CPU USAGE PATTERN DISCOVERY USING SUFFIX TREE FOR COMPUTATIONAL RESOURCE ADVISORY SYSTEM

OOI BOON YAIK

UNIVERSITI SAINS MALAYSIA
2006

CPU USAGE PATTERN DISCOVERY USING SUFFIX TREE FOR COMPUTATIONAL RESOURCE ADVISORY SYSTEM

by

OOI BOON YAIK

Thesis submitted in fulfilment of the requirements for the degree of Master of Science

JULY 2006

ACKNOWLEDGEMENTS

I do like to thank everyone who have helped and supported me to accomplish my master degree at the Grid Lab in Computer Science faculty of Universiti Sains Malaysia.

In particular I want to thank Dr. Chan Huah Yong for giving me a chance to pursue my master degree under his supervision and I am very grateful to have all his support, guidance, advice and encouragement through-out the entire postgraduate study.

I am also very grateful to my second supervisor, Dr. Fazilah Haron, for her guidance, motivation and encouragement throughout the period of conducting this project. Her moral support has kept me to continue the project till the end.

I thank Institute of Postgraduate Study (IPS), USM for allowing me to pursue my postgraduate study and also having selected me under the Graduate Assistant Scheme which provides me with financial security during the period study.

I would like to thank my colleagues especially Cheng Wai Khuen for providing constructive criticism and valuable suggestions on my study.

Last but not least, special thanks to my parent, my two brothers and Chan Jer Jing for their unconditional love and unlimited morale supports to me. Without their support and understanding, I may not be able to accomplish this postgraduate study. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

ACF	Autocorrelation function
ADAPT_AVG	Adaptive Average
AR	Autoregression
ARFIMA	Autoregression Factionally Integrated Moving Average
ARMA	Autoregression and Moving Average
ARIMA	Autoregression Intergrated Moving Average
CPU	Central Processing Unit
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
HMM	Hidden Markov Model
MA	Moving Average
MModel	Multi-Resource Prediction Model
MSE	Mean Square Error
MPE	Mean Percentage Error
NWS	Network Weather Services
PAA	Piecewise Aggregation Approximation
RUN_AVG	Running Average
SAX	Symbolic Aggregation Approximation
SVD	Singular Value Decomposition
SW_AVG	Sliding Window Average
XCF	Cross-correlation Function

PENCARIAN CORAK PENGGUNAAN CPU MENGGUNAKAN PEPOHON SUFIX UNTUK SISTEM PENASIHAT PENGGUNAAN SUMBER PENGKOMPUTERAAN

ABSTRAK

Dalam alam pengkomputeraan grid, sumber pengkomputeraan yang boleh diguna sentiasa berubah dari masa ke masa. Penjadual memerlukan aktiviti ramalan supaya ia dapat berfungsi dengan cekap. Setiap keputusan penjadual sumber pengkomputeraan perlu ditentukan sebelum penghantaran sesuatu kerja. Pemilihan dan gabungan sumber pengkomputeraan yang sesuai adalah penting untuk mencapai kecekapan yang boleh diterima oleh pengguna. Kebanyakkan teknik ramalan sumber pengkomputer yang direka hanya berobjektif untuk mencapai ramalan yang lebih tepat, tetapi mereka tidak dapat memenuhi keperluan kes-kes yang memerlukan tempoh ramalan yang lebih jauh ke masa depan. Contohnya, kerja yang memerlukan tempoh beberapa hari dan tidak boleh bertolak ansur dengan gangguan. Oleh demikian, ramalan penggunaan CPU pada jam atau hari yang berikutnya adalah lebih berguna dari ramalan pada saat seterusnya.

Tesis ini mencadangkan model penasihat penggunaan CPU untuk mencari corak penggunaan CPU melalui sejarah penggunaannya melalui pepohon sufix. Ia dapat memberi laporan corak and ramalan penggunaan sumber pengkomputeraan untuk pengguna supaya lebih memahami sumber tersebut. Mklumat tersebut juga berguna kepada penjadual sumber pengkomputeran untuk meningkatkan keupayaan penjadual serta proses pencarian peluang untuk menggunakan sumber pengkomputeraan. Model ini memproses CPU data sebagai satu sesiri masa dan diproses secara berperingkat seperti ”data reduction”, ”normalisation”, ”discretisation” dan pencarian corak penggunaan. Dalam proses pencarian corak, pepohon sufix telah digunakan.

Penemuan corak penggunaan dalam sumber pengkomputeraan dalam memberi keupayaan kepada kami melakukan ramalan yang lebih jauh ke masa depan, jika dibandingkan dengan teknik ramalan yang sedia ada. Maklumat seperti bila sesuatu sumber yang boleh digunakan dan tempoh yang boleh digunakan dapat diperolehi dengan corak penggunaan yang ditemui. Kami juga memperkenalkan satu strategi ramalan dengan menggunakan anggapan bahawa corak penggunaan yang ditemui akan berulang pada masa depan. Jika perulangan corak dapat ditentukan, maka ramalan yang lebih jauh ke masa depan dapat dilakukan. Dua cara untuk melaksanakan strategi ramalan ini telah diggunakan. Penggunaan corak ”cyclic” dilakukan dengan mencari tempoh selang masa perulangan corak tersebut dan menggunakan tempoh tersebut untuk meramalkan perulangan corak tersebut pada masa hadapan. Manakala ramalan dengan corak ”non-cyclic” dilakukan dengan memadankan perubahan penggunaan CPU dengan corak penggunaan yang ditemui. Jika perubahan penggunaan CPU mirip kepada permulaan sesuatu corak penggunaan, maka bahagian corak seterusnya tersebut akan diramalkan berulang.

