



Intelligent Management of Virtualised Computer Based Workloads and Systems

Submitted for the Degree of
Doctor of Philosophy
At the University of Northampton

2020

James Daniel Oakes

© James Oakes 2020.

This thesis is copyright material and no quotation from it may be published without proper acknowledgement.

Acknowledgements

I would like to acknowledge the support of my supervisory team, Dr Mark Johnson, Dr James Xue and Dr Scott Turner for their invaluable advice and guidance throughout my research degree. I dedicate this work to my three wonderful children, Joshua, Jacob and Olivia.

Abstract

Managing the complexity within virtualised IT infrastructure platforms is a common problem for many organisations today. Computer systems are often highly consolidated into a relatively small physical footprint compared with previous decades prior to late 2000s, so much thought, planning and control is necessary to effectively operate such systems within the enterprise computing space. With the development of private, hybrid and public cloud utility computing this has become even more relevant; this work examines how such cloud systems are using virtualisation technology and embedded software to leverage advantages, and it uses a fresh approach of developing and creating an Intelligent decision engine (expert system). Its aim is to help reduce the complexity of managing virtualised computer-based platforms, through tight integration, high-levels of automation to minimise human inputs, errors, and enforce standards and consistency, in order to achieve better management and control. The thesis investigates whether an expert system known as the Intelligent Decision Engine (IDE) could aid the management of virtualised computer-based platforms. Through conducting a series of mixed quantitative and qualitative experiments in the areas of research, the initial findings and evaluation are presented in detail, using repeatable and observable processes and provide detailed analysis on the recorded outputs. The results of the investigation establish the advantages of using the IDE (expert system) to achieve the goal of reducing the complexity of managing virtualised computer-based platforms. In each detailed area examined, it is demonstrated how using a global management approach in combination with VM provisioning, migration, failover, and system resource controls can create a powerful autonomous system.

Abbreviations

AI	Artificial Intelligence
AMI	Amazon Machine Image
CLR	Cognitive Load Rating
CLT	Cognitive Load Theory
DaaS	Database as a Service
DNS	Domain Name Service
ESX	Elastic Sky X (VMware hypervisor)
FC	Fibre Channel
Gb	Gigabits
GB	Gigabytes
GHz	Gigahertz
GNU	Unix like operating system (free software foundation)
HA	High Availability
Hyperv	Microsoft's Hypervisor
ICMP	Internet Control Message Protocol
I/O	Input/Output
i86pc	8086 intel/AMD architecture
IaaS	Infrastructure as a Service
IDE	Intelligent Decision/Design Engine
IEEE	Institute of Engineering Electrical and Electronics
IOPS	Input/Output Operations per second
IoT	Internet of Things
iSCSI	Internet Small Computer Systems Interface
JSON	JavaScript Object Notation
KBS	Knowledge Based System
KVM	Kernel-based Virtual Machine

LACP	Link Aggregation Control Protocol defined by IEEE 802.1ax
LAMP	Linux Apache MySQL PHP
LAN	Local Area Network
LDOM	Logical Domain
LOM	Lights Out Management
LUN	Logical Unit
Mb	Megabits
MTBF	Meantime Between Failure
NAS	Network Attached Storage
NFS	Network Filesystem
NGZ	Non-Global Zone, Solaris container
N+1	Indicates a resilient backup component is available
OEM	Oracle Enterprise Manager
OOC	Oracle Ops Center, formally Sun Management Center
OS	Operating System
OVM	Oracle Virtual Machine Server
PaaS	Platform as a Service
PXE	Pre-boot Execution Environment
RAM	Random Access Memory
RDBMS	Relational Database Management System
RPM	Redhat Package Manager
SaaS	Software as a Service
SAIL	Stanford Artificial Intelligence Language
SAMP	Solaris Apache MySQL PHP
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial AT Attachment

SCSI	Small Computer Systems Interface
Solaris OS	Sun Microsystems Solaris Operating System (now Oracle)
SPARC	Scalable Processor Architecture
SPoF	Single Point of Failure
SSH	Secure Shell
sun4u	SPARC-Enterprise SPARC V9 Unix Kernel
sun4v	SPARCV9 Unix Kernel
SunOS	UNIX System V, now known as Solaris
TCP/IP	Transmission Control Protocol / Internet Protocol
VAX	Computers manufactured by Digital Equipment Corporation
VirtualBox	Innotek/Sun's type II Hypervisor now owned by Oracle Corporation
VIP	Virtual IP
VLAN	Virtual LAN
VLAN Tagging	VLAN encapsulation defined by IEEE 802.1Q
VM	Virtual Machine
VMDK	Virtual Machine Disk
vNAS	Virtual Network Attached Storage
x86	Intel 8086 CPU Compatible Architectures
Xen	Xen Hypervisor, Open-source virtualisation project
YAML	YAML Ain't Markup Language
ZFS	Zetabyte Filesystem
Zpool	A ZFS dataset / pool

Nomenclature

Greek and Latin Script, Letters & Maths Symbols

- **Relating to Cluster Quorum**

η denotes a cluster node that is available or unavailable
 ϵ denotes a cluster node that is unavailable
 T denotes the total number of cluster node votes possible
 v denotes the number of cluster node votes currently available
 ϖ denotes the minimum number of votes needed to establish a quorum
 ϱ denotes the ability to establish a cluster quorum

- **Relating to Cognitive Load Rating**

β denotes the Cognitive Load Rating (CLR) for one task
 Δ denotes the Task Complexity
 \emptyset denotes the Process Mechanism
 λ denotes the CLR for a set (sum) of tasks
 n denotes the number of tasks
 t denotes the task identifier
 K denotes the CLR mean average for a set of tasks

- **Relating to User Task Complexity**

R denotes the derived result
 μ denotes user input
 s denotes a simple task
 m denotes a moderate task
 d denotes a difficult task
 x denotes a manual task
 y denotes a semi-automatic task
 z denotes an automatic task

- **Relating to VM provisioning Timing**

\mathcal{T} denotes the total time to deploy a VM
 n denotes the task number
 t denotes the task identifier
 θ denotes the participant time taken to complete a task (in seconds)
 ψ denotes the average (mean) participant time taken per task (in seconds)

Table of Contents

CHAPTER 1: INTRODUCTION	20
1.1 Background.....	20
1.1.1 Enterprise Computer Virtualisation	21
1.1.2 Type I Hypervisors: Advantages and Disadvantages.....	24
1.1.3 Type II Hypervisors: Advantages and Disadvantages.....	24
1.1.4 Cloud Computing (Public, Hybrid and Private)	24
1.1.5 Common Virtualisation Problems.....	28
1.2 Thesis Motivation and Aims.....	29
1.3 Thesis Benefits and Targeted Applications	30
1.4 Thesis Limitations	31
1.5 Thesis Summary.....	34
CHAPTER 2: EXPERT, CLOUD AND VIRTUALISED SYSTEMS	36
2.1 Introduction.....	36
2.2 Intelligent Organisation	37
2.3 The Origins of Artificial Intelligence	38
2.4 Expert System Applications.....	39
2.4.1 Introduction	39
2.4.2 R1/XCON	39
2.4.3 MYCIN	40
2.4.4 INTERNIST-I	41
2.4.5 DENDRAL (DENDritic ALgorithm).....	41
2.4.6 HEARSAY I and II	42
2.4.7 MACSYMA (MAC's SYmbolic MANipulator).....	42
2.4.8 PROSPECTOR	42
2.4.9 Expert Systems: Why Have They Been Considered?.....	43
2.5 Public Cloud Systems	43
2.5.1 Introduction	43
2.5.2 Case Study 1: Amazon EC2	44
2.5.3 Case Study 2: Oracle Cloud	47
2.5.4 Cloud Computing: How it Has Created Utility Based Computing?.....	49
2.6 Current Virtualisation and Cloud Management Approaches	50
2.6.1 Introduction	50
2.6.2 Reviewed Approaches	50
2.6.3 Conclusions	56

2.7 Intelligent System Approaches	61
2.7.1 Introduction	61
2.7.2 Algorithms.....	61
2.7.3 Text Mining	62
2.7.4 Natural Language Analysis	63
2.7.5 Inference Engine (Forward Chaining)	64
2.7.6 Cognitive Load Theory	65
2.8 Summary	66
2.8.1 Introduction	66
2.8.2 Gap Analysis.....	67
2.8.3 Approach Challenges	68
2.8.4 Justifications.....	70
CHAPTER 3: METHODOLOGY AND EVALUATION STRATEGY	71
3.1 Methodology Introduction	71
3.2 Development Framework	72
3.2.1 Laboratory Setup.....	72
3.2.2 Software Configuration.....	74
3.3 Evaluation Strategy	74
3.3.1 Evaluation Approaches	74
3.3.2 Expert System Evaluation.....	75
3.3.3 Experiment Design.....	77
3.4 Qualitative Versus Quantitative Methods	78
3.4.1 Qualitative Evaluation.....	78
3.4.2 Quantitative Evaluation	80
3.4.3 Data Analysis.....	82
3.5 Evaluation of Comparative Systems	83
3.5.1 Investigation 1: Autonomous VM Deployment	83
3.5.2 Investigation 2: Cognitive Complexity System Evaluation.....	85
3.5.3 Investigation 3: Workload Migration and Evacuation of VMs.....	87
3.5.4 Investigation 4: Overload of VM Memory Usage, Detection Time, and Resolution Time.....	89
3.5.5 Investigation 5: Overload of VM CPU usage, Detection Time, and Resolution Time.....	89
3.6 Summary	90
CHAPTER 4: THE INTELLIGENT DECISION ENGINE	92
4.1 Introducing the Intelligent Decision Engine	92
4.2. IDE Characteristics	93
4.2.1 Data Organisation.....	93
4.2.2 Decision Making	94
4.2.3 System Learning	94
4.2.4 Algorithms and Procedures.....	94

4.3 IDE Components	103
4.4 Defining the IDE Model	108
4.5 Data-storage, Memory and Information Retrieval	109
4.5.1 Long Term Storage Strategy	109
4.5.2 Short and Medium Term Storage Strategy.....	109
4.6 Data Processing and Organisation.....	109
4.6.1 Data flows Between Systems	110
4.6.2 Creating the Inference Engine	111
4.6.3 System Self-management and Learning.....	111
4.6.4 System Real-time and Source Data.....	112
4.7 System Availability and Autonomy	114
4.7.1 Establishing a Quorum	115
4.7.2 Command Zone Concept.....	117
4.7.3 Keep Alive Critical Processes	119
4.8 IDE Rule-base and Inference Engine	120
4.8.1 IDE Trigger Events.....	120
4.8.2 Physical System Events	120
4.8.3 VM System Events	121
4.8.4 Text Analysis.....	124
4.8.5 Knowledge Rule Justifications	127
4.9 Summary.....	133
CHAPTER 5: SIMPLIFIED DEPLOYMENT OF VIRTUAL MACHINES.....	134
5.1 Introduction.....	134
5.2 Simplified VM Provisioning	135
5.2.1 Experiment Process	135
5.2.1.1 Task Complexity Definition	135
5.2.1.2 User Types.....	137
5.2.1.3 Task Types.....	137
5.2.1.4 Process Types and Complexity Value Weightings.....	138
5.2.1.5 User Results: Mode Average of Task Complexity Description.....	139
5.2.1.6 VM Provisioning Process.....	140
5.2.1.7 Hardware Provisioning Platform	143
5.2.1.8 VM Sizing Methods.....	143
5.2.2 Experiment 1: VM Provisioning Timing Comparison.....	144
5.2.2.1 Formalisation	144
5.2.2.2 VM Provisioning Expert Users	145
5.2.2.3 VM Provisioning Experienced Users	145
5.2.2.4 VM Provisioning Novice Users.....	145
5.2.2.5 VM Provisioning Build Methods.....	146
5.2.2.6 Re-visiting the IDE Provisioning Experiment with Queued Pre-built System Images.....	146
5.2.3 Experiment 2: Cognitive Evaluation Performance	146

5.2.3.1 Converting Qualitative Data into Quantitative Data: Is This Possible?	147
5.2.3.2 Cognitive Experimental Process	148
5.2.3.3 Cognitive Load Rating Formula.....	150
5.2.3.4 User Task Complexity Formula.....	151
5.2.3.5 Cognitive Load Rating Chart	152
5.3 Results	152
5.3.1 VM Provisioning Timed Results	152
5.3.1.1 Expert Users	153
5.3.1.2 Experienced Users	155
5.3.1.2 Novice Users.....	156
5.3.2 Aggregated VM Provisioning Timed Results.....	158
5.3.2.1 Expert Users	158
5.3.2.2 Experienced Users	159
5.3.2.3 Novice Users.....	161
5.3.3 Cognitive Load Rating Results	163
5.3.4 Overall Results.....	164
5.4 Summary.....	165
CHAPTER 6: IMPROVING WORKLOAD MIGRATION STRATEGIES	167
6.1 Introduction	167
6.2 Workload Migration Methods.....	167
6.3 Experiment Process.....	168
6.4 Experiment 3: Workload Migration and Evacuation of VMs	169
6.4.1 Experiment 3.1: IDE VM Migration/failover Process	170
6.4.2 Experiment 3.2: vMotion VM Migration/failover Process	172
6.4.3 Experiment 3.3: vMotion and XenMotion VM Migration/failover Process.....	173
6.5 Results	175
6.6 Summary.....	179
CHAPTER 7: OPTIMISING PERFORMANCE AND AVAILABILITY OF VIRTUAL MACHINES	180
7.1 Introduction	180
7.2 Experiment Process.....	183
7.2.1 IDE VM Performance Algorithm	183
7.2.2 VirtualBox Memory Balloon Driver	184
7.2.3 Simulate VM CPU and Memory Stress.....	185
7.2.4 Characteristics Compared Against Other Studies.....	186
7.2.5 IDE Global Resource Management	189
7.2.6 Comparative Methods Analysis.....	189
7.3 Optimisation of System Performance and Availability	193

7.3.1 x86-64-bit Architectures and Memory Ballooning	193
7.3.2 x86-64-bit Architectures with CPU Hotplug Features	193
7.4 Experiment 4: Overload of VM Memory Usage, Detection Time, and Resolution Time	194
7.4.1 IDE VM Memory Ballooning Process	194
7.4.2 Study 1 VM Memory Balloon Process.....	194
7.4.3 Study 2 VM Memory Balloon Process.....	196
7.5 Experiment 5: Overload of VM CPU usage, Detection Time, and Resolution Time	197
7.5.1 IDE CPU Hotplug Process	197
7.5.2 Study 1 VM CPU Hotplug Process.....	198
7.5.3 Study 2 VM CPU Hotplug Process.....	198
7.6 Results	198
7.6.1 IDE Characteristics (VirtualBox Balloon)	200
7.6.2 Study 1 Characteristics (XenBalloon)	203
7.6.3 Study 2 Characteristics (iBalloon)	205
7.6.4 Platform Characteristic Scores (IDE, Study 1, Study 2)	207
7.6.5 Binomial Scores (IDE, Study 1, Study 2)	208
7.7 Summary.....	209
CHAPTER 8: CONTRIBUTION, CONCLUSIONS AND FURTHER WORK.....	211
8.1 Thesis Contribution.....	211
8.1.1 Development of an Expert System Framework for Virtualised Computer Systems	211
8.1.2 Simplified VM Provisioning	212
8.1.3 CLR formula to Determine Task Complexity.....	213
8.1.4 Efficient VM Migration, Evacuation and Restart Routines	213
8.1.5 Global Scheduling Mechanism for CPU Hot-plug and Memory Resource Management	213
8.1.6 Summary	214
8.2 Overall Results and Conclusions.....	214
8.2.1 Simplified VM Deployment Experiment Conclusions.....	214
8.2.2 Cognitive Evaluation Performance Experiment Conclusions.....	216
8.2.3 Workload Migration/Failover Experiment Conclusions	217
8.2.4 Performance and Availability (CPU & Memory Overload) Experiment Conclusions.....	217
8.2.5 Significance of Results.....	218
8.3 Future Work.....	222
8.3.1 Prebuilding and Queuing VMs.....	222
8.3.2 Development with Additional Operating Systems.....	223
8.3.3 VirtualBox Teleport Development	223
8.3.4 Quorum Cluster Node Testing.....	223
8.3.5 Bootstrap Development.....	223
8.3.6 Knowledge Rules	224
8.3.7 Self-Learning	224
8.3.8 Data Sources and Trigger Events	224
8.3.9 Laboratory Build for VMware, KVM and Xen Clusters	224
8.3.10 Knowledge Rule Testing with SLAs	225

8.3.11 Global Resource Management	225
8.3.12 Terraform, AWS CloudFormation and AMIs.....	225
REFERENCES.....	226
APPENDIX.....	237
APPENDIX A – VM DEPLOYMENT PROCESS.....	237
A.1 Expert Users Results	237
A.2 IDE Results	237
A.3 Oracle Results	237
A.4 AWS Results	237
A.5 Experienced Users Results	237
A.6 IDE Results	237
A.7 Oracle Results	238
A.8 AWS Results	238
A.9 Novice Users Results.....	238
A.10 IDE Results	238
A.11 Oracle Results	238
A.12 AWS Results	238
APPENDIX B – BLIND PEER REVIEW COMMENTS (PUBLISHED PAPERS).....	239
B.1 Simplified Deployment of Virtual Machines Using an Intelligent Design Engine (Oakes et al, 2016)	239
B.2 Blind Review 1.....	239
B.3 Blind Review 2.....	240
B.4 Measuring and Reducing the Cognitive Load for End Users of Complex Systems (Oakes et al, 2019)	242
B.5 Blind Review 1.....	242
B.6 Blind Review 2.....	243
APPENDIX C - VM PLATFORM BUILD PROCESS	245
C.1 IDE Provisioning.....	245
C.2 VM Deployment Steps.....	245
C.3 Oracle Cloud Provisioning	251
C.4 VM Deployment Steps.....	251
C.5 AWS Cloud Provisioning.....	256
C.6 VM Deployment Steps.....	256
APPENDIX D – IDE BUILD PROCEDURES.....	266

D.1 Procedure 1	266
D.2 Procedure 2	266
D.3 Procedure 3	267
D.4 Procedure 4	267
D.5 Source of Knowledge Rules	268
APPENDIX E – VM FAILOVER & MIGRATION	269
E.1 IDE Results	269
APPENDIX F – VM CPU AND MEMORY	269
F.1 IDE Results	269
APPENDIX G – ORIGINAL PROPOSAL	270
G.1 Aims & Objectives	270

List of Figures

Figure 1.1 Example of Virtualisation of x86 hardware (public domain image).....	22
Figure 1.2 Type I Hypervisor v Type II Hypervisors	23
Figure 1.3 Common Virtualisation Problems	29
Figure 2.1 Cloud Leaders Market Share (Source: Forbes, 2018)	44
Figure 2.2 Cloud Service Stack	49
Figure 2.3 The moSAIC Architecture	54
Figure 2.4 Machine Event Response Mechanism.....	63
Figure 2.5 Natural Learning Mechanism – Information Organisation.....	64
Figure 2.6 Machine Learning Mechanisms Event Response.....	65
Figure 3.1 Initial Laboratory Setup	73
Figure 3.2 Final Minimum Laboratory Setup	73
Figure 4.1 IDE Program Components	103
Figure 4.2 IDE Architecture Model	108
Figure 4.3 IDE Network Flows	110
Figure 4.4 IDE Trigger Events	120
Figure 4.5 IDE Example of Forward-chaining.....	123
Figure 5.1 Cognitive Load Rating Chart	152
Figure 5.2 IDE Timed VM Provisioning – Expert Users	153
Figure 5.3 Oracle Timed VM Provisioning – Expert Users	154
Figure 5.4 AWS Timed VM Provisioning – Expert Users	154
Figure 5.5 IDE Timed VM Provisioning – Experienced Users	155
Figure 5.6 Oracle Timed VM Provisioning – Experienced Users	155
Figure 5.7 AWS Timed VM Provisioning – Experienced Users	156
Figure 5.8 IDE Timed VM Provisioning – Novice Users.....	156
Figure 5.9 Oracle Timed VM Provisioning – Novice Users.....	157
Figure 5.10 AWS Timed VM Provisioning – Novice Users.....	157
Figure 5.11 IDE Aggregated Timed VM Provisioning – Expert Users.....	158
Figure 5.12 Oracle Aggregated Timed VM Provisioning – Expert Users.....	159
Figure 5.13 AWS Aggregated Timed VM Provisioning – Expert Users	159
Figure 5.14 IDE Aggregated Timed VM Provisioning – Experienced Users.....	160
Figure 5.15 Oracle Aggregated Timed VM Provisioning – Experienced Users.....	160
Figure 5.16 AWS Aggregated Timed VM Provisioning – Experienced Users	161
Figure 5.17 IDE Aggregated Timed VM Provisioning – Novice Users	161
Figure 5.18 Oracle Aggregated Timed VM Provisioning – Novice Users	162
Figure 5.19 AWS Aggregated Timed VM Provisioning – Novice Users.....	162
Figure 5.20 CLR VM Provisioning – Expert Users.	163
Figure 5.21 CLR VM Provisioning – Experienced Users.	163
Figure 5.22 CLR VM Provisioning – Novice Users.....	164
Figure 6.1 Experiment 3.1 VM Failover Method (IDE).....	171
Figure 6.2 Experiment 3.2 VM Failover Method (study 1)	172
Figure 6.3 Experiment 3.3 VM Failover Method study 2 (Feng et al, 2011)	174
Figure 6.4 IDE VM Failure Detection Time Experiment 3.1 (IDE)	176
Figure 6.5 IDE VM Failure Detection and Migration Time Experiment 3.1 (IDE)	176

Figure 6.6 Study 1 VM Failure Detection and Migration Time Experiment 3.2 (vMotion)..... 177

Figure 6.7 Study 2 VM Failure Detection and Migration Time Experiment 3.2 (vMotion)..... 177

Figure 6.8 Study 2 VM Failure Detection and Migration Time Experiment 3.2 (XenMotion)..... 178

Figure 6.9 Comparative Mean Average VM Migration Time for Experiments 3.1, 3.2 and 3.3..... 178

Figure 7.1 IDE VM Simulated Tests for Load Stress (using stress-ng)..... 186

Figure 7.2 IDE Global Resource Management 193

Figure 7.3 IDE Performance Monitoring and Memory Ballooning Results..... 194

Figure 7.4 Study 1 Xen Balloon Process (Zhang et al, 2017) 195

Figure 7.5 Study 1 VM Memory Balloon Process (Zhang et al, 2017) 196

Figure 7.6 Study 2 iBalloon system overview (Zhang et al, 2016)..... 197

Figure 7.7 IDE Performance Monitoring and CPU Hot-plug Results 198

Figure 7.8 Binomial System Characteristic Results 208

List of Tables

Table 2.1 Comparing AWS features.....	47
Table 2.2 Comparing Oracle features.....	49
Table 2.3 Current Virtualisation/Cloud Management Findings.....	61
Table 2.4 Gap Analysis.....	68
Table 2.5 Approach Challenges.....	69
Table 2.6 Research Justifications.....	70
Table 3.1 Qualitative Experiment Methods.....	80
Table 3.2 Quantitative Experiment Methods.....	82
Table 3.3 VM Deployment Experiment.....	85
Table 3.4 VM Deployment Cognitive Load Experiment.....	87
Table 3.5 VM Evacuation, Workload Migration and Load Management Experiment.....	88
Table 3.6 VM Memory Overload, Detection and Resolution Experiment.....	89
Table 3.7 VM CPU Overload, Detection and Resolution Experiment.....	90
Table 4.1 Algorithm/Procedure 1: Remote System Discovery.....	95
Table 4.2 Algorithm/Procedure 2: Messaging Command Process.....	96
Table 4.3 Algorithm/Procedure 3: Text Mining.....	96
Table 4.4 Algorithm/Procedure 4: Data Organisation.....	97
Table 4.5 Algorithm/Procedure 5: Pattern Analysis and Learning.....	98
Table 4.6 Algorithm/Procedure 6: Forward Chaining.....	99
Table 4.7 Algorithm/Procedure 7: VM Deployment.....	100
Table 4.8 Algorithm/Procedure 8: Preliminary Performance Monitoring.....	100
Table 4.9 Algorithm/Procedure 9: Event Trigger and Decision Making.....	101
Table 4.10 Algorithm/Procedure 10: Self-Monitoring.....	102
Table 4.11 IDE Program Function Suite.....	107
Table 4.12 IDE Rule Matching Process.....	114
Table 4.13 IDE Cluster Resource Evacuation.....	118
Table 4.14 Event Knowledge Rule: Physical host down.....	120
Table 4.15 Event Knowledge Rule: Physical Host Memory Capacity.....	121
Table 4.16 Event Knowledge Rule: Physical Host Memory CPU.....	121
Table 4.17 Event Knowledge Rule: Memory overload.....	122
Table 4.18 Event Knowledge Rule: CPU overload.....	122
Table 4.19 Event Knowledge Rule: VM Migration.....	123
Table 4.20 Event Knowledge Rule: VM Unresponsive.....	124
Table 4.21 Event Knowledge Rule: VM Evacuate.....	124
Table 4.22 Example of Keyword Pattern Analysis.....	125
Table 4.23 Example of Pattern Keyword Matching.....	126
Table 4.24 Knowledge Rules Justifications.....	133
Table 5.1 Task Complexity Rating.....	136
Table 5.2 End-User Types.....	137
Table 5.3 Process Mechanism Definition.....	138
Table 5.4 Process, Task, Sub-component Definitions.....	138
Table 5.5 VM Provisioning 10-Step Complexity (Mode Average).....	140

Table 5.6 VM Provisioning Sequence	143
Table 5.7 Allocated VM Compute Resource	143
Table 5.8 Similar Cognitive Load Studies (PaaS et al, 2003)	149
Table 5.9 VM Provisioning Experiment Results.....	164
Table 6.1 Simulated VM Failover/Migration IDE Preparation Steps	169
Table 6.2 Simulated VM Failover/Migration IDE Steps	169
Table 6.3 Experiment 3.1, Downtime and Total Migration Timed Results (IDE)	172
Table 6.4 Experiment 3.2, Downtime and Total Migration Results vMotion (Shirinbab et al, 2016)	173
Table 6.5 Experiment 3.3, Downtime and Total Migration Results vMotion (Feng et al, 2011).....	175
Table 6.6 Experiment 3.3, Downtime and Total Migration Results XenMotion (Feng et al, 2011) ...	175
Table 7.1 IDE VM Extended Performance Resource Management Algorithm	184
Table 7.2 Binomial Comparative Resource Performance Features/Characteristics	187
Table 7.3 Experiment Balloon/Hotplug Drivers	189
Table 7.4 Comparative Performance Resource Management Studies.....	192
Table 7.5 IDE Resource Management Evaluation	202
Table 7.6 Study 1 Resource Management Evaluation	205
Table 7.7 Study 2 Resource Management Evaluation	207
Table 7.8 Overall System Characteristic Scores %.....	208
Table 8.1 Thesis Contributions	214
Table 8.2 IDE versus AWS VM Provisioning Time.....	219
Table 8.3 IDE versus Oracle VM Provisioning Time	219
Table 8.4 IDE versus AWS CLR.....	219
Table 8.5 IDE versus Oracle CLR	220
Table 8.6 IDE v Paper1 (vMotion) Avg. (Mean)Failover/Migration Time	220
Table 8.7 IDE v Paper1 (vMotion) Best Failover/Migration Time	220
Table 8.8 IDE v Paper2 (vMotion, XenMotion) Avg. (Mean) Failover/Migration Time	221
Table 8.9 IDE v Paper2 (vMotion, XenMotion) Best Failover/Migration Time	221
Table 8.10 Platform Features, Availability and Capability Scores	222
Table 8.11 Platform Binomial Characteristic Assessment Scores	222

Chapter 1: Introduction

1.1 Background

In the 1960s it was IBM's research Cambridge scientific centre (Massachusetts) experiment with CP-40 that paved the way for the beginning of the full virtualisation of computer systems ([Pugh et al, 1991](#)). Full virtualisation is defined as multiple operating system instances sharing the same computer hardware resource. From there on developments have continued on apace, and since the late 1990s onwards, there has been a shift in virtualising computer systems on the more popular x86 system architectures ([Rosenblum, 2004](#)). This progression has led to widespread use of virtualisation technology to consolidate computer systems within the modern datacentre space. Ironically, while this event was something that was deemed beneficial in the IT industry using virtualised systems to reduce the need for physical system datacentre space, power consumption and cooling, there was one aspect that many organisations failed to factor in and that was the overhead of increased complexity due to the increased density ratios of VMs to physical (bare metal) systems ([Rasmussen, 2009](#); [Al-Ou'n et al, 2015](#)). Indeed, managing a set of physical computers with different hypervisors that are for example hosting hundreds (or even thousands) of Virtual Machines (VMs) with different operating systems is not a simple task, especially when you start considering inter-dependencies ([Su, K. et al, 2015](#)). It is this challenge that leads to the possibility of using an intelligent system to manage such a complex virtualised environment; ultimately, this leads to the concept of machines managing machines, which is in part one of the motivations of the author's research discussed later on ([Gazis, 2016](#)).

The very idea of designing and building an intelligent system in order to simulate a human expert administrator that has some level of autonomy, logic processing and functional self-awareness is an exciting prospect in terms of what potential it has to improve VM systems provisioning and management ([Haugeland, 1989](#); [Diao et al, 2009](#)). Indeed, being able to imitate human natural intelligence and behaviours closely allows the system to exhibit synthetic intelligence when he or she interacts within the controlled environment ([McCorduck](#)

[et al, 1977](#)). The term ‘intelligent system’ can be described as having the means or ability to be able to analyse information, understand or make sense of it in the context of a certain knowledge area, and subsequently process and organise. Once organised, the information is then made accessible and used to create methods to build system and environmental interactions, which ultimately allows it to solve problems in an efficient or elegant way ([Mei et al, 2010](#)). Humans have long strived to imitate and replicate the processes and systems that exist in nature and in some way, transfer this expertise (expert knowledge) into machine like systems. The challenge to devise an intelligent expert system to provide knowledge for solving the complexity of managing enterprise virtualised systems is something that provides the opportunity to create a unique solution approach ([Callaos, 1994](#); [Spangler, 1991](#)).

The ability to extend the control of such intelligent systems is potentially further enhanced by network technology advances, that have resulted in many end-user devices now having an Internet Protocol (IP) address and connectivity to the internet; indeed, there are now literally billions of devices which represent a modern paradigm now known as the Internet of Things (IoT). Given this level of connectivity, either through data networks, mobile telecommunications, wireless protocols and others, it follows that this can be used as an advantage to control remote systems ([Gazis, 2016](#); [Jing, 2011](#)).

1.1.1 Enterprise Computer Virtualisation

With the beginning of virtualisation on the x86 architectures, there has been a clear shift towards the use of popular hypervisors like VMware, Xen and others ([Scroggins, 2013](#); [Oludele et al, 2014](#)); it was in the late 1990s that the first modern hypervisors began to make inroads into the datacentre space ([Rosenblum, 2004](#)). Below is a diagram that shows how virtualisation maps on to a physical host:

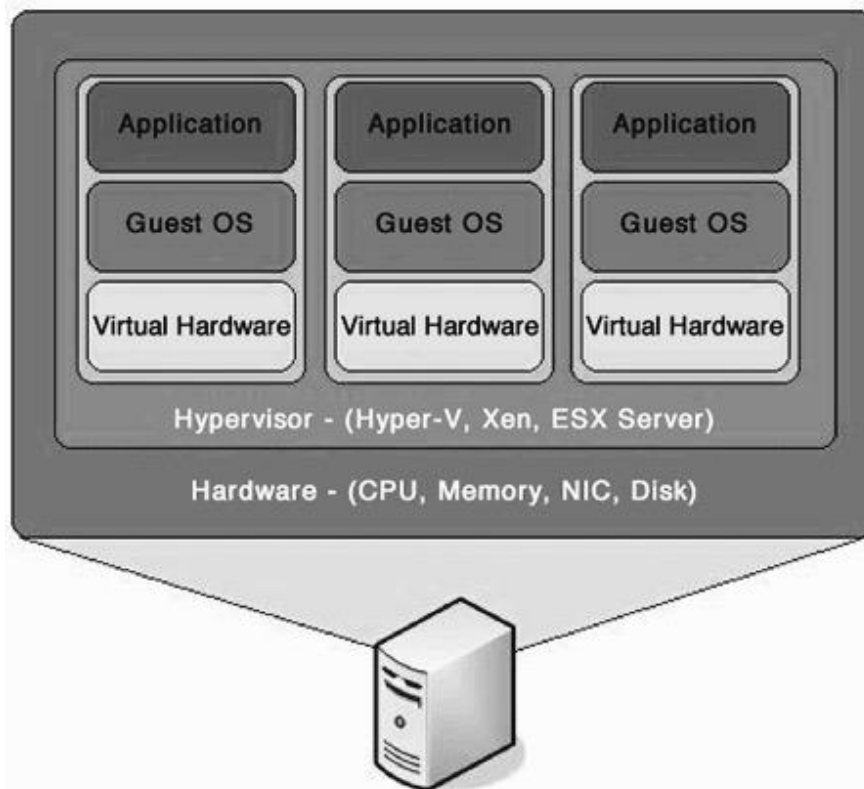


Figure 1.1 Example of Virtualisation of x86 hardware ([public domain image](#))

To provide further details of the above virtualisation example, the diagram figure [1.1](#) shows a typical x86 computer. Such a system (like any) has a finite amount of CPU, memory, network and disk capacity and performance. Given that most enterprise computers now have a large amount of CPU cores, threads, physical RAM, multiple Network Interface Cards (NICs) and disks, for most types of applications it makes sense to divide these resources amongst the VMs that host them ([Tsai, 2009](#)). The hypervisor layer is the critical layer that manages the hardware resources (synchronising, queuing and scheduling), typically presenting virtual CPU, memory, network and disk devices ([Lakshmi, 2010](#)). These virtual devices are made available to the local VM and are assigned to it as resources.

In the example figure [1.1](#) above, it is assumed there are three VMs that can divide the total resources available, therefore sharing the complete resource pool of the physical host computer. Each VM is often referred to as a guest of the physical host computer, in that it resides as an entity on that particular host system. Typically, a guest VM has its own operating system installed and configured. One of the advantages of virtualised systems is that each

guest may have a different operating system type, which can reside on the same physical host. For example, one VM may have a Linux type OS, another a Windows type OS and another a Solaris x86 type OS. In this way, virtualised systems provide a great deal of flexibility to the end-user, in terms of increasing the number of configuration permutations available and applications that can be installed, configured and supported ([Wood, 2011](#)).

At this stage, it is worth highlighting the difference between type one and type two hypervisors. The figure below shows the fundamental differences:

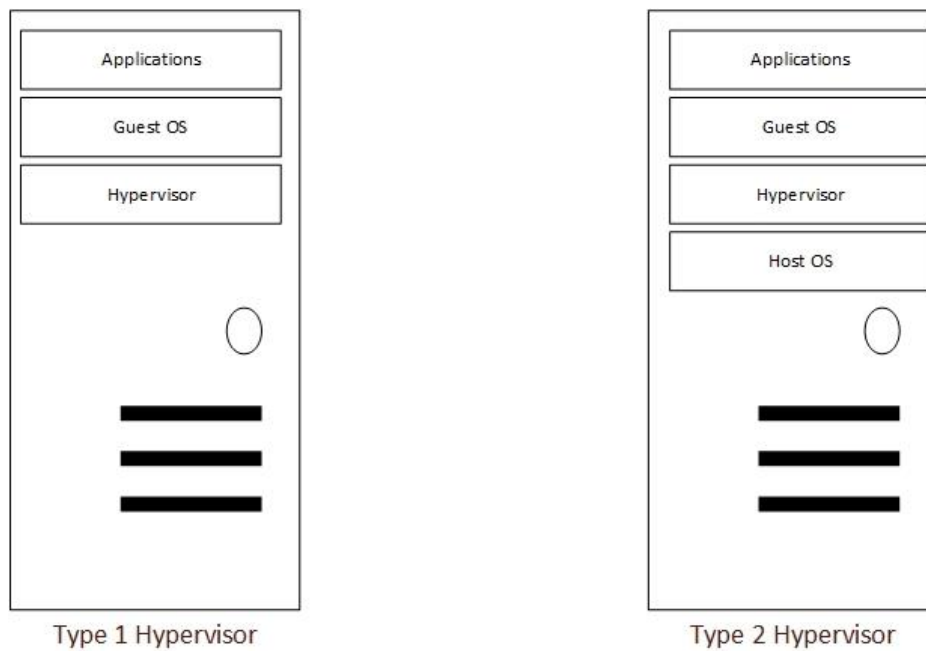


Figure 1.2 Type I Hypervisor v Type II Hypervisors

As can be seen in figure [1.2](#), the fundamental difference is the fact that type one hypervisors install direct on to the physical system, whereas, the type two hypervisor requires a host operating system and then the addition of the type two hypervisor install on top ([Morabito et al, 2015](#)). There are various commercial and opensource type one and two hypervisors available for use. As an example, a popular commercial type I hypervisor would be VMWare ESXi, and for an opensource type two hypervisor Oracle’s Virtualbox ([Bakhshayeshi, 2014](#)).

1.1.2 Type I Hypervisors: Advantages and Disadvantages

- Hypervisor occupies less Random-Access Memory (RAM).
- Relatively fast to re-install hypervisor.
- Highly optimised for running virtual machines, which is the primary function.
- Reduced driver support; only certain hardware is supported.

1.1.3 Type II Hypervisors: Advantages and Disadvantages

- Takes advantage of any hardware the host OS has driver support for.
- Host OS allows greater potential to monitor and interact with (via client agents).
- Possible to create multiple virtual machines of the identical guest OS as the host operating system, thus increasing performance and reducing overheads.
- Advantageous for developer type environments, where access to multiple guest operating systems and their variants is required.

Thus, there are different types of scenarios where the type one and type two hypervisors both have advantages and disadvantages. Either way, the type one or two hypervisor can both be used successfully to achieve virtualised systems deployments ([Pagare and Koli, 2014](#)). The next sections describe how this virtualisation technology has evolved into cloud-based services and how this is generally being applied and used within the IT enterprise space.

1.1.4 Cloud Computing (Public, Hybrid and Private)

Cloud based computing is a relatively new term used to describe the use of internet service-based computer resources. These cloud resources represent typical enterprise

datacentre systems, comprising of virtualised server hardware, network infrastructure, disk storage, and applications. In the case on public clouds, being an internet-based service allows organisations to acquire computer resources remotely at third-party hosted datacentres ([Bhise and Mali, 2013](#)). Very often these cloud service datacentre locations are based around different parts of the world and organised into functional operational regions (such as North America, or Western Europe). This is to allow end-user groups to take advantage of the cloud vendors distributed infrastructure and provide better resilience and availability of services ([Larumbe and Sanso, 2011](#)).

Cloud based public services offer the advantage for organisations to setup their IT infrastructure very quickly, without any significant investment of their own in terms of purchasing computer hardware; the only minimal costs would be ensuring their own organisation has internet connectivity and suitable end-user devices, such as employee desktop or laptop computers. Nearly all public cloud infrastructure services offer a utility or 'pay as you go' type cost model, whereby, the end-user organisation is charged directly for the use of compute resources based on how long (the amount of time) they need the type of resources they request, such as the amount of VMs they build, the amount of storage consumed, and the number of IPs required ([Kokkinos et al, 2013](#)).

Indeed, cloud providers minimally provide what is known as Infrastructure as a Service (IaaS), as well as other service offerings that extend their capabilities beyond the base infrastructure functionality, into further areas. These layers on top are known as Platform as a Service (PaaS), which is essentially the mechanism responsible for configuring necessary middleware and integration on top of the infrastructure stack. Finally, advanced cloud providers have Software as a Service (SaaS), which provisions applications to enable a full end-user interactive experience ([Bojanova and Samba, 2011](#)).

Typically, the cloud infrastructure organisations hide the complexity and management of their infrastructures away from the end customer. This is advantageous, in that such organisations can provision their cloud systems quickly, focus their efforts primarily on development and their business needs. It should be noted, that while there are many advantages, some disadvantages exist; these are often security related, in that the

organisation data is stored in the cloud infrastructure potentially at any geographical location managed by the cloud vendor. Lack of awareness of the cloud pricing model may also be an inhibitor, if the organisation cloud administrators are not aware of how the cost model works and they do not follow strict housekeeping procedures ([Imai et al, 2013](#)). Maintenance schedules and terms of service offered are very specific around system patching and overall VM life-cycles. The aggressive maintenance schedules imposed mean that any organisation embracing cloud services, needs to have an operating model that fits such terms imposed by the service provider.

Hybrid models adopt a slightly different approach. These are often more commonly found with established enterprise organisations who already have their own datacentres and investment in computer hardware and associated infrastructure ([Hwang, 2016](#)). Given the popularity of cloud computing, and the general strategic shift of many organisations to use such platforms, it is not unusual for there to exist a hybrid model. Usually, there are two fundamental drivers:

- Enabling quick provisioning of resources (in effect a burst type model), so that extra computer resources can be acquired to support on-demand type services such as online marketing campaigns.
- Migration and transformation from old deprecated (out of support) systems, into the cloud; for example, moving an on premise (traditional) email system to a vendor cloud service.
- Having a dual approach (private or cloud) allows IT security to decide what applications and data may or may not be considered for migration to a public cloud service.
- Changes to organisational workforce; often companies are adopting different ways of working and access via the internet to cloud type solutions offers easier ways of working.
- Due to the nature of public cloud, it offers a convenient method to ensure continuity of service in the event of a local disaster, whereby data can be securely transmitted via the internet and service restored within that environment.

Hybrid models therefore offer great flexibility for organisations to slowly transition, using a controlled approach, allowing them to decide to continue using their own private systems, or alternatively migrate services to the public cloud ([Fadel and Fayoumi, 2013](#)).

Private cloud services follow the nature and build approach that a public cloud service provider would follow, apart from the fact that there is no internet of public access. Rather, the cloud service provides a private service to one or more specifically known organisations. Often, this model is followed by larger entities who want to move away from the traditional approach to building infrastructure systems, installing middleware and software applications. Instead, they perceive that a cloud like service model provides a much more agile method of satisfying their business IT requirements, while retaining full control and security. Therefore, being able to utilise cloud services, although taking a significant amount of initial investment enables organisations to acquire infrastructure resources in an efficient way.

Indeed, many large cloud service providers now effectively bring their own proven cloud technology direct into their customer's datacentres to enable them to leverage the delivery methods already tested, tried and proven. As an example, this would include as a minimum to support IaaS:

- VM provisioning: build of virtual guest machines of various OS types.
- Network provisioning: build of necessary network zones.
- Security provisioning: enabling the opening of firewall ports between network zones.
- Storage provisioning: enabling appropriate storage to be made available via network or Storage Area Network (SAN).

There are many options to provide further functional layers on top of this basic one available, as discussed previously, PaaS, DaaS (Database as a Service), and finally SaaS ([Jin, 2016](#)).

1.1.5 Common Virtualisation Problems

The following figure [1.3](#) describes some of the problems encountered in virtualised environments. For this study, an examination of the following three principal areas is conducted:

- Over-utilisation of a VM (or set of VMs) – whereby a combination of one or more CPU, Memory or I/O resources have become exhausted and the system has become very slow or even unresponsive (example figure [1.3](#), workload 2)
- Under-utilisation of a VM (or set of VMs); this is where spare compute resource is not being used effectively (example figure [1.3](#), workload 3). This could be considered wasted resource.
- Maintaining effective n+1 failover and high-availability while the virtualised platform is in operation (example figure [1.3](#), workload 4). A common issue, even on architectures designed to run in such a fashion, is for human configuration errors to be made, or systems to become overloaded accidentally. On platforms with many hundreds or even thousands of VMs, it is a problem an administrator may overlook, resulting in a system that does not continue to function with its original objective of providing high availability.

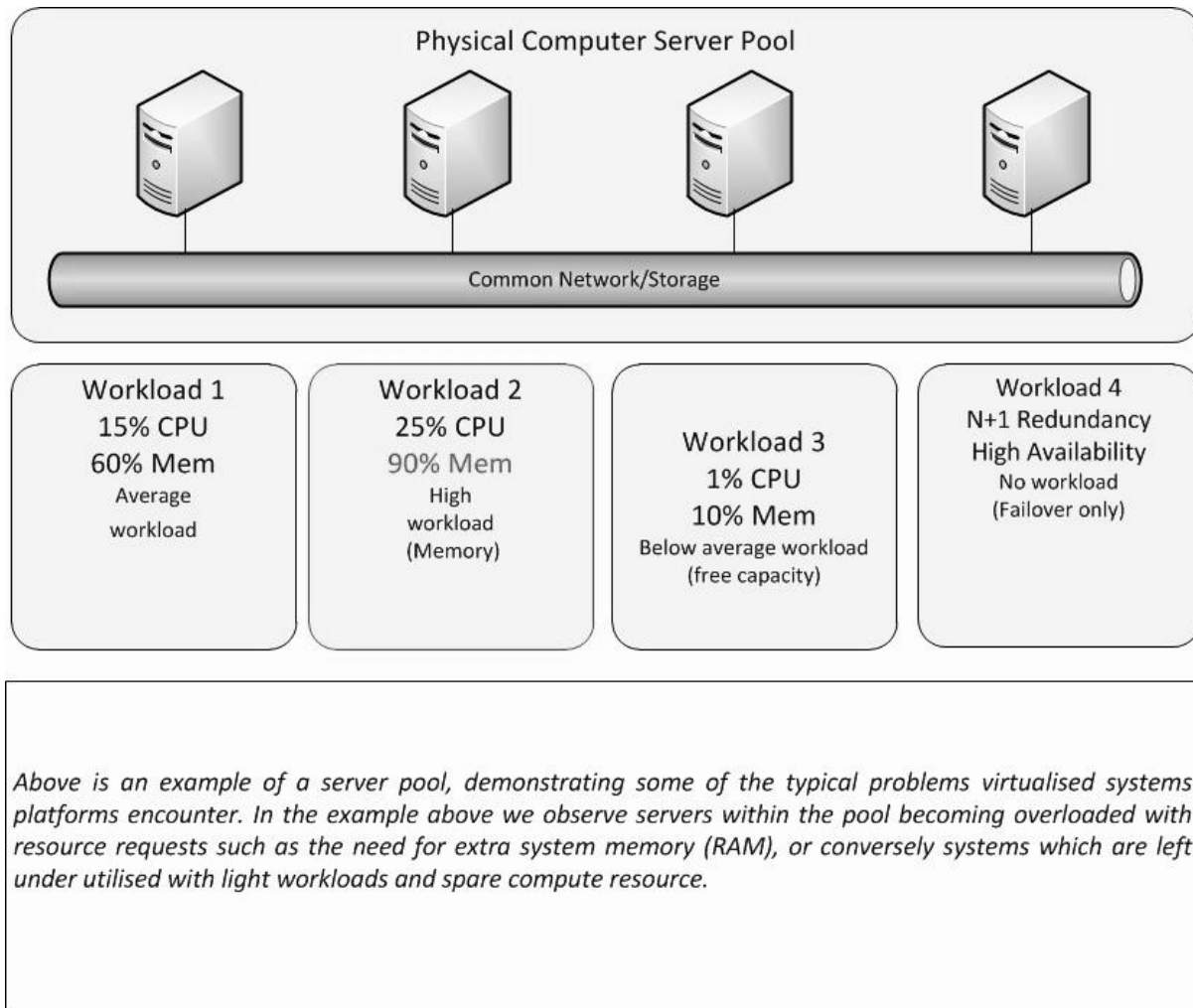


Figure 1.3 Common Virtualisation Problems

- Reducing the complexity of VM provisioning; many platforms use intricate processes and can be confusing to end-users. The build procedure often requires advanced technical skills to deploy systems ([Scroggins, 2013](#)).

1.2 Thesis Motivation and Aims

Within the enterprise computing space since the late 1990s, there has been a transition and evolution from single physical computer systems, on architectures such as Intel x86, Sun SPARC, HP PA-RISC, IBM POWER Series with a single OS instance. These have shifted towards fully virtualised platforms, running Hypervisors such as Xen, VMWare, VirtualBox and many

more, with many types of guest OSs and applications embedded.

This development has been intriguing. Organisations generally have embraced the modern technology, and the ability to consolidate systems into fewer, more powerful machines. With the advent of cloud computing, it can be observed that commodity type x86 architectures have taken a predominate hold, with huge distributed computing resources located around the world by various cloud vendors like Amazon, Rackspace, Google, IBM, Oracle and many more.

In all this, it has been a continuous struggle to effectively manage such technologies in a seamless way, without having on hand lots of technical able people to administrate and control such platforms. Indeed, even maintaining build and configuration standards is an almost impossible task, given the number of possible permutations to build virtualised systems ([Vrijders et al, 2016](#); [Poghosyan et al, 2016](#)).

It is this concept, which has led the author to be motivated to want to research this particular field, in order to provide a better solution, method and process for managing and controlling such platforms. While it is impossible for a single researcher (on his or her own) to address every technology area that a cloud provider like Amazon Web Services or Oracle can, there is opportunity to demonstrate by focusing on a few areas how improvements can be realised. The author hopes that such an opportunity taken will provide some original and useful additional research outputs in the following key areas:

- To develop a prototype system known as the Intelligent Decision Engine (IDE) to provide domain knowledge expertise around computer virtualisation and management.
- To provide a simplified VM provisioning process.
- To improve VM workload migration processes.
- To improve VM performance and availability.

1.3 Thesis Benefits and Targeted Applications

The project aims to deliver some benefits for various end-user organisation types, who

use computer technology and require very fast, automated deployment of IT resources. Consider the example of an organisation that requires resources in a series of pre-engineered blocks; effectively private based IaaS units that minimally provide a necessary IT hardware footprint, with integrated networks, computer hardware, security, storage, software management and control. There are many scenarios where this might be useful:

- Schools, universities, education – providing quick resources for classroom students, colleges, or university researchers.
- Private and public sector industries (e.g. utilities and manufacturing), scientific government and military, using fast deployment of resources at any physical location, that may or may not be connected to the internet. Examples could be telecommunication providers, national health services, pharmaceuticals, or security, or military organisations that need to collect data and deploy IT based systems quickly, because of an incident or event.

A specific example of this includes:

- [Purunak \(1996\)](#) shows that multi-agent systems are in demand in industry and such rapid deployment of systems can provide benefits to organisations.

1.4 Thesis Limitations

As part of the undertaking of the project, there are several limitations that were encountered, that are acknowledged as follows:

- The number of nodes in the IDE cluster was not tested beyond three nodes. This could extend to much larger numbers (i.e. hundreds of nodes), however, it is envisaged that that would be continued as future work. Consult section [4.7](#) for more details on system availability and clustering.
- IDE Operating systems – Linux (CentOS) was the primary guest operating system tested; the Windows OS is also supported, but has not been developed against

extensively; again, this is envisaged as future work.

- Physical computer, network and storage resources were limited to that described in the Laboratory setup in section [3.2.1](#), primarily as a result of having to keep financial costs within a constrained budget.
- System data sources only tested against Linux (CentOS) platform during experiments. See section [4.6.4](#) for more details. Windows alerts/logs and events are expected to be captured at a future point.
- The number of knowledge rules was purposefully limited to a relatively small number of 8, as defined in section [4.8.1](#), [4.8.2](#) and [4.8.3](#). The project limited the rules, in order to be able to test the fundamental functions (such as physical host and VM failure) of the intelligent design engine, without creating many additional rules at this stage, which could not be developed and tested fully at this stage. Justifications for the knowledge rules and why these were selected can be found in section [4.8.5](#). Of course, the system has been devised so additional rules can be created/added as part of future development; see section [8.3.6](#) future work for more details regarding this. As an example, the current IDE system did not include a specific knowledge rule for 'filesystem full' (warning/critical), however, this could be added in a later development stage.
- The VM provisioning and cognitive load experiments in section [5.2.2](#) and [5.2.3](#) respectively were snapshot (point in time) experiments, and not tested for repeatability (i.e. The user repetitively creating VMs); It would be expected/predicted for example, that the end-users would quickly move from the 'novice' group to 'experienced' should such future experiments be conducted; however, the results would need to be collected and analysed in detail to prove this hypothesis is either true, false or inconclusive.
- The number of end-users which made up the experimentation process in chapter [5](#) was limited as defined in section [5.2.1.2](#). It is feasible that future work could be completed to include larger numbers of end-users.

- The hardware components used in section [5.2.1.7](#) was as kept close as could be made possible subject to physical costs. The IDE Platform used hardware of a lower specification for the provisioning tests, to avoid the costs of replacing for newer higher specification systems. Therefore, it is feasible that the results for the IDE could yet be improved if repeated using the latest compute resource type.
- VM provisioning experiments did not use the potential queuing idea described in section [5.2.2.6](#); implementation and development of this idea could see large potential reduction time in provisioning, and it is described further in section [8.3.1](#) future work.
- The papers used to compare failover/VM migration times for vMotion and XenMotion were limited data sets of six iterations/tests; while the IDE could be repeat tested extensively (limitlessly), for even more detailed comparisons, a larger volume of repeat tests could be undertaken as described in further work section [8.3.9](#) by building local VMWare, KVM and Xen test clusters.
- The IDE did not have a live migration facility yet developed (pending the use of VirtualBox teleport see future work in section [8.3.3](#)), so the studies that were compared against were not functionally exact, however, the results from the IDE – even in the full restart migration scenario are promising, in that that the overall migration time was fast in comparison to the other studies described in chapter [6](#).
- The knowledge rules developed in section [4.8](#) and justified in section [4.8.5](#) could not all be tested through direct experimentation due to the limitation of time and resources to fully complete all the development and setup the appropriate test and experimentation process. The knowledge rules not fully tested at this stage are knowledge rule 1, 2, 3, and 8.

1.5 Thesis Summary

This thesis aims to research a unique approach into how an Inference Engine (the IDE) can be used to leverage the use of expert domain knowledge to provide process and performance improvements in specifically three areas:

- In chapter [2](#) an in-depth study of intelligent (expert) systems and virtualised technologies is undertaken, along with an examination of two public cloud providers; then a comparison of the features, quality and characteristics of the cloud vendors is highlighted by focusing on the relevant features that exist.
- Chapter [3](#) considers the methodology and approach used to provide the platform and system to be utilised to perform and support the necessary experimentation phase. To compliment this, the formal metrics and expert user evaluation methods to be used are defined, to measure the success and value of the research and experiments carried out.
- Chapter [4](#) explores the development and characteristics necessary of an expert system (the IDE) to aid and improve the way in which virtualised resources are effectively managed and controlled.
- Chapter [5](#) specifically focuses on the core research outputs, including the following areas; the simplification of deployment of VMs, by using the IDE expert knowledge base, whereby the inference and logic engine are able to build and provision VMs with absolutely minimal information from the end-user, using a 'one-click' method. Minimal end-user inputs are required, such as hostname (or a reference to standards) and VM size, and VM type. The IDE then completes the entire end-to-end provisioning process.
- Chapter [6](#) continues by describing the means for improving methods for workload migration; it specifically targets new improved methods of event handling, resource re-location and effective processes to migrate workloads.
- Chapter [7](#) examines ways to optimise VM performance and health, in the scenario of physical hardware failures, software failures, and human errors.

- Finally, in chapter [8](#) a review is conducted on the thesis contribution, to draw overall conclusions on each area of the work, by focusing on the results and their value to the research field; additionally, consideration is given to what further work can be done to enhance and continue the work already undertaken.

Chapter 2: Expert, Cloud and Virtualised Systems

2.1 Introduction

As part of the investigation into expert and intelligent systems within the field of systems virtualisation, it is necessary to analyse existing methods and work in the subject area to determine how best to approach the management of such compute platforms. This chapter pulls together some key areas that the author believes to be most relevant to the study undertaken. Firstly, the author examines in brief how the organisation of information is critical to being able to imitate human intelligence, in terms of the key traits that can be expected to be evident and observable. By ordering information and logically categorising it in such a way that it can be easily referenced, effectively made sense of and essentially used in some capacity to make decisions and reach an effective conclusion. Following this, a light overview of the origins of Artificial Intelligence is presented to allow the author to set the context for the reader, in particular around several key historical moments that have been fundamental to the advancement of human knowledge in the field of work that is being considered.

The next area that is delved into in detail is that of expert systems; this type of system is essential for review as the methods and applications in this subject demonstrate how expert human domain knowledge can be applied to a variety of technology and scientific study areas. This provides the platform for the author to consider what knowledge domains have already had such applications made, such as in the medical field, along with the historical reported outcomes of such projects. It thus enables a comparison into the techniques used and allows for conclusions to be drawn to help provide insight into what techniques might be useful in the context of management of virtualised computer systems. Therefore, by reflecting on the lessons learnt from previous endeavours made in the field of expert systems, it makes common sense to consider combining, adapting and enhancing the most successful methods used ([Crittenden, 1990](#)). The next section examines two Public Cloud providers; one currently holds the largest commercial market share, and the other is less predominate, although clearly operating providing Enterprise Cloud services to global businesses. As Cloud computing is

considered to be leading the way in terms of automation and service-based delivery of IT, it is critical to investigate the mechanisms that such cloud providers use, to allow the author to compare those functional areas that correspond to the author's study ([Rokne, 2013](#)).

Following the focused review of two Cloud based providers, the analysis continues by considering other management approaches used by other researchers in the field of automation and management of virtualised computer systems. This is particularly useful, as it widens the overall view of what efforts are have already been made in this study area, along with strengths, weaknesses of each approach and an overall gap analysis. Finally, based on the gap analysis and weaknesses identified, we consider how the IDE could contribute to the field in several key areas by combining new algorithms, pattern analysis methods, natural language processing techniques, and an inference engine to improve the management of virtualised computer systems. This is captured, and an explanation is provided to show the advantages of using such a system in the overall context of existing works, systems and approaches.

