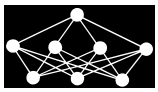

PREDICTING THE PERFORMANCE MEASURES OF A 2-DIMENSIONAL MESSAGE PASSING MULTIPROCESSOR ARCHITECTURE BY USING MACHINE LEARNING METHODS

M.F. Akay, Ç.İ. Aci, F. Abut

Abstract: 2-dimensional Simultaneous Optical Multiprocessor Exchange Bus (2D SOME-Bus) is a reliable, robust implementation of petaflops-performance computer architecture. In this paper, we develop models to predict the performance measures (i.e. average channel utilization, average channel waiting time, average network latency, average processor utilization and average input waiting time) of a message passing architecture interconnected by the 2D SOME-Bus by using Multilayer Feed-forward Artificial Neural Network (MFANN), Support Vector Regression (SVR) and Multiple Linear Regression (MLR). OPNET Modeler is used to simulate the message passing 2D SOME-Bus multiprocessor architecture and to create the training and testing datasets. Using 10-fold cross validation, the performance of the prediction models have been evaluated using several performance metrics. The results show that the SVR model using the radial basis function kernel (SVR-RBF) yields the lowest prediction error among all models.

Key words: *Support vector regression, neural networks, multiprocessors, message passing*

Received: September 22, 2014 **DOI:** 10.14311/NNW.2015.25.013
Revised and accepted: March 15, 2015

1. Introduction

With high computing power of parallel computers, it is now possible to address many applications that were until recently beyond the capability of conventional computing techniques [15]. Message passing environments are considered the most popular programming methods for parallel computers. Their flexibility permits to parallelize all types of applications (client-server, data-parallel, embarked and real-time systems etc.). Message passing is easy to understand, portable, interoperable, and effective. The principle of message passing rests on tasks cooperation through

Mehmet Fatih Akay – Corresponding Author, Çiğdem İnan Aci, Fatih Abut, Department of Computer Engineering, Cukurova University, Adana, Turkey, Tel.: +90-322-3387101, Fax: +90-322-3386326, E-mail: mfakay@cu.edu.tr, caci@cu.edu.tr, abut@hotmail.de.

explicit message exchanges carried out as point-to-point communication between two processes or between several processes and a unique communication task [12].

Two methods are typically used for multiprocessor studies: analytical and simulation modeling. Analytical models become intractable when multiprocessor dynamics are considered, and are not practical for application-dependent studies. Simulation, with the correct assumptions, is a feasible approach that can produce an accurate picture of the dynamic behavior of multiprocessors [6].

Statistical simulation is a method that characterizes the behavior of the program and architecture with some probability distributions. The idea of statistical simulation is to measure a number of important program execution characteristics, generate a synthetic trace, which shows the memory references of a workload, and simulate that synthetic trace. The important benefit is that a synthetic trace is very small compared to real program traces [10]. A statistical simulation is a robust, flexible, and suitable tool in multiprocessor design, but it can still be time consuming especially when multiprocessor systems to be simulated have many parameters and these parameters have to be tested with different probability distributions or values.

There exist some studies in literature [4, 23], which prove the fact that artificial intelligence methods could be applied to predict the performance measures of a multiprocessor architecture. Akay and Abasikeles [4] predicted the performance measures of a multiprocessor architecture employing the distributed shared-memory programming model on the 1-dimensional Simultaneous Optical Multiprocessor Exchange Bus (1D SOME-Bus) architecture. In that study, statistical simulation of the architecture was carried out to generate the dataset. The dataset contained the following input variables: ratio of the mean message channel transfer time to the mean thread run time (T/R), probability that a block can be found in modified state, probability that a data message is due to a write miss, probability that a cache is full and probability of having an upgrade ownership request. Support Vector Regression (SVR) was used to build prediction models for predicting average network latency, average channel waiting time and average processor utilization. It was concluded that SVR model is a promising tool for predicting the performance measures of a distributed shared-memory multiprocessor. In a follow-up work [23], Multi-layer Feed-forward Artificial Neural Network (MFANN) has been used to predict the performance measures of the 1D SOME-Bus architecture employing the message passing programming model. OPNET Modeler [16] was used to statistically simulate the message passing 1D SOME-Bus architecture, the details of which is given in Section 4. The input variables of the prediction model included T/R, node number, thread number and traffic pattern. The output variables of the prediction model included average channel waiting time, average channel utilization, average network latency, average processor utilization and average input waiting time. It was concluded that MFANN-based prediction model performs the best for predicting the performance measures of the message passing 1D SOME-Bus architecture.

Additional studies with different programming models on different multiprocessor architectures are definitely required in order to generalize the effectiveness of machine learning methods for predicting the performance measures of a multiprocessor architecture. There are major differences between the current study

and the study of [23]. First of all, the architecture used in this study is the 2-dimensional Simultaneous Optical Multiprocessor Exchange Bus (2D SOME-Bus), which operates differently and is more complex than the 1D SOME-Bus as outlined in Section 2. Second, the message passing protocol used in this study and [23] is different. In [23], only message passing without acknowledgments (ACK's) protocol has been considered in which the source is not in need to learn whether or not the sent packet has arrived to its destination (i.e. the source node only broadcasts the packet). In this study, we have utilized two different message passing protocols: message passing without ACK's (also known as asynchronous traffic mode) and message passing with ACK's (also known as client-server traffic mode), in which for every packet sent by a source node, there is a returned ACK after the packet has reached the destination node. As a result of this, the dataset used in this study includes one more predictor variable (i.e. spatial distribution of traffic) than the dataset used in [23]. Finally, the range for the number of threads used in [23] can not be used in this study as low values of thread number result in low processor utilization while very high values cause large latencies but not higher processor utilization. Therefore, it is important to select the range of the number of threads depending on the architecture.

In this paper, SVR, MFANN and Multiple Linear Regression (MLR) have been employed to predict the performance measures of the 2-D SOME-Bus multiprocessor architecture using the message passing programming model. OPNET Modeler is used to simulate the message passing 2D SOME-Bus multiprocessor architecture and to create the training and testing datasets. The obtained dataset has five input variables (T/R, node number, thread number, spatial distribution of traffic and traffic mode) and five output variables (average channel utilization, average channel waiting time, average network latency, average processor utilization and average input waiting time). Using 10-fold cross validation, the performance of the prediction models are evaluated by calculating their multiple correlation coefficients (R's), root mean square errors (RMSE's), mean absolute errors (MAE's), root absolute errors (RAE's) and relative root square errors (RRSE's). The results show that the SVR model using the radial basis function kernel (SVR-RBF) has the lowest prediction error. The performance of the linear SVR (SVR-Linear) and MLR models are much lower than the performance of MFANN model, which shows the second best performance among all models.

The rest of this paper is organized as follows: Section 2 summarizes the optical 1D SOME-Bus and 2D SOME-Bus interconnection networks. Section 3 presents an overview of MFANN, SVR and MLR. Section 4 describes simulation framework and dataset generation. Section 5 is devoted to the details of prediction models. Section 6 presents results and discussion. Finally, Section 7 concludes the paper.

2. Overview of the Some-Bus architecture

The 2D SOME-Bus consists of N horizontal and N vertical 1D SOME-Bus [13] networks and N^2 nodes (Fig. 1). Each of N nodes connected to one 1D SOME-Bus has a dedicated broadcast channel and an input channel interface based on an array of N receivers monitoring all N channels and allowing multiple simultaneous broadcasts. At each node, an electro-optical converter consisting of a dual receiver

and transmitter pair allows messages broadcast on one bus to be forwarded and broadcast (in a cut-through manner) on the bus in the other dimension [14].

