PERFORMANCE PREDICTION FOR PARALLEL SCIENTIFIC APPLICATIONS

Rafiqul Zaman Khan
Information & Computer Science Department
King Fahd University of Petroleum and Minerals
P.O.Box: - 1394, Dahran -31261
K. Saudi Arabia
email: rafiq@ccse.kfupm.edu.sa

Abdul Quaiyum Ansari
Department of Computer Science
Faculty of Management Studies & IT
Jamia Hamdard, New Delhi -62, India
email: aqansari@ieee.org

Kalim Qureshi
Information & Computer Science Department
King Fahd University of Petroleum and Minerals
P.O.Box: - 30, Dahran -31261
K. Saudi Arabia
email: qureshi@ccse.kfupm.edu.sa

ABSTRACT

In this paper we studied performance predictions for parallel scientific applications on a homogeneous cluster of workstations. Performance prediction is important for analyses of the scalability of parallel applications and the estimation of the processing time for the application in a loaded/unloaded environment. We developed ***A****utomatic* ***P****arallel* ***A****pplication* ***P****rediction* ***S****ystem (****APAPS****) for cluster computing environments. Here we are reporting the accuracy of the APAPS using two scientific applications. The measured result shows that APAPS has high prediction accuracy.*

Keywords: ***Performance Tool, Prediction of Resources, Computational & Communication Modeling, Performance Predictions, Parallel Program Performance Predictions***

1.0 INTRODUCTION

Message passing parallel programs that run on a cluster of workstations are often used to solve problems that would take too long to run on conventional sequential machines. Understanding and predicting the performance of such programs is a complex task. It is important to understand the performance characteristics of computation, communication and message passing software used for parallelization [1].

One of the keys to making useful, accurate performance predictions in such situations is to have a clear picture of the communication characteristics of the application. There are many hurdles to obtaining useful, accurate performance measurements of communications operations for parallel computers and applications.

Performance prediction is an important tool for the performance analysis and debugging of scalable parallel applications [2]. A performance prediction system models the program performance as a function of the hardware and software parameters of the system. The execution time of a program can be predicted by changing the software and hardware parameters in the model on a variety of platforms and configurations. Performance prediction tools have been implemented for several high-level parallel languages [3, 4].

Due to the popularity of workstation clusters as a parallel computing platform, there is a growing need for performance tools that support these platforms. One of the most common ways to write parallel programs for a workstation cluster environment is to use a sequential language augmented with a message-passing library such as Message Passing Interface (MPI). Although there are some performance debugging tools that work with MPI programs, to our best knowledge there are no tools capable of accurate performance prediction.

This paper presents the APAPS model as a performance prediction tool designed to solve many MPI programs. Performance prediction tools and performance debuggers are designed to solve different types of problems. For example, a debugging tool such as Paradyn [5] would be well suited to discovering deadlocks in a parallel algorithm, while a performance prediction tool such as APAPS would be better at assessing an application's sensitivity to network latency.

2.0 RELATED WORK

There are several performance debugging tools that support MPI. The Paradyn [5] performance tools, which were designed for runtime performance monitoring and bottleneck detection, have been ported to the MPI platform. Other debuggers are available, including [6, 7] and [8].

Performance prediction tools and performance debuggers are designed to solve different types of problems. For example, a debugging tool such as Paradyn would be well suited to discovering deadlocks in a parallel algorithm, while a performance prediction tool such as one similar to APAPS would be better at assessing an application's sensitivity to network latency.

3.0 PERFORMANCE MODELS

Performance Models are used to represent the performance aspects based on the hardware and software of the system. This section describes the performance models used to represent different performance aspects of different parallel programs. In our designed system several performance models were used which are discussed in the following subsections.

3.1 Computation Model

In our computational model computational costs are assigned to each operation. To model computation costs, we assign a cost factor C_i, measured in execution time, to each of the N_{ops} different operations that may be performed during program execution. We use simple benchmark programs to obtain values and confidence intervals for each C_i on hardware platforms of interest. It is observed that, on the workstations used, some operations have significantly different costs than others. For example, a division operation takes over twice as long as a multiplication and some integer operations take significantly longer than their floating-point counterparts.

Given the cost factors C_i, the predicted computation time T_{comp} spent during program execution can be obtained by the equation given below:

$$T_{comp} = \sum_{i=1}^{N_{ops}} C_i \, N_{op(i)} \tag{1}$$

Where $N_{op(i)}$ is the number of times operation i is performed during program execution.

