

A Web-based Prophecy Automated Performance Modeling System

Xingfu Wu, Valerie Taylor
Department of Computer Science, Texas A & M University, College Station, TX 77843, USA
Email: {wuxf, taylor}@cs.tamu.edu

Joseph Paris
Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, USA

ABSTRACT

Prophecy system is a performance analysis and modeling infrastructure that allows users to record many different parameters relevant to an application's performance. A key component of Prophecy system is the web-based automated performance modeling system, which allows a developer to quickly gain insight into the performance of an application code or functions within a code such that the developer can cut down on the time required for developing efficient applications, locate potential bottlenecks, and approximate the runtime on different systems. In this paper, we present the design and implementation of the web-based automated performance modeling system, and use parallel matrix-matrix multiplication and NAS parallel benchmark BT as examples to illustrate how to automatically model these parallel applications using online web-based interfaces.

Keywords

Web-based system, performance modeling, scientific applications, and Prophecy system

1. Introduction

Efficient execution of scientific applications requires insight into how the 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 or to explore the performance impact of the use of different algorithms to solve a given task. This paper discusses our effort to automate the process of developing analytical models for scientific applications using web-based interfaces, so as to significantly decrease the time required for model development and to allow users around the world populate the Prophecy database [9] and to share models and application information using web-based interfaces.

The web-based automated performance modeling system is a key component of the Prophecy system [8]. This component must have the following requirements:

robust enough to fit user needs, easy enough to be self-explanatory, and quick enough to produce a graph output of a model within a reasonable amount of time. Currently, the automated performance modeling system provides three different modeling methods: curve fitting, parameterization, and kernel coupling. Curve fitting is a least square fit of the empirical data. This method is the most accessible of all three as it can work purely off the application information that is already contained within the Prophecy database. Parameterization combines manual code analysis with system performance information to generate an application runtime model. The kernel coupling method focuses on how to represent an application in terms of its functions, or kernels, or components.

Today, advanced web technologies let data from databases be merged automatically with templates to produce dynamic standard HTML pages. The Prophecy automated performance modeling system takes advantage of them, and uses the combination of scripts generated in PHP [4] and GNU Octave [3] to query Prophecy database and generate an analytical model. Octave executes the generated scripts and returns a graph of the model, which is then displayed to the user along with the actual model itself. The web-based performance modeling system satisfies the aforementioned requirements and provides the user with an easy-to-use tool for quick model generation and display.

The remainder of this paper is organized as follows. Section 2 provides a background on the Prophecy system. Section 3 presents the detailed design and implementation of the web-based Prophecy automated performance modeling system, and uses parallel matrix-matrix multiplication and NAS parallel benchmark BT [1] as examples to illustrate how to model the applications online. Section 4 concludes the paper.

2. Background: Prophecy System

The Prophecy framework, shown in Figure 1, consists of three major components: data collection, data analysis, and three centralized databases. The data collection component deals with the retrieval and storage of information via one of two ways. The first way involves

automatic instrumentation of code at the levels of basic block, procedures, or functions [10]. The results are then automatically uploaded and stored in the performance database. The second way allows the user to manually input the performance information into the database. The data analysis component allows the user to produce an analytical performance model based on data from the performance database shown in Figure 2, model templates from the template database, and system characteristics from the systems database. The interface to Prophecy system is a set of database-driven dynamic web pages.

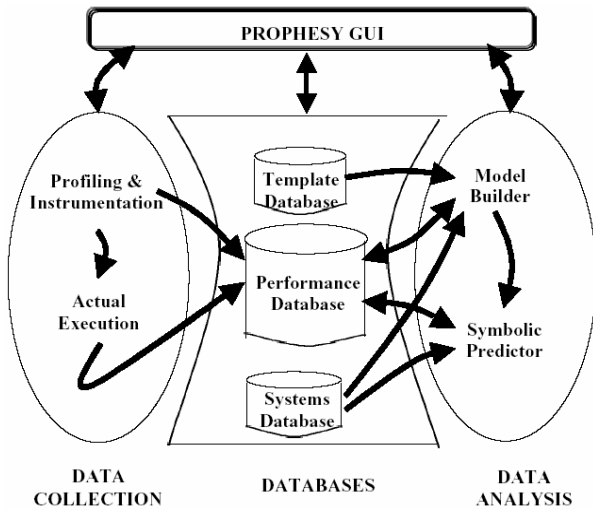


Figure 1: Prophecy framework.

