# Automation of validation environment for ST Microelectronics Set Top Box Audio Firmware

By

**Rahul Goradia**

**10MCEC23**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2012**

# Automation of validation environment for ST Microelectronics Set Top Box Audio Firmware

**Major Project**

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science Engineering

By

**Rahul Goradia**

**10MCEC23**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AHMEDABAD-382481**

**May 2012**

# DECLARATION

I, **Rahul Goradia**, **10MCEC23**, give undertaking that the Major Project entitled **"Automation of validation environment for ST Microelectronics Set Top Box Audio Firmware"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Institute of Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

**Rahul Goradia**

# DEDICATION

I would like to dedicate this thesis to my mother ***TINY***, who taught me lessons of love and kindness. In addition to her I have always been inspired by my father who taught me to fight against all the odds till the end. Whatever good is in me, I have learnt from you.

# Certificate

This is to certify that the Major Project entitled "Automation of validation environment for ST Microelectronics Set Top Box Audio Firmware" submitted by Rahul Goradia (10MCEC23), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mr. Ayaz Siddiqui
External Guide,
CHD-HED Department,
ST Microelectronics,
Greater Noida

Dr. S.N. Pradhan
Guide, Professor,
Department Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

# Abstract

Audio Firmware of Set Top Box needs to go through regression testing and validation process as per the certification criteria of firmware whenever any audio codec of firmware is upgraded. Since the amount of work needed to validate these audio components is quite high, there is a need for a unified validation environment that automates the certification process and provides detailed reporting of failure and certifiable cases. When cluster of workstations is available for validation process there is a need of global scheduler which helps to make effective scheduling decision by predicting resource availability in cluster. By analyzing the historical CPU load data of workstations a prediction of average CPU load of workstation can be estimated. Validation process can be hastened by assigning jobs to available resources . Various linear and non-linear time-series prediction methods are developed for CPU load prediction where CPU usage data is treated as a time-series. This thesis devises a CPU load forecast model to predict CPU load from analyzing historical CPU load data using suffix tree. A probability based one-step ahead prediction table is prepared according to frequently occurring patterns in time-series data. This probability based prediction approach is capable for one-step and multi-step ahead prediction with improving prediction table according to prediction hit-miss. Proposed method has lower time-complexity, better indexing power compared to other linear and non-linear time-series forecasting methods.

# Acknowledgements

I am deeply indebted to **Mr. Ayaz Siddiqui** for his constant guidance and motivation. He has devoted significant amount of his valuable time to plan and discuss the thesis work. Without his experience and insights, it would have been very difficult to do quality work.

I would also like to extend my gratitude to **Dr. S N Pradhan** for for his consistent and invaluable, inspirations, prolific and introspective guidance with constructive suggestions, deliberative discussions and active persuasion encouragement throughout the course of my study.

I am very thankful to **Sanjay Sir, Dixit Sir, Ripinder Sir, Shalabh Sir, Ashwani Sir and Anuj Sir** from Audio Firmware Team at ST Microelectronics, Greater Noida for their invaluable guidance and support at ST Microelectronics.

I further acknowledge the untiring assistance rendered by my friends **Sanket Parmar, Parth Desai and Bhargav Patel** for being a helping hand whenever I am in need and for their constant inspiration. I am obliged to **Amish Parikh, Sunny Sindhy, Vatsal Shah and Harshit Patel** who were always being supportive and encouraging me during my research work. You are the best friends one can have in his life.

Words felt short to say thanks to **my parents, my uncle and aunty, my brother and my cousins**. You put your trust in me and helped me in each way so that I can achieve my goals.

Last but certainly not the least, I prostrate to **Almighty God** who made everything possible, who made everything easy and always showered his blessings on me.

**- Rahul Goradia**

# Contents

# List of Tables

# List of Figures

# Abbreviations

AR ..... Auto Regression
ARFMA ..... Auto Regression Fractionally Integrated Moving Average
ARIMA ..... Auto Regression Integrated Moving Average
ARMA ..... Auto Regression Moving Average
CPU ..... Central Processing Unit
DTV ..... Digital Tele-Vision
MA ..... Moving Average
MPE ..... Mean Percentage Error
MSE ..... Mean Squared Error
NWS ..... Network Weather Service
PAA ..... Piecewise Aggregate Approximation
SAX ..... Symbolic Aggregate aproXimation
STB ..... Set Top Box
STSTB ..... ST Microelectronics Set Top Box

# Chapter 1

# Introduction

Firmware today is facing greater complexity and shorter schedules. As complexity of firmware increases testing of the firmware is becoming more complex and time consuming, which is the deciding factor of the total cost of the project. Due to low time to market of a product and technology development firmware needs to be updated frequently and validated after each update. In a grid environment validation can be hastened by developing a global scheduler which takes care of free computational resources and uses them effectively.

## 1.1 Introduction

In developing a firmware system, validation and verification are recognized as vital activities. Validation plays an important role in any firmware design as it starts at the modular level and works outward towards the integration of the complete system. Validation time and effort put in validation directly decides the cost of the project. As the new technologies evolve updates in the software is mandatory to implement and thus frequent changes in the software are made to implement these updates. These updates are validated by the process of exhaustive testing of the all test-cases on the firmware, which is called "Regression testing". Effective reporting of validation is also equally important because common mistakes affecting all features of software are easily caught when continuous reporting of validation is communicated to all the developers and validation team members. Nowadays industries prefer to automate the test and validation process because

- Manually testing of all the test cases frequently is a very tedious and time consuming job.
- Manual testing is also prone to provide more errors.
- Manual testing can only be performed while office hours only.
- In distributed environment where workstations are connected through LAN or other high speed networks validation can also be hastened by maximizing the usage of available resources if it is automated.
- Automation is essential to ensure constant communication between testing and developing efforts.
- Provides quick visibility into code quality.

The distributed computing environment to which all team members have access consist of a collection of loosely interconnected hosts running different operating systems. As users run their jobs, the computational load on the individual hosts changes over time. In most of the cases computational power of the cluster is not used efficiently as global scheduler is not available which controls job management on different hosts.[1] If the computation power of whole cluster is utilized effectively the overall load to single user can be minimized and testing and validation process can be hastened providing better quality in software development and low time to market.

Global scheduling can be implemented by analyzing statistical property of host load and finding the best host to handle the job. Scheduling of job becomes even more easier when average time taken by a job is already known. Analyzing statistical properties of host load in cluster, method can be formed to predict average load of each host in the cluster for future time. Most of prediction techniques use historical data to understand trends present in the historical data and predict the future values. Most widely used prediction method is to transform the data into time series. Various methods have been developed to represent continuous as well as discrete time series data in time-domain, frequency domain or modern data-structures like tree. Each method have its own advantages over other. Similarly various linear methods, non linear methods and probability base forecasting algorithms have been

developed to predict the time-series data.
Peter and David[1, 13] suggested a linear model to predict 1 to 30 second load average on host machine using AR, MA and ARIMA processes. Last-value predictor method shown by Harchol-Balter and Downey[12] has low computation and storage overhead compared to ARIMA processes. L. Yang et al. [6] presented a tendency-based one-step-ahead time series prediction strategy to follow current trend in time-series. Above described all methods are static time-series analysis methods and do not predict on basis of whole historical time-series data. Also they have higher time complexity due to iterative nature. Eamonn Keogh and et al. [16] showed a new suffix tree based method which provides whole historical coverage but fails to follow current trend in time-series while prediction.

