

6-2018

NEURAL NETWORK ON VIRTUALIZATION SYSTEM, AS A WAY TO MANAGE FAILURE EVENTS OCCURRENCE ON CLOUD COMPUTING

Khoi Minh Pham
CSUSB, minhkhoy89us@gmail.com

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd>



Part of the [Computer and Systems Architecture Commons](#)

Recommended Citation

Pham, Khoi Minh, "NEURAL NETWORK ON VIRTUALIZATION SYSTEM, AS A WAY TO MANAGE FAILURE EVENTS OCCURRENCE ON CLOUD COMPUTING" (2018). *Electronic Theses, Projects, and Dissertations*. 670.

<https://scholarworks.lib.csusb.edu/etd/670>

This Thesis is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

NEURAL NETWORK ON VIRTUALIZATION SYSTEM, AS A WAY TO MANAGE
FAILURE EVENTS OCCURRENCE ON CLOUD COMPUTING

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Khoi Minh Pham
June 2018

NEURAL NETWORK ON VIRTUALIZATION SYSTEM, AS A WAY TO MANAGE
FAILURE EVENTS OCCURANCE ON CLOUD COMPUTING

A Thesis
Presented to the
Faculty of
California State University,
San Bernardino

by
Khoi Minh Pham

June 2018

Approved by:

Yasha Karant, Advisor, School of Computer Science and Engineering

Ernesto Gomez, Committee Member

David A. Turner, Committee Member

© 2018 Khoi Minh Pham

ABSTRACT

Cloud computing is one important direction of current advanced technology trends, which is dominating the industry in many aspects. These days Cloud computing has become an intense battlefield of many big technology companies, whoever can win this war can have a very high potential to rule the next generation of technologies. From a technical point of view, Cloud computing is classified into three different categories, each can provide different crucial services to users: Infrastructure (Hardware) as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS). Normally, the standard measurements for cloud computing reliability level is based on two approaches: Service Level Agreements (SLAs) and Quality of Service (QoS). This thesis will focus on IaaS cloud systems' Error Event Logs as an aspect of QoS in IaaS cloud reliability. To have a better view, basically, IaaS is a derivation of the traditional virtualization system where multiple virtual machines (VMs) with different Operating System (OS) platforms, are run solely on one physical machine (PM) that has enough computational power. The PM will play the role of the host machine in cloud computing, and the VMs will play the role as the guest machines in cloud computing. Due to the lack of fully access to the complete real cloud system, this thesis will investigate the technical reliability level of IaaS cloud through simulated virtualization system. By collecting and analyzing the event logs generated from the virtualization system, we can have a general overview of the system's technical reliability level based on number of error

events occur in the system. Then, these events will be used on neural network time series model to detect the system failure events' pattern, as well as predict the next error event that is going to occur in the virtualization system.

ACKNOWLEDGEMENTS

I would like to thank the entire faculty and staff of the School of Computer Science and Engineering at California State University of San Bernardino, especially my former advisor Dr. Zhengping Wu and my advisor Dr. Yasha Karant, they have contributed a lot to my recent achievements. I would like to thank my committee member Dr. Ernesto Gomez and Dr. David Turner. I would also like to thank my extended family for their support.

DEDICATION

To my dad who has always been giving me inspiration about scientific knowledge and beyond, I thank you.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS.....	v
LIST OF TABLES	ix
LIST OF FIGURES	x
1. INTRODUCTION	
1.1 Cloud Computing Classification and Reliability Levels.....	1
1.2 Feed Forward Neural Network.....	7
1.3 Process Mining, Episode Mining in Data Science	11
1.3.1 Frequent Episode Mining.....	12
1.3.2 Events Log and Process Mining	14
1.4 Outline of Thesis	17
2. VIRTUALIZATION IN CLOUD ENVIRONMENT SIMULATION	
2.1 Chapter Introduction	19
2.2 Virtualization in IaaS	19
2.3 Virtual Machine Monitor	24
2.3.1 Xen Hypervisor.....	26
2.3.2 Oracle VirtualBox	39
2.4 System Events Log.....	42
2.4.1 Windows Events Logging	44
2.4.2 Linux System Logging	51
2.5 Common Users' Behavior Simulation	54
2.6 Data Collection Strategy	57

2.7	Chapter Summary.....	59
3.	NEURAL NETWORK TIME SERIES MODEL AND SYSTEM FAILURE PATTERN DETECTION	
3.1	Chapter Introduction	60
3.2	NARX Network Model.....	60
3.3	Data Preparation.....	65
3.4	Levenberg-Marquardt Backpropagation Training.....	69
3.5	Training Validation	75
3.6	Chapter Summary.....	81
4.	CONCLUSION AND FUTURE DIRECTION	
4.1	Conclusion	83
4.2	Future Direction	84
	APPENDIX A: SETUP VIRTUALIZATION ENVIRONMENT AND DATA COLLECTING.....	86
	APPENDIX B: DATA PREPROCESSING	104
	APPENDIX C: DESIGNING NEURAL NETWORK NON-LINEAR TIME SERIES AND PLOT VALIDATION	110
	REFERENCES.....	136

LIST OF TABLES

Table 1.1	SLAs Evaluating Metrics	6
Table 1.2	A Fragment of a Simulated Cloud Computing System Event Log, Simplified Version, each Line Represent an Activity Instance.....	16
Table 2.1	Virtual Machine Specifications	27
Table 2.2	The Sample Set of Traditional Event Log.....	43
Table 2.3	Windows Event Log's Severity Levels [37].....	45
Table 2.4	EventType Value Reference Table from MSDN [37].....	51
Table 2.5	Severity Level in Ubuntu 14.04 LTS x64 [39].	52
Table 3.1	NARX Net Input Data	62
Table 3.2	Visualization of NARX Net Taking Input Data with Input-Delay Equal to '0'	64
Table 3.3	Severity Level Reference Table	65
Table 3.4	Program Name Reference Table	66
Table 3.5	Data Collection during December 2017	76
Table 3.6	Training Validation with each Dataset Using MSE'	76

LIST OF FIGURES

Figure 1.1	Cloud Computing Paradigm [25].....	2
Figure 1.2	Cloud Services Clustering [25].....	4
Figure 1.3	Feed Forward Neural Network Model	8
Figure 1.4	A Neuron in Feed Forward Neural Network.. ..	9
Figure 1.5	Sigmoid Transfer Function Visualization.....	11
Figure 1.6	Example of an Event Sequence.....	12
Figure 1.7	Three Types of Process Mining Techniques [19].....	17
Figure 2.1	User's Manageable Components in IaaS.....	20
Figure 2.2	IaaS Simplified Architecture [26]	21
Figure 2.3	Generic Virtualization Architecture.....	22
Figure 2.4	Reliability Model of Virtualization Servers and VMs [26]	23
Figure 2.5	Type 1 and Type 2 Hypervisor [33].....	25
Figure 2.6	Xen Project Architecture [34]	26
Figure 2.7	List CPU Specifications.....	29
Figure 2.8	Xen Architecture and Kernel in dmesg Log File	31
Figure 2.9	Configure Network Interface in Xen Virtualization.....	33
Figure 2.10	Ubuntu Virtual Machine Configuration File.....	34
Figure 2.11	Windows Virtual Machine Configuration File.....	35
Figure 2.12	Display VMs' Logical Volumes in the Hosted LVM Partition.....	37
Figure 2.13	Create Ubuntu 14.04 LTS VM and Windows XP SP2 VM.....	38
Figure 2.14	Xen Hypervisor Domains List.....	38

Figure 2.15	Destroy Ubuntu 14.04 LTS VM and Windows XP SP2 VM Command in Xen Hypervisor	39
Figure 2.16	VirtualBox and VMware Architecture.....	41
Figure 2.17	The Flow-net Logs Approach [36].	44
Figure 2.18	Common User's Use Case Diagram.	56
Figure 2.19	Auto User Simulation Activity Diagram.....	57
Figure 3.1	NARX Net Model Deployed in this Thesis Experiments.....	64
Figure 3.2	Weight Adjusting in One Epoch in NN Training Process.....	70
Figure 3.3	Visualize the Chain Rules in Finding Gradient between MSE and Weight.....	71
Figure 3.4	Plot Regression of NARX Net 1 When Training with [X2, T2]	78
Figure 3.5	Plot Regression of NARX Net 2 When Training with [X4, T4]	79
Figure 3.6	Plot Response of NARX Net 1 When Training with [X2, T2]	80
Figure 3.7	Plot Response of NARX Net 2 When Training with [X4, T4]	81

1. INTRODUCTION

1.1 Cloud Computing Classification and Reliability Levels

Cloud computing nowadays has become an optimized solution in IT industry. It has many major advantages compare to traditional service model, such as the ability to “pay as you use”, and the ability to access the computational resources immediately without spending time building and configuration the system. Saving not only time and money, cloud computing has also broken the hardware limitation in IT industry, due to the flexibility and scalability it has. One user can now use a thousand-times more powerful computational system, without having to pay the enormous amount of money to build that system. Moreover, by moving to cloud, users do not have to spend time on repairing, maintaining and updating their business computational system, which normally cost as much as 30% of the amount paying to build and configure the system.

In general, the cloud-computing model includes the cloud providers and the cloud users. Cloud providers are companies that provide cloud services to users. Figure 1.1 shows a brief explanation about cloud computing, users use their electronics devices to access their cloud storage, cloud media contents, or their cloud virtualization machines [25]. The big advantage of this model is user's electronics devices specs can be as low as possible, possibly only need to have

good quality screens and medium-high speed network adapters, to keep user's experiencing cloud services fluently.

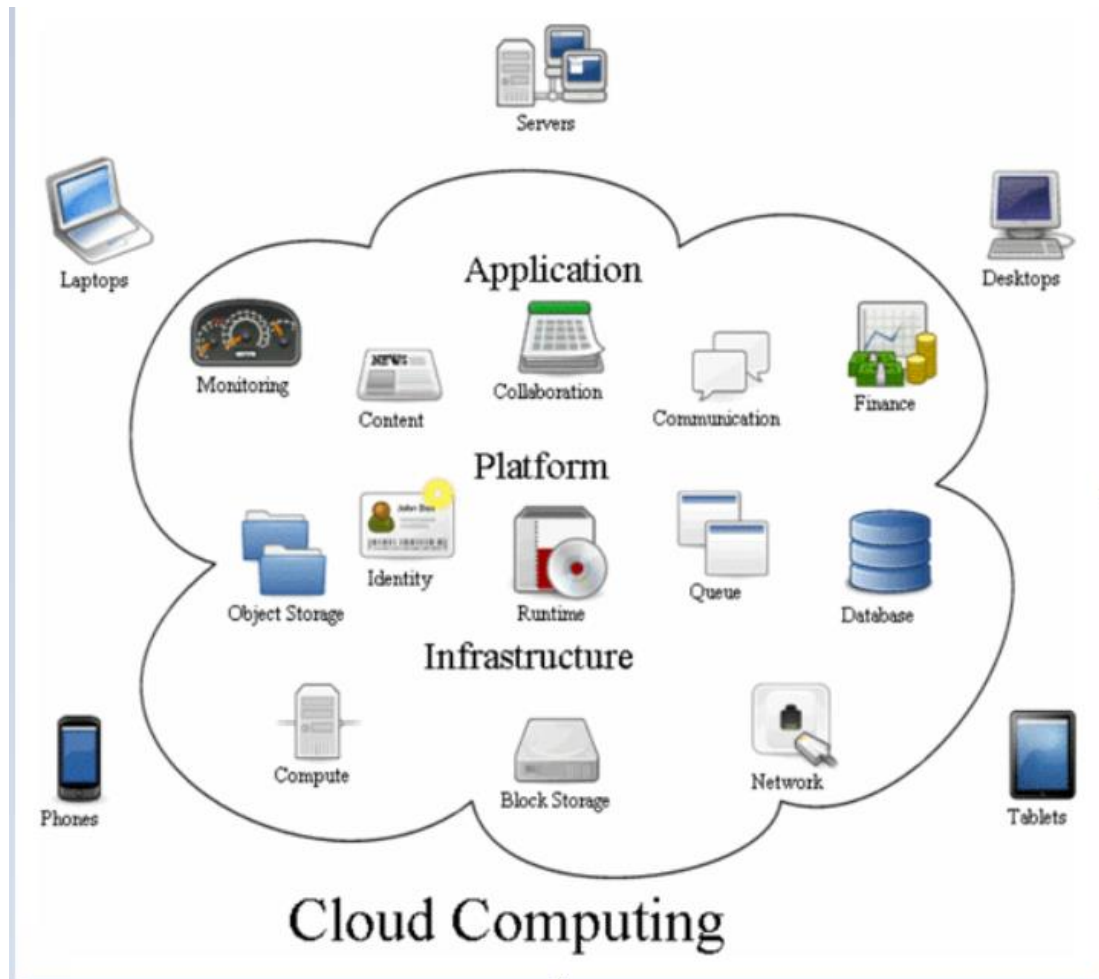


Figure 1.1: Cloud Computing Paradigm [25].

Let UserHours be the number of hours users consume on their service, revenue be a total income the service providers can have, and cost is the expense service

providers have to pay to maintain their services, the pre-condition to consider cloud computing towards traditional server models is described as below [22]:

$$UserHours_{cloud} \times (revenue - Cost_{cloud}) \\ \geq UserHours_{datacenter} \times (revenue - \frac{Cost_{datacenter}}{Utilization})$$

Cloud computing refers to delivering application as a service through the internet, together with hardware and software systems needed to launch that application. According to different service patterns, cloud computing has three different types of models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

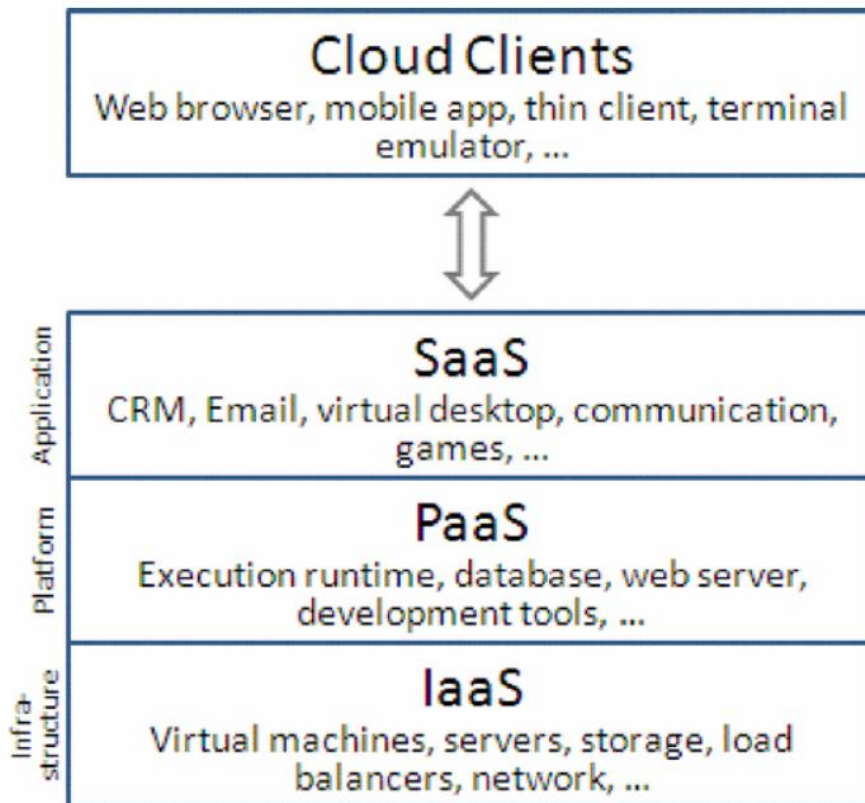


Figure 1.2: Cloud Services Clustering [25].

Figure 1.2 gives a brief summarization of cloud services that belong to those three models [25]. Among those three, IaaS is the most basic layer, IaaS is the model that cloud providers are using to provide hardware computational power through the internet. Many advanced technology company is considering IaaS as their crucial strategies in cloud computing such as Amazon EC2, VMware vCloud ... A virtualization environment will be simulated in chapter two of this thesis, to experiment the IaaS virtual machines cloud's reliability level.

To validate the reliability level of IaaS, there are two major approaches: Service Level Agreements (SLAs) - is the contract signed between cloud service

provider and cloud users; and Quality of Service (QoS) - is the guarantee from cloud service providers about providing cloud service continuously to cloud users, with the qualities no lower than the qualities mentioned in SLAs.

Reliability level refers to the amount of continuous time the service providers promise to provide the stable service to users, it is measured by the number of failures occur on any users' virtual machines [24]. Consider P_R as the probability errors may occur, n_f is number of failure tasks occurred, n_t is the total number of tasks expecting to run normally at the meantime, T is the total time cloud users running the service.

$$Reliability = P_R \times T = \left(1 - \frac{n_f}{n_t}\right) \times T$$

Table 1.1: SLAs Evaluating Metrics [23].

	Parameter	Description
Provider SLAs	Throughput	The amount of data can be received within per time unit
	Security	Encryption, authentication and authorization
	Billing	Cost of service and method of calculating
	Monitoring	Monitoring agencies and monitoring method
	Privacy	The way of Storage and transmission
	Compensation	measures for SLA violations
	Load balance	Ability to handle concurrent access
	Recovery	Backup and ability to recovery from a disaster
User SLAs	Availability	Time proportion of Continuous delivery
	Reliability	Probability to keep providing specific service
	Response time	Time from request to receive a response
	Service level	Different service editions
	internationalization policy	Multi-language support
	Migration	Service migration to other platform
	Support service	Service of help and support

Table 1.1 introduces some common metrics used for cloud services SLAs evaluation [23].

1.2 Feed Forward Neural Network

Artificial Intelligence (AI) has gained massive attentions from the computer science communities these days [30]. In general, AI is a technique to design and program the machine's intelligence simulating some certain level of human intelligence. Hence, AI can help the machine to think and do like us human in some specific cases, under particular conditions.

Neural network, in some cases can be referred as machine learning, is one direction of AI that is inspired by the human brain neuron model. In theory, neural network resemblances biological neuron model in the way knowledge are obtained through learning, and are stored within inter-connection neuron nodes (technically known as weight). Accordingly, the purpose of neural network is to train the AI system by feeding it some input dataset, after several learning process, the AI system will have some general knowledge about the dataset, and can show some general understanding regarding to the dataset. There are two types of learning: supervised learning - where users feed the machine with both input dataset and target dataset, and unsupervised (blind) learning - where users feed the machine with only input dataset so the machine has to figure out rules or patterns all by itself.

Neural network has huge varieties of distinct models, each model serves different learning purposes. Feed forward neural network is one of the most commonly use neural network model.

The basic concept behind neural network learning process is, let X be the input dataset with 1000 elements, T be the target dataset with 1000 elements. The first 500 elements in X and T will be fed to the neural network for training purpose. When the network reaches optimized training result (training complete), it will take the last 500 elements in X as the input and generate a set of 500 elements Y as the output prediction. This Y set will be compared with the last 500 elements in T to validate the neural network's accuracy (or correctness).

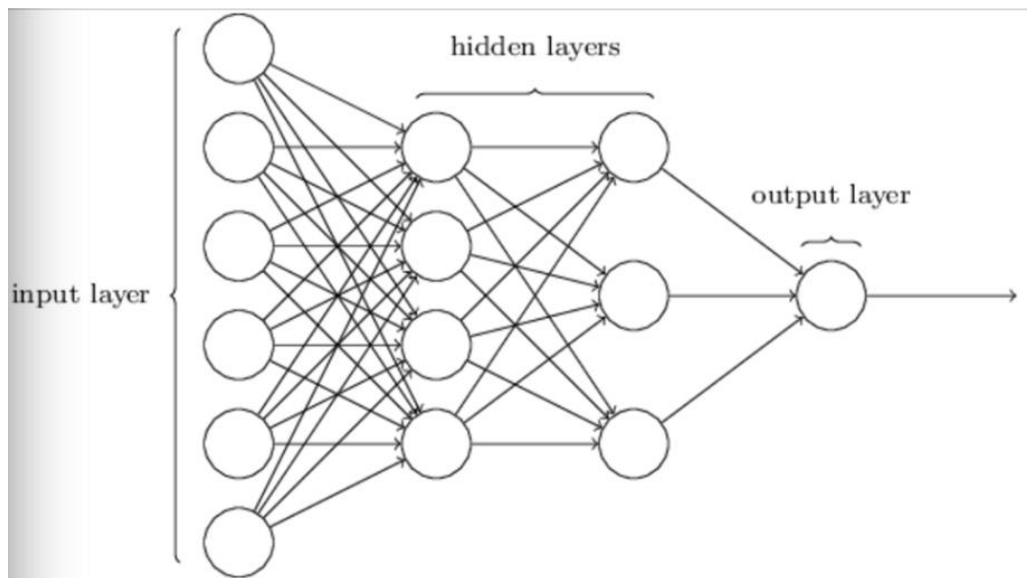


Figure 1.3: Feed Forward Neural Network Model.

Figure 1.3 illustrates the basic feed forward neural network model, the network model includes three layers, one input layer, one or more hidden layer(s), and one output layer; each layer has multiple nodes, and one node

connects directionally to all adjacent nodes from the previous layer and the next layer [29]. When being used, the data are passed from nodes in one layer to nodes in the next layer, until reaching the output layer, and no feed-back between layers.

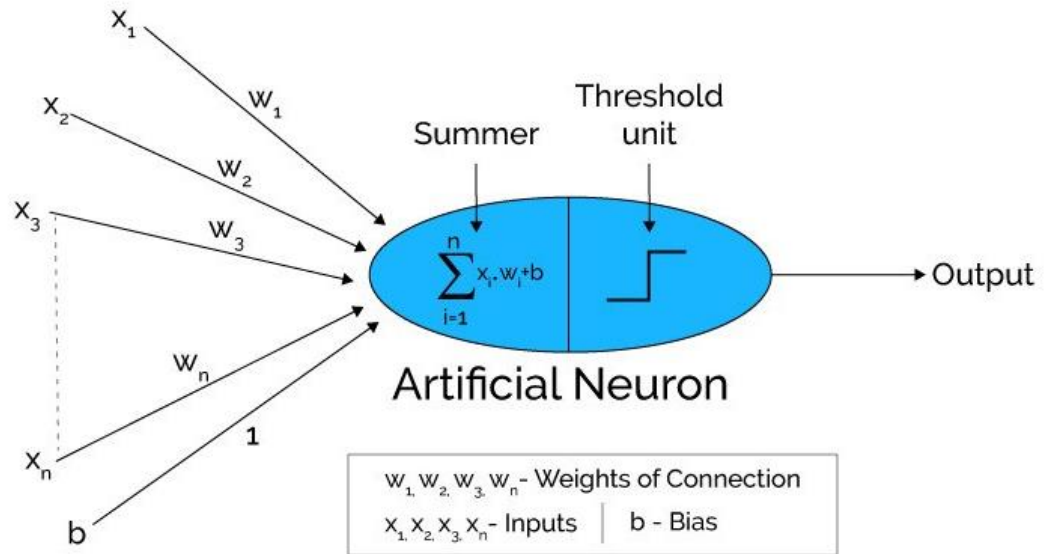


Figure 1.4: A Neuron in Feed Forward Neural Network.

Figure 1.4 visualizes a neuron in feed forward neural network; $x_1, x_2 \dots x_n$ are the nodes in one previous layer; $w_1, w_2 \dots w_n$ are the weights of the connections from $x_1 \dots x_n$ to a node x_j in current layer, bias b is an extra node in one hidden layer with $x_b + w_b = 1$, all the input factors are defined as

$$a = \sum_{i=1}^n x_i \cdot w_i + b_i \quad (1)$$

The output of x_j to a node a_k in the next layer is defined as

$$a_k = f(a) = \frac{1}{1+e^{-a}} \quad (2)$$

f is sigmoid transfer function, the default transfer function of neural network.

During the training process, the dataset is separated into two smaller dataset, the first dataset is used for learning, the second dataset is use for validating, when the training process is done, the validating result is measured by MSE value. Let $Y = \{y_1, y_2 \dots y_n\}$ be the set of output results, $T = \{T_1, T_2 \dots T_n\}$ be the set of original target dataset, the formula for MSE is

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - T_i)^2 \quad (3)$$

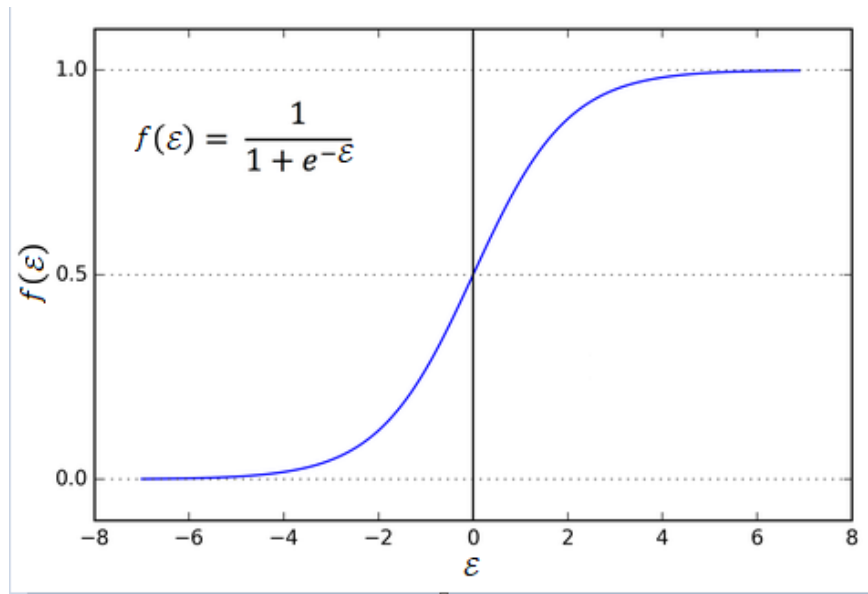


Figure 1.5: Sigmoid Transfer Function Visualization.

Feed forward neural network can be used for multi purposes such as input-output curve fitting, pattern recognition and classification, clustering (un-supervised learning), and dynamic time series prediction. This thesis will use neural network time-series model to predict next failure event in simulated IaaS cloud environment.

1.3 Process Mining, Episode Mining in Data Science

As the amount of data has drastically increased in sizes and categories, all thanks to the support of mass storage technology, the term “Data Science” is now getting more popular in this industry. In general, Data Science is the techniques that exploit existed data and try to give answers for the questions grouped into these four categories [19]: what happened? (Reporting), why did it

happen? (Diagnosis), what will happen? (Prediction), and what is the best that can happen? (Recommendation).

