# An Analysis of System Balance Requirements for Scientific Applications

Sadaf R. Alam and Jeffrey S. Vetter
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

## Abstract

*Scientific applications are diverse in terms of the resource requirements, and tend to vary significantly from commercial applications. In order to provide sustained performance, a target high performance computing (HPC) platform must offer a balance between CPU performance to memory, interconnect and I/O subsystems performance. We characterize the system balance requirements for two large-scale Office of Science applications, GYRO (fusion simulation) and POP (climate modeling), and develop platform-independent parameterized requirement models. We measure the parallel efficiencies for GYRO and POP on three multiprocessor systems: an SMP cluster (IBM p690), a shared-memory system (SGI Altix) and a vector supercomputer (Cray X1). The higher computational intensity and interconnect bandwidth requirements of GYRO result in higher performance efficiencies on the vector platform. At the same time, small message sizes in POP benefit from low MPI latencies of the shared-memory platform. Overall results confirm system balance requirements that are generated by the requirement models.*

## 1. Introduction

The data access patterns and inter-processor communication requirements of a number of large-scale scientific applications vary significantly from database and online transaction processing applications. As a result, several critical Office of Science applications tend to achieve a small fraction of theoretical peak performance on SMP clusters [17]. The current supercomputing systems resources thereby limit the application scientists to simulate only small variants or problem sizes. The large problem configurations, which give a better approximation and understanding of the real science phenomenon, are often beyond the capabilities of current supercomputing resources. These larger simulation runs however often share the underlying algorithms and workload mapping and distribution schemes within a given implementation of an application. We therefore aim at designing the machine independent requirement models of large-scale scientific workloads that capture underlying algorithmic characteristics and that are driven by workload and scaling parameters. The focus of our study is the system balance requirements for scientific workloads. The system balance is represented as a set of ratios, for instance, ratio of the CPU performance to memory and interprocessor performance. We characterize the system balance requirements for two large-scale Office of Science applications, GYRO and POP. GYRO is a fusion science code [4] while the Parallel Ocean Program (POP) is a climate code [12].

As a first step to identify the system balance requirements of basic calculation phases, we collect empirical performance data for two applications on two parallel systems, IBM p690 and SGI Altix. GYRO and POP are written in Fortran, and use MPI communication library. We define the workload requirements in terms of floating-point computation, memory operations and message passing communication patterns and sizes. The first two costs depend on the compiler and the underlying system hardware, but the patterns and sizes of the MPI messages are platform independent. We therefore normalize the floating-point and memory costs but precisely measure the MPI message patterns and sizes. Although vendor-specific performance tools are available on the p690 and Altix platforms, we use two portable performance tools, TAU and mpiP, to collect a consistent set of performance data. TAU (Tuning and Analysis Utilities) is a performance analysis framework, which has been designed to support performance analysis for a general model of parallel computation [9]. It interfaces with a number of tools including the hardware counter API called PAPI (Performance Application Programming Interface) [2]. mpiP is a profiling library for MPI-based applications [15]. It collects statistical information about MPI functions.

We collect PAPI hardware counter data and identify the scaling behavior of functions and MPI messages within a basic phase of scientific calculations. We then develop a hierarchical (multiple functions in a basic phase), phase-based parameterized simulation models for GYRO and POP

that represent control flow in a simulation time-step and are independent of the target platform. The models are validated using the experimental results. From the quantitative data generated by the models, we derive memory byte-to-flop ratio for the basic phases of calculation and inter-node message transfer rates. This derived data is useful in identifying the appropriate machine balance requirements for these applications. Micro-benchmarking and vendor supplied data are used to determine the target system characteristics. We measure parallel efficiencies on three target platforms: IBM p690 (an SMP cluster), SGI Altix (a shared memory system) and Cray X1 (a vector supercomputer) at the Oak Ridge National Laboratory [16]. The higher computational intensity and interconnect bandwidth requirements of GYRO result in higher performance efficiencies on the Cray X1 vector platform. At the same time, small message sizes in POP benefit from low MPI latencies of the SGI Altix platform. Overall results confirm the system balance requirements that are generated by the requirement models.

The paper outline is as follows: section 2 presents an overview of GYRO and POP applications. Section 3 describes our workload characterization and modeling methodology. Section 4 provides the validation and performance prediction experiments and results. An outline of the related work in the area of scientific workload characterization and performance prediction is presented in section 5. Finally, section 6 concludes the research.