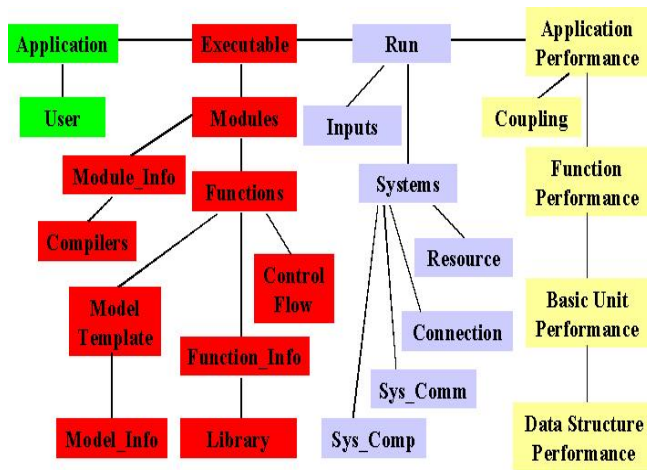


Figure 2: Prophecy database schema.

Before an analytical model can be generated, an application must go through a number of steps. First the code can be instrumented automatically by PAIDE [10], and executed. The performance data shown in Figure 2 is then collected from these runs and automatically uploaded into the database [9] using SOAP [6]. Finally, a model is developed using optimization techniques.

3. Web-based Automated Performance Modeling System

The automated modeling component is a very important tool for aiding in performance evaluation for a given application or kernel code. The main goal of the automated modeling system is to generate an interface that was very easy to use, yet robust enough to satisfy a developer needs. Currently, the performance modeling system uses three modeling techniques: curve fitting, parameterization, and kernel coupling methods. The first two are well-established techniques for model generation, which Prophecy facilitates by automation. The last method is brought about by Prophecy system because of the amount of archived information about an application. Each model type has its own unique set of options and parameters that a user can easily modify to produce large amount of different performance functions. In this section, we discuss three modeling techniques: curve fitting, parameterization, and kernel coupling methods, and demonstrate the implementation and usage of the modeling system in detail.

The interface to Prophecy system is a set of database-driven dynamic web pages [5]. This allows most, if not all, web users around the world to access the system; also, performance prediction can be achieved based on the historical database, which may reduce the time needed to develop efficient software and allows users to share their models with others.

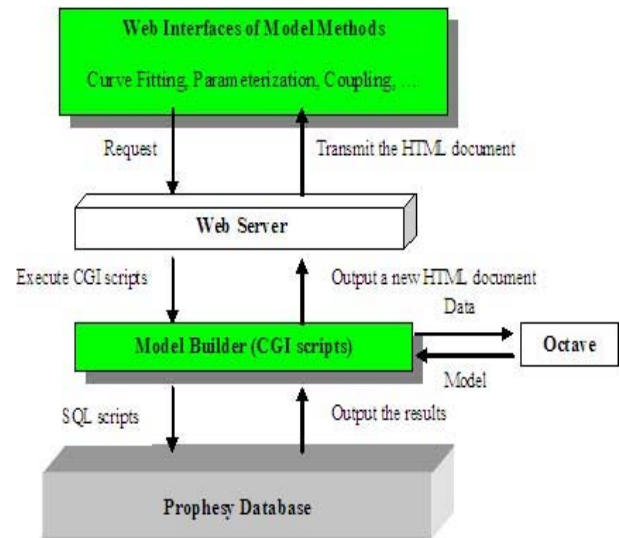


Figure 3: Automated modeling system framework

The framework for the web-based automated performance modeling system is shown in Figure 3. When a user chooses a model method to request analytical model and plot for an application through a web browser, a request is sent to the web server. In turn, the web server services the request by executing a CGI (Common Gateway Interface) program (PHP script), which returns a result to the server, which in turn downloads that result onto the user's browser. If the user

requests a form that must be filled out in order to service a request for data, the request is first sent to the browser which executes a CGI script asking for the correct form, which is then returned to the browser. In the event that the server is processing a filled out HTML form, it invokes the CGI script which makes various calls to SQL scripts, processes and formats the data, sends the data to the Octave to generate an analytical model and plot, then sends the result back up through the server, and is downloaded on the clients browser. This method allows for dynamic HTML pages to be generated based on user requests, and does not require hard coded design into a document.