Data Science includes different fields of computer science: Data mining, Machine learning, Process mining, Databases, Large scale distributed computing, Visualization and visual analytics, Behavioral/social sciences, Privacy and security... many of them are used in this thesis.

1.3.1 Frequent Episode Mining

In some particular dataset such as System event logs dataset, where events are stored in the order of time, an episode is a set of events starting from an initial activity event and ending at a finalizing activity event. A frequency of an episode indicates how frequently that episode occurs in one particular sequence. Frequent episode mining is the technique that is used to detect all episodes that has frequency higher or equal to the user's pre-defined boundary [21].

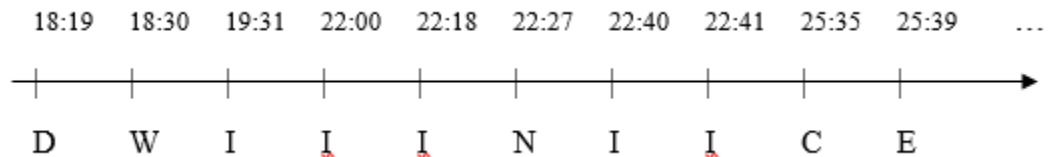


Figure 1.6: Example of an Event Sequence.

Let $E = \{e_1, e_2, e_3 \dots e_m\}$ be the set of events, with $e_i = \{E_{i1}, E_{i2} \dots E_{in}, t_i\}$, an *event sequence* \vec{S} in the order of time is denoted as $\vec{S} = \{(E_1, t_1), (E_2, t_2) \dots (E_n, t_n)\}$

(With $n \leq m$). As a result, $E_i \subseteq E$, t_i is the time stamp of event E_i where $t_i < t_k \Leftrightarrow 1 \leq i < k \leq n$. Figure 1.6 shows an event sequence $\vec{S} = \langle (\{D\}, \text{epoch (18:19)}), (\{W\}, \text{epoch (18:30)}), (\{I\}, \text{epoch (19:31)}), (\{I\}, \text{epoch (22:00)}), (\{I\}, \text{epoch (22:18)}), (\{N\}, \text{epoch (22:27)}), (\{I\}, \text{epoch (22:40)}), (\{I\}, \text{epoch (22:41)}), (\{C\}, \text{epoch (25:35)}), (\{C\}, \text{epoch (25:39)}) \rangle$. Here epoch (t) is a function to convert t from mm:ss format to epoch timestamp format.

Episode α ($\alpha \neq \emptyset$) is an ordered set of events $e_1 \rightarrow e_2 \rightarrow \dots \rightarrow e_i \rightarrow \dots \rightarrow e_k$, where $e_i \in E$ and $i \in [1, k]$, e_i happens before $e_j \Leftrightarrow 1 \leq i < j \leq k$. The length of an episode is determined by the number of events in that episode. For example $\alpha = I \rightarrow I \rightarrow C \rightarrow E$ has length of 4, and is called a 4-episode.

Given two episodes α and β , $\alpha = e_1 \rightarrow \dots \rightarrow e_i \rightarrow \dots \rightarrow e_n$, $\beta = e'_1 \rightarrow \dots \rightarrow e'_j \rightarrow \dots \rightarrow e'_k$. β is *sub-episode* of α (or α is *super-episode* of β) denote as $\beta \preceq \alpha \Leftrightarrow$ In the order of time, $\forall e'_o \in [1, k], \exists e_p \in [1, n]$ in the same order of time, that $e'_o = e_p$. For example, 2-episode $\beta = I \rightarrow E$, 3-episode $\alpha = I \rightarrow C \rightarrow E$, $\beta \preceq \alpha$, but $\beta' = E \rightarrow I$ is not sub-episode of α .

Given episode $\alpha = e_1 \rightarrow \dots \rightarrow e_i \rightarrow \dots \rightarrow e_k$, $[t_1 \dots, t_i \dots, t_k]$ is an *occurrence* of α if and only if (1) $\forall i \in [1, k], t_i \subseteq e_i$; (2) $t_1 < t_2 < \dots < t_k$; and (3) $t_k - t_1 < \delta$ (δ is a *maximum occurrence window* pre-defined by users). In figure 1.6, [epoch (18:19), epoch (18:30), epoch (19:31)] is an occurrence of episode $D \rightarrow W \rightarrow I$

Let episode $\alpha = e_1 \rightarrow \dots \rightarrow e_i \rightarrow \dots \rightarrow e_k$, α has two occurrences $[t_1 \dots, t_i \dots, t_k]$ and $[t'_1 \dots, t'_i \dots, t'_k]$. These two occurrences are *equivalent* if $t_1 = t'_1$ and $t_k = t'_k$, in such situation, these two occurrences are considered the same.

Accordingly, an occurrence of α is denoted as $(\alpha, [t_1, t_k])$, and $[t_1, t_k]$ is called an occurrence window of α . For example, in figure 1.6, episode $\alpha = N \rightarrow I \rightarrow C$ is denoted as $(\alpha, [\text{epoch (22:27), epoch (25:35)}])$, the two occurrences $[\text{epoch (22:27), epoch (22:40), epoch (25:35)}]$ and $[\text{epoch (22:27), epoch (22:41), epoch (25:35)}]$ of α are equivalent.

Given two occurrences $[t_1, t_2]$ and $[t'_1, t'_2]$, if $t_1 < t'_1$ and $t'_2 < t_2$, $[t'_1, t'_2]$ is subsumed by $[t_1, t_2]$. An occurrence $[t_1, t_2]$ is a *minimal occurrence* if there is no occurrence $[t'_1, t'_2]$ that can be subsumed by $[t_1, t_2]$. We denote $\text{moSet}(\alpha)$ as the set of all distinct minimal occurrences of α . The *support of an episode α* (Denote as $\text{sp}(\alpha)$) is the number of all elements in $\text{moSet}(\alpha)$. In figure 1.6, $\text{moSet}(I \rightarrow I) = \{[\text{epoch (19:31), epoch (22:00)}], [\text{epoch (22:00), epoch (22:18)}], [\text{epoch (22:40), epoch (22:41)}]\}$ when $\delta = 2$, and $\text{sp}(\alpha) = 3$.

An episode α is a frequent episode if $\text{sp}(\alpha) \geq \text{min_sup}$ (min_sup is a minimum support threshold pre-defined by user).

1.3.2 Event Logs and Process Mining

Process mining techniques use the system transactions data or business event logs data to discover different types of models of the business process. Sometimes, people in the industry/business do not even know about the existence of such models until applying the process mining technique. In many cases, process mining can also be used to monitor and improve existed business process, by investigating the event logs.

Let Event Log $L = (E, AN, AI, C, act, type, time, res, case, name)$ be a tuple [20], E, AN, AI, C are the sets of events, activity names, activity instances, and cases relatively, $E = \{e_1, e_2 \dots e_n\}$. $U_{ET} = \{Information, Warning, Error\dots\}$ is the list of event types, $U_{res} = \{Dhcp, IPsec, HTTP, DCOM, W32Time, Setup\dots\}$ is the list of system resources. $type \in E \rightarrow U_{ET}$ and $res \in E \rightarrow U_{res}$ are the functions mapping events to event types and resources. $case \in AI \rightarrow C$ and $act \in AI \rightarrow AN$ are the functions mapping activity instances to cases and activity names. In Tab 1.2, the first process has case ID = 1, starting from activity = "WG-Warning" where $res = "Dhcp"$, ending at activity = "WG-Error" where $res = "Setup"$. The starting point and ending point of one process is pre-defined by user, and the rule will be applied to the whole event log dataset.

Table 1.2: A Fragment of a Simulated Cloud Computing System Event Log, Simplified Version, each Line Represent an Activity Instance.

case ID	resource	activity	time generated	...
1	Dhcp	WG-Warning	11/10/2017 00:18:19	...
1	IPSec	WG-Information	11/10/2017 00:18:19	...
1	Workstation	WG-Information	11/10/2017 00:18:30	...
1	HTTP	WG-Information	11/10/2017 00:19:31	...
1	Setup	WG-Information	11/10/2017 00:22:00	...
1	Setup	WG-Error	11/10/2017 00:22:00	...
2	DCOM	WG-Information	11/10/2017 00:22:18	...
2	EventLog	WG-Information	11/10/2017 00:22:18	...
2	EventLog	WG-Information	11/10/2017 00:22:18	...
2	IPSec	WG-Information	11/10/2017 00:22:27	...
2	SRService	WG-Information	11/10/2017 00:22:27	...
2	W32Time	WG-Information	11/10/2017 00:22:27	...
2	DCOM	WG-Error	11/10/2017 00:22:27	...
...

There are mainly three types of process mining: *process discovery* is most useful in establishing new process model; *conformance checking* is used to compare and monitor existing process model, for compliance checking and repairing; *performance analysis* use the alignments between process model and real-time system performance (created by conformance checking) to reveal the system bottle necks.

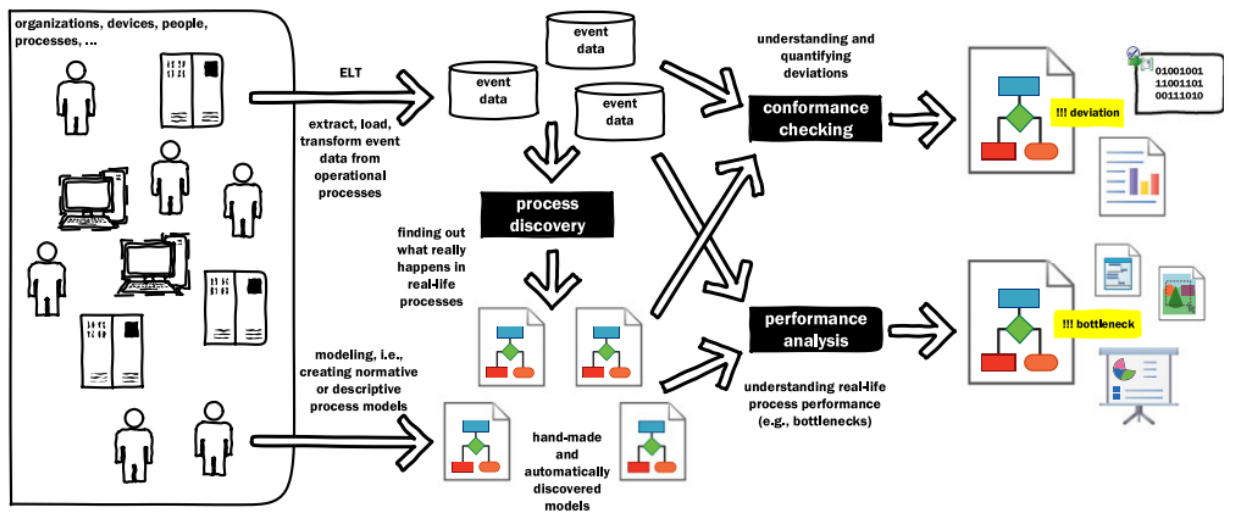


Figure 1.7: Three Types of Process Mining Techniques [19].

1.4 Outline of Thesis

From the natural characteristics of system event logs in operating systems (OS), it is obvious that a set of system event logs in a running OS can be considered as a set of events in process mining. As, described above in Chapter 1.3, process mining itself is a type of data science that achieves knowledge from feeding input. Therefore, we believe the right neural network model can achieve the same goal if we feed it the same data. What we want to investigate here, is whether this theory is still applicable on cloud-alike platform where multiple OSs running on one physical machine. Is it is, there is a very high chance our neural network can recognize and detect frequent events that lead to system failure, this will be very helpful for cloud providers in many situations. Thus, the remainder of the thesis is structured as follows: virtualization systems and cloud environment simulation, as well as generating dataset for later training purpose will be

described in Chapter 2. Chapter 3 deploys neural network time-series system, implements data training process and experimenting results. Chapter 4, the final chapter, evaluates the experimenting results, gives conclusion, mentions about challenges and proposes future work. The Appendix provides additional details about development platform that we were using, and original source codes that were created exclusively for this thesis purpose.

2. VIRTUALIZATION IN CLOUD ENVIRONMENT SIMULATION

2.1 Chapter Introduction

In order to serve the neural network training purpose, at the first stage, it is necessary to have a well-prepared dataset collected from actual running cloud system. However, due to the lack of accessibility to the cloud provider's system event logs, an alternative solution has been approached. Technically, cloud server is a normal virtualization server that can be accessed by users through the internet, regardless of physical location. Therefore, there is a possibility to investigate the cloud system performance through the similar set-up virtualization system, assuming the network issue in this environment is set to 0. The difficulties in this approach are, besides the system set up similarly, there must also have virtual user simulator or auto macro that simulates common users activity on cloud guest machines (or virtual machines). In fact, this is a key point to generate random system event logs dataset for the experiments.

2.2 Virtualization in IaaS

In IaaS, users are given a guest machine which has its own storage, CPU (possibly GPU in some cases), memory, network, and operating system. This provides users the ability to do everything they theoretically can do on their own PC. By using IaaS, users can save a lot of time spending to configure and set up the machine. It is also very useful for those who only need some computational

power for a short time, because IaaS often allows user to scale up and scale down their rented machines very flexibly. For example, an Amazon EC2 user who is using two guest machines with a single core 3.4GHz CPU and 2GB of memory on each, can extend it to 100 machines each has the same specs, if he needs to use that much of computational power within the next 60 seconds (or more). In addition, Amazon EC2 will only charge that user extra one minute of using 50 times more service's resources. Because Amazon service is pay per minute, therefore 60 seconds is actually the minimum threshold.

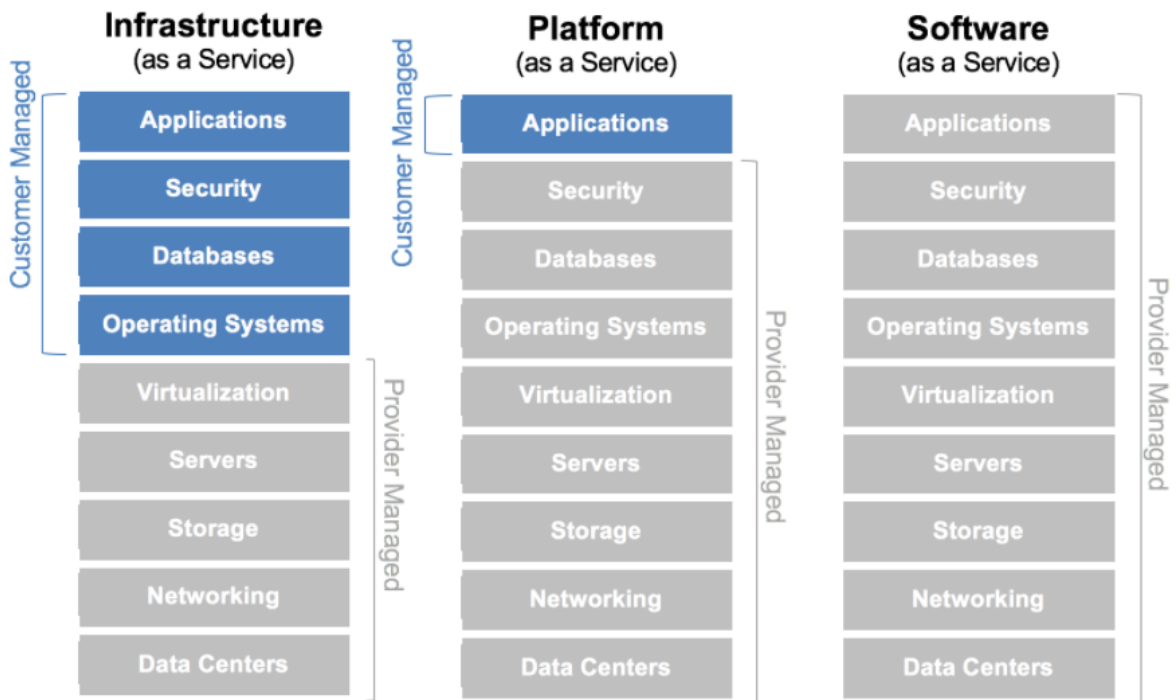


Figure 2.1: User's Manageable Components in IaaS.

Figure 2.2 is a simplified version of a complete IaaS architecture. In this example, IaaS is hosted by physical servers, and it includes 3 major components: network node, compute node, and storage node. Each guest machine given to IaaS's user is a various combination of these 3 nodes. This architecture is also very similar to the architecture of virtualization system, as shown in figure 2.3. This leads to the concept of virtualization, which is the key technology behind IaaS. In fact, those guest machines in IaaS are the virtual machines running on their physical servers.

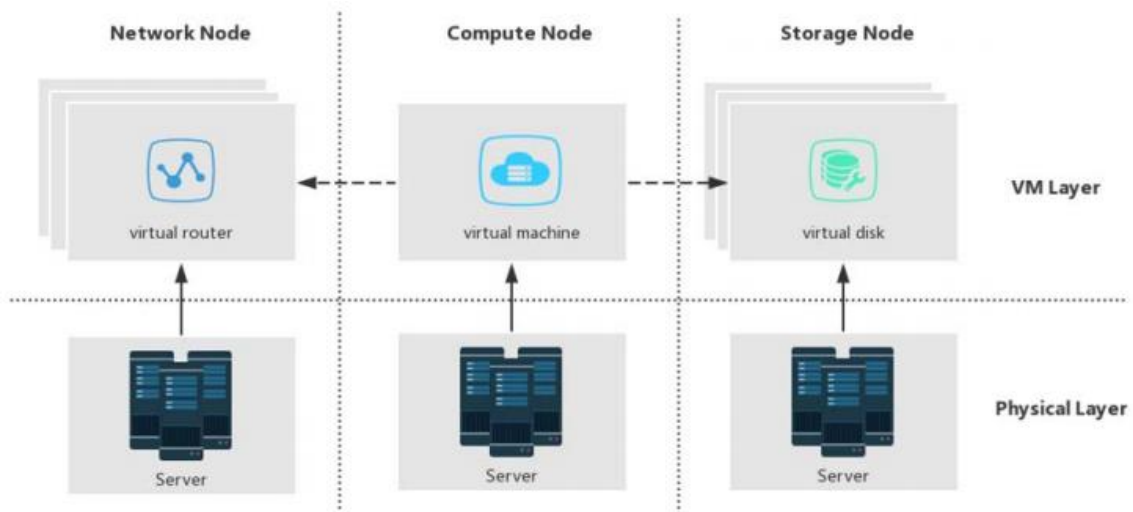


Figure 2.2: IaaS Simplified Architecture [26].

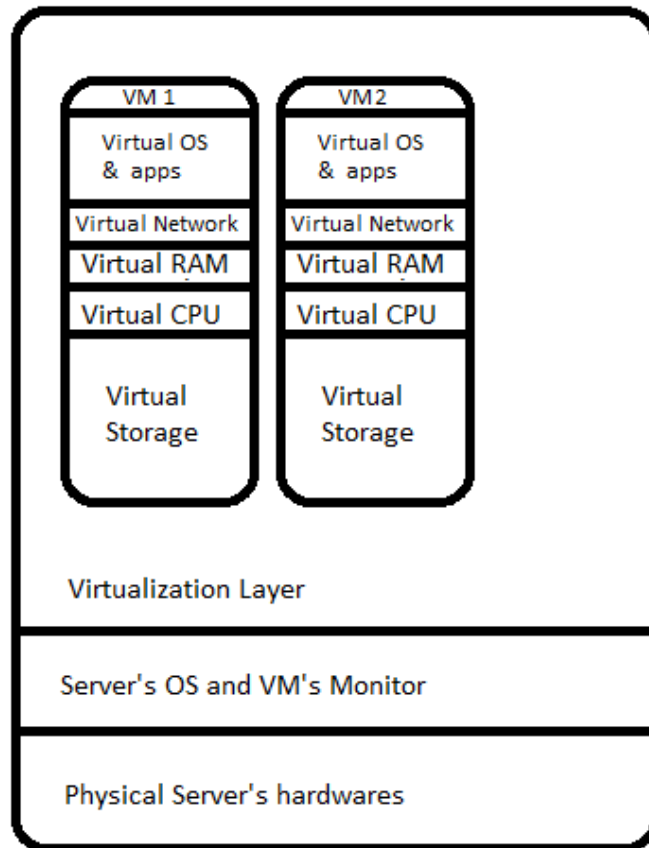


Figure 2.3: Generic Virtualization Architecture.

Virtualization is the technique to partition one mainframe computer running on single computer hardware into smaller logical instances. This is believed to give better efficiency and reduce maintenance effort. In virtualization system, every user has their own choice of OS running on their virtual machine, multiple virtual machines (VM) can run on the same computer, each virtual machine theoretically has full features of a real computer, the host OS is the only thing that communicate with the physical computer hardware.

The two significant benefits of virtualization are sharing resources – one physical hardware is shared between multiple VMs, this helps maximize the efficiency of hardware usage; and isolation – all the VMs in one virtualization system are isolated from each other, thus becomes invisible to each other, apps running on one VM cannot see apps running on other VMs.

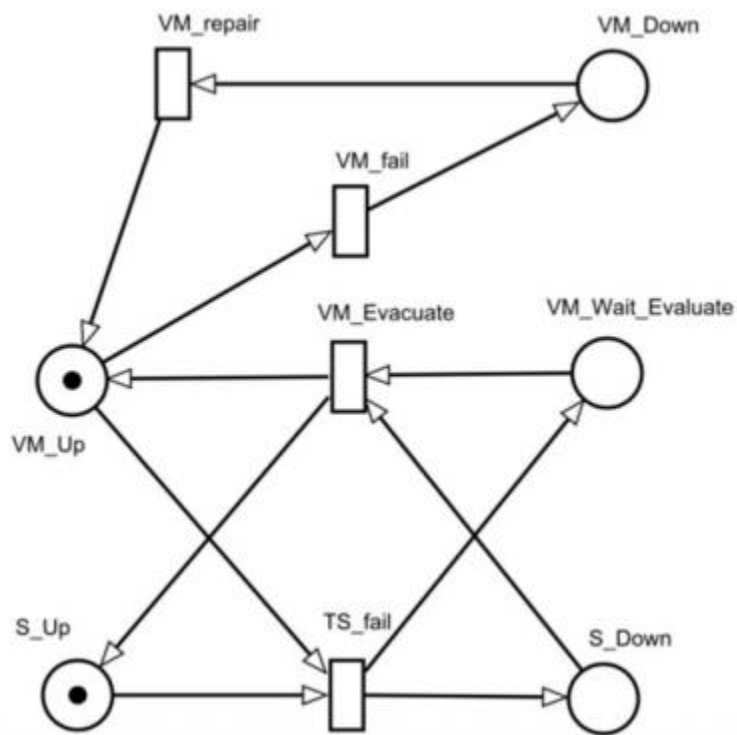


Figure 2.4: Reliability Model of Virtualization Servers and VMs [26]

Figure 2.4 illustrates the reliability model of a virtualization system that has multiple physical servers and VMs. TS_fail and VM_fail represent the failure

events occur in servers and in VMs. These failures can interrupt the continuity of VM usage on the user's side, because the system will need more time to repair it or restart completely, thus affect the reliability of the service.

In PC industry, virtualization technology integrates in AMD's CPU is called AMD-V, while Intel calls it VT-x. Only those CPUs that have AMD-V or VT-x can have VMs running on them. These features can be enabled/disabled in Bios under CPU settings tab. Besides, in order to have high performance computing (HPC) VMs, the CPUs also need to support Input-Output Management Unit (IOMMU). This feature can be turned on/off in the Bios settings, under Northbridge settings tab.

2.3 Virtual Machine Monitor

In Virtualization environment, all the VMs are isolated from each other by the virtual machine monitor (VMM). VMM provides HW interfaces to VMs, and control how VMs use these HW. VMM manages information about VM's states, CPU usage, RAM, database, these are called visible resources. Some modern CPUs also have invisible resources such as shared bus, shared memory, shared cache... these invisible resources can affect system behavior under different situations.

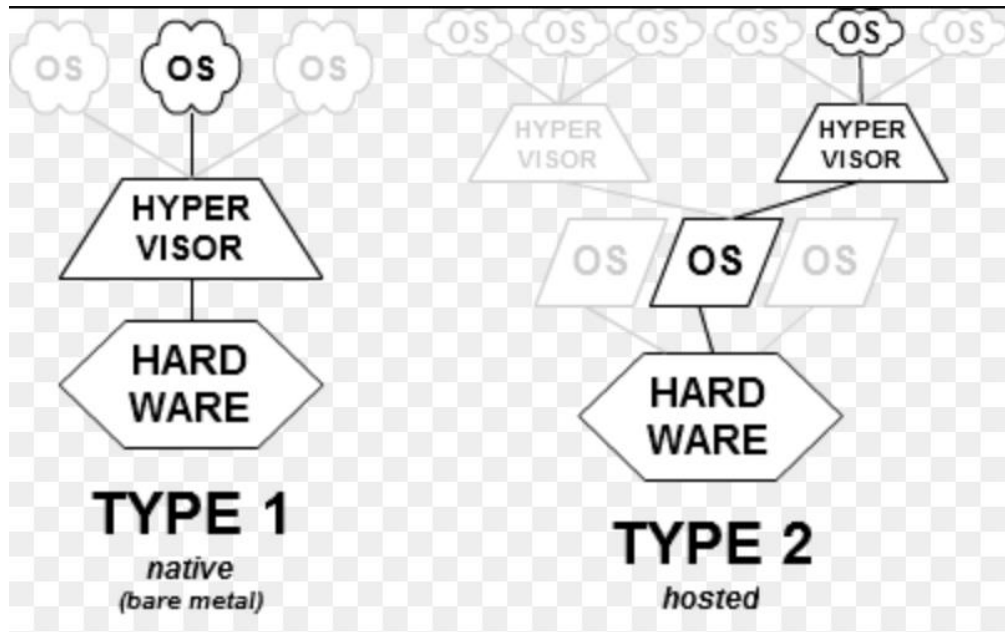


Figure 2.5: Type 1 and Type 2 Hypervisor [33]

VMM, or Hypervisor, are classified in two types [33]: Hypervisor type-1, also called bare metal hypervisor, runs directly on physical HW, and acts as a complete equivalent OS; Hypervisor type-2, or hosted hypervisor, runs on host OS, and is similar to any other distinct application.

The virtualization technology using in this thesis are Xen Project (hypervisor type-1) and VirtualBox (hypervisor type-2), because these two are the most typical technology that are being used by leading cloud providers. Xen Project is being used by Amazon EC2, which is the biggest cloud provider in the industry. And VirtualBox has a very similar architecture to VMware vCloud, which is the biggest cloud providers that is using hypervisor type-2 virtualization technology.

