

ABSTRACT

RAGHAVACHAR, KAVITHA. Performance Modeling using a Genetic Programming based Model Error Correction Procedure. (Under the direction of Dr. G Mahinthakumar.)

Application performance models provide insight to designers of high performance computing (HPC) systems on the role of subsystems such as the processor or the network in determining application performance and allow HPC centers to more accurately target procurements to resource requirements. Performance models can also be used to identify application performance bottlenecks and to provide insights about scalability issues. The suitability of a performance model, however, for a particular performance investigation is a function of both the accuracy and the cost of the model.

A semi-empirical model developed in an earlier publication for an astrophysics application was shown to be inaccurate when predicting communication cost for large numbers of processors. It was hypothesized that this deficiency is due to the inability of the model to adequately capture communication contention (threshold effects) as well as other un-modeled components such as noise and I/O contention. This thesis demonstrates a new approach to capture these unknown features to improve the predictive capabilities of the model. This approach uses a systematic model error correction procedure that uses evolutionary algorithms to find an error correction term to augment the existing model. Four variations of this procedure were investigated and all were shown to produce improved results than the old model. Successful cross-platform application of this approach showed that it adequately captures machine dependent characteristics. This approach was then extended to a second application, which too showed improved results than the standard semi-empirical modeling approach.

**PERFORMANCE MODELING USING A GENETIC PROGRAMMING
BASED MODEL ERROR CORRECTION PROCEDURE**

By

KAVITHA RAGHAVACHAR

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

CIVIL ENGINEERING

Raleigh, North Carolina
August 2006

APPROVED BY:

S. Ranji Ranjithan
Associate Professor of Civil Engineering
Water Resources/Environmental Engineering

Co-chair of Advisory Committee

John W. Baugh Jr.
Professor of Civil Engineering
Computer-Aided Engineering

Co-chair of Advisory Committee

G. (Kumar) Mahinthakumar
Assistant Professor of Civil Engineering
Computer-Aided Engineering

Chair of Advisory Committee

To my Parents, Husband, Brother and in-laws

BIOGRAPHY

Kavitha Raghavachar was born in Hassan, Karnataka, India in July 1977. She completed her schooling before joining a Bachelors in engineering program with a major in Civil Engineering at the Malnad College of Engineering, Hassan. She graduated with a first class with distinction in Aug 1999. In August 2003, she joined the NCSU as a PBS (Post Baccalaureate Studies) student. In January 2004, she was admitted to the graduate program in Civil Engineering at the North Carolina State University at Raleigh, North Carolina. She specializes in Computer Aided Engineering.

ACKNOWLEDGEMENT

OM NAMO NARAYANAYA

I would like to express my sincere thanks to everyone who helped me with this project and made my graduate school days a wonderful experience for me.

My foremost thanks to Dr. G. Mahinthakumar for providing me a assistantship to support myself and for his guidance throughout my stay at NCSU. My Special thanks to Dr. S. Ranji Ranjitahn for his constant encouragement and guidance throughout my thesis project. My sincere thanks to Dr. John W Baugh for his encouragement to join NCSU. My special thanks to Dr. Patrick Worley (ORNL) for all his support and guidance to the project. I thank Dr. Emily for her help with this project.

I am extremely indebted to my husband, Arun Krishna, for his support and encouragement to pursue a Master of Science degree. I thank him for being such a wonderful person in my life.

I am grateful to my grandparents, parents, brother, in-laws and the rest of my great family, whose blessings and wishes are the core reason for everything that I am today.

This work was supported by the Department of Energy's SciDAC program (Scientific Discovery through Advanced Computing). I gratefully acknowledge the supercomputer resources provided by the National Center for Supercomputing Applications and the Oak Ridge National Laboratory that was necessary for this work.

TABLE OF CONTENTS

	Page
LIST OF FIGURES.....	vi
LIST OF TABLES.....	vii
1. INTRODUCTION.....	1
2. MODEL ERROR CORRECTION PROCEDURE (MECP).....	3
1.1 General approach and case studies.....	6
2.2 MECP applied to evh1 communication model.....	10
3.3 EVH1 Model results.....	11
3. EXTENSION OF MECP TO SECOND APPLICATION GYRO.....	16
1.1 Computation timing model.....	16
2.2 Communication timing model.....	17
3.3 GYRO Model results.....	18
4. CROSSPLATFORM PREDICTION AND DISCUSSIONS.....	22
5. CONCLUSIONS.....	26
6. REFERENCES.....	27

LIST OF FIGURES

		Page
Figure 1.	Example of GP procedure.....	5
Figure 2.	A sample solution representation of the GP solution.....	5
Figure 3.	Semi empirical modeling.....	7
Figure 4.	Flow chart of general approach.....	7
Figure 5.	Model Error Correction procedure.....	9
Figure 6.	Prediction errors for original model and four cases of MECP.....	14
Figure 7.	Frequency of improved prediction than the original model in the random trials	14
Figure 8.	Comparison of the observed values and predicted values by the original model and the best models for Cases-1, 2, 3 and 4	15
Figure 9.	Frequency and prediction errors comparison with Gyro Semi empirical model and 4 cases of MECP model.....	19
Figure 10.	Gyro computation model for B1-std, B2-cy and B3-gtc problems.....	20
Figure 11.	Gyro communication time model for B1-std, B2-cy and B3-gtc problems (for fitting data).....	21
Figure 12.	Gyro communication time model for B1-std, B2-cy and B3-gtc problems (for prediction data).....	21
Figure 13.	Cross Platform prediction for Teragrid using Cheetah: MECP-based Models for EVH1 application.....	23
Figure 14.	Cross Platform prediction for Cheetah using a Teragrid: MECP model for GYRO application.....	24

LIST OF TABLES

	Page
Table 1. Summary of four different cases	8
Table 2. GA/GP parameter settings for MECP.....	13
Table 3. Summary of fitted values for model parameters.....	14
Table 4. Message size and Processor mode for the three standard problems of GYRO.....	20
Table 5. Description of computer architecture.....	25
Table 6. Example of function form of <i>ect</i>.....	25

1. INTRODUCTION

Achieving high performance on parallel applications can be challenging. Performance of a parallel application depends primarily on two characteristics, one pertaining to the application and the other pertaining to the machine. Application characteristics include the primary algorithmic kernels, programming language, parallelization strategy, problem size and other input parameters. Machine characteristics include hardware metrics, such as processor performance, memory performance and network performance, and software metrics, including compiler performance and efficiency of communication and math libraries. Producing applications that work well across a wide range of high performance computing (HPC) platforms is a non-trivial task because the machine characteristics and their interdependence on application characteristics vary from platform to platform. The aforementioned reasons also make performance prediction a difficult task. In recent years, performance prediction for parallel architectures has attracted considerable attention, ranging from kernel benchmarking studies, application performance studies, performance modeling, and detailed comparative analysis of newer architectures [e.g., [2], [10], [12], [9]].

Performance prediction across platforms is increasingly important for developers and scientists to determine the performance of their specific applications on a plethora of platforms in deciding the system that best fits their needs in the long run. If scientists were provided with performance estimations for their applications on several large clusters, the process of choosing and gaining accesses to suitable platforms would be considerably simplified.

Standard modeling methodologies may include semi-empirical modeling and use of modeling assertion tools [1]. Both methodologies have advantages and caveats. In the former approach, application parameters are captured easily but do not adequately model machine dependencies such as noise, IO contention, and/or message contention. In the latter approach, machine parameters are captured well but do not adequately capture application/problem specific dependencies. This thesis develops and demonstrates a general approach for modeling performance of parallel applications. This approach starts by developing a semi-empirical model and then improving this further by using a genetic programming-based model error correction procedure (MECP).

