RESEARCH ARTICLE OPEN ACCESS

A Survey Host Load Prediction In Computational Grid Environment

Shweta Jaiswal
Department of computer science Patel college of science and technology Indore,India

Abstract
Grid computing is a new dimension in computing where computational resources are utilized in an optimum manner.Grid is a computer network organization where multiple CPUs are working together to solve complex and large computational problem.Now in these days interaction on the grid are increased considerably, thus number of requests per second is increased in the time variant manner.Software and hardware level conflicts are arising due to increasing load on host machines. In addition of that to manage these conflicts required to prepare a fault tolerance and management technique to handle these load problems.This paper provides the survey of predictive techniques which can be utilized to design an effective and efficient load forecasting technique to handle faults and manage them.
Keywords-*grid computing, host load, fault, predictive techniques.*

I. INTRODUCTION

Grid computing can mean different things to different individuals.The impressive vision is repeatedly presented as an analogy to power grids where users or electrical appliances get access to electricity through wall sockets with no care or consideration for where or how the electricity is really generated.In similar ways grid computing, computing becomes general and individual users or client applications gain access to computing resources as needed with little or no knowledge of where those resources are located or what the underlying technologies. [1]

One of the basic uses of grid computing is to run an existing application on a different device.The device on which the application is normally run might be unusually busy due to a peak in activity. There are at least two fundamentals for this situation.

1. First, the application must be executable remotely and without undue overhead.
2. Second, the remote machine must have any special type of software, hardware, or resource requirements executed by the application.

If the amount of input and output are huge, more thinking and planning might be required to efficiently use the grid for such kind of task.

The potential for massive parallel CPU capacity is one of the most common visions and features of a grid. In addition to scientific need, such as computing power is driving a new evolution in industries such as the bio-medical field, financial modeling and many others.

The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts.A CPU-concentrated grid application can be thought of as many smaller suburbs, each executing on a different system in the grid. To the extent that subjobs do not need to communicate with each other, the more accessible the application becomes perfectly scalable application.

In this section of the paper contains the overview of grid computing and provide the information about the concept behind the computational grid. In the next section we discuss various grid resources and load parameters.

The design objectives and target applications for a Grid motivate the architecture of the RMS. According to [2] group's design objectives into three themes:

(a) improving application performance,
(b) data access, and
(c) enhanced services.

Using these methods, Grid systems are placed into the categories shown in Figure 1.

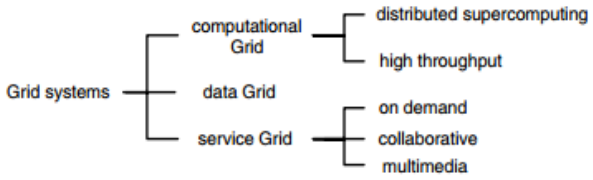


Figure 1. shows the grid design

The computational Grid category represents systems that have higher computational ability for single applications than the capacity of any fundamental machine in the system. Depending on how this capacity is employed, these systems can be further subdivided into distributed supercomputing and high throughput class. A distributed supercomputing Grid executes the application in parallel on multiple machines to reduce the completion time of a job. Typically, applications that require distributed supercomputer are grand challenge problems such as weather modeling and simulation. A high throughput Grid increases the rate of completion

of a stream of jobs and are well suited for 'parameter sweep' type applications.

The service Grid category is for systems that provide services that are not provided by any single device. This category is further subdivided as on-demand, collaborative, and multimedia Grid systems. A cooperative Grid connects users and submissions into cooperative workgroups. These systems enable real time interaction between humans and applications via a virtual workspace. An on-demand Grid category dynamically combinations different resources to provide new services. Most ongoing research activities developing Grid systems fall into one of the above categories. The development of truly general-purpose Grid systems that can support multiple or all of these categories remains a hard problem.

In this section we introduced overview of computational gird enviournment and the process of request and response of the gird and their aspects.

II. BACKGROUND

Infurther we present a brief introduction of the previously made efforts, tools and techniques that are promising to provide the highest accurate predictive accuracy most frequently used models and methods are provided here.