## 2. Applications

A number of large-scale scientific applications have some common characteristics: there is a notion of discretization of continuum to a multi-dimensional grid, a mapping of time to discretized events and mapping and distribution of grid points to the processing nodes of a parallel system. In a Single Program Multiple Data (SPMD) programming paradigm, sequence of operations are repeated during logical, simulation time-steps on all processing nodes. GYRO and POP are the examples of such applications.

### 2.1. GYRO

GYRO is a code for the numerical simulation of tokamak microturbulance, solving time-dependent, nonlinear Gyro-Kinetic-Maxwell (GKM) equations [4]. It is used by the researcher worldwide to study phase physics, which helps in understanding the concept of power production by fusion reactions. GYRO uses a five-dimensional grid (three spatial and two velocity coordinates) and propagates the system forward in time using a fourth-order, explicit, Eularian algorithm. There are three problem instances of GYRO, two instances use multiple of 16 processors while the most demanding uses multiple of 64 processors. GYRO has three problem instances: B1-std, B2-cy and B3-gtc [3] where B1-std has a 16 x 140 x 8 x 8 x 20 x 2 problem resolution. The input parameters for the GYRO requirement models are listed in table 1.

| Parameter | Description |
|---|---|
| nstep | number of simulation steps |
| time_skip | write output files per nstep |
| n_proc | number of processors |
| n_x, n_n, n_stack, n_orbit, n_energy | grid dimensions |
| n_kinetic | kinetic or adiabatic processes |

**Table 1. Input parameters for GYRO requirement models**

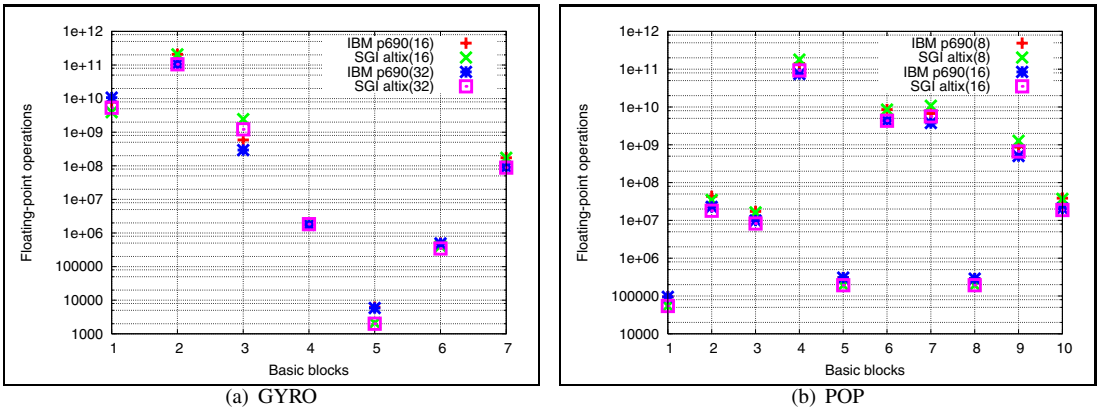
The cost per time-step in a GYRO simulation is presented in equation (1).

$$\begin{aligned} C_{nstep} &= C_{collision} + C_{timestep} + C_{field\_plot} + \\ &\quad C_{adaptive} + C_{field\_fluxave} + \\ &\quad C_{error} + C_{write@TIME\_SKIP} \end{aligned} \quad (1)$$

The control flow of the application also depend on input parameters. For instance, $C_{collision}$ does not take place in all problem instances and $C_{write}$ cost only incur at the time skip intervals, which are specified in the input scripts. In addition, a separate $C_{timestep}$ function is called for B3-gtc. The number of floating-point and memory access is collected for individual functions within a basic calculation phase. The communication cost can be point-to-point or collective. The collective communication operations, MPI_ALLTOALL and MPI_ALLREDUCE in GYRO, not only send and receive different message sizes but also take place in different sub-communicators. There are altogether five MPI communicators with three different sizes.

