Cross-Platform Performance of a Portable Communication Module and the NASA Finite Volume General Circulation Model

William M. Putman [a,*], Shian-Jiann Lin [b], Bo-Wen Shen [c,a]

[a] *NASA Goddard Space Flight Center*
[b] *NOAA Geophysical Fluid Dynamics Laboratory*
[c] *Science Applications International Corporation (SAIC)*

Abstract

The NASA finite volume general circulation model (fvGCM) is a global atmospheric model, originally developed for long-term climate simulations. Recently, the NASA fvGCM has been applied in a variety of weather prediction applications including hurricane and winter storm forecasts. Achieving efficient throughput on a variety of computational platforms is essential to meet the needs of the climate and weather prediction community. We have developed a scalable and portable climate/weather prediction system by applying a portable communication module within a fast numerical algorithm that exceeds the current community demands for computational performance on a variety of high performance computing platforms. The low-level communication module, Mod_Comm, simplifies inter-process communication within general circulation models (GCMs) and provides an efficient means of communicating between decomposed global domains using a variety of single-threaded and multi-threaded data communication paradigms (MPI-1, MPI-2, SHMEM, and MLP). Mod_Comm has been implemented within the NASA finite-volume GCM (fvGCM) and the community atmosphere model (CAM) at NCAR. It is shown that the optimal choice of data communication paradigm varies from system to system, and can have a significant impact on the overall model performance. Performance studies with the NASA fvGCM reveal substantial improvements in the computational performance when using this low-level communication module, throughput improvements of 40 percent or more have been observed on various platforms including the SGI Altix 3700, SGI Origin 3000, Compaq AlphaServerSC, IBM SP, and Cray.

Key words: Finite-Volume GCM, Communication Library, Scalability, Portability, Benchmark, MPI-1, MPI-2, SHMEM, MLP
PACS:

1 Introduction

The NASA fvGCM was developed as a joint effort between the NASA Data Assimilation Office (DAO) and the National Center for Atmospheric Research (NCAR). Originally developed for climate applications and data assimilation, the NASA fvGCM has recently been applied in various numerical weather prediction (NWP) applications in at 0.5-degree horizontal resolution (55km). These NWP applications include 5-10 day weather forecasting, hurricane prediction, and atmospheric chemistry (forecasting of ozone concentrations and transport of a variety of chemical/biological/radioactive species). The NASA fvGCM resolves the atmosphere in the form of finite control volumes that cover the earth from pole-to-pole and surface-to-thermosphere (80 km above the ground). Each control volume contains all of the dynamic state variables (wind, temperature, moisture and pressure) to model the atmosphere[1–5]. The physics and land components of the NASA fvGCM are based on community built parameterizations developed at NCAR encompassing clouds, solar heating, infrared cooling, evaporation/condensation of moisture, and others[6].

Parallelization is achieved through a hybrid distributed memory and multi-threaded parallelism approach. The global domain is decomposed over latitudes for single program multiple data (SPMD) parallelism. Further parallelism is attained via multi-threading directives using OpenMP in the vertical direction in the dynamics (the physics uses OpenMP directives over the decomposed latitudes). Efficient data communication is acheived via a low-level general communication module (Mod_Comm) developed for the NASA fvGCM. This module provides a simple interface to common communication operations within the NASA fvGCM and allows for compile-time switching among a variety of single-threaded and multi-threaded data communication paradigms (MPI-1[7], MPI-2[8], SHMEM[9], and MLP [10]) commonly used on today's high-end computers. The ability to easily switch between data communication paradigms provides a level of portablility critical to acheiving scalable performance across a variety of platforms.

In this article, we describe the development of Mod_Comm within the NASA fvGCM. The performance of various data communication paradigms within Mod_Comm for typical communication operations is examined for a variety of problem sizes on SGI, Compaq, and IBM systems. The overall performance of the NASA fvGCM is described across these systems, and the communication versus computational efficiency is explored in detail for each computational platform. These performance measures are obtained for a typical high reso-

* Corresponding author.
Email addresses: William.M.Putman@nasa.gov (William M. Putman), Shian-Jiann.Lin@noaa.gov (Shian-Jiann Lin), bshen@gmao.gsfc.nasa.gov (Bo-Wen Shen).

lution weather forcasting scenario and a lower resolution climate prediction case.

2 The Evolution of Mod_Comm

We have developed a general communication module, written in fortan90, to control all SPMD parallelism within the NASA fvGCM. This module was designed to isolate all communication operations in one complete and concise module keeping the main source code easy to read and update for the scientific developers, while also providing the highest possible communication performance across all applicable computing platforms. This was accomplished by defining two levels of routines within Mod_Comm, a user interface level and a low-level set of routines to perform all data communication and synchronization. The design allows for easy and efficient overlap of computation with communication, and performs all synchronization within Mod_Comm providing an absolute minimum number of global synchronizations which can dramatically degrade overall model performance. In addition, by seperating the data communication routines from the user interface level we were able to effectively resuse common code minimizing the total lines of code and simpifying the process of updating the communication module to include other data communication paradigms.

Mod_Comm currently supports MPI-1, MPI-2, SHMEM and MLP (multi-level parallelism, an SGI Origin specific inter-process communication paradigm developed at NASA Ames Research Center which uses the shared memory features of SGI Origin systems to perform all communication). In addition to general data communication, a set of simple parallel input/output routines have been implemented to allow a user to read/write from multiple SPMD processes to a single IEEE binary file. These routines have been implemented using MPI-IO, an MPI-2 specific feature. The ability to easily switch between these data communication paradigms using C preprocessor tokens at compile-time allows us to effectively use the most efficient data communication calls which can vary considerably from platform to platform.

2.1 The User Interface Level

Often times, a science developer working on a GCM is not familiar with the technical demands of data communication in a parallel programming environment. Modifying code, and performing common communication operations for such a developer may take an extensive period of time. The user interface level of Mod_Comm was written to simplify this process to relieve the demands on a

science developer to understand all aspects of parallel computing from process management to synchronization.