## 1.2 Motivation

While validation of the firmware if whole validation is carried out by a single workstation then the workload of the workstation increases and which increases the time consumed by the overall validation process. Instead of that if automated validation environment will schedule the workload in available cluster of workstations and will be able to perform validation task even in non-office hours. This hastens the validation process, constant and good reporting can be achieved by the automated validation which increases quality of designed firmware and cuts the coast of the project.

## 1.3 Objective

Objective of this thesis is to develop a new probability based time-series tree prediction method using suffix tree time-series which provides good prediction power and consumes less computation power. By providing dimensionality reduction to time-series indexing power is increased in time-series representation. Proposed method provides total historical coverage for prediction and also provides a mechanism to change prediction according to changing trend in time-series.

## 1.4 Thesis Organization

The rest of the thesis is organized as follows.

**Chapter Two 2**, *Literature Survey of time-series analysis and prediction*, describes various linear, nonlinear and probability based attempts to analysis time-series and predict time-series data.

**Chapter Three 3** , *Validation Environment Detail and Validation Set-up*, explains the details of the validation environment in which this new approach is developed and tested. How to set up test environment is described in this chapter.

In **chapter Four 4**, *Proposed Method*, an advanced approach using tree based adaptive time-series representation and probability based forecasting method is explained.

**Chapter Five 5**, *Simulation Methodologies and Performance Evaluation*, describes the methods of implementation. Tools used to implements this method and results achieved through this implementation.

Finally, in **chapter Six6** concluding remarks and scope for future work is presented.

# Chapter 2

# Literature Survey

As stated in introduction prediction techniques rely on the historical data processing upon which underlying trend or trends in the historical data are extracted and are used to derive a forecast decision. These method assumes that trend or trends present in historical data will be continued into future. Peter A Dinda [2] showed that traces of long, fine grain load measurements on a wide variety of workstation exhibit low means but very high standard deviations and maximums which suggests that workstations have plenty of cycles to spare to execute jobs, but the execution time of these jobs will vary drastically. Load is strongly correlated over time, but has a broad, almost noise-like frequency spectrum which implies that history-based load prediction schemes are feasible to effective forecast [1]. The key to making accurate predictions is to correctly model the relationship of the history data with the future values.

## 2.1 Non-adaptive time-series forecasting methods

The oldest and still the most widely used method for forecasting is time-series analysis, more simply known as trend-extrapolation. The method is often used where time and data are limited, produces the forecast of a single variable, through the use of historical data for the particular variable. The historical data can be manipulated through the use of sophisticated smoothing techniques. Time-series analysis is especially useful in producing short-term forecasts. The most common methods used for analyzing fluctuating patterns

were simple and exponential smoothing techniques. As these methods assumes that statistical property of time series remains constant over time they fail to provide good result because real time time-series data shows fluctuation in trend and statistical properties may change over time. To improve the results new methods were developed to normalize the time-series data by differentiating time series like random walk and then applying basic time-eries methods.

The last-value predictor method shown by Harchol-Balter and Downey[12] uses the current measured value as the predicted value of the next measurement. It can be expressed by the following formula

$$P_{T+1} = V_T \tag{2.1}$$

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

This method has comparatively low computation and storage overhead and is the default predictor in several current systems because of its simplicity.

Yule presented the notion of stochasticity in time series by saying that every time series can be regarded as the realization of a stochastic process[9]. Based on the wold decomposition theorem presented by Yule and walker Box and Jenkinns developed a coherent three stage iterative cycle for time series identification, estimation and verification. Box-Jenkins time-series processes Auto Regressive(AR), Moving Average(MA), Auto Regressive Moving Average(ARMA) and Auto Regressive Integrated Moving Average (ARIMA) were developed for computation of time-series and predict values[10, 11]. But all regression method developed on bases of Box-Jenkins are static methods. Co-efficient found from regression does not change when trend of the time-series changes. Whole iterative process has to be performed again to update co-efficient values. Even AR process find co-efficient of the model in deterministic time but MA process does not find co-efficient value in deterministic time. However there is no unique algorithm to specify model uniquely. Most resent version of ARIMA models X-12-ARIMA handles seasonality better for econometric time-series.

As one of the most famous resource prediction strategies, Network Weather Service (NWS) [3, 4, 5] is a distributed system that provides one step ahead prediction. It monitors

the CPU load periodically and performs a short-term forecast dynamically based on the history data. NWS maintains a set of various predictive models. To generate a forecast, the predictive models in the set are used simultaneously. The one with the lowest prediction error is chosen to predict the next value. This method of dynamically identifying a forecasting model can select the best model in the set and generate the most accurate prediction value.

Dinda et al. evaluated multiple linear models, including autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and autoregressive fractionally integrated moving average (ARFIMA) models [13]. Their results show that the simple AR model (also used in NWS) is the best model of this class because of its good predictive power and low overhead. More complex liner models are expensive to fit and hence difficult to use in a dynamic or real-time setting.

In [6] L. Yang et al. presented a tendency-based one-step-ahead time series prediction strategy, which assumes that the next value will increase or decrease along the change of direction, i.e. tendency. Compared to the common methods used in Network Weather System (NWS), tendency-based prediction outperforms in prediction errors with 36% less on average. However, Yangs approach is based on the linear increment of the last two values. In order to improve this method, Y.Zhang et al. [7, 8] proposed a strategy based on the tendency in several past steps and using polynomial fitting to generate the prediction value. The experimental results show that their method has a much better performance.

## 2.2 Adaptive time-series forecasting Methods

Above described all methods are not data adaptive means no feedback mechanism is provided to change the prediction co-efficient found while initial training period as the data changes. As the data trend changes change in prediction formula is necessary. The Kalman filter [14] is a set of mathematical relationships that processes the available measurements optimally (i.e. with the minimum mean squared error) in order to achieve the best pos-

sible estimate of the dependent quantities. The filter is simply a sophisticated method of exponential smoothing that can estimate bias errors and can cancel out much of the random errors. The basic filtering algorithm consists of forecasting future values of the dependent quantities, and then processing measurements with the filter in order to update the estimates of the dependent quantities to reflect the new information contained in the measurements.

These linear and non-linear approaches work well on time-series but takes a lot of computation poewr because of iterative calculation needed to perform prediction. Also decision is taken on the basis of last few historical analysis of time-series whole historical data is not taken into account. Time-series data falls into real numbers and as time-series representation method does not provide dimensionality reduction above method can not provide good indexing power [15, 16]. Eamonn Keogh and et al. showed a new suffix tree based time-series representation [17] to discover frequently occurring patterns and perform anomaly pattern detection which used symbolic representation of time-series shown in [18, 19]. Using same tree representation concept Ooi Boon Yaik et al. [20] presented CPU usage pattern discovery method using suffix tree to enable the suffix tree to discover variable length patterns and perform predictions.

## 2.3 Tree-based time-series forecasting method

Method shown by Ooi Boon Yaik et al. [20] showed good prediction result and higher computation speed compared to previous methods but is not adaptive method as change in time-series trend after training period does not reflect in prediction. Rather than pattern matching approach using suffix tree a probability based approach using suffix tree which creates prediction table for each pattern occurring in time-series data is represented. One step ahead or multi step ahead prediction is possible by using prediction table. After training period also prediction hit and miss is also taken into consideration and prediction table is changed accordingly.