3.2 Communication Model

The communication model developed is used to represent the performance of MPI communications over Ethernet. Similar models could be developed for other networking technologies such as Asynchronous Transfer Mode (ATM). Many researchers who have modeled the performance of inter-processor communication report that a simple model that accounts for message latency and network bandwidth gives adequate results [10]. According to this model, the predicted communication time T_{comm} spent during program execution can be expressed in the equation given below:

$$T_{comm} = \sum_{i=1}^{N_c} \left(\alpha + \frac{B_i}{\beta} \right) \tag{2}$$

where N_c is the total number of communications per processor, α is the message latency, β is the network bandwidth, and B_i is the size of message i. With shared-medium networks such as Ethernet, contention for bandwidth can significantly affect network throughput. Therefore, to model the performance of MPI communications over Ethernet, a contention factor is added to Equation (2),

$$T_{comm} = \sum_{i=1}^{N_c} \left(\alpha + \frac{\gamma \times B_i}{\beta} \right) \tag{3}$$

where a contention factor of $\lambda = P$, in which P is the number of processors, gives a good approximation of Ethernet contention, assuming that all P processors are communicating simultaneously.

3.3 Parallel Application Model

Adding together equations 1 and 3 provides a general equation for predicted execution time.

$$T_{total} = T_{comp} + T_{comm} = \sum_{i=1}^{N_{ops}} C_i \times N_{op(i)} + \sum_{i=1}^{N_c} (\alpha + \frac{\lambda \times B_i}{\beta}) \tag{4}$$

In this equation $N_{op(i)}$, N_c and B_i vary with the application, the problem size N, and the number of processors P. An *MPI* application model consists of equations for these parameters expressed as functions of N and P.

Setting λ to 1.0 effectively provides the communication Equation (2) for local area connections. By adjusting the value of λ Equation (4) may be used for a variety of network connections.

4.0 SYSTEM STRUCTURE

The APAPS system is designed based on several assumptions. First, the system is restricted to the set of programs that conform to the SPMD (single program, multiple data) model. As many of the scientific problems are amenable to the SPMD model, this is not a serious limitation [11]. Another assumption concerns the homogeneity of the workstations used in the cluster of parallel machines. In our mode, al homogeneous environment is considered to reduce the complexity of performance prediction. A block diagram of the APAPS system is shown in Fig 1. The performance prediction process consists of three main phases: static analysis, dynamic analysis, and prediction. These are explained in detail in the following subsections.

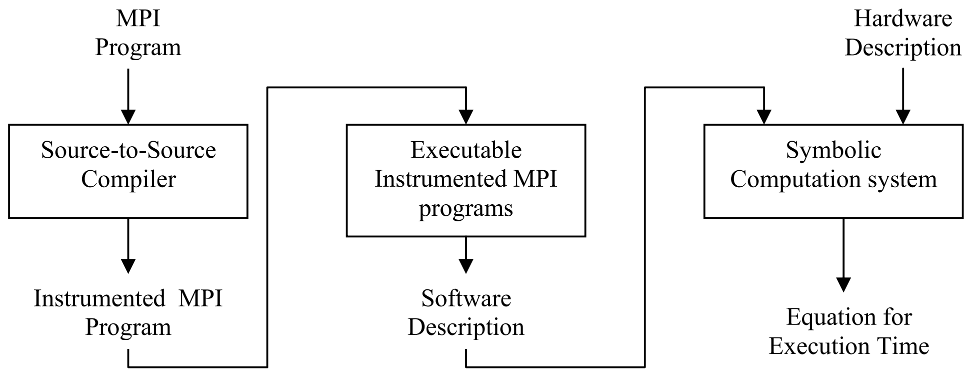

MPI Program
Source-to-Source Compiler
Instrumented MPI Program
Executable Instrumented MPI programs
Software Description
Hardware Description
Symbolic Computation system
Equation for Execution Time

Fig. 1: Structure of the APAPS Performance Prediction System

4.1 Static Analysis

In the static analysis phase, the source-to-source compiler constructs a call graph of the MPI program. For each basic block, the compiler records the number of times that each type of operation occurs in that block, loop initialization expressions and termination conditions, and the number of calls to message passing routines. For the example code of a sample C++ program, the call graph is shown in Fig. 2.