The user interface level provides functionality for process initialization and finalization, domain decompostion, halo exchanges (ghosting), global reductions, scatter/gather operations, and parallel input/output (I/O) operations. Simple calls to `mp_init` and `y_decomp` perform all process initialization, domain decompostition, and allocate memory required to perform all data communication tasks throughout the simulation of the GCM. These tasks are performed based on the simple user input of global domain sizes and the desired number of computational processes. After initialization, an implementor is only required to understand where a communication operation is required (ie, when data is needed from another process) and what type of operation is necessary (ghosting, collective, etc.). Synchronization, one of the most critical aspects of parallel computing in terms of potential performance impacts, is completely controlled within Mod_Comm. A developer will never need to perform a synchronization operation from the main source code when using Mod_Comm; this is optimally controlled to effectively reduce the CPU (central processing unit) time spent waiting in unecessary barrier calls.

Ghosting and collective communication operations are provided with a simple interface requiring the user to supply the appropriate decomposed and global arrays and extents. The communication module then performs any synchronization required, packs all data into appropriate buffers and sends the messages to the appropriate destinations. For most operations, the user does not need to know which processor should receive the data, this is controlled based on the dimensions supplied through the argument lists. In addition, these operations allow for effective overlapping of computation with communication. A user can initiate a series of send calls, perform any overlapping work, and then return to receive the appropriate data once the computation completes. This overlaying of computation during communication can substantially mask the impact of communication on the total run time of the application.

2.2 The Low-Level Communication Routines

The user interface routines within Mod_Comm essentially perform all of the organizational tasks required for the communcation operations; a series of low-level communication routines within Mod_Comm do most of the work required in sending, receiving and synchronizing the messages involved and are called from the user interface routines. The low-level routines are coded based on a *global array* design. The intent of the *global array* design is to provide a series of routines for communicating with various datatypes (real*8, real*4, integer*4) using predefined buffers that, depending on the data communi-

cation paradigm, are truly globally accessible by all other processes, or emmulate global accessibility through direct message-passing between processes. Each of the buffers are partitioned based on the domain decomposition and a predefined maximum number of consecutive communication calls allowed. This partitioning ensures that required data is not overwritten during a communication event, and also aides in minimizing the total amount of global synchronizations required. All communication operations pass through these *global array* routines effectively reducing the total lines of code required by reusing common code segments. The *global array* routines also greatly reduce the work required to implement new data communication paradigms within Mod_Comm.

The *global array* approach is based on communication windows and epochs as defined in the MPI-2 documentation. A communication epoch is defined as a data communication event, beginning with the initial request to send and/or receive data to the final synchronization or confirmation of delivery/receipt. Communication windows are defined as direct pathways to a specific memory location on another process. For data communication paradigms which perform one-sided message-passing communication (ie, PUT/GET) this is an obvious definition, for those that require a send/receive handshake (ie, MPI-1) the window is setup as complimentary buffers which exist on all processes but do not directly share memory locations.

2.3 The Message-Passing Paradigms

Our MPI-2 implementation uses solely the `MPI_PUT` one-sided communication operations. A call to `MPI_PUT` sends data through a supplied MPI-2 window directly into the memory space of the receiving process. The MPI-2 implementation uses active target synchronization; the receiving process is actively involved in the communcation during the window synchronization step by calling the `MPI_WIN_FENCE` routine (it does not need to perform a receive call, and thus it is a one-sided communication). On applicable systems, a set of assertions are used to determine the level of synchronization required. These assertions can greatly reduce the cost of the `MPI_WIN_FENCE` synchronization. In addition, the thread-safe capabilities of MPI-2 are exploited on shared memory systems by breaking up large messages based on the number OpenMP threads being used and multi-threading the calls to `MPI_PUT`. This multi-threaded message-passing[11] provides a substantial increase in performance on SGI Origin systems, allowing the MPI-2 implementation to considerably outperform the MPI-1 implementation. It is noted that multi-threading of communication is not supported on the SGI Altix systems, and can degrade performance on certain platforms such as the IBM and Compaq systems.

The SHMEM implementation is very similar to MPI-2. It uses the one-sided `SHMEM_PUT` operation, and all processes are actively involved in the communication by calls to the `SHMEM_BARRIER` routines. Windows are not explicity defined within shmem, the global arrays are set up as shared memory accessible regions during the memory allocation process. The SHMEM routines are also safely multi-threaded, further improving the SHMEM performance.

MLP (multi-level parallelism) is a unique communication paradigm. It is a shared memory inter-process communication paradigm developed at NASA Ames Research Center, and is an SGI Origin specific implementation. MLP uses the shared memory features of the UNIX system to define globally accessible arrays that can be accessed directly by all SPMD processes. The main benefit of this method is a very simple coding style in which data communication events are coded simply as copies in/out of the global memory locations. The performance of MLP on SGI Origin systems is subtantially better than MPI-1, however falls short of MPI-2 as will be described in later sections of this article.

The MPI-1 implementation is the most portable of those within Mod_Comm. It utilizes exclusively non-blocking `MPI_ISEND` and `MPI_IRECV` routines. All message synchronization within Mod_Comm for MPI-1 is performed with `MPI_WAIT` routines. The `MPI_WAIT` routines are called to wait for message delivery or completion at the ideal time when it is needed, thus optimally reducing the total time spent waiting on communication (global `MPI_BARRIER` calls are never needed for the MPI-1 implementation within Mod_Comm). This weak synchronization feature of our MPI-1 implementation provides subtantial gains on distributed memory systems where global synchronizations can be extremely costly, or load imbalances caused by a poor choice of CPU configurations or the computational aspects of the model significantly impact the overall runtime. The impact of load imbalances will be expanded upon in later sections.