A semi-empirical parallel performance model typically consists of two components: computation time model and communication time model. Earlier work involving standard semi-empirical modeling [7] for an astrophysics code EVH1 (Enhanced Virginia Hydrodynamics code) showed that while the computation model component prediction was satisfactory, the communication model prediction, particularly for large processor counts, was poor. This deficiency is probably due to the inability of a typical semi-empirical model to capture difficult to model components such as message contention and noise. The MECP procedure demonstrated in this thesis seeks to address this deficiency by augmenting an existing model with an error correction term.

The overall structure of this thesis is as follows. First we describe the overall MECP approach, followed by a section that shows the application of this approach to our first application code EVH1. Then we extend this approach to a second application GYRO [6], [8]. Then we validate both performance models across two parallel platforms.

2. MODEL ERROR CORRECTION PROCEDURE

Parallel performance models typically have several components (e.g., communication contention, non-uniform memory access, I/O, etc) that are not adequately represented by terms in the performance model or not adequately captured by the model parameters. One way to address these deficiencies is to create another function (hereafter referred to as the model error correction term) that augments the original model by capturing these un-modeled features. In this thesis, the Model Error Correction Procedure (MECP) [13] is used to identify a model error correction term while tuning the original model parameters.

Missing or misrepresented processes in the construction of a model will result in a systematic error in modeled predictions (i.e., modeled communication time). The model error correction procedure is designed to improve capabilities of mechanistic models through simultaneous tuning of parameter values and construction of an error correction term (*ect*), which is a function that represents the systematic error and appropriately adjusts the model outputs to match the observed data. The systematic error is dependent on the input conditions and the errors in the simulated outputs, and accordingly, the error correction term should be constructed as a function of both the model inputs and outputs to more accurately predict for new input conditions. A simultaneous search for model parameter values and the *ect* forms the basis for MECP to enable the appropriate correction of each source of error. This requires a numeric search method for the parameter estimation and a symbolic search method for *ect* determination. The following section describes an implementation of MECP using evolutionary computation (EC) procedures, namely genetic algorithms (GAs) and genetic programming (GP), to enable the numeric and symbolic searches, respectively.

Evolutionary algorithms (EA) are search methods that operate on a population of solutions and mimic the mechanisms of natural selection to find optimal solutions for a defined problem. In the EA-based MECP used in this thesis, GP operators are used for generating the error correction term (*ect*) and GA operators are used for determining the model parameter values. GA performs a numeric search by representing a solution as a string of real, binary or integer values. GP conducts a symbolic search over a set of mathematical operators, such as $+$, $-$, \times and $/$, to construct a function that minimizes the error between a set of data and a function output.

Figure 1 shows an example of how GP works on a set of data to form a function. GP works with two sets of variables. One set is called the terminal set, which contains independent variables, such as x . The other set, known as the functional set, which contains operators. Symbolic search is performed using these two variable sets. GP explores a large number of possible functional relationships to identify the function that best fits the data.

A tree data structure (see example in Figure 2) is used to represent each potential solution for *ect*, which is composed of a subset of the terminal set variables and functional set elements. These potential solutions in a population undergo crossover, mutation, and selection as in any EA-based search.

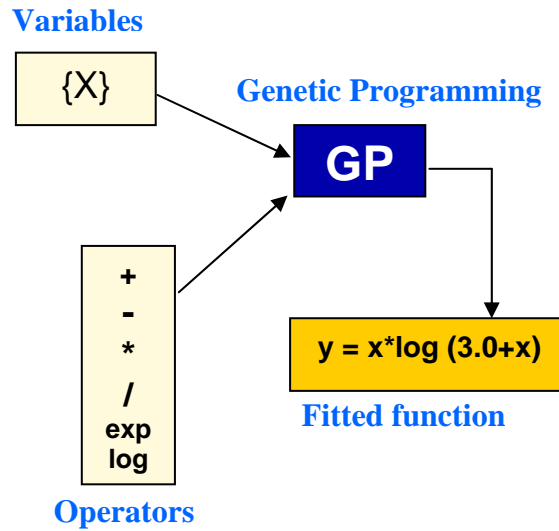


Figure 1 Example of GP procedure

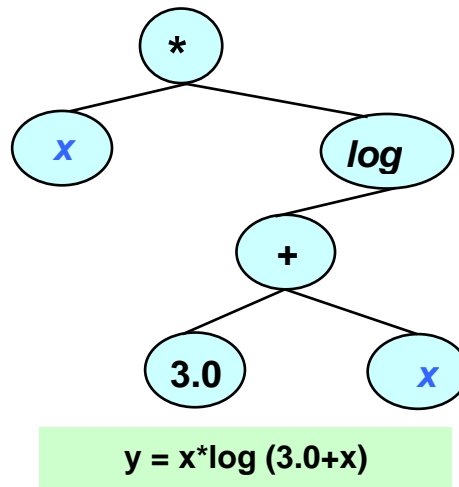


Figure 2 A sample solution representation of the GP solution

In the GP procedure, each potential solution (i.e., error correction term), composed of a subset of the terminal set variables and functional set elements, is represented by a tree data structure as shown in Figure 2. These potential solutions in a population undergo crossover, mutation, and selection as in any EA-based search.

2.1 General Approach and Case Studies

A semi empirical model has been developed to predict computation time and communication times for any given HPC application. A set of data for developing the semi-empirical model was collected by recording the total run time and communication times for several executions of the application code using different application parameters (problem size, simulation time) and machine parameters (different machines, number of processors). The model parameter values were identified using the Levenberg-Marquadt search procedure to best fit the data by minimizing the error between the observed data and the model. Lower and upper bounds were specified to constrain the parameter values within acceptable ranges. The MATLAB function *lsqcurvefit* was used to perform the nonlinear regression. For problems with bound constraints, *lsqcurvefit* uses the conjugate gradient trust-region method for non-linear regression. Our goal here is to first fit these times using an appropriate model and a subset of the measured data, and then test the effectiveness of this model by predicting the times for the remainder of the data. The procedure for modeling using semi empirical approach is shown in Figure 3.

Figure 4 shows the flowchart of our general approach. We start by developing a semi empirical model for any HPC application as described earlier. If the communication and computation time model predictions are satisfactory, we use these models to validate the rest of the data. If the semi empirical model predictions are not satisfactory, we correct the model using MECP to improve the predictive capability of the model and then use the corrected model for prediction.