In addition to the construction of a call graph of the program, the source-to-source compiler produces an instrumented version of the MPI program for use during the dynamic analysis phase. The instrumentation code consists of statements to count the number of times each basic block of the program is executed, instrumented versions of the MPI communication routines that record the size of each message transmitted, and additional library routines used during dynamic analysis.

```
#include <stdlib.h>
class A
{
 public:
      void a( );
      void b( );
};


int main( )
{
      A t;
      t.a( );
      t.b( );
      exit(0);
}
```

$$\begin{tikzcd} & \text{A::a} \\ \text{main} \arrow[ru] \arrow[r] \arrow[rd] & \text{exit} \\ & \text{A::c} \end{tikzcd}$$

Fig. 2: Sample C++ code and the call graph for that code

4.2 Dynamic Analysis

During the dynamic analysis phase, the instrumented MPI program produced by the source-to-source compiler is executed, and the instrumentation code counts the number of times each basic block is executed and the size of each message transmitted. Block iteration counts are used to determine true loop iteration counts and branch ratios of conditional constructs.

The information gathered during dynamic analysis is combined with the per-block operation counts gathered during static analysis to produce equations for total computation and communication requirements of the program. The equations, which conform to the syntax of the Maple Symbolic Computation System, constitute the "Software Description" indicated in Fig. 1.

The "Hardware Description" referred to in Fig. 1 consists of the cost factors C_i in Equation (1), and the α, β and λ terms describing the network characteristics in Equation (3). As mentioned above, we use benchmark programs to obtain values and confidence intervals for these parameters.

4.3 Prediction

The equations for computation and communication requirements are combined with the values constituting the hardware description to produce an equation for total execution time. This equation can be used to predict program execution time for different problem sizes and numbers of processors and examine other aspects of program performance. In addition, the factors in the equation can be changed to explore the performance impact of different types of processors and network media.

5.0 EXPERIMENTAL RESULTS

This section illustrates the use of the APAPS system for predicting the execution time of programs implementing parallel matrix multiplication. Here we will be discussing two parallel applications, one is Matrix Multiplication and the other is the Linear Equation Solver. First we discuss matrix multiplication in which our implementation of parallel matrix multiplication computes C = **A** × **B,** where **A** and **B** are $N \times N$ matrices. To parallelise the computation, we distribute *N/P* contiguous rows of A and *N/P* contiguous columns of B to each processor. Each processor computes the appropriate sub block of **C** and then passes its columns of **B** to the neighboring processor. After *P* iterations, the computation is complete, with the resulting **C** matrix distributed across the processors. Figs. 3(a) and 3(b) show the predicted execution time (in seconds) and percent error of the time predicted by APAPS for the matrix multiplication program respectively. The predictions generated by the APAPS system are quite good with less than 15% prediction error. Fig. 3(a) shows some non-linear transition in the middle; this is because of some heterogeneity in the nature of the cluster of workstations used and the communication overhead between a numbers of processors.

