and Rodney J. Bartlett, Localized Hartree Product Orbitals in Correlated Studies of Molecules, Int. J. Quantum Chem. 49, 559 (1994).

Robert J. Harrison, Martyn F. Guest, Rick A. Kendall, David E. Bernholdt, Adrian T. Wong, Mark Stave, James Anchell, Anthony Hess, Rik Littlefield, George I. Fann, Jarek Nieplocha, Greg S. Thomas, David Elwood, Jeff Tilson, Ron L. Shepard, Albert F. Wagner, Ian T. Foster, Ewing Lusk, and Rick Stevens, High Performance Computational Chemistry. II. A Scalable SCF Program,J. Computat. Chem. 17, 124 (1995).

M. F. Guest, E. Apra, D. E. Bernholdt, H. A. Fruechtl, R. J. Harrison, R. A. Kendall, R. A. Kutteh, J. B. Nicholas, J. A. Nichols, M. S. Stave, A. T. Wong, R. J. Littlefield, and J. Nieplocha, High Performance Computational Chemistry: Towards Fully Distributed Parallel Algorithms, in A. M. Tentner, editor, High Performance Computing 1994: Grand Challenges in Computer Simulation, pages 511--521, San Diego, 1994, Society for Computer Simulation.

9 Description of Facilities and Resources

This work will be performed using computing facilities available at PNNL, NPAC and other DoE supercomputing sites. Both PNNL and NPAC already have sufficient office space, workstations, network access, system administration support, etc., to commence this project. During the lifetime of the project necessary computer equipment will be upgraded as necessary, as described in the Section "Explanation of Budget."

The computing resources at PNNL include 512-node IBM SP with Power-2 Super nodes, SMP multiprocessor SGI systems (Origin-2000 and PowerChallenge) and numerous workstations. A smaller, next generation IBM SP system with SMP nodes will be available at PNNL in 1998. The full suite of parallel software, including HPF, supported by IBM is provided. In addition, through other arrangements, access to Cray T3E is available at NERSC. EMSL support for GA development and maintenance includes modest access to the 512 process IBM SP at EMSL. If this project is funded, PNNL staff will apply for a more significant allocation of resources in direct support of this project (both PNNL and NPAC researchers) which will be provided at no cost to this project.

A complete description of NPAC facilities is available on-line from http://www.npac.syr.edu/NPACoverview/facilities/infra2.html. For this work NPAC will use SMP 8-processor SGI Challenge L system, 12 node IBM SP-2 system, and two high performace workstations clusters: 8 SUN ultra SPARCS workstations connected via ATM switch (OC3c) and 8 DEC Alpha workstations supported by a high performance networking backbone consisting of the dedicated, switched FDDI segments. Software available on these systems includes HPF compiles (PGI on SGI SP-2, and SUN cluster, IBM on SP-2 and DEC on Alpha cluster). In addition, software developed at NPAC,including PCRC infrastructure,HPF compiler, HPF interpreter, and HPF run-time system will be available.