2.4 A Ghosting Performance Example

To benchmark the performance of the various paradigms implemented within Mod_Comm a simple communication epoch which performs a north/south ghosting operation on two 3-dimensional arrays is iterated 500 times to produce an average communication rate. The size of the global domain is gradually increased up to the resolution of a typical numerical weather prediction run of 0.5x0.625 degree horizontal resolution with 55 vertical levels. Results are included for four platforms (Table 1): Columbia, an SGI Altix 3700 located at the NASA Advanced Supercomputing Division (NAS) at the NASA Ames Research Center; Daley, an SGI Origin 3000 located at the NASA

Center for Computational Sciences (NCCS) at Goddard Space Flight Center (GSFC); Halem, a Compaq AlphaServerSC located at NCCS; and Eagle, an IBM RS/6000 SP located at the Center for Computational Sciences (CCS) at Oak Ridge National Laboratory (ORNL). Columbia is an SGI Altix 3700 cluster. The Columbia cluster consists of 20 nodes each with 512 processors providing global shared memory within a node via the SGI non-uniform memory access (NUMAlink) interconnect. The SGI Message-Passing Toolkit (MPT) version 1.9.1.0 is used to provide the standard message-passing libraries on Columbia throughout this article. Daley is a 512 processor SGI Origin NUMA system; the system runs a single IRIX operating system and memory is distributed over the processors but shows a single memory image to the user. The SGI MPT version 1.4.0.3 is used to provide the standard message-passing libraries on Daley throughout this article. Halem and Eagle are both distibuted memory systems with 4-way symetric multiprocessing (SMP) interconnected nodes. Halem's SMP nodes are connected to the single rail Quadrics QsNet network switch through an ELAN PCI adaptor. Eagle's SMP nodes are connected over a scalable parallel (SP) switch using the user space protocol in non-shared mode. Each system is benchmarked with a 32 processor configuration using 8 SPMD processes and 4 OpenMP threads.

The cost of communication varies greatly across the sample platforms, and can be significantly impacted by the choice of data communication paradigm. The communication rates (Fig. 1) for the sample ghosting vary the most on the SGI Origin. The SGI Origin provides the largest choice of implementations allowing us to test all four data communication paradigms within Mod_Comm (MPI-1, MPI-2, SHMEM, and MLP). The observed rate for the MPI-2 implementation on the SGI Origin is greatest at about 1400 MB per second. This is a huge improvement over the MPI-1 implementation which levels off at about 350 MB per second. A closer look at the speedup (T_m/T_{MPI-2}, where T_m is the time for MPI-1, SHMEM, or MLP, and T_{MPI-2} is the time for MPI-2) in the actual times for the MPI-2 implementation on the SGI Origin (Fig. 2) show the MPI-2 implementation is 3.8 times faster than MPI-1, 2.9 times faster than SHMEM and 1.5 times faster than MLP. The improvements over MPI-1 were expected due to the one-sided operations and multi-threading of the communication, however similar features are used with SHMEM and MLP. A possible reason for the poor performance of the SHMEM and MLP implementations are their use of global synchronization routines where MPI-2 allows special assertions to be made that can decrease the cost of synchronization.

The communication performance on the IBM and Compaq do not vary as much as the SGI Origin. The Compaq produces nearly equal rates for MPI-1 and MPI-2 and only very slight improvements with SHMEM. The same is true for the IBM, although MPI-1 is slightly better than MPI-2. The barrier synchronization assertions mentioned on the SGI Origin have not been implemented on either of these systems; this may account for the nearly identical

performance of MPI-1 and MPI-2 on these systems. On the SGI Altix, the MPI-1 performance is significantly better for small message sizes (5mb), after this peak the performance is comparable for both MPI-1 and MPI-2. In addition, none of these systems were able to handle the multi-threaded features of MPI-2.

The downfall of MPI-2 on the Compaq and IBM appears to be that the `WIN_FENCE` operation on these systems is a strong synchronization function (essentially a global barrier synchronization), all processes must wait at the fence until every MPI process has reached that point (this is not the case on the SGI Origin). Thus, many processes are held up much longer than needed during ghosting exchanges when really they just need to wait on their nearest neighbors. MPI-1, as implemented in Mod_Comm, is entirely non-blocking and based on a weak synchronization technique, so each process just waits for it's nearest neighbor data, and then resumes. This is amplified by fabricating a load imbalance during the overlapped computation between send and receive calls for the ghosting example. A 5 second pause is passed from process 0 to process 7 at each iteration for a total of 16 iterations, during each iteration only one process waits in the fabricated load imbalance. Vampir [12], a message-passing performance visualization tool, is used to visualize the impact of the load imbalance using MPI-1 and MPI-2 on the Compaq (Figs. 3 and 4). For MPI-1, where processes only wait on their neighbors the total execution time is 11.976 seconds, 10 seconds for each processes two 5 second pauses, plus the communication costs. With MPI-2, since each process needs to wait for the slowest process, the total execution time is 81.427 seconds, a considerable impact due to the strong nature of the `WIN_FENCE` synchronization. Therefore any gain from the one-sided communication is lost in synchronization overhead.

3 The NASA fvGCM Performance

3.1 0.5 Degree NWP Experiments

A typical 7-day NWP forecast is benchmarked on the three test systems using all available data communication paradigms. The NWP run is performed at 0.5x0.625 degree horizontal resolution with 32 vertical levels, a total of 6,653,952 grid points. Performance numbers are provided for the test systems using the best performing data communication paradigm and CPU configuration for each system. For both the Compaq and IBM we are restricted to 4 CPUs per node and thus all runs use 4 OpenMP threads within a node. The SGI systems do not restrict the number of OpenMP threads we can use, therefore optimal configurations were chosen based on the model resolution. It is noted that optimal OpenMP performance is obtained on the SGI Altix

when restricting to 4 threads, thus all configurations for this system use only 4 OpenMP threads per process. Overall performance is described in terms of model throughput (Fig. 5), defined as the total number of forecasted days possible per wall clock day (Days/day).

Due to the varying processor speeds on these systems, a direct comparison of throughput is not possible. The throughput is proportional to the system processor speed. The throughput does provide a measure of overall computing capability with the NASA fvGCM. Using 240 CPUs of Columbia, the NASA fvGCM completes more than 900 simulated days per wall-clock day. This equates to about 10 minutes to complete a 7-day NWP forecast with 240 CPUs, well within the range of acceptability for a production NWP forecast where model updates every 6 hours are required. On the Compaq, the NASA fvGCM is capable of about 1 simulation year per day using 120 CPUs at the NWP resolution, and 521 Days/day using 288 CPUs; roughly 30 minutes of wall-clock time to complete a 7-day NWP forecast with 120 CPUs, again within an acceptable range for production NWP forecasts. The throughput is cut almost in half on the SGI Origin and IBM, due to the slower processor speeds.