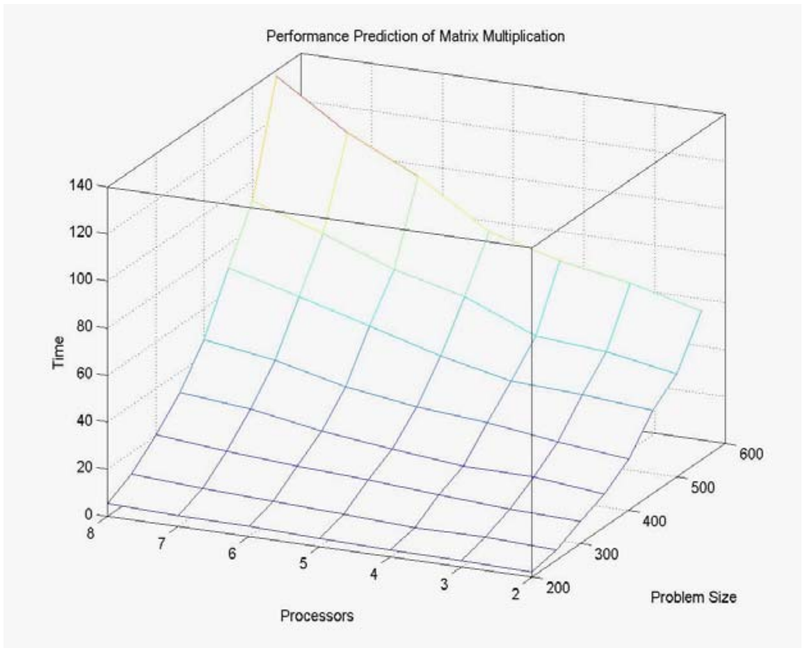


Fig. 3(a): Execution time for the matrix multiplication predicted by APAPS

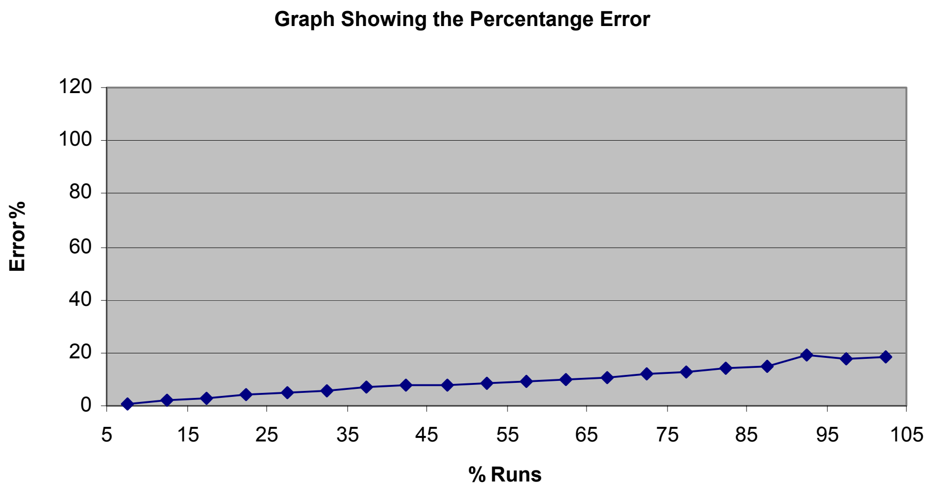


Fig. 3(b): % Error of the actual time for the matrix multiplication program

The second parallel application tested with the APAPS system was a Linear Equation Solver, which is shown below:

$$
\begin{array}{ll}
a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 + \ldots\ldots\ldots\ldots\ldots + a_{n-1,n-1}x_{n-1} & = b_{n-1} \\
\quad --- & \\
\quad --- & \\
a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 & = b2 \\
a_{1,0}x_0 + a_{1,1}x_1 & = b1 \\
a_{0,0}x_0 & = b0
\end{array}
$$

where the a's and b's are constants and the x's are unknowns to be found. The method used to solve for the unknowns x_0, x_2,----,x_{n-1} is simple repeated "back" substitution. First, the unknown x_0 is found from the last equation; i.e.,

$$x_0 = \frac{b_0}{a_{0,0}}$$

The value obtained for x_0 is substituted into the next equation to obtain x_1; i.e;

$$x_1 = \frac{b_1 - a_{0.0}x_0}{a_{1,1}}$$

The values obtained for x_1 and x_0 are substituted into the next equation to obtain $x_{2;}$ and is implemented with the following steps:

The above Linear Equation Solver using pipelining with MPI is used to get the resultant values of X_0 to X_n. In our approach the following steps are being used:

1. Initialise random values to arrays a, b from 0 to n.
2. Calculate X_0 and send the value of X_0 from process P_0 to P_1.
3. Receive X_0 from P_0 and Send this X_0 to P_2 and then Compute X_1 and Send this to P_2.
4. Receive X_0 and X_1 from P_1 and Send this to P_3, then compute X_2 and Send it to P_3.
5. Receive X_0, X_1 and X_2 from P_2 and calculate X_3 to get the resultant X_0 X_1 X_2 X_3.
6. Repeat the steps 1 to 5 for the problem size (set of equations) taken in the graph.

Figs. 3(c) and 3(d) show the predicted execution time (in seconds) and percent error of actual times for the Linear Equation Solver program respectively. The predictions generated by the APAPS system are quite satisfactory, and the prediction error is between 10 to 15%.