If a node N_{ij} (connected to vertical bus V_i and horizontal bus H_j sends a message to node N_{mn}, and either $i = m$ or $j = n$, then only one bus (vertical or

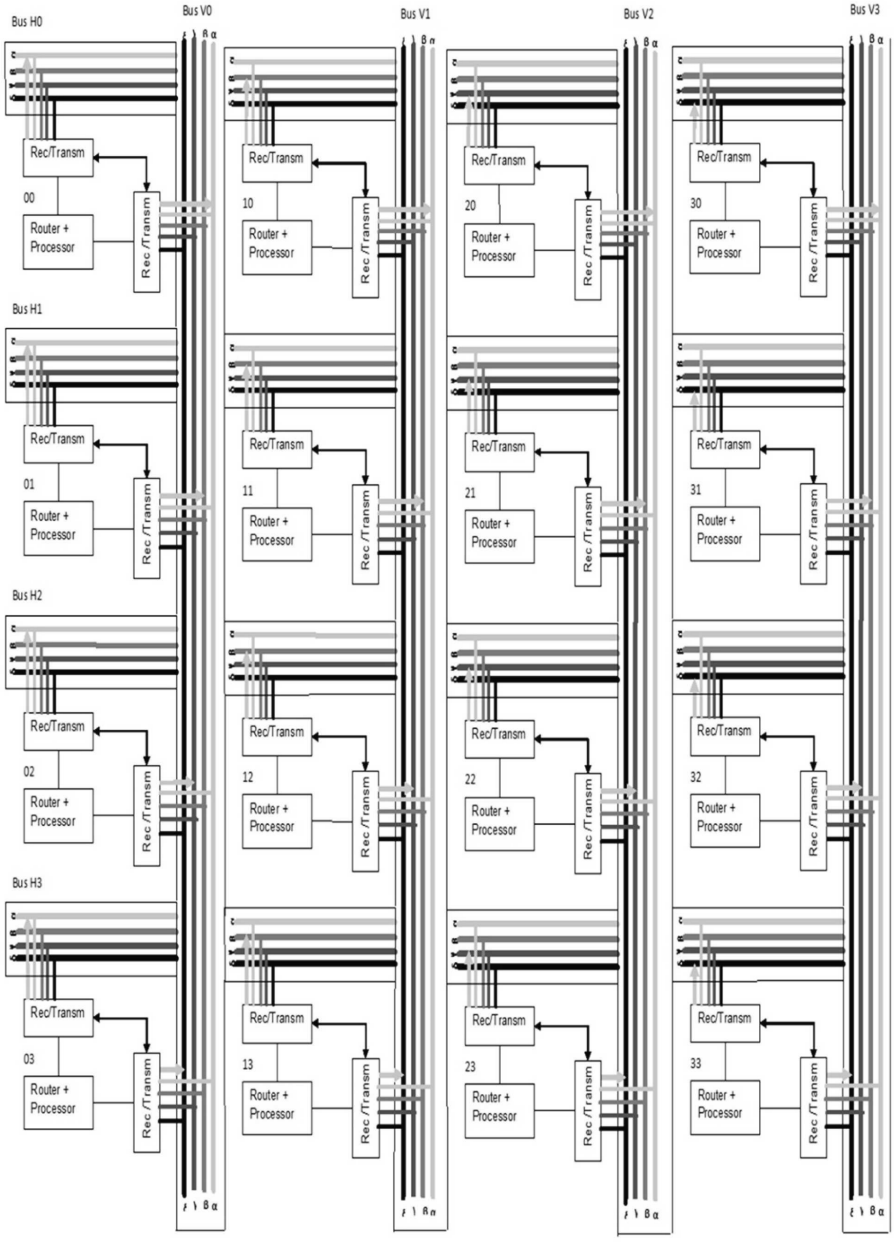

Fig. 1 *2D SOME-Bus architecture.*

horizontal, respectively) is used. The message header includes the identification of the destination node and the message is broadcast on the proper bus. If both $i \neq m$ and $j \neq n$, then the message is broadcast first on horizontal bus H_j to node N_{mj} with the header containing an indication that node N_{mj} broadcast the message on vertical bus V_m so it can be delivered to its final destination, node N_{mn}. By symmetry, the source node may choose to broadcast the message first on vertical bus V_i to intermediate node N_{in} for rebroadcast on horizontal bus H_n [14].

We summarize below the performance characteristics of the 1D SOME-Bus and 2D SOME-Bus in terms of the order of time required to perform a number of basic operations as functions of the number of processors so that one can understand the differences among the complexities of the two architectures:

1. With regard to the task of having an all-to-all communication, where each processor sends a distinct message to every other processor, the 1D SOME-Bus can accomplish this in $O(N)$ time, while the 2D SOME-Bus can do this in $O\left(N^2\right)$ time.

2. With regards to the complexity of performing a barrier synchronization (where all processors synchronize with each other), the 1D SOME-Bus provides a synchronization time of $O(1)$ whereas the 2D SOME-Bus provides a synchronization time of $O(N)$.

3. The diameter of the 1D SOME-Bus architecture is 1, whereas the diameter of the 2D SOME-Bus architecture is N.

4. The number of transmitters and channels required in the 1D SOME-Bus is $O(N)$, whereas the same in the 2D SOME-Bus is $O\left(N^2\right)$.

5. The number of receivers required in the 1D SOME-Bus is $O\left(N^2\right)$, whereas the same in the 2D SOME-Bus is $O\left(N^3\right)$.

6. The 1D SOME-Bus grows by $O(N)$ whereas the 2D SOME-Bus grows by $O\left(N^2\right)$.

3. Overview of prediction methods

3.1 Multi-layer feed-forward neural networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined by the connections among elements. A neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between the elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Fig. 2. Here, the network is adjusted, based on a comparison of the output and the target, up to the network output matches the target. Typically many such input/target output pairs are used to train a network. Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. Incremental training changes the weights and

biases of a network as needed after presentation of each individual input vector. Incremental training is sometimes referred to as "online" or "adaptive" training. Neural networks have been trained to perform complex functions in different fields of application including pattern recognition, identification, classification, speech, and vision and control systems. Today neural networks can be trained to solve problems that are difficult for conventional computer programs or human beings [18].

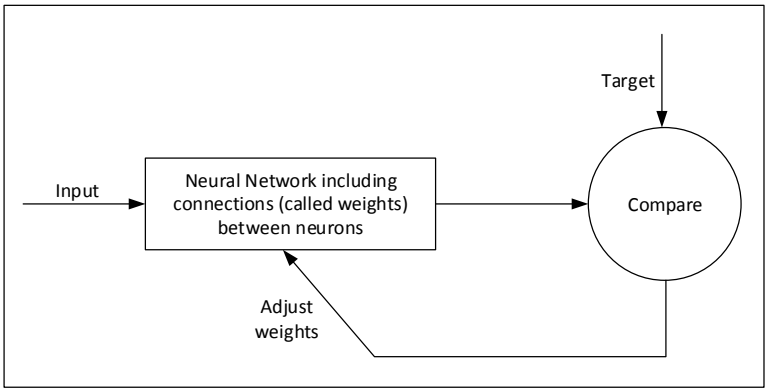

Fig. 2 *Basic principles of artificial neural networks.*

Considering the geometry of the network, there are several types of neural networks: Hopfield, Hamming, Campenter and Grossberg, Kohonen, MFANN and others. Neural networks with different geometry are used for solving different problems. The first three types of neural networks are usually used for binary input data and with problems of classification into classes. The last two types of neural networks are appropriate for the approximation of an unknown function [5].