Model ini telah dibangunkan dan diuji dengan 20 set data penggunaan CPU dari 10 mesin yang berlainan. Keputusan yang diperolehi adalah positif. Corak penggunan CPU sememangnya wujud dalam CPU data. Terdapat juga corak penggunaan CPU yang ditemui padan dengan sifat penggunanya. Corak penggunaan CPU boleh digunakan untuk melakukan ramalan. Kebanyakan corak yang digunakan untuk ramalan merupakan satu garisan melintang yang bererti tiada perubahan penggunaan. Ini adalah kerana, semasa CPU yang aktif digunakan kurang corak penggunaan dapat sama akan ditemui dalam tempoh tersebut. Teknik ramalan kami berupaya meramalan jauh ke masa depan berbanding dengan teknik seperti ”Network Weather Services” dan ”Tendency-based”. Ramalan dengan corak penggunaan dapat memberi ramalan yang lebih baik semasa penggunaan CPU mengalami perubahan yang ketara.

CPU USAGE PATTERN DISCOVERY USING SUFFIX TREE FOR COMPUTATIONAL RESOURCE ADVISORY SYSTEM

ABSTRACT

In grid computing environment, resource availability often changes from time to time. Thus, schedulers require resource prediction help to make effective scheduling decision. This is because the decision for each job submission must be made prior to submitting a job. The selection of suitable combination of resources is essential to obtain acceptable application performance level. Most of the resource prediction methods were designed with the objective of getting the prediction as accurate as possible. However, there are cases where longer term of prediction is preferred. For example, a job that requires days to complete and has less tolerance to interruption. Therefore, estimation of the average CPU usage for the next few hours or days is going to be more useful than CPU usage prediction at a single future point in time.

This thesis devises a CPU usage advisory model to discover CPU usage patterns from its usage data history using suffix tree. The model produces pattern reports and CPU usage predictions to allow users to understand the CPU usage better, provide extra information to enhance scheduling decision and help to find opportunities for exploiting the available CPU resources. CPU usage data is treated as a time series and processed with a series of steps namely data reduction, normalisation discretisation and pattern discovery. In the pattern discovery process, a suffix tree constructed and used to reveal usage pattern in the CPU data.

The discovery of usage patterns allows us to perform more step-ahead predictions as compared to the conventional CPU usage prediction methods. Information such as the time when a resource is usually available and how long will this resource continue

to be available could be derived from the discovered patterns. We introduce a CPU usage prediction strategy that assumes a frequent pattern will reoccur in the future and implemented this strategy with two approaches. If the reoccurrences of a frequent pattern is known in advance, then a longer term prediction is feasible. The cyclic pattern prediction approach captures the reoccurrence cycle of a pattern and performs prediction by estimating the reoccurrence of the next cycle. The non-cyclic frequent pattern approach performs prediction when the CPU usage pattern shows some similarity to the starting part of a segment of a frequent pattern.

This model is implemented and evaluated with 20 sets of CPU usage data from 10 different machines. The experimental results are promising. Frequent patterns do exist in CPU usage data. We discovered that some patterns could be related to the CPU owner's usage behaviour. CPU frequent patterns can be used to perform prediction. Most of the patterns used for prediction are horizontal line which indicates no usage changes. This is because the usage of an active CPU changes dynamically and rarely exhibit similar patterns. Our pattern prediction approach is capable of providing longer term prediction compared to Network Weather Services and Tendency-based predictors. Pattern prediction approach is able to provide better estimation especially when the CPU usage changes drastically.

CHAPTER 1
INTRODUCTION

1.0 Background

Grid computing has emerged as a new paradigm for high performance computing. The advancement of the Internet and growing availability of inexpensive and powerful networking hardware has facilitated the growth of grid computing. According to I. Foster [5] “A grid consists of a large collection of interconnected heterogeneous and distributed computational resources that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities”. Computational resources in a grid environment could be processors, storage, sensors, software applications and data. In this thesis, the term computational resource refers to processing power. Grid computing can be used to serve many purposes such as exploiting under utilised resources, access to additional resources, parallel computing, data collaboration and application sharing.

One of the initial objectives of grid computing is to enable users to access remote resources and pool computers for large-scale computational process. Therefore, a grid environment is also known as a multi-user time-shared environment. A computational resource is often loaded with unknown workloads introduced by other users. All applications that use the same resource have to share and compete with one another. As a result, it will cause the load and resource availability to vary over time. Besides that, resource owners have the ultimate local control over their resources. Resource owners have unlimited authorization on accessing their resources and may modify or upgrade the type and quantity of resources that are available. The lack of centralized control and demanding end-to-end quality of service has made resource allocation a very challenging problem [32].

In grid environments, job schedulers and resource allocators, regardless software or human must choose the suitable combination of resources to obtain acceptable application performance level. The performance and availability characteristics of each individual resource in a grid environment changes from time to time and the scheduling decision for each job submission must be made before the job is submitted to a remote resource. These have made grid scheduling process requires resource prediction activity to be effective. Therefore, prediction of system performance and resource availability is necessary for efficient use of resources in such dynamic environments [36]. As a result, there are numerous techniques being proposed to perform prediction on grid resource performance and availability.

CPU load data is often viewed as continuous or ordered values; many series modeling techniques have been investigated and applied on CPU load prediction. Time series modeling has been studied widely in other areas such as financial, manufacturing and seismology. In the area of CPU load prediction, the suitability of different linear models on CPU load prediction was investigated by Peter A. Dinda [25]. They showed that simpler AR model is the best model to perform CPU prediction because of its good predictive power and low overhead.