2.3.1 Xen Hypervisor

Xen hypervisor is an open-source type-1 hypervisor that is widely used in high-tech industries: server/desktop virtualization, IaaS, embedded system, and hardware appliances. The Xen Project hypervisor is the most common use virtualization system among cloud providers these days.

Xen supports two types of VM guest:

Para-virtualized (PV) Guest is introduced by Xen Project as a virtualization technique that does not require virtualization extension (AMD-V or VT-x) from physical CPU [34].

Hardware-assisted Virtual Machine (HVM) Guests: is a traditional virtualization technique that requires virtualization extension (AMD-V or VT-x) from physical CPU.

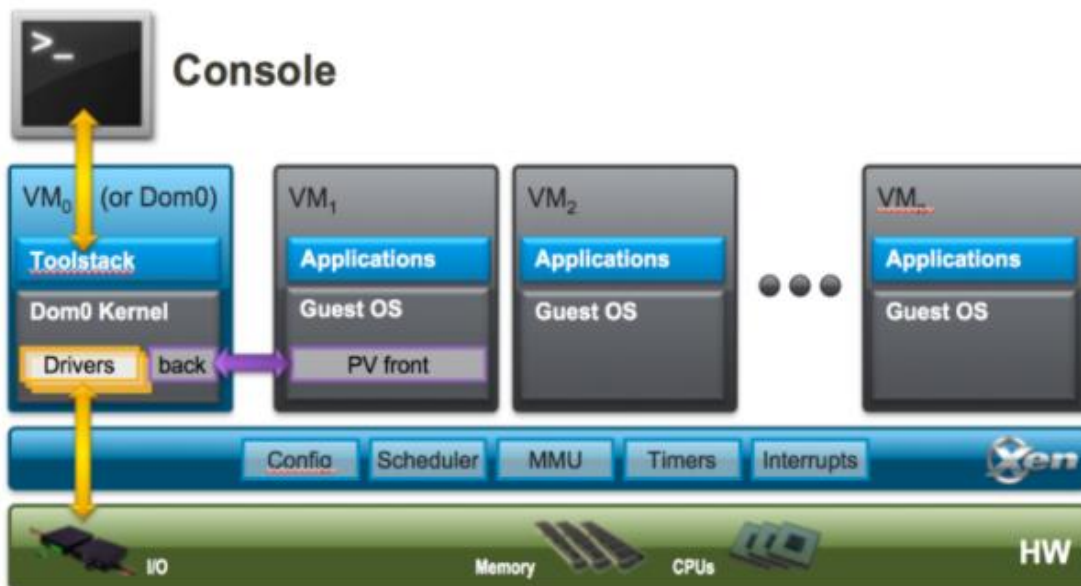


Figure 2.6: Xen Project Architecture [34]

Figure 2.6 displays the Xen Hypervisor architecture [34]. Accordingly, Xen Hypervisor runs directly on physical HW, and have control of CPU, memory, but not I/O functions. The I/O functions are handled by Domain 0 (control domain, Dom0), which is a special VM that always run on Xen host by default. Dom0 is the most important VM in Xen hypervisor virtualization system, it is the only VM that can access HW directly. Dom0 contains the toolstack to manage VMs create, destroy, and configuration. Users can access those features through the command line interface (CLI) that is included in Xen Project by default. Besides Dom0, other VMs are called guest domains, or unprivileged domain (DomU).

There are many benefits considering Xen hyervisor project over other hypervisor type-1 virtualization. Xen hypervisor is designed to run with most Linux-based OS, NetBSD, and Solaris. Xen hypervisor has device driver run inside the VM, this ensures driver isolation, if the driver crashes, that VM can reboot without affecting other VMs in the system. Moreover, Xen hypervisor is the only virtualization technique that supports fully PV guest VM, this allows PV instances to run as fast as HVM instances, in normal computational usage.

The Xen hypervisor virtualization environment in this thesis includes one physical host machine (running Ubuntu 14.04 LTS) and two virtual machines (running Ubuntu 14.04 LTS and Windows XP SP2)

Table 2.1: Virtual Machine Specifications

Host machine

RAM	16 GBytes
CPU	AMD FX8350, 4.0 GHz, 8 cores
HDD	50 GB
OS	Ubuntu 14.04 LTS x64
VM 1	
RAM	4048 MBytes
CPU	AMD FX8350, 4.0 GHz, 1 cores
HDD	12 GB
OS	Ubuntu 14.04 LTS x64
VM 2	
RAM	4048 MBytes
CPU	AMD FX8350, 4.0 GHz, 1 cores
HDD	9 GB
OS	Windows XP SP2 x64

Implementation of virtualization environment follows these sequential steps:

Step 1: Install Ubuntu 14.04 LTS x64 on the physical machine, using logical volume manager (LVM) configuration.

Step 2: Install Xen Project 4.4 on Ubuntu 14.04 LTS x64 with this command

```
$ sudo apt-get install xen-hypervisor-amd64
```

Step 3: Check if Xen hypervisor was installed successfully

```
root@xenproject-desktop: /home/xenproject
File Edit View Search Terminal Help
root@xenproject-desktop:/home/xenproject# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    1
Core(s) per socket:    8
Socket(s):              1
NUMA node(s):          1
Vendor ID:              AuthenticAMD
CPU family:             21
Model:                  2
Stepping:               0
CPU MHz:                4015.412
BogoMIPS:               8030.82
Hypervisor vendor:     Xen
Virtualization type:   none
L1d cache:              16K
L1i cache:              64K
L2 cache:                2048K
L3 cache:                8192K
NUMA node0 CPU(s):     0-7
root@xenproject-desktop:/home/xenproject#
```

Figure 2.7: List CPU Specifications.

Figure 2.7 shows Hypervisor vendor as Xen, this means Dom0 has been running in this system. Technically, Xen hypervisor will try to hide virtualization flag, to make all VMs think they are physical machine. Therefore, in the figure, virtualization type is none.

Step 4: Confirm if the physical HW supports virtualization technology:

xl command is used to interact with Xen hypervisor toolstack, thus xl command will only work with Xen hypervisor.

```
$ sudo xl dmesg
```

```
root@xenproject-desktop: /home/xenproject
File Edit View Search Terminal Help
(XEN) Initing memory sharing.
(XEN) xstate_init: using cntxt_size: 0x3c0 and states: 0x4000000000000007
(XEN) PCI: Not using MCFG for segment 0000 bus 00-ff
(XEN) AMD-Vi: IOMMU 0 Enabled.
(XEN) I/O virtualisation enabled
(XEN) - Dom0 mode: Relaxed
(XEN) Interrupt remapping enabled
(XEN) ENABLING IO-APIC IRQs
(XEN) -> Using new ACK method
(XEN) Platform timer is 14.318MHz HPET
(XEN) Allocated console ring of 16 KiB.
(XEN) HVM: ASIDs enabled.
(XEN) SVM: Supported advanced features:
(XEN) - Nested Page Tables (NPT)
(XEN) - Last Branch Record (LBR) Virtualisation
(XEN) - Next-RIP Saved on #VMEXIT
(XEN) - VMCB Clean Bits
(XEN) - DecodeAssists
(XEN) - Pause-Intercept Filter
(XEN) - TSC Rate MSR
(XEN) HVM: SVM enabled
(XEN) HVM: Hardware Assisted Paging (HAP) detected
(XEN) HVM: HAP page sizes: 4kB, 2MB, 1GB
(XEN) Brought up 8 CPUs
(XEN) mtrr: your CPUs had inconsistent fixed MTRR settings
(XEN) mtrr: your CPUs had inconsistent variable MTRR settings
(XEN) *** LOADING DOMAIN 0 ***
(XEN) Xen kernel: 64-bit, lsb, compat32
(XEN) Dom0 kernel: 64-bit, PAE, lsb, paddr 0x1000000 -> 0x2405000
(XEN) PHYSICAL MEMORY ARRANGEMENT:
(XEN) Dom0 alloc.: 0000000418000000->0000000420000000 (4041676 pages to be al
located)
(XEN) Init. ramdisk: 000000042dc21000->000000042efff10b
(XEN) VIRTUAL MEMORY ARRANGEMENT:
(XEN) Loaded kernel: ffffffff81000000->ffffffff82405000
(XEN) Init. ramdisk: ffffffff82405000->ffffffff837e310b
(XEN) Phys-Mach map: ffffffff837e4000->ffffffff85703d58
(XEN) Start info: ffffffff85704000->ffffffff857044b4
(XEN) Page tables: ffffffff85705000->ffffffff85734000
(XEN) Boot stack: ffffffff85734000->ffffffff85735000
(XEN) TOTAL: ffffffff80000000->ffffffff85800000
(XEN) ENTRY ADDRESS: ffffffff81d341f0
(XEN) Dom0 has maximum 8 VCPUs
(XEN) Scrubbing Free RAM: .done.
```

Figure 2.8: Xen Architecture and Kernel in dmesg Log File.

In figure 2.8, I/O virtualization, SVM, AMD-Vi and IOMMU are all enabled, these are all the flags that indicate if the VM can run on the system or not. At the end it also shows that Dom0 is allocate with 8 virtual CPUs from the physical HW.

Step 5: Configure the network interface

Use this command to open network interface location

```
$ sudo nano /etc/network/interfaces
```

. The interface should be set as in figure 2.9

```
interfaces (/etc/network) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Cut
interfaces x
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo eth0 xenbr0

iface lo inet loopback
#iface lo inet loopback

iface xenbr0 inet dhcp
    bridge_ports eth0
    bridge_stp on
    bridge_maxwait 0

iface eth0 inet manual|
Plain Text Tab Width: 8 Ln 12, Col 23 INS
```

Figure 2.9: Configure Network Interface in Xen Virtualization.

Step 6: Configure Ubuntu 14.04 LTS x64 VM on Xen Project environment

Create logical volume (LV) on host machine virtual group (VG) to store VM's virtual hard drive

```
$ sudo lvcreate -L 12G -n ubuntu-hvm /dev/ubuntu-vg
```

Create ubuntu guest VM config file in this directory /etc/xen/ubuntu-hvm.cfg

```
builder = "hvm"
name = "ubuntu-hvm"
memory = "4048"
vcpus = 1
vif = [ 'bridge=xenbr0' ]
#disk = [ 'phy:/dev/ubuntu-vg/ubuntu-
hvm,hda,w', 'file:/home/xenproject/ubuntu-14.04.1-desktop-
amd64.iso,hdc:cdrom,r' ]
disk = [ 'phy:/dev/ubuntu-vg/ubuntu-hvm,hda,w' ]
vnc = 1

#stdvga = 1
videoram = 128

boot="dc"
|
```

Figure 2.10: Ubuntu Virtual Machine Configuration File.

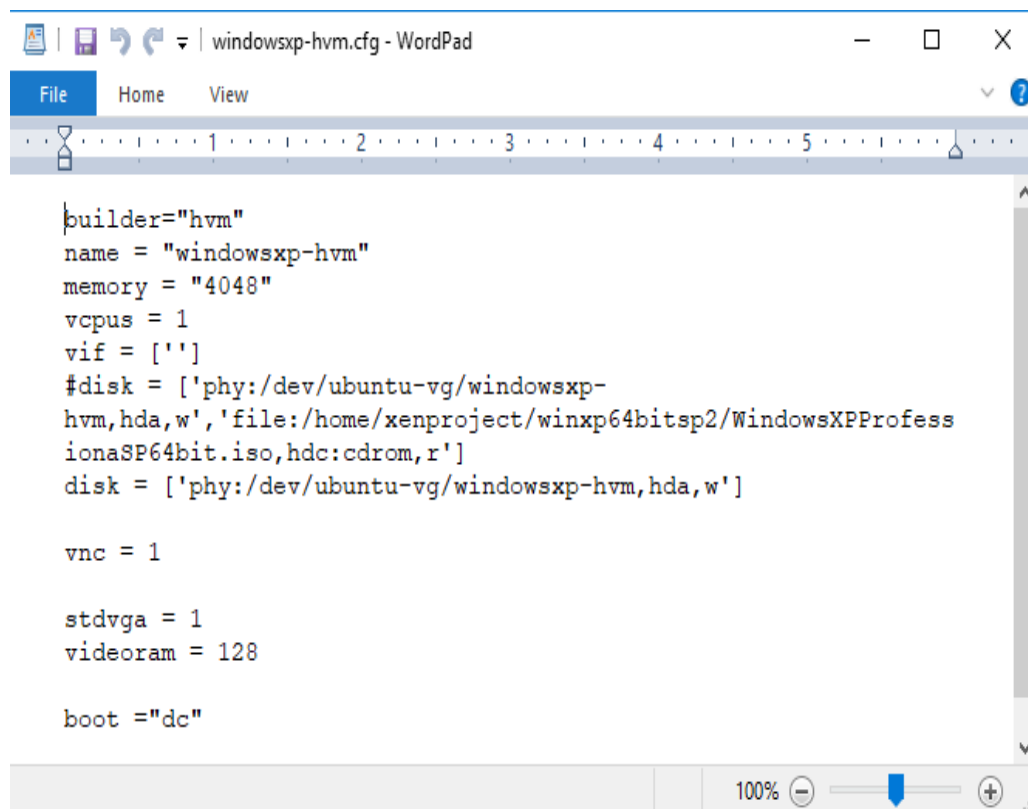
In figure 2.10, guest VM name is ubuntu-hvm, VM type is HVM, virtual memory allocate to this VM is 4048 MBytes, number of virtual cpu assign to this VM is 1. At first, before installing Ubuntu 14.04 to this VM, the boot drive path is set to /home/xenproject/ where the iso file of Ubuntu 14.04 LTS is located, after installing Ubuntu on this VM, the boot drive path is set to the logical volume directory created at the beginning of step 4: /dev/ubuntu-vg/Ubuntu-hvm.

Step 7: Configure Windows XP SP2 x64 VM on Xen Project environment.

Similar to step 6, create logical volume (LV) on host machine virtual group (VG) to store VM's virtual hard drive

```
$ sudo lvcreate -L 9G -n windowsxp-hvm /dev/ubuntu-vg
```

Create windows guest VM config file in this directory /etc/xen/windowsxp-hvm.cfg



The image shows a screenshot of a WordPad window titled "windowsxp-hvm.cfg - WordPad". The window contains the following configuration text:

```
builder="hvm"
name = "windowsxp-hvm"
memory = "4048"
vcpus = 1
vif = ['']
#disk = ['phy:/dev/ubuntu-vg/windowsxp-
hvm,hda,w', 'file:/home/xenproject/winxp64bitsp2/WindowsXPProfess
ionaSP64bit.iso,hdc:cdrom,r']
disk = ['phy:/dev/ubuntu-vg/windowsxp-hvm,hda,w']

vnc = 1

stdvga = 1
videoram = 128

boot = "dc"
```

Figure 2.11: Windows Virtual Machine Configuration File.

In figure 2.11, similar to step 6, guest VM name is windowsxp-hvm, VM type is HVM, virtual memory allocate to this VM is 4048 MBytes, number of virtual cpu assign to this VM is 1. At first, before installing Ubuntu 14.04 to this VM, the boot drive path is set to /home/xenproject/ where the iso file of Windows XP SP2 is located, after installing Ubuntu on this VM, the boot drive path is set to the logical volume directory created at the beginning of step 4: /dev/ubuntu-vg/windowsxp-hvm.

Step 8: Verify VMs' logical volume on the system

pvdisplay, vgdisplay, and lvdisplay are used to display physical volume (PV), volume groups (VG) in PV, and logical volumes (LV) in VG. This is the way LVM manage the partition.

```
--- Logical volume ---
LV Path                /dev/ubuntu-vg/ubuntu-hvm
LV Name                ubuntu-hvm
VG Name                ubuntu-vg
LV UUID                6fgdRG-mLus-g00J-0YKu-I9sC-zTSU-CjV8Vz
LV Write Access        read/write
LV Creation host, time xenproject-desktop, 2017-10-28 15:13:40 -0700
LV Status              available
# open                 0
LV Size                12.00 GiB
Current LE             3072
Segments               2
Allocation              inherit
Read ahead sectors     auto
- currently set to    256
Block device           252:2

--- Logical volume ---
LV Path                /dev/ubuntu-vg/windowsxp-hvm
LV Name                windowsxp-hvm
VG Name                ubuntu-vg
LV UUID                6d5NAX-atRe-d1iI-NkZt-2Me6-ZUSf-L6UD46
LV Write Access        read/write
LV Creation host, time xenproject-desktop, 2017-10-28 22:22:13 -0700
LV Status              available
# open                 0
LV Size                9.00 GiB
Current LE             2304
Segments               1
Allocation              inherit
Read ahead sectors     auto
- currently set to    256
Block device           252:3
```

Figure 2.12: Display VMs' Logical Volumes in the Hosted LVM Partition.

According to figure 2.12, LV ubuntu-hvm, which stores the Ubuntu 14.04 VM OS, has the size of 12 GBytes, and is located under ubuntu-vg which is the volume group of Xen Project system. Similarly, LV windowsxp-hvm, which stores the Windows XP VM OS, has the size of 9 GBytes, and is also located under ubuntu-vg.

Step 9: Create VMs on Xen hypervisor

In Xen hypervisor, use create command to create VM.

```
xenproject@xenproject-desktop: ~  
File Edit View Search Terminal Help  
xenproject@xenproject-desktop:~$ sudo xl create /etc/xen/windowsxp-hvm.cfg  
Parsing config from /etc/xen/windowsxp-hvm.cfg  
xenproject@xenproject-desktop:~$
```

```
xenproject@xenproject-desktop: ~  
File Edit View Search Terminal Help  
xenproject@xenproject-desktop:~$ sudo xl create /etc/xen/ubuntu-hvm.cfg  
[sudo] password for xenproject:  
Parsing config from /etc/xen/ubuntu-hvm.cfg  
xenproject@xenproject-desktop:~$
```

Figure 2.13: Create Ubuntu 14.04 LTS VM and Windows XP SP2 VM.

Use `xl list` to check if the VMs were created and guest domains are running together with `domain0`.

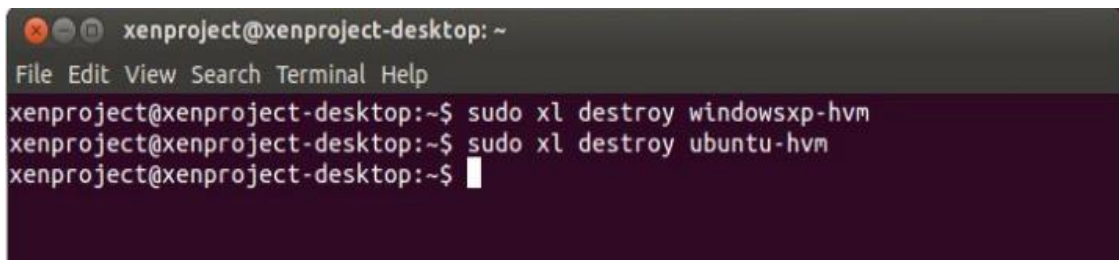
```
xenproject@xenproject-desktop: ~  
File Edit View Search Terminal Help  
xenproject@xenproject-desktop:~$ sudo xl list  
Name                ID    Mem VCPUs    State    Time(s)  
Domain-0            0    7705     8    r-----    237.8  
ubuntu-hvm          3    3928     1    -b----     24.7  
windowsxp-hvm       4    3936     1    -b----      7.2  
xenproject@xenproject-desktop:~$
```

Figure 2.14: Xen Hypervisor Domains List.

In figure 2.14, Xen hypervisor is domain 0 with 8 virtual CPUs, Ubuntu-hvm is running on domain 3 with 1 virtual CPU, and windowsxp-hvm is running on domain 4 with 1 virtual CPU.

gncviewer can be used to view the VM's desktop.

Step 10: Destroy VMs on Xen hypervisor

A terminal window with a dark background and light text. The window title is 'xenproject@xenproject-desktop: ~'. The menu bar shows 'File Edit View Search Terminal Help'. The terminal content shows three lines of commands: 'xenproject@xenproject-desktop:~\$ sudo xl destroy windowsxp-hvm', 'xenproject@xenproject-desktop:~\$ sudo xl destroy ubuntu-hvm', and 'xenproject@xenproject-desktop:~\$' followed by a cursor. The background of the terminal window is a dark purple color.

```
xenproject@xenproject-desktop: ~
File Edit View Search Terminal Help
xenproject@xenproject-desktop:~$ sudo xl destroy windowsxp-hvm
xenproject@xenproject-desktop:~$ sudo xl destroy ubuntu-hvm
xenproject@xenproject-desktop:~$
```

Figure 2.15: Destroy Ubuntu 14.04 LTS VM and Windows XP SP2 VM Command in Xen Hypervisor.

At the end, to shutdown Xen hypervisor, all VMs must be destroyed first, otherwise next time the system can't boot into GUI, this may be a technical issue on Xen Project v4.4

2.3.2 Oracle Virtualbox

VirtualBox is a free open source hypervisor type-2 virtualization software provided by Oracle, that requires VT-x or AMD-V from the HW [35].

Due to the very friendly graphic user interface (GUI), VirtualBox has become one of the most common use virtualization environment in normal daily PC usage. The VirtualBox's GUI simply provides basic feature for users such as create VM, start VM, pause VM, and stop VM (including hibernate and shutdown).

VirtualBox works with many different OSs such as Windows, Linux, MacOS, OpenSolaris, FreeBSD... In general, VirtualBox supports 3 types of virtual hard drive format: VDI which is VirtualBox standard, VMDK that is provided by VMware, and VHD from Windows Virtual PC.

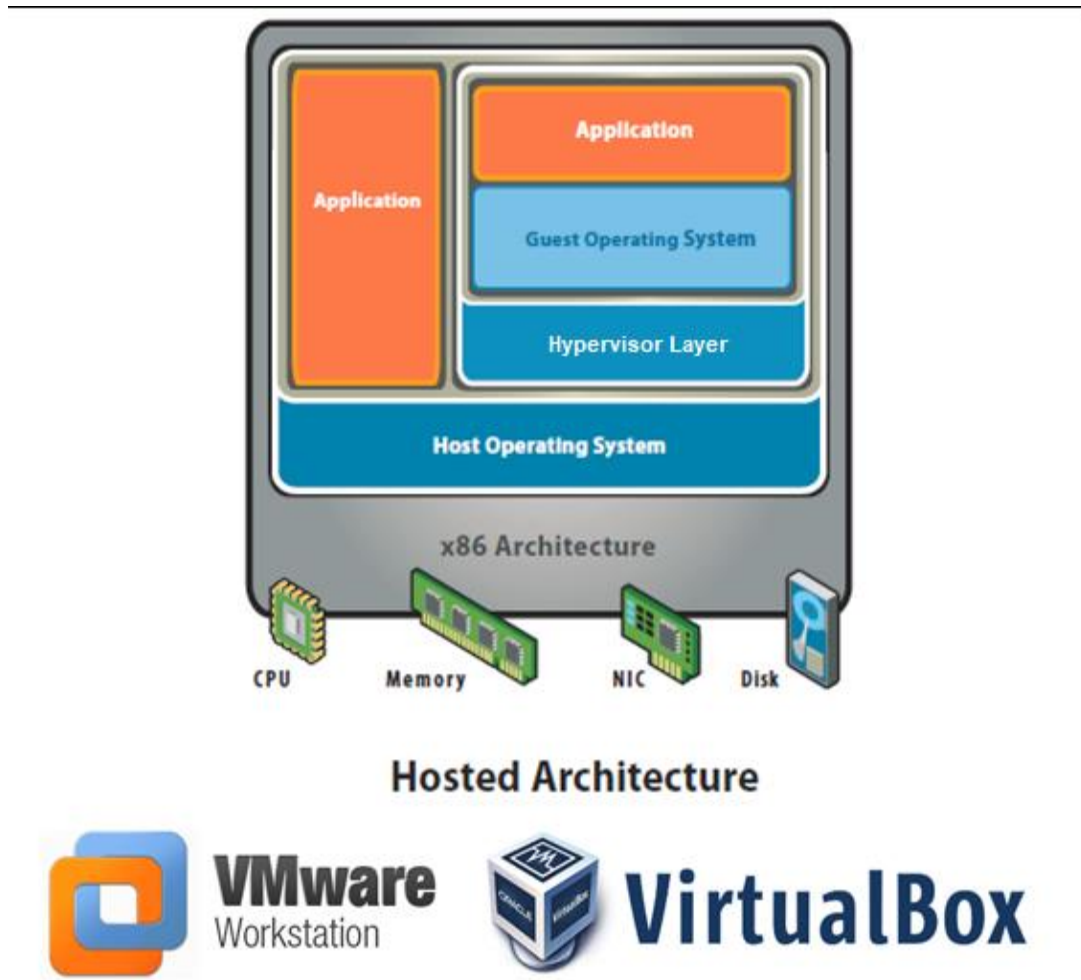


Figure 2.16: VirtualBox and VMware Architecture.

As shown in figure 2.16, VirtualBox and VMware shares the same architecture, where the hypervisor is treated as an application on host OS. Since these tools are too popular among common users, there will be no detailed discussion about the virtualization environment creating process in this thesis. The virtualization system created by VirtualBox will have the same specifications as mentioned in table 2.1.

2.4 System Events Log

In data science, the data collected during system's operation can provide valuable information about actual or potential error/failure behaviors that already occurred or may happen in the future, to verify assumptions made in analytical models.

Logging includes recording system activities and network activities and maintaining the recorded data of the system at the same time. Normally people refer to the recorded data as logging. Logging is essential to understand the activities of complicated system, especially system with less users' interaction such as server system.

Event logs is a set of recorded events, which are generated during the execution session of a system, in order to provide an audit trail that can be used in the future to understand the activity of the system and to diagnose problems if any occurs.

In computing, an event is an action or occurrence, either from normal or error/failure activity, recognized synchronously by event logging software. PC's events can be generated or triggered by the system, by users, or in the other ways.

Traditionally, computer's system event logs are standardized following syslog standard (RFC 5424, 2009) as shown in table 2.x.

Table 2.2: The Sample Set of Traditional Event Log.

...
4:47:24 PM	udisksd	LG-notice
4:47:25 PM	NetworkManager	LH-info
4:47:32 PM	NetworkManager	LG-info
4:47:40 PM	kernel	LG-warning
4:47:40 PM	kernel	LG-notice
4:47:52 PM	dbus	LG-notice
4:47:52 PM	kernel	LG-warning
4:50:03 PM	kernel	LH-debug
4:50:15 PM	IPSec	WG-Information
4:50:28 PM	DCOM	WG-Information
4:50:28 PM	EventLog	WG-Information
4:50:35 PM	pulseaudio	LG-err
...