The geometry of an MFANN is shown in Fig. 3. Input units are connected to the first layer of hidden units which are further connected to the units of the next hidden layer. The units of the last hidden layer are connected to the output units. The input units represent the input data, and the output units represent the output data. The hidden layers may be considered as a black box which performs the necessary transformations of the input data so that the target output data are obtained. The i-th unit in k-th layer is represented by its value y_i^k. Each connection between the units is represented by its weight w_{ij}^k, where index i corresponds to the unit number of the $(k-1)$-st layer and j to the k-th layer. The input layer is denoted by 0, whereas the output layer is denoted by l. The signals travel in one direction only, i.e. from the input layer towards the output layer. The output value of a unit is multiplied by the corresponding weight and added to the value of the signal of the unit of the next layer. In addition, the value of bias neuron or threshold ϑ_i^k is added to the input. The output of a unit in k-th layer can be therefore computed as

$$y_i^k = f\left(\sum_{i=1}^{n_{k-1}} w_{ij}^k y_i^{k-1} + \vartheta_i^k\right). \tag{1}$$

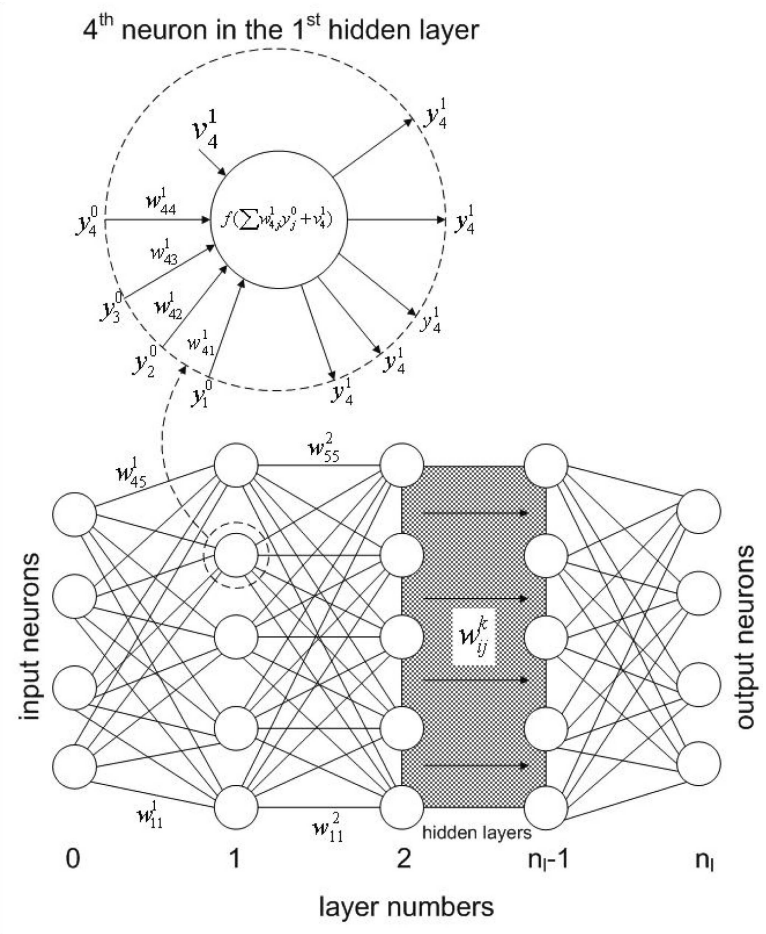

Fig. 3 *A typical Multi-layer Feed-forward artificial neural network.*

Here, f is the so-called *activation function* which enables the modeling of an arbitrary nonlinear relation between input and output variables. Different functions could be used as an activation function. The usual choices of activation function are a sigmoid function, Gaussian and RBF.

The behavior of the neural network depends on the values of weights w_{ij}^k and thresholds ϑ_i^k, which have to be determined by the training procedure. The supervised training is in fact a general optimization problem in which the minimum of back-propagation error $\mathrm{E_p}$ is sought

$$\mathrm{E_p} = \frac{1}{2}\sum_{i=1}^{n_0}\left(t_{pi} - y_{pi}^{nl}\right)^2, \tag{2}$$

where t_{pi} are the target output values, y_{pi}^{nl} are the values of neurons in the output layer, i.e. the output values evaluated by neural network, n_0 is the number of neurons in output layer, i.e. the number of output variables [5].

3.2 Support vector regression

3.2.1 Linear SVR

We are given the training data $(x_i, y_i), (i = 1, \ldots, \ell)$, where $\mathbf{x}$ is a d-dimensional input with $\mathbf{x} \in \Re^d$ and the output is $y \in \Re$. The linear regression model can be written as follows [21]:

$$f(\mathbf{x}) = \langle \boldsymbol{\omega}, \mathbf{x} \rangle + b, \qquad \boldsymbol{\omega}, \mathbf{x} \in \Re^d \; b \in \Re, \tag{3}$$

where $f(\mathbf{x})$ is an unknown target function and $\langle ., . \rangle$ denotes the dot product in $\Re^d$.

In order to measure the empirical risk [7], we should specify a loss function. The most common loss function is the ε-insensitive loss function proposed by Vapnik [21] which is defined as

$$L_\varepsilon(y) = \begin{cases} 0 \text{ for } |f(\mathbf{x}) - y| \leq \varepsilon \\ |f(\mathbf{x}) - y| - \varepsilon \text{ otherwise} \end{cases} . \tag{4}$$

The optimal parameters $\bar{\boldsymbol{\omega}}$ and $\bar{b}$ in (3) are found by solving the primal optimization problem

$$\min \frac{1}{2} \|\boldsymbol{\omega}, \|^2 + C \sum_{i=1}^{\ell} (\xi_i^- + \xi_i^+), \tag{5}$$

with constraints

$$\begin{array}{l} \mathrm{y_i} - \langle \boldsymbol{\omega}, , x_i \rangle - b \leq \varepsilon + \xi_i^+, \\ \langle \boldsymbol{\omega}, , x_i \rangle + b - \mathrm{y_i} \leq \varepsilon + \xi_i^-, \\ \xi_i^+, \xi_i^- \geq 0, i = 1, \ldots, \ell. \end{array} \tag{6}$$

In (6), C is a pre-specified value that determines the tradeoff between the flatness of $f(\mathbf{x})$ and the amount up to which deviations larger than the precision ε are tolerated. The slack variables ξ^- and ξ^+ represent the deviations from the constraints of the ε-tube.

Usually the dual problem is solved. The corresponding dual optimization problem is defined as

$$\max_{\alpha, \alpha^*} -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \langle x_i, x_j \rangle - \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) - \varepsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i), \tag{7}$$

with constraints

$$\begin{array}{l} 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, \ell, \\ \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0. \end{array} \tag{8}$$

Solving the optimization problem defined by (7) and (8) gives the optimal Lagrange multipliers α and α^*, while $\bar{\boldsymbol{\omega}}$, and $\bar{b}$ are given by

$$\begin{array}{l} \bar{\boldsymbol{\omega}} = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) x_i \\ \bar{b} = -\frac{1}{2} \langle \bar{\boldsymbol{\omega}}, (\mathbf{x_r} + \mathbf{x_s}) \rangle \end{array} \tag{9}$$

where $\mathbf{x_r}$ and $\mathbf{x_s}$ are support vectors.

3.2.2 Nonlinear SVR

For nonlinear regression problems, a nonlinear mapping φ of the input space onto a higher dimension feature space can be used, and then linear regression can be performed in this space [19]. The nonlinear model is written as

$$f(\mathbf{x}) = \langle \boldsymbol{\omega},\, \phi(\mathbf{x}) \rangle + b, \boldsymbol{\omega},\, \mathbf{x} \in \Re^{\mathrm{d}}, b \in \Re \tag{10}$$

where

$$\begin{aligned} \boldsymbol{\omega} \;&= \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*)\phi(x_i), \\ \langle \boldsymbol{\omega},\, \phi(\bar{\mathbf{x}}) \rangle &= \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \langle \phi(x_i), \phi(\mathbf{x}) \rangle = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) K(x_i, \mathbf{x}), \\ \bar{b} &= -\tfrac{1}{2} \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*)(K(x_i, \mathbf{x_r}) + K(x_i, \mathbf{x_s})), \end{aligned} \tag{11}$$

where $\mathbf{x_r}$ and $\mathbf{x_s}$ are support vectors. Note that we express dot products through a kernel function Kthat satisfies the so-called Mercer's conditions [21].

If the term $\bar{b}$ is accommodated within the kernel function, Eq. (10) can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) K(x_i, \mathbf{x}). \tag{12}$$

Several kernel functions have appeared in the literature. The RBF has received significant attention, most commonly in the Gaussian form:

$$K(\mathbf{x}, x') = \exp\left(-\frac{\|\mathbf{x} - x'\|^2}{2\rho^2} \right), \tag{13}$$

where ρ is the width of the RBF kernel.