Network Weather Services (NWS) is one of the most well known resource prediction system [31]. It provides one-step-ahead prediction for any time-series fed to its predictor module. It uses a list of statistical models for the prediction of one resource and then chooses the model that provides prediction with the lowest mean square error for the next prediction. The list of statistical models for prediction can be categorized into three categories, namely mean-based, median-based and autoregressive methods.

Other than statistical models, there are also other prediction techniques that are being used to perform CPU load prediction. For example, the last value predictor has been shown to be a very useful prediction technique for CPU load by M. Harchol-Balter *et al.* [18]. It uses the last measured value as the next predicted value. The advantages of this technique implementation are simplicity and low computation overhead.

The Homeostatic and Tendency-based predictors were introduced by Lingyun Yang *et al.* [13] to predict CPU load. Both predictors use various ways to weight recent data for the next value prediction. According to their experiments, between the two families, the tendency-based predictor's performance is better. It predicts the future by assuming the next value will change according to the tendency of the changes of the previous value. The model requires no model fitting; it has low computation and storage overhead.

Besides using a single resource to perform prediction, there are also prediction techniques that use cross correlation between two different types of resources to achieve higher prediction accuracy, shown by Jin Liang *et al.* [11] and M. Swany *et al.* [19].

However, estimation of average CPU load experienced by an application during execution is going to be more useful for a scheduler then CPU information at a single future point in time [14]. For example, a predictor is able to provide a single or multiple-step-ahead prediction, this information is a good estimation for that particular time but it will be a less-effective prediction if an application requires longer execution time. From our study, we found that most of the CPU load prediction methods being used are designed with the objective of getting their prediction as close as possible to the actual value. However, there are cases where by a longer term prediction is preferred [14, 30].

1.1 Overview of CPU Load

CPU load information is represented by a load index which quantifies the measure of a CPU's load. A load index usually takes a zero value if the resource is idle, and increasing the value as the load increases. In our study, we found that there is a wide variety of load indices, such as CPU queue length and CPU utilization. A comparative study of different load indices has been done by Domenico *et al.*. [3] and it is reported that CPU load information based upon the CPU queue length does much better in load balancing compared to CPU utilization. The reason CPU queue length did better is because when a host is heavily loaded, its CPU utilization is likely to be close to 100% and unable to reflect the exact load level of the utilization. In contrast, CPU queue lengths can directly reflect the amount of load on a CPU. As an example, both resources with different average queue length, one with 3 and the other with 6 probably have utilizations close to 100% while they are obviously different. In this study, we use "CPU usage" as a general term to represent the indices. CPU load readings need to be done periodically because the CPU load changes dynamically. However, the length of this period has to be carefully selected, this is because too frequent updates may consume a significant amount of computational power, and infrequent updates may cause the CPU load indices to be insensitive to changes.

In order to better understand the CPU usage's predictability, a study on the statistical properties of CPU load over a large number of machines has been done by Peter A. Dinda [23]. The study showed that CPU load traces show strong correlation over time. Therefore historical-based prediction methods seem feasible but linear models could have difficulty. Self similarity characteristic is also discovered in CPU load and self-similarity is often a symptom of unpredictable and chaotic series. Besides that, CPU load display epochal behaviour and they suggested that the problem of predicting load could be able to be decomposed into smaller problems.

In another study made by Rich Wolski *et al.*. [30], they also discovered the self-similarity characteristic in CPU load, but their results show that self-similarity does not necessarily imply short-term unpredictability. According to Peter A. Dinda [23], the epochal behaviour displayed in CPU load is different from the seasonality in time series analysis. Seasonality in a time series can be identified as regularly spaced peaks or valleys, which occur at reasonably stable with respect to timing, direction and magnitude. Further in the study, they also concluded that it is unreasonable to expect seasonality, because the examination of the power spectrums and autocorrelations of the traces show that CPU load does not exhibit seasonality. However, the explanation given to other works that discover seasonality in CPU load example by Mutka, M. W. [20] is that the changes in CPU load simply insufficient to qualify as seasonality in the strict time series sense. Therefore, the self-similarity and non-seasonality properties that are being discovered in CPU load do not mean that our approach to mine CPU usage patterns from its historical usage data in order to perform prediction is invalid.

1.2 A Brief Overview of Prediction

According to JiaWei Han *et al.*. [10], prediction is a form of data analysis that can be used to determine future data trends and make intelligent decisions. Prediction problems can be divided into two major categories, namely classification and regression. Classification is used to predict discrete values where as regression is used to predict continuous values. Prediction can be viewed as a process of constructing and use of a model to assess the value or value ranges of an attribute in a given sample is likely to have. Data that is being used to construct the model is known as training data set. The constructed model is then being tested for prediction accuracy by using the test data set. Test data could be samples taken randomly from training data set or new data that has not been encountered by the model. Accuracy of different type of the test data set has different implication. Test data taken from the training data set is usually used to evaluate the accuracy of a model fits the data, where as the test data

from previously unseen data is used to evaluate the model prediction accuracy. There are some prediction systems which require domain knowledge while some do not. Prediction systems that require domain knowledge are systems that need to acquire knowledge from human; domain expert and the system uses the expert's methodology and knowledge to perform predictions. Systems that do not required domain knowledge are systems that are capable of learning and generalize knowledge from a given set of data [15].