However, this thesis will look at system logs in the other perspective, similar to the Flow-net Logging approach [36] as demonstrated in figure 2.17. The advantage of this approach is it also contains specific relationships between events, thus can be very useful trace-back diagnostic.

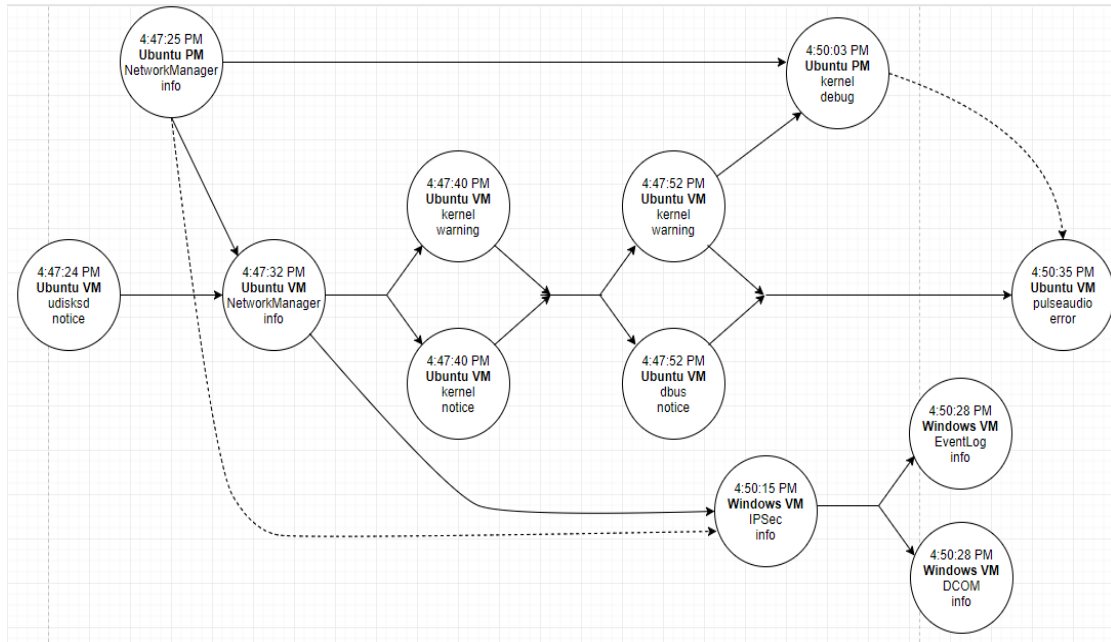


Figure 2.17: The Flow-net Logs Approach [36].

2.4.1 Windows Events Logging

Windows Event logging provides a standard, centralized way for software and system applications to record important software and hardware events. Windows application, operating system, and other system services record those important events, such as low-memory conditions or excessive attempts to access a disk. Later on if an error or failure occurs, the system administrator will use those important events to determine what caused the error, attempt to recover any lost data, and prevent the error from recurring. By periodically viewing the event log, the system administrator may be able to identify problems (such as a failing hard disk) before they cause damage [37].

In Windows OS, event logging is implemented as a system service that runs in the background and wait for triggering activity to start recording events. Windows event logging service stores recorded events in a log file called evt extension file. Windows OS has three types of log file: application log, security log, and system log. For system availability and reliability investigation purposes, this thesis will only focus on system log.

System event logs are classified into five severity levels, as explained in table 2.3 [37].

Table 2.3: Windows Event Log's Severity Levels [37].

Event type	Description
Error	An event that indicates a significant problem such as loss of data or loss of functionality. For example, if a service fails to load during startup, an Error event is logged.
Warning	An event that is not necessarily significant, but may indicate a possible future problem. For example, when disk space is low, a Warning event is logged. If an application can recover from an event without loss of functionality or data, it can generally classify the event as a Warning event.
Information	An event that describes the successful operation of an application, driver, or service. For example, when a network

	driver loads successfully, it may be appropriate to log an Information event. Note that it is generally inappropriate for a desktop application to log an event each time it starts.
Success Audit	An event that records an audited security access attempt that is successful. For example, a user's successful attempt to log on to the system is logged as a Success Audit event.
Failure Audit	An event that records an audited security access attempt that fails. For example, if a user tries to access a network drive and fails, the attempt is logged as a Failure Audit event.

Below is an event-log file structure designed by Microsoft;

```

_EVENTLOGHEADER

_EVENTLOGRECORD 1
_EVENTLOGRECORD 2
...
_EVENTLOGRECORD N
<BLANK SPACE>

```

The **_ELF_LOGFILE_HEADER** is a structure that is written at the beginning of an event log by the event-logging service, to define information about the event log.

```

typedef struct _EVENTLOGHEADER {
    ULONG HeaderSize;
    ULONG Signature;
    ULONG MajorVersion;
    ULONG MinorVersion;
    ULONG StartOffset;
    ULONG EndOffset;
    ULONG CurrentRecordNumber;
    ULONG OldestRecordNumber;
    ULONG MaxSize;
    ULONG Flags;
    ULONG Retention;
    ULONG EndHeaderSize;
} EVENTLOGHEADER, *PEVENTLOGHEADER;

```

The `_EVENTLOGRECORD` is a structure that contain an event record detailed information. One evt file can have N `_EVENTLOGRECORD`, depends on the user's predefined file size.

```

typedef struct _EVENTLOGRECORD {
    DWORD Length;
    DWORD Reserved;
    DWORD RecordNumber;
    DWORD TimeGenerated;
    DWORD TimeWritten;
    DWORD EventID;
    WORD EventType;
    WORD NumStrings;
    WORD EventCategory;
    WORD ReservedFlags;
    DWORD ClosingRecordNumber;
    DWORD StringOffset;
    DWORD UserSidLength;
    DWORD UserSidOffset;
    DWORD DataLength;
    DWORD DataOffset;
} EVENTLOGRECORD, *PEVENTLOGRECORD;

```

Details about those parameters in `_ELF_LOGFILE_HEADER` and `_EVENTLOGRECORD` are available on MSDN official website [37].

By default, Windows provides a software named Event Viewer as a tool that allows user to view and examine event logs. However, many attributes values in `_EVENTLOGRECORD` cannot be seen in Event Viewer. Besides Event Viewer, there are several other tools serving this purpose, some are quite popular such as *Wevtutil* or *LogParser*. However, after some considerations, none of them can provide complete information about one `_EVENTLOGRECORD`. Therefore, for specific investigation, a converter was built to convert evt file format to txt file format using C++ programming language. The converter is named `evt2txt.cpp`, the source code will be provided later in Appendix A of the thesis. Below is the sample event log in txt format after conversion.

HeaderSize: 48,
Signature: 1699505740,
MajorVersion: 1,
MinorVersion: 1,
StartOffset: 48,
EndOffset: 32436,
CurrentRecordNumber: 142,
OldestRecordNumber: 1,
MaxSize: 32476,
Flags: 0,
Retention: 0,
EndHeaderSize: 48,

Event Record ID: 0,
Length: 192,
Reserved: 1699505740,
RecordNumber: 1,
TimeGenerated In epoch: 1464232167,
TimeGenerated In local datetime: Wed May 25 20:09:27 2016,
TimeWritten In epoch: 1464232167,
TimeWritten In local datetime: Wed May 25 20:09:27 2016,
EventID in hexadecimal: 2147489657,
EventID:6009,
EventType: 4,
NumStrings: 4,
EventCategory: 0,
ReservedFlags: 0,
ClosingRecordNumber: 0,
StringOffset: 98,
UserSidLength: 0,
UserSidOffset: 98,
DataLength: 0,
DataOffset: 186,
Source name: EventLog,
Computer name: MACHINENAME,
Keyword:Classic,
User:N/A,
UserName:N/A,
Event Data (String):
5.02.,
3790,
Service Pack 2,
Uniprocessor Free,
Data: 00E93020,
Length: 192,
Description: Microsoft (R) Windows (R) 5.02. 3790 Service Pack 2
Uniprocessor Free.,

```

Event Record ID: 1,
  Length: 128,
  Reserved: 1699505740,
  RecordNumber: 2,
TimeGenerated In epoch: 1464232167,
  TimeGenerated In local datetime: Wed May 25 20:09:27 2016,
  TimeWritten In epoch: 1464232167,
  TimeWritten In local datetime: Wed May 25 20:09:27 2016,
  EventID in hexadecimal: 2147489653,
    EventID:6005,
  EventType: 4,
  NumStrings: 7,
  EventCategory: 0,
  ReservedFlags: 0,
  ClosingRecordNumber: 0,
  StringOffset: 98,
  UserSidLength: 0,
  UserSidOffset: 98,
  DataLength: 0,
  DataOffset: 120,
  Source name: EventLog,
  Computer name: MACHINENAME,
  Keyword:Classic,
  User:N/A,
  UserName:N/A,
  Event Data (String):
,
,
,
,
,
8,
0,
0 ,
Data: 00E930B0,
  Length: 128,
  Description: The Event log service was started.,

```

According to the information provided in this log file, this file contains 142 recorded events, with the total size of 32.476 Bytes. The first recorded event was written at Wed May 25 20:09:27 2016, by the EventLog service, and the EventType value is 4, this means the severity level of this event is 'information', as refer to table 2.4.

Table 2.4: EventType Value Reference Table from MSDN [37].

Value	Description
0	Success
1	Error
2	Warning
4	Information
8	Audit Success
16	Audit Failure

2.4.2 Linux System Logging

Linux OS has syslogd (or syslog-ng) as the syslog daemons to record system event log. Syslogd includes two system applications: klogd and syslogd. Klogd manages the all event logging process from kernel, while syslogd manages event logging process from applications. Besides, there are other third parties application that can produce their own logs. Event logging in Linux are recorded in the time generated, and follow the syslog standard, thus it lacks the ability to store relationship between events.

In Linux event logging process, the triggered event information is first kept in buffer, then the event logging daemon will write the event to the log file periodically. Accordingly, the timestamp of the event doesn't necessarily mean the actual time the event happened, it is the time when an event is generated and written.

Table 2.5: Severity Level in Ubuntu 14.04 LTS x64 [39].

Severity	Keyword	Description
Emergency	<code>emerg</code>	System is unusable. A panic condition.
Alert	<code>alert</code>	Action must be taken immediately. A condition that should be corrected immediately, such as a corrupted system database.
Critical	<code>crit</code>	Critical conditions, such as hard device errors.
Error	<code>err</code>	Error conditions.
Warning	<code>warning</code>	Warning conditions.
Notice	<code>notice</code>	Normal but significant conditions. Conditions that are not error conditions, but that may require special handling.

Informational	info	Informational messages.
Debug	debug	Debug-level messages. Messages that contain information normally of use only when debugging a program

In Ubuntu 14.04 LTS, an event's structure is vary depends on user's preferable format in /etc/rsyslog.conf file. Below is the sample modified-template and output.

```
$template strtpl,"PRI: %pri-text%,\nSeverity: %syslogseverity-text%,\nFacility: %syslogfacility-text%,\nTimeGenerated: %timegenerated%,\nTimeReported: %timereported%,\nHostname: %HOSTNAME%,\nProgramName: %programname%,\nFromHost: %fromhost%,\nTAG: %syslogtag%,\nInfoUnitType: %iut%,\nMsg: %msg%,\nRawMsg: %rawmsg%\nProtocolVersion: %protocol-version%,\nStructuredData: %structured-data%,\nAppName: %app-name%,\nProcid: %procid%,\nMsgid: %msgid%,\nInputname: %inputname%\n\n"
```

```
PRI: daemon.info,  
Severity: info,  
Facility: daemon,  
TimeGenerated: May 26 03:43:40,  
TimeReported: May 26 03:43:40,  
Hostname: UbuntuHost,  
ProgramName: NetworkManager,  
FromHost: UbuntuHost,  
TAG: NetworkManager[698]:,  
InfoUnitType: 1,  
Msg: <info> Activation (eth0) Stage 3 of 5 (IP Configure Start)  
started...,  
RawMsg: <30>May 26 03:43:40 NetworkManager[698]: <info> Activation  
(eth0) Stage 3 of 5 (IP Configure Start) started...  
ProtocolVersion: 0,  
StructuredData: -,  
AppName: NetworkManager,  
Procid: 698,  
Msgid: -,  
Inputname: imuxsock
```

The parameters included in the event above are quite similar to the parameters in a Windows event, therefore it is not necessary to go any deeper here.

2.5 Common Users' Behavior Simulation

This thesis considers common users of a mid-range cloud renting PC regarding to a normal PC user whose daily tasks on the computer are as simple as using MS Office, drawing, internet surfing, listening to music, watching movie, and other simple calculating. The reason behind this consideration is, with such renting cloud PC specs, the user can't do any special overload computation, or dedicated gaming or visual art designing. Apparently, there is a potential for

further experiment, to test overload computing on cloud machine with higher specs, this will be mentioned later in Chapter 4.

A use case diagram is designed as displayed in figure 2.18, to specify all task that may be run on a cloud VM. From this use case diagram, an auto tool named AutoUser.cpp was built using C++ programming language, this tool will automatically perform all the task a user can do in the order of time, one by one, the second task will be performed 5 seconds after the first task, etc. When all the tasks are running, the tool will set a 30 seconds timing, after 30 seconds, it will start killing all the tasks one by one. When finish killing all the tasks, the tool will wait for 10 seconds before repeating all the process all over again. The tool will do these repeatedly infinite times, until getting a terminating command from keyboard.

For better understanding, the user activities simulation process from the beginning until the end is described in an activity diagram as shown in figure 2.19.



Figure 2.18: Common User's Use Case Diagram.

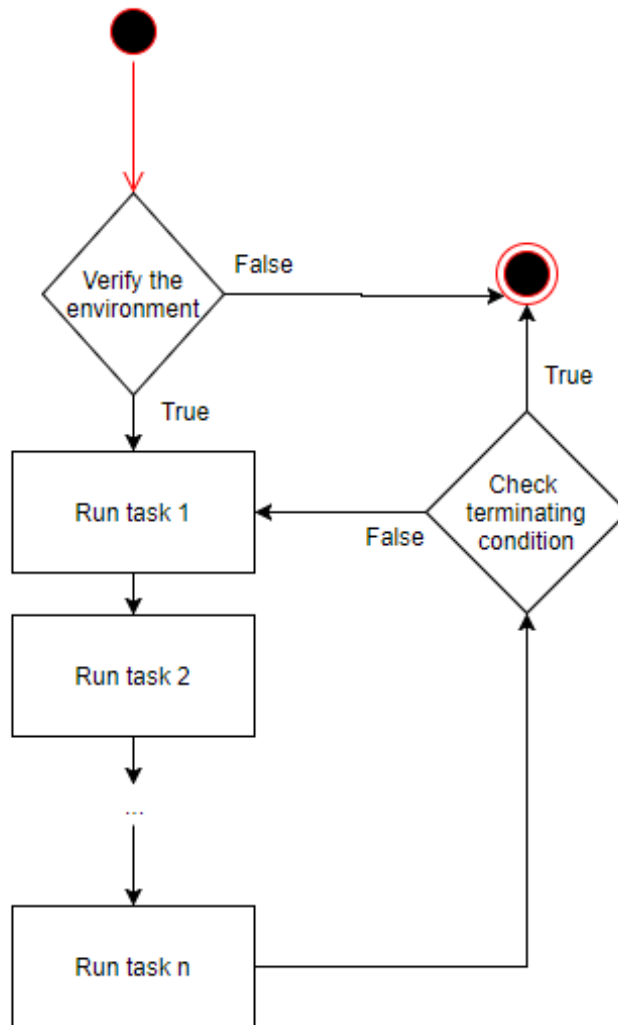


Figure 2.19: Auto User Simulation Activity Diagram.

2.6 Data Collection Strategy

Data collection is performed once every third days of the three days sessions and, consists of these main steps:

- The virtualization hypervisor type-1 and type-2 environments were deployed sequentially every 3 to 6 days, the specifications of each machine in the environment were as described in Chapter 2.3;
- All the VMs in the virtualization environment were set to run AutoUser.cpp since day one to day 3;
- All the machines in the virtualization environment have the event-log format template configure as described above in Chapter 2.4;
- Backup System event logs of each machines (PMs and VMs) to a dedicated machine used for data preprocessing and network training. The other extra back up files are stored on Google Cloud to prevent data loss from hardware failure.

The back-up event logs collected from each machine, which was generated during the same running period, are concatenated into a single file in the order of time. Each event in this concatenating process only contains useful attributes values. All these values at the end will be quantify into numbers to make them understandable with neural network.

The data collected using this strategy corresponds to a one month experimenting period (from December 2017 to January 2018). At the end there were five distinct concatenated sets, three from VirtualBox environments during three discrete performing time, and two from Xen Hypervisor environments during two discrete performing time.

2.7 Chapter Summary

This chapter provided an overview about virtualization in IaaS cloud, and described the process to simulate specific environments with detailed information. It also explained about target user expectation and data collection process. Next chapter will discuss about neural network time series, as well as how those data will be used in the network.

3. NEURAL NETWORK TIME SERIES MODEL AND SYSTEM FAILURE PATTERN DETECTION

3.1 Chapter Introduction

Neural network time series model is a new application of neural network in detecting and predicting sequential events. Similar to episode mining introduced in Chapter 1, given a set of event sequence and the length of maximum support threshold, neural network time series model can detect all frequent episodes. But, more than what episode mining can do, neural network time series model can also learn the rules of how a frequent episode occurs, and predict which event is going to happen following those rules. This gives motivation to find out answer for the question: “Whether neural network time series model can predict failure events that may occur in cloud environment before it actually happens ?”.

3.2 NARX Network Model

In MATLAB, neural network time series model is classified into three different types:

- Nonlinear Autoregressive (NAR net) - Predict series $y(t)$ given n_y past values of $y(t)$. The function for NAR net is defined as:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y))$$

- Nonlinear Input-Output - Predict series $y(t)$ from $x(t)$ past values input, given n_x past values of $x(t)$.

$$y(t) = f(x(t - 1), x(t - 2), \dots, x(t - n_x))$$

- Nonlinear Autoregressive with External Input (NARX net) – Predict series $y(t)$ given n_y past values and another series $x(t)$ with n_x past values (n_x can be equal to n_y). The function for NARX net is defined as:

$$y(t) = f\left(y(t - 1), y(t - 2), \dots, y(t - n_y), x(t - 1), x(t - 2), \dots, x(t - n_x)\right) \quad (4)$$

Among those three, NARX net solution has more input vectors, thus, obviously it can generate the most accurate results. Therefore, the thesis will use NARX net model for further experiments.

Technically, NARX net is a type of feed forward neural network that takes a series of event as the input. The length of the series is defined as Input-Delay (ID) which is referred to n_x value in equation (4), and Feedback-Delay (FD) which is referred to n_y value in equation (4). For example, the NARX net with ID = 15 and FD = 30, when predicting the next value y_t , will take the input series including 15 last $x(t)$ (from x_{t-15} to x_{t-1}) and 30 last $y(t)$ (from y_{t-30} to y_{t-1}).

Accordingly, choosing the right value for ID and FD is very important to help improve the NARX net outcome accuracy. Using the cross-correlation and auto-correlation are among the techniques that are widely used for this purpose.

Cross-correlation is used between two discrete time series, the higher cross-correlation value means the more similarities between the two time series, this method is used to find out the best ID value between $x(t)$ and $y(t)$.

$$xcorr(X, Y, lag) = \sum_{n=0}^{n-1} X[n] \times Y[n] \quad (17)$$

In actual experiments, after several tries, shifting $y(t)$ to the right side of $x(t)$ 30 columns ($lag = 30$) gives the most significant $xcorr(X, Y, 30) = 4.38$.

Therefore, ID value is chosen as 30.

Calculating auto-correlation function is the same as calculating cross-correlation function, the only difference here is, auto-correlation is used between one time series and itself with a lag value time shift.

$$autocorr(Y, lag) = \sum_{n=0}^{n-1} Y[n] \times Y[n + lag] \quad (18)$$

In actual experiments, $autocorr(Y, 30) = 12.75$ is the most significant auto-correlation value. Therefore, FD value is also chosen at 30.

Table 3.1: NARX Net Input Data.

ProgramName	SeverityLevel
...	...
udisksd	LG-notice
NetworkManager	LH-info
NetworkManager	LG-info
kernel	LG-warning

kernel	LG-notice
dbus	LG-notice
kernel	LG-warning
kernel	LH-debug
IPSec	WG-Information
DCOM	WG-Information
EventLog	WG-Information
pulseaudio	LG-err
...	...

In the NARX net model used in this thesis experiment, the number of hidden neuron in hidden layer is defined as

$$n_{Hidden} = \left\lceil \frac{n_{Input} + n_{Output}}{n_{InputVector}} \right\rceil \quad (4)$$

As shown in table 3.1, the NARX net input data will have 2 vectors Program-Name and Severity-Level, so $n_{InputVector}$ will be equal to 2. FD and ID are both equal to 30, therefore the NARX net will take 30 last values of Program-Name vector and 30 last values of Severity-Level vector, this means n_{Input} is equal to $30 \times 2 = 60$. And the network output only return one $y(t)$ value, this means $n_{Output} = 1$. At the end $n_{Hidden} = \text{upper-bound}((60+1)/2) = 31$.

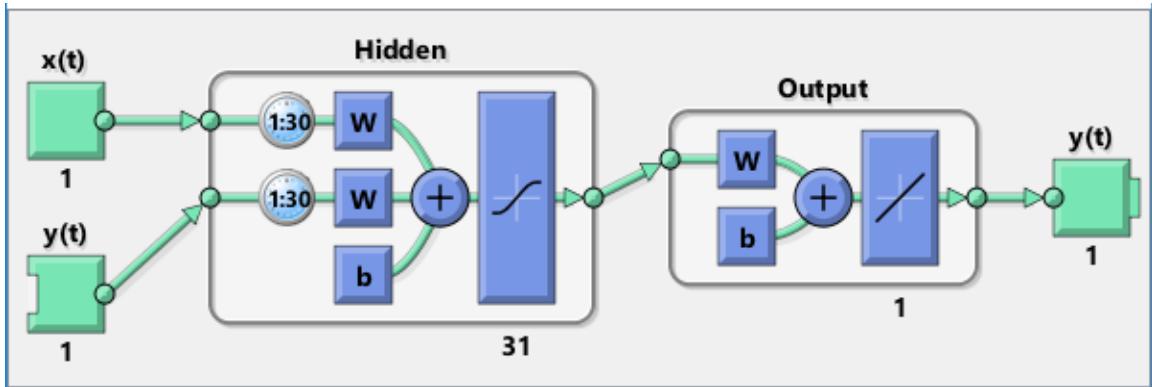


Figure 3.1: NARX Net Model Deployed in this Thesis Experiments..

In some special case, ID can be 0 which can be described as $y(t) = f(y(t - 1), y(t - 2), \dots, y(t - n_y), xt, x(t - 1), x(t - 2), \dots, x(t - n_x))$ by adding xt , or can be visualized as shown in table 3.2. Setting ID = 0 can improve the prediction output Y significantly, but in this situation, ProgramName and SeverityLevel of one event are generated at the same time, and are written to the system log file at the same time, therefore this approach cannot be used.

Table 3.2: Visualization of NARX Net Taking Input Data with Input-Delay Equal to '0'.

ProgramName	SeverityLevel
...	...
udisksd	LG-notice
NetworkManager	LH-info
NetworkManager	LG-info
kernel	LG-warning

kernel	LG-notice
dbus	LG-notice
kernel	LG-warning
kernel	LH-debug
IPSec	WG-Information
DCOM	WG-Information
EventLog	WG-Information
pulseaudio	?

3.3 Data Preparation

Because neural network only take quantified input value, therefore every value in raw data in table 3.1 has to be mapped with specific reference tables as shown in table 3.3 and 3.4.

Table 3.3: Severity Level Reference Table.

LH-Emerg	1
LH-Alert	2
LH-Crit	3
LH-Err	4
LH-Warning	5
LH-Notice	6
LH-Info	7
LH-Debug	8
LG-Emerg	9
LG-Alert	10
LG-Crit	11
LG-Err	12
LG-Warning	13

LG-Notice	14
LG-Info	15
LG-Debug	16
WG-Error	17
WG-Warning	18
WG-Information	19

Table 3.4: Program Name Reference Table.

winOS	LinuxOS Guest	LinuxOS Host	mapping number
		anacron	0
		NetworkManager	1
		kernel	2
		bluetoothd	3
		gnome-session	4
		colord	5
		acpid	6
		rtkit-daemon	7
		pulseaudio	8
		dbus	9
		whoopsie	10
		rsyslogd	11
		nvidia-persistenced	12
		avahi-daemon	13
		polkitd	14
		wpa_supplicant	15
		cron	16
		ntpdate	17
		restorecond	18
		accounts-daemon	19
		ModemManager	20
		udisksd	21
		dhclient	22
		dnsmasq	23
		ntfs-3g	24
		rsyslogd-2039	25
		mtp-probe	26

	polkitd	27
	failsafe	28
	restorecond	29
	AptDaemon	84
	AptDaemon.PackageKit	85
	AptDaemon.Worker	86
	AptDaemon.Trans	87
	dnsmasq-dhcp	88
	irqbalance	89
	xenstored	90
	xenproject	91
	logger	92
	cracklib	93
	avahi-autoipd(xenbr0)	101
	anacron	30
	NetworkManager	31
	kernel	32
	bluetoothd	33
	gnome-session	34
	colord	35
	acpid	36
	rtkit-daemon	37
	pulseaudio	38
	dbus	39
	whoopsie	40
	rsyslogd	41
	nvidia-persistenced	42
	avahi-daemon	43
	polkitd	44
	wpa_supplicant	45
	cron	46
	ntpdate	47
	restorecond	48
	accounts-daemon	49
	ModemManager	50
	udisksd	51
	dhclient	52
	dnsmasq	53
	ntfs-3g	54

	rsyslogd-2039		55
	mtp-probe		56
	polkitd		57
	failsafe		58
	restorecond		59
	AptDaemon		94
	AptDaemon.PackageKit		95
	AptDaemon.Worker		96
	AptDaemon.Trans		97
	dnsmasq-dhcp		98
	irqbalance		99
	cracklib		100
EventLog			60
DCOM			61
PlugPlayManager			62
IPSec			63
Workstation			64
NtServicePack			65
Setup			66
USER32			67
Dhcp			68
Tcpip			69
AeLookupSvc			70
W32Time			71
Service Control Manager			72
WinHttpAutoProxySvc			73
AeLookupSvc			74
Windows Update Agent			75
SideBySide			76
HTTP			77
Print			78
Application Popup			79
Win32k			80
WindowsMedia			81
SRService			82
sr			83

After completely quantified, the input data will be divided into three subsets:

- Training set (70%) – Neural network analyzes this set, detects rules and repeating pattern, these will be learnt by adjusting weight between connections of input neurons and hidden neurons.
- Validating set (15%) – This set is used to validate the pre-trained neural network after each epoch (or iteration). If the network performance is not close to the goal, the network training process will continue the next epoch. Neural network completes training process only when the neural network performance can reach a certain goal, or the training process has reached maximum epochs limit or training time limit.
- Testing set (15%) – The use of this set is similar to validating set, but this set will only be used once, the output of this set is the final outcome of the neural network in a neural network training strategy.