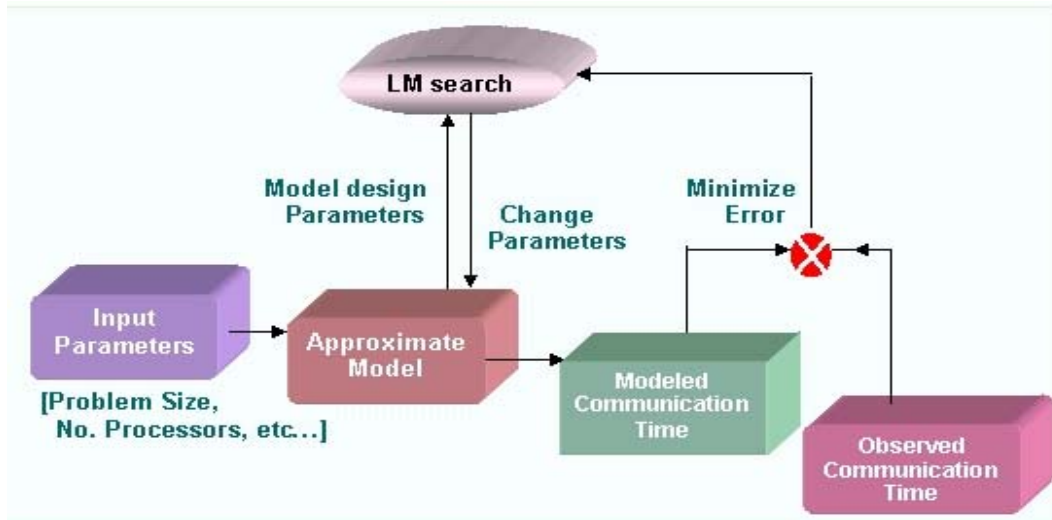


Figure 3 Semi empirical modeling

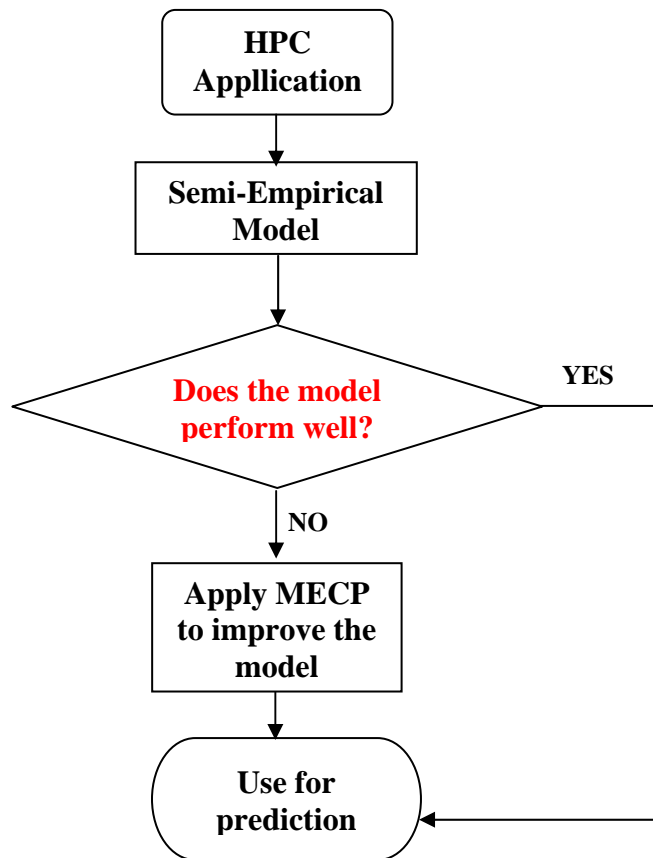


Figure 4 Flow chart of general approach

Typically, the level of investigation that can be done when using MECP for a particular model depends upon the user's knowledge about the application and architecture. Here we have investigated four variations of MECP-based communication models. These cases are summarized in Table 1 and are described below. Generally, we can start with any of these four variations for a particular model. Initially, we started our investigation with only a symbolic search excluding the numeric search. The parameter values are fixed to those as determined by the Levenberg-Marquadt procedure for cases-1 and 2. For case-1, ect is identified and its output replaces the modeled output. In case-2, ect is identified as a term that is added to the modeled output. We further investigated MECP-based models by including the numeric search. Cases 3 and 4 extend cases 1 and 2, respectively, by including the search for parameter values. The parameters values may vary within a small range bounded around the values determined using the Levenberg-Marquadt procedure.

Table 1. Summary of four different cases: ect = Error correction term, t_{comm} = original communication model, $ect = f(nx, np, T)$

Cases	Model	Parameter Values
Case - 1	$t_{calculated} = f(t_{comm}, ect)$	Set
Case - 2	$t_{calculated} = t_{comm} + ect$	Set
Case - 3	$t_{calculated} = f(t_{comm}, ect)$	Search
Case - 4	$t_{calculated} = t_{comm} + ect$	Search

The error correction term is a function of input parameters n_x , n_p , and T . Since the parameters are not fixed in equations (1) and (2), the unknown machine parameters are found by GA for these cases. The *ect* is not shown here, but includes operators (+, -, × and /) and input parameters (application and machine parameters). The objective function to be minimized in the GA/GP procedure is a measure of the error between the calculated and measured communication times, which is given as:

$$Error = \frac{\left[\sqrt{\sum (t_{observed} - t_{calculated})^2} \right]}{n}$$

Where, n = Number of data points, $t_{observed}$ = Observed communication time, $t_{calculated}$ = Calculated communication time.

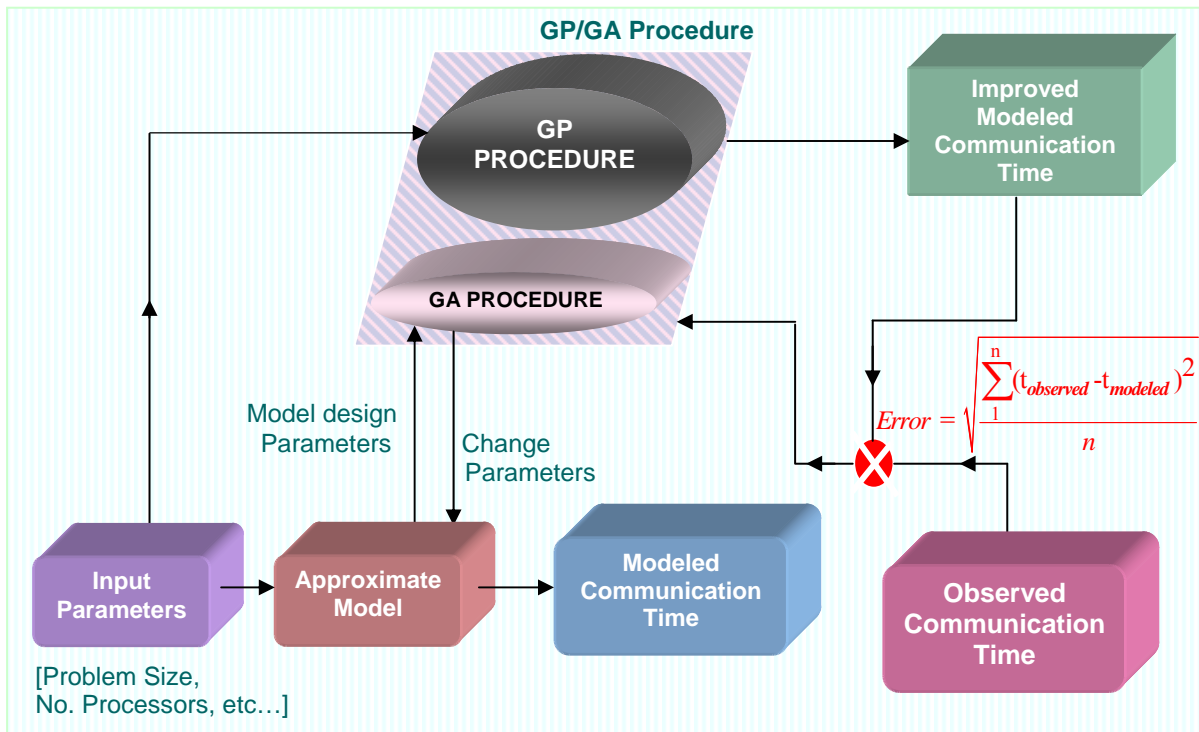


Figure 5 Model Error Correction Procedure

2.3 MECP applied to EVH1 communication model

EVH1 (Enhanced version of Virginia Hydrodynamics) is a finite-difference astrophysics code containing over 6500 lines of Fortran and MPI (Message Passing Interface) [4]. In earlier work [7], EVH1 runtime was modeled for 2-Dimensional (2D) and 3-Dimensional (3D) problems on two different architectures. A semi-empirical model was used in this study involving a set of machine and application parameters. These model parameters were then fitted using a Levenberg-Marquadt procedure using a subset of the measured data. In this paper we restrict our attention to modeling communication time for 2D problems on a single architecture. While EVH1 deals with a specific application area, the algorithmic kernels used are representative of many numerically intensive finite-difference codes. Therefore modeling methodologies used in this paper have more general applicability.

