Face Page

1. TITLE OF PROPOSED RESEARCH:

   Performance Portability for Coupled Atmosphere-Ocean General Circulation Models

2. ER PROGRAM ANNOUNCEMENT TITLE:

   DOE LAB 98-05 Climate Change Prediction Program

3. NAME AND ADDRESS OF LABORATORY:

   Oak Ridge National Laboratory
   P.O. Box 2008
   Oak Ridge, Tennessee 37831

4. PRINCIPAL INVESTIGATOR TITLE, ADDRESS AND PHONE NUMBER

   Patrick H. Worley

Title	:	Research Staff, Computer Science and Mathematics Division
Address	:	Oak Ridge National Laboratory P.O. Box 2008, Bldg. 6012 Oak Ridge, Tennessee 37831-6367
Phone	:	423-574-3128
Fax	:	423-574-0680
E-mail	:	worleyph@ornl.gov

Contents

Abstract

This proposal addresses the difficulties in attaining high performance in climate models when ported to new computer systems or new configurations of existing systems. The short-term goal is to evaluate the current and emerging scientific supercomputers in the 100 GFlop to 1 TFlop range and identify how they can be used most efficiently for production scale computing of coupled atmosphere-ocean general circulation models (GCMs). Primary performance goals are achieving good scalability on large numbers of processors and good per-node performance on the individual processors; but, of critical importance, the performance gains must be portable. Massive restructuring of a code for efficiency on one platform that impairs efficiency on others is ultimately counter-productive. Thus, the long-term goal is to develop and demonstrate performance portability techniques that allow what appears to be a single source code to the computational scientist to be easily ported and optimized on different platforms.

Previous research established the feasibility of such techniques for components of the current coupled models. New research will extend the techniques to additional components, integrate the different approaches, and examine new issues that arise when using hybrid shared memory/distributed memory computer systems. Performance portability is crucial to the development of state-of-the-science GCMs. The development time of these models has already exceeded the lifespan of the previous generation of supercomputers, and the promise of climate simulation will only be achieved if the models can take advantage of the computational power of succeeding generations of supercomputers.

1 Narrative

1.1 Background and Significance

Statement of Problem. This proposal addresses computer science and computer engineering research issues in area number one of Program Announcement LAB 98-05:

> Modifications and improvement required to port state-of-the-science coupled atmosphere-ocean general circulation models (GCMs) or model components to advanced, next generation massively-parallel scientific supercomputers.

One of the current assumptions of the Department of Energy (DOE) Climate Change Prediction Program is that climate-change studies and assessments will use sophisticated atmospheric and ocean GCMs at moderate to high spatial resolutions and with increasingly realistic physics. The computational requirements of using such models for studies of decade-to-century climate change and climate variability requires use of the largest available scientific supercomputers. For example, in a recent experiment using the National Center for Atmospheric Research (NCAR) Climate System Model (CSM), almost 8 days of dedicated time on a 32 processor NEC SX-4 (which achieved a sustained computation rate of 15 GFlop/sec on this model) was required for a single 125 year simulation using a moderate resolution atmosphere and a low resolution ocean [16]. Computational requirements will continue to increase as more physics is added and spatial resolution is increased.

The need to use the most powerful supercomputers is traditional in the meteorological community, and GCMs have always been coded so as to be efficient on these machines. But, in the past, this requirement was simplified by the similarity in architectural design of (vector) supercomputers. A single code ran efficiently on most of the available platforms. Current and next-generation scientific supercomputers are much more diverse, using

- Vector, RISC, or CISC processors,
- Distributed or shared memory subsystems,
- A large variety of interconnect technologies and topologies, and
- Data-parallel, message-passing, or shared-memory programming models.

Recent research has shown how much performance can be lost if a GCM is not tuned for a particular architecture [5, 22]. This is not always stated explicitly in the literature, but can be seen from execution timings that represent only a small percentage of peak performance. While peak performance is rightfully criticized as being unobtainable even with optimized code, timings from models or kernels written specifically for a given architecture often show significantly better performance, as demonstrated by the "before" and "after" timings found in [1, 9, 10, 38]. This and similar research [4, 40, 41] has also shown how unpredictable this optimization can be. Even logically similar architectures require different techniques to achieve good performance.

Despite the dissimilarity of techniques required to attain good performance on high performance platforms, a (simple) single source view of a GCM is as important as ever. The science and mathematics incorporated in the GCMs improves over time in an evolutionary and modular fashion. Careful, studied change is a requirement in climate research given the difficulty in validating the models. The manpower and expertise are not available for frequent rewrites or model restructuring. Moreover, since the GCMs do evolve, the computational scientists must find the code easy to modify.

The problem we propose to address arises because the porting exercise never ends. These large codes are developed on one (or more) platforms, targeted for others, and eventually run on whatever platforms are available, for many reasons:

1) The lifespan of the code structure and many of the components of the coupled model will be much longer than the lifespan of the typical high performance computer system. The development time of the current models has already proved to be longer than the lifespan of the previous generation of supercomputers.

Table 1:

		Issues	Code Analysis and Transformation	Tool Support	Metric
	Limits to parallelism	Parallelism over multiple dimensions of domain	Parallelize additional dimensions; Numerous manual and some automated approaches, depending on memory model and access to tools	1, 2, 3	Scaling
	Load Imbalance	Ability to relocate state data to provide even distr. of work between processors.	Tiling, multi-level decomposition; restructuring of data to allow movement of state data.	1, 2, 3, 4	Tmean/Tmax
Coherency	DM	Message passing; latency, bandwidth, interconnection topology	Analyze code for data dependencies; data structures for communication (e.g. halos, buffers). Modify loops, domain indices for multiple local address space; insert communication.	1,2,3,6	Model run times, scaling.
	SM	Multi-tasking: identify parallel loops, private/shared data. Barriers, atomic sections.	Analyze code to identify parallel loops, identify private/shared data and encode as directives. Implement barriers, etc. Push parallel loops up the call to do maximum work on each iteration.	1,7	
	DM-SM	Issues of both DM and SM.	1. Multi-level decomposition, two-dimensional. 2. Mixed dimension approach (one dimension SM, other DM).	1,2,3,6	
	DSM	…plus attention to data locality to minimize cost of non-local access and false sharing.		1,2,3,6	
Cache misses	Capacity	Accesses to same set of arrays over multiple loops can overflow cache.	Loop fusion, blocking. Fusion may also depend on converting array syntax to explicit loops.	1,6,7	Measurement of number of cache misses relative to memory accesses and processor operations.
	Conflict	References to misaligned single arrays or multiple different arrays bump each other out of cache.	Padding and alignment so that subsequent references avoid hitting same cache line. Merging data (structures) to improve locality.	1,6,7	
	Compulsory	First access always incurs a hit	Array index reordering/loop nest reordering may provide stride allows taking advantage of multi-word cache reads, avoiding hits on next *b* iterations, where b is line size.	1,6,7	
	FP/Memory Balance	Register reuse may be improved for loops with recurrences.	Loop-unrolling and jamming, with scalar replacement.	7	Number of references relative to flops
	Vectorization	Vector efficiency depends on keeping pipelines full	Long vectors over dimensions without certain data dependencies motivate choice of longest domain dimension for inner loops; also motivates 1-d expression of multi-dimensional domains.	1,6,7	Efficiency relative to peak.

1. Data dependency analysis
2. Run time libraries
3. Code frameworks
4. Run time instrumentation
5. Special libraries
6. Loop restructuring
7. Optimizing compiler

2) The computing resources available for climate research make up a rapidly shifting landscape. In order to achieve the science goals requires that the models be able to run across a large percentage of the high performance platforms available to the DOE research community.

3) The heterogeneous makeup of the design of the coupled models and the computational requirements of the individual components encourage the use of heterogeneous computing, running different components on (potentially radically) different platforms.

4) Decadal and longer simulations will require long runs on even the fastest computers currently envisioned. Thus, runs will be check-pointed and restarted, perhaps on a different machine configuration or even platform.

To obtain the expected performance when porting a GCM requires a degree of *performance portability*, which we define as the ability to easily port a code and achieve a large fraction of the available performance. To be practical, performance portability must also include support for simple static views of the mathematics and science in the code, to allow model evolution.