### 2.2. POP

POP is an ocean modeling code developed at the Los Alamos National Laboratory, which executes in a time-step fashion and has a standard latitude-longitude grid with (km) vertical levels. There are two main processes in a POP time-step: baroclinic and barotropic. Baroclinic requires only point-to-point communication and is highly parallelizable. Barotropic contains a conjugate gradient solver, which requires global reduction operations. Moreover, the discretized POP grid is mapped and distributed evenly on available two-dimensional processor grid. POP has two standard




(a) GYRO (b) POP

**Figure 1. Number of floating-point operations for each basic phase of calculation**

problem instances: x1 and .01 [12]. Processor grid dimensions are compile time parameters in POP. Input parameters for the POP requirement model are listed in table 2.

| Parameter | Description |
|---|---|
| n_procx | number of processor in x dimension |
| n_procy | number of processor in y dimension |
| nstep | number of simulation steps |
| imt, jmt, km, nt | POP grid points |

**Table 2. Input parameters for POP models**

The cost of a simulation time-step in POP is composed of the computation and communication costs listed below.

$$\begin{aligned} C_{nstep} &= C_{surface\_forcing} + C_{tavg\_2d} + C_{dhdt} + \\ &\quad C_{diag} + C_{bclinic\_driver} + C_{boundary\_bclinic} + \\ &\quad C_{btropic\_driver} + C_{bclinic\_correct} + \\ &\quad C_{boundary\_btropic} + 2 * km * C_{state} \end{aligned} \quad (2)$$

Local workload depends on the number of grid points assigned to a processor. POP uses a virtual, two-dimensional mesh topology `nproc_x X nproc_y`. The local grid dimensions imt_local for instance is calculated as:

$$imt\_local = \frac{imt - 1}{nproc\_x} + 1 + 2 * num\_ghost\_cells$$

The two grid dimensions, `km` and `nt`, are not divided among processors. The point-to-point communication is nearest neighbor only (in two dimensions) and the message volume depends on the local workload volume (`imt_local and jmt_local`). POP has a single communicator for the MPI operations.

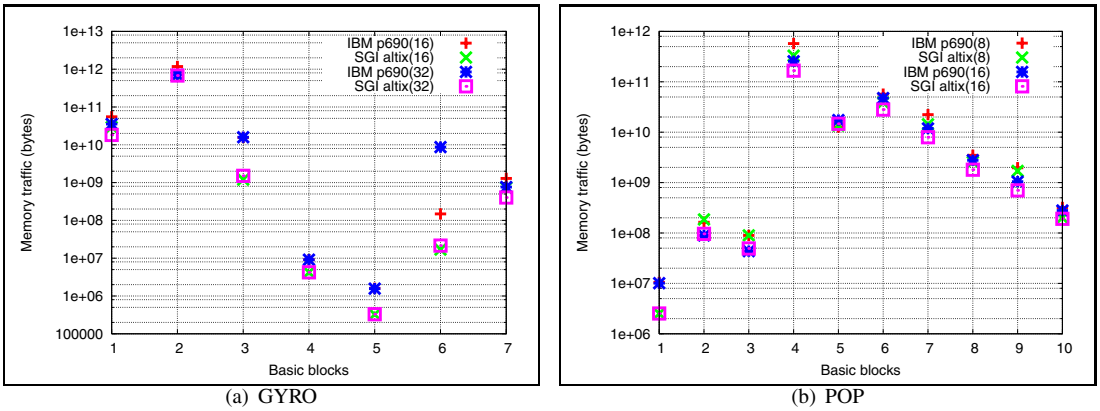
## 3. Methodology

The creation of the requirement models is a three-step process. First, the simulation phases and workload parameters are identified using the profiling and debugging tools, and a first-order analytical requirement model is created (presented in the previous section). Second, the empirical data is collected on the two parallel platforms, IBM p690 and SGI Altix, with different processor count and workload distribution schemes. The empirical data is analyzed and utilized in designing the requirement models. Finally, the workload models are verified against the experimental results.

### 3.1. Data collection and analysis

The empirical performance data is collected by altering the processor count, workload parameters and the number of simulation time-steps. This enabled us to identify the functions that depend on the local workload volume as well as functions that are scaling invariant. Figure 1 shows the floating-point operation count for GYRO and POP. The hardware counter values are collected on p690 and Altix for basic phases of computation with two different processor counts. For GYRO, experiments are run on 16 and 32 processing nodes and for POP two different processor grid sizes are used: `2x4` and `4x4`. PAPI hardware counters are used together with the TAU callgraph facility, such that counter values are collected on the call tree nodes. There are some variations in the floating-point and load-store operation count on the two systems, however, typically the differences in the PAPI hardware counter values is approximately 10%.