# Chapter 3

# Validation Environment Detail and Validation Set-up

## 3.1 ST Microelectronics Set top box (STSTB)

A set-top box is a device that enables a television set to become a user interface to the Internet and also enables a television set to receive and decode digital television (DTV) broadcasts. DTV set-top boxes are sometimes called receivers. In the DTV realm, a typical digital set-top box contains one or more microprocessors for running the operating system, possibly Linux or Windows CE, and for parsing the MPEG transport stream. A set-top box also includes RAM, an MPEG decoder chip, and more chips for audio decoding and processing. The contents of a set-top box depend on the DTV standard used.

Audio Firmware Team develops software that is embedded in a ST STB and ports various kind of Audio Codec with STM STB. These codec are encoders (like AC3, MP3, MP2A etc), Decoders (like MP3, MP2A, AC3, DTS) and post process (like Bass Mgt, True Volume, Dolby Volume). Structure of ST STB is shown in figure 3.1

## 3.2 Software Build and execution environment

Flow chart of validation and testing is shown in figure 3.2.[21] To ease the testing procedure without relaxing the certification criteria are many software methodologies used as per ease

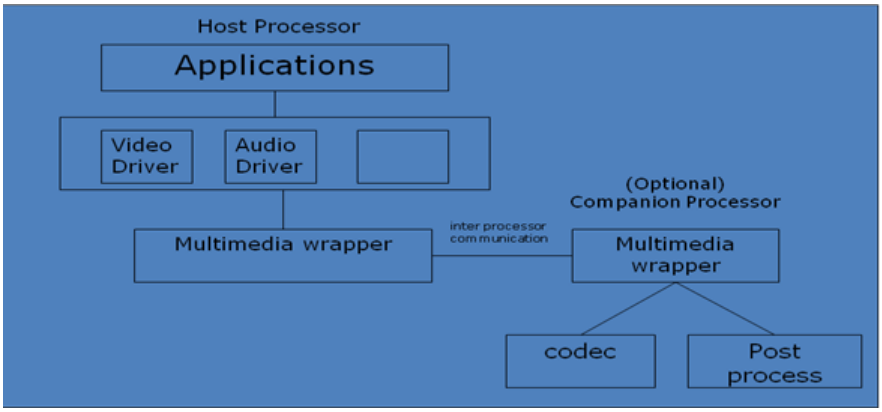



Figure 3.1: Structure of ST Set Top Box

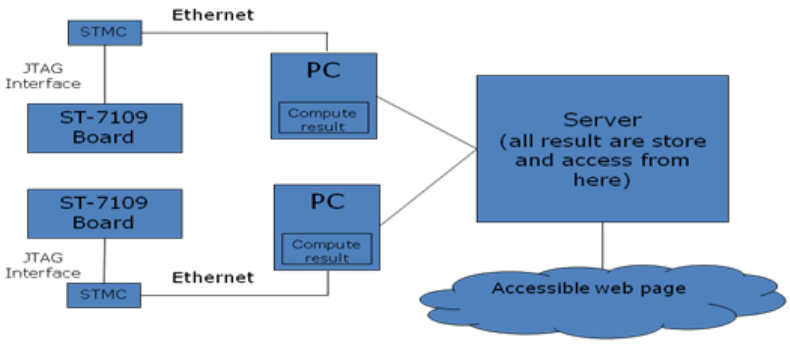



Figure 3.2: Flow chart of testing

and the certification requirements. Such an approach would ensure a maintainable, reproducible environment for reporting and publishing the required data. C language is the basic language which is used to deploy code for the embedded STB environment for its ease of programming and maintenance of a code base. Cygwin provides a primarily platform to test for compilation as well as intrinsic mapping errors while porting code for companion executable. The integrated bash shell provides a plethora of tools that could be used for an automated testing of the audio firmware.

Typically validation of an audio firmware build involves evaluating bit exactness between reference and testing platform. Implementation of a decoder or an audio post process

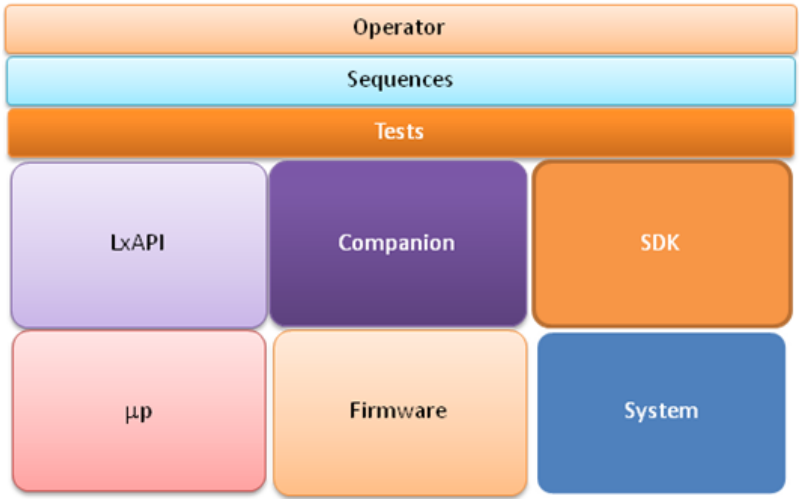



Figure 3.3: Layered architecture of the system

under test is abbreviated as DUT, here STSTB firmware. It is an C executable built to run in Cygwin hence called process.cygwin. Platform specific build of the audio firmware is called the board executable. It is required to capture the output from the decoder and the test against a reference implementation. Layered architecture of the whole system is shown in figure 3.3.

Above all layers the main layer is the operator interface. The command given by operator decides the sequences of tests to be executed on system under test (STSTB). The operator interface allows the operator to execute, stop and configure test sequence at runtime. Below operator layer comes the layer of global files and configuration files for system. Global files contain general codec related test-vector information which is common for all tests. Configuration files are of two types: test configuration files and hardware configuration files. Test configuration files are operator specific which contains the information of which test sequences are to be executed and mode of execution. Hardware configuration files contain configuration data of hardware like IP address of STSTB. According to configuration of test database of test-vector is accessed and libraries and binary file is selected for multicom. LxAPI code compile for board is loaded in board to carry out processing on stand-alone codec. To validate audio codec and firmware together companion is loaded on

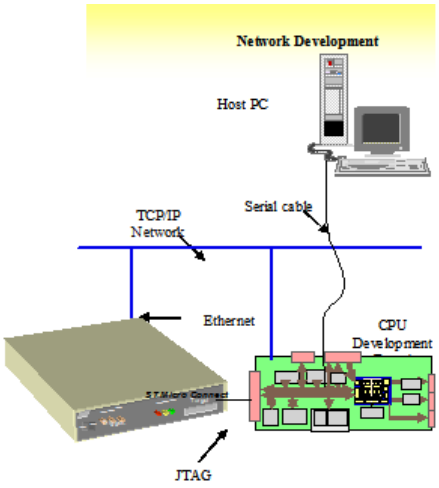



Figure 3.4: Testing Set-up

the board.

In the figure 3.4[22] whole setup of testing on STB board is shown. In this we have Host PC, Serial cable, Ethernet, JTAG, STMC and ST-7109 Board. All are connected through the TCP/IP networks. Host PC can also connect to the board with serial cable. We load the code on board through the networks to STMC and STMC to board. STMC is providing communication between PC and Board. And through Ethernet the results store in Host PC or server.