3.1 Curve Fitting Method

Curve Fitting is a method that uses optimization techniques to develop a model. In this case, the web-based performance modeling system uses a least squares fit to the empirical data to get the graph type with the least norm of residuals if the graph type is not specified. The empirical data to be used is determined by the user via the interface, and then Octave is used to generate the resultant model shown in Figure 4. The weakness in this model is exemplified by the lack of exposure of the system parameters versus the application parameters. The models generated from the curve fitting are generally a function of some input parameters of the application.

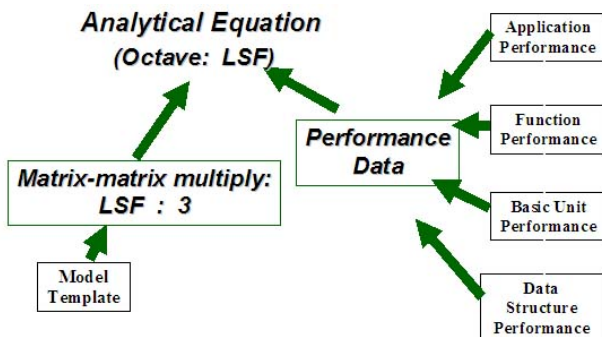


Figure 4. Modeling framework of curve fitting

The following example will cover a high level illustration of the curve fitting method on a matrix-matrix multiplication kernel executed on SGI Origin2000 with 8 processors at Northwestern University (NU). The kernel was executed for different problem sizes, with the matrix rank ranging between 100 and 1000. The interface is split into three steps. Step 1 allows the user to select an application and associated executables. Once the application is selected, the list of corresponding executables is displayed for the user to select one. Step 2 allows the user to set options correlated to the curve fitting method. For example, the user is allowed to select one of 13 different graph types: unknown, linear, quadratic, cubic, 4th-order, 5th-order, 6th-order polynomial, inverse linear, inverse quadratic, inverse cubic, inverse 4th-order, inverse 5th-order, inverse 6th-order polynomial. If the user does not specify a graph type (“Unknown”),

the automated modeling system can specify the graph type with the least norm of residuals. For now, the Y-axis only has one option, runtime (It can be extended to multiple options such as MFLOPS, speedup, and efficiency); however, the X-axis can be either “Number of Processors” or “Inputs.” In the matrix-matrix multiplication example, “Inputs” correspond to the matrix rank. If X-axis is set as “Number of Processors” the user is allowed to select the range of processors used to generate the model; only one input value can be selected for this case. In the alternate case (X-axis is “Inputs”), the user can only select one processor number and a range of input values. The user can also select the end point of the X-axis, allowing for the model to be used to predict the performance for values outside of the range used for model development. Step 3 entails identifying the range of values for the inputs as well as the value for prediction. Figure 5 illustrates the selection for step 3 assuming the X-axis to be “Inputs”: “Number of Processors” is ‘8’; the “Inputs” range used for model development is 100 to 1000; the desired prediction is for a matrix rank of 1200.

Figure 5. Step 3 of curve fitting model.