3.3 Multiple linear regression

MLR analysis is a widely used and well documented statistical procedure [20]. MLR attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to the observed data. The dependent variable is modeled as

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \varepsilon, \tag{14}$$

where x_i are the explanatory independent variables, β_i are the regression coefficients, k is the number of independent variables, and ε is the error associated with the regression and assumed to be normally distributed with both expectation value zero and constant variance [3].

The predicted value given by the regression model is then calculated as

$$y' = \beta_0 + \sum_{i=1}^{k} \beta_i x_i. \tag{15}$$

The most common method to estimate the regression parameters β_i is the minimization of the sum of square errors. The equation is as follows [17]:

$$\beta_i = \arg\min + \sum_{i=1}^{n} \left(y_i - y_i^{'} \right)^2 . \tag{16}$$

4. Simulation and dataset generation

OPNET Modeler [16] provides a development environment for the specification, simulation and performance analysis of networks. OPNET provides different tools called editors (for instance Node and Process editors) to develop a representation of a system to be modeled. The node model contains highly programmable modules referred to as processors and queues. The logic flow and behavior of these modules are defined by their process models.

In this study, we use OPNET Modeler as a simulation environment. Communication between nodes is performed by broadcasting messages. The processor at each node extracts a data message from an input queue, processes it for a period of time and when that period expires, it generates an output data message. A processor becomes idle only when all its input queues are empty. The underlying process model that controls queue modules' behavior is OPNET's built-in *acb_fifo* model, which is given in Fig. 4. The "*init*" state is used to initialize the process and setting the appropriate variables. If a packet arrives when the process is in "*init*" state, the process transitions to the "*arrival*" state, else it transitions to the "*idle*" state where it waits for packet arrival. The "*arrival*" state is used for receiving packets and starting service. In the "*arrival*" state, if the server is not busy then the process moves into the "*svc_start*" state, which in turn transitions to the "*idle*" state, where it waits either for packet arrival or service completion. While in the "*idle*" state, if the processing of a packet is completed, the process moves into the "*svc_comp*" state. While in the "*svc_comp*" state, if the queue is not empty, the process moves into the "*svc_start*" state [1].

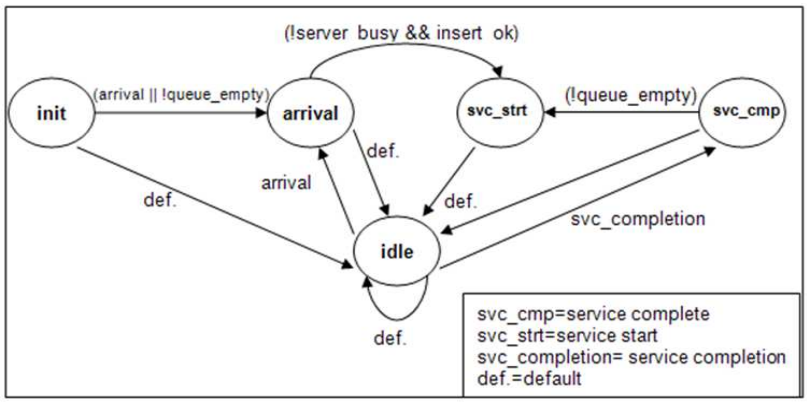

Fig. 4 *A typical process model for the queues.*

Synthetic traffic workloads have been used in OPNET simulation. Regarding spatial distributions, we have used a collection of well-known distributions: bit reverse, perfect shuffle, uniform and hot region. Bit permutations such as bit reverse and perfect shuffle are those in which each bit d_i of the b-bit destination address is a function of the one bit of the source address [8]. Uniform traffic pattern can be represented by a traffic matrix, where each matrix element $\lambda_{s,d}$ gives the fraction of traffic sent from node s destined to node d. In the uniform traffic, the destination node is selected using uniform distribution with the mean in range from 1 to N. In the hot region pattern, the destinations of the 25% of the packets are chosen randomly within a small hot-region consisting of 12.5% of the nodes [2]. Tab. I lists the destination node selection for these traffic patterns.

Name	**Pattern**
Uniform	$\lambda_{s,d} = 1/N$
Bit reverse	$d_i = b_{i+1}$
Perfect shuffle	$d_i = s_{i-1} \bmod b$
Hot region	The 25% of the packets are sent to 12.5% of the node group

Tab. I *Destination node selection for traffic patterns.*

In the 2D SOME-Bus, if the destination of a message is not on the same 1D SOME-Bus as the source, then the message is sent to an intermediate node for retransmission. At that intermediate node the message may be queued or pass through the bypass hardware without any queuing. If bypass is not possible for a certain message, then that message is queued twice, once at some input queue in one dimension and once in the output queue in the other dimension.

In our simulations, we utilized client-server (i.e. a server node sends packets to respond to the reception of packets from clients) and asynchronous traffic modes. The processing time (R) is assumed to be exponentially distributed with a mean of 100 clock cycles. The message transfer time (T) is assumed to be uniformly distributed with mean in range from 5 to 100 clock cycles. The other parameters of the simulation are the number of nodes in the system (selected as 16 and 64), the initial number of threads run by each processor (selected as 4, 6, 8 and 10). The outputs of the simulation are average channel utilization (i.e. percent of time that the channel server is busy), average channel waiting time (i.e. average waiting

Predictor Variables	**Min.**	**Max.**	**Mean**	**Standard Deviation**
T/R	0.10	1	0.55	0.29
Node number	16	64	40	24.02
Thread number	4	10	7	2.24
Spatial distribution of traffic	1	4	2.50	1.12
Spatial distribution of traffic mode	1	2	1.50	0.50

Tab. II *Descriptive statistics of the predictor variables.*

time of a packet in the channel queue until it is transmitted on the link), average network latency (i.e. the time between a request message is enqueued at the output channel and the corresponding data message is received in the input queue), average processor utilization (i.e. percent of time that threads are run by a processor) and average input waiting time (i.e. waiting time of a packet in the input queue until it is serviced by the processor).

The dataset obtained as a result of the statistical simulation includes 640 samples. Tab. II and Tab. III give descriptive statistics of the predictor variables and performance measures, respectively.

Performance Measures	**Min.**	**Max.**	**Mean**	**Standard Deviation**
Average channel utilization	0.14	0.99	0.72	0.20
Average channel waiting time	0.95	1627.33	377.46	381.98
Average network latency	18.98	6065.74	1529.87	1191.29
Average processor utilization	0.20	0.99	0.65	0.20
Average input waiting time	33.04	892.85	333.25	233.72

Tab. III *Descriptive statistics of the performance measures.*

5. Methodology

5.1 Motivation

The performance analysis of a multiprocessor architecture is a very crucial factor in designing message passing multiprocessor systems. Very often, simulation is the only feasible method because of the nature of the problem and because analytical techniques become too difficult to handle. Simulation occurs at many levels, from circuit to system, and at different degrees of detail as the design evolves. Execution-driven and trace-driven multiprocessor simulations have been extensively used to obtain a reliable and accurate prediction of the final design. One of the problems with simulation is that although simulations can be done at a high level of abstraction, they still are extremely time consuming. There are several reasons why this is the case. First, the benchmarks that need to be simulated typically consist of several hundreds of billions of dynamically executed instructions. Second, multiple of these benchmarks need to be simulated to cover a representative set of applications. Third, the complexity of the target system reflects itself in the complexity of the simulator making the simulator at least four orders of magnitude slower than native hardware execution. Fourth, during design space exploration, all benchmarks need to be simulated multiple times to identify the optimal design for a given cost function covering performance, power, area, cost, and reliability.