Prediction has been used in a wide range of application domains. The following are some of the area that uses prediction. Credit approval and risk assessment system has been developed to help in assessing the credit quality of borrowers and assist in making accurate leading decisions. Lenders need to quickly assess the creditworthiness of prospective borrowers to reduce the probability of issuing bad loans while attempting to maintain profitability. Those decisions are actually predictions on whether the borrowers are capable to repay the loan in the future based on the analysis of hundred of variables [43]. In medical domain, association rules analysis has been applied to discover heart disease prediction rules. It uses a real data set containing records of patients who suffer heart disease to create a set of association rules that can be used for predicting an absence or existence of heart disease [1]. Besides that, prediction is also being used in drug design, protein structure prediction [34]. Weather and earth prediction is also one of the most popular domains for prediction. In general, predicting the weather and earth changes is very difficult. The weather and earth prediction is done by constructing a model and predicts the next movement using computer simulations [16, 38]. Stock prediction refers to the forecast of financial markets based on price movements. It uses the assumption that the price movement of a share reflects all information about that share. Therefore, stock predictors model the historical price movement to perform prediction [22].

1.3 Research Motivation, Objectives and Scope

Motivation:

In this thesis, we study various CPU usage prediction techniques to analyze the characteristics of existing solutions and we found that most of the CPU usage prediction techniques only perform estimation of CPU load at a single future point in time. In our opinion, long-term predictions would be more useful in resource management context compare to short-term predictions. This observation has motivated us to devise a model to enable user to understand a CPU usage and perform better resource selection and allocation.

Previous efforts that made by Peter A. Dinda [23] indicated that CPU load is strongly collated over time when studied using the statistical properties of CPU load over large number of systems. This implies that history-based prediction techniques are possible to predict CPU load. Another study made by Mutka, M. W. [20] shows that availability of computing resources change regularly over the hours of the working day and the days of the working week. Therefore, it is feasible to mine for CPU usage patterns from its historical usage data to perform prediction.

Scope:

According to Rich Wolski *et al.*. [33], a grid resource monitoring and predicting system that support resource allocation based on performance should have monitoring, predicting and reporting components. In this thesis, our resource advisory model components are slightly different. Our model has the following four components.

- **Monitoring** - responsible for gathering data from a set of distributed resources. The data includes CPU load, network latency and memory usage.

- **Pattern Discovering** - responsible for discovering frequent patterns from the gathered data. It also includes all the data preprocessing processes such as data reduction and discretization.
- **Reporting** - responsible for reporting the discovered patterns to user.
- **Predicting** - responsible for predicting the future performance using the discovered frequent patterns. It also known as the forecasting engine.

In this study, our concentration is on discovering frequent pattern, generating reports for user to better understand a CPU usage and performing prediction on the CPU usage future state by using frequent discovered patterns.

Objectives:

The objective of this thesis is to find a way to discover frequent CPU usage patterns from its historical usage data and explore the solution of using frequent pattern discovery to find opportunities for exploiting the available CPU resources. Frequent pattern reports and predictions on the CPU usage future state are generated to help user to better understand a CPU usage. We intended to create a model which is capable of providing usage advices to user based on the analysis and discovered frequent CPU usage pattern.

1.4 Contribution

The contribution of this thesis is that we explored an alternative solution to find opportunities for exploiting the available CPU resources. We successfully designed and implemented a model that is capable of discovering frequent CPU usage pattern and perform CPU usage prediction which is capable of providing advice to user based on the analysis and discovered frequent CPU usage pattern. This is the difference between our model and the conventional CPU usage prediction model, which only perform estimation of CPU usage at a single future point in time. From the discovered

frequent patterns, this model could be able to discover when a resource is available and helps user to understand the opportunities for exploiting the idle resource. Besides that, discovered frequent patterns could also help user to monitor a resource better compare to monitor raw CPU usage data. The capabilities of discovering frequent CPU usage patterns and performing longer-term prediction could provide essential information to allow job schedulers and resource allocators to perform a better choice of combination of resources to obtain acceptable application performance level. Figure 1.1 shows the contribution of our work in relation to existing approach.

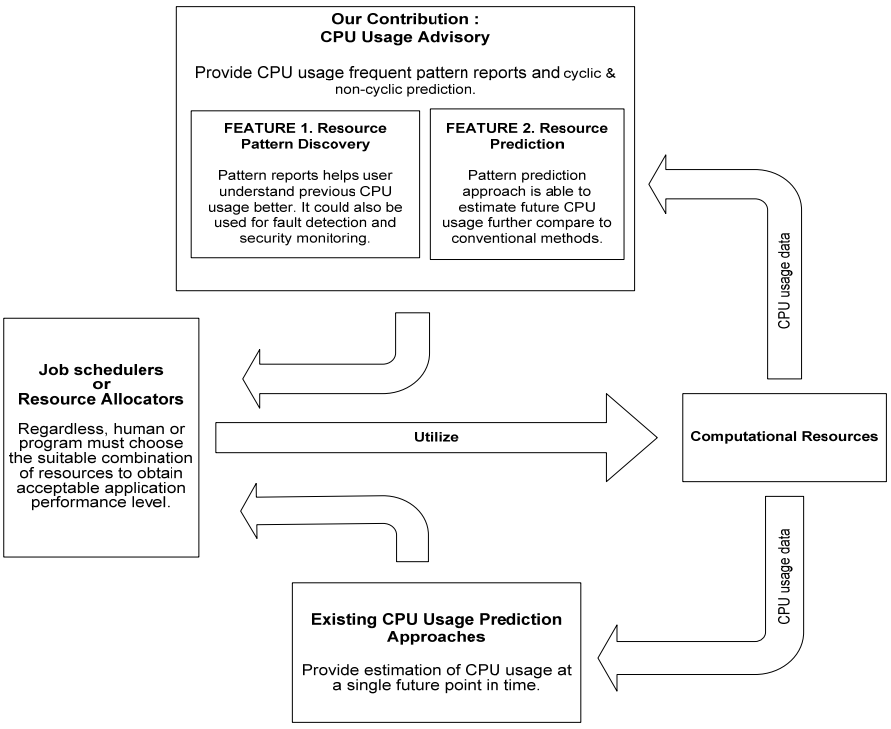

Figure 1.1: The contribution of this thesis in relation to existing approach.