Previous Work. Coding for a particular architecture affects understandability, maintainability, extensibility, reusability, portability, and efficiency on other dissimilar architectures. Concern for efficiency on a given architecture may manifest itself in how arrays are dimensioned, aligned, and allocated in memory; how loops are nested or otherwise structured (blocked, unrolled, fused), at what level they are positioned in the subroutine call hierarchy, and whether iteration is expressed with loops at all; how data is communicated between subroutines; and, from the point of view of implementing codes on distributed memory architectures, how interprocessor communication is handled (Table 1) [20, 25, 28]. As mentioned previously, the ability to exploit a range of computer architectures with a *single source* code provides obvious benefits in reducing software costs for an institution charged with developing and maintaining a large code. Common approaches are (1) to maintain separate sources, (2) conditional compilation, (3) data-parallel languages, (4) distributed-shared memory, and (5) distributed memory using application-specific parallelization libraries.

1-2) Maintaining separate codes is difficult and time consuming because changes and enhancements must be made by hand and tested over all versions; without large amounts of concerted and disciplined effort, some versions invariably fall behind. Conditional compilation, the use of preprocessors such as the Unix C-preprocessor (CPP) to enable or disable sections of architecture-specific code through compile time switching, produces messy and difficult to read source code. The approach merely obscures the fact that multiple codes are being maintained (just in the same set of files).

3) Parallel languages such as High Performance Fortran (HPF) [26] aim to provide a means for writing efficient, single-source expressions of model software in which architecture specific details are handled transparently by the HPF compiler on the target machine. Unfortunately, existing models may require substantial rewriting (which may be addressed using source translators [17]). HPF affords limited decomposition options, e.g., for load balancing, and may be ill-suited for expressing less structured parts of a model computation, e.g., for nesting. Portable performance also continues to be an area of concern.

4) Distributed shared-memory (DSM) machines deploy nodes with one or more processors in distributed memory configurations but include sophisticated hardware and system software to provide a single address space, shared-memory model to the running application. By allowing shared-memory programming, DSM preserves investment in codes that have been programmed to run shared-memory parallel on multi-processor vector machines (PVP). The new OpenMP [2] standard addresses the concern about non-standard directives for expressing shared memory parallelism. However, the appeal of the shared-memory programming model is based at least in part on the questionable notion that shared-memory programming is easier than distributed memory for SPMD applications. In fact, for codes that have not been parallelized to begin with, both approaches have their own complexities, pitfalls, and needs for restructuring. Moreover, in existing models that have been parallelized for PVPs, multi-tasking is invariably expressed over only one dimension at a high level in the call-tree (leaving the other for vectorization at a low-level); this one-dimensional approach severely limits parallelism and thus scalability, not even considering parallel efficiency. Finally, shared-memory codes are not portable to distributed memory machines, whereas distributed-memory codes port trivially to shared memory, raising concerns for true architecture independence.

5) Programming for distributed memory provides portability to any architecture and allows the program to exploit scalable parallel machines to their fullest. At one time, programming for distributed memory — which, unlike shared memory programming, entails decomposition of data structures over multiple processor memories — *was* more difficult, although partly because it was unfamiliar. Now, however, much of the painful low-level detail originally associated with message passing programming — domain-decomposition, message-passing, distributed I/O, and load-balancing — can be encapsulated in libraries, as many groups have demonstrated [19, 36, 31, 27, 37]. These approaches still require significant modification to the code for handling loops over local data, global and local index translation, and distributed I/O. Additionally, the ability to encapsulate does not guarantee that the encapsulation provides the desired performance portability.

As evidenced by the series of workshops held at the European Center for Medium-Range Weather Forecasting (ECMWF) over the past decade [21, 22, 23, 24] and by the recent workshop on software engineering in meteorological models sponsored in part by DOE, most model developers concerned with performance portability have adopted approach (5), and the current generation of massively parallel production-quality meteorological models are based at least partly on Fortran 77 and message-passing. In particular, the following projects address aspects of the problem of performance portability (implicitly or explicitly), and employ techniques that we consider important to investigate as part of this proposal. The first three represent previous work by this consortium's senior personnel.

a) The development of the massively parallel message-passing version of the Community Climate Model (CCM) by researchers from ANL, NCAR, and ORNL [6, 7]. This code uses an additional "high level" call structure or framework to support multiple parallel algorithms and a low level encapsulation of the message passing to support runtime tuning of the parallel code, allowing easy optimization of the message-passing and load balancing on different platforms [9, 10, 43, 44].

b) The development of the parallel version of the PSU/NCAR MM5 regional atmospheric model [15]. This more than five year research collaboration between ANL and the Mesoscale and Microscale Meteorology Division (MMM) at NCAR explored numerous issues of relevance to this proposal. These include multi-level domain decomposition, irregular domain decomposition for dynamic load balancing, dynamic memory allocation, modular data and code design utilizing Fortran90, and automatic source-to-source translation for hiding distributed memory loop and index-translation for single-source distributed memory parallelization [8, 30, 33, 34].

c) The optimization of the Los Alamos National Laboratory (LANL) Parallel Ocean Program (POP) by John Levesque of Applied Parallel Research Inc. (APR) using source translation tools. Specific transformations included replacing array syntax with explicit loops, fusing the resulting loops, and other translations for improved cache reuse. APR tools were also used in developing the one-dimensional parallelization of the Geophysical Fluid Dynamics Laboratory (GFDL) ocean model MOM.

d) The development of the massively data-parallel version of the Community Climate Model by researchers from NCAR [17]. Much of the data-parallel version was generated from the "base" PVP version of the code via source-to-source translation.

e) The development of the Integrated Forecast System (IFS) at ECMWF [1]. This code was written from scratch, and uses a technique for tuning the code for either vector or cache-based processors. The technique is very effective at retaining vector performance, and has some impact on improving cache utilization on cache-based processors.

f) The development of SC-MICOM, the implementation of the Miami Isopycnal Coordinate Ocean Model (MICOM) targeted for clusters of shared memory machines [38]. By the use of a two level domain decomposition technique, similar to that pioneered in the parallel version of MM5, SC-MICOM exploits both distributed- and shared-memory parallelism.

These projects demonstrate both that progress has been made in the writing of parallel GCMs and that much is still left to be done. All of these efforts are somewhat specific to their target architectures and application codes and address only aspects of the performance portability problem. Further, the effectiveness of the approaches on or the generalization to current and new architectures is not clear and there is little support for dynamic, heterogeneous computing environments.

Statement of Work. We propose to revisit, unify, and generalize this previous work on how to achieve good performance on high performance computer architectures, by using both current and evolving supercomputer platforms and application codes drawn from the climate system models. We will address the performance evaluation task of determining what are the important performance aspects of these architectures, and how to code for efficiency. And we will emphasize tools and strategies for implementation that hide modifications for performance from the scientist, including frameworks, libraries, and source translation.

Unique aspects of this proposal are:

- Concentration on *complete codes*, both for the performance evaluation and for the implementation of the performance portability techniques,

- Performance portability for a wide spectrum of *architectures*, including dynamic and heterogeneous computing environments,
- *Tools* that hide and automate many of the necessary modifications, and
- Focus on both *existing and new* codes.

Impact. The impact of this research on the modus operandi for developing climate models should be immediate. The performance evaluation results will provide needed information on the performance sensitivity of the current supercomputer architectures, in terms relevant to climate modelers. Source translation tools and subroutine libraries will make exploitation of this information feasible, resulting in an increase in productivity of current models. Relevance will be shown by using codes drawn from or representative of components in the DOE coupled climate system models, including codes such as the NCAR CCM3 and the GFDL MOM codes. Initial platforms will include machines such as the SGI/Cray Origin and T3E, the IBM SP, and the Intel Paragon. As the techniques mature, they will then be validated on other candidate architectures such as traditional parallel vector processors and networked "virtual" parallel machines.

Significant long term impact includes the potential for single-source, high performance, portable codes. The result of the research should be an increased computational efficiency and increased throughput of coupled climate simulations of interest to DOE and the broader climate research community.