Speedup and efficiency can be used to compare the scalability of the NASA fvGCM across the three systems (Fig. 6). Speedup (Eq. 1) and efficiency (Eq. 2) are defined as:

$$S_n = T_{32}/T_n \tag{1}$$

$$E_n = S_n/(n/32) \tag{2}$$

where T_n is the total time using n CPUs and T_{32} is the total time using 32 CPUs. This equation for speedup is modified from the standard definition in which T_1 is used instead of T_{32} due to the large problem size of this benchmark experiment. The time and memory required to complete the NWP simulation using 1 CPU is unreasonable for this problem.

For processor counts less than 120 CPUs the Compaq and SGI speedups are very similar; each of these systems outperform the IBM. The SGI Origin scales much better than the SGI Altix and Compaq as we move to larger CPU configurations, where the SGI Origin continues to improve reaching a speedup of 6.51 at 304 CPUs versus a speedup of only 4.17 for the Compaq at 288 CPUs and 4.49 for the SGI Altix for 360 CPUs. In terms of efficiency, the SGI Origin remains better than 80 percent up to 200 CPUs, and about 70 percent for 300 CPUs. The SGI Altix and Compaq efficiency drop off dramatically to less than 60 percent at 200 CPUs and about 46 percent at 300 CPUs. This is due in-part to the ability to chose ideal CPU configurations based on the domain resolution. Choosing an optimal domain decomposition

can avoid computational load imbalances that may impact the overall model performance. In addition, the superior communication performance of the SGI Origin, as shown in section 2, effectively reduces the percent of the total time spent in communication on the SGI Origin (Fig. 7), where at 128 CPUs the SGI Origin run spends 24 percent of the total execution time in communication opertations and barriers as opposed to 35 and 42 percent on the SGI Altix and Compaq with 120 CPUs. As processors are added, the percent of time spent in communication increases rapidly on the Compaq, reaching nearly 60 percent as the total CPUs approach 300; the increase on the SGI Altix is less severe. This increase is due to growing message volume as we are forced to use 72 SPMD processors due to the node structure of the Compaq and OpenMP performance on the SGI Altix. On the SGI Origin we are able to efficiently use more OpenMP threads to scale up to 300 CPUs and beyond, and keep the impact on total communication time low, only 29 percent for 304 total CPUs (19 MPI processes and 16 OpenMP threads). In addition, the 16 OpenMP threads for this configuration on the SGI also participate in the data communication further improving the communication time.

The difference in model throughput with different data communication paradigms is most dramatic on the SGI Origin. MPI-2 outperforms all other paradigms on the SGI Origin, where as on the SGI Altix, IBM, and Compaq MPI-1 is still the top performer. MPI-1 is 10 percent faster then MPI-2 on the IBM, and less than 5 percent faster then MPI-2 on the Compaq. This is further support for providing a general communication module such as Mod_Comm that allows easy swapping among data communication paradigms. On the SGI Origin, the MPI-2 performance is significantly greater than MPI-1. Overall model performance is more than 20 percent faster with MPI-2, and the communication time speeds up by about 90 percent as the CPU count increases (Fig. 8) indicating that the MPI-2 communication scales better than MPI-1. MPI-2 outperforms SHMEM by 10 percent and MLP by 6 percent, in terms of total model throughput.

3.2 Lower Resolution Climate Runs

Climate simulations are typically low resolution long-term runs spanning decadal to century time-scales. For the climate benchmarks, we have examined the 2x2.5 degree horizontal resolution with 55 vertical levels. The NASA fvGCM is capable of simulating more than 1 decade per day using only 92 CPUs on the Compaq (Fig. 9). This level of performance provides a huge scientific capability for climate researchers allowing 1000 year simulations to complete in about 3 months, and inter-decadal studies to complete in a matter of days. The throughput across platforms again mirrors the processor speeds on each system, and the speedup and communication efficiency results are consistant

with the NWP benchmarks.

4 Summary

The optimal choice of data communication paradigm can vary substantially across computational platforms. The development of Mod_Comm, a portable communication module for the NASA fvGCM, combined with a fast numerical algorithm and a hybrid parallelism approach allow the NASA fvGCM to maintain scalable performance across a variety of computational platforms. The one-sided communication features of MPI-2, along with a unique multithreading implementation within Mod_Comm produce significant improvements over the standard MPI-1 communication operations on SGI systems. The MPI-2 implementations on other systems (SGI Altix, IBM, and Compaq) currently do not perform up to expected levels, however, the flexibilty of Mod_Comm permits the NASA fvGCM to use the most efficient communication paradigm for each platform and maintain sufficent scalability across platforms. The NASA fvGCM is capable of producing 7-day NWP forecasts at 55 km horizontal resolution in about 10 minutes using 240 CPUs of the SGI Altix, and 30 minutes using 120 CPUs on the Compaq AlphaServerSC. Climate simulations at 200 km horizontal resolution are capable of completing more than a decade in a single wall clock day. This computational throughput exceeds the current demands of the climate and weather prediction community.

5 Acknowledgment

The authors would like to thank Dr. Robert Atlas for his support during the development and evaluation of the finite-volume general circulation model.

References

[1] S.-J. Lin, A "vertically lagrangian" finite-volume dynamical core for global models, Submitted to Mon. Wea. Rev. .

[2] S.-J. Lin, R. Rood, A flux-form semi-lagrangian general circulation model with a lagrangian control-volume vertical coordinate, The Rossby-100 symposium, 1998.

[3] S.-J. Lin, A finite-volume integration method for computing pressure gradient forces in general vertical coordinates, Q. J. Roy. Met. Soc. 123 (1997) 1749–1762.

[4] S.-J. Lin, R. Rood, An explicit flux-form semi-lagrangian shallow water model on the sphere, Q. J. Roy. Met. Soc. 123 (1997) 2477–2498.

[5] S.-J. Lin, R. Rood, Multidimensional flux form semi-lagrangian transport schemes, Mon. Wea. Rev. 124 (1996) 2046–2070.

[6] J. Kiehl, J. Hack, G. Bonan, B. Boville, B. Briegleb, D. Williamson, P. Rasch, Description of the NCAR community climate model (CCM3), Tech. Rep. NCAR/TN-420+STR, Boulder, CO (1996).

[7] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, The MIT Press, Cambridge, Massachusetts, 1999.