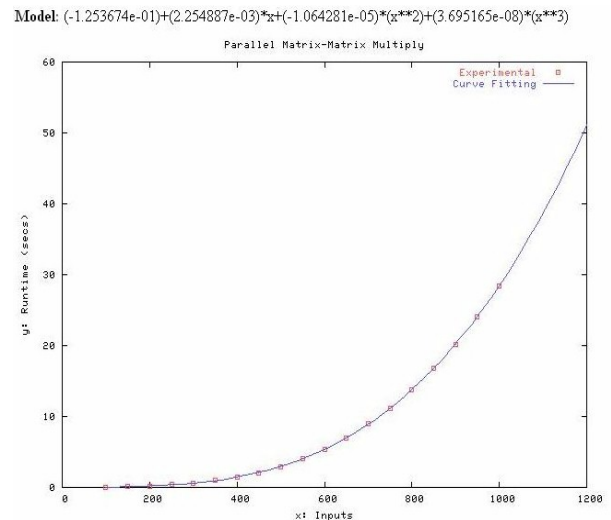


Figure 6. Cubic curve fitting of the data.

When the user submits this form, the model is generated using Octave, and the result is displayed to the user. The resultant model for the matrix-matrix multiplication example using the cubic graph type is given in Figure 6, with the model made explicit to the user. The empirical data based upon the execution of the

application code is given along with the generated graph. Figure 6 illustrates a good match of the analytical model to the experimental data.

3.2 Parameterization Method

Parameterization combines manual code analysis with system performance measurements to generate a performance model of the application on a given system. The manual code analysis is the hand counting of operations within a code. It is assumed that manual analysis is done for kernel codes with a reasonable number of lines of the code and done only once per code. The idea stems from the fact that a few major kernels can be found in a large number of applications. Hence, detailed analysis of a small number of kernels can aid in the analysis of many applications. The manual analysis allows for explicit representation of system parameters as well as application parameters in the analytical model, thereby allowing one to use the resultant model to explore different systems or application features.

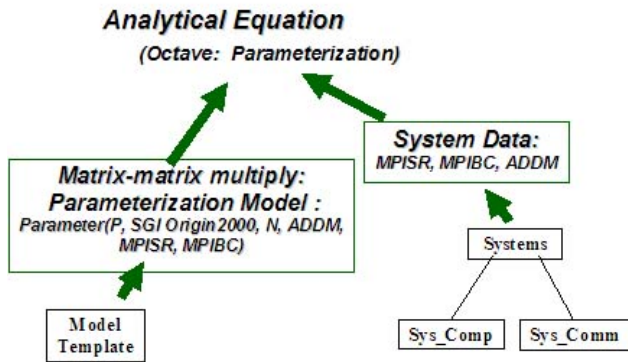


Figure 7. Modeling framework of parameterization

Again, we use matrix-matrix multiplication application, which can be considered a kernel, to provide the web-based implementation of the parameterization method shown in Figure 7. The template database holds a string representation of the model to be used. For example, the following model, corresponding to a cubic model, is stored in the template database for the matrix-matrix multiplication kernel: $\alpha_1 N^3 + \alpha_2 N^2 + \alpha_3 N + \alpha_4$ (N is the matrix rank). Further, each of the α terms is expanded within the template database. For a Master-Slave implementation of the matrix-matrix multiplication kernel (whereby the rows of the multiplicand matrix are distributed among the P-1 slaves and the columns of the multiplier matrix are broadcasted to the P-1 slaves), the following parameterization model is stored in the database, with the coefficients in front of the N terms corresponding to the α terms:

$$(1/(P-1) * T_{comp}) N^3 + ((2/(P-1)) * T_{sr} + T_{bc}) N$$

After a user specifies the desired system for the analytical model, the values for α terms are determined by information found within the systems database. The systems database contains the communication times for

MPI Broadcast (T_{bc}) and MPI Send/Receive (T_{sr}) as well as execution times for a variable amount of core computations (T_{comp}). The communication times for the MPI calls were obtained using MPPTTEST performance test suite [2]. For the matrix-matrix multiplication code, we use the following core computations: integer add ($a + b$) for single dimensional array indexing, integer multiply add ($a * b + c$) for multi-dimensional array indexing, and floating point multiply add ($a * b + c$) for the actual computation.