The original EVH1 2D semi empirical communication model [7] which is dominated by MPIALLTOALL calls involved in a matrix transpose operation is given as:

$$t_{comm} = 2f \cdot T \cdot nx \cdot \left\{ \begin{array}{l} \left(\frac{nx^2}{np} - nx \right) g \\ + (np - 1)^h \left(i + j \frac{nx^2}{np^2} \right) \end{array} \right\}$$

where, g , h , i and j are the constants to be fitted and f is fixed. T = stop time, nx = horizontal resolution, and np = number of processors. It is noted that the communication time includes memory copy time (the term with the g parameter) involved in the matrix transpose operation.

Application parameters such as nx and T are known and are input to the model. All machine dependent parameters are to be fitted are unknown. Here f represents the number of cycles proportional to the simulation time, g represents the per word memory-to-memory copy time, i represents the message latency, j represents the message bandwidth, and h represents non-linearity due to message contention. The value of f , which is obtained from the computation model, was fixed equal to 4965 [7]. In the original model, these fitted machine parameters were bounded based on acceptable values for the IBM P690 architecture (see [7] for details) as shown in Table 2.

The observed communication times ($t_{observed}$) used in this process are called the “training data.” The combined GA/GP procedure is depicted in Figure 5. The parameters nx , np , T and the measured communication time are the inputs to the current model for estimating t_{comm} . GA searches for the new values for the model parameter g , h , i and j in the original model t_{comm} . GP searches over symbols and constants to construct the error correction term. These two searches in concert provide the improved communication time model that tends to minimize the error.

2.3 EVH1 Model Results

A total of 32 data points used in [7] are used for demonstrating the application of MECP. Half the data was used for fitting and the other half for testing the prediction accuracy.

Table 2 summarizes the GA/GP parameters and settings used in MECP. As GA and GP are based on probabilistic operators, the robustness of the procedure was first tested for

30 random trials for each of the four cases [Cases-1, 2, 3 and 4 (see Table 1)]. The procedure did not converge in four out of the 120 trials, and they were excluded from these calculations. These results are based on a validation data set consisting of 16 data points that were used for testing the predictive capability.

We tested the fifth variation where we exclude the original model and just use the *ect* as a model. Although this search produced very few good results, it is not robust. For instance, out of 50 random trials, this variation produced only 3 good results. Hence we conclude that MECP requires an approximate model to start with.

A comparison of the prediction errors for the original model and the best models obtained (out of the random trials) for the four cases are shown in Figure 6. This figure shows that the best solutions for all four cases outperform the original model. Also, case 1 results in the best model with an error less than half of the original model. The frequency of each of the cases performing better than the original model in the random trials is compared in Figure 7. This gives an estimate of the robustness of each approach. This shows that cases 1, 2, 3 and 4 out perform the original model in 77%, 73%, 50% and 90%, respectively, of the trials, indicating that case 4 is the most robust. From these two comparisons, we observe that case 4 overall outperforms the other three cases.

The predictive capability of the best model found for each case is shown in Figure 8. This figure compares for the 16 validation data points the predicted values from the original model and the new model. The observed values are also shown in this figure. Note that the timings (y-axis) are in log scale. The data points on the x-axis represent different problem instances (i.e., different sizes or different number of time steps) and different processor

counts. Generally, the larger data point numbers (>4) correspond to larger problem sizes or larger processor counts hence exhibiting larger communication times. Figure 8 shows that while the original model prediction is better for smaller data point numbers (corresponding to smaller communication times), models obtained using MECP are better for larger data point numbers (for larger communication times). From a modeling point of view, a model that performs well for larger communication times is more valuable as these times contribute more to the overall time. Therefore we conclude that the models obtained using MECP are more useful.

Table 2. GA/GP parameter settings for MECP

Parameters	Settings
Function set	+, -, x, /, ^, log, exp
Terminal set	R, nx, np, T, tcomm
Population	3000
Max. No. Generations	100
Crossover	90 %
Mutation	10 %
Selection Strategy	Graduated Elitist
Max. Initial depth	7

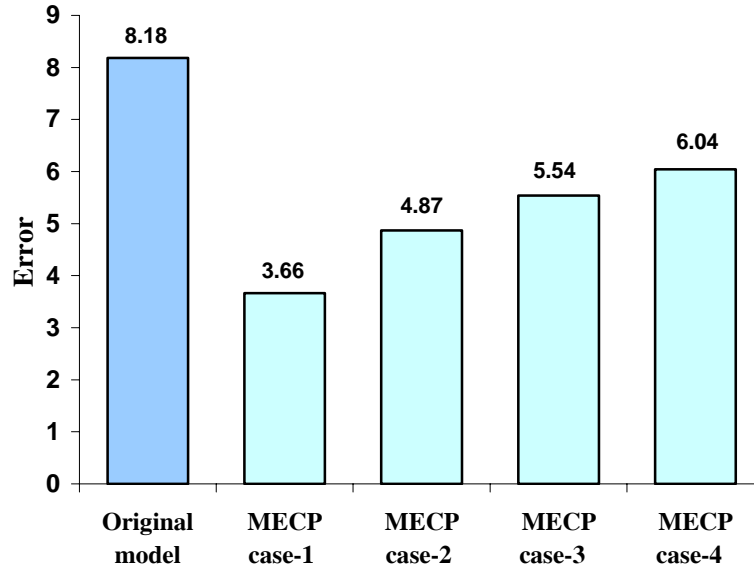


Figure 6 Prediction errors for original model and four cases of MECP

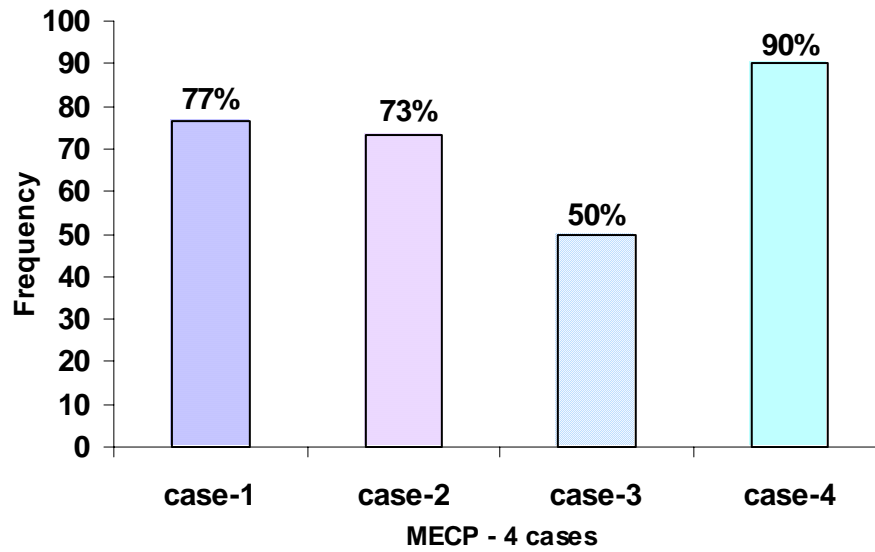


Figure 7 Frequency of improved prediction than the original model in the random trials