2.2 Intelligent Organisation

The concept of Artificial Intelligence (AI), as opposed to natural occurring intelligence, is to enable computer systems designed and built by humans to exhibit intelligent behaviours to some degree or level ([Callaos, 1994](#)). In respect of this, part of the objective of this work includes investigating the potential for a system to include some of the following characteristics:

- The ability to keep itself functioning or rapidly replicate to survive.
- To be able to make small functional improvements to itself; this has to be initially defined by its creator, with the possible potential to extend this function.
- To have the necessary function to make decisions based on available information.
- To have the function to be able to automatically invoke other programs as necessary, based on its own decision-making process.
- To have the potential to change itself either by developing, analysing and modifying

its own routines and processes, or perhaps even introducing new processes and procedures altogether based on an evolutionary, or self-analytical development model.

- The ability to organise, store information appropriately and retrieve it as necessary.

The following sections will discuss in detail the critical areas for consideration in respect of how a system could utilise AI to effectively manage virtual machines.

2.3 The Origins of Artificial Intelligence

Humans have long been fascinated by the concept of transferring natural intelligence to their own mechanised creations. These ideas stretch back as far as writings recorded in Jewish history via the Ten Commandments and events recorded in Greek mythology ([McCorduck et al, 1977](#)). In more recent modern history, circa 1843, it was Charles Babbage and his colleague Countess Lovelace, who created the first general purpose computers, such as the Analytical Engine, which included an arithmetic unit and programs in the form of data punch cards, concepts which are familiar in modern computing ([Tanenbaum, 2006](#)). More recent is the achievement Alan Turing and his team made in breaking the German Enigma codes using the famous Turing machine, during World War II ([Haugeland, 1989](#)).

As demonstrated above, it is feasible to therefore use computerised programmed systems to help simulate or imitate human like natural intelligence, in such a way as to perform complex tasks to help problem solve. The author of this research aspires that the work undertaken will demonstrate benefits in the subject area of applying natural intelligence to complex virtualised compute platforms using existing and potentially new AI techniques.

2.4 Expert System Applications

2.4.1 Introduction

The following sections consider real word examples of expert systems. The case studies below are of importance, because despite being orientated towards other expert knowledge domains, the principles and techniques used can be applied equally to any expert system that uses a knowledge base and inference engine. In the cases below, this allows the study of similar approaches undertaken by other projects, and assists focus on the strengths and weaknesses of other systems to help overcome commonly encountered problems from the past; each system covered lists the advantages and disadvantages based on the approach taken by the creator.

2.4.2 R1/XCON

R1/XCON (Expert Configurator) was an expert system designed by Digital Equipment Corporation (DEC) to be a system configurator for computer hardware. It was developed in the 1970s to provide sales staff with expert domain knowledge around what components to include in Virtual Address Extension (VAX) computer hardware sales. The system ensured that systems were shipped complete with all necessary components and was a successful commercial example of the application of expert systems within industry ([Winston and Prendergast, 1986](#)).

The advantages from this example of an expert system are:

- It was a commercially successful application.
- Proven quality in the domain of expertise – VAX computer systems configuration.
- The closest example of how an expert system can be used in the field of computer engineering to demonstrate how configuration knowledge can be used to assemble

the complex list of components for VAX computer systems. This is probably the nearest comparative system to managing virtualised computer platforms.

The main disadvantage from this example:

- The R1/XCON system was very specific – its expert knowledge was narrow around the VAX-11/870 ([McDermott, 1982](#)). Conversely, some may not consider this a disadvantage at all, as being narrowly focused on a very small knowledge area could allow for the potential to focus the expertise to a greater level.

2.4.3 MYCIN

MYCIN is an example of an expert system developed at Stanford University in the 1970s, to support medical staff help diagnose bacterial infections and suggest an appropriate antibiotic treatment using its inference engine and knowledge base. There are many positive aspects from the system that was developed, primarily that its ability to correctly diagnose and prescribe correctly, out-performed medical staff during the trials and experimentation phase. Given the positive trials, MYCIN had only around six hundred rules, which given the relative complexity and permutations within the field (there are well over one hundred antibiotics types), leads us to the conclusion that it was in fact a successful expert system concept ([Alty and Coombs, 1984](#)). It was only the fact that there were ethical challenges presented, over who would be responsible for any mis-diagnosis, that inhibited its further progress into mainstream medical practice. In that respect, using an expert system purely in a computer management type environment (outside of medicine), reduces the risk of failure in terms of improving its potential to be able to be applied into its particular field of expertise ([Musen et al, 2006](#)).

The advantages from this example of an expert system are:

- Expert systems did provide improved diagnostics.
- A relatively simple rule set provides the necessary functions.

The main disadvantage from this example:

- Ethical challenges due to the complexity of understanding who would be responsible

for a misdiagnosis, potentially resulting in patient harm.

2.4.4 INTERNIST-I

INTERNIST-I is another interesting expert system developed at the University of Pittsburgh, that captured the knowledge of just one medical expert Jack Meyers. Unlike other systems, INTERNIST-1 used an advanced ranking algorithm to arrive at a diagnosis of a disease. It excelled when only one disease was present, however, struggled to deal with more complex scenarios, where two or more were evidenced in a patient. Additionally, using a heuristic based problem-solving approach, it did not guarantee the best diagnosis method and the system interface was slow to operate, resulting in poor uptake by those medical professionals using it in the field ([Miller et al, 1982](#); [Ravindranath, 2015](#)).

The advantages from this example of an expert system are:

- Powerful heuristic ranking system to provide most probable diagnostic.

The main disadvantages from this example:

- Narrow expert view – knowledge derived from one expert source only.
- Poor at dealing with multiple problems, for example, patients with two or more illnesses.
- Overly time-consuming user interface, resulting in poor uptake and use of the system.

2.4.5 DENDRAL (DENDritic Algorithm)

This was a very early expert system, developed at Stanford University in mid 1960s ([Feigenbaum and Buchanan, 1994](#)). Its expert subject field was organic chemistry, with the objective of performing an analysis of molecular structures using mass spectra. The primary approach of the systems was to use a heuristic search/algorithm. The rule base was successfully engineered using the LISP programming language, which resulted in advances in

knowledge engineering which were made available and published ([Lindsay et al, 1993](#)).

2.4.6 HEARSAY I and II

Another example of an early expert system is Hearsay, developed at Carnegie-Mellon in the late 1960s ([Reddy et al, 1976](#)). The domain expertise was in the field natural speech understanding for structured database queries. The primary approach used a blackboard type problem solving method (a way of aggregating partial solutions to provide a complete one), through recorded application of ongoing expertise, to reach a consensus on the hypotheses using independent knowledge sources. The system was engineered using the Stanford Artificial Intelligence Language (SAIL), however, it was not very successful, initially. Nevertheless, it proved the feasibility of automated speech recognition and provided the inspiration for the development of other expert systems.

2.4.7 MACSYMA (MAC's SYmbolic MANipulator)

The system was developed at MIT from 1968 onwards. Its expert subject was to perform complex mathematical procedures (e.g. algebra), using a primary approach of brute force encoded algorithms. It too was engineered using LISP, and was a widely used, powerful system. It is available today as GNU freeware via Maxima ([Fateman, 1989](#)).

2.4.8 PROSPECTOR

Developed at SRI International, located at Menlo Park, California in late 1970s, with its expert subject field in exploratory geology and evaluation of geological sites. The primary approach of the control architecture involved the use of an inference network and a rule-based judgmental reasoning system that evaluated the mineral potential of a site or region, with respect to inference network models of specific classes of ore deposit. The system was engineered using INTERLISP (a derivative of the LISP programming language). In one

controlled test, the expert system successfully identified a previously undiscovered site, thus further demonstrating its commercial viability ([McCammon, 1989](#)).

2.4.9 Expert Systems: Why Have They Been Considered?

The introduction of this section alluded to the point that expert system principles are a transferable feature across knowledge domains ([Brooks and Heiser, 1979](#)). Based on this idea of transferability, it enables the investigation to proceed on the basis that such systems can re-use, evaluate and improve previous methods undertaken. From the historical expert systems investigated, it appears that the management of virtualised computer systems has not previously been undertaken, or fully explored by other researchers; therefore, this can be considered a new knowledge domain in relation to currently available expert systems. The above examples of the application of expert systems show how such methods can be applied to almost any field that requires human intelligence, demonstrated through problem solving skills.

2.5 Public Cloud Systems

2.5.1 Introduction

Another area of investigation is public cloud service-based offerings, which have become popular since 2006 and the advent of Amazon and their elastic cloud service. Figure [2.1](#) below provides a representation on the current market share figures for the various cloud providers – currently led by Amazon and Microsoft:

Cloud Computing Services Market Leaders

Year	2017	2018
Organisation % Market Share		
Amazon	40%	37%
Microsoft	29%	30%
Google	10%	10%
IBM	7%	8%
All others including Oracle Corporation	14%	15%

Figure 2.1 Cloud Leaders Market Share (Source: [Forbes, 2018](#))

This is particularly of importance, because public cloud providers like Amazon and Microsoft lead the way in commercial offerings. It is therefore necessary to explore how these providers compare in certain key areas such as, expert systems and reasoning, systems (VM) provisioning, VM migration strategies and performance monitoring.

2.5.2 Case Study 1: Amazon EC2

Amazon's EC2 public service is available via the internet in the form of Amazon web services. Like the following case study with Oracle's Cloud, it is very useful to functionally compare the cognitive load complexity and performance of their systems, against the research areas addressed by this work ([Plass et al, 2010](#)). Amazon's Elastic Compute Cloud (EC2) offers a web-service compute service offering to its end-users. The compute service works on the basis of buying compute time, storage and network services based around a certain set of parameters supplied by the end-user. This invokes a computerised in-house cost/billing model based on the type of instance(s) configured and the amount of time the components runs for in hours, minutes and seconds (system uptime). Typically, this would be configured based on the machine type, operating system (OS), CPU processing, memory, storage and networks requirements. Other factors that would affect cost would include any applications that may be requested; for example, Oracle RDBMS, or Microsoft SQL server. Once configured, VMs are

then accessed remotely using standard access protocols, for example, secure shell (SSH).

Amazon's EC2 is a public cloud solution that is service based, whereby, the infrastructure supporting the platform is largely transparent to the end-user ([Amazon Web Services, 2015](#)). The levels of automation behind the Amazon EC2 cloud are advanced, in terms of the level of automation, provisioning and resiliency achieved through their large-scale datacentre infrastructure footprint ([Bhise and Mali, 2014](#); [Awal et al, 2014](#)). One of the key differences of the authors research project is to alleviate even further the inputs from the end-user, by introducing an Intelligent Decision Engine (IDE), with the goal of vastly reducing the complexity to an end-user via a one-click provisioning methodology, much the same way Amazon allow purchasing of retail items on-line via their website ([Amazon Web Services, 2015](#)). Amazon's EC2 interface remains quite complex, aimed at developers and other advanced end-user computing groups, such as scientific research teams, Information Technology (IT) service businesses and IT departments ([Akioka and Muraoka, 2010](#)).

Leading providers of Cloud services such as Amazon EC2 have a web service that uses an advanced/complex Browser User Interface (BUI); further to this, the end-user has the ability to configure certain application (PaaS) offerings such as a MySQL database, or Apache web server (amongst many other features). Below is a table which summarises the EC2 service areas Amazon provides in respect to the similar areas of investigation for this work; the specific target areas of the author's study are highlighted to demonstrate the originality, which contribute to alternative strategies in the overall field of work.

The following table summarises the AWS areas that are being analysed, compared and evaluated against the IDE and Oracle platforms:

AWS Feature	Description	Comparative Project Investigation Area
Machine Learning	Allows you to build 'smart' applications, such as flagging fraudulent transactions and predicting user activity. This is an area of	Investigation developed further in chapter 4 of this work under the

AWS Feature	Description	Comparative Project Investigation Area
	<p>interest which is being investigated as part of the author’s research, however, the author is conducting more research effort around machine self-management, rather than smart end-user applications (for example, smart programs that analyse credit card spend patterns and analyse, risk assess them for suspicious activity).</p>	<p>section The Intelligent Design engine (IDE). AWS seeks to apply its 'machine learning' around applications rather than the 'infrastructure layer' which is a perceived gap.</p>
<p>EC2 (Elastic Compute)</p>	<p>This is Amazon's standard compute provisioning platform. From here you can launch Amazon EC2 instances which are individual VMs made of CPU, Memory and Disk. The high-level process flows are generally understood; however, the actual detailed provisioning process is unknown. This would specifically be referring to the code, logic and exact method (e.g. PXE boot, using image templates (AMIs), kickstart, or VM image snapshots). Most of this information is private to the company; they would not want to necessarily share their trade secrets. What is known is that the deployment mechanism is advanced and uses AMI (Amazon Machine Images), which is a quick and efficient provisioning method. This is an area of interest, which is being investigated as part of the author’s research work which undertakes an alternative one-click VM deployment</p>	<p>VM Provisioning mechanisms are developed further in chapter 5 of this work under the section Simplified Deployment of Virtual Machines.</p>

AWS Feature	Description	Comparative Project Investigation Area
	strategy for small scale to large scale Enterprises. It offers an advantage for end-users who are potentially less familiar with complex virtualised compute platforms and adds in considerable expert knowledge in order to provision VMs. This is to be compared and contrasted against the Simplified Deployment of Virtual Machines using an Intelligent Design Engine, using the evaluation strategy defined in section 3.3 .	
CloudWatch (Area of Research)	Monitoring for applications and resources – alarms and auto-scaling features. This is an area of interest, which is being investigated as part of the author’s research.	These areas are developed further in chapter 6 and 7 of this work under the sections Improving Workload Migration Strategies and Optimising Performance and Availability of Virtual Machines.

Table 2.1 Comparing AWS features

2.5.3 Case Study 2: Oracle Cloud

Oracle’s Public Cloud service, while advanced, is regarded as lagging behind the market leader cloud providers like Amazon and Microsoft ([Serrano et al, 2015](#)). However, it is interesting to examine a smaller niche cloud providers approach, such as Oracle, given their pedigree in the enterprise compute space ([Finkle and Scoresby, 2012](#)). Below, the table describes the essence of the core investigation areas that are to be undertaken in respect to

the Oracle cloud and the IDE and AWS platforms:

Oracle Cloud Feature	Description	Comparative Project Investigation Area
Oracle Advanced Analytics	Oracle’s advanced analytics aims to provide the ability to mine large datasets that can predict customer behaviour, estimate values, profiling people or items, identify rare events or anomalies and organise items into baskets of co-occurring events.	Investigation developed further in chapter 4 of this work under the section The Intelligent Design engine (IDE). As per AWS, this provides further evidence that most cloud providers are more interested in the AI aspects with regard to applications, rather than features lower in the stack e.g. infrastructure. This work concentrates on applying this to the lower down infrastructure components.
Oracle Cloud Machine	Provisioning, manage and maintain the Cloud Machine IaaS resources and PaaS infrastructure.	VM Provisioning mechanisms are developed further in chapter 5 of this work under the section Simplified Deployment of Virtual Machines.
Oracle Management Cloud	Oracle Cloud Management allows customers to build, deploy, and operate application environments on-premise, in a private cloud and/or on Oracle’s public cloud infrastructure. It maximises visibility and control over services and	These areas are developed further in chapter 6 and 7 of this work under the sections Improving Workload Migration Strategies and Optimising Performance and Availability of Virtual Machines.

Oracle Cloud Feature	Description	Comparative Project Investigation Area
	provides monitoring and reporting solutions to ensure adherence to IT standards and policies.	

Table 2.2 Comparing Oracle features

2.5.4 Cloud Computing: How it Has Created Utility Based Computing?

Most public cloud systems are only visible to the end-user from an internet browser-based interface. The complexity is hidden away purposefully, by design, and is presented as a service, so end-users need not be concerned with the technology that powers and creates VMs and containers (Biner, 2015). The typical cloud computing stack is represented as follows:

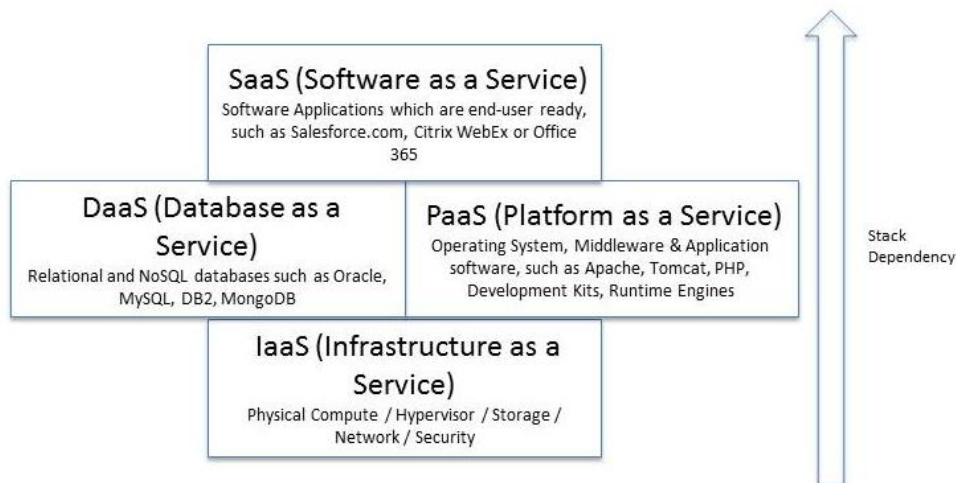


Figure 2.2 Cloud Service Stack

As above in figure 2.2, the cloud stack begins with Infrastructure, and works its way up to SaaS where end-user applications are made available directly to the user, such as a word

processing or email application. Public cloud systems use a variety of virtualisation technology to achieve their goal of providing such services to their end-users ([Bojanova and Samba, 2011](#)). Therefore, it is of relevance that the interface and methods they use be compared when considering how to improve aspects such as VM provisioning, performance monitoring, and migration.

2.6 Current Virtualisation and Cloud Management Approaches

2.6.1 Introduction

Using expert systems in the medical field has been previously well developed, understood and applied to the medical/clinical world; examples are MYCIN and INTERNIST-I (see sections [2.4.3](#) and [2.4.4](#)). They are particularly relevant in the case of the research, because they identify the potential benefits that can be achieved by the application of expert systems to problem solving within a knowledge domain, such as computer virtualisation, or the clinical diagnosis of bacterial infections. The study now examines in detail some existing methods and strategies employed by other research works to compare the strengths and weaknesses of other similar works. These are examined in detail, and specific care is taken to explain how this approach differs from those previously undertaken, by focusing on the advantages and unique methodology and ideas of this research project.

2.6.2 Reviewed Approaches

Virtualisation now has many applications across all infrastructure components; not only can computer hardware be virtualised, but so can other infrastructure components, such as the network and storage devices. Given the potential to use this technology to save space, power and consolidate systems, it makes sense for organisations to leverage this to their advantage ([Scroggins, 2013](#)). The question of how organisations effectively manage these complex environments forms the fundamental problem that this research work addresses; with the IDE utilising AI features, the algorithms and methods employed help to reduce complexity to end-user organisations, thus enabling the delivery of a fully virtualised compute

platform ([Scroggins, 2013](#)).

Given the various virtualised platforms now deployed in the field, attempts and approaches have already been made to automate deployments of VMs and other such hardware layers, for example, storage and network ([Oludele et al, 2014](#)). One interesting approach described how resources can be automatically provisioned in virtualised datacentres ([Elprince, 2013](#)). This study discusses how in the modern virtualised datacentre, there is a requirement to automatically provision and manage resources effectively due to the spiky nature of processing (i.e. a sudden shift upward in demand). One of the impacts of this naturally occurring event is that breaches in Service Level Agreements (SLAs) can occur due to VMs being impacted as they are under-resourced from a CPU, memory and storage point of view. The proposal here to deal with such events was to create an autonomic resource controller ([Elprince, 2013](#)). The system has two parts, a resource modeller (machine learning) and a fuzzy tuner (fuzzy logic) that allows dynamic resource allocation (or changes) to VMs to allow them to manage their computational load effectively. The resource controller also attempted to ensure no SLA breaches were made. The first obstacle mentioned is dealing with complexity of ensuring scalability (or elasticity) of virtualised systems. The system itself was modelled using a data trace only, and not on a real interactive environment. While this simulation provides real work-load patterns and opportunity to model different jobs, scheduling, and priorities, it may not always provide a real-world complete data-set from all relevant log files and system data. In this study, development and experiments are conducted in a real lab-based environment to enable true testing against live systems. This provides several advantages, the primary being that you can model the behaviour of intelligent systems with a higher degree of certainty, in terms of being able to observe and record how things operate and perform in a live situation ([Elprince, 2013](#)).

Nowadays, when you consider cloud services and their evolution and standard enterprise model of delivery, there are a whole host of resources that require controlling such as networks, servers, storage, applications and services ([Bojanova and Samba, 2011](#)). The requirement for control is clear, in that all these hardware and software resources need effectively managing, collectively and in harmony; one of the common disadvantages of today's enterprise datacentres, is the silo approach taken by many organisations to their data

centre build and delivery mechanism. By adopting this old, non-agile model, they make it far more difficult to automate delivery and manage the control of their systems, as responsibilities across technology space, as described above are handled by separate teams. This means it is advantageous to move away from diversified control mechanisms, and instead use a single team or entity much like the IDE to achieve a centralised management approach ([Gren et al, 2014](#)).

Another alternative approach was investigated to deploy VMs and applications using OpenStack, which is an open source toolset designed to allow automatic cloud configurations ([Zhang and Shang, 2014](#)). While some of the tasks were automated, there were several additional add-ons that had to be configured such as:

- An algorithm to control the network IP addresses allocated.
- Having to convert ISO images to allow installation.
- Configuring Dynamic Host Configuration Protocol (DHCP), firewall, and SSH public key infrastructure components.
- Shutting down the VM and registering in Glance (the OpenStack discovery and registration module).

Based on the above, the devised system leaves many further opportunities for automation and simplification of the VM deployment process and could be considered incomplete in its development.

Interestingly, a recent investigation explored proactive management for cloud-based architectures ([Dong and Herbert, 2013](#)). Rather than use a traditional method of reactive management, they suggest that a far better management strategy is to be proactive rather than to just react to occurring events. They programmed in certain intelligent traits, such as suggestions for tasks to be carried out such as VM migration in the case of a set of criteria being fulfilled. These suggestions are then evaluated in turn, to decide whether they should be acted upon. The evaluation process used a manual cloud build methodology, using IBM's SmartCloud, which was a noted problem, as the management system was not tightly integrated into the VM provisioning process; inherently, not being tightly coupled, means the

intelligent system will struggle to manage the IaaS (Infrastructure as a Service). For the actual management aspect, a private cloud simulator was used to allow this process to allow the theoretical management of between 50-500 VMs.

Further systems examine how workload schedulers can be applied to heterogeneous systems, which are able to run a combination of workload types ([Kim et al, 2011](#)). This methodology and approach are interesting; however, it differs in principle from the work being carried out by this project, with the key differential being one uses a controlled, tightly integrated modular approach. On the other hand, the alternative aims to generically schedule workloads across various cloud and computer resources an organisation may have available. This approach used CometCloud, a grid computing tool, designed towards heterogeneous compute environments. Other approaches to managing virtualised environments have solely addressed a single compute entity, like CPU resource ([Menasce and Bennani, 2006](#)). Their work demonstrates the ability to dynamically provision CPU shares to various VMs, depending on overall systems priorities; however, this work presents opportunity to build further on performance management aspects.

A Distributed Artificial Intelligence (DAI) system consists of multiple physically separated processing machines, with each having at least one expert system or knowledge source. No one node has the ability to entirely solve a problem. Instead, it must work together in a co-operative manner in order to resolve a problem. Typically, such a multi-agent system comprises of a number of components, described as a receiver and transmitter, meta-level knowledge, planner, scheduler, blackboard, solver and multiple knowledge sources. The components rely on interactions between themselves, with the receiver/transmitter using a defined protocol and language set to communicate with other nodes. Meta level knowledge allows for general node or environment awareness, so the problem once defined at a high level can be addressed and resolved by the correct candidate node. Task planning allows a specific problem to be broken down into a structured set of sub-tasks that can be addressed in a logical order by one or multiple nodes. The scheduler's goal is to decide upon the most effective way of reaching an overall solution by prioritising and ordering sub-tasks. In order to work effectively, the blackboard is used and accessed by each node as required, to allow node-to-node communications, with information and data stored about plans, tasks and

results. Finally, the solver is responsible for reaching the end objective of a final solution by tracking and determining the best path forward for all sub-tasks to complete (Yang et al, 1985).

An exciting work around the use of the mOSAIC framework has been completed to work to provision an IaaS/PaaS environment with intelligent management to help manage distributed cloud resources. The primary advantage of the tool is the fact it is geared towards any cloud platform service and can be considered vendor agnostic. This provides a great deal of flexibility, in that it can be applied and used and configured against various cloud platforms. On the other hand, however, the main drawback is the complex configuration and setup of the framework, along with the dependency for AI and automation that does not work by default without a considerable amount of customisation (Sandru et al, 2012). The mOSAIC product comprises of numerous modules shown in figure 2.3:

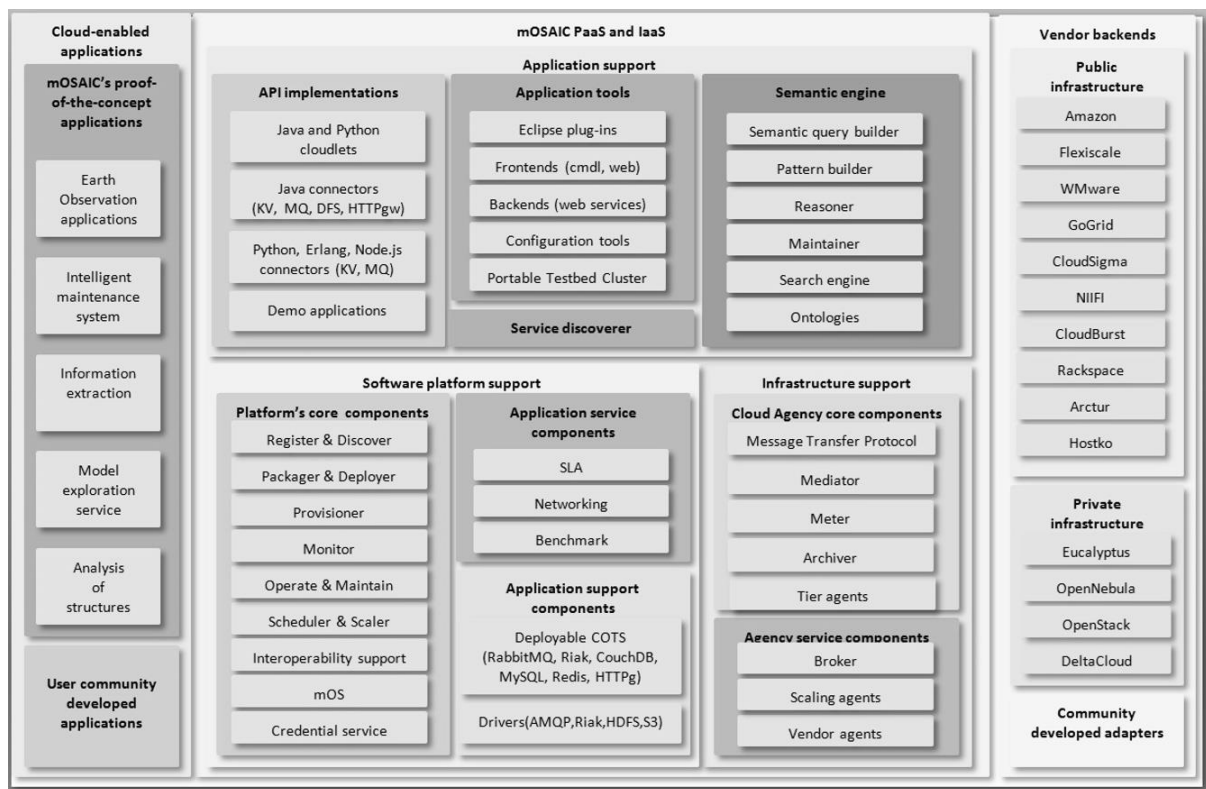


Figure 2.3 The mOSAIC Architecture

The above architecture demonstrates the product is feature rich; however, the framework requires a large amount of configuration, especially around specific vendor agents. This leads to the conclusion that the architecture components require considerable post-deployment activities to produce automated processes (such as VM deployment) for the organisation that uses the framework.

It follows, that if a system can be created to hold expert human knowledge, it can be applied to any field of human expertise. As part of the output of this work, the author endeavours to apply AI strategies in a novel way to help manage virtualised compute platforms more effectively. Indeed, at a very high level initially, this can be described as follows. To develop and build an expert system (IDE) which uses the following techniques:

- Data text mining and analysis to extract (quick and slow methods) from the platform, both real-time information and retrospective data analysis methods to help re-evaluate rules and logic base; typically, this would involve the identification of critical log and information files to allow the system to process and perform its own analytics. In effect, this is a three-step phase, with 1) identification of critical files – the system must be able to determine this and 2) real-time extractor – critical platform updates 3) retrospective extractor – thinking extractor and textual analyser.
- Performance and availability monitoring; the system needs to control all physical infrastructure components, and virtualised systems (VMs, Storage and Network). This includes application of SLAs and predictive failover for all VMs, with shadow instances for critical VMs. Some of this capability already exists in market leading commercial products; as an example, VMware are using similar systems with features such as Distributed Resource Scheduling and High Availability ([Shirinbab et al, 2016](#)). The goal is to improve the approach employed by using new techniques to enhance the overall performance using dynamic resizing of resources and faster failure detection and recovery times.
- Rule based and forward chaining decision making allows information to be extracted,

processed and applied to the virtualised platform. As an example, this would include building a forward chain for new VMs based in information available in the knowledge base.

2.6.3 Conclusions

The following table describes the conclusions drawn from the areas investigated, with analysis of the strengths and weaknesses for each finding:

Author(s)/Date	Summary of findings	Strengths and Weaknesses
Oludele et al, 2014	Attempts to fully automate VM deployments.	Requirement to build and improve on this methodology, as automation techniques are not fully developed.
Elprince, 2013a	Automatic provisioning of resources in virtualised datacentres.	A useful model which can be improved upon in terms of adding more automation steps.
Elprince, 2013b	Creation of an autonomic resource controller.	The idea of a resource controller is novel; however, it is concentrated mainly on prediction of the resources in a certain application may need in a VM container.
Elprince, 2013c	Resource modeller/controller uses (machine learning) and a fuzzy tuner (fuzzy logic) that allows dynamic resource allocation.	The approach primarily uses Fuzzy logic, which deals with partial truths, as opposed to Boolean values which are true or false only. As part of this work, further opportunities are available to

Author(s)/Date	Summary of findings	Strengths and Weaknesses
		investigate and apply other AI techniques to modelling and control, such as natural language processing, forward chaining and text mining.
Bojanova and Samba, 2011	Enterprise infrastructure delivery models applied into the cloud.	Interesting discussion on cloud architectures, which highlights how important delivery models will be in this particular field. Presents the idea that this area of work will be critical to shaping the future of cloud/virtualised computing environments.
Gren et al, 2014	Automate delivery and manage the control of their systems by using centralised management systems.	Argues for centralised management of distributed systems. Centralised services must, however, be resilient.
Zhang and Shang, 2014	Investigation into deploying VMs and applications using OpenStack.	An interesting approach using OpenStack. However, there is limited effort into how AI techniques may be applied to the environment and numerous manual steps listed, such as setting up a cloud computing platform in OpenStack. This in itself adds considerable complexity to the end-user.
Dong and	Study into proactive	Concentrates primarily on cloud

Author(s)/Date	Summary of findings	Strengths and Weaknesses
Herbert, 2013	management for cloud-based architectures.	management; however, the Operational Management Service (OMS) requires the build of at least 5 virtual machines in order to function, implying there are multiple steps for the end-users to effectively use the tool.
Kim et al, 2011	Feasibility of application of workload schedulers to heterogeneous systems.	This study focused on using a tool call CometCloud which is a framework for supporting workloads across distributed systems, such as Public cloud, Private cloud, Private clusters and so on. The ideas presented are interesting; however, in the paper the experiment phase describes a large amount of manual build steps, such as configuring and building public cloud VMs. In other words, automated VM and system provisioning did not appear to feature.
Menasce and Bennani, 2006a	Work around the dynamic VM allocation of resources.	Primarily deals with dynamic CPU resource allocation only, leaving potential for lots of other resource controls such as memory, network and disk I/O.

Author(s)/Date	Summary of findings	Strengths and Weaknesses
Yang et al, 1985	<p>Distributed Artificial Intelligence (DAI) system consists of multiple physically separated processing machines, with each having at least one knowledge source.</p>	<p>An interesting approach using distributed expert system components, however, this model, although effective at delegating load and tasks across multiple agents, presents the possibility of have more single points of failure, due to the single entities requiring replication (such as the ‘solver’ or ‘scheduler’); further work is needed to ensure each critical component is highly available. Questions also remain over the ability of the system to perform and effectively problem solve when using the black-board to communicate and share information with other nodes.</p>
Sandru et al, 2012	<p>This paper discusses the use of the mOSAIC framework to provide IaaS and PaaS, which attempt to use the tool to deliver automated provisioning of various cloud infrastructure and middleware components; for example, VMs, RabbitMQ, and MySQL.</p>	<p>The cloud management approach used is vendor agnostic, which can be perceived as a strength, as this allows the tool to be customised against any cloud provisioning service. However, this does leave the complexity of having to create the agents to support the multitude of vendors. The AI techniques are not fully explained,</p>

Author(s)/Date	Summary of findings	Strengths and Weaknesses
		<p>with only minimal references which allude to it being a necessary component to manage the complexity of the platform.</p>
<p>Ajila and Bankole, 2013</p>	<p>This paper discusses in detail three methods for predicting cloud resource utilisation of web applications using three machine learning techniques. Firstly, using a Neural Network, secondly via Linear Regression, and finally using Support Vector Regression (SVR).</p>	<p>The prediction model is interesting as the authors compare three different machine learning strategies. They determine through their experimentation that the SVR method is most effective at resource prediction and adaptation. However, little information is made available on how once the information is collated, VMs in the cloud are automatically modified should it be determined they require more or less compute resource.</p>
<p>Tian et al, 2012</p>	<p>This work considers using a Decision model for provisioning VMs on Amazon EC2, in terms of providing cost optimisation and capacity planning.</p>	<p>An investigation into how to best acquire Amazon EC2 resources/capacity based on three different pricing models. Those types are on-demand instances, spot instances and reserved instances. The idea was to reduce the cost to a minimum for EC2 provisioning plans. The results showed a promising strategy for</p>

Author(s)/Date	Summary of findings	Strengths and Weaknesses
		reducing overall cost, as well as little advantage to using spot pricing for short-term planning.
Lokshina and Insinga, 2004	Discusses the feasibility of using an expert system as a replacement for a human system administrator, acting in a support function.	Looks at how an expert system can use a combination of event driven decision making, utilising forward-chaining to reach conclusions on how to problem solve in a distributed and heterogeneous computing environment.

Table 2.3 Current Virtualisation/Cloud Management Findings

2.7 Intelligent System Approaches

2.7.1 Introduction

Given the gaps and challenges identified, a further examination and consideration of intelligent systems is undertaken in the areas of algorithms, pattern analysis, machine learning and inference engine. These areas are explained in detail below:

2.7.2 Algorithms

An algorithm is a step of sequenced actions that can be made up of a combination of reasoning, mathematical calculations and processing tasks ([Huang et al, 2012](#)). They link intrinsically to expert systems methodologies described in the above examples in section [2.4](#), which are essentially made up of a single or series of algorithms ([Mülayim and Alaybeyoğlu, 2016](#); [Wenbin et al, 2010](#); [Ashouri and Savoji, 2004](#)). Instead of being clinically based, the knowledge domain of this investigation is focused on the application (knowledge engineering)

of those principles to virtualised computer systems. Indeed, the field of expert systems (and associated algorithms) are equally applicable in helping to solve any type of problem that requires a level of human natural intelligence to create a solution ([Durkin, 1990](#); [Beckman, 1990](#)). Currently, this research field is wide and intensive as the studies examined demonstrate. As examples, consider further that significant efforts are being made in:

- Autonomic virtualised environments; the concept of automatically assigning CPU resources dynamically within a virtualised computational environment ([Menasce and Bennani, 2006](#)).
- Autonomous resource provisioning; the idea here is to design an autonomic resource controller capable of learning adaptively, by utilising Machine Learning techniques, effectively being able to make resource changes to meet Service Level Agreements ([Elprince, 2013](#)).
- Predicting cloud resource using support vector regression ([Ajila and Bankole, 2013](#)).

2.7.3 Text Mining

The IDE proposes using data text mining processes to analyse key data and log files ([Wong and Manickam, 2010](#)). This enables quick extraction of key data to enable the platform to make decisions and trigger key events. Examples of platform events specific to this work include VM deployment, VM failure, VM migration, and intervention to improve VM performance. Analysis of patterns is essential for the system to be able to perform two critical activities:

- Event response (reactive) based on real-time data.
- Event prediction (proactive) based on historical analysis.

The first activity, event response, is a classical trait for an expert system to exhibit ([Kulikowski, 1980](#); [Lokshina and Insinga, 2004](#)). Usually, a pattern of events is recognised, and a conclusion reached through logically joining those identified patterns to match an event response using

a method such as forward chaining ([Windriyani et al, 2013](#)). Once matched and initiated, a series (or even single) of actions are performed to provide a satisfactory system response; once completed, the tasks carried out can be evaluated and measured as successful or non-successful. Likewise, a further less common method is to use historical or collected data to proactively perform a set of actions, again using a method such as forward chaining ([Kwon, 2012](#)). Figure 2.4 demonstrates the basic approach:



Figure 2.4 Machine Event Response Mechanism

For example, you may be able to predict busy system times, such as just before batch processes start at 7pm on a Sunday evening. Therefore, it would be feasible to predict this event due to pattern analysis of historical data and invoke a procedure to increase CPU and memory available, to allow the system to perform more effectively. It is an objective for this work to incorporate both methods to support the IDE function.

2.7.4 Natural Language Analysis

Natural language processing is used to understand and organise information ([Lebowitz, 1983](#)). The IDE aims to use a knowledge base, with a thesaurus, an English based lexicon, along with grammatical rules to allow the system to make sense of all collected platform data and thus classify and formulate appropriately ([Gaikwad and Joshi, 2016](#)). Through organisation of information, the system will be able to use these resources to build a sequence of reasoning steps. Figure 2.5 demonstrates the process:

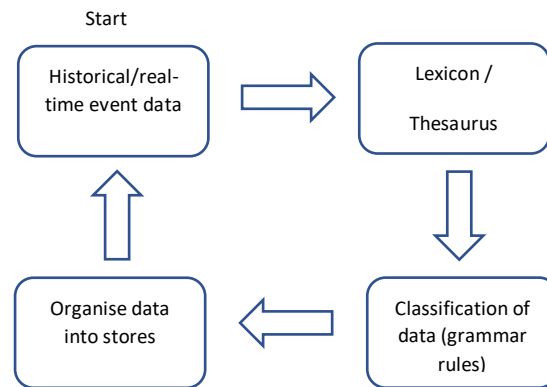


Figure 2.5 Natural Learning Mechanism – Information Organisation

2.7.5 Inference Engine (Forward Chaining)

Inference engine architectures can use backward and forward chains to logically create rules or new facts about the knowledge domain they operate in. With forward chaining, known facts are connected together to result in a new conclusion or fact. Conversely, with backward chaining, a desired goal is stated, and the facts required to achieve this goal are reverse engineered ([Mettrey, 1991](#)). Given the two approaches, initially an examination of forward chaining to build reasoning and conclusions (facts) will be undertaken. This will result in a suitable knowledge rule-based approach for managing complex procedures within the virtualisation of computer systems context ([Spangler, 1991](#)). The platform will therefore be expected to make data driven decisions, which are triggered primarily by real time events from information collected from the various components, such as VMs, storage and network devices. By utilising forwarding chaining of statements, this enables the system to reach a conclusion and invoke necessary functions described above to satisfy event responses or event predictions ([Novaliendry et al, 2015](#)). The premise at a simple level is presented as follows:

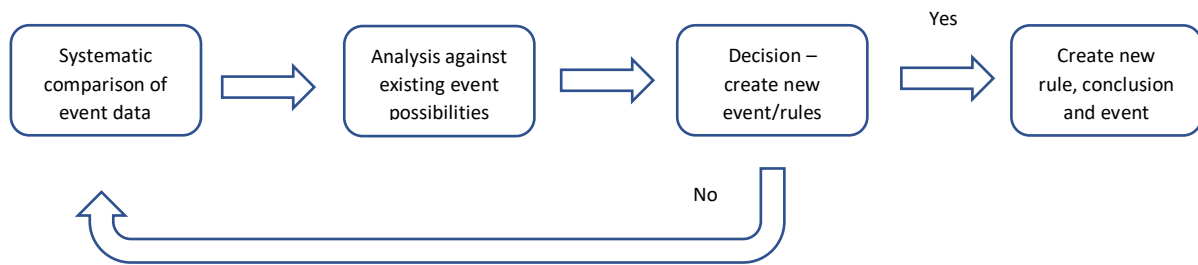


Figure 2.6 Machine Learning Mechanisms Event Response

2.7.6 Cognitive Load Theory

Cognitive Load Theory (CLT) emerged in the late 1980s, as a branch of cognitive science, with several key researchers involved in studies around how users are impacted by mental load during problem solving exercises ([Sweller et al, 1998](#); [Paas et al, 2003](#)). The key idea introduced, was being able to measure and capture the amount of mental power, or mental effort that is required to complete a certain task or set of tasks (process), in a controlled experiment setting. According to Paas and his colleagues, CLT is concerned with the design of instructional methods that efficiently use people’s limited cognitive processing capacity to apply acquired knowledge and skills to new situations, for example the transfer of knowledge ([Paas et al, 1994](#)).

Analytical methods are directed at estimating the mental load and collect subjective data with techniques such as expert opinion and analytical data, with techniques such as mathematical models and task analysis. Empirical methods, which are directed at estimating the mental effort and the performance, gather subjective data using rating scales. The application of rating scale techniques are based on the assumption that people are able to introspect on their cognitive processes and to report the amount of mental effort expended. Although self-ratings may appear questionable, it has been demonstrated that people are quite capable of giving a numerical indication of their perceived mental burden ([Gopher and Braune, 1984](#)).

2.8 Summary

2.8.1 Introduction

Throughout chapter [2](#), there is an overall review of the foundation, origins and motivations around the historical work done around AI and its many branches. By alluding to the origins and early success stories, we set the scene for further developments around expert systems (of all types), by highlighting their suitability towards imitating the way humans problem solve. Furthermore, an in-depth discussion of two well-known public cloud organisations brings the study up to the most current recent technology advances made. This allows for comparisons to be made in the areas being examined by this investigation. Following this, a detailed analysis of current research initiatives in the field allows a comprehensive view of what has been achieved to date, how this work fits into the existing body of knowledge; additionally, discussion around the identified gaps and expected challenges helps define and justify the future effort and work undertaken by this project. By completing this, it allows us to finally review some of the identified AI techniques that are deemed most likely to provide the best results for the intelligent management of virtualised computer systems.

2.8.2 Gap Analysis

The following table describes and captures those areas that are deemed to form part of the gap analysis from the literature reviewed in the field:

Subject Area	Gap Analysis Conclusion
Expert systems application to virtualised computer management platforms and cloud-based systems.	The author believes the work to be unique, in that no existing expert systems exist within the knowledge domain of virtualised computer management (Duda and Shortliffe, 1983).
Event detection combined with forward chaining (inference) to improve automated system response.	Examining existing work in the field of virtualisation, suggests little has been done around combining system event driven response with forwarding chaining to allow an expert system to evaluate and perform an automated reaction for example cause, effect and response (Anicic et al, 2009 ; Lokshina and Insinga, 2004).
Natural language processing for text analysis and advanced trigger generation.	Work to integrate natural language processing techniques for text-based analysis of the virtualised platform environment data to improve trigger detection and effective event response strategy (Gandhe et al, 2013).
Simplified one-click VM deployment.	Investigation into the reduction of the cognitive load rating for complex user activities like building and deploying VMs. From analysis undertaken of literature so far, opportunity exists to simplify processes and minimise required human interventions (Oakes et al, 2016).
VM performance and availability.	Builds on existing work completed around the dynamic allocation of resources (like CPU, Memory and Disk) by utilising the event driven processes and inference capability to drive

Subject Area	Gap Analysis Conclusion
	intelligent decision making on how best to increase or remove resources, in conjunction with service level agreements without the need for human intervention (Antonescu et al, 2013 ; Sarathy et al, 2010).
VM Migration.	Work to improve existing methods around VM migration between hosts and automated balancing workloads, by examining how to reduce service outage time, without the requirement for human intervention (Benet et al, 2016).

Table 2.4 Gap Analysis

2.8.3 Approach Challenges

The table below describes the challenges, analysis and conclusions that were reached based on the options available for the project:

Challenges	Analysis/Conclusion
Distributed versus centralised management approach.	As part of the solution approach, it is important to decide which design is the better suited to solving the problem (Yang et al, 1985). Conclusion: Choose centralised management.
Functional capability prioritisation.	Any solution provided requires an initial starting point. Deciding on what functional capability is critical in managing the project effectively. Rather than over-extend the initial capability, it is deemed advantageous to focus on the most critical functionality required and deliver the perceived improvements. Conclusion: Prioritised IDE functional capability areas as per section 1.2 Motivation and Aims.

Challenges	Analysis/Conclusion
Deciding which AI strategies are the most effective to utilise with the IDE.	<p>There are many AI strategies/approaches available, which could be selected to support the IDE. Choosing the most appropriate AI component is critical to the project. Current areas considered:</p> <ul style="list-style-type: none"> • Fuzzy Logic - Partial v Absolute truth (Elprince, 2013) • Support Vector Machines (Ajila and Bankole, 2013) • Machine Learning (Arnaldo et al, 2015; Melekhova, 2013) • Data/Text Mining (Prangchumpol et al, 2009) • Natural Language Processing (Mei and Cheng, 2010) • Forward and backward chaining (Anicic et al, 2009) • Expert systems (Spanger, 1991; Lokshina and Insinga, 2004) <p>Conclusion: Choose expert systems, based on the analysis completed in chapter 2 and section 2.4.</p>
Overcoming on-premise private and hybrid cloud limitations.	<p>Optimising on-premise private/hybrid cloud management techniques (Dong and Herbert, 2013; Jin et al, 2016; Zhang et al, 2014)</p> <p>Conclusion: Choose a private cloud management approach to explore tight integration, improved automation, better controls, based on section 2.6.2 Reviewed Approaches.</p>

Table 2.5 Approach Challenges

2.8.4 Justifications

The following table highlights the justifications for the decisions and choices made for each of the proposed areas of investigation:

Justifications	Analysis/Conclusion
Reducing Cognitive Load for complex VM provisioning and management tasks.	Working to reduce complexity of VM provisioning and management by developing high levels of automation, reduced requirements for human inputs and improved automation (Sweller, 1988).
Improved Automation.	Working towards full automation and minimal human intervention for any functional procedures, such as VM provisioning, VM performance monitoring and migration (Benet et al, 2016 ; Steinder et al, 2007).
Selection of most appropriate AI strategies.	Use of natural language process to aid understanding of log/textual outputs, complimented by the selection of text-based analysis for improved/automatic pattern recognition, event processing and selection of forward chaining to reach facts (Anicic, 2009 ; Mettrey, 1991).
Simplification of VM deployment.	Minimisation of end-user inputs and full automation of provisioning VMs (Oakes et al, 2016).

Table 2.6 Research Justifications

Based on the justifications described in table [2.6](#), it is feasible to move forward into the methodology in chapter [3](#), which describes in more detail the experiment processes and mechanisms used to evaluate the IDE and other comparative virtualised management platforms.

Chapter 3: Methodology and Evaluation Strategy

3.1 Methodology Introduction

The methods used for the investigation work include the build and development of a prototype laboratory environment to support the experimentation processes undertaken for the IDE and associated functions. By using this platform, it enables functional testing of all the infrastructure components in unison and allows development of program code, algorithms and system interactions. It is envisaged that once the platform reaches a mature configuration point, the build could be easily replicated using automatic system package type installation on standard Linux type systems. This would enable the easy deployment of additional evaluation systems that are effectively replicas of the initial primary system. In this way, the evaluation processes can be carried out easily, without transporting excessive amounts of hardware and system configuration data (from the development laboratory). Simply, this could be a set of software components for:

- A software package to configure the IDE with the primary, secondary and tertiary systems.
- A software package to configure the Network Attached Storage (NAS) appliance.
- A software package for all required local source/packaged repository software.
- A software package to allow platform internet access (direct, or via a proxy).

Using these software deployment packages, it should be feasible to easily replicate the experimental development platform, assuming the standard hardware devices are physically available:

- At least 3 x86 architecture computers (Compute, minimum: 8GB memory, 2 internal disks, 2 CPU Cores at 2GHz or higher, 1 x 1Gb Network Interface).
- At least 1 Network Attached Storage Appliance (Storage, minimum: Dual 1Gb network, 4 disks, 7200RPM, SATA/SAS/FC).

- At least 2 Network Switches (Network, minimum: 16 x 1Gb ports).
- At least 1 x86 computer to act as a router/gateway for internet access (optional); this must have at least 1 physical network interface and a wireless network interface. (Network/Compute, minimum: 4GB memory, 1 internal disk, 2 CPU Cores at 2GHz or higher, 1 x 1Gb Network Interface).

3.2 Development Framework

The development framework is very important for the project to progress; the aim is to control code releases using the Redhat Hat Package (RPM) format and source control versions appropriately. This method will allow control of four key RPM software bundles (listed above), which will be version tested together and the results recorded, to build up a valid laboratory set of working configurations. It can be summarised as follows:

- RPM – All software will be bundled into Package format for ease of installation and distribution.
- Source code – all code will be version controlled in a system.

3.2.1 Laboratory Setup

The laboratory setup for the design, build and experimentation phase included setting up an initially small scale set of systems; the approach taken was to build a single x86 IDE server – the primary system which controls all aspects of the environment. In the final model design, there will be a primary and secondary system to provide high availability. Further to this, three other x86 systems are required to perform normal VM build, development and test operations. This platform would allow for all activities to be carried out on a small scale, with the view of acquiring more powerful systems further into the research process, for example, when experiments demanded higher performing systems specifications. Below is a diagram of the initial and final laboratory setup:

i) The initial simple configuration:

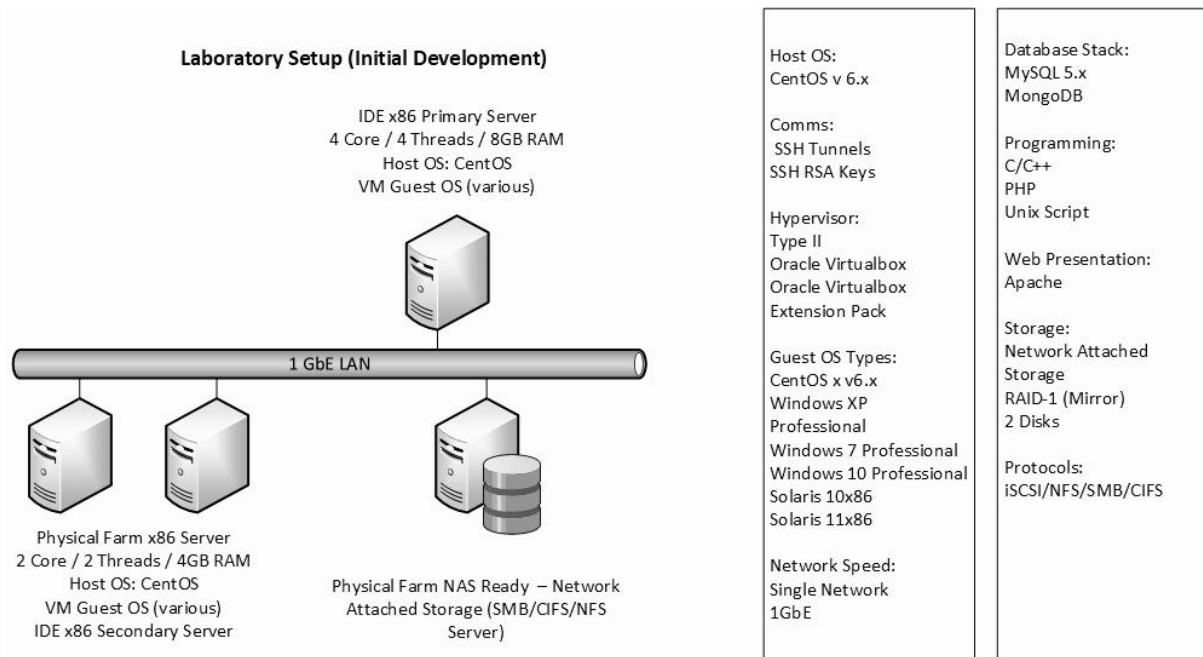


Figure 3.1 Initial Laboratory Setup

ii) The final configuration as a recommended minimum:

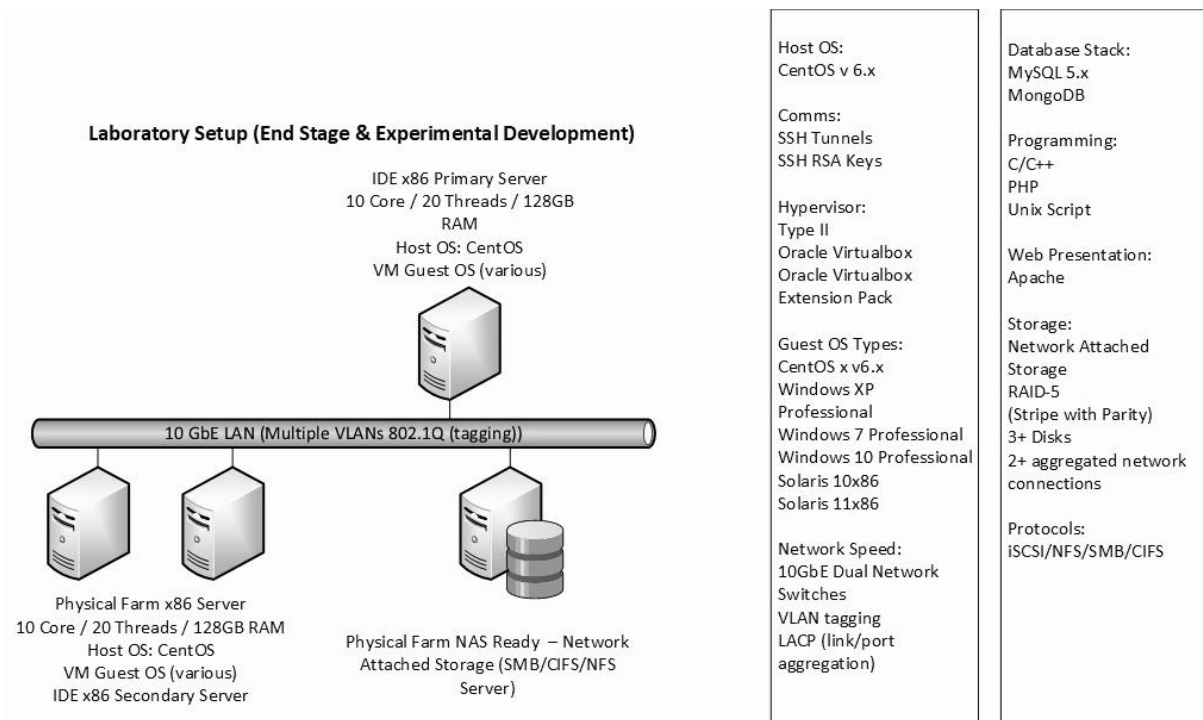


Figure 3.2 Final Minimum Laboratory Setup

3.2.2 Software Configuration

The software configuration of the IDE platform is fully automated; there exists a number of software Redhat Packages (RPMs) that make up the core of the programs required for the system to function. Please see [Appendix B](#) for full details.

3.3 Evaluation Strategy

3.3.1 Evaluation Approaches

There are several different approaches that can be used in evaluating complex expert systems. In the case of the IDE and its associated functional components, a combination of two approaches have been chosen to provide both empirical evidence comprising of qualitative data based on user feedback, and formal performance metrics providing measured outputs made up of quantitative data. It is the goal of this work to produce results output that have a combination of qualitative and quantitative methods. Some effort has also been taken to evaluate and test methods to convert qualitative feedback and accurately define and assign numerical values to help better represent user feedback in charts and graph format. Carefully constructed process and thought are placed into such a method to ensure fair, robust and meaningful values are accurately gathered and presented. Section [5.2.3.1](#) explains this approach in detail ([Srnka and Koeszegi, 2007](#)).

Therefore, the two combinations are summarised below:

- The first evaluation method will use qualitative methods collated by obtaining expert user feedback via interviews. This will be obtained through live demonstrations similar to the RAND (Research AND Development) framework methodology ([Rothenburg et al, 1987](#)). Therefore, a collection of expert qualitative data will be created based on a set of structured interview questions, through interviewing three independent groups with a defined experience and capability rating in the field of computer-based virtualisation (see section [5.2.1.2](#) for definitions).