Figure 2 shows the memory read and write traffic generated per time-step for GYRO and POP. These results show the scaling pattern and behavior of the basic phases and en-




(a) GYRO (b) POP

**Figure 2. Data transfer requirements within the basic phases of calculation**

abled us to identify the most demanding calculation phases of an application. In GYRO for instance, phase 2 is most demanding both in terms of the floating-point computation and memory operations.

The MPI profiling was done using the mpiP profiling tool. GYRO has a very small number of point-to-point communication operations. A large number of MPI calls are MPI_ALLTOALL and MPI_ALLREDUCE with two different sub-communicator sizes. POP, on the other hand, has a large number of nearest-neighbor, point-to-point communication operations as well as frequent, fixed-size, MPI_ALLREDUCE operations. The distribution of MPI message volume in presented in section 4.2 (figure 6).

### 3.2. Inter-comparison

GYRO and POP applications have a number of similar characteristics including an SPMD programming paradigm and a notion of simulation time-steps. However, the way in which the workload is mapped and distributed on parallel resources is substantially different for the two applications. For example, the distribution of array indices onto MPI tasks are not fixed in GYRO. The indices are reassigned and redistributed using transpose operations that involves calling MPI_ALLTOALL functions for different phases of calculation. GYRO simulations employ MPI subcommunicators to reduce the collective communication and memory requirements per node. POP on the other hand has a single communicator. The two-dimensional nearest-neighbor communication and global reduction operations in its conjugate gradient algorithm dominate the POP simulations.

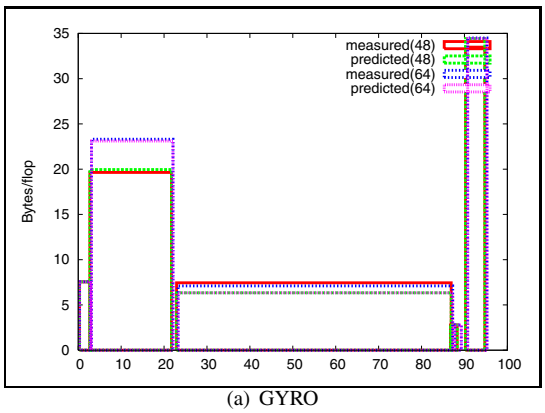
Except for a couple of user-defined input-output and check-pointing steps, both applications repeat a fixed sequence of operations in logical time steps. Thus, the number of time-step iterations are an input parameter in workload models for POP and GYRO. However, the number of conjugate gradient iterations in POP's barotropic process rely on the runtime residual values. Hence, requirement model of POP takes number of conjugate gradient iterations as a fixed input along with the number of time-steps. Unlike the real application, number of iterations remain constant in the performance model; the number of iteration parameter provides an upper bound to the barotropic process runtime.

## 4. Experiments

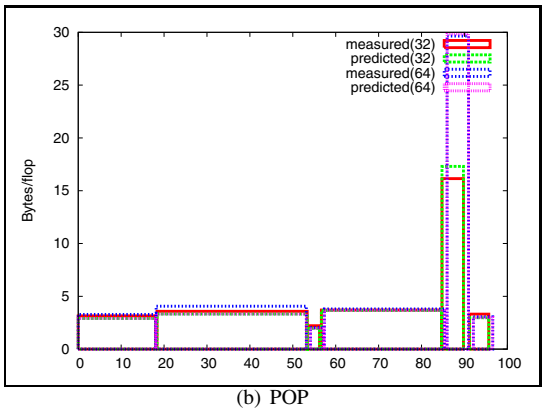
### 4.1. Model validation

The requirement models are validated by running multiple experiments with different processor count and by collecting hardware counter data on the call tree nodes. The measured values are then compared with the data generated by the requirement models. Figure 3 shows the ratio of data transfered as a result of load-store operations to the floating-point operations during GYRO and POP simulations. In figure 3, rather than comparing the byte-to-flop ratio for basic phases, we show the dominant system balance requirements during a simulation time-step. We calculate the percentage of floating-point operations per basic phase and only show the compute-intensive phases of calculation in figure 3. These compute-intensive phases account for over 95% of total floating-point operations per simulation time-step. For instance, the phase 2 in GYRO and phase 4 in POP (figure 1) are the most expensive calculation phases in terms of the floating-point calculation requirements per simulation time-step. The byte-to-flop ratio for these blocks are around 7 and 4 respectively.