NEURAL NETWORK

The neural network is a supervised learning algorithm where the algorithmis learnt from examples or target values by calculating the weight values. To forecast host load an effort in [3] is observed where the capability to predict the host load of a system is significant for computational grids to make efficient use of shared resources. This paper [3] attempts to improve the accuracy of host load predictions by applying a neural network predictor to reach the goal of best performance and load balance. Author describes the feasibility of the proposed predictor in a dynamic environment, and performan experimental evaluation using collected load traces. The results show that the neural network achieves consistent performance improvement with surprisingly low overhead in most cases. Compared with the best previously proposed method, our typical 20:10:1 network reduces the Mean of prediction errors by approximately up to 79%. The training and testing time is tremendously low, in order to make tens of thousands of accurate predictions within just a second.

The implementation of neural network is defined in two phases' one training and prediction: training process consumes data and designs the data model. Using this data model in the next phase prediction of data is taking place.

Training:

1. Initialize two vectors one input and hidden unit and second output unit.
2. Here first is a two dimensional array W_{ij} is used and the output is a one dimensional array Y_i.
3. Initial weights are random values put inside the victors after that then we calculate the output as

$$x_j = \sum_{i=0} y_i w_{ij}$$

Where y_i is the activity level of the j^{th} unit in the previous layer and W_{ij} is the weight of the connection between the i^{th} and the j^{th} unit.

4. Next, activity level of y_i is calculated by some function of the total weighted input.

$$y_i = \left[\frac{e^x - e^{-x}}{e^x + e^{-x}}\right]$$

When the activity of the all output units has been eveluated, the network computes the error E,

$$E = \frac{1}{2}\sum_i (y_i - d_i)^2$$

Where y_i is the activity level of the j^{th} unit in the top layer and d_i is the desired output of the ji unit.

Calculation of the errorin the Back propagation algorithm is as follows:

- Compute Error Derivative (EA) is the difference between the actual and the desired activity:

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j$$

- Calculate the error changes as the total input received by an output changed

$$EI_j = \frac{\partial E}{\partial X_j} = \frac{\partial E}{\partial y_j} X \frac{dy_j}{dx_j} = EA_j\, y_j\,(1 - y_i)$$

- Calculate the error changes as a weight on the connection into an output unit is changed:

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial X_j} = \frac{\partial X_j}{\partial W_{ij}} = EI_j\, y_i$$

- Calculate the overall effect on the error:

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} X \frac{\partial x_j}{\partial y_i} = \sum_j EI_j\, W_{ij}$$

SVM

According to [4]the problem of empirical data modeling is germane to many engineering applications. In empirical data modeling a process of induction is used to build up a model of the system, from which it is hoped to infer responses of the system that have yet to be observed. Ultimately the quantity and quality of the observations govern the performance of this empirical model. The term SVM is typically used to describe classification with support vector methods and support vector regression is used to describe regression with support vector methods. In this [4] the term SVM will refer to both classification and regression methods, and the terms Support Vector Classification (SVC) and Support Vector Regression (SVR) will be used for specification.

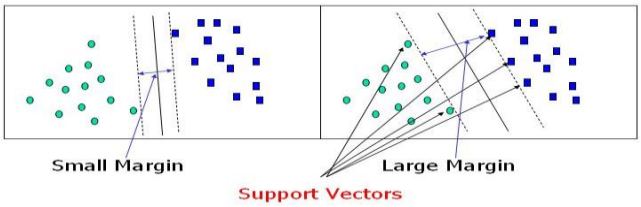
In machine learning, SVM is supervised learning models with associated learning algorithms. SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. SVMs are a group of supervised learning methods that can be applied to classification or

regression. It is primarily a two class classifier. SVMs can efficiently perform non-linear classification using what is called the kernel function; indirectly map their inputs into high-dimensional feature spaces. It can also solve the multiclass problem with the help of kernel methods and kernel function. It aims to maximize the width of the margin between classes, that is, the vacant area between the decision boundary and the nearest training pattern. The basic idea of SVM classifier is to choose the hyper plane that has maximum margin. The dashed lines drawn parallel to the separating line mark the distance between the dividing line and the closest vectors to the crease. The distance between the dashed lines is called the margin. The vectors (points) that constrain the width of the margin are the support vector. Suppose the two forms can be delivered by two hyper planes parallel to the optimal hyper plane.