- The second evaluation method will use several quantitative simulations with formal metrics, that will specifically target key system functions in performance aspects, such as deployment of virtual machines and migration of virtual machines ([Chen and Suen, 1993](#)).

The objective is to use a combination of complimentary methods. Thus, the desired outcome is to create a richer and more robust evaluation methodology by using both qualitative and quantitative data. Qualitative data that focuses and reflects on the characteristics and capabilities of the system, rather than just its features alone, and quantitative data that measures specific system functions. The following sections discuss these two approaches in more detail.

3.3.2 Expert System Evaluation

In chapter [2](#), section [2.4](#) an examination of how real life examples of expert systems can be used as a way to assist with automated complex decision making processes is discussed. It was established that such systems have a broad application to problem solving, and the examples included medical diagnosis assistance, and system configurators. Section [2.4.9](#) provided the basis and justification as to why expert systems had been considered as a feasible way of utilising expert knowledge to manage complex systems and processes. A variety of existing and known methods are used to evaluate the IDE and comparative systems, using experimentation process; they are described further below:

- The expert system rules are fired and tested using a simple first-come first-served approach, and are therefore ordered in priority. This avoids complex conflict sets, whereby many rules may be valid to execute, and provides a clear conflict resolution strategy ([Mettrey, 1991](#); [Alty and Coombs, 1984](#)).
- Additionally, a mixture of evaluation methods is employed to test the IDE. This includes, traditional qualitative and quantitative evaluation methods, as described in section [3.4.1](#) and [3.4.2](#) respectively, as well as less well known methods such as measuring the characteristics and features of expert systems and their capability ([Beckman, 1990](#); [Rothenburg et al, 1987](#)).

- Mixed quantitative and qualitative methods are used to measure VM and cognitive load, and are used with the end-user participant groups to test the IDE expert system. The experiments are devised in section [3.5.1](#) for investigation 1, Autonomous VM Deployment, and in section [3.5.2](#) investigation 2, Cognitive Complexity System Evaluation ([Massimiliano and Tamburri, 2017](#)).
- Quantitative methods are employed in section [3.5.3](#) for investigation 3, Workload Migration and Evacuation of VMs to investigate the IDE performance against comparative systems ([Madarasz et al, 2014](#)).
- For the performance management evaluation process, section [3.5.4](#) describes investigation 4, the Overload of VM Memory, and section [3.5.5](#) investigation 5 the Overload of VM CPU. A series of simulated experiments are to be conducted to evaluate how well the global resource manager for the IDE performs in comparison to other similar studies. As an extension to testing the effectiveness of all the resource management systems, a binomial evaluation is used in chapter [7.6.5](#) to enrich and provide details on the features and characteristics of the IDE and comparative systems ([Conrath and Sharma, 1991](#)).

As demonstrated above, the author is using a wide variety of standard evaluation methods (qualitative or quantitative), which are well known, tried and tested. As alluded to, there are several less well known approaches used by some researchers, however, for the most part these have been avoided unless it was apparent additional methods were needed or useful, as in the case where qualitative or quantitative methods would not suffice entirely; for example determining the effectiveness and capability of the global resource management features of a system like the IDE. Therefore, with the exception of some influences from researchers such as Conrath and Sharma and their use of the binomial evaluation method, and Rothenburg and his colleagues at the RAND institute, the methods remain standard where possible.

As described earlier in section [3.3.1](#) the RAND work establishes some other useful

methods for evaluating expert systems, which have been partly considered and incorporated ([Rothenburg et al, 1987](#)). One of the points suggested as providing additional value is evaluating the expert system capabilities, and that is explored further as part of this work. In chapter [7](#), section [7.6.1](#), [7.6.2](#) and [7.6.3](#), the IDE and comparative tools are analysed, scored, and the results presented.

3.3.3 Experiment Design

The following points detail how and why the experiments were devised:

- Experiment 1 (section [3.5.1](#)) and Experiment 2 (section [3.5.2](#)): Autonomous VM deployment, and Cognitive Complexity System Evaluation. This experiment was created and based around the 10-step procedure to provision VMs. The author used empirical observation methods to initially work through each of the key stages (10-Steps), to determine what inputs are required and necessary for a system to create and provision a VM, for example selecting the required CPU, memory, and disk parameters. The process methods to achieve this is consistent for the IDE, AWS and Oracle platforms, and hence repeatable for each ([Massimiliano and Tamburri, 2017](#); [Bhise and Mali, 2013](#)).
- Experiment 3 (section [3.5.3](#)): Workload Migration and Evacuation of VMs. The author realised early on at the proposal stage, that this area was a key element to management. Similar studies have been done using other technologies, such as VMWare vMotion and XenMotion. Migration, evacuation and failover of VMs are key to maintaining availability, and therefore can be considered a fundamental process for the intelligent management of virtualised platforms ([Benet et al, 2016](#); [Feng et al, 2011](#); [Shirinbab et al, 2016](#); [Toyoshima et al, 2010](#); [Calzolari, 2006](#); [Wood, 2011](#)).
- Experiment 4 (section [3.5.4](#)) and Experiment 5 (section [3.5.5](#)): Overload of VM Memory Usage, Detection Time and Resolution Time, and Overload of CPU Memory Usage, Detection Time and Resolution Time. As with VM provisioning, and migration

and failover, this is another critical area for the effective control of virtual machines. Utilising a global performance management strategy for the flexible consumption of CPU and memory is regarded as important feature of an intelligent management system ([Flinta et al 2017](#); [Imai et al 2013](#); [Jeong and Lee 2012](#); [Jing 2011](#)).

3.4 Qualitative Versus Quantitative Methods

There was a desire to provide a mixture of both qualitative and quantitative studies to enhance the overall data set, with the objective of providing a richer set of results for evaluation. This is backed up by the concept of utilising mixed-methods. The aim of such mixed-methods is to support the cause and effect claims (analysis and conclusion) by combining multiple types of data, from various sources, to allow analyses that provide software practitioners and academics a solid rationale, balanced and practical value to the research results and conclusions reached ([Massimiliano and Tamburri, 2017](#)).

3.4.1 Qualitative Evaluation

For the five experiments, it was not deemed appropriate to use this method for all as described by the following table, primarily because of the type of measurements that could be taken require some form of human interaction:

Experiment No	Experiment Description	Qualitative Evaluation (Yes/No)	Justification of Method
1	Simplified VM provisioning.	Yes	Interactive systems such as the AWS, Oracle and IDE public/private cloud platforms allow for direct user interaction and experience using the system interface,

Experiment No	Experiment Description	Qualitative Evaluation (Yes/No)	Justification of Method
			<p>albeit being automated for many of the provisioning features. This therefore allows opportunity for structured qualitative user feedback.</p> <p>Justification: Human Interactive.</p>
2	VM Provisioning cognitive evaluation performance.	Yes	<p>Using structured qualitative feedback from the end-users of the system provides the ability to create a model to collate the user experiences and convert that data from words to numbers. (Srnlca and Koeszegi, 2007).</p> <p>Justification: Human Interactive.</p>
3	Workload Migration and Evacuation of VMs.	No	<p>This feature within the tested platforms is an automatic system event, whereby it detects a full VM failure, and works to evacuate and migrate the resource to a remaining healthy host. This means such an event/task can be easily observed and quantified, but as it does not involve the end-user experience as such, there is no possibility to extract qualitative data.</p> <p>Justification: Non-human Interactive.</p>
4	Overload of VM memory usage, detection Time,	No	<p>This type of event is detected automatically by the platform and steps are taken to resolve. This experiment does</p>

Experiment No	Experiment Description	Qualitative Evaluation (Yes/No)	Justification of Method
	and resolution time.		not involve the end-user and involves observations of the platform behaviour only. Justification: Non-human Interactive.
5	Overload of VM CPU usage, detection time, and resolution time.	No	This type of event is detected automatically by the platform and steps are taken to resolve. This experiment does not involve the end-user and involves observations of the platform behaviour only. Justification: Non-human Interactive.

Table 3.1 Qualitative Experiment Methods

3.4.2 Quantitative Evaluation

For the five experiments, it was appropriate to use this method for all as described by the following table, primarily because for each, there was opportunity to collect meaningful measurable data:

Experiment No	Experiment Description	Quantitative Evaluation (Yes/No)	Justification of Method
1	Simplified VM provisioning.	Yes	Data is collected from the provisioning process for all platforms in respect to the timings for each of the 10-steps identified. Section 5.2.1 explains in further detail.

Experiment No	Experiment Description	Quantitative Evaluation (Yes/No)	Justification of Method
			Justification: Measurable metrics available.
2	VM provisioning cognitive evaluation performance.	Yes	Qualitative data is taken from the end-users of the VM provisioning process and converted into quantitative data, to provide a mix-method analysis (Srnka and Koeszegi, 2007 ; Massimiliano and Tamburri, 2017). Justification: Measurable metrics available.
3	Workload migration and evacuation of VMs	Yes	This experiment phase will support the detection process for a VM failure and subsequent evacuation and migration to a remaining healthy host. The timing related data associated with this process will be available. Justification: Measurable metrics available.
4	Overload of VM memory usage, detection time, and resolution time.	Yes	This type of event is detected automatically by the platform and steps are taken to resolve. This experiment will involve being able to time the observations of the platform behaviour in respect to how it can (dynamically) balloon a VMs memory and resolve any resource issue. Justification: Measurable metrics available.
5	Overload of VM CPU usage,	Yes	This type of event is detected automatically by the platform and steps

Experiment No	Experiment Description	Quantitative Evaluation (Yes/No)	Justification of Method
	detection time, and resolution time.		are taken to resolve. This experiment will involve being able to time the observations of the platform behaviour in respect to how it can (dynamically) increase a VMs CPU allocation and resolve any resource issue. Justification: Measurable metrics available.

Table 3.2 Quantitative Experiment Methods

3.4.3 Data Analysis

This will be conducted using well-known established methods ([Xu and Liu, 2003](#); [Madarasz et al, 2014](#)):

- Providing three user groups to represent a combined total of ninety-three users for the IDE, AWS and Oracle platform experiments made up of, thirty-one novice users, thirty-one experienced and thirty-one expert users. See section [5.2.2.2](#), [5.2.2.3](#) and [5.2.2.4](#) respectively for details on how these groups are defined. Having a reasonable sized set of controlled end-user groups will provide a wider and richer set of results for analysis, and improves the reliability of subsequent drawn conclusions.
- Using observation techniques to record events, outcomes and timings during the experiment process.
- Comparison of data against multiple similar academic studies using comparative tools; for example, the IDE provisioning with AWS and Oracle cloud provisioning techniques.

- Recording the data associated with timed experiments, to produce tables and charts to allow for visual representation in graphical form.
- Interpretation of statistical data using mathematical methods, for example determining sum or averages such as the mean, mode and median for substantiating results, outcomes and conclusions.

3.5 Evaluation of Comparative Systems

The following five experiments capture the fundamental processes the IDE aims to deliver against, as per the initial project proposal – see Appendix [G](#) for more details. They are undertaken in a controlled way and use a simulated, step by step approach to record results.

3.5.1 Investigation 1: Autonomous VM Deployment

The following mechanism is designed to evaluate the deployment process of VMs. It involves a study of the time taken to create a virtual machine compared to other case studies. This would include the process time to evaluate cloud build questions/response against the automation processes of the IDE.

Experiment Flow	Process Description	Methodology	Result
Access the provisioning system.	The end-user must be able to access the provisioning platforms to provide an interface to produce the deployed VMs.	Browser based access via the internet or private cloud system via a local network.	IDE v AWS v Oracle Record the time taken to perform (e.g. gain access to the system and authenticate).
Configure role.	The end-user must have the relevant access control and permission to	Observe and allow the system to configure	IDE v AWS v Oracle Record the time

Experiment Flow	Process Description	Methodology	Result
	create VMs.	appropriately.	taken to perform the configuration.
Select compute as the option for VM deployment.	Use the VM (Compute) provisioning process via the BUI.	The end-user must be able to locate the compute provisioning mechanism.	IDE v AWS v Oracle - record the time taken to access the VM provisioning tool.
Select the image you wish to use to install to the VM (OS type/version).	There must be a data source to install an OS image.	The end-user must be able to locate an appropriate data install source.	IDE v AWS v Oracle - record the time taken to access the data source.
Select the VM CPU, memory, and disk parameters.	The VM must have parameters associated with its configuration.	The end-user must specify appropriate CPU, memory and disk values.	IDE v AWS v Oracle - record the time taken to provide the VM shell parameters.
Define VM parameters.	The VM requires IP configuration, software packages and OS release version specified.	All the additional parameters must be provided to the provisioning system.	IDE v AWS v Oracle - record the time taken to provide the additional parameters.
Define VM storage.	The VM requires at least one virtual disk associated.	All disk devices must be defined for the VM to use.	IDE v AWS v Oracle - record the time taken to provide the disk information

Experiment Flow	Process Description	Methodology	Result
			parameters.
Add SSH Key, create a key and upload the public key.	To access the VM, an appropriate secure key mechanism must be in place to allow the user to access the VM post deployment.	The public and private key must be deployed to the system to allow access.	IDE v AWS v Oracle - record the time taken to setup and provide the key to allow access.
VM creation process.	This is the actual time taken to install and configure the VM using the OS data source.	Observe the installation process mechanism.	IDE v AWS v Oracle - record the time taken to install the OS and provision the VM.
Process for accessing the VM via the internet, or via network.	The VM must be accessible via the network/firewalls post install. Therefore, it must be available over the network or internet.	All network access protocols like SSH must be working; the end-user must be able to login to the VM.	IDE v AWS v Oracle - record the time taken to access and login to the VM.

Table 3.3 VM Deployment Experiment

3.5.2 Investigation 2: Cognitive Complexity System Evaluation

The steps below in table [3.4](#) will be followed to formally evaluate the VM provisioning process using a structured feedback survey from the three groups, namely those characterised into groups of novice, experienced and expert users:

Experiment Flow	VM Data Parameter Gathering	Methodology	Result
Obtain VM hostname and size classification.	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle - observe the complexity for each end-user and record results.
Obtain VM size parameters (CPU/Memory/Disk).	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle - observe the complexity for each end-user and record results.
VM shell creation.	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle - observe the complexity for each end-user and record results.
VM guest installation.	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle – observe the complexity for each end-user and record results.
VM Post installation methods.	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle - observe the complexity for each end-user and record results.

Experiment Flow	VM Data Parameter Gathering	Methodology	Result
VM access method and authentication.	Data source method.	Automatic or manual (description).	IDE v AWS v Oracle – observe the complexity for each end-user and record results.

Table 3.4 VM Deployment Cognitive Load Experiment

3.5.3 Investigation 3: Workload Migration and Evacuation of VMs

This experiment set analyses the following three scenarios listed in table [3.5](#):

- When does a VM need to evacuate?
- Time policy, critical services weighting; for example, when moving a VM of least importance (i.e. a VM hosting non critical applications).
- Scenario based evaluation of the IDE VM migration versus XenMotion and VMWare’s vMotion.
- Utilisation strategy percentage use of resources, detection time, inference and subsequent actions.

Experiment Flow	Process Description	Methodology	Result
Evacuation Scenario 1: VM failure.	Simulation of VM failure and subsequent actions.	Simulate VM failure event, ensure VM fails, then observe the resulting actions to the event.	IDE versus XenMotion and VMWare’s vMotion. Record the failure time, VM failure time and VM restored

Experiment Flow	Process Description	Methodology	Result
			time.
Evacuation Scenario 2: Physical Host failure.	Simulation of physical host failure and subsequent actions.	Simulate physical host failure event, ensure VM fails, then observe the resulting actions to the event.	IDE versus XenMotion and VMWare's vMotion. Record the failure time, VM failure time and VM restored time.
Evacuation Scenario 3: Redistribution and equalisation of platform VM load based against defined SLA.	Simulation of platform resource view with distributed load with some physicals overloaded with VMs, with high CPU/memory load, and with some physicals underutilised. The expectation is a controlled redistribution of resources subject to no impact to agreed SLAs.	Simulate a platform with three physical hosts and a defined number of VMs. Create the scenario of one overloaded physical system, one within normal operating levels and one under-utilised. Demonstrate the behaviour of each platform under the scenario-controlled conditions.	IDE versus XenMotion and VMWare's vMotion. Record the start time of the scenario, create the situation and the observe the redistribution of load and the platform actions. Determine the overall result for each platform, in terms of the final configuration and overall distributed load (ideally evenly loaded systems).

Table 3.5 VM Evacuation, Workload Migration and Load Management Experiment

3.5.4 Investigation 4: Overload of VM Memory Usage, Detection Time, and Resolution Time

Simulate memory use to threshold (based on expert standards), detect, invoke actions:

Experiment Flow	Process Description	Methodology	Result
Simulate VM memory load over a set period of time (defined in minutes).	Load simulator to drive the memory load of a single VM to create overload (for example over 75%, for a period >5 minutes).	Deploy a VM, simulate the load on the system against memory, and perform controlled experiment.	IDE versus XenMotion and VMWare’s vMotion. Record the overload process experiment, memory values, and detection time of the event occurring, and resolution time – these would be the systems’ automatic resolution steps.

Table 3.6 VM Memory Overload, Detection and Resolution Experiment

3.5.5 Investigation 5: Overload of VM CPU usage, Detection Time, and Resolution Time

Simulate CPU clock time to threshold (based on expert standards), detect, and invoke actions:

Experiment Flow	Process Description	Methodology	Result
Simulate VM CPU load over a set period of time (defined in minutes).	Load Simulator to drive the CPU load of a single VM to create overload (for example over 75%, for a period >5 minutes).	Deploy a VM, simulate the load on the system against CPU, and perform controlled experiment.	IDE versus XenMotion and VMWare's vMotion. Record the overload process experiment, CPU values, and detection time of the event occurring, and resolution time – these would be the systems' automatic resolution steps.

Table 3.7 VM CPU Overload, Detection and Resolution Experiment

3.6 Summary

At the beginning of the chapter the development framework is introduced, as a means for supporting the IDE system. It proposes a method for deploying a suitable laboratory setup as well as software configuration, to enable the delivery of the IDE experimentation process. A mixed approach is used for evaluating the systems under investigation, using empirical evidence collected through observational data from end-users participants. The data sets collected are to be analysed using quantitative and qualitative methods to ensure the output results are as rich and diverse as possible, in terms of being able to analyse measurable processes and interpret the cognitive end-user experience during system use. The work comprises of five principle investigations, in the areas of autonomous VM deployments, cognitive load analysis, workload migration and failover methods, and global resource management of memory and CPU resources. The next chapter discusses the characteristics

and components of the IDE framework and the way in which it is used to enhance the management of virtualised computer based systems and workloads.

Chapter 4: The Intelligent Decision Engine

4.1 Introducing the Intelligent Decision Engine

The potential for automated intelligent systems being able to interface into complex computer infrastructures, poses an opportunity to vastly improve the control, management and end-user experience of virtualised computing platforms. Having the ability to leverage an Artificial Intelligence (AI) system to enable this is a possible way of accomplishing a primary project objective (Appendix [G](#)). As part of this investigation, there has been an emphasis on development work to create an Intelligent Decision Engine (IDE), to assist managing virtualised computer systems; typically, this would include the control of:

1. Data-storage, memory and information retrieval.
2. Data processing and organisation.
3. Data flows between systems.
4. Creating intelligent rules and procedures.
5. System self-management and self-learning.
6. System real-time data processing and decision making.
7. System availability and autonomy (High Availability and recovery).

As part of the objectives of the proposal, this included a detailed analysis of other intelligent computerised management systems available (see section [3.5](#)). While the list of systems is quite extensive, the author has concentrated on AWS to attempt to compare and contrast the methods and processes, against the area of research being conducted within this thesis.

For this research proposal it is not feasible to address all the areas Amazon Web Services (AWS) currently span, or any other mainstream cloud-based provider, such as Oracle, IBM or Rackspace ([Finkle and Scoresby, 2012](#); [Hwang, 2015](#); [Ullah et al, 2016](#)). While the investigative work does cover some wider areas, by way of re-focusing, the author's proposal includes three specific areas of analysis and development.

- The IDE - Systems Intelligent Management; creating an intelligent method of deploying and managing VMs, including the development of highly automated programmed methods and algorithms, using different data storage strategies.
- Workload migration – the movement of virtualised computer resources based on performance metrics and ensuring High Availability (HA) of VMs.
- Systems performance and health monitoring – establishing metrics around virtualised resources and interpreting large amounts of real-time data, allowing it to feed into the IDE for processing.

4.2. IDE Characteristics

The following sections examine fundamental areas of investigation, and the traits developed for the IDE. A detailed explanation is provided below around the characteristics that have been designed and inbuilt into the expert system to assist with the automatic management of virtualised computer-based platforms.

4.2.1 Data Organisation

Organisation of information is critical for the effective management, storage, retrieval, processing and intelligent machine decision making. The IDE uses the following combination to organise data:

- A shared structured filesystem, with data files stored on a NAS system. This allows instant access to data from any cluster node.
- A Relational database to maintain long-term information, knowledge rules, metadata, and statistics.

4.2.2 Decision Making

The ability for the IDE to be able to perform decision making is critical. The methods for processing and decision making are comprised around the forward-chaining method. This has been selected primarily for the following fundamental reasons:

- The ability to be driven from events occurring.
- The ability to chain events together, to lead to a conclusion and consequent action.
- The ability to reason toward a goal, rather than from one (backward chaining).
- Backward chain reasoning something that is discussed further in section [8.3.6](#).

4.2.3 System Learning

The concept that the system can analyse patterns and learn from their available data set, is a possibility for intelligent machines. Computerised systems are effective at handling large amounts of I/O (Input and Output). Such systems invariably generate lots of information and data. Being able to manage the data and store it in a meaningful way represents a challenge. Indeed, many organisations are now investing in big data analytics using computer software packages designed to make sense of vast amounts of data, such as log files, access lists, error logs and many other types of stored information ([Jin et al, 2016](#)).

4.2.4 Algorithms and Procedures

The IDE uses the following algorithms, devised by the author, to enable the system to make intelligent based decisions. These are inbuilt into the system to enable the experimentation phase to compare against other systems and platforms selected (see section [3.5](#) Comparative Systems). The definition of an algorithm is “a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem” ([Cambridge Advanced Learner's Dictionary, 2019](#)). The IDE’s principle algorithms and procedures are explained in detail below:

- Algorithm/Procedure 1: Remote system discovery mechanism, with system OS fingerprint analysis and advanced OS system type detection. This algorithm allows for the discovery of systems on the network, within managed subnet ranges, which can be brought under IDE control. This does assume that the system hypervisor matches those that the IDE presently understands, for example VirtualBox. Additionally, a network scanner, such as nmap can be utilised with this algorithm to fingerprint the remote system OS. Once a system is detected, scanned and a connection tested successfully, the IDE can use an initial set of credentials to place its SSH public key securely on the remote system to allow full control from that point forward.

```

// High level discovery and analysis algorithm

INPUT: network scan range, and all known hosts
OUTPUT: Return all remote host values, fingerprints and status

FOR each network
  FOR each IP
    Scan IP address
    Use ICMP protocol stack to establish TCP/IP connectivity status
    Use network analyser to finger print analyse any unknown hosts
    Use SSH process to determine access status
    Establish access and control if able
    RETURN (return code)
  END FOR
END FOR

FOR each discovered host
  Use SSH process to determine access status
  Establish access and control if able
  RETURN (return code)
END FOR

// End of algorithm

```

Table 4.1 Algorithm/Procedure 1: Remote System Discovery

- Algorithm/Procedure 2: Improved system communication strategy using SSH to build a secure framework for remote host management and control groups. The SSH framework allows for commands, code and scripts to be executed automatically using key exchanges for authorisation across remotely controlled systems. Return codes are received back to the IDE master to enable it to determine the outcome of executed commands.

```

// High level command execution/messaging algorithm

INPUT: all known controlled hosts, or a subset of controlled hosts
OUTPUT: command status return codes

FOR each host
    Use SSH framework to execute remote command/script/code
    RETURN (return code)
END FOR
// End of algorithm

```

Table 4.2 Algorithm/Procedure 2: Messaging Command Process

- Algorithm/Procedure 3: Improved data extraction and analysis methods to enable two methods of a) quick response and b) slower background analysis of environment data, to allow for reference knowledge data to be added and cleansed. This procedure allows the IDE to probe all the remote managed system text files identified in section [4.8.4](#). The algorithm works through each identified knowledge source and will check for certain known patterns and keywords. The relevant data is extracted locally, and is sent back for centralised processing.

```

// High level text mining algorithm

INPUT: all known controlled hosts, or a subset of controlled hosts
OUTPUT: key message string(s), criticality

FOR each host
    Text mine for all IDE key phrases of interest against all files
    FOR each file
        Analyse IDE knowledge base (thesaurus, lexicon, and grammatical entities) against
        Key phrases
        Check each file status, check for text-based files of interest
        Auto process for key phrases
        Extract, compress and return data collection/information
        RETURN (return code)
    END FOR
END FOR

// End of algorithm

```

Table 4.3 Algorithm/Procedure 3: Text Mining

- Algorithm/Procedure 4: Information and knowledge organisation to process and create core and reference data, which affects how the forward chaining mechanisms work when the IDE is decision making. The IDE makes specific use of two distinct data store systems for unstructured, and for structured data. Each data store also retains a filesystem cache for high speed data access to recently accessed or used data. Thus, depending on the type of data, determines where it is stored and retained.

```
// High level data organisation algorithm  
  
INPUT: datafile (structured/unstructured)  
OUTPUT: process completed flag  
  
FOR each datafile  
    IF data is structured THEN  
        Organise data into structured store  
        Place in data cache for analysis  
    ENDIF  
    IF data is unstructured THEN  
        Organise data into non-structured store  
        Place in data cache for analysis  
    ENDIF  
    RETURN (return code)  
END FOR  
  
// End of algorithm
```

Table 4.4 Algorithm/Procedure 4: Data Organisation

- Algorithm/Procedure 5: Pattern analysis and learning from data. The intention is to use this mechanism to create new knowledge rules based on previously unidentified triggers, that may not be initially dealt with effectively by the existing minimalist ruleset. This particular algorithm is a precursor leading to the experimental process required to develop the additional work identified in section [8.3.6](#), regarding the invention and build of new knowledge rules.

```

// High level pattern analysis and learning algorithm

INPUT: data cache (structured/unstructured)
OUTPUT: process completed flag

FOR each data store
  WHILE data cache not empty
    IF data is structured THEN
      Review tables and data and match against known triggers
      IF new trigger required THEN
        Determine trigger type
        Create trigger
      ENDIF
      Review tables and data and match against known conditions
      IF new condition required THEN
        Create condition set (condition1) ... (condition x)
        Link trigger event
      ENDIF
    ENDIF
    IF data is unstructured THEN
      Create new objects
      Create new tree structures
    ENDIF
  RETURN (return code)
END WHILE
END FOR

// End of algorithm

```

Table 4.5 Algorithm/Procedure 5: Pattern Analysis and Learning

- Algorithm/Procedure 6: Knowledge based forward chaining. This procedure creates the framework for connecting either single or multiple system or platform events together, and then enabling the evaluation of the appropriate conditions as listed in section [4.8.1](#), then subsequently calling the necessary consequent to remediate the condition.

```

// Forward chaining algorithm

INPUT: real time and trailing data feed
OUTPUT: forward chaining result

// Dynamic Forward Chaining
WHILE (conditions)
  Reference Known Conditions AND analyse for triggers
  IF (condition 1) and (condition 2) ... (condition x) THEN
    Result (x) AND Execute trigger actions
  ENDIF
  RETURN (return code)
END WHILE

// End of algorithm

```

Table 4.6 Algorithm/Procedure 6: Forward Chaining

- Algorithm/Procedure 7: VM deployment mechanism used by the IDE builds forward chained rules, to allow it to provision and deliver VMs. This procedure directly automates the build and delivery of VMs, as described by section 5.2 Simplified VM Provisioning ([Oakes et al, 2016](#)).

```

// VM shell deployment algorithm

INPUT: VM size parameter, VM type parameter
OUTPUT: Return configured/built VM

FOR each VM
  FUNCTION Lookup VM values (condition1, condition 2)
    IF VM (standard configuration)
      FOR each value
        Allocate VM parameter
      END FOR
    ELSE (custom)
      Allocate hard VM parameters
    END IF
  END FUNCTION

  FUNCTION Provision VM Shell (Hostname, CPU, Memory, OS Disk, Application Disk, Data Disk,
  CPU execution Cap)
    FOR each value
      Add shell value
    END FOR
    Write shell values
    Commit shell
  END FUNCTION

```

```

FUNCTION Install VM
  Initialise VM shell
  Connect boot ISO to VM shell
  Network install VM
END FUNCTION

FUNCTION Post Configure VM
  Execution post install configuration
  Reboot VM
  Health Check VM
END FUNCTION
RETURN (return code)
END FOR

// End of algorithm

```

Table 4.7 Algorithm/Procedure 7: VM Deployment

- Algorithm/Procedure 8: VM performance and monitoring management (preliminary). This was the initial procedure defined to work generically with identified system thresholds described in section 4.8. Chapter 7 builds on this, and creates an extended or enhanced algorithm in section 7.2.1 for managing CPU and memory resource.

```

// VM performance and monitoring algorithm

INPUT: VM host
OUTPUT: VM performance and health status

FOR each VM
  IF VM down THEN
    Invoke recovery processes
  ENDIF
  IF VM performance > system alert thresholds THEN
    WHILE VM performance degraded
      Invoke analysis against knowledge base
      Invoke performance improvement processes
      check performance
    END WHILE
  ENDIF
  RETURN (return code)
END FOR

// End of algorithm

```

Table 4.8 Algorithm/Procedure 8: Preliminary Performance Monitoring

- Algorithm/Procedure 9: Real-time platform event trigger, with decision processing-based delivery event response. This algorithm addresses trigger alert events, which are manifest through the search and evaluation process of live data sources as described in section [4.6.4](#), and the text analysis for keywords and patterns listed in section [4.8.4](#). By utilising these two techniques, the procedure below allows for rule matching and forward-chaining functions to execute and fire knowledge rules as appropriate.

```

// Platform event trigger algorithm

INPUT: Event trap
OUTPUT: Event response

WHILE Trap events true
  FOR EACH Trap event
    Search and evaluate trap against knowledge base
    Match for trigger response
    IF match on trigger response THEN
      Perform platform response
    END IF
  RETURN (return code)
END FOR
END WHILE

// End of algorithm

```

Table 4.9 Algorithm/Procedure 9: Event Trigger and Decision Making

- Algorithm/Procedure 10: Self-monitoring and high availability (HA) features. The IDE system uses its own mechanism to maintain HA in the event that the primary IDE service is interrupted for an unexpected reason, such as a hardware or software failure. If this occurs the procedure below is invoked to bring services automatically back online on an alternative node in the quickest way possible. This mechanism uses the quorum algorithm defined in section [4.7.1](#), in order to establish a cluster majority, to avoid and mitigate against any split brain type scenario from occurring.

```

// Platform self-monitoring algorithm

INPUT: Quorum vote count, health check
OUTPUT: IDE response

IF IDE failure detected THEN
  IF (IDE primary down true AND Quorum votes >= Quorum value) THEN
    // when primary faults
    Failover from primary on to secondary
    Check Quorum, Failover and Primary server
    Alert response on failed component
  END IF
  IF (IDE primary up true AND IDE secondary failover up true AND Quorum votes >= Quorum value) THEN
    // when primary is repaired fallback
    Check Quorum, Failover and Primary server
    Rebalance cluster and failover IDE from secondary to primary
  END IF
  IF (Quorum votes < Quorum value) THEN
    // more than a single failure has occurred (no quorum reached)
    Check Quorum, Failover and Primary server
    Report critical IDE error
    Alert Response
    Manual Intervention to recover
  END IF
END IF
RETURN (return code)

// End of algorithm

```

Table 4.10 Algorithm/Procedure 10: Self-Monitoring

4.3 IDE Components

The diagram below outlines the high-level components for the IDE:

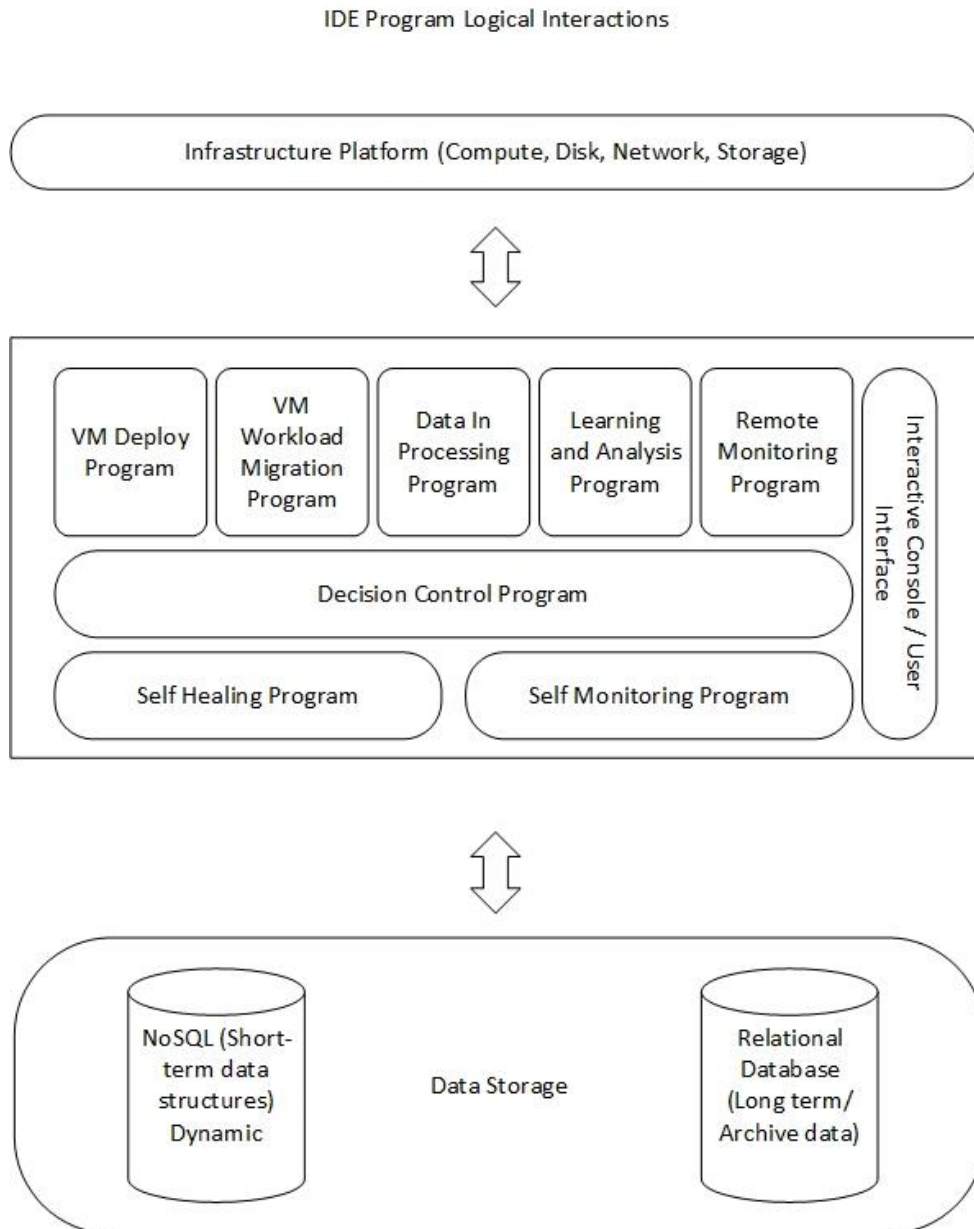


Figure 4.1 IDE Program Components

The concept ideas presented in figure [4.1](#) provide the basis and opportunity to create an IDE based on a set of functional computer programs. While intelligent programs can exist in an isolated environment, equally, they can also exist in a distributed and interactive environment; this in fact presents an opportunity to more closely mimic natural 'social interaction' with other systems, by sharing information between entities ([Callaos, 1994](#)).

Table [4.11](#) below, discusses the elements captured in figure [4.1](#) above and discusses in detail how these elements will interact with each component, and the envisaged benefits based on the research conducted in the field leading to a desired IDE end state.

Program Element	Description, Role and Envisaged Benefits
Decision Control Program.	<p>The Decision Control Program is at the very core of the IDE. This particular program needs to be efficient and potentially based on a typical Unix C like daemon program (Kwon, 2012). This program will construct dynamic decision tables based on data inputs and interacts with other programs defined in figure 4.1. Its function must include:</p> <ul style="list-style-type: none"> • Build dynamic decision-making tables (as necessary). • Suggest/Modify/Improve static decision tables (background). • Provision to process real-time data and interact with other programs effectively. • Analysis and rationale of old data from short and long term sources. • Use its compute facility effectively (scale up or down thresholds). • Handle interrupts, inputs and outputs.

Program Element	Description, Role and Envisaged Benefits
VM Deploy Program.	<p>The VM Deploy program implements an advanced unique one-click deploy mechanism (ref to paper), using a combination of advanced deployment tools. The program must allow:</p> <ul style="list-style-type: none"> • One-click web based VM provisioning. • Fully automated VM deployment.
VM Workload Migration Program.	<p>The VM Migration program handles the movement of VM resources, either to manage computer resources or as a result of failure of systems. This facility must:</p> <ul style="list-style-type: none"> • Migrate VMs as a result of performance issues/thresholds. • Migrate VMs as a result of hardware failure detection. • Migrate with minimal VM downtime. • Migrate in a fully automated way.
Data in Processing Program.	<p>The Data in Processing program is responsible for collecting and managing data inputs from the VM platform environment, storing it in an appropriate data-store (depending on defined criteria), either in:</p> <ul style="list-style-type: none"> • Short-term (NoSQL) / File-based. • Long-term (relational/archive). <p>and in the background handle:</p>

Program Element	Description, Role and Envisaged Benefits
	<ul style="list-style-type: none"> • Data transference. • Data archive (big data management).
<p>Learning and Analysis Program.</p>	<p>The Learning and Analysis program is initially responsible for developing small new components:</p> <ul style="list-style-type: none"> • Hints and tips based on data. • Data rules for dynamic decision tables. • VM analysis reports. • New system functionality invention/development. <p>Note, this last initiative is quite ambitious. While it does not make up a specific investigation for this project, it remains a desirable characteristic for the system.</p>
<p>Remote Monitoring Program.</p>	<p>The Remote Monitoring Program interacts with all VM elements on the virtualised platform. The program is able to continually and dynamically monitor all VM hosts and the overall platform health. Monitoring would include:</p> <ul style="list-style-type: none"> • Network monitoring using base utilities such as ICMP Ping. • System kernel and OS monitoring. • System log monitoring. • Performance monitoring (CPU, Memory, I/O). • Hypervisor health monitoring.
<p>Self-Monitoring Program.</p>	<p>Self-monitoring program is text interactive, includes a console feed, Browser User Interface (BUI) and is has the</p>

Program Element	Description, Role and Envisaged Benefits
	<p>function of monitoring the health of the suite of programs (Calzolari, 2006). Critically, there will always be at least three components running; a master instance, and two shadow instances, which must always run on separate physical hosts.</p> <p>The purpose of this is to:</p> <ul style="list-style-type: none"> • Ensure no Single Point of Failure (Spof) in existence in the system. • Ensure there is always one master instance running, and two shadow instances.
Self-Healing Program.	<p>The Self-Healing Program reacts to the self-monitoring program and takes corrective actions to ensure its continuing operation in the face of failure of a single or any number of components.</p> <ul style="list-style-type: none"> • Ensure there is a process defined to convert a shadow to master instance (and vice versa). • Ensure there is a facility to spawn new shadow instances where possible (for example in clusters greater than 3 physical hosts).

Table 4.11 IDE Program Function Suite

4.4 Defining the IDE Model

Below is diagram figure 4.2 showing the process and interaction of the decision engine, and its overall architecture and design:

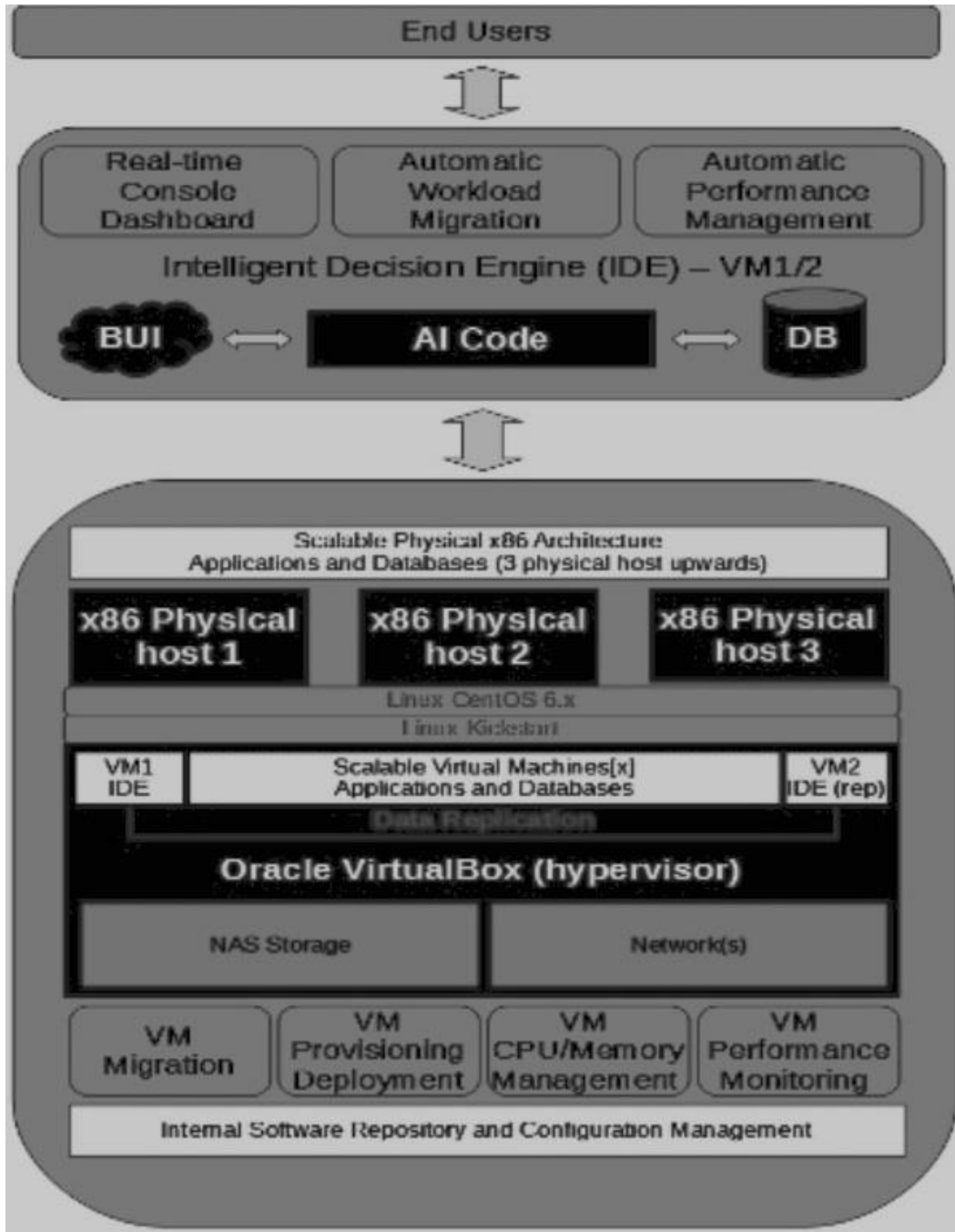


Figure 4.2 IDE Architecture Model

4.5 Data-storage, Memory and Information Retrieval

As with any intelligent system, information must be stored, ordered and arranged in such a way as to allow efficient location and retrieval ([Cattell, 2010](#)). Given the amount of information required for storage, there are several ways this can be achieved. Typically, in an intelligent system, there is usually a requirement for short-term (fast data access) and long-term memory (archive/slow data access). This requirement can be potentially solved and mapped within typical computing architectures to RAM (Random Access Memory) and disk storage, for example SAS (Serial Attached SCSI) or another such like device. As in the example of a human subject, short-term memory structures (frequently accessed) usually outperform long-term (infrequently accessed) memory and likewise in a computer system, the same principle holds true ([Sanzo et al, 2012](#); [Sweller, 1998](#)).

4.5.1 Long Term Storage Strategy

The author proposes a relational database mechanism such as MySQL as a long-term storage strategy ([Martin et al, 2007](#)). Section [4.2.1](#) provides additional details, along with [Appendix D](#).

4.5.2 Short and Medium Term Storage Strategy

The author proposes using a NoSQL, or bespoke file(s)-based solution using a mechanism such as MongoDB as a short-medium storage strategy ([Cattell, 2010](#)). Section [4.2.1](#) provides additional details, along with [Appendix D](#).

4.6 Data Processing and Organisation

The IDE stores and processes information continually and real-time. In order to perform these types of activities it needs to handle and organise the data effectively, using several mechanisms, depending on the type of task or activity the following methods are employed:

- A Relational Database structure – designed for long term storage on information such as cluster nodes, cluster configuration, IP and Network information, nodes information, and critical log warning and alerts. Additional knowledge rules are also stored here to enable reference.
- Custom File-based configuration and cache – designed to manage globally shared lookup data for activities such as recording the quorum vote, cluster health status, and acting as a general fast data cache. In memory structures are used in conjunction for very fast response and low latency tasks.

4.6.1 Data flows Between Systems

The following diagram illustrates the data flows between the systems and explains in detail how the IDE communicates with cluster nodes, and extracts information for processing and sends commands to remote nodes, based on the expert rules employed to manage the virtualised platform in an intelligent fashion. Below is a diagram showing the network protocol flows between the IDE systems:

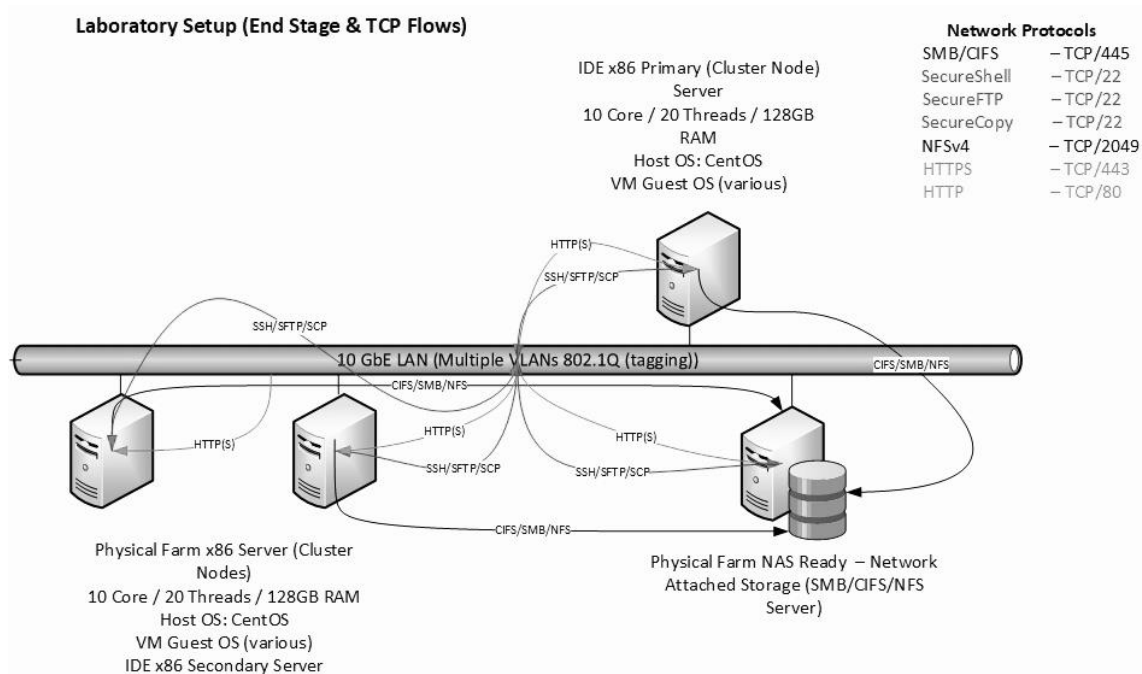


Figure 4.3 IDE Network Flows

4.6.2 Creating the Inference Engine

The inference engine is covered extensively in section [4.8](#). The purpose for the inference engine is to enable the IDE system to be able to reason on the information it gathers and then be able to infer and make decisions based on that data, to manage VMs and the virtualised platform more effectively. The act of Inference is defined in the Cambridge Dictionary as “a guess that you make or an opinion that you form based on the information that you have” ([Cambridge Advanced Learners Dictionary, 2019](#)).

4.6.3 System Self-management and Learning

A critical part of the IDE is to have the concept of self-management; this specifically covers these areas, as follows:

- High availability and being able to maintain the system and its services within the cluster framework.
- Maintain data repositories, cleanse, order, and archive data as necessary and maintain filesystem structure sizes to ensure they do not fill up.
- Text analysis from log files – using the data extracted and pattern matched against trigger rules, to take necessary actions.
- Migration of VMs as necessary due to a complete failure scenario (i.e. crash and restart VM and its services).
- Resizing (dynamically) of VMs as necessary to accommodate extra memory and CPU increases.
- Learning from typical application sizing footprint, depending of the software deployed for example, Apache, Apache Tomcat, MySQL and so on.
- The ability to learn and manage aggregation of application and database footprints for VMs.

4.6.4 System Real-time and Source Data

For the system to manage virtualised systems effectively, it must be able to perform the following two functions:

- Real-time data processing.
- Decision making capability, based around forward-chaining.

Forward chaining begins with the source data set that is made available to the decision engine; for example, it uses knowledge rules to match against the relevant data from the end user, or a system log file, until a decision, or interpretation can be made. This is reached by analysing the available rules until a conditional match (or set of matched conditions) can be satisfied, for example by using an if clause statement. If the conditions are true, then a resulting action can be triggered, or invoked to perform a remedial task (or set of tasks) for the managed systems, with the goal of resolving a certain issue ([Martin et al, 2007](#)). The Inference engine will continue to iterate through this process until a goal is reached, upon where it is executes its matching rule and then continues to iterate. In terms of real-time data sources, the following generic log file data inputs are available on CentOS Linux systems (data sources):

- `/var/log/messages` (generic system)
- `/var/log/auth.log` (security logs)
- `/var/log/secure` (security logs)
- `/var/log/boot.log` (boot up issues)
- `/var/log/dmesg` (system boot up console)
- `/var/log/kern.log` (kernel messages)
- `/var/log/faillog` (failed login attempts)
- `/var/log/cron` (cronjobs output)
- `/var/log/yum.log` (new packages)

- /var/log/mail.log (smtp log)
- /var/log/httpd (apache logs)
- /var/log/mysql.log (db logs)

Interface Rules, to search against the following defined condition types; see section [4.8.1](#) for trigger events, and section [4.8.5](#) for more detail on the justifications:

1. Physical host down - evacuate all VMs (path to truth, rule match). This rule applies when a physical host fails, typically as a result of a hardware failure. Such events are relatively common depending on the amount of physical hardware deployed, and the mean-time between failure rate. Should this event occur, all VMs should be evacuated and restarted on alternative healthy nodes in order to maintain high availability ([Tsai, 2009](#)).

2. Physical host memory capacity hit - migrate VMs back within memory threshold (path to truth, rule match). As with any physical system, there are always limits to memory resource. Therefore, it is necessary to be able to migrate VMs and virtual resources to other physical systems, in order to distribute the load across the platform as evenly as possible ([Sarathy et al, 2010](#)).

3. Physical host CPU capacity hit - migrate VMs back within CPU threshold (path to truth, rule match). This explanation is the same as rule 2 above, but for CPU rather than memory resource.

4. VM memory overload - threshold hit, dynamically resize VM (path to truth, rule match). The virtual machine's allocated memory is at capacity, or underutilised, and requires resizing ([Antonescu et al, 2013](#); [Dhiman, 2011](#)).

5. VM CPU overload - threshold hit, dynamically resize VM (path to truth, rule match). This explanation is the same as rule 4 above, but for CPU rather than memory resource.

6. VM Migration - system wide re-utilisation algorithm and no SLA impact - move resources (path to truth, rule match). This rule allows for the movement of VMs within the managed platform environment ([Benet et al, 2016](#); [Shirinbab and Lundberg, 2016](#)).

7. VM unresponsive, hung-state or non-accessible - evacuate to new (path to truth, rule match). Should a VM no longer be operational, a hard restart is required, or failing that an evacuation to a new physical host ([Shirinbab et al, 2016](#)).

8. VM compulsory move - Unable to dynamically resize the VM for rule 2 or 3 due to physical resource constraints, therefore, forced to move and relocate the VM to new a physical host with sufficient spare capacity. (path to truth, rule match).

The matching process is described as follows:

```

WHILE true
DO
  FOR EACH host
    evaluate all real-time data sources text
    evaluate critical alerts text
    search through forward chains
    IF pattern match true
      invoke trigger (consequent)
    ENDIF
  DONE

```

Table 4.12 IDE Rule Matching Process

4.7 System Availability and Autonomy

Traditional clusters often use a common standard deployment of two-nodes plus a 'quorum device' ([Vogels et al, 1998](#)). Such a device is normally configured to provide a 'third vote' mechanism, such as race condition to place a SCSI reservation on disk. In days gone by when hardware was relatively expensive, this was a good option; however, given the fact that x86 commodity hardware is now so cheap comparatively, having a minimum of three nodes in a cluster is a simple and optimal method to achieve high availability of systems, thus eradicating Single Points of Failure ([White et al, 2004](#)). Therefore, to maintain simplicity, it is instead proposed to use a simple formula defined below in section [4.7.1](#).

4.7.1 Establishing a Quorum

Quorum Definition: When a cluster node fails, or when a subset of nodes lose contact with another subset, the surviving remnant of nodes need to verify that they now constitute the *majority* of the cluster nodes that remain. If they cannot confirm that, they will go offline, and cease to operate as a protective measure to mitigate against events that can happen such as ‘split brain’, where a cluster partitions into two or more parts, which simultaneously believe they have a majority quorum and attempt to run services. An event such as this can lead to data corruption, which is a highly undesirable outcome. Therefore, the concept of a *majority* only works if there more than 50% of cluster node votes available (rounded up) to establish a quorum. For an IDE cluster, the *minimum starting number of cluster nodes is three*.

The mechanism is represented as below:

- Where η denotes a cluster node that is *available or unavailable*
- Where ϵ denotes a cluster node that is *unavailable*

Where T denotes the total number of cluster node votes possible (each node has one vote)

$$T = \sum \eta \quad (1)$$

Where v denotes the number of cluster node votes currently available

$$v = T - \epsilon \quad (2)$$

Where ϖ is the minimum number of votes to establish a quorum, which is always an integer; when a cluster has an even number of total cluster nodes step 3a is followed, or if an odd number of cluster nodes exist, step 3b is followed. For example:

Where there are an even number of cluster nodes

$$\text{if } (T \bmod 2) = 0 \text{ then } \varpi = \frac{T}{2} + 1 \quad (3a)$$

or where there are an odd number of cluster nodes

$$\text{else } \varpi = \left\lceil \frac{T}{2} \right\rceil \quad (3b)$$

Where ϱ denotes the ability to establish a cluster quorum

$$\text{if } v \geq \varpi \text{ then } \varrho \quad (4)$$

As an example scenario, take a **3-node healthy cluster**.

- η is the sum of current number of cluster nodes, available or not available, which is 3 (step 1)
- v is the total number of active healthy node votes, so 3 minus 0 (step 2)
- ϖ is 3 divided by 2 rounded up (an integer), so 1.5 rounded up to 2 (step 3b)
- Therefore ϱ is possible as 2 is greater or equal to 2; therefore, the cluster can establish a quorum (step 4)

As an alternative example, take a **3-node cluster with only 2 healthy nodes**, assuming 1 has failed.

- η is the sum of current number of cluster nodes, available or not available, which is 3 (step 1)
- v is the total number of active healthy node votes, so 3 minus 1, resulting in 2 (step 2)
- ϖ is 3 divided by 2 rounded up (an integer), so 1.5 rounded up to 2 (step 3b)
- Therefore ϱ is possible as 2 is greater or equal to 2; therefore, the cluster can establish a quorum (step 4)

As a further example, take a **3-node cluster with only 1 healthy node**, assuming 2 have failed.

- η is the sum of current number of cluster nodes, available or not available, which is 3 (step 1)
- v is the total number of active healthy node votes, so 3 minus 2, resulting in 1 (step 2)
- ϖ is 3 divided by 2 rounded up (an integer), so 1.5 rounded up to 2 (step 3b)
- Therefore q is not possible as 1 is not greater or equal to 2; therefore, the cluster cannot establish a quorum (step 4)

As a final example, take a **4-node cluster healthy cluster**

- η is the sum of current number of cluster nodes, available or not available, which is 4 (step 1)
- v is the total number of active healthy node votes, so 4 minus 0, resulting in 4 (step 2)
- ϖ is 4 divided by 2, plus 1 resulting 3 (step 3a – remember this cluster has an even number of cluster nodes)
- Therefore q is possible as 4 is greater or equal to 3; therefore, the cluster can establish a quorum (step 4)

Examples of valid cluster node configurations are any number of nodes three or more (e.g. 3, 4, 5 and so on). This simplified method eradicates the need for adding a special vote, such as a SCSI disk reservation, or a witness node.

