PERFORMANCE ANALYSIS AND MODELING OF GYRO

A Thesis

by

CHARLES WESLEY LIVELY III

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Computer Engineering

PERFORMANCE ANALYSIS AND MODELING OF GYRO

A Thesis

by

CHARLES WESLEY LIVELY III

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Valerie E. Taylor
Committee Members,	Karen L. Butler-Purry
	Eun Jung Kim
Head of Department,	Valerie E. Taylor

August 2006

Major Subject: Computer Engineering

ABSTRACT

Performance Analysis and Modeling of GYRO. (August 2006)
Charles Wesley Lively III, B.S.E., Mercer University
Chair of Advisory Committee: Dr. Valerie E. Taylor

Efficient execution of scientific applications requires an understanding of how system features impact the performance of the application. Performance models provide significant insight into the performance relationships between an application and the system used for execution. In particular, models can be used to predict the relative performance of different systems used to execute an application. Recently, a significant effort has been devoted to gaining a more detailed understanding of the performance characteristics of a fusion reaction application, GYRO. GYRO is a plasma-physics application used to gain a better understanding of the interaction of ions and electrons in fusion reactions. In this thesis, we use the well-known Prophesy system to analyze and model the performance of GYRO across various supercomputer platforms. Using processor partitioning, we determine that utilizing the smallest number of processors per node is the most effective processor configuration for executing the application. Further, we explore trends in kernel coupling values across platforms to understand how kernels of GYRO interact.

In this work, experiments are conducted on the supercomputers Seaborg and Jacquard at the DOE National Energy Research Scientific Computing Center and the supercomputers DataStar P655 and P690 at the San Diego Supercomputing Center. Across all four platforms, our results show that utilizing one processor per node (ppn) yields better performance than full or half ppn usage. Our experimental results also show that using kernel coupling to model and predict the performance of GYRO is more accurate than summation. On average, kernel coupling provides for prediction estimates that have less than a 7% error. The performance relationship between kernel coupling values and the sharing of information throughout the GYRO application is explored by understanding the global communication within the application and data locality.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the people who have supported me in completing this thesis. I am especially indebted to my advisor, Dr. Valerie E. Taylor, who has provided encouragement, insight, and guidance over the course of the work presented in this thesis. I would also like to thank Dr. Xingfu Wu for always providing guidance, support, and immediate feedback. I would like to thank Dr. Patrick Worley for allowing me to be involved in such an important project. Also, I would like to thank Dr. David Bailey for allowing me access to several of the supercomputer platforms utilized in this thesis. Finally, I would like to give thanks to my family and friends who have been a source of never-ending encouragement and support.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Performance models are effective methods for providing thorough understanding of the relationship that exists between an application and the system that is utilized in executing the application [11]. Currently, there is a need for more accurate scientific methods that will allow for a more precise relationship between these two entities. The relationship between the application and the system is expected to vary across different system architectures. Having a thorough understanding of the functions that compose an application can lead to a better understanding of the application and therefore yield better performance. This work is focused on analyzing this relationship for a particular application, called GYRO, executed on multiple parallel systems.

GYRO is an application that is capable of facilitating a better understanding of plasma microinstabilities and turbulence flow in the tokamak geometry. GYRO is the most advanced Eulerian gyrokinetic-Maxwell equation solver used by scientists world-wide [1]. Power production by fusion reactions through the use of the tokamak is a highly studied area. To gain a better understanding of the confinement properties needed to control the tokamak plasmas, one must use a gyrokinetic-maxwell equation solver, such as GYRO.

GYRO is an open-source physics application developed primarily in FORTRAN90. Improving the performance of the application will yield better results and further the application's future performance capabilities. By identifying bottlenecks within the application and understanding how the application performs on various supercomputing platforms, feedback can be given to developers to improve the application's performance. This work investigates GYRO's performance in relation to four key parameters: application input, application execution time, number of processors, and system architecture. A comparative analysis of these four parameters is done to yield a more detailed understanding of the application's performance. Processor partitioning is used to determine the most effective processor configuration for executing GYRO. In addition, kernel coupling is used to investigate the sharing of information within the application [2].

This thesis follows the style of ACM Sigmetrics.

Experiments for this research were executed on several supercomputer platforms. These platforms include the DataStar P655 and P690 available at the San Diego Supercomputing Center (SDSC) [10] and Seaborg available at the DOE National Energy Research Scientific Computing Center (NERSC) [9]. Additionally, a Linux Opteron cluster, the NERSC Jacquard, was also utilized to provide a different platform for comparison [8].

This thesis focuses on the following areas of inquiry in analyzing the performance characteristics of the GYRO application:

1. What processor configuration is most effective for GYRO's execution?
2. How do the kernels of GYRO affect the application's performance?
3. Which supercomputer platforms are most efficient for executing GYRO?
4. How do the kernel coupling values change across various platforms?
5. What functional components of GYRO can be improved to reduce execution time and increase constructive coupling?
6. How do the results of the coupling analysis compare to that of other methods to predict the application performance?

The experimental results indicate that in terms of processor partitioning it is most efficient to execute GYRO using one processor per node (ppn) on all platforms. The global communication of the application is extremely high and utilizing one ppn allows for a reduction in the communication in the application by reducing intra-node communication (or communication within a node) that is incurred from MPI_Alltoall and MPI_Allreduce operations. For example, in terms of GYRO's full execution, using 32 processors per nodes, the execution of GYRO on 2 nodes with 16 processors per node (2 x 16) is 38% more than utilizing 32 nodes with 1 processor per node (32 x 1) on the P655. Further, utilizing 1 ppn, trends in kernel coupling values were investigated to show that there is a strong relationship between the coupling values and the communication and data locality. For example, on all platforms we see that as the number of processors (using 1 ppn) increases the average message size tends to decrease. This results in more data being transmitted overall because there is less congestion in the network. On the NERSC Seaborg we see a strong relationship between kernel coupling values and data locality. On this platform, as the kernel coupling values become less than one, the L1 cache hit rate increases.

The remainder of this thesis is organized as follows. Section 2 presents background for the experimental systems and the application. Section 3 discusses the implications of an effective processors partitioning scheme for the application. Section 4 identifies trends in kernel coupling values and discusses the relationship between coupling values and communication and data locality for GYRO on four different supercomputing platforms. Section 5 discusses previous work that relates to this thesis. Section 6 provides conclusions for this thesis and discusses future directions.

2. BACKGROUND

2.1 Experimental Platforms

In this section we describe the various platforms used to analyze the performance characteristics of GYRO and provide pertinent background information about the application. Performance data is obtained from four different parallel platforms: the DOE National Energy Research Scientific Computing Center Seaborg and Jacquard platforms, and the San Diego Supercomputing Center's DataStar P655 and P690 platforms. Table 1 provides an overview of the technical specifications for the four parallel platforms.

Table 1. Parallel Platforms Overview

Configurations	**P655**	**P690**	**Seaborg**	**Jacquard**
Number of Nodes	176	7	380	356
CPUs per Node	8	32	16	2
CPU type	1.5GHz P4	1.7GHz P4	375MHz P3	2.2 GHz Opteron
CPU Peak Speed	6.0 GFlops	6.8 GFlops	1.5GFlops	4.4 GFlops
Memory per Node	16GB	128GB	16-64GB	6GB
L1 Cache	64/32 KB	64/32 KB	64/32 KB	64/64 KB
L2 Cache	1.5MB	1.5MB	8MB	1.0MB
L3 Cache	128MB	128MB	N/A	N/A
Network	Federation	Federation	Colony	Infiniband

The NERSC Seaborg machine is an IBM cluster with 380 16-way Power3 (375 Mhz) compute nodes. The network switch utilized by Seaborg is the IBM "Colony" which connects to two "GX Bus Colony" network adapters per node. The peak performance of a processor on this platform is 375 x 4 = 1.5 GFlops / s.

The SDSC DataStar P655 machine is an IBM p655 cluster with 176 8-way Power4 (1.5 GHz) compute nodes. The nodes of the P655 are connected by the Federation interconnect. The peak performance of each node for the P655 platform is 8 x 1.5 x 4 = 144 GFlops / s.

The SDSC DataStar P690 machine is an IBM p690 cluster with 8 32-way Power4 (1.7 GHz) compute nodes. The nodes of the P690 are connected by the Federation interconnect. The peak performance of each node for the P690 platform is 32 x 1.7 x 4 = 217.6 GFlops / s.

The NERSC Jacquard machine is an AMD Linux cluster with 320 2-way Opteron (2.2 GHz) compute nodes. The nodes of Jacquard are connected using a high-speed Infiniband switch interconnect. The peak performance of a processor on this platform is 4.4 GFlops / s.

Extensive network performance testing in relation to the bandwidth and latency on the shared memory symmetric multiprocessors (SMPs) used in this thesis have been completed [14]. The results indicated that the Federation interconnect, used on the DataStar platforms, allows for shorter latency and a larger bandwidth when compared to the Colony interconnect, used on the Seaborg platform.

2.2 Description of Methodology and GYRO

In this section, we describe the methodology and techniques for measuring coupling for the adjacent kernels within the GYRO application. A detailed explanation of the kernels analyzed in the application is also given. To evaluate the performance characteristics of GYRO kernel coupling is used. Kernel coupling provides a metric that determines whether adjacent functions within an application are sharing resources efficiently. Therefore, analyzing trends in kernel coupling values for adjacent functions provides a baseline indicator of the performance of GYRO.

2.3 Adjacent Coupling

In previous work, Geisler and Taylor -provided the specifications for quantifying the interaction between adjacent kernels in an application [2]. A kernel is defined as a unit of computation that denotes a logical entity within the larger context of an application. In general, a kernel may be a loop, procedure, or file depending on the level of granularity of detail that is desired from the measurements. In this work, the GYRO application is divided into eight kernels, six of the kernels being computational kernels. These kernels represent top-level subroutines that have been grouped together.

To compute the coupling parameter c_{ij}, there are three measurements that must be taken:

1. p_i is the performance of kernel i in isolation,
2. p_j is the performance of kernel j in isolation, and
3. p_{ij} is the performance of kernels i and j together, assuming that kernel i immediately precedes kernel j in the application.

The value c_{ij} represents the interaction between two adjacent kernels in an application. In general, for an application consisting of N kernels, only N-1 pairwise kernel interactions need to be measured.

The parameter c_{ij} can be grouped into three categories:

- $c_{ij} = 1$ indicates no interaction between the two kernels, yielding no change in performance.
- $c_{ij} < 1$ results from resource(s) being shared between the kernels, producing a performance gain.
- $c_{ij} > 1$ occurs when the kernels interfere with each other, resulting in a performance loss.

The equation for computing c_{ij} is presented as equation 1.

$$C_{ij} = \frac{P_{ij}}{P_i + P_j} \tag{1}$$

The coupling parameter can be generalized to apply to chains of kernels, as shown in equation 2. The parameter C_{ij} becomes C_w, assume that W represents an ordered chain of K kernels. Therefore, P_w represents the execution time of the chain W. Note that for K=2, equation 1 is equal to equation 2.

$$C_w = \frac{P_w}{\sum_{j \in W} P_j} \tag{2}$$

The coupling parameter is used in the estimation of execution time using equation 3. N_i represents the number of times that kernel i is executed. In the case of GYRO, the kernels occur in loops and N_i represents the number of times that the loop is executed. P_i represents the execution time of kernel i, and α_i is the weighted average of the coupling values that are associated with kernel i.

$$T = \sum_{i=1}^{n} \alpha_i N_i P_i \tag{3}$$

The parameter Q_i represents the set of all ordered chains of k ($2 \leq k \leq$) kernels that are involved with kernel i. The size of the Set Q_i is $\left| Q_i \right| = k$. The coefficient α_i (i = 1, 2,…, n) is represented in equation 4.

$$\alpha_i = \frac{\sum_{W \in Q_i} C_W P_w}{\sum_{W \in Q_i} P_w} \tag{4}$$

2.4 GYRO

A highly intensive application, GYRO was designed to benefit from distributed-memory supercomputer platforms using a grid-distribution scheme. Previous work on GYRO has outlined the importance of the grid-distribution scheme [1]. Using a eulerian scheme, GYRO must distribute the gyro-center distribution function $f(r, \tau, n_{tor}, \lambda, E)$, after it has been discretized to $f(i, j, n, k, e)$. The base of the distribution scheme with GYRO requires that all values of i and j be readily available on all processors. The values for n are distributed along the rows and the values for {e, k} are distributed along columns. Figure 1 provides a clear depiction of the data distribution scheme for GYRO.

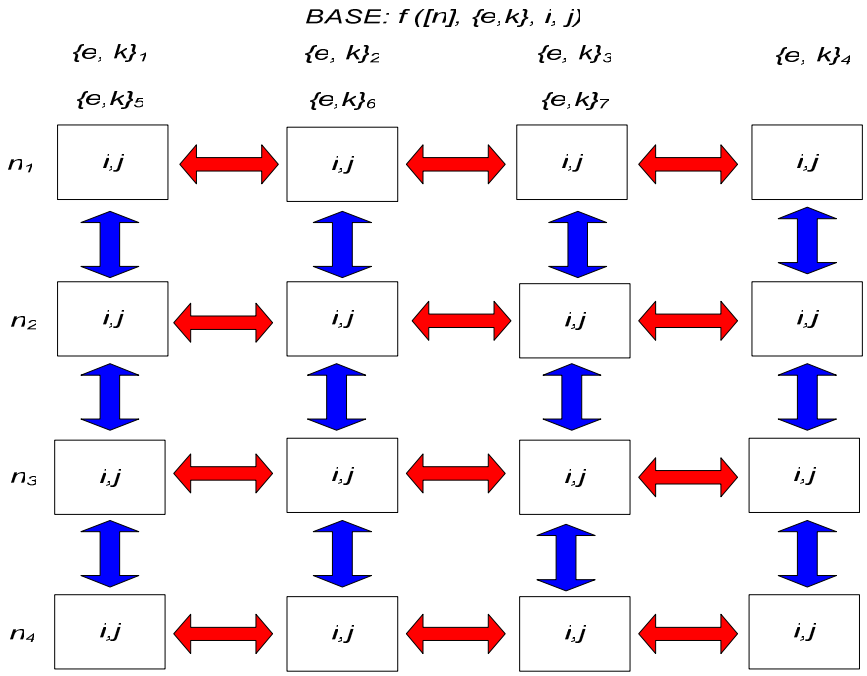

Figure 1. GYRO Base Grid Distribution Scheme

GYRO utilizes two transpose operations in order to communicate all values. A 3-index row transpose is utilized in order to place values {e,k} on processor rows. The 3-index row transpose results in i values being replaced by e on processor rows. Figure 1 provides for an illustration of the effect of the transpose.