Due to the reasons given above, one needs alternative methods to predict the performance measures (average channel utilization, average processor utilization, average network latency, average channel waiting time and average input waiting

time) of a multiprocessor system without carrying out simulations for all different values of the parameters such as the node number, thread number, traffic pattern and traffic mode. With no doubt, the alternative methods will use machine learning techniques such as MFANN and SVR which have shown big success in the solution of learning problems on different fields.

5.2 SVR model

The type of kernel function, kernel function parameters, the value of C, and the value of ε for the ε-insensitive loss function are the major components that affect the quality and performance of a SVR model. We chose to use the RBF kernel for developing our SVR model. The parameter that should be optimized for the RBF kernel is the function parameter γ.

It is not known beforehand which C, $\varepsilon$and γ are the best for a given regression problem, therefore some sort of parameter search must be conducted. The purpose is to find optimized values of the triple (C, ε, γ) so that the regression model can predict testing data with minimum error. Many methods for selecting the optimal parameters have been proposed, for example, cross validation, grid search [22], particle swarm optimization [11], genetic algorithm [9], etc. For medium-sized problems, the grid search method is an efficient method for finding the optimum values of C, ε and γ. In grid search, the parameters are varied by fixed step-sizes through a range of values, and the performance of each set of parameters is measured and compared. When performing a search for optimal parameters, different criteria such as maximizing correlation coefficient, minimizing root mean squared error or mean absolute error can be used for determining the optimum function value. To improve the generalization ability, grid search can use a cross validation process. In k-fold cross validation, the original dataset is partitioned into k subsets. Of the k subsets, a single subset is retained as the validation data for testing the model, and the remaining $k - 1$ subsets are used as training data. The cross validation process is then repeated k times (the *folds*), with each of the k subsets used exactly once as the validation data. The k results from the folds then can be averaged (or otherwise combined) to produce a single estimation.

In this study, we used grid search together with 5-fold cross validation to determine the best values of C, ε and γ. To guarantee that the models developed by using SVR are valid and can be generalized for making new predictions regarding new data, the dataset is partitioned into training and independent test sets via a 10-fold cross validation. Fig. 5 shows the flowchart of our SVR-RBF model for a single fold. Initially, the train and test subsets are normalized. This process avoids predictor variables in greater numeric ranges dominate those in smaller numeric ranges and the computational difficulty associated with using larger numeric values. For each value of (C, ε, γ) triple, 5-fold cross validation is conducted on the training subset and root mean squared errors of each prediction are calculated. The (C, ε, γ) triple that yields the lowest overall root mean squared error is chosen for training the train subset. The prediction model is developed after training subset is trained with the optimized values of C, ε and γ. Lastly, the prediction model is used to estimate target values in the test subset.

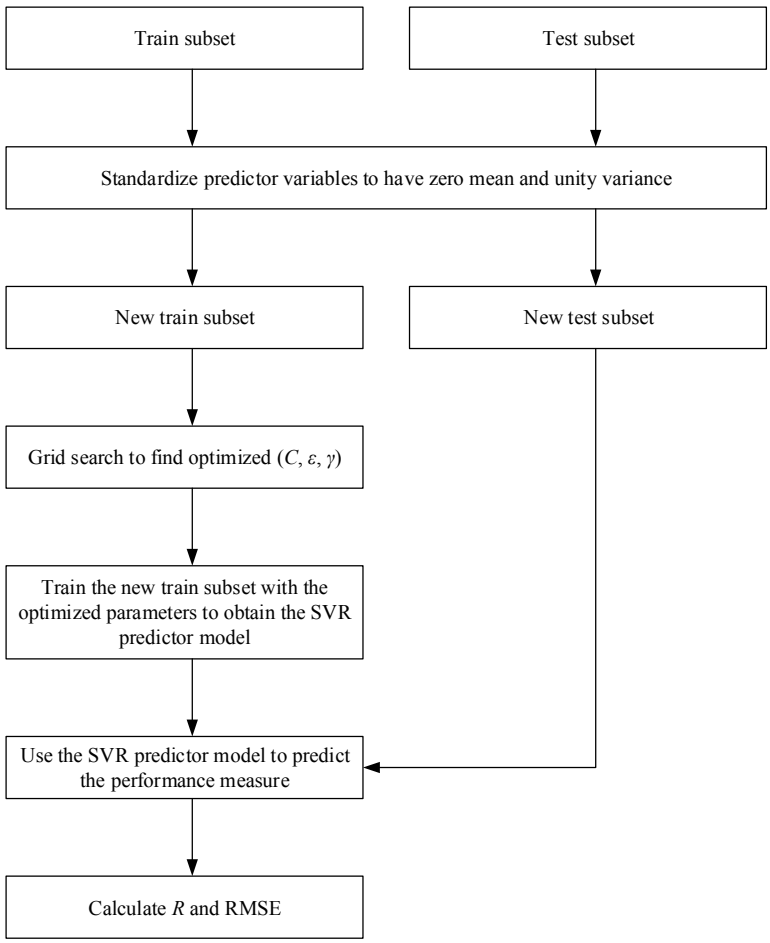

Fig. 5 *SVR model using grid search to optimize model parameters.*

5.3 MFANN model

Back-propagation MFANN is used to develop a model for predicting the performance measures. The performance of an MFANN depends on the number of hidden layers and the number of neurons in each hidden layer. The less neuron used, the less information will be obtained; in contrast, the local minimum may increase, and the network may converge to a local minimum mostly. Thus the network's precision will fall. However, there is not a rule in defining the number of neurons in a hidden layer. In general, the optimal number is empirically chosen based on the physical complexity of the problem. In this study, the number of hidden layers and the number of neurons in the hidden layer are selected by trial-and-error for all MFANN prediction models.

The inputs and outputs are normalized before training the networks. The activation function in the hidden layer is a sigmoid function and a linear function is used in the output layer. The Levenberg-Marquardt algorithm is utilized for

training the networks. The other important parameters of the MFANN models are the number of epochs, the learning rate and momentum. The values of these parameters have been selected as 500, 0.3 and 0.2, respectively, due to fact that using these values yielded the lowest RMSE's for the prediction models.

6. Results and discussion

The performance of all models are evaluated by using 10-fold cross validation and calculating the R, RMSE, MAE, RAE and RRSE, whose formulas are as

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n} (Y - Y')^2}{\sum_{i=1}^{n} \left(Y - \bar{Y}\right)^2}}, \tag{17}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (Y - Y')^2}\ , \tag{18}$$

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} |Y - Y'|, \tag{19}$$

$$\mathrm{RAE} = \sqrt{\frac{\sum_{i=1}^{n} |Y - Y'|}{\sum_{i=1}^{n} \left|Y - \bar{Y}\right|}}\ , \tag{20}$$

$$\mathrm{RRSE} = \sqrt{\frac{\sum_{i=1}^{n} (Y - Y')^2}{\sum_{i=1}^{n} \left(Y - \bar{Y}\right)^2}}. \tag{21}$$

In (17) through (21), Y is the actual value, $Y'$is the predicted value, $\bar{Y}$is the mean of the actual values, $\bar{Y}'$is the mean of the predicted values and n is the number of samples in the test set.

Tab. IV and Tab. V show the values of the utilized parameters of SVR-RBF and SVR-Linear for reproducibility purposes of the presented results. The compared convergence curves of RMSE's in training process for each machine learning method are given in Fig. 6 through Fig. 10. The performance of the prediction models are summarized in Tab. VI through Tab. X, which shows the R, RMSE, MAE, RAE (%), and RRSE (%) for the prediction of the average channel utilization, average channel waiting time, average network latency, average processor utilization and average input waiting time, respectively. Fig. 11 illustrates the percentage decrease rates in RMSE's of the predicted variables for SVR-RBF, SVR-Linear and MFANN methods compared to RMSE's obtained by MLR. Finally, Fig. 12 through Fig. 16 show the scatter plots for each performance measure using the SVR-RBF method.

