

A Methodology for Developing High Fidelity Communication Models for Large-scale Applications Targeted on Multicore Systems

Charles W. Lively and Valerie E. Taylor
Texas A&M University, College Station, TX
{clively,taylor}@cs.tamu.edu

Sadaf R. Alam and Jeffrey S. Vetter
Oak Ridge National Laboratory, Oak Ridge, TN
{alamsr,vetter}@ornl.gov

Abstract

Resource sharing and implementation of software stack for emerging multicore processors introduce performance and scaling challenges for large-scale scientific applications, particularly on systems with thousands of processing elements. Traditional performance optimization, tuning and modeling techniques that rely on uniform representation of computation and communication requirements are only partially useful due to the complexity of applications and underlying systems and software architecture. In this paper, we propose a workload modeling methodology that allows application developers to capture and represent hierarchical decomposition and distribution of their applications thereby allowing them to explore and identify optimal mapping of a workload on a target system. We demonstrate the proposed methodology on a Teraflops-scale fusion application that is developed using message-passing (MPI) programming paradigm. Using our analysis and projection results, we obtain insight into the performance characteristics of the application on a quad-core system and also identify optimal mapping on a Teraflops-scale platform.

1. Introduction

Parallel computing systems are showing a significant trend towards the use of chip multiprocessors (multicore). Systems composed of multicore processors represent a strong dichotomy from the traditional computing system configuration. As multicore systems become more prevalent it remains to be seen how large-scale scientific applications will perform and scale efficiently on these systems. Specifically, it remains to be understood what components of these emerging systems will influence the achievable performance of large-scale scientific applications.

Workload characterization and performance modeling provides for a possible solution to understanding how applications will perform on these multicore systems. There are many modeling techniques available for the analysis of large-scale scientific applications [1][3][4][10][16][17]. In each of the modeling techniques used a specific goal or measurement can be seen as the main motivation for the modeling process, which in turn determines strengths as well as shortcomings of a given approach. In order for scientific applications to perform efficiently on multicore systems, a modeling approach is needed that provides insight into an application's code structure, algorithmic characteristics, scaling parameters, and communication requirements. Moreover, flexibility and extensibility are critical requirements for a modeling scheme such that it would enable both application and system developers to project workload requirements and understand workload sensitivity for future system and problem configurations.

In this paper, we propose a workload modeling methodology that allows application developers to capture and represent hierarchical decomposition and distribution of their applications thereby allowing them to explore and identify optimal mapping of a workload on a target system. We introduce a method to symbolically model the message-passing (MPI) communication requirements and code hierarchy of large-scale scientific applications. The workload requirements of scientific applications can be seen as extensions of several key input parameters within the application. Varying these parameters causes changes in the performance of the application by increasing message volume for communication, altering the number of time steps for nested loop operations, and changing the number of floating point calculations being done within multiple nested loops. Furthermore, such changes are often directly related to a select number of key input parameters. We also present a method for understanding the MPI collective

communication involved in scientific applications as these operations relate to the application hierarchy and are often considered as the critical scaling bottlenecks.

We demonstrate the proposed methodology on a Teraflops-scale fusion application, GYRO, that is developed using the message-passing (MPI) programming paradigm [13]. GYRO, an advanced Eulerian Gyrokinetic-Maxwell equation solver [5], is a fusion application that has scaled to thousands of processing nodes on high-end supercomputing platforms including the Cray XT and IBM Blue Gene systems [6][8] and is studied by many scientists in the high-performance computing community [15]. Within the GYRO application, multiple communicators are used for decomposition of tasks and several costly collective operations, such as MPI_Allreduce and MPI_Alltoall. The symbolic models generated in this work are used to gain a detailed understanding of GYRO's performance on a quad-core system and future projections of the application's performance requirements. This allows for a sensitivity analysis of the workload requirements for larger system configurations and future problem sizes for the application. For example, we studied the growth rate of message sizes for MPI_Allreduce and MPI_Alltoall with respect to input parameters used for those operations. The contribution of the work presented in this paper is not only a method to model communication hierarchies of large-scale applications but also a framework to support systematic model generation and validation for high-end applications written in Fortran or C/C++ with MPI programming paradigm.