2.4.1 Application Input

It is clear that the performance of an application is likely to vary in response to the baseline input parameters that are selected by the user. In order to provide a baseline for analysis, we utilize three widely available benchmarks for the GYRO application – the Waltz Standard Benchmark, the Cyclone Base Benchmark, and the Global Cyclone Base Benchmark. These three benchmarks provide inputs that mimic typical production characteristics for GYRO [3]. These benchmarks have been widely utilized in previous studies analyzing the performance characteristics of GYRO [1, 5, 12, 15]. Input for the application is based upon the following grid parameters: the toroidal grid, radial grid, pass grid, trap grid, energy grid, and blend grid.

2.4.1.1 Waltz Standard Benchmark

The Waltz Standard benchmark, B1-std, is a test case that consists of 500 timesteps. This benchmark uses kinetic electrons and electron collisions in computing the necessary physics data. In general, this benchmark consists of grids of size 6 x 140 x 4 x 4 x 8 x 6. The Waltz Standard benchmark is able to make use of a flux-tube 16-toroidal mode electrostatic case. This grid size is used because it represents typical minimum grid size requirements for a practical run of the application.

2.4.1.2 Cyclone Base Benchmark

The Cyclone Base Benchmark, B2-cy, is a test case that consists of 1250 timesteps. This benchmark uses kinetic electrons and electron collisions in computing the necessary physics data. In general, this benchmark consists of grids of size 6 x 128 x 4x 4 x 8 x 6. The grid size for the Cyclone base benchmark is slightly smaller than the Waltz Standard case; however, the number of timesteps of this benchmark is more than double of the Waltz Standard.

2.4.1.3 Global Cyclone Base Benchmark

The Global Cyclone Base Benchmark, B3-gtc, consists of 100 timesteps. This benchmark uses kinetic electrons and electron collisions in computing the necessary physics data. In general, this benchmark consists of grids of size 6 x 400 x 4 x 4 x 8 x 64. The radial grid size of this benchmark is more than triple the size of the Waltz Standard and Cyclone Base benchmarks, which results in a larger execution time.

Overall, the executing of GYRO on these four platforms provides for a thorough understanding of the performance of the application across different platforms utilizing various processor speeds, processors per node, memory configurations, and interconnection networks.

The three benchmarks provide for different guidelines for analyzing the performance characteristics of the application. GYRO is broken into eight kernels for Benchmarks B1-std and B2-cy. Six of the kernels are computationally intensive kernels that are executed sequentially within a subroutine utilized to control the second-order, implicit-explicit (IMEX) Runge Kutta (RK) integrator. Figure 2 illustrates the control flow of all eight kernels involved in the analysis and modeling of GYRO.

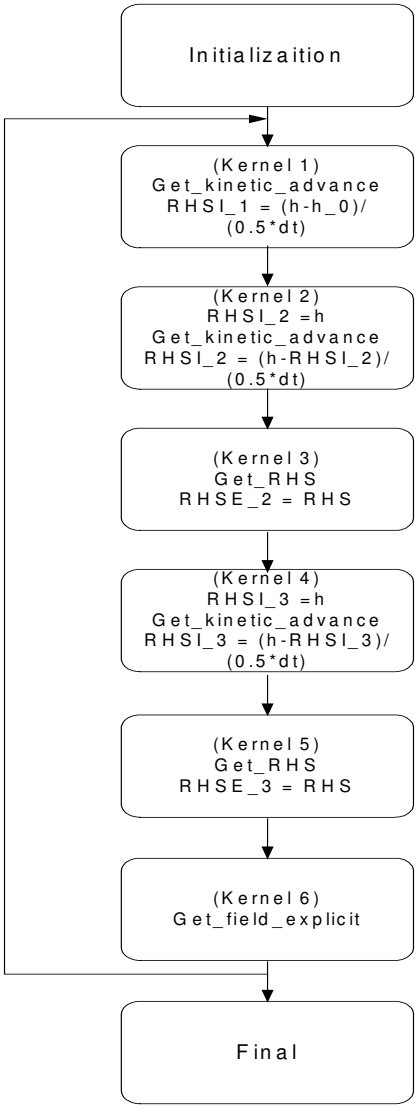

Figure 2. GYRO Control Flow for B1-std and B2-cy

The initialization kernel is used to process input data, generate variables, and compute several computational grids. The subroutine IMEX RK integrator contains the most computationally intensive kernels. Kernel 1 manages the implicit advance of fields and electron distributions for the initial distribution function h. Kernel 2 manages the implicit advance of fields and electron distributions after h has advanced in time using RHSI. Kernel 3 computes the RHS of the electron and ion GKEs for both periodic and nonperiodic boundary conditions. Kernel 4 computes the implicit advance of fields and electron distributions after h has advanced with respect to RHSE_2 and RHSI_2. Kernel 5 is then utilized to compute the newly needed RHS. Kernel 6 calls a sequence of routines to develop the Explicit Maxwell equation solution.

Kernel 1, Kernel 2, and Kernel 4 use the get_kinetic_advance subroutine throughout their calculations for the advance of fields and electron distributions. The get_kinetic_advance subroutine must perform several intensive derivatives for the calculation of the explicit portion within this subroutine.

Kernel 3 and Kernel 5 are based on the get_RHS subroutine. This routine requires extensive calculations of both the periodic and nonperiodic boundary conditions for the electron and ion global knowledge exchanges. Within get_RHS a mechanism for the collection of torodial mode numbers is provided and a parameterization of the poloidal angle is distributed to all processes. The collection and distribution of such parameters is completed through a column transpose operation that utilizes the MPI_ALLTOALL operation. The catch_blowup subroutine is utilized in the get_RHS routine in order to provide a boundary condition based on several electron and ion field coefficients. Therefore, the MPI_ALLREDUCE is utilized to provide for a collective reduction. To evaluate the Krook ion collision operator the do_neo_collision subroutine is utilized during get_RHS. This subroutine must compute several blending projections and distribute the maximum value to all processes.

Overall, these computational kernels require that various operators be available to all processes after discretization. GYRO makes use of the MPI_COMM_WORLD communicator as well as two new communicators, COMM1 and COMM2. These new communicators determine an appropriate number of subgroups based on the number of processors.

In the case of B3-gtc GYRO is decomposed into six kernels. As with B1-std and B2-cy there is an initialization kernel and a final kernel. For B3-gtc, there are 4 computation kernels

that are utilized in the Implicit-Explicit Runge-Kutta integrator for this benchmark. Figure 3 provides an illustration of the application control flow when utilized on this benchmark.

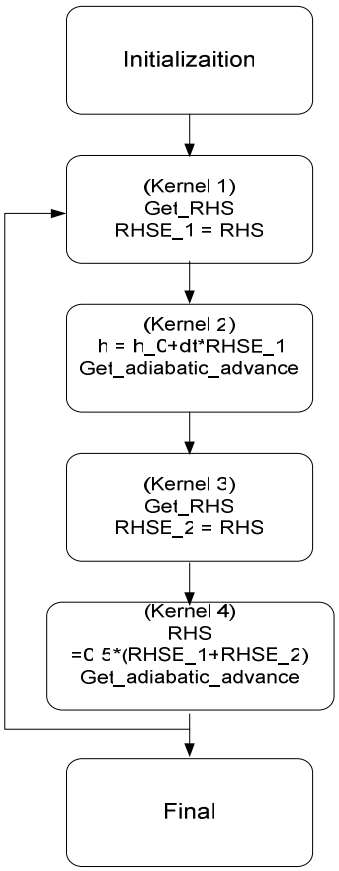

Figure 3. GYRO Control Flow for B3-gtc

As with B1-std and B2-cy, the initialization kernel is used to process input data, generate variable, and compute several computational grids. This kernel utilizes the same subroutine features up until the IMEX RK integrator executed. Kernel 1 manages the implicit advance of fields and electron distributions for the initial distribution function h. Kernel 2 is utilized to manage the explicit advance of fields during this integration method. Kernel 3 manages the implicit advance of fields and electron distributions after h has advanced in time using RHSE. Kernel 4 computes the implicit advance of fields and electron distributions after h has advanced with respect to RHSE_1 and RHSE_2.

3. PROCESSOR PARTITIONING

In this section, we analyze the performance characteristics of GYRO by examining the most effective processor partitioning scheme. We conduct an experimental analysis of the various platform environments to identify the application factors that impact the selection of the best number of processors per node (ppn) to use for execution of GYRO. Note that, in this section and subsequent sections, ppn stands for processors per node and N x M stands for N nodes with M processors per node.

This section will provide for a comparison of the performance characteristics of GYRO for B1-std and B2-cy providing for an overall comparison of the best-case and worst-case processor partitioning results. This detailed analysis of the processor partitioning scheme allows for attention to be given to the execution of GYRO on 32 processors for various M nodes and N ppn (M x N). Overall, these experiments address the impact that the high global communication of the application has on the performance across various platforms. For example, the 32 x 1 scheme shows more than 38% improvement in the execution time of GYRO in comparison to the 4 x 8 scheme on the P655 platform.

3.1 Processor Partitioning on Seaborg

In Table 2, we examine the results of different processor partitioning schemes for B1-std on NERSC Seaborg. The general trend on this platform shows that the execution time of the application increases as the processors per node increase. Therefore, the best-case partitioning scheme is illustrated when there is only one ppn. In addition, there is an increase in the ratio of communication to total execution time as the number of processors per node increases. The smallest communication to computation ratio occurs during the 32 x 1 scheme. When the ppn are increased to two there is an increase in the ratio of communication to total execution. This ratio shows a decrease again when the number of ppn is increased to 4. Overall, there is a more significant increase in the communication to total execution ratio if full-node usage is utilized.

Table 2. Processor Partitioning on Seaborg for B1-std

M x N	32 x 1	16 x 2	8 x 4	4 x 8	2 x 16
Runtime (s)	2076.094	2150.029	2132.258	2266.012	2584.013
Communication (s)	315.9064	384.1556	356.2706	438.1656	590.7759
Comm/Execution	17.9%	21.7%	20.1%	24.0%	29.6%
Initialization (s)	226.9509	227.851	228.9114	229.8697	244.3562
Kernel 1 (s)	179.4944	180.5818	182.684	192.4092	225.625
Kernel 2 (s)	182.8203	183.9135	188.2891	196.0743	233.0817
Kernel 3 (s)	557.3081	578.2349	574.2512	621.3676	701.2268
Kernel 4 (s)	183.3476	199.6244	187.2085	196.2591	232.5416
Kernel 5 (s)	550.3275	567.4831	568.1513	608.863	695.5358
Kernel 6 (s)	109.7601	124.5772	112.3565	116.5854	132.7252
Final (s)	58.35325	63.45799	60.31956	69.61948	69.40522

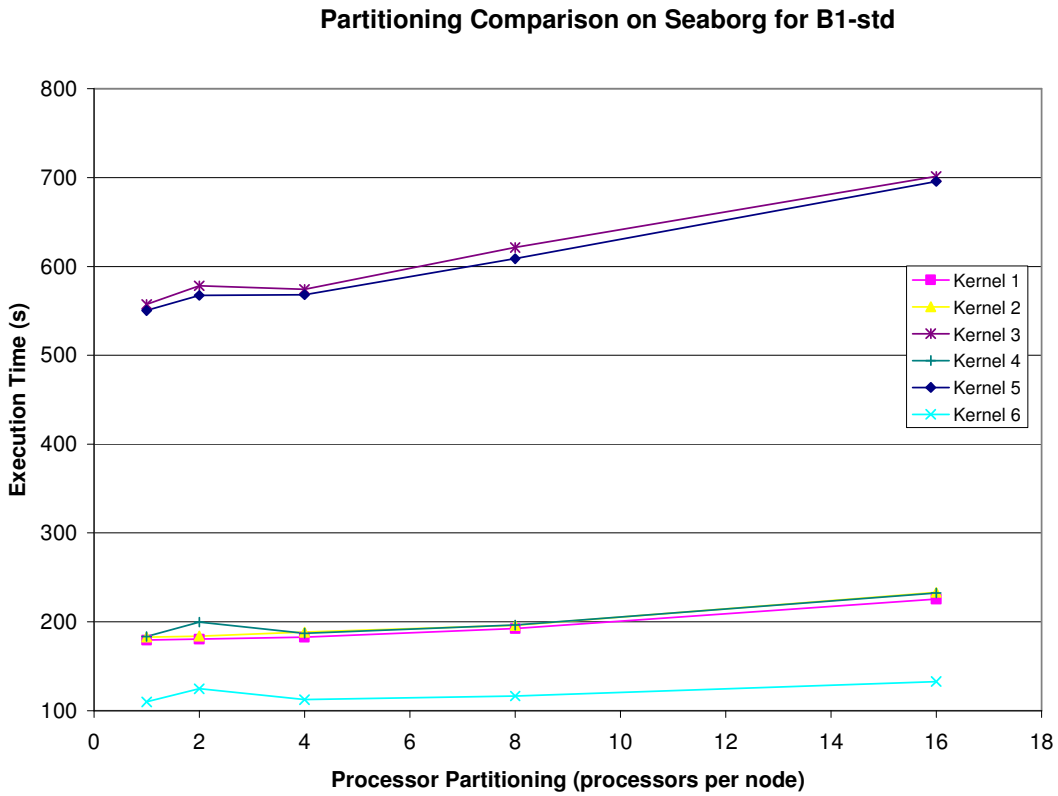

Figure 4. Partitioning Comparison on Seaborg for B1-std

In Figure 4, we see an overall comparison of the performance characteristics for kernels 1, 2, 3, 4, 5 and 6 on Seaborg. Kernels 1, 2, and 4 call the get_adiabatic_advance subroutine. As the number of ppn increase, the execution time of these kernels also increases. In the case of

kernel 4 and kernel 6, the execution time decreases for the case of 4 ppn as compared to 2 ppn; the execution time then continues to increase for up to 16 ppn. In all cases, the smallest execution time occurs with 1 ppn.