1.2 Preliminary Studies

We have been actively involved in research on performance measurement, modeling, and evaluation over the past 10 years, having worked with the AP1000, the CM5, the Cogent, the Convex Exemplar, the Cray T3D and T3E, the IBM SP series of systems, the Intel iPSC series, the Intel Paragon series, the nCUBE series, the KSR series, the Meiko series, and the SGI Origin, as well as traditional PVP systems and networks of workstations. This previous work includes experience with the full range of message-passing and shared-memory libraries and performance measurement, evaluation, and tuning tools. With the advent of the DOE Computer Hardware, Advanced Mathematics, and Model Physics (CHAMMP) program, much of this work was motivated by or applied directly to the question of performance evaluation for climate system model components. For example, prototypes of the performance tools PICL [12] and ParaGraph [18] were used early in our work on parallel algorithms for spectral atmospheric models, motivating many of the features of these tools and significantly aiding parallel algorithm research [39, 42].

Our interest in encapsulation techniques for performance portability arose from a previous CHAMMP project, in which we attempted to evaluate the numerous parallel spectral transform methods in a single testbed code called PSTSWM. The important results of this research were the following [9, 10, 43, 44, 46]:

- No single parallel algorithm was best all of the time, depending instead on the parallel platform, the number of processors, and the problem size.
- The details of the parallel implementation, including the choice of the message-passing transport layer and the communication protocol used with this layer, could be more important than the choice of the parallel algorithm.
- Tuning the parallel algorithm and implementation for the problem size and parallel platform could have a significant impact on performance.
- Multiple algorithms and implementations could be supported in a single code, allowing runtime tuning.

In a very short period of time, we were able to move many of these "encapsulations" into our ongoing parallelization of the CCM, called PCCM2, including multiple "high level" parallel algorithms. This allowed significant performance improvements on the Paragon and IBM SP target systems [7], and made the porting of PCCM2 to other platforms significantly simpler. Note that many of the low level options supported in the encapsulation applied to the parallel semi-Lagrangian transport and load-balancing algorithms as well as to the parallel spectral transform algorithms in PCCM2.

Since that time, we have continued to use the testbed code PSTSWM for performance studies, looking at the relative importance of latency and bandwidth in current parallel application codes [45] and the promise and limitations of using performance models for algorithm comparisons [3]. We have also implemented some of the low level encapsulation techniques developed in PSTSWM in POP. Most relevant to this proposal is an ongoing evaluation of the Message Passing Interface (MPI) [35] as a tool for performance portability [41]. MPI and PVM are currently the defacto standards for message passing, and MPI in particular was designed to include those features that had proven to be be important for performance in the previous generation of message-passing libraries. MPI includes a very rich set of commands, both for traditional point-to-point communications and for global operations. With vendor support, MPI is now available on most of the current parallel platforms, but our research indicates that it does not solve the performance portability question, in that the choices of communication layer and protocol are still very important in achieving good performance, and that the global operations provided with the various instantiations of MPI often perform poorly. Our initial study involved the IBM SP, Intel Paragon, Cray T3D and Convex Exemplar. More recent studies utilizing the Cray T3E and the SGI Origin indicate that this situation is not improving, and that encapsulation techniques are useful even as a self-defense technique, allowing the developer to use vendor-supplied, optimized parallel libraries when they are available, but not having to depend on their existence or efficacy. Unless performance requirements become part of the MPI (and other parallel tools) standard, application-specific layers and parallel algorithm implementations will continue to be important for performance-critical applications.

Source analysis and translation will play an important role in performance portability by allowing modifications to be automatically and transparently applied to the code; for example, by hiding calls to encapsulation layers, restructuring code and data for computational efficiency, uncovering data-dependencies for implementing parallelism, and inserting instrumentation for performance analysis. This group of proposers has extensive background and practical experience with source level tool design and implementation. A number of Fortran analysis and translation tools have been developed and utilized in previous CHAMMP-related and other atmospheric modeling work, including tools for data dependency analysis, array index reordering, and ad-hoc tools.

We have used source translation to successfully parallelize a large existing atmospheric model, the PSU/NCAR MM5. Previous parallel implementations [8, 30] demonstrated efficiency and scalability on distributed memory parallel computers and provided, as well, a more modular, dynamically configurable code. Nevertheless, these earlier efforts fell short in that extensive modification for parallelization prevented integration with the official version. Using source translation, we produced a new *same-source* parallel implementation of MM5 that is line-for-line identical with nearly all of the official MM5 [33]. It is efficient and scalable, and the modifications have no effect whatsoever on function, operation, or performance of the model on other, non-DM parallel computers. The same-source approach to parallelization employs a lightweight yet powerful source translator, The Fortran Loop and Index Converter (FLIC) [29], to generate the parallel code from the source model when the user types "make." The approach is essentially directive-less, requiring only a small amount of information – sufficiently general and concise to fit on the tools command line – to direct the translation. Of 35,000 MM5 source lines, only 4,000 differ and this is confined primarily to code for nesting and I/O. Changes in dynamics (non-hydrostatic and time-split solvers) is limited to the insertion of calls to halo-update library routines and the number of lines changed is *less* than the number directives for shared memory parallelism. Physics is virtually untouched and, in effect, *parallel as written.* Identifying data dependencies and implementing communication is still manual, at present, but this is also amenable to automated source analysis and translation as APR and other groups have demonstrated.

The same-source parallel MM5, designated "MM5e," is running operationally at the Air Force Weather Agency, Offutt AFB, Nebraska and in research mode at a number of other sites. During development of MM5e, NCAR generated four updates to the model (from release 2 to the current release 6). Because the development version is so close to the official model, these updates were incorporated into the MM5e code automatically using CVS on each occasion. Further, because of this minimal impact to the code, NCAR has agreed to integrate this parallel option into the official MM5 during the first quarter of 1998, which will eliminate the need for a separate parallel version of the model entirely.

This group's capabilities will be further enhanced by a proposed subcontract with Applied Parallel Research Incorporated of Sacramento, California, providing access to APR tools and supporting collaboration with John Levesque. APR has extensive experience in the field of commercial quality software for automated restructuring of large scientific codes. APR's Forge [11] software provides whole-code (inter-procedural) Fortran analysis and restructuring for automatic shared, distributed, and shared-distributed memory parallelization, cache instrumentation and optimization. The unique capability of the APR tools is the ability to analyze and restructure across subroutines and source files, permitting whole-code optimization – for example, wholesale two-dimensional tiling of loops regardless of their position in the program call tree. Initially, and to a very limited extent, the tools will be utilized in their commercial form; however, the thrust of the collaboration will not be automatic parallelization using APR software (which will remain proprietary). Rather, the primary thrust will be to employ APR's underlying code parsing, analysis, and transformational infrastructure for developing advanced analysis and translation software for performance portability and single-source climate models. This new software will be in the public domain. As noted above, company founder and president John Levesque has direct experience using APR tools on the MOM code under contract with NOAA/GFDL, with the POP code under contract to LANL, and has also collaborated extensively with Argonne on the same-source parallel implementation of MM5.

API and run-time support development for dynamic performance adaptability is also an active area of research and development for this group of proposers. Higher level systems that encapsulated lower-level message passing mechanisms, dynamic domain decomposition over virtual machine resources, distributed I/O, checkpointing, fault-tolerance, adaptive mesh refinement, and computational steering insulate model code from changes in the underlying computational and network hardware [13, 14, 27, 31, 32, 36, 37]. A current example directly applicable to atmospheric codes is the Runtime System Library (RSL) [31, 32]. In addition to providing high-level communication constructs for parallel atmospheric models, RSL provides dynamic decomposition and automatic state data remapping capabilities. The utility of RSL in facilitating scalable efficient performance with dynamic load balancing is described in [30]. Fully processor independent runs and restarts of the code are possible, and the API is also being expanded to support shared/distributed memory execution on distributed-memory clusters of SMPs. Support for domain nesting in RSL may also be applied to model coupling in the context of global climate simulation.

1.3 Research Design and Methods

Adapting models to run efficiently on 100-Gflop to 1-Tflop class computers involves evaluating and addressing obstacles to scaling and per-node performance, and doing so in a way that fulfills the goal of supporting portable, maintainable software. Previous work emphasized performance evaluation and optimization with kernel codes; for example, the parallel spectral shallow water testbed for spectral GCM dynamics. While necessary, such work is not sufficient because whole-code performance is our metric of interest. Therefore, we do not intend to merely consider stand-alone kernels but rather, overall code performance. Likewise, knowing what provides high-performance for each particular platform is necessary, but again, not sufficient since our goal is achieving high-performance across all platforms with a single-source code. Thus, while development of high-performance ports of particular models to particular machines is an expected and welcome side benefit of this work, our purpose is to enable good performance from a single-source code over all available architectures, not porting and hyper-optimization for a single architecture.