[8] W. Gropp, E. Lusk, R. Thakur, Using MPI-2: Advanced Features of the Message-Passing Interface, The MIT Press, Cambridge, Massachusetts, 1999.

[9] Quadrics, SHMEM Programming Manual, Quadrics Supercomputers World Ltd., QSW Limited One Bridewell Street Bristol, United Kingdom, 2001.

[10] J. Taft, Multi-level parallelism, a simple highly scalable approach to parallelism for CFD, HPCCP/CAS Workshop 98 Preceedings, 1998.

[11] B. V. Protopopov, A multithreaded message passing interface (MPI) architecture: Performance and program issues, Journal of Parallel and Distributed Computing 61 (2001) 449–466.

[12] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, K. Solchenbach, VAMPIR: Visualization and analysis of MPI resources, Supercomputer XII (1) (1996) 69–80.

Computer	Manufacturer/Model	Processor (Speed)	Total CPUs (Nodes)
Columbia	SGI Altix 3700	Intel Itanium-2 (1.5 GHz)	10,240 (20)
Halem	Compaq AlphaServer SC45	Alpha-EV68 (1.25 GHz)	1388 (347)
Daley	SGI Origin 3000	R14000 (0.5 GHz)	512 (1)
Eagle	IBM RS/6000 SP	Power3-II (0.375 GHz)	704 (176)

Table 1
Description of the computational platforms used in testing communication and the overall model performance of the NASA fvGCM.

Fig. 1 The communication rates for 2 ghosting events with Mod_Comm using 8 SPMD processes and 4 OpenMP threads on SGI, Compaq and IBM systems.

Fig. 2 The communication times and speedup with MPI-2 for 2 ghosting events with Mod_Comm on the SGI Origin 3000.

Fig. 3 Vampir timeline for the MPI-1 execution of a fabricated load imbalance test on the Compaq AlphaSeverSC. Areas of red or orange represent time spent in communication or synchronization operations, areas of green represent time spent in the fabricated computation.

Fig. 4 Same as Fig. 3 except MPI-2 is used for data communication.

Fig. 5 Throughput, forecasted days per wall clock day, of the NASA fvGCM for a typical NWP forecast at 0.5x0.625 degree horizontal resolution with 32 vertical levels.

Fig. 6 Speedup, T_{32}/T_n, of the NASA fvGCM for a typical NWP forecast at 0.5x0.625 degree horizontal resolution with 32 vertical levels.

Fig. 7 Percent of the total execution time spent in communication for a typical NWP forecast with the NASA fvGCM at 0.5x0.625 degree horizontal resolution with 32 vertical levels.

Fig. 8 The total communication time in seconds for the NWP test case on the SGI Origin 3000 using MPI-2 and MPI-1, and the effective speedup of MPI-2 over MPI-1.

Fig. 9 The throughput (modeled days per wall clock day) of the NASA fvGCM for a 2x2.5 degree 55 level climate simulation on the SGI Origin 3000, Compaq AlphaServerSC, and IBM RS6000/SP.

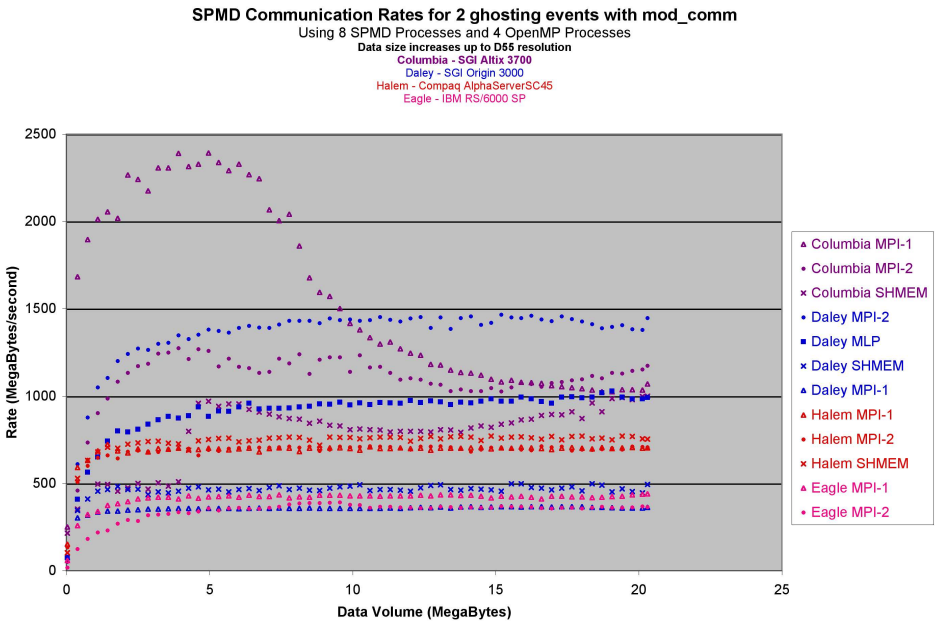
SPMD Communication Rates for 2 ghosting events with mod_comm
Using 8 SPMD Processes and 4 OpenMP Processes
Data size increases up to D55 resolution
Columbia - SGI Altix 3700
Daley - SGI Origin 3000
Halem - Compaq AlphaServerSC45
Eagle - IBM RS/6000 SP
Rate (MegaBytes/second)
2500
2000
1500
1000
500
0
0
5
10
15
20
25
Data Volume (MegaBytes)
Columbia MPI-1
Columbia MPI-2
Columbia SHMEM
Daley MPI-2
Daley MLP
Daley SHMEM
Daley MPI-1
Halem MPI-1
Halem MPI-2
Halem SHMEM
Eagle MPI-1
Eagle MPI-2

Fig. 1.