## 3.3 Command-line example

An example Command-line to run test-vector on STB board is explained below

```
-process.st231 -mp3 -i mp3TV/test.mp3 -o out_st.pcm -le -nch 2 -ws 16 -pcm_il
-md5 -no_out -flen
```

In regressing testing above command-line is used to run "test.mp3" named mp3 test-vector on board.

where,

a. -mp3 is the codec type

b. -i shows the next argument is input path of the test-vector

c. .mp3 is the extension of encoded file type

d. -o indicates that output

e. out_st.pcm is the raw PCM file generated by board

f. -le is use for little endian

g. -nch shows the next arguments is no of channel present in the output file

h. -ws shows next argument is word-size of each sample in bits for output file

i. pcm_il shows output Channels are in interlace mode (LRC LFE LS RS)

j. -md5 is checksum of output file which is to be calculated

Here in example some command-line options for input mp3 file have been shown. For different codecs like aac, ddpulse, flac etc command-line options may vary. E.g. for wav file as an input test-vector instead of -i -iwav is used and for output wave file -owav is used instead of -o. Some options have finite set of varying values. These all options show the type of output file system is desired to generate and kind of operation system needs to perform on the input test-vector.

# Chapter 4

# Proposed Method

In this chapter probability based CPU load time-series prediction method has been explained. This prediction approach is based on the observation that, those subsequences that appear frequent in history have a higher probability of occurring again in the future. The first stage in this method is to quantized the time-series data for dimensionality reduction and better indexing power. This quantized time series data is filled into suffix tree for training period and probability table is prepared for each pattern occurring into time-series. Thus all historical time-series data is utilized for prediction rather than last few historical data compared to other methods described in literature survey. Single step or multi step prediction for particular pattern is done from the probability values stored in prediction table for particular pattern. Prediction table is dynamically updated on basis of correct or wrong predicted value.

## 4.1 Symbolic representation of time-series

A time-series is first converted to Piecewise Aggregate Approximation (PAA) and then it is converted to a Symbolic aggregate approXimation (SAX).

### 4.1.1 PAA representation

A time series C of length n can be represented in a w-dimensional space by a vector $\bar{c} = \bar{c}_1, ...., \bar{c}_w$. The $i^{th}$ element of $\bar{C}$ is calculated by the following equation

$$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{4.1}$$

Simply stated, to reduce the time series from n dimensions to w dimensions, the data is divided into w equal sized frames. The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. The representation can be visualized as an attempt to approximate the original time series with a linear combination of box basis functions.

### 4.1.2 Discretization

After converting a time series database into PAA, we can apply a further transformation to obtain a discrete representation. It is desirable to have a discretization technique that will produce symbols with equiprobability. Given that the normalized time series have Gaussian distribution, we can simply determine the breakpoints that will produce a equal-sized areas under Gaussian curve. All break points are mapped to different symbols. The first smallest break point is mapped to "a". all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol b,. This dicretization process is shown in figure.4.1 In the figure 4.1 with n = 128, w = 8 and no. of SAX symbols are 3 namely a, b and c. So the time series is mapped to the word "baabccbc"

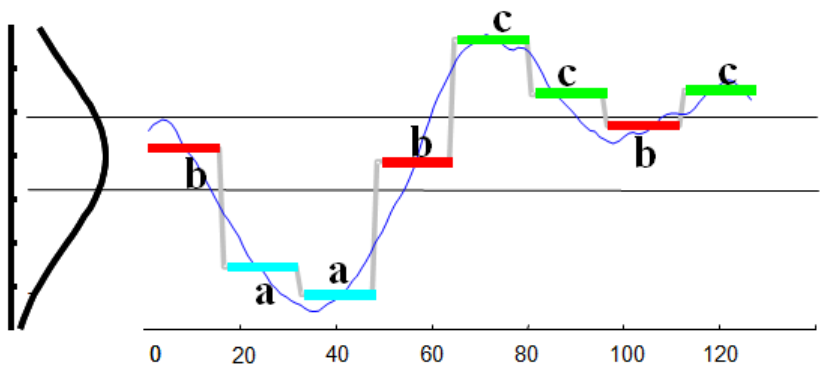



Figure 4.1: Dicretization Process

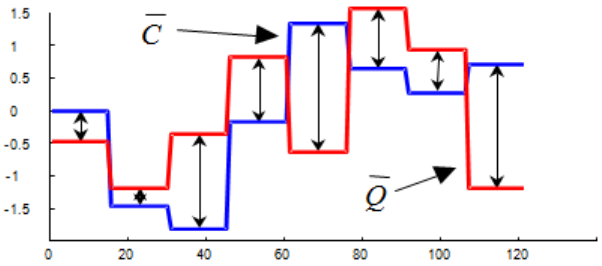



Figure 4.2: Distance between two time-series

### 4.1.3 Distance Measures

Given two time series Q and C of the same length n, Eq. 4.2 defines their Euclidean distance, and Figure 4.2 illustrates a visual intuition of the measure.

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \tag{4.2}$$

## 4.2 Training Period : Preparing Prediction Table

After transforming the historical data of time-series into SAX representation next step is to fill the suffix tree with the data and prepare a prediction table for one-step ahead

prediction. Suffix tree is is a tree with each node representing a pattern. I prepare a suffix tree with fixed length pattern. If the pattern length is 3 the suffix tree nodes will represent all combination of time-series symbols having length 3. Nodes of suffix tree with pattern length 3 and time-series representing symbols 3 will contain values aaa, aab, aac, aba, .. , ccc(total 27 pattern combinations). For pattern length 3 and time-series symbols 3 generated tree-structure is shown in figure 4.3.

Each node has 3 branches representing a symbol of time-series and each node represents a unique combination of 3 symbols. Each leaf has 3 counters representing occurrence of each symbol after respected pattern.

Sliding window of size 3 slides one step ahead at a time and sub sequence of size 3 is extracted. At each next step counter of the next occurring symbol is increased at the previously occurred sub sequence. This increases the occurrence of symbol following the pattern by one step.

e.g. for time series **a b c a c b b**

- Sub sequences having length 3 are {a b c},{b c a}, {c a c}, {a c b}, {c b b}
- So, sliding window slides across {a b c}, {b c a}, {c a c}, {a c b}, {c b b} one by one at each step.
- On second step when sliding window extracts sub sequence {b c a} counter of symbol "a" is increases at leaf having pattern {a b c}
- Similarly on third step when sliding window moves on {c a c} counter of symbol "c" is increased at leaf having pattern {b c a}

This process continues throughout the training period. After the training period is over each leaf has counter showing occurrence of each symbol next to the pattern. A table is prepared by counting probability of each symbol after each pattern. e.g. Pattern {a a a} has counters 20, 46, 34 for symbols "a", "b" and "c" respectively probability of occurring