Figure 3 shows that our model results are within the 20% range of the experimental values. It also shows the scaling characteristics for the two applications. In POP, the byte-to-flop ratio does not scale from 32 to 64 processing

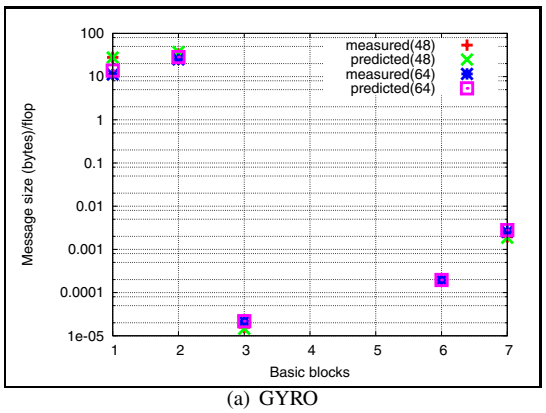



(a) GYRO

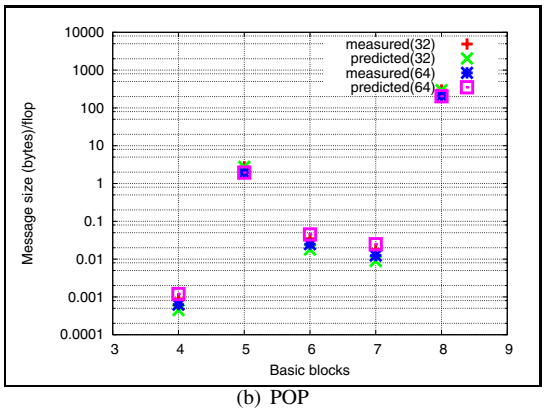



(b) POP

**Figure 3. Byte-to-flop ratio distribution**




(a) GYRO




(b) POP

**Figure 4. Communication volume (bytes)-to-flop ratio distribution**

nodes except for a single code execution block that represents approximately 5% of total floating-point operations. On the other hand, the byte-to-flop ratio increases slightly for over 80% code blocks as the workload volume decreases in GYRO for experiments on 48 and 64 processing nodes. This is attributed to a large number of collective communication operations.

Since the size and pattern of the MPI operations do not vary with the underlying system hardware and compiler, the communication requirements generated by the workload requirement models are identical to the results collected for GYRO and within 2% error range for POP. POP uses a conjugate gradient iterator and the number of iterations required to converge vary from one simulation step to other. Figure 4 shows the communication volume (byte) per floating-point operation for POP and GYRO respectively.

We also compare and contrast the two applications in terms of their overall system balance requirements. Figure 5(a) compares the data memory byte-to-flop ratio for GYRO and POP. The aggregate memory requirement increases rapidly for GYRO. Figure 5(b) compares the communication volume to flop ratio for GYRO and POP. Although the overall communication volume increases significantly for large processor runs in POP, we will show in the next section that the message size for individual processing nodes remain in order of a few KBytes. In GYRO, since the size of one sub-communicator remain constant, the message sizes reduce at a faster rate than the total floating-point operation count.

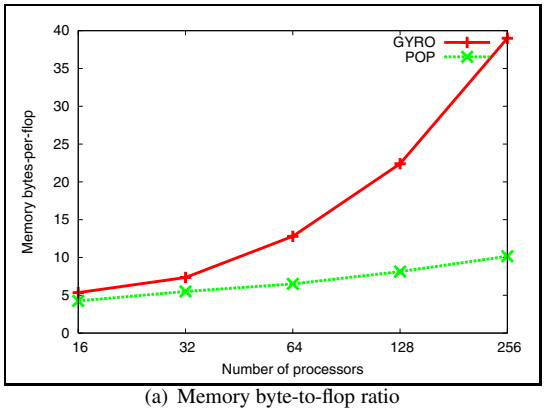
## 4.2. Architecture mapping

After the validation of requirement models, we assess the mapping of the two scientific applications on three architecturally different parallel systems: IBM p690, SGI Altix and Cray X1. The key architectural features of the three systems are listed in table 3. Details can be found at [14].