The remainder of this paper is organized in the following manner. Section 2 describes several motivating examples that have led to the development of this work and provides background to the targeted application, platforms, and modeling infrastructure. Section 3 presents the modeling methodology followed by sensitivity analysis of the workload requirements of the GYRO application in section 4. Section 5 concludes this paper and discusses future work.

2. Motivation and Background

Achieving high performance efficiencies on existing Teraflops-scale supercomputing systems has become a chief motivator in the refinement of performance modeling methodologies. The recent rise in hierarchical architectures, such as those utilizing chip multiprocessors (multicores) has further exacerbated current performance results as these systems offer parallelism at multiple scales: within core, within chip, across chips, within a compute node, and

across the compute nodes. A new method is needed to fully decompose an application's requirements and project those parameters for future systems, such as petascale compute systems. Particularly, the process of tuning and optimization has become extremely challenging on emerging multi-core based supercomputing systems. We present preliminary results and analysis for two large-scale applications on a contemporary quad-core based massively parallel computing platform.

2.1 High Performance Computing Platforms

Emerging compute systems that utilize multicore processors are configured hierarchically with multiple processors within a node. These systems also utilize various levels of sharing for memory subsystems. The quad-core AMD opteron processor is used in the Jaguar Cray XT4 system, available at Oak Ridge National Laboratory (ORNL)[6]. Each core on the processor has an individual 512KB L2 cache and a 2MB L3 cache is shared amongst all four cores. The Jaguar system makes use of the Cray SeaStar 3D torus network that is design to provide reliability and a very high bandwidth.

Bassi is a distributed, shared memory compute system that is available at the National Energy Research Scientific Computing Center (NERSC) [14]. Bassi utilizes the IBM POWER5 chip. The system configuration has varying levels of hierarchy because of the use of a dual core module (DCM). The POWER5 chip is a dual-core processor that shares L2 and L3 caches between both processors on chip. Bassi has eight processors per node.

Jaguar and Bassi are representatives of a strong dichotomy from previous computing systems. Within these systems, there are various levels of resource sharing involved. As systems become more complex in composition the tools that are used for analyzing the performance of such applications must evolve as well.

2.2. Large-scale Scientific Applications

Computer simulations are now considered as a critical component of scientific discovery along with theoretical and experimental studies [15]. A number of these breakthrough simulations require massive computing power that has only recently become available. With the availability of hundreds of Teraflops-scale platforms like Blue Gene and Cray XT systems, the leading issue is now scaling these applications to tens of thousands of processing core. A number of these applications are written using the

message-passing (MPI) programming environment, however, the application developers are experimenting with different hierarchical decompositions schemes to scale applications on thousands of multicore-based processing elements.

In this study, we focus on a plasma fusion application called GYRO. GYRO is an advanced Eulerian gyrokinetic-Maxwell equation solver and employs a hierarchical decomposition scheme using sub-communicators [11]. The application is capable of facilitating a better understanding of plasma micro-instabilities and turbulence flow in the tokamak geometry. The code ports to a variety of systems and can scale to several thousand processors [12].

Our target application has shown sensitivity to communication infrastructure in earlier studies [2]. Modeling efforts of GYRO have led to a more detailed understanding of the communication characteristics of this communication intensive application [7]. Specifically, the performance of GYRO on multicore platforms has been evaluated to a great degree. The collective communications of MPI_Allreduce and MPI_Alltoall in GYRO cause performance bottlenecks within the application. On multicore systems, using the smallest number of processors or cores within a node will give the best performance for several scientific applications. However, the effects of using multiple communicators within the application for performing costly collective operations remains to be understood and represented in current or previous modeling efforts.

2.3. Preliminary Results

Figure 1 shows the scaling characteristics of a molecular dynamics simulation on an AMD quad-core based massively parallel system.¹ The scaled version of the rhodopsin benchmark was used for these simulation experiments.² The experiments are run utilizing one core per node (N1), two cores (N2) and all four cores (N4).