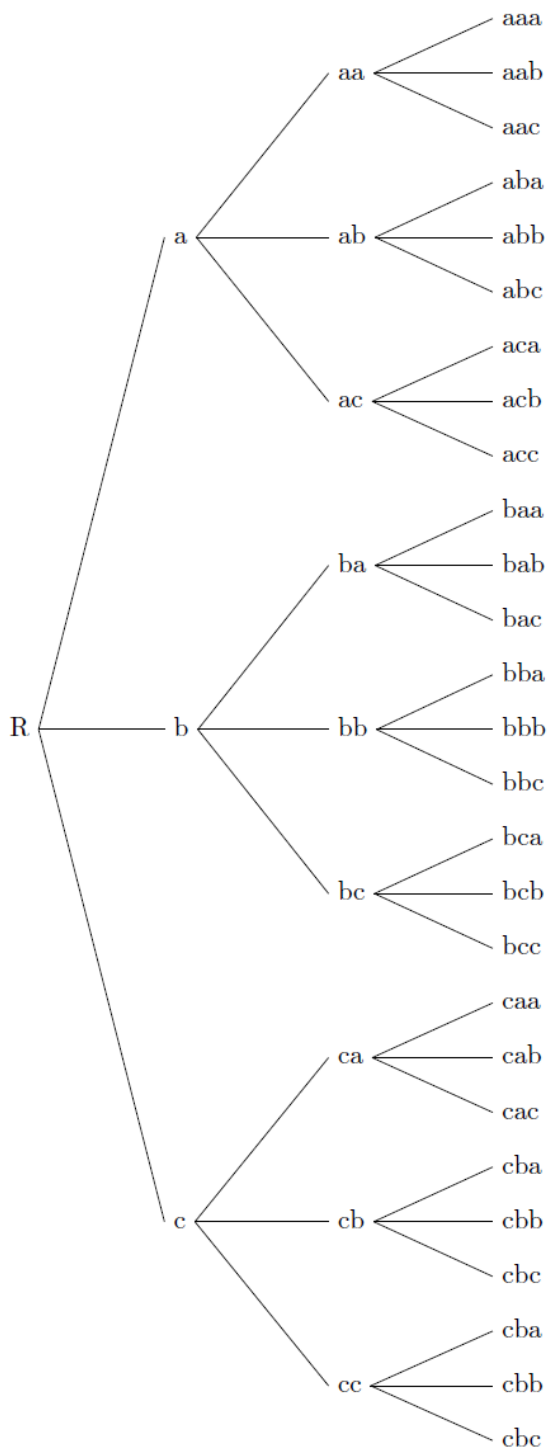



Figure 4.3: Suffix tree for time series

"a", "b" and "c" is 0.2, 0.46 and 0.34 respectively. Thus a table entry for each possible pattern if length equal to depth of tree is prepared and one step symbol occurrence is kept in it.

## 4.3 Updating Prediction Table

Initially while making the prediction table we stored the frequency of the symbol. Now we add hit and miss for each entry. Thus each table entry has three elements **(*prob , hit, miss*)**. ***prob*** is the probability that a symbol occurs after the pattern string (from the root to leaf) appears in the time series, ***hit*** is the times that our prediction succeeds and ***miss*** is that it fails. Intuitively, if the frequency freq of a particular string is high, then it will have a higher probability of appearing in the future. On the other hand, if our prediction of a certain string fails quite often, that is, its aggregate miss is large, then we have to lower the chance of choosing this string as the prediction result. In other words, our prediction error can be fed back by such aggregates. When prediction hit occurs ***hit*** count is increased else ***miss*** count is increased. For each symbol this count is applied and the symbol having the maximum count is the symbol which may occur after this pattern.

## 4.4 Prediction

### 4.4.1 Prediction while training

While training period is going on prediction is performed by moving average of last 10 time-series data values. This is because trend of time-series is best reflected by moving average method. After the training period is over occurrence of each character after each pattern is normalized and according to that probability table entry is done.

### 4.4.2 Single-step prediction

When time-series symbol event occurs, a sub-sequence string of last occurred symbols having length equal to depth of suffix-tree is extracted. For that string pattern prediction table entry is found. For each symbol probability of occurrence is count by ***(prob)*(hit)/(hit***

| Pattern | **abc** | | |
|---|---|---|---|
| Symbol | probability-percentage | hit | miss |
| a | 40 | 10 | 2 |
| b | 35 | 3 | 6 |
| c | 25 | 2 | 8 |

Table I: Prediction Table Entry

***+ miss)***. The symbol having maximum probability count is the symbol most likely to be appear as the future time-series symbol.

E.g. for pattern length three, quantization level three and sampling time 15 second if last occurred pattern is **a b c** Prediction of next value is done through prediction table. An example of prediction table entry for pattern **a b c** is shown in I. So considering hit-miss of each symbol the probability of symbols **a, b, c** is 0.33, 0.12 and 0.05 respectively. So the future symbol will be **a** according to probability table.

Here by emphasizing on ***prob*** value importance to historical data is given. While the trend of time-series changes ***miss*** count occurs and for correct prediction when trend is not changed the count of ***hit*** increases. So dynamically prediction mechanism is changed according to the current trend.

### 4.4.3 Multi-step prediction

The procedure of n-step multi-step prediction is n times single-step prediction assuming that the last predicted value is occurred. If CPU-load data is measures at each 15 second and 1 minute average load prediction is to be done than four one step ahead predictions are performed and averaged out considering last predicted value has occurred and it is used for next prediction. E.g. for pattern length three, quantization level three and sampling time 15 second if last occurred pattern is **a b c** and prediction table suggest that the most probable value in next 15 second is **b** then a second prediction is done for pattern **b c b** assuming that **b** has occurred. Averaging the both predicted output value we can find the average CPU load for next 30 seconds as two one-step ahead prediction for 15 seconds are made.

Prediction table is updated according to single-step prediction hit-miss only because n-step prediction is calculated from single step calculations only. For each one step prediction the prediction table is updated. Error in predicting earlier stages propagates into further prediction.

# Chapter 5

# Implementation and result

In this chapter implementation methodology is described and result of that implementation is discussed. To analyze the CPU-load I have measured CPU load of workstations from validation team at ST Microelectronis. Two days CPU-load analysis of five different workstations is tested to produce results. Average CPU-load of 15 seconds represents one sample. The whole code is written in C-programming language. Gnu Regression, Econometrics and Time-series Library (gretl) is used to generate results of linear time-series forecasting methods AR, MA and ARIMA as well as non-linear time-eries forecasting methods GARCH and VAR.

Accuracy of each prediction method is examined using mean squared error (MSE) between original value and the predicted value. In case of SAX representation original value is also mapped to lower bound of PAA representation and euclidean distance is found for comparison as shown in equation 4.2. So the equation to compare accuracy of t sampled data-set is

$$MSE_f(t) = \frac{1}{t}\sum_{i=0}^{t}(value(t) - prdiction(t))^2 \tag{5.1}$$

I started with sampling the CPU-load at equidistance time and representing it as CPU-load time-series. Then implemented PAA and SAX representation of the time-series. After PAA and SAX representation I produced results with and without implementing hit-miss updated prediction table structure and compared results. For various quantization levels and pattern length prediction results are compared in this chapter.