Kernel 3 and kernel 5 call the get_RHS subroutine that is used in the calculation of periodic and nonperiodic boundary conditions. This subroutine requires extensive global communication through the use of the MPI_Alltoall and MPI_Allreduce operations. It can be seen that the performance of this routine is highly inefficient for full-node usage on NERSC Seaborg. In the performance of kernel 3 and kernel 5 it can be seen that the execution pattern remains fairly constant from 2 ppn to 4 ppn, with a slight decrease in execution time for kernel 3. Figure 4 shows that there is a rather large increase in the execution time of these kernels as the ppn are increased to 16.

In Table 3, we examine the results of processor partitioning for B2-cy on Seaborg. B2-cy makes use of the same kernels as B1-std and although this is a different dataset it follows the same trend as B1-std with a large decrease in the execution time for smaller ppn. As with B1-std, the best-case execution results for the 32 x 1 scheme. The smallest communication to total execution ratio occurs during the 32 x 1 scheme. When the ppn are increased to two there is a slight increase in the communication to total execution ratio. The communication to total execution ratio decreases again when the ppn are increased to eight. There is a more significant increase in the communication to computation ratio if the full ppn are utilized. Communication increases by over 83% when comparing the best-case (32 x 1) and worst-case (2 x 16) processor partitioning schemes.

Figure 5 gives a comparison of the execution time of kernels 1, 2, 3, 4, 5 and 6 for B2-cy. There is a large increase in execution time as the ppn are increased from 1 to 2, but there is a slight decrease for 4 ppn. In evaluating kernel 6, it can be seen that utilizing 2 ppn yields almost as high an execution time as utilizing full-node usage. Kernel 6 utilizes the get_field_explicit subroutine that coordinates the calling of explicit functions to solve the Maxwell equations. This subroutine requires little communication and is more computation-intensive. Kernel 3 and kernel 5 exhibits a general trend that shows the execution time of the kernels increasing as the ppn are increased. In addition, it can be seen that the overall performance of these two kernels are very close despite the computation of different components.

Table 3. Processor Partitioning on Seaborg for B2-cy

M x N	32 x 1	16 x 2	8 x 4	4 x 8	2 x16
Runtime (s)	4976.312	5152.328	5169.782	5396.069	6176.221
Communication (s)	757.821	859.504	859.857	979.114	1392.583
Comm/Execution	18.0%	20.02%	19.95%	22.17%	29.11%
Initialization (s)	74.714	75.280	74.7910	75.464	79.736
Kernel 1 (s)	480.2385	485.8304	490.882	513.9064	611.771
Kernel 2 (s)	484.8676	489.9996	497.0721	517.9891	612.771
Kernel 3 (s)	1398.887	1435.028	1473.165	1547.15	1749.071
Kernel 4 (s)	485.4982	532.8427	493.445	518.0042	617.448
Kernel 5 (s)	1398.071	1427.84	1471.023	1531.212	1742.083
Kernel 6 (s)	330.792	374.909	334.178	346.182	386.263
Final (s)	172.399	178.185	181.574	184.746	200.045

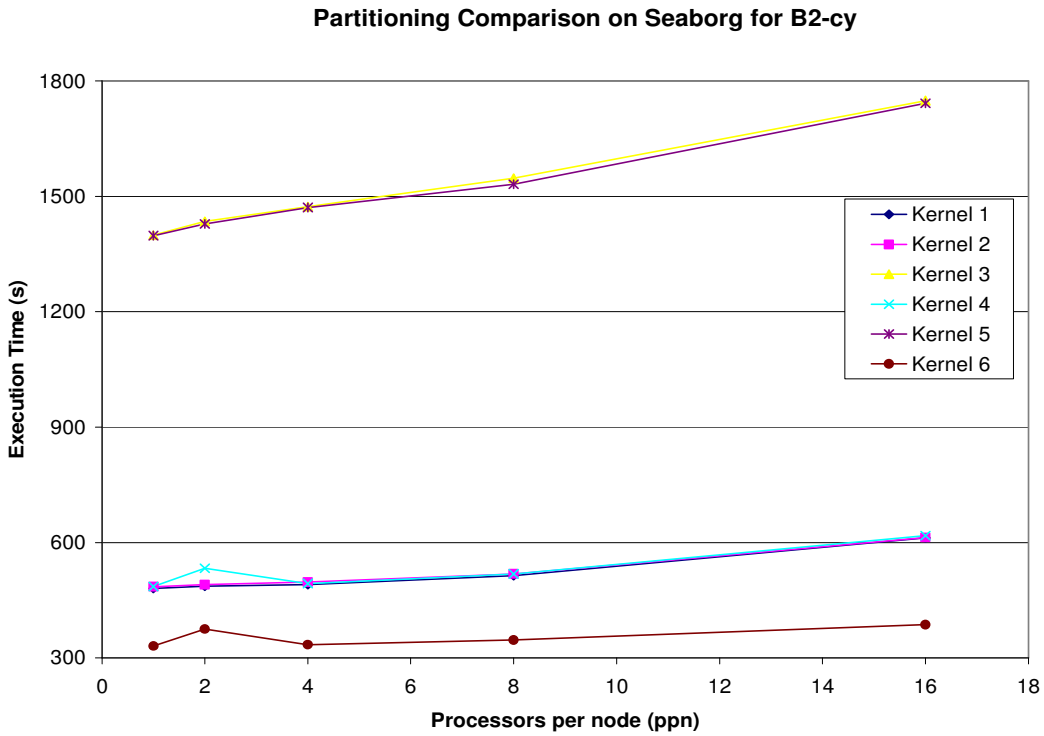

Figure 5. Partitioning Comparison on Seaborg for B2-cy

3.2 Processor Partitioning on the DataStar P655

In Table 4, we examine the results of different processor partitioning schemes for B1-std on the DataStar P655 platform. The general trend on this platform shows that the execution time of the

application increases as the ppn increase. The most significant change in this platform can be seen when comparing the communication times for the 32 x 1 and the 4 x 8 scheme. There is a more than 79% increase in communication as the ppn are increased from 1 to 8. In addition, the execution time increases by more than 38% between the 32 x 1 and 4 x 8 schemes.

Table 4. Processor Partitioning on DataStar P655 for B1-std

M x N	32 x 1	16 x 2	8 x 4	4 x 8
Runtime (s)	564.29	592.1824	631.00	780.87
Communication (s)	77.66	79.89	92.77	139.38
Comm/Execution	15.96%	15.59%	17.24%	21.72%
Initialization (s)	70.74923	73.3364	75.0167	91.08963
Kernel 1 (s)	47.9089	53.17531	56.51523	65.46934
Kernel 2 (s)	46.72206	51.99831	57.24473	66.97979
Kernel 3 (s)	145.6257	147.9556	156.9033	206.0236
Kernel 4 (s)	47.73898	52.96069	57.99906	65.94845
Kernel 5 (s)	144.9817	147.6143	156.4767	203.3095
Kernel 6 (s)	32.38877	36.2532	40.2827	46.98644
Final (s)	25.10579	25.26396	26.04492	28.71204

Figure 6 illustrates the steady increase in the execution time of kernels 1, 2, and 3 as the ppn are increased. There is a larger increase in the execution time when the processors per node are increased from 1 to 2. The get_kinetic_advance subroutine requires very little communication and is mainly used to calculate gyroaverages and perform identity operations for the electron and ion distributions. The get_RHS subroutine requires a tremendous amount of global communication. Kernel 3 and kernel 5 utilize this subroutine and Figure 6 illustrates the inefficiency of using more than 2 ppn when evaluating these kernels. As the ppn increase from 1 to 2 there is a slight increase in the execution time. The increase in the execution time grows for 4 ppn and using 8 ppn tends to be inefficient.

Table 5 illustrates the results of effective processors partitioning on GYRO for B2-cy. GYRO's execution time is more than 30% faster when executed using 1 ppn. The communication time grows over 60% slower when the full 8 ppn are utilized. It is interesting to note that the communication time does not change that much from 1 ppn to 2 ppn. The overall runtime of the application does increase.

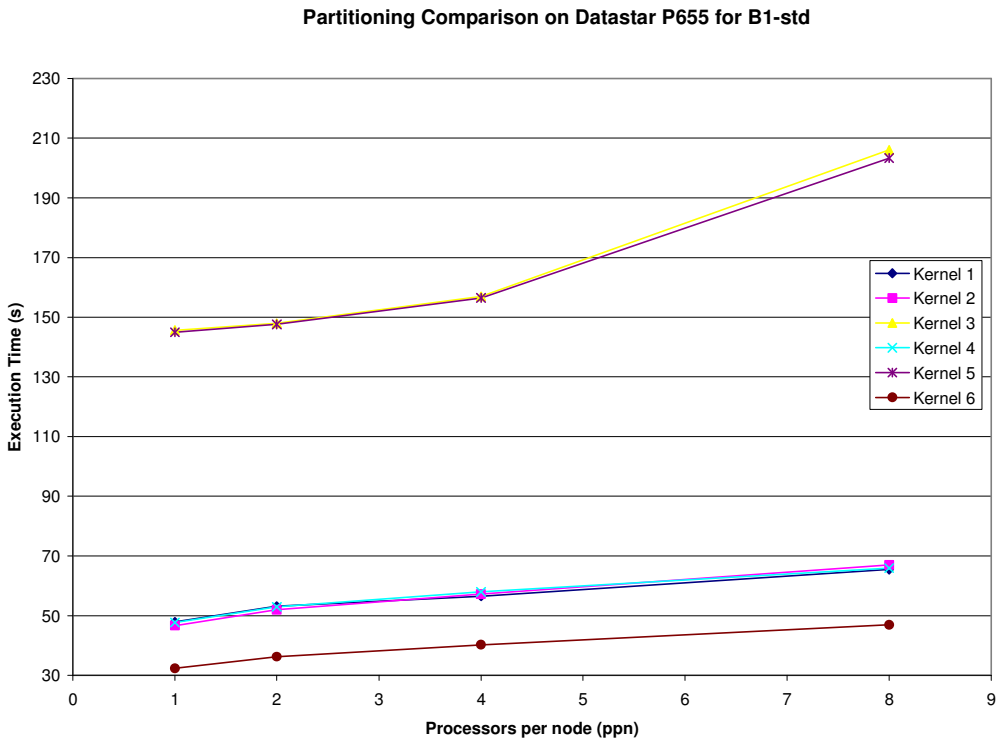

Figure 6. Partitioning Comparison on DataStar P655 for B1-std

Table 5. Processor Partitioning on DataStar P655 for B2-cy

M x N	32 x 1	16 x2	8 x 4	4 x 8
Runtime (s)	1291.09	1355.09	1461.64	1729.19
Communication (s)	163.12	162.84	186.61	261.21
Comm/Execution	14.46%	13.75%	14.63%	17.79%
Initialization (s)	18.79627	19.65377	20.08025	21.50701
Kernel 1 (s)	135.3301	148.2993	159.2199	182.5569
Kernel 2 (s)	133.8043	147.4374	160.35	182.5569
Kernel 3 (s)	345.3141	353.1002	381.9275	465.6942
Kernel 4 (s)	138.3458	151.1742	161.4479	183.3948
Kernel 5 (s)	344.245	352.4977	382.0242	465.1057
Kernel 6 (s)	107.1449	114.3494	122.6825	141.2738
Final (s)	59.03264	58.62708	62.02006	70.52438

Figure 7 shows a comparison of kernels 1, 2, 3, 4, 5 and 6 as the ppn increase when GYRO is executed on 32 processors for B2-cy. On the P655, we see a steady increase in execution time for these kernels as the ppn increase. There is a larger increase in the execution

time for kernels 1, 2, and 4 when going from 1 to 2 ppn. The increase in execution from 2 ppn to 4 is not as large.