4.7.2 Command Zone Concept

The command zone operates and hosts all the intelligence for the clustered systems. It is essential that this always remains active, otherwise the cluster automatic intelligent management will fail to continue to operate. Within the cluster, all machines are connected to the core highly available network – this can be any pre-defined network address range with static IP addresses assigned; for example, this could be an internal private IP address range

within an organisation, such as network 192.168.1.0 with a 24-bit netmask.

Every system connected in the cluster needs to be constantly aware of the other machine statuses. There are only three possible results:

- Online with resource capability (i.e. compute resource available).
- Online but with no resource capability (i.e. compute resource exhausted).
- System down and unavailable.

As discussed previously, the minimum number of machines in the cluster must be three (see section [4.7.1](#)); the cluster may scale indefinitely, such is the design. Each machine within the cluster probes the other systems systematically and reports the status output to a file stored on a highly available NFS (Network Filesystem) share. Each system then routinely interrogates the share to determine the status of the cluster. Where two independent machines both identify another system is down and unavailable, and or the machine itself reports it is isolated, the cluster will immediately seek to evacuate and establish the unavailable systems workload on other available systems. This is demonstrated by the procedure below:

```

INPUTS: cluster_node_list, cluster_resource_list, cluster_active_node_list, cluster_inactive_node_list
OUTPUTS: cluster_resource_evacuation_notification, cluster_resource_running_notification

WHILE true
DO
  FOR EACH cluster_resource
    IF cluster_resource failed AND on inactive_node THEN
      IF cluster_cpu+mem_space >= sum(cluster_resources_cpu+mem) THEN
        Evacuate cluster_resource and start on least loaded remaining node
        Send evacuation notice
      ELSE
        Unable to evacuate cluster_resource due to space constraint
      ENDIF
    ENDIF
  ENDIF
DONE

```

Table 4.13 IDE Cluster Resource Evacuation

4.7.3 Keep Alive Critical Processes

There are four principles that are required to operate a cluster.

- The Command System (IDE) must always remain alive in the cluster, until such time as a Quorum can no longer be achieved.
- A Quorum must be maintained to operate the cluster.
- Each node must be able to monitor the health of every other node in the cluster.
- The network the cluster operates on must be Highly Available (HA) with no SPoF.

The following processes are therefore deemed critical and must operate as follows:

- One or more slave node(s): [ide_slave_node]
- One master node: [ide_master_node]
- One shadow master node: [ide_shadow_node]

4.8 IDE Rule-base and Inference Engine

4.8.1 IDE Trigger Events

Below is a diagram showing the initial IDE Trigger event mechanism:

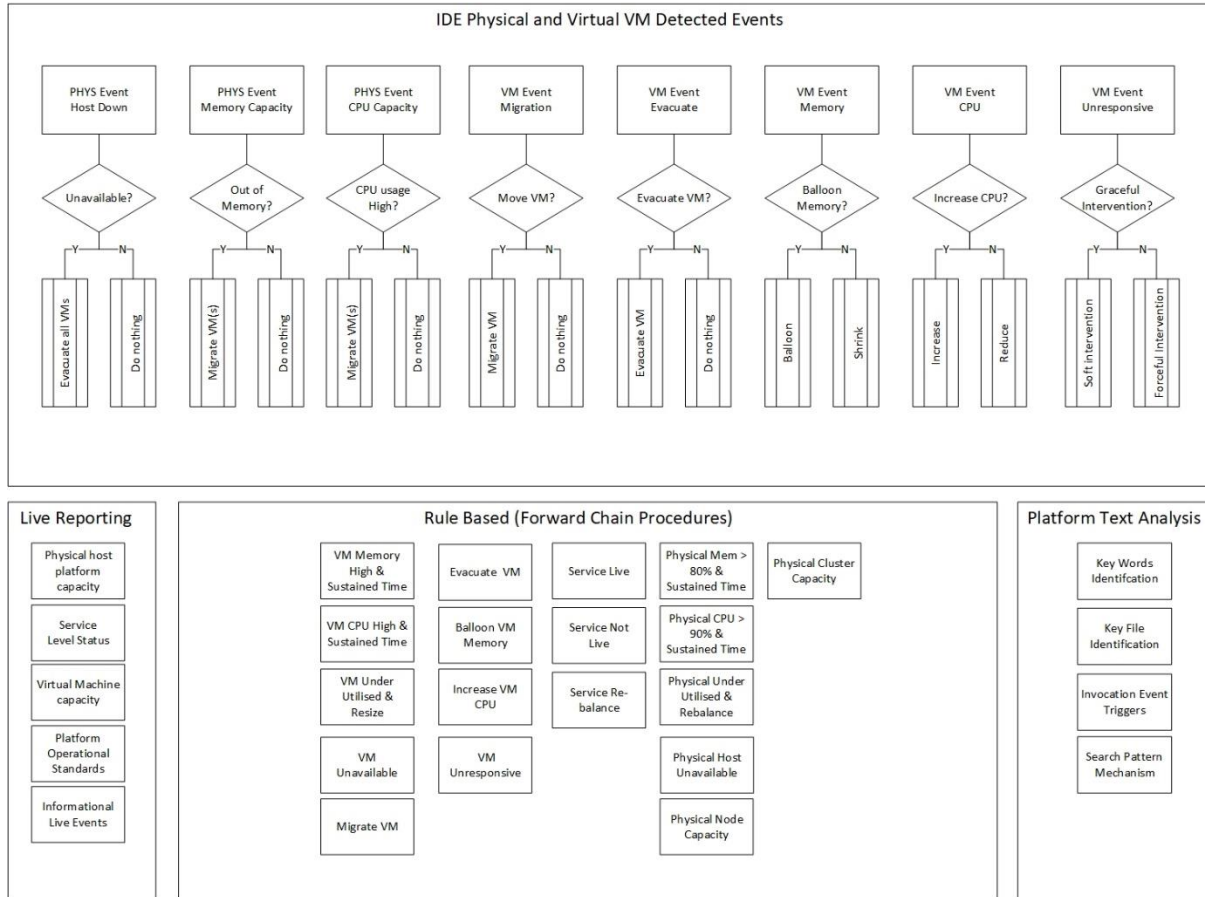


Figure 4.4 IDE Trigger Events

4.8.2 Physical System Events

System event number	1
System event description	Physical host down.
Rule match	Physical machine down AND all local (guest) VMs down.
Conclusion	Evacuate all VMs to most appropriate remaining good systems using most effective load-balancing strategy.
Event response	Recovery of all VMs from failed physical host to other remaining systems.

Table 4.14 Event Knowledge Rule: Physical host down

System event number	2
System event description	Physical host memory capacity Exceeded AND time period > 15 minutes.
Rule match	Physical machine physical memory > 80% AND no SLA breach will be invoked because of migration.
Conclusion	Migrate VMs to most appropriate remaining good systems using most effective load-balancing strategy.
Event response	Migrate VMs UNTIL physical memory within threshold.

Table 4.15 Event Knowledge Rule: Physical Host Memory Capacity

System event number	3
System event description	Physical Host CPU capacity exceeded AND time period > 15 minutes.
Rule match	Physical machine physical CPU > 80% AND no SLA breach will be invoked because of migration.
Conclusion	Migrate VMs to most appropriate remaining good systems using most effective load-balancing strategy.
Event response	Migrate VM(s) UNTIL physical CPU within threshold.

Table 4.16 Event Knowledge Rule: Physical Host Memory CPU

4.8.3 VM System Events

Sections [4.2.3](#) and [4.6.3](#) describe how the IDE intends to learn by continually self-evaluating its own event possibility matrix, by creating new event types as necessary, with appropriate rules and event responses. For example, event type 4 ‘VM system memory overload’ could be an event that does not currently exist; therefore, it is created as an event. The rule required to match that event could be system memory at utilisation of over 75% and for a sustained time period of more than 5 minutes (see table [4.17](#)). The conclusion is to

provide more memory and the event response is to dynamically re-size the VM by adding an extra 25% memory. Table [4.17](#) describes the values:

System event number	4
System event description	VM system memory overload.
Rule match	System memory utilisation > 75% AND time period > 5 minutes.
Conclusion	Provide additional memory to the VM.
Event response	Invoke dynamic VM memory re-size +25% of original total.

Table 4.17 Event Knowledge Rule: Memory overload

Another example could be event type 5 ‘system CPU overload’ that follows a similar approach to table [4.17](#) ‘VM system memory overload’. Table [4.18](#) describes the values:

System event number	5
System event description	VM System CPU overload.
Rule match	System CPU utilisation > 75% AND time period > 5 minutes.
Conclusion	Provide additional CPU to the VM.
Event response	Invoke dynamic VM CPU re-size +25% of original total, or by a minimum of one CPU core, whichever is larger.

Table 4.18 Event Knowledge Rule: CPU overload

Based on the example certain events being successfully captured (or detected) as listed in table [4.17](#) and [4.18](#) above, further development possibilities are considered particular around forward-chaining (or forward-reasoning). To provide a little more context, figure [4.5](#) below provides an example of forward chaining in relation to how a CPU management alert is handled:

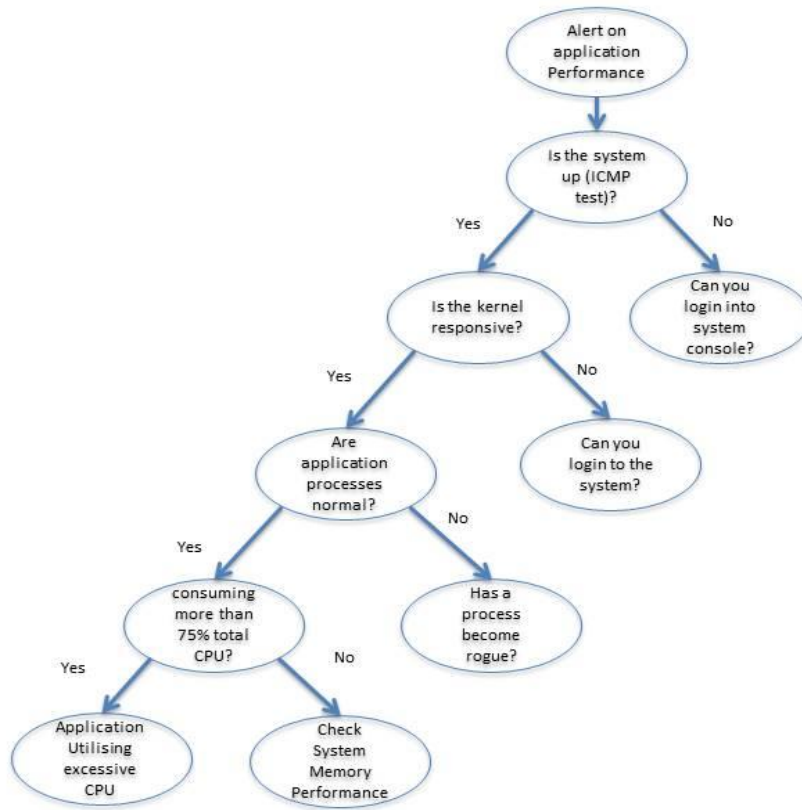


Figure 4.5 IDE Example of Forward-chaining

Thus, we can consider that this is exactly the type of mechanism (forward-chaining) that provides unique opportunity to use an inference engine to explore innovative ways of effectively managing virtualised cloud-based systems.

System event number	6
System event description	VM Migration.
Rule match	No SLA breach AND system wide utilisation re-balance of Resource (Performance Optimisation).
Conclusion	Migrate the VM to a new Physical host.
Event response	Invoke VM migration routine based on continuous VM load-balancing strategy.

Table 4.19 Event Knowledge Rule: VM Migration

System event number	7
System event description	VM Unresponsive.
Rule match	VM does not respond within 10 seconds AND VM inaccessible (1 attempt to access).
Conclusion	Evacuate the VM to a new Physical host.
Event response	Attempt one local restart of VM and then Invoke VM migration routine to new appropriate physical host.

Table 4.20 Event Knowledge Rule: VM Unresponsive

System event number	8
System event description	VM Evacuate.
Rule match	Unable to increase CPU (Local physical limit) OR Memory (Local physical limit) dynamically due to performance alert AND SLA not breached by VM migration.
Conclusion	Evacuate the VM to a new Physical host.
Event response	Migrate VM to new most appropriate physical host.

Table 4.21 Event Knowledge Rule: VM Evacuate

4.8.4 Text Analysis

In addition to section [4.6.4](#), which discusses the data sources that are used by the IDE, it is necessary to perform continual analysis and pattern matching to extract useful information from those files listed, to be able to invoke rule matches and complete forward-chain type reasoning on event driven data sets acquired by the IDE and the aggregation of useful data, to link events together ([Anicic et al, 2009](#); [Mei, L. and Cheng, 2010](#)). The following table provides examples of pattern keywords used by the IDE to help monitor trigger events and through forward-chain reasoning potentially invoke one of its matching rules in its knowledge base.

Pattern Keyword	Data Type	Information Priority Relevance
%error%	varchar	High
%warning%	varchar	Medium
%critical%	varchar	High
%full%	varchar	Medium
%lock%	varchar	Low
%failed%	varchar	High
%evacuate%	varchar	Medium
%invalid%	varchar	Low
%fatal%	varchar	High
%not found%	varchar	Medium
%missing%	varchar	Low
%invalid%	varchar	Low
%terminated%	varchar	Medium
%abort%	varchar	High
%execute%	varchar	Medium
%kernel%	varchar	High
%memory%	varchar	High
%cpu%	varchar	High

Table 4.22 Example of Keyword Pattern Analysis

Typically, the information and data extracted using this process is extracted in a priority order and then aggregated together to begin the rule matching and forward-chaining process, or perhaps even the ability to predict where resources may be required ([Flinta et al, 2017](#)).

Upon successful matching of keyword patterns, analysis work is undertaken to ascertain if the pattern and associated string is of relevance to the system. For example, the following table provides sample pattern matches and their entire string. Being able to then determine the status and value of the information is critical for the IDE to be able to decide whether to take any further actions on an event driven forward chain event.

Pattern Keywords	Associated Matched String (non-case sensitive)
%warning%	/dev/mapper/vg_app12-lv_root: ***** WARNING: Filesystem still has errors *****
%critical%	passwd: Critical error - immediate abort
%full%	ERROR cannot create datafile /vol/data/standalone/journals/logfile-321497.db: filesystem full
%error%	ERROR cannot create datafile /vol/data/standalone/journals/logfile-890354.db: filesystem full
%kernel% AND %memory%	kernel: Out of memory: Kill process 8796 (mysqld) score 719 or sacrifice child

Table 4.23 Example of Pattern Keyword Matching

In the table above we can see the last row as an example shows two keyword matches for a single detected string from a data source discussed in section [4.6.4](#); typically this type of message would be found in /var/log/messages data source. Intelligent information retrieval is a key aspect for AI systems, and it is an excellent mechanism for the IDE to adopt, in order to support the forward-chain reasoning methodology, discussed previously in section [4.8.3](#) and figure [4.5](#) (Mei and Cheng, 2010). By utilising such a retrieval method, the system can correlate particular events happening in real-time on the system and wider platform, and subsequently check to match those detected events to an appropriate knowledge rule and its consequential action (Melekhova, 2013; Matthias, 2008).

4.8.5 Knowledge Rule Justifications

The following examines the knowledge rules defined in sections [4.8.1](#), [4.8.2](#) and [4.8.3](#) and defines and justifies the reasons why these rules are beneficial, worthy of inclusion and useful for the IDE platform function.

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
1	Physical host down.	<p>Justification: A physical computer system fails as a result of either human error, data corruption, or hardware failure, such as a disk, memory, or CPU fault.</p> <p>Human error can occur at any time, although the IDE aims through its automation and AI/knowledge rules to remove the need for human intervention where possible because of this risk, although it is vital to understand that this access is not restricted and is allowed. Human interventions can be easily be mistaken and often be inconsistent, as administrators often perform their duties based upon their personal preference for completing a certain task using certain method.</p> <p>Data corruption can occur, when a particular data set has its integrity compromised, either through</p>	<p>Not fully tested, see limitations in section 1.4 and future work in sections 8.3.3 and 8.3.5 for more detail.</p> <p>Some work simulating basic host failure was completed in chapter 6 around this, but more can be done as described in future work.</p>

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		<p>inconsistent writes, multiple data access requests, where one is perhaps unauthorised, unexpected or unknown. The result is a data set that is no longer trustworthy, inconsistent, incomplete, inaccurate, and even possibly unreadable.</p> <p>Hardware failures are inevitable events, that will occur to any computer system, or set of systems. Typically, all systems have a mean-time between failure (MTBF) rate, which means that a computer system may fail at any given moment, due to power loss, or a CPU, memory, system board, I/O adapter, or disk fault.</p>	
2	Physical host memory capacity exceeded.	Any guest systems (VMs) as defined in figure 1.1 show that it is feasible for a physical host to run short of memory, through explained reasons, such as a too highly consolidated VM (guest) to physical (host) ratio, or unexplained means, such as a hypervisor failing to manage a memory leak or another unexpected platform event. If this becomes apparent over a sustained period of time, actions are needed to	Not fully tested, see limitations in section 1.4 and future work in section 8.3 for more detail.

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		alleviate this situation. Hence, this knowledge rule once its conditions are met, will seek to alleviate the situation automatically (Sanzo et al, 2012 ; Chen et al, 2013).	
3	Physical host CPU capacity Exceeded.	Any guest systems (VMs) as defined in figure 1.1 show that it is feasible for a physical host to run short of CPU, through explained reasons, such as a too highly consolidated VM (guest) to physical (host) ratio, or unexplained means, such as a hypervisor failing to manage a shared CPU cores or another unexpected platform event. If this becomes apparent over a sustained period of time, actions are needed to alleviate this situation. Hence this knowledge rule once its conditions are met, will seek to resolve the error condition automatically (Makridis et al, 2017 ; Ismail and Riasetiawan, 2016).	Not fully tested, see limitations in section 1.4 and future work in section 8.3 for more detail.
4	VM System memory overload.	This rule deals directory with a guest VM over utilising its allocated memory for a sustained period of time. This rule is essential to allow for a global resource scheduling mechanism, whereby the IDE can monitor all of its	Partially tested in chapter 7 .

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		VMs across the platform and manage the hypervisor layer effectively with respect to memory management (dynamic ballooning add/reduce and resizing). Chapter 7 deals with this in greater detail, as it highlights the benefits of implementing this within the platform (Zhang et al, 2017 ; Zhang et al, 2016).	
5	VM System CPU overload.	This rule deals directory with a guest VM over utilising its allocated CPU cores for a sustained period of time. This rule is essential to allow for a global resource scheduling mechanism, whereby the IDE can monitor all of its VMs across the platform and manage the hypervisor layer effectively with respect to CPU management (dynamic Hotplug add/remove and resizing). Chapter 7 deals with this in greater detail, as it highlights the benefits of implementing this within the platform (Zhang et al, 2017 ; Zhang et al, 2016).	Partially tested in Chapter 7 .
6	VM Migration.	Based on workload balancing strategy, and sometimes the result the invocation of high resource contention within the IDE platform, it may be	Partially tested in chapter 6 .

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		<p>necessary to migrate VMs between physical hosts, to better align VM resource management, either to free up memory, CPU or other physical resources such as network or I/O.</p> <p>Other reasons for migration, could include running VMs across different physical hosts (i.e. not collating them) and introducing negative affinities in order to ensure for example, that two VM web-servers run on different hosts.</p> <p>In the event of a hardware issue type event where one fails for any reason, the other VM web server will remain online. Chapter 6 expands on how the IDE makes use of work load balancing/availability strategies especially for unplanned failure type events; note, that the limitations section 1.4 discusses live migration for planned migration events and the future work section 8.3.3 goes into additional detail on what work could be considered (Feng et al, 2011; Shirinbab and Lundberg, 2016).</p>	
7	VM Unresponsive.	The IDE continually self-monitors the platform it manages and checks all	Partially tested in chapter 6 .

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		<p>virtual resources for their availability. Should a VM become unresponsive to health check probes and the IDE is unable to access the guest VM, steps will be taken to restart the VM locally to make it once again available, or if that fails, an evacuation, failover and restart will be completed. Chapter 6 provides more detail with an example of a simulated VM failure, and how the system deals with and recovers from this situation (Benet et al, 2016).</p>	
8	VM Evacuate.	<p>Occasionally on the IDE platform, it may happen that an attempt to dynamically add CPU or memory resources, or even a hard resize/restart for a VM fails, because there is simply not enough resource remaining on the physical host where the guest VM resides. In this case, there is no option but to consider an evacuation and migration of the VM in question, or perhaps one of less importance in terms of avoiding an SLA breach, to another physical host. Should an option be feasible the IDE system will attempt to evacuate and move the selected guest VM to an appropriate</p>	<p>Not fully tested, see limitations in section 1.4 and future work in section 8.3 for more detail.</p>

Knowledge Rule ID	Knowledge Rule Description	Knowledge Rule Justification	Testing Through Direct Experimentation
		physical host with suitable remaining resources (Feng et al, 2016).	

Table 4.24 Knowledge Rules Justifications

4.9 Summary

This chapter has described how the IDE is the key framework for extending control and management over virtualised computer systems. The characteristics of the system allow it to organise its data structures to store key information in relation to the managed systems. Additionally, the embedded knowledge rules allow the IDE to take actions to improve VM provisioning processes, failover or migrate VMs as required to restore services, or rebalance resources across the platform, through interpretation of real-time data to allow the invocation and execution of knowledge rules and its consequent. The appropriate algorithm is then used to determine how to best recover, remediate or resolve the consequent; for example, using the forward-chaining algorithm to knowledge rule match as required. The next sections cover how the IDE interacts with its controlled components over the network, and the data sources it uses as its real-time inputs, which are analysed for key text patterns as well as the retention of that useful information, for example relating to VM or application sizing. Next the way the IDE maintains its high availability is covered in detail, and data of the knowledge rules are discussed, highlighting how they cover critical system events. Finally, some justifications are provided as to why each of the knowledge rules were selected and chosen for inclusion into the IDE system. The next chapter addresses the experiment processes undertaken, the results gathered, and includes a summary view on the outputs.

Chapter 5: Simplified Deployment of Virtual Machines

5.1 Introduction

Developing the ideas from the motivation and aims listed in section [1.2](#) of this work, experiments were conducted to help demonstrate that VMs could be provisioned in a simplified way. There are two main areas that are covered by the experiment process; they are based on the methodology approach defined in section [3.5.1](#) and [3.5.2](#) respectively:

- Real-time experiments to compare and measure how long it took the three end-user group types defined in section [5.2.1](#) to provision a single VM across the platforms described in section [5.2.2](#); this was a one-off experiment which took a considerable amount of elapsed time, using the three platforms see Appendix [A](#), with data sourced from anonymous distributed users, who self-categorised themselves based on the criteria described in section [5.2.1.2](#). Once the groups reached 31 in total, they were then closed to new users; please see section [1.4](#) which discusses and identifies the limitations with respect to the number of participants in the study.
- The participants in the study had the following characteristics:
 - Population: A large selection of cloud administrators, including even those not working professionally in IT.
 - Target population technical ability: Mixed ability of novice, experienced, expert users of cloud systems.
 - Study population: Voluntary, 3 anonymous groups of 31 users, for a grand total of 93 users, who have self-categorised their ability.
 - Sample error for estimate: Low, as clearly defined steps 1 to 10, with the goal to provision a VM, were provided to the participants.
 - Studies related to problem discovery show that a user participant group size of around 30 users will capture around 97-99% of all issues. Increasing

this to around 90 users, should capture nearly all problems, or around 98-100% of issues; therefore, the numbers in the study provide a strong basis to show that the participant numbers are viable for the experiments conducted ([Macefield, 2009](#); [Faulkner, 2003](#)).

- Additionally, following experiment 1, and the testing around VM provisioning timing which yielded quantitative data, based on similar studies it was determined that the experiment process could be widened to take advantage of the qualitative data also obtained. By using the available results, it would be beneficial to be able to measure the cognitive load complexity on the end-user delivery of VMs ([Oakes et al, 2019](#); [Rothenburg et al, 1987](#)). Therefore, significant effort has been made as part of this research to investigate how to measure and reduce the complexity of building virtual machines ([Plass et al, 2010](#)). Section [5.2.3](#) describes the experiment process and presents the findings in greater detail.

5.2 Simplified VM Provisioning

5.2.1 Experiment Process

The process to evaluate the IDE simplified VM deployment mechanism is as follows, using several experiment processes/parameters that are defined in the sections below.

5.2.1.1 Task Complexity Definition

Defining task complexity and associated subjective techniques more often than not involve a set of questions containing one or many semantic differential scales, on which the participant can indicate their personal experience, in respect to cognitive load during the experiment process ([Paas et al, 2003](#)). Similar scales have been developed by researchers previously, who based it on a measure of the perceived task difficulty ([Borg et al, 1971](#); [Gopher and Braune, 1984](#)). In Paas's study, participants had to report their invested mental effort on a symmetrical scale ranging from 1 (very, very low mental effort) to 9 (very, very high mental effort) after each problem during training and testing. Using a similar method, the following

table describes the guide for process complexity for tasks defined in section [5.2.1.4](#). Note for the 10-step VM provisioning process expanded in section [5.2.1.6](#), each step can be considered as ‘task’ made up of ‘sub-tasks’.

Process Complexity Key	Definition
Simple	Intuitive, no training required. An example of a simple tasks would be answering a question such as: “What is your age?”, accessing a URL via a browser to load a website, or sending a 10-20 worded SMS (Short Message Service) message.
Moderate	Basic training required, some experience and know-how necessary to execute the task. An example of a moderate complex tasks would be following a recipe with 3-4 ingredients to prepare and make a meal, writing a BASIC computer program to calculate the Body Mass Index (BMI) value of a human being, or being able to describe and use Pythagoras theorem to calculate the length of the hypotenuse.
Difficult	Advanced training required, experience essential on how to implement and complete the task. Examples of a difficult task would be completing a residential home extension architectural drawing to conform to local government planning and building regulations, being able to write a computer program to graphically draw a chessboard or being able to explain in a classroom the full implementation of Internet Protocol version 4 (IPv4), providing examples of network classes, subnets and network routing.

Table 5.1 Task Complexity Rating

5.2.1.2 User Types

As part of the experiment process, the volunteer user participants were able to anonymously define themselves into one of three designated groups by self-determining which group they belonged to as defined in each of the definitions:

End-User Type	Definition	Quantity of Users in the group
Novice User	A user with little (less than a year) or no formal training in computer science and no work experience in computing disciplines.	31
Experienced User	A user with some training in computing disciplines, up to A-level standard, with some formal training or 1-3 years' work experience in the field.	31
Expert User	A user with training in computing disciplines, with a bachelor's degree level or above, or with more than 5 years' work experience in the field.	31

Table 5.2 End-User Types

5.2.1.3 Task Types

Task types for the experiment are listed below. Importantly, each task type is based on the requirement or non-requirement for human/end-user inputs.

Task Type	Definition
Manual	All sub-components of the task require manual user inputs.

Task Type	Definition
Semi-Automated	Some of the sub-components of the task require manual user inputs, some are automated.
Automated	No sub-components of the task require any user inputs.

Table 5.3 Process Mechanism Definition

5.2.1.4 Process Types and Complexity Value Weightings

The table below features definitions of how the process analysis was broken down into the tasks and sub-task components. For the purposes of this experiment and calculating the user feedback, we acknowledge sub-components of tasks, but never ask the users to provide their results at this level of granularity; instead, we take and record the qualitative result given at the task level:

Process/Task/Sub-components	Definition
Process	A set of tasks which make up a complete process flow; for example, the steps/tasks required for the building of a virtual machine.
Task	An action, which is part of a process, such as creating an RSA public and private key pair for a user and then deploying it.
Sub-Component	A task may be made up of sub-components, such as key generation, key distribution, and setting key permissions, and testing the private and public key handshake.

Table 5.4 Process, Task, Sub-component Definitions

5.2.1.5 User Results: Mode Average of Task Complexity Description

The following table allows for a reliable method of obtaining a mode average (most frequent) of the end-users interpretation of the task complexity description: this is either simple, moderate, or difficult. Therefore, as an example for step 8 (create and add SSH key), Oracle cloud, the mode average, or most common description recorded was 'difficult'.

User Type	User Numbers	Step Number	Qualitative Mode Average Complexity Description (Oracle)	Qualitative Mode Average Complexity Description (AWS)	Qualitative Mode Average Complexity Description (IDE)
Novice, Experienced, and Expert	3 groups of 1-31	1	Simple	Simple	Simple
Novice, Experienced, and Expert	3 groups of 1-31	2	Simple	Moderate	Simple
Novice, Experienced, and Expert	3 groups of 1-31	3	Simple	Simple	Simple
Novice, Experienced, and Expert	3 groups of 1-31	4	Simple	Moderate	Simple
Novice, Experienced, and Expert	3 groups of 1-31	5	Simple	Moderate	Simple
Novice, Experienced, and Expert	3 groups of 1-31	6	Moderate	Difficult	Simple
Novice, Experienced, and Expert	3 groups of 1-31	7	Moderate	Moderate	Simple
Novice, Experienced, and Expert	3 groups of 1-31	8	Difficult	Difficult	Simple
Novice, Experienced, and Expert	3 groups of 1-31	9	Simple	Simple	Simple

User Type	User Numbers	Step Number	Qualitative Mode Average Complexity Description (Oracle)	Qualitative Mode Average Complexity Description (AWS)	Qualitative Mode Average Complexity Description (IDE)
Novice, Experienced, and Expert	3 groups of 1-31	10	Moderate	Difficult	Simple

Table 5.5 VM Provisioning 10-Step Complexity (Mode Average)

5.2.1.6 VM Provisioning Process

During the evaluation and experiment process, there were 10 steps for VM provisioning that were followed, using empirical testing methods, whereby users were allowed to evaluate the IDE, AWS and Oracle VM provisioning platforms each in turn, while under observation by a moderator who used an unobtrusive approach as a ‘fly on the wall’ ([Seaman, 1999](#)). The provisioning steps are defined as follows, based on the methodology described in section [3.5.1](#), and using the complexity guide and mode values from the previous section [5.2.1.5](#), which are incorporated in the table below:

Step No	Description	Process Mechanism	Information Input			Complexity Amazon Web Services (Mode)	Complexity Oracle Cloud (Mode)	Complexity IDE Provisioning (Mode)
			Oracle	AWS	IDE			
Step 1	Cloud Provisioning Access.	This is the process needed to access and authenticate to use the cloud	M	M	M	Simple	Simple	Simple

Step No	Description	Process Mechanism	Information Input			Complexity Amazon Web Services (Mode)	Complexity Oracle Cloud (Mode)	Complexity IDE Provisioning (Mode)
			Oracle	AWS	IDE			
		platform, typically username/password.						
Step 2	Configure Role.	Setting up role-based access controls, such as administrator.	S	A	A	Simple	Moderate	Simple
Step 3	Select compute as the option for VM deployment	Public cloud offerings prefer to allow manual choices for other offerings such as DaaS, PaaS or SaaS. This experiment only deals with IaaS.	S	S	S	Simple	Simple	Simple
Step 4	Select the image you wish to use to install to the VM (OS type/version).	Typically, the OS version and software packages, add-on's and any other supporting application	S	S	A	Simple	Moderate	Simple

Step No	Description	Process Mechanism	Information Input			Complexity Amazon Web Services (Mode)	Complexity Oracle Cloud (Mode)	Complexity IDE Provisioning (Mode)
			Oracle	AWS	IDE			
		software.						
Step 5	Select the VM CPU, memory, and Disk Parameters.	VM Shell parameter definition phase.	S	S	S	Simple	Moderate	Simple
Step 6	Define VM Parameters.	Define, IP addresses, netmasks, OS version, packages and other such configurable parameters.	S	S	A	Moderate	Difficult	Simple
Step 7	Define VM Storage.	Select type and amount of disk storage to use.	S	A	A	Moderate	Moderate	Simple
Step 8	Add SSH key, create a key and upload the public key.	Generation of an appropriate SSH encryption key to secure communications and authentications.	S	S	A	Difficult	Difficult	Simple
Step 9	VM creation process.	VM shell creation, install and boot process.	A	A	A	Simple	Simple	Simple

Step No	Description	Process Mechanism	Information Input			Complexity Amazon Web Services (Mode)	Complexity Oracle Cloud (Mode)	Complexity IDE Provisioning (Mode)
			Oracle	AWS	IDE			
Step 10	Process for accessing the VM via the internet, or via network.	Typically involves opening up Firewall ports to access e.g. TCP 22 SSH.	M	M	A	Moderate	Difficult	Simple

Table 5.6 VM Provisioning Sequence

5.2.1.7 Hardware Provisioning Platform

VM container parameters are defined as follows, using by default the smallest VM component available for each platform for the initial testing/experimentation:

VM Vendor Type	CPU Cores	Memory (GB)	Disk (GB)	Architecture	Hypervisor
Oracle	1 (Intel Xeon processor E5 Series, 3.3 GHz).	7.5GB	34GB	x86	Oracle Cloud (OVM).
AWS	1 (Intel Xeon processor E5 Series, 3.3 GHz).	8GB	8GB	x86	AWS (Xen).
IDE	1 (Intel Celeron Processor 1017U, 1.6 GHz).	2GB	8GB	x86	Oracle VirtualBox.

Table 5.7 Allocated VM Compute Resource

5.2.1.8 VM Sizing Methods

This section examines allocation of CPU and Memory resource for VMs and the most

effective approach using a minimum recommended model, and a scale-up as required methodology to help avoid over commitment of resources and potentially under-utilised systems. The IDE approach to sizing consists of:

- Assigning minimal recommended memory and CPU for a particular OS flavour supported. Research suggests that minimalisation is an optimal approach to use to effectively utilise IaaS platforms ([Stage et al, 2009](#)). This immediately avoids waste, as Piraghaj et al show in their research ([2015, p. 33](#)); therefore, this ensures VMs are functioning at the minimal recommended level of resources, thus avoiding over-allocating CPU and memory from the outset.
- Expand and balloon memory (grow/shrink) as required by the applications; note, most support for memory ballooning requires 64-bit operating systems, that theoretically allows support up to 2^{64} bytes (~16 exabytes) for dynamic memory allocation/de-allocation and garbage collection ([Liu et al, 2015](#)).
- Memory/CPU monitoring agents running via IDE will continually monitor the entire environment, allowing for dynamic changes to occur as appropriate (i.e. shrink or expand resources). This will support the continual resource re-assessment of VMs to allow them to increase or reduce as needed.

5.2.2 Experiment 1: VM Provisioning Timing Comparison

5.2.2.1 Formalisation

Where \mathcal{T} is the total time to deploy a VM

Where n stands for the task number

Where t stands for the task identifier

Where θ is the participant time taken to complete a task (in seconds)

Where ψ is the average (mean) participant time per task (in seconds)

To find the total time to complete the 10-Step VM provisioning process we use the following:

$$\mathcal{T} = \sum_{t=1}^n \theta \quad (1)$$

To find the average time per step for each user:

$$\psi = \frac{\sum_{t=1}^n \theta}{10} \quad (2)$$

5.2.2.2 VM Provisioning Expert Users

The following table records in Appendix [A.1](#) the observed time in seconds (sec) taken for each step, for a total of 31 expert users (see definition above); a time of zero represents automatic processing, with no necessary end-user intervention.

5.2.2.3 VM Provisioning Experienced Users

The following table records in Appendix [A.5](#) show the observed time in seconds (sec) taken for each step, for a total of 31 experienced users (see definition above); a time of zero represents automatic processing, with no necessary end-user intervention.

5.2.2.4 VM Provisioning Novice Users

The following table records in Appendix [A.9](#) show the observed time in seconds (sec) taken for each step, for a total of 31 novice users (see definition above); a time of zero represents automatic processing, with no necessary end-user intervention. A time recorded as 9999 represents a user who was unable to complete a task, due to having a lack of knowledge, or understanding.

5.2.2.5 VM Provisioning Build Methods

The VM build methods are recorded in Appendix [C](#); there is a sequence recorded for each platform which are AWS, Oracle and the IDE respectively. Each user listed in section [5.2.1.2](#) accessed each platform in turn, to perform the 10-step provisioning steps listed in section [5.2.1.6](#), and recorded their timings for each step. These results can be found in Appendix [A](#) and graphed and presented in section [5.3.1](#).

5.2.2.6 Re-visiting the IDE Provisioning Experiment with Queued Pre-built System Images

It was determined that step [9](#) (of the VM provisioning process) time could almost be eliminated by adopting a new process, whereby, the IDE system pre-built VMs, which have their system configuration put into a unconfigured status, for example, using the 'sys-unconfig' ([sys-unconfig, 2019](#)) command will achieve this on Linux type systems, and allow re-configuration. This approach makes system performance and time to build somewhat irrelevant, due to the pre-build and dynamic reconfiguration process continually running in the background, anticipating a future build. This method effectively allows the IDE to store pre-built system (queued) images, which it can choose to keep in reserve to act as a future build pipeline supply. This work is a possible future development, described further in section [8.3.1](#).

5.2.3 Experiment 2: Cognitive Evaluation Performance

Every task a person undertakes requires a certain amount of cognitive power, or mental effort, in order to enact to conclusion ([Sweller, 1998](#); [Yang et al, 2017](#)). As an example, this could be from a singular simple task, such as clicking a mouse button, to a set of activities that need to be carried out in a specific sequence in order to complete an overall task successfully, such as cooking a meal using a set of ingredients and following a recipe. The question that rises from this, is how can the complexity of a task or set of tasks be measured,

and is it possible to understand the cognitive load for a user or group of users? The evidence from other studies and this work suggest it is feasible to understand this, and similar experiments in controlled conditions have provided evidence and results that demonstrate how to measure cognitive load ([Pass et al, 1994](#); [Pass et al, 2003](#); [Kotova, 2016](#)).

Within the field of work, this method of study is generally referred to as Cognitive Load Theory (CLT) in relation to task orientated problem solving. One requirement for making this feasible is because, like most processes, there is usually a start and an end, and a subsequent number of tasks in-between that are usually performed in a certain sequence. Once the process completes, this can result in a successful end and objective being met, or perhaps even in a full or partial failure. Understanding the sum of all the tasks in process is therefore essential to be able to measure the overall complexity load ([Feinberg, 2000](#)). Some processes are simple, for example pressing a power on or off button on a Television (TV) remote control. Consider that there are a few steps to this process, one locating the TV remote, two locating the correct button (power), and three physically pressing this button, to achieve the desired effect (e.g. switching the TV on/off). Conversely, other processes can be considered complex, such as the creation of a Virtual Machine (VM), due to the number of steps and the inherent know-how and technical expertise required to complete ([Selvi et al, 2014](#)).

5.2.3.1 Converting Qualitative Data into Quantitative Data: Is This Possible?

Further to existing studies, this paper examines how the cognitive load for a complex process (set of tasks) can be measured using a unique formula and method, referred to as the Complexity Load Rating (CLR). The work examines the feasibility and challenges around recording qualitative feedback and results from end-users, and proposes a method to translate this into numerical or quantitative data ([Green, 2001](#); [Srnka and Koeszegi, 2007](#); [Verdinelli, and Scagnoli, 2013](#); [Franzosi, 2004](#)). The results are then calculated for each group of users and are then evaluated to present evidence on how a complex process (such as VM provisioning) can be simplified as a result of the steps being developed with higher levels of

automation and the use of pre-coded system intelligence ([Oakes et al, 2016](#); [Lokshina and Insinga, 2004](#); [Menasce and Bennani, 2006](#)).

5.2.3.2 Cognitive Experimental Process

How to measure the cognitive load of a task is based upon the following general conditions, described by the qualitative (subjective) terms below:

- 1) The end-user interpretation of the task as either simple, moderate or difficult.
- 2) Is the task (or set of tasks) which make up the process automated, semi-automatic, or manual.

It is important when collecting qualitative data, that not too many options are presented for the end-user evaluation data outputs, based on their experience and the experiment process undertaken. For example, allowing human test subjects to input unstructured data such as free-text, or even handwritten text, makes the collation and analysis of data somewhat more difficult to interpret, simply because of the number of permutations and recognition of what the written data means ([Rusu et al, 2013](#)).

Therefore, in the context of this study, when we refer to task complexity, this is defined or described (subjectively) by the end-user as simple, moderate or difficult. Furthermore, each task undertaken has a process mechanism described as either automatic, semi-automatic, or manual. Of the three process outcomes, if a task is automated it requires no input, and is automatically set to simple; semi-automatic and manual task steps therefore require partial or full end-user inputs and can receive a simple, moderate or difficult rating.

It is natural that humans prefer providing qualitative feedback for some activity they personally take part in ([Lui et al, 2017](#)). Simple statements of whether something was good or bad is often typical of how people prefer to relate their experiences ([Austermann and Yamada, 2008](#)). By capturing all the tasks for a process, it is possible to begin to measure the

results from the experimentation method by converting qualitative data into quantitative data, thus, in effect, performing a translation of words into numbers (Franzosi, 2004). This leads us to the next phase of the experiment framework on how to use these sets of parameter variables, for Task Complexity (Table 5.1) and Process Mechanism (Table 5.3), by creating a unique method for measuring the Cognitive Load Rating (CLR) for a task or set of tasks; in this study we examine the complete process, of how an end-user would deploy a VM within a computer based cloud environment, as described in the 10-step provisioning process in section 5.2.1.6. Note, that this exercise was completed as a one-off (snapshot) exercise, and end-users were not able to repeat the experimental tests, either immediately following, or at a later point in time; see limitations for more detail in section 1.4.

The cognitive experiment process invoked is very similar to other studies in the field, as listed in section 2.7.6, although it does utilise its own scaling systems and devised formula as described below in section 5.2.3.3. Similar previous studies for cognitive load are listed in table 5.8 below:

Studies That Measured Cognitive Load and Calculated Mental Efficiency and the Measurement Technique They Used		
<i>Studies</i>	<i>Cognitive Load Measurement Technique</i>	<i>Mental Efficiency</i>
Sweller (1988)	PS, ST	
Paas (1992)	RS9	
Paas & van Merriënboer (1993)	RS9	ME
Paas & van Merriënboer (1994b)	RS9, HRV	ME
Cerpa, Chandler, & Sweller (1996)	RS9	ME
Chandler & Sweller (1996)	ST	
Marcus, Cooper, & Sweller (1996)	RS7, ST	ME
Tindall-Ford, Chandler, & Sweller (1997)	RS7	ME
Yeung, Jin, & Sweller (1997)	RS9	ME
de Croock, van Merriënboer, & Paas (1998)	RS9	
Kalyuga, Chandler, & Sweller (1998)	RS7	ME
Kalyuga, Chandler, & Sweller (1999)	RS7	ME
Tuovinen & Sweller (1999)	RS9	ME
Yeung (1999)	RS9	ME
Kalyuga, Chandler, & Sweller (2000)	RS7	ME
Kalyuga, Chandler, & Sweller (2001)	RS7	ME
Kalyuga, Chandler, Tuovinen, & Sweller (2001)	RS9	ME
Mayer & Chandler (2001)	RS7	
Pollock, Chandler, & Sweller (2002)	RS7	ME
Stark, Mandl, Gruber, & Renkl (2002)	RS9	
Tabbers, Martens, & van Merriënboer (2002)	RS9	
Tabbers, Martens, & van Merriënboer (in press)	RS9	
Van Gerven, Paas, van Merriënboer, Hendriks, & Schmidt (2002)	RS9	ME
Van Gerven, Paas, van Merriënboer, & Schmidt (2002a)	RS9	ME
Van Gerven, Paas, van Merriënboer, & Schmidt (2002b)	PR	
Van Gerven, Paas, van Merriënboer, & Schmidt (2002c)	RS9, ST	ME
van Merriënboer, Schuurman, de Croock, & Paas (2002)	RS9	ME

Note. Studies are listed in chronological order. PS = production system; ST = secondary task technique; RS = rating scale (9-point or 7-point scale); ME = mental efficiency; HRV = heart rate variability; PR = pupillary responses.

Table 5.8 Similar Cognitive Load Studies (PaaS et al, 2003)

The CLR formula used below does not include monitoring any participant physiological aspects, such as heart rate variability, or pupillary responses. The reason for this exclusion is to avoid the collection of personal data relating to the study participants, to maintain simplicity for the experiment using a Rating Scale (RS), and the known task automation classification.

5.2.3.3 Cognitive Load Rating Formula

The proposed formula for measuring the complexity of a singular task is as follows:

Where the Cognitive Load Rating (CLR) for one task stands for β

Where Task Complexity stands for Δ

Where Process Mechanism stands for \emptyset

$$\beta = \Delta \times \emptyset \quad (1)$$

This general formula can be applied to any process type, or cumulatively to a set of processes, and is not just applicable to the field of computer science and VM provisioning. In order to apply this formula to a set of processes it is necessary to make this calculation able to measure the sum complexity of a set of tasks, represented as follows:

Where λ stands for the CLR for a set (sum) of tasks

Where n stands for the number of tasks

Where t stands for the task identifier

$$\lambda = \sum_{t=1}^n (\Delta \times \emptyset) \quad (2)$$

Additionally, the formula can then be adjusted to work out the mean average of a process's task complexity, by using the following method (divides by the total number of tasks represented by n):

Where K stands for the CLR mean average for a set of tasks

$$K = \frac{\sum_{t=1}^n (\Delta \times \phi)}{n} \quad (3)$$

The CLR calculation for each user task results are derived in detail using the following formula described in the next section.

5.2.3.4 User Task Complexity Formula

Where R is the derived result

Where μ is the user input

Where s is simple

Where m is moderate

Where d is difficult

Where x is manual

Where y is semi-automatic

Where z is automatic

$$R = (\mu = (s \wedge x) \rightarrow (1 \times 10)) \vee (\mu = (m \wedge x) \rightarrow (3 \times 10)) \vee (\mu = (d \wedge x) \rightarrow (5 \times 10)) \vee (\mu = (s \wedge y) \rightarrow (1 \times 5)) \vee (\mu = (m \wedge y) \rightarrow (3 \times 5)) \vee (\mu = (d \wedge y) \rightarrow (5 \times 5)) \vee (\mu = (s \wedge z) \rightarrow (1 \times 1)) \vee (\mu = (m \wedge z) \rightarrow (3 \times 1)) \vee (\mu = (d \wedge z) \rightarrow (5 \times 1))$$

5.2.3.5 Cognitive Load Rating Chart

The following chart provides the CLR scale, on how difficult a set of tasks are for an end-user; the guide below provides the information on how to rate each overall process, in terms of the mental power requirement:

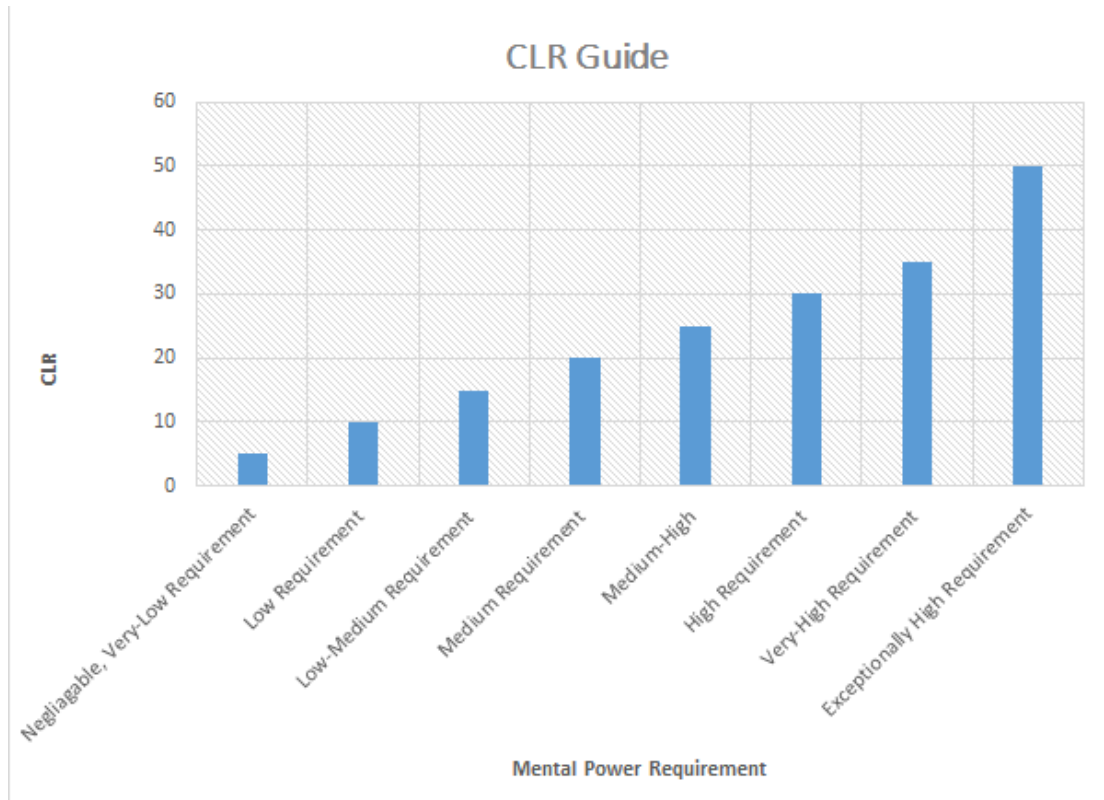


Figure 5.1 Cognitive Load Rating Chart

5.3 Results

5.3.1 VM Provisioning Timed Results

The following graphs represent the VM provisioning results from the Novice, Experienced and Expert user groups for each of the 10-steps in the provisioning process; note that the conclusions can be found in section [8.2.1](#):

5.3.1.1 Expert Users

The charts below show the results for the 'expert' user group, who performed the VM provisioning experiment on all platforms; the first graph of results presented is for the IDE (expert users):

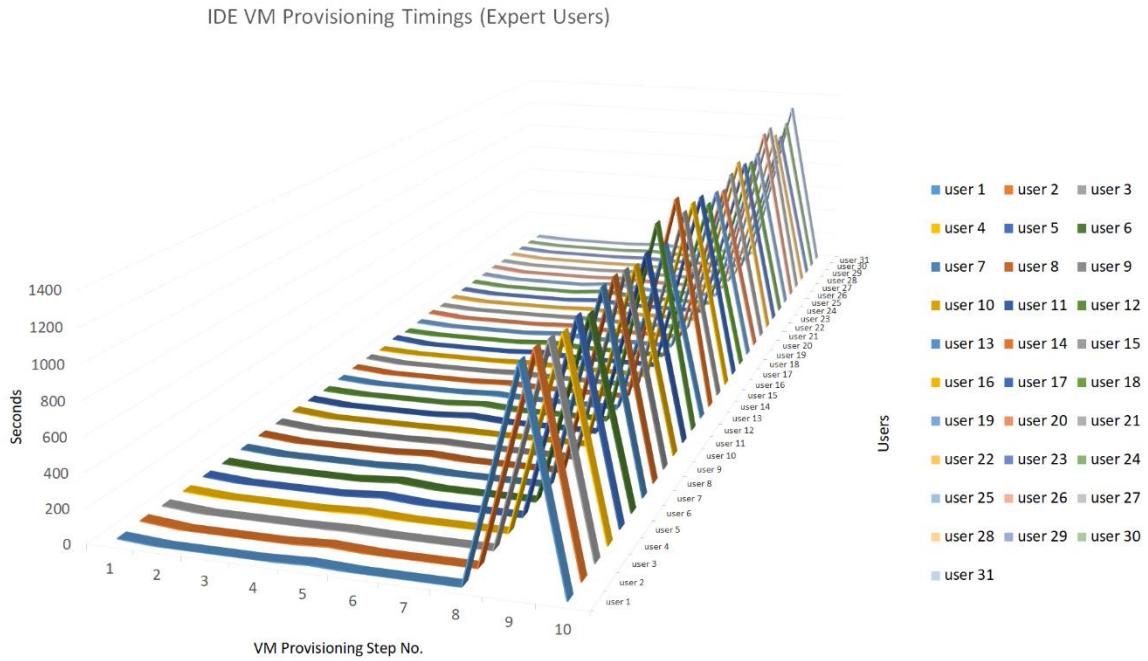


Figure 5.2 IDE Timed VM Provisioning – Expert Users

The second graph of results presented is for the Oracle cloud platform (expert users):

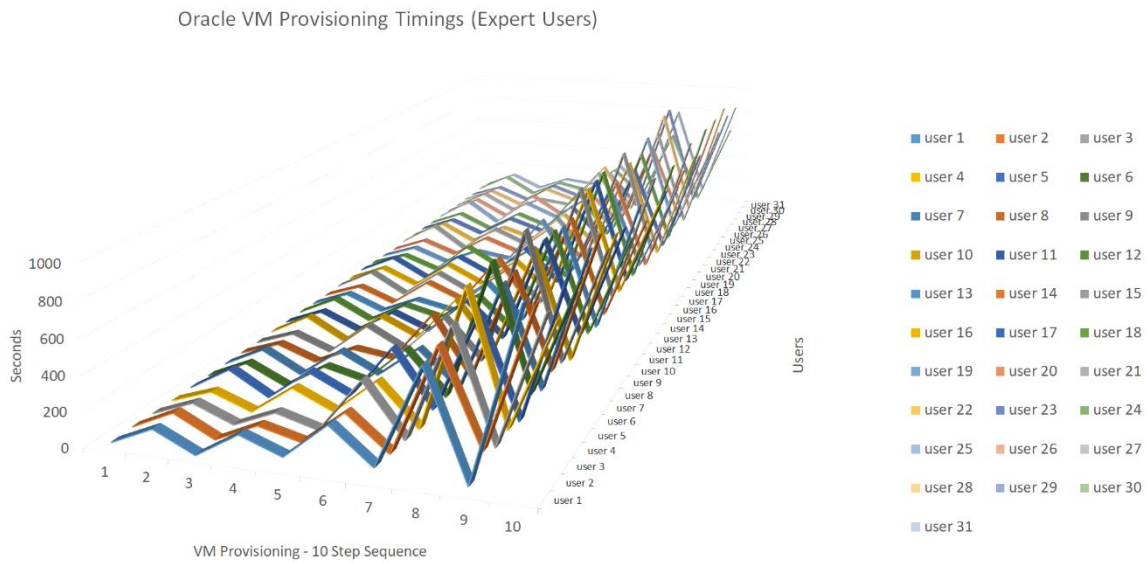


Figure 5.3 Oracle Timed VM Provisioning – Expert Users

The third graph of results presented is for the AWS cloud platform (expert users):

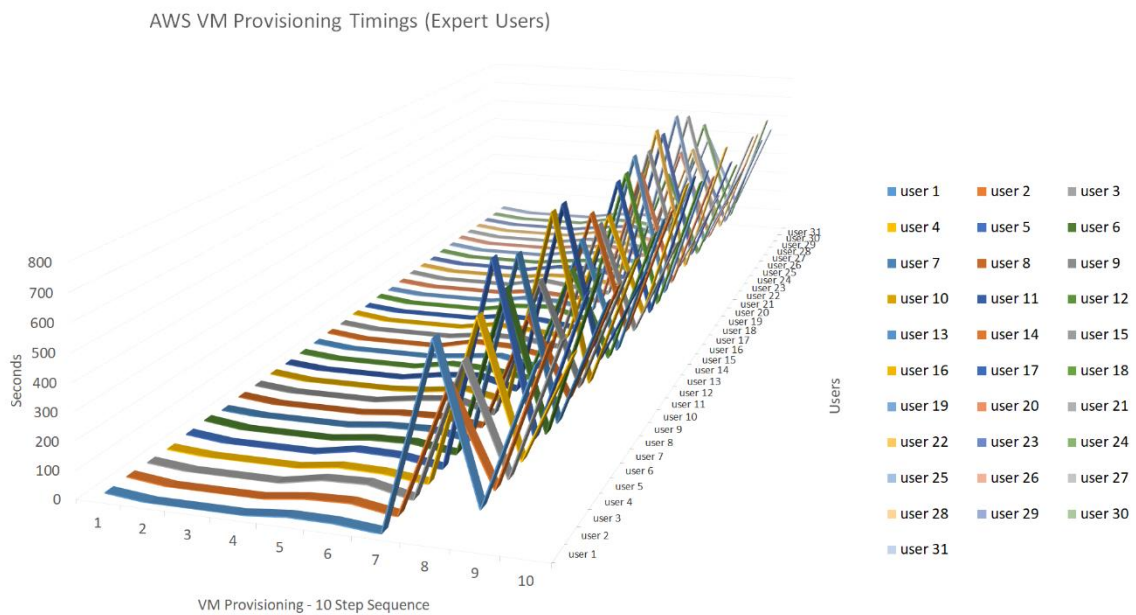


Figure 5.4 AWS Timed VM Provisioning – Expert Users

5.3.1.2 Experienced Users

The charts below show the results for the ‘experienced’ user group, who performed the VM provisioning experiment on all platforms; the first graph of results presented is for the IDE (experienced users):