In most CPU usage prediction techniques, accuracy is the main factor to determine the superiority of a prediction technique over the others. However, in our opinion, there are also other useful features needed and would be more useful in the resource management context such as discovering CPU usage pattern to enable user to have an overview of a CPU usage and longer term of prediction. In this model, we introduce two approaches to perform CPU usage prediction with discovered frequent patterns,

cyclic pattern and non-cyclic pattern prediction. This model prediction uses the assumption that a frequent pattern will reoccur. If the reoccurrences of a frequent pattern can be known, CPU usage prediction could be made from those discovered frequent patterns. Figure 1.2 shows the prediction approach proposed in this thesis in relation to existing approaches.

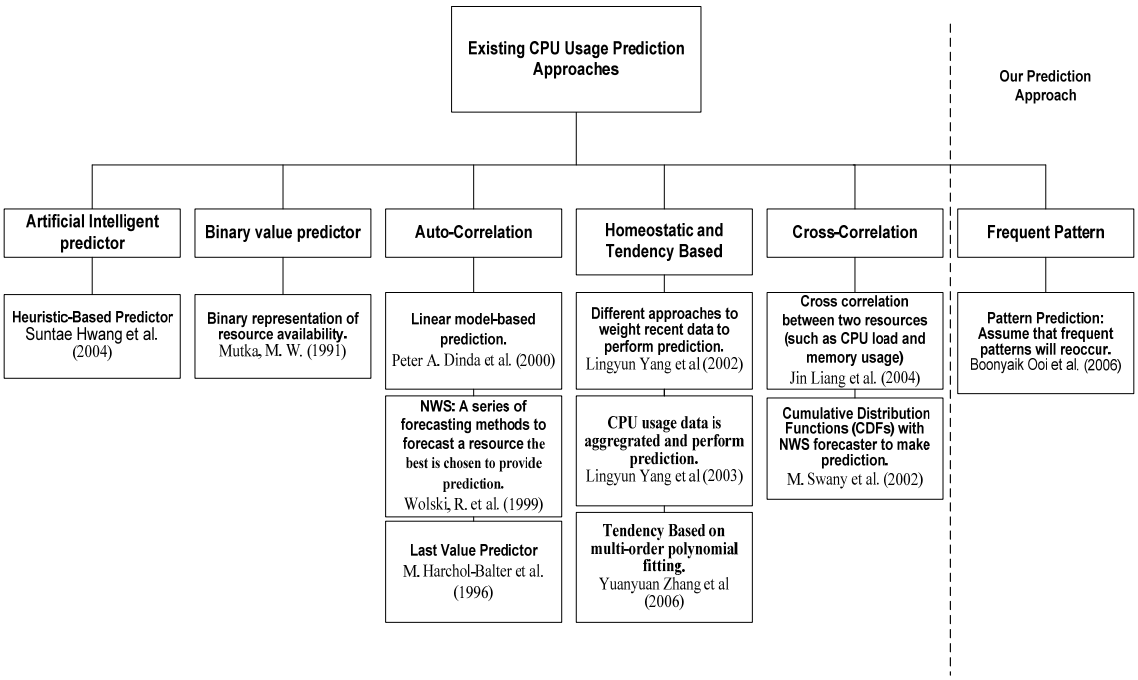

Figure 1.2: The prediction approach of this thesis in relation to existing works.

1.5 The Pattern Discovery and Prediction Model Overview.

We propose a model to discover CPU usage frequent patterns. These patterns could allow us to understand a CPU usage behaviour. Our CPU usage pattern discovery technique was inspired by an earlier works done by Jessica Lin *et al.*. [8]. They used suffix tree as a time series visualization technique. This technique is able to visually summarize both the global and local structures of time series data to enable humans to discover frequently occurring patterns and perform anomaly pattern detection. At a glance, our methodology is similar to the work of Jessica Lin *et al.*. [9] in the sense that it also uses a suffix tree to perform pattern discovery. However, in our

model, there are several extensions made to enable the suffix tree to discover variable length patterns and perform predictions.

Our prediction strategy assumes that a frequent pattern will reoccur. In order to know when a frequent pattern going to reoccur, we introduce two approaches. The first approach is known as the non-cyclic pattern prediction; the main idea is that when the current CPU data has shows some similarity to the beginning segment of a frequent pattern, it will predict that the remaining segment of the pattern may have the possibility of reoccurring. The second approach is known as the cyclic pattern prediction, this model analyze each discovered frequent pattern reoccurring timing to predict a pattern's next occurrence. The basis of the second approach is that constant reoccurring pattern will constantly reoccur again.

Our model design incorporated a series of processes such as data reduction, normalisation, discretized, pattern discovery and prediction engine. From the experimental results, the model shows a promising result. Our model is capable of discovering frequent patterns and performs CPU usage prediction. We evaluate our model by using ten sets of different machines' CPU data; each of the CPU data set consists of a minute set of 28 days long traces taken from two different types of operating system, Windows and Linux. The predictability of this model is then compared with NWS and Tendency Based predictors.

1.6 Outline of the Thesis

This thesis consist of six chapters, chapter 1 covers the background, motivation, objectives and scope of the research.