5.1 Performance Debugging

The APAPS system can be used to characterise programs for performance debugging purposes. For example, Fig. 4 shows the percentage of time spent in computation and communication for the matrix multiplication program as *N* and *P* vary. This type of graph is useful for determining when communication becomes the limiting factor in program performance.

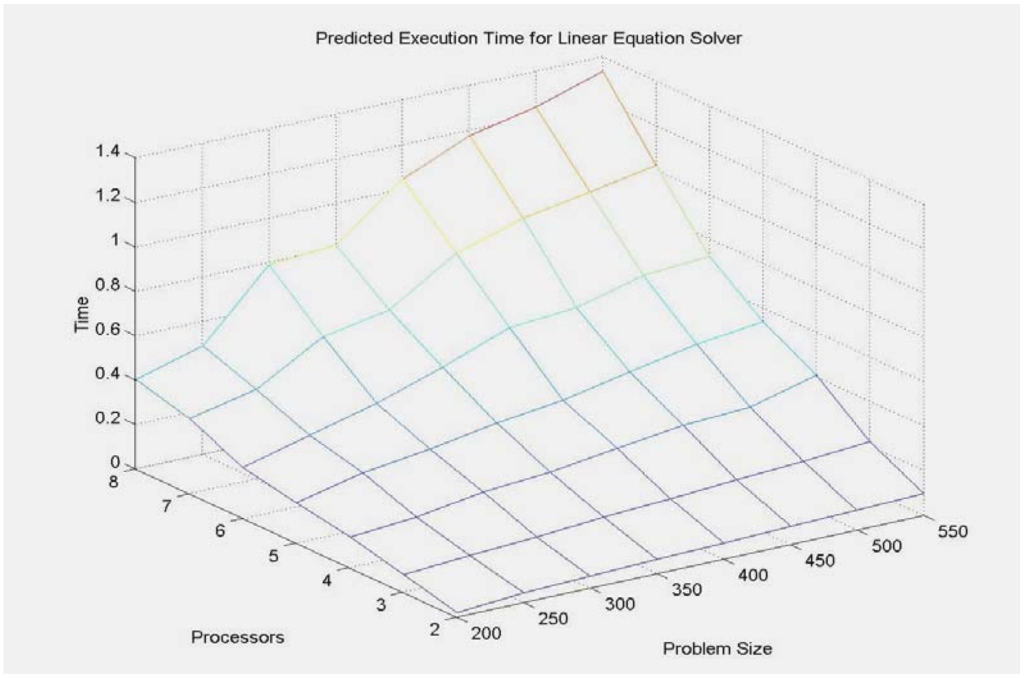


Fig. 3(c): Execution time for Linear Equation Solver predicted by APAPS

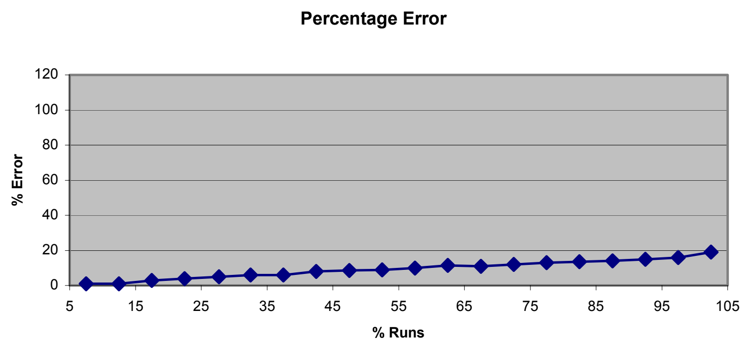


Fig. 3(d): % error of the actual time for Linear Equation Solver

6.0 CONCLUSIONS AND FUTURE WORK

In conclusion some of the experimental results obtained with the APAPS performance prediction system for MPI programs were shown. The results were shown through two parallel applications, Matrix Multiplication and Linear Equation Solver using MPI with the system showing good performance with lower error rate. In the future, further studies can be made to improve the accuracy of the system and refine the performance models as necessary. The APAPS system can be used, as a tool for identifying applications that can benefit from high speed LAN technology such as ATM. Our performance prediction tool is well suited to this problem because it allows the hardware characteristics in the performance model to be changed easily. Finally, the developed system can be used as a basis for performance debugging for MPI programs. The APAPS system allows performance models to be constructed for each basic block of a MPI program. This information can be used to explore the predicted program performance of different parts of an application for varying system configurations.