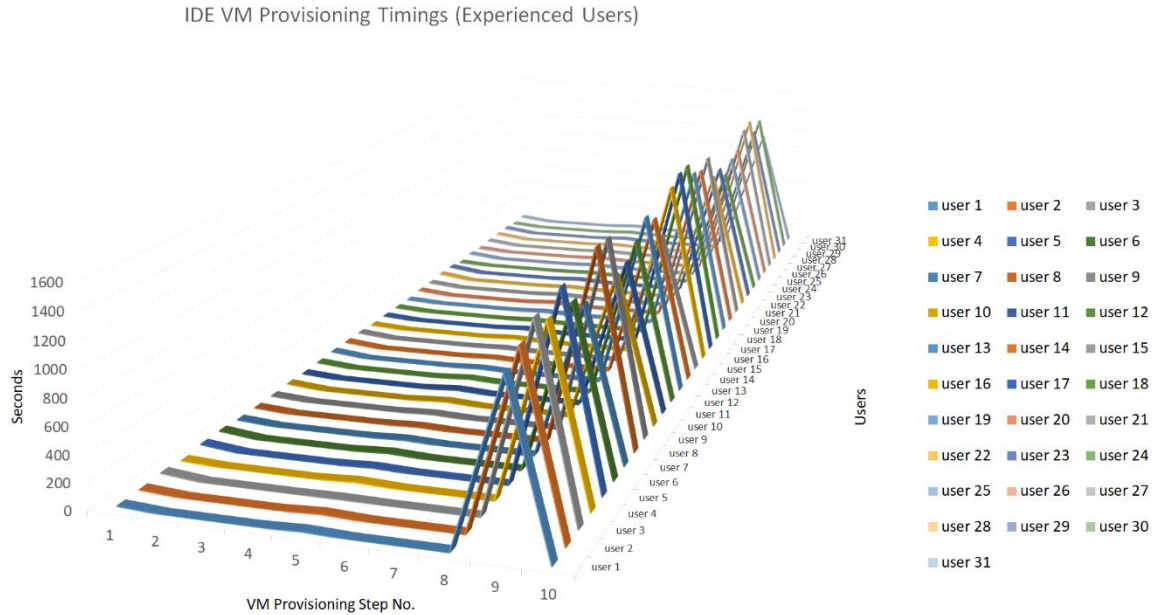


Figure 5.5 IDE Timed VM Provisioning – Experienced Users

The second graph of results presented is for the Oracle cloud platform (experienced users):

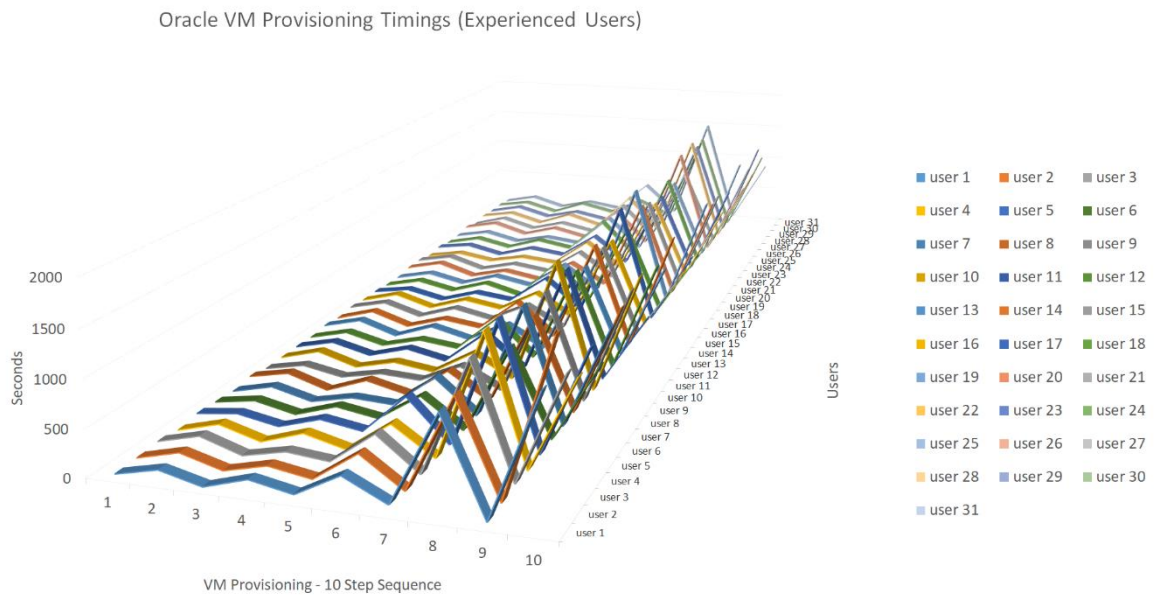


Figure 5.6 Oracle Timed VM Provisioning – Experienced Users

The third graph of results presented is for the AWS cloud platform (experienced users):

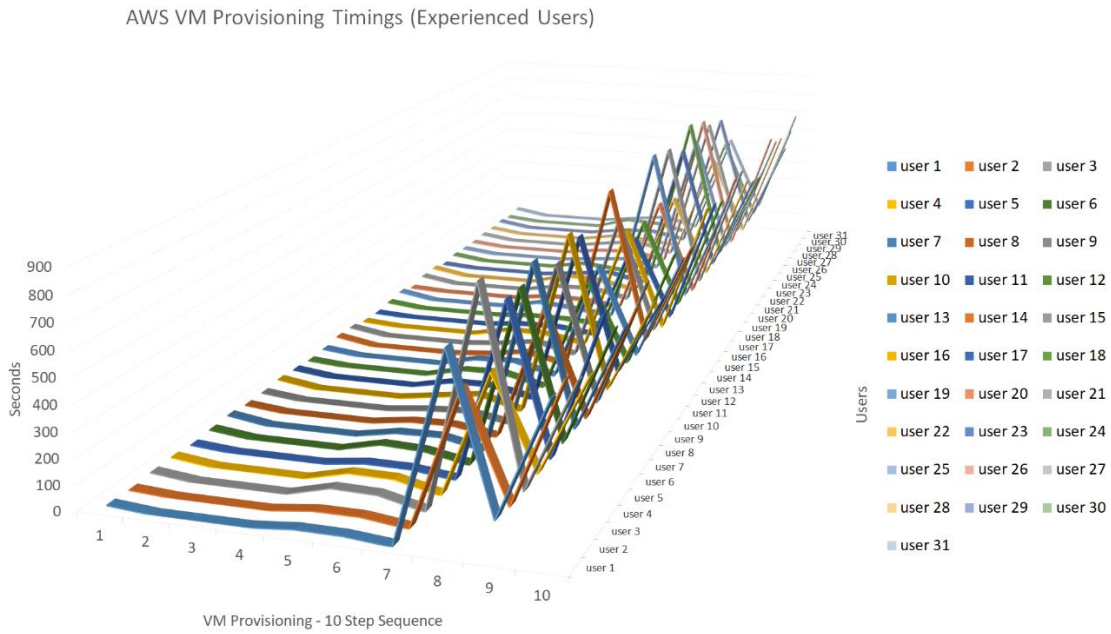


Figure 5.7 AWS Timed VM Provisioning – Experienced Users

5.3.1.2 Novice Users

The charts below show the results for the ‘novice’ user group, who performed the VM provisioning experiment on all platforms; the first graph of results presented is for the IDE (Novice users):

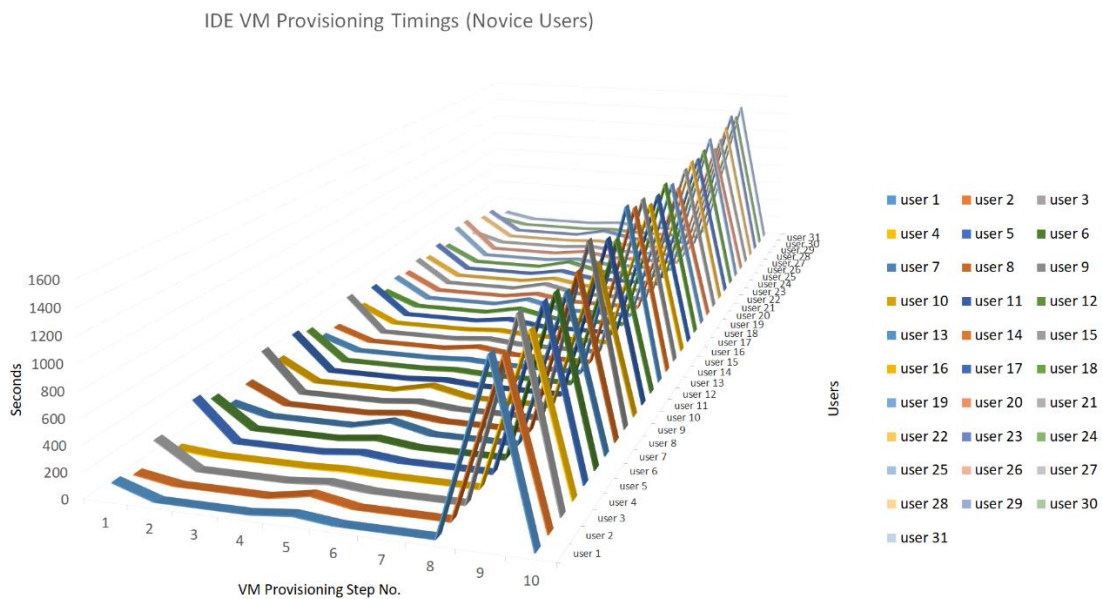


Figure 5.8 IDE Timed VM Provisioning – Novice Users

The second graph of results presented is for the Oracle cloud platform (Novice users):

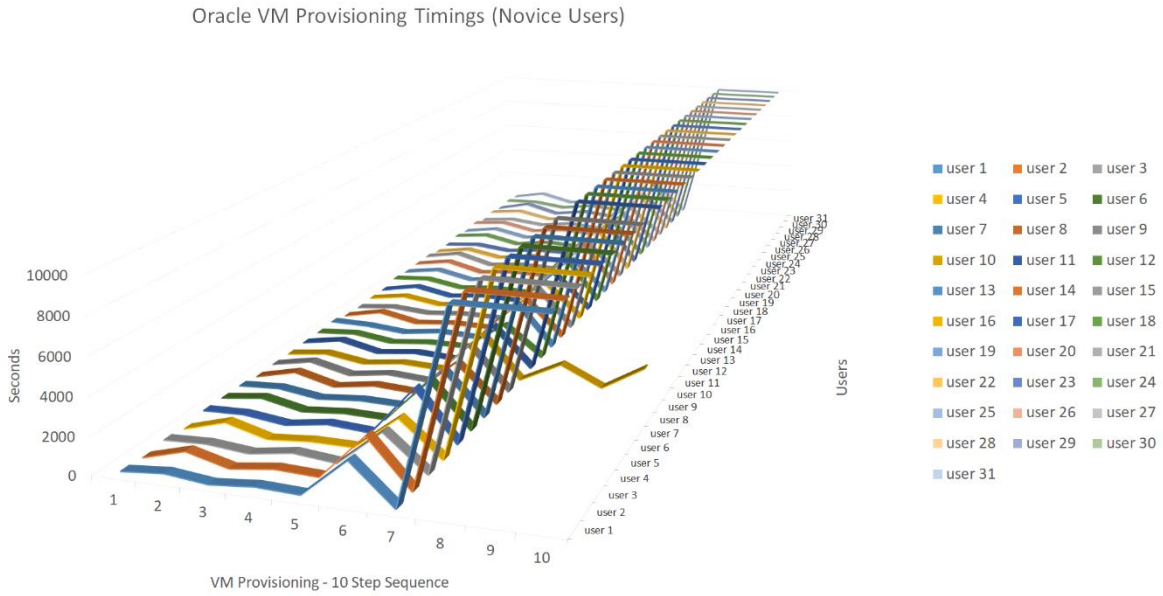


Figure 5.9 Oracle Timed VM Provisioning – Novice Users

The third graph of results presented is for the AWS cloud platform (Novice users):

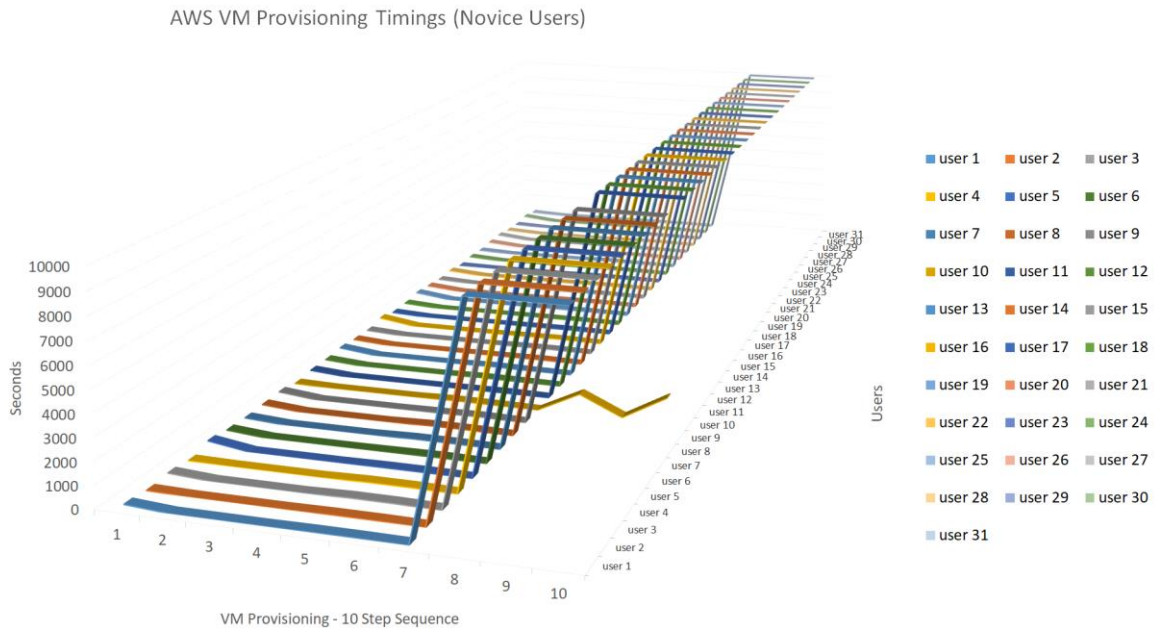


Figure 5.10 AWS Timed VM Provisioning – Novice Users

5.3.2 Aggregated VM Provisioning Timed Results

The following graphs show the aggregated total timed VM provisioning results for Expert, Experienced and Novice users; note that the conclusions can be found in section [8.2.1](#):

5.3.2.1 Expert Users

The charts below show the results for the ‘expert’ user group, who performed the VM provisioning experiment on all platforms; the first graph of results presented are for the IDE (expert):

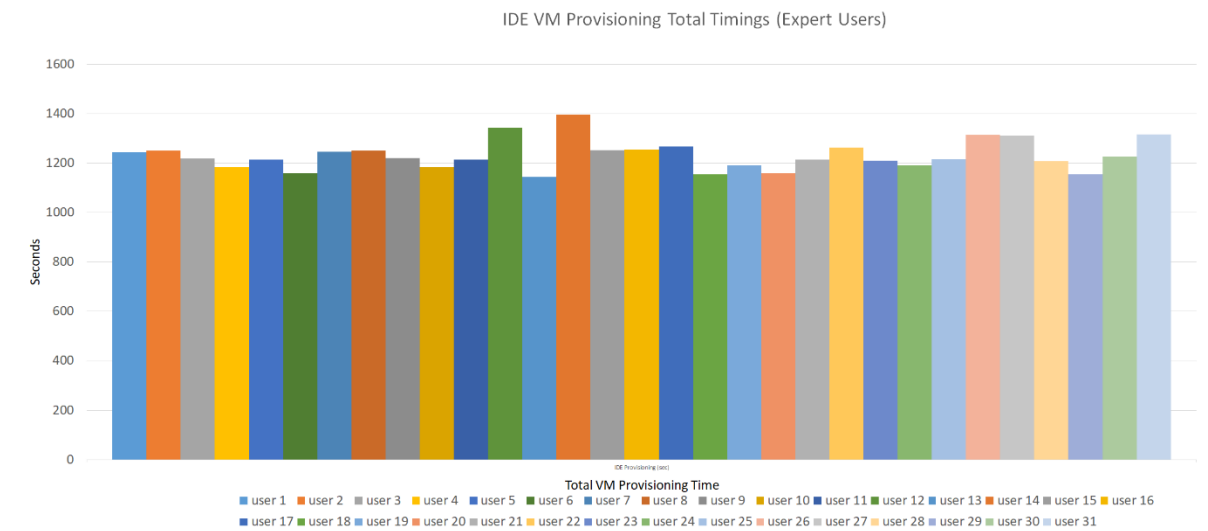


Figure 5.11 IDE Aggregated Timed VM Provisioning – Expert Users

The second graph of results presented are the for the Oracle cloud platform (expert):

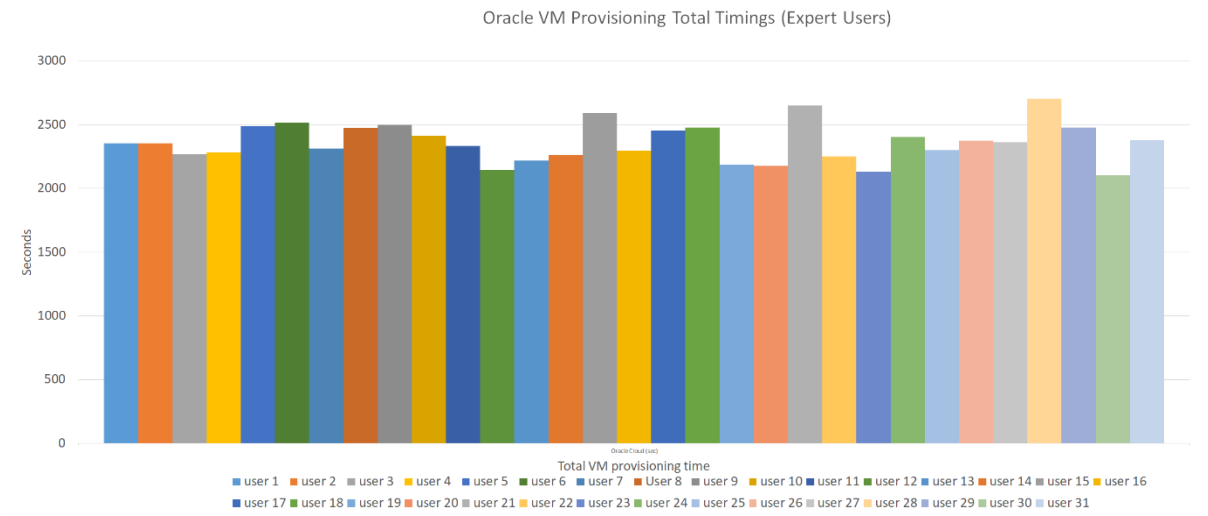


Figure 5.12 Oracle Aggregated Timed VM Provisioning – Expert Users

The third graph of results presented are the for the AWS cloud platform (expert):

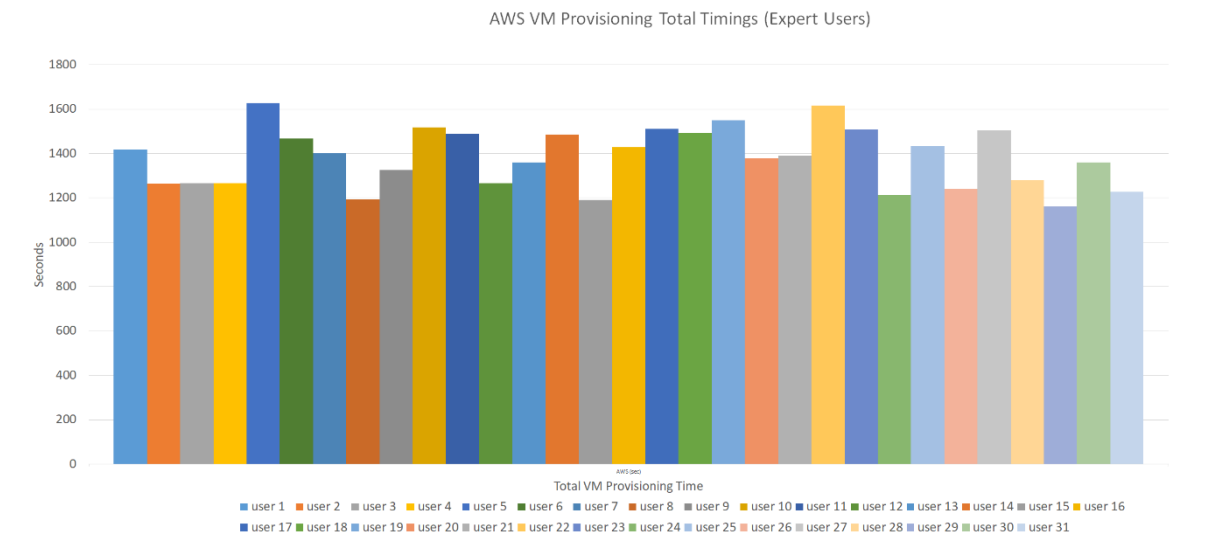


Figure 5.13 AWS Aggregated Timed VM Provisioning – Expert Users

5.3.2.2 Experienced Users

The charts below show the results for the ‘experienced’ user group, who performed

the VM provisioning experiment on all platforms; the first graph of results presented are for the IDE (experienced):

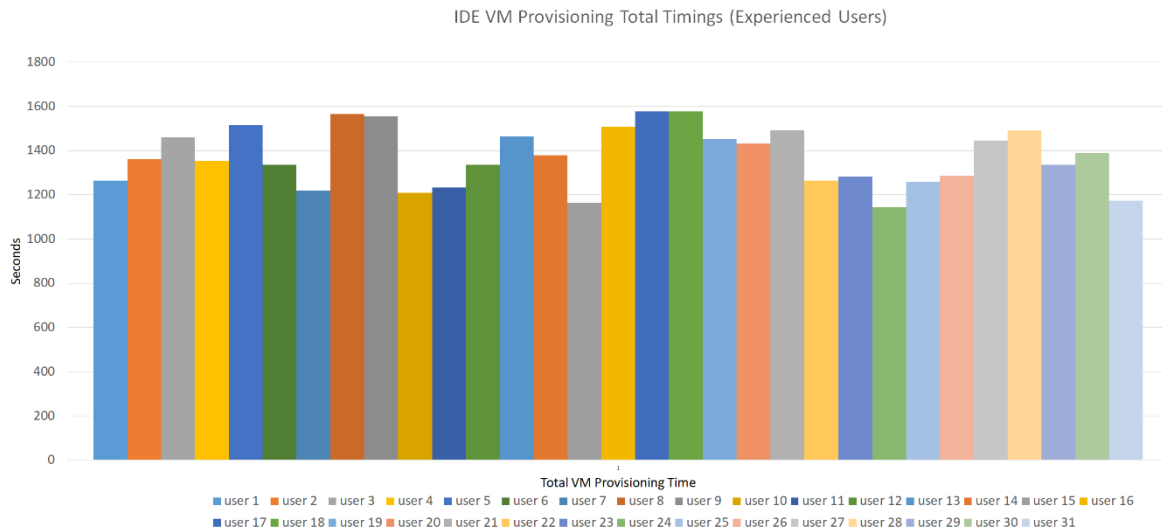


Figure 5.14 IDE Aggregated Timed VM Provisioning – Experienced Users

The second graph of results presented are the for the Oracle cloud platform (experienced):

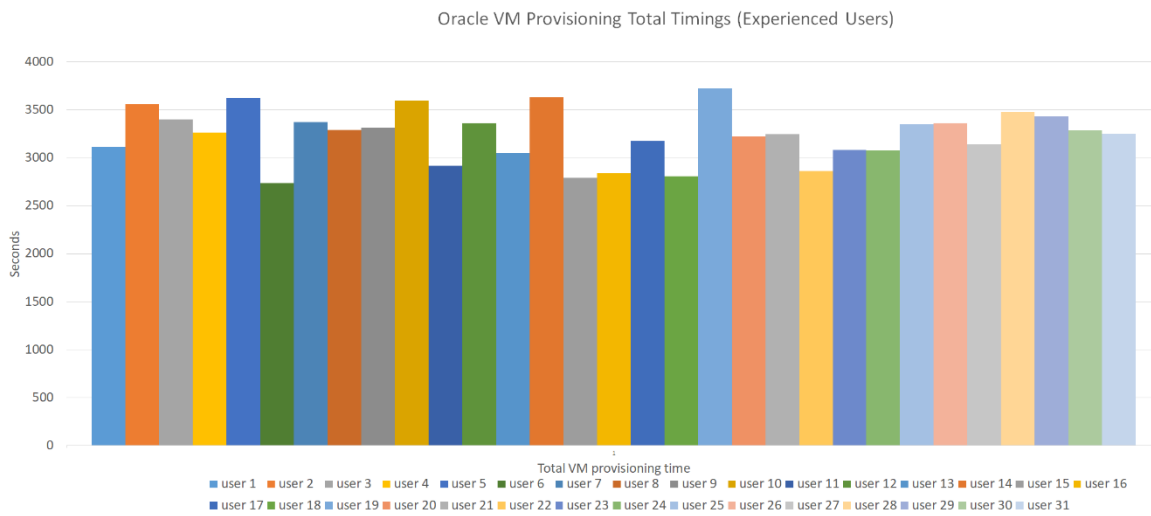


Figure 5.15 Oracle Aggregated Timed VM Provisioning – Experienced Users

The third graph of results presented are the for the AWS cloud platform (experienced):

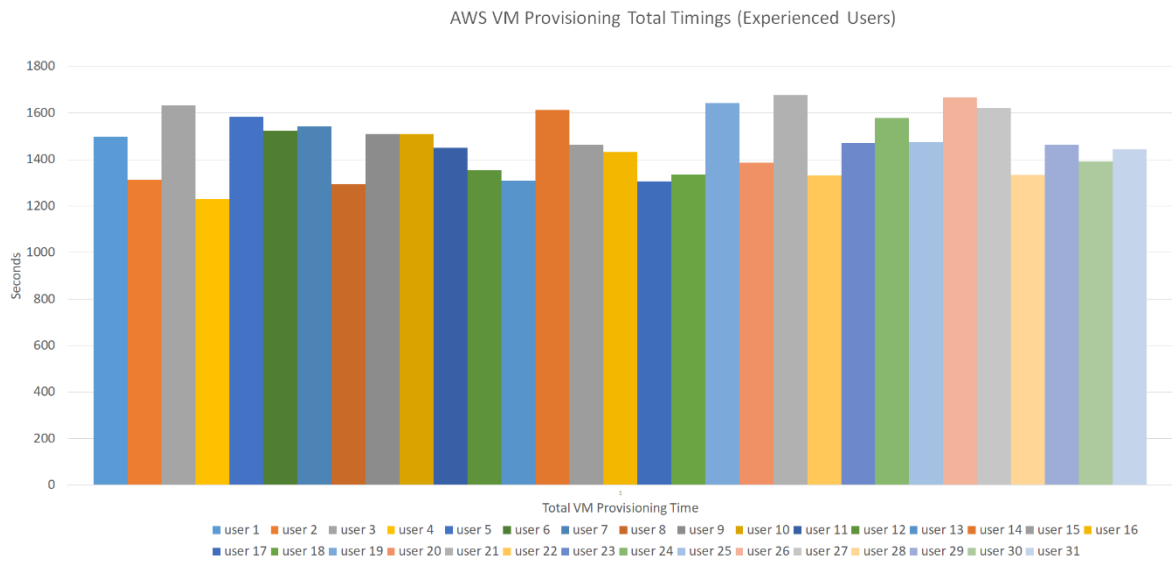


Figure 5.16 AWS Aggregated Timed VM Provisioning – Experienced Users

5.3.2.3 Novice Users

The charts below show the results for the ‘novice’ user group, who performed the VM provisioning experiment on all platforms; the first graph of results presented are for the IDE (Novice):

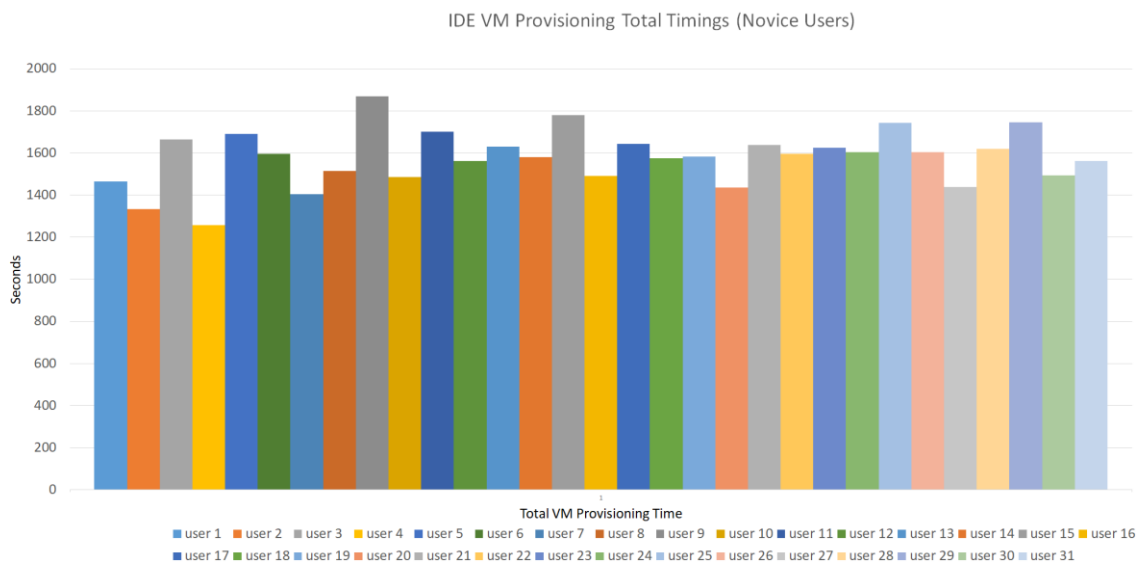


Figure 5.17 IDE Aggregated Timed VM Provisioning – Novice Users

The second graph of results presented are the for the Oracle cloud platform (Novice):

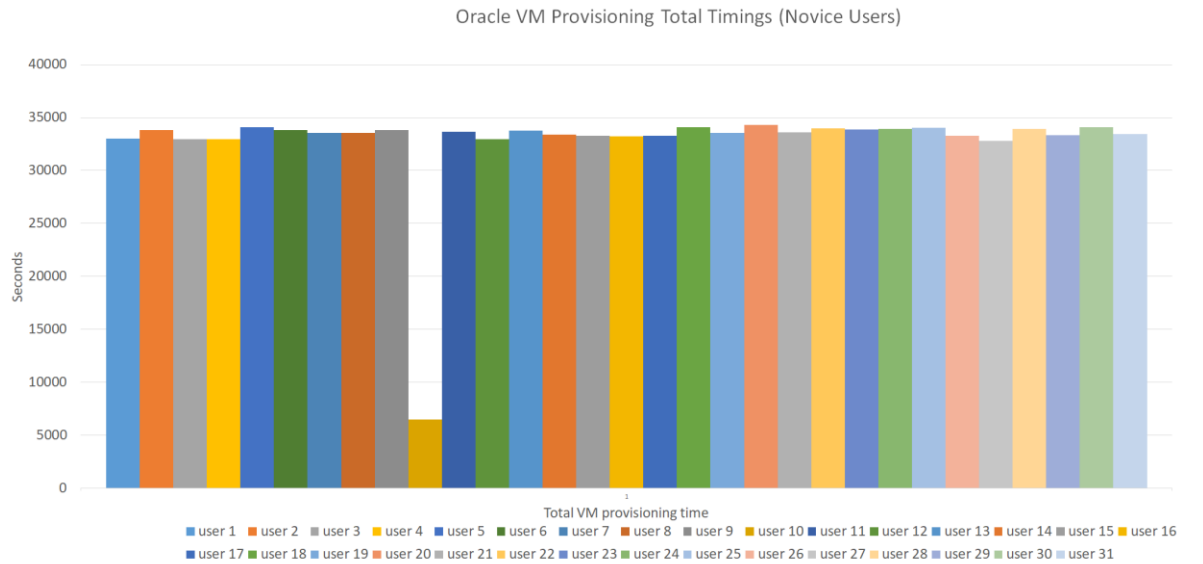


Figure 5.18 Oracle Aggregated Timed VM Provisioning – Novice Users

The third graph of results presented are the for the AWS cloud platform (Novice):

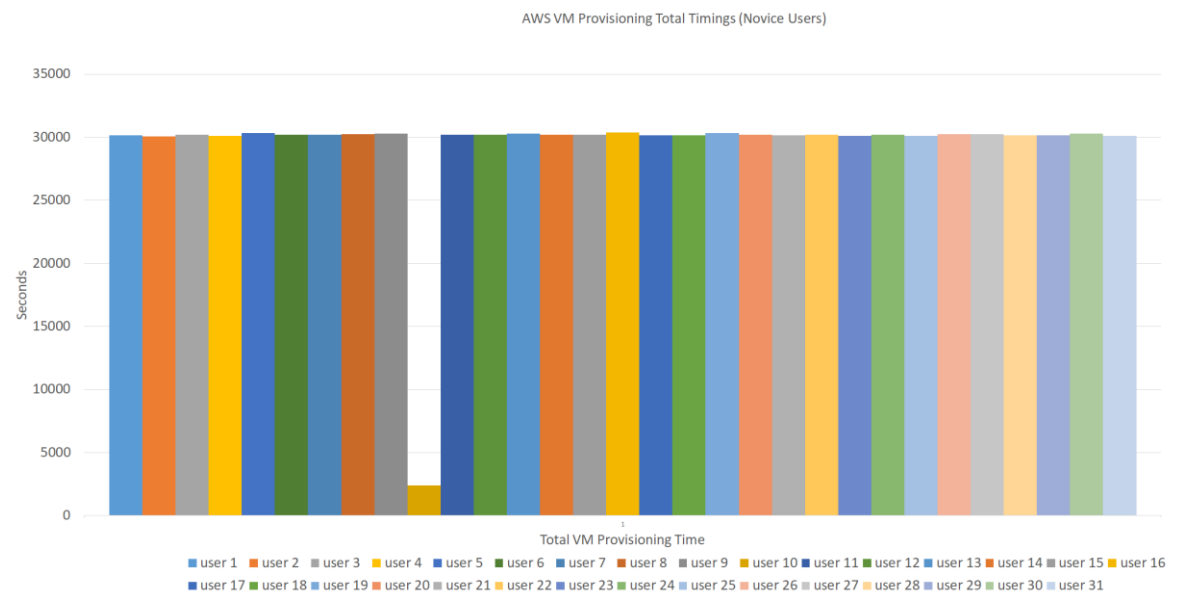


Figure 5.19 AWS Aggregated Timed VM Provisioning – Novice Users

5.3.3 Cognitive Load Rating Results

As part of the experiments undertaken, three principle sets of results for the end-user demographic were collected. These include expert, experienced and novice user groups (as defined previously in table 5.2 End-User Types). Each user was observed, and the result for the 10-step VM Provisioning process are listed in table 5.6 VM provisioning sequence; the results below provide the output for 3 sets of users listed in figure 5.20 (experts users), 5.21 (experienced users) and 5.22 (novice users) respectively, which describe the cognitive load experienced by each group of users, as described by the CLR guide in section 5.2.3.5:

The charts below show the combined results for all three experimental platforms (IDE, Oracle and AWS), who performed the cognitive evaluation performance; the first graph of CLR results presented are the for the ‘expert’ user group:

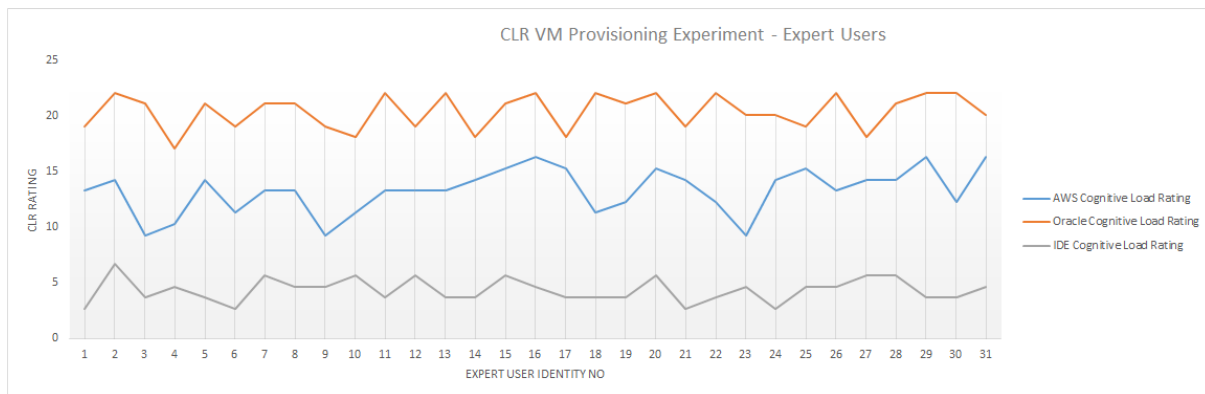


Figure 5.20 CLR VM Provisioning – Expert Users.

The second graph of CLR results presented are the for the ‘experienced’ user group:

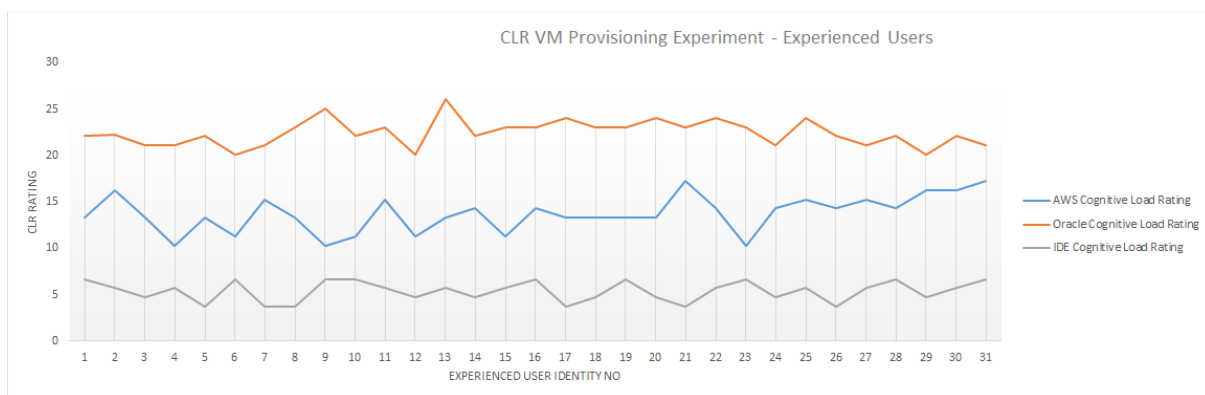


Figure 5.21 CLR VM Provisioning – Experienced Users.

The third graph of CLR results presented are the for the ‘novice’ user group:

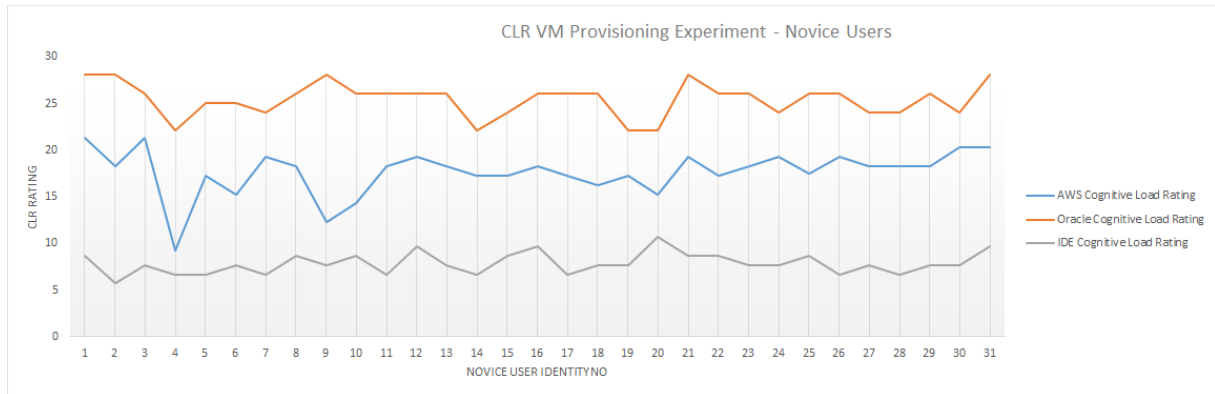


Figure 5.22 CLR VM Provisioning – Novice Users.

Note that the conclusions for the experiment can be found in section [8.2.2](#).

5.3.4 Overall Results

Table [5.9](#) below summarises the results for VM provisioning times and the CLR for the AWS, Oracle and IDE platforms respectively:

Tested Platform and User Group	Mean Average VM Provisioning Time (Sec)	Mean Average CLR (Descriptor) - See CLR guide chart in section 5.2.3.5
AWS Novice Users	9999*	Medium
Oracle Novice Users	9999*	Medium-High
IDE Novice Users	1578	Low
AWS Experienced Users	1464	Low-Medium
Oracle Experienced Users	3237	Medium-High
IDE Experienced Users	1372	Very-Low
AWS Expert Users	1382	Low-Medium
Oracle Expert Users	2362	Medium
IDE Expert Users	1231	Low

Table 5.9 VM Provisioning Experiment Results

* 9999 results are recorded where end user participant groups were unable to complete the VM provisioning process.

Note, the table provides an average VM provisioning time for each participant group, with the appropriate CLR descriptor; the descriptors are an average of the entire sequence of the provisioning 10-steps, therefore, this should be taken into account, even though for step 8 (SSH Key) the novice groups for AWS and Oracle platforms generally failed to load a public key, and the majority described the cognitive load for this step as an ‘Exceptionally high requirement’. The significance of the results are discussed in more detail for each platform in Chapter [8](#).

5.4 Summary

This chapter is key in providing the details and results of two of the five experiment processes conducted as part of this study. Firstly, the VM simplified deployment experiment was described with the 10-step procedure for end-user participants, along with the controls to provide definitions, user types, complexity value weightings and task types. The end-user results are then captured, recorded and presented in graph format. Secondly, the end-user experience data is captured to analyse the cognitive load and mental power requirement for each of the respective systems, using the CLR formula to allow a comparison against the cognitive load guide chart. As with the first experiment, the data is presented graphically. The results generated for both experiments show a reduction in VM provisioning time for the IDE and a lower mental power requirement, when compared to the other platforms, which are AWS and Oracle respectively. The three key groups show a similar pattern, albeit with reduced times for provisioning for ‘expert’ and ‘experienced’ level users. For the IDE, step 9 accumulates most of the VM build time, due to the fact most of the other steps are automatic, or semi-automatic. In comparison, the standard AWS and Oracle end-user cloud provisioning platform interface (see [Appendix C](#)) has a requirement for more manual user inputs, thus adding more time to the aggregated VM provisioning time. Typically, we observe time consuming manual inputs around step 8, the SSH-key load, and additionally for the Oracle cloud, step 4 selecting the machine image and step 6 for defining the VM parameters. Novice

VM provisioning end-user results show that the SSH-key load at step 8 was for the most part, too challenging to complete. Finally, we compare the cognitive load rating results for the expert, experienced and novice end-user groups, to ascertain how mentally challenging the participants found the experimental VM provisioning exercises. The next chapter discusses experiment 3, which addresses VM workload, migration and failover strategies.

Chapter 6: Improving Workload Migration Strategies

6.1 Introduction

For most organisations, being able to maintain highly available (HA) systems is essential to ensure their business operations continue to run effectively ([Fernado et al, 2016](#)). As discussed previously in section [4.7](#), we examined in detail how the IDE maintains HA using a cluster and quorum voting mechanism. Public cloud vendors like AWS and Oracle hide this complexity from their customers and end-users, by using concepts such as regions, which are physically distinct geographical locations, such as Western Europe London, or the US East North Virginia; most regions have at least two physically separate datacentres to make them resilient to local failures, and each datacentre has its own Reliability, Availability, and Serviceable (RAS) features, such as redundant power, network switches, servers and so on.

Some of the users are therefore unaware of the engineering expertise, effort and cost associated with creating this type of availability and resiliency, which is one of the reasons for the commercial success of such platforms ([Kokkinos et al, 2013](#)). That being the case, because many commercial cloud providers keep the complexity and know-how as intellectual property secrets, this makes it harder for researchers to compare and study such technologies in lab-based experimental conditions ([Hataba and El-Mahdy, A, 2012](#)). Therefore, as part of this work, we analyse two well-known VM failover technologies called XenMotion and vMotion, for which there are available comparative studies completed, to allow a detailed analysis and comparison against the IDE failover/migration process ([Feng et al, 2011](#); [Shirinbab and Lundberg, 2016](#)).

6.2 Workload Migration Methods

There are several workload migration methods available, however, this study approach initially begins with the ‘full restart’ VM scenario, although comparisons are made against ‘live migration’ methodologies, and the results obtained therein ([Feng et al, 2011](#); [Shirinbab and](#)

[Lundberg, 2016](#)). It was beyond the scope of this investigation to address the in-memory/disk VM replication migration aspects described below (e.g. using VirtualBox teleport); please see limitations section [1.4](#) which describe the constraints, and section [8.3.3](#) that describes the further work to be completed in this area. The commonly available migration/failover methods include the following techniques and methods:

- VM 'full restart' and migration scenario; VM OS is stopped abruptly, crashes, or halts as a result of a physical host failure, typically an uncontrolled failure.
- Planned VM in-memory migration (VM migrates between two physical hosts, and has its memory replicated and is restarted; typically used as a controlled failover); this method being controlled, usually results in less actual downtime of the VM and its associated services, especially when used in-conjunction with 'live migration' techniques.

The next section provides details on the experiment process employed as part of the experiment.

6.3 Experiment Process

The experiment process covered two principle components or stages, listed as follows:

- Detection of a simulated VM failure event via loss of the physical host machine, measured in time taken (seconds).
- Migration and restart of the failed VM to the point it is restarted and operational once more, again measured in time taken (seconds).

The details are described in the two tables below, firstly the preparation steps to ensure the experiment is valid, and secondly the failover/migration process is implemented:

Step No	Description	Measurement/Observation
1	Ensure IDE Cluster is online and operational – 3 node cluster.	Observation of cluster health.
2	Ensure VM test subject is up and running and is also accessible (e.g. using secure shell).	Observation/log into VM and ensure working normally; note, the physical host where the VM is operational (resident).

Table 6.1 Simulated VM Failover/Migration IDE Preparation Steps

Step No	Description	Measurement/Observation
1	Invoke Simulated Failure event.	Observation of physical host failure event for guest VM.
2	IDE detection time of failure event.	Observe and record the time taken to complete the detection process.
3	IDE Failover/migration and restart VM process.	Observe and record the time taken to complete the detection process.

Table 6.2 Simulated VM Failover/Migration IDE Steps

6.4 Experiment 3: Workload Migration and Evacuation of VMs

The subsections below show the three experiments ([3.1](#), [3.2](#) and [3.3](#)) conducted around the VM migration/failover processes for the IDE, vMotion (study 1), and vMotion and XenMotion (study 2) respectively. Considerable effort has been made to ensure the comparisons are as closely matched as possible; some of the experiment conditions vary slightly, but this is noted by the study and highlighted to allow clear results, with acknowledged (minor) differences. The key elements are described here:

- Network bandwidth; this is especially relevant for failover, where there is no shared storage and the network is used to physically copy the VM container devices, such as virtual disks e.g. Virtual Machine Disks (VMDKs). For the experiments which used shared storage such as NAS, this is of little impact to the experiment process in terms of adding time to the migration/failover event.
- CPU/Memory; it is important that the base operating system has the recommended hardware resources available for CPU/Storage/Memory. For VMs heavily laden with applications and databases, this can affect the migration/failover time. To avoid this as a complication factor, VMs with the base OS installed were used, and it was ensured that any applications had the recommended memory/CPU available.
- VM Storage type; very significant if using shared cluster storage, such as NAS or SAN. In cases, where there is no shared storage between cluster nodes, the VM's virtual disks (operating system, applications, and databases), need to be copied to the target system as part of the process. This creates very intensive network traffic (due to replication), and usually results in longer sustained outages ([Awal et al, 2014](#); [Toyoshima et al, 2010](#)).
- Operating system – Linux (Redhat 6x, or CentOS 6.x, ensuring that the OS instance and applications have the recommended resources available ([Redhat, 2019](#)).

6.4.1 Experiment 3.1: IDE VM Migration/failover Process

It was expected that the invocation of IDE rule listed in section [4.8.2](#) and table [4.14](#) IDE knowledge rule, would take effect as part of the experiment process, to evacuate the failed VM from the failure physical host for that particular guest VM. As part of the experiment, this rule was observed to detect the failure event, and invoke its knowledge rule

and consequence, which was to perform a migration and failover to a new healthy cluster node. The diagram below in figure 6.1 illustrates the process.

- Node CPU/Memory: 4 Core / 4 Threads / 1.6GHz / 8GB RAM
- Network Bandwidth: 1Gb
- Storage: Shared – NAS
- Hypervisor: VirtualBox 5.2
- Operating System: CentOS 6.2

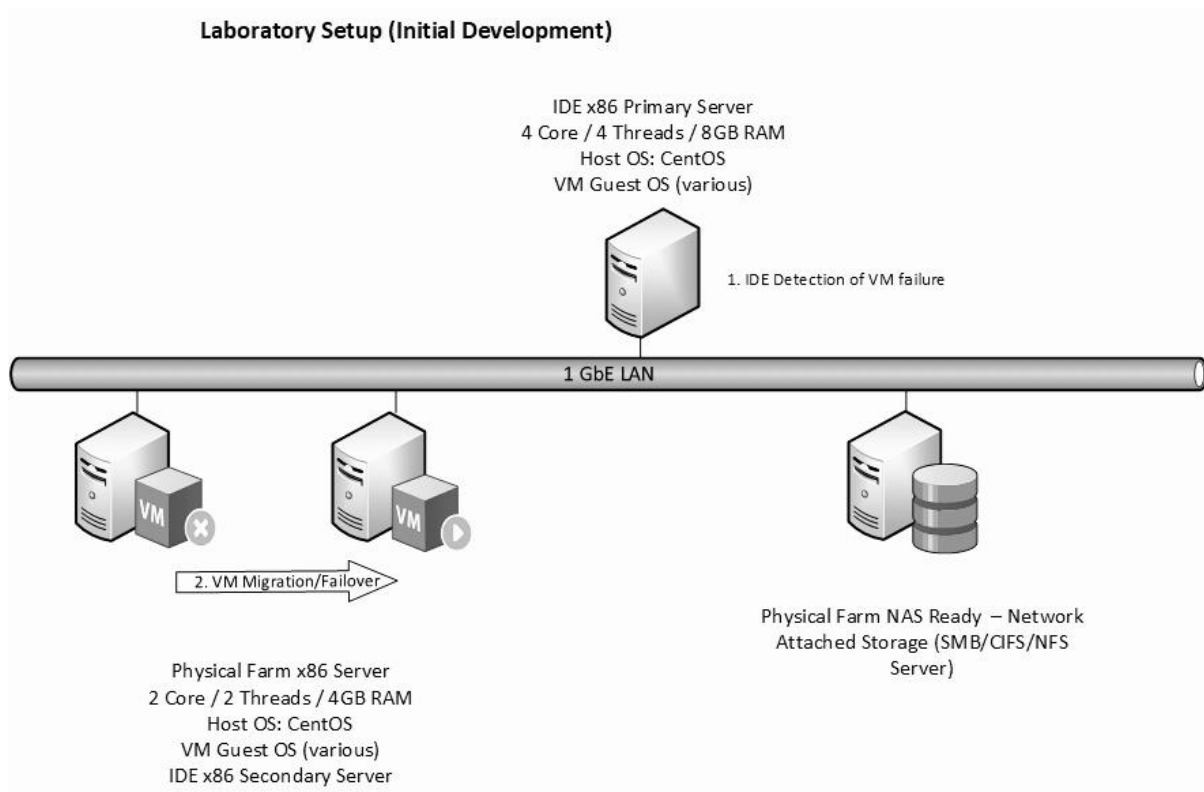


Figure 6.1 Experiment 3.1 VM Failover Method (IDE)

Following the experiment, each test result iteration (Test ID) was recorded 1-6, and had its VM downtime, with detection of the physical host/VM failure listed, along with the notation of the available network bandwidth for potential consumption.

Test ID	Downtime (sec) / Maximum Detection Time (sec) / Network Bandwidth Speed (Gb)	Storage Type	Total Migration Time (sec)
1	14.22 / 7.34 / 1Gb	NAS	21.56
2	15.10 / 7.22 / 1Gb	NAS	22.32
3	16.45 / 7.56 / 1Gb	NAS	24.01
4	15.30 / 7.13 / 1Gb	NAS	22.43
5	15.21 / 7.21 / 1Gb	NAS	22.42
6	14.96 / 7.19 / 1Gb	NAS	22.15

Table 6.3 Experiment 3.1, Downtime and Total Migration Timed Results (IDE)

6.4.2 Experiment 3.2: vMotion VM Migration/failover Process

The process below shows the details on the vMotion migration/failover process; The methodology for the experiment is captured in detail with the diagram below:

- Node CPU/Memory: 12 Core / 24 Threads / 2GHz / 128GB RAM
- Network Bandwidth: 10Gb
- Storage: Shared - virtual NAS
- Hypervisor: ESXi 5.5
- Operating System: Redhat 6.2

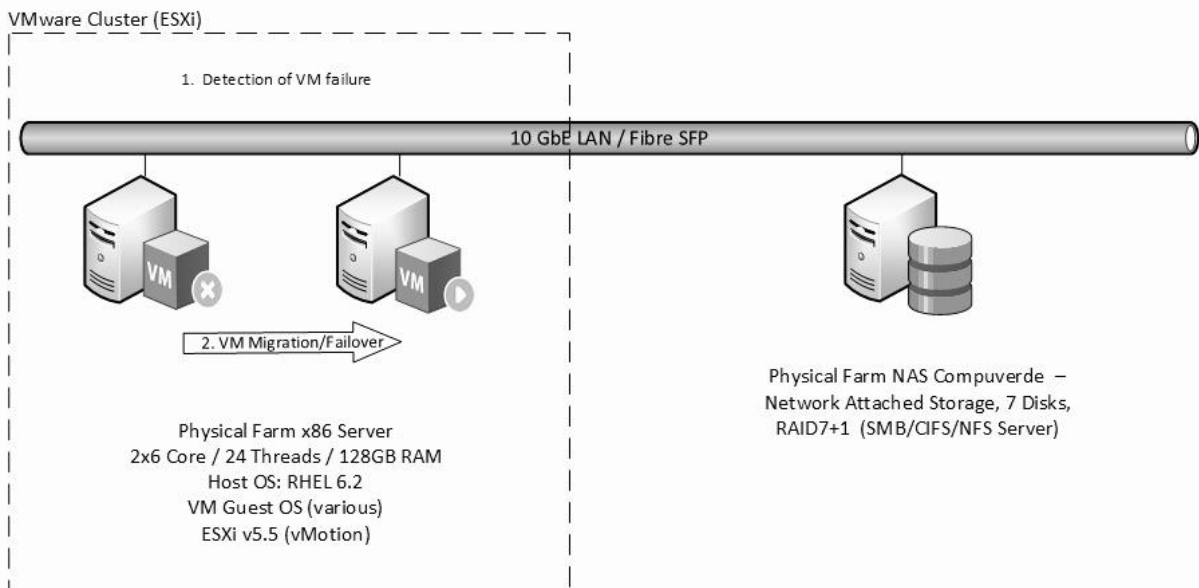


Figure 6.2 Experiment 3.2 VM Failover Method (study 1)

Comparative study 1 ([Shirinbab and Lundberg, 2016](#)) shows that several experiment tests were completed with a maximum down-time and overall total migration time listed in table [6.4](#) below; note the value for Downtime and Maximum Response Time are combined in this experiment:

Test ID	<i>Downtime & Maximum Response Time (sec) / Network Bandwidth Speed (Gb)</i>	<i>Storage Type</i>	<i>Total Migration Time (sec)</i>
1	2.21 / 10Gb	vNAS	30
2	4.01 / 10Gb	vNAS	38
3	2.17 / 10Gb	vNAS	48
4	4.94 / 10Gb	vNAS	52
5	2.92 / 10Gb	vNAS	48
6	4.48 / 10Gb	vNAS	53

Table 6.4 Experiment 3.2, Downtime and Total Migration Results vMotion ([Shirinbab et al, 2016](#))

It should be highlighted that the network bandwidth available for the experiment was 10Gb, which exceeded the other experiments; however, it can be discounted as a large advantage, as the process utilised shared storage (vNAS) for the VM's virtual disks ([Aladyshev et al, 2018](#)). Therefore, this avoided the requirement for virtual disk replication, which would incur high network I/O.