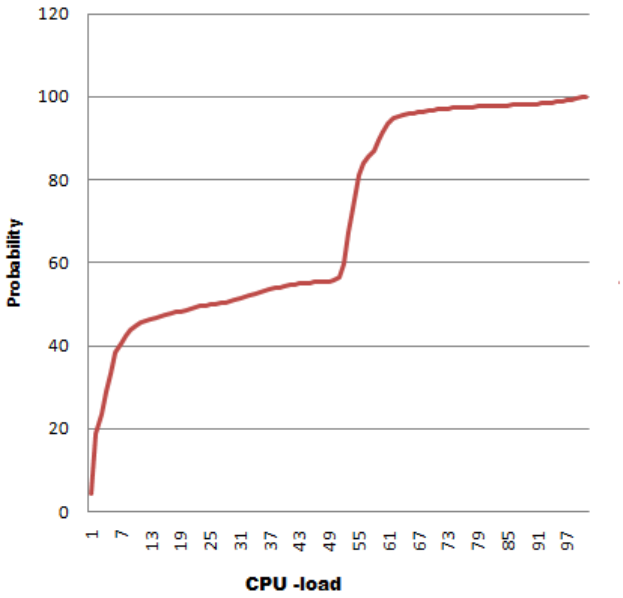



Figure 5.1: CDF plot of CPU-load time-series

## 5.1 SAX representation

To represent time-series of CPU-load in SAX model equiprobable region of CDF of time-series has to be found. The CDF of the CPU-load time-series is shown in figure 5.1

Given the CDF distribution of time series, we can simply determine the breakpoints that will produce a equal-sized areas under CDF. The number of breakpoints is decided by the number of symbol needed for SAX representation I gives the breakpoints for values of a from 3 to 10.

For three SAX representation symbols according to table I symbol "a" will be assigned to time-series data less than 6. Symbol "b" will be assigned to data greater than or equal to 6 but less than 53. Symbol "c" will be assigned to data greater than or equal to 53. The similar method can be followed for dividing the time-series in equiprobable area.

| breakpoint | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| B1 | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 3 |
| B2 | 53 | 26 | 8 | 6 | 5 | 5 | 4 | 4 |
| B3 | | 55 | 53 | 26 | 10 | 7 | 6 | 6 |
| B4 | | | 55 | 53 | 52 | 26 | 11 | 8 |
| B5 | | | | 57 | 54 | 53 | 50 | 26 |
| B6 | | | | | 58 | 55 | 53 | 53 |
| B7 | | | | | | 59 | 55 | 55 |
| B8 | | | | | | | 60 | 58 |
| B9 | | | | | | | | 62 |

Table I: A lookup table that contains the breakpoints that divide a CDF in an equiprobable area(3 to 10)

## 5.2 Implementation Strategy

### 5.2.1 Symbolic representation without PAA

CPU-load was sampled at each 15 second time. This CPU-load time series was then represented into discrete symbols according to probability distribution. This method of CPU-load time-series representation is having a drawback that it does not provide good results when outlier is present in that sampling interval. When sampling interval has an outlier sampling may fail to represent overall load of the sampling period as average of the CPU load is not presented by sampling.

### 5.2.2 SAX representation

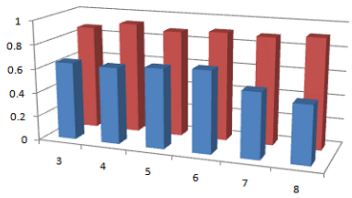
To create suffix tree of time-series symbol patterns we have two variables. One is how many symbols we need to represent the time-series and the second is the length of the symbol pattern. E.g. If we choose to represent time series in to symbols and pattern length two than we have four patterns to observe in the time-series. So here I have generated result with various combination of no of symbol and pattern length. To see the impact of the feedback based on hit os miss in prediction result considering hit-miss feedback structure and well as without considering hit-miss feedback structure are compared. For the comparison of results MSE error is considered as shown in equation 5.1. The result having lower MSE is better than others.

Training period to fill the tree is kept 6 hours. Average load of 15 seconds is represented

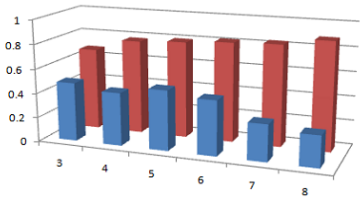
by a sample. So for training first 1500 samples are used to fill the suffix-tree. After filling the tree one-step occurrence frequency of each symbol is normalized. After normalization process prediction of the symbols is performed. If hit-miss structure is to be considered than hit-miss counters are updated according to prediction hit-miss and while prediction hit-miss ratio is also considered as described i 4.4.2.

## 5.3 Implementation Results

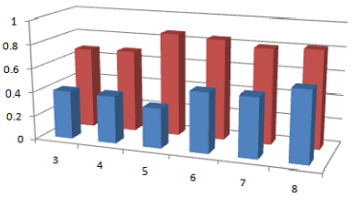
### 5.3.1 Hit-miss feedback implementation



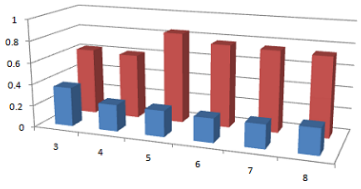
3 Quantization levels



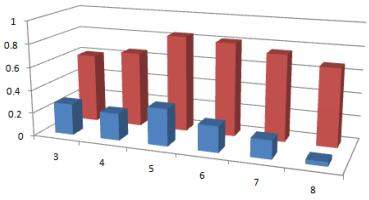
4 Quantization levels



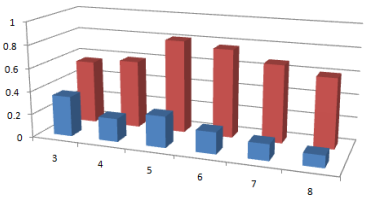
5 Quantization levels



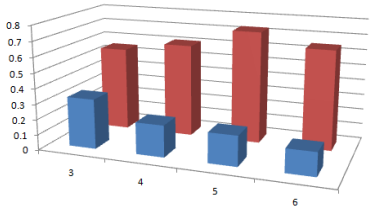
6 Quantization levels



7 Quantization levels



8 Quantization levels



9 Quantization levels

These figures show prediction hit ratio for 52416 predictions using different quantization levels with 3-10 pattern length. Red bar shows hit-ratio with hit-miss feedback implementation and blue bar shows hit-ratio without hit-miss feedback implementation. X-axis shows pattern length and Y-axis shows hit-ratio

Table II: Comparison of hit-miss implementation

To see the effect of feedback based hit-miss implementation prediction is performed using hit-miss implementation as well as without using hit-miss implementation was done. As seen from the figures of table II where hit-ratio of predicting correct quanta for one

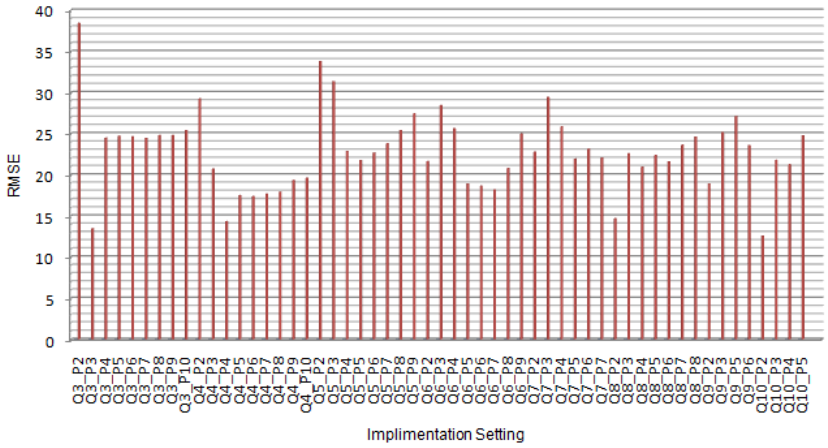



Figure 5.2: RMSE for different combination of quantization level and pattern length

step ahead prediction for various pattern length is shown. Blue bar shows hit-ratio without hit-miss feedback implementation and red bar shows hit-ratio with hit-miss feedback implementation. Implementation with hit-miss feedback performs better in each case. It is observed that for long pattern matching or for higher quantization levels feedback mechanism performs far better than the implementation without hit-miss feedback.