We compare this with ideal scaling, which is calculated by dividing runtimes by a factor of 2. We note that performance of the code is consistent for small number of MPI tasks (<1024) where computation-to-communication ratio is higher. As the MPI message volume grows with increases in the number of MPI tasks, we observe an increased slowdown in performance where all four cores compete for the resources.

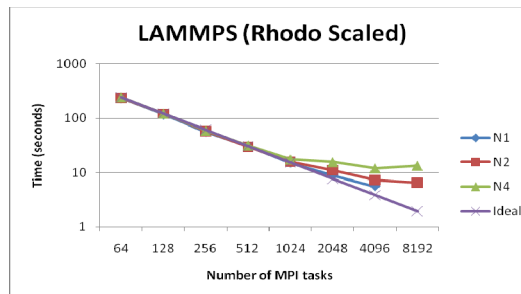


Figure 1: Parallel efficiency on a massively-parallel system with quad-core processor

Initial experimental results on GYRO have been conducted for B1-std from 16 to 1024 processors. The B1-std benchmark uses a grid resolution size of 16 x 140 x 8 x 8 x 20 and closely represents a typical production run for GYRO. At each processor configuration utilizing all four cores on the XT4 gives the worst performance. Utilizing half of the cores gives approximately 15-20% better performance. Utilizing one core the application performs more than 20-25% better than using all four cores available.

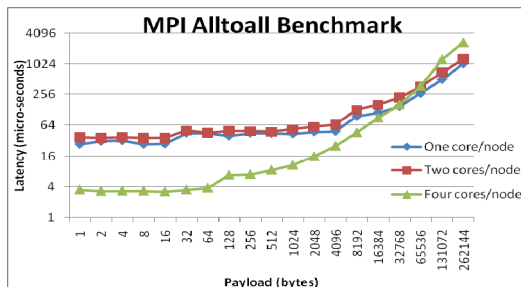


Figure 2: Performance sensitivity to on-socket vs. off-socket operations

In order to understand MPI-only performance characteristics, we ran Intel MPI benchmarks on the target platform [9]. Results are shown for the MPI_Alltoall collective operation in Figure 2 for four MPI tasks. We note that when all tasks are on a single node, we have a much lower MPI latency. In other words, the MPI is aware of the locality of the task. As we move tasks across the nodes, the latencies grow. For very large messages (> 32Kbytes), the effect of this locality is minimum. Hence, if the tasks exchanging a large number of small messages are placed on a single core, there could be a performance improvement of a factor of 10 or more. In summary, the application should be made aware of the system hierarchy so that optimal mappings can be identified. For large-scale scientific applications, this is a phenomenal challenge since there is no mechanism to identify and expose application communication

¹ <http://lammps.sandia.gov/>

² <http://lammps.sandia.gov/bench.html>

characteristics and hierarchy. This is one of the primary motivations for our proposed modeling scheme for multicore based supercomputing systems.

2.4. Related Work

Researchers have used performance modeling to understand the performance of applications on various systems. Almasi et. al use performance models to study the computational and communication kernels of a protein folding application [3]. Their study led to performance predictions for the application across a variety of computing systems including an IBM Blue Gene/C.

In their work, Wu et. al use full-scale models of large-scale scientific applications for two codes, GTC and LBM [18]. The models are used to understand how application kernel's can be improved on cluster systems with multicore processors. The performance results presented led to a hybrid approach for optimizing collective communication on cluster systems with chip multiprocessors.

Our unique contribution is the introduction of a hierarchical modeling scheme within a modeling framework that enables application developers to capture code hierarchy and system developers to explore optimal design configurations for complex applications efficiently.

3. Methodology

In this section we outline the model creation process for the GYRO application. Symbolic parameters and hierarchical models are presented for the application. For the rest of this work, results presented equate n cores or n processors to n MPI tasks.