6.4.3 Experiment 3.3: vMotion and XenMotion VM Migration/failover Process

For comparative study 2 ([Feng et al, 2011](#)), the process below shows the details on the vMotion and XenMotion migration/failover mechanism:

- Node CPU/Memory: 12 Core / 24 Threads / 2.66 GHz / 24GB RAM
- Network Bandwidth: from 100Mb to 1Gb (sliding upwards)
- Storage: SAN (Shared - Storage Area Network)
- Hypervisor: ESXi 4.1 & Citrix XenServer 5.6
- Operating System: not specified

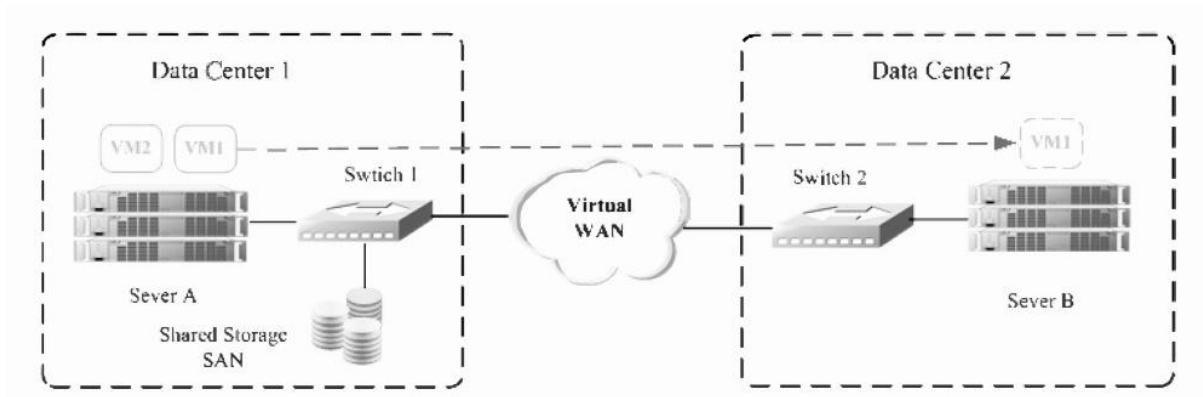


Figure 6.3 Experiment 3.3 VM Failover Method study 2 (Feng et al, 2011)

Note, in this experiment, we observe a sliding scale in time (listed in tables [6.5](#) and [6.6](#)), representing the difference in available bandwidth for the VM migration/failover event to consume. The study initially throttles the bandwidth heavily at only 100Mb; for each test the bandwidth is increased (doubled initially, then by 200Mb) and the results (1-6) are compiled based on a network bandwidth speed rate from 100Mb-1000Mb (scaled up bandwidth with each integration). This still provides interesting comparative results; however, as discussed, the experiment utilises shared SAN storage, and the IDE and previous study 1 both operate their platforms using 1000Mb (or 1Gb) network speeds, which is equivalent for at least the last test – number/ID 6. It can be observed that there are vastly reducing total migration times in the results compiled in tables [6.5](#) and [6.6](#). Note, this study has two sets of results available, one for vMotion, and the other for XenMotion, which is useful in terms of being able to analyse two alternative hypervisor technologies against the IDE.

The following table has the vMotion results for study 2:

Test ID	<i>Downtime & Maximum Response Time (sec) / Network Bandwidth Speed</i>	<i>Storage Type</i>	<i>Total Migration Time (sec)</i>
1	Not recorded / 100Mb	SAN	150
2	Not recorded / 200Mb	SAN	90
3	Not recorded / 400Mb	SAN	50
4	Not recorded / 600Mb	SAN	40
5	Not recorded / 800Mb	SAN	30
6	Not recorded / 1Gb	SAN	20

Table 6.5 Experiment 3.3, Downtime and Total Migration Results vMotion ([Feng et al, 2011](#))

The following table has the XenMotion results for study 2:

Test ID	<i>Downtime / Maximum Response Time (sec) / Network Bandwidth Speed</i>	<i>Storage Type</i>	<i>Total Migration Time (sec)</i>
1	Not recorded / 100Mb	SAN	700
2	Not recorded / 200Mb	SAN	400
3	Not recorded / 400Mb	SAN	200
4	Not recorded / 600Mb	SAN	120
5	Not recorded / 800Mb	SAN	100
6	Not recorded / 1Gb	SAN	80

Table 6.6 Experiment 3.3, Downtime and Total Migration Results XenMotion ([Feng et al, 2011](#))

6.5 Results

The charts below show the event VM failure detection time for the IDE, based on system becoming aware of the failure event described earlier in section [6.4.1](#). Note that the conclusions from the experiment can be found in section [8.2.3](#), along with notes in the further work section [8.3.9](#), which provide more information on a detailed laboratory analysis and study opportunity, focused on a vMotion and XenMotion configuration and build, to enable the exact same tests for all three platforms investigated during experiments [3.1](#), [3.2](#) and [3.3](#).

The following chart show the IDE failure detection time for a VM failure event (VM down):

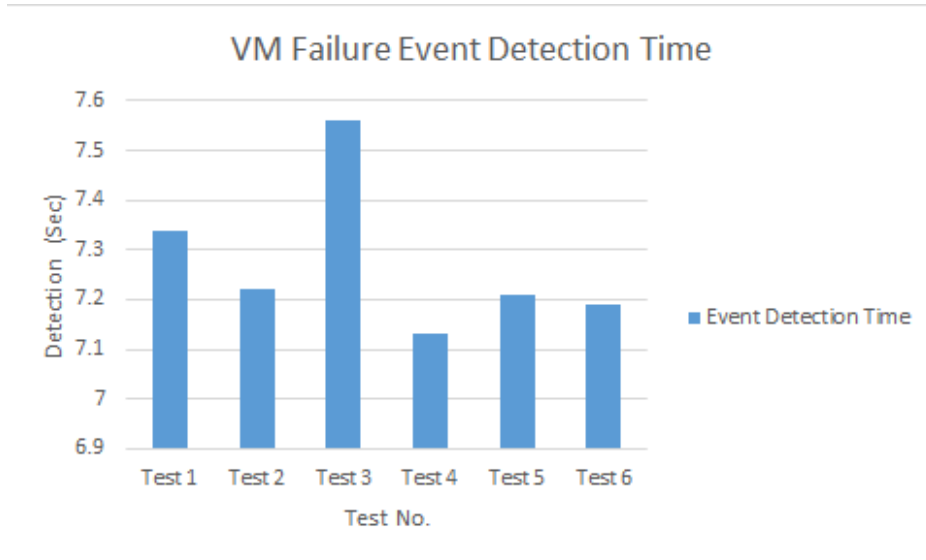


Figure 6.4 IDE VM Failure Detection Time Experiment 3.1 (IDE)

Additionally, the migration time is included in the chart below to show the overall time to complete the end-to-end event detection, migration and failover process:

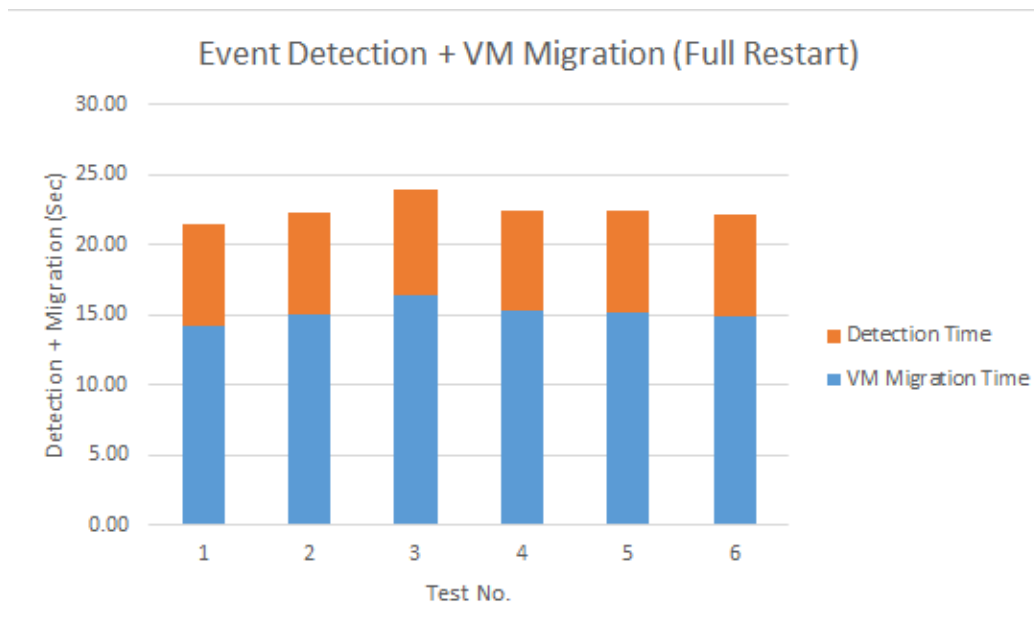


Figure 6.5 IDE VM Failure Detection and Migration Time Experiment 3.1 (IDE)

The graph below shows study 1 results for VM migration and full restart:

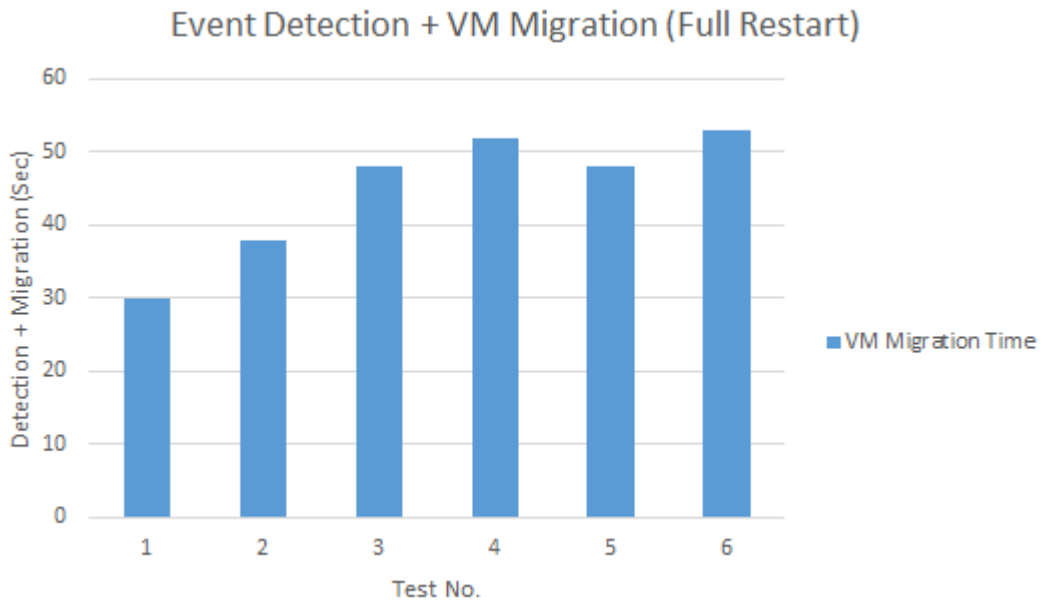


Figure 6.6 Study 1 VM Failure Detection and Migration Time Experiment 3.2 (vMotion)

The graph below shows study 2 results for VM migration and full restart for part a (vMotion):

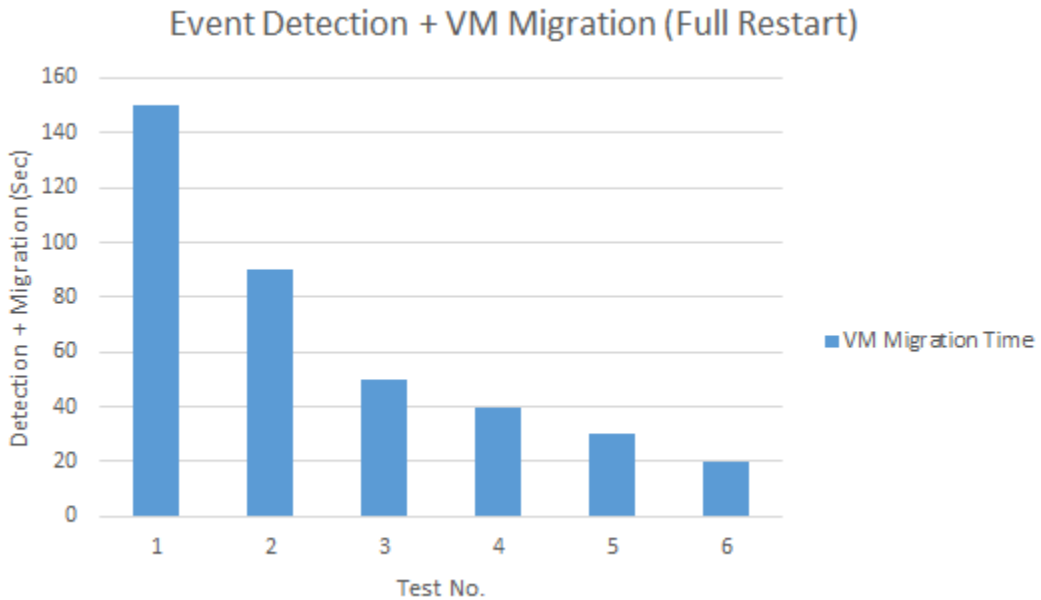


Figure 6.7 Study 2 VM Failure Detection and Migration Time Experiment 3.2 (vMotion)

The graph below shows study 2 results for VM migration and full restart for part b (XenMotion):

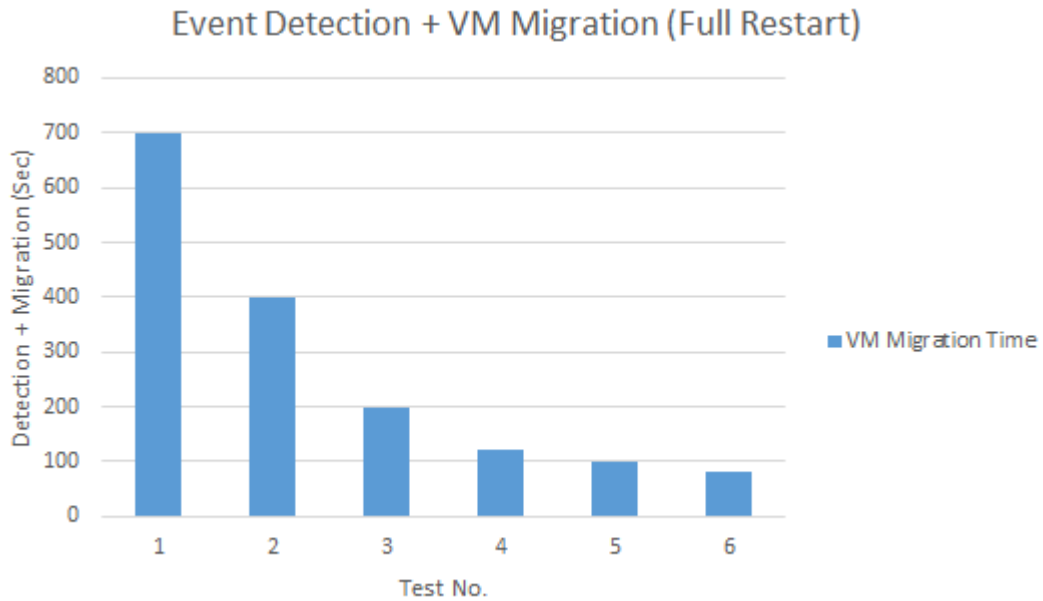


Figure 6.8 Study 2 VM Failure Detection and Migration Time Experiment 3.2 (XenMotion)

The graph below shows the IDE, study1, and study 2 results (part a and b) and the mean average time in seconds for VM migration and full restart:

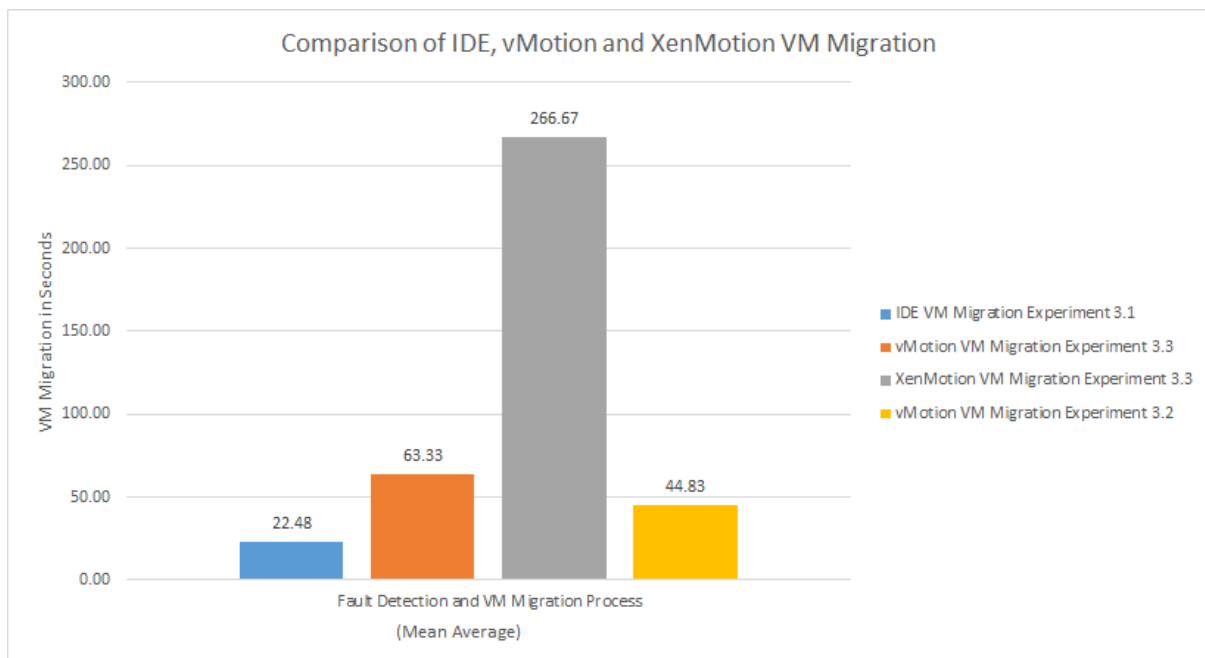


Figure 6.9 Comparative Mean Average VM Migration Time for Experiments 3.1, 3.2 and 3.3

6.6 Summary

This chapter provides the detail for experiment three, which is concerned with VM migration/failover strategies to improve availability of virtualised systems and resources. Two approaches are considered, firstly around full VM restart and secondly around live VM migration. This experiment deals with the full restart scenario following a physical or virtual system failure event. During the experiment, a simulated failure is invoked to allow the IDE to take the necessary intervention steps to recover the VM and associated resources. The experiment captures the amount of time the IDE takes to migrate and failover the VM and its resources, to the point where it has been successfully restarted. The IDE results are then compared against two independent papers, which utilise two well-known products vMotion and XenMotion to demonstrate similar VM migration and failover processes. The IDE performs well when compared with its lower average failure detection and migration VM failover time. The following chapter discusses the final two experiments on the topic of global resource management of virtualised computer systems.

Chapter 7: Optimising Performance and Availability of Virtual Machines

7.1 Introduction

In this chapter we examine the concept of managing VM resources online using direct invention of a system to change resource parameter settings, such as CPU and memory, with the desired goal of being able to dynamically change these values without interruption to service, while the system is live and in a running state. It is important at this point to define the difference between global and local resource management; the definitions are given below for the purposes of this study:

- A local resource management strategy features the resource controls (e.g. for CPU/memory) being applied to a single physical host and its associated local guest VMs. The control never extends to other physical hosts, and there is no overall global view of a pool of physical hosts clustered, either locally or in a remote/distributed fashion.
- A global resource management strategy features resource controls (e.g. for CPU/memory and I/O) being extended across an entire cluster of physical systems and their associated VMs. The resource scheduler is able to continually work and control the overall global capacity/performance across all physical hosts.

There are two resource management scenarios that the IDE can currently work with based on its rule-base, which are as follows:

- A scenario 1 whereby a physical host in the cluster (globally managed) is running short on memory or CPU resource, and it needs to start evacuating guest VMs (in least important order to service) to another physical host in the cluster to free up resources. Please see section [4.8.2](#) physical system events for more details.

- A scenario 2 where a single VM has CPU and/or memory resource issues, and it needs to be given more in order to keep itself processing and functioning effectively. The IDE system will attempt to dynamically resize the VM accordingly based on the physical resources remaining within the cluster in the most effective way possible. Please see section [4.8.3](#) VM system events for more details.

As part of this study we examine scenario 2 in detail, while comparing the IDE function and capability to two similar studies on memory and CPU resource management techniques in virtualised environments ([Zhang et al, 2017](#); [Zhang et al, 2016](#)). The first study looks at automated memory management on a physical system with VMs using the Xen Balloon driver; the second uses an iBalloon driver to help dynamically manage and optimise physical systems with VMs, initially using the KVM (Kernel-based Virtual Machine) driver. The aim is to work on the IDE's rules to test and ensure they are invoking correctly as described earlier by section [4.8.3](#) VM System events, tables [4.17](#) (Memory overload) and [4.18](#) (CPU overload). They describe the automatic intervention being taken against a VM during a sustained 5-minute interval where the CPU and/or memory is utilised above a 75% threshold for either total, as reported by the system performance measurement tools; for example, Linux OS monitoring tools such as vmstat, iostat, and top ([Lui et al, 2015](#)).

More details on the knowledge rules and justifications for those figures can be found in section [4.8.5](#) where we explore in depth in the reasons for certain thresholds (such of VM memory utilisation). Consequently, the questions that arise from this potentially complex resource management process are:

- How long should the intelligent systems wait in terms of time (seconds) before taking direct intervention? ([Song et al, 2013](#); [Ismail and Riasetiawan, 2016](#))
- How often (frequency) should the VM performance statistics be sampled? This would include taking a resource snapshot samples at point in time intervals to record CPU and memory usage on the VM ([Jeong and Lee, 2012](#)).
- Should VM's resources only ever grow, rather than grow and shrink? What is the most effective method for the virtualised platform, for example a grow only policy,

or a grow and shrink policy ([Makridis et al, 2017](#))

- By what amount should CPU and/or memory be increased, and should there be safety thresholds? For example, the 'intelligent system' may add an arbitrary figure of 25% additional CPU and/or memory resource for a VM which has passed its thresholds for a sustained period of time. However, if there were a process on the system that had become rogue ([Joy et al, 2014](#)), and it continued to consume resources, a never-ending pattern of adding additional resource could potentially be used to exhaust all resources, and even starve the physical host, if safety features are not built in to the intelligent system ([Hwang et al, 2010](#); [Chen et al, 2013](#)).
- What comparative features and methods are used by each of the systems, and what are the most effective? For example, examination of the key areas would include:
 - VM resource measurement poll interval.
 - VM resource grow and/or shrink policy.
 - VM resource increase strategy.
 - VM sustained time threshold trigger (for CPU and memory).
 - Overall time taken to resolve a resource issue affecting a VM.

The answers to these questions, are not necessarily easy to identify, as there can be a number of events that compound to cause single or multiple effects, such as a number of rogue processes consuming CPU resource, or a process with a memory leak so consuming all memory (RAM). Killing off these processes, and restarting could potentially resolve the issue; however, as a complication, once restarted, they could begin to malfunction again, thus creating a repeat problem. Therefore, being able to spot and identify a re-occurring pattern is a useful technique for CPU and memory resource management function. As part of the discovery process, in terms of being able to test and observe and compare similar methods, further experiments are conducted based on the rules created to enhance system utilisation and better manage VMs within the virtualised environments.

7.2 Experiment Process

The experiment process looks to take advantage of existing hypervisor memory balloon drivers, as well as CPU hot-plug drivers ([VirtualBox Memory, 2019](#); [VirtualBox CPU 2019](#)). Table [7.3](#) identifies the drivers used by each unique study. This is critical, as the performance strategy for each approach must use that type of hypervisor technology driver to enable dynamic resource controls and implement the most effective performance management approach. In addition to the preliminary VM performance algorithm found in table [4.8](#) 'Preliminary Performance Monitoring', the capability is extended further in table [7.1](#), by moving from an initial preliminary performance algorithm, which explains at a high level how the IDE manages generic resource controls, to how, in this instance, it specifically controls CPU and memory resource. This extended algorithm builds on the preliminary idea by extracting the specific knowledge performance rules found at tables [4.17](#) and [4.18](#), and introduces controls and processes around CPU and memory resource; for example, by setting threshold alert values, the interval sampling rate, overall monitoring period and the resulting specific consequent actions to be invoked:

7.2.1 IDE VM Performance Algorithm

```
// IDE VM Performance Algorithm

INPUT: VM Knowledge CPU/Memory Performance Rules
OUTPUT: Return performance metrics, and invoke CPU / Memory resource management if needed

WHILE True
  SSH-to ${host} & Run Local Perf Script
  Capture 75% values for Total CPU & Memory respectively (thresholds)
  FOR each second up to 300
    Sleep 1
    Use local perf tools to capture stats
    Let Total CPU%+=CPU Performance Increment Value
    Let Total Memory%+=Memory Performance increment Value
  END FOR
  Evaluate  $\sum$  (Total CPU%) / 300
  Evaluate  $\sum$  (Total Memory%) / 300

  IF (Average Total CPU% >= 75% average) THEN
    Invoke VM CPU HOT PLUG + 25% or+1 CPU Core
  END IF
```

```

IF (Average Total Memory% >= 75% average) THEN
  Invoke VM Memory Balloon + 25%
  IF VM Memory Balloon Exhausted THEN
    Invoke VM Restart/Memory Resize
  END IF
END IF
DONE

// End of algorithm

```

Table 7.1 IDE VM Extended Performance Resource Management Algorithm

7.2.2 VirtualBox Memory Balloon Driver

It should be noted that the VirtualBox balloon driver works by overcommitting memory to the VM or set of VMs during its initial configuration. The memory remains in a committed state within a managed reserve pool by the hypervisor. Therefore, a strategy is needed to develop a memory reserve pool to allow the VM to flex upwards or downwards as necessary for example by 25%. The only way to currently manage and resize the VM CPU/memory maximums is to power it off and then physically alter the VM parameters as needed, and then restart. Therefore, this means that if the 'over-commit memory' value is not sufficient (or high enough) in size, the only option is to then perform a controlled stop of the VM and then resize and power on and restart the VM ([Zhang et al, 2016](#)).

With the build of the VM we allow an overcommit of 25% of the total memory allocated for the VM, to provide dynamic memory ballooning potential. The upfront over commitment later enables the IDE global resource scheduler to flex the memory up to a 25% increase at a given point in time. The exact amount of overcommitment is one of the critical questions, as there is a trade-off, in that it is reserved upfront by the hypervisor and may not be used outside that framework easily, and it can result in an under-utilisation of the overall system memory resources. Therefore, providing some potential for dynamic memory allocation/ballooning is useful, without diminishing the overall memory utilisation too excessively ([Chen et al, 2013](#)).

7.2.3 Simulate VM CPU and Memory Stress

To perform the experiment and create the correct simulation for constrained CPU and memory resource we use the stress utility to perform this process ([Ismail and Riasetiawan, 2016](#)). As part of the author's experiment process described in section [7.4](#) and [7.5](#), the following *stress-ng* command is used to simulate systems resource stress on CPU and memory respectively.

Where x is equal to the number of CPU cores the VM has:

- `stress-ng --vm 4 --vm-bytes 85% --timeout 300s -v`

The above command runs a simulated stress event against the memory resource for VM and will consume up to 85% of the overall resource available and then cease after 300 seconds.

- `stress-ng --c [x] -l 85 --timeout 300s -v`

The above command runs a simulated stress event against the CPU resource for VM and will consume up to 85% of the overall resource available and then cease after 300 seconds.

The following graph at figure [7.1](#) shows the simulated stressed VM under load for 300 seconds (a 5-minute period), while experiencing a high (but expected) sustained CPU and memory load, as a result of the above commands. As can be observed, the CPU and memory load average are ~80-83% during the monitoring period for each critical resource, as is the intention for the experiment process. This is performed in order to observe the IDE knowledge rules invoke and trigger a response, as defined in the VM System events and tables [4.17](#) (Memory overload) and [4.18](#) (CPU overload):

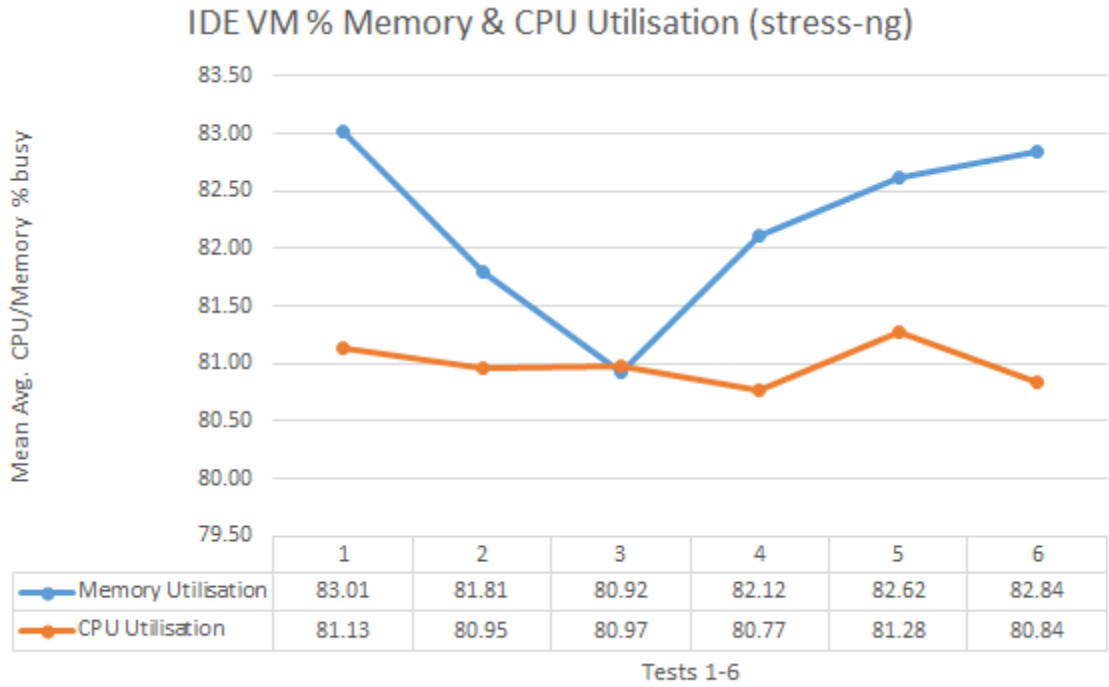


Figure 7.1 IDE VM Simulated Tests for Load Stress (using stress-ng)

7.2.4 Characteristics Compared Against Other Studies

The following table describes the characteristics (binomial yes-no) and controls that could be potentially applied to a dynamic resource manager/scheduler for virtualised systems. This is useful, because it shows the overall capability being provided by each of the studies experimental approach. Having more characteristics available potentially allows for improved resource management for VMs, due to it being feature rich and having less requirement for any manual human/administrator type interventions ([Rothenberg et al, 2017](#); [Chen and Suen, 1993](#); [Conrath and Sharma, 1991](#)).

Reference no	Available Features	IDE (VirtualBox Driver)	Study1 (Xen Balloon, Zhang et al, 2017)	Study2 (iBalloon, Zhang et al, 2016)
1	Dynamic CPU increase.	Yes	No	No
2	Dynamic memory increase.	Yes	Yes	Yes
3	Dynamic I/O increase (Network, Storage).	No	No	No
4	Automatic resource issue detection.	Yes	Yes	Yes
5	Dynamic CPU reduction.	Yes	No	No
6	Dynamic memory reduction.	Yes	Yes	Yes
7	Dynamic I/O reduction.	No	No	No
8	x64-bit architectures support (Balloon Driver).	Yes	Yes	Yes
9	x32-bit architectures support (Balloon Driver).	No	Yes	No
10	Manual administrator intervention required to increase (Balloon) Memory.	No	No	Yes
11	Manual administrator intervention required to increase (Hot-plug) CPUs.	No	No	No
12	Manual administrator configuration and setup of resource management utility.	No	Yes	Yes
13	Is the solution a global resource scheduler?	Yes	Yes	Yes
14	Is the global scheduler part of an Integrated System?	Yes	No	No

Table 7.2 Binomial Comparative Resource Performance Features/Characteristics

It is necessary at this point, to provide some additional detail regarding the characteristics and properties of the features listed in table [7.2](#). Reference points, 1, 2, 3, 5, 6

and 7 are parameters that can be potentially altered, that have a direct effect the resource capability for a VM. In this case, the ability to add or reduce CPU, memory or I/O resource for a VM. I/O resource could include adding or removing network capability such as virtual network interfaces (vNICs), storage devices or fibre channel host-bus adapters. All such functions involve direct communication and manipulation of the hypervisor layer of the system, in this case either VirtualBox, Xen or KVM. Reference points 4 and 13 are quite closely linked, although subtly different. Point 4 includes automatic detection of resource issues could be either a local or global function; in other words, it could run locally on a single physical host, or be globally managed across an entire suite of systems. This leads us to reference point 13, global scheduling, which is the ability for a system to monitor and control resources across the entire collection of machines it administers. For example, you may have a local scheduler, running on a single machine, where the context is management of just that local system, irrespective of the wider view of the entire cluster of managed systems. A global resource scheduler on the other hand, has an entire view of the cluster and uses algorithms to control resources across the entire pool it manages.

This is advantageous because it potentially allows for the more flexible use of resources, whereby a system which is not as busy for a time can lend its resources back into a collective pool, to be consumed and used by a system demanding more resource. This ability to variate resource controls across a group of systems is therefore is an attractive feature. Reference points 8 and 9 are interesting, as they revolve around support for 32-bit and 64-bit architectures respectively. 32-bit support is available for the legacy architectures, however, the practical use of this is somewhat limited by the fact that 32-bit systems have a maximum of 4096MB (4GiB) of addressable memory ([Adl-Tabatabai et al, 2004](#)). This hard limit is compared to 64-bit systems, which can manage up to 16 exabytes of memory ([Mohammad and Ramananjanyulu, 2012](#)). The final point 14, critically records if the system has been integrated as part of an overall controlled system. This is very important as it means the resource (global) scheduler feature can contribute to a list of compounded benefits for a systems overall management capability; in other words, build a critical mass of useful characteristics that can be argued as substantiating the features of an 'intelligent system' ([Guerlain et al, 2000](#)). The following balloon drivers were used by each comparative study:

Study/Engine	Hypervisor Driver
IDE	VirtualBox Balloon Driver / CPU Hotplug Driver
Study 1 – Xen Balloon (Zhang et al, 2017)	Xen Balloon Driver
Study 2 – iBalloon Service (Zhang et al, 2016)	KVM (Kernel-based VM) Balloon Driver

Table 7.3 Experiment Balloon/Hotplug Drivers

7.2.5 IDE Global Resource Management

The diagram below in figure [7.2](#) explains how the IDE addresses global resource management in the following ways:

- Using an SSH probe monitor to remotely access and measure performance against all platform physical hosts and guest VMs to enable the retrieval and analysis of all performance data and metrics.
- Where appropriate using local VM CPU hotplug and memory ballooning techniques to increase or reduce resources.
- Where appropriate re-balancing and moving guest VMs to alternative physical hosts.

7.2.6 Comparative Methods Analysis

The following table highlights the three methods undertaken by each study with respect to the global performance resource management of virtualised computer systems; it includes the author's IDE solution, and comparative work completed in study 1's XenBalloon, and study 2's iBalloon investigation ([Zhang et al, 2017](#); [Zhang et al, 2016](#)). A critical analysis

for each platform is provided, and a review of the strengths and weaknesses for each method is highlighted:

Platform Name / General Features	Strengths	Weaknesses
<p>Intelligent Decision Engine (IDE) /</p> <p>The IDE uses a global management system utilising an SSH control algorithm for remote hosts. Additionally, it makes use of its expert system knowledge rules to apply them consistently across the entire platform. It is able to dynamically control the reduction and increase of memory and CPU for VMs, which includes reduction down to a minimum of 1 CPU core per VM.</p>	<ul style="list-style-type: none"> I. Global HA management technique for remote hosts. II. Expert Knowledge rules for the application of consistent platform behaviour, and for adaptive rules. III. Dynamic Reduction of Memory. IV. Dynamic increase of Memory. V. Dynamic Reduction of CPU cores to a minimum of 1. VI. Dynamic increase of CPU cores. 	<ul style="list-style-type: none"> I. Only supports the VirtualBox Balloon driver, and hot-plug features. II. The IDE is highly integrated, meaning it can only be deployed as a whole entity, or not at all.

Platform Name / General Features	Strengths	Weaknesses
<p>Study 1 (Xen Balloon, Zhang et al, 2017) /</p> <p>Study 1 uses an automatic memory control process for guest VMs on physical hosts. It uses a global resource scheduling mechanism, with a resulting toolkit which is opensource. The system runs a VM called domain 0, which has privileges to perform hypervisor operations across the platform. It uses linear equations to determine target VM memory, and uses a memory overcommitment ballooning technique. It can increase or lower memory allocation and is able to balance memory across the managed platform.</p>	<ul style="list-style-type: none"> I. Global management technique for remote hosts II. Opensource software; potentially easy to install as an add-on, as it has been built as a toolkit. III. Dynamic Reduction of Memory. IV. Dynamic increase of Memory. 	<ul style="list-style-type: none"> I. Only supports the XenBalloon driver. II. Does not support dynamic reduction of CPU cores. III. Does not support dynamic increase of CPU cores. IV. Has no documented HA features.

Platform Name / General Features	Strengths	Weaknesses
<p>Study 2 (iBalloon, Zhang et al, 2016) /</p> <p>Study 2 has adopted the following process for its memory management of VMs. It runs two principle user processes, or programs to simultaneously manage the platform. The first, is a VM monitor daemon that continually analyses the memory resources. In conjunction, a balancer process daemon is able to change the memory resource parameters for VMs, by interfacing with the remote hosts KVM balloon driver to dynamically change values.</p>	<ul style="list-style-type: none"> I. Global management technique for remote hosts II. Dynamic Reduction of Memory. III. Dynamic increase of Memory. 	<ul style="list-style-type: none"> I. Only supports the KVM Balloon driver. II. Does not support dynamic reduction of CPU cores. III. Does not support dynamic increase of CPU cores. IV. Has no documented HA features, and two independent daemons which must both be available.

Table 7.4 Comparative Performance Resource Management Studies

IDE Global Resource Scheduler (CPU & Memory)

- Capability to detect performance issues (Physical Host/Guest VMs)
- Modify VM CPU/Memory resources
- Migrate guest VMs to free up or balance CPU/Memory resources

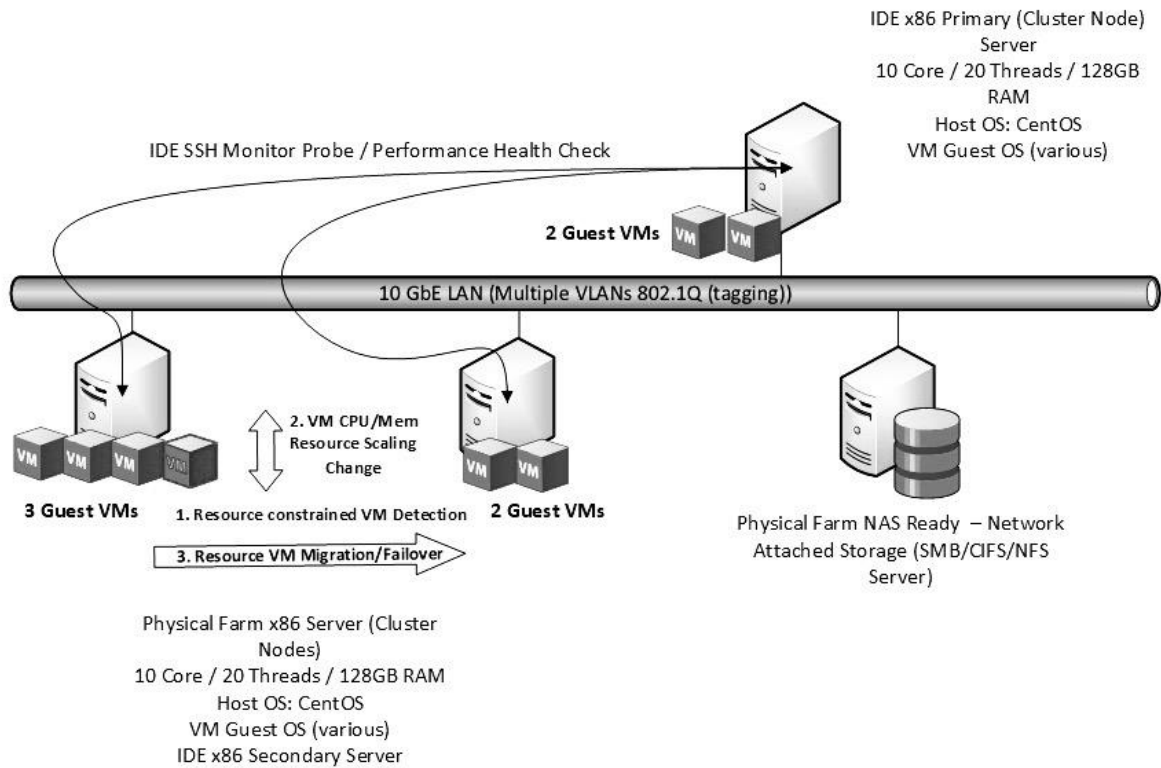


Figure 7.2 IDE Global Resource Management

7.3 Optimisation of System Performance and Availability

7.3.1 x86-64-bit Architectures and Memory Ballooning

Most architectures do not support the memory ballooning function for 32-bit OS systems which have a maximum of 4096MB addressable memory, compared to more modern 64-bit OS systems which can address ~16 exabytes.

7.3.2 x86-64-bit Architectures with CPU Hotplug Features

Most 64-bit architectures support CPU hot-plug features. In the case of VMs, this allows the hypervisor to provision extra CPUs up to the maximum allowed or reduce them to a minimum of one (usually listed as CPU 0).

7.4 Experiment 4: Overload of VM Memory Usage, Detection Time, and Resolution Time

7.4.1 IDE VM Memory Ballooning Process

In figure [7.3](#) below, we observe the results of the IDE VM memory ballooning process while under a simulated memory stress event, as described in section [7.2.3](#) to enable the demonstration of the ballooning process. As VMs within the IDE platform have an over-commitment of memory by 25%, this therefore allows the monitoring period of 300 seconds (5 minutes) to evaluate the VM memory capacity and utilisation state. Once the knowledge rule is validated, the forward-chain reasoning process is initiated, and steps taken to provide the VM with additional memory resource using the balloon driver technique.

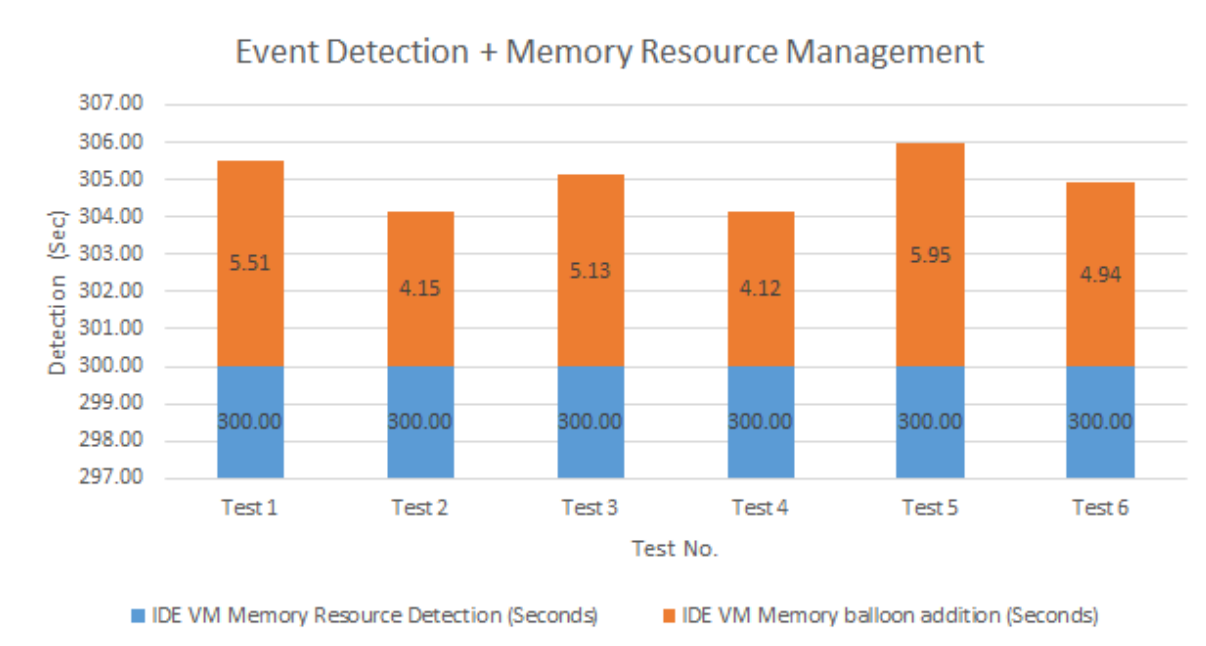


Figure 7.3 IDE Performance Monitoring and Memory Ballooning Results

7.4.2 Study 1 VM Memory Balloon Process

Study 1 utilised the following mechanisms to deliver a memory management system:

- Automatic memory control for physical/guest VMs.

- A global resource scheduling mechanism.
- Runs a VM called domain 0 which has privileges to perform hypervisor operations.
- Uses linear equations to determine target VM memory.
- Uses memory overcommitment.
- Can increase or lower memory allocation.
- Can balance memory across the managed platform.

The diagram in figure 7.4 shows how the Xen Balloon driver manages memory between VMs (Guest OS's). This method for memory ballooning is utilised by the author of study 1.

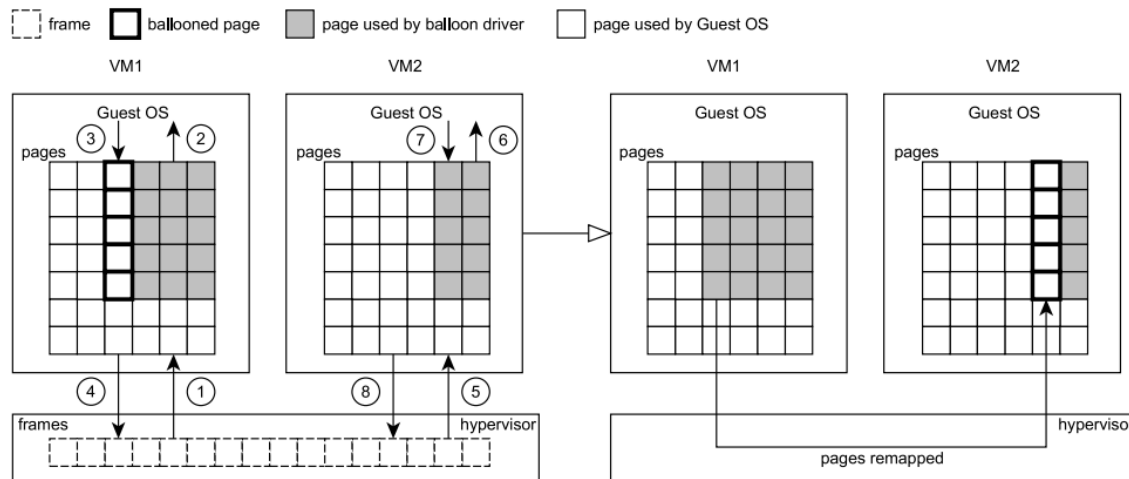


Figure 7.4 Study 1 Xen Balloon Process (Zhang et al, 2017)

The diagram in figure 7.5 below shows study 1's global resource management process using Domain-0 as the control system, to manage memory resources through the ballooning process.

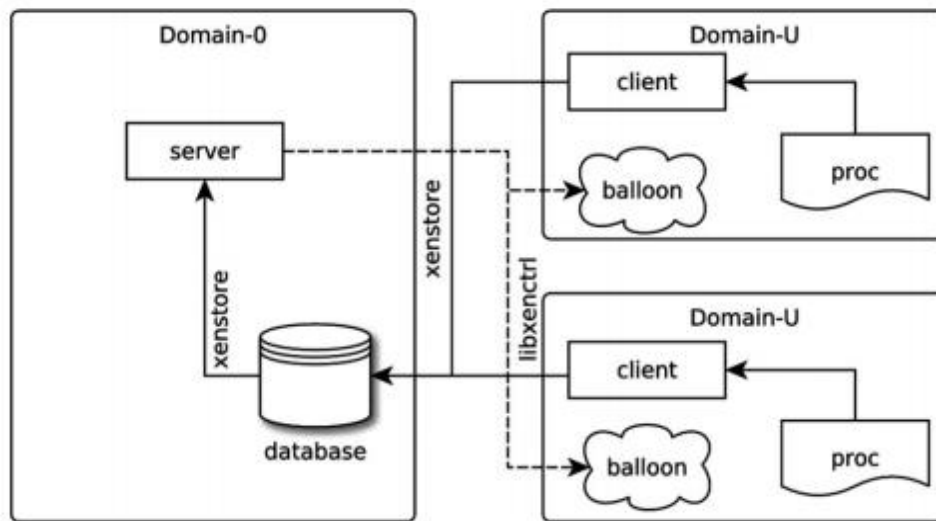


Figure 7.5 Study 1 VM Memory Balloon Process ([Zhang et al, 2017](#))

7.4.3 Study 2 VM Memory Balloon Process

Study 2 (iBalloon) adopted the following mechanism for its memory management approach:

- Runs two principle daemons/user processes simultaneously to manage the platform.
- A VM monitor daemon to continually analyse the memory resource.
- A balancer process daemon which changes the memory resource parameters by interfacing into the KVM balloon driver.

The diagram below at figure [7.6](#) shows the iBalloon memory management process. Notice the different levels of separation (granularity) between the hypervisor (guest levels) and the physical host (Host levels).

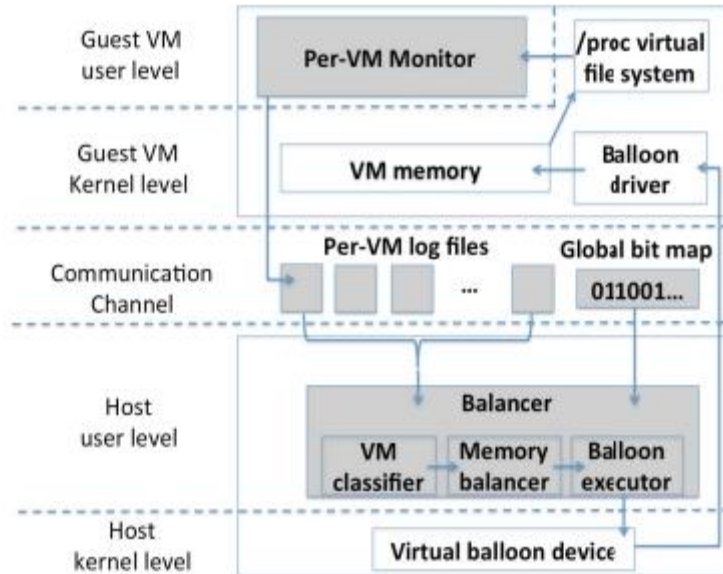


Figure 7.6 Study 2 iBalloon system overview (Zhang et al, 2016)

7.5 Experiment 5: Overload of VM CPU usage, Detection Time, and Resolution Time

7.5.1 IDE CPU Hotplug Process

The IDE uses (as it does with memory management) a standard five-minute poll interval with a sample per second taken. As with the memory stress simulation listed in section [7.2.3](#), in this case the CPU is driven above the threshold alert over the monitoring period. This in turn allows us to demonstrate that the IDE can dynamically increase (hot-plug) spare CPU cores and makes the additional compute power available to the VM in around 5-7 seconds. Figure [7.7](#) shows the detail for detection and the actual CPU hot-plug process time taken:

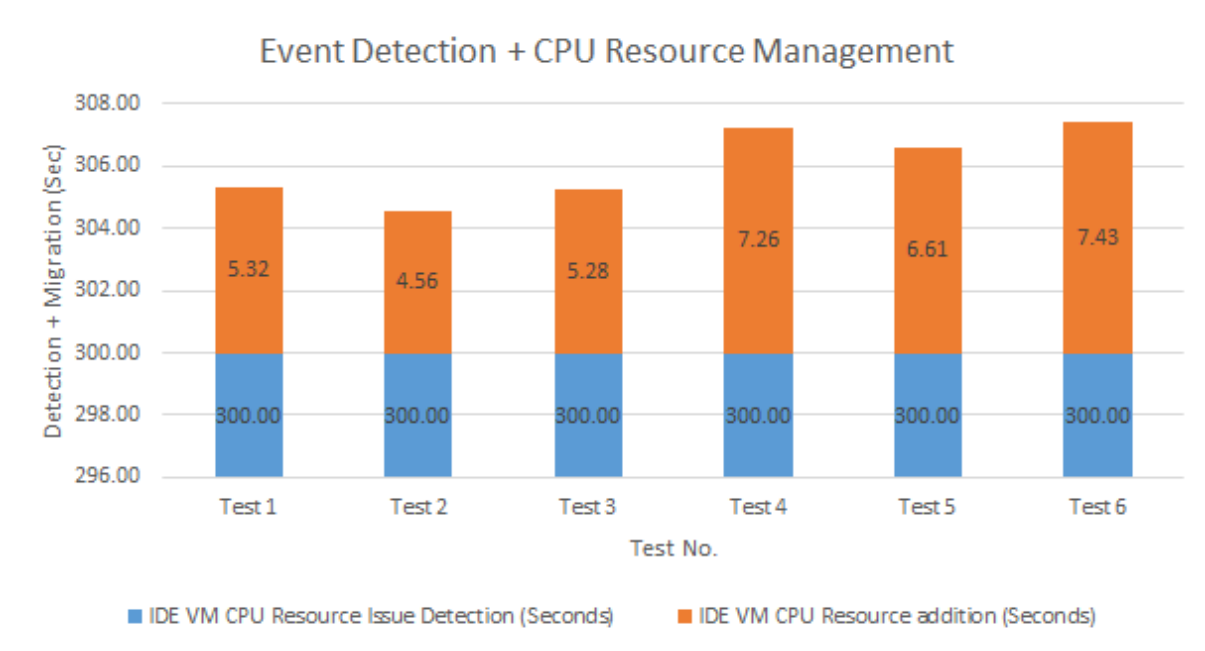


Figure 7.7 IDE Performance Monitoring and CPU Hot-plug Results

7.5.2 Study 1 VM CPU Hotplug Process

Study 1 (Xen Balloon) does not feature a CPU hot-plug process, nor a global CPU resource scheduling system.

7.5.3 Study 2 VM CPU Hotplug Process

Study 2 (iBalloon) does not feature a CPU hot-plug process, nor a global CPU resource scheduling system.

7.6 Results

Based on the characteristics of the features enabled by all the studies listed in section [7.2.4](#) and table [7.2](#), it is possible to perform an evaluation on the results by examining and comparing the overall capabilities for each experiment, which includes the IDE, study 1 (Xen Balloon) and Study 2 (iBalloon). As discussed earlier, the resource management of VMs is a complex matter, and a certain process for handling events is not necessarily something that can be described as “the best”, simply by being the quickest to perform a dynamic memory increase (ballooning) for a VM, which for example, has had a short memory spike up to 90%

for 1 minute, or by adding (hot-plugging) a CPU to a VM which has had its CPU peak at 85% for 20 seconds.

There were several questions raised at the beginning of this chapter which alluded to how, why and what method and approach is the best. The answers are not immediately clear; however, we can evaluate the characteristics, features and scheduling mechanism to determine the overall effectiveness of intelligently managing virtualised resources, in a similar fashion adopted by Rothenberg and his fellow researchers ([Rothenberg et al, 2017](#); [Conrath and Sharma, 1991](#)). Based similarly on these approaches (of expert system evaluation), the table below summarises each of those initial questions and provides a mixture of qualitative and quantitative feedback on the three different approaches to the process for the IDE, study 1 and study 2. Scoring is performed using the following method; for *feature availability*:

- If there is feature is available a score of 3 is allocated.
- If the feature is emerging and partially developed, then it receives a score of 2.
- If the feature has been designed, but not evaluated or experimented against at all, then it receives a score of 1.
- If there is no feature, then a score of 0 is allocated.

For *feature capability*:

- If the feature worked effectively during experimentation, then a score of 3 is allocated.
- If the feature worked with mixed results during experimentation, then a score of 2 is allocated.
- If the feature worked, but fails to deliver any perceived benefits, then a score of 1 is allocated.
- If there is no feature, then a score of 0 is allocated.

7.6.1 IDE Characteristics (VirtualBox Balloon)

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
1 (IDE)	How long a period should a VM be monitored for, before taking intervention?	VM poll interval, and performance result processing.	Yes	5-minute poll interval, data collected each second. Results evaluated by IDE after each 5-minute sample period against knowledge rules as per section 7.2.1 .	3 / 3
1 (IDE)	How often should performance stats be sampled during the poll interval?	IDE Performance Sampling frequency for each VM.	Yes	CPU and memory stats collected every second, as per section 7.2.1	3 / 3
1 (IDE)	Can CPU resource be increased	Ability to hotplug CPUs.	Yes	As per section 7.5.1 .	3 / 3

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	and decreased?				
1 (IDE)	Can memory resource be increased and decreased?	Ability to balloon memory.	Yes	As per section 7.4.1.	3 / 3
1 (IDE)	Can I/O resource be increased and decreased?	Ability to increase or reduce I/O for network or disk.	No	Not available.	0 / 0
1 (IDE)	How well do the ballooning and CPU hotplug features safeguard and protect the Hosts/Guest VMs?	Proactive monitoring and reaction to resource shortage or observed waste events.	Yes	As per section 7.2.1.	2 / 2
1 (IDE)	How	The ability for the	Yes	As per section	2 / 2

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	advanced are the overall platform management features and can the system globally resource manage?	management control system to communicate/issue commands to other hosts under its control.		7.2.1 , 7.4.1 and 7.5.1 . Further testing and experiments can be conducted as per section 8.3.11 .	