Performance Measure	SVR-RBF Parameters			
	Epsilon (ε)	**Loss**	**Gamma** (γ)	**Cost** (C)
Channel utilization	0.001	0.1	0.1	1900
Channel waiting time	0.001	0.1	0.3	4000
Network response time	0.010	0.8	0.8	5000
Processor utilization	0.001	0.1	0.3	1200
Input waiting time	0.001	0.1	0.2	1100

Tab. IV *List of the values of the utilized SVR-RBF parameters.*

Performance Measures	SVR-Linear Parameters		
	Epsilon (ε)	**Loss**	**Cost** (C)
Channel utilization	0.001	0.01	700
Channel waiting time	0.001	0.10	1500
Network response time	0.200	0.01	1600
Processor utilization	0.001	0.08	1200
Input waiting time	0.001	0.01	900

Tab. V *List of the values of the utilized SVR-Linear parameters.*

Models	R	MAE	RMSE	RAE (%)	RRSE (%)
SVR-RBF	0.92	0.05	0.07	37.12	37.44
MFANN	0.90	0.06	0.08	40.66	43.87
SVR-Linear	0.70	0.11	0.14	69.95	71.30
MLR	0.69	0.15	0.18	89.80	89.65

Tab. VI *Results for prediction of average channel utilization.*

Models	R	MAE	RMSE	RAE (%)	RRSE (%)
SVR-RBF	0.97	42.73	82.62	13.04	21.63
MFANN	0.96	58.88	96.99	17.97	25.39
SVR-Linear	0.91	116.37	153.71	35.52	40.24
MLR	0.91	127.51	161.14	38.92	42.18

Tab. VII *Results for prediction of average channel waiting time.*

Models	R	MAE	RMSE	RAE (%)	RRSE (%)
SVR-RBF	0.97	184.60	277.96	19.76	23.31
MFANN	0.96	245.09	331.83	26.24	27.83
SVR-Linear	0.92	380.97	520.12	37.56	38.16
MLR	0.87	462.36	629.08	49.50	52.76

Tab. VIII *Results for prediction of average network latency.*

Models	R	MAE	RMSE	RAE (%)	RRSE (%)
SVR-RBF	0.94	0.05	0.06	34.96	35.73
MFANN	0.91	0.06	0.07	35.40	40.01
SVR-Linear	0.76	0.10	0.12	59.47	64.77
MLR	0.74	0.12	0.14	72.83	73.64

Tab. IX *Results for prediction of average processor utilization.*

Models	R	MAE	RMSE	RAE (%)	RRSE (%)
SVR-RBF	0.97	39.48	53.66	19.96	22.95
MFANN	0.96	49.39	63.04	24.97	26.96
SVR-Linear	0.92	72.75	90.34	36.79	38.64
MLR	0.91	84.97	105.90	42.97	45.32

Tab. X *Results for prediction of average input waiting time.*

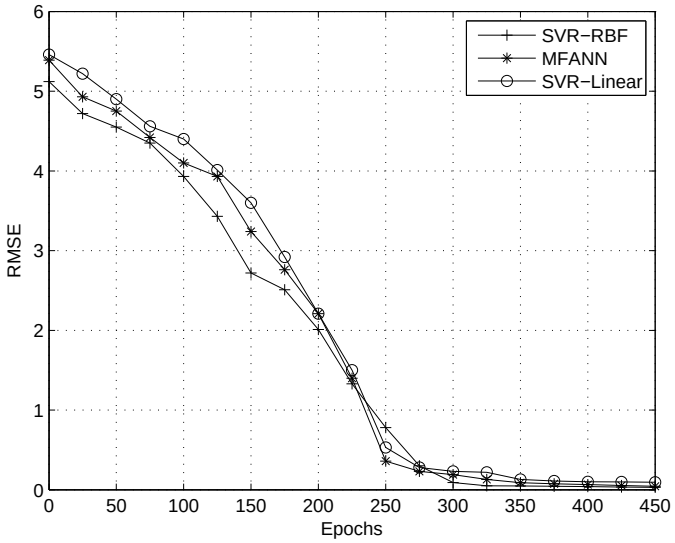

Fig. 6 *Convergence curves of RMSE for channel utilization.*

The following observations could be gained from the results:

- In general, the results show that the SVR-RBF method performs much better than SVR-Linear, MFANN and MLR methods, independent of whether the average channel utilization, average channel waiting time, average network latency, average processor utilization or average input waiting time is predicted. Similarly, MFANN-based prediction models yield more accurate results than SVR-Linear-based and MLR-based prediction models, and finally SVR-Linear-based prediction models lead to better results than the ones obtained by MLR-based prediction models.

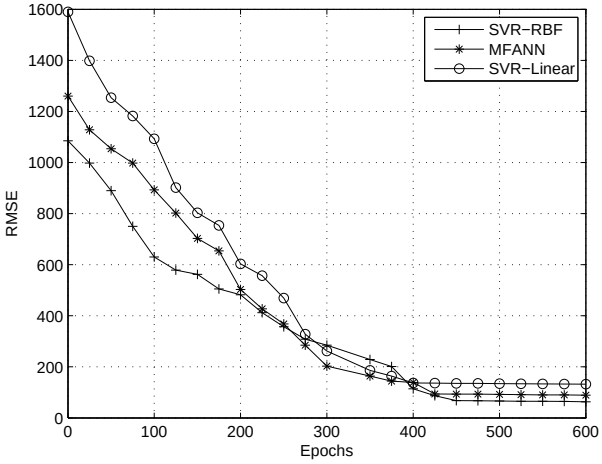

Fig. 7 *Convergence curves of RMSE for channel waiting time.*

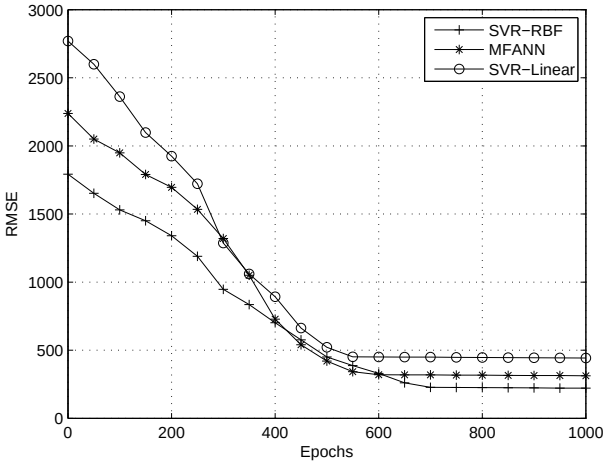

Fig. 8 *Convergence curves of RMSE for network response time.*

- It is not known beforehand whether or not the characteristics of the performance measures are nonlinear. The performance of both the SVR-Linear and MLR models are significantly lower than that of SVR-RBF and MFANN models. This result suggests us that the performance measures have nonlinear characteristics.

- The SVR-RBF model yields the lowest error (RMSE = 0.06) for the prediction of average processor utilization.

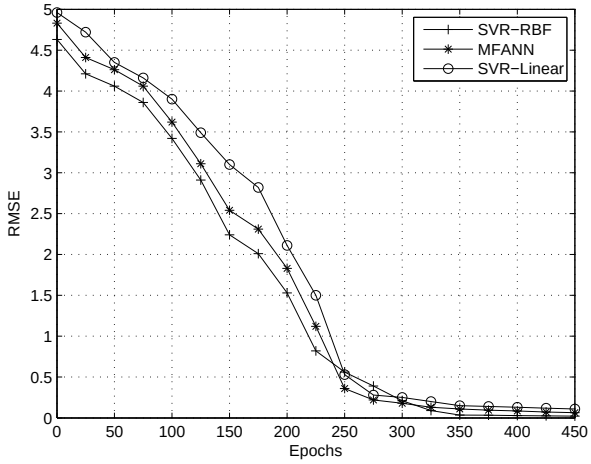

Fig. 9 *Convergence curves of RMSE for processor utilization.*

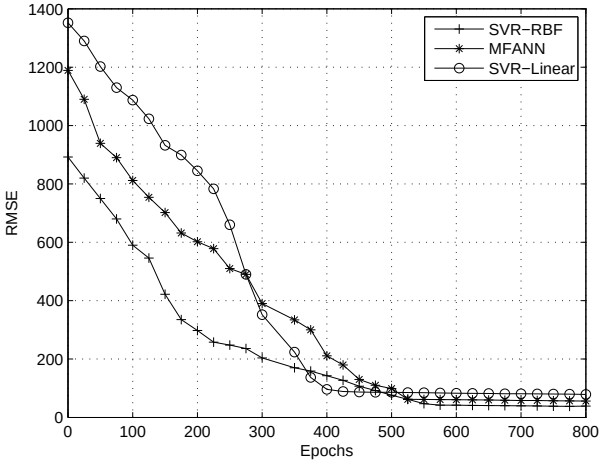

Fig. 10 *Convergence curves of RMSE for input waiting time.*

- The MFANN model yields the lowest error (RMSE = 0.07) for the prediction of average processor utilization.