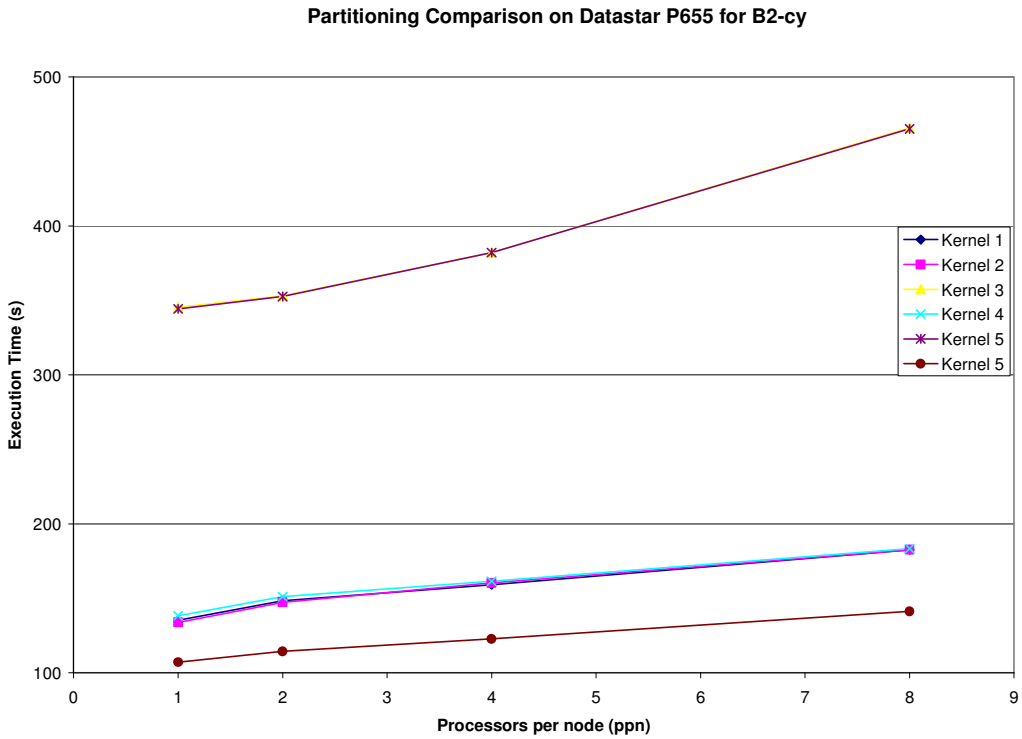

Figure 7. Partitioning Comparison on DataStar P655 for B2-cy

For kernels 3 and 5 the execution times are very close together. This is a strong indication of the get_RHS subroutine's timestep stability on the P655. The federation interconnect yields a slight increase in the execution times for the subroutine when the ppn increase from 1 to 2. There is a more dramatic increase when the ppn reach full capacity. The global communication required in this subroutine is not largely affected when adding one additional processor to the node. The partitioning scheme proves that full node usage is not the most efficient method for executing GYRO when evaluating the performance of these two kernels.

3.3 Processor Partitioning on the DataStar P690

Table 6 provides for performance metrics relating to GYRO for B1-std on the DataStar P690 platform. It can be seen that utilizing 8 ppn yields a 36% improvement in the execution time of GYRO for B1-std in comparison to using the full 32 ppn. Communication is over 65% faster

using only 4 ppn on this platform. Theoretically, 1 ppn would yield the best performance for 16 to 128 processors but the P690 only has 4 nodes and therefore experimental results can not be obtained for the execution of GYRO utilizing the smallest number of processors, the16 x 1 scheme. The results in this section do show that overall the 8 ppn, or the smallest number of ppn, yields the most efficient performance for the application. Furthermore, on the P690 there are only 4 nodes available to users, each node consists of 32 ppn. To evaluate the performance of different processor partitioning schemes 8 ppn can be used as the smallest number of ppn. This allows for all four nodes to be utilized with the smallest number of ppn. Utilizing 1 ppn on this platform would not be executable on GYRO because the application must be run on multiples of 16 processors for B1-std and B2-cy.

Table 6. Processor Partitioning on DataStar P690 for B1-std

M x N	4 x 8	2 x 16	1 x 32
Runtime (s)	681.94	747.43	928.69
Communication (s)	130.20	162.71	216.11
Comm/Execution	0.235972	0.278271	0.303282
Initialization (s)	74.55149	78.21067	128.6658
Kernel 1 (s)	61.42072	64.21788	83.03325
Kernel 2 (s)	61.54672	66.12146	82.99894
Kernel 3 (s)	176.824	207.8554	226.9632
Kernel 4 (s)	65.91273	65.38923	84.16051
Kernel 5 (s)	166.1609	188.5733	225.3313
Kernel 6 (s)	44.08134	44.15997	56.67782
Final (s)	26.78493	27.07489	30.43768

Figure 8 provides for a comparison of kernels 1, 2, 3, 4, 5, and 6 when executed on the P690 for various processor partitioning schemes. It can be seen that kernel 4 and kernel 6 exhibit the same trend line. It is interesting to note that both of these kernels occur after the kernels that utilize the get_kinetic_advance subroutine. The communication that occurs within the get_kinetic_advance subroutine might have an effect on the performance of the subsequent kernels. Increasing the processors per node from 8 to 16 has very little effect on the overall execution of these two kernels.

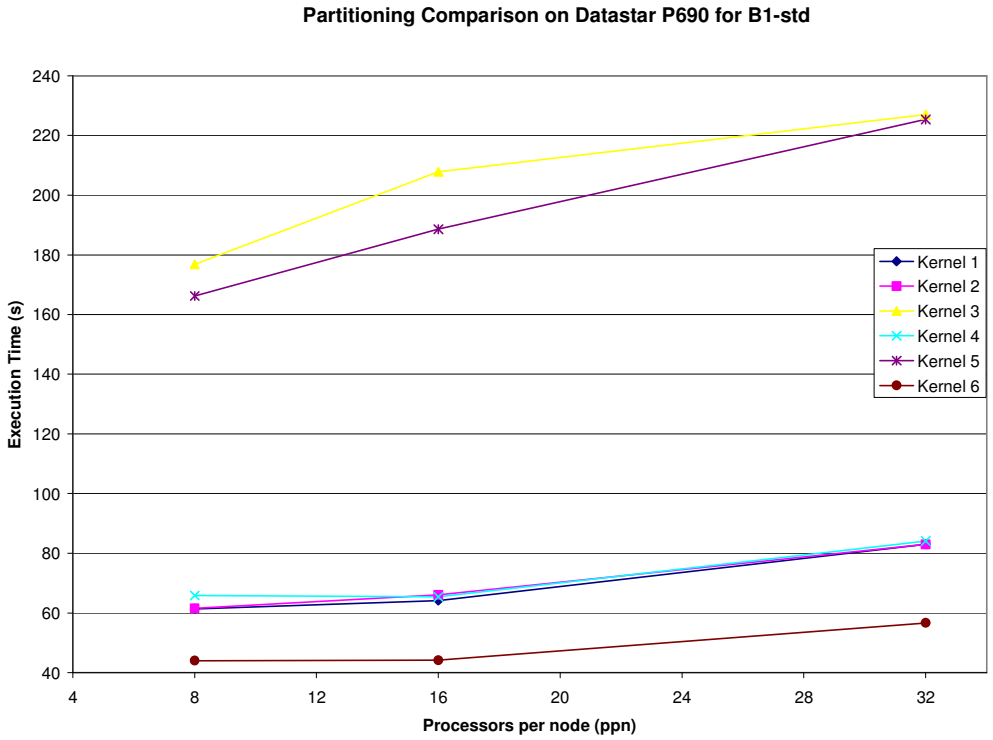

Figure 8. Partitioning Comparison on DataStar P690 for B1-std

Both kernel 3 and kernel 5 are based upon the get_RHS subroutine; however, they exhibit slightly different trend lines on this platform. Kernel 3 has a sharp increase in execution time as the ppn are doubled, but the increase is less drastic when the ppn increase to 32. Kernel 5 shows a more steady increase in execution time as the ppn are increased.

Table 7 illustrates the results of effective processors partitioning of GYRO for B2-cy on the DataStar P690 platform. The execution time of GYRO is more than 40% faster when executed on 32 processors utilizing 8 ppn. The full node usage provides for significantly slower performance. The communication time grows over 78% slower when the full 32 ppn are utilized. Even doubling the ppn from 8 to 16 causes a 35% increase in communication time. The overall runtime of the application does increase however during this setting.

Table 7. Processor Partitioning on DataStar P690 for B2-cy

M x N	4 x 8	2 x 16	1 x 32
Runtime (s)	1431.40	1633.83	1997.42
Communication (s)	208.36	281.94	371.97
Comm/Execution	17.0%	21.1%	22.9%
Initialization (s)	18.3402	19.29219	23.06745
Kernel 1 (s)	159.8138	178.8529	222.1844
Kernel 2 (s)	160.1895	180.1864	222.1305
Kernel 3 (s)	361.5661	431.7177	521.2546
Kernel 4 (s)	166.537	184.4917	225.9845
Kernel 5 (s)	357.1083	427.0129	522.7713
Kernel 6 (s)	118.3986	134.2507	165.1405
Final (s)	59.85066	62.85298	69.6818

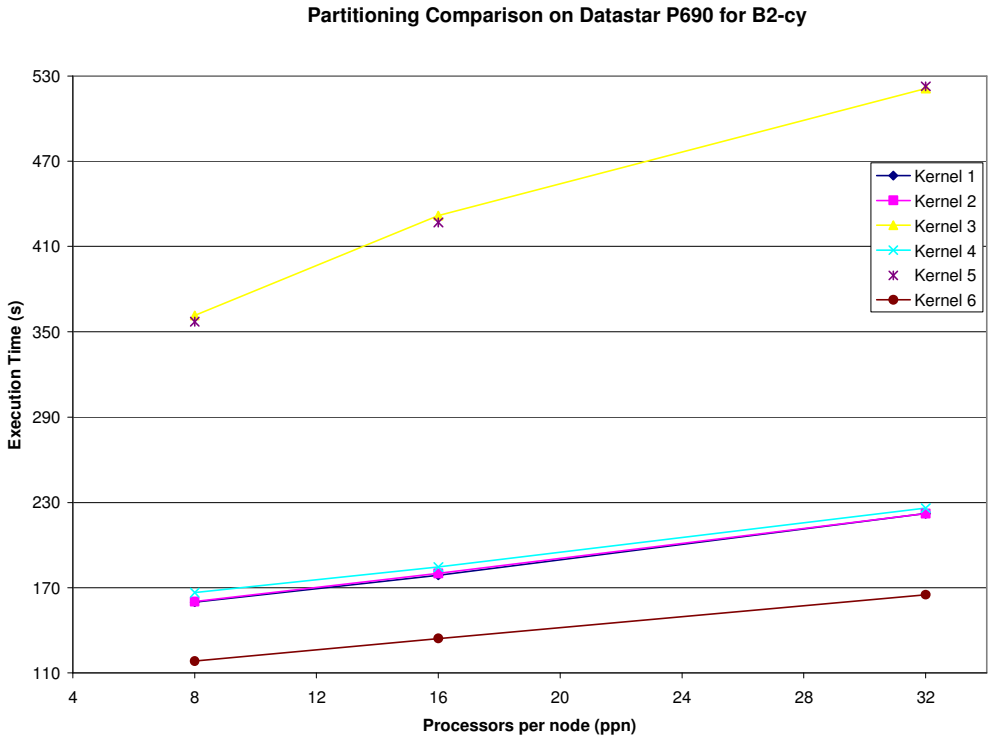

Figure 9. Partitioning Comparison on DataStar P690 for B2-cy

Figure 9 provides for a comparison of kernels 1, 2, 3, 4, 5, and 6 when executed on the P690 for B2-cy. It is interesting to note that the performance of B2-cy increases linearly as the ppn are increased for kernels 1, 2, 4, and 6. In Figure 9, we see that B1-std did not exhibit this

behavior for kernels 1, 2, 4, and 6. Kernel 1 and kernel 2 exhibit similar execution results. Kernel 4 results in a slightly larger execution time even though it is based off of the get_kinetic_advance subroutine that kernel 1 and kernel 2 utilize. The slight increase in execution time is likely the result of the communication that occurs in kernel 3 between kernel 2 and kernel 4. Kernel 6 does a lot of computation and global communication and therefore on this benchmark follows an almost linear growth in execution time as the ppn are increased.

The trends in execution time for kernel 3 and kernel 5 are fairly consistent. As the ppn are double from 8 to 16 there is a very large increase in the performance of these two kernels. When the ppn are doubled again from 16 to 32 the execution time increases significantly as well. As noted earlier, 8 is the smallest number of ppn that can be utilized on this platform.

3.4 Processor Partitioning on Jacquard

In Table 8, we examine the results of different processor partitioning schemes for B1-std on the NERSC Jacquard platform. This platform is a cluster system and therefore consists of only 2 ppn. Previous results on other platforms have shown that there is only a slight increase in performance from 1 to 2 ppn. In this section, we see that the improvement is only slight, but utilizing the 32 x 1 scheme yields better performance than the 16 x 2 scheme. The improvement in performance from the 32 x 1 scheme to the 16 x 2 is 0.22%.

Table 8. Processor Partitioning on Jacquard for B1-std

M x N	32 x 1	16 x 2
Runtime (s)	690.77	692.4799
Communication (s)	173.87	174.64
Comm/Execution	25.1%	25.2%
Initialization (s)	62.80644	61.65928
Kernel 1 (s)	50.61698	51.37382
Kernel 2 (s)	51.83867	52.50066
Kernel 3 (s)	208.0138	210.6411
Kernel 4 (s)	51.79123	52.51419
Kernel 5 (s)	206.9256	209.6249
Kernel 6 (s)	36.32475	32.58252
Final (s)	16.53471	11.56793

Figure 10 shows the slight variations in kernels 1, 2, 3, 4, 5, and 6 as the ppn are increased. There is a small increase in the execution times for each kernel. In the case of kernel 6, there is a small decrease in the execution time.