### 5.3.2 Effect of quantization level and pattern length

To find the effect of different combination of quantization level and pattern length on prediction accuracy I have used two parameters. One parameter is to analyze RMSE of prediction and the second parameter is hit-ratio of prediction. These both parameters reflects effect of quantization level and pattern length respectively. As quantization level decreases even for successful prediction hit RMSE may higher. On the contrary even for wrong prediction RMSE may be very small because of higher quantization levels. So analysis of both parameters presented here is done through the result generated by hit-miss feedback implementation for one-step ahead prediction.

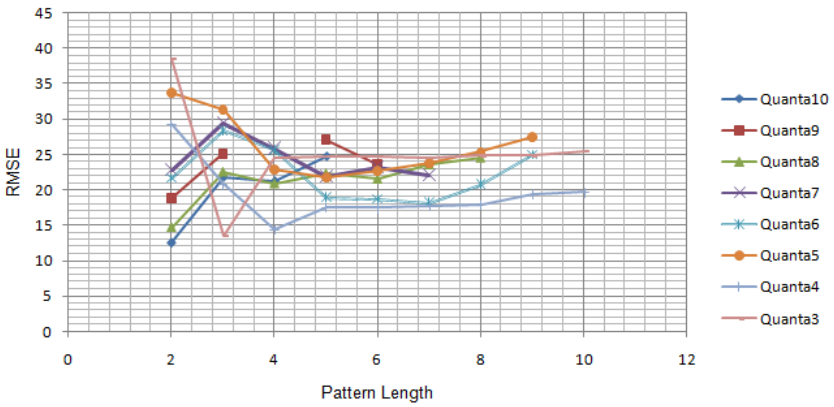



Figure 5.3: Scatter diagram of RMSE relationship with implementation setting

### RMSE analysis

Figure 5.2 shows the RMSE value for different combinations of quantization level and pattern length. In figure X-axis data shows the quantization level as well as pattern length and Y-axis shows RMSE. Here Q3_P4 means quantization level is 3 and pattern length is 4. From figure 5.2 we can see that for lower quantization levels 3, 4, 5 short pattern lengths and very long pattern lengths have higher RMSE. But lower quantization with medium range pattern lengths like 3, 4, 5 has lower RMSE compared to other pattern lengths. To see this clearly figure 5.3 provides scatter diagram of RMSE value for differen configuration of implementation. Lower quantization levels start with higher RMSE value, moves towards lower RMSE as pattern length increases but for higher pattern length the again start moving to ward higher RMSE value. On the contrary higher quantization levels like 8, 9, 10 starts with lower RMSE for shorter pattern length and RMSE increases as pattern length increases.

### Hit-ratio analysis

Figure 5.4 shows the hit-ratio of one-step ahead prediction for different combination of quantization level and pattern length. In figure X-axis data shows the quantization level as well as pattern length and Y-axis shows hit-ratio. Here Q3_P4 means quantization level is 3

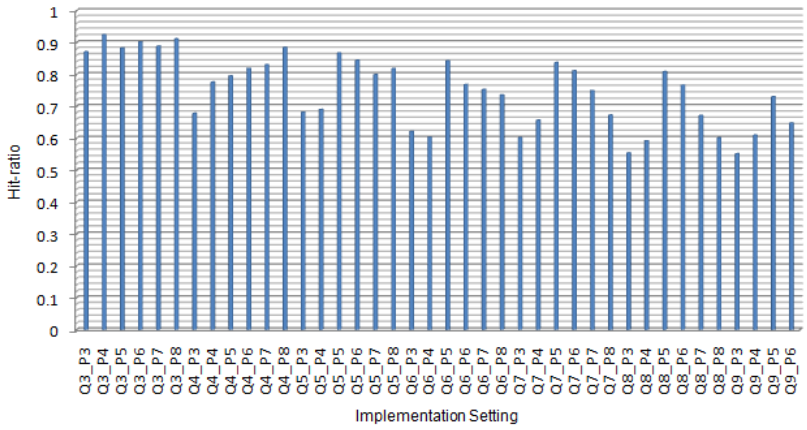



Figure 5.4: RMSE for different combination of quantization level and pattern length

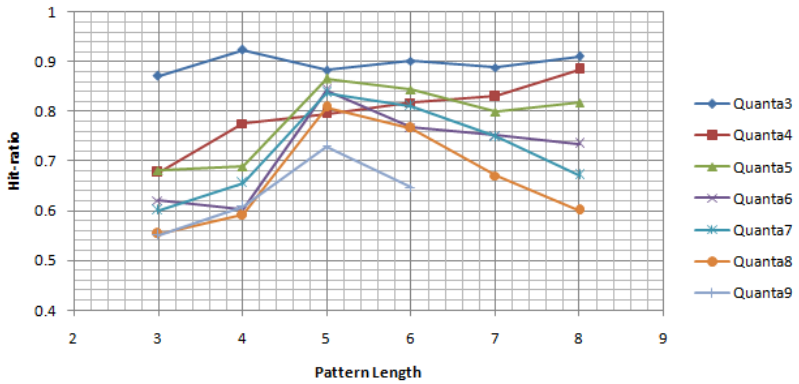



Figure 5.5: Scatter diagram of RMSE relationship with implementation setting

| Config | RMSE | Hit-Ratio | Config | RMSE | Hit-Ratio | Config | RMSE | Hit-Ratio | Config | RMSE | Hit-Ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Q3_P3 | 13.48 | 0.87 | Q4_P7 | 17.67 | 0.83 | Q6_P5 | 18.90 | 0.84 | Q8_P4 | 20.93 | 0.59 |
| Q3_P4 | 24.46 | 0.92 | Q4_P8 | 17.89 | 0.88 | Q6_P6 | 18.63 | 0.76 | Q8_P5 | 22.37 | 0.80 |
| Q3_P5 | 24.72 | 0.88 | Q5_P3 | 31.32 | 0.68 | Q6_P7 | 18.17 | 0.75 | Q8_P6 | 21.58 | 0.76 |
| Q3_P6 | 24.63 | 0.90 | Q5_P4 | 22.84 | 0.68 | Q6_P8 | 20.79 | 0.73 | Q8_P7 | 23.62 | 0.67 |
| Q3_P7 | 24.44 | 0.88 | Q5_P5 | 21.73 | 0.86 | Q7_P3 | 29.43 | 0.60 | Q8_P8 | 24.61 | 0.60 |
| Q3_P8 | 24.78 | 0.91 | Q5_P6 | 22.64 | 0.84 | Q7_P4 | 25.84 | 0.65 | Q9_P3 | 25.14 | 0.55 |
| Q4_P3 | 20.72 | 0.67 | Q5_P7 | 23.80 | 0.79 | Q7_P5 | 21.92 | 0.83 | Q9_P5 | 27.08 | 0.73 |
| Q4_P4 | 14.33 | 0.77 | Q5_P8 | 25.37 | 0.81 | Q7_P6 | 23.12 | 0.81 | Q9_P6 | 23.57 | 0.64 |
| Q4_P5 | 17.49 | 0.79 | Q6_P3 | 28.43 | 0.62 | Q7_P7 | 22.05 | 0.74 | | | |
| Q4_P6 | 17.38 | 0.81 | Q6_P4 | 25.61 | 0.60 | Q8_P3 | 22.59 | 0.55 | | | |