- The SVR-Linear and MLR models yield the lowest error (RMSE = 0.12 and RMSE = 0.14, respectively) for the prediction of average processor utilization.

- The R of all SVR-RBF models is over 0.92, whereas the R of all MFANN models is over 0.90. The R of all SVR-Linear and MLR models is over 0.69.

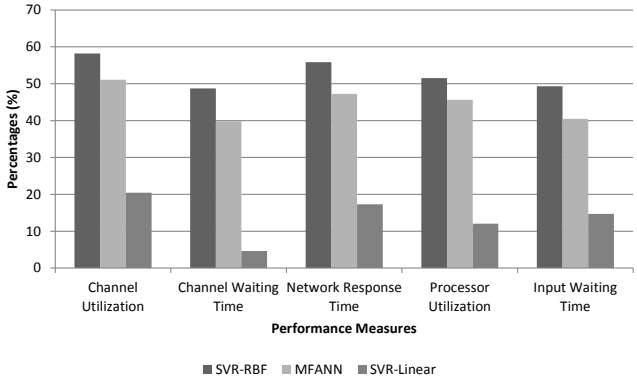

Fig. 11 *Percentage decrease rates in RMSE's of predicted performance measures for SVR-RBF, MFANN and SVR-Linear compared to RMSE's obtained by MLR.*

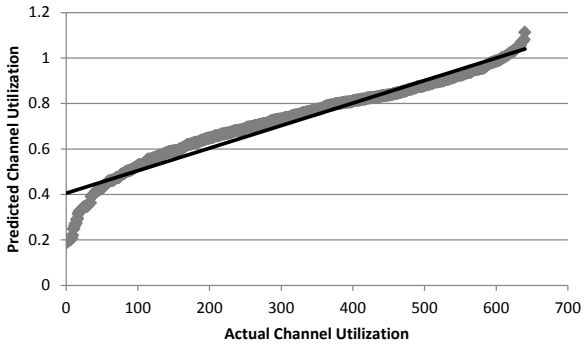

Fig. 12 *Scatter plot of actual channel utilization vs. predicted channel utilization using SVR-RBF.*

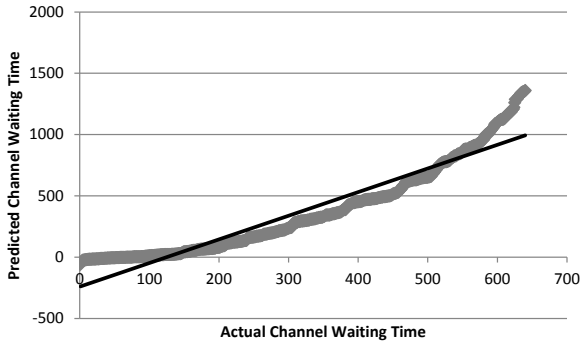

Fig. 13 *Scatter plot of actual channel waiting time vs. predicted channel waiting time using SVR-RBF.*

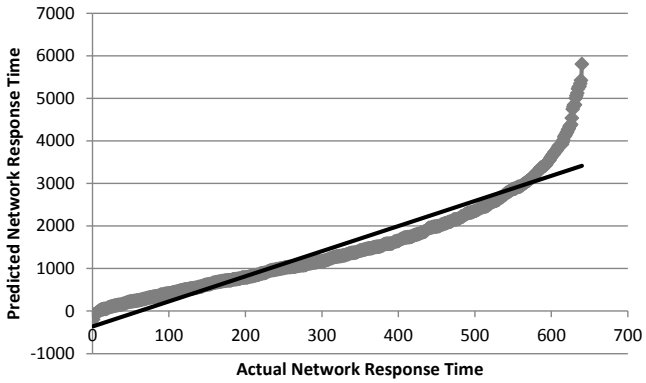

Fig. 14 *Scatter plot of actual network response time vs. predicted network response time using SVR-RBF.*

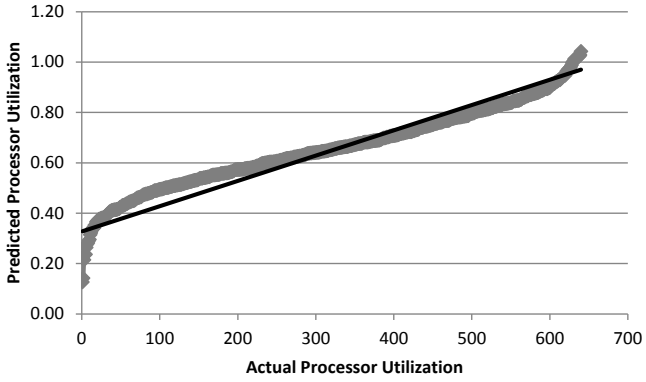

Fig. 15 *Scatter plot of actual processor utilization vs. predicted processor utilization using SVR-RBF.*

- For all performance measures, as compared to the RMSE's obtained by MLR, the maximum percentage decrement rates in RMSE's obtained by SVR-RBF, MFANN and SVR-Linear are 58.20%, 51.07% and 20.42%, respectively.

- The training phase for SVR-RBF model consumes the longest time, as compared to that of the other methods. This is because of the usage of the grid search algorithm in the SVR-RBF model to compute the optimum values of the related parameters.

- The execution times of the SVR-RBF and SVR-Linear prediction models range from 6 to 30 seconds. The execution times of MFANN models are between 1 and 2 seconds. MLR models have negligible execution times (close to zero).

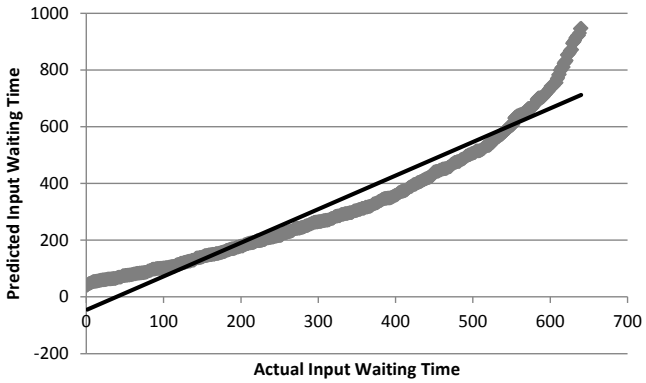

Fig. 16 *Scatter plot of actual input waiting time vs. predicted input waiting time using SVR-RBF.*

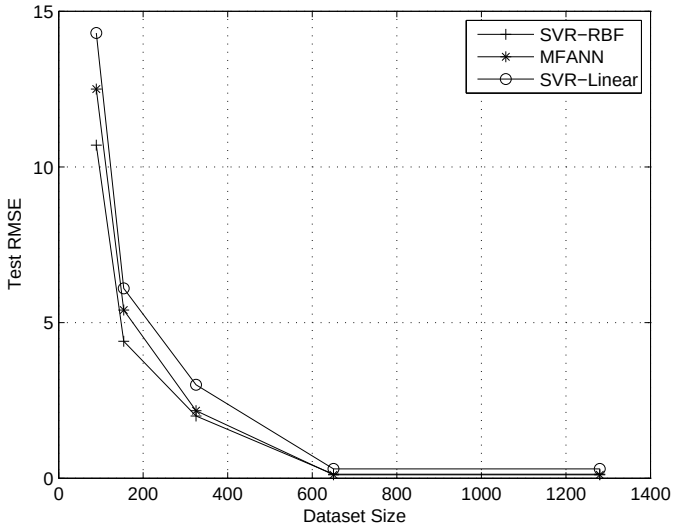

Fig. 17 *RMSE of the test set for varying dataset sizes.*

- Since there is no training phase in MLR, the MLR models produce results much faster than MFANN and SVR prediction models.
- The lowest RMSE's have been observed for prediction of average processor utilization and average channel utilization. This is due to the fact that the range of processor and channel utilizations is between 0 and 1.
- The highest RMSE's have been observed for prediction of average network latency because the gap between the minimum and maximum values of the

average network latency is the highest compared to that of the other performance measures.