In chapter 2, we present our study on various prediction techniques and available CPU load prediction techniques. This allows us to understand the characteristics of

different existing solutions. The motivation and overview of these prediction strategies have provided us the direction for constructing our ideas.

Chapter 3 presents the design of our proposed CPU usage frequent pattern discovery and prediction model. The design incorporates a series of processes such as data reduction, normalisation, discretized, pattern discovery and prediction engine.

In chapter 4, we present the implementation detail of our model. The implementation of each module is presented in this chapter.

In chapter 5, we present all the experiments and results that have been used to evaluate our model. Our model is compared with the two well known methods Network Weather Services and Tendency Based predictors.

Finally, in chapter 6 we conclude this thesis. In this chapter, besides the conclusion, we also present some direction for future work pertaining to this research.

CHAPTER 2
LITERATURE REVIEW

2.0 Introduction

In this chapter, we first present various CPU usage prediction techniques and analyze the characteristics of existing solutions. The overview of different prediction strategies will provide us the motivation and direction for our study. Then, we present a survey on the general prediction approaches. A comprehensive survey of techniques will provide us with sufficient knowledge to achieve the objectives that we have stated in the previous chapter. At the end of the chapter, we explain the background and the justification of our chosen approach that is pattern discovery using suffix tree.

2.1 A Survey of CPU Load Prediction Approaches

From our study, there are several works that have been done on resource performance and availability prediction. Although these approaches differ in computational complexity and ability to adapt to dynamic changes, but their aim are similar that is to provide accurate resource performance predictions.

2.1.1 Last Value Predictor

A previous work done by M. Harchol-Balter *et al.*. [18] shows the last value predictor is a good CPU load predictor. From their observation, the probability that a process that uses 1 second CPU time has 1/T probability to use T seconds of total CPU time and process that uses less then 1 second (T < 1) has greater than 0.5 probability to use another T second. Therefore, the last value predictor uses the last measurement only to predict the next value. It can be expressed by the following equation.

$$P_{T+1} = V_T \qquad \text{Equation 2.1}$$

Where T = current time, P = predicted value and V = current value.

The advantages of this approach are low computing power consumption and low storage overhead.

2.1.2 Linear Model Predictor

A detailed study using linear models to perform host load prediction has been done by Peter A. Dinda [24,25]. These linear models are Box-Jerkins models and Autoregression Fractionally Integrated Moving Average (ARFIMA). The Box-Jerkins model consists of Autoregression (AR), Moving Average (MA), Autoregression and Moving Average (ARMA) and Autoregression Integrated Moving Average (ARIMA) models. According to their study:-

- The class of AR models is highly desirable because they can perform fitting in a deterministic amount of time.
- The class of MA models is much more difficult to use because MA models takes a nondeterministic amount of time to perform fitting.
- The class of ARMA models combining the AR and MA models hopes to reduce computation so that it is possible to fit more models in shorter time. However, similar to MA models, it requires a nondeterministic amount of time to perform fitting.
- The class of ARIMA models allows modeling for non stationary sequences; sequences range can vary over an infinite value and have no natural mean. Although CPU load does not vary infinitely, but it does not have a natural mean either.

- The class of AFRIMA models is fractionally integrated ARMA models. This class of models is being used because the CPU load statistical properties exhibit self-similarity.

Their conclusion was that AR models of order 16 or higher are sufficient for predicting 1 Hz data up to 30 seconds in the future with low overhead. The other more complex linear models are expensive to fit and are difficult to use in a dynamic such as resource prediction

2.1.3 Network Weather Service (NWS)

Network Weather Service (NWS) system [30] is one of the most popular resource prediction methods. NWS was developed for the use of scheduler in a networked computational environment. In the design, NWS uses a series of forecasting methods to forecast a resource performance and then chooses the method that yields the lowest mean square error (MSE) or lowest mean percentage error (MPE) to perform prediction for the next value. The following are the MSE and MPE equations.

$$MSE_f(t) = \frac{1}{t}\sum_{i=0}^{t}(err_f(i))^2 \qquad \text{Equation 2.2}$$

$$MPE_f = \frac{1}{t}\sum_{i=0}^{t}(|(err_f(i)|/value(i))x100 \qquad \text{Equation 2.3}$$

where $err_f(t) = value(t) - prediction_f(t-1)$

They believe that a resource may conform to the assumptions of one prediction method for a period of time and change to conform to other prediction method over time due to the dynamic behaviour of the resource. Instead of choosing one predictor in the beginning, they dynamically choose the best of the predictors to perform

predictions. The predictors used in NWS can be categorized into 3 categories, mean-based predictor, median-based predictor and AR model-based predictor. The following is the detail of each category.

Mean-Based Predictor – this class of predictors uses arithmetic averaging over a number of historical data to perform prediction for the next value. In the implementation of NWS, running average, sliding average, last value, adaptive average, and stochastic gradient predictors are in this category. The running average uses the average from previous measurements until the measurement taken at time *t* as the prediction for the next value. Its equation is defined as the following.

$$RUN\_AVG(t) = \frac{1}{t+1}\sum_{i=0}^{t} value(i) \qquad \text{Equation 2.4}$$

Instead of using all the previous measurements to perform prediction, sliding average uses the average taken from a fixed length of previous measurements to predict the next value. Its strategy is that recent values could predict the next value better. The following is its equation.

$$SW\_AVG(t,K) = \frac{1}{K+1}\sum_{i=t-K}^{t} value(i) \qquad \text{Equation 2.5}$$

The number of previous measurements to be taken for prediction from time *t* is specified as *K*. Instead of using a number of previous measurements, the last value predictor only uses one last measurement as the prediction for the next value. It can be represented using the sliding average equation where *K* = 0.

$$LAST(t) = SW\_AVG(t,0) \qquad \text{Equation 2.6}$$

According to Rich Wolski [29], it is difficult to choose the value for *K* because it could be different for each resource and may vary over time. In order to set *K* dynamically, they employed a gradient-decent strategy to adapt to the measurement change and presented another predictor called adaptive average (*ADAPT_AVG)*. Mean Square Error is consistently calculated for each prediction and the *K* value is adjusted frequently to the value that yields the lowest error. In their experiments, the *K* value is restricted to change between 5 and 50 to limit the computational complexity.