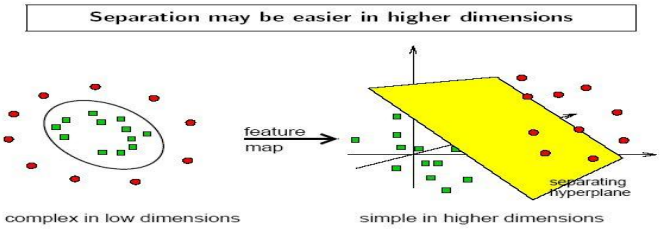
$W_n x_t + b \geq 1$ for $y_t = 1$, t = 1,2,3.....,k

$w_n x_t + b \leq -1$ for $y_t = -1$

where w = {w1, w2, w3,.....,wn} is a vector of n element Figure represents the small margin, large margin and support vectors during classification of a two class data set.



The kernel method consists of two modules: First one is the choice of the kernel and the second one is the algorithm which takes kernel as input. The basic idea of kernel method is to map the data from the input space to feature space F using Ø [11], Ø: X □ F where X = "inputs", F = "feature space", Ø = "feature map". The space of the original data is called input space We say that k(x, y) is a kernel function if there is a feature map Ø such that for all x, y K(x, y) = Ø(x) . Ø (y). In pattern recognition a feature space is an abstract space where each pattern sample is represented as a point in n-dimensional space. Its dimension is resolute by the number of features used to describe the patterns. The concept of a kernel mapping function is powerful. It allows SVM models to perform partings even with very complex boundaries. Figure shows that how to map the data from low dimensional space to a higher dimensional space.



The mapping function needs to be computed because of a tool called a kernel trick. The kernel trick is a mathematical tool which can be applied to any algorithm which only depends upon the dot product between two vectors. Every place a dot product is applied. It is substituted by a core function. When appropriately applied, those candidate linear algorithms are transformed into non-linear algorithms. Those non-linear algorithms correspond to their linear originals operating in the orbit space of a feature space Ø. However, because kernels are used, the Ø function does not need to be ever explicitly computed. Kernel functions must be continuous, symmetric, and most rather should have a positive (semi-definite) fixed Gram matrix. Kernels which are said to satisfy the Mercer's theorem are positive semi-definite, meaning their kernel matrices has no non-negative Eigen values. A positive definite kernel insures that the optimization problem will be convex and a solution will be unique.

Types of kernel functions:

Linear Kernel: The Linear kernel is the simplest kernel function. It is given by the inner product <x, y>plus an optional constant **c**.

$$\boldsymbol{k(x, y) = (x^T y + c)}$$

Polynomial Kernel: The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well appropriate for problems where all the training data is normalized.

$$\boldsymbol{k(x, y) = (\alpha x^T y + c)^d}$$

Adaptable parameters are the slope **alpha**, the constant term **c** and the polynomial degree **d**.

Gaussian kernel: The Gaussian kernel is an example of a radial basis function kernel.

$$\boldsymbol{k(x, y) = e^{\left(\frac{\|x-y\|^2}{2\sigma^2}\right)}}$$

BAYESIAN CLASSIFIER

The Bayes Classifier [5] is a classic supervised learning classifier used in data mining. Bayesian classification consists of five main steps: (1) determine the set of target states (denoted as the vector W=(ω1,ω2,⋯,ωm)T, where m is the number of states), and the evidence vector with h mutually-independent features (denoted as χ=(x1,x2,⋯,xh)T); (2) compute the prior probability distribution for the target states, P(ωi), based on the samples; (3) compute the joint probability distribution p(χ |ωi) for each state ωi; (4) compute the posterior probability based on some evidence, according to Formula (3); (5) make the decision based on a risk function λ(ωi', ωi), where ωi' and ωi indicate the true value and predicted value of the state, respectively.