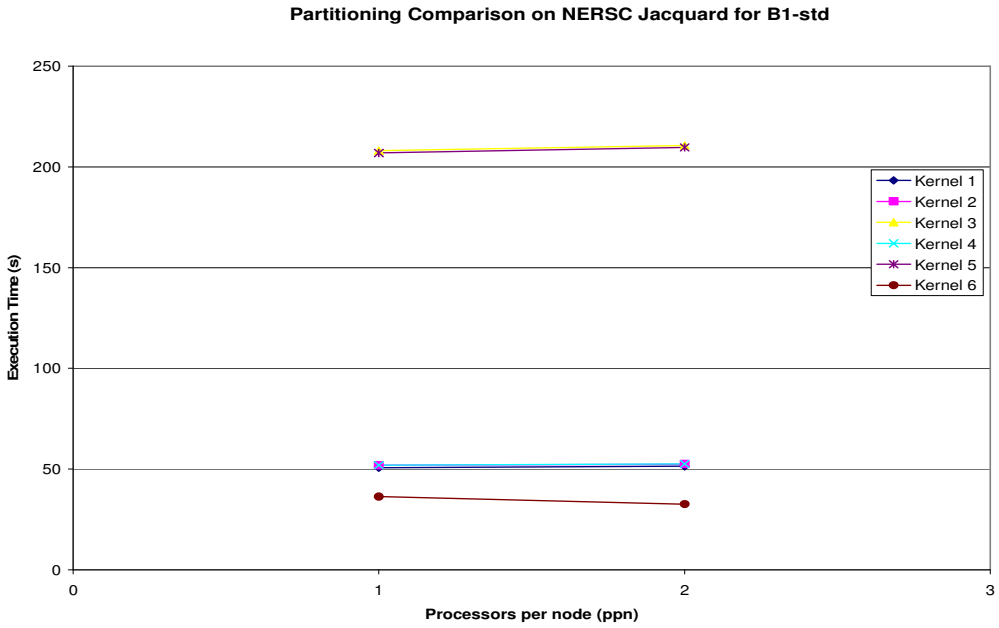

Figure 10. Partitioning Comparison on Jacquard for B1-std

Table 9 provides for an overview of the processor partitioning schemes for B2-cy on Jacquard. There are slight increases in the communication to total execution ratio as the ppn are increased from 1 to 2. Overall, there is only a small increase of 1.5% when going from half (32 x 1) to full (16 x 2) node usage.

Table 9. Processor Partitioning on Jacquard for B2-cy

M x N	32 x 1	16 x 2
Runtime (s)	1622.72	1631.916
Communication (s)	288.795	293.2026
Comm/Execution	17.8%	18.0%
Initialization (s)	22.25453	22.57371
Kernel 1 (s)	146.2832	144.7444
Kernel 2 (s)	148.9492	146.5725
Kernel 3 (s)	494.22	493.9729
Kernel 4 (s)	150.1747	157.0103
Kernel 5 (s)	494.1128	490.4124
Kernel 6 (s)	102.3926	120.8982
Final (s)	41.16599	42.14557

Figure 11 illustrates the trends that occur for B2-cy on Jacquard. Overall, there are fairly small increases in the execution time of the kernels as the ppn are increased from 1 to 2. The largest increase seems to occur for kernel 6 as it grows by 15% from half to full-node usage.

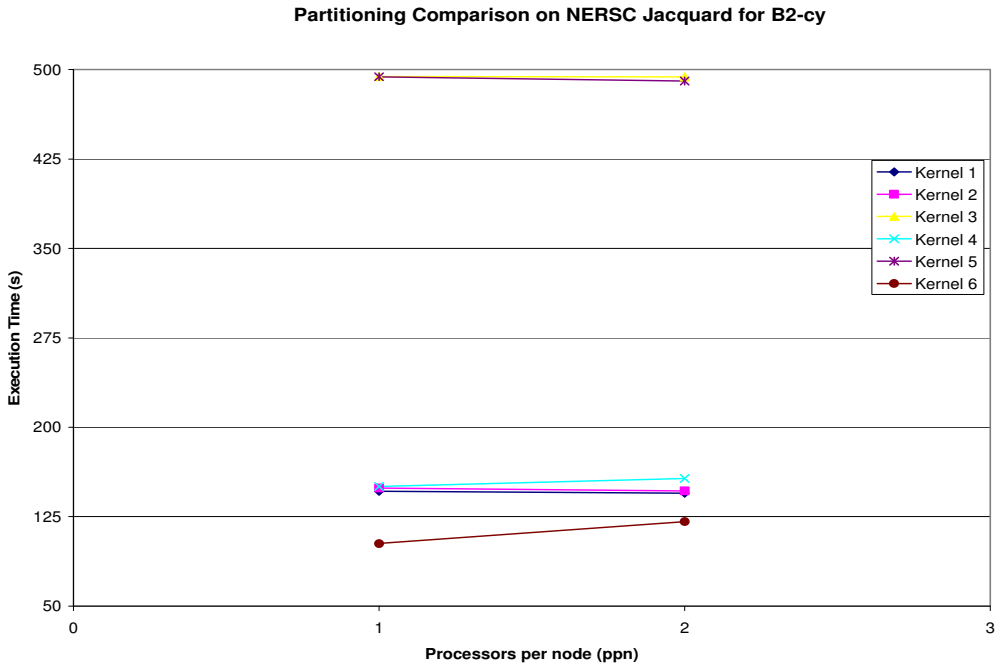

Figure 11. Partitioning Comparison on Jacquard for B2-cy

3.5 Summary

Overall, it can be seen that processor partitioning is a necessary component of performance analysis. The performance of GYRO depended largely on the global communication required by several of its computational kernels. Processor partitioning was able to determine the efficient processor partitioning configurations that would lead to performance losses for GYRO. This section has shown that utilizing 1 ppn or the smallest number of ppn greatly reduces the communication required by GYRO on the shared memory symmetric multiprocessors. Through processor partitioning we were able to determine the most effective processor partitioning scheme for executing GYRO for further analysis. The global communication required by GYRO requires constant broadcasting. This constant communication results in congestion because the intra-node communication is too high. Utilizing the smallest number of ppn decreases intra-node communication and results in overall performance improvements for the GYRO application.

4. PERFORMANCE MODELING

4.1 Performance Prediction

In this section, we present experimental results that show the effectiveness of using kernel coupling for performance prediction across the four supercomputing platforms. In order to yield a baseline of comparison, the kernel coupling prediction method is compared to a common method of summing the average execution time of the individual kernels together. The summation method is representative of when there is no interaction between adjacent kernels in an application ($C_{ij} = 1$). Therefore, for B1-std and B2-cy, two-kernel, three-kernel, and five-kernel predictions are compared to the summation method. For B3-gtc, two-kernel and three-kernel predictions are compared to the summation method.

4.1.1 Performance Prediction on Seaborg

Figure 12 provides for a graphical comparison of the average errors in prediction using summation, 2-kernel, 3-kernel, and 5-kernel coupling. Also, Table 10 gives an overall comparison of GYRO's actual execution time in comparison to performance prediction using summation and the various kernel coupling chains. Overall, in each case the coupling prediction method performs better that the summation method. Performance prediction for 16 processors on B1-std shows that 2-kernel prediction yields the most accurate result (4.41%).

Table 10 provides for a more detailed comparison of the performance prediction error rates on Seaborg for 16 to 128 processors. Overall, 2-kernel and 3-kernel coupling do not have error estimates larger than 5 % across all processors. 5-kernel coupling results in larger error percentages but in all cases the 5-kernel coupling prediction is more accurate than summation. Table 11 and Table 12 present detailed error percentages for B2-cy and B3-gtc.

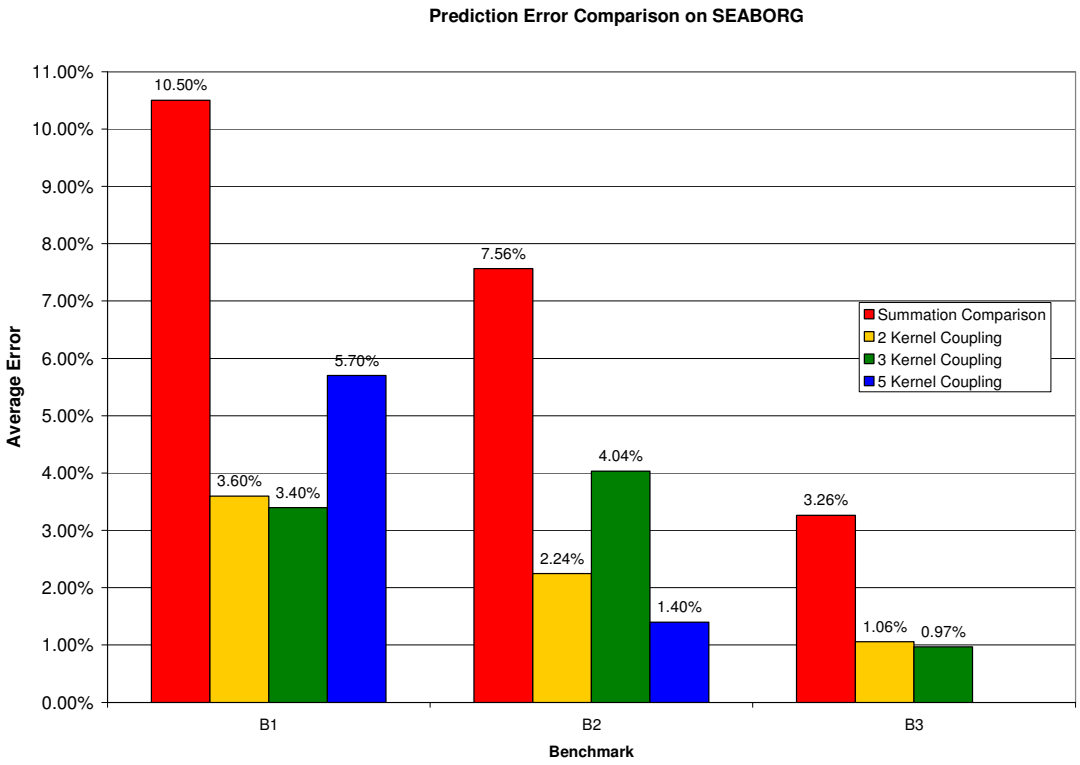

Figure 12. Prediction Comparison on Seaborg

Table 10. Seaborg Performance Prediction for B1-std

Processors	16	32	48	64	96	128
Actual Exe. Time (s)	4169.87	2077.37	1408.86	1062.55	726.49	547.94
Summation Error	6.60%	10.39%	11.92%	11.41%	10.62%	12.07%
2-k Error	4.41%	4.41%	2.62%	4.02%	3.08%	3.05%
3-k Error	4.80%	3.36%	2.80%	4.45%	3.23%	1.73%
5-k Error	4.39%	5.31%	0.86%	7.10%	9.89%	6.67%

Table 11 provides for detailed results of performance prediction on NERSC Seaborg for B2-cy. The summation method results in a much larger error for all processors. 2-kernel and 5-kernel coupling are the most accurate prediction methods on this benchmark. 3-kernel coupling results in slightly larger predictions for 48 and 96 processors, but it has less than 4% error when predicting for 16, 32, 64, and 128 processors. 5-kernel coupling has less than 3.5% error for all processors.

Table 11. Seaborg Performance Prediction for B2-cy

Processors	16	32	48	64	96	128
Actual Exe. Time (s)	12342.43	6170.35	4074.43	3031.09	2155.69	1661.53
Summation Error	7.38%	8.21%	9.45%	3.05%	5.83%	11.47%
2-k Error	2.86%	1.79%	1.03%	1.26%	6.01%	0.50%
3-k Error	1.63%	1.63%	8.93%	1.21%	7.39%	3.42%
5-k Error	0.35%	1.49%	1.36%	1.42%	0.32%	3.45%

Table 12 shows performance prediction results for B3-gtc on NERSC Seaborg. The summation method results in error estimates that are more than 100% larger for all processors. In the case of 3-kernel coupling, the largest error occurs for 256 processors, but this percentage is still less than 2.

Table 12. Seaborg Performance Prediction for B3-gtc

Number of Processors	64	128	256
Actual Execution Time (s)	5597.05	2777.75	1433.134
Summation Error	2.45%	1.45%	5.89%
2-kernel Error	0.96%	0.06%	2.16%
3-kernel Error	0.98%	0.12%	1.81%

4.1.2 Performance Prediction on the DataStar P655

In this section, we present experimental results that show the effectiveness of using kernel coupling for performance prediction on the SDSC DataStar P655 platform (Figure 13). In order to yield a baseline of comparison, the kernel coupling prediction method is compared to a common method of summing the average execution time of the individual kernels together. For B1-std and B2-cy, two-kernel, three-kernel, and five-kernel predictions are compared to the summation method. For B3-gtc, two-kernel and three-kernel predictions are compared to the summation method.