3.1. Hierarchical Modeling

The methodology presented in this work addresses an issue of sharing communication both between nodes in a compute system and within nodes of a compute system. The dichotomy presented in multicore compute systems from systems consisting of uniprocessors must be addressed in the hierarchical modeling of the application. The collective communication that occurs on multicore systems requires an understanding of the processes involved. Traditional modeling techniques focus on a methodology that assumes MPI_COMM_WORLD is acting as the communicator for many scientific applications. As seen in GYRO, different

communicators can be utilized to achieve collective communication within an application. Our approach provides a hierarchical understanding of the collective communication characteristics of an application on multicore architectures. We not only encapsulate the message volume generated by a communicator but also retain the communicator identities and sizes.

In order to provide the modeling capability that other applications can exploit, we extend a framework that is used for modeling parallel scientific applications. Modeling Assertions (MA) framework [1] is used to facilitate the creation of symbolic performance models by combining analytic and empirical modeling approaches into an incremental model validation tool. The framework allows for the user to instrument their code with appropriate MA calls to thoroughly encapsulate application requirements.

3.2. MPI Collective Operations

MPI Collective communication plays an important role in large-scale scientific applications. Many applications make use of multiple MPI communicators, or sub-communicators, and MPI collective operations, such as MPI_Allreduce, MPI_Alltoall, and MPI_Alltoallv. When combined with MPI collective operations, sub-communicators perform the respective collective operation and exchange the data with other sub-communicators or with MPI_COMM_WORLD.

The MA framework has been extended to encapsulate the application requirements with respect to sub-communicators and how the sizes of these communicators will affect the message volume of MPI collective communications. The *ma_mpi_comm_split* assertion is used to encapsulate new communicator sizes based off of an MPI_Comm_split operation. The *ma_mpi_comm_split* assertion provides for symbolic representations of the original communicator size and the new communicator size that will be formed.

To further encapsulate the communication requirements of an application an understanding of the interaction between the collective communication and the communicator used for the operation is needed. In many scientific applications collective communication is achieved through MPI_Allreduce and MPI_Alltoall operations. In this paper, we will illustrate the use of two newly introduced MPI collective assertions, *ma_mpi_allreduce* and *ma_mpi_alltoall* to model message volume and the number of MPI tasks. These provide for symbolic representations of the message volume being transmitted by the MPI_Allreduce and MPI_Alltoall operations in an application.

These calls allow for symbolic representation of the message sizes for the collective operations with respect

to application parameters. The assertions also provide a tag for the communicator that is used for the operation so that the number of processes involved is included in the model. The communicator tag provides the user the size of the communicator as it relates to the MPI collective operation. Modeling of fortran codes is accomplished by using the corresponding *maf_mpi_xxx* function call.

3.3. Model Creation

In this section, we discuss a motivating example using a large-scale scientific application, GYRO. GYRO makes use of a five-dimension grid, consisting of three spatial coordinates and two velocity coordinates, to propagate the system forward in time using a fourth-order, explicit, Eulerian algorithm. The code is written in Fortran90 and uses MPI for communication. Table 1 presents a subset of input parameters of GYRO that define the problem configuration and in turn affect the overall workload requirements and execution of the application.

Table 1. GYRO input parameters

Parameter	Description
n_x, n_n, n_stack, n_energy, n_lambda	grid dimensions
n_kinetic	Kinetic or adiabatic processes
Nstep	Number of simulation steps
n_proc	number of processors

The parameters in Table 1 are used throughout the execution of the GYRO application. Therefore, these parameters can be used to express the application requirements of GYRO in terms of the number of loop iterations and message size requirements for communication in the application. Also, some of the most commonly used variables can be derived from these input parameters. One such variable in GYRO is *jsplit*, which depends on a number of input values including *n_n* and *n_energy*. The parameter *jsplit* is used as a component in several loop operations of the GYRO column transpose routines and as a multiple of the message size parameter for MPI_Alltoall communication. Hence, we can express the application workload requirements and control flow in terms of just a handful of important input parameters.

