PERFORMANCE ANALYSIS OF VIRTUALISATION IN A CLOUD COMPUTING PLATFORM

K. M. MAIYAMA

Ph.D

2019
PERFORMANCE ANALYSIS OF VIRTUALISATION IN A CLOUD COMPUTING PLATFORM

An application driven investigation into modelling and analysis of performance vs security trade-offs for virtualisation in OpenStack infrastructure as a service cloud computing platform architectures

Kabiru Muhammad MAIYAMA

Submitted for the Degree of
Doctor of Philosophy

Department of Computer Science
Faculty of Engineering and Informatics
University of Bradford

2019
Kabiru Muhammad Maiyama

Performance Analysis of Virtualisation in a Cloud Computing Platform

An application driven investigation into modelling and analysis of performance vs security trade-offs for virtualisation in OpenStack infrastructure as a service (IaaS) cloud computing platform architectures.


Virtualisation is one of the underlying technologies that led to the success of cloud computing platforms (CCPs). The technology, along with other features such as multi-tenancy allows delivering of computing resources in the form of service through efficient sharing of physical resources. As these resources are provided through virtualisation, a robust agreement is outlined for both the quantity and quality-of-service (QoS) in a service level agreement (SLA) documents. QoS is one of the essential components of SLA, where performance is one of its primary aspects. As the technology is progressively maturing and receiving massive acceptance, researchers from industry and academia continue to carry out novel theoretical and practical studies of various essential aspects of CCPs with significant levels of success.

This thesis starts with the assessment of the current level of knowledge in the literature of cloud computing in general and CCPs in particular. In this context, a substantive literature review was carried out focusing on performance modelling, testing, analysis and evaluation of Infrastructure as a Service (IaaS), methodologies.
To this end, a systematic mapping study (SMSs) of the literature was conducted. SMS guided the choice and direction of this research.

The SMS was followed by the development of a novel open queueing network model (QNM) at equilibrium for the performance modelling and analysis of an OpenStack IaaS CCP. Moreover, it was assumed that an external arrival pattern is Poisson while the queueing stations provided exponentially distributed service times. Based on Jackson’s theorem, the model was exactly decomposed into individual M/M/c (c ≥ 1) stations. Each of these queueing stations was analysed in isolation, and closed-form expressions for key performance metrics, such as mean response time, throughput, server (resource) utilisation as well as bottleneck device were determined.

Moreover, the research was extended with a proposed open QNM with a bursty external arrival pattern represented by a Compound Poisson Process (CPP) with geometrically distributed batches, or equivalently, variable Generalised Exponential (GE) interarrival and service times. Each queueing station had c (c ≥ 1) GE-type servers. Based on a generic maximum entropy (ME) product form approximation, the proposed open GE-type QNM was decomposed into individual GE/GE/c queueing stations with GE-type interarrival and service times. The evaluation of the performance metrics and bottleneck analysis of the QNM were determined, which provided vital insights for the capacity planning of existing CCP architectures as well as the design and development of new ones. The results also revealed, due to a significant impact on the burstiness of interarrival and service time processes, resulted in worst-case performance bounds scenarios, as appropriate.

Finally, an investigation was carried out into modelling and analysis of performance and security trade-offs for a CCP architecture, based on a proposed generalised stochastic Petri net (GSPN) model with security-detection control model (SDCM). In this context, ‘optimal’ combined performance and security metrics were defined with both M-type or GE-type arrival and service times and the impact of security incidents on performance was assessed. Typical numerical experiments on the GSPN model were conducted and implemented using the Möbius package, and an ‘optimal’ trade-offs were determined between performance and security, which are crucial in the SLA of the cloud computing services.
DECLARATION

The candidate confirmed that the research work presented in this thesis has been carried out by himself and has not been submitted for the award any other degree. Wherever other information retrieved from other sources, appropriate acknowledgement has been indicated in this thesis. Also, some part of chapters 2, 3, 4 and 5 are either published or under review for publication as stated in the list of publications section.

Kabiru Muhammad MAIYAMA
ACKNOWLEDGEMENTS

All praise be to Allah, the Lord of the Universe, the most gracious, and the most merciful. May His peace and blessings go to His noble Messenger, Prophet Muhammad SAW.

I sincerely wish to express my profound gratitude to my supervisor, Professor Demetres Kouvatsos, who worked round the clock and provide not just academic but personal advice, guidance, and support. I will be indebted to his invaluable assistance. I also like to appreciate my other supervisors, Dr Mariam Kiran, and Dr Mumtaz Kamala, for their help. Dr Mariam if not for you, I will not be in Bradford, you helped me to get in, guide through my stay, and continue to support me even after you left the university.

Similar appreciation goes to my examiners Professor Karim Djemame – University of Leeds, Professor Irfan-Ullah Awan and Dr Diana Wood who contributed with their comment, suggestions and feedback which has significantly improved the quality of this thesis.

My invaluable thanks go to my late parent, grandparent and uncles, Alhaji Muhammad Nata’ala Maiyama, Hajiya Amina, and Alhaji Atiku Maiyama, Alhaji Abdullahi Maiyama. May Almighty Allah send mercy and blessings to your graves as you had compassion on me when I was little. You raised me to the point of pursuing this degree and know you are proud of this achievement even though you are not here with us to witness it. Special thanks go to my living uncles and aunts such as Dr Faruku Abdullahi Maiyama, Alhaji Malami Abdullahi Maiyama, Dr Buhari Abdullahi Maiyama, Alhaji Aliyu Abdullahi Maiyama, Alhaji Samaila Abdullahi Maiyama, Alhaji Farouk Ahmed Bashar, Hajiya Dr Amina Shamaki and all other Uncles and Aunts who are too many to mention here.

A big thanks to my brothers, sisters, cousins such as Hajiya Aisha Muhammad Maiyama, Alhaji Adamu Muhammad Maiyama, Alhaji Dr Abubakar Atiku Muslim, Dr Nurudeen Almustapha, Dr Aminu Mustapha, Alhaji Aminu Ibrahim Gandhi, Dr Abubakar Garba and all other family members that help with support, prayers, and money, Jazakallah to all of you.
Especial thanks to my beloved wife, RN Suwaiba Muhammad Musa, and my children Nana Aisha, Lubaba, Khalid, and Mahmud, who witnessed this PhD journey and supported me with lots of sacrifices, prayers, and care. I love you all more than you can imagine and will continue to love you forever.

My appreciation also goes to other university staff and my colleagues during the study. Same courtesy goes to the only Dr Bashir Mohammed, Dr Anita Namanya, late Dr Neelkamal Shah, Dr Ejike Chuku, Dr Esmaeil Habib Zadeh, Dr Abimbola Sangodoyin Dimeji, Dr Babagana Modu, Dr Sadiq Sani, Dr Abubakar Dahiru and others who are too many to mention here. Similar appreciation goes to my colleagues at Usmanu Danfodiyo University, Sokoto such as Professor L.S. Bilbis, Professor Aminu Mohammed, Dr Abubakar Roko, Dr Abba Almu, Dr Aminu Bui, Dr Ahmed Y. Tambuwal, Dr Ahmad I. Tambuwal, Dr Habiba Sani and others that are too many to mention here.

Finally, I would like to acknowledge and thank the government of Nigeria through the petroleum technology development fund (PTDF) for a full scholarship to pursue this degree. Especial thanks to the amazing staff who help me during difficult times. Similar thanks go to my employer Usmanu Danfodiyo University, Sokoto for allowing me to go on study leave and continue to support me throughout the journey.
LIST OF PUBLICATIONS

Journals


Conference


Kiran M, Maiyama K., Mir H., Mohammad B., Al-Oun A.

Workshop


Technical Report


Poster


Presentations


# LIST OF ACRONYMS AND NOTATIONS

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>c.f.</td>
<td>Refer to</td>
</tr>
<tr>
<td>CCP</td>
<td>Cloud Computing Platform</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CPP</td>
<td>Compound Poisson Process</td>
</tr>
<tr>
<td>CPSM</td>
<td>Combined performance security model</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Supported Platform</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DTN</td>
<td>Delay Tolerant Network</td>
</tr>
<tr>
<td>EC2</td>
<td>Elastic Cloud</td>
</tr>
<tr>
<td>FCFS</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>fiCloud</td>
<td>Future Internet of Things and Cloud</td>
</tr>
<tr>
<td>GE</td>
<td>Generalised Exponential</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>$H_2$</td>
<td>Hyperexponential-2</td>
</tr>
<tr>
<td>HA</td>
<td>High Availability</td>
</tr>
<tr>
<td>HDD</td>
<td>Hard Disk Drive</td>
</tr>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>JMT</td>
<td>Java Modelling Tool</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Matric Laboratory</td>
</tr>
<tr>
<td>ME</td>
<td>Maximum Entropy</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MQL</td>
<td>Mean Queue Length</td>
</tr>
<tr>
<td>M-type</td>
<td>Markovian (Memoryless) type</td>
</tr>
<tr>
<td>NetPEn</td>
<td>Networks Performance Engineering</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PM</td>
<td>Physical Machine</td>
</tr>
<tr>
<td>PN</td>
<td>Petri Nets</td>
</tr>
<tr>
<td>PTDF</td>
<td>Petroleum Technology Development Fund</td>
</tr>
<tr>
<td>QNM</td>
<td>Queueing Network Model</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RESTful</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>RLP</td>
<td>Request Lost Probability</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SCV</td>
<td>Squared Coefficient of Variation</td>
</tr>
<tr>
<td>SDCM</td>
<td>Security Detection Control Model</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Network</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SPN</td>
<td>Stochastic Petri Net</td>
</tr>
<tr>
<td>SRN</td>
<td>Stochastic Reward Net</td>
</tr>
<tr>
<td>UKPEW</td>
<td>United Kingdom Performance Engineering Workshop</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VMM</td>
<td>Virtual Machine Monitor</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>LXC</td>
<td>Linux Container</td>
</tr>
<tr>
<td>Notations</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>$P_n$</td>
<td>Probability of having $n$ request in the system</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Station Utilization</td>
</tr>
<tr>
<td>$TT\hat{\diamond}$</td>
<td>Mean Response time</td>
</tr>
<tr>
<td>$\hat{\diamond}$</td>
<td>Mean Waiting time</td>
</tr>
<tr>
<td>$M\hat{\diamond}$</td>
<td>Mean Number of requests in the station</td>
</tr>
<tr>
<td>$Q\hat{\diamond}$</td>
<td>Mean Number of requests in the queue</td>
</tr>
<tr>
<td>$\lambda\pi$</td>
<td>Mean System Throughput</td>
</tr>
<tr>
<td>$pp_{iii}$</td>
<td>Transition (routing) Probabilities</td>
</tr>
<tr>
<td>$\lambda\lambda$</td>
<td>Mean external arrival rate</td>
</tr>
<tr>
<td>$\Lambda_{ii}$</td>
<td>Mean station arrival rate</td>
</tr>
<tr>
<td>$\mu_{ii}$</td>
<td>Mean station service time</td>
</tr>
<tr>
<td>$\Lambda_{ddii}$</td>
<td>Mean station departure rate</td>
</tr>
<tr>
<td>$CC^2_{ua}$</td>
<td>External arrival squared coefficient variation (SCV)</td>
</tr>
<tr>
<td>$CC^2_{umi}$</td>
<td>Mean station arrival SCV</td>
</tr>
<tr>
<td>$CC^2_{ssii}$</td>
<td>Mean station service time SCV</td>
</tr>
<tr>
<td>$CC^2_{uddii}$</td>
<td>Mean station interdeparture time SCV</td>
</tr>
<tr>
<td>$CC^2_{ddii}$</td>
<td>Mean splitting interdeparture time SCV</td>
</tr>
<tr>
<td>$CC^2_{TT}$</td>
<td>Mean system departure SCV</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

ABSTRACT .......................................................................................................................... i
DECLARATION .................................................................................................................... iii
ACKNOWLEDGEMENTS ....................................................................................................... iv
LIST OF PUBLICATIONS ..................................................................................................... vi
LIST OF ACRONYMS AND NOTATIONS ............................................................................... viii
TABLE OF CONTENTS ....................................................................................................... xi
LIST OF TABLES ................................................................................................................... xiv
LIST OF FIGURES ................................................................................................................ xv

1 INTRODUCTION ............................................................................................................... 1
  1.1 Problem Definition and Motivation .............................................................................. 1
  1.2 Research Aims and Objectives ...................................................................................... 2
  1.3 Thesis Contributions .................................................................................................... 4
  1.4 Thesis Structure ........................................................................................................... 5

2 BACKGROUND AND LITERATURE REVIEW ................................................................. 7
  2.1 Introduction .................................................................................................................. 7
  2.2 Background on Relevant Technologies and Tools ...................................................... 7
    2.2.1 Virtualisation .......................................................................................................... 7
    2.2.2 Cloud Computing .................................................................................................... 10
    2.2.3 Open Source Cloud Computing Platforms ........................................................... 13
      2231 OpenNebula Cloud Computing Platform ............................................................... 13
      2232 Eucalyptus Cloud Computing Platform ................................................................. 14
      2233 OpenStack Cloud Computing Platform .................................................................. 14
      2234 Detailed Messaging Flow for OpenStack VM Provisioning Request .................. 16
  2.3 System and Software Testing ....................................................................................... 19
    2.3.1 Cloud Testing ......................................................................................................... 20
    2.3.2 Cloud Testing Ecosystem a Systematic Mapping Study ......................................... 21
  2.4 Performance Modelling, Analysis and Techniques ...................................................... 32
    2.4.1 Measurement ......................................................................................................... 34
    2.4.2 Simulation ............................................................................................................... 36
    2.4.3 Analytical .............................................................................................................. 37
      2431 Queueing Theory and Queueing Network Model .................................................. 37
      2432 Petri Nets ............................................................................................................... 39
  2.5 A Literature Review on Performance and Security Trade-off Models ....................... 42
# Table of Contents

## Chapter 2

2.6 Research Design and Methodologies .......................................................... 45

## Chapter 3

3.1 Introduction .................................................................................................... 48

3.2 M-Type QNM for Performance Modelling and Analysis ............................... 49

3.3 M/M/1 and M/M/c QNM for VM Provisioning Request in OpenStack IaaS CCP .................................................................................................................. 51

3.3.1 Performance Modelling and Analysis ...................................................... 54

3.3.2 Modelling and Analysis of the Stable M/M/c Queueing Stations 2, and 4.56

3.3.12 Analysis of Stable M/M/1 Queueing Stations 1, 3, 5 and 6 ................... 57

3.4 Numerical Experiments and Results ........................................................... 57

3.4.1 Bottleneck Analysis .................................................................................. 60

3.5 Summary ....................................................................................................... 61

## Chapter 4

4.1 Introduction .................................................................................................... 62

4.2 Maximum Entropy (ME) Principle .................................................................. 64

4.3 Generalised Exponential (GE) Distribution .................................................. 66

4.3.1 Matching $H_2$ as GE Distribution .......................................................... 68

4.4 A Stable Open GE-type QNM Model for VM Provision in OpenStack IaaS CCP .......................................................................................................... 70

4.5 The Proposed Stable Open GE-type QNM for OpenStack IaaS CCP .......... 72

4.5.1 Traffic Flow Analysis of the Stable Open GE-type QNM ....................... 75

4.5.2 Analytical Computation of GE/GE/$c_i$ ($c_i \geq 1$) ................................ 79

4.5.3 The Computational Algorithm for the Analysis of GE-type QNM .......... 80

4.6 Numerical Experiments and Results ............................................................ 81

4.6.1 Bottleneck Analysis .................................................................................. 87

4.7 Summary ....................................................................................................... 88

## Chapter 5

PERFORMANCE AND SECURITY TRADE-OFFS IN OPENSTACK IaaS CCPs .................................................................................................................. 90

5.1 Introduction .................................................................................................... 90

5.2 Computer and Information System Security Overview .............................. 91

5.3 Security Metrics ............................................................................................ 94

5.4 Performance Metrics .................................................................................... 95

5.4.1 External .................................................................................................... 96
LIST OF TABLES

Table 2.1: Search string keywords and synonyms .................................................23
Table 2.2: Summary of fora that publishes three or more papers .........................28
Table 2.3: Summary of articles classification based on testing on each cloud service layer ........................................................................................................30
Table 3.1: Performance metrics, notations and definitions ....................................50
Table 3.2: Numerical values of routing probabilities and mean service rates for the open QNM of Figure 3.3 .................................................................58
Table 4.1: Numerical values of Routing probabilities, Mean service rates and SCVs for the Open GE-type QNM (c.f., Figures. 4.12 – 4.18) ........82
Table 4.2: Values used to Match H₂ to GE for GE-type QNM analysis of OpenStack IaaS CCP ..............................................................83
Table 5.1: Assigned values for the parameters of the Trade-off Model .................107
Table C.1: Table of values (k = 10) used to Match H₂ to GE for GE-type analysis of OpenStack IaaS CCP .........................................................140
Table C.2: Table of values (k = 50) used to Match H₂ to GE for GE-type analysis of OpenStack IaaS CCP .........................................................140
Table C.3: Table of values (k = 100) used to Match H₂ to GE for GE-type analysis of OpenStack IaaS CCP .........................................................141
LIST OF FIGURES

Figure 2.1: Full or native virtualisation ................................................................. 8
Figure 2.2: Hardware assisted virtualisation ............................................................ 8
Figure 2.3: Paravirtualisation .................................................................................. 9
Figure 2.4: OS level virtualisation ......................................................................... 9
Figure 2.5: Cloud Computing System [2] .................................................................. 10
Figure 2.6: Cloud Computing Service model layers ................................................. 11
Figure 2.7: OpenStack CCP services [40] ................................................................. 15
Figure 2.8: Typical OpenStack messaging flow for VM instance provision
  provisioning request .................................................................................................. 17
Figure 2.9: Systematic Mapping Studies phases ....................................................... 22
Figure 2.10: Detail article selection process ............................................................... 24
Figure 2.11: Articles classification based on research type ...................................... 26
Figure 2.12: Relevant publications per year ............................................................... 27
Figure 2.13: Authors with their number of publications .......................................... 28
Figure 2.14: A simple queueing system .................................................................. 38
Figure 2.15: Petri Net with typical components ....................................................... 40
Figure 3.1: M/M/1 and M/M/c queues with arrival and service time symbols ....... 50
Figure 3.2: Typical OpenStack private cloud setup ................................................. 52
Figure 3.3: Schematic diagram of the new open QNM for OpenStack VM
  provisioning request ............................................................................................... 53
Figure 3.4: General state transition rate for birth-death process............................ 55
Figure 3.5: M/M/c queueing station with arrival rate λ and service time μ .......... 55
Figure 3.6: Overall system response time (𝑊) vs. λλλ .............................................. 59
Figure 3.7: Mean number of request in the system vs λλλ ........................................ 59
Figure 3.8: Mean system waiting time ( hdr ) vs λλ ................................................ 59
Figure 3.9: Mean number of requests in the queue (hdr ) vs λλ ................................ 59
Figure 3.10: Server utilisation at station 1: Firewall vs. λλ ...................................... 60
Figure 3.11: Server utilisation at station 2: Controller vs. λλ ................................... 61
Figure 3.12: Server utilisation at station 5: Storage vs. λλ ...................................... 61
Figure 4.1: The GE-type distribution with parameters τ and σ (0 ≤ τ ≤ 1) (c.f.,
  [23]) ....................................................................................................................... 67
Figure 4.2: Batch arrivals according to the GE distribution [24] ......................... 67
Figure 4.3: Hyperexponential distribution with phases ......................................... 69
Figure 4.4: Approximating (Matching) H2 to the GE distribution ....................... 70
Figure 4.5: Typical OpenStack private CCP setup ............................................... 71
Figure 4.6: Schematic diagram for an open GE-type QNM of OpenStack VM
  provisioning requests with cc1, cc4, cc5 = 1; cc2, cc3 > 1 ................................... 72
Figure 4.7: Decomposition of the stable open GE-type QNM into individual
  GE-type queueing stations ...................................................................................... 74
Figure 4.8: The departure process of heavy traffic approximation for stable 
\text{GE/GE/}\text{ccu} (\text{ccu} \geq 1) queueing station ii, ii = 1, 2, ..., 5 .............................. 77
Figure 4.9: Stream splitting at nodes (c.f., Split\textsubscript{1} and Split\textsubscript{3} in Figure 4.6) ............... 77
Figure 4.10: Streams merging at node (c.f., Merge\textsubscript{1, 5} and Merge\textsubscript{2, 4} in Figure 4.6) ................................................................. 78
Figure 4.11: Mean number of requests (\(K]\text{K}\)) in firewall (station 1) with different values of k vs. \(\lambda\lambda\)................................................................. 83
Figure 4.12: Mean number of requests (\(K]\text{K}\)) in controller (station 2) of different values of k vs. \(\lambda\lambda\)................................................................. 84
Figure 4.13: Mean number of requests (\(K]\text{K}\)) in compute (station 3) with different values of k vs. \(\lambda\lambda\)................................................................. 84
Figure 4.14: Mean number of requests (\(K]\text{K}\)) in storage (station 4) with different values of k vs. \(\lambda\lambda\)................................................................. 84
Figure 4.15: Mean number of requests (\(K]\text{K}\)) in output (station 5) with different values of k vs. \(\lambda\lambda\)................................................................. 85
Figure 4.16: Mean system throughput (\(\lambda\lambda TT\)) with different values of k vs. \(\lambda\lambda\)............... 85
Figure 4.17: Mean system response time (\(TT\)) with different values of k vs. \(\lambda\lambda\)............. 86
Figure 4.18: Mean system response time (\(TT\)) with different values of SCV vs. \(\lambda\lambda\).............................. 86
Figure 4.19: Min and max utilisations \{\(\rho\rho\), ii = 1, 2, ..., 5\} for all five stations 
with different values of k........................................................................ 87
Figure 5.1: Cloud Security Vulnerabilities [157]......................................................... 93
Figure 5.2: Security Metric by analogy with dependability metric [131]..................... 95
Figure 5.3: Schematic diagram for an open GE-type QNM of OpenStack VM 
provisioning requests with propose decryption and encryption 
extension ........................................................................................................... 98
Figure 5.4: Schematic diagram of combined performance and security trade- 
off analysis model [127]............................................................................. 99
Figure 5.5: Security detection control model for CPSM............................................. 101
Figure 5.6: Implemented CPSM using Mobius SAN............................................. 104
Figure 5.7: M-type CPSM1............................................................................. 108
Figure 5.8: M-Type CPSM2............................................................................. 108
Figure 5.9: M-Type, \(H2(k=2,10,50,100\geq GE)\) CPSM1............................................. 109
Figure 5.10: M-Type, \(H2(k=2,10,50,100\geq GE)\) CPSM2............................................. 110
Figure B.1 A parameterised diagram of a departure process of heavy traffic 
approximation stable GE/GE/c (c \geq 1) queueing station with the 
associated interarrival times service times and interdeparture 
times with parameters \((\lambda\lambda_{\text{ii}}, \text{CC}^2), (\text{cc}_{\text{ii}}, \text{CC}^2)\) and \((\lambda\lambda_{\text{ddii}}, \text{CC}^2),\) 
respectively............................................................................................................. 137
Figure B.2 A parameterised diagram of the splitting stream with mean 
departure rate, \(\lambda\lambda_{\text{ddii}}\) and the SCV of the interdeparture times, \(\text{CC}^2_{\text{ii}}\)
stream and resulting effective splitting streams each with its associated transition probability $p_{ii}, ii = 1, 2, ..., LL; jj = 0, 1, ..., LL$ with splitting departure rates $\lambda_{addit}$ and SCV of interdeparture times, $C_{addit}^2, jj = 0, 1, 2, ..., LL$.

Figure B.3: A parameterised diagram of the overall merging stream at each queueing station $i, (i = 1, 2, ..., L)$ is comprised from the superposition of i) external arrivals with mean rate $\lambda_{0i}$ and SCV of external interarrival times, $C_{0i}^2$ and ii) internal splitting of interdeparture times with mean rates $\lambda_{i}i$ and SCVs, $C_{i}^2, j = 1, 2, ..., L$ with associated transition probabilities $p_{i}i, j = 0, 1, 2, ..., L$ stream and iii) an overall merged stream with mean overall merged arrival rate, $\lambda_{ii}$ and SCV, $C_{ii}^2$. 
1 INTRODUCTION

1.1 Problem Definition and Motivation

Cloud computing is a computing model that delivers computing resources in the form of service rather than a traditional product using multitenant and utility model [1, 2]. These services ranging from compute-cycle, data storage, and network, among others, are defined in well written documents known as service level agreements (SLAs). The documents describe the rules and expectations from both parties, especially the providers [3, 4]. Developing and improving SLA is becoming crucial as more organisations migrate their data, application and even systems to the cloud [3, 5]. Some of the critical components of the SLA are the quality-of-service (QoS), reliability, availability, security and privacy, among others. Cloud SLA warrants cloud service providers to meet specific enterprise-level requirements and a clearly defined set of deliverables for their customers. Also, the defined SLA should be measurable and specified in each service. In Infrastructure as a service (IaaS), the SLA should include many components as it involves the whole computing resources in the form of virtual machines (VMs) [6].

Cloud computing performance analysis and optimisation are some of the areas that are gaining attention as the technology enters its maturity level. Performance analysis of cloud computing platforms (CCPs) will help in evaluating the QoS as cloud IaaS is built on top of virtualisation technology. Cloud service providers claim to offer secure, scalable, high performance, and high availability (HA) services to their clients. However, many reported performance issues are questioning the QoS of cloud systems. HA is one of the exciting attributes of QoS with focus on ensuring 99.999% availability, and thus, performance is a crucial attribute of achieving it [7, 8]. For example, provisioning of cloud resources in the form of VM for different kinds of workloads depends on the QoS requirements defined in the SLA. Based on users' specification of the required resources, the identification and assignment of an optimal workload-resource pair are of paramount importance in achieving acceptable levels of workload performance [9]. Also, being able to evaluate the
performance of individual components within CCP on both normal and peak workload at a design stage is of significant importance, mainly during capacity planning and design. Thus addressing these kinds of problems will be welcome by a community of cloud engineers. These issues attracted global attention within research communities of both industry and academia, in which only a few works reported progress in performance and other QoS related problems [10].

This research work is motivated by the need to devise new performance evaluation methods as well as the application of existing techniques of the virtualisation in IaaS CCPs. It is based on the application of queueing theory and open queueing network models (QNM) for the analysis of OpenStack IaaS in order to identify CCP’s bottleneck devices and carry out performance prediction and capacity planning towards ‘optimal’ performance vs security trade-offs. Although there have been many works using analytical models such as QNMs [11-13], generalised stochastic Petri nets (GSPNs) [14-18], and those based on performance evaluation process algebra (PEPA) [19, 20]. However, none of these works employs batch arrival processes, such as a compound Poisson process (CPP) with the generalised exponential (GE-type) interarrival times [21-24].

Furthermore, as CCP is composed of a network of computers and related devices, approximate quantitative and simulation techniques for the analysis of open QNMs and GSPNs have already applied for the performance evaluation the of CCPs [25, 26]. In particular, using discrete-event simulation tools can be effectively applied for the validation of approximate analytic solutions. In this context, this thesis addresses the most crucial problem of performance modelling and evaluation solutions for CCPs and the identification of best and worst performance scenarios in CCPs.