Table 3. Summary of fitted values for model parameters

Fitted Parameters	Original model prediction	Lower bound	Upper bound
<i>g</i>	2.45E-08	1.00E-09	1.00E-03
<i>h</i>	1.13216	1	1.5
<i>i</i>	1.69E-05	1.00E-09	1.00E-03
<i>j</i>	1.62E-07	1.00E-09	1.00E-03

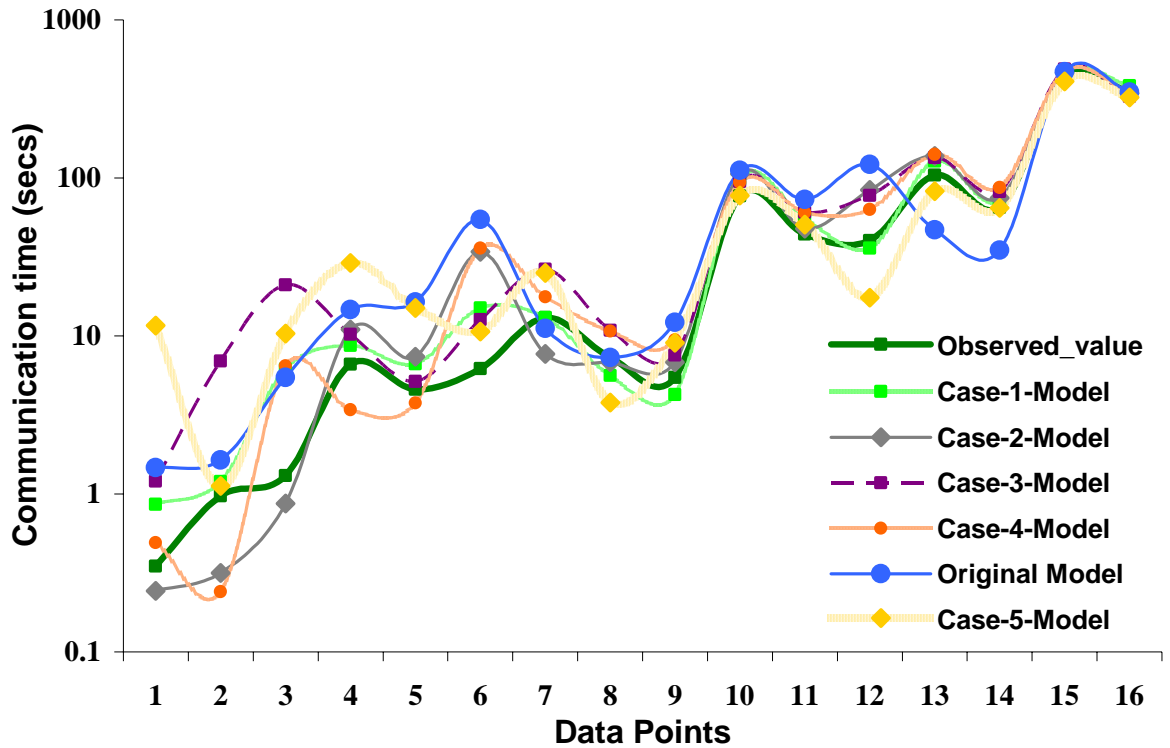


Figure 8 Comparison of the observed values and predicted values by the original model and the best models for Cases-1, 2, 3 and 4

3. EXTENSION OF MECP TO SECOND APPLICATION GYRO

GYRO [[6], [8]], is an Eulerian gyrokinetic-Maxwell solver, computes numerical simulation of tokamak micro-turbulence solving time-dependent, nonlinear equations. We collected the total run time data for GYRO by running the code for the three standard problems B1-std, B2-cy, B3-gtc and for different machine parameters such as number of processors. The mpiP (MPI Profiling) was run for each problem and for each processor number. The communication time was calculated as the MPI percentage of total runtime. We developed a semi empirical model as described earlier.

3.1 Computation Timing Model

We developed a semi empirical model for computation time for GYRO application by considering the input parameters that contribute significantly to the computation time (grid dimensions, time steps etc.). The model for computation time can be expressed as:

$$t_{comp} = \frac{1}{np} \cdot ts \cdot \left\{ Tg^a \cdot rg^b \cdot tc \right\} \cdot k$$

In the above equation, a , b and tc are constants to be fitted and k is a known constant, np = number of processors, ts = time step, Tg = toroidal grid, rg = radial grid, $k = 1$ for the B1-std and B2-cy problems, and $k = 2$ for the B3-gtc problem. The three parameters a , b , and tc were estimated using a single measured computation time from each problem (B1-std, B2-cy, and B3-gtc). Thus, we had three unknowns, three equations, and three data points. We solved for these unknown parameters using the standard nonlinear solver, *fsolve* in MATLAB. Then

the model was validated for the rest of 23 data points. Figure.10 shows the model results. It is clear this semi-empirical model has excellent predictive capabilities for computation time and MECP is not needed (see flowchart in Figure.4).

3.2 Communication Timing Model

A semi empirical communication model was developed using the standard formula:

$$t_{comm} = no.cycles \times \left[\left(\frac{Message_size}{Bandwidth} \right) + Latency \right] \times no.processors^e$$

Where no. of cycles represents the total number of messages and e is an exponent that accounts for message contention. The above equation could be used in an expanded as follows.

$$t_{comm} = k \cdot ob \cdot np^e \cdot nt \cdot \left\{ \left(rg \cdot ((tg + pg) \cdot eg) \cdot Tg \right) \cdot \left(\frac{1}{np} \right) \cdot \left(\frac{1}{bw} \right) + tl \right\}$$

In the above equation, e is a constant to be fitted, tl = latency, bw = bandwidth, np = number of processors, ob = orbit grid, ts = time step, tg = trap grid, Tg = toroidal grid, pg = pass grid, eg = energy grid, rg = radial grid, and k = a known constant that could be different for

each of the three problems. Again, the model parameter values were obtained by using Levenberg-Marquadt procedure using the *lsqcurvefit* function in MATLAB.

3.3 GYRO Model Results

Figure.10 shows the computation time model predictions. Since the semi empirical computation model predicted well for the rest of the 23 data points, we did not apply MECP to this model. Figures 11 and 12 show the communication time model predictions. The semi empirical model, which is also called as original model in the graph, fitted and predicted well for B1-std and B2-cy problems. Again for fitting and prediction, the data was halved just as in the EVH1 study. Out of 24 data points, 12 were used for fitting and 12 were used for prediction. For the B3-gtc problem, the model failed to fit and predict. This is because the message size is bulky for the B3-gtc problem. The application parameter radial grid contributes to the message size. The message size for the B1-std and B2-cy problems are similar and smaller in size; however, the B3-gtc problem has nearly three-times larger message size. Also, the processor mode starts from 16 for the B1-std and B2-cy problems and from 64 for the B3-gtc problem. The message sizes are tabulated in Table.4. All these factors cause the failure of GYRO model to fit and predict for the B3-gtc problem. Since we were interested in a single model that predicts well for all three problems of GYRO, we applied MECP to improve the prediction performance of this model. As in EVH1, we tested all 4 cases (See Table 1) of MECP for the GYRO communication model. All four cases performed nearly equally well (Figures 11 and 12). Totally 15 random trials were run for each case. Out of these 15 trails, in terms of frequency, case-1 showed improved results than the original model 100% of the time, case-2 93%, case-3 83% and case-4 73% of the time.