Codes. We will target codes representative of climate model components and computer systems that have promise as platforms for production climate simulations. The goal is to produce results that are general enough to be applicable to the full range of component models and platforms.

Coupled ocean-atmosphere systems for climate prediction comprise a mix of computational kernels (Eulerian finite-difference, spectral transform, semi-Lagrangian transport, implicit solvers, column physics, etc.) and attendant, interrelated issues for parallel performance and processor efficiency. We expect to work with a number of representative codes, including CCM, MOM, MICOM, MM5, and POP. Initial studies will employ components from the existing NCAR CSM or Parallel Climate Model (PCM), assuring relevance to the computational and scientific goals of the climate change program. Depending on the performance issue being addressed, the code examined will range from a loop fragment to a subroutine to a full component model, but the metric used will be the performance impact on the full model.

As noted earlier, we have extensive experience with CCM, the atmospheric models used in the NCAR

CSM and PCM, and have worked with the other components, especially the coupler and the messaging layer (MCL, MPI, and PVM). Similarly, we have extensive experience with MM5 and have worked with POP. At a software engineering level, we are familiar with both regular and “less regular” grid models (such as MICOM and MM5, respectively), spectral methods, particle methods (such as the NCAR spectral semi-Lagrangian atmospheric models), and column physics packages.

We have collaborated with the developers of these models, and although such collaboration is not a requirement for this proposal, we hope and expect new opportunities will arise naturally as this research progresses. We are also actively involved in the numerical research to produce algorithms for the next generation of climate models, and are aware of the algorithmic trends and software engineering implications. In later years of the research, we expect to validate performance portability techniques by using them to develop new algorithmic kernels based on this numerical research.

Platforms. Target platforms are differentiated by processor type (primarily vector versus microprocessor/cache) and parallel architecture. Within the latter, shared and distributed memory are the principal classes; however, within these are variants of pure distributed-memory (DM), pure shared memory (SM), distributed clusters of SM nodes (DM/SM), and distributed shared memory (DSM). Network topology, native programming model, and other features may also differentiate machine from machine. Currently, we have access to the Cray T3E at the National Energy Research Scientific Computing Center (NERSC), the SGI Origin 2000 at LANL, the IBM SP system at ANL and the Intel Paragon systems at ORNL. These resources are sufficient to begin the proposed research. We also expect to have access to a C90 and an HP/Convex Exemplar by the time this proposal would begin, and will pursue access to additional platforms as the opportunities arise. These other machines are important for validating that the methodology and tools support performance portability.

As succeeding generations of supercomputers become available to the DOE laboratories, we are confident that we will be able to continue to gain access to these resources, especially any that are sited at ANL or ORNL, or supported by the Office of Biological and Environmental Research CCPP program. Note that our computational requirements are minimal by production standards. We need access to the full range of machine configurations, but not for any significant length of time. In the past, courtesy accounts were readily available for the purposes of performance evaluation, and we have had accounts on machines at other government and university sites, and around the world.

Performance measurement and evaluation. We propose to determine what are the important issues affecting performance of a given code on a particular architecture. The ability to exploit parallelism, to maintain efficient distribution of work, and to hide or otherwise minimize coherency overheads are made easy or hard by the structure of the code and data. Likewise, the ability to achieve good percentages of peak performance requires good cache reuse, computation to memory balance, or vector utilization. The performance issues that will be investigated include those relating communication between distributed memories, multiprocessor execution within shared memories, and effects of code and data structure on per-processor performance.

On distributed memory, message-passing transport layers, protocols, and global communication algorithms have a significant role in maintaining efficiency. We have extensive experience in this type of analysis and have semi-automatic evaluation procedures that can be performed on each new platform [9, 10, 41]. For shared memory, low level issues concern synchronization and other coordination mechanisms such as semaphores, as well as decisions on when and how to update shared data structures. We propose to apply a similar analysis on those platforms where such programming styles are appropriate, initially exploiting our own experience with SHMEM on the T3D/T3E series of machines and the experiences of other researchers on the SGI and HP/CONVEX shared memory machines, in particular [38].

Code and data structural issues include the number and order of array indices, padding, alignment, and buffering; loop vector length and cache blocking, nesting order, and nesting structure of loops, vector length, blocking, and level in program call tree. We intend to utilize vendor supplied cache and performance monitoring software – e.g. Perfex and HPM (SGI/Cray), the Power2 Performance Monitor (IBM), etc. In the process, we will contribute to new cache and data-layout instrumentation tools now under development by APR within the Forge explorer tool framework.

The performance measurement and evaluation phase of the research will produce a mixture of performance measurements and a list of available options for each parallel platform. The performance measurements will be used to generate a decision matrix for evaluating characteristics of the target platform (communication cost parameters, processor speed, cache and memory configuration and access performance) and characteristics of the code and algorithms: data access patterns, dependencies, interaction between major software components of the model (eg. physics and dynamics), etc. Nor will performance monitoring and evaluation be a strictly a priori activity of this research. As will be discussed in the next section, monitoring at run-time, coupled with run-time support in the form of parallel frameworks, provides the ability to delay choice of performance options until runtime.

Performance implementation. The measurement and evaluation phase of this work involves rigorously determining what to do to achieve superior parallel as well as per-processor performance on various platforms. For the implementation phase we propose to develop, demonstrate, and deploy tools and techniques for applying this knowledge in new and existing codes. These will include encapsulation, source-to-source translation, and run-time system support.

Encapsulation techniques are of two sorts: encapsulating low level details in subroutine libraries or macros and "encapsulating" the user code within the context of higher level frameworks. Examples of low level encapsulation include the choice of message-passing transport layer, message passing protocols, and global communication algorithms. (Parallel) frameworks support parallel algorithm choice. While these choices can be application-dependent, the approach is quite general. We have extensive experience with frameworks that support choice in domain decomposition and number of parallel processes, including when and how to remap the domains, and whether to invoke distributed or serial algorithms for different phases of the code. On the new generation of cluster-based parallel platforms, the potential application of frameworks increases significantly. Now the parallel algorithm choices include decisions on when and how to invoke message-passing across a heterogeneous system, message-passing across a homogeneous (sub)system, shared-memory parallelism, and vectorization.

Note that the best choices for the various parallelization options are not only platform- and application-specific, but often also problem size and system configuration specific. A priori evaluation may not always indicate how to optimize performance for any given run. A strength of encapsulation for performance portability is that multiple options can be supported within the same executable. In this case, encapsulation is not simply hiding the difference in implementation between different problem sizes or across the different platforms, but also potentially delaying the decision until runtime. This also makes it possible to change options during a run, if there is information that performance can be improved.

Source translation, in combination with libraries, frameworks, and run-time support, is applicable to both parallelization and per-node performance optimization, particularly for existing codes. Modifying a code for parallelization entails extensive but otherwise mechanical modifications to parallel loops and array index expressions, as was successfully employed for MM5 (above). We will also explore using source translation for exploiting shared distributed memory hybrid architectures, through dimensional differentiation, in which one dimension is DM-parallel and the other is SM-parallel – or vector), through two-level decomposition, where the domain is partitioned to create a logical mesh of tiles that is, itself, then decomposed over the machine, or through parallel frameworks that allow the decision to be delayed until runtime. FLIC, combined with the RSL runtime layer, supports dimensional differentiation by translating loops to their appropriate parallel form independently for each dimension. Likewise, the APR Forge tools are able to parallelize for distributed-memory then, in a second sweep, parallelize the second dimension for shared-memory. For two-level decomposition, existing translation techniques for loop blocking will be applicable to compound loop structures for expressing iteration over and within tiles.

Performance adaptation There are limits to the flexibility that can be incorporated using encapsulation techniques and source translation tools alone. Run-time reconfiguration during model runs may require more radical changes in programming paradigms than those proposed above in order to retain efficiency. We propose to investigate such runtime techniques and the required runtime support.

The simplest practical application of performance adaptation techniques would be to achieve a level of fault tolerance during long running jobs. It is quite common for a processor to fail. On present day multiprocessors this often means halting all the processors and rebooting the entire machine. In the future, components will be hot swappable and operating systems will provide for smooth transitions. But this will not help much for climate modeling if a job must be halted because one processor is temporarily out of service. Operating systems and compilers are not designed to deal with the performance issues of the application during execution.