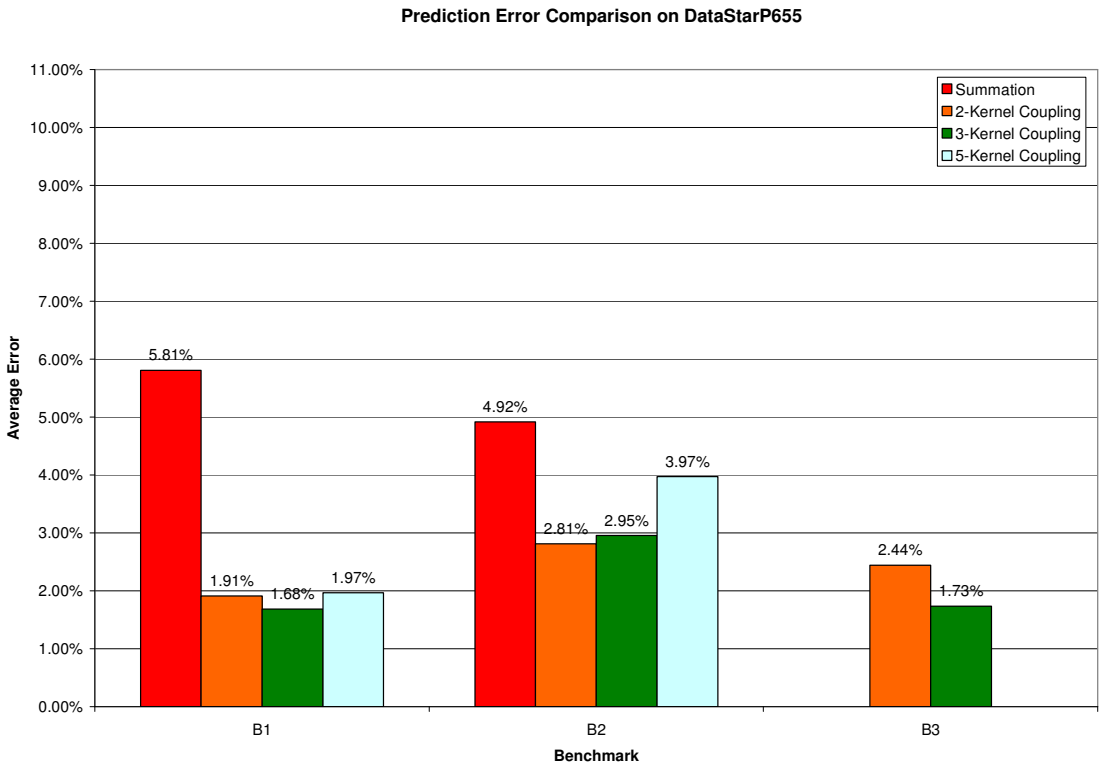

Figure 13. Prediction Comparison on DataStar P655

Table 13 provides for a more detailed comparison of the performance prediction error rates on the DataStar P655 for 16 to 128 processors. Overall, 2-kernel and 3-kernel coupling do not have error estimates larger than 5 % across all processors. 5-kernel coupling results in only slightly larger error percentages but in all cases the kernel coupling prediction methods are more accurate than summation. The largest prediction error in all four methods occurs at 128 processors. The summation method's error percentage is almost twice as large as the 2-kernel prediction method.

Table 13. DataStar P655 Performance Prediction for B1-std

Processors	16	32	48	64	96	128
Exe. Time	1546.46	767.69	521.17	387.28	273.54	206.46
Summ. Error	0.17%	7.30%	5.37%	3.40%	5.45%	13.16%
2-k Error	0.68%	0.54%	1.34%	2.18%	2.14%	4.58%
3-k Error	0.73%	0.96%	1.77%	0.21%	0.74%	7.68%
5-k Error	0.17%	3.33%	0.87%	0.69%	0.32%	6.75%

Table 14 presents detailed performance prediction error percentages for B2-cy on the DataStar P655. It can be seen that there are slightly larger error estimates produced by the kernel coupling methods on this benchmark. Overall, kernel coupling continues to predict the execution time more accurately than the summation method. The only case in which the summation method performs better is when B2-cy is executed on 32 processors.

Table 14. DataStar P655 Performance Prediction for B2-cy

Processors	16	32	48	64	96	128
Exe. Time	3466.77	1710.88	1181.54	877.30	609.83	477.42
Summ. Error	2.32%	2.69%	8.57%	5.18%	3.17%	7.58%
2-k Error	1.08%	0.64%	12.44%	0.02%	0.61%	2.06%
3-k Error	0.20%	0.81%	12.07%	0.59%	1.25%	2.82%
5-k Error	1.16%	4.47%	12.45%	0.35%	0.39%	3.44%

Table 15 presents detailed performance prediction percentages for B3-gtc on DataStar P655. Overall, all three methods do not result in more than 4% error. The summation method predicts with over 3% error as the number of processors are increased. The 3-kernel prediction method yields the best performance on this benchmark.

Table 15. DataStar P655 Performance Prediction for B3-gtc

Number of Processors	64	128	256
Actual Execution Time	2366.85	1216.80	614.92
Summation Error	3.18%	3.63%	3.57%
2-kernel Error	2.11%	1.61%	3.60%
3-kernel Error	3.16%	0.37%	1.67%

4.1.3 Performance Prediction on the DataStar P690

In this section, we present experimental results that show the effectiveness of using kernel coupling for performance prediction on the SDSC DataStar P690 platform (Figure 14). In order to yield a baseline of comparison, the kernel coupling prediction method is compared to a common method of summing the average execution time of the individual kernels together. For B1-std and B2-cy, two-kernel, three-kernel, and five-kernel predictions are compared to the

summation method. The performance prediction results indicate that the 2-kernel coupling method gives the most accurate prediction for both B1-std and B2-cy. The average error for 2-kernel coupling is 5.47% for both B1-std and B2-cy.

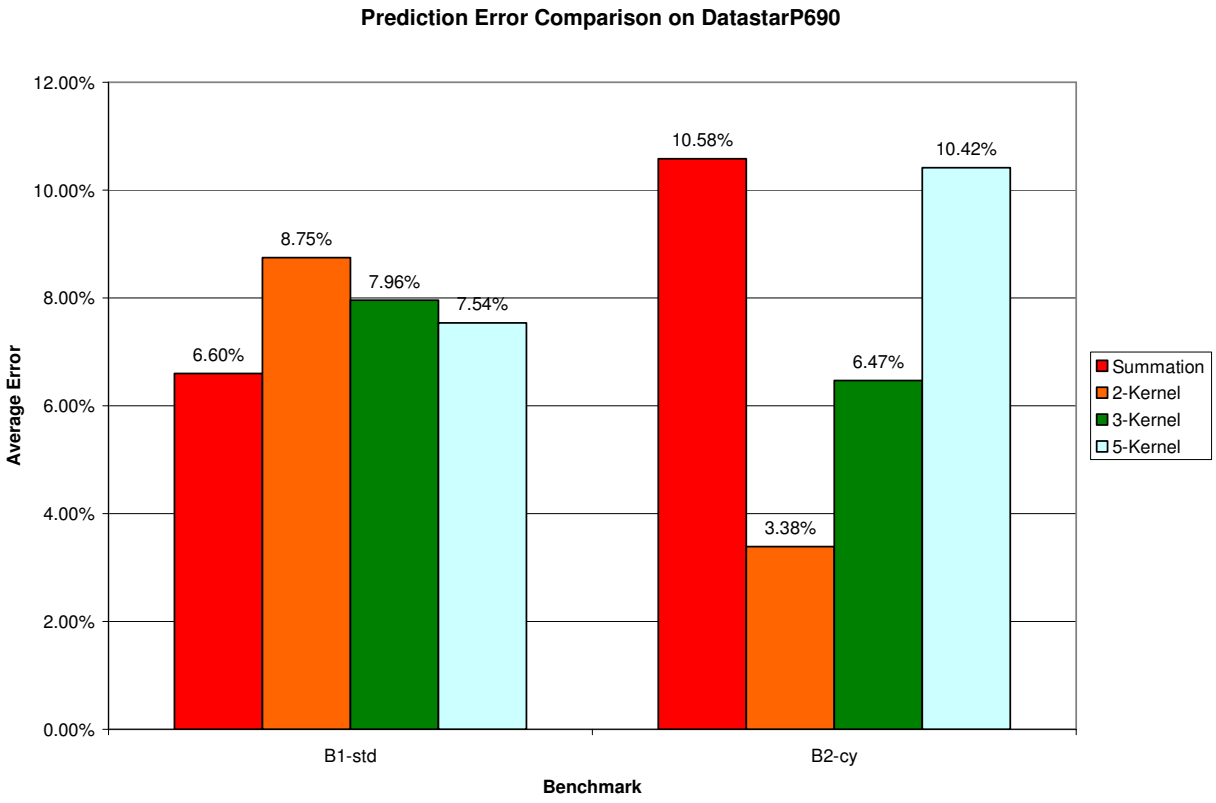

Figure 14. Prediction Comparison on DataStar P690

Table 16 provides detail performance prediction results for B1-std on DataStar P690. In the case of B1-std, the summation method provides for the smallest average error. This is largely because the 2-kernel method provides for a large error of over 15% when evaluated for 16 processors. In all other cases of comparison for B1-std 2-kernel coupling is more accurate than the summation method.

Table 16. DataStar P690 Performance Prediction for B1-std

Processors	16	32	64	96	128
Exe. Time	1418.26	815.61	509.53	353.75	271.72
Summ. Error	2.73%	6.50%	10.57%	15.77%	18.38%
2-k Error	15.11%	3.89%	5.94%	9.89%	1.70%
3-k Error	7.48%	3.83%	8.85%	12.35%	16.31%
5-k Error	2.98%	7.42%	9.26%	7.86%	18.91%

In the case of B2-cy (Table 17), the 2-kernel method provides for the more accurate prediction method. For most test cases, 2-kernel coupling prediction errors are less than 5%. When the 2-kernel prediction method is used for predicting the execution time on 128 processors the error is just over 6%, but it is almost half the error of summation for this processor count. In all cases for B2-cy, the kernel coupling prediction methods provide for more accurate average predictions when compared to summation.

Table 17. DataStar P690 Performance Prediction for B2-cy

Processors	16	32	64	96	128
Exe. Time	3365.86	1943.42	1050.57	760.05	601.85
Summ. Error	8.73%	5.24%	11.06%	14.80%	12.05%
2-k Error	1.57%	1.39%	3.28%	4.41%	6.14%
3-k Error	1.56%	5.77%	2.62%	7.04%	15.26%
5-k Error	1.24%	0.76%	4.65%	11.05%	34.39%

4.1.4 Performance Prediction on the Jacquard

In this section, we present experimental results that show the effectiveness of using kernel coupling for performance prediction on the NERSC Jacquard platform. The performance prediction results indicate that overall the kernel coupling methods are more accurate than the traditional method of summation. For both B1-std and B2-cy 3-kernel coupling yields the most accurate prediction

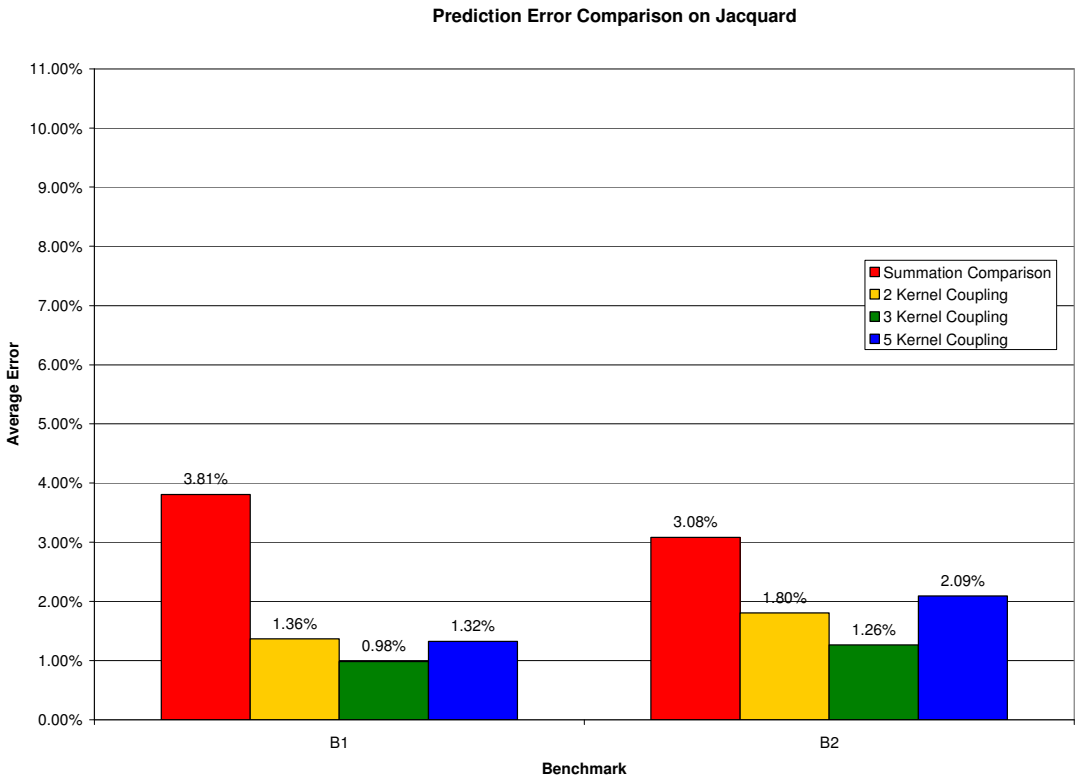

Figure 15. Prediction Comparison on Jacquard

Figure 15 shows that on average 3-kernel coupling yields the smallest prediction error. On B1-std (Table 18), the summation method's error percentage is more than double the error of the 2-kernel prediction method. On B2-cy (Table 19), the summation method results in a 50% increase in the error estimate when compared to the 5-kernel prediction method.

Table 18. Jacquard Performance Prediction for B1-std

Processors	16	32	48	64	96	128
Exe. Time	1300.23	690.77	481.87	358.10	253.18	194.13
Summ. Error	1.77%	3.41%	3.88%	2.99%	4.69%	6.11%
2-k Error	2.09%	0.06%	1.43%	1.43%	0.43%	1.68%
3-k Error	0.80%	0.31%	0.36%	0.36%	2.11%	1.25%
5-k Error	2.85%	0.47%	0.63%	0.63%	0.38%	1.82%

Table 19. Jacquard Performance Prediction for B2-cy

Processors	16	32	48	64	96	128
Exe. Time	3159.37	1632.92	1129.61	826.06	601.13	446.80
Summ. Error	2.16%	4.59%	3.28%	2.67%	0.81%	4.98%
2-k Error	0.57%	1.22%	1.22%	3.11%	3.04%	1.66%
3-k Error	4.06%	0.79%	0.50%	0.40%	1.11%	0.70%
5-k Error	3.07%	1.02%	1.83%	2.59%	2.60%	1.46%

4.2 Extending the Coupling Concept

In this section, we provide for an understanding of the importance of obtaining optimal kernel coupling values and there use in developing performance models for an application. Previous work by Geisler and Taylor provided for a detailed explanation of extending the coupling concept, in this section we present a small overview of that work [2]. Kernel coupling provides a metric that quantifies the interaction between adjacent kernels. In terms of chains of kernels, this metric is computed using equation 2 presented in Section 1. The performance of the kernels computed using the coupling parameter $C_{W.}$ is measured in terms of the execution time of the kernels.