Table III: RMSE and hit ratio for different configurations

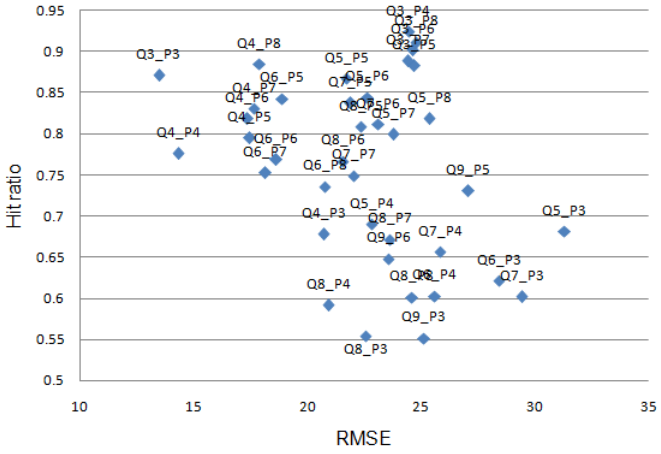



Figure 5.6: RMSE and Hit-ratio relationship

and pattern length is 4. For lower quantization medium higher pattern length gives higher hit ratio. Whereas for higher quantization level very short and very long pattern length gives lower hit-ratio. From scatter diagram 5.5 it is visible that higher quantization level have lower hit-ratio for shorter pattern length, it increases until very long pattern length comes and for very long pattern length it again approaches towards lower hit-ratio.

### Combined analysis of hit-ration and RMSE

The relationship of RMSE and hit-ratio for different configuration setting of quantization level and hit-ratio is shown in figure 5.6. Configurations like Q3_P3, Q4_P4 have lower

RMSE and higher hit-ratio. Configurations like Q8_P3 and Q9_P3, though provides comparatively lower RMSE has very low hit-ratio. Configuration of pattern length 3 has very high RMSE except pattern length 3. The best choice is to have configuration with lower RMSE and higher hit-ratio. Most of higher quantization configuration provides low hit-ratio but Some of them have very low RMSE too. So for Higher accuracy prediction we can apply high quantization levels and higher pattern length as increasing pattern length in higher quantization level increases hit-ratio. But in applications like CPU-load qunata prediction we can go for lower quantization levels with neither long nor small pattern lengths as the hit ration for them is high as shown in figure 5.6.

# Chapter 6

# Conclusion and Future Scope

## 6.1 Conclusion

In this dissertation, a probabilistic approach which extracts the symbolic representation of historical CPU load subsequences as well as their aggregate information is proposed, to predict the future symbols with probabilities based on aggregates that summarize the entire historical data, and finally output future subsequences for similarity search. The experiments were done to predict one-step ahead prediction of time-series data. The results of RMSE and hit-ratio of prediction for various quantization levels and pattern length with hit-miss feedback implementation shows that for lower quantization levels hit-ratio is high and RMSE value is low for higher pattern length. So this probabilistic approach can be used to predict the CPU load quanta. This method however fails to predict accurate results when quantization level is increased. So to predict results having very low RMSE, this method does not provide good performance.

## 6.2 Future Scope

- There is no method to determine the quantization level and pattern length which produces best results for given historical time-series data. A dynamic method can be developed to determine the quantization level and pattern length.

- This analysis was done only on bases of CPU-load measurement. A multivariate analysis method can be developed to see the effect of other parameters like memory usage on overall CPU-load.
- Time-series representation can be improved by taking MAXIMA and MINIMA into consideration for better performance for outliers in time-series.

# References


[1] P. A. Dinda and D. R. O'Hallaron, "The Statistical Properties of Host Load," presented at Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers (LCR 98), Pittsburgh, PA, 1998.

[2] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU availability of Time-shared Unix Systems," presented at Proceedings of 8th IEEE High Performance Distributed Computing Conference (HPDC8), 1999.

[3] R. Wolski, Dynamically Forecasting Network Performance Using the Network Weather Service, Journal of Cluster Computing, 1998.

[4] R. Wolski, N. Spring, and C. Peterson, Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service, Proceedings of SC97, 1998.

[5] R. Wolski, N.T. Spring, and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Future Generations of Computer Systems, 1999.

[6] L. Yang, I. Foster, and J.M. Schopf, Homeostatic and Tendency-based CPU Load Predictions, Intl Parallel and Distributed Processing Symp. (IPDPS03), pp. 42-50, 2003.

[7] Y. Zhang, W. Sun, and Y. Ingonuichi, CPU Load Predictions on the Computational Grid," IEICE Trans. Inf. and Syst., Vol. E90-D, No. 1, January 2007.

[8] Y. Zhang, W. Sun, and Y. Inoguchi, Predict task running time in grid environments based on CPU load predictions, Future Generation Computer Systems Volume 24, Issue 6, pp. 489-497, July 2008.

[9] Yule, G. U. (1927). "The method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers". Philosophical Transactions of the Royal Society London, Series A, 226, 267 298.

[10] Box, G. E. P., Jenkins, G. M. (1970)." Time series analysis: Forecasting and control". San Francisco7 Holden Day (revised ed. 1976).

[11] Box, G. E. P., Jenkins, G. M., Reinsel, G. C. (1994)." Time series analysis: Forecasting and control" (3rd ed.). Englewood Cliffs, NJ7 Prentice Hall.

[12] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing," presented at Proceedings of ACM Sigmetrics' 96 Conference on Measurement and Modeling of Computer Systems, Philadelphia, PA, 1996.

[13] P. A. Dinda and D. R. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction," presented at Proceedings of the 8th IEEE International Symposium on High-Performance Distributed Computing (HPDC-8), Redondo Beach, CA, 1999.

[14] Harvey, A. C. (1989). " Forecasting, structural time series models and the Kalman filter". Cambridge7 Cambridge University Press.

[15] Henrik Andre-Jonsson, "Indexing Strategies for Time Series Data", Linkping Studies in Science and Technology, Linkping University, Sweden.

[16] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, Sharad Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases",IN 4TH PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2000

[17] Jessica Lin, Eamonn Keogh, "Visually Mining and Monitoring Massive Time Series", KDD'04, August 22-25, 2004, Seattle, Washington, U.S.A.

[18] ]Keogh Eamonn, Kaushik Chakrabarti,"Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases", Journal of Knowledge and Information Systems, 3(3):263–286, 2000.

[19] Jessica Lin, Eamonn Keogh, "A Symbolic Representation of Time Series, with Implication for Streaming Algorithms", 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2003.

[20] Ooi Boon yaik, Chan Huah Yonh and Fazilah Haron, "CPU Usage Pattern Discovery Using Suffix Tree", IEEE, 2006, School ofComputer Science, Malaysia

[21] Data sheet of ST Set-top Box, STMicroelectronics.

[22] Set-Top Box ; by Dr. P. C. Jain, STMicroelectronics

# Index