REFERENCES

[1] M. J. Clement and M. J. Quinn, "Multivariate Statistical Techniques for Parallel Performance Prediction", *Proceedings of the 30th Hawaii International Conference on System Sciences*, HICSS-30, January 2000.

[2] T. Fahringer and H. P. Zima, "A Static Parameter Based Performance Prediction Tool for Parallel Programs", *Proceedings of International Conference on Supercomputing*, pages 207–219. ACM SIGARCH, ACM Press, 1999.

[3] H. Wabnig and G.Haring, "Performance Prediction of Parallel Systems with Scalable Specifications-Methodology and Case Study". *Performance Evaluation Review***,** 22 (2–4), 46–62 (1995).

[4] M. Gupta and P. Banerjee, "Compile-Time Estimation of Communication Costs of Programs", *Second Workshop on Automatic Data Layout and Performance Prediction*, Rice University, Houston, April 1995.

[5] B. P. Miller, J. K. Hollingsworth, and M. D. Callaghan, "The Paradyn Parallel Performance Tools and PVM", *Environments and Tools for Parallel Scientific Computing*. SIAM Press, 1994.

[6] Message Passing Interface Forum (MPIF). *MPI: A Message Passing Interface Standard*. Available from http://www.mpi-forum.org.

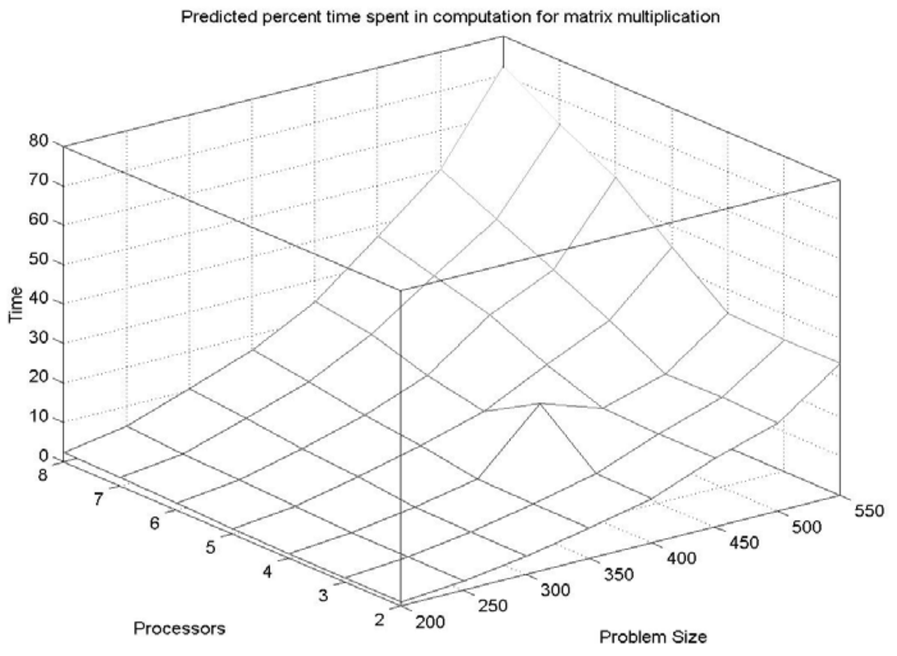
[7] A. L. Beguelin, "Xab: A Tool for Monitoring PVM Programs", *Tech. Rep. CMUCS*, pages 93-164 (June 1993), Carnegie Mellon Case Study. University School of Computer Science.

[8] Sun Microsystems. Sun HPC ClusterTools Software. http://www.sun.com/software/hpc/.

[9] S. K. Damodaran-Kamal and J. M. Francioni, "Mdb: A Semantic Race Detection Tool for PVM", *Proceedings of the 1999 Scalable High Performance Computing Conference*, May 1999.

[10] D. A. Grove and P. D. Coddington, "Precise MPI Performance Measurement Using MPIBench", *Proceedings of HPC Asia*, September 2001.

[11] Kalim Qureshi and Masahiko Hatanaka, "A Practical Approach of Task Scheduling and Load Balancing on Heterogeneous Distributed Raytracing System", *Information Processing Letter (IPL)*, Elsevier Press, Vol. 79, issue 7, 30 June 2001, pp. 65-71.