- Additional experiments have been conducted with different sample sizes to predict average network latency. Fig. 17 shows the RMSE of the test set for dataset sizes of 160, 320, 640 and 1280 samples using SVR-RBF, MFANN and SVR-Linear as regression methods. It is clearly seen from Fig. 12 that RMSE of the test set decreases considerably as the dataset size is increased from 160 to 640 samples; however we do not observe a significant difference when the dataset size is further increased from 640 to 1280 samples. Therefore, using 640 samples in our dataset is enough in order not to increase the training times of MFANN and SVR.

7. Conclusions

The dynamic, large scale applications which will run on a petaflops-scale computer will require very high data rates and small latencies in order to be successful. Through advances in devices and optical technology, the 2D SOME-Bus interconnection network has become a realistic, highly competitive candidate which achieves the necessary high bandwidth, low latency and large fan-out, and promises to deliver the necessary performance. Obtaining the performance measures of the 2D SOME-Bus multiprocessor architecture for different values of the message passing parameters is an important task and simulation is a time-consuming process for this operation. In this paper, we developed data-driven SVR-RBF, SVR-Linear MFANN and MLR models to predict the performance measures of the message passing 2D SOME-Bus multiprocessor architecture. These models let us estimate the values of the performance measures accurately and fast without running the time-consuming simulation. Among the models, the SVR-RBF model shows the best performance for predicting the performance measures of the message passing 2D SOME-Bus multiprocessor architecture.

Acknowledgement

We would like to thank Dr. Constantine Katsinis for letting us include his material [13] about the SOME-Bus architecture in this paper. We also would like to thank OPNET Technologies, Inc. for letting us use the OPNET Modeler under the University Program.

References

[1] ACI C.I., AKAY M.F. A new congestion control algorithm for improving the performance of a broadcast-based multiprocessor architecture. *Journal of Parallel and Distributed Computing.* 2010, 70(9), pp. 930–940, doi: 10.1016/j.jpdc.2010.06.003.

[2] ADIGA N.R., BLUMRICH M.A., CHEN D., COTEUS P., GARA A., GIAMPAPA M.E., HEIDELBERGER P., SINGH S., STEINMACHER-BUROW B.D., TAKKEN T., TSAO M., VRANAS P. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development.* 2005, 49(2.3), pp. 265–276, doi: 10.1147/rd.492.0265.

[3] AGIRRE-BASURKO E., IBARRA-BERASTEGI G., MADARIAGA I. Regression and multilayer perceptron-based models to forecast hourly O_3 and NO_2 levels in the Bilbao area. *Environmental Modelling & Software*. 2006, 21(4), pp. 430–446, doi: 10.1016/j.envsoft.2004.07.008.

[4] AKAY M.F., ABASIKELEŞ I. Predicting the performance measures of an optical distributed shared memory multiprocessor by using support vector regression. *Expert Systems with Applications*. 2010, 37(9), pp. 6293–6301, doi: 10.1016/j.eswa.2010.02.092.

[5] AMBROZIC T., TURK G. Prediction of subsidence due to underground mining by artificial neural networks. *Computers & Geosciences*. 2003, 29(5), pp. 627–637, doi: 10.1016/S0098-3004(03)00044-X.

[6] BUNTINAS D., MERCIER G., GROPP W. Design and evaluation of Nemesis, a scalable, low-latency, message-passing communication subsystem. In: *Proceedings of Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID '06)*, Singapore. IEEE, 2006, 10 pp. – 530.

[7] CRISTIANINI N., SHAWE-TAYLOR J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[8] DALLY W.J., TOWLES B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2004.

[9] FRIEDRICHS F., IGEL C. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*. 2005, 64, pp. 107–117, doi: 10.1016/j.neucom.2004.11.022.

[10] GENBRUGGE D., EECKHOUT L. Statistical simulation of chip multiprocessors running multi-program workloads. In: *Proceedings of 25th International Conference on Computer Design*. IEEE, 2007, pp. 464–471.

[11] GUO X.C., LIANG Y.C., WU C.G., WANG C.Y. PSO-Based Hyper-Parameters Selection for LS-SVM Classifiers. In: *Neural Information Processing, Proceedings of 13th International Conference (ICONIP 2006), Part II*, Hong Kong, China. Berlin, Heidelberg: Springer, 2006, pp. 1138–1147, doi: 10.1007/11893257_124. Lecture Notes in Computer Science 4233.

[12] HOCHSTEIN L., BASILI V.R., VISHKIN U., GILBERT J. A pilot study to compare programming effort for two parallel programming models. *Journal of Systems and Software*. 2008, 81(11), pp. 1920–1930, doi: 10.1016/j.jss.2007.12.798.

[13] KATSINIS C. Performance analysis of the simultaneous optical multi-processor exchange bus. *Parallel Computing*. 2001, 27(8), pp. 1079–1115, doi: 10.1016/S0167-8191(01)00071-0.

[14] KATSINIS C., NABET B. A Scalable Interconnection Network Architecture for Petaflops Computing. *The Journal of Supercomputing*. 2004, 27(2), pp. 103–128, doi: 10.1023/B:SUPE.0000009318.91562.b0.

[15] KUMAR V., GRAMA A., GUPTA A., KARYPIS G. *Introduction to parallel computing*. 2nd ed. Pearson, 2003.

[16] OPNET Modeler. OPNET Technologies [software]. [accessed 2015-06-19]. Available from: www.opnet.com.

[17] PIRES J.C.M., MARTINS F.G., SOUSA S.I.V., ALVIM-FERRAZ M.C.M., PEREIRA M.C. Selection and validation of parameters in multiple linear and principal component regressions. *Environmental Modelling & Software*. 2008, 23(1), pp. 50–55, doi: 10.1016/j.envsoft.2007.04.012.

[18] SALTAN M., TERZI S. Modeling deflection basin using artificial neural networks with cross-validation technique in backcalculating flexible pavement layer moduli. *Advances in Engineering Software*. 2008, 39(7), pp. 588–592, doi: 10.1016/j.advengsoft.2007.06.002.

[19] SCHOLKOPF B., SMOLA A.J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge: MIT Press, 2001.

[20] STEEL S.J., UYS D.W. Influential data cases when the criterion is used for variable selection in multiple linear regression. *Computational Statistics & Data Analysis*. 2006, 50(7), pp. 1840–1854, doi: 10.1016/j.csda.2005.02.003.

[21] VAPNIK V. The Nature of Statistical Learning Theory. 2nd ed. New York: Springer, 2000.

[22] WENWEN L., XIAOXUE X., FU L., YU Z. Application of Improved Grid Search Algorithm on SVM for Classification of Tumor Gene. *International Journal of Multimedia & Ubiquitous Engineering*. 2014, 9(11), pp. 181–188, doi: 10.14257/ijmue.2014.9.11.18.

[23] ZAYID E.I.M., AKAY M.F. Predicting the performance measures of a message-passing multiprocessor architecture using artificial neural networks. *Neural Computing and Applications*. 2012, 23(7–8), pp. 2481–2491, doi: 10.1007/s00521-012-1267-9.