Table 2. Analysis of GYRO communicators

Communicator	Communicator Alias in Application	Size Parameter
MPI COMM WORLD	GYRO COMM WORLD	n_proc
NEW_COMM_1	TRANSP_COMM	n_proc/n_n
NEW_COMM_2	SSUB_COMM	n_n
MUMPS COMM	-	1

The GYRO application makes use of four communicators that are depicted in Table 2. The GYRO_COMM_WORLD communicator is analogous to MPI_COMM_WORLD. It composes all MPI tasks that are used during the execution of the application.

NEW_COMM_1 communicator is based on the *n_proc_1* parameter and it is derived from the *n_proc* and *n_n* base parameters. The NEW_COMM_1 communicator is used in the row transpose operations of GYRO. It is redefined as TRANSP_COMM when called in those respective subroutines. The NEW_COMM_2 communicator is used extensively within the column transpose subroutines and is renamed SSUB_COMM when called by the column transpose initialization subroutine. The MUMPS_COMM communicator is used for sparse matrix operations and is always initialized to one MPI task within the GYRO application.

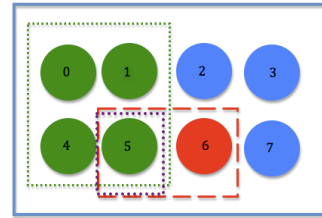


Figure 3. GYRO communicator depiction

Figure 3 depicts the possible breakdown of the various communicators within GYRO for the case of 8 processors with parameters *n_proc* = 8 and *n_n* = 2. The entire collection of processors represents GYRO_COMM_WORLD. The subset consisting of {0, 1, 4, 5} represents TRANSP_COMM. The subset consisting of {5, 6} represents SSUB_COMM. The entity {5} is representative of MUMPS_COMM. Traditional modeling strategies assume that collective communication occurs across MPI_COMM_WORLD for all communications. As seen in GYRO, this is not the case and it does not lead towards an effective modeling strategy for optimization and mapping of the application.

With the instrumentation of the application a user selects a number of key input parameters that will affect the performance of the application. Below we illustrate application instrumentation of one of GYRO's MPI_Comm_split calls. The sizes of the GYRO_COMM_WORLD communicator and the NEW_COMM_1 communicator depend on the *numproc* and *n_proc_1* parameters.

```
call MAF_MPI_COMM_SPLIT("NEW-1", "n_proc",
n_proc, "n_proc_1", n_proc_1, "NEW_COMM_1")
```

The `n_proc` parameter is declared using the standard `MPI_Comm_size` function for retrieving the size of a communicator. In this case, `n_proc` represents the number of processes in the `MPI_COMM_WORLD` or the `GYRO_COMM_WORLD` communicator. The `n_proc_1` parameter in GYRO is based upon the input arguments `n_proc` and `n_n`. Hence, it can be seen that the `NEW_COMM_1` communicator sizes will be directly dependent upon these two parameters. Within the column transpose operations the message sizes for the `MPI_Alltoall` operation depend on the `nv1` and `jsplit` parameters mentioned earlier.

```
call MAF_MPI_ALLTOALL("fssub-a2a", "nv1*jsplit
* MPI_DOUBLE_COMPLEX", nv1*jsplit*16,
"SSUB_COMM")
```

Upon execution of the instrumented code there is an immediate creation of an application call graph. At the same time, the asserted values are compared with the runtime-measured values to validate the MA calls or assertions. The intermediate model representation of GYRO can show functional depths for the MPI collective operations and the derived variables that have influenced them. We provide a representation of the intermediate model representation for GYRO in Figure 4. In this model representation, the user is able to see detailed application information in relation to the key input parameters and how they are affected in loops, MPI communicators, and MPI collective operations for the GYRO application.

```
mpi_comm_split (NAME=NEW-1) (ORIG=numproc)
                (NEW=n_proc_1) (COMM=NEW_COMM_1);

do_fulladvance()
loop (NAME=timestep) (COUNT=nstep)

    do_collision()
    rTRANSP_INIT()

    mpi_alltoall (NAME=rTRANSP_INIT-1)
                 (SIZE=s_dim*MPI_INTEGER)
                 (COMM=TRANSP_COMM);
```