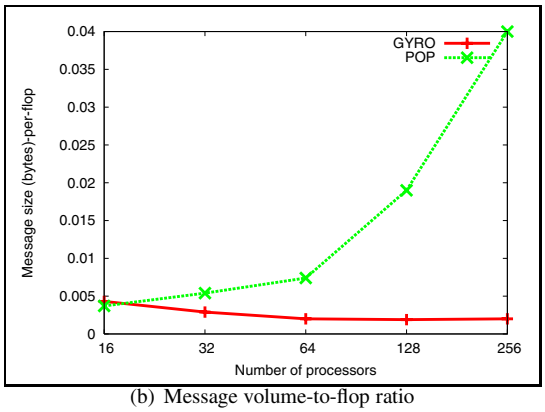
In addition to system metrics, we compare the machine balance offered by the three systems in terms of data memory bytes accessed per floating-point operation and internode byte communication volume per floating-point operations. Byte-per-flop for p690 and Altix is 1, while for the X1 it is around 3. The bytes communicated per flop is less than 0.1 for p690, about 0.2 for Altix and is 1 for the X1. Thus,

| Feature | IBM p690 | SGI Altix | Cray X1 |
|---|---|---|---|
| System classification | SMP cluster | shared memory | vector MPP |
| Processor peak performance (GFlops) | 5.2 | 6 | 12.8 |
| Main memory bandwidth (GB/s) | 51 (per Multi- chip-module=32 processors) | 6.4 | 26 |
| MPI bandwidth (MB/s) | 174 within node, 2186 inter-node | 1968 | 12125 |

**Table 3. Architectural features of the IBM p690, SGI Altix and Cray X1**



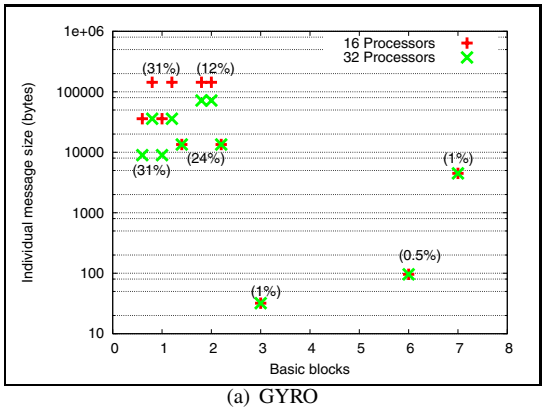

(a) Memory byte-to-flop ratio



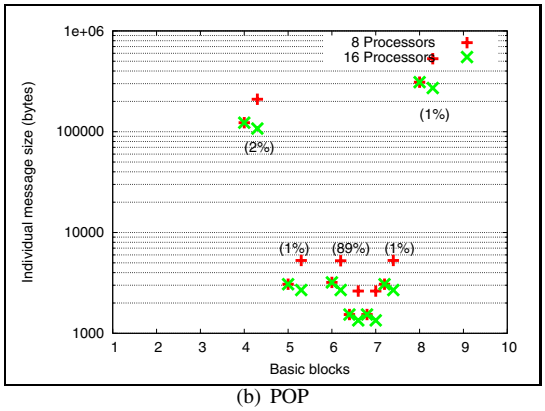

(b) Message volume-to-flop ratio

**Figure 5. Overall system balance requirements for GYRO and POP**

the Cray X1 system provides the highest memory byte-to-flop and communication byte-to-flop ratio. Although, the MPI bandwidth in Cray X1 is the highest, the MPI latencies are about 3 times large as well compared to the other two systems. The high MPI latencies affect the small message sizes. In order to understand the MPI message distribution and scaling characteristics, we calculate the MPI message sizes and patterns for the two applications. Figure 6(b) shows point-to-point message distribution in POP and figure 6(a) shows distribution of collective communication operations in GYRO. As shown in the figure 6(b), the large volume of communication in POP takes place with small data packets (order of few KBytes). The frequent MPI_ALLREDUCE operation is always performed on 8 bytes. On the other hand, data packets of 10-100s KByte are exchanged in the frequent collective communication operations in GYRO (figure 6(a)).