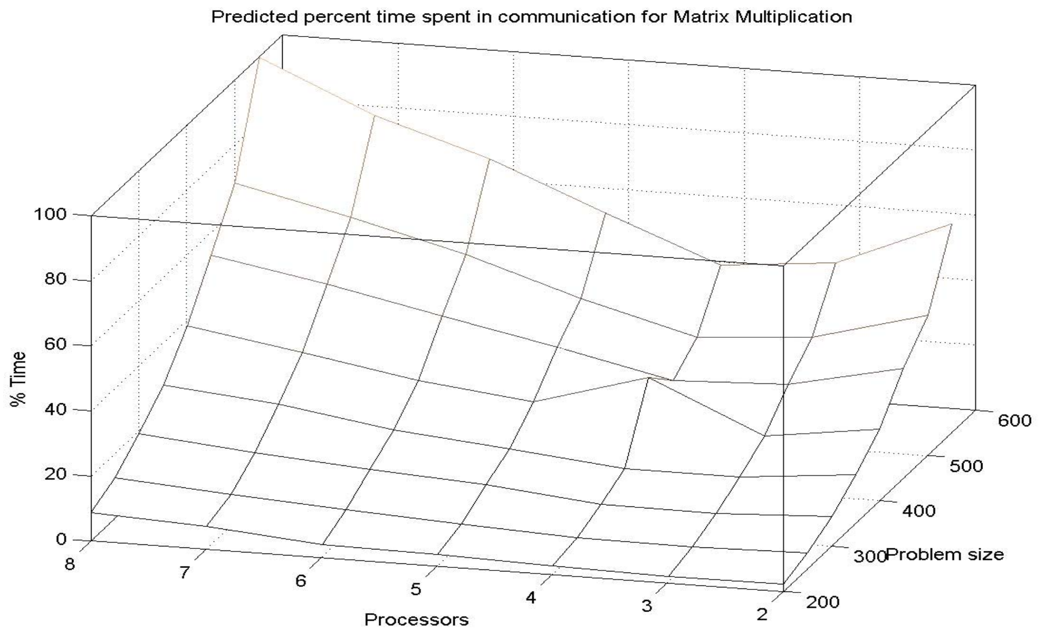


Fig. 4: APAPS predicted percent time spent in communication and computation for the matrix multiplication program

BIOGRAPHY

Rafiqul Zaman Khan is a Lecturer in Information & Computer Science Department at King Fahd University of Petroleum & Minerals (K.F.U.P.M), Dhahran, Saudi Arabia. His research interests include Distributed Image Computing, Thread Programming, Concurrent Algorithms Designing, Task Scheduling, and Performance Measurement He has more then 10 Years University Teaching and Research Experience.

Abdul Quaiyum Ansari is a Professor in the Dept. of Electrical Engineering, Jamia Millia Islamia, New Delhi, presently serving on deputation to Jamia Hamdard (Deemed University) as Professor and Head in the Department of Computer Science. He is also the Dean of Faculty of Management & Information Technology, Jamia Hamdard. He received B.Sc.Engg.(Hons.) degree in Electrical (Low Current) Engg. from AMU, M.Tech. (Integrated Electronics and Circuits) from I I T Delhi and Ph.D. (Hierarchical Fuzzy Systems) from JMI, New Delhi. He has about 25 research papers in refereed journals and conferences. He is reviewer of papers for the International Journal of Fuzzy Systems and the Journal of the Computer Society of India. He is a Chartered Engineer of IETE (India) and IEE (UK), Fellow of the Institution of Engineers (India) as well as of IETE, Sr. Member of Computer Society of India and IEEE (USA) and Member of IEE (UK). He is also a Member of the National Executive Council of ISTE. His research interests include AI, Image Processing, Soft Computing, Intuitionistic Fuzzy Sets, Data Communication and Computer Networks. He is currently working on AICTE sponsored R & D Project entitled “Fuzzy Decision Support System for Metropolitan Traffic Control”.

Kalim Qureshi is an Assistant Professor in Information & Computer Science Department at King Fahd University of Petroleum & Minerals (K.F.U.P.M), Dhahran, Saudi Arabia. His research interests include Network Parallel Distributed Computing, Thread Programming, Concurrent Algorithms Designing, Task Scheduling, and Performance Measurement. He published about 15 Journals/International-Conference papers. He is a senior member of IEEE Computer Society.