$$P(w_i|x_i) = \frac{P(x_j|w_i)P(w_i)}{\sum_{k=1}^{m} P(x_j|w_i)P(w_i)}$$

Based on risk functions, there are two ways of making decisions, Native Bayes Classifier and Minimized MSE (MMSE) based Bayes Classifier.

$$\lambda(w_i, w_i') = \begin{cases} 0 & |w_i - w_i'| < \delta \\ 1 & |w_i - w_i'| \geq \delta \end{cases}$$

$$\lambda(w_i, w_i') = \lambda(w_i - w_i')^2$$

According to the different risk functions, the predicted value of the state (w_i') is determined by Formula (6)

and Formula (7) respectively. It is easy to prove that the former leads to the minimal error rate and the latter results in the minimal MSE, where the error rate is defined as the number of wrong decisions over the total number of tries.

$$w_i^{'} = \arg\max P(w_i|x_j)$$

$$w_i^{'} = E(w_i|x_j) = \sum_{i=1}^{m} w_i P(w_i|x_j)$$

Based on the above analysis, the target state vector and the evidence feature vector are the most critical for accurate prediction.

HIDDEN MARKOV MODEL

The Hidden Markov Model (HMM) is a variant of a finite state machine having a set of hidden states, Q, an output alphabet (observations), O, transition probabilities, A, output (emission) probabilities, B, and initial state probabilities, Π. The current state is not observable. Instead, each state produces an output with a certain probability (B). Usually the states, Q, and outputs, O, are understood, so an HMM is said to be a triple, (A, B, Π). [6]

Hidden states Q = { qi }, i = 1, . . . , N .

Transition probabilities $A = \{a_{ij} = P(q_j \ at \ t+1 \ | q_i \ at \ t)\}$, where P (a | b) is the conditional probability of a given b, t = 1. . . T is time, and qi in Q. Informally, A is the probability that the next state is q_j given that the current state is q_i.

Observations (symbols) O = { ok }, k = 1, . . . , M .

Emission probabilities B = { bik = bi(ok) = P(ok | qi) }, where ok in O. Informally, B is the probability that the output is ok given that the current state is qi.

Initial state probabilities Π = {pi = P(qi at t = 1)}.

Let αt(i) be the probability of the partial observation sequence Ot = {o(1), o(2), ... , o(t)} to be produced by all possible state sequences that end at the i-th state.

αt(i) = P(o(1), o(2), ... , o(t) | q(t) = qi).

The Forward Algorithm is a recursive algorithm for calculating αt (I) for the observed sequence of increasing length t. First, the probabilities for the single-symbol sequence are calculated as a product of initial i-th state probability and emission probability of the given symbol o(1) in the i-th state. Then the recursive formula is applied. Assume we have calculated αt (I) for some t. To calculate αt+1(j), we multiply every αt(i) by the corresponding transition probability from the i-th state to the j-th state, sum the products over all states, and then multiply the result by the emission probability of the symbol o(t+1). Iterating the process, we can eventually calculate αT(i), and then summing them over all states, we can obtain the required probability.

$$\alpha_1(i) = p_i b_i(o(i)), i = 1, \dots, N$$

$$\alpha_{t+1}(i) = \left[\sum_{j-1}^{N} \alpha_t(j)\alpha_{ji}\right] b_i(o(t+1))$$

Here i =1, ... , N , t =1, ... , T – 1

$$P(o(1)o(2) \dots o(T)) = \sum_{j-1}^{N} \alpha_T(j)$$

After training of the hidden Markov model it is able to predict the next state of the recommendation system. In addition of that in order to improve more search engine we implement additional parameters to user query with the help of query manipulation and suggestion for writing the nearest query keywords for search. That is implemented using Hidden Markov models.

DECISION TREES

The decision tree is also a supervised algorithm where data is organized in a tree data structure. Where the root node provides the initial condition and intermediate node provides the pathway by which the decisions are predictable. There are various algorithms are now in these days found and most frequently used for classification problems.Most frequently used decision tree is SLIQ, C4.5, ID3 and CART algorithm which are promised to provide accurate classification over large datasets.Here we provide the C 4. 5 algorithm steps that are widely used and accepted for classification and prediction using previous data analysis.