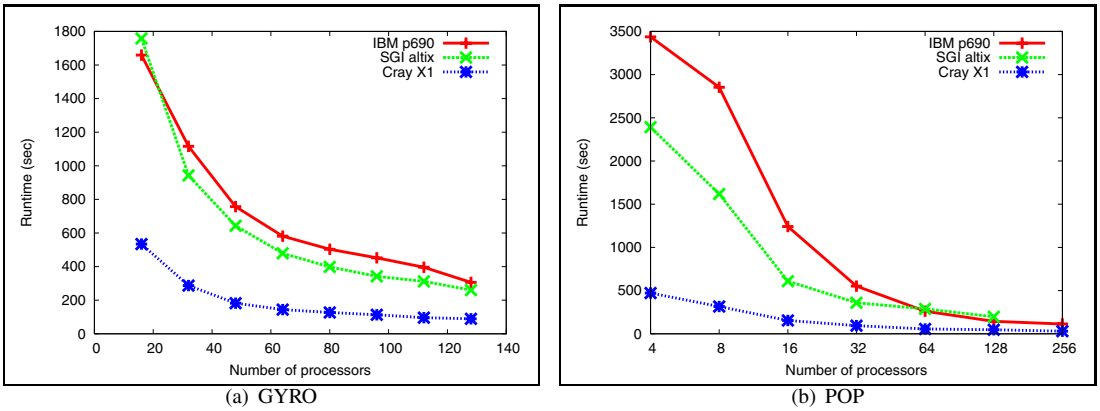

(a) GYRO




(b) POP

**Figure 6. Distribution of MPI message volume**

We ran Pallas MPI benchmarks (PMB) to identify the native MPI implementation characteristics [18]. The SGI Altix and IBM p690 systems show better performance for the PingPong benchmark for small message sizes (less than 1 KByte) due to low point-to-point communication latencies. However, large message sizes benefit from high bandwidth network of the Cray X1 system. The latencies for the collective communication operations, MPI_ALLTOALL




(a) GYRO (b) POP

**Figure 7. Parallel efficiency of the applications on IBM p690, SGI Altix and Cray X1**

and MPI_ALLREDUCE, are measured using the PMB collective communication benchmarks. Like the point-to-point communication, performance for small message sizes in collective communication operations are lower on the X1 system. Moreover, the latencies for MPI_ALLTOALL and MPI_ALLREDUCE operations increase rapidly on the Altix system as compared to the other two systems.

From the machine balance view-point, the Cray X1 system has a clear advantage on the other two systems for the two applications. GYRO has a high memory access to flop count ratio and message volume to flop count ratio; therefore, the overall performance is expected to be consistently higher than the other two platforms. At the same time, the implementation of the MPI collective operations, particularly for small message sizes, can slightly reduce the overall performance gains. Altix has a performance advantage over p690 because of its higher clock speed, comparatively high memory bandwidth and very low point-to-point MPI latencies. Thus, the overall performance of POP in particular is expected to be higher on Altix than on the p690 system due to a relatively low memory and message volume to flop ratios. The collective communication latencies are typically higher on the Altix system than the p690 system with processor count $> 16$, therefore, Altix could not sustain a higher performance for GYRO with a large number of MPI tasks.

The performance and scaling behavior of GYRO and POP are shown in figure 7. Results in the figure 7 confirm our characterization of the two workloads and analysis of the target system behavior. In terms of the peak processor performance, X1 vector processor is twice as powerful as the Altix Itanium2 processor, which in turn is almost twice as fast as the p690 Power4 processor. The Cray X1 systems show largest performance gains for experiments with up to 8 and 48 MPI tasks for POP and GYRO respectively. This is primarily because of the large workload volume (computation and communication requirements) per processor with small processor count. The SGI Altix system does not show significant performance gains over p690 due to the MPI_ALLTOALL bottleneck in GYRO, which we identified in the PMB benchmark runs. Likewise, high collective communication latencies with a large processor count (greater than 32) affect performance of POP on the Altix system. At the same time, the low-latency point-to-point communication latencies on Altix contribute to the higher performance for the nearest-neighbor communication operations in POP for up to 32 processors.

## 5. Related work

Luo and Cameron [8] collected the hardware counter values of ASCI workloads on two parallel systems and devised analytical models that represent the CPU requirements. I/O characterization of I/O intensive scientific workload was presented by Pasquale and Polyzos [10]. Although these workload characterization and analytical modeling studies provide insights into a single feature of a given application, the workload models do not provide a mechanism to understand and to study larger problem instances, which are of interest to the scientific community.

Predictive performance models for a number of benchmarks and a couple of large-scale scientific workloads have been developed using a variety of approaches. Designing accurate and reliable models of large-scale, production-level scientific code is considered as a time-consuming process requiring an expert knowledge of the application design and its underlying algorithms [5, 6, 7]. These performance models precisely capture the communication and computation characteristics of the application by extensive micro-benchmarking, simulation and actual measurement. Alternatively, performance characteristics are investigated by gathering simulation traces [11]. The tracing schemes,

which generates substantial amount of data for full-scale applications, provides limited information about the application runs with different problem sizes and machine configurations.