### 1.2 Research Aims and Objectives

The main aims of this thesis is to investigate the issues of performance engineering and ‘optimal’ performance-security trade-offs in CCPs through analytical modelling and evaluation using state of the art analytic techniques and associated numerical experiments. Moreover, the objectives of this research are:
1. To carry out an extensive and critical literature review on non-functional software testing, performance testing, performance modelling and analysis subject to adopted techniques, quantitative methods, and state of the art tools in the fields of the thesis. To this end, knowledge gaps in this domain will be identified and form the foundation of this research. In this context, a systematic review and mapping in the area of cloud testing, in general, and performance modelling and analysis of CCPs, in particular, will be undertaken in conjunction with the identification of ‘optimal’ performance vs security trade-offs in CCPs. Most of the components, as mentioned earlier, are addressed in Chapter 2.

2. To evaluate the performance of CCPs using existing analytic techniques incorporating both exact and approximation metrics such as platform’s mean response time of a service request, system utilisation, system throughputs as well as individual node’s/station’s utilisation thereby identifying the bottleneck node(s)/station(s). These metrics will aid towards credible engineering and tuning of the platform’s performance by CCP’s engineers at the provider’s side. This objective is addressed in Chapter 3.

3. To extend the analytic modelling and evaluation described in the narrative of objective 2, using the generalised exponential (GE) distribution with compound Poisson (batch) arrival process, towards the characterisation of ‘worst case’ (pessimistic) performance metrics. This objective is dealt with in Chapter 4.

4. To determine an ‘optimal’ performance and security trade-offs model as part of the performance engineering and tuning of a CCP, which will provide an insight into the impact of security on performance and vice versa. This will help IaaS cloud service providers as well as service users with the potential enhancement of SLA. This objective is addressed in Chapter 5.
1.3 **Thesis Contributions**

This thesis has contributed to the body of new knowledge in the performance modelling and analysis of virtualisation in CCPs, including the identification of ‘optimal’ performance vs security trade-offs. In this context, the knowledge gap was extended for the future work on cloud testing and performance modelling and evaluation using both exponential and GE-type distributions as well as defining ‘optimal’ trade-offs between performance and security of virtualisation in IaaS CCP. More details on the contributions of the thesis are stated below:

a) A systematic mapping study relates to a research publication in the area of cloud testing, in general, and performance testing and analysis in cloud computing, in particular. The study identified several articles, which reported research in different layers of service model. It transpired that IaaS had received substantive attention, however, the quantitative evaluation of an example CCP was not adequately investigated. The latter provides the motivation to follow up IaaS in CCPs and address the problem.

b) Developed a new stable open QNM consisting of single and multiple server queueing stations and focusing on the virtualisation in OpenStack IaaS CCP. The model was analysed using M/M/c, (c ≥ 1) and some key performance metrics were computed. Numerical experimentations for the quantitative determination of performance metrics, such as server utilisation, mean response time and utilisation were conducted. The model also determined the bottleneck node within the platform. The model serves as a guideline for the prediction of the system’s behaviour, especially at the design stage as well as during tuning and engineering of the existing CCPs.

c) Adopted the typical setup architecture of OpenStack CCP from the contribution described in paragraph b) above and developed an innovative open GE-type QNM at equilibrium with random routing and a First-Come-First-Served (FCFS) rule. The model uses bursty processes for both external arrival and service times for the virtualisation in OpenStack CCP according to a CPP with geometrically distributed batches. A maximum entropy (ME) product form
approximation was adopted to decompose the platform into individual queuing stations, each of which can be analysed in isolation. Key performance metrics were also computed such as throughput, utilisation, mean platform response time and determination of bottleneck station(s) of a typical snapshot of queues within the CCP. This work may serve as a guideline for new and more realistic performance prediction studies as well as the capacity planning of new infrastructure. Moreover, it can be employed for the tuning/upgrading of existing CCP architectures for the efficient delivery of HA service during the provisioning and allocation of VMs instance.

d) The last contribution of this thesis is the development of a performance and security trade-off model for the same OpenStack IaaS CCP during the processing and creation of VMs. A combined performance and security trade-offs model was developed, analysed and implemented. Results show the impact of security on performance at both normal and during bursty bulk arrival patterns.

1.4 Thesis Structure

The structure of this thesis is scheduled as follows: Chapter 2 provides a background of relevant themes such as virtualisation, cloud computing platforms such as OpenNebula, Eucalyptus and OpenStack CCPs. Moreover, it provides the narratives of a critical literature review and the method used in conducting the review and the findings of the knowledge gap identified. Chapter 3 presents the first contribution of the thesis on ‘the modelling and analysis of OpenStack IaaS CCP using an M-type QNM’. This includes the modelling of the architectural topology of the typical setup of the platform, which is based on the bottleneck analysis of an M-type stable open QNM of a CCP and the computation of relevant performance metrics. Chapter 4 follows with the extension of the M-type QNM of Chapter 3, to a stable open GE-type QNM and its station-by-station decomposition analysis and associated numerical results. Chapter 5 presents the study that investigates the relationship between security and performance, specifically on ‘how performance is affected by a security mechanism’. A performance security trade-off model was
devised, and the SDCM was used for the analysis. Numerical results were also presented, interpreted, and discussed, as appropriate. The thesis is concluded in Chapter 6, which also includes suggestions for future work. The thesis also contains a list of references, which itemised the articles cited. It also presents the Appendices of relevant algorithms and techniques, as appropriate.
2 BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter provides background information on relevant terminologies, technologies, and domains. It starts with essential background on virtualisation technology, cloud computing, and OpenStack IaaS CCP. The chapter also describes a brief on the parent area of system and software testing as well as cloud testing. In addition, it presents an extract from an earlier study of literature using systematic mapping studies (SMSs). The SMS entails the stages of the study and some results. Moreover, the chapter presents literature on performance modelling and analysis, including the three main techniques of measurement, simulation, and analytical approaches to performance analysis as well as some tools.

Next section provides a background on technologies and tools relevant to this research.

2.2 Background on Relevant Technologies and Tools

2.2.1 Virtualisation

Despite the complex nature of computer systems, their evolution continues drastically because of the different hierarchical layers and well-designed interfaces providing various levels of abstraction. These loosely coupled interfaces enable the isolation and independence of the system’s components development by various teams such as hardware, software, and services. In addition, the hierarchical architecture of systems and services help to develop more abstraction, such as virtualisation [40]. The virtualisation of CCPs refers to the provisioning of virtual rather than the traditional physical computing resources such as CPU, memory, storage, and network in virtual form. The physical machine (host) is the one that shares its resource by creating guest (virtual) machines. Even though virtualisation is an old technology that rooted since the time of mainframe computers in the 1960s, it allows distributed computing models without creating dependencies on physical resources [27, 28].
Virtualisation can also be achieved using different types of abstraction. These include but not limited to; full or native, hardware assistive and paravirtualisation, among others [28, 29].

**Full Virtualisation:** In full (also known as Native) virtualisation (c.f., Figure 2.1), the VM simulate enough hardware to allow running of an unmodified ‘guest’ OS in isolation. A typical example of the native virtualisation tools are VirtualBox, Virtual PC, VMWare, and Qemu.

![Figure 2.1: Full or native virtualisation](image)

**Hardware-Assisted Virtualisation:** In this type of virtualisation (c.f., Figure 2.2), VM has its own physical hardware such as the processors, which supports the host OS capability of allowing a guest OS to run in isolation. Intel virtualisation technology (IVT), AMD virtualisation (AMD-V), Parallel Workstation and Parallel Desktop for Mac are some of the examples of virtualisation achieved with the aid of hardware.

![Figure 2.2: Hardware assisted virtualisation](image)

**Paravirtualisation:** In paravirtualisation technique (c.f., Figure 2.3), the VM does not necessarily simulate hardware, but instead (or beside) offers a specific API that can only be used to modify the ‘guest’ OS. A successful paravirtual mechanism is the
Cambridge University project for the development of the technique to support HPC, which resulted in the Xen platform [30].

Operating-System-Level Virtualisation: OS-level virtualisation (c.f., Figure 2.4) enables virtualising a physical server at the OS level, allowing multiple isolated and secure virtualised servers to run on a single physical server. Linux Vserver, Solaris containers, Linux containers (LXC) [31], and Dockers are examples of OS-level virtualisation.

The success of cloud computing was achieved because of virtualisation technology, thereby allowing the renting of computing resources from a pool in the form of a multitenant utility model. As these resources reside inside and in most cases, shared hardware, QoS such as performance and security are the major issues affecting the SLA [6, 32]. In its basic form, VM is created in a single physical machine (PM). However, in most of the existing CCPs, the VMs are created from a shared PMs linked through complex layers of networking with very high-speed links that allow data transfer rate similar to the bus-links in computer’s system unit. This enables the deployment of compute cycle, storage, and networking component from
different PMs. In OpenStack CCP architectures, for example, various components reside in separate PM, thereby making its deployment monotonous as well as provisioning of its resources from many hardware puzzling. Furthermore, deployment options add to the complexity as it can be deployed using both Virtual Machine Monitor (VMM) otherwise known as a hypervisor or using OS-based virtualisation that is mainly achieved using containers such as LXC and Dockers [6, 28].

### 2.2.2 Cloud Computing

Cloud Computing is a buzz phrase that represents a form of computing that provides information technology resources (hardware and software) in the form of a utility service rather than the traditional products or licencing of products [1]. The most widely used definition adopted is the one by United State National Institute of Standards and Technology (NIST) which defines it as: “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [2]. This definition is adopted because it represents the typical characteristics of the cloud.

![Figure 2.5: Cloud Computing System](http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html)
Clouds deployment models are grouped into Public, Private, Hybrid and Community clouds (c.f., Figure 2.5).

- **Public Clouds**: Here, end-users can access commercially available cloud infrastructures at any of the three layers for their needs. The cloud provider is responsible for managing the systems and catering to client demands. End-users are only responsible for the service they are using. Connections to a public cloud are through the internet and involve handling multiple clients simultaneously.

- **Private Clouds**: In private clouds, systems and resources are hosted inside the premises of the organisations’ using their services. The organisation itself is liable for the software and the infrastructure connected through a local area network (LAN).

- **Hybrid Clouds**: The deployments in the hybrid cloud is usually a mixture of public and private cloud models, with multiple organisations involved. The flexibility of the usage depends on permissions and rights offered to end-users.

- **Community Clouds**: Community clouds are collaborative sharing of infrastructure between two or more organisations, both private and government agencies with a common mission. However, this model creates concerns as the control and management are usually through a third party or one of the collaborating organisations. Resources are lease to only the community members alone.

![Cloud Computing Service model layers](image)

Figure 2.6: Cloud Computing Service model layers
Clouds are further characterised using business or service models. The main service models are (c.f., Figure 2.6):

**Software as a Service (SaaS):** This is a cloud service model that provides software applications and databases service through a thin client platform such as web browser, console or application programming interface (API). The underlying hardware, operating system and platform all reside at the premises of the provider sites and is managed and controlled by the providers. User data is the only component of the stack layers that user controls. In the SaaS model, users have no control over the application they are using, such as installing components, or add-ons, configuring the application, and so on. A typical example of SaaS is a web-based email such as Gmail or Office 360, customer relationship management (CRM) software and gaming applications.

**Platform as a Service (PaaS):** This is also a cloud service model that mainly provide resources for platform and software development, collaboration suites such as development environments, programming editors, as well as APIs. A typical PaaS service is Google App Engine, Microsoft Azure platforms such as a Visual Studio.

**Infrastructure as a Service (IaaS):** This is a service model that provides primary offerings of computing infrastructure resources such as servers, hosts, networks and virtualisation as mainly (virtual machines). IaaS also referred to online services that abstract the user from the details of infrastructure such as physical computing resources, location, data partitioning, scaling, security, backup, among others. These resources are mainly in the form of VM rather than physical infrastructure. The provision of VM is achieved through a hypervisor, such as Xen, VirtualBox, and kernel virtual machine (KVM). Although providers control the infrastructure, clients have the liberty to deploy their operating system of choice from a range of available OS image from the provider or their customised one. Also, IaaS clouds are both proprietary and open source. Amazon EC2, Microsoft Azure, and RackSpace are some of the popular providers of IaaS. IaaS is the service that provides the highest level of cloud resource layer in terms of complete control of the hardware through virtualisation. The companies mentioned above offer proprietary tools, while other open source tools such as OpenStack are also available.
2.2.3 Open Source Cloud Computing Platforms

Open source Cloud computing platforms refer to the publicly available software application that enables the creation, management and complete control of cloud computing resources that are free to access and modify as needed. A number of research articles provide a proper assessment of the different CCPs with various features that distinguishes them from each other, which made some more favourable for research. For example, more recently, some user feedback and comparison website ¹ shows that OpenStack supports more followers, stackers and receive higher votes among developer communities. Moreover, Joshi et al. [33] present a structured overview of the different services offered by both commercial and open source CCP solutions with OpenStack, OpenNebula and Eucalyptus among the top open-source platforms. Most of the services by both commercial vendors such as Microsoft Azure, Google cloud platform (GCP) as well as open-source tends to mimic the AWS services which serve as the de facto benchmark of cloud service delivery. The research did not favour any open source solution, even though it reported more services in OpenStack than its other open-source competitors.

On the contrary, [34] provide another comparison which positioned OpenNebula with higher flexibility and support for data centre virtualisation. Nevertheless, the research presented in this thesis chose OpenStack CCP based on the initial comparative assessment [35-37] as well as the hands-on exploration of the platform which gives a better understanding of its internal structure, deployment options and its distributed and loosely coupled architecture.

Next subsections present a brief overview of a couple of open-source CCPs that provides virtual infrastructure management solutions including OpenStack CCP.

2.2.3.1 OpenNebula Cloud Computing Platform

OpenNebula is an open source CCP that enables the creation and management of computing clusters and data centre infrastructures. It serves as virtual infrastructure manager allowing the deployment of both public, private and

¹ https://stackshare.io/stackups/apache-cloudstack-vs-eucalyptus-vs-openstack
hybrid IaaS implementation. It also supports a wide range of virtualisation offerings such as KVM, LXC and VMWare among others. OpenNebula platform manages a data centre’s virtual infrastructure to build private, public and hybrid implementations of infrastructure as a service. Llorente and Monteros’ research project in 2005 gave birth to the OpenNebula, which serve as a toolkit for open source CCP accommodating multiple hardware and software. One of the main attributes of the platform is its support for an elastic platform for easy scalability and fast delivery of services to meet the dynamic expectation of service end-users [38]. The virtual machine hosts the services and is submitted, monitored and controlled in the cloud by using a virtual interface.

2.2.3.2 Eucalyptus Cloud Computing Platform

Designed to be compatible with AWS, Eucalyptus is another open source CCP software for provisioning of compute, storage and other related services on demand for both public, private and hybrid delivery model. It uses similar terminologies with AWS for describing services. Its simplicity limits its services in comparison with OpenNebula and OpenStack. However, it pays in its easy deployment and configuration. Eucalyptus is designed to target educational and research based organisations that wish to build their private CCP [35].

2.2.3.3 OpenStack Cloud Computing Platform

OpenStack is a cloud computing ecosystem platform that controls and manages a large pool of compute, storage, and network resources within a small, medium and large/complex network of computing clusters and datacentre. The platform provides IaaS resource for both private and public clouds. It has support for different hypervisors, live migration, load balancing and distributed architecture. The provisioning, management and control of OpenStack services are accessed via a web-based graphical user interface (GUI) dashboard (Figure 2.7), command-line-interface (CLI) and a third-party management tools [39].
Some of the principal service components of OpenStack relevant to this thesis (c.f., Figure 2.7) are briefly described as follows [40]:

- **Compute (Nova):** Nova provides compute services, which serve as a fabric controller, which is the main component of IaaS. It manages the pool of systems resources, supports all the major hypervisors, and supports high-performance computing (HPC) and bare metal configurations. Nova compute is responsible for spinning the VM instance and management of its life cycle.

- **Storage (Cinder and Swift):** Cinder provides persistent block storage service for use with compute instances. The creation, assignment, and management of block storage devices to the servers are all handled by the service. Swift, on the other hand, provides Object and files scalable storage service that writes data across multiple drives spread throughout the data centre for redundancy.

- **Networking (Neutron):** Neutron provides networking services such as management of IP address pool including DHCP floating IP assignment, firewall, load balancing, and VPN, among others. It also allows users to create their
network with the support of software-defined-network (SDN) technologies such as OpenFlow.

Other services include an identity service for authentication and authorisation of users and services; Image for the provision of operating system (OS) image to VMs and dashboard for GUI access to the services. As a primary component of OpenStack, Nova [41] is the computing fabric controller for the OpenStack cloud. It has various components that provide complementing services that are hosted on high specification servers such as Controller or Compute nodes depending on the network design. Nova compute is the most complicated and distributed component of OpenStack that facilitates the creation and orchestration of service request such as VM instance creation. The main components are nova-API, message-queue, nova-compute, nova-conductor, nova-volume, and its API’s. Other processes that help in turning user requests into VMs are Cinder, Glance, and Neutron, alongside their respective APIs and dependencies.

Nova interacts with many other OpenStack services depending on the user request and the physical cloud architecture used. Likewise, most of the communications are through message-queue, which serves as a central communication pathway between Nova components. Similarly, authentication of each request is handled by the Keystone service through authentication token (auth-token) passing forth and back.

**2.2.3.4 Detailed Messaging Flow for OpenStack VM Provisioning Request**

The following conceptual steps describe the processing of a user’s request for VM creation, subject to request type, OpenStack release, and a particular topology of the CCP. It is also subject to other factors such as OS type and system resource specification like Hard Disk Drive (HDD) volume, Random Access Memory (RAM) capacity, and some processing (CPU) cores (c.f., Figure 2.8). These conceptual steps were adopted from [42-45] and also itemised in the earlier published work [46].
2.2.3.4 Detailed Messaging Flow for OpenStack VM Provisioning Request

Figure 2.8: Typical OpenStack messaging flow for VM instance provision request

Conceptual Steps:

Step 1 – The client sends a request for VM provision through OpenStack CCP either dashboard or CLI or third-party API to the Restful service, which receives the request and forwards it to the Nova-API;

Step 2 – The Nova-API interacts with Nova-DB (i.e., Nova-database), which then creates the entry for the VM instance (including instance type and specifications);

Step 3 – The Nova-API sends the Remote Procedure Call (RPC) request to the Nova-Scheduler through Message-queue (which is visible through any messaging service’s management interface such as RabbitMQ\(^2\));

Step 4 – The Nova-scheduler interacts with Nova-DB to find the available and suitable host (the process is mostly achieved by going through complex scheduling and VM allocation algorithm by weighing and filtering (c.f.,[47, 48]));

\(^2\) https://www.rabbitmq.com/
2.2.3.4 Detailed Messaging Flow for OpenStack VM Provisioning Request

**Step 5** – The Nova-scheduler sends the RPC call request to the Nova-compute for launching instance to the appropriate host. Note that if the host is an external system, then communication takes place to the compute nodes;

**Step 6** – The Nova-compute sends a message to nova-conductor to fetch detail instance information such as host-id, flavor and system specification;

**Step 7** – The Nova-conductor interacts with Nova-DB through messaging queue and makes a RESTful call by passing on the token to the Glance-API to get image data (Uniform Resource Identifier (URI), image name, and size among others);

**Step 8** – The Nova-compute also passes the Restful token to Neutron through Neutron-API to configure Neutron based on user network specification including IP address;

**Step 9** – The Nova-compute also passes a similar Restful token to Cinder through Cinder-API for volume based on user request to attach storage volume;

**Step 10** – The Nova-compute generates required data for the hypervisor driver and executes the request on the hypervisor. More processes leading to the launching of VM will continue at the hypervisor level which may need further operations from host OS of hypervisors in the case of bare metal configuration;

Physical OpenStack services deployment architecture determines how Nova interacts with many other OpenStack services. In addition, user request type, resource preference affects the communication within the OpenStack service stack. Likewise, the message-queue service (such as RabbitMQ) serves as the central pathway for most of the interaction between Nova’s components. Similarly, authentication of each request is handled by the Keystone service through authentication token passing forth and back.

The above steps denote the conceptually fundamental activities that, when executed successfully, will result in the processing and creation of the VM. The steps are used in both chapter 3 and chapter 4 to formulate the topology of the proposed stable open QNM, which is decomposed into individual $M/M/\text{cci}_i$ and $GE/GE/\text{cci}_i$ $(\text{cci}_i \geq 1), i = 1, 2, ..., 5$ queueing stations (c.f., [24]), the individual solutions of which are used for the performance analysis of the OpenStack IaaS CCP architecture.
2.3 System and Software Testing

System and software testing is the assessment of system or software product and service for its correctness. That can be achieved through the execution of programs and observing the actual outcome and comparing it with the expected one. It comprises any activity aimed at measuring and evaluating the capability of a program or system to determine whether it meets its given specification [49]. Testing is difficult and worthwhile in every aspect of an invention because products and services need to pass some level of satisfaction before deploying them to the user community [50]. Testing varies in many perspectives and characteristics that are categorised broadly as functional and non-functional testing. In functional testing, system or software are examined to see whether it meets a given specification and user’s requirements. Functional testing includes a module or component test, system test, and regression test. Non-functional testing, on the other hand, checks the reliability, availability, performance, and security, among others. Software testing is about demonstrating the presence, not the absence of errors in the program [49, 51]. There are different techniques and approaches for both functional and non-functional testing and analysis of systems or software. Software type determines the suitability of the approach or technique to be used in testing its readiness. The software type ranges from standalone, distributed, web-based and cloud computing systems or platforms. This thesis considered CCP and performance testing and analysis as the type of software and testing technique, respectively. Specifically, modelling and performance evaluation was the core investigation of this thesis and OpenStack IaaS was the chosen CCP.

This subsection presents an overview of cloud testing as well as the research progress made in the area of cloud testing in general as well as a road map towards different categories of the area. A systematic literature review and mapping method were adopted, and an initial mapping study was completed, which shape the chosen area of performance analysis in this study. The literature starts with term Cloud Testing and eventually narrowed to the ‘performance modelling and analysis of IaaS CCP.'
2.3.1 Cloud Testing

Cloud testing (CT) [52-63] is a phrase that describes the use of cloud computing to perform both functional and non-functional testing of software systems. The non-functional testing includes testing software and systems for their performance, security, availability, reliability, among others. Furthermore, various approaches, mainly empirical, are adopted to investigate the cloud testing ecosystem at both functional and non-functional aspects were reported. In this context, however, cloud testing is distinguished in the following levels.

- **SaaS level:** Is the testing of software application that resides in the virtual machines on CCP. That involves testing SLA configurations, usability, performance, compliance, security, and verification, among others. That can be the responsibility of the developers writing the software services hosted as a SaaS but work in close relationship with the providers to identify performance issues. Depending on the services hosted, various testing methods will be used, such as web sites tested through app hosting methods.

- **PaaS level:** Testing the integration, connection, and interoperability of applications as well as the provision of environment for PaaS applications for companies offering platforms to use. That includes testing software database interactions for smart city applications or developer tools testing for examples.

- **IaaS level:** Testing the cloud itself. That includes testing hardware and software configurations, networking, virtualisation, resource provisioning, data transfer rate, scalability, availability, task scheduling, compliance to SLA, among others. This includes checks for cloud availability and fault tolerance. The cloud service providers usually carry out this type of testing.

- **Testing as a service:** Providing this service to application developers or testers to use. This service is provided to software and platforms to use (such as test cases for unit, integration and system, performance, load, and stress testing. Developers test in the cloud environments has used multiple methods. Most of the studies [53, 57, 64] have focused on one aspect of testing (e.g. performance).
2.3.2 Cloud Testing Ecosystem a Systematic Mapping Study

Systematic literature review (SLR) and systematic mapping studies (SMS) are some of the well-known methods of systematic searching, recording, analysing, synthesising, and reporting about articles from the literature in a particular subject of interest. This study, adopted from scientific sources, which is finding and evaluating the available research related to the researcher’s questions, subject area, and related phenomena. The guiding articles gave the reasons for conducting systematic review/mapping [65-69]. These include:

a. To summarise the existing evidence concerning technology
b. To identify gaps in the existing literature in order to suggest further investigation and

c. To come up with a framework or background that will be used to position additional research activities.

The phases followed in conducting this SMS are expressed as follows (c.f., Figure 2.9.):

I. Planning

The study starts with the identification of objectives, protocol definition, and coming up with research questions. Protocol definition involves the selection of methodology to be adopted in undertaking the review [66, 68, 69] so that it satisfies all the main characteristics of the study such as comprehensive, reproducible, and explicit so that other professionals may reproduce the same protocol and judge the adequacy and quality of the research.

Mapping questions:

The mapping questions (MQ) were formulated as follows:

- MQ1: What is the distribution pattern of Cloud testing publications across the years under review?
- MQ2: Which cloud testing area receives more publications, and who are the most active authors?
- MQ3: Which publication forums publish more research in the cloud testing service ecosystem?
• MQ4: Which Cloud layer (SaaS, PaaS, IaaS) have received the most attention in testing literature?

II. Searching

The research questions drive the selection of search terms used in searching for articles from various sources. The search query, keyword terms, and their synonyms were carefully composed. The search comprises of two main terms, ‘Cloud’ and ‘Testing’ and their synonyms. The search strings present a set of statements that combine keywords and their synonyms. Table 2.1 below shows the search strings used and their connecting Boolean operators. The wordings in the
quest string $S_1$ and $S_2$ were defined based on previous systematic literature reviews.

<table>
<thead>
<tr>
<th>Category</th>
<th>Search terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>Cloud (C) Cloud software, Service-oriented, distributed, virtualised</td>
</tr>
<tr>
<td>$S_2$</td>
<td>Testing (T) Testing, Verification, Assessment, Analysis</td>
</tr>
</tbody>
</table>

The search terms were defined by combining $S_1$ and $S_2$ shown in Table 2.1 using “AND” and “OR” Boolean operators, thus: $(C_1 \ OR \ C_2 \ OR \ … \ OR \ C_n) \ AND \ (T_1 \ OR \ T_2 \ OR \ … \ OR \ T_n)$ resulting to the following search strings:

$$(\text{Cloud} \ OR \ “\text{Cloud Software}” \ OR \ “\text{Service Oriented}” \ OR \ “\text{Service-Oriented}” \ OR \ distributed \ OR \ virtual*) \ AND \ (\text{Test*} \ OR \ \text{Verification} \ OR \ \text{Validation} \ OR \ \text{Assessment})$$

The strings above were also augmented with another string formulated using keywords of related studies as follows:

$$(“\text{Cloud}” \ AND \ \text{Testing}) \ OR \ ("\text{Testing as a Service}”) \ OR \ (“\text{Cloud Services}” \ AND \ \text{Testing}) \ OR \ (“\text{Cloud-based Testing”) \ OR \ (“\text{Cloud-based Software Testing”) \ OR \ (“\text{Infrastructure as a Service}” \ AND \ \text{Testing}) \ OR \ (“\text{Platform as a Service}” \ AND \ \text{Testing}) \ OR \ (“\text{Software as a Service}” \ AND \ \text{Testing})}$$

The strings formed were used to retrieve candidate articles from the following databases: IEEE Xplore Digital Library$^3$, ACM Digital Library$^4$, Engineering Village$^5$, Web of Science$^6$, Scopus$^7$ and Science direct$^8$. The databases were selected based on previous studies [70, 71]. In addition to this, in-depth manual searches were conducted using snowballing the citations of the related reviews.

$^3$ http://ieeexplore.ieee.org
$^4$ http://portal.acm.org
$^5$ https://www.engineeringvillage.com
$^6$ apps.webofknowledge.com
$^7$ http://www.scopus.com/
$^8$ http://www.sciencedirect.com/
III. Extracting, Screening and Filtering

After successfully searching the articles, they are immediately extracted from various databases. The articles were also screened at the same time by reading titles and abstracts. This is followed by inclusion and exclusion criteria whereby articles are selected based on relevance to the research topic and the underlining research questions. The criteria used were based on [66, 67] guidelines augmented by other related research [72]. Figure 2.10 highlights the details of the process leading to the final articles considered for primary studies.

All the selected articles were analysed based on methods, selection criteria and ideas discussed. Articles that discuss testing/evaluating IaaS/PaaS/SaaS but relevant to the provision of testing service were not included as deemed not relevant to the research topic and research questions. The following were the type of articles included and excluded from the main primary study articles:

![Figure 2.10: Detail article selection process.](image)

**Inclusion and exclusion criteria**

The type of articles included are published journals, conference papers and proceedings, both from academic and industrial fora. All the papers selected were also written in the English language. On the other hand, the excluded articles are short papers, published books, book chapters or sections, unpublished articles, academic projects (e.g., thesis) and web pages/blogs, and social media content.
IV. Quality check

Although the stages of this SMS seem linear, in reality, many activities in various stages are carried out simultaneously. For example, the searching, extraction and practical screening all happens at the database results screen. It was then followed by the first level of screening the articles based on their titles. The second level of screening was reading the abstract, introduction and conclusion. In addition, to minimise bias and ensure quality, more than one person conducted stages 3 and 4.

V. Mapping and Reporting

At this juncture, articles are sorted into schemes form different classification and categorisation. The categorisation will be to address the research questions of the study. Other schemes can be deducted because of a particular identified pattern. In this study, the primary study articles were categorised to address the research questions (MQs) as follows:

a) Classification of articles into categories

The selected articles were first classified base on the evaluation criteria recommended by Wieringa et al. [70] to help present a landscape of research in the field of cloud testing.

- Philosophical papers (PP): These papers sketch a new way of looking at things, a new conceptual framework, among others.
- Evaluation research (ER): Investigation of a problem in cloud testing ecosystem practice or implementation of a CT technique in practice. Its novelty is based on knowledge claim and soundness of the research method used
- Solution Proposal (SP): Papers that propose a solution technique and argues for its relevance, without a full-blown validation. The method must be novel or at least a significant improvement of existing ones, for example, proof of concept papers.
• Opinion Papers (OP): These papers contain the author’s opinion about what is wrong or right about something, how things should be done, among others.

• Validation Research (VR): Papers investigate the properties of a solution proposal that has not yet been implemented in cloud testing practice. The solution may have been proposed elsewhere, by the author or by someone else. The investigation uses a thorough, methodologically sound research setup. Possible research methods are experiments, simulation, prototyping, mathematical analysis, mathematical proof of properties, among others.

• Experience Papers (EP): In these papers, the emphasis is on what and not on why. The experience may concern one project or more, but it should be a personal experience of the author. The paper should contain a list of lessons learned by the authors. Papers in this category usually come from researchers or industry practitioners who have used their tools in practice, and the experience will be reported without a discussion of research methods.

The first results present the classification of the articles based on the categorisation of the research type adopted from [70]. Figure 2.11 shows the research types with philosophical papers were reported the most. It also shows the increasing interest in the solution proposal, especially in the years 2013 and 2014.

---

Figure 2.11: Articles classification based on research type.
b) Classification based on mapping questions (MQ1)

To answer mapping question MQ1 What is the distribution pattern of Cloud testing publications across the years under review? Figure 2.12 shows that the number of articles published is increasing every year. This number is from all the five categories of research presented earlier (i.e., PP, ER, SP, OP, and VR), with 2015 reported the highest number of studies related to cloud testing, although the articles search ended in July 2016.

![Figure 2.12: Relevant publications per year.](image)

```
<table>
<thead>
<tr>
<th>Year</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>2</td>
</tr>
<tr>
<td>2009</td>
<td>5</td>
</tr>
<tr>
<td>2010</td>
<td>12</td>
</tr>
<tr>
<td>2011</td>
<td>15</td>
</tr>
<tr>
<td>2012</td>
<td>20</td>
</tr>
<tr>
<td>2013</td>
<td>36</td>
</tr>
<tr>
<td>2014</td>
<td>47</td>
</tr>
<tr>
<td>2015</td>
<td>48</td>
</tr>
<tr>
<td>2016</td>
<td>17</td>
</tr>
</tbody>
</table>
```

To answer mapping question MQ2; Which cloud testing area receives more publications, and who are the most active authors? Figure 2.13 shows the authors that published more articles. The research did not consider whether an author is first or second (primary or associate) authors. From the figure, it can be seen that Tsai, Wei-Tek published more papers in the area than the remaining authors do.

```
<table>
<thead>
<tr>
<th>Author</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsai, Wei-Tek</td>
<td>48</td>
</tr>
</tbody>
</table>
```


c) Classification based on mapping questions (MQ2)
d) Classification based on mapping questions (MQ3)

To answer mapping question MQ3 Which publication forums published more research articles in the cloud testing service ecosystem? In order to answer this mapping question, the primary study articles were grouped into conferences/workshops and journals. Also, a total of 20 different journal articles were used for publishing the 35 (17.32 %) articles, while the remaining 167 (82.67%) papers were published in conference papers/proceedings and workshops. In addition, 99 different conferences/workshops were used for publishing the papers from which International conference on software testing, verification, and validation workshops (ICSTW) have the highest number of papers (totalling 12). Likewise, the conference that follows with publication track of 10 papers is the IEEE International Symposium on Service-Oriented System Engineering (SOSE). Some of the published papers were in the same proceedings while others were in subsequent (proceeding and succeeding) ones. A summary of the papers and conferences were presented in Table 2.2, and the detailed list of all the primary study articles were enumerated and presented in appendix D.

Table 2.2: Summary of fora that publishes three or more papers

<table>
<thead>
<tr>
<th>SN</th>
<th>Conferences/Workshops</th>
<th>Number of papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ICSTW</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>SOSE</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>CLOUD</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>CLOUDCOM</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>UCC, SERE, AST, COMPSAC, CCGRID, MIPRO, SERVICES</td>
<td>3</td>
</tr>
</tbody>
</table>
Conferences/workshops key: International Conference on Software Testing, Verification, and Validation Workshops (ICSTW); IEEE International Symposium on Service-Oriented System Engineering (SOSE); IEEE International Conference on Cloud Computing (CLOUD); IEEE/ACM International Conference on Utility and Cloud Computing (UCC); International Conference on Software Security and Reliability (SERE); International Workshop on Automation of Software Test, (AST); IEEE International Conference on Cloud Computing Technology and Science (CloudCom); IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID); International Convention on Information and Communication Technology, Electronics and Microelectronics, (MIPRO); IEEE Annual International Computer Software and Applications Conference (Compsac); IEEE World Congress on Services (SERVICES).

e) Classification based on mapping questions (MQ4)

To answer mapping question MQ4 Which Cloud layer (SaaS, PaaS, IaaS) have received the most attention in testing literature? Here, the articles were first categorised into two broad testing types, functional (F) and non-functional (NF). The second categorisation was based on testing types reported in the papers. For example, some papers reported security testing, unit testing, and integration testing, among others. The testing types were also categorised according to testing types of the cloud ecosystem layers. A detailed classification was represented in Table 2.3 below.

Moreover, the research found different testing techniques used in various layers, for instance, at an IaaS layer, some articles reported research on the performance testing but mainly at framework level hence considered as philosophical papers.

For SaaS, most testing techniques adopted a functional testing method. Some examples did look at measures of performance such as using model-based tools. For PaaS layer, test case generation, functional approach to performance measures, as well as security testing, has the least focused area.
TaaS, however, uses a range of testing methods from functional to non-functional testing techniques. Most of these techniques use outsourcing tools on clouds for another testing. However, there is also some research on `Testing as a Service’ (TaaS) for the Cloud [73-75]. That allows an application to be tested online before deploying it, taking advantage of the fast configurability and scalability of cloud computing.

Another exciting and expecting testing area that has remarkable research reported in it is the mobile-based testing. In this category, most of the articles presented mobile testing as a service (MTaaS), which is a sub of TaaS. This further shows that many of the cloud testing research focuses more on the benefit of outsourcing testing service using cloud computing resources with little on testing the cloud itself. In the over 206 articles classified based on the cloud service layers, over 100 reported studies, the most researched area of cloud testing concentrates in the TaaS service model.

On the IaaS layer [76-79], more than 50 articles reported cloud testing with different testing types at this layer, with performance testing having the highest of 26 articles. Further closer examination of the 26 articles reveals a limited report on the modelling and analysis aspect of the cloud and hence the need to seek further reported publication using different search strings.

Table 2.3: Summary of articles classification based on testing on each cloud service layer

<table>
<thead>
<tr>
<th>Testing topic (F-functional, NF-non-functional)</th>
<th>TaaS</th>
<th>SaaS</th>
<th>PaaS</th>
<th>IaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance testing (NF)</td>
<td>PS - 50, 52, 56, 192, 204, 209, 237</td>
<td>PS - 104, 105, 182, 225</td>
<td>PS - 153</td>
<td>PS - 52, 70, 91, 92, 109, 111, 114, 115, 128, 131, 142, 144, 146, 147, 163, 170, 171, 173, 177, 188, 189, 197, 199, 200, 211, 242</td>
</tr>
<tr>
<td>Unit testing and test case generations (F)</td>
<td>PS - 51, 53, 55, 57, 239, 247</td>
<td>PS - 103, 120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoperability (NF)</td>
<td>PS - 66</td>
<td></td>
<td>PS - 182</td>
<td>PS - 156</td>
</tr>
<tr>
<td>Load and stress testing (NF)</td>
<td>PS - 125, 205, 235</td>
<td>PS - 97, 149, 201</td>
<td></td>
<td>PS - 116, 148</td>
</tr>
<tr>
<td>Category</td>
<td>PS - 12, 9, 106, 145, 162, 173, 177, 199</td>
<td>PS - 8</td>
<td>PS - 195</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------</td>
<td>--------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td><strong>Web service testing (NF)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>VM migration (NF)</strong></td>
<td>PS - 17, 19, 147</td>
<td>PS - 14</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Testing environment (NF/F)</strong></td>
<td>PS - 13, 15, 18, 28, 34, 53, 75, 83, 97, 126, 128, 144, 150, 156, 157, 167, 171, 172, 176, 178, 194, 203, 205</td>
<td>PS - 84, 90, 133, 134</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault injection methods (F)</td>
<td>PS - 16</td>
<td>PS - 83, 93, 101, 113, 197</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Survey of methods (F/NF)</strong></td>
<td>PS - 21, 33, 49, 54, 66, 76, 77, 105, 119, 129, 146, 190, 200</td>
<td>PS - 84, 90, 133, 134</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SLA based testing (NF)</strong></td>
<td>PS - 23, 104, 108, 139, 185, 192</td>
<td>PS - 83, 93, 101, 113, 197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regression (NF)</td>
<td>PS - 27</td>
<td>PS - 83, 93, 101, 113, 197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>**Scalability testing/elasticity/availability</td>
<td>PS - 56, 170, 198</td>
<td>PS - 80, 73</td>
<td>PS - 83, 93, 101, 113, 197</td>
<td></td>
</tr>
<tr>
<td><strong>Testing tools (F/NF)</strong></td>
<td>PS - 35, 132, 140, 174, 175, 184</td>
<td>PS - 137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security testing (NF)</td>
<td>PS - 37, 71, 72, 138, 169, 183</td>
<td>PS - 81[80], 60, 61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path coverage and test case generation (F)</td>
<td>PS - 38, 42, 55</td>
<td>PS - 137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomaly injection (F)</td>
<td>PS - 78</td>
<td>PS - 88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search based testing (NF)</td>
<td>PS - 81</td>
<td>PS - 88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluating cloud users (out of scope)</td>
<td>PS - 122</td>
<td>PS - 88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network analysis and offerings (NF)</td>
<td>PS - 87, 181</td>
<td>PS - 88</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VI. SMS Remarks

The systematic mapping studies carried out was the starting point in understanding the direction of research in the domain. All the mapping questions posted were addressed, and the future direction of the next study was determined. For example, in the study from the table above, Performance testing and analysis of the IaaS layer receives more attention. However, further investigation of the performance testing and analysis of all the service delivery layers is needed. For example, which technique (measurement, simulation, and analytical) is the most used method of performance testing and analysis? Answers to this question could potentially be the next phase of the systematic literature review analysis. However, these findings indicate a reasonable number of studies in the area, especially in the measurement base performance testing of CCP at the IaaS layer. The next section focuses on the literature of the chosen methods of performance modelling and analysis of CCP.

2.4 Performance Modelling, Analysis and Techniques

One of the most recent developments affecting systems performance is the rise of cloud computing and the virtualisation technology upon which the cloud is commonly built on. Performance analysis is a form of non-functional testing and evaluation of software and systems. Performance can be viewed differently from different system engineers, administrators and users [26]. A system’s performance is the degree to which a system or its components accomplishes its designated functions within given constraints such as speed, accuracy and resource usage. Systems are modelled based on its predicted or actual architecture, comprising its structural and functional components. Analytical, simulation models and numerical solutions of Markov models are among the most popular tools used for the modelling and analysis [11, 13]. In analytical models such as QNM, a routed job flow paradigm in which actual or anticipated system is conceptualised as service stations. The queue is oriented towards the structure of the system as a discrete event model.
Another form of modelling technique can be used to simplify the complexity of mathematical calculations. Simulation is one of the simplified performance modelling and analysis. In simulation modelling, the actual system is depicted using simulation tools, which help in displaying its behaviours through discrete event simulation tools such as MATLAB. The third modelling and performance analysis technique is the measurement-based technique whereby observation, as well as empirical techniques such as controlled experiments, are employed. Measurement is the commonly used technique when the system is up and running, especially for the small and medium-sized system. Modelling process includes a listing of various tests, the definition of performance acceptance criteria and identification of required metrics. Measurement requires a systematic approach in using it, as one of its drawbacks be the enormous resource and time it takes to execute [81].

Cloud performance evaluation is not straightforward because there are other underlying activities of the tenants (guest) as it is difficult to observe at the host system level. Another issue of evaluating the performance of the cloud is that it may be composed of several networked components that are needed to service the input workload. This includes load balancers, web servers, database servers and storage systems. Modelling and mapping of CCP help in revealing the overlooked sources of system perturbation [26]. The CCP can be modelled as a network of queueing systems for the analytical study of the individual machine. Especially with Clouds, all performances are in advance defined in SLAs, any breaches of which can lead to penalties and business loss to the cloud providers.

Performance issues may originate from complex interactions between subsystems that perform well when analysed in isolation. This can occur because of some cascading failure when a failed component causes performance degradation on others. To understand the resulting issues, one must untangle the relationships between components and understand how they contribute towards overall performance. Bottlenecks can also be complicated and related in unexpected ways; fixing one may move the bottleneck elsewhere in the system, with overall performance not improving as much as expected. Apart from the complexity of the system, a complex character of the production workload may cause performance
issues. In these cases, they may never be reproducible in a lab environment or can only be achieved intermittently [82].

Conversely, even though analytical modelling enables computation of large-scale systems such as enterprise or CCP without any expenditure of setting up the physical environment, the complex nature of such system as well as not careful planning and execution may yield inaccurate results. Moreover, as a cloud is scalable, its resource can be rented from many providers to test a large number to executions for short a period [83]. If correctly designed and executed, it will give more direct result than the estimate of the mathematical models. So, the workload can be generated, analyses visually and then validated analytically or vice versa. Measured results, such as knee points, maybe similar to the analytical or simulated but is based on the actual data rather than theoretical models [26].

Moreover, for a performance modelling and analysis to be robust, a combination of at least two techniques is required. One of the prevalent robust strategies is to combine at least two of the methodologies. For instance, starting with the measurement of the base test workload and expand it to the capacity of the current system resource. Then obtain the result and use it to initiate another technique such as simulation or analytical, first to validate the result and predict what will happen if the system capacity is expanded. Below is a brief description of the three primary technique for performance modelling and evaluation for both enterprise systems and application to a CCP.

2.4.1 Measurement

Measurement is one of the widely adopted methods of determining many systems performance parameters of interest. This type of performance analysis continues to evolve because of developing additional tools and OS kernel functions that mainly uses system time to determine the real flow of information across the different component of the system. Other widely used data to measure performance is the process logs that are usually stored in OS directories and application folder locations in a different form of files. Measuring the performance of a computer system or network of computers of various forms (such as enterprise network, and clusters among others), requires a monitoring tool to display the result and a load
generator to generate the workload [25, 26, 84]. These monitoring tools include Ganglia\(^9\) – mainly for monitory collaborative servers such as clusters and grids, Nagios\(^10\) – for server resources and services monitoring and alerting, Zabbix\(^11\) for graphical monitoring and alerting, among others.

In order to perform meaningful measurements, the workload should be carefully selected, the visualisation tool should be appropriate, and acceptable evaluation mechanism should be adopted. To achieve that goal, the performance analyst needs to understand the following before performing the measurements. Identification and/or setting up of the test environment, benchmarking the performance acceptance criteria, plan-design-and execute the test, generate and analyse the report [82].

Current literature shows various attempt to evaluate the performance of enterprise systems and cloud computing at various service layers using measurement-based techniques. This ranges from planning and developing a test strategy in a multitenant virtualised environment [84], to setting up and configuring the environment [85], as well as and executing, recording, visualising and reporting the results [86]. For example, Mdhaffar et al. [87], propose a framework for the reactive performance monitoring of CCP implemented using a toolkit that generates visual data about various components of the platform. Wolski and Brevik [88] presents an extensive workload characterisation and analysis of some open source CCPs and come up with parametric statistical data for the platform including VMs request data, performance data, measuring attributes and instrumentations. The study uses datasets from the datacentres to generate and visualise the performance metrics similar to the analytical approach. Different VM instant type of AWS was also analysed to obtain performance values of their resources which will help in gauging it for scientific computing [89]. Similar measurement based evaluation of different VM instance OS level performance was conducted by simulating different workload level using agent-based modelling [90]. The approach demonstrated capabilities and applicability of the flame framework for agent-based modelling. Performance

\(^9\) http://ganglia.info/
\(^10\) https://www.nagios.org/
\(^11\) https://www.zabbix.com/
evaluation at the PaaS layer was also investigated with the development of framework proposing 19 performance functions applicable to three different aspects of PaaS [91]. The three categories are CPU, memory, and database performance. The functions can be used to measure the read, write, compute-cycles and other platforms components. The authors supported their arguments with implementation on some of the popular commercial PaaS providers such as, Heroku, OpenShift, and Cloud Foundry.

Despite the presence of various strategies for measurement performance evaluation, one key issue here is the difference between test workload and production workload, which makes it hard to obtain results applicable to the production environment. Furthermore, real-time performance testing may introduce extra transaction workload, which may affect the user experience of using the systems. Moreover, with introduced transaction workloads, the performance evaluating may not be accurate. Performance predication is hardly feasible as the measurement results are strictly base on the execution data and within the capacity of the existing infrastructure. Next technique of performance evaluation is the simulation approach.

2.4.2 Simulation

In contrast to the measurement, simulation and analytical approaches, both modelled the system at capacity planning and during the design stage. While the analytical approach is mainly mathematical abstraction and numerical computation, simulation techniques, mimics an actual environment with similar characteristics. Performance evaluation using simulation approach is usually achieved using toolkits in the form of software applications. For instance, tools such as Wireshark\textsuperscript{12} and JMeter\textsuperscript{13} can be used to simulate the traffic generation, while JMT and Mobius enable a visual design of the model and generating random numbers to represent traffic flow and the computation by CPU. Most of the reported performance evaluation that uses a simulation approach does it in combination with either

\textsuperscript{12} \url{https://www.wireshark.org}
\textsuperscript{13} \url{https://jmeter.apache.org}
measurement or analytical technique. For example, Khazaei [92] modelled and evaluated the performance of cloud data centre using the analytical approach and validate the findings via simulation approach achieved via Artifex engine toolkit developed by RSoftDesign Inc [93]. CloudSim\(^{14}\) is one of the most widely used tools to simulate a diverse kind of experimentations aimed at examining the CCP. Many performance evaluation studies at different layers of cloud services ecosystem such as SaaS [94], PaaS, TaaS and IaaS utilises CloudSim and other related tools. More recently, Hussein et al. [85] use simulation to model the workload during performance evaluation for maximising utilisation of VMs for container placement architecture in CCP. The study augments the method with an analytical approach in order to validate the results. The following presents an overview of analytical methods for performance modelling and evaluation.

### 2.4.3 Analytical

Analytical modelling in contrast to empirical methods of measurement is a deductive method of model-based performance evaluation that is mostly applicable in situations where the system of interest does not yet exist. It is applied in the earlier design phase of system development to ensure the final product meets its performance and reliability requirements. Other advantages of analytical modelling are it is cheaper to do, and not risky. Hardware architecture such as network, scheduling algorithms and protocols can all be tested using analytical methods. Systems performance behaviour can be predicted without using real production workload as well as capacity planning [95]. Typical analytical methods are QNM, SPN and its variants as well as the Neural Networks (NN).

#### 2.4.3.1 Queueing Theory and Queueing Network Model

Queueing theory (QT) is an old mathematical method of studying waiting lines or queues. Founded by a Danish Mathematician Agner Krarup Erlang in his work of telephone networks, QT technique is applied in different areas. A queueing model

\(^{14}\)http://www.cloudbus.org/cloudsim/
represents a mechanism for computing and predicting the waiting time and the length of a queue. As QT gained worldwide recognition more than half a century ago, and continue to be used for computation, estimation, and prediction of waiting times, service times and related probabilities.

It comprises of an arrival process, services process and in most cases departure process (c.f., Figure 2.14).

Queuing network model (QNM) is simply a network of queues that are usually analysed by decomposition into individual stations. A queuing station is usually a node that is represented mathematically using a simplified notation known as Kendall's notation in the form A/B/c/D/E.

![Figure 2.14: A simple queueing system](image)

Where A describes job arrival process distribution, B describes jobs service time distribution; c shows the number of servers. D and E represent the buffer/queue capacity and the scheduling discipline (e.g. FCFS), respectively. An example of queuing station notation is the M/M/1 queue, where M stands for Markovian or memoryless (hence the M-type) which has both arrival and service processes following Poisson with exponential distribution.

QNM, in general, evolved from queueing theory founded analysis. Since then, the theory is in use until today for analytical modelling of the system as a network of interconnected queues. As it has been applied in various systems, notable evolution and contributions are in Burke’s theorem [96], Jackson open Queueing networks [97] and Baskett’s generalisation of types of queues [98] among others. The theory has been applied successfully in different fields of computing and telecommunication systems such as ad-hoc networks as well as wireless networks [99, 100].
The model was also applied in CCP in various attempt to demonstrate the application of QNM for performance modelling and analysis. For example, Khazai [101] used performance metrics such as task blocking and delays to give a good indication of how well the cloud centre is performing. Their work presented a need for an efficient admission controller that can examine the time every job will take and perform an intelligent trade-off analysis of which jobs to accept. This work has been extended to use Queueing models to improve performance during live migration of virtual machines to reduce task rejection probabilities not compromising on task service time [102]. The Queueing models M/M/c to measure inter-arrival and service times between jobs were broken down to waiting service and service execution time to optimise response [103]. Vilaplana et al. [104] discussed a cloud architecture based on QoS optimisation with a combination of M/M/1 and M/M/c to improve response time, reduce system bottleneck and guarantee SLAs. The authors conducted experiments through simulations of an OpenStack private CCP. The authors also show considerable improvement in job performances but did not discuss if the jobs were with different arrival rates and service times, as this would affect the performance of completion times. Using queueing networks in [105], the authors combined two Queueing Systems M/M/c connected serially. The first queue receives the tasks and the second queue represented the database system. In this way, the system was able to ensure every user with access to the database was able to allocate servers efficiently and thus the time for service completion reduced with more servers added to the system. A stochastic model to investigate data performance was also modelled analytically, examining different performance metrics against different distributions such as exponential, periodic, and bursty arrival process, although no specific IaaS cloud was used for the validation of the result and followed its internal processes for guidance [106]. A similar approach was used for capacity planning with simulated and analytical model [107, 108].

2.4.3.2 Petri Nets

Petri Nets (PN) is one of the mathematical modelling techniques founded and developed by Carl Adam Petri in 1962 as part of his PhD Thesis [18]. Ever since its development, it evolves as a robust modelling framework for the study of both
2.4.3.2 Petri Nets

qualitative and quantitative aspect of systems. PN has been used to model the functional behaviour of a system. The application area includes Computer system, Communication Networks/protocols, manufacturing systems, complex scheduling systems and controllers, among others. Petri Nets are considered as suitable for particularly modelling systems with concurrency, conflict, synchronisation and mutual exclusions. A PN is a bipartite directed graph that contains two types of nodes: Place, \( pp \) and transition, \( tt \). A place is a placeholder or buffer representing a tasks/jobs/customer depicted by a round/ circular shape, while the task inside the place is called tokens. The total number of tokens defines the state of a place. A transition, on the other hand, is the activities/action/events that change the system from one state to the other. A rectangular bar represents transitions. Other components are the Arcs, which connects places and transitions only, which shows their interdependencies. A transition is fired only when there is at least one token in all the input places, and once fired one token is removed from all input places, and one token is added to all the output places (c.f., Figure 2.15).

![Petri Net with typical components](image)

Figure 2.15: Petri Net with typical components

A PN is formally defined as a tuple as follows: \( \text{PN} = (\text{PP}, \text{TT}, \text{II}, \text{OO}, \text{MM}(t_0)) \), where \( \text{PP} \) is a set of places = \( \{p_1, p_2, ..., p_m\} \), \( \text{TT} \) is a set of transitions = \( \{t_1, t_2, ..., t_n\} \) and \( \text{II} \) is a set of input places associated with corresponding output \( \text{OO} \) places. A marking of a PN at time \( t \) \( \text{MM}(t) = \{MM_1(t), MM_2(t), ..., MM_m(t)\} \) with \( MM_i(t) \) as the number of tokens in place \( p_i \) at time \( t \). Moreover, at an initial time \( t_0 \), \( \text{MM}(t_0) \) represents the initial marking.

PN is also characterised by the reachability set which represents a resulted sequential firing of a group of markings reached from the initial marking \( \text{MM}_0 \). Therefore, a state-space of reachability for a PN is determined through a structure
of the system, based on the enabling firing rules. In addition, a PN state space is
defined as finite or infinite, which may explode with a large set of markings of the
PNs (c.f., [14, 16, 17]). A transition is activated, only if there exists at least one token
in all of its input places. Once activated, the firing will take place, thereby reducing
the number of tokens from input place, adding the number of output places will
decrease, and thereby enabling the change of system state.

In order to evaluate the performance of systems or network, many extensions
of PN were proposed. These extensions such as ‘time delay at places’, arcs, tokens
and transitions, enable the introduction of timing thereby referring to it as a Timed
Petri Nets (TPNs), which is either stochastic or deterministic (c.f., [14, 16, 17]).

Furthermore, another extension was explicitly developed when the time is
associated with transition and is known as a timed transition Petri net (TTPN), where
‘time’ is associated with ‘transitions’. This means a ‘time’ signifies ‘action’,
corresponding to the start and the end of firing a transition [14]. Moreover, if the
timed delay is distributed exponentially, the resulting TTPN is known as a stochastic
Petri net (SPN). In SPN, the firing policy is ‘atomic’, because a transition is fired after
a random amount of time, which resulted in removing one token from an input place
and taking it to the output place as one action (c.f., [109]).

Furthermore, a new type of extension of SPN was developed with ‘immediate’
transitions and is known as generalised stochastic Petri net (GSPN). GSPN have a
transition with higher priority than a ‘Timed’ transition, which may have a zero firing
time. The extension of GSPN also introduces another type of arc known as ‘inhibitors
arc’. The inhibitor arc allows the firing even if the there is no tokens in an input place
connected to it, as long as the other input places have at least one token each (c.f.,
[14, 16, 17]).

Other extensions are the coloured Petri net (CPN) which uses different colours
to distinguish between tokens [15]. Stochastic reward nets (SRNs) is another SPN
extension, which is mainly for determination of reliability of complex systems by
specifying output measures as reward-based functions mostly used for modelling
the fault tolerance systems [110]. Despite, many extensions of PNs [14, 16, 17], it
has limited state space explosion as a result of the expansion of system size. On the
contrary, PNs are still sufficient in the descriptive and quantitative ability for explicitly
modelling and evaluation of performance as well as performance and security trade-offs in complex systems.

PNs and all its extensions are adopted to model and analyse different kinds of systems performance with substantive results. For example, R. Ghosh et al. [111] provided valuable insight into the use of a monolithic availability model of IaaS CCP by evaluating the reliability and availability of cloud resources using an SRN model, which utilises failure analysis and prediction for both host node and their resident VMs. Similar work in [107] reported a detailed performance analysis of various provisioning steps of a generic IaaS CCP architecture. A recent survey conducted in [112] focuses on the area of cloud performance modelling and evaluation based on classical queueing theory methods, SPNs, SRNs, process algebra and measurements. An SRN was also adopted in [113] where memory virtualisation was modelled with a failure level and system’s memory leaks. More measurement-based performance modelling studies of CCPs can be seen in [114] and [86].

Moreover, performance studies in cloud computing research have also focused on cloud networking issues using software-defined-network (SDN)\(^\text{15}\) and OpenFlow\(^\text{16}\), in particular, Kashinath, et al. [115].

### 2.5 A Literature Review on Performance and Security Trade-off Models

Performance and security analysis in computing system has been in the literature long ago with analysis of each one in isolation or as part of the QoS components. In addition, investigations to identify the relation between the two, such as how one affects the other has also been studied mostly using a quantitative approach. For example, Cho et al. [116, 117] studies used the wireless group communication between wireless entities to study and analyse a trade-off between performance properties of the intrusion detection system (IDS) and its security. The study showed how the frequency of executing IDS can affect the performance of the group communication system and how reducing it improves performance. The

\(^\text{15}\)[https://www.opennetworking.org/sdn-definition/]

\(^\text{16}\)[https://www.opennetworking.org/]
quantitative analysis utilises SPN for modelling and mean time to security failure (analogous to mean time to failure in dependability and reliability analysis [118, 119]) for security metric, while service response time was used as the performance metric.

In another study of performance and security, Alentina C. et al. [120, 121], use PerfCloud tool to represent a cloud-Grid architecture for the investigation of a trade-off between performance and security. The study implemented a grid architecture on top of a CCP and studied the overhead introduced by having an additional layer. The study also compares the performance overhead introduced by cloud services with different security levels. This study differs from others by using a measurement approach to record the response time of messages with and without security at the transport layer and at data layer (encryption) which shows how adding security affected performance and how to attain acceptable security with decent performance.

Zeng W. et al. [122] studied a trade-off model for performance and security in secured networked control systems. The study employs encryption as security element and uses a quantitative security metric of key length to determine the strength of the system security and used it to assign value to the time it takes to crack (penetrate through) a system. The mathematical model implemented in differential equations considers a brute force to determine the time it takes to gain access. The trade-off was determined by computing the level of both security and performance needed to reach optimal trade-off. The study was implemented using discrete event simulation. Although the study was conducted on the secure DC motor, their findings reflect how high-performance encryption algorithms can provide secure environments and limit the performance overhead.

Another approach to determine the trade-off between performance and security is the analysis of block cipher encryption using the data encryption standard (DES) algorithms. This kind of encryption, specifically with block cipher requires a second level of verification by correcting all the errors generated during the first level process. The study investigated the effect of giving up the error correction stage of encryption in order to improve performance [123].

Mean Time To Security Failure (MTTSF) was one of the most used security metric (analogous to mean time to failure MTTF in reliability analysis [119]) in many
other trade-off models. For example, in the trade-off model of combined sensing, performance and security [124], robotic ad hoc network [125-127], and mobile CCP using timing attack case study [128]. All these models use quantitative approach through GSPN implemented using different tools and illustrates either the use of encryption key length or firewall through access control to adjust the security level in return to gain more performance. The performance uses either, mean response time, node or systems throughput, node or system utilisation, among others as the performance metrics.

Another approach to performance and security analysis using a delay tolerant network (DTN) is presented in [129]. The study considers an anonymous routing protocol for DTN performance analysis aspect. A mathematical model was developed to describe the performance and security guarantee for onion-based anonymous routing in DTN. The nature of onion ring anonymous routing requires different encryption at each layer of the (onion) message. The work is mainly aimed at bringing out the complex nature of the various layers of the onion ring (anonymous) protocol and how it affects the performance of communication between group members through evaluating the number of message copies which dictates the performance metric [129].

Lastly, Wolter K. et al. [130], developed a detailed model to study the trade-off between performance and security, by first adopting and modifying the reliability model [131, 132] in an attempt to identify a metric for security modelling. The study then identifies a performance metric of utilisation and use the two QoS components for modelling the trade-off between them. A GSPN was employed, which resulted in generating some illustrative results that show how an increase in one can drastically affect the other and vice versa. The model assumed an abstract communication system and can also be used to forecast the potential revenue generation base on satisfying both parameters of security and performance. This chapter adopted the approach proposed in [130] and similar studies to illustrate how the trade-off model of CCP can be implemented primarily during a VM provisioning request in CCP. Next section presents discussions of the performance and security metrics employed for the analysis.
2.6 Research Design and Methodologies

Research issues in software performance testing and analysis are numerous, and different approaches are available for a different type of the situation [81, 133, 134]. However, since the main aim of this thesis is to investigate the performance issues in CCPs, different research methods will be applied at different stages. A combination of analytic methods is employed, based on the derivation of a mix of exact and approximate solutions for the modelling and analysis of CCPs. This analysis is complemented with simulation and empirical validation, as appropriate. Highlights of the methods used in each chapter are presented below.

1. Literature Review: At this stage, the identification of the knowledge gap in the area of CCP’s testing and specifically is undertaken on performance testing, analysis, and associated metrics, as appropriate. Systematic mapping studies (SMSs) were adopted, [67, 69, 135, 136] which provide comprehensive guidance on the process. Further related works were reviewed, such as security and ‘optimal’ performance security trade-off, among others. Details are presented in the literature review subsection of chapter 2.

2. Analytical Modelling and Analysis: An open Markovian (M-type) QNM [96, 137, 138] with an external Poisson arrival process and exponential distributions for the service times for the performance modelling and exact analysis of a CCP was employed. Based on Jackson’s theorem [97], the open QNM was decomposed into individual M/M/1 and M/M/c (c > 1) queueing stations, each of which is analysed in isolation having a Poisson arrival process with an overall mean arrival rate and exponential distribution for both interarrival and service times [139]. In this way, the external Poisson’s process with exponential service time distribution at each station was employed for the development of the QNM of virtualisation in OpenStack CCP. The performance metrics of mean response time, server utilisation and mean throughput were computed. The underlying detailed method was specified, as appropriate (c.f., Section 3.3).

3. Extended Analytical Modelling and Analysis: A stable open QNM with an external CPP arrival process and queueing stations with GE-type external interarrival and
service times were applied for the modelling and performance evaluation of a CCP. The stable GE/GE/1 and GE/GE/c queueing models were analysed by using overall mean arrival rates and squared coefficients of variation (SCVs) greater than 1 [24, 140] as well as employing two-phase Hyperexponenti-2 (H2) distributions with the same mean and SCV with the aid of packages such as java modelling tool JMT17. The method provides a worse case performance scenario as well as identifying CCP’s bottleneck(s) station(s), which is an essential metric of interest to cloud performance engineers. A detailed methodology with stages is presented in Chapter 4.

4. Modelling Performance and Security ‘Optimal’ trade-offs’: Discrete event simulation was carried out in Chapter 5, where GSPNs was employed to simulate a suitable GE-type analytic model by adopting a security detection control model (SDCM) (c.f., [130]). Mobius18 package was used for the model simulation, while model validation using machine learning techniques were suggested for future work.

2.7 Summary

This chapter presents an elaborate background on the virtualisation technology, cloud computing and OpenStack IaaS CCP. Virtualisation is the technology that enables the provision of cloud computing service, and OpenStack is one of the open source and widely used CCP and was also applied in this thesis. Next, the chapter elaborates on the system and software testing as this thesis is focused on performance analysis, which is an integral part of system and software testing. An extract from the systematic mapping study was also presented, which pave the way for this research. Moreover, the chapter presents a literature review on the chosen methodology of performance modelling and analysis and present related work in the area with a critical review of the literature. The literature on the performance and security trade-off model is also discussed, highlighting a critical

17 http://jmt.sourceforge.net/
18 https://www.mobius.illinois.edu/
review of the related literature on different approaches of investigating the impact of security on performance and the associated trade-off models.

Overall the literature reveals some related past and current studies that report quantitative investigations on the performance modelling and analysis of different computing systems, including CCPs. Although most of the reported research centres on the development of models for computing and analysing abstract systems, some studies reported specific research on CCPs. However, the reviewed researches do not analyse CCP architecture using similar typical setup architecture as presented in this thesis. Moreover, the impact of security on performance were also on other communication systems, not service-based systems such as the cloud.

Next chapter presents the first attempt to model CCP using the M-type QNM.
3 PERFORMANCE MODELING AND ANALYSIS OF OPENSTACK IaaS CCP USING M-TYPE QNM

3.1 Introduction

The main popular techniques for performance modelling and analysis of computing, networking and related systems are either measurement, simulation and analytic techniques. Large scale and complex systems performance evaluation are tedious, time-consuming and prone to error if not carefully planned and executed. Measurement tends to be more promising as direct observation can be done. However, it is economically not feasible to set up and configure environments to check its resilience against workloads. That leads to the popularity of mathematical modelling as well as a simulation in order to obtain acceptable system behaviour predictions at the design stage. Analytic modelling, simulation and combination of the two techniques are widely adopted.

Moreover, recently, statistical learning [141] also known as machine learning (ML) approaches are gaining popularity due to the availability of large scale dataset from commercial data centres such as Google, Facebook, Amazon, Alibaba, Yahoo among others. These companies releases terabytes of datasets that are complex and requires systematic approaches to download and specialised tools to process. For example, Alibaba\(^{19}\), Google\(^{20}\) and Yahoo have dedicated websites containing their production workload datasets with guidelines and links to download extract and store. The datasets are publically available for analysis and prediction of systems behaviour with such workload as well as its characteristics such as performance, and security. ML techniques such as deep learning, reinforcement learning and neural networks are adopted to predict system behaviour when subjected to different workload.

Nevertheless, as the prediction using ML techniques is gaining popularity, and producing promising outcomes, the more widely adopted prediction approach

\(^{19}\) https://github.com/alibaba/clusterdata/blob/master/README.md
\(^{20}\) https://cloud.google.com/public-datasets/
remains mathematical modelling techniques. However, one can use a modelling technique and validate it with simulation and vice versa.

This chapter presents a new proposed stable open QNM consisting of single and multiple servers queueing stations focusing on OpenStack IaaS CCP, the model starts with developing a typical setup architecture of OpenStack IaaS CCP (c.f., Figure 3.2). A performance modelling and analysis study of virtualisation in OpenStack IaaS CCP using an open QNM with stable M/M/1 and M/M/c queueing stations is carried out. Jackson’s theorem was applied, and the QNM was decomposed into individual queueing stations, each of which was analysed in isolation. The model assumed Poisson process for both arrival and service times, which is an optimistic performance evaluation that is more of finding best performance scenario. Consequently, performance metrics for each queueing station, such as a mean number of requests, server utilisation and throughput were computed. In addition, numerical experiments are conducted in order to identify the performance characteristics of a typical snapshot of queues within the CCP. In this way, performance predictions can be carried out by interpreting the bottleneck analytic results towards future design and development of credible OpenStack IaaS CCP.

The rest of the chapter is structured as follows: Section 3.2 provides an overview of M-type QNM processes for modelling and analysis of system performance. This was followed by this research approach of developing a typical setup architecture of the OpenStack IaaS CCP. Modelling and analysis using M/M/1 and M/M/c were followed in section 3.3. Numerical experiments and results that determine performance metrics, as well as the bottleneck evaluation of the CCP, based on the analysis of the proposed QNM, are presented in section 3.4. This was followed by the chapter’s summary and suggestions for future work in section 3.5.

### 3.2 M-Type QNM for Performance Modelling and Analysis

QNM approaches models the system based on the proposed system design specifications for the systems under consideration and from similar recommendations documented during tuning as well as an upgrade during capacity planning review of existing systems. The documents may contain information such
as number of resources including servers, type of servers (e.g. processing and database servers), anticipated messaging flow (including directions) from one server to another and other information such as sequence diagrams. This information is used to depict the system using queueing notations and results in developing a model.

![M/M/1 and M/M/c queues](image)

Figure 3.1: M/M/1 and M/M/c queues with arrival and service time symbols

Definition of relevant performance metrics, notations and meaning are presented in Table 3.1 below. The table also represents other relevant notations with their meaning.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>External arrival rate ($\lambda$)</td>
<td>The average rate at which external job (request) enters the system. It is represented by a well-known arrival process such as the Poisson process. While the interarrival time is defined as the time interval between successive arrivals denoted by $\frac{1}{\lambda}$.</td>
</tr>
<tr>
<td>Stations arrival rate ($\Lambda_{ii}$)</td>
<td>The average arrival rate at each station $ii$</td>
</tr>
<tr>
<td>Stations service rate ($\mu\mu$)</td>
<td>The average rate at which a station processes a request. A service time, on the other hand, is the time taken by a station to process a request. This is represented by distributions such as exponential distribution.</td>
</tr>
<tr>
<td>Probability of having $n$ request in the system ($P_m$)</td>
<td>The marginal probability of the time $n$ jobs (requests) spent in the system (CCP).</td>
</tr>
<tr>
<td>Station Utilization ($\rho$)</td>
<td>The measure of how busy a server is. It can also be determined as the ration of arrival rate to the service rate at a given time.</td>
</tr>
<tr>
<td>Mean Response time ($TT$)</td>
<td>Also known as sojourn time, is the total time a request spent in a system.</td>
</tr>
<tr>
<td>Mean Waiting time ($\mu$)</td>
<td>Is the average total time a request spent waiting to be served in the system.</td>
</tr>
<tr>
<td>Mean Number of requests in the station ($KK$)</td>
<td>Is the average number of jobs (requests) in the queueing station.</td>
</tr>
<tr>
<td>Mean Number of requests in the queue ($QQ$)</td>
<td>Is the average number of jobs (requests) waiting in the queueing station’s buffer (queue) for processing.</td>
</tr>
<tr>
<td>Mean system Throughput ($\lambda_{TT}$)</td>
<td>Is the actual rate at which a request is transferred out of the system.</td>
</tr>
</tbody>
</table>
Modelling involves the depiction of the system diagrammatically using queueing theory notations (i.e. Kendal’s notation) [11-13] and assignment of symbols representing critical input data such as arrival rates, service time, number of servers, among other (c.f., Figure 3.1). A markovian also refer to as memoryless (M-Type) QNM assumes the job arrival process follow Poissons process according to the exponential distribution. Also, the service time distribution is considered as exponential and represented by $M_{\text{arrival rate}}/M_{\text{service rate}}/1_{\text{server}}$ and $M_{\text{arrival rate}}/M_{\text{service rate}}/c_{\text{servers}}$. To model, analyse and determine performance metrics using M-type QNM, an arrival, service time distributions need to be determined or assumed. For example, one of the standard random distributions applied is the Poissons arrival process with exponential interarrival and service time distributions.

The analysis in this contexts starts with the decomposition of the model from QNM in tandem into an individual queueing station in order to follow Jackson’s theorem [13, 97] and analyse the stations individually. This is achieved by determining the individual arrivals, transition probabilities as well as service time at each station. The system is assumed to be in equilibrium which simplifies the evaluation and determination of the relevant performance metrics to be used in performance computation using appropriate formulae. Detailed processes yielding to the computations are described in the following subsections.

### 3.3 M/M/1 and M/M/c QNM for VM Provisioning Request in OpenStack IaaS CCP

Analytic modelling in contrast to empirical methods of measurement is a deductive method of model-based performance evaluation, which is mostly applicable in situations where the system of interest does not yet exist. The proposed analytical model is based on the OpenStack IaaS Cloud computing architecture presented in Figure 3.2. It comprises of three network types: Public network, Admin network and high-speed data network between computing resources. It also comprises of Controller nodes (which operate using subsystems as described in Figure 3.2, i.e. gathering requirements for VM instance creation and similar services). Compute nodes (which host the VMs instances created for the compute service), Admin node (responsible for managing the hardware, i.e. booting and
shutting down as needed), Storage nodes (responsible for housing/hosting the VM instance volumes, i.e., hard disk drive (HDD)).

The topology of the architecture for the OpenStack IaaS CCP is represented by the innovative, stable open QNM of Figure 3.3, consisting of six queueing stations either one server or multiple servers. These stations are connected via a random routing and provide exponential service times with homogeneous mean rates \( \{\mu_i, i = 1, 2, \ldots, 6\} \) and number of servers of queueing station \( c_i, (c_i \geq 1) \) such that \( c_1 = 1, c_2 > 1, c_3 = 1, c_4 > 1, c_5 = 1, c_6 = 1 \).

The use-case considered is the users requesting the creation of VM instance in the OpenStack CCP (c.f., Figure 3.2). The clients generate via the Internet requests for VM provisioning according to a Poison external arrival process with mean rate \( \lambda \). Moreover, the requests seek ‘service’ from OpenStack IaaS CCP relating to the provisioning of a VM instance. Initially, the client’s service request passes through a firewall (station 1), where it is assumed that some requests will be dropped, based on some security criteria. The request will then be forwarded to the Controller (station 2). The Controller will process the request internally by calling its services (c.f., Steps 2 – 4 of section 2.2.3.1). The Controller node will then forward the request (message) to the Compute node (station 4), which will start the process of creating the VM if the designated Compute node is already powered. If the Compute node is not powered on, the Admin node (station 3) will receive the instruction to switch on the Compute node before forwarding the request. The
Compute node will create the VM if the detail specification does not require external (distributed) storage. If the request needs external (distributed) storage, it will be sent to the Storage node (station 5), which will process it by following the allocation of the required storage volume. The Output server (station 6) will send back the successful creation message or otherwise to the requesting client. The detailed flow is described in section 2.2.3.1.

The assignment of single and multiple nodes in the setup topology was due to the number and size of the operation and processing at those nodes. The bulk of the processing during VM creation task takes place in the Controller and the Compute Nodes, hence the reason for using multiple nodes. Controller nodes host several services (c.f., Section 2.2.3), while the task of spinning the VM instance happens in the Compute nodes, which takes a substantive amount of resource.

Storage node only hosts the instance volume (similar to HDD). However, only some type of request will seek for such distributed storage systems. Similarly, the Firewall node does the Ingress filtering of the oncoming traffic into the network. In addition to the Firewall's security processing, further security checks of authenticating the user and the request type happen at OpenStack's identity and management service called Keystone (c.f., Figure 2.8) where the bulk of the security operation happens, thus the reason for using a single node to represent the Firewall node. Detailed tasks of each service that resides in either of the nodes are specified in section 2.2.3.

![Figure 3.3: Schematic diagram of the new open QNM for OpenStack VM provisioning request](image)
The \{p_{10}, p_{12}, p_{23}, p_{24}, p_{45} \text{ and } p_{46}\} are the non-zero routing probabilities of the open QNM, which is assumed to operate under random routing. Thus, the following normalising conditions are satisfied: \(p_{10} + p_{12} = 1, p_{23} + p_{24} = 1 \text{ and } p_{45} + p_{46} = 1\).

The mean overall arrival rates \(\{\Lambda_i, ii = 1, 2, \ldots, 6\}\) satisfy the requests flow-rate linear equations of the stable open QNM of Figure 3.3, namely:

\[
\begin{align*}
\Lambda_1 &= \lambda \\
\Lambda_2 &= p_{12} \Lambda_1 \\
\Lambda_3 &= p_{23} \Lambda_2 \\
\Lambda_4 &= p_{24} \Lambda_2 + \Lambda_3 + \Lambda_5 \\
\Lambda_5 &= p_{45} \Lambda_4 \\
\Lambda_6 &= p_{46} \Lambda_4
\end{align*}
\]

Solving the above equations in term of \(\lambda \lambda\) and probabilities as a set of linear equations yields the following equations:

\[
\begin{align*}
\Lambda_1 &= \lambda \\
\Lambda_2 &= p_{12} \lambda \\
\Lambda_3 &= p_{23} p_{12} \lambda \\
\Lambda_4 &= \frac{p_{12}}{p_{46}} \lambda \\
\Lambda_5 &= \frac{p_{45} p_{12}}{p_{46}} \lambda \\
\Lambda_6 &= p_{12} \lambda
\end{align*}
\]

3.3.1 Performance Modelling and Analysis

Jackson’s open QNM was used to compute the system model. To achieve a steady state, this study ensures that the mean service time is always higher than the
mean arrival time. Likewise, the analysis was as a result of decomposing the network into individual stations [97, 137], using the general birth-death process Figure 3.4.

![Figure 3.4: General state transition rate for birth-death process](image)

A QNM is said to be in steady state when all its transient behaviour settled down. If all its performance measures are independent of time, then it is said to be in steady state. Similarly, the network is said to be in statistical equilibrium when the rate of the request \( in = rate \), at which request is served, in such situation the system is said to be stable [11].

The performance analysis of the OpenStack IaaS CCP architecture of Figure 3.2 is carried out at equilibrium by solving the proposed stable open QNM of Figure 3.3, which operates according to a random routing. It is assumed that the external arrival process of the QNM is Poisson and the service times at the queueing stations are exponentially distributed. Moreover, the multiple server queues have homogeneous servers and at each station has a first-come-first-served (FCFS) scheduling discipline.

By means of Jackson’s theorem [97, 137], the open QNM of Figure 3.3 can be decomposed into individual M/M/1 and M/M/c queues, each of which is analysed in isolation. Explicitly, stations 1, 3, 5 and 6 are modelled by M/M/1 queues, while M/M/c queues model stations 2 and 4, are \( c, (c > 1) \) (c.f., Figure 3.5).

![Figure 3.5: M/M/c queueing station with arrival rate \( \lambda \) and service time \( \mu \)](image)
3.3.1.1 Modelling and Analysis of the Stable M/M/c Queueing Stations 2, and 4

The queueing stations 2 and 4 of the open QNM are modelled as stable M/M/c \((cc > 1, \lambda \lambda < cc\mu)\) queues. The analytic expressions for the performance metrics of Table 3.1 for a stable M/M/c \((cc > 1)\) queue are presented below (c.f., [11-13]).

The probability of having \(n\) requests in the station

\[
P_{mn} = \begin{cases} 
PP_0 \frac{\lambda \lambda}{(ii+1)\mu} \cdot \frac{1}{nn!}, & 0 \leq nn \leq cc \\
PP_0 \frac{\lambda}{(ii+1)\mu} \cdot \frac{cc\mu}{cc}, & nn > cc 
\end{cases}
\]

with an individual stations utilisation.

\[
\rho p = \frac{\lambda \lambda}{cc\mu} \tag{3.14}
\]

Therefore,

\[
P_{mn} = PP_0 \frac{cc\rho}{nn!} 0 \leq nn \leq cc
\]

\[
P_{mn} = PP_0 \frac{cc\rho cc}{cc!} nn > cc
\]

where

\[
\rho_0 = PP_0 \frac{(cc\rho)^{cc}}{cc!} + \frac{(cc\rho)^{cc}}{cc!} \cdot \frac{1}{1-\rho p}^{-1}
\]

and

\[
P_{cc} = PP(NN \geq cc) = PP_{mc} = \frac{(cc\rho)^{cc}}{cc! (1-\rho p)} PP_{mn}
\]

Mean response time

\[
\mathcal{T} = \frac{cc\rho}{\lambda \lambda} + \frac{\rho p}{\lambda \lambda (1-\rho p)} \cdot p \tag{3.18}
\]

Mean waiting time

\[
\mathcal{W} = \frac{\rho p}{\lambda \lambda (1-\rho p)} \cdot p \tag{3.19}
\]

Mean number of requests in each station

\[
\mathcal{K} = cc\rho + \frac{\rho p}{1-\rho p} \cdot p \tag{3.20}
\]
### 3.4 Numerical Experiments and Results

**Mean queue length**

\[
\bar{Q} = \frac{\rho \cdot p}{1 - \rho \cdot c}
\]

**Mean system throughput**

\[
\lambda_T = \lambda_{T_1} + \lambda_{T_6} = \lambda_i
\]

### 3.3.1.2 Analysis of Stable M/M/1 Queueing Stations 1, 3, 5 and 6

The queueing stations 1, 3, 5 and 6 of the open QNM are modelled as stable M/M/1 queues and analysed by using the performance metrics of the stable M/M/c queue (c.f., Table 3.1) with one server (\(c = 1\)).

### 3.4 Numerical Experiments and Results

In this section numerical experiments are carried out to compute some of the main performance metrics and predict the presence of bottleneck queueing stations, subject to an extensive parameterisation focusing on the external mean arrival rate \(\lambda\), the routing probabilities \(p_{iii}\) and the external mean service rates \(\{\mu_i\}\) of the queueing stations of the open QNM of Figure 3.3.

The parameterisation was carried out by assigning values to the routing probabilities based on chances of splitting the traffic flows of the QNM, as appropriate. For example, the allocated probability of dropping a request is 0.2, while the probability of passing it to station 2 is 0.8. Similarly, the probability of a request passing through the admin node is set to 0.3. This is justified because, in typical settings of a small OpenStack, most of the compute nodes are all powered-on and configured. Typical values are also allocated for the mean service rates \(\{\mu_i\}\) at each station, which can be seen in Table 3.2 together with the allocated values for the non-trivial routing probabilities (< 1). Note that the values of the mean external arrival rate are set from 0.1 to 2.0, with an interval of 0.1 amounting into 20 sample values. The results have a 95% confidence interval and less than 5% relative error for each execution.
3.4 Numerical Experiments and Results

Table 3.2: Numerical values of routing probabilities and mean service rates for the open QNM of Figure 3.3

<table>
<thead>
<tr>
<th>Probabilities</th>
<th>Values</th>
<th>Mean Service Rates</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{10}$</td>
<td>0.2</td>
<td>$\mu_1$</td>
<td>2.5</td>
</tr>
<tr>
<td>$p_{12}$</td>
<td>0.8</td>
<td>$\mu_2$</td>
<td>0.6</td>
</tr>
<tr>
<td>$p_{23}$</td>
<td>0.3</td>
<td>$\mu_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$p_{24}$</td>
<td>0.8</td>
<td>$\mu_4$</td>
<td>0.7</td>
</tr>
<tr>
<td>$p_{45}$</td>
<td>0.15</td>
<td>$\mu_5$</td>
<td>0.4</td>
</tr>
<tr>
<td>$p_{46}$</td>
<td>0.85</td>
<td>$\mu_6$</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Without loss of generality, the overall mean response time, $T\hat{\theta}$, the overall mean number of requests, $\hat{N}$, the overall mean waiting time, $\hat{w}$, and the overall mean queueing time, $\hat{h}$, are defined by the sums of the mean response times, $\hat{\theta}$, mean number of requests, $\hat{N}$, mean waiting times, $\hat{w}$, and mean queueing times, $\hat{h}$, at each queueing station $I$, $I = 1, 2, \ldots, 6$, respectively. Figure 3.6 – 3.9 show the plots of the overall mean response time $T\hat{\theta}$, mean number of requests, $\hat{N}$, mean system waiting time, $\hat{w}$, and overall mean queue length $\hat{h}$ of the open QNM vs. the external mean arrival rate $\lambda\lambda$, respectively. It can be observed, as the mean arrival rate, $\lambda\lambda$ is progressively increasing, all the four aforementioned overall metrics, as expected, also increase and burst towards $+\infty$ as $\lambda\lambda \to c\mu\mu$, $(c \geq 1)$. The arrival rate starts at a low rate and hence there was no much significant effect on the performance metrics. As the rate is increasing gradually and burst towards higher value, each of the metrics will saturate, which will lead to the failure or freezing of the operation. Parameters for computing performance metrics at stations such as $\hat{N}$ and $\rho \rho$ will show the weakest node, which will determine which station’s components needs an upgrade or replacement with higher specification one.

The overall performance metrics of the open QNM of the entire OpenStack IaaS data centre and those at each queueing station can assist cloud engineers in identifying the bottleneck(s) stations of the CCP at both the design stage as well as prior to the tuning and upgrading of the architecture of the OpenStack IaaS CCP.
3.4 Numerical Experiments and Results

Figure 3.6: Overall system response time ($TT$) vs. $\lambda$

Figure 3.7: Mean number of request in the system vs $\lambda$

Figure 3.8: Mean system waiting time ($W$) vs $\lambda$

Figure 3.9: Mean number of requests in the queue ($Q$) vs $\lambda$
3.4.1 Bottleneck Analysis

Figures 3.10 – 3.12 show the utilisation of the Firewall (station 1), Controller (station 2) and Storage server (station 5) respectively vs increasing values of the mean external arrival rate, $\lambda \lambda$. It can be observed that for every value of $\lambda \lambda$, station 2: Controller has higher utilisations than those of station 1: Firewall and station 5: Storage. As these stations begin to reach higher utilisation values for more extreme values of $\lambda \lambda$, it is noted that the station 2: Controller has the highest utilisation equal to 0.9 while the station 1: Firewall and station 5: Storage reach their highest utilisations at 0.8 and 0.7 respectively. From these typical numerical experiments, it is implied that the station 2: Controller (station 2) is the bottleneck station of the open QNM and thus, it is the one that imposes limits on the performance levels of the CCP. Further value increases of $\lambda \lambda$, will force the Controller, the bottleneck station of the CCP, to reach saturation point (i.e., to operate with utilisation 100%) and thus, degrade the CCP’s overall performance. As long as the bottleneck stations are predicted either at the design stage or during a capacity planning process of the CCP, appropriate steps should be taken to add on time additional resources at those stations and thus, avoid incoming catastrophic CCP performance degradation as the workload rate $\lambda \lambda$ progressively increases.

Future work will extend the proposed open QNM in order to incorporate bursty traffic flows of requests, which can be modelled by the generalised exponential distribution (GE) [22-24] and other models [142].

![Figure 3.10: Server utilisation at station 1: Firewall vs. $\lambda \lambda$](image)
3.5 Summary

A stable open QNM with an external Poisson arrival process and exponential inter-arrival and service times was proposed for the performance modelling and bottleneck analysis of an OpenStack IaaS CCP with single and multiple server stations. Employing Jackson’s theorem, this model was decomposed into individual stations represented by M/M/1 and M/M/c, \((c \geq 1)\) queues each of which was analysed in isolation. To this end, the main performance metrics were computed as well as the determination of the bottleneck stations.
4 PERFORMANCE MODELLING AND ANALYSIS OF OPENSTACK IaaS CCP USING GE-TYPE QNM

4.1 Introduction

Cloud computing technology continues to gain grounds as the technology enters its maturity level. However, QoS is the prime issue affecting the SLA [143-150] mainly at the IaaS layer [2]. The IaaS computing model provides infrastructure resources, such as hosts, servers, networks, storage and virtualisation leased to the subscribers with complete control of the host OS image, virtualisation technology, middleware and application layer. Also, in the IaaS business model, the services abstract the user from the details of the underlying resources, such as location data, physical computing resources, data partitioning, and security backup [32]. Hypervisors such as kernel virtual machine for Linux, VMware, and Containers facilitates the provisioning of the IaaS resource [28]. Subscribers have complete control of the VM instance with the liberty to deploy guest OS from providers’ application store or upload it from the customer premises. Amazon web service’s (AWS) Elastic Compute (EC2), Microsoft Azure, Google cloud platform, and IBM are some of the famous examples of IaaS CCP providers [1]. The providers mentioned above have established dominance in the market with proprietary services. As these platforms are for commercial use, understanding a typical setup and topology was not possible, to let alone computation of their performance. However, the development of emerging competition from the open-source platforms such as Eucalyptus, OpenNebula and OpenStack [151] made it possible to understand the internal structure, components and services that help in studying various QoS within CCP architectures.

One of the most recent developments affecting system performance is the rapidly evolving nature of CCPs, which is commonly built on virtualisation technology [152]. HA, high performance, scalability, reliability, security and privacy are some of the essential attributes of QoS that service providers claim to offer to their subscribers [153].
Cloud Supported Platforms (CSPs) reports several performance-related issues that adversely affected the QoS. Reliability, performance, security and other related problems have attracted research community’s attention from both industry and academia [7, 8, 114]. HA is one of the exciting attributes of QoS with focus on ensuring 99.999% availability, and thus, performance is a crucial attribute of achieving it [7, 8]. For example, provisioning of cloud resources in the form of VM for different kinds of workloads depends on the QoS requirements defined in the SLA. Based on users’ specification of the required resources, the identification and assignment of an optimal workload-resource pair are of paramount importance in achieving acceptable levels of workload performance [9]. As the CCP uses virtualisation technology that enables sharing of collocated resources, performance modelling and analysis is required in order to ensure efficient delivery of HA cloud computing resources and acceptable QoS to satisfy the SLA. However, most of the analytical performance evaluation models are of limited applicability as they are based on ‘classical’ open Jacksonian QNMs with periodic or constant arrival processes and exponential service times. Other methods involve the use of bursty arrival processes with more realistic variability, as seen in [106]. However, the reported bursty approach is conventional and adopts a family of $H_2$ distributions.

None of the approaches studied has considered the deployment of a typical CCP architecture, with main components of the platform that provide different types of actual services.

This chapter proposes extended application-driven research for the performance modelling and analysis of OpenStack IaaS CCP architecture [46, 154] as follows:

a) Adopted the typical setup architecture of OpenStack CCP system from earlier work [46], comprising networked servers that host different OpenStack component, to develop an innovative open QNM at equilibrium with random routing and a First-Come-First-Served (FCFS) rule. It is assumed that the QNM has a bursty external arrival process of clients requesting VMs according to a bursty Compound Poisson Process (CPP) with geometrically distributed batches, or equivalently, variable Generalized Exponential (GE) interarrival times.
Moreover, it consists of $L \{L \geq 1\}$ queueing stations with infinite capacity and $c_i \{c_i \geq 1, i = 1, 2, ..., L\}$ GE-type servers for the provisioning of VMs, as appropriate.

b) Analysis based on a generic Maximum Entropy (ME) product form approximation, the proposed open stable QNM is decomposed into $L \{L \geq 1\}$ individual $GE/GE/c_i$, $i = 1, 2, ..., L$ queueing stations with GE-type external interarrival and service times, each of which can be analysed in isolation.

c) Closed-form expressions for key performance metrics are devised, such as those for the throughput, server (resource) utilisation, mean response time and steady-state probability for the processing and responding to the requests for VM provisioning at each queueing station. Moreover, typical numerical computations are conducted in order to identify performance characteristics and

d) Determination of bottleneck server(s) of a typical snapshot of queues within the CCP architectures.

Such performance predictions should help the cloud service providers in planning and designing new infrastructure as well as tuning/upgrading existing CCP architectures for efficient delivery of HA service during the provisioning and allocation of VMs instances.

The rest of the chapter is structured as follows: Section 4.2 provides ME principles and formalism. Section 4.3 presents the basics of GE distribution, while subsection 4.3.1 presents a matching of $H_2$ distribution into GE. Section 4.4 describes the proposed stable open GE-type QNM model of an OpenStack IaaS CCP Architecture for the provisioning of VMs. Subsections 4.5.1, 4.5.2 and 4.5.3 respectively presents the traffic flow analysis of the stable GE-Type QNM, analytical computation of $GE/GE/c_i$ ($c_i \geq 1$), and the computational algorithm for the analysis of the open stable GE type QNM. Numerical experiments focusing on key performance metrics as well as a bottleneck analysis of the CCP are presented in section 4.6. Chapter’s conclusions and remarks on future work extensions follow in section 4.7.

4.2 Maximum Entropy (ME) Principle

The principle of maximum entropy (ME) rooted from the principle of "insufficient reason" developed by Bernoulli in the early 18th century. It infers that the assignment of probability is a state of knowledge, and the outcomes of an event are considered
equally probable in the initial state unless evidence shows otherwise. That was the motivation behind the ME, which was discussed in details by Jaynes [155, 156]. ME principle provides a self-consistent inference method for the estimation of unknown probability distribution using the known information of mean value constraints based on the concept of functional entropy in information theory developed by Shannon [157].

The early application of ME was in the Markovian queueing systems that determine the stationary queue length distribution (QLD) of the M/M/1/N queue. ME steady-state probability also prove to corresponds to the open Jacksonian queuing network subject to the marginal mean queue length (MQL). Furthermore, Bard [158] proposes the analysis of contention in the input/output subsystems using the ME solutions, which also turn out to be identical to the QLD of the stable M/M/1 queue. These also motivate the independent development of the new ME QLD for a stable M/G/1 queue subject to MQL and utilisation constraints [159].

ME solution was also expressed as a product of Lagrangian coefficients corresponding to the constraint of normalising constant. Moreover, in the context of information theory [18] ME solution corresponds to the state of maximum system disorder and therefore considered a least bias distribution estimate for all the solution that satisfies the system constraint. Moreover, ME solution can be experimentally realised in overwhelmingly more ways than any other distribution in the context of sampling terms, given the imposed constraints [160]. Significant discrepancies between the experimentally observed distribution and the ME distribution shows that the essential physical constraints have been overlooked. Equally, experimentally the ME solution represents evidence of the proper identification of the system constraints.

In systems modelling field, the predictable performance distributions values such as those representing the number of requests at each queueing station may exist and, therefore, can be inherent constraints utilised in ME solution. On the other hand, classical queuing theory applied using the first two moments on jobs arrival rates (or interarrival times) and service times distributions to establish these expected solutions analytically. An exact or approximate implementation of the efficient analytic ME solution, however, may require a prior Lagrangian coefficients
estimate via asymptotic and/or exact expressions that involve the system parameters [161]. Note that using classical queueing theory to determine the performance distributions may be difficult even with the simpler systems or networks. Thus, making the ME solution more appropriate in characterising the use of information-theoretic approximations of performance distributions QNM.

To evaluate an arbitrary open QNM, the joint-state probability, subject to marginal mean value constraints, of the ME solution can be considered as a product-form approximation. Therefore, the decomposition of QNM with various degree of complexity into individual queueing stations can be achieved using the ME principle each of which can be analysed in isolation with revised interarrival and service times. Moreover, the marginal state probability of ME solution of the single queue system, using an appropriate first two moments formula for the effective flow, can be a cost-effective part of analytic building block for the computation of the systems performance metrics [22].

The ME solution becomes exact when the underlying service-time distribution is represented by the Generalised Exponential (GE) distribution [24]. A detailed description, including probability density function for GE-type distributions, is presented in the next section.

### 4.3 Generalised Exponential (GE) Distribution

The computational implementation of the ME solutions for general QNMs has been largely achieved by using the GE-type distributions [22, 23, 159, 162, 163]. The GE-type distribution (c.f., Figure 4.1) is described in the following form (c.f.[22-24]):

\[
F_F(t) = P(X \leq t) = 1 - \tau e^{-\sigma t} \quad t \geq 1
\]

where \(X\) is the random variable of interevent-time, \(\tau = 2/(\sigma^2 + 1)\), \(\sigma = \tau \tau\), and \(1/\nu\), \(\sigma^2 = \nu \nu/(\mu^2)\), \(\sigma^2 > 1\) are the mean and SCV (i.e. variance \(\nu \nu / (\mu^2)\)) divide by squared of mean \(EE^2(XX)\) of the interevent time distribution, respectively. Note that measurements of actual traffic or service times in complex queueing systems are generally limited and only a few parameters, such as mean and variance, can be computed reliably. In this case, the choice of a GE distribution - which is completely determined in terms of its first two moments - implies the least bias [22, 24]. The GE
distribution is versatile, possessing pseudo-memoryless properties, which makes the solution of many GE-type queueing systems analytically tractable.

\[
1 - \tau = \frac{C^2 - 1}{C^2 + 1} \\
\tau = \frac{2}{C^2 + 1}
\]

Figure 4.1: The GE-type distribution with parameters \(\tau\) and \(\sigma\) (0 \(\leq\) \(\tau\) \(\leq\) 1) (c.f., [23])

Moreover, it has been experimentally established that the GE model, due to its extremal characteristics, defines performance bounds over corresponding solutions based on two-phase H2 distributions with the same first two moments as the GE (c.f., [23]). The GE distribution can be interpreted as ME solution, subject to the constraints of normalisation, discrete-time zero probability and expected value. In this sense, it can be viewed as the least biased distribution estimate, given the available information in terms of the constraints. Furthermore, the cumulative density function of GE distribution is interpreted as the interevent time distribution of a compound Poisson process (CPP) with geometrically distributed batch sizes with mean, \(1/\nu\) (c.f., Figure 4.2, [23, 24]).

Specifically, the GE-type distribution has a corresponding counting process equivalent to a CPP with parameters \(2\tau\nu/(CC^2 + 1)\) and geometrically distributed bulk sizes with mean \((1 + CC^2)/2\) and SCV, \((CC^2 - 1)/(CC^2 + 1)\) given by:
where $N_{cp}$ is a CPP random variable of the number of events per unit time corresponding to a stationary GE-type interevent random variable.

The motivation behind the choice of the GE distribution was due to the limitations of the measurements of actual interarrival or service times with only little reliable computed parameters. Typically, the mean and variance are the only reliable parameters and hence make the distribution as the least bias one (i.e., the introduction of arbitrary and, therefore, false assumptions) is that of GE-type distribution.

Moreover, under renewality assumptions, the GE distribution is most appropriate to model simultaneous job arrivals at output node queues produced by various bursty sources with known first two moments. Here, the burstiness of the arrival process is achieved by the interarrival time SCV or, equivalently, the mean size of the incoming bulk (c.f., Figure 4.2).

### 4.3.1 Matching H2 as GE Distribution

Modelling flow of traffic in most modern systems such as telecommunication network uses distributions such as the Weibull, lognormal, and Hyperexponential. These distributions’ common characteristic is the coefficients of variation are greater than 1 ($CC > 1$) for certain values of parameters, which implies that the probability of large values of the random variable is much higher than for the classical exponential distribution. A Hyperexponential distribution is a continuous probability distribution with density function showing phases (c.f., Figure 4.3) given by equation (4.3).

$$
n = \sum_{i=1}^{m} \lambda_{ii} \cdot \pi_{ii} \cdot e^{-\lambda_{ii} x} = \sum_{i=1}^{m} \lambda_{ii} \cdot \pi_{ii} \cdot e^{-\lambda_{ii} x}$$

where parameters $\lambda_{ii}$ and $\pi_{ii}$ (c.f., Figure 4.3) are the job flow rates and phase transition probability that $x$ will take an exponential distribution on $Y$ random variable,
which is also exponentially distributed. Hyperexponential distribution also has families that are mainly determined by a number of phases.

![Hyperexponential distribution with phases.](image)

A two-phase Hyperexponential $(h_{yypp}(pp, \lambda_1, \lambda_2))$ distribution, in particular, describes a random variable characterised by a higher variability than exponential distribution with the same mean [11-13, 164]. It is the result of a weighted sum of two independent exponentially distributed random variables with parameters $\lambda_1$ and $\lambda_2$ respectively. The weight $pp$ is the probability that the random variable behaves like the exponential variable with parameters $\lambda_1$ and $1 - pp$ is the probability that it behaves like the exponential variable with parameter $\lambda_2$. The $H_2$ probability density function (PDF) is defined as:

$$f(x) = pp\lambda_1e^{-\lambda_1x} + (1 - pp)\lambda_2e^{-\lambda_2x}$$

When this distribution is used to model job interarrival time or a station service time, with probability $pp$ the next interval before a new arrival (or the next service time) is distribution. Recall, the GE-type PDF is given by [23].

$$\phi\phi(tt) = \frac{c^2 - 1}{c^2 + 1} \mu_0(tt) + \frac{4\pi}{(c^2 + 1)^2} e^{\frac{2\pi}{cc+1}}, tt \geq 0$$
In addition, GE is defined as the extremal member of a family of two-phase Hyperexponential model with the first two moments, and the PDF of the universal form is given as:

\[ ii(t, k) = a_1(k)\tau_1(k)e^{-\nu_1(k)t} + a_2(k)\tau_2(k)e^{-\nu_2(k)t} \]  \hspace{1cm} (3.6)

where \( t \) is the \( H_2 \) random variable and \( k \) is the set of tuning parameters, which is used to generate the \( H_2 \) distribution. Similarly, \( K \in (1, \infty), a_1(k), a_2(k), \tau_1(k), \tau_2(k) \) and \( \tau_2(k) \) are the matching values from (4.6) as \( a_1(k) = pp, \tau_1(k) = \lambda_1, a_2(k) = 1 - pp, \) and \( \tau_2(k) = \lambda_2 \). Extended description are presented in Appendix B [23].

![Hyperexponential distribution](image)

**Figure 4.4:** Approximating (Matching) \( H_2 \) to the GE distribution.

GE distribution was also defined as an extremal case of the family of two-phase exponential distributions (e.g. Hyperexponential-2 \( H_2 \)) having the same mean and SCV, where one of the two phases has zero service time. This makes GE as the limiting case of \( H_2 \) with a tuning parameter \( k \) going towards infinity. That is the higher the value of \( k \), the closer \( H_2 \) is to GE. Therefore, when \( k \to \infty, H_2 \approx GE \) (c.f., Figure 4.6).

### 4.4 A Stable Open GE-type QNM Model for VM Provision in OpenStack IaaS CCP

A typical architecture for an OpenStack IaaS CCP is displayed in Figure 4.5. It consists of three network types: public network, admin network and high-speed data network that facilitates communication and sharing the resources of the CCP. It also includes i) Controller nodes, which operate using other services component that resides as subsystems within controller nodes, i.e. gathering requirements for VM
instance creation. ii) Compute nodes (which host the VM instances created for the compute service), and iii) Storage node (responsible for housing/hosting the VM instance volumes, i.e., HDD) [41]. As this architecture is an extension of the one used in the previous chapter and hence the more accurate representation of the actual OpenStack services, Admin Node was not used which made the number of nodes to be five. However, the computation procedure is similar with only difference in the analysis as the distributions used were different as explicitly specified in the rest of the chapter.

![Figure 4.5: Typical OpenStack private CCP setup](image)

It has been often assumed in earlier works incorporating an open stable QNM of OpenStack IaaS CCP architecture that

i. The external and internal interarrival times, as well as service times of the open QNM, follow the traditional M-type exponential distribution;

ii. The external arrival pattern at each queueing station of the Open QNM complies with the Poisson Process or equivalently, exponential external interarrival times.

These underlying assumptions, however, restrain the credible computation of performance metrics of the open QNM due to the inherent optimistic nature of the associated analytic performance bounds (i.e. best-case scenario) for an open QNM.

This motivates the adoption of the GE-type distribution for both interarrival and service times. Typically, the mean and variance may be the only reliable available metrics in a CCP domain. To this end, the GE distribution, which is completely defined by its first two moments, may be applied as the least biased distribution
towards a more reliable computation of performance metrics. Note that the GE distribution is the extremal member of the family of $H_2$ distributions with the same mean and SCV (c.f., [23]).

As presented in [23], the open stable GE-type QNMs with general topology provide pessimistic performance bounds in comparison to those of its corresponding $H_2$-type family QNMs. Thus, for an open stable QNM of an OpenStack IaaS of a multi-node CCP, the GE distribution is particularly applicable by cloud computing engineers who are more interested in worst-case performance scenarios.

The proposed open stable GE-type QNM of an OpenStack IaaS CCP architecture in conjunction with the associated traffic flow analysis and approximate decomposition of into individual queueing stations can be found in the next section.

### 4.5 The Proposed Stable Open GE-type QNM for OpenStack IaaS CCP

The proposed stable open GE-type QNM of the OpenStack IaaS CCP architecture is displayed in Figure 4.5. It has an external arrival CCP and consists of five queueing stations with FCFS service discipline namely Firewall, Controller, Compute, Storage and Output.

![Figure 4.6: Schematic diagram for an open GE-type QNM of OpenStack VM provisioning requests with $cc_1, cc_4, cc_5 = 1; cc_2, cc_3 > 1$](image)

Each of these stations has either a single server or multiple homogeneous servers. It is assumed that the stations are connected via random routing and each
of them provide GE-type service times with mean service rates \( \{\mu_i, I = 1, 2, \ldots, 5\} \) and SCVs, \( \{CC^2_i, I = 1, 2, \ldots, 5\} \). Moreover, \( cc_6 (cc_6 \geq 1) \) is the number of servers in each queueing station such that \( \{cc_1 = 1, cc_2 > 1, cc_3 > 1, cc_4 = 1, cc_5 = 1\} \) (note, for \( CC^2_6 = 1 \), the GE-type distribution reduces to an M-type exponential distribution). Finally, the parameters \( \{pp_{10}, pp_{12}, pp_{34}, and pp_{35}\} \) (c.f., Figure 4.6), are the non-zero and \( \leq 1 \) routing (or, transition) probabilities of the stable open GE-type QNM (Note, ‘cloud sign’ indicates the population of clients on the Internet).

In this context, the clients generate via the Internet requests to the CCP for VM provisioning service according to an external arrival of CPP with geometrically distributed batches, having an equivalent GE-type external interarrival times with mean, \( 1/\lambda \lambda \) and SCV, \( \{CC^2_\alpha \geq 1\} \), where, \( CC^2_\alpha \) reflects the burstiness of the external arrival process [22, 24]. Note that, a request seeks service from the OpenStack IaaS CCP relating to the provision of a VM instance. Initially, the client’s request passes through a Firewall (station 1), where it is assumed that some requests will be dropped, based on some authentication criteria. The request will then be forwarded to the Controller node (station 2). The Controller will process the request internally by calling its services as described in Steps 2 – 4 of subsection 2.2.3.1. The Controller node will then forward the request to the Compute node (station 3), which will start the process of VM provisioning. The Compute node will create the VM if the detail specification of the request does not require external (distributed) storage. Otherwise, the request will be sent to the Storage node (station 4), which will process it by allocating the required storage volume. The Output server (station 5) will either send back the successful creation response or otherwise an appropriate message to the requesting client (e.g. an error or exception message). The detailed flow is described in subsection 2.2.3.1.

The performance analysis of the OpenStack IaaS CCP architecture is carried out by focusing on the quantitative analysis of the proposed stable open GE-type QNM of (c.f., Figure 4.5). It is assumed that both external interarrival and service times follow the GE-type distribution and each queueing station is represented as a stable \( GE/GE/c (c \geq 1) \) queue.

By means of the ME product-form approximation (c.f. [23] ), Figure 4.6 and [22, 24]), namely:
\[ pp(n_{1}, n_{2}, n_{3}, n_{4}, n_{5}) = \prod_{i=1}^{5} pp_{ii}(n_{ii}) \]  

(3.7)

where \( n_{ii}, ii = 1, 2, \ldots, 5 \) is the number of requests at queueing station \( i \).

The open QNM of Figure 4.5 can be decomposed into individual \( GE/GE/1 \) and \( GE/GE/c \) queues, each of which can be analysed in isolation [22, 24]. Specifically, stations 1, 4 and 5 are modelled by \( GE/GE/1 \) queues, while stations 2 and 3 are represented by \( GE/GE/c \) queues, where \( c > 1 \) (c.f., Figure 4.6).

![Figure 4.7: Decomposition of the stable open GE-type QNM into individual GE-type queueing stations](image)

The performance analysis of the OpenStack IaaS CCP architecture for the VM creation service requests is carried out by using the corresponding analytic traffic flow expressions for GE-type open QNMs with arbitrary configuration reported in
Kouvatssos [22, 24]. Consequently, the overall mean rates and SCVs of all the traffic flow at each queueing station $ii$, $ii = 1, 2, \ldots, 5$ of the proposed stable open GE-type QNM are determined in terms of the input parameters. Specifically, these traffic flows relate to the overall mean rates and SCVs of (i) merging of interarrival times $\{\lambda_{aii}, CC_{aii}\}$, (ii) interdeparture times $\{\Lambda_{ddii} = \Lambda_{ii}, CC_{ddii}\}$ and interdeparture splitting times $\{\Lambda_{ddiii}, CC_{ddiii}\}$ for $ii = 1, 2, \ldots, 5; j = 0, 2, 4, 5$.

Note that the input parameters assumed to be known are those of the (i) external interarrival times $\{\lambda_i, CC_i\}$ and (ii) service times $\{\mu_i, CC_i\}$, $ii = 1, 2, \ldots, 5$ as well as (iii) non-zero routing (transition) probabilities $\{pp_{10}, pp_{12}, pp_{34}, \text{ and } pp_{35}\}$. In this context, the aforementioned mean rates and SCVs are determined by the well-known method for computing departing, splitting and merging of GE-type streams. The analysis employs the ME product-form approximation (c.f., [22, 24]), the credibility of which has been experimentally verified via discrete event simulation for many different topologies of stable GE-type open QNMs (e.g., [22, 24]). This enables the decomposition of the proposed stable open GE-type QNM into individual GE/GE/ci ($c_i \geq 1$) queueing stations, each of which has bursty CPP external arrivals with geometrically distributed batches (or, equivalently, GE-type interarrival times) and can be analysed in isolation.

### 4.5.1 Traffic Flow Analysis of the Stable Open GE-type QNM

Regarding the schematic diagram of the open QNM of Figure 4.6, the $\{pp_{10}, pp_{12}, pp_{34}, \text{ and } pp_{35}\}$ are the non-zero routing probabilities of the open QNM, the traffic flow of which is assumed to operate under random routing. Thus, the following normalizing conditions are satisfied:

$$p_{10} + p_{12} = 1 \quad (3.8)$$
$$p_{34} + p_{35} = 1 \quad (3.9)$$

Moreover, the mean overall arrival rates $\{\Lambda_{aii}, ii = 1, 2, \ldots, 5\}$ at each stable queueing station $ii$, $ii = 1, 2, \ldots, 5$ with $cc_{ii}$ ($cc_{ii} \geq 1$) servers, comply with the principle of Flow Balance i.e., ‘Mean Flow Rate In to station $i = Mean Flow Rate Out of station ii, ii = 1, 2, \ldots, 5$. Clearly, the following mean flow rate linear equations of the stable open GE-type QNM of Figure 4.5 hold:
4.5.1 Traffic Flow Analysis of the Stable Open GE-type QNM

\[
\begin{align*}
\Lambda_1 &= \lambda \lambda \\
\Lambda_2 &= p_{12} \Lambda_1 \\
\Lambda_3 &= \Lambda_2 + \Lambda \Lambda_4 \\
\Lambda_4 &= p_{34} \Lambda_3 \\
\Lambda_5 &= p_{35} \Lambda_3
\end{align*}
\] (3.10)

\[
\begin{align*}
\Lambda_2 &= p_{12} \Lambda_1 \\
\Lambda_3 &= \frac{p_{35}}{p_{12} p_{34}} \lambda \lambda \\
\Lambda_4 &= \frac{p_{35}}{p_{12} p_{34}} \lambda \lambda \\
\Lambda_5 &= p_{12} \Lambda_1
\end{align*}
\] (3.11)

Solving the above system of linear equations in terms of \(\lambda \lambda\) and transition probabilities \(\{pp_{10}, pp_{12}, pp_{34},\text{ and } pp_{35}\}\), the following expressions for the overall arrival rates, \(\{\Lambda \Lambda_{ii}, ii = 1, 2, \ldots, 5\}\) are obtained:

\[
\begin{align*}
\Lambda_1 &= \lambda \lambda \\
\Lambda_2 &= p_{12} \Lambda_1 \\
\Lambda_3 &= \frac{p_{35}}{p_{12} p_{34}} \lambda \lambda \\
\Lambda_4 &= \frac{p_{35}}{p_{12} p_{34}} \lambda \lambda \\
\Lambda_5 &= p_{12} \Lambda_1
\end{align*}
\] (3.15)

Moreover, the values of the mean departure rates \(\{\Lambda_{ddii}, ii = 1, 2, \ldots, 5\}\) and SCVs of the interdeparture times, \(\{CC_{dd}^2, ii = 1, 2, \ldots, 5\}\) can be determined at each queueing station \(ii = 1, 2, \ldots, 5\) by using the GE-type traffic flow equations (c.f., Appendix A, [22, 24]). To simplify, without loss of generality, the computation of the SCV of the interdeparture process of a stable GE/GE/\(c_{ii}\) \((c_{ii} > 1)\) queue, a heavy-traffic approximation is adopted in order to consider the GE/GE/ \(c_{ii}\) queue as a GE/GE/1 queue with single exponential super-server with a mean service rate \(c_{ii} \mu_{ii}\) such that \(\Lambda \Lambda_{ii} < c_{ii} \mu_{ii}\) (c.f., [24]). To this end, the diagram of the departure process is presented below (c.f., Figure 4.8), where \(CC_{a}^2\) and \(CC_{dd}^2\) are the SCVs of the interarrival and inerdeparture times, respectively.

The stable \(GE/GE/c_{i} (c_{i} \geq 1), ii = 1, 2, \ldots, 5\) satisfies the stability condition \(\Lambda \Lambda_{ii} < c_{ii} \mu_{ii}\), \(ii = 1, 2, \ldots, 5\), where the mean throughput (or output rate), \(\Lambda_{ddii}\) for each queueing station, \(i\) is equal to \(\Lambda \Lambda_{ii}\) i.e. \(\Lambda_{ddii} = \Lambda_{ii}\). Note that for each queueing station \(i\), the following notation is employed. \(SCV_{arrival} = CC_{a}^2; SCV_{interdeparture} = CC_{dd}^2\) and \(SCV_{service\ time} = CC_{ii}^2\).
4.5.1 Traffic Flow Analysis of the Stable Open GE-type QNM

Figure 4.8: The departure process of heavy traffic approximation for stable $GE/GE/\alpha_i$ ($\alpha_i \geq 1$) queueing station $ii, ii = 1, 2, ..., 5$

It is assumed that the interdeparture times at each queueing station $ii, ii = 1, 2, ..., 5$ follow the GE-type distribution with mean departure rate and SCV (c.f., [24]). (c.f., (B2) and (B3) of Appendix A), respectively, given by:

$$\lambda_{ddii} = \lambda_{ii}$$  \hspace{1cm} (3.20)

$$CC_{ddii}^2 = pp_{ii} (1 - pp_{ii}) + CC_{dii}^2 (1 - pp_{ii}) + pp_{ii}^2 CC_{dii}^2$$  \hspace{1cm} (3.21)

However, the resulting server utilisation, $\rho_{ii}$ ($ii = 1, 2, ..., 5$) is modified as appropriate, where $c$ signifies the number of servers used for the numerical experiments in section 4.6 for the multi-server queueing stations 2 and 3., i.e.,

$$\rho_{ii} = \frac{\lambda_{ii}}{c_i \mu_i}, \hspace{1cm} ii = 1, 2, ..., 5$$  \hspace{1cm} (3.22)

Similarly, the mean and SCV of the splitting process of the QNM of Figure 4.6 are computed via the splitting flow expressions of Split1 (4.23) – (4.26) and Split3 (4.27) – (4.30) (c.f., (B4) and (B5) of Appendix B) where it implied $CC_{dd12}^2 = CC_{d2}^2, CC_{dd34}^2 = CC_{a4}^2$ and $CC_{dd35}^2 = CC_{a5}^2$. Thus, the two splitting streams are computed as follows:

Figure 4.9: Stream splitting at nodes (c.f., Split1 and Split3 in Figure 4.6)
4.5.1 Traffic Flow Analysis of the Stable Open GE-type QNM

Split 1

\[ \Lambda_{10} = p_{10} \Lambda_1 \quad (3.23) \]

\[ CC_{d10}^2 = 1 + pp_{10}(CC_{d1}^2 - 1) \quad (3.24) \]

\[ \Lambda_{12} = p_{12} \Lambda_1 \quad (3.25) \]

\[ CC_{d12}^2 = 1 + pp_{12}(CC_{d1}^2 - 1) \quad (3.26) \]

Split 3

\[ \Lambda_{34} = p_{p34} \Lambda_3 \quad (3.27) \]

\[ CC_{d34}^2 = 1 + pp_{34}(CC_{d3}^2 - 1) \quad (3.28) \]

\[ \Lambda_{35} = p_{p35} \Lambda_3 \quad (3.29) \]

\[ CC_{d35}^2 = 1 + pp_{35}(CC_{d3}^2 - 1) \quad (3.30) \]

Finally, the mean and SCV of the merging flow process are specified using equations (B6) and (B7) of the Appendix B. However, the yielded merged equation of the QNM is presented in equation (4.32).

\[ p_{10}, \Lambda_1, C_{d10}^2 \rightarrow \Lambda_T, C_T^2 \rightarrow \text{Merge}_{1,5} \]

\[ p_{50}, \Lambda_5, C_{d5}^2 \rightarrow \Lambda_3, C_{d3}^2 \rightarrow \text{Merge}_{2,4} \]

Figure 4.10: Streams merging at node (c.f., Merge$_{1,5}$ and Merge$_{2,4}$ in Figure 4.6)

Applying the principle of the merging point, the effective interarrival parameters are at each station’s node, which is already defined in routing probabilities and arrival rates equations (c.f., (4.8) – (4.9) and (4.15) – (4.19) respectively).
Therefore, merging streams out of the network (Merge$_1$, 5) and to station 3 (Merge$_2$, 4) are respectively presented in equations (4.31) and (4.32):

\[
CC_1^2 = -1 + \frac{\lambda_{TT}}{CC_{dd10}^2 + 1} + \frac{M_{10}}{CC_{dd10}^2 + 1} \tag{3.31}
\]

\[
CC_2^2 = -1 + \frac{M_3}{CC_{dd2}^2 + 1} + \frac{M_4}{CC_{dd4}^2 + 1} \tag{3.32}
\]

Note the merging formula considered from Figure 4.6 is the Merge$_2$, 4 (c.f., (4.32) and Figure 4.10) because of Merge$_1$, 5 stream flows out of the system without branching in any station and hence not providing the needed value.

4.5.2 Analytical Computation of GE/GE/c$_i$ (c$_i$ ≥ 1)

The queueing stations 1, 2, ..., 5 of the stable open GE-type QNM (c.f., Figure 4.6) of typical OpenStack private CCP setup (c.f., Figure 4.5) are modelled as stable GE/GE/c$_i$ queues with c$_{ci}$ ≥ 1. The analytic expressions for the performance metrics of these GE/GE/c$_i$ queueing stations are presented below. For the analysis of the stable GE/GE/c$_i$ stations, a heavy traffic approximation is adopted in order to mitigate the associated analytic complexity. To this end, it is assumed that the stations with multiple servers, GE/GE/c$_i$ (c$_i$ > 1) operate as being in a heavy traffic period and thus, they act as single GE-type super server with a mean service rate c$_{ci}$$\mu_{ii}$, (c$_{ci}$ ≥ 1).

In this context, the formulae of the performance metrics of the mean number of requests \( N_{ii} \) and mean response time per request, \( W_{ii} \) for a stable GE/GE/1 queue are given by equations (4.33) and (4.34) (c.f., [24]).

\[
N_{ii} = \frac{\rho_{ii}}{2} + \frac{CC_{ddii} + \rho_{ii}CC_{ddii}}{1 - \rho_{ii}} \quad \text{for } ii \in {1, 2, ..., 5} \tag{3.33}
\]

where

\[
\rho_{ii} = \frac{\Lambda_{ii}}{\mu_{ii}} \quad \text{for } jj = 1, 4 \quad V
\]

\[
= \frac{\Lambda_{ii}}{CC_{ii} + \mu_{ii}} \quad \text{for } jj = 2 \quad V
\]

79
Moreover, by applying Little’s law [139], the mean response time \( \hat{\mu}_i \) for a stable GE/GE/1 (c.f., [24]) are given by
\[
\hat{\mu}_i = \frac{\mu_i}{\lambda_i} \quad \text{for} \quad i \in 1, 2, ..., 5
\]  

(3.34)

Next section provides an algorithm that describes detailed steps of computing the GE-type performance metrics.

4.5.3 The Computational Algorithm for the Analysis of GE-type QNM

The main steps of the computational algorithm for the analysis of the proposed stable open GE-type QNM of the OpenStack IaaS CCP architecture is highlighted below. This credible numerical algorithm was tested and verified in several earlier publications [22-24].

BEGIN

Step 1: Set the values of input data \( \{\lambda_i, \mu_i, \lambda_i^2, \mu_i^2, \lambda_i \mu_i \} \);

Step 2: Solve the request-flow balance linear equations of the stable GE-type, open QNM (c.f., (4.15) – (4.19)) to obtain the overall mean arrival rates of requests \( \lambda_i \) at each individual queueing station;

Step 3: Initialise the values of the SCVs \( \lambda_i \) and \( \lambda_i^2 \) for all queueing stations \( ii \);

Step 4: Solve the system of non-linear equations iteratively until convergence. For each queueing station \( ii \);

a) Calculate the values of interdeparture SCV, \( \{C_{i} \} \) (c.f., (4.21));

b) Calculate the splitting interdeparture formulae of SCV \( C_{i} \) (c.f., (4.23) – (4.30));

c) Calculate the overall (merged) SCVs of the interarrival times, \( \{C_{i} \} \) and (c.f., (4.32));
4.6 Numerical Experiments and Results

In this section, typical numerical experiments are carried out in order to assess the impact of evolving parameterisation on the computation of primary performance metrics of the proposed QNM of the OpenStack IaaS CCP architecture and predict the bottleneck queueing stations and their adverse implications on the virtualisation of the CCP architecture (c.f., Figure 4.6). Numerical computations and visualisation of the results are presented as follows:

The traffic flow system of non-linear equations, the departure, splitting and merging formulae of the GE-type are solved iteratively using MATLAB\(^{21}\) package to generate numerical values of the merged (overall) SCVs for the interarrival times \(\{CC^2_{ii}, ii = 1, 2, \ldots, 5\}\), (c.f., the flow formulae (4.20) – (4.32) and computational algorithm section 4.5.3). The computed values of \(CC^2_{ii}, ii = 1, 2, \ldots, 5\) and the assigned values of \(\{CC^2_{ii}, ii = 1, 2, \ldots, 5\}\) in Table 1 are used to compute the performance metrics such as Mean number of requests (\(\mu\)), Mean system throughput (\(\lambda T\)) and Mean system response time (\(\theta\)) among others.

For illustration purposes, the parameterisation of the experimental study focuses on the traffic flows of the proposed open GE-type QNM relating to the processing, provisioning and leasing of VMs to requesting clients. Moreover, it includes typical numerical values for i) external mean arrival rates with \(\lambda\) ranging from 10 to 100 ii) non-zero routing probabilities \(\{pp_{10}, pp_{12}, pp_{34}, pp_{35}\}\) iii) external interarrival-times SCV \(CC^2\) iv) mean service rates \(\{\mu_{ii}, ii = 1, 2, \ldots, 5\}\) and v) service times SCVs \(\{CC^2_{ii}, ii = 1, 2, \ldots, 5\}\) displayed in Table 4.1. These values were chosen as a natural traffic flow of light, medium and heavy traffic. Similar to the

\(^{21}\) https://www.mathworks.com/products/matlab.html
parameterisation values were applied in related work for modelling of different systems (e.g. [99]).

Table 4.1: Numerical values of Routing probabilities, Mean service rates and SCVs for the Open GE-type QNM (c.f., Figures. 4.12 – 4.18)

<table>
<thead>
<tr>
<th>Non-Zero Routing probabilities</th>
<th>Mean Service Rates</th>
<th>SCVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{10}=0.2, p_{12}=0.8$</td>
<td>$\mu_1 = 2.5, \mu_2=0.6$</td>
<td>$C_{20}^{z} = 10, C_{33}^{z} = 35$</td>
</tr>
<tr>
<td>$p_{34}=0.3, p_{35}=0.7$</td>
<td>$\mu_3 = 0.7, \mu_4=0.4$</td>
<td>$C_{22}^{z} = 45, C_{35}^{z} = 52$</td>
</tr>
<tr>
<td></td>
<td>$\mu_5 = 2.0$</td>
<td>$C_{24}^{z} = 40, C_{33}^{z} = 30$</td>
</tr>
</tbody>
</table>

Since the analytical formulae require several iterations to compute each performance metric for one case, it will be cumbersome with high prone to error to iterate many times manually to get the results. Consequently, the approximate analytic results of the open GE-type QNM were devised using the Java Modelling Tool\(^\text{22}\) (JMT) [165, 166] with input parameters of those in Table 4.1, as appropriate for each queuing station $ii, ii = 1, 2, ..., 5$;

The JMT package was used to compute the family of stable $HH_2(kk)$ distributions with tuning parameter $k \in (1, +\infty)$ in order to create open QNMs with $HH_2(kk)$ external interarrival and service times having the same first two moments (i.e., means and SCVs) as the GE distribution satisfying the limiting case $\text{GE} \cong \lim_{k \to +\infty} HH_2(kk)$ . Moreover, the algorithm specified in Appendix A and reported in [23] was used to match the moments of the $HH_2(kk)$ distributions to those of GE as discussed in section 4.3.1. The resulting matching values, as well as the assigned values (c.f., [24]), were used to compute open QNM of Figure 4.6 and obtain the main $HH_2(kk)$-type performance metrics (c.f., Figure 4.4). The results has a 99% confidence interval and less than 4 % relative error for each execution.

Moreover, the following tables (c.f., Table 4.2) show the values of the conversion (matching) of the $H_2$ values for the computation of the appropriate values used in the JMT to analyses the model. Remaining tables (c.f., Table 4.2) for different values of $k \in (5, 10, 20, 50, 100)$ are presented in Appendix C.

\[^{22}\text{http://jmt.sourceforge.net/}\]
4.6 Numerical Experiments and Results

Table 4.2: Values used to Match H₂ to GE for GE-type QNM analysis of OpenStack IaaS

<table>
<thead>
<tr>
<th>H₂</th>
<th>GE</th>
<th>Firewall</th>
<th>Controller</th>
<th>Compute</th>
<th>Storage</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ν:</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>C:</td>
<td>105</td>
<td>125</td>
<td>130</td>
<td>120</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>kk:</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A:</td>
<td>48.5000</td>
<td>61</td>
<td>63.5</td>
<td>58.5</td>
<td>56.0000</td>
<td></td>
</tr>
<tr>
<td>B:</td>
<td>50.9779</td>
<td>63.4823</td>
<td>65.983</td>
<td>60.9816</td>
<td>58.4808</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>α₁(k):</td>
<td>0.9849</td>
<td>0.988</td>
<td>0.9884</td>
<td>0.9875</td>
<td>0.9869</td>
</tr>
<tr>
<td>1-p</td>
<td>α₂(k):</td>
<td>0.0151</td>
<td>0.012</td>
<td>0.0116</td>
<td>0.0125</td>
<td>0.0131</td>
</tr>
<tr>
<td>λ₂</td>
<td>v₂(k):</td>
<td>0.3014</td>
<td>0.2409</td>
<td>0.2316</td>
<td>0.251</td>
<td>0.2619</td>
</tr>
</tbody>
</table>

For illustration purposes, Figures 4.11 – 4.15 show, respectively, for different values of the tuning parameter k, the plots of the mean number of requests $\{K_i^c, i = 1, 2, ..., 5\}$ vs the mean external arrival rate $\lambda \in (10, 100)$ for the Firewall (station 1), Controller (station 2), Compute (station 3), Storage (station 4) and Output (station 5) queueing stations of the stable open QNM of Figure 4.6.

Figure 4.11: Mean number of requests ($K_i^c$) in firewall (station 1) with different values of k vs. $\lambda \lambda$
4.6 Numerical Experiments and Results

Figure 4.12: Mean number of requests $\langle K \rangle$ in controller (station 2) of different values of $k$ vs. $\lambda \bar{\lambda}$

Figure 4.13: Mean number of requests $\langle K \rangle$ in compute (station 3) with different values of $k$ vs. $\lambda \bar{\lambda}$

Figure 4.14: Mean number of requests $\langle K \rangle$ in storage (station 4) with different values of $k$ vs. $\lambda \bar{\lambda}$
4.6 Numerical Experiments and Results

Figure 4.15: Mean number of requests $\langle k \rangle$ in output (station 5) with different values of k vs. $\lambda$.

Moreover, the mean overall response time, $\mu_{\bar{u}}$, is defined by the sums of the mean response times, $\mu_i$, at each queueing station $i$, $i = 1, 2, ..., 5$ of the open GE-type QNM. The GE-type performance metrics focusing on the overall mean throughput, $\lambda_{\bar{u}}$, and the overall mean response time, $\mu_{\bar{u}}$, of the open QNM of Figure 4.6 vs $\lambda \in (10, 100)$ are presented in Figure 4.16 and Figure 4.17, respectively.

Figure 4.16: Mean system throughput $\langle \bar{u} \rangle$ with different values of k vs. $\lambda$.

It can be observed, as the mean arrival rate, $\lambda \overline{\lambda}$ is progressively increasing, the $\mu_{\bar{u}}$ and $\lambda_{\bar{u}}$ metrics increase and burst, as expected, to $+\infty$ as $\overline{\lambda} \rightarrow c\overline{\mu}$, $c \geq 1$. As the tuning parameter $k$ increases, the multiple $H_2(k)$-type metrics continue to deteriorate, hence indicating the most pessimistic performance bounds when they go towards the worst-case scenario of GE-type performance bounds.
Numerical Experiments and Results

Figure 4.17: Mean system response time \( (T'') \) with different values of \( k \) vs. \( \lambda \)

Finally, Figure 4.18 displays higher overall (system) mean response times of the open QNM of Figure 4.6 vs \( \lambda \in [10, 100] \) for increasing values of the SCV of the external interarrival times, \( C_2 \). Thus, it is observed that larger values of SCV have an inimical effect on the performance of the CCP similar to the increased values of tuning parameter \( k \) and thus, on the efficient provisioning and allocation of VMs. This figure verifies the analytical computation of GE with the approximate computation of \( H_2(k) \) as \( kk \rightarrow \infty \) with more than 95% confidence interval and less than 4% relative error.

Figure 4.18: Mean system response time \( (T'') \) with different values of SCV vs. \( \lambda \)

The overall performance metrics of the stable open GE-type QNM of the entire OpenStack IaaS CCP and those at each queueing station can assist cloud engineers
Bottleneck Analysis

in predicting and identifying the bottleneck station(s) of the CCP at both the design stage as well as towards the tuning and upgrading of the OpenStack IaaS CCP architecture. As GE type QNM provides pessimistic performance bounds, worst-case performance scenario will be paramount important, especially at the design stage as well as during upgrading the existing platforms.

4.6.1 Bottleneck Analysis

Bottleneck analysis is one of the critical aspects of performance prediction for the efficient delivery of QoS in CCPs. This subsection includes the experimental results focusing on the assessment of utilisations \( \{\rho_{ii}, ii = 1, 2, \ldots, 5\} \) of the queueing stations of the core components vs \( \lambda \in [10, 100] \) of the CCP architecture as it is illustrated in Figure 4.19, respectively.

![Figure 4.19: Min and max utilisations \( \{\rho_{ii}, ii = 1, 2, \ldots, 5\} \) for all five stations with different values of k](image)

In the analysis of the open M-type QNM, Figure 4.19 show the utilisation of the Firewall (station 1), Controller (station 2), Compute (station 3), Storage (station 4) and Output (station 5), respectively at the mean external arrival rates, \( \lambda=10 \) and \( \lambda=100 \). It can be observed that at \( \lambda =10 \), the firewall has a higher (minimum) utilisation than those of the remaining nodes. As these nodes begin to reach higher utilisation values for \( \lambda=100 \) and under more extreme values of \( \lambda \), it is noted that firewall node (station 1) has the highest utilisation of 19.57% while controller, compute, storage and output stations reach their highest utilisations at 7.1%, 20.49%, 19.47% and 19.57% respectively. The same analysis is applied for the family of \( H_2(k) \) as well as the open GE-type QNM where a significant deterioration is observed with the value of the tuning parameter \( k = 100 \) with firewall node (station
1) having the highest utilisation at 96.6% as the value of λ progressively increases to 100 (c.f., Figure 4.19).

From both M-type and GE-type model’s typical numerical experiments, it implied that the firewall node (station 1) is the bottleneck station of the open QNM and, thus, imposes a limit on the overall performance level of CCP. The results show that as the mean arrival rate, λ, as well as the tuning parameter (k) increase, the bottleneck station of the CCP will reach saturation point (i.e., to operate with utilisation 100%) and thus, degrade the CCP’s overall performance and lead to catastrophic outage, which will breach the SLA. That will cause significant dropping of requests for VMs as well as a significant delay in both the provisioning of VMs as well as the replying response back to the requesting client. Therefore, further to the reactive measures of using monitors to alert the cloud administrators of any station that tends to reach a constant utilisation, more preventive measures should be adopted at the design stage. Our approach of determining the potential bottleneck device(s) correctly will drastically reduce chances of failure or incomplete transaction, which will contradict one of the essential attributes of CCP; Moreover, scalability will result in catastrophic consequences that will lead to SLA breach litigation.

As long as the bottleneck station(s) are predicted either during at the design stage of new CCP architectures at the design and development stages or at the capacity planning process of existing ones, appropriate steps should be taken to add on time additional resources to bottleneck station(s). This issue can be addressed by adopting a scalable firewall with more nodes, implementing a load-balanced firewall or upgrading it to a high specification system. In this way, as the workload rate λ progressively increases, an incoming catastrophic Open Stack IaaS CCP performance degradation will be avoided towards effective provisioning, placement and allocation of VMs.

4.7 Summary

An extended application-driven open GE-type QNM at equilibrium with \( L > 1 \) single and multiple server queueing stations \{i, i = 1, 2, ..., L\}, random routing and FCFS rule was proposed for the performance modelling and analysis of OpenStack
IaaS CCP architectures for the efficient provision of HA services to clients and live migration for VM instances. Based on a generic ME product-form approximation (c.f., [24]), the open QNM was decomposed into \( L, (L > 1) \) individual GE/GE/c\(_i\)(c\(_i\) ≥ 1), \( i = 1, 2, \ldots, L \) queueing stations with \( c_i \) servers and GE-type external interarrival and service times, each of which can be analysed in isolation. Consequently, closed-form expressions for key performance metrics of each queueing station of the open GE-type QNM were devised, such as those for the throughput and server (resource) utilisation, mean response time and the steady-state probability of the number of VM requests by clients.

Moreover, typical numerical experiments are carried out incorporating probability distributions consisting of an M-type, and a GE type of interarrival and service time distributions with the same first and second moments, as appropriate (c.f., Figures 4.11 – 4.18). It transpired that these experiments define respectively at each queueing station optimistic M-type and pessimistic GE-type analytic performance bounds, using the MATLAB tool, over those of the family of H\(_2\)(k)-type distributions with tuning parameter \( k \in (1, +\infty) \), generated via the JMT tool. The QNM’s performance metrics and the prediction of related bottlenecks provide vital insights for the efficient capacity planning of OpenStack IaaS CCP architectures as well as the design and development of new ones towards the effective processing and leasing of VMs to requesting clients.
5 PERFORMANCE AND SECURITY TRADE-OFFS IN OPENSTACK IaaS CCPs

5.1 Introduction

Computing and Information systems security is an area that rooted since the significant advancement in the electronic storage and processing of data and continues to evolve to date [167-170]. While information security essential concerns with the data and information, computing system security embodied the computing hardware, software, application, data, and services. Information security provides the mechanisms of ensuring the system and its resources are protected against any unauthorised access. It also ensures authorisation of access is granted and maintain confidentiality, integrity and availability of the resources [171] by using appropriate techniques. These techniques range from access control through sophisticated authentication and authorisation as well as masking of data through encryption technology [171-174].

The cloud computing ecosystem, including IaaS CCP, also requires a robust and adequate security mechanism to ensure the preservation of privacy, integrity and availability of resources. These mechanisms are especially critical as CCPs are built using multitenant, utility model and on top of virtualisation technology. To establish trust with subscribers, providers need to protect user privacy, including information secrecy from the adversaries, especially with cloud computing technology. Collaborative computation and data exchange need to be coordinated to guarantee adequate enforcement of authentication and authorisation in order to avoid leakage and unauthorised access to the resource. Cloud services customers require robust end-to-end protection of their resources within the platform for both external and internal access. Firewalls and encryption/decryption mechanisms are at the heart of protecting user data and ensuring the availability of the resources. However, applying strong protections through access control, encryption, and many other mechanisms have drawbacks, as they generate additional processing cycle and thereby, affecting other QoS components such as performance and HA [122, 175]. Hence, there is a need for 'optimal' security and performance trade-offs, that
is compromising security for improving performance and the other way round [127, 176]. In order to address the imbalance, quantitative modelling tools are required to optimise these trade-offs through analytic modelling and evaluation.

This chapter extends the study of OpenStack IaaS CCP to the investigation of the impact of security on the performance. In this context, a performance and security trade-off model is proposed for the CCP to examine and measure security vs performance. To this end, an optimal trade-off between performance and security will help in attaining and maintaining the SLAs. This will help in ensuring smooth processing and responding to the client request for a VM. A combined performance and security metric (CPSM) is formulated, and for illustration purposes. To enhance its computation, two CPSMs are used [127]. Then, a parameterisation and numerical experimentation with results are presented and interpreted.

The rest of the chapter describes the categories of system and cloud security Section 5.2. Section 5.3 presents selected literature on different trade-off models Security and performance metrics, are discussed in sections 5.4 and 5.5, respectively. In addition, the impact of security on performance was assessed in section 5.6. Section 5.7 presents a proposed combined performance and security trade-off model for the OpenStack CCP. Numerical experiments including parameterisation, results and interpretations are presented in Section 5.7.3. Finally, the chapter’s concluding remarks are presented in section 5.9.

5.2 Computer and Information System Security Overview

In the early 70s, securing data, which is usually a hard print, is not more than watching the copier and monitoring the users that go in and out of the ‘data processing centre’ also known as a computer room. Nowadays, the information system is revolutionising to an array of devices that can hold billions of data and transporting it across the network worldwide. Portable devices from laptops, desktops, smartphones and wearables hold an enormous amount of data creating millions of computer rooms, which makes the industry more vulnerable. With computer networks, including enterprise and cloud data centres, evolving by the day, securing the data, preserving its integrity, privacy, and making it available and accessible to only authorised users is challenging. As the cloud is built on sharing
its resources to simultaneous and heterogeneous users, providing the security at every layer of granularity is paramount. This is to not only attain the essential attributes of the data security but also maintain the SLA, in which violating the agreement may result in business losses and legal battle. Several challenges confront security professionals in almost every organisation.

System security or computer security is an umbrella term used to describe any form of mechanisms used to protect the information system from theft, damage, or access to the content. Computer and information security, on the other hand, can be described as the protection of an information system for the purpose of achieving its intended purpose of preserving the confidentiality, availability and integrity of the system resource such as hardware, middleware, software, communication systems, information and data as well as services among others [168, 171]. Confidentiality, Integrity and Availability are the three main widely used building blocks of computer and information security acronym known as CIA. Ensuring the confidentiality of computing resource (mainly data) is usually achieved by disguising the sensitive information through either encryption, controlling access, and authorising access to part or whole resources. To protect the integrity of the resources involves making sure that unauthorised entity does not modify the content of the resource. The integrity is achieved through intrusion detection and control mechanisms that deter and prevent alteration of resources (e.g., programs and data) content while the availability component ensures that all authorised parties gain timely access to the system resource according to the agreed access right and privilege [177].

Cloud security, on the other hand, concerns with all the components highlighted above with additional constraints due to its distinctive properties such as virtualise service layered architecture, multitenancy and scalability, among others. The abstracted service layers (IaaS, PaaS, and SaaS) provide additional security vulnerabilities and prone to both internal (between subscribed cloud users) and external (from outside the CCP) security incidents. For example, as the cloud is built on virtualisation technology at the IaaS layer, security vulnerabilities may be associated with hardware, storage, hypervisor and VMs. Any flaws or faults in the complicated code that builds the hypervisor that varies from platform to platform could compromise the isolation between VMs hosted by the same PM. VM
snapshots, provisioning, live migration and other distinctive features pose vulnerabilities and susceptible to attacks, such as the denial of service (DoS) attack resulting in data disclosure and integrity compromise. Despite these and many other security threats to CCP, various strategies are applied to deter, prevent and react to any (potential) attack by adversaries [178-180].

Figure 5.1: Cloud Security Vulnerabilities [157]

Several security mechanisms are employed in order to minimise the impact of any security incidence and thereby, help to mitigate the adverse effects of a security attack. Some of these security mechanisms may include digital signatures, hashing, as well as virtualisation and data security [178].

Digital signature refers to any mechanism adapted to ensure message authenticity. Asymmetric cryptography is one of the simple examples of digital signature that uses a combination of public-key shared by sender and receiver's private key to authenticate the source of the message and validate the receiver's identity. Hashing, on the other hand, represents data in single unchangeable form converting the plain text into fixed-size hash code. Hashing ensures every single message has a distinct hash code that is unique to that message. Hashing ensures that the properties of data are the same, and when changed, can tell whether it has been compromised. Likewise, virtualisation security comprises of mechanisms specifically developed to ensure hypervisors or any other virtualisation technology provides secure isolation of VMs, including the shared OS features such as shared libraries. Frameworks such as closed box execution environment, secure live migration environment, hypervisor integrity through prevention of any changes
during runtime [179, 181]. Data integrity, on the other hand, is an approach that focuses more on protecting data visibility by masking its content via encryption. Encryption is the process of converting a plain text into cipher text in order to protect the data against sniffing at the data store and when in transit. In this method, different approaches of encrypting and decrypting data exist with various implementations using sophisticated algorithms [171, 182]. While the algorithms’ primary purpose is similar, they vary in terms of execution time (performance), reliability and applicability. Some algorithms are faster in one data type but slower on the other. Different comparisons of encryption algorithm’s performance are carried out in the literature (c.f., [183-185]).

Securing CCP through either controlling user access or masking the data through encryption helps in preserving its integrity. However, these mechanisms add overhead in system resources, thereby affecting the QoS defined in the SLA such as performance. As the process consumes system resource (such as CPU time), the effect, especially with stronger algorithms that are usually implemented using a longer encryption key on system performance, could result in significant breach of the SLA. In order to address this issue, a compromise may require that would improve one aspect and give away the other. An optimal trade-off will lead to not only a secure system, but also an acceptable level of performance.

In order to attain the highlighted level of optimal performance vs security trade-offs, an analysis is required that will examine the approaches of trade-off and the applicability in a complex environment such as the cloud. The analysis should first identify as appropriate the metrics used in evaluating computing system performance and security and then model and evaluate the optimal trade-offs between these two QoS components that are vital to SLA for virtualisation in CCPs.

### 5.3 Security Metrics

Quantifying security and defining its metrics has long been studied [119] mostly in the area of dependability and reliability analysis. Metrics assigned are mainly centred on the applicability of the security area. In their detailed survey on security metrics, Pendleton et al. [186] propose four main categories of system security metrics mainly centred on the measurement of the dynamics of security issues. The
metrics are the severity of the threats, the power of defence mechanisms, situation awareness and the vulnerability levels. However, these groups of metrics are further expanded to show the low-level metrics used to measure the security level of the system. Another intuitive approach to quantify security is expressing it in the concept of dependability, which is represented by either of the two values of work or fails. This is analogous to secure or insecure and the transition between both states as a probability by a stochastic process. This chapter assumes the system exist in either secure, insecure or recovering state as outlined by Wolter (c.f., [130]). Wolters model-based their choice on the reliability analysis with parameters of mean Time-Between-Incidents (TBI), mean Time-To-Incident-Discovery (TTID), mean Time-To-Incident-Recovery (TTIR), and mean Time-Between-Detection-and-Recovery (TBDR). The corresponding security parameters are highlighted in Figure 5.2.

![Diagram](image.png)

**Figure 5.2: Security Metric by analogy with dependability metric [131]**

The security detection control model (SDCM) can assign parameters of mean TBSI as the rate of an Attack transition, mean TTSID as the rate of Detect transition and mean TBSDR as Recover transition.

### 5.4 Performance Metrics

Measuring performance and assigning value to gauge its impact is determined using different indicators depending on the circumstance and interest of the enterprise. At each performance evaluation, given criteria and metrics are chosen mainly from the list of system service. While performance is mainly about how fast a
particular task is executed, it is also, about how slow some tasks are executed. The
three metrics [25] relevant to the fast or slow execution time are productivity,
responsiveness and utilisation of a given task. The task ranges from retrieving data
from a disk drive, time taken to process data in memory and time it takes to transport
result to the file, among others. In this context, the performance metrics that can be
applied mainly in CCP performance can be the end-to-end delay for transmitting the
data, response time to user’s request, how busy a system is being utilised, and the
probability of request being lost among others. In this context of performance and
security trade-off model, the performance metrics chosen are the request loss
probability and utilisation. Moreover, the metrics can be categorised as external
evaluating from outside the system and internal measuring inside the system.

5.4.1 External

Mean System Response time: This is the average time taken from the total time
it takes from the minute a user request is sent to the time a reply is received. This
include, time at senders device, the time it takes on transit, and the time it takes to
process the request and send it back to the user [25, 26].

Mean System Throughput: Is the average measure of the number of messages
(the rate at which a user request is serviced) are transmitted at a given time
between two entities (nodes) [25].

5.4.2 Internal

Mean System Utilisation ($\rho$): The utilisation of a node is measured as the average
fraction of time a given device is busy processing a request before forwarding it to
the next node or send it back to the requesting user [33]. The other fraction is when
the device is idle and the percentage of busy plus the percentage of idle must be
equal to 100.

Request Loss Probability: A request loss probability describes the fraction of
messages (request) that was dropped as a result of either buffer capacity is full or
the processing (encryption) time is longer than usual which result in message time
to live expirations.
5.5 **Impact of Security on Performance**

In general, the impact is in two folds, that is encryption & decryption and authentication & authorisation:

a) **Encryption and Decryption**: Algorithms that implement the encryption and the security protocol cost the computing power time and effort in encrypting and/or decrypting the message from plain text to cipher text. The longer the encryption key, the harder to break (i.e. more secure), as well as longer to process. The computational effort spent in encrypting degrades the performance. In other words, the longer the encryption key, the higher the security of encrypted messages.

b) **Authentication and Authorisation**: This is one of the levels of checking access credentials (e.g. username and password) supplied by the user during the quest to gain access to the resource against the credentials in the system, and when matched, access is granted. The time it takes to authenticate depends on the network speed, query engine and location of the database. Many additional layers of authentication exist, ranging from checking senders browser certificate to the most widely used two-level authentication. The two-level authentication sends verification message (text messages, email, and through an app) back to the user to reaffirm that he/she is the one trying to gain access. The message contains some one-time-password that the user will enter and when matched, will gain access. Although the two factor (level) authentication adds a layer of security, it resulted in a long time to gain access to the resource.

5.6 **Combined Performance and Security Model in OpenStack CCP**

The following illustrative model was based on the schematic architecture of the combined performance and security trade-off model (CPSM) of OpenStack CCP described earlier (c.f., Chapters 3 and 4). The scenarios assumed is for the receiving, processing and returning user request for the provisioning of VM instance. However, this model is only concerned with the security processing component that encrypts the message before sending it back to the user. The Output node is responsible for packaging and sending the result back to the requesting user with the successful creation of VM or otherwise. The following diagram (c.f., Figure 5.3)
Combined Performance and Security Model in OpenStack CCP shows the additional component from the figure that contains both the encryption and the decryption part. Meanwhile, this model is limited to the encryption component, which encrypts and transmit the message back to the requesting user.

This model depicts the architecture and method proposed in [130] and apply it into the OpenStack CCP architecture. This is through zooming the Output node and splitting it into two sub-nodes that is an Encrypt and Transmit sub-nodes (c.f., Figure 5.3). The model also considers the source of the message from the previous node and forwarding it to the next node onward.

As described in the literature, one of the mechanisms for securing an information system, including CCP, is through access control and encryption of the data. On encrypting data, the longer the encryption key, the more secure the system is, and the more processing cycle is required to encrypt or decrypt it.

Figure 5.4 shows the CPSM using a GSPN with named places and transitions. In this context, the performance model component (c.f., Figure 5.4 left) can be evaluated by obtaining the performance metric such as the utilisation $\rho_{cc}$. Security part (c.f., Figure 5.4 right) likewise, can be evaluated using the probability of system in Secure state $PP_{cc}$.
In this context, the CPSM is considered as an optimisation problem by computing two CPSMs with maximisation and minimisation metrics. These are CPSM-Maximum (CPSM1), and CPSM-Minimum (CPSM2) [127]. These two CPSMs are defined as follows:

CPSM1 is defined as the sum of the probability of the system in a ‘Secure’ place plus the utilisation at the Transmit transition. In order to optimise the encryption time, the CPSM1 is maximised, namely

$$\text{max } CPSM1 = \max \{P(\text{Secure}) + \rho \rho (\text{Transmit})\}$$ (5.1)

Similarly, the CPSM2 define as the sum of the probability of the system in an ‘Insecure’ place plus the request loss probability (RLP) at QueueIn place. The QueueIn place is considered as a finite capacity buffer and whenever the SDCM fires frequently, the buffer can fill up, and the request that cannot be accommodated in the QueueIn will be dropped. This will logically increase the utilisation and throughput at the Transmit transition. In order to compliment the max optimisation, the CPSM2 is minimised, namely

$$\text{min } CPSM2 = \min \{P(\text{Insecure}) + RLP\}$$ (5.2)

By maximising the CPSM1, as well as minimising the CPSM2 is expected to point to the same place and hence as expected, will provide the optimal trade-off between the two QoS components.
5.6.1 \textbf{Performance Model Component}

The performance model component is represented by three nodes represented by the GSPN notations as Arrival, Encrypt and Transmit transitions. The arrival represents the packaged reply that is going to be sent back to the requesting user. The response represented by Arrival is placed in the QueueIn and subsequently fired into the Encrypt transition. As long as the inhibitor arc from the security sub-model is not activated, it will process (encrypt) the message and forward it to the QueueOut which eventually fire and forwards it to the Transmit transition for processing (sending) it to the next node. The Encrypt transition will continue to fire until when a message is sent from the security component telling it otherwise (e.g. to freeze the operation because of detecting an attack).

5.6.2 \textbf{Security Model Component}

Security detection control model (SDCM) is the model detailing the scenario of security handling when an attack happens. For illustration purpose, this research considered stages and the changes that happen, which triggers the firing of the token in the model. The model is presented using the GSPN, which comprises of three places (Secure, Insecure and Restore) as well as the transitions (Fail, Detect, and Recover). The initial step happens when the token is in a secure state and moving from one place to another through respective transitions. The model starts with a service time that is exponentially distributed, with a mean service rate of $\mu_F$, $\mu_D$ and $\mu_R$ representing Fail, Detect and Recover transitions, respectively. A token moves from one input place to the next output place at the end of each transition.

It is assumed that the system starts with a ‘Secure’ state (c.f., Figure 5.5 – Step 1) that signifies that it is enabling the Fail transition. The mean firing rate $\mu_F$, $\mu_D$ and $\mu_R$ represents Fail, Detect and Recover rates, respectively. When a security breach happens, the ‘Fail’ transition is fired, and the security state changes from ‘Secure’ to ‘Insecure’. In between the two successive security breaches is the mean inter-fail time, when a new breach happens, the ‘Fail’ transition fires with a mean rate $\mu_F$. A token is then taken from the ‘Secure’ place into the ‘Insecure’ state, which enables the ‘Detect’ transition (c.f., Figure 5.5 – Step 2).
Once the system notices a security incident, the Detect transition will fire, and the state will change from ‘Insecure’ to ‘Restore’. At the end of the time between two successive detects (inter-detect time), the security breach is detected, the ‘Detect’ transition fires with a mean rate $\mu_{DD}$ and the token is removed from ‘Restore’ place, thereby enabling the ‘Recover’ transition (c.f., Figure 5.5 – Step 3). Then, the system is recovered, which allow the ‘Recover’ transition to fire making the token move from ‘Restore’ state back to the ‘Secure’ state (c.f., Figure 5.5 – Step 4). At this point, the time between two successive recoveries is given by $\mu_{RR}$.

At the end of one cycle, the time a token spent at one place is recorded in the period under observation $T$. Hence, the probability of the time spent by the token is cumulatively determined with the observational period $T$. In addition, the probability that the SDCM is at each state (i.e., ‘Secure’, ‘Insecure’, and ‘Restore’) during period $T$ can be determined. Thus, at the end of $T$, the aggregate time spent at each state is divided by $T$ to obtain the proportion of time that the CCP is found at a random period to be in ‘Secure’, ‘Insecure’, and ‘Restore’ places.

### 5.6.3 CPSM Implementation using Möbius Petri Net Tool

The CPSM was implemented using a Mobius package. The package was configured, and input data were supplied using the illustrative data from Table 5.1. The simulation was executed, and results were recorded presented graphically.
5.6.3.2 Möbius Main Features

5.6.3.1 Möbius Petri Net Overview

Möbius is a software tool developed at the University of Illinois for modelling and analysis of the behaviour of complex systems. Initially developed for the modelling and evaluation of system dependability, availability, reliability and performance of computing and communication systems, its usage expanded more. Its usage includes discrete event simulation, performance modelling and evaluation, biochemical and gene sequencing analysis. Much academic research adopts and utilises the tool because of its loosely coupled architecture and ease of use [187, 188]. It supports hierarchical modelling approach, customisable system components as well as numerical solution techniques.

Modelling with Möbius allows for the combinations of different modelling approaches. For example, it enables the Replica/Join composition technique via its SPN equivalent objects known as Stochastic Activity Networks (SAN). This feature allows the implementation of a number of formalisms such as GSPN, and PEPA. SANs features enable a compact representation of systems with the ability to use stochastic processes such as PN which gives it more acceptance compare to other packages [189].

5.6.3.2 Möbius Main Features

In addition to the GSPN set of tuples (places, transitions, Input, Output and markings), Mobius has extensible features that allow designing, parameterising and specifying distributions, as well as executing the model. It enables users to obtain results in different formats, including the popular comma-separated values (CSV) that can be accessed using any spreadsheet packages. The following are some of Mobius features relevant to this thesis. The following features are the components of the project that system modelling requires for designing, parameterising, running and collecting the result.

Atomic Formalisms: This feature allows the user to create models either using a single primary model (Similar to main class in object-oriented language) or use multiple sub models which are loosely coupled allowing adding and removing sub models with ease. Atomic model enables the use of editors such as SAN, PEPA,
Fault Tree, among others. SAN was used for the modelling of the CPSM presented in this chapter.

**Composed Formalisms:** This feature allows the user to compose models from previous models on top of the Atomic model in the form of sub models. This feature is mainly used when modelling large or complex systems that require breaking it down into small sub models. Example of this formalism is the use of Action Synchroniser feature or Replica/Join to replicate and merge sub models in tree-like hierarchical architecture.

**Reward Formalisms:** This feature measures information about the system being modelled. This includes the definition of variables (e.g. performance metrics) with the ability to input the data either manually or extract from an external source such as text file. Each variable’s state, time and source of data can also be defined.

**Studies Feature:** Study feature allows the modeller to either design an experiment with graphical elements or manually input the data for the global variables such as those specified in the Reward Formalisms or import the data from an external source especially for a range of input values. This feature was also used in the CPSM with the input data used in this chapter.

**Solver Feature:** Solver is either simulator or numerical solver. The choice of solver depends on the user data, result type and especially the problem at hand. Simulator solver, for example, allows the modeller to specify many settings depending on how the simulation is to be executed. The settings include the simulation parameters (simulation name, number of runs, number of random seeds, among others), running the simulation, results settings including result data format, among others. This feature is essential in both numerical and simulation executions.

### 5.6.3.3 Modelling Using Mobius Package

Modelling can be achieved using either of the Atomic formalisms described above. SAN models, in particular, are stochastic extensions of Petri Net. In this context, the SAN formalism was employed, and relevant objects were used. The objects of the SAN model include four primitive objects analogous to the tuples of the GSPN (c.f., Figure 5.6). These are places, activities, input gate, and output gates corresponding to the places, transitions, input places and output places, respectively.
5.6.3.3 Modelling Using Mobius Package

Figure 5.6: Implemented CPSM using Mobius SAN

**Places** (denoted by circles) represents the state of the modelled system. Each place contains some tokens which represent the markings (either set of input marking or output markings).

**Activities** (denoted by a bar or upright rectangle) represent actions that take an amount of time to complete. They are of two types: timed and instantaneous. Timed activities have durations that impact the performance of the modelled system, such as a packet transmission. Instantaneous activities, on the other hand, represents actions that complete as soon as they are enabled in the system.

**Input gates** (denoted by a red triangle) controls the enabling of activities and define the marking changes that will occur when an activity completes.

**Output gates** (also denoted by a black triangle with flat side connected to the activity), similar to the input gates, defines the marking changes that will occur when activities complete but only with a single case.

Modelling a SAN requires one to use the graphical feature to sketch the system model by dragging and dropping the object and connecting them with a connecting line. A designed is usually followed by defining the global variables and creating them in the *Reward* feature if the system does not contain sub models. This can be followed by a *Study* feature, which allows the modeller to define whether data values are inputted manually or imported from an external source. Execution type (simulation or numerical evaluation) is then chosen and defining the attributes. Simulator studies, for example, requires the modeller to specify the simulation name, type and number of runs. This feature also allows the selection of either timed
simulation or incremental simulation enabling the user to define the stepping values. Once everything is set, a simulation button click will start execution, and the result can be collected via the output files such as the CSV for easy analysis.

5.6.3.4 Möbius Firing Rule

The dynamism of the input gate determines token movement is achieved using sets of input and output states. With its two-component, an input gate can enable function or input state. An input state function defines the rule of changing models state, while the function state enables the predicate. A predicate is a condition that must be satisfied in before a token is fired. A token is fired (enabling an activity) when for every connected input gate, the enabling predicate is true, and for each input arc, the number of tokens in the connected place is greater than or equals to the number of arcs [188].

5.7 Numerical Experimentations and Result

This subsection discusses the parameterisation and experimentation to illustrate the trade-off model. It presents the results and interpretation showing how security (longer key length) affects the performance of CCP and how optimal trade-off is attained. The ‘optimal’ trade-off here is the highest degree of compromise between performance and security. That is to demonstrate an acceptable level of compromise between these two QoS components for an acceptable level service delivery defined in the SLA. This model assumes the longer the security key, the longer it takes to encrypt the message, which will affect the performance of the CCP. An optimal trade-off (obtained by optimal key length) may be determined by assigning suitable metrics for the CPSM.

For illustration purpose, max (CPSM1) and min (CPSM2) are determined. CPSM1 is the value for the probability of the system in a Secure state and the utilisation of the Transmit transition that is $P(\text{Secure}) + \rho P(\text{Transmit})$. Likewise, the min (CPSM2) is the sum of probability of system in Insecure state and the RLP of the Encrypt transition that is $P(\text{Insecure}) + RLP(\text{Encrypt})$. The rationale behind augmenting these metrics is to find the optimal point on the curve for both CPSM1 and CPSM2. If the optimal value points to the same encryption time, then the model is evaluating the optimal combined performance and security as expected. Next
section provides details of parameters used and the experimentations with a discussion of the results.

### 5.7.1 Parameterisation and Experiments

Performance and security trade-off model is implemented by choosing and assigning numerical values and executing the representative experiment for the identification of optimal trade-off between performance and security. The architecture operates when a message arrives for encryption and then transmitting back to the requesting user. As most of the modelling approach, this work chooses a firing delay using both exponential (M-type) distribution and bursty (GE-type) distribution. As approximated in the previous chapter, GE-type was determined by H$_2$ with a higher value of tuning parameter (k). The parameters (c.f., Table 5.1) that handle the processing (encrypting) and transmitting the message will determine the performance measure while the security aspect is determined by the values assigned to the SDCM transitions. Specifically, the Encrypt and Fail vary, as the research is interested in the effect of security on performance. The study also assumes that the longer the encryption key, the more secure the system is and will take a longer time to process, thereby adding more performance overhead.

For illustration purpose, the parameters used are presented in Table 5.1 and were carefully chosen to show the impact of security on performance. The performance metrics are the utilisation of the Transmit transition and the request drop probability at the Encrypt transition, while the security metrics are represented by the probabilities of a request being in a secure or insecure state. As probability values are less than 1, the values for the performance used were also closer to the security in order to see the apparent effect of security on performance. This model’s problem mainly affects the Encrypt and Fail transitions as the time required encrypting the message and the time taken during a security breach to break the encryption key. These values were inputted into the Möbius tool in order to obtain analytical results and to visualise the impact of security on performance as well as determining the optimal trade-off.
5.7.2 Results and Interpretations

Table 5.1: Assigned values for the parameters of the Trade-off Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (time)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival</td>
<td>0.125</td>
<td>'Mean Inter-Arrival’ time</td>
</tr>
<tr>
<td>Encrypt</td>
<td>0.01 to 0.34 by 0.01</td>
<td>'Mean Encryption’ time</td>
</tr>
<tr>
<td>Transmit</td>
<td>0.1</td>
<td>'Mean Transmission’ time</td>
</tr>
<tr>
<td>Fail</td>
<td>1.25, 2.5, 5, 10, …, 1510</td>
<td>'Mean Security Inter-Fail' time (after 10, rises by 50)</td>
</tr>
<tr>
<td>Detect</td>
<td>12</td>
<td>'Mean Inter-Detect’ time</td>
</tr>
<tr>
<td>Recover</td>
<td>36</td>
<td>'Mean Inter-Recover’ time</td>
</tr>
<tr>
<td>Finite capacity</td>
<td>100</td>
<td>Finite capacity of QueueIn and QueueOut places</td>
</tr>
</tbody>
</table>

The experiment was simulated with 100 runs in order to collect the result and subsequent interpretation. The execution has less than a 5% confidence level, and the results were collected and plotted in the next section.

5.7.2 Results and Interpretations

The M-type arrival process of the max (CPSM1) = p(Secure) + Utilisation(Transmit) (c.f., (5.1)), is presented in Figure 5.7. It shows the throughput of the Transmit transition. The red curve stands for the utilisation, while the blue curve represents the security metric of the probability of the system in a secure state. As the red curve raises and continues to move towards the maximum value of 1, it signifies the system will continue to encrypt the message before forwarding it to the next transition because it is in a secure state. When encryption time is short, the message waiting time is short, and that indicates that there is a high possibility that an attack can compromise it. In addition, when the encryption time is high, and the process takes a longer time, the messages will saturate the buffer of the encryption node (Encrypt transition). Hence, an increase in overall processing time, which in turn affects the utilisation and throughput of the Transmit transition. In addition, for an encryption key length, the security level is maintained until when the key length is increased, which will affect the next node (Transmit transition) performance attributes such as utilisation and utilisation.
5.7.2 Results and Interpretations

Therefore, the short encryption key lengths make the utilisation of the transmitting node (Transmit transition) starts at a low level due to the small security key length. This will result in the firing of the ‘Attack’ transition and eventual recurrent enabling of the inhibitor arc (represented by diamond shape), thereby freezing the encryption process and denying the movement of a message to the Transmit transition more often. Likewise, when encryption key length increases, the firing of the ‘Attack’ transition and subsequent enabling of the inhibitor arc reduces, hence increasing the throughput and utilisation of the Encrypt transition.

Conversely, when then encryption key length is more than the threshold, the Encrypt transition will become a bottleneck, which results in a slow rate of departure and declining the throughput and utilisation of the Transmit node. The optimal value of the encryption time in this scenario max (CPSM1) is around 0.09 seconds.
5.7.2.1 Impact of Traffic Burstiness on CPSM

Figure 5.8 presents a min (CPSM2) with the performance metric RLP in 'QueueIn' place of the performance component as well as the probability of the CCP in 'Insecure' state, i.e. min (CPSM2) = RLP + p(Insecure) (c.f., (5.2)). Increase in key length led to rising encryption time, which leads to the build-up of messages in the 'QueueIn' place, and results in loss of messages affecting the performance (c.f., Figure 5.8). Therefore, the probability of an encryption key being compromise must be reduced, leading to the platform moving into the recovery mode. Hence, whenever the encryption is quick enough, and the corresponding key length is secure, then the encryption process is said to be at its optimum as described in section 5.7.1.

5.7.2.1 Impact of Traffic Burstiness on CPSM

The max (CPSM1) does not affect the CCP when the arrival process bursts using a family of H2 distribution or even GE-type distribution. Figure 5.9 presents the max (CPSM1) for different values processed at all the transitions in both performance and security sub-models with varying level of traffic burstiness. The figure clearly shows that when the value of tuning parameter k is less, it signifies that the interarrival times results in declining the rate of transfer of request to the 'QueueIn' place. As the traffic rate increases, the messages drop rate will increase thereby affecting the throughput and utilisation at the Transmit transition, which will cause the max (CPMS1) curves to shift down appropriately.
The increase in traffic burstiness, according to the computed result, did not affect the optimal value for both max (CPSM1) and min (CPSM2). More importantly, the optimal value at both max (CPSM1) and min (CPSM2) are at the same place (i.e. 0.09 sec) as expected.

Figure 5.10 depicts how increasing traffic burstiness affects request drop rates. Clearly, it shows that when arrival follows bursty flow at the QueueIn place, it results in a higher drop rate of messages entering the Encrypt transition.

![Figure 5.10: M-Type, H2(k=2,10,50,100≤GE) CPSM2](image)

### 5.8 Summary

This chapter presents a performance and security trade-off model of the OpenStack IaaS CCP architecture that was modelled and analysed in the previous Chapter. The architecture was zoomed, and one of the components (Output) node was considered for the model because it is the last node to package and return the reply back to the user. The model assumed the processing and replying user after requesting VM involves the process of encrypting the message before sending it. The process of encrypting and sending was modelled using performance and security trade-off model. The chapter also presents a choice of performance and security metrics based on the discussions of how security affects system performance. The combined performance and security trade-off model was optimised using both maximisation and minimisation as appropriate. Numerical experiments were conducted as well as parameterisation and implementation of the model using the Mobius tool. This chapter presents the model that computes the
impact of security on performance using the length of the encryption key. This will help the CCP solution architects and performance engineers during tuning of existing CCP and design of new one to consider evaluating trade-offs by analysing the impact of security on performance with both normal and peak workload.
6 CONCLUSIONS AND FUTURE WORK

This chapter presents a summary of this thesis and associated conclusions. It also makes suggestions for future work.

6.1 Conclusions

This thesis examined in Chapter 2, the body of literature and presented an elaborate background on the virtualisation technology, cloud computing and OpenStack IaaS CCP. A systematic mapping study was also conducted in order to identify cloud testing areas as well as the knowledge gap. An extract from the systematic mapping study was also presented, which pave the way for this research. Moreover, a critical literature review was carried out focusing on methodologies of performance modelling and analysis of related works in the area with CCPs and virtualisation.

Chapter 3 proposed an open QNM with FCFS scheduling discipline, Poisson external arrival process and exponential service times. A typical setup architectural topology for the deployment of OpenStack IaaS CCP was developed consisting of multi-node systems connected via a network. The topology was used to develop a novel open QNM at equilibrium with single and multiple servers, as appropriate. By means of Jackson’s theorem [97, 137], the open QNM was decomposed into an individual queueing stations, each of which was analysed in isolation. Closed-form expressions for key performance metrics, such as the mean number of requests and utilisation at each queueing station, were employed in numerical experimentations, as appropriate. The queueing stations of the open QNM represent the nodes that form the OpenStack IaaS CCP. The analysis of the QNM led to the prediction and identification of bottleneck stations. The outcomes of these experimentations can guide the evaluation of the performance characteristics of the individual nodes that host the OpenStack IaaS CCPs. The analytical result will also help CC solution architects and performance engineers during the design of new CCPs as well as the capacity planning and or tuning of the existing platforms to produce an optimistic M-type performance bound for an OpenStack IaaS CCP.

Chapter 4 presents a new and extended application-driven open QNM at equilibrium with L, \((L>1)\) single and/or multiple server queueing stations \(\{i, i = 1, 2,\)
..., $L$) and random routing under FCFS rule. The model adopted and slightly modified the CCP architecture of Chapter 3 and used an open GE-type QNM for the performance modelling and analysis for the efficient provision of HA virtualisation services to clients. Based on a generic ME product-form approximation (c.f., [24]), the open QNM was employed to decompose the model into individual GE/GE/$c_i$ ($c_i \geq 1$), $i = 1, 2, ..., L$ queueing stations with $c_i$ servers and GE-type external interarrival and service times, each of which can be analysed in isolation. Closed-Form expressions for key performance metrics of each queueing station were formulated, such as those for the utilisation and server (resource) utilisation, mean response time and the steady-state probability of the number of VM requests by clients. For illustrative purposes, numerical experimentations were carried out alongside the M-type, $H_2$–type and a GE type interarrival and service time distributions with the same first and second moments, as appropriate (c.f., Figures 4.11 – 4.18). These experiments produce both optimistic M-type and pessimistic GE-type analytic performance bounds. The QNM’s performance metrics and the prediction of related bottlenecks provide vital insights for the efficient capacity planning as well as the design and development of new CCPs.

Lastly, Chapter 5 presents a model for the analysis of performance and security in OpenStack IaaS CCP. The earlier architectural topology was used to identify two nodes that in theory, deals with the decryption of request that is going into the platform and the encrypting the response before sending it back to the client. The study looked at the messages going out, which are encrypted then transmitted to the user. The model of Wolters and Reinecke's [130] was adopted and modified to suit the CCP architecture. The employed security detection control model (SDCM) was based on activity diagrams (ADs) that were implemented using the Möbius tool. Suitable parameterisation and numerical experiments were conducted with encryption and transmission of the message. Whenever an encryption key is short, the SDCM predicts a security breach (an attack) with high probability. This implies that there is a transition from the Secure state of the SDCM to the Insecure state. When this occurs, it freezes the operation of encryption. In this context, the throughput of the transmitting (sending) transition, as well as the request loss probability at the encryption (Encrypt) transition, are the performance metrics of
SDCM while the probabilities of the SDCM being in the Secure and Insecure states are the security metrics. A parameterisation and numerical optimisation of combined performance and security metrics show how security influenced performance and the determination of optimal trade-offs between security and performance.

Overall this thesis presents an innovative quantitative approach for modelling and analysis of virtualisations in CCPs performance as well as the impact of security on systems and its components performance. This approach predicts the performance of the system at a design stage and for capacity planning as well as for the tuning and upgrading of a new CCP. Both regular and bursty arrival processes adopted shows optimistic best-case and pessimistic worst-case performance bounds. The later will be of more interest to the performance engineers and solution architects as knowing the worst and preparing for it, is good disaster and fault/failure management strategy that will make the system with higher availability. This will enhance trust and future revenue. This research also demonstrated the modelling and evaluation of the combined trade-off model of performance and security, which are two essential aspects of QoS defined in SLA.

### 6.2 Recommendations for Future Research

Each of the contributions of this thesis requires an extension that will provide further research. The following are some of the areas recommended and intended to pursue as direction for future research:

a) A detailed systematic literature review will extend the systematic mapping studies (SMS) earlier conducted with more emphasis on the performance modelling and analysis in the cloud computing ecosystem. An SLR guideline such as [65, 67, 69] can be employed to explore the techniques employed in performance evaluation of CCPs.

b) Application of multiclass arrival traffic flow for M-type as well as GE-type QNM with finite capacity and priority-based scheduling discipline can be adopted for similar performance modelling and analysis of service usage in IaaS and associated virtualisation. Priority and a finite capacity queue are closer to realistic implementation of the CCP architecture, and quantitative methods [22,
190] exist for an adequate evaluation base on multiclass (representing request for different service) as well as priority base queue.

c) Future research could also explore the performance modelling and analysis of other layers of CCP services such as PaaS [91] and SaaS with applications to more efficient software performance engineering and database services with enhanced concurrency control mechanisms.

d) Experimentation for the validation of the model using either CloudSim (or any available simulation) or configuring the OpenStack IaaS CCP on physical machines in order to conduct a measurement base validation of the model.

e) To apply a new quantitative technique that extends the trade-off to obtain the level of compliance of the SLA trade-off model with false-positive (i.e., falsely alarming an attack) and false-negative (i.e., a real attack that is not detected) capability thereby producing more accurate computation.

f) To use the identity, access control and management through authentication and authorisation services (such as Keystone in OpenStack [191]) and measurement-based validation to verify the trade-off model numerical results such as the impact of arrival process on the CPSM. This may include the determination of how performance and security influence the QoS as well as how it affects the SLA.

g) To employ machine learning techniques [141] for the evaluation of trade-off analysis as well as the prediction of the likelihood of an intruders’ attack based on existing historical attack data and the scheduling of intrusion prevention measures based on optimal performance vs security trade-offs.
REFERENCES


REFERENCES


References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


[115] B. M. Kashinath, Yunas; Frank, Ball, "Performance Comparison of a SDN Network between Cloud-Based and Locally Hosted SDN Controllers," in


REFERENCES

*Experimental Software Engineering Latin American Workshop (ESELAW'05), Brazil, 2005.*


REFERENCES


REFERENCES


REFERENCES


A. **Generalised Exponential (GE) Distribution Formalism**

This section provides the GE formalism that leads to the computational algorithm of the matching parameters of H2 to GE distribution adopted from [23]. The GE pdf is defined as (c.f., [23]):

\[
\varphi(t) = \frac{CC - 1}{CC^2 + 1} \psi_0(t) + \frac{CC}{(CC^2 + 1)^2} e^{-\frac{2\tau}{\tau^2+1}}, \ t \geq 0
\]  

where \( \frac{1}{\tau} \) is the mean value and \( CC \) is the coefficient of variation, is an 'extremal' member of two-phase models with the same first two moments and a pdf of the universal form:

\[
ii(t, kk) = a_1(kk) \tau_1(kk) e^{-\nu_1(kk)t} + a_2(kk) \tau_2(kk) e^{-\nu_2(kk)t} \ t 
\]

\[
\geq 0
\]

such that

\[
\frac{a_1(kk)}{\tau_1(kk)} + \frac{a_2(kk)}{\tau_2(kk)} = \frac{1}{\tau}, \ a_1(kk) + a_2(kk) = 1
\]

\[
(A3)
\]

and

\[
2\tau^2 \frac{a_1(kk)}{\tau_1(kk)^2} + \frac{a_2(kk)}{\tau_2(kk)^2} - 1 = CC^2
\]

\[
(A4)
\]

Also,

\[
\lim_{kk \to +\infty} ii(tt, kk) \to \varphi(t), tt \geq 0, \ kk \to +\infty \text{ } CC^2 > 1 \text{ or } -\infty \text{ } CC^2 < 1
\]

\[
(A5)
\]

where \( \frac{1}{\nu_1}(kk) \), \( \frac{1}{\nu_2}(kk) \) are the mean values of the two service phases respectively and \{k\} is a set of 'tuning' parameters. \{k\} can be used to generate a family of H2
distributions with parameters \( \{\tau, CC^2 > 1\} \), with phase-selection probabilities \( \{\alpha_1(kk) > 0, \alpha_2(kk) > 0\} \) and \( kk \in (1, +\infty) \), or 'special', general 'improper' \( h_2 \) models with parameters \( \{\tau, CC^2 < 1\} \) with phase-selection indexes and satisfying the relations

\[
\tau_1(kk) = kk\alpha_1(kk)\tau
\]

\[
\tau_2(kk) = \frac{kk\alpha_2(kk)\tau}{kk - 1}
\]

Note that for \( CC^2 > 1 \), the GE model (AI) can be interpreted as a limiting case of an \( H_2 \) distribution where one of the two phases has zero service time. Moreover, the underlying counting (e.g. arrival) process of a GE inter-event time is equivalent to a compound Poisson process (CPP) with parameter \( = \frac{2v}{1+\alpha^2} \) and geometrically distributed batches with mean \( = 1 + \frac{\alpha^2}{2} \) and SCV \( = (\alpha^2 - 1)/(\alpha^2 + 1) \). For \( \alpha^2 < 1 \), the 'special' \( h_2 \) model of the form (A2) may produce, in some instances, an 'improper' distribution when the probability distribution function

\[
FF(xx, kk) = \sum_{x=0}^{\infty} \text{i}(tt, kk)aat, xx \geq 0
\]

produces negative values for a small range of \( x \) values close to zero, especially for \( CC^2 < 0.5 \) and \( kk \to -\infty \). For \( CC^2 = 0.5 \), the \( h_2 \) model is identical to a two-phase Erlang (E2) distribution at \( kk \to 0 \) (0- indicates a negative value close to zero, e.g. -0.001) and is very close to a 'proper' classical Hypoexponential-2 distribution for \( 0.5 CC^2 < 1 \) at \( kk \to 0 \). Although the GE model of (AI) is improper for \( CC^2 < 1 \), since

\[
\lim_{kk \to -\infty} FF(0,kk) = \frac{(CC^2 - 1)}{(CC^2 + 1)} < 0,
\]

Nevertheless, it represents, in general, a handy and robust overall approximation of general distributions (c.f., [23, 162, 163]). The utility of other improper two-phase distributions with \( CC^2 < 1 \) in systems performance modelling has also been pointed out (c.f., [192]). However, note that closed-form GE-type solutions of isolated queues and networks are simpler to derive owing to the pseudo-memoryless properties of the GE distributional model.
The following algorithm generates different two-stage pdfs of the form (Al) with the same first two moments.

**Algorithm A.1**

Given \( \tau > 1 \) and \( CC^2 > 1 \), this algorithm will produce the parameters \( \tau_1(kk) \) and \( \tau_2(kk) \) for an \( H_2 \) or \( h_2 \) random variable \( Y \).

**BEGIN**

**Step 1**: Choose a real value for the two-stage tuning parameter \( \{kk\} \),

if \( CC^2 > 1 \), then set \( kk \) such that \( kk \in (1, +\infty) \) otherwise \( kk \in (-\infty, 0) \);

**Step 2**: Calculate \( \alpha \alpha_1(kk) \) and \( \alpha \alpha_2(kk) \),

Set

\[
\omega_1(kk) = \begin{cases} 
\frac{AA + BB}{CC^2 + 1} & \text{if } CC^2 > 1, \\
\frac{AA - BB}{CC^2 + 1} & \text{if } CC^2 < 1
\end{cases}
\]  

(A10)

\[
\alpha \alpha_2(kk) = 1 - \alpha \alpha_1(kk),
\]  

(A11)

\[
AA = \frac{CC^2 - 1}{2} - \frac{2}{kk'}
\]  

(A12)

\[
BB = \frac{1}{2} CC^2 - 1 + \frac{8(1-C C^2)}{kk} + \frac{8(1-C C^2)}{kk^2}^{\frac{1}{2}}
\]  

(A13)

**Step 3**: Calculate \( \tau_1(kk) \) and \( \tau_2(kk) \) using (A6) and (A7);

**Step 4**: Produce \( ii(tt, kk) \) given at (A2);

**END**
B. Approximate Traffic Flow Analysis of a Stable GE-type Open QNM with General Topology

This section provides a GE distribution traffic flow analysis of a stable open QNM using generic topology as follows:

Consider an open GE-type QNM with general topology and \( L \) \((L>1)\) queueing stations, subject to the ME product-form approximation (c.f., [22]).

\[
pp(m_1, m_2, \ldots, m_L) = \prod_{i=1}^{L} pp_i(n_{ii}) \tag{B1}
\]

where \( pp(m_1, m_2, \ldots, m_L) \) is the joint state probability of having \( m_i, i = 1, 2, \ldots, L \) requests at queueing station and \( pp_i(n_{ii}) \) is the corresponding ME marginal state probability. In this context, the ME solution \( pp_i(n_{ii}) \), in conjunction with GE-type flow approximation formulae, associated with the departure, splitting and merging processes, can be used as cost-effective building blocks within an iterative queue-by-queue decomposition algorithm especially for complex QNMs with arbitrary configuration and blocking (c.f., [22]).

a) The Departure Process

In a stable GE-type QNM, for each queueing station \( i \) \((i = 1, 2, \ldots, L)\), the departure process comprises from the mean departure rate, \( \lambda_{di} \) and the SCV of the interdeparture times, \( C_{di}^2 \) \((i = 1, 2, \ldots, L)\) (c.f., Figure B.1).

![Figure B.1](image-url)

Figure B.1 A parameterised diagram of a departure process of heavy traffic approximation stable GE/GE/c \((c\geq1)\) queueing station with the associated interarrival times service times and interdeparture times with parameters \((\mu_i, c_i^2)\), \((\sigma_{\mu_i}, c_i^2)\) and \((\lambda_{di}, c_{di}^2)\), respectively.
In this context, each queueing station \( i \) \((i = 1, 2, \ldots, L)\) can be credibly represented by a stable ‘virtual’ \( GE/GE/c/FCFS \) \((cc \geq 1)\) queue \( ii \) with infinite capacity, FCFS service discipline and GE-type interarrival and service times with parameters \((\lambda_{ii}, CC^2)\) and \((cC\mu_{ii}, CC^2)\) – a heavy traffic flow approximation, respectively (c.f., [24]). Consequently, using GE-type probabilistic arguments (c.f., [24]), the mean departure rate, \( \lambda_{ddi} \) and SCV of the interdeparture times, \( CC^2 \) of the virtual \( GE/GE/c/FCFS \) queue \( i \) \((i = 1, 2, \ldots, L)\) can be determined by:

\[
\lambda_{ddi} = \lambda_{i}, (ii = 1, 2, \ldots, LL) \tag{B2}
\]

\[
CC^2_{ddi} = \rho_{ii} (1 - \rho_{ii}) + CC^2_{aii} (1 - \rho_{ii}) + \rho_{ii}^2 CC^2_{aii} (ii = 1, 2, \ldots, LL) \tag{B3}
\]

for \( i = 1, 2, \ldots, L \) where the server utilisation \( \rho_{ii} \), \((ii = 1, 2, \ldots, LL)\) is given by:

\[
\rho_{ii} = \begin{cases} 
\frac{\lambda_{ii}}{cC\mu_{ii}} & cc = 1 \\
\frac{\lambda_{ii}}{cC\mu_{ii}} & cc > 1
\end{cases}
\]

b) The Splitting of the Departure Process

In a typical GE-type QNM, a stream of traffic can split into two or more sub-streams (c.f., Figure B.1).

\[\lambda_{di}, CC^2_{di}\]

\[p_{io}, \lambda_{io}, CC^2_{dio}, p_{ii}, \lambda_{ii}, CC^2_{dii}, \ldots, p_{il}, \lambda_{il}, CC^2_{dil}\]

\[\text{Splitting Process}\]

Figure B.2 A parameterised diagram of the splitting stream with mean departure rate, \( \lambda_{ddii} \) and the SCV of the interdeparture times, \( CC^2_{ddii} \) stream and resulting effective splitting streams each with its associated transition probability \( p_{ii}, ii = 1, 2, \ldots, L; jj = 0, 1, \ldots, L \) with splitting departure rates \( \lambda_{ddii} \) and SCV of interdeparture times, \( CC^2_{ddii} \).

The mean splitting departure rate, \( \lambda_{ddii} \) and SCV of the interdeparture times, \( CC^2_{ddii} \) (c.f., [24]) are determined by
for $ii = 1, 2, ... , LL; jj = 0, 1, ... , LL$

c) The Merging Process of the External Arrival and Internal Departure Streams

In a stable open GE-type QNM, two or more arriving streams at a queueing station $i$ ($i = 1, 2, ..., L$) may be a resulting merger of external arrivals and internal GE-type splitting traffic flows (c.f., [22, 24] with overall mean arrival rate $\lambda$ and SCV of interarrival times, $C_2^i$, $i = 1, 2, ... , L$ (c.f., Figure B.2):

\begin{equation}
\lambda_{\text{dii}} = \sum_{i=1}^{L} \lambda_{i} p_{i} \tag{B6}
\end{equation}

\begin{equation}
\frac{\lambda_{\text{dii}}}{C_2^i} = 1 - pp_{i} + pp_{i} C_2^i \tag{B5}
\end{equation}

By using GE-type probabilistic arguments, it has been shown (c.f., [22, 24]) that mean arrival rate $\lambda_{ii}$ and SCV of interarrival times, $CC_2^i$, $i = 1, 2, ... , L$ of the merging process are given by:

\begin{equation}
\lambda_{\text{di}} = \sum_{i=0}^{\infty} \frac{\lambda_{ii} pp_{i}}{CC_2^i + 1} \tag{B7}
\end{equation}
C. \( H_2 \) to GE Matching Data

This section provides data generated by matching analysing the input values of the schematic diagram (c.f., Figure 4.6). The values were obtained by computation of the algorithm (c.f., Appendix A), which were used to obtain the results of both chapter 4 and part of chapter 5.

Table C.1: Table of values (k = 10) used to Match \( H_2 \) to GE for GE-type analysis of OpenStack IaaS CCP

<table>
<thead>
<tr>
<th>H2</th>
<th>GE</th>
<th>Firewall</th>
<th>Controller</th>
<th>Compute</th>
<th>Storage</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \nu \nu )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>( C )</td>
<td>105</td>
<td>125</td>
<td>130</td>
<td>120</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>( k \ k )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>51.8</td>
<td>61.8</td>
<td>64.3</td>
<td>59.3</td>
<td>56.8</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>52.2195</td>
<td>62.2196</td>
<td>64.7196</td>
<td>59.7196</td>
<td>57.2196</td>
<td></td>
</tr>
<tr>
<td>( p ) ( \alpha_1(k) )</td>
<td>0.9813</td>
<td>0.9843</td>
<td>0.9849</td>
<td>0.9836</td>
<td>0.9829</td>
<td></td>
</tr>
<tr>
<td>1-( p ) ( \alpha_2(k) )</td>
<td>0.0187</td>
<td>0.0157</td>
<td>0.0151</td>
<td>0.0164</td>
<td>0.0171</td>
<td></td>
</tr>
<tr>
<td>( \lambda_1 ) ( v_1(k) )</td>
<td>98.1316</td>
<td>98.4283</td>
<td>98.4883</td>
<td>98.3633</td>
<td>98.2927</td>
<td></td>
</tr>
<tr>
<td>( \lambda_2 ) ( v_2(k) )</td>
<td>0.2076</td>
<td>0.1746</td>
<td>0.168</td>
<td>0.1819</td>
<td>0.1897</td>
<td></td>
</tr>
</tbody>
</table>

Table C.2: Table of values (k = 50) used to Match \( H_2 \) to GE for GE-type analysis of OpenStack IaaS CCP

<table>
<thead>
<tr>
<th>H2</th>
<th>GE</th>
<th>Firewall</th>
<th>Controller</th>
<th>Compute</th>
<th>Storage</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \nu \nu )</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>( C )</td>
<td>105</td>
<td>125</td>
<td>130</td>
<td>120</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>( k \ k )</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>A</td>
<td>51.96</td>
<td>61.96</td>
<td>64.46</td>
<td>59.46</td>
<td>56.96</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>52.0408</td>
<td>62.0408</td>
<td>64.5408</td>
<td>59.5408</td>
<td>57.0408</td>
<td></td>
</tr>
<tr>
<td>( p ) ( \alpha_1(k) )</td>
<td>0.9811</td>
<td>0.9841</td>
<td>0.9847</td>
<td>0.9835</td>
<td>0.9828</td>
<td></td>
</tr>
<tr>
<td>1-( p ) ( \alpha_2(k) )</td>
<td>0.0189</td>
<td>0.0159</td>
<td>0.0153</td>
<td>0.0165</td>
<td>0.0172</td>
<td></td>
</tr>
<tr>
<td>( \lambda_1 ) ( v_1(k) )</td>
<td>490.5697</td>
<td>492.0666</td>
<td>492.3694</td>
<td>491.7388</td>
<td>491.3827</td>
<td></td>
</tr>
<tr>
<td>( \lambda_2 ) ( v_2(k) )</td>
<td>0.1925</td>
<td>0.1619</td>
<td>0.1557</td>
<td>0.1686</td>
<td>0.1759</td>
<td></td>
</tr>
</tbody>
</table>
Table C.3: Table of values (k = 100) used to Match H₂ to GE for GE-type analysis of OpenStack IaaS CCP

<table>
<thead>
<tr>
<th>H2 GE</th>
<th>Firewall</th>
<th>Controller</th>
<th>Compute</th>
<th>Storage</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>vv:</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>C:</td>
<td>105</td>
<td>125</td>
<td>130</td>
<td>120</td>
<td>115</td>
</tr>
<tr>
<td>kk:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>A:</td>
<td>51.98</td>
<td>61.98</td>
<td>64.48</td>
<td>59.48</td>
<td>56.9800</td>
</tr>
<tr>
<td>B:</td>
<td>52.0202</td>
<td>62.0202</td>
<td>64.5202</td>
<td>59.5202</td>
<td>57.0202</td>
</tr>
<tr>
<td>p α₁(k):</td>
<td>0.9811</td>
<td>0.9841</td>
<td>0.9847</td>
<td>0.9835</td>
<td>0.9828</td>
</tr>
<tr>
<td>1-p α₂(k):</td>
<td>0.0189</td>
<td>0.0159</td>
<td>0.0153</td>
<td>0.0165</td>
<td>0.0172</td>
</tr>
<tr>
<td>λ₁ v₁(k):</td>
<td>981.1339</td>
<td>984.1285</td>
<td>984.7343</td>
<td>983.4727</td>
<td>982.7603</td>
</tr>
<tr>
<td>λ₂ v₂(k):</td>
<td>0.1906</td>
<td>0.1603</td>
<td>0.1542</td>
<td>0.1669</td>
<td>0.1741</td>
</tr>
</tbody>
</table>
D. Primary Study Articles for Systematic Mapping Studies (SMS)


PS-2 L. S. Yu, Shuang;Zhao, Jing;Zhao, Wenbo;Luo, Shan;Fang, Qing;Tung, Frank;Liu, Alice Ying;Zhu, Jun;Su, Hui, "Performing unit testing based on testing as a service (TaaS) approach," Journal of Harbin Institute of Technology (New Series), vol. 15, pp. 207-212, 2008.


APPENDICES


<table>
<thead>
<tr>
<th>Page</th>
<th>Author(s)</th>
<th>Title</th>
<th>Conference/Proceedings</th>
<th>Year</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS-86</td>
<td>B. M. Xiaoying, Li; Xiaofei, Huang; Wei-Tek, Tsai; Gao, J.</td>
<td>&quot;Vee@Cloud: The virtual test lab on the cloud,&quot;</td>
<td>Automation of Software Test (AST), 2013 8th International Workshop on</td>
<td>2013</td>
<td>15-18</td>
</tr>
<tr>
<td>PS-87</td>
<td>Z. C. Xiaoyun, Wu; Su, Xue</td>
<td>&quot;Petri Nets Based Test Case Selection Model for Service Composition in Cloud,&quot;</td>
<td>Digital Manufacturing and Automation (ICDMA), 2013 Fourth International Conference on</td>
<td>2013</td>
<td>914-917</td>
</tr>
<tr>
<td>PS-89</td>
<td>R. A. Asprilla, I.</td>
<td>&quot;Improving mobile device performance using cloudlets,&quot;</td>
<td>Central America and Panama Convention (CONCAPAN XXXIV), 2014 IEEE</td>
<td>2014</td>
<td>1-5</td>
</tr>
<tr>
<td>PS-93</td>
<td>O. D. Cico, Zamir</td>
<td>&quot;Performance and load testing of cloud vs. classic server platforms (Case study: Social network application),&quot;</td>
<td>Embedded Computing (MECO), 2014 3rd Mediterranean Conference on</td>
<td>2014</td>
<td>301-306</td>
</tr>
</tbody>
</table>
APPENDICES


PS-109 R. M. B. Llamas, Quentin;Elmsheuser, Johannes;Legger, Federica;Sciacca, Gianfranco;Sciaba, Andrea;Ster, Daniel Van Der, "Testing as a service with HammerCloud," in 20th International Conference on Computing in High Energy and Nuclear Physics, CHEP 2013, October 14, 2013 - October 18, 2013, Amsterdam, Netherlands, 2014.


PS-116 C. M. M. Prathibhan, A.;Venkatesh, N.;Sundarakantham, K., "An automated testing framework for testing Android mobile applications in the


PS-123 H. G.-H. Sheng-Jen, Luo; Shyan-Ming, Yuan; Hsiao-Wei, Chen, "A flexible public cloud based testing service for heterogeneous testing targets,"


APPENDICES


160