3.4 Levenberg-Marquardt Backpropagation Training

Back propagation is a method that uses output results to adjust input parameters' values, in order to achieve better results in the future. This plays an important role in neural network training, after training and validating process, backpropagation technique is used to update weight values on each edges connect between neurons.

Let $T = \{T_1, T_2 \dots T_n\}$ be the set of target, $Y = \{y_1, y_2 \dots y_n\}$ be the set of output, T and Y from one network training solution must have the same size. The set of errors $E = \{E_1, E_2 \dots E_n\}$ is defined as

$$E = T - Y \quad (5)$$

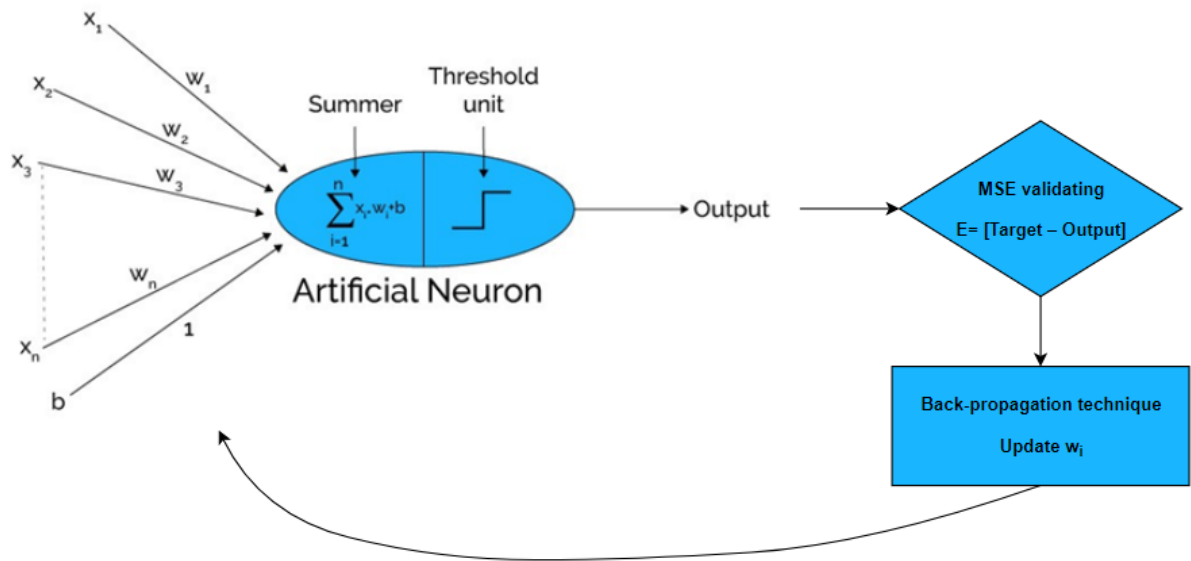


Figure 3.2: Weight Adjusting in One Epoch in NN Training Process.

In neural network, one epoch is equivalent to one training iteration. After one epoch, new output $Y = \{y_1, y_2 \dots y_n\}$ is generated. From equation (1) and (2)

$$a_{i-1} = y_i = \frac{1}{1 + e^{-y_i}} = \frac{1}{1 + e^{-(\sum_{i=1}^n x_i \cdot w_i + b_i)}} = \frac{e^{\sum_{i=1}^n x_i \cdot w_i + b_i}}{1 + e^{\sum_{i=1}^n x_i \cdot w_i + b_i}} \quad (6)$$

Let $W = \{w_1, w_2 \dots w_n\}$ be the set of weights, given total mean squared error MSE and neuron's input value a as described in equation (1) and equation (3) in Chapter 1, respectively, applying the chain rules in partial derivatives, the gradient between W and MSE, or how much changing a value in W can affect MSE, is defined as

$$\frac{\partial \text{MSE}}{\partial w_i} = \frac{\partial \text{MSE}}{\partial y_i} * \frac{\partial y_i}{\partial a_{i-1}} * \frac{\partial a_{i-1}}{\partial w_i} \quad (7)$$

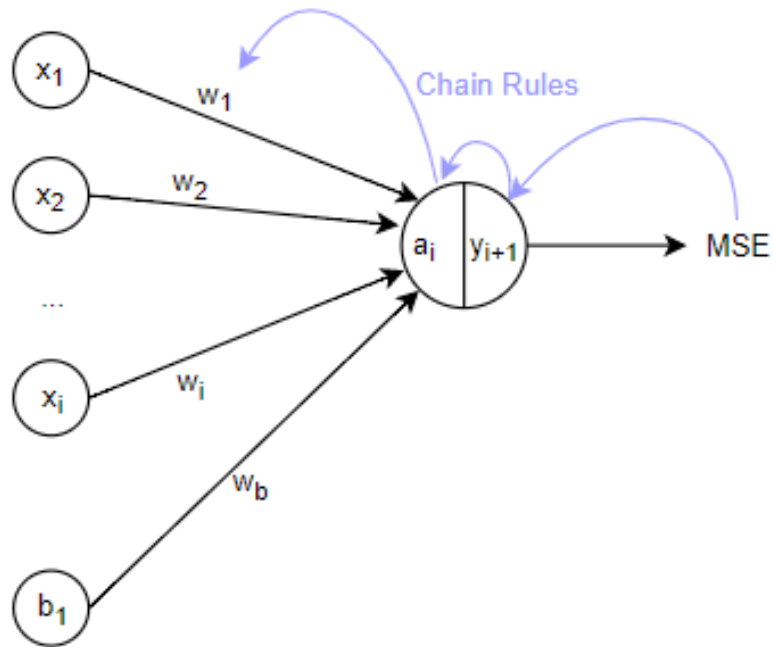


Figure 3.3: Visualize the Chain Rules in Finding Gradient between MSE and Weight.

From equation (6), apply the derivative of the logistic function

$$\frac{\partial y_i}{\partial a_i} = y_i(1 - y_i) = \frac{1}{1 + e^{-(\sum_{i=1}^n x_i w_i + b_i)}} \left(1 - \frac{1}{1 + e^{-(\sum_{i=1}^n x_i w_i + b_i)}} \right) \quad (8)$$

From equation (3), $MSE = \frac{1}{n} [(T_i - y_i)^2 + (T_{i-1} - y_{i-1})^2 + \dots + (T_1 - y_1)^2]$, consider partial derivative of MSE with respect to y_i , all the values from y_{i-1} to y_1 does not affect it, therefore $(T_{i-1} - y_{i-1})^2 + \dots + (T_1 - y_1)^2$ becomes 0. Let y_i be a constant in this partial derivative equation, using the rules for ordinary differentiation:

$$\frac{\partial MSE}{\partial y_i} = 2 * \frac{(T_i - y_i)^{(2-1)}}{n} = \frac{2(T_i - y_i)}{n} \quad (9)$$

Similarly, from equation (1), $a_{i-1} = w_i y_i + w_{i-1} y_{i-1} + \dots + w_1 y_1 + b w_b$, consider partial derivative of a_{i-1} with respect to w_i , all the values from w_{i-1} to w_1 does not affect it, therefore $w_{i-1} y_{i-1} + \dots + w_1 y_1 + b w_b$ becomes 0. Let w_i be a constant in this partial derivative equation, using the rules for ordinary differentiation:

$$\frac{\partial a_{i-1}}{\partial w_i} = 1 * w_i y_i^{(1-1)} = w_i \quad (10)$$

Replace equation (8), (9), (10) to equation (7):

$$\begin{aligned}
\frac{\partial MSE}{\partial w_i} &= \frac{\partial MSE}{\partial y_i} * \frac{\partial y_i}{\partial a_{i-1}} * \frac{\partial a_{i-1}}{\partial w_i} \\
&= \frac{2(T_i - y_i)}{n} * \frac{1}{1 + e^{-(\sum_{i=1}^n x_i \cdot w_i + b_i)}} \left(1 - \frac{1}{1 + e^{-(\sum_{i=1}^n x_i \cdot w_i + b_i)}} \right) * w_i \\
&= \frac{2w_i(T_i - y_i)}{n} * \frac{1}{1 + e^{-(\sum_{i=1}^n x_i \cdot w_i + b_i)}} \left(1 - \frac{1}{1 + e^{-(\sum_{i=1}^n x_i \cdot w_i + b_i)}} \right) \quad (11)
\end{aligned}$$

Having the steep from equation (11), let η be the learning parameter, to decrease the error in next training iteration from $w_{\text{inextEpoch}}$, this steep value will be subtracted from current weight w_i , the back propagation algorithm is defined as

$$w_{\text{inextEpoch}} = w_i - \eta * \frac{\partial MSE}{\partial w_i} \quad (12)$$

Levenberg-Marquardt (LM) algorithm is an update version of Gauss-Newton algorithm, both can only be used when the network performance is calculated in mean or sum of squared errors. Gauss Newton algorithm is a commonly used method to find nonlinear least-squared errors curve fitting line. Consider J as a Jacobian matrix between error set $E = \{E_1, E_2 \dots E_n\}$ and weight set $W = \{w_1, w_2 \dots w_m\}$, with $m \geq n$, matrix J is defined as

$$J = \begin{bmatrix} \frac{\partial E_1}{\partial w_1} & \dots & \frac{\partial E_1}{\partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial E_n}{\partial w_1} & \dots & \frac{\partial E_n}{\partial w_m} \end{bmatrix} \quad (13)$$

$J^T J$ will become the Hessian matrix and $J^T E$ will become the gradient, this will replace the gradient in equation (12), and the weight adjusted for next epoch $w_{\text{nextEpoch}}$ will be calculated by Gauss-Newton equation below

$$w_{\text{nextEpoch}} = w_i - (J^T J)^{-1} J^T E \quad (14)$$

Levenberg-Marquardt backpropagation improve Gauss Newton algorithm by adding μI to equation (14) to ensure the matrix is invertible. Matrix I is an

Identity matrix: $\begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$, parameter μ will decrease after every successful

epoch that makes network performance value goes down, and will increase if the next training epoch may potentially make network performance goes up.

$$w_{\text{nextEpoch}} = w_i - (J^T J + \mu I)^{-1} J^T E \quad (15)$$

When parameter $\mu = 0$, LM backpropagation becomes a Gauss Newton backpropagation.

3.5 Training Validation

Since the purpose of the thesis is to predict failure events, after NARX net has generated the output matrix, all the non-failure events predicted should be neglected to preserve the accuracy of the experiment. Let $F = \{\text{WG-failure, LG-failure, LH-failure}\}$ be the set of failure events, the target set of failure events $T' = \{T'_1, T'_2 \dots T'_m\}$, with $m < n$, is generated by selecting all the elements from T that has SeverityLevel value listed in F ; $Y' = \{y'_1, y'_2 \dots y'_m\}$ is the set of output corresponding to T' . The new network performance valuation equation MSE' is defined as:

$$MSE' = \frac{1}{m} \sum_{i=1}^m (y'_i - T'_i)^2 \quad (16)$$

The data collection using in this experiments was accumulated during December 2017, including 5 datasets, 3 from VirtualBox environment and 2 from Xen Project environment, as shown in table 3.5. Two NARX net had been trained with a dataset in each environment, particularly, the NARX net 1 was trained with VirtualBox 2nd dataset [X2,T2], and the NARX net 2 was trained with Xen Project 1st dataset [X4,T4].

Table 3.5: Data Collection during December 2017.

		Number of events included	Dataset	
VirtualBox	1 st	19,745	[X1, T1]	
	2 nd	38,938	[X2, T2]	Trained with NARX net 1
	3 rd	3,725	[X3, T3]	
Xen Hypervisor	1 st	335,933	[X4, T4]	Trained with NARX net 2
	2 nd	209,943	[X5, T5]	

After being trained with dataset [X2, T2] and [X4, T4], NARX net 1 and NARX net 2 have learnt failure pattern in System Events Log and can predict the next coming failure event. They are then fed other input dataset such as [X1, T1], [X3, T3], and [X5, T5] to generate Y1, Y3 and Y5. Table 3.6 validate the accuracy in each output set Y generated in each NARX net model.

Table 3.6: Training Validation with each Dataset Using MSE'.

MSE'				
[X1, T1]	[X2, T2]	[X3, T3]	[X4, T4]	[X5, T5]

NARX net 1	0.0682	0.0117	0.0973	0.0377	0.0483
NARX net 2	0.1196	0.0196	0.3356	0.000215	0.0451

Obviously, the NARX net will always give better prediction accuracy with the set they were trained with. In the first situation, NARX net 1 was trained with [X2, T2] and 0.0117 is the lowest MSE' it can perform. In the second situation, NARX net 2 was trained with [X4, T4] and 0.000215 is also the lowest MSE' it can perform.

Consider MSE' = 0.1 as 90% accuracy in failure event predicting, according to table 3.6, the NARX net 1 can always give prediction with the accuracy higher than 90%, not only in Virtual Box environment, but also in Xen Hypervisor environment, this makes it a better choice when using in an undefined virtualization environment. However, the NARX net 2 can give higher accuracy when it comes to Xen hypervisor environment (over 99.989% is almost a perfect prediction strategy).

Besides MSE', plotRegression() and plotResponse() are some good tools for training validation. Plot Regression shows the relationship between the outputs of the network and the targets. If the training were perfect, the network outputs and the targets would be exactly equal, but the relationship is rarely perfect in practice. In figure 3.4, the dashed line in each plot represents the perfect result – outputs = targets, the solid line represents the best fit linear

regression line between outputs and targets, the R value is an indication of the relationship between the outputs and targets. $R = 1$ indicates that there is an exact linear relationship between outputs and targets. If R is close to zero, then there is no linear relationship between outputs and targets.

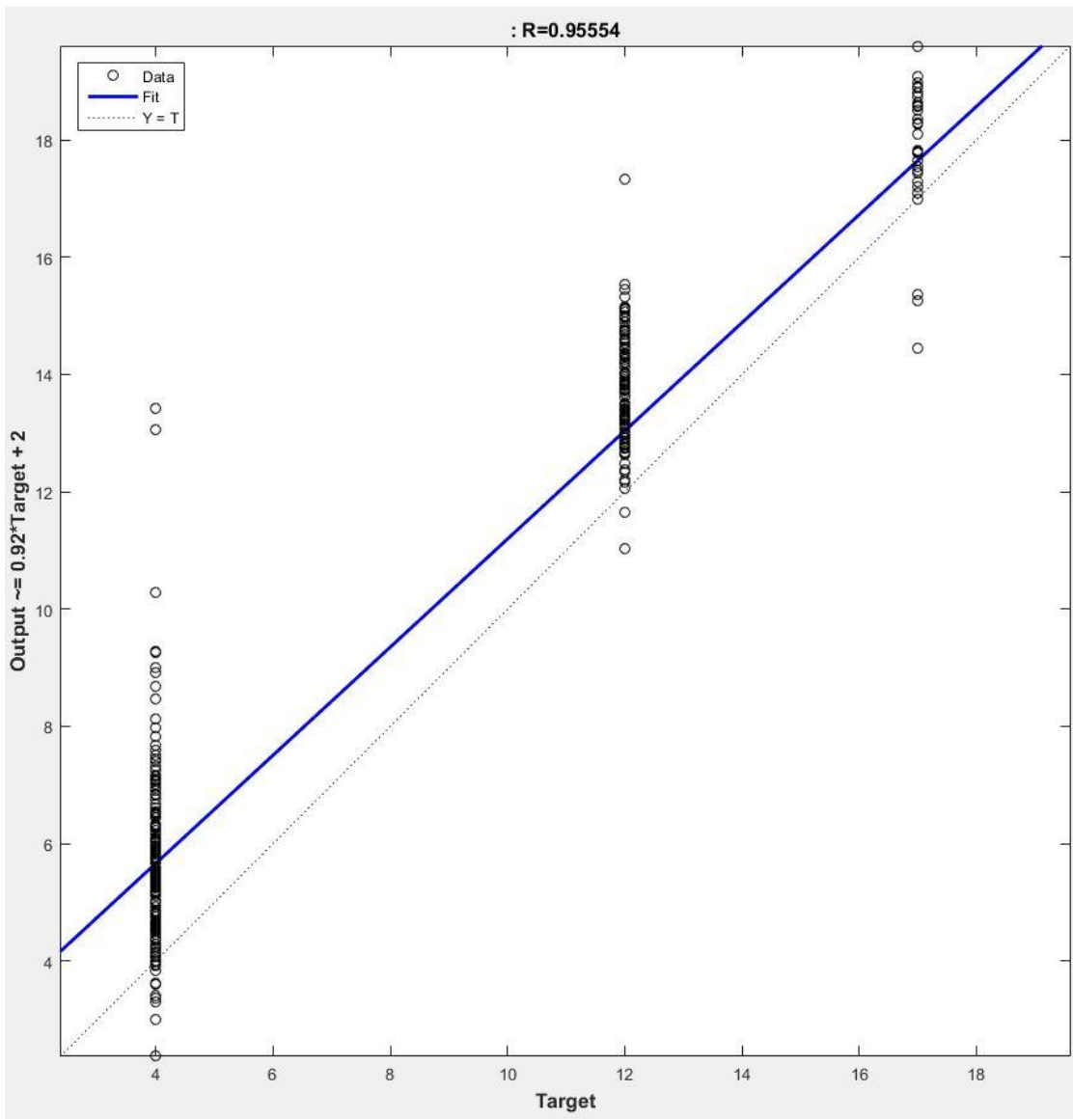


Figure 3.4: Plot Regression of NARX Net 1 When Training with $[X_2, T_2]$.

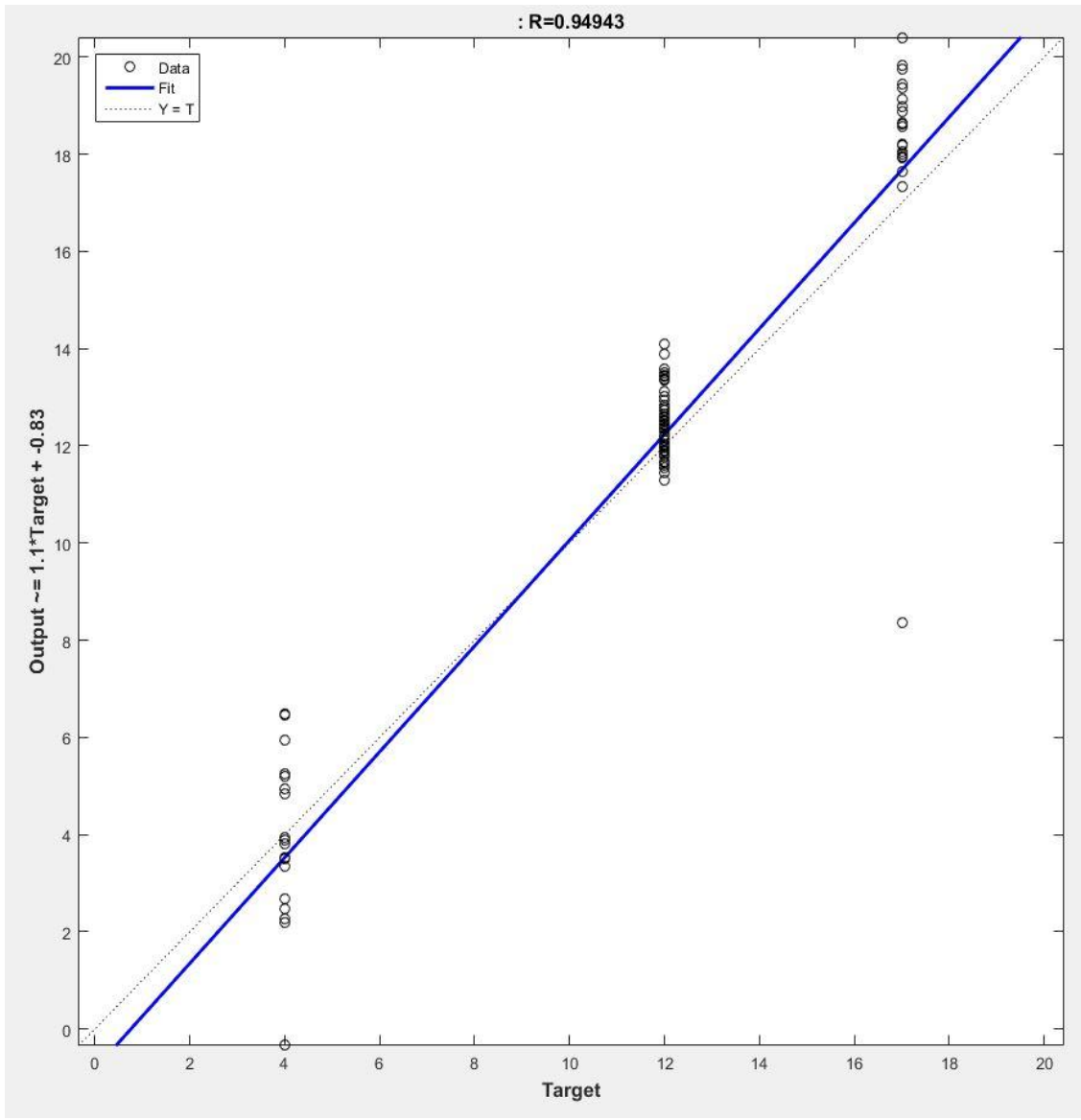


Figure 3.5: Plot Regression of NARX Net 2 When Training with [X4, T4].

Plot Response plot the target set Ts and output set Y on the same axis, to show the errors between them.

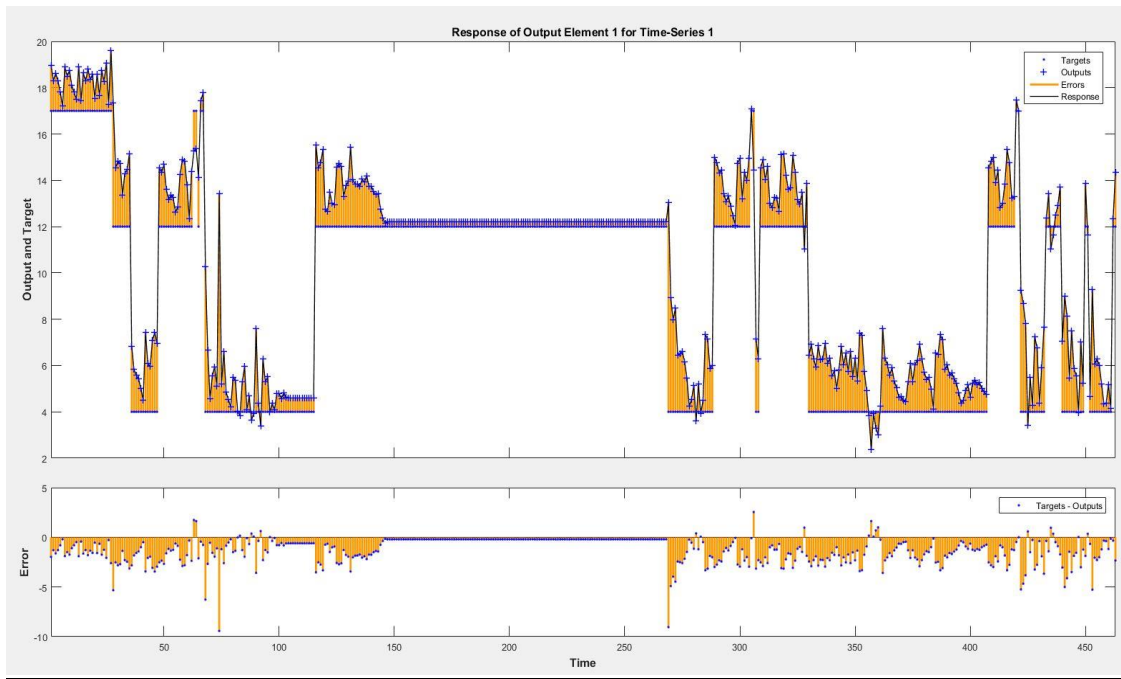


Figure 3.6: Plot Response of NARX Net 1 When Training with [X2, T2].

In figure 3.6, among thousands of events, only three have predicting errors bigger than 5. Most of them have predicting errors between 0 and 2, which indicates this is a success training. 30% of the failure events have a completely predictable pattern, as the errors are 0.

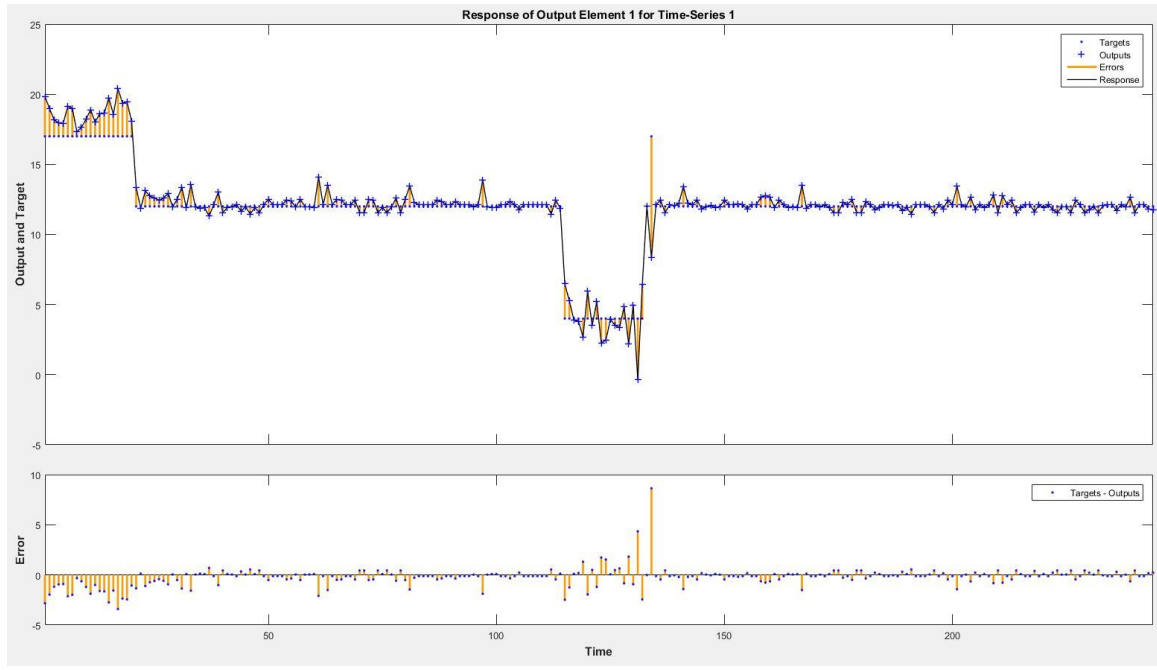


Figure 3.7: Plot Response of NARX Net 2 When Training with [X4, T4].