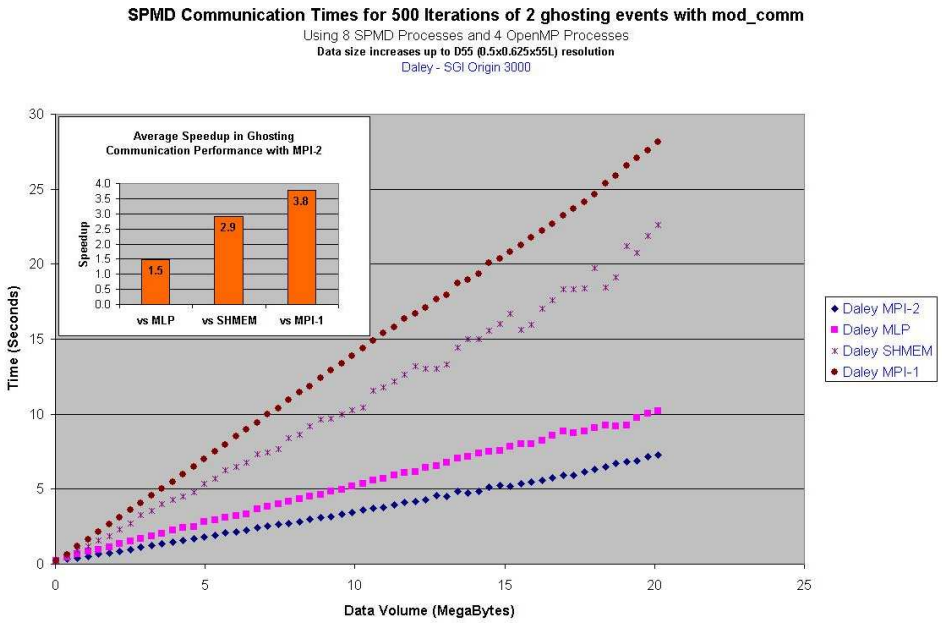
SPMD Communication Times for 500 Iterations of 2 ghosting events with mod_comm
Using 8 SPMD Processes and 4 OpenMP Processes
Data size increases up to D55 (0.5x0.625x55L) resolution
Daley - SGI Origin 3000
Average Speedup in Ghosting Communication Performance with MPI-2
Speedup
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.5
2.9
3.8
vs MLP
vs SHMEM
vs MPI-1
Time (Seconds)
30
25
20
15
10
5
0
0
5
10
15
20
25
Data Volume (MegaBytes)
Daley MPI-2
Daley MLP
Daley SHMEM
Daley MPI-1

Fig. 2.

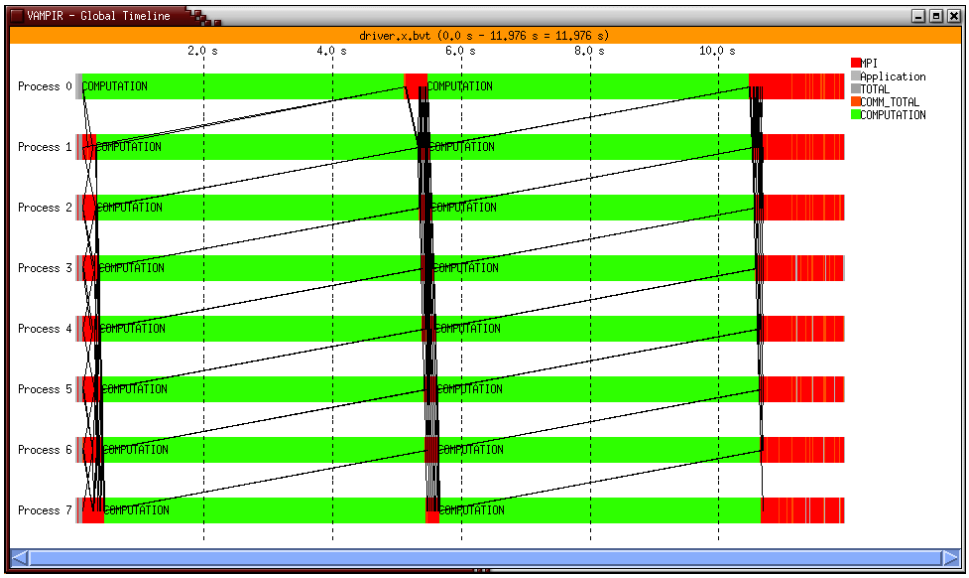
VAMPIR - Global Timeline
driver.x.bvt (0.0 s - 11.976 s = 11.976 s)
2.0 s
4.0 s
6.0 s
8.0 s
10.0 s
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
COMPUTATION
MPI
Application
TOTAL
COMM_TOTAL
COMPUTATION

Fig. 3.

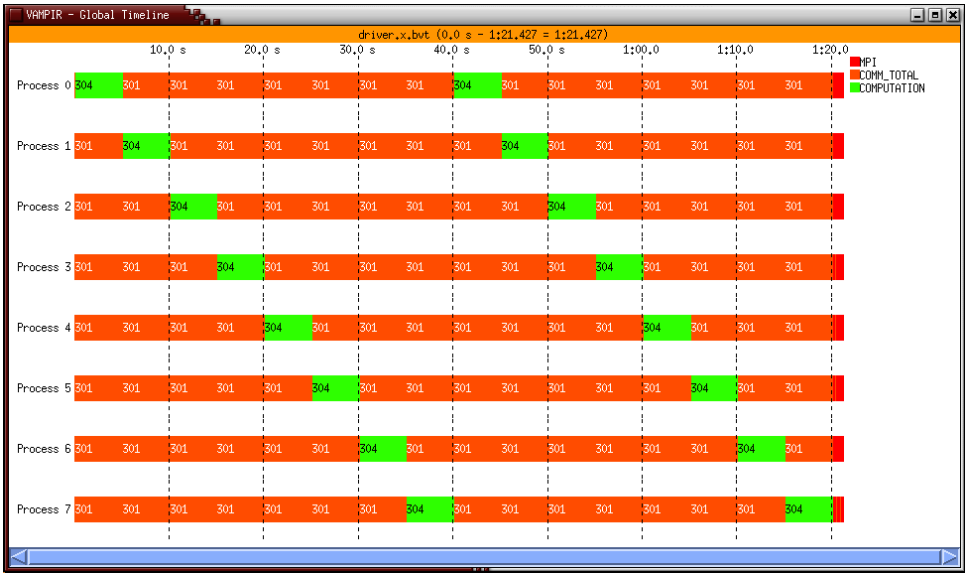
VAMPIR - Global Timeline
driver.x.bvt (0.0 s - 1:21.427 = 1:21.427)
10.0 s
20.0 s
30.0 s
40.0 s
50.0 s
1:00.0
1:10.0
1:20.0
MPI
COMM_TOTAL
COMPUTATION
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7

Fig. 4.