It is the overall goal of kernel coupling to obtain the measurements that will result in the smallest value for $C_{W.}$ When there is no interaction between the adjacent kernels that are being measured then the kernel coupling value amounts to the summation of the individual kernels. For GYRO, we provide for performance prediction results that utilize three different chain lengths to determine which will result in the more accurate performance predictions. In this section, we provide an illustration of how kernel coupling values are used in predicting the performance of an application.

In order to model an application it is important to have a strong understanding of the relationship between adjacent kernels that compose the application. If a parallel application is composed of four kernels (A, B, C, D) that are executed together in one loop then there exists an analytical model for each respective kernel. The analytical models for these four kernels are E_A, E_B, E_C, and E_D. Therefore, the execution time of the application is given by equation 5.

$$T = \alpha E_A + \beta E_B + \delta E_C + \gamma E_D \quad (5)$$

The coefficients (α, β, δ, γ) represent a weighted average of the coupling values associated with a respective kernel. In determining the pairwise coupling values for kernel A, B, C, and D, these coefficients have the following values:

$$\alpha = \frac{(C_{AB} * P_{AB}) + (C_{DA} * P_{DA})}{(P_{AB} + P_{DA})}$$

$$\beta = \frac{(C_{AB} * P_{AB}) + (C_{BC} * P_{BC})}{(P_{AB} + P_{BC})}$$

$$\delta = \frac{(C_{BC} * P_{BC}) + (C_{CD} * P_{CD})}{(P_{BC} + P_{CD})}$$

$$\gamma = \frac{(C_{CD} * P_{CD}) + (C_{DA} * P_{DA})}{(P_{CD} + P_{DA})}$$

Therefore, the coefficient, γ, for kernel D, includes the coupling values that include kernel D, C_{CD} and C_{DA}. The coefficients for the other kernels have similar equations. Accurate use of the weighted average of the coupling values is based upon execution of a kernel for a fixed number of times using the same input. Additional work is needed to understand the characteristics of an application that determine which group of equations will yield the best performance prediction [13].

4.3 Kernel Coupling Analysis

In this section, we conduct an analysis of the performance of GYRO using kernel coupling. This analysis focuses on identifying the causes of the slight increases and decreases in the kernel coupling values as the number of processors increase. Across the four platforms there are significant changes in the kernel coupling values as the number of processors increase. We use the five-kernel coupling chain for B1-std as a guideline for comparison in this section. The five-kernel coupling chain closely provides an overall representation of the sharing of information amongst the six kernels within the application. For this analysis, we utilize a variety of communication and hardware metrics that will provide for a stronger understanding of the sharing of information amongst the platforms. In this analysis the focus is on utilizing the smallest number of ppn because this has demonstrated that it will yield the best performance.

One of the main characteristics of GYRO that affects the application's performance is its large use of several MPI global communication mechanisms as discussed in the previous section. Global communication in GYRO is largely incurred through the use of the MPI_Alltoall and

MPI_Allreduce operations within kernel 3 and kernel 5. Both of these kernels make use of the get_RHS subroutine. To measure appropriate communication rates we utilize the Integrated Performance Monitor (IPM) [6] on Seaborg and the IBM Trace Library [7] on DataStar P655 and P690.

To understand the data locality and utilization of the L1 data cache, hardware-level measurements are obtained using the Hardware Performance Monitor toolkit (HPM) [4]. Using HPM, we obtain the following metrics: utilization rate, average number of loads per TLB miss, L1 cache hit rate, and Percentage Accesses from L2 per cycle. The utilization rate provides for an overall ratio of the CPU time and wall clock time utilized by a parallel job. A utilization rate that is close to 100% indicates a highly optimized code. The use of the average number of loads per TLB miss provides for an understanding of the data locality for an application. Generally, a value in the range of 500 indicates that most data of a page in the translation look-aside buffer is loaded once. More importantly, the average number of loads per TLB miss gives a better understanding of the possibility of fast access to a new page of data is possible after a TLB miss. This metric is a ratio of the number of load instructions copying data to a register to the number of misses that occur when pages do not have an entry in the TLB. Improvements to the data structures utilized in the program will help to increase this value and improve the performance of an application. Values that are much higher than 500 indicate high data locality. The L1 cache hit rate gives an understanding of the ratio of load/store instructions that can be completed without accessing main memory. Throughout this section, Figure 16 will be referenced to provide an overall understanding of the kernel coupling trends across all platforms. In subsequent subsections, we analyze what causes the changes in kernel coupling values.

4.3.1 Seaborg Analysis

On NERSC Seaborg the coupling values become more constructive as the number of processors increases from 48 to 96. Specifically, at 96 processors the coupling value is less than one. On Seaborg, IPM provides several measurements that lead to an understanding of communication patterns for this range of processors. Table 20 provides for a brief summary of various communication measurements obtained on Seaborg for B1-std. The noticeable drop in kernel coupling values from 48 to 96 processors is believed to be a result of many changes in the sharing of information for GYRO. IPM provides details of a large decrease in the overall communication percentage within the GYRO application from 48 to 64 processors. According

to Table 20, it can be seen that there is a significant drop in the communication percentage from 48 to 64 processors.

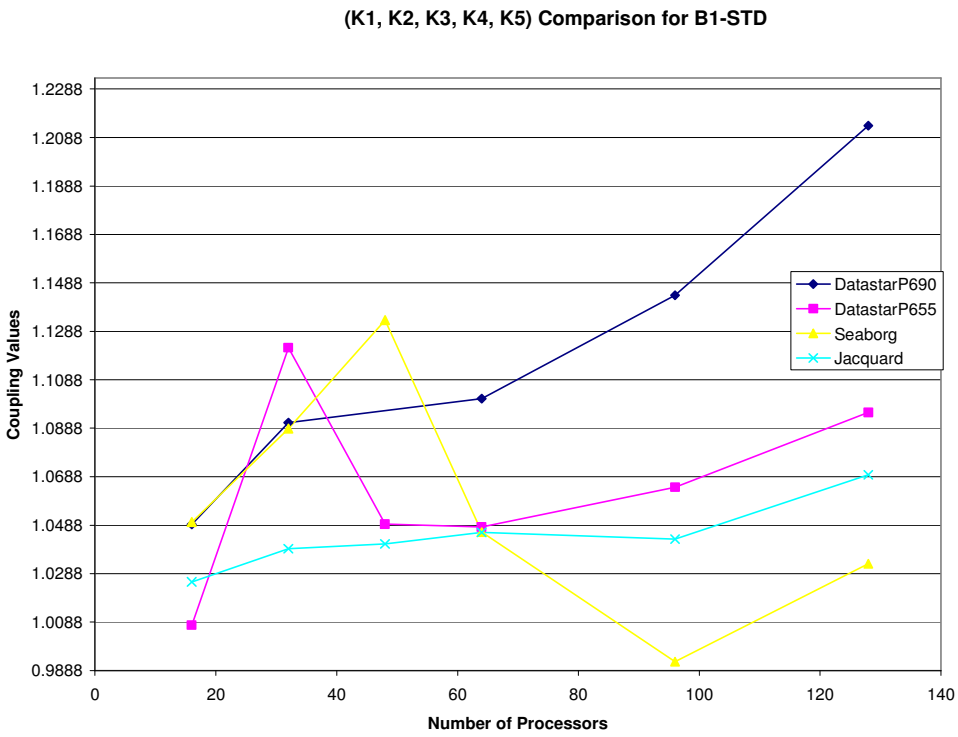

Figure 16. Cross-Platform Comparison

At 48 processors, the overall communication percentage utilizing 1 processor per node is 19.72%. Most of this communication occurs within the get_RHS subroutine that utilizes the all-to-all and all-reduce communication routines. These two routines account for around 99.20%, of the total MPI communication within the application when executed on 48 processors. The all-to-all routine accounts for the majority of the communication time across all processors because it is utilized for the communication of several discretized values in GYRO through the column transpose operations. The all-to-all communication routine accounts for more than 8% of the total execution time for GYRO on B1-std.

Table 20. Communication on Seaborg

Processor size	16	32	48	64	96	128
Total Comm. %	17.80%	21.43	19.72%	12.73%	12.47%	12.84%
All-to-all routine	85.83%	79.15%	74.22%	74.00%	69.15%	74.44%
All-reduce routine	13.83%	20.17%	25.02%	25.18%	29.73%	23.01%
Avg Message size (gbytes)	0.28846	0.159852	0.118293	0.095518	0.075597	0.064762
Total Transfer (gbytes)	4.61534	5.11131	5.67811	6.11314	7.25727	8.28954

In addition, the average message size drops significantly for 64 processors this results in less congestion in the network. The drop in the average message size continues for 96 processors. Additionally, it can be seen that as the processors increase the total amount of data transferred increases. This does not affect communication because smaller messages allow for more messages in the network. At 48 processors there is a slight increase in the communication percentage to 12.84%. The message size continues to decrease and the total data transferred follows the same trend and continues to increase. These changes cause for slight increases in the kernel coupling values after 96 processors but the coupling values do not become as destructive as they were for less than 48 processors.

HPM gives a better understanding of the drops in kernel coupling values as the number of processors increases on NERSC Seaborg. Table 21 provides detailed information about the hardware measurements encountered on NERSC Seaborg utilizing hpm. It can be seen that as the processor sizes increase that the utilization rate for this code improves slightly. This improvement in utilization means that the application is becoming more optimized as the number of processor increases. The utilization rate is a metric that can be improved easily by optimizing the data structures. Within GYRO, the sizes of the data structures decrease as the processor sizes increase. Improving the sizes of the data structures or further optimizing them for the lower processor counts could lead to more constructive kernel coupling values.

The average number of loads per TLB Miss shows slight decreases as the processor sizes rise. This can be attributed to the fact that the sizes of the data structures decrease as processors are increased. At 48 processors, the average number of Loads per TLB Miss is much higher because there is more data that is being processed in the application than successive processor

counts. This metric decreases for successive processor counts but there is a slight increase in the utilization of the L1 cache and the L2 cache as the number of processors increase.

Table 21. Data Locality on Seaborg

Processor size	16	32	48	64	96	128
Utilization Rate	99.32%	99.36%	98.07%	98.27%	98.49%	94.86%
Number of Loads per TLB Miss	1093.82	1003.70	1368.09	1011.78.	1010.38	1066.32
L1 Cache Hit Rate	94.40%	94.42%	94.37%	94.72%	95.08%	95.32%
% Accesses from L2 per cycle	2.982%	2.96%	2.43%	2.75%	2.73%	2.87%

The L1 cache hit rate shows that the ratio of load/store instructions completed increases as the processor sizes increase. This increase could have a direct effect on the improvements in the kernel coupling values as the number of processors are increased from 48 to 96. The combination of decreases in the overall communication to execution percentage, increases in the number of loads per TLB miss, and increases in the L1 and L2 caches results in more constructive kernel coupling values for 48 to 128 processors on Seaborg.

4.3.2 DataStar P655 Analysis

On DataStar P655, the communication trend continues to have a direct influence on the kernel coupling values of the application. The DataStar system utilizes the IBM Federation switch to connect nodes. This provides for a better performance than the Seaborg platform which utilizes a Colony interconnect. The change in 5-kernel coupling values for the $K_{1, 2, 3, 4, 5}$ chain shows a decrease in coupling values as the number of processors increase from 32 to 48. There is only a slight increase in the communication percentage for this change, as Table 22 shows. There is a large decrease in the overall message sizes for the all-to-all routine. This allows for more messages to move through the system. The all-to-all routine accounts for almost 70% of the total communication for 32, 48, and 64 processors. Therefore, smaller message sizes allow for more messages to move through the network. From 32 to 48 processors there is over a 40% increase in the message size for the all-reduce routine. This does not have as large an effect on the kernel coupling of the application because there is an almost identical decrease in the all-to-all routine. Furthermore, the all-to-all routine accounts for more communication in the application.

When the number of processors increases from 48 to 64 the kernel coupling values for the application show very little deviation. The overall communication of the application only increases by only 2.1%. The combined communication between the all-to-all and all-reduce routines show a slight decrease of 0.15%. This could indicate that additional communication is being done outside of the main kernels of the application that would not affect the kernel coupling values. As the number of processors increase beyond 64 processors there is an increase in the kernel coupling values. Specifically from 64 to 96 processors we see that the communication to total execution ratio for the application grows by over 6%. There is a slight increase in the combined communication for the all-to-all and all-reduce routines of less than 0.025%. The message sizes for the all-reduce subroutine continue to rise. It is believed that this has an effect on the kernel coupling values for GYRO as the number of processors increase beyond 64.