From figure 3.7, Xen hypervisor environment is clearly more predictable, because of the overwhelmingly large amount of system events generated by Dom0. Not only the failure event, but all other types of events in Xen environment is potentially predictable, as the MSE was 0.1842 (equivalent to 91.38% accuracy).

3.6 Chapter Summary

This chapter presented the complete process that was used to design a neural network time series model, the techniques behind how to choose the Input-Delay & Feedback-Delay values, how to choose the number of hidden neuron, and most importantly, the implementation of LM training techniques in

traditional neural network back propagation approach. At the end, it also mentioned about some successful experimenting results using this neural network design and the dataset described in chapter 2.

4. CONCLUSION AND FUTURE DIRECTION

4.1 Conclusion

Cloud failure event is predictable

According to recent experiments, failure in virtualization system is predictable by using time delay neural network. The predictability level in Hypervisor type 2 system is less accuracy than in hypervisor type 1 system, due to the lack of directly and fully control all hardware components of the physical machine. Furthermore, in hypervisor type 2, not only the failure events, but other events are also predictable, with the accuracy fall between 80% - 92% when using MSE measurement, this promises a great potential in cloud computing supervising effectiveness.

However, like many other experiments, all the results are generated from simulated environment, therefore it still needs more practical and realistic experimenting data. This can be considered as one of the future directions. The lack of real users activities and real cloud network interface are two of the major things that can cause incorrect assumptions in this paper.

There was an argument about whether or not the real cloud system with VMs on different geographic location can cause delay in event recording, thus may shuffle the events order and incorrect the pattern recognition process. However, according to Windows and Linux official documents about system event logs, the characteristic of system event log recording is to record an event

that has happened in the past, and use it for system forensics if there's any system crash in the future, accordingly, delay in event generating and event writing is acceptable and tolerable.

In conclusion, the success of this technique opens a new approach to cloud computing data analysis, virtualization system's failure prediction and prevention. If it can be used in the industry, It will help increasing the stability of cloud service, the reliability of cloud providers, as well as reducing frustration and depression in cloud users.

4.2 Future Direction

In the small scale or experiment, applying the same approach on slightly different platform or changing the way of data collecting can provide better understanding about this technique. The future recommending virtualization system should has the same design, but should be launched on the more powerful physical machine, that can handle 8 or 16 VMs with identical specs or higher. More VMs will force system hypervisor to work harder, and record more interrelation events. The more communications between VMs can be fed to the neural network system, the more accurate predictions the neural network system can generate. For data collecting method, the simplest way is to change virtual users' behavior (described in chapter 2.6). By consecutively launch heavy applications in all VMs, the system will reach its peak, where the sharing resource is overwhelmed by the amount of tasks, this will squeeze the

performance of the hypervisor domain, thus can generate more valuable result about its VMs managements strategies. Another variant of this direction should be designed to focus only on a particular part of the physical machine, such as network card, or GPU, or BUS controller, to experience how predictable it is to predict failure that may occur in certain part of the physical machine in sharing environment.

The second direction is to answer the question: “whether or not, these predictable failures can be prevented from happening in the virtualization system? “. To answer this question, after fully trained, the neural network system should be adapted in real time virtualization system as an AI virtual assistant. Throughout the time, this AI virtual assistant will take all the system events as input and return upcoming events with high failure rate. The job of the researcher in this future study is to find a way to prevent these failure events from happening. In the end if there are significantly fewer failures occurrence, this can be considered as a positive answer for that question.

And last but not least, from managing, recognizing and predicting system events, there’s a potential to propose a new cloud computing reliability frameworks. This new framework should measure IaaS cloud reliability level based on certain characteristic such as the frequency of critical/failure events, the mean risk rate that failures may occur in the system, the frequency of event patterns with high failure rate... As mentioned in chapter 1.1, reliability level is one of the most important characteristic in cloud computing environment. This

new framework can give great contributions and more accuracy measurements to cloud systems that are trying to use the adaptive version of this neural network system as an AI virtual assistant, hence can enormously improve this AI virtual assistant performance.

APPENDIX A
SETUP VIRTUALIZATION ENVIRONMENT AND DATA COLLECTING

UbuntuAutoUsers.cpp

```
1. #include <iostream>
2. #include <stdlib.h>
3. #include <unistd.h>
4. using namespace std;
5.
6. int main()
7. {
8.     int i=1;
9.     while (i==1)
10.    {
11.        system("libreoffice --
12.        norestore wordtest.doc &");
13.        sleep(3);
14.        system("libreoffice --
15.        norestore exceltest.xlsx &");
16.        sleep(3);
17.        system("libreoffice --
18.        norestore ppttest.pptx &");
19.        sleep(3);
20.        system("gedit txttest.txt &");
21.        sleep(3);
22.        system("gimp painttest.jpg &");
23.        sleep(4);
24.        system("totem videotest.mp4 &");
25.        sleep(3);
26.        system("firefox http://ebay.com &");
27.        sleep(3);
28.        system("midori http://ebay.com &");
29.        sleep(3);
30.        system("gnome-calculator &");
31.        sleep(3);
32.        system("nautilus &");
33.        sleep(3);
34.        system("gnome-system-monitor &");
35.        sleep(3);
36.        sleep(20);
37.        system("killall soffice.bin");
38.        sleep(2);
39.        system("killall gedit");
40.        sleep(2);
41.    }
```

```

39.         system("pkill -e script-fu");
40.         sleep(2);
41.         system("pkill -e gimp");
42.         sleep(2);
43.         system("killall totem");
44.         sleep(2);
45.         system("killall firefox");
46.         sleep(2);
47.         system("killall midori");
48.         sleep(2);
49.         system("killall gnome-calculator");
50.         sleep(2);
51.         system("killall nautilus");
52.         sleep(2);
53.         system("killall gnome-system-monitor");
54.         sleep(1);
55.         system("rm -r /tmp/*");
56.         sleep(5);
57.         system("rm -
r /home/vboclinuxguest/.cache/mozilla/firefox/*");
58.         sleep(5);
59.     }
60.     return 0;
61. }

```

WindowsAutoUsers.cpp

```

1. #include <iostream>
2. #include <windows.h>
3. #include <stdlib.h>
4. #include <time.h>
5.
6. using namespace std;
7.
8. int main()
9. {
10.     int loop = 1;
11.     while (loop == 1)
12.     {
13.         system("start C:\\wordtest.doc");
14.         Sleep(2000);
15.         system("start C:\\exceltest.xlsx");

```

```
16. Sleep(2000);
17. system("start C:\\ppttest.pptx");
18. Sleep(2000);
19. system("start notepad.exe C:\\txtttest.txt");
20. Sleep(2000);
21. system("start mspaint.exe C:\\paintttest.jpg");

22. Sleep(2000);
23. system("start C:\\videotest.mp4");
24. Sleep(2000);
25. system("start firefox.exe http://ebay.com");
26. Sleep(2000);
27. system("start iexplorer.exe http://bing.com");
28. Sleep(2000);
29. system("start calc");
30. Sleep(2000);
31. system("start explorer.exe");
32. Sleep(2000);
33. system("start taskmgr Performance");
34.
35.
36. Sleep(10000);
37. system("taskkill /im winword.exe");
38. Sleep(2000);
39. system("taskkill /im excel.exe");
40. Sleep(2000);
41. system("taskkill /im powerpnt.exe");
42. Sleep(2000);
43. system("taskkill /im notepad.exe");
44. Sleep(2000);
45. system("taskkill /im mspaint.exe");
46. Sleep(2000);
47. system("taskkill /im vlc.exe");
48. Sleep(2000);
49. system("taskkill /im calc.exe");
50. Sleep(2000);
51. system("taskkill /im iexplorer.exe");
52. Sleep(2000);
53. system("taskkill /im chrome.exe");
54. Sleep(2000);
55. system("taskkill /im ExplorerXP.exe");
56. Sleep(2000);
57. system("taskkill /im taskmgr.exe");
```

```
58.
59.         Sleep(1000);
60.     }
61.     return 0;
62. }
```

rsyslog.conf

```
1. # /etc/rsyslog.conf Configuration file for rsyslog.
2. #
3. # For more information see
4. # /usr/share/doc/rsyslog-
  doc/html/rsyslog_conf.html
5. #
6. # Default logging rules can be found in /etc/rsyslog.d/50-
  default.conf
7.
8.
9. #####
10.     #### MODULES ####
11.     #####
12.
13.     $ModLoad imuxsock # provides support for local system
  logging
14.     $ModLoad imklog # provides kernel logging support
15.     #$ModLoad immark # provides --MARK-
  - message capability
16.
17.     # provides UDP syslog reception
18.     #$ModLoad imudp
19.     #$UDPServerRun 514
20.
21.     # provides TCP syslog reception
22.     #$ModLoad imtcp
23.     #$InputTCPServerRun 514
24.
25.
26.     #####
27.     #### GLOBAL DIRECTIVES ####
28.     #####
29.
30.     #
```



```

31.     # Use traditional timestamp format.
32.     # To enable high precision timestamps, comment out the
      following line.
33.     #
34.     $ActionFileDefaultTemplate RSYSLOG_TraditionalFileForm
      at
35.     $template strtpl,"PRI: %pri-
      text%,\nSeverity: %syslogseverity-
      text%,\nFacility: %syslogfacility-
      text%,\nTimeGenerated: %timegenerated%,\nTimeReported: %time
      reported%,\nHostname: %HOSTNAME%,\nProgramName: %programname
      %,\nFromHost: %fromhost%,\nTAG: %syslogtag%,\nInfoUnitType:
      %iut%,\nMsg: %msg%,\nRawMsg: %rawmsg%\nProtocolVersion: %pro
      tocol-version%,\nStructuredData: %structured-
      data%,\nAppName: %app-
      name%,\nProcid: %procid%,\nMsgid: %msgid%,\nInputname: %inpu
      tname%\n\n"
36.     $ActionFileDefaultTemplate strtpl
37.
38.     # Filter duplicated messages
39.     $RepeatedMsgReduction on
40.
41.     #
42.     # Set the default permissions for all log files.
43.     #
44.     $FileOwner syslog
45.     $FileGroup adm
46.     $FileCreateMode 0640
47.     $DirCreateMode 0755
48.     $Umask 0022
49.     $PrivDropToUser syslog
50.     $PrivDropToGroup syslog
51.
52.     #
53.     # Where to place spool and state files
54.     #
55.     $WorkDirectory /var/spool/rsyslog
56.
57.     #
58.     # Include all config files in /etc/rsyslog.d/
59.     #
60.     $IncludeConfig /etc/rsyslog.d/*.conf

```

evt2txt.cpp

```
1. // ConsoleApplication1.cpp : Defines the entry point for the
   console application.
2. //
3. #include <iostream>
4. #include <fstream>
5. #include <vector>
6. #include <string.h>
7. #include <string>
8. #include <time.h>
9. #include <iomanip>
10.
11.     using namespace std;
12.     using PCHAR = char *;
13.
14.     typedef unsigned long ULONG;
15.     typedef struct _EVENTLOGHEADER {
16.         ULONG HeaderSize;
17.         ULONG Signature;
18.         ULONG MajorVersion;
19.         ULONG MinorVersion;
20.         ULONG StartOffset;
21.         ULONG EndOffset;
22.         ULONG CurrentRecordNumber;
23.         ULONG OldestRecordNumber;
24.         ULONG MaxSize;
25.         ULONG Flags;
26.         ULONG Retention;
27.         ULONG EndHeaderSize;
28.     } EVENTLOGHEADER, *PEVENTLOGHEADER;
29.
30.     typedef unsigned long DWORD;
31.     typedef unsigned short WORD;
32.     typedef struct _EVENTLOGRECORD {
33.         DWORD Length;
34.         DWORD Reserved;
35.         DWORD RecordNumber;
36.         DWORD TimeGenerated;
37.         DWORD TimeWritten;
38.         DWORD EventID;
39.         WORD EventType;
```

```

40.     WORD   NumStrings;
41.     WORD   EventCategory;
42.     WORD   ReservedFlags;
43.     DWORD  ClosingRecordNumber;
44.     DWORD  StringOffset;
45.     DWORD  UserSidLength;
46.     DWORD  UserSidOffset;
47.     DWORD  DataLength;
48.     DWORD  DataOffset;
49.     } EVENTLOGRECORD, *PEVENTLOGRECORD;
50.
51.
52.     struct WevtutilTextFile {
53.         string newEventID;
54.         string newKeyword;
55.         string newUserSID;
56.         string newUsername;
57.         string newDescription;
58.     };
59.
60.
61.     void main()
62.     {
63.         _EVENTLOGHEADER logheader;
64.         _EVENTLOGRECORD logRecord;
65.
66.         vector<WevtutilTextFile> WevtutilTxtFileVector;
67.
68.         ifstream file, wevtutilTxtFile;
69.         //file.open("C:\\Windows\\System32\\winevt\\Logs\\
Application.evtx", ios::in | ios::binary);
70.         file.open("D:\\win2k3.evt", ios::in | ios::binary)
;
71.
72.         wevtutilTxtFile.open("D:\\win2k3.txt", ios::in);
73.
74.         string wevtutilTxtFind, newUserSID, newUsername, n
ewEventID, newKeyword, newDescription;
75.
76.
77.         wofstream output;
78.         output.open("D:\\out.txt");
79.

```

```

80.
81.     if (!file)
82.         printf("couldn't open evt file\n");
83.     else
84.         if (!output)
85.             printf("couldn't open output.txt file");
86.         else
87.             if (!wevtutilTxtFile)
88.                 printf("couldn't open wevtutil text fi
89. le");
90.             else
91.                 {
92.                     //Reading the header
93.                     //file.read((char*)&logheader, sizeof(
94. _EVENTLOGHEADER));
95.                     file.read((PCHAR)&logheader, sizeof(_E
96. VENTLOGHEADER));
97.                     output <<
98.                         /*"number of bytes in Header" << "
99. : " <<
100. sizeof(_EVENTLOGHEADER) << "\n" <<
101. */
102. "HeaderSize" << ": " <<
103. logheader.HeaderSize << ",\n" <<
104. "Signature" << ": " <<
105. logheader.Signature << ",\n" <<
106. "MajorVersion" << ": " <<
107. logheader.MajorVersion << ",\n" <<
108. "MinorVersion" << ": " <<
109. logheader.MinorVersion << ",\n" <<
110. "StartOffset" << ": " <<
111. logheader.StartOffset << ",\n" <<
112. "EndOffset" << ": " <<
113. logheader.EndOffset << ",\n" <<
114. "CurrentRecordNumber" << ": " <<
115. logheader.CurrentRecordNumber - 1
116. << ",\n" <<
117. "OldestRecordNumber" << ": " <<

```

```

113.         logheader.OldestRecordNumber << "\n" <<
114.         "MaxSize" << ": " <<
115.         logheader.MaxSize << ",\n" <<
116.         "Flags" << ": " <<
117.         logheader.Flags << ",\n" <<
118.         "Retention" << ": " <<
119.         logheader.Retention << ",\n" <<
120.         "EndHeaderSize" << ": " <<
121.         logheader.EndHeaderSize << ",\n" <
    < endl;
122.
123.     }
124.     //find string from wevtutil export txt file
125.
126.     for (int vectorcounting = 0; vectorcounting < logh
eader.CurrentRecordNumber - 1; vectorcounting++) {
127.         WevtutilTxtFileVector.push_back(WevtutilTextFi
le()); //add 1 more element to vector
128.         while (!wevtutilTxtFile.eof()) {
129.             getline(wevtutilTxtFile, wevtutilTxtFind);
130.             //go through text file line by line
131.             if (wevtutilTxtFind.find("Event ID:") != s
tring::npos) {
132.                 newEventID = wevtutilTxtFind;
133.                 for (int i = 0; i < newEventID.length(
); i++)
134.                     if (newEventID[i] == ' ') newEvent
ID.erase(i, 1);
135.                 WevtutilTxtFileVector[vectorcounting].
newEventID = newEventID;
136.                 //strcpy_s(str2charEventID, newEventID
.c_str());
137.             }
138.             if (wevtutilTxtFind.find("Keyword:") != st
ring::npos) {
139.                 newKeyword = wevtutilTxtFind;
140.                 for (int i = 0; i < newKeyword.length(
); i++)
141.                     if (newKeyword[i] == ' ') newKeywo
rd.erase(i, 1);
142.                 WevtutilTxtFileVector[vectorcounting].
newKeyword = newKeyword;

```

```

142.                                     //strcpy_s(str2charnewKeyword, newKeyw
ord.c_str());
143.                                     }
144.                                     if (wevtutilTxtFind.find("User:") != strin
g::npos) {
145.                                         newUserSID = wevtutilTxtFind;
146.                                         for (int i = 0; i < newUserSID.length(
); i++)
147.                                             if (newUserSID[i] == ' ') newUserS
ID.erase(i, 1);
148.                                         WevtutilTxtFileVector[vectorcounting].
newUserSID = newUserSID;
149.                                         //strcpy_s(str2charnewnewUserSID, newU
serSID.c_str());
150.                                     }
151.                                     if (wevtutilTxtFind.find("User Name:") !=
string::npos) {
152.                                         newUsername = wevtutilTxtFind;
153.                                         for (int i = 0; i < newUsername.length
()); i++)
154.                                             if (newUsername[i] == ' ') newUser
name.erase(i, 1);
155.                                         WevtutilTxtFileVector[vectorcounting].
newUsername = newUsername;
156.                                         //strcpy_s(str2charnewnewUsername, new
Username.c_str());
157.                                     }
158.                                     if (wevtutilTxtFind.find("Description:") !
= string::npos) {
159.                                         newDescription = wevtutilTxtFind;
160.                                         while (!(wevtutilTxtFind.empty())) {
161.                                             getline(wevtutilTxtFile, wevtutilT
xtFind);
162.                                             newDescription = newDescription +
wevtutilTxtFind;
163.                                         }
164.                                         WevtutilTxtFileVector[vectorcounting].
newDescription = newDescription;
165.                                         //strcpy_s(str2charnewDescription, new
Description.c_str());
166.                                     }
167.                                     if (wevtutilTxtFind.empty()) break;
168.                                     }

```

```

169.     }
170.     char str2charnewnewUserSID[100];
171.     char str2charnewnewUsername[100];
172.     char str2charEventID[100];
173.     char str2charnewKeyword[100];
174.     char str2charnewDescription[2000];
175.
176.     //Loop on every record
177.     int startOfLog;
178.     int totalElement = logheader.CurrentRecordNumber -
179.     1;
180.     for (int numberFile = 0; numberFile < logheader.Cu
181.     rrentRecordNumber - 1; numberFile++) {
182.         //assign vector value
183.         totalElement--;
184.         strcpy_s(str2charEventID, WevtutilTxtFileVecto
185.         r[totalElement].newEventID.c_str());
186.         strcpy_s(str2charnewKeyword, WevtutilTxtFileVe
187.         ctor[totalElement].newKeyword.c_str());
188.         strcpy_s(str2charnewnewUserSID, WevtutilTxtFil
189.         eVector[totalElement].newUserSID.c_str());
190.         strcpy_s(str2charnewnewUsername, WevtutilTxtFi
191.         leVector[totalElement].newUsername.c_str());
192.         strcpy_s(str2charnewDescription, WevtutilTxtFi
193.         leVector[totalElement].newDescription.c_str());
194.
195.         //Save the position
196.         startOfLog = file.tellg();
197.         //Read log record
198.         file.read((char*)&logRecord, sizeof(_EVENTLOGR
199.         ECORD));
200.
201.         ////////////////////////////////////////////////////Here are the other information
202.         (section 'Remarks' on the 'EVENTLOGRECORD structure' link
203.
204.         //Reading sourcename
205.         wchar_t buffData;
206.         wstring SourceName;
207.         file.read((char*)&buffData, sizeof(wchar_t));
208.
209.         while (buffData != L'\0') {

```

```

202.             SourceName.push_back(buffData);
203.             file.read((char*)&buffData, sizeof(wchar_t
));
204.         }
205.         string wstr2strSourceName(SourceName.begin(),
SourceName.end());
206.         char str2charSourceName[100];
207.         strcpy_s(str2charSourceName, wstr2strSourceNam
e.c_str());
208.
209.         //Reading computer name
210.         wstring ComputerName;
211.         file.read((char*)&buffData, sizeof(wchar_t));
212.         while (buffData != L'\0') {
213.             ComputerName.push_back(buffData);
214.             file.read((char*)&buffData, sizeof(wchar_t
));
215.         }
216.         string wstr2strComputerName(ComputerName.begin
(), ComputerName.end());
217.         char str2charComputerName[100];
218.         strcpy_s(str2charComputerName, wstr2strCompute
rName.c_str());
219.
220.
221.         //Sets the position to the SID offset
222.         int readCursor = startOfLog + logRecord.UserSi
dOffset;
223.         file.seekg(readCursor);
224.
225.
226.         char *userSid = NULL;
227.         //char str2charuserSid[100];
228.         if (logRecord.UserSidLength != 0)
229.         {
230.             userSid = (char*)malloc(logRecord.UserSidL
ength*sizeof(char));
231.             file.read((char*)userSid, logRecord.UserSi
dLength); //Reading the sid
232.
                //Here you can work on the SiD (but you need win32
API)

```



```

233.
234.         //string chr2struserSid = string(userSid);
235.         //char str2charuserSid[100];
236.         //strcpy_s(str2charuserSid, chr2struserSid.c_str()
    );
237.     }
238.     //free(userSid);
239.
240.
241.     //Sets the position to the Strings offset
242.     readCursor = startOfLog + logRecord.StringOffs
    et;
243.     file.seekg(readCursor);
244.     wstring buffString;
245.     vector<wstring> allStrings;
246.
247.     //Reading all the strings
248.     for (int i = 0; i < logRecord.NumStrings; i++)
    {
249.         file.read((char*)&buffData, sizeof(wchar_t
    ));
250.         while (buffData != L'\0') {
251.             buffString.push_back(buffData);
252.             file.read((char*)&buffData, sizeof(wch
    ar_t));
253.         }
254.         allStrings.push_back(buffString);
255.
256.         buffString.clear();
257.     }
258.
259.
260.     //Sets the position to the Data offset
261.     readCursor = startOfLog + logRecord.DataOffset
    ;
262.     file.seekg(readCursor);
263.     unsigned char *Data = (unsigned char *)malloc(
    logRecord.DataLength*sizeof(unsigned char));
264.     file.read((char*)Data, logRecord.DataLength);
    //Lecture des données

```

```

265.         char *uc2cData = reinterpret_cast<char *> (Data);
266.
267.         string chr2strData = string(uc2cData);
268.         char str2charData[100];
269.         strcpy_s(str2charData, chr2strData.c_str());
270.
271.
272.         //Sets the position to the end of log offset
273.         readCursor = startOfLog + logRecord.Length - s
sizeof(DWORD);
274.         file.seekg(readCursor);
275.         DWORD length;
276.         file.read((char*)&length, sizeof(DWORD));
277.
278.
279.         //Do what you want with the log record
280.         output <<
281.             "Event Record ID" << ": " <<
282.             numberFile << ",\n " <<
283.             "Length" << ": " <<
284.             logRecord.Length << ",\n " <<
285.             "Reserved" << ": " <<
286.             logRecord.Reserved << ",\n " <<
287.             "RecordNumber" << ": " <<
288.             logRecord.RecordNumber << ", " << endl;
289.
290.
291.         //convert DWORD -> epoch time_t -> time -
> string
292.         struct tm timet2timeTimeGenerated;
293.         time_t dword2timetTimeGenerated;
294.         dword2timetTimeGenerated = logRecord.TimeGener
ated;
295.         localtime_s(&timet2timeTimeGenerated, &dword2t
imetTimeGenerated);
296.         char time2stringTimeGenerated[32];
297.         asctime_s(time2stringTimeGenerated, 32, &timet
2timeTimeGenerated);
298.         time2stringTimeGenerated[24] = ',';
299.         time2stringTimeGenerated[25] = '\0';
300.         output <<

```

```

301.         "TimeGenerated In epoch" << ": " <<
302.         logRecord.TimeGenerated << ",\n " <<
303.         "TimeGenerated In local datetime" << ": "
304.         <<
305.         (char*)&time2stringTimeGenerated << "\n ";
306.
307.         //convert DWORD -> epoch time_t -> time -
308.         > string
309.         struct tm timet2timeTimeWritten;
310.         time_t dword2timetTimeWritten;
311.         dword2timetTimeWritten = logRecord.TimeWritten
312.         ;
313.         localtime_s(&timet2timeTimeWritten, &dword2tim
314.         etTimeWritten);
315.         char time2stringTimeWritten[32];
316.         asctime_s(time2stringTimeWritten, 32, &timet2t
317.         imeTimeWritten);
318.         time2stringTimeWritten[24] = ',';
319.         time2stringTimeWritten[25] = '\0';
320.         output <<
321.         "TimeWritten In epoch" << ": " <<
322.         logRecord.TimeWritten << ",\n " <<
323.         "TimeWritten In local datetime" << ": " <<
324.
325.         (char*)&time2stringTimeWritten << "\n ";
326.
327.         output <<
328.         "EventID in hexadecimal" << ": " <<
329.         logRecord.EventID << ",\n " <<
330.         (char*)&str2charEventID << ",\n " <<
331.         "EventType" << ": " <<
332.         logRecord.EventType << ",\n " <<
333.         "NumStrings" << ": " <<
334.         logRecord.NumStrings << ",\n " <<
335.         "EventCategory" << ": " <<
336.         logRecord.EventCategory << ",\n " <<
337.         "ReservedFlags" << ": " <<
338.         logRecord.ReservedFlags << ",\n " <<
339.         "ClosingRecordNumber" << ": " <<