A coding style for climate models that delays the decision on the number of processors and their distribution until run time could be extended to allow dynamic adaptation. That is, the number of processors could change during the run. One approach under consideration is using modules that adapt to changes in computational resources by using the dynamic memory allocation available in F90 on MIMD nodes and choosing their loop parameters (eg. beglat,endlat, beglon,endlon) and other parallel algorithm parameters depending on where they fit in the global solution and loop order. The code will dynamically reconfigure to achieve best performance given the number and type of processors available.

Reconfiguring is much like restarting from a previous checkpoint. In our experience with PCCM2, we often found it necessary to restart on a different number of processors due to computer access rules for interactive or batch jobs. This required developing checkpoint and restart procedures that were not dependent on the number of processors used. For example, with a 2-D domain decomposition, the restart data can be organized by column. On restart, each processor retrieves from the checkpoint files the columns it needs.

What we propose to implement for performance adaptation goes significantly beyond checkpoint and restart capabilities. The parallel algorithms, the chunk sizes and the low level messaging systems of the code may need to change in response to changing system conditions. One of the challenges of the research will be to identify what decisions can be delayed until runtime and how to best implement the response to a dynamic system change.

An API or coding convention will be developed and illustrated with atmospheric and ocean model components. Source translation tools will also be adapted to restructure existing codes to use these conventions and to keep appearance of the code simple.

Work plan, Timelines and Deliverables. We intend to use the battery of translation, encapsulation, and instrumentation tools and techniques in their current preliminary forms to implement several key codes quickly in performance-portable fashion over a range of platforms, providing a realistic set of single-source testbed codes. These codes will allow comprehensive exploration and evaluation of parameters affecting (and improving) performance on each platform. This will enable us to begin the performance characterization of the target platforms, and establish (in metrics relevant to the climate research community) the promise of the performance portability techniques in their current form. Specifically, we will:

- Introduce communication encapsulations (transport layer, protocol, and global communications);
- Introduce the parallel frameworks (parallel spectral transform, parallel semi-Lagrangian, physics column-based load balancing, grid partitioning);
- Employ run-time libraries
- Employ source analysis and translation tools to:
  - Uncover data-dependencies and shared/local data attributes for distributed and shared-memory parallelization,
  - Instrument code for performance monitoring, cache simulation, data memory-alignment,
  - Restructure loops,
  - Restructure data, and
  - Hide the effects of the new encapsulations, frameworks, and run-time libraries on the structure of the application code.

The resulting performance data will be one or more reports covering per-node performance and parallel efficiency over the range of code, data, and library modifications employed; the range of platforms investigated; and the several codes with attention to constituent kernels but emphasizing overall model performance. This

Table 2: Timeline

Month 0-12	-	Implement single-source versions of CCM, MOM or POP, MM5 on distributed-memory, shared-memory, and hybrid machines.
	-	Instrument and evaluate performance and sensitivity to permutations of code and data structure, protocols, domain size, etc.
	-	Refine and enhance tools and techniques. Develop whole-code translations, runtime API and mechanism, focusing in particular on shared-distributed memory.
	-	Report empirical results and conclusions.
Month 13-24	-	Continue instrumentation and evaluation of testbed codes. Apply and evaluate additional transformations, including whole-code analysis and restructuring of code and data.
	-	Complete development of run-time API, apply to testbed codes, and demonstrate performance adaptation.
	-	Demonstrate true performance portability with one of the target models over the full range of target architectures.
	-	Report results and conclusions. Release performance portability tools. Release demonstrated single source model code.
Month 25-42	-	Continue instrumentation and evaluation of testbed codes, including whole-code translations and transformations; run-time adaptation.
	-	Make available all tools, libraries, and codes. Report on final results and conclusions.

will shed light on underlying causal mechanisms – specific interactions between code structure and architecture and their affect on performance – towards a more general, formal understanding not only of what works with a given code and platform but also why. Benefits include support for

- compile- and/or run-time decision-making for performance adaptation,
- the ability to generalize about and anticipate performance issues for other existing or new computers, and
- information suggesting new and fruitful directions for additional investigation.

This will result in enhancement and refinement of the performance portability tools and techniques, yielding a "phase 2" set of tools and deliverables: single-source performance-portable versions of candidates codes, and additional follow-on reports, presentations, and journal publications to disseminate our findings and provide benefit to the community at large. A timeline outlining the work plan is provided in Table 2.

1.4 Subcontract or Consortium Arrangements

This proposal is joint between the Argonne National Laboratory (ANL) and the Oak Ridge National Laboratory (ORNL), involving the participation of John G. Michalakes (PI at ANL), Patrick H. Worley (PI at ORNL), Thomas R. Canfield (ANL). and John B. Drake (ORNL). This proposal also includes a subcontract to Applied Parallel Research, covering research collaboration with John Levesque and access to both commericial and internal APR tools. This consortium provides the necessary expertise in performance evaluation, performance tuning, source translation, compilers, and the computational and software engineering aspects of climate models, as indicated in the Preliminary Studies (§1.2) and the Biographical Sketches (§5) sections.

2 Literature Cited

References

[1] S. R. M. Barros, D. Dent, L. Isaksen, G. Robinson, G. Mozdzynski, and F. Wollenweber. The IFS model: A parallel production weather code. *Parallel Computing*, 21(10):1621–1638, 1995.

[2] OpenMP Architecture Review Board. OpenMP: A proposed standard api for shared memory programming. (available from http://www.openmp.org/openmp/mp-documents/paper/paper.ps), October 1997.

[3] J. Brehm, P. H. Worley, and M. Madhukar. Performance modeling for SPMD message-passing programs. Technical Report ORNL/TM-13254, Oak Ridge National Laboratory, Oak Ridge, TN, June 1996. (also, *Concurrency: Practice and Experience* (accepted)).

[4] Rob F. Van der Wijngaart. Efficacy of code optimizations on cache-based processors. Technical Report NAS-97-012, NASA Ames Research Center, Moffett Field, CA, April 1997.

[5] J. B. Drake and I. T. Foster. Special issue on parallel computing in climate and weather modeling. *Parallel Computing*, 21(10), 1995.

[6] J. B. Drake, I. T. Foster, J. J. Hack, J. G. Michalakes, B. D. Semeraro, B. Tonen, D. L. Williamson, and P. H. Worley. PCCM2: A GCM adapted for scalable parallel computer. In *Fifth Symposium on Global Change Studies*, pages 91–98. American Meteorological Society, Boston, 1994.

[7] J. B. Drake, I. T. Foster, J. G. Michalakes, B. Toonen, and P. H. Worley. Design and performance of a scalable parallel community climate model. *Parallel Computing*, 21(10):1571–1591, 1995.

[8] I. Foster and J. Michalakes. MPMM: A Massively Parallel Mesoscale Model. In Geered-R Hoffmann and Tuomo Kauranne, editors, *Parallel Supercomputing in Atmospheric Science*, pages 354–363. World Scientific, River Edge, New Jersey, 1993.

[9] I. T. Foster, B. Toonen, and P. H. Worley. Performance of parallel computers for spectral atmospheric models. *J. Atm. Oceanic Tech*, 13(5):1031–1045, 1996.

[10] I. T. Foster and P. H. Worley. Parallel algorithms for the spectral transform method. *SIAM J. Sci. Comput.*, 18(3):806–837, May 1997.

[11] R. Friedman, J. Levesque, and G. Wagenbreth. *Fortran Parallelization Handbook.* Applied Parallel Research Inc., Sacramento, California, June 1995.

[12] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. PICL: a portable instrumented communication library, C reference manual. Technical Report ORNL/TM-11130, Oak Ridge National Laboratory, Oak Ridge, TN, July 1990.

[13] G. A. Geist, J. A. Kohl, and P. M. Papadopoulus. CUMULVS: Providing fault-tolerance, visualization, and steering of parallel applications. *International Journal of Supercomputer Applications and High Performance Computing*, 11(3):224–236, 1997.

[14] G. A. Geist, J. A. Kohl, P. M. Papadopoulus, and S. Scott. Beyond PVM 3.4: What We've Learned, What's Next, and Why. In *Proceedings of the EuroPVM-MPI 97*, 1997.