INPUT: Experimental data set D which is shown by discrete value attributes.
OUTPUT: A decision tree T which is created using data set.
Process: 1. Create the node N; 2. If instance belongs to the same class 3. Then return node N as the leaf node and marked with CLASS C; 4. IF attribute List is null, THEN 5. Return the node N as the leaf node and signed with the most common CLASS; 6. Selecting the attribute with highest information gain in the attribute List, and signing the test_attribute; 7. Signing the node N as the test_attribute; 8. FOR the known value of each test_attribute to divide the samples; 9. Generating a new branch which is fit for the test_attribute= a_i from node N; 10. Suppose that C_i is the set of test_attribute=a_i in the samples; 11. IF C_i is null THEN 12. Adding a leaf node and signed with the most common CLASS; 13. Else, add a leaf node return by the Generate_decision_tree.

III. ANALYSIS

In search of predictive methods for CPU load prediction we evaluate various machine learning techniques and algorithms that are promising to provide the highest classification accuracy in different kinds of data given in input. In addition of that we

learnt about data sets where we found that the data is the main ingredient for algorithms to be trained.Here we provide our two key observations to evaluate the CPU load forecast using algorithms.

1. Data to be evaluated:CPU load is directly proportional to the number of jobs are appearing for processing. In addition of that the number of request is also an important factor. During the study of network load and the number of parameters we found that is we predict the number user request and number and length of jobs appeared at a time is predictable thus we can predict the CPU load for that accurately.
Thus we can say there are no parameters are defined to predict the CPU load values accurate. That is much similar to the linear data analysis, thus hidden Markov model is much more adoptable for the proposed work. Which works on number of transections.

2. Performance of the algorithm: In this section of the paper we provide the review of study algorithms. The evaluated algorithm and their description are given below using tables.

Neural Network	The neural network is an efficient classifier accuracy of classification is adaptable, but training time of neural network is increased as the number of instances in a data set is increasing.
SVM	Support vector classifier is another classifier which is working over geometric analysis concept. But the training time of the algorithm is quite high in addition of that the implementation of SVM based classifier is complex enough.
Bayes classifier	That is suitable classifier where data is distributed in binary classification objects. But to implement more than two classes the computational complexity is increasing as data instance are increasing and the number of class labels are increasing.
Hidden Markov	This algorithm works on previous data and its transitions. That is more suitable where data analysis is not dependent with more attributes.
Decision tree	The decision tree algorithm is a transparent model of prediction and classification, but the classification accuracy of the decision trees are not effective enough in addition of that accuracy is depends upon the attributes and their contribution.

IV. CONCLUSION AND FUTURE WORK

In this paper we provide a brief introduction about the study domain as review of predictive techniques that can be used for prediction of CPU load. In addition of that here we study data and their effect on classifier. The classification accuracy is depends upon the data supplied to be analysis and the selected data model algorithm computational nature.According to the nature of CPU load data that is not much dependent on other paramaters, thus a transactional basis model is appropriate for data analysis and next value prediction. in future we design and implement a Hidden markov based predictive technique which promises to provide high accurate classification of the CPU load parameters.

REFERENCES

[1] Introduction to Grid Computing,Bart Jacob, Michael Brown, Kentaro Fukui, Nihar Trivedi,ibm.com/redbooks.
[2] A taxonomy and survey of grid resource management systems for distributed computing,Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran,Softw. Pract. Exper. 2002; 32:135–164 (DOI: 10.1002/spe.432).
[3] Improving Accuracy of Host Load Predictions on Computational Grids by Artificial Neural Networks,Truong Vinh Truong Duy, Yukinori Sato, Yasushi Inoguchi,Received 20 July 2009; final version received 20 February 2010.
[4] Support Vector Machines for Classification and Regression,by Steve R. Gunn, Technical ReportFaculty of Engineering, Science and MathematicsSchool of Electronics and Computer Science, 10 May 1998.
[5] Host Load Predictionin a Google Compute Cloud with a BayesianModel, Sheng Di, Derrick Kondo, Walfredo Cirne,978-1-4673-0806-9/12/$31.00, 2012 IEEE.