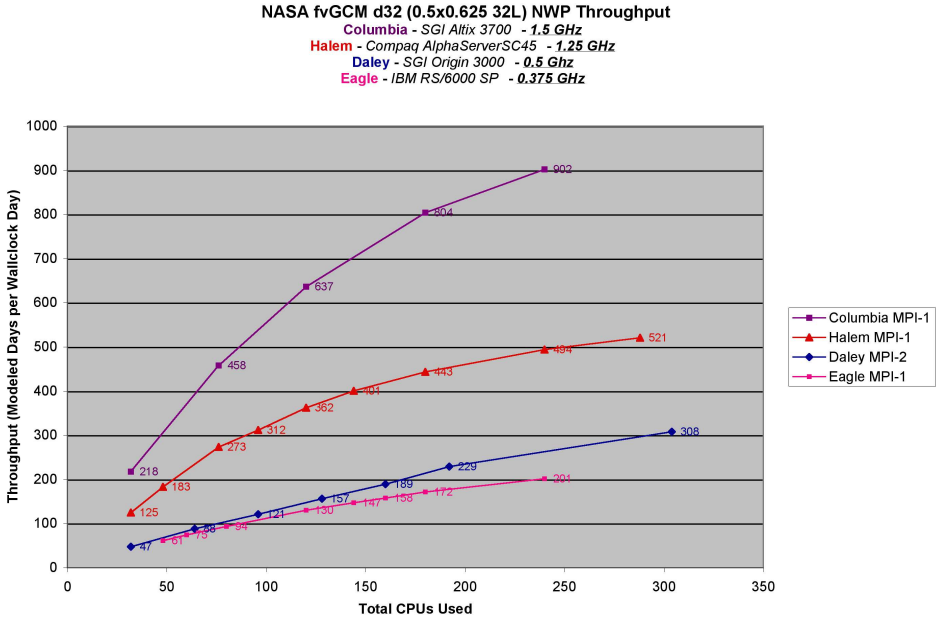
NASA fvGCM d32 (0.5x0.625 32L) NWP Throughput
Columbia - SGI Altix 3700 - 1.5 GHz
Halem - Compaq AlphaServerSC45 - 1.25 GHz
Daley - SGI Origin 3000 - 0.5 Ghz
Eagle - IBM RS/6000 SP - 0.375 GHz
Throughput (Modeled Days per Wallclock Day)
Total CPUs Used
Columbia MPI-1
Halem MPI-1
Daley MPI-2
Eagle MPI-1

Fig. 5.

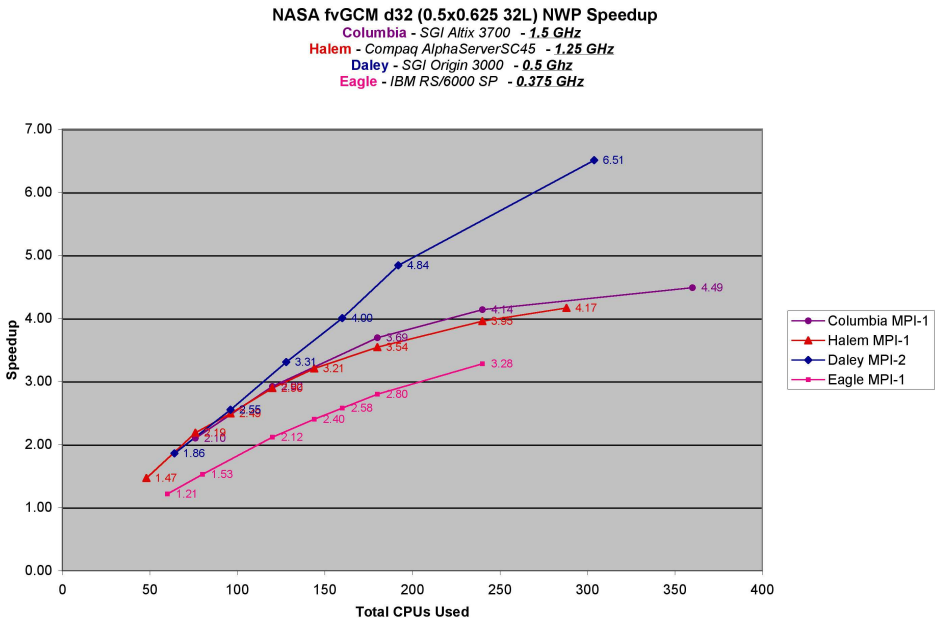
NASA fvGCM d32 (0.5x0.625 32L) NWP Speedup
Columbia - SGI Altix 3700 - 1.5 GHz
Halem - Compaq AlphaServerSC45 - 1.25 GHz
Daley - SGI Origin 3000 - 0.5 Ghz
Eagle - IBM RS/6000 SP - 0.375 GHz
Speedup
Total CPUs Used
Columbia MPI-1
Halem MPI-1
Daley MPI-2
Eagle MPI-1

Fig. 6.