```

```

336.         logRecord.ClosingRecordNumber << ",\n " <<
337.         "StringOffset" << ": " <<
338.         logRecord.StringOffset << ",\n " <<
339.         "UserSidLength" << ": " <<
340.         logRecord.UserSidLength << ",\n " <<
341.         "UserSidOffset" << ": " <<
342.         logRecord.UserSidOffset << ",\n " <<
343.         "DataLength" << ": " <<
344.         logRecord.DataLength << ",\n " <<
345.         "DataOffset" << ": " <<
346.         logRecord.DataOffset << ",\n " <<
347.
348.         "Source name" << ": " <<
349.         (char*)&str2charSourceName << ",\n " <<
350.
351.         "Computer name" << ": " <<
352.         (char*)&str2charComputerName << ",\n " <<
353.
354.         //"User SID" << ": " <<
355.         //logRecord.UserSidLength << ",\n " <<
356.         (char*)&str2charnewKeyword << ",\n " <<
357.         (char*)&str2charnewnewUserSID << ",\n " <<
358.         (char*)&str2charnewnewUsername << ",\n " <
359.         <
360.         "Event Data (String):" << endl;
361.         for (int i = 0; i < logRecord.NumStrings; i++)
362.         {
363.             string wstr2strEventData(allStrings.at(i).
364.             begin(), allStrings.at(i).end());
365.             char str2charEventData[1000];
366.             strcpy_s(str2charEventData, wstr2strEventD
367.             ata.c_str());
368.             output <<
369.             (char*)&str2charEventData << ", " << en

```

```
370.         "Data" << ": " <<
371.         Data << ",\n " <<
372.         "Length" << ": " <<
373.         length << ",\n" <<
374.         (char*)&str2charnewDescription << ",\n " <
    < endl;
375.
376.
377.         //Clean before reading next log
378.         ComputerName.clear();
379.         SourceName.clear();
380.         allStrings.clear();
381.         //free(Data);
382.     }
383.
384.     file.close();
385.     output.close();
386.
387. }
```

APPENDIX B
DATA PREPROCESSING

UbuntuPreprocessedMacro.vba

```
1. Sub UbuntuPreprocessed()  
2. '  
3. ' UbuntuPreprocessed Macro  
4. '  
5.  
6. '  
7.     Columns("G:G").EntireColumn.AutoFit  
8.     Columns("G:G").Select  
9.     Range("G217").Activate  
10.    Selection.Copy  
11.    Columns("C:C").Select  
12.    Range("C217").Activate  
13.    ActiveSheet.Paste  
14.  
15.    Columns("A:A").Select  
16.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF  
    ormatFromLeftOrAbove  
17.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF  
    ormatFromLeftOrAbove  
18.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF  
    ormatFromLeftOrAbove  
19.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF  
    ormatFromLeftOrAbove  
20.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF  
    ormatFromLeftOrAbove  
21.    Columns("I:I").Select  
22.    Selection.Copy  
23.    Columns("A:A").Select  
24.    ActiveSheet.Paste  
25.    Range("B1").Select  
26.    Application.CutCopyMode = False  
27.    ActiveCell.FormulaR1C1 = "=RC[-1]+0"  
28.    Range("B1").Select  
29.    Selection.AutoFill Destination:=Range("B1:B966")  
30.    Range("B1:B966").Select  
31.    Columns("B:B").Select  
32.    Selection.Copy  
33.    Columns("C:C").Select  
34.    Selection.PasteSpecial Paste:=xlPasteValues, Opera  
    tion:=xlNone, SkipBlanks _
```

```

35.         :=False, Transpose:=False
36.     Application.CutCopyMode = False
37.     Selection.NumberFormat = "mm/dd/yyyy hh:mm:ss"
38.     Columns("A:B").Select
39.     Range("B1").Activate
40.     Selection.Delete Shift:=xlToLeft
41.     Columns("E:E").Select
42.     Selection.Copy
43.     Columns("C:C").Select
44.     ActiveSheet.Paste
45.     Columns("F:F").Select
46.     Application.CutCopyMode = False
47.     Selection.Copy
48.     Columns("B:B").Select
49.     ActiveSheet.Paste
50.     Columns("D:AA").Select
51.     Application.CutCopyMode = False
52.     Selection.Delete Shift:=xlToLeft
53.     Selection.Delete Shift:=xlToLeft
54.     Range("D5").Select
55.
56.     Range("D1").Select
57.     ActiveCell.FormulaR1C1 = "="&RC[-1]"
58.     Range("D1").Select
59.     Selection.AutoFill Destination:=Range("D1:D966")
60.     Range("D1:D966").Select
61.     Selection.Copy
62.     Columns("E:E").Select
63.     Selection.PasteSpecial Paste:=xlPasteValues, Opera
tion:=xlNone, SkipBlanks _
64.         :=False, Transpose:=False
65.     Columns("C:D").Select
66.     Range("D1").Activate
67.     Application.CutCopyMode = False
68.     Selection.Delete Shift:=xlToLeft
69.     End Sub

```

WindowsPreprocessedMacro.vba

```

1. Sub WindowsPreprocessed()
2. '
3. ' WindowsPreprocessed Macro

```



```

4. '
5.
6. '
7.     Columns("D:D").Select
8.     Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatF
romLeftOrAbove
9.     Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatF
romLeftOrAbove
10.    Range("E1").Select
11.    ActiveCell.FormulaR1C1 = "="&"WG-"&"&RC[1]"
12.    Range("E1").Select
13.    Selection.AutoFill Destination:=Range("E1:E536")
14.    Range("E1:E536").Select
15.    Columns("E:E").Select
16.    Selection.Copy
17.    Range("D1").Select
18.    Selection.PasteSpecial Paste:=xlPasteValues, Opera
tion:=xlNone, SkipBlanks _
19.        :=False, Transpose:=False
20.
21.    Columns("E:N").Select
22.    Selection.Delete Shift:=xlToLeft
23.    Columns("C:C").Select
24.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF
ormatFromLeftOrAbove
25.    Range("C1").Select
26.    ActiveCell.FormulaR1C1 = "=RC[-2]+RC[-1]"
27.    Range("C1").Select
28.    Selection.AutoFill Destination:=Range("C1:C536")
29.    Range("C1:C536").Select
30.    Columns("C:C").Select
31.    Selection.NumberFormat = "mm/dd/yyyy hh:mm:ss"
32.    Columns("C:C").EntireColumn.AutoFit
33.    Columns("C:C").Select
34.    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF
ormatFromLeftOrAbove
35.    Columns("D:D").Select
36.    Selection.Copy
37.    Columns("C:C").Select
38.    Selection.PasteSpecial Paste:=xlPasteValues, Opera
tion:=xlNone, SkipBlanks _
39.        :=False, Transpose:=False
40.    Columns("A:B").Select

```

```

41.         Application.CutCopyMode = False
42.         Selection.Delete Shift:=xlToLeft
43.         Columns("B:B").Select
44.         Selection.Delete Shift:=xlToLeft
45.     End Sub

```

CombineDataMacro.vba

```

1. Sub CombineDataPreprocessed()
2. '
3. ' CombineDataPreprocessed Macro
4. '
5.
6. '
7.     Columns("A:A").Select
8.     ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
9.     ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add
10.    Key:=Range("A1"), _
11.    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
12.    With ActiveWorkbook.Worksheets("Sheet1").Sort
13.        .SetRange Range("A1:C3725")
14.        .Header = xlNo
15.        .MatchCase = False
16.        .Orientation = xlTopToBottom
17.        .SortMethod = xlPinYin
18.        .Apply
19.    End With
20.    Selection.Delete Shift:=xlToLeft
21.    Range("C1").Select
22.    ActiveCell.FormulaR1C1 = _
23.    "=IF(LEFT(RC[-1],2)=""WG"", VLOOKUP(RC[-2],R80C11:R103C14,4,FALSE),IF(LEFT(RC[-1],2)=""LG"",VLOOKUP(RC[-2],R43C12:R79C14,3,FALSE),VLOOKUP(RC[-2],R2C13:R42C14,2,FALSE)))"
24.    Range("C1").Select
25.    Selection.AutoFill Destination:=Range("C1:C3725")
26.
27.    Range("C1:C3725").Select
28.    Selection.Copy

```

```

27.         Columns("D:D").Select
28.         Selection.PasteSpecial Paste:=xlPasteValues, Opera
           tion:=xlNone, SkipBlanks _
29.             :=False, Transpose:=False
30.         Range("E1").Select
31.         Application.CutCopyMode = False
32.         ActiveCell.FormulaR1C1 = ""
33.         Range("H10:H17").Select
34.         Selection.Copy
35.         Range("I10").Select
36.         ActiveSheet.Paste
37.         Range("G18:G20").Select
38.         Application.CutCopyMode = False
39.         Selection.Copy
40.         Range("I18").Select
41.         ActiveSheet.Paste
42.         Application.CutCopyMode = False
43.         Range("E1").Select
44.         ActiveCell.FormulaR1C1 = "=VLOOKUP(RC[-
           3],R2C9:R20C10,2,FALSE)"
45.         Range("E1").Select
46.         Selection.AutoFill Destination:=Range("E1:E3725")
47.         Range("E1:E3725").Select
48.         Selection.Copy
49.         Columns("F:F").Select
50.         Selection.PasteSpecial Paste:=xlPasteValues, Opera
           tion:=xlNone, SkipBlanks _
51.             :=False, Transpose:=False
52.         Range("F6").Select
53.         Application.CutCopyMode = False
54.         Columns("E:E").Select
55.         Selection.Delete Shift:=xlToLeft
56.         Columns("C:C").Select
57.         Selection.Insert Shift:=xlToRight, CopyOrigin:=xlF
           ormatFromLeftOrAbove
58.         Selection.Delete Shift:=xlToLeft
59.         Selection.Delete Shift:=xlToLeft
60.         Columns("A:B").Select
61.         Selection.Delete Shift:=xlToLeft
62.         Columns("C:J").Select
63.         Selection.Delete Shift:=xlToLeft
64.         End Sub

```

APPENDIX C
DESIGNING NEURAL NETWORK NON-LINEAR TIME SERIES AND PLOT
VALIDATION

nnetGenerated.m

```
1. X = Input_set;
2. T = Target_set;
3. net = narxnet(1:30,1:30,32);
4. [Xs,Xi,Ai,Ts] = preparets(net,X,{},T);
5. net = train(net,Xs,Ts,Xi,Ai);
6. view(net)
7. Y = net(Xs,Xi,Ai)
```

mse_editted.m

```
1. function y = mse_editted(Reference_Image, Target_Image)
2.
3.     Reference_ImageReference_Image = Reference_Image;
4.     Target_ImageTarget_Image = Target_Image;
5.     [M N] = size(Reference_Image);
6.     count =0;
7.     sum_err =0;
8.     value1 =0;
9.     value2 =0;
10.
11.     for i=1:M
12.         for j=1:N
13.             value1 = Target_Image{i,j};
14.             value2 = Reference_Image{i,j};
15.             %fprintf('%d \n',value1);
16.             if value1 == 4 || value1 == 12 || value1 == 1
17.                 %if value1 == 4
18.                     countcount= count+1;
19.                     %error = Target_Image{i,j} - Reference_Ima
20.                     ge{i,j};
21.                     error = value1 - value2;
22.                     sum_err = error * error;
23.                 end
24.             end
25.         end
26.     end
27.     y = sum_err/count;
end
```

plotregression_editted.m

```
1. function out1 = plotregression(varargin)
2. %PLOTREGRESSION Plot linear regression.
3.
4. %% =====
5. % BOILERPLATE_START
6. % This code is the same for all Transfer Functions.
7.
8. persistent INFO;
9. if isempty(INFO), INFO = get_info; end
10.     if nargin == 0
11.         fig = nnplots.find_training_plot(mfilename);
12.         if nargout > 0
13.             out1 = fig;
14.         elseif ~isempty(fig)
15.             figure(fig);
16.         end
17.         return;
18.     end
19.     in1 = varargin{1};
20.     if ischar(in1)
21.         switch in1
22.             case 'info',
23.                 out1 = INFO;
24.             case 'data_suitable'
25.                 data = varargin{2};
26.                 out1 = nnet.train.isNotParallelData(data);
27.             case 'suitable'
28.                 [args,param] = nnparam.extract_param(varargin,
                INFO.defaultParam);
29.                 [net,tr,signals] = deal(args{2:end});
30.                 update_args = standard_args(net,tr,signals);
31.                 unsuitable = unsuitable_to_plot(param,update_a
                rgs{:});
32.                 if nargout > 0
33.                     out1 = unsuitable;
34.                 elseif ~isempty(unsuitable)
35.                     for i=1:length(unsuitable)
36.                         disp(unsuitable{i});
37.                     end
38.                 end
```

```

39.         case 'training_suitable'
40.             [net,tr,signals,param] = deal(varargin{2:end})
41.             ;
42.             update_args = training_args(net,tr,signals,param);
43.             unsuitable = unsuitable_to_plot(param,update_a
44.             rgs{:});
45.             if nargin > 0
46.                 out1 = unsuitable;
47.             elseif ~isempty(unsuitable)
48.                 for i=1:length(unsuitable)
49.                     disp(unsuitable{i});
50.                 end
51.             end
52.         case 'training'
53.             [net,tr,signals,param] = deal(varargin{2:end})
54.             ;
55.             update_args = training_args(net,tr,signals);
56.             fig = nnplots.find_training_plot(mfilename);
57.             if isempty(fig)
58.                 fig = figure('Visible','off','Tag',['TRAININ
59.                 G_' upper(mfilename)]);
60.             plotData = setup_figure(fig,INFO,true);
61.             else
62.                 plotData = get(fig,'UserData');
63.             end
64.             set_busy(fig);
65.             unsuitable = unsuitable_to_plot(param,update_a
66.             rgs{:});
67.             if isempty(unsuitable)
68.                 set(0,'CurrentFigure',fig);
69.                 plotData = update_plot(param,fig,plotData,up
70.                 date_args{:});
71.                 update_training_title(fig,INFO,tr)
72.                 nnplots.enable_plot(plotData);
73.             else
74.                 nnplots.disable_plot(plotData,unsuitable);
75.             end
76.             fig = unset_busy(fig,plotData);
77.             if nargin > 0, out1 = fig; end
78.         case 'close_request'
79.             fig = nnplots.find_training_plot(mfilename);
80.             if ~isempty(fig),close_request(fig); end

```

```

75.         case 'check_param'
76.             out1 = ''; % TODO
77.         otherwise,
78.             try
79.                 out1 = eval(['INFO.' in1]);
80.             catch me, nnerr.throw(['Unrecognized first arg
argument: '' in1 '''])
81.             end
82.         end
83.     else
84.         [args,param] = nnparam.extract_param(varargin,INFO
.defaultParam);
85.         update_args = standard_args(args{:});
86.         if ischar(update_args)
87.             nnerr.throw(update_args);
88.         end
89.         [plotData,fig] = setup_figure([],INFO,false);
90.         unsuitable = unsuitable_to_plot(param,update_args{
:});
91.         if isempty(unsuitable)
92.             plotData = update_plot(param,fig,plotData,update
_args{:});
93.             nnplots.enable_plot(plotData);
94.         else
95.             nnplots.disable_plot(plotData,unsuitable);
96.         end
97.         set(fig,'Visible','on');
98.         drawnow;
99.         if nargin > 0, out1 = fig; end
100.    end
101. end
102.
103. function set_busy(fig)
104.     set(fig,'UserData','BUSY');
105. end
106.
107. function close_request(fig)
108.     ud = get(fig,'UserData');
109.     if ischar(ud)
110.         set(fig,'UserData','CLOSE');
111.     else
112.         delete(fig);
113.     end

```



```

114.     drawnow;
115.     end
116.
117.     function fig = unset_busy(fig,plotData)
118.         ud = get(fig,'UserData');
119.         if ischar(ud) && strcmp(ud,'CLOSE')
120.             delete(fig);
121.             fig = [];
122.         else
123.             set(fig,'UserData',plotData);
124.         end
125.         drawnow;
126.     end
127.
128.     function tag = new_tag
129.         tagnum = 1;
130.         while true
131.             tag = [upper(mfilename) num2str(tagnum)];
132.             fig = nnplots.find_plot(tag);
133.             if isempty(fig), return; end
134.             tagnumtagnum = tagnum+1;
135.         end
136.     end
137.
138.     function [plotData,fig] = setup_figure(fig,info,isTraining)
139.         PTFS = nnplots.title_font_size;
140.         if isempty(fig)
141.             fig = get(0,'CurrentFigure');
142.             if isempty(fig) || strcmp(get(fig,'NextPlot'),'new
')
143.                 if isTraining
144.                     tag = ['TRAINING_' upper(mfilename)];
145.                 else
146.                     tag = new_tag;
147.                 end
148.                 fig = figure('Visible','off','Tag',tag);
149.                 if isTraining
150.                     set(fig,'CloseRequestFcn',[mfilename '('close
_request'')']);
151.                 end
152.             else
153.                 clf(fig);

```

```

154.         set(fig,'Tag','');
155.         set(fig,'Tag',new_tag);
156.     end
157. end
158. set(0,'CurrentFigure',fig);
159. ws = warning('off','MATLAB:Figure:SetPosition');
160. plotData = setup_plot(fig);
161. warning(ws);
162. if isTraining
163.     set(fig,'NextPlot','new');
164.     update_training_title(fig,info,[]);
165. else
166.     set(fig,'NextPlot','replace');
167.     set(fig,'Name',[info.name ' (' mfilename ')']);
168. end
169. set(fig,'NumberTitle','off','ToolBar','none');
170. plotData.CONTROL.text = uicontrol('Parent',fig,'Style',
    'text',...
171.     'Units','normalized','Position',[0 0 1 1],'FontSize',
    'PTFS,...
172.     'FontWeight','bold','ForegroundColor',[0.7 0 0]);

173.     set(fig,'UserData',plotData);
174. end
175.
176. function update_training_title(fig,info,tr)
177.     if isempty(tr)
178.         epochs = '0';
179.         stop = '';
180.     else
181.         epochs = num2str(tr.num_epochs);
182.         if isempty(tr.stop)
183.             stop = '';
184.         else
185.             stop = [', ' tr.stop];
186.         end
187.     end
188.     set(fig,'Name',['Neural Network Training ' ...
189.         info.name ' (' mfilename '), Epoch ' epochs stop])
190. ;
191. end
192. % BOILERPLATE_END

```

```

193.     %% =====
        ====
194.
195.
196.     % TODO - Implement try/catch & CloseRequestFcn to avoid errors when figure
197.     % is closed during call to a plot function
198.
199.     function info = get_info
200.         info = nnfcnPlot(mfilename,'Regression',7.0,[]);
201.     end
202.
203.     function args = training_args(net,tr,data)
204.         yall = nncalc.y(net,data.X,data.Xi,data.Ai);
205.         y = {yall};
206.         t = {gmultiply(data.train.mask,data.T)};
207.         names = {'Training'};
208.         if data.val.enabled
209.             y = [y {yall}];
210.             t = [t {gmultiply(data.val.mask,data.T)}];
211.             names = [names {'Validation'}];
212.         end
213.         if data.test.enabled
214.             y = [y {yall}];
215.             t = [t {gmultiply(data.test.mask,data.T)}];
216.             names = [names {'Test'}];
217.         end
218.         if length(t) >= 2
219.             t = [t {data.T}];
220.             y = [y {yall}];
221.             names = [names {'All'}];
222.         end
223.         args = {t y names};
224.     end
225.
226.     function args = standard_args(varargin)
227.         if nargin < 2
228.             args = 'Not enough input arguments.';
229.         elseif (nargin > 2) && (rem(nargin,3) ~= 0)
230.             args = 'Incorrect number of input arguments.';
231.         elseif nargin == 2
232.             % (t,y)
233.             t = { nntype.data('format',varargin{1}) };

```

```

234.     y = { nntype.data('format',varargin{2}) };
235.     names = {' '};
236.     args = {t y names};
237.     else
238.         % (t1,y1,name1,...)
239.         % TODO - Check data is consistent
240.         count = nargin/3;
241.         t = cell(1,count);
242.         y = cell(1,count);
243.         names = cell(1,count);
244.         for i=1:count
245.             t{i} = nntype.data('format',varargin{i*3-2});
246.             y{i} = nntype.data('format',varargin{i*3-1});
247.             names{i} = varargin{i*3};
248.         end
249.         param.outputIndex = 1;
250.         args = {t y names};
251.     end
252. end
253.
254. function plotData = setup_plot(fig)
255.     plotData.numSignals = 0;
256. end
257.
258. function fail = unsuitable_to_plot(param,t,y,names)
259.     fail = '';
260.     tt1 = t{1};
261.     if numsamples(tt1) == 0
262.         fail = 'The target data has no samples to plot.';
263.     elseif numtimesteps(tt1) == 0
264.         fail = 'The target data has no timesteps to plot.'
265.     ;
266.     elseif sum(numelements(tt1)) == 0
267.         fail = 'The target data has no elements to plot.';
268.     end
269. end
270. function plotData = update_plot(param,fig,plotData,tt,
    yy,names)
271.     PTFS = nnplots.title_font_size;
272.     trainColor = [0 0 1];

```

```

273.     valColor = [0 1 0];
274.     testColor = [1 0 0];
275.     allColor = [1 1 1] * 0.4;
276.     colors = {trainColor valColor testColor allColor};
277.
278.     % Number of signals
279.     for i=numel(tt):-1:1
280.         if numsamples(tt{i}) == 0
281.             % Remove empty datasets
282.             tt(i) = [];
283.             yy(i) = [];
284.             names(i) = [];
285.         end
286.     end
287.     numSignals = length(names);
288.     fprintf('numSignals %d \n',numSignals);
289.
290.     % Create axes
291.     if (plotData.numSignals ~= numSignals)
292.         set(fig,'NextPlot','replace');
293.         plotData.numSignals = numSignals;
294.         if numSignals == 1
295.             plotData.titleStyle = {'fontweight','bold','font
size',PTFS};
296.         else
297.             plotData.titleStyle = {'fontweight','bold','font
size',PTFS};
298.         end
299.         plotcols = ceil(sqrt(numSignals));
300.         plotrows = ceil(numSignals/plotcols);
301.         for plotrow=1:plotrows
302.             for plotcol=1:plotcols
303.                 i = (plotrow-1)*plotcols+plotcol;
304.                 if (i<=numSignals)
305.
306.                     a = subplot(plotrows,plotcols,i);
307.                     cla(a)
308.                     set(a,'DataAspectRatio',[1 1 1],'Box','on');
309.
310.                     xlabel(a,'Target',plotData.titleStyle{:});
311.                     hold on
312.                     plotData.axes(i) = a;

```

```

313.             plotData.eqLine(i) = plot([NaN NaN],[NaN NaN
    ],':k');
314.             color = colors{rem(i-
    1,length(colors))+1};
315.             plotData.regLine(i) = plot([NaN NaN],[NaN Na
    N],'LineWidth',2,'Color',color);
316.             % plotData.regLine(i) = plot([NaN NaN],[NaN N
    aN],'LineWidth',2,'Color',colors(0.5));
317.             plotData.dataPoints(i) = plot([NaN NaN],[NaN
    NaN],'ok');
318.             plotData.dataPoints2(i) = plot([NaN NaN],[Na
    N NaN],'ok','Color',color);
319.             legend([plotData.dataPoints(i),plotData.regL
    ine(i),plotData.eqLine(i)], ...
320.                 {'Data','Fit','Y = T'},'Location','NorthWe
    st');
321.
322.         end
323.     end
324. end
325.     screenSize = get(0,'ScreenSize');
326.     screenSizeScreenSize = screenSize(3:4);
327.     if numSignals == 1
328.         windowSize = [500 500];
329.     else
330.         windowSize = 700 * [1 (plotrows/plotcols)];
331.     end
332.     pos = [(screenSize-windowSize)/2 windowSize];
333.     set(fig,'Position',pos);
334. end
335.
336. % Fill axes
337. % fprintf('%d \n',numSignals);
338. for i=1:numSignals
339.     set(fig,'CurrentAxes',plotData.axes(i));
340.     y = cell2mat(yy{i}); yy = y(:)';
341.     t = cell2mat(tt{i}); tt = t(:)';
342.
343.
344.     t2 = [];
345.     y2 = [];
346.     count =1;
347.     for j=1: numel(t)

```

```

348.         if t(1,j) == 4 || t(1,j) == 12 || t(1,j) == 17
349.             t2(1,count) = t(1,j);
350.             y2(1,count) = y(1,j);
351.             countcount = count +1;
352.         end
353.     end
354.
355.     name = names{i};
356.     [r,m,b] = regression(t2,y2);
357.     mm = m(1); bb = b(1); rr = r(1);
358.     lim = [min([y2 t2]) max([y2 t2])];
359.     line = m*lim + b;
360.
361.     fprintf('r %d \n',r);
362.     fprintf('m %d \n',m);
363.     fprintf('b %d \n',b);
364.     fprintf('lim %d \n',lim);
365.     fprintf('line %d \n',line);
366.     %fprintf('y %d \n',y);
367.     %fprintf('t %d \n',t);
368.
369.     set(plotData.dataPoints(i),'XData',t2,'YData',y2);
370.
371.     set(plotData.regLine(i),'XData',lim,'YData',line)
372.
373.     set(plotData.eqLine(i),'XData',lim,'YData',lim);
374.
375.     set(gca,'XLim',lim);
376.     set(gca,'YLim',lim);
377.     axis('square')
378.
379.     ylabel(['Output ~= ',num2str(m,2),'*Target + ', num2str(b,2)],...
380.         plotData.titleStyle{:});
381.     title([name ': R=' num2str(r)],plotData.titleStyle{:});
382. end
383. drawnow
384. end

```

plotresponse_editted.m

```

1. function out1 = plotresponse(varargin)
2.
3. % Throw error if called with a Phased Array object.
4. if (nargin > 0) && ~isempty(strfind(lower(class(varargin{1}
    ),'phased'))
5.     error(message('nnet:plotresponse:PhasedArrayMethodShouldBe
        Called'));
6. end
7.
8. %% =====
9. % BOILERPLATE_START
10.     % This code is the same for all Transfer Functions.
11.
12.     persistent INFO;
13.     if isempty(INFO), INFO = get_info; end
14.     if nargin == 0
15.         fig = nnplots.find_training_plot(mfilename);
16.         if nargout > 0
17.             out1 = fig;
18.         elseif ~isempty(fig)
19.             figure(fig);
20.         end
21.         return;
22.     end
23.     in1 = varargin{1};
24.     if ischar(in1)
25.         switch in1
26.             case 'info',
27.                 out1 = INFO;
28.             case 'data_suitable'
29.                 data = varargin{2};
30.                 out1 = nnet.train.isNotParallelData(data);
31.             case 'suitable'
32.                 [args,param] = nnparam.extract_param(varargin,
                    INFO.defaultParam);
33.                 [net,tr,signals] = deal(args{2:end});
34.                 update_args = standard_args(net,tr,signals);
35.                 unsuitable = unsuitable_to_plot(param,update_a
                    rgs{:});
36.                 if nargout > 0
37.                     out1 = unsuitable;
38.                 elseif ~isempty(unsuitable)
39.                     for i=1:length(unsuitable)