The best model out of 15 trials for each case and the frequency of improved prediction for each case is shown in Figure 9. In terms of error reduction, when compared to our original semi empirical model, case-1 best model showed 82% reduction in error, case-2 78%, case-3 77% and case-4 75% error reduction. This indicates that for GYRO case-1 is clearly the best even though other cases are not far behind.

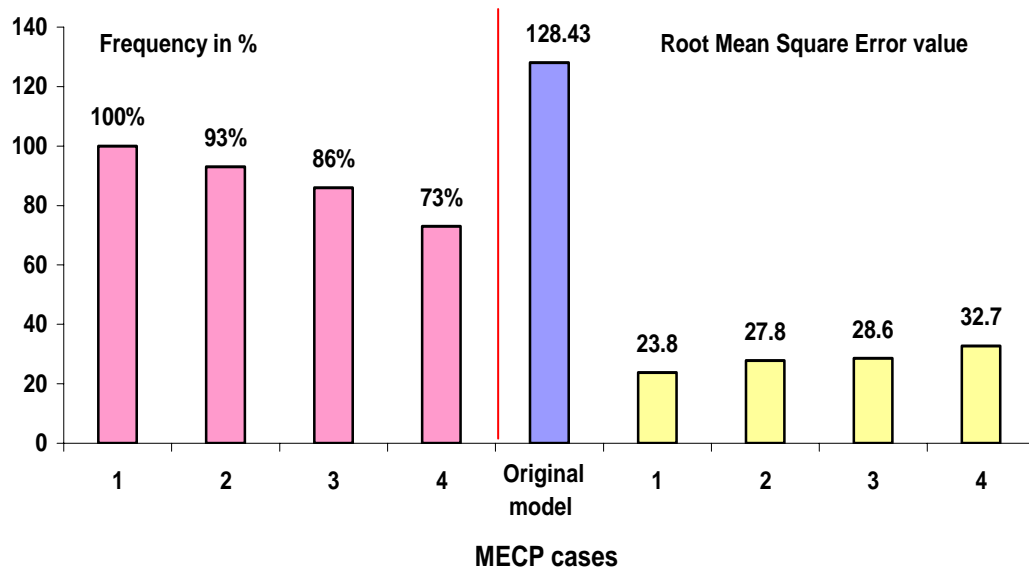


Figure 9 Frequency and prediction errors comparison with Gyro Semi empirical model and 4 cases of MECP model

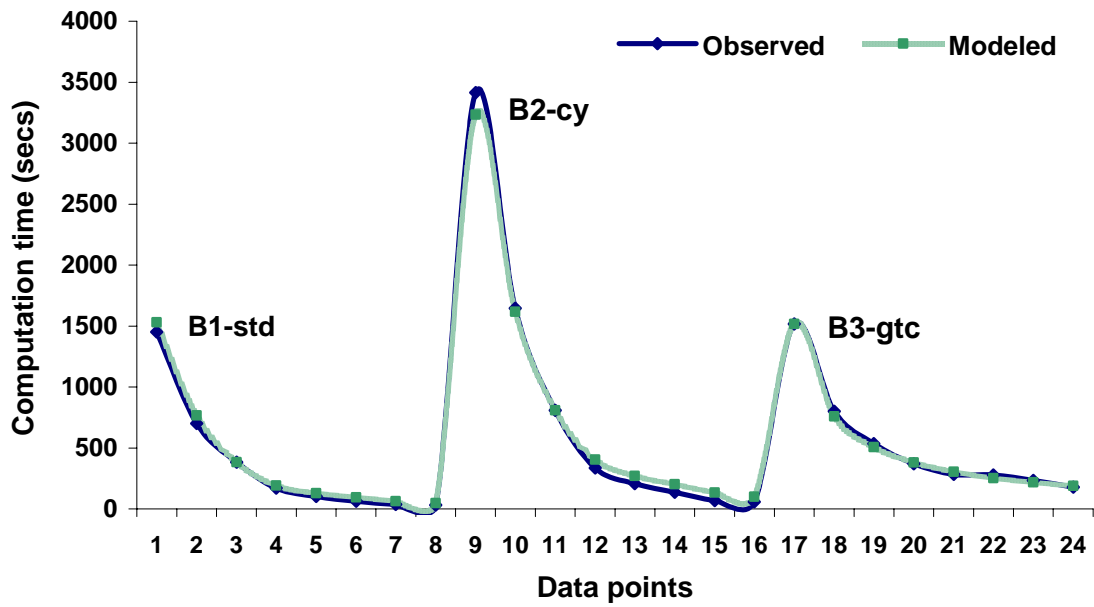


Figure 10 Gyro computation model for B1-std, B2-cy and B3-gtc problems

Table 4. Message size and Processor mode for the three standard problems of GYRO

Message Size			Message Size	
Nprocs	B-std	B2-cy	Nprocs	B3-gtc
16	8960	8192	---	---
32	4480	4096	---	---
64	2240	2048	64	25600
128	1120	1024	128	12800
192	840	768	192	8800
256	560	512	256	6400
384	420	384	384	4400
512	280	256	512	3200

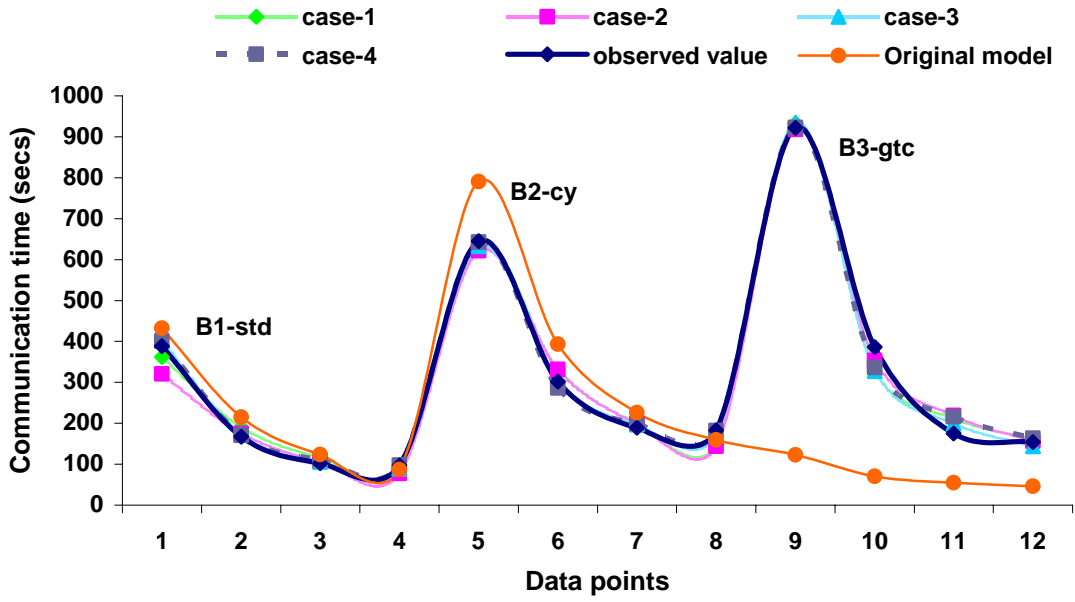


Figure 11 Gyro communication time model for B1-std, B2-cy and B3-gtc problems (for fitting data)