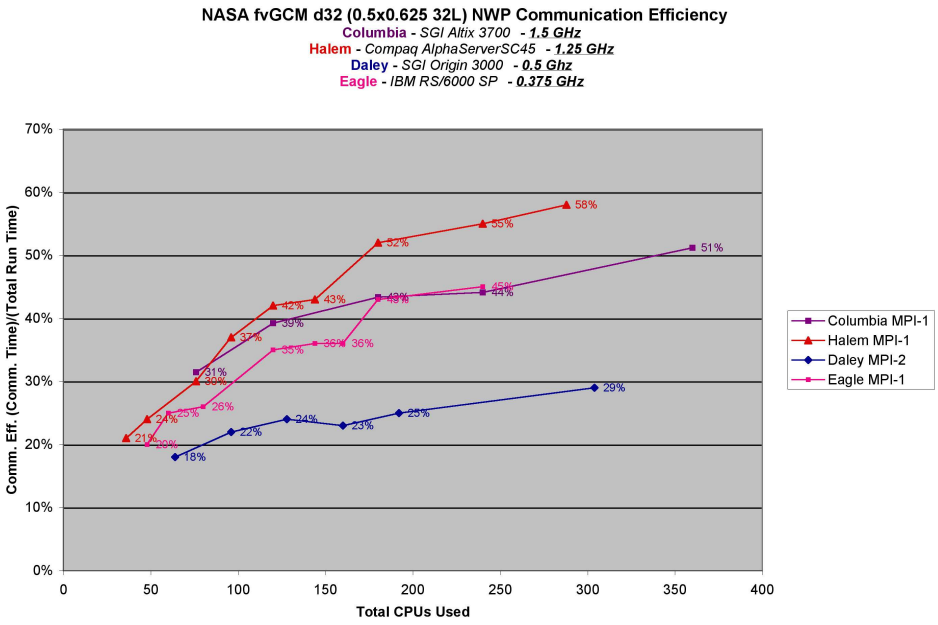
NASA fvGCM d32 (0.5x0.625 32L) NWP Communication Efficiency
Columbia - SGI Altix 3700 - 1.5 GHz
Halem - Compaq AlphaServerSC45 - 1.25 GHz
Daley - SGI Origin 3000 - 0.5 Ghz
Eagle - IBM RS/6000 SP - 0.375 GHz
Comm. Eff. (Comm. Time)/(Total Run Time)
Total CPUs Used
Columbia MPI-1
Halem MPI-1
Daley MPI-2
Eagle MPI-1

Fig. 7.

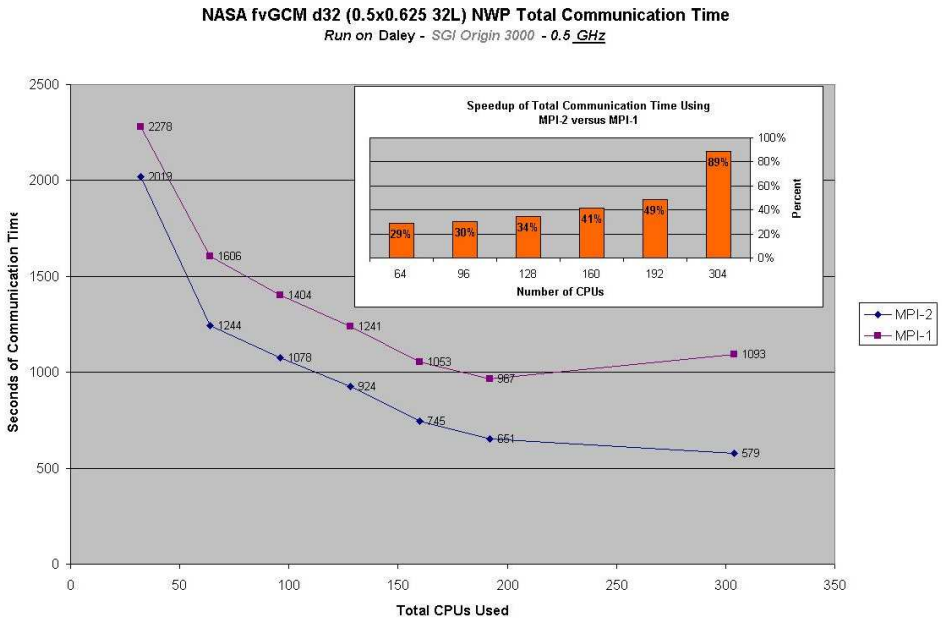
NASA fvGCM d32 (0.5x0.625 32L) NWP Total Communication Time
Run on Daley - SGI Origin 3000 - 0.5 GHz
Seconds of Communication Time
Total CPUs Used
Speedup of Total Communication Time Using MPI-2 versus MPI-1
Percent
Number of CPUs
MPI-2
MPI-1
2278
2019
1606
1404
1244
1241
1078
1053
967
924
1093
745
651
579
29%
30%
34%
41%
49%
89%

Fig. 8.

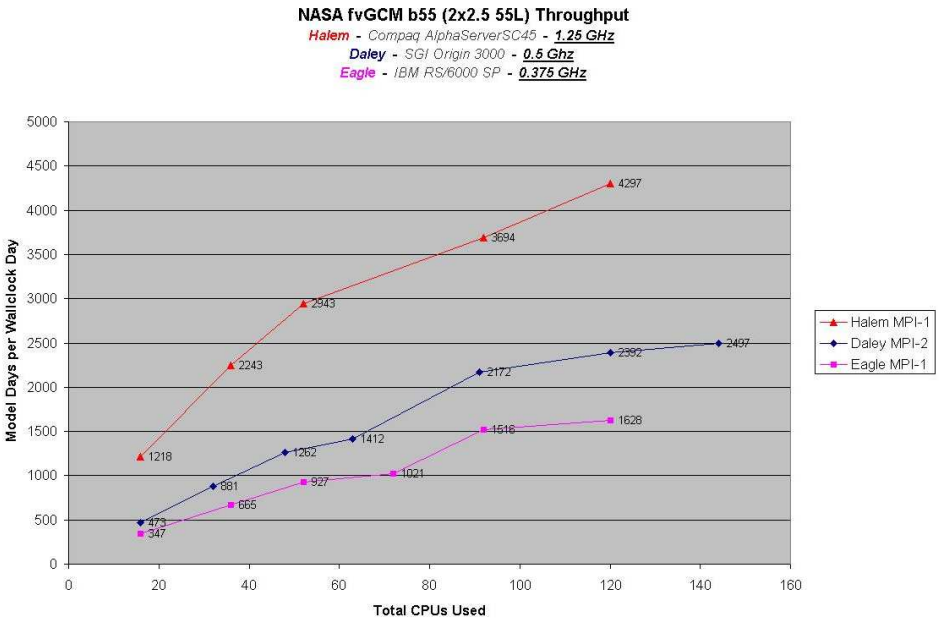
NASA fvGCM b55 (2x2.5 55L) Throughput
Halem - Compaq AlphaServerSC45 - 1.25 GHz
Daley - SGI Origin 3000 - 0.5 Ghz
Eagle - IBM RS/6000 SP - 0.375 GHz
Model Days per Wallclock Day
Total CPUs Used
Halem MPI-1
Daley MPI-2
Eagle MPI-1
4297
3694
2543
2243
1218
2497
2392
2172
1412
1262
881
473
1628
1516
1021
927
665
347

Fig. 9.