The web-based user interface for the parameterization method requires two distinct steps. Step 1 allows a user to select an application from the database and the desired system. The application list is generated by checking the template database for the lists of applications that have parameterization models. The desired system list is generated by checking the system database for the list of systems with detailed system performance data. Step 2 allows the user to specify distinct values or ranges for the inputs to the kernel. The interface for step 2 is given below in Figure 8; the interface for step 1 is very similar to that given for curve fitting method.

| Executable Name | System Name |
|---|---|
| matmat_sgi | IBM SP2 |
| Model Type | |
| parameter(N, SGI Origin2000, P, ADDM, MPISR(L), MPIBC(L)) | |
| Optimization Method | |
| ADDM*N ³ /(P-1)+2*N*MPISR(L)/(P-1)+N*MPIBC(L) | |
| Parsed Method | |
| IBM SP2: 0.00000004830*N ³ /(P-1)+2*N*0.001*(15.94+0.0608*L)/(P-1)+N*log(P) * 0.001*(15.94+0.0608*L) | |
| User Defined (range: min, max) | |
| N | <input type="text" value="100"/> <input type="text" value="10000"/> |
| P | <input type="text" value="8"/> <input type="text" value="64"/> |
| L | <input type="text" value="1024"/> |
| <input type="button" value="Submit"/> | |

Figure 8. Step 2 of the Parameterization Interface

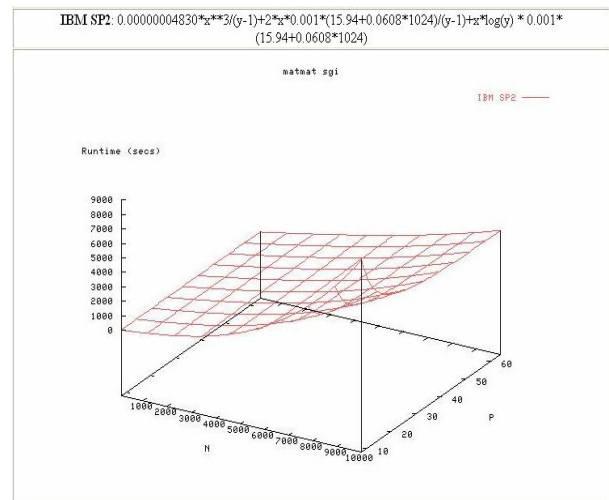


Figure 9. 3D Output of parameterization method.

In the event that the user selects two input variables to have a range, a 3-D plot is generated. Figure 9 gives the resultant output of an example of the 3-D output for

the matrix-matrix multiplication kernel, with the fixed 1KB messages, the matrix rank (N) having a range from 100 to 10000 and the number of processors (P) having a range from 8 to 64. This shows the application performance trend on IBM SP2.

3.3 Kernel Coupling Method

The kernel coupling method focuses on how to represent the performance of a function in terms of its kernels or components [7, 11]. When developing performance models of an application, it is extremely useful to understand the relationships between the different functions that compose the application. Basically, the goal is to determine how one kernel affects another, identifying if the relation is beneficial or detrimental. The coupling method encapsulates this relation into coefficients that can be used to develop analytical models. As an example, assume we have an application that is composed of three kernels in a loop --- kernel A, kernel B, and kernel C. Also, assume that we analyzed each kernel and produced *modelA*, *modelB*, and *modelC* respectively. The coupling method provides the coefficients for the following model:

$$T = \psi \text{ modelA} + \zeta \text{ modelB} + \kappa \text{ modelC}$$

where the coefficients ψ , ζ , and κ represent the performance relations among the three kernels. Kernel coupling is used to generate the coefficients in the following manner. First, the coupling value, C_{AB} , is generated as the ratio of the measured performance of the kernels A and B together to the expected performance resulting from combining the isolated performance of each kernel. Hence, the coupling value is generated according to the following equation:

$$C_{AB} = \frac{P_{AB}}{P_A + P_B}$$

where P_A is the performance of kernel A alone, P_B is the performance of kernel B alone, and P_{AB} is the performance of kernel A and kernel B executed together. The coupling values are stored in the database for the different pairs of kernels.