Besides that, they also uses stochastic gradient. According to Rich Wolski [28], stochastic gradient predictors are powerful predictive techniques with recursive formulation. It will oscillate randomly about a time series to estimate the mean value of the time series until it converges to a stable estimate. The computation complexity to continuously converge is high as the mean of a time series moves over time. Therefore, they empirically decide appropriate parameters before they successfully identify a dynamic method to perform adaptation.

Median-Based Predictor – this class of predictors uses the median over a number of recent measurements as the prediction for the next value. The median predictors are interesting because they could remove the effect of outlier. In the implementation of NWS, median, sliding median and α-trimmed mean predictors are in this category. The median predictor is defined as the following.

If *K* is odd then $MEDIAN(t,K) = Sort_K((K+1)/2)$ Equation 2.7

Else if *K* is even then $MEDIAN(t,K) = \frac{Sort_K(K/2) + Sort_K(K/2+1)}{2}$

where $Sort_K$ = a sorted sequence of *K* measurement values and $Sort_K(j)$ = the j^{th} value in the sorted sequence.

The *K* represents the number of measurements to be taken for prediction. In order to dynamically adjust the value of *K,* adaptive median (*ADAPT_MED)* uses similar technique that is being employed in *ADAPT_AVG.* According to Rich Wolski [28], median predictions come with a considerable amount of jitter because they lack smoothing power and it is possible to combine with mean-based predictors to have the advantages of both type of predictors. The α-trimmed mean, instead of just using *ADAPT_MED,* averages the central of sliding windows to perform prediction.

Autoregressive Predictor – the objective of this class of predictors is to describe the relationship between variables by finding the equation that represents the relationship. However, fitting these models could require high computational power. Therefore, in NWS implementation, they use purely autoregressive (AR) model which is strictly for linear system and can be solved recursively. The following is a general form of *pth*-order autoregressive model.

$$AR(t, p) = \sum_{i=o}^{p} a_i \times value(t - i) \qquad \text{Equation 2.8}$$

The a_i sequence minimizes the overall error. The sequence can be determined using the following equation if the time series is stationary where $r_{i,j}$ is the autocorrelation function for the series of *N* measurements. The value *p* should be set according to the decay of the autocorrelation function.

$$\sum_{i=0}^{N} a_i \times r_{i,j} = 0 \qquad j=1, 2\ldots N \qquad \text{Equation 2.9}$$

According to Rich Wolski [28], the algorithm that is being implemented in NWS takes *O (p.N)* for *N* measurements to be fitted into an AR model. Therefore, it is unsuitable to

use the entire time series of AR fitting. NWS implemented AR over a sliding window of *K* recent measurements. The choices of parameters depend on the computational complexity that NWS is willing to tolerate. In NWS, the value of parameters *p*=15 and *K*=60 was preset in order to estimate the values using autocorrelation function which could be computationally expensive. Table 1 shows the summary of prediction methods that are being implemented in NWS.

Table 2.1: Prediction methods in NWS

Predictor	Parameters
Running Average (avg.)	
Sliding Window avg.	*K* = 20
Last Measurement	
Adaptive Window avg.	max = 50, min = 5
Median filter	*K* = 20
Adaptive Window median	max = 50, min = 5
α- trimmed mean	α = 0.1
Stochastic gradient	*g* = 0.05
Autoregression	*K* = 60, p = 15

2.1.4 Homeostatic and Tendency-Based Predictor

Homeostatic and tendency Based Predictors were introduced by Lingyun Yang *et al.*. [13]. Both of these predictors are one-step-ahead and low-overhead predictors which use different strategy to predict CPU load. Homeostatic and tendency-based predictors use different approaches to weight recent data to perform predictions. The homeostatic predictors' strategy assumes that CPU load to be self-correcting and will return to the mean of the history value. According to Lingyun Yang *et al.*. [13], the strategy can be described by the following.

If (V_T > *Mean* $_T$) *then*
$P_{T+1} = V_T$ - *DecrementalValue;*
[DecrementalValue is optional depends on the adaptation process]

If (V_T < *Mean* $_T$) *then*
$P_{T+1} = V_T$ + *IncrementalValue;*
[IncrementalValue is optional depends on the adaptation process]

Else
$P_{T+1} = V_T$;

Where $Mean_T = \left(\frac{\sum_{i=0}^{N} V_i}{N}\right)$

The increment and decrement values could be static (fix for all prediction steps) or dynamic (adapted according previous measurements) and the values could be independent value or a relative value which is proportional to the current measurement. As a result, four different homeostatic prediction strategies were investigated. The following is the description of each strategy.