[15] Georg A. Grell, Jimmy Dudhia, and David R. Stauffer. A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5). Technical Report NCAR/TN-398+STR, National Center for Atmospheric Research, Boulder, CO, June 1994.

[16] J. J. Hack. personal communication. See also http://www.ucar.edu/staffnotes/9705/scibrief.html.

[17] S. W. Hammond, R. D. Loft, J. M. Dennis, and R. K. Sato. Implementation and performance issues of a massively parallel atmospheric model. *Parallel Computing*, 21(10):1593–1620, 1995.

[18] M. T. Heath and J. A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29–39, September 1991.

[19] R. Hempel and H. Ritzdorf. The GMD Communications Library for Grid-oriented Problems. Technical Report GMD-0589, German National Research Center for Information Technology, 1991.

[20] M. D. Hill and A. J. Smith. Evaluating associativity in cpu caches. *ACM Trans. Computers*, 38(12):1612–1630, 1989.

[21] G.-R. Hoffman and T. Kauranne, editors. *Parallel Supercomputing in Atmospheric Science: Proceedings of the Fifth ECMWF Workshop on Use of Parallel Processors in Meteorology.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1993.

[22] G.-R. Hoffman and N. Kreitz, editors. *Coming of Age: Proceedings of the Sixth ECMWF Workshop on Use of Parallel Processors in Meteorology.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1995.

[23] G.-R. Hoffman and D. K. Maretis, editors. *The Dawn of Massively Parallel Processing in Meteorology: Proceedings of the Fourth ECMWF Workshop on Use of Parallel Processors in Meteorology.* Springer-Verlag, Berlin, 1990.

[24] G.-R. Hoffman and D. F. Snelling, editors. *Multiprocessing in Meteorological Models: Proceedings of the Third ECMWF Workshop on Use of Parallel Processors in Meteorology.* Springer-Verlag, New York, 1988.

[25] K. Kennedy and S. Carr. Improving the ratio of memory operations to floating-point operation s in loops. *ACM Transactions on Programming Languages and Systems*, 16(6):1768–1810, 1994.

[26] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, and M. Zosel. *The High Performance Fortran Handbook.* MIT Press, Cambridge, 1994.

[27] S. R. Kohn and S. B. Baden. A Parallel Software Infrastructure for Structured Adaptive Mesh Methods. In *Proceedings of Supercomputing '95*, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[28] A. R. Lebeck and D. A. Wood. Cache profiling and the spec benchmarks: A case study. *IEEE Computer*, 27(10):15–27, October 1994.

[29] J. Michalakes. FLIC: A Translator for Same-source Parallel Implementation of Regular Grid Applications. Technical Report ANL/MCS-TM-223, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, March 1997.

[30] J. Michalakes. MM90: A Scalable Parallel Implementation of the Penn State/NCAR Mesoscale Model (MM5). *Parallel Computing*, 23(14):2173–2186, December 1997. also Argonne National Laboratory preprint ANL/MCS-P659-0597.

[31] J. Michalakes. RSL: A Parallel Runtime System Library for Regional Atmospheric Models with Nesting. *to appear in proceedings of the IMA workshop "Structured Adaptive Mesh Refinement Grid Methods," March 12-13, Minneapolis*, 1997. Also preprint ANL/MCS-P663-0597.

[32] J. Michalakes. Runtime System Library for Parallel Finite Difference Models with Nesting. Technical Report ANL/MCS-TM-197, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, March 1997.

[33] J. Michalakes. Same-Source Parallel Implementation of the PSU/NCAR MM5. *to appear in proceedings of the "Symposium On Regional Weather Prediction on Parallel Computer Environments," October 15–17, 1997, Universit y of Athens, Greece*, 1998. Also preprint ANL/MCS-P702-1297.

[34] J. Michalakes, T. Canfield, R. Nanjundiah, S. Hammond, and G . Grell. Parallel Implementation, Validation, and Performance of MM5. In *Coming of Age: Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology*, pages 266–276. World Scientific, River Edge, New Jersey, 1995.

[35] MPI Committee. MPI: a message-passing interface standard. *Internat. J. Supercomputer Applications*, 8(3/4):165–416, Fall/Winter 1994.

[36] M. Parashar and J. C. Browne. Distributed dynamic data-structures for parallel adaptive mesh-refinement. *Proceedings of the International Conference for High Performance Computing*, pages 22–27, December 1995.

[37] B. Rodriguez, L. Hart, and T. Henderson. A Library for the Portable Parallelization of Operational Weather Forecast Models. In G.-R. Hoffman and N. Kreitz, editors, *Coming of Age: Proceedings of the Sixth ECMWF Workshop on Use of Parallel Processors in Meteorology*, pages 148–161. World Scientific Publishing Co. Pte. Ltd., Singapore, 1995.

[38] A. Sawdey, M O'Keefe, and W. Jones. A general programming model for developing scalable ocean circulation applications. In *Proceedings of the Seventh ECMWF Workshop on Use of Parallel Processors in Meteorology, Reading, England, December 1996.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1997. (in press).

[39] D. W. Walker, P. H. Worley, and J. B. Drake. Parallelizing the spectral transform method. Part II. *Concurrency: Practice and Experience*, 4(7):509–531, October 1992.

[40] H. J. Wasserman, O. M. Lubeck, Y. Luo, and F. Bassetti. Performance evaluation of the SGI Origin2000: A memory-centric characterization of LANL ASCI applications. Technical Report LA-UR-97-3464, Los Alamos National Laboratory, Los Alamos, NM, 1997. (also, *Supercomputing '97*, San Jose, CA).

[41] P. H. Worley. MPI performance evaluation and characterization using a compact application benchmark code. In *Proccedings of the Second MPI Developers Conference and Users' Meeting*, pages 170–177. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[42] P. H. Worley and J. B. Drake. Parallelizing the spectral transform method. *Concurrency: Practice and Experience*, 4(4):269–291, June 1992.

[43] P. H. Worley and I. T. Foster. Parallel Spectral Transform Shallow Water Model: a runtime–tunable parallel benchmark code. In J. J. Dongarra and D. W. Walker, editors, *Proc. Scalable High Performance Computing Conf.*, pages 207–214. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[44] P. H. Worley, I. T. Foster, and B. Toonen. Algorithm comparison and benchmarking using a parallel spectral transform shallow water model. In G.-R. Hoffman and N. Kreitz, editors, *Coming of Age: Proceedings of the Sixth ECMWF Workshop on Use of Parallel Processors in Meteorology*, pages 277–289. World Scientific Publishing Co. Pte. Ltd., Singapore, 1995.

[45] P. H. Worley, A. C. Robinson, D. R. Mackay, and E. J. Barragy. A study of application sensitivity to variation in message passing latency and bandwidth. Technical Report ORNL/TM-13250, Oak Ridge National Laboratory, Oak Ridge, TN, June 1996. (also, *Concurrency: Practice and Experience* (accepted)).

[46] P. H. Worley and B. Toonen. A users' guide to PSTSWM. Technical Report ORNL/TM–12779, Oak Ridge National Laboratory, Oak Ridge, TN, July 1995.

3 Biographical Sketches

3.1 Thomas R. Canfield

CONTACT

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois, 60439
phone: 630-252-4588
fax: 630-252-5986
e-mail: canfield@mcs.anl.gov.gov
WWW: http://www.mcs.anl.gov/people/canfield.

EDUCATION

Ph.D. (Applied Mechanics) University of Illinois at Chicago, 1980.
M.S. (Applied Mechanics) University of Illinois at Chicago, 1975.
B.S. (Structural Engineering) University of Illinois at Chicago, 1974.

PROFESSIONAL EXPERIENCE

Research Staff,	Argonne National Laboratory,	1988–present.
Faculty,	Loyola University of Chicago,	1986–1988.
Senior Structural Analyst,	Sargent & Lundy, Engineers,	1985–1986.
Member of the Technical Staff,	Sandia National Laboratory,	1980–1985.
Senior Engineer,	Motorola, Inc.,	1978–1980.