Table 7.5 IDE Resource Management Evaluation

7.6.2 Study 1 Characteristics (XenBalloon)

Study reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
Study 1 VM Memory Balloon Process (Zhang et al, 2017)	How long a period should a VM be monitored for, before taking intervention?	VM poll interval, and performance result processing.	Unknown	Implied feature, as per sections 7.2.4 , 7.4.2 and 7.5.2 .	1 / 1
Study 1	How often should be performance stats be sampled during the poll interval?	Performance sampling frequency for each VM.	Unknown	Implied feature, as per sections 7.2.4 , 7.4.2 and 7.5.2 .	1 / 1
Study 1	Can CPU resource be increased and decreased?	Ability to hotplug CPUs.	No	Not available.	0 / 0
Study 1	Can memory resource be increased and	Ability to balloon memory.	Yes	Feature available, as per sections 7.2.4	3 / 3

Study reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	decreased?			and 7.4.2 .	
Study 1	Can I/O resource be increased and decreased?	Ability to increase or reduce I/O for network or disk.	No	Not available.	0 / 0
Study 1	How well do the ballooning and CPU hotplug features safeguard and protect the Hosts/Guest VMs?	Proactive monitoring and reaction to resource shortage or observed waste events.	Yes	Feature available, as per sections 7.2.4 , 7.4.2 and 7.5.2 . Note, the ability is implied as tested against 10 VMs, however, not available for CPU hotplug.	2 / 2
Study 1	How advanced are the overall platform management features and can the system	The ability for the management control system to communicate/ issue commands	Yes	Feature available, as per sections 7.2.4 , 7.4.2 and 7.5.2 . Note, the feature is implied as tested against	2 / 2

Study reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	globally resource manage?	to other hosts under its control.		10 VMs.	

Table 7.6 Study 1 Resource Management Evaluation

7.6.3 Study 2 Characteristics (iBalloon)

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
Study 2 iBalloon system overview (Zhang et al, 2016)	How long a period should a VM be monitored for, before taking intervention?	VM poll interval, and performance result processing.	Yes	Feature present, as per sections 7.2.4 , 7.4.3 and 7.5.3 .	2 / 2
Study 2	How often should be performance stats be sampled	Performance sampling frequency for each VM.	Yes	Varying interval frequency with min/max, as	2 / 2

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	during the poll interval?			per sections 7.2.4 , 7.4.3 and 7.5.3 .	
Study 2	Can CPU resource be increased and decreased?	Ability to hotplug CPUs.	No	Not available.	0 / 0
Study 2	Can memory resource be increased and decreased?	Ability to balloon memory.	Yes	Feature available, as per sections 7.2.4 and 7.4.3 .	3 / 3
Study 2	Can I/O resource be increased and decreased?	Ability to increase or reduce I/O for network or disk.	No	Not available.	0 / 0
Study 2	How well do the ballooning and CPU hotplug features	Proactive monitoring and reaction to resource shortage or observed waste events.	Yes	Feature available, as per sections 7.2.4 , 7.4.3 and 7.5.3 . Note, the	2 / 2

Study Reference	Question	System Characteristic	Feature Available (Yes/No/Unknown)	Feature Availability (Av) and Capability (Cp)	Result Score [Av/Cp]
	safeguard and protect the Hosts/Guest VMs?			ability is implied as tested against 4 VMs. However, not available for CPU hotplug.	
Study 2	How advanced are the overall platform management features and can the system globally resource manage?	The ability for the management control system to communicate/issue commands to other hosts under its control.		Feature available, as per sections 7.2.4 , 7.4.3 and 7.5.3 . Note, the feature is implied as tested against 4 VMs.	2 / 2

Table 7.7 Study 2 Resource Management Evaluation

7.6.4 Platform Characteristic Scores (IDE, Study 1, Study 2)

The following table provides indicative score values (%) for the identified features and characteristics for the IDE, Study 1 (XenBalloon) and Study 2 (iBalloon). The score values are able to reflect the feature availability and capability that the 3 systems have to offer, in terms of 'intelligent management' of virtualised platforms, with the higher value indicating such.

	IDE	Study 1 (XenBalloon)	Study 2 (iBalloon)
Total Characteristic Score (%)	30 out of a possible 42 (71%).	18 out of a possible 42 (43%).	22 out of a possible 42 (52%).

Table 7.8 Overall System Characteristic Scores %

As table 7.8 only provides indicative results, as at this stage, it would be necessary to rebuild the platforms in study 1 and study 2, to perform a detailed re-test and comparison. Section 8.3.9 deals with the opportunities to develop this work further.

7.6.5 Binomial Scores (IDE, Study 1, Study 2)

The following chart figure 7.8 displays the binomial results from table 7.2, which again provides an indication of the capability and feature richness, within each investigated platform.

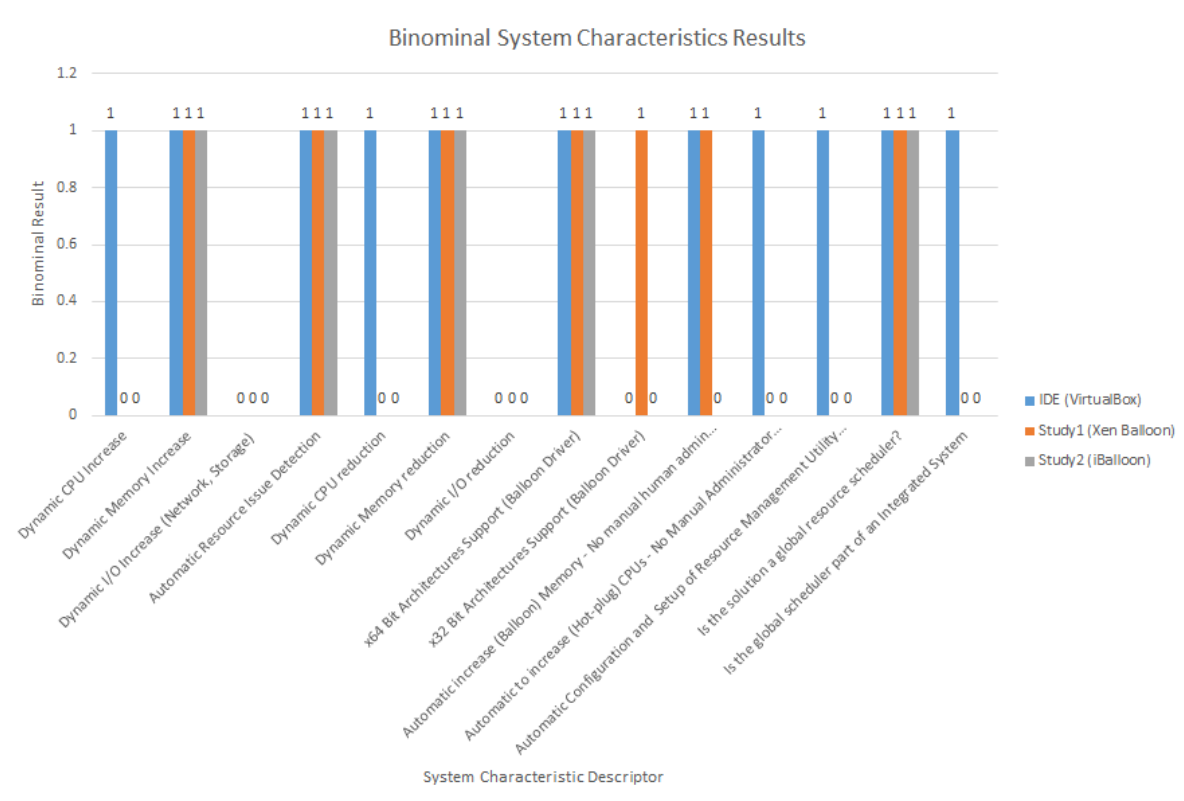


Figure 7.8 Binomial System Characteristic Results

The greater the number of features that work together in combination, the more potential that the system can be regarded as 'being intelligent' (see section [2.6.2](#) on reviewed approaches, and section [4.2](#) for details on the traits for intelligent systems). Note that the conclusions for this chapter can be found in section [8.2.4](#) for memory overload and [8.2.5](#) for CPU overload experiments.

7.7 Summary

The final two experiments covered in this chapter discuss global resource management, in particular around CPU hotplug and memory ballooning features. The IDE utilises its extended performance algorithm to manage CPU and memory resource across its controlled physical hosts and their virtualised system components. This allows for ballooning using the VirtualBox driver to facilitate the over allocation of memory resource to enable system memory to be dynamically increased or reduced as desired, to match the VMs requirement for performance. The system characteristics and features are compared, and additionally include global management capabilities for each comparative platform as highlighted in table [7.2](#). The IDE is again contrast against two alternative similar papers which present their results on their resource management processes. The first study utilises a pure global memory management system using the XenBalloon driver, and the second study uses a custom iBalloon system, which is a control system built on top of the underlying KVM memory balloon driver. An analysis and simple scoring mechanism are used to measure each of the capabilities and features of the system. By using this scoring approach, it is possible to calculate overall results for each system, and determine how effective the overall performance management is for the IDE, study 1 and study 2 respectively. In addition, a simple binomial procedure is used to represent all platform characteristics that are available, thus allowing additional comparisons to be made on the richness and depth of each global resource management system. The last chapter discusses the contribution of the thesis, converges the results for all five experiments undertaken, and provides a suitable conclusion

for each area of investigation. Finally, each potential area for further work is considered with a view to providing an introduction into a new research area, topic or sub-topic.

Chapter 8: Contribution, Conclusions and Further Work

8.1 Thesis Contribution

The following sections summarise the main contribution of this work to the field of virtualised computer management:

8.1.1 Development of an Expert System Framework for Virtualised Computer Systems

The work proposed and developed the use of an expert system (IDE) framework to enhance the management of virtualised computer-based systems, and enable fast real-time decision making within a complex virtualised computer environment, with the purpose of having control of VMs, workloads and other virtualised components. The decision engine controlled several core functions, described by chapters [5](#) (VM provisioning), [6](#) (VM migration/failover) and [7](#) (VM resource management), which were investigated through experimentation. The IDE itself remains open to be developed further, as its functionality can be extended through the development and addition of knowledge rules and their associated automation code routines. The following areas were investigated as part of the IDE framework:

- Remote system discovery mechanism, with system OS fingerprint analysis and advanced OS system type detection; see algorithm/procedure 1, table [4.1](#).
- Improved system communication strategy using SSH to build a secure framework for remote host management and control; see algorithm/procedure 2, table [4.2](#).
- Improved data extraction and analysis approach to enable two methods of 1) quick response and 2) slower background analysis of environment data to allow for reference knowledge information to be added and cleansed; see

algorithm/procedure 3, table [4.3](#).

- Information and knowledge organisation to process and create reference data structures, which affects how the forward chaining mechanisms work when the IDE is decision making; see algorithm/procedure 4, table [4.4](#).
- Improved pattern analysis and learning from data; see algorithm/procedure 5, table [4.5](#).
- Construction of a knowledge based forward chained events algorithm; see algorithm/procedure 6, table [4.6](#).
- Development of an advanced VM deployment/provisioning mechanism; see algorithm/procedure 7, table [4.7](#).
- Creation of a preliminary and extended VM performance and monitoring management mechanism; see algorithm/procedure 8, table [4.8](#) for the preliminary, and table [7.1](#) for the extended.
- Real-time platform event trigger with a decision processing-based delivery event response; see algorithm/procedure 9, table [4.9](#).
- Improved self-monitoring and high availability features; see algorithm/procedure 10, table [4.10](#).

Consult chapter [4](#) 'The intelligent Decision Engine' for further information.

8.1.2 Simplified VM Provisioning

Based on the findings in chapter [5](#), a simplified VM provisioning methodology was provided, along with improved delivery times through automation and intelligent decision-making utilising the IDE processes. This included the ability to deploy VMs using a web browser interface utilising a '1 click' VM deployment mechanism, and the simplification of VM provisioning for end-users through higher levels of automation. This resulted in an overall

end-user VM provision time reduction (i.e. an aggregated user experience build-time reduction for VMs). Consult chapter [5](#) and section [5.2.2](#) ‘Experiment 1: VM Provisioning Timing Comparison’ for more information on the outcomes.

8.1.3 CLR formula to Determine Task Complexity

An algorithm was devised for the analysis of a Cognitive Load Rating (CLR) for human interactions, using a computer system and its interface, such as a VM provisioning mechanism. This provided a method for conversion of qualitative data into quantitative data (i.e. words to numbers); please see section [3.4.1](#) for more information. This method and approach could be used against any type of system, where user survey feedback is acquired and processed. Consult section [5.2.3](#) ‘Experiment 2: Cognitive Evaluation Performance’ for additional information.

8.1.4 Efficient VM Migration, Evacuation and Restart Routines

It was demonstrated how VMs can be migrated and evacuated more effectively using the IDE in a ‘full-restart’ scenario, compared to other studies using alternative technologies such as vMotion and XenMotion. This included improving VM failover patterns utilising the IDE to perform VM relocation as necessary, and faster average VM fault detection and failover processes. Please refer to chapter [6](#) for further in-depth analysis and discussion, along with the details described in section [6.4](#) ‘Experiment 3: Workload Migration and Evacuation of VMs’.

8.1.5 Global Scheduling Mechanism for CPU Hot-plug and Memory Resource Management

Evidence was provided to show how VMs can be protected even more effectively from an overload of CPU and/or memory consumption, when compared to other research papers

on vMotion and XenMotion. This included the faster detection of VM CPU and memory performance issues, dynamic VM CPU and memory resource resizing, and faster VM recovery should full failure events occur. This is described further in section [7.4](#) ‘Experiment 4: Overload of VM Memory Usage, Detection Time, and Resolution Time’ and [7.5](#) ‘Experiment 5: Overload of VM CPU usage, Detection Time, and Resolution Time’.

8.1.6 Summary

The knowledge areas this thesis contributes towards are summarised in the table below:

- | |
|--|
| <ul style="list-style-type: none">• Creation of the Intelligent Decision Engine (IDE). Then the subsequent utilisation of this framework to contribute to the following topics:<ul style="list-style-type: none">i. The simplification of the VM deployment mechanism.ii. The reduction of the CLR for the VM provisioning process.iii. Improvement of the IDE VM migration/failover average time.iv. Enhancement of the IDE global performance and availability management capability. |
|--|

Table 8.1 Thesis Contributions

8.2 Overall Results and Conclusions

The next sections provide details on the conclusions reached, based on the data and results recorded in each of the experiment sections.

8.2.1 Simplified VM Deployment Experiment Conclusions

The results for the conclusions reached are recorded in section [5.3](#). They are focused on the provisioning aspect of VMs, in terms of being able to prove that the IDE could efficiently deliver new VMs in the least amount of time, using the 10-step technique described by section [5.2.1.6](#) ‘VM Provisioning Process’. This was completed anonymously by 3 groups of users classified as, expert, experienced, and novice. Results for each group of 31 users were

recorded in turn, while using each different platform (IDE, AWS and Oracle) and graphed for each of the 10-steps. Each end-user was therefore able to provision a VM in a certain amount of time, summarised as follows:

- For the IDE platform, 'Expert', 'Experienced' and 'Novice' users averaged 1231, 1372 and 1578 seconds respectively to provision a VM.
- For the AWS platform, 'Expert' and 'Experienced' users averaged 1382 and 1464 seconds respectively, to provision a VM. Unfortunately, for 'Novice' users, all but one of the 31 users were *unable to complete* the provisioning process.
- For the Oracle platform, 'Expert' and 'Experienced' users averaged 2362 and 3237 seconds respectively, to provision a VM. Unfortunately, for 'Novice' users, all but one of the 31 users were *unable to complete* the provisioning process.

The reason for the IDE outperforming the other two platforms during provisioning was primarily as a result of the *extra level of automation* for the 10-steps. This is especially true for step 8, which includes the mechanism to copy over the SSH keys to ensure the user can access the VM. As this was automated for the IDE provisioning process, the users did not need to manually perform this step. It is true, that once a step has been automated, it becomes a *simple step irrespective* of its actual complexity, because the code created takes this mental effort away from the end-user. In other words, the complexity is hidden by the automatically executed code, which performs the necessary tasks on behalf of the user. Again, the results backup the fact that for each platform there were the following step mechanisms listed:

- For the IDE provisioning platform, there were 7 *automatic*, 2 *semi-automatic* and 1 *manual* step recorded.
- For the AWS provisioning platform, there were 3 *automatic* steps, 5 *semi-automatic* and 2 *manual* steps recorded.
- For the Oracle provisioning platform, there were 1 *automatic*, 7 *semi-automatic* and 2 *manual* steps recorded.

Therefore, it can be concluded from the results, that as the number of steps that are automated increases, the more fast, efficient, consistent, and reliable the VM provisioning process is. For example, even when using the AWS platform and AMIs for the provisioning process, time is lost, by having steps which require human administrative intervention. End to end automation eradicates these negative aspects, and results in an overall reduction in VM delivery time.

8.2.2 Cognitive Evaluation Performance Experiment Conclusions

In addition to the VM provisioning experiment, it was possible to extract some qualitative feedback using a process to convert 'words to numbers' as previously discussed in sections [3.4.1](#) and [5.2.3.1](#). This data provided an alternative set of results presented in section [5.3.3](#) and were intended to provide a complimentary viewpoint. For each of the 3 user groups, feedback was provided based on the 'cognitive load' experience for each end-user, described in section [5.2.3](#). For the 'expert', 'experienced' and 'novice' groups, we have the overall following conclusions, based on section [5.2.3.5](#) the 'Cognitive Load Rating' chart:

- 'Expert', 'Experienced' and 'Novice' users using the IDE platform had an average CLR result of 6.77, 5.38, and 7.89, which according to the CLR chart guide indicates they found the cognitive load to have a mental power requirement of 'Low', 'Very-Low' and 'Low' respectively.
- 'Expert', 'Experienced' and 'Novice' users using the AWS platform had an average CLR result of 13.3, 13.72 and 17.69, which according to the CLR chart guide indicates they found the cognitive load to have a mental power requirement of 'Low-Medium', 'Low-Medium' and 'Medium' respectively.
- 'Expert', 'Experienced' and 'Novice' users using the Oracle platform had an average CLR result of 20.42, 22.46 and 25.45, which according to the CLR chart guide indicates they found the cognitive load to have a mental power

requirement of 'Medium', 'Medium-High' and 'Medium-High' respectively.

8.2.3 Workload Migration/Failover Experiment Conclusions

As part of the IDE's control mechanism over potentially large numbers it was crucial to be able to test how effective it is as managing the migration, failover or evacuation of VMs under certain conditions, such as a physical host failure. Sections [6.4](#), [6.4.1](#), [6.4.2](#) and [6.4.3](#) examine the experiment process, and section [6.5](#) confirms the results:

- The IDE was able to achieve an average migration/failover time for a VM in a time of 22.48 seconds, with a best time of 21.56 seconds.
- Study 1 (vMotion) was able to achieve an average migration/failover time for a VM in a time of 44.83 seconds, with a best time of 30 seconds.
- Study 2 (vMotion) was able to achieve an average migration/failover time for a VM in a time of 63.33 seconds, with a best time of 20 seconds.
- Study 2 (XenMotion) was able to achieve an average migration/failover time for a VM in a time of 266.67 seconds, with a best time of 80 seconds.

From the findings, we can observe that the IDE had the best average migration/failover time, but not the best individual time, which was for a study 2 (vMotion) failover experiment, where the network bandwidth peaked at 1Gb/s. Based on this, further work can be completed to try to improve the IDE, using the 'teleport' feature as described in section [8.3.3](#).

8.2.4 Performance and Availability (CPU & Memory Overload) Experiment Conclusions

The Performance and availability experiments described in section [7.2](#) provide a view

into the resource management capability for the IDE platform, study 1 (XenBalloon) and study 2 (iBalloon). The results are presented in section [7.6](#), and are focused around the capability, features and richness of the functionality offered by each respective system, rather than the speed to complete a particular task, such as increasing (ballooning) the memory in a VM. The indicative results and findings are presented below, based on tables [7.5](#), [7.6](#) and [7.7](#). Note, the higher percentage indicates a better result, and the possible 42 is calculated as 7 primary characteristic areas, each with a potential score of 6:

- The IDE had a ‘feature availability and capability’ score of 30 out of a possible 42 (71%).
- Study 1 (XenBalloon) had a ‘feature availability and capability’ score of 18 out of a possible 42 (43%).
- Study 2 (iBalloon) has a ‘feature availability and capability’ score of 22 out of a possible 42 (52%).

Further to this, a binomial evaluation based in figure [7.8](#) containing detailed features/characteristics, which are summarised as below:

- The IDE had a binomial ‘characteristic’ score of 11 out of a possible 14 (79%).
- Study 1 (XenBalloon) had a binomial ‘characteristic’ score of 7 out of a possible 14 (50%).
- Study 2 (iBalloon) has a binomial ‘characteristic’ score of 5 out of a possible 14 (36%).

8.2.5 Significance of Results

The following section takes the results obtained and shows the significance of the IDE versus the alternative systems involved in the experimentation process; namely, the AWS and

Oracle platforms for experiment 1 and 2, then the alternative study papers compared against for experiments 3, 4 and 5. Firstly, the results are shown in the following tables for experiment 1 – VM provisioning time:

End-User group	IDE (sec) / AWS (sec) Provisioning time	Platform, Percentage (%) Faster Provisioning
Expert	1231 / 1382	IDE / 12.27% faster provisioning time
Experienced	1372 / 1464	IDE / 6.71% faster provisioning time
Novice	1578 / N/A*	IDE / unable to present comparative data*

Table 8.2 IDE versus AWS VM Provisioning Time

* Novice users in the experiment failed to complete the VM provisioning process.

End-User group	IDE (sec) / Oracle (sec)	Platform / Percentage (%) Faster Provisioning
Expert	1231 / 2362	IDE / 91.88% faster provisioning time
Experienced	1372 / 3237	IDE / 135.93% faster provisioning time
Novice	1578 / N/A*	IDE / unable to present comparative data*

Table 8.3 IDE versus Oracle VM Provisioning Time

* Novice users in the experiment failed to complete the VM provisioning process.

Secondly, the following tables for experiment 2 VM provisioning are shown, which highlight the improvement in the CLR for the IDE platform (see section [5.2.3.5](#) for the CLR guide chart):

End-User group	IDE (CLR) / AWS (CLR)	Platform / Percentage (%) Improved CLR
Expert	6.77 / 13.30	IDE / 96.45% improved CLR
Experienced	5.38 / 13.72	IDE / 155.02% improved CLR
Novice	7.89 / 17.69	IDE / 124.21% improved CLR

Table 8.4 IDE versus AWS CLR

End-User group	IDE (CLR) / Oracle (CLR)	Platform / Percentage (%) Improved CLR
Expert	6.77 / 20.42	IDE / 201.62% improved CLR
Experienced	5.38 / 22.46	IDE / 317.47% improved CLR
Novice	7.89 / 25.45	IDE / 222.56% improved CLR

Table 8.5 IDE versus Oracle CLR

Thirdly, the following tables for experiment 3 – VM Failover/migration between physical host timings are shown:

IDE Average Failover/Migration Time (Sec)	Paper 1 (vMotion) Mean Average Failover/Migration Time (Sec)	Platform / Percentage (%) Improved for Mean Average Failover/Migration Time
22.48	44.83	IDE / 99.42% improved Failover/Migration time

Table 8.6 IDE v Paper1 (vMotion) Avg. (Mean)Failover/Migration Time

IDE Best Failover/Migration Time (Sec)	Paper 1 (vMotion) Best Failover/Migration Time (Sec)	Platform / Percentage (%) Improved Best Failover/Migration Time
21.56	30.00	IDE / 39.15% improved Failover/Migration time

Table 8.7 IDE v Paper1 (vMotion) Best Failover/Migration Time

IDE Average Failover/Migration Time (Sec)	Paper 2 (vMotion) Mean Average Failover/Migration Time (Sec)	Platform / Percentage (%) Improved for Mean Average Failover/Migration Time
22.48	63.33	IDE, 181.72% improved Failover/Migration time
IDE Average Failover/Migration Time (Sec)	Paper 2 (XenMotion) Mean Average Failover/Migration Time (Sec)	Platform / Percentage (%) Improved for Mean Average Failover/Migration Time
22.48	266.67	IDE, 1086.25% improved Failover/Migration time

Table 8.8 IDE v Paper2 (vMotion, XenMotion) Avg. (Mean) Failover/Migration Time

IDE Best Failover/Migration Time (Sec)	Paper 2 (vMotion) Best Failover/Migration Time (Sec)	Platform / Percentage (%) Improved Best Failover/Migration Time
21.56	20.00	vMotion, 7.8% improved Failover/Migration time
IDE Best Failover/Migration Time (Sec)	Paper 2 (XenMotion) Best Failover/Migration Time (Sec)	Platform / Percentage (%) Improved Best Failover/Migration Time
21.56	80.00	IDE, 271.06% improved Failover/Migration time

Table 8.9 IDE v Paper2 (vMotion, XenMotion) Best Failover/Migration Time

Experiments 4 and 5 are related to the global performance management of the platforms, in relation to resource controls over consumables such as CPU and memory. The results are presented for the characteristics and feature richness (based on tables [7.5](#), [7.6](#) and [7.7](#)), and additionally, for the binomial system analysis (see figure [7.8](#)):

Platform	Percentage (%) Feature Availability and Capability Score (Note: A higher % indicates a stronger capability)
IDE	71%
Study1 (XenBalloon)	43%
Study2 (iBalloon)	53%

Table 8.10 Platform Features, Availability and Capability Scores

Platform	Percentage (%) Binomial Characteristic Assessment Score (Note: A higher % indicates a stronger platform characteristic richness)
IDE	79%
Study1 (XenBalloon)	50%
Study2 (iBalloon)	36%

Table 8.11 Platform Binomial Characteristic Assessment Scores

8.3 Future Work

Following the work conducted as part of this research project, there is opportunity for a considerable amount of future work to continue, to build on the work completed so far; some of the areas identified are as follows:

8.3.1 Prebuilding and Queuing VMs

To develop and add in a prebuilt VM build for each OS type, which is queued and

waiting for deployment. It would be feasible to build many VMs for each OS type and have a queuing pipeline system in place; section [5.2.2.6](#) developed this idea initially, however, it is beyond the scope of this thesis to fully develop this to fruition. It is anticipated, pending further experimentation and result outcomes, that this would reduce VM provisioning time even further.

8.3.2 Development with Additional Operating Systems

The majority of the development was completed against the Linux CentOS operating system; therefore, further work and experiments are required against other operating system types, including Windows, AIX and Solaris.

8.3.3 VirtualBox Teleport Development

Memory VM replication and migration/failover so far has concentrated on VM ‘failed state’ and ‘full restart’ scenarios – see section [6.2](#) and [6.3](#) for more details. Further work is needed to utilise the VirtualBox ‘teleport’ function to develop advanced ‘live migration’ techniques further for the IDE ([VirtualBox, 2019](#)).

8.3.4 Quorum Cluster Node Testing

The IDE was developed and tested using 3-node clusters. It is desirable to test a larger cluster node configuration > 3 IDE nodes, as per section [4.7](#).

8.3.5 Bootstrap Development

It would be beneficial for the IDE to be able to self-replicate its core functions. Each IDE cluster should be able to create (generate) another. Further to section [3.2](#), the idea would be to increase the IDE to use larger scale systems, by additional testing of the integrated/engineered hardware components; for example, being able to deploy repeatable IDE ‘building blocks’, comprising of the same CPU, memory and storage stack. The goal would be for the re-creation of the IDE from a standard bare metal hardware configuration (e.g. a

boot-strap type function with access to a global code repository source - always on).

8.3.6 Knowledge Rules

There is opportunity to increase the number of knowledge rules for additional expert system functionality, e.g. filesystem capacity, or storage devices that are at full capacity (or nearing full) are good examples that could be considered. Additionally, backward chain reasoning could be considered, whereby the system works to achieve a set of goals; this could be very useful for proactive type initiatives, such as reducing the number of known security vulnerabilities a system has.

8.3.7 Self-Learning

Self-development of knowledge rules – internal introspection and a self-learning function could be added to create new rules. This would include understanding its learning requirements, developing its learning goals and how to achieve them, identifying the resources needed to support the learning process, and evaluating the outcomes. A validation and scoring system could also work to rank each knowledge rule, to ensure they are functioning as purposed.

8.3.8 Data Sources and Trigger Events

Development and testing of additional data sources for trigger events to those defined in sections [4.8.1](#) and [4.8.4](#). Additional exploration into the virtual platform, to determine what other data sources could be useful, as well as the identification of new trigger events.

8.3.9 Laboratory Build for VMware, KVM and Xen Clusters

A VMWare, KVM and Xen cluster build using the same hardware as the IDE stack, to enable direct experimental ‘failover testing’ and ‘resource management’ testing with real

VMWare, KVM and Xen systems, to allow for better comparisons against each in a laboratory setting. This would build additional data on top of the similar studies examined in [chapter 6](#) and [chapter 7](#).

8.3.10 Knowledge Rule Testing with SLAs

Further experimental testing of the IDE knowledge rules listed in section [4.8](#) and [4.8.5](#). Extra validation and checks to be conducted on the existing rule set with a Service Level Agreement (SLA) in place; to be investigated as to how this would impact the consequent(s) for invoked knowledge rules.

8.3.11 Global Resource Management

Further global platform management as discussed in section [7.2.5](#) is required to ensure other areas are built on, including the continual reduction of resource waste. Investigation to continue on how to make the global resource management process even more effective and efficient.

8.3.12 Terraform, AWS CloudFormation and AMIs

Perform an analysis on the IDE against Terraform and AWS CloudFormation with AMIs (A common DevOps AWS approach) in a laboratory type exercise. This work would produce interesting results, as AWS CloudFormation and Terraform provide 'infrastructure as code' provisioning modules, which would first need to be developed using notation such as YAML or JSON.

References

- Adl-Tabatabai, A., Bharadwaj, J., Cierniak, M., Eng, M., Fang, J., Lewis, B., Murphy, B. and Stichnoth J. (2004). *Improving 64-Bit Java IPF Performance by Compressing Heap References*, Proceedings of the International Symposium on Code Generation and Optimization
- Aladyshev, O., Baranov, A., Ionin, R., Kiselev, E., and Shabanov, B. (2018). *Variants of Deployment the High-Performance Computing in Clouds*, IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)
- Al-Ou'n, A., Kiran, M., and Kouvatsos, D. (2015). *Using Agent-based VM Placement Policy*, IEEE 3rd International Conference on Future Internet of Things and Cloud
- Alty, J. L., and Coombs, M. J., (1984). *Expert systems: concepts and examples*, John Wiley and Sons, Inc., New York, NY
- Ajila, S. and Bankole, A., (2013). *Cloud Client Prediction Models Using Machine Learning Techniques*. IEEE 37th Annual Computer Software and Applications Conference
- Akioka, S. and Muraoka, Y., (2010). *HPC benchmarks on Amazon EC2*. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops
- Anicic, D., Fodor, P., Stuhmer R., and Stojanovic N., (2009). *Event-driven Approach for Logic-based Complex Event Processing*, International Conference on Computational Science and Engineering
- Antonescu, A., Oprescu, A., Demchenko, Y., Laat C. D., Braun T., (2013). *Dynamic Optimization of SLA-Based Services Scaling Rules*, IEEE International Conference on Cloud Computing Technology and Science
- Amazon Web Services, (2015). *Amazon Elastic Compute Cloud - User Guide for Linux API Version*. Amazon Web Services
- Arnaldo, I., Veeramachaneni, K., Song, A. and O'Reilly, U., (2015). *Bring Your Own Learner! A Cloud-Based, Data-Parallel Commons for Machine Learning*. IEEE Computational intelligence magazine
- Ashouri K., and Savoji, M.H, (2004). *Automatic and Accurate Pitch Marking of Speech Signal using an Expert System Based on Logical Combinations of Different Algorithms Outputs*, 12th European Signal Processing Conference
- Austermann A., and Yamada, S. (2008). *"Good Robot", "Bad Robot" – Analyzing Users' Feedback in a Human-Robot Teaching Task*, Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, Technische Universität München, Munich, Germany, August 1-3
- Awal, A., Shoeb, M., Hasan, R., Haque, M. and Hu, M., (2014). *A Comparative Study on I/O Performance between Compute and Storage Optimized Instances of Amazon EC2*. IEEE International Conference on Cloud Computing
- Bakhshayeshi, R., (2014). *Performance Analysis of Virtualized Environments using HPC Challenge Benchmark Suite and Analytical*, Iranian Conference on Intelligent Systems (ICIS)
- Beckman, T., J. (1990). *Methods for Selecting Promising Expert System Applications*, Proceedings, The Fifth Annual AI Systems in Government Conference, IEEE

Benet C. H., Noghani K. A., and Kassler, A. J. (2016). *Minimizing Live VM Migration Downtime Using OpenFlow based Resiliency Mechanisms*, 5th IEEE International Conference on Cloud Networking

Bhise, V. and Mali, A., (2013). *Cloud Resource Provisioning for Amazon EC2*. IEEE - 31661, 4th ICCNT July 4-6, Tiruchengode, India

Biner M. (2015). *Cloud Computing and Management Processes*, DOI: 10.1109/ECAI.2015.7301151, ECAI, Bucharest

Bojanova, I. and Samba, A., (2011). *Analysis of Cloud Computing Delivery Architecture Models*. Workshops of International Conference on Advanced Information Networking and Applications

Borg, G., Bratfisch, O., and Dornic, S. (1971). *On the problems of perceived difficulty*. *Scandinavian Journal of Psychology*, 12(4), 249–260

Brooks R. E. and Heiser, J., F., (1979). *Transferability of a Rule-Based Control Structure to a New Knowledge Domain*, AMIA Annual Symposium Proceedings

Callaos, B., (1994). *Artificial Organizational Intelligence. Expert Systems for Development*, Proceedings of International Conference of The World Congress on Expert Systems

Calzolari, F., (2006). *High Availability Using Virtualisation*, University of Pisa

Cambridge Advanced Learner's Dictionary (2019), [online]. Available from: <https://dictionary.cambridge.org/dictionary/english/alqorithm> Cambridge: Cambridge University Press [06/07/2019]

Cambridge Advanced Learner's Dictionary (2019), [online]. Available from: <https://dictionary.cambridge.org/dictionary/english/inference> Cambridge: Cambridge University Press [06/07/2019]

Cattell, R. (2010). *Scalable SQL and NoSQL Data Stores*, ACM SIGMOD Record archive Volume 39 Issue 4, December, Pages 12-27

Chen, X., Chen, W., Long, P., Lu, Z., and Wang Z. (2013) *SEMMA: Secure Efficient Memory Management Approach in Virtual Environment*, International Conference on Advanced Cloud and Big Data

Chen, Z. and Suen, C.Y., (1993). *Evaluating Expert Systems by Formal Metrics*. Proceedings of Canadian Conference on Electrical and Computer Engineering

Conde, C. and Narin, A., (2012). *Development and Test on Amazon Web Services*. Amazon Web Services

Conrath D. and Sharma, R. (1991). *Evaluating Expert Systems Using A Multiple-Criteria, Multiple-Stakeholder Approach*, Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs

Crittenden, R., (1990). *Building on success-lessons learned (expert systems)*. Proceedings IEEE Conference on Managing Expert System Programs and Projects

Dhiman, G., (2011). *Dynamic Workload Characterization for Energy Efficient Computing*, University of California, San Diego

Diao, L., Zuo, M. and Liu, Q., (2009). *The Artificial Intelligence in Personal Knowledge Management*. Second International Symposium on Knowledge Acquisition and Modeling

Dong D., and Herbert J., (2013). A Proactive Cloud Management Architecture for Private Clouds, IEEE Sixth International Conference on Cloud Computing

Duda, R. O., and Shortliffe, E., (1983). *Expert System Research*. Science (New York, N.Y.). 220. 261-8. 10.1126/science.6340198.

Durkin, J., (1990). *Research Review: Application of Expert Systems in the Sciences*, The Ohio Journal of Science, v90, n5, 171-179

Elprince, N., (2013). *Autonomous Resource Provision in Virtual Data Centers*, 2013 IFIP/IEEE International Symposium on Integrated Network Management

Fateman R. J., (1989). *A Review of Macsyma*, IEEE Transactions on Knowledge and Data Engineering, Vol. I, No. I, March 1989

Fadel, A. S., Fayoumi, A. G., (2013). *14th ACIS Cloud Resource Provisioning and Bursting Approaches*. International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing

Faulkner, L. (2003). *Beyond the five-user assumption: Benefits of increased sample sizes in usability testing*. Behavior Research Methods, Instruments and Computers, 35(3), 379-383.

Feigenbaum E. A., and Buchanan B. G, (1994). *DENDRAL and Meta-DENDRAL: roots of knowledge systems and expert system applications*. Artificial intelligence in perspective, Pages 233-240, MIT Press Cambridge, MA, USA

Feinberg, S. and Murphy, M. (2000). *Applying Cognitive Load Theory to the Design of Web-Based Instruction*, 18th Annual Conference on Computer Documentation Technology and Teamwork Proceedings

Feng, X., Tang, J., Luo, X., and Jin, Y. (2011) A Performance Study of Live VM migration Technologies: vMotion vs XenMotion, SPIE-OSA-IEEE/Vol. 8310 831018-2, Asia Communications and Photonics

Ferraris, F., Franceschelli, D., Gioiosa, M., Lucia, D., Ardagna, D., Di Nitto, E. and Sharif, T., (2012). *Evaluating the Auto Scaling Performance of Flexiscale and Amazon EC2 Clouds*. 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

Fernando, D., Bagdi, H., Hu, Y., Yang, P., Gopalan, K., Kamhoua, C., and Kwiat, K. (2016). *Quick Eviction of Virtual Machines Through Proactive Live Snapshots*, IEEE/ACM 9th International Conference on Utility and Cloud Computing

Finkle, T. A., and Scoresby, R. B., (2012). *Larry Ellison and Oracle Corporation*, Journal of the International Academy for Case Studies, Volume 18, Number 7

Flinta, C. Johnsson, A., Ahmed, J., Moradi, F., Pasquini, R., and Stadler, R. (2017). *Real-Time Resource Prediction Engine for Cloud Management*, IFIP/IEEE International Symposium on Integrated Network Management

Franzosi, R. (2004). *From Words to Numbers: Narrative, data, and social science*, Cambridge University Press

Forbes (2018), [online]. Available from: <https://www.forbes.com/sites/louiscolombus/2018/09/23/roundup-of-cloud-computing-forecasts-and-market-estimates-2018/#29474389507b> Forbes: [1/04/2020]

- Gandhe, A., Qin, L., Metze, F., Rudnicky, A., Lane I., and Eck, M. (2013) *Using Web Text to Improve Keyword Spotting in Speech*, 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, IEEE Workshop on Automatic Speech Recognition and Understanding
- Gazis, V., (2016). *A Survey of Standards for Machine-to-Machine and the Internet of Things*, IEEE Communications Surveys & Tutorials
- Gaikwad G., Joshi D. J., (2016). *Multiclass Mood Classification on Twitter using Lexicon Dictionary and Machine Learning Algorithms*, International Conference on Inventive Computation Technologies (ICICT)
- Gopher, D. and Braune, R. (1984). *On the Psychophysics of Workload: Why Bother with Subjective Measures?* Human Factors: The Journal of the Human Factors and Ergonomics Society, Volume: 26 issue: 5, pages: 519-532
- Green E. C., (2001). *Can Qualitative Research Produce Reliable Quantitative Findings?* Field Methods, Vol. 13, No. 1, February 2001 3–19, Sage Publications, Inc.
- Gren L., Torkar R., Feldt R. (2014). *Work Motivational Challenges Regarding the Interface Between Agile Teams and a Non-Agile Surrounding Organization: A Case Study*. 978-0-7695-5222-4/14 IEEE DOI 10.1109/AGILE.2014.13
- Guerlain, S., Brown, D. and Mastrangelo, C. (2000). *Intelligent Decision Support Systems*, IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions
- Hataba, M. and El-Mahdy, A. (2012). *Cloud Protection by Obfuscation: Techniques and Metrics*, Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing
- Haugeland, J., (1989). *Artificial Intelligence: The very Idea*. pp. 124, MIT Press, Cambridge, MA
- Hill, Z. and Humphrey, M., (2009). *A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?* Grid Computing, 10th IEEE/ACM International Conference on Grid Computing
- Hwang J., (2015). *Computing Resource Transformation, Consolidation, and Decomposition in Hybrid Clouds*, IBM T.J. Watson Research Center, 978-3-901882-77-7 IFIP
- Hwang, J., (2016). *Toward Beneficial Transformation of Enterprise Workloads to Hybrid Clouds*. IEEE Transactions on Network and Service Management, Vol. 13, No. 2, June 2016
- Hwang, W., Roh, Y., Park, Y., Park, K., and Park K. H. (2010). *HyperDealer: Reference-pattern-aware Instant Memory Balancing for Consolidated Virtual Machines*, IEEE 3rd International Conference on Cloud Computing
- Huang, L., Milne, D., Frank, E., Witten, I. H., (2012). *Learning a Concept-based Document Similarity Measure*, Journal of the American Society for Information Science and Technology banner, Volume 63, Issue8, August 2012, Pages 1593-1608
- Imai, S., Chestna, T. and Varela, C., (2013). *Accurate Resource Prediction for Hybrid IaaS Clouds Using Workload-Tailored Elastic Compute Units*. IEEE/ACM 6th International Conference on Utility and Cloud Computing
- Ismail H. and Riasetiawan M. (2016). *CPU and Memory Performance Analysis on Dynamic and Dedicated Resource Allocation using XenServer in Data Center Environment*, 2nd International Conference on Science and Technology-Computer (ICST), Yogyakarta, Indonesia

- Jeong, H. and Lee, S. (2012). *Dynamic CPU Resource Allocation for Multicore CE Devices Running Multiple Operating Systems*, IEEE International Conference on Consumer Electronics (ICCE)
- Jin, H., Kai, Z., Zhijun, W., and Jinzhou, Y., (2016). *PaaS Construction of Large Scale Enterprise*. IEEE International Conference on Cloud Computing and Big Data Analysis Discussion on Private Cloud
- Jing, X., (2011). *Autonomic application and resource management in virtualized Distributed Computing Systems*, University of Florida
- Jing-xue Lui, J. and Fei, Q., (2005). *The Arithmetic Research of Intelligence Retrieval Based on Commanding Decision-Making*. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August
- Joy, M., Mueller, W., and Rammig, F. (2014). *Source Code Annotated Memory Leak Detection for Soft Real Time Embedded Systems with Resource Constraints*, IEEE 12th International Conference on Dependable, Autonomic and Secure Computing
- Katz, J., Papadopoulos, P. and Bruno, G., (2002). *Leveraging Standard Core Technologies to Pragmatically Build Linux Cluster Appliances*, Proceeding of the IEEE International Conference on Cluster Computer
- Kotova, E., (2016). *Intellectual Support of the Learning Content Planning Considering the Cognitive Load*, XIX IEEE International Conference on Soft Computing and Measurements.
- Kim, H., El-Khamra, Y., Rodero, I., Jha, S. and Parashar, M., (2011). *Autonomic Management of Application Workflows on Hybrid Computing Infrastructure*. Scientific Programming 19, pg. 75–89, IOS Press
- Kokkinos, P., Varvarigou, T., kretsis, A., Soumplis, P. and Varvarigos, E., (2013). *Cost and Utilization Optimization of Amazon EC2 instances*. IEEE Sixth International Conference on Cloud Computing
- Kulikowski, C., A., (1980). *Artificial Intelligence Methods and Systems for Medical Consultation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. Pami-2, No. 5
- Kwon K., (2012). *Forward Reasoning via Sequential Queries in Logic Programming*, ISSN 1392–124X Information Technology and Control, Vol. 41
- Lakshmi, J., (2010). *System Virtualization in the Multi-core Era - a QoS Perspective*, Supercomputer Education and Research Center Indian Institute of Science
- Lokshina, I. and Insinga, R. (2004), *Expert System Supporting System Administrators Managing in a Distributed, Heterogeneous Environment*, Joint IST Workshop on Mobile Future and the Symposium on Trends in Communications
- Larumbe, F., Sanso B., (2012). *Optimal Location of Data Centers and Software Components in Cloud Computing Network Design*, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing
- Lindsay R. K., Buchanan B. G., Feigenbaum, E. A., Lederberg, J. (1993) *DENDRAL: A Case Study of the First Expert System for Scientific Hypothesis Formation*, Artificial Intelligence, Volume 61, Issue 2, June 1993, Pages 209-261
- Lebowitz, M., (1983). *Generalization from Natural Language Text*, *Cognitive Science*, Volume 7, Issue 1 January 1983, Pages 1–40

- Liu, H., Jin, H., Liao, X., Deng, W., He, B., and Xu, C. (2015) IEEE Hotplug or Ballooning: A Comparative Study on Dynamic Memory Management Techniques for Virtual Machines, IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 5
- Lui, X., Zeng, S., Guo J. and Zhou, G. (2017). *Human Workload Monitoring Method Considering Qualitative and Quantitative Data Fusion*, Second International Conference on Reliability Systems Engineering, IEEE
- Macefield, R. (2009). *How to Specify the Participant Group Size for Usability Studies: A Practitioner's Guide*, Journal of Usability Studies, Vol. 5, Issue 1, November 2009, pp. 34-45
- Madarasz, L., Lazar, T., Gaspar V., and Andoga, R. (2014). *Perspectives in Evaluating Quality of Complex Technical Systems*, IEEE International Symposium on Intelligent Control (ISIC), IEEE Multi-conference on Systems and Control, October 8-10. Antibes, France
- Makridis, E., Deliparaschos, K., Kalyvianakiy, E. and Charalambous, T. (2017). *Dynamic CPU Resource Provisioning in Virtualized Servers using Maximum Correntropy Criterion Kalman Filters*, 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)
- Martin, T., Azvine, B. and Shen, Y., (2007). *Computational Intelligence Support for Smart Queries and Adaptive Data*. IEEE Symposium on Computational Intelligence in Security and Defense Applications
- Massimiliano, P. D. and Tamburri, D. A. (2017). *Combining Quantitative and Qualitative Studies in Empirical Software Engineering Research*, IEEE/ACM 39th IEEE International Conference on Software Engineering Companion
- Matthias, K., (2008). *Towards autonomic management in system administration*, University of Oslo Department of Informatics
- McCammom R. B., (1989). *Prospector II Expert System*, Prospector II U.S. Geological Survey, VA 22092
- McCorduck, P., Minsky, M., Selfridge, O.G., Beranek, B. and Simon, H.A., (1977). *History of Artificial Intelligence*. International Joint Conference on Artificial Intelligence, pp. 951-952, 953
- McDermott, J., (1982). *Artificial Intelligence*, R1: A Rule-Based Configure of Computer Systems, Volume 19, Issue 1, September 1982, Pages 39-88, Elsevier
- Melekhova, A., (2013). *Machine Learning in Virtualization: Estimate A Virtual Machine's Working Set Size*, IEEE Sixth International Conference on Cloud Computing
- Mei, L. and Cheng, F., (2010). *The Use of Artificial Intelligence in the Information Retrieval System Epoch-making Changes in Information Retrieval System*. Information Management and Engineering (ICIME), The 2nd IEEE International Conference
- Menasce, D. and Bennani, M., (2006). International Conference on Autonomic and Autonomous Systems ICAS06, Volume: 00, Issue: C, IEEE
- Mettrey, W. (1991). A Comparative Evaluation of Expert System Tools, Computer, Volume: 24, Issue: 2
- Miller R. A., Pople H. E., Myers J. D., (1982). *Internist-1, An Experimental Computer Based Diagnostic Consultant for General Internal Medicine*. New England Journal of Medicine, 307(8), 468-476

- Mohammad, I., and Ramananjanyulu K. (2012). *FPGA Implementation of a 64-Bit RISC Processor Using VHDL*, *International Journal of Engineering Research and Applications (IJERA)*, Vol. 2, Issue 3, May-Jun 2012, pp.2544-2549
- Morabito, R., Kjällman, J., and Komu, M., (2015). *Hypervisors vs. Lightweight Virtualization: A Performance Comparison*, 2015 IEEE International Conference on Cloud Engineering
- Mülayim, N. and Alaybeyoğlu, A., (2016). Designing of an expert system based on firefly algorithm for diagnosis of Heart Disease, 20th National Biomedical Engineering Meeting (BIYOMUT), 1-4
- Musen M.A., Shahar Y., Shortliffe E.H. (2006) *Clinical Decision-Support Systems*. In: Shortliffe E.H., Cimino J.J. (eds) *Biomedical Informatics*. Health Informatics. Springer, New York, NY
- Nath, A., Das, S. and Chakrabarti, A., (2010). *Data Hiding and Retrieval*. International Conference on Computational Intelligence and Communication Networks
- Novaliendry P. D., Yang C., Labukti, A.D.G., (2015). *The Expert System Application for Diagnosing Human Vitamin Deficiency Through Forward Chaining Method*, International Conference on Information and Communication Technology Convergence (ICTC)
- Oakes, J., Johnson, M., Xue, J. and Turner, S., (2016). *Simplified Deployment of Virtual Machines using an Intelligent Design Engine*. SAI Computing Conference 2016 July 13-15, London, UK
- Oakes, J., Johnson, M., Xue, J., and Turner, S. (2020) *Measuring and Reducing the Cognitive Load for the End Users of Complex Systems*. In: Bi Y., Bhatia R., Kapoor S. (eds) *Intelligent Systems and Applications*. IntelliSys 2019. *Advances in Intelligent Systems and Computing*, vol 1037. Springer.
- Oludele, A., Ogu E., C., Shade, K., Chinecherem, U., (2014). *On the evolution of virtualization and Cloud Computing: A review*. *Journal of Computer Sciences and Applications*, Volume 2, Issue 3, Pages 40-43
- Padala, P., (2010). *Automated Management of Virtualized Data Centers*, University of Michigan
- Pagare, J. and Koli, N., (2014). *A technical review on comparison of Xen and KVM hypervisors: An analysis of virtualization technologies*. *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 3, Issue 12, December 2014
- Prangchumpol, D., Sanguansintukul, S. and Tantasanawong, P. (2009). *Server Virtualization by User Behaviour Model using a Data Mining Technique – A Preliminary Study*. International Conference for Internet Technology and Secured Transactions: ICITST
- Parunak H., V., D., (1996). *“Go to the Ant”: Engineering Principles from Natural Multi-Agent Systems*, *Annals of Operations Research*, Special Issue on Artificial Intelligence and Management Science
- Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., and Buyya, R. (2015). *Efficient Virtual Machine Sizing for Hosting Containers as a Service*, 2015 IEEE World Congress on Services
- Paas, F., Merrienboer, J., (1994). *Measurement of Cognitive Load in Instructional Research*, *Perceptual and Motor Skills*, p79, 419-430.
- Paas, F., Tuovinen, J., Tabbers, H., and Gerven, P. (2003). *Cognitive Load Measurement as a Means to Advance Cognitive Load Theory*, *Educational Psychologist*, p.63–71, Lawrence Erlbaum Associates, Inc.
- Plass, J. L., Moreno, R., & Brünken, R. (2010). *Cognitive Load Theory*. Cambridge: Cambridge University Press

Prathibha, D., Latha, B. and Sumathi, G., (2014). *Efficient Scheduling of Workflow in Cloud Environment Using Billing Model Aware Task Clustering*. Journal of Theoretical and Applied Information Technology 31st July 2014, Vol. 65 (No.3)

Prodan, R., Sperk, M. and Ostermann, S., (2012). *Evaluating High-Performance Computing on Google App Engine*. IEEE SOFTWARE

Poghosyan A., V., Harutyunyan, A., N., Grigoryan, N., M., (2016). *Managing Cloud Infrastructures by a Multi-layer Data Analytics*, IEEE International Conference on Autonomic Computing (ICAC)

Pugh, Emerson W.; Johnson, Lyle R.; Palmer, John H., (1991). *IBM's 360 and Early 370 Systems*. Cambridge MA: MIT Press

Ranjan, R. and Zhao, L., (2013). *Peer-to-peer service provisioning in cloud computing environments*. Journal Supercomputing (2013) 65:154–184 DOI 10.1007/s11227-011-0710-5

Rasmussen E. R., (2009). *Reducing IT Costs and Increasing IT Efficiency by Integrating Platform Virtualization in the Enterprise*, University of Oregon

Ravindranath, K. R., (2015). *Clinical decision Support System for Heart Diseases Using Extended Sub Tree*, International Conference on Pervasive Computing (ICPC)

Reddy D. R., Erman L. D., Fennell R. D., and Neely R. B., (1976). *The Hearsay-I Speech Understanding System: An Example of the Recognition Process*, IEEE Transactions on Computers, Vol. C-25, No. 4, April 1976

Redhat (2019). [online]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/virtualization_host_configuration_and_guest_installation_guide/chap-virtualization_host_configuration_and_guest_installation_guide-system_requirements, Redhat: Redhat Incorporated, wholly owned by IBM Corporation [13/10/2019]

Rokne, J., (2013). *Computing: Transforming Information Technology*. IEEE Computer Society, Cloud Computing

Rosenblum M., (2004). *The Reincarnation of Virtual Machines*, Stanford University and VMWare

Rothenburg, J., Paul, J., Kameny, I., Kipps, J. and Swenson, M., (1987). *Evaluation Expert Systems: A Framework and Methodology*, Defense Advanced Research Projects Agency

Rusu, O., Halcu, I., Grigoriu, O., Neculoiu, G., Sandulescu, V., Marinescu, M., and Marinescu V. (2013). *Converting Unstructured and Semi-structured Data into Knowledge*, 11th RoEduNet International Conference, IEEE

Sandru, C., Petcu D., Munteanu V. I., (2012). *Building an Open-Source Platform-as-a-Service with Intelligent Management of Multiple Cloud Resources*, IEEE/ACM Fifth International Conference on Utility and Cloud Computing

Sanzo, P., Rughetti, D., Ciciani, B. and Quaglia, F., (2012). *Auto-tuning of Cloud-based In-memory Transactional Data Grids via Machine Learning*. IEEE Second Symposium on Network Cloud Computing and Applications

Sarathy, V., Narayan, P. and Mikkilineni, R., (2010). *Next Generation Cloud Computing Architecture Enabling Real-time Dynamism for Shared Distributed Physical Infrastructure*, Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises

Schiller, K., (2011). *Amazon EC2 Outage Highlights Risks*. Volume 28, Number 6, www.infotoday.com

Scroggins, R., (2013). *Virtualization Technology Literature Review*. Vol 13, Global Journal of Computer Science and Technology

Seaman, C. B., (1999). *Qualitative methods in empirical studies of software engineering*, IEEE Transactions on Software Engineering, vol. 25, no. 4, pp. 557-572, July-Aug. 1999

Selvi, S., Valliyammai, C., and Dhatchayani, V., (2014) *Resource Allocation Issues and Challenges in Cloud Computing*, International Conference on Recent Trends in Information Technology

Serrano N., Gallardo G., Hernantes J., (2015). *Infrastructure as a Service and Cloud Technologies*, IEEE Software 32 (2), 30-36

Shirinbab S. and Lundberg L. (2016). *Performance Implications of Resource Over-Allocation during the Live Migration*, IEEE 8th International Conference on Cloud Computing Technology and Science

Shirinbab, S., Lundberg, L. and Håkansson J. (2016). *Comparing Automatic Load Balancing using VMware DRS with a Human Expert*, IEEE International Conference on Cloud Engineering Workshop

Song, Y., Sun, Y., and Shi W. (2013). *A Two-Tiered On-Demand Resource Allocation Mechanism for VM-Based Data Centers*, IEEE Transactions on Services Computing, Vol. 6, No. 1, January-March

Spangler, W.E., (1991). *The Role of Artificial Intelligence in Understanding the Strategic Decision-Making Process*. IEEE Transactions on Knowledge and Data Engineering, Vol. 3, No. 2

SPARC International Inc, V., (1992). *The SPARC Architecture Manual v8. Version 8 ed*. SPARC International.