```



```

40.             disp(unsuitable{i});
41.         end
42.     end
43.     case 'training_suitable'
44.         [net,tr,signals,param] = deal(varargin{2:end})
45.         ;
46.         update_args = training_args(net,tr,signals,param);
47.         unsuitable = unsuitable_to_plot(param,update_a
48.             rgs{:});
49.         if nargin > 0
50.             out1 = unsuitable;
51.         elseif ~isempty(unsuitable)
52.             for i=1:length(unsuitable)
53.                 disp(unsuitable{i});
54.             end
55.         end
56.     case 'training'
57.         [net,tr,signals,param] = deal(varargin{2:end})
58.         ;
59.         update_args = training_args(net,tr,signals);
60.         fig = nnplots.find_training_plot(mfilename);
61.         if isempty(fig)
62.             fig = figure('Visible','off','Tag',['TRAININ
63.                 G_' upper(mfilename)]);
64.             plotData = setup_figure(fig,INFO,true);
65.         else
66.             plotData = get(fig,'UserData');
67.         end
68.         set_busy(fig);
69.         unsuitable = unsuitable_to_plot(param,update_a
70.             rgs{:});
71.         if isempty(unsuitable)
72.             set(0,'CurrentFigure',fig);
73.             plotData = update_plot(param,fig,plotData,up
74.                 date_args{:});
75.             update_training_title(fig,INFO,tr)
76.             nnplots.enable_plot(plotData);
77.         else
78.             nnplots.disable_plot(plotData,unsuitable);
79.         end
80.         fig = unset_busy(fig,plotData);
81.         if nargin > 0, out1 = fig; end

```

```

76.         case 'close_request'
77.             fig = nnplots.find_training_plot(mfilename);
78.             if ~isempty(fig),close_request(fig); end
79.         case 'check_param'
80.             out1 = ''; % TODO
81.         otherwise,
82.             try
83.                 out1 = eval(['INFO.' in1]);
84.                 catch me, nnerr.throw(['Unrecognized first arg
            ument: '' in1 '''])
85.             end
86.         end
87.     else
88.         [args,param] = nnparam.extract_param(varargin,INFO
            .defaultParam);
89.         update_args = standard_args(args{:});
90.         if ischar(update_args)
91.             nnerr.throw(update_args);
92.         end
93.         [plotData,fig] = setup_figure([],INFO,false);
94.         unsuitable = unsuitable_to_plot(param,update_args{
            :});
95.         if isempty(unsuitable)
96.             plotData = update_plot(param,fig,plotData,update
            _args{:});
97.             nnplots.enable_plot(plotData);
98.         else
99.             nnplots.disable_plot(plotData,unsuitable);
100.        end
101.        set(fig,'Visible','on');
102.        drawnow;
103.        if nargin > 0, out1 = fig; end
104.    end
105. end
106.
107. function set_busy(fig)
108.     set(fig,'UserData','BUSY');
109. end
110.
111. function close_request(fig)
112.     ud = get(fig,'UserData');
113.     if ischar(ud)
114.         set(fig,'UserData','CLOSE');

```

```

115.     else
116.         delete(fig);
117.     end
118.     drawnow;
119. end
120.
121. function fig = unset_busy(fig,plotData)
122.     ud = get(fig,'UserData');
123.     if ischar(ud) && strcmp(ud,'CLOSE')
124.         delete(fig);
125.         fig = [];
126.     else
127.         set(fig,'UserData',plotData);
128.     end
129.     drawnow;
130. end
131.
132. function tag = new_tag
133.     tagnum = 1;
134.     while true
135.         tag = [upper(mfilename) num2str(tagnum)];
136.         fig = nnplots.find_plot(tag);
137.         if isempty(fig), return; end
138.         tagnumtagnum = tagnum+1;
139.     end
140. end
141.
142. function [plotData,fig] = setup_figure(fig,info,isTraining)
143.     PTFS = nnplots.title_font_size;
144.     if isempty(fig)
145.         fig = get(0,'CurrentFigure');
146.         if isempty(fig) || strcmp(get(fig,'NextPlot'),'new
')
147.             if isTraining
148.                 tag = ['TRAINING_' upper(mfilename)];
149.             else
150.                 tag = new_tag;
151.             end
152.             fig = figure('Visible','off','Tag',tag);
153.             if isTraining
154.                 set(fig,'CloseRequestFcn',[mfilename '('close
_request''')]);

```

```

155.         end
156.     else
157.         clf(fig);
158.         set(fig,'Tag','');
159.         set(fig,'Tag',new_tag);
160.     end
161. end
162. set(0,'CurrentFigure',fig);
163. ws = warning('off','MATLAB:Figure:SetPosition');
164. plotData = setup_plot(fig);
165. warning(ws);
166. if isTraining
167.     set(fig,'NextPlot','new');
168.     update_training_title(fig,info,[]);
169. else
170.     set(fig,'NextPlot','replace');
171.     set(fig,'Name',[info.name ' ( ' mfilename ')']);
172. end
173. set(fig,'NumberTitle','off','ToolBar','none');
174. plotData.CONTROL.text = uicontrol('Parent',fig,'Style',...
175.     'text',...
176.     'Units','normalized','Position',[0 0 1 1],'FontSize',PTFS,...
177.     'FontWeight','bold','ForegroundColor',[0.7 0 0]);
178. set(fig,'UserData',plotData);
179. end
180. function update_training_title(fig,info,tr)
181.     if isempty(tr)
182.         epochs = '0';
183.         stop = '';
184.     else
185.         epochs = num2str(tr.num_epochs);
186.         if isempty(tr.stop)
187.             stop = '';
188.         else
189.             stop = [', ' tr.stop];
190.         end
191.     end
192. set(fig,'Name',['Neural Network Training ' ...
193.     info.name ' ( ' mfilename '), Epoch ' epochs stop])
;

```

```

194.     end
195.
196.     % BOILERPLATE_END
197.     %% =====
    ====
198.
199.     function info = get_info
200.         info = nnfcnPlot(mfilename,'Time-
    Series Response',7.0,[...
201.             nnetParamInfo('outputIndex','Output Index','nntype.p
    os_int_scalar',1,...
202.                 'Index of output/target element to plot. '), ...
203.             nnetParamInfo('sampleIndex','Sample Index','nntype.p
    os_int_scalar',1,...
204.                 'Index of the time-
    series sample to plot. '), ...
205.         ]);
206.     end
207.
208.     function args = training_args(net,tr,data)
209.         y = nncalc.y(net,data.X,data.Xi,data.Ai);
210.         tt = {gmultiply(data.train.mask,data.T)};
211.         names = {'Training'};
212.         if ~isempty(data.val.indices)
213.             tt = [tt {gmultiply(data.val.mask,data.T)}];
214.             names = [names {'Validation'}];
215.         end
216.         if ~isempty(data.test.indices)
217.             tt = [tt {gmultiply(data.test.mask,data.T)}];
218.             names = [names {'Test'}];
219.         end
220.         args = {y tt names net.sampleTime};
221.     end
222.
223.     function args = standard_args(varargin)
224.         if nargin < 2
225.             args = 'Not enough input arguments.';
226.             return;
227.         end
228.         if nargin > 2
229.             z = varargin{end};
230.             if nntype.pos_int_scalar('isa',z)

```

```

231.         sampleTime = z;
232.         varargin(end) = [];
233.     else
234.         sampleTime = 1;
235.     end
236. end
237. if nargin == 2
238.     % plotresponse(t,y)
239.     t = nntype.data('format',varargin{1});
240.     y = nntype.data('format',varargin{2});
241.     if nargin < 3, sampleTime = 1; else sampleTime = v
arargin{3}; end
242.     err = nntype.data('check',y);
243.     if ~isempty(err),nnerr.throw(nnerr.value(err,'Outp
uts')); end
244.     err = nntype.data('check',t);
245.     if ~isempty(err),nnerr.throw(nnerr.value(err,'Targ
ets')); end
246.     args = {y {t} {''} sampleTime};
247.     else
248.     y = nntype.data('format',varargin{end});
249.     varargin(end) = [];
250.     count = length(varargin)/2;
251.     if length(varargin) ~= (count*2)
252.         error(message('nnet:Args:IncorrectNum'));
253.     end
254.     tt = cell(1,count);
255.     names = cell(1,count);
256.     for i=1:count
257.         tt{i} = nntype.data('format',varargin{i*2-1});
258.         names{i} = nntype.string('format',varargin{i*2})
;
259.     end
260.     args = {y,tt,names,sampleTime};
261. end
262. end
263.
264. function plotData = setup_plot(fig)
265.     PTFS = nnplots.title_font_size;
266.     plotData.numSignals = 0;
267.     plotData.axis1 = subplot(2,1,1);
268.     plotData.axis2 = subplot(2,1,2);
269.     pos1 = get(plotData.axis1,'Position');

```

```

270.     pos2 = get(plotData.axis2,'Position');
271.
272.     bottom = pos2(2);
273.     top = pos1(2) + pos1(4);
274.     totalHeight = top - bottom;
275.     topHeight = totalHeight * 0.70;
276.     middleHeight = totalHeight * 0.05;
277.     bottomHeight = totalHeight - middleHeight - topHeight;
278.     pos1(2) = bottom + bottomHeight + middleHeight;
279.     pos1(4) = topHeight;
280.     pos2(4) = bottomHeight;
281.
282.     set(plotData.axis1,'Position',pos1);
283.     set(plotData.axis2,'Position',pos2);
284.
285.     set(plotData.axis1,'Box','on',...
286.         'XTickLabelMode','manual','XTickLabel',{})
287.     plotData.title1 = title(plotData.axis1,'Time-
    Series Response','FontWeight','bold','FontSize',PTFS);
288.     plotData.ylabel1 = ylabel(plotData.axis1,'Output and
    Target','FontWeight','bold','FontSize',PTFS);
289.
290.     set(gca,'Box','on')
291.     plotData.ylabel2 = ylabel(plotData.axis2,'Error','Font
    Weight','bold','FontSize',PTFS);
292.     plotData.xlabel2 = xlabel(plotData.axis2,'Time','Font
    Weight','bold','FontSize',PTFS);
293.
294.     windowSize = [700 500];
295.     screenSize = get(0,'ScreenSize');
296.     screenSizeScreenSize = screenSize(3:4);
297.     pos = [(screenSize-windowSize)/2 windowSize];
298.     set(fig,'Position',pos);
299. end
300.
301. function fail = unsuitable_to_plot(param,y,tt,names,sampleTime)
302.     fail = '';
303.     t1 = tt{1};
304.     if numsamples(t1) == 0
305.         fail = 'The target data has no samples to plot.';

```

```

306.     elseif sum(numelements(t1)) == 0
307.         fail = 'The target data has no elements to plot.';

308.     elseif numtimesteps(t1) == 0
309.         fail = 'The input data has no timesteps to plot.';

310.     elseif numelements(t1) < param.outputIndex
311.         fail = {...
312.             sprintf('Output Index is out of range: %g',param
313.                 .outputIndex),'',...
314.             sprintf('The target data only has %g elements.',
315.                 numelements(t1))};
316.     elseif numsamples(t1) < param.outputIndex
317.         fail = {...
318.             sprintf('Sample Index is out of range: %g',param
319.                 .outputIndex),'',...
320.             sprintf('The target data only has %g elements.',
321.                 numelements(t1))};
322.     end
323. end
324.
325. function plotData = update_plot(param,fig,plotData,yy,
326.     tt,names,sampleTime)
327.     numSignals = length(names);
328.     if (plotData.numSignals ~= numSignals)
329.         plotData.numSignals = numSignals;
330.         cla(plotData.axis1);
331.         hold(plotData.axis1,'on');
332.         errorColor = [1 0.6 0];
333.         fitColor = [0 0 0];
334.         colors = {[0 0 1],[0 0.8 0],[1 0 0],[1 1 1]*0.5};

335.         plotplotData.errorLine = plot(plotData.axis1,[NaN
336.             NaN],[NaN NaN],'LineWidth',2,...
337.             'Color',errorColor);
338.         plotplotData.fitLine = plot(plotData.axis1,[NaN Na
339.             N],[NaN NaN],'LineWidth',1,...
340.             'Color',fitColor);
341.         plotData.targetLines = zeros(1,numSignals);
342.         plotData.outputLines = zeros(1,numSignals);
343.         targetLegends = cell(1,numSignals);
344.         outputLegends = cell(1,numSignals);
345.         for i=1:numSignals

```



```

339.         c = colors{min(i,4)};
340.         plotData.targetLines(i) = plot(plotData.axis1,[N
aN NaN],[NaN NaN],'.',...
341.         'LineWidth',1.5,'Color',c);
342.         plotData.outputLines(i) = plot(plotData.axis1,[N
aN NaN],[NaN NaN],'+',...
343.         'MarkerSize',6,'LineWidth',1,'Color',c);
344.         if ~isempty(names{1})
345.             targetLegends{i} = [names{i} ' Targets'];
346.             outputLegends{i} = [names{i} ' Outputs'];
347.         else
348.             targetLegends{i} = 'Targets';
349.             outputLegends{i} = 'Outputs';
350.         end
351.     end
352.     legend(plotData.axis1,[interleave(plotData.targetL
ines,plotData.outputLines),...
353.     plotData.errorLine,plotData.fitLine], ...
354.     [interleave(targetLegends, outputLegends),{'Erro
rs','Response'}]);
355.
356.     cla(plotData.axis2);
357.     hold(plotData.axis2,'on')
358.     plotData.baseLine2 = plot(plotData.axis2,[NaN
NaN],[NaN NaN],'k');
359.     plotData.errorLines2 = cell(1,numSignals);
360.     plotData.errorPoints2 = cell(1,numSignals);
361.     for i=1:numSignals
362.         c = colors{min(i,4)};
363.         plotData.errorLines2{i} = plot(plotData.axis2,[N
aN NaN],[NaN NaN],'LineWidth',2,...
364.         'Color',errorColor);
365.         plotData.errorPoints2{i} = plot(plotData.axis2,[
NaN NaN],[NaN NaN],'.',...
366.         'LineWidth',1.5,'Color',c);
367.     end
368. end
369.
370.     TIME = cell(1,numSignals);
371.
372.     Y = cell(1,numSignals);
373.     T = cell(1,numSignals);
374.     yy = nnfast.getsamples(yy,param.sampleIndex);

```

```

375.     yy = nnfast.getelements(yy,param.outputIndex);
376.     yy = cell2mat(yy);
377.
378.     for i=1:numSignals
379.         t = nnfast.getsamples(tt{i},param.sampleIndex);
380.         t = getelements(t,param.outputIndex);
381.         t = cell2mat(t);
382.         y = yy;
383.
384.         fprintf('y %d \n',y);
385.         fprintf('t %d \n',t);
386.         t2 = []; y2 = []; count = 1;
387.         for j=1: numel(t)
388.             if t(1,j) == 4 || t(1,j) == 12 || t(1,j) == 17
389.                 t2(1,count) = t(1,j);
390.                 y2(1,count) = y(1,j);
391.                 countcount = count +1;
392.             end
393.         end
394.
395.         fprintf('t2 %d \n',t2);
396.
397.         TS = size(t2,2);
398.         time = (1:TS)*sampleTime;
399.
400.         nani = find(isnan(t2));
401.         t2(nani) = [];
402.         y2(nani) = [];
403.         time(nani) = [];
404.
405.         TIME{i} = time;
406.         T{i} = t2;
407.         Y{i} = y2;
408.         set(plotData.outputLines(i),'XData',time,'YData',y
2);
409.         set(plotData.targetLines(i),'XData',time,'YData',t
2);
410.
411.         fprintf('y2 %d \n',numel(y2));
412.
413.         e = t2-y2;
414.         q = length(e);

```

```

415.         ydata = reshape([e; zeros(1,q); nan(1,q)],1,q*3);
416.         xdata = reshape([time; time; nan(1,q)],1,q*3);
417.         %     fprintf('xdata %d \n',xdata);
418.         %     fprintf('ydata %d \n',ydata);
419.         set(plotData.errorLines2{i},'XData',xdata,'YData',
            ydata);
420.         xdata = time; %[time time];
421.         ydata = e; %[e zeros(1,q)];
422.         set(plotData.errorPoints2{i},'XData',xdata,'YData'
            ,ydata);
423.         end
424.         TIME = [TIME{:}];
425.         Y = [Y{:}];
426.         T = [T{:}];
427.         time = (1:TS)*sampleTime;
428.         set(plotData.fitLine,'XData',time,'YData',y2);
429.
430.         %     fprintf('y %d \n',numel(yy));
431.
432.         numPoints = length(Y);
433.         spaces = nan(1,numPoints);
434.         TIME = [TIME; TIME; spaces];
435.         Y = [Y; T; spaces];
436.         TIMETIME = TIME(:)';
437.         YY = Y(:)';
438.         xlim = minmax(TIME);
439.         %fprintf('T %d \n',T);
440.         %fprintf('Y %d \n',Y);
441.         if (xlim(1) == xlim(2)), xlimxlim = xlim + [-
            1 1]; end
442.         set(plotData.errorLine,'XData',TIME,'YData',Y);
443.         set(plotData.axis1,'XLim',xlim);
444.         set(plotData.axis1,'YLimMode','auto');
445.         set(plotData.title1,'String',...
446.             sprintf('Response of Output Element %g for Time-
            Series %g',...
447.                 param.outputIndex,param.sampleIndex));
448.
449.         set(plotData.axis2,'XLim',xlim);
450.         set(plotData.axis2,'YLimMode','auto');
451.         set(plotData.baseLine2,'XData',xlim,'YData',[0 0]);

```

```
452.     legend(plotData.axis2,plotData.errorPoints2{1}, 'Targets - Outputs');
453.     end
454.
455.     % SUPPORT
456.
457.     function y = interleave(a,b)
458.         y = reshape([a;b],1,2*length(a));
459.     end
```

REFERENCES.

- [1] L. J. Tobin, V. Saurabh, "We've looked at Clouds from Both Sides Now," SRII Global Conference (SRII), 2011 Annual, pp.342-348, doi: 10.1109/SRII.2011.46
- [2] N. Sang-Ho, H. Eui-Nam, "A methodology of Assessing Security Risk of Cloud Computing in User Perspective for Security-Service-Level Agreements," Innovative Computing Technology (INTECH), IEEE, 2014 Fourth International Conference, pp.87-92, doi: 10.1109/INTECH.2014.6927759
- [3] Lifeng Wang; Zhengping Wu, "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection," in Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on , vol., no., pp.101-106, 15-18 Dec. 2014, doi: 10.1109/CloudCom.2014.107
- [4] Li, W.; Yang, Y.; Yuan, D., "Ensuring Cloud data reliability with minimum replication by proactive replica checking," in Computers, IEEE Transactions on , vol.PP, no.99, pp.1-1, doi: 10.1109/TC.2015.2451644
- [5] Zhengping Wu, "Multi-cloud policy enforcement through semantic modeling and mapping," in Semantic Computing (ICSC), 2015 IEEE International Conference on , vol., no., pp.448-451, 7-9 Feb. 2015, doi: 10.1109/ICOSC.2015.7050849

- [6] Zhengping Wu; Nailu Chu; Peng Su, "Improving Cloud Service Reliability -- A System Accounting Approach," in Services Computing (SCC), 2012 IEEE Ninth International Conference on , vol., no., pp.90-97, 24-29 June 2012, doi: 10.1109/SCC.2012.33
- [7] Farhad Ahamed, Seyed Shahrestani and Athula Ginige, "Cloud Computing: Security and Reliability Issues," Communications of the IBIMA, vol. 2013, Article ID 655710, 12 pages, doi: 10.5171/2013.655710
- [8] V. K. Upadhyay, V. L. Desai, "Cloud Computing Security Issue And Barriers," International Journal of Modern Trends in Engineering and Research (IJMTER), Volume 02, Issue 03, [March - 2015] e-ISSN: 2349-9745, p-ISSN: 2393-8161
- [9] A. Ghobadi, R. Karimi, F. Heidari and M. Samadi, "Cloud computing, reliability and security issue," Advanced Communication Technology (ICACT), 2014 16th International Conference on, Pyeongchang, 2014, pp. 504-511, doi: 10.1109/ICACT.2014.6779012
- [10] O. Beaumont, P. Duchon and P. Renaud-Goud, "Approximation algorithms for energy minimization in Cloud service allocation under reliability constraints," *20th Annual International Conference on High Performance Computing*, Bangalore, 2013, pp. 295-304. doi: 10.1109/HiPC.2013.6799123
- [11] M. I. M. Almanea, "A Survey and Evaluation of the Existing Tools that Support Adoption of Cloud Computing and Selection of Trustworthy and

Transparent Cloud Providers," Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on, Salerno, 2014, pp. 628-634. doi: 10.1109/INCoS.2014.42

- [12] Cloud Security Alliance. About the CSA Security, Trust & Assurance Registry (STAR). [Online] Cloud Security Alliance <https://cloudsecurityalliance.org/star/> [Accessed: November 10, 2015]
- [13] Cloud Security Alliance. Cloud Controls Matrix (CCM). [Online] Cloud Security Alliance <https://cloudsecurityalliance.org/research/ccm> [Accessed: November 25, 2015]
- [14] European Network Information Security Agency. [Online] ENISA <https://cloudsecurityalliance.org/star/> [Accessed: December 5, 2015]
- [15] Michael Sheng, Talal H Noor, Abdullah Alfazi, Jingning Lin and Jeriel Law, "Trust Feedback Dataset," [Online] Cloud Armor <http://cs.adelaide.edu.au/~cloudarmor/ds.html> [Accessed: October 25, 2015]
- [16] F. Díaz-del-Río, J. Salmerón-García and J. L. Sevillano, "Extending Amdahl's Law for the Cloud Computing Era," in *Computer*, vol. 49, no. 2, pp. 14-22, Feb. 2016. doi: 10.1109/MC.2016.49
- [17] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems* 2 (1989), no. 4, 303–314, Feb. 1989, p-ISSN: 0932-4194

- [18] SPEC RG Cloud Working Group, Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics. (2016). Technical Report: SPEC-RG-2016-01. Retrieved from <https://arxiv.org/pdf/1604.03470.pdf>
- [19] W. v. d. Aalst and E. Damiani, "Processes Meet Big Data: Connecting Data Science with Process Science," in IEEE Transactions on Services Computing, vol. 8, no. 6, pp. 810-819, Nov.-Dec. 1 2015. doi: 10.1109/TSC.2015.2493732
- [20] W. M. P. van der Aalst, W. Z. Low, M. T. Wynn and A. H. M. ter Hofstede, "Change your history: Learning from event logs to improve processes," 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Calabria, 2015, pp. 7-12. doi: 10.1109/CSCWD.2015.7230925
- [21] X. Ao, P. Luo, C. Li, F. Zhuang and Q. He, "Online Frequent Episode Mining," 2015 IEEE 31st International Conference on Data Engineering, Seoul, 2015, pp. 891-902. doi: 10.1109/ICDE.2015.7113342
- [22] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009
- [23] Wu Chenkang, Zhu Yonghua, Pan Shunhong, "The SLA Evaluation Model for Cloud Computing", Proc. International Conference on Computer

Networks and Communication Engineering (ICCNCE 2013), pp. 331-334,
May. 2013. doi: 10.2991/iccnce.2013.83

- [24] M. H. Ghahramani, M. Zhou and C. T. Hon, "Toward cloud computing QoS architecture: analysis of cloud systems and cloud services," in IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 1, pp. 6-18, Jan. 2017. doi: 10.1109/JAS.2017.7510313
- [25] A. Kovari and P. Dukan, "KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE," 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, Subotica, 2012, pp. 335-339. doi: 10.1109/SISY.2012.6339540
- [26] D. Qingfeng, L. Huan, Y. Kanglin and Q. Juan, "VM Reliability Modeling and Analysis for IaaS Cloud," 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Nanjing, China, 2017, pp. 258-265. doi: 10.1109/CyberC.2017.77
- [27] A. Stanik, M. Hovestadt and O. Kao, "Hardware as a Service (HaaS): The completion of the cloud stack," 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), Seoul, Korea (South), 2012, pp. 830-835. p-ISSN: 978-1-4673-0893-9
- [28] Mathworks. (2017). Neural Network Toolbox: User's Guide (r2017b). Retrieved December 08, 2017 from www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf

- [29] D. Svozila, V. Kvasnickab, J. Pospichalb, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43-62, Nov. 1997. doi: 10.1016/S0169-7439(97)00061-0
- [30] Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach* (2 ed.). Pearson Education. ISBN: 0137903952
- [31] J. Sahoo, S. Mohapatra and R. Lath, "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues," 2010 Second International Conference on Computer and Network Technology, Bangkok, 2010, pp. 222-226. doi: 10.1109/ICCNT.2010.49
- [32] Wei Jing, Nan Guan and Wang Yi, "Performance isolation for real-time systems with Xen hypervisor on multi-cores," 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications, Chongqing, 2014, pp. 1-7. doi: 10.1109/RTCSA.2014.6910557
- [33] G. J. Popek, R. P. Goldberg. "Formal requirements for virtualizable third generation architectures," *Commun. ACM* 17, pp. 412-421, Jul. 1974. doi: <http://dx.doi.org/10.1145/361011.361073>
- [34] Xen Project Documentation (2017, October 28). Retrieved from <https://www.xenproject.org/help/documentation.html>
- [35] User Manual (2018), Oracle VM VirtualBox User Manual. Retrieved from <https://www.virtualbox.org/manual/>

- [36] L. Zeng, Y. Xiao and H. Chen, "Accountable logging in operating systems," 2015 IEEE International Conference on Communications (ICC), London, 2015, pp. 7163-7167. doi: 10.1109/ICC.2015.7249469
- [37] Windows Events (2016, October 22). In Windows Desktop app technologies. Retrieved from [https://msdn.microsoft.com/en-us/library/windows/desktop/aa964766\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa964766(v=vs.85).aspx)
- [38] C. Simache, M. Kaaniche and A. Saidane, "Event log based dependability analysis of Windows NT and 2K systems," 2002 Pacific Rim International Symposium on Dependable Computing, 2002. Proceedings., 2002, pp. 311-315. doi: 10.1109/PRDC.2002.1185651
- [39] R. Gerhards, "The Syslog Protocol," RFC 5424, Mar. 2009. doi 10.17487/RFC5424
- [40] A. Reynaldi, S. Lukas and H. Margaretha, "Backpropagation and Levenberg-Marquardt Algorithm for Training Finite Element Neural Network," 2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation, Valetta, 2012, pp. 89-94. doi: 10.1109/EMS.2012.56