RESEARCH INTERESTS AND EXPERTISE

Tom Canfield's research interest are in the development of parallel algorithms and methodology for modeling solid and fluid dynamic systems. He has been involved in the development of high performance computer applications on a wide variety of computer architectures. He developed a specialized computer program, MARBLE, to model dynamic penetration of subseabed sediments for the Subseabed Disposal Program on Cray Vector computers. He developed custom finite elements for modeling piezoelectric crystals the Intel Touchstone Delta. He was involved in the development of a parallel program, FIFEA (Fast Implicit Finite Element Analysis), for use in the analysis of automotive disk brakes. More recently he developed a reduced grid implementation of PSTSWM the parallel testbed and benchmarking code for the shallow water equations. He is currently involved in the development and enhancement of the CHAD a parallel transient thermal hydraulics analysis program that will be used to analyze reactor subassemblies and automotive subsystems.

SELECTED PUBLICATIONS

1. V. E. Taylor, J. Chen, T. Canfield, and R. Stevens, "A Decomposition Method for Efficient Use of Distributed Supercomputers for Finite Element Applications," *Proceedings of the International Conference on Applications-Specific Systems, Architectures and Processors*, August 18–21, 1996.
2. T. Canfield, T. Disz, M. Papka, R. Stevens, M. Huang, V. Taylor, and J. Chen, *Toward Real-time Interactive Virtual Prototyping of Mechanical Systems: Experience Coupling Virtual Reality with Finite Element Analysis*, to be presented at High Performance Computing 1996, Soc. Comp. Sim., New Orleans, April 1996.
3. T. R. Canfield and Philip B. Dobrin, "Mechanics of Blood Vessels," chapter in *Biomedical Engineering Handbook,* CRC Press, Boca Raton, Florida, 1995.
4. J. Michalakas, T. Canfield, R. Nanjundiah, S. Hammond, and G. Grell, "Parallel Implementation, Validation, and Performance of MM5," preprint MCS-P483-1194, 1995.

5. V. E. Taylor, R. Stevens, and T. Canfield, *Performance Models of Interactive, Immersive, Visualization for Scientific Applications*, Proc. International Workshop on High Performance Computing for Computer Graphics and Visualization in Swansea, UK, July 3–4, 1995.

6. A. M. Tentner, R. N. Blomquist, T. R. Canfield, P. L. Garner, E. M. Gelbard, K. C. Gross, M. Minkoff, and R. A. Valentin, *Advances in Parallel Computing for Reactor Analysis and Safety*, 1993.

7. T. Canfield, M. Jones, P. Plassmann, and M. Tang, "Modeling Piezoelectric Crystals on the Intel Delta," Proc. Sixth SIAM Conference on Parallel Processing for Scientific Computing, ed. R. Sincovec et al., SIAM, Philadelphia, 1993.

3.2 John B. Drake

CONTACT
Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge TN 37831-6367
phone: 423-574-8670
fax: 423-574-0680
e-mail: drakejb@ornl.gov
WWW: http://www.epm.ornl.gov/~bbd/

EDUCATION

Ph.D.	(Mathematics)	University of Tennessee,	1991.
M.S.	(Applied Mathematics)	Purdue University,	1979.
B.A.	(Mathematics)	University of Kentucky,	1977.

PROFESSIONAL EXPERIENCE
Research Staff, Mathematical Sciences Oak Ridge National Laboratory, 1984–present.
Research Staff, Engineering Mechanics Oak Ridge National Laboratory, 1979–1984.

PROFESSIONAL ACTIVITIES
Member of SIAM.
Host of "PDE's on the Sphere Conference" 1995.
Organizing committee for other "PDE's on the Sphere" meetings.
Technical Reviewer for several journals and for proposals from DOE and NSF.

RESEARCH INTERESTS AND EXPERTISE
John Drake's research interests include

- Numerical methods and parallel algorithms for climate modeling.
- Numerical solution of partial differential equations and integral equations.

The development of parallel algorithms for the solution of partial differential equations on distributed memory multiprocessors is the current focus of his work. Fluid flow algorithms for massively parallel, distributed memory computers, used in the simulation of atmospheric flows and in plasma turbulence modeling are the subject of continuing projects. The formulation of atmospheric flow equations to utilize semi-Lagrangian advection algorithms and high order spatial discretizations is also a part of his research. Moving boundary problems associated with change of phase in thermal and fluid flow problems were the focus of his doctoral research. This research employed finite element and control volume methods in a "weak" formulation applied to the Navier-Stokes equations and the equation of energy conservation.

FIVE RECENT PUBLICATIONS RELATED TO THIS PROJECT

1. *The Cartesian Method for solving partial differential equations in spherical geometry*, with P.N. Swarztrauber, D.L. Williamson, to appear in Dynamics of Oceans and Atmospheres, 1998.
2. *A Parallel Performance Study for the Cartesian Method on a Sphere*, with Matthew P. Coddington, ORNL/TM-13304, February 1997
3. *Parallel Computing Special Issue: Introduction to Weather and Climate Modeling*, with I.T. Foster, Parallel Computing, Vol 21, No.10, pp. 1537-1544, October 1995
4. *Design and Performance of a Scalable Parallel Community Climate Model*, with I.T. Foster, J.G. Michalakes, Brian Toonen and P.H. Worley, Parallel Computing, Vol 21, No.10, pp. 1571-1591, October 1995
5. *Parallel Algorithms for Semi-Lagrangian Transport in Global Atmospheric Circulation Models*, with I.T. Foster, J.G. Michalakes and P.H. Worley, in Parallel Processing for Scientific Computing, SIAM, Feb. 1995, pp. 119-124.

3.3 John Levesque

CONTACT
Applied Parallel Research
1723 Professional Drive
Sacramento, CA 95825
phone: 916-481-9891
fax: 916-481-7924
e-mail: `Levesque@apri.com`
WWW: `http://www.apri.com`.

EDUCATION
M.A. (Mathematics) University of New Mexico, 1970.
B.A. (Mathematics) University of New Mexico, 1968.

PROFESSIONAL EXPERIENCE
President, Applied Parallel Research, 1992–present.
Vice President, Pacific Sierra Research, 1979–1992.

RESEARCH INTERESTS AND EXPERTISE
John Levesque's research interests are in the evaluation of optimization techniques and software for current high speed scientific processors and in the design of software tools for automating the optimization process.

Levesque's optimization work ranges from computational fluid dynamics to ocean modeling and weather simulation. He has expertise on the full range of Intel and nCUBE parallel platforms and on the Cray Research T3D and T3E, IBM SP, and Convex Exemplar SPP, and knowledge of optimization techniques for a single processors chip designs from SGI, Digital, IBM, HP and Intel. He was a principal designer of the FORGExplorer FORTRAN programming environment and Applied Parallel Research's HPF and shared memory parallelization systems.

Levesque's work begins with the investigation of existing FORTRAN production codes for optimization for a range of potential targets. In the process of the optimization, ideas are formulated for incorporation into APR's tools. The intent of the process is to achieve an automatic process that has a high probability for generating optimized code for a range of target systems. Levesque's work includes sequential and parallel optimization. In a recent project on the POP ocean model from Los Alamos National Laboratory, the FORTRAN 90 program that was running inefficiently was converted to a cleaner FORTRAN 90 program that could to automatically optimized to achieve close to 100 MFLOPS.

3.4 John G. Michalakes

CONTACT
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois, 60439
phone: 630-252-6646
fax: 630-252-5986
e-mail: michalak@mcs.anl.gov.gov
WWW: http://www.mcs.anl.gov/people/michalakes.

EDUCATION

M.S. (Computer Science)	Kent State University,	1988.
B.A. (English)	Cleveland State University,	1984.

PROFESSIONAL EXPERIENCE
Research Staff, Argonne National Laboratory, 1989–present.

RECENT PROFESSIONAL ACTIVITIES
Member of ACM and IEEE Computer Society
Reviewer for 2 journals, NASA, NSF, and numerous conferences
Scientific and Organization Committee member of First International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications
Scientific Committee member of 2nd International Workshop on Software Engineering ....
Guest co-editor of *Parallel Computing* special issue on regional weather modeling, December 1997

RESEARCH INTERESTS AND EXPERTISE
John Michalakes's research interests are development of tools and programming techniques for scalable parallel implementation of scientific computer codes, focusing the past seven years on global and regional atmospheric models. Research includes evaluation and remediation of load imbalances, development of high-level application specific run-time libraries; dynamic, flexible, and modular code methodologies exploiting Fortran90; and automatic source code analysis and translation for single-source.