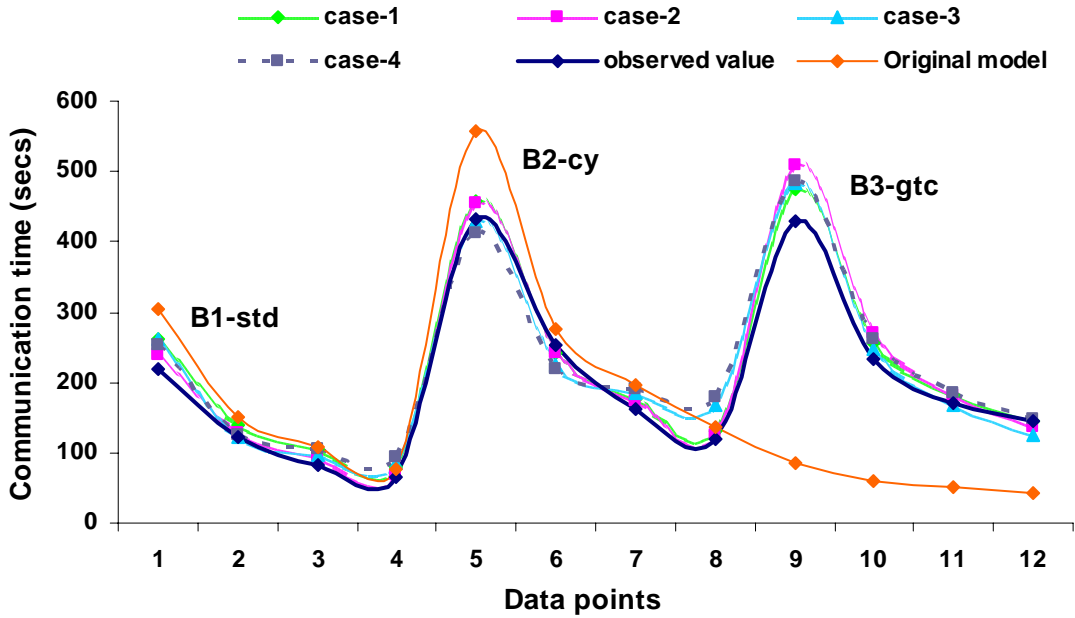


Figure 12 Gyro communication time model for B1-std, B2-cy and B3-gtc problems (for Prediction data)

4. CROSS PLATFORM PREDICTION

In the preceding sections the MECP models were developed and validated using data from a single architecture (IBM P690) for EVH1. The question remains whether these same models could be applied to a different architecture simply by fitting or adjusting only the machine parameters (i.e., latency, bandwidth, etc.) A test of this hypothesis would indicate how well our previously developed models delineated the application specific parameters from machine specific ones. The Teragrid linux cluster at NCSA was chosen for this test as the timing data for both applications were already available for this architecture [[7][8]]. This cluster consists of 1024 Itanium II processors connected by a Myrinet network. We could predict the performance of our models either by fitting the machine parameters or by plugging the published MPI latency and bandwidth values for the Myrinet network. In this paper we demonstrate the use of both approaches. For EVH1, we used the Cheetah model and only the machine parameters were fitted (using the Teragrid data), instead of using published values for the Teragrid architecture. As shown in Figure 13, Case 2 and Case 4 models for EVH1 predict well for the Teragrid architecture. Case 1 and Case 3 predictions are not as satisfactory. This implies that keeping the base model separate from *ect* (as in Cases 2 and 4) yields a better model for cross platform prediction. This is expected as it would be difficult to delineate machine characteristics from application characteristics in a combined model (cases 1 and 3). It is also noted that all cases underpredict data point 1 that corresponds to $np = 1$ which is dominated by the memory to memory copy operation. This means that the fitted parameter value g is too small for the Teragrid architecture. This is expected since very few data points have a strong dependence on this parameter (the first two or three points).

These tests show that MECP approach could be effectively used to predict performance of an application on a completely different architecture by simply changing the appropriate machine values. The only caveat of this approach is that unless existing data is available for fitting, accurate values for machine parameters such as latency and bandwidth, memory to memory copy time, etc can be difficult to obtain.

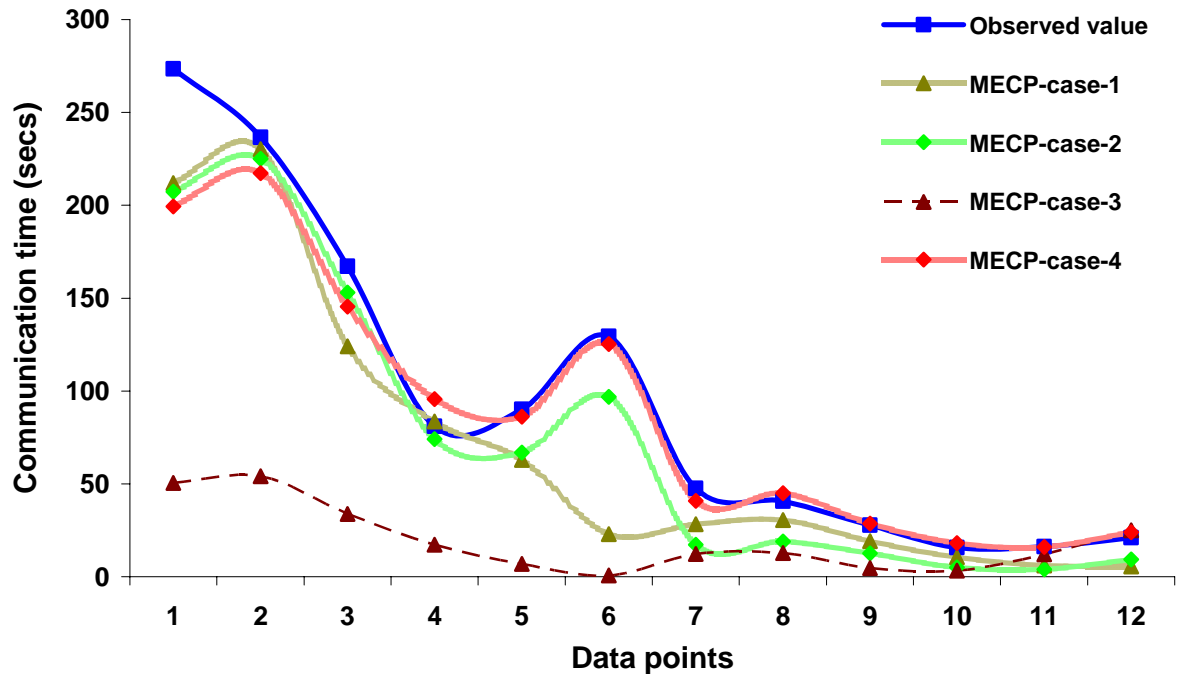


Figure 13 Cross Platform prediction for Teragrid using Cheetah: MECP-based Models for EVH1 application

The MECP-based model was developed for the GYRO application in the previous section using the data from Teragrid architecture. We chose IBM P690 (Cheetah) architecture for testing the cross platform prediction. Unlike EVH1, here we applied the average MPI published values of *tl* and *bw* for IBM P690 to the MECP-based GYRO models.

As shown in Figure 14, all four models predict very well for the IBM P60 architecture, indicating the versatility of these models for cross-platform application.