Once generated, the coupling values are used to generate the coefficients in the equation above for T as a weighted average of the appropriate coupling values as given below:

$$\psi = \frac{(C_{AB} \times P_{AB}) + (C_{CA} \times P_{CA})}{P_{AB} + P_{CA}}$$

$$\zeta = \frac{(C_{AB} \times P_{AB}) + (C_{BC} \times P_{BC})}{P_{AB} + P_{BC}}$$

$$\kappa = \frac{(C_{BC} \times P_{BC}) + (C_{CA} \times P_{CA})}{P_{BC} + P_{CA}}$$

When using a weighted average, it is assumed that at the time the coupling values were created the number of

times the kernel is executed is fixed, and all the coupling values have the same input.

In the following paragraphs, we use the kernel coupling to implement web-based automated performance modeling shown in Figure 10. The coupling modeling interface requires a three-step approach. Step 1 entails selecting an application. A list of applications is generated based on information contained in the coupling table. When a user selects an application, the form is automatically submitted, and the step is redisplayed containing the list of systems associated with the application. The user then selects a system and submits the form.

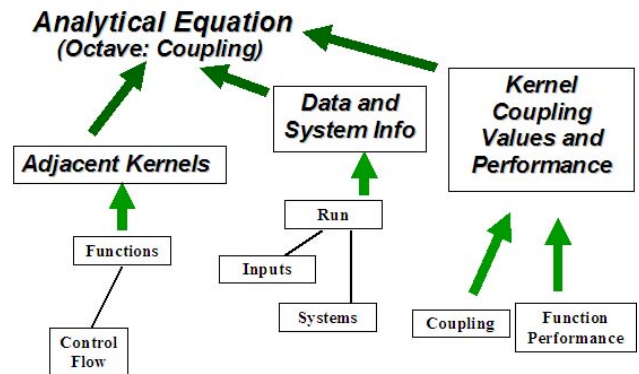


Figure 10. Modeling framework of kernel coupling

| | |
|---------------------------------------|---|
| Application | NAS Parallel BT Benchmarks |
| System | IBM SP2 at ANL |
| Model Type | Coupling |
| Select Kernel Models | |
| copy_faces | Unknown <input type="button" value="v"/> |
| x_solve | Unknown <input type="button" value="v"/> |
| y_solve | Unknown <input type="button" value="v"/> |
| z_solve | Unknown <input type="button" value="v"/> |
| add | Unknown <input type="button" value="v"/> |
| Final | Unknown <input type="button" value="v"/> |
| Initialization | Unknown <input type="button" value="v"/> |
| Arrangement | Number of Processors <input type="button" value="v"/> |
| <input type="button" value="Submit"/> | |

Figure 11. Step 2 of coupling model.

From the application information the list of kernels is retrieved from the database. At this point the user selects the type of modeling system method (curve fitting or parameterization) for each kernel. The user also identifies the setup for the generated plot, identifying if "Inputs" or "Number of Processors" should be used along the X-axis. Figure 11 illustrates step 2 of the coupling model builder. The Prophecy automated modeling system provides 13 different graph types: unknown, linear, quadratic, cubic, 4th-order, 5th-order, 6th-order polynomial, inverse linear, inverse quadratic, inverse cubic, inverse 4th-order, inverse

5th-order, inverse 6th-order polynomial. If the user does not specify a graph type (“Unknown”), the modeling system can automatically specify the graph type with the least norm of residuals.

| Number of Processors | Inputs | Processor Prediction |
|---|--|---------------------------------|
| <input type="checkbox"/> 4 <input checked="" type="checkbox"/> 9 <input type="checkbox"/> 16 <input type="checkbox"/> 25 | <input type="text" value="24.000000"/> <input type="text" value="64.000000"/> | <input type="text" value="81"/> |
| <input type="button" value="Submit"/> | | |

Figure 12. Step 3 of coupling model.