Table 22. Communication on DataStar P655

Processor size	16	32	48	64	96	128
Total Comm. %	9.20%	19.50%	20.41%	20.84%	22.10%	12.75%
All-to-all avg msg size (bytes)	137035.3	46908.2	27329.4	18051.8	11407.3	7675.3
All-reduce avg msg size (bytes)	291.6	559.4	790.2	1052.7	1458.1	1898.2
All-to-all % comm	55.11%	67.03	69.44%	68.54%	63.54%	51.33%
All-reduce % comm	44.67%	31.64	29.27%	30.02%	34.77%	48.14%

Table 23 provides detailed information about the hardware measurements encountered on the DataStar P655 utilizing hpm. For the analysis of the DataStar P655 we focus largely on what causes the peak in the kernel coupling values for 32 processors on the K1, 2, 3, 4, 5 chain. At 32 processors it can be seen that GYRO has a better utilization rate than it experiences when executed on 48 processors. It can be seen that the average number of loads per TLB miss is lower for 32 processors. Although the data structure sizes decrease for 48 processors, there is a higher average number of loads than at 32 processors. The kernel coupling trend on the P655 platform shows that as the number of processors is increased so does the average number of loads per TLB miss. In terms of the L1 cache hit rate, there are slight improvements as the number of processors are increased. The increase in the constructiveness of the kernel coupling values from 32 to 48 processors shows a strong correlation to the improvements in the L1 cache

hit rate for these two processors. This metric improves slight as the number of processors increases from 48 to 64 processors. At 48 to 64 processors there are very small changes in the kernel coupling values, they remain almost constant. After 64 processors, the kernel coupling values become more destructive which follows the trend of the L1 cache hit rate decreasing again. The kernel coupling value at 96 processors increases to greater than the value at 48 processors but not greater than the value at 32 processors. Similarly, the L1 Cache Hit Rate follows the same trends as it drops below the value for 48 processors but not as low as the value for 32 processors.

Table 23. Data Locality on DataStar P655

Processor size	16	32	48	64	96	128
Utilization Rate	99.6%	90.79%	89.16%	91.84%	88.817%	97.67
Number of Loads per TLB Miss	1973.26	2654.41	2981.58	3230.18	3397.42	3438.32
L1 Cache Hit Rate	85.35	86.14%	86.50%	86.68%	86.48%	86.726
% Accesses from L2 per cycle	5.74%	5.78%	5.53%	5.70%	5.81%	5.71%

4.3.3 DataStar P690 Analysis

On the DataStar P690 platform we explore the factors that cause the kernel coupling values to increase as the number of processors is increase from 16 to 64. Table 24 provides details of the communication of GYRO as the number of processors are increased. It can be seen that there are consistent rises in the overall ratio of communication to total execution time. When GYRO is executed utilizing 16 processors it yields the most constructive kernel coupling values, and it also has the lowest communication to total execution ratio for this number of processors. It should be noted that the rate of increase from 16 to 32 processors is much greater than the increase from 32 to 64 processors.

Table 25 provides an overview of key measurements taken using hpmcount for GYRO on B1-std when executed on the P690. It is interesting to note that as the number of processors increase the L1 Cache hit rate increases as well on this benchmark. The utilization rate for B1-std also increases showing that the code is becoming more optimized for this architecture as the processor sizes increase. These measurements would lead to the conclusion that there should be

an increase in the sharing of information amongst adjacent kernels, which should result in lower kernel coupling values.

Table 24. Communication on DataStar P690

Processor size	16	32	64	96	128
Total Comm. %	13.49%	14.42%	20.26%	20.89%	21.02%
All-to-all avg msg size (bytes)	137035.3	46908.2	18051.8	11407.3	7675.3
All-reduce avg msg size (bytes)	291.6	559.4	1052.7	1458.1	1898.2
All-to-all % comm	74.21%	65.70%	70.58%	69.38%	65.71%
All-reduce % comm	25.73%	33.87%	29.25%	30.43%	34.02%

However, on the P690 the kernel coupling values continue to increase. It can be observed from Table 25 that there are slight decreases in the utilization of the L2 cache as the number of processors increase. Specifically, there is a large decrease from 16 to 32 processors (half node to full node usage) on this platform. It could be concluded that utilizing less of the L2 cache results in less sharing of information despite increases in the utilization rate and L1 hit rate.

Table 25. Data Locality on DataStar P690

Processor size	16	32	64	96	128
Utilization Rate	95.07%	96.91%	97.74%	97.54%	97.24%
Number of Loads per TLB Miss	2111.21	2729.33	3294.08	3672.99	3897.07
L1 Cache Hit Rate	86.90%	87.74%	87.98%	88.26%	88.69%
% Accesses from L2 per cycle	4.35%	3.58%	3.38%	3.41%	3.36%

4.4 Coupling Analysis and Modeling Summary

The kernel coupling trends show that Seaborg and the P655 platform follow the exact same trends, but reaching their peaks and descents for different numbers of processors. The P655 Platform reaches its peak at 32 processors, while Seaborg reaches its peak at 48 processors. This roughly equates to both platforms reaching peaks when the total memory available for the application is around 57-64GB. In terms of cache settings, the Seaborg and the P655 contain 64/32 (Data/Instr) L1 cache. However, the P655 is equipped with 1.5MB L2 cache and Seaborg

has an 8MB L2 cache. Consistent trends amongst both of these platforms show that variations in the kernel coupling values are largely attributed to communication and data locality.

The P690 platform does not experience a large rise and fall in the kernel coupling values for the 5-kernel chain. It mainly rises to the right. The main difference with the execution of GYRO on this platform involves full to half node usage. On the Seaborg and P655 platforms only 1 ppn is utilized. Throughout this analysis, we show the percentage of communication utilized to demonstrate the large effect that the performance of the platform's interconnection network can have on the application. Furthermore, we are able to see that there is a correlation in the changes in the communication percentage to changes in kernel coupling values. Currently, Jacquard is not equipped with the necessary performance measurement tools so we are unable to present communication and data locality metrics to provide insight into its performance.

5. RELATED WORK

Currently, there is an intensive effort to analyze and improve the performance characteristics of GYRO. Worley et al [12] gathered extensive runtime information of GYRO across a variety of parallel platforms. The portability of the GYRO application is one of the major advantages of the single source application of the code. Therefore, Worley was able to evaluate the application on various supercomputing platforms. Their work focused on understanding the combined computational rates of selected embedded timers and trace events. Their work was beneficial in showing which nonlinear evaluation methods were efficient and the platforms that were most efficient for that method.

Worley analyzed GYRO on a variety of IBM cluster platforms including the NERSC Seaborg and an IBM P690 cluster. Through their work, they were able to identify for which scenarios performance models and tools would be beneficial, such as variance across different parallel platforms and processor sizes and then identifying how computational rates change for each. Our work complements Worley's work by showing how performance models can form accurate predictions for different platforms and across various processor sizes. Further, our work can help to explain why the performance differences occur.

Huck and Malony further analyzed the performance data obtained using GYRO through the PerfExplorer framework [5]. Their work demonstrated the effectiveness of using data mining techniques to reduce or aggregate extensive performance profiles. The benefit of utilizing the PerfExplorer was exhibited by drilling through excessive performance data to identify the performance bottlenecks across a selected platform. Our work complements the findings of Huck and Malony by identifying the bottlenecks of GYRO at a functional level thereby allowing for further performance improvements.

Yang et al contributed work characterizing the performance characteristics of GYRO in relation to benchmarks B1-std, B2-cy, and B3-gtc [15]. Yang's work focuses on a partial execution scheme for performance prediction. This work utilizes the static nature of the timesteps involved with GYRO's benchmarks. The partial execution method was able to yield percentage errors of 5.6% on the Seaborg platform but larger errors on other platforms. Our work complements Yang's work nicely by applying kernel coupling to predict GYRO's execution time and understanding how the most computationally intensive kernels of the

application share resources. The percentage errors resulting from our predictions are in the range of 5% across the platforms given in Table 1.

6. CONCLUSION

Performance models offer an important viability to researchers. They are able to accurately model applications so that scientists can further analyze performance characteristics of applications and determine if the resources are available to obtain experimental data. Increasing the understanding of the relationship between application and the system on which it is executed is increasingly important.

In this thesis, we have shown that processor partitioning must be used to understand the best processor configuration for the execution of an application. Through our analysis we have investigated the effects of processor partitioning on changes in GYRO's total execution and communication time. GYRO has a large amount of global communication through the use of the all-to-all and all-reduce routines. On all four platforms, when the smallest number of ppn are utilized there is a significant decrease in communication. This translates into better performance, for example, processor partitioning on the NERSC Seaborg identifies a 46.5% improvement in communication time when using the worst-cast (2 x 16) to the best-case (32 x 1) processor partitioning scheme. The improvement in the overall execution of the application is 24.4%. Also, on the DataStar P655 there is a 79.5% improvement in communication and 38.4% improvement in total execution time when comparing the worse-case (4 x 8) and best-case (32 x 1) processor partitioning schemes. The improvement in communication is much greater than the improvement in overall execution, which is a strong indication that the communication can have a severely negative impact on the application's performance. Overall, optimal performance is obtained for this application when executed on 1 ppn or smallest number of processors per node on each platform.

This work illustrates the viability and accuracy of using kernel coupling as a performance modeling technique across several supercomputing parallel platforms. The analysis allows for a better understanding of the optimal performance platforms for GYRO. The results indicate that kernel coupling can be used to predict application performance to less than 5 % across all platforms. For example, on NERSC Jacquard the 3-kernel coupling prediction method provides for less than 0.98% error versus 3.81% error from the summation method for B1-std. On the DataStar P655, the 5-kernel coupling prediction method provides for less than 2% error versus 5.81% error from the summation method for B1-std.

In addition, this work will serve as an extension of previous work on kernel coupling in relation to the execution of programs on single processors and the reuse of kernel coupling values for parallel performance prediction. Through our analysis we have been able to identify the strong relationship between communication and data locality within the application to trends in kernel coupling values. For example, on the NERSC Seaborg and the DataStar P655 and P690, it can be seen that improvements in the L1 cache hit rate, number of loads per TLB miss, and % accesses from L2 per cycle will result in more constructive kernel coupling values.

Future work on this application will focus on gaining an understanding of how the GYRO application can be improved so that it results in more constructive coupling. Improvements in the algorithms used by the subroutines in GYRO can result in performance gains when evaluated in isolation, but kernel coupling will detail if these improvements translate into an overall improvement in the applications performance. Further analysis will discuss the effects of such improvements on communication and data locality across various platforms. Another issue that will be focused upon in future work is gaining an understanding of the performance of GYRO on GRID environments and the results of kernel coupling from GRID platforms.

REFERENCES

[1] M. Fahey, and J. Candy. *GYRO: A 5-d gyrokinetic-maxwell solver.* Proceedings of Supercomputing, November 2004. *(http://www.csm.ornl.gov/SC2004/pap213.pdf)*

[2] J. Geisler, and V. Taylor. *Performance Coupling: Case Studies for Improving the Performance of Scientific Applications.* Journal on Parallel and Distributed Computing, Vol. 62, no 8, August 2002, pp. 1227-1247.

[3] GYRO. (n.d.). Retrieved June 7, 2006, from *http://fusion.gat.com/comp/parallel/.*

[4] Hardware Performance Monitor Toolkit, (n.d). Retrieved June 7, 2006, from *http://www.nersc.gov/nusers/resources/software/ibm/hpmcount/HPM_2_4_2.html.*

[5] K. A. Huck, and A. D. Malony. *PerfExplorer: A Performance Data Mining Framework for Large-Scale Parallel Computing*, Proceedings of the SC 2005 Conference, IEEE/ACM, Seattle, Nov. 2005. *(http://sc05.supercomputing.org/schedule/pdf /pap348.pdf)*

[6] Integrated Performance Monitor Toolkit, (n.d.). Retrieved June 7, 2006, from *http://www.nersc.gov/nusers/resources/software/tools/ipm.php.*

[7] Message Passing Interface Trace Library, (n.d.). Retrieved June 6, 2006, from *http://www.sdsc.edu/user_services/datastar/docs/trace.html.*

[8] National Energy Research Supercomputing Center Jacquard, (n.d.). Retrieved June 6, 2006, from *http://www.nersc.gov/nusers/resources/jacquard/.*

[9] National Energy Research Supercomputing Center Seaborg, (n.d.) Retrieved June 6, 2006, from *http://www.nersc.gov/nusers/resources/SP/.*

[10] San Diego Supercomputing Center DataStar, (n.d). Retrieved June 6, 2006 from *http://www.sdsc.edu/user_services/datastar/.*

[11] V. Taylor, X. Wu, and R. Stevens. *Prophesy: An Infrastructure for Performance Analysis and Modeling of Parallel and Grid Applications*, ACM SIGMETRICS Performance Evaluation Review, Volume 30, Issue 4, March 2003. *(http://prophesy.cs.tamu.edu/publications/per03.pdf)*

[12] P. Worley, J. Candy, L. Carrington, and K. Huck. *Performance Analysis of GYRO: A Tool Evaluation*. 2005 Journal of Physics: Conference Series 16. pp. 551-555.

[13] X. Wu, V. Taylor, J. Geisler, and R. Stevens. *Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance*, Proceedings of the 18th International Parallel and Distributed Processing Symposium *(IPDPS2004)*, Santa Fe, New Mexico, April 26-30, 2004. *(http://prophesy.cs.tamu.edu/publications/ipdps04.pdf)*

[14] X. Wu, V. Taylor, C. Lively, and S. Sharkawi. *Processors Partitioning: A Performance-Based Analysis for MPI Applications*, Supercomputing 2006 Conference, IEEE/ACM, Tampa, November 2006, (Submitted)

[15] L. Yang, X. Ma, and F. Mueller. *Cross-Platform Performance Prediction of Parallel Applications Using Partial Execution*, Proceedings of the SC 2005 Conference, IEEE/ACM, Seattle, November 2005. *(http://sc05.supercomputing.org/schedule/pdf/pap245.pdf?sc05=df8b0c98ebb78b8edcbbde1557c568a6)*

VITA

NAME:	Charles Wesley Lively III
ADDRESS:	301 Harvey R. Bright Building, College Station, TX 77843-3112
EMAIL ADDRESS:	clively@cs.tamu.edu
EDUCATION:	B.S.E., Computer Engineering, Mercer University, 2004
	M.S., Computer Engineering, Texas A&M University, 2006