Srnka K. J. and Koeszegi, S. T. (2007). *From Words to Numbers: How to Transform Qualitative Data into Meaningful Quantitative Results*, Schmalenbach Business Review, Vol. 59

Stage, A., Setzer, T., and Bichler, M. (2009). *Automated Capacity Management and Selection of Infrastructure-as-a-Service Providers*, IFIP/IEEE Intl. Symposium on Integrated Network Management — Workshops

Steinder, M., Whalley I., Carrerat D., Gawdat I. and Chess D. (2007). *Server Virtualization in Autonomic Management of Heterogeneous Workloads*, 1-4244-0799-0/07, IEEE

Su, K., (2015). *Affinity and Conflict-Aware Placement of Virtual Machines in Heterogeneous Data Centers*, IEEE Twelfth International Symposium on Autonomous Decentralized Systems

Sweller, J. (1988). *Cognitive Load During Problem Solving: Effects on Learning*, Cognitive Science 12, p.257-285

Tanenbaum, A.S., ed, (2006). *Structured Computer Organization*. 5th ed. Prentice Hall

Tian, C., Wang, Y., Qi, F. and Yin, B., (2012). *Decision Model for Provisioning Virtual Resources in Amazon EC2*. 2012 8th International Conference on Network and Service Management (CNSM 2012): Short Paper

Toyoshima, S., Yamaguchi, S. and Oguchi, M., (2010). *Storage Access Optimization with Virtual Machine Migration and Basic Performance Analysis of Amazon EC2*. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops

Tsai, C., (2009). *System Architectures with Virtualized Resources in a Large-Scale Computing Infrastructure*, The University of Michigan

Ullah S., Awan, M., Khiyal, S (2016). *A Price-Performance Analysis of EC2, Google Compute and Rackspace Cloud Providers for Scientific Computing*, Journal of Mathematics and Computer Science 16, p. 178–192

Unix.com (2019), [online]. Available from: <https://www.unix.com/man-page/centos/8/SYS-UNCONFIG/>
Unix.com: Free Unix Support [10/09/2019]

Vanmechelen, K., De Munck, S. and Broeckhove, J., (2013). *Simulation Modelling Practice and Theory*. Simulation Modelling Practice and Theory 34 (2013)126–143

Vanmechelen, K., De Munck, S. and Broeckhove, J., (2012). *Conservative Distributed Discrete Event Simulation on Amazon EC2*. 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Verdinelli, S., and Scagnoli, N., (2013) *Data Display in Qualitative Research*, International Journal of Qualitative Methods

Virtualbox.org (2019), [online]. Available from: <https://www.virtualbox.org/manual/ch04.html#guestadd-balloon> VirtualBox: virtualbox.org [17/11/2019]

Virtualbox.org (2019), [online]. Available from: <https://www.virtualbox.org/manual/ch09.html#cpuhotplug>
VirtualBox: virtualbox.org [17/11/2019]

Virtualbox.org (2019), [online]. Available from: <https://www.virtualbox.org/manual/ch07.html#teleporting>
VirtualBox: virtualbox.org [21/11/2019]

Vogels, W., Dumitriu, D., Birman K., Gamache R., Massa M., Short R., Vert J., Barrera J., and Gray J. (1998). *The Design and Architecture of the Microsoft Cluster Service-a Practical Approach to High-Availability and Scalability, Digest of Papers*. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing (Cat. No.98CB36224), IEEE

Vrijders, S., Maffione, V., Staessens, D., Salvestrini, F., Biancani, M., Grasa, E., Colle, D., Pickavet, M., Barron, J., and Day, J., (2016). *Reducing the Complexity of Virtual Machine Networking*, IEEE Communications Magazine, p.152-158

Wang, G. and Ng, T., (2010). *The Impact of Virtualization on Network Performance of Amazon EC2 Data Center*. IEEE Infocom 2010 proceedings

Wenbin, C., Xiaoling, L., Yijun, L. and Yu, F., (2010). *A Machine Learning Algorithm for Expert System Based on MYCIN Model*. 2nd International Conference on Computer Engineering and Technology

White S.R., Hanson J.E., Whalley I., Chess D.M. and Kephart J.O. (2004) *An Architectural Approach to Autonomic Computing*, International Conference on Autonomic Computing, Proceedings.

Wikimedia.org (2019), [online]. Available from: [wikimedia.org](https://www.wikimedia.org/) Wikimedia: wikimedia.org [21/12/2019]

Windriyani, P., & Kom, S., Wiharto, W., and Widya S., S. (2013). *Expert System for Detecting Mental Disorder with Forward Chaining Method*. 10.1109/ICTSS.2013.6588068.

Winston P., and Prendergast K. (1986). *XCON: An Expert Configuration System at Digital Equipment Corporation*, MIT Press

Wong, D. and Manickam, S., (2010). *Intelligent Expertise Classification Approach: An Innovative Artificial Intelligence Approach to Accelerate Network Data Visualization*. 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)

- Wood, T., (2011). *Improving Data Center Resource Management Deployment and Availability with Virtualization*, University of Massachusetts
- Wright, F.L. and Gdowski, S., (1987) *An Artificial Intelligence Schema to Perform Automatic Santization of Data*. Monarch Systems Inc, Beverly Hills, California
- Xiong, P., (2012). *Dynamic Monitoring Modeling and Management of Performance and Resources For Applications In The Cloud*, Georgia Institute of Technology
- Xu W. and Liu, X. (2003). *Research on Evaluating Methods of Projects for Complex Systems*, Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xa'an, 2-5 November
- Xue, J., (2009). *Performance Evaluation and Resource Management in Enterprise Systems*, University of Warwick
- Yang J. D., Huhns M. N., and Stephens, L. M., (1985). *An Architecture for Control and Communications in Distributed Artificial Intelligence Systems*, IEEE transactions on Systems, Man, and cybernetics, Vol. SMC-15, No. 3
- Yang, R., Wei, W., and Cummins, M. (2017). *Application of Cognitive Load Theory to the Design and Evaluation of Usability Study of mHealth applications: Opportunities and challenges* IEEE International Conference on Healthcare Informatics
- Zhang, R. and Shang, Y., (2014). *An Automatic Deployment Mechanism on Cloud Computing Platform*. *Cloud Computing Technology and Science (CloudCom)*, 2014 IEEE 6th International Conference
- Zhang, W., Xie, H. and Hsu, C. (2017). *Automatic Memory Control of Multiple Virtual Machines on a Consolidated Server*, IEEE Transactions on Cloud Computing, Vol. 5, NO. 1, January-March
- Zhang, Q., Liu, L., Ren, J., Su, G., and Iyengar, A. (2016). *iBalloon: Efficient VM Memory Balancing as a Service*, IEEE International Conference on Web Services

Appendix

Appendix A – VM Deployment Process

A.1 Expert Users Results

A.2 IDE Results

Step Number	IDE Provisioning (sec)																														
	user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8	user 9	user 10	user 11	user 12	user 13	user 14	user 15	user 16	user 17	user 18	user 19	user 20	user 21	user 22	user 23	user 24	user 25	user 26	user 27	user 28	user 29	user 30	user 31
1	18	25	21	17	19	14	19	24	23	25	21	13	18	22	23	13	27	12	17	25	16	19	20	21	22	27	19	21	18	16	13
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	11	14	7	12	19	21	12	15	9	4	18	19	12	13	9	10	16	19	18	14	6	11	18	27	28	12	11	6	10	13	16
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	1214	1212	1190	1156	1176	1123	1214	1212	1190	1156	1176	1311	1113	1361	1221	1231	1225	1123	1155	1119	1191	1232	1172	1143	1166	1274	1282	1180	1127	1198	1286
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Total Time	1243	1251	1218	1185	1214	1158	1245	1251	1221	1185	1215	1343	1143	1396	1253	1254	1268	1154	1190	1158	1213	1262	1210	1191	1216	1313	1312	1207	1155	1227	1315

A.3 Oracle Results

Oracle Cloud (sec)																															
user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8	user 9	user 10	user 11	user 12	user 13	user 14	user 15	user 16	user 17	user 18	user 19	user 20	user 21	user 22	user 23	user 24	user 25	user 26	user 27	user 28	user 29	user 30	user 31	
20	25	21	19	22	20	22	21	19	23	20	25	21	19	24	20	27	21	19	13	20	19	21	18	21	20	17	21	19	29	20	
124	133	110	99	156	122	134	101	99	143	113	133	109	99	142	115	133	97	99	101	165	129	110	99	164	131	133	110	99	147	123	
10	12	14	9	17	9	8	14	9	12	13	15	14	8	10	15	9	13	8	10	17	9	12	19	9	11	15	16	14	9	17	21
153	120	112	175	194	153	174	120	195	192	152	137	112	170	176	179	183	199	112	158	176	137	155	145	187	196	122	160	143	126	114	
63	59	64	67	79	89	57	85	70	83	56	78	62	72	59	65	71	69	76	73	79	68	80	62	91	58	94	82	65	75	88	
262	250	350	367	385	325	240	380	299	346	274	247	267	380	242	246	265	230	382	269	296	275	291	258	243	241	382	386	286	275	263	
69	55	45	25	56	53	28	25	51	47	57	36	46	60	37	39	63	44	31	20	50	71	59	38	36	25	48	49	29	64	25	
631	659	704	840	622	856	673	680	875	695	712	709	683	684	842	791	803	818	648	646	828	711	607	697	784	663	737	852	865	609	776	
35	129	65	79	46	122	70	111	91	37	119	70	45	66	123	98	117	94	110	56	98	37	39	47	84	107	35	90	119	81	108	
984	912	801	704	913	782	906	935	791	835	815	692	857	705	938	728	761	893	691	773	828	791	751	919	681	919	777	940	841	681	839	
2354	2354	2266	2284	2490	2515	2312	2472	2499	2413	2331	2142	2216	2263	2593	2296	2452	2478	2179	2176	2649	2250	2132	2392	2302	2375	2361	2704	2475	2104	2377	

A.4 AWS Results

AWS (sec)																															
user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8	user 9	user 10	user 11	user 12	user 13	user 14	user 15	user 16	user 17	user 18	user 19	user 20	user 21	user 22	user 23	user 24	user 25	user 26	user 27	user 28	user 29	user 30	user 31	
16	18	15	12	17	20	15	15	15	17	15	15	16	18	16	20	15	19	15	16	20	19	15	20	20	22	20	18	17	16	19	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	20	30	21	28	19	17	15	24	11	26	28	16	35	22	26	23	29	27	20	35	21	26	19	28	24	25	28	23	33	29	
14	25	35	18	25	23	21	19	41	32	41	21	33	29	23	16	21	39	25	11	15	25	33	36	35	33	21	42	39	28	35	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
650	451	489	998	735	610	697	432	521	751	750	457	341	615	539	541	656	862	709	604	677	744	702	480	786	533	686	493	446	570	473	
127	129	111	118	144	113	103	141	141	112	153	121	109	144	106	139	109	102	113	121	153	123	143	113	134	126	152	112	105	104	137	
601	622	587	699	659	683	552	670	584	595	504	624	644	645	485	693	692	642	661	606	490	684	587	545	480	503	601	588	532	608	535	
1418	1265	1267	1266	1628	1468	1405	1192	1326	1518	1489	1266	1359	1486	1191	1429	1510	1493	1550	1378	1390	1616	1506	1213	1433	1241	1505	1281	1161	1359	1228	

A.5 Experienced Users Results

A.6 IDE Results

Step Number	IDE Provisioning (sec)																														
	user 1	user 2	user 3	user 4	user 5	user 6	user 7	user 8	user 9	user 10	user 11	user 12	user 13	user 14	user 15	user 16	user 17	user 18	user 19	user 20	user 21	user 22	user 23	user 24	user 25	user 26	user 27	user 28	user 29	user 30	user 31
1	20	23	34	17	37	38	25	23	30	24	29	36	34	33	21	24	22	19	31	32	45	44	19	37	25	36	41	42	27	38	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	12	15	15	15	13	16	13	12	13	19	19	18	13	13	11	16	16	11	13	18	15	13	16	11	14	12	14	19	17	18	
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	1223	1223	1421	1221	1466	1282	1181	1530	1519	1160	1190	1289	1415	1323	1120	1472	1537	1545	1421	1383	1443	1205	1224	1114	1209	1348	1395	1435	1275	1344	1118
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Total Time	1265	1361	1460	1363	1516	1336	1219	1565	1555	1209	1233	1336	1464	1379	1164	1509	1577	1578	1453	1432	1490	1263	1284	1144	1260	1285	1445	1490	1336	1388	1174

A.7 Oracle Results

Table with columns Oracle Cloud (sec), user 1 through user 31. Rows show performance metrics for various user configurations.

A.8 AWS Results

Table with columns AWS (sec), user 1 through user 31. Rows show performance metrics for various user configurations.

A.9 Novice Users Results

A.10 IDE Results

Table with columns Step Number, IDE Provisioning (sec), user 1 through user 31. Rows show provisioning times for various users.

A.11 Oracle Results

Table with columns Oracle Cloud (sec), user 1 through user 31. Rows show performance metrics for various user configurations.

A.12 AWS Results

Table with columns AWS (sec), user 1 through user 31. Rows show performance metrics for various user configurations.

Appendix B – Blind Peer Review Comments (Published Papers)

B.1 Simplified Deployment of Virtual Machines Using an Intelligent Design Engine ([Oakes et al, 2016](#))

B.2 Blind Review 1

Review Questions:

Detailed Comments	<p>The paper presents an interesting mechanism to enable simplified deployment of VMs.</p> <p>It is mainly focused on the implementation details rather than the mechanism advantages, comparisons and motivation.</p> <p>The presentation should be improved. E.g. the quality of figure 2 is very bad, and the procedures should be declared as such, not as figures.</p> <p>The related work should be improved, and the position of the proposal should be made more clear.</p>
Please rate your satisfaction with the basic sections (introduction, conclusion, works cited, etc.)?	Fair.
The material is ordered in a way that is logical, clear, and easy to follow?	Good.
The writer adequately summarizes and discusses the topic?	Good.
The writer makes some contribution of thought to the paper or merely summarizes data or publications?	Good.
The writer introduces and documents sources adequately and appropriately?	Fair.
The formatting of the manuscript is in accordance to the prescribed paper format?	Fair.

The paragraphs and sentences are cohesive (flow together smoothly without disruption in the train of thought)?	Fair.
Potential interest to research community	Acceptable.
Originality of the work	Acceptable.
Significance of the main idea(s)	Acceptable.
Technical quality of the paper	Acceptable.
Author response	No concerns with review comments raised.

B.3 Blind Review 2

Detailed Comments	<p>This paper proposes an alternative solution for the deployment of an intelligent private or public cloud compute platform, built around a set of predefined rule-based parameters with the purpose of providing a simplified process for provisioning VMs.</p> <p>The paper reads more like a technical project report than a research paper. The main problem with this paper is that it does not clearly identify how it is different and better than previous work in this area.</p> <p>The presentation and writing of the paper should also be improved. The paper's writing and organization need significant improvement in order for it to be readable and technically clear.</p> <p>The main weakness of the paper lies in its lack of originality and novelty; without any performance evaluation and comparison with other implementations to show its advantages or uniqueness.</p>
Please rate your satisfaction with the basic sections (introduction, conclusion, works cited, etc.)?	Fair.
The material is ordered in a way that is logical, clear, and easy to follow?	Fair.

The writer adequately summarizes and discusses the topic?	Poor.
The writer makes some contribution of thought to the paper or merely summarizes data or publications?	Poor.
The writer introduces and documents sources adequately and appropriately?	Poor.
The formatting of the manuscript is in accordance to the prescribed paper format?	Fair.
The paragraphs and sentences are cohesive (flow together smoothly without disruption in the train of thought)?	Fair.
Potential interest to research community	Unattractive.
Originality of the work	Unattractive.
Significance of the main idea(s)	Unattractive.
Technical quality of the paper	Unattractive.
Author response	At the time the paper was written only the provisioning mechanism, IDE engine, and preliminary evaluation results were available. There has since been significant work completed to provide further evidence that the simplified VM deployment approach does reduce the time to create and access VMs, reduce human errors, and improve build consistency. The additional details can be found in section 5.2 'Simplified VM Provisioning', section 5.3.1 'VM Provisioning Timed Results' and section 5.3.2 'Aggregated VM Provisioning Results'.

B.4 Measuring and Reducing the Cognitive Load for End Users of Complex Systems
[\(Oakes et al, 2019\)](#)

B.5 Blind Review 1

<p>Detailed Comments</p>	<p>This paper examines a method and approach to measure how complex a system is to use, and how to reduce the complexity of such systems by minimising the requirement for human inputs as much as possible, in order to reduce the cognitive load for that user, or group of users.</p> <p>This paper addresses a study completed around using virtualised computer management systems interfaces of two well-known products AWS, Oracle Cloud, and compares the complexity of the steps and interface for end users to a private cloud less well-known system called the IDE.</p> <p>This paper is very well written. I have just one suggestion. The virtualised computer management systems introduced in this paper are very powerful. They can be potentially applicable to the study of social opinion evolution.</p> <p>See the seminal paper 'Hybrid consensus for averager-copier-voter networks with non-rational agents'. This future direction can be mentioned in the conclusion section to further guide the readers and establish a new connection to a wider audience.</p>
<p>Please rate your satisfaction with the basic sections (introduction, conclusion, works cited, etc.)?</p>	<p>Good.</p>
<p>The material is ordered in a way that is logical, clear, and easy to follow?</p>	<p>Very Good.</p>
<p>The writer adequately summarizes and discusses the topic?</p>	<p>Good.</p>

The writer makes some contribution of thought to the paper or merely summarizes data or publications?	Good.
The writer introduces and documents sources adequately and appropriately?	Very Good.
The formatting of the manuscript is in accordance to the prescribed paper format?	Very Good.
The paragraphs and sentences are cohesive (flow together smoothly without disruption in the train of thought)?	Very Good.
Are there any grammar, punctuation, or spelling errors?	Little error.
Author Response	No concerns with review comments raised.

B.6 Blind Review 2

Detailed Comments	<p>The paper lacks crucial parts: related work, evaluation.</p> <p>Limitation of the study must be highlighted.</p> <p>Add portion of discussion to share your thoughts.</p> <p>Future work is not explained / more analysis of results is needed. More conclusions and recommendations, also.</p> <p>References must be recent; references older than five years should only be cited if necessary.</p>
Please rate your satisfaction with the basic sections (introduction, conclusion, works cited, etc.)?	Fair.
The material is ordered in a way that is logical, clear, and easy to follow?	Fair.

The writer adequately summarizes and discusses the topic?	Fair.
The writer makes some contribution of thought to the paper or merely summarizes data or publications?	Fair.
The writer introduces and documents sources adequately and appropriately?	Fair.
The formatting of the manuscript is in accordance to the prescribed paper format?	Fair.
The paragraphs and sentences are cohesive (flow together smoothly without disruption in the train of thought)?	Fair.
Are there any grammar, punctuation, or spelling errors?	No.
Author response	No concerns with review comments raised.

Appendix C - VM Platform Build Process

C.1 IDE Provisioning

The following appendix details the process experiment steps for deploying as simply as possible a VM using the IDE.

C.2 VM Deployment Steps

Step 1 &2: IDE access internally web-based on private network:

Hostname:	<input type="text" value="saturn"/>	Hint: Enter hostname of the VM
VM Size:	<input type="text" value="small"/>	Hint: Small, Medium or Large
Server:	<input type="text" value="server_21"/>	Hint: Host System
<input type="button" value="Deploy VM"/>		

Step 3, 4, 5, 6, 7, and 8: Select one click deploy VM – web browser output:

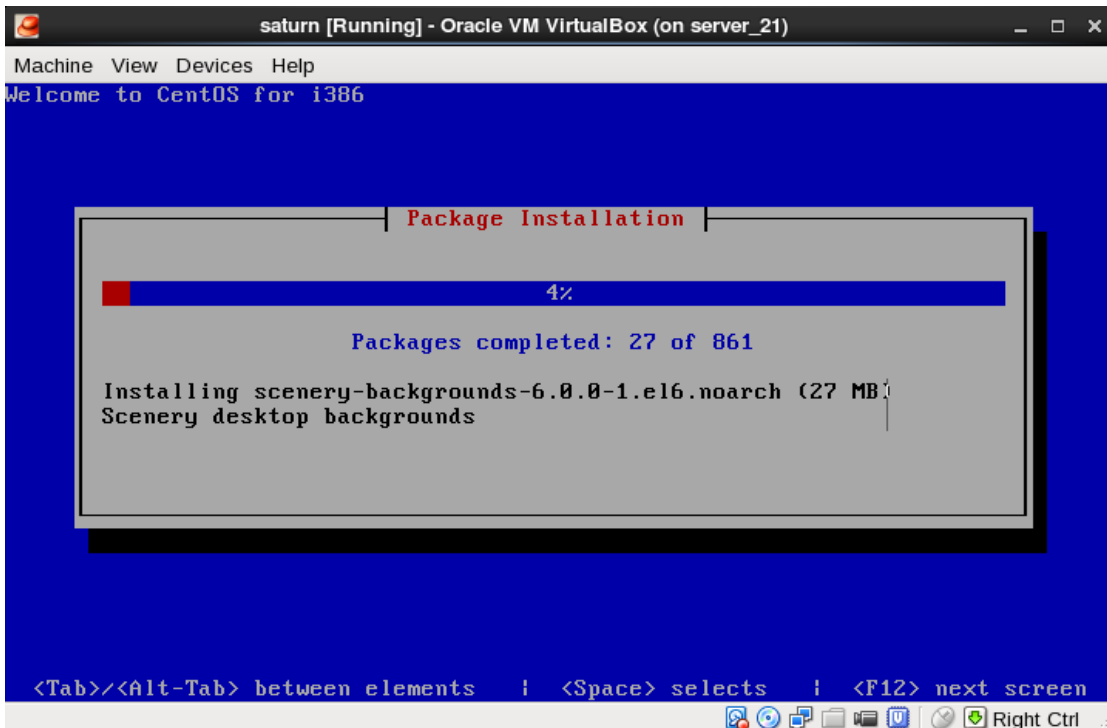
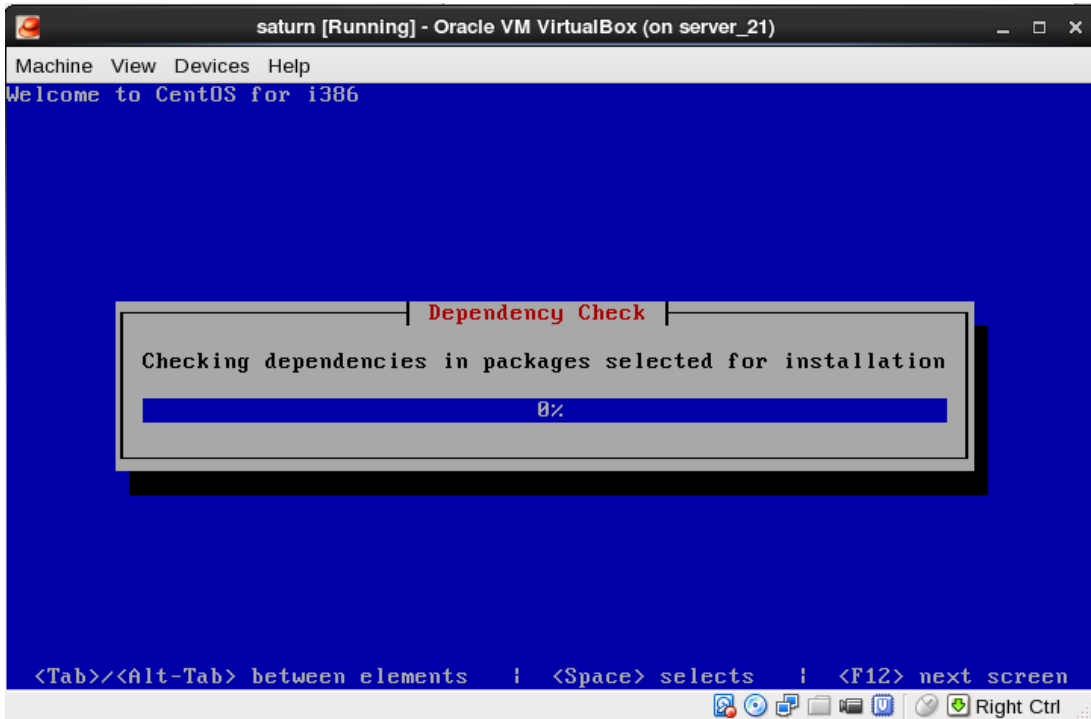
```

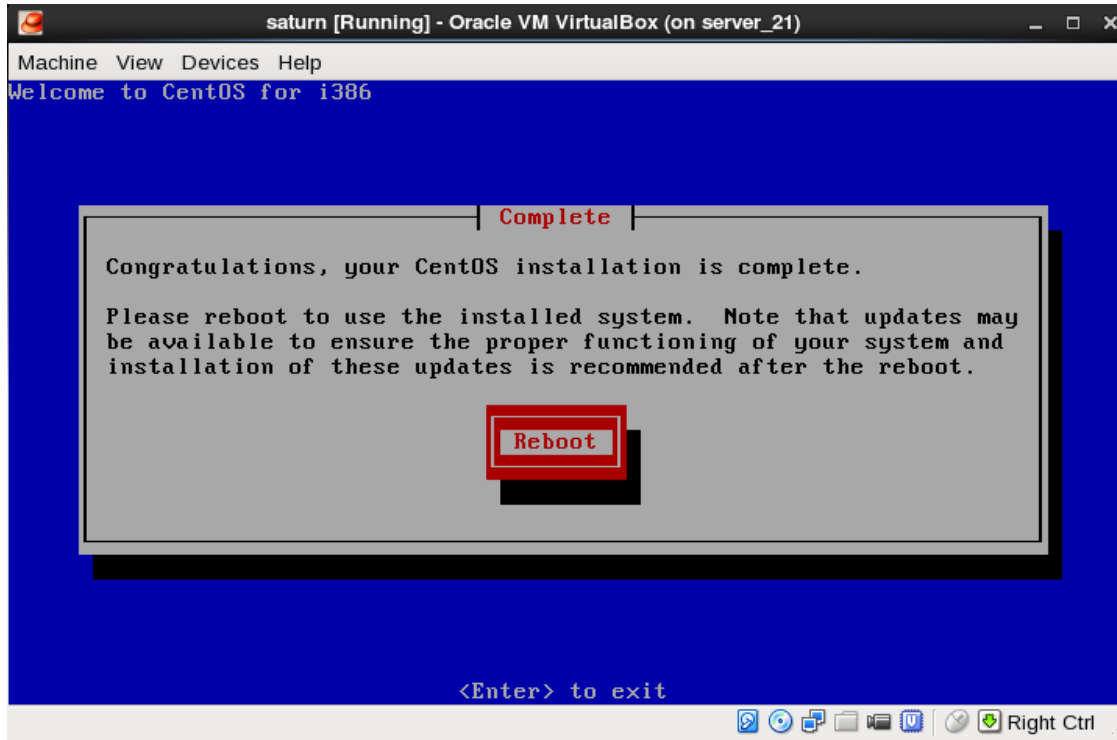
192.168.3.250/system.php
Google
begin creation.saturnsmallserver_21
/usr/bin/sudo /server/scripts/nyx_createvm.sh saturn small server_21
Pseudo-terminal will not be allocated because stdin is not a terminal. Failed to
open the X11 display! ssh -t -X root@server_21 /server/scripts
/nyx_vboxcreate.sh saturn small Pseudo-terminal will not be allocated because
stdin is not a terminal.
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100% Interface
'vboxnet15' was successfully created Virtual machine 'saturn' is created and
registered. UUID: ce177659-021a-47e3-b2f7-2f2de4a3cdf Settings file: '/root
/VirtualBox VMs/saturn/saturn.vbox'
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100% Disk
image created. UUID: 0be0d437-c324-4509-a78d-7e90ed4575ee Waiting for
VM "saturn" to power on... VM "saturn" has been successfully started. end
creation.

```

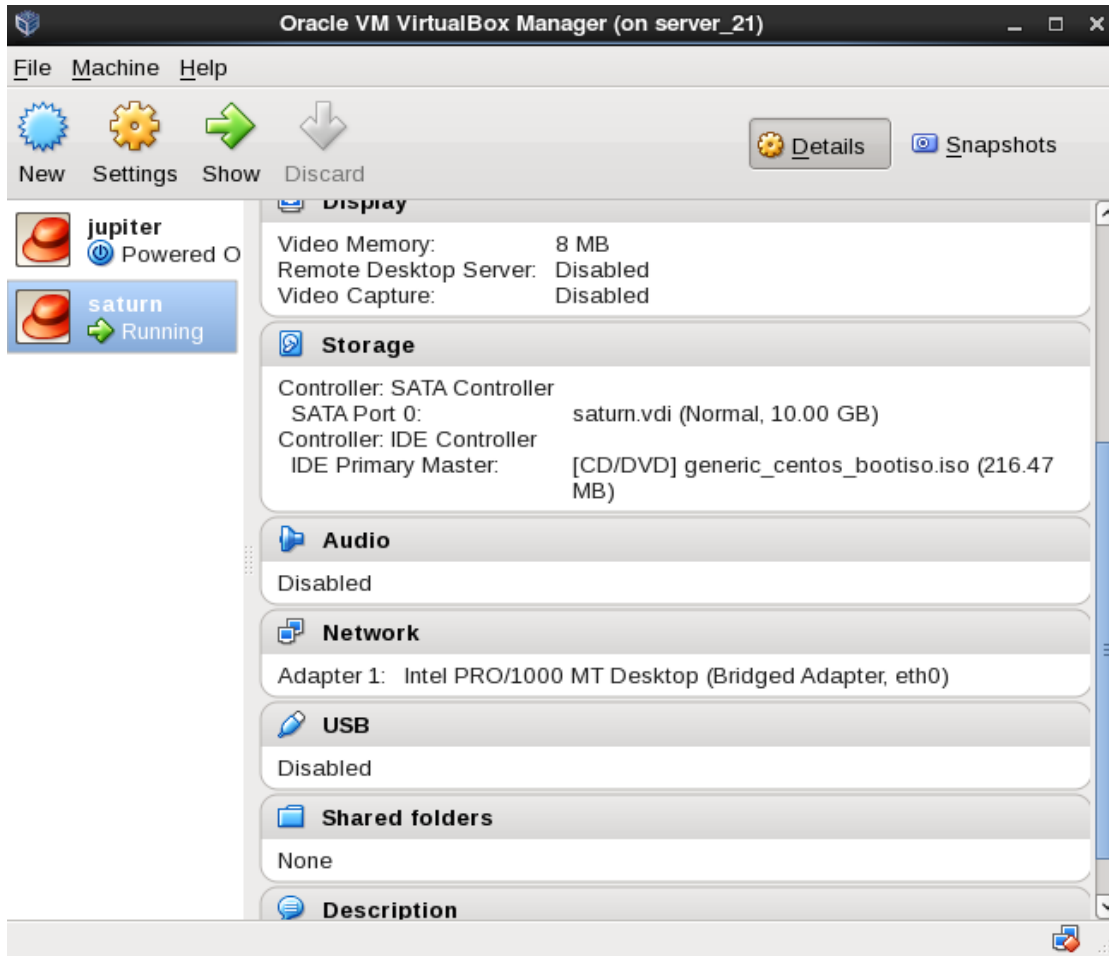
Step 9 Kickstart configuration, build and install example:



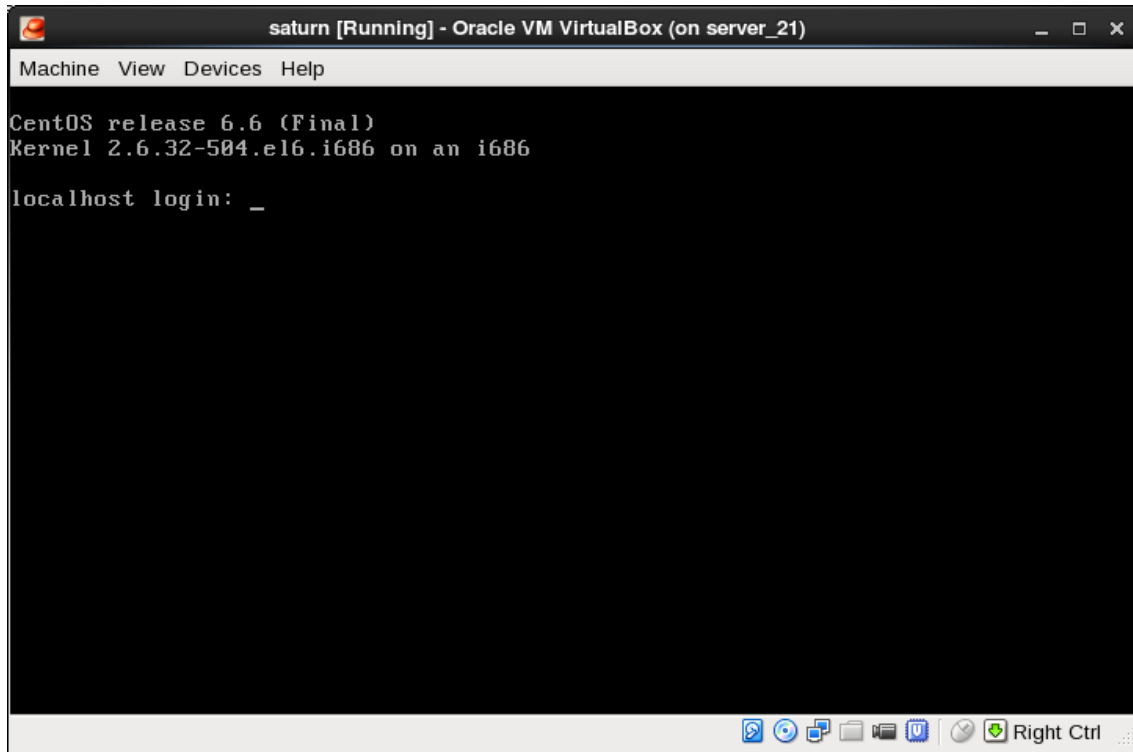




Step 9: VM automatically created & running:



Step 10: Process for accessing the VM – automatic SSH access and secure RSA key installed:



Automatic key SSH RSA configuration and subsequent access:

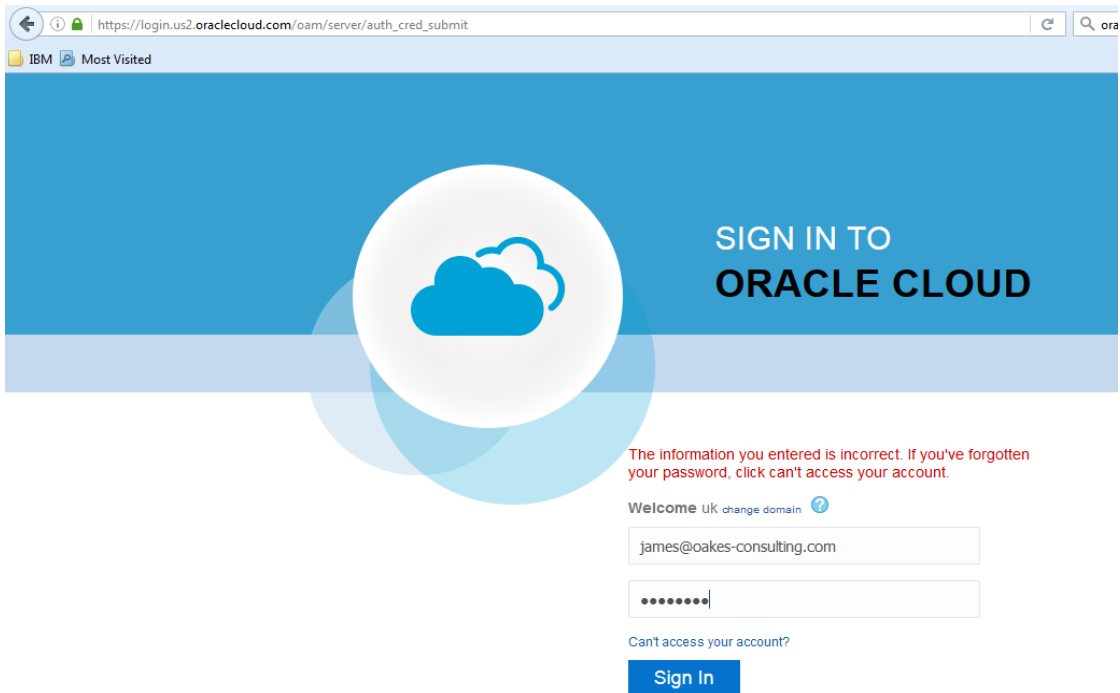
```
[root@ide_server VMbuild]# ssh root@saturn
The authenticity of host 'saturn (192.168.3.229)' can't be established.
RSA key fingerprint is 57:9a:97:ef:ba:ed:d8:2e:25:27:62:a8:26:1c:06:0f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'saturn' (RSA) to the list of known hosts.
Last login: Mon Nov 14 22:09:48 2016 from 192.168.3.250
[root@localhost ~]#
[root@localhost ~]# pwd
/root
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        8.3G  1.9G  6.0G  25% /
tmpfs            504M   0    504M   0% /dev/shm
/dev/sda1        488M  25M  439M   6% /boot
[root@localhost ~]#
[root@localhost ~]# █
```

C.3 Oracle Cloud Provisioning

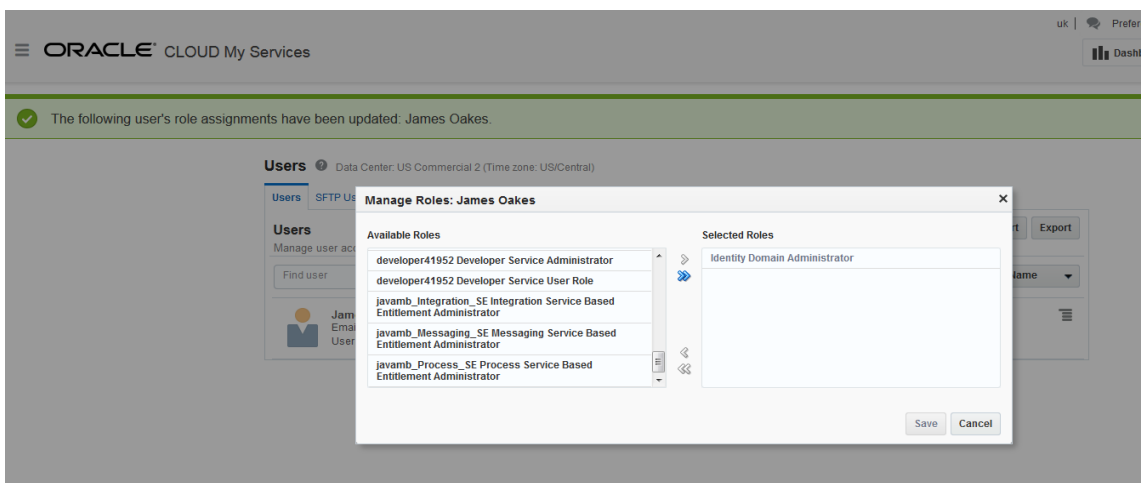
The following appendix details the process experiment steps for deploying as simply as possible a VM in the Oracle Cloud; the know-how required to provision a VM is considerable in terms of complexity.

C.4 VM Deployment Steps

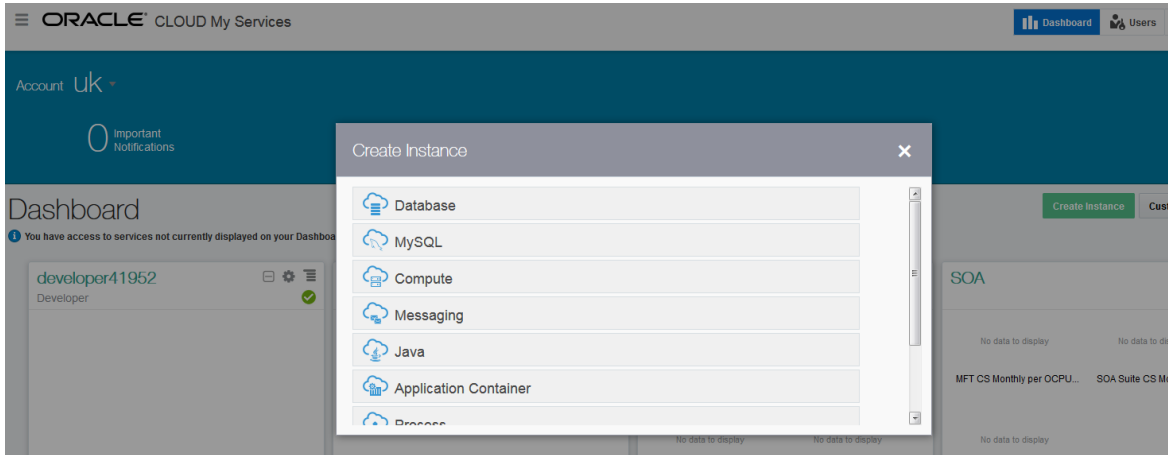
- Step 1: Access Oracle Cloud:



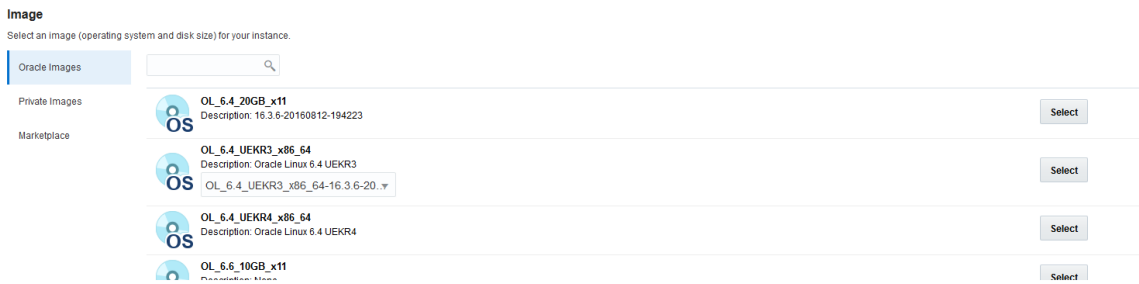
- Step 2: Configure Role:



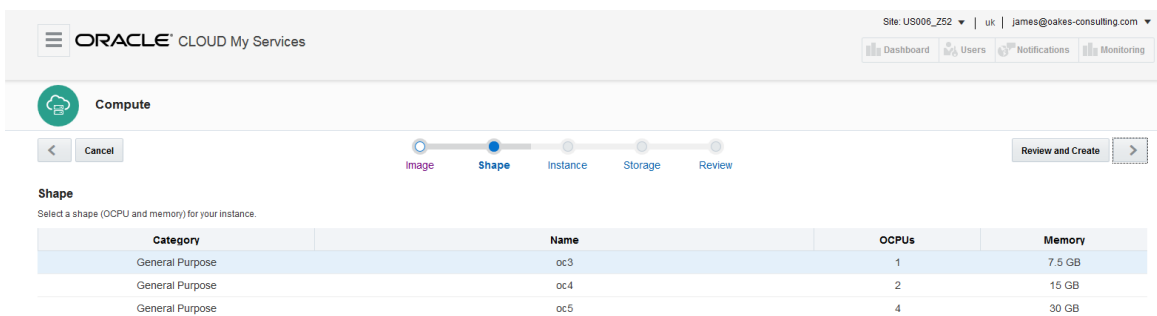
- Step 3: Select compute as the option for VM deployment:



- Step 4: Select the image you wish to use to install to the VM (OS version):



- Step 5: Select the VM CPU and Memory Parameters:



- Step 6: Define VM Parameters:

Instance

Enter the required details to create your instance. [Learn more.](#)

High Availability Policy: Active

Name: Oracle_Solaris_11_3_2016110715363

Label: Oracle_Solaris_11_3_2016110715363

Description: [Empty]

Tags: [Empty]

DNS Hostname Prefix: [Empty]

Public IP Address: Auto Generated

IP Networks: Click here to select existing IP networks [Create IP Network]

Security Lists: Click here to select existing lists [Create Security List]

SSH Keys: Click here to select existing keys [Add SSH Public Key]

Custom Attributes: [Empty]

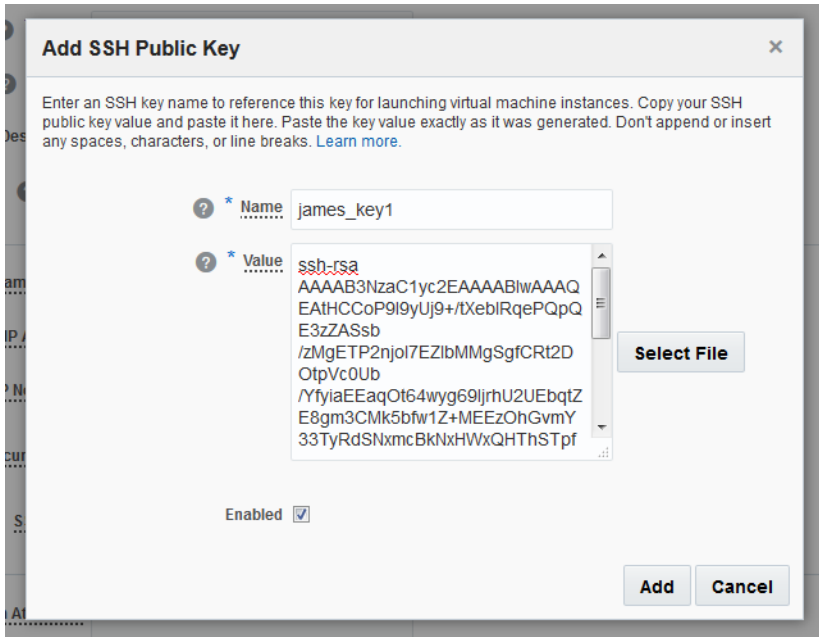
- Step 7: Define VM Storage:

Storage

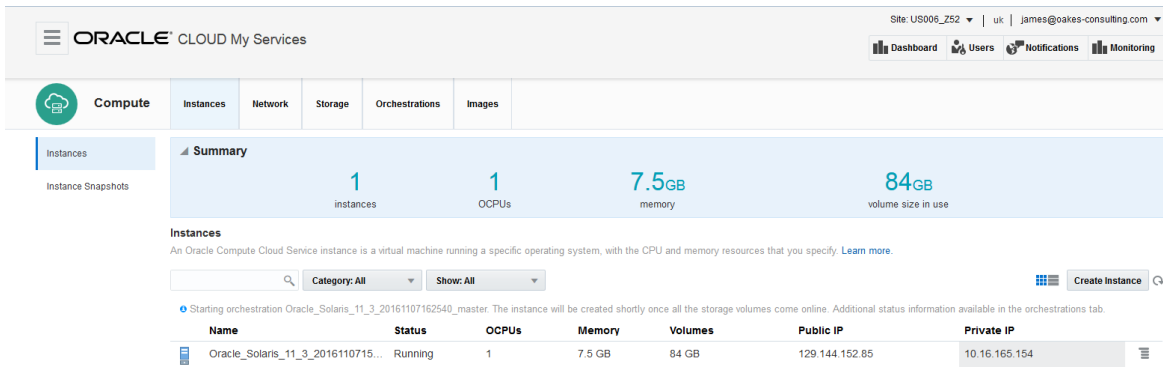
You can attach existing storage volumes, or create and attach a storage volume to the instance. A persistent boot volume is created and used to boot your instance by default. You can specify a different boot disk, or remove the persistent boot disk and boot from a nonpersistent boot disk instead. You can also attach additional storage volumes to an instance after the instance is created.

Name	Disk #	Size	Type	Delete On Termination	Boot Drive
Oracle_Solaris_11_3_20161107153632_storage	1	34 GB	storage/default	<input type="checkbox"/>	<input checked="" type="checkbox"/>
james_test01	2	50 GB	storage/default	<input type="checkbox"/>	<input type="checkbox"/>

- Step 8: Add SSH key, create a key and upload the public key:



- Step 9: VM Creation process:



- Step 10: Process for accessing the VM via the internet:

10 Accessing an Oracle Solaris Instance Using SSH

In instances created by using any of the Oracle-provided Oracle Solaris images, a user named `opc` is preconfigured. The `opc` user is assigned the System Administrator profile and can perform basic administration tasks without entering a password by using `pfexec`.

Prerequisites

- Ensure that the SSH private key corresponding to the public key that you associated with your instance while creating it is available on the host from which you want to `ssh` to the instance.
- Ensure that the instance has a public IP address. To find out the public IP address of your instance, view the information on the Instances page. See [Listing Instances](#).

Procedure

You can use SSH to log in to your instance as the default user, `opc`, by using the following command:

```
ssh opc@ip_address -i private_key
```

- `ip_address` is the public IP address of the instance.

If you've enabled a VPN tunnel to your Oracle Compute Cloud Service site, you can use the private IP address of your instance to connect to the instance. To set up a VPN tunnel, see [Connecting to Instances in a Dedicated Site Using VPN](#).

- `private_key` is the full path and name of the file that contains the private key corresponding to the public key associated with the instance that you want to access.

If an error occurs, see [Can't connect to an instance using SSH](#).

When you're logged in as the `opc` user, you can use the `pfexec` command to run administrative tasks.

C.5 AWS Cloud Provisioning

The following appendix details the process experiment steps for deploying as simply as possible a VM in the AWS Cloud; the know-how required to provision a VM is considerable in terms of complexity.

C.6 VM Deployment Steps

- Step 1 and 2 – Login and obtain role/access:



Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

E-mail or mobile number:

I am a new user.

I am a returning user
and my password is:

Keep me signed in. [Details](#)

- Step 3 – Select Compute & Quick Launch:

Build a solution

Get started with simple wizards and automated workflows.



Launch a virtual machine

With EC2
~1 minutes



Build a web app

With Elastic Beanstalk
~6 minutes



Deploy a serverless microservice

With Lambda, API Gateway
~2 minutes



Host a static website

With S3, CloudFront, Route 53
~5 minutes



Create a backend for your mobile app

With Mobile Hub
~5 minutes



Register a domain

With Route 53
~3 minutes



Quick Launch an EC2 Instance

Amazon EC2 provides virtual machines in the AWS cloud, known as EC2 instances.

This quick launch wizard lets you create an EC2 instance with AWS-recommended default configuration. If you need more options or fine-grained control over instance parameters, please use the [advanced EC2 Launch Instance wizard](#).

Get Started

- Step 4,5,6 and 7 – Configure VM parameters, OS image and more:

Quick Launch an EC2 Instance

Get started creating a General Purpose instance in the **US West (Oregon)** region that is powerful enough to run most web apps.

Name your EC2 instance

This is how you will identify your instance in AWS console. Choose a name that is easy for you to remember.

Example: MyFirstInstance

Use this name

Cancel

Quick Launch an EC2 Instance

Get started creating a General Purpose instance in the **US West (Oregon)** region that is powerful enough to run most web apps.

james-test

Select an Operating System



Next

Quick Launch an EC2 Instance

Get started creating a General Purpose instance in the **US West (Oregon)** region that is powerful enough to run most web apps.

james-test

Select an Operating System



Red Hat Enterprise Linux 7.2

Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type

Don't see the OS you are looking for? AWS offers additional options through the [advanced EC2 Launch Instance wizard](#) or you can explore the [AWS Marketplace](#).

Next

Select an Operating System



Red Hat Linux



Select an instance type



t2.micro

1 Core vCPU (up to 3.3 GHz), 1 GiB Memory RAM, 8 GB Storage **FREE TIER ELIGIBLE**

Need a different instance type? AWS offers additional options through the [advanced EC2 Launch Instance wizard](#).

Next

Select an Operating System



Red Hat Linux



Select an instance type



t2.micro

1 Core vCPU (up to 3.3 GHz), 1 GiB Memory RAM, 8 GB Storage

FREE TIER ELIGIBLE



Create a key pair

Amazon EC2 secures your instance using a key pair. In this step you will download the private key to your computer.

Save it in a safe place and use it when you connect to your instance.

Download

Cancel

- Step 8 – configure SSH key:

Create a key pair

Amazon EC2 secures your instance using a key pair. In this step you will download the private key to your computer.

Save it in a safe place and use it when you connect to your instance.

AWS does not keep a copy of your private key and it cannot be recovered if lost. **Please save it in a safe place.**

Okay! Start Download

Select an instance type



t2.micro

1 Core vCPU (up to 3.3 GHz), 1 GiB Memory RAM, 8 GB Storage **FREE TIER ELIGIBLE**

Private Key



james-test


[Generate a new key](#)

Create a key pair

Amazon EC2 secures your instance using a key pair. In this step you will download the private key to your computer.

Save it in a safe place and use it when you connect to your instance.

Private Key

 **james-test-key**
Generate a new key

- Step 9 & 10 – VM creation process:

The screenshot shows the AWS Management Console interface. At the top, there is a navigation bar with 'Services' and 'Resource Groups' dropdown menus. The main heading is 'Your Instance is Launching!'. Below this, a message states: 'Amazon EC2 is launching your instance. This process should only take a few minutes. You can proceed to the [EC2 Console](#) while this process takes place.' A central box displays a blue cube icon, the instance name 'james-test', and the status 'Status: In progress...'. Below this box, a section titled 'While you wait, learn more about...' contains three columns of information: 'Managing your Instance', 'Connecting to your Instance', and 'Securing your Instance'. Each column includes a brief description and a link to the EC2 console. At the bottom center, there is a blue button labeled 'Proceed to EC2 console'.

This screenshot is identical to the one above, but the instance 'james-test' is now in a 'Completed!' state. The central box shows a green checkmark icon instead of the cube, and the status text reads 'Status: Completed!'. The rest of the interface, including the navigation bar, introductory text, and 'While you wait' section, remains the same.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name
james-test	i-048214b4dd811c7ef	t2.micro	us-west-2b	running	Initializing	None	ec2-35-164-244-66.us-...	35.164.244.66	james-test-key

- Step 10 – Connect to your VM instance:

Connect To Your Instance ✕

I would like to connect with

A standalone SSH client
 A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to [connect using PuTTY](#))
2. Locate your private key file (james-test-key.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:


```
chmod 400 james-test-key.pem
```
4. Connect to your instance using its Public DNS:


```
ec2-35-164-244-66.us-west-2.compute.amazonaws.com
```

Example:

```
ssh -i "james-test-key.pem" ec2-user@ec2-35-164-244-66.us-west-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

Appendix D – IDE Build Procedures

D.1 Procedure 1

Procedure 1: Deploy VM

```

DECLARE strings v, p, s, o, c, h, b, n // VM name, VM path, source media path, OS name type,
// controller type, VM disk path, boot order, network type

DECLARE an integer z, m, u, y // hard disc size in MB, memory in MB, CPU (threads), CPU
// execution cap percentage (zero means none)

DECLARE a Boolean x // pae (physical address extension) true(1) or false(0)

GET values for v,p,s,o,c,h,b,n,z,m,u,y,x // get values from IDE for system values

PROCEDURE deployvm (v,p,s,o,c,h,b,n,z,m,u,y,x) // pass parameters through
FOR EACH component v,p,s,o,c,h,z // process each component in order
    build virtualmanagement component // register each component in the hypervisor
END FOR EACH
FOR EACH modification b,z,m,u,n,y,x // modify VM configuration
    IF (x = 0 OR y = 0) THEN // if pae false or execution cap zero
        ignore modification // ignore or skip modification
    ELSE
        modify virtualmanagement component // modify VM values
    ENDIF
END FOR EACH
start virtualmachine v option headless // start the VM in headless mode
END PROCEDURE

```

D.2 Procedure 2

Procedure 2: Configure Bespoke Source

```

DECLARE strings v, s, o, b, l, m, x // VM name, General source ISO media path, ISO build directory,
// unique VM ISO source identifier, isolinux source identifier,
// images source identifier, current isolinux.cfg

GET values for v,s,o,b,l,m,x // get values from IDE for system values

PROCEDURE Buildsource
IF media S not mounted THEN // check media is available to extract from
    mount ISO media using loopback // create the loopback mountpoint
ENDIF
IF media target o does not exist THEN // check the target ISO directory exists
    create ISO build directory // create the directory
ENDIF
IF a modified version of x exists THEN // protect existing configuration
    preserve a copy of x // take backup copy of isolinux.cfg
ENDIF
recursively copy over source l to target o // recursively copy of source isolinux directory
recursively copy over source m to target o // recursively copy of the source images directory
IF a modified version of x was taken THEN // check if restore is required
    restore a copy of x to target o // restore configuration
ENDIF
change to directory o // make current directory bespoke ISO directory
dynamically build customised image b // build customised mini ISO using mdisofs
IF build of b exit zero THEN // check mdisofs successful exit zero
    restore a copy of x to target o // restore configuration
ELSE
    report error // log an error – IDE decides further action(s)
ENDIF
END PROCEDURE

```

D.3 Procedure 3

Procedure 3: Export Shares

```

DECLARE a string s, p, q, z           // ISO media filename, ISO OS mountpoint, kickstart
                                       // configuration directory, kickstart configuration mountpoint

GET values for s,p,q,z              // get values from IDE for system values

PROCEDURE exportfs (s,p,q,z)        // pass parameters through
  create mountpoint q                // create kickstart configuraton directory
  create mountpoint p                // create OS directory
  loop back mount file s to mountpoint p // create loopback mount for OS from ISO
  bind directory q to mountpoint z   // create bind mount for kickstart configuration
  NFS export mountpoint p readonly   // NFS export for OS data
  NFS export mountpoint z readonly   // NFS export for install data
END PROCEDURE

```

D.4 Procedure 4

Procedure 4: Build Install Configuration

```

DECLARE an array type string c       // Array S to hold all kickstart data
DECLARE a string s                   // kickstart output configuration file
DECLARE integers b, e                // Array size definition, Array counter

GET values for c,s,b                 // get values from IDE for system values

backup original kickstart config      // backup original kickstart configuration
set b equal to array element size    // set variable equal to array dimension
set e to zero                          // array counter

FUNCTION writefile (x)
  write value x to file s
  return
END FUNCTION

WHILE e is less than or equal to b   // process all array values
  Call FUNCTION writefile (array c[element e]) // repeat write values
  increment e by one                  // increment array counter
END WHILE

```

D.5 Source of Knowledge Rules

- Knowledge rules:
 - Simple set of initial rules (expert heuristic knowledge, e.g. self-discovery)
 - Avoid having redundant rules within the set (i.e. rules not used).
- Focus on doing things well (e.g. high utilisation/relevance of rules), with a structured set of situations based on the subjects areas investigated.
- Source of knowledge rules is based on:
 - The authors expert knowledge.
- Supporting common-sense of rules/actions from other experts in the same field of study
- Avoidance of:
 - Subsequent human modification.

Appendix G – Original Proposal

G.1 Aims & Objectives

1. Perform a detailed investigation and analysis of existing computer virtualisation and intelligent management systems, in order to provide underpinnings and evidence of originality of the project.
2. Design and develop a real-time system performance monitoring tool to provide statistical data on CPU/Memory/IO usage and health, enabling data to be gathered reliably from all remote systems ready for processing by the planned intelligent management system.
3. Investigate relevant Artificial Intelligence (AI) techniques for use within the development of an Intelligent Decision Engine (IDE) to automatically manage workloads and virtualised components.
4. Integrate the system performance monitoring tool with the IDE to enable it to process real-time data inputs and make effective management decisions/actions based on the data feeds/analysis.
5. Undertake a series of experimental trials to evaluate the performance monitoring tool and the IDE, within a suitable development framework using formulated test scenarios and data.
6. To undertake a live demonstration of the final working platform as a proof of concept in operation.