Our scheme primarily relies on capturing the control flow, scaling characteristics and workload mapping and distribution scheme of the application as well as on the collection of empirical performance data using portable performance tools. Depending on the target platform and the underlying compiler implementation, we anticipate that additional platform-dependent parameters will be necessary to predict achievable performance precisely using the requirement models. For instance, detailed performance models of POP presented in [7] include both application parameters and a number of target system parameters. In addition, such a model development requires an expert understanding of the code implementation and the underlying algorithms in addition to extensive measurements and empirical data collection on a range of parallel platforms.

## 6. Conclusions

Future supercomputing systems are likely to have a large number of processing nodes and are going to offer unconventional combinations of system balances. Bluegene/L [1] and Cray X1E [13] are examples of such systems. It is therefore essential to characterize and to get an insight into the applications performance behavior that are going to exploit future supercomputing resources. Our low-overhead characterization scheme involve collecting and analyzing empirical performance data from existing supercomputing systems to identify system balance preferences for scientific applications. We identify key calculation phases and, using profile data, develop machine-independent requirement models for two Office of Science applications. We measure parallel efficiencies for the two applications on a combination of scalar and vector parallel systems. The results confirm the system balance requirements that are generated by the requirement models. The workload characterization and parameterized requirement models are thus useful not only in identifying optimal architectural mapping for an application but also in leveraging the designs of the next generation supercomputing systems.

### Acknowledgments

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.

## References

[1] N. R. Adiga *et. al. An overview of the BlueGene/L Supercomputer*, Proceedings of the ACM/IEEE conference on Supercomputing, 2002.

[2] S. Browne, *et. al. A Portable Programming Interface for Performance Evaluation on Modern Processors,* The International Journal of High Performance Computing Applications, Volume 14, number 3, 2000.

[3] 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.

[4] M. R. Fahey and J. Candy, *GYRO: A 5-D Gyrokinetic-Maxwell Solver*, Proceedings of the ACM/IEEE conference on Supercomputing, 2004.

[5] A. Hoisie, *et. al. A General Predictive Performance Model for Wavefront Algorithms on Clusters of SMPs,* Proceedings of the International Conference on Parallel Processing, 2000.

[6] D. J. Kerbyson, *et. al. Predictive performance and scalability modeling of a large-scale application*, Proceedings of the ACM/IEEE conference on Supercomputing, 2001.

[7] D. J. Kerbyson and P. W. Jones, *A Performance Model of the Parallel Ocean Program*, International Journal of High Performance Computing Applications, Vol. 19, No. 3, 261-276 (2005).

[8] Y. Luo, *et. al. Instruction-level Characterization of Computational Physics and Multimedia Applications Using Performance Counters*. Workshop on Workload Characterization, 1998.

[9] A. Malony and S. Shende, *Performance Technology for Complex Parallel and Distributed Systems*, Third Austrian-Hungarian Workshop on Distributed and Parallel Systems, DAPSYS 2000.

[10] B. K. Pasquale, and G. C. Polyzos, *Dynamic I/O characterization of I/O-intensive Scientific Application*, In Proceedings of Supercomputing, 1994.

[11] A. Snavely, *et. al. A framework for performance modeling and prediction*, Proceedings of the ACM/IEEE conference on Supercomputing, 2002.

[12] P. Worley and J. Levesque, *The performance evolution of the Parallel Ocean Program on the Cray X1*, Proceedings of the 46th Cray User Group Conference, 2004.

[13] Cray X1E Supercomputer, http://www.cray.com/products/x1e/

[14] Evaluation of Early Systems at ORNL. http://www.csm.ornl.gov/evaluation/

[15] *mpiP: Lightweight, Scalable MPI Profiling* (http://www.llnl.gov/CASC/mpip/).

[16] The National Center for Computational Sciences (NCCS) at ORNL. http://www.ccs.ornl.gov/user/computers.html

[17] Networking and Information Technology Research and Development (Blue book). Available at http://www.hpcc.gov/pubs/bluebooks/2005/index.html

[18] *Pallas MPI Benchmarks*, http://www.pallas.com/e/products/pmb/