Figure 4. Partial GYRO control-flow model generated from MA post-processing tool

The control-flow representation in Figure 4 gives the user a detailed overview of all of the parameters that are tracked using the modeling assertions framework. In this example, we are able to see that the `MPI_Alltoall` function of `rTRANSP_INIT` occurs through the `TRANSP_COMM` communicator. The user is then able to see that parameters `numproc` and `n_proc_1` have a direct effect on the sizes of these communicators. This model gives details into the message volume generated for `MPI_Alltoall` calls, in addition to loop iteration parameters that affect the number of collective calls that occur.

4. Evaluation of GYRO Model Assertions

Using our modeling methodology, we demonstrate a new technique for encapsulating application requirements with respect to MPI collective communication. In this section, we utilize the models developed to understand growth rates in message sizes with respect to communicator sizes and sensitivity analysis of GYRO's collective communication routines. The effects that variances in these application parameters have on the message volume of the collective routines are studied. All results are plotted on \log_2 to understand the correlation with the number of MPI tasks, which increase by a factor of 2.

4.1. Growth Rates Analysis

As shown in Figure 5, the four communicators used in GYRO change at varying rates as the MPI tasks increase. The `GYRO_COMM_WORLD` communicator increases linearly with the number of MPI tasks. The `SSUB_COMM` and `MUMPS_COMM` communicators maintain a constant number of processors for the same problem size. `SSUB_COMM`'s size is based on the `n_n` input parameter. Varying `n_n` will increase the number of processes in the communicator for different problem sizes. `MUMP_COMM` always has a value of 1 and is not used for collective communications. `TRANSP_COMM` increases with the number of MPI tasks. `TRANSP_COMM`'s size is derived by dividing `n_proc` by `n_n`.

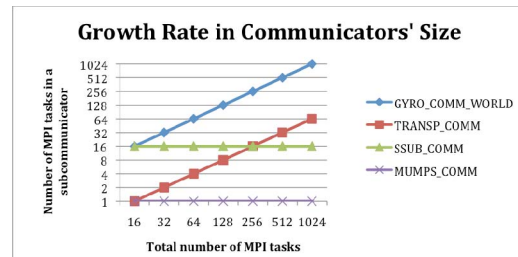


Figure 5. Comparison of communicator sizes

The number of messages in each communicator is analyzed in Figure 6. Key input parameters and the size of the `SSUB_COMM` communicator influence the communicator's behavior. The number of messages grows linearly with the communicator sizes for both `GYRO_COMM_WORLD` and `TRANSP_COMM`. Both of these communicators have `MPI_Allreduce` and `MPI_Alltoall` operations that occur based on the `nstep` loop parameter. Together these two results reveal a very important behavior that is quite challenging to understand from a flat modeling approach.

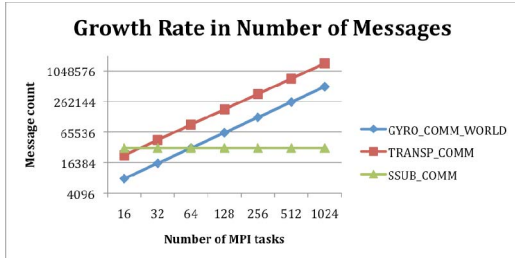


Figure 6. Message count comparison in GYRO communicators

We notice that TRANSP_COMM is increasing and the collective communication operations count in both TRANSP_COMM and GYRO_COMM_WORLD also increases. TRANSP_COMM is likely to be a scaling bottleneck as MPI tasks increase. Also, the message volume in this sub-communicator is increasing at a much faster rate than other communicators.

The symbolic models provide a range of alternative methods to obtain insight into message distribution. Figure 7 shows how the MPI_Allreduce operation is used extensively in three of GYRO’s communicators and allows for us to study message scaling in different communicators. In GYRO_COMM_WORLD and TRANSP_COMM, MPI_Allreduce increases linearly with the number of MPI tasks within each communicator. We may also study the behavior of MPI_Alltoall in this same manner.