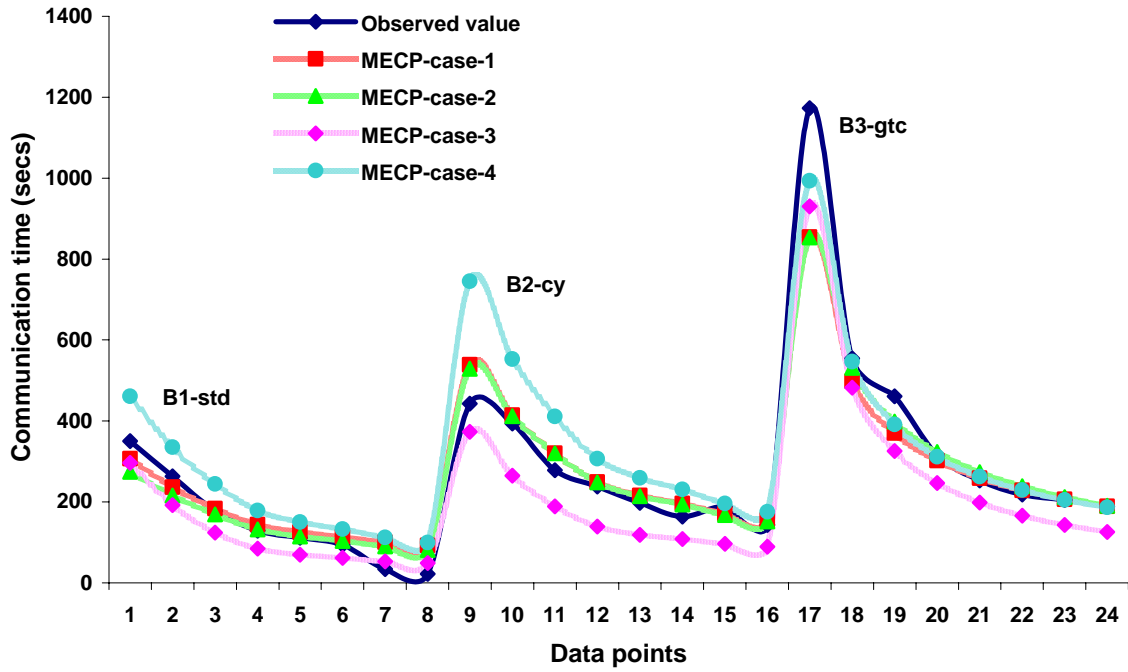


Figure 14 Cross Platform prediction for Cheetah using a Teragrid: MECP model for GYRO application

Table 5. Description of Computer architecture

Architecture	Description
Cheetah at ORNL	Cheetah is a 4.5 TF IBM pSeries, includes 27 p690 nodes, each with 32, 1.3 GHz Power4 processors.
Teragrid at NCSA	IA-64 TeraGrid Linux Cluster consists of 887 IBM nodes: 256 nodes with dual 1.3 GHz Intel Itanium 2 processors and 631 nodes with dual 1.5 GHz Intel Itanium 2 processors

Table 6. Example of function form of *ect*: t_{comm} = semi empirical model

$$ect = \left[\frac{(mc + t_{comm})}{mc - \log(t_{comm} - 0.16589)} \right] \times \left(\left(\left(t_{comm} + (\log(rg) + (mc \times mc)) \right) + t_{comm} \right) + \left(\left(t_{comm} + \log(2 \times t_{comm}) \right) + rg \right)^{(-0.16589)} \right) \left(\left(\frac{(mc + (mc + t_{comm}))}{\log(np)} + t_{comm} \right) \right)$$

$$ect = \left[\frac{nt + t_{comm}}{\left(\frac{nt}{TG + np + rg} - TG \right)} \right] - \left(\frac{TG \times rg}{\log(nt) + np} \times \left(\exp \left(\log \left(\log \left(2 \times \log(rg) \right) \right) \right) \right) \right)$$

5. CONCLUSIONS

The following points summarize the results and conclusions from this paper:

- We have developed and demonstrated a new ‘GA/GP Model Error Correction Procedure’ approach for modeling performance of parallel applications through two HPC applications across two platforms.
- Four variations of this approach were investigated all variations showed improved results than a standard semi-empirical approach.
- The MECP approach captures difficult to model features such as noise, message contention, or I/O contention thus providing improved results than a standard semi-empirical approach.
- This approach augments an existing semi-empirical model either by adding an error correction term (cases 2 and 4) or by including the error correction term in the function (cases 1 and 3).
- Overall, case 1 performed reasonably well for both applications across both platforms even though in several specific instances other models outperformed. This suggests that an integrated model that uses parameter values from the original model will work most of the time.
- While the semi-empirical component of the model contains certain specificities to the target application, the approach used to obtain the error correction term of the model is generic and could be easily used to model performance of other parallel applications.

Therefore we conclude that the MECP approach shows promise for wider applicability in parallel performance modeling.

6. REFERENCES

- [1] David H. Bailey and Allan S. Snavely, "Performance Modeling: Understanding the present and predicting the future," *Proceedings of SIAM PP04, 2005*.
- [2] Dunigan, T.H., M.R. Fahey, J. B. White, and P. H. Worley, (2003). Early evaluation of the Cray X1, Proceedings of SC 2003, Phoenix, AZ.
- [3] Genetic Programming for Subjective Fitness Function Identification. Springer Link-Volume 3003/2004. Title: Genetic Programming: 7th European Conference, EuroGP 2004, Coimbra, Portugal, April 5-7, 2004. Proceedings.
- [4] Gropp, W., Lusk W., and Skjellum A., (1999). Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd edition, The MIT Press, Cambridge, MA.
- [5] Koza, John R.: Genetic Programming II. The MIT Press, Cambridge, Massachusetts (1994)
- [6] M. Fahey and J. Candy. GYRO: A 5-d gyrokinetic-maxwell solver. In *Proceedings of Supercomputing, 2004*.
- [7] Mahinthakumar, G., M. Sayeed, J. Blondin, P. Worley, A. Mezzacappa, and R. Hix (2004). Performance Evaluation and Modeling of a Parallel Astrophysics Application, *Proceedings of the High Performance Computing Symposium 2004*, p. 27-33, Ed: Joerg Meyer, The Society for Modeling and Simulation International, ISBN 1-56555-278-4, Arlington, VA
- [8] P. Worley, J. Candy, L. Carrington, K. Huck, T. Kaiser, G. Mahinthakumar, A. Maloney, S. Moore, D. Reed, P. Roth, H. Shan, S. Shende, A. Snavely, S. Sreepathi, F. Wolf, and Y. Zhang, *Performance Analysis of GYRO: A Tool Evaluation*, in Proceedings of the 2005 SciDAC Conference, San Francisco, CA, June 26-30, 2005

- [9] Parashar, M., and Hariri, S., (2000). Interpretive Performance Prediction for Parallel Application Development, *Journal of Parallel and Distributed Computing*, Vol. 60, No. 1, pp. 17 – 47, January 2000.

- [10] Petrini, F., D. J. Kerbyson, and S. Pakin, (2003). The case of the missing supercomputer performance: achieving optimal performance on the 8,192 processors of ASCI Q, *Proceedings of SC 2003*, Phoenix, AZ.

- [11] Tao Yang, Xiaosong Ma and Frank Mueller, “Predicting Parallel Applications' Performance Across Platforms Using Partial Execution”, *Supercomputing 2005*.

- [12] Vetter, J.S., and A. Yoo, (2002). An empirical performance evaluation of scalable scientific applications, *Proceedings of SC 2002*, Dallas, TX.

- [13] Zechman, E. M. (2005). Improving Predictability of Simulation Models using Evolutionary Computation-Based Methods for Model Error Correction, Ph.D. dissertation, North Carolina State University, Raleigh, NC.