Michalakes performed the first comprehensive and quantitative analysis of GCM load imbalance by instrumenting and studying CCM2. This and a follow on study provided the basis for successful remediation of load imbalances in short-wave radiation physics. Michalakes was a member of the software team (with Worley and Drake) that developed PCCM2, the message passing version of CCM2, under the CHAMMP program. He has also worked (with Canfield) on the reduced-grid version of the parallel spectral shallow-water testbed code (in development).

Michalakes led the group of Argonne researchers who developed the first scalable parallel implementation of the PSU/NCAR Mesoscale Model, now in use operationally at the Air Force Weather Agency, Offutt AFB, Nebraska, and at other sites. As part of this work, Michalakes developed the the Runtime System Library, which abstracts low-level communication, run-time decomposition and reallocation for load balancing as well as interprocessor communication supporting 2-way interacting nested grids; MM90, a Fortran90 version of the model that employed dynamically allocated domain *objects* and supported irregularly shaped nested domains (a necessary precursor to true AMR in atmospheric models) as well as dynamic run-time allocation of work to processors for load balancing. Additional work has focused on four-dimensional data assimilation in MM5, and current interest includes parallelization of the adjoint MM5 for four-dimensional variational assimilation.

Michalakes has extensive experience with computer languages, translators, and compilers. He wrote the Associative Processing Compiler while at Kent State University. At Argonne he has developed numerous

Fortran translation and analysis tools applicable to weather and climate modeling, including the Fortran Loop and Index Converter (FLIC), a Fortran array index reordering utility (Twst), and a data dependency analyzer (depend), all of which have been used in the MM5 project. FLIC is the translation engine that facilitates the latest "same-source" implementation of the parallel MM5 that is being adopted into the official version of the model at NCAR.

RECENT PUBLICATIONS RELATED TO THIS PROJECT

1. *Regional Weather Modeling on Parallel Computers* with C. Baillie and R. Skålin, Parallel Computing, 23(14), December, 1997, pp. 2135–2142.
2. *MM90: A Scalable Parallel Implementation of the Penn State/NCAR Mesoscale Model (MM5)* Parallel Computing, 23(14), December, 1997, pp. 2173–2186.
3. *Design and Performance of a Scalable Parallel Community Climate Model,* with J. Drake, I. Foster, B. Toonen, and P. Worley, Parallel Computing 21(10), October 1995, pp. 1571-1592.
4. *RSL: A Parallel Runtime System Library for Regional Atmospheric Models with Nesting* to appear in the proceedings of the IMA Workshop on Structured Adaptive Mesh Refinement Grid Methods, March 12-13, 1997, Minneapolis. S. Baden, N. Chrisochoides, D. Gannon, and M. Norman, editors. Also ANL preprint ANL/MCS-P663-0597.
5. *Same-Source Parallel Implementation of the PSU/NCAR MM5* to appear in the proceedings of the "Symposium On Regional Weather Prediction on Parallel Computer Environments," October 15–17, 1997, University of Athens, Greece. G. Kallos and Y. Cotronis, editors. Also ANL preprint ANL/MCS-P702-1297.
6. *FLIC: A Translator for Same-source Parallel Implementation of Regular Grid Applications* ANL Technical Report ANL/MCS-TM-223, March 1997.
7. *Computational Load in Model Physics of the Parallel NCAR Community Climate Model* with R. Nanjundiah, ANL Technical Report ANL/MCS-TM-186, January, 1994.

3.5 Patrick H. Worley

CONTACT

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge TN 37831-6367
phone: 423-574-3128
fax: 423-574-0680
e-mail: `worleyph@ornl.gov`
WWW: `http://www.epm.ornl.gov/~worley/`

EDUCATION

Ph.D.	(Computer Science)	Stanford University,	1988.
M.S.	(Computer Science)	Stanford University,	1983.
B.A.	(Computer Science)	Indiana University,	1980.
B.A.	(Mathematics)	Indiana University,	1980.

PROFESSIONAL EXPERIENCE

Research Staff, Oak Ridge National Laboratory, 1987–present.

RECENT PROFESSIONAL ACTIVITIES

Member of ACM and SIAM
Member of editorial board for the Electronic Journal of Performance Evaluation and Modeling
Member of PARKBENCH (PARallel Kernels and BENCHmarks) committee
Moderator of ACM SIGMETRICS Electronic BBoard
Co-chair of 1997 Gordon Conference on High Performance Computing
Tutorial speaker on performance evaluation at SC97
Reviewer for 7 journals, DOE, NSF, and numerous conferences

RESEARCH INTERESTS AND EXPERTISE

Patrick Worley's research interests are in the design, implementation, and evaluation of parallel algorithms, in the performance evaluation of high performance computer systems, and in the design and evaluation of numerical algorithms for the solution of differential equations.

Worley's parallel algorithm work ranges from theoretical investigations into what is feasible based on information theoretic concepts, including seminal work on the limitations of scalability, to the design and implementation of parallel algorithms in atmospheric and ocean simulation models. He has expertise on the full range of Intel and nCUBE parallel platforms and on the Cray Research T3D and T3E, IBM SP, and Convex Exemplar SPP. He was a coauthor of PCCM2, a massively parallel implementation of the Community Climate Model CCM2.

Worley's work on performance evaluation has three aspects: performance data collection, visualization, and analysis, benchmarking and benchmarking methodology, and performance portability. Worley is the author of the performance data collection facility in the Portable Instrumented Communication Library (PICL), which he continues to maintain and extend for use in evaluation work. Recent research includes constructing and evaluating models for the performance prediction of parallel programs, and comparing the performance characteristics of the Message Passing Interface between parallel systems. Worley was the primary author of the tunable benchmark code PSTSWM, and has an active interest in techniques for "fair" benchmarking and in programming styles and tools that support performance portability.

Worley's numerical work includes research into novel algorithms that are suitable for parallel implementation and the implementation and evaluation of numerical algorithms for use in atmospheric and ocean models. Recent research includes an evaluation of proposed fast Legendre transform algorithms in the context of spectral atmospheric models.

RECENT PUBLICATIONS RELATED TO THIS PROJECT

1. *Parallel algorithms for the spectral transform method,* with I. Foster Sci. Stat. Comput., 18(3), 1997, pp. 806-837.
2. *Performance of parallel computers for spectral atmospheric models,* with I. Foster and B. Toonen J. Atm. Oceanic Tech., 13(5), 1996, pp. 1031-1045.
3. *A Study of Application Sensitivity to Variation in Message Passing Latency and Bandwidth* with A. Robinson, D. Mackay, and E. Barragy, ORNL Technical Report ORNL/TM-13250, June 1996 (also, Concurrency: Practice and Experience (accepted)).
4. *Performance Modeling for SPMD Message-Passing Programs* with J. Brehm and M. Madhukar, ORNL Technical Report ORNL/TM-13254, June 1996 (also, Concurrency: Practice and Experience (accepted)).
5. *MPI performance evaluation and characterization using a compact application benchmark code* Proceedings of the Second MPI Developers Conference and Users' Meeting, IEEE Computer Society Press, 1996, pp. 170-177.
6. *Design and Performance of a Scalable Parallel Community Climate Model,* with J. Drake, I. Foster, J. Michalakes, and B. Toonen, Parallel Computing 21(10), October 1995, pp. 1571-1592.

4 Description of Facilities and Resources

The research proposed will be performed at the Argonne National Laboratory and the Oak Ridge National Laboratory. The computing facilities we will use at ORNL for the development of test codes and testing methodology include the Intel Paragons in the Center for Computational Science (CCS), a 4 processor SGI Oynx and local workstation clusters. At Argonne we will use the Argonne Quad Machine in the Center for Computational Science and Technology (CCST) consisting of an IBM SP with 80 "thin" nodes and two 8-way SMP nodes. A detailed description is provided at http://www.mcs.anl.gov/CCST.

Performance evaluation and validation of the performance portability techniques will concentrate initially on the Cray T3E in the National Energy Research Supercomputing Center (NERSC) at LBL and the SGI Origin in the Advanced Computing Laboratory (ACL) at LANL, as well as the Paragon and IBM SP systems.

We currently have access and sufficient computing time available on all these systems.

A Appendix

Papers by the investigators cited in this proposal can be found at the following URL:

http://www.epm.ornl.gov/~worley/LAB98-05/index.html

A letter signifying APR and John Levesque's willingness to participate in this proposal follows.