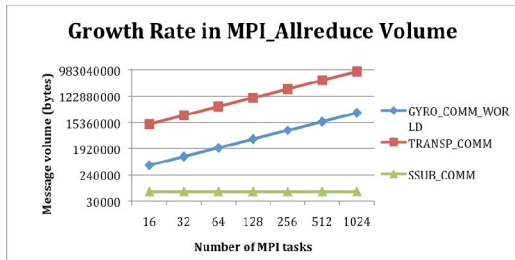


Figure 7. MPI_Allreduce message volume growth in GYRO

The TRANSP_COMM communicator makes use of multiple MPI_Allreduce operations; therefore, it has a much larger message volume than GYRO_COMM_WORLD. The SSUB_COMM shows no increase in message volume as its communicator size and input parameters for communication remain constant.

As we showed in section 2, MPI latencies are sensitive of message sizes. We investigate message size distribution and scaling using the symbolic models. Figure 8 shows that a decrease in message size occurs for the MPI_Alltoall functions in two of the communicators.

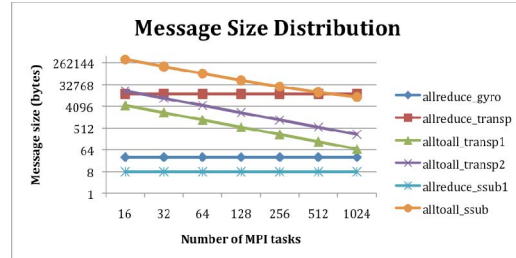


Figure 8. Message size distribution in GYRO

There are large decreases in the message size distribution for alltoall_ssub, alltoall_transp1, and alltoall_transp2. These decreases occur as parameter values in each operation decrease with increases in the number of MPI tasks. MPI_Allreduce operations in GYRO show no decreases in message size distribution.

4.2. Sensitivity Analysis

An important component of scientific discovery is the scaling of problem configuration. This is domain specific and could depend on the number of grid points, grid resolution, number of particles, granularity of simulation time, etc. We consider two grid parameters in GYRO, the radial grid and toroidal grid size, and identify the effects of these parameters on different communicators. For these experiments, the number of MPI tasks are fixed (1024 tasks).

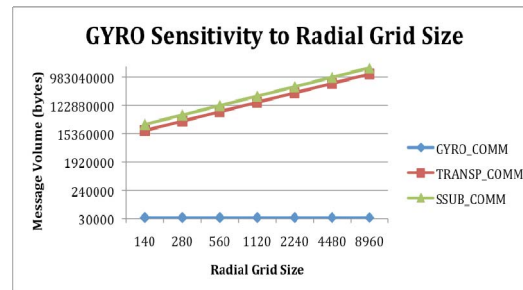


Figure 9. Message volume sensitivity to the radial grid size

TRANSP_COMM and SSUB_COMM show large increases in message volume as radial grid size increases as seen in Figure 9. TRANSP_COMM uses this parameter directly in determining its message size for a large MPI_Allreduce operation. SSUB_COMM is largely affected by MPI_Alltoall operations. As grid size increases the message size for the MPI_Alltoall operation in SSUB_COMM increases with it as the column transpose operations are directly proportional to the radial grid size.

In Figure 10, we evaluate the affects of changing the toroidal grid size parameter, and how it affects the message volume within the communicators of GYRO.

GYRO_COMM_WORLD and SSUB_COMM show no changes in message volume with respect to changes in n_n . TRANSP_COMM shows increases in the message volume when n_n is increased.