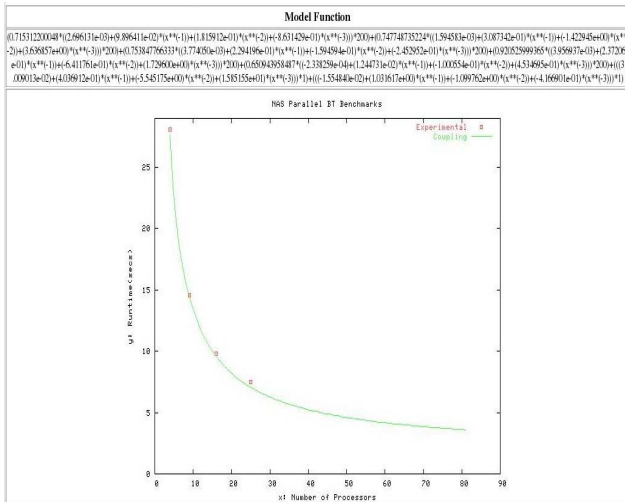


Figure 13. Example output of coupling method

Step 3 shown in Figure 12 entails identifying the range of values for the inputs as well as the value for prediction, as done with curve fitting method. For example, if the arrangement type is “Number of Processors”, the user is allowed to select a range of processors (4, 9, 16, 25) in the column “Number of Processors” for a given fixed input size (24: problem size of 24x24x24) in the column “Inputs” and to input the number of processors (81) in the column “Processor Prediction” for predicting the application execution time on the number of processors (81).

Once the user has chosen among all the options, the model is generated. Figure 13 shows the results of all three steps of the coupling modeling system for the NAS Parallel BT benchmark, which has 7 kernels: *Initialization*, *copy_faces*, *x_solve*, *y_solve*, *z_solve*, *add* and *Final*. The model function and its plot are automatically generated. The predicted execution time on 81 processors is very close to the actual execution time. Of course, the more data we use to populate the Prophecy database, the more accurate the predicted result is.

4. Conclusions

This paper presented the web-based Prophecy automated performance modeling system in terms of the following

modeling techniques: curve fitting, parameterization, and kernel coupling. The web-based implementation of the three techniques was discussed in detail with examples illustrating the use. The curve fitting method allows users to easily generate models based on a least squares fit to empirical data; the disadvantage is the lack of exposure of system parameters. The parameterization method allows a user to develop a more robust model and tweak system as well as application parameters. The coupling method allows for insights into the relationships between the kernels that compose the application. The web-based automated performance system allows users to gain insight into how their applications are performing, allows them to predict how their applications may run on other systems, and generate online models using any one of three types of methods. It also allows users around the world populate the Prophecy database and to share models and application information using web-based interfaces.

References

- [1] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow, *The NAS Parallel Benchmarks*, Tech. Report NAS-95-020, 1995. <http://www.nas.nasa.gov/Software/NPB/>.
- [2] MPPTTEST --- Measuring MPI Performance, <http://www.mcs.anl.gov/mpi/mpptest>.
- [3] GNU Octave, <http://www.octave.org/>.
- [4] PHP: Hypertext preprocessors, <http://www.php.net/>.
- [5] Prophecy project, <http://prophecy.cs.tamu.edu>
- [6] SOAP, <http://www.w3.org/TR/soap/>
- [7] Valerie Taylor, Xingfu Wu, Jonathan Geisler, and Rick Stevens, Using Kernel Couplings to Predict Parallel Application Performance, in *Proc. of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC2002)*, July 2002.
- [8] Valerie Taylor, Xingfu Wu, and Rick Stevens, PROPHECY: An Infrastructure for Performance Analysis and Modelling System of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, Volume 30, Issue 4, March 2003.
- [9] Xingfu Wu, Valerie Taylor, J. Geisler, X. Li, Z. Lan, R. Stevens, M. Hereld and I. R. Judson, Design and Development of the Prophecy Performance Database for Distributed Scientific Applications, in *Proc. of the 10th SIAM Conference on Parallel Processing for Scientific Computing*, SAIM, 2001.
- [10] Xingfu Wu, Valerie Taylor and Rick Stevens, Design and Implementation of Prophecy Automatic Instrumentation and Data Entry System, in *Proc. Of the 13th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2001.
- [11] Xingfu Wu, Valerie Taylor, Jonathan Geisler, and Rick Stevens, Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance, *the 18th International Parallel and Distributed Processing Symposium (IPDPS2004)*, Santa Fe, New Mexico, April 26-30, 2004.