Independent static homeostatic predictor – this predictor generates a prediction by increasing and decreasing the current value with a constant value. According to Lingyun Yang *et al.*. [13], this constant value can be set between 0.05 and 1. This predictor can be expressed by the following expression.

$$DecrementValue = DecrementConstant$$

$$IncrementValue = IncrementConstant$$

Independent dynamic homeostatic predictor - this predictor generates a prediction by increasing and decreasing the current value with a dynamic amount. The increment and decrement constants are adapted to predict the next value and the degree of

adaptation is adjusted between 0 and 1. This predictor's adaptation process can be expressed by the following expression.

$$DecrementConstant(t+1)=DecrementConstant(t)+(ActualChange(t)-DecrementConstant(t))*AdaptDegree$$

$$IncrementConstant(t+1)= IncrementConstant\ (t)+(ActualChange(t)-IncrementConstant(t))*AdaptDegree$$

$$Where\ ActualChange(t) = |\ (value(t) – value(t-1))\ |$$

Relative static homeostatic prediction strategy – this predictor adjust the increment and decrement values proportionally to the current value instead of using constant value to perform a prediction. This strategy assumes that larger load is more likely to change compare to smaller load. According to Lingyun Yang *et al.*. [13], the increment and decrement factors can be set between 0.05 and 1. This predictor can be expressed by the following expression.

$$DecrementValue = V(t)\ x\ DecrementFactor$$

$$IncrementValue = V(t)\ x\ IncrementFactor$$

Relative dynamic homeostatic prediction strategy – this predictor adjust the increment and decrement values similar to the relative static homeostatic prediction strategy but it allows the increment and decrement factor to be adapted dynamically. The adaptation process is similar to independent dynamic homeostatic prediction strategy.

The approach used by tendency based predictors to perform prediction is different from homeostatic predictors. The Tendency-based prediction strategy assumes that

the next value will increase if the current value increases and if the current value decreases, it will predicts that the next value will also decrease. The strategy can be described by the following algorithm.

If (V_T -V_{T-1} < 0) then
Tendency = "Decrease";

Else If (V_{T-1} - V_T < 0) then
Tendency = "Increase";

If (Tendency = "Decrease") then
P_{T+1} = V_T - DecrementalValue;
[DecrementalValue is optional depends on the adaptation process]

Else If (Tendency = "Increase") then
P_{T+1} = V_T + IncrementalValue;
[IncrementalValue is optional depends on the adaptation process]

According to Lingyun Yang *et al.*. [13], the tendency based strategies have extra possible source of error. This is because tendency based predictors are unable to predict when a CPU load going to change direction, from decreasing to increasing or vice-versa. If the variation of the direction changes is large, a large prediction error could occur. In order to minimize this kind of error, the variation of direction change must be minimize. Therefore, they use a basic idea that if a value increases to a very high value or decreases to a very low value, the possibility that a turning point is about to occur is high. The variation of tendency based predictors is similar to homeostatic predictors. From their observation, independent tendency prediction strategy predicts better during the increment and the relative tendency prediction strategy generally performs better during the decrement. This mix variation predictor can be expressed by the following.

*DecrementValue=V(t)*DecrementFactor*

IncrementValue=IncrementConstant

In their experiments, they concluded that tendency prediction strategies outperform other prediction strategies. The mix variation strategy gives better performance on average.

Modification were made to the tendency based method were made by Yuanyuan Zhang *et al.*. [44] to achieve higher accuracy of prediction. Instead of making prediction based on the increases or decreases of one previous measurement, they use several measurements. The next predicted value is produced with multi-order polynomial fitting. In order to reduce the turning point error, it predicts the next possible turning point based on the information of previous similar patterns. For example, after the time series increases successively for several times and one turning point happened and it continuously decrease for a number of times before another turning point happened is considered as a pattern. It was shown that this prediction approach is better then its predecessor.

2.1.5 Predicting Resource Availability Using Binary Values

It is useful to predict the availability of a resource and the duration of the available state. Availability is represented in a series of binary values, 1 indicates available and 0 indicates unavailable at a particular time. A study made by Mutka, M. W. [20], shows that CPU load was low during evening and night, and changed often during the day in a university environment. The basic idea of Mutka, M. W. [20] is that a resource is available for sharing only when the owner leaves the resource. Therefore, by analyzing the usage patterns of resource owners enable them to discover the opportunities for exploiting idle resources. After observing the resources capacity over a period of time, they introduced a method of predicting resource availability called adaptive prediction.

Adaptive prediction strategy predicts the amount of available resources for any particular hour in a day to be the same amount as was available for the same hour on

the previous day. However, it is obvious that there are cases where adaptive prediction cannot handle, such as anomalies in usage patterns. Besides that, they also noticed that the usage patterns on weekdays are significantly different from weekends. In order to solve and take the advantages of such characteristics, they applied two strategies. In order to reduce the amount of prediction error due to anomalies, in the first strategy, the prediction for the next day is made not only according to the day's actual availability but also the day's predicted value. In the second strategy, in order to improve the quality of prediction, the number of prediction periods was increased. This means that besides considering the hour of the day prediction, they also consider the day of the week and distinguish weekdays from weekends. For example, if it is a weekend, besides considering the availability of the day before, it will also consider the availability the weekend before.

2.1.6 Heuristic Predictor

Heuristic predictor was introduced by Suntae Hwang *et al.*. [39]. This predictor is designed to perform prediction on machine with consistent usage pattern such as computers in a teaching lab in a university. This is because the policies of the lab's availability to whom and when can use the machines made the usage pattern consistently reoccur. With this assumption they employed a heuristic rule that usage pattern will repeat over a certain period of time such as weekly basis. In heuristic prediction, they predict this week usage according to the usage patterns in the previous two weeks. For example the CPU usage that had happened on the Monday of the first week and second week is the same; the CPU usage on the Monday of the third week will be predicted to be the same as what happened in the previous Monday.

Besides introducing this heuristic predictor, they also used Hidden Markov Model (HMM) to model the CPU usage. The comparison between the heuristic and HMM predictor was done and the results showed that the heuristic approach is able to out