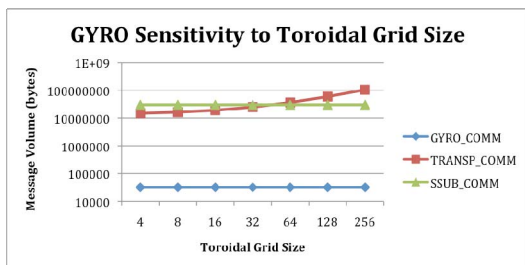


Figure 10. Message volume sensitivity to the toroidal grid size

5. Conclusions and Future Plans

It is evident that emerging computing systems will continue to make use of multicore processors and therefore modeling attempts must provide a higher level of detail for decomposing large-scale scientific applications. We have presented a new methodology for allowing application developers to hierarchically decompose and represent an application's communication requirements. Furthermore, the parameterized models allow for developers to fully represent workload requirements and conduct sensitivity analysis for future systems and problem configurations. This work will allow for applications to be mapped efficiently on capable systems to optimize MPI performance for collective communications within communicators. Future work will focus on modeling other scientific applications using the hierarchical modeling technique. In addition, we will use our models to optimize application performance on massively-parallel multicore systems.

6. Acknowledgements

This research was sponsored by the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Batelle, LLC. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. The authors would like to thank Xingfu Wu (TAMU), Heike Jagode (UTK-ORNL), Jeff Candy (General Atomi), and Mark Fahey (ORNL) for reviews of this work.

7. References

- [1] S. R., Alam, J. S. Vetter. "A Framework to Develop Symbolic Performance Models of Parallel Applications", 5th Int. Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS), 2006.
- [2] S. R., Alam, J. S. Vetter. "An Analysis of System Balance Requirements for Scientific Applications". The 35th International Conference on Parallel Processing (ICPP), 2006.
- [3] G. S. Almasi, C. Cascaval *et al.*, "Demonstrating the scalability of a molecular dynamics application on a Petaflops computer", Proc. Int'l Conf. Supercomputing, 2001, pp. 393-406.
- [4] D. H. Bailey, A. Snavey, "Performance Modeling: Understanding the Past and Predicting the Future", Euro-Par 2005: 185-195
- [5] J. Candy and M. Fahey, "GYRO Performance on a Variety of MPP Systems", in Proceedings of the 47th Cray User Group Conference, Knoxville, TN, May 16-19, 2005.
- [6] Cray XT4 Supercomputer, <http://www.cray.com/products/xt4>.
- [7] C. Lively, "Performance Analysis and Modeling of GYRO", Master's thesis, Texas A&M University, 2006.
- [8] IBM Blue Gene, http://domino.research.ibm.com/comm/research_projects.nsf/pages/bluegene.index.html.
- [9] Intel MPI Benchmarks, Users Guide and Methodology Description (Version 3.1), <http://www3.intel.com/cd/software/products/asmo-na/eng/219848.htm>.
- [10] D. J. Kerbyson, A. Hoisie, H. J. Wasserman, "Performance of Large-Scale Systems", In IEE Proceedings: Software, 150 (4): 214--221, August 2003
- [11] M. R. Fahey and J. Candy, "GYRO: Analyzing New Physics in Record Time on the Cray X1", Proceedings of the 46th Cray User Group Conference, 2004
- [12] M. R. Fahey and J. Candy, "GYRO: A 5-D Gyrokinetic-Maxwell Solver", Proceedings of the ACM/IEEE conference on Supercomputing, 2004
- [13] Message Passing Interface (MPI), <http://www-unix.mcs.anl.gov/mpi/>
- [14] NERSC Bassi, <http://www.nersc.gov>.
- [15] SciDAC-DOE's Scientific Discovery through Advanced Computing, <http://www.scidac.gov>.
- [16] A. Snavey *et al.*, "A Framework for Performance Modeling and Prediction, Proceedings of the ACM/IEEE Conference on Supercomputing, 1994.
- [17] V. E. Taylor, X. Wu, and R. Stevens, "Prophecy: An Infrastructure for Performance Analysis and Modeling System of Parallel and Grid Applications", ACM SIGMETRICS Performance Evaluation Review, Volume 30, Issue 4, March 2003.
- [18] X. Wu, V. Taylor, C. Lively, and S. Sharkawi, "Performance Analysis and Optimization of Parallel Scientific Applications on CMP Cluster Systems", ICPP2008